Lehrstuhl für
Datenverarbeitung

Vorgehensweise bei Projektarbeiten mit Git

D. Meyer (Februar 2015)

Einführung

Es freut mich, dass Sie sich für eines der ausgeschriebenen Projekte interessieren und dieses im Rahmen eines Studierendenprojekt bearbeiten möchten.

In diesem Dokument sollen die Voraussetzungen beschrieben und eine Anleitung gegeben werden, wie das Studierendenprojekt mit der Unterstützung der Versionsverwaltungssoftware Git und eines zugehörigen Issue-Trackers ausgeführt werden soll. Zudem müssen einige Voraussetzungen erfüllt und Verpflichtungen eingegangen werden, die nun im nächsten Abschnitt beschrieben werden sollen. Zusätzlich gibt es als eine Grundvoraussetzung eine Anmeldeprozedur, die erfolgreich durchlaufen werden muss.

Voraussetzungen

- Git muss vorher bekannt sein/erlernt werden.
- Code muss in Git verwaltet werden und im entsprechenden Branch bearbeitet werden.
- Es müssen die Verhaltensregeln für das Arbeiten mit Git sowie dem Issue-Tracker eingehalten werden.
- Es müssen alle notwendigen Kriterien für den erfolgreichen Abschluss erfüllt werden.
- Es muss am Ende eines jeden Arbeitstages ein kurzes Tagesfazit/Tageskommentar verfasst werden.
- Es muss genau dokumentiert werden, sodass nachfolgende Arbeiten nahtlos anschließen können.

Definitionen

Bevor das Vorgehen näher erläutert wird, müssen zunächst einmal einige Grundbegriffe geklärt werden:

Git Git ist ein sogenanntes Versionsverwaltungssystem. Dabei können nicht nur die Quellcodedateien oder Dokumente auf einem entfernten Server gespeichert und mit anderen geteilt werden, sondern es wird auch eine komplette Versionshistorie mitgeführt. Dies ermöglicht es leicht, wieder zu vorigen Versionen zurück zu springen oder problematische Änderungen nachzuvollziehen.

Zusätzlich gibt es sogenannte Branches, die das parallele Arbeiten am gleichen Code erlauben.

Branch In Git gibt es die sogenannten Branches, mit denen die Entwicklung von unabhängigen Programmteilen vorangetrieben werden kann. Der Vorteil liegt dabei darin, dass der “Master” Branch nicht beeinflusst wird, somit also für alle anderen Nutzer während der Entwicklungszeit eine stabile Codebasis erhalten werden kann. Dabei wird von dem sogenannten “Master” Branch, zu einem gewissen Zeitpunkt eine unabhängige Kopie erstellt, die unabhängig von zukünftigen Änderungen am Master bearbeitet werden kann. Natürlich sollten die Änderungen weiterhin mit diesem kompatibel gehalten werden (zum Beispiel Interfaces, Funktionsdefinitionen, Namen, etc.), da nach der Entwicklung eines Features wieder zurückgemerged, sprich die Änderungen zurückfließen sollen.

Merging Ist ein neues Feature in einem Branch zuendeentwickelt (wenn Sie Beispielsweise alle Issues ihrer Ingenieurpraxis abgearbeitet und umfangreich getestet haben), so kann der Branch für dieses Feature in den "Master" Branch gemerged werden. Dies bedeutet, dass die Änderungen, die Sie vorgenommen haben nun für alle anderen Entwickler, die ihre Entwicklung auf dem Masterbranch basieren, verfügbar werden.

/Issue-Verwaltung Im Gitlab-System, das wir verwenden muss grundsätzlich zwischen der Sourcecodeverwaltung Git und dem Issue-Tracker unterschieden werden. Wobei ersteres alleinig für die Verwaltung der Quellcode und sonstiger Dateien verantwortlich ist und auch unabhängig von der Issue-Verwaltung eingesetzt werden kann, so ist die Issue-Verwaltung ein Zusatzfeature des Gitlab-Systems, das alleine in der Weboberflächer verfügbar ist.

Milestone Ein sogenannter Meilenstein ist ein Zwischenziel im Gesamtprojekt. In der Weise, wie wir die Projektverwaltung verwenden werden, entspricht ein Meilenstein immer genau einer Ingenieurpraxis.

Issue Jeder Meilenstein hat sogenannte Issues untergeordnet. Das können Bugs oder Featurerequests sein. Diese müssen alle abgearbeitet werden und geschlossen werden, dass der übergeordnete Meilenstein als erfüllt gilt. Konkret bedeutet das für Sie, dass sie alle Issues, die dem Meilenstein ihrer Praxis zugeordnet sind, erfolgreich bearbeiten müssen.

Anmeldung

Nach der Zulassung durch den Themensteller erhalten sie Zugang zum Git-Repository, in dem die Anmeldung zum Studierendenprojekt stattfindet. Erst nachdem Sie diese Prozedur durchlaufen haben wird das Studierendenprojekt beginnen.

In dem entsprechenden Git Repository finden Sie einen Branch mit Ihrem Namen. In diesem wird durch den Betreuer ein Unterordner angelegt, der eine Datei mit einer Codezahl enthält.

Anschließend muss ein Programm implementiert werden (in einer Programmiersprache Ihrer Wahl (kein MATLAB!), bevorzugt natürlich in der Programmiersprache in der das Projekt durchgeführt werden soll), das eine TCP Verbindung zu dem Server `cruncher.clients.ldv.ei.tum.de` auf Port `6666` aufbaut. Der Server wird Sie mit einem kleinen Begrüßungstext begrüßen und erwartet anschließend die Codezahl als 64bit Unsigned Integer, Big Endian kodiert. Als Rückgabe erhalten Sie anschließend ein Codewort in folgendem Format:

```
+-------+-----------------------+
| Länge | Codewort (ascii, A-Y) |
+-------+-----------------------+
```

Die Länge der Rückgabe wird in Bytes angegeben (64bit unsigned Integer, Little Endian) und jeder Buchstabe als 8 Bit Zeichen in ASCII kodiert. Die endgültige Lösung erhalten Sie, indem sie dieses Codewort in eine sogenannte "Telefonzahl" transformieren, wobei Sie eine Kodierung wie nach den Tasten eines Telefon vornehmen müssen. Gehen Sie dabei nach folgender Tabelle vor:

```
2   ABC
3   DEF
4   GHI
5   JKL
6   MNO
7   PRS
8   TUV
9   WXY
```

Die Zahl, die sie als Ergebnis erhalten speichern Sie in das Anmelderepository in den gleichen Ordner, aus dem Sie Ihre Codezahl entnommen haben in eine neue Datei mit dem Namen `ergebnis`.

Den Quellcode, den Sie für Ihre Anmeldung implementiert haben, packen sie in eine `.zip` Datei und laden diesen anschließend unter `https://deepsea.ldv.ei.tum.de/u/d/d6a0e0ab12/` hoch. Die Datei

muss dabei nach dem Schema `vorname_nachname_matrikelnummer_jahr_monat_tag.zip` benannt sein (4 Stellen Jahr, jeweils 2 Stellen Monat und Tag).

Ein Beispiel, mit dem sie Ihren Code testen können ist `723401728380766730`. Dafür müssen Sie als Ergebnis `2344792846588` erhalten.

Vorgehen

Nach erfolgreicher Anmeldung gilt es zunächst einen Projektplan zu erstellen. Die Vorlage für einen solchen kann im Git, in dem auch die Anmeldung erfolgt ist, bezogen werden. Die Vorlage ist für das Textsatzsystem Latex und es empfiehlt sich Kenntnisse in diesem anzueignen, da es insbesondere später im Studium sehr angenehm ist darin auch längere Abschlussarbeiten zu verfassen. Der Projektplan sollte maximal 3 (minimal 2, Deckblatt wird nicht mitgezählt) Seiten umfassen und muss einen Zeitplan enthalten.

Wird das Projekt begonnen, so wird für jeden Milestone (also jedes Studierendenprojekt) ein neuer Branch erzeugt. Alle Änderungen im Rahmen eines Projekts werden dann in diesem Branch vorgenommen. Zum Abschluss muss dieser Branch ohne manuelle Modifikationen wieder in den "Master" Branch gemerged werden können.

Jedes Issue, das am betreffenden Tag bearbeitet wird muss mit einem Tageskommentar versehen werden. Dabei sollte kurz zusammengefasst beschrieben werden, was bearbeitet wurde und sehr kurz die Ergebnisse und/oder Probleme aufsummiert werden.

Zum Abschluss des Projekts muss ein beispielsweise für Ingenieurpraxen ein Praktikumsbericht verfasst werden (für Forschungspraxen analog dazu ein mehrseitiges Paper im IEEE Stil). Jetzt zahlt es sich aus, dass für die bearbeiteten Issues Tageskommentare verfasst wurden. Orientieren Sie sich daran und arbeiten Sie ihre Erkenntnisse und Ergebnisse in einen Projektbericht um. Endgültig abgeschlossen werden die Studierendenprojekte wenn alle Issues erfolgreich geschlossen werden konnten, ein Projektbericht vorliegt und gegebenenfalls der Abschlussvortrag gehalten wurde.

Evaluation und Dokumentation

Während der Arbeit sollte stets sorgfältig an der zugehörigen Dokumentation gearbeitet werden. Dies ist sehr wichtig, um nachfolgenden Studierenden einen leichten Einstieg in das Projekt zu ermöglichen. Zudem helfen Ihnen die bereits erwähnten mindestens täglich zu erstellenden Issue-Kommentare sowie die Dokumentation beim späteren Abschlussbericht. Hier kann gerne teilweise Text aus den Kommentaren oder der Dokumentation übernommen werden.

Die Arbeit, die Sie geleistet haben, kann dann einfach anhand der Commits und der Kommentare an den zugehörigen Issues nachvollzogen werden. So haben auch Sie einen guten Überblick über den Fortschritt Ihres Projektes.

Die Abschlussdokumentation erfolgt in einem Projektbericht, der im Latex Textsatzsystem verfasst werden sollte. Die zugehörigen `.tex` Dateien sind in einem entsprechenden Unterordner im Git mitzuführen. Hierbei ist besonders darauf zu achten, dass keine Hilfsdateien und automatisch generierten Dateien mit eingecheckt werden.

Git Verhaltensregeln

Beim Umgang mit Git gibt es einige Grundregeln zu beachten, außerdem sollten Sie sich einen der vielfach verfügbaren Guides zu Herzen nehmen, beispielsweise

- `http://www.git-tower.com/learn/ebook/command-line/appendix/best-practices` oder
- `https://sethrobertson.github.io/GitBestPractices/`.

Die wichtigsten Grundregeln sind:

1. Jeder arbeitet in seinem eigenen Branch, der nach Fertigstellung des Projektes dann in den "Master" Branch gemerged wird. Dieses Merging muss ohne Konflikte durchgeführt werden können.

2. Eingecheckt werden sollten auf keinen Fall: Temporäre Dateien, große Binärdateien, Kompilierte Dateien. Für einen einfachen Umgang können unerwünschte Dateien mit Hilfe von `.gitignore` von der Verwaltung ausgeschlossen werden.

3. Es sollte möglichst oft committed werden, um Fehler leichter wieder rückgängig machen zu können. Auf den Server sollte meist mindestens am Ende des Tages gepusht werden, um ein zusätzliches Backup zu haben. Gemerged wird aber erst, nachdem alles nachgewiesenermaßen funktioniert und getestet wurde.