Praktisches Labor: Machen Sie sich mit Git-Befehlen vertraut

Geschätzte benötigte Zeit: 30 Minuten

Ziele

Nach Abschluss dieses Labors werden Sie in der Lage sein:

1. Ein persönliches Zugriffstoken zu erstellen
2. Ein bestehendes Repository über die Benutzeroberfläche zu forken
3. Das geforkte Repository in der Laborumgebung zu klonen
4. Änderungen zum lokalen Branch hinzuzufügen und zu committen
5. Änderungen in das geforkte Repository zu pushen

`Hinweis` Die in diesen Anweisungen enthaltenen Bilder dienen ausschließlich zu Illustrationszwecken.

Übung 1: Erstellen Sie ein PAT zur Authentifizierung der `git commit` und `git push` Befehle

1. Zunächst gehen Sie zu Ihrem GitHub-Konto und klicken Sie auf Ihr Profilbild in der oberen rechten Ecke. Dann klicken Sie auf **Einstellungen**.

2. Wählen Sie als Nächstes **Entwicklereinstellungen**. Diese Option ist normalerweise am unteren Ende des Fensters verfügbar.

3. Navigieren Sie zu **Tokens (klassisch)** unter **Persönliche Zugriffstoken**.

4. Um ein Zugriffstoken zu generieren, klicken Sie auf **Ein persönliches Zugriffstoken generieren**.

5. Füllen Sie auf der Seite **Token generieren** die erforderlichen Angaben aus und aktivieren Sie das Kontrollkästchen **repo**, um den Zugriff für `git`-Befehle zu ermöglichen.

6. Klicken Sie dann auf **Token generieren**.

7. Ihr persönliches Zugriffstoken wird generiert. Das Token ist nur **30 Tage** gültig. Sie müssen ein neues Token generieren, sobald das aktuelle Token abläuft.

   DENKEN SIE DARAN: Stellen Sie sicher, dass Sie Ihr persönliches Zugriffstoken jetzt kopieren. Sie werden es nicht wiedersehen!

Übung 2: Das Repository forken

Um ein Quell-Repository zu forken, führen Sie die folgenden Schritte aus:

1. Melden Sie sich bei GitHub an und gehen Sie zu diesem Projekt's Beispiel-Quell-Repository. Dies ist das upstream-Repository für Ihr Projekt.

2. Klicken Sie oben rechts auf dem Bildschirm auf `Fork` und wählen Sie Ihr eigenes GitHub-Konto als Ziel für den Fork aus.

Eine Kopie des Quell-Repositorys wurde nun als eines Ihrer GitHub-Repositories hinzugefügt. Dies ist das origin-Repository.

Übung 3: Klonen des geforkten Repositories

Ein Klon ist eine lokale Kopie eines Repositories. Bevor du das geforkte Repository klonen kannst, benötigst du zunächst dessen HTTPS-URL, die sicheren Zugriff darauf bietet.

Um das geforkte Repository zu klonen, führe die folgenden Schritte aus:

1. Klicke in deiner Liste der Repositories auf das geforkte Repository. Klicke auf der Hauptseite des Repositories auf die Schaltfläche **Code**.

2. Klicke auf das Clipboard-Symbol, um die URL zu kopieren. Stelle sicher, dass der `HTTPS`-Tab aktiv ist.

3. Öffne das Terminal in der Laborumgebung, indem du das Menü im Editor verwendest: Terminal > Neues Terminal und klone das geforkte Repository mit dem Befehl git clone

```
git clone <deine Repository HTTPS-URL>
```

4. Klicke auf Projekt und erweitere den Ordner des Projekts, das du gerade geklont hast. Du kannst die Dateien im Editor auf der rechten Seite öffnen, indem du auf den Dateinamen klickst.

Übung 4: Fügen Sie Ihre Änderungen hinzu und committen Sie sie

Ein Commit ist Git's Weg, Ihre Dateiänderungen aufzuzeichnen, ähnlich wie Sie ein bearbeitetes Dokument speichern würden. Um die Änderung, die Sie in der vorherigen Übung vorgenommen haben, zu committen, müssen Sie sie zuerst in einen Staging-Bereich hinzufügen. Git wird dann den gestellten Snapshot der Änderungen nehmen und diese im Projekt committen. Denken Sie daran, dass Git Dateien niemals ändert, es sei denn, Sie fordern es ausdrücklich dazu auf.

Um Ihre neue Datei zu committen, führen Sie die folgenden Schritte aus:

1. Um die Änderungen von Ihrem Arbeitsprojektverzeichnis in den Staging-Bereich zu verschieben, geben Sie den folgenden Befehl im Terminalfenster ein:

```
git add .
```

Der `git add` Befehl hat mehrere Optionen. Das einzelne `.` fügt alle untracked Dateien im aktuellen Verzeichnis und in Unterverzeichnissen zum Staging-Bereich hinzu. Alternativ können Sie die einzelne Datei, die Sie erstellt haben, mit dem Befehl `git add <file-name>` hinzufügen. Schließlich können Sie `git add -A` verwenden, um rekursiv alle Dateien vom obersten Git-Ordner hinzuzufügen.

2. Um die neue Datei im lokalen Repository zu committen, müssen Sie zuerst Git mitteilen, wer Sie sind. Geben Sie die folgenden Befehle ein, um Ihre E-Mail und Ihren Benutzernamen festzulegen. Die E-Mail sollte die gleiche wie Ihre GitHub-E-Mail sein.

   Legen Sie Ihre E-Mail fest:

   ```
   git config --global user.email "email@example.com"
   ```

   Legen Sie Ihren Namen fest:

   ```
   git config --global user.name "Ihr GitHub-Benutzername"
   ```

3. Geben Sie den folgenden Befehl im Terminalfenster ein, um die Datei zu committen.
   Hinweis: Es ist immer eine gute Praxis, eine Beschreibung für den Commit hinzuzufügen, damit Sie sich erinnern können, was die Änderung war, falls Sie später darauf zurückgreifen müssen.

- **`-m`-Flag**: Es wird in Git-Commit-Befehlen verwendet, um die Commit-Nachricht direkt in der Befehlszeile anzugeben, sodass Sie eine kurze Beschreibung der Änderungen, die Sie committen, bereitstellen können.

  ```
  git commit -m "Initial Commit"
  ```

Sie können den Befehl `git status` überprüfen, jetzt sagt es, dass es nichts zu committen gibt und der Arbeitsbaum sauber ist. Die neue Datei ist jetzt bereit, von Ihrem lokalen System an den Ursprung auf GitHub gepusht zu werden.

Übung 5: Übertragen Sie Ihre Änderungen nach origin

Dieser Push synchronisiert alle Änderungen, die Sie an Ihrem lokalen System vorgenommen haben, mit Ihrem Fork-Repository auf GitHub.

Um Ihr Update nach GitHub zu übertragen, befolgen Sie die folgenden Schritte:

1. Führen Sie im Terminalfenster den folgenden Befehl aus:

   ```
   git push
   ```

   Hinweis: Wenn Sie dazu aufgefordert werden, geben Sie Ihren GitHub-Benutzernamen und den PAT-Schlüssel ein. Machen Sie sich keine Sorgen, wenn der PAT-Schlüssel nicht sichtbar ist, wenn Sie ihn im Terminal einfügen. Dass der Schlüssel nicht sichtbar ist, ist eine Sicherheitsfunktion. Sobald Sie die Eingabetaste drücken, beginnt das System, die neuesten Änderungen in das Repository zu übertragen.

Wenn Ihr Benutzername und Ihr Passwort akzeptiert wurden, sollten Sie die Änderungen im Terminal sehen, die nach GitHub übertragen wurden.

2. Gehen Sie zu dem Fork-Repository in Ihrem GitHub-Konto und überprüfen Sie, ob die lokalen Änderungen jetzt in den Hauptzweig aufgenommen wurden.

Für ein umfassendes Verständnis der Git-Befehle, tauchen Sie in diese detaillierten Erklärungen ein.

git config mit E-Mail

	Beschreibung
Syntax	`git config --global user.email 'email'`
Zweck	Legt die globale E-Mail-Adresse für Git fest, die sicherstellt, dass jeder Commit, den Sie in allen Repositories machen, mit der angegebenen E-Mail-Adresse verknüpft wird
Beispiel	`git config --global user.email 'testuser3@abcmail.com'` Setzt die E-Mail-Adresse auf testuser3@abcmail.com

git config mit Benutzername

	Beschreibung
Syntax	`git config --global user.name 'username'`

	Beschreibung
Zweck	Setzt den globalen Benutzernamen für Git, wodurch sichergestellt wird, dass jeder Commit, den Sie in allen Repositories machen, mit dem angegebenen Benutzernamen verknüpft wird
Beispiel	`git config --global user.name 'John Doe'` Setzt den Benutzernamen auf John Doe

git add [für Dateiaktualisierungen]

	Beschreibung
Syntax	`git add [Datei(en)]`
Zweck	Füge Dateiänderungen zum Staging-Bereich hinzu
Beispiel	`git add` Fügt alle Änderungen im aktuellen Verzeichnis zum Staging-Bereich hinzu

git add [für alle Updates]

	Beschreibung
Syntax	`git add --a` / `git add -A` / `git add –all`
Zweck	Alle Änderungen, einschließlich neuer Dateien, Modifikationen und Löschungen, für den nächsten Commit vorbereiten Alle drei Befehlsvarianten sind identische Befehle, die austauschbar verwendet werden, um dieselbe Funktionalität zu erreichen
Beispiel	`git add --a` Bereitet alle Änderungen vor

git status

	Beschreibung
Syntax	`git status`
Zweck	Den Status des Arbeitsverzeichnisses und des Staging-Bereichs anzeigen

git commit

	Beschreibung
Syntax	`git commit -m "[commit message]"`
Zweck	Änderungen im Repository mit einer Commit-Nachricht aufzeichnen

	Beschreibung
Beispiel	`git commit -m "Initial commit"` Commitet die gestagten Änderungen mit der Nachricht „Initial commit“

git push

	Beschreibung
Syntax	`git push [remote] [branch]`
Zweck	Überträgt bestätigte Änderungen in ein entferntes Repository
Beispiel	`git push origin main` Überträgt bestätigte Änderungen vom "main"-Zweig in das entfernte Repository

Hinweis zur Datenverwaltung und Persistenz

Um die ordnungsgemäße Verwaltung und Persistenz Ihrer Daten in einem GitHub-Repository sicherzustellen, ist es wichtig, einige wesentliche Schritte zu befolgen:

Regelmäßige Updates: Wann immer Sie Änderungen vornehmen oder neue Komponenten zu Ihrem Projekt hinzufügen, ist es unerlässlich, die Updates zu Ihrem GitHub-Repository hinzuzufügen, zu committen und zu pushen. Dies stellt sicher, dass Ihre neuesten Arbeiten sicher gespeichert und für Mitwirkende zugänglich sind.

Sitzungspersistenz: Während einer aktiven Sitzung sind Ihre Daten zugänglich. Es ist jedoch wichtig zu beachten, dass Sie das Repository erneut klonen müssen, um die Arbeit fortzusetzen, wenn Ihre Sitzung abläuft oder Sie sich abmelden.

Ignorieren von Node-Modulen: Beim Pushen von Daten zu GitHub ist es eine bewährte Praxis, den Node-Module-Ordner sowohl aus Ihrem Server- als auch aus Ihrem Client-Verzeichnis auszuschließen. Dieser Ordner enthält externe Abhängigkeiten und kann ziemlich groß sein, was das Repository schwer macht und den Prozess verlangsamt. Indem Sie ihn zur .gitignore-Datei hinzufügen, verhindern Sie, dass er ins Repository gepusht wird, und halten Ihre Commits sauberer und fokussierter.

Durch die Einhaltung dieser Richtlinien können Sie ein gut organisiertes und effizientes GitHub-Repository pflegen, das sicherstellt, dass Ihre Arbeiten sicher gespeichert und für Sie und Ihre Mitwirkenden leicht zugänglich sind.

Author(s)

Ritika Joshi