## Undoing a 'git push'

Asked 11 years, 11 months ago Active 3 months ago Viewed 808k times



Here's what I did on my supposed-to-be-stable branch...

671



×

323

```
% git rebase master
First, rewinding head to replay your work on top of it...
Fast-forwarded alpha-0.3.0 to master.
% git status
# On branch alpha-0.3.0
# Your branch is ahead of 'origin/alpha-0.3.0' by 53 commits.
nothing to commit (working directory clean)
% git push
Fetching remote heads...
  refs/
  refs/heads/
  refs/tags/
  refs/remotes/
'refs/heads/master': up-to-date
updating 'refs/heads/alpha-0.3.0'
  from cc4b63bebb6e6dd04407f8788938244b78c50285
     83c9191dea88d146400853af5eb7555f252001b0
    done
```

That was all a mistake as I later realized. I'd like to undo this entire process, and revert the alpha-0.3.0 branch back to what it was.

What should I do?

'refs/heads/unstable': up-to-date
Updating remote server info

git git-push

Share Edit Follow Flag



asked Aug 13 '09 at 7:23

Cyrus

7,359 3 18 23

this stackoverflow post perhaps? <a href="mailto:stackoverflow.com/questions/134882/undoing-a-git-rebase">stackoverflow.com/questions/134882/undoing-a-git-rebase</a> - Steen Feb 24 '11 at 3:27

4 — It's not really the same situation, undoing a rebase is a local repository scenario, undoing a git push involves a remote repository and can be more tricky depending on the access you have. – CB Bailey Feb 24 '11 at 3:27

Steen - you're right - I probably should have I suppose. I figured that the blessed repository that all pull from is more of an admin task and so belongs here, where general client-side git is a stackoverflow question. – Cyrus Feb 24 '11 at 3:27

Quick clarification - I'm guessing if you refer to a git commit by a *partial* hash value, git will assume you're talking about the commit whose hash begins with that string? – Gershy Nov 27 '15 at 16:32

13 Answers Active Oldest Votes



You need to make sure that no other users of this repository are fetching the incorrect changes or trying to build on top of the commits that you want removed because you are about to rewind history.

1078

Then you need to 'force' push the old reference.



git push -f origin last\_known\_good\_commit:branch\_name



or in your case

```
git push -f origin cc4b63bebb6:alpha-0.3.0
```

You may have receive.denyNonFastForwards set on the remote repository. If this is the case, then you will get an error which includes the phrase [remote rejected].

In this scenario, you will have to delete and recreate the branch.

```
git push origin :alpha-0.3.0
git push origin cc4b63bebb6:refs/heads/alpha-0.3.0
```

If this doesn't work - perhaps because you have receive.denyDeletes set, then you have to have direct access to the repository. In the remote repository, you then have to do something like the following plumbing command.

```
git update-ref refs/heads/alpha-0.3.0 cc4b63bebb6 83c9191dea8
```

Share Edit Follow Flag



answered Aug 13 '09 at 7:47



- A perfect and well explained response thank you very much. For anyone else who stumbles accross this, for academic reasons I tried both of the first 2 approaches, and both worked obviously if the first one works, it's the cleanest approach. If I chould UP you 10 times Charles, I would. :) Cyrus Aug 13 '09 at 8:51
- 146 For quick-reference, the first line here is git push -f origin last\_known\_good\_commit:branch\_name philfreo Aug 29 '11 at 23:16
- git push -f origin cc4b63bebb6:alpha-0.3.0 => this one helped me, Note alpha-0.3.0 is the branch name and cc4b63bebb6 is the commit id we wish to revert back to. so, after carrying out this command we wil be in cc4b63bebb6 commit id. kumar Dec 28 '11 at 11:51
- This solution is highly dangerous if you are working in a shared repo. As a best practice, all commits pushed to a remote repo that is shared should be considered 'immutable'. Use 'git revert' instead:

  <a href="https://example.com/git/docs/...">kernel.org/pub/software/scm/git/docs/...</a> Saboosh Jan 13 '12 at 20:47
- jww compared to everything else, git is the most feature-rich and efficient source control tool available. Every team uses it differently. It's worth spending a weekend playing around with a fresh repository and going through all the common scenarios. Once you've spent some time working with it, development is a lot less stressful. user1491819 Oct 23 '15 at 4:29



If you want to ignore the last commit that you have just pushed in the remote branch: this will not remove the commit but just ignoring it by moving the git pointer to the commit one earlier, refered by HEADA or HEAD^1





git push origin +HEAD^:branch



But if you have already pushed this commit, and others have pulled the branch. In this case, rewriting your branch's history is undesirable and you should instead revert this commit:

```
git revert <SHA-1>
git push origin branch
```

Share Edit Follow Flag



answered Jun 22 '17 at 13:22







mkebri 1.461 14



The question is about "push" then it concern the Remote branch. No to move the HEAD about one commit which mean ignore the last commit pushed just do this: git push origin +HEAD^:your\_branch - mkebri Apr 20 '18 at 15:43



**Scenario 1**: If you want to undo the last commit say 8123b7e04b3, below is the command(this worked for me):

```
git push origin +8123b7e04b3^:<br/>branch_name>
```



Output looks like below:

```
Total 0 (delta 0), reused 0 (delta 0)
To https://testlocation/code.git
+ 8123b7e...92bc500 8123b7e04b3^ -> master (forced update)
```

Note: To update the change to your local code (to remove the commit locally as well):

```
$ git reset --hard origin/<branchName>
Message displayed is : HEAD is now at 8a3902a comments_entered_for_commit
```

**Additional info: Scenario 2**: In some situation, you may want to revert back what you just undo'ed (basically undo the undo) through the previous command, then use the below command:

```
git reset --hard 8123b7e04b3
git push
```

Output:

```
HEAD is now at cc6206c Comment_that_was_entered_for_commit
```

More info here: https://github.com/blog/2019-how-to-undo-almost-anything-with-git

Share Edit Follow Flag

edited Mar 30 at 20:27

answered Dec 19 '17 at 12:02



**1.453** 1 17 20



Scenario 1 should the accepted answer since the question did not specify which commit to delete. The accepted answer only deletes the *last* commit. This answer deletes *any* commit. – Dominic Cerisano Oct 7 '19 at 20:46



git reset --hard HEAD^
git push origin -f



This will remove the last commit from your local device as well as Github



Share Edit Follow Flag





Undo multiple commits git reset --hard @ad5a7a6 (Just provide commit SHA1 hash)

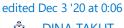
12 Undo last commit



git reset --hard HEAD~1 (changes to last commit will be removed) git reset --soft HEAD~1 (changes to last commit will be available as uncommitted local modifications)



Share Edit Follow Flag



DINA TAKLIT

answered May 12 '17 at 6:40





you can use the command reset



git reset --soft HEAD^1



then:



git reset <files>
git commit --amend

and

git push -f



The existing answers are good and correct, however what if you need to undo the push but:



1. You want to keep the commits locally or you want to keep uncommitted changes



2. You don't know how many commits you just pushed



Use this command to revert the change to the ref:

git push -f origin refs/remotes/origin/<branch>@{1}:<branch>

Share Edit Follow Flag

answered Sep 19 '19 at 17:57



Vlad274

1 27 40



git push origin +7f6d03:master

This will revert your repo to mentioned commit number

Share Edit Follow Flag

answered Jan 23 '17 at 7:52



ireshika piyumalie **1.616** 18 21



This is most straight forward answer. You are a live saver. – Ekundayo Blessing Funminiyi Sep 19 '17 at 10:43

3

Remember, that won't reset your local files. – K-Gun Nov 21 '18 at 5:29

This is what I needed, as I wanted to simply undo a push without losing my local changes. - Hashim Aziz Apr 4 at 0:52



Another way to do this:



1. create another branch



2. checkout the previous commit on that branch using "git checkout"



3. push the new branch.



- 4. delete the old branch & push the delete (use git push origin --delete <branch\_name>)
- 5. rename the new branch into the old branch
- 6. push again.

Share Edit Follow Flag





Replace 35f6af6f77f116ef922e3d75bc80a4a466f92650 with your own commit.

35

116

Share Edit Follow Flag edited Apr 24 '15 at 18:24

edited Apr 24 '15 at 18:24

Peter Mortensen
28.6k 21 95 123

answered Sep 3 '12 at 11:39

neoneye
45k 23 156 144

- How do I come up with the 35f6af6f77f116ef922e3d75bc80a4a466f92650 ID? This answer would be better if you could explain that. Volomike Jun 12 '13 at 2:58
- @Volomike (and Googling devs of the future), this question describes many ways of obtaining it: <u>version control</u> and hash question on SO Jaime Oct 28 '13 at 17:32
  - This is the right answer, because with "git reset" you should not be able to push (Updates were rejected because the tip of your current branch is behind its remote counterpart) or you need to force the pull which is not really clean. Thomas Decaux Aug 19 '14 at 10:54
  - This was working for me. However, be careful as revert will revert all changes in your local files. user1941537 Mar 12 '19 at 10:14
  - I opted for this approach multiple times but also I use git rebase -i <id-before-last-good-commit> to do an interactive rebase and clean up history as suggested here, <a href="mailto:stackoverflow.com/questions/5189560/...">stackoverflow.com/questions/5189560/...</a>. Ernesto Allely Jun 17 '19 at 7:38



A way to do it without losing the changes you wanted:

41



```
git reset cc4b63b
git stash
git push -f origin alpha-0.3.0
git stash pop
```

Then you can choose the files you meant to push

Share Edit Follow Flag

edited Mar 2 '12 at 4:30

answered Nov 12 '11 at 0:04



This saved my day! – jawsofdoom Feb 16 at 21:17

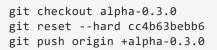


Thank you for this! – Mekky\_Mayata May 1 at 12:54



I believe that you can also do this:

185





This is very similar to the last method, except you don't have to muck around in the remote repo.

Share Edit Follow Flag

answered Nov 24 '09 at 16:48



**Benny Wong 6,363** 5 28 25

11 — This worked for me as well, but it's worth noting that this will "re-write" history on the remote. This may be what you want, but it may not be! – Tom Aug 25 '11 at 17:09

+1 for this answer that really helped me out. I also wanted to add (and make things clear) that the commit ID (which comes after the " --hard " parameter) should be the ID of whatever commit you want to reset your branch to. – Michael Dautermann Jul 27 '12 at 20:20

Rewrote history nicely... anyone who could have pulled the changes, I just made sure they did a git reset -- hard [commit\_id] so we didn't mess with the space-time continuum. – Alien Life Form Sep 14 '15 at 22:36

11 — What is the + for in "git push origin +alpha-0.3.0"? – jpierson Mar 31 '17 at 17:55

@jpierson + forces the push to take place, similarly to -f (but slightly different: stackoverflow.com/a/25937833/1757149). Without it, if you try git push origin alpha-0.3.0 the push will fail: Updates were rejected because the tip of your current branch is behind . - A\_ Nov 28 '18 at 15:37

l