

PB4

Git

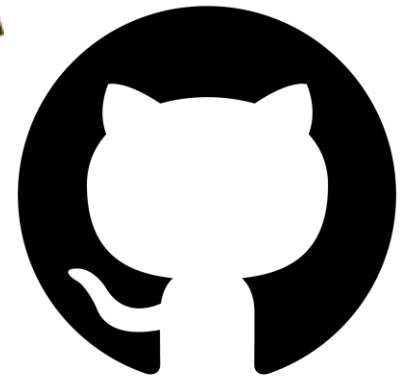
Felix Bärtschi / Blendor Uruci



git



GitLab



GitHub

<https://github.com/>

Inhaltsverzeichnis

- / Theoretische Grundlagen von GIT
- / Einführung in GitHub
- / Grundlegende GIT-Befehle
- / Erweiterte GIT-Funktionen

Theoretische Grundlagen von GIT

GIT

- Was ist Versionskontrolle
- Unterschiede zwischen GIT und anderen Versionskontrollsystemen
- Grundkonzepte

Versionskontrolle

- / **Versionskontrolle** (auch bekannt als **Version Control System, VCS**) ist ein System, das die Änderungen an Dateien im Laufe der Zeit verfolgt und verwaltet.
- / Es ermöglicht Entwicklern, Änderungen an ihren Projekten nachzuverfolgen und frühere Versionen wiederherzustellen, falls nötig.
- / Hauptfunktionen:
 - / Nachverfolgung von Änderungen
 - / Wiederherstellung von Versionen
 - / Zusammenarbeit
 - / Verzweigungen und Zusammenführung (**Branching** und **Merging**)

Unterschiede zwischen GIT und anderen VCSs

- / Verteilte (GIT) vs. Zentrale Versionskontrolle (Subversion, CVS)
- / Geschwindigkeit und Effizienz
- / Branching und Merging
- / Repository Struktur
- / Sicherheit und Datenintegrität

- / **GIT** bietet Flexibilität, Geschwindigkeit und eine robuste Verteilung von Daten, was es ideal für moderne Softwareentwicklungsprojekte macht.
- / **Zentrale Versionskontrollsysteme** wie Subversion und CVS sind einfacher zu verstehen und zu verwalten, jedoch weniger flexibel und oft langsamer.

Grundkonzepte

/ **Repository (Repo):**

- / Ein Repository ist ein Speicherort, der alle Dateien und deren Historie eines Projekts enthält.
- / Kann lokal auf einem Entwicklercomputer oder auf einer Plattform wie GitHub gehostet sein.

/ **Commit:**

- / Ein Commit ist eine gespeicherte Momentaufnahme des Projekts.
- / Enthält eine Nachricht, die die Änderungen beschreibt, und einen eindeutigen Hash, der den Commit identifiziert.

/ **Branch:**

- / Ein Branch ist ein Zeiger auf eine Reihe von Commits.
- / Ermöglicht parallele Entwicklungsstränge, um an verschiedenen Features oder Bugfixes zu arbeiten, ohne den Hauptentwicklungszweig zu stören.
- / Der Standardbranch heisst typischerweise „main“ oder „master“

/ **Merge:**

- / Merging ist der Prozess des Zusammenführens von Änderungen aus verschiedenen Branches
- / Ein Merge-Commit verbindet die Historie der zusammengeführten Branches und löst potenzielle Konflikte.

Grundkonzepte

/ **Clone:**

- / Ein Clone ist eine vollständige Kopie eines Repositorys, einschließlich der gesamten Historie und aller Branches.
- / Ermöglicht es Entwicklern, eine lokale Kopie eines Projekts zu erstellen und offline zu arbeiten.

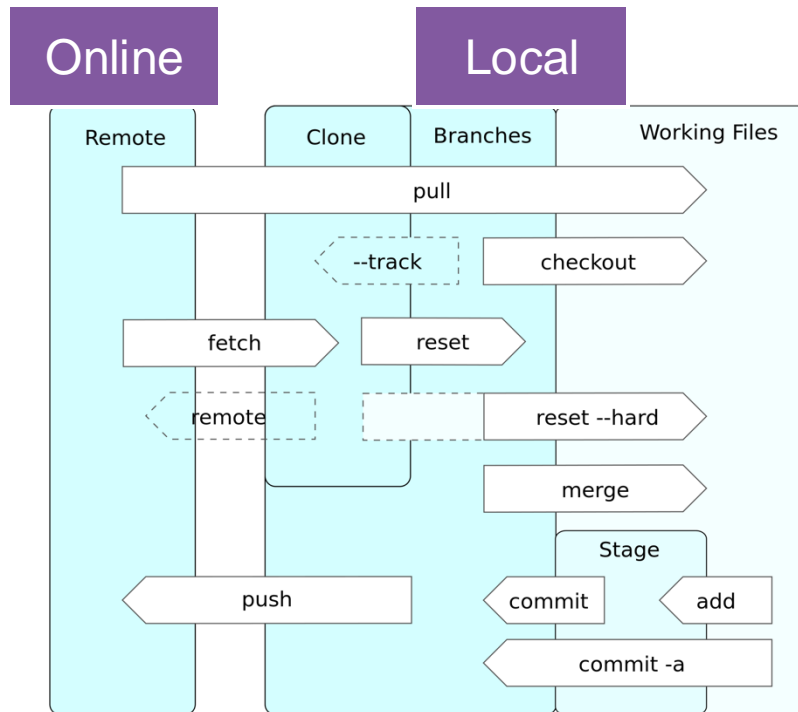
/ **Pull:**

- / Pull ist der Prozess des Herunterladens und Zusammenführens von Änderungen aus einem Remote-Repository in das lokale Repository.
- / Typischerweise verwendet, um die lokale Kopie auf dem neuesten Stand zu halten.

/ **Push:**

- / Push ist der Prozess des Hochladens lokaler Commits in ein Remote-Repository.
- / Wird verwendet, um Änderungen mit anderen Entwicklern zu teilen.

Funktionsweise GIT



Einführung in GitHub

GitHub

- Was ist GitHub und warum wird es verwendet
- Unterschiede zwischen GIT und GitHub
- Grundkonzepte

Was ist GitHub

- / Webbasierte Plattform die auf GIT basiert
- / Ermöglicht den Entwicklern, ihre Projekte zu hosten, zu verwalten und gemeinsam daran zu arbeiten
- / Wird verwendet aus den Gründen die GIT bietet und zusätzlich:
 - / Community und Open Source
 - / Continuous Integration/Continuous Deployment (CI/CD)
 - / Dokumentation und Wiki
 - / Sicherheit
 - / Integrationen und APIs

Unterschied zwischen GIT und GitHub

- / **GIT** ist ein Versionskontrollsystem, das lokal auf dem Rechner installiert wird und für die Verwaltung und Nachverfolgung von Codeänderungen verwendet wird.
- / **GitHub** ist eine Online-Plattform, die auf GIT basiert und zusätzliche Funktionen für die Zusammenarbeit, das Hosting und das Projektmanagement bietet

Feature	GIT	GitHub
Typ	Versionskontrollsystem	Webbasierte Plattform auf GIT-Basis
Installation	Lokal	Online-Dienst
Hauptfunktionen	Versionskontrolle, Branching, Merging	Repository-Hosting, Zusammenarbeit, Projektmanagement
Nutzung	Lokale Quellcodeverwaltung	Zentrale Plattform für Zusammenarbeit und Hosting
Kollaboration	Grundlegende Unterstützung	Erweiterte Funktionen (Pull Requests, Issues, etc.)
Benutzeroberfläche	Command-Line Tool	Web-Interface, Desktop-Anwendung

Grundkonzepte (zusätzlich zu GIT)

/ **Fork:**

- / Ein Fork ist eine Kopie eines Repositories, die auf Ihrem GitHub-Konto erstellt wird. Es ermöglicht Ihnen, Änderungen vorzunehmen, ohne das Original-Repository zu beeinflussen.
- / Häufig genutzt, um Open-Source-Projekte zu bearbeiten und Änderungen vorzuschlagen

/ **Pull Request (PR):**

- / Ein Pull Request ist eine Anfrage, um Änderungen aus einem Fork in das Original-Repository zu integrieren.
- / Ermöglicht es anderen, Ihre Änderungen zu überprüfen, Feedback zu geben und die Änderungen zu mergen.

/ **Wikis und Projekte:**

- / Ermöglichen die Erstellung von Projektdokumentationen direkt im Repository.
- / Kanban-Boards und Issues-Management-Tools zur Organisation von Aufgaben und Sprints.

Einzelaufgabe (30‘)

- / Installiere GIT
- / Richte dir ein GitHub Konto ein
- / Konfiguriere GIT
- / Erstelle ein neues Repository und erstelle deinen ersten Commit
- / **INFO:** Die nachfolgenden Folien helfen dir dabei

Installation von Git

- / Download Git
 - / <https://git-scm.com/downloads>
- / Erstelle einen Account
 - / <https://github.com/>

Konfiguriere GIT

/ Öffne „GIT BASH“

/ Setup: (Setze dein Username und deine E-Mail)

/ git config --global user.name "github_username,,

/ git config --global user.email "email_address,,

/ <https://phoenixnap.com/kb/how-to-install-git-windows>

Repository und Commit

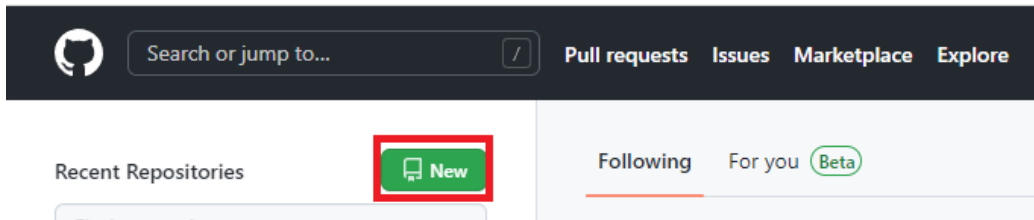
- / Erstelle einen neuen Ordner an einem geeigneten Ort (da wird deine Repository angelegt!)
- / Gehe im VS Studio command zu diesem Pfad
- / Gib den command ein «git init»
- / Erstelle ein neues File im VS Studio Code
- / Fülle es mit einem Python Code
- / Öffne die Source Control View in VS Code
- / Klicke auf das «+» Symbol, um die Datei zum Staging-Bereich hinzuzufügen
- / Gebe eine Commitnachricht an
- / Klicke auf das Häckchen

Gruppenaufgabe (45‘)

- / Erstelle ein Repository im GitHub mit Gruppenmitglied
- / Forke ein Repository um Pull Requests zu erstellen
- / Bearbeite und führe Pullrequest mit Gruppenmitglied aus
- / **INFO:** Die nachfolgenden Folien helfen dir dabei

Erstellen von Repository

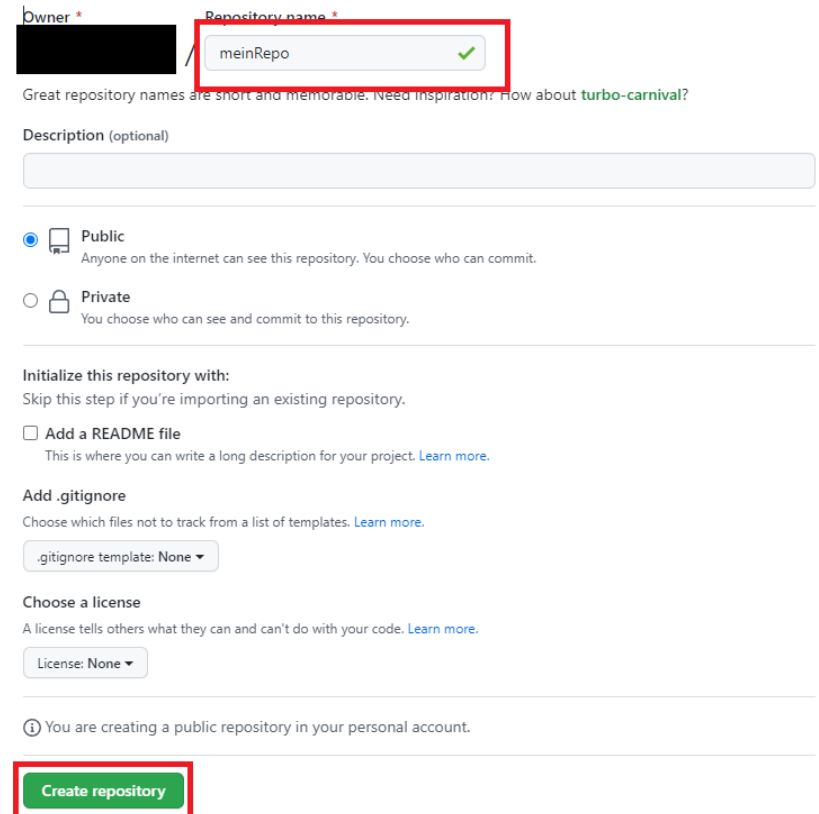
/ Klicke auf “New”



/ Vergebe einen Namen
/ Klick “Create repository”

Create a new repository

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

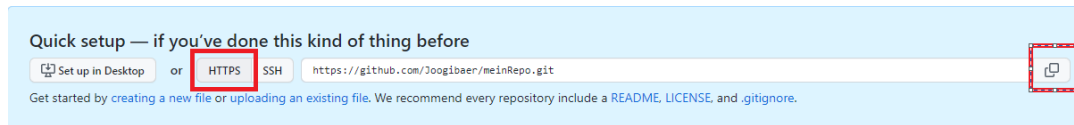
A screenshot of the 'Create a new repository' form on GitHub. The form is divided into several sections. The 'Owner' field is a black box. The 'Repository name' field contains the text 'meinRepo' and is highlighted with a red box. Below this, there is a note about repository names. The 'Description (optional)' field is empty. The 'Public' radio button is selected. The 'Initialize this repository with:' section has the 'Add a README file' checkbox selected. The 'Add .gitignore' section has a dropdown menu set to 'None'. The 'Choose a license' section has a dropdown menu set to 'None'. At the bottom, there is a red box around the 'Create repository' button.

Verbinde dein VS mit Repo

- / Geh zu den Einstellungen
- / Suche nach "GIT" und stelle sicher, dass der Pfad zur GIT-Installation korrekt ist.
- / Öffne den Extensions-Tab
- / Suche nach „GitHub“ und installiere die Erweiterung „GitHub Pull Requests and Issues“ von Microsoft

Hochladen des Lokalen Repos

/ „git push https://github.com/Name/meinRepo.git master“



Repository clonen

- / Öffne die Command Palette im VS Studio
- / Gebe „Git:Clone“ ein und wähle den Befehl „Git:Clone“
- / Füge die komplett kopierte URL aus dem GitHub Repository ein und wähle ein geeignetes Zielverzeichnis
- / Nach dem Clonen wird dir angeboten, das geklonte Repository zu öffnen. Klicke auf „Open“
- / Bearbeite jetzt Dateien im Repo (oder erstelle neue) und committe die Änderungen durch einen push auch in das Repository auf GitHub.

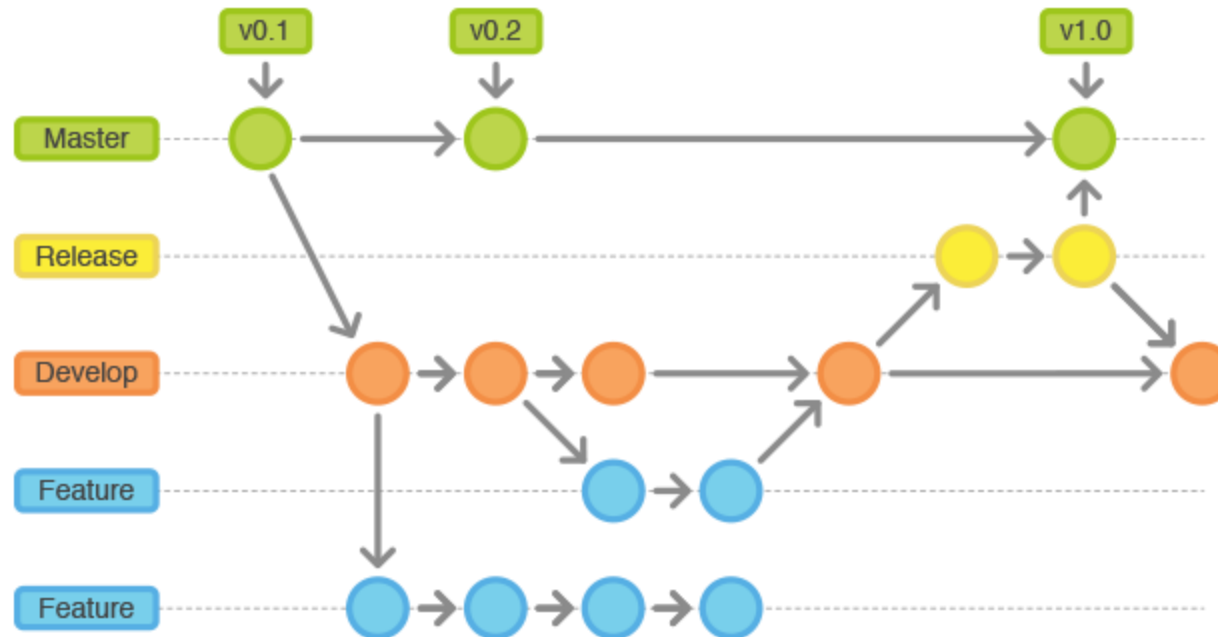
GitHub Pull Request

/ Im GitHub kannst du nun einen Pull Request erstellen:
<https://docs.github.com/de/pull-requests/collaborating-with-pull-requests/proposing-changes-to-your-work-with-pull-requests/creating-a-pull-request>

Konfliktlösung und Rollbacks

/ <https://docs.github.com/de/pull-requests/collaborating-with-pull-requests/incorporating-changes-from-a-pull-request/reverting-a-pull-request>

Branchien



Git Merge

- / git diff
 - / Anzeigen von Code unterschied
- / HEAD
 - Code von jemand anderem.
- / Passe den Code in dem Editor an
 - / Lösche den nicht benötigten Code
- / git add
 - / Hinzufügen des neuen Codes
- / git commit -m „some text“
 - / Hinzufügen von Änderungen als Text
- / git push „Git URL“

- / <https://git-scm.com/docs/git-merge>

GIT BASICS

<code>git init</code> <code><directory></code>	Create empty Git repo in specified directory. Run with no arguments to initialize the current directory as a git repository.
<code>git clone <repo></code>	Clone repo located at <code><repo></code> onto local machine. Original repo can be located on the local filesystem or on a remote machine via HTTP or SSH.
<code>git config</code> <code>user.name <name></code>	Define author name to be used for all commits in current repo. Devs commonly use <code>--global</code> flag to set config options for current user.
<code>git add</code> <code><directory></code>	Stage all changes in <code><directory></code> for the next commit. Replace <code><directory></code> with a <code><file></code> to change a specific file.
<code>git commit -m</code> " <code><message></code> "	Commit the staged snapshot, but instead of launching a text editor, use <code><message></code> as the commit message.
<code>git status</code>	List which files are staged, unstaged, and untracked.
<code>git log</code>	Display the entire commit history using the default format. For customization see additional options.
<code>git diff</code>	Show unstaged changes between your index and working directory.

UNDOING CHANGES

<code>git revert</code> <code><commit></code>	Create new commit that undoes all of the changes made in <code><commit></code> , then apply it to the current branch.
<code>git reset <file></code>	Remove <code><file></code> from the staging area, but leave the working directory unchanged. This unstages a file without overwriting any changes.
<code>git clean -n</code>	Shows which files would be removed from working directory. Use the <code>-f</code> flag in place of the <code>-n</code> flag to execute the clean.

REWRITING GIT HISTORY

<code>git commit</code> <code>--amend</code>	Replace the last commit with the staged changes and last commit combined. Use with nothing staged to edit the last commit's message.
<code>git rebase <base></code>	Rebase the current branch onto <code><base></code> . <code><base></code> can be a commit ID, branch name, a tag, or a relative reference to HEAD.
<code>git reflog</code>	Show a log of changes to the local repository's HEAD. Add <code>--relative-date</code> flag to show date info or <code>--all</code> to show all refs.

GIT BRANCHES

<code>git branch</code>	List all of the branches in your repo. Add a <code><branch></code> argument to create a new branch with the name <code><branch></code> .
<code>git checkout -b</code> <code><branch></code>	Create and check out a new branch named <code><branch></code> . Drop the <code>-b</code> flag to checkout an existing branch.
<code>git merge <branch></code>	Merge <code><branch></code> into the current branch.

REMOTE REPOSITORIES

<code>git remote add</code> <code><name> <url></code>	Create a new connection to a remote repo. After adding a remote, you can use <code><name></code> as a shortcut for <code><url></code> in other commands.
<code>git fetch</code> <code><remote> <branch></code>	Fetches a specific <code><branch></code> , from the repo. Leave off <code><branch></code> to fetch all remote refs.
<code>git pull <remote></code>	Fetch the specified remote's copy of current branch and immediately merge it into the local copy.
<code>git push</code> <code><remote> <branch></code>	Push the branch to <code><remote></code> , along with necessary commits and objects. Creates named branch in the remote repo if it doesn't exist.

GIT CONFIG

<code>git config --global user.name <name></code>	Define the author name to be used for all commits by the current user.
<code>git config --global user.email <email></code>	Define the author email to be used for all commits by the current user.
<code>git config --global alias.<alias-name> <git-command></code>	Create shortcut for a Git command. E.g. <code>alias.glog "log --graph --oneline"</code> will set "git glog" equivalent to "git log --graph --oneline".
<code>git config --system core.editor <editor></code>	Set text editor used by commands for all users on the machine. <code><editor></code> arg should be the command that launches the desired editor (e.g., vi).
<code>git config --global --edit</code>	Open the global configuration file in a text editor for manual editing.

GIT LOG

<code>git log --<limit></code>	Limit number of commits by <code><limit></code> . E.g. "git log -5" will limit to 5 commits.
<code>git log --oneline</code>	Condense each commit to a single line.
<code>git log -p</code>	Display the full diff of each commit.
<code>git log --stat</code>	Include which files were altered and the relative number of lines that were added or deleted from each of them.
<code>git log --author="<pattern>"</code>	Search for commits by a particular author.
<code>git log --grep="<pattern>"</code>	Search for commits with a commit message that matches <code><pattern></code> .
<code>git log <since>..<until></code>	Show commits that occur between <code><since></code> and <code><until></code> . Args can be a commit ID, branch name, HEAD, or any other kind of revision reference.
<code>git log -- <file></code>	Only display commits that have the specified file.
<code>git log --graph --decorate</code>	<code>--graph</code> flag draws a text based graph of commits on left side of commit msgs. <code>--decorate</code> adds names of branches or tags of commits shown.

GIT DIFF

<code>git diff HEAD</code>	Show difference between working directory and last commit.
<code>git diff --cached</code>	Show difference between staged changes and last commit

GIT RESET

<code>git reset</code>	Reset staging area to match most recent commit, but leave the working directory unchanged.
<code>git reset --hard</code>	Reset staging area and working directory to match most recent commit and overwrites all changes in the working directory.
<code>git reset <commit></code>	Move the current branch tip backward to <code><commit></code> , reset the staging area to match, but leave the working directory alone.
<code>git reset --hard <commit></code>	Same as previous, but resets both the staging area & working directory to match. Deletes uncommitted changes, and all commits after <commit> .

GIT REBASE

<code>git rebase -i <base></code>	Interactively rebase current branch onto <code><base></code> . Launches editor to enter commands for how each commit will be transferred to the new base.
---	---

GIT PULL

<code>git pull --rebase <remote></code>	Fetch the remote's copy of current branch and rebases it into the local copy. Uses git rebase instead of merge to integrate the branches.
---	---

GIT PUSH

<code>git push <remote> --force</code>	Forces the <code>git push</code> even if it results in a non-fast-forward merge. Do not use the <code>--force</code> flag unless you're absolutely sure you know what you're doing.
<code>git push <remote> --all</code>	Push all of your local branches to the specified remote.
<code>git push <remote> --tags</code>	Tags aren't automatically pushed when you push a branch or use the <code>--all</code> flag. The <code>--tags</code> flag sends all of your local tags to the remote repo.

Git Tools

- / Visualisierung von Git
 - / <https://git-scm.com/download/gui/windows>Visualisierung
- / Visualisierung von Commits
 - / <https://gource.io/>
- / Visualisierung von Branches
 - / <https://github.com/FredrikNoren/ungit>
 - / <https://sourceforge.net/projects/ungit.mirror/>