УДК 004.58

Разработка архитектуры для реализации CI/CD процессов и их внедрение в процесс разработки программного обеспечения

Смирнов Г.А. (ИТМО)

Научный руководитель – кандидат технических наук, доцент Харитонов А.Ю. (ИТМО)

Введение. В современном быстро меняющемся мире разработки программного обеспечения, компании постоянно ищут способы ускорить вывод новых функций на рынок, повысить качество выпускаемого продукта и оптимизировать процессы разработки. Одним из наиболее эффективных подходов к достижению этих целей является внедрение практик CI/CD (Continuous Integration/Continuous Delivery или Continuous Deployment) [3]. CI/CD – это не просто набор инструментов, а скорее философия и набор практик, которые автоматизируют и оптимизируют процесс разработки, тестирования и развертывания программного обеспечения. Внедрение CI/CD позволяет командам разработчиков быстрее реагировать на изменения требований, оперативно выявлять и устранять ошибки, а также чаще выпускать обновления и новые версии продукта.

Основная часть. Чтобы организовать максимально быструю доставку цифрового продукта пользователю, необходимы 2 составляющие:

1. Непрерывная интеграция (continuous integration). Разработчики как можно чаще сливают изменения в основную ветку, используя систему контроля версий (например, Git). При этом, любые изменения проходят через автоматические тесты. Иными словами, продукт дополняется постепенно, а не лавинообразно. Если представить таймлайн разработки в виде некой линии, а обновления в виде размещенных на ней точек, то все они будут равномерно распределены по всей протяженности таймлайна. [2]
2. Непрерывная поставка (continuous deployment). Это дополнительное расширение непрерывной интеграции, которое автоматически развертывает новые изменения кода после этапа сборки в производственной среде клиента. Цель автоматизации очевидна — снизить нагрузку с разработчиков и свести к минимуму человеческий ошибки, при этом поддерживая согласованный процесс выпуска ПО. [2]
3. Выбор инструмента для реализации CI/CD зависит от специфики проекта, размера команды и инфраструктуры. Вот обзор нескольких популярных инструментов CI/CD с их основными характеристиками:
   - 3.1. Jenkins: Один из самых популярных инструментов с открытым исходным кодом для CI/CD. Он предоставляет множество плагинов и интеграций, что делает его гибким решением. Jenkins позволяет проводить тестирование в реальном времени и составлять отчеты об изменениях в коде. Он прост в установке и настройке через веб-интерфейс и может распределять работу между несколькими машинами [1]
   - 3.2. GitLab CI: Этот инструмент встроен в GitLab и предлагает различные функции, такие как просмотр кода, CI/CD и непрерывное развертывание в рамках единой панели. Для использования GitLab CI/CD необходимо разместить кодовую базу в репозитории Git и указать сценарии для выполнения сборки, тестирования и развертывания в файле YAML. GitLab CI предлагает API для интеграции со сторонними инструментами и доступен для Windows, Linux и macOS
   - 3.3. TeamCity: Коммерческий CI/CD-сервер от JetBrains, который предлагает мощные возможности для автоматизации сборки и развертывания. TeamCity поддерживает интеграцию с Docker и другими инструментами разработки, предоставляет отчеты о ходе сборки в реальном времени и позволяет управлять сложными конвейерами с помощью параллельных сборок. Бесплатная версия доступна для небольших команд и проектов с открытым исходным кодом, а платные подписки предлагают дополнительные функции и поддержку

4. При выборе инструмента для организации локального репозитория, важны такие аспекты, как поддержка нужных форматов пакетов, возможности управления зависимостями и интеграция с существующими процессами разработки. Вот обзор нескольких популярных инструментов
   4.1. Aptly: Это инструмент для управления APT-репозиториями, который позволяет создавать, управлять и развертывать локальные репозитории Debian. Aptly поддерживает создание репозиториев с возможностью управления версиями и зависимостями, а также позволяет создавать снапшоты репозиториев.
   4.2. LocalRepo: Это решение для создания локальных репозиториев для RPM-пакетов. LocalRepo позволяет управлять пакетами и их зависимостями в локальной сети, обеспечивая возможность кэширования и организации доступа к пакетам.
   4.3. Nexus Repository Manager: Это мощный инструмент для управления артефактами, который поддерживает множество форматов пакетов, включая Maven, npm, NuGet и Docker. Nexus предоставляет возможности для создания локальных и прокси-репозиториев, а также интеграцию с CI/CD процессами.
   4.4. JFrog Artifactory: Это универсальный менеджер артефактов, который поддерживает все популярные форматы пакетов. Artifactory предлагает мощные функции для управления зависимостями, кэширования удаленных репозиториев и интеграции с CI/CD инструментами. Он также предоставляет REST API для автоматизации процессов.

Выводы. Проведено исследование современных методологий в сфере разработки программного обеспечения. Были изучены методологии непрерывной интеграции и непрерывной поставки. Рассмотрены существующие технологии для реализации CI/CD процессов.

Список использованных источников:

1. «Jenkins» [Электронный ресурс]. – Режим доступа: URL: https://www.jenkins.io/ (дата обращения: 15.12.2024);
2. « Что такое CI/CD? Разбираемся с непрерывной интеграцией и непрерывной поставкой» [Электронный ресурс]. – Режим доступа: URL: https://habr.com/ru/companies/otus/articles/515078/ (дата обращения: 15.12.2024);
3. «Пайплайн CI/CD: что это такое, как применяется в разработке» [Электронный ресурс]. – Режим доступа: URL: https://timeweb.cloud/tutorials/ci-cd/pajplajn-ci-cd-chto-ehto-takoe/ (дата обращения: 20.12.2024);