IWGS2_SS20_Zettel2

July 7, 2020

1 Informatische Werkzeuge in den Geistes- und Sozialwissenschaften II

1.1 Hausaufgabe 2 (Einführung in Git)

Erschienen: 02.05.2020
Abgabe bis: 10.05.2020

Bitte laden Sie Ihre Notebooks bis 23:59 Uhr am Abgabetag in Ihrer Übungsgruppe bei StudOn hoch.

Wenn Ihnen einige der hier verwendeten Konzepte unbekannt sind oder Sie nicht wissen, wie Sie fortfahren sollen, können Sie die Vorlesungsunterlagen zu Rate ziehen oder jederzeit Fragen im Forum oder auf Slack stellen, im Tutorium nachfragen, oder Ihrem Tutor eine Mail schreiben.

1.2 Aufgabe 2.1 (Gitlab)

Unser Projekt in diesem Semester ist klar als Teamprojekt ausgelegt. In einer Welt nach dem Studium werden Sie auch in aller Regel in Teams arbeiten und deshalb lohnt es sich, frühzeitig zu lernen, wie solche Projekte verwaltet werden können. Das de-facto Standardtool in der Praxis ist dafür Git (zumindest für Softwareentwicklung). Git erlaubt es uns, in Teams am selben Projekt zu arbeiten. Wir können gleichzeitig an unserem Code arbeiten und eine Versions-Historie verwalten.

Wie in der Vorlesung gelernt, werden sogenannte Git-Repositorien typischerweise auf einem Server verwaltet. Jedes Teammitglied lädt sich dann eine lokale Kopie auf den eigenen Rechner, bearbeitet dort den Code oder die Daten und stellt danach die Änderungen den anderen Mitgliedern wieder zur Verfügung. Der Server, den wir in dieser Vorlesung verwenden, ist das Gitlab der FAU: https://gitlab.cs.fau.de/

Aufgabe: Registrieren Sie sich im Gitlab. Verwenden Sie als Benutzernamen Ihre IDM-Kennung. In Gitlab kann man Gruppen anlegen. Wir haben eine solche Gruppe angelegt für alle Studierenden der IWGS: https://gitlab.cs.fau.de/iwgs-students-ss20. Wenn Sie eingeloggt sind, fragen Sie Zugang zu dieser Gruppe an. Klicken Sie dazu auf der verlinkten Seite oben auf "Request Access". Sie müssen für die nächsten Teilaufgaben nicht warten, bis Ihnen Zugang zu dieser Gruppe gewährt wird. Für die nächste Teilaufgabe brauchen Sie den Zugriff noch nicht. Allerdings kann es ein wenig dauern, bis wir den Zugang freischalten, also machen Sie das lieber früher als später.

1.3 Aufgabe 2.2 (Eigenes Repository anlegen)

Legen Sie im Gitlab ein eigenes Repository an. Dazu klicken Sie oben auf "New project". Geben Sie dem Projekt einen Namen und eine kurze Beschreibung. Setzen Sie außerdem den Haken bei "Initialize repository with a README". Abschließend klicken Sie auf "Create project".

Wir müssen ein paar Anpassungen an den Einstellungen des neuen Repositoriums machen. Die Einstellungen finden Sie in der linken Leiste unter "Settings". 1. In den Einstellungen wählen Sie den Unterpunkt "Repository" aus. Klappen Sie dort "Protected Branches" aus und klicken dann auf den orangenen Knopf "Unprotect". Diese Einstellung dient dazu, dass Mitglieder Ihres Teams eigene Änderungen auf Ihrem Repositorium veröffentlichen dürfen. 2. Als nächstes geben Sie Ihren TeamkollegInnen Zugang zu dem Repositorium. Wählen Sie dazu den Unterpunkt "Members" links in den Einstellungen aus. Suchen Sie Ihre Teammitglieder und geben Sie ihnen die Rolle "Developer". 3. Sobald sie Zugriff zu der Gruppe "iwgs-students-ss20" haben, geben Sie außerdem der Gruppe Reporter-Zugang zu Ihrem Repository. Nur so können wir Ihre Schritte nachverfolgen. Um Gruppen Zugriff zu geben, wechseln Sie oben den Reiter zu "Invite Group".

1.4 Aufgabe 2.3 (Clone)

Wie oben beschrieben arbeiten wir nicht direkt auf den Daten auf dem Server, sondern auf lokalen Kopien. Um diese Kopie anzulegen, führen wir den Befehl `clone` auf unserem Rechner (bzw für unseren Fall im Jupyter) aus. Öffnen Sie dazu hier im Jupyter ein Terminal und navigieren Sie (mit `cd`) in ein Verzeichnis Ihrer Wahl, in dem Sie arbeiten möchten. Auf der Gitlab-Webseite navigieren Sie zu der Hauptseite Ihres Projektes. Klicken Sie rechts auf den blauen Knopf "Clone" und kopieren Sie den Link, der dort unter "HTTPS" auftaucht. In Ihrem Terminal im Jupyter tippen Sie dann den Befehl `git clone LINK`, wobei Sie natürlich LINK mit dem kopierten Link ersetzen. Nachdem Sie den Befehl ausgeführt haben, werden Sie nach einem Benutzernamen und Passwort gefragt. Nach erfolgreicher Eingabe sollte ein neuer Ordner mit dem Namen Ihres Repositoriums in Ihrem Dateisystem auftauchen.

1.5 Aufgabe 2.4 (Commit & Push)

Legen Sie in Ihrem Repository eine neue Textdatei an, schreiben Sie ein paar Sätze Inhalt und führen Sie dann die Schritte durch, die nötig sind, um die Datei auf Gitlab zu laden (`add`, `commit`, `push`). Geben Sie eine aussagekräftige Commit-Nachricht an.

Beschreiben Sie in der nächsten Zelle in eigenen Worten kurz, welche Befehle in welcher Reihenfolge nötig waren. Was genau macht jeder Befehl? Warum brauchen wir 3 Befehle, um die Änderungen zu veröffentlichen?

Außerdem: Wie kommen Ihre Teammitglieder an die veränderten Daten? Welcher Befehl ist dazu nötig?

Ihre Antwort in dieser Zelle.

1.6 Aufgabe 2.5 (Merge-Konflikte)

Dadurch, dass wir auf lokalen Kopien arbeiten, kann es passieren, dass Teammitglieder auf entweder veralteten Kopien arbeiten oder am selben Stück Code Änderungen vornehmen. Wenn Git nicht automatisch entscheiden kann, wie es diese Änderungen in das Repositorium einpflegen soll, kommt es zu *Merge-Konflikten*.

Beschreiben Sie in der nächsten Zelle ein Szenario, wie eine solche Situation entsteht. Beschreiben Sie Ihr Szenario dabei so konkret wie möglich. Geben Sie dazu genau an, in welcher Reihenfolge welches Teammitglied welchen Befehl ausführen muss (dh. `add`, `commit`, `push`, `pull`).

Spielen Sie mit Ihren Teamkollegen ein solches Szenario in Ihrem Repositorium durch. Woran erkennen Sie, dass es zu einem Konflikt gekommen ist? Welche Schritte müssen Sie durchführen, um den Konflikt zu beheben? Wir möchten hier am Ende den Konflikt und seine Auflösung in der Commit-Historie Ihres Repositoriums nachverfolgen können.

Ihre Antwort in dieser Zelle.