Курс DevOps

Лабораторная работа №1

GitHub webhooks

1 апреля 2026 г.

Общие инструкции

Рекомендации к выполнению

- Настоятельно рекомендуется выполнять команды из листинга отдельно, одну за другой.
- Рекомендуется копировать команды через промежуточную вставку в текстовый редактор во избежание лишних символов.
- Перед началом работы ознакомтесь с инструкцией по сдаче лабораторных работ.

Требования к выполняемой работе

- Добавьте в репозиторий все необходимые файлы: код приложения, файлы конфигурации и скрипты и т.д.

1 Цель работы

Научиться использовать GitHub webhooks для решения задач автоматизации сборки приложения и его развертывания.

2 Содержание работы

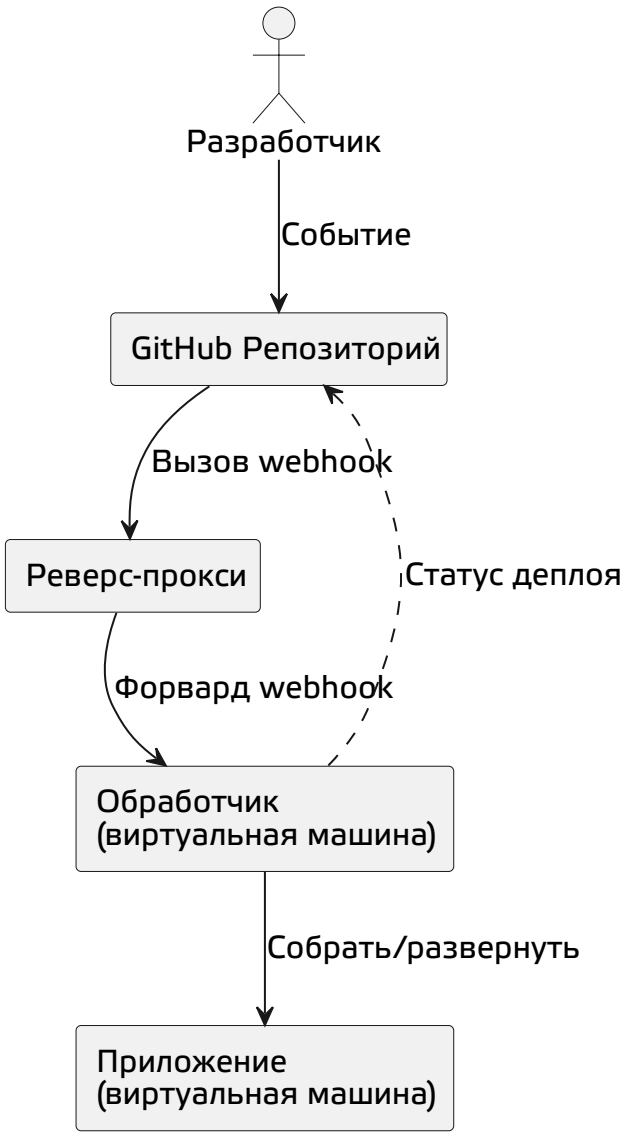

Рис. 1: Схема работы GitHub webhooks

В данной работе мы изучим механизм GitHub webhooks — автоматических HTTP запросов, которые отправляются при определенных событиях в репозитории. Как показано на схеме выше, процесс работы состоит из следующих этапов:

1. **Событие в репозитории**: разработчик выполняет push, создает pull request или другое действие

2. **Активация webhook**: GitHub автоматически отправляет HTTP POST-запрос на настроенный URL
3. **Обработка через прокси**: реверс-прокси перенаправляет запрос к обработчику
4. **Автоматическое развертывание**: обработчик получает HTTP запрос и запускает процесс сборки/развертывания приложения
5. **Обратная связь**: обработчик может отправить статус выполнения обратно в GitHub

Этот подход позволяет использовать Git-репозиторий не только как средство совместной работы с кодом, но и как центральный элемент автоматизации процессов разработки и развертывания ПО, что является ключевой практикой DevOps.

2.1 Задание 1

Подготовьте хост под управлением UNIX-подобной операционной системы.

2.2 Задание 2

На хосте настроить доступ до прокси сервера frp.

```
# Настройка переменных. Актуальные значения узнайте у преподавателя
PROXY=course.prafdin.ru
TOKEN=tokentoken
ID=prafdin

# Скачать и установить frp как systemd сервисы.
# Команда запросит пароль текущего пользователя
wget -qO- https://gist.github.com/lawrenceching/^
41244a182307940cc15b45e3c4997346/raw/^
0576ea85d898c965c3137f7c38f9815e1233e0d1/^
install-frp-as-systemd-service.sh | sudo bash

# Настройка доступа до frp server.
# Команда запросит пароль текущего пользователя
sudo tee /etc/frp/frpc.toml > /dev/null <<EOF # Начало команды
serverAddr = "$PROXY"
serverPort = 7000

auth.method = "token"
auth.token = "$TOKEN"

[[proxies]]
name = "hook-$ID"
type = "http"
localPort = 8080
customDomains = ["webhook.$ID.$PROXY"]

[[proxies]]
name = "app-$ID"
```

```
type = "http"
localPort = 8181
customDomains = ["app.$ID.$PROXY"]
EOF # Конец команды

# Запустите frp client. Команда запросит пароль текущего пользователя
sudo systemctl start frpc

# Проверьте корректность настройки
echo "Адрес для проверки: webhook.$ID.$PROXY"

wget -qO- https://gist.githubusercontent.com/\
prafdin/b9ff40c8bc6dc8c55ca7ac911e278ecc\
/raw/8134f5f34220576bfc8cd205910e1625d66b58cb\
/main.py | python3

# Перейдите по адресу для проверки в браузере.
# Если всё настроено правильно, то в браузере получите ok
```

2.3 Задание 3

Настроить webhook в GitHub репозитории

1. Перейдите в Settings → Webhooks → Add webhook
2. Укажите Payload URL: http://webhook.$ID.$PROXY (используйте свои значения из задания 2)
3. Укажите Content type: application/json
4. Выберите нужное события для активации webhook или укажите Send me everything
5. Сохраните webhook

Для отладки можете воспользоваться инструкцией по настройке проксирования вебхуков без использования FRP.

2.4 Задание 4

Создать обработчик webhook и настроить автоматическое развертывание приложения.

1. Создайте веб-сервер — обработчик webhook, который будет получать HTTP запросы от GitHub
2. В качестве обработки события реализуйте логику автоматического развертывания приложения при получении push события:
   - Клонирование или обновление кода из репозитория
   - Сборка приложения (если требуется)
   - Запуск тестов
   - Развертывание и перезапуск веб-сервера или приложения

3. Протестируйте работу webhook сервера, создав событие в репозитории
4. Убедитесь, что приложение успешно разворачивается и доступно
5. Отправьте статус развертывания в GitHub через REST API. Опционально

Примечание:

- Все необходимые зависимости приложения предварительно могут быть установлены вручную на виртуальной машине. Список зависимостей должен быть зафиксирован в документации или специальном файле, например, в requirements.txt в случае Python приложения.
- Веб-сервер — обработчик должен быть запущен на порту 8080, а развертываемое приложение должно быть доступно по адресу http://app.$ID.$PROXY и запущено на порту 8181
- Автоматическое развертывание должно работать для любой ветки в репозитории
- Учтите, что основное приложение должно работать, даже если webhook сервер отключен. Для решения этой задачи настоятельно рекомендуется превратить приложение в systemd сервис (инструкция 1, инструкция 2)

Шаблон веб-сервера обработчика webhook: prafdin/devops-demo-website

3 Контрольные вопросы

1. Что такое HTTP веб-сервер и для чего он нужен? Сколько веб-серверов используется в данной работе? Назовите их.
2. Что такое URL и какую роль он играет при сетевом взаимодействии?
3. Что такое reverse proxy и для чего он используется? Использовали ли вы его в данной работе?
4. Расскажите про HTTP-протокол: структуру запроса и ответа, методы, основные заголовки.
5. Как формируется HTTP-заголовок Host и для чего он используется в данной работе?
6. Какие проблемы могут возникнуть из-за использования NAT в контексте данной работы?
7. Зачем используется FRP в данной работе и какие проблемы он решает?
8. Какие преимущества и недостатки имеет использование webhook по сравнению с polling (регулярной проверкой изменений в репозитории)?

9. Какие задачи решает автоматизация развертывания через webhook?
10. Как можно обеспечить безопасность веб-сервера — обработчика webhook? Какая уязвимость присутствует в данной работе?
11. Как организовать работу с разными окружениями (тестовое, продакшн) с использованием webhook автоматизации?
12. Что произойдет, если на момент создания события в репозитории webhook сервер был выключен? Как избежать последствий?
13. Предложите способы отслеживания статуса развертывания при использовании webhook сервера.
14. Как можно протестировать логику webhook сервера без выполнения реального push в репозиторий?
15. Почему долгие операции внутри webhook обработчика считаются плохой практикой? К каким последствиям это может привести в данной работе?
16. Что произойдет при изменении файлов в репозитории, которые не относятся к исходному коду приложения, например README.md?
17. Как можно избежать race condition при двух последовательных коммитах? Опишите, как это реализовано в вашем решении.
18. Представьте, что после коммита ваше приложение обновилось. Как откатить версию приложения на предыдущую?