**Setting Up Your First CI/CD Pipeline**

Here’s how you can set up a basic pipeline:

**1. Create a Workflow File**

Workflows are written in a special file called **YAML**. This file tells GitHub what to do and when to do it.

* Go to your repository.
* Create a folder called .github/workflows (if it doesn’t already exist).
* Inside that folder, create a file like ci-cd.yml.

**2. Add the Pipeline Steps**

Here’s an example workflow file for a **Node.js project** (a common type of web app):

**What’s Happening in This Workflow?**

1. **Trigger**: The pipeline runs automatically every time you:
   * Push new code to the main branch.
   * Open a pull request for the main branch.
2. **Build Job**:
   * Fetch the latest code from GitHub.
   * Set up the programming language environment (Node.js in this case).
   * Install the dependencies.
   * Run tests to make sure the code works as expected.
   * Build the application so it’s ready for deployment.
3. **Deploy Job**:
   * Once everything is tested and built, you can deploy the app (e.g., send it to a live server).

Important information

**1. What does "Open a pull request for the main branch" mean and why do I need to run the pipeline in this case?**

Opening a Pull Request (PR)

A pull request (PR) is a way to ask others to review your code changes and suggest improvements before merging them into the "main" branch (the central, stable branch of your project).

When you open a PR, you are essentially saying, "I made changes in my code, and I’d like to merge them into the main branch. Can someone review them for me?"

Why Run the Pipeline for a PR?

The pipeline ensures that the changes introduced in your PR do not break the main branch.

It automatically:

Builds the code: Checks your code for errors.

Runs tests: Ensures your changes don’t introduce bugs.

Prepares the app: Gets it ready for deployment.

This guarantees that only high-quality, tested code reaches the main branch.

**2. Explain Each Step of the "Build Job" in Plain Words**

Here’s what each step in the **Build Job** does:

1. **Checkout Code**:
   * This step fetches the code from the GitHub repository so the pipeline can work on it.
   * Think of it like downloading the latest version of a document to make sure you’re editing the most up-to-date copy.
2. **Set Up Node.js**:
   * Node.js is an environment for running JavaScript code. This step sets it up.
   * It’s like installing the tools you need before starting a project (e.g., downloading a text editor or a calculator for math homework).
3. **Install Dependencies**:
   * Dependencies are external libraries or packages your project needs to work.
   * For example, if you’re building a web app, you might use a library to handle user login. This step installs all those libraries.
   * It’s like gathering all the supplies (e.g., pens, rulers, erasers) before starting your work.
4. **Run Tests**:
   * This step runs automated tests to check if your code is working correctly.
   * For example, if you wrote a calculator program, there could be a test to ensure 2 + 2 equals 4.
   * It’s like proofreading your essay to make sure there are no spelling or grammar errors.
5. **Build Application**:
   * This step packages your code into a format that’s ready for deployment (sending it live).
   * It’s like converting a Word document into a PDF before sharing it with others—it makes it easier to use and distribute.

**3. Where is this pipeline getting the tests from? Where am I specifying them?**

* **Where Are the Tests Coming From?**
  + The pipeline runs the tests that are defined in your project. These test definitions are typically written in a special folder (e.g., a tests/ folder) or inside your project files.
* **Where Are They Specified?**
  + The run: npm test command in the workflow file tells the pipeline to run the tests.
  + The actual tests are defined in your project’s files. For example:
    - If you’re using Jest (a JavaScript testing library), the tests might be in files like test/example.test.js.
    - The package.json file in your project specifies how to run the tests (e.g., "scripts": { "test": "jest" }).

So, the pipeline doesn’t create the tests—it just runs the ones you’ve already written.