
Архитектура распределённых приложений

Лабораторная работа №2

Выполнил: студент группы ИП-312
Дорогин Н. С.
Проверил: доцент Кафедры вычислительных систем Романюта А.А.

Задание:

1. Настроить CI для сборки образа и публикации его в реестр образов контейнеров Gitlab.

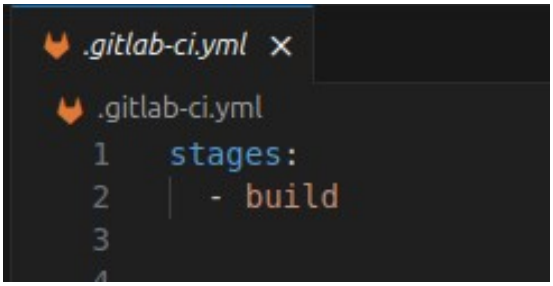
   -- Образ собирать при помощи docker buildx под архитектуры linux/arm64 и linux/amd64.
   -- Сборка должна производиться при отправке коммита или тега в репозиторий. (git push, git push --tags)
   -- При работе с реестром gitlab в CI использовать переменные `CI_REGISTRY*` --- они уже содержат и параметры подключения к реестру и имя образа для вашего репозитория.
   -- Тегом образа должен являться тег системы git (git tag), при его отсутствии --- короткий хэш коммита (Первые 8 символов хэша).
      - Сборка с тега v1, репозиторий test --- образ `test:v1`
      - Сборка с коммита (хэш abcd1234...), репозиторий test --- образ `test:abcd1234`

2. Проверить что собранный образ появился в реестре gitlab.

3. Продемонстрировать запуск (Включая стадию pull) образа из реестра.

Ход работы:

1) Файл `.gitlab-ci.yml`.




```yaml
build-job:

  stage: build

  image: docker:28.1

  services:
    - docker:28.1-dind


  before_script:
    #- docker login ${HAWK_REGISTRY} не работает интерактив
    # - echo "$HAWK_REGISTRY_PASSWORD" | docker login -u "$HAWK_REGISTRY_USER" --password-stdin $HAWK_REGISTRY
    - echo "$CI_JOB_TOKEN" | docker login -u "$CI_REGISTRY_USER" --password-stdin "$CI_REGISTRY"
    - docker context create ci-context
    - docker context use ci-context
    - docker buildx create --name multiarch-builder --driver docker-container --use ci-context

  script:
    - echo "Building..."
    - |
      if [ -n "$CI_COMMIT_TAG" ]; then
      IMAGE_TAG=$CI_COMMIT_TAG
      else
      IMAGE_TAG=${CI_COMMIT_SHORT_SHA:0:8}
      fi
      echo "READING THE IMAGE TAG"

    - docker buildx build --platform linux/amd64,linux/arm64 -t ${CI_REGISTRY_IMAGE}:${IMAGE_TAG} --push .
    - echo "${CI_REGISTRY_IMAGE}:${IMAGE_TAG} WAS SUCCESFULLY BUILT"


  rules:
    - if: $CI_COMMIT_TAG
      when: always
    - when: manual
```

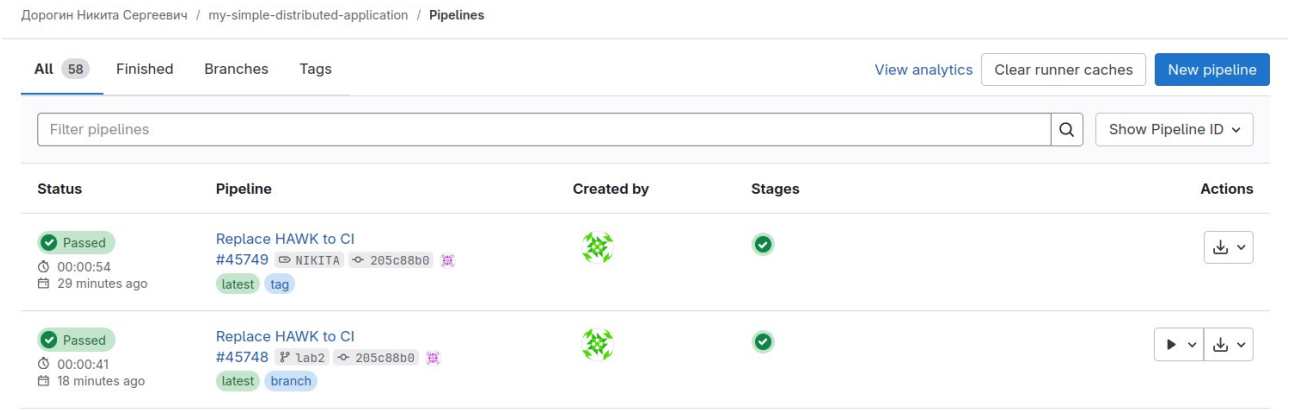
(из истории терминала)

```
1007  git push
1008  git tag -a NIKITA
1009  git tag -a NIKITA -m "TRIGERRED"
1010  git push --tags
```

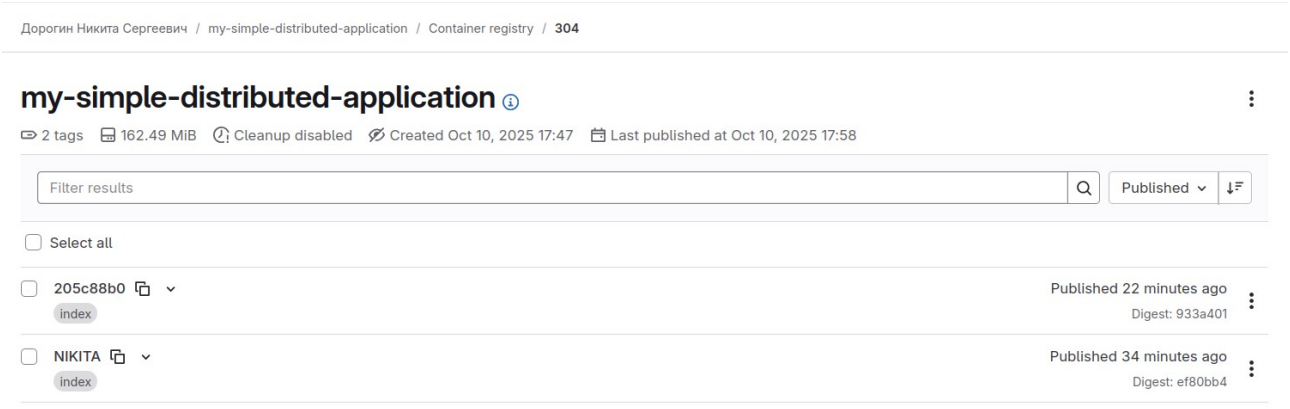
Обычный коммит отправленный на удалённый репозиторий приводит к созданию пайплайна, но он не запускается автоматически. Его можно запустить только вручную.

Отправленный тэг приводит к созданию и автоматическому запуску пайплайна.

Оба пайплайна проходят успешно.



2) Оба образа оказываются в реестре gitlab:



3) Запуск образа из реестра

Загружаем образ с помощью docker pull, предварительно скопировав путь к нему из реестра.

```
1018  sudo docker pull registry.csc.sibsutis.ru/ip312s07/my-simple-distributed-application:NIKITA
1019  sudo docker run -p 7777:5000 registry.csc.sibsutis.ru/ip312s07/my-simple-distributed-application:NIKITA
```

Запускаем контейнер.

```
gastello123@gastello123:~/Desktop/my-simple-distributed-application$ sudo docker run -p 7777:5000 registry.csc.sibsutis.ru/ip312s07/my-simple-distributed-application:NIKITA
[sudo] пароль для gastello123:
info: Microsoft.Hosting.Lifetime[14]
      Now listening on: http://0.0.0.0:5000
info: Microsoft.Hosting.Lifetime[0]
      Application started. Press Ctrl+C to shut down.
info: Microsoft.Hosting.Lifetime[0]
      Hosting environment: Production
info: Microsoft.Hosting.Lifetime[0]
      Content root path: /
```

Переходим по ссылке, меняем порт на тот, что указывали для прослушивания в run, как и в первой лабораторной, и видим, что всё работает: