Практика CI/CD

Краткое описание

Данный репозиторий содержит результаты практической работы по настройке CI/CD конвейера для автоматического тестирования, сборки и развертывания приложения на облачном сервере.

Начало работы

Склонировать репозиторий и перейти в директорию `blue_green`

```
git clone https://github.com/arsenier/cicd-task.git
cd cicd-task
```

В файле `scp_to_serv.sh` указать актуальные SSH-ключ и адрес сервера где необходимо развернуть приложение, после чего запустить скрипт (либо просто перекинуть все файлы в катало `/root/` на сервере)

```
./scp_to_serv.sh
```

Подключится к серверу по SSH и выполнить скрипты инициализации:

```
./server_init.sh
./bg_init.sh
```

PROF

После чего по адресу сервера должна быть доступна веб страница с развернутым приложением.

Обновление приложения

Для обновления приложения реализован метод blue-green развертывания. Чтобы обновить приложение с помощью нового докер-образа достаточно выполнить скрипт `bg_switch.sh`

```
./bg_switch.sh
```

Скрипт проверяет цвет запущенного образа, пуллит новый образ с DockerHub, запускает его и переключает настройки nginx-прокси. После чего выключает старый цвет.

Остановка сервера

Для остановки сервера нужно выполнить скрипт:

```
./bg_shutdown.sh
```

Структура проекта

В качестве приложения для развертывания был взят пример Python-приложения из урока: https://www.youtube.com/watch?v=8gtEtEY0ofM

Ниже объясняется файловая структура, имеющая отношение непосредственно к CI/CD, игнорируя файлы самого приложения:

```
cicd-task/
├── .github/
│   └── workflows/
│       └── ci_cd.yml # Файл пайплайна CI/CD с использованием Github
Actions
├── blue_green/ # Набор скриптов для Blue-Green развертывания приложения
│   ├── bg_init.sh
│   ├── bg_shutdown.sh
│   ├── bg_switch.sh
│   ├── docker-compose.yml
│   ├── scp_to_serv.sh
│   └── server_init.sh
├── nginx/ # Конфигурации nginx прокси для Blue-Green
│   ├── Dockerfile
│   ├── nginx_blue.conf
│   └── nginx_green.conf
...
```

PROF

CI/CD

Файл пайплайна GitHub Actions `./.github/workflows/ci_cd.yml` описывает процесс CI/CD для проекта. Вот краткое описание его структуры и шагов:

Основные компоненты:

1. **Имя и триггеры**:
   - Пайплайн называется `CI/CD`.
   - Он запускается при `push` в ветку `master` и может быть инициирован вручную через `workflow_dispatch`.
2. **Разрешения**:
   - Устанавливаются разрешения для чтения содержимого и записи токена идентификации.

3. **Работы (jobs)**:

    - **run_tests**:

        - Запускается на `ubuntu-latest`.
        - Шаги:
            - Проверка кода из репозитория.
            - Установка Python версии 3.12.
            - Установка утилиты `make`.
            - Запуск тестов с помощью команды `make test`.

    - **publish_image**:

        - Запускается на `ubuntu-latest` и зависит от успешного завершения `run_tests`.
        - Шаги:
            - Проверка кода из репозитория.
            - Вход в Docker Hub с использованием секретов.
            - Извлечение метаданных для Docker (теги, метки).
            - Сборка и публикация Docker-образа.

    - **publish_nginx**:

        - Запускается на `ubuntu-latest`.
        - Шаги аналогичны `publish_image`, но для Nginx-образа, с указанием контекста и Dockerfile для Nginx.

    - **deploy_application**:

        - Запускается на `ubuntu-latest` и зависит от успешного завершения `publish_image` и `publish_nginx`.
        - Шаги:
            - Вход на удаленный сервер через SSH и выполнение скрипта для развертывания образа.

PROF Этот файл пайплайна обеспечивает автоматизацию тестирования, сборки и развертывания приложения, что упрощает процесс CI/CD.