git happens

¯_(ツ)_/¯

Felix Breidenstein
<code>monauts

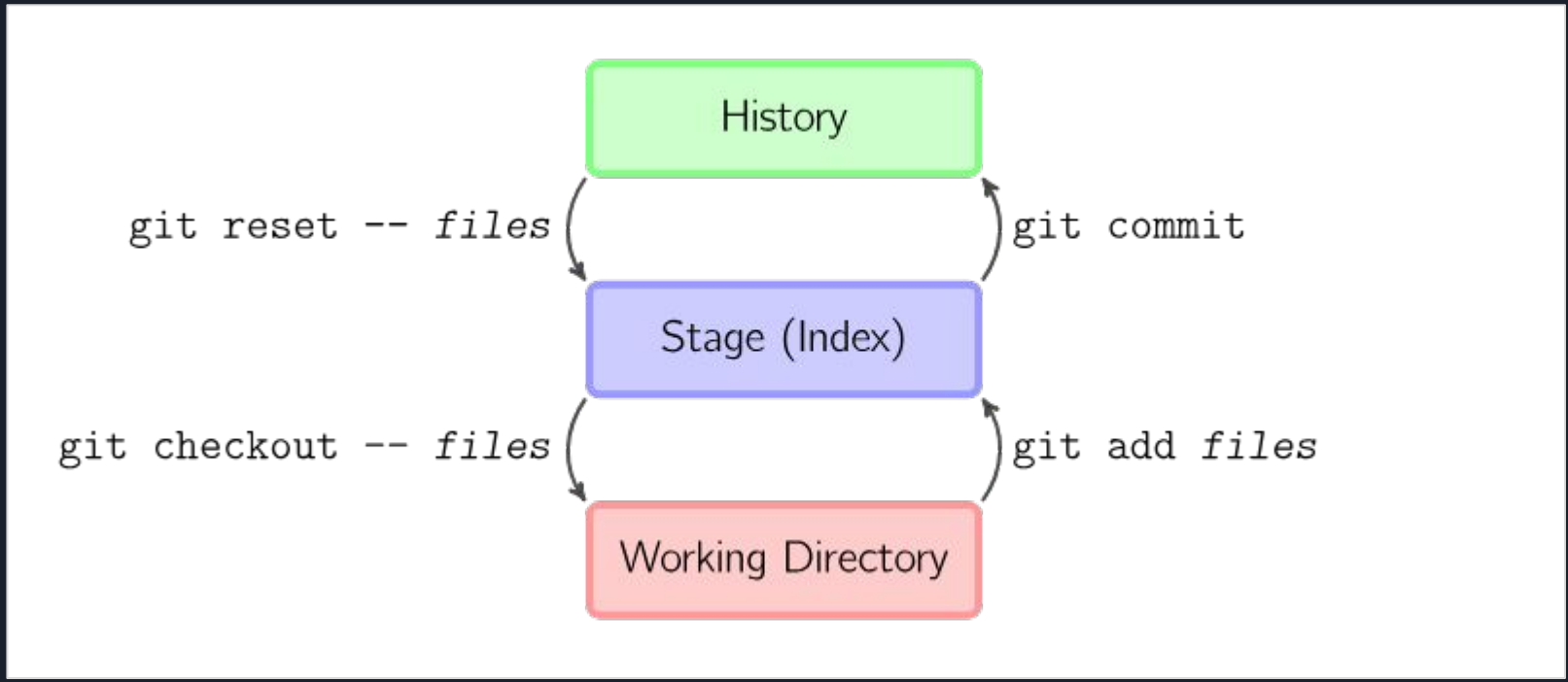
2019-03-07

1. Grundlagen
2. Workflow
3. Unsere CI/CD Pipeline
4. Oh shit git! - Jetzt helfe ich mir selbst

Grundlagen

- Keine Angst vorm Terminal!
  - Kann “alles”
  - Sagt euch wo das Problem ist
  - Weiß meist was ihr als nächstes machen wollt
- GUI und CLI machen zwar meist das gleiche
  - Anleitungen im Netz sind fast immer CLI Befehle
  - GUI benutzt evtl andere Worte / Versteckt Features / Benutzt Defaults
  - GUI kann evtl nicht alles
- Ein Platz zum Austoben
  - github.com/codemonauts/playground

Grundlagen - The three trees



Grundlagen - Dateizustände

Staged	Datei wird in den nächsten commit aufgenommen
Modified	Datei wurde seit dem letzten commit verändert
Unmodified	Datei ist seit dem letzten commit unverändert
Untracked	Datei wird (noch) nicht von git beobachtet
Ignored	Datei wird von git ignoriert

Grundlagen - Commit

- Ein Commit == Ein “Ding”
- Atomare Änderung
  - Wichtig für Cherry-Pick und Revert

- How to Commit-Message:
  - max. 50 Zeichen Zusammenfassung in die erste Zeile
  - <Leerzeile>
  - Body (Fließtext mit Erklärung, Aufzählung, Links, etc)

Faustregel

Wenn die Zusammenfassung nicht in 50 Zeichen passt, sind es wohl 2 Commits.

Grundlagen - Commit

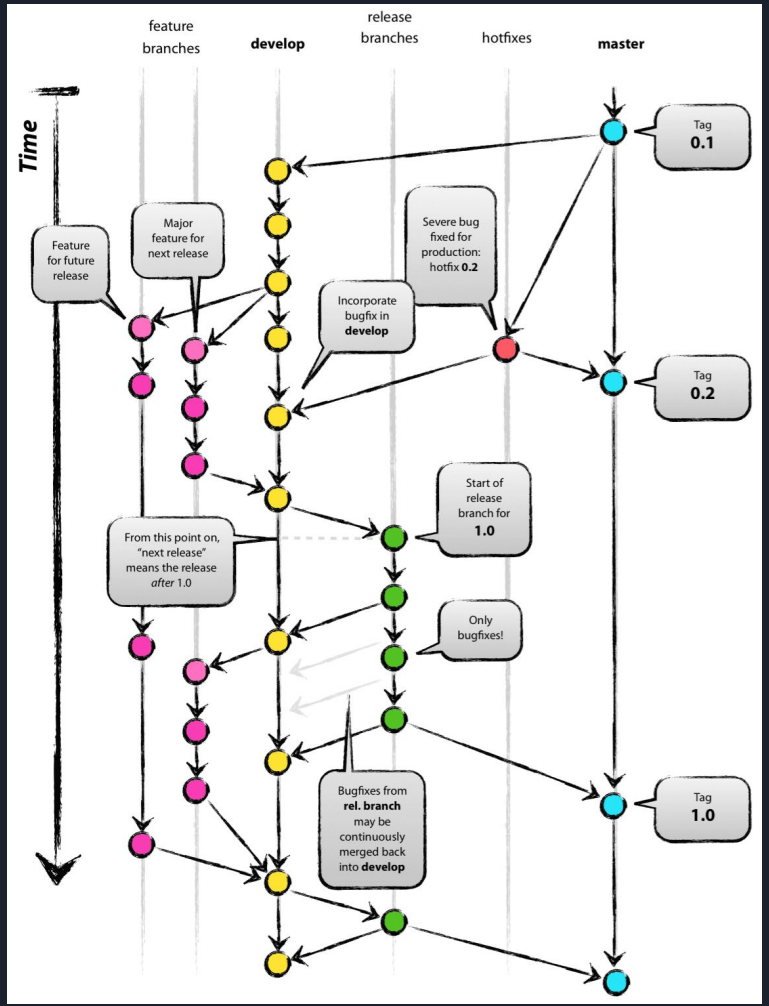
- Regeln
  - Kein auskommentierter Code
  - Keine Backups (index2.php)
  - Keine Formatierung zusammen mit Features
  - Kein umbenennen/verschieben zusammen mit Änderungen

- Sausage Making
  - Commit Often, Perfect Later, Publish Once
  - Räumt auf **bevor** ihr pusht beziehungsweise **bevor** ihr einen PR erstellt

Grundlagen - Branches

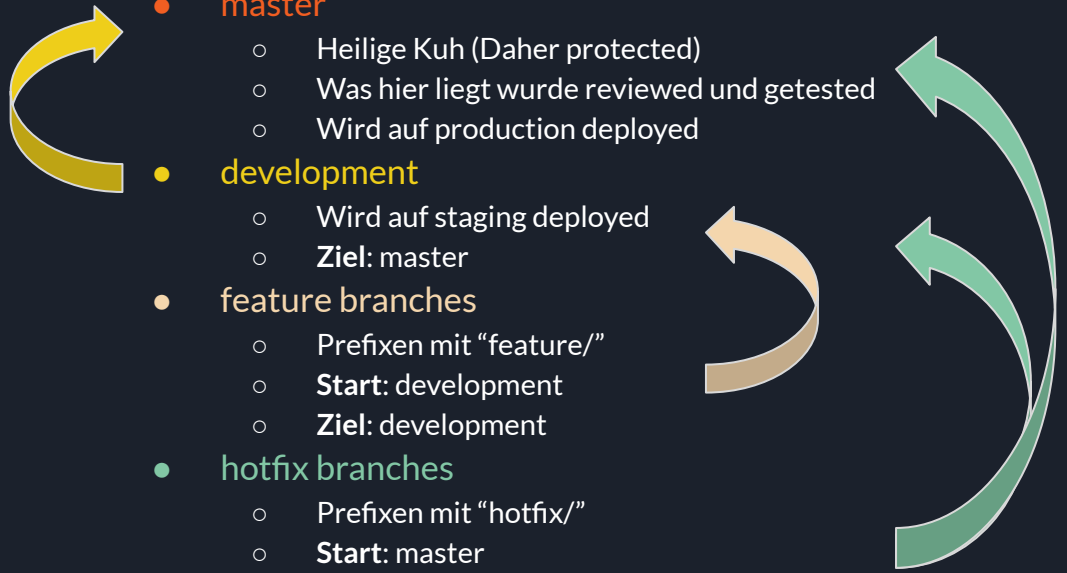
- Für git ist alles das gleiche
- Prefix beachten
  - *feature/foobar* statt *foobar*
- Aus dem Branch abzweigen, in den später auch reingemerged werden soll
- Protected Branches
  - Nachträgliche Access-Control für git

Workflow


feature branches
develop
release branches
hotfixes
master
Time
Tag 0.1
Feature for future release
Major feature for next release
Severe bug fixed for production: hotfix 0.2
Incorporate bugfix in develop
Tag 0.2
Start of release branch for 1.0
From this point on, "next release" means the release after 1.0
Only bugfixes!
Bugfixes from rel. branch may be continuously merged back into develop
Tag 1.0

Workflow

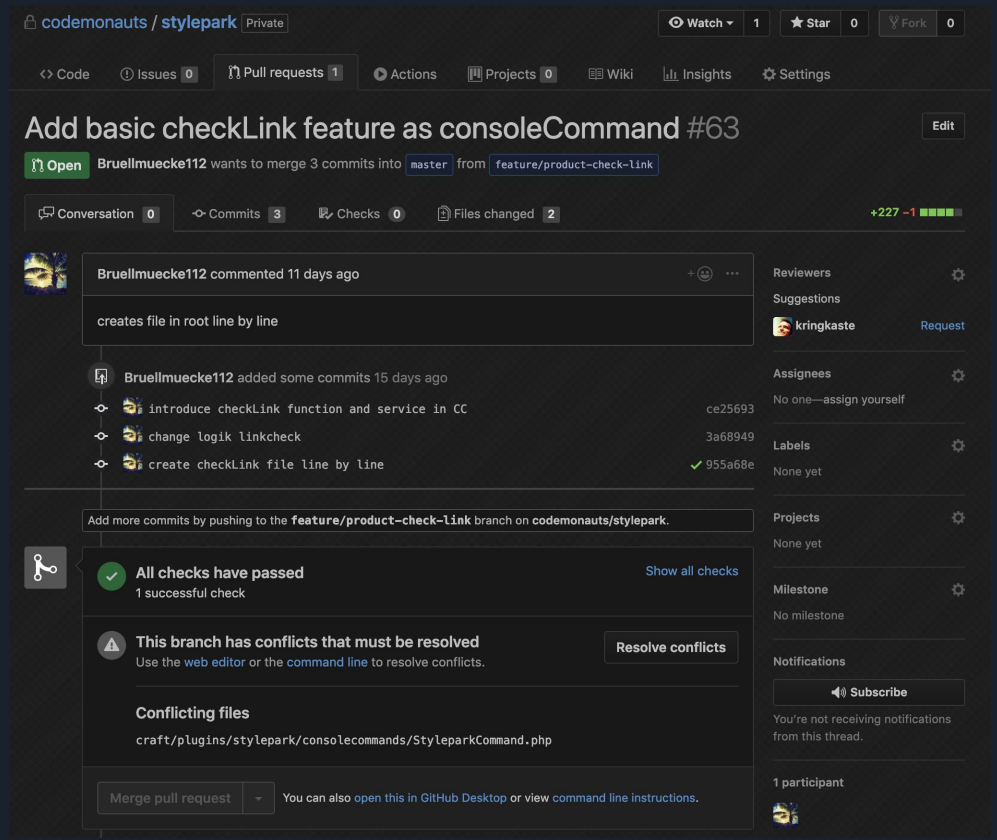
- master
  - Heilige Kuh (Daher protected)
  - Was hier liegt wurde reviewed und getested
  - Wird auf production deployed
- development
  - Wird auf staging deployed
  - **Ziel**: master
- feature branches
  - Prefixen mit “feature/”
  - **Start**: development
  - **Ziel**: development
- hotfix branches
  - Prefixen mit “hotfix/”
  - **Start**: master
  - **Ziel**: master + development



Workflow - PullRequests

- Vorteile
  - 4-Augen-Prinzip
  - Codestyle
  - Tests
- Nachteil
  - Mehr arbeit (Dank GitHub aber nur 2 Mausklicks)
- Integriertes Tool für CodeReviews
- Je nach Zielbranch unterschiedliche Regeln für einen PR
  - Min. X Reviewer
  - Tests laufen durch
  - Branch muss aktuell sein

Workflow - PullRequests



Workflow - PullRequests

- Wer den PR macht, hat “Bringschuld”
  - Mergekonflikte
  - Styleguide
  - Tests
- Featurebranches regelmäßig auf den aktuellen Stand bringen
  - Schützt vor dem großen “Oh!” wenn der PR gestellt wird
  - Freundet euch dafür mit “git rebase” an

- rebase um entweder
  - eigenen Branch auf den aktuellen Stand zu bringen (git fetch; git rebase origin/development)
  - eigenen Branch vor einem PR aufzuräumen (git rebase -i HEAD~5)

Rebase Regeln

Rebase bedeutet das ihr die Vergangenheit verändert

Vergangenheit verändern bedeutet dass ihr einen force push machen müsst

Forcepusht **NUR** auf euren eigenen Branches (oder wenn ihr eure Kollegen hasst…)

Ausnahme: git pull -r

CodeStyle


Tabs
Spaces


CamelCase
snake_case


if {
if
{

Codestyle

PHP Stylecheck auf codemonauts/stylepark ...

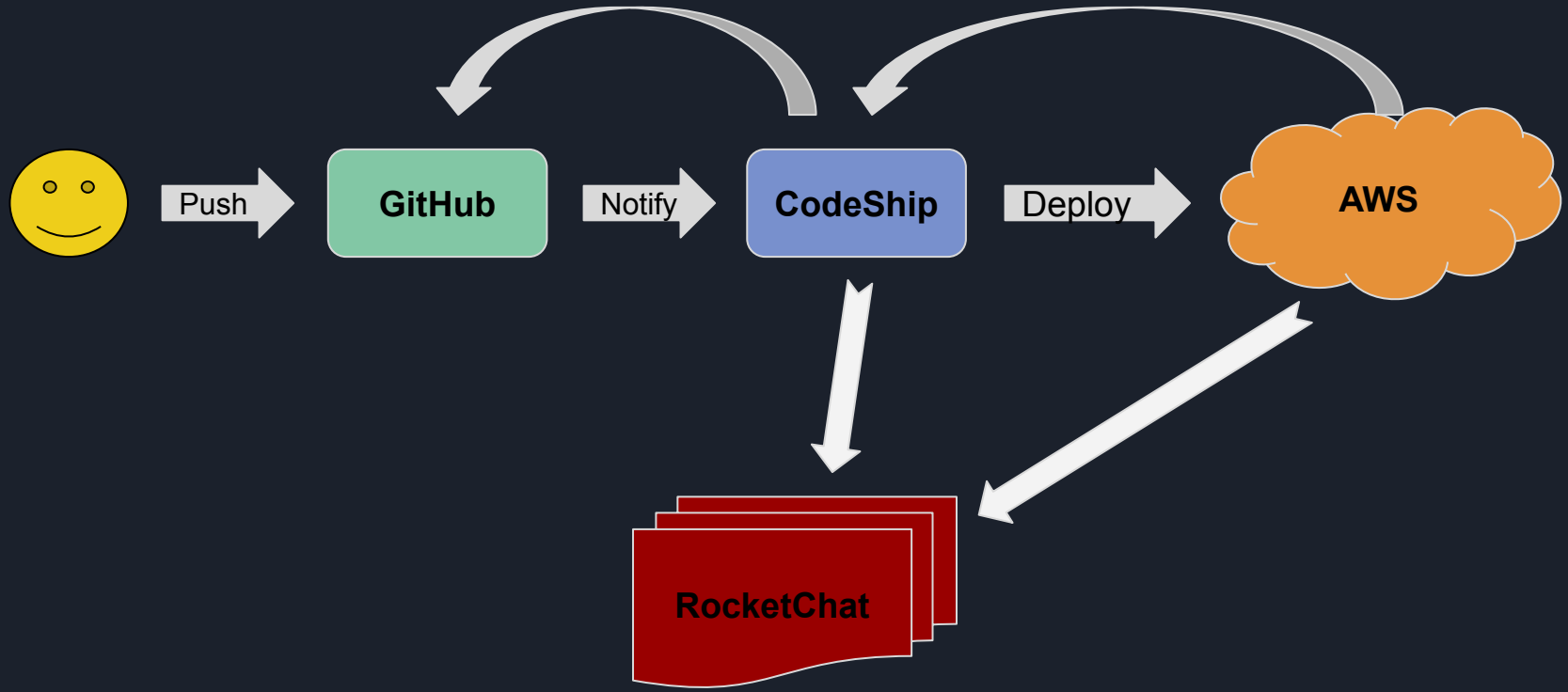
```
--------------------------------------------------------------------------------

A TOTAL OF 244585 ERRORS AND 11902 WARNINGS WERE FOUND IN 1535 FILES

--------------------------------------------------------------------------------
```

Codestyle

- Einheitliche Formatierung hilft beim Lesen
  - ⇒ Hilft bei CodeReviews
- Verringert Mergconflicts und macht Commits sauberer
- Macht neuen Mitarbeiterinnen den Einstieg einfacher
- Auf public Standards einigen oder Teamintern eigene definieren
  - PHP hat PSR1, PSR2
  - Python hat PEP8
  - JS hat eslint mit fertigen Configs von bspw Google oder Air BnB
  - Go hat offiziellen Standard über gofmt
  - SASS/CSS hat auch was (*https://css-tricks.com/css-style-guides*)
- Kann eurer IDE beigebracht werden

CI/CD Pipeline

Unsere CI/CD Pipeline

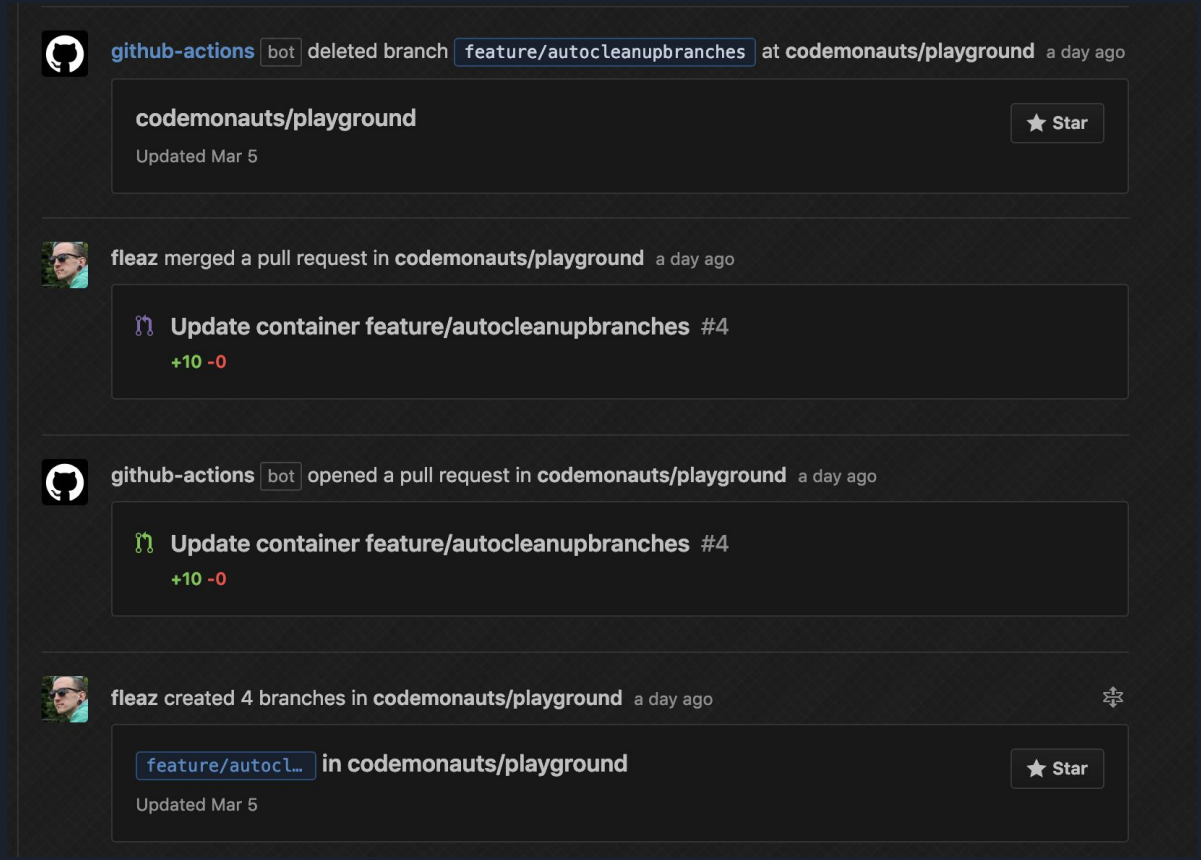


Lifehacks

GitHub Actions

- Automation für GitHub
- Kann einem repetitive Arbeit abnehmen

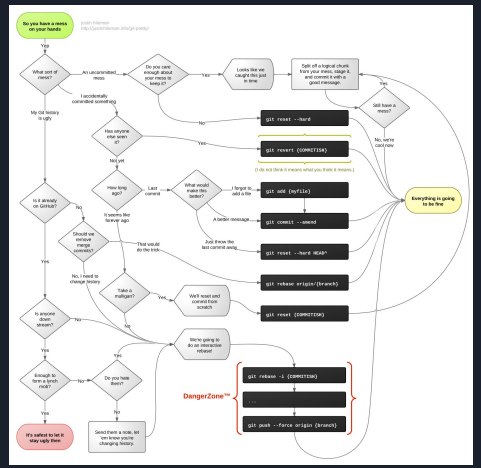
GitHub Actions



Oh shit git

- Es ist schwer ein Repo wirklich richtig kaputt zu machen (Das war keine Aufforderung!)
- Worst case: Ordner löschen und neu clonen hilft immer
- In allen anderen Fällen:

Git-Pretty



http://justinhileman.info/article/git-pretty/

git fetch

- Holt die aktuellsten Änderungen vom origin OHNE sie in euer Repo einzupflegen
- git pull = git fetch + git merge
- Hilfreich für bspw rebase
- Tower macht fetches im Hintergrund automatisch

git add -p

- p steht für “partial”
- Hilft commits besser aufzusplitten

```
diff --git a/commit.c b/commit.c
index ae5b519..a1f2b00 100644
--- a/commit.c
+++ b/commit.c
@@ -1,6 +1,6 @@
 #include <stdio.h>

-void foo(void);
+void xpto(void);
 void bar(int);

 int main(int argc, char **argv)
Stage this hunk [y,n,q,a,d,/,j,J,g,e,?]? n
```

git --amend

- Verändert den letzten commit
  - Vergessene Dateien
  - Typo in der Message
- Super hilfreich wenn noch nicht gepusht wurde

- Alternative:
  - git commit --fixup <hash>

git stash

- Nimmt alle Changes in eurem Repo und parkt sie zwischen
  - Egal ob schon gestaged
- Ist nicht an branches gebunden
- Kann wie ein Commit mit einer Message versehen werden

Usecases:

- Aus versehen auf dem falschen Branch angefangen zu entwickeln
- Entwicklung pausieren für spontanen Hotfix

git revert

- Nimmt einen commit und invertiert ihn vollständig
- Änderungen rückgängig machen und gleichzeitig
  - schöne Historie
  - wenig Arbeit

git checkout

- Verwirft Änderungen an einem File oder schaue einen anderen Commit an

- git checkout development
- git checkout -- foobar.php
- git checkout 4c6366a37d0870
  - Detached HEAD

git reset

- “löscht” Commits

- --hard
  - Arbeite auf Working dir, Index und Historie
- --mixed
  - Arbeitet auf Historie und Index
  - Changes im index werden zurück ins Workdir geschoben
- --soft
  - Arbeitet nur auf der Historie
  - Workdir und Index bleiben erhalten

- default ist ‘--mixed HEAD’

git cherry-pick

- Nimmt einen einzelnen Commit aus einem anderen Branch

Repo maintenance

- git fsck
  - Findet “dangling objects”
- git gc
  - Räumt alte Dateien und Referenzen auf
- git remote update --prune
  - Löscht branches die es im Origin nicht mehr gibt
- git stash list
  - Listet alle gestashten Änderungen auf

- Tutorials und Erklärungen mit vielen Grafiken
  - https://www.atlassian.com/git

- “Offizielles” Buch
  - https://git-scm.com/book/en/v2