

# CI/CD

В рамках данного задания нужно продолжить работу с CI приложения из лекции. То есть для начала выполнения этого домашнего задания необходимо проделать то, что показывалось в лекции. Все задание должно выполняться применительно к файлам в директории `practice/8.ci-cd/app`

1.

Переделайте шаг деплоя в CI/CD, который демонстрировался на лекции таким образом, чтобы при каждом прогоне шага `deploy` в кластер применялись манифесты приложения. При этом версия докер образа в деплойменте при апплее должна подменяться на ту, что была собрана в шаге `build`.

Для этого самым очевидным способом было бы воспользоваться утилитой `sed`.

⑩ Измените образ в деплойменте приложения (файл `kube/deployment.yaml`) на плейсхолдер.

Вот это

```
image: nginx:1.12 # это просто плейсхолдер
```

На это

```
image: __IMAGE__
```

2.

Измените шаг деплоя в `.gitlab-ci.yml`, чтобы изменять **IMAGE** на реальное имя образа и тег

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
```

Вторую строчку шага деплоя (которая отслеживает статус деплоя) оставьте без изменений.

3.

Попробуйте закоммитить свои изменения, запустить их в репозиторий (тот же, который вы создавали во время лекции на Gitlab.com) и посмотреть на выполнение CI в интерфейсе Gitlab.

Так как окружений у нас два (stage и prod), то помимо образа при апплее из CI нам также было бы хорошо подменять host в ingress.yaml. Попробуйте реализовать это по аналогии, подставляя в ингресс вместо плейсхолдера значение переменной `$CI_ENVIRONMENT_NAME`

4.

Так же попробуйте протестировать откат на предыдущую версию, при возникновении ошибки при деплое

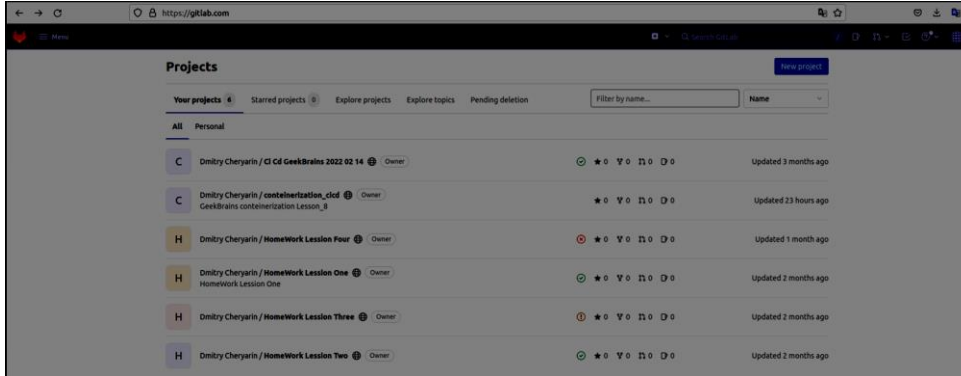
Для этого можно изменить значение переменной `DB_HOST` в `deployment.yaml` на какоенибудь несуществующее. Тогда при старте приложения оно не сможет найти БД и будет постоянно рестартовать. CI должен в течении `progressDeadlineSeconds: 300` и после этого запустить процедуру отката. При этом не должно возникать недоступности приложения, так как старая реплика должна продолжать работать, пока новая пытается стартовать.

0.

Поскольку для начала выполнения этого домашнего задания необходимо проделать то, что показывалось в лекции – выполняю все действия:

## Подготовка

- 10 Зарегистрируйте аккаунт GitLab



- 10 добавьте в настройках своего аккаунта на Gitlab.com свой публичный SSH ключ:

Сгенерировать SSH ключ можно командой

ssh-keygen

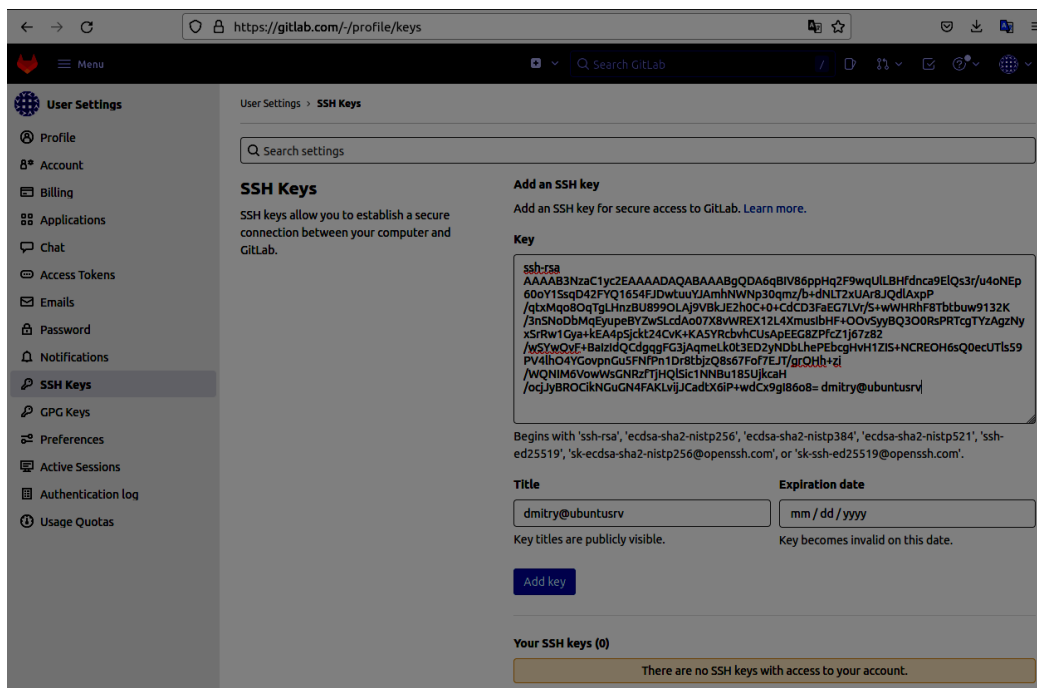
```
dmity@ubuntu:~/c4cd/c4cd-practice/1_containerization/lesson_00/ops$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/dmity/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/dmity/.ssh/id_rsa
Your public key has been saved in /home/dmity/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:sLn3tY3kAvLZnrUqn3ESbMrJfR7dq0yTkKoAI6q0mxU dmity@ubuntusrv
The key's randomart image is:
+---[RSA 3072]-----+
|
|  .   +
| . oE  o S +.
|.. o. .+. =o. . o
|... .. .o* =+.B.o .
|o + . . =.o#oB
|.+. . . +B+X..
+---[SHA256]-----+
dmity@ubuntu:~/c4cd/c4cd-practice/1_containerization/lesson_00/ops$
```

Чтобы вывести содержимое ключа выполните

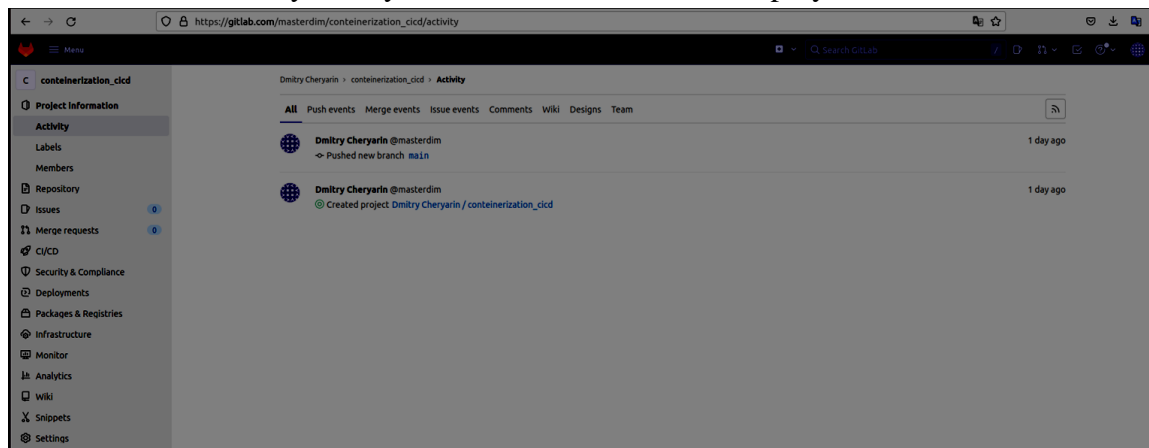
cat ~/.ssh/id\_rsa.pub

```
dmity@ubuntu:~/c4cd/c4cd-practice/1_containerization/lesson_00/ops$ cat ~/.ssh/id_rsa.pub
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAQDA6qBIV86ppHq2F9wqULLBHfdnca9ELQs3r/u4oNEp60oY1Ssq042FYQ1654FJDwtuuYJAm
hNWNp30qgz/b+dNLT2xUAR8JQdLaxpP/qtxMqo80qTgLNzBU8990LAj9VBkJE2h0C+0+CdCD3FaEG7Lvr/S+wWHRhF8Tbtbuw9132K/3nSNo
DbMqEyupeBYZwSLcdAo07X8vWREX12L4XmusIbHF+00vSyyBQ300RsPRTcgTYzAgzNyxSRw1Gya+kEA4pSjckt24CvK+KA5YRcbvhCUsApEE
G8ZPfcZ1j67z82/wSYwovF+BaiZIdQcdggqFG3jAqneLk0t3ED2yNdbLhePEbcgHvH1ZIS+NCRE0H6sQ0ecUTls59PV4lh04YgovpGu5FNFP
n1Dr8tbjzQ8s67Fof7EJT/gr0Hh+zt/WQNI6VowWsGNRzftJHQL5ic1NNBu185UjKcah/ocJyBROctkNGuGN4FAKLvIjJCadtX6tP+wdCx9
gI86o8= dmity@ubuntusrv
dmity@ubuntu:~/c4cd/c4cd-practice/1_containerization/lesson_00/ops$
```

В правом верхнем углу выбираем **Preferences -> SSH keys** и добавляем SSH ключ



10 Создайте новый проект с именем geekbrains. Если выберете другое имя проекта, в дальнейшем нужно будет также изменить имя Deployment.



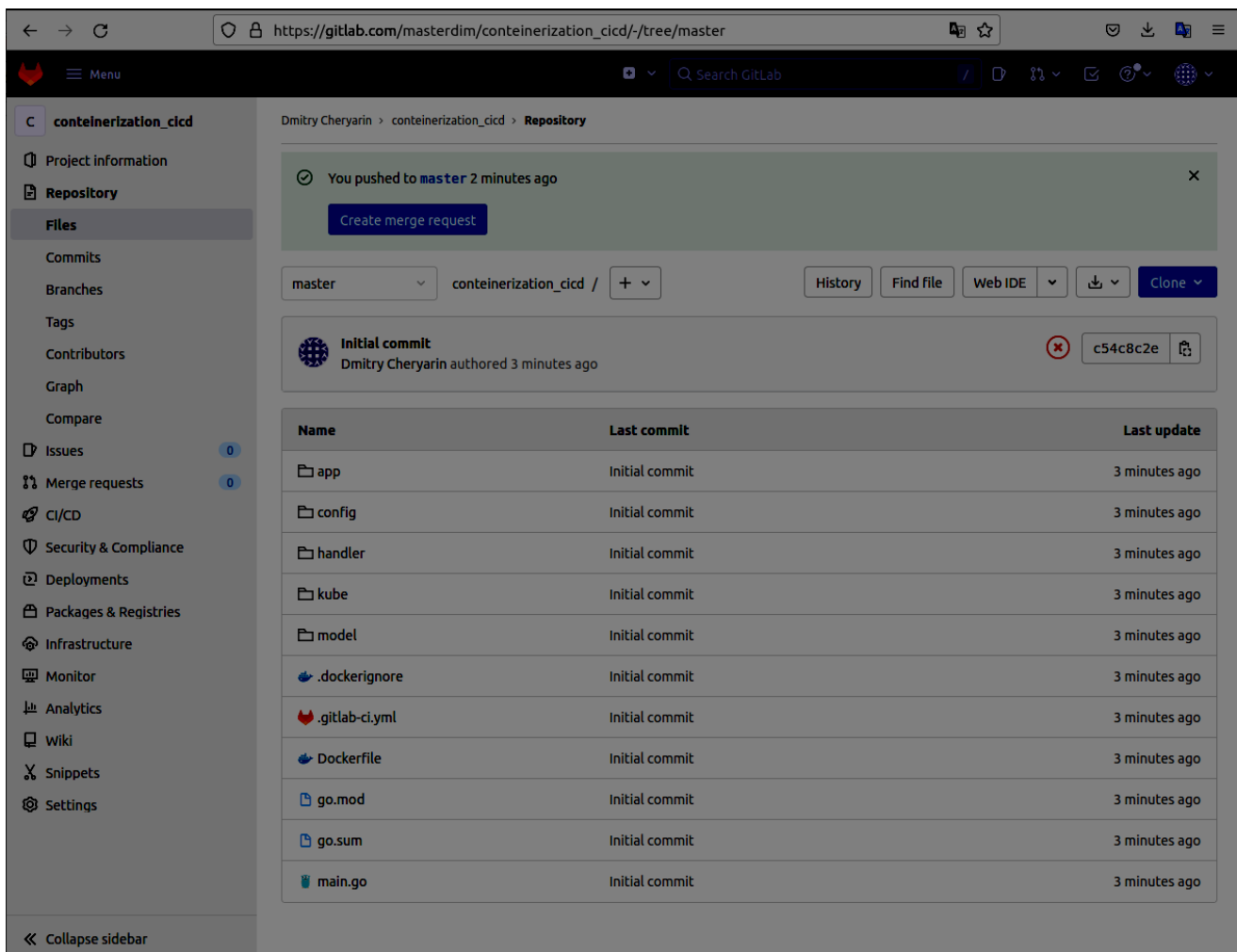
10 Скопируйте файлы практики в ваш репозиторий

```
cd app
git init
git remote add origin git@gitlab.com:<your_gitlab_account>/geekbrains.git
git add .
git commit -m "Initial commit"
git push -u origin master
```

```

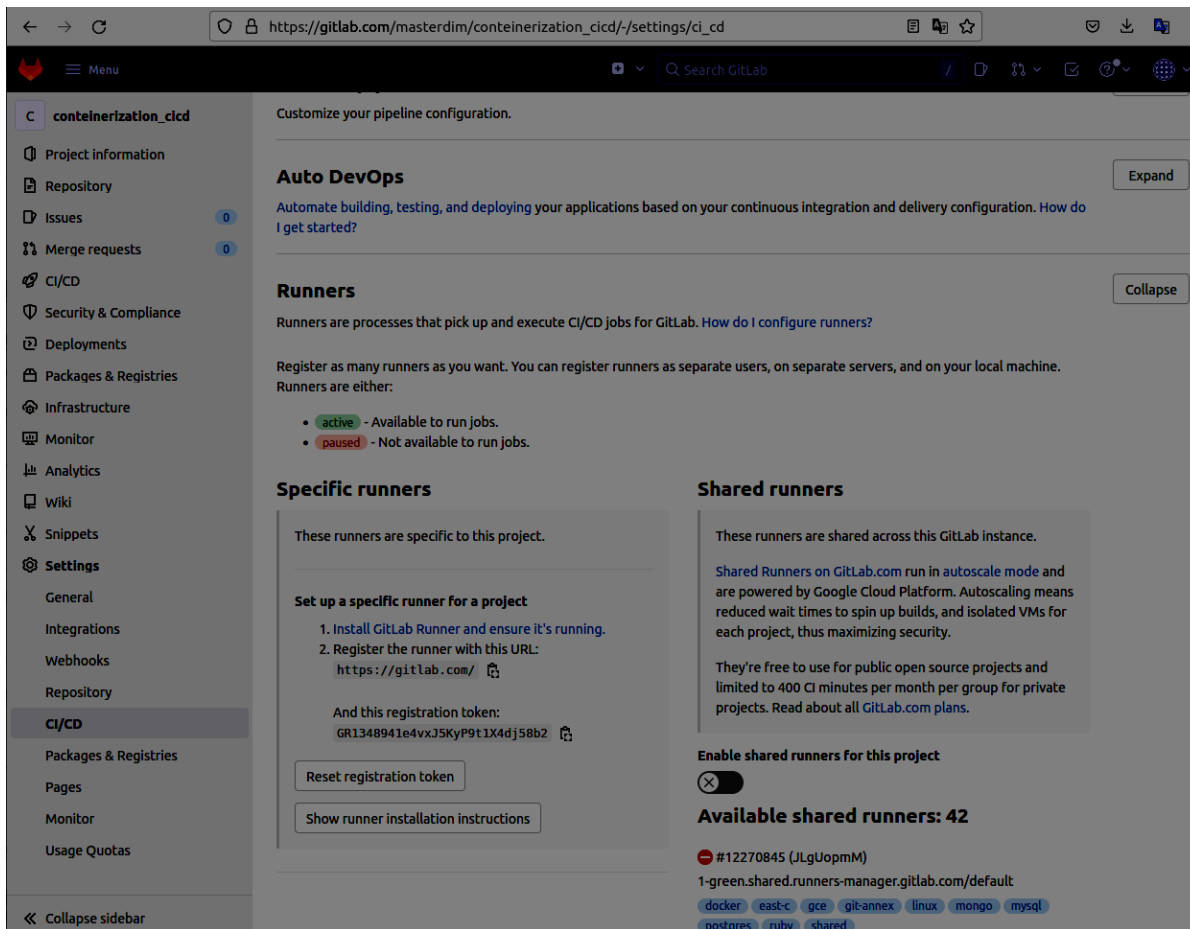
dmitry@dmityev: /codebrains/4_Semestr/practice/1_containerization/lesson_08$ cd app
dmitry@dmityev: /codebrains/4_Semestr/practice/1_containerization/lesson_08/app$ git init
Initialized empty Git repository in /home/dmitry/GeekBrains/4_Semestr/practice/1_containerization/lesson_08/app/.git/
dmitry@dmityev: /codebrains/4_Semestr/practice/1_containerization/lesson_08/app$ git remote add origin git@gitlab.com:masterdm/containerization_cicd.git
dmitry@dmityev: /codebrains/4_Semestr/practice/1_containerization/lesson_08/app$ git add .
dmitry@dmityev: /codebrains/4_Semestr/practice/1_containerization/lesson_08/app$ git commit -m "Initial commit"
[master (root-commit) c54c8c2] Initial commit
17 files changed, 560 insertions(+)
create mode 100644 .dockerignore
create mode 100644 .gitlab-ci.yml
create mode 100644 Dockerfile
create mode 100644 app/app.go
create mode 100644 config/config.go
create mode 100644 go.mod
create mode 100644 go.sum
create mode 100644 handler/common.go
create mode 100644 handler/users.go
create mode 100644 kube/deployment.yaml
create mode 100644 kube/ingress.yaml
create mode 100644 kube/postgres/secret.yaml
create mode 100644 kube/postgres/service.yaml
create mode 100644 kube/postgres/statefulset.yaml
create mode 100644 kube/service.yaml
create mode 100644 main.go
create mode 100644 model/model.go
dmitry@dmityev: /codebrains/4_Semestr/practice/1_containerization/lesson_08/app$ git push -u origin master
The authenticity of host 'gitlab.com (172.65.251.78)' can't be established.
ECDSA key fingerprint is SHA256:HbW3g8zUjNSksFbqTiUWPWg2Bq1x8xdGurliXFzSnUw.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added 'gitlab.com,172.65.251.78' (ECDSA) to the list of known hosts.
Enumerating objects: 25, done.
Counting objects: 100% (25/25), done.
Delta compression using up to 4 threads
Compressing objects: 100% (21/21), done.
Writing objects: 100% (25/25), 7.22 KiB | 1.80 MiB/s, done.
Total 25 (delta 0), reused 0 (delta 0)
remote:
remote: To create a merge request for master, visit:
remote:   https://gitlab.com/masterdm/containerization_cicd/-/merge_requests/new?merge_request%5Bsource_branch%5D=master
remote:
To gitlab.com:masterdm/containerization_cicd.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
dmitry@dmityev: /codebrains/4_Semestr/practice/1_containerization/lesson_08/app$

```



## Настраиваем интеграцию GitLab и Kubernetes

- 10 Переходим в настройки проекта **Settings -> CI/CD -> Runners**. Отключаем Shared Runners. Мы будем настраивать Specific runners.



## 10 Создаем нэйmspэйс для раннера

kubectl create ns gitlab

```

root@ubuntu18: ~# kubectl create ns gitlab
namespace/gitlab created
root@ubuntu18: ~# kubectl get ns
NAME                STATUS    AGE
default              Active    33d
gitlab               Active    13s
ingress-nginx        Active    23d
kube-node-lease      Active    33d
kube-public           Active    33d
kube-system           Active    33d
kubedoom             Active    33d
root@ubuntu18: ~#

```

## 10 Меняем регистрационный токен Для этого открываем gitlab-runner/gitlab-runner.yaml Там ищем и вставляем вместо него токен, который мы взяли в настройках проекта на Gitlab (Set up a specific runner manually -> Registration token)

```

GNU nano 4.8 gitlab-runner.yaml
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: "GR1348941e4vxJ5KyP9tIX4dj58b2"
  runner-token: ""
---
apiVersion: v1
kind: ConfigMap

```

## 10 Применяем манифесты для раннера

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

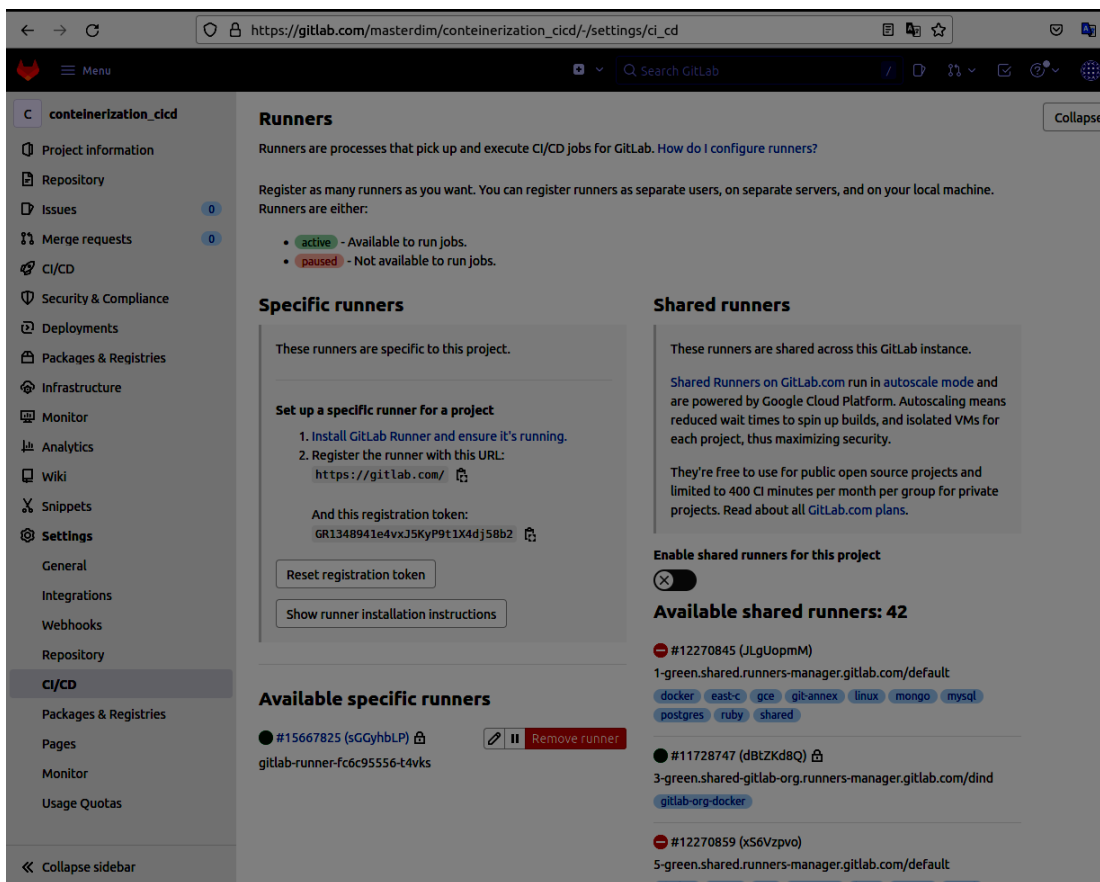
```

dmitry@dubuntu: ~/GeekBrains/4_Semestr/practice/1_containerization/lesson_08/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
dmitry@dubuntu: ~/GeekBrains/4_Semestr/practice/1_containerization/lesson_08/gitlab-runner$

```

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





## 10 Создаем нэйmspэйсы для приложения

```
kubectl create ns stage
kubectl create ns prod
```

```

gitlab-runner: /workspace/4_semestr/practice/1_containerization/lesson_08/gitlab-runner$ kubectl create ns stage
namespace/stage created
gitlab-runner: /workspace/4_semestr/practice/1_containerization/lesson_08/gitlab-runner$ kubectl create ns prod
namespace/prod created
gitlab-runner: /workspace/4_semestr/practice/1_containerization/lesson_08/gitlab-runner$ kubectl get ns
NAME                STATUS   AGE
default              Active   35d
gitlab               Active   44h
ingress-nginx        Active   24d
kube-node-lease      Active   35d
kube-public          Active   35d
kube-system          Active   35d
kubedoom             Active   35d
prod                 Active   10s
stage                Active   16s
gitlab-runner: /workspace/4_semestr/practice/1_containerization/lesson_08/gitlab-runner$

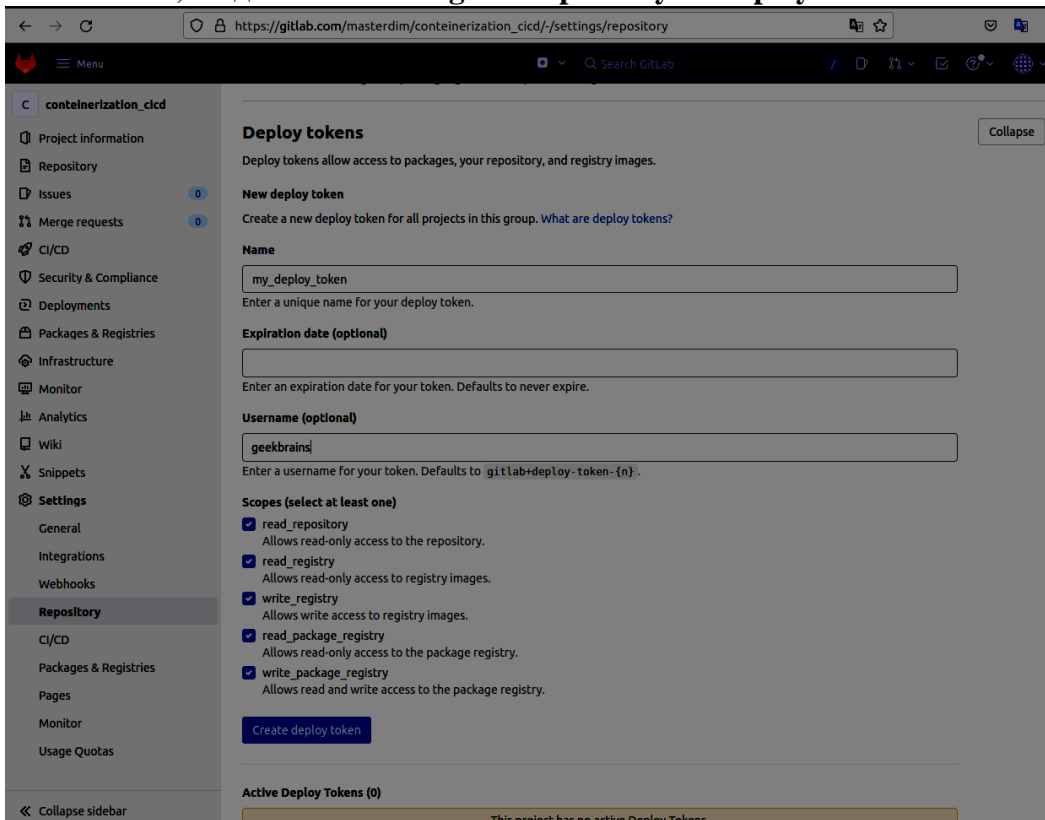
```

## 10 Создаем авторизационные объекты, чтобы раннер мог деплоить в наши нэйmspэйсы

```
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
```



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



```
kubectl create secret docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=<USERNAME> --docker-password=<PASSWORD> --docker-email=admin@admin.admin --namespace stage
kubectl create secret docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=<USERNAME> --docker-password=<PASSWORD> --docker-email=admin@admin.admin --namespace prod
```

```
root@gitlab-runner:~/containerization/lesson_08/gitlab-runner# kubectl create secret
docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=geekbrains --docker-password=
J4JfvJzQhpX45JHmZZk2 --docker-email=admin@admin.admin --namespace stage
secret/gitlab-registry created
root@gitlab-runner:~/containerization/lesson_08/gitlab-runner# kubectl create secret
docker-registry gitlab-registry --docker-server=registry.gitlab.com --docker-username=geekbrains --docker-password=
J4JfvJzQhpX45JHmZZk2 --docker-email=admin@admin.admin --namespace prod
secret/gitlab-registry created
```

- 10 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
```

```
root@gitlab-runner:~/containerization/lesson_08/gitlab-runner# kubectl patch
serviceaccount default -p '{"imagePullSecrets": [{"name": "gitlab-registry"}]}' -n stage
serviceaccount/default patched
root@gitlab-runner:~/containerization/lesson_08/gitlab-runner# kubectl patch
serviceaccount default -p '{"imagePullSecrets": [{"name": "gitlab-registry"}]}' -n prod
serviceaccount/default patched
root@gitlab-runner:~/containerization/lesson_08/gitlab-runner#
```

# Запуск приложения

## 10 Создаем манифесты для БД в stage и prod

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

```
int@ubuntu:~/workspace/semestr/practice/1_containerization/lesson_09$  
int@ubuntu:~/workspace/semestr/practice/1_containerization/lesson_09$ kubectl apply --namespace stage -f app/kube/postgres/  
secret/app created  
service/database created  
statefulset.apps/database created  
int@ubuntu:~/workspace/semestr/practice/1_containerization/lesson_09$  
int@ubuntu:~/workspace/semestr/practice/1_containerization/lesson_09$ kubectl apply --namespace prod -f app/kube/postgres/  
secret/app created  
service/database created  
statefulset.apps/database created  
int@ubuntu:~/workspace/semestr/practice/1_containerization/lesson_09$
```

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

```
gitlab-runner.yaml x gitlab.txt x *Ingress.yaml x  
1 apiVersion: networking.k8s.io/v1  
2 kind: Ingress  
3 metadata:  
4   name: containerization-cicd  
5   annotations:  
6     nginx.ingress.kubernetes.io/rewrite-target: /$1  
7 spec:  
8   rules:  
9     - host: stage  
10     http:  
11       paths:  
12         - path: /user  
13           pathType: Prefix  
14           backend:  
15             service:  
16               name: containerization-cicd  
17               port:  
18                 number: 8000
```

Далее применяем на stage

```
kubectl apply --namespace stage -f app/kube
```

```
int@ubuntu:~/workspace/semestr/practice/1_containerization/lesson_09/app$ kubectl apply --namespace stage -f kube/  
deployment.apps/containerization-cicd created  
ingress.networking.k8s.io/containerization-cicd created  
service/containerization-cicd created  
int@ubuntu:~/workspace/semestr/practice/1_containerization/lesson_09/app$
```

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

```
gitlab-runner.yaml x gitlab.txt x *Ingress.yaml x  
1 apiVersion: networking.k8s.io/v1  
2 kind: Ingress  
3 metadata:  
4   name: containerization-cicd  
5   annotations:  
6     nginx.ingress.kubernetes.io/rewrite-target: /$1  
7 spec:  
8   rules:  
9     - host: prod  
10     http:  
11       paths:  
12         - path: /user  
13           pathType: Prefix  
14           backend:  
15             service:  
16               name: containerization-cicd  
17               port:  
18                 number: 8000
```

Далее применяем на prod

```
kubectl apply --namespace prod -f app/kube
```

```
:  
deployment.apps/containerization-cicd created  
ingress.networking.k8s.io/containerization-cicd created  
service/containerization-cicd created  
: ~$ kubectl apply --namespace prod -f app/kube/
```

## Проверяем работу приложения

Поздравляю! 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, вам нужно заменить его на EXTERNAL-IP вашего 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"
```

1.

Переделайте шаг деплоя в CI/CD, который демонстрировался на лекции таким образом, чтобы при каждом прогоне шага deploy в кластер применялись манифесты приложения. При этом версия докер образа в деплойменте при апплее должна подменяться на ту, что была собрана в шаге build.

Для этого самым очевидным способом было бы воспользоваться утилитой sed.

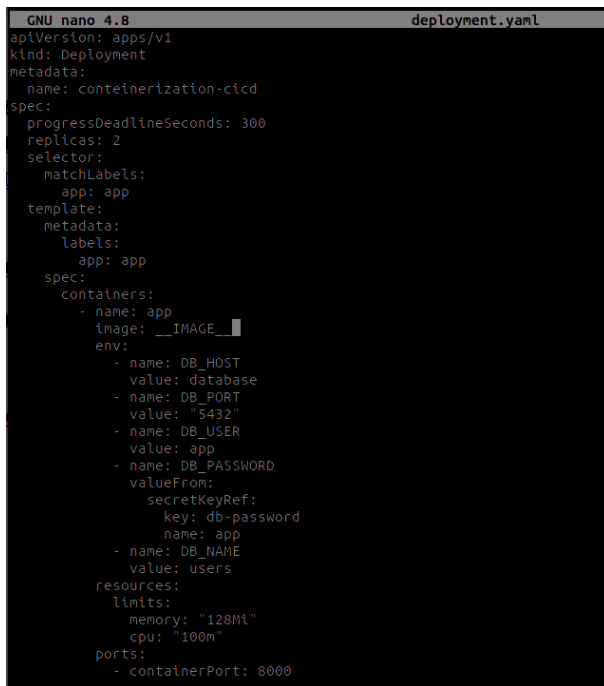
⑩ Измените образ в деплойменте приложения (файл kube/deployment.yaml) на плейсхолдер.

Вот это

```
image: nginx:1.12 # это просто плейсхолдер
```

На это

```
image: __IMAGE__
```



```
GNU nano 4.8 deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: containerization-cicd
spec:
  progressDeadlineSeconds: 300
  replicas: 2
  selector:
    matchLabels:
      app: app
  template:
    metadata:
      labels:
        app: app
    spec:
      containers:
        - name: app
          image: __IMAGE__
          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
```

2.

Измените шаг деплоя в .gitlab-ci.yml, чтобы изменять **IMAGE** на реальное имя образа и тег

Это

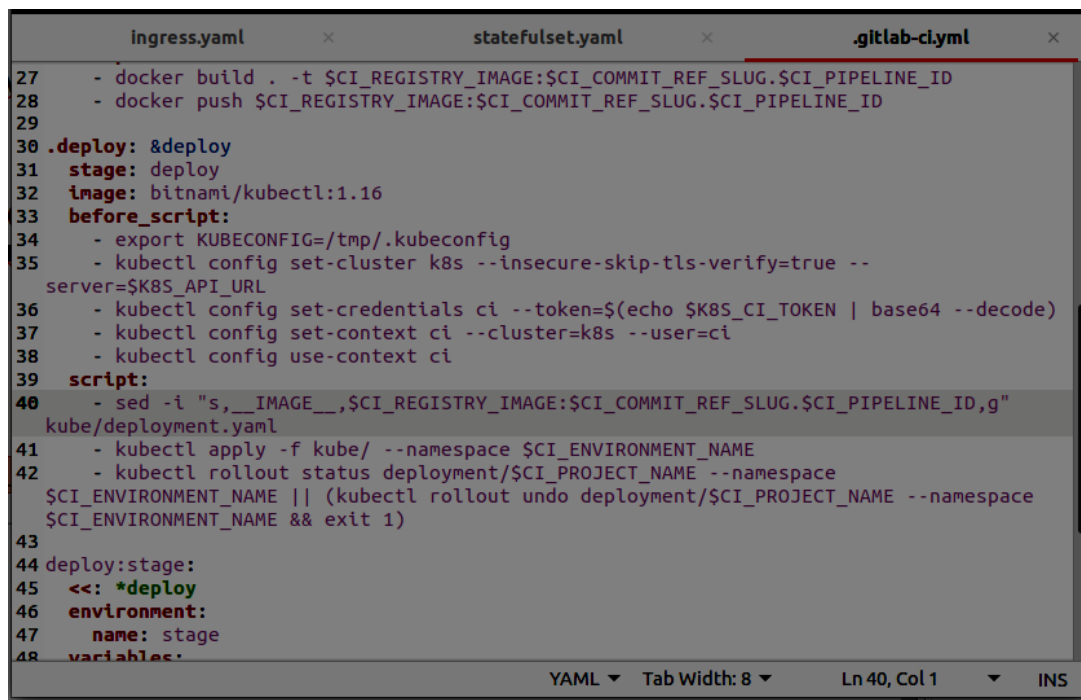
```
- kubectl set image deployment/$CI_PROJECT_NAME
*=$CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID --namespace $CI_ENVIRONMENT_NAME
```

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

Вторую строчку шага деплоя (которая отслеживает статус деплоя) оставьте без изменений.



```
ingress.yaml x statefulset.yaml x .gitlab-ci.yml x
27 - docker build . -t $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID
28 - docker push $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID
29
30 .deploy: &deploy
31   stage: deploy
32   image: bitnami/kubectl:1.16
33   before_script:
34     - export KUBECONFIG=/tmp/.kubeconfig
35     - kubectl config set-cluster k8s --insecure-skip-tls-verify=true --
      server=$K8S_API_URL
36     - kubectl config set-credentials ci --token=$(echo $K8S_CI_TOKEN | base64 --decode)
37     - kubectl config set-context ci --cluster=k8s --user=ci
38     - kubectl config use-context ci
39   script:
40     - sed -i "s,__IMAGE__, $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID,g"
      kube/deployment.yaml
41     - kubectl apply -f kube/ --namespace $CI_ENVIRONMENT_NAME
42     - kubectl rollout status deployment/$CI_PROJECT_NAME --namespace
      $CI_ENVIRONMENT_NAME || (kubectl rollout undo deployment/$CI_PROJECT_NAME --namespace
      $CI_ENVIRONMENT_NAME && exit 1)
43
44 deploy: stage:
45   <<: *deploy
46   environment:
47     name: stage
48   variables:
```

3.

Попробуйте закоммитить свои изменения, запустить их в репозиторий (тот же, который вы создавали во время лекции на Gitlab.com) и посмотреть на выполнение CI в интерфейсе Gitlab.

Так как окружений у нас два (stage и prod), то помимо образа при апплае из CI нам также было бы хорошо подменять host в ingress.yaml. Попробуйте реализовать это по аналогии, подставляя в ингресс вместо плейсхолдера значение переменной \$CI\_ENVIRONMENT\_NAME

```
ingress.yaml  x  .gitlab-ci.yml  x
1 apiVersion: networking.k8s.io/v1
2 kind: Ingress
3 metadata:
4   name: containerization-cicd
5   annotations:
6     nginx.ingress.kubernetes.io/rewrite-target: /$1
7 spec:
8   rules:
9     - host: __MYHOSTNAME__
10     http:
11       paths:
12         - path: /user
13           pathType: Prefix
14           backend:
15             service:
16               name: containerization-cicd
17               port:
18                 number: 8000
```

```
.gitlab-ci.yml  x  ingress.yaml  x  deployment.yaml
34 - export KUBECONFIG=/tmp/.kubeconfig
35 - kubectl config set-cluster k8s --insecure-skip-tls-verify=true --server=$K8S_API_URL
36 - kubectl config set-credentials ci --token=$(echo $K8S_CI_TOKEN | base64 --decode)
37 - kubectl config set-context ci --cluster=k8s --user=ci
38 - kubectl config use-context ci
39 script:
40 - sed -i "s, IMAGE, $SCI_REGISTRY IMAGE:$SCI_COMMIT_REF_SLUG.$SCI_PIPELINE_ID,g" kube/deployment.yaml
41 - sed -i "s, MYHOSTNAME, $SCI_ENVIRONMENT_NAME,g" kube/ingress.yaml
42 - kubectl apply -f kube/ --namespace $SCI_ENVIRONMENT_NAME
43 - kubectl rollout status deployment/$SCI_PROJECT_NAME --namespace $SCI_ENVIRONMENT_NAME || (kubectl rollout undo
  deployment/$SCI_PROJECT_NAME --namespace $SCI_ENVIRONMENT_NAME && exit 1)
44
45 deploy:stage:
46   <<: *deploy
47   environment:
48     name: stage
49   variables:
50     K8S_CI_TOKEN: $K8S_STAGE_CI_TOKEN
51   only:
52     - master
53
54 deploy:prod:
55   <<: *deploy
```

Добавил переменные “prod” и “stage” в файл /etc/hosts :



```

dmtry@ubuntu:~$ kubectl get namespace
NAME                STATUS   AGE
default             Active   39d
gitlab              Active   5d16h
ingress-nginx       Active   28d
kube-node-lease     Active   39d
kube-public         Active   39d
kube-system         Active   39d
kubedoom            Active   39d
prod                Active   3d20h
stage               Active   3d20h
dmtry@ubuntu:~$ echo "$(minikube ip) prod" | sudo tee -a /etc/hosts
[sudo] password for dmtry:
192.168.59.100 prod
dmtry@ubuntu:~$ echo "$(minikube ip) stage" | sudo tee -a /etc/hosts
192.168.59.100 stage
dmtry@ubuntu:~$ cat /etc/hosts
127.0.0.1 localhost
127.0.1.1 ubuntu_srv

# The following lines are desirable for IPv6 capable hosts
::1          ip6-localhost ip6-loopback
fe00::0      ip6-localnet
ff00::0      ip6-mcastprefix
ff02::1      ip6-allnodes
ff02::2      ip6-allrouters
192.168.59.100 hello-world.info
192.168.59.100 ww38.hello-world.info
192.168.59.100 my-service-deploy.local
192.168.59.100 my-service-stage.local
192.168.59.100 my-service-prod.local
192.168.59.100 prod
192.168.59.100 stage
dmtry@ubuntu:~$ ping prod
PING prod (192.168.59.100) 56(84) bytes of data:
64 bytes from hello-world.info (192.168.59.100): icmp_seq=1 ttl=64 time=0.360 ms
^C
--- prod ping statistics ---
1 packets transmitted, 1 received, 0% packet loss, time 0ms
rtt min/avg/max/mdev = 0.360/0.360/0.360/0.000 ms
dmtry@ubuntu:~$

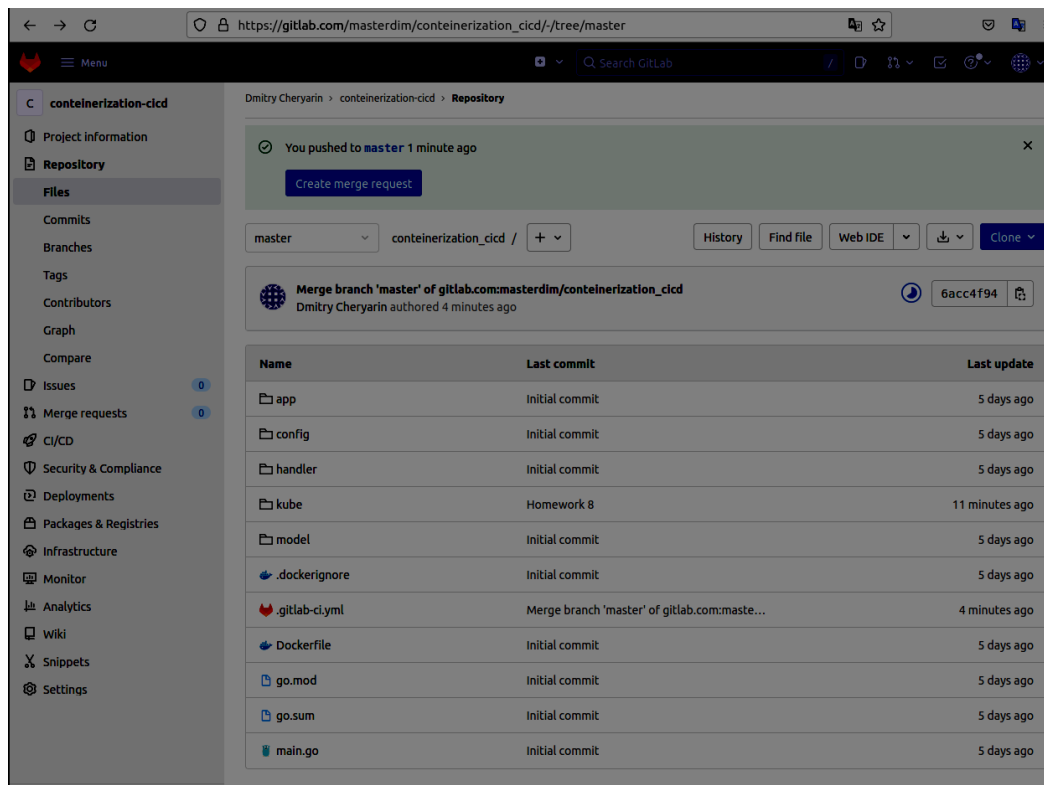
```

Делаю push в удаленный репозиторий GitLab:

```
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$ ls -la
total 56
drwxrwxr-x 8 dmitry dmitry 4096 Jun  3 12:30
drwxrwxr-x 5 dmitry dmitry 4096 Jun  3 12:34
drwxrwxr-x 2 dmitry dmitry 4096 May 28 19:41 app
drwxrwxr-x 2 dmitry dmitry 4096 May 28 19:41 config
-rw-rw-r-- 1 dmitry dmitry 213 May 28 19:41 Dockerfile
-rw-rw-r-- 1 dmitry dmitry 27 May 28 19:41 .dockerignore
drwxrwxr-x 8 dmitry dmitry 4096 May 28 19:59 .git
-rw-rw-r-- 1 dmitry dmitry 1715 Jun  3 12:30 .gitlab-ci.yml
-rw-rw-r-- 1 dmitry dmitry 161 May 28 19:41 go.mod
-rw-rw-r-- 1 dmitry dmitry 2852 May 28 19:41 go.sum
drwxrwxr-x 2 dmitry dmitry 4096 May 28 19:41 handler
drwxrwxr-x 3 dmitry dmitry 4096 Jun  2 18:05 kube
-rw-rw-r-- 1 dmitry dmitry 289 May 28 19:41 main.go
drwxrwxr-x 2 dmitry dmitry 4096 May 28 19:41 node
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$ git remote
origin
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$ git remote --help
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$ git remote -v
origin  git@gitlab.com:masterdim/containerization_cicd.git (fetch)
origin  git@gitlab.com:masterdim/containerization_cicd.git (push)
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$ git add .
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$ git commit -m "Homework 8"
[master ceaeaa6] Homework 8
 5 files changed, 24 insertions(+), 18 deletions(-)
 rewrite kube/ingress.yaml (74%)
```

```
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$ git push -u origin master
Enumerating objects: 28, done.
Counting objects: 100% (22/22), done.
Delta compression using up to 4 threads
Compressing objects: 100% (12/12), done.
Writing objects: 100% (12/12), 1.34 KiB | 688.00 KiB/s, done.
Total 12 (delta 8), reused 0 (delta 0)
remote:
remote: To create a merge request for master, visit:
remote:   https://gitlab.com/masterdim/containerization_cicd/-/merge_requests/new?merge_request%5Bsource_branch%5D=master
remote:
To gitlab.com:masterdim/containerization_cicd.git
 ce51f25..6acc4f9 master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
dmityy@ubuntu:~/geekbrains/4_semestr/practice/1_containerization/lesson_08/app$
```

Проверяю, что в GitLab все залилось обновленное:



Дополнительно добавляю новую переменную в GitLab с содержимым файла настроек моего локального кластера mikikube с типом “FILE”:

```
dmity@dmity:~/joseph-alms/4_semester/practice/1_containerization/lesson_08/app$ cat /home/dmity/.kube/config
apiVersion: v1
clusters:
- cluster:
  certificate-authority: /home/dmity/.minikube/ca.crt
  extensions:
  - extension:
    last-update: Wed, 15 Jun 2022 10:47:44 MSK
    provider: minikube.sigs.k8s.io
    version: v1.25.2
    name: cluster_info
  server: https://192.168.59.100:8443
  name: minikube
contexts:
- context:
  cluster: minikube
  extensions:
  - extension:
    last-update: Wed, 15 Jun 2022 10:47:44 MSK
    provider: minikube.sigs.k8s.io
    version: v1.25.2
    name: context_info
  namespace: default
  user: minikube
  name: minikube
current-context: minikube
```

### Update variable

#### Key

KUBECONFIG

#### Value

```
apiVersion: v1
clusters:
- cluster:
  certificate-authority: /home/dmity/.minikube/ca.crt
  extensions:
  - extension:
    last-update: Wed, 15 Jun 2022 10:47:44 MSK
```

#### Type

File

#### Environment scope

All (default)

#### Flags

☒ Protect variable

Export variable to pipelines running on protected branches and tags only.

☐ Mask variable

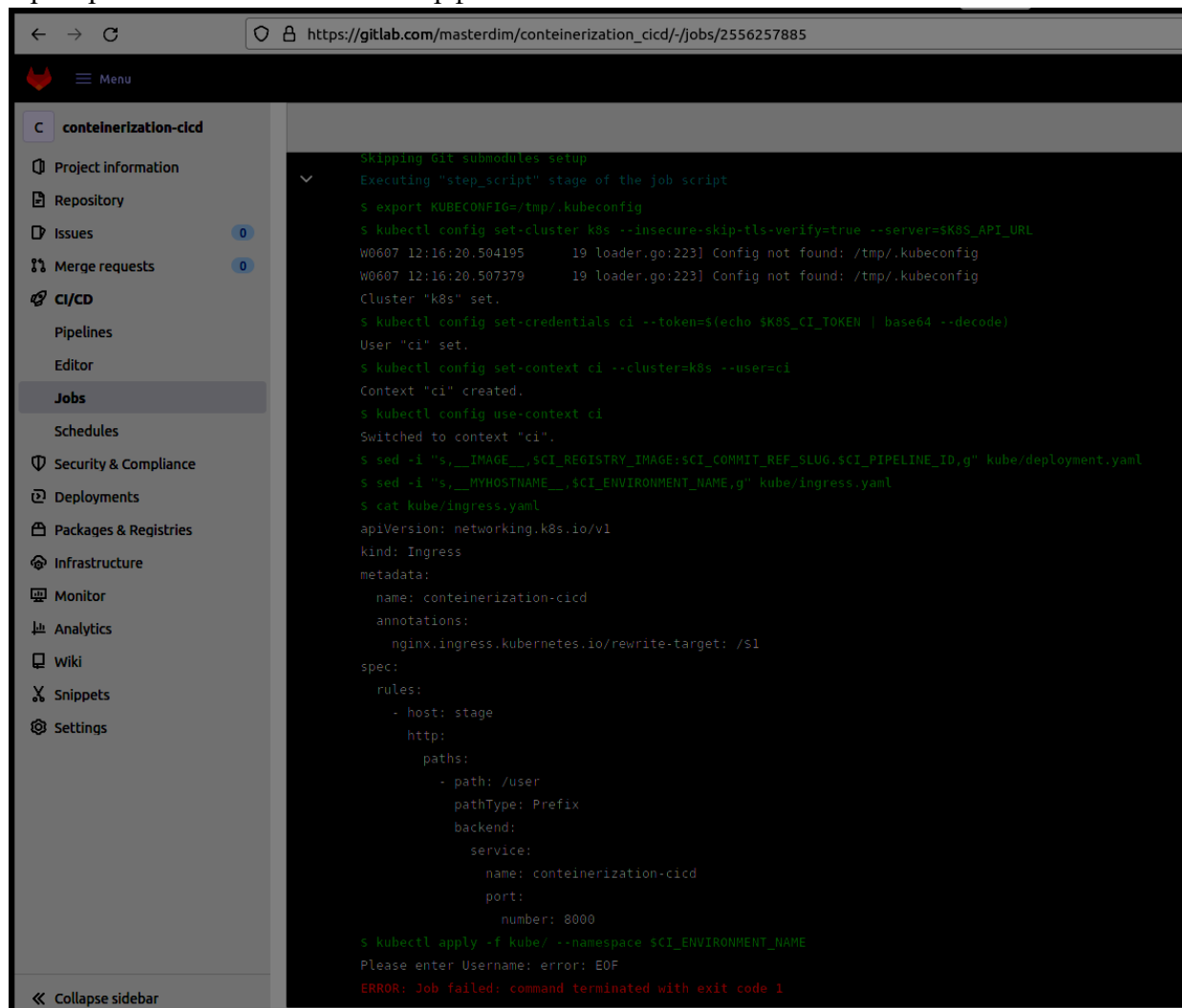
Variable will be masked in job logs. Requires values to meet regular expression requirements. [More information](#)

Cancel

Delete variable

Update variable

Проверяю выполнение CI в интерфейсе Gitlab:



```
Skipping Git submodules setup
Executing "step_script" stage of the job script
$ export KUBECONFIG=/tmp/.kubeconfig
$ kubectl config set-cluster k8s --insecure-skip-tls-verify=true --server=${K8S_API_URL}
W0607 12:16:20.504195      19 loader.go:223] Config not found: /tmp/.kubeconfig
W0607 12:16:20.507379      19 loader.go:223] Config not found: /tmp/.kubeconfig
Cluster "k8s" set.
$ kubectl config set-credentials ci --token=$(echo $K8S_CI_TOKEN | base64 --decode)
User "ci" set.
$ kubectl config set-context ci --cluster=k8s --user=ci
Context "ci" created.
$ kubectl config use-context ci
Switched to context "ci".
$ sed -i "s, __IMAGE__, $CI_REGISTRY_IMAGE:$CI_COMMIT_REF_SLUG.$CI_PIPELINE_ID,g" kube/deployment.yaml
$ sed -i "s, __MYHOSTNAME__, $CI_ENVIRONMENT_NAME,g" kube/ingress.yaml
$ cat kube/ingress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: containerization-cicd
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$1
spec:
  rules:
    - host: stage
      http:
        paths:
          - path: /user
            pathType: Prefix
            backend:
              service:
                name: containerization-cicd
                port:
                  number: 8000
$ kubectl apply -f kube/ --namespace $CI_ENVIRONMENT_NAME
Please enter Username: error: EOF
ERROR: Job failed: command terminated with exit code 1
```

Получаю ошибку, связанную с некорректной авторизацией токенов.

В интернете сказано, что такое иногда бывает.

Удалил раннер и переменные и заново их пересоздал и прописал в yaml-манифесты.

Не помогло.

Ошибка осталась.

Поиски причины ошибки в интернете и попытки ее исправить – успеха не принесли.

Для проверки подстановки контура ( stage ) добавил в файл .gitlab-ci.yml вывод ingress-манифеста: параметр “host” указан верно – “stage”.

По причине ошибки закончить данную работу не удалось, к сожалению.