Git Branching Naming Convention - Best Practices

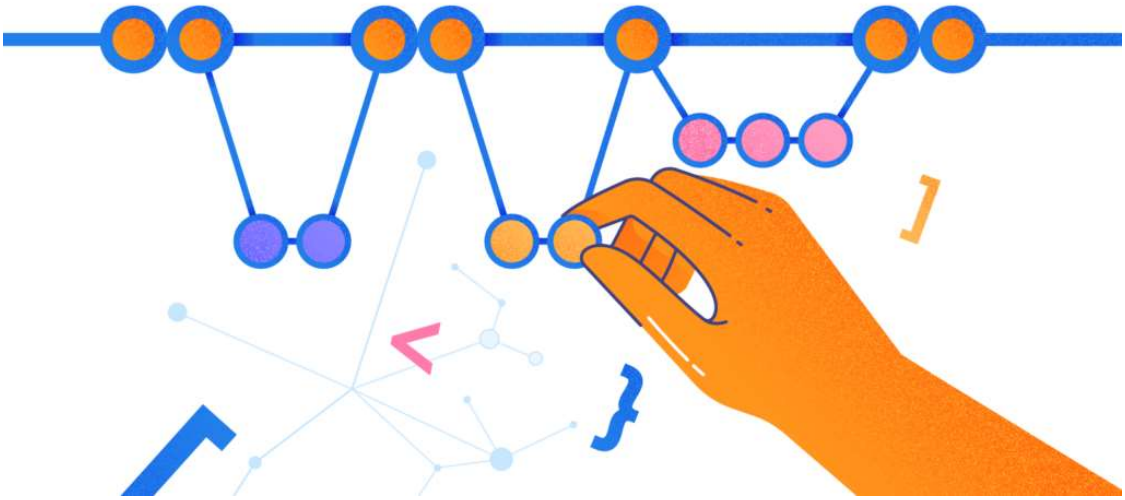
Hiteshjethva March 5, 2021 11,490 0

In this guide, we will show you how the git branching naming convention works with some examples

Git offers flexible branching strategies, but what does it mean? In simple words, a branching strategy is a set of rules, a convention that helps teams and developers – they can follow these rules and conventions to create a new branch, its flow, etc.

Not using appropriate naming conventions leads to confusion and complicates the code maintenance team. We can't ignore Git best practices in branching naming conventions.

Git branching strategies allow separation of work. Broadly, we can divide Git branches into two categories: Regular & Temporary Branches.



Regular Git Branches

These branches will be available in your repository on permanent bases. Their naming convention is simple and straightforward.

- Development (dev) is the main development branch. The dev branch's idea is to make changes in it and restrict the developers from making any changes in the master branch directly. Changes in the dev branch undergo reviews and, after testing, get merged with the master branch.
- Master (master) is the default branch available in the Git repository. It should be stable all the time and won't allow any direct check-in. You can only merge it after code review. All team members are responsible for keeping the master stable and up-to-date.
- QA (QA), or test branch, contains all the code for QA testing and automation testing of all changes implemented. Before any change goes to the production environment, it must undergo the QA testing to get a stable codebase.

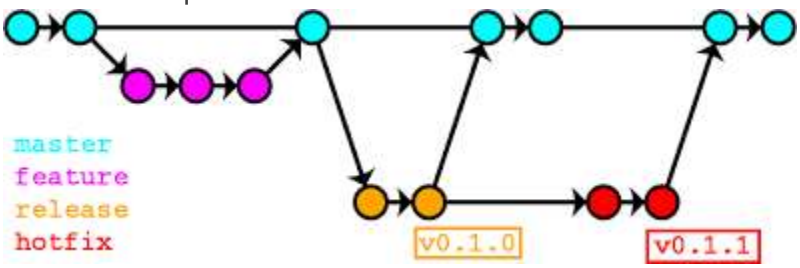
Temporary Git Branches

As the name indicates, these are the branches that can be created and deleted when needed. They can be as follows:

- Bug Fix
- Hot Fix
- Feature Branches
- Experimental Branches
- WIP branches

There are many formats and naming conventions recommended by experts for temporary branches.

Here is a simple workflow of Git branches.



Git Branching Naming Convention

In this article, I'll review and share the seven best naming conventions which I used personally in the past to ensure their efficiency.

1. Start branch name with a Group word

It is one of the best practices. The group word can be anything to match your workflow.

I like short words like the following:

- Bug – The bug which needs to be fixed soon
- WIP – The work is in progress, and I am aware it will not finish soon

By looking at the branch name, you can understand what this Git branch is about and its purpose.

Have a look at the below examples:

- bug-logo-alignment-issue – the developer is trying to fix the logo alignment issue;
- wip-ioc-container-added – the branch relates to the task to add an IoC container in progress.

2. Use Unique ID in branch names

You can use the issue tracker Id in your branch name. I prefer this method when I work on fixing some bugs. For instance:

```
wip-8712-add-testing-module
```

The name shows that the branch applies to the task of adding a testing module, the tracking Id of the issue is 8712, and the work is in progress.

One more advantage of using an external tracking ID in the branch name is the possibility to track the progress from an external system.

3. Use Hyphen or Slash as Separators

Many developers use slash as a separator, and many use hyphens. Which one to use – depends on you and your team's preferences.

My opinion is that hyphens make the name more comfortable to read, so it's a suitable separator in branch names. You can use slashes, hyphens, and underscores. The point is to be consistent.

There are two main advantages of using a separator in the branch name:

- It increases the readability and helps to avoid confusion;
- It makes it easier to manage, especially if you are dealing with many branches.

Example 1. Git branch name without any separator:

```
featureupgradejqueryversionloginmodule
```

Example 2. By adding a separator (in this case, it is an underscore), you make the Git branch name readable:

```
feature_upgrade_jquery_version_login_module
```

4. Git Branch with Author Name

Many companies prefer to add authors' names into the branch names according to the format below:

```
<author>_<branch-type>_<branch-name>
```

E.g., rajeev.bera_feature_new-experimental-changes

This method allows for easy tracking of different developers' work and progress with additional systems.

5. Avoid using numbers only

Some developers only use the issue Id in the branch name, which is not helpful in the work progress.

For instance, there is a branch name 9912 – what should this magic number tell us? It only means more confusion and risk of mistakes, especially during merging with other git branches.

6. Avoid using all naming convention simultaneously

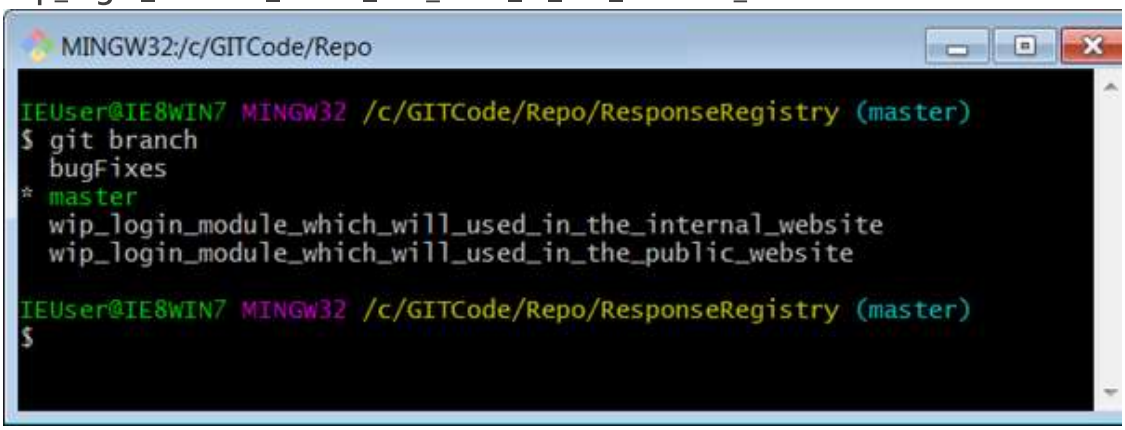
Mixing and matching all Git branch naming conventions are not the best practice. It only adds confusion and complicates the overall processes.

A team should decide the naming conventions to use in work once, and stick to them. Consistency is the most critical thing.

7. Avoid long descriptive names for long-lived branches

The essential quality of a branch name is that it should be precise and informative. Let's have a look at some examples again:

wip_login_module_which_will_used_in_the_public_website
wip_login_module_which_will_used_in_the_internal_website



There, the branch names are long and detailed. It is not necessary. Instead, you might use the following variant:

wip_feature_login_module

This name is short, but it explains the purpose of this branch.

Conclusion

The Git branching model is powerful, but you need to manage the branches correctly and effectively. One of the necessary factors is following the same conventions by all teams, particularly – the naming conventions for the local repository.

To make sure your team is using the agreed conventions, enforce the standards. One of the easiest ways is to use Git hooks, like the pre-commit