Непрерывная интеграция и развертывание (CI/CD)

1. Создаем само приложение

2. Создаем юнит тесты для приложения

3. Докеризируем: Создаем докерфайл для сборки контейнера

4. Описываем CI-сценарии для Github Actions производящие lint, unit-test, build-test

5. Куберизируем: Создаем манифесты для запуска приложения и сервисов в кластере

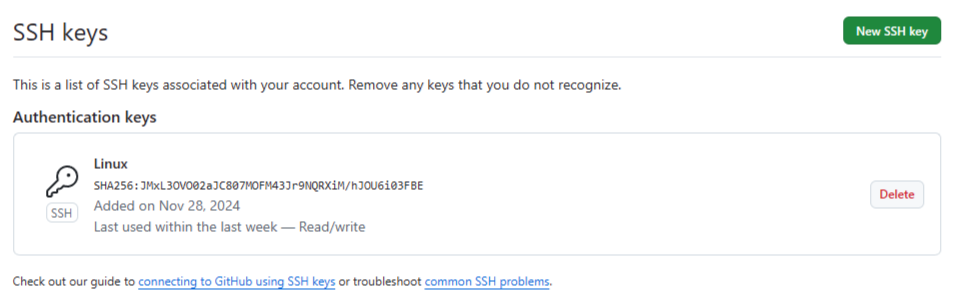
6. Описываем CD-сценарии для Github Actions производящие docker build и push в Docker Hub

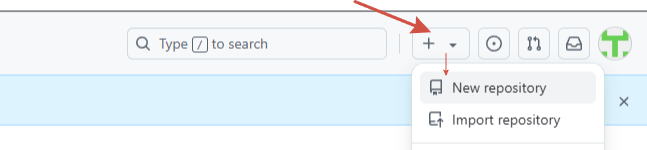
7. Устанавливаем ArgoCD и подключаем к GitHub

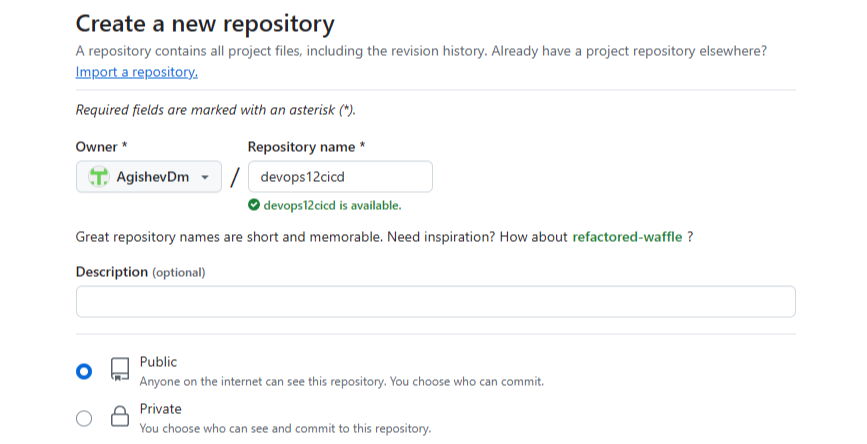
8. Проверяем весь CI/CD workflow в сборе

Целью данной работы являлась настройка CI/CD процесса для простого веб-приложения с использованием GitHub, Docker, и Kubernetes.

Начальная настройка:

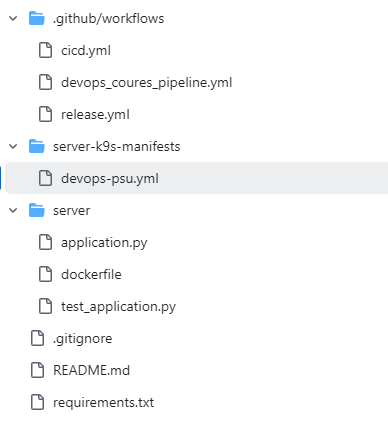
Добавление SSH ключа на GitHub:

Создание репозитория на GitHub: был создан новый репозиторий на GitHub для хранения кода приложения.



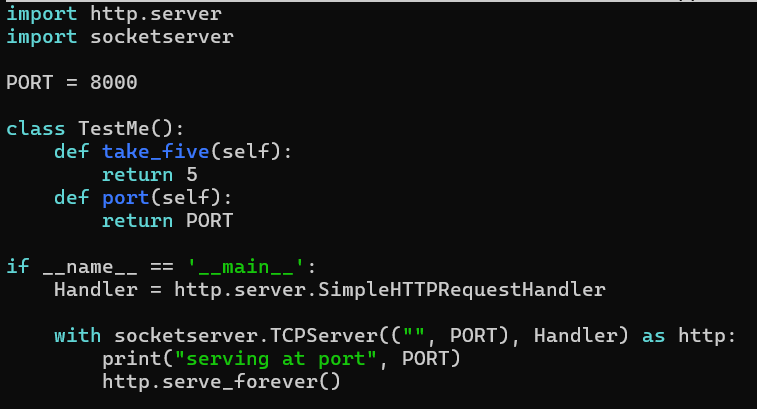
Клонирование репозитория: Репозиторий был склонирован на локальную машину, чтобы можно было работать с кодом.

Структура проекта:

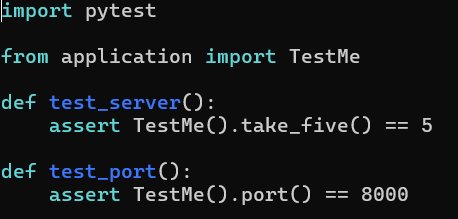


Разработка приложения и начальный коммит:

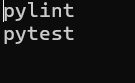
Код приложения: В файл server/application.py был добавлен код веб-приложения, который был протестирован на локальной машине.



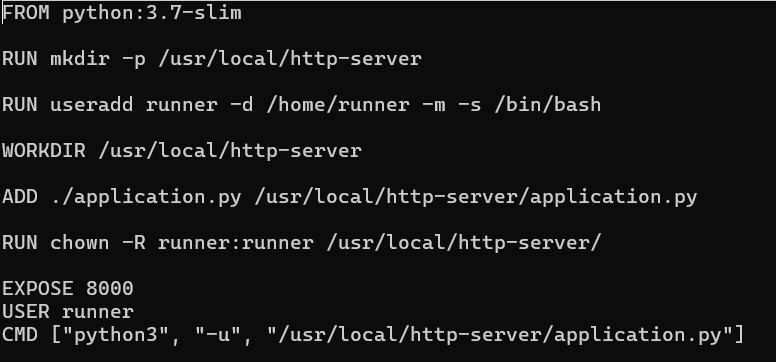
Юнит-тесты: В файл server/test\_application.py были добавлены юнит-тесты для проверки кода приложения.



Файл зависимостей: В файл requirements.txt были добавлены необходимые зависимости для проекта



Docker файл для контейнеризации: В файл dockerfile были добавлены инструкции для сборки Docker-образа приложения.

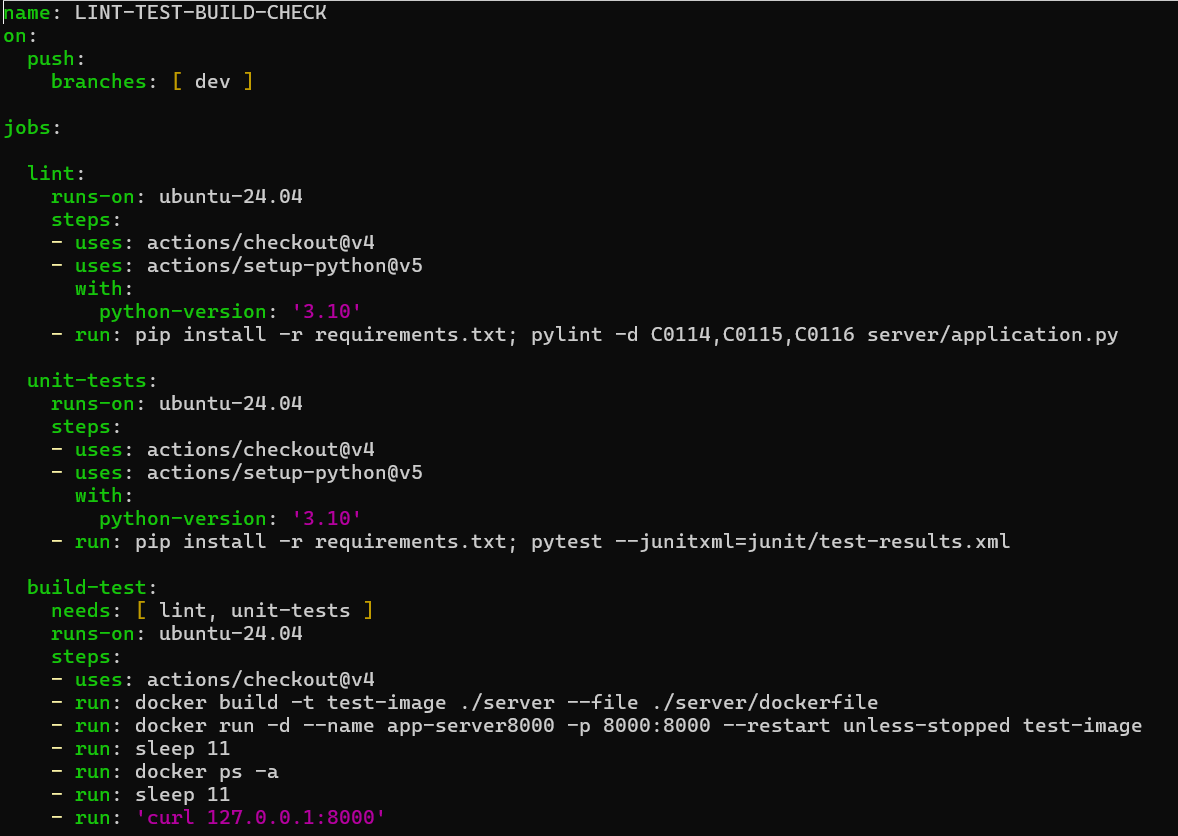


Все созданные файлы были добавлены в git.

Continuous Integration

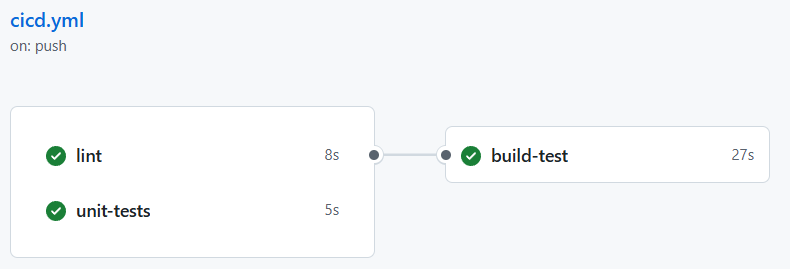
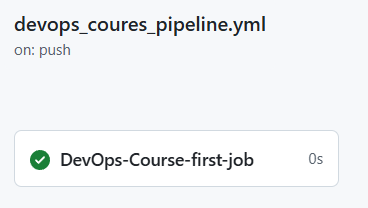
Настройка GitHub Actions: Для автоматизации процесса тестирования был добавлен CI-пайплайн с использованием GitHub Actions.

Файл CI пайплайна: Файл пайплайна был сохранен в .github/workflows/cicd.yml.



Проверка работы CI: После добавления и коммита cicd.yml был запущен пайплайн, который автоматически выполнил юнит-тесты. Успешное выполнение пайплайна было проверено на GitHub.





**Kubernetes и ArgoCD Deployment**

Кубернизация приложения: Для развертывания приложения в Kubernetes, были созданы необходимые K8s манифесты.

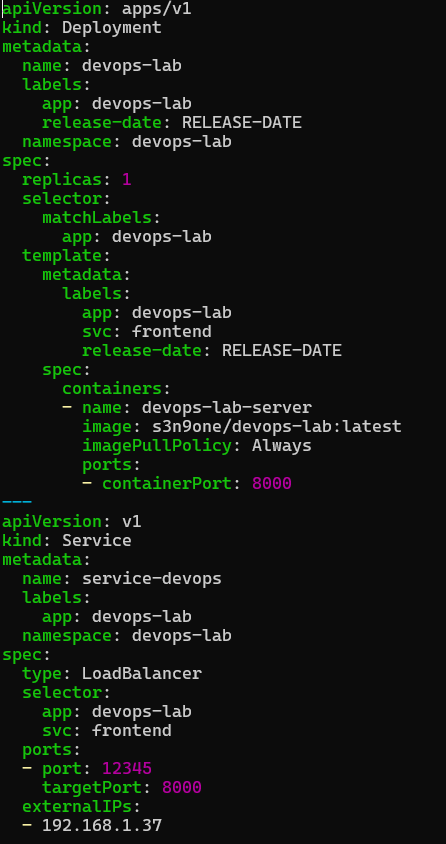
Каталог для манифестов: Манифесты были помещены в каталог server-k8s-manifests.

Файл манифеста: Был создан K8s манифест в файле server-k8s-manifests/devops-lab.yml для деплоя приложения.

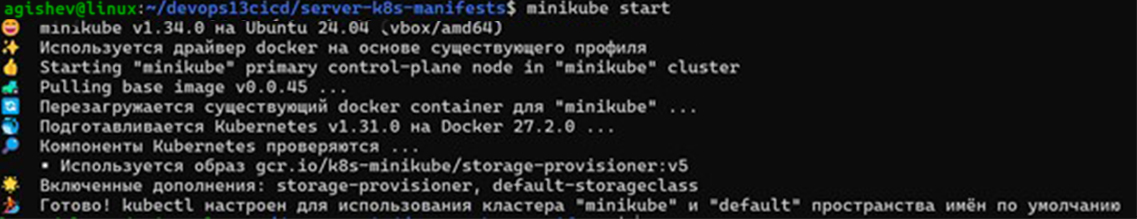


Создание репозитория на Docker Hub: На Docker Hub был создан публичный репозиторий с именем devops-lab.

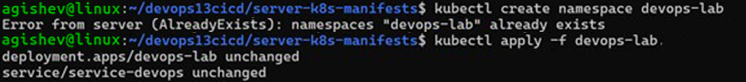
Куберизируем приложение: создаем K8s манифест ./server-k8s-manifests/devops-lab.yml



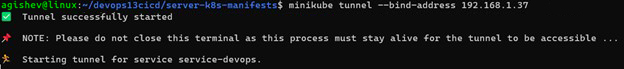
Запуск Minikube: Был запущен minikube на локальной машине для тестирования.



Применение K8s манифеста: Манифест devops-lab.yml был применен к Minikube: kubectl apply -f server-k8s-manifests/devops-lab.yml.



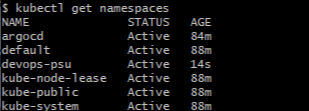
Проброс портов: был настроен проброс портов для сервисов LoadBalancer, чтобы можно было получить доступ к приложению.

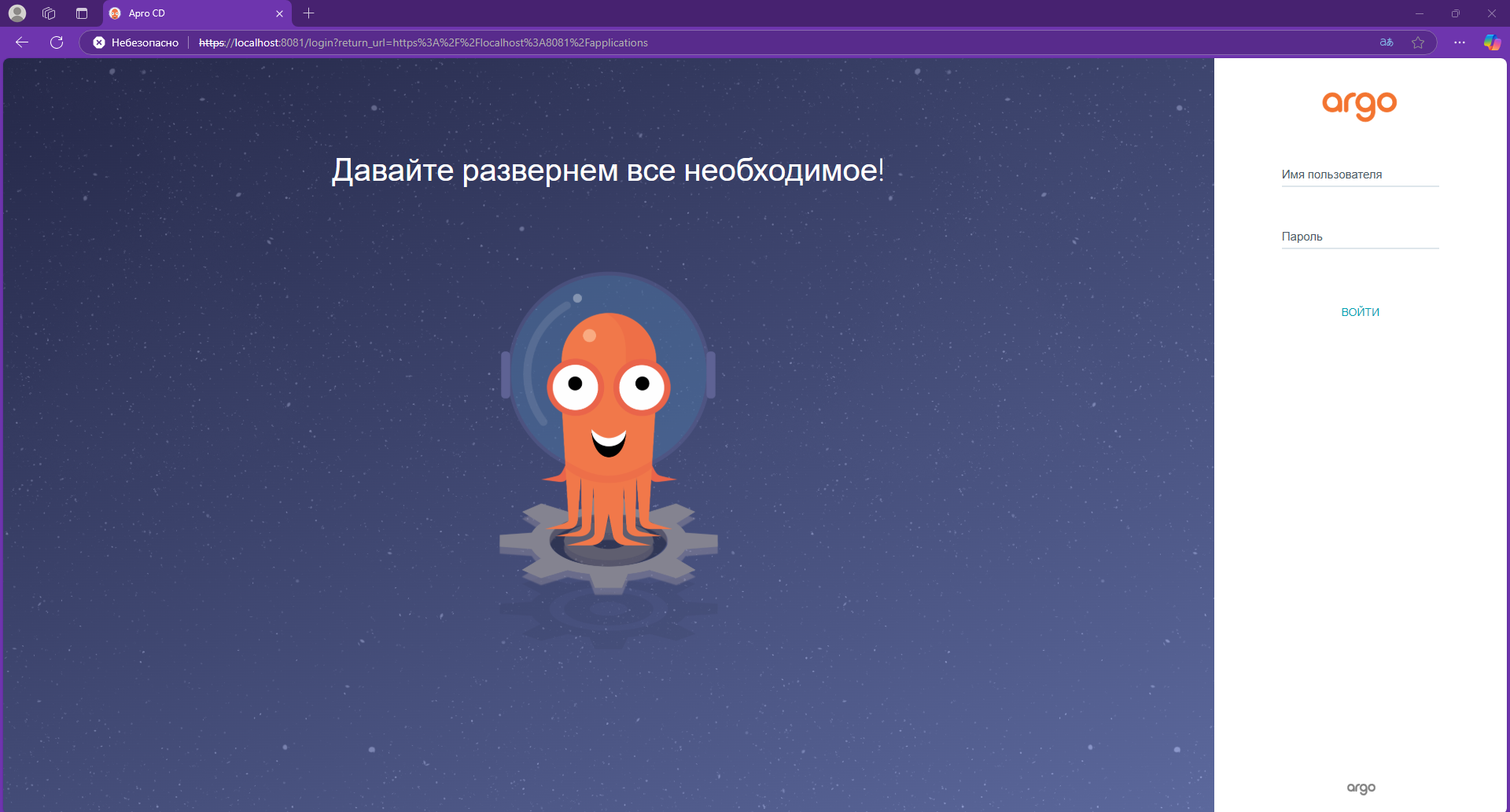


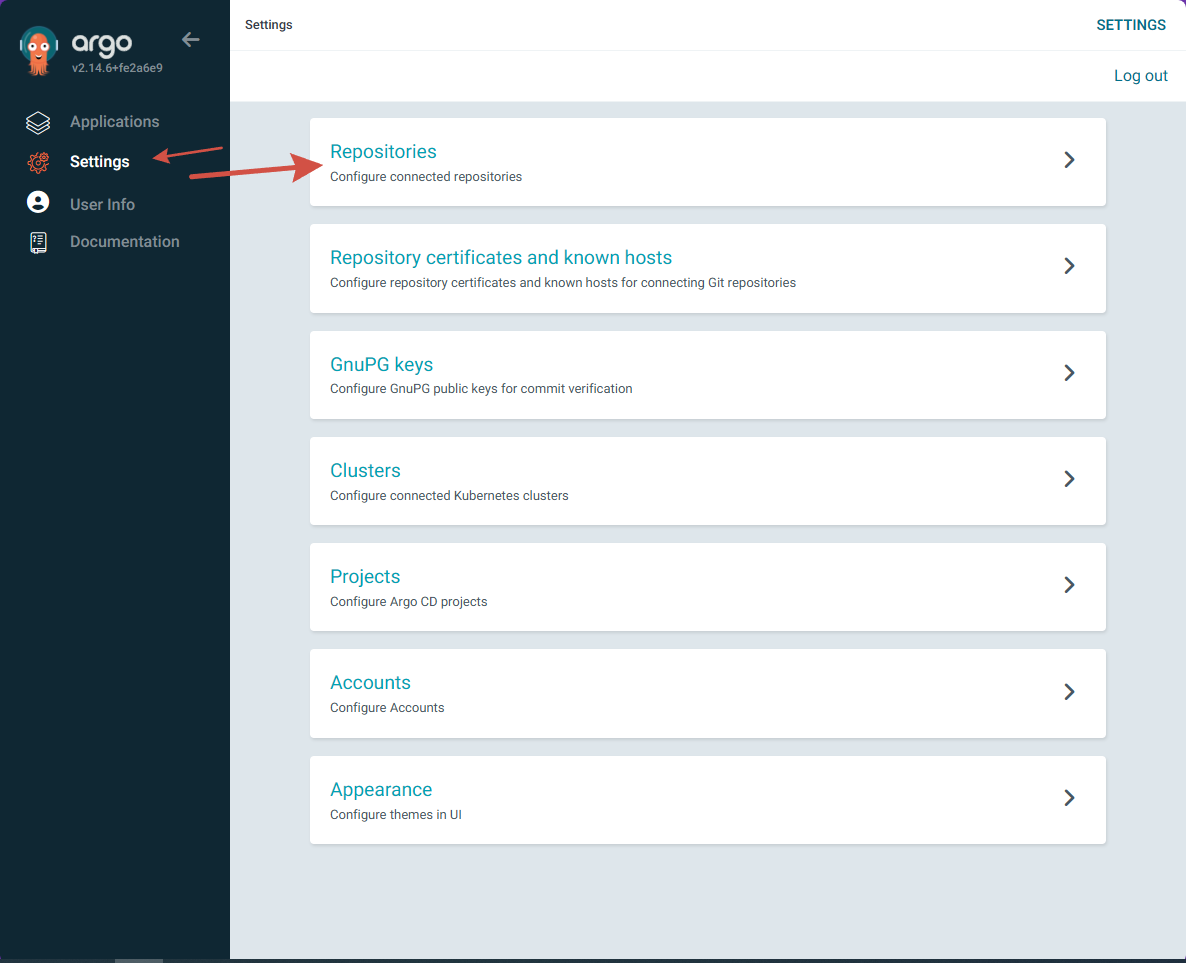
Манифесты были сохранены в репозитории на GitHub.

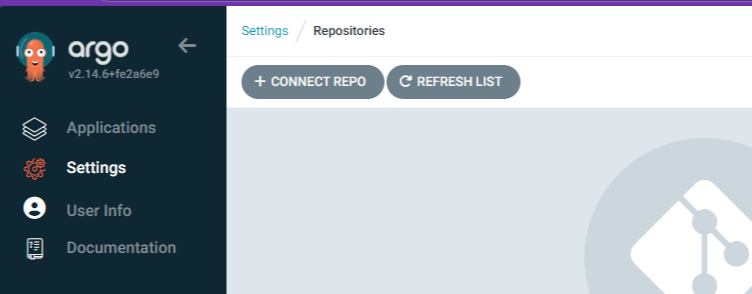


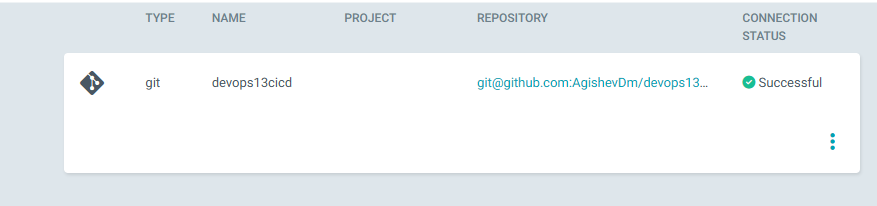


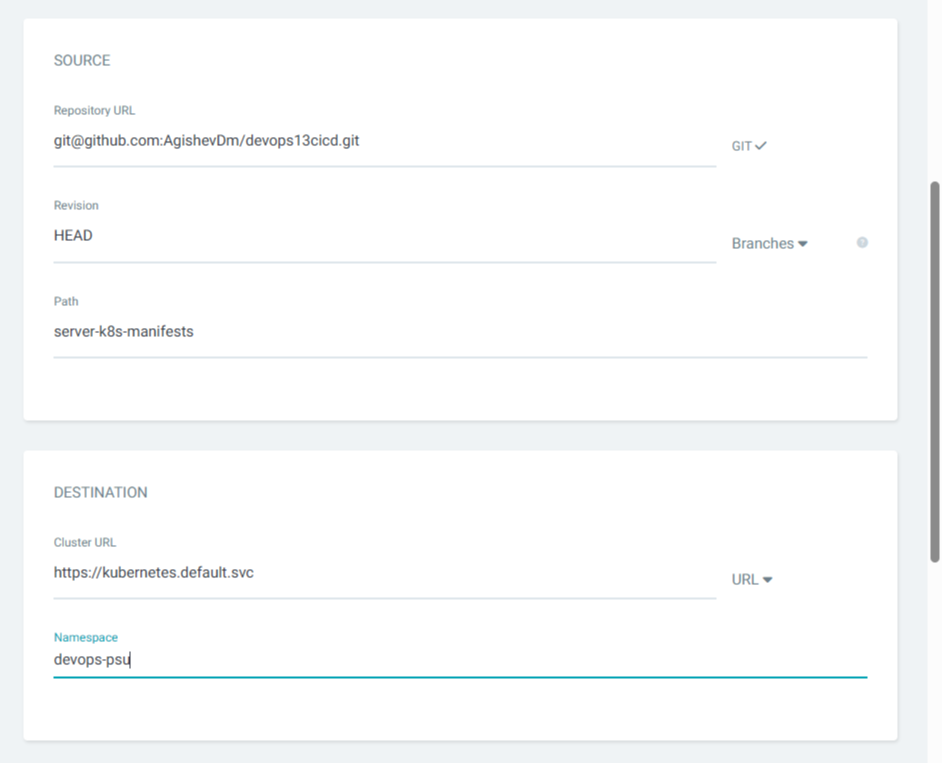


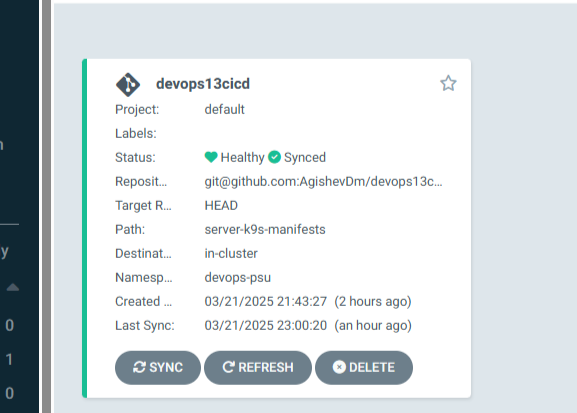






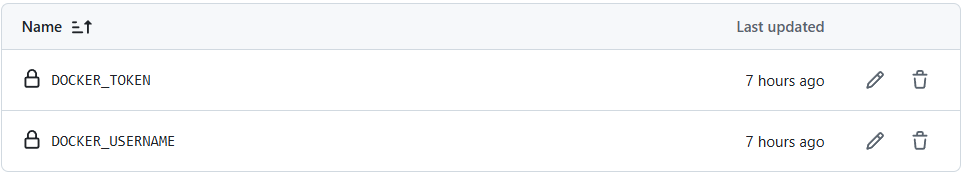






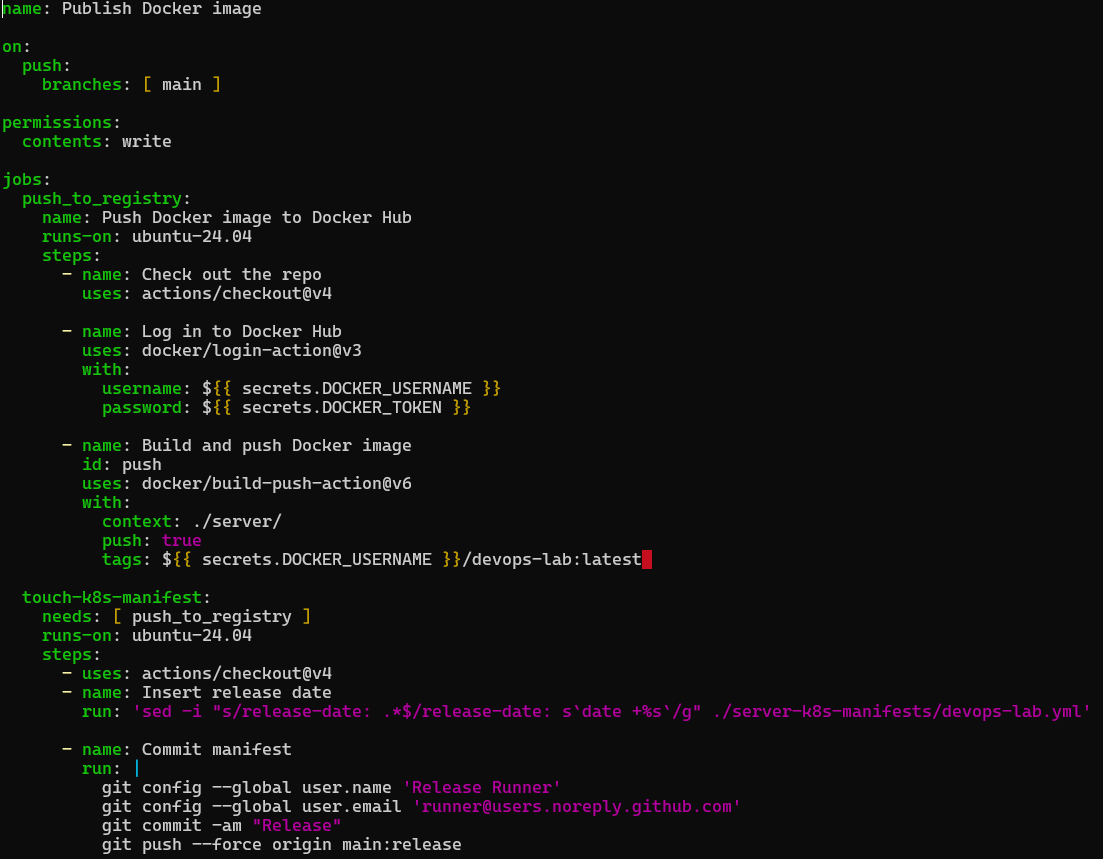
Continuous Delivery

Создание GitHub secrets: В репозитории были созданы два секрета: DOCKER\_USERNAME и DOCKER\_TOKEN для аутентификации в Docker Hub.



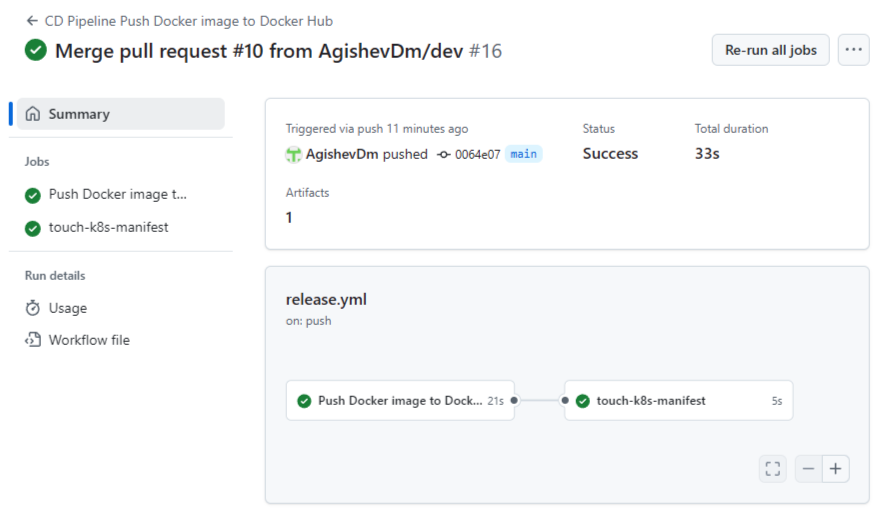
Настройка пайплайна CD: был создан CD-пайплайн для автоматической сборки и публикации Docker обр аза на Docker Hub при слиянии в main ветку.

Файл CD пайплайна: Пайплайн был сохранен в файле .github/workflows/release.yml.

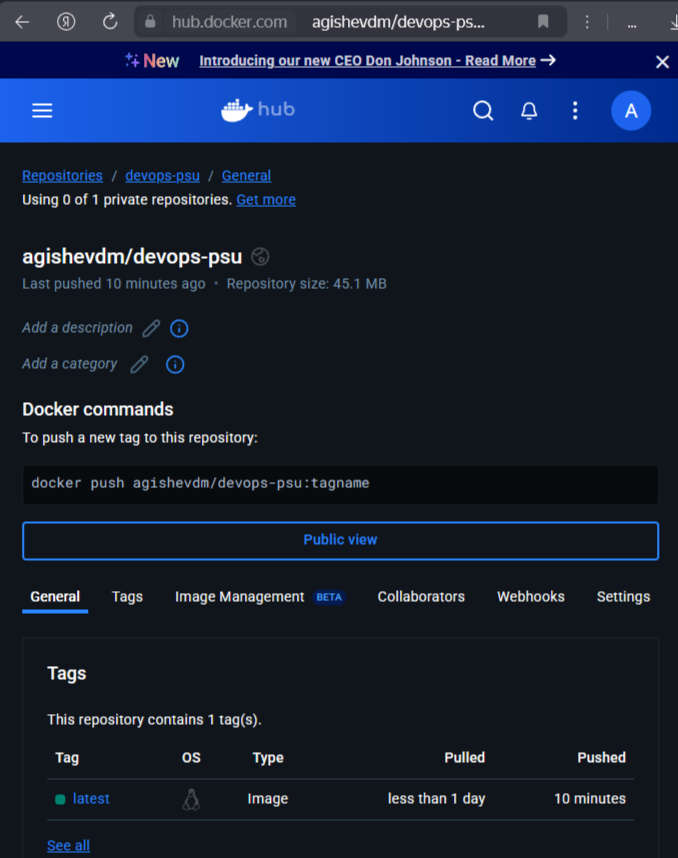


Коммит и Pull Request: Файл release.yml был закоммичен и отправлен в репозиторий. Был создан Pull request из ветки dev в ветку main.

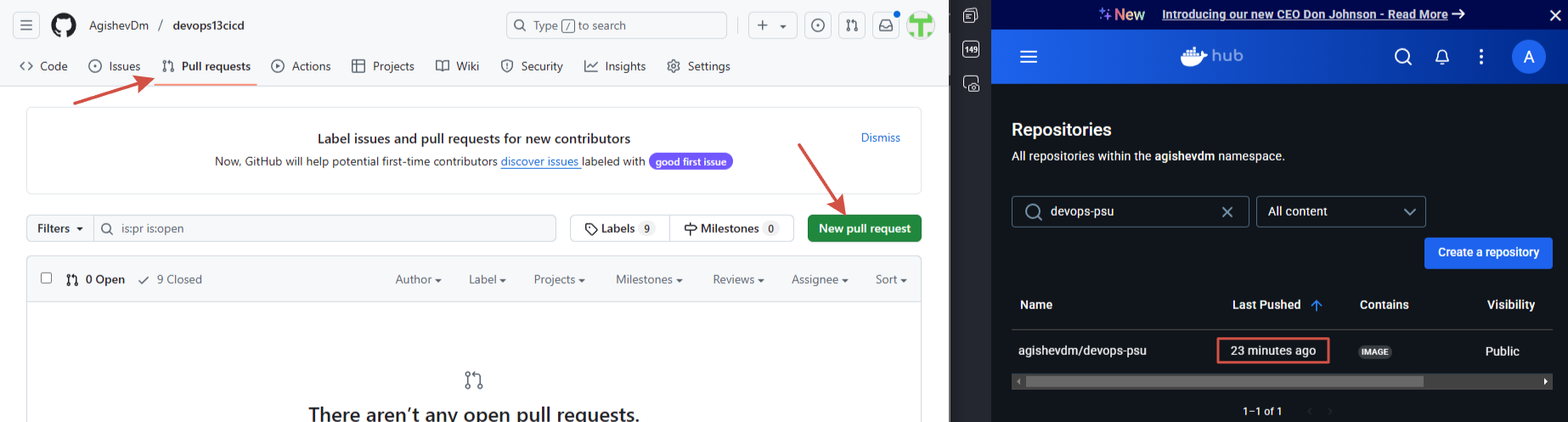
Проверка CD пайплайна: После слияния в ветку main был запущен CD-пайплайн.

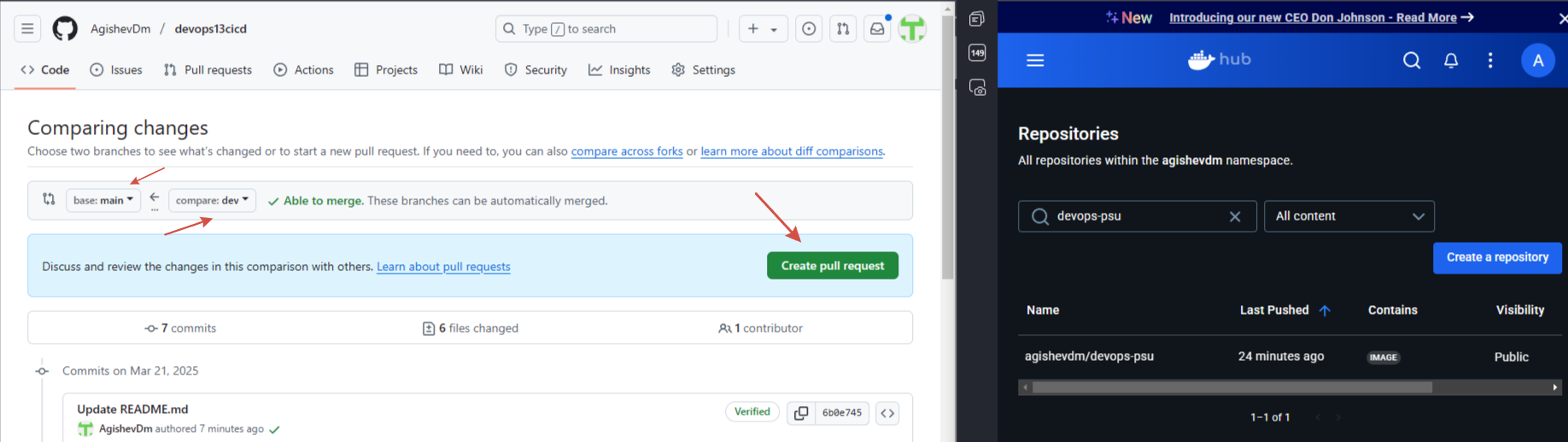


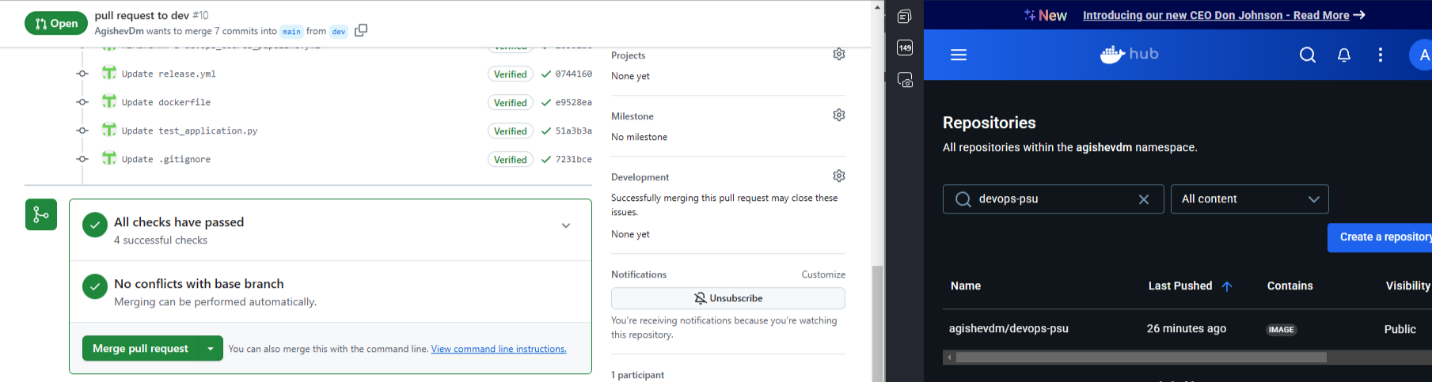
Проверка Docker Hub: Проверено, что новый Docker-образ был успешно опубликован в Docker Hub.

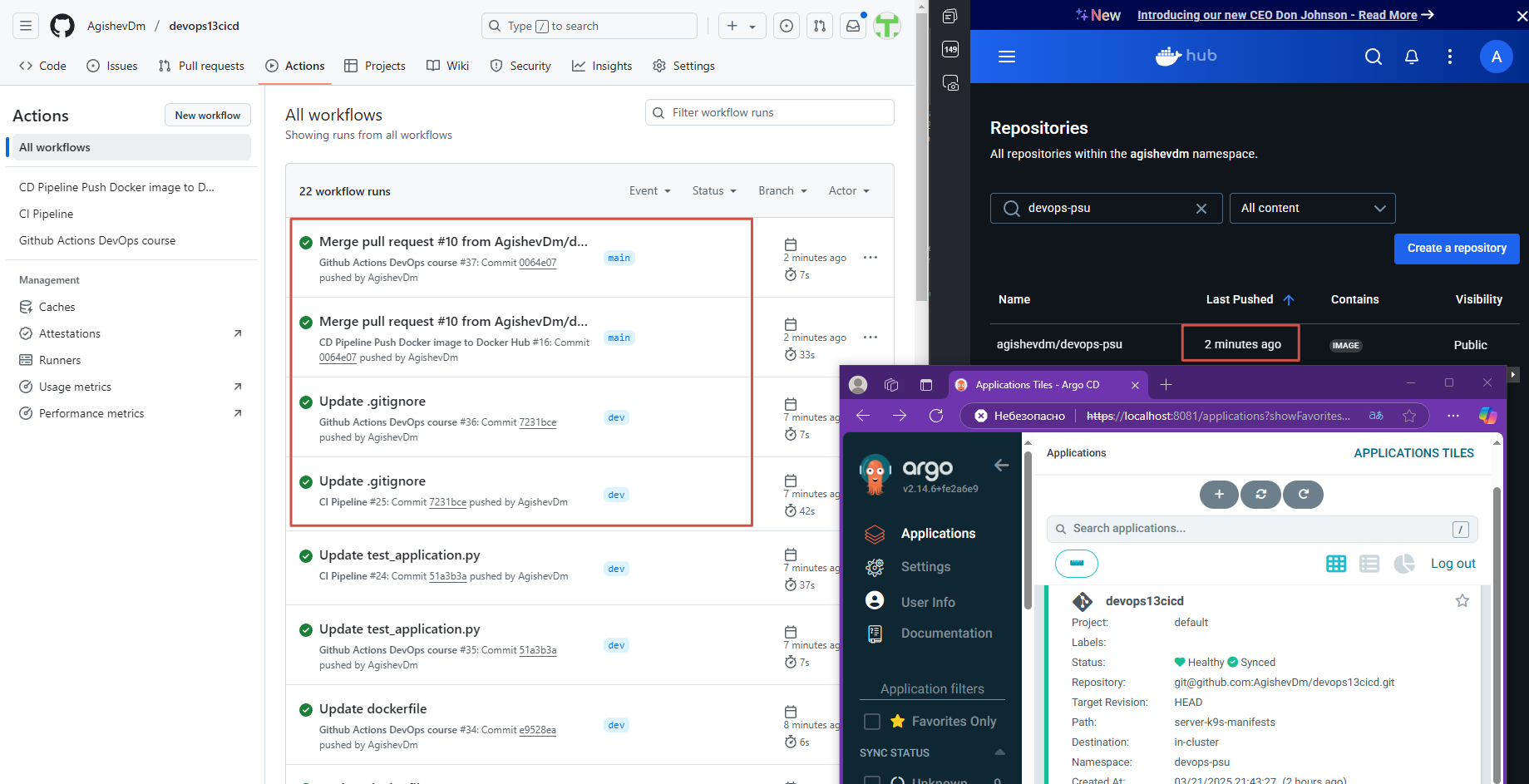


Финальная проверка работы всех настроенных компонентов





1. 



Как видим все работает исправно, все этапы CI/CD настроены корректно. Приложение автоматически тестируется, собирается, публикуется и разворачивается в Kubernetes.