Московский государственный технический университет им. Н.Э. Баумана

Факультет «Информатика и системы управления»

Кафедра «Системы обработки информации и управления»



Рубежный контроль №2

по дисциплине «Автоматизация разработки и эксплуатации ПО»

«Итоговый проект»

ИСПОЛНИТЕЛЬ:

студент ИУ5-73Б

Алёшин А.Д.

1. Подлключаем раннер по инструкции в Gitlab

sudo curl -L --output /usr/local/bin/gitlab-runner https://gitlab-runner-downloads.s3.amazonaws.com/latest/binaries/gitlab-runner-linux-amd64

# Give it permissions to execute

sudo chmod +x /usr/local/bin/gitlab-runner

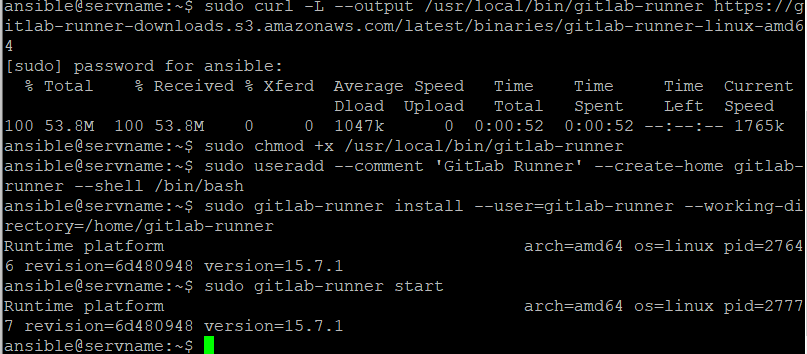
# Create a GitLab CI user

sudo useradd --comment 'GitLab Runner' --create-home gitlab-runner --shell /bin/bash

# Install and run as service

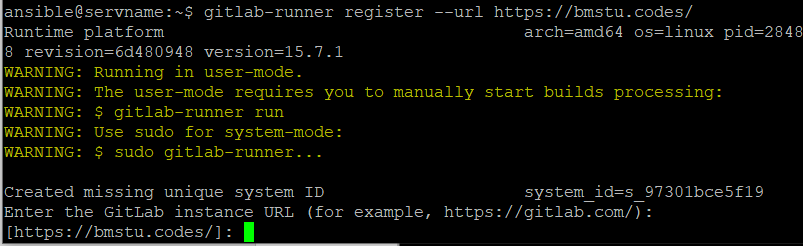
sudo gitlab-runner install --user=gitlab-runner --working-directory=/home/gitlab-runner

sudo gitlab-runner start



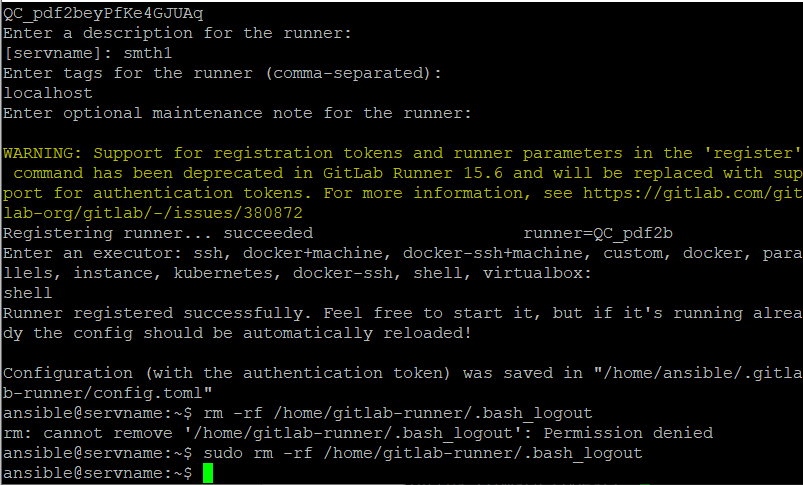
Создаем раннер на сервере командой

gitlab-runner register --url https://bmstu.codes/



Немного магии

rm -rf /home/gitlab-runner/.bash\_logout



1. Создаем в корне проекта .gitlab-ci.yml вставляем в него тестовую джобу

job:

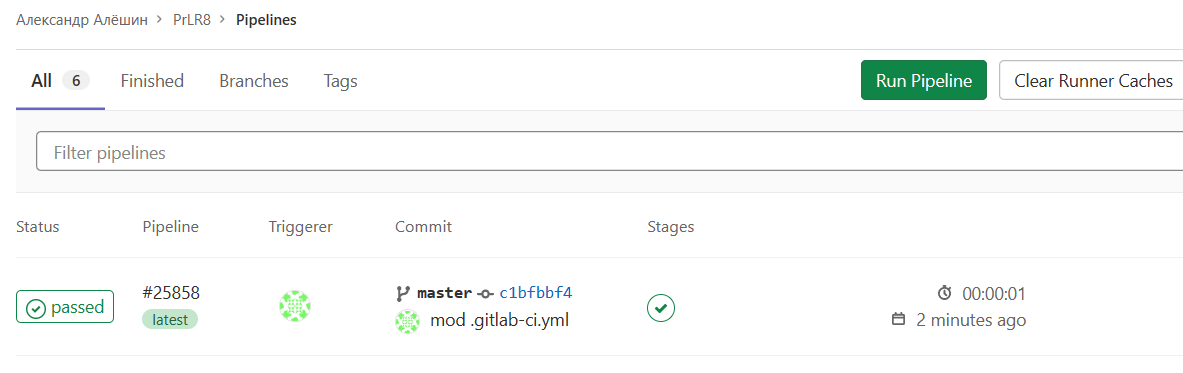
tags:

- localhost

script:

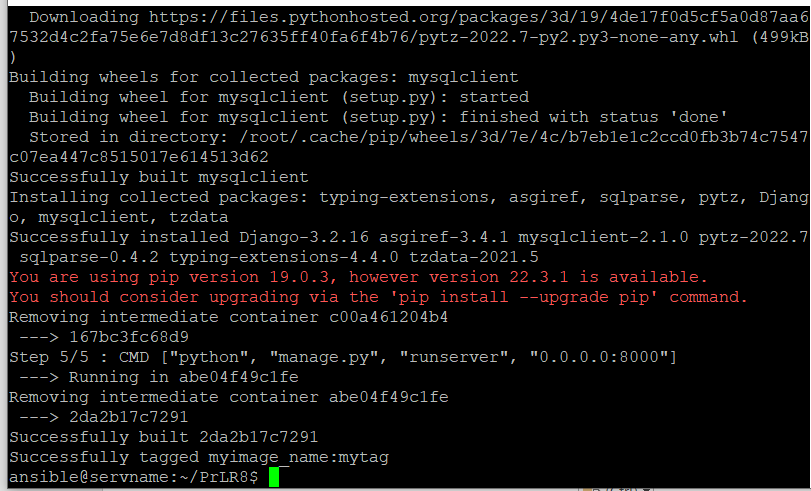
- echo test

Коммитим, пушим, проверяем, что CI отработал.



1. Создаем в репозитории Dockerfile и собираем образ из нашего проекта.

docker build -t myimage\_name:mytag .



1. В .gitlab-ci.yml создаем 3 стейджа: build, upload, deploy
2. В build stage создаем джобу build в которой собираем наш образ и тегируем его по хешу коммита docker build -t myimage\_name:$CI\_COMMIT\_SHORT\_SHA ...

script:

- docker build -t myimage\_name:$CI\_COMMIT\_SHORT\_SHA ...

1. В upload stage создаем джобу upload в которой загружаем наш образ в kind

script:

- kind load docker-image myapp:${CI\_COMMIT\_SHORT\_SHA}

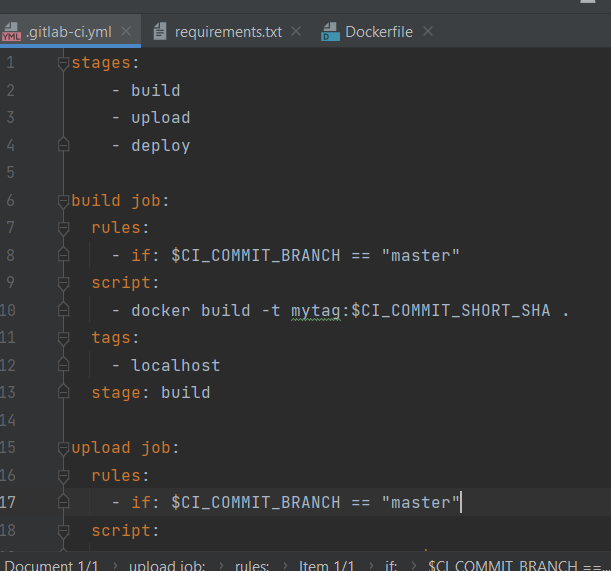
1. В deploy stage создаем джобу deploy.

environment:

name: production

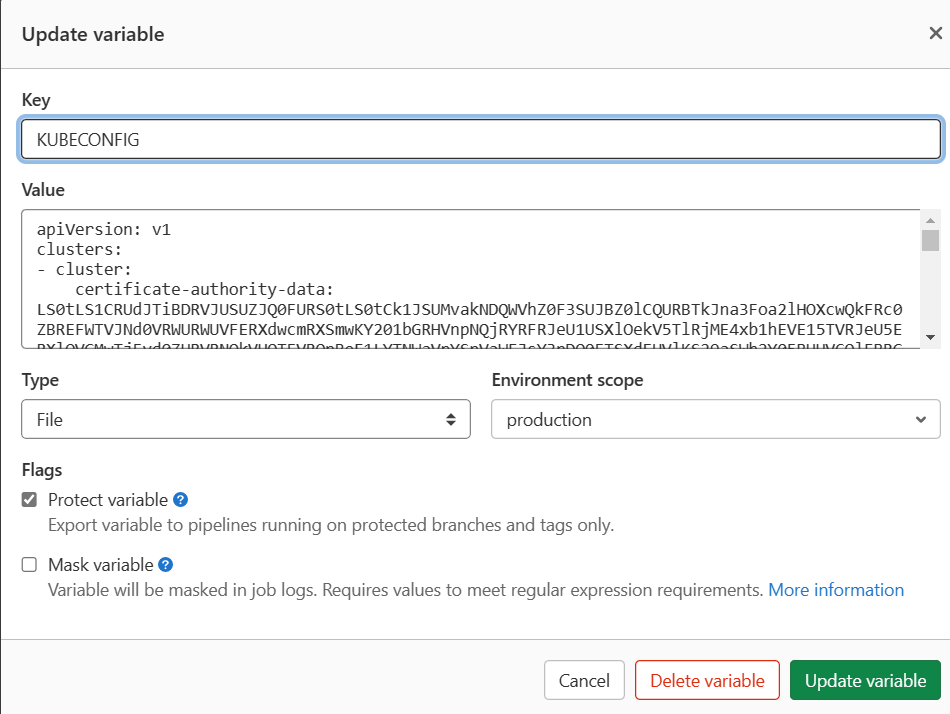
script:

- kubectl get pods



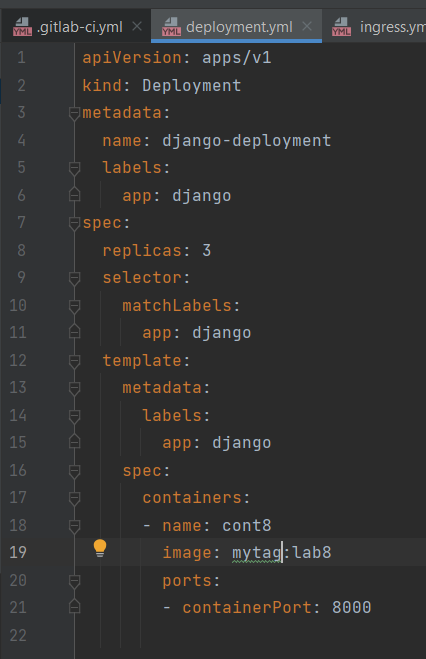
Откроем настройки проекта Settigns -> CI/CD -> Variables и создадим переменную с именем KUBECONFIG типа File, в environment scope должен был появиться наш production, и включить protected variable. Скопируем значение нашего конфига из /root/.kube/config и вставим его

Перезапускаем пайплайн, CI должен пройти и в логах увидим список подов.

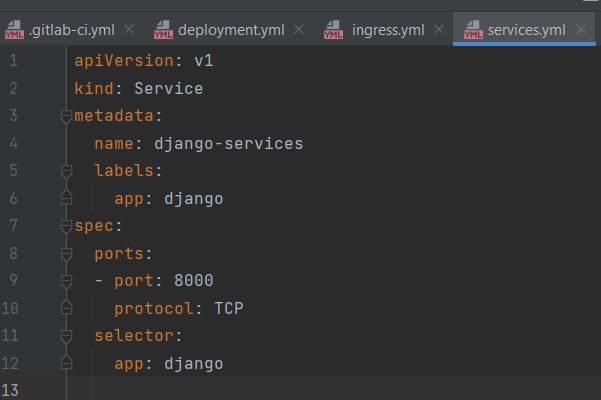


1. Создаем папку manifests, в ней либо один manifest.yml, либо несколько service.yml, deployment.yml, ingress.yml. Описываем манифесты для service, deployment и ingress. Также в корне проекта создайте папку etc или configs и скопируйте туда все конфиги приложения.

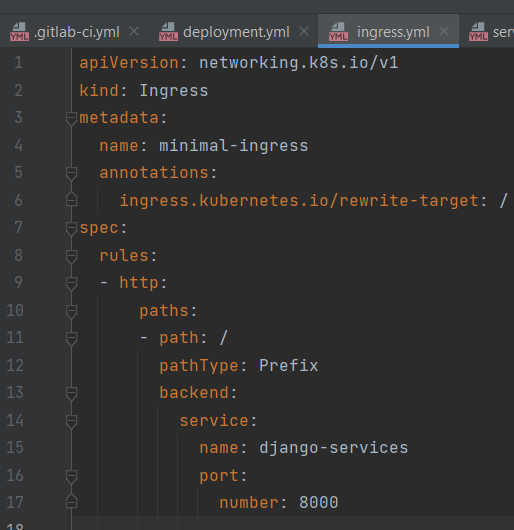
Deployment.yml



Service.yml

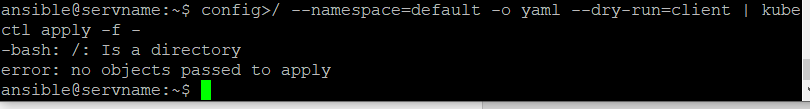


Ingress.yml



1. Создаем конгфигмап

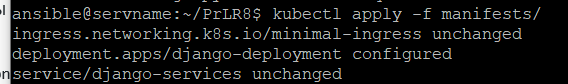
kubectl create configmap myapp-conf --from-file=< etc или config>/ --namespace=default -o yaml --dry-run=client | kubectl apply -f –



Конгфигмапов у меня нет – мне достаточно стандартного конифга nginx

1. Деплоим руками первоначальные манифесты

kubectl apply -f manifests/



1. Обновляем джобу deploy, чтобы обновлять конфиги и образ автоматически

script:

- kubectl create configmap myapp-conf --from-file=< etc или config>/ --namespace=default -o yaml --dry-run=client | kubectl apply -f -

- kubectl set image deployment/my-deployment mycontainer=myapp:${CI\_COMMIT\_SHORT\_SHA}

