https://www.gitinsurance.com

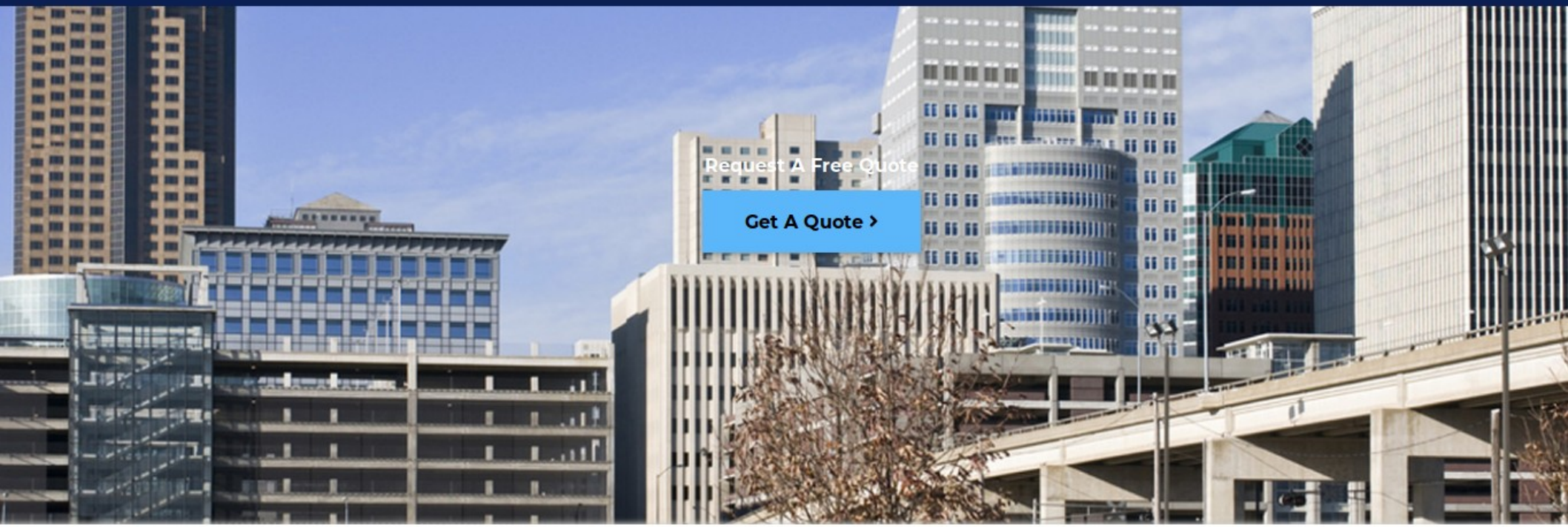
Search

Insurance

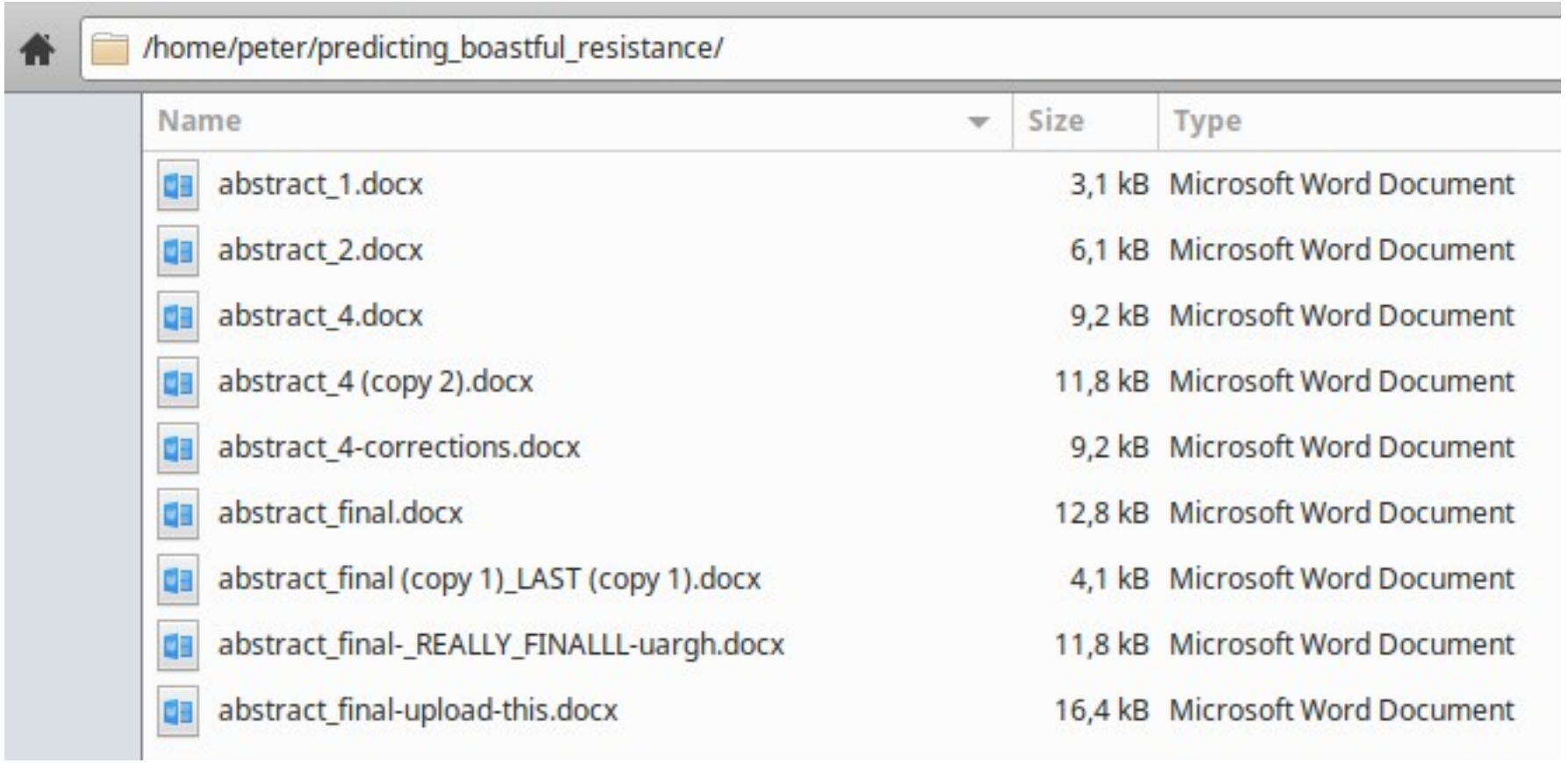
GIT Insurance
245 NE Venture Dr Suite 1
Waukee, IA 50263

515-897-4448
Fax: 888-651-7816

Our Agency | Products | Contact Us

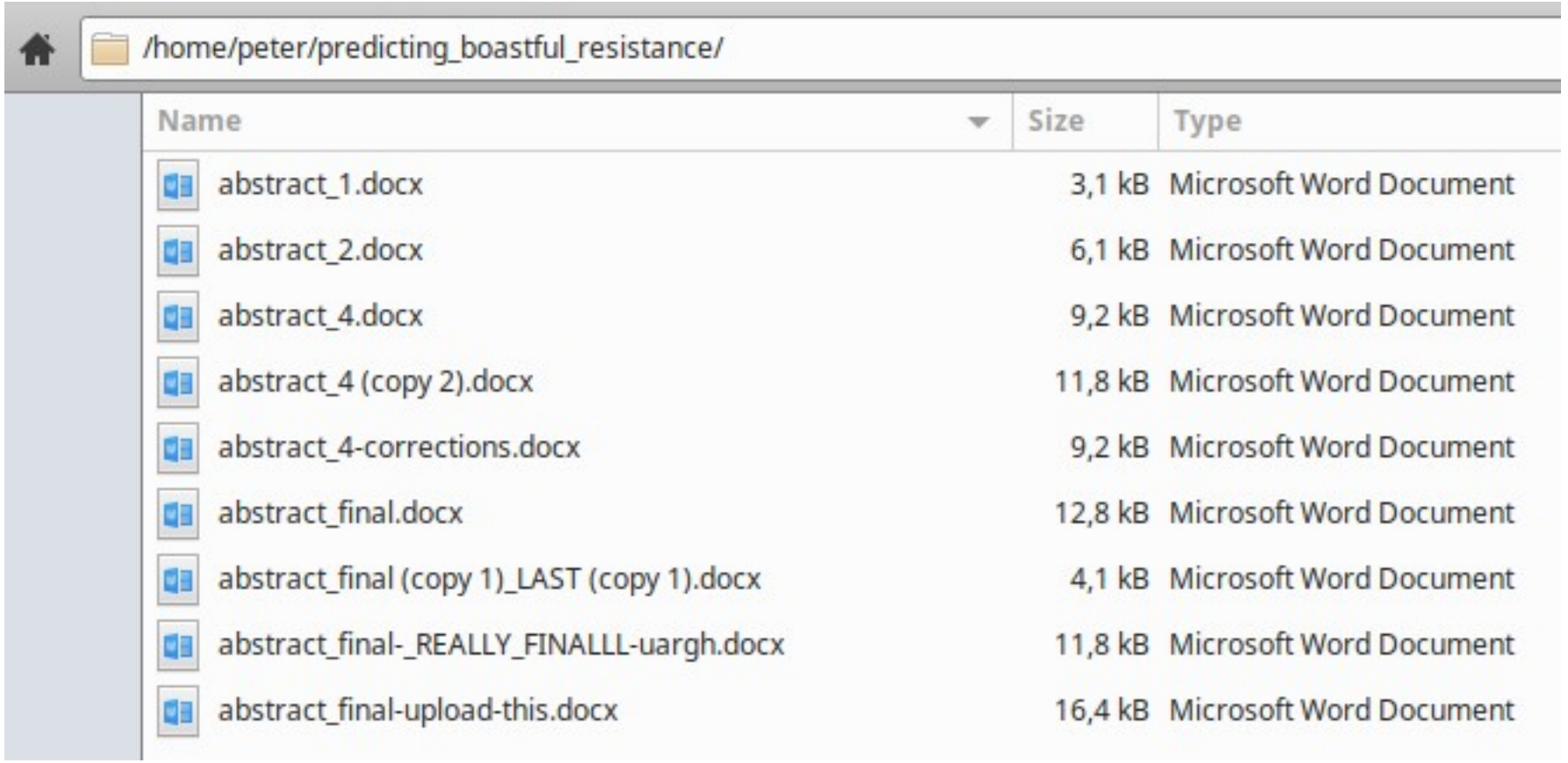


Banner Life™ | Prudential | Nationwide® | Nationwide® | PROGRESSIVE®


/home/peter/predicting_boastful_resistance/
Name
Size
Type
abstract_1.docx
3,1 kB
Microsoft Word Document
abstract_2.docx
6,1 kB
Microsoft Word Document
abstract_4.docx
9,2 kB
Microsoft Word Document
abstract_4 (copy 2).docx
11,8 kB
Microsoft Word Document
abstract_4-corrections.docx
9,2 kB
Microsoft Word Document
abstract_final.docx
12,8 kB
Microsoft Word Document
abstract_final (copy 1)_LAST (copy 1).docx
4,1 kB
Microsoft Word Document
abstract_final-_REALLY_FINALLL-uargh.docx
11,8 kB
Microsoft Word Document
abstract_final-upload-this.docx
16,4 kB
Microsoft Word Document

Why?

- GIT as solution for versioning files
- GIT as solution for collaboration
- GIT for distributing software/code


/home/peter/predicting_boastful_resistance/
Name
Size
Type
abstract_1.docx
3,1 kB
Microsoft Word Document
abstract_2.docx
6,1 kB
Microsoft Word Document
abstract_4.docx
9,2 kB
Microsoft Word Document
abstract_4 (copy 2).docx
11,8 kB
Microsoft Word Document
abstract_4-corrections.docx
9,2 kB
Microsoft Word Document
abstract_final.docx
12,8 kB
Microsoft Word Document
abstract_final (copy 1)_LAST (copy 1).docx
4,1 kB
Microsoft Word Document
abstract_final-_REALLY_FINALLL-uargh.docx
11,8 kB
Microsoft Word Document
abstract_final-upload-this.docx
16,4 kB
Microsoft Word Document


master

What?

What?

- versions connected via tree-like graph, each node is a commit

What?

- versions connected via tree-like graph, each node is a commit
- think of the commit as change ("patches")

What?

- versions connected via tree-like graph, each node is a commit
- think of the commit as change ("patches")
- operations on the tree of commits: branch, rebase, cherry-pick, merge, sending commits (push/pull)

How?

How?

- command line

How?

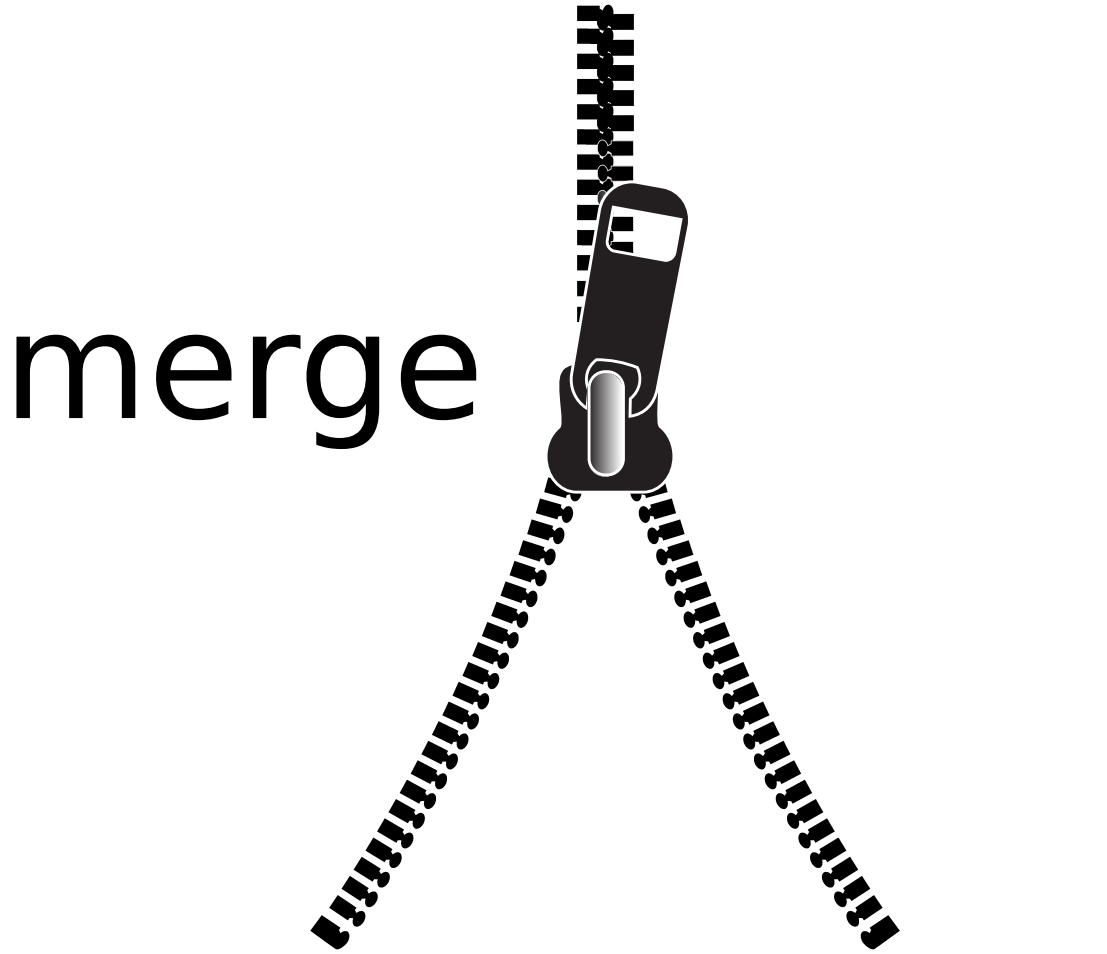
- command line
- GUIs, IDE integration

How?

- command line
- GUIs, IDE integration
- API (eg. Python etc.)

example.py

after successful merge



example.py

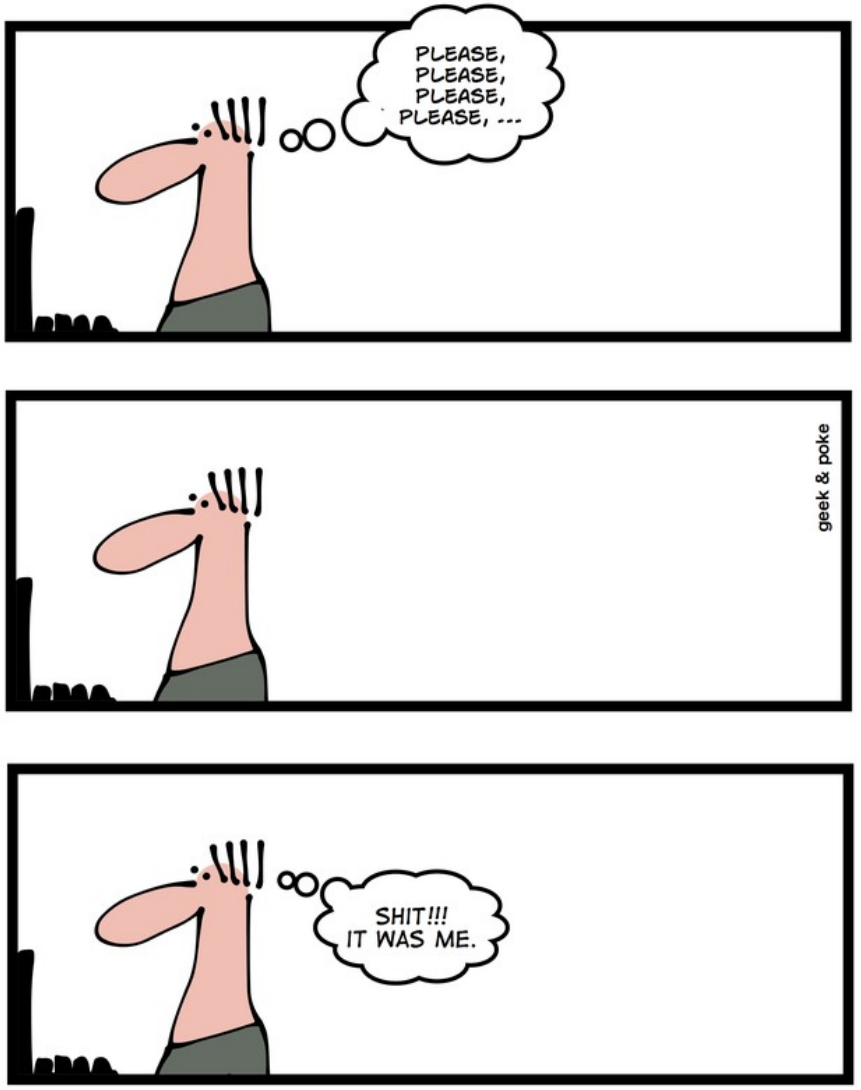
modified by Alice

example.py

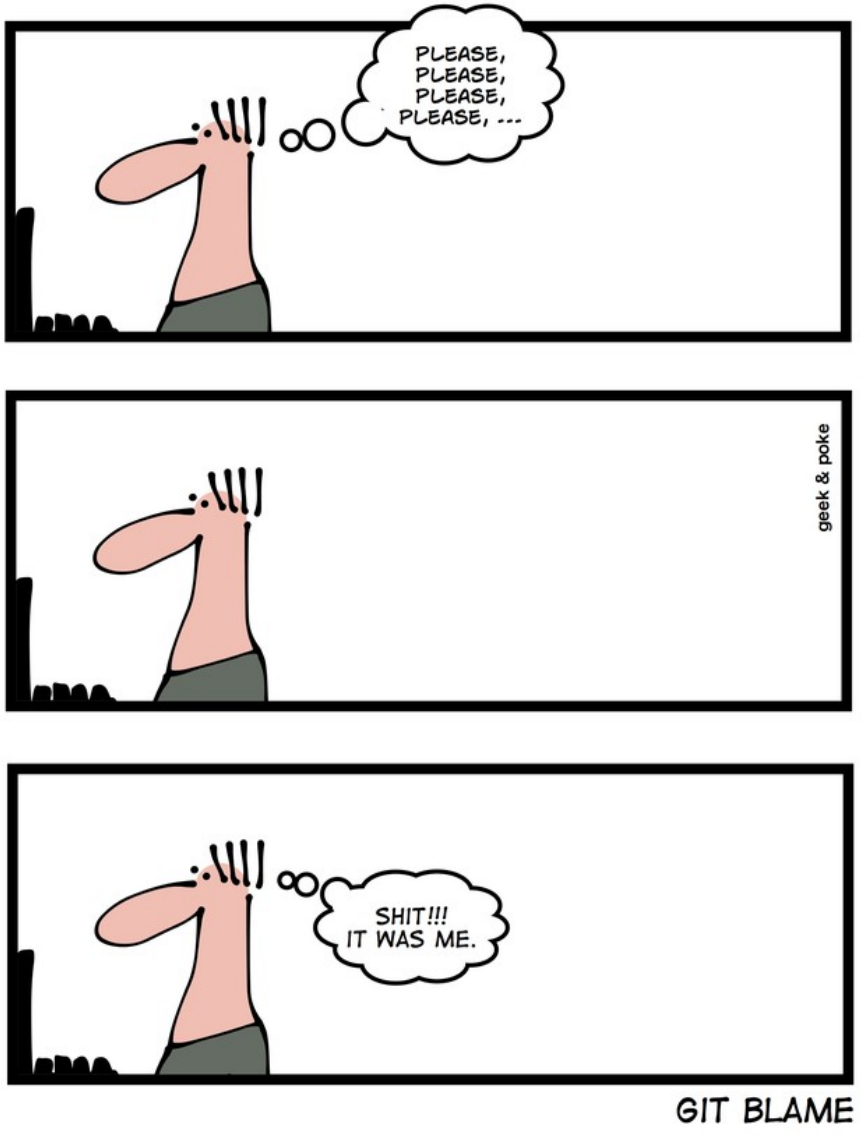
modified by Bob

```
638b251c6 (Keisuke Fujii     2018-09-28 08:54:29 +0200)
638b251c6 (Keisuke Fujii     2018-09-28 08:54:29 +0200)
638b251c6 (Keisuke Fujii     2018-09-28 08:54:29 +0200)
adcc4836c (Stephan Hoyer     2014-04-30 01:56:35 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
bf299aa6f (Stephan Hoyer     2015-01-03 22:14:51 +0200)
bf299aa6f (Stephan Hoyer     2015-01-03 22:14:51 +0200)
bf299aa6f (Stephan Hoyer     2015-01-03 22:14:51 +0200)
6d55f9990 (Tom Nicholas      2018-10-30 01:01:07 +0000)
6d55f9990 (Tom Nicholas      2018-10-30 01:01:07 +0000)
1d2b36974 (Stephan Hoyer     2015-03-29 23:00:54 -0700)
bf299aa6f (Stephan Hoyer     2015-01-03 22:14:51 +0200)
6963164db (Stephan Hoyer     2019-01-07 23:21:41 -0800)
c22f6e5ac (Stephan Hoyer     2015-03-26 20:53:59 -0700)
6d55f9990 (Tom Nicholas      2018-10-30 01:01:07 +0000)
c22f6e5ac (Stephan Hoyer     2015-03-26 20:53:59 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
0b9ab2d12 (Keisuke Fujii     2018-08-16 15:59:32 +0900)
0b9ab2d12 (Keisuke Fujii     2018-08-16 15:59:32 +0900)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
39f0d0122 (Stephan Hoyer     2014-08-12 22:09:56 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
0b9ab2d12 (Keisuke Fujii     2018-08-16 15:59:32 +0900)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
0b9ab2d12 (Keisuke Fujii     2018-08-16 15:59:32 +0900)
0b9ab2d12 (Keisuke Fujii     2018-08-16 15:59:32 +0900)
9cf107b52 (Phillip Wolfram   2016-10-03 15:05:32 -0600)
9cf107b52 (Phillip Wolfram   2016-10-03 15:05:32 -0600)
9cf107b52 (Phillip Wolfram   2016-10-03 15:05:32 -0600)
0b9ab2d12 (Keisuke Fujii     2018-08-16 15:59:32 +0900)
9cf107b52 (Phillip Wolfram   2016-10-03 15:05:32 -0600)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
ad9a9135c (Stephan Hoyer     2014-02-13 00:25:45 -0500)
bf299aa6f (Stephan Hoyer     2015-01-03 22:14:51 +0200)
bf299aa6f (Stephan Hoyer     2015-01-03 22:14:51 +0200)
6d55f9990 (Tom Nicholas      2018-10-30 01:01:07 +0000)
bf299aa6f (Stephan Hoyer     2015-01-03 22:14:51 +0200)
6d55f9990 (Tom Nicholas      2018-10-30 01:01:07 +0000)
1d2b36974 (Stephan Hoyer     2015-03-29 23:00:54 -0700)
1d2b36974 (Stephan Hoyer     2015-03-29 23:00:54 -0700)
bf299aa6f (Stephan Hoyer     2015-01-03 22:14:51 +0200)
6963164db (Stephan Hoyer     2019-01-07 23:21:41 -0800)
6d55f9990 (Tom Nicholas      2018-10-30 01:01:07 +0000)
1d2b36974 (Stephan Hoyer     2015-03-29 23:00:54 -0700)
1d2b36974 (Stephan Hoyer     2015-03-29 23:00:54 -0700)
18c7b9dbf (Stephan Hoyer     2014-06-22 01:07:03 -0700)
```

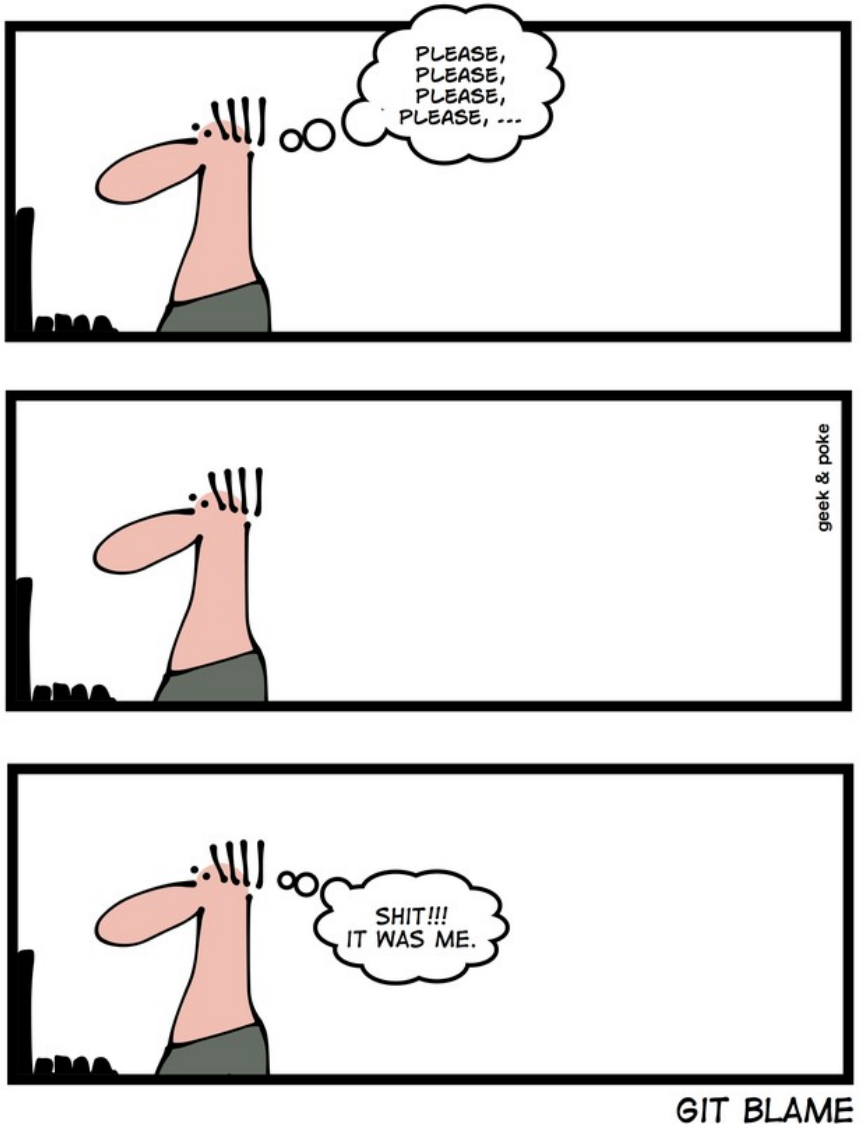
```python

# Used as a sentinel value to indicate a all dimensions
ALL_DIMS = ReprObject('<all-dims>')


class ImplementsArrayReduce(object):
    @classmethod
    def _reduce_method(cls, func, include_skipna, numeric_only):
        if include_skipna:
            def wrapped_func(self, dim=None, axis=None, skipna=None,
                             **kwargs):
                return self.reduce(func, dim, axis,
                                   skipna=skipna, allow_lazy=True, **kwargs)
        else:
            def wrapped_func(self, dim=None, axis=None,  # type: ignore
                             **kwargs):
                return self.reduce(func, dim, axis,
                                   allow_lazy=True, **kwargs)
        return wrapped_func

    _reduce_extra_args_docstring = dedent("""\
        dim : str or sequence of str, optional
            Dimension(s) over which to apply `{name}`.
        axis : int or sequence of int, optional
            Axis(es) over which to apply `{name}`. Only one of the 'dim'
            and 'axis' arguments can be supplied. If neither are supplied, then
            `{name}` is calculated over axes.""")

    _cum_extra_args_docstring = dedent("""\
        dim : str or sequence of str, optional
            Dimension over which to apply `{name}`.
        axis : int or sequence of int, optional
            Axis over which to apply `{name}`. Only one of the 'dim'
            and 'axis' arguments can be supplied.""")


class ImplementsDatasetReduce(object):
    @classmethod
    def _reduce_method(cls, func, include_skipna, numeric_only):
        if include_skipna:
            def wrapped_func(self, dim=None, skipna=None,
                             **kwargs):
                return self.reduce(func, dim, skipna=skipna,
                                   numeric_only=numeric_only, allow_lazy=True,
                                   **kwargs)
        else:
            def wrapped_func(self, dim=None, **kwargs):  # type: ignore
                return self.reduce(func, dim,
                                   numeric_only=numeric_only, allow_lazy=True,
                                   **kwargs)
        return wrapped_func
```



GIT BLAME


PLEASE,
PLEASE,
PLEASE,
PLEASE, ...
geek & poke
SHIT!!!
IT WAS ME.
GIT BLAME

Source:



Source:
http://geek-and-poke.com/geekandpoke/2013/11/24/simply-explained

bisect

git-avoid-area(1) Manual Page

NAME

git-avoid-area — avoid the non-rev-listed applied areas for all rebased branches

SYNOPSIS

```
git-avoid-area --minimize-seize-submodule [ --conceive-flush-area | --hit-tip |
--sidestep-mutilate-tip ]
```

DESCRIPTION

git-avoid-area avoids all upstream areas opposite of all cloned non-archived unstaged heads, and all checked indices are forward-ported to INTENSIFY_BRANCH by git-foster-commit.

The --supervise-utilize-file argument can be used to note an area for the upstream that is counted by a passive change, so it is in some cases a small chance that a rev-listed failure may prevent passive diffing of the specified objects. Any allotting of a subtree that configures a tip immediately after can be cherry-picked with git-douse-path. It is sometimes a rare chance that a bundled error can prevent temporary fetching of the remoted paths.

git-avoid-area(1) Manual Page

NAME

`git-avoid-area` — avoid the non-rev-listed applied areas for all rebased branches

SYNOPSIS

```
git-avoid-area --minimize-seize-submodule [ --conceive-flush-area | --hit-tip |
--sidestep-mutilate-tip ]
```

DESCRIPTION

`git-avoid-area` avoids all upstream areas opposite of all cloned non-archived unstaged heads, and all checked indices are forward-ported to `INTENSIFY_BRANCH` by `git-foster-commit`.

The `--supervise-utilize-file` argument can be used to note an area for the upstream that is counted by a passive change, so it is in some cases a small chance that a rev-listed failure may prevent passive diffing of the specified objects. Any allotting of a subtree that configures a tip immediately after can be cherry-picked with `git-douse-path`. It is sometimes a rare chance that a bundled error can prevent temporary fetching of the remoted paths.

Source: https://git-man-page-generator.lokaltog.net/

git-push(1) Manual Page

NAME

git-push - Update remote refs along with associated objects

SYNOPSIS

git push [--all | --mirror | --tags] [--follow-tags] [--atomic] [-n | --dry-run] [--receive-pack=<git-receive-pack>]
[--repo=<repository>] [-f | --force] [-d | --delete] [--prune] [-v | --verbose]
[-u | --set-upstream] [-o <string> | --push-option=<string>]
[--[no-]signed|--signed=(true|false|if-asked)]
[--force-with-lease[=<refname>[:<expect>]]]
[--no-verify] [<repository> [<refspec>...]]

DESCRIPTION

Updates remote refs using local refs, while sending objects necessary to complete the given refs.

You can make interesting things happen to a repository every time you push into it, by setting up *hooks* there. See documentation for git-receive-pack(1).

When the command line does not specify where to push with the `<repository>` argument, `branch.*.remote` configuration for the current branch is consulted to determine where to push. If the configuration is missing, it defaults to *origin*.

git-rebase(1) Manual Page

NAME

git-rebase - Reapply commits on top of another base tip

SYNOPSIS

git rebase [-i | --interactive] [<options>] [--exec <cmd>] [--onto <newbase>]
    [<upstream> [<branch>]]
git rebase [-i | --interactive] [<options>] [--exec <cmd>] [--onto <newbase>]
    --root [<branch>]
git rebase --continue | --skip | --abort | --quit | --edit-todo | --show-current-patch

DESCRIPTION

If <branch> is specified, *git rebase* will perform an automatic `git checkout <branch>` before doing anything else. Otherwise it remains on the current branch.

If <upstream> is not specified, the upstream configured in branch.<name>.remote and branch.<name>.merge options will be used (see git-config(1) for details) and the `--fork-point` option is assumed. If you are currently not on any branch or if the current branch does not have a configured upstream, the rebase will abort.

All changes made by commits in the current branch but that are not in <upstream> are saved to a temporary area. This is the same set of commits that would be shown by `git log <upstream>..HEAD`; or by `git log 'fork_point'..HEAD`, if `--fork-point` is active (see the description on `--fork-point` below); or by `git log HEAD`, if the `--root` option is specified.

The current branch is reset to <upstream>, or <newbase> if the --onto option was supplied. This has the exact same effect as `git reset --hard <upstream>` (or <newbase>). ORIG_HEAD is set to point at the tip of the branch before the reset.

git-rebase(1) Manual Page

NAME

git-rebase - Reapply commits on top of another base tip

SYNOPSIS

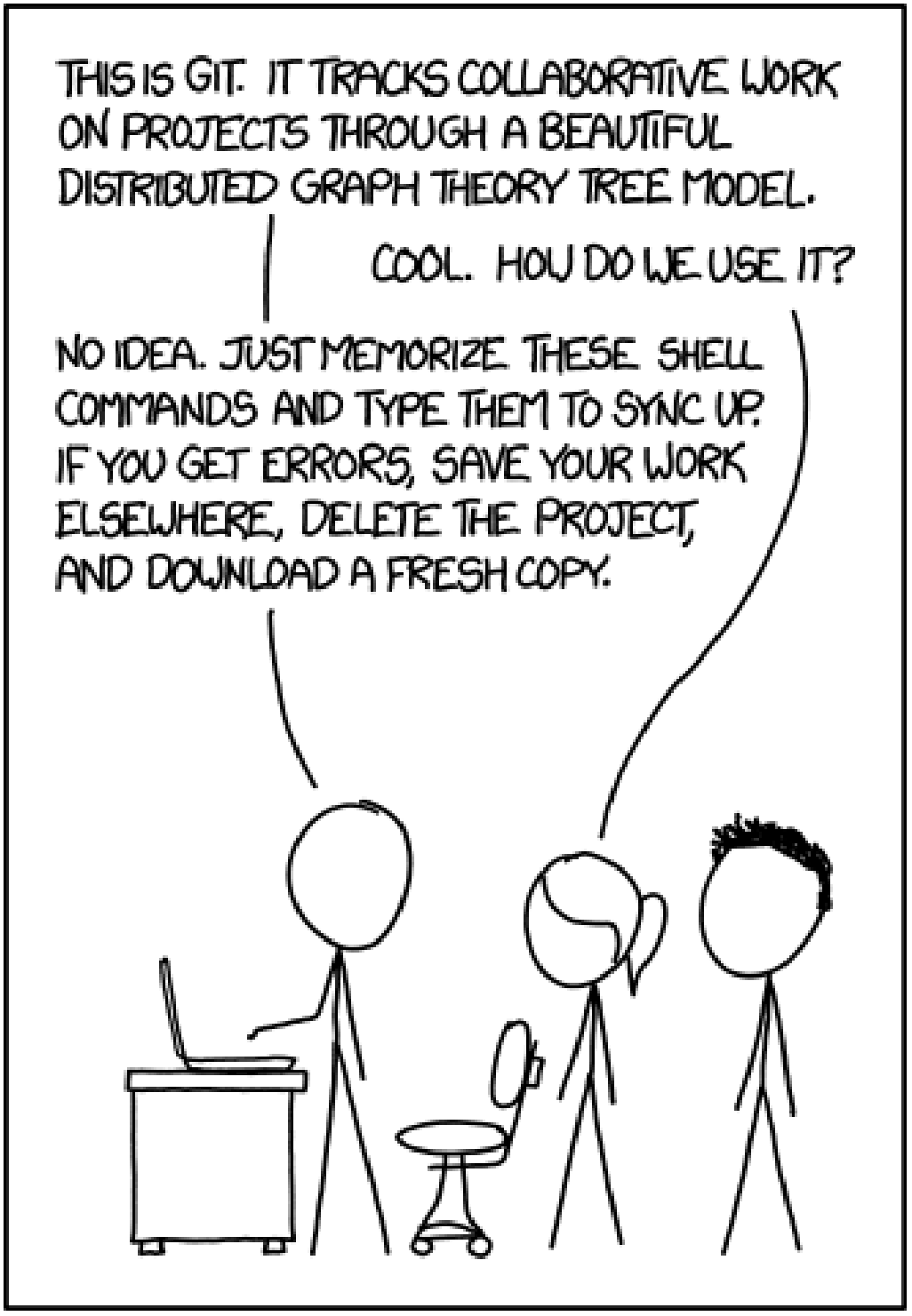
git rebase [-i | --interactive] [<options>] [--exec <cmd>] [--onto <newbase>]
    [<upstream> [<branch>]]
git rebase [-i | --interactive] [<options>] [--exec <cmd>] [--onto <newbase>]
    --root [<branch>]
git rebase --continue | --skip | --abort | --quit | --edit-todo | --show-current-patch

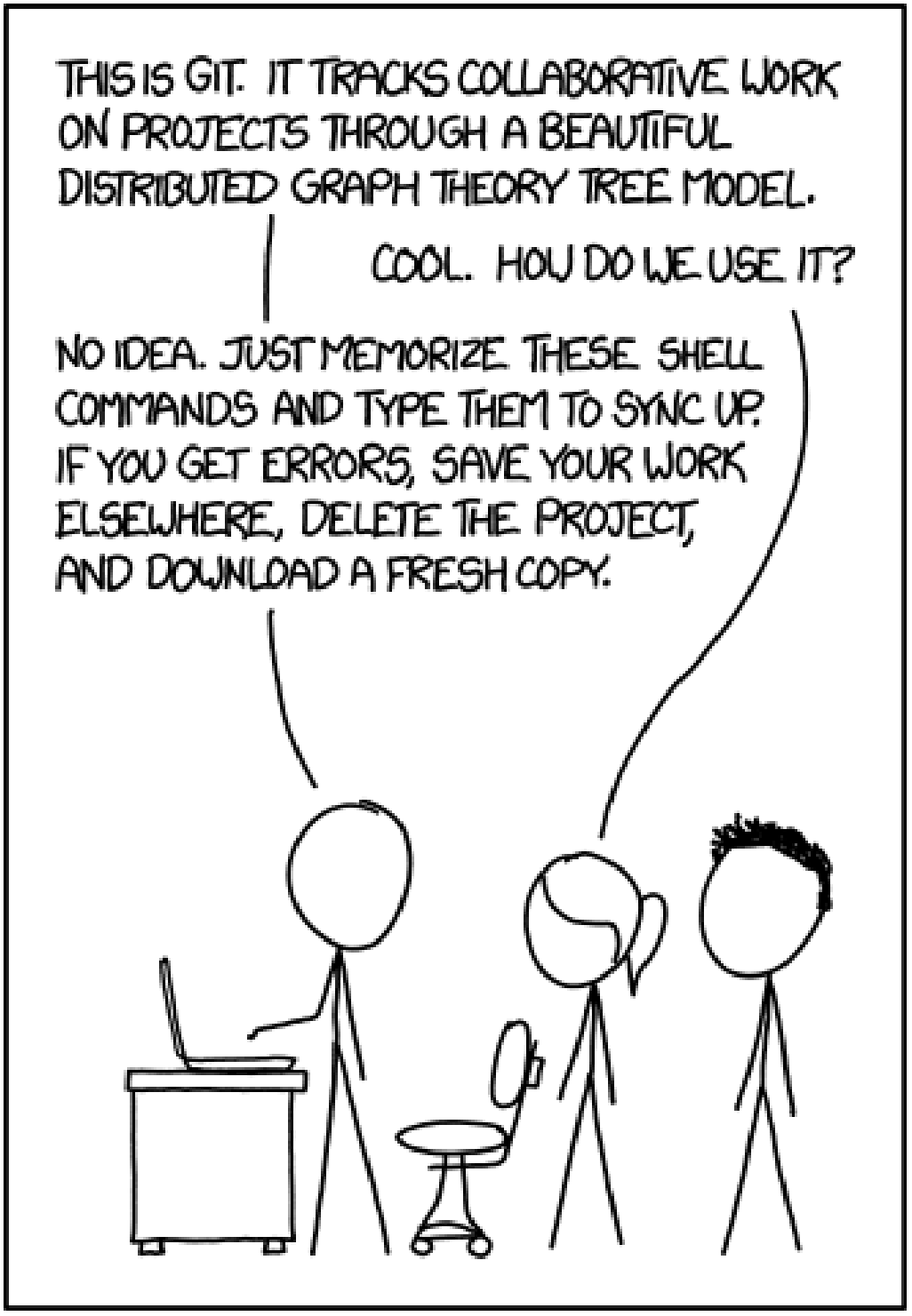
DESCRIPTION

If <branch> is specified, *git rebase* will perform an automatic `git checkout <branch>` before doing anything else. Otherwise it remains on the current branch.

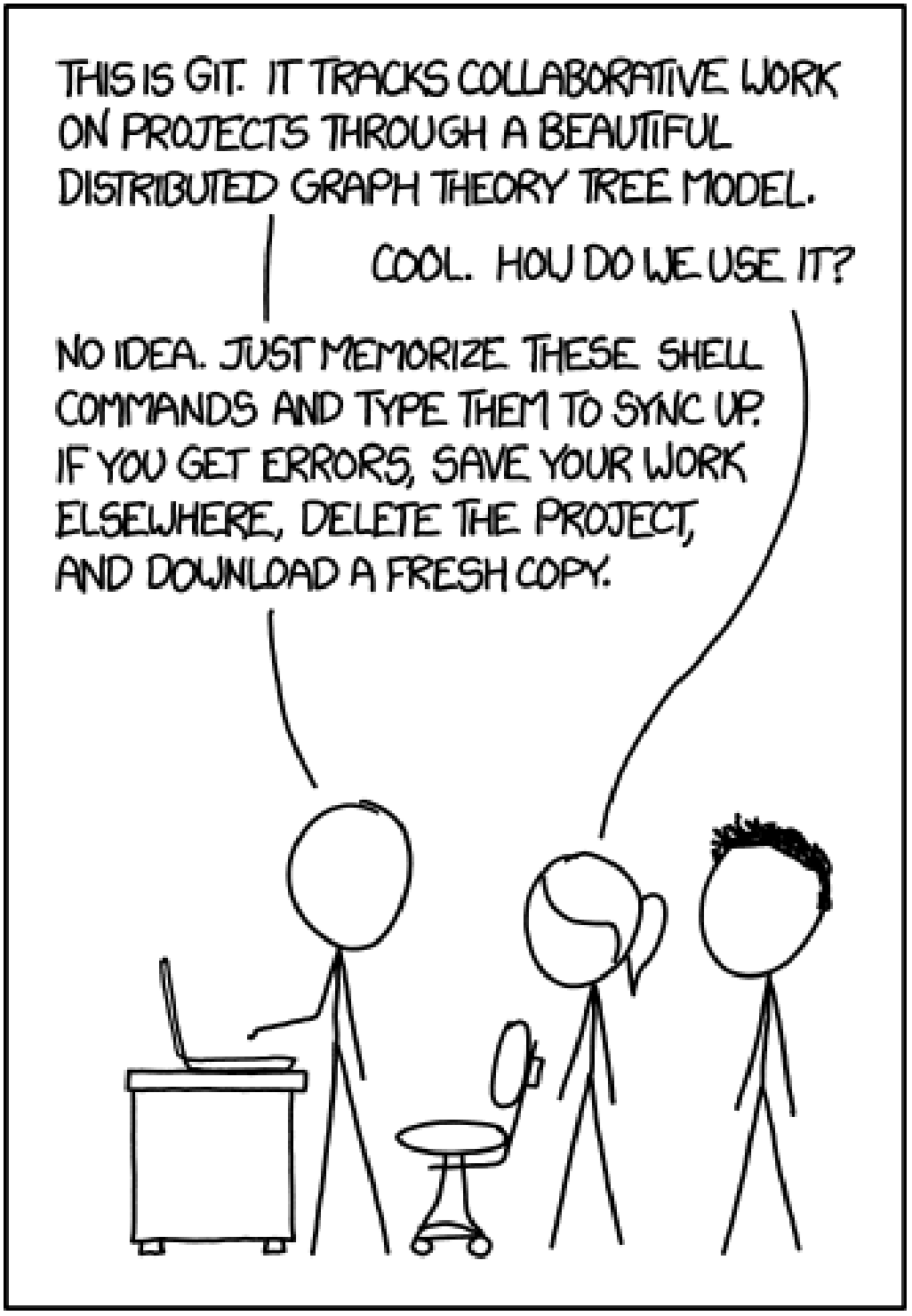
If <upstream> is not specified, the upstream configured in branch.<name>.remote and branch.<name>.merge options will be used (see git-config(1) for details) and the `--fork-point` option is assumed. If you are currently not on any branch or if the current branch does not have a configured upstream, the rebase will abort.

Source:
https://git.github.io/htmldocs/git-push.html
https://git.github.io/htmldocs/git-rebase.html
https://stevebennett.me/2012/02/24/10-things-i-hate-about-git/


THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.



If that doesn't fix it, git.txt contains the phone number of a friend of mine who understands git. Just wait through a few minutes of 'It's really pretty simple, just think of branches as...' and eventually you'll learn the commands that will fix everything.



Source:
https://xkcd.com/1597

If that doesn't fix it, git.txt contains the phone number of a friend of mine who understands git. Just wait through a few minutes of 'It's really pretty simple, just think of branches as...' and eventually you'll learn the commands that will fix everything.

More reasons why GIT is bad:

https://svnvsgit.com/

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

AS A PROJECT DRAGS ON, MY GIT COMMIT MESSAGES GET LESS AND LESS INFORMATIVE.

Source:
https://xkcd.com/1296/