# CI/CD and GitHub Actions

This document explains the basics of CI/CD, GitHub Actions, examples, and common mistakes while using GitHub Actions.

## Table of Contents

1. What is CI/CD
2. What is GitHub Actions
3. Why create a CI/CD pipeline with GitHub Actions
4. Creating a basic CI/CD pipeline with GitHub Actions
5. Common Mistakes
6. Glossary
7. References/Resources

## What is CI/CD

“CI” stands for “Continuous Integration”.

Continuous Integration is the automation process of building, testing and merging code changes of an application to a shared repository.

“CD” stands for “Continuous Delivery” and/or “Continuous Deployment”.

Continuous Delivery ensures that it takes little effort to deliver new code that has been automatically tested and uploaded to the repository.

Continuous Deployment is similar to Continuous Delivery. Instead of delivering new code however, it deploys changes from repository to production.

This new release is usable by customers.

## What is GitHub Actions

GitHub Actions is a CI/CD platform for GitHub.

It allows for the automation of builds, tests, and deployment pipelines.

GitHub also provides Linux, Windows, and macOS virtual machines to run your workflows.

## Why create a CI/CD pipeline with GitHub Actions

GitHub Actions automates most of the manual processes like tests and builds.

GitHub Actions will also perform the actions much faster than performing them manually.

## Creating a basic CI/CD pipeline with GitHub Actions

For this example, i’ll be using a repository that I own.

<https://github.com/notlega/Bloggers>

I will be creating a single workflow that only runs when a pull request is made to the main branch.

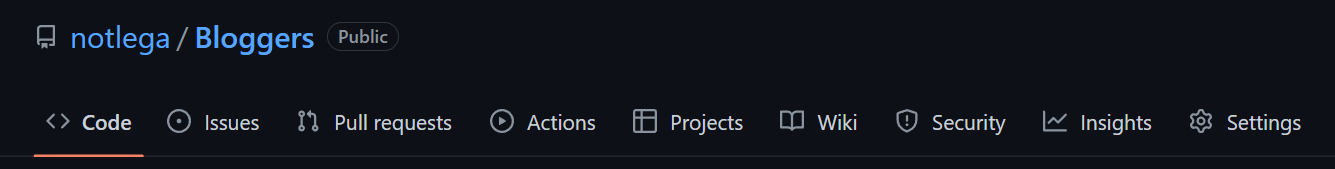
This workflow will test the build of the application only.

### Prerequisites

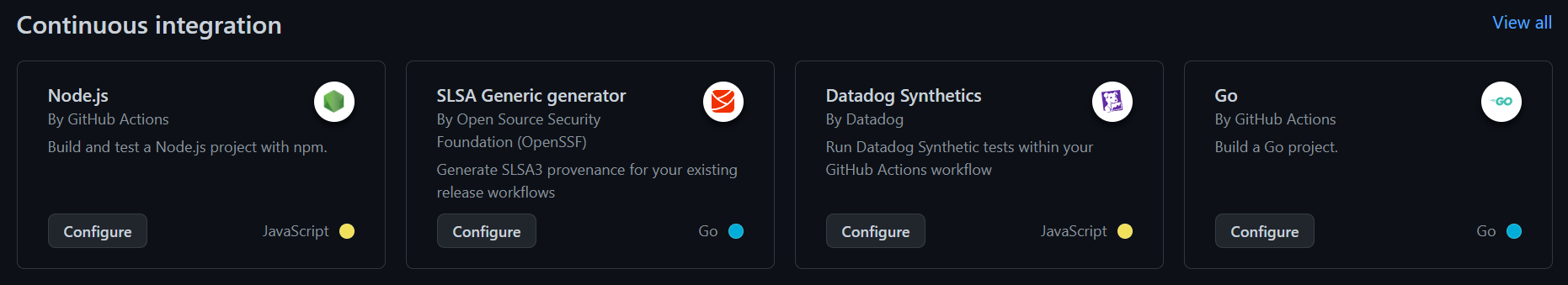
* A GitHub repository
* Some knowledge of YAML
* Some knowledge of Git

### Choosing a workflow

Open your repository and select the Actions tab at the top of your screen.



Under Continuous integration, select Node.js and configure it.



The current screen should display a pre-written yml file.

### Modifying the workflow

Currently, the following code should be shown on screen.

# This workflow will do a clean installation of node dependencies, cache/restore them, build the source code and run tests across different versions of node  
# For more information see: https://docs.github.com/en/actions/automating-builds-and-tests/building-and-testing-nodejs  
  
name: Node.js CI  
  
on:  
 push:  
 branches: ["main"]  
 pull\_request:  
 branches: ["main"]  
  
jobs:  
 build:  
 runs-on: ubuntu-latest  
  
 strategy:  
 matrix:  
 node-version: [14.x, 16.x, 18.x]  
 # See supported Node.js release schedule at https://nodejs.org/en/about/releases/  
  
 steps:  
 - uses: actions/checkout@v3  
 - name: Use Node.js ${{ matrix.node-version }}  
 uses: actions/setup-node@v3  
 with:  
 node-version: ${{ matrix.node-version }}  
 cache: "npm"  
 - run: npm ci  
 - run: npm run build --if-present  
 - run: npm test

This workflow basically runs whenever a push or a pull request is made to the main branch of the repository.

I will now split this workflow up into several parts to explain it more in depth.

name: Node.js CI

The code block above lets GitHub know what is the name of the workflow.

on:  
 push:  
 branches: ["main"]  
 pull\_request:  
 branches: ["main"]

The code block above lets GitHub know when to trigger the workflow.

For example, the current workflow will only trigger if a push or a pull request is made to the main branch.

If we wanted to modify the workflow to only trigger when a pull request is made, we just need to remove the push section of the code block.

Therefore, it should look something like this:

on:  
 pull\_request:  
 branches: ["main"]

The code block above will now only trigger the workflow if a pull request is made to the main branch of the repository.

jobs:  
 build:

The code block above show a simple example of how to define jobs.

A workflow run is made up of one or more jobs, and they run in parallel by default.

The build is the name of the current job, which will be useful when observing the pipeline graph.

Each job runs in a runner environment specified by runs-on:

runs-on: ubuntu-latest

The code block above specifies that the runner environment to use for this job is the latest ubuntu version.

strategy:  
 matrix:  
 node-version: [14.x, 16.x, 18.x]

The code block above defines a matrix of different job configurations.

A job will run for each possible combination of the variables.

In this case, the job will run three times, once for each node version present within the matrix.

steps:  
 - uses: actions/checkout@v3  
 - name: Use Node.js ${{ matrix.node-version }}  
 uses: actions/setup-node@v3  
 with:  
 node-version: ${{ matrix.node-version }}  
 cache: "npm"  
 - run: npm ci  
 - run: npm run build --if-present  
 - run: npm test

The code block above is the main bulk of the workflow.

steps are a sequence of tasks to be carried out.

They can run commands, run setup tasks, run another action within the repository, etc.

name is the display name of the step to be displayed on GitHub.

uses selects an action to be run as a part of a step in the job.

It is a reusable chunk of code.

The action being used in the code block above is a public action that will help us setup Node with the versions specified in the strategy.

with is a map of the input parameters defined by the action.

Each parameter is a key/value pair.

run runs command-line programs using the OS’s shell.

If a name is not provided, the step name will default to the text specified in the run command.

The modified default code should look something like this:

name: Node.js CI  
  
on:  
 pull\_request:  
 branches: ["main"]  
  
jobs:  
 build:  
 runs-on: ubuntu-latest  
  
 strategy:  
 matrix:  
 node-version: [14.x, 16.x, 18.x]  
  
 steps:  
 - uses: actions/checkout@v3  
 - name: Use Node.js ${{ matrix.node-version }}  
 uses: actions/setup-node@v3  
 with:  
 node-version: ${{ matrix.node-version }}  
 cache: "npm"  
 - run: npm ci  
 - run: npm run build --if-present

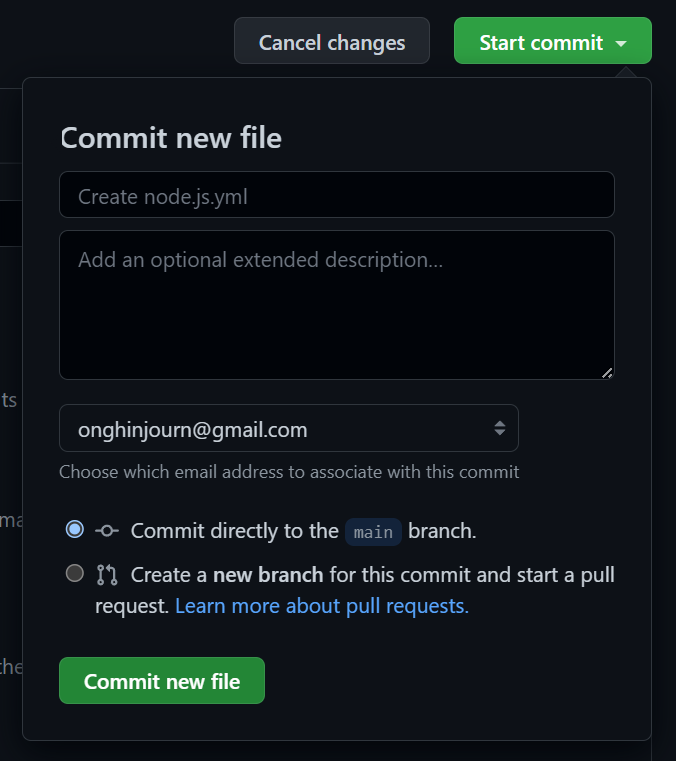
The above modified code block will now be triggered if a pull request is made to the main branch of the repository.

It will test the build of the application only when run.

### Committing the modified workflow

Once the modified code is pasted within the web code editor, click on the Start commit button on the top right.

A popup should appear with default configurations.



Click on the Commit new file button with the default commit configs if you do not wish to add any custom message or description.

### Testing the workflow

After the commit, merge any branches and test the workflow by creating a pull request to the main branch.

A popup should appear at the bottom of the pull request page to run the pipeline.

## Common Mistakes

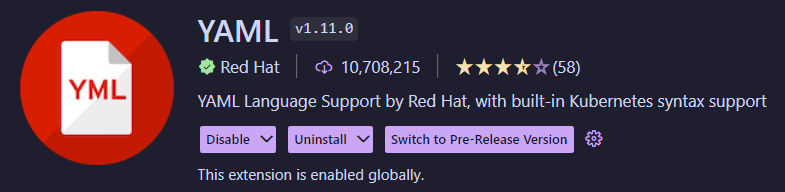
### Linting

Since this might be the first time learning YAML, syntax errors might be normal occurrences.

Testing YAML files is also quite tough as there is no normal way to debug them.

To solve this issue, a linter can be installed to link the file before submitting it.

If you are using Visual Studio Code, I recommend the YAML extension by Red Hat.



### GitHub Secrets

Instead of storing important information like API keys, tokens, or passwords using plaintext, store them using GitHub Secrets.

GitHub Secrets operate similarly to environment variables.

This allows for heightened security as important information will now be secure under another layer of protection instead of being exposed.

## Glossary

**Git** - An open-source version control software

**GitHub** - A company that offers a cloud-based Git repository hosting service

**Build(s)** - A set of executable code that is ready for customer usage

**Test(s)** - A set of executable code that is used to test code

**Deployment** - Pushing changes and/or updates from one environment to another

**Workflows** - A series of activities that are necessary to complete a task

**CI/CD Pipeline** - A series of automated workflows

## References/Resources

* What is Ci/CD and how does it work? (no date) Synopsys. Synopsys. Available at: <https://www.synopsys.com/glossary/what-is-cicd.html> (Accessed: January 25, 2023).
* What is CI/CD? (2022) What is CI/CD VB. Red Hat. Available at: <https://www.redhat.com/en/topics/devops/what-is-ci-cd> (Accessed: January 25, 2023).
* Douglas, B. (2022) How to build a CI/CD pipeline with github actions in four simple steps, The GitHub Blog. GitHub. Available at: <https://github.blog/2022-02-02-build-ci-cd-pipeline-github-actions-four-steps/> (Accessed: January 25, 2023).
* Segura, T. (2022) GitHub actions security best practices [cheat sheet included], GitGuardian Blog - Automated Secrets Detection. GitGuardian. Available at: <https://blog.gitguardian.com/github-actions-security-cheat-sheet/> (Accessed: January 25, 2023).