railsware blog

visit our site

Git housekeeping tutorial: clean-up outdated branches in local and remote repositories

In Development

by Sergii Boiko by August 11, 2014

After working with branch per feature for a while any Git-repository becomes a mess of outdated and not finished branches. To deal with this issue, we need to clean-up three kinds of branches:

- Local branches – our day-to-day working branches
- References to remote branches – aka origin/branch-name items
- Actual remote branches – branches on remote server(e.g.: github, bitbucket, gitorius)

In this tutorial we suppose, that “master” – is a default branch, where everything is merged(for someone it could be “develop”, or “release” branch), and “origin” – it’s a name of remote. If you use different names, just change them in appropriate commands.

Local branches

At first, list all local branches:

```
$ git branch
```

We need to know what branches are already merged in “master” and can be easily removed:

```
$ git checkout master
$ git branch --merged
```

Now, remove all outdated branches with:

```
$ git branch -d old-merged-feature
```

Next, decide what to do with not merged branches:

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

If some of them is just abandoned stuff that you don’t need anymore, remove it with “-D” option:

```
$ git branch -D old-abandoned-feature
```

References to remote branches

After each `git pull` or `git fetch` command Git creates references to remote branches in local repository, but doesn’t clean up stale references.

List referenced remote branches:

```
$ git branch -r
```

Clean-up outdated references:

```
$ git remote prune origin
```

Tip

Update repository with:

```
$ git fetch -p
```

and Git automatically prunes all stale references.

Remote branches

Usually, remote repository is a big garbage heap of stale branches, if there is no responsible housekeeping person.

After previous `git remote prune origin` we should have synched list of remote branches.

At first, we can find branches which are already merged in “master”:

```
$ git checkout master
$ git branch -r --merged
```

But this command does not provide much information. What if this branch is merged, but still used for feature development. Would be cool to know last commit date and author.

This magic snippet provides all required information:

```
$ for branch in `git branch -r --merged | grep -v HEAD`; do echo -e `git show --format="%ci %cr %a
n" $branch | head -n 1` \\t$branch; done | sort -r
```

Now, you can delete own remote branches, and ask other authors to clean-up theirs:

```
$ git push origin --delete branch-name
```

Similar snippet for not merged branches:

```
$ for branch in `git branch -r --no-merged | grep -v HEAD`; do echo -e `git show --format="%ci %cr
 %an" $branch | head -n 1` \\t$branch; done | sort -r
```

This list should be reviewed more thoroughly to avoid losing important commits.

Tip for Github users

After the last Github update, **Branches** page is divided into “Your branches”, “Active branches” and “Stale branches”, and it shows same information as previous commands.

Bonus snippets

Usually, it’s simple to remove local and appropriate remote branches at once.

This snippet shows only local merged branches, which have appropriate remote merged branches:

```
$ comm -12  <(git branch --merged|awk '{print($1)}') <(git branch -r --merged|awk '{print($1)}'|awk
 -F \/ '{print($2)}')
```

More hardcore snippet with date and author information:

```
$ for branch in `comm -12  <(git branch --merged|awk '{print($1)}') <(git branch -r --merged|awk
'{print($1)}'|awk -F \/ '{print($2)}')`;do echo -e `git show --format="%ci %cr %an" $branch | head
-n 1` \\t$branch; done | sort -r
```

Same snippets for not merged branches:

```
$ comm -12  <(git branch --no-merged|awk '{print($1)}') <(git branch -r --no-merged|awk '{print
($1)}'|awk -F \/ '{print($2)}')
```

```
$ for branch in `comm -12  <(git branch --no-merged|awk '{print($1)}') <(git branch -r --no-
merged|awk '{print($1)}'|awk -F \/ '{print($2)}')`; do echo -e `git show --format="%ci %cr %an" $br
anch | head -n 1` \\t$branch; done | sort -r
```

Moving stuff into .gitconfig

It’s hard to remember such code, so the best way is to create shell scripts and put them in local “bin” folder. Note that these snippets work only in bash(and zsh). For example, let’s put first snippet into “bin/git-both-merged” file:

```
1 #!/usr/bin/env bash
2
3 comm -12  <(git branch --merged|awk '{print($1)}') <(git branch -r --merged|awk '{print($1)}'|a
  wk -F \/ '{print($2)}')
```

Don't forget to make it executable(`chmod 755 git-both-merged`), and you can also make a git alias for this script. Put next line in .gitconfig:

```
[alias]
  # ... other aliases
  both-merged = !git-both-merged
```

Now you can call it via git command:

```
$ git both-merged
```

In the same vein you can create git aliases for all snippets from this article.

That's it, folks. Now, it's time to clean-up all stale stuff!

Sergii Boiko

Sergii Boiko is a software engineer at Railsware*. His main scopes of interest are language design, scaling and troubleshooting. He likes to collect different pieces of knowledge about debugging and apply them for resolving pesky production issues.

More blog posts by Sergii Boiko

* Railsware is a premium software development consulting company, focused on delivering great web and mobile applications. Learn more about us.

Spindle • a year ago

In addition to local merged branches, you can also run

```
$ git branch --merged | grep -v '\*\|master\|develop' | xargs -n 1 git branch -d
```

to delete all the local merged branches as you saw in

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

6 ^ v • Reply • Share ›

Grégory Joseph • 2 years ago

If `git-both-merged` is in your PATH, you shouldn't need an alias.

2 ^ v • Reply • Share ›

Anil • 2 years ago

Nice article . is there any way we can delete remote branches based the particular date range. Ex : delete all the remote branches which are merged to master & not modified in last 4 months.

1 ^ v • Reply • Share ›

Vahagn Grigoryan ➚ Anil • a year ago

thanks for useful article Sergii Boiko.
I would also be interested in answer to this question, it would be very useful.

^ v • Reply • Share ›

Pablo Ezequiel Leone Signetti • 10 months ago

This is a very valuable knowledge! Thank you for sharing.

^ v • Reply • Share ›

Christine Shaffer • a year ago

Thank you so much for this! Very clear & concise

^ v • Reply • Share ›

Tom Leo • a year ago

Very useful thanks!

^ v • Reply • Share ›

James • a year ago

What if I want to delete all branches that [are merged] + [are older than 30 days] + [do not start with "origin/rel"]?

^ v • Reply • Share ›

florian • a year ago

Thanks! Very helpful!

^ v • Reply • Share ›

John Steill ➚ florian • a year ago

Ditto. Exactly what I was looking for. Thanks mate.

^ v • Reply • Share ›

Mirko Akov • a year ago

Did not know about `git fetch -p`

Reply • Share ›

Pablo • 2 years ago

chmod +x git-both-merged is better, it doesn't touch r and w privileges.

Reply • Share ›

ari gold • 2 years ago

From the git man page
"
--merged [<commit>]
Only list branches whose tips are reachable from the specified commit (HEAD if not specified).
"
which implies that if you're working on a branch that hasn't diverged from develop - say a new one - that it will show up in `git branch --merged`. It's not really "merged" but it's tip is reachable from the HEAD of where ever you run the command from (in this article `master`, with me `develop`). It's not really a big deal to delete a branch that is equivalent to `master` (or `develop`) but it's good to know why it's showing.

Speaking of articles, thanks for the great one!

Reply • Share ›

Artur Termenji • 3 years ago

Try to use 'git branch -avv'. It shows all branches in one command (local + remote), and shows HEAD commit with hash and message for each branch.
Pretty useful command to alias it for something like 'gb'

Reply • Share ›

Alex Mishyn • 3 years ago

Is there any command to sync(smth like remove local branch is remote doesn't exists) local branches with remote ?

Reply • Share ›

Dmitry Pliska ➚ Alex Mishyn • 3 years ago

git fetch --prune

Reply • Share ›

ayanko ➚ Dmitry Pliska • 2 years ago

DRY:
git config --global fetch.prune true

3 Reply • Share ›

Search for: Search

Choose a taste

- Android 1
- Business 13
- Conferences 11
- Design 14
- Development 105
- Development process 16
- Front End 23
- Google 4
- Hackathon 1
- iOS 9
- JIRA 1
- Mac OSX 9
- Mailtrap 8
- Management 16
- Marketing 4
- Mobile 9
- Monitoring 3
- New Relic 1
- Office 5
- Operations 2
- Partners 1
- Performance 8
- Product development 8
- Products 17
- Random 48
- Scripts 6
- Swift 2
- Testing 8
- Thoughtful leadership 2

Subscribe to our mailing list

email address

RSS feed

We're always ready to help!

CONTACT US

Most recent posts

- Jun 23, 2017Famous Web Apps Built with Ruby on Rails
- Jun 20, 2017Hideable React component using HOC
- Jun 5, 2017How Airtable can help you in real life: Railsware example
- May 25, 2017Kotlin announced as a first-class citizen in Android. Why does it matter?

- May 12, 2017Hive53 Calendly Swarm meetup on building global business from scratch

Most popular posts

193,358 viewsGit housekeeping tutorial: clean-up outdated branches in local and remote repositories
64,451 viewsAPI with Ruby on Rails: useful tricks
41,282 viewsCreation of pure Swift module
30,137 viewsPostgreSQL most useful extensions
29,222 viewsUseful Google Spreadsheet Script Triggers

Signup for our weekly newsletter

Email address

Want to get more of Railsware blog?

RSS Feed

We're always ready to help!

Contact us

railsware blog

Products

- Booster
- Piro
- Mailtrap

Experience

- Open Source
- Blog

Contact

+1 (646) 397 4918

Social

Railsware is a leading Ruby on Rails company which provides contracting and consulting services around the world. We are among the premium ruby on rails companies on the US market and our ROR development company is the exeperts at Ruby, Rails, HTML 5, and CSS3.