Git cheatsheet

Stefan Hornburg (Racke)

Contents

Commits

Lists and Graphs

List just the hash and the commit message for the ten most recent commits:

```
git log --pretty=oneline -n 10
```

Show a nice graph:

```
git log --oneline --graph --decorate
```

Display and listing files

Display list of files in a commit

Last commit:

```
git show --pretty="format:" --name-only
```

Commit before that:

```
git show --pretty="format:" --name-only HEAD~1
```

Display file contents for a given revision

```
git show ee63c9b61122bfe7e24f8512a3359157ae10066b:components/customer_info
```

Display file contents for in a branch

```
git show master:components/customer_info
```

Log commits for a file that was moved or removed

```
git log -- components/customer_info
```

List untracked files

```
git ls-files --others --exclude-standard
```

List files deleted in the working directory

```
git ls-files -d
```

To restore them:

```
git checkout `git ls-files -d`
```

Display list of files with the last commit date

```
racke@ambas:$ for file in $(find lib/ -type f ); \
  do DT=$(git log -n 1 --pretty=%ci -- $file;); echo "$file:$DT "; done
lib/PerlDance.pm:2016-09-05 15:58:07 +0200
lib/PerlDance/Routes.pm:2016-08-08 14:42:54 +0200
lib/PerlDance/Filter/Wiki.pm:2015-09-15 14:39:36 +0200
lib/PerlDance/Filter/MongerGroups.pm:2016-08-19 11:09:05 +0200
lib/PerlDance/Filter/Markdown.pm:2015-08-06 15:53:25 +0200
```

Merges

Prevent dirty merges

```
$ git config --global pull.ff only
$ git pull
fatal: Not possible to fast-forward, aborting.
$ git pull --rebase
Successfully rebased and updated refs/heads/topic/clean-123
```

Show all commits in a merge

Given that the hashref of the merge is `6d31d51`, you can use the following command to show all commits in that merge:

```
git log 6d31d51^..6d31d51
```

Store empty directory in the Git repository

In case you want to keep a directory for dynamic content in the Git repository, but no files in it, use the following `.gitignore` file:

```
# Explanation about the purpose of the directory
*
!.gitignore
```

Removing and purging files

Remove dynamic files

Sometimes you accidentally add dynamically created files to the Git repository. In this case, you usually want to remove the file from the repository but not from the working directory.

You can do this as follows, e.g for the file `adjtime`:

```
git rm --cached adjtime
echo adjtime >> .gitignore
git add .gitignore
git commit -m "Remove dynamically generated file"
```

Purge untracked files

Please *doublecheck first* that you don't remove files with **valuable content**.
Purge untracked files:

```
git clean -f ...
```

Purge untracked files including directories:

```
git clean -df sql/
```

Reverting commits

Reverting multiple commits

If the commits are consecutive, you can do:

```
git revert  4ea5493d9ac1d6ebcdb0bd29d70275e82ef23467^..d56fe7e13df7ec26c73d744077d7b079b70cc029
```

Where `4ea5493` is the oldest and `d56fe7e` is the newest commit to revert.
Please note that Git will still create one revert commit for each of the commits to be reverted.

Get rid of Git commits

Remove the last three commits (~3):

```
git reset --hard HEAD~3
```

Please note that you **will loose uncommitted** work as well.

"Root" commits

The following command shows you all parentless (root) commits accessible from current branch, useful to find the **initial** commit in the repository:

```
git rev-list --max-parents=0 HEAD
```

It is useful to create an empty commit as "root" commit as the "root" commit cannot be rebased:

```
$ git commit --allow-empty -m "Initial commit."
[master (root-commit) 99dbf81] Initial commit.
```

Signing commits/tags

Show signatures

```
$ git log --show-signature
```

Force signing

```
$ git config --global commit.gpgsign true
```

Multiple keys

If you have multiple keys with the same email address, you can configure the correct one as follows:

```
git config --global user.signingkey FA2720F8
```

Signing on remote hosts

Add extra socket to your GPG agent configuration file (`~/.gnupg/gpg-agent.conf`):

```
extra-socket /path/to/extra-socket
```

Stage

Stage changes for the next commit:

```
git add README.md
```

Display staged files:

```
git diff --name-only --cached
```

Unstage changes:

```
git reset HEAD
```

Tags

Show all tags and corresponding messages

```
git tag -l '*' -n 1
```

Display a tag

```
$ git show 6.2.70
tag 6.2.70
Tagger: Stefan Hornburg (Racke) <racke@linuxia.de>
Date:   Thu Nov 10 12:41:43 2022 +0100

[-release]Preparing version 6.2.70
-----BEGIN PGP SIGNATURE-----
...
```

Add a signed tag

```
git tag -a -s -m "Release 0.21" v0.21
```

Push tags to remote repository

Tags are not automatically pushed with `git push`:

```
git push --tags
```

Diffs & Logs

Show all files changed in a branch

```
$ git checkout mybranch
$ git diff --name-only master
...
```

Compare releases

```
git diff 2.0.0 2.0.3
```

The diff can be restricted to changes in a file or a directory:

```
git diff 2.0.0:tasks 2.0.3:tasks
```

Commits

Diff for the last three commits:

```
git diff HEAD~3 HEAD
```

Ignore whitespace

Options for a diff ignoring whitespaces:

–**ignore-space-at-eol** Changes at end of line (e.g. LF to CRLF)

-b As above plus collapses multi whitespaces into one

-w Ignores all whitespace changes (usually not what you want ...)

Reverse diff

```
git diff -R
```

This can be useful, e.g to display deleted whitespace.

Formatting

Show the short hash:

```
git log --abbrev-commit
```

Remotes

Display remotes

Short form (just the names):

```
~# git remote
origin
upstream
```

Long form (includes URL):

```
~# git remote -v
origin    git@github.com:racke/dancer2.git (fetch)
origin    git@github.com:racke/dancer2.git (push)
upstream    git@github.com:PerlDancer/Dancer2.git (fetch)
upstream    git@github.com:PerlDancer/Dancer2.git (push)
```

Add remote repository

```
~# git remote add origin git@github.com:racke/dancer2.git
```

Rename remote repository

```
~# git remote rename origin upstream
```

Change URL for remote repository

```
~# git remote set-url upstream git@github.com:racke/dancer2.git
```

Reset after remote rebase

```
~# git reset --hard origin/topic/returns-user-control-3964
```

Please note that you **will loose uncommitted** changes.

Checkout

Checkout a new branch from a tag

Creates a new branch *topic/v2.1-fixes* from tag *v2.1* and switches to it:

```
$ git checkout -b topic/v2.1-fixes v2.1
Switched to a new branch 'topic/v2.1-fixes'
```

Stash

Add to stash

With message:

```
$ git stash save "Old cruft."
```

List stash entries

```
$ git stash list
```

Show list with creation date of the entries:

```
$ git stash list --date=local
```

Show stash contents

```
$ git stash show -p stash@{1}
```

Include untracked files

```
git stash -u
```

```
git stash save -u "Stash away experiments"
```

Stash specific file(s)

```
git stash push default/web_tt2/subscriber_table.tt2
```

Stash a directory including untracked files:

```
git stash push -u roles/sympa
```

Retrieve stash

```
$ git stash pop stash@{1}
On branch topic/header/from-as-envelope-sender
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

    modified:   src/lib/Sympa/Spindle/ToList.pm
```

```
no changes added to commit (use "git add" and/or "git commit -a")
Dropped stash@{1} (9579fe0d020a8e63733dff961fc3bd0865100959)
```

Pick a file from the stash

```
git checkout stash@{0} -- src/lib/Sympa/Bulk.pm
```

Branches

Tracking branches (set upstream)

Track remote branch *pr-fix-typo*:

```
git branch -u origin/pr-fix-typo
```

This effectively sets the upstream for the current branch to *origin/pr-fix-typo*.

Compare files between branches

```
git diff master:README pr/fix-typo:README
```

This also works for directories.

Files added in a branch

This lists all files added in the branch *topic/transaction-status* and not present in the *master* branch.

```
git diff --name-only --diff-filter=A master topic/transaction-status
```

Similar for modified files only:

```
git diff --name-only --diff-filter=M master topic/transaction-status
```

Find branch for a commit

Local repository:

```
$ git branch --contains c0cc8e3e185e1de308b0c86b01f8fccc98dd4485
  topic/imap-search-header-2020-03-12
```

Remote repositories:

```
$ git branch -r --contains c0cc8e3e185e1de308b0c86b01f8fccc98dd4485
  github/topic/imap-search-header-2020-03-12
  origin/topic/imap-search-header-2020-03-12
```

First commit for a branch

Show starting point of the branch in relationship to the *main* branch.

```
git log --reverse main..topic/pr-foo-bar
```

Show commits in a branch not merged into master

```
git cherry -v master feature/mobile
```

You can also compare to a remote branch:

```
git cherry -v master origin/feature/mobile
```

If there are no unmerged commits left, you can delete the local and remote branch with the following commands:

```
git branch -d feature/mobile
```

```
git push origin :feature/mobile
```

Merge base

The merge base is the common commit where the current branch diverged from. You can determine this commit with the *merge-base* command:

```
git merge-base master topic/org-addons
```

It is quite a common case to only see the logs or the diff of the changes in your current branch since they diverged.

But the following command will also consider the changes in master since the diverge:

```
git diff master..topic/org-addons
```

Instead you can use

```
git diff $(git merge-base master topic/org-addons)..topic/org-addons
```

or the shorthand (**three** dots):

```
git diff master...topic/org-addons
```

The same applies of course to the *log* command as well.

Just show the name of the current branch

```
git symbolic-ref --short HEAD
```

Rename branch

This renames the current branch. It doesn't affect the remote branch.

```
git branch -m topic/newname
```

To rename a branch other than the current one:

```
git branch -m topic/oldname topic/newname
```

In order to rename the remote branch we are doing two steps.

First we delete the old remote branch and push the new local branch:

```
git push origin :topic/oldname topic/newname
```

Second we update the remote tracking information:

```
git branch -u origin/topic/newname
```

Move branch

Move from `master` to `main`:

```
git branch -M main
```

Merged and unmerged branches

Show merged branches:

```
git branch --merged
```

Show remote merged branches:

```
git branch -r --merged
```

Show unmerged branches:

```
git branch --no-merged
```

Sorted by age

This sorts local branches by date of the latest commit and displays the commit date and the branch name:

```
for C in $(git for-each-ref --sort=committerdate refs/heads --format='%(refname)');
do
   git show -s --format="%ci $C" "$C";
done
```

Same for remote branches from *origin*:

```
git fetch origin

for C in $(git for-each-ref --sort=committerdate refs/remotes/origin --format='%(refname)');
do
   git show -s --format="%ci $C" "$C";
done
```

Configuration

Name and email

```
git config --global user.name "Stefan Hornburg (Racke)"
git config --global user.email "racke@linuxia.de"
```

Splitting a subdirectory out into a new repository

Install git-filter-repo

For example:

```
$ sudo apt install git-filter-repo
```

Clone into new repository

Clone repository and move into its directory:

```
$ git clone ansible ansible-role-postgresql
$ cd ansible-role-postgresql
```

Filter subdirectory to the root of the new subdirectory

```
$ git filter-repo --subdirectory-filter roles/postgresql
```

Remove obsolete branches (optional)

You probably doesn't need all the branches of the bigger repository. In this case you can remove all branches besides your **main** branch as follows:

```
$ git branch | grep -v main | xargs git branch -D
```

Shell

It is quite helpful to show Git related information in your shell prompt.

If you are using the `~/.bashrc` from the `/etc/skel` directory on Debian your shell prompts are configured as:

```
if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w\$ '
fi
```

So a prompt could look like:

```
racke@linuxia:~/provisioning/ansible$
```

After we modified the prompts to include the Git branch information, the definitions look like:

```
if [ "$color_prompt" = yes ]; then
    PS1='${debian_chroot:+($debian_chroot)}\[\033[01;32m\]\u@\h\[\033[00m\]:\[\033[01;34m\]\w\[\033[
else
    PS1='${debian_chroot:+($debian_chroot)}\u@\h:\w$(__git_ps1 " (%s)")\$ '
fi
```

After reloading your `~/.bashrc` the prompt will change to include the Git branch information for a directory (if any):

```
racke@linuxia:~/provisioning/ansible (master)$
```

We can also add further information with setting the *GIT_PS1_SHOWDIRTYSTATE* environment variable.

So if you have changes in this working directory, the prompt shows an asterisk after the branch name:

```
racke@linuxia:~/provisioning/ansible (master *)$
```

Linuxia Wiki

Stefan Hornburg (Racke)

Git cheatsheet

wiki.linuxia.de