MASTERING GIT

Sebastian Feldmann

CHECK24

@movetodevnull

sebastianfeldmann

MASTERING GIT

- Dos & Don’ts
- Tips & Tricks
- Hooks

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

Make 'getCommandLine' dependency explicit by adding the abstract meth... ··

sebastianfeldmann committed on 21 Feb ✓

6513235

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
```

TIPS & TRICKS

AUTOCOMPLETION

Download the completion script

```
$ curl -O https://raw.github.com/git/git/master/contrib/completion/git-completion.bash
$ mv git-completion.bash .git-completion.bash
```

Then in your `.bash_profile` add the following

```
if [ -f ~/.git-completion.bash ]; then
    . ~/.git-completion.bash
fi
```

.GIT TEMPLATE

- Default `/usr/local/share/git-core/templates`

- Custom hooks

- Custom excludes

```
$ git config --global init.templatedir '~/.git_template/template'
```

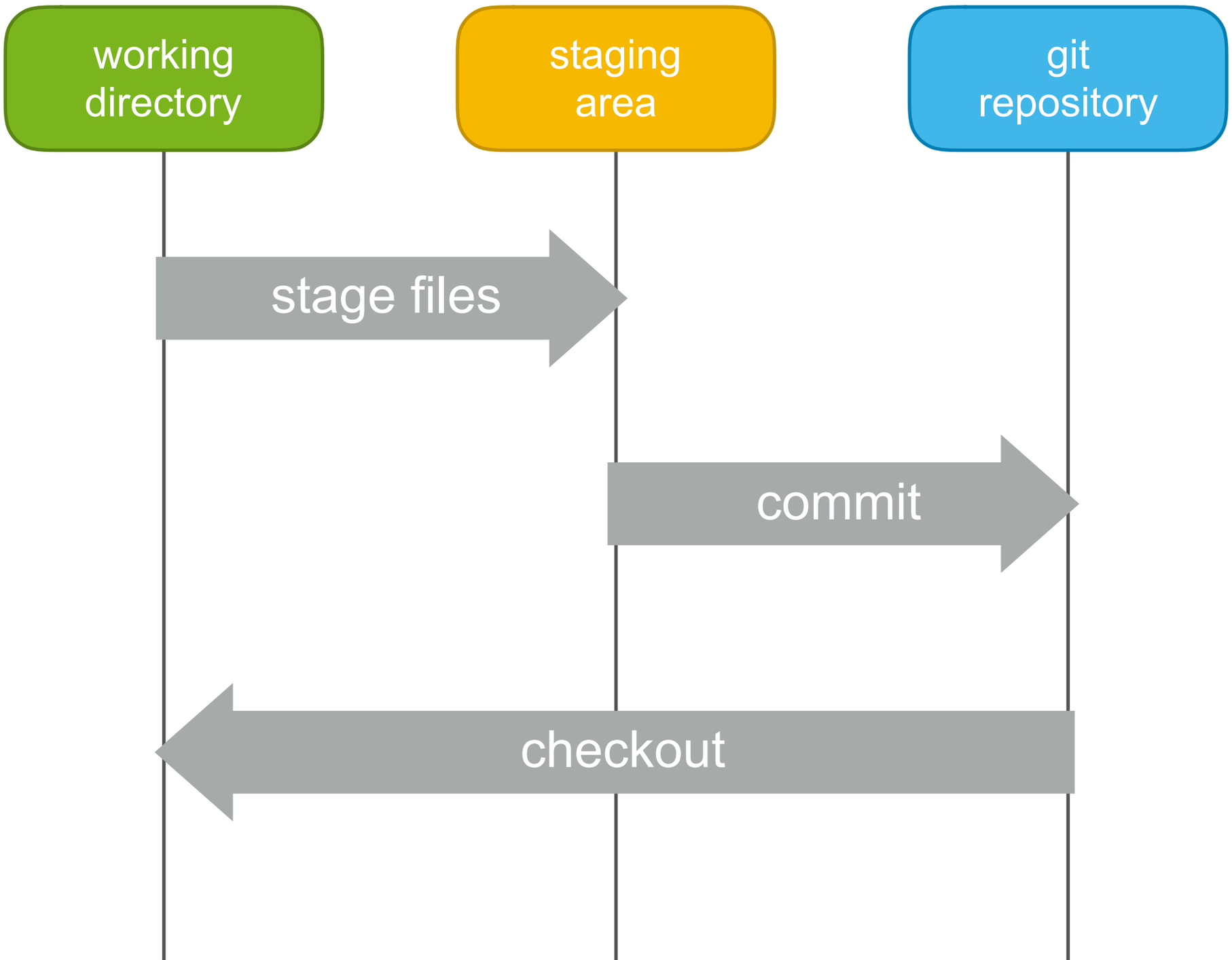
OS ALIASES

```
# I'm lazy as hell
alias g='git'

# git status in a flash
alias gs='git status'
alias gss='git status -s'

# go to repository root directory
alias gr='[! -z `git rev-parse --show-cdup`] && cd `git rev-parse --show-cdup || pwd`'
```

REMINDER


working directory
staging area
git repository
stage files
commit
checkout

RESET FILES

Return to a previous version and discard all changed

```
$ git reset --hard {{some-commit}}
```

Return to a previous version, changes are unstaged

```
$ git reset {{some-commit}}
```

Return to a previous version, changes are staged

```
$ git reset --soft {{some-commit}}
```

GIT ALIASES

Add via terminal or edit ~/.gitconfig

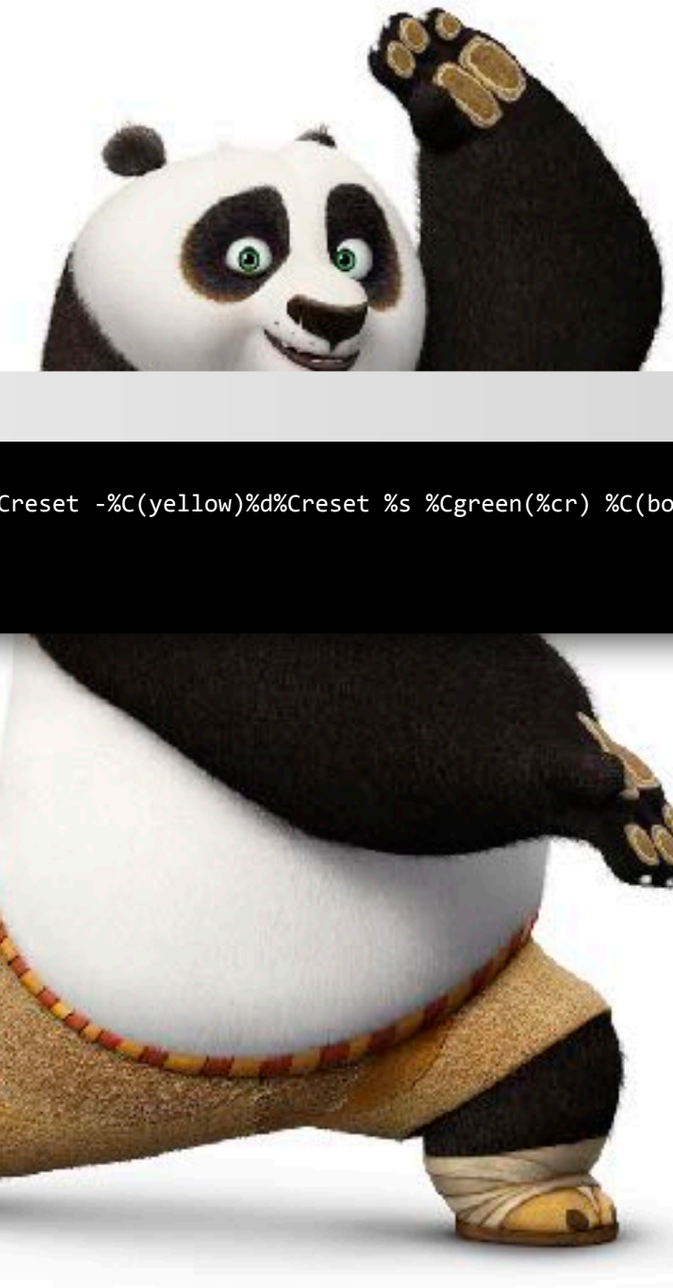
```
$ git config --global alias.unstage "reset HEAD"
$ git config --global alias.undo-commit "reset --soft"
$ git config --global alias.clean "reset --hard"
$ git config --global alias.b "branch"
$ git config --global alias.c "commit"
$ git config --global alias.p "pull –rebase“
```

LOG

`--author="Sebastian"`	Only show commits made by a certain author
`--name-only`	Only show names of files that changed
`--oneline`	Show commit data compressed to one line
`--graph`	Show dependency tree for all commits
`--reverse`	Show commits in reverse order (Oldest commit first)
`--after`	Show all commits that happened after certain date
`--before`	Show all commits that happened before certain data
`--stat`	Show all commits with some statistics

```
$ git log --author="Sebastian" --after="1 week ago" --oneline
```

LOG



```
$ git log --color --graph --pretty=format:'%Cred%h%Creset -%C(yellow)%d%Creset %s %Cgreen(%cr) %C(bold blue)<%an>%Creset’ --abbrev-commit
```

MORE ALIASES

Add via terminal or edit ~/.gitconfig

```
$ git config --global alias.wtf "log -p"
$ git config --global alias.l "log --oneline --graph"
```

GIT LOG

You can use the regular less command to search

```
/{{your-search-here}}
```

Use lower case **`n`** to navigate to the next occurrence and upper case **`N`** to the previous one.

IGNORE WHITESPACE

You can easily ignore whitespace for diff and blame

```
$ git diff -w
$ git diff --ignore-space-at-eol
$ git blame -w
```

GIT STASH

Store uncommitted changes

```
$ git stash
```

Show stashed changes

```
$ git stash list
```

GIT STASH

Apply latest stashed state and remove from list

```
$ git stash pop
```

Apply any stashed state and keep in list

```
$ git stash apply stash@{n}
```

Clean up the stash

```
$ git stash drop stash@{n}
$ git stash clear
```

demo

PARTIAL ADD

add only some changes in a file

```
$ git add -p
```

PARTIAL ADD

y stage this hunk for the next commit

n do not stage this hunk for the next commit

d do not stage this hunk or any of the later hunks in the file

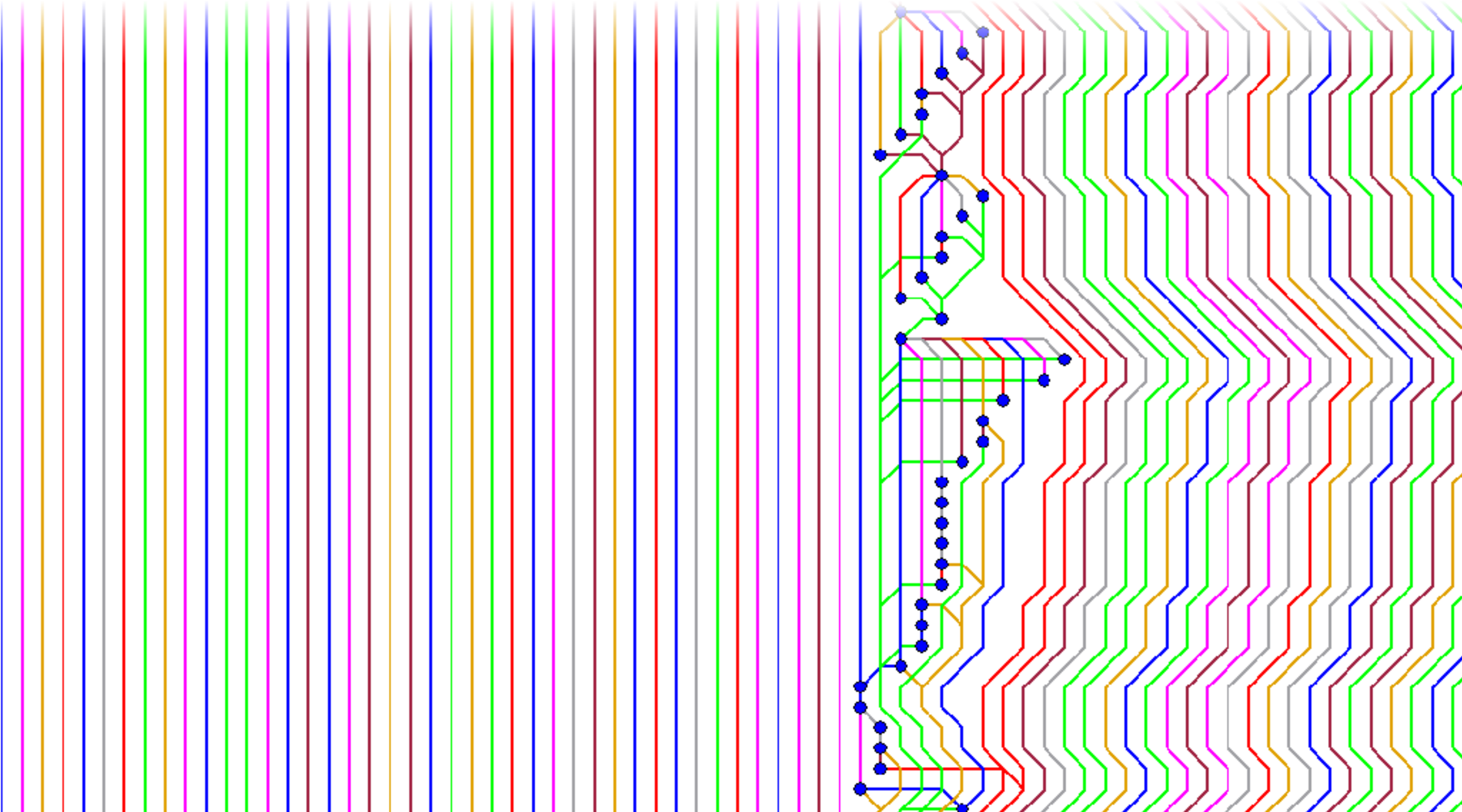
s split the current hunk into smaller hunks

e manually edit the current hunk

? print hunk help

demo

GitFLOW



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 REBASE

”

Ahh, but the bliss of rebasing isn’t without its drawbacks, which can be summed up in a single line:

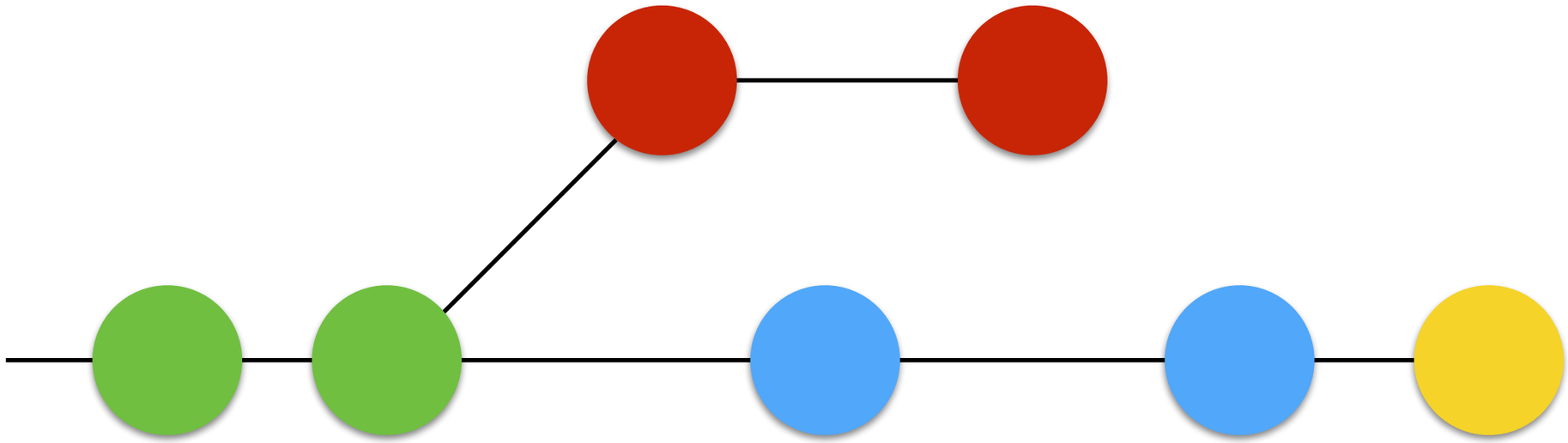
Do not rebase commits that exist outside your repository

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.

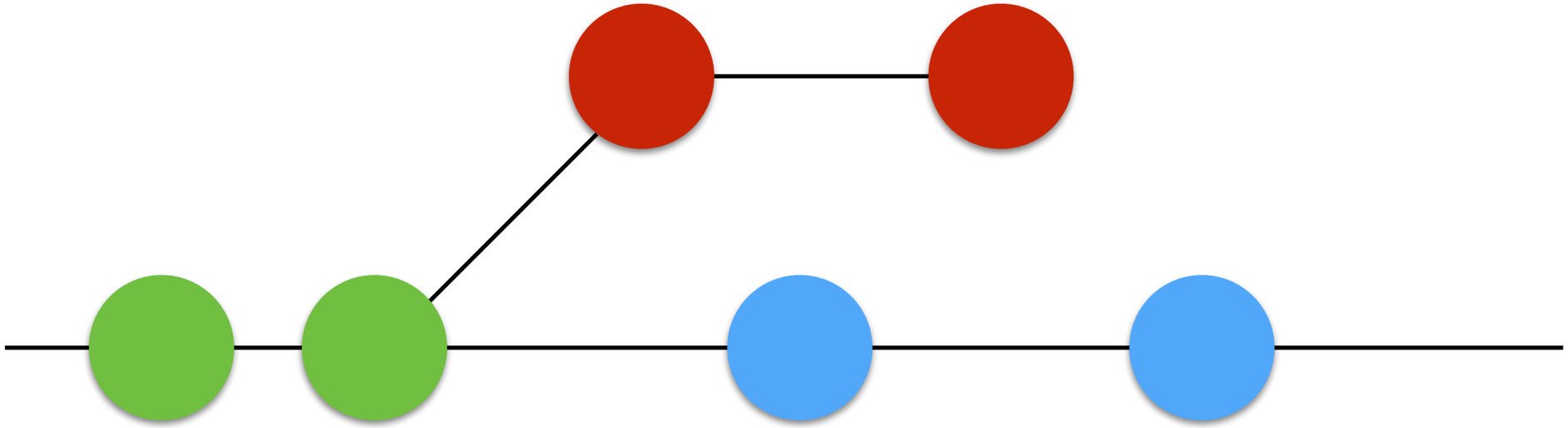
”

–git book

GIT MERGE



GIT REBASE



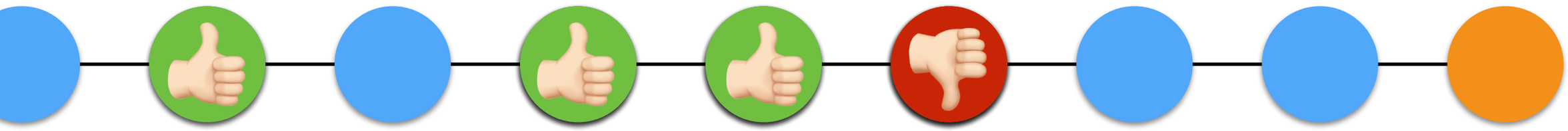
demo

GIT AMEND

```
$ git add path/toFile/withCodingStandardFix.php
$ git commit --amend
```

demo

INVESTIGATE



GIT BISECT

Find commits that break your code fast

```
$ git bisect start
$ git bisect good {{some-commit}}
$ git bisect bad HEAD
```

Test and flag the commit

```
$ git bisect good|bad
```

GIT BISECT

Get a summary report of last bisect run

```
$ git bisect log
```

Go back to the starting point

```
$ git bisect reset
```

GIT BISECT

Use a command to determine good or bad

```
$ git bisect run phpunit
```

demo

GIT HOOKS

HOOKS

Local

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

Remote

- pre-receive
- update
- post-receive

sebastianfeldmann/captainhook

CAPTAINHOOK

- Easy to use
- Configurable
- Shareable configuration
- Extendable

sebastianfeldmann/captainhook

INSTALL

- Install via composer

  `sebastianfeldmann/captainhook`

- Guided setup via CLI

  `vendor/bin/captainhook`

CONFIGURATION

- JSON configuration (captainhook.json)
- Add hook configuration to repository
- Everybody uses the same hooks

sebastianfeldmann/captainhook

CONFIGURATION

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

sebastianfeldmann/captainhook

CONFIGURATION

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

sebastianfeldmann/captainhook

CONFIGURATION

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

sebastianfeldmann/captainhook

CONFIGURATION

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

sebastianfeldmann/captainhook

CONFIGURATION

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

sebastianfeldmann/captainhook

CONFIGURATION

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

sebastianfeldmann/captainhook

demo

EXTEND

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

sebastianfeldmann/captainhook

RULES

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

use SebastianFeldmann\CaptainHook\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!!!';
    }
}
```

sebastianfeldmann/captainhook

RULES

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

sebastianfeldmann/captainhook

RULES

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

sebastianfeldmann/captainhook

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) {...}

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

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

sebastianfeldmann/captainhook

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);
            }
        }
    }
```

sebastianfeldmann/captainhook

ACTIONS

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

sebastianfeldmann/captainhook

THANK YOU

Sebastian Feldmann

phpbu

https://phpbu.de

php User Group Munich

CHECK24

@movetodevnull

sebastianfeldmann

Q & A