Gitcd Documentation

Release 1.6.16

Claudio Walser

Jun 04, 2018

Contents:

CHAPTER 1

Continuous tool for working with git

Development Status **Package Info**

1.1 Description

gitcd is a little helper for continuous delivery workflows, using git as scm.

1.2 Installation of gitcd

1.2.1 Pre requisites

Gitcd is written in Python3. Most systems still deliver with Python2 as default. You need to install Python3 in order to run gitcd properly.

MacOSX

```
brew install python3
```

Ubuntu / Debian

```
sudo apt-get install python3 python3-pip
```

1.2.2 Installation of gitcd itself

Now you are ready to install gitcd itself, which is quite easy using pip.

```
pip3 install --user --upgrade gitcd
```

1.2.3 Trouble using git-cd?

If the command “git-cd” or “git cd” is not available now, you probably need to add the pip binary path to your $PATH variable.

MacOSX

Open ~/.bash_profile in your favorite editor and add the following lines at the end of the file.

Replace <python-version> with your currently installed python version

```
if [ -d "$HOME/Library/Python/<python-version>/bin" ] ; then
    PATH="$HOME/Library/Python/<python-version>/bin:$PATH"
fi
```

Ubuntu / Debian

Open ~/.profile in your favorite editor and add the following lines at the end of the file.

```
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi
```

1.2.4 Argument Completion

Gitcd supports argument completion, to activate it execute the following steps.

MacOSX

Under OSX it isn’t that simple unfortunately. Global completion requires bash support for complete -D, which was introduced in bash 4.2. On OS X or older Linux systems, you will need to update bash to use this feature. Check the version of the running copy of bash with echo $BASH_VERSION. On OS X, install bash via Homebrew (brew install bash), add /usr/local/bin/bash to /etc/shells, and run chsh to change your shell.

You might consider reading the docs for argcomplete https://argcomplete.readthedocs.io/en/latest/#global-completion

Activate Global argcomplete

You are now ready to activate global argcompletion for python with the following command.

```
activate-global-python-argcomplete
```

1.3 CLI Usage of gitcd

For convenience, you can call gitcd as a git sub command as well as directly. Therefore, you can replace "git cd" in any of the following commands with "git-cd" if you like it more.

Note: Python argument completion wont work if you use it as a git sub command!

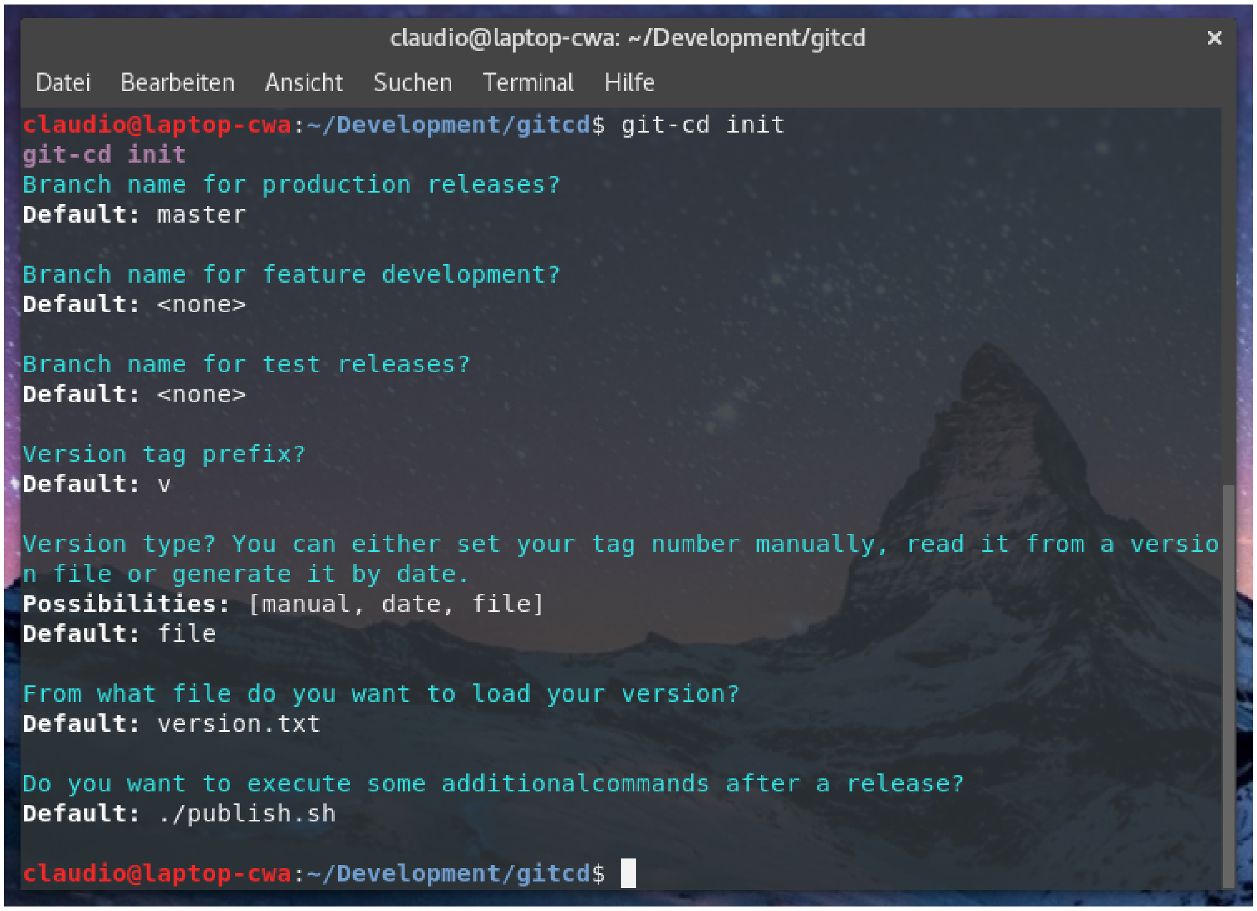
1.3.1 Initializing gitcd

First of all you probably want to initialize one of your local git repositories with gitcd. Change directory to one of your local git repositories and run git-cd init. Most of the values should be very self-explanatory. Still, here is a complete list of values you can pass.

- **Branch name for production releases?**
  - This is the branch git-cd is creating a tag from if you execute the release command, you probably want to go with **master** here.
- **Branch name for feature development?**
  - This is more kind of a prefix for feature branches, it is empty by default. If you wish your feature branch has a name like feature/my-new-feature, you can set this prefix to **feature/**.
- **Branch name for test releases?**
  - Pass your branch name where you want to merge code into while executing git-cd test. Let it empty if you don't want to use that feature. At work, we have this for many repositories set to **test**.
- **Version tag prefix?**
  - Prefix for your release tags, this is **v** by default which would result in a tag equals to v0.0.1 for example.
- **Version type? You can either set your tag number manually, read it from a version file or generate it by date.**
  - This is about how git-cd release gets your current version number you want to release.
    * manual means you'll get asked to enter the version number by hand
    * file means gitcd reads the version number from a file, you'll be asked from which file in the next step
    * date means you generate a version number from a date scheme, you'll be asked for the scheme later. As a date version scheme, you can pass any directive for http://strftime.org/.
- **Do you want to execute some additional commands after a release?**
  - This is useful if you want to execute any cli script after creating a tag, for example, gitcd itself uses such a script to publish the new release on pypi after creating a new tag. You can see the script here https://github.com/gitcd-io/gitcd/blob/master/publish.sh.

```
git cd init
```

The image below represents the configuration for gitcd itself.

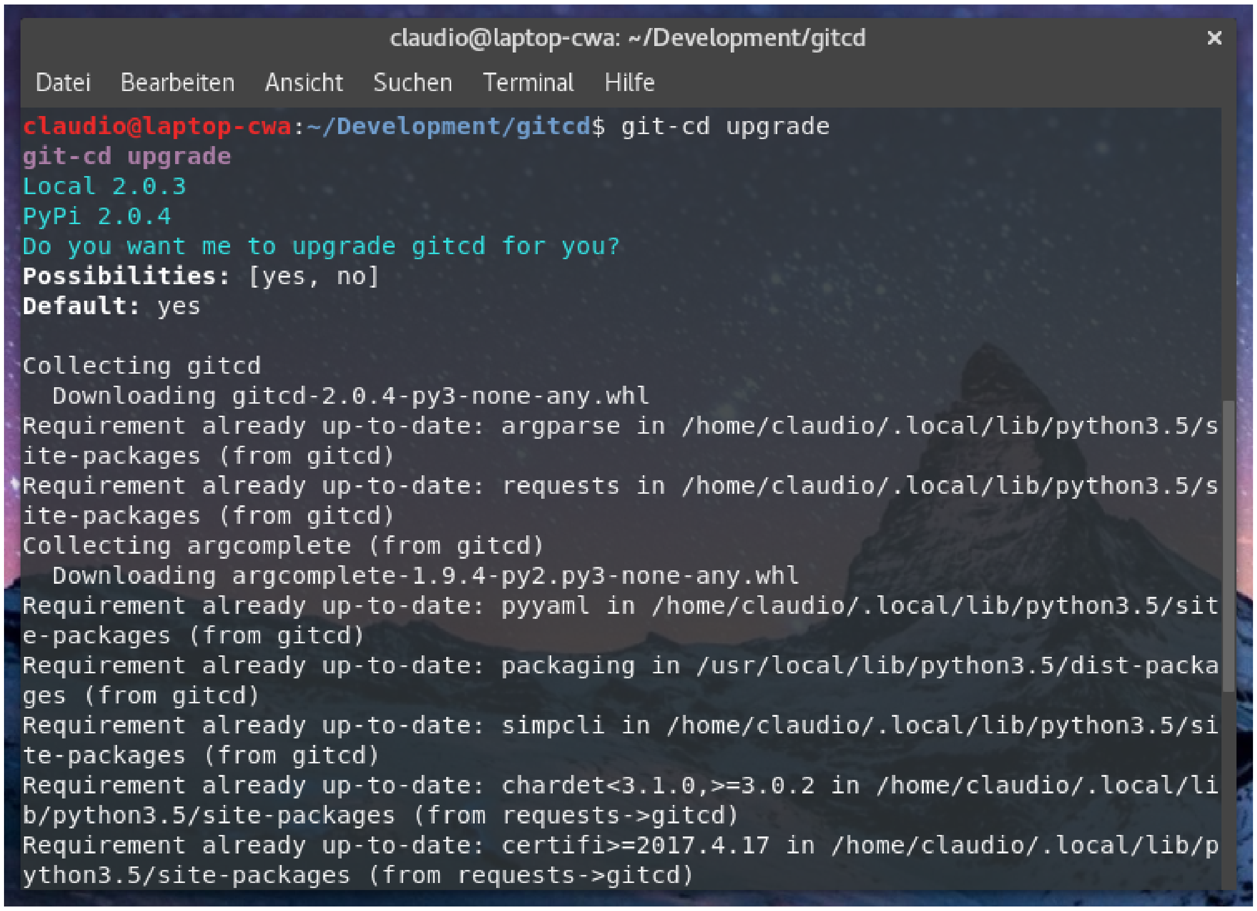


```
claudio@laptop-cwa:~/Development/gitcd$ git-cd init
git-cd init
Branch name for production releases?
Default: master

Branch name for feature development?
Default: <none>

Branch name for test releases?
Default: <none>

Version tag prefix?
Default: v

Version type? You can either set your tag number manually, read it from a versio
n file or generate it by date.
Possibilities: [manual, date, file]
Default: file

From what file do you want to load your version?
Default: version.txt

Do you want to execute some additionalcommands after a release?
Default: ./publish.sh

claudio@laptop-cwa:~/Development/gitcd$
```

1.3.2 Check version and upgrade

Gitcd is able to check your local version with the one published on pypi and upgrade itself if you wish so.

```
git cd upgrade
```

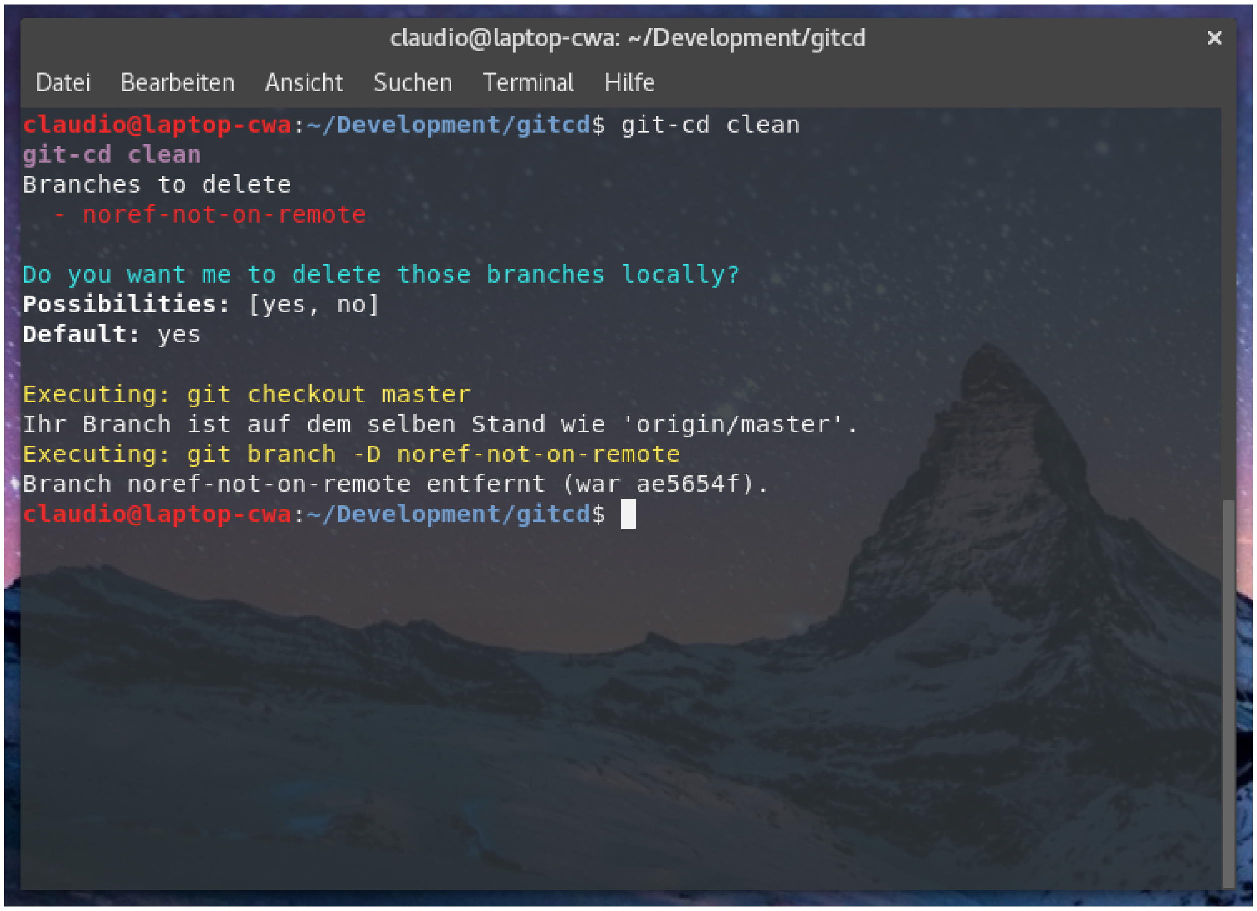

claudio@laptop-cwa: ~/Development/gitcd
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
claudio@laptop-cwa:~/Development/gitcd$ git-cd upgrade
git-cd upgrade
Local 2.0.3
PyPi 2.0.4
Do you want me to upgrade gitcd for you?
Possibilities: [yes, no]
Default: yes

Collecting gitcd
  Downloading gitcd-2.0.4-py3-none-any.whl
Requirement already up-to-date: argparse in /home/claudio/.local/lib/python3.5/s
ite-packages (from gitcd)
Requirement already up-to-date: requests in /home/claudio/.local/lib/python3.5/s
ite-packages (from gitcd)
Collecting argcomplete (from gitcd)
  Downloading argcomplete-1.9.4-py2.py3-none-any.whl
Requirement already up-to-date: pyyaml in /home/claudio/.local/lib/python3.5/sit
e-packages (from gitcd)
Requirement already up-to-date: packaging in /usr/local/lib/python3.5/dist-packa
ges (from gitcd)
Requirement already up-to-date: simpcli in /home/claudio/.local/lib/python3.5/si
te-packages (from gitcd)
Requirement already up-to-date: chardet<3.1.0,>=3.0.2 in /home/claudio/.local/li
b/python3.5/site-packages (from requests->gitcd)
Requirement already up-to-date: certifi>=2017.4.17 in /home/claudio/.local/lib/p
ython3.5/site-packages (from requests->gitcd)
```

1.3.3 Clean up local branches

The tool is able to cleanup all local branches which doesn't exist on remotes. This is done with the clean command.

```
git cd clean
```

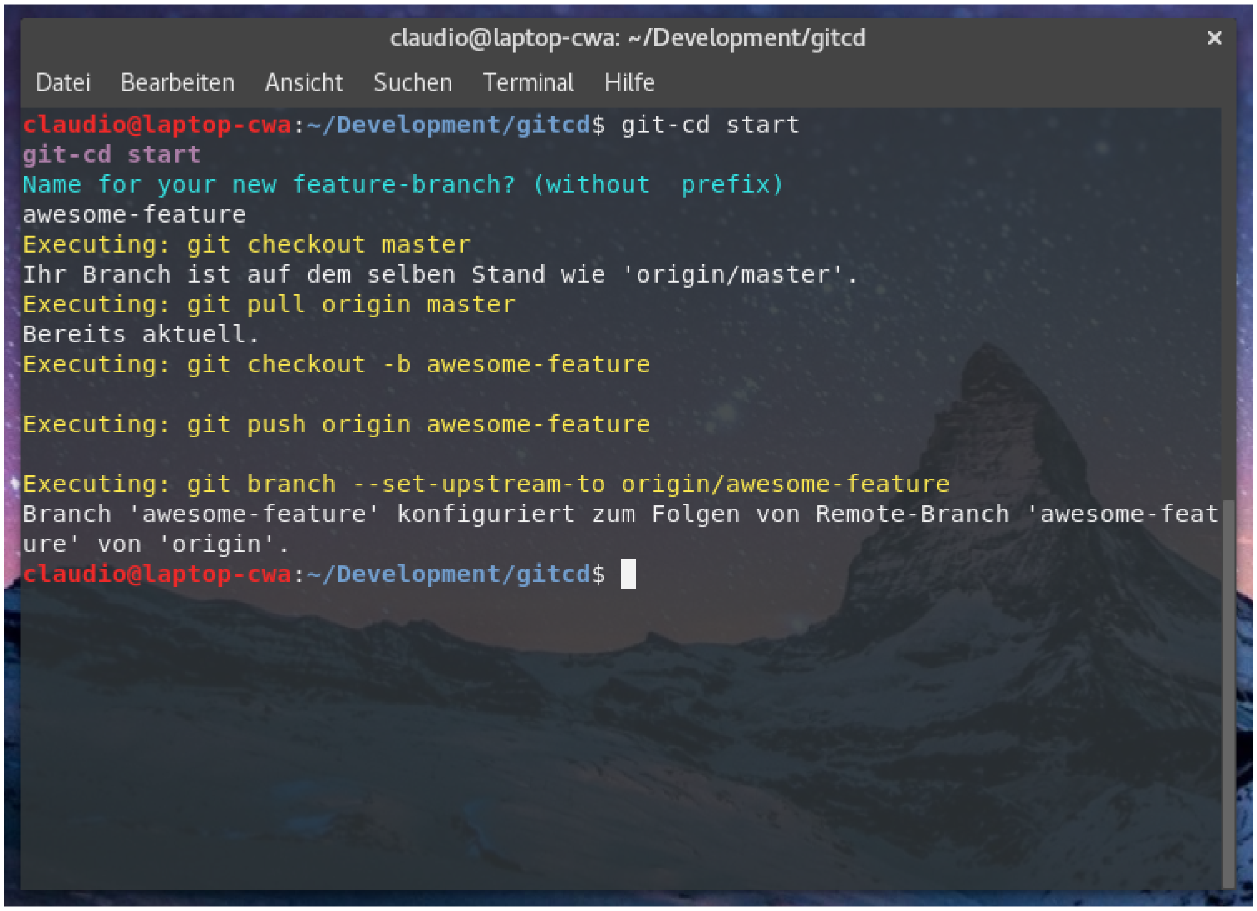

claudio@laptop-cwa: ~/Development/gitcd
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
claudio@laptop-cwa:~/Development/gitcd$ git-cd clean
git-cd clean
Branches to delete
  - noref-not-on-remote

Do you want me to delete those branches locally?
Possibilities: [yes, no]
Default: yes

Executing: git checkout master
Ihr Branch ist auf dem selben Stand wie 'origin/master'.
Executing: git branch -D noref-not-on-remote
Branch noref-not-on-remote entfernt (war ae5654f).
claudio@laptop-cwa:~/Development/gitcd$
```

1.3.4 Start a new feature

Starts a new feature branch from your master branch. If you don't pass a branch name, you will be asked later.

```
git cd start <branchname>
```

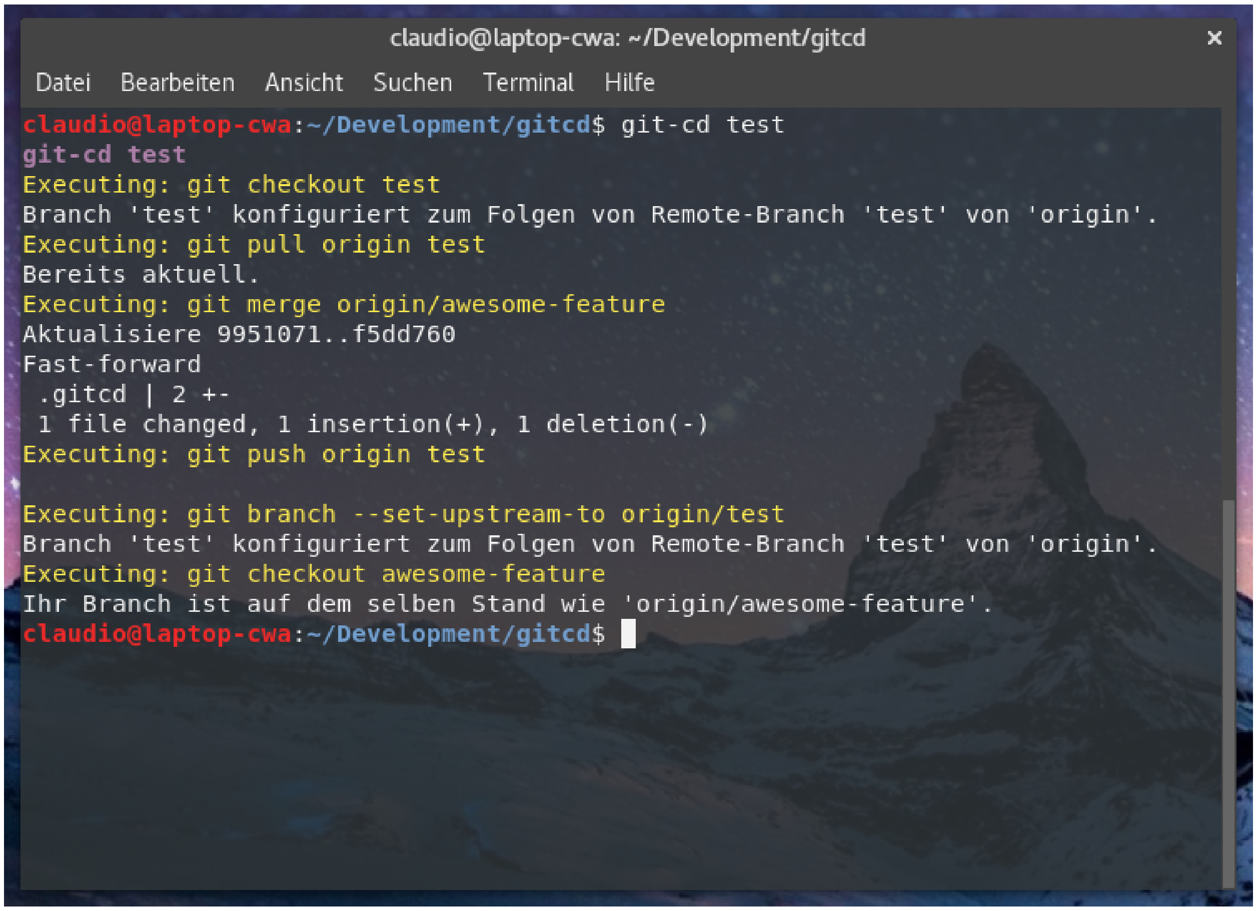

claudio@laptop-cwa: ~/Development/gitcd
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
claudio@laptop-cwa:~/Development/gitcd$ git-cd start
git-cd start
Name for your new feature-branch? (without  prefix)
awesome-feature
Executing: git checkout master
Ihr Branch ist auf dem selben Stand wie 'origin/master'.
Executing: git pull origin master
Bereits aktuell.
Executing: git checkout -b awesome-feature

Executing: git push origin awesome-feature

Executing: git branch --set-upstream-to origin/awesome-feature
Branch 'awesome-feature' konfiguriert zum Folgen von Remote-Branch 'awesome-feat
ure' von 'origin'.
claudio@laptop-cwa:~/Development/gitcd$
```

1.3.5 Testing a feature

You might have a testing environment or want to run some integration test on a shared or common branch without the need to push out your feature with the next release. Therefore you can't merge it into the master. That's exactly why the git-cd test command exists. You might even have some dedicated tester checking the new feature on this specific branch. So to merge your new feature into your testing branch you call this command, if you don't pass a branch name, your current feature branch will be merged.

```
git cd test <branchname>
```

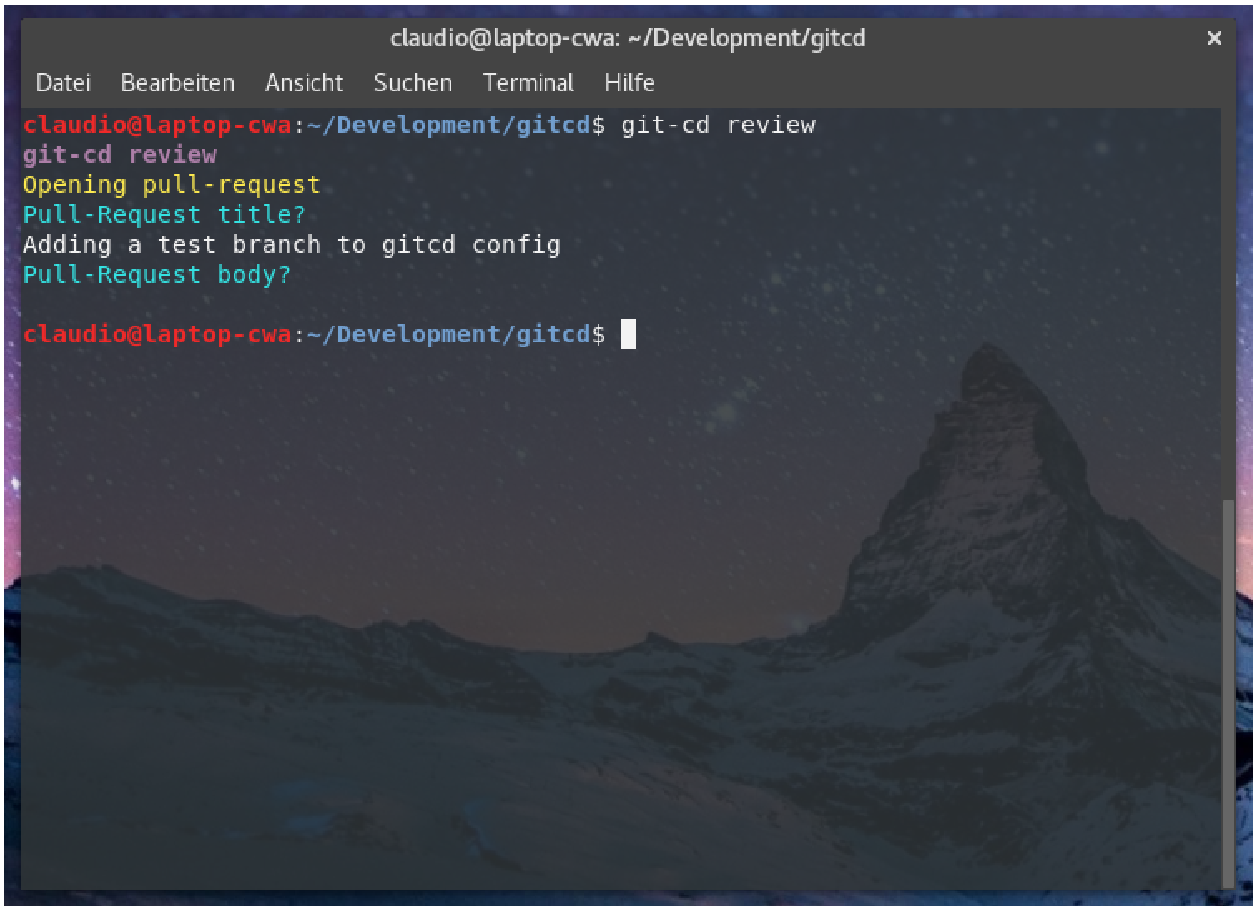

claudio@laptop-cwa: ~/Development/gitcd
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
claudio@laptop-cwa:~/Development/gitcd$ git-cd test
git-cd test
Executing: git checkout test
Branch 'test' konfiguriert zum Folgen von Remote-Branch 'test' von 'origin'.
Executing: git pull origin test
Bereits aktuell.
Executing: git merge origin/awesome-feature
Aktualisiere 9951071..f5dd760
Fast-forward
 .gitcd | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
Executing: git push origin test

Executing: git branch --set-upstream-to origin/test
Branch 'test' konfiguriert zum Folgen von Remote-Branch 'test' von 'origin'.
Executing: git checkout awesome-feature
Ihr Branch ist auf dem selben Stand wie 'origin/awesome-feature'.
claudio@laptop-cwa:~/Development/gitcd$
```

1.3.6 Open a pull request for code review

Opens a pull request to your master branch. If you don't pass a branch name, your current branch will be taken.

```
git cd review <branchname>
```

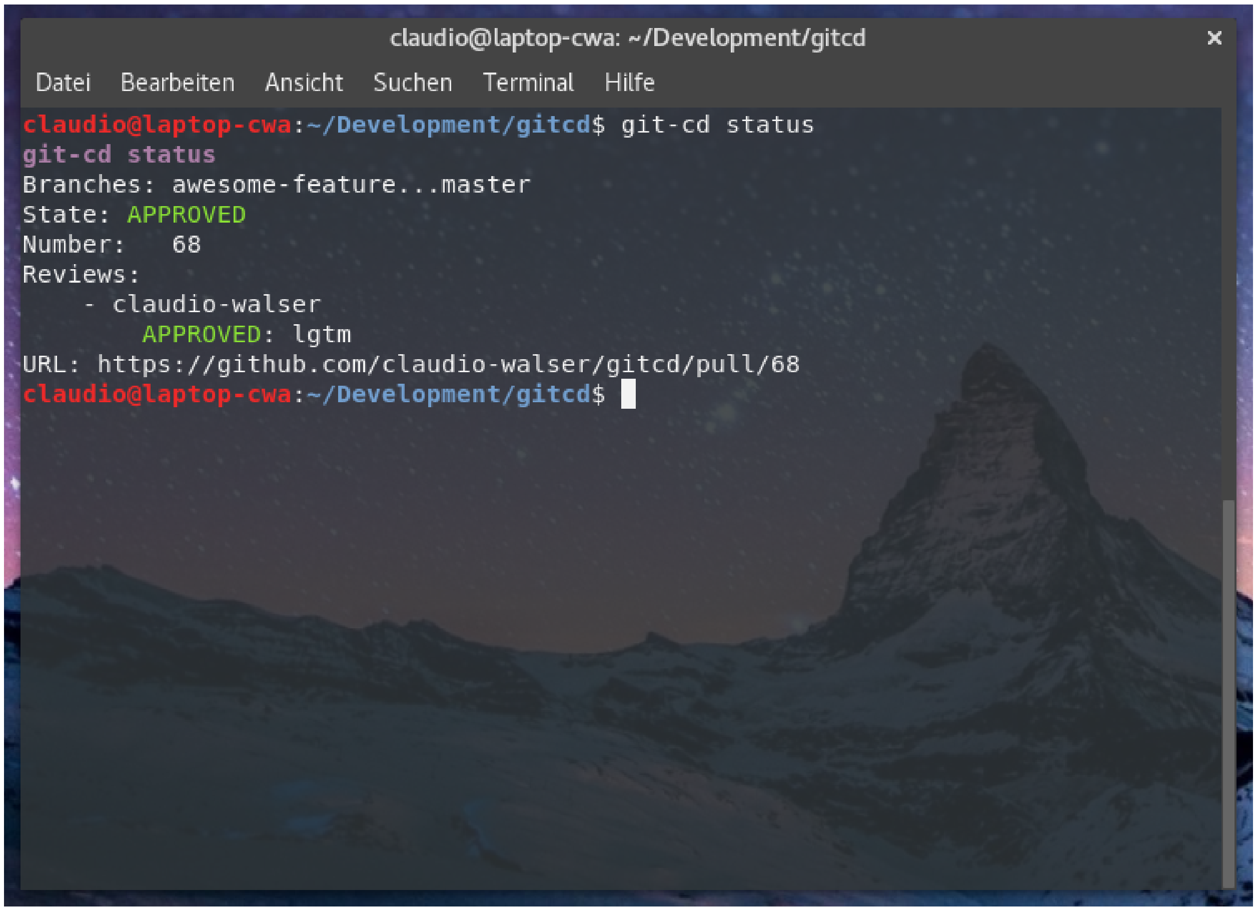


```
claudio@laptop-cwa:~/Development/gitcd$ git-cd review
git-cd review
Opening pull-request
Pull-Request title?
Adding a test branch to gitcd config
Pull-Request body?

claudio@laptop-cwa:~/Development/gitcd$ 
```

1.3.7 See the status of a pull request

You can see the status of a pull request directly in the command line. If you don't pass a branch name, your current branch will be taken.

```
git cd status <branchname>
```

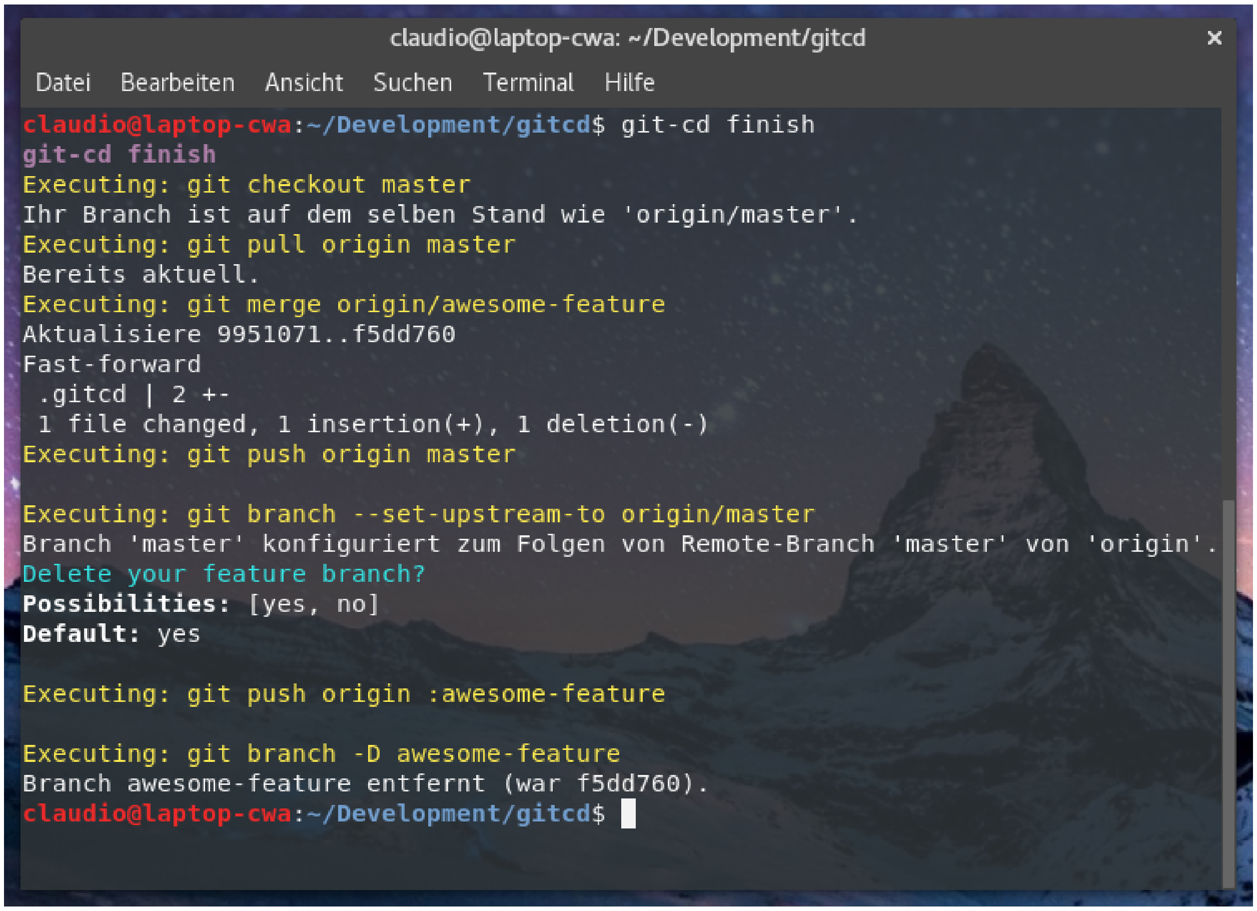

claudio@laptop-cwa: ~/Development/gitcd
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
claudio@laptop-cwa:~/Development/gitcd$ git-cd status
git-cd status
Branches: awesome-feature...master
State: APPROVED
Number:   68
Reviews:
    - claudio-walser
        APPROVED: lgtm
URL: https://github.com/claudio-walser/gitcd/pull/68
claudio@laptop-cwa:~/Development/gitcd$
```

1.3.8 Finish a feature branch

If your pull request got approved by a fellow developer and all your tests were running properly, you probably want to merge your feature into the master branch. If you don't pass a branch name, your current branch will be taken.

```
git cd finish <branchname>
```

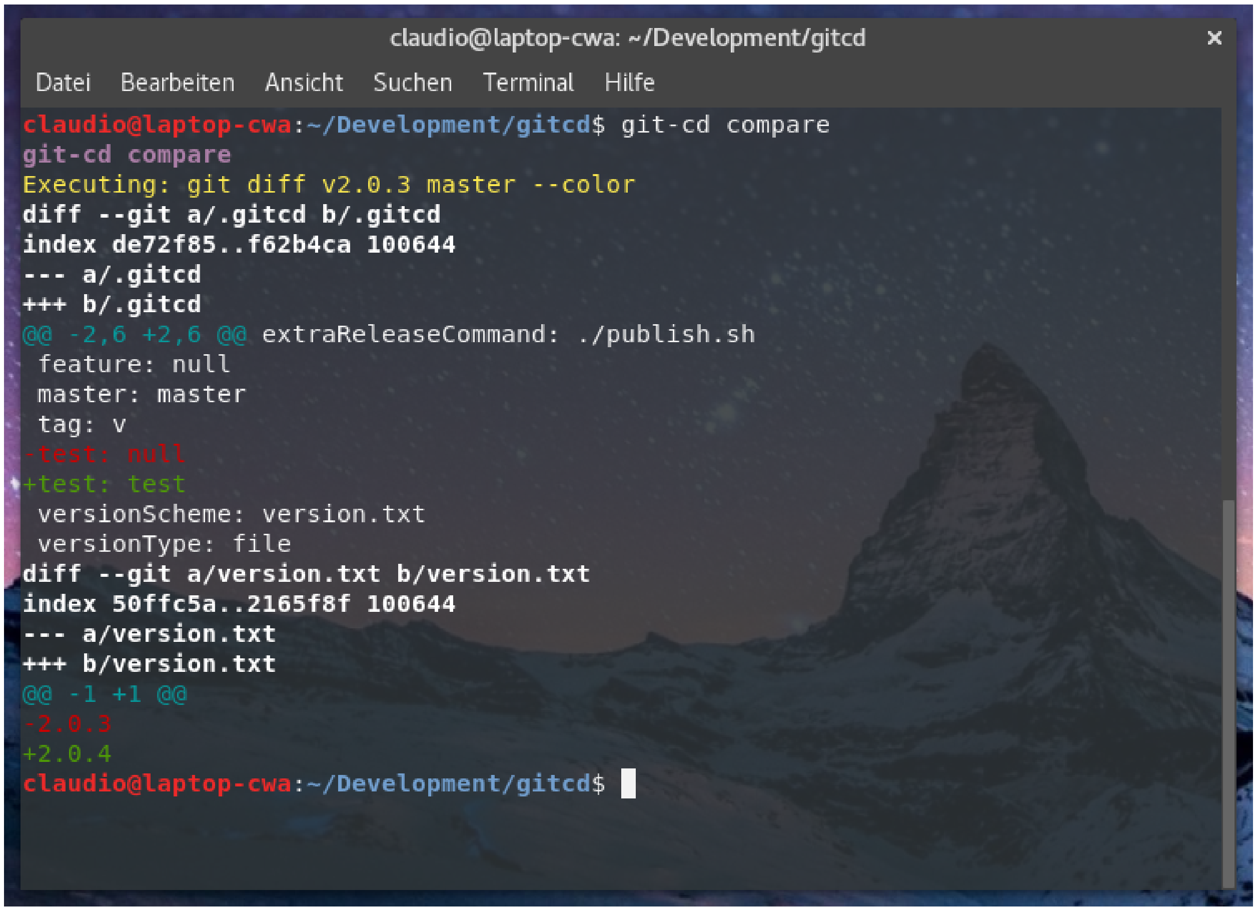

claudio@laptop-cwa: ~/Development/gitcd
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
claudio@laptop-cwa:~/Development/gitcd$ git-cd finish
git-cd finish
Executing: git checkout master
Ihr Branch ist auf dem selben Stand wie 'origin/master'.
Executing: git pull origin master
Bereits aktuell.
Executing: git merge origin/awesome-feature
Aktualisiere 9951071..f5dd760
Fast-forward
 .gitcd | 2 +-
 1 file changed, 1 insertion(+), 1 deletion(-)
Executing: git push origin master

Executing: git branch --set-upstream-to origin/master
Branch 'master' konfiguriert zum Folgen von Remote-Branch 'master' von 'origin'.
Delete your feature branch?
Possibilities: [yes, no]
Default: yes

Executing: git push origin :awesome-feature

Executing: git branch -D awesome-feature
Branch awesome-feature entfernt (war f5dd760).
claudio@laptop-cwa:~/Development/gitcd$
```

1.3.9 Compare different branches or tags

By now, your code is in the master branch. Personally, I always like to see what I am going to release by comparing the current branch (which is master after the finish) against the latest tag. If you don't pass a branch or tag name, the latest tag will be taken.

```
git cd compare <branchname>||<tagname>
```

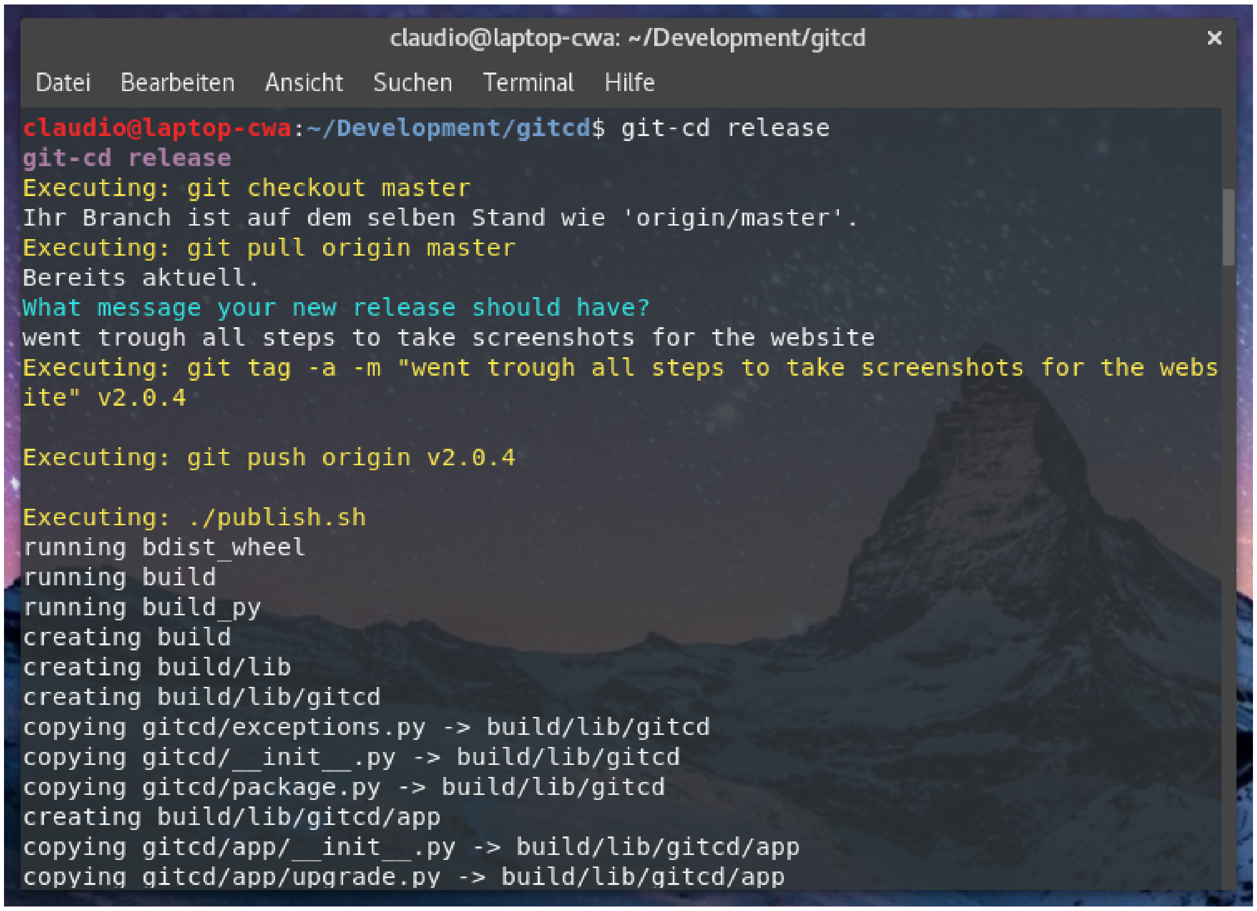

claudio@laptop-cwa: ~/Development/gitcd
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
claudio@laptop-cwa:~/Development/gitcd$ git-cd compare
git-cd compare
Executing: git diff v2.0.3 master --color
diff --git a/.gitcd b/.gitcd
index de72f85..f62b4ca 100644
--- a/.gitcd
+++ b/.gitcd
@@ -2,6 +2,6 @@ extraReleaseCommand: ./publish.sh
 feature: null
 master: master
 tag: v
-test: null
+test: test
 versionScheme: version.txt
 versionType: file
diff --git a/version.txt b/version.txt
index 50ffc5a..2165f8f 100644
--- a/version.txt
+++ b/version.txt
@@ -1 +1 @@
-2.0.3
+2.0.4
claudio@laptop-cwa:~/Development/gitcd$
```

1.3.10 Release a new version

Now your feature is merged and you made sure you know the changes going out, you are ready to ship it. This command creates a new tag from the master branch and executes any command you've setup in the initialize command.

```
git cd release
```


claudio@laptop-cwa: ~/Development/gitcd
Datei Bearbeiten Ansicht Suchen Terminal Hilfe

```
claudio@laptop-cwa:~/Development/gitcd$ git-cd release
git-cd release
Executing: git checkout master
Ihr Branch ist auf dem selben Stand wie 'origin/master'.
Executing: git pull origin master
Bereits aktuell.
What message your new release should have?
went trough all steps to take screenshots for the website
Executing: git tag -a -m "went trough all steps to take screenshots for the webs
ite" v2.0.4

Executing: git push origin v2.0.4

Executing: ./publish.sh
running bdist_wheel
running build
running build_py
creating build
creating build/lib
creating build/lib/gitcd
copying gitcd/exceptions.py -> build/lib/gitcd
copying gitcd/__init__.py -> build/lib/gitcd
copying gitcd/package.py -> build/lib/gitcd
creating build/lib/gitcd/app
copying gitcd/app/__init__.py -> build/lib/gitcd/app
copying gitcd/app/upgrade.py -> build/lib/gitcd/app
```

1.4 Known Issues

If you discover any bugs, feel free to create an issue on GitHub or fork this repository and send us a pull request.

Issues List.

1.5 Authors

- Claudio Walser (https://github.com/claudio-walser)
- Urban Etter (https://github.com/mms-uret)
- Gianni Carafa (https://github.com/mms-gianni)

1.6 Contributing

1. Fork it
2. Add this repository as an origin (`git remote add upstream https://github.com/gitcd-io/gitcd.git`)
3. Create your feature branch (`git cd start my-new-feature`)

4. Commit your changes (`git commit -am 'Add some feature'`)
5. Push to the branch (`git push origin feature/my-new-feature`)
6. Create new Pull Request against upstream (`git cd review my-new-feature`)

1.7 License

Apache License 2.0 see https://github.com/gitcd-io/gitcd/blob/master/LICENSE

CHAPTER 2

Todo's and features to implement

- Check for updates initially on every command - not even sure if this is smart
- implement all the assertions mentioned in the ./travis bash scripts
- test it with different remotes if possible

CHAPTER 3

Indices and tables

- genindex
- modindex
- search