Was ist git?

Versionsverwaltungssystem

speichert in sogenannten repositories:

- Inhalt aller Dateien eines Projekts
- gesamte Änderungsgeschichte eines Projekts
- Metadaten (Änderungszeitpunkt, Autor, Kommentare, Versionsnummer)

verteiltes Versionsverwaltungssystem

Repositories:

- verschiedene repositories befinden sich auf verschiedenen Computern
- die Repositories können untereinander synchronisiert werden
- die Arbeitskopie wird mit dem lokalen repository synchronisiert
- enthalten commits, also Änderungen/Versionen

Snapshot-basiertes Versionsverwaltungssystem

Dateien:

- werden vollständig im aktuellen Zustand gespeichert
- unveränderte Dateien werden dupliziert
- Diff wird bei Bedarf generiert

Staging Area/Index

Zwischenschritt zwischen working copy und repository
ermöglicht bessere Kontrolle darüber was in die neue Version übernommen wird

Hash Adressierung

Versionen werden mittels Hash-Adressierung gespeichert und referenziert

Kommandozeilen Anwendung

Viele nützliche Kommandozeilenbefehle

Was ist GitHub?

GitHub ist ein Unternehmen, dass repositories hostet.
Dies erlaubt uns ein remote repository aufzusetzen auf dass wir alle zugreifen.
Dieses remote repository stellt für uns eine Art "single source of truth" dar.

Was ist ein Branch?

Branches sind pointer zu bestimmten commits.
Branches erlauben uns parallel an features zu arbeiten, während die Hauptversion intakt bleibt.

Head ist ein pointer der zum aktuell betrachteten commit zeigt.
main ist der pointer zur "Hauptversion".

Was ist ein Merge?

Bei einem merge werden mehrere commits zu einem neuen commit zusammengefasst.

Was ist ein Merge Konflikt?

Ein merge funktioniert im besten Fall automatisch, falls dies bspw. durch Änderungen in beiden commits in der gleichen Zeile nicht möglich sein sollte, müssen diese "Merge Konflikte" manuell gelöst werden.

Was sind Pull Requests und Issues

Issues helfen uns zu organisieren was zu tun ist.
Pull Requests helfen uns dabei abzusichern, dass das was getan wurde sicher in die Hauptversion übernommen werden kann.

Issue

Ein Issue kann manuell auf GitHub erstellt werden.
Es handelt sich um eine präzise Beschreibung einer Aufgabenstellung für eine Änderung.
Wenn man ein Issue lösen möchte trägt man sich selbst als Assignee ein, damit das Issue klar vergeben ist.
In den meisten Fällen arbeitet man für die Bearbeitung auf einem neuen Branch.

Pull Request

Ein Pull Request kann ebenfalls auf GitHub erstellt werden, nachdem man einen commit auf einem anderen Branch als main gemacht hat.
In den body der PR gehört "resolves #nummerDesGelöstenIssues" und eine Beschreibung der Änderung.
Die Idee ist, dass andere die Abgabe nochmal überprüfen, bevor sie in die Hauptversion übernommen wird.
Man kann andere als Reviewer für seine Pull Request berufen.

Aufgaben als Reviewer

Den commit der Pull Request auf Korrektheit prüfen.
Falls nein, einen konstruktiven Kommentar schreiben und auf neue Version der Pull Request warten.
Falls ja, die Korrektheit zurückmelden und danach je nach workflow selbst mit main mergen oder vom Ersteller der PR mergen lassen.
Die Person die merged ist dann für die Überprüfung der Korrektheit des merges und für das pushen von diesem ans remote repository zuständig.

einfache git Kommandozeilenbefehle

Tipps zum Nutzen der Shell

Einfügen und Kopieren funktioniert mit Rechtsklick, nicht mit Strg + V.
Mit den Pfeiltasten oben und unten können vorherig genutzte Befehle abgerufen werden.
Ein Punkt . bedeutet alles.
Wenn die Ausgabe größer ist als das Fenster der Kommandozeile: mit Enter erweitern, mit q entkommen.

Kommandozeilenbefehle

Befehle zum Verstehen der Situation

Erklärung eines Befehls:
git help *command*

Änderungsgeschichte ansehen:
git log
Änderungsgeschichte als Graph ansehen:
git log --oneline --graph --all --decorate

Geänderte Dateien, gestagede Dateien, fehlende Dateien etc. mit Beschreibungen:
git status

Unterschiede zwischen Arbeitskopie und Head, bzw. Arbeitskopie und Staging Area sehen:
git diff [--staged]

Befehle zum managen der Repositories

Aktuellen Dateipfad ändern:
cd *dateipfad*
(Man sollte auf den Ordner des repositories verweisen wenn man in diesem arbeiten will.)

Neues Repository erschaffen:
git init

Repository klonen:
git clone *dateipfad/link*
(Der Link zu einem remote repository auf gitHub ist unter dem grünen "Code" Button zu finden.)

Änderungen in die Staging Area bringen:
git add *file/*.

Änderungen von der Staging Area entfernen:
git reset

Änderungen von der Staging Area in das lokale repository übernehmen:
git commit -m "*description of the change*"

Datei oder Version zur Arbeitskopie hinzufügen:
git checkout *file/commit*

Die Änderungen von einem lokalen Branch ans remote repository schicken:
git push [origin *branch*]

Lokales repository mit dem remote repository "ersetzen":
git fetch --all

Befehle zu Branches

Neuen Branch erstellen oder zwischen branches wechseln:
git checkout *branchname*
lokalen Branch löschen:
git branch -d *branchname*
remote Branch löschen:
git push origin --delete *branchname*

Neuesten commit eines anderen branches in den aktuellen commit mergen:
git merge *branchname*

commits interaktiv umorganisieren:
git rebase -i *branchname*
(Es werden meist neue commits generiert, rebase ist ein etwas fortgeschrittener Befehl)