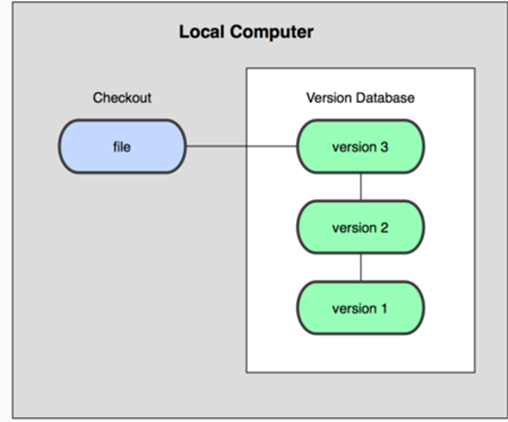
**Zweck Versionskontrolle :**

Versionskontrollsysteme(VCS) verwalten Änderungen an einer Datei oder einer Anzahl von Dateien im Lauf der Zeit, damit man zu jeden Zeitpunkt auf Versionen und Änderungen zugreifen kann.

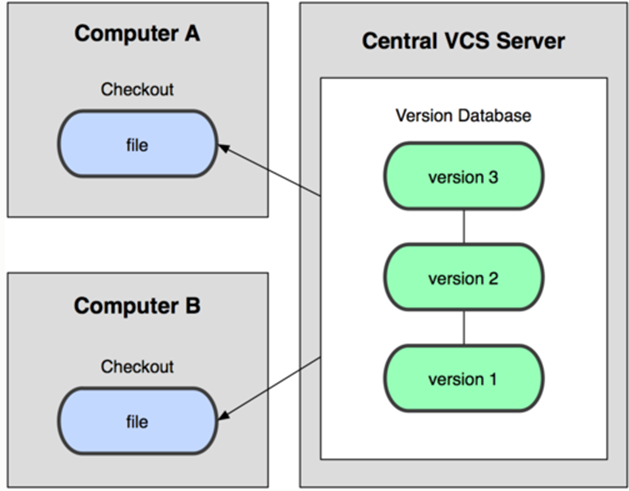
Es existiert 3 Arten von Versionskontrollsystemen:

* Lokale Versionskontrollsystem
* Zentralisierte Versionskontrollsysteme
* Verteiltes Versionskontrollsystem.

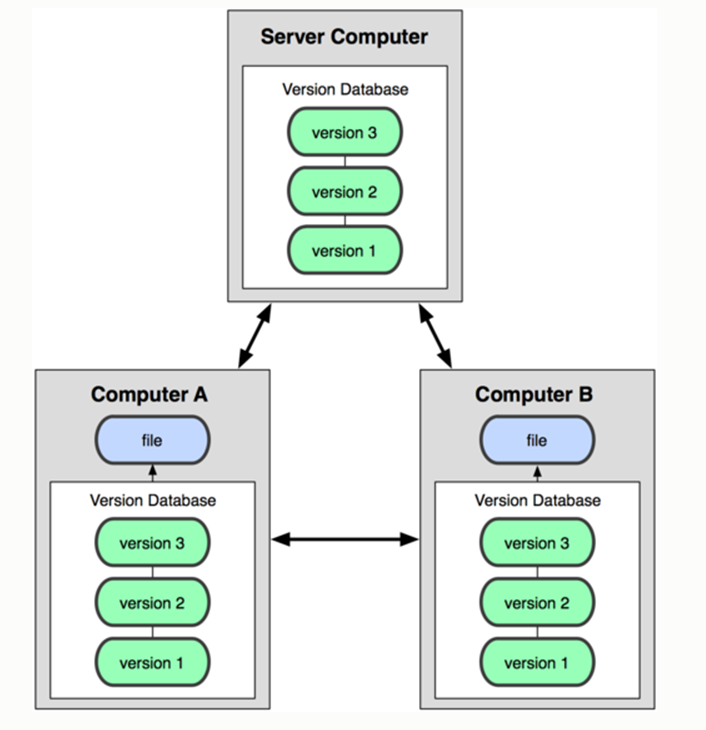
1. Lokale Versionskontrollsystem :



1. Zentralisierte Versionskontrollsysteme (CVCS)



1. Verteiltes Versionskontrollsystem (DVCS)

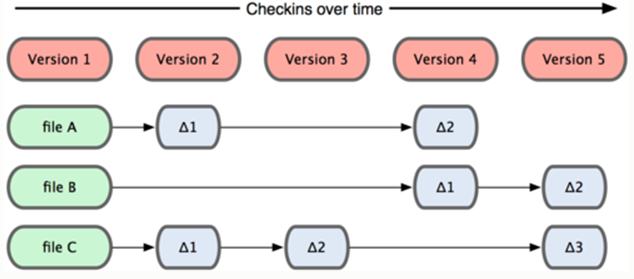


* Ziele des neuen Systems waren:
* Geschwindigkeit .
* Einfaches Design
* Gute Unterstützung von nicht-linearer Entwicklung
* Vollständige verteilt

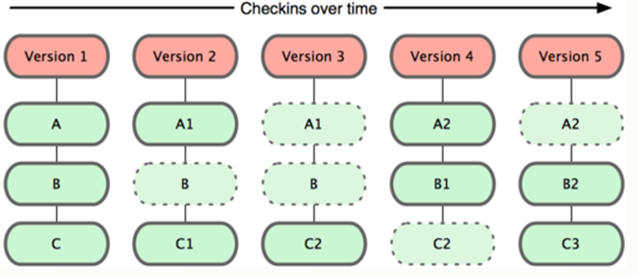
**Git Grundlagen**

* Snapshots , nicht Diffs

Der Hauptunterschied zwischen Git und anderen Versionskontrollsystemen besteht in der Betrachtung der Daten von Git .Die Meisten anderen Systeme speichern Information als eine fortlaufende Liste von Änderungen an Dateien („Diffs“).



Git betrachtet Daten nicht in dieser Art, sondern als Snapshots eines Mini-Dateisystems. Jedes Mal, wenn man committest(Speicherung der Status-Änderungen) sichert Git den Zustand sämtlicher Dateien in dieser Zeit (Snapshot) und speichert eine Referenz auf diesem Snapshot. Zur effizient und schnell Möglichkeit kopiert Git unveränderte Dateien nicht, sondern legt nur eine Verknüpfung zu der vorherigen Version der Datei an.



Operation in Lokal:

Die meisten Operationen bei Git brauchen lediglich die lokalen Dateien und Ressourcen auf deinem Rechner zum Funktionieren. Man benötigt im Netzwerk keine Informationen von einem anderen Rechner .Bei CVCS sind die Operationen nicht lokale, das bringt für Nachteile einen große Latency Overhead(Wartezeiten).

Zum Beispiel:

Wenn man Lust seinen Projekt zu Hause weiterarbeiten will, kann man ohne Probleme seine Arbeit committen und später auf den Server hochladen bei wiederherstellten Netzwerksverbindung.

Gewährleistung der Integrität in Git:

In Git werden Änderungen in Checksummen umgerechnet, bevor sie gespeichert werden. Dann wird dies mit dieser Checksumme referenziert. Mit dieser Integrität wäre es unmöglich eine Änderung der Inhalte von Dateien oder Repository, ohne dass Git das mitbekommt.

Der Mechanismus, der Git benutzt, zur Erstellung dieser Checksummen heißt SHA-1 Hash. Diese Checksumme besteht aus 40 hexadezimalen Zeichen lange Zeichenkette. Ein SHA-1 Hash sieht wie folgt aus:

24b9da6552252987aa493b52f8696cd6d3b00373

Die Speicherung und das Referenzieren der Git-Informationen über Dateien in der Datenbank werden nicht nach ihren Dateinamen sondern nach den Hash Werten ihrer Inhalte.

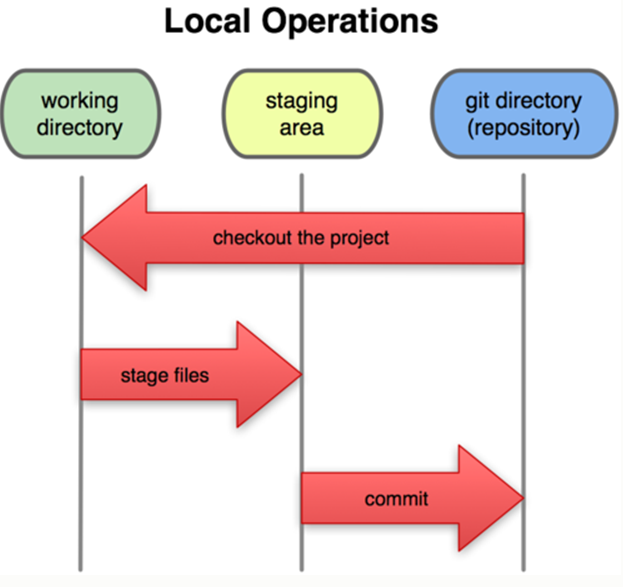
Die drei Zustände von Git:

Git definiert drei Haupt-Zustände, in denen sich eine Datei befinden kann:

* Commited: Die Daten in der lokalen Datenbank gesichert sind.
* Modified(geändert): Die Daten werden geändert, aber noch nicht committed.
* Staged(vorgemerkt): Die geänderten Daten werden in gegenwärtigen Zustand für den nächsten Commit vorgemerkt.

Das führt zu den drei Hauptbereichen eines Git Projektes:

* Das Git Verzeichnis
* Das Arbeitsverzeichnis
* Die Staging Area



***Das Git Verzeichnis*** ist der Ort, an dem Git Metadaten und die lokale Datenbank für das Projekt speichert. Dieser Teil wird kopiert, wenn ein Repository von einem anderen Rechner geklont wird.

***Das Arbeitsverzeichnis*** ist ein Checkout einer spezifischen Version des Projekts. Diese Dateien werden aus der komprimierten Datenbank geholt und auf dem Festplatte in einer Form gespeichert, damit man einfach bearbeiten und modifizieren kann.

***Das Staging Area***  auch ***Index*** bezeichnet ist eine Datei im Git Verzeichnis, in der vorgemerkt wird, welche Änderungen dein nächster Commit umfassen soll.

Der grundlegend Git Arbeitsprozess:

1. Bearbeitung der Dateien in Arbeitsverzeichnis.
2. Markierung der Dateien für den nächsten Commit, indem Snapshots zur Staging Area hinzugefügt werden.
3. Anlegen des Commit, wodurch die in der Staging Area vorgemerkten Snapshots dauerhaft im Git Verzeichnis gespeichert werden.

**Git installieren**

Git lässt sich unter verschieden Weisen installieren:

* Installation unter Linux:
* Unter Fedora zum Beispiel kann man yum verwenden:

$ yum install git-core

* Auf einem Debian-basierten System wie Ubuntu steht dir apt-get zur Verfügung.

$ apt-get install git

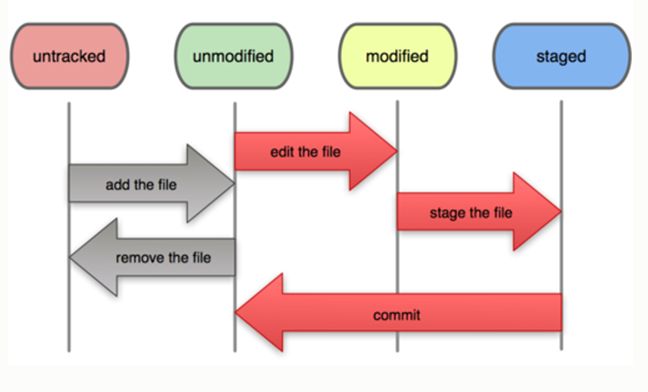
* Installation unter Windows:

Das msysGit Projekt macht die Installation von Git unter Windows einfach. Von der GitHub Webseite:

http://msysgit.github.com/

**Git Konfigurieren**

**File Status LifeCycle**



**Dateien ignorieren**

In einem Projektverzeichnis gibt es eine Reihe von Dateien, die nicht versionieren werden sollen, wie z.B automatisch generierte Dateien: Logdateien, Build-Systemdateien.In diesem Fällen werden diese Dateien in der Datei .gitignore angeben.

z.B touch .gitignore

echo “bin“ >>.gitignore

echo “#” >>.gitignore // ein Kommentar wird ignoriert.

**Ansteigen mit Git**

Das Zweck dieses Abschnitts ist zu herstellen eine Datei, ein lokales Git-Repository und zur Commit die Dateien durch dies Repository.

**Neues repository erstellen:**

cd ~

mkdir ~/my\_repository

cd my\_repository

# a new directory

mkdir my\_directory

# zu initialisieren das Repository

git init

git status Befehl liefert uns eine aktuelle Übersicht über unser Repository:Welche Dateien wurden seit unserem letzten commit bearbeitet und welche Dateien werden von git noch nicht getracked.

Hinzüfügen im Staging Bereich

Bei den meisten anderen Versionskontrollsystemen gibt es zwei Orte für das Speichern der Daten: Die Arbeitskopie (die Verzeichnisse/Dateien , die man gegenwärtige benutzt) und der Datenspeicher (wo die Versionskontrolle entscheidet wie eine Änderung verpackt und gespeichert werden).In Git gibt es eine dritte Option: Die Staging area oder index.es ist ein Ladebock , wo man entscheidet welche Änderungen verschifft werden.

Vor dem Commit der Dateien im Git Repository, soll man zunächst diese Dateien im Staging Bereich hinzufügen mit dem Befehl: git add.