**Versionierungstools & Buildchains**

Jonas Krahl, Tom Leutner, Alexander Bartz de Cárdenas, Sven Gapski

**Übersicht der Schulungsinhalte:**

1. Versionierungstools
   1. Allgemeines & Nutzen
   2. Git vs. Subversion
   3. Git Einführung & Basics
   4. Übungsaufgaben zur Nutzung von Git
2. Buildchains
   1. Allgemeines & Verwendung
   2. Verknüpfung von Versionierungs- und Buildtools

**Arten von Versionierungstools:**

**Lokale Versionierung:**

* Alle Arbeiten sind lokal
* Es gibt immer nur eine Realität
* Die Versionshistorie ist lokal
* Dateien werden als Ganzes versioniert
* Vergleiche sind möglich
* Performance ist gut

**Zentrale Versionierung (z.B. SVN):**

* Es gibt mehr als eine Realität (ein Server, n Workingcopies)
* Revisionen werden zentral verwaltet, Versionsnummern zentral vergeben
* Vergleiche sind nur direkt mit dem Server möglich
* Die Versionshistorie ist nur auf dem Server verfügbar
* Die Zentralisierung ermöglicht ein Zugriffs- und Rechtemanagement

**Dezentrale Versionierung (z.B. Git):**

* Es gibt mehr als eine Realität (ein Server, n Workingcopies)
* Revisionen werden zentral verwaltet, Versionsnummern zentral vergeben
* Vergleiche sind nur direkt mit dem Server möglich
* Die Versionshistorie ist nur auf dem Server verfügbar
* Die Zentralisierung ermöglicht ein Zugriffs- und Rechtemanagement

**Git Cheat-sheet:**

git **init** : Erstellt ein leeres Git-Repository.

git **clone** <repo> : Klont das Repository unter <repo> auf das lokale System.

git **add** <directory> : Fügt alle Veränderungen im <directory> zum Commit hinzu.

git **commit** -m : Committet alle hinzugefügten Dateien mit der Commit-“<message>“ Nachricht <message>.

git **reset** –-hard : Setzt das komplette lokale repository auf den Stand des remote-Repositories zurück

git **status**  : Übersicht der Dateien, die Teil des Commits, nicht Teil des Commits und „untracked“ sind.

git **branch :** Zeigt eine Liste aller Branches im Repository

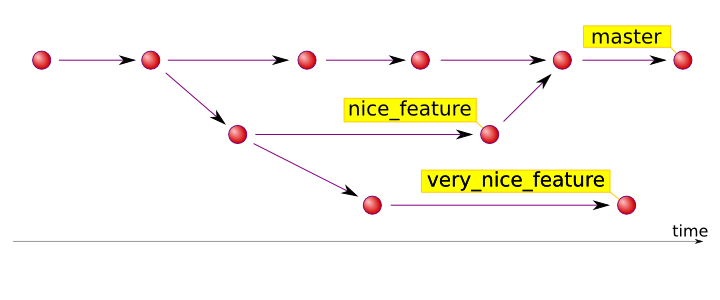
git **checkout** -b : Erstellt einen neuen Branch mit dem name <branch>. Ohne -b   
<branch> kann ein existierender Branch ausgecheckt werden.

git **merge** <branch> : Merged <branch> in den aktiven Branch.

git **pull** <remote> : Holt den neuesten Stand des aktiven Branches vom remote-

Repository und merged diesen in die Lokale Kopie.

git **push** <remote> : Lädt den aktiven lokalen Branch ins das remote-Repository <branch> hoch.

****

**Buildserver und deren Nutzen:**

**Was ist ein Buildserver:**

* Dedizierter Server um Softwareprojekte zu kompilieren
* Meist direkt an eine Versionsverwaltung gekoppelt
  + Anschließende Automatische Tests möglich
* Buildchain kann beliebig erweitert werden (Realease/Debug Builds, Versionen, QA, Tests usw.)

**Welche Vorteile hat die Nutzung eines Buildservers:**

* Sicherstellung, dass Code immer in selber Konfiguration kompiliert
* Kopilierbarer Code z.B. auf einem Branch immer vorhanden
* Deutliche Performance Vorteile bei großen Softwareprojekten
* Sofortiges Feedback bei fehlerhaftem Code
* Einfaches bauen eines Historie Standes, trotz nicht Archivierung.