Git mit kommandozeile (Allgemeine Verwendung von git)

Teil 1: Einrichtung von git

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| 1. Bei der ersten Verwendung von Git muss zunächst ein Benutzer mit Namen und E-Mail eingetragen werden. Falls ein Benutzer in mehreren Repositorys wiederverwendet werden soll, kann ein globaler Benutzer per --global eingetragen werden. Der Kommandozeilenbefehl lautet:   *git config user.name <username>*  *bzw.*  *git config user.email <email>* |  |

Teil 2: Repository erstellen

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| 1. In das Verzeichnis navigieren, in dem das Repository angelegt werden soll. |  |
| 1. Git Repository initialisieren:   git init |  |
| 1. Das Repository ist einsatzbereit! |  |

Teil 3: Staging and Commiting

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| 1. Sobald Sie mit den Änderungen in Ihrem Arbeitsverzeichniss zufrieden sind können Sie eine Liste aller geänderten Dateien anzeigen lassen:   git status |  |
| 1. Nachdem eine neue Datei erstellt wurde ist sie zunächst „untracked“. Damit sie von Git verwaltet werden kann muss sie mit *git add <dateiname>* mit aufgenommen werden. („Staging“). 2. Nun zeigt *git status* einen neuen Zustand: „zum Commit vorgemerkte Änderungen“. Alle Änderungen der hier gelisteten Dateien werden beim nächsten Commit abgespeichert. |  |
| 1. Ein Commit wird mit dem Befehl *git commit -m <message>* durchgeführt. Dadurch werden die Änderungen der „staged“ Dateien abgespeichert. |  |
| 1. Falls alle Änderungen gestaged wurden, zeigt *git status* nach dem commit an, dass das repo clean ist und keine Änderungen vorliegen |  |
| 1. Dieses Vorgehen wird für jeden Commit wiederholt. |  |

Teil 4: Arbeiten mit branches

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| 1. Einen neuen Branch erstellen:   git branch <branchname> |  |
| 1. In den neuen Branch wechseln:   git checkout <branchname> |  |
| 1. Neue Änderungen machen und commiten |  |
| 1. In den master Branch zurückwechseln |  |
| 1. Den neuen Branch in den aktuellen mergen:   git merge <branchname> |  |
| 1. Alle Änderung, die im Branch javaHelloWorld gemacht wurden, sind jetzt auch in master. |  |

Teil 5: Github als Remote für ein neues projekt

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| 1. Anmeldung auf <https://github.com/login>   **Hinweis**:  Falls benötigt oder erwünscht ist die Erstellung eines neuen Accounts unter dem Link „Create an account“ möglich. Standard Accounts auf GitHub sind umsonst, können aber nur öffentliche Repositorys erstellen |  |
| 1. Ein neues Repository anlegen |  |
| 1. Den Namen des Repositorys eintragen und „Create Repository“ klicken. |  |
| 1. Ein neues Projekt anlegen und ein git Repository mit git init initialisieren. In diesem Projekt die Readme anlegen:   echo "# RemoteTutorial" >> README.md |  |
| 1. Die Readme commiten |  |
| 1. Die URL des GitHub Repository kopieren |  |
| 1. Das GitHub Repository als Remote hinzufügen:     *git remote add origin <URL>* |  |
| 1. Den Status des Projektes in das GitHub Repository hochladen:   git push -u origin master  Wenn angefragt Benutzername und Passwort angeben. |  |
| 1. Auf GitHub ist nun der aktuelle Status zu sehen |  |

Teil 6: Ein bestehendes Repository clonen, Remote branches auschecken und merge-Konflikte lösen

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
| 1. Klonen des Tutorial Repository:   git clone https://github.com/THI-Tutorial-User/RemoteTutorial.git |  |
| 1. In das Verzeichnis wechseln und ein erweitertes Log betrachten:   ﻿  git log --graph --all –decorate |  |
| 1. Erstellen eines lokalen Branches, der dem Remote Branch *origin/tutorialBranch* folgt und den neuen Branch auschecken:   git checkout -b tutorialBranch origin/tutorialBranch |  |
| 1. Die Datei example.txt mit neuem, beliebigen Text füllen und die Änderungen commiten |  |
| 1. Zurück in den Branch master wechseln |  |
| 1. Die Änderungen des Branches *tutorialBranch* in *master* zusammenführen:   git merge tutorialBranch  Es entsteht ein merge-Konflikt. |  |
| 1. Die Datei *example.txt* mit einem Texteditor oder merge-Tool so bearbeiten, dass alle Änderungen behalten werden. |  |
| 1. Die gefixte Datei erneut stagen und commiten. In dem sich öffnenden Texteditor die Commit Nachricht anpassen und beschreiben, wie der merge-Konflikt gelöst wurde. |  |

Teil 2: xxx

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Teil 2: xxx

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

Teil 3: xxx

|  |  |
| --- | --- |
| **Explanation** | **Screenshot** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |