



## **Gemeinsame Nutzung von Git im Team**

Hochschule Bremen Fakultät 4

Softwaretechnik 1 WS 18/19

Bewertete Aufgabe 3

Autoren: Baanu Rajakumar, Fatma Donbay, Natia Gabrichidze

## Inhaltsverzeichnis

Einführung .....	3
1.1 Erstellung einer Repository auf dem grafischen Git-Client der Github .....	4
1.2 Hinzufügen der Team-Mitglieder .....	4
Bearbeiten im Team.....	5
2.1 Dateien in das Repository committen .....	5
2.2 Bearbeitungskonflikt bei gemeinsamer Bearbeitung einer Datei.....	5
3 Arbeiten mit Branches .....	5
3.1 Development Branch und Topic-Branch erzeugen.....	5
3.2 Dateien bearbeiten, committen und mergen .....	6
4 Integration-Manager Workflow .....	6
4.1 Umsetzung für den Repository und des Teams.....	6
5 Sicherungskopie erstellen .....	9
5.1 Anlegen einer lokalen Sicherungskopie des Repositories .....	9

## Einführung

### 1.1 Vorbereiten um die Aufgaben zu bearbeiten

Zuerst musste mithilfe der Anleitung von Frau Bondarenko der ssh-Key erstellt werden. Dafür wurde zunächst ein Verzeichnis angelegt namens: `.ssh` und anschließend über die Konsoleneingabe `ssh-keygen` eingegeben.

Dann wurden auf der Konsole die beiden generierten keys angezeigt und der public key wurde dann an Frau Bondarenko geschickt, damit sie diese im Server eintragen kann. Die beiden Keys wurden zusätzlich in dem erstellten Verzeichnis `.ssh` gespeichert.

```
brajakumar:~> ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/brajakumar/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in git.ssh.
Your public key has been saved in git.ssh.pub.
The key fingerprint is:
SHA256:sNTZRZiDTEHNHAs2sTz1MB1F+uH/5VH7KyrYpe+gwTM brajakumar@pc18
The key's randomart image is:
+---[RSA 2048]---+
  +OB=*++o
  +oBB*..
  o * o.o .
  . o . o .
  . S   o .
  .   . .o
  .Eo.o oo|
  .+=. . .|=
  . ++. . .|=
+---[SHA256]---+
brajakumar:~>
```

Abbildung 1: Erstellen des ssh keys

Im Anschluss wurde überlegt mit welchem Programm gearbeitet wird. Zunächst wurde an den Aufgaben gearbeitet über der Konsole jedoch wurde in der Gruppe dann entschieden sich für das Programm GitHub.

## 1.2 Erstellung einer Repository auf dem grafischen Git-Client der Github

Für die Erstellung einer Repository wurde überlegt mit welchem Programm gearbeitet wird. Zunächst wurde an den Aufgaben gearbeitet über der Konsole jedoch wurde in der Gruppe dann entschieden sich für das Programm GitHub.Dafür mussten alle Teammitglieder zunächst einen Account über Github erstellen. Nach der Erstellung einer Repository durch ein Teammitglied wurden die restlichen Mitglieder über ' *Settings > Collaborators > Eingabe der Adresse des jeweiligen Teammitglieds > Add collaborator* ' hinzugefügt. Nach der Bestätigung der Mail sind diese Mitglieder in der Repository bestätigt und können mit an darin arbeiten. (siehe Abbildung 2)

## 1.3 Hinzufügen der Team-Mitglieder

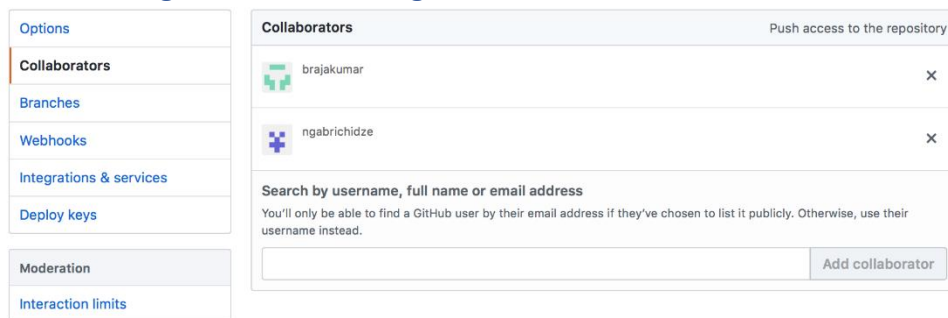


Abbildung 2 : Hinzufügen der Teammitglieder

## Bearbeiten im Team

### 2.1 Dateien in das Repository committen

Als nächstes wurde zunächst einmal von jedem Gruppenmitglied jeweils zwei Dokumente erstellt. Bei dem einen Dokument handelt es sich um eine normale Text Datei und bei dem anderen wurde ein PDF Dokument erstellt.

Dadurch das wir uns dazu entschieden haben mit dem grafischen Git Programm zu arbeiten, konnte hier die Datei einfach in per drag and drop Prinzip zum Repository hinzugefügt werden.(siehe Abbildung 3)








 README.md	Initial commit	6 hours ago
 brajakumar.pdf	Add files via upload	5 hours ago
 brajakumar.txt	Add files via upload	5 hours ago
 fdonbay.pdf	Add files via upload	5 hours ago
 fdonbay.txt	Add files via upload	6 hours ago
 gabrichidze.pdf	Add files via upload	5 hours ago
 gabrichidze.txt	Add files via upload	5 hours ago

Abbildung 3: Ausschnitt vom Repository mit allen Dokumenten

### 2.2 Bearbeitungskonflikt bei gemeinsamer Bearbeitung einer Datei

Als nächstes wurde wie in der Aufgabe beschrieben eine Datei mit dem Namen team.txt erzeugt. Anschließend haben alle Gruppenmitglieder die neu erstellte Datei geöffnet, überarbeitet und gleichzeitig wurde dann versucht diese Änderungen zu *committen*. Bei diesem Versuch meldete sich GitHub schon mit der ersten Fehlermeldung. (siehe Abbildung 4)

File could not be edited	×
ngabrichidze has committed since you started editing. <a href="#">See what changed</a>	×

Abbildung 4: Die beiden ersten Fehleranzeigen

Nun kann man wie bei der Abbildung zwei zu sehen ist auf „*See what changed*“ klicken und nachvollziehen wie das Dokument verändert wurde. Dabei wird Zeile für Zeile angezeigt was neu hinzugefügt bzw. gelöscht wurde. Git begutachtet also die Veränderungen und zeigt diese an.

Um solche Konflikte zu vermeiden, kann man zum einen untereinander kommunizieren, wann wer im Repository arbeitet. Oder die Gruppenmitglieder arbeiten lokal an den eigenen Rechnern und laden diese später hoch.

## 3 Arbeiten mit Branches

### 3.1 Development Branch und Topic-Branch erzeugen

Neben der master-Branch wurde zusätzlich eine Branch development und eine Topic-Branch mit der Bezeichnung topic1 erzeugt. In beiden neu erzeugten Branches sind alle Dateien von der master-Branch übernommen worden.

### 3.2 Dateien bearbeiten, committen und mergen

Die Aufgabe bestand darin die Dateien unter der topic1-Branch zu bearbeiten und anschließend zu committen. Dafür wurde Beispielsweise die Datei *fdonbay.txt* überarbeitet und committet. Die Änderungen die unter der Branch topic1 vorgenommen wurden sind in derselben Datei unter der Branch development nicht zu sehen. (siehe Abbildung 5,6)

Durch das Mergen von topic1 in development werden die Überarbeitungen in der Branch development übernommen. (siehe Abbildung 7,8)

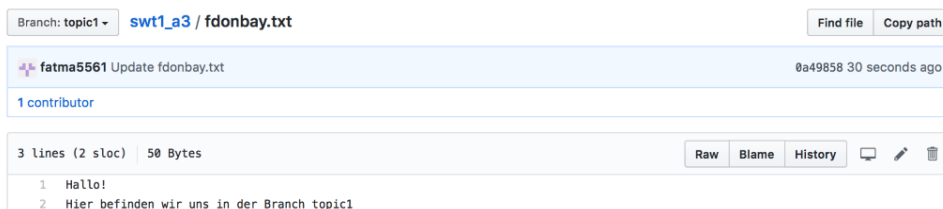


Abbildung 5: Bearbeitung der Datei *fdonbay.txt* unter der Branch *topic1*

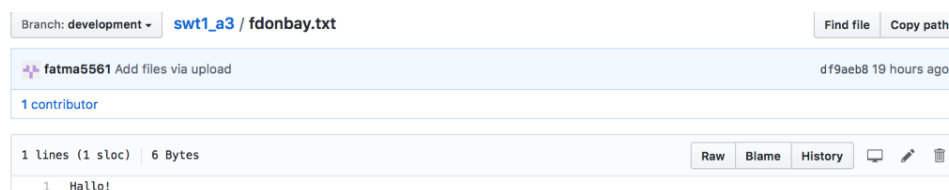


Abbildung 6: Mergen von *topic1* in *development*

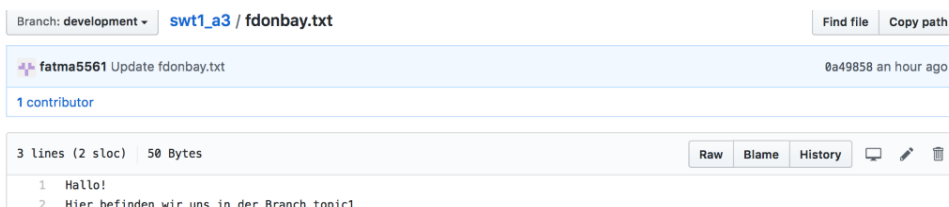


Abbildung 7: Ausgabe der Datei *fdonbay.txt* unter der Branch *development* nach dem mergen

## 4 Integration-Manager Workflow

Eine Integration-Manager Workflow ermöglicht es, eine Vielzahl von externen Repositories zu betreiben, indem jeder Developer z.B. vom master - Branch eine Fork-Kopie erstellt und so lokal an dem Projekt weiterarbeiten kann. Diese Veränderungen werden im blessed repository zunächst nicht übernommen.

### 4.1 Umsetzung für den Repository und des Teams

Für diese Aufgabe wurde mit dem vorhandenen Repository gearbeitet.

Zunächst wurde mit Fork eine externe Repository von den Developer Mitgliedern erstellt. (siehe Abbildung 8,9)

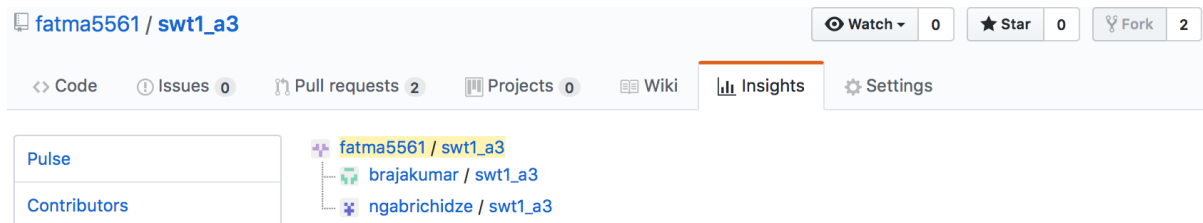


Abbildung 8: Anzeige das Forks erstellt worden sind

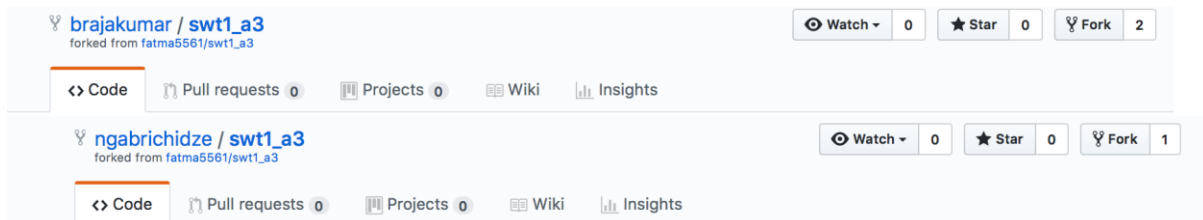


Abbildung 9: Externe Repositories durch Fork

Anschließend haben die Developer lokal mit den neu erstellten externen Repositories gearbeitet und so Veränderungen vorgenommen. Danach wurden durch die Developer Dateien im externen Repository verändert bzw. hinzugefügt. Diese Änderungen wurden vom Developer *committed* und anschließend wurde durch *new pull request* eine Anfrage an den Integration-Manager gestellt, um die Änderungen im blessed- repository zu übernehmen. (siehe Abb. 10)

Der Integrations-Manager kann sich jetzt die ganzen *pull requests* ansehen und entscheiden ob die Änderungen von Relevant für das Projekt sind. Ist dies der Fall kann der Integrations Manager die Anfrage annehmen und die Änderungen im blessed- Repository mit dem Befehl *merge pull request* übernehmen. (siehe Abb. 11,12)

## Comparing changes

Choose two branches to see what's changed or to start a new pull request. If you need to, you can also [compare across forks](#).

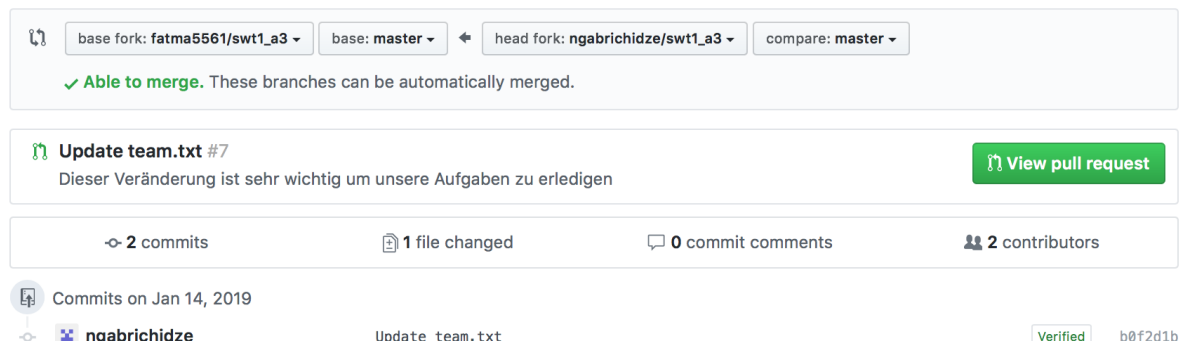






Abbildung 10: Anfrage vom Developer and den Integrations Manager

## Update team.txt #6

 **Open** ngabrichidze wants to merge 1 commit into `fatma5561:master` from `ngabrichidze:master`


 Conversation **0**  Commits **1**  Checks **0**  Files changed **1**




ngabrichidze commented 2 minutes ago


Collaborator + 👤 ...


No description provided.

 Update team.txt Verified 6bffffdf

Add more commits by pushing to the **master** branch on **ngabrichidze/swt1\_a3**.




 **Continuous integration has not been set up**  
Several apps are available to automatically catch bugs and enforce style.





 **This branch has no conflicts with the base branch**  
Merging can be performed automatically.


**Merge pull request** You can also [open this in GitHub Desktop](#) or view [command line instructions](#).

Abbildung 11: Admins Mergen vom commit des Developer

## Update team.txt #6

 **Merged** fatma5561 merged 1 commit into `fatma5561:master` from `ngabrichidze:master` just now


 Conversation **0**  Commits **1**  Checks **0**  Files changed **1**



ngabrichidze commented 3 minutes ago

Collaborator + 👤 ...

No description provided.

 Update team.txt Verified 6bffffdf


 fatma5561 merged commit **9fc8d4a** into `fatma5561:master` just now Revert

Abbildung 12: Änderung vom dem Developer wurde vom Integration-Manager Workflow/Admin übernommen



## 5 Sicherungskopie erstellen

### 5.1 Anlegen einer lokalen Sicherungskopie des Repositories

Da gibt es einige Möglichkeiten eine Sicherheitskopie des Repositories zu erstellen.

Die erste Möglichkeit wäre es z.B. den Repository Ordner auf ein USB-Stick zu ziehen. Dafür gibt es auch in GitHub eine Funktion womit man das ganze Repository in einem Zip-Ordner runterladen kann. Diese Zip-Datei kann dann auf einem USB-Stick oder einer externen Festplatte gesichert werden.

Des Weiteren kann man auch ein ganzes Repository runterladen und lokal zur Sicherung z.B. auf GitHub behalten. Runterladen kann man z.B. mit *git clone*.

Wenn man zusätzlich eine Binärdatei als Sicherung erstellen will, kann man dies auch mit dem Befehl *git bundle create<name of repository>-- all* erreichen.

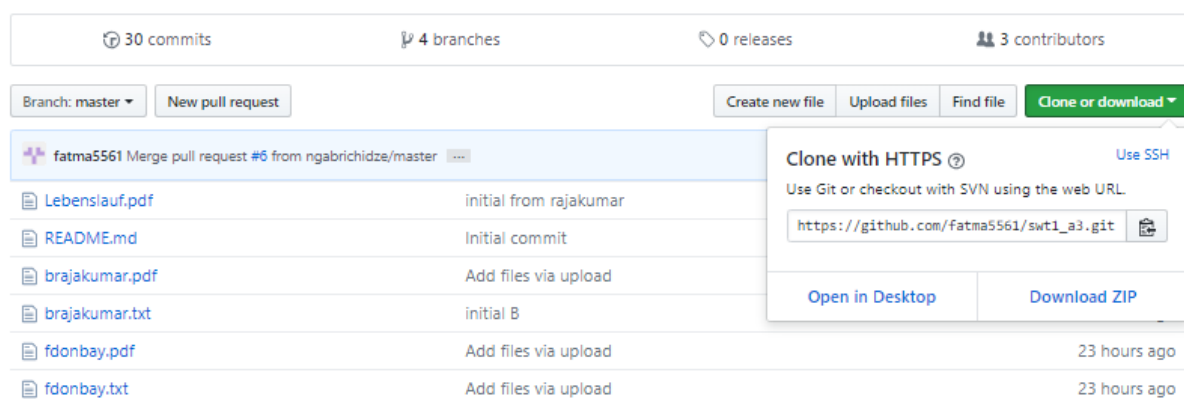


Abbildung 13: Fenster zum Runterladen von Repository