

**couchcamote**

Posted on Mar 31, 2020 • Updated on Jun 16, 2020



103



10



2



3



3

Git Branch Naming Convention

#git #branching #name #convention

Working on a big company with projects that could scale from a one-man team then suddenly to a 20 developers team, having a manageable code repository is a need. Most *Proof of Concept* projects start with a repository with all changes being applied directly to the **master** branch. Elevating one into a proper big scale repository is a common path being taken by new developers when their small-scale project is suddenly noticed.

On my end, having to work on dozens of different projects like these, I came up with this branch naming convention.

I separated these branches into two categories:

Code Flow Branches

These branches which we expect to be permanently available on the repository follow the flow of code changes starting from development until the production.

- **Development** (*dev*)
All new features and bug fixes should be brought to the development branch. Resolving developer codes conflicts should be done as early as here.
- **QA/Test** (*test*)
Contains all codes ready for QA testing.
- **Staging** (*staging* , Optional)
It contains tested features that the stakeholders wanted to be available either for a demo or a proposal before elevating into the production. Decisions are made here if a feature should be finally brought to the production code.
- **Master** (*master*)
The production branch, if the repository is published, this is the default branch being presented.

Except for Hotfixes, we want our codes to follow a one-way merge starting from development > test > staging > production.

Temporary Branches

As the name implies, these are disposable branches that can be created and deleted by need of the developer or deployer.

- **Feature**

Any code changes for a new module or use case should be done on a feature branch. This branch is created based on the current development branch. When all changes are **Done**, a Pull Request/Merge Request is needed to put all of these to the development branch.

Examples:

- feature/integrate-swagger
- feature/JIRA-1234
- feature/JIRA-1234_support-dark-theme

It is recommended to use all lower caps letters and hyphen (-) to separate words unless it is a specific item name or ID. Underscore (_) could be used to separate the ID and description.

- **Bug Fix**

If the code changes made from the feature branch were rejected after a release, sprint or demo, any necessary fixes after that should be done on the bugfix branch.

Examples:

- bugfix/more-gray-shades
- bugfix/JIRA-1444_gray-on-blur-fix

- **Hot Fix**

If there is a need to fix a blocker, do a temporary patch, apply a critical framework or configuration change that should be handled immediately, it should be created as a Hotfix. It does not follow the scheduled integration of code and could be merged directly to the production branch, then on the development branch later.

Examples:

- hotfix/disable-endpoint-zero-day-exploit
- hotfix/increase-scaling-threshold

- **Experimental**

Any new feature or idea that is not part of a release or a sprint. A branch for playing around.

Examples:

- experimental/dark-theme-support

- **Build**

A branch specifically for creating specific build artifacts or for doing code coverage runs.

Examples:

- build/jacoco-metric

- **Release**

A branch for tagging a specific release version

Examples:

- release/myapp-1.01.123

Git also supports tagging a specific commit history of the repository. A release branch is used if there is a need to make the code available for checkout or use.

- **Merging**

A temporary branch for resolving merge conflicts, usually between the latest development and a feature or Hotfix branch. This can also be used if two branches of a feature being worked on by multiple developers need to be merged, verified and finalized.

Examples:

- merge/dev_lombok-refactoring
- merge/combined-device-support

Any organization can come up with their own convention. This applies to my current Team's need and there could be a better approach which could improve upon these. What are your conventions on your own organization?

Top comments (15) ⚡



Shahed • Dec 21 '20 • Edited



Very thorough summary of all usual conventions. Just want to add after the Black Lives Matter movement, git provided option to use names other than `master`, the option is: `git init --initial-branch=stable`. Hence technically speaking `master` is deprecated in favour of other names such as `stable` or `main`.



jstewart8053 • Jan 26 '21 • Edited



You mean after the Black Lives Matter movement. It's not a 'thing'. *smh*
Very helpful post though, what do you think about emojis in commits? I have seen a lot on the issue lately.



Shahed • Feb 5 '21 • Edited



Sure you are right, since English is not my native language 😅. The correct word is movement, and thanks. I edited my original reply.



jstewart8053 • Feb 9 '21



No worries :)



Ravi Ponamgi • May 6 '21



Liked "No worries", considering the OP was made with the best of intentions.



slidenerd • Jul 15 '22 • Edited



What do you call a branch where you start a python project empty, add a gitignore, readme, poetry, tox, pre-commit, add a main.py setup pyproject.toml install pytest and run some tests inside the tests directory which you also create



EvilKittenLord • Oct 2 '23



Everything needs a naming scheme. :)
Folks might be interested in the naming scheme I've been using that enables automatic semver detection.
[github.com/marketplace/actions/git...](https://github.com/marketplace/actions/git-branch-naming-convention)



Eduardo Zepeda • Nov 10 '20



This is so useful, most people should read this post.



Clifton Long Jr. • Aug 30 '20



Great post, and very helpful! 😊



Roberto Tonino • Aug 19 '20



Great post, thanks! Definitely gonna use this convention :)

I would suggest a branch for dependencies upgrading too (especially ones with breaking changes), what do you think?



couchcamote 🌟 • Oct 15 '20 • Edited



Glad to have helped.

For new dependencies upgrading we can consider that as a new feature. Basically, it is like new code changes with new libraries and code updated to use them. It should also follow the dev-test-stage-prod flow and it usually has planned schedule for release.



ponyjackal • Nov 12 '20



Thanks for your post,



Ravi Ponamgi • May 6 '21



Great summary!



ahmad faaiz • Jun 29 '20



Great! Thanks~



Adjie Djaka Permana • Dec 18 '20



Much new knowledge, thanks for sharing it

[Code of Conduct](#) • [Report abuse](#)

Update Your DEV Experience Level:

Content

What is your approximate experience level (1-5)?

1 Novice	2 Beginner	3 Mid-level	4 Advanced	5 Expert
-------------	---------------	----------------	---------------	-------------

This will not be displayed on your profile or anywhere publicly. It helps gently determine what content you are shown along with tags you follow etc.

Go to [your customization settings](#) to nudge your home feed to show content more relevant to your developer experience level. 🛠️



couchcamote

Full Stack Software Engineer Codes in Java, Python, JavaScript, C, C++

JOINED

Sep 17, 2019

Trending on DEV Community 🔥



What are you learning about this weekend? 🧠

#codenewbie #learning #beginners #discuss



Meme Monday

#discuss #watercooler #jokes



SOLID Principles: They're Rock-Solid for Good Reason!

#programming #productivity #career #beginners

DEV Community



Do you have an org account on DEV?

Maximize the impact of your presence on DEV with our new **Pro Tools** suite.

Get access to:

- Advanced analytics and insights
- The opportunity to use **this space** next to your own posts to help your readers engage more deeply with your message.

[Try Pro Tools for Free](#)