| C:\Users\DENACT~1\AppData\Local\Temp\lu135925on38x.tmp\lu135925on3bu_tmp_3360867a00ce4d37.jpg | **Министерство науки и высшего образования Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования**  **«Московский государственный технический университет имени Н.Э. Баумана**  **(национальный исследовательский университет)» (МГТУ им. Н.Э. Баумана)** |
| --- | --- |

ФАКУЛЬТЕТ Информатика и системы управления и искусственный интеллект

КАФЕДРА                  Системы обработки информации и управления

**Лабораторная работа №8**

### По курсу

### «Автоматизация разработки и эксплуатации ПО»

**«Project Deployment»**

Подготовил:

Студент группы

**ИУ5-75Б Кириллов Д.С.**

15.12.2022

Проверил:

**Кучеренко М.А.**

*2022 г*.

**Полученное задание:**

* Взять какой-нибудь проект (например, курсовая по сетям, заготовки диплома или что-нибудь еще)
* Развернуть его на виртуальной машине с kubernetes

**Ход работы:**

Склонировал свой репозиторий из ЛР-4.

Поставлю раннер в ВМ.

Подкллючаем 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

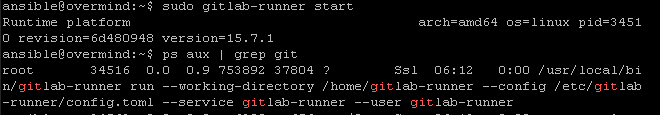
# Create a GitLab CI user

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



Добавим юзера gitlab-runner в группу docker

sudo usermod -aG docker gitlab-runner

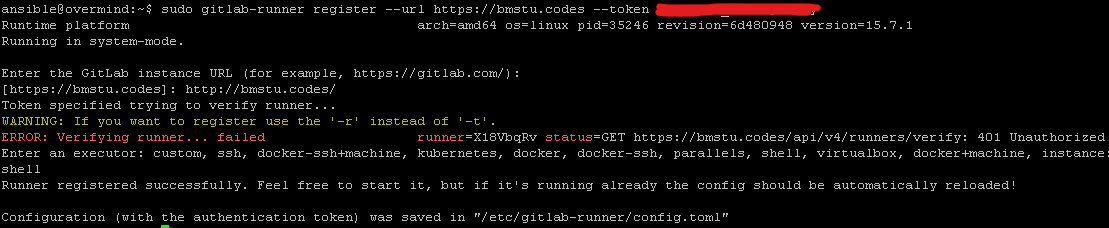
Создаем раннер на сервере командой

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

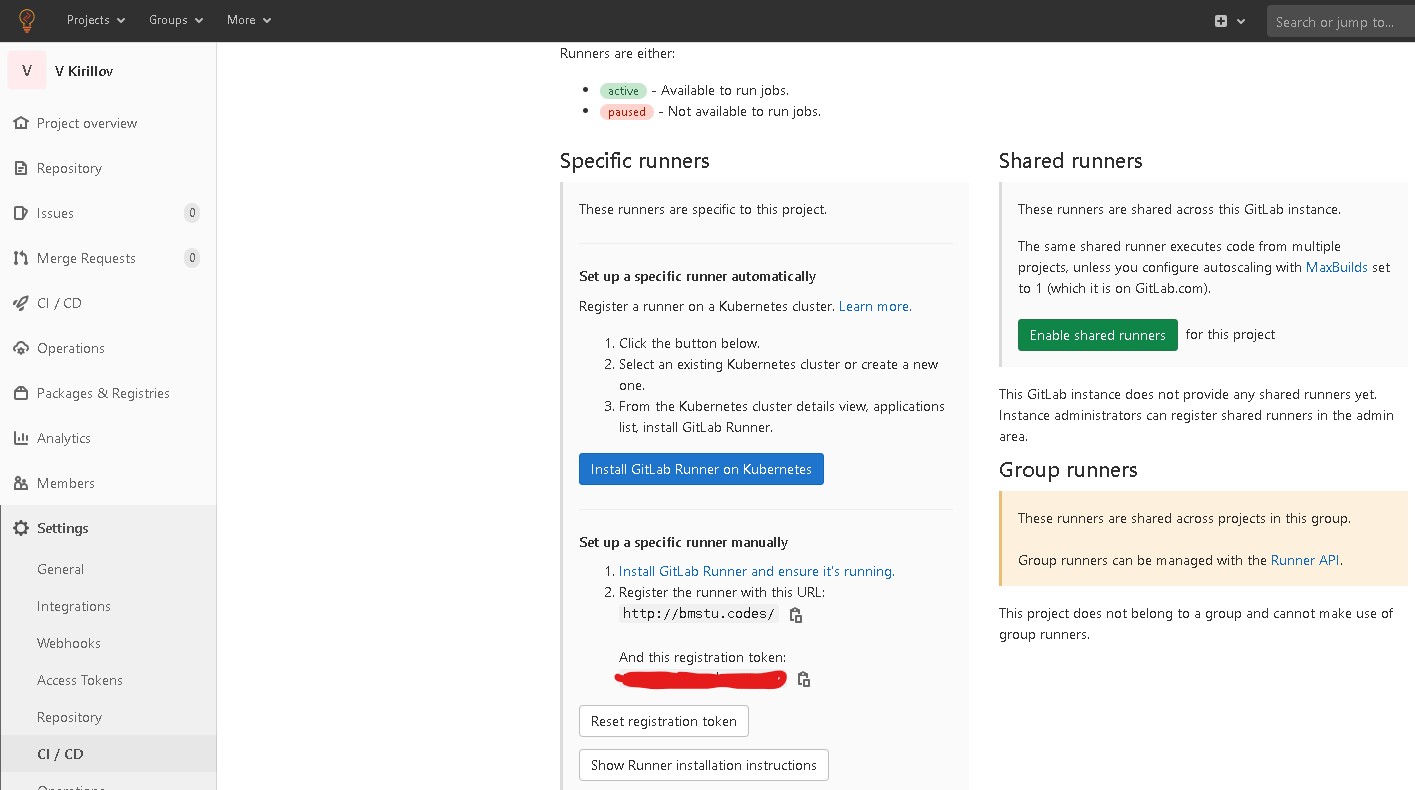
можно так:

sudo gitlab-runner register --url https://bmstu.codes --registration-token Pod1xSQkEUrnqnFZzTzE

! Важно, проверьте, что хост гитлаба с https. Вставляем registration token, в списке тегов укажите что угодно, например, localhost. Тип раннера **shell**



Токен брать специальный для раннера тут: проект > settings > CI/CD > Runners > expand

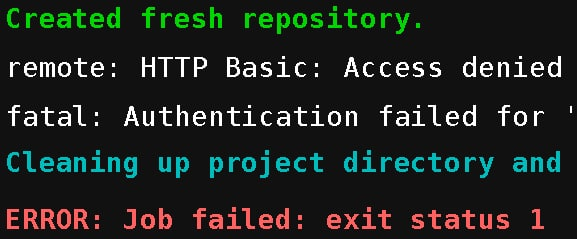


Немного магии

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

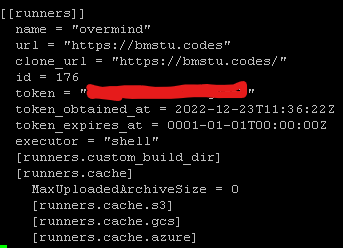
Проверяю работу пайплайна. Проект > CI/CD > Run, подтвердить

Ниче не работает



Решается магией:

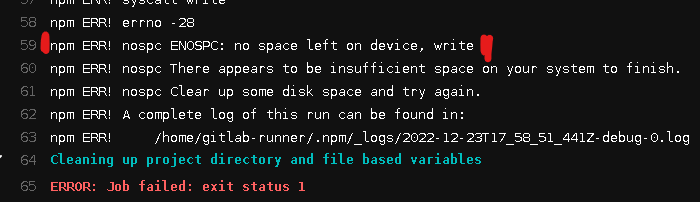
sudo nano /etc/gitlab-runner/config.toml

 - решение

sudo gitlab-runner restart



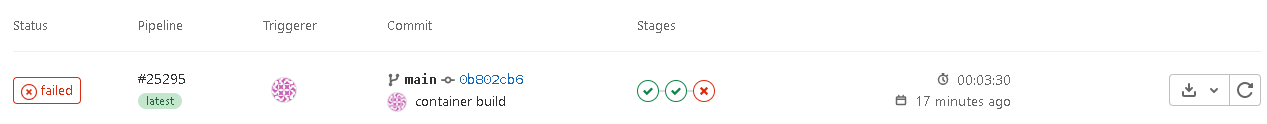
Ну все

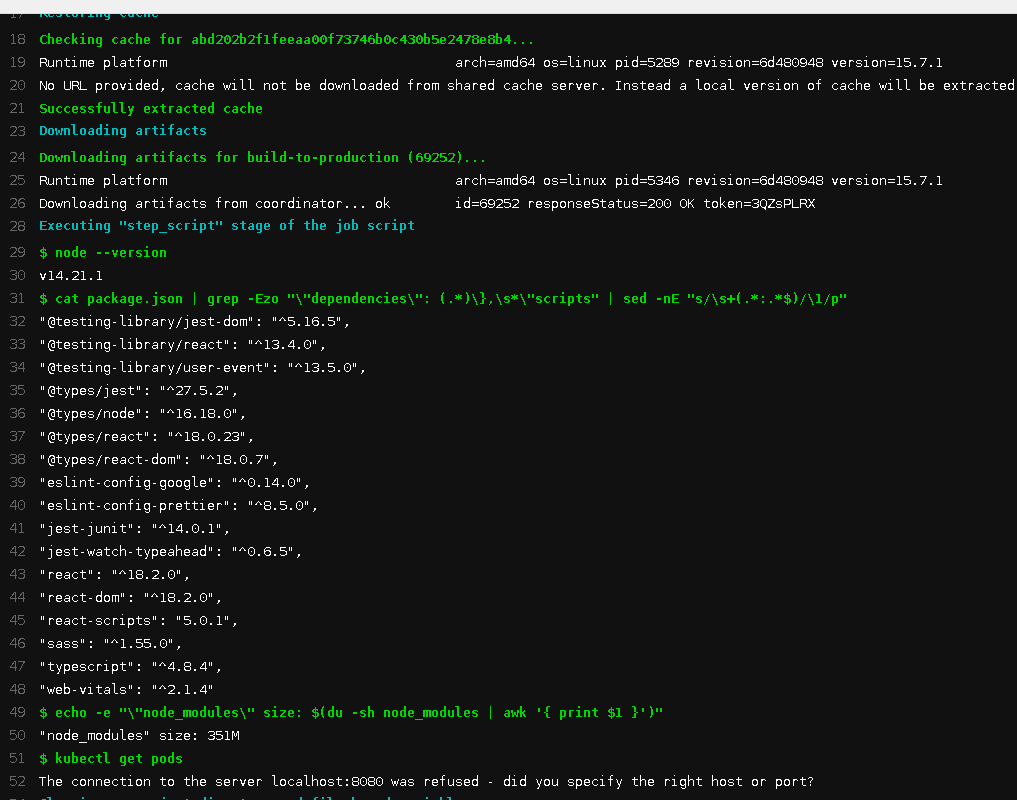


docker system prune –a – освободил аж 3 Гб из 19. Можно жить

ВМ с раннером жрет очень много ОЗУ. Когда зависает, просто перезагружаем хост, лишние процессы умирают.

По итогу, проект собрался, контейнер в прод собрался. Сам деплой (3-я задача) упал, как и должен был, т.к. я не указал параметры подключения к кластеру.

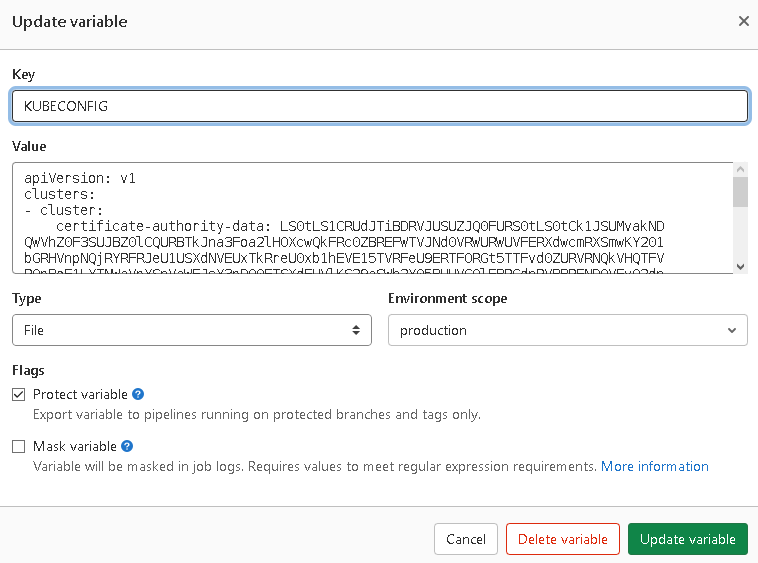


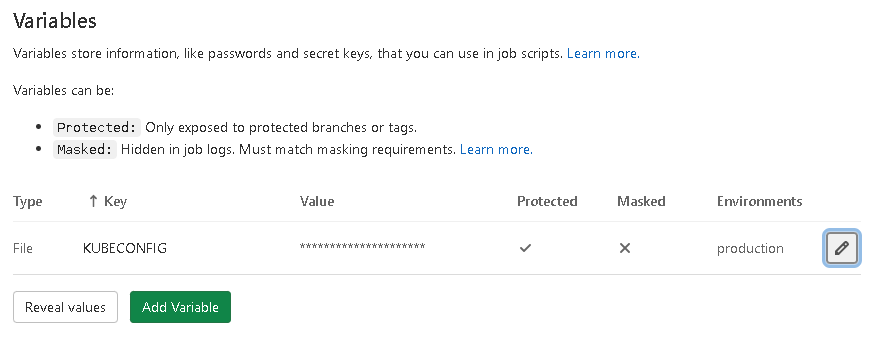


Откроем настройки проекта Settings -> CI/CD -> Variables и создадим переменную с именем KUBECONFIG типа File, в environment scope должен был появиться наш production, и включить protected variable. Скопируем значение нашего конфига из ~~/root/.kube/config~~ ~ /.kube/config и вставим его.

Копировать лучше через scp, а то какие-то беды с кодировкой

scp -P 2222 ansible@127.0.0.1:.kube/config ./config.txt



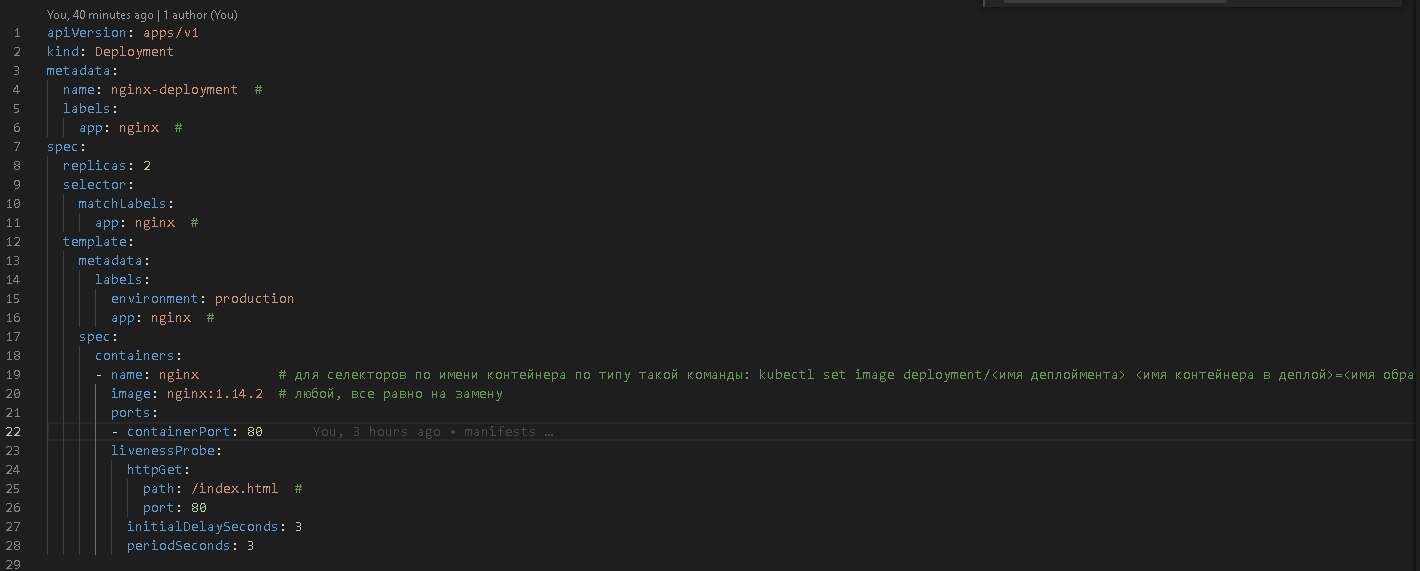


Перезапускаем пайплайн, CI должен пройти и в логах увидим список подов.

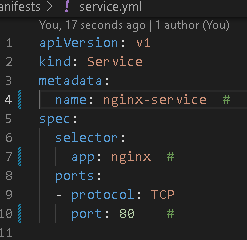
1. Создаем папку manifests, в ней либо один manifest.yml, либо несколько service.yml, deployment.yml, ingress.yml. Описываем манифесты для service, deployment и ingress. Также в корне проекта создайте папку etc или configs и скопируйте туда все конфиги приложения.

Решетками пометил все то, что потенциально должно быть изменено

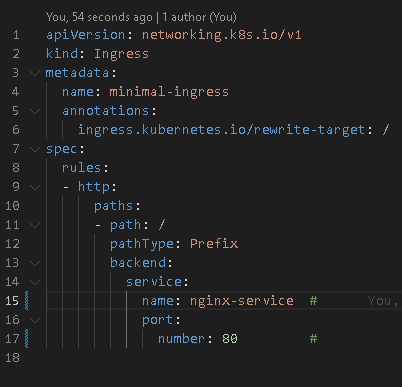
Deployment.yml



Service.yml

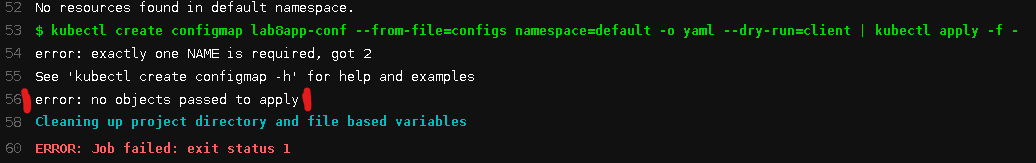


Ingress.yml



1. Создаем конгфигмап

Конгфигмапов у меня нет – мне достаточно стандартного конифга nginx.



1. Деплоим руками первоначальные манифесты

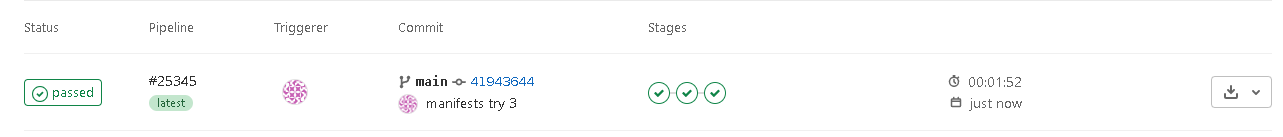
kubectl apply -f manifests/

Можно вогнать это прямо в пайплайн, т.к. если manifests не изменены, то строчка будет игнорироваться. А можно скопировать конфиги на виртуальный кластер ( через MINGW64: scp -P 2222 -r ./manifests ansible@127.0.0.1:./ ) и прогнать kubectl apply -f manifests/

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

kubectl set image deployment/<имя деплоймента> <имя контейнера в деплой>=<имя образа>:${CI\_COMMIT\_SHORT\_SHA}

Прошел!

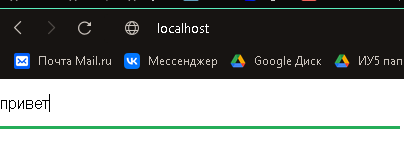


Но увы. При этом иконка react app прогрузилась, частично работает



Проблема была в образе nginx:alpine. У него где-то в docker history падает entrypoint.sh. Заменил на nginx:1.14.2 – все хорошо

Вот мое приложение, которое отдает мне ingress. Порт 80 проброшен с хоста на ВМ.



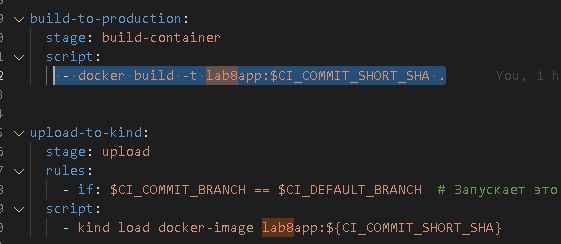
Допы.

1. Настройте CI, чтобы джобы build, upload, deploy запускались только по мастеру ❗ Этих пунктов хватит для сдачи ЛР.
2. Доп задание. Настройте запуск тестов в по открытому МРу

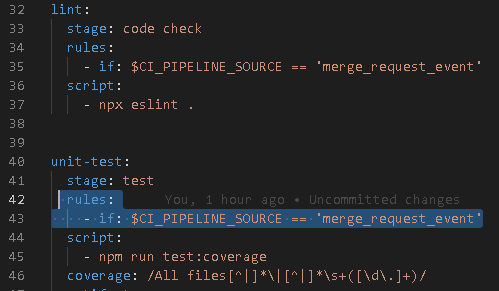
Добавляю тесты линтера и unittest с отчетом из своей ЛР 4.

Отделяю джобу сборки проекта от джобы сборки контейнера. Папка build/ передается через артефакты.

Условие запуска в главной ветке делается так:



Условие запуска при открытии МР делается так:



Джоба билд запускается всегда. Джобы unit-test и lint – при открытии МР. Остальные – в ветке main.

**Выводы:**

Были выполнены все задачи по плану данной лабораторной работы. Соответствующие действия по выполнению задач лабораторной работы были описаны и внесены в настоящий отчет. Получены соответствующие навыки работы с ОС семейства GNU/Linux:

Я успешно настроил виртуальную машину, и изучил основные команды kuber. С помощью CI мне удалось отправить версию приложения на виртуальный кластер.