УДК 004.413

СОЗДАНИЕ GIT FLOW ПРОЦЕССА ДЛЯ МИКРОСЕРВИСНОЙ АРХИТЕКТУРЫ С ПОЛНЫМ АВТОМАТИЗИРОВАННЫМ ЦИКЛОМ ТЕСТИРОВАНИЯ И АВТОМАТИЧЕСКИМ РАЗВЕРТЫВАНИЕМ НА ПРОДУКТОВОМ ОКРУЖЕНИИ

Решетов С. П.[1]
Научный руководитель – канд. техн. наук, Белозубов А. В.[1]
[1]Университет ИТМО

Введение

Настоящая работа посвящена разработке процесса Git Flow [1] для проекта с микросервисной архитектурой с полным автоматизированным циклом тестирования и автоматическим развертыванием проекта на продуктовом окружении. Процесс представляет собой интегрированный конвейер непрерывной интеграции и поставки [2], использующий формализованную модель управления жизненным циклом конфигураций микросервисов, основанную на модифицированной методологии Git Flow. Данная модель регламентирует правила ветвления, определяет сценарии автоматизированного тестирования и реализует стратегию автоматического развертывания на продуктовом окружении, что позволяет минимизировать риски деградации системы при внесении изменений.

Современная разработка программного обеспечения характеризуется переходом от монолитных приложений к распределенным, микросервисным системам. Такой переход позволяет повысить отказоустойчивость, масштабируемость, гибкость разработки системы, однако создает ряд инженерных проблем, усложняются процессы интеграции, тестирования, доставки изменений кода до пользователей. В условиях, когда связанные микросервисы разрабатываются независимо, традиционные подходы к управлению версиями, релизами становятся неэффективными, увеличивая время доставки цельного проекта до продуктового окружения.

Основная часть

Разрабатываемый процесс представляет собой автоматизированный конвейер непрерывной интеграции и доставки, где модель ветвления Git Flow служит источником событий, инициирующих автоматическое выполнение определенных сценариев. В основе модели лежит модифицированная схема ветвления Git Flow, адаптированная под микросервисную архитектуру: основная ветка «master» хранит актуальное состояние кода, соответствующее пре-продуктовому и продуктовому окружениям, для интеграции текущих изменений используется «dev», ветки «feature/*», «fix/*» и т. д. создаются для реализации конкретных задач, «hotfix/*» для исправлений критических ошибок в релизах. Ключевая особенность предлагаемого подхода заключается в отказе от отдельной релизных веток – интеграция изменений происходит непосредственно в «master» через механизм pull request, а фиксация релизных версий выполняется путем автоматического проставления тегов и версий перед развертыванием на окружениях.

Важным аспектом предлагаемого процесса является разделение кода микросервисов и конфигурации цельного проекта, состоящий из множества сервисов. В связи с этим вводится отдельный репозиторий для хранения версий, развернутых на окружении. Такой подход позволяет централизировано управлять состоянием пре-продуктовым и продуктовым окружениями.

Сам конвейер непрерывной интеграции и доставки имеет двухкомпонентную структуру: изменения в исходном коде сервисов инициируют сборку, статическое и

динамическое тестирование кода, а изменения в репозитории конфигурации запускают процессы развертывания сервисов на окружениях.

Выводы

Предложенный процесс рекомендуется к внедрению в проектах, использующих микросервисную архитектуру и стремящихся к минимизации ручных операций при релизах. Апробация разработанного подхода может быть проведена путем поэтапного развертывания конвейера на одном из пилотных проектов с последующим анализом ключевых метрик: времени доставки изменений до продуктового окружения, количества инцидентов, связанных с человеческим фактором при деплое, и частоты успешных релизов. Результаты внедрения позволят количественно оценить эффективность предложенного процесса и тиражировать его на другие проекты.

Литература

1. GitFlow workflow [Электронный ресурс] // Atlassian. – Режим доступа: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow (Дата обращения 16.02.2026).
2. CI/CD Pipelines Evolution and Restructuring: A Qualitative and Quantitative Study // 2021 IEEE International Conference on Software Maintenance and Evolution (ICSME). – Luxembourg, 2021. – Режим доступа: https://ieeexplore.ieee.org/document/9609201 (Дата обращения 16.02.2026).