

Git und Github

Hossein Sherkat

Materialien und Quellen:

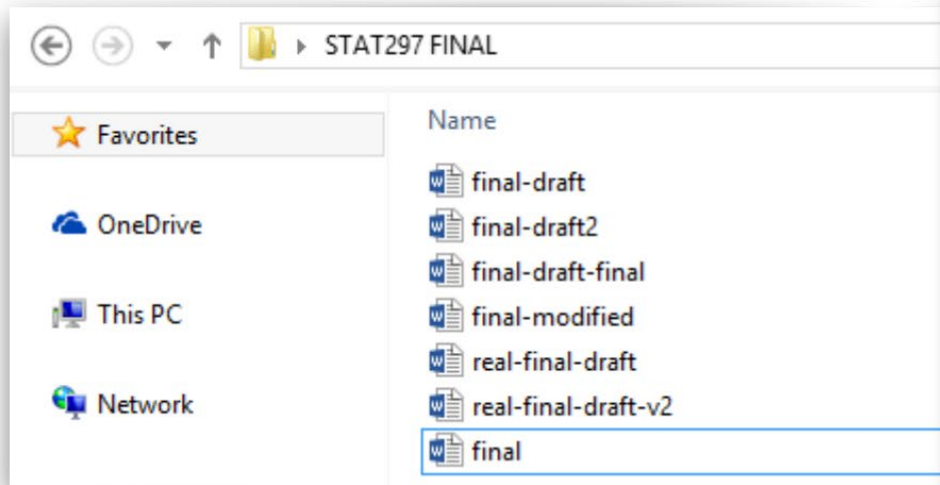
<https://docs.github.com/en/get-started/start-your-journey>

<https://rogerdudler.github.io/git-guide/>

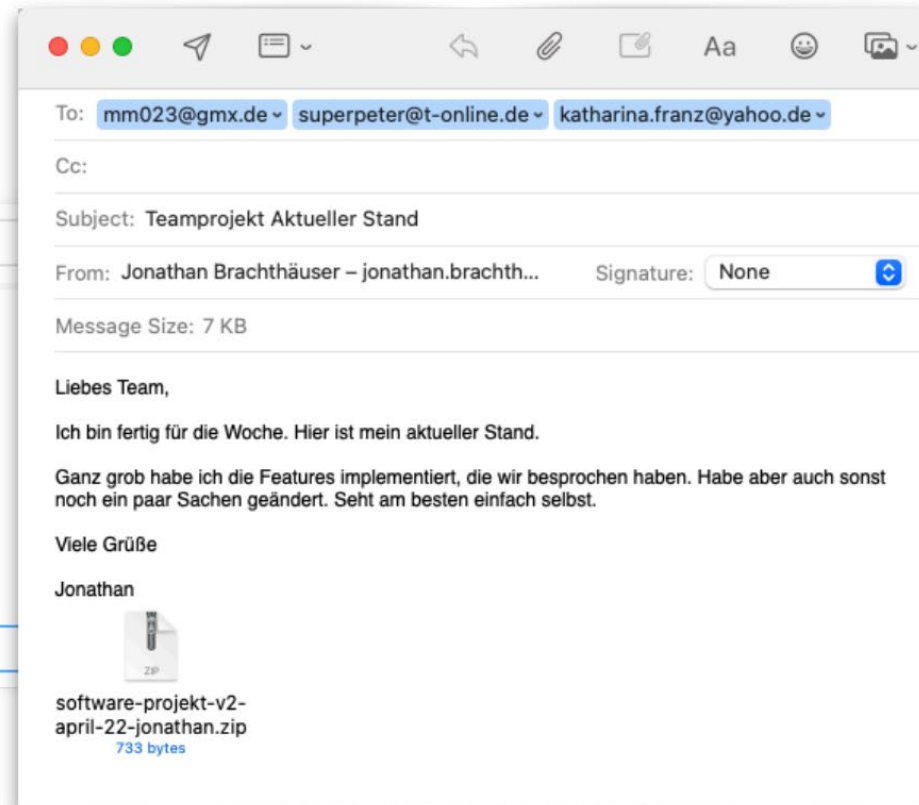
Skript von SE (SoSe23)

Git -> Versionskontrollsystem oder Versionsverwaltungssystem

Versionsverwaltung — so nicht!



Quelle: <https://smac-group.github.io/ds/github.html>



VCS

- ▶ Gemeinsames Arbeiten an dem gleichen Dokument
- ▶ Änderungen gezielt rückgängig machen
- ▶ Verschiedene Versionen parallel bearbeiten
- ▶ Änderungen diskutieren / feedback einholen
- ▶ Änderungen im Nachhinein nachvollziehen
- ▶ Konflikte in der Bearbeitung lösen
- ▶ ...

Git:

- ▶ den Inhalt aller Dateien eines Projektes
- ▶ die gesamte Änderungsgeschichte (Versionsgeschichte / history) eines Projektes
- ▶ Metadaten wie Änderungszeitpunkt, Author, Kommentare, Versionsnummer

Begriffe

Repository

Änderungsgeschichte und Dateiinhalte für alle Versionen

Arbeitskopie

Lokale Kopie aller Dateien einer Version

Status

Zusammengefasster Unterschied zwischen Arbeitskopie und Repository

Diff

Unterschiede zwischen Dateien der Arbeitskopie, des Repositories, oder anderer Versionen

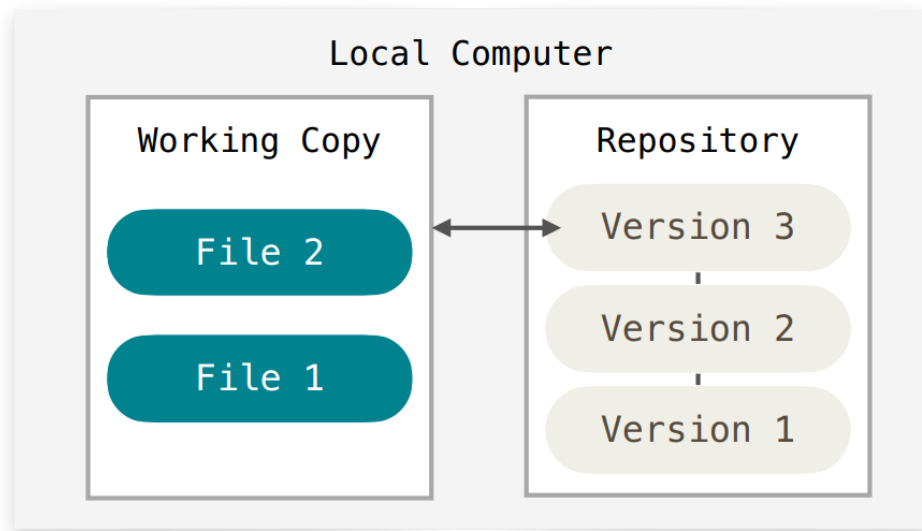
Checkout

Dateiinhalte einer Version aus dem Repository in die Arbeitskopie kopieren

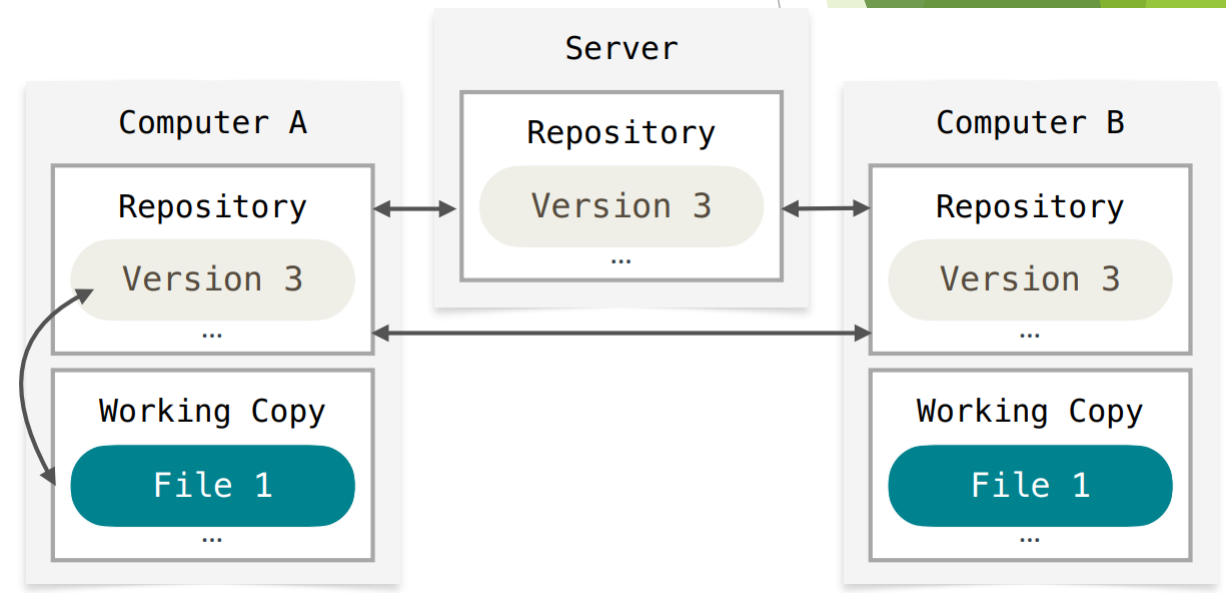
Commit

Derzeitigen Stand der Arbeitskopie (selektiv) als neue Version in das Repository übernehmen

Repository



Die Arbeitskopie (links) enthält Kopien der Dateien einer bestimmten Version, die im Repository (rechts) hinterlegt ist.



LIVE DEMO (1)

Was sollte in einen Commit? Was sollte nicht?

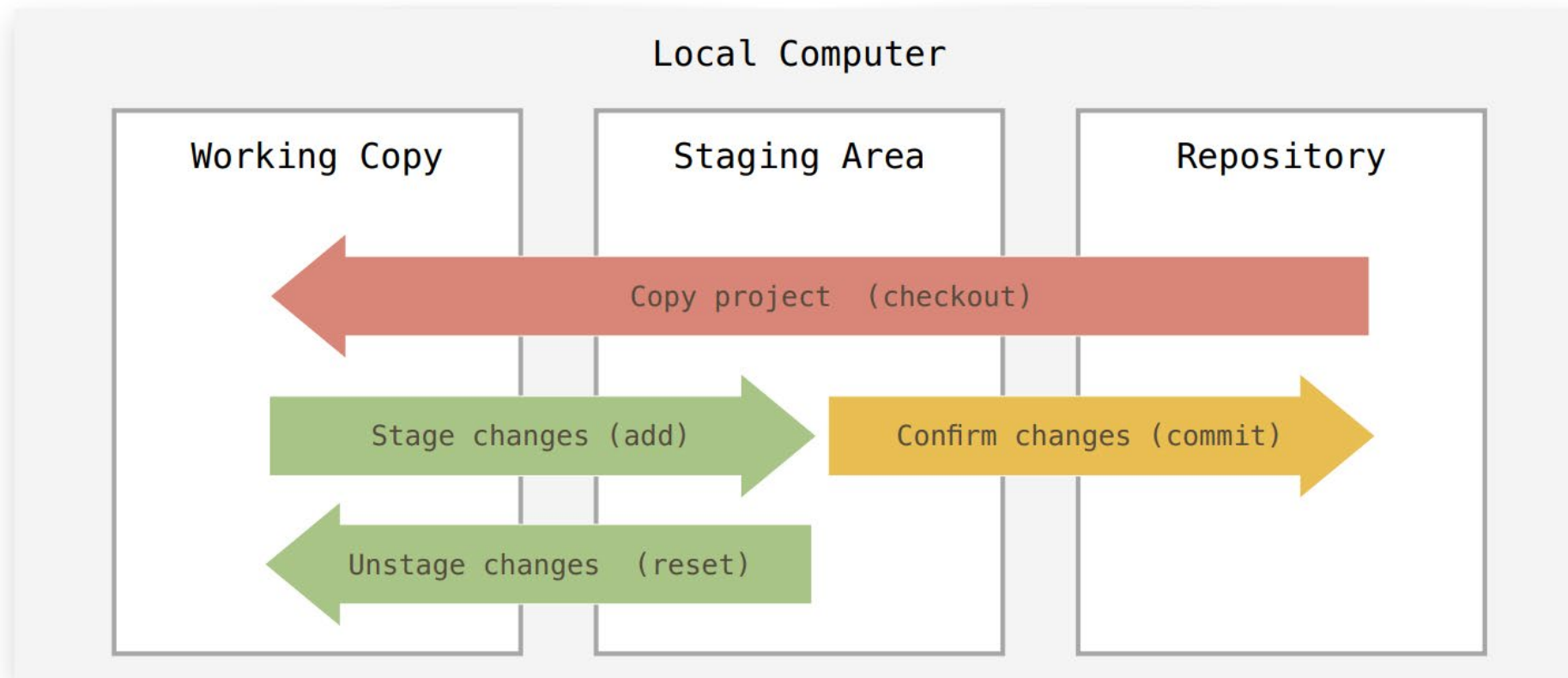
- ▶ Metadaten
- ▶ Inhalt
- ▶ Unnötige Inhalte
- ▶ Gemischte Inhalte
- ▶ Kosmetische Änderungen



<https://github.com/lampepfl/dotty/commit/20e7269f78916026fc98a7609b5c6a0bc4f8c1e3>

Wie funktioniert Git?

Working Copy, Staging Area und Repository



LIVE DEMO (2)

git init

git add

git reset

git commit

git diff (-- staged)

git status

git checkout

git log

git log --oneline --graph --decorate --all

Workflow



Commit-hash

Die wichtigsten Befehle

git help *command*

Hilfe (zu einzelnen Befehlen)

git add *file* [-p]

Datei zum Index hinzufügen / stagen

git checkout *file* [-p]

Repository / Index zu Arbeitskopie

git status *folder*

Info zur Arbeitskopie und *HEAD* anzeigen

git log [--oneline] [--decorate] [--graph] [--all]

Änderungsgeschichte anzeigen

git init

Neues Repository initialisieren

git reset *file* [-p]

Datei vom Index entfernen

git commit [-m "*message*"]

Aktuellen Index als neue Version ins Repository

git diff [--staged]

Arbeitskopie vs. Index (bzw. Index vs. Head)

git show *commit*

Details zu einzeltem Commit anzeigen

Weitere Begriffe:

► Branching

Git illustriert: [Learn Git Branching](#)

Ein “Entwicklungszweig”, der inhaltliche Änderungen thematisch gruppiert

► Merging

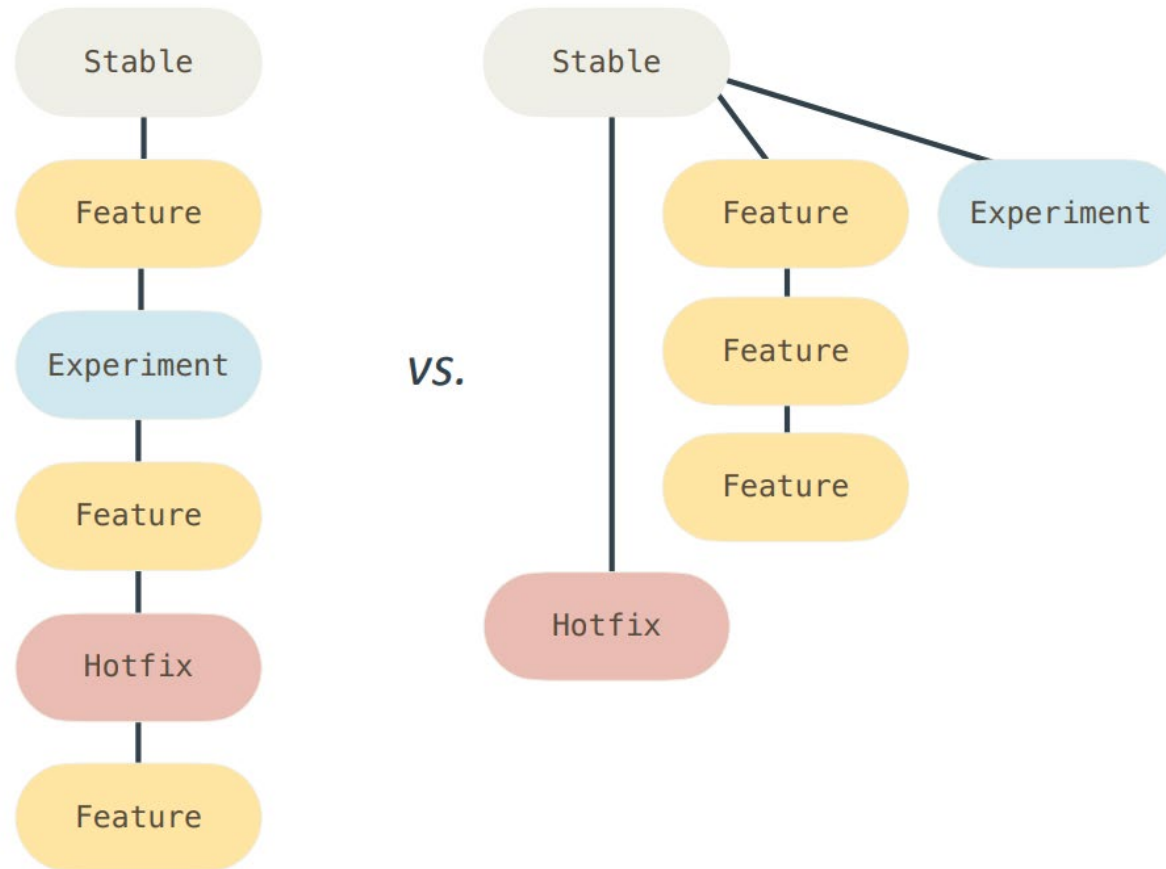
Prozess des Zusammenfügens zweier (oder mehr) Entwicklungszweige

► Rebasing

Umorganisieren von Commits innerhalb eines Branches

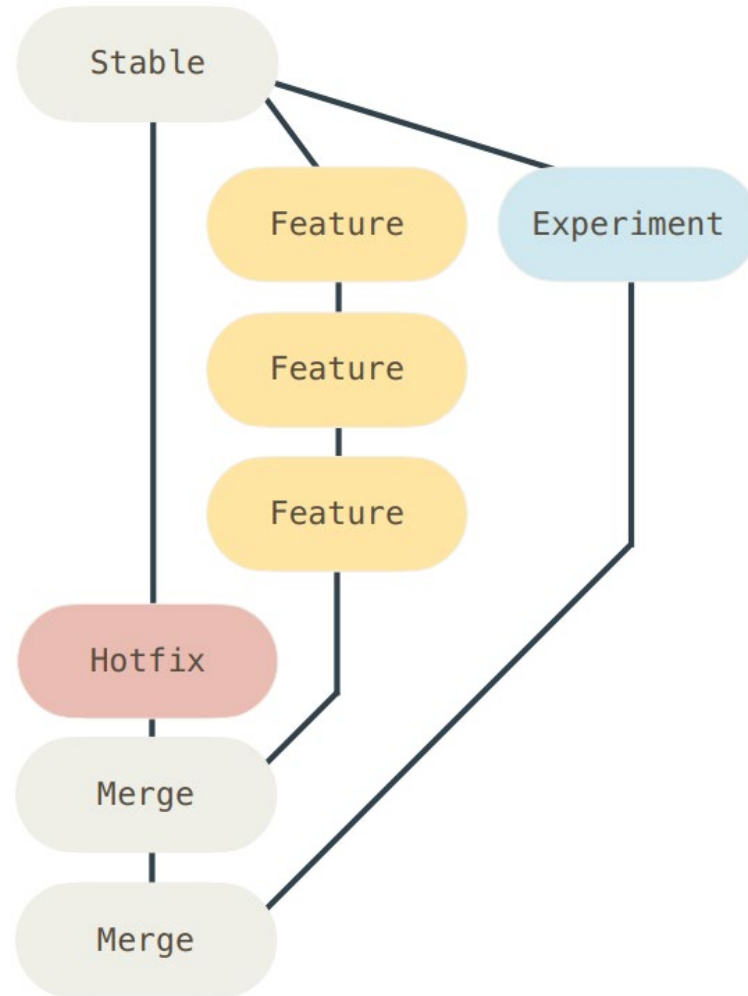
Branching

Git branch [...]
Git checkout branch_name
Git merge branch_name

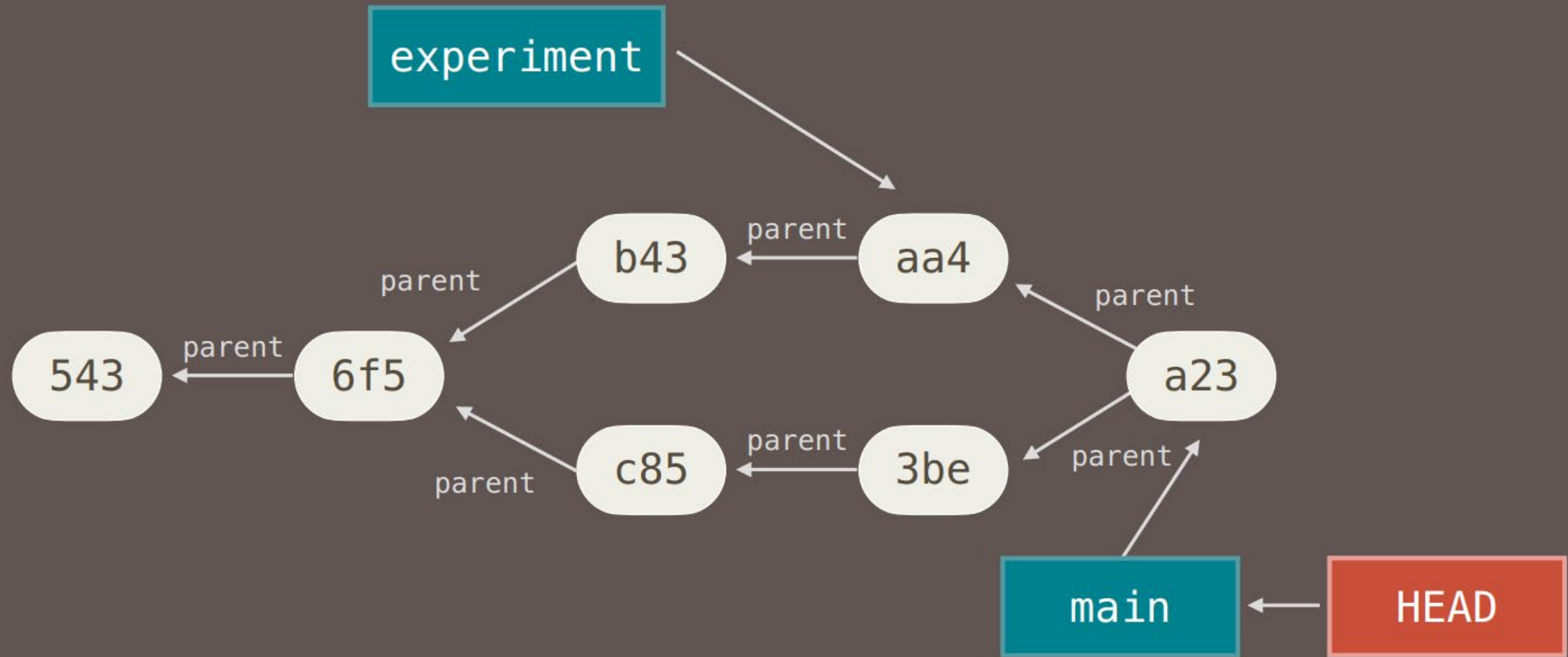


Merging

Pass auf Merge-Konflikte auf!



LIVE DEMO (3)



Rebasing

Git rebase branch

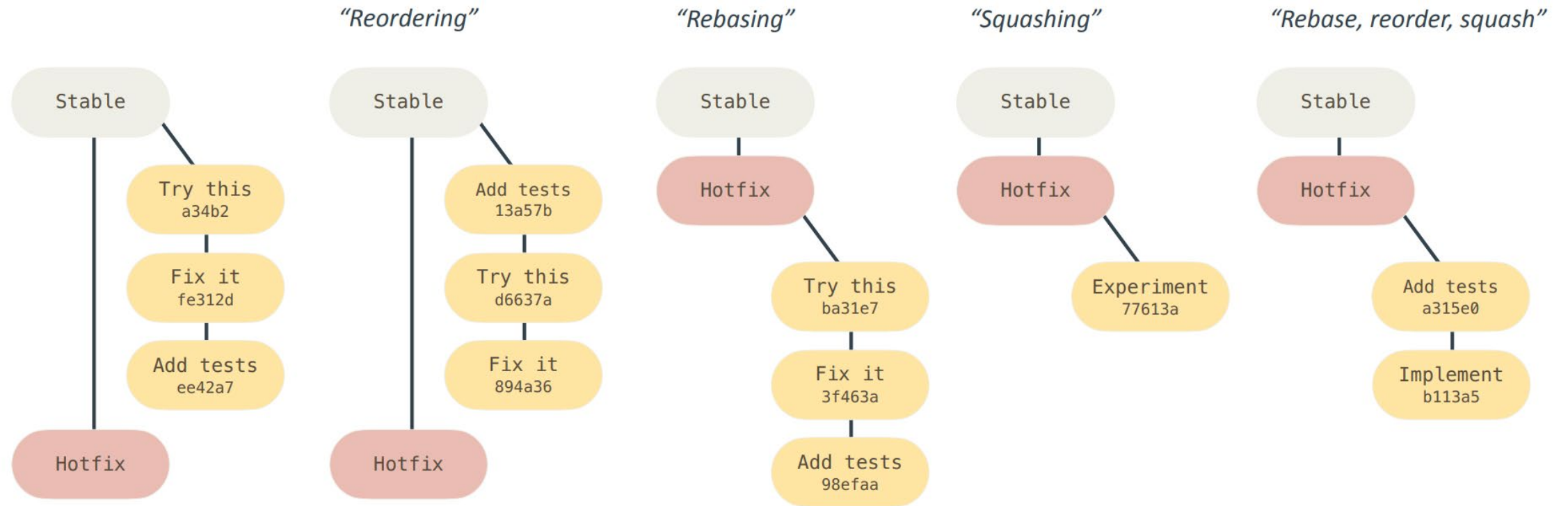
Rebasing erlaubt es Commits umzusortieren, zusammenzulegen, umzubenennen, u.v.m.

- ▶ Rebasing wendet die Änderungen nochmals an und erzeugt potentiell neue Commits (replaying)
- ▶ Wichtigstes Werkzeug, um die Versionsgeschichte aufzuräumen
- ▶ **Achtung** Rebasing ändert den Hash eines Commits!



Rebasing ist das “Schweizer Taschenmesser” unter den git Konzepten.

Anwendungsbeispiele: Rebasing



LIVE DEMO (4)

Workflows

Ein Team sollte sich (im Vorfeld!) auf einen Workflow (eine Branchingstrategie) einigen:



Feature Branch Workflow


Quelle: <https://buddy.works/blog/5-types-of-git-workflows>



Gitflow Workflow

Quelle: <https://buddy.works/blog/5-types-of-git-workflows>

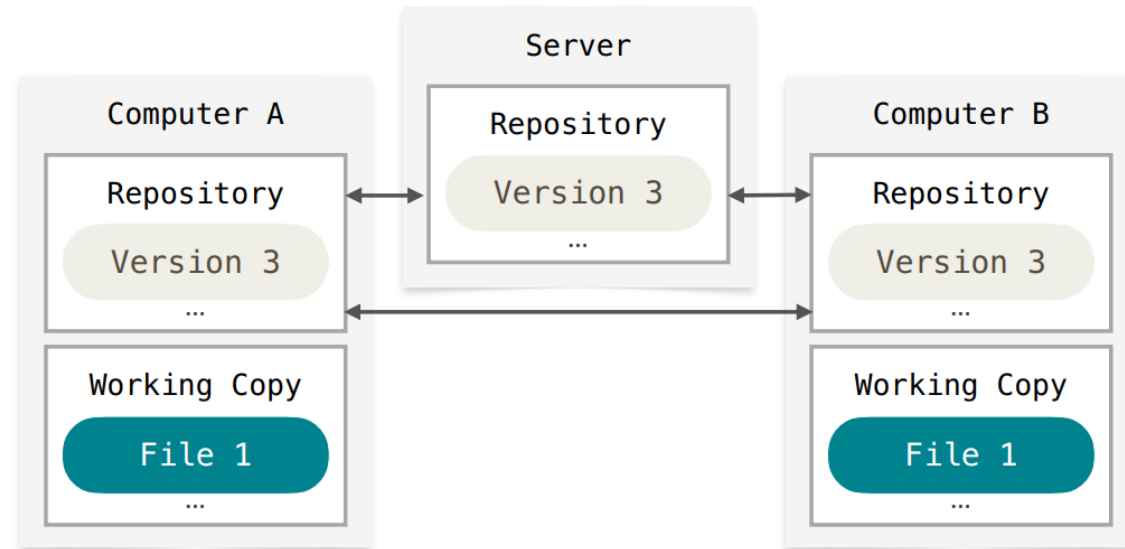
Merge Konflikte Praktisch Vermeiden

- ▶ Abgedriftete Branches 
- ▶ lieber früh und oft mergen
- ▶ Kommunizieren innerhalb des Teams

Remote Repositories (auf einem Server wie Github)

Zusammenarbeit mit git

- Das Repository ist **lokal** (`.git` Verzeichnis)
- Wie können mehrere Personen zusammenarbeiten?
- Indem die **Repositories synchronisiert** werden.



Begriffe

Remote Repository

Geteilte Kopie eines Repositories im Netzwerk / Internet

Fetch

Informationen über das Remote Repository abrufen; Tracking Branches aktualisieren

Pull

Abkürzung für Fetching und Merging

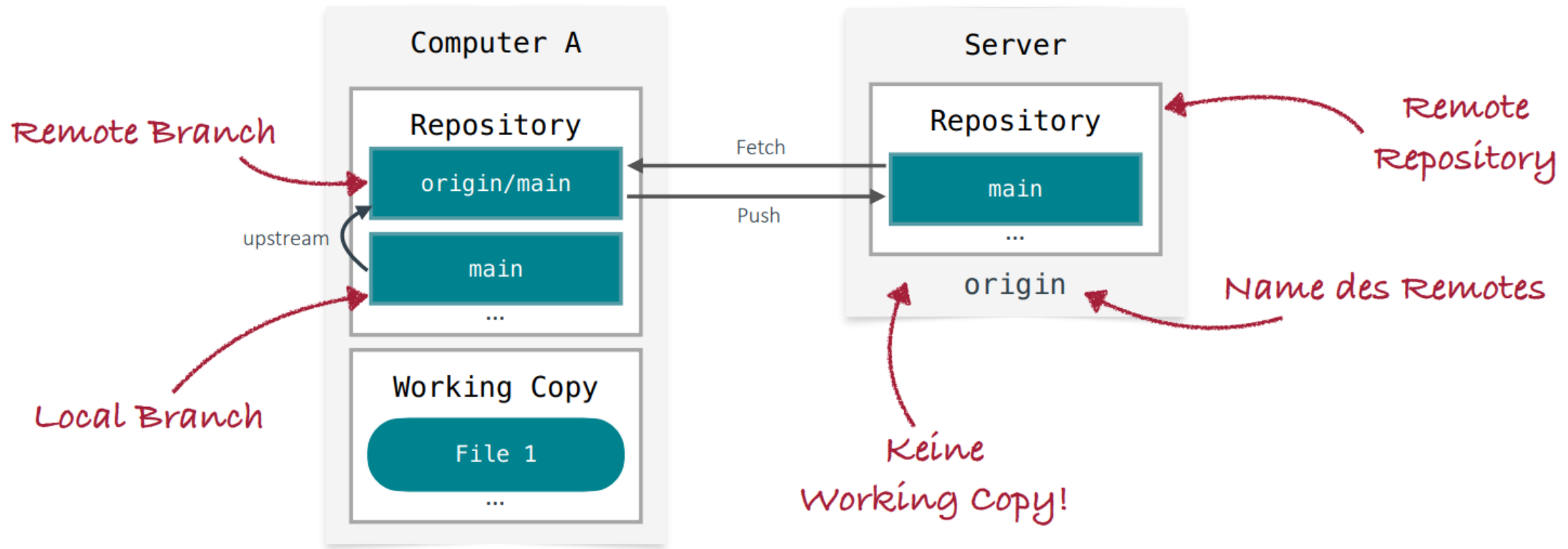
Remote-Tracking Branch

Automatisch verwalteter Branch; enthält den zuletzt bekannten Stand des Remote Repositories

Push

Informationen von einem lokalen Branch an ein Remote Repository senden

Terminologie



LIVE DEMO (5)

- ▶ Git clone ...

LIVE DEMO (6)

- ▶ Git pull and git fetch
- ▶ git push origin <branch-name>

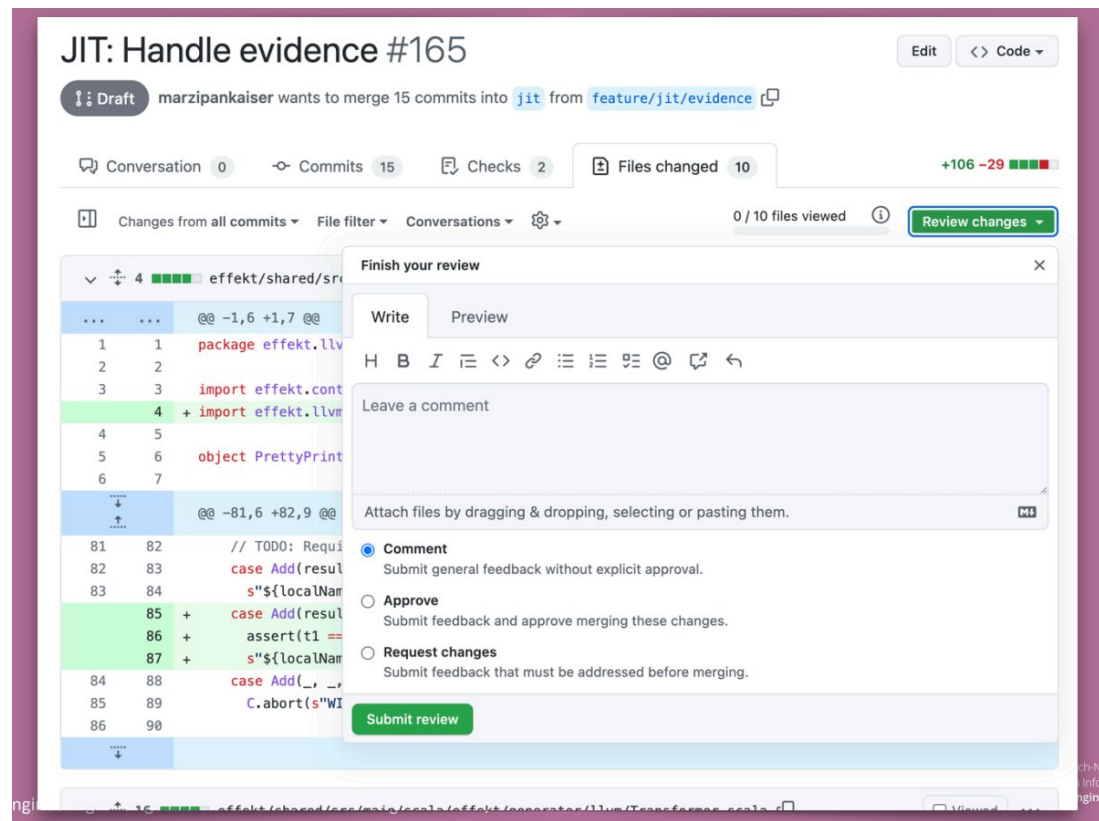
Social coding und Github

The screenshot shows the GitHub repository page for `effekt-lang/effekt`. The repository is public and has 151 stars, 5 forks, and 7 watchers. The main content area displays a list of files and folders, including `.github/workflows`, `bin`, `effekt`, `examples`, `kiama @ 7a3fdfe`, `libraries`, `project`, `.gitignore`, and `.gitmodules`. The right sidebar contains the repository description: "A research language with effect handlers and lightweight effect polymorphism", the website `effekt-lang.org`, and various tags like `language`, `algebraic-effects`, `effects`, `handlers`, `control-flow`, `language-design`, `research-project`, and `delimited-continuation`. The bottom of the sidebar shows the repository's statistics: 151 stars, 7 watching, and 5 forks.

The screenshot shows the GitHub Issues page for `effekt-lang/effekt`. The page displays 33 open issues and 28 closed issues. The issues are listed in a table with columns for Author, Label, Projects, Milestones, Assignee, and Sort. The issues are filtered by the search query `is:issue is:open`. The issues are sorted by the number of comments, with the most commented issue at the top. The issues are:

- `Resuming an object operation produces confusing error message` (bug:errorMessage, bug) - #161 opened 13 days ago by jiribenes
- `Check return type annotation of object operations` (enhancement) - #157 opened 14 days ago by jiribenes
- `Effectful signatures are not implemented for objects` (bug) - #152 opened 15 days ago by b-studios
- `Special characters in file names break js backend` (bug) - #149 opened 18 days ago by marzipankaiser
- `Explicitly handling the same effect twice should be fine` (bug:errorMessage, bug) - #146 opened 19 days ago by b-studios
- `Explicitly binding capabilities triggers spurious warning` (bug:errorMessage, bug) - #148 opened 19 days ago by b-studios

Pull requests



Ein paar Aufgaben ...

- ▶ What is git and is git = github?
- ▶ When we want to update our local repository to reflect changes made in the remote repository, which command would we use?
- ▶ What command would we use to change the base of the current branch?
- ▶ Which files are in the working area? Which ones in the staging area?

`git init`

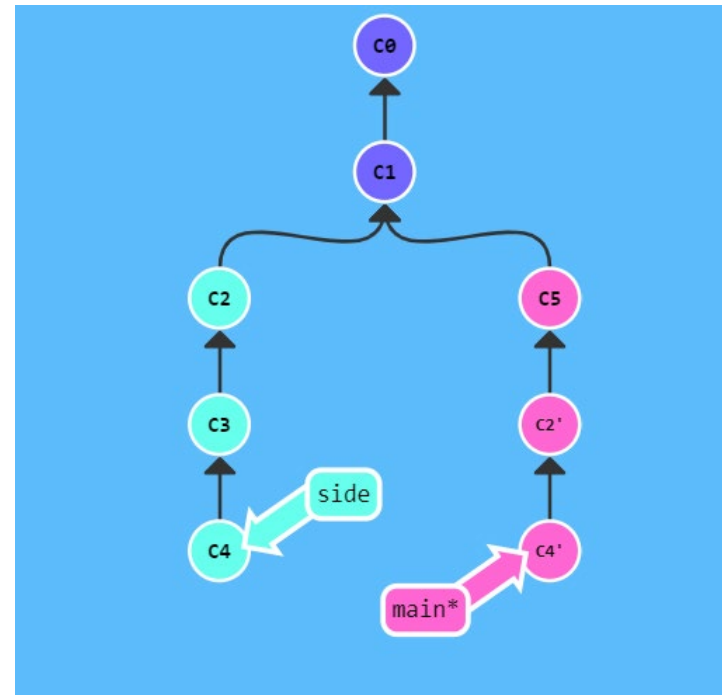
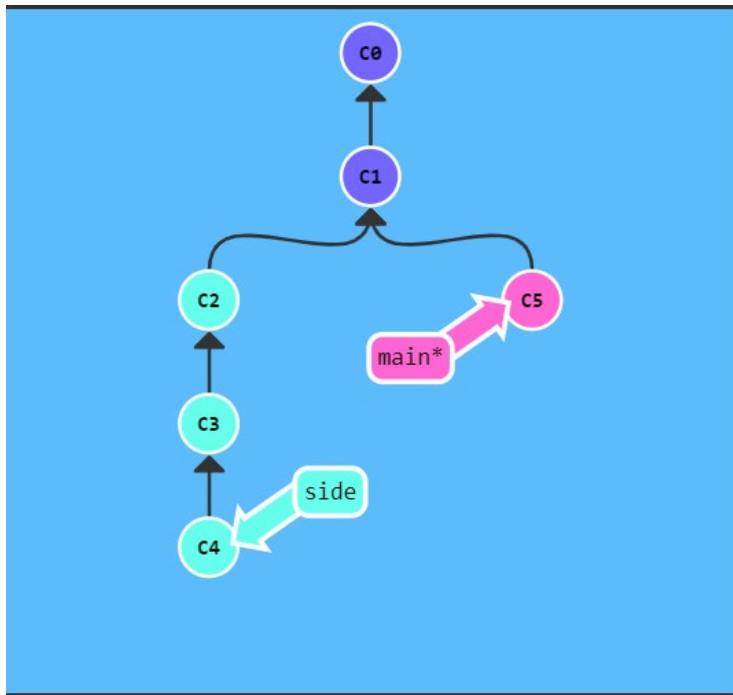
`touch foo.txt`

`touch bar.txt`

`git add foo.txt`

`git commit -m "This is a really bad commit message"`

Machen Sie sich mit dem Befehl cherry-pick vertraut und verwenden Sie ihn, um das folgende Bild zu erstellen (stellen Sie sich das erste Bild mit normalen Git-Befehlen vor und verwenden Sie dann cherry-pick im letzten Schritt, um das gewünschte Ergebnis zu erzielen)



erstellen sie mit Hilfe des Befehls merge
das folgende Bild.((stellen Sie sich das
erste Bild mit normalen Git-Befehlen vor
und verwenden Sie dann merge im
letzten Schritt)

