

Versionsverwaltungssysteme

GRUNDLAGEN UND VERSIONSVERWALTUNG MIT GIT

Grundlagen

- Repository
- Commit
- Update
- Merging
- Branching

Versionsverwaltung mit git

- Initialisierung
- Staging
- Checkout / Commit
- Branches
- Push / Pull

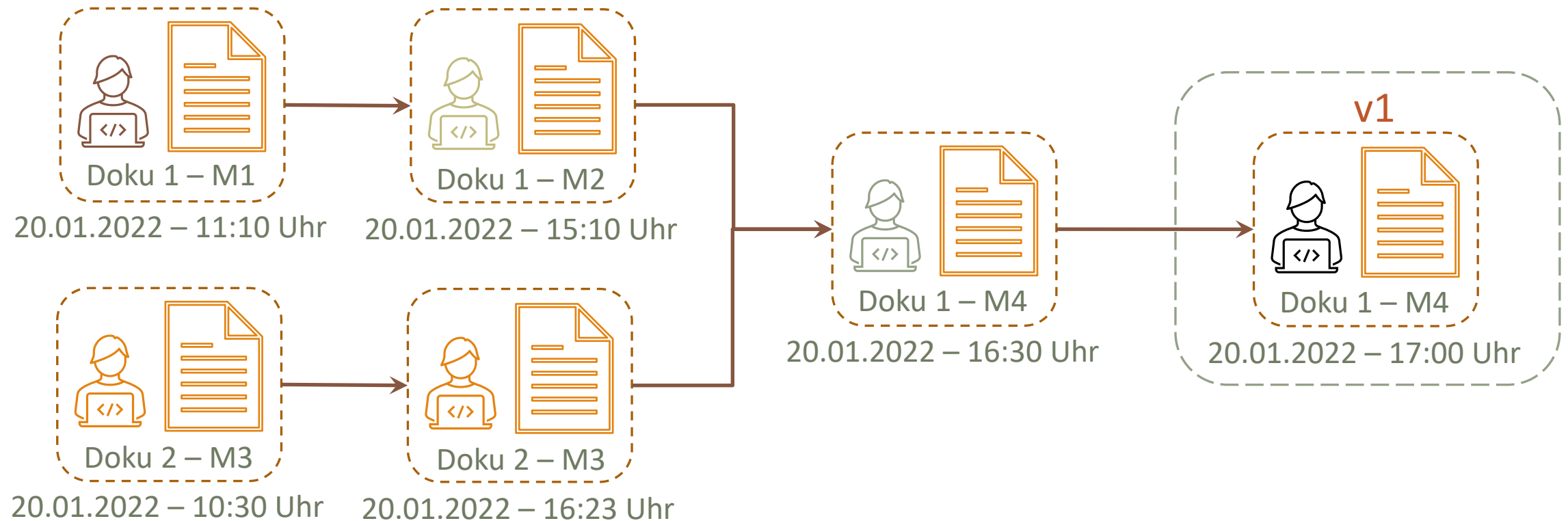
Versionsverwaltung - Definition

- Alle Änderungen von Quelltexten (von anderen Dokumenten auch!) protokollieren
- Versionsverwaltung ist dann Protokollierung
- Protokollierung ist die systematisierte Aufschreibung zu einem spezifischen Zeitpunkt, welche Modifikationen erledigt wurden. (Beispiele zu Modifikationen: Erstellung, Löschen, Hinzufügung, Bearbeiten und usw. eines Quelltexts oder eines anderen Dokumentes)
- Protokollierung ist auch die Zuordnung von Modifikationen den Beteiligten, die die Modifikationen durchgeführt haben.
- Zeitpunkt spielt eine wichtige Rolle für die Erstellung eines Snapshots bzw. einer Version mit bestimmter Versionsnummer. Das ist, eine Version wird zu bestimmten Zeitpunkten genommen
- Vorteil:
 - Die Aufwand der manuellen Protokollierung wird beseitigt
 - Die Verfolgungen von Änderungen und Versionen ist möglich (was jede Änderung war und was die Änderungen bei einer Version waren)
 - gilt als detailliertes Backup
 - Die Möglichkeit zu einer bestimmten Version zurückzugehen und von da wieder weiter zu ändern

Bestandteile eines Versionsverwaltungssystems

- Zeitstemple
- Wer der Verantwortliche ist und seine Rolle
- Die Änderungen der Dokumente und deren Zusammenführung (Merging)

Versionen „Snapshots“



Zusammenfassung

Protokollierung

- Aufnahme der Änderungen und deren Verantwortlichen und Uhrzeiten

Wiederherstellung

- Änderungen sind rückgängig

Gemeinsame Entwicklung

- Ein oder mehrere Entwickler können gemeinsam an einem Teil der zu entwickelnden Software arbeiten

Effiziente Entwicklung

- Ein Projekt kann mehrere Branche enthalten



master ▾


git_vorbereitung / git_vorbereitung /

Go to file

Add file ▾



Samlanar Fixed

8a12e8e 4 days ago  History

..



.settings

Erste Commit

4 days ago



src/main

Fixed

4 days ago



.classpath

Erste Commit

4 days ago



.gitignore

Erste Commit

4 days ago



.project

Erste Commit

4 days ago

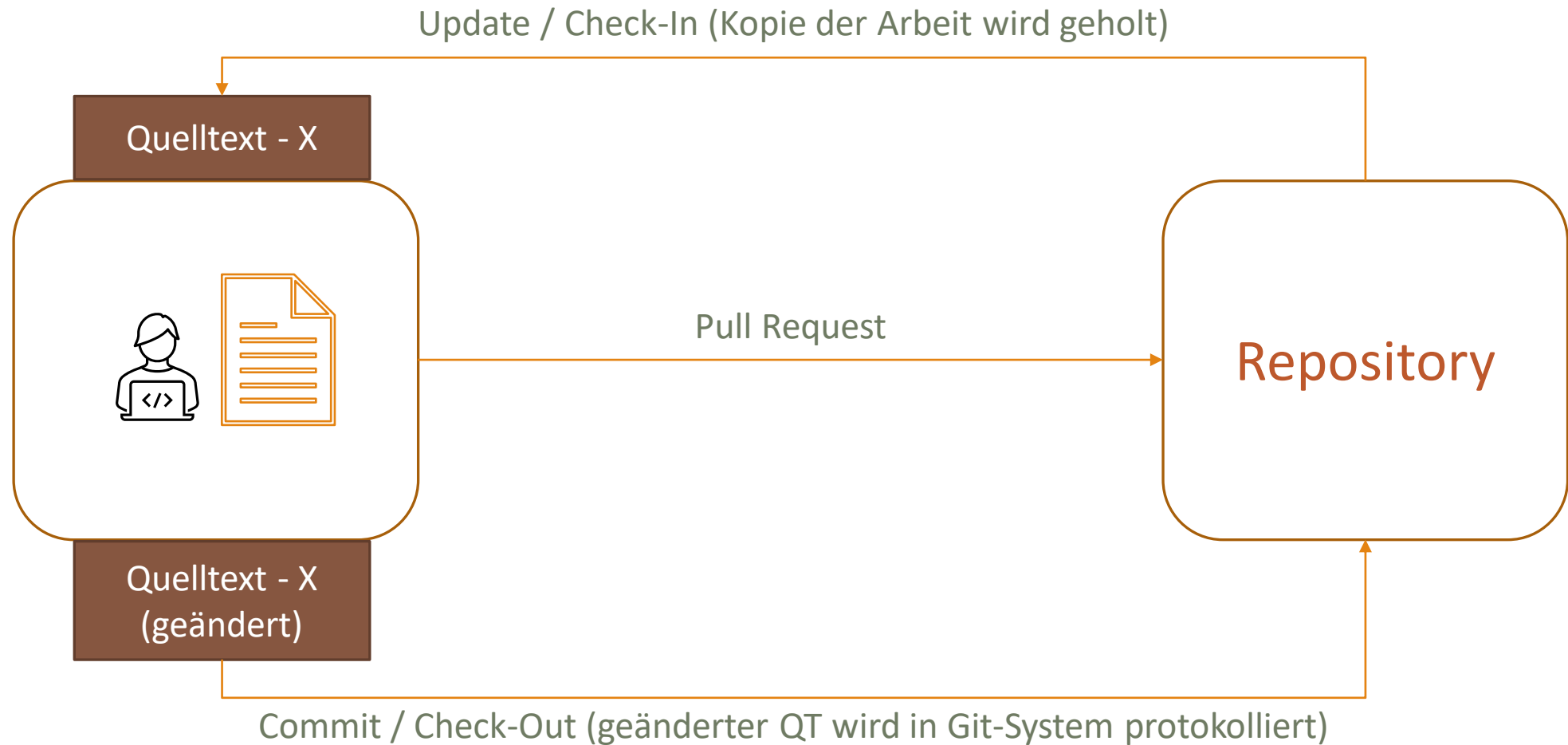
Versionsverwaltungssysteme (Version Control System - VCS)

- Meist verwendetes Versionsverwaltungssystem: git
- Git-System Host bzw. Dienstleister: github.com

Repository

- Ein Repository ist ein verwaltetes Verzeichnis zur Speicherung und Beschreibung digitaler Objekte für ein digitales Archiv.

Repositorien



Update - Szenarien

Lokale Datei und die im VCS haben die gleiche Version

- Keine Änderung im VCS und lokal passiert

Datei im VCS ist aktueller als die lokale Datei

- Die lokale Datei wird mit dem vom VCS zusammengeführt oder ersetzt (merged!)
- bei der Zusammenführung „merging“ können Konflikte auftreten
- oder eine neue lokale Datei wird angelegt

Die lokale Datei ist aktueller als die im VCS

- Nichts passiert, da die lokale Version die letzte Änderungen hat

Commit - Szenarien

Lokale Datei und die im VCS haben die gleiche Version

- Keine Änderung im VCS und lokal passiert

Datei im VCS ist aktueller als die lokale Datei

- Es wird einen Fehler geben
- Lösung: Update (Pull) muss zuerst ausgeführt werden

Die lokale Datei ist aktueller als die im VCS

- Datei im VCS wird überschrieben

Sperrung einer Datei

- Eine Datei kann gesperrt werden, bevor sie bearbeitet wird.
- Freigabe der Sperrung erfolgt erst nach der Änderung
- Gesperrte Datei verhindert alle anderen Commits

Merging

- Ist die Zusammenführung zweier Versionen
- wird ausgeführt, wenn die zu zusammenführenden Versionen keine Konflikt haben
(unterschiedlichen Codezeilen in den zwei Versionen wurden hinzugefügt oder geändert)
- Kann automatisch bzw. manuell ausgeführt
- Die Änderungen können auch graphisch dargestellt werden

Merging mit keiner Konflikt

2 git_vorbereitung/src/main/Programm.java

		↑
		@@ -7,6 +7,8 @@ public static void main(String[] args) {
7	7	System.out.println("Clone-Repository is all set up!");
8	8	
9	9	System.out.println("Good Job!");
10	+	
11	+	System.out.println("Fixed!");
10	12	
11	13	}
12	14	
		↓



master ▾

git_vorbereitung / git_vorbereitung /

Go to file

Add file ▾

Samlanar [main Methode angepasst](#)

fa3fda3 5 minutes ago History

..



.settings

Erste Commit

4 days ago



src/main

main Methode angepasst

5 minutes ago



.classpath

Erste Commit

4 days ago



.gitignore

Erste Commit

4 days ago



.project

Erste Commit

4 days ago

main Methode angepasst

Browse files

master

Samlanar committed 6 minutes ago

1 parent 8a12e8e commit fa3fda3dfefe5f21c6d9bfaf5548aa12e9ef3f53

Showing 1 changed file with 0 additions and 2 deletions.

Split Unified

2 git_vorbereitung/src/main/Programm.java

↑	@@ -7,8 +7,6 @@ public static void main(String[] args) {
7	7 System.out.println("Clone-Repository is all set up!");
8	8
9	9 System.out.println("Good Job!");
10	-
11	- System.out.println("Fixed!");
12	10
13	11 }
14	12
↓	

Merging mit Konflikt

 Push Branch master

Push Confirmation

Confirm following expected push result.



 master → master [rejected - non-fast-forward]



Zentralisiertes und verteiltes VCS

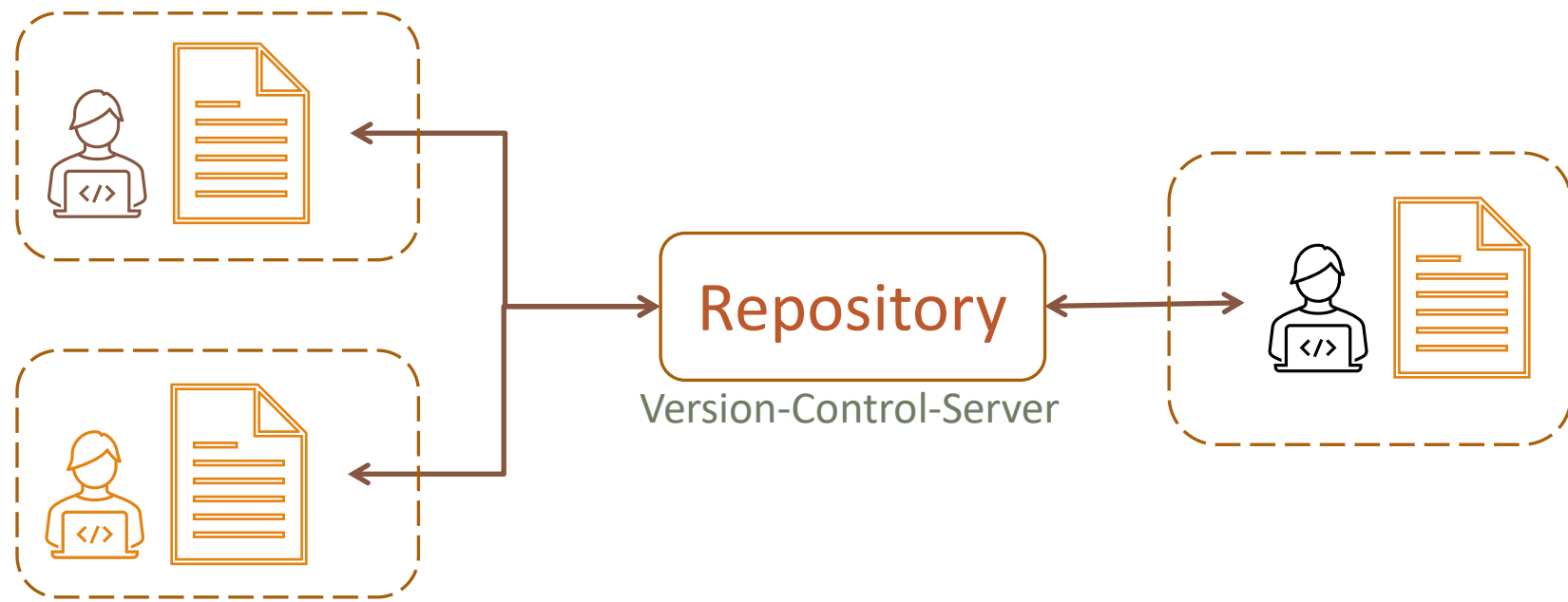
➤ Zentralisiert:

- Spezieller Version-Control-Server
- Alle Arbeitskopien von den Beteiligten Entwicklern werden mit den im Server abgeglichen
- Commit → Server
- Server → Update

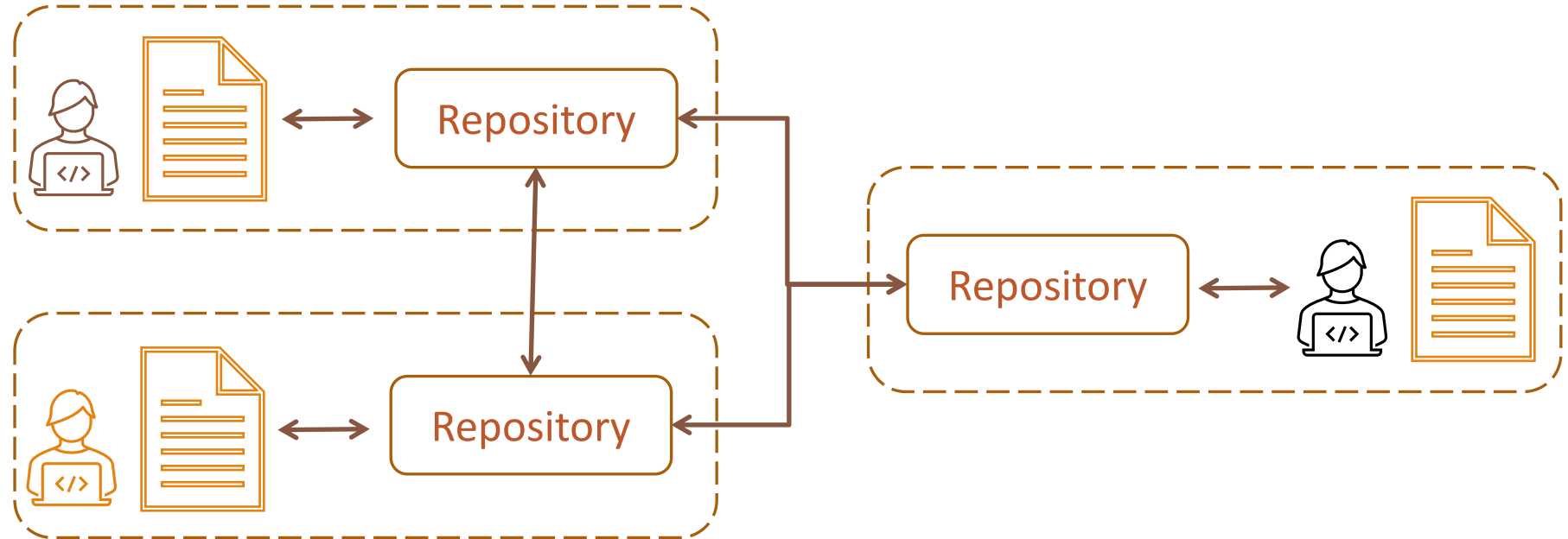
➤ Verteilt:

- git
- Jeder Beteiligte hat eine lokale Repository auf dem Rechner
- zwischen den Repositorien kann es abgeglichen werden
- „Commits“ und „Update“ bleiben lokal auf der Festplatte
- Mit „Push“ werden sie ans Ziel abgesendet
- Es gibt ein dediziertes Repository auf dem Server z.B. „github“
- Es wird auch zwischen dem lokalen und dem auf dem Server durch „pull“ und „push“ abgeglichen
- nicht nur einzelne Dateien sondern die ganzen Repositorien unter dem jeweiligen Branch werden abgeglichen

zentralisiertes VCS



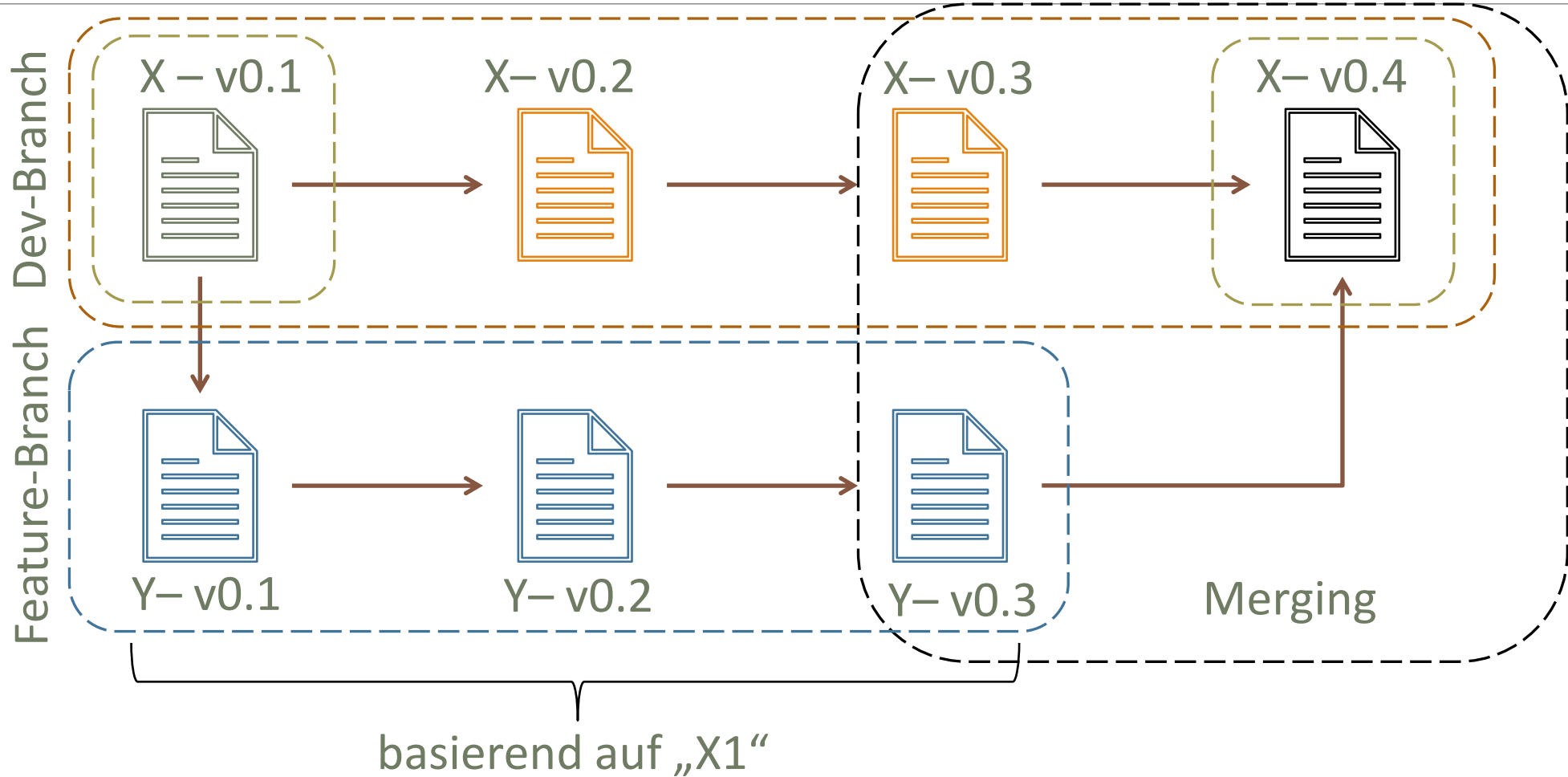
verteiltes VCS



Branching / Gabelung

- Erstellung von voneinander unabhängigen Kopien (Branchen) der Artefakten (Dateien, die sich in einem Repository befinden).
- Gleichzeitige Bearbeitung in den obengenannten Branchen.
- Die erledigten Änderungen in einem Branch werden zum späteren Zeitpunkt in einen anderen Branch zusammengeführt (merged)
- Drei meist verwendete Branche sind: Master-, Dev- und Feature-Branches

Branch



Meist verwendete Branches

Master-Branch „Head“

- ist die aktuellste und stabile Version der Anwendung

Development-Branch

- ist eine Entwicklungsversion

Feature-Branch

- ist ein Branch für neue Features der zu entwickelnden Anwendung

Release

- ist eine stabile nicht unbedingt finale Version der Anwendung

Hotfix

- ist ein Branch für die in der stabilen Version auftretenden Fehler zum Beseitigen

Versionsverwaltung mit git

git - Überblick

- Open-Source und verfügbar für alle
- global
- verfügt über ein verteiltes Architektur das bedeutet, dass git kein zentrales Repository ist
- Branchen zu betrachten
- lässt sich grafisch oder mit Command-Line konfigurieren

git – Initialisierung „init“

- Mit „init“ wird ein neues Repository angelegt
- Repository-Verwaltungsdaten befinden im Projektordner unter dem Ordner „.git“
- „.git“ kann einfach mitgeteilt

git – „clone“

- Eine Kopie eines vorhandenen Repositorys komplett lokal bzw. von Remote vornehmen
- hat Verweis auf „origin“ für „Push“ und „Pull“

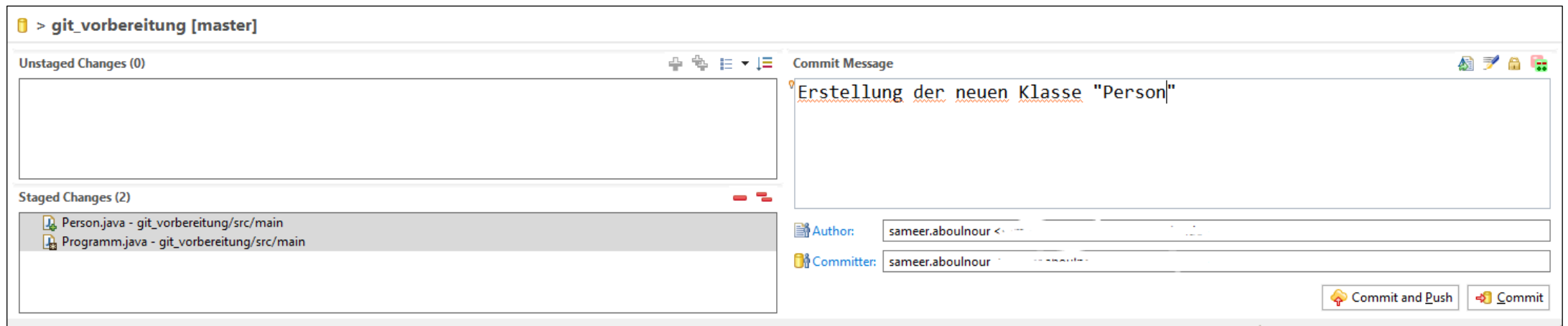
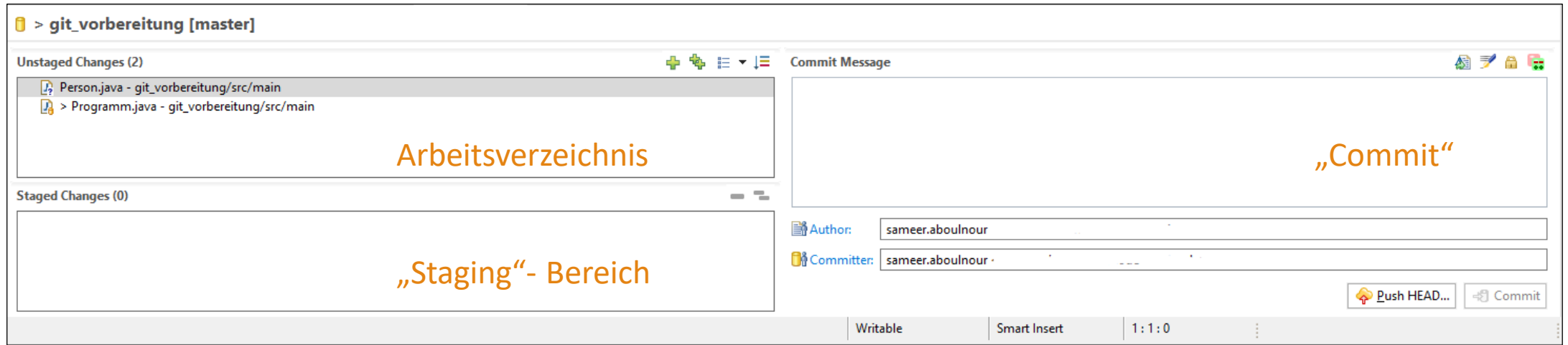
Name	Date modified	Type
.git	21/01/2022 16:22	File folder
git_vorbereitung	17/01/2022 12:23	File folder

Name	Date modified	Type	Size
branches	17/01/2022 12:23	File folder	
hooks	17/01/2022 12:23	File folder	
logs	17/01/2022 12:23	File folder	
objects	21/01/2022 16:17	File folder	
refs	17/01/2022 12:23	File folder	
COMMIT_EDITMSG	21/01/2022 16:17	File	1 KB
config	21/01/2022 16:22	File	1 KB
FETCH_HEAD	17/01/2022 12:23	File	1 KB
HEAD	17/01/2022 12:23	File	1 KB
index	21/01/2022 16:17	File	1 KB

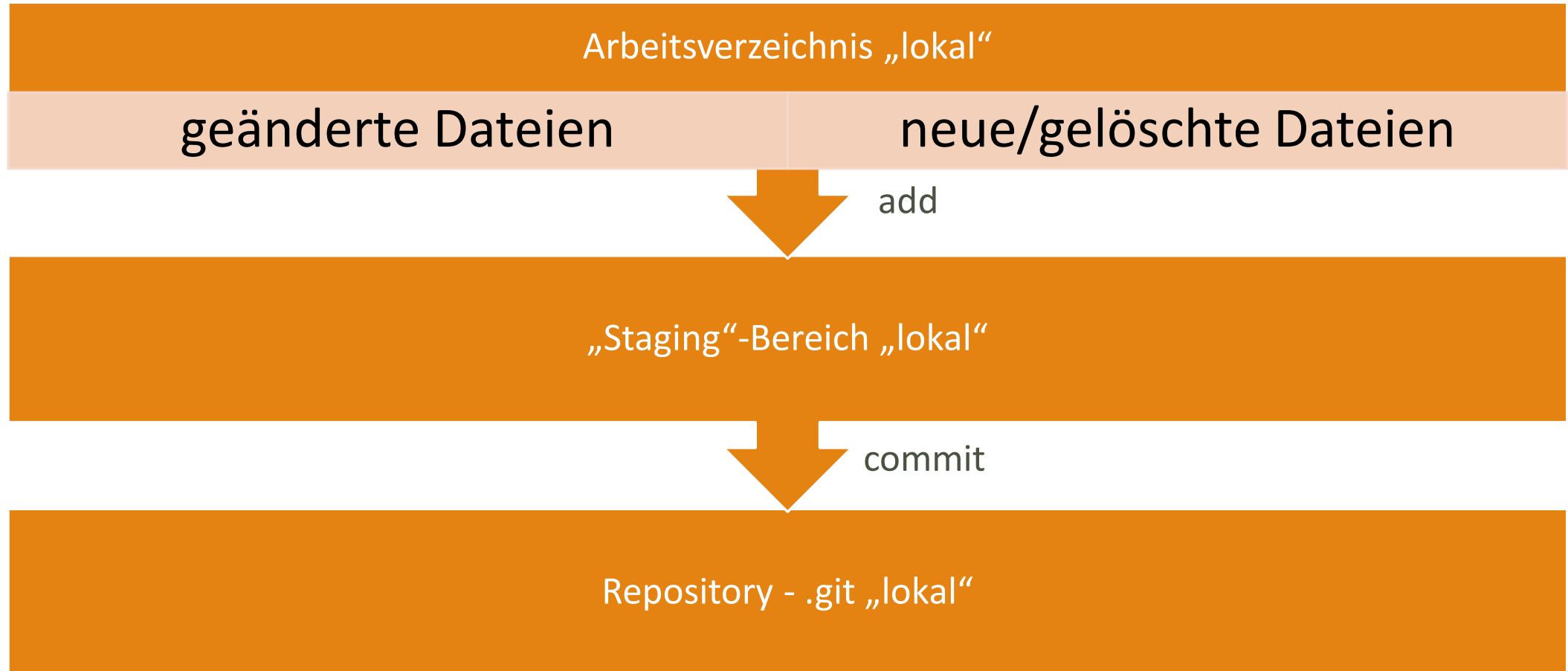
git – „config“

- Repositorien lassen sich in „config“ konfigurieren z.B. Benutzername, Email-Adresse usw.

git - Aufführung „Staging“



git - Aufführung „Staging“



git – Befehle

- „add“: Das Hinzufügen von Dateien vom Arbeitsverzeichnis in den „Staging“-Bereich „Snapshot“
- „commit“ Das Hinzufügen von Snapshots dem lokalen Repository
- „push“ lädt die aktuellste lokale Version in „origin“ hoch
- „pull“ importiert Commits und führt die Dateien vom „origin“ mit den lokalen zusammen
- „status“ der Status der Dateien in einem Repository (staged, unstaged...)
- „log“ Der gesamte Verlauf im Repository
- „checkout“ in einen anderen Bereich wechseln