Якість та тестування програмного забезпечення

Лекція №5

Тема: Тестування в CI/CD

ЖИТОМИРСЬКА ПОЛІТЕХНІКА
ДЕРЖАВНИЙ УНІВЕРСИТЕТ

Лектор: асистент кафедри комп'ютерних наук Українець Микола Олександрович

Питання лекції

1. Еволюція підходів до тестування
2. Безперервна інтеграція (CI) та безперервна доставка та розгортання (CD)
3. Місце та роль тестування в CI/CD

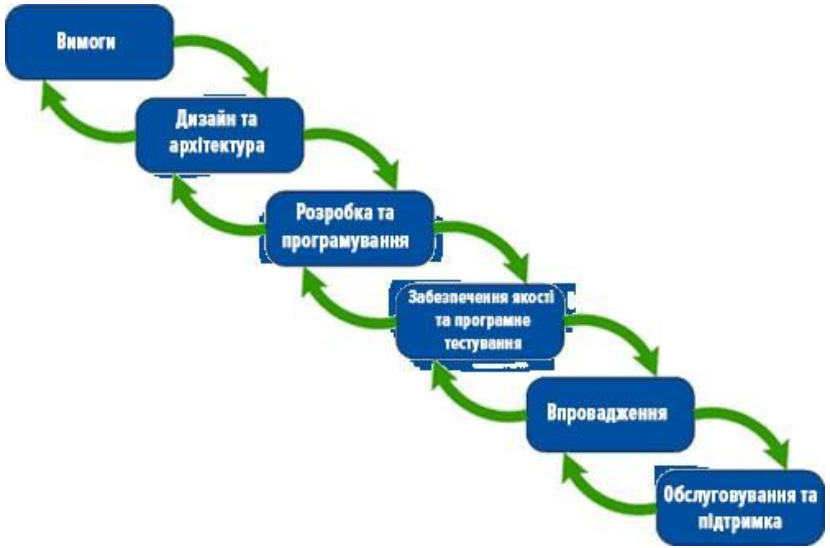
Еволюція підходів до тестування

Під час розробки програмного забезпечення за каскадною моделлю, тестування відбувається після завершення розробки.

Основними проблемами цього підходу є:

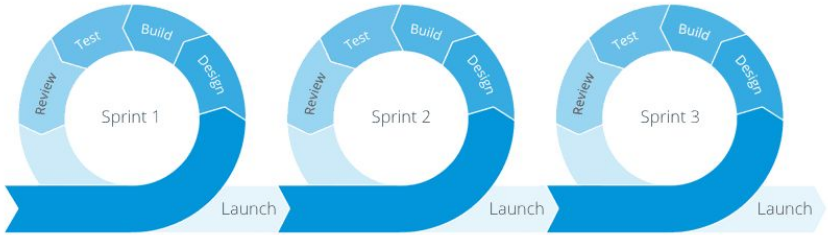
- дефекти виявляються надто пізно;
- високі витрати на виправлення;
- тривалі цикли випусків;
- обмежений зворотний зв’язок між розробниками і тестувальниками.

У таких умовах якість часто розглядається як етап, а не як безперервний процес.

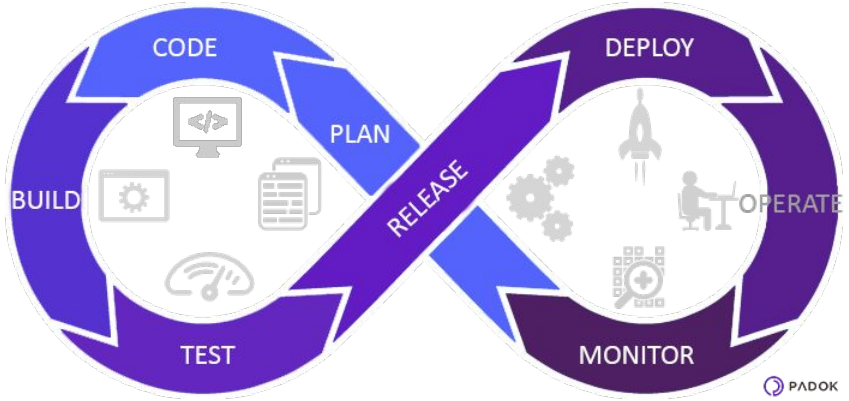


Еволюція підходів до тестування

Agile software development — клас методологій розробки програмного забезпечення, що базується на ітеративній розробці, в якій вимоги та розв'язки еволюціонують через співпрацю між багатофункціональними командами здатними до самоорганізації. Гнучка розробка — засіб для підвищення продуктивності розробників програмного забезпечення.

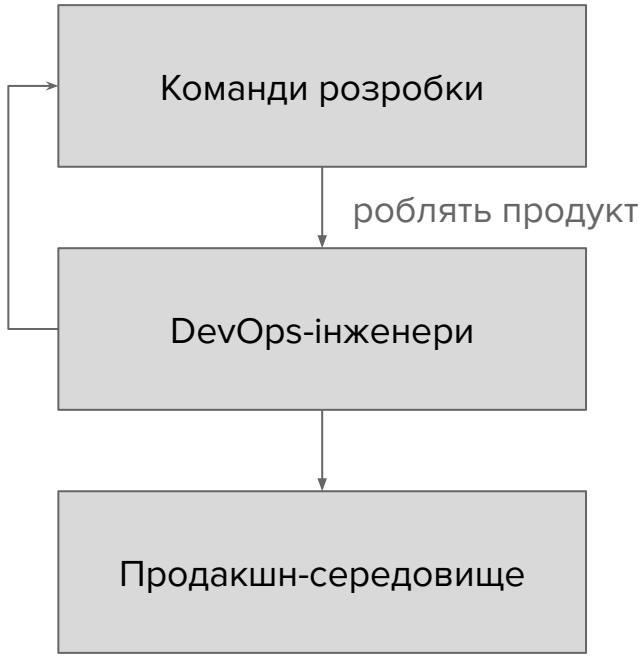


DevOps — це набір практик, інструментів та культурна філософія, що автоматизують та інтегрують процеси між командами розробників програмного забезпечення та IT-фахівцями. Цей підхід робить акцент на розширенні повноважень команд, тісній комунікації та співпраці між ними, а також на технологічній автоматизації.



Еволюція підходів до тестування

DevOps-інженер — це доволі молода професія, яка з’явилась 5-8 років тому. Її назва утворена від слів Developers та Operations. Тобто DevOps-інженери ніби стоять між командами розробників та операційними командами та налагоджують їх взаємодію заради впровадження на проєктах принципів безперервних інтеграції, доставки, розгортання.



Основні завдання DevOps-інженерів:

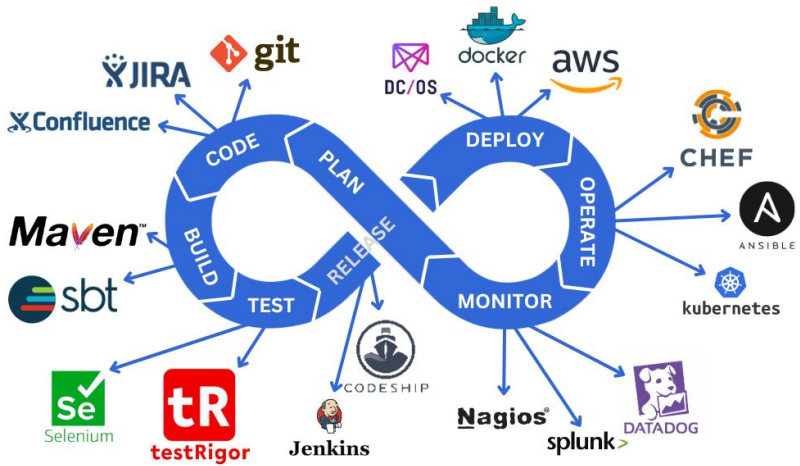
- забезпечення інфраструктури для конвеєрів CI/CD
- налагоджують автоматизацію
- відстежують процеси на кожному з етапів CI/CD
- консультують стейкхолдерів

Безперервна інтеграція (CI) та безперервна доставка та розгортання (CD)

CI/CD (Continuous Integration/Continuous Delivery або Continuous Deployment) — це набір практик у розробці програмного забезпечення, що автоматизують процеси злиття коду, його тестування та розгортання для прискорення та підвищення надійності випуску оновлень.

CI (безперервна інтеграція) означає часте об'єднання коду від різних розробників у спільний репозиторій з автоматичним запуском тестів для виявлення помилок.

CD (безперервна доставка або розгортання) завершує процес, автоматично готуючи код до релізу (доставка) або відправляючи його на продакшн (розгортання), що забезпечує швидше та якісніше постачання продукту користувачам.



Безперервна інтеграція (CI) та безперервна доставка та розгортання (CD)

Основні принципи CI/CD:

- Зниження ризиків у процесі.
- Організація середовища для всіх, хто в ньому працює.
- Швидкий зворотний зв’язок через автоматизацію процесів та мінімізацію посередників.
- Рівномірний розподіл відповідальності між працівниками.

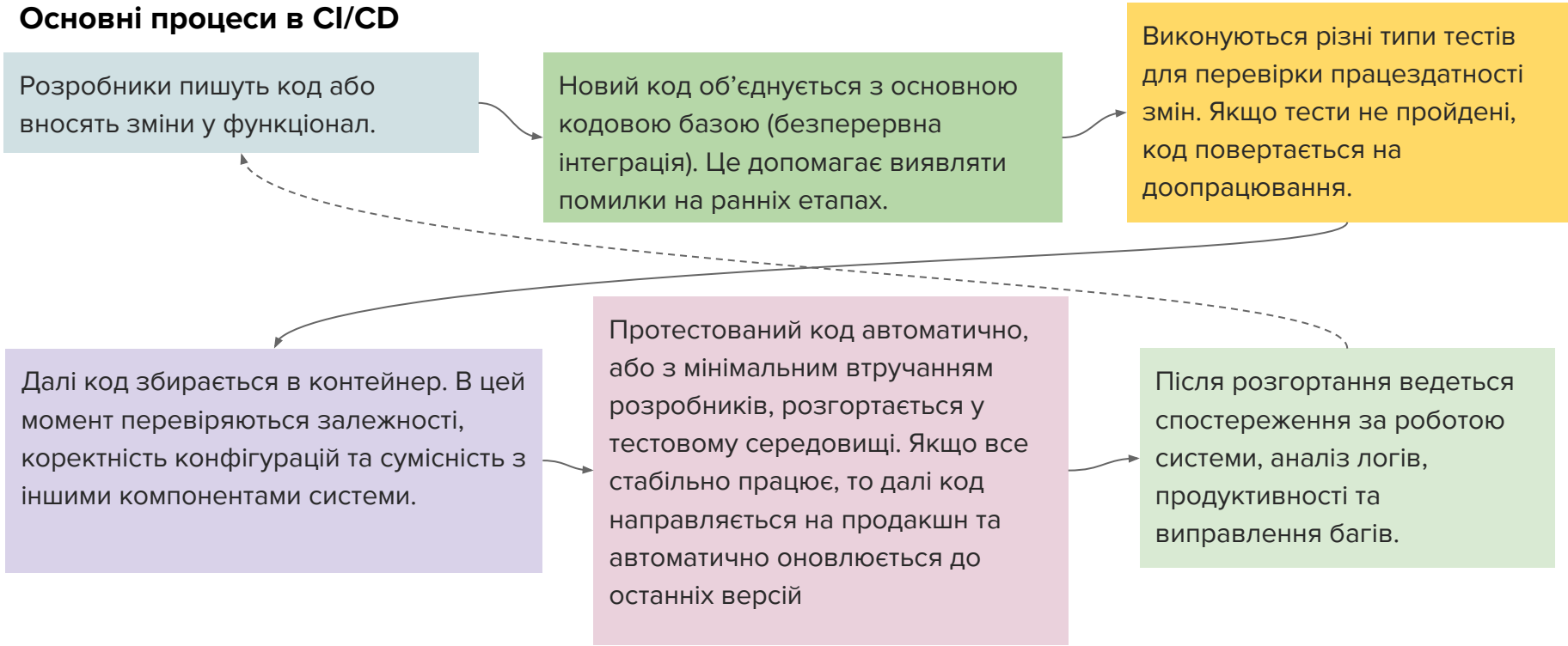
CI означає, що кожна зміна у коді має бути:
об’єднана з основною гілкою (merge);
автоматично зібрана (build);
протестована.

Ціль — раннє виявлення дефектів і забезпечення стабільності коду.

CD — це автоматизація процесу доставки програмного забезпечення:
Continuous Delivery — автоматично готує реліз, але вимагає ручного підтвердження для деплою;
Continuous Deployment — здійснює деплой повністю автоматично.

Безперервна інтеграція (CI) та безперервна доставка та розгортання (CD)

Основні процеси в CI/CD



Безперервна інтеграція (CI) та безперервна доставка та розгортання (CD)

Конвеєр безперервної інтеграції та безперервного розгортання (CI/CD pipeline) — це послідовність встановлених кроків, яких розробники повинні дотримуватися для випуску нової версії програмного забезпечення.

CI/CD-пайплайни — це практика, зосереджена на покращенні доставки програмного забезпечення протягом усього життєвого циклу його розробки за допомогою автоматизації.

У деталях конвеєр CI/CD може відрізнятися від проєкту до проєкту: залежно від програмного забезпечення, яке створюють розробники, самої команди розробників, наявних ресурсів тощо.

Типовий pipeline складається з кількох етапів:

Етап	Завдання
Checkout	Отримання коду з репозиторію
Build	Компіляція, збірка
Test	Запуск юніт, інтеграційних і UI-тестів
Analysis	Аналіз коду, покриття, безпека
Deploy	Розгортання на staging або production
Monitor	Моніторинг, логування

Місце та роль тестування в CI/CD

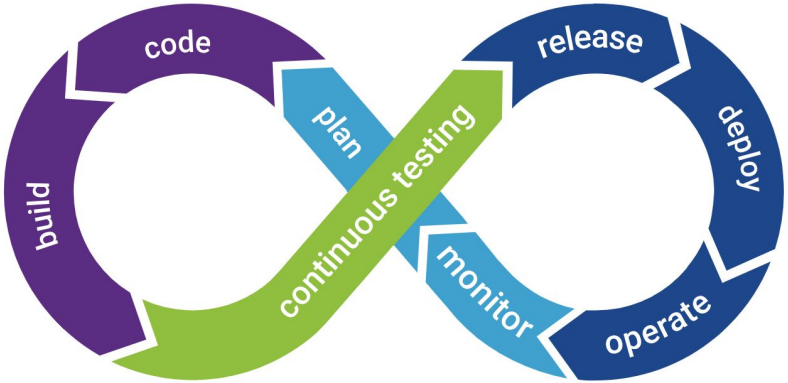
CI/CD стосується не лише розробників — QA-команди теж повною мірою залучені в цій схемі. В ідеалі це мають бути максимально комплексні та, що важливіше, автоматизовані тести коду та продукту загалом.

Етап	Типи тестів	Мета
Build	Unit tests	Перевірка окремих функцій/методів
Integration	Integration tests	Взаємодія між компонентами
Pre-deploy	Smoke / UI / E2E / Regression tests	Перевірка базового функціоналу
Post-deploy	Monitoring, A/B tests, Canary tests	Оцінка роботи системи в реальному середовищі

Місце та роль тестування в CI/CD

Безперервне тестування (Continuous Testing, CT) — це процес розробки програмного забезпечення, під час якого додатки тестуються постійно протягом усього життєвого циклу розробки (SDLC).

Мета безперервного тестування полягає в тому, щоб оцінювати якість програмного забезпечення на всіх етапах SDLC. Це дозволяє отримувати критично важливий зворотний зв'язок на ранніх стадіях, що, у свою чергу, забезпечує вищу якість та прискорює доставку продукту.



Місце та роль тестування в CI/CD

	Shift Left Testing	Shift Right Testing
Головна мета	**Запобігти** дефектам, виявляючи та виправляючи їх якомога раніше.	**Оцінити** продуктивність, надійність та користувацький досвід в реальних умовах.
Коли проводиться?	Протягом етапів дизайну, кодування та інтеграції.	Після розгортання продукту для кінцевих користувачів.
Типи тестування	- Юніт-тести - Інтеграційні тести - Статичний аналіз коду - Тестування API	- A/B тестування - Canary releases - Моніторинг продуктивності - Збір відгуків користувачів - Chaos Engineering
Що виявляє?	Помилки на рівні коду, проблеми з архітектурою та інтеграцією компонентів.	Проблеми з продуктивністю, масштабованістю, користувацьким досвідом та інші дефекти, що проявляються лише в реальному середовищі.
Основна перевага	Значно **знижує вартість** виправлення багів та прискорює загальний час розробки.	Допомагає зрозуміти, як додаток поводиться під **реальним навантаженням** та як з ним взаємодіють користувачі, що покращує бізнес-цінність продукту.

Місце та роль тестування в CI/CD

CI/CD системи: Jenkins, GitHub Actions, GitLab CI/CD, Azure DevOps, CircleCI
Тестові фреймворки: JUnit, NUnit, Pytest, Mocha, Cypress
E2E тестування: Selenium, Playwright, Cypress
Аналіз коду: SonarQube, ESLint, Pylint
Моніторинг: Grafana, Prometheus, Datadog

Місце та роль тестування в CI/CD

Приклад yml файлу для налаштування автоматичного запуску unit тестів після кожного push та pull request:

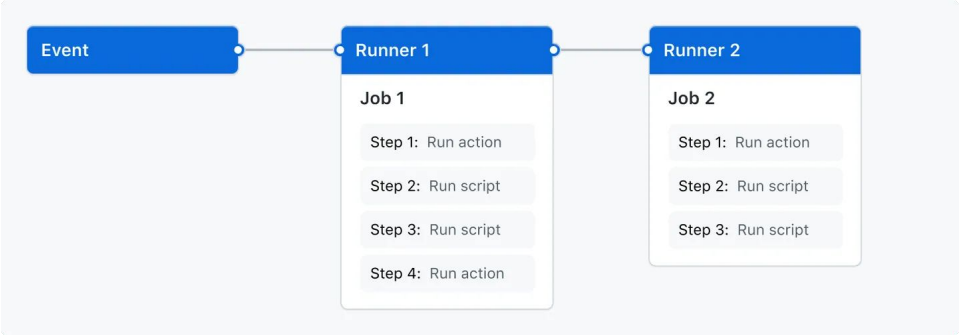
tests.yml

```yaml
name: Run tests
on: [push, pull_request]

jobs:
  test:
    runs-on: ubuntu-latest
    steps:
      - uses: actions/checkout@v3
      - name: Set up Python
        uses: actions/setup-python@v3
        with:
          python-version: '3.11'
      - name: Install dependencies
        run: pip install pytest
      - name: Run tests
        run: pytest
```

структура проекту:

```
my_project/
  ├── main.py
  ├── test_main.py
  └── .github/workflows/tests.yml
```



https://docs.github.com/en/actions/get-started/quickstart

Місце та роль тестування в CI/CD

Типові помилки при впровадженні тестів у CI/CD

- Надмірна кількість повільних E2E-тестів ➜ pipeline стає повільним.
- Відсутність розмежування тестів за рівнями (unit, integration, UI).
- Відсутність моніторингу або quality gates.
- Ігнорування результатів тестів (pipeline “завжди зелений”).

Місце та роль тестування в CI/CD

Основні завдання QA-інженерів у DevOps:

1. Розуміти структуру pipeline і точки запуску тестів.
2. Розробляти сценарії для автоматизованого тестування.
3. Брати участь у розробці стратегій Shift-left і Shift-right тестування.
4. Аналізувати дані моніторингу (quality gates, build logs).
5. Співпрацювати з DevOps-інженерами для покращення процесу.

Навички сучасного тестувальника у DevOps:

- знання Git і CI/CD платформ;
- вміння писати автотести;
- розуміння контейнеризації (Docker);
- знання API-тестування;
- базове розуміння скриптових мов (Python, Bash, PowerShell).

Корисні посилання

1. https://itedu.center/ua/blog/review/what-is-ci-cd/?srsltid=AfmBOoo96Be3jFCcTewF_1Hbuw9018G5m-oqT1eQ0urJOq7xmsgql1sT
2. https://www.redhat.com/en/topics/devops/what-cicd-pipeline
3. https://testgrid.io/blog/ci-cd-test-automation/
4. https://www.browserstack.com/guide/shift-left-vs-shift-right
5. https://www.nixsolutions.com/ua/blog/for-developer/ci-cd-shho-cze-yak-povyazano-z-devops-perevagy-najkrashhi-praktyky/
6. https://www.geeksforgeeks.org/devops/difference-between-agile-and-devops/
7. https://docs.github.com/en/actions/get-started/quickstart
8. https://learn.gitlab.com/ci-awareness-mgr1/modernize-your-ci-cd?_pfses=krPriodZg3eTLEQiFcSSs9xQ

Дякую за увагу!