**Content**

[1. Git Workflows 2](#_Toc29559898)

[1. Gitflow 2](#_Toc29559899)

[2. Merge A-Z 3](#_Toc29559900)

[1. git stash 3](#_Toc29559901)

[3. git aliasis 5](#_Toc29559902)

[4. Others 6](#_Toc29559903)

[1. .gitignore 6](#_Toc29559904)

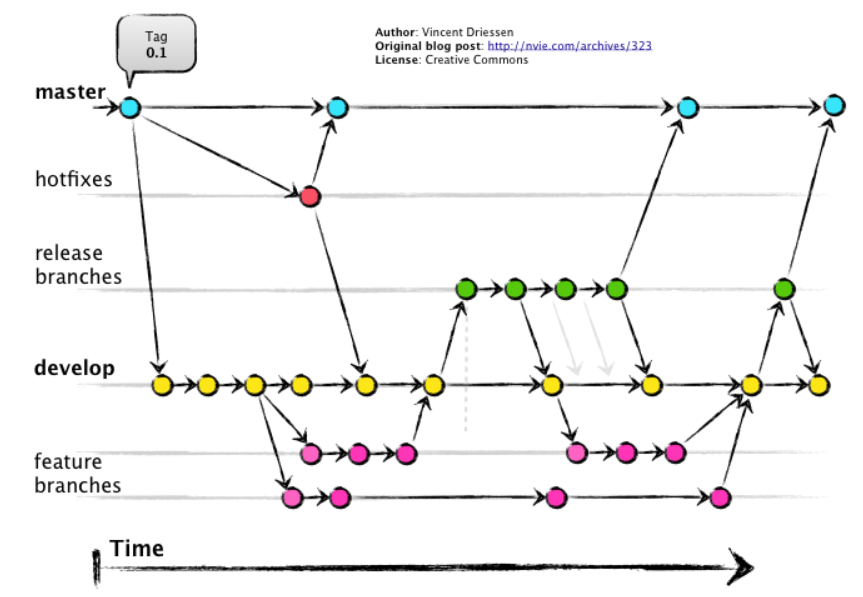
[2. Reset vs Revert 6](#_Toc29559905)

[3. Amend a commit 6](#_Toc29559906)

1. Git Workflows

A Git Workflow is a recipe or recommendation for how to use Git to accomplish work in a consistent and productive manner.

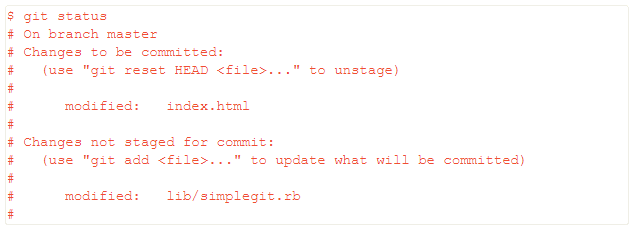
* Does this workflow scale with team size?
* Is it easy to undo mistakes and errors with this workflow?
* Does this workflow impose any new unnecessary cognitive overhead to the team?
  1. Gitflow

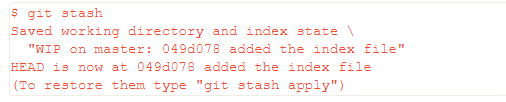


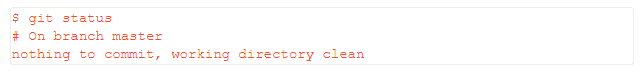
1. Merge A-Z
   1. git stash

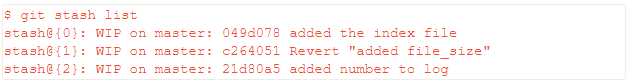
Often, when you’ve been working on part of your project, things are in a messy state and you want to switch branches for a bit to work on something else. The problem is, you don’t want to do a commit of half-done work just so you can get back to this point later. The answer to this issue is the git stash command.

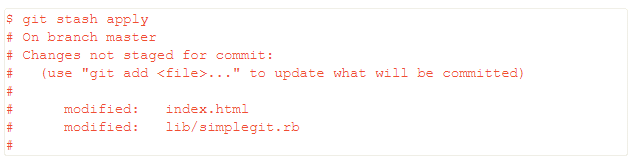
Stashing takes the dirty state of your working directory — that is, your modified tracked files and staged changes — and saves it on a stack of unfinished changes that you can reapply at any time.



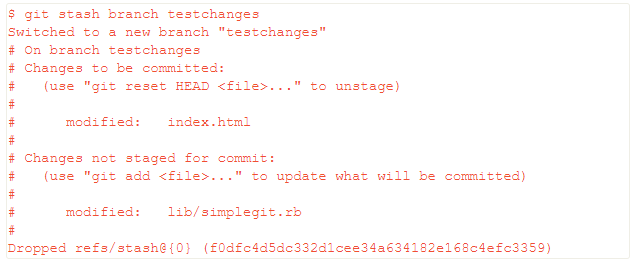












1. git aliases
2. Others
   1. .gitignore

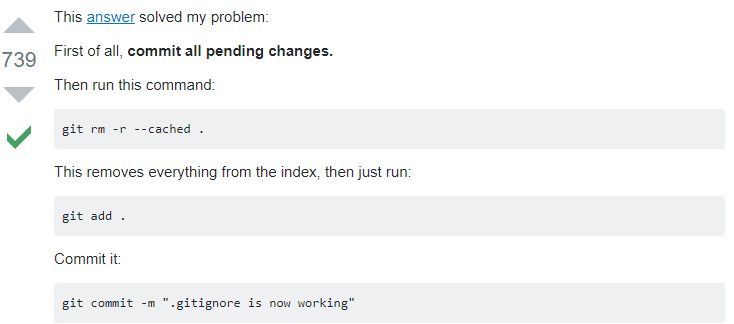
The files/folder in your version control will not just delete themselves just because you added them to the .gitignore. They are already in the repository and you have to remove them. You can just do that with this:

(Remember to commit everything you've changed before you do this.)

git rm -rf --cached .

git add .

This removes all files from the repository and adds them back (this time respecting the rules in your .gitignore).



* 1. Reset vs Revert
  2. Amend a commit
* Upon unsuccessful merge request?!