EVERYBODY IS GIT FU DEVELOPING

bephpug

@movetodevnull
sebastianfeldmann

ABOUT ME

CHECK24

~ 30 years

CAPTAIN HOOK

phpbu

@movetodevnull

Commits on Sep 19, 2017

It's working!
Mc Coy
c705261

Commits on Sep 18, 2017

minor changes
Mr Spock
273c090

Fixed Tpyo
Mc Coy
3d0cec6

for real 4 real this time
Skotty
c37252e

Commits on Aug 30, 2017

fix bug, for realz
Skotty
e616626

fixed bug
Scotty
c8d3c95

Commits on Aug 28, 2017

added security.
Lt Worf
689b123

Merge pull request #67 from pille/fix-spocks-shit Fix spocks shit
Mc Coy
19a23a7

commented out failing tests
Mr Spock
3c46a0a

- Temporary commit.
James T. Kirk
8850b9f

Commits on Aug 8, 2017

Debug suff


#InTheCut
BOUNCE

ATOMIC COMMITS

ATOMIC COMMITS

Implemented new voting feature, fixed an internal messaging issue and refactored some inheritance menace

Add voting api endpoints and actions

Fix internal messaging issue

Refactor Formatters to be composable

Add voting persistence services

RULES

- Single irreducible **useful** set of changes
- Everything **works**
- No „**and**“ in commit message subjects

COMMIT MESSAGES

WHATTHECOMMIT.COM

- Major fixup
- Fixed tpyo
- One does not simply merge into master
- Best commit ever

RULES

- Separate subject from body with one blank line
- Limit the subject line to 50 characters
- Do not end the subject line with a period
- Capitalize the subject line
- Use imperative mood for the subject line
- Wrap the body at 72 characters
- Use the body to explain what and why vs. how

SUBJECT LINE

SUBJECT LINE

Use the imperative mood for the subject line
like git is doing it

```
Merge branch 'myfeature'
```

Always complete the following:
If applied, this commit will *your subject line here*

SUBJECT EXAMPLES

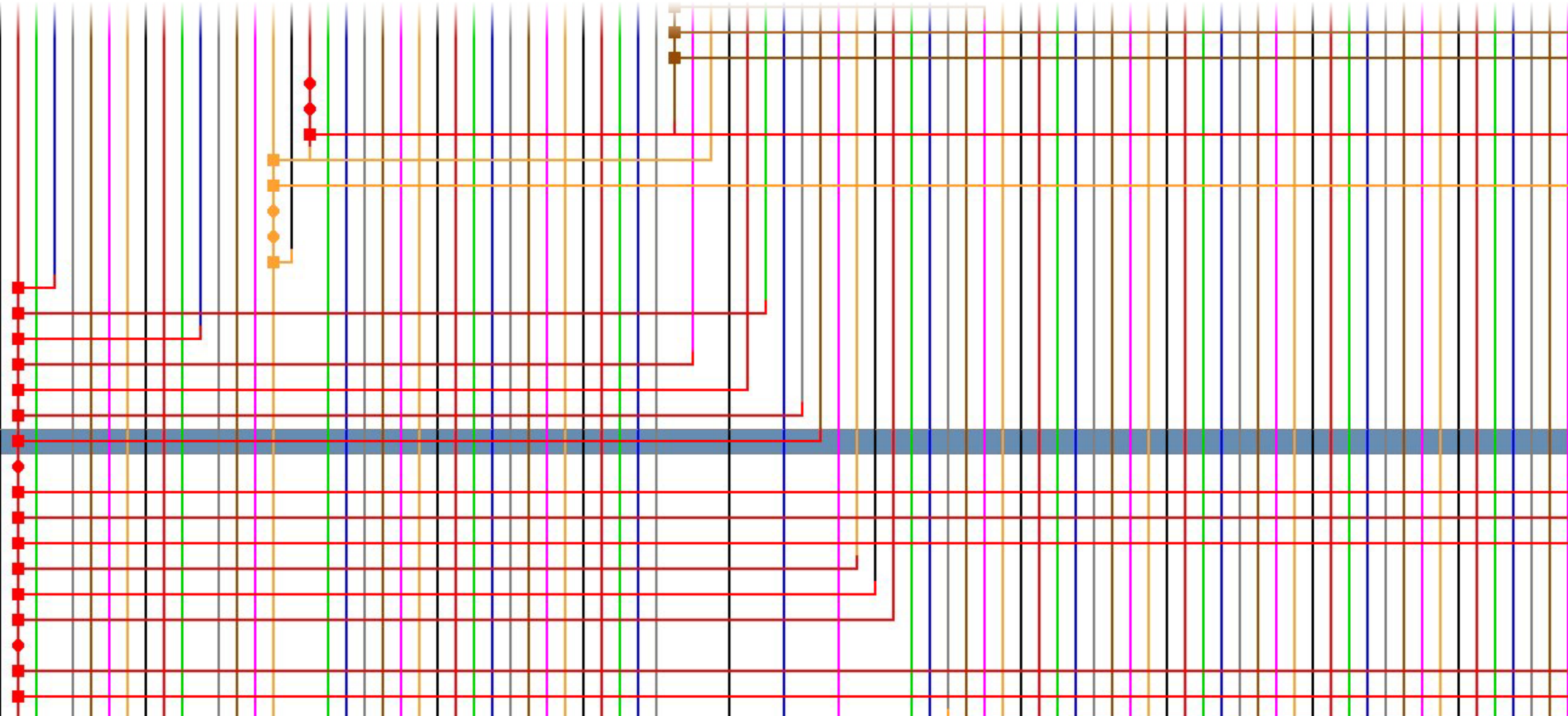
Fixed bug #123
Changing behavior of X
More fixes for broken stuff
Added sweet new API methods

this commit will Merge branch 'feature-x'
this commit will Update the getting started documentation
this commit will Remove all deprecated methods
this commit will Prepare version 1.0.0

FULL EXAMPLE

```
Summarize changes in around 50 characters or less

More detailed explanatory text, if necessary. Wrap it to about 72
characters or so. In some contexts, the first line is treated as the
subject of the commit and the rest of the text as the body. The
blank line separating the summary from the body is critical (unless
you omit the body entirely); various tools like `log`, `shortlog`
and `rebase` can get confused if you run the two together.

Explain the problem that this commit is solving. Focus on why you
are making this change as opposed to how (the code explains that).
Are there side effects or other unintuitive consequenses of this
change? Here's the place to explain them.

Further paragraphs come after blank lines.

 - Bullet points are okay, too

 - Typically a hyphen or asterisk is used for the bullet, preceded
   by a single space, with blank lines in between, but conventions
   vary here

If you use an issue tracker, put references to them at the bottom,
like this:

Resolves: #123
See also: #456, #789
```

GIT GRAPH



GIT GRAPH

> Your mind is like this
> git graph my friend.
> When it is agitated it
> becomes difficult to see
> but if you allow it to settle
> the answer becomes clear

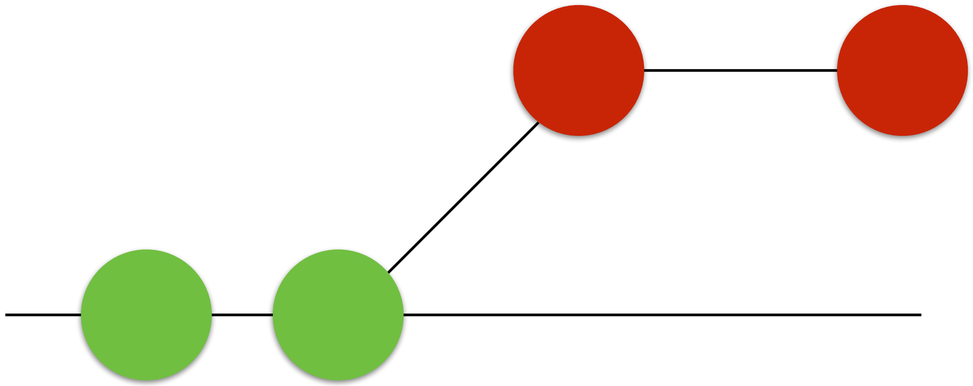
–Master Oogway

GIT MERGE

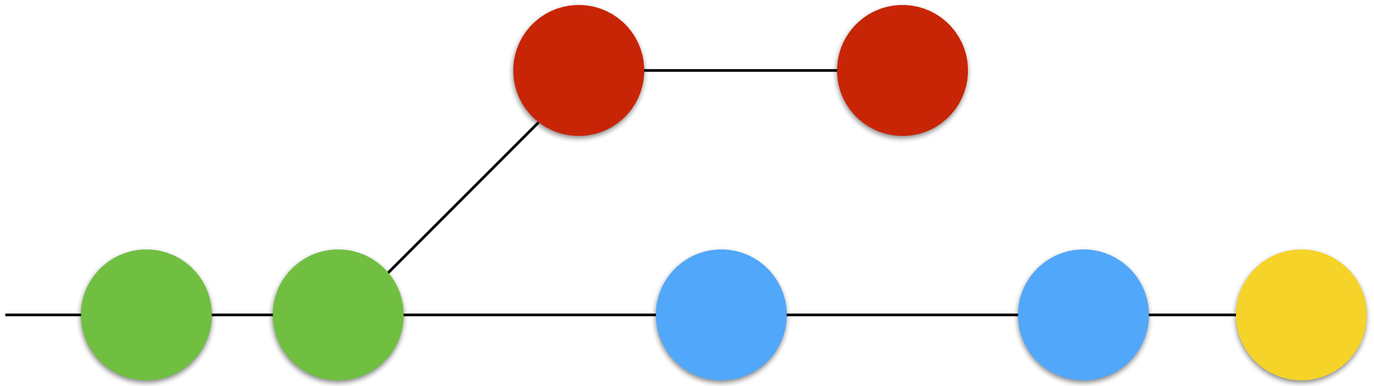
”Merge remote-tracking branch ‘origin/master’„

No big deal and completely safe, but still messes up the log history ***"a bit"***.

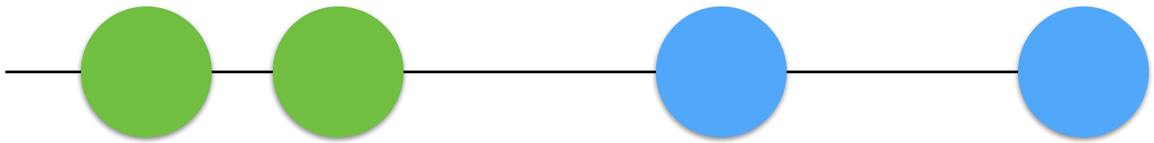
GIT MERGE



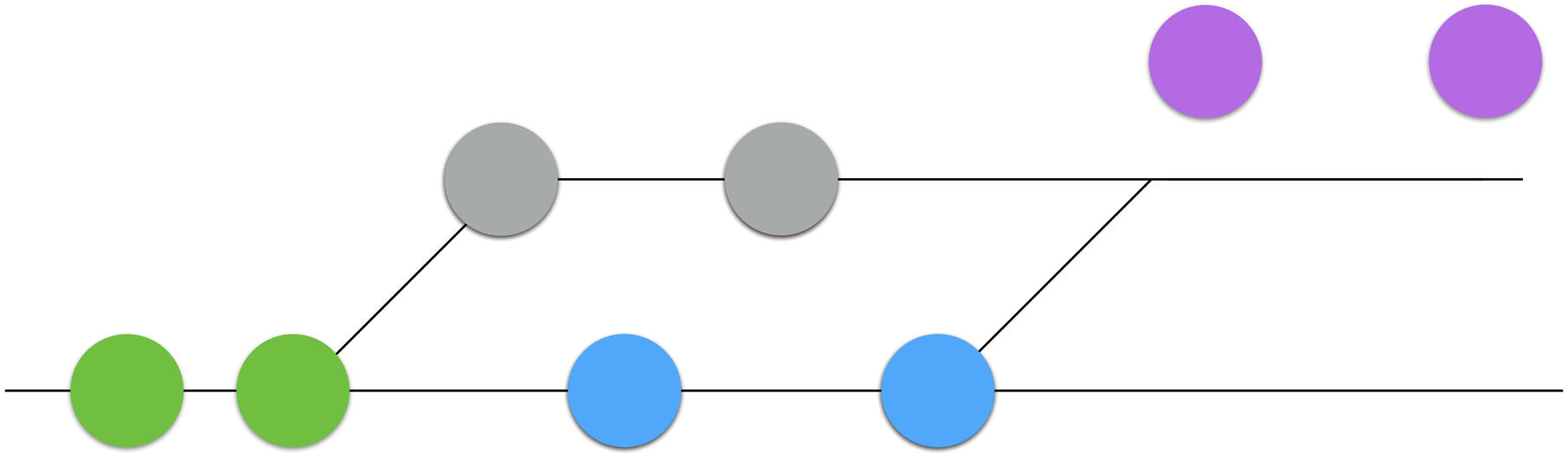
GIT MERGE



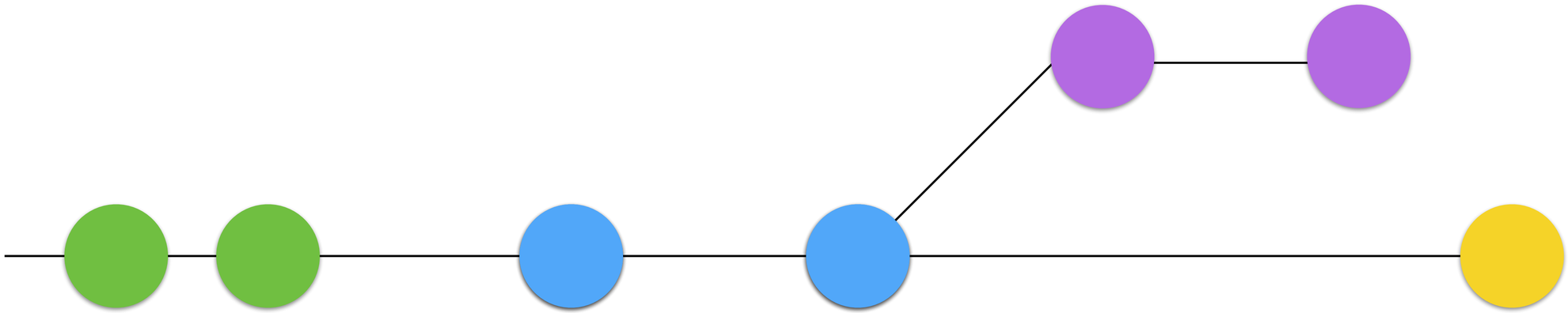
GIT MERGE



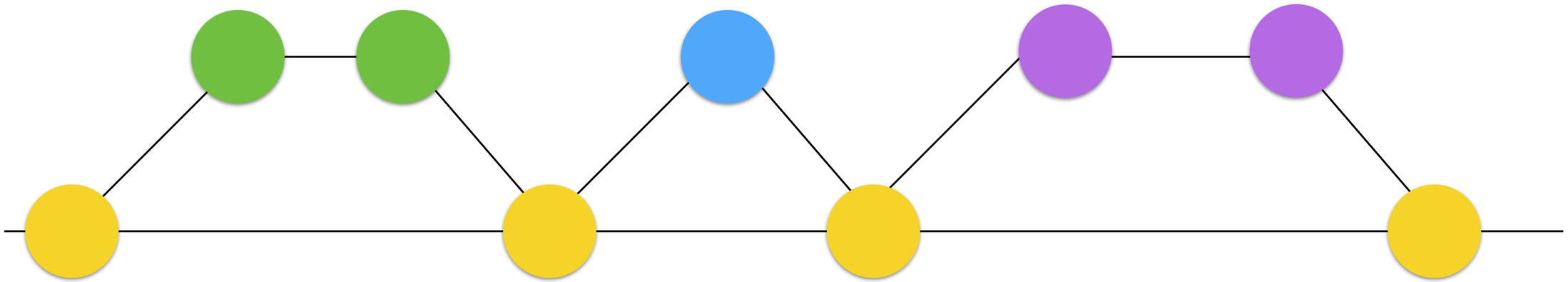
GIT REBASE



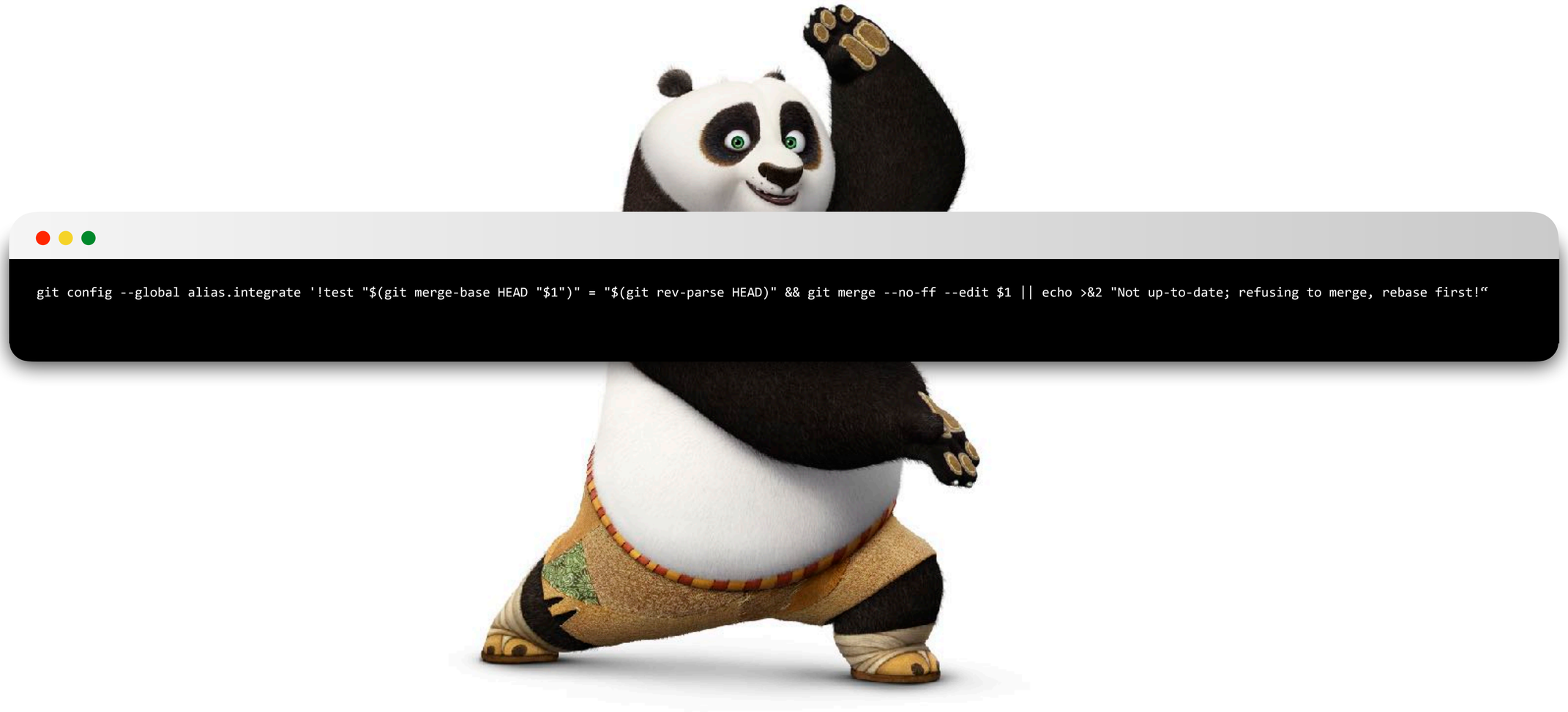
GIT REBASE



GIT REBASE



GIT LOG



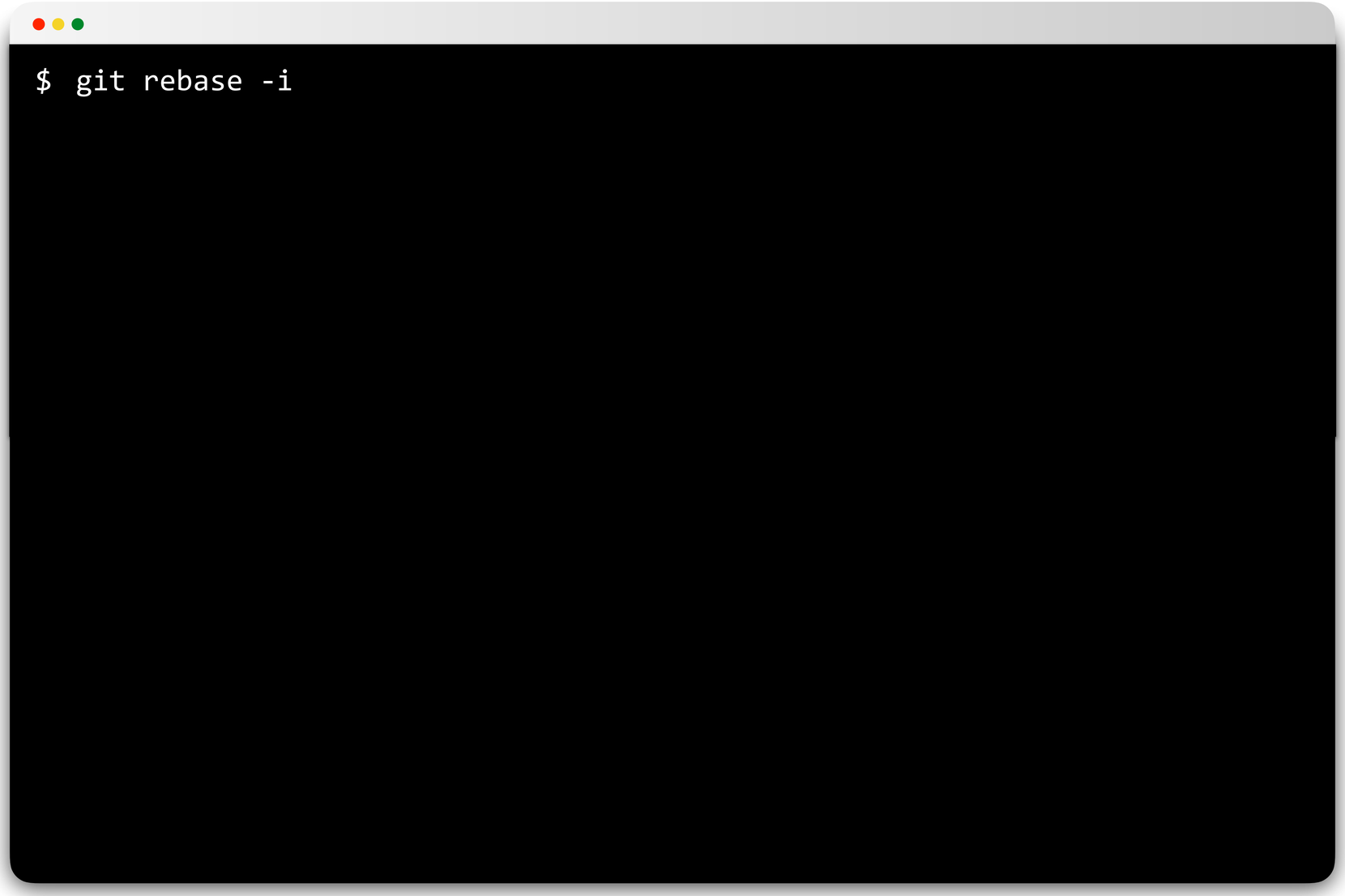
```
git config --global alias.integrate '!test "$(git merge-base HEAD "$1")" = "$(git rev-parse HEAD)" && git merge --no-ff --edit $1 || echo >&2 "Not up-to-date; refusing to merge, rebase first!“
```

GIT REBASE

> *Ahh, but the bliss of rebasing isn't without its drawbacks, which can be summed up in a single line:*
>
> *Do not rebase commits that exist outside your repository*
>
> *If you follow that guideline, you'll be fine. If you don't, people will hate you, and you'll be scorned by friends and family.*

–git book

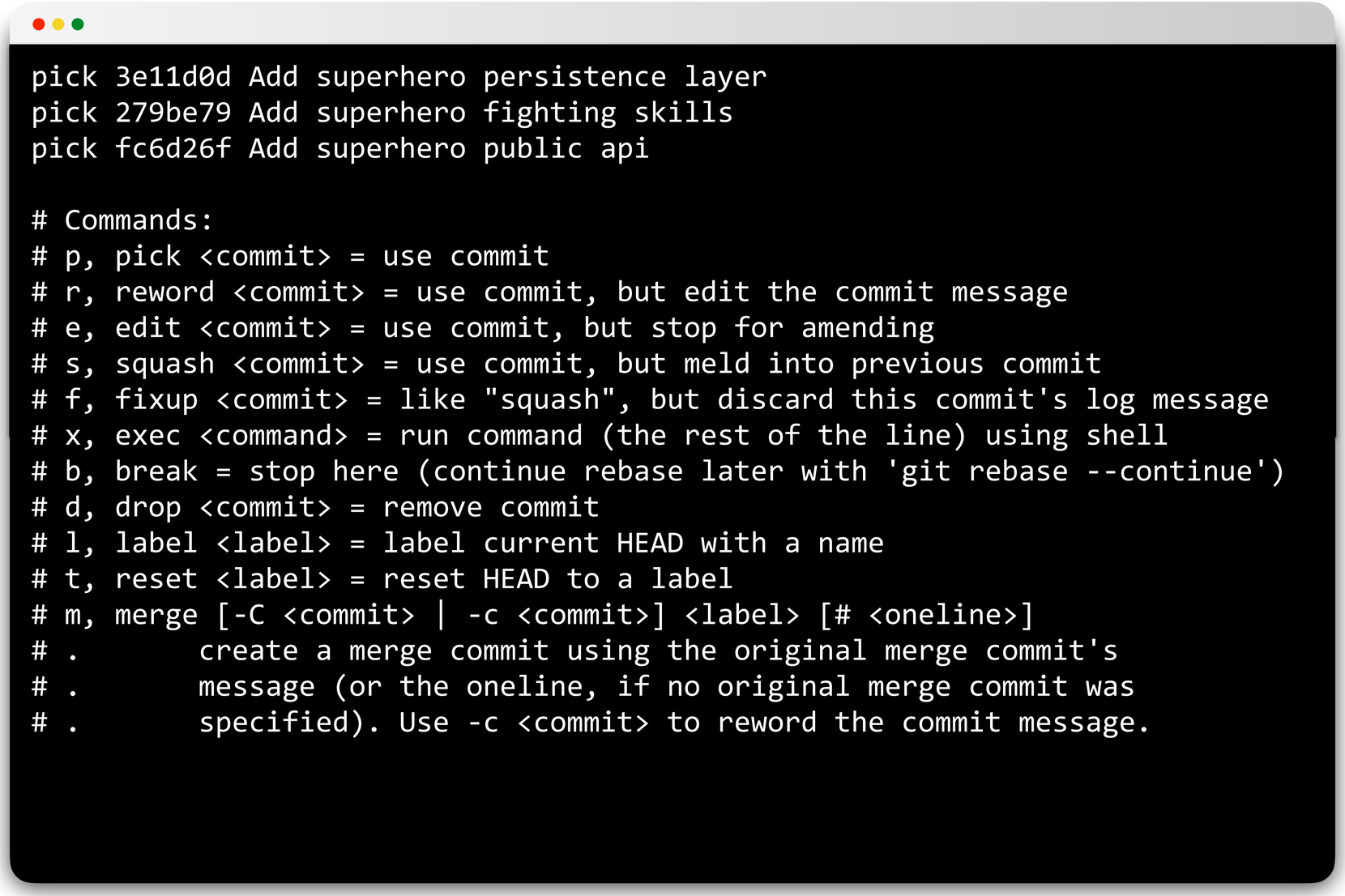
GIT REBASE



GIT REBASE

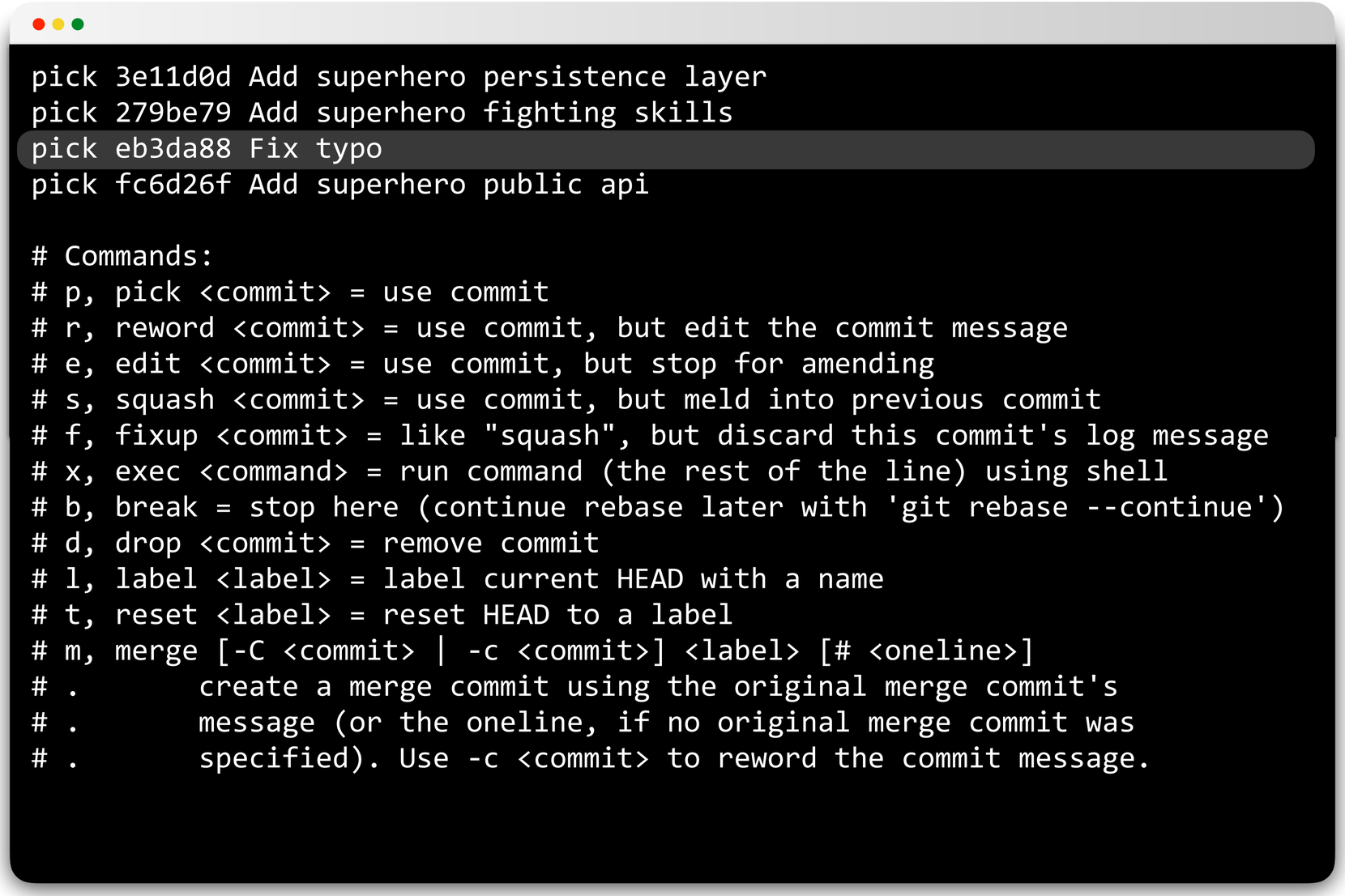
```
pick 3e11d0d Add superhero persistence layer
pick 279be79 Add superhero fighting skills
pick fc6d26f Add superhero public api
pick eb3da88 Fix typo

# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
```

GIT REBASE



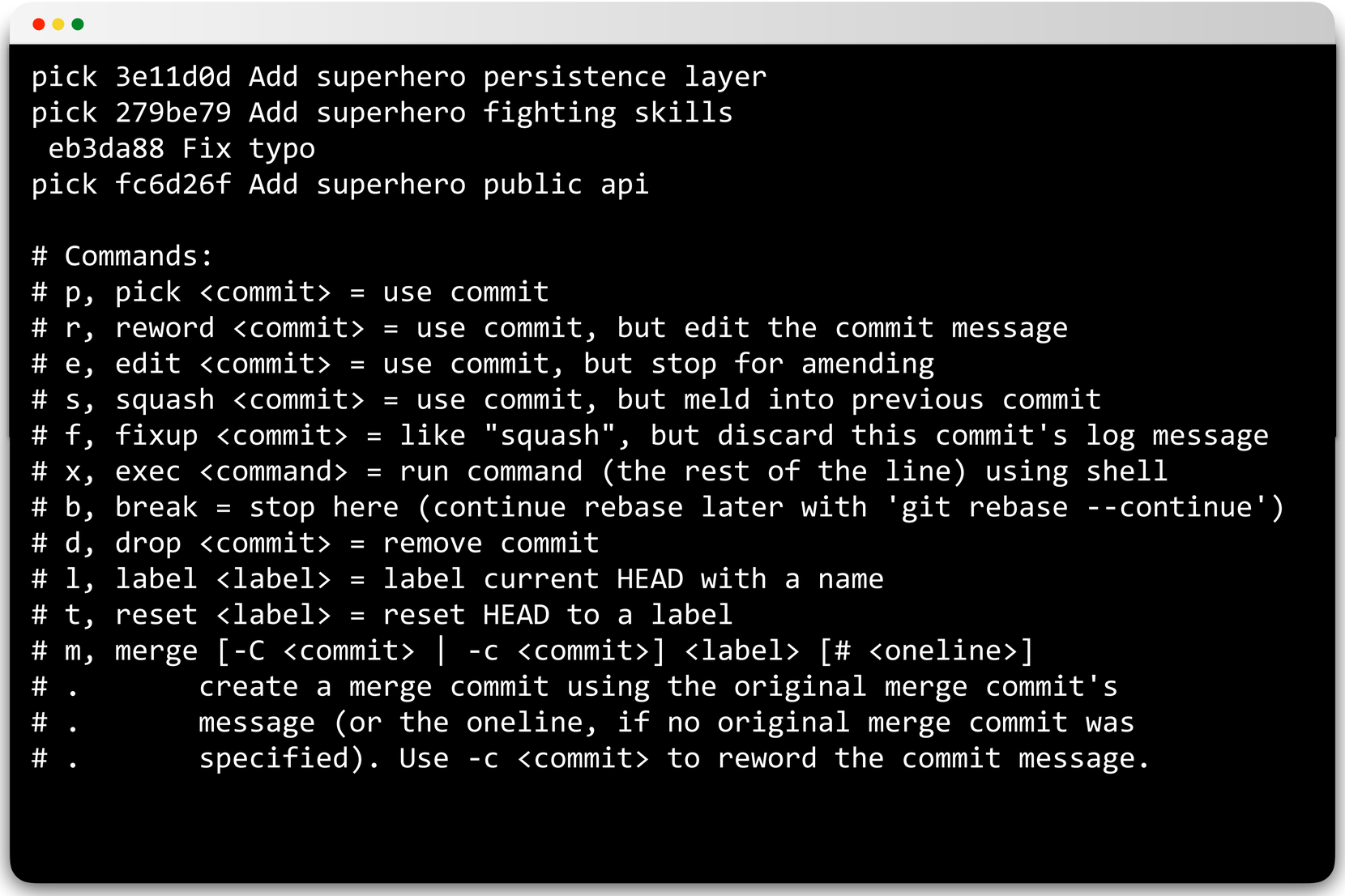
```
pick 3e11d0d Add superhero persistence layer
pick 279be79 Add superhero fighting skills
pick fc6d26f Add superhero public api

# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
```

GIT REBASE



```
pick 3e11d0d Add superhero persistence layer
pick 279be79 Add superhero fighting skills
pick eb3da88 Fix typo
pick fc6d26f Add superhero public api

# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
```

GIT REBASE

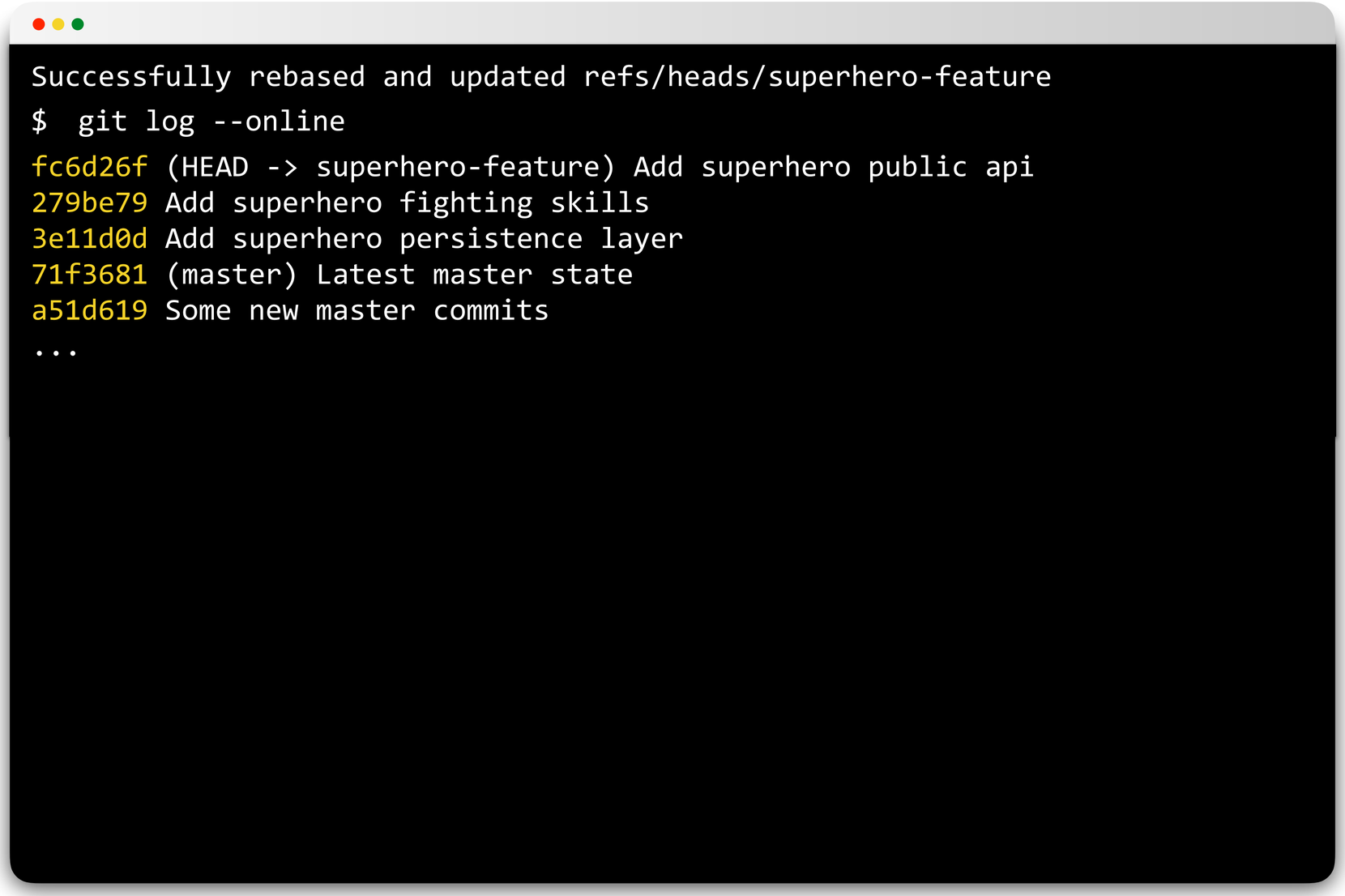


```
pick 3e11d0d Add superhero persistence layer
pick 279be79 Add superhero fighting skills
 eb3da88 Fix typo
pick fc6d26f Add superhero public api

# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
```

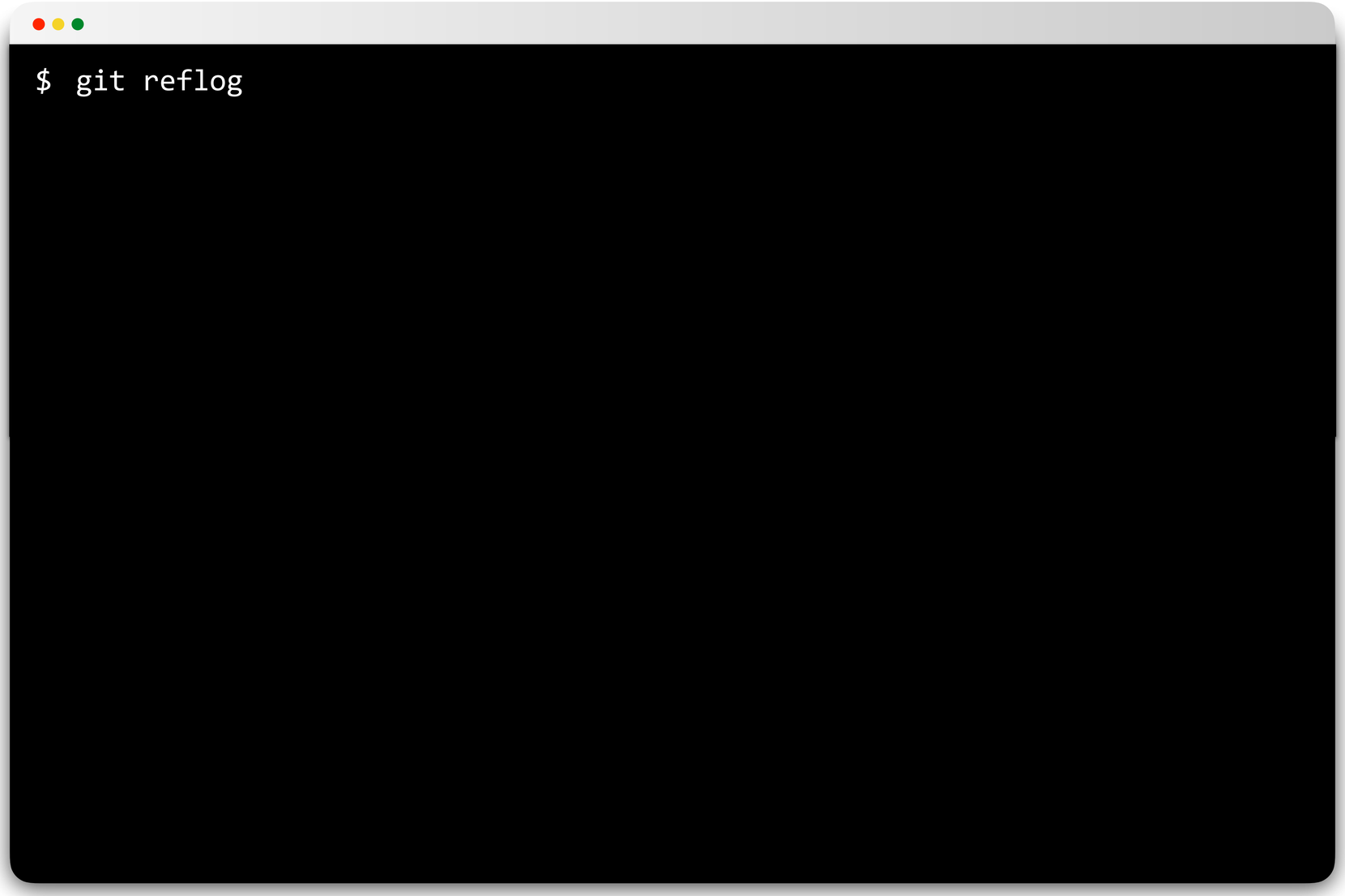
GIT REBASE

```
pick 3e11d0d Add superhero persistence layer
pick 279be79 Add superhero fighting skills
fixup eb3da88 Fix typo
pick fc6d26f Add superhero public api

# Commands:
# p, pick <commit> = use commit
# r, reword <commit> = use commit, but edit the commit message
# e, edit <commit> = use commit, but stop for amending
# s, squash <commit> = use commit, but meld into previous commit
# f, fixup <commit> = like "squash", but discard this commit's log message
# x, exec <command> = run command (the rest of the line) using shell
# b, break = stop here (continue rebase later with 'git rebase --continue')
# d, drop <commit> = remove commit
# l, label <label> = label current HEAD with a name
# t, reset <label> = reset HEAD to a label
# m, merge [-C <commit> | -c <commit>] <label> [# <oneline>]
# .       create a merge commit using the original merge commit's
# .       message (or the oneline, if no original merge commit was
# .       specified). Use -c <commit> to reword the commit message.
```

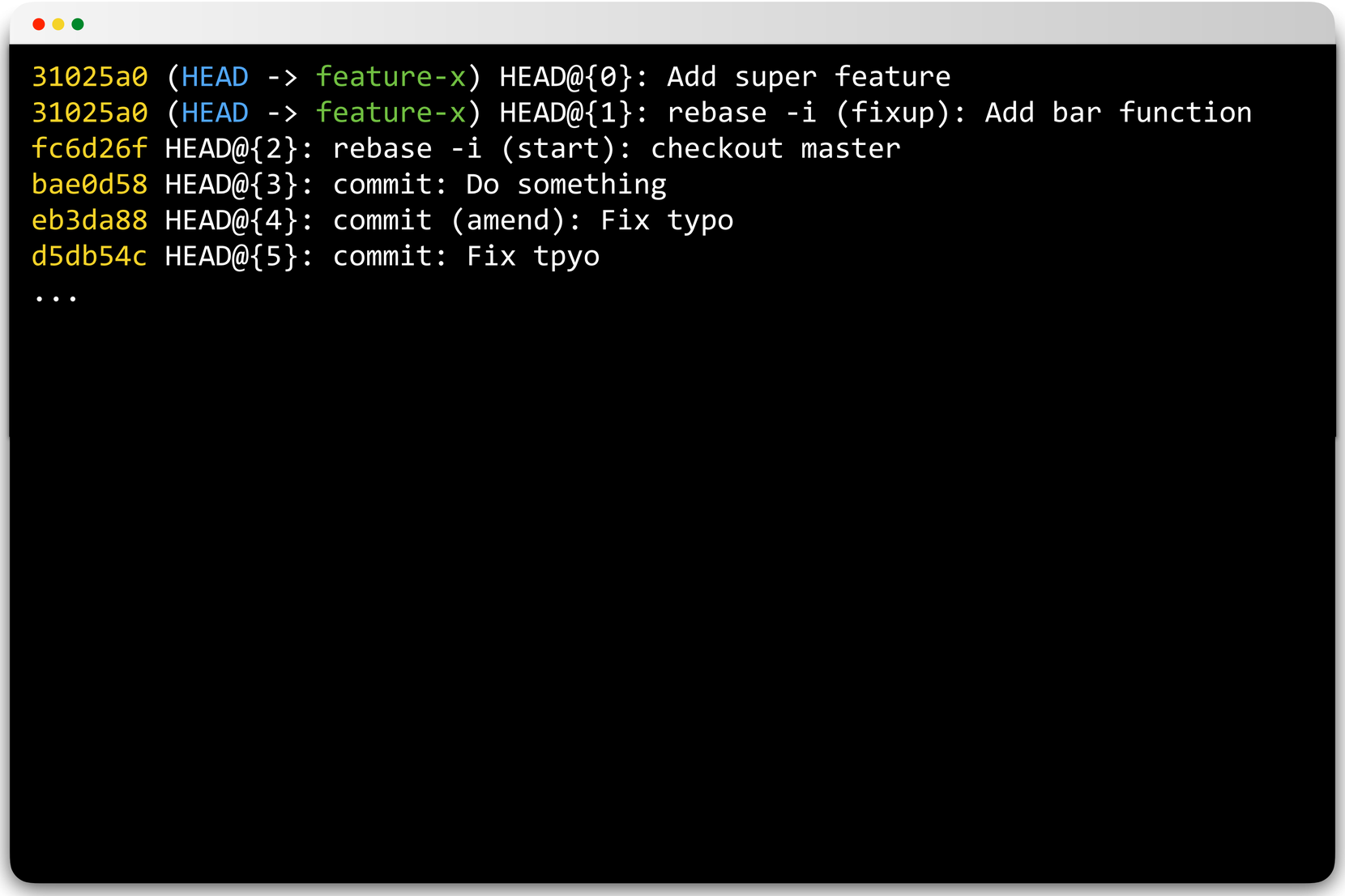
GIT REBASE


Successfully rebased and updated refs/heads/superhero-feature
$ git log --online
fc6d26f (HEAD -> superhero-feature) Add superhero public api
279be79 Add superhero fighting skills
3e11d0d Add superhero persistence layer
71f3681 (master) Latest master state
a51d619 Some new master commits
...

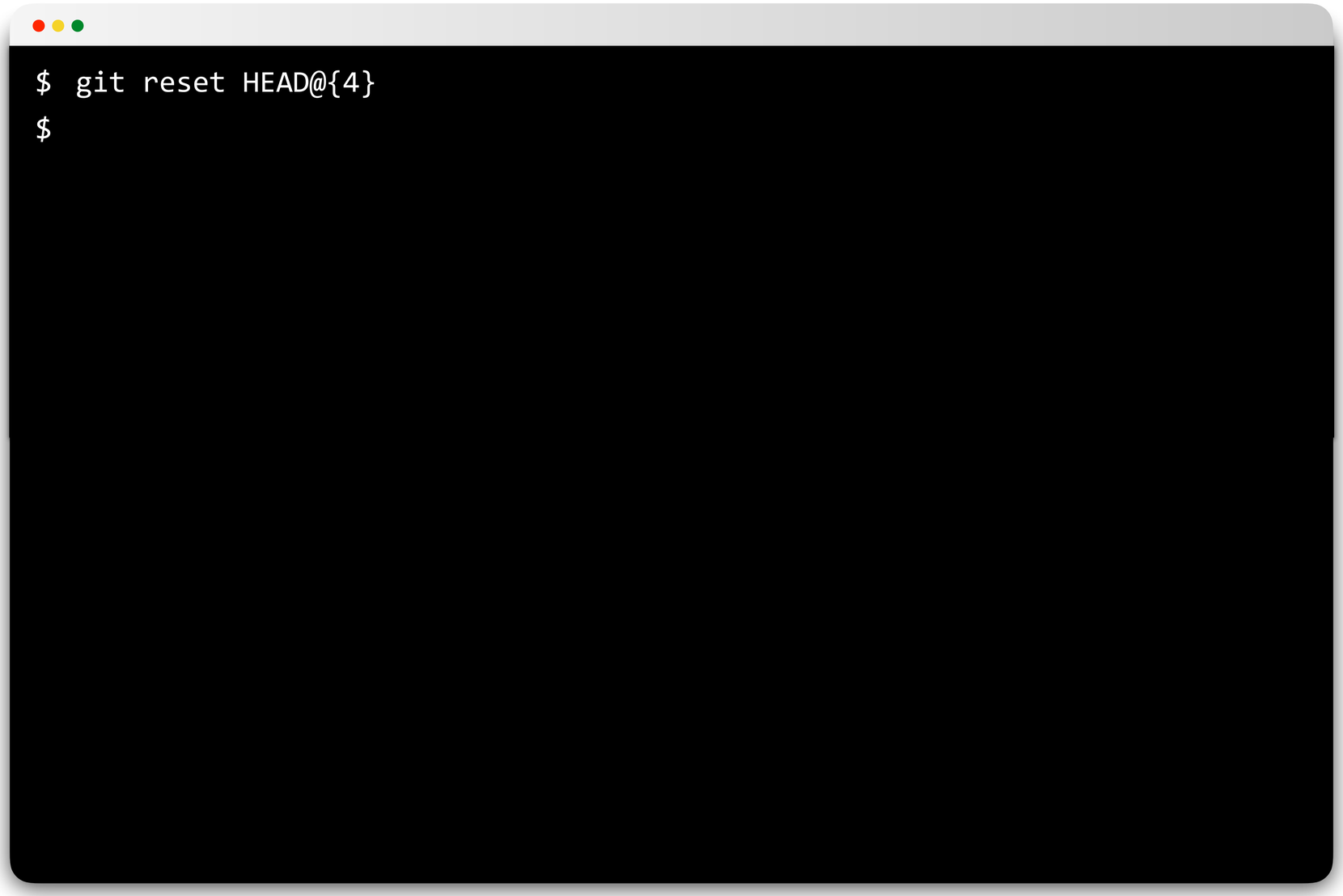
GIT REFLOG



GIT BEFLOG


31025a0 (HEAD -> feature-x) HEAD@{0}: Add super feature
31025a0 (HEAD -> feature-x) HEAD@{1}: rebase -i (fixup): Add bar function
fc6d26f HEAD@{2}: rebase -i (start): checkout master
bae0d58 HEAD@{3}: commit: Do something
eb3da88 HEAD@{4}: commit (amend): Fix typo
d5db54c HEAD@{5}: commit: Fix tpyo
...

GIT BEFLOG



FORCE PUSH

```
$ git push --force-with-lease
```

GIT HOOKS

HOOKS

Local

- commit-msg
- post-checkout
- post-merge
- pre-commit
- pre-push

Remote

- pre-receive
- update
- post-receive

PROBLEMS

- Not activated by default
- Manual install
- Execution may system dependent (bash)
- Difficult to share hooks cross project or cross team

CaptainHookPHP/captainhook

CONFIGURATION

```
{
    "commit-msg": {"enabled": true...},
    "pre-commit": {"enabled": true...},
    "pre-push": {"enabled": false...}
}
```

CONFIGURATION

```
{
    "commit-msg": {
        "enabled": true,
        "actions": [
            {
                "action": „\\CaptainHook\\App\\Hook\\Message\\Action\\Beams",
                "options": {
                    "subjectLength": 50,
                    "bodyLineLength": 72
                }
            }
        ]
    },
    "pre-commit": {"enabled": true...},
    "pre-push": {"enabled": false...}
}
```

CONFIGURATION

```
{
    "commit-msg": {"enabled": true...},
    "pre-commit": {"enabled": true...},
    "pre-push": {"enabled": false…}
}
```

CONFIGURATION

```
{
    "commit-msg": {"enabled": true...},
    "pre-commit": {
        "enabled": true,
        "actions": [
            {
                "action": „\\CaptainHook\\App\\Hook\\PHP\\Action\\Linting",
                "options": []
            },
            {
                "action": "phpcs --standard=psr2 src",
                "options": []
            }
        ]
    },
    "pre-push": {"enabled": false...}
}
```

CONFIGURATION

```
{
    "commit-msg": {"enabled": true...},
    "pre-commit": {
        "enabled": true,
        "actions": [
            {
                "action": „\\CaptainHook\\App\\Hook\\PHP\\Action\\Linting",
                "options": []
            },
            {
                "action": "phpcs --standard=psr2 src",
                "options": []
            }

        ]
    },
    "pre-push": {"enabled": false...}
}
```

CONFIGURATION

```
{
    "commit-msg": {"enabled": true...},
    "pre-commit": {
        "enabled": true,
        "actions": [
            {
                "action": "\\CaptainHook\\App\\Hook\\PHP\\Action\\Linting",
                "options": []
            },
            {
                "action": "phpcs --standard=psr2 src",
                "options": []
            },
            {
                "action": „\\CaptainHook\\App\\Hook\\PHP\\Action\\TestCoverage",
                "options": {
                    "minCoverage": 75
                }
            }
        ]
    },
    "pre-push": {"enabled": false...}
}
```

demo

EXTEND

- Commit message "Rules"
- Custom Actions
- Static methods
- Commands

RULES

```php
<?php
namespace Acme\GitHook;

use \CaptainHook\App\Hook\Message\Rule;
use SebastianFeldmann\Git\CommitMessage;

class DoNotYell implements Rule
{
    /**
     * Make sure nobody yells in commit messages.
     *
     * @param  \SebastianFeldmann\Git\CommitMessage $message
     * @return bool
     */
    public function pass(CommitMessage $message): bool
    {
        return $message->getContent() !== strtoupper($message->getContent());
    }

    /**
     * Returns error message in case of 'pass' fails.
     *
     * @return string
     */
    public function getHint() : string
    {
        return 'YOU SHOULD NOT YELL IN COMMIT MESSAGES!!!';
    }
}
```

RULES

```
{
  "commit-msg": {
    "enabled": true,
    "actions": [
      {
        "action": "\\CaptainHook\\App\\Hook\\Message\\Action\\Rules",
        "options": [
          "\\Acme\\GitHook\\DoNotYell"
        ]
      }
    ]
  }
}
```

sebastianfeldmann/captainhook

RULES

```
{
  "commit-msg": {
    "enabled": true,
    "actions": [
      {
        "action": „\\CaptainHool\\App\\Hook\\Message\\Action\\Rules",
        "options": [
          "\\Acme\\GitHook\\DoNotYell",
          "\\CaptainHook\\App\\Hook\\Message\\Rule\\SubjectStartsWithCapitalLetter"
        ]
      }
    ]
  }
}
```

ACTIONS

```php
<?php
namespace Acme\GitHook;

use CaptainHook\App\Config;
use CaptainHook\App\Console\IO;
use CaptainHook\App\Hook\Action;
use SebastianFeldmann\Git\Repository;

class TicketIDValidator implements Action
{
    /**
     * Execute the action.
     *
     * @param  \SebastianFeldmann\CaptainHook\Config         $config
     * @param  \SebastianFeldmann\CaptainHook\Console\IO     $io
     * @param  \SebastianFeldmann\Git\Repository             $repository
     * @param  \SebastianFeldmann\CaptainHook\Config\Action  $action
     * @throws \Exception
     */
    public function execute(Config $config, IO $io, Repository $repository, Config\Action $action) {...}

    private function findTicketIdsIn($text): array {...}

    private function isValidTicketId($id): bool {...}
}
```

ACTIONS

```php
<?php
namespace Acme\GitHook;

use SebastianFeldmann\CaptainHook\Config;
use SebastianFeldmann\CaptainHook\Console\IO;
use SebastianFeldmann\CaptainHook\Hook\Action;
use SebastianFeldmann\Git\Repository;

class TicketIDValidator implements Action
{
    /**
     * Execute the action.
     *
     * @param  \SebastianFeldmann\CaptainHook\Config         $config
     * @param  \SebastianFeldmann\CaptainHook\Console\IO     $io
     * @param  \SebastianFeldmann\Git\Repository             $repository
     * @param  \SebastianFeldmann\CaptainHook\Config\Action  $action
     * @throws \Exception
     */
    public function execute(Config $config, IO $io, Repository $repository, Config\Action $action)
    {
        $message = $repository->getCommitMsg();

        foreach ($this->findTicketIdsIn($message->getContent()) as $id) {
            if (!$this->isValidTicketId($id)) {
                throw new \Exception('invalid ticket ID: ' . $id);
            }
        }
    }

    /**
```

ACTIONS

```
{
  "commit-msg": {
    "enabled": true,
    "actions": [
      {
        "action": „\\Acme\\GitHook\\TicketIDValidator“,
        "options": {
          "key": "value",
          "use": "what you need"
        }
      }
    ]
  }
}
```

EXISTING PLUGINS

- bitExpert/captainhook-validateauthor
- bitExpert/captainhook-rejectpush

THANK YOU