Министерство образования Республики Беларусь

«ПОЛОЦКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИМ. ЕВФРОСИНИИ ПОЛОЦКОЙ»

Факультет информационных технологий

Кафедра технологий программирования

Методические указания для выполнения лабораторной работы №19 по курсу «Веб-технологии»

«Развертывание приложения в облаке и настройка CI/CD»

Полоцк, 2025 г.

ЦЕЛЬ РАБОТЫ

Целью данной лабораторной работы является изучение методологии CI/CD (Continuous Integration/Continuous Deployment) как ключевого элемента DevOps, получение практических навыков по настройке автоматизированного конвейера развертывания NestJS-приложения в облачной среде и освоение инструментов CI/CD (например, GitHub Actions) для автоматизации рутинных задач.

ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

CI/CD, что расшифровывается как непрерывная интеграция и непрерывная доставка/развертывание, представляет собой набор практик, направленных на автоматизацию и ускорение процесса разработки и доставки программного обеспечения.

Непрерывная интеграция (Continuous Integration, CI) – это практика, при которой разработчики часто и регулярно интегрируют свой код в общую ветку репозитория. При каждом слиянии автоматически запускается конвейер (pipeline) CI, который выполняет сборку проекта и запускает автоматические тесты (юнит-тесты, интеграционные тесты). Основная цель CI – как можно раньше выявлять ошибки интеграции и конфликты, когда их исправление обходится значительно дешевле и не приводит к долгим, болезненным циклам отладки.

Непрерывная доставка (Continuous Delivery, CD) и **непрерывное развертывание (Continuous Deployment)** являются логическим продолжением CI, но с одним ключевым отличием.

Continuous Delivery: предполагает, что изменения в коде, прошедшие CI-пайплайн, автоматически готовы к развертыванию в продакшене, но сам деплой требует ручного подтверждения.

Continuous Deployment: это полностью автоматизированный процесс, где изменения, успешно прошедшие все тесты, автоматически развертываются в продакшене без какого-либо ручного вмешательства.

Одной из самых популярных платформ для автоматизации этих процессов является GitHub Actions. Это мощная система автоматизации, которая позволяет создавать рабочие процессы (workflows), определяемые в YAML-файлах. Рабочий процесс запускается по определенным событиям (triggers), таким как push в репозиторий или pull_request. Каждый рабочий процесс состоит из одного или нескольких заданий (jobs), которые, в свою очередь, включают последовательность шагов (steps).

Для развертывания NestJS-приложений в облачных средах (таких как Heroku, Google Cloud Run) приложение должно быть адаптировано. Например, вместо жестко

заданного порта (например, 3000) приложение должно слушать динамически назначаемый порт, который облачный провайдер передает через переменную окружения PORT. В некоторых PaaS-средах, например, в Heroku, также требуется создание файла Procfile, который указывает, как запускать приложение.

Использование CI/CD-пайплайнов поднимает важный вопрос безопасности. Для развертывания в облачных средах необходимы конфиденциальные учетные данные (логины, пароли, API-ключи). Коммитить эти данные в репозиторий, даже если он приватный, крайне небезопасно. GitHub Actions, как и другие CI/CD-системы, предоставляет механизм **хранения секретов** (secrets). Эти переменные доступны только внутри пайплайна и позволяют безопасно автоматизировать процесс развертывания, не раскрывая конфиденциальную информацию.

Этап CI/CD	Описание	Ключевая цель	Используемые инструменты
Сборка	Транспиляция исходного кода, установка зависимостей, создание исполняемого файла или Docker-образа.	Подготовить артефакт для дальнейшего использования.	npm install, npm run build, docker build
Тестирование	Автоматический запуск юнит- и интеграционных тестов.	Обеспечить качество и стабильность кода, выявить ошибки на ранних этапах.	npm test
Развертывание	Автоматическая или полуавтоматическая доставка готового артефакта в целевую среду (тестовая, продакшн).	Обеспечить быструю и предсказуемую доставку новых функций.	docker push, gcloud run deploy, ssh

Варианты заданий

1. Подготовка приложения к деплою.

В приложении из ЛР №18 настроить прослушивание порта, используя process.env.PORT или дефолтное значение 3000.

Создать Procfile в корне проекта с командой для запуска продакшн-версии приложения.

Подготовить package.json для продакшн-сборки (npm run build).

2. Настройка базового CI/CD-пайплайна в GitHub Actions.

Создать файл .github/workflows/deploy.yml.

Настроить workflow, который запускается при push в ветку main.

Пайплайн должен включать следующие шаги:

- Получение исходного кода (actions/checkout).
- Настройка среды Node.js (actions/setup-node).
- Установка зависимостей (npm install).
- Сборка приложения (npm run build).
- Сборка Docker-образа из оптимизированного Dockerfile из ЛР №18.
- Отправка (push) образа в контейнерный реестр (например, Docker Hub).
- Развертывание приложения на облачной платформе (например, Heroku).

3. Добавление автоматического тестирования в CI/CD.

Дополнить CI/CD-пайплайн из Задания 2.

Добавить шаг для запуска тестов (npm run test) после установки зависимостей, но до сборки приложения.

Настроить пайплайн таким образом, чтобы он останавливался и сигнализировал об ошибке, если тесты не пройдены.

Объяснить, почему этот шаг является критически важным для методологии CI.

Контрольные вопросы

1. Объясните основные принципы и преимущества CI/CD.

2. В чем разница между непрерывной интеграцией, непрерывной доставкой и непрерывным развертыванием?

3. Опишите роль GitHub Actions в процессе CI/CD. Каковы ключевые компоненты файла workflow.yml?

4. Как следует подготавливать NestJS-приложение к развертыванию в облачной среде? Какие параметры необходимо учитывать (например, порт, Procfile)?

5. Объясните, почему важно использовать "секреты" (secrets) в пайплайнах CI/CD.