What

A Git Workflow is a recipe or recommendation for how to use Git to accomplish work in a consistent and productive manner. Git workflows encourage users to leverage Git effectively and consistently.

····——————————

Centralized Workflow

the Centralized Workflow uses a central repository to serve as the single point-of-entry for all changes to the project.

Instead of trunk, the default development branch is called master and all changes are committed into this branch. This workflow doesn’t require any other branches besides master.

create the central repository on a server

each developer creates a local copy of the entire project

a developer can make changes using the standard Git commit process: edit, stage, and commit.

fetch the updated central commits and rebase their changes on top of them to resolve a merge conflict

push their changes to the central repository,

——————————————

Feature Branch Workflow

The core idea behind the Feature Branch Workflow is that all feature development should take place in a dedicated branch instead of the master branch.

master represents the official project history.

developers create a new branch every time they start work on a new feature.

developers can edit, stage, and commit changes to a feature branch.

“pull request” to Resolve feedback

Merge your pull request to server

Gitflow

The Gitflow Workflow defines a strict branching model designed around the project release.

In addition to feature branches, it uses individual branches for preparing, maintaining, and recording releases.

Instead of a single master branch, this workflow uses two branches to record the history of the project. The master branch stores the official release history, and the develop branch serves as an integration branch for features.

Each new feature should reside in its own branch, which can be [pushed to the central repository](https://www.atlassian.com/git/tutorials/syncing/git-push) for backup/collaboration. But, instead of branching off of master, feature branches use develop as their parent branch. When a feature is complete, it gets [merged back into develop](https://www.atlassian.com/git/tutorials/using-branches/git-merge). Features should never interact directly with master.

release branches are based on the develop branch. Once develop has acquired enough features for a release (or a predetermined release date is approaching), you fork a release branch off of develop. Once the release is ready to ship, it will get merged it into master and develop, then the release branch will be deleted.

Maintenance or “hotfix” branches are based on master instead of develop. As soon as the fix is complete, it should be merged into both master and develop (or the current release branch), and master should be tagged with an updated version number.

Forking Workflow

Instead of using a single server-side repository to act as the “central” codebase, it gives every developer their own server-side repository. This means that each contributor has not one, but two Git repositories: a private local one and a public server-side one.

1. A developer 'forks' an 'official' server-side repository as their own server-side copy.
2. The new server-side copy is cloned to their local system.

The developer makes changes on the new branch and they push the commit to their own public repository

1. The developer opens a pull request from the new branch to the 'official' repository.
2. The pull request gets approved for merge and is merged into the original server-side repository

Github flow

GitHub Flow is a lightweight, branch-based workflow that supports teams and projects where deployments are made regularly.

create a branch to experiment and commit changes

Whenever you add, edit, or delete a file, you're making a commit, and adding them to your branch.

open a Pull Request at any point during the development process to initiate discussion about your commits

Discuss and review your code to fix bugs in your branch and push up the change

Once your pull request has been reviewed and the branch passes your tests, you can deploy your changes to verify them in production and merge your code into the master branch.

[深入理解学习Git工作流（git-workflow-tutorial）](https://segmentfault.com/a/1190000002918123) <https://segmentfault.com/a/1190000002918123#articleHeader20>

A successful Git branching model » nvie.com - <http://nvie.com/posts/a-successful-git-branching-model/?utm_source=qq&utm_medium=social>

Understanding the GitHub Flow · GitHub Guides - <https://guides.github.com/introduction/flow/?utm_source=qq&utm_medium=social>

git-workflow-tutorial <https://www.atlassian.com/git/tutorials/comparing-workflows>

Git 工作流程<http://www.ruanyifeng.com/blog/2015/12/git-workflow.html>