Vishnu's Blog

+ Follow

HOME SYSTEM DESIGN CHEATSHEETS COURSES SUPPORT ME CONTACT ME BADGES

Git Cheatsheet

Git commands that developers should know

vishnu chilamakuru
Published on Apr 1, 2021
8 min read

Subscribe to my newsletter and never miss my upcoming articles

Email address SUBSCRIBE

Index

Setup

Show current configuration:

COPY

```
$ git config --list
```

Show repository configuration:

COPY

```
$ git config --local --list
```

Show global configuration:

COPY

```
$ git config --global --list
```

Show system configuration:

COPY

```
$ git config --system --list
```

Set a name that is identifiable for credit when review version history:

COPY

```
$ git config --global user.name “[firstname lastname]”
```

Set an email address that will be associated with each history marker:

COPY

```
$ git config --global user.email “[valid-email]”
```

Set automatic command line coloring for Git for easy reviewing:

COPY

```
$ git config --global color.ui auto
```

Set global editor for commit

COPY

```
$ git config --global core.editor vi
```

Configuration Files

Repository specific configuration file [--local]:

COPY

```
<repo>/.git/config
```

User-specific configuration file [--global]:

COPY

```
~/.gitconfig
```

System-wide configuration file [--system]:

COPY

```
/etc/gitconfig
```

Create

Clone an existing repository:

There are two ways:

Via SSH

COPY

```
$ git clone ssh://user@domain.com/repo.git
```

Via HTTP

COPY

```
$ git clone http://domain.com/user/repo.git
```

Create a new local repository in the current directory:

COPY

```
$ git init
```

Create a new local repository in a specific directory:

COPY

```
$ git init <directory>
```

Local Changes

Changes in working directory:

COPY

```
$ git status
```

Changes to tracked files:

COPY

```
$ git diff
```

See changes/difference of a specific file:

COPY

```
$ git diff <file>
```

Add all current changes to the next commit:

COPY

```
$ git add .
```

Add some changes in <file> to the next commit:

COPY

```
$ git add -p <file>
```

Add only the mentioned files to the next commit:

COPY

```
$ git add <filename1> <filename2>
```

Commit all local changes in tracked files:

COPY

```
$ git commit -a
```

Commit previously staged changes:

COPY

```
$ git commit
```

Commit with message:

COPY

```
$ git commit -m 'message here'
```

Commit skipping the staging area and adding message:

COPY

```
$ git commit -am 'message here'
```

Commit to some previous date:

COPY

```
$ git commit --date="`date --date='n day ago'`" -am "<Commit Message Here>"
```

Change last commit:

Don't amend published commits!

COPY

```
$ git commit -a --amend
```

Amend with last commit but use the previous commit log message

Don't amend published commits!

COPY

```
$ git commit --amend --no-edit
```

Change committer date of last commit:

COPY

```
GIT_COMMITTER_DATE="date" git commit --amend
```

Change Author date of last commit:

COPY

```
$ git commit --amend --date="date"
```

Move uncommitted changes from current branch to some other branch:

COPY

```
$ git stash
$ git checkout branch2
$ git stash pop
```

Restore stashed changes back to current branch:

COPY

```
$ git stash apply
```

Restore particular stash back to current branch:

> *{stash_number}* can be obtained from `git stash list`

COPY

```
$ git stash apply stash@{stash_number}
```

Remove the last set of stashed changes:

COPY

```
$ git stash drop
```

Search

A text search on all files in the directory:

COPY

```
$ git grep "Hello"
```

In any version of a text search:

COPY

```
$ git grep "Hello" v2.5
```

Show commits that introduced a specific keyword

COPY

```
$ git log -S 'keyword'
```

Show commits that introduced a specific keyword (using a regular expression)

COPY

```
$ git log -S 'keyword' --pickaxe-regex
```

Commit History

Show all commits, starting with newest (it'll show the hash, author information, date of commit and title of the commit):

COPY

```
$ git log
```

Show all the commits(it'll show just the commit hash and the commit message):

COPY

```
$ git log --oneline
```

Show all commits of a specific user:

COPY

```
$ git log --author="username"
```

Show changes over time for a specific file:

COPY

```
$ git log -p <file>
```

Display commits that are present only in remote/branch in right side

COPY

```
$ git log --oneline <origin/master>..<remote/master> --left-right
```

Who changed, what and when in <file>:

COPY

```
$ git blame <file>
```

Show Reference log:

COPY

```
$ git reflog show
```

Delete Reference log:

COPY

```
$ git reflog delete
```

Move or Rename

Rename a file:

Rename Index.txt to Index.html

COPY

```
$ git mv Index.txt Index.html
```

Branches and Tags

List all local branches:

COPY

```
$ git branch
```

List local/remote branches

COPY

```
$ git branch -a
```

List all remote branches:

COPY

```
$ git branch -r
```

Switch HEAD branch:

COPY

```
$ git checkout <branch>
```

Checkout single file from different branch

COPY

```
$ git checkout <branch> -- <filename>
```

Create and switch new branch:

COPY

```
$ git checkout -b <branch>
```

Switch to the previous branch, without saying the name explicitly:

COPY

```
$ git checkout -
```

Create a new branch from an exiting branch and switch to new branch:

COPY

```
$ git checkout -b <new_branch> <existing_branch>
```

Checkout and create a new branch from existing commit

COPY

```
$ git checkout <commit-hash> -b <new_branch_name>
```

Create a new branch based on your current HEAD:

COPY

```
$ git branch <new-branch>
```

Create a new tracking branch based on a remote branch:

COPY

```
$ git branch --track <new-branch> <remote-branch>
```

Delete a local branch:

COPY

```
$ git branch -d <branch>
```

Rename current branch to new branch name

COPY

```
$ git branch -m <new_branch_name>
```

Force delete a local branch:

You will lose unmerged changes!

COPY

```
$ git branch -D <branch>
```

Mark `HEAD` with a tag:

COPY

```
$ git tag <tag-name>
```

Mark `HEAD` with a tag and open the editor to include a message:

COPY

```
$ git tag -a <tag-name>
```

Mark `HEAD` with a tag that includes a message:

COPY

```
$ git tag <tag-name> -am 'message here'
```

List all tags:

COPY

```
$ git tag
```

List all tags with their messages (tag message or commit message if tag has no message):

COPY

```
$ git tag -n
```

Update and Publish

List all current configured remotes:

COPY

```
$ git remote -v
```

Show information about a remote:

COPY

```
$ git remote show <remote>
```

Add new remote repository, named <remote>:

COPY

```
$ git remote add <remote> <url>
```

Rename a remote repository, from <remote> to <new_remote>:

COPY

```
$ git remote rename <remote> <new_remote>
```

Remove a remote:

COPY

```
$ git remote rm <remote>
```

Note: git remote rm does not delete the remote repository from the server. It simply removes the remote and its references from your local repository.

Download all changes from <remote>, but don't integrate into HEAD:

COPY

```
$ git fetch <remote>
```

Download changes and directly merge/integrate into HEAD:

COPY

```
$ git remote pull <remote> <url>
```

Get all changes from HEAD to local repository:

COPY

```
$ git pull origin master
```

Get all changes from HEAD to local repository without a merge:

COPY

```
$ git pull --rebase <remote> <branch>
```

Publish local changes on a remote:

COPY

```
$ git push remote <remote> <branch>
```

Delete a branch on the remote:

COPY

```
$ git push <remote> :<branch> (since Git v1.5.0)
```

OR

COPY

```
$ git push <remote> --delete <branch> (since Git v1.7.0)
```

Publish your tags:

COPY

```
$ git push --tags
```

Configure the merge tool globally to meld (editor)

COPY

```
$ git config --global merge.tool meld
```

Use your configured merge tool to solve conflicts:

COPY

```
$ git mergetool
```

Merge and Rebase

Merge branch into your current HEAD:

COPY

```
$ git merge <branch>
```

List merged branches

COPY

```
$ git branch --merged
```

Rebase your current HEAD onto <branch>:

Don't rebase published commit!

COPY

```
$ git rebase <branch>
```

Abort a rebase:

COPY

```
$ git rebase --abort
```

Continue a rebase after resolving conflicts:

COPY

```
$ git rebase --continue
```

Use your editor to manually solve conflicts and (after resolving) mark file as resolved:

COPY

```
$ git add <resolved-file>
```

COPY

```
$ git rm <resolved-file>
```

Squashing commits:

COPY

```
$ git rebase -i <commit-just-before-first>
```

Now replace this,

COPY

```
pick <commit_id>
pick <commit_id2>
pick <commit_id3>
```

to this,

COPY

```
pick <commit_id>
squash <commit_id2>
squash <commit_id3>
```

Undo

Discard all local changes in your working directory:

COPY

```
$ git reset --hard HEAD
```

Get all the files out of the staging area(i.e. undo the last `git add`):

COPY

```
$ git reset HEAD
```

Discard local changes in a specific file:

COPY

```
$ git checkout HEAD <file>
```

Revert a commit (by producing a new commit with contrary changes):

COPY

```
$ git revert <commit>
```

Reset your HEAD pointer to a previous commit and discard all changes since then:

COPY

```
$ git reset --hard <commit>
```

Reset your HEAD pointer to a remote branch current state.

COPY

```
$ git reset --hard <remote/branch> e.g., upstream/master, origin/my-feature
```

Reset your HEAD pointer to a previous commit and preserve all changes as unstaged changes:

COPY

```
$ git reset <commit>
```

Reset your HEAD pointer to a previous commit and preserve uncommitted local changes:

COPY

```
$ git reset --keep <commit>
```

Remove files that were accidentally committed before they were added to .gitignore

COPY

```
$ git rm -r --cached .
$ git add .
$ git commit -m "remove xyz file"
```

Thank you for reading

If you like what you read and want to see more, please support me with coffee or a book ;)

Buy me a coffee

Did you find this article valuable?

Support **vishnu chilamakuru** by becoming a sponsor. Any amount is appreciated!

Sponsor Author

See recent sponsors | Learn more about Hashnode Sponsors

#git #cheatsheet #github #2articles1week #version-control

👍 5 ❤️ 4 🦄 4 👏 4 🍺 4 🏆 4 😍 4 💰 2 🎉 2 🚀 1

ARTICLE SERIES

Cheatsheets

1

Kubernetes Cheatsheet

Kubernetes is a portable, extensible, open-source platform for managing containerized workloads and ...

Kubernetes Cheatsheet

2

Docker Cheatsheet

Docker is an open platform for developing, shipping, and running applications. Docker enables you to...

Docker Cheatsheet

••• Show all 5 posts

8

JAVA 8 Cheat Sheet

Index Lambda Expression Collections Method Expressions Streams Optional Lambda Expression (int ...

Java 8 Cheet Sheet

9

Git Cheatsheet

Index Set Up Configuration Files Create Local Changes Search Commit History Move or Rename Branches...

git

Comments

+ Write a comment

© 2021 Vishnu's Blog. See privacy policy and terms.

Publish with Hashnode

Powered by Hashnode - a blogging community for software developers.