Timo Grossenbacher, Datenjournalist SRF Data, timo@timogrossenbacher.ch
Twitter: @grssnbchr

Git, GitHub und Co. – was geht mich das an?

Git ist eine sogenannte Versionskontrollsoftware. Was kompliziert tönt, ist es auch. Und trotzdem erleichtert Git den Alltag von Entwicklern und Journalisten, die mit Entwicklern zu tun haben (müssen) enorm. Ein Journalist, der regelmässig mit professionellen Entwicklern zusammenarbeitet, und Git nicht kennt, wird früher oder später massiv ins Hintertreffen kommen. Deshalb fasst dieses Handout das 101 von Git (Vorteile, Nachteile, wichtigste Begriffe) zusammen.

Geschichte: Git gibt es in seiner heutigen Form seit rund 10 Jahren und – wer hätte es gedacht – wurde von Linus Torvalds, dem Erfinder von Linux, höchstpersönlich auf die Welt gesetzt. Er nutzte das System vorderhand, um den immer unübersichtlicher gewordenen Quellcode von Linux besser zu verwalten. Git ist FOSS (Free and Open Source Software) und daher durch jedermann auf jedem Rechner installier- und verwendbar.

Git schaffte erst in den letzten paar Jahren durch die steigende Popularität von «Tauschbörsen» wie **GitHub**, BitBucket und Co. den absoluten Durchbruch und ist heute zumindest bei Open-Source-Projekten der De-Facto-Standard.

GitHub und Git sind keineswegs das gleiche. GitHub ist ein Server für Git – salopp ausgedrückt eine Maschine, auf der eine (weitere) Kopie von einem Projekt gespeichert wird. https://github.com ermöglicht es, Code, Daten und Software im Sinne von Open Source zu teilen und – viel wichtiger – gemeinsam an Projekten zu arbeiten, ohne einen eigenen Server aufzusetzen. GitHub ist grundsätzlich kostenlos, der Betrieb von «Organisationen» (mehrere Mitglieder arbeiten an einem Projekt) und privaten Projekten kostet aber (für 5 Mitglieder zahlt man etwa 300 Franken jährlich).

Jetzt mal zur Sache: **Was macht Git überhaupt?** Moderne Softwareentwicklung (auch redaktioneller Art) findet in Iterationen statt. Werden neue «Features» entwickelt (z.B. ein Filter-Button bei einer Datenvisualisierung) oder Fehler behoben («Bugfixing») geschieht dies immer in sogenannten **Commits** oder Entwicklungsschritten. Jeder Commit beinhaltet somit eine neue Version einer Software (oder eines Datensatzes, eines Bilds, eines Videos, usw.). Git ermöglicht es, einfach zwischen diesen Commits hin- und herzuschalten, die Arbeit von mehreren Leuten zu koordinieren, einfach Backups zu erstellen, usw.

Die 5 wichtigsten Vorteile von Git:

1. **Versionierbarkeit:** «Halt, beim letzten Review mit der Designerin gefiel mir die andere Schriftart doch besser». Kein Problem, alte Versionen (von einzelnen Dateien) lassen sich problemlos wiederherstellen und in den aktuellen Stand integrieren. Besonders hilfreich, wenn plötzlich unerklärbare Fehler auftauchen.
2. **Kooperation:** Viele Redaktionen haben heute mindestens zwei oder sogar mehr Entwickler und Designer, die gemeinsam an Projekten arbeiten. Und auch (Daten-)Journalisten greifen vermehrt in den Code und in Daten ein: Sei es, um ein paar Texte abzuändern oder Rundungsfehler in Datensätzen zu beheben. Mit Git ist es faktisch unmöglich, Änderungen gegenseitig zu überschreiben – und es ist jederzeit klar, wer welche Datei zuletzt geändert hat.
3. **Vielfältigkeit:** Mit Git lässt sich nicht nur Code, sondern auch Daten versionieren – von einfachen Datensätzen (Textdateien) zu Bildern und Videos (Binärdateien). Jemand hat sogar deutsche Gesetzestexte versioniert.
4. **Transparenz:** In der Zeit von «Lügenpresse» wird es für (Daten-)Journalisten immer wichtiger, die Methoden und Daten hinter einer Geschichte transparent zu machen. GitHub (und andere) sind perfekte Plattformen, um allen Interessierten Einblick hinter die Kulissen zu gewähren. Beispiel: Alle Daten zur Uni-Recherche von SRF Data.

5. **Datensicherheit / Backups:** Das Besondere an Git ist unter anderem, dass es dezentralisiert organisiert ist. Sprich: Jeder Entwickler / Journalist / Teilnehmer behält seine eigene, lokale Kopie des Projekts auf seiner Maschine. Dazu kommt in der Regel noch ein zentraler Git-Server (normalerweise GitHub u.a.). Aktualisieren die Teilnehmer ihre lokalen Kopien ab und zu, ist es also sehr unwahrscheinlich, wertvollen Code und Daten zu verlieren – ganz ohne teure Backup-Infrastruktur.

Und nun noch zu den wenigen **Nachteilen** (besser: Vorwarnungen). Wie jedes Informatiksystem bedingt Git ein gewisses **Grundwissen**, das sich jedoch in 2-3 Tagen aneignen lässt. Entwickler und andere Informatiker verfügen in der Regel bereits über dieses Wissen und können interne Schulungen durchführen. Auch die **Systemumgebung** kann Kopfzerbrechen bereiten: Obwohl Git und GitHub mittlerweile gut über **Windows** zu bedienen sind (ja, man muss etwas installieren können dürfen), funktioniert es auf **UNIX** (Mac, Linux) einfach doch besser. Zur Bedienung gibt es mittlerweile grafische Oberflächen mit Buttons usw., aber wer Git wirklich gut verstehen möchte, tut besser daran, sich auf die **Kommandozeile** zu wagen (für Laien: so etwas wie DOS, bevor es das farbige Windows gab). Zuletzt: Für GitHub gilt wie auch für alle anderen grossen Cloud-Dienste, dass die Daten von US-amerikanischen Firmen gehosted werden und man deshalb **sensible Datensätze** lieber auf dem lokalen Rechner ruhen lassen sollte.

Die wichtigsten Befehle / Begriffe von Git:

- **Clone:** Download eines Projekts (zum Beispiel von GitHub) auf den lokalen Rechner.
- **Stage:** Mit diesem Befehl werden die zu versionierenden Dateien auf die «Bühne» gehoben – sie werden bereitgemacht für den nächsten Commit. Das impliziert: Nicht alles muss versioniert werden, z.B. können sensible Daten «unstaged» bleiben.
- **Commit:** Werden Dateien committet, wird ihr aktueller Stand eingefroren. Jeder Stand kann nun mit anderen geteilt und/oder wiederhergestellt werden.
- **Push/Pull:** Mit diesen beiden Operationen kommuniziert man mit einem anderen Server, also z.B. mit GitHub oder einem privaten Server, und damit über Umwegen mit anderen Projektteilnehmern. Dabei werden lokale Versionen mit denen auf jenem Server abgeglichen – falls es Konflikte gibt (wenn zum Beispiel zwei Personen die gleiche Datei bearbeitet haben) müssen diese zwingend «gelöst» werden. Das verhindert viele Nervenzusammenbrüche.
- **Merge:** Bei einem Merge werden zwei Versionen in eine «verschmolzen». Wurden Dateien an unterschiedlichen Stellen verändert, geschieht dies in der Regel reibungslos. Nur wenn wirklich die selben Stellen gleichzeitig bearbeitet wurden, gibt es einen Konflikt. Merge wird bei Push/Pull automatisch ausgeführt.
- **Branch:** Projekte können in Branches verzweigt werden – ein Entwickler arbeitet zum Beispiel an einem Branch mit einer neuen Social-Media-Funktion, und nur wenn sich diese als brauchbar erweist, wird sie in den Hauptzweig integriert («master»-Branch).
- **Fork:** Etwa das gleiche wie Clone, aber von GitHub zu GitHub. In der Regel wird ein Projekt geforkt, um es selber weiterentwicklen zu können, ohne mit den ursprünglichen Entwicklern in Kontakt treten zu müssen. Ist die Weiterentwicklung für alle von Nutzen (z.B. weil Fehler behoben wurden), kann es sein, dass sie vom ursprünglichen Projekt wiederum integriert wird.

Weiterführende Links/Literatur

- Ein Einstieg: http://rogerdudler.github.io/git-guide/index.de.html (gibt es auch auf Englisch)
- Wenn das obige bereits zu nerdig ist, fängt der offizielle Guide etwas behutsamer an: https://git-scm.com/book/de/v1 – idealerweise liest man neben Kapitel 1 auch noch 2, 3 und 4 (ca. 2 Tage).
- Alle publizierten Projekte von SRF Data in einer Übersicht (zur Inspiration): https://srfdata.github.io
- Auch andere Medien publizieren ihre Geschichten auf GitHub, z.B. die Berliner Morgenpost, die NYT, 538, Pro Publica.
- Die Slides dieser Präsentation: https://grssnbchr.github.io/mazmmt-git