

CI/CD Overview

ЧАСТИНА 1

Практичний посібник для тестирувальників та інженерів з автоматизації про те, як влаштовані процеси безперервної інтеграції та доставки, яку роль відіграють тести в пайплайні та які інструменти використовуються в сучасних DevOps-командах.

Вступ

Для кого цей матеріал

- Тестувальники та інженери з автоматизації

Всі, хто пише автотести, аналізує їхні результати та підтримує тестові фреймворки.

- Колеги з пайплайнів та релізів

Ті, хто налаштовує та підтримує CI/CD-процеси і відповідає за доставку продукту.

Навіщо це тестувальнику

Запуск автотестів у CI/CD та аналіз результатів пайплайнів – щоденна задача тестувальника. Розуміння того, де і коли виконуються тести та як вони вписуються у випуск продукту, дає змогу ефективніше знаходити дефекти та комунікувати з командою.

План матеріалу

01

Ключові терміни

02

CI / CD / Continuous Deployment

03

Тести в пайплайні

04

Конвеєр та інструменти

05

Висновки

Ключові терміни

Перш ніж говорити про CI/CD, варто зафіксувати кілька базових понять, які будуть використовуватися далі протягом усього матеріалу.

Репозиторій (repository, «репо»)

Місце, де зберігається код проекту разом з історією змін (наприклад, Git). Усі учасники команди беруть код звідти і відправляють туди свої зміни.

Гілка (branch)

Окрема лінія змін у репозиторії. Розробники часто роблять нову фічу в окремій гілці, щоб не зламати основний код.

Злиття, мердж (merge)

Об'єднання змін з однієї гілки в іншу (наприклад, з гілки фічі в основну гілку).

Білд (build)

Процес збирання програми з вихідного коду (компіляція, упаковка залежностей тощо). Результат – готовий до запуску артефакт.

Деплой (deploy)

Розміщення зібраної програми на сервері або в середовищі, де її можуть використовувати.

DevOps

Підхід, що поєднує розробку (development) та експлуатацію (operations): автоматизація збірки, тестування, доставки та підтримки програмного забезпечення.

Пайплайн (pipeline)

Послідовність автоматичних кроків (наприклад: взяти код → зібрати → запустити тести → задеплоїти), які виконуються один за одним.

Що таке CI/CD і навіщо це потрібно

Безперервна інтеграція (CI) і безперервна доставка (CD) – це набір практик у рамках DevOps, які дають змогу команді **частіше і безпечноше** вносити зміни в код і випускати продукт.

Без автоматизації кожна зміна вимагає ручної збірки, ручного запуску тестів і ручного деплою. Це повільно, схильно до помилок і не масштабується. CI/CD переносить ці кроки в автоматичний пайплайн: код після злиття автоматично збирається, тестирується і – залежно від налаштувань – готовується до деплою або автоматично деплоїться.

- **Підсумок:** CI відповідає за те, щоб зміни коду регулярно зливались у спільне місце (репозиторій) і проходили автоматичні збірку та тести. CD відповідає за те, щоб перевірений код можна було швидко і безпечно доставити в потрібне середовище – тестове, стейджинг або прод.



Швидкість

Автоматизовані кроки замінюють повільні ручні процеси.

Безпека

Помилки виявляються автоматично ще до релізу.

Масштабованість

Пайплайн однаково добре працює для малих і великих команд.

Безперервна інтеграція (Continuous Integration)

Навіщо вона потрібна

У сучасній розробці кілька людей одночасно змінюють код у різних гілках. Якщо зливати все рідко і великими об'ємами, зміни часто **конфліктують** між собою. Вирішення таких конфліктів вручну займає багато часу – це іноді називають **«пеклом мерджу» (merge hell)**.

Ідея CI: зливати зміни в спільну гілку часто – наприклад, кілька разів на день – невеликими порціями. Після кожного такого злиття автоматично запускається збірка та набір тестів. Таким чином помилки виявляються швидко, ще до того, як накопичиться великий об'єм змін.

Протилежний підхід (без CI)

Якщо всі пишуть тільки в одну гілку і деплой робиться лише з неї, збірка і тестування часто відкладаються до моменту «великого злиття». Тоді збірка займає довше, тестів стає багато, а виявлені баги важче прив'язати до конкретних змін. CI саме це і змінює.

Як це працює

1 Коміт / пуш

Розробник вносить зміни у свою гілку і відправляє їх у репозиторій.

2 Збірка (Build)

CI-система реагує на новий код та запускає збірку.

3 Автоматизовані тести

Після успішної збірки запускаються тести.

4 Зворотний зв'язок

Результати (успіх/невдача, логи) повертаються команді.



Швидший фідбек

Розробник і тестувальник майже одразу бачать, чи зламав останній коміт збірку або тести.



Менше «пекла мерджу»

Невеликі злиття простіше вирішувати, ніж одне велике в кінці спринту.



Вища якість релізів

Помилки виявляються рано і можуть бути виправлені до релізу.

Безперервна доставка (Continuous Delivery)



Що це таке

Безперервна доставка (CD) – це підхід, при якому після успішної збірки та автоматичних тестів код **завжди** приведений у стан, готовий до розгортання в цільовому середовищі. Тобто артефакт можна в будь-який момент безпечно задеплоїти; рішення про те, *коли* саме виконати деплой на прод, часто приймає людина – натискання кнопки в CI/CD.

Для ефективного CD у пайплайн також зазвичай вбудовано автоматичне розгортання в тестове/стейджинг середовище; реліз у прод може залишатися ручним кроком.



Швидший вихід на ринок

Код постійно готовий до релізу, не потрібно «добивати» його перед релізом.



Постійний цикл зворотного зв'язку

Команда і користувачі можуть оцінювати роботу в тестовому/стейджинг середовищі і вносити правки в наступні ітерації.



Менше рутини

Вирішення конфліктів і підготовка до деплою автоматизовані в пайплайні, команда більше часу витрачає на розробку і тести.



Менший ризик при релізах

Дрібні часті зміни простіше відкотити і зрозуміти, ніж один великий реліз.

CI vs CD vs Continuous Deployment

Ці три поняття часто плутають. Ось чітке розмежування, яке допоможе розставити акценти у вашій роботі з пайплайнами.

1

Continuous Integration

Короткочасні гілки, часті злиття в спільну гілку, автоматична збірка та тести після кожного кроку.

Фокус: стабільність коду та швидкий фідбек.

2

Continuous Delivery

Після успішного CI код завжди в стані «готовий до деплою»; реліз у прод зазвичай **ручний** – кнопка або схвалення.

3

Continuous Deployment

Кожна успішна збірка після тестів **автоматично** потрапляє на прод без жодного ручного кроку.

- ☐ Усі три можуть існувати в одному процесі по черзі: спочатку CI, потім CD, і за бажанням – повністю автоматичний деплой (Continuous Deployment). Основа – саме **безперервна інтеграція**.

Continuous Testing: тести в CI/CD

Під час CI/CD код проходить **серію автоматизованих тестів**. Це називають **continuous testing** – тестування як невід'ємна частина пайплайну. У піраміді тестування рівні описані знизу вгору: від швидких і дешевих до повільних і дорогих.

1

2

3

4

5

1 Навантажувальні / приймальні

Поведінка під навантаженням та стабільність системи.

2 E2E тести

Імітують дії користувача в інтерфейсі – повні сценарії від початку до кінця.

3 Функціональні тести

Перевіряють, чи функціонал відповідає очікуванням: правильні відповіді API, правильна логіка.

4 Інтеграційні тести

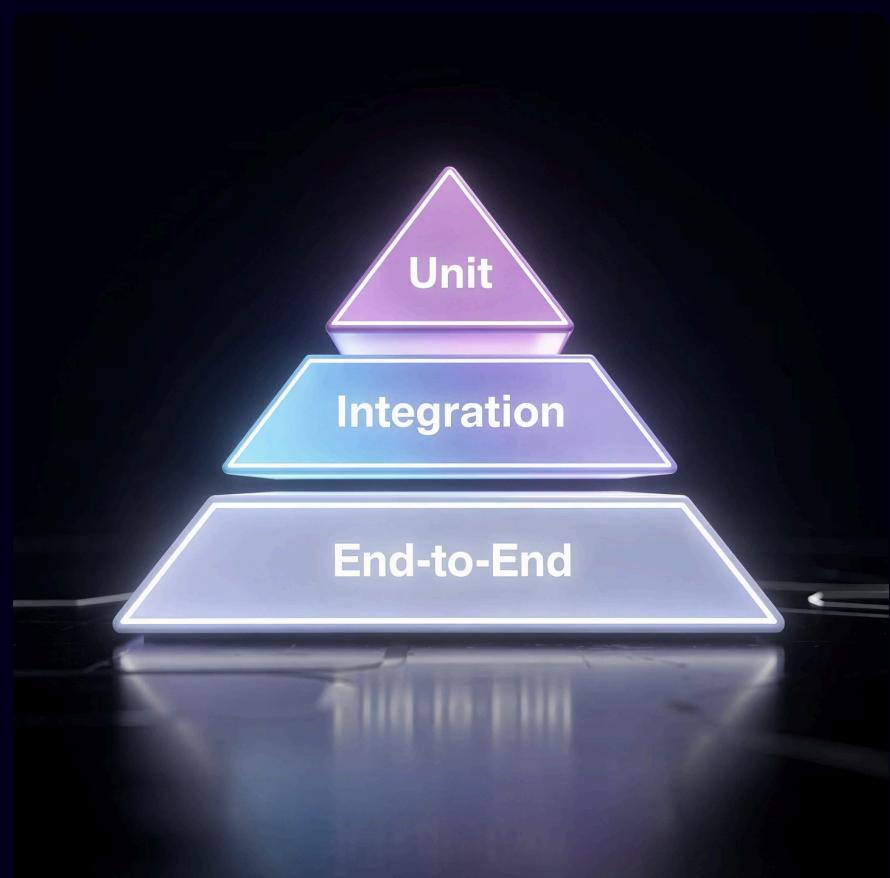
Перевіряють, як кілька компонентів працюють разом.

5 Юніт-тести

Швидкі, їх багато; перевіряють окремі функції та класи ізольовано. Основа піраміди.

❑ **Для початківців:** не обов'язково мати всі рівні одразу. Часто достатньо юніт-тестів + частини інтеграційних або e2e. Важливо, щоб у пайплайні був хоча б один автоматичний рівень тестування – це вже дає відчутну користь від CI.

❑ **Порада:** юніт-тести швидкі і створюють основу для миттєвого фідбеку; UI/e2e тести повільніші і складніші. Баланс між рівнями піраміди залежить від проекту, але постійне автоматизоване тестування в CI/CD – ключовий компонент якості продукту.



CI/CD Pipeline (конвеєр)

CI/CD пайплайн – це послідовність автоматичних кроків, які супроводжують код від репозиторію до збірки, тестів і (за налаштуванням) до деплою в потрібне середовище. Побудова та підтримка CI/CD пайплайну допомагає підтримувати культуру DevOps: швидкі цикли, спільна відповідальність за якість, менший ризик при релізах.



Інструменти CI/CD

Короткий огляд найпопулярніших інструментів на ринку. Вибір залежить від хостингу коду, інфраструктури та звичок команди – не існує єдиного «правильного» рішення.



Jenkins

Популярний open-source сервер автоматизації, часто використовується як основа для CI і CD. Гнучкий завдяки великій екосистемі плагінів, підходить для складних корпоративних пайплайнів.



CircleCI

Хмарний або self-hosted CI/CD з готовими інтеграціями та оркестрацією. Відомий швидкістю та зручністю налаштування для команд будь-якого розміру.



GitLab CI/CD

Вбудований у GitLab – єдине місце для коду, тестів і деплою. Ідеальний вибір для команд, які вже використовують GitLab як репозиторій.



Travis CI

Платформа CI/CD з простим налаштуванням через файл `.travis.yml`, підтримка багатьох мов програмування. Популярний серед open-source проектів.



GitHub Actions

Автоматизація тестування та деплою прямо в репозиторії GitHub. Зручна інтеграція з екосистемою GitHub, великий маркетплейс готових actions.

Висновки

Розуміння CI/CD – це фундаментальна компетенція сучасного тестувальника. Ось ключові тези, які варто запам'ятати після орацювання цього матеріалу.



CI

Часті невеликі злиття коду + автоматична збірка та тести + швидкий зворотний зв'язок. Основа всього процесу.



CD

Код завжди в стані «готовий до деплою». Реліз у прод часто залишається ручним кроком – рішення за командою.



Continuous Deployment

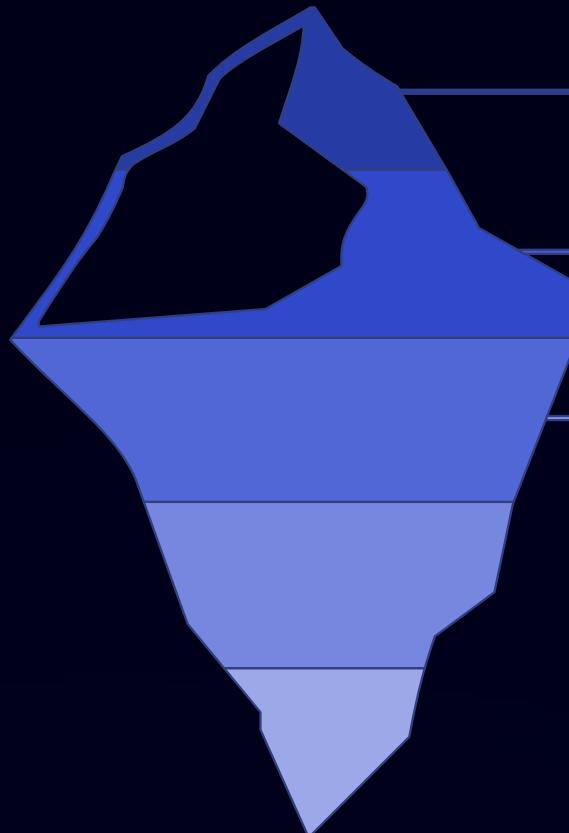
Повністю автоматичний деплой після успішних тестів – без жодного ручного втручання.



Continuous Testing

Ключ до якості. Достатньо одного рівня (юніт або e2e) для початку – далі розширювати піраміду.

Роль тестувальника в CI/CD: оцінювати готовність коду до релізу та швидко знаходити регресії за результатами пайплайнів. Тестувальник – це той, хто дає команді впевненість у якості кожного релізу.



Запустіть тести

Аналізуйте результати

Оцініть готовність