Git常用命令速查表

本文列出Git常用命令，点击下图查看大图

如果想及时了
解Spark、Hadoop或者Hbase相关的文章，欢迎关注微信公共帐号：iteblog_hadoop

入门

git init

or

git clone url

配置

```
git config --global color.ui true
git config --global push.default current
git config --global core.editor vim
git config --global user.name "John Doe"
git config --global user.email foo@citrix.com
git config --global diff.tool meld
```

使用本地分支

```
# See the list of all local branches
git branch

# Switch to existing local branch
git checkout branchname

# Checkout current branch into a new branch, named new-branch-name
git checkout -b new-branch-name

# Merge branch-name into the current branch
git merge branchname

# Merge branch without fast forwarding. This is what pull requests do.
# It helps to preserve history of the changes as relavant to that branch
# It's an advanced feature, but try it out with GUI to see the difference
# between the regular merge and merge --no-ff
git merge --no--ff branchname

# Soft branch delete, will complain if the branch is not merged
git branch -d branchname

# Hard branch delete, will not complain about nothing. Like rm -rf in bash
git branch -D branchname
```

更新当前分支

```
# See all commits
git log
```

```
# Pretty commit view, you can customize it as much as you want.
# Just google it □ □
git log --pretty=format:"%h %s" --graph

# See what you worked on in the past week
git log --author='Alex' --after={1.week.ago} --pretty=oneline --abbrev-commit

# See only changes made on this branch (assuming it was branched form master branch)
git log --no-merges master..

# See status of your current git branch.
# Often will have advice on command that you need to run
git status

# Short view of status. Helpful for seeing things at a glance
git status -s

# Add modified file to be commited(aka stage the file)
git add filename

# Add all modified files to be commited(aka stage all files)
git add .

# Add only text files, etc.
git add '*.txt'

# Tell git not to track file anymore
git rm filename

# Record changes to git. Default editor will open for a commit message.
# (Visible via git log)
# Once files are commited, they are history.
git commit

# A short hand for commiting files and writing a commit message via one command
git commit -m 'Some commit message'

# Changing the history □ □ If you want to change your previous commit,
# you can, if you haven't pushed it yet to a remote repo
# Simply make new changes, add them via git add, and run the following command.
# Past commit will be ammended.
git commit --amend
```

高级

```
# Unstage pending changes, the changes will still remain on file system
git reset

# Unstage pending changes, and reset files to pre-commit state. If
git reset --hard HEAD

# Go back to some time in history, on the current branch
git reset tag
git reset <commit-hash>

# Save current changes, without having to commit them to repo
git stash

# And later return those changes
git stash pop

# Return file to it's previous version, if it hasn't been staged yet.
# Otherwise use git reset filename or git reset --hard filename
git checkout filename
```

比较更改

```
# See current changes, that have not been staged yet.
# Good thing to check before running git add
git diff

# See current changes, that have not been commited yet (including staged changes)
git diff HEAD

# Compare current branch to some other branch
git diff branch-name

# Same as diff, but opens changes via difftool that you have configured
# -d tells it to open it in a directory mode, instead of having to open
# each file one at a time.
git difftool -d

# See only changes made in the current branch (compared to master branch)
# Helpful when working on a stand alone branch for a while
git difftool -d master..
```

```
# See only the file names that has changed in current branch
git diff --no-commit-id --name-only --no-merges origin/master...

# Similar to above, but see statistics on what files have changed and how
git diff --stat #Your diff condition
```

使用远程分支

```
# See list of remote repos available. If you did git clone,
# you'll have at least one named "origin"
git remote

# Detailed view of remote repos, with their git urls
git remote -v

# Add a new remote. I.e. origin if it is not set
git remote add origin <https://some-git-remote-url>

# Push current branch to remote branch (usually with the same name)
# called upstream branch
git push

# If a remote branch is not set up as an upstream, you can make it so
# The -u tells Git to remember the parameters
git push -u origin master

# Otherwise you can manually specify remote and branch to use every time
git push origin branchname

# Just like pushing, you can get the latest updates from remote.
# By defaul Git will try to pull from "origin" and upstream branch
git pull

# Or you can tell git to pull a specific branch
git pull origin branchname

# Git pull, is actually a short hand for two command.
# Telling git to first fetch changes from a remote branch
# And then to merge them into current branch
git fetch && git merge origin/remote-branch-name

# If you want to update history of remote branches, you can fetch and purge
```

```
git fetch -p

# To see the list of remote branches
# -a stands for all
git branch -a
```

原文链接：Git Cheat Sheet