CPR SW

Wintersemsester Git/GitHub

Organisatorisches

- WS: Git/Github / SS: Docker
- Eigene Laptops (Installationsrechte)
- Leistungsbeurteilung Mitarbeit/Übungsdurchführung

Einführung

Was ist Versionsverwaltung

Unterschiedliche Versionen von Dateien/Dokumenten erfassen

Wesentliche Aufgaben:

- Archivierung der Versionen
- Entwicklungsprozess nachvollziehbar machen
- Veränderungen mitprotokollieren (Wer, Wann, Was)
- Entscheidungsgrundlage – was kommt in nächsten Release
- Wiederherstellung einer früheren Version
- Koordination bei mehreren Team-Mitgliedern

Wie funktioniert die Versionsverwaltung

- Nur Änderungen zu Vorgängerversion werden gespeichert (Speicherplatz)
- Zeitpunkt und Autor
- Kommentare bei jedem Commit
- Rechteverwaltung (Bearbeitung, Konfliktauflösung, Bestimmung der Hauptversion)

Organisationsform

Lokal

- Erste Systeme
- Nur auf eigenem Rechner

Quelle: https://pixabay.com/de/photos/laptop-arbeitsplatz-tisch-336369/

https://pixabay.com/de/photos/technologie-server-1587673/

Zentral

- Server
- Erleichtert Zusammenarbeit
- Rechtevergabe!
- Zusatzkosten

+

Verteilt

- Sowohl Lokal als auch Zentral
- Offline möglich
- Abgleich mit Team (Konfliktlösung)

Geschichte von Git

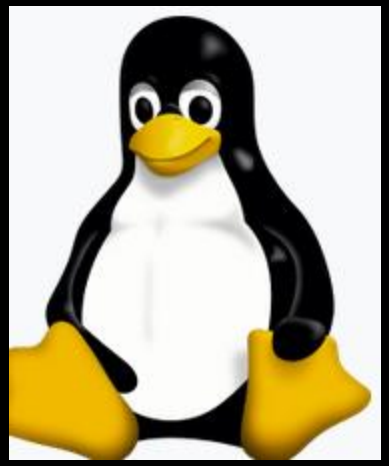
?

Geschichte von Git

- Verschiedene Versionsverwaltungssysteme
- Vorläufer in 60er Jahren
- 1972 SCCS (Source Code Control System) - heute nicht mehr bedeutend

Git ist relativ jung - Erste Version:

2005



https://de.wikipedia.org/wiki/Linux

Geschichte von Git

Git entstand im Rahmen der Entwicklung eines bekannten BS

Rätsel

- Sehr weit verbreitet heutzutage
- v.a. für den Betrieb von Servern aber auch Embedded Systems
- Zu Beginn der 90er Jahre entwickelt
- **OS kostenlos verwendbar**

Das gesuchte BS heißt:

LINUX

Router, Android-Geräte, Smart-TV, CERN-Teilchenbeschleuniger, RaspPi

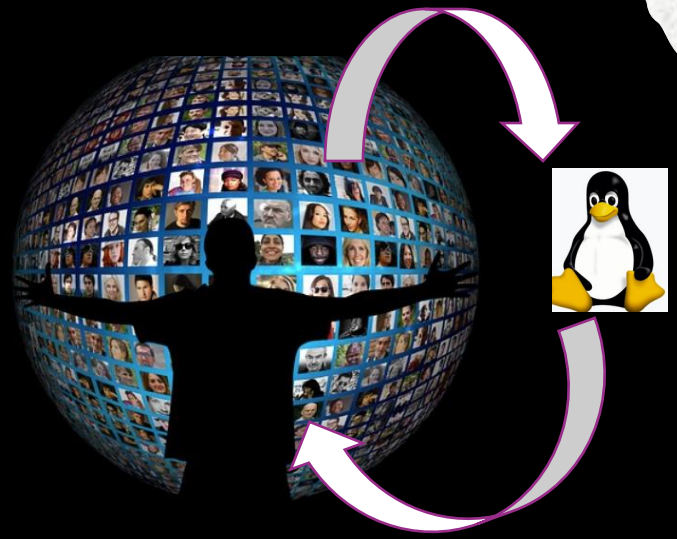
https://pixabay.com/de/photos/dollar-w%c3%a4hrung-geld-us-dollar-499481/

Geschichte von Git

Was? Software, die nix kostet?

Open Source Software

- Quellcode ist offen und frei zugänglich
- Verbesserter/Geänderter Quellcode darf auch weitergegeben werden (Achtung Lizenz – meist wieder Open Source)
- Entwicklercommunity – Mitarbeit auf freiwilliger Basis UND ohne Gehalt! (ev. Indirekte Entlohnung)
- Motive:
  - Altruistisch (Uneigennützigkeit)
  - Entwicklungskosten teilen
  - Marktanteile gewinnen
- Beispiele: Mozilla Firefox, Open Office, Gimp, Tesla (Stichwort Ladestationen)



Quelle https://pixabay.com/de/photos/umarmung-silhouette-menschen-2709635/

Geschichte von Git

Was hat Linux mit Git zu tun

- Viele Entwickler an Linux beteiligt
  - Firmen, die auf Linux-Kernel bauen
  - Universitäten und Forschungseinrichtungen
  - Großteil jedoch: freiwillige Entwickler, unentgeltlich

- -> SEHR umfangreiches Projekt
  - Erfordert Versionsverwaltungssystem
  - Bitkeeper -> für Open Source-Projekte kostenlos
  - 2005: Ausnahmeregelung aufgehoben -> Lizenzgebühren
  - -> erhebliche Kosten für jeden (freiw.) Entwickler!

Lösung?

Geschichte von Git

https://de.wikipedia.org/wiki/Linus_Torvalds#/media/Datei:LinuxCon_Europe_Linus_Torvalds_03_(cropped).jpg

Was hat Linux mit Git zu tun

Lösung? -> Eigene Source-Code-Verwaltung

"I'm an egotistical bastard, and I name all my projects after myself.
First 'Linux', now 'Git'." (L. Torvalds)

git (dt): Blödmann, Idiot

- kurz
- Einfach auszusprechen
- Schnell zu tippen
- Bisher nicht in Verwendung

Geschichte von Git

https://de.wikipedia.org/wiki/Linus_Torvalds#/media/Datei:LinuxCon_Europe_Linus_Torvalds_03_(cropped).jpg

Was hat Linux mit Git zu tun

Lösung? -> Eigene Source-Code-Verwaltung

Eigenschaften

- Vollkommen neue Lösung (Arbeitsabläufe angelehnt an Bitkeeper)
- Ebenfalls Open-Source -> kostenlos!
- Verteiltes System

Wesentliche Eigenschaften

Quelle: https://pixabay.com/de/photos/eiche-baum-riesig-alt-charleston-2018822/

Entwicklungs-Zweige (Branches/Forks)

- Für experimentelle Entwicklungen
- Verschiedene Versionen einer SW
- Ganz neue Projekte abspalten

Bei Git sehr effizient umgesetzt

Branch:

- Innerhalb gleichem Repository ("Ablage-Ort")
- Interne Strukturierung/abschließend Merge (Verschmelzung)

Quelle: https://pixabay.com/de/photos/werkzeug-garten-forke-mistgabel-4958040/

Fork ("Gabel"):

- Vollständige Abspaltung
- Urspr. Version nur Ausgangspunkt
- Meist in Open-Source-Projekten (Rechteübertragung!)

Eigenes Protokoll

Quelle: https://pixabay.com/de/photos/schafe-bl%c3%b6ken-kommunikation-2372148/

Protokoll = "Regeln für den Datenaustausch"

- Für die Synchronisation Server-Repo <- lokales Repo

Bei Git sehr effizient umgesetzt

- Schnelles Laden – auch umfangreicher Projekte (nur von Server -> Lokal)

Quelle: https://pixabay.com/de/photos/babuschka-matroschka-puppe-holz-1666132/

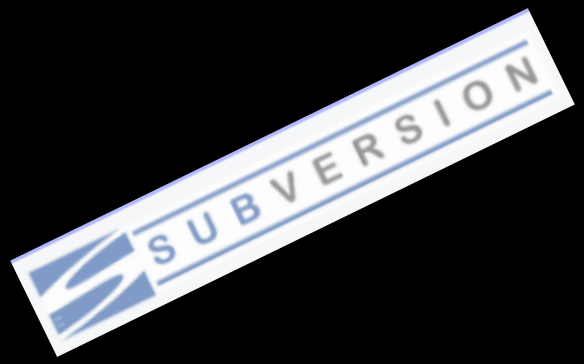
Hohe Sicherheit der Versions-geschichte

Verhinderung Manipulation der früheren Versionen!

Bei Git sehr hoher Fokus

Mittels Hash-Wert:

- Berechnung eines "eindeutigen" Werts und mitspeichern in der Version
- Jede Nachfolge-Version enthält Vorgänger + dessen Hashwert
- Änderung der Version -> Unstimmigkeit mit dessen Hashwert (Git bemerkt das)
-



https://de.wikipedia.org/wiki/Apache_Subversion#/media/Datei:Subversion_logo.svg

Inter-operabilität

…mit anderen Versionsverwaltungs-systemen

Git: Vermeidung hoher Einarbeitung

Dafür stehen versch. Hilfsprogramme zur Verfügung
Z.b. für
- Subversion
- CVS, GNU arch (nicht mehr weiterentwickelt)
- ...
•