System kontroli wersji

Czyli jak w łatwy sposób uchronić się przed utratą efektów pracy.

reaktor

PWN

Wprowadzenie

- Jest to swego rodzaju wprowadzenie do systemu kontroli wersji, który będziesz wykorzystywać podczas kursu Back-end Developer.
- Jeśli czegoś nie zrozumiesz to się nie martw dokładnie zrozumiesz zasadę działania Gita jak będziesz miał okazję na co dzień go wykorzystywać podczas kursu.
- Twoim zadaniem jest zrozumienie na czym polega praca z system Git, tj.:
  - Podstawowe polecenia stosowane w pracy z repozytorium lokalnym.
  - Podstawowe polecenia stosowane w komunikacji ze zdalnym repozytorium.
  - Wykonanie pierwszego projektu i umieszczenie go w repozytorium GitHub.

reaktor

PWN

Agenda

1. Czym jest Git?
2. Konfiguracja
3. Praca z repozytorium lokalnym
4. Praca z repozytorium zdalnym
5. Praca z GitHub

reaktor

PWN

Czym jest Git?

- Git to rozproszony system kontroli wersji.

System kontroli wersji śledzi wszystkie zmiany dokonywane na pliku (lub plikach) i umożliwia przywołanie dowolnej wcześniejszej wersji. Pozwala on przywrócić plik(i) do wcześniejszej wersji, odtworzyć stan całego projektu, porównać wprowadzone zmiany, dowiedzieć się kto jako ostatnio zmodyfikował część projektu powodującą problemy, kto i kiedy wprowadził daną modyfikację. Nawet jeśli popełnisz błąd lub stracisz część danych, naprawa i odzyskanie ich powinno być łatwe.

- Funkcjonalności:
  - śledzenie zmian w plikach, które są wykorzystywane w projekcie.
  - łatwy powrót do wcześniejszej wersji pliku
  - sprawną zdalną współpracę

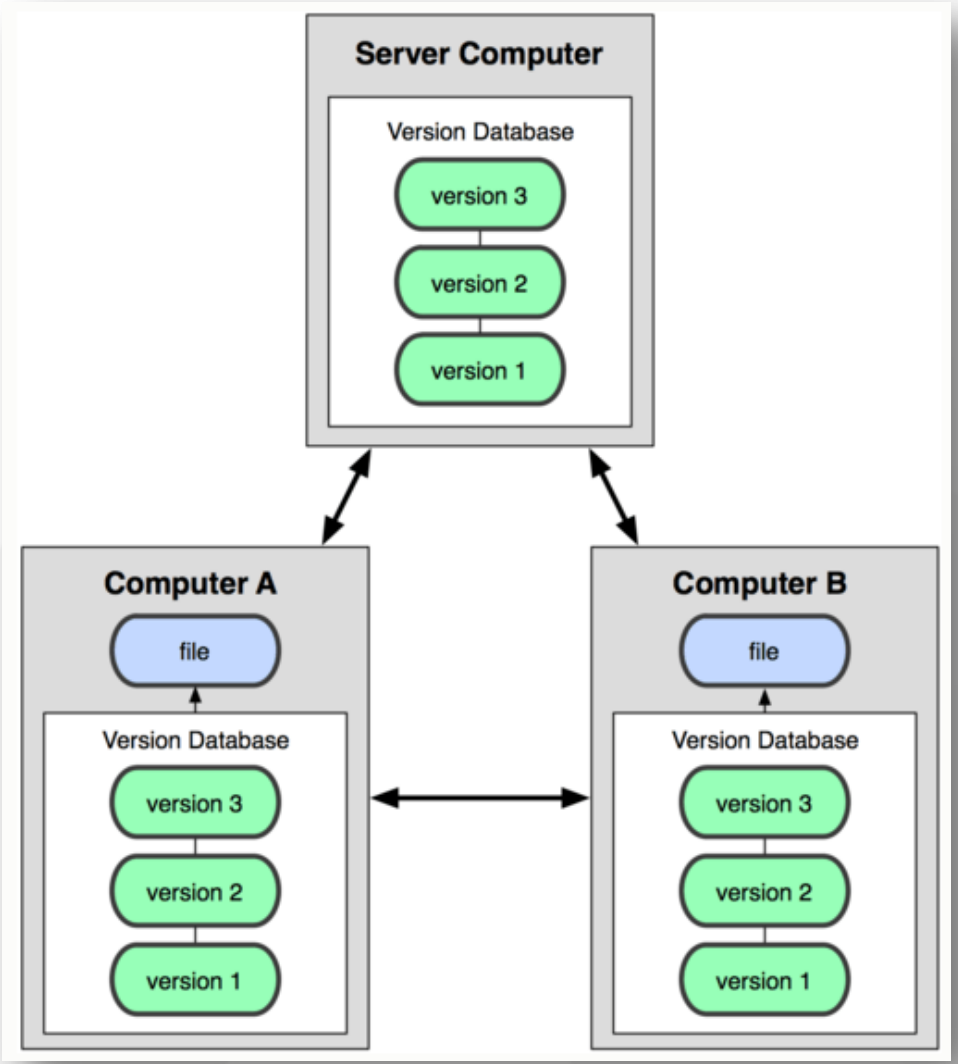
reaktor

PWN

Czym jest rozproszony system kontroli wersji?

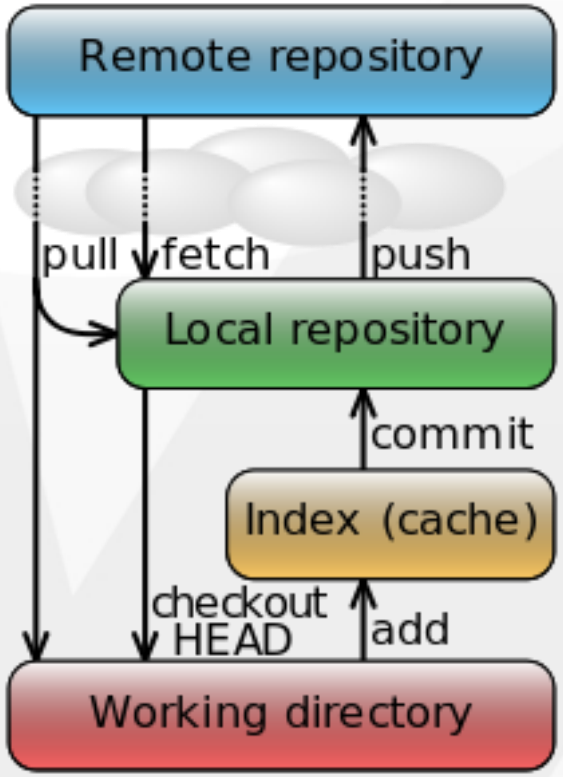
- Rozproszony system kontroli wersji (DVCS - Distributed Version Control System) działają w taki sposób, że klienci nie dostają dostępu jedynie do najnowszych wersji plików ale w pełni kopiują całe repozytorium. Gdy jeden z serwerów, używanych przez te systemy do współpracy, ulegnie awarii, repozytorium każdego klienta może zostać po prostu skopiowane na ten serwer w celu przywrócenia go do pracy.
- Ponadto, wiele z tych systemów dość dobrze radzi sobie z kilkoma zdalnymi repozytoriami, więc możliwa jest jednoczesna współpraca z różnymi grupami ludzi nad tym samym projektem. Daje to swobodę wykorzystania różnych schematów pracy, nawet takich które nie są możliwe w scentralizowanych systemach, na przykład modeli hierarchicznych.

Schemat rozproszonego systemu kontroli wersji



Czym są repozytoria?

- Repozytorium to nic innego jak miejsce w który przechowywane są jakieś zasoby.
- Wyróżniamy repozytoria:
  - Lokalne – znajdujące się na komputerze na którym pracujesz.
  - Zdalne – znajdujące się na serwerach zewnętrznych z którymi możesz się połączyć wykorzystując sieci komputerowe (np. Internet).



Oprogramowanie

- Pobieramy oprogramowanie dla systemu operacyjnego Windows ze strony www.git-scm.com/download/win.
- Cała procedura pobrania i instalacji oprogramowania została dokładnie opisana w prezentacji dotyczącej oprogramowania niezbędnego do realizacji kursu Back-end Developer.
- Jeśli mamy już zainstalowane oprogramowanie to otwieramy plik Git CMD który możemy znaleźć wyszukując go w menu Windows.

Git CMD

- Poruszanie się w drzewie katalogów Git CMD odbywa się tak samo jak w wierszu poleceń Windows.
  - **cd nazwaKat – przejście do katalogu o nazwie nazwaKat**
  - **dir – wypisanie zawartości bieżącego katalogu**
- Proponuję utworzyć na pulpicie katalog Git-start i za pomocą w/w komend ustawić się wewnątrz tego kalogu, tak aby dalsze działania wykonywać w kontrolowanym przez nas miejscu.

reaktor

PWN

Konfiguracja

- Rozpoczynamy od ustawienia swoich danych:

git config --global user.name „nazwa_usera”

git config --global user.email aders_email

- Aby wyświetlić ustawienia wykonujemy polecenie:

git config --list

Praca z repozytorium lokalnym

Polecenie git init

- Polecenie inicjalizacji repozytorium Git w danym katalogu.
- Wybierz katalog w którym będzie znajdowało się Twoje pierwsze repozytorium zdalne (np.: Git-projekt) i wykonaj polecenie.

git init

Odpowiedź:

Initialized empty Git repository

reaktor

PWN

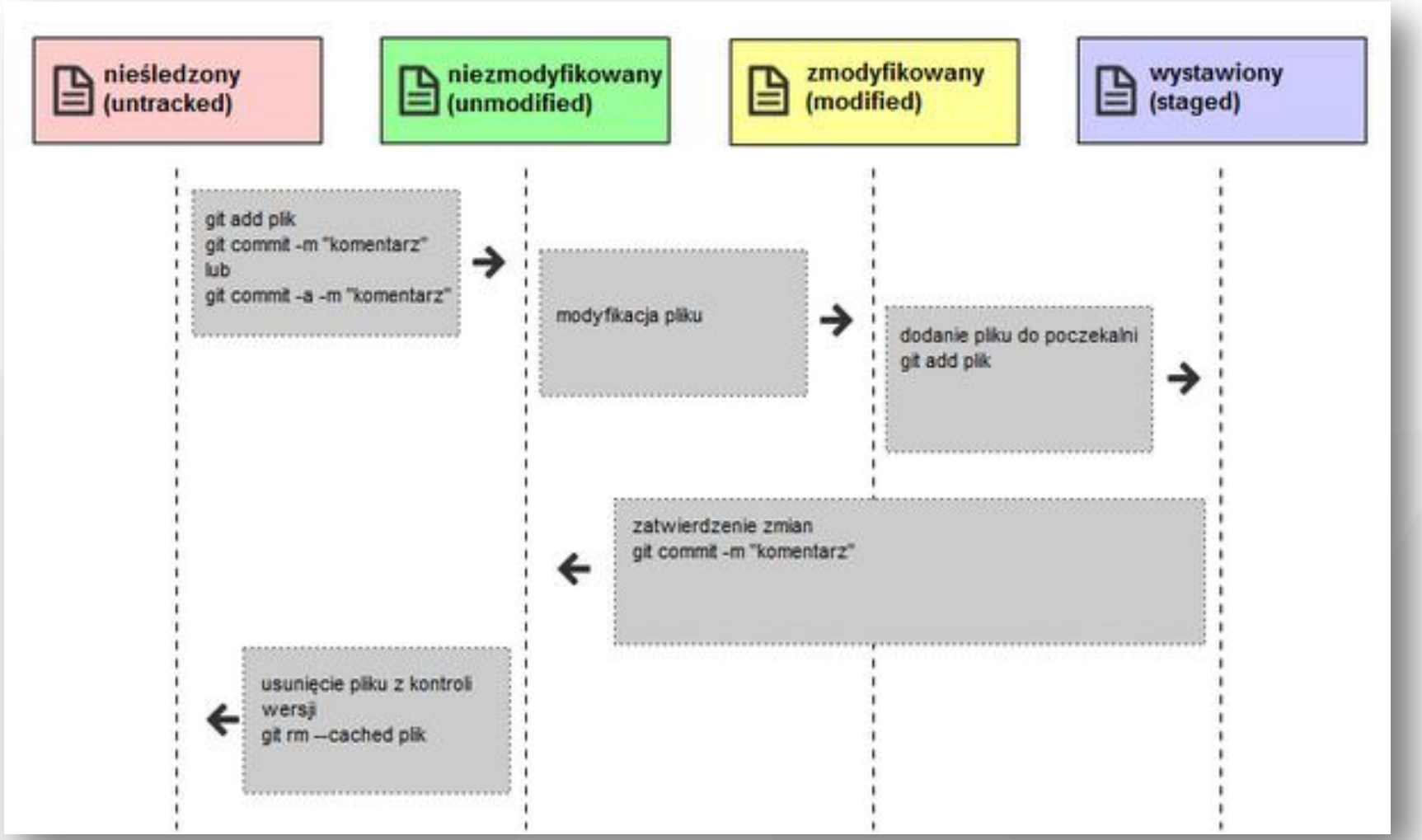
Jak to działa?

- Plik może znajdować się w czterech stanach:
  - **nieśledzony** (eng. untracked, zaraz po stworzeniu nowego pliku)
  - **zmodyfikowany** (eng. modified, plik zmieniony, zmiany niezapisane do bazy)
  - **śledzony** (eng. staged, zmiany przeznaczone do zatwierdzenia przy następnej operacji commit)
  - **zatwierdzony/niezmodyfikowany** (eng. commited/unmodified, bezpiecznie zachowany w lokalnej bazie danych)

reaktor

PWN

Cykle życia plików



reaktor

PWN

Jak to w uproszczeniu działa?

- Dokonujesz modyfikacji plików w katalogu roboczym.
- Oznaczasz zmodyfikowane pliki jako śledzone (polecenie: *add*), dodając ich bieżący stan do przechowalni.
- Dokonujesz zatwierdzenia (polecenie: *commit*), podczas którego zawartość plików z przechowalni jest zapisywana.

Polecenie git status

- Polecenie stosowane do przeglądania statusu plików, czyli wyświetlenie listy zmienionych plików.

git status

Odpowiedź:

On branch master

nothing to commit, working directory clean

reaktor

PWN

.gitignore

- Plik o nazwie .gitignore zawiera informacje, których plików nigdy nie chcemy dodawać do repozytorium, czyli jest to swego rodzaju lista zawierająca nazwy plików, które chcemy ignorować.

- Nazwy plików w .gitignore nie będą widoczne również po wywołaniu git status:
  - secret.js — Ukryje plik secret.js
  - *.txt — Spowoduje, że git zignoruje wszystkie pliki o rozszerzeniu .txt

Polecenie git add / rm

- Pracując na plikach (wprowadzając w nich jakieś zmiany) w pewnym momencie chcemy przenieść pliki do poczekalni lub je z tej poczekalni usunąć. Stosujemy do tego następujące polecenia:

  - Dodanie wszystkich zmian:

git add .

  - Dodanie konkretnego pliku:

git add nazwa_pliku

  - Gdy nie chcęsz już śledzić pliku:

git rm –cached nazwa_pliku

reaktor

PWN

Polecenie git commit

- Kiedy nasze pliki już znajdują się w poczekalni i jesteśmy przekonani że chcemy już je przenieść do naszego lokalnego repozytorium stosujemy polecenie commit i opcjonalnie możemy dodać komentarz przy każdym commitcie.
- Zapisanie zmian:

git commit -m "mój pierwszy commit"

git commit . -m "mój pierwszy commit"

reaktor

PWN

Polecenie git log

- Kiedy nasz projekt trwa już dość długo to może wystąpić sytuacja, że chcielibyśmy przejrzeć nasze dotychczasowe commity. Stosujemy do tego polecenie git log.
- Wyświetlamy wszystkie commity:

git log

git log --graph –-pretty=oneline

Polecenie git diff

- Operacja ta pozwala zaobserwować co się zmieniło się od ostatniej operacji commit?
- Wyświetla różnice w jednej linii pomiędzy dwoma ostatnimi commitami w jednej linii:

git diff –-color-words

Polecenie git reset

- Cofanie ostatnich commitów:

git reset HEAD~1

- Ściąganie plików ze 'sceny':

git reset HEAD .

git reset HEAD nazwa_pliku

Uwaga na opcję –-hard, nieodwracalnie, usuwa wszystkie zmiany!