1. Install Allure plugin

You need the **shelex/cypress-allure-plugin** (most popular).

```
npm install -D @shelex/cypress-allure-plugin
```

Also install Allure commandline (for generating HTML reports):

```
npm install -g allure-commandline --save-dev
```

2. Project Setup Changes

✅ Add to `cypress/support/e2e.js`

```
import '@shelex/cypress-allure-plugin';   // Allure support
import './commands';                      // your custom commands
require('cypress-xpath');                 // example plugin
require('cypress-file-upload');           // file upload if needed
```

Update `cypress.config.js`

Add reporter configuration:

```
const allureWriter = require('@shelex/cypress-allure-plugin/writer');

module.exports = {
  e2e: {
    setupNodeEvents(on, config) {
      allureWriter(on, config);
      return config;
    },
    baseUrl: "http://localhost:3000,

    env: {
      allure: true,              // enable allure
      allureResultsPath: "allure-results", // where JSON files will be saved
    }
  }
};
```

. Using Allure in Tests + POM

The plugin provides a `Cypress.Allure.reporter` API with helpers.

✅ Start step / End step

Inside your **Page Object class** methods:

```
class LoginPage {
  visit() {
    cy.allure().startStep("Visiting login page");
    cy.visit('/login');
```

```
    cy.allure().endStep();
  }

  enterUsername(username) {
    cy.allure().startStep(`Entering username: ${username}`);
    cy.get('#username').type(username);
    cy.allure().endStep();
  }

  enterPassword(password) {
    cy.allure().startStep("Entering password");
    cy.get('#password').type(password, { log: false }); // don't show real password
    cy.allure().endStep();
  }

  submit() {
    cy.allure().startStep("Click submit");
    cy.get('#loginBtn').click();
    cy.allure().endStep();
  }
}

export default LoginPage;
```

Attaching data to report

Attach text, JSON, or images/screenshots:

```
// inside test or POM method
cy.allure().attachment("User Data", JSON.stringify({ username: "testuser" }),
"application/json");

cy.screenshot("after-login");
cy.allure().attachment("Login Screenshot", "cypress/screenshots/login.png",
"image/png");
```

- `cy.allure().startStep("desc") / endStep()` → log structured steps in reports.
- `cy.allure().attachment("name", data, "mime/type")` → attach files, screenshots, logs.
- Can add labels:

```
cy.allure().severity('critical');
cy.allure().tag('smoke', 'login');
cy.allure().owner('Abhishek');
```

package.json

```
{
  "name": "cypx",
  "version": "1.0.0",
  "description": "",
```

```
  "main": "index.js",
  "directories": {
    "test": "cypress/e2e"
  },
  "scripts": {
    "test": "cypress run --browser chrome --headless",
    "test:chrome": "cypress run --browser chrome --headless=new --env allure=true",
    "test:ui": "cypress open --env allure=true",
    "allure:generate": "allure generate allure-results --clean -o allure-report",
    "allure:open": "allure open allure-report",
    "allure:serve": "allure serve allure-results",
    "test:allure": "npm run test && npm run allure:generate && npm run allure:open",
    "test:debug": "cypress open --env allure=true,debug=true",
    "clean:reports": "rimraf allure-results allure-report cypress/screenshots cypress/videos",
    "ci:test": "cypress run --record --key YOUR_KEY --env allure=true"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "devDependencies": {
    "@shelex/cypress-allure-plugin": "^2.41.2",
    "allure-commandline": "^2.34.1",
    "cypress": "^14.5.4",
    "cypress-xpath": "^2.0.1",
    "mochawesome": "^7.1.3",
    "mochawesome-merge": "^5.0.0",
    "mochawesome-report-generator": "^6.2.0",
    "rimraf": "^6.0.1"
  }
}
```

Data-Driven Testing

- **Fixtures** → Store data in `fixtures/*.json`, load with `cy.fixture()`.
- **Loops** → Use `forEach` or `it.each` style to run same test with multiple datasets.
- **Parameterization** → Pass test data as arguments in `Cypress.env()` or from JSON.
- **External sources** → Read from CSV/Excel/DB via `cy.task()`.

Test Isolation

- **Independent tests** → Each test should set up its own state and not rely on previous test.
- **beforeEach hooks** → Reset app state before each test (`cy.visit()`, clear cookies/storage).
- **Unique data** → Generate random user/email IDs to avoid conflicts across tests.
- **DB clean-up** → Rollback or re-seed DB between tests in integration environments.

Dockerfile (for Cypress + Allure)

```
# Base Cypress image with dependencies
FROM cypress/included:12.17.1

# Install allure-commandline globally
RUN npm install -g allure-commandline --save-dev

# Set work directory
WORKDIR /e2e

# Copy project files
COPY package.json package-lock.json ./
RUN npm install

COPY . .

# Run tests (default command, can be overridden in Jenkins)
CMD ["npx", "cypress", "run", "--env", "allure=true", "--reporter", "cypress-
allure-plugin"]
```

Jenkinsfile (Declarative Pipeline)

```
pipeline {
  agent {
    docker {
      image 'cypress-allure:latest'   // Build this Dockerfile with tag cypress-allure
      args '-u root:root'             // Run as root to avoid permission issues
    }
  }

  environment {
    CYPRESS_baseUrl = 'http://your-app-url.com'
  }

  stages {
    stage('Checkout') {
      steps {
        git branch: 'main', url: 'https://github.com/your-repo/cypress-tests.git'
      }
    }

    stage('Install Dependencies') {
      steps {
        sh 'npm install'
      }
    }

    stage('Run Cypress Tests') {
      steps {
        sh 'npx cypress run --env allure=true --reporter cypress-allure-plugin'
      }
    }

    stage('Generate Allure Report') {
      steps {
        sh 'allure generate allure-results --clean -o allure-report'
```

```
            }
        }

        stage('Publish Allure Report') {
            steps {
                allure includeProperties: false, jdk: '', results: [[path: 'allure-results']]
            }
        }
    }

    post {
        always {
            archiveArtifacts artifacts: 'allure-report/**', fingerprint: true
        }
    }
}

describe('Login Test with Allure', () => {
  it('should login successfully', () => {
    cy.allure().feature('Login')
    cy.allure().story('User logs in with valid credentials')

    cy.allure().step('Visit login page')
    cy.visit('/login')

    cy.allure().step('Enter username')
    cy.get('#username').type('testuser')

    cy.allure().step('Enter password')
    cy.get('#password').type('password123')

    cy.allure().step('Click login button')
    cy.get('button[type="submit"]').click()

    cy.allure().step('Verify login success')
    cy.url().should('include', '/dashboard')
  })
})
```