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.
`git 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.