

Systemy kontroli wersji

GitLab

Patryk Jasik

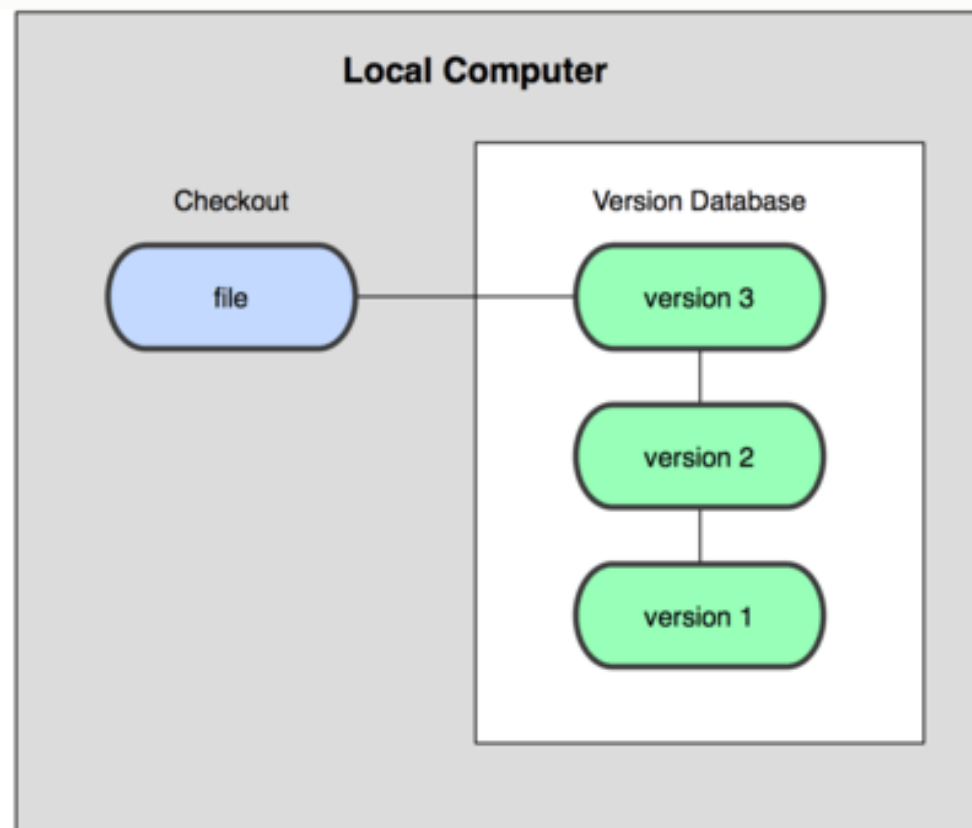
Katedra Fizyki Teoretycznej i Informatyki Kwantowej
Wydział Fizyki Technicznej i Matematyki Stosowanej
Politechnika Gdańska

System kontroli wersji (VCS - Version Control System) śledzi wszystkie zmiany dokonywane na pliku (lub plikach) i umożliwia przywołanie dowolnej wcześniejszej wersji.

Za pomocą VCS można odtworzyć stan całego projektu, porównać wprowadzone zmiany, dowiedzieć się kto jako ostatni zmodyfikował część projektu powodującą problemy, kto i kiedy wprowadził daną modyfikację. Nawet jeśli popełniony zostanie błąd lub część danych zostanie stracona, naprawa i odzyskanie ich powinno być łatwe.

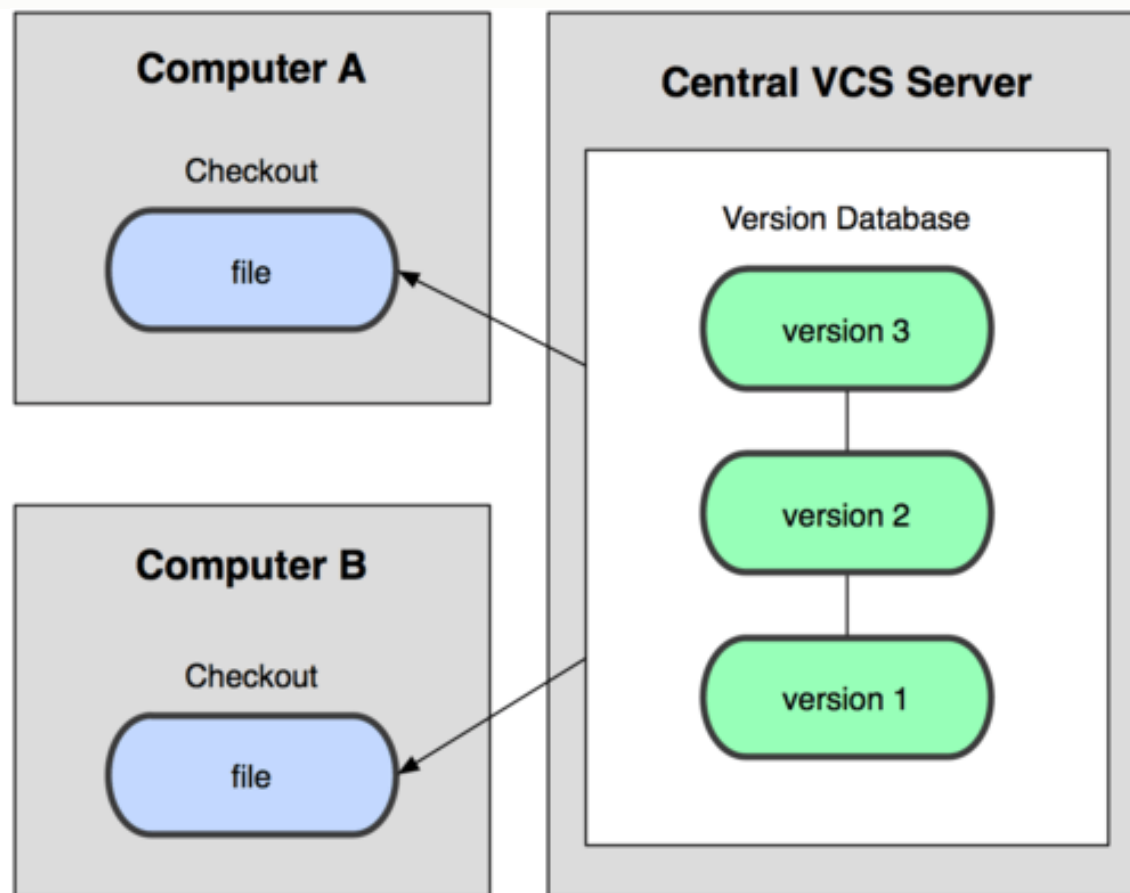
Lokalne systemy kontroli wersji
Scentralizowane systemy kontroli wersji
Rozproszone systemy kontroli wersji

Lokalny system kontroli wersji



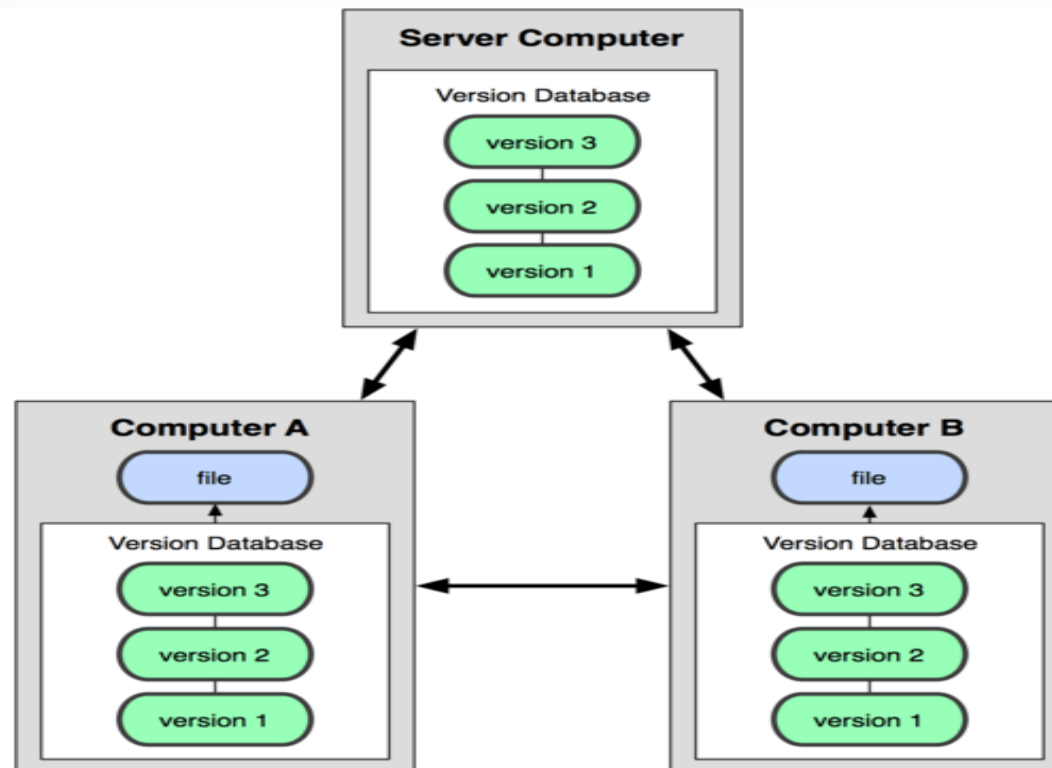
Przykładem może być system **rcs**. Program ten działa zapisując, w specjalnym formacie na dysku, dane różnicowe (to jest zawierające jedynie różnice pomiędzy plikami) z każdej dokonanej modyfikacji. Używając tych danych jest w stanie przywołać stan pliku z dowolnego momentu.

Scentralizowany system kontroli wersji



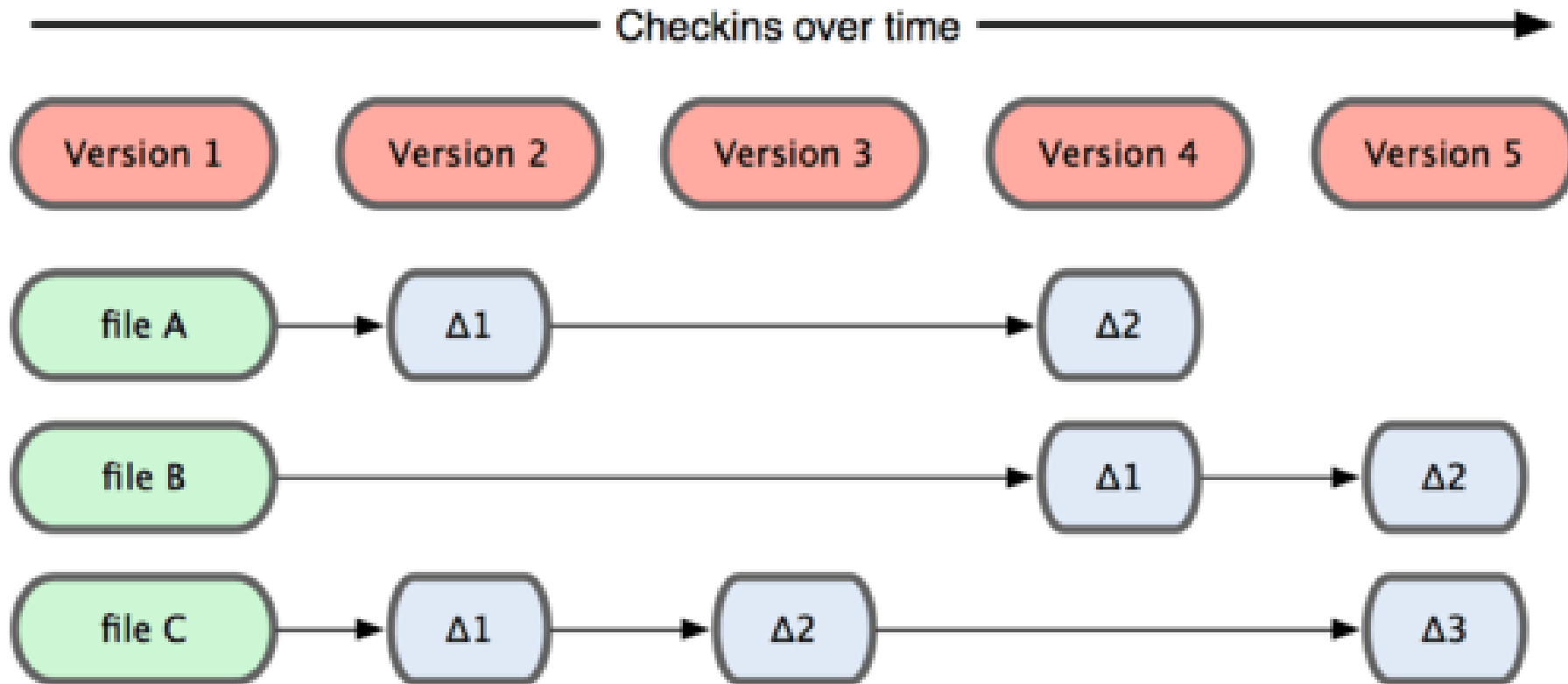
Systemy takie jak Subversion (SVN) czy Perforce składają się z jednego serwera, który zawiera wszystkie pliki poddane kontroli wersji, oraz klientów którzy mogą się z nim łączyć i uzyskać dostęp do najnowszych wersji plików.

Rozproszony system kontroli wersji



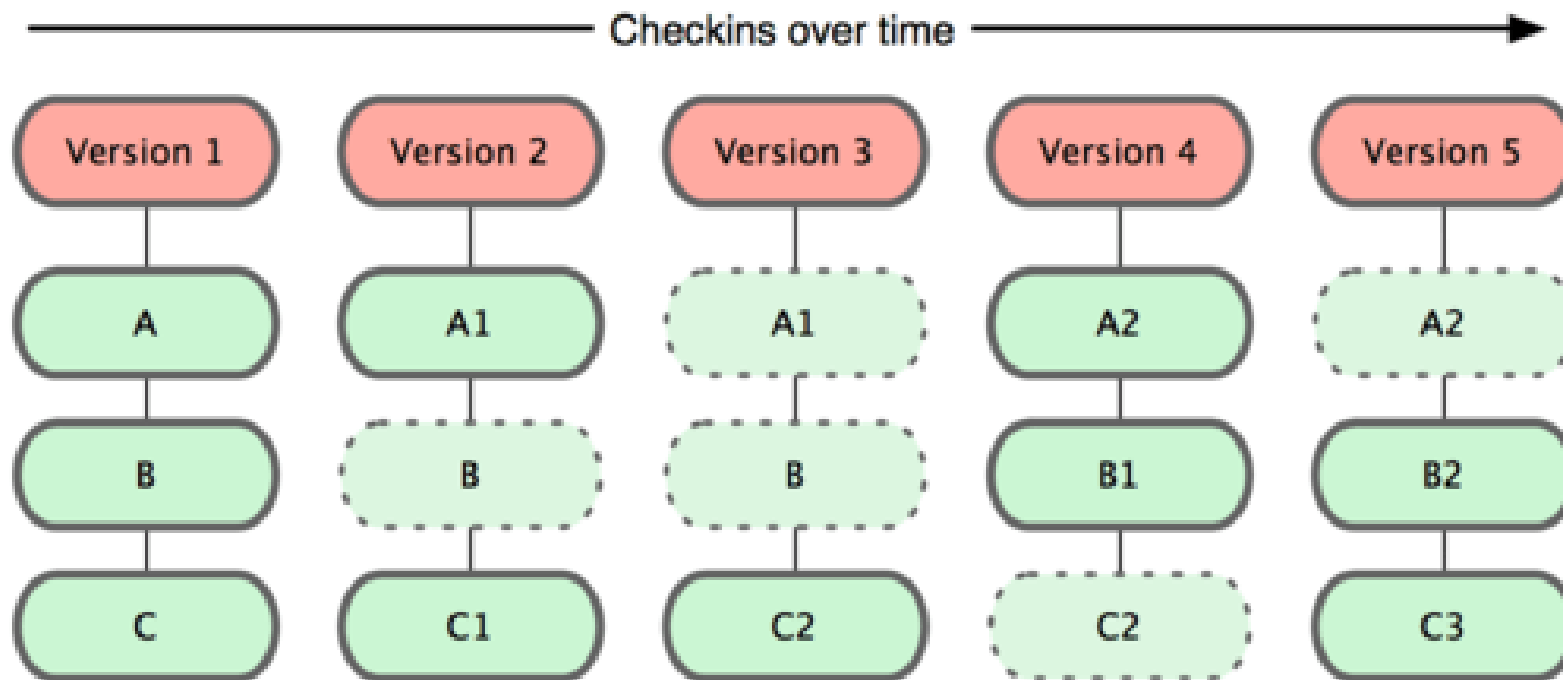
W systemach takich jak Git, Mercurial, Bazaar lub Darcs klienci nie dostają dostępu jedynie do najnowszych wersji plików, ale w pełni kopiują całe repozytorium. Gdy jeden z serwerów, używanych przez te systemy do współpracy, ulegnie awarii, repozytorium każdego klienta może zostać po prostu skopiowane na ten serwer w celu przywrócenia go do pracy.

Różne metodologie



W systemach takich jak CVS, Subversion, Perforce, czy Bazaar przechowywane są pliki oraz zmiany dokonywane na każdym z nich w czasie (podejście różnicowe).

Różne metodologie



Git traktuje dane jako zestaw migawek (snapshots) z określonego systemu plików.

Gdy tworzony jest commit lub zapisywany stan projektu, Git tworzy obraz przedstawiający wygląd wszystkich plików w danym momencie i przechowuje referencję do tej migawki. Jeśli dany plik nie został zmieniony, Git nie zapisuje ponownie tego pliku, a tylko referencję do jego poprzedniej, identycznej wersji, która jest już zapisana.

Dobre cechy Git'a

- Prawie wszystkie operacje są lokalne:
 - większość operacji w Git do działania wymaga jedynie dostępu do lokalnych plików i zasobów, czyli nie są potrzebne żadne dane przechowywane na innym komputerze w sieci.
 - kompletna historia projektu znajduje się w całości na dysku.
- Wbudowane są mechanizmy spójności danych:
 - dla każdego obiektu Git wyliczana jest suma kontrolna przed jego zapisem i na podstawie tej sumy można odwoływać się do danego obiektu.
 - nie ma możliwości zmiany zawartości żadnego pliku, czy katalogu bez reakcji ze strony Git.
 - do wyznaczenia sumy kontrolnej Git wykorzystuje tzw. skrót SHA-1. Jest to 40-znakowy łańcuch składający się z liczb szesnastkowych (0–9 oraz a–f), wyliczany na podstawie zawartości pliku lub struktury katalogu (**24b9da6552252987aa493b52f8696cd6d3b00373**)

Dobre cechy Git'a

- Git przechowuje wszystko nie pod postacią plików i ich nazw, ale we własnej bazie danych, w której kluczami są skróty SHA-1, a wartościami - zawartości plików, czy struktur katalogów.
- Git dodaje wyłącznie nowe dane
 - niemal zawsze jedynie dodajemy dane do bazy danych Git.
 - trudno jest zmusić system do zrobienia czegoś, z czego nie można się następnie wycofać, albo sprawić, by niejawnie skasował jakieś dane.
 - można stracić lub nadpisać zmiany, które nie zostały jeszcze zatwierdzone; ale po zatwierdzeniu migawki do Git, bardzo trudno jest stracić te zmiany, zwłaszcza jeśli regularnie pchasz (push) własną bazę danych Git do innego repozytorium.

Dobre cechy Git'a

- Trzy stany, w których mogą znajdować się pliki:
 - **zatwierdzony, zmodyfikowany i śledzony.**
 - zatwierdzony oznacza, że dane zostały bezpiecznie zachowane w lokalnej bazie danych.
 - zmodyfikowany oznacza, że plik został zmieniony, ale zmiany nie zostały wprowadzone do bazy danych.
 - śledzony oznacza, że zmodyfikowany plik został przeznaczony do zatwierdzenia w bieżącej postaci w następnej operacji **commit**.
 - Git posiada trzy główne sekcje projektu: katalog Git, katalog roboczy i przechowalnia (ang. staging area).
 - Katalog Git jest miejscem, w którym Git przechowuje własne metadane oraz obiektową bazę danych projektu. To najważniejsza część Git i to właśnie ten katalog jest kopiowany podczas klonowania repozytorium z innego komputera.

Dobre cechy Git'a

- Katalog roboczy stanowi obraz jednej wersji projektu. Zawartość tego katalogu pobierana jest ze skompresowanej bazy danych zawartej w katalogu Git i umieszczana na dysku w miejscu, w którym można ją odczytać lub zmodyfikować.
- Przechowalnia to prosty plik, zwykle przechowywany w katalogu Git, który zawiera informacje o tym, czego dotyczyć będzie następna operacja commit. Czasami można spotkać się z określeniem indeks, ale ostatnio przyjęło się odwoływać do niego właśnie jako przechowalnia.
- Podstawowy sposób pracy z Git wygląda następująco:
 - Dokonujesz modyfikacji plików w katalogu roboczym.
 - Oznaczasz zmodyfikowane pliki jako śledzone, dodając ich bieżący stan (migawkę) do przechowalni.
 - Dokonujesz zatwierdzenia (commit), podczas którego zawartość plików z przechowalni zapisywana jest jako migawka projektu w katalogu Git.

Dobre cechy Git'a

Local Operations

