ddag_logo_blue_70_vekt



**Pro Git Buch**

Zusammenfassung Kapitel 1 bis 2

Version 1.0.0 von 2025

Ein Bild, das Text, Screenshot, Schrift, Logo enthält.

KI-generierte Inhalte können fehlerhaft sein.

Inhaltsverzeichnis

[1. Einleitung 3](#_Toc192249921)

[1.1 Absicht & Ziele 3](#_Toc192249922)

[1.2 Dokumentaufbau 3](#_Toc192249923)

[1.3 Voraussetzungen 4](#_Toc192249924)

[1.4 Voreinstellungen 4](#_Toc192249925)

[1.5 Aufgabenstellung 5](#_Toc192249926)

[1.6 Die drei Zustände 6](#_Toc192249927)

[2. Hauptteil 7](#_Toc192249928)

[2.1 Ein Git Repository anlegen 7](#_Toc192249929)

[2.2 Erstellen der Projektstruktur 7](#_Toc192249930)

[2.3 Zustand von Dateien prüfen 8](#_Toc192249931)

[2.4 Neue Dateien zur Versionsverwaltung hinzufügen 9](#_Toc192249932)

[2.5 Der erste commit 10](#_Toc192249933)

[2.6 Anzeigen der commit-Historie 11](#_Toc192249934)

[2.7 branches auf einen Blick 11](#_Toc192249935)

[2.8 Verzweigung zusammenführen 13](#_Toc192249936)

[2.9 branch löschen 14](#_Toc192249937)

[3. Schlusswort 15](#_Toc192249938)

[3.1 Vorgestellte Git Konzepte 15](#_Toc192249939)

[3.2 Vorgestellte Git Aktionen 15](#_Toc192249940)

# Einleitung

## Absicht & Ziele

Die Absicht des Verfassers ist es, die umfangreichen Kapitel (1, 2 & 3) der offiziellen Git Dokumentation markant zusammenzukürzen. Ziel ist es, dass Sie das Dokument inkl. aller Aufgaben in maximal einer Stunde und dreissig Minuten vollständig bearbeitet haben. Das Dokument fasst die ersten drei Kapitel der Git-Dokumentation zusammen. Diese Zusammenfassung ist weder vollständig noch abschliessend. Ziehen Sie dort, wo Sie mehr Informationen brauchen, unbedingt die offizielle Dokumentation zur Hilfe. [Hier](https://git-scm.com/book/de/v2/Git-Interna-Git-Objekte) gelangen Sie zur offiziellen Git Dokumentation. Falls Sie noch nie mit Git oder einem vergleichbaren Versionsverwaltungstool gearbeitet haben, empfehle ich Ihnen das [Kapitel 1.1](https://git-scm.com/book/de/v2/Erste-Schritte-Was-ist-Versionsverwaltung%3F) der Git-Dokumentation zu lesen. Für diejenigen mit ausgeprägtem geschichtlichem Interesse, kann im [Kapitel 1.2](https://git-scm.com/book/de/v2/Erste-Schritte-Kurzer-%c3%9cberblick-%c3%bcber-die-Historie-von-Git) zudem die Entstehungsgeschichte Git’s nachgelesen werden.

Ihr Ziel als Konsument dieses Dokumentes ist es, dass Sie in der Lage sind, eine zweckmässige Versionierung lokaler Projekte unter dem Einsatz von Git aufzubauen sowie sicherzustellen. Um das Ziel zu erreichen, setzen Sie dieses Dokument nur punktuell als Nachschlagewerk ein.

Aufbauend auf dem Wissen, dass Sie sich mithilfe dieses Dokumentes aneignen, werden Sie die remote Arbeit mit Git gleich zusammen mit den Entwicklungsguidelines der ddag erlernen. Dafür gibt es ein separates Dokument.

## Dokumentaufbau

Dieses Dokument ist traditionell aufgebaut. Es gliedert sich in Einleitung, Haupt- & Schlussteil.

## Voraussetzungen

Folgendes wird vorausgesetzt, um mit dem Kurs fortfahren zu können:

* Git wurde bereits auf dem Computer installiert

Überprüfen Sie die Installation von Git, indem Sie in der Eingabeaufforderung folgenden Befehl ausführen:

git --version

Ist Git bereits installiert sehen Sie diese Nachricht auf Ihrem Bildschirm. Die Versionsnummer kann sich dabei von meiner unterscheiden.



Haben Sie eine andere Anzeige? Dann installieren Sie Git, indem Sie der „Git installieren Schritt für Schritt Anleitung“ folgen.

…

## Voreinstellungen

Sie haben sicher bereits festgestellt, dass sich Computerprogramme konfigurieren lassen. Git stellt dabei keine Ausnahme dar.

Es ist wichtig das Sie, sobald Sie gemeinsam mit anderen Mitstreitern an einem Projekt arbeiten, an ihrem Namen erkennt werden. Deshalb personalisieren Sie Git, indem Sie nun Ihren Namen und E-Mail-Adresse festlegen. Sie werden die Auswirkung dieser Einstellungen in diesem Dokument noch nicht spüren, denn Sie arbeiten ausschliesslich lokal. Trotzdem ist es für Sie vorteilhaft, wenn Sie wissen, wie Sie Git für sich personalisieren können. Dafür ist es nie zu früh.

Um in Git Ihren Namen zu setzen, starten Sie die Eingabeaufforderung. Tippen Sie zuerst folgenden Befehl in die Kommandozeile ein.

git config --global user.name "<Ihr Name>"

Leider teilt Ihnen Git in diesem Fall nicht mit, ob der Name wirklich gesetzt wurde oder nicht. Sie können aber mithilfe des folgenden Befehls alle durchgeführten Konfigurationen einsehen:

git config --list

Hier sehen Sie die vorgenommene Einstellung (gelb markiert).

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Nun legen Sie mithilfe dieses Befehles noch Ihre E-Mail-Adresse fest. Hinterlegen Sie Ihre ddag-Firmenadresse.

git config --global user.email "<Ihre E-Mail-Adresse>"

Selbstverständlich können in Git viele weitere Konfigurationen durchgeführt werden. Da es aber nicht unser Ziel ist diese hier alle aufzulisten, müssen sich Interessierte abseits vertiefen.

## Aufgabenstellung

Ausgehend von der Aufgabe 6 aus dem C/SIDE Kurs 3 soll der Report rekonstruiert werden. Es soll eine Web Page für die Faktura Übersichtsliste bereitgestellt werden. Der Aufbau ist dabei sehr ähnlich. Das Resultat könnte wie folgt aussehen.

Ein Bild, das Text, Reihe, Schrift, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Sie erstellen und designen in diesem Kurs diese Web Page. Zum Einsatz kommen die gängigen Web-Technologien (HTML, CSS, JavaScript). Aufbauend auf Ihrer Arbeit könnte in einem weiteren Schritt die Anwendungslogik implementiert werden, um Daten aus der Datenbank zu holen und auf Ihrer stylisch designten Web Page anzuzeigen.

## Die drei Zustände

Git definiert drei Hauptzustände, in denen sich eine Datei befinden kann. Diese sind:

|  |  |  |
| --- | --- | --- |
| **Nr.** | **Name** | **Beschreibung** |
| 1 | modified | Eine Datei wurde geändert. Die Änderung wurde aber noch nicht versioniert. |
| 2 | staged | Die Änderung wird für eine künftige Versionierung vorgemerkt. |
| 3 | committed | Die Änderung wird versioniert und in der Datenbank abgespeichert. |

Betrachten Sie die Reihenfolge der drei Hauptzustände chronologisch. Zuerst verändert man eine Datei (modified). Danach merkt man diese Änderung für eine künftige Versionierung vor (staged). Als letztes versioniert man die Änderung und speichert diese in der Datenbank ab (committed).

Sie sollten nun bereit sein, um sich die Grundlagen von Git im Hauptteil mithilfe eines praktischen Beispiels anzueignen.

# Hauptteil

Obwohl Sie Git zwar bei sich installiert haben, wird momentan noch nichts versioniert. In anderen Worten bedeutet das, dass Git zwar bei Ihnen installiert ist, Sie Git aber noch nicht aktiv nutzen. Das liegt daran, dass Git nicht automatisch alle Dokumente auf Ihrem Computer versioniert – das wäre allenfalls ein ganz schön grosses Projekt. Um Git aktiv nutzen zu können, müssen Sie Git ein Verzeichnis mitteilen, welches Sie versionieren möchten. Git übernimmt dann die Versionierung für Sie.

## Ein Git Repository anlegen

Wir erstellen nun gemeinsam ein sogenanntes Git Repository an einem spezifischen Ort auf Ihrem Computer. Falls Sie den Begriff Repository nicht verstehen sollten, nehmen Sie sich einige Minuten Zeit und suchen Sie im Internet nach Synonymen. Sie werden dabei auf Wörter wie storage, depot, container oder ähnliches stossen. Es reicht aus, wenn Sie ein Repository als Container sehen, und verstehen, dass wir innerhalb dieses Containers von den zahlreichen Vorteilen von Git profitieren wollen.

Aufgabe: Erstellen Sie unter folgendem Pfad „C:\Users\<username>\Desktop“ einen neuen Ordner mit der Bezeichnung „learning\_git“. Wechseln Sie daraufhin in der Eingabeaufforderung in das eben erstellte Verzeichnis. Vergewissern Sie sich, dass Ihr Pfad in der Eingabeaufforderung mit dem im untenstehenden Bild übereinstimmt, bevor Sie Git-Befehle ausführen.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Aufgabe: Legen Sie mithilfe dieses Befehls ein neues Git Repository an.

git init

Öffnen Sie nun den Ordner „learning\_git“ in einem Explorer und schauen Sie sich das Resultat des eben ausgeführten Git-Befehls an. Sie sehen ein Unterverzeichnis „.git“. Darin enthalten sind alle relevanten Git-Repository-Daten. Wenn Sie sich fundiertere Kenntnisse aneignen möchten, erfahren sie [hier](https://git-scm.com/book/de/v2/Git-Interna-Basisbefehle-und-Standardbefehle-Plumbing-and-Porcelain#ch10-git-internals) mehr über das Unterverzeichnis .git.

## Erstellen der Projektstruktur

Sie werden jetzt das Grundgerüst des Projektes implementieren. Erstellen Sie dazu in einem ersten Schritt folgende Elemente. Als Ausgangpunkt dient Ihnen das Hauptverzeichnis „learning\_git“.

|  |  |  |  |
| --- | --- | --- | --- |
| **Nummer** | **Elementtyp** | **Name** | **Beschreibung** |
| 1 | Datei | index.html |  |
| 2 | Ordner | css |  |
| 2.1 | - Datei | - style.css | Befindet sich im Ordner css. Enthält die css-Anweisungen. |
| 3 | Ordner | Js |  |
| 3.1 | - Datei | - script.js | Befindet sich im Ordner js. Enthält den javascript-Code. |
| 4 | Datei | README.md | Beschreibt, was mit Ihrem Projekt gemacht werden kann und wie es benutzt wird. |

Kopieren Sie in einem zweiten Schritt den nachfolgenden Code in die Datei „index.html“.

<!DOCTYPE html>

<html lang="de">

  <head>

    <meta charset="utf-8" />

    <title>Faktura - Übersichtsliste</title>

    <link rel="stylesheet" href="./css/style.css">

    <script src="./js/script.js"></script>

  </head>

  <body>

  </body>

</html>

Fügen Sie in einem letzten Schritt folgenden Text in die Datei „README.md“ ein.

*Dieses Projekt dient dazu die Grundlagen von Git praktisch zu erlernen. Das Projekt wurde am <Datum> durch <Ihr Kürzel > erstellt.*

## Zustand von Dateien prüfen

Sie haben jetzt mehrere Ordner & Dateien erstellt sowie letztere modifiziert. Schauen Sie sich nun an, ob und falls ja, was Git davon mitbekommen hat. Führen Sie folgenden Befehl in der Kommandozeile aus.

git status

Wie Sie sehen, war Git über jeden einzelnen Schritt bestens informiert.

Ein Bild, das Text, Screenshot, Software, Multimedia-Software enthält.

KI-generierte Inhalte können fehlerhaft sein.

Beschriftung: Antwort auf den obenstehenden Befehl

Legen Sie Ihren Fokus nur auf die gelb markierten Zeilen. In der oberen heisst es „Untracked files“. „To track something“ bedeutet auf Deutsch „etwas verfolgen.“ Untracked steht somit für das Gegenteil. In unserem Kontext bedeutet das, dass Git die Dateien als neue Dateien erkannt hat. Untracked ist aktuell deren Status, weil noch keine der Dateien in der Versionierung auftaucht. Git kennt diese Dateien aktuell noch und verfolgt sie aus diesem Grund auch nicht.

**Vorsicht in Bezug auf die 3 Hauptzustände:** Verwechseln Sie die die Tracked/ Untracked Zustände einer Datei nicht mit den 3 Hauptzuständen, die weiter oben im Dokument beschrieben wurden. Eine Datei kann sich ausschliesslich in einem der 3 Hauptzuständen befinden, wenn sie bereits versioniert wurde – d.h. durch Git getracked wird.

## Neue Dateien zur Versionsverwaltung hinzufügen

Fügen Sie nun die neuen Dateien zur Versionsverwaltung hinzu, indem Sie folgenden Befehl ausführen.

git add <Datei- oder Verzeichnisname>

Führen Sie den Befehl so lange aus, bis alle neu erstellten Ordner und Dateien hinzugefügt wurden. Prüfen Sie dann den Zustand der Dateien erneut.

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.Nun sind alle Dateien grün und wurden als Change (Änderung) erkannt.

Lassen Sie sich noch einmal die 3 Hauptzustände durch den Kopf gehen und fragen Sie sich, welchen Zustand die Dateien durch den git add Befehl angenommen haben.

## Der erste commit

Die Dateien befinden sich jetzt im Hauptzustand staged – d.h. für eine künftige Versionierung vorgemerkt. Nun geht es darum eine Versionierung durchzuführen. Tippen Sie dafür folgenden Befehl in die Eingabeaufforderung ein.

git commit

Daraufhin öffnet sich Ihr Standardeditor. Jede Versionierung braucht einen aussagekräftigen Beschreibtext. Dies setzt Git so voraus. Geben Sie nachfolgenden Satz ein.

*Grundgerüst Web Page Projekt erstellt.*

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.Speichern Sie den Text ab und schliessen Sie den Editor. Damit haben Sie Ihre erste Versionierung durchgeführt. In der Sprache von Git spricht man dabei von einem commit. Jeder commit wird durch einen eindeutigen Wert identifiziert – der sogenannten Prüfsumme basierend auf dem SHA1-Algorithmus. Für diejenigen, die gerne mehr über diese Prüfsumme erfahren möchten, gibt es [hier](https://git-scm.com/book/de/v2/Git-Interna-Git-Objekte) mehr Informationen. Es reicht aber wenn Sie wissen, dass ein Teil dieser Prüfsumme beim Erstellen eines commits angezeigt wird.

In meinem Fall startet die Prüfsumme des commits mit **b00b8d6**.

Es gibt auch eine schnellere Variante, um einen commit zu erstellen. Indem Sie den Befehl erweitern, verhindern Sie, dass sich der Editor zum eingeben des commit-Beschreibtextes öffnet.

git commit -m „Grundgerüst Web Page Projekt erstellt.“

Nun gibt es in Git unzählige Kurzvarianten für einzelne Befehle, dass es den Rahmen sprengt, um in diesem Dokument darauf einzugehen.

Üben Sie den beschriebenen Ablauf, indem Sie folgende Aufgaben lösen. **Pro Aufgabe erstellen Sie einen commit.**

Aufgabe 1: Erstellen Sie im Hauptverzeichnis einen neuen Ordner. Benennen Sie ihn „img“. Fügen Sie dem Ordner folgendes Bild hinzu.



Aufgabe 2: Implementieren Sie den Webseiten-Header. Orientieren Sie sich dabei an dieser Vorlage.

Ein Bild, das Screenshot, Text, Reihe enthält.

KI-generierte Inhalte können fehlerhaft sein.

## Anzeigen der commit-Historie

Lass uns kurz durchatmen und eine Pause einlegen. In einer modernen Welt ist es normal Aufgaben zu erhalten, diese zu bearbeiten und dann direkt mit neuen wieder bombardiert zu werden. Viel zu selten nehmen wir uns die Zeit und blicken zurück auf das, was wir bereits erreicht haben. Umso besser bietet uns Git die Möglichkeit genau das zu tun. Tippen Sie nachfolgenden Befehl in die Eingabeaufforderung und führen Sie ihn aus.

git log

Sie sehen Ihre drei bereits erstellten commits in umgekehrter chronologischer Reihenfolge. Studieren Sie die Ausgabe und notieren Sie sich weitere Eigenschaften, die Sie aus der Ausgabe herauslesen können und bereits kennen. Notieren Sie sich des Weiteren die ersten sieben Zeichen der Prüfsumme des neusten commits.

## branches auf einen Blick

Ein Bild, das Baum, Frucht, Zweig, Beeren enthält.

KI-generierte Inhalte können fehlerhaft sein.Zuerst beseitigen wir allfällige Unklarheiten in Bezug auf den Begriff „branch“. Das Wort branch bedeutet im deutschen Zweig. Dazu ein Bild eines jungen, gesunden und grünen Asts😉. Auf die Nummern und unterschiedlichen Farben gehen wir gleich ein.

Führen Sie sich noch einmal das Big-Picture vor Augen. Sie wollen eine Web Page nach dem Vorbild der Aufgabe 7 aus dem C/SIDE Kurs 3 erstellen. Mit dem Page Header haben Sie bereits einen Teil der Aufgabe gelöst – bravo. Jetzt soll die Tabelle implementiert werden, die die Daten aus dem ERP System hält.

Angenommen Sie haben nur eine vage Vorstellung davon, wie Sie diese Implementierung umsetzen. Ihnen gehen verschiedene Lösungswege durch den Kopf, jedoch haben Sie Angst, beim Ausprobieren Ihren aktuellen Projektstand zu beschädigen. Erstellen Sie mithilfe des folgenden Befehls einen neuen branch, welcher als Ort zum Ausprobieren dient. Nennen Sie den branch dev.

git branch dev

Schauen Sie mithilfe des folgenden Befehls nach, ob der neue branch existiert.

git branch

Ein Bild, das Text, Screenshot, Schrift enthält.

KI-generierte Inhalte können fehlerhaft sein.

Was fällt auf? Nebst dem soeben erstellten branch dev existiert ein weiterer branch – in meinem Fall mit dem Namen main. Das ist der default branch. Dieser wurde automatisch erstellt, als Sie Ihren ersten commit durchgeführt haben. Daraus lässt sich ableiten, dass **branches immer an einen commit gekoppelt sind**. Ich zitiere an dieser Stelle einen Auszug aus der Git Dokumentation: „Ein branch in Git ist einfach ein leichter, beweglicher Zeiger auf einen dieser commits.“ Sie finden [hier](https://git-scm.com/book/de/v2/Git-Branching-Branches-auf-einen-Blick) eine ausführliche Beschreibung über das Verhältnis von branches und commits.

Zurück zum Ast und den Nummern in verschiedenen Farben. Der default branch ist der Hauptzweig des Projekts und wird durch die blaue eins repräsentiert . Von ihm können weitere Zweige abzweigen. Repräsentiert durch die roten Nummern  .

Der default branch ist in Ihrem Fenster grün markiert und wird durch ein führendes \* Symbol eingeleitet. Das bedeutet nur, dass Sie sich gerade auf diesem branch befinden. Wechseln Sie in den dev branch, um die Implementierung der Tabelle vorzunehmen. Sie können folgendermassen zwischen branches hin- und herwechseln.

git checkout dev

|  |
| --- |
| Beachten Sie, dass Git das Wechseln zu einem anderen Branch blockiert, falls Ihr Arbeitsverzeichnis oder Ihr Staging-Bereich nicht committete Modifikationen enthält. Generell ist es am besten, einen sauberen Zustand des Arbeitsbereichs anzustreben, bevor Sie Branches wechselst.Informationen Silhouette |

Aufgabe: Erstellen Sie nun die Tabelle, fügen Sie alle geänderten Dateien zur Staging-Area hinzu und führen Sie einen commit durch.

Ein Bild, das Text, Reihe, Schrift, Screenshot enthält.

KI-generierte Inhalte können fehlerhaft sein.

Aufgabe: Überprüfen Sie mithilfe dieses bereits bekannten Befehls auf welchen commit der branch dev zeigt.

git log

Beantworten Sie zudem, ob sich die Prüfsumme dieses commits von der unterscheidet, auf die der branch main zeigt. Wenn ja, warum unterscheiden sie sich?

## Verzweigung zusammenführen

Es scheint so, als war eine Ihrer Lösungsvariante, die Ihnen vorhin durch den Kopf ging, erfolgreich – herzlichen Glückwünsch! Denn ansonsten wären Sie jetzt nicht an dieser Stelle im Dokument angelangt. Es war durchaus sinnvoll einen neuen branch zu erstellen, um die Erweiterung zu implementieren. Damit haben Sie verhindert, dass Ihr Projekt aufgrund Fehlentwicklungen oder Gedankenfehlern beschädigt wird. Sie haben während allen Phasen über ein sicheres Backup verfügt, welches getrennt von Ihrem Arbeitsbereich im default branch lag.

Angenommen Ihre Erweiterung ist abgeschlossen und getestet. Dann können Sie diese mit zwei einfachen Befehlen mit dem Hauptzweig des Projektes zusammenführen. Wechseln Sie dafür in einem ersten Schritt zurück zum default branch.

git checkout <name default branch>

Führen Sie Ihre Erweiterung mit dem Projektstand in Ihrem default branch zusammen, indem Sie folgenden Befehl ausführen.

git merge dev

Öffnen Sie das Projekt erneut in Visual Studio Code und überprüfen Sie, ob Ihre Erweiterung eingebunden wurde. Stellen Sie dabei sicher, dass Sie sich in Ihrem default branch befinden.

Dieses Beispiel beschreibt die einfachste Variante einer Zusammenführung – ruhen Sie sich also nicht darauf aus! Sie finden [hier](https://git-scm.com/book/de/v2/Git-Branching-Einfaches-Branching-und-Merging) weitere Informationen inkl. anspruchsvolleren Situationen, die eine Zusammenführung massiv verkomplizieren. Lesen Sie die Abschnitte *Einfaches Merging* und *Einfache* *Merge-Konflikte*.

## branch löschen

Sobald Ihre Erweiterung erfolgreich in den Hauptzweig des Projektes integriert wurde, können Sie den branch mit dem Namen dev löschen. Führen Sie dazu folgenden Befehl aus.

git branch -d dev

Überprüfen Sie mit einem Ihnen bereits bekannten Befehl, ob das Löschen wirklich erfolgreich war. Welchen Befehl verwenden Sie dafür?

# Schlusswort

Abschliessend werden alle in diesem Dokument vorgestellten Git Konzepte und Aktionen noch einmal aufgelistet. Beschreiben Sie jeden Punkt kurz, aber prägnant und in eigenen Worten.

## Vorgestellte Git Konzepte

Folgende Git Konzepte wurden in diesem Dokument vorgestellt:

* Die drei Zustände in Git
* Tracked & untracked
* Kurzvarianten in Git
* Abzweigen vom Hauptzweig/ Hauptlinie (branch)

## Vorgestellte Git Aktionen

Folgende Aktionen wurden in diesem Dokument beschrieben:

* Basiskonfiguration in Git
* Anlegen eines lokalen repository
* Zustandsüberprüfung einzelner Dateien
* Neue Dateien zur Versionsverwaltung hinzufügen
* Einen Projektschnappschuss erstellen (commit)
* Anzeigen der commit-Historie
* Verzweigung erstellen (branch)
* Zwischen Zweigen hin- und herwechseln
* Verzweigungen zusammenführen
* Verzweigung löschen