Systemy kontroli wersji (VCS)

Projekt (definicja na potrzeby wykładu) – katalog z dowolną zawartością (np. kody źródłowe, dane, ...), aktywnie rozwijany (np. powstająca aplikacja).

Typowe problemy związane z zarządzaniem projektem:

- Konieczność tworzenia kopii zapasowych.
- Utrudnione śledzenie zmian pomiędzy kopiami.
- Utrudnione eksperymentowanie.
- Kolaboracja: wielu użytkowników edytujących te same pliki, konfliktujące zmiany.

Rozwiązanie: **system kontroli wersji/wersjonowania** (*version control system*).
Przykłady: git, Mercurial, Subversion, CVS.

Systemy kontroli wersji (VCS)

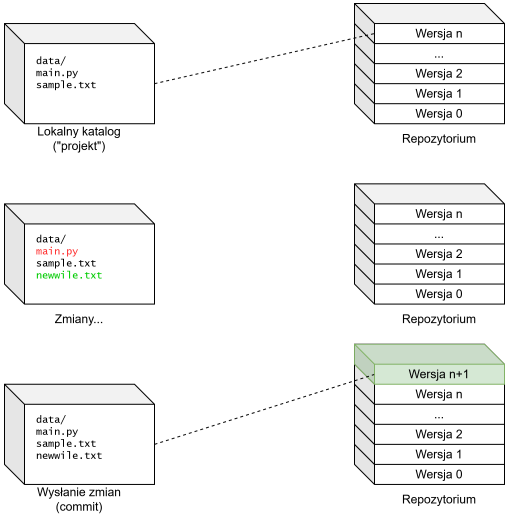
Zadanie systemu kontroli wersji: systematyczne śledzenie zmian wykonanych w projekcie (dodawania, usuwania i zmian w plikach i katalogach).

Repozytorium (w uproszczeniu) – baza danych, zawierająca kopie (snapshoty) kolejnych* wersji projektu. Pozwala na pobranie wybranej wersji oraz zapisanie nowej.

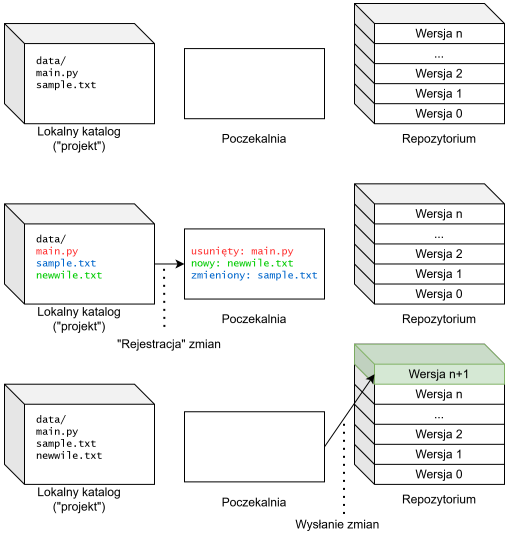
Repozytorium może być lokalne (typowe dla projektów rozwijanych przez jedną osobę), zdalne (z dostępem dla wielu użytkowników) lub rozproszone (każdy użytkownik ma lokalną kopię repozytorium, wszystkie repozytoria są synchronizowane).

Systemy kontroli wersji (VCS)

Poglądowy rysunek dla lokalnego repozytorium z jednym użytkownikiem.

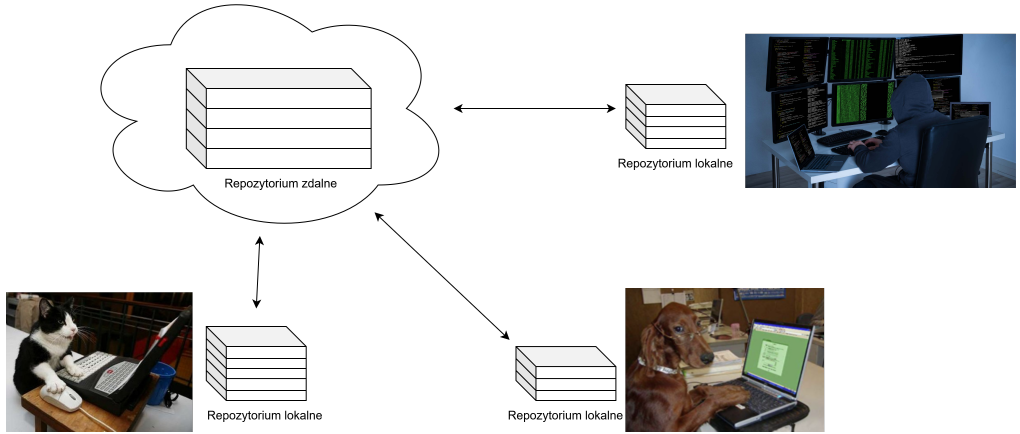


Poczekalnia (staging area) – rejestracja dokonanych zmian przed wysłaniem.



Systemy kontroli wersji (VCS)

Przykład systemu rozproszonego z „głównym” repozytorium. Synchronizacja („w górę” lub „w dół”) na żądanie użytkownika.



git

git – dominujący system kontroli wersji.

- Rozproszona architektura (każdy ma lokalną kopię repozytorium).
- Szybki, z efektywną reprezentacją wersji (np. jeśli plik nie zmienia się pomiędzy wersjami, pamiętana będzie tylko jedna jego kopia; kompresja dużych plików).
- Elastyczna struktura repozytorium: możliwość rozgałęziania projektu i sklejania rozgałęzień.
- GitHub (https://github.com) oraz podobne – hosting repozytoriów z dodatkową funkcjonalnością (bug trackery, wiki projektów, „social media", ...).

Konkretne narzędzia:

- Polecenie powłoki (używane bezpośrednio lub przez pozostałe oprogramowanie) `git`: dostępne pod Linuksem (typowa dystrybucja ma gotowy pakiet, niekoniecznie od razu zainstalowany) i MacOS (pakiet). Windows: Git for Windows (https://github.com/git-for-windows/git/releases/).
- GitHub Desktop – graficzny frontend do zarządzania repozytoriami hostowanymi na GitHubie oraz GitHub CLI (linia poleceń).
- Integracja z typowymi IDE (PyCharm, VS Code, Visual Studio, ...). Często obsługiwane są również Mercurial i SVN (Subversion).
- ...
- (ww. narzędzia wewnętrznie używają powłokowego `git` lub GitHub CLI)

Live demo (w przybliżeniu)

```
$ mkdir git_demo && cd git_demo
$ git init
$ git config user.name me
$ git config user.email my@mail.org
$ echo "Hello world" > a.txt
$ git add a.txt
$ git commit -m "Pierwsza wersja"

$ touch b.txt c.txt
$ git add .
$ git commit -m "Drugi commit"

$ rm b.txt
$ git add .
$ git commit -m "Usunięcie b.txt"
```

Live demo (w przybliżeniu) c.d.

- Integracja z PyCharmem.
- GitHub (autoryzacja – więcej w materiałach do wykładu).
- Klonowanie repozytorium (powłoka, PyCharm).

Repozytoria zdalne (np. publiczne repozytorium na GitHubie: https://github.com/GJ-Demo/PWAI/):

```
$ git clone https://github.com/GJ-Demo/PWAI/
$ git remote -v
...
$ git push origin main
```