**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

**Дисциплина: Бек-энд разработка**

Отчет

Домашнее задание №6: Автодеплой приложения на удалённый сервер с использованием Github Actions

Выполнила:

Пронина Александра

K33392

Проверил:

Добряков Д. И.

Санкт-Петербург

2024 г.

**Задание:**

Необходимо настроить автодеплой (с триггером на обновление кода в вашем репозитории, на определённой ветке) для вашего приложения на удалённый сервер с использованием Github Actions или Gitlab CI (любая другая CI-система также может быть использована).

Шаги для настройки автодеплоя:

**1.** Настройка репозитория

Убедимся, что репозиторий содержит необходимый код и конфигурационные файлы для развертывания. Например, Dockerfile, docker-compose.yml, или скрипты для деплоя.

**2**. Подготовка удаленного сервера:

Я арендовала на квоту Yandex Cloud(IP скрыт).

Изображение выглядит как текст, Шрифт, снимок экрана

Автоматически созданное описание

Установим необходимое ПО, такое как Docker, ssh, и другие зависимости для приложения.

Настроим доступ по SSH для безопасного соединения CI/CD системы с сервером.

Создание SSH-ключей:

После создания директории (если её не было) повторим команду для создания ключей:

Изображение выглядит как текст, электроника, снимок экрана, программное обеспечение

Автоматически созданное описание

Изображение выглядит как текст, снимок экрана, Шрифт, информация

Автоматически созданное описание

Затем добавим этот ключ в файл ~/.ssh/authorized\_keys на сервере. Для этого выполним через PowerShell следующие шаги:

Подключаемся к серверу по SSH (если уже есть доступ):

ssh <тут юзер>@<тут api сервера>

Создадим директорию .ssh, если она не существует:

mkdir -p ~/.ssh

Для дальнейшего выполненния у меня было 2 варианта (но я выбрала первый из-за удобства в частном случае):

Вариант 1: добавим публичный ключ в файл authorized\_keys:

echo " " >> ~/.ssh/authorized\_keys

Вариант 2: Или с использованием SCP:

scp C:\Users\Елена\.ssh\id\_rsa.pub <тут юзер>@<тут api сервера>:~/.ssh/temp\_id\_rsa.pub

ssh <тут юзер>@<тут api сервера>: ~/.'cat ~/.ssh/temp\_id\_rsa.pub >> ~/.ssh/authorized\_keys && rm ~/.ssh/temp\_id\_rsa.pub'Изображение выглядит как текст, снимок экрана, Шрифт, программное обеспечение

Автоматически созданное описаниеИзображение выглядит как текст, снимок экрана, программное обеспечение, дисплей

Автоматически созданное описание

Добавим приватный ключ в GitHub Secrets:

Откроем репозиторий на GitHub и выполним следующие шаги:

1.Перейдём в раздел Settings.

2.Выберите Secrets and variables > Actions.

3.Нажмём на кнопку New repository secret.

4. Добавим следующие секреты:

SSH\_PRIVATE\_KEY: Содержимое файла C:\Users\Елена\.ssh\id\_rsa.Для этого откроем файл id\_rsa и скопируем его содержимое:

type C:\Users\Елена\.ssh\id\_rsa – в cmd в роли админмстратора

SSH\_HOST: IP-адрес сервера.

SSH\_USER: Имя пользователя для подключения через SSH.

**3.** Настроим workflow в GitHub Actions, добавим команды для перезапуска контейнеров

Создади файл workflow в репозитории (например, .github/workflows/deploy.yml):

SSH\_PRIVATE\_KEY: Содержимое файла C:\Users\Елена\.ssh\id\_rsa.

Изображение выглядит как текст, снимок экрана, Шрифт

Автоматически созданное описание

SSH\_HOST: IP-адрес вашего сервера. (скрыто)

SSH\_USER: Имя пользователя для подключения через SSH. (скрыто)

**4.** Настройка workflow в GitHub Actions:

Далее создали файл workflow в репозитории (.github/workflows/deploy.yml):

SSH\_PRIVATE\_KEY, SSH\_HOST, secrets.SSH\_USER – секретики для деплоя на сервер.

Содержимое: name: Deploy  
  
on:  
 push:  
 branches:  
 - main *# укажем нужную ветку*jobs:  
 deploy:  
 runs-on: ubuntu-latest  
  
 steps:  
 - name: Checkout repository  
 uses: actions/checkout@v2  
  
 - name: Set up SSH  
 uses: webfactory/ssh-agent@v0.5.3  
 with:  
 ssh-private-key: ${{ secrets.SSH\_PRIVATE\_KEY }}  
  
 - name: Deploy to server  
 env:  
 SSH\_HOST: ${{ secrets.SSH\_HOST }}  
 SSH\_USER: ${{ secrets.SSH\_USER }}  
 run: |  
 ssh -o StrictHostKeyChecking=no $SSH\_USER@$SSH\_HOST << 'EOF'  
 cd /path/to/app  
 git pull  
 docker-compose down  
 docker-compose up -d  
 EOF

Теперь, при каждом коммите в указанную ветку (в данном случае main), GitHub Actions будет автоматически запускать деплой приложения на удалённый сервер.

**Вывод:** В процессе выполнения лабораторной работы была настроена автоматическая развертка приложения на удаленный сервер с использованием GitHub Actions. Эта развертка срабатывает при обновлении кода в репозитории, определенной на определенной ветке.