

Курс DevOps

Лабораторная работа №1

GitHub webhooks

5 сентября 2025 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.

Формат сдачи

- Создайте GitHub репозиторий для выполнения лабораторной работы или заведите отдельную ветку в репозитории с прошлой работы
- Добавьте в репозиторий все необходимые файлы: код приложения, файлы конфигурации и скрипты
- Предварительно отправьте ссылку на репозиторий преподавателю, а также ссылку на результат работы (если применимо)
- Во время сдачи будьте готовы описать и продемонстрировать выполненные шаги, а также ответить на возникающие вопросы
- Контрольный вопрос может быть задан на усмотрение преподавателя

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

Настроить виртуальную машину с Ubuntu 24.

2.2 Задание 2

На виртуальной машине настроить доступ до прокси сервера frp.

```
# Настройка переменных. Актуальные значения узнайте у преподавателя!  
PROXY=course.prafdin.ru  
TOKEN=token  
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/ea80e83e6a36220e2e9ccdaca8fde49ee888f2ab\
/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. Создайте или *форкните* репозиторий с выбранным приложением на *GitHub*
2. Перейдите в *Settings* → *Webhooks* → *Add webhook*
3. Укажите *Payload URL*: `http://webhook.$ID.$PROXY` (используйте свои значения из задания 2)
4. Укажите *Content type*: `application/json`
5. Выберите нужное события для активации *webhook* или укажите *Send me everything*
6. 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 события:
 - Клонирование или обновление кода из репозитория
 - Сборка приложения (если требуется)
 - Запуск тестов (при наличии)
 - 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](#) в случае push события. 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 разворачиваемое приложение должно быть доступно по адресу `http://app.$ID.$PROXY` и запущено на порту 8181
- Автоматическое развертывание должно работать для любой ветки в репозитории
- Перед выполнением задания необходимо выбрать приложение, событие для активации webhook (`push`, `pull request`, `release`) и решить о необходимости реализации обратной связи GitHub API.
- Учтите, что основное приложение должно работать, даже если webhook сервер отключен. Для решения этой задачи рекомендуется превратить приложение в `systemd` сервис ([инструкция 1](#), [инструкция 2](#))

Рекомендуемые приложения:

- **Статический сайт:** [cloudacademy/static-website-example](#)
- **C# приложение:** [danlla/Csharp-example](#)
- **Python веб приложение:** [prafdin/catty-reminders-app](#)

Шаблон веб-сервера обработчика webhook: [prafdin/devops-demo-website](#)

3 Контрольные вопросы

1. Какие проблемы решает автоматизация развертывания через webhook?
2. Как обеспечить безопасность веб-сервера — обработчика webhook? Какая уязвимость присутствует в данной работе?
3. Зачем использовать FRP в данной работе? Какие проблемы это решает?
4. В чем преимущества и недостатки webhook по сравнению с polling (регулярной проверкой изменений в репозитории)?
5. Как организовать работу с разными окружениями (тест, прод) с использованием webhook автоматизации?