ПОСТРОЕНИЕ ЭФФЕКТИВНОГО CI/CD КОНВЕЙЕРА

Яцевич К.В.

гр.367041

Белорусский государственный университет информатики и радиоэлектроники,
г. Минск, Республика Беларусь

Научный руководитель: Кореневский С.А. – кандидат технических наук, доцент кафедры ИКТ

Аннотация. Доклад посвящен построению эффективного ci/cd конвейера, который является неотъемлемой частью современного процесса разработки программного обеспечения. В нем рассматриваются ключевые этапы автоматизации: от интеграции изменений в код до развертывания в рабочем окружении, включая тестирование, управление артефактами и мониторинг.

Ключевые слова: Continuous integration, continuous deployment

Введение. Современная разработка программного обеспечения требует не только высокой скорости внедрения изменений, но и стабильности, предсказуемости и надежности процессов. В условиях динамично развивающихся проектов ручное управление тестированием, сборкой и развертыванием становится сложной и ресурсоемкой задачей[1].

Continuous integration и continuous deployment позволяет автоматизировать эти процессы, обеспечивая постоянную интеграцию изменений, их тестирование и доставку в рабочее окружение. Это снижает вероятность ошибок, ускоряет выпуск новых версий и упрощает совместную работу команд.

Эффективный ci/cd-конвейер включает в себя несколько ключевых этапов, таких как автоматическая сборка, тестирование, управление артефактами, развертывание и мониторинг. Для его реализации существует множество инструментов, каждый из которых подходит для различных масштабов проектов и инфраструктурных решений.

Основная часть. Continuous integration и continuous deployment — это набор практик, направленных на автоматизацию процессов разработки, тестирования и развертывания приложений.

Continuous integration — это процесс интеграции изменений в основной код проекта как можно чаще. Цель continuous integration — обеспечить, чтобы код, написанный различными разработчиками, не конфликтовал между собой. Он включает автоматическую сборку и тестирование, чтобы оперативно выявлять ошибки на ранних стадиях. Continuous deployment — это процесс автоматической доставки проверенного кода в рабочее окружение. Continuous deployment подразумевает автоматическое развертывание изменений в среду, которая предназначена для конечного пользователя, сразу после успешного тестирования.

Все начинается с разработки кода. При использовании ci/cd подхода коммиты, сделанные разработчиками, автоматически проверяются системой, что позволяет предотвратить проблемы в кодовой базе на ранних этапах.

После того как код попадает в репозиторий, система ci/cd инициирует процесс сборки, который включает компиляцию исходного кода, зависимостей и создание артефактов, таких как исполнимые файлы или контейнеры docker. Это критический момент, так как любые ошибки на этом этапе могут повлиять на стабильность и функциональность приложения.

Одним из ключевых преимуществ ci/cd является автоматическое тестирование, которое проводится сразу после сборки[2].

Во время процесса сборки и тестирования важно проверять качество кода с помощью таких инструментов, как sonarqube для статического анализа, которые помогут выявить потенциальные

проблемы до того, как код попадет в продакшн.

После успешной сборки и тестирования приложение должно быть готово к развертыванию. Артефакты хранятся в репозиториях артефактов, таких как nexus или artifactory, которые позволяют эффективно управлять версиями и зависимостями[2].

После успешного прохождения всех тестов система ci/cd автоматизирует развертывание в тестовом или продакшн окружении. Этот процесс может быть настроен для безостановочного развертывания или через стратегию канареечных релизов, при которой новая версия приложения доступна лишь небольшой части пользователей, чтобы минимизировать риски.

После развертывания приложения важно отслеживать его состояние с помощью инструментов мониторинга, таких как prometheus, grafana или elk stack. Эти инструменты позволяют своевременно обнаруживать проблемы и отправлять уведомления. В случае необходимости система должна поддерживать возможность быстрого отката на предыдущую стабильную версию.

Не все приложения одинаковы, и ci/cd конвейеры должны быть адаптированы под особенности конкретных типов приложений. Для мобильных приложений можно настроить автоматическое тестирование на разных устройствах и платформах с использованием инструментов fastlane и firebase test lab. Также важно автоматизировать сборку и деплой на платформы app store и google play.

К популярным ci/cd инструментам можно отнести jenkins, gitlab ci, github actions, circle ci, travis ci.

Jenkins — это один из самых популярных инструментов для автоматизации ci/cd. Он поддерживает множество плагинов, что позволяет интегрировать его с различными системами и инструментами для тестирования, сборки, развертывания и мониторинга. Jenkins хорош для проектов любого масштаба и легко масштабируется.

Gitlab ci/cd встроен непосредственно в Gitlab, что делает его удобным для команд, использующих Gitlab как основную систему контроля версий. Gitlab ci/cd поддерживает автоматическую сборку, тестирование, деплой и мониторинг. Он также позволяет интегрировать дополнительные инструменты для обеспечения качества кода и безопасности.

Github Actions предоставляет удобную платформу для continuous integration и continuous deployment непосредственно в Github. Это мощный инструмент, позволяющий автоматизировать сборку, тестирование и развертывание приложений на различных облаках и сервисах.

Circle ci — это облачное решение для ci/cd, которое отличается высокой скоростью и простотой интеграции с github и bitbucket. Оно поддерживает масштабируемость и быструю настройку конвейеров.

Travis CI — облачный continuous integration инструмент, который интегрируется с github и обеспечивает автоматическую сборку и тестирование. Он особенно популярен среди проектов с открытым исходным кодом.

Для построения действительно эффективного ci/cd конвейера недостаточно просто автоматизировать процесс интеграции и развертывания. Важно учитывать такие аспекты, как безопасность, масштабируемость, отказоустойчивость и удобство сопровождения.

Автоматизация процессов развертывания не должна приводить к компрометации безопасности. Безопасность должна быть не дополнительным этапом, а неотъемлемой частью ci/cd. Для этого стоит внедрить практики devsecops, которые интегрируют безопасность на всех этапах разработки и развертывания. Для безопасного хранения токенов, паролей и ключей используются инструментов типа hashicorp vault или aws secrets manager. Также применяется принцип минимально необходимого доступа в настройках прав разработчиков, devops-инженеров и автоматических процессов. Для проверки целостности создаваемых бинарных файлов используется цифровые подписи на этапах сборки. Важным аспектом является автоматический анализ безопасности на этапе программирования. Для этого используются checkmarx или snyk для выявления уязвимостей в коде и зависимостях. На регулярной основе необходимо производить сканирование docker образов для поиска известных уязвимостей.

При увеличении нагрузки ci/cd должен масштабироваться без потери скорости развертывания. Для этого применяется горизонтальное масштабирование исполняющих узлов: добавление дополнительных исполняющих узлов позволяет выполнять тесты и сборки параллельно. Важным фактором является кэширование зависимостей. Использование кеша для хранения ранее загруженных пакетов ускоряет сборку.

Документация играет важную роль в успешной настройке и поддержке ci/cd конвейера. Без четкой документации команды могут столкнуться с проблемами при развертывании, тестировании и мониторинге. Важно описать каждый этап конвейера, используемые инструменты, а также процедуры для откатов, восстановления и реагирования на ошибки. Это поможет новым сотрудникам быстрее разобраться в процессе и минимизировать ошибки. Регулярные тренинги и семинары по ci/cd, внедрению новых инструментов и поддержке уже существующих пайплайнов обеспечат команду необходимыми знаниями для эффективной работы с ci/cd.

Заключение. Построение эффективного ci/cd конвейера является ключевым элементом в современной разработке программного обеспечения. Он позволяет автоматизировать многие рутинные процессы, такие как сборка, тестирование и развертывание, тем самым снижая риски ошибок и ускоряя выпуск новых версий приложений. Этапы ci/cd, включая автоматизацию тестирования, управление артефактами и деплоймент, требуют тщательной настройки и использования правильных инструментов.

Чтобы ci/cd действительно принесло пользу, важно учитывать несколько факторов. Это грамотный выбор инструментов, настройка подходящих процессов, а также наличие культуры devops в команде. Только слаженная работа всех участников процесса — разработчиков, тестировщиков, devops-инженеров и других — может обеспечить полную автоматизацию и эффективную работу ci/cd конвейера.

Таким образом, continuous integration и continuous deployment — это не просто набор инструментов, а целая философия, которая помогает ускорить процесс разработки, повысить качество приложений и снизить затраты на поддержку. Понимание и грамотное применение этих практик позволит командам быстрее реагировать на изменения и эффективно управлять жизненным циклом разработки.

Список использованных источников:

1. Eberhard Wolf. Continuous delivery. The practice of continuous updates – Person Education, Inc 2018. – 320 c.

2. Continuous Integration: Improving Software Quality and Reducing Risk. – O'Reilly Media, Inc., 2007. – 336 c.

BUILDING AN EFFICIENT CI/CD PIPELINE

Yatsevich K.V.

gr.367041

Belarusian State University of Informatics and Radioelectronics, Minsk, Republic of Belarus

Korenevksy S.A. – Ph. D

Annotation. The report is devoted to building an effective ci/cd pipeline, which is an integral part of the modern software development process. It covers key stages of automation: from integrating changes into the code to deploying to the production environment, including testing, artifact management and monitoring.

Keywords: Continuous integration, continuous deployment