GIT & Versionierung

Christoph Rinne*

09. Januar 2022

Inhaltsverzeichnis

1 Grundlegende Information

GIT ist eine freie Software für das verteilte Arbeiten an Dateien. Sie gewährleistet die Synchronisation und Kontrolle aller Änderungen oder kurz gesagt die Versionierung aller Dateien. Bei textbasierten Dateien, z.B. Markdown und csv-Tabellen, erfolgt dies auf Zeilenniveau. Die Kooperation an Projekten mit Texten, Daten, Abbildungen und Programmcode, z.B. R, ist damit ohne den sonst üblichen Aufwand und oft auftretende Verwirrung möglich.

Die Synchronisation erfolgt zu einem Server und einem dort betriebenen Dienst, hier sind zwei Webanwendungen sehr bekannt: GITLab und GITHub. Die lokale Kontrolle wird ebenfalls durch GIT gesteuert, dafür gibt es diverse grafische Oberflächen (GUI), z.B. GIT Extensions. Auch R-Studio bietet einen vollständige *workflow* für die alltägliche Arbeit mit GIT. Für das Schreiben auf dem Server ist eine Authentifizierung notwendig die meist über einen ssh-key erfolgt.

Bei der asymetrische Verschlüsselung ist ssh ein weit verbreiteter Standard und in allen gängigen Betriebssystemen integriert. Grundlegende Informationen zu dieser Verschlüsselung gibt es u.a beim BSI: Arten der Verschlüsselung. Eine Einführung zu ssh und den Befehlen finden Sie u.a. bei GITLab: GITLab and SSH keys.

2 GIT nutzen

2.1 Szenarien zur Nutzung

Das Naheliegendste ist es, GIT für das Entwickeln von Programmen zu nutzen. In einem archäologischen Kontext sind aber auch andere Szenarien und Kombinationen daneben denkbar.

2.1.1 Daten sammeln

Das Sammeln von Daten erfolgt lokal, auf dem eigenen PC, gerne in MS Excel alternativ in LO Calc oder in echten Datenbankmanagementsystemen wie MS Access oder SQLite. Wollen Sie diese Daten aber gemeinschaftlich erheben oder nutzen kommen Sie um einen Datenaustausch nicht umhin.

Eine zentrale Datenhaltung kann innerhalb eines lokalen Netzwerkes mit Freigaben und in MS Access mit verknüpften Datenbanken (backend) oder in SQL mit angehängten Datenbeständen (attached) arbeiten. Das ist nicht wirklich kompliziert, setzt aber für eine performante Nutzung eine gute Datenstruktur voraus. Sobald Sie das lokale Netzwerk verlassen sind Sie auf eine Serverdatenbank (z.B. PostrgeSQL)

*Christian-Albrechts Universität zu Kiel, crinne@ufg.uni-kiel.de

angewiesen und haben einen sehr hohen administrativen Aufwand auch bei Sicherheitsaspekten. Für Projekte einer überschaubaren Größe also keine Alternative.

Alternativ: sammeln Sie Daten als CSV-Tabellen und synchronisieren (pushen) diese regelmäßig zum Repositorium. Wie Sie diese CSV-Tabelle lokal editieren ist fast egal, möglich sind MS Excel, LO Calc oder ein Texteditor. **Allerdings** ist der Wechsel zwischen MS Excel und LO Calc sehr umständlich und fehleranfällig, da MS Excel die Spalten mit einem ";" und LO Calc diese mit einem "," trennt. Optional können Sie auch ein anderes Trennzeichen verwenden, z.B. . Allerdings werden diese Dateien mit der Endung .txt oder .csv nicht mehr automatisch richtig erkannt und beim Speichern muss in MS Excel jedes mal das Trennzeichen definiert werden während im LO Calc automatisch ein ',' gesetzt wird. Der **Vorteil** ist die Versionierung durch GIT, also das Protokoll der vollzogenen Änderungen, und die zentrale Ablage auf einem Server um dessen Sicherheit und Zugangsberechtigungen Sie sich nicht unmittelbar kümmern müssen.

2.1.2 Texte Verfassen

Sofern Sie sich an die Arbeit mit Rtudio und das Schreiben von Markdown gewöhnt haben, können Sie über GIT natürlich auch gemeinsam Texte, Artikel und ganze Bücher mit *Bookdown* schreiben. Sie sollten innerhalb von RStudio nicht mit einzelnen Dateien sondern mit Projekten und Versionskontrolle arbeiten. In diesem Projekt können Sie dann die benötigten Daten (CSV) als auch die Literatur als BibTeX-Datei pflegen und in Ihrem Text einbinden. Tabellen, Analysen und resultierende Grafiken werden mittels R-Code auf Grundlage der vorliegenden Daten im Text generiert und sind stets aktuell. Natürlich ist dies eine grundlegend andere Arbeitsweise als eine gemeinsame Ablage in der Cloud und die Versionierung von Hand, selbst bei der Verwendung von Online Office. Es ist eine Umstellung mit anfänglichen Reibungsverlusten und benötigt Einarbeitungszeit.

Anmerkung: Wenn Sie dieses Thema *Online Texteditieren* interessiert, sehen Sie sich auch mal folgendes an Fidus Writer.

2.1.3 Reproducible Research

Wissenschaftliche Arbeit wird belegt und ist nachvollziehbar. In einer digitalen Arbeitsumgebung sollten wir diese Grundlage wissenschaftlichen Arbeitens angemessen umsetzen. Die Arbeit mit allgemein zugänglichen Scriptsprachen wie R für die Analysen und das Bereitstellen des Codes und der zugehörigen Daten in einem öffentlichen Repositorium erfüllt diesen Anspruch vollumpfänglich und ist *up to date*.

2.2 GIT-Plattform und GIT-Account

Neben den bekannten Onlineanbietern (s. Grundlegende Informationen) bieten Universitäten selber GIT an. Informieren Sie sich hierzu bei Ihrem Rechenzentrum. Richten Sie sich in dem von Ihnen gewählten System ein Konto ein. In diesem Konto wird der öffentliche Teile Ihres SSH-Schlüssels abgelegt, nur so können Sie aus Anwendungen wie GIT Extensions und RStudio Daten synchronisieren.

Beachten Sie bitte: kostenlose GIT-Konten sind oft öffentlich.

2.3 GIT lokal einrichten

Der Vorgang ist hier für Windows 10 beschrieben.

2.3.1 SSH

Sie können OpenSSH auf der Kommandozeile verwenden, es gehört zu den optionalen Programmen (*optional features*) bei Windows 10 oder Sie nutzen PuTTy mit einer grafischen Oberfläche. Für beide Wege gibt es ergänzende Anleitungen im Netz (u.a. hier).

Laden Sie das Programmpaket (*.msi) von PuTTy herunter https://www.chiark.greenend.org.uk und installiernen Sie es. Starten Sie dann das Programm **puttygen**. Die Vorgabe ist ein RSA-Schlüssel, belassen Sie dies und wählen Sie [Generate]. Danach müssen Sie die Maus über dem leeren Fensterausschnitt bewegen, um einen zufälligen Schlüssel zu erzeugen. Im folgenden Fenster ergänzen Sie einen für Sie sinnvollen *Key comment*, z.B. ich@pc-home und tragen ein Passwort (*Key passphrase*) ein. Anschließend müssen Sie nacheinander mit [Save public key] und [Save private key] beide Schlüssel speichern. Der

Standardorder dafür in Windows ist c:\Benutzer\<nutzername>\.ssh . Nennen Sie beide identisch, ergänzen Sie ggf. das Datum im Namen und verwenden Sie die vorgegebene Endung ppk nur für den privaten Teil, den öffentlichen belassen Sie ohne Endung. Kopieren Sie bereits in diesem Fenster den öffentlichen Schlüssel in die Zwischenablage (*Public key for pasting ...*). Melden Sie sich auf dem Server in Ihrem GIT Konto an, wechseln Sie zu dem Abschnitt der SSH Keys, ergänzen Sie hier einen neuen Key und fügen den öffentlichen Teil aus der Zwischenablage in dem entsprechenden Fenster ein.

2.3.2 GIT für Windows

Dann benötigen Sie GIT. GIT nutzt keinen eigenen Editor sondern fragt nach einem Verweis auf ein existierendes Programm. Ich nutze hier den Editor Notepad++ den Sie ggf. vorab installieren sollten.

2.3.3 GIT Extensions

Laden Sie sich die aktuelle Version (*.msi) von GIT Extensions herunter. Zudem wird kdiff3 für die Synchronisation (merge) verwendet. Laden Sie auch dieses Programm herunter und installieren Sie es vorab mit den Vorgaben. Installieren Sie nun GIT Extensions und folgen Sie den Schritten der Installation mit den Vorgaben. Folgendes bedarf der Aufmerksamkeit:

- Select SSH Client: verwenden Sie hier PuTTy.
- Telemetry privacy policy: Im Prinzip handelt es sich um Geben und Nehmen. Sie nutzen die Software und geben Teile Ihres Nutzerverhaltens für die Optimierung derselben preis. Ihre Entscheidung.

Starten Sie GIT Extensions. Vermutlich werden Sie zuerst mit einem Fenster zu den Einstellungen "Settings - Checkliste" konfrontiert, wenn nicht gehen Sie dorthin: Werkzeuge > Einstellungen.

- Benutzername und E-Mail-Adresse: Bitte entsprechendes eintragen
- Direkt darunter im selben Fenster fehlen weitere Angaben
- Merge-Werkzeuge: kdiff3
- Pfad zu Merge-Werkzeug: [Durchsuchen] und dort kdiff3 auswählen, die nächtste Zeile zu den Befehlen sollte automatisch ergänzt werden.
- Wiederholen Sie den Vorgang für die Vergleichswerkzeuge.

Zeilenenden werden in Windows anders kodiert als in Linux/Mac. Aktivieren Sie deshalb die Option "Zeilenumbrüche im Windows-Format auschecken...". Ansonsten würde jedes Zeilenende zu einer protokollierten Änderung.

2.3.4 GIT Extensions Projekt nutzen

Im folgenden wird nur ein Projekt auf den lokalen Rechner übertragen (clone), eine Änderung kommentiert (*commit*) und hochgeladen (*push*). Für die vielen Möglichkeiten von GIT Extensions konsultieren Sie bitte das Benutzerhandbuch: Hilfe > Benutzerhandbuch.

Sofern nach dem Start kein Fenster zum Öffnen oder Klonen von Repositorien angezeigt wird: Start > Repository klonen ... . In dem neuen Fenster "Klonen" ergänzen Sie folgendes:

- Zu klonendes Repository: Die SSH-URL aus dem Online-Repositorium, z.B. git@cau-git.rz.uni-kiel.de:.
- Ziel: den Ordner in dem ein neuer Ordner für das Repositorium angelegt wird.
- Zu erstellendes Unterverzeichnis: Wird automatisch sinnvoll ergänzt.
- Branch: Standard ist Remote-HEAD

Alles weitere nach Vorgabe. Mit [SSH-Schlüssel laden] können Sie jetzt ihren privaten Schlüssel laden und klonen. Direktes [Klonen] fragt nachträglich nach dem privaten Schlüssel. Der private Schlüssel muss mit dem hinterlegten Passwort authentifiziert werden. Danach sollte der Kopiervorgang erfolgreich laufen.

Nehmen Sie Änderungen immer in überschaubaren und möglichst abgeschlossenen Schritten vor. Nach einer Änderung wird Ihnen an dem Symbol "Committen" in der Menüleiste die Anzahl der geänderten Dateien angezeigt. Starten Sie über die Iconleiste "Committen" oder über Befehle > Committen den Upload. Markieren Sie oben links die zu kopierende Datei, die Änderungen werden rechts oben angezeigt. Fügen Sie alle zu *pushenden* Dateien mit "Stagen" in den Uploadbereich ein. Schreiben Sie rechts unten einen Kommentar zu den Änderungen. Mit [Committen] werden die Kommentare angehängt und ein Update-Paket erstellt das unmittelbar auch *gepushed* werden kann.

Im unteren Bereich des Programmfensters haben Sie neben “Committen” vier weitere Reiter: Diff, Dateibaum, GPG, Konsole. Der Reiter Diff zeigt Ihnen die änderungen gegenüber der letzten Version, entsprechend der Ansicht vom letzten Commit. Der Reiter Dateibaum erlaubt das Editieren einzelner Dateien: links der dateibaum, recht ein Fenster mit dem Editor aus der Konfiguration von GIT Extensions. Mir fehlt hier die gewohnte Menüleiste meines Editors, deshalb nutze ich diesen Bereich nicht.

2.3.5 RStudio & GIT

Anleitungen hierzu finden Sie mehrfach im Internet, u.a. unter der wichtigen Überschrift **Reproducible Research** bei der Uni Zürich. Auch hier wird ein ein SSH-Schlüssel und ein installiertes GIT vorausgesetzt (s.o.), ebenso natürlich die Installation von R und RStudio.

Ergänzend zu den Informationen der eingangs genannten Anleitung folgende Hinweise:

- Der Menüeintrag GIT ist nur sichtbar, wenn ein entsprechendes Projekt geöffnet ist.
- Haben Sie GIT bereits konfiguriert, auch über GIT Extensions (s.o.), brauchen Sie den Abschnitt *Git Setup* nicht erneut ausführen. Um zu testen, ob Ihr Name und E-Mail korrekt hinterlegt sind öffnen Sie die Git Kommandozeile (Git Bash) und schreiben Sie folgende Befehle: `git config --get user.name` und dann `git config --get user.email`. Die jeweilige Rückabe sollten Ihr Name und die E-Mail-Adresse sein.