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

Asked 16 years, 3 months ago Modified 10 months ago

Viewed 1.6m times

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

5445

git directory git-add

Share

Improve this question

Follow

edited Jul 18, 2022 at 18:41

John Smith

7,399 ● 7 ● 51 ● 63

asked Sep 22, 2008 at 16:41

Laurie Young

138k ● 13 ● 49 ● 55

24 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, 2012 at 14:18

37 Answers

Sorted by: Highest score (default)

1 2 Next

5186

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:

```
# Ignore everything in this directory
*
# 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".

Share Improve this answer

Follow

edited Nov 20, 2018 at 11:59

HelloGoodbye

3,902 ● 9 ● 45 ● 59

answered May 31, 2009 at 22:10

Jamie Flournoy

53.9k ● 1 ● 27 ● 11

48 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, 2013 at 11:11

27 @pedromanoel I write the documentation you would put in the `README` inside the `.gitignore` file (as comments). – Carlos Campderrós Jul 19, 2013 at 8:20

107 spot the 1 difference: 1.) an empty folder, 2.) a folder with .gitignore file in it. ;-) – Peter Perháč Feb 11, 2014 at 14:31

11 This is perfect for *cache* folders. – redolent Mar 26, 2014 at 2:30

24 Unfortunately, this results in a non-empty directory, it has a single hidden file. – pedorro Dec 15, 2014 at 20:09

1230

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.

Share Improve this answer

Follow

edited May 4, 2012 at 13:12

Gilles 'SO- stop being evil'

107k ● 38 ● 215 ● 260

answered Sep 22, 2008 at 16:42

Andy Lester

93.5k ● 15 ● 104 ● 159

Should specify that it's not possible if you consider having a .gitignore *not* empty. Also, if a directory is removed from the remote, pulling will remove the files but not the empty directories locally – MayTheSForceBeWithYou Jul 16, 2023 at 1:29

1159

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

This will be a hidden file on Unix-like systems by default but it will force Git to acknowledge the existence of the

directory since it now has content.

Also note that there is nothing special about this file's name. You could have named it anything you wanted. All Git cares about is that the folder has something in it.

Share Improve this answer

Follow

edited Oct 31, 2022 at 4:29

jpmc26

29.8k ● 14 ● 99 ● 151

answered Dec 7, 2011 at 16:03

Artur79

13.5k ● 1 ● 23 ● 23

333 `.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, 2014 at 1:44

25 @t-mart "The `.git` prefix convention should be reserved..." Why? Does git request this reservation? – lmat - Reinstate Monica Aug 28, 2014 at 18:13

33 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, 2014 at 0:59

14 Doesn't work if you're writing a unit test that should test code on an empty directory... – thebjorn Dec 23, 2015 at 10:22

22 @szablica I don't think it's confusing at all. In fact I think it's very intuitive to call it .gitkeep. Calling it .gitignore is what sounds contradictory to me. So this is just a matter of personal taste. – Mig82 May 15, 2017 at 18:17

567

You could always put a README file in the directory with an explanation of why you want this, otherwise empty, directory in the repository.

Share Improve this answer

Follow

edited Apr 26, 2012 at 2:51

Will

24.6k ● 14 ● 98 ● 110

answered Mar 14, 2011 at 23:38

John Mee

52.1k ● 38 ● 155 ● 196

487

```
touch .placeholder
```

On Linux, this creates an empty file named `.placeholder`. For what it's worth, this name is agnostic to git, and this approach is used in various other places in the system, e.g. `/etc/cron.d/.placeholder`. Secondly, as another user has noted, the `.git` prefix convention can be reserved for files and directories that Git itself uses for configuration purposes.

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

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

Share Improve this answer

edited Nov 14, 2021 at 20:55

Follow

answered Jan 29, 2014 at 4:29

Asclepius

63k ● 19 ● 186 ● 156

3 The reasons to use `.keep` over `.gitkeep` are very good. A good balance between simple and less confusing. Namespaces seem important and it should not be broken that `.git....` looks like git's own file namespace. Additional documentation could go the projects global readme or coding style guides or a local readme. If you have adopted the .gitkeep habit, simply find the files and rename them: `git mv .gitkeep .keep`. Embrace change! The `.gitignore` solution recommended by the git documentation seems a hacky workaround and over complicated. – mit Oct 29, 2018 at 13:54

On Windows, the `touch` command is part of *Git Bash* (the explorer plugin); another option is by using WSL. – Wolf Jan 26, 2023 at 9:57

383

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

Use an *empty* file called `.gitkeep` in order to force the presence of the folder in the versioning system.

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.

Share Improve this answer

Follow

edited Jun 9, 2019 at 9:53

Manu Manjunath

6,381 ● 3 ● 35 ● 32

answered Dec 4, 2013 at 23:32

Cranio

9,807 ● 4 ● 36 ● 55

141

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:

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

Share Improve this answer

Follow

edited Dec 23, 2014 at 11:18

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered May 3, 2011 at 15:17

mjs
65.2k • 27 • 96 • 129

23 You may want to ignore the .git directory: `find . -name .git -prune -o -type d -empty -exec touch {}/.gitignore \;` – steffen Aug 12, 2013 at 12:51

4 A simpler variation for most situations is `find * -type d -empty -exec touch {}/.gitignore \;` – akhan Oct 24, 2013 at 8:26

3 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, 2015 at 15:57

2 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, 2017 at 17:18

2 @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, 2017 at 11:58

83

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.

Share Improve this answer

Follow

edited Jan 29, 2015 at 18:46

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Sep 22, 2008 at 17:28

Aristotle Pagaltzis

118k ● 23 ● 101 ● 100

51

Add a *.gitkeep* file inside the empty directory and commit it.

```
touch .gitkeep
```

.gitkeep is a hidden file, to list it in linux run command

```
ll -a
```

Share Improve this answer

edited Feb 1 at 13:52

Follow

answered Dec 1, 2021 at 11:20

vidur punj

5,843 ● 5 ● 52 ● 71

3 Do you have a reference for "It is the standard followed by Git"? I don't think it is. I think Git doesn't care how the file is named. – mkrieger1 Oct 29, 2022 at 8:30

47

TL;DR: slap a file in the directory and it will be tracked by git. (seriously. that is the official workaround)

But I recommend instead: let a build script or deploy script create the directory on site.

The official way:

Git does not track empty directories. See the official Git FAQ for more detail. The suggested workaround is to put a `.gitignore` file in the empty directory. With the file in place the directory is no longer empty and will be tracked by git.

I do not like that workaround. The file `.gitignore` is meant to ignore things. Here it is used for the opposite: to keep something.

A common workaround (to the workaround) is to name the file `.gitkeep`. This at least conveys the intention in the filename. Also it seems to be a consensus among

some projects. Git itself does not care what the file is named. It just cares if the directory is empty or not.

There is a problem shared by both `.gitkeep` and `.gitignore`: the file is hidden by unix convention. Some tools like `ls` or `cp dir/*` will pretend the file does not exists and behave as if the directory is empty. Other tools like `find -empty` will not. Newbie unix users might get stumped on this. Seasoned unix users will deduce that there are hidden files and check for them. Regardless; this is an avoidable annoyance.

Side note: a simple solution to the "hidden file problematic" is to name the file `gitkeep` (without the leading dot). We can take this one step further and name the file `README`. Then, in the file, explain why the directory needs to be empty and be tracked in git. That way other developers (and future you) can read up why things are the way they are.

Summary: slap a file in the directory and now the (formerly empty) directory is tracked by git.

Potential Problem: the directory is no longer empty.

The assumption is that you need the empty directory because you are collecting files in it to process later. But now you not only have the files you want but also one rogue `.gitignore` or `gitkeep` or what have you. This might complicate simple bash constructs like `for file in dirname/*` because you need to exclude or special case the extra file.

Git does not want to track empty directories. By trying to make git track the empty directory you sacrifice the very thing you were trying to preserve: the empty directory.

Also: spurious "empty" directories in the project directory are problems-to-be. A new developer, or future you, will stumble upon the empty directory and wonder why it needs to be there. You might delete the directory. You might put other files in it. You might even create a second workflow which also uses the empty directory. Then some time in the future it happens that both workflows using the empty directory run simultaneously and all scripts fail because the files make no sense and both teams wonder where the other files have come from.

My recommendation: let a build script or deploy script create the directory on site.

Lets take a few steps back. To before you asked how to make git track an empty directory.

The situation you had then was likely the following: you have a tool that needs an empty directory to work. You want to deploy/distribute this tool and you want the empty directory to also be deployed. Problem: git does not track empty directories.

Now instead of trying to get git to track empty directories lets explore the other options. Maybe (hopefully) you have a deploy script. Let the deploy script create the directory after git clone. Or you have a build script. Let

the build script create the directory after compiling. Or maybe even modify the tool itself to check for and create the directory before use.

If the tool is meant to be used by humans in diverse environments then I would let the tool itself check and create the directories. If you cannot modify the tool, or the tool is used in a highly automatized manner (docker container deploy, work, destroy), then the deploy script would be good place to create the directories.

I think this is the more sensible approach to the problem. Build scripts and deploy scripts are meant to prepare things to run the program. Your tool requires an empty directory. So use those scripts to create the empty directory.

Bonus: the directory is guaranteed to be truly empty when about to be used. Also other developers (and future you) will not stumble upon an "empty" directory in the repository and wonder why it needs to be there.

TL;DR: let the build script or the deploy script create the empty directory on site. or let the tool itself check for and create the directory before use.

The following commands might help you if you inherited a project containing empty or "empty" directories.

To list every empty directory:

```
find -type d -empty
```

Same but avoid looking in the `.git` directory:

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

To list every directory containing a file named `.gitkeep`:

```
find -type f -name .gitkeep
```

To list every directory and the number of files it contains:

```
find -type f -printf "%h\n" | sort | uniq -c |
sort -n
```

Now you can examine all directories containing exactly one file and check if it is a "git keep" file. Note this command does not list directories that are truly empty.

Share Improve this answer

Follow

edited Sep 17, 2023 at 13:00

answered May 6, 2011 at 15:45

Lesmana

26.9k ● 12 ● 83 ● 87

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

34

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:

   ```
   $ git update-index --index-info
   040000 tree
   4b825dc642cb6eb9a060e54bf8d69288fbee4904
   path/to/empty-folder
   ```

   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, 2014 at 11:35

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Jan 20, 2012 at 15:50

ofavre

4,786 ● 2 ● 28 ● 23

After putting the empty folder into the index and committing, is it then possible to `git svn dcommit` it with the desired result? – lmat - Reinstate Monica Aug 28, 2014 at 18:16

5 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, 2014 at 18:15

2 I've created a better solution based on this that doesn't have these drawbacks: stackoverflow.com/a/58543445/277882 – ntninja Oct 24, 2019 at 14:46

33

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

```
$ mkdir tmp
$ touch tmp/.gitignore
$ git add tmp
$ echo '*' > tmp/.gitignore
$ git commit -m 'Empty directory' 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).

Share Improve this answer

Follow

edited Apr 21, 2016 at 11:34

GAMITG

3,818 ● 7 ● 34 ● 51

answered Oct 8, 2008 at 0:13

m104

1,126 ● 7 ● 6

14 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, 2010 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, 2014 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, 2015 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. – Seldom 'Where's Monica' Needy May 25, 2016 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, 2016 at 8:37

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.

Share Improve this answer

Follow

edited Oct 26, 2018 at 20:52

rogerdpack

66.5k • 39 • 282 • 401

answered Oct 22, 2012 at 13:24

Thomas E

3,828 • 2 • 21 • 13

32 What does that have to do with Ruby on Rails? – Quolonel Questions Sep 29, 2015 at 9:11

@QuolonelQuestions github.com/rails/rails/blob/master/activerecord/test/migrations/… – Resigned June 2023 Apr 29, 2017 at 1:15

1 It has now been renamed to `.keep` github.com/rails/rails/blob/main/activerecord/test/migrations/… – David Jul 27, 2021 at 4:27

25

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.

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, 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
  even if they're empty.

  Read more about why this is necessary at: https://gi
  x.php/Git_FAQ#Can_I_add_empty_directories.3F

Options:
  -r, --recursive     Add or remove the .gitkeep files
                      sub-directories in the specified
  -l, --let-go        Remove the .gitkeep files from t
  -e, --empty         Create empty .gitkeep files. Thi
```

```
                        message provided
  -m, --message TEXT    A message to be included in the
                        used to explain why it's importa
                        directory to source control even
  -v, --verbose         Print out everything.
  --help                Show this message and exit.
```

Share Improve this answer

Follow

edited Oct 28, 2022 at 22:28

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered May 15, 2017 at 19:08

Mig82

5,446 ● 4 ● 45 ● 68

for what it's worth, a windows powershell one-liner could be `ls -r -dir | ?{$_.getFileSystemInfos().Count -eq 0} | %{ni -p $_.FullName -n .gitkeep}` – Gregor y May 31, 2022 at 15:16

22

Maybe adding an empty directory seems like it would be the *path of least resistance* because you have scripts that expect that directory to exist (maybe because it is a target for generated binaries). Another approach would be to **modify your scripts to create the directory as needed**.

```
mkdir --parents .generated/bin ## create a folder
for storing generated binaries
mv myprogram1 myprogram2 .generated/bin ##
populate the directory as needed
```

In this example, you might check in a (broken) symbolic link to the directory so that you can access it without the

".generated" prefix (but this is optional).

```
ln -sf .generated/bin bin
git add bin
```

When you want to clean up your source tree you can just:

```
rm -rf .generated ## this should be in a "clean"
script or in a makefile
```

If you take the oft-suggested approach of checking in an almost-empty folder, you have the minor complexity of deleting the contents without also deleting the ".gitignore" file.

You can ignore all of your generated files by adding the following to your root .gitignore:

```
.generated
```

Share Improve this answer

Follow

answered Oct 26, 2011 at 16:33

Brent Bradburn

54.7k ● 19 ● 162 ● 184

1 Note: The symbolic link that I suggested is "broken" in a clean checkout because the `.generated` directory does not initially exist. It will no longer be broken once you do your build. – Brent Bradburn Mar 14, 2012 at 0:14

3 I agree in some cases this is a very good idea, but in others (such as distributing a project where you have an otherwise empty skeleton with folders such as models/ and views/) you

would want the user to have these directories at hand rather than manually having to read read the docs, and it could be a bit much to expect them to run some sort of installation script after cloning the repo. I think this answer in combination with @john-mee's README answer should cover most if not all cases. – moopet Jun 17, 2014 at 8:28

17

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.

Share Improve this answer

Follow

edited Apr 21, 2016 at 11:35

GAMITG

3,818 ● 7 ● 34 ● 51

answered Mar 15, 2015 at 18:17

user2334883

411 ● 4 ● 4

2 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, 2015 at 8:32

3 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, 2017 at 20:12

Patrick, he also uses the word "idiotic" there. I suspect his wording adresses the people here in this thread and so I assume he will not implement something "idiotic" into Git by himself. – user2334883 Aug 3, 2017 at 20:38

17

This solution worked for me.

1. Add a `.gitignore` file to your empty directory:

```
*
*/
!.gitignore
```

- `*` ignore all files in the folder
- `*/` Ignore subdirectories
- `!.gitignore` include the .gitignore file

2. Then remove your cache, stage your files, commit and push:

```
git rm -r --cached .
git add . // or git stage .
git commit -m ".gitignore fix"
git push
```

Share Improve this answer

Follow

edited Jun 18, 2021 at 13:23

answered Sep 3, 2020 at 15:04

DevonDahon

8,320 ● 9 ● 88 ● 142

15

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

Share Improve this answer

Follow

edited Jun 22, 2014 at 17:38

answered Jul 23, 2009 at 22:33

Jonny Dee

13

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.

Share Improve this answer

Follow

edited Apr 21, 2016 at 11:34

GAMITG

3,818 ● 7 ● 34 ● 51

answered Sep 24, 2008 at 6:43

Michael Johnson

2,307 ● 16 ● 21

13

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.
```

To convert all existing empty directories to submodules:

```
find . -type d -empty -delete -exec git submodule add
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 co
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
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 +00

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

Creating reproducible Git commits is surprisingly hard…

Share Improve this answer

Follow

edited Oct 28, 2022 at 22:26

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Oct 24, 2019 at 14:23

ntninja

1,325 ● 17 ● 23

I've gone the path of getting really dirty in the git codebase in 2012: github.com/git/git/compare/master...ofavre:git:permdir. Sorry I don't remember the state of this project. I'm pretty sure that, besides being non standard, probably never accepted upstream because they are (were) really into tracking *content*, there may have been a few bugs left. The way basically it worked in the CLI was `git add folder/`

with the trailing `/` adds a permanent directory. – ofavre Dec 26, 2023 at 8:24

code `git submodule add --force https://gitlab.com/empty-repo/empty.git repo/a` makes new file `repo/a/.git` wich contains `gitdir: ../../../../../../../../.git/modules/repo/a`. So your way is wrong? – Андрей Тернити Aug 19 at 10:06

11

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`).

The prefixed `.` marks it as a hidden file.

Another idea would be to add a `README` file explaining what the directory will be used for.

Share Improve this answer

Follow

answered Apr 26, 2015 at 22:54

Zaz

48.6k ● 15 ● 88 ● 105

10

A PowerShell version:

> 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($_.Full
-eq 0} | ForEach-Object { New-Item ($_.FullName +
"\.gitkeep") -ItemType file}
```

Share Improve this answer

Follow

edited Oct 28, 2022 at 22:12

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Aug 13, 2019 at 9:34

Hainan Zhao

2,082 ● 20 ● 19

also, `ni -p $_.FullName -n .gitignore -v "*`r`n!.gitignore"` – Gregor y May 31, 2022 at 15:29

9

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.

Share Improve this answer

Follow

edited Apr 21, 2016 at 11:34

GAMITG

3,818 ● 7 ● 34 ● 51

answered Apr 19, 2011 at 14:10

Peter Hoeg

901 ● 9 ● 13

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, 2011 at 6:35

9

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

```
# Ignore everything in this directory
*
# Except this file
!.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.

Share Improve this answer

edited May 23, 2017 at 12:02

Follow

Community Bot
1 ● 1

answered Jun 22, 2014 at 13:06

Roman
2,549 ● 2 ● 30 ● 53

2 He wants to keep a empty directory and not a file.
– gvsrepins Jul 29, 2014 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, 2014 at 8:17

If you have a .htaccess file that's under version control, then you already have the directory containing it under version control. Thus, the problem is already solved - the .gitignore file becomes irrelevant. – Ponkadoodle Jan 8, 2017 at 9:11

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, 2017 at 8:10

8

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.

Share Improve this answer

Follow

edited Jan 29, 2015 at 18:48

Peter Mortensen

31.6k ● 22 ● 109 ● 133

answered Apr 4, 2011 at 10:06

Mild Fuzz

30.6k ● 34 ● 104 ● 151

4 Just so we're all on the same page, I do not do this anymore. It's a waste of time. The `.gitkeep` convention is a much better practise. – Mild Fuzz Mar 25, 2014 at 15:41

I can't see how this can be a waste of time. When your TEMPLATEPATH is obviously dynamic you can't use the .gitkeep solution. And even with a nondynamic folder

structure you should add some more stuff instead of removing the very good solution of checking directories e.g. check for permissions and chmod the files. Adding a way to mark directories inside a global .gitignore would be perfect for me. Something like #keep /path/to/dir – Jochen Schultz Jun 2, 2015 at 13:58

8

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

- Add a submodule to any repository (`git submodule add path_to_repo`)
- 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.

Share Improve this answer edited Jan 29, 2015 at 18:54

Follow

Peter Mortensen
31.6k ● 22 ● 109 ● 133

answered Dec 24, 2014 at 10:24

Stanislav Bashkyrtsev
15.3k ● 7 ● 44 ● 49

This actually works but I think confuses the heck out of IntelliJ... :| – rogerdpack Jun 8, 2018 at 18:09

I've created a better solution based on this that doesn't have these drawbacks: stackoverflow.com/a/58543445/277882 – ntninja Oct 24, 2019 at 14:46

8

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...

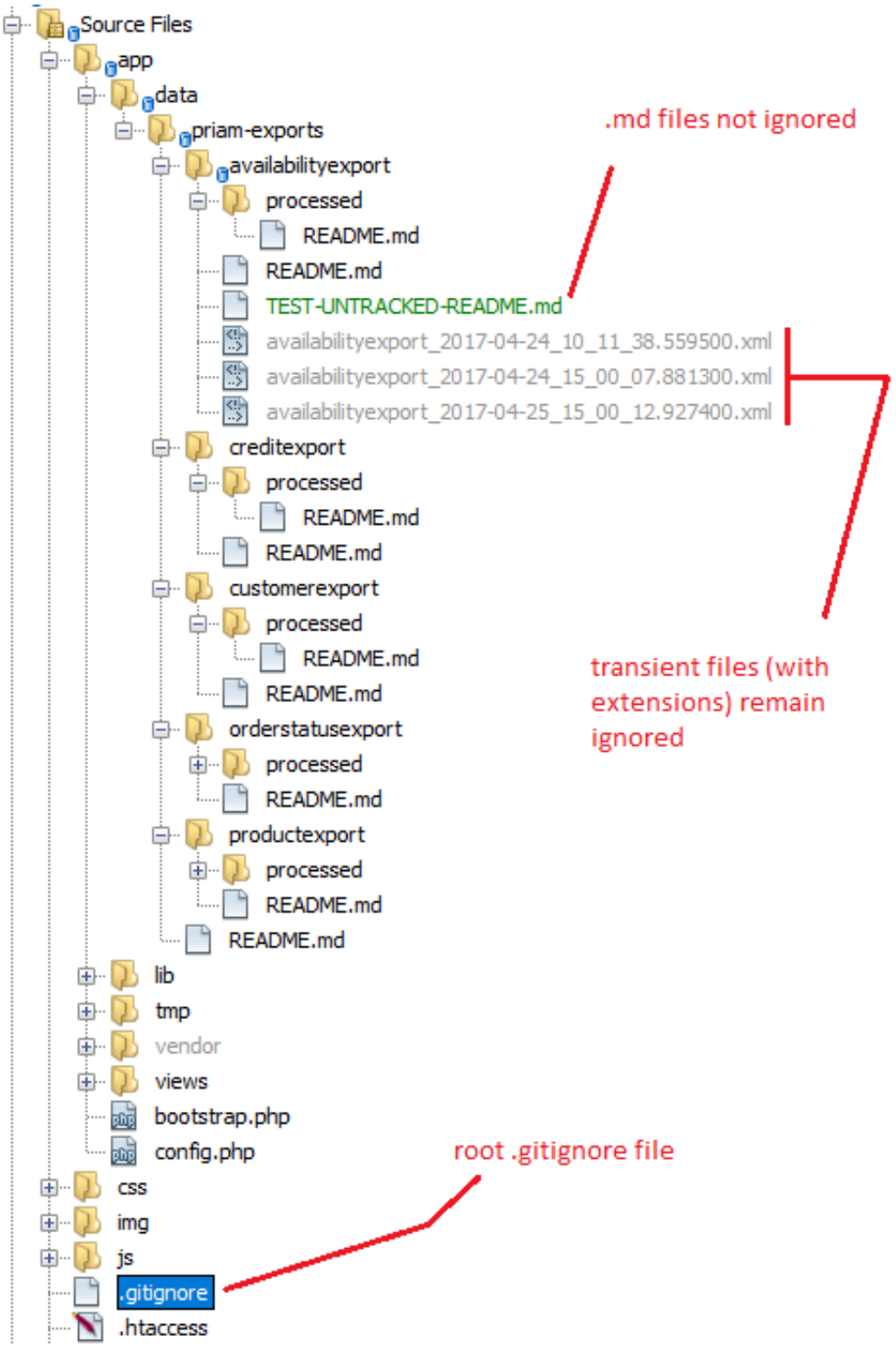
```
/app/data/**/*.*
!/app/data/**/*.md
```

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! ;)



Share Improve this answer Follow

edited May 11, 2017 at 14:14

answered May 11, 2017 at 13:36

ajmedway
1,492 ● 14 ● 29

7

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...

First create the needed directory:

```
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...)

Share Improve this answer

Follow

answered Jun 2, 2016 at 16:42

Trendfischer

7,622 • 7 • 43 • 53

7

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

See this SOF answer for further info - which also explains why some people find the competing convention of adding a .gitignore file (as stated in many answers here) confusing.

Share Improve this answer

Follow

answered Jun 17, 2019 at 5:03

arcseldon

37k • 17 • 125 • 126

5

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 not

ignore *me*", but otherwise, the directory is empty and tracked.

Your `.gitignore` file would look like:

```
!.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.

Share Improve this answer

Follow

edited May 28, 2016 at 16:38

answered May 26, 2016 at 1:18

Mike

1,988 ● 18 ● 35

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.