|  |  |  |
| --- | --- | --- |
| Git Cheat Sheet | | |
| GIT BASICS | | |
| git init <directory> | 在指定的⽬录下创建⼀个空的git repo。不带参数将在当前⽬录下创建⼀个git repo。 | Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository. |
| git clone <repo> | 克隆⼀个指定repo到本地。指定的repo可以是本地⽂件系统或者由HTTP或SSH指定的远程路径。 | Clone repo located at <repo> onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH. |
| git config user.name <name> | 针对当前repo配置⽤户名。使⽤--global参数将配置全局⽤户名。 | Define author name to be used for all commits in current repo. Devs commonly use --global flag to set config options for current user. |
| git add <directory> | 将指定⽬录的所有修改加⼊到下⼀次 commit中。把<directory>替换成<file>将添加指定⽂件的修改。 | Stage all changes in <directory> for the next commit.Replace <directory> with a <file> to change a specific file. |
| git commit -m "<message>" | 提交暂存区的修改，使⽤指定的<message>作为提交信息，⽽不是打开⽂本编辑器输⼊提交信息。 | Commit the staged snapshot, but instead of launching a text editor, use <message> as the commit message. |
| git status | 显示哪些⽂件已被staged、未被staged以及未跟踪(untracked)。 | List which files are staged, unstaged, and untracked. |
| git log | 以缺省格式显示全部commit历史。更多⾃定义参数请参考后续部分。 | Display the entire commit history using the default format. For customization see additional options. |
| GIT DIFF | | |
| git diff | ⽐较⼯作区和暂存区的修改。 | Show unstaged changes between your index and working directory. |
| git diff HEAD | ⽐较⼯作区和上⼀次commit后的修改。 | Show diﬀerence between working directory and last commit. |
| git diff =--D15A14cached | ⽐较暂存区和上⼀次commit后的修改。 | Show diﬀerence between staged changes and last commit |
| UNDOING CHANGES | | |
| git revert <commit> | 对指定<commit>创建⼀个undo的 commit，并应⽤到当前分⽀。 | Create new commit that undoes all of the changes made in <commit>, then apply it to the current branch. |
| git reset <file> | 将<file>从暂存区移除，但保持⼯作区不变。此操作不会修改⼯作区的任何⽂件。 | Remove <file> from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes. |
| REWRITING GIT HISTORY | | |
| git commit -m <message> --amend | 将当前staged修改合并到最近⼀次的 commit中。 | Replace the last commit with the staged changes and last commit combined. |
| git rebase <base> | 基于<base>对当前分⽀进⾏rebase。<base>可以commit、分⽀名称、tag或相对于HEAD的commit。 | Rebase the current branch onto <base>. <base> can be a commit ID, branch name, a tag, or a relative reference to HEAD. |
| git reflog | 显示本地repo的所有commit⽇志。 | Show a log of changes to the local repository’s HEAD. |
| GIT BRANCHES | | |
| git branch | 显示本地repo的所有分⽀。 | List all of the branches in your repo. |
| git switch -c <branch> | 创建并切换到⼀个新的名为<branch>的分⽀。去掉-c参数将切换到⼀个已有分⽀。 | Create and switch to a new branch named <branch>. Drop the -c flag to switch to an existing branch. |
| git merge <branch> | 将指定<branch>分⽀合并到当前分⽀。 | Merge <branch> into the current branch. |
| REMOTE REPOSITORIES | | |
| git remote add <name> <url> | 添加⼀个新的远程连接。添加后可使⽤ <name>作为指定<url>远程连接的名称。 | Create a new connection to a remote repo. After adding a remote, you can use <name> as a shortcut for <url> in other commands. |
| git fetch <remote> <branch> | 从指定<remote>抓取指定<branch>的所有commit到本地repo。去掉<branch>将抓取远程所有分⽀的修改。 | Fetches a specific <branch>, from the repo. Leave oﬀ <branch> to fetch all remote refs. |
| git pull <remote> | 从指定<remote>抓取所有分⽀的commit并⽴刻合并到本地repo。 | Fetch the specified remote’s copy of current branch and immediately merge it into the local copy. |
| git push <remote> <branch> | 将本地指定<branch>推送到指定远程<remote>。如果远程没有对应的分⽀，将⾃动在远程创建此分⽀。 | Push the branch to <remote>, along with necessary commits and objects. Creates named branch in the remote repo if it doesn’t exist. |
| GIT CONFIG | | |
| git config -- global user.name <name> | 配置当前⽤户名，使⽤--global参数将针对当前系统登录⽤户⽣效。 | Define the author name to be used for all commits by the current user. |
| git config -- global user.email <email> | 配置当前⽤户Email。 | Define the author email to be used for all commits by the current user. |
| git config -- global alias.<alias-name> <git-command> | 配置⼀个git命令的快捷⽅式。例如：配 置”alias.glog log --graph --oneline”使”git glog”相当于”git log --graph --oneline”. | Create shortcut for a Git command. E.g. alias.glog “log -- graph --oneline” will set ”git glog”equivalent to ”git log --graph --oneline. |
| git config -- system core.editor <editor> | 配置⽂本编辑器，例如vi，在必要时⾃动打开此⽂本编辑器。 | Set text editor used by commands for all users on the machine. <editor> arg should be the command that launches the desired editor (e.g., vi). |
| git config -- global --edit | 打开当前⽤户的git全局配置并编辑。 | Open the global configuration file in a text editor for manual editing. |
| GIT LOG | | |
| git log -<limit> | 限制log的显示数量。例如：”git log -5”仅显示最新5条commit。 | Limit number of commits by <limit>. E.g. ”git log -5” will limit to 5 commits. |
| gi log --oneline | 每⾏显示⼀条commit。 | Condense each commit to a single line. |
| git log --author= "<pattern>" | 按提交者名字搜索并显示commit。 | Search for commits by a particular author. |
| git log --grep= "<pattern>" | 按指定内容搜索并显示commit。 | Search for commits with a commit message that matches <pattern>. |
| git log <since>..<until> | 显示指定范围的commit。范围参数可以是 commit ID、分⽀名称、HEAD或任意相对位置。 | Show commits that occur between <since> and <until>. Args can be a commit ID, branch name, HEAD, or any other kind of revision reference. |
| git log --<file> | 仅显示包含指定⽂件修改的commit。 | Only display commits that have the specified file. |
| git log --graph | 使⽤--graph参数显示图形化的branch信息。 | --graph flag draws a text based graph of commits on left side of commit msgs. |
| GIT RESET | | |
| git reset | 移除所有暂存区的修改，但不会修改⼯作区。 | Reset staging area to match most recent commit, but leave the working directory unchanged. |
| git reset --hard | 移除所有暂存区的修改，并强制删除所有⼯作区的修改。 | Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory. |
| git reset <commit> | 将当前分⽀回滚到指定<commit>，清除暂存区的修改，但保持⼯作区状态不变。 | Move the current branch tip backward to <commit>, reset the staging area to match, but leave the working directory alone. |
| git reset --hard <commit> | 将当前分⽀回滚到指定<commit>，清除暂存区的修改，并强制删除所有⼯作区的修改。 | Same as previous, but resets both the staging area & working directory to match. Deletes uncommitted changes, and all commits after <commit>. |
| GIT REBASE | | |
| git rebase -i <base> | 以交互模式对当前分⽀做rebase。 | Interactively rebase current branch onto <base>. Launches editor to enter commands for how each commit will be transferred to the new base. |
| GIT PULL | | |
| git pull --rebase <remote> | 抓取所有远程分⽀，并以rebase模式并⼊本地repo⽽不是merge。 | Fetch the remote’s copy of current branch and rebases it into the local copy. Uses git rebase instead of merge to integrate the branches. |
| GIT PUSH | | |
| git push <remote> --force | 将本地分⽀推送到远程。不要使⽤--force参数，除⾮你完全明⽩此操作的后果。 | Forces the git push even if it results in a non-fast-forward merge. Do not use the --force flag unless you’re absolutely sure you know what you’re doing. |
| git push <remote> --tags | 使⽤push命令并不会⾃动将本地tag推送到远程。加上--tags参数会将所有本地tag推送到远程。 | Tags aren’t automatically pushed when you push a branch or use the --all flag. The --tags flag sends all of your local tags to the remote repo. |