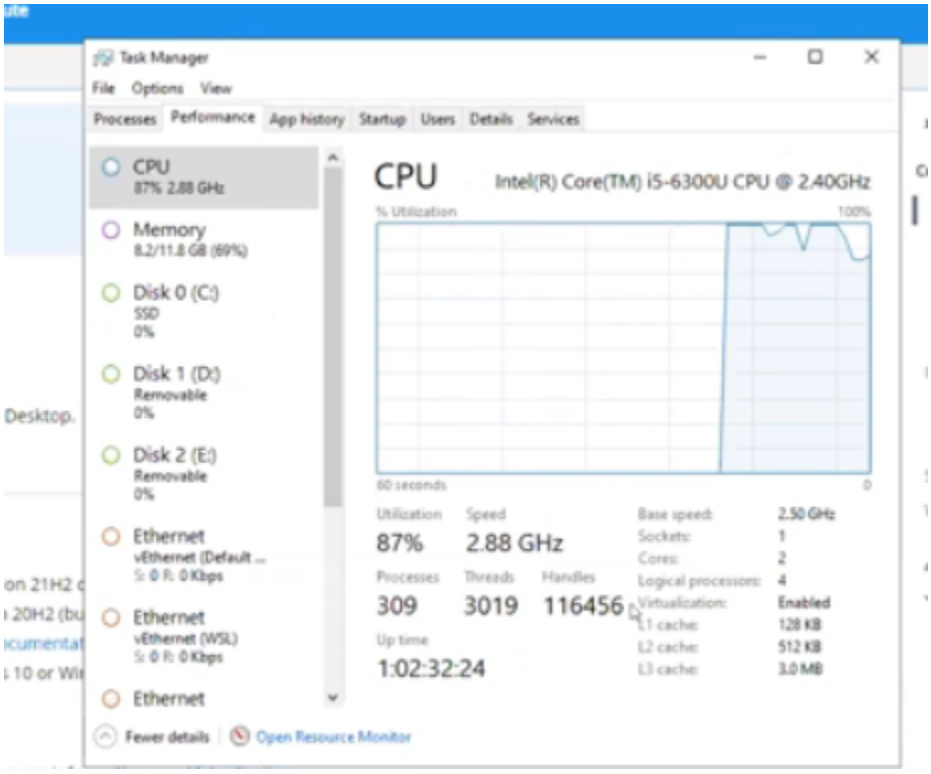
Download docker desktop.

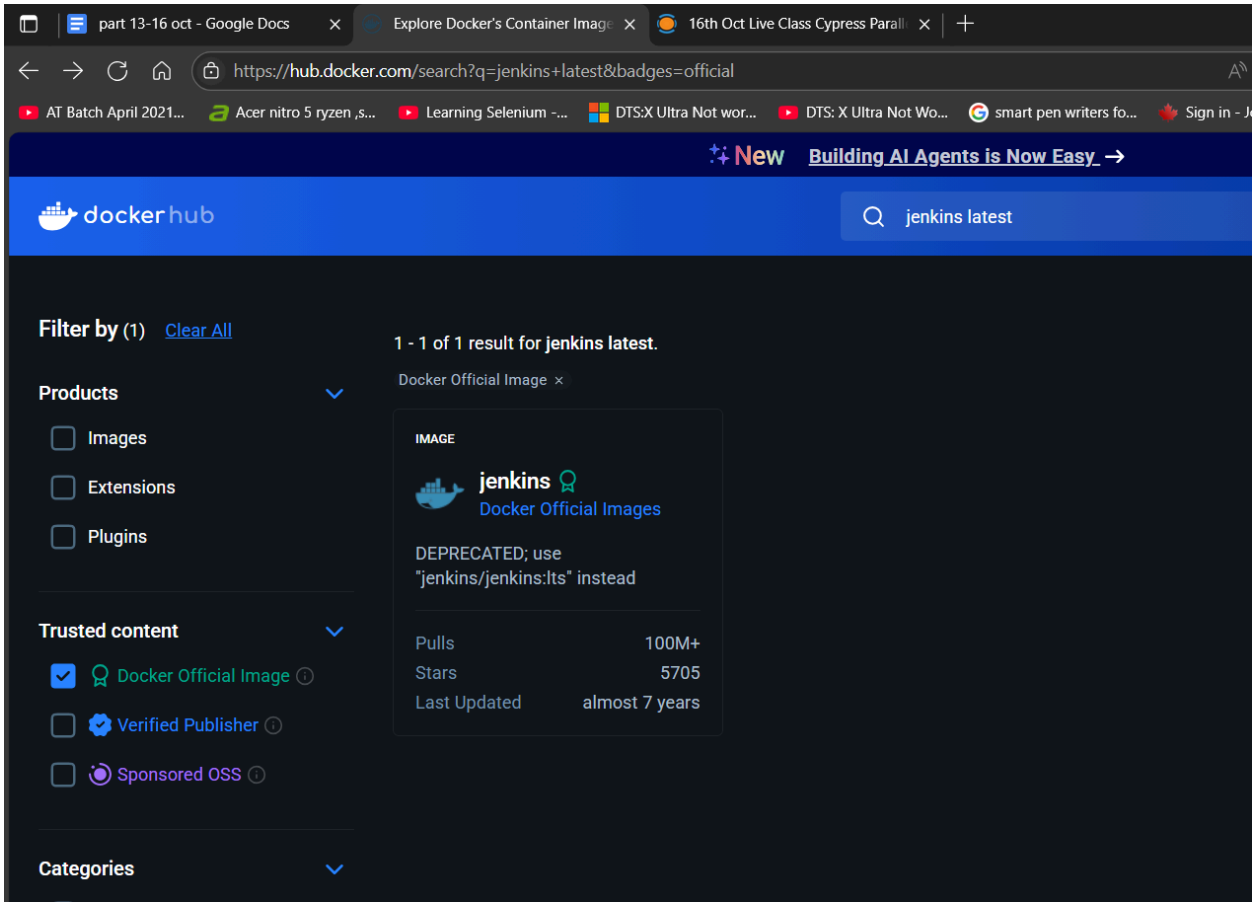
pre -requisite -
Check virtualisation is enabled.



It means on your machine you can create multiple virtual machines.

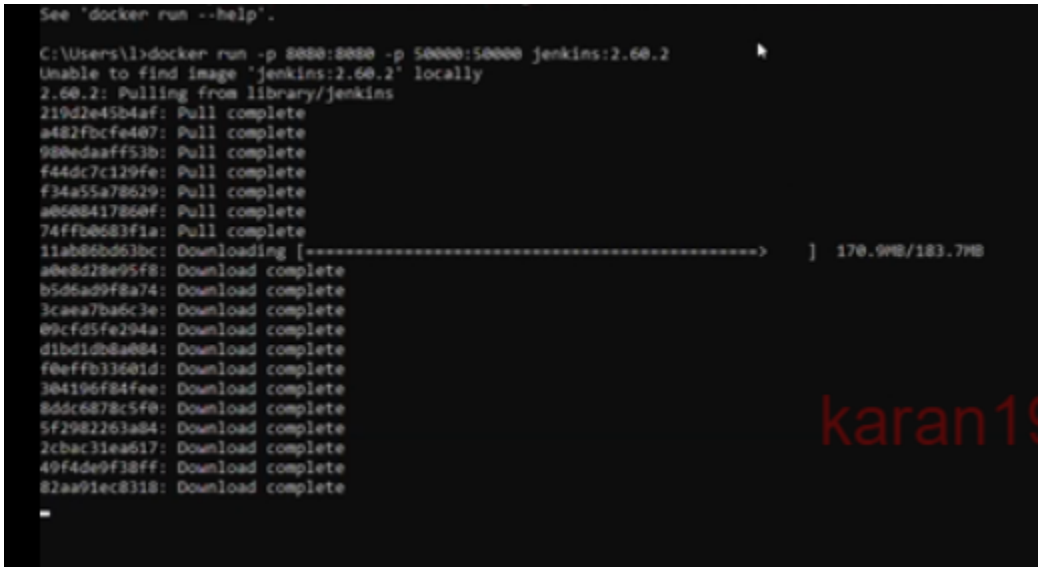
To see all commands in docker-
Open cmd.
Type docker
hit enter and you can know list of commands.

How to use docker and pull images and run on local-
Go to docker hub.
Search for the software.
Click docker original image on left side.

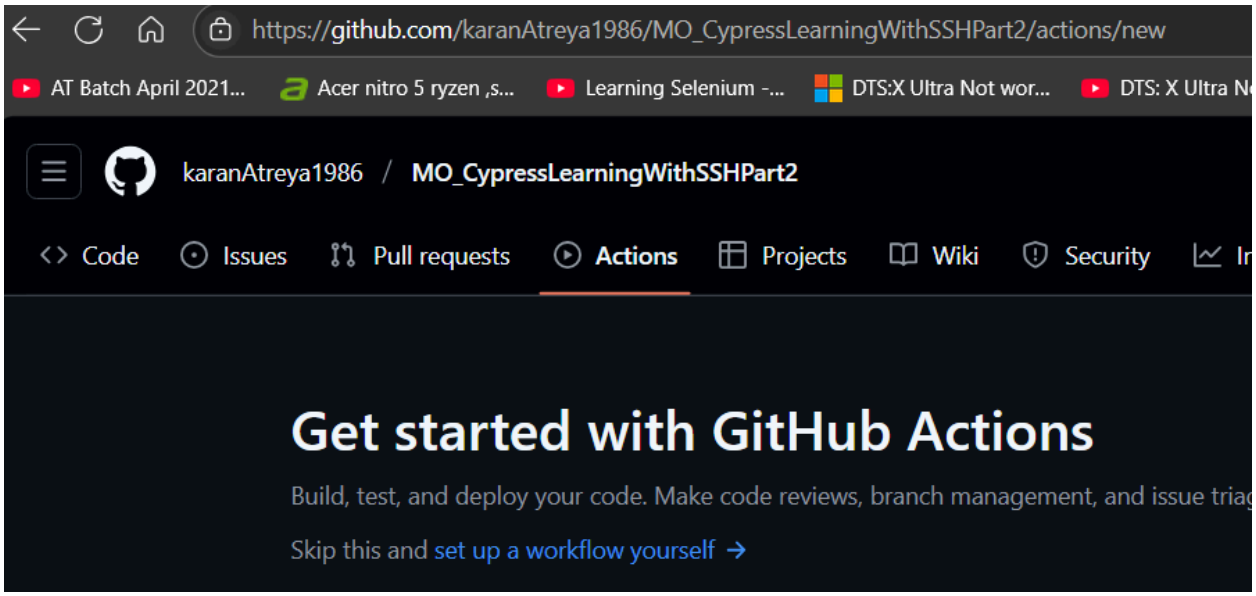


When you scroll down you can see run command.
Docker run pulls image and runs or if image exist then just runs.

```
docker run -p 8080:8080 -p 50000:50000 jenkins:2.60.2
```



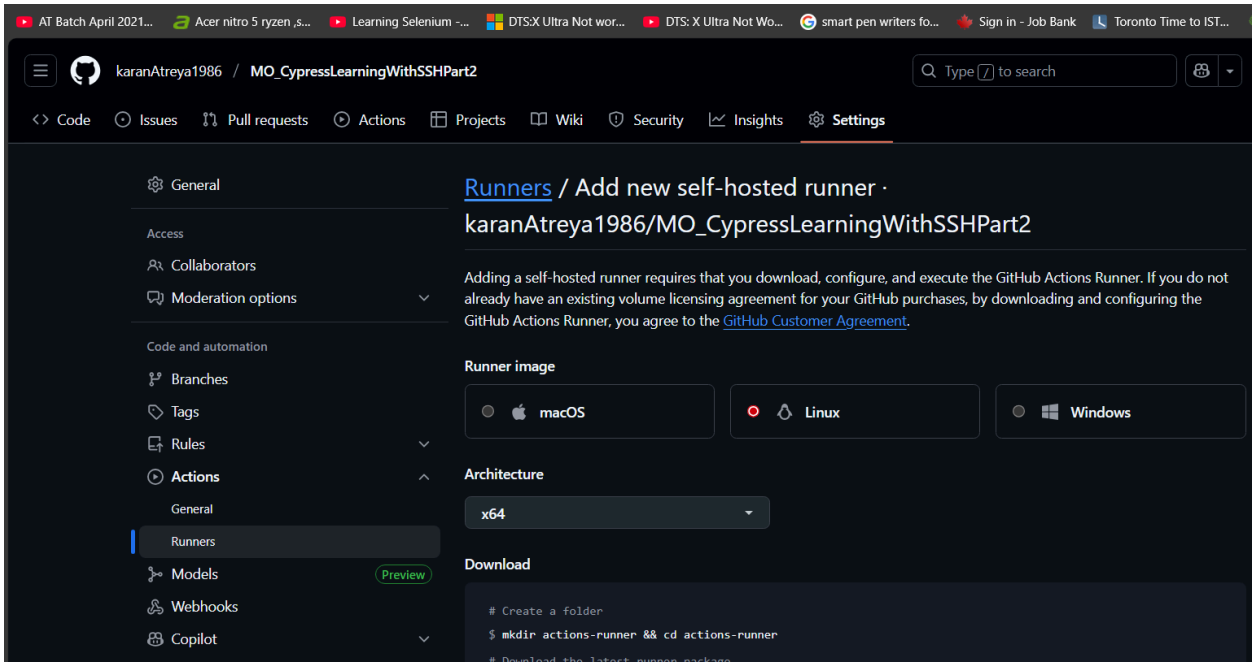
Git hub actions-
Go to the required git hub repo.
Click actions.
Click set up workflow



If you want to run on multiple events, use
On: [push, pull, merge]

How to make additional containers and run in parallel-
Containers: [1,2,3]

To get the os name - search for github self hosted runner and then in that page search for how to add runner. There are some changes.
To create own runner this is the area:
Project – settings – actions – runners.



Older mocha reports dont work properly or do not generate in github actions–
Json report is generated but not html.

❌ Why the HTML report was blank

This typically happens due to one of the following:

1. Generated with `marge` CLI instead of `cypress-mochawesome-reporter`

If you're using `mochawesome` as the reporter, you must **not** use `marge` directly anymore. Use the full plugin:
📦 `cypress-mochawesome-reporter`

Using `marge` manually can cause HTML rendering issues due to missing assets or format mismatches.

Recommended Fix — Use `cypress-mochawesome-reporter`

Update your workflow like this:

🔧 **Install correct dependencies:**
npm install --save-dev cypress-mochawesome-reporter

🔧 In `cypress.config.js`:

```
const { defineConfig } = require('cypress');

module.exports = defineConfig({
  reporter: 'cypress-mochawesome-reporter',
  reporterOptions: {
    reportDir: 'cypress/results',
    overwrite: false,
    html: true,
    json: true,
  },
  e2e: {
    setupNodeEvents(on, config) {
      require('cypress-mochawesome-reporter/plugin')(on);
      return config;
    }
  }
});
```

🔧 In your `e2e` support file (`cypress/support/e2e.js`):

```
import 'cypress-mochawesome-reporter/register';
```

🔁 Then update your GitHub Actions step:

```
- name: Run Cypress tests
  uses: cypress-io/github-action@v6
  with:
    command: npx cypress run
  continue-on-error: true
```

You can then remove the `mochawesome-merge` and `marge` steps entirely if you're using the plugin — it generates clean HTML automatically per test run.

📦 Optional — Upload Reports

```
- name: Upload HTML report
  if: always()
  uses: actions/upload-artifact@v4
  with:
    name: cypress-mochawesome-report
    path: cypress/results
```

✅ Summary

Issue	Fix

Blank HTML report	Likely due to using `marge` instead of correct plugin
Correct JSON	Yes – you have full test results
Best Practice	Use `cypress-mochawesome-reporter` and register it properly

Absolutely! Here's a clear explanation of **why you can remove `mochawesome-merge` and `marge`** when using the `cypress-mochawesome-reporter` plugin — and how the plugin simplifies everything.

🧠 Understanding the Difference

❌ The Old Manual Way (you're doing now)

If you use the plain `mochawesome` reporter:

1. Cypress creates **multiple JSON files** in `cypress/results/` — one per spec file.
2. You have to:
    - 🧩 **merge** them manually using `mochawesome-merge`
    - 🧾 **generate HTML** using `marge` from the merged JSON
3. These tools are **separate**, and not built into Cypress.
   It's a manual, multi-step process.

✅ The Plugin Way: `cypress-mochawesome-reporter`

When you use the **`cypress-mochawesome-reporter` plugin**, it:

- Hooks directly into Cypress test lifecycle
- Automatically collects all results across spec files
- Generates:
    - A single merged **JSON**

    - And a single, beautiful **HTML report**

- ✅ **No need to merge manually**
- ✅ **No need to run `marge`**
- ✅ Everything is done *automatically at the end of the test run*

📁 What You'll See in `cypress/results/`

When using the plugin, after `npx cypress run`, you'll find:

```
cypress/results/
├── mochawesome-report/
│   ├── mochawesome.html   ← ✅ Full test report
│   ├── mochawesome.json
│   └── assets/...         ← CSS/JS for the report
```

You can open `mochawesome.html` and get a full dashboard: test list, errors, screenshots, etc.

✅ What You Can Remove from Your Workflow

If you switch to `cypress-mochawesome-reporter`, **you can delete these**:

```
- name: Merge Mochawesome JSON reports
  run: npx mochawesome-merge "cypress/results/*.json" > mochawesome.json

- name: Generate HTML report from merged JSON
  run: npx marge mochawesome.json --reportDir=cypress/results
--reportFilename=mochawesome-report.html
```

And instead, **just upload the whole report folder**:

```
- name: Upload test report
  if: always()
  uses: actions/upload-artifact@v4
  with:
    name: mochawesome-report
    path: cypress/results/mochawesome-report/
```

🚀 Result

This gives you:

- Cleaner workflow
- No need for manual merging or extra dependencies
- Fully styled interactive HTML report

✅ Summary

Step	Old Way (`mochawesome`)	Plugin (`cypress-mochawesome-reporter`)
Merge JSON manually	✅ Required	❌ Not needed
Generate HTML manually	✅ Required (`marge`)	❌ Not needed
Auto HTML + JSON report	❌ No	✅ Yes
Simpler GitHub Actions flow	❌ No	✅ Yes

Exit codes -

❌ 1. `exit code 14` — Cypress couldn't run or crashed

`npx cypress run` returning `exit code 14` usually means:

> Cypress failed to **launch**, often due to missing plugins, misconfiguration, or runtime errors in `cypress.config.js`.

🔍 This happens frequently when `cypress-mochawesome-reporter` is **installed but not registered correctly** in the config.

⚠️ 2. Report folder is missing (`cypress/results/mochawesome-report/`)

The warning:

No files were found with the provided path: cypress/results/mochawesome-report/

means the report **was never generated** — likely because Cypress never completed its run, so the reporter plugin never ran either.

✅ Fix Both Issues

Here's how to fix the root problem: **correct installation and registration of the reporter plugin**.

🛠️ Step 1: Install the plugin

In your project:

npm install --save-dev cypress-mochawesome-reporter

🛠️ Step 2: Update `cypress.config.js`

Make sure this file exists and looks like this:

```
const { defineConfig } = require('cypress');

module.exports = defineConfig({
  reporter: 'cypress-mochawesome-reporter',
  reporterOptions: {
    reportDir: 'cypress/results/mochawesome-report',
    overwrite: false,
    html: true,
    json: true
  },
  e2e: {
    setupNodeEvents(on, config) {
      require('cypress-mochawesome-reporter/plugin')(on);
```

```
      return config;
    }
  }
});
```

🛠️ Step 3: Add this to your support file

At the top of your test support file (`cypress/support/e2e.js`):

```
import 'cypress-mochawesome-reporter/register';
```

🛠️ Step 4: Update GitHub Actions YAML

Here's your fixed and simplified `yaml` using the plugin:

```
name: Cypress Tests

on: push

jobs:
  cypress-run:
    runs-on: ubuntu-24.04
    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Install dependencies
        run: npm ci

      - name: Run Cypress tests
        id: cypress
        continue-on-error: true
        uses: cypress-io/github-action@v6
        with:
          command: npx cypress run

      - name: Upload Mochawesome report
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: mochawesome-report
```

```
      path: cypress/results/mochawesome-report

  - name: Fail if Cypress tests failed
    if: steps.cypress.outcome == 'failure'
    run: exit 1
```

✅ Summary

Problem	Fix
`exit code 14`	Register `cypress-mochawesome-reporter` correctly
HTML report not generated	Because Cypress never finished / plugin not initialized
Artifact upload warning	Report folder didn't exist due to the failure above

Great question!

✅ Where to install `cypress-mochawesome-reporter`

You should run **Step 1** — the install command — in the **root directory of your Cypress project** (same place as your `package.json`).

🧭 Here's how to do it:

📁 Example structure of your project:

```
my-project/
├── cypress/
├── cypress.config.js
├── package.json
└── ...
```

💻 In your terminal:

```
cd my-project
npm install --save-dev cypress-mochawesome-reporter
```

🔍 Why it goes in your project

This ensures:

- The reporter is listed in your `package.json` under `devDependencies`
- It's available during your GitHub Actions run (`npm ci` will install it)

🧪 After installing:

You should see this in `package.json`:

```
"devDependencies": {
  "cypress-mochawesome-reporter": "^3.4.0", // or latest
  ...
}
```

This config.js file –

e2e: { setupNodeEvents(on, config) { require('cypress-mochawesome-reporter/plugin')(on); return config;is return mandatory?

Great question!

✅ Is `return config` mandatory in `setupNodeEvents`?

Yes — **in most cases, it's best practice to include `return config`**, especially if:

- You're using plugins that **modify or rely on** the `config` object (like environment variables, base URLs, etc.)
- You're chaining or extending the Cypress config with other settings

🔍 What happens if you don't return `config`?

If you omit `return config`, Cypress may not apply any modifications made to the configuration during plugin setup. Some plugins (like `cypress-mochawesome-reporter`) **may still work**, but others that change config dynamically **won't**.

🧠 Example with proper return:

```
e2e: {
  setupNodeEvents(on, config) {
    require('cypress-mochawesome-reporter/plugin')(on);
    return config; // ✅ ensures the updated config is passed back to Cypress
  }
```

}

✅ Bottom Line

- 🔁 Returning `config` is **safe and recommended**
- ❌ Not returning it might break plugin behavior in certain cases
- ✔️ For `cypress-mochawesome-reporter`, it may still work — but better to **always return it**

✅ Why it worked without `npm ci`

If you're running Cypress **locally**, or even in CI (like GitHub Actions), and you **didn't run `npm ci`**, here's why it may still work:

🧠 1. You may have already installed dependencies using `npm install`

If you've previously run:

npm install

Then your `node_modules/` directory already contains everything needed — including `cypress-mochawesome-reporter`.

➡️ So running Cypress tests afterward just works.

🧠 2. In GitHub Actions, dependencies might be cached or auto-installed

If your GitHub Actions workflow **does not run `npm ci`**, but the action still works, then:

- You might be committing your `node_modules/` (not ideal).
- You might be using a base Docker image or action that installs Cypress + reporter already.
- GitHub Actions may still run `npm install` **implicitly** if you don't override it.

🧠 3. Cypress GitHub Action can install dependencies for you

If you're using:

- uses: cypress-io/github-action@v6

That action can **automatically install dependencies** unless you explicitly tell it not to.

📦 From official docs:

> If your project does not have a `cypress` binary already installed, the action will automatically install it using `npm install`.

✅ Best Practice: Still Add `npm ci`

Even if it "works", best practice in CI is:

- run: npm ci

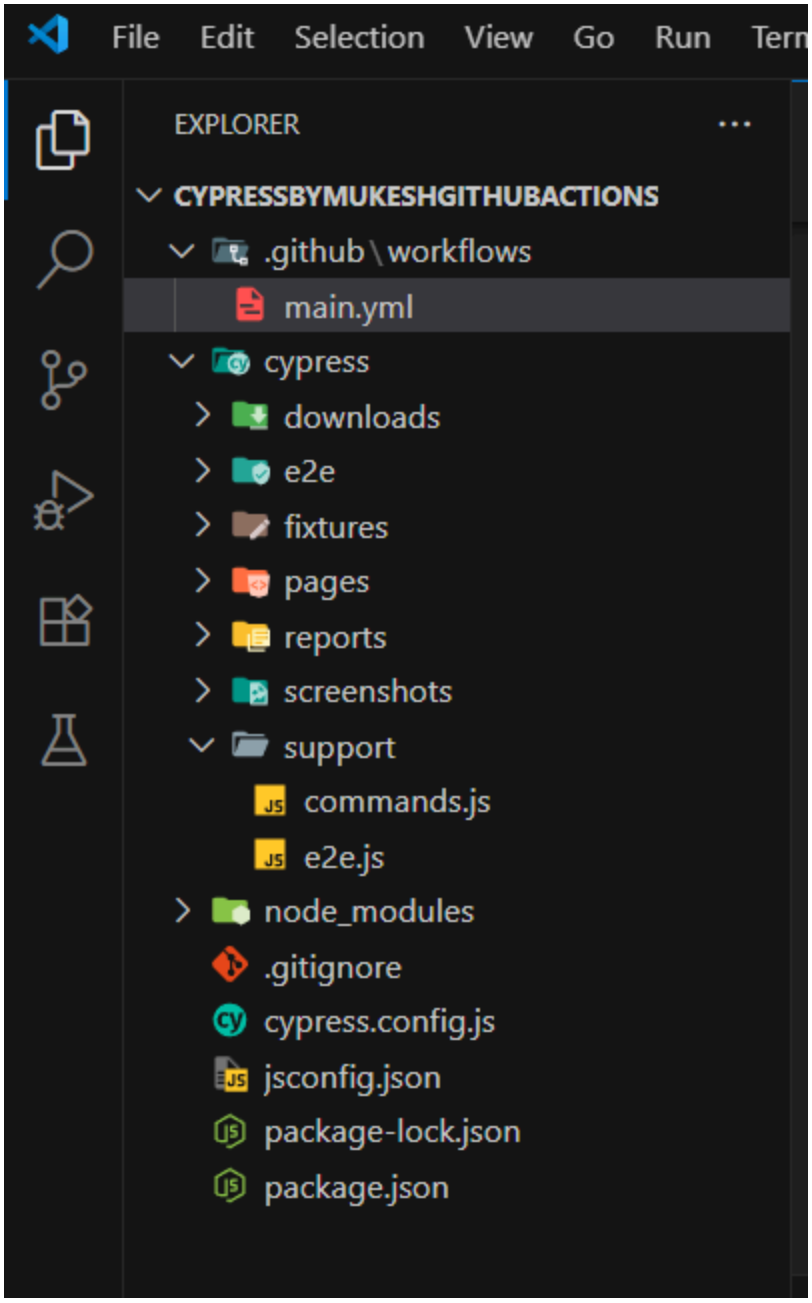
Because:

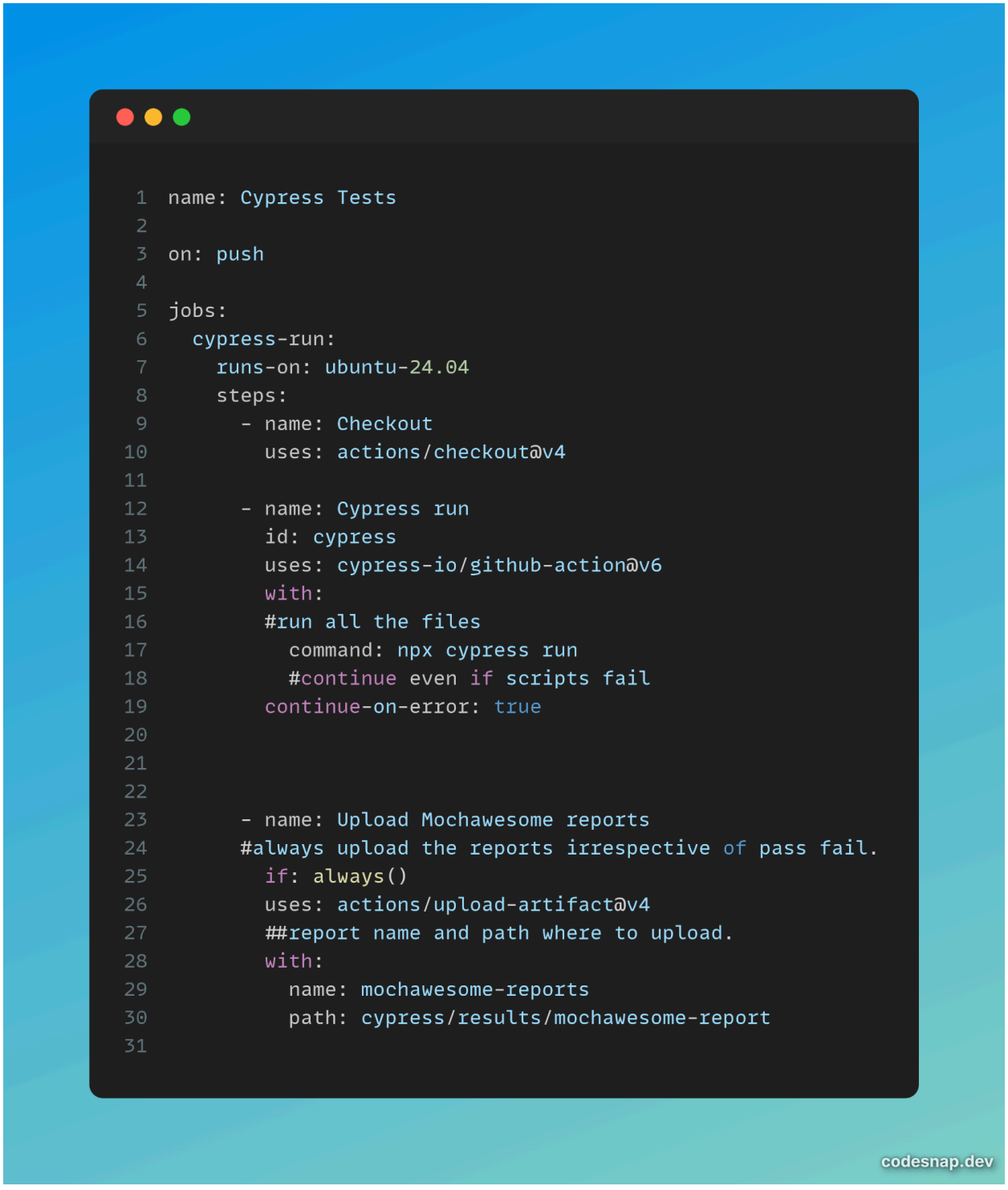
- It ensures clean, reproducible installs from `package-lock.json`
- It's faster and more predictable than `npm install`
- It avoids dependency drift

✅ Summary

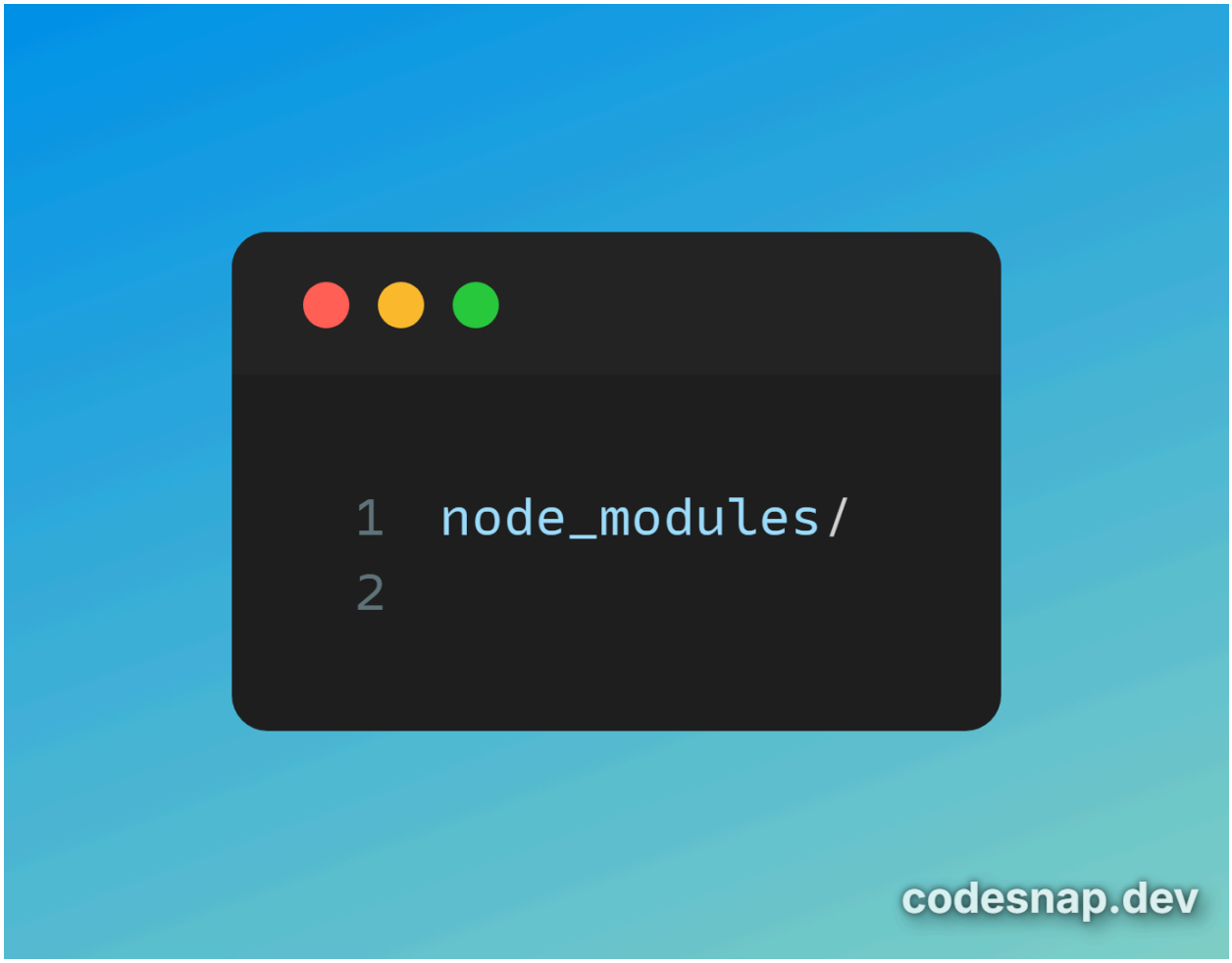
Reason it worked	Explanation
Already ran `npm install`	Dependencies were in `node_modules`
Cypress GitHub Action helped	Automatically installs packages unless told not to
Local dev environment	Had all the right packages already installed

Yaml file –

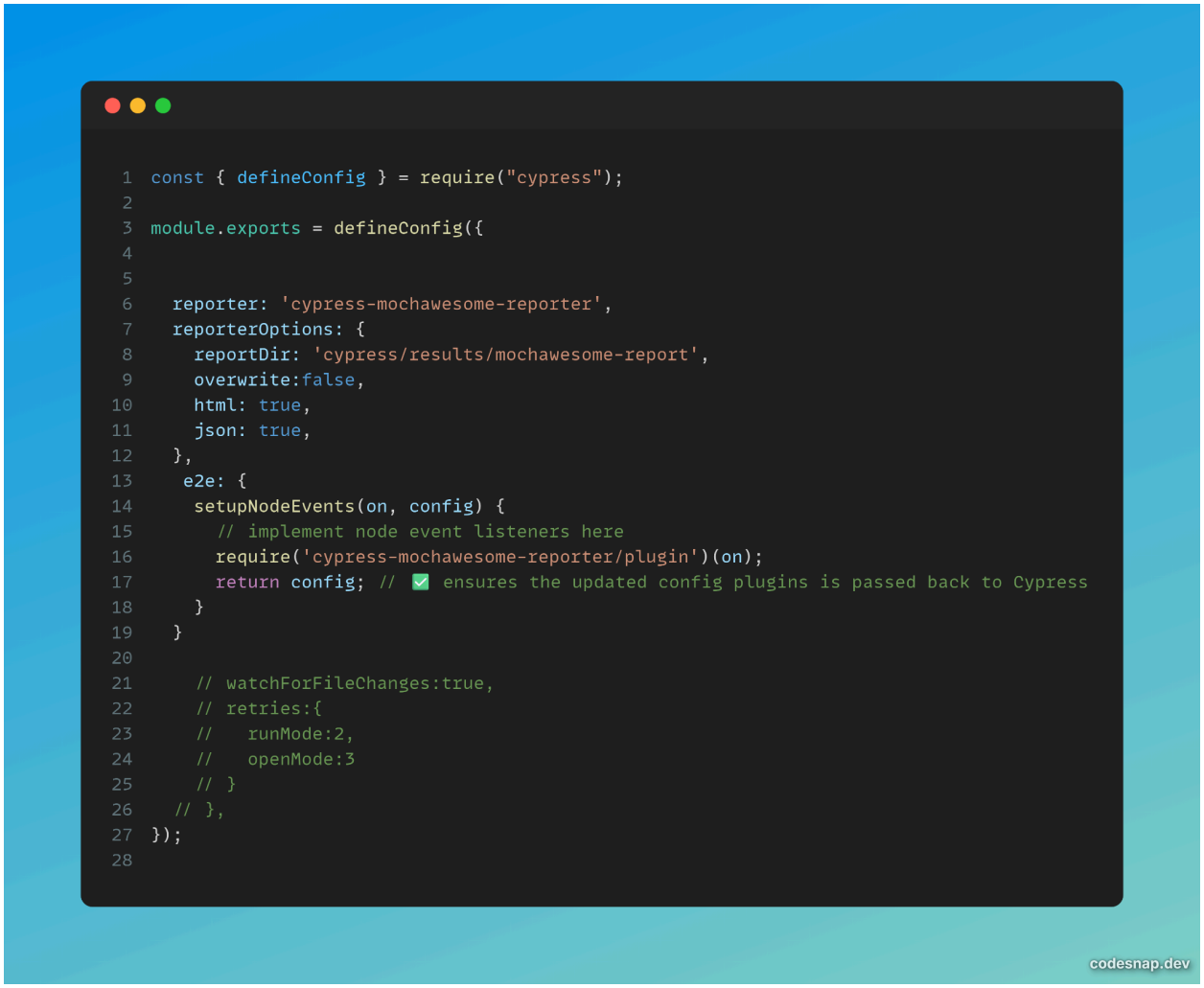

File
Edit
Selection
View
Go
Run
Terr
EXPLORER
CYPRESSBYMUKESHGITHUBACTIONS
.github \ workflows
main.yml
cypress
downloads
e2e
fixtures
pages
reports
screenshots
support
commands.js
e2e.js
node_modules
.gitignore
cypress.config.js
jsconfig.json
package-lock.json
package.json



Git ignore file–

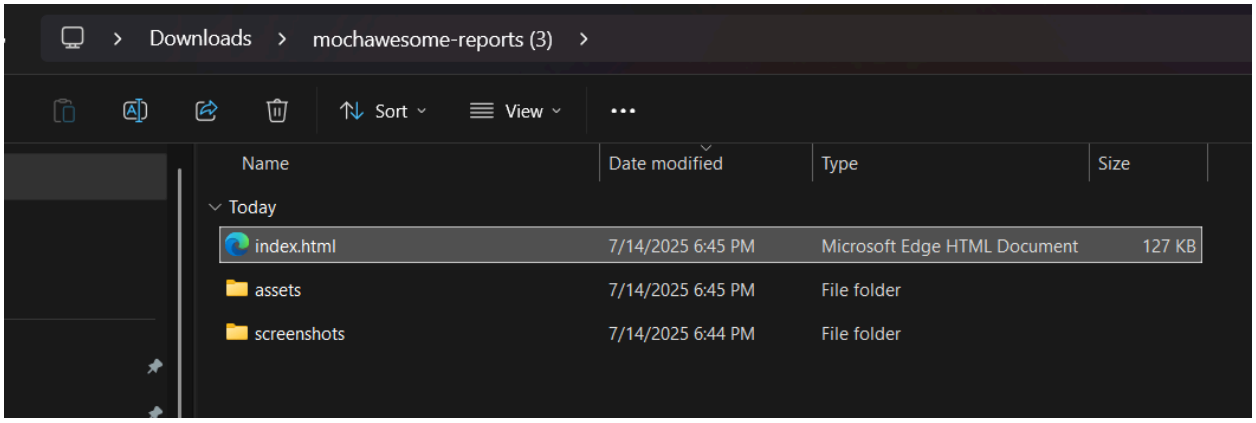

1 node_modules/
2
codesnap.dev

Cypress config.js -



Package.json –

Download the artifact, then unzip it then check index.html.
Opening index.html directly from the zipped file wont open the report.
Unzip and open-



🧱 Code Block:

```
- name: Fail if Cypress tests failed
  if: steps.cypress.outcome == 'failure'
  run: exit 1
```

🎯 What does it do?

This step **explicitly fails the GitHub Actions job** if the Cypress tests failed — even if you previously allowed them to "continue on error."

It's a way to:

- ✅ Let **report steps run no matter what**
- ✅ Still make the workflow **fail overall** if Cypress failed

🔧 Why is it needed?

Earlier in your workflow, you likely used:

```
- name: Run Cypress
  id: cypress
  continue-on-error: true
  ...
```

This is good because:

- It allows the pipeline to **continue to reporting**, even if tests fail
- But GitHub Actions will **not fail the job** unless you manually do it later

So we fix that by checking:

if: steps.cypress.outcome == 'failure'

…and force the workflow to fail with:
exit 1

🧠 Think of it like this:

- `continue-on-error: true` lets other steps run (like generating and uploading test reports)
- `exit 1` ensures your CI still **fails if tests failed**, so you don't accidentally merge broken code

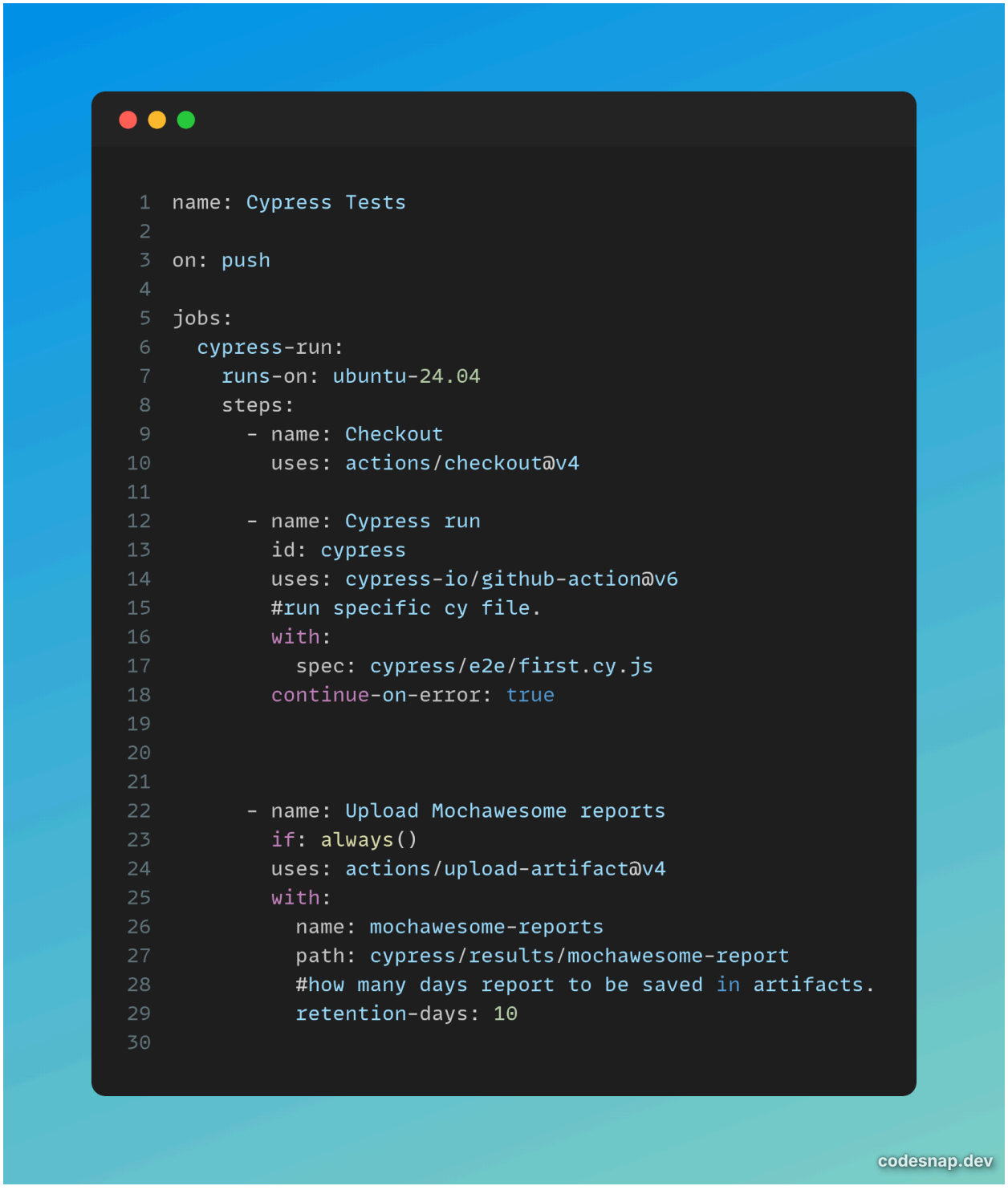
✅ When should you use it?

Use this **any time you want to:

1. Run Cypress,
2. Always generate reports/screenshots/videos,
3. Still fail the job if tests didn't pass.

Run single file-

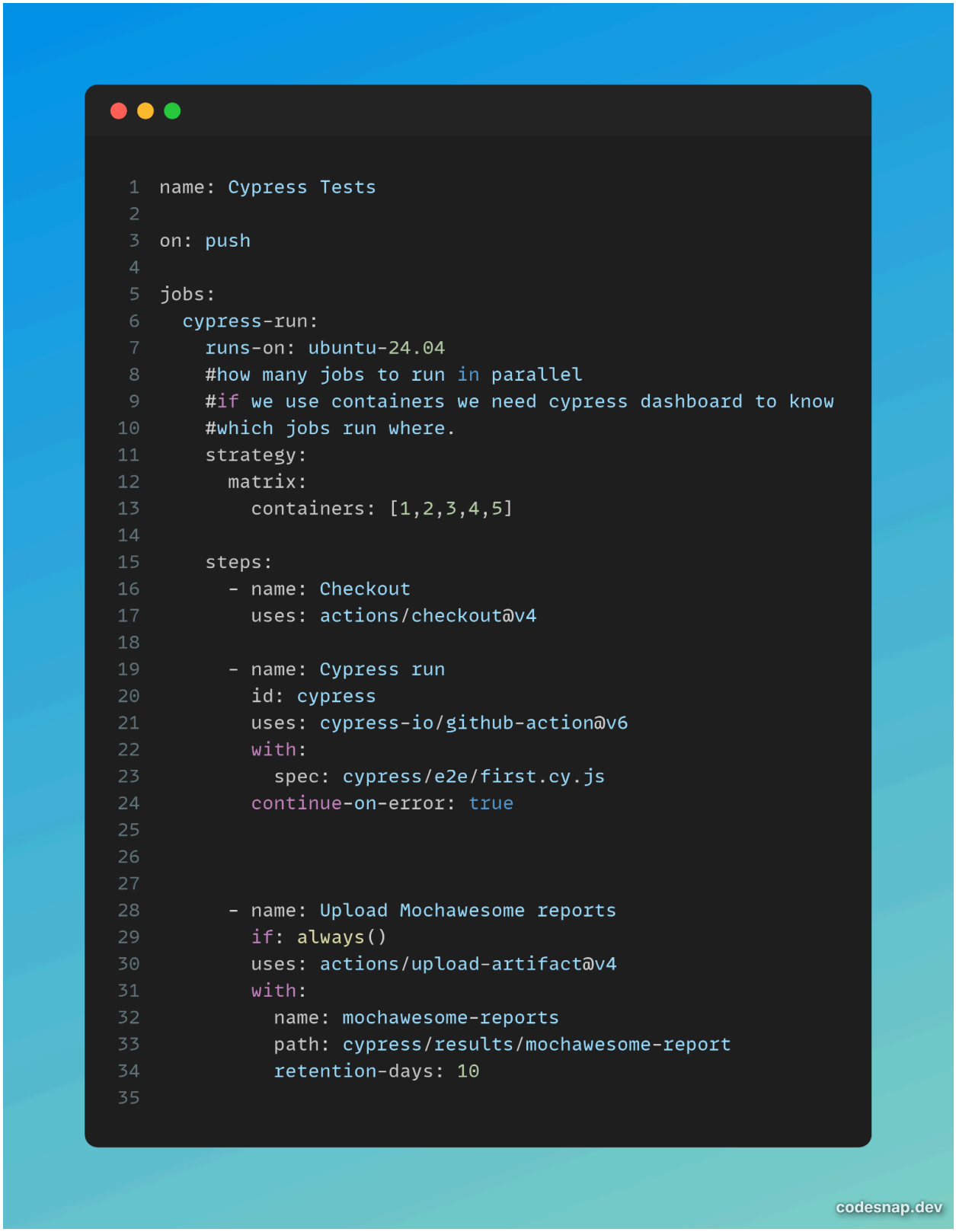
Clearer picture –



Another way to run specific spec file - npm run <scriptname>

How to add more containers for parallel runs-

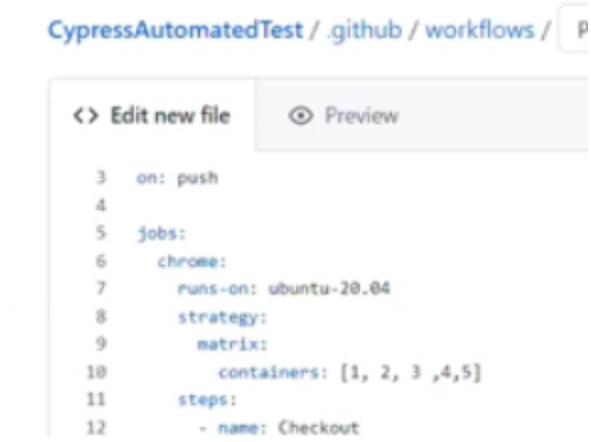
When we dont use the cypress dashboard and we dont separate at job level we get this error - the test runs on five containers but reports wont be generated.

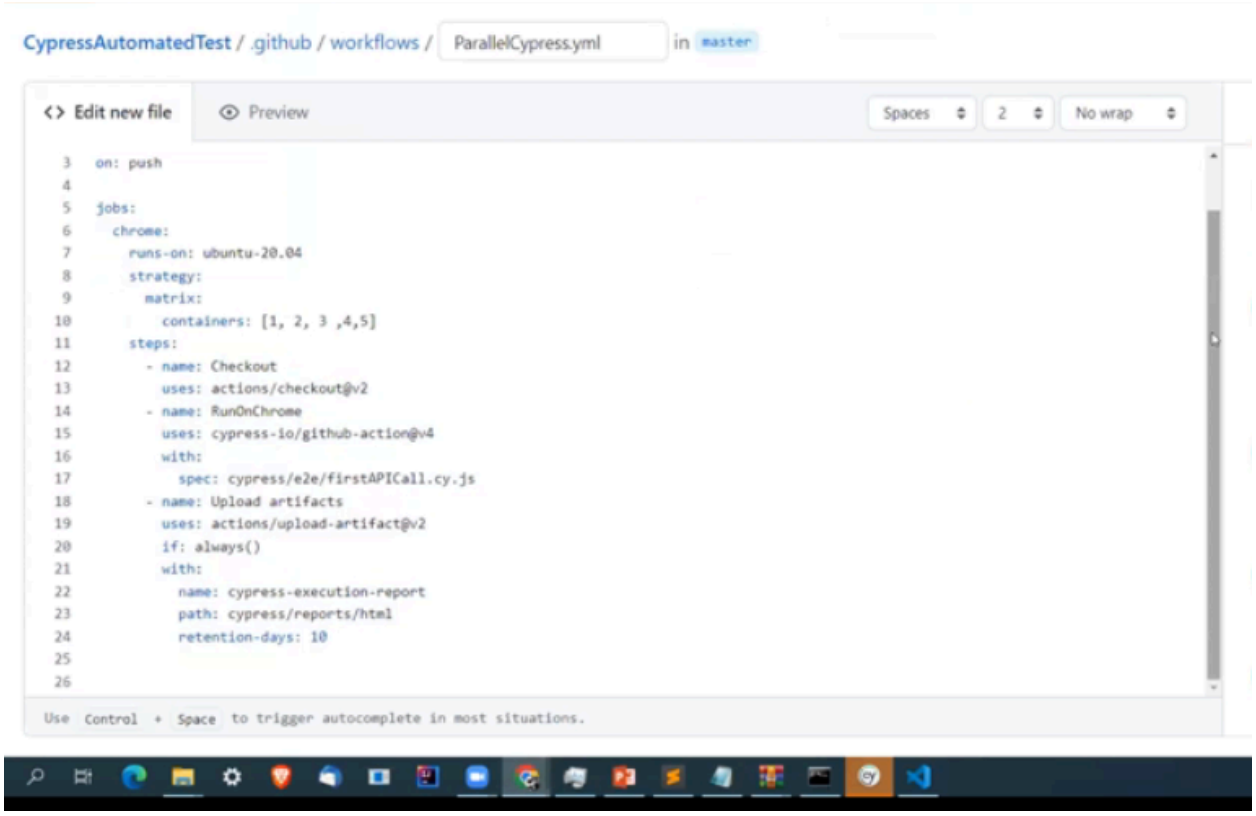


We get this error message in the upload mocha reports-

```
Error: Failed to CreateArtifact: Received non-retryable error: Failed request: (409)
Conflict: an artifact with this name already exists on the workflow run
```

Create new file parallelcypress.yaml.
Created above.





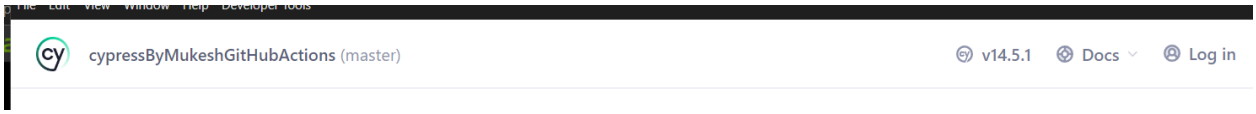
In cypress for parallel runs, we need to use github actions, no internal feature given by cypress.

Cypress dashboard-
Where to run which test on which container.
To run in parallel.
Open cypress.

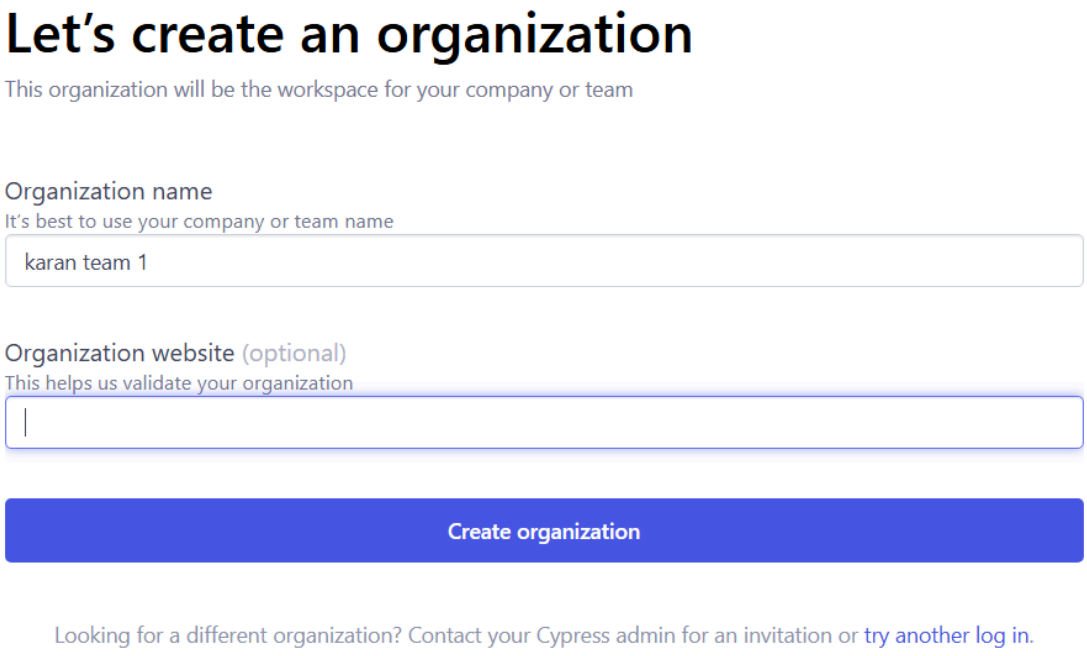
Npx cypress open.

```
C:\Users\karan\Desktop\cypressByMukeshGitHubActions>npx cypress open

DevTools listening on ws://127.0.0.1:59195/devtools/browser/b4aab81e-813a-401c-a9c0-1cec4561c538
```

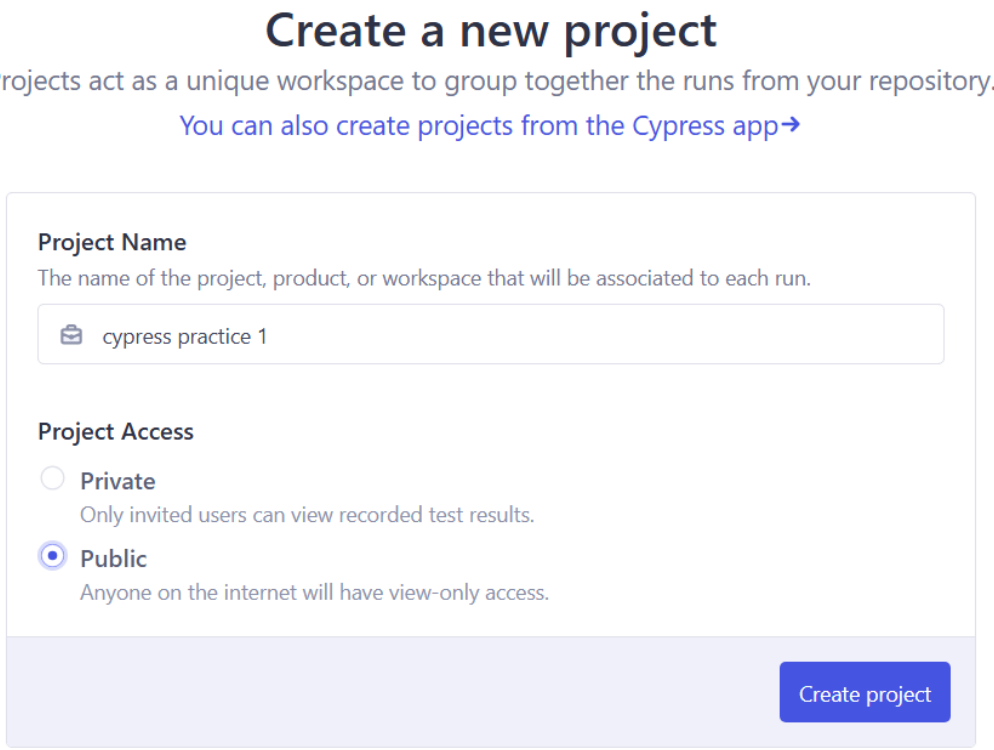
Logout and login to cypress dashboard.



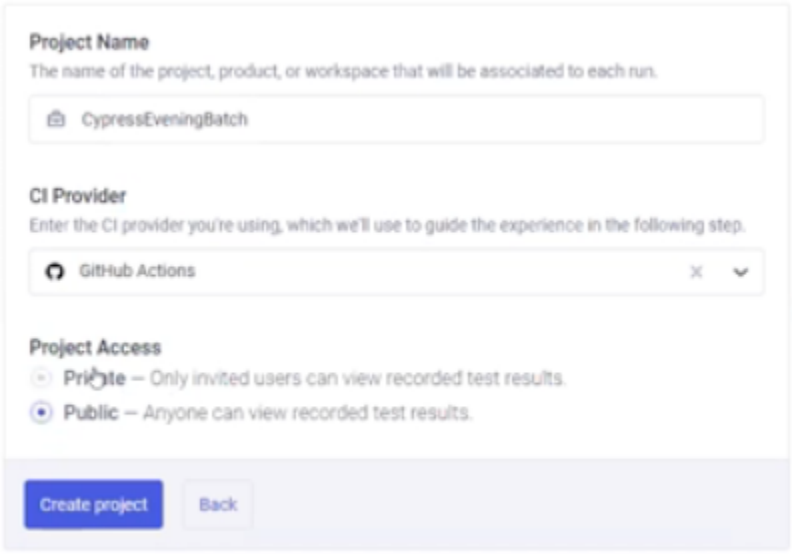
Create organization.



One organisation can have one or more projects.
Create new project.

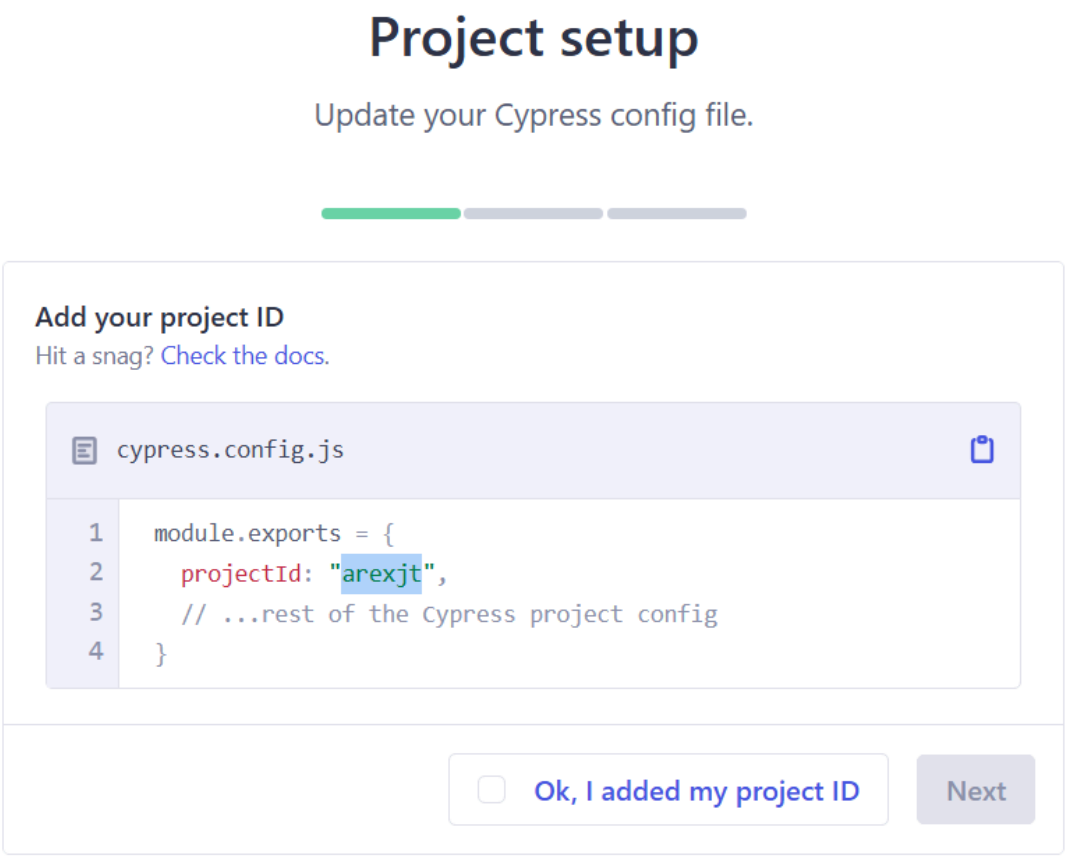


Below is not a clear picture.



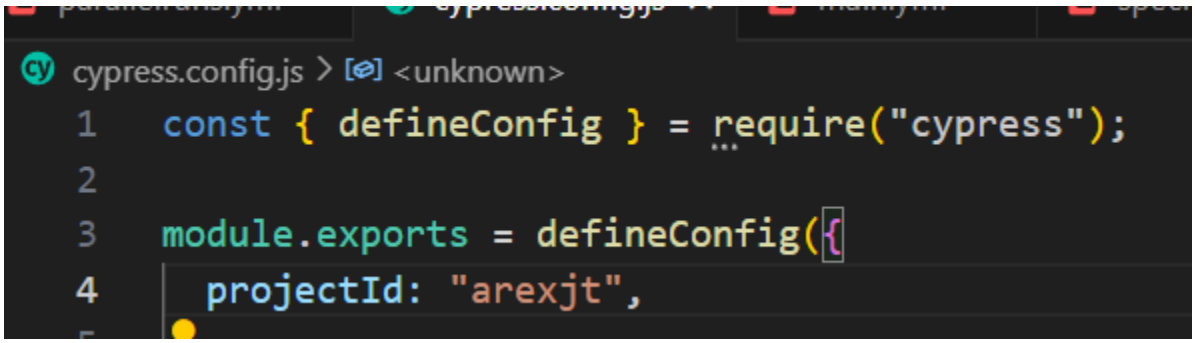
Create project.

You will get this project id -

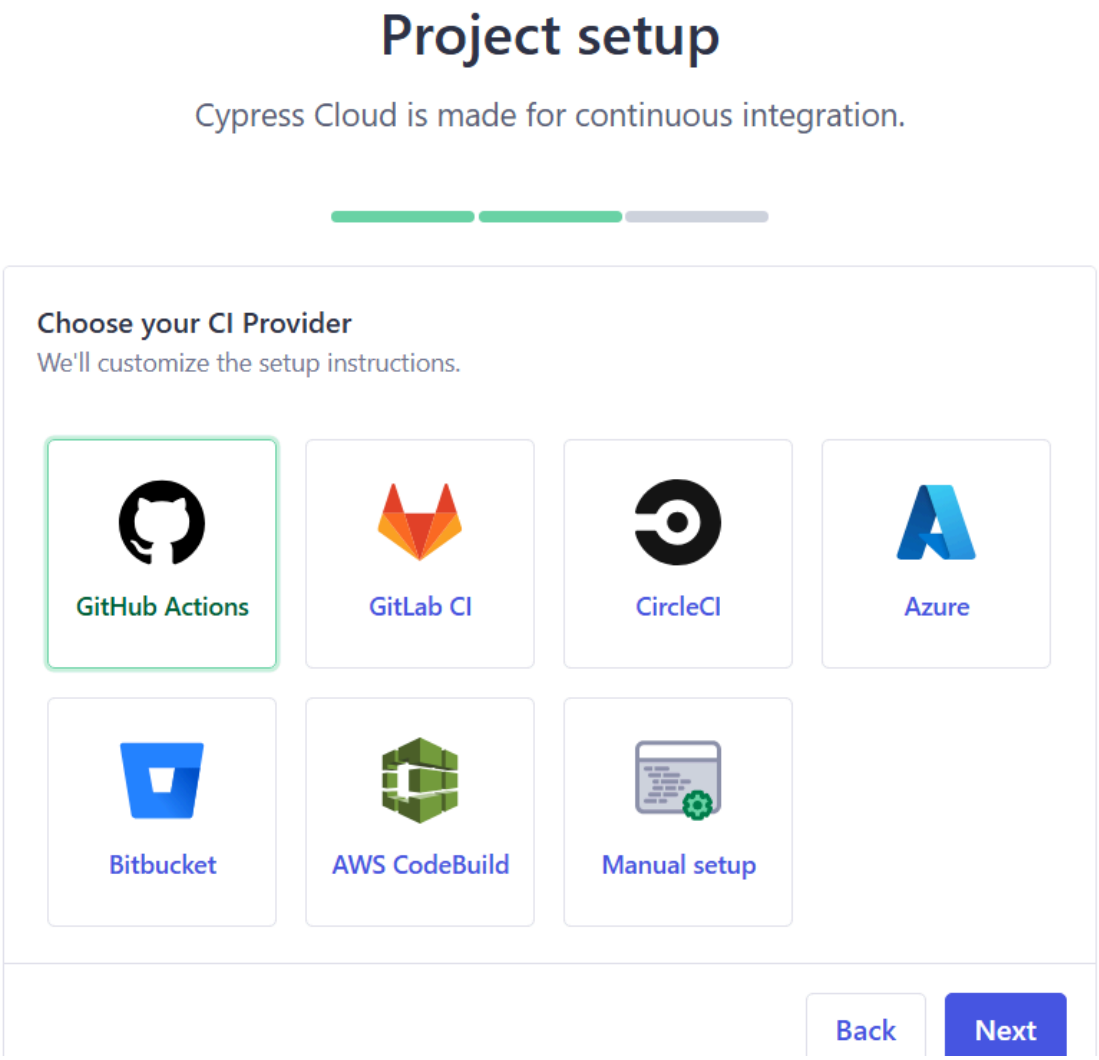


Go to cypress config.js file-
Add here.

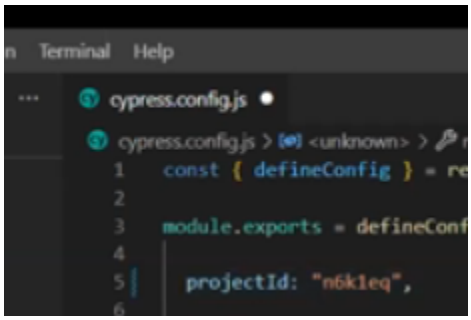
Add this project id.



Select ci provider-



Not clear picture below-



Add cypress record key changes to github actions–
There are some changes and multiple steps.

Project setup

Trigger a run with GitHub Actions.

Try it first

Record a run in your terminal.

```
$ npx cypress run --record --key 8f946559-7c18-4216-ba0b-87f633a78703
```

GitHub Actions Setup

Run Cypress tests every time you push.

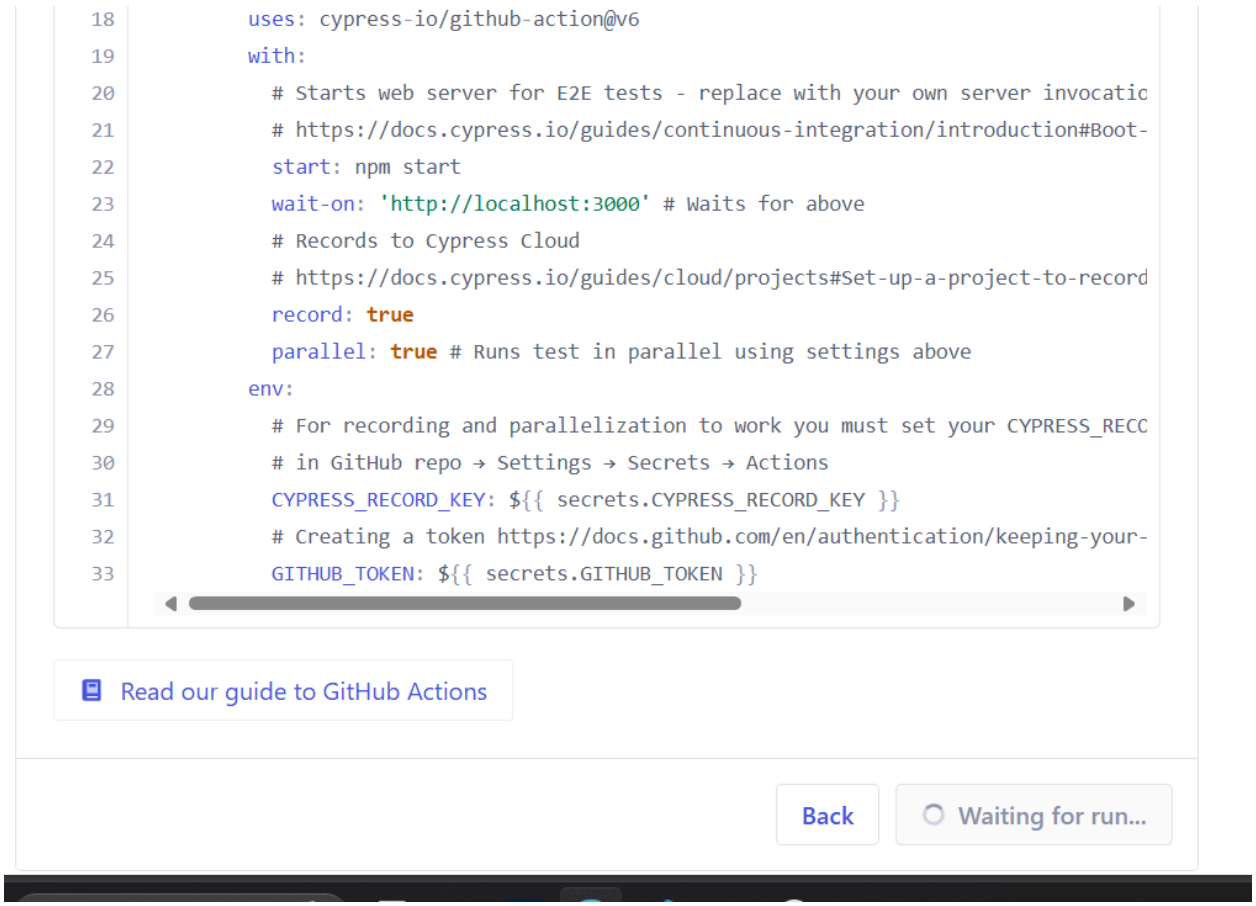
1 Set `CYPRESS_RECORD_KEY` in your GitHub repo → Settings → Secrets → Actions:

CYPRESS_RECORD_KEY	8f946559-7c18-4216-ba0b-87f633a78703

2 Then add this file to your repository:

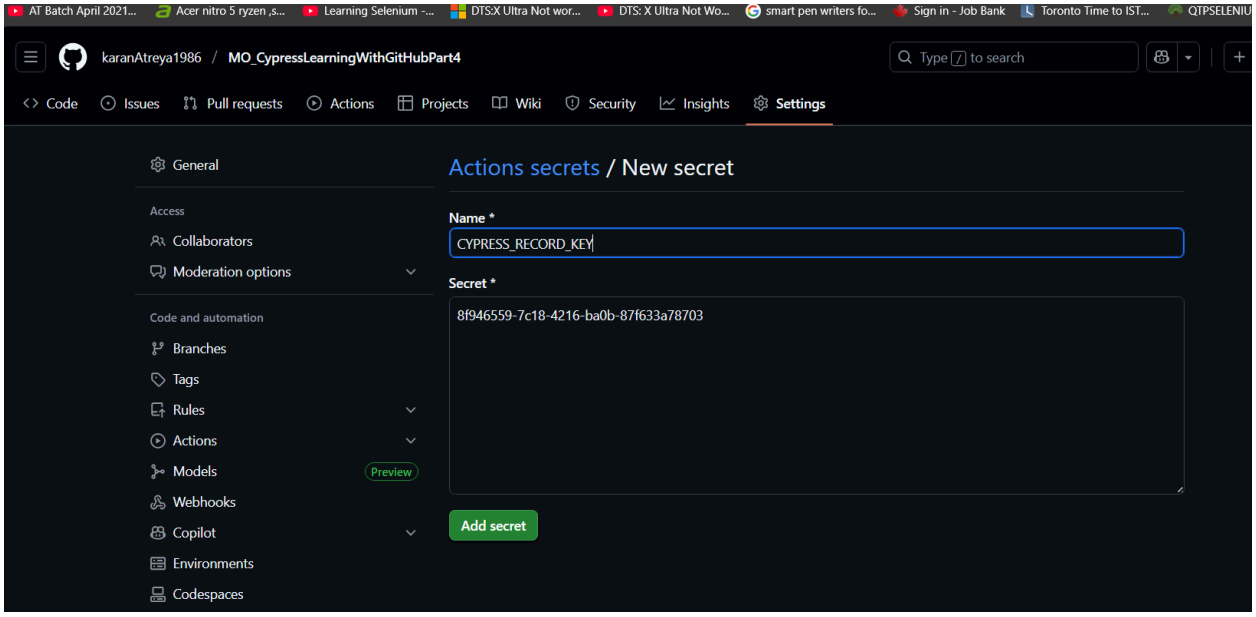
.github/workflows/cypress.yml

```yaml
name: Cypress Tests
on: [push]
jobs:
  cypress-run:
    runs-on: ubuntu-latest
    # Runs tests in parallel with matrix strategy https://docs.cypress.io/guides/gui
    # https://docs.github.com/en/actions/using-jobs/using-a-matrix-for-your-jobs
    # Also see warning here https://github.com/cypress-io/github-action#parallel
    strategy:
      fail-fast: false # https://github.com/cypress-io/github-action/issues/48
      matrix:
        containers: [1, 2] # Uses 2 parallel instances
    steps:
      - name: Checkout
        uses: actions/checkout@v4
      - name: Cypress run
        # Uses the official Cypress GitHub action https://github.com/cypress-io/gith
        uses: cypress-io/github-action@v6
```

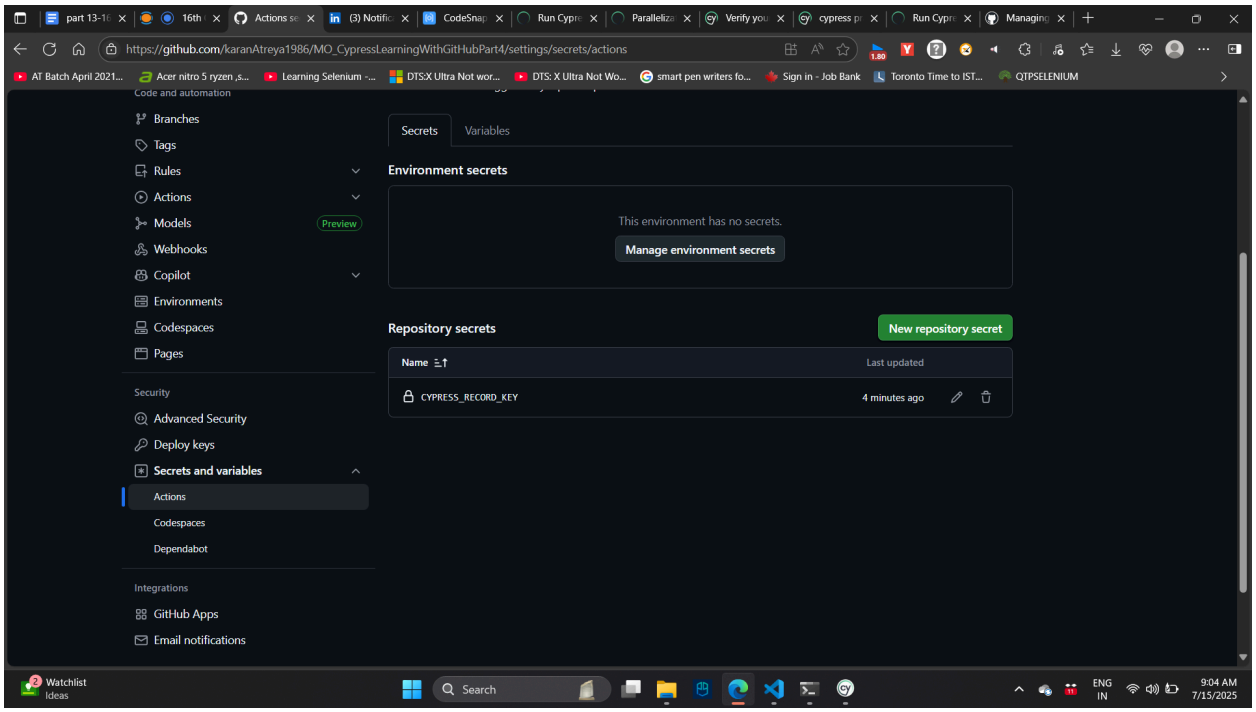


NOTE-
Containers: [windows, mac, linux] can also come.
Containers: [chrome, firefox] can also come.

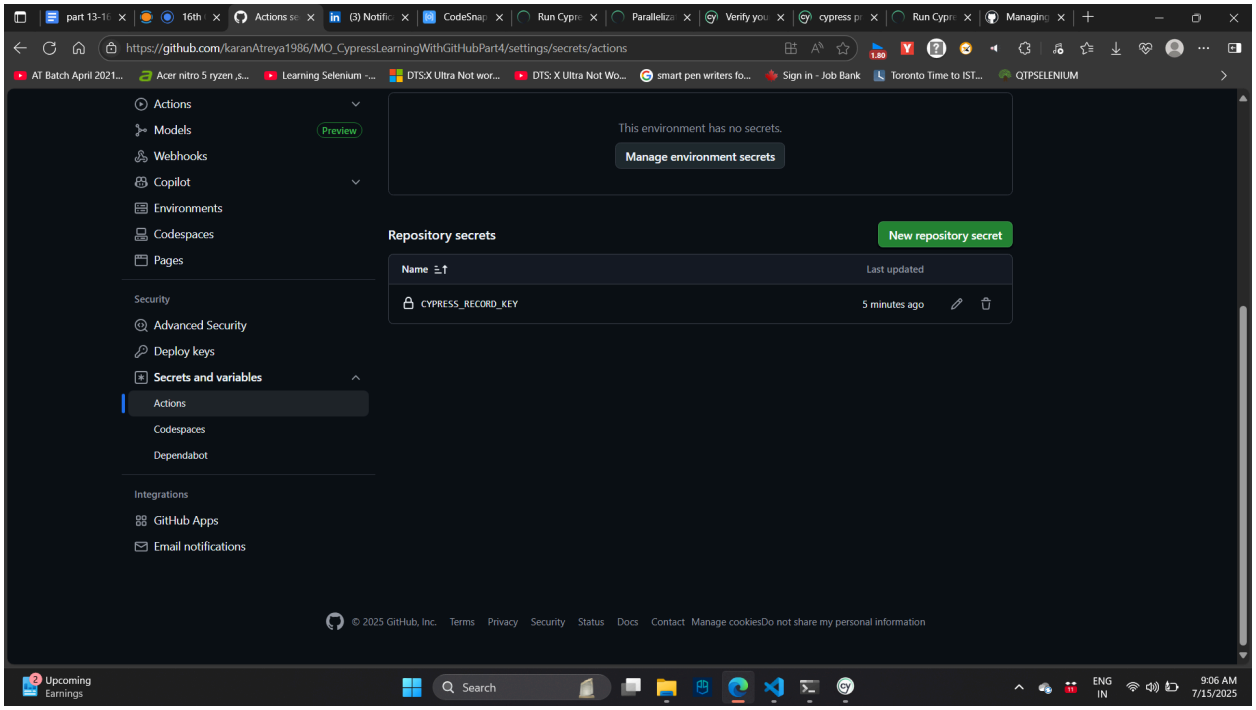
Go to the corresponding repo.
Click settings.
Click actions /secrets in left side.
Enter new secret.
Store cypress secret here. Click add secret.



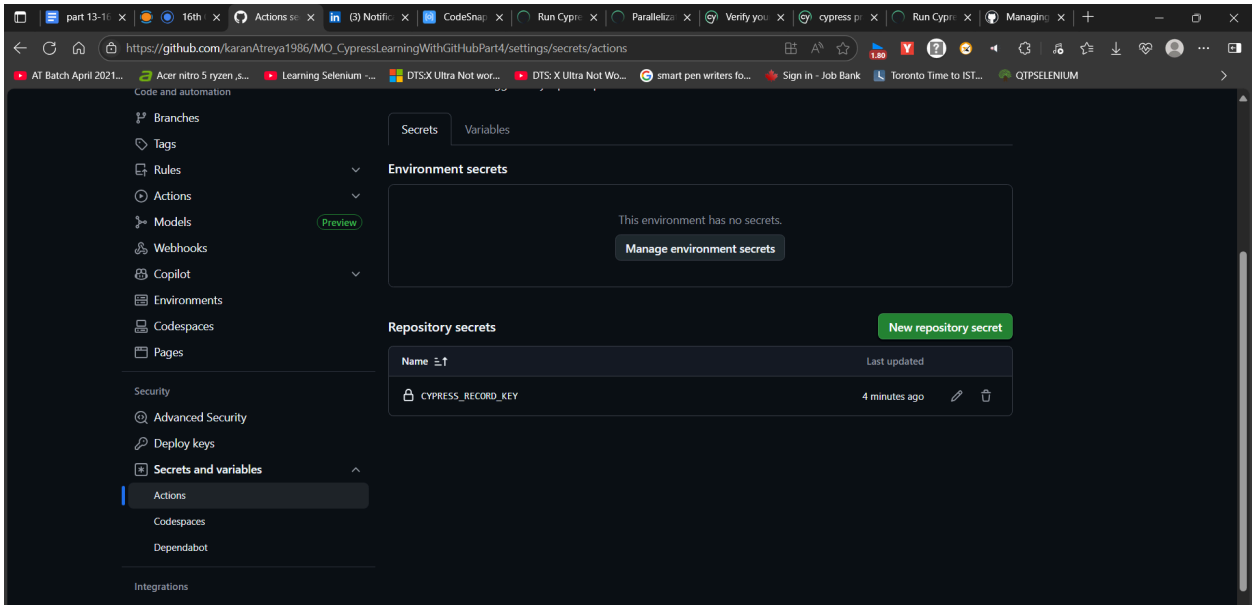
ocd=



Settings – secrets – actions –

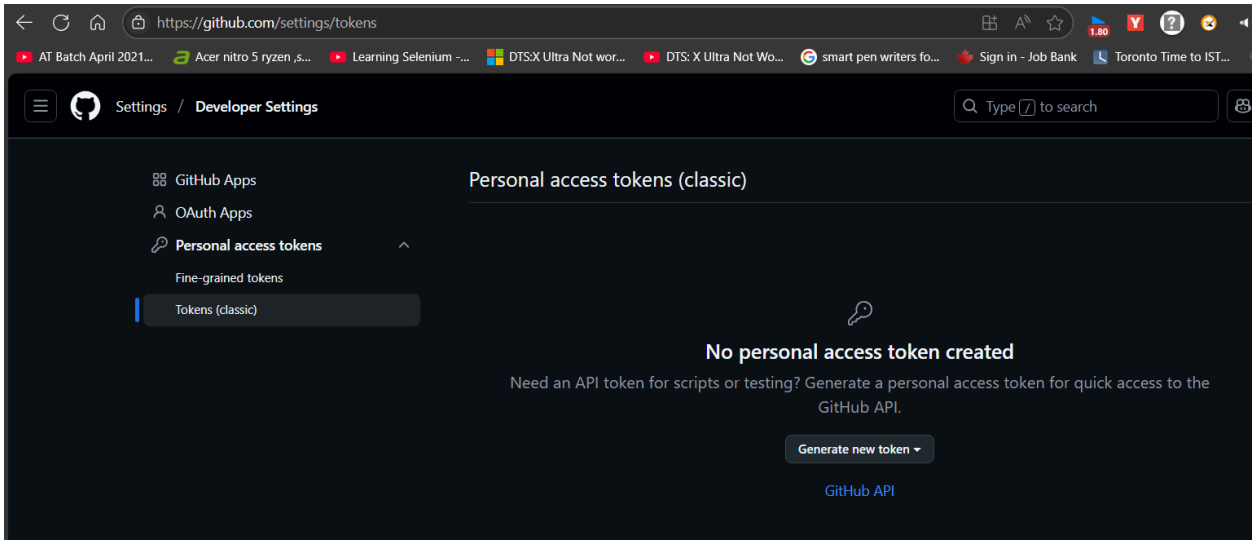


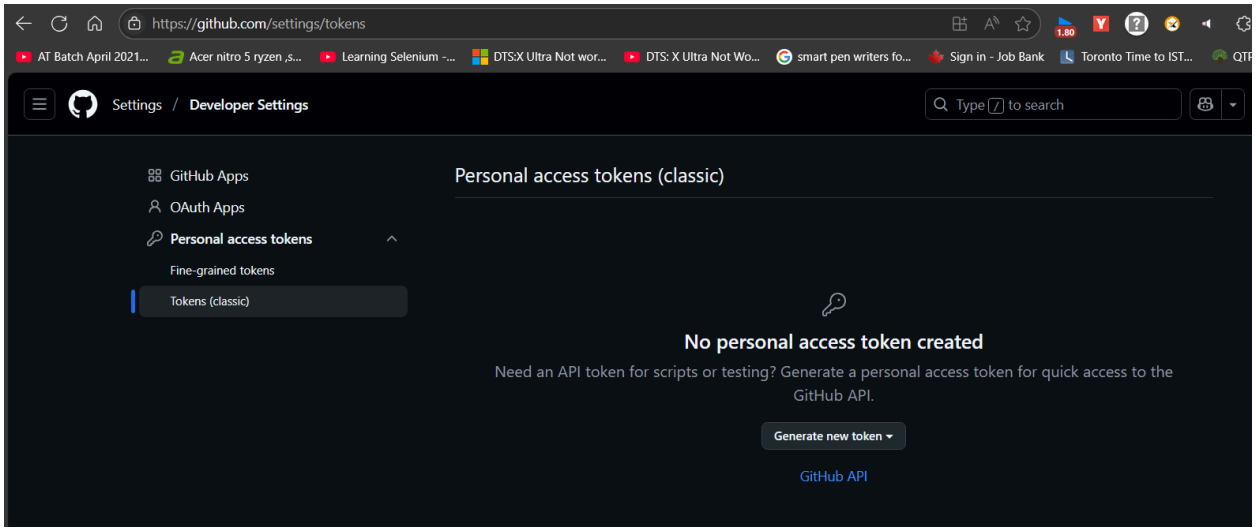
ocd=

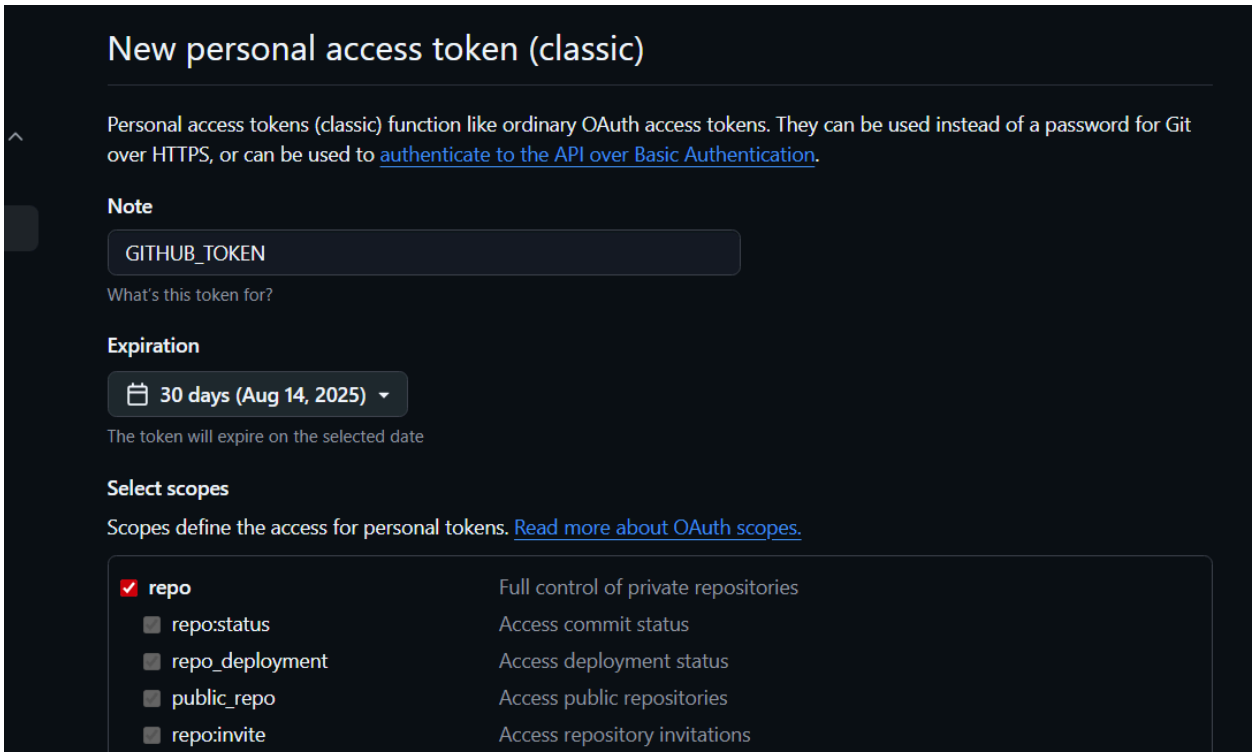


Set up github personal access token –

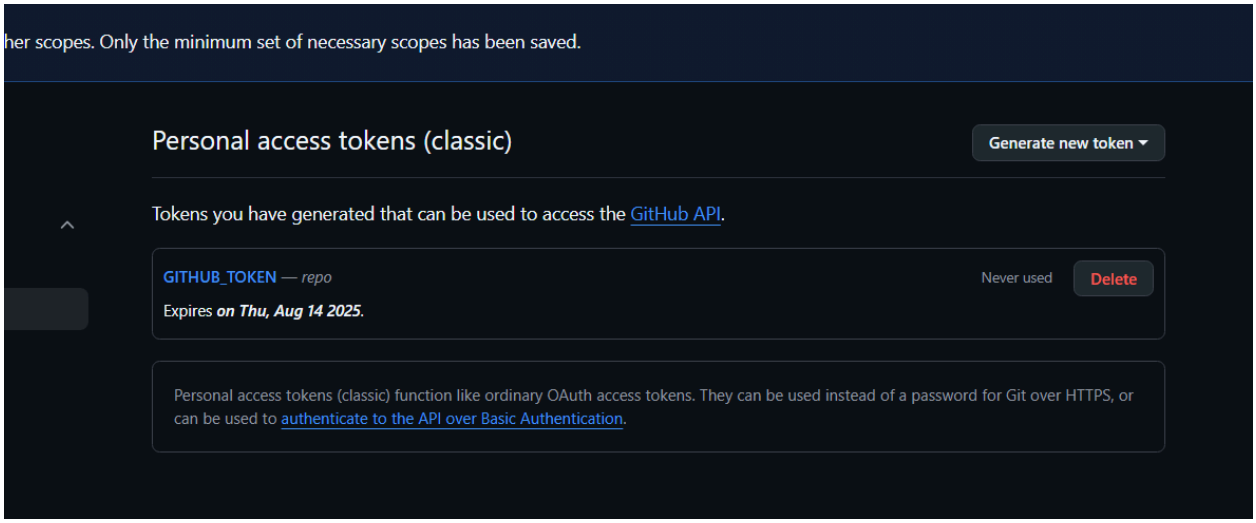
Github settings – developer settings - personal access tokens – tokens classic –


https://github.com/settings/tokens
AT Batch April 2021...
Acer nitro 5 ryzen ,s...
Learning Selenium -...
DTS:X Ultra Not wor...
DTS: X Ultra Not Wo...
smart pen writers fo...
Sign in - Job Bank
Toronto Time to IST...
Settings / Developer Settings
Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)
Personal access tokens (classic)
No personal access token created
Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.
Generate new token
GitHub API


https://github.com/settings/tokens
AT Batch April 2021...
Acer nitro 5 ryzen ,s...
Learning Selenium -...
DTS:X Ultra Not wor...
DTS: X Ultra Not Wo...
smart pen writers fo...
Sign in - Job Bank
Toronto Time to IST...
Settings / Developer Settings
Type / to search
GitHub Apps
OAuth Apps
Personal access tokens
Fine-grained tokens
Tokens (classic)
Personal access tokens (classic)
No personal access token created
Need an API token for scripts or testing? Generate a personal access token for quick access to the GitHub API.
Generate new token
GitHub API



Generated token–

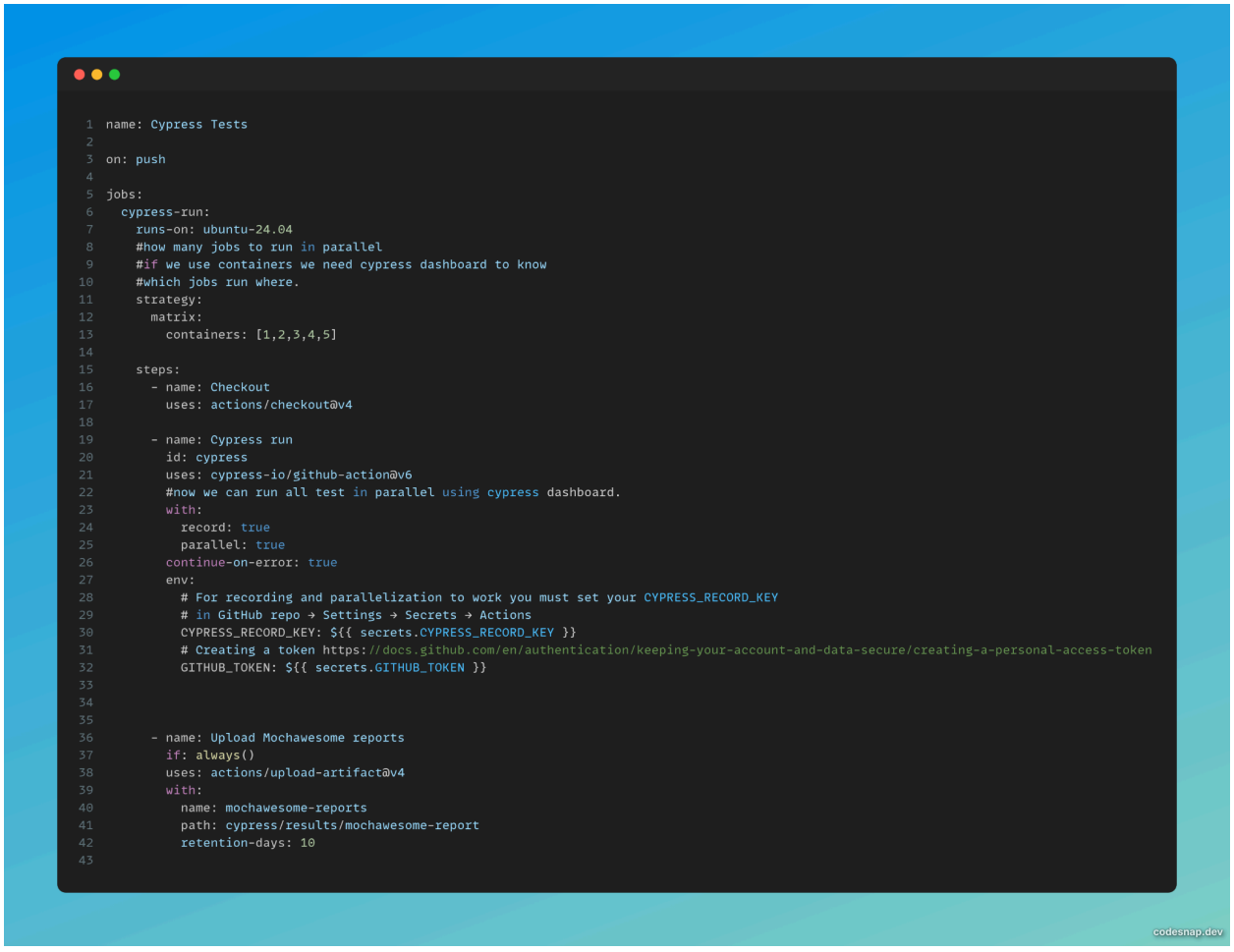


Below one is unclear screenshot.

Edit new file | Preview

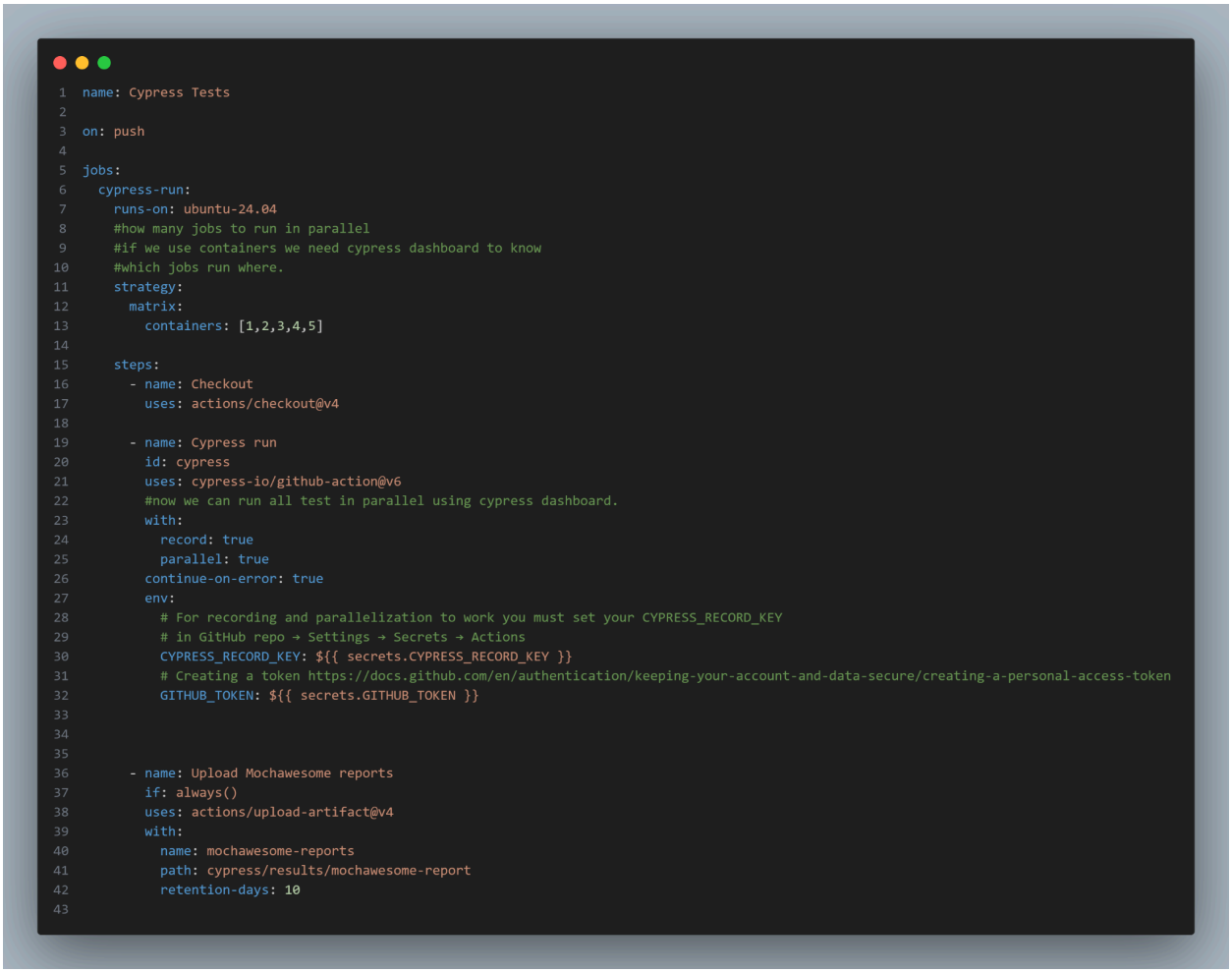
```
      runs-on: ubuntu-20.04
      strategy:
        matrix:
          containers: [1, 2, 3 ,4,5]
      steps:
        - name: Checkout
          uses: actions/checkout@v2
        - name: RunOnChrome
          uses: cypress-io/github-action@v4
          with:
            record: true
            parallel: true
            group: 'Actions example'
          env:
            CYPRESS_RECORD_KEY: 909af02f-29b9-4d4d-af90-815a9b55e30d
        - name: Upload artifacts
```

Clear screenshot-



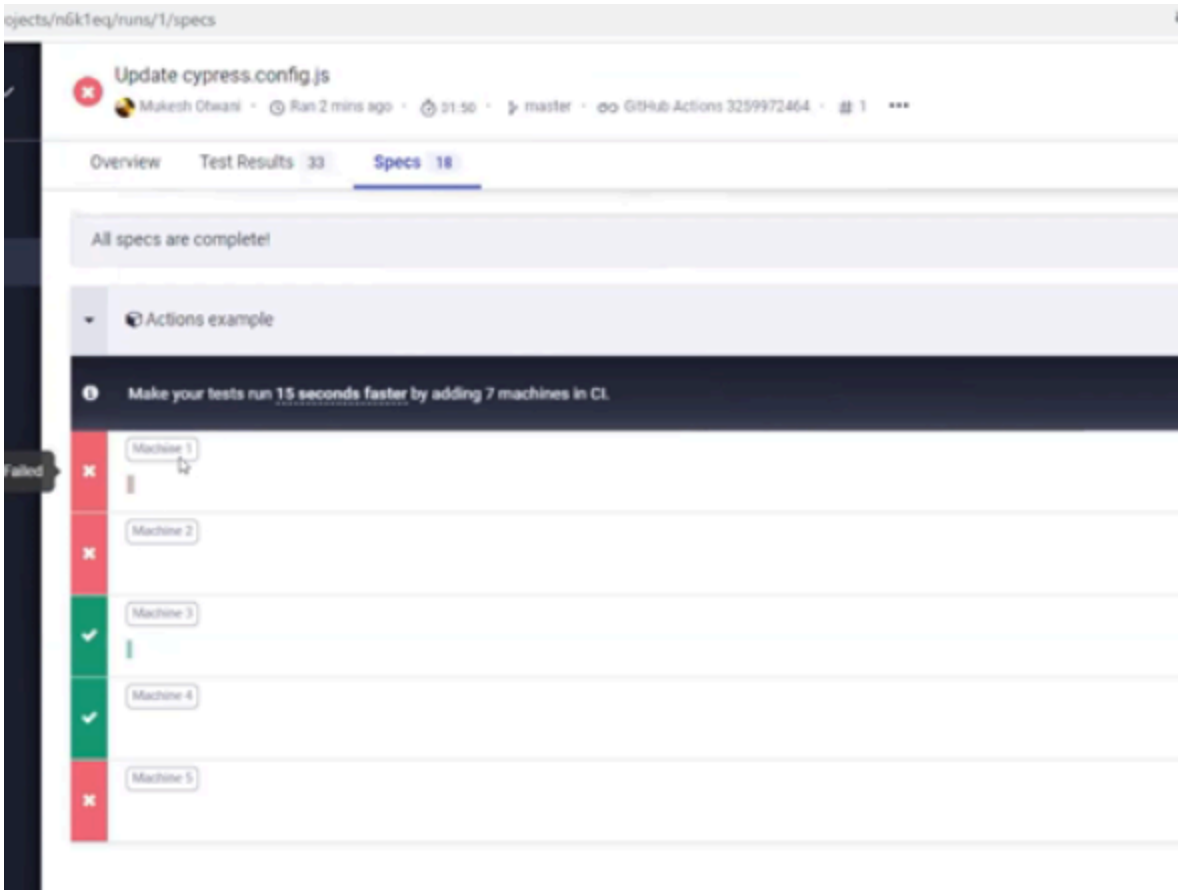
```yaml
name: Cypress Tests

on: push

jobs:
  cypress-run:
    runs-on: ubuntu-24.04
    #how many jobs to run in parallel
    #if we use containers we need cypress dashboard to know
    #which jobs run where.
    strategy:
      matrix:
        containers: [1,2,3,4,5]

    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Cypress run
        id: cypress
        uses: cypress-io/github-action@v6
        #now we can run all test in parallel using cypress dashboard.
        with:
          record: true
          parallel: true
        continue-on-error: true
        env:
          # For recording and parallelization to work you must set your CYPRESS_RECORD_KEY
          # in GitHub repo → Settings → Secrets → Actions
          CYPRESS_RECORD_KEY: ${{ secrets.CYPRESS_RECORD_KEY }}
          # Creating a token https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}



      - name: Upload Mochawesome reports
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: mochawesome-reports
          path: cypress/results/mochawesome-report
          retention-days: 10

```


No difference between web code snap and vs code snap-



```yaml
name: Cypress Tests

on: push

jobs:
  cypress-run:
    runs-on: ubuntu-24.04
    #how many jobs to run in parallel
    #if we use containers we need cypress dashboard to know
    #which jobs run where.
    strategy:
      matrix:
        containers: [1,2,3,4,5]

    steps:
      - name: Checkout
        uses: actions/checkout@v4

      - name: Cypress run
        id: cypress
        uses: cypress-io/github-action@v6
        #now we can run all test in parallel using cypress dashboard.
        with:
          record: true
          parallel: true
        continue-on-error: true
        env:
          # For recording and parallelization to work you must set your CYPRESS_RECORD_KEY
          # in GitHub repo → Settings → Secrets → Actions
          CYPRESS_RECORD_KEY: ${{ secrets.CYPRESS_RECORD_KEY }}
          # Creating a token https://docs.github.com/en/authentication/keeping-your-account-and-data-secure/creating-a-personal-access-token
          GITHUB_TOKEN: ${{ secrets.GITHUB_TOKEN }}



      - name: Upload Mochawesome reports
        if: always()
        uses: actions/upload-artifact@v4
        with:
          name: mochawesome-reports
          path: cypress/results/mochawesome-report
          retention-days: 10

```

Now cypress dashboard can connect with github.
After run check in the dashboard–

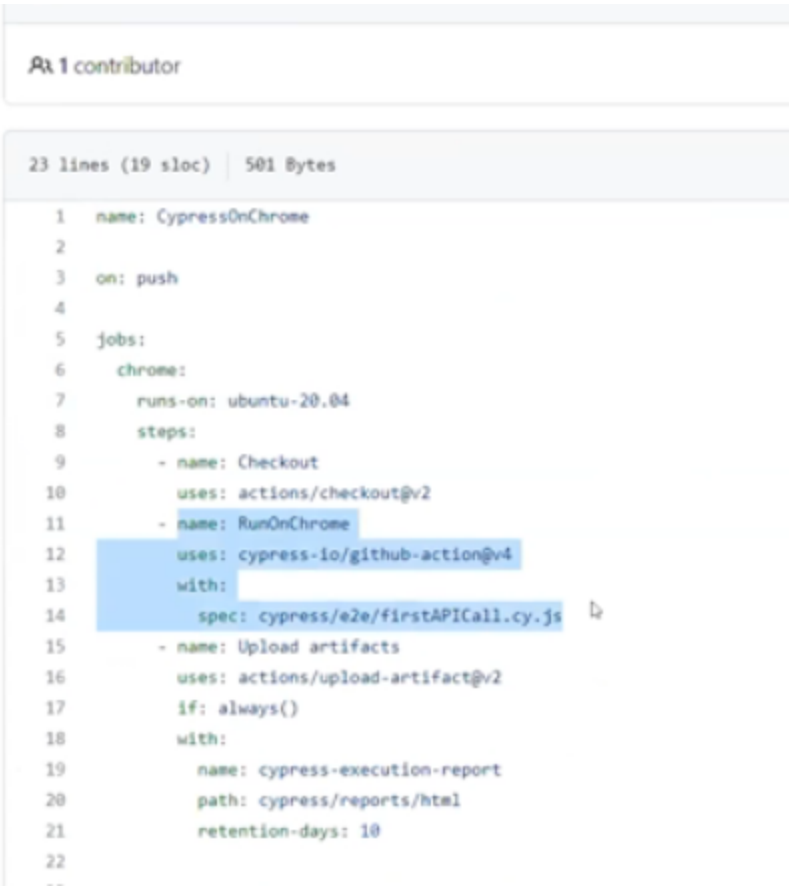


To run on multiple browsers-
Under same job create different step for chrome, ff etc.

To run parallely on different browsers-
Create different jobs one for each browser in the same yaml file.

To run on different environment like qa, staging etc-
Create three steps in one job for each environment and run the script for the specific environment. OR
Three different jobs for each environment.
OR
Three yamls for running on each environment.

Just change highlighted line with browser name in each steps:



Mukesh notes-

16 Oct Notes

1- Basic Workflow which will run all cypress test using single container

```
name: Chrome

on: push

jobs:
  chrome:
    runs-on: ubuntu-18.04
    steps:
      - name: Checkout
        uses: actions/checkout@v1
      - name: Chrome
        uses: cypress-io/github-action@v4
        timeout-minutes: 10
```

```
      - name: Upload artifacts
        uses: actions/upload-artifact@v2
        if: always()
        with:
          name: cypress-execution-report
          path: cypress/reports/html
          retention-days: 10
```

2- Basic Workflow which will run specific spec file test using single container

```
name: Chrome

on: push

jobs:
  chrome:
    runs-on: ubuntu-18.04
    steps:
      - name: Checkout
        uses: actions/checkout@v1
      - name: Chrome
        uses: cypress-io/github-action@v4
        timeout-minutes: 10
        with:
          spec: cypress/e2e/firstAPICall.cy.js
      - name: Upload artifacts
        uses: actions/upload-artifact@v2
        if: always()
        with:
          name: cypress-execution-report
          path: cypress/reports/html
```

3- Workflow which will run all cypress test /spec file in parallel mode

Precondition before running test in parallel

1- Open cypress
2- Login to cypress dashboard using github
3- Create org
4- Create project
5- Copy project id and record key
6- Add project id in cypress.config.js
Sample
projectId: "szztj7"
7 - Use record key in workflow

```
name: Chrome

on: push

jobs:
  chrome:
    runs-on: ubuntu-18.04
    strategy:
      matrix:
        containers: [1, 2, 3,4,5]
    steps:
      - name: Checkout
        uses: actions/checkout@v1
      - name: Chrome
        uses: cypress-io/github-action@v4
        timeout-minutes: 10
        with:
          record: true
          parallel: true
          group: 'Actions example'
        env:
          CYPRESS_RECORD_KEY: please use recordkey
      - name: Upload artifacts
        uses: actions/upload-artifact@v2
        if: always()
        with:
          name: cypress-execution-report
          path: cypress/reports/html
          retention-days: 10
```

Some link to refer

1- Cypress Github Actions- https://docs.cypress.io/guides/continuous-integration/github-actions

2- Parallel- https://docs.cypress.io/guides/guides/parallelization

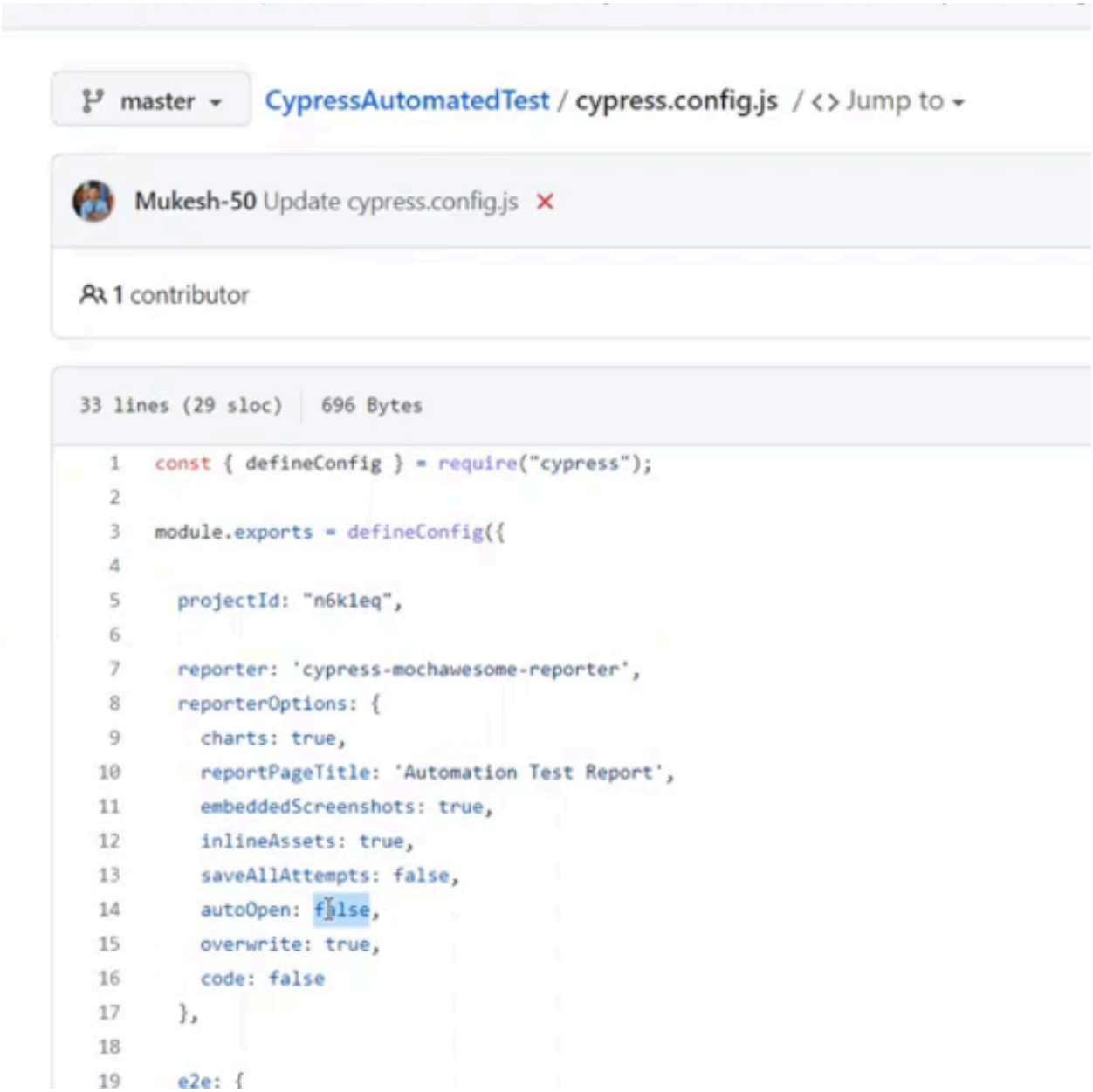
3- Cypress Github Actions - https://github.com/cypress-io/github-action#artifacts

4- Github Hosted Runner - https://docs.github.com/en/actions/using-github-hosted-runners/about-github-hosted-runners

5- All workflow - https://github.com/Mukesh-50/CypressAutomatedTest/tree/master/.github/workflows

6- Docker Step By Step - https://www.youtube.com/watch?v=pBTH9FsgQuc&list=PL6flErFppaj1quyeiIF8Rz7nlBPOZTk6d

MUKESH SIMPLE SOLUTION TO CREATE HTML STUCK PROBLEM IN GITHUB-



Make auto open false in cypress config.js file.
Every test runs and opens report so subsequent test are not able to add to the reports.