UnCodemy

Playwright Parallel Testing and Sharding PDF

Playwright runs test files in parallel using worker processes, and fullyParallel lets tests inside one file run in parallel too. Keep tests independent with unique data, use serial mode only when order truly matters, and split big suites across machines with the shard option, then merge the blob reports into one report.

Key takeaways

- Each worker is a separate process with its own browser instance.
- Parallel tests must not share mutable data or depend on order.
- fullyParallel runs tests inside a file in parallel as well.
- Use --shard=x/y to split the suite across jobs.
- Merge blob reports to get one combined HTML report.

What is Playwright parallelism and sharding?

Parallel testing in Playwright means running several tests at the same time in separate worker processes, and sharding means dividing the whole suite into parts that run on different machines.

How does Playwright run tests in parallel?

By default, test **files** run in parallel across worker processes, while tests inside one file run in order in the same worker. Each worker starts its own browser, so there is no shared state between them. The number of workers defaults to a portion of the CPU cores and can be set with `workers` in the config or `--workers=4` on the command line.

How do I parallelise tests inside a file?

Set `fullyParallel: true` in the config or call `test.describe.configure({ mode: 'parallel' })` in a file. Now every test can run in a different worker. This gives the best speed, but it needs independent tests: no test may rely on data created by another.

What makes a test safe to run in parallel?

Rule	Example
Unique data	Add a random suffix to names and emails

Rule	Example
No shared records	Create data per test through the API
No dependence on order	Do not assume the previous test ran
Own authentication	Use storage state files, not one shared session that changes
Clean up	Delete what you create
If the application resets a shared account on login, give each worker its own account using a worker-scoped fixture.	

When is serial mode needed?

Use `test.describe.configure({ mode: 'serial' })` for a short flow in which steps genuinely build on each other, such as a wizard. In serial mode, later tests are skipped when an earlier one fails. Prefer a single longer test with `test.step` calls, since serial groups are harder to retry and debug.

What is sharding?

Sharding splits the suite across machines. Run `npx playwright test --shard=1/4` on the first machine, `--shard=2/4` on the second and so on. Playwright divides test files, or tests when fullyParallel is on, into balanced parts. On CI, use a matrix to start four jobs at once. This is the usual way to bring a long suite down to a reasonable time.

How do I get one combined report?

Use the blob reporter on each shard (`reporter: 'blob'`), upload the blob folders as artifacts, then download them in a final job and run `npx playwright merge-reports --reporter html ./all-blob-reports`. You get a single HTML report covering every shard, which is easier to read than four separate ones.

How do I choose the worker count?

Start with the default, watch CPU and memory on the machine, and watch the application under test. Too many workers cause timeouts and false failures when the test environment is small. On a laptop, leave some cores free. In CI, match the number to the runner size and try a few values.

How do I find hidden dependencies?

Run the suite with `--workers=1` and then with many workers. Run a single test alone and in random order with repeat and shuffle options if available. Failures that appear only in parallel usually point to shared data, shared accounts or tests that assume global state.

What should I avoid?

Avoid one common test user for everyone, global counters, tests that read data created elsewhere, and sleeping to wait for another test. Avoid using serial mode as a shortcut for poor isolation.

How do I measure whether parallelism helped?

Record the total run time before and after changes, plus the slowest tests, using the JSON or HTML reporter. Compare runs with one, two and four workers and note where the gain stops. If speed does not improve, the bottleneck may be the application, the database or a few very long tests. Splitting a long test into smaller independent ones usually helps parallel runs more than adding more machines.

How do I balance shards fairly?

Shards divide tests by count or file, not by duration, so one shard can end up slower. Enable fullyParallel to allow finer splitting, keep test lengths similar and review the slowest tests regularly. Some teams store timing data and arrange files to even out the load. Aim for shards that finish within a few minutes of each other, because the job that takes longest decides the total pipeline time.

What is a simple rollout plan for parallel runs?

Start with workers set to two, fix any failing tests that depend on shared data, raise the number slowly and add sharding only when one machine is no longer enough. Record each change and its effect in a short note so the team understands why the numbers were chosen and can revisit them as the suite grows.

Frequently asked questions

How does Playwright run tests in parallel?

It runs test files in separate worker processes by default, and with fullyParallel it also runs tests within a file in parallel.

What is sharding in Playwright?

Sharding splits the suite into parts using the shard option so that each part can run on a different machine at the same time.

Why do tests fail only in parallel?

Usually because tests share data, accounts or global state, so make data unique per test and remove dependence on order.

When should I use serial mode?

Only for a short flow whose steps really depend on each other, and prefer one test with steps where possible.

How do I merge reports from shards?

Use the blob reporter on each shard, collect the blob folders, then run merge-reports to produce one HTML report.

Sources and further reading

- Playwright documentation
- Playwright parallelism
- Playwright sharding

About this PDF

These PDFs are study material for learning and revision. They are not a certificate, a degree or a promise of a job, and technology changes, so check official documentation for the latest details. Spot an error? Write to info@uncodemy.com and the trainer team will review it.

More free resources and courses: uncodemy.com | info@uncodemy.com | +91 98183 66550