Введение

В современном мире стремление к автоматизации – естественный процесс. Сама по себе, автоматизация – это передача техническим средствам функций управления и контроля, ранее выполнявшихся человеком вручную.

В IT индустрии актуальность применения автоматизированных средств для разработки программных продуктов обусловлена тем, что оперативная подготовка программного продукта к релизу — ключевое конкурентное преимущество во многих отраслях бизнеса. Поэтому в основе быстро растущих и гибко реагирующих на любые изменения современных компаний лежит большое разнообразие инженерных культур и практик. Одной из таких практик является CI/CD, которой занимаются DevOps-инженеры.

В [разработке ПО](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%9F%D0%9E), CI/CD или CICD — это комбинация [непрерывной интеграции](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F) (continuous integration) и [непрерывной доставки и развертывания](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B4%D0%BE%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0) (continuous delivery или continuous deployment) программного обеспечения в процессе разработки.

Сам по себе перевод исходного кода в рабочее состояние на конкретном сервере (другими словами, деплой) — рутинная задача. Для этого необходимо проверить и собрать код, провести тестирование, перенести все файлы на сервер, выполнить миграции, поправить конфигурации и др. На выполнение этих задач вручную уходит значительное количество времени. В тоже время высока вероятность упустить мелкую деталь, которая в итоге сломает конвейер. Таким образом, на подготовку кода к релизу уйдут месяцы, что недопустимо в условиях постоянной конкуренции.

Прежде чем новый программный продукт появится на рынке, программисты вынуждены преодолевать большое количество проблем, которые могут внезапно возникнуть на любом этапе разработки.

Наиболее часто возникающие проблемы в процессе разработки:

— **нехватка времени на разработку нового функционала;**

— низкое качество кода за счет необходимости выполнить работу в срок;

— большое количество ошибок в коде, приводящие к **тратам времени на** поиск их причин и исправление;

— проблемы с интеграцией, исправление которых обходится дорого;

— постоянная необходимость в наличии текущей стабильной версии вместе с продуктами сборок — для тестирования, демонстрации;

— **разобщённость между разработкой и операционной деятельностью.**

Решая перечисленные проблемы, CI/CD нормализует работу и коммуникацию бизнеса с IT — исчезают противоречия, а процесс построения CI/CD конвейера делает процесс разработки и выпуска ПО более компактным и эффективным. Пока компьютеры решают однообразные задачи, разработчики могут сосредоточиться на решении проблем бизнеса и проведении экспериментов.

Исходя из выше написанного были выделены цель и задачи.

Целью работы является автоматизация CI/CD процесса на выделенные сервера.

Задачи исследования:

— провести анализ предметной области;

— провести обзор существующих программных решений и выявить их преимущества и недостатки;

— определить требования к разрабатываемому программному продукту;

— провести проектирование базы данных исходя из требований;

— разработка приложения;

— тестирование разработанного программного продукта.

Объект исследования — непрерывная интеграция, доставка и развертывание программного обеспечения.

Предмет исследования — исследование методов автоматизации процесса continuous integration, continuous delivery, continuous deployment.

Методы исследований. Для решения поставленных задач были использованы методы математического моделирования, системного анализа, программной инженерии, объектно-ориентированного программирования, технологии проектирования человеко-машинного взаимодействия.

Практическая значимость работы заключается в том, что разработанное программное обеспечение будет помогать разработчикам ускорять разработку программных средств и предоставлять клиенту необходимый функционал.

1 глава

В IT индустрии используется большое разнообразие инженерных культур и практик, которые позволяют командам разработчиков быстро предоставить клиенту качественный продукт и в дальнейшем непрерывно добавлять в него функционал. Оперативная подготовка программного продукта к релизу — ключевое конкурентное преимущество во многих отраслях бизнеса, поэтому в основе быстро растущих и гибко реагирующих на любые изменения современных компаний лежит методология непрерывной интеграции и доставки (CI/CD). Этим этапом разработки занимаются [DevOps](https://ru.wikipedia.org/wiki/DevOps)-инженеры, в настоящий момент стремящиеся применять CI/CD практически для всех задач.

Сам по себе перевод исходного кода в рабочее состояние на конкретном сервере (другими словами, деплой) — рутинная задача. Для этого необходимо скомпилировать код, запустить тесты, перенести все файлы на сервер, выполнить миграции, поправить конфигурации и др. На выполнение этих задач вручную уходит значительное количество времени. В тоже время высока вероятность упустить мелкую деталь, которая в итоге сломает конвейер. Таким образом, на подготовку кода к релизу уйдут месяцы, что недопустимо в условиях постоянной конкуренции.

В [разработке ПО](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D1%80%D0%B0%D0%B1%D0%BE%D1%82%D0%BA%D0%B0_%D0%9F%D0%9E), CI/CD или CICD — это комбинация [непрерывной интеграции](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F) (continuous integration) и [непрерывной доставки и развертывания](https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F_%D0%B4%D0%BE%D1%81%D1%82%D0%B0%D0%B2%D0%BA%D0%B0) (continuous delivery или continuous deployment) программного обеспечения в процессе разработки.

Цель состоит в том, чтобы увеличить раннее обнаружение дефектов, повысить производительность и ускорить циклы выпуска. Этот процесс отличается от традиционных методов, когда набор обновлений программного обеспечения интегрировался в один большой пакет перед развертыванием более новой версии.

Автоматизированный CI/CD pipeline позволяет свести к нулю ручной труд при выполнении рутинных операций: сборки и выкладки новых версий кода. Все это делается по нажатию на одну кнопку.

На рынке существует большое количество платных и бесплатных инструментов для настройки такого процесса, который мы называем “процессом непрерывной интеграции” или CI/CD. Однако для построения действительно зрелого процесса недостаточно просто установить инструмент. За каждым этапом конвейера стоит сложная логика того, что должно быть запущено, на каких вычислительных ресурсах и как эти ресурсы используются.

Для сравнения функциональности уже существующих сервисов были выбраны следующие инструменты: GitLab CICD, Jenkins, Github Actions, Buddy. Их сравнение представлено в таблице 1.

Таблица 1 — Достоинства и недостатки существующих средств CICD

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Характеристика** | **Jenkins** | **GitLab CI/CD** | Github actions | Buddy |
| Язык, на котором написан | Java | Go, Ruby | Ruby on Rails и Erlang |  |
| Открытый или закрытый код | Открытый | Открытый |  |  |
| Установка | Требуется | Не требуется, т.к. это встроенная возможность платформы GitLab |  |  |
| Операционные системы (ОС) | Linux, Windows, Mac | Linux, Windows, Mac |  | Linux, Windows, Mac, iOS, Android |
| Уникальные особенности | Поддержка плагинов | Глубокая интеграция в систему управления версиями |  |  |
| Язык для описания workflow | Yaml | Yaml | Yaml | Yaml |
| Самостоятельное развёртывание системы | Это — единственный вариант использования системы | + | + | + |
| Распараллеливание | + | + |  | + |
| Создание CI/CD-конвейеров | +  (Используется Jenkins Pipeline) | + |  | + |
| Распознавание языка и автоматическое создание скриптов-шаблонов |  | +  AutoDevOps | +  (Сильно отличается от AutoDevOps) |  |
| Мониторинг производительности приложений | - | + |  |  |
| Экосистема | Существует более 1000 плагинов | Система развивается в рамках GitLab |  |  |
| API | + | Предлагает API для более глубокой интеграции в проекты |  | + |
| Интеграция с другими инструментами | + | + | + | + |
| Контроль качества кода | Поддерживается с помощью плагина SonarQube и других плагинов | + |  |  |

Список использованных источников

1.CI/CD Википедия [Электронный ресурс] URL: <https://ru.wikipedia.org/wiki/CI/CD>

2.Непрерывная интеграция Википедия [Электронный ресурс] URL: https://ru.wikipedia.org/wiki/%D0%9D%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%B0%D1%8F\_%D0%B8%D0%BD%D1%82%D0%B5%D0%B3%D1%80%D0%B0%D1%86%D0%B8%D1%8F

3. Gitlab Documentation [Электронный ресурс] URL: <https://docs.gitlab.com/>

4. Jenkins Documentation [Электронный ресурс] URL: https://www.jenkins.io/doc/

5. Github Actions Documentation [Электронный ресурс] URL: https://docs.github.com/en/actions

6. Buddy Documentation [Электронный ресурс] URL: https://buddy.works/docs

7. Метод CI/CD: чистый код и строгая дисциплина [Электронный ресурс] URL: https://habr.com/ru/company/automacon/blog/671918/

8. Когда и зачем нужен CI/CD [Электронный ресурс] URL: <https://habr.com/ru/company/southbridge/blog/649027/>

9. [Электронный ресурс] URL: <https://habr.com/ru/company/ruvds/blog/522334/>

10. [Электронный ресурс] URL: <https://about.gitlab.com/devops-tools/github-vs-gitlab/>

11. [Электронный ресурс] URL: https://softlist.biz/articles/gitlab-vs-github-kto-luchshe-d/