Version Control mit Git

Links:

- Git - About Version Control
- Cloning a repository - GitHub Docs
- GitHub Git Cheat Sheet

Was ist Version Control

Versionskontrolle ist ein System, das alle Änderungen und Modifikationen an Dateien aufzeichnet, um den Verlauf dieser Änderungen nachvollziehbar zu machen. Es ermöglicht Entwicklern, den gesamten Änderungsverlauf eines Projekts einzusehen und bei Bedarf zu früheren Versionen zurückzukehren. Dieses System spielt eine zentrale Rolle in der Softwareentwicklung und insbesondere in modernen Entwicklungspraktiken wie DevOps.

Vorteile der Versionskontrolle:

- Revisionshistorie: Jeder Schritt und jede Änderung im Projekt wird dokumentiert und kann jederzeit nachvollzogen werden.
- Identitätsnachverfolgung: Änderungen können einzelnen Entwicklern zugeordnet werden, was die Verantwortung und Transparenz erhöht.
- Zusammenarbeit: Mehrere Entwickler können gleichzeitig an einem Projekt arbeiten, ohne sich gegenseitig zu behindern.
- Automatisierung: Viele Prozesse wie das Erstellen und Testen von Software können automatisiert und in den Versionskontrollprozess integriert werden.
- Effizienz: Projekte lassen sich schneller und organisierter entwickeln und verwalten, wodurch Fehler minimiert und die Qualität der Software gesichert werden.

Beispiele für den Einsatz von Version Control

1. Webentwicklung: Entwickler nutzen Versionskontrolle, um gleichzeitig an verschiedenen Teilen einer Website zu arbeiten. Änderungen können sicher zusammengeführt und bei Bedarf zurückgesetzt werden.

2. Data Science und Datenanalyse: Fortschritt von Analysen und Modellen nachzuverfolgen. Es werden Änderungen dokumentieren. Falls neue Ansätze nicht erfolgreich sind, kann der Analyst jederzeit zu früheren Versionen zurückkehren und von dort neu beginnen.
3. Dokumentenmanagement: In Unternehmen wird Versionskontrolle auch bei der Bearbeitung wichtiger Dokumente eingesetzt. Mehrere Personen können an einem Bericht oder Vertrag arbeiten, ohne den Überblick über Änderungen zu verlieren. Alle Versionen werden gespeichert, sodass frühere Fassungen leicht wiederhergestellt werden können.

Verschiedene Typen von Versionskontroll-Systemen

CVCS (centralized version control systems)

Bei einem zentralen Versionskontrollsystem (CVCS) befindet sich die vollständige Versionshistorie des Codebestands auf einem zentralen Server. Entwickler müssen den Code vom Server auf ihren lokalen Rechner herunterladen, um daran zu arbeiten. Änderungen, die von Entwicklern vorgenommen werden, müssen zurück auf den Server übertragen werden, damit andere sie sehen können.

Beispiele für CVCS:

- Subversion (SVN)
- Perforce

Vorteile:

- Einfach zu erlernen und zu verwenden.
- Klare Zugriffs- und Rechteverwaltung.

Nachteile:

- Abhängigkeit von der Serververfügbarkeit
- Erfordert ständige Verbindung zum Server

DVCS (distributed version control systems)

Jeder Clien ist ein eigenständiger „Mini-Server“. Jeder Entwickler hat eine vollständige Kopie der gesamten Versionshistorie auf seinem lokalen Rechner. Die Zusammenarbeit erfolgt durch das Abrufen (Pull) und

Hochladen (Push) von Änderungen zwischen den lokalen Repositories und dem Server.

Beispiele für DVCS:

- Git
- Mercurial

Vorteile:

- Möglichkeit offline zu arbeiten
- Bessere Leistung und Geschwindigkeit, da viele Operationen lokal ausgeführt werden können.
- Dezentrale Struktur: Kein Single Point of Failure
- parallele Entwicklungsarbeit

Nachteile:

- Komplexere Lernkurve durch die zusätzliche Flexibilität.
- Weniger strikte zentrale Kontrolle, was zu möglichen Konflikten bei der Zusammenarbeit führen kann.

Development Environments

Bevor neue Features oder Änderungen in einer Anwendung veröffentlicht werden, müssen diese gründlich getestet werden, um Fehler und Probleme zu vermeiden. Dazu nutzen Entwicklungsteams verschiedene Umgebungen wie Entwicklungs-, Test- und Staging-Umgebungen.

Staging

Die Staging-Umgebung ist eine Nachbildung der Live-Umgebung. Hier kann der Code unter realistischen Bedingungen getestet werden, bevor er live geht.

In der Staging-Umgebung können neue Features getestet und so mögliche Probleme aufgedeckt werden, die in der Produktion vermieden werden sollen.

Vorteile der Staging-Umgebung:

- Testen neuer Features und ihrer Auswirkungen auf das Gesamtsystem.

- Durchführung von Tests wie Unit-, Integrations- und Performance-Tests unter realistischen Bedingungen.
- Sicheres Testen von Datenmigrationen und Konfigurationsänderungen.

Production

Die Produktionsumgebung ist die live geschaltete Umgebung, in der die Anwendung für Nutzer verfügbar ist.

Jegliche Fehler in dieser Umgebung können schwerwiegende Auswirkungen haben. Daher sollten alle Änderungen, die in die Produktion gehen, vorher gründlich in der Staging-Umgebung getestet worden sein.

Risiken in der Produktionsumgebung:

- Ausfallzeiten: Beeinträchtigen den Zugang der Nutzer und können Umsatzverluste verursachen.
- Sicherheitslücken: Müssen durch regelmäßige Updates und Patches geschlossen werden.
- Reputation: Probleme in der Produktion können das Vertrauen der Nutzer in das Unternehmen schädigen.