A1 Git: Repository erzeugen

1. Lassen Sie sich die installierte Version von Git ausgeben.
2. Konfigurieren Sie in Git Ihren Benutzernamen und Ihre E-Mail-Adresse. Überprüfen Sie, ob die Konfiguration richtig ist.
3. Erzeugen Sie auf einem Git-Server Ihrer Wahl, zum Beispiel auf dem GitLab-Server der FH Aachen, ein leeres Remote-Repository.
4. Klonen Sie das neue Repository in ein neues Arbeitsverzeichnis auf Ihrem Rechner.
5. Erstellen Sie in dem neuen Arbeitsverzeichnis eine Hello-World-Anwendung in der Programmiersprache Ihrer Wahl, z.B. Java, C# oder Python (die Sprache sollte allerdings auf Textdateien mit Quellcode basieren).
6. Lassen Sie sich den aktuellen Status ausgeben und übertragen Sie die Änderungen in Ihr lokales Repository.
7. Bringen Sie das Remote-Repository auf den neuesten Stand.
8. Ergänzen Sie in Ihrem Quellcode einige Inline-Kommentare, z.B. Autor und Datum.
9. Übertragen Sie alle Änderungen in das Remote-Repository.
10. Ergänzen Sie eine geeignete .gitignore-Datei für Ihr Projekt, um keine generierten Dateien in die Repositories zu übertragen.2 Löschen Sie ggf. alle Dateien im lokalen und im Remote-Repositories, die dort nicht hingehören.
11. Benennen Sie mindestens eine Datei lokal um und stellen Sie sicher, dass die Umbenennung auch im Remote-Repository erfolgreich war.
12. Führen Sie eine Änderung am Quellcode durch und machen Sie die Änderung wieder rückgängig, indem Sie den letzten Stand des lokalen Repository wiederherstellen.

Lösung 1

A2 Git: Konflikte bereinigen

Nutzen Sie weiterhin Ihr erstelltes Repository aus Aufgabe A1.

1. Für diesen Aufgabenteil müssen zwei entwickelnde Personen an verschiedenen Rechnern mit eigenen Klonen des Repository arbeiten. Alternativ können Sie auch an einem Rechner einen zweiten Klon des Repository in einem anderen Verzeichnis erzeugen und über `git config --local` einen anderen Benutzernamen

und eine andere Mailadresse wählen. Die verschiedenen Namen der Entwickler sind für den Commit-Verlauf wichtig.

2. Stellen Sie zunächst sicher, dass beide Repositories auf dem aktuellen Stand sind.
3. Erzeugen Sie nun in einer bestehenden Quellcodedatei einen Konflikt in einer Codezeile, z.B. indem Sie im ersten Repository „Hello World!“ durch einen Parameter und im zweiten Repository durch einen String in einer anderen Sprache ersetzen.
4. Versuchen Sie nun, beide Änderungen in das Remote-Repository zu übertragen. Bei einem Repository wird es gelingen, beim anderen wird es mit einer Fehlermeldung scheitern.
5. Beheben Sie den Konflikt in dem Repository, bei dem das Übertragen gescheitert ist, indem Sie beide Änderungen geeignet kombinieren.
6. Übertragen Sie dann die neue, konfliktfreie Version zurück ins Remote-Repository.
7. Prüfen Sie den Commit-Verlauf mit `git log --graph`.

Lösung 2

A3 Git: Branching

Sie wollen Ihre Hello-World-Anwendung aus Aufgabe A2 um ein neues Feature zur Ausgabe von »Goodbye Moon!« ergänzen. Erzeugen Sie dazu eine neue Klasse in einer neuen Datei. Nutzen Sie den Feature Branch Workflow aus der Vorlesung und dokumentieren Sie zentrale Schritte durch Screenshots (eine Iteration des Workflows genügt). Achten Sie dabei auf eine angemessene Namensgebung, wie sie in der Vorlesung empfohlen wurde.

Lösung 3

A4 Git: Änderungen rückgängig machen

Recherchieren Sie, wie Sie die folgenden drei Herausforderungen lösen können und geben Sie eine Lösung an:

a) Sie haben bei Ihrem letzten Commit ins lokale Repository leider eine Datei vergessen und Sie wollen den letzten Commit um diese Datei erweitern, ohne dass aber ein ganz neuer Eintrag im Commit-Verlauf entsteht.

b) Sie möchten die bereits ins lokale Repository per Commit übertragenen Änderungen wieder zurücknehmen, so dass dieser Commit ganz aus dem Commit-Verlauf gelöscht wird.

c) Sie möchten den letzten Commit ins Remote-Repository zurücknehmen, so dass aber die Rücknahme selbst Teil des Commit-Verlaufs wird und für andere Entwickler nachvollziehbar bleibt.

Lösung 4