Домашнее задание к уроку 8 - CI/CD

Подготовка

```
cd ~/.ssh
cat ~/.ssh/id_rsa_gitlab.pub
```

```
cd ~/.ssh
eval "(ssh-agent -s)"
eval $(ssh-agent)
ssh-add id_rsa_gitlab
```

```
cd ~/geekbrains
git push -u origin master
git status
echo "test" >> test.txt
git add .
git commit -m "Initial commit"
```

```
igor@ubuntu-server: ~/geekbrains - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/.ssh$ cd ~/geekbrains
igor@ubuntu-server:~/geekbrains$ git push -u origin master
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
igor@ubuntu-server:~/geekbrains$ git status
On branch master
nothing to commit, working tree clean
igor@ubuntu-server:~/geekbrains$ echo "test" >> test.txt
igor@ubuntu-server:~/geekbrains$ git add .
igor@ubuntu-server:~/geekbrains$ git commit -m "Initial commit"
[master da3b785] Initial commit
 1 file changed, 1 insertion(+)
igor@ubuntu-server:~/geekbrains$
```

git push origin master

```
igor@ubuntu-server: ~/cicd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/cicd$ git push origin master
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 3 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (7/7), 552 bytes | 276.00 KiB/s, done.
Total 7 (delta 1), reused 0 (delta 0)
remote:
remote: To create a merge request for master, visit:
remote:   https://gitlab.com/TolstikovIgor/geekbrains/-/merge_requests/new?merge_request%5Bsource_branch
%5D=master
remote:
To gitlab.com:TolstikovIgor/geekbrains.git
 * [new branch]      master -> master
igor@ubuntu-server:~/cicd$
```

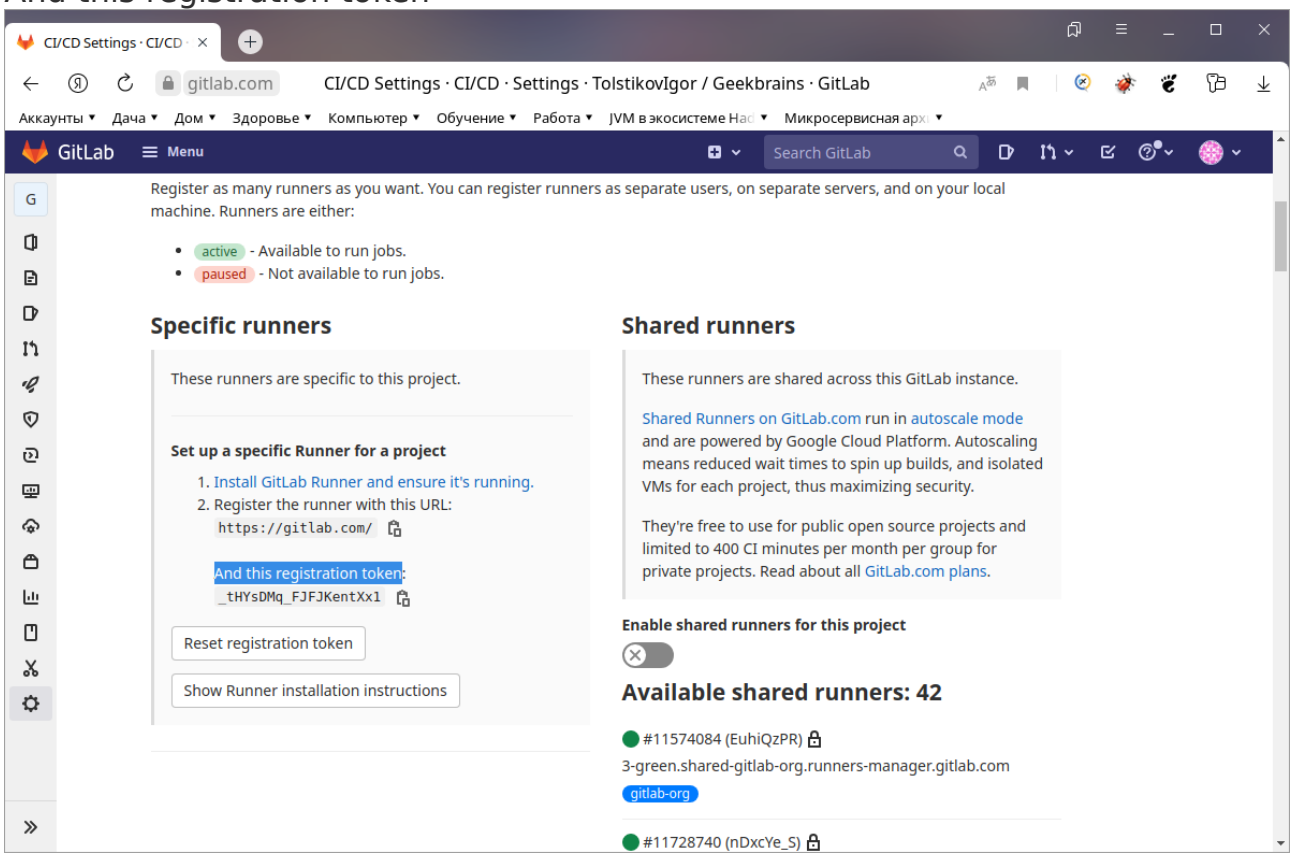
Настраиваем интеграцию GitLab и Kubernetes

kubectl create ns gitlab
kubectl get ns
kubectl delete ns gitlab
kubectl delete ns prod
kubectl delete ns stage
kubectl create ns gitlab
kubectl create ns stage
kubectl create ns prod

```
igor@ubuntu-server: ~/geekbrains - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/geekbrains$ kubectl create ns gitlab
Error from server (AlreadyExists): namespaces "gitlab" already exists
igor@ubuntu-server:~/geekbrains$ kubectl get ns
NAME              STATUS   AGE
default           Active   19d
gitlab            Active   4d17h
ingress-nginx     Active   11d
kube-node-lease   Active   19d
kube-public       Active   19d
kube-system       Active   19d
prod              Active   4d17h
redmine           Active   11d
stage             Active   4d17h
igor@ubuntu-server:~/geekbrains$ kubectl delete ns gitlab
namespace "gitlab" deleted
igor@ubuntu-server:~/geekbrains$ kubectl create ns gitlab
namespace/gitlab created
igor@ubuntu-server:~/geekbrains$ kubectl apply --namespace gitlab -f gitlab-runner/gitlab-runner.yaml
error: the path "gitlab-runner/gitlab-runner.yaml" does not exist
igor@ubuntu-server:~/geekbrains$ kubectl delete ns prod
namespace "prod" deleted
igor@ubuntu-server:~/geekbrains$ ^C
igor@ubuntu-server:~/geekbrains$ kubectl delete ns stage
namespace "stage" deleted
igor@ubuntu-server:~/geekbrains$ kubectl create ns stage
namespace/stage created
igor@ubuntu-server:~/geekbrains$ kubectl create ns prod
namespace/prod created
igor@ubuntu-server:~/geekbrains$
```

Shared runners

And this registration token



git add .
Vi gitlab-runner.yaml

igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd/gitlab-runner - Терминал

Файл Правка Вид Терминал Вкладки Справка

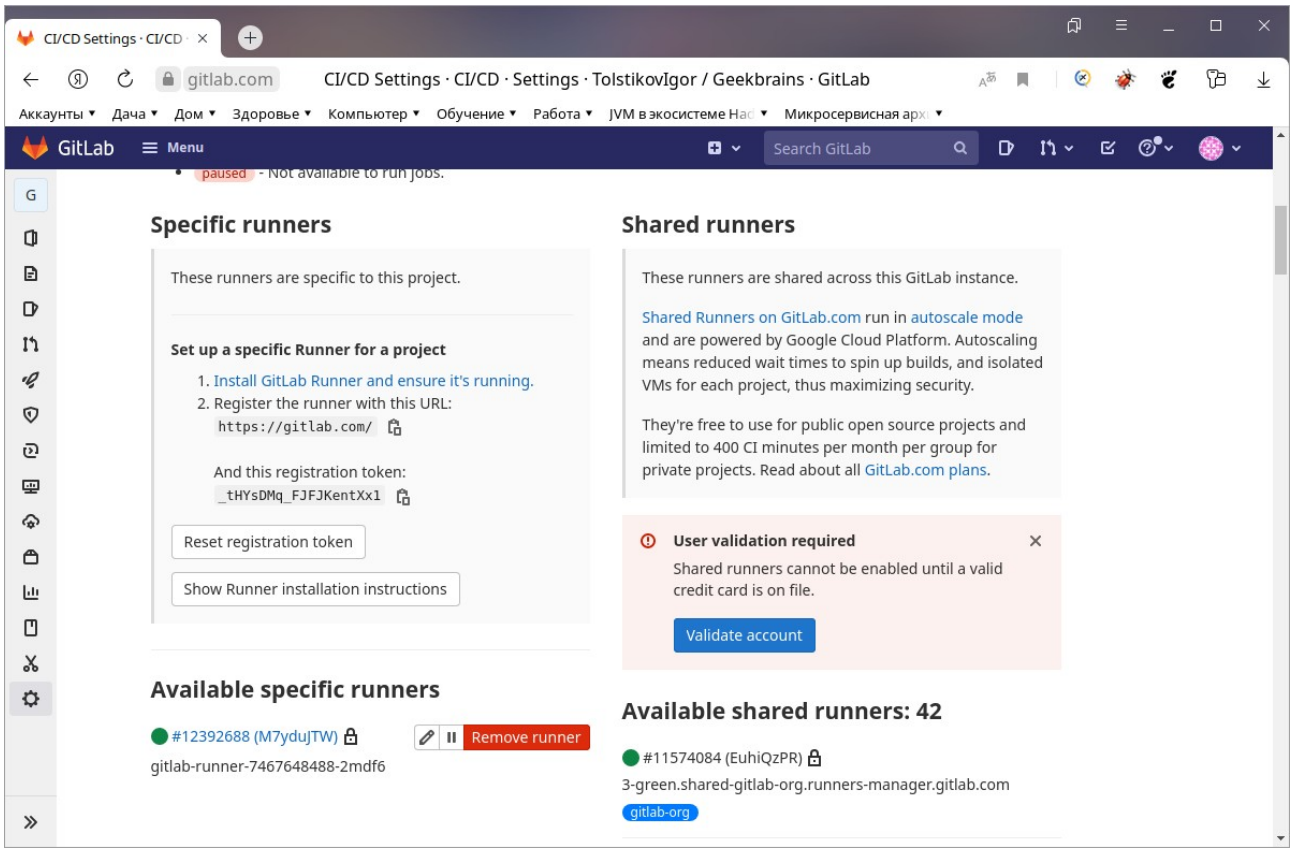
```yaml
---
apiVersion: v1
kind: ServiceAccount
metadata:
  name: gitlab-runner
  labels:
    app: gitlab-runner
---
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: gitlab-runner
  labels:
    app: gitlab-runner
rules:
- apiGroups: [""]
  resources: ["*"]
  verbs: ["*"]
---
apiVersion: 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: _tHYsDMq_FJFJKentXx1
  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=$(cat /secrets/gcs-access-id)
        # echo -e used to make private key multiline (in google json auth key private key is oneline wit
h \n)
        export CACHE_GCS_PRIVATE_KEY=$(echo -e $(cat /secrets/gcs-private-key))
      fi
    fi

    if [[ -f /secrets/runner-registration-token ]]; then
      export REGISTRATION_TOKEN=$(cat /secrets/runner-registration-token)
    fi

    if [[ -f /secrets/runner-token ]]; then
      export CI_SERVER_TOKEN=$(cat /secrets/runner-token)
    fi

    if ! sh /scripts/register-the-runner; then
      exit 1
    fi

    # Start the runner
    exec /entrypoint run --user=gitlab-runner \
      --working-directory=/home/gitlab-runner

  config.toml: |
    concurrent = 3
    check_interval = 30
    log_level = "info"
  configure: |
    set -e
    cp /init-secrets/* /secrets
  register-the-runner: |
    #!/bin/bash
    MAX_REGISTER_ATTEMPTS=30

    for i in $(seq 1 "${MAX_REGISTER_ATTEMPTS}"); do
      echo "Registration attempt ${i} of ${MAX_REGISTER_ATTEMPTS}"
      /entrypoint register \
        --non-interactive

      retval=$?

      if [ ${retval} = 0 ]; then
        break
      elif [ ${i} = ${MAX_REGISTER_ATTEMPTS} ]; then
        exit 1
      fi

      sleep 5
    done
:wq
```

Применяем манифесты для раннера

kubectl apply --namespace gitlab -f gitlab-runner.yaml

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd/gitlab-runner - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
app  gitlab-runner  README.md
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$ cd gitlab-runner
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ ls
gitlab-runner.yaml
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ git add .
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ vi gitlab-runner.yaml
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ kubectl apply --namespace gitlab -f gitlab-runner.yaml
serviceaccount/gitlab-runner created
role.rbac.authorization.k8s.io/gitlab-runner created
rolebinding.rbac.authorization.k8s.io/gitlab-runner created
secret/gitlab-runner created
configmap/gitlab-runner created
deployment.apps/gitlab-runner created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$
```

Обновляем страницу на GitLab, runner должен появиться в списке Available specific runners



Создаем авторизационные объекты, чтобы раннер мог деплоить в наши нэймспэйсы

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

kubectl create sa deploy --namespace prod

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

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd/gitlab-runner - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ kubectl create sa deploy --namespace stage
serviceaccount/deploy created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ kubectl create rolebinding deploy --serviceaccount stag
e:deploy --clusterrole edit --namespace stage
rolebinding.rbac.authorization.k8s.io/deploy created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ kubectl create sa deploy --namespace prod
serviceaccount/deploy created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ kubectl create rolebinding deploy --serviceaccount prod
:deploy --clusterrole edit --namespace prod
rolebinding.rbac.authorization.k8s.io/deploy created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$
```

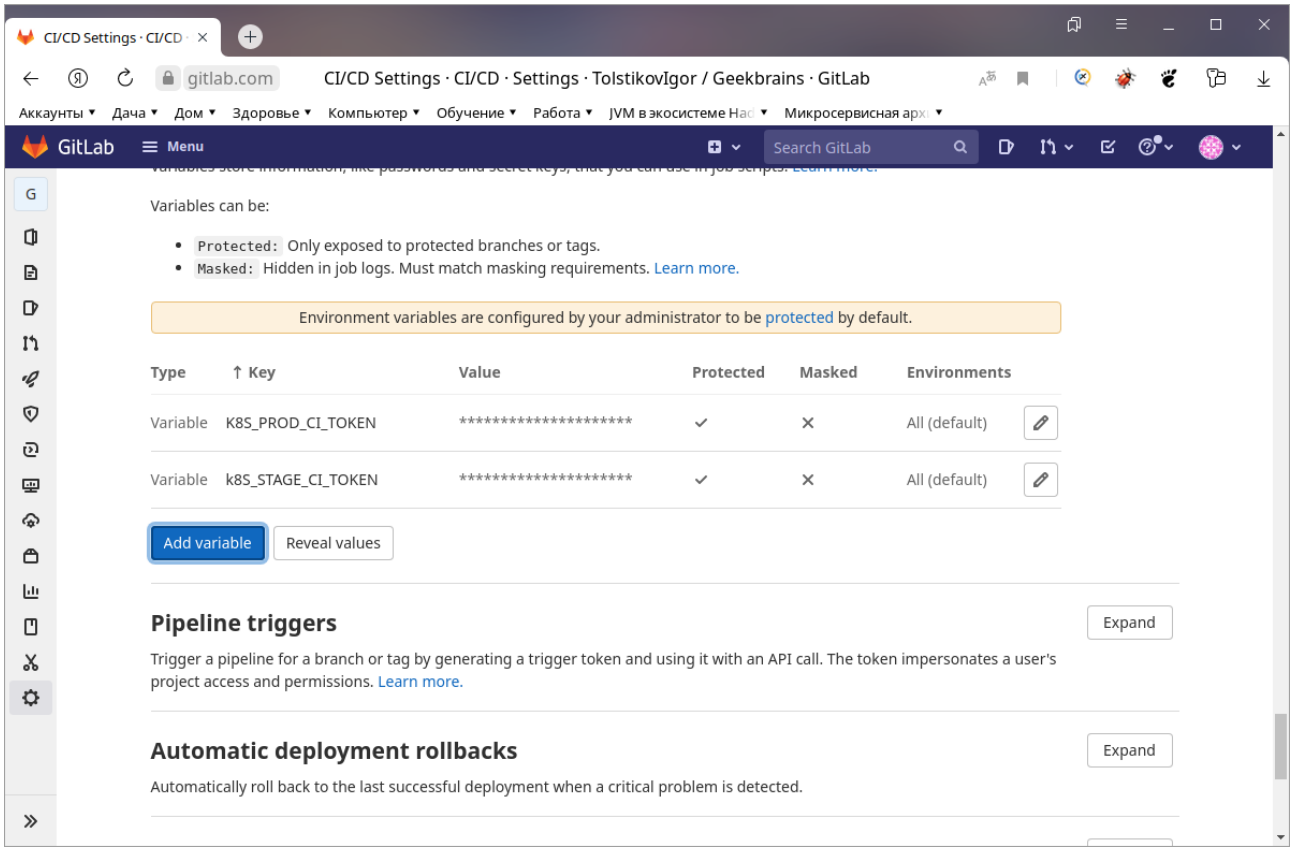
Получаем токены для деплоя в нэймспэйсы

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

ZXlKaGJHY2lPaUpTVXpJMU5pSXNJbXRwWkNJNklsSkxTMFI2YWpOWVh6UkVkalJqT0hReGVTMVhkRzUxU21oQmRtMXpTbWRCTVc1UVVWRlhZMngzYkVFaWZRLmV5SnBjM01pT2lKcmRXSmxjbTVsZEdWekwzTmxjblpwWTJWaFkyTnZkVzUwSWl3aWEzVmlaWEp1WlhSbGN5NXBieTl6WlhKMmFXTmxZV05qYjNWdWRDOXVZVzFsYzNCaFkyVWlPaUp6ZEdGblpTSXNJbXQxWW1WeWJtVjBaWE11YVc4dmMyVnlkbWxqWldGalkyOTFiblF2YzJWamNtVjBMbTVoYldVaU9pSmtaWEJzYjNrdGRHOXJaVzR0YW1vM05qa2lMQ0pyZFdKbGNtNWxkR1Z6TG1sdkwzTmxjblpwWTJWaFkyTnZkVzUwTDNObGNuWnBZMlV0WVdOamIzVnVkQzV1WVcxbElqb2laR1Z3Ykc5NUlpd2lhM1ZpWlhKdVpYUmxjeTVwYnk5elpYSjJhV05sWVdOamIzVnVkQzl6WlhKMmFXTmxMV0ZqWTI5MWJuUXVkV2xrSWpvaU9ERTJZV1ptWVRrdE9UZzRaQzAwTTJFekxUbGpZbVF0TUdNM1ptSmlNalk0TVdZeElpd2ljM1ZpSWpvaWMzbHpkR1Z0T25ObGNuWnBZMlZoWTJOdmRXNTBPbk4wWVdkbE9tUmxjR3h2ZVNKOS5oYU9iLWJseHpQbkFRSVlYa1BoVGZFcWNfb1JxTmY4Q3Q2TzM4Z2RUdF9ITU9aVFlPMmFDRWpQWktHMy1BWTJCdTYzOEpDanBIVDlDNk9QaTVJZWpQRGVjakRsZkcydjM4QTFjR2JNRjh3cExuSVZTeHFkSFRUZHNDel9QMHQydTQ4RHRaWWd2ZVhoamhpOWlva29GckZBM25NamtYRHQ2ZW1aWlJUN2VXWXh4djNQUjhNd2hETE5QanZ6ZWJPa21RSVBGbV9GWWNub1JmVkRyakZWY1dhWTdkT0FpWGdiSDVweUg4Y1AtLU1FMF9KZFBuRmJCMFRkQ3NOam5VNThPbHhFYW1GakhkTTZZdTc0U0d1YkpRdWQwSFdhYXN1MXRLV0s4ZXdNM04tQk01RXJJcnIyNnBOclBZajBQTXRtSUNqQm50bmE3SVJqR2M3STdFOWNkcHc=

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

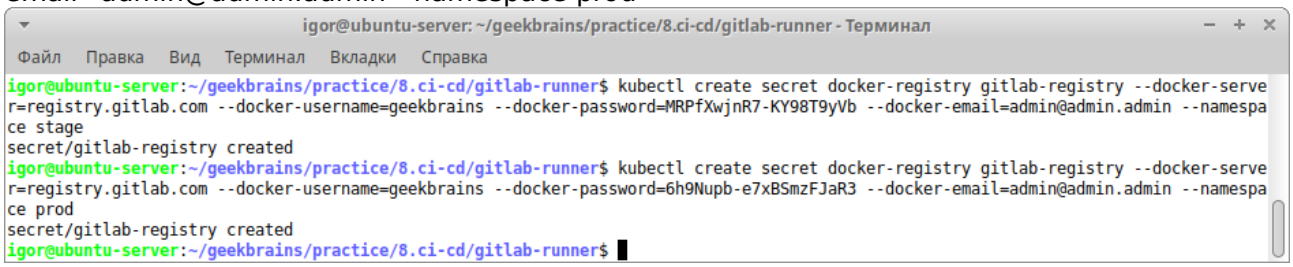
ZXlKaGJHY2lPaUpTVXpJMU5pSXNJbXRwWkNJNklsSkxTMFI2YWpOWVh6UkVkalJqT0hReGVTMVhkRzUxU21oQmRtMXpTbWRCTVc1UVVWRlhZMngzYkVFaWZRLmV5SnBjM01pT2lKcmRXSmxjbTVsZEdWekwzTmxjblpwWTJWaFkyTnZkVzUwSWl3aWEzVmlaWEp1WlhSbGN5NXBieTl6WlhKMmFXTmxZV05qYjNWdWRDOXVZVzFsYzNCaFkyVWlPaUp3Y205a0lpd2lhM1ZpWlhKdVpYUmxjeTVwYnk5elpYSjJhV05sWVdOamIzVnVkQzl6WldOeVpYUXVibUZ0WlNJNkltUmxjR3h2ZVMxMGIydGxiaTFtTW1od2VpSXNJbXQxWW1WeWJtVjBaWE11YVc4dmMyVnlkbWxqWldGalkyOTFiblF2YzJWeWRtbGpaUzFoWTJOdmRXNTBMbTVoYldVaU9pSmtaWEJzYjNraUxDSnJkV0psY201bGRHVnpMbWx2TDNObGNuWnBZMlZoWTJOdmRXNTBMM05sY25acFkyVXRZV05qYjNWdWRDNTFhV1FpT2lJMIpXRmxPRGxsWXkwNVpEbGtMVFF3T0dFdFlUazRaOaTFoTVdVM05HSm1Oamd3TnpVaUxDSnpkV0lpT2lKemVYTjBaVzA2YzJWeWRtbGpaV0ZqWTI5MWJuUTZjSEp2WkRwa1pYQnNiM2tpZlEuTDZzdVpic25mdi1DcG1NbWpOX3huSE82TXJIejAwaV9BSkFmdU1OMHMwMGtyaktKcDROT1dlTzNVQTJwT3VuTXlIcnZobXZYRWpKblAyYURxUjVCVU1JS2QzdFFENktRbUl4aTFHWTFCNDRVZ3c1OXdtUjJyNFEwb3ZGMmQ2T19FclY2Q3B5WVZ3aEh3djRVdFRxZ2dYR3UtQjYteUh5RC1UT1lGODlHQVBoZkpuNzZmNTUtQzdqT1FRQ3JaMEt1MmNrS2VYbU1pMnZWcTBjZkVoT0tOT3JJaVlvQlFOcVhlTkdndnVMX2Ftb25GaXZGUmZBWTF1NVBzSHdVWmE1cE8yNTQ0bjZNTV9XaDd5YTB6S01oWGp6SlNrVG9QUEJuNHg4Ri1lNEt6RElOcDdkMXF0ZjNIWUtHMmtJOWdRT1R0ajBMaGlDZmR0SGdJX0p0YllpeHBRigor
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$
```

## Variables
```



Создаем секреты для авторизации Kubernetes в Gitlab registry. При создании используем Token, созданный в **Settings -> Repository -> Deploy Tokens**.

geekbrains stage
MRPfXwjnR7-KY98T9yVb
geekbrains prod
6h9Nupb-e7xBSmzFJaR3

kubectl create secret docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=geekbrains --docker-password=MRPfXwjnR7-KY98T9yVb --docker-email=admin@admin.admin --namespace stage
kubectl create secret docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=geekbrains --docker-password=6h9Nupb-e7xBSmzFJaR3 --docker-email=admin@admin.admin --namespace prod



Патчим дефолтный сервис аккаунт для автоматического использование pull secret

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

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd/gitlab-runner - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ kubectl patch serviceaccount default -p '{"imagePullSecr
ets": [{"name": "gitlab-registry"}]}' -n stage
serviceaccount/default patched
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$ kubectl patch serviceaccount default -p '{"imagePullSecr
ets": [{"name": "gitlab-registry"}]}' -n prod
serviceaccount/default patched
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/gitlab-runner$
```

Запуск приложения

Создаем манифесты для БД в stage и prod

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

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$ kubectl apply --namespace stage -f app/kube/postgres/
secret/app created
service/database created
statefulset.apps/database created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$ kubectl apply --namespace prod -f app/kube/postgres/
secret/app created
service/database created
statefulset.apps/database created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$
```

Меняем хост в ингрессе приложения и применяем манифесты Для этого открываем app/kube/ingress.yaml Там ищем и вставляем вместо него stage

Далее применяем на stage

vi app/kube/ingress.yaml

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd - 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:
  rules:
    - host: postgres.stage.info
      http:
         paths:
           - path: /users
             pathType: Prefix
             backend:
                service:
                  name: prometheus
                  port:
                     number: 8080
~
:wq
```

kubectl apply --namespace stage -f app/kube

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$ vi app/kube/ingress.yaml
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$ kubectl apply --namespace stage -f app/kube
deployment.apps/geekbrains created
ingress.networking.k8s.io/geekbrains created
service/geekbrains created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$
```

Повторяем для прода. Открываем тот же файл ingress и вставляем вместо stage prod

Далее применяем на prod
vi app/kube/ingress.yaml

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd - Термин
Файл  Правка  Вид  Терминал  Вкладки  Справка
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: geekbrains
  annotations:
          kubernetes.io/ingress.class: nginx-external
spec:
  rules:
    - host: postgres.prod.info
      http:
         paths:
           - path: /users
             pathType: Prefix
             backend:
                service:
                  name: prometheus
                  port:
                     number: 8080
~
:wq
```

kubectl apply --namespace prod -f app/kube

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$ vi app/kube/ingress.yaml
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$ kubectl apply --namespace prod -f app/kube
deployment.apps/geekbrains created
ingress.networking.k8s.io/geekbrains created
service/geekbrains created
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$
```

Проверяем работу приложения

Мы развернули приложение, теперь убедимся, что оно работает. Наше приложение - это REST-API. Можно выполнять к нему запросы через curl. В примерах указан недействительный ip адрес - 1.1.1.1, нужно заменить его на EXTERNAL-IP сервиса ingres-controller (Load Balancer).

Записать информацию о клиенте в БД

curl 1.1.1.1/users -H "Host: stage" -X POST -d '{"name": "Vasiya", "age": 34, "city": "Vladivostok"}'

Получить список клиентов из БД

curl 1.1.1.1/users -H "Host: stage"

kubectl get svc -A

igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd - Терминал

Файл Правка Вид Терминал Вкладки Справка

```
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$ kubectl get svc -A
NAMESPACE       NAME                                 TYPE        CLUSTER-IP       EXTERNAL-IP   PORT(S)                      AGE
default         database                             ClusterIP   10.107.154.203   <none>        5432/TCP                     3d19h
default         geekbrains                           ClusterIP   10.109.211.170   <none>        8000/TCP                     3d19h
default         kubernetes                           ClusterIP   10.96.0.1        <none>        443/TCP                      19d
default         prometheus                           ClusterIP   10.105.48.84     <none>        80/TCP                       7d22h
ingress-nginx   ingress-nginx-controller             NodePort    10.97.230.241    <none>        80:31507/TCP,443:31952/TCP   11d
ingress-nginx   ingress-nginx-controller-admission   ClusterIP   10.109.110.129   <none>        443/TCP                      11d
kube-system     kube-dns                             ClusterIP   10.96.0.10       <none>        53/UDP,53/TCP,9153/TCP       19d
prod            database                             ClusterIP   10.110.9.145     <none>        5432/TCP                     45m
prod            geekbrains                           ClusterIP   10.108.39.26     <none>        8000/TCP                     36m
redmine         pg-service                           ClusterIP   10.109.52.102    <none>        5432/TCP                     11d
redmine         redmine-service                      ClusterIP   10.100.187.140   <none>        80/TCP                       11d
stage           database                             ClusterIP   10.99.97.253     <none>        5432/TCP                     45m
stage           geekbrains                           ClusterIP   10.111.112.52    <none>        8000/TCP                     38m
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd$
```

Вносим правки в kube/deployment.yaml - меняем image: nginx:1.12 на image: __IMAGE__

vi kube/deployment.yaml

igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd/app - Термин

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: image: __IMAGE__ # 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
```

Вносим правки в kube/ingress.yaml - меняем значение в host на __HOST__
vi kube/ingress.yaml

```
igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd/app
Файл  Правка  Вид  Терминал  Вкладки  Справка
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: geekbrains
  annotations:
          kubernetes.io/ingress.class: nginx-external
spec:
  rules:
    - host: __HOST__
      http:
         paths:
           - path: /users
             pathType: Prefix
             backend:
                service:
                  name: prometheus
                  port:
                     number: 8080
:wq
```

Вносим правки в .gitlab-ci.yml, для деплоя добавляем строки, шаг деплоя в .gitlab-ci.yml, чтобы изменять **IMAGE** на реальное имя образа и тег

меняем это

- kubectl set image deployment/$CI_PROJECT_NAME *=$CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID --namespace $CI_ENVIRONMENT_NAME

На это
- sed -i "s,__IMAGE__,$CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID,g" kube/deployment.yaml
- kubectl apply -f kube/ --namespace $CI_ENVIRONMENT_NAME

vi .gitlab-ci.yml

igor@ubuntu-server: ~/geekbrains/practice/8.ci-cd/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 $K8S_CI_TOKEN | base64 --decode)
    - kubectl config set-context ci --cluster=k8s --user=ci
    - kubectl config use-context ci
  script:
    - sed -i "s,__IMAGE__,$CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID,g" kube/dep
loyment.yaml
    - kubectl apply -f kube/ --namespace $CI_ENVIRONMENT_NAME
    - kubectl rollout status deployment/$CI_PROJECT_NAME --namespace $CI_ENVIRONMENT_NAME ||
 (kubectl rollout undo deployment/$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
```

git push -u origin master

```
igor@ubuntu-server: ~/.ssh - Терминал
Файл  Правка  Вид  Терминал  Вкладки  Справка
igor@ubuntu-server:~/geekbrains/practice/8.ci-cd/app$ git push -u origin master
git@github.com: Permission denied (publickey).
fatal: Could not read from remote repository.

Please make sure you have the correct access rights
and the repository exists.
```