

# Git

---

30. November 2020

Seminargruppe INF-Ba/04

# Gliederung

1. Mathe-Prüfung

2. Git

    Warum git?

    Installation

    Einführung

3. Ausblick

# Mathe-Prüfung

---

# Nikolausklausur

... die erste Mathe-Klausur naht!

- **Termin:** 11.12.2020 16:40 Uhr
- **Einschreibung:** endet am 6.12.2020
- Hausaufgaben bringen Bonuspunkte!
- zeitig (am besten *vorgestern*) mit Lernen anfangen
- lernt zusammen!
- **Hilfsmittel:** ein A4-Blatt ein- oder beidseitig handschriftlich beschrieben

**Eine 2.0 jetzt sichert euch das Modul!**

# Git

---

# Was ist git?

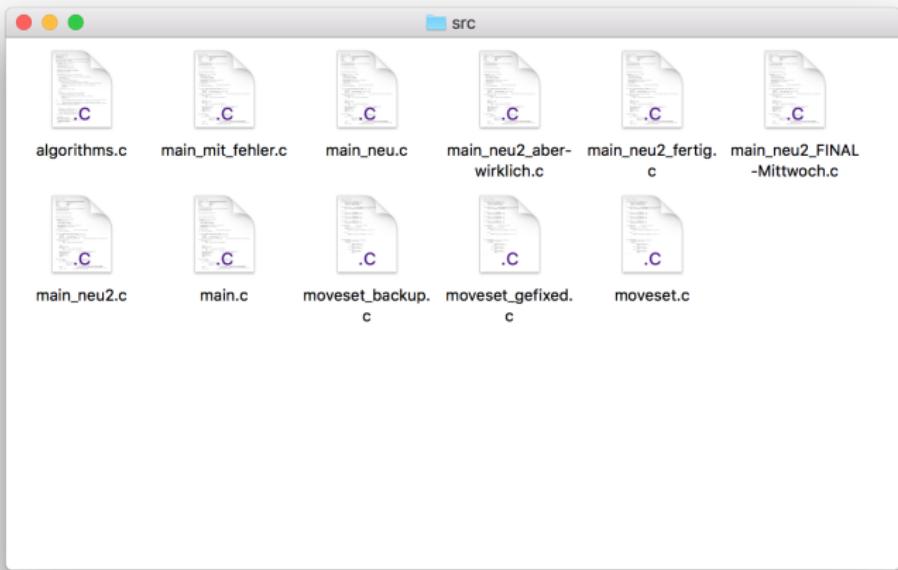
- Ein Versionsverwaltungssystem!

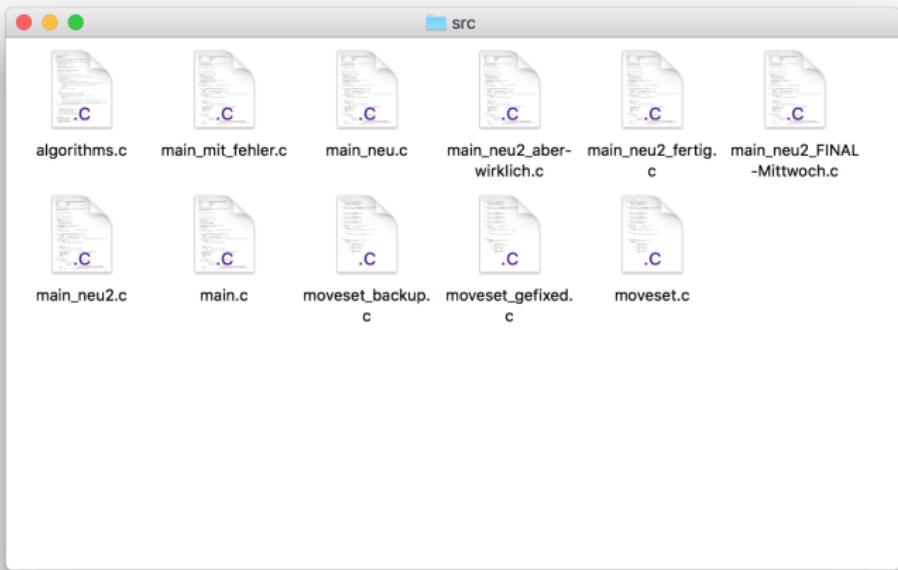
# Und wozu ist das gut?

- um den Überblick über verschiedene Versionen deines Projektes zu behalten
- zum einfacheren Zusammenarbeiten mehrerer Personen an einem gemeinsamen Projekt

# **Warum sollte ich git nutzen?**

Was spricht gegen Dropbox? Oder Facebook?





Das.

# Warum Versionierung?

- Dropbox ist (für solche Zwecke) unbenutzbar
  - man überschreibt gegenseitig Änderungen oder hat 20 Dateien à la (In Konflikt stehende Version von xx am dd.mm.yyyy)
- Verlust des Überblicks bei 20 Kopien der selben Datei mit anderem Namen

# Warum Versionierung?

- Dropbox ist (für solche Zwecke) unbenutzbar
  - man überschreibt gegenseitig Änderungen oder hat 20 Dateien à la (In Konflikt stehende Version von xx am dd.mm.yyyy)
- Verlust des Überblicks bei 20 Kopien der selben Datei mit anderem Namen

## Lösung:

- immer selbe Datei bearbeiten
- Zwischenstand irgendwo speichern, mit Info, was geändert wurde (zum rückgängig machen)

# Wie das aussehen könnte...

The screenshot shows a GitHub merge commit interface for a 'Merge branch 'software'' commit. The commit message is 'Version 1.0'. It was merged from 'master' into 'Version\_1.2 Version\_3.0' by 'Felixix42' on 25 Mar 2015. The commit has 2 parents: c91fe6e and 420e424, with a commit hash of 844b2e74b03ad8d108eb8156ca702c2302ebfc7f. The diff shows 19 changed files with 1,372 additions and 45 deletions. The 'Unified' view is selected.

**Merge branch 'software'**  
Version 1.0

master → Version\_1.2 Version\_3.0  
Felixix42 committed on 25 Mar 2015  
2 parents c91fe6e + 420e424 commit 844b2e74b03ad8d108eb8156ca702c2302ebfc7f

Showing 19 changed files with 1,372 additions and 45 deletions. Unified Split

View

5 ~~=====~~ .gitignore

```
5 ===== .gitignore
...
1  Simulation-Workspace/simulation/src/RobolabSimClient
2
3  *.elf
4  *.rxe
5  *.bin
6  =====
7  *.elf
8  *.rxe
9  *.bin
10 =====
11 *.exe
12 *.out
13 *.app
14 =====
15 #####
16 ## Eclipse
17 =====
18 *.bat
19 =====
20 #####
21 ## Eclipse
22 =====
23 *.bat
24 =====
25 #####
26 ## Eclipse
27 =====
28 *.bat
29 =====
30 #####
31 ## Eclipse
32 =====
33 *.bat
```

grün: hinzugefügte Zeilen

## weitere Vorteile

- man sieht, wer Fehler ins Programm eingeschleust hat
- zurück springen auf jeden beliebigen vorherigen Stand  
→ jede Änderung rückgängig machbar
- auch online speicherbar

# Programme zur Versionsverwaltung

Bekannte Programme:

- SVN
- Mercurial
- Git

# Programme zur Versionsverwaltung

Bekannte Programme:

- SVN
- Mercurial
- Git - **zurzeit das Beliebteste**



# Installation

# **Installation - Linux**

## **Ubuntu/Debian**

```
$ sudo apt-get install git
```

## **Arch Linux**

```
$ sudo pacman -Sy git
```

## **Fedora**

```
$ dnf -y install git
```

# Installation - macOS

**Über Homebrew (empfohlen)**

```
$ brew install git
```

**Download von der Webseite**

<https://git-scm.com>

# Installation - Windows

**Download von der Webseite**

<https://git-scm.com>

# Clients

Es gibt eine Reihe von grafischen Clients, die auf dem Kommandozeilenprogramm aufsetzen. Empfehlenswert sind:

- GitKraken
- SourceTree (viele Konfigurationsmöglichkeiten)
- GitHub Desktop
- IDE

**Aaaaber:** Lieber Kommandozeilen-Client nutzen, um die Funktionsweise zu lernen.

# **Hands on!**

Zeit für eine praktische Einführung!

# Vorbereitung

- git installiert?
- neuen (leeren) Ordner anlegen
- Git Bash (Windows) oder Terminal öffnen  
per **cd pfad/zum/ordner** in den neuen Ordner wechseln

# Navigation in einem Terminal

**noch nie eine Shell benutzt?**

# Navigation in einem Terminal

noch nie eine Shell benutzt?

- `cd mein_ordner` wechsle in `mein_ordner`
- `mkdir mein_ordner` erstelle `mein_ordner`
- `ls` liste mir alle Dateien/Ordner im aktuellen Verzeichnis
- `touch meine_datei` erstelle `meine_datei`
- `cat meine_datei` zeige den Inhalt von `meine_datei` an
- `rm meine_datei` lösche `meine_datei`

# Initiales Setup

Noch nie git benutzt?

# Initiales Setup

Noch nie git benutzt?

```
git config --global user.name "[name]"
```

Setzt Benutzernamen, der unter dem Commit stehen wird.

# Initiales Setup

Noch nie git benutzt?

```
git config --global user.name "[name]"
```

Setzt Benutzernamen, der unter dem Commit stehen wird.

```
git config --global user.email "[email]"
```

Mailadresse des Nutzers (**wichtig:** sollte GitHub-Adresse sein!)

# Initiales Setup

Noch nie git benutzt?

`git config --global user.name "[name]"`

Setzt Benutzernamen, der unter dem Commit stehen wird.

`git config --global user.email "[email]"`

Mailadresse des Nutzers (**wichtig:** sollte GitHub-Adresse sein!)

```
[felix@Samaritan] $ git config --global user.name "Max Mustermann"  
[felix@Samaritan] $ git config --global user.email "mustermann@ifsr.de"
```

Und am Anfang war...

`git init`

Erstellt ein neues Repository (*Lager*) im aktuellen Ordner.

## Und am Anfang war...

**git init**

Erstellt ein neues Repository (*Lager*) im aktuellen Ordner.

```
[felix@Samaritan] $ git init
Initialized empty Git repository in /Users/felix/test/.git/
```

## Und was ist jetzt passiert?

- .git-Ordner wurde angelegt
- in ihm geschieht „Magie“ hinter den Kulissen
- dient als Ablage für gespeicherte Änderungen an Dateien
- Einstellungen für Repository liegen hier

In diesem Ordner müsst ihr für gewöhnlich nichts ändern.

**Ein kleiner Blick unter die Haube...**

# git - Workflow

Intern organisiert git die überwachten Dateien in 3 Bereichen:

**Working Directory** Da liegen alle bearbeiteten Dateien aus eurem Projekt-Ordner drin.

**Staging Area** Wollt ihr euren Zwischenstand speichern, könnt ihr einzelne Änderungen (einzelne Dateien oder sogar Zeilen) dafür auswählen. Diese landen dann hier.

**Repository** Nachdem die ausgewählten Änderungen aus der Staging Area committed wurden, liegen sie als Gesamtpaket (Commit) hier.

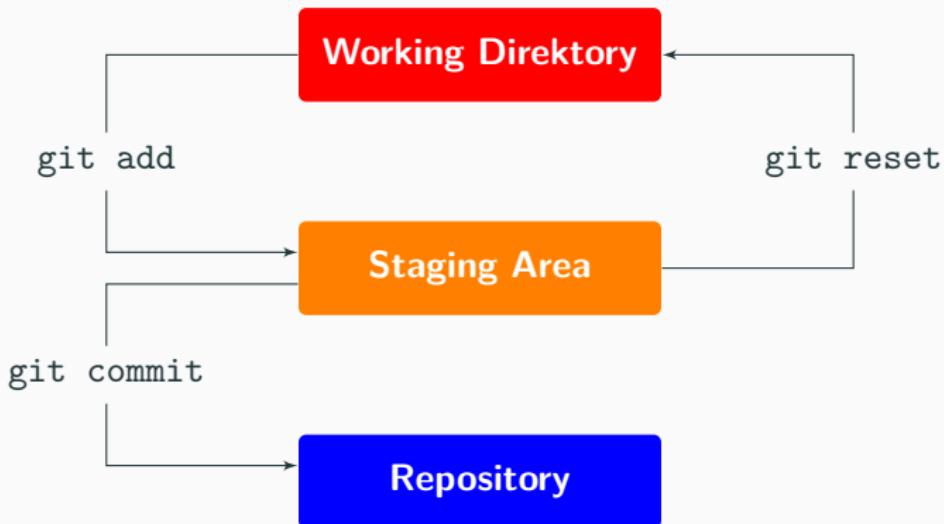
# git - Workflow

Working Direkto

# git - Workflow



# git - Workflow



**Zurück zur Praxis!**

# Her mit dem Inhalt!

*(Normalerweise kommt jetzt der Teil, an dem ihr programmiert.)*

Erstellt eine Textdatei und schreibt ein paar Zeilen.

# Änderungen speichern

Genug Inhalt zusammengekommen?  
Zeit für den *Commit*.

# Änderungen speichern

Genug Inhalt zusammengekommen?  
Zeit für den *Commit*.

**Den was...?**

# Commitment leicht gemacht

Ein **Commit** ist der Vorgang, bei dem die geänderten Dateien aus dem Working Tree gespeichert werden.

Das spielt sich nach dem immer gleichen Schema ab:

1. Dateien bearbeiten
2. zu speichernde Änderungen auswählen (**addn** oder **stagen**)
3. Änderungen **committen**

# Änderungen speichern (1)

Mit `git status` wird Status des Repositories angezeigt:

```
[felix@Samaritan] $ git status
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    hallowelt.txt

nothing added to commit but untracked files present (use "git
add" to track)
```

## Änderungen speichern (2)

Jetzt müssen alle geänderten Dateien *gestaged* werden:

**git add [Datei(en)]**

```
[felix@Samaritan] $ git add hallowelt.txt
```

## Änderungen speichern (2)

Jetzt müssen alle geänderten Dateien *gestaged* werden:

**git add [Datei(en)]**

```
[felix@Samaritan] $ git add hallowelt.txt
```

Immer überlegen, welche Dateien geaddet werden sollen. Passwörter und Access-Token haben in git nichts zu suchen!

# Wo stehen wir jetzt?

Probiert doch noch mal git status...

# Wo stehen wir jetzt?

```
[felix@Samaritan] $ git status
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:  hallowelt.txt
```

# Zeit, den Sack zu zu machen!

Bleibt nur noch der eigentliche *Commit* übrig!

**git commit -m "[Nachricht]"**

```
[felix@Samaritan] $ git commit -m "Add hallowelt.txt"

[master (root-commit) 6386d2f] Add hallowelt.txt
 1 file changed, 1 insertion(+)
 create mode 100644 hallowelt.txt
```

Glückwunsch! Euer erster Commit! :)

**Tipp:** Verwendet *aussagekräftige* Commit-Nachrichten, aus denen man sofort herauslesen kann, welche Änderungen ihr vorgenommen habt!

### Negativ-Beispiel:

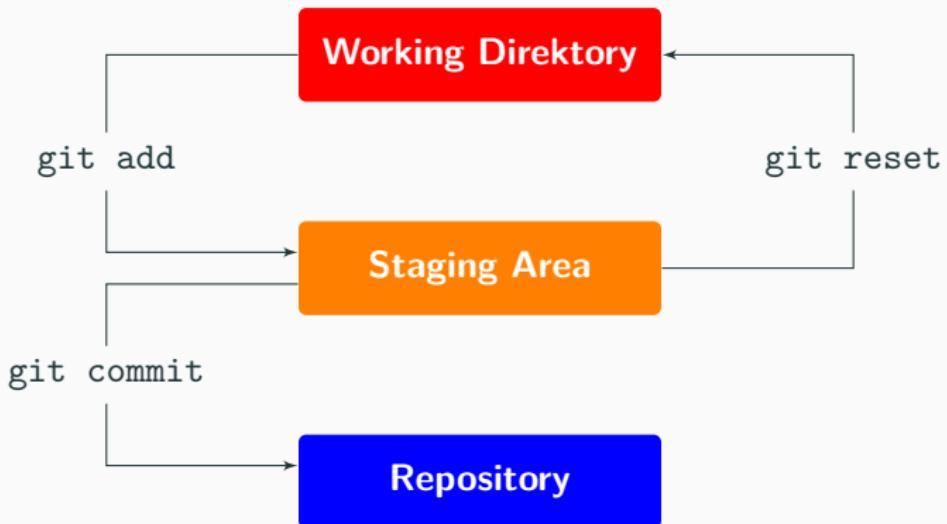
```
3d06ff9 Aariesntdyaus
9bffefc Fuuuuuuuuuuu
1829923 I are the master of code!!!111
3d06ff9 Added/removed a file
9bffefc This time it is probably fixed
1829923 Prettify the output
```

**Nett. Aber alles immer noch lokal. Wie funktioniert jetzt die Zusammenarbeit?**

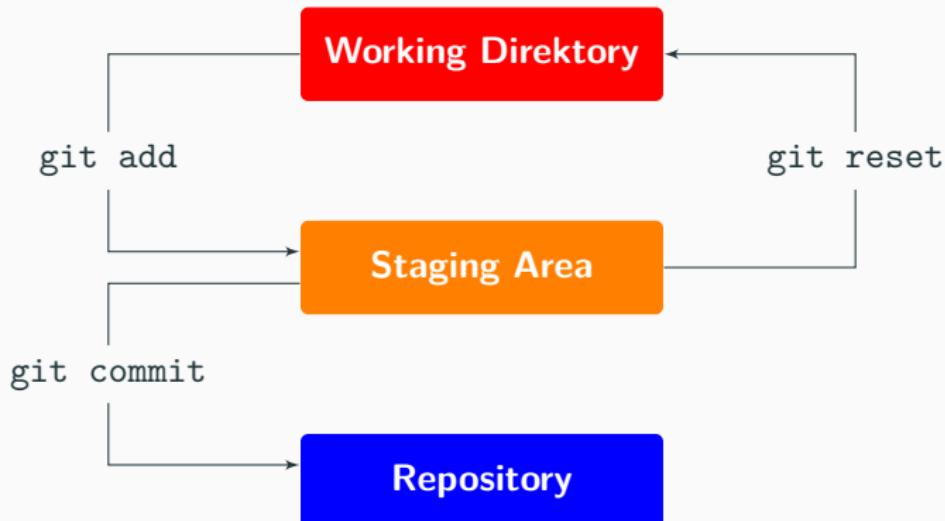
**Nett. Aber alles immer noch lokal. Wie funktioniert jetzt die Zusammenarbeit?**

Mit Remote Repositories!

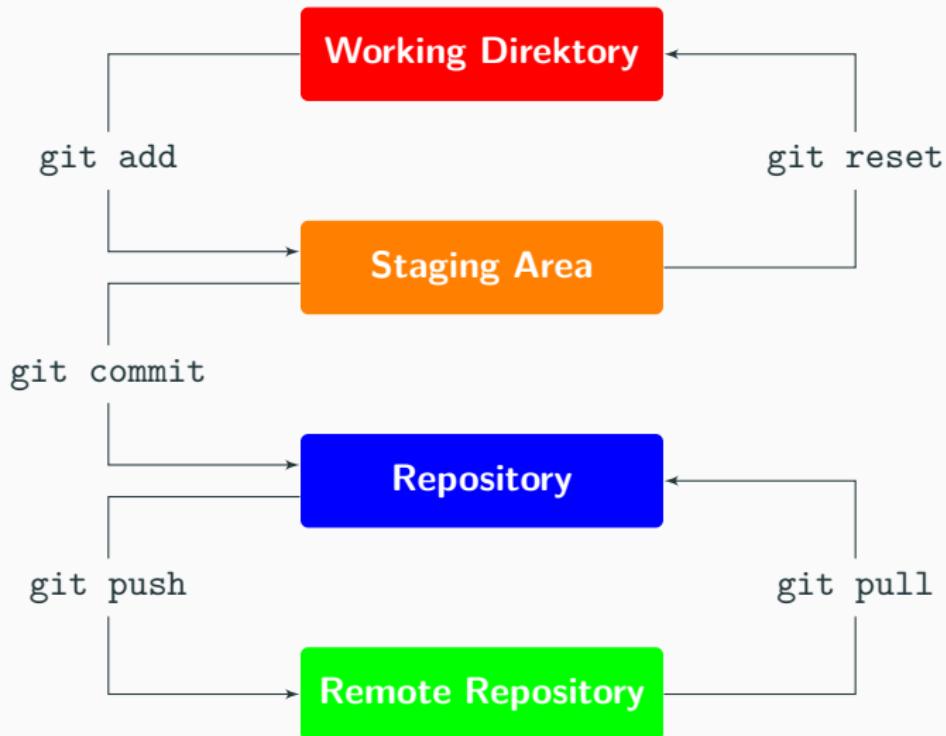
# git - Workflow



# git - Workflow



# git - Workflow



# Ab in die Cloud! - Befehle

**git push [Remote] [Branch]**

lädt eure Commits hoch (man nennt es auch *pushen*)

**git pull [Remote] [Branch]**

lädt fremde Änderungen runter (man nennt es auch *pullen*)

# Ab in die Cloud! - Befehle

**git push [Remote] [Branch]**

lädt eure Commits hoch (man nennt es auch *pushen*)

**git pull [Remote] [Branch]**

lädt fremde Änderungen runter (man nennt es auch *pullen*)

Aber dazu brauchen wir erstmal ein Remote Repository...

The screenshot shows a GitHub repository page for the user [jkrbs](#). The left sidebar lists the user's repositories, including [fsriese20-ansible](#), [fsriese-spiel-scoreboard](#), [fsrseminargruppen](#), [fsriehopanic](#), [jkrbsorganize](#), [jkrbszih\\_web\\_sim](#), [jkrbespiel\\_scoreboard](#), [fsriese-tutorienbriefing](#), [fsriese2020-hacky-feedback](#), [fsriese-esspiel-scoreboard](#), [jkrbsjitsirolette](#), [jkrbselecture\\_notes](#), [fsriese-system-status](#), and [fsriese-distribute](#). A red box highlights the [New](#) button in the sidebar. The main area displays a timeline of recent activity:

- Felixx42 pushed to fsriese-system-status** 5 days ago
  - 2 commits to [main](#)
    - [67be1dc](#) Add explanation on how to re-enable the workflow
    - [a53a631](#) Disable workflow for now
- Felixx42 pushed to fsriese20-services** 5 days ago
  - 1 commit to [main](#)
    - [82fc64c](#) Disable workflow for now
- bennofs created a repository ascii-dresdenascii.coffee** 11 days ago
  - [Star](#)

**ascii-dresdenascii.coffee**  
Infrastructure configuration for all ascii.coffee services  
Updated Nov 12
- kalinni pushed to fsrseminargruppen** 14 days ago
  - 1 commit to [master](#)

Search or jump to...  Pull requests Issues Marketplace Explore    

## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

---

Owner \*  Repository name \*  

Great repository names are short and memorable. Need inspiration? How about [special-eureka](#)?

Description (optional)

---

 **Public**  
Anyone on the Internet can see this repository. You choose who can commit.

 **Private**  
You choose who can see and commit to this repository.

---

Initialize this repository with:

Skip this step if you're importing an existing repository.

**Add a README file**  
This is where you can write a long description for your project. [Learn more.](#)

**Add .gitignore**  
Choose which files not to track from a list of templates. [Learn more.](#)

**Choose a license**  
A license tells others what they can and can't do with your code. [Learn more.](#)

---

**Create repository**

This screenshot shows a GitHub repository page for 'Felix42 / cooles-test-repo'. The page includes a navigation bar with links for 'Code', 'Issues 0', 'Pull requests 0', 'Projects 0', 'Wiki', 'Pulse', 'Graphs', and 'Settings'. Below the navigation bar, there's a section titled 'Quick setup — if you've done this kind of thing before' with instructions for setting up the repository. It provides links for 'Set up in Desktop' (with options for HTTPS or SSH), the repository URL (<https://github.com/Felix42/cooles-test-repo.git>), and a note about including README, LICENSE, and .gitignore files. There are also sections for creating a new repository on the command line (with a code block showing git commands) and pushing an existing repository from the command line (also with a code block). A third section for importing code from another repository is present but not highlighted.

Quick setup — if you've done this kind of thing before

Set up in Desktop or [HTTPS](#) [SSH](#) <https://github.com/Felix42/cooles-test-repo.git>

We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# cooles-test-repo" >> README.md
git init
git add README.md
git commit -m "first commit"
git remote add origin https://github.com/Felix42/cooles-test-repo.git
git push -u origin master
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/Felix42/cooles-test-repo.git
git push -u origin master
```

...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

## Ab in die Cloud! (2)

```
[felix@Samaritan] $ git remote add origin https://github.com/
    Feliix42/cooles-test-repo.git
[felix@Samaritan] $ git push -u origin master
Username for 'https://github.com': Feliix42
Password for 'https://Feliix42@github.com':
Counting objects: 3, done.
Writing objects: 100% (3/3), 869 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Feliix42/cooles-test-repo.git
 * [new branch]      master -> master
Branch master set up to track remote branch master from origin.
```

The screenshot shows a GitHub repository page. At the top, the URL is GitHub, Inc. [github.com/Felix42/cooles-test-repo](https://github.com/Felix42/cooles-test-repo). The repository name is Felix42 / cooles-test-repo. Below the header, there are tabs for Code, Issues (0), Pull requests (0), Projects (0), Wiki, Pulse, Graphs, and Settings. The Code tab is selected. A message says "No description or website provided. — Edit". Below this, there are summary statistics: 1 commit, 1 branch, 0 releases, and 1 contributor. A dropdown menu shows "Branch: master". Buttons for "New pull request", "Create new file", "Upload files", "Find file", and "Clone or download" are present. A commit history shows a single commit by Felix42 titled "Add hallowelt.txt" from 15 hours ago. A note at the bottom encourages adding a README, with a "Add a README" button.

© 2016 GitHub, Inc. Terms Privacy Security Status Help

Contact GitHub API Training Shop Blog About

Es funktioniert!

## Zum Abschluss: Kollaboration!

- findet euch zu zweit zusammen

## Zum Abschluss: Kollaboration!

- findet euch zu zweit zusammen
- Partner 1 fügt Partner 2 zu seinem Repository hinzu

Screenshot of a GitHub repository settings page for "jkrbs / my-test-repo".

The top navigation bar includes links for Pull requests, Issues, Marketplace, Explore, and a search bar.

The main header shows the repository name "jkrbs / my-test-repo" and icons for Unwatch (with 1), Star (0), Fork (0), and a gear icon.

The "Settings" tab is selected in the navigation bar.

The left sidebar contains the following options:

- Manage access (selected)
- Security & analysis
- Webhooks
- Notifications
- Integrations
- Deploy keys
- Autolink references
- Actions
- Secrets
- Moderation settings

The "Who has access" section compares two access levels:

- PUBLIC REPOSITORY**: Described as "This repository is public and visible to anyone." with a "Manage" button.
- DIRECT ACCESS**: Described as "0 collaborators have access to this repository. Only you can contribute to this repository." with a "Manage" button.

The "Manage access" section displays a message: "You haven't invited any collaborators yet" with an "Invite a collaborator" button.

## Zum Abschluss: Kollaboration!

- findet euch zu zweit zusammen
- Partner 1 fügt Partner 2 zu seinem Repository hinzu

## Zum Abschluss: Kollaboration!

- findet euch zu zweit zusammen
- Partner 1 fügt Partner 2 zu seinem Repository hinzu
- Partner 2 klont das Repository von Partner 1

The screenshot shows a GitHub repository page for 'Felix42 / cooles-test-repo'. The repository has 1 commit, 1 branch, 0 releases, and 1 contributor. It contains one file, 'hallowelt.txt', which has been added. A modal window is open, showing the 'Clone with HTTPS' URL: <https://github.com/Feliix42/cooles-test-repo>. The modal also includes 'Open in Desktop' and 'Download ZIP' buttons.

Felix42 / cooles-test-repo

1 commit · 1 branch · 0 releases · 1 contributor

Branch: master · New pull request

Clone with HTTPS ⓘ · Use SSH

Use Git or checkout with SVN using the web URL.

<https://github.com/Feliix42/cooles-test-repo>

Open in Desktop · Download ZIP

```
[harold@Machine] $ git clone https://github.com/Feliix42/cooles-
    test-repo.git
Cloning into 'cooles-test-repo'...
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
Checking connectivity... done.
```

## Zum Abschluss: Kollaboration!

- findet euch zu zweit zusammen
- Partner 1 fügt Partner 2 zu seinem Repository hinzu
- Partner 2 klont das Repository von Partner 1

## Zum Abschluss: Kollaboration!

- findet euch zu zweit zusammen
- Partner 1 fügt Partner 2 zu seinem Repository hinzu
- Partner 2 klont das Repository von Partner 1
- Bearbeitet beide `hallowelt.txt`  
(fügt z.B. den Satz „Ich heiße [...]“ hinzu und löscht den Rest.)

## Zum Abschluss: Kollaboration!

- findet euch zu zweit zusammen
- Partner 1 fügt Partner 2 zu seinem Repository hinzu
- Partner 2 klont das Repository von Partner 1
- Bearbeitet beide `hallowelt.txt`  
(fügt z.B. den Satz „Ich heiße [...]“ hinzu und löscht den Rest.)
- Einer von euch committed und pusht seine Änderungen  
(`git add [...]`, `git commit -m "[...]"`, `git push origin master`)

## Zum Abschluss: Kollaboration!

- findet euch zu zweit zusammen
- Partner 1 fügt Partner 2 zu seinem Repository hinzu
- Partner 2 klont das Repository von Partner 1
- Bearbeitet beide `hallowelt.txt`  
(fügt z.B. den Satz „Ich heiße [...]“ hinzu und löscht den Rest.)
- Einer von euch committed und pusht seine Änderungen  
(`git add [...]`, `git commit -m "[...]"`, `git push origin master`)
- Dann macht der andere Partner das gleiche

```
[felix@Samaritan] $ git commit -m "Ich bin Felix"

[master 50126f1] Ich bin Felix
 1 file changed, 1 insertion(+), 1 deletion(-)

[felix@Samaritan] $ git push origin master
To https://github.com/Feliix42/cooles-test-repo.git
 ! [rejected]           master -> master (fetch first)
error: failed to push some refs to 'https://github.com/Feliix42/
       cooles-test-repo.git'
hint: Updates were rejected because the remote contains work
hint: that you do not have locally. This is usually caused
hint: by another repository pushing to the same ref. You may
hint: want to first integrate the remote changes (e.g.,
hint: 'git pull ...') before pushing again. See the 'Note
hint: about fast-forwards' in 'git push --help' for details.
```

Huch.

Euer push wurde zurückgewiesen, weil es neuere Änderungen gibt, die ihr noch nicht auf eurem PC habt.

**Lösung:**

```
git pull origin master
```

```
[felix@Samaritan] $ git pull origin master
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/Feliix42/cooles-test-repo
  6386d2f..90188bf  master      -> origin/master
Auto-merging hallowelt.txt
CONFLICT (content): Merge conflict in hallowelt.txt
Automatic merge failed; fix conflicts and then commit the result
```

```
[felix@Samaritan] $ git pull origin master
remote: Counting objects: 3, done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/Feliix42/cooles-test-repo
  6386d2f..90188bf  master      -> origin/master
Auto-merging hallowelt.txt
CONFLICT (content): Merge conflict in hallowelt.txt
Automatic merge failed; fix conflicts and then commit the result
```

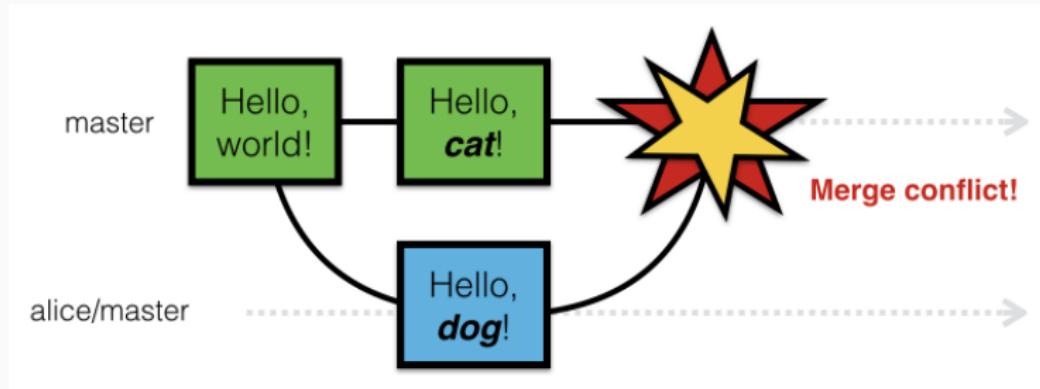
Glückwunsch! Euer erster *Merge-Conflict!*

# GIT PULL

# CONFLICT

memegenerator.net

## Okay... Und jetzt?



Überschneiden sich die Änderungen von zwei Commits, die git nicht automatisch beheben kann, entsteht ein **Merge Conflict**, der von Hand behoben werden muss.

# Wie?

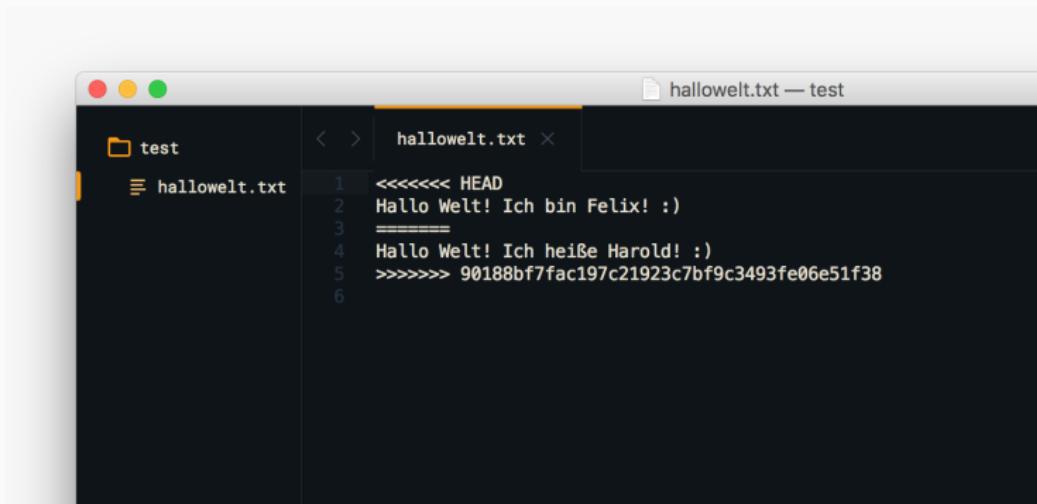
Öffnet hallowelt.txt im Texteditor.

Das sollte etwa so aussehen:

# Wie?

Öffnet hallowelt.txt im Texteditor.

Das sollte etwa so aussehen:



```
<<<<< HEAD
Hallo Welt! Ich bin Felix! :)
=====
Hallo Welt! Ich heiße Harold! :)
>>>>> 90188bf7fac197c21923c7bf9c3493fe06e51f38
```

## Was sehe ich da?

- oberhalb des ===== stehen eure Änderungen (markiert mit *HEAD*)
- darunter die Änderung des Konflikte verursachenden Commits  
(markiert mit der *Commit-ID*)

## Was sehe ich da?

- oberhalb des ===== stehen eure Änderungen (markiert mit *HEAD*)
- darunter die Änderung des Konflikte verursachenden Commits (markiert mit der *Commit-ID*)
- kompletter Abschnitt zwischen <<< und >>> muss jetzt von Hand gemerged werden
- löscht die Änderungen, die weg sollen und fasst alles so zusammen, dass es euch passt

## Was sehe ich da?

- oberhalb des ===== stehen eure Änderungen (markiert mit *HEAD*)
- darunter die Änderung des Konflikte verursachenden Commits (markiert mit der *Commit-ID*)
- kompletter Abschnitt zwischen <<< und >>> muss jetzt von Hand gemerged werden
- löscht die Änderungen, die weg sollen und fasst alles so zusammen, dass es euch passt
- seht euch mit `git status` den Zwischenstand an

```
[felix@Samaritan] $ git status
On branch master
Your branch and 'origin/master' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)
You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)

    both modified:  hallowelt.txt

no changes added to commit (use "git add" and/or "git commit -a"
")
```

## Was sehe ich da?

- oberhalb des ===== stehen eure Änderungen (markiert mit *HEAD*)
- darunter die Änderung des Konflikte verursachenden Commits (markiert mit der *Commit-ID*)
- kompletter Abschnitt zwischen <<< und >>> muss jetzt von Hand gemerged werden
- löscht die Änderungen, die weg sollen und fasst alles so zusammen, dass es euch passt
- seht euch mit `git status` den Zwischenstand an

## Was sehe ich da?

- oberhalb des ===== stehen eure Änderungen (markiert mit *HEAD*)
- darunter die Änderung des Konflikte verursachenden Commits (markiert mit der *Commit-ID*)
- kompletter Abschnitt zwischen <<< und >>> muss jetzt von Hand gemerged werden
- löscht die Änderungen, die weg sollen und fasst alles so zusammen, dass es euch passt
- seht euch mit git status den Zwischenstand an
- addet die „reparierte“ Datei
- committed das Ergebnis

# Merge Konflikte sind kein Beinbruch!

Generell gilt: Ruhe bewahren!

- Merge Konflikte immer mit Sorgfalt beheben, besonders wenn große Code-Segmente betroffen sind!
- Ein Konflikt kann sich über mehrere Blöcke erstrecken
- schaut euch betroffene Dateien genau an!



# Glückwunsch!

Ihr könnt jetzt:

- Euer Projekt mit git verwalten
- euer Repository mit GitHub über mehrere PCs und User synchronisieren
- Merge Konflikte beheben
- fremde Projekte klonen und mitmachen!

Ihr seid für eure ersten Schritte mit git gewappnet!

# Doch mit git ist noch viel mehr möglich!



Werft zum Beispiel mal ein Blick auf folgende Dinge:

- die .gitignore-Datei
- Branches
- git log
  - auch mit den Zusätzen
    - oneline
    - graph
- git diff
- git pull --rebase
- git commit --amend
- git fetch vs. git pull

# Hilfe, so viele Befehle! Das merk ich mir nie!

Musst du auch nicht, es gibt ja Cheat Sheets:

- [GitHub git Cheat Sheet Education](#)
- [Cheat Sheet von Tower](#)
- und viele mehr - einfach mal googlen

Außerdem gibt es zahlreiche Guides und Einführungen im Netz:

- [git - Der einfache Einstieg](#)
- [Git Explained: For Beginners](#)
- und viele mehr, schaut einfach, was euch anspricht!

**Tipp:** Unbedingt über Good Practices informieren!

## Ausblick

---

# Nächster Termin:

kommt noch per Mail

Freut euch auf:

- Wie geht es weiter? - Ausblick auf kommende Semester
- Tipps zum Stundenplanbau
- Wie gehe ich mit dem Prüfungsamt um?
- Und...ein paar letzte Tipps bevor ihr in die Welt der höheren Semester entlassen werdet :)