# How do I force "git pull" to overwrite local files?

Asked 14 years, 3 months ago Modified 18 days ago Viewed 8.0m times

9460

How do I force an overwrite of local files on a `git pull`? My local repository contains a file of the same filename as on the server.

> error: Untracked working tree file 'example.txt' would be overwritten by merge

git version-control overwrite git-pull git-fetch

Share Improve this question Follow

edited Jul 18, 2022 at 18:42
John Smith
7,253 6 49 61

asked Jul 14, 2009 at 14:58
Jakub Troszok
99.7k 11 42 53

45 anyone reading this who thinks they might lose files, I've been in this position and found Sublime Text's buffer has saved me - if I'm working on something, then accidentally delete everything by trying to solve a similar problem to this or by using an answer on this question and have had the files open in Sublime (which there's a good chance of) then the files will still be there is Sublime, either just there, or in the undo history – Toni Leigh Jan 20, 2016 at 8:51

296 `git reset --hard origin/branch_to_overwrite` – Andrew Atkinson Mar 22, 2016 at 8:37

4 basically, only do a pull from develop after the initial checkout -b. do your work, then push back in. – ldgorman Aug 22, 2018 at 9:09

4 Short answer: delete and re-create branch. 1. Delete branch: `git branch <branch> -D` 2. Reset to a commit before the conflict: `git reset <commit> --hard` 3. Re-create the branch: `git branch <branch>` 4. Set tracking to the server: `git --set-upstream-`

Nino Filiu Sep 24, 2018 at 8:54

`fig core.autocrlf false; git`
oe Jan 17, 2019 at 2:46

Highest score (default)

**Your privacy**

By clicking “Accept all cookies”, you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies Necessary cookies only

Customize settings

13026

## ⚠ Warning:

Any uncommitted local change to tracked files will be lost, **even if staged**.

But any local file that's *not* tracked by Git will *not* be affected.

First, update all `origin/<branch>` refs to latest:

```
git fetch --all
```

Backup your current branch (e.g. `master` ):

```
git branch backup-master
```

Jump to the latest commit on `origin/master` and checkout those files:

```
git reset --hard origin/master
```

## Explanation:

`git fetch` downloads the latest from remote without trying to merge or rebase anything.

`git reset` resets the master branch to what you just fetched. The `--hard` option changes all the files in your working tree to match the files in `origin/master` .

## Maintain current local commits

nt local commits by creating a

anch-to-save-current-commits .

, will be lost. Make sure to `stash`
llowing:

```
git stash
```

And later (after `git reset`), reapply these uncommitted changes:

```
git stash pop
```

> Which may create merge conflicts.

Share Improve this answer Follow

edited May 1 at 18:27
Top-Master
7,730 5 40 73

answered Jan 17, 2012 at 0:02
RNA
148k 15 52 72

34 Watch out! If you have local unpushed commits this will remove them from your branch! This solution keeps untracked files not in the repository intact, but overwrites everything else. – Matthijs P May 17, 2012 at 8:18

562 It's a popular question, so I'd like to clarify on the top comment here. I just executed commands as described in this answer and it hasn't removed ALL the local files. Only the remotely tracked files were overwritten, and every local file that has been here was left untouched. – Red Nov 22, 2012 at 10:38

45 in case you're pulling from a repo that has its remote branch name different from "master", use `git reset --hard origin/branch-name` – Abbas Gadhia Dec 17, 2013 at 11:17

241 Given the amount of upvotes to this question and answer, I think that git should incorporate a command like `git pull -f` – Sophivorus Aug 26, 2014 at 1:33

23 Commits that weren't pushes before the hard reset can be recovered using `git reflog`, which list all commits, also those without a base. Until you cleanup your local copy using `git gc`, then all is lost – Koen. Feb 10, 2015 at 22:24

This will remove all uncommitted changes, even if staged,

e affected.

**Your privacy**

By clicking “Accept all cookies”, you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies Necessary cookies only

Customize settings

answered May 9, 2010 at 19:45
Travis Reeder
38.9k 12 88 87

33 I've done this and some local files that were no longer in repo were left on the disk.
– Piotr Owsiak Apr 8, 2011 at 16:00

41 I do not think that this is correct. the above will perform a merge, not overwrite which was requested in the question: "How to force git to overwrite them?" I do not have the answer, I am currently looking for it.. at the moment I switch to the branch with with the code that I want to keep "git checkout BranchWithCodeToKeep", then do "git branch -D BranchToOverwrite" and then finally "git checkout -b BranchToOverwrite". you will now have the exact code from BranchWithCodeToKeep on the branch BranchToOverwrite without having to perform a merge.
– felbus Jul 13, 2011 at 10:11

306 instead of merging using 'git pull', try git fetch --all followed by 'git reset --hard origin/master'
– Lloyd Moore Feb 21, 2012 at 14:56

10 yep, the @lloydmoore solution worked for me. Could do with being an answer rather than just a comment. – Max Williams Nov 19, 2012 at 9:54

5 This will reset the current changes back to the last branch commit pulled. Then git pull merges the changes from the latest branch. This did exactly what I wanted it to do.. Thanks!
– Codeversed Dec 5, 2014 at 17:42

**WARNING: `git clean` deletes all your untracked files/directories and can't be undone.**

592

Sometimes just `clean -f` does not help. In case you have untracked DIRECTORIES, -d option also needed:

```
# WARNING: this can't be undone!

git reset --hard HEAD
git clean -f -d
git pull
```

**WARNING: `git clean` deletes all your untracked files/directories and can't be undone.**

Consider using `-n` ( `--dry-run` ) flag first. This will show you what will be deleted without actually deleting anything:

```
git clean -n -f -d
```

Example output:

```
Would remove untracked-file-1.txt
Would remove untracked-file-2.txt
Would remove untracked/folder
...
```

Share Improve this answer Follow

edited Aug 17, 2018 at 19:32

answered Mar 19, 2011 at 9:10
Davic Avsaj David Avsajanishvili
7,738 2 22 24

and avoid deleting untracked files that

't really want to throw away the content. ne files. @Lauri, this should not have ad the essence of scenario description

:lean everything. – earthmeLon Jun 23,

ing Fingers Jun 13, 2013 at 6:58

The `-x` ignores .gitignore. Typically .2, 2015 at 18:28

487

Like Hedgehog I think the answers are terrible. But though Hedgehog's answer might be better, I don't think it is as elegant as it could be. The way I found to do this is by using `fetch` and `merge` with a defined strategy. Which should make it so that your local changes are preserved as long as they are not one of the files that you are trying to force an overwrite with.

## First do a commit of your changes

```
git add *
git commit -a -m "local file server commit message"
```

## Then fetch the changes and overwrite if there is a conflict

```
git fetch origin master
git merge -s recursive -X theirs origin/master
```

`-X` is an option name, and `theirs` is the value for that option. You're choosing to use `their` changes (the other option is `ours` changes) if there is a conflict.

Share Improve this answer Follow

edited Feb 24, 2021 at 14:23
TheTechRobo the Nerd
1,266 15 28

answered Apr 11, 2012 at 20:13
Richard
5,612 1 19 22

75 This is the best answer I've seen so far. I haven't tried it, but unlike other answers, this doesn't attempt to nuke all your untracked files, which is very dangerous for obvious reasons. – huyz May 7, 2012 at 9:36

8 Ditto - this worked for me when doing a very large merge (GitHub pull request) where I just wanted to accept it all on top of what I had. Good answer! In my case the last two commands were: 1) `-X theirs other-repo/master`

d not your local ones, correct?

se on detached head. I switched back `igin/master` – petergus Mar 11,

about it. after all im just using it
ing for a force overwrite option, at

419

Instead of doing:

```
git fetch --all
git reset --hard origin/master
```

I'd advise doing the following:

```
git fetch origin master
git reset --hard origin/master
```

No need to fetch all remotes and branches if you're going to reset to the origin/master branch right?

Share Improve this answer Follow edited Sep 14, 2016 at 9:46 answered Apr 26, 2013 at 13:48 Johanneke **5,473** 4 20 33

3 Your answer is just what you needed for your rep. I must ask, does this also remove all untracked files? – Nicolas De Jay Jan 7, 2014 at 6:38

6 Yeah, most of my rep is coming from here :) This will also remove all untracked files. Something I had forgotten and was painfully reminded of just 2 days ago... – Johanneke Jan 9, 2014 at 12:01

1 See the comments on this other answer: stackoverflow.com/a/8888015/2151700 – Johanneke Jan 9, 2014 at 12:02

This did not remove my untracked files; which is actually what I'd expect. Is there a reason it might for some people and not for others? – Ada Richards Apr 19, 2016 at 15:27

1 This is exactly what I needed: something that overwrites untracked files that exist in the remote, and leaves everything else intact. – Ledazinha Dec 20, 2017 at 22:37

ısual `git pull` ...

answered Jul 14, 2009 at 15:16 Jakub Troszok **99.7k** 11 42 53

31

4 I tried using "git clean" to solve the same issue, but it did not resolve it. git status says "Your branch and 'origin/master' have diverged, # and have 2 and 9 different commit(s) each, respectively." and git pull says something similar to what you have above. – slacy Sep 24, 2009 at 4:25

48 git clean is a rather blunt instrument, and could throw away a lot of things that you may want to keep. Better to remove or rename the files that git is complaining about until the pull succeeds. – Neil Mayhew Jul 2, 2010 at 13:21

2 I do not think this works in general. Isn't there a way to do basically a git clone remote via a forced git pull? – mathtick Nov 29, 2010 at 18:30

19 @mathick: `git fetch origin && git reset --hard origin/master` – Arrowmaster Feb 23, 2011 at 4:24

3 Is `git clean` the best answer here? Seems like removing files isn't necessarily what the OP wants. They asked for 'an overwrite of local files' not deletion. – JohnAllen Mar 4, 2014 at 8:28

133

**Warning, doing this will permanently delete your files if you have any directory/* entries in your gitignore file.**

Some answers seem to be terrible. Terrible in the sense of what happened to @Lauri by following David Avsajanishvili suggestion.

Rather (git > v1.7.6):

```
git stash --include-untracked
git pull
```

Later you can clean the stash history.

Manually, one-by-one:

```
$ git stash list
stash@{0}: WIP on <branch>: ...
stash@{1}: WIP on <branch>: ...

$ git stash drop stash@{0}
$ git stash drop stash@{1}
```

Brutally, all-at-once:

```
$ git stash clear
```

Of course if you want to go back to what you stashed:

```
$ git stash list
...
```

answered Feb 11, 2012 at 23:00
Hedgehog
5,497 4 36 43

2 No I don't think so. Stashing just moves uncommitted files out of the way. The above also moves (stashes) files that git does not track. This prevents files that have been added to the remote, which have not yet pulled down to your machine - but which you have created (!) - to be pulled down. All without destroying the uncommitted work. Hope that makes sense? – Hedgehog Mar 20, 2012 at 23:54

3 If you don't have 1.7.6, you can mimic `--include-untracked` simply by temporarily `git add` -ing your entire repo, then immediately stashing it. – nategood May 1, 2012 at 22:48

3 I agree with Hedgehog. If you do the popular answers here, you are more than likely going to find you've inadvertently killed a lot of stuff that you didn't really want to lose. – AfroRick Jan 31, 2013 at 21:28

1 I had other untracked files--besides the one the merge/pull wanted to overwrite, so this solution worked best. `git stash apply` brought back all my untracked files with the exception (rightly) of the ones that the merge had already created: "already exists, no checkout." Worked perfectly. – BigBlueHat Apr 25, 2013 at 4:55

2 This is the cleanest answer, and should be the accepted one. To save some typing you can use the short form: `git stash -u` . – ccpizza Mar 23, 2017 at 8:30

116

You might find this command helpful to throw away local changes:

```
git checkout <your-branch> -f
```

And then do a cleanup (removes untracked files from the working tree):

```
git clean -f
```

If you want to remove untracked directories in addition to untracked files:

answered Aug 5, 2010 at 18:06

Vishal
**20k** 23 80 93

31

t really want to throw away the content.
e files. See my suggestion.

ill saved me from the frustration of git
When git reset --hard HEAD does not
elpful. Thanks a bunch. – Kellindil Jan

104

Instead of merging with `git pull`, try this:

```
git fetch --all
```

followed by:

```
git reset --hard origin/master.
```

Share Improve this answer Follow

edited Mar 21, 2018 at 7:21
sbarb
95 1 2 9

answered Nov 22, 2012 at 10:56
Lloyd Moore
3,127 1 32 32

this wont work in scripts cause you have to know the branch name. Look at my solution for a generic way – warch Nov 25, 2021 at 9:02

76

The only thing that worked for me was:

```
git reset --hard HEAD~5
```

This will take you back five commits and then with

```
git pull
```

I found that by looking up [how to undo a Git merge](#).

Share Improve this answer Follow

edited May 23, 2017 at 10:31
Community Bot
1 1

answered May 5, 2011 at 21:53
Chris BIllante
801 6 3

ed my branch to the origin repo and
te repo.. – jwfrench May 7, 2014 at

ctive. Because some conflicts may
ke sure no conflicts with remote code.

68

The problem with all these solutions is that they are all either too complex or, an even bigger problem, is that they remove all untracked files from the webserver, which we don't want since there are always needed configuration files which are on the server and not in the Git repository.

Here is the cleanest solution which we are using:

```
# Fetch the newest code
git fetch

# Delete all files which are being added, so there
# are no conflicts with untracked files
for file in `git diff HEAD..origin/master --name-status | awk '/^A/ {print $2}'`
do
    rm -f -- "$file"
done

# Checkout all files which were locally modified
for file in `git diff --name-status | awk '/^[CDMRTUX]/ {print $2}'`
do
    git checkout -- "$file"
done

# Finally pull all the changes
# (you could merge as well e.g. 'merge origin/master')
git pull
```

- The first command fetches the newest data.
- The second command checks if there are any files that are being added to the repository and deletes those untracked files from the local repository which would cause conflicts.
- The third command checks-out all the files which were locally modified.
- Finally, we do a pull to update to the newest version, but this time without any
  ) don't exist anymore and all the
  repository.

answered Nov 5, 2012 at 23:32

Strahinja Kustudic
**4,205** 1 25 18

in your note) instead of "git pull" will be
jit repo. – Josh May 6, 2013 at 6:21

and probably even safer. Since if
; of this script (which is not likely to
son I put `pull` in there is because
me other branch and I wanted the
?:25

`.gitignore` . – Sebi Nov 21, 2017 at

65

First of all, try the standard way:

```
git reset HEAD --hard # To remove all not committed changes!
git clean -fd         # To remove all untracked (non-git) files and folders!
```

**Warning**: Above commands can results in data/files loss only if you don't have them committed! If you're not sure, make the backup first of your whole repository folder.

Then pull it again.

If above won't help and you don't care about your untracked files/directories (make the backup first just in case), try the following simple steps:

```
cd your_git_repo  # where 'your_git_repo' is your git repository folder
rm -rfv *         # WARNING: only run inside your git repository!
git pull          # pull the sources again
```

This will REMOVE all git files (excempt `.git/` dir, where you have all commits) and pull it again.

Why `git reset HEAD --hard` could fail in some cases?

1. Custom rules in `.gitattributes file`

   Having `eol=lf` rule in .gitattributes could cause git to modify some file changes by converting CRLF line-endings into LF in some text files.

   If that's the case, you've to commit these CRLF/LF changes (by reviewing them in `git status`), or try: `git config core.autcrlf false` to temporary ignore them.

rt permission attributes. In
lac ( `ext3` / `hfs+` ) and another one

stems, so the one which doesn't
ermissions on system which
ter how `--hard` you try, git always

answered Oct 26, 2012 at 9:17

kenorb
157k 88 680 745

62

## Bonus:

In speaking of pull/fetch/merge in the previous answers, I would like to share an interesting and productive trick,

```
git pull --rebase
```

This above command is the most useful command in my Git life which saved a lot of time.

Before pushing your newly commit to server, try this command and it will automatically synchronise the latest server changes (with a fetch + merge) and will place your commit at the top in the Git log. There isn't any need to worry about manual pull/merge.

Find details in *What does "git pull --rebase" do?*.

Share Improve this answer Follow

edited Jun 20, 2020 at 9:12
Community Bot
1 1

answered Dec 23, 2015 at 15:41
Sazzad Hissain Khan
38.1k 34 194 260

7 In short: `git pull -r` . – kenorb Oct 15, 2019 at 9:29

In my case, before doing that, I had to 1) `git add -A` , 2) `git commit -m` 3) and finally `git pull rebase` . Thank you. – Pathros Feb 3, 2021 at 20:10

this removes my committed changes. what am I doing wrong? – Nikhil S Mar 2 at 19:20

I had the same problem. No one gave me this solution, but it worked for me.

I solved it by:

**Your privacy**

By clicking “Accept all cookies”, you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies Necessary cookies only

Customize settings

answered Jan 12, 2011 at 23:58
John John Pichler
4,457 8 43 72

happens often that only reset and

t 11:28

50

Here is a **generic solution** if you do not always want to paste the branch name or you want to **automate this within a script**

```
git fetch
git reset --keep origin/$(git rev-parse --abbrev-ref HEAD)
```

If you want to reset your local changes too:

```
git fetch
git reset --hard origin/$(git rev-parse --abbrev-ref HEAD)
```

You also could add a bash alias using this command:

```
alias gplf='git fetch && echo "HEAD was at $(git rev-parse --short HEAD)" && git
reset --hard origin/$(git rev-parse --abbrev-ref HEAD)'
```

Share Improve this answer Follow

edited Dec 1, 2021 at 12:28

answered May 25, 2018 at 6:31
warch
2,427 2 27 44

2 If you find yourself using this frequently add a bash shortcut `alias gplf='git fetch && echo "HEAD was at $(git rev-parse --short HEAD)" && git reset --hard origin/$(git rev-parse --abbrev-ref HEAD)'` – Paul Odeon Nov 13, 2019 at 9:43

1 Brilliant. Thanks! People do not consider automated scripts when answering. This is very elegant when you just can't pass the branch name along. – Zeno Popovici Sep 24, 2020 at 16:24

2 This middle one worked for me: `git fetch` followed by `git reset --hard origin/$(git rev-parse --abbrev-ref HEAD)`. I chose it for its simplicity and the recentness of the answer. – Nike Jul 18, 2022 at 19:33

answered Jan 14, 2011 at 15:18
Ryan
439 4 2

6:39

35

I summarized other answers. You can execute `git pull` without errors:

```
git fetch --all
git reset --hard origin/master
git reset --hard HEAD
git clean -f -d
git pull
```

**Warning**: This script is very powerful, so you could lose your changes.

Share Improve this answer Follow

edited Jan 14, 2017 at 15:42
Peter Mortensen
**30.8k** 22 106 131

answered Aug 7, 2015 at 3:03
Robert Moon
**1,025** 10 17

2 This will overwrite modified files (files that were previously checked in) and it will remove untracked files (files that have never been checked in). Exactly what I was looking for, thanks! – styfle Mar 3, 2016 at 16:01

3 I suspect the third line `git reset --hard HEAD` may be redundant; my local man page (2.6.3) say that `reset` in the second line `git reset --hard origin/master` *"defaults to HEAD in all forms."* – Ada Richards Apr 19, 2016 at 15:40

2 @arichards I think your suspect is right but if second line will not work(by any reason) third line work well to reset. This solution doesn't need to be optimized. I just summarized other answers. That's all. Thank you for your comment. :) – Robert Moon Apr 20, 2016 at 2:12

Thanks for the summary. These steps are indeed powerful :) – Glorian Dec 10, 2020 at 17:28

29

Based on my own similar experiences, the solution offered by Strahinja Kustudic above is by far the best. As others have pointed out, simply doing hard reset will remove **all** the untracked files which could include lots of things that you don't want removed, such as config files. What is safer, is to remove only the files that are about to be added, and for that matter, you'd likely also want to checkout any locally-modified files that are about to be updated.

That in mind, I updated Kustudic's script to do just that. I also fixed a typo (a missing ' in the original).

```
#/bin/sh

# Fetch the newest code
git fetch

# Delete all files which are being added,
# so there are no conflicts with untracked files
for file in `git diff HEAD..origin/master --name-status | awk '/^A/ {print $2}'`
do
    echo "Deleting untracked file $file..."
    rm -vf "$file"
done

# Checkout all files which have been locally modified
for file in `git diff HEAD..origin/master --name-status | awk '/^M/ {print $2}'`
do
    echo "Checking out modified file $file..."
    git checkout $file
done

# Finally merge all the changes (you could use merge here as well)
git pull
```

Share Improve this answer

edited Aug 13, 2015 at 23:12

answered Feb 27, 2013 at 14:43

Rolf Kaise Rolf Kaiser **561** 6 9

in your note) instead of "git pull" will be
git repo. – Josh May 6, 2013 at 6:20

% of times. I updated my script with that
it a little differently than you. I checkout
rks all the time. – Strahinja Kustudic

**Your privacy**

By clicking "Accept all cookies", you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies Necessary cookies only

Customize settings

25

It seems like most answers here are focused on the `master` branch; however, there are times when I'm working on the same feature branch in two different places and I want a rebase in one to be reflected in the other without a lot of jumping through hoops.

Based on a combination of RNA's answer and torek's answer to a similar question, I've come up with this which works splendidly:

```
git fetch
git reset --hard @{u}
```

Run this from a branch and it'll only reset your local branch to the upstream version.

This can be nicely put into a git alias ( `git forcepull` ) as well:

```
git config alias.forcepull "!git fetch ; git reset --hard @{u}"
```

Or, in your `.gitconfig` file:

```
[alias]
  forcepull = "!git fetch ; git reset --hard @{u}"
```

Enjoy!

Share Improve this answer Follow

edited May 23, 2017 at 12:18
Community Bot
1 1

answered Feb 25, 2014 at 17:19
JacobEvelyn
3,911 1 40 51

This answer is also nice because it works regardless of which branch you are on! – leafmeal Sep 11, 2018 at 20:14

it clean -f -d would not do it.
Git (via a .gitignore entry, I
r *pull*, but a *clean* will not remove

answered Aug 3, 2011 at 9:23
Tierlieb
31
287 3 4

**Your privacy**

By clicking “Accept all cookies”, you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies Necessary cookies only

Customize settings

23

I believe there are two possible causes of conflict, which must be solved separately, and as far as I can tell none of the above answers deals with both:

- Local files that are untracked need to be deleted, either manually (safer) or as suggested in other answers, by `git clean -f -d`
- Local commits that are not on the remote branch need to be deleted as well. IMO the easiest way to achieve this is with: `git reset --hard origin/master` (replace 'master' by whatever branch you are working on, and run a `git fetch origin` first)

Share Improve this answer Follow edited Dec 12, 2011 at 20:05 answered Dec 12, 2011 at 19:54 tiho 6,685 4 32 31

23

I am not sure why anyone did not talk about `FETCH_HEAD` yet.

```
git fetch origin master && git reset --hard FETCH_HEAD
```

If you want to put it in an alias, the command would be:

```
git config --global alias.fpull '!git fetch origin master && git reset --hard
FETCH_HEAD'
```

Share Improve this answer Follow edited Jul 19, 2022 at 7:57 answered Jun 30, 2022 at 15:00 Ahmad Ismail 11.9k 6 53 91

Because SO does not trust someone to make a 1-char edit (?!?!?): & is not same as &&! – Connor Clark Jul 7, 2022 at 4:02

7, 2022 at 4:29

**Your privacy**

By clicking “Accept all cookies”, you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies Necessary cookies only

Customize settings

answered May 5, 2015 at 8:03 maximus 69 1,418 4 23 35

20

I have a strange situation that neither `git clean` or `git reset` works. I have to remove the conflicting file from `git index` by using the following script on every untracked file:

```
git rm [file]
```

Then I am able to pull just fine.

Share Improve this answer Follow

edited Dec 5, 2017 at 4:35
Jacob Gunther
351 5 14

answered Sep 19, 2011 at 14:18
Chen Zhang
227 2 3

Thanks! I found that this is needed if you've made any special adjustments to ignore changes on file in the repo. i.e `git update-index --assume-unchanged <file>` – skupjoe Jan 20 at 23:25

20

I know of a much easier and less painful method:

```
$ git branch -m [branch_to_force_pull] tmp
$ git fetch
$ git checkout [branch_to_force_pull]
$ git branch -D tmp
```

That's it!

Share Improve this answer Follow

edited Sep 5, 2018 at 16:52
Ricky McMaster
4,339 2 24 23

answered Sep 5, 2015 at 18:23
ddmytrenko
796 7 16

I tried doing as suggested in this answer. NO FILES AT ALL were pulled down from the remote
it - after all there is no reference at all
2020 at 11:58

racked files in my working copy with my
on a handful of files, it is unwieldy to do
n this thread. – Keith Knauber Mar 8 at

**Your privacy**

By clicking “Accept all cookies”, you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies Necessary cookies only

Customize settings

19

I just solved this myself by:

```
git checkout -b tmp # "tmp" or pick a better name for your local changes branch
git add -A
git commit -m 'tmp'
git pull
git checkout master # Or whatever branch you were on originally
git pull
git diff tmp
```

where the last command gives a list of what your local changes were. Keep modifying the "tmp" branch until it is acceptable and then merge back onto master with:

```
git checkout master && git merge tmp
```

For next time, you can probably handle this in a cleaner way by looking up "git stash branch" though stash is likely to cause you trouble on the first few tries, so do first experiment on a non-critical project...

Share Improve this answer Follow

edited Jan 14, 2017 at 15:13
Peter Mortensen
30.8k 22 106 131

answered Dec 3, 2010 at 15:00
Simon B.
2,550 24 30

17

Just do

```
git fetch origin branchname
git checkout -f origin/branchname // This will overwrite ONLY new included files
git checkout branchname
git merge origin/branchname
```

s or directories you wanted to

answered Oct 19, 2015 at 9:54
user2696128
191 1 2

31

to merge from, that got rid of all the
destination, and finally merge without

16

Requirements:

1. Track local changes so no-one here ever loses them.
2. Make the local repository match the remote origin repository.

Solution:

1. **Stash** the local changes.
2. **Fetch** with a **clean** of **files** and **directories** ignoring **.gitignore** and **hard reset** to **origin**.

```
git stash --include-untracked
git fetch --all
git clean -fdx
git reset --hard origin/master
```

Share Improve this answer Follow

edited Jan 14, 2017 at 15:44
Peter Mortensen
**30.8k** 22 106 131

answered Sep 1, 2015 at 23:00
vezenkov
**4,019** 1 26 27

---

15

Reset the index and the head to `origin/master`, but do not reset the working tree:

```
git reset origin/master
```

Share Improve this answer Follow

answered Feb 15, 2013 at 13:41
user811773

vorking tree so you can check it in
dded so it was stuck. Weird, I know.

**Your privacy**

By clicking “Accept all cookies”, you agree Stack Exchange can store cookies on your device and disclose information in accordance with our Cookie Policy.

Accept all cookies Necessary cookies only

Customize settings

15

These four commands work for me.

```
git reset --hard HEAD
git checkout origin/master
git branch -D master
git checkout -b master
```

To check/pull after executing these commands

```
git pull origin master
```

I tried a lot but finally got success with these commands.

Share Improve this answer Follow

answered Mar 20, 2014 at 4:24
Vishesh Chandra
**6,951** 6 35 38

2 "git branch -D master" delete the branch. so be careful with it. I prefer to use "git checkout origin/master -b <new branch name>" which create a new branch with a new name and you done need 3,4 lines. Also recommended to use "git clean -f" as well. – Chand Priyankara Apr 5, 2014 at 11:49

1 2 Next

**Highly active question**. Earn 10 reputation (not counting the association bonus) in order to answer this question. The reputation requirement helps protect this question from spam and non-answer activity.