

Организация процессов непрерывной интеграции и развертывания микросервисной архитектуры в изолированных технологических окружениях

Вихляев Дмитрий Романович

Приамурский государственный университет имени Шолом-Алейхема

Студент

Аннотация

В данной работе описываются этапы процесса разворачивания микросервисов. В содержании приводятся стандартные принципы непрерывной интеграции и развёртывания сервисов в окружения dev, test и prod. Результаты работы позволяют определить оптимальное количество использованных окружений между различными вариантами непрерывного разворачивания микросервисов.

Ключевые слова: микросервисная архитектура, непрерывная интеграция, непрерывное развёртывание, конвейер сборки.

Organizing continuous integration and deployment processes for microservice architecture in isolated technological environments

Vikhlyayev Dmitry Romanovich

Sholom-Aleichem Priamursky State University

Student

Abstract

This paper describes the stages of the microservices deployment process. The content provides standard principles for continuous integration and deployment of services in dev, test, and prod environments. The results of the work allow determining the optimal number of used environments between different options of continuous deployment of microservices.

Keywords: microservice architecture, continuous integration, continuous deployment, and build pipeline.

1 Введение

1.1 Актуальность

Главным преимуществом непрерывной интеграции является проверка на то, что внесённые в код изменения не сломали существующих функций. Стандартная архитектура включает разбиение на стенды разработки, которые защищают пользователей от ошибок. Они изолируют код и инфраструктуру на разных этапах жизненного цикла. Для этого при каждом комите в репозитории исходного кода запускается автоматическая сборка сервиса и прогон Unit-тестов. Полученные в результате сборки артефакт в дальнейшем используются

для запуска интеграционных тестов и развёртывания в тестовых и рабочих средах. При этом важно помнить, что код следует собирать один раз и уже использовать для тестирования и запуска полученный артефакт. Это не только экономит ресурсы на сборке. Но и добавляет уверенности в том, что в рабочей среде запускается именно то, что было протестировано на тестовых средах.

1.2 Обзор исследований

Статья А.М.Шитько посвящена изучению технологии построения микросервисной архитектуры, которая позволяет сохранять модульность и независимость развёртывания компонент Enterprise приложения [1]. В работе Д.С.Кугушева дано описание принципа работы микросервисной архитектуры, приведены ее отличительные особенности, позволяющие существенно упростить процесс разработки больших продуктов, а также упомянуты основные технологические особенности микросервисного подхода, которые необходимо учитывать при выборе этой технологии разработки. [2]. В исследовании М.Э.Джалалова особое внимание уделяется шаблонам API Gateway, Backend For Frontend и Circuit Breaker, их реализации и оптимизации в контексте микросервисных архитектур. [3]. В качестве результата исследования А.Абдирашитулы, стала предложенная концепция LLM-ориентированной микросервисной архитектуры, обеспечивающая изоляцию ресурсов и независимое масштабирование компонентов. [4]. В статье М.Г.Городничев и Р.В.Полонский сделан обзор типов современных веб-приложений и архитектуры, используемой для их разработки, дано сравнение микросервисной и монолитной архитектуры [5]. М.Н.Карпович исследовал особенности проектирования микросервисно-событийных архитектур для высоконагруженных распределенных систем обработки информации [6]. С.В.Маличенко рассмотрел проблемы перехода от монолитной к микросервисной архитектуре [7]. Г.Р.Берайлик описал способы спроектирования информационных систем с помощью микросервисной архитектуры [8].

1.3 Цель исследования

Цель исследования – описать организационные процессы непрерывной интеграции и развёртывания микросервисной архитектуры в изолированных технологических окружениях.

2 Материалы и методы

В работе приводится пример разделения поставок на стенды окружений для разработки, финального проекта, интеграционного и приёмочного тестирования. Описываются способы организации интеграции различных версий сервисов.

3 Результаты и обсуждения

Непрерывная интеграция позволяет разработчикам быстрее узнать о проблемах с внесёнными изменениями и может их скорее исправить до того,

как забыт контекст изменений. Автоматизация непрерывной интеграции позволяет исключить человеческий фактор как при тестировании кода, так и при сборке артефактов.

Таким образом, можно быть уверенным, что всегда тестирование и сборка выполняются одинаково. Кроме того, непрерывная интеграция даёт повторяемость. То есть можно создать один и тот же артефакт из одного и того же коммита кода, и он не будет отличаться ничем. Это полезно на тот случай, если артефакт был утерян или произошла другая непредвиденная ситуация (рис.1).

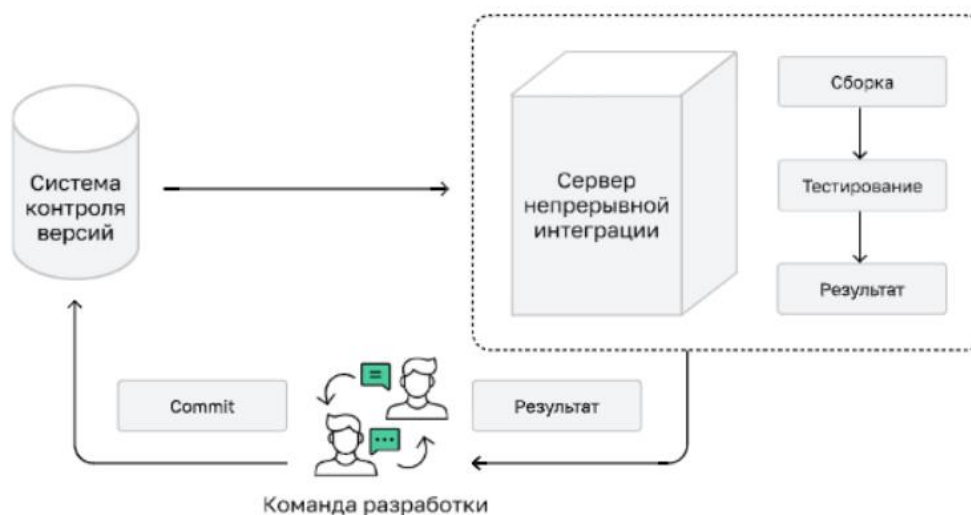


Рис. 1. Процессы непрерывной интеграции

Непрерывная интеграция является. Насущной необходимостью при разработке системы, состоящей из большого количества сервисов. При этом есть разные подходы использования непрерывной интеграции в микросервисной архитектуре. В первую очередь, они связаны с тем, каким образом хранится код сервисов.

Существуют различные варианты. Например, можно хранить код всех сервисов в одном репозитории и при изменении любого из сервисов собирать все сервисы одновременно. В альтернативном варианте хранить весь код сервисов в одном репозитории, но для каждого сервиса иметь отдельную конфигурацию в BID-сервере. Также для каждого сервиса использовать отдельный репозиторий и отдельную конфигурацию для сборки и тестирования. Последний вариант выглядит наиболее выгодно. Сервисы максимально изолируются друг от друга, начиная от кода и заканчивая конечным артефактом (рис.2).

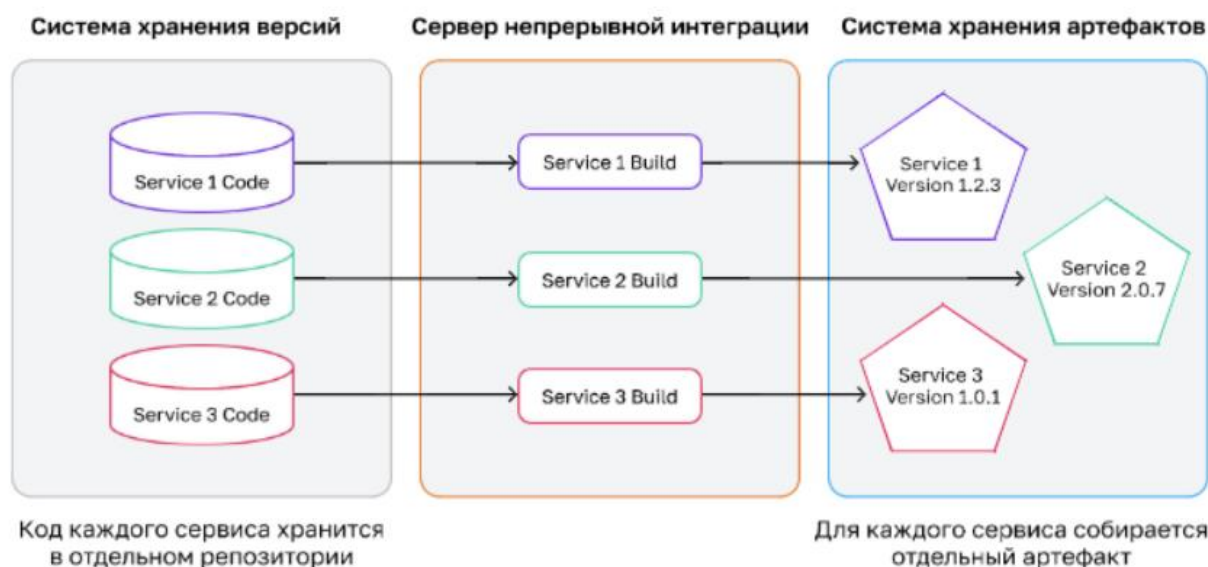


Рис. 2. Непрерывная интеграция разных версий микросервисов

Подход несёт дополнительную сложность, которая может быть нивелирована дополнительной автоматизацией. Можно автоматизировать создание репозитория, создание конфигураций, завести шаблоны именования сервисов. Также можно заранее создать шаблоны структур репозитория для разных технологических стеков и подготовить шаблоны конфигурации сборок. В итоге такой автоматизацией можно получить все выгоды изолирования сервисов друг от друга, при этом, не сильно усложняя общую систему непрерывной интеграции.

Непрерывная интеграция тесно связана с непрерывной поставкой. Непрерывная поставка означает то, что в результате сборки сервиса публикуется некий артефакт, который уже может быть вручную проверен и далее установлен в рабочую среду в ручном режиме.

Когда все проверки полученного в результате непрерывной поставки артефакта автоматически проверяются при помощи интеграционных, нагрузочных и других типов тестов. Для того чтобы исключить регресс и влияние его на продуктивную систему. Таким образом, наличие таких тестов позволяет быть уверенным в том артефакте, который поставляется, и позволяет автоматически поставить его в рабочую среду (рис.3).

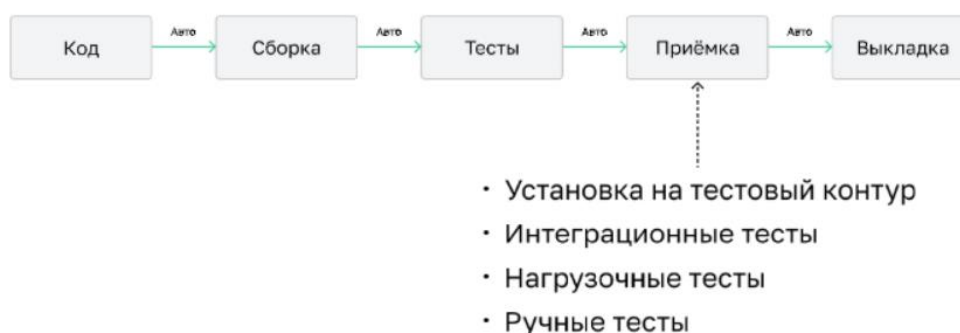


Рис. 3. Непрерывная установка

Вне зависимости от того, используется ли непрерывная поставка или нет, обычно в больших крупных системах существует набор стендов, которые позволяют разрабатывать и тестировать систему на всех этапах производства. На сервисе непрерывной интеграции осуществляется сборка, Unit-тестирование и публикация артефактов, полученных в результате сборки. Дальше идёт набор стендов.

Например, это может быть разработческий стенд, на котором разработчики могут проверять новый функционал. Также обязательно должен быть стенд для интеграционного тестирования, где разработчики и тестировщики могут проверять сервисы по взаимодействию друг с другом. Обычно также создаётся стенд для приёмочного тестирования. Где заказчики, а также нагрузочные тестировщики, аналитики могут выполнять приёмку нового функционала, проверяя её не только на наличие багов, но и на то, что было реализовано именно то, что они заказывали. И рабочий стенд, на который уже выполняется конечная поставка в результате всех проверок (рис.3).

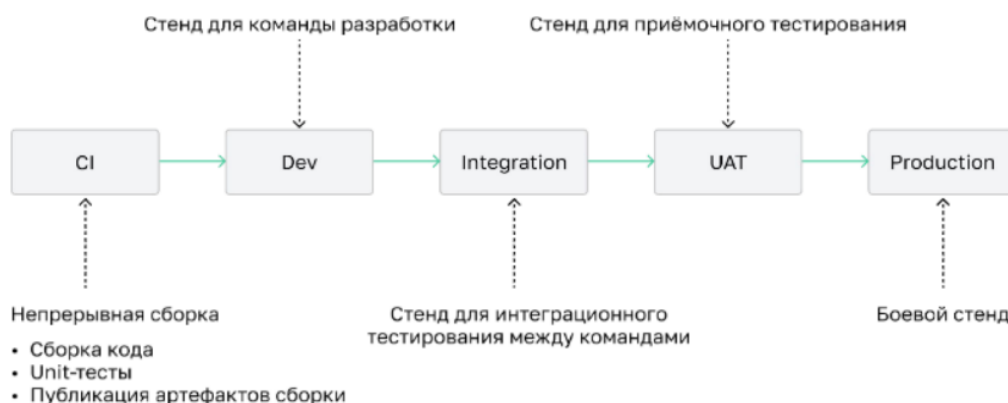


Рис. 4. Разбиение поставки на стенды

В непрерывном разворачивании, существует большое количество различных вариантов развернуть новую версию сервиса на существующем стенде. Среди них Rescreat является один из самых простых, но требует полной остановки сервиса.

4 Выводы

Таким образом, использование процессов непрерывной интеграции и развёртывания в сочетании с изолированными стендами позволяет разработчикам быстрее узнавать и исправлять ошибки без непредвиденной остановки системы. Изолированная проверка сложных бизнес-сценариев, с параллельным тестированием независимы функций повышает вероятность, что в продаваемый продукт попадает только полностью валидный и работоспособный функционал.

Библиографический список

1. Шитько А.М. Проектирование микросервисной архитектуры программного обеспечения //Труды БГТУ. Серия 3: Физико-математические науки и информатика. 2017. №. 9 (200). С. 122-125.
2. Кугушева Д.С. Проектирование сложного программного обеспечения с использованием микросервисной архитектуры //Инновации и инвестиции. 2020. №. 5. С. 188-190.
3. Джалалов М.Э. Применение шаблонов проектирования для управления API в микросервисной архитектуре //Экономика и качество систем связи. 2024. №. 1 (31). С. 128-136.
4. Абдирашитулы А. Микросервисная архитектура в NLP платформах: проектирование и масштабируемость //Вестник науки. 2026. Т. 1. №. 5 (98). С. 609-619.
5. Городничев М.Г., Полонский Р.В. Оценка возможности использования микросервисной архитектуры при разработке пользовательских интерфейсов клиент-серверного программного обеспечения //Экономика и качество систем связи. 2020. №. 3 (17). С. 33-43.
6. Карпович М.Н. Особенности проектирования микросервисно-событийных архитектур для высоконагруженных распределенных систем обработки информации //Труды БГТУ. Серия 3: Физико-математические науки и информатика. 2023. №. 1 (266). С. 89-95.
7. Маличенко С.В. Проблемы перехода от монолитной к микросервисной архитектуре //Евразийский научный журнал. 2022. №. 5. С. 8-19.
8. Берайлик Г.Р. Проектирование информационных систем с помощью микросервисной архитектуры //Инновационные подходы к решению технико-экономических проблем. 2020. С. 187-191.