Protokoll

G 5a.3.2: Mit git auf der Konsole arbeiten

Andreas Sünder 3BHIT

2021

Inhalt

[1. Erstelle dir einen eigenen Ordner mit dem Namen gitLoginname. 2](#_Toc88206028)

[2. Erstelle in diesem Ordner ein neues leeres git-Repository 2](#_Toc88206029)

[3. Erstelle in diesem Ordner eine Datei mit dem Namen Nachname.txt und schreibe den Text Nachname gefolgt von der Ziffer 1 hinein. 2](#_Toc88206030)

[4. Überprüfe den Status von git, wobei die in Schritt 3 erstellte Datei noch untracked sein soll. 2](#_Toc88206031)

[5. Füge diese Datei zum Staging-Bereich hinzu und überprüfe den Status von git. 2](#_Toc88206032)

[6. Ändere die Ziffer in deiner Datei auf den Wert 2, speichere und überprüfe den Status von git. 3](#_Toc88206033)

[7. Füge die geänderte Datei zum Staging-Bereich hinzu und überprüfe den Status von git. 3](#_Toc88206034)

[8. Übertrage alle Dateien im Staging-Bereich als neue Version in das Repository und zeige die Log-Einträge deiner Versionen im Repository an 4](#_Toc88206035)

[9. Füge nun die Datei *Jahrgang*.**info** zu deinem Verzeichnis hinzu und schreibe deinen Jahrgang hinein. Überprüfe den Status von git. 4](#_Toc88206036)

[10. Schließe alle .info-Dateien von der Versionierung aus und überprüfe danach den Status von git 4](#_Toc88206037)

[11. Füge alle Änderungen als neue Version zu deinem Repository hinzu und zeige die Log-Einträge deiner Versionen im Repository an. 5](#_Toc88206038)

[12. Erstelle einen neuen leeren Unterordner mit dem Namen *Vorname* und überprüfe den Status von git. 5](#_Toc88206039)

[13. Füge in den Unterordner eine Datei mit dem Namen *Vorname*.txt ein, die deinen Vornamen als Text enthält. Ändere in deinem Basis-Ordner die Datei *Nachname*.txt so dass die Ziffer den Wert 3 hat, speichere und überprüfe den Status von git 5](#_Toc88206040)

[14. Füge alle Dateien mit \* zum Staging-Bereich hinzu und überprüfe den Status von git. 6](#_Toc88206041)

[15. Erstelle eine neue Version und zeige deinen Versionsverlauf (Log-Einträge) an. 6](#_Toc88206042)

[16. Erstelle auf Github ein neues leeres Repository unter eurem Github Account 7](#_Toc88206043)

[17. Füge dieses Github Repository bei deinem lokalen Repository als Remote-Repository unter dem origin hinzu 7](#_Toc88206044)

[18. Übertrage die Versionen von deinem lokalen Repository auf dein Github-Repository und überprüfe dann dort, ob deine Dateien vorhanden sind und ob du auch alle Versionen siehst. 8](#_Toc88206045)

# Erstelle dir einen eigenen Ordner mit dem Namen gitLoginname.

Dies kann mit dem Befehl mkdir gitAsuender erreicht werden.



# Erstelle in diesem Ordner ein neues leeres git-Repository

Der Befehl git init erstellt in dem aktuellen Verzeichnis ein neues lokales Repository und fügt den Unterordner .git hinzu.



# Erstelle in diesem Ordner eine Datei mit dem Namen Nachname.txt und schreibe den Text Nachname gefolgt von der Ziffer 1 hinein.

Unter Linux kann man den Befehl touch verwenden, um eine Datei zu erstellen. Bearbeitet wurde diese mit nano.

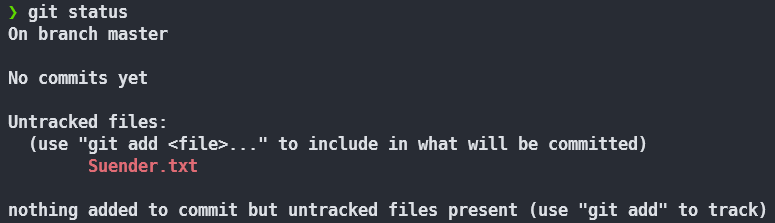




# Überprüfe den Status von git, wobei die in Schritt 3 erstellte Datei noch untracked sein soll.

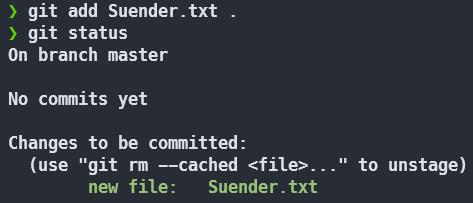
Wie hier zu sehen ist, werden erstelle Dateien nicht automatisch zur Versionskontrolle hinzugefügt, weshalb sie unter git status rot angezeigt werden.





# Füge diese Datei zum Staging-Bereich hinzu und überprüfe den Status von git.

Dies geschieht mit dem Befehl git add <file>. git status zeigt jetzt den gestagten Bereich an inkl. der hinzugefügten Datei.



# Ändere die Ziffer in deiner Datei auf den Wert 2, speichere und überprüfe den Status von git.



# Füge die geänderte Datei zum Staging-Bereich hinzu und überprüfe den Status von git.

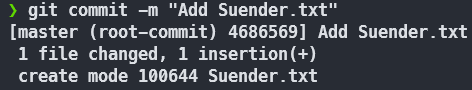
Die Datei Suender.txt ist zwar schon hinzugefügt, jedoch noch nicht commitet. Würde ich dies jetzt tun, so wäre die neue Version der Datei nicht dabei, sie scheint unter git status rot. Nach dem Aufruf von git add Suender.txt ist dies behoben.

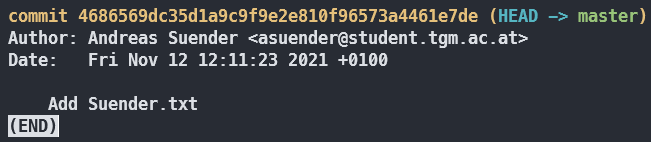
Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

# Übertrage alle Dateien im Staging-Bereich als neue Version in das Repository und zeige die Log-Einträge deiner Versionen im Repository an

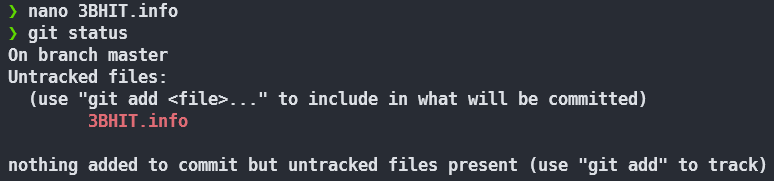
In der Kommandozeile commitet man alle Changes im Stage-Bereich mittels git commit -m “<message“>. Lässt man die letzten zwei Argumente aus, so öffnet sich der Standard-Editor von git (etwa nano, vscode etc.), in welchem man die Commit-Message einfügen muss.





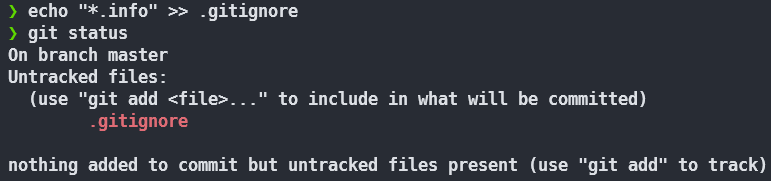
# Füge nun die Datei *Jahrgang*.**info** zu deinem Verzeichnis hinzu und schreibe deinen Jahrgang hinein. Überprüfe den Status von git.

Die Datei erscheint rot, nachdem sie noch nicht zur Versionskontrolle hinzugefügt worden ist.

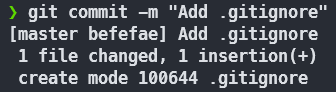


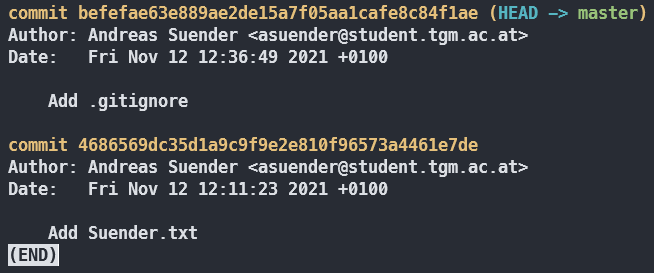
# Schließe alle .info-Dateien von der Versionierung aus und überprüfe danach den Status von git

Dies kann man mit einer .gitignore -Datei machen. \*.info bedeutet, dass alle Dateien von der Versionskontrolle ausgeschlossen werden, welche die Endung .info besitzen.



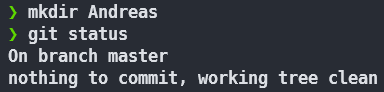
# Füge alle Änderungen als neue Version zu deinem Repository hinzu und zeige die Log-Einträge deiner Versionen im Repository an.





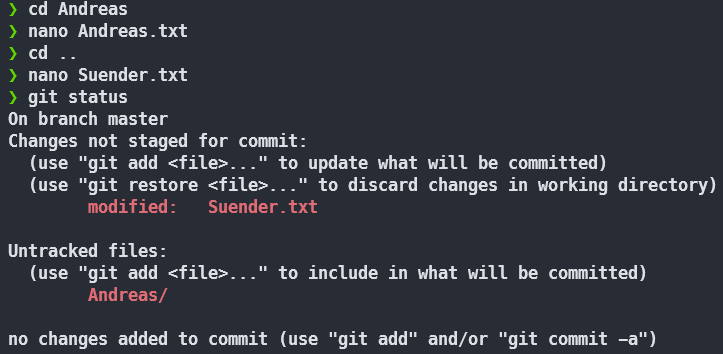
# Erstelle einen neuen leeren Unterordner mit dem Namen *Vorname* und überprüfe den Status von git.

Wie man hier schön sieht, werden leere Ordner von git nicht berücksichtigt.

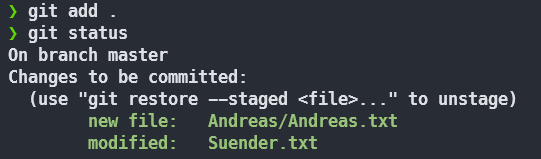


# Füge in den Unterordner eine Datei mit dem Namen *Vorname*.txt ein, die deinen Vornamen als Text enthält. Ändere in deinem Basis-Ordner die Datei *Nachname*.txt so dass die Ziffer den Wert 3 hat, speichere und überprüfe den Status von git

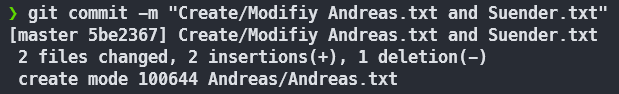
Nachdem ich eine Datei im Andreas/ Ordner erstellt hatte, schien dieser auch schon rot in git status auf. Des Weiteren habe ich die Suender.txt wie vorgegeben verändert.

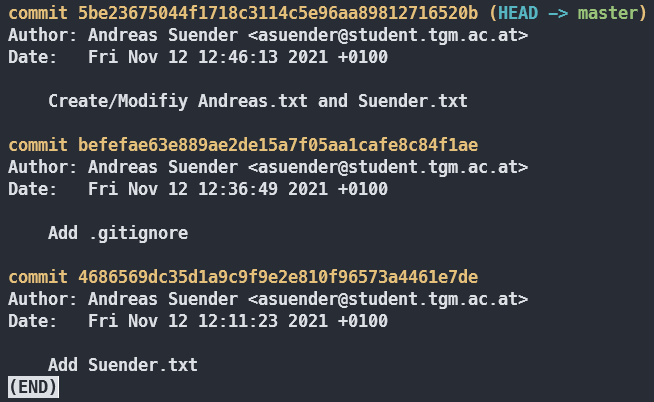


# Füge alle Dateien mit \* zum Staging-Bereich hinzu und überprüfe den Status von git.

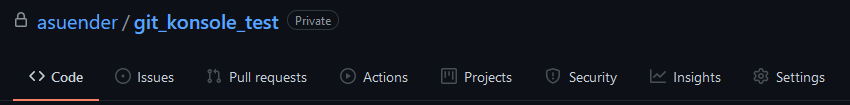


# Erstelle eine neue Version und zeige deinen Versionsverlauf (Log-Einträge) an.



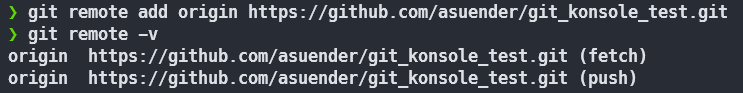


# Erstelle auf Github ein neues leeres Repository unter eurem Github Account



# Füge dieses Github Repository bei deinem lokalen Repository als Remote-Repository unter dem origin hinzu

Mit dem Befehl git remote add <remote> <url> kann man ein Remote-Repository mit dem Namen <remote> hinzufügen. In git remote -v scheint dieses dann sowohl fürs Fetchen als auch Pushen auf.



# Übertrage die Versionen von deinem lokalen Repository auf dein Github-Repository und überprüfe dann dort, ob deine Dateien vorhanden sind und ob du auch alle Versionen siehst.

Ein Bild, das Text enthält.

Automatisch generierte Beschreibung

Ein Bild, das Text, Monitor, Screenshot, drinnen enthält.

Automatisch generierte Beschreibung

Es ist alles da 😊!