

## CI Pipeline

### GitHub Actions:

GitHub Actions is GitHub's CI/Cd service. It's the mechanism used to run workflows from development to production systems. Actions are triggered by GitHub events (a pull request is submitted, an issue opened, a PR is merged, etc....) and can execute pretty much any command.

### GitHub Workflows:

A workflow is made of one or more jobs, which are run inside their own virtual machine or container (a runner), that execute one or more steps. A step can be a shell script or an action, which is a reusable piece of code specially packaged for the GitHub CI ecosystem.

Workflow includes:

#### PR Labeler:

A GitHub Action that automatically applies labels to your PRs based on branch name patterns like feature/\* or fix/\*. Can be used in combination with Release Drafter to automatically categorize pull requests.

#### Release Drafter:

Drafts a GitHub release with the changes introduced by a newly created version tag.

Once you've added Release Drafter to your repository, it must be enabled by adding a `.github/release-drafter.yml` configuration file to each repository. The configuration file must reside in your default branch, no other configurations will be accepted.

Example

For example, take the following `.github/release-drafter.yml` file in a repository:

```
template: |  
  ## What's Changed  
  
  $CHANGES
```

As pull requests are merged, a draft release is kept up-to-date listing the changes, ready to publish when you're ready.

Draft

## Draft

 release-drafter drafted this a minute ago


### What's Changed


- 🛠️ Integrate Alien technology #1 (@toolmantim)
- 1 Switch to a monorepo #2 (@toolmantim)
- 🐮 Moar cowbell #3 (@toolmantim)

### PR Validator:


During the creation of a PR or after its closure, we set automation processes helping us to verify the correctness of the code compiling and the successful execution of tests. Example:pytest,Sonarcloud,Wiki-publish,Snyk....etc.


Add more commits by pushing to the `ci:cd` branch on `blexin/raptor-framework`.




**Some checks haven't completed yet**[Hide all checks](#)

1 in progress and 1 successful checks

**Raptor.UI / build (12.x) (pull\_request)** *In progress — This check has started...* [Details](#)

**Raptor.UI / build (12.x) (push)** Successful in 2m [Details](#)

**This branch has no conflicts with the base branch**  
Merging can be performed automatically.

Merge pull request

You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

### Branch Naming Checks:

It is a tool that checks whether the current branch of a git project matches a certain name pattern (specified as a regular expression). This tool is primarily used as a git hook to enforce teams naming conventions.

This Includes:

- Enforcing branch naming policy.
- Prefix commit messages with the branch name ticket.
- branch pattern: Example  
'feat|fix|docs|style|docs|refactor|perf|test|build|ci|chore|revert',

### Change Log:

A changelog is a file that shares a chronologically ordered list of the changes you've made on your project. It's often organized by the version; the usual way is to create a file and start to enumerate all your changes.

### CodeQL Analysis:

CodeQL is the code analysis engine developed by GitHub to automate security checks. You can analyze your code using CodeQL and display the results as code scanning alerts.

### Build And Push:

This Includes:

- Getting the image tag which is nothing but the Release Draft Tag
- Building the image with the tag
- Snyk Vulnerability test for the images
- Pushing the image to Azure Container Registry.

### Monorepo Update:

The monorepo for this submodule is updated with the recent code changes.

### Monorepo:

It is a git repository that contains all the modules that compose a software solution. This makes refactoring and code dependency management easier.

Submodules:

- Backend Repo
- Frontend Repo

