### Слайд 1. Введение

Здравствуйте!  
Тема моего доклада — *“Continuous Integration и Continuous Delivery (CI/CD) в Agile и DevOps”*.

Сегодня мы живём в мире, где скорость и качество разработки программного обеспечения решают всё.  
Компании стремятся выпускать обновления всё быстрее, исправлять ошибки почти мгновенно и при этом не терять стабильность.

Именно поэтому стали популярны подходы **Agile** и **DevOps**, которые позволяют работать гибко и эффективно.  
А сердцем этих подходов является **CI/CD** — система, которая автоматизирует процесс сборки, тестирования и доставки кода в рабочую среду.

**Слайд 2. Что такое CI/CD**

Давайте разберёмся, что такое CI/CD.

**Continuous Integration**, или “непрерывная интеграция”, — это практика, когда разработчики часто объединяют свои изменения в общий репозиторий.  
После каждого коммита система автоматически собирает проект, запускает тесты и проверяет, что код работает корректно.

**Continuous Delivery**, или “непрерывная доставка”, — это следующий этап.  
Он гарантирует, что после успешных тестов код можно быстро и безопасно развернуть на сервере.  
А если мы идём дальше — к **Continuous Deployment** *это практика, при которой****каждое изменение в коде****, прошедшее автоматические тесты,****автоматически разворачивается в рабочей (production) среде****—****без ручного одобрения***., то весь процесс поставки в продакшн происходит полностью автоматически, без участия человека.

Проще говоря, CI/CD — это механизм, который сам проверяет и выпускает продукт. Разработчики только пишут код.

**Слайд 3. Роль CI/CD в автоматизации**

CI/CD — это фундамент современной автоматизации разработки.

Он берёт на себя всё, что раньше делалось вручную:

* собирает проект,
* запускает тесты,
* проверяет качество кода,
* разворачивает приложение на серверах.

Это значит, что ошибки замечаются раньше, а новые версии продукта выходят значительно быстрее.  
Разработчик может несколько раз в день “заливать” изменения — и уже через несколько минут увидеть результат своей работы на тестовом сервере.

Таким образом, CI/CD экономит время, снижает риск ошибок и делает разработку предсказуемой и прозрачной.

**Слайд 4. CI/CD и DevOps**

Теперь давайте посмотрим, как CI/CD связан с **DevOps**.

DevOps — это философия, объединяющая разработку (Dev) и эксплуатацию (Ops).  
Раньше эти команды работали раздельно: одни писали код, другие поддерживали сервера.  
Из-за этого между ними часто возникали конфликты — “у нас всё работает” против “у вас всё упало”.

CI/CD решает эту проблему.  
Он становится **мостом** между Dev и Ops.  
Когда код автоматически проходит все проверки и деплоится, обе команды видят одни и те же результаты, а процесс становится единым.

Таким образом, CI/CD — это не просто технология, это инструмент, который помогает воплотить философию DevOps на практике: быстрее, надёжнее и совместнее.

**Слайд 5. CI/CD и Agile**

Agile — это гибкий подход к разработке.  
Он основан на коротких итерациях, частых релизах и постоянной обратной связи от пользователей.

CI/CD идеально вписывается в эту концепцию.  
Он позволяет:

* автоматизировать сборку и тестирование в каждом спринте,
* быстро выпускать новые версии,
* мгновенно реагировать на замечания заказчика.

Например, после каждой итерации можно автоматически собрать и протестировать новую версию продукта, выложить её на тестовый сервер и показать заказчику.

Так Agile получает свою техническую поддержку — через CI/CD.

**Слайд 6. Популярные инструменты CI/CD**

Для реализации CI/CD существует множество инструментов.  
Самые известные из них — **Jenkins, GitLab CI/CD, GitHub Actions, CircleCI и Azure DevOps**.

* **Jenkins** — это один из старейших и самых мощных инструментов. Он гибкий и расширяемый, благодаря множеству плагинов.
* **GitLab CI/CD** встроен прямо в систему контроля версий GitLab. Это очень удобно для команд, которые уже работают с Git.
* **CircleCI** и **GitHub Actions** популярны благодаря своей простоте и облачному подходу — не нужно настраивать серверы.
* **Azure DevOps** часто используют крупные компании, работающие с Microsoft-технологиями.

Все эти инструменты решают одну задачу — автоматизировать и ускорить путь от кода до релиза.

**Слайд 7. Пример CI/CD пайплайна**

Как выглядит процесс CI/CD на практике?

1. Разработчик коммитит изменения в репозиторий.
2. CI запускает сборку и тесты.
3. Если всё успешно — создаётся артефакт, например Docker-образ.
4. CD автоматически разворачивает его на тестовом сервере.
5. После проверки — деплой в продакшн.

Всё это может занимать минуты, а не дни, как раньше.  
Пайплайн CI/CD — это своего рода “конвейер”, который никогда не останавливается.

**Слайд 8. Пример Jenkins Pipeline**

Вот пример простого пайплайна на Jenkins:

Он содержит три стадии — **Build**, **Test** и **Deploy**.  
На первой стадии код собирается,  
на второй — проходит тесты,  
а на третьей — разворачивается на сервере.

Jenkins делает это автоматически при каждом коммите.  
Это наглядно показывает, как легко CI/CD можно встроить в любой проект.

**Слайд 9. Преимущества внедрения CI/CD**

Теперь поговорим о преимуществах.

* Во-первых, **скорость**. Новые версии можно выпускать хоть каждый день.
* Во-вторых, **качество**. Ошибки обнаруживаются на ранних этапах.
* В-третьих, **надёжность**. Автоматизация снижает влияние человеческого фактора.
* И наконец, **эффективность команды**. Все видят результат своей работы и могут быстрее реагировать на изменения.

CI/CD помогает компаниям выпускать продукт быстрее и при этом улучшать его стабильность.

**Слайд 10. Сложности внедрения**

Однако внедрение CI/CD — это тоже вызов.

Нужно подготовить инфраструктуру, выбрать инструменты, обучить команду.  
Иногда приходится менять подход к разработке, а это непросто.  
Есть и вопросы безопасности — ведь автоматизация должна быть надёжной.

Но, несмотря на все сложности, эффект от внедрения CI/CD окупается многократно.  
После первых успешных релизов команда уже не захочет возвращаться к старой ручной работе.

**Слайд 11. Современные тенденции**

CI/CD не стоит на месте.  
Сегодня активно развиваются новые направления:

* **GitOps** — когда вся инфраструктура описана в Git и управляется как код.
* **AI в CI/CD** — искусственный интеллект помогает прогнозировать сбои и автоматически исправлять ошибки.
* **Контейнеризация и Kubernetes** — позволяют разворачивать приложения в изолированных средах.
* **Облачные CI/CD-сервисы** — избавляют от необходимости настраивать серверы вручную.

Будущее CI/CD — это ещё больше автоматизации, интеллектуальных систем и интеграции с облаками.

**Слайд 12. Заключение**

Подводя итог:

CI/CD — это не просто набор инструментов.  
Это философия, которая объединяет Agile и DevOps, ускоряет разработку, делает релизы надёжными, а продукт — качественным.

В современном мире, где всё меняется очень быстро, **автоматизация — ключ к успеху**.  
CI/CD позволяет командам сосредоточиться не на рутине, а на создании ценности для пользователя.

**Слайд 13. Завершение**

В заключение хочу сказать:

CI/CD — это путь к непрерывному улучшению.  
А компании, которые внедряют эти практики, становятся более гибкими, быстрыми и конкурентоспособными.

Спасибо за внимание!