

1. Cypress in CI

Cypress in CI means running your automated tests on a CI server (like Jenkins, GitHub Actions, GitLab CI) whenever code is pushed or a PR is created.

Goal: **catch bugs early**, run tests automatically, and keep test runs consistent across environments.

2. Configuring Cypress in Jenkins, GitHub Actions, GitLab CI

High-level steps are similar in all CI tools:

1. Install dependencies

- Install Node.js
- Run `npm install` or `yarn` to install Cypress and project deps.

2. Add a CI job / pipeline step

- Jenkins: add a stage in `Jenkinsfile` to run `npx cypress run`.
- GitHub Actions: define a workflow (`.yaml` in `.github/workflows`) with a step to run Cypress.
- GitLab CI: define a job in `.gitlab-ci.yml` with script `npx cypress run`.

3. Cache dependencies (optional but recommended)

- Use built-in caching of `node_modules` to speed up runs.

4. Store reports & videos as artifacts

- Configure the CI to upload `cypress/videos` and `cypress/screenshots`.
-

3. Running Tests in Headless Mode

Headless mode runs the browser **without UI**, ideal for CI:

- Use: `npx cypress run` (default headless mode with Electron).
 - To specify browser: `npx cypress run --browser chrome`.
 - Headless mode = **faster, CI-friendly** and doesn't require a desktop environment.
-

4. Reporting and Test Artifacts

Reports and artifacts help you **understand failures after the run**:

- **JUnit / Mochawesome / Cypress Cloud reports**
 - Generate XML/HTML/JSON reports that CI can display in dashboards.
- **Screenshots & videos**
 - Cypress automatically captures screenshots on failure and videos per spec (in `cypress/screenshots` and `cypress/videos`).
- **CI integration**
 - Configure your CI to **archive these folders as build artifacts**, so you can download and review them for debugging.