

# Development with Git

*“Digging in your eye-sockets with a fondue fork is strictly considered to be bad for your health, and seven out of nine optometrists are dead set against the practice.”*

- Linus Torvalds, on Git mailing list



Every time you  
don't use version  
control, Linus  
Torvalds get's a  
little angrier!

# Outline

- Version control software
- Git for software development
- A few words on packaging in Python

# Version control

To 0'th order

- Development history for your source code

# Version control

To 0'th order

- Development history for your source code
- Collaboration with other developers

# Version control

To 0'th order

- Development history for your source code
- Collaboration with other developers
- Allows experimentation without breaking existing code

# Version control

To 0'th order

- Development history for your source code
- Collaboration with other developers
- Allows experimentation without breaking existing code
- Might take a little time to learn, but well worth it!

# Version control

## The generics

- Files and development history stored in repositories
- Check out files to a working directory
- Commit changes back to repository
- Update your working directory with commits from other developers
- Centralized vs. decentralized



# Version control

## Centralized

- Everyone commits to a server
- Does not encourage offline development
- Single point of failure
- 90's: CVS
- 00's: Subversion
- Now?

# Version control

## Decentralized

- Everyone has a copy
- Local commits
- Push to and pull from shared copy
- Encourages experimentation
- Many contenders
  - Mercurial
  - Bazaar
  - ....
  - Git



# git

<http://git-scm.com>

# Git

## Git basics

← → ↻ ⬆ 📄 www.kernel.org/pub/softw

---

### gittutorial(7) Man

---

#### NAME


gittutorial - A tutorial introduction

#### SYNOPSIS

git \*

#### DESCRIPTION

This tutorial explains how to im

 **git** --everything-is-local

#### About

- Documentation**
  - Reference
  - Book**
  - Videos
  - External Links
- Downloads**
- Community**

## Book

The entire Pro Git book, written by Scott Chacon and Ben Collins, is available here. All content is licensed under the [Creative Commons Attribution Non Commercial Share Alike 4.0 International License](#). Previous versions of the book are available on [Amazon](#).

- 1. Getting Started**
  - 1.1 About Version Control
  - 1.2 A Short History of Git
  - 1.3 Git Basics
  - 1.4 Installing Git
  - 1.5 First-Time Git Setup
  - 1.6 Getting Help
  - 1.7 Summary
- 2. Git Basics**
  - 2.1 Getting a Git Repository

# Git

## Getting started

```
$ mkdir myawesomesoftware  
$ cd myawesomesoftware  
$ git init
```

# Git

## Getting started

```
$ mkdir myawesomesoftware
```

```
$ cd myawesomesoftware
```

```
$ git init
```

```
Initialized empty Git repository in [path]
```

# Git

## Getting started

```
$ mkdir myawesomesoftware
$ cd myawesomesoftware
$ git init
Initialized empty Git repository in [path]
$ ls -a
. .. .git
```

# Git

## Getting started

```
$ mkdir myawesomesoftware
$ cd myawesomesoftware
$ git init
Initialized empty Git repository in [path]
$ ls -a
. .. .git
$ echo "My awesome software" > README
```



# Git

## Getting started

```
$ mkdir myawesomesoftware
$ cd myawesomesoftware
$ git init
Initialized empty Git repository in [path]
$ ls -a
. .. .git
$ echo "My awesome software" > README
$ git add README
```

# Git

## Getting started

```
$ mkdir myawesomesoftware
$ cd myawesomesoftware
$ git init
Initialized empty Git repository in [path]
$ ls -a
. .. .git
$ echo "My awesome software" > README
$ git add README
$ git commit -m "Initial commit with README file."
```

# Git

## Getting started

```
$ mkdir myawesomesoftware
$ cd myawesomesoftware
$ git init
Initialized empty Git repository in [path]
$ ls -a
. .. .git
$ echo "My awesome software" > README
$ git add README
$ git commit -m "Initial commit with README file."
[master (root-commit) 421659d] Bla bla
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 README
```

# Git

## Getting started

```
$ mkdir myawesomesoftware
$ cd myawesomesoftware
$ git init
Initialized empty Git repository in [path]
$ ls -a
. .. .git
$ echo "My awesome software" > README
$ git add README
$ git commit -m "Initial commit with README file."
[master (root-commit) 421659d] Bla bla
1 files changed, 1 insertions(+), 0 deletions(-)
create mode 100644 README

$ git help
```

# Git

## Under the hood

- Git keeps track of a database of commits.
- Every directory under version control has `.git/`
- A commit consists of [tree, author, timestamp, log message, parent commit(s)]
- Commits are named with hashes, eg.  
0d30e664c0839392a0ec8c7c266e9e194b8bb7f6
- Formally a directed acyclic graph

# Git

## Working and staging

```
$ git status
# On branch master
# Changes to be committed:
#   (use "git reset HEAD <file>..." to unstage)
#
#   new file:   another.txt
#
# Changes not staged for commit:
#   (use "git add <file>..." to update what will be committed)
#   (use "git checkout -- <file>..." to discard changes in working
directory)
#
#   modified:   README
#
# Untracked files:
#   (use "git add <file>..." to include in what will be committed)
#
#   athird.txt
```

# Git

## Local workflow

```
$ echo "Hope you like it" >> README
```

# Git

## Local workflow

```
$ echo "Hope you like it" >> README  
$ git status
```



# Git

## Local workflow

```
$ echo "Hope you like it" >> README
$ git status
[...]
modified:   README
[...]
```

# Git

## Local workflow

```
$ echo "Hope you like it" >> README
$ git status
[...]
modified:   README
[...]
$ git diff
```

# Git

## Local workflow

```
$ echo "Hope you like it" >> README
$ git status
[...]
modified:   README
[...]
$ git diff
[...]
    My awesome software
+Hope you like it
```

# Git

## Local workflow

```
$ echo "Hope you like it" >> README
$ git status
[...]
modified:   README
[...]
$ git diff
[...]
    My awesome software
+Hope you like it
$ git commit -a -m "Added more info to the README."
```

# Git

## Local workflow

```
$ echo "Hope you like it" >> README
$ git status
[...]
modified:   README
[...]
$ git diff
[...]
    My awesome software
+Hope you like it
$ git commit -a -m "Added more info to the README."
$ git status
```

# Git

## Local workflow

```
$ echo "Hope you like it" >> README
$ git status
[...]
modified:   README
[...]
$ git diff
[...]
    My awesome software
+Hope you like it
$ git commit -a -m "Added more info to the README."
$ git status
$ git log
```

# Git

## There and back again

- Checkout and revert
- If committed, you can (almost) never lose it!

Amend the previous commit

```
$ git commit --amend
```

# Git

## There and back again

- Checkout and revert
- If committed, you can (almost) never lose it!

Amend the previous commit

```
$ git commit --amend
```

Discarding changes to files

```
$ git checkout -- <file>
```



# Git

## There and back again

- Checkout and revert
- If committed, you can (almost) never lose it!

Amend the previous commit

```
$ git commit --amend
```

Discarding changes to files

```
$ git checkout -- <file>
```

Unstaging changes

```
$ git reset HEAD <file>
```

# Git

## There and back again

- Checkout and revert
- If committed, you can (almost) never lose it!

Amend the previous commit

```
$ git commit --amend
```

Discarding changes to files

```
$ git checkout -- <file>
```

Unstaging changes

```
$ git reset HEAD <file>
```

Create a new commit that removes some old commits

```
$ git revert
```

# Git

## There and back again

- Different levels of undo
- If committed, you can (almost) never lose it!

Amend the previous commit

```
$ git commit --amend
```

Discarding changes to files

```
$ git checkout <file>
```

Unstaging changes

```
$ git reset HEAD <file>
```

Create a new commit that removes some old commits

```
$ git revert
```

Rewinding commits. Only if they have not been pushed!

```
$ git reset --hard
```

# Git

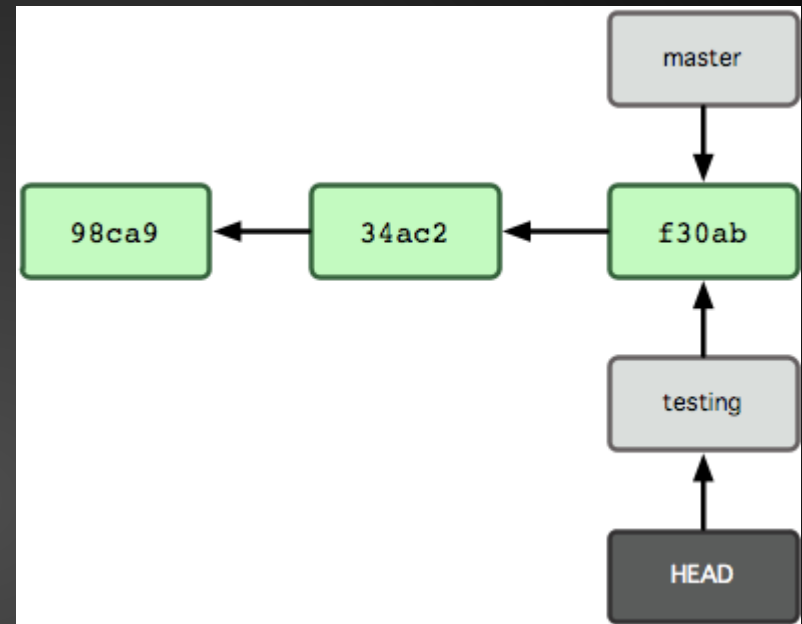
## Branches

```
$ git branch -a  
* master
```

# Git

## Branches

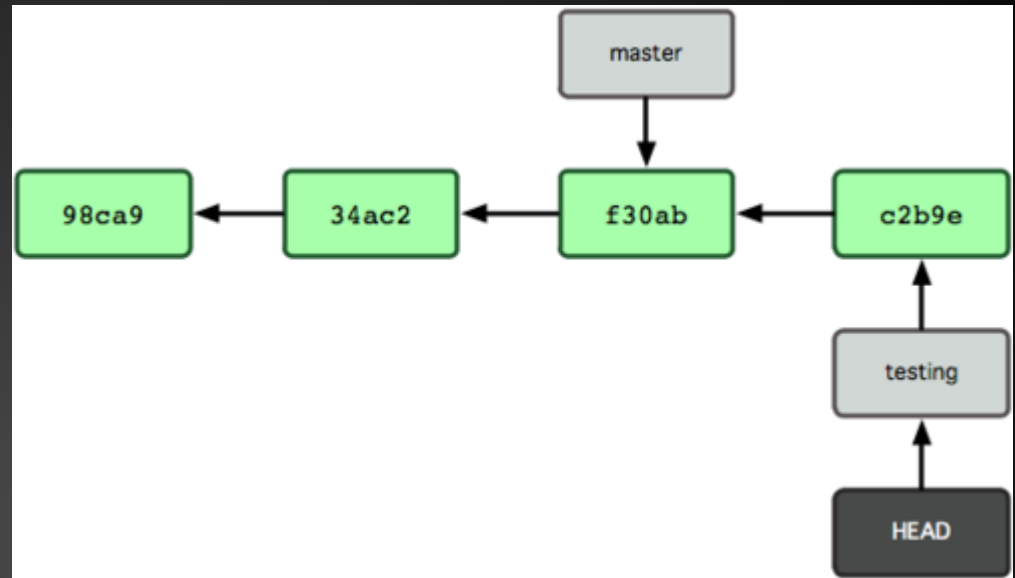
```
$ git branch -a
* master
$ git branch a_new_hope
$ git checkout a_new_hope
Switched to branch 'a_new_hope'
```



# Git

## Branches

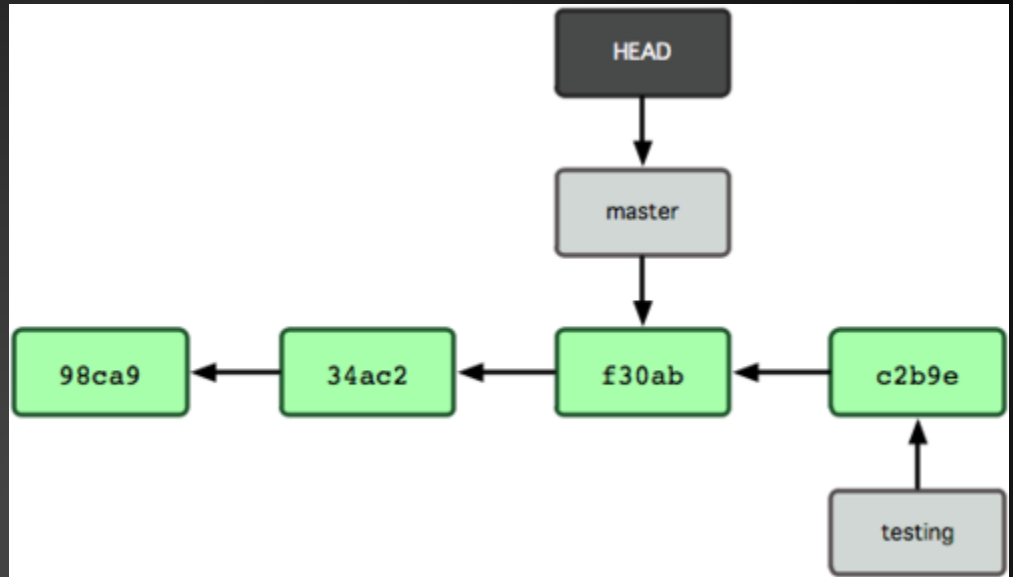
```
$ git branch -a
* master
$ git branch a_new_hope
$ git checkout a_new_hope
Switched to branch 'a_new_hope'
$ echo "CHANGES ARE COMING" >> README
$ git commit -a -m "Put changes warning in README"
```



# Git

## Branches

```
$ git branch -a
* master
$ git branch a_new_hope
$ git checkout a_new_hope
Switched to branch 'a_new_hope'
$ echo "CHANGES ARE COMING" >> README
$ git commit -a -m "Put changes warning in README"
$ git checkout master
$ git merge a_new_hope
Updating 16dbb3f..7351b65
Fast-forward
 README |      1 +
 1 files changed, 1 insertions(+), 0 deletions(-)
```



# Git

## Collaboration

```
$ git clone /path/on/shared/disk  
$ git clone git://git-server.com/...  
$ git clone user@host:path/to/repo  
$ git clone http://host/repo.git
```



# Git

## Collaboration

```
$ git clone /path/on/shared/disk  
$ git clone git://git-server.com/...  
$ git clone user@host:path/to/repo  
$ git clone http://host/repo.git
```

```
$ echo "My 5 cents" >> README  
$ git diff  
$ git commit -a -m "Changed README to include my 5 cents."
```

# Git

## Collaboration

```
$ git clone /path/on/shared/disk
$ git clone git://git-server.com/...
$ git clone user@host:path/to/repo
$ git clone http://host/repo.git

$ echo "My 5 cents" >> README
$ git diff
$ git commit -a -m "Changed README to include my 5 cents."
$ git pull
$ git push
```

# Git

## Setting up a shared repo

```
$ ssh myserver
```

```
$ cd /path/to/repos
```

```
$ mkdir myrepo.git
```

```
$ cd myrepo.git
```

```
$ git init --bare --shared
```

```
$ exit
```

# Git

## Setting up a shared repo

```
$ ssh myserver
```

```
$ cd /path/to/repos
```

```
$ mkdir myrepo.git
```

```
$ cd myrepo.git
```

```
$ git init --bare --shared
```

```
$ exit
```

```
$ cd /path/to/code
```

```
$ git remote add origin ssh://myserver/path/git/repos
```

```
$ git push -u origin master
```

# Git

## Resolving conflicts

```
$ git pull
```

```
CONFLICT (content): Merge conflict in file.txt
```

# Git

## Resolving conflicts

```
$ git pull
CONFLICT (content): Merge conflict in file.txt
$ cat file.txt
<<<<<< HEAD:file.txt
Hello world
=====
Goodbye
>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt
```

# Git

## Resolving conflicts

```
$ git pull
CONFLICT (content): Merge conflict in file.txt
$ cat file.txt
<<<<<< HEAD:file.txt
Hello world
=====
Goodbye
>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt
$ vim file.txt
Hello world
Goodbye
```

# Git

## Resolving conflicts

```
$ git pull
CONFLICT (content): Merge conflict in file.txt
$ cat file.txt
<<<<<< HEAD:file.txt
Hello world
=====
Goodbye
>>>>>> 77976da35a11db4580b80ae27e8d65caf5208086:file.txt
$ vim file.txt
Hello world
Goodbye
$ git add file.txt
$ git commit -m "Merged conflicts in file.txt"
```



# Git

## Blaming people

```
$ git blame README
^421659d (Henrik Brink 2012-08-19 16:27:25 -0700 1) My awesome
software
16dbb3f1 (Henrik Brink 2012-08-19 16:36:37 -0700 2) Hope you like it
7351b652 (Henrik Brink 2012-08-19 18:31:00 -0700 3) Bla
```

- Use for good
- Come up with good excuses

# Git

## "Github" flow

```
$ git clone ...
```

```
$ git checkout -b my_new_feature
```

# Git

## "Github" flow

```
$ git clone ...  
$ git checkout -b my_new_feature  
$ vim crazy_feature.py  
$ git commit ...  
$ git rebase master  
$ git push -u origin my_new_feature
```

Tell someone about your new branch and get feedback!

# Git

## "Github" flow

```
$ git clone ...  
$ git checkout -b my_new_feature  
$ vim crazy_feature.py  
$ git commit ...  
$ git rebase master  
$ git push -u origin my_new_feature
```

**Tell someone about your new branch and iterate... until:**

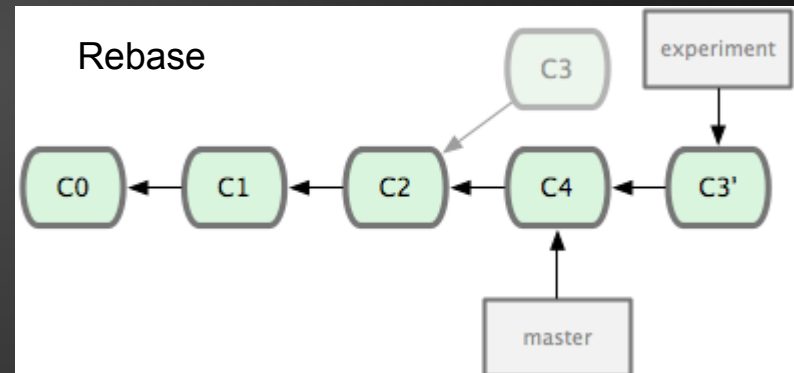
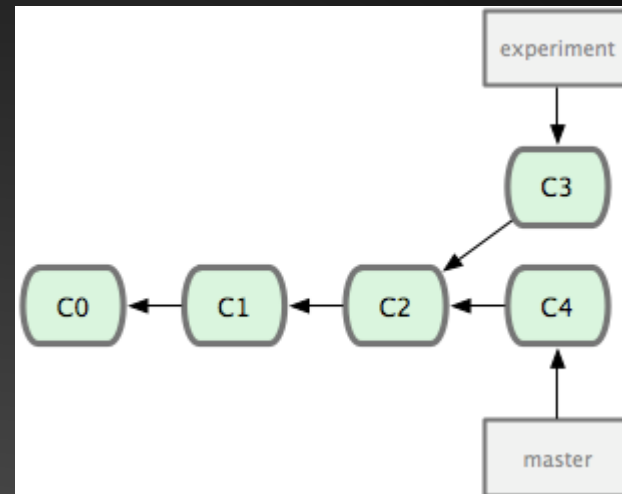
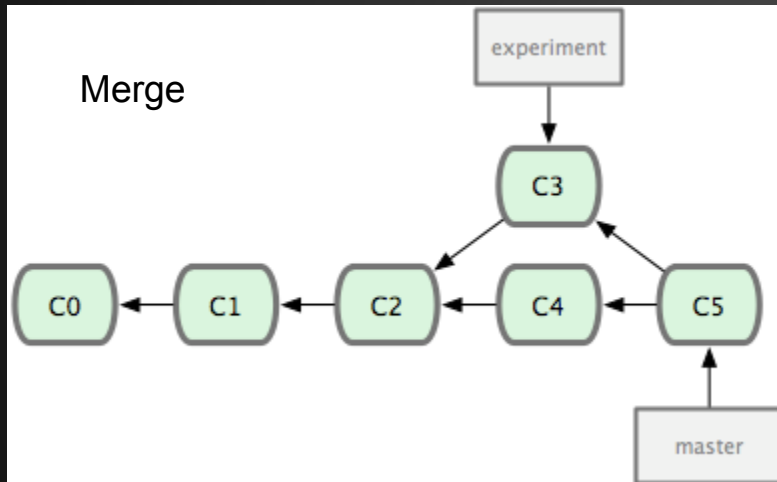
On branch master

```
$ git merge my_new_feature  
$ git pull  
$ git push
```

**Anything in master is deployable.**

# Git

## Merge vs Rebase



- Interactive rebase - powerful commit management.
- "Do not rebase commits that you have pushed to a public repository!!"

# Git

## Tagging

```
$ git tag  
v1.0  
$ git show v1.0  
[...]
```

# Git

## Tagging

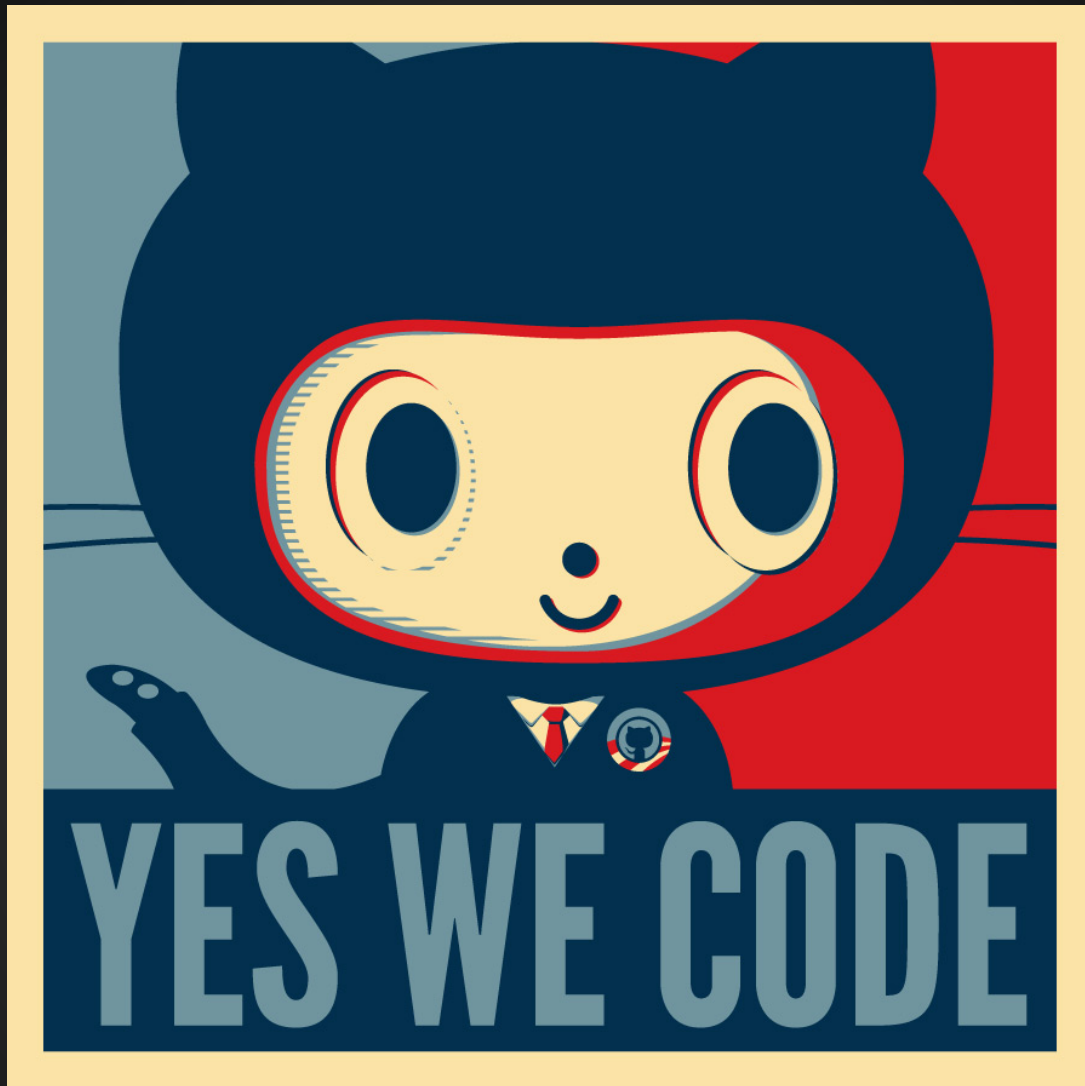
```
$ git tag  
v1.0  
$ git show v1.0  
[...]  
$ git tag -a v2.0 -m "2.0 release."  
$ git show v2.0
```

# Git

## Tagging

```
$ git tag  
v1.0  
$ git show v1.0  
[...]  
$ git tag -a v2.0 -m "2.0 release."  
$ git show v2.0  
  
$ git push origin v2.0
```





<http://github.com>


# Git

## Github?

- Your code in the cloud
- Handles all the plumbing of code collaboration
- Adds project management and social components
- Free for open source
- Revolutionizing software development!

# Git

## Github pull requests


 **octocat / Spoon-Knife**

[Admin](#) [Unwatch](#) [Fork](#) [Pull Request](#) [7,448](#) [6,255](#)

[Code](#) [Network](#) **[Pull Requests](#) 88** [Issues 121](#) [Stats & Graphs](#)

**Open** **cameronmcefee** wants someone to merge 1 commit into **octocat:master** from **cameronmcefee:master** **#248**

[Discussion](#) [Commits <> 1](#) [Diff >= 1](#)






cameronmcefee opened this pull request 10 minutes ago

### Sending a pull request


No one is assigned [+](#) No milestone [+](#)


I made some changes. Please review.


2 participants  




[+](#) **cameronmcefee** added some commits 32 minutes ago


[a4610fa](#)  Made some changes for a pull request



 **octocat** commented a minute ago

Awesome, thanks!



 **cameronmcefee** commented just now

Why yes, of course.

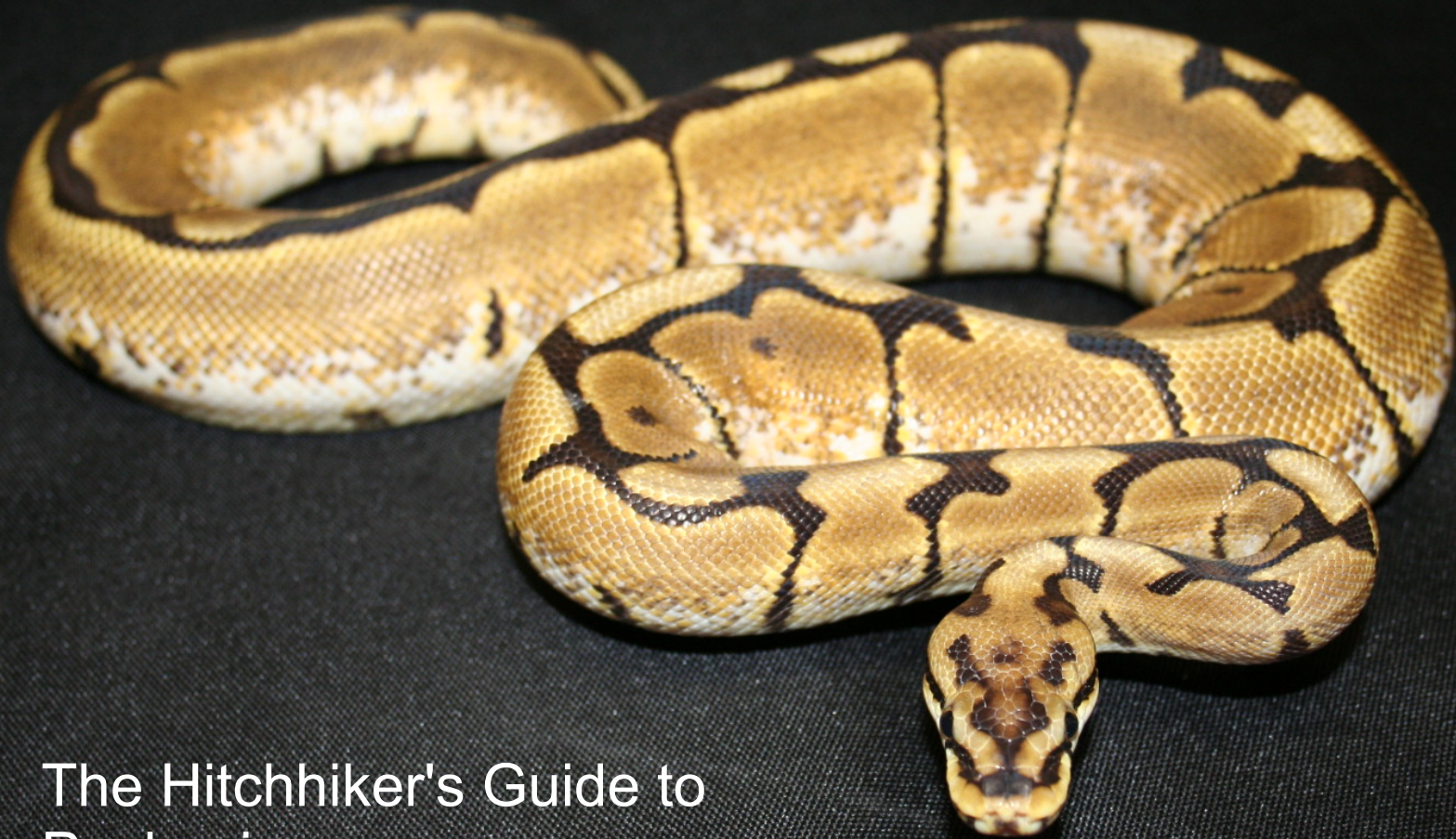
[i](#) This pull request can be automatically merged. [Merge pull request](#)

# Git

## Github alternatives

- <https://bitbucket.org/>
  - Unlimited free private repos
  - A step behind Github
- <http://gitlabhq.com/>
  - Self-hosted Github-like
  - Only basics





## The Hitchhiker's Guide to Packaging

<http://guide.python-distribute.org>

# Python packaging

## Directory structure

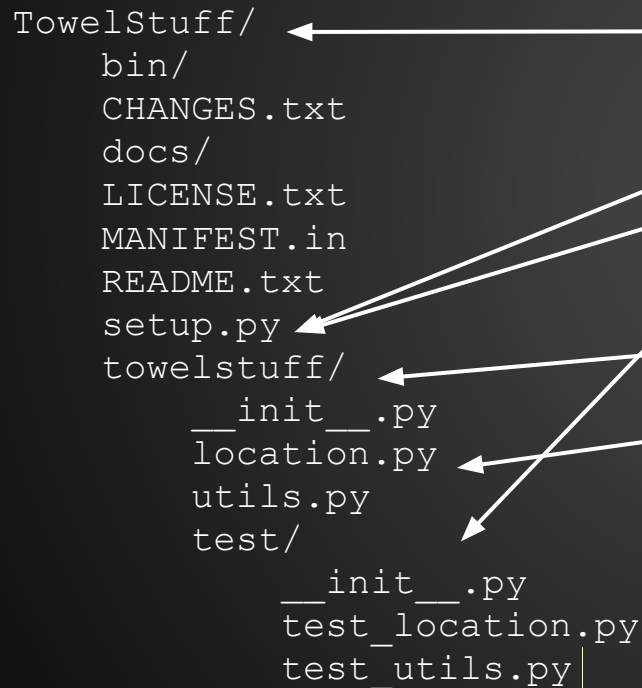
```
TowelStuff/  
  bin/  
  CHANGES.txt  
  docs/  
  LICENSE.txt  
  MANIFEST.in  
  README.txt  
  setup.py  
  towelstuff/  
    __init__.py  
    location.py  
    utils.py  
    test/  
      __init__.py  
      test_location.py  
      test_utils.py|
```



# Python packaging

## Directory structure

```
TowelStuff/
  bin/
  CHANGES.txt
  docs/
  LICENSE.txt
  MANIFEST.in
  README.txt
  setup.py
  towelstuff/
    __init__.py
    location.py
    utils.py
    test/
      __init__.py
      test_location.py
      test_utils.py
```



The diagram shows arrows pointing from the code snippets on the right to specific files and directories in the structure on the left. An arrow points from the first command '\$ cd TowelStuff' to the 'TowelStuff/' directory. An arrow points from '\$ python setup.py test' to 'setup.py'. An arrow points from '\$ python setup.py install' to 'setup.py'. An arrow points from '\$ cd /path/to/mytowelprog' to 'towelstuff/'. An arrow points from '\$ vim mytowelprog.py' to 'test/'. An arrow points from 'import towelstuff' to 'towelstuff/'. An arrow points from 'import towelstuff.location' to 'location.py'. An arrow points from the final command '\$ python mytowelprog.py' to 'test/'.

```
$ cd TowelStuff
$ python setup.py test
$ python setup.py install
$ cd /path/to/mytowelprog
$ vim mytowelprog.py

import towelstuff
import towelstuff.location
[...]

$ python mytowelprog.py
```

# Python packaging

## setup.py

```
from distutils.core import setup

setup(
    name='TowelStuff',
    version='0.1.0',
    author='J. Random Hacker',
    author_email='jrh@example.com',
    packages=['towelstuff', 'towelstuff.test'],
    scripts=['bin/stowe-towels.py', 'bin/wash-towels.py'],
    url='http://pypi.python.org/pypi/TowelStuff/',
    license='LICENSE.txt',
    description='Useful towel-related stuff.',
    long_description=open('README.txt').read(),
    install_requires=[
        "Django >= 1.1.1",
        "caldav == 0.1.4",
    ],
)
```



# Python packaging

## Publishing

- Create repository on Github (or other)
- Then it's just a matter of...

```
$ cd TowelStuff
$ git init
$ git add . ## Bad practice!
$ git commit -m "Imported TowelStuff package into Git."
$ git remote add origin https://github.com/<username>/TowelStuff.git
```

# Python packaging

## Publishing

- Create repository on Github (or other)
- Then it's just a matter of...

```
$ cd TowelStuff
$ git init
$ git add . ## Bad practice!
$ git commit -m "Imported TowelStuff package into Git."
$ git remote add origin https://github.com/<username>/TowelStuff.git
$ git push -u origin master
```

# Breakout session

```
$ git clone git://github.com/brinkar/bloomdemo.git  
$ cd bloomdemo  
$ less INSTRUCTIONS
```