Technische Universität Braunschweig

Institut für Flugführung

Versionskontrolle mit GitHub

Prof. Dr.-Ing. Peter Hecker, Dipl.-Ing. Paul Frost, Andreas Dekiert M. Sc.,
24. April 2018

Agenda

03. April	Einführung
10. April	Softwareprojektmanagement
17. April	Entwicklungstools
24. April	**GitHub**
08. Mai	Einführung Arduino/Funduino
15. Mai	Dateieingabe und -ausgabe
22. Mai	Exkursionswoche
29. Mai	Dokumentation und Bug-Reporting
05. Juni	Einführung von Qt
12. Juni	GUI-Erstellung mit Qt
19. Juni	Anleitung erstellen
26. Juni	Projektarbeit
03. Juli	Vorbereitung der Abgabe
10. Juli	Abgabe

Slack-Update

Themen der letzten Woche:

- User-Stories
  - Unsicherheit bei der Auslegung von User-Stories

#coding_cpp

- Zielkriterien und Aufgaben
  - Was ist der Unterschied zwischen einer Aufgabe und einem Zielkriterium?

#projektmappe

Lehrziele

Versionskontrolle mit GitHub

Als Teilnehmer soll ich am Ende dieser Übung...

- ☐ Konflikte in git lösen können
- ☐ an bestehenden Projekten weiterarbeiten können
- ☐ das lokale Repository mit GitHub synchronisieren können
- ☐ das Projekt-Wiki erstellen und mit Inhalt füllen können


git-Grundlagen


Recap
Konflikte

Repository .git

Der versteckte `.git`-Ordner ist das eigentliche Repository („Repo“).

- git speichert dort alle Informationen
- Das Repository kann nur wachsen
- Neben dem Repo liegt das Working Directory. Dort wird gearbeitet.

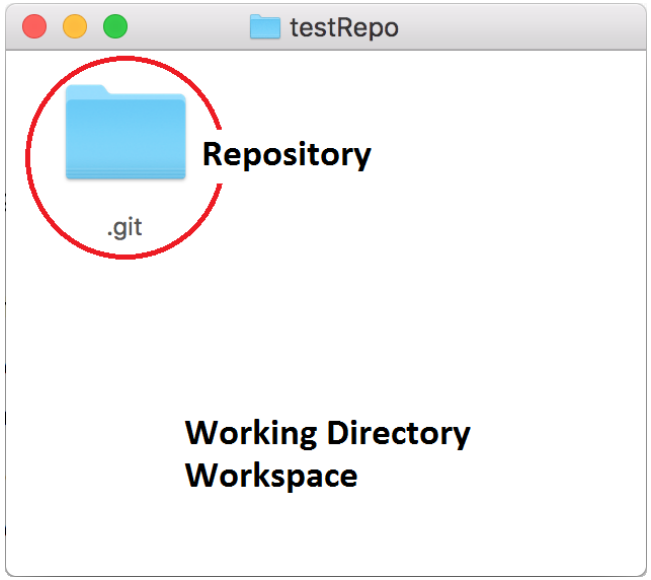


Abbildung 1: Arbeitsverzeichnis, Staging-Area und Git Verzeichnis [1]

Der `.git`-Ordner darf nicht gelöscht werden, wenn das Repository und die Versionsverwaltung lokal weiter zur Verfügung stehen sollen.

Repository .git

Der versteckte `.git`-Ordner ist das eigentliche Repository („Repo“).

- git speichert dort alle Informationen
- Das Repository kann nur wachsen
- Neben dem Repo liegt das Working Directory. Dort wird gearbeitet.

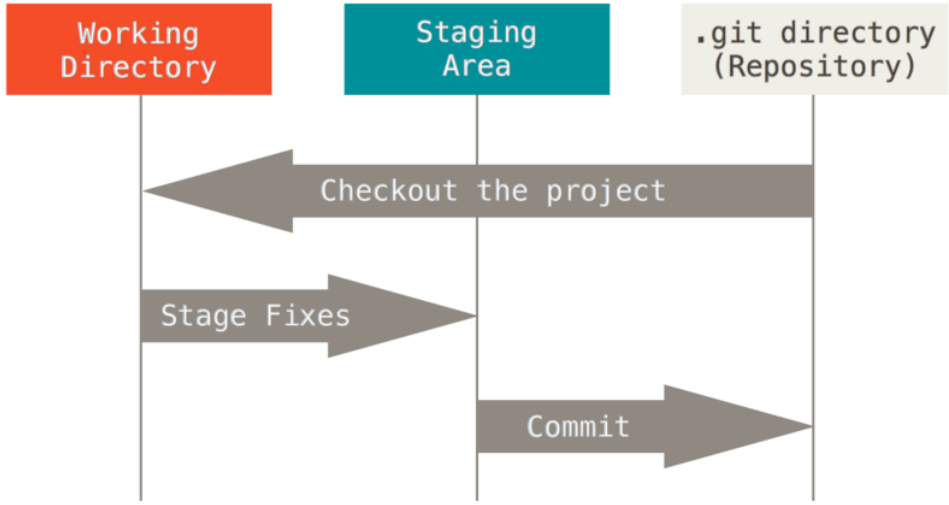


Abbildung 1: Arbeitsverzeichnis, Staging-Area und Git Verzeichnis [1]

Der `.git`-Ordner darf nicht gelöscht werden, wenn das Repository und die Versionsverwaltung lokal weiter zur Verfügung stehen sollen.

Daten sichern

Commit

Mit einem Commit werden Daten als Revision im Repository gesichert.

Shell

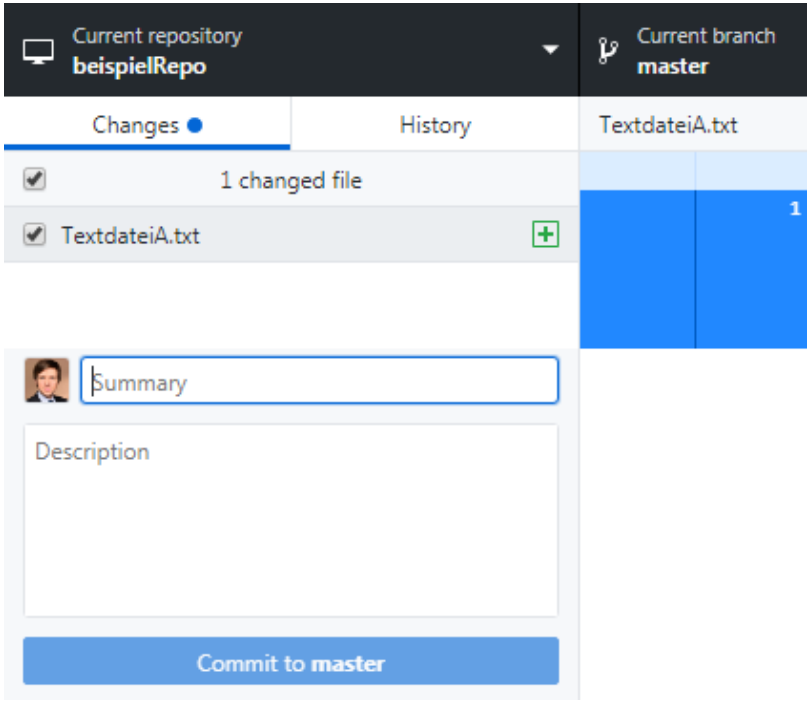
1. Neue Dateien und Änderungen müssen zur Staged-Ebene hinzugefügt werden

   ```
   git add -all
   ```

2. Von der Staged-Ebene werden die Daten im Repository gesichert

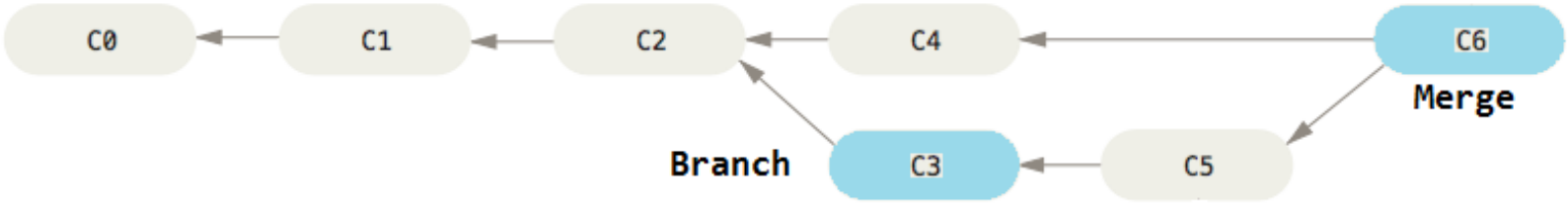
   ```
   git commit -m "Beschreibung
    des Commits"
   ```

GitHub Desktop



Zweige

Branch und Merge



Shell

C3 Neuen Zweig erzeugen

```
git checkout -b "dev"
```

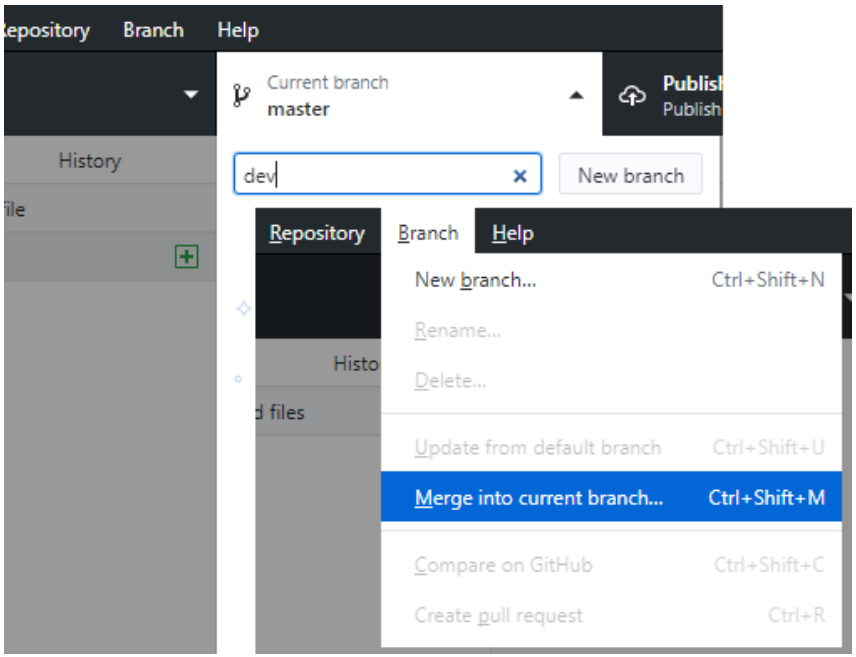
C6 In Hauptzweig wechseln

```
git checkout "master"
```

Hauptzweig mit Nebenzweig zusammenführen

```
git merge "dev"
```

GitHub Desktop



Was sind Konflikte?

Konflikte ...

- sind gleichzeitige Änderungen in gleichen Abschnitten einer Datei
  1. in unterschiedlichen Zweigen, oder
  2. von unterschiedlichen Nutzern im gleichen Zweig
- tauchen beim Committen sowie Zusammenführen von Zweigen auf
- müssen händisch aufgelöst werden

Kommunikation mit anderen Teammitgliedern erforderlich!

Beispiel Entstehung eines Konfliktes

`Readme.md` wird in zwei Zweigen in der selben Zeile bearbeitet.
→ Konflikt beim Merge.

master-Branch

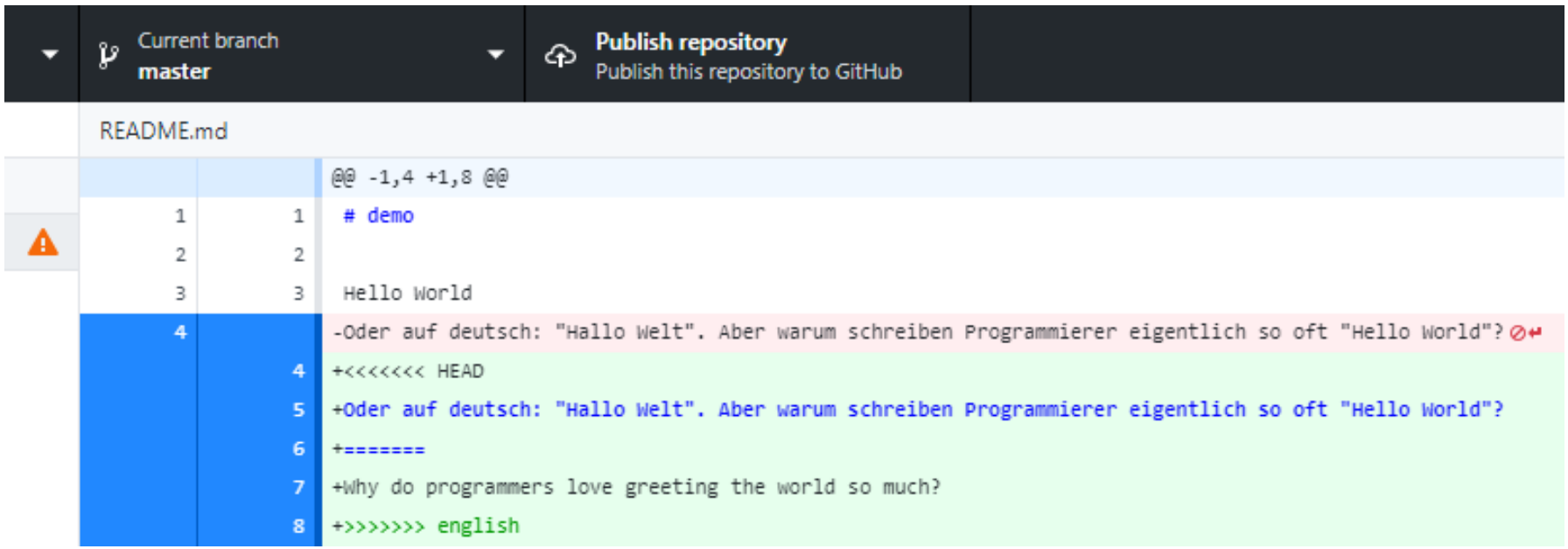
```
1 #demo
2
3 Hello World
4 Oder auf deutsch: "Hallo Welt". Aber warum schreiben
      Programmierer eigentlich so oft "Hello World"?
```

english-Branch

```
1 #demo
2
3 Hello World
4 Why do programmers love greeting the world so much?
```

Darstellung von Konflikten

`git` fügt an der Stelle des Konfliktes eine Sequenz entsprechend des untenstehenden Shemas ein.



Darstellung von Konflikten

`git` fügt an der Stelle des Konfliktes eine Sequenz entsprechend des untenstehenden Shemas ein.

Inhalt von Readme.md

```
1  # demo
2
3  Hello World
4  <<<<<<< HEAD
5  Oder auf deutsch: "Hallo Welt". Aber warum schreiben
       Programmierer eigentlich so oft "Hello World"?
6  =======
7  Why do programmers love greeting the world so much?
8  >>>>>>> english
```

Behebung von Konflikten

Die betroffene Datei muss manuell bearbeitet werden.

Neuer Inhalt von Readme.md

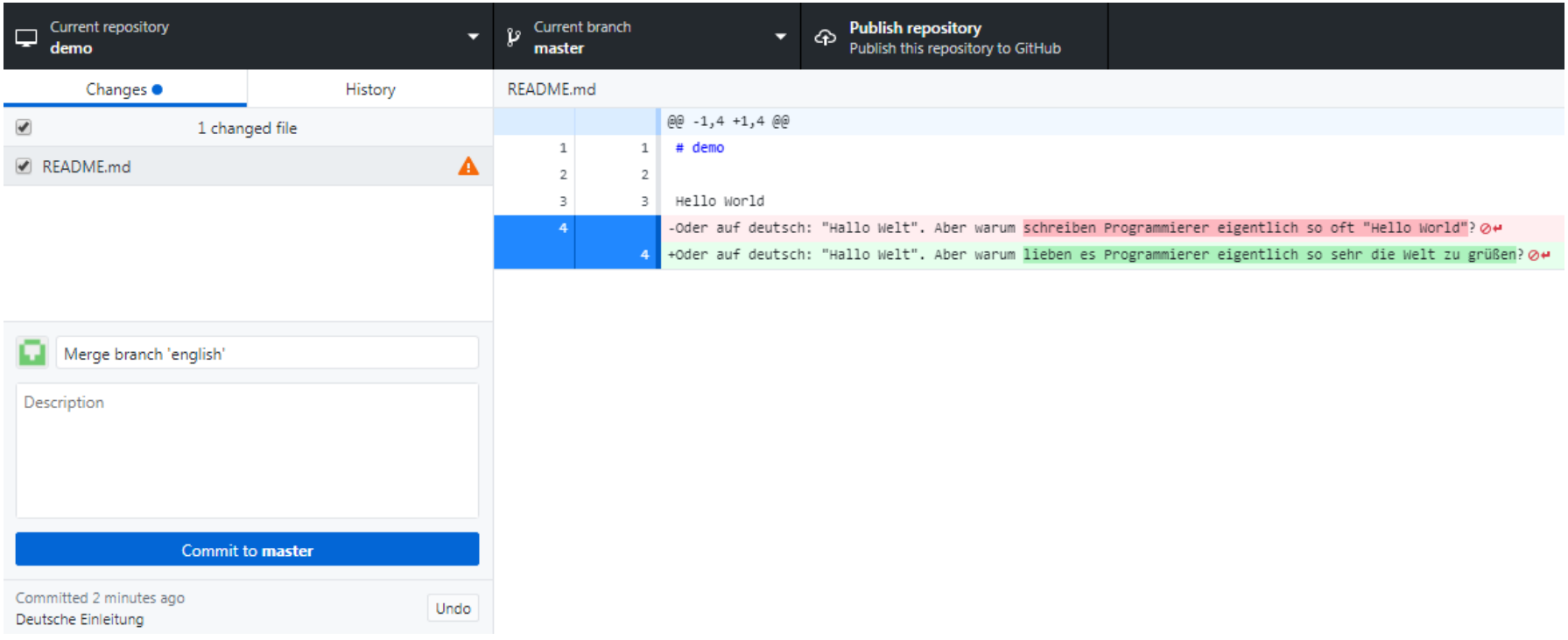
```
1 # demo
2
3 Hello World
4 Oder auf deutsch: "Hallo Welt". Aber warum lieben
      Programmierer es eigentlich so sehr, die Welt zu grüßen?
```

Hinweise

1. Die Konfliktmarkierungen müssen entfernt werden.
2. Es muss nicht zwangsläufig eine der beiden Konfliktversionen verwendet werden. Eine Kombination ist auch möglich (siehe oben).

Behebung von Konflikten

Nachdem alle Konflikte beseitigt wurden, muss die neue „zusammengeführte“ Version committed werden.



Gibt es Fragen oder Anmerkungen zu dem Thema
git-Grundlagen?

Abgehakt

Versionskontrolle mit GitHub

Als Teilnehmer soll ich am Ende dieser Übung...

- [x] Konflikte in git lösen können
- [ ] an bestehenden Projekten weiterarbeiten können
- [ ] das lokale Repository mit GitHub synchronisieren können
- [ ] das Projekt-Wiki erstellen und mit Inhalt füllen können

Teamarbeit
mit git

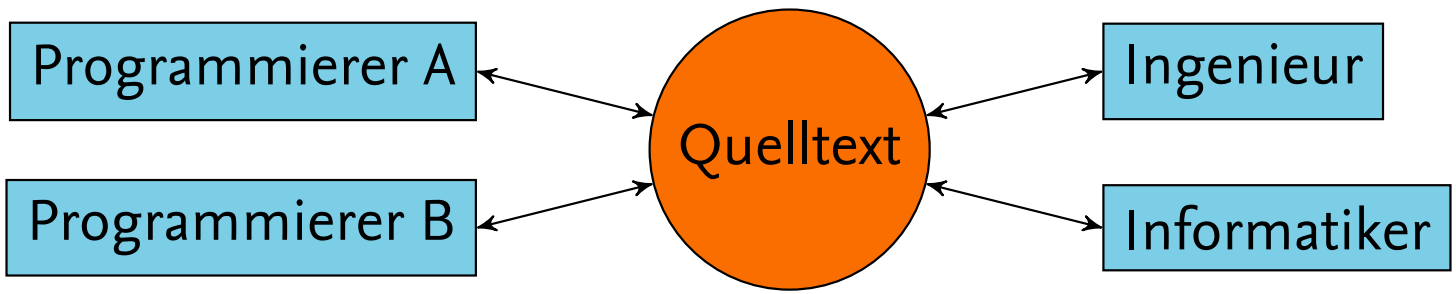
Teamarbeit

Repo-
Varianten

Softwareentwicklung im Team

Ausgangsszenario

- Implementierung einer Steuerungssoftware
- Softwareentwicklung im Team
  - Programmierer
  - Ingenieure
  - Informatiker
- Arbeit an gemeinsamer Quelltextbasis



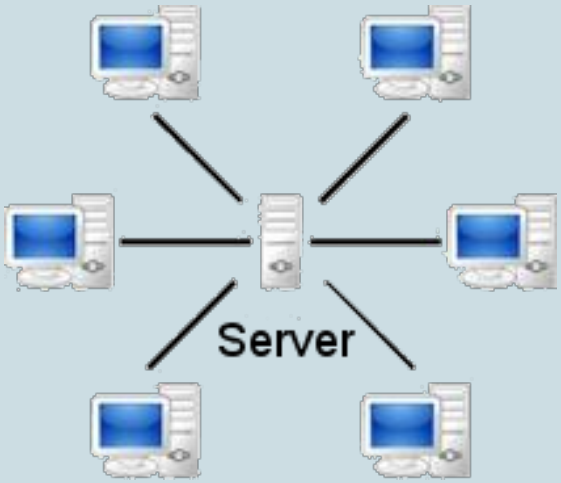
Varianten von Versionskontrollsystemen

Lokal

Revisionen werden im lokalen Dateisystem gespeichert

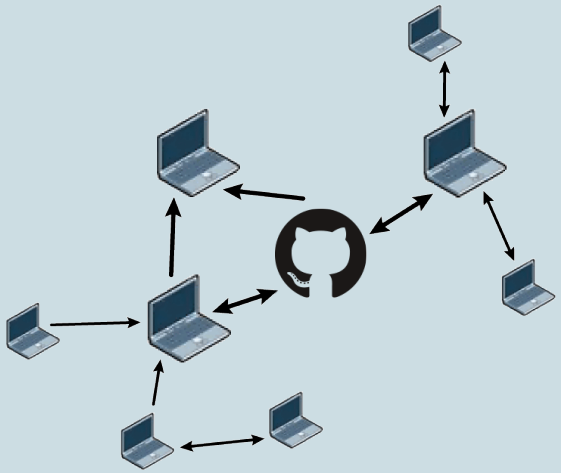
Zentral

Revisionen werden zentral auf einem Server verwaltet



Verteilt

Jede Instanz ist eine Kopie des gesamten Repositorys



Lokale Versionskontrollsysteme

- Einfachste Möglichkeit des Versionskontrollsystems
- Änderungen und Verwaltungsinformationen werden lokal gespeichert
  - Kein geregelter Austausch zwischen Entwicklern
  - In der Regel nur Verwaltung einzelner Dateien
  - Überblick über gemachte Änderungen dennoch möglich
  - Verwendung heutzutage z. B. für Konfigurationsdateien

Zentrale Versionskontrollsysteme: Funktion

- Zentraler Server, auf dem alle Revisionen gespeichert werden
- Authentifizierung möglich (z. B. über Benutzername und Passwort)
- Ein Benutzer hat auf seinem Rechner eine Arbeitskopie
  - Synchronisation mit Server $\Rightarrow$ Update der Arbeitskopie
  - Entwickler verändern ihre Arbeitskopie
  - Einchecken der Änderungen beim zentralen Server

Spätestens bei einem Commit muss der Entwickler den aktuellen Stand vom Server beziehen.

Verteilte Versionskontrollsysteme: Funktion

- Kein zentrales Repository erforderlich
  - Jeder Entwickler hat eine vollständige Kopie des Repositorys
  - Änderungen und Commits werden ins lokale Repository gepflegt
  - Einfaches Erzeugen und Mergen von Entwicklungszweigen
- Ohne zentrales Repo ist der Quelltextaustausch umständlich

  $\rightarrow$ Hauptrepository ermöglicht einfachen Austausch des Quellcodes
- Kein Single-Point-of-Failure

  $\rightarrow$ Höhere Datensicherheit

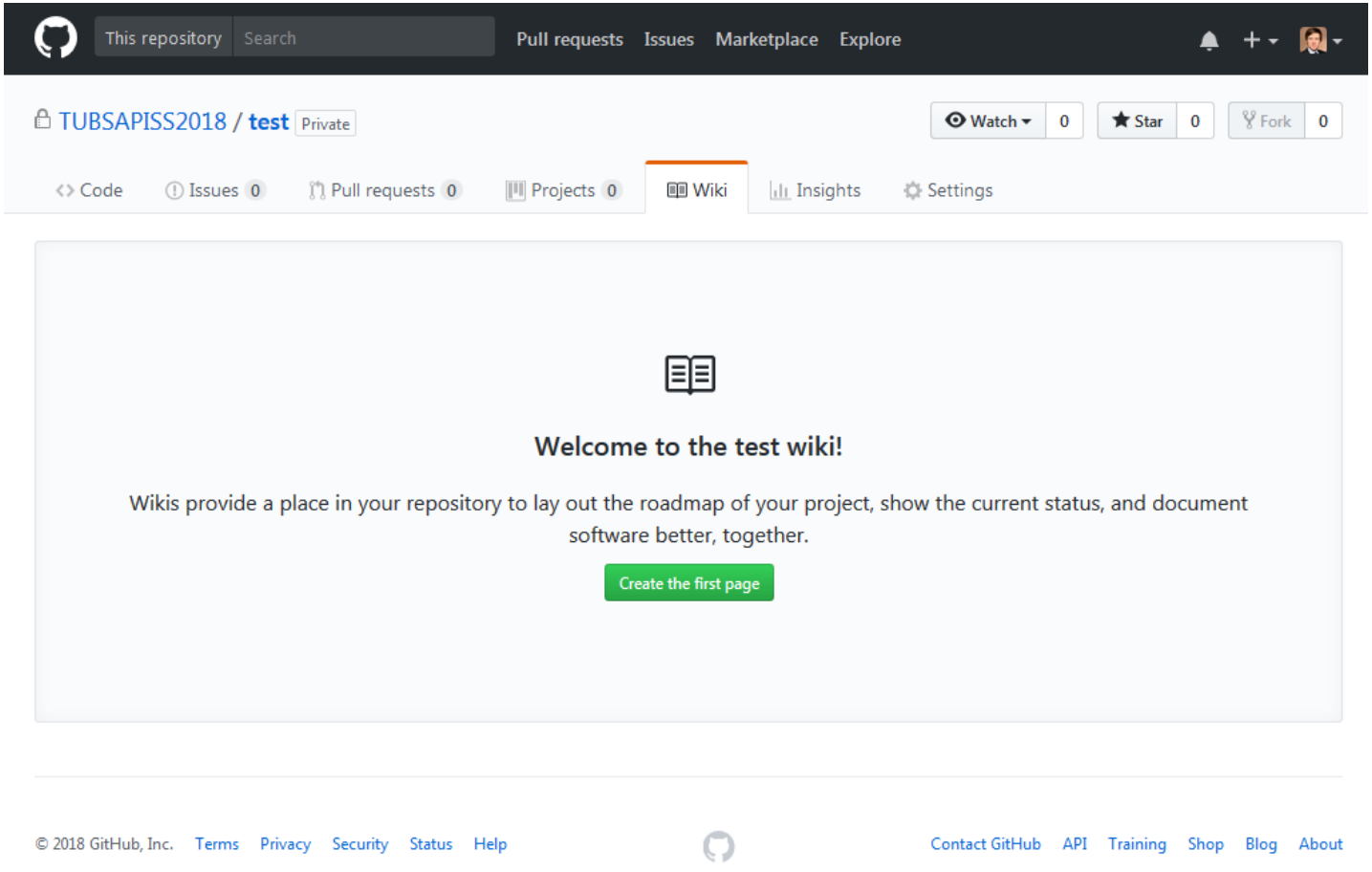
Verteilte Versionskontrollsysteme versuchen die Vorteile von lokaler und zentraler Versionskontrolle zu verbinden

Übersicht der Aufgaben

1. Das Projekt-Wiki initialisieren
2. Das Projekt-Wiki klonen
3. Das Template-Wiki klonen
4. Inhalt des Template-Wikis lokal in das Projekt-Wiki kopieren
5. Das Template-Wiki lokal löschen
6. Den eigenen Namen in die Datei `Home.md` eintragen
7. Den lokalen Stand auf den Server übertragen
8. Falls erforderlich:
   - 8.1 Den Stand vom Server herunterladen
   - 8.2 Konflikte auflösen
   - 8.3 goto Aufgabe 7

Projekt-Wiki initialisieren

Das Wiki kann primär online erstellt und bearbeitet werden.



Situation

Ich möchte an einem existierendem Projekt weiterarbeiten
oder das existierende Projekt verwenden.

git clone

Adresse

Die Adresse des Code-Repositorys auf GitHub kann mit einem Klick auf die „Clone or Download“-Schaltfläche eingesehen und kopiert werden.

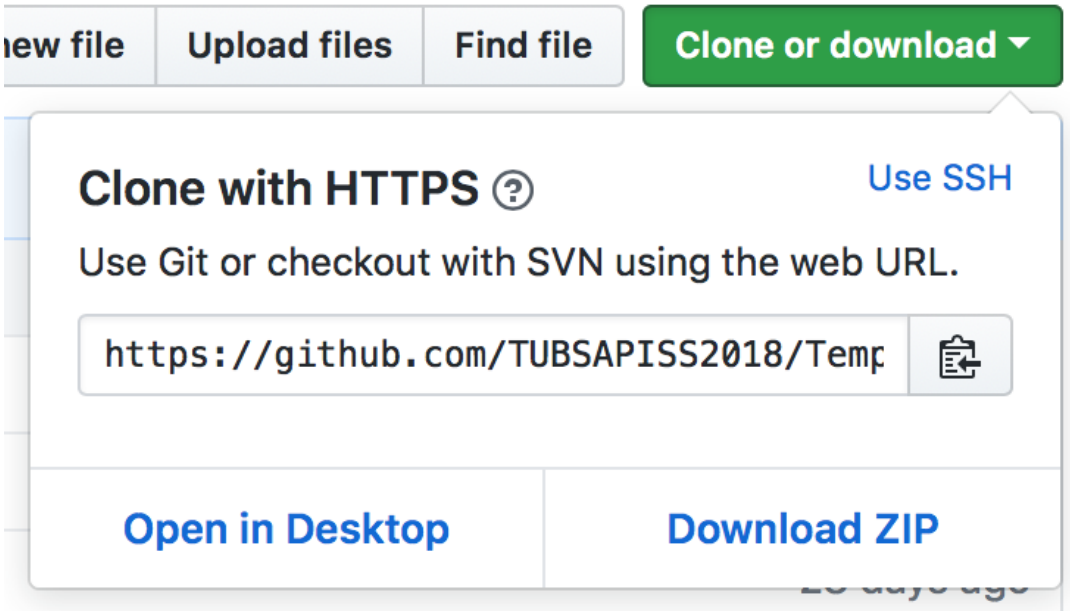


Abbildung 2: Adresse eines Code-Repositorys in GitHub

git clone

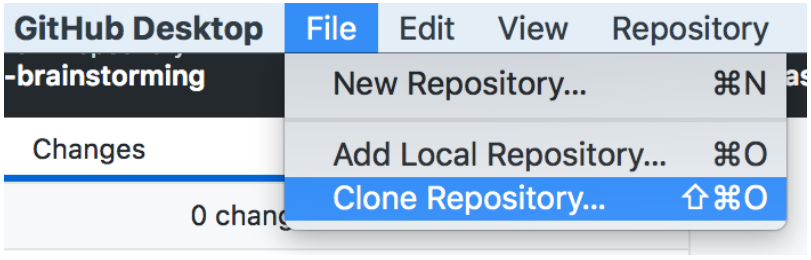
Shell

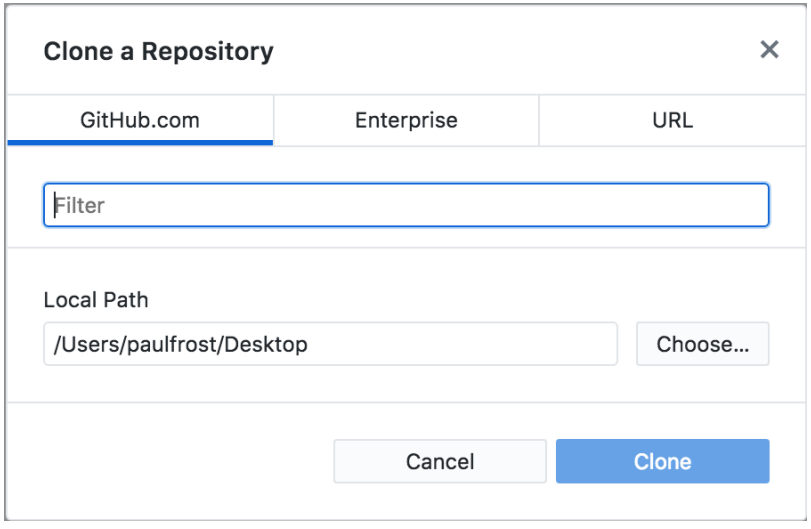
Über den Befehl `git clone` wird das gesamte Repository in ein neu erstelltes Verzeichnis geklont

Listing 1: Repository klonen

```
git clone "https://Adresse.git"
```

GitHub Desktop





git clone

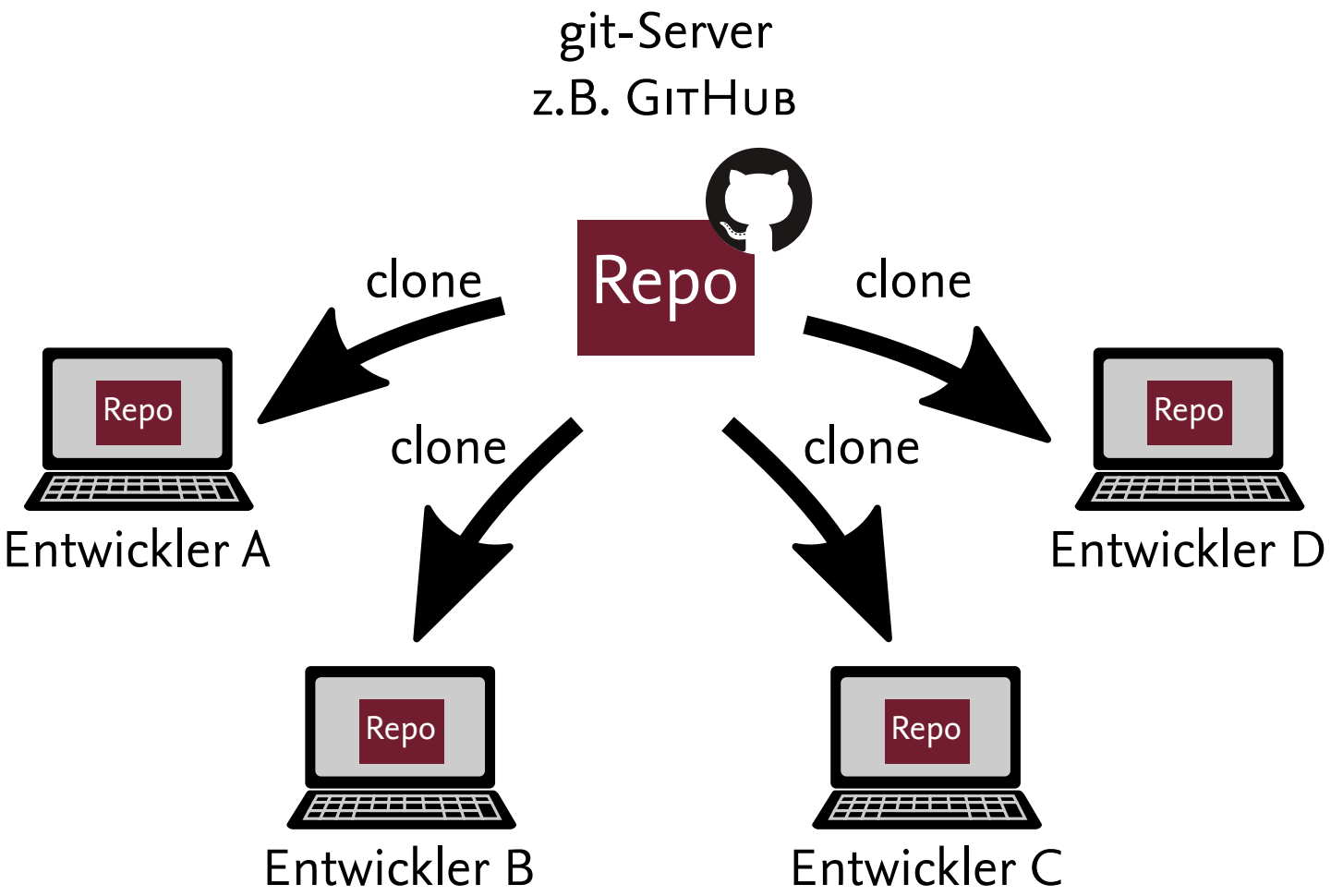


Abbildung 3: git clone eines GitHub-Repositorys

Aufgabe

1. Das Projekt-Wiki initialisieren
2. Das Projekt-Wiki klonen

   Adresse des Projektmappen-Wikis über GitHub herausfinden

   Befehl: `git clone`
3. Das Template-Wiki klonen

   `https://github.com/TUBSAPISS2018/Template.wiki.git`

Situation

Ich möchte den aktuellen Stand einsehen und überprüfen, aber noch nicht in den aktuell aktiven Zweig überführen.

git fetch

Über den Befehl `git fetch` wird der aktuelle Stand von dem Server heruntergeladen. Dabei bleibt das Arbeitsverzeichnis (engl. Working Directory) unverändert.

Listing 2: Repository synchronisieren

```
git fetch
```

Listing 3: In den Zweig wechseln

```
git checkout "Zweigname"
```

Ein anderer Zweig kann so überprüft und über `git merge` übernommen werden.

Situation

Ich möchte den aktuellen Stand herunterladen und direkt in meinen aktiven Zweig überführen.

git pull

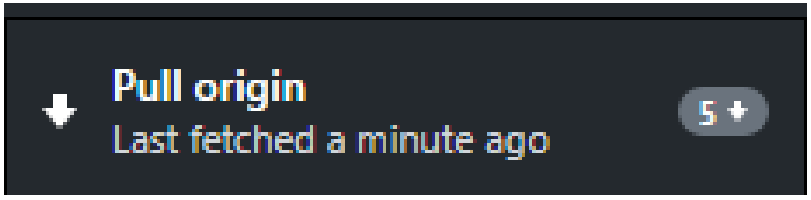
Der Befehl `git pull` fasst die Befehle `git fetch` und `git merge` zusammen.

Shell

Listing 4: Repository synchronisieren und Working Directory anpassen

```
git pull
```

GitHub Desktop



Änderungen müssen vor dem Ausführen von `git pull` über `git commit` gesichert werden

Situation

Ich möchte meine Commits an den Server übertragen.

git push

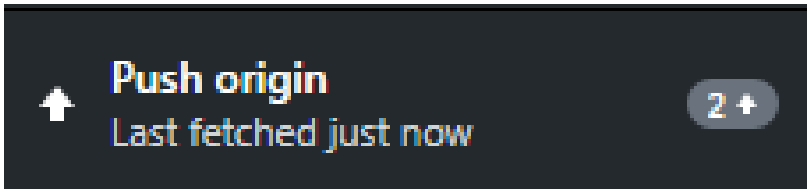
Über den Befehl `git push` können die Commits des aktiven lokalen Zweiges an den Server übertragen werden.

Shell

Listing 5: Commits hochladen

```
git push
```

GitHub Desktop



`git push` kann nur dann angewendet werden, wenn das lokake Repository auf dem aktuellen Stand ist.

git-Arbeitsfluss

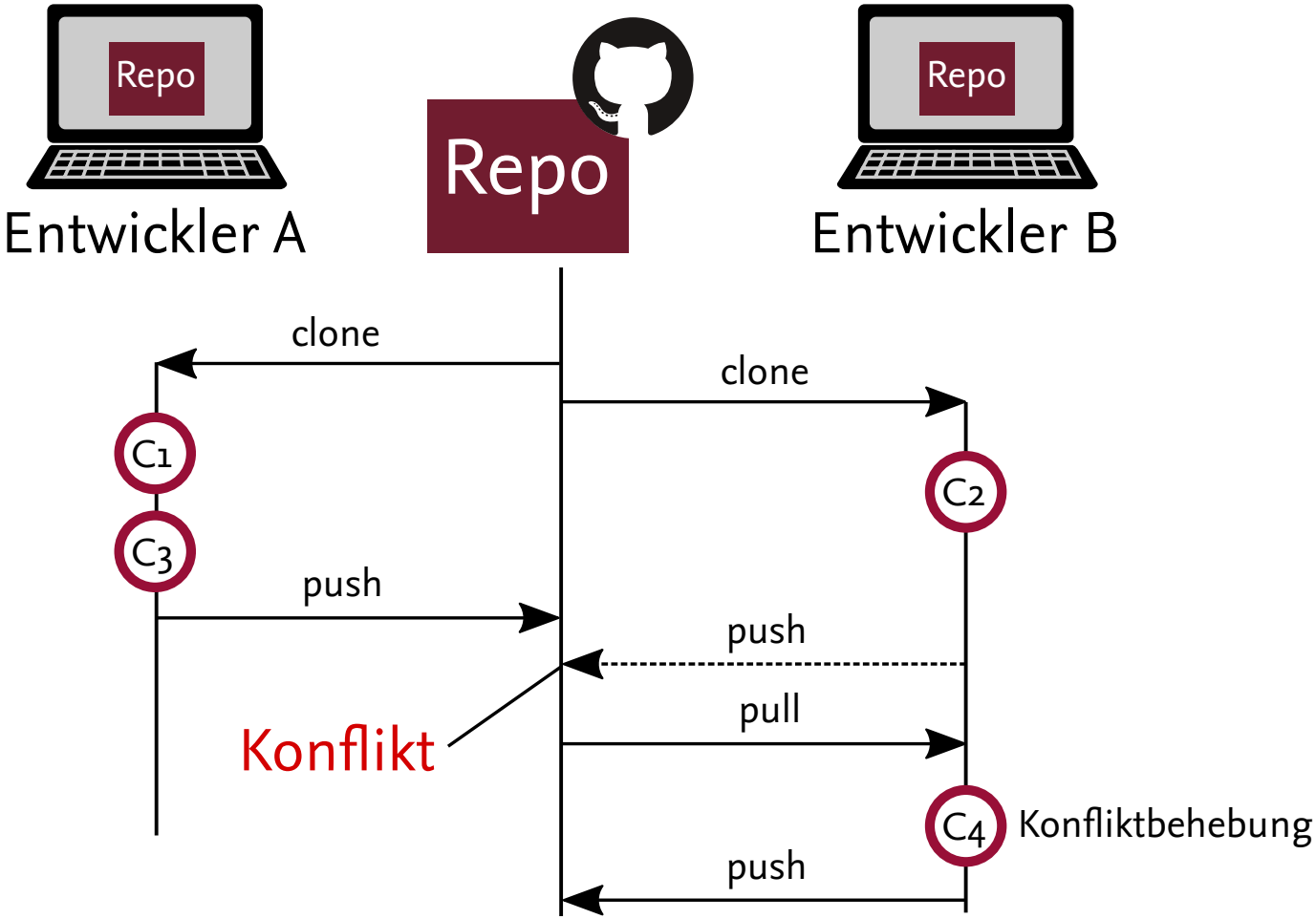


Abbildung 4: git-Arbeitsfluss

Gibt es Fragen oder Anmerkungen zu dem Thema
Teamarbeit mit git?

Aufgaben

4. Inhalt des Template-Wikis lokal in das Projekt-Wiki kopieren
5. Das Template-Wiki lokal löschen
6. Den eigenen Namen in die Datei `Home.md` eintragen
7. Den lokalen Stand auf den Server übertragen
   Befehl: **`git push`**

8. Falls erforderlich:
   - 8.1 Den Stand vom Server herunterladen
     Befehl: **`git pull`**
   - 8.2 Konflikte auflösen
   - 8.3 Den Stand vom Server herunterladen
     Befehl: **`git add`** und **`git commit`**
   - 8.4 goto Aufgabe 7

Abgehakt

Versionskontrolle mit GitHub

Als Teilnehmer soll ich am Ende dieser Übung...

- ☑ Konflikte in git lösen können
- ☑ an bestehenden Projekten weiterarbeiten können
- ☑ das lokale Repository mit GitHub synchronisieren können
- ☑ das Projekt-Wiki erstellen und mit Inhalt füllen können

Aufgaben

Projekt-Code-Repo

1. Klont das Code-Repository.
2. Optional: Mit der Programmierung beginnen.

Projekt-Wiki-Repo

1. Wie können Bilder in das Projekt-Wiki hochgeladen werden?
2. Fügt das Projektschema in das Projekt-Wiki hinzu.
3. Übertragt sämtliche Punkte für die Projektmappe in das Projekt-Wiki.

Literatur

[1] S. Chacon und B. Straub, *Pro Git*. 2014.

Cheat Sheets

Cheat Sheets geben eine gute Übericht über die verfügbaren git-Befehle.

- git und GitHub

  `https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf`

- Dynamisches Cheat Sheet

  `http://ndpsoftware.com/git-cheatsheet.html`

Sprintmeeting 15 Min

Jetzt besteht die Möglichkeit, das Sprintmeeting durchzuführen.

Protokolliert bitte

- die bearbeiteten Aufgaben der Vorwoche.
- die Zwischenstände der geplanten Aufgaben.
- die in der kommenden Woche zu bearbeitenden Aufgaben.

Ende

Vielen Dank für eure Aufmerksamkeit!