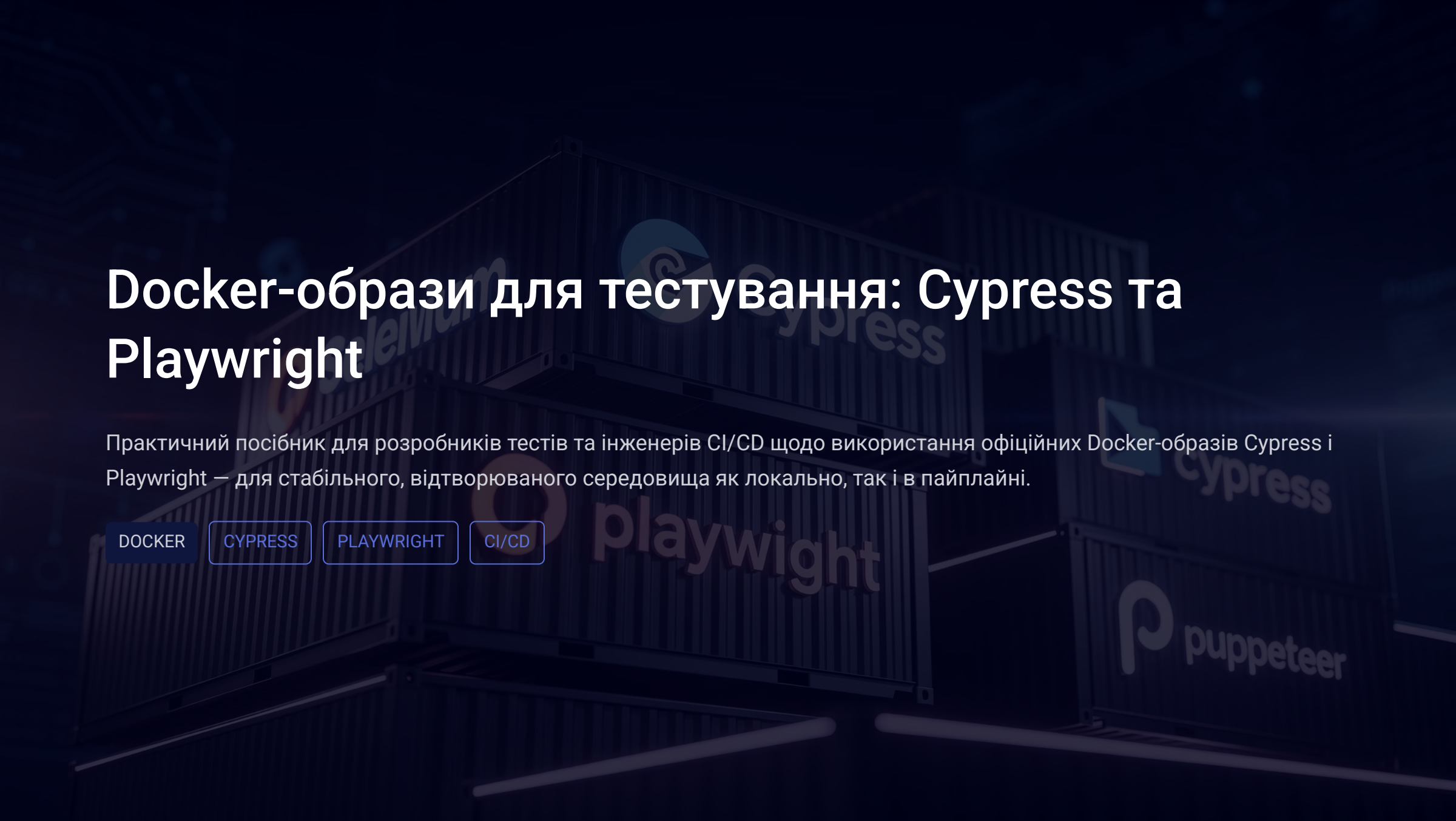


Docker-образи для тестування: Cypress та Playwright

Практичний посібник для розробників тестів та інженерів CI/CD щодо використання офіційних Docker-образів Cypress і Playwright — для стабільного, відтворюваного середовища як локально, так і в пайплайні.

DOCKER CYPRESS PLAYWRIGHT CI/CD

Вступ

Навіщо запускати тести в Docker

Одна з найпоширеніших проблем у команд — розбіжність між локальним середовищем розробника та CI-середовищем. Docker вирішує цю проблему радикально: контейнер містить однакові версії Node.js, браузерів та системних бібліотек незалежно від того, де він запускається — на ноутбуці розробника чи у хмарному пайплайні.

- Однакове середовище (Node.js, браузери, бібліотеки) локально і в CI
- Менше ситуацій «у мене працює, на CI — ні»
- Відтворюваність результатів між членами команди

Що охоплює цей документ

Ми розглянемо готові офіційні образи для двох провідних фреймворків e2e-тестування та покажемо, як запустити локальні тести в контейнері з мінімальними зусиллями.

- Готові образи Cypress (Docker Hub) та Playwright (MCR)
- Практичні команди для локального запуску тестів у контейнері
- Поради щодо версіонування та безпеки

План документа

Образи Cypress

Офіційні образи публікуються на Docker Hub. Є три основні варіанти, кожен з яких орієнтований на різний рівень контролю та зручності:

cypress/base

cypress/base на Docker Hub

Містить залежності ОС, необхідні для запуску Cypress, але **без** встановленого Cypress і без попередньо встановлених браузерів. Підходить, коли ви самі встановлюєте Cypress у своєму Dockerfile і контролюєте версію Node та ОС. Теги вказують на версію Node або ОС, наприклад `cypress/base:18`.

cypress/browsers

cypress/browsers на Docker Hub

ОС + частина браузерів вже встановлена, але **без** самого Cypress. Зручно для кастомних збірок, де ви встановлюєте Cypress у Dockerfile. **Примітка:** образ за замовчуванням може працювати під користувачем root; для безпеки в продакшені часто створюють не-root користувача в Dockerfile.

cypress/included

cypress/included на Docker Hub

ОС + Cypress + браузери вже встановлені. Достатньо змонтувати проект з тестами і запустити Cypress однією командою. Теги відповідають версії Cypress, наприклад `cypress/included:13.6.0`. Найзручніший варіант для **швидкого старту** і запуску в headless або інтерактивному режимі.

Headless-режим — режим без видимого вікна браузера; тести виконуються у фоновому режимі. Саме цей режим найчастіше використовується в CI-пайплайнах, де немає графічного інтерфейсу.

Запуск локальних Cypress-тестів у Docker

Нижче використовується образ **cypress/included** (на прикладі версії 13.6.0). Логіка команд однакова для всіх підтримуваних версій.

Базова команда

З кореня проекту (де лежать тести):

```
docker run -it \
  -v $PWD:/e2e \
  -w /e2e \
  cypress/included:13.6.0
```

Запуск у конкретному браузері

```
docker run -it -v $PWD:/e2e -w /e2e \
  cypress/included:12.8.1 \
  --browser firefox

docker run -it -v $PWD:/e2e -w /e2e \
  cypress/included:12.8.1 \
  --browser chrome
```

Перевірка контейнерів

```
docker ps -a
```

Що робить кожен параметр

→ `docker run`

Створити і запустити контейнер із вказаного образу.

→ `-it`

Інтерактивний режим з терміналом — корисно для перегляду виводу і інтерактивного Cypress.

→ `-v $PWD:/e2e`

Змонтувати поточну директорію хоста в контейнер як `/e2e`. Файли синхронізовані: зміни на хості одразу видно в контейнері.

→ `-w /e2e`

Робоча директорія всередині контейнера, де лежать ваші тести.

Чому краще фіксувати тег версії

❌ Не використовуйте `latest`

Тег `latest` змінюється при виході нових версій. Тести можуть несподівано перейти на іншу версію Cypress і почати поводитися інакше без жодних змін у коді.

✅ Фіксуйте конкретну версію

Наприклад, `cypress/included:13.6.0` гарантує однакове середовище на всіх машинах і в CI. Оновлення — свідома зміна тегу в команді або в CI-конфігурації.

Образи Playwright

Офіційні образи Playwright публікуються в **Microsoft Container Registry (MCR)**, а не на Docker Hub. У образах вже є потрібні залежності ОС та браузери — тести можна запускати так само, як локально, передаючи параметри CLI.

[Playwright на Microsoft Container Registry]

Користувач root і Chromium sandbox

За замовчуванням контейнер виконується від імені **root**. У такому режимі Chromium sandbox часто вимкнено — він недоступний під root. Для e2e-тестів у навчальних і тестових середовищах це прийнятно. Для продакшену рекомендується налаштувати не-root користувача в Dockerfile.

Параметр `--ipc=host` для Chrome

При використанні Chrome у контейнері **рекомендується** додавати `--ipc=host`, інакше Chrome може зіткнутися з проблемами пам'яті (out of memory тощо). Це пов'язано з тим, як Chrome використовує розділювану пам'ять між процесами.

```
docker run -it --rm \
  --ipc=host \
  mcr.microsoft.com/playwright:v1.40.0-jammy \
  /bin/bash
```

Версії ОС у тегах образів

jammy

Ubuntu 22.04 LTS — актуальна довгострокова версія. Рекомендована для нових проєктів.

focal

Ubuntu 20.04 LTS — для зворотної сумісності або специфічних вимог середовища.

Узгодження версій

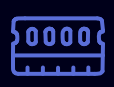
Важливо узгоджувати версію Playwright у контейнері з версією у вашому проєкті (`package.json`), щоб уникнути розбіжностей у поведінці тестів між локальним запуском і CI.

Документація CLI

Усі параметри запуску тестів (фільтри, workers, retries тощо) передаються в контейнер так само, як при локальному запуску. Деталі: [Playwright Test CLI].

Sandbox і безпека

Root-режим вимикає Chromium sandbox. Для тестових середовищ прийнятно, для продакшену — налаштовуйте не-root користувача.



IPC та пам'ять Chrome

Параметр `--ipc=host` дозволяє Chrome використовувати IPC хоста, запобігаючи збоям через нестачу розділюваної пам'яті.

Теги і версії

Обирайте тег, що відповідає потрібній версії Ubuntu (`jammy` або `focal`) та версії Playwright у вашому проєкті.

Запуск локальних Playwright-тестів у Docker

Підхід ідентичний Cypress: монтуємо поточну директорію в контейнер і задаємо робочу директорію. Далі всередині контейнера можна виконувати звичайні команди Playwright або передати їх у `docker run` як команду замість стандартної оболонки.

Команда запуску

```
docker run \
  -v $PWD:/e2e \
  -w /e2e \
  --rm \
  -it \
  --ipc=host \
  mcr.microsoft.com/playwright:v1.40.0-focal
```

Запуск тестів усередині контейнера

```
npx playwright test
```

Або передати команду напряму

```
docker run -v $PWD:/e2e -w /e2e \
  --rm --ipc=host \
  mcr.microsoft.com/playwright:v1.40.0-focal \
  npx playwright test --reporter=list
```

Розбір параметрів команди

`-v $PWD:/e2e`

Монтування поточної папки хоста в директорію `/e2e` контейнера. Зміни у файлах тестів на хості одразу доступні в контейнері без перезбірки образу.

`-w /e2e`

Встановлює робочу директорію всередині контейнера. Усі відносні шляхи у командах будуть відносно цієї директорії.

`--rm`

Автоматично видаляє контейнер після завершення роботи. Зберігає порядок і не залишає «мертвих» контейнерів.

`--ipc=host`

Рекомендується при використанні Chrome. Дозволяє браузеру використовувати IPC хоста, зменшуючи ризик збоїв через нестачу пам'яті.

Примітка щодо --ipc=host: цей параметр потрібен насамперед для стабільної роботи **Chrome** у контейнері (зменшення ризику збоїв через пам'ять). Для Firefox та WebKit він зазвичай не є обов'язковим, але його використання є безпечним загальним правилом для Playwright-контейнерів.

Висновки

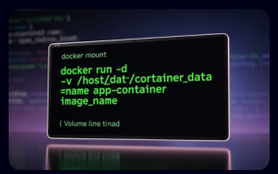
Використання Docker для запуску e2e-тестів усуває класичну проблему невідповідності середовищ і робить результати тестів стабільними та відтворюваними. Ось ключові рекомендації для обох фреймворків:

Cypress: вибір образу

- **cypress/included** — швидкий старт, все вже є у образі
- **cypress/base** або **cypress/browsers** — для кастомних Dockerfile
- Завжди фіксуйте конкретний тег версії

Playwright: ключові нюанси

- Образи з **MCR**, не з Docker Hub
- Для Chrome — обов'язково `--ipc=host`
- Теги **jammy** (Ubuntu 22.04) / **focal** (Ubuntu 20.04)



Спільний підхід

- Монтування: `-v $PWD:/e2e`
- Робоча директорія: `-w /e2e`
- Результат — як при локальному запуску

3	2	1
образи Cypress	**теги ОС Playwright**	**головне правило**
base, browsers, included — для різних сценаріїв використання	jammy (Ubuntu 22.04) та focal (Ubuntu 20.04)	Фіксований тег = однакове середовище локально та в CI

> **Головний принцип:** фіксований тег образу забезпечує однакове середовище локально та в CI, спрощує відтворення проблем і робить оновлення свідомим рішенням команди, а не випадковим наслідком релізу нової версії.