

**САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ
ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО**

Дисциплина: Бэк-энд разработка

Отчет

Лабораторная работа №6

Выполнил:

Пиотуховский Александр

К3441

Проверил:

Добряков Д. И.

Санкт-Петербург

2026 г.

Задача

Необходимо настроить автодеплой (с триггером на обновление кода в вашем репозитории, на определённой ветке) для вашего приложения на удалённый сервер с использованием Github Actions.

Ход работы

1. Настройка окружения и безопасности CI/CD

Для реализации автодеплоя приложения был выбран инструмент GitHub Actions. Поскольку проект размещается в публичном репозитории, критически важные данные, такие как IP-адрес сервера, SSH-ключи доступа и переменные окружения, не могут храниться в открытом виде в коде.

Для решения этой проблемы был использован механизм GitHub Secrets. Переменные окружения для сервисов mail_service и backend, а также данные для SSH-подключения были зашифрованы и сохранены в настройках репозитория (рисунок 1).

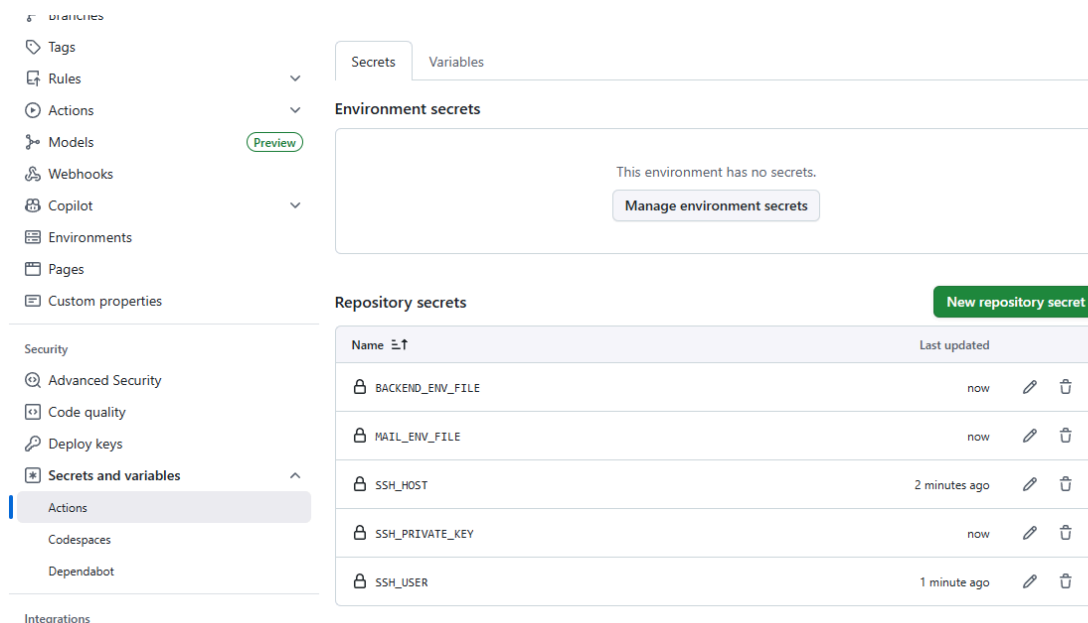


Рисунок 1 – Настройка секретов в репозитории

2. Конфигурация сценария деплоя

В корне репозитория был создан файл конфигурации `.github/workflows/deploy.yml`. Сценарий настроен на автоматический запуск при обновлении ветки `lr6`.

Пайплайн подключается к удалённому серверу по SSH, обновляет кодовую базу из репозитория, генерирует актуальные `.env` файлы из секретов `github` и последовательно перезапускает контейнеры. Фрагмент конфигурации, отвечающий за деплой, представлен на рисунке 2.

```

name: Production Deploy

on:
  push:
    branches:
      - lr6

jobs:
  deploy:
    name: Deploy to Server
    runs-on: ubuntu-latest
    steps:
      - name: Checkout Repository
        uses: actions/checkout@v3

      - name: Setup SSH Key
        uses: webfactory/ssh-agent@v0.8.0
        with:
          ssh-private-key: "${{ secrets.SSH_PRIVATE_KEY }}"

      - name: Add Known Hosts
        run: |
          mkdir -p ~/.ssh
          ssh-keyscan -H "${{ secrets.SSH_HOST }}" >> ~/.ssh/known_hosts

      - name: Deploy Mail Service
        run: |
          ssh -o StrictHostKeyChecking=no "${{ secrets.SSH_USER }}"@${{ secrets.SSH_HOST }} << 'EOF'
          cd /opt/film_app/mail_service
          git pull origin main
          echo "${{ secrets.MAIL_ENV_FILE }}" > .env

          docker-compose down
          docker-compose up -d --build
          EOF

      - name: Deploy Backend App
        run: |
          ssh -o StrictHostKeyChecking=no "${{ secrets.SSH_USER }}"@${{ secrets.SSH_HOST }} << 'EOF'
          cd /opt/film_app/backend
          git pull origin main
          echo "${{ secrets.BACKEND_ENV_FILE }}" > .env

```

Рисунок 2 – Фрагмент конфигурации GitHub Actions для деплоя

3. Результаты автоматического деплоя

После внесения изменений в ветку `lr6` процесс деплоя был инициирован автоматически. На рисунке 3 представлен лог успешного выполнения пайплайна.

The screenshot displays the GitHub Actions interface for a workflow run titled "another fix deploy workflow #5". The workflow is in a "Production Deploy" environment. The left sidebar shows the workflow steps: "Summary", "All jobs", and "Deploy to Server" (which is selected and marked as successful). Below the sidebar, there are links for "Run details", "Usage", and "Workflow file". The main panel shows the details of the "Deploy to Server" job, which succeeded 1 minute ago in 11s. The job steps are listed as follows:

- > ☒ Set up job
- > ☒ Checkout Repository
- > ☒ Setup SSH Key
- > ☒ Add Known Hosts
- > ☒ Deploy Mail Service
- > ☒ Deploy Backend App
- > ☒ Post Checkout Repository
- > ☒ Complete job

The "Deploy Mail Service" step is expanded, showing the following log output:

```
1 ▶ Run echo "Connecting to remote host..."
24 Connecting to remote host...
25 cd /opt/project/mail_service
26 git pull origin main
27 Already up to date.
28 Writing .env file...
29 Running docker-compose down...
30 Stopping mail_service_app ... done
31 Removing mail_service_app ... done
32 Removing network mail_net ... done
33 Running docker-compose up -d --build...
34 Creating network 'mail_net' with driver 'bridge'
35 Building mail_service...
36 Step 1/7 : FROM node:18-alpine
37 ----> b3e16sc2
38 Successfully built b3e16sc2
39 Creating mail_service_app ... done
```

Рисунок 3 – Лог успешного выполнения деплоя и настройки сети

Вывод

В ходе выполнения лабораторной работы был настроен конвейер CI/CD для автоматического деплоя микросервисной архитектуры на удалённый сервер. Была решена задача безопасной передачи переменных окружения через Github Secrets, а также реализована корректная оркестрация контейнеров с учётом зависимостей общей сети между независимыми docker-compose проектами.