Verziókövető rendszerek (VCS)

VCS (Version Control Systems)

Verziókezelés alatt több verzióval rendelkező adatok kezelését értjük. Leggyakrabban a mérnöki tudományokban és a szoftverfejlesztésben használnak verziókezelő rendszereket **fejlesztés alatt álló dokumentumok, tervek, forráskódok és egyéb olyan adatok verzióinak kezelésére, amelyeken több ember dolgozik egyidejűleg**.

A verziókezelő rendszerek segítségével ugyanazon digitális állományok változásai nyomon követhetők: ki, mikor, milyen változtatást végzett. A változások nemcsak nyomon követhetők, hanem igény szerint visszaállíthatók a korábbi állapotok.

Szolgáltatásai

Backup és visszaállítás:

A fájlok a szerkesztés közben mentésre kerülnek, és bármikor visszaállíthatók.

Szinkronizáció:

Ha egy szoftveren egynél több ember dolgozik, fontos, hogy folyamatosan a kódbázis legfrissebb verzióját lássák mindannyian.

Változások követése:

Ahogy a fájlok változnak, az egyes változtatások üzenetekkel láthatók el, amik leírást adnak arról, hogy mit és miért csináltunk.

Felelősség követése:

Mivel az egyes változtatások egyértelmű felhasználóhoz/fejlesztőhöz köthetőek, mindig pontosan tudhatjuk, ki és mikor követett el bizonyos változtatásokat.

Elágazás (branching) és összefésülés (merging):

A kódbázis egy másolatát lemásolhatjuk egy külön ágba, hogy ott aztán elkülönítve követett változtatásokat hajtsunk végre. Amint elkészült, ezeket a változtatásokat akár össze is fésülhetjük az eredeti kódbázissal.

Alapfogalmak:

Tároló (repository):

A verziókezelők projektünk aktuális verzióját és annak előzményeit külön erre a célra használt tárolókban (repository) raktározzák el.

Munkamásolat (working copy):

A kód egy részének (általában egy adott) példánya, amelyen a fejlesztő éppen dolgozik a saját gépén. A munka befejeztével ennek állapota kommitok formájában tárolásra kerül a repositoryban.

Kommit:

A kódunkon történt legfrissebb változtatásokat úgynevezett **kommitok** formájában érvényesíthetjük a tárolókon belül, melyek mintegy pillanatképként tartalmazzák azokat, illetve projektünk aktuális állapotát. Ha zsákutcába futnánk fejlesztés közben, akkor ezek alapján kereshetjük vissza kódunk korábbi verzióját. Ezért célszerű minden nagyobb módosítást követően kommitolnunk.

Fejlesztési történet (development history):

Az adott projekt kommitjainak összességét értjük alatta, az első (initial commit) kommittól kezdve az utolsóig (head commit).

Checkout/clone:

A módosítani kívánt fájlok lekérése a repositoryból és egy helyi változat létrehozása (a Subversion-ben checkout, a Git-ben clone a neve)

Pushing:

Adatok feltöltése a központi repository-ba.

Pulling:

Ha letöltöttünk egy tárolót valamely szerverről, és később szeretnénk azt frissíteni az aktuális állapotra, akkor a pull parancsot kell kiadnunk.

Fetching:

Adatok lekérése a központi repository-ból úgy, hogy nem szeretnénk azonnal egybe is olvasztani a letöltött változatot az éppen aktuális lokálissal.

Fejlesztési ágak (branch-ek):

Lehetőségünk van ún. fejlesztési ágak létrehozására is, ha projektünket esetleg a tervezettől eltérő irányban is szeretnénk továbbfejleszteni. Valamint az eredeti verziót érintetlenül hagyva tudunk kísérletezni.

Összefésülés (merging):

A fejlesztési ágak létrehozása mellett lehetőségünk van ezek egyesítésére is. Ennek folyamatát összefésülésnek *(angolul: merging)* nevezik.