Fallstudie Entwicklungswerkzeuge

Ausarbeitung

über das Thema

GIT Versionsverwaltungssystem

Autor: Vedad Hamamdzic
email@email.de

Prüfer: Paul Lajer

Abgabedatum: 18.11.2014

I Zusammenfassung

Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet. Lorem ipsum dolor sit amet, consetetur sadipscing elitr, sed diam nonumy eirmod tempor invidunt ut labore et dolore magna aliquyam erat, sed diam voluptua. At vero eos et accusam et justo duo dolores et ea rebum. Stet clita kasd gubergren, no sea takimata sanctus est Lorem ipsum dolor sit amet.

Abstract

Das ganze auf Englisch.

II Inhaltsverzeichnis

III Abbildungsverzeichnis

IV Tabellenverzeichnis

V Listing-Verzeichnis

VI Abkürzungsverzeichnis

1 GIT

1.1 Was ist ein Versionskontrollsystem

GIT ist ein Versionsverwaltungssystem, soviel wissen wir. Doch was ist das und was macht es im Detail? Ein Versionsverwaltungssystem ist ein System, welches Änderungen an einer Datei oder eine Reihe von Dateien protokolliert, so dass bestimmte Versionen später wieder aufrufbar sind.[1] Um Problemen entgegenzuwirken die eine amateurhafte Methoden der Versionsverwaltung mit sich bringen, wie z.B das ständige kopieren neuer Versionen in ein Verzeichnis, hierfür wurden diese Systeme entwickelt.Dabei unterscheidet man 3 Arten von Systemen. Der wesentlichste Unterschied, besteht darin wie und wo die Daten gehalten werden.

1.1.1 Lokale Versionskontrollsysteme

Von Lokalen Versionskontrollsystemen spricht man, wenn die Daten auf dem Lokalen System vorliegen (siehe Abbildung 1). Dabei werden die Dateien in einer Version Database (Repository) gehalten. Nach jedem Checkout wird automatisch eine neue Version im Repository erstellt. Somit entgeht man der Gefahr, durch das oben erwähnte Kopieren in andere Verzeichnisse, eine der Versionen zu überschreiben, da man vergessen hat die Datei umzubenennen. Natürlich ist diese Variante der Versionskontrolle, für große Projekte die im Team bearbeitet werden eher destruktiv. Ein Beispiel für Lokale Systeme ist RCS(Revision Control System). Für Teamwork, eignen sich eher die anderen beiden Architekturen.

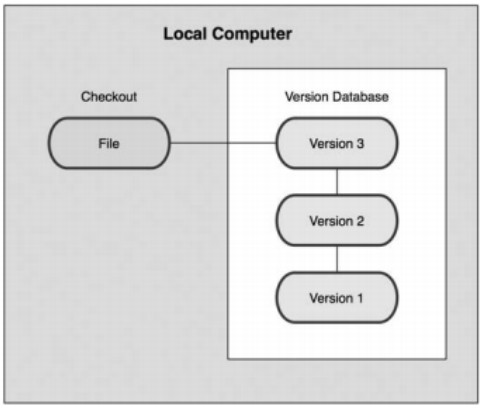

Abbildung 1: Lokale Architektur [Cha09][2]

[1] [Cha09] Seite 1 Zeile 1

1.1.2 Zentralisierte Versionskontrollsysteme

Bei zentralisierten Versionskontrollsystemen wird die Versionierung nicht lokal vorgenommen. Die Entwickler haben einen Zentralen Punkt(Abbildung 2) , einen Server und dort befindet sich der Quellcode des Projektes in einem Reposytory, zu deutsch Lager. Der unterschied zu einfachen Lokalen Systemen ist nun Offensichtlich, man braucht zumindest ein Netzwerk, um solche Systeme zu nutzen. Ein sehr beliebtes zentralisiertes System ist Subversion. Ein weiterer Vorteil gegenüber der lokalen Versionnierung, besteht darin das gemeinsames Arbeiten an einem Projekt möglich ist und bei Verwendung eines Servers der Online erreichbar ist, kann das Arbeiten auch ohne Ortsbindung ablaufen. Doch dieser Vorteil der Ortsungebundenheit, bietet einen enormen „Single Point of Failure“, denn wenn der Server ausfällt ist man nicht in der Lage seiner Arbeit nachzugehen.

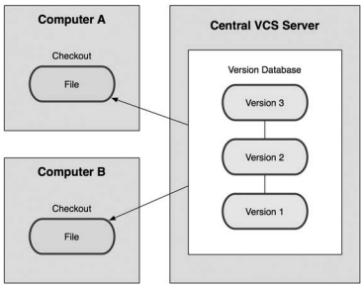

Abbildung 2: Zentralisierte Architektur[3]

1.1.3 Verteilte Versionskontrollsysteme

GIT gehört zu den verteilten Systemen, der Unterschied zu den Varianten davor ist das sie beides können. Einer Art hybride Lösung, man ist in der Lage Lokal zu Versionieren aber auch im Netzwerk Versionen anderen zur Verfügung zu stellen (Abbildung 3). Jeder kann als Server fungieren und somit wird der „Single Point of Failure“ eliminiert den zentralisierte Systeme haben. In der Praxis ist aber eher üblich, dass man einen Server nutzt vor allem bei Teamarbeiten. Wenn dieser jedoch ausfällt ist man in der Lage, weiter seine Arbeit zu verrichten.

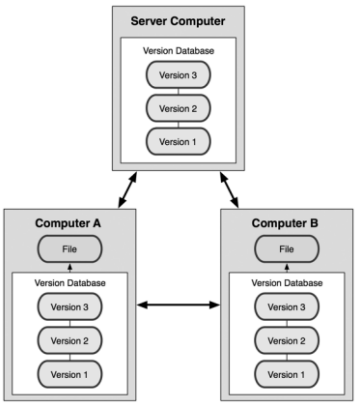

Abbildung 3: Verteilte Architektur[4]

1.2 GIT Historie

Im Jahre 2005 ist es zu Unstimmigkeiten gekommen, zwischen der Entwicklercommunity von Linux und dem Anbieter des proprietären BitKeeper-Systems, dass vorher kostenfrei genutzt wurde. Die Linux-Kernel-Entwickler mussten sich etwas einfallen lassen. Deswegen begann Linus Torvalds im April 2005 mit der Entwicklung von GIT und präsentierte auch sehr schnell die erste Version. Git baute auf den Erfahrungen mit BitKeeper auf, doch die Hauptziele des neuen Systems waren[5]:

- Geschwindigkeit
- Einfaches Design
- Gute Unterstützung von nicht-linearer Entwicklung (tausende paralleler verschiedener Verzweigungen der Versionen)
- Vollständig verteilt
- Fähig, große Projekte wie den Linux Kernel effektiv zu verwalten

Durch die kontinuierliche Weiterentwicklung des Systems und die Benutzerfreundlichkeit wurde Git zu einen sehr beliebten Tool. Ein großen Einfluss auf den Erfolg von Git hat auch die Social Coding Plattform GitHub auf der man viele Open Source Projekte findet wie z.B :

- Der Linux Kernel[6]
- Ruby on Rails[7]
- Die Javascript Bibliothek JQuery [8]
- Das CMS Joomla[9]

Das sind natürlich nicht alle Open Source Projekte die GIT in Verbindung mit GitHub nutzen, aber einige bekannte die sich für Git entschieden haben. Der Dienst, den GitHub bereitstellt ist kostenfrei, doch nur unter der Bedingung das die Projekte öffentlich zugänglich sind. Des Weiteren gibt es Optional wählbare Services, die gegen Bezahlung verfügbar sind, aber es gibt auch eine Enterprise Version, die für Firmen interessant sein kann.

[5] [Cha09] Seite 5
[6] [Tor]
[7] [Rub]
[8] [Jqu]
[9] [joo]

2 GIT Grundlagen

Um grundlegende Funktionen von GIT zu nutzen, ist es unumgänglich gewisse Begriffe zu kennen. Elementar hingegen, ist der Umgang mit der Konsole des jeweiligen Systems. Es existieren einige plugins für Entwicklungsumgebungen, wie z.B. Eclipse die mit einem GUI ausgestattet sind. Jedoch sind diese Plugins meistens nicht soweit entwickelt, um den kompletten Funktionsumfang des Systems bedienbar zu machen, weswegen meine Erläuterungen zum Git System, sich auf Linux als Betriebssysteme beziehen und nur mit der Konsole zu bedienen sind.

2.1 Begriffe die man kennen sollte

Bevor man mit Versionierungssystemen arbeitet, sollte man einige Begriffe kennenlernen.

2.2 Repository

Der Begriff Repository (Englisch für Lager), kommt aus dem Lateinischen Repositorium. Eine Repository ist eine spezielle Datenbank, zur systematischen Ablage von Modellen und deren Bestandteilen, das Herz der Datenhaltung eines Versionskontrollsystems. Grundlegende Funktion ist die Speicherung und das Abrufen von gespeichertem Inhalt samt aller Bestandteile wie z.B. Bilder. [10]

2.3 Clone

Ein Clone im Git Kontext, ist äquivalent zu dem begriff in der Biologie. Daher ist eine exakte Kopie von etwas existierendem, in diesem Fall der Repository. Im Fall Git lässt sich ein Clone auch durch verschiedene Protokolle umsetzten z.B. git:// ein eigens Protokoll oder auch das https:// Protokoll. Weitere Erläuterungen dazu folgen später.

2.4 Commit

Zu deutsch "übergeben", wenn man also eine Änderung im Arbeitsverzeichnis vornimmt, wird diese getrackt(verfolgt). Um diese zu bestätigen, bzw. an Git zu übergeben, ist ein commit erforderlich.

2.5 Branch

Zu Deutsch SZweigïst eine Gabelung des Quellcodes. Aus verschiedenen Gründen, kann es erforderlich sein eine Version des Quellcodes vom Original abzuzweigen, um ggf. eine neue Funktion zu implementieren. Dies läuft dann parallel zur Entwicklung des Originalcodes. Der initiale Commit, wird auch als Master bezeichnet

[10] [Ley]