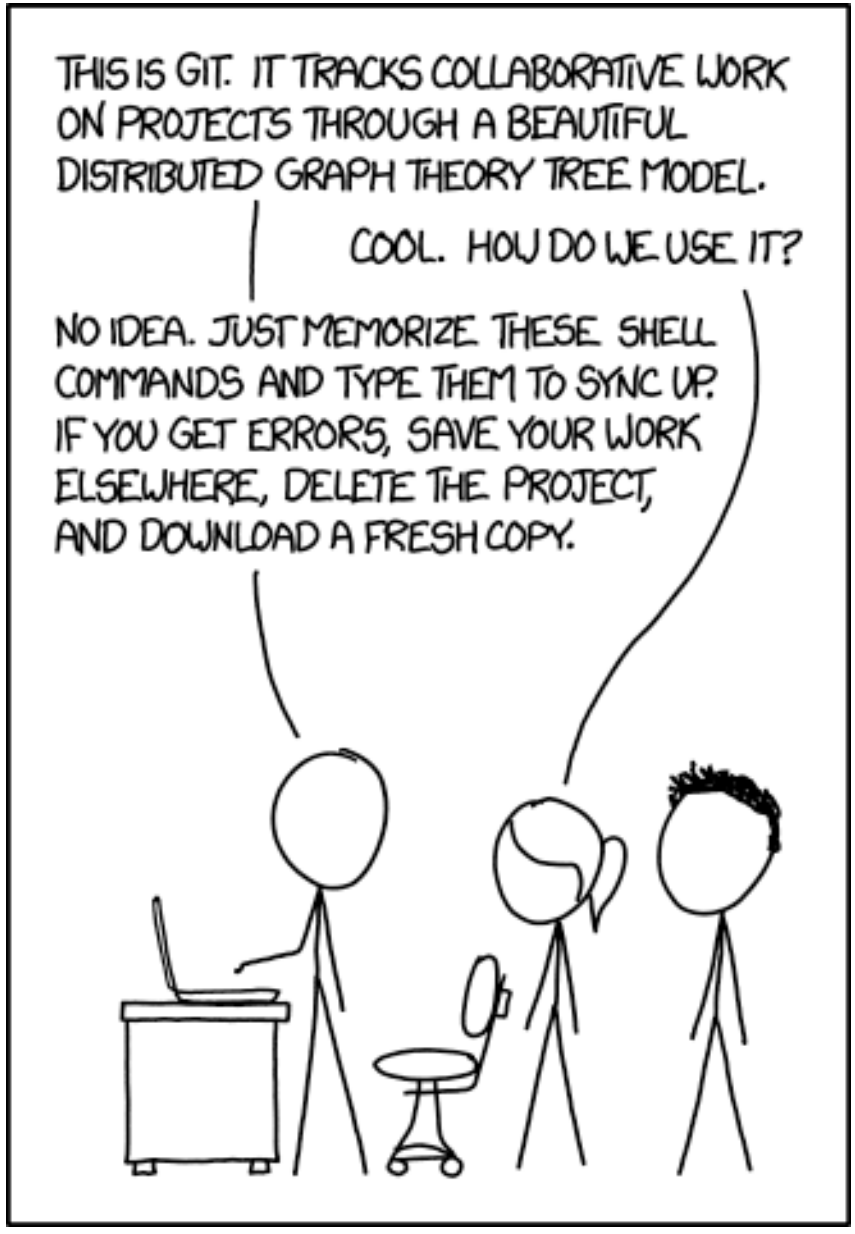
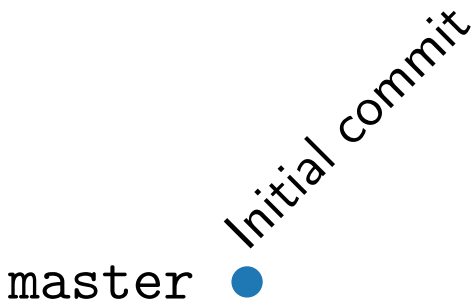
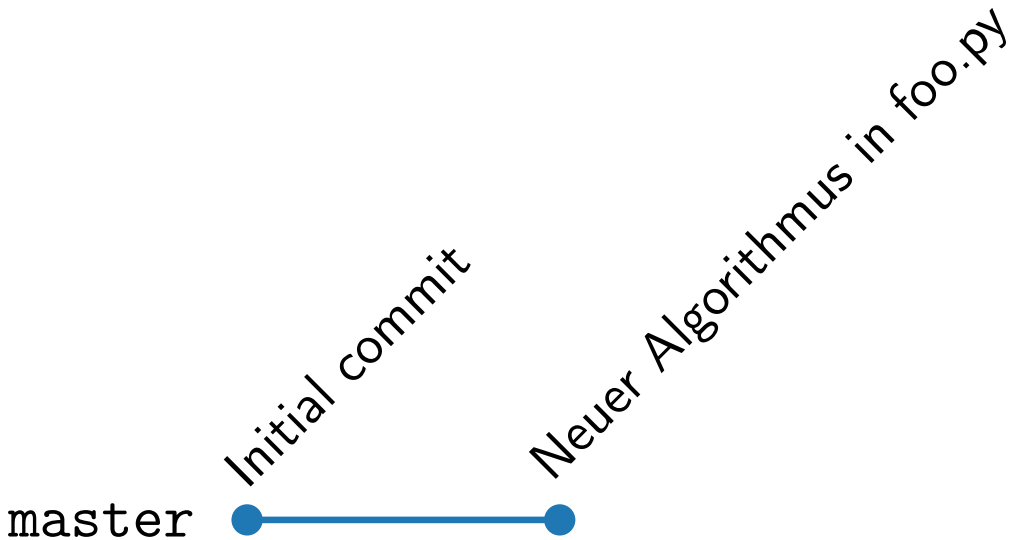

THIS IS GIT. IT TRACKS COLLABORATIVE WORK ON PROJECTS THROUGH A BEAUTIFUL DISTRIBUTED GRAPH THEORY TREE MODEL.
COOL. HOW DO WE USE IT?
NO IDEA. JUST MEMORIZE THESE SHELL COMMANDS AND TYPE THEM TO SYNC UP. IF YOU GET ERRORS, SAVE YOUR WORK ELSEWHERE, DELETE THE PROJECT, AND DOWNLOAD A FRESH COPY.

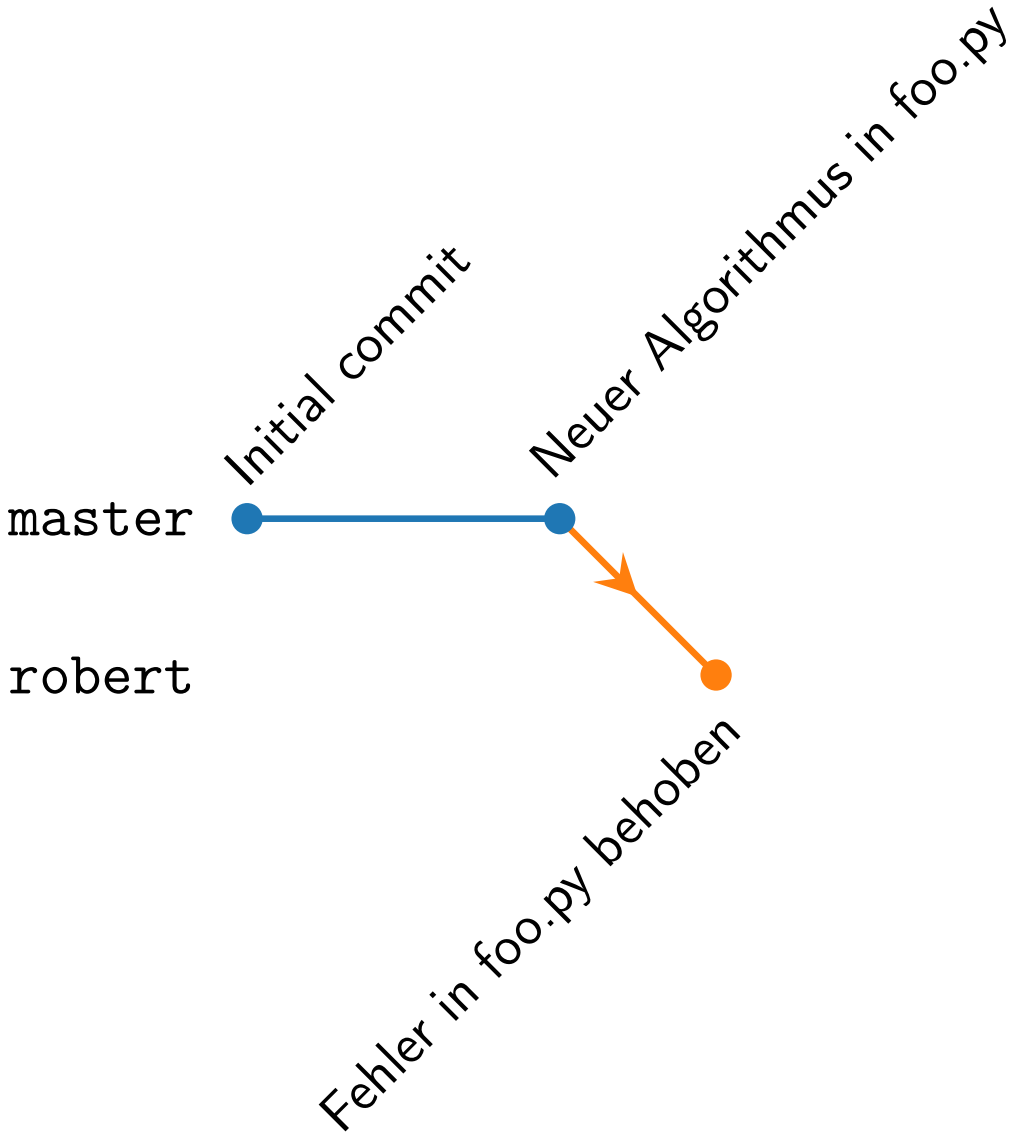
Einführung in Git



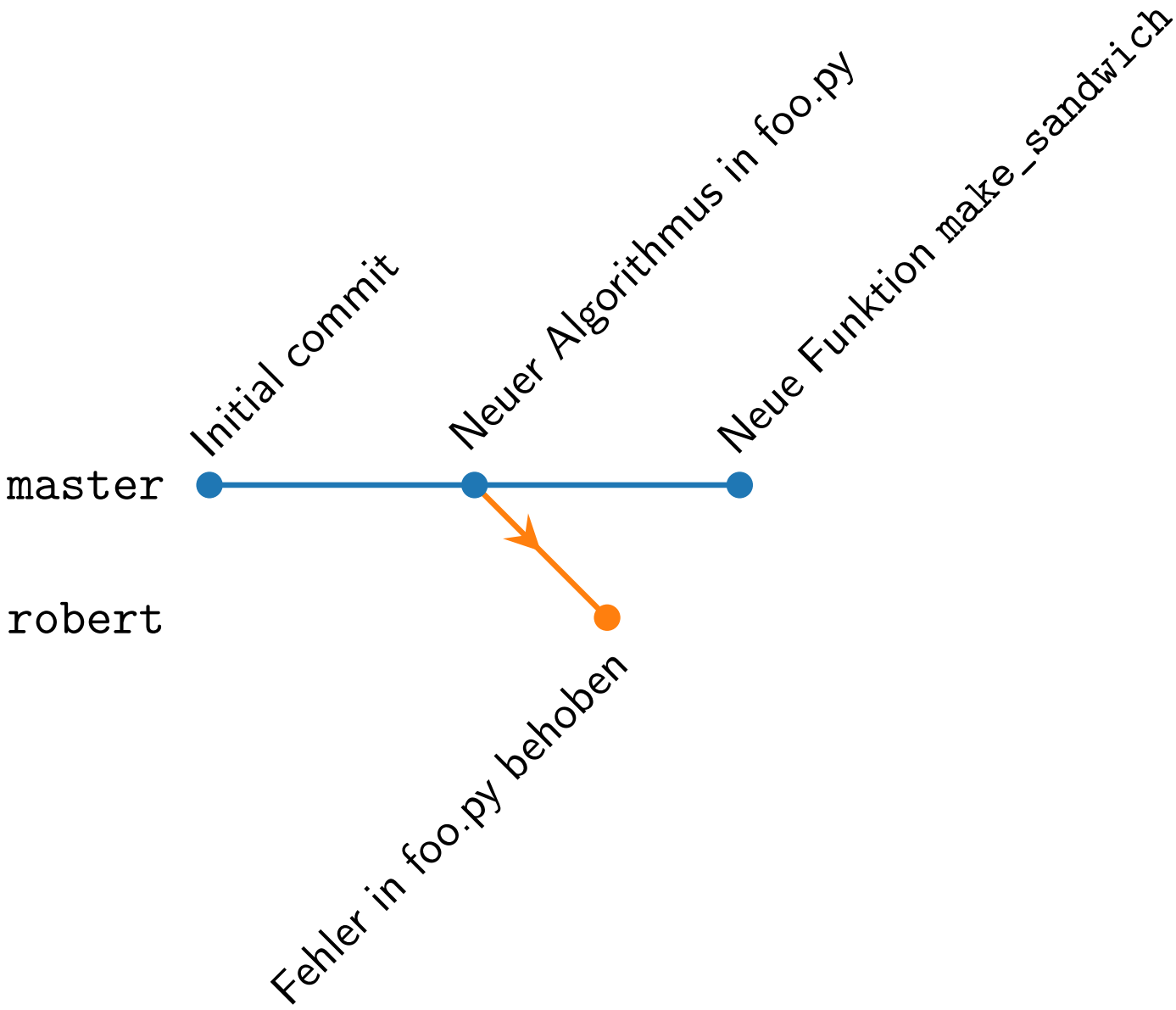
Einführung in Git



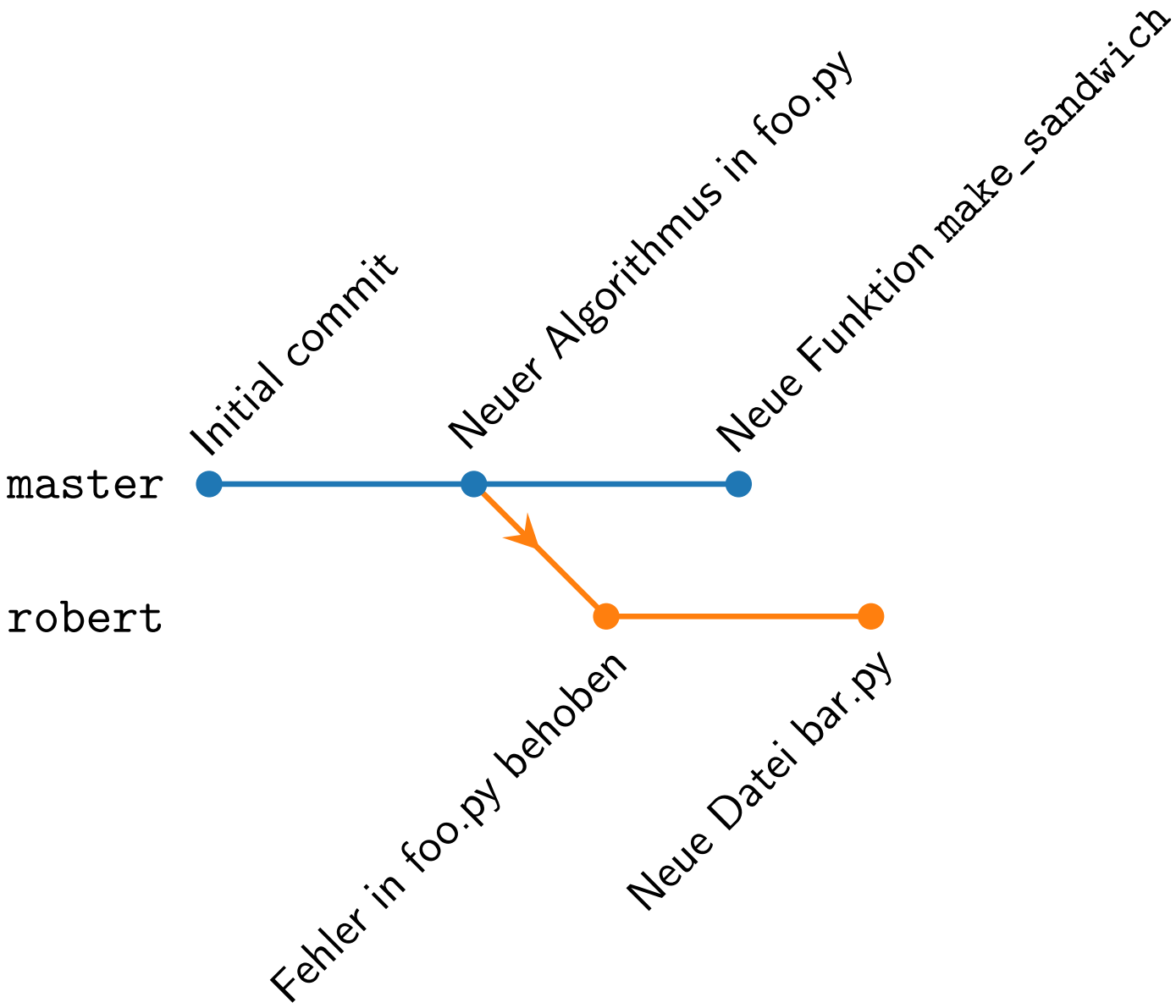
Einführung in Git



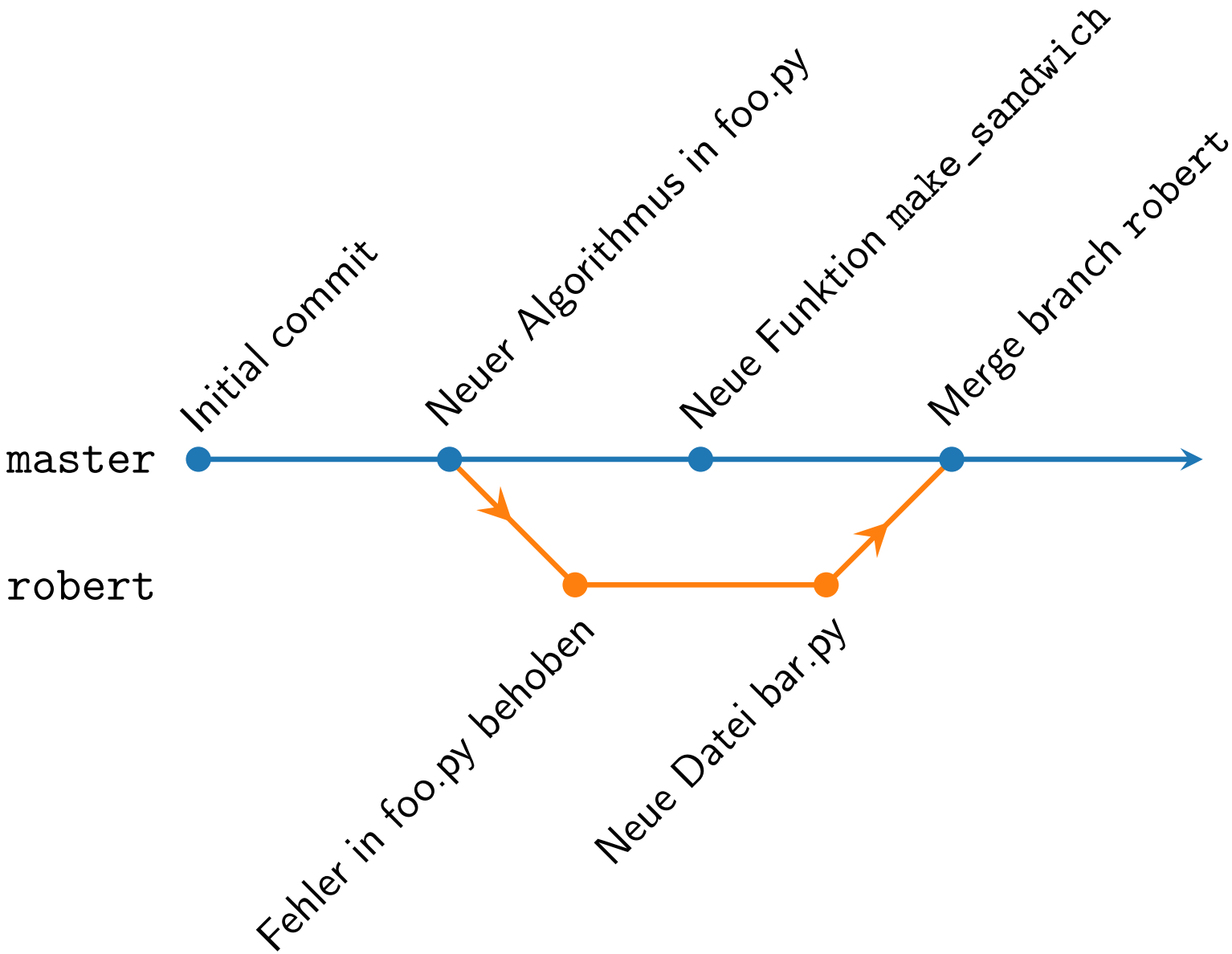
Einführung in Git



Einführung in Git



Einführung in Git



Git-Befehle

- `git config --global user.name "Max Mustermann"`
  `git config --global user.email muster@example.com`
  Legt Namen und E-Mail fest
- `git init`
  Erzeugt ein neues Git-Repository im aktuellen Ordner
- `git status`
  Zeigt den aktuellen Status an (auf welchem Branch bin ich, welche Dateien wurden erstellt, geändert, gelöscht, etc.)
- `git log`
  Zeigt die Liste der letzten Commits an
- `git add <file>`
  Fügt die Änderungen in der Datei (oder den Dateien) `<file>` dem nächsten Commit hinzu
- `git commit`
  Speichert einen neuen Commit.

Branches nutzen

- `git branch <name>`
  Erstellt einen neuen Branch mit dem Namen `<name>`
  (basierend auf dem aktuellen Branch)
- `git checkout <branch>`
  Wechselt auf den Branch `<branch>`
- `git merge <branch>`
  Führt die Änderungen aus `<branch>` mit dem aktuellen Branch zusammen $\rightarrow$ Konflikt möglich

Hochladen zu GitLab

SSH-Key anlegen: siehe Dokumentation unter
`https://gitlab.physik.uni-kiel.de/help/ssh/README`

- `git remote add <name> <url>`
  Fügt das Repository unter der Adresse `<url>` als Ziel mit dem Namen `<name>` hinzu
- `git push --set-upstream <remote> <branch>`
  Lädt den aktuellen Branch zum Ziel `<remote>` unter dem Branch-Namen `<branch>` hoch und setzt dies als Standardziel für diesen Branch $\rightarrow$ in Zukunft reicht `git push` ohne Optionen
- `git pull`
  Lädt Änderungen für den aktuellen Branch vom Server herunter $\rightarrow$ Konflikt möglich
- `git clone <url>`
  Lädt das Repository unter `<url>` vom Server in einen neuen Ordner herunter und setzt den Server als Standardziel

Weitere Kommandos

- `git commit -a`
  Shortcut für `git add` für geänderte (nicht neue) Dateien und dann `git commit`
- `git commit --amend`
  Ändert den letzten Commit
- `git rebase <viele Optionen>`
  Bearbeitet bisherige Commits und/oder fügt zusätzliche von einem anderen Branch dazwischen ein
- `git checkout -- <file>`
  Setzt die Datei `<file>` auf den Stand des letzten Commits zurück
- `git diff <commit-Hash oder Branch>`
  Zeigt die Änderungen zwischen dem aktuellen Zustand und dem genannten Commit bzw. Branch an

Noch mehr hilfreiche Dinge

Wenn irgendetwas schief geht, hilft z.B. `http://ohshitgit.com/`

In einer Datei mit dem Namen `.gitignore` können Dateien aufgelistet werden, die von Git nicht versioniert werden sollen (z.B. temporäre Dateien, Ausgabedaten von Code, ...)

Beispiel – Seite 1

```
$ mkdir repo              # Ordner "repo" anlegen (Name egal)
$ cd repo                 # in Ordner wechseln
$ git init                # Git-Repository anlegen
Initialisierte leeres Git-Repository in ~/repo/.git/

# Dateien im Ordner "repo" erstellen... Hier helloworld.py

$ git status              # Status anzeigen
Auf Branch master

Initialer Commit

Unversionierte Dateien:
(benutzen Sie "git add <Datei>...", um die Änderungen zum Commit
 ↪  vorzumerken)

helloworld.py

nichts zum Commit vorgemerkt, aber es gibt unversionierte Dateien
 ↪  (benutzen Sie "git add" zum Versionieren)
```

Beispiel – Seite 2

```
$ git add helloworld.py    # helloworld.py zu Git hinzufügen
$ git commit               # Commit anlegen
reading ~/repo/.git/COMMIT_EDITMSG

wrote ~/repo/.git/COMMIT_EDITMSG, 11 lines, 302 chars
[master (Basis-Commit) 8710a87] initial commit
1 file changed, 1 insertion(+)
create mode 100644 helloworld.py

# Datei helloworld.py bearbeiten

$ git commit -a            # geänderte Dateien committen
reading /home/asterix/forstner/git/repo/.git/COMMIT_EDITMSG

wrote /home/asterix/forstner/git/repo/.git/COMMIT_EDITMSG, 8
 ↪  lines, 268 chars
[master dfa2569] ET
1 file changed, 1 insertion(+)
```

Beispiel – Seite 3

```
$ git log   # Commits auflisten
commit dfa2569ac09bef7667efba117053ed2a95967500
Author: Johan von Forstner <forstner@physik.uni-kiel.de>
Date:   Fri Nov 24 10:53:33 2017 +0100

ET

commit 8710a87abef913d74992d4d4e0fd2108eb306bb9
Author: Johan von Forstner <forstner@physik.uni-kiel.de>
Date:   Fri Nov 24 10:50:51 2017 +0100

initial commit

# Zu GitLab hochladen
$ git remote add origin
 ↪   git@gitlab.physik.uni-kiel.de:forstner/git-test.git
$ git push --set-upstream origin master
```