How can I add an empty directory to a Git repository?

Asked 11 years, 2 months ago Active 12 days ago Viewed 888k times

4039

777

How can I add an empty directory (that contains no files) to a Git repository?

git directory git-add

edited Dec 20 '14 at 15:49

jub0bs

38.9k 19 120 143

asked Sep 22 '08 at 16:41

Laurie Young

123k 13 43 53

14 While it's not useful, there is a way to hack an empty (really empty) directory into your repo. It won't `checkout` with current versions of Git, however. – tiwo Jul 22 '12 at 14:18

315 @tiwo I for one disagree that it's not useful. Your directory hierarchy is part of your project, so it should be version controlled. – JBentley Jan 29 '13 at 20:19

105 In my case, I'd like to add a directory structure for tmp files, but not the tmp files themselves. By doing this, my tester has the correct structure (otherwise there are errors) but I don't clog my commits with tmp data. So yes, it's useful to me! – Adam Marshall Mar 13 '13 at 3:32

43 @AdamMarshall I think tiwo was saying that the hack is not useful, since it is ignored by checkout. Tmp dirs do sound like a useful feature for a VCS. – Quantum7 Apr 22 '13 at 21:33

29 Why not have the procedure that creates the tmp files also create the tmp directory? – RyPeck Jul 9 '13 at 3:11

33 Answers

1 2 next

3907

Another way to make a directory stay (almost) empty (in the repository) is to create a `.gitignore` file inside that directory that contains these four lines:

```
# Except this file
!.gitignore
```

Then you don't have to get the order right the way that you have to do in m104's solution.

This also gives the benefit that files in that directory won't show up as "untracked" when you do a git status.

Making @GreenAsJade's comment persistent:

> I think it's worth noting that this solution does precisely what the question asked for, but is not perhaps what many people looking at this question will have been looking for. This solution guarantees that the directory remains empty. It says "I truly never want files checked in here". As opposed to "I don't have any files to check in here, yet, but I need the directory here, files may be coming later".

edited Nov 20 '18 at 11:59
HelloGoodbye
2,032 3 20 36

answered May 31 '09 at 22:10
Jamie Flournoy
41k 1 20 11

21 I think the README solution proposed by @JohnMee should be used together with this one; the .gitignore file provides an explanation of what we want to keep out of version control, while the README file explains what is the purpose of the directory, which are both very important pieces of information. – pedromanoel Jan 17 '13 at 11:11

15 @pedromanoel I write the documentation you would put in the `README` inside the `.gitignore` file (as comments). – Carlos Campderrós Jul 19 '13 at 8:20

55 spot the 1 difference: 1.) an empty folder, 2.) a folder with .gitignore file in it. ;-) – Peter Perháč Feb 11 '14 at 14:31

4 This is perfect for *cache* folders. – redolent Mar 26 '14 at 2:30

7 Unfortunately, this results in a non-empty directory, it has a single hidden file. – pedorro Dec 15 '14 at 20:09

1

I search into this question becuase: I create a new directory and it contains many files. Among these files, some I want to add to git repository and some not. But when I do "git status". It only shows:

```
Untracked files:
  (use "git add <file>..." to include in what will be committed)
```

It does not list the seperate files in this new directory. Then I think maybe I can add this directory only and then deal with the seperate files. So I google "git add directory only".

In my situation, I found I can just add one file in the new directory that I am sure I want to add it to git.

```
git add new_folder/some_file
```

After this, "git status" will show the status of seperate files.

answered Nov 20 at 2:21

user547960
43 1 5

1

Reading @ofavre's and @stanislav-bashkyrtsev's answers using broken GIT submodule references to create the GIT directories, I'm surprised that nobody has suggested yet this simple amendment of the idea to make the whole thing sane and safe:

Rather than *hacking a fake submodule into GIT*, just **add an empty real one**.

Enter: https://gitlab.com/empty-repo/empty.git

A GIT repository with exactly one commit:

```
commit e84d7b81f0033399e325b8037ed2b801a5c994e0
Author: Nobody <none>
Date: Thu Jan 1 00:00:00 1970 +0000
```

No message, no committed files.

Usage

To add an empty directory to you GIT repo:

```
git submodule add https://gitlab.com/empty-repo/empty.git path/to/dir
```

```
find . -type d -empty -delete -exec git submodule add -f https://gitlab.com/empty-
repo/empty.git \{\} \;
```

Git will store the latest commit hash when creating the submodule reference, so you don't have to worry about me (or GitLab) using this to inject malicious files. Unfortunately I have not found any way to force which commit ID is used during checkout, so you'll have to manually check that the reference commit ID is `e84d7b81f0033399e325b8037ed2b801a5c994e0` using `git submodule status` after adding the repo.

Still not a native solution, but the best we probably can have without somebody getting their hands *really*, *really* dirty in the GIT codebase.

Appendix: Recreating this commit

You should be able to recreate this exact commit using (in an empty directory):

```
# Initialize new GIT repository
git init

# Set author data (don't set it as part of the `git commit` command or your default data
will be stored as “commit author”)
git config --local user.name "Nobody"
git config --local user.email "none"

# Set both the commit and the author date to the start of the Unix epoch (this cannot be
done using `git commit` directly)
export GIT_AUTHOR_DATE="Thu Jan 1 00:00:00 1970 +0000"
export GIT_COMMITTER_DATE="Thu Jan 1 00:00:00 1970 +0000"

# Add root commit
git commit --allow-empty --allow-empty-message --no-edit
```

Creating reproducible GIT commits is surprisingly hard…

answered Oct 24 at 14:23

ntninja

453 4 13

1

```
# Ignore files but not directories. * matches both files and directories
# but */ matches only directories. Both match at every directory level
# at or below this one.
*
!*/

# Git doesn't track empty directories, so track .keepdir files, which also
# tracks the containing directory.
!.keepdir

# Keep this file and the explanation of how this works
!.gitignore
!Readme.md
```

answered Oct 22 at 18:46

aball

36 2

Many have already answered this question. Just adding a PowerShell version here.

6

> Find all the empty folders in the directory
>
> Add a empty .gitkeep file in there

```
Get-ChildItem 'Path to your Folder' -Recurse -Directory | Where-Object
{[System.IO.Directory]::GetFileSystemEntries($_.FullName).Count -eq 0} | ForEach-Object
{ New-Item ($_.FullName + "\.gitkeep") -ItemType file}
```

answered Aug 13 at 9:34

Hainan.Z

699 6 9

An easy way to do this is by adding a `.gitkeep` file to the directory you wish to (currently) keep empty.

answered Jun 17 at 5:03

arcseldon
23.9k 7 81 98

290

Why would we need empty versioned folders

First things first:

> An empty directory *cannot be part of a tree under the Git versioning system*.

It simply won't be tracked. But there are scenarios in which "versioning" empty directories can be meaningful, for example:

- scaffolding a **predefined folder structure**, making it available to every user/contributor of the repository; or, as a specialized case of the above, creating a folder for **temporary files**, such as a `cache/` or `logs/` directories, where we want to provide the folder but `.gitignore` its contents
- related to the above, some projects *won't work without some folders* (which is often a hint of a poorly designed project, but it's a frequent real-world scenario and maybe there could be, say, permission problems to be addressed).

Some suggested workarounds

Many users suggest:

1. Placing a `README` file or another file with some content in order to make the directory non-empty, or
2. Creating a `.gitignore` file with a sort of "reverse logic" (i.e. to include all the files) which, at the end, serves the same purpose of approach #1.

While *both solutions surely work* I find them inconsistent with a meaningful approach to Git versioning.

- Why are you supposed to put bogus files or READMEs that maybe you don't really want in your project?
- Why use `.gitignore` to do a thing (*keeping* files) that is the very opposite of what it's meant for (*excluding* files), even though it is possible?

.gitkeep approach

Although it may seem not such a big difference:

- You use a file that has the *single* purpose of keeping the folder. You don't put there any info you don't want to put.

  For instance, you should use READMEs as, well, READMEs with useful information, not as an excuse to keep the folder.

  Separation of concerns is always a good thing, and you can still add a `.gitignore` to ignore unwanted files.
- Naming it `.gitkeep` makes it very clear and straightforward from the filename itself (and also *to other developers*, which is good for a shared project and one of the core purposes of a Git repository) that this file is
  - A file unrelated to the code (because of the leading dot and the name)
  - A file clearly related to Git
  - Its purpose (**keep**) is clearly stated and consistent and semantically opposed in its meaning to **ignore**

Adoption

I've seen the `.gitkeep` approach adopted by very important frameworks like Laravel, Angular-CLI.

edited Jun 9 at 9:53

Manu Manjunath
4,678 1 21 28

answered Dec 4 '13 at 23:32

Cranio
8,253 2 26 49

6 You missed one thought - whats the reason for keeping and empty folder (e.g. /logs, /tmp, /uploads)? Yes - its to keep the folder empty. :) So if you want to keep a folder empty, you have to ignore the files inside it. – Roman Oct 3 '14 at 0:08

14 @RomanAllenstein: not necessarily. It could be that you create a repo with a given structure which can become populated later. Those files will be added to the repo as soon as they are created, and it will be annoying to start deleting or editing .gitignore files (and dangerous, because probably you do not even realize that they are not being tracked: git is ignoring them) – dangonfast Feb 17 '15 at 16:06

44 @Behnam: I'll take the downvote, but my research on the S.O. meta shows no concern towards verbose answers, as long as they provide enough detail and clarity to be useful for every reader (and every skill level). Still I'm very open to any criticism and thank you for having declared the reason publicly, I take it very positively. – Cranio Oct 10 '16 at 13:22

4 If you edit your answer to replace `.gitkeep` with any other non git-prefixed file name you get my upvote, I think this one is the best and most informative answer. Reason: I think ".git*" should be reserved for git prescribed files, while this is just a mere placeholder. My first guess when I saw that is that for example a ".gitkeep" file would be auto-ignored (that would be a nice feature) but that is not the case, right? – Johnny Nov 21 '16 at 9:34

2 I wonder why people have such a hard time to understand why one wants to add "empty" folders to git. You have to start somewhere, right? So, usually you start with your projects folder structure and - alas - at the start of the project there is nothing there yet. Once your project repo is done, team

14

I like the answers by @Artur79 and @mjs so I've been using a combination of both and made it a standard for our projects.

```
find . -type d -empty -exec touch {}/.gitkeep \;
```

However, only a handful of our developers work on Mac or Linux. A lot work on Windows and I could not find an equivalent simple one-liner to accomplish the same there. Some were lucky enough to have Cygwin installed for other reasons, but prescribing Cygwin just for this seemed overkill.

Edit for a better solution

So, since most of our developers already have Ant installed, the first thing I thought of was to put together an Ant build file to accomplish this independently of the platform. This can still be found here

However, I later thought It would be better to make this into a small utility command, so I recreated it using Python and published it to the PyPI here. You can install it by simply running:

```
pip3 install gitkeep2
```

It will allow you to create and remove `.gitkeep` files recursively, and it will also allow you to add messages to them for your peers to understand why those directories are important. This last bit is bonus. I thought it would be nice if the `.gitkeep` files could be self-documenting.

```
$ gitkeep --help
Usage: gitkeep [OPTIONS] PATH

  Add a .gitkeep file to a directory in order to push them into a Git repo
  even if they're empty.

  Read more about why this is necessary at: https://git.wiki.kernel.org/inde
  x.php/Git_FAQ#Can_I_add_empty_directories.3F

Options:
  -r, --recursive     Add or remove the .gitkeep files recursively for all
                      sub-directories in the specified path.
  -l, --let-go        Remove the .gitkeep files from the specified path.
  -e, --empty         Create empty .gitkeep files. This will ignore any
                      message provided
  -m, --message TEXT  A message to be included in the .gitkeep file, ideally
```

```
-v, --verbose       Print out everything.
--help              Show this message and exit.
```

I hope you find it useful.

edited Dec 22 '18 at 0:20

answered May 15 '17 at 19:08

Mig82

2,464 22 41

32

The Ruby on Rails log folder creation way:

```
mkdir log && touch log/.gitkeep && git add log/.gitkeep
```

Now the log directory will be included in the tree. It is super-useful when deploying, so you won't have to write a routine to make log directories.

The logfiles can be kept out by issuing,

```
echo log/dev.log >> .gitignore
```

but you probably knew that.

edited Oct 26 '18 at 20:52

rogerdpack

40.5k 21 153 277

answered Oct 22 '12 at 13:24

Thomas E

3,550 2 16 13

22 What does that have to do with Ruby on Rails? – Quolonel Questions Sep 29 '15 at 9:11

28

Git does not track empty directories. See the Git FAQ for more explanation. The suggested workaround is to put a `.gitignore` file in the empty directory. I do not like that solution, because the `.gitignore` is "hidden" by Unix convention. Also there is no explanation why the directories are empty.

The real question is why do you need the empty directory in git? Usually you have some sort of build script that can create the empty directory before compiling/running. If not then make one. That is a far better solution than putting empty directories in git.

So you have some reason why you need an empty directory in git. Put that reason in the README file. That way other developers (and future you) know why the empty directory needs to be there. You will also know that you can remove the empty directory when the problem requiring the empty directory has been solved.

To list every empty directory use the following command:

```
find -name .git -prune -o -type d -empty -print
```

To create placeholder READMEs in every empty directory:

```
find -name .git -prune -o -type d -empty -exec sh -c \
  "echo this directory needs to be empty because reasons > {}/README.emptydir" \;
```

To ignore everything in the directory except the README file put the following lines in your `.gitignore`:

```
path/to/emptydir/*
!path/to/emptydir/README.emptydir
path/to/otheremptydir/*
!path/to/otheremptydir/README.emptydir
```

Alternatively, you could just exclude *every* README file from being ignored:

```
path/to/emptydir/*
path/to/otheremptydir/*
!README.emptydir
```

To list every README after they are already created:

```
find -name README.emptydir
```

edited Feb 8 '18 at 13:41 answered May 6 '11 at 15:45

6

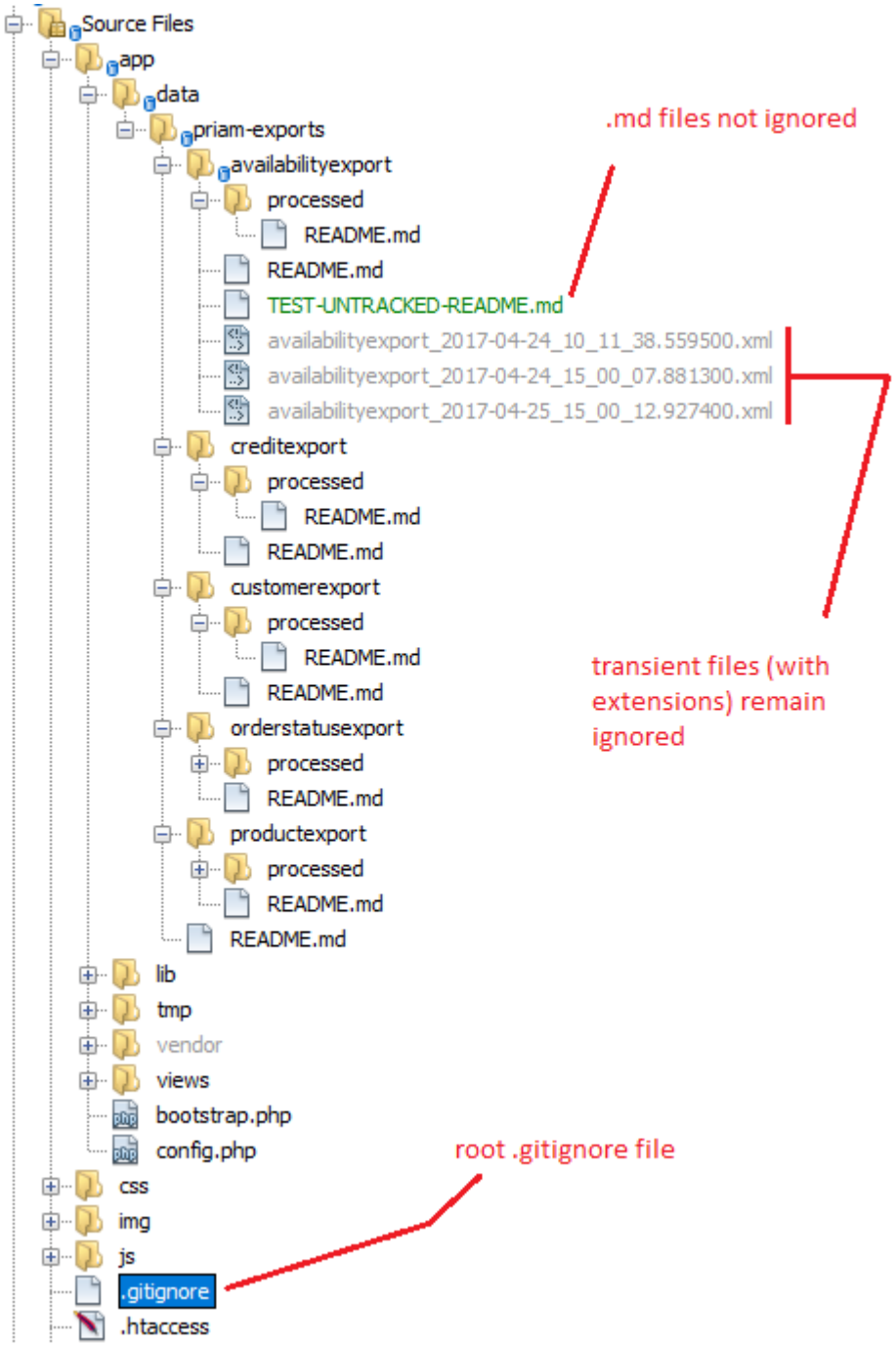
If you want to add a folder that will house a lot of transient data in multiple semantic directories, then one approach is to add something like this to your root .gitignore...

```
/app/data/**/*.*
!/app/data/**/*.md
```

Then you can commit descriptive README.md files (or blank files, doesn't matter, as long as you can target them uniquely like with the `*.md` in this case) in each directory to ensure that the directories all remain part of the repo but the files (with extensions) are kept ignored. LIMITATION: `.` 's are not allowed in the directory names!

You can fill up all of these directories with xml/images files or whatever and add more directories under `/app/data/` over time as the storage needs for your app develop (with the README.md files serving to burn in a description of what each storage directory is for exactly).

There is no need to further alter your `.gitignore` or decentralise by creating a new `.gitignore` for each new directory. Probably not the smartest solution but is terse gitignore-wise and always works for me. Nice and simple! ;)



ajmedway
1,294 9 26

2

Sometimes I have repositories with folders that will only ever contain files considered to be "content"—that is, they are not files that I care about being versioned, and therefore should never be committed. With Git's .gitignore file, you can ignore entire directories. But there are times when having the folder in the repo would be beneficial. Here's a excellent solution for accomplishing this need.

What I've done in the past is put a .gitignore file at the root of my repo, and then exclude the folder, like so:

```
/app/some-folder-to-exclude
/another-folder-to-exclude/*
```

However, these folders then don't become part of the repo. You could add something like a README file in there. But then you have to tell your application not to worry about processing any README files.

If your app depends on the folders being there (though empty), you can simply add a .gitignore file to the folder in question, and use it to accomplish two goals:

Tell Git there's a file in the folder, which makes Git add it to the repo. Tell Git to ignore the contents of this folder, minus this file itself. Here is the .gitignore file to put inside your empty directories:

```
*
!.gitignore
```

The first line (*) tells Git to ignore everything in this directory. The second line tells Git not to ignore the .gitignore file. You can stuff this file into every empty folder you want added to the repository.

answered Jul 11 '16 at 18:36

Rahul Sinha
761 7 14

Sometimes you have to deal with bad written libraries or software, which need a "real" empty and existing directory. Putting a simple `.gitignore` or `.keep` might break them and cause a bug. The following might help in these cases, but no guarantee...

```
mkdir empty
```

Then you add a broken symbolic link to this directory (but on any other case than the described use case above, please use a `README` with an explanation):

```
ln -s .this.directory empty/.keep
```

To ignore files in this directory, you can add it in your root `.gitignore`:

```
echo "/empty" >> .gitignore
```

To add the ignored file, use a parameter to force it:

```
git add -f empty/.keep
```

After the commit you have a broken symbolic link in your index and git creates the directory. The broken link has some advantages, since it is no regular file and points to no regular file. So it even fits to the part of the question "(that contains no files)", not by the intention but by the meaning, I guess:

```
find empty -type f
```

This commands shows an empty result, since no files are present in this directory. So most applications, which get all files in a directory usually do not see this link, at least if they do a "file exists" or a "is readable". Even some scripts will not find any files there:

```
$ php -r "var_export(glob('empty/.*'));"
array (
  0 => 'empty/.',
  1 => 'empty/..',
)
```

But I strongly recommend to use this solution only in special circumstances, a good written `README` in an empty directory is usually a better solution. (And I do not know if this works with a windows filesystem...)

answered Jun 2 '16 at 16:42

4

Adding one more option to the fray.

Assuming you would like to add a directory to `git` that, for all purposes related to `git`, should remain empty and never have it's contents tracked, a `.gitignore` as suggested numerous times here, will do the trick.

The format, as mentioned, is:

```
*
!.gitignore
```

Now, if you want a way to do this at the command line, in one fell swoop, while *inside* the directory you want to add, you can execute:

```
$ echo "*" > .gitignore && echo '!.gitignore' >> .gitignore && git add .gitignore
```

Myself, I have a shell script that I use to do this. Name the script whatever you whish, and either add it somewhere in your include path, or reference it directly:

```
#!/bin/bash

dir=''

if [ "$1" != "" ]; then
    dir="$1/"
fi

echo "*" > $dir.gitignore && \
echo '!.gitignore' >> $dir.gitignore && \
git add $dir.gitignore
```

With this, you can either execute it from within the directory you wish to add, or reference the directory as it's first and only parameter:

```
$ ignore_dir ./some/directory
```

Another option (in response to a comment by @GreenAsJade), if you want to track an empty folder that *MAY* contain tracked files in the future, but will be empty for now, you can ommit the `*` from the `.gitignore` file, and check *that* in. Basically, all the file is saying is "do

```
!.gitignore
```

That's it, check that in, and you have an empty, yet tracked, directory that you can track files in at some later time.

The reason I suggest keeping that one line in the file is that it gives the `.gitignore` purpose. Otherwise, some one down the line may think to remove it. It may help if you place a comment above the line.

edited May 28 '16 at 16:38

answered May 26 '16 at 1:18

Mike
1,659 10 31

12

You can't and unfortunately will never be able to. This is a decision made by Linus Torvald himself. He knows what's good for us.

There is a rant out there somewhere I read once.

I found *Re: Empty directories..*, but maybe there is another one.

You have to live with the workarounds...unfortunately.

edited Apr 21 '16 at 11:35

GAMITG
3,472 7 28 49

answered Mar 15 '15 at 18:17

user2334883
351 3 4

1 I know you posted this as an example of a bad argument, but I appreciate the link because it's actually a well-reasoned argument against tracking directories. ;-) – clacke Mar 16 '15 at 8:32

1 This answer seems to be inconsistent, since in the next post on the referenced thread, Linus Torvald says he expects that they will need to add directory tracking: markmail.org/message/libip4vpvvxhyqbl . In fact, he says he "would welcome patches that [add support for tracking empty directories]" – Patrick M Aug 1 '17 at 20:12

8

The solution of Jamie Flournoy works great. Here is a bit enhanced version to keep the `.htaccess` :

```
# Ignore everything in this directory
```

```
!.gitignore
!.htaccess
```

With this solution you are able to commit a empty folder, for example `/log` , `/tmp` or `/cache` and the folder will stay empty.

edited May 23 '17 at 12:02
Community ♦
1 1

answered Jun 22 '14 at 13:06
Roman
1,725 2 20 37

2 He wants to keep a empty directory and not a file. – gvsrepins Jul 29 '14 at 2:55

2 And i have mentioned that it will keep the .htaccess, too. Example: if a software has a directory for log-files (like oxid eshop) that should not be accesible via web, there is a .htaccess in the directory. If you put the above mentioned .gitignore in the folder, the .htaccess will not be comitted and the folder will be accessible via web. – Roman Jul 31 '14 at 8:17

1 @Wallacoloo Related to the question you're right, nevertheless the file is useful, I'll use it for an upload-directory like that where files shall be protected by .htaccess. Contrary to Romans explanation the .htaccess-file will be committed as it's excluded by the ignore-rule. [old thread, I know] – David Aug 28 '17 at 8:10

2

You can save this code as create_readme.php and run the PHP code from the root directory of your Git project.

```
> php create_readme.php
```

It will add README files to all directories that are empty so those directories would be then added to the index.

```
<?php
    $path = realpath('.');
    $objects = new RecursiveIteratorIterator(new RecursiveDirectoryIterator($path),
RecursiveIteratorIterator::SELF_FIRST);
    foreach($objects as $name => $object){
        if ( is_dir($name) && ! is_empty_folder($name) ){
            echo "$name\n" ;
            exec("touch ".$name."/"."README");
        }
    }

    function is_empty_folder($folder) {
```

```
                return true; // Not empty
            }
        }
?>
```

Then do

```
git commit -m "message"
git push
```

edited Apr 21 '16 at 11:34
GAMITG
3,472 7 28 49

answered Jun 26 '11 at 22:41
user665190
37 2

8

As mentioned it's not possible to add empty directories, but here is a one liner that adds empty .gitignore files to all directories.

```
ruby -e 'require "fileutils" ; Dir.glob(["target_directory","target_directory/**"]).each { |f| FileUtils.touch(File.join(f, ".gitignore")) if File.directory?(f) }'
```

I have stuck this in a Rakefile for easy access.

edited Apr 21 '16 at 11:34
GAMITG
3,472 7 28 49

answered Apr 19 '11 at 14:10
Peter Hoeg
833 9 12

6 I'd rather use `find . -type d -empty -print0 | xargs --null bash -c 'for a; do { echo "*"; echo "!.gitignore"; } >>"$a/.gitignore"; done' --` – Tino Oct 21 '11 at 6:35

20

Let's say you need an empty directory named *tmp* :

```
$ mkdir tmp
$ touch tmp/.gitignore
$ git add tmp
```

In other words, you need to add the .gitignore file to the index before you can tell Git to ignore it (and everything else in the empty directory).

edited Apr 21 '16 at 11:34

GAMITG
3,472 7 28 49

answered Oct 8 '08 at 0:13

m104
978 5 6

11 Two things: You could just "echo '*' > tmp/.gitignore" instead of touching, and "git commit -m" does not commit changes done after you've added the files to the index. – Christoffer Hammarström Jan 28 '10 at 15:50

6 If you just do `echo bla > file` you will not get `file: File exists` because `>` will overwrite the file if it's already there or create a new one if it doesn't exist. – psyrendust Apr 1 '14 at 19:53

3 `/bin/sh` cultural assumption!* If "here" is `csh` and the variable `noclobber` is set, you will indeed get `file: File exists` . If someone says "I get this", don't assume they're an idiot and reply "No you don't". * c2.com/cgi/wiki?AmericanCulturalAssumption – clacke Mar 16 '15 at 8:26

1 @clacke If someone decides to use a different shell than everyone else, they should state that expressly if they are encountering problems. Unlike with nationality, everyone has their free choice of shell. – SeldomNeedy May 25 '16 at 19:38

2 @SeldomNeedy Maybe they are looking for help because they don't even know they are using a different shell than everybody else. – clacke May 30 '16 at 8:37

10

When you add a `.gitignore` file, if you are going to put any amount of content in it (that you want Git to ignore) you might want to add a single line with just an asterisk `*` to make sure you don't add the ignored content accidentally.

edited Apr 21 '16 at 11:34

GAMITG
3,472 7 28 49

answered Sep 24 '08 at 6:43

Michael Johnson
2,081 15 19

8

There's no way to get Git to track directories, so the only solution is to add a placeholder file within the directory that you want Git to track.

The file can be named and contain anything you want, but most people use an empty file named `.gitkeep` (although some people prefer the VCS-agnostic `.keep`).

answered Apr 26 '15 at 22:54

Zaz
34.4k 10 64 87

313

```
touch .keep
```

On Linux, this creates an empty file named `.keep`. This name is preferred over `.gitkeep` as the former is agnostic to Git, whereas the latter is specific to Git. Secondly, as another user has noted, the `.git` prefix convention should be reserved for files and directories that Git itself uses.

Alternatively, as noted in another answer, the directory can contain a descriptive `README` or `README.md` file instead.

Of course this requires that the presence of the file won't cause your application to break.

edited Jan 30 '15 at 5:39

answered Jan 29 '14 at 4:29

Acumenus
29.1k 8 80 85

1 This is good for an initial bare directory, but what if it starts to fill with files? Then Git will notice them and claim them as untracked files. The selected answer here works far more elegantly to allow one to keep a directory but then safely ignore the contents. – JakeGould Sep 1 '14 at 16:20

13 The question and the predominant general concern is about adding an empty directory. If it later has a resident file, obviously delete the `.keep` file or just disregard it. If instead the files in the directory are to be ignored, that's a different question altogether. – Acumenus Sep 1 '14 at 21:30

3 It was suggested that `git clean -nd | sed s/'^Would remove '// | xargs -I{} touch "{}.keep"` will do this in all untracked empty directories. – Acumenus Oct 7 '14 at 17:16

1 Don't like this solution, it is tough to guess what this file does. Also, if you are generating files in your dev environment (like logs or images, etc.), this isn`t keeping those file from being versioned and making their way into production, which is not nice. – danielrvt May 19 '16 at 16:04

1 Windows doesn't like files without names and requires special magic to accomplish this (aka a bash-like terminal app or equivalent). – EntangledLoops Aug 3 '16 at 19:46

Here is a hack, but it's funny that it works (Git 2.2.1). Similar to what @Teka suggested, but easier to remember:

- This will add a folder and a file `.submodules`. Commit a change.
- Delete `.submodules` file and commit the change.

Now, you have a directory that gets created when commit is checked out. An interesting thing though is that if you look at the content of tree object of this file you'll get:

> fatal: Not a valid object name b64338b90b4209263b50244d18278c0999867193

I wouldn't encourage to use it though since it may stop working in the future versions of Git. Which may leave your repository corrupted.

edited Jan 29 '15 at 18:54

Peter Mortensen
24.1k 19 89 118

answered Dec 24 '14 at 10:24

Stanislav Bashkyrtsev
9,155 4 29 33

3 A **fatal** hack. – John_West Feb 20 '16 at 16:48

7

I always build a function to check for my desired folder structure and build it for me within the project. This gets around this problem as the empty folders are held in Git by proxy.

```
function check_page_custom_folder_structure () {
    if (!is_dir(TEMPLATEPATH."/page-customs"))
        mkdir(TEMPLATEPATH."/page-customs");
    if (!is_dir(TEMPLATEPATH."/page-customs/css"))
        mkdir(TEMPLATEPATH."/page-customs/css");
    if (!is_dir(TEMPLATEPATH."/page-customs/js"))
        mkdir(TEMPLATEPATH."/page-customs/js");
}
```

This is in PHP, but I am sure most languages support the same functionality, and because the creation of the folders is taken care of by the application, the folders will always be there.

edited Jan 29 '15 at 18:48

Peter Mortensen
24.1k 19 89 118

answered Apr 4 '11 at 10:06

Mild Fuzz
23.4k 24 90 138

25 '14 at 15:41

65

Andy Lester is right, but if your directory just needs to be empty, and not *empty* empty, you can put an empty `.gitignore` file in there as a workaround.

As an aside, this is an implementation issue, not a fundamental Git storage design problem. As has been mentioned many times on the Git mailing list, the reason that this has not been implemented is that no one has cared enough to submit a patch for it, not that it couldn't or shouldn't be done.

edited Jan 29 '15 at 18:46

Peter Mortensen
24.1k 19 89 118

answered Sep 22 '08 at 17:28
Aristotle Pagaltzis
91.7k 17 93 94

4 That's exactly what I said. Both paragraphs are addressed in the snippet of FAQ I posted. – Andy Lester Sep 22 '08 at 17:36

1 I think the aside is unteresting and useful to know -- it can be fixed, just don't expect it anytime soon when there's such an easy workaround for most cases. – wnoise Sep 22 '08 at 22:10

2 Of course, this extra answer does serve to point out the fact. – Michael Johnson Sep 24 '08 at 6:44

27

WARNING: This tweak is not truly working as it turns out. Sorry for the inconvenience.

Original post below:

I found a solution while playing with Git internals!

1. Suppose you are in your repository.
2. Create your empty directory:

```
$ mkdir path/to/empty-folder
```

3. Add it to the index using a plumbing command and the empty tree SHA-1:

Type the command and then enter the second line. Press Enter and then Ctrl + D to terminate your input. Note: the format is *mode* [SPACE] *type* [SPACE] SHA-1hash **[TAB]** path (the tab is important, the answer formatting does not preserve it).

4. That's it! Your empty folder is in your index. All you have to do is commit.

This solution is short and apparently works fine (**see the EDIT!**), but it is not that easy to remember...

The empty tree SHA-1 can be found by creating a new empty Git repository, `cd` into it and issue `git write-tree`, which outputs the empty tree SHA-1.

EDIT:

I've been using this solution since I found it. It appears to work exactly the same way as creating a submodule, except that no module is defined anywhere. This leads to errors when issuing `git submodule init|update`. The problem is that `git update-index` rewrites the `040000 tree` part into `160000 commit`.

Moreover, any file placed under that path won't ever be noticed by Git, as it thinks they belong to some other repository. This is nasty as it can easily be overlooked!

However, if you don't already (and won't) use any Git submodules in your repository, and the "empty" folder will remain empty or if you want Git to know of its existence and ignore its content, you can go with this tweak. Going the usual way with submodules takes more steps that this tweak.

edited Dec 23 '14 at 11:35
Peter Mortensen
24.1k 19 89 118

answered Jan 20 '12 at 15:50
ofavre
3,175 1 22 23

2 It's unlikely that this tweak will work with any other tool. Like stated in the warning and the edit, I discourage using it unless in a quite restricted case. – ofavre Sep 2 '14 at 18:15

1 @PyRulez well, in software world, nothing is impossible. :D Actually, I followed the answer. – abhisekp Jan 10 '16 at 16:49

125

As described in other answers, Git is unable to represent empty directories in its staging area. (See the Git FAQ.) However, if, for your purposes, a directory is empty enough if it contains a `.gitignore` file only, then you can create `.gitignore` files in empty directories only via:

edited Dec 23 '14 at 11:18

Peter Mortensen
24.1k 19 89 118

answered May 3 '11 at 15:17

mjs
50.4k 22 76 106

21 You may want to ignore the .git directory: `find . -name .git -prune -o -type d -empty -exec touch {}/.gitignore \;` – steffen Aug 12 '13 at 12:51

2 A simpler variation for most situations is `find * -type d -empty -exec touch {}/.gitignore \;` – akhan Oct 24 '13 at 8:26

2 Since OS X creates a .DS_Store file in almost every directoy, this does not work there. The only (DANGEROUS!) workaround i found, was to delete all the .DS_Store files first via `find . -name .DS_Store -exec rm {} \;` and then use the preferred variant from this answer. Be sure to only execute this in the correct folder! – zerweck Apr 28 '15 at 15:57

1 Does anyone know a way to do this in Windows from the command line? I've seen some solutions here in Ruby and Python, but I'd like a barebones solution if it can be managed. – Mig82 Jan 3 '17 at 17:18

1 @akhan Adding something to `.gitignore` has no influence on the `-empty` flag of the `find` command. My comment is about removing the `.DS_Store` files in a directory tree, so the `-empty` flag can be applied. – zerweck Apr 6 '17 at 11:58

14

I've been facing the issue with empty directories, too. The problem with using placeholder files is that you need to create them, and delete them, if they are not necessary anymore (because later on there were added sub-directories or files. With big source trees managing these placeholder files can be cumbersome and error prone.

This is why I decided to write an open source tool which can manage the creation/deletion of such placeholder files automatically. It is written for .NET platform and runs under Mono (.NET for Linux) and Windows.

Just have a look at: http://code.google.com/p/markemptydirs

edited Jun 22 '14 at 17:38

answered Jul 23 '09 at 22:33

Jonny Dee

692

Create an empty file called `.gitkeep` in the directory, and add that.

edited Oct 9 '13 at 22:21

answered Dec 7 '11 at 16:03

54 I have added an answer encouraging to create `.keep` instead. – Acumenus Jan 29 '14 at 4:31

183 `.gitkeep` has not been prescribed by Git and is going to make people second guess its meaning, which will lead them to google searches, which will lead them here. The `.git` prefix convention should be reserved for files and directories that Git itself uses. – t-mart Feb 10 '14 at 1:44

9 @t-mart "The `.git` prefix convention should be reserved..." Why? Does git request this reservation? – Limited Atonement Aug 28 '14 at 18:13

46 It doesn't. The point is that it can be confusing. – szablica Aug 28 '14 at 23:22

8 In this case a `README` or `ABOUT` file would be just as good or better. Leaving a note for the next guy, just like we all used to do it before URLs. – Dave Nov 15 '14 at 0:59

1055

You can't. See the Git FAQ.

> Currently the design of the git index (staging area) only permits files to be listed, and nobody competent enough to make the change to allow empty directories has cared enough about this situation to remedy it.
>
> Directories are added automatically when adding files inside them. That is, directories never have to be added to the repository, and are not tracked on their own.
>
> You can say " `git add <dir>` " and it will add files in there.
>
> If you really need a directory to exist in checkouts you should create a file in it. .gitignore works well for this purpose; you can leave it empty, or fill in the names of files you expect to show up in the directory.

edited May 4 '12 at 13:12

Gilles 'SO- stop being evil'

82.1k 21 174 213

answered Sep 22 '08 at 16:42

Andy Lester

73.1k 12 83 139

64 Below answer is MUCH better. The fact that git the low level software doesn't allow it doesn't matter to me as much as HOW to actually use Git when I need an empty directory. Adding a 2 line .gitignore seems acceptable to me. – Amala Apr 26 '11 at 15:21

1 Well if one want to move files into a new directory, they can't do it through `git mv` as git will complain that new directory is not under version control – lulalala Nov 2 '11 at 2:58

15 You can read "*it's impossible, you can't, etc.*" all over the Internet for this frequent question. The `.gitignore` trick is a frequent answer, and satisfies

tree, unless it would be impossible to tell whether that object is a tree or a blob. – Emil Lundberg Jul 9 '13 at 9:47

19 I've seen a lot of repos that use an empty file called `.gitkeep` for this purpose. – Sukima Nov 13 '13 at 1:38

1 2 next