Система управления тестированием (Test Management System) — это программное обеспечение, которое помогает организовать и управлять процессом тестирования. Она предоставляет средства для создания, выполнения, отслеживания и отчетности по тест-кейсам и другой связанной информации, такой как дефекты и требования.

Вот некоторые основные функции и возможности, которые предоставляет система управления тестированием:

1. Управление тест-кейсами: Система позволяет создавать и организовывать тест-кейсы в структурированном виде. Вы можете определить входные данные, ожидаемые результаты, шаги выполнения и другую информацию для каждого тест-кейса.

2. Планирование тестирования: Вы можете определить план тестирования, распределить тест-кейсы по различным релизам или итерациям, а также назначить ответственных за их выполнение.

3. Выполнение тестирования: Система позволяет запускать тест-кейсы и отмечать результаты их выполнения. Вы можете фиксировать пройденные и неудачные тесты, а также вести отметки о проблемах или замечаниях, связанных с выполнением тестов.

4. Управление дефектами: Система позволяет отслеживать и управлять дефектами, обнаруженными в процессе тестирования. Вы можете создавать новые дефекты, присваивать им статусы, отслеживать их исправления и проверять результаты.

5. Отчетность: Система предоставляет возможность генерировать различные отчеты о процессе тестирования, такие как сводные таблицы по выполненным тестам, статистика по дефектам и покрытию тестирования. Это помогает визуализировать прогресс тестирования и делать информированные решения.

6. Интеграция с другими инструментами: Некоторые системы управления тестированием могут интегрироваться с другими инструментами разработки, такими как системы контроля версий, трекеры задач или автоматизированные инструменты тестирования, для обеспечения более гладкого и эффективного рабочего процесса.

Наличие системы управления тестированием способствует повышению эффективности и качества процесса тестирования, а также облегчает организацию и координацию работы внутри команды тестирования.

**Л**

Test IT TMS (Test Management System) - это программное обеспечение, предназначенное для управления и организации процесса тестирования программного обеспечения. Он помогает команде разработчиков и тестировщиков эффективно планировать, создавать, выполнять и отслеживать тестовые сценарии и результаты тестирования.

Преимущества использования Test IT TMS:

1. Централизованное хранение информации: Test IT TMS позволяет хранить исходные коды, тестовые сценарии, техническую документацию и другие данные в одном месте. Это облегчает доступ к информации и обеспечивает легкость совместной работы между разработчиками и тестировщиками.

2. Планирование и управление тестовыми проектами: С использованием Test IT TMS можно создавать планы тестирования, определять области тестирования и назначать задачи разработчикам и тестировщикам. При необходимости можно отслеживать прогресс выполнения задач и контролировать время выполнения проекта.

3. Создание и выполнение тестовых сценариев: Test IT TMS предоставляет возможность создания тестовых сценариев с использованием различных методик и параметров. Это позволяет обеспечить максимальное покрытие кода и выявить возможные ошибки в программном обеспечении. После создания сценариев можно легко выполнить тестирование, а Test IT TMS автоматически отслеживает результаты.

4. Отчетность и анализ результатов тестирования: Test IT TMS предоставляет множество инструментов для анализа результатов тестирования. Он автоматически генерирует подробные отчеты о прогрессе и качестве тестирования, позволяет идентифицировать и отслеживать ошибки, а также проводить анализ эффективности тестового набора и роста качества программного обеспечения.

5. Интеграция с другими инструментами разработки: Test IT TMS позволяет интегрироваться с другими популярными инструментами разработки, такими как системы управления версиями и отслеживания ошибок. Это облегчает совместную работу и обмен информацией между различными процессами разработки.

Редактор тестовых сценариев в Test IT TMS (Test Management System) - это инструмент, который позволяет создавать, редактировать и управлять тестовыми сценариями для проведения тестирования.

**Л**

С помощью редактора тестовых сценариев вы можете:

1. Создавать новые тестовые сценарии: Вы можете создавать новые тестовые сценарии с нуля или использовать уже существующие шаблоны. Вы можете указать название сценария, описание, предусловия, шаги тестирования и ожидаемые результаты.

2. Редактировать существующие тестовые сценарии: Вы можете вносить изменения в уже созданные тестовые сценарии, чтобы обновить информацию или внести исправления. Вы можете изменить название, описание, предусловия, шаги тестирования и ожидаемые результаты.

3. Управлять версиями тестовых сценариев: Test IT TMS предоставляет возможность отслеживать и управлять версиями тестовых сценариев. Вы можете сохранять различные версии сценариев, возвращаться к предыдущим версиям и просматривать историю изменений.

4. Использовать параметры и переменные: Редактор тестовых сценариев позволяет использовать параметры и переменные в сценарии для более гибкой настройки и повторного использования. Вы можете задать параметры, такие как имя пользователя или пароль, и затем использовать их в разных сценариях.

5. Группировать и структурировать сценарии: Вы можете группировать и организовывать тестовые сценарии в проекте. Это позволяет легко найти нужный сценарий и сохранить четкую структуру проекта.

6. Экспортировать и импортировать сценарии: Test IT TMS позволяет экспортировать и импортировать тестовые сценарии в различных форматах, что упрощает совместную работу с другими системами и инструментами.

Редактор тестовых сценариев в Test IT TMS обеспечивает удобный и гибкий способ управления тестированием и повышает эффективность работы команды тестирования.

**Л**

Библиотека тестов в Test IT TMS (Test Management System) представляет собой центральное хранилище всех тестовых артефактов, которое обеспечивает удобное управление и организацию тестовых случаев, наборов тестов и других элементов тестирования.

Основные функции и возможности библиотеки тестов в Test IT TMS включают:

1. Хранение тестовых артефактов: Библиотека тестов позволяет хранить все необходимые тестовые объекты, такие как тест-кейсы, наборы тестов, шаблоны тестовых сценариев и другие. Все артефакты могут быть удобно организованы и структурированы в соответствии с требованиями проекта.

2. Поиск и фильтрация: Благодаря функциям поиска и фильтрации, библиотека тестов позволяет быстро находить нужные тестовые артефакты по различным критериям, таким как название, ключевые слова или атрибуты. Это упрощает процесс поиска и повторного использования существующих тестов.

3. Переиспользование тестовых артефактов: Библиотека тестов позволяет переиспользовать уже существующие тестовые артефакты. Вы можете создавать наборы тестов, которые содержат несколько тест-кейсов, и использовать их в разных проектах или при повторном тестировании. Это помогает сократить время и усилия, затрачиваемые на создание новых тестов.

4. Управление версиями и историей изменений: Библиотека тестов в Test IT TMS обеспечивает управление версиями тестовых артефактов. Вы можете сохранять различные версии тестовых артефактов, вносить изменения и отслеживать историю изменений каждого артефакта.

5. Кастомизация и настройка: Test IT TMS позволяет кастомизировать библиотеку тестов в соответствии с потребностями проекта. Вы можете создавать собственные поля и атрибуты для тестовых артефактов, чтобы добавить дополнительные данные и информацию.

6. Интеграция с другими модулями: Библиотека тестов тесно интегрируется с другими модулями Test IT TMS, такими как редактор тестовых сценариев и система управления тестированием, что обеспечивает единый и централизованный процесс управления тестированием.

Благодаря библиотеке тестов в Test IT TMS, команды тестирования могут эффективно управлять и организовывать все аспекты тестирования, повышая производительность и качество тестирования проектов.

**Л**

Кастомизация позволяет настроить и адаптировать систему к нуждам конкретного проекта и команды тестирования.

В Test IT TMS есть несколько основных функций, которые помогают в кастомизации проектов:

1. Настройка ролей и прав доступа: Администратор системы может создавать различные роли пользователей и настраивать их права доступа к различным функциям системы. Это позволяет определить уровень доступа и функционал каждого пользователя в соответствии с его ролями и обязанностями.

2. Настройка рабочего процесса: Test IT TMS предоставляет возможность настраивать рабочий процесс, создавая тестовые циклы и задачи, определяя статусы и этапы выполнения задач, а также устанавливая время выполнения и сроки для каждой задачи. Это позволяет адаптировать систему под определенные методологии разработки или установить собственные рабочие процессы.

3. Создание собственных полей и атрибутов: Test IT TMS позволяет создавать собственные поля и атрибуты для различных элементов системы, таких как проекты, задачи, тест-кейсы и т. д. Это позволяет добавить дополнительные данные и информацию, которая может быть важна для конкретного проекта или команды тестирования.

4. Конфигурация отчетов и аналитики: Test IT TMS предоставляет возможность создания кастомизированных отчетов и аналитических дашбордов. Администратор системы может настраивать отображение данных, выбирать нужные метрики и показатели для отслеживания и анализа процесса тестирования.

Кастомизация проектов в Test IT TMS позволяет создать индивидуальную систему управления тестированием, которая идеально подходит для конкретных потребностей и требований проекта. Это повышает эффективность работы команды тестирования и улучшает контроль и отчетность о процессе тестирования.

**Л**

Интеграция автотестов в Test IT TMS позволяет автоматизировать выполнение тестовых сценариев и интегрировать результаты автотестирования в общую систему управления тестированием. Вот основные функции и возможности интеграции автотестов в Test IT TMS:

1. Загрузка автотестов: Test IT TMS позволяет загружать автотесты из различных источников, таких как системы автоматизации тестирования (например, Selenium, JUnit, TestNG) или другие инструменты автоматизации.

2. Управление автотестами: Вы можете организовать автотесты в структурированный набор тестовых случаев и группировать их по модулям, функциональности или другим критериям. Test IT TMS предоставляет возможность создания иерархической структуры тестов, что упрощает организацию и навигацию по автотестам.

3. Планирование и запуск автотестов: Test IT TMS позволяет планировать выполнение автотестов в определенное время или по расписанию. Вы можете указать параметры запуска, такие как целевой окружение, конфигурацию тестируемой системы, набор данных и другие настройки.

4. Отслеживание результатов: Test IT TMS отображает результаты выполнения автотестов, включая пройденные, не пройденные или пропущенные тесты. Вы можете просматривать подробности каждого теста, включая журнал выполнения, скриншоты ошибок и другую информацию.

5. Интеграция с отчетами и аналитикой: Test IT TMS предоставляет возможность создания отчетов и аналитических дашбордов на основе результатов автотестирования. Вы можете отслеживать общую статистику прохождения автотестов, тренды, дефекты и другую полезную информацию.

6. Уведомления и оповещения: Test IT TMS позволяет настраивать уведомления и оповещения о результатах автотестирования. Вы можете задать получателей, типы уведомлений (например, электронная почта, SMS) и условия для отправки уведомлений при определенных событиях, таких как неудачное прохождение автотеста или завершение выполнения всех тестов.

7. Интеграция с системами управления дефектами: Test IT TMS обеспечивает интеграцию с системами управления дефектами, такими как Jira, Bugzilla или другими популярными инструментами. Вы можете создавать дефекты прямо из интерфейса Test IT TMS, указывать связанные тестовые случаи и отслеживать статус исправления дефектов.

Интеграция автотестов в Test IT TMS помогает централизованно управлять тестированием, повышает эффективность и улучшает отчетность о процессе тестирования.

**Л**

Расширенный PUBLIC API (интерфейс прикладного программирования) может предоставлять ряд преимуществ пользователям, включая следующие:

1. **Интеграция с другими системами**: Расширенный PUBLIC API позволяет интегрировать Test IT TMS с другими приложениями и системами в вашей организации. Это может быть полезно для автоматизации процессов и обмена данными между системами.
2. **Автоматизация задач**: API позволяет создавать собственные сценарии автоматизации и выполнять различные задачи в Test IT TMS без необходимости ручного вмешательства. Например, вы можете автоматизировать создание тестовых сценариев, добавление тест-кейсов, запуск тестов и получение результатов.
3. **Расширенные возможности отчетности**: Используя API, вы можете получать дополнительные данные и метрики из Test IT TMS для создания более подробных и настраиваемых отчетов и анализа результатов тестирования.
4. **Управление данными**: API позволяет управлять данными в Test IT TMS, включая создание, обновление и удаление записей, таких как тест-кейсы, дефекты и задачи.
5. **Расширенные возможности автоматизации тестирования**: Если Test IT TMS включает возможности автоматизации тестирования, то через API можно управлять этими сценариями автоматизации, запускать их и анализировать результаты.
6. **Пользовательские интеграции и плагины**: С помощью API вы можете создавать собственные интеграции и плагины для Test IT TMS, чтобы расширить его функциональность и адаптировать под свои нужды.
7. **Улучшенная эффективность и производительность**: Автоматизация задач и интеграция с другими системами может значительно повысить эффективность работы и снизить риск ошибок.

**Л**

Вот основные функции и возможности планов тестирования в Test IT TMS:

1. Создание планов тестирования: Вы можете создать планы тестирования, определяя цели, ожидаемые результаты и основные шаги для каждого тестового случая.

2. Организация тест-кейсов: Вы можете структурировать тест-кейсы внутри планов тестирования и группировать их по модулям, компонентам или другим критериям для удобного управления.

3. Автоматическое назначение тест-кейсов: Test IT TMS может автоматически назначать тест-кейсы на исполнение разным членам команды тестирования, оптимизируя процесс распределения задач.

4. Управление ресурсами: Вы можете указывать ресурсы (людей, оборудование, инструменты и т. д.), необходимые для выполнения тестовых случаев, и просматривать доступность ресурсов для планирования.

5. Расписание тестирования: Test IT TMS позволяет создавать расписание выполнения тестовых случаев, устанавливать даты и временные интервалы для проведения тестирования.

6. Отслеживание прогресса: Вы можете отслеживать прогресс выполнения планов тестирования, видеть, сколько тест-кейсов выполнено, сколько осталось и какое время потребуется для завершения.

7. Сообщения и уведомления: Test IT TMS предоставляет возможности отправки сообщений и уведомлений о задачах тестирования, оповещая команду о новых заданиях, изменениях или сроках.

8. Управление ошибками: Вы можете записывать и отслеживать ошибки, обнаруженные в процессе тестирования, и связывать их с соответствующими тест-кейсами для последующего исправления.

9. Анализ результатов: Test IT TMS предоставляет возможности анализа результатов тестирования, генерации отчетов и статистики по выполненным тест-кейсам, общему прогрессу и другим метрикам.

**Л**

Вот основные функции и возможности, связанные с отчетами и метриками в Test IT TMS:

1. Создание пользовательских отчетов: Test IT TMS позволяет пользователям создавать настраиваемые отчеты, соответствующие их потребностям. Вы можете выбрать нужные поля, фильтры и сортировку данных, чтобы получить требуемую информацию.

2. Предопределенные отчеты: Система предлагает некоторое количество предварительно созданных отчетов, которые обычно используются при управлении процессом тестирования. Это может быть отчет о прогрессе тестирования, отчет о дефектах, отчет о покрытии тестами и другие.

3. Метрики и статистика: Test IT TMS собирает данные о тестировании и предоставляет различные метрики и статистические показатели. Вы можете получить информацию о количестве выполненных тестов, процентном покрытии, количестве найденных дефектов, времени выполнения и других параметрах.

4. Графики и диаграммы: Для визуализации данных Test IT TMS предоставляет возможность создания графиков и диаграмм. Это может быть столбчатая диаграмма, круговая диаграмма, линейный график и т. д. Визуальное представление данных помогает лучше понять статус и результаты тестирования.

5. Планирование и мониторинг прогресса: Test IT TMS позволяет планировать и отслеживать прогресс выполнения тестовых заданий. Вы можете проверить, сколько тестов было запланировано, сколько выполнено и сколько осталось. Это помогает управлять процессом тестирования и контролировать его эффективность.

6. Экспорт данных: В Test IT TMS можно экспортировать данные из отчетов и метрик в различные форматы, такие как Excel, CSV или PDF. Это позволяет сохранять результаты тестирования или делиться ими с другими заинтересованными сторонами.

Эти функции и возможности помогают управлять информацией о тестировании, анализировать результаты и принимать решения на основе надежных данных.

**Л**

Atlassian Confluence — система для совместной работы с информацией, которая подходит рабочим командам независимо от их размера и типа. Это программное обеспечение, с помощью которого удобно создавать и редактировать страницы с текстом, изображениями, инфографикой и прикрепленными файлами. Широко используется для создания технических текстов и проектной документации, разработки корпоративных информационных порталов.

Confluence - это платформа для совместной работы над документацией и знаниями внутри компании. Он предоставляет ряд функций и возможностей для организации работы и обмена информацией внутри команды.

Программа построена на вики-движке и внешне похожа на онлайн-офис по аналогии с Google Документами. Страницы могут редактировать разные участники, которым предоставлен соответствующий доступ. История правок сохраняется — при желании можно откатить изменения, вернувшись к определенной правке, а также узнать, кто именно был ее автором.

Контент размещается в так называемых «пространствах» — контейнерах для страниц. Система позволяет структурировать данные по проектам и командам, создавать страницы с уникальной структурой и внешним видом. Для этого можно использовать готовые шаблоны или придумывать проект «с нуля».

Структурируйте данные об эффективности рекламных кампаний и работе маркетинговых инструментов с помощью Сквозной аналитики Calltouch. Получайте простые и удобные отчеты, в которых все нужные вам метрики будут в одном окне. На основании этой информации вы оптимизируете рекламный бюджет и увеличите рентабельность рекламы.

## **Функции программы**

Базовые функции программного обеспечения позволяют:

* **Создавать страницы**. Любую из них можно легко и быстро отредактировать.
* **Структурировать информацию**. База знаний Confluence может состоять из неограниченного количества рабочих пространств. Вы можете устанавливать связи между ними через индексы, перекрестные ссылки, теги.
* **Получать уведомления о новостях и изменениях**. Это полезно, если вы работаете в команде, но хотите быть в курсе свежих правок, касающихся только какого-то конкретного проекта.
* **Искать нужную информацию**. Поиск работает по всей базе данных.

Расширенный функционал предоставляет дополнительные возможности по работе с контентом. Вы сможете создавать блоги, новости, заметки и бюллетени. Информация в этом случае будет привязана ко времени создания.

Также расширенные возможности позволяют:

* **Работать с документами**. Их можно присоединять, просматривать, отслеживать изменения в них и использовать для поиска нужных данных.
* **Проводить дискуссии**. Возможности программного обеспечения позволяют обсуждать работу над проектом с участниками команды.
* **Управлять проектами**. Раздел Confluence можно подключить к Jira Software.
* [**Создавать персональные пространства**](https://atlassian.engexp.ru/docs/confluence/prostranstva/lichnoe-prostranstvo-v-confluence/#:~:text=%D0%94%D0%BB%D1%8F%20%D1%81%D0%BE%D0%B7%D0%B4%D0%B0%D0%BD%D0%B8%D1%8F%20%D0%BB%D0%B8%D1%87%D0%BD%D0%BE%D0%B3%D0%BE%20%D0%BF%D1%80%D0%BE%D1%81%D1%82%D1%80%D0%B0%D0%BD%D1%81%D1%82%D0%B2%D0%B0%20%D0%BD%D0%B0%D0%B6%D0%BC%D0%B8%D1%82%D0%B5,%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82%D0%BD%D1%8B%D0%BC%20%D0%BD%D0%B0%D0%B1%D0%BE%D1%80%D0%BE%D0%BC%20%D0%B8%D0%BD%D1%81%D1%82%D1%80%D1%83%D0%BC%D0%B5%D0%BD%D1%82%D0%BE%D0%B2%20%D0%B8%20%D1%81%D0%B2%D0%B5%D0%B4%D0%B5%D0%BD%D0%B8%D0%B9.?utm_source=blog.calltouch.ru&utm_medium=referral&utm_campaign=Что такое Confluence и как работает программа). Сформируйте комфортную для работы личную среду с нужным набором инструментов и сведений.

Вы можете подключить к Confluence внешние системы, добавить недостающие макросы и пользовательские функции, [интегрировать](https://www.calltouch.ru/blog/poyavilas-integracziya-s-nethouse-kak-podklyuchit-sajt-k-analitike/?utm_source=blog.calltouch.ru&utm_medium=referral&utm_campaign=Что%20такое%20Confluence%20и%20как%20работает%20программа) стандартные офисные инструменты.

На коммерческих сайтах тоже присутствуют функциональные графические элементы — виджеты обратного звонка, чат-боты и прочие. Их можно оформить в соответствии с фирменным стилем, от чего дизайн главной страницы только выиграет. Используйте на своем сайте Виджеты Calltouch, с помощью которых посетители выберут наиболее удобный способ связи с вами: закажут обратный звонок, оставят умную заявку или напишут в мессенджер. Чутко реагируйте на потребности потенциальных клиентов и не упускайте ни одного обращения

Права доступа в Atlassian Confluence устроены следующим образом. Есть три уровня — страницы, пространства и глобальный. Права доступа можно определить для каждого из них — например, в соответствии с иерархией, действующей в вашей компании для всех проектов или индивидуально для каждого из них.

Atlassian Confluence выделяет три группы пользователей:

* **Системный администратор**. Это тот, кто установил Atlassian Confluence. Осуществляет все функции администрирования, в том числе предоставление прав другим пользователям.
* **Администраторы Confluence**. Это пользователи, которые имеют большую часть администраторских функций.
* **Пользователи Confluence**. Это группа, в которую добавляются пользователи по умолчанию — в том числе и незарегистрированные, если нужно предоставить им доступ к работе над контентом.

Чтобы добавлять информацию на страницы, используют стандартные инструменты. Они находятся в верхней части страницы на панели редактирования. Файлы, оглавления, диаграммы, отчеты добавляются с помощью макросов.

Для расширения функций Atlassian Confluence удобно использовать платные и бесплатные дополнения. Они публикуются на сайте Atlassian Marketplace. Здесь представлены собственные разработки компании, а также дополнения, созданные сторонними разработчиками.

**Л**

CI/CD (Continuous Integration, Continuous Delivery — непрерывная интеграция и доставка) — это технология автоматизации тестирования и доставки новых модулей разрабатываемого проекта заинтересованным сторонам (разработчикам, аналитикам, инженерам качества, конечным пользователям и др.). Непрерывная интеграция (Continuous Integration, CI) и непрерывная поставка (Continuous Delivery, CD) представляют собой культуру, набор принципов и практик, которые позволяют разработчикам чаще и надежнее развертывать изменения программного обеспечения.

## Принципы CI/CD

Концепция непрерывной интеграции и развертывания относится к [agile-методологиям](https://blog.skillfactory.ru/glossary/agile/) разработки программного обеспечения. Ее основная цель — уделение достаточного внимания бизнес-требованиям, безопасности и качеству кода конечного продукта. В рамках подхода решаются следующие задачи:

* автоматизация последовательной сборки, упаковки и тестирования программных продуктов;
* автоматизация развертывания приложения в различных окружениях;
* минимизация ошибок и уязвимостей программного продукта.

Разработка по методике CI/CD соответствует таким основным принципам:

* **Распределение ответственности.** Задачи и этапы разработки разделяются между членами команды или ее подгруппами (при работе над большим проектом). Рабочий процесс организуется с учетом бизнес-логистики, внедрения сквозных функций, проведения тестов, безопасности хранения данных и т.д.
* **Сокращение рисков.** Каждый разработчик или подгруппа разработчиков должны стремиться минимизировать уязвимости и ошибки на всех этапах разработки. Для этого постоянно контролируется бизнес-логистика, проводится пользовательское тестирование продукта, оптимизируется хранение, обработка данных и т.д.
* **Оптимизация обратной связи.** Успех проекта зависит от того, как работают друг с другом разработчики, клиенты и пользователи. Это влияет на скорость внесения в приложение корректировок и обновлений. Если сборку и тестирование можно автоматизировать, то во многих других операциях требуется участие человека. Чтобы взаимодействие происходило конструктивнее, уменьшается количество посредников между заказчиком, исполнителями и пользователями.
* **Создание рабочей среды.** Для удобства совместной работы у разработчиков должно быть общее рабочее пространство. Помимо основной ветки процесса в нем должна быть побочная – в ней удобнее проводить тестирование, вносить корректировки, отслеживать отказоустойчивость и т.д.

СI/CD представляет собой современную аналогию конвейерного производства. Их объединяют четкое распределение труда, непрерывный, потоковый характер рабочего процесса, параллельное выполнение сразу нескольких задач (например, кодинга и тестирования). Сегодня эта концепция является доминирующей в DevOps.

## Этапы CI/CD

**Написание кода.** Каждый разработчик создает код отведенного ему модуля и тестирует его в ручном режиме. Затем разработанный и проверенный программный блок интегрируется в основной ветке с текущей версией продукта. Как только все модули будут опубликованы в главной ветке, команда переходит к следующему этапу.

**Сборка.** Заранее подобранная система контроля версий запускает автоматизированную сборку и тестирование всего продукта. Триггеры могут быть настроены автоматически или вручную. Автоматическая сборка выполняется с помощью Jenkins или другого сервера непрерывной интеграции.

**Ручное тестирование.** Как только CI-сервер закончит автоматизированную сборку продукта, он передается тестировщикам на проверку. Они используют различные методики тестирования для выявления и устранения ошибок и уязвимостей программы.

**Релиз.** После исправления ошибок вычищенный и отлаженный код переходит на этап релиза для клиентов. Его проверяет заказчик, возможно, с привлечением своих специалистов или ограниченной группы пользователей. По результатам проверки код отправляется на доработку или согласуется.

**Развертывание.** Текущая версия программы размещается на продакшн-серверах разработчика. Заказчик может работать с программой, исследовать ее функции, искать уязвимости.

**Поддержка и отслеживание.** После развертывания приложение становится доступным конечным пользователям. Параллельно этому разработчики выполняют его поддержку и одновременно мониторят реакцию пользователей, анализируют их опыт взаимодействия с программой.

**Планирование.** На основании данных, полученных при изучении пользовательского опыта, разработчик подготавливает план доработок, включающий новые функции, исправление ошибок и т.д. После этого он вносит все корректировки в продукт — и цикл разработки начинается снова.

Таким образом, рабочий процесс по методологии CI/CD включает как последовательные, так и параллельные этапы. Именно для распараллеливания в рабочем пространстве создается побочная ветка — в ней проще вести работу, не вмешиваясь в основной код до тех пор, пока программируемый модуль не будет готов к интеграции. Условно рабочий процесс по методологии CI/CD можно представить в виде следующей схемы:



## Преимущества CI/CD

* **Сокращение сроков разработки.** Методология уменьшает время доработок до нескольких дней, в сложных проектах — недель. Это позволяет разработчикам быстрее тестировать и опробовать нововведения, а затем внедрять их в продукт раньше конкурентов.
* **Отбор перспективных вариантов.** Быстрое тестирование и большое количество итераций позволяют разработчику вовремя отсеивать бесперспективные варианты кода на начальных этапах. Это также способствует экономичному расходованию времени и ресурсов без их распыления на тупиковые направления.
* **Качество тестирования.** Сочетание ручной и автоматизированной проверки позволяет выявлять ошибки на ранних этапах разработки. Это снижает вероятность их накопления на этапе релиза, что еще больше сокращает время работы над проектом.

## Недостатки CI/CD

* **Высокие требования к опыту.** Рабочий процесс в любой компании можно перевести на методологию CI/CD. Однако это требует от разработчиков как знания самой концепции на практическом уровне, так и умения быстро реорганизовать процессы в самой организации. Иными словами, CI/CD имеет достаточно большой порог вхождения в сравнении со многими традиционными методологиями.
* **Сложность постоянного взаимодействия.** Непрерывная интеграция и доставка программного продукта требуют от разработчиков высокой скоординированности действий. На практике это означает, что должно быть отдельное лицо, которое занимается организацией рабочего процесса и налаживанием взаимодействия между членами команды.

## Инструменты для CI/CD

Так как непрерывная интеграция и развертывание подразумевает автоматизацию многих процессов в ходе разработки, для этого созданы различные программные инструменты и сервисы:

* **GitLab.** Эта платформа позволяет управлять хранилищами проекта, документировать результаты тестирования и доработок, анализировать и дополнять функциональность проекта, выявлять и устранять ошибки.
* [**Docker**](https://blog.skillfactory.ru/glossary/docker/)**.** СD-система, позволяющая контейнеризировать проект, то есть упаковать его со всем окружением и зависимостями.
* **Travis-CI.** Сервер, который можно подключать к виртуальным репозиториям [GitHub](https://blog.skillfactory.ru/glossary/github/) с минимальными настройками. Благодаря использованию облачных технологий его не нужно отдельно устанавливать.
* **Jenkins.** Один из самый популярных DevOps-инструментов, совместимый со всевозможными плагинами для адаптации под различные проекты и задачи.
* **PHP Censor.** CI-сервер, автоматизирующий сборку PHP-проектов. Может работать с репозиториями GitLab, Mercurial и другими, с библиотеками для тестирования Atoum, PHP Spec, Behat.

Возможность оперативно вносить изменения, постоянно тестировать и дорабатывать продукт, взаимодействовать не только друг с другом, но и с клиентом — вот что делает концепцию CI/CD популярной среди разработчиков. Сегодня ее понимание и практическое освоение являются важной рекомендацией при разработке как крупных, так и небольших проектов.

**Л**

**CD-конвейер автоматизирует поставку изменений в различные окружения**

Непрерывная поставка — это автоматическое развертывание приложения в целевое окружение. Обычно разработчики работают с одним или несколькими окружениями разработки и тестирования, в которых приложение развертывается для тестирования и ревью. Для этого используются такие CI/CD-инструменты как [Jenkins](https://www.infoworld.com/article/3239666/devops/what-is-jenkins-the-ci-server-explained.html), CircleCI, AWS CodeBuild, Azure DevOps, Atlassian Bamboo, Travis CI.  
  
Типичный CD-конвейер состоит из этапов сборки, тестирования и развертывания. Более сложные конвейеры включают в себя следующие этапы:

* Получение кода из системы контроля версий и выполнение сборки.
* Настройка инфраструктуры, автоматизированной через подход “инфраструктура как код”.
* Копирование кода в целевую среду.
* Настройка переменных окружения для целевой среды.
* Развертывание компонентов приложения (веб-серверы, API-сервисы, базы данных).
* Выполнение дополнительных действий, таких как перезапуск сервисов или вызов сервисов, необходимых для работоспособности новых изменений.
* Выполнение тестов и откат изменений окружения в случае провала тестов.
* Логирование и отправка оповещений о состоянии поставки.

Например, в Jenkins конвейер определяется в [файле](https://jenkins.io/doc/book/pipeline/jenkinsfile/) Jenkinsfile, в котором описываются различные этапы, такие как сборка (build), тестирование (test) и развертывание (deploy). Там же описываются переменные окружения, секретные ключи, сертификаты и другие параметры, которые можно использовать в этапах конвейера. В разделе post настраивается обработка ошибок и уведомления.  
  
В более сложном CD-конвейере могут быть дополнительные этапы, такие как синхронизация данных, архивирование информационных ресурсов, установка обновлений и патчей. CI/CD-инструменты обычно поддерживают плагины. Например, у [Jenkins есть более 1500 плагинов](https://plugins.jenkins.io/) для интеграции со сторонними платформами, для расширения пользовательского интерфейса, администрирования, управления исходным кодом и сборкой.  
  
При использовании CI/CD-инструмента разработчики должны убедиться, что все параметры сконфигурированы вне приложения через переменные окружения. CI/CD-инструменты позволяют устанавливать значения этих переменных, маскировать пароли и ключи учетных записей, а также настраивать их во время развертывания для конкретного окружения.  
Также в CD-инструментах присутствуют дашборды и отчетность. В случае сбоя сборки или поставки они оповещают об этом. При интеграции CD с системой контроля версий и agile-инструментами облегчается поиск изменений кода и пользовательских историй, вошедших в сборку.

**Реализация CI/CD-конвейеров с Kubernetes и бессерверными архитектурами**

Многие команды, использующие CI/CD-конвейеры в облаках используют контейнеры, такие как [Docker](https://www.infoworld.com/article/3204171/what-is-docker-the-spark-for-the-container-revolution.html), и системы оркестрации, такие как [Kubernetes](https://www.infoworld.com/article/3268073/what-is-kubernetes-your-next-application-platform.html). Контейнеры позволяют стандартизировать упаковку, поставку и упростить масштабирование и уничтожение окружений с непостоянной нагрузкой.  
  
Есть множество вариантов совместного использования контейнеров, инфраструктуры как код и CI/CD-конвейеров. Подробнее изучить это вы можете в статьях [Kubernetes with Jenkins](https://medium.com/containerum/configuring-ci-cd-on-kubernetes-with-jenkins-89eab7234270) и [Kubernetes with Azure DevOps](https://cloudblogs.microsoft.com/opensource/2018/11/27/tutorial-azure-devops-setup-cicd-pipeline-kubernetes-docker-helm/).  
  
Архитектура бессерверных вычислений представляет собой еще один способ развертывания и масштабирования приложений. В бессерверном окружении инфраструктурой полностью управляет поставщик облачных услуг, а приложение потребляет ресурсы по мере необходимости в соответствии с его настройками. Например, в AWS бессерверные приложения запускаются через [функции AWS Lambda](https://aws.amazon.com/lambda/features/), развертывание которых может быть [интегрировано в CI/CD-конвейер Jenkins](https://aws.amazon.com/blogs/compute/continuous-integration-deployment-for-aws-lambda-functions-with-jenkins-and-grunt-part-2/) с помощью плагина.

**Л**

## О GitLab

[GitLab](https://gitlab.com) — популярный веб-сервис для совместной разработки и поддержки программного обеспечения. Вы можете работать с [Git-репозиториями](https://doka.guide/tools/version-control/), управлять задачами, обсуждать правки с вашей командой, писать wiki-документацию, оценивать качество, выпускать релизы и даже мониторить работающие программы — и всё это в одном месте.

GitLab CI/CD обеспечивает управление конфигурациями через yaml-файлы, стабильный запуск в различных средах, сборку и выполнение в разных операционных системах. Кроме того, с помощью этого инструмента можно выполнять интеграцию с кластерами Kubernetes и работать с задачами в окружениях Docker.

## Кратко

GitLab CI — инструмент, встроенный в GitLab для автоматизации рутинных задач, возникающих в процессе разработки программного обеспечения. Спектр таких задач огромен и отличается от проекта к проекту, но основные — это [тестирование](https://doka.guide/tools/how-to-test-and-why/), [статический анализ](https://doka.guide/tools/static-analysis/), [проверка стиля написания кода](https://doka.guide/tools/code-style/) и деплой (выпуск) приложения. GitLab CI — конкурент другого популярного инструмента, [GitHub Actions](https://doka.guide/tools/github-actions/). Эти два сервиса во многом похожи, но есть некоторые отличия.

## Пример

Допустим, мы договорились в команде об особых правилах оформления кода при помощи [EditorConfig](https://editorconfig.org/), установили его как дев-зависимость и сделали его доступным с помощью команды npm run editorconfig. Можно запускать проверку каждый раз перед коммитом, но всегда будут ситуации, когда это забудут сделать, и код, оформленный неправильно, попадёт в репозиторий. Здесь приходит на помощь GitLab CI/CD — достаточно создать в корне проекта файл

И теперь каждый раз, когда в репозиторий попадает новый код, он будет проверяться на соответствие правилам, а ошибки будут видны в интерфейсе GitLab.

## Как пользоваться

### Основные понятия

Основной сущностью в GitLab CI/CD является **пайплайн** (pipeline) — конвейер, который может состоять из:

* **джобов** (jobs), описывающих что нужно выполнить;
* **этапов** (stages), указывающих когда или в какой последовательности нужно выполнить джобы.

Джобы в одном этапе обычно выполняются параллельно. Если все джобы завершились успешно, выполнение переходит к следующему этапу и так далее. Если любой из джобов завершился ошибкой, то выполнение останавливается, и весь пайплайн (обычно) считается проваленным.

### Создаём .gitlab-ci.yml

GitLab CI полностью конфигурируется с помощью одного файла в формате [YAML](https://doka.guide/tools/yaml/), который нужно создать в корне проекта — .gitlab-ci.yml.

Джобы часто могут иметь одинаковые свойства, например, образ среды, в которой выполняются действия, предварительные команды и т. д. Чтобы не повторять их каждый раз, нужно объявить их в секции default. Если какому-то джобу нужны другие параметры, можно указать их внутри этого джоба, и они перезапишут глобальные параметры.

В первую очередь нужно указать Docker-образ (подробнее в статье «[Что такое Docker](https://doka.guide/tools/docker/)»), в котором будут выполняться джобы. В большинстве случаев нам подойдёт официальный образ [Node.js](https://doka.guide/tools/nodejs/) node:lts — это означает, что наши команды будут выполняться внутри операционной системы Linux с установленными Node.js, npm и даже Yarn. Про буквы lts можно почитать в разделе про [версионирование Node.js](https://doka.guide/tools/nodejs/#osobennosti-versionirovaniya).

### Задаём подготовительные команды

При работе с CI/CD во фронтенд-проектах чаще всего перед выполнением основного действия необходимо установить зависимости. Для этого мы можем указать их в секции before\_script — эти команды будут выполняться в каждом джобе перед основным действием.

### Указываем этапы

Предположим, что мы хотим запускать сначала проверку кодовой базы с помощью EditorConfig и [Stylelint](https://doka.guide/tools/static-analysis/#linter-dlya-css), а потом, если они обе завершатся успешно, запустить тесты. В этом примере можно выделить два этапа: стиль кода и тесты. Определить этапы можно при помощи ключевого слова stages

### Описываем джобы и задаём команду

Теперь укажем все три джоба. Для этого мы вначале указываем название джоба, указываем его этап при помощи ключевого слова stage и передаём список команд в script. В нашем примере каждый джоб будет запускать по одному npm-скрипту.

