Programowanie zespołowe

Laboratorium 8 - wprowadzenie do systemów kontroli wersji i GitHuba

mgr inż. Krzysztof Szwarc

krzysztof@szwarc.net.pl

Sosnowiec, 11 kwietnia 2017

Czym jest system kontroli wersji?

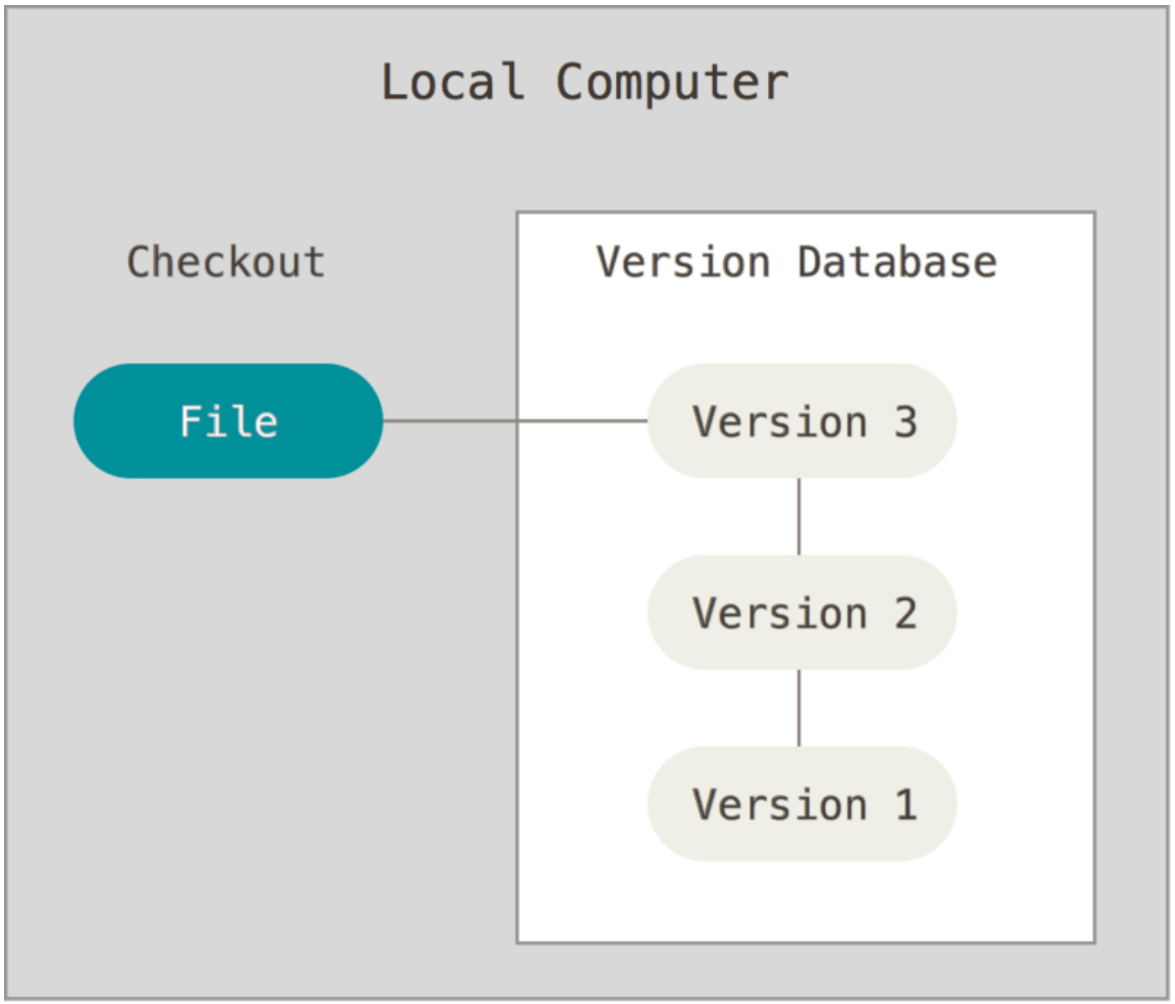
System kontroli wersji

System kontroli wersji (ang. version control system) jest oprogramowaniem wykorzystywanym do śledzenia zmian (przede wszystkim w kodzie źródłowym) oraz wspomagającym łączenie modyfikacji, które zostały naniesione w plikach przez zespół w różnym okresie czasu.

Podział systemów kontroli wersji wg architektury

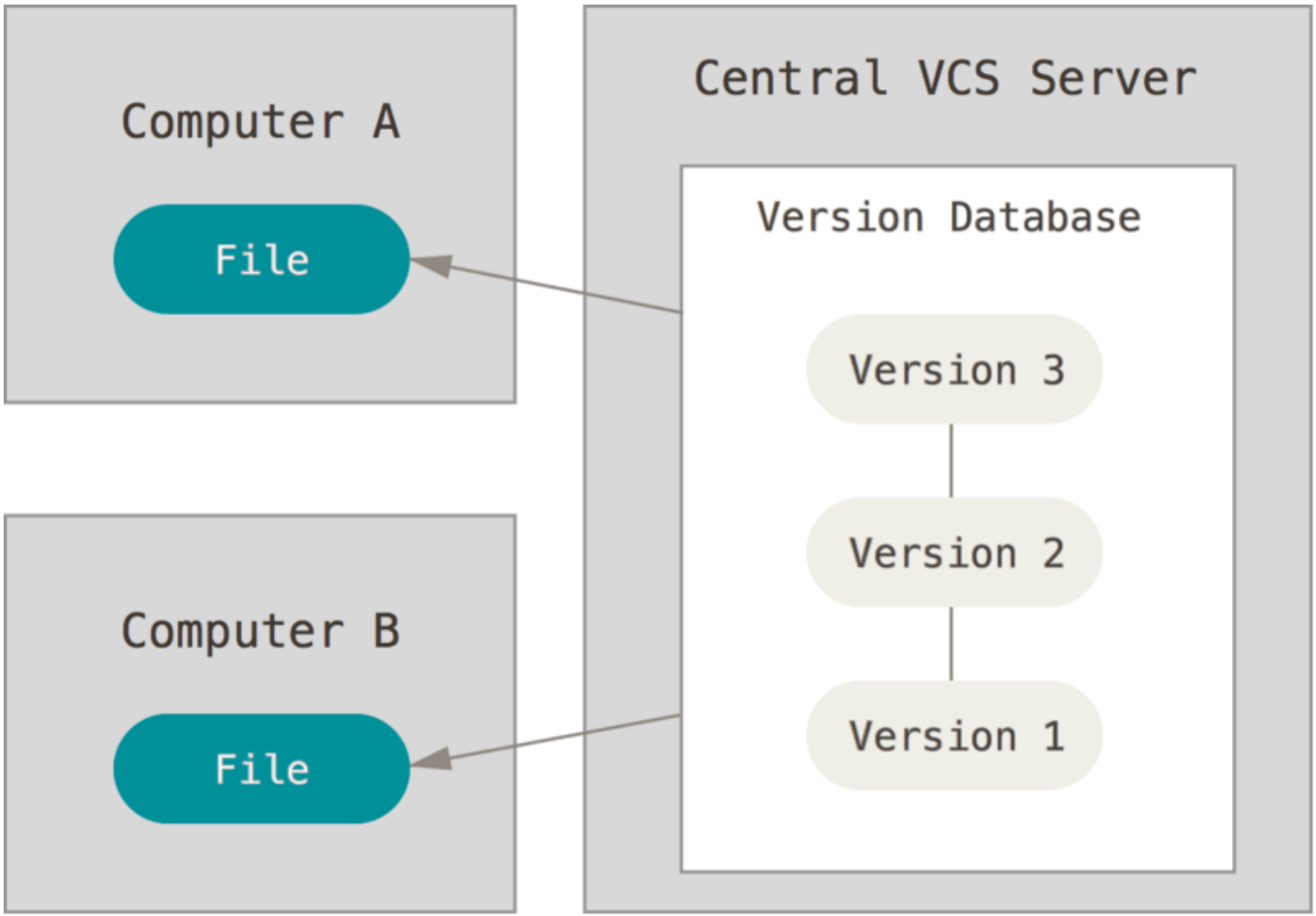
1. Lokalne (np. RCS) - zapisujące dane jedynie na lokalnym komputerze.
2. Scentralizowane (np. SVN) - oparte na architekturze klient-serwer, gdzie występuje jedno centralne repozytorium, z którym następuje synchronizacja zmian przez wszystkich użytkowników.
3. Rozproszone (np. Git) - oparte na architekturze P2P, gdzie prowadzi się równoprawne i niezależne gałęzie, które można dowolnie synchronizować ze sobą.

Działanie lokalnego systemu kontroli wersji



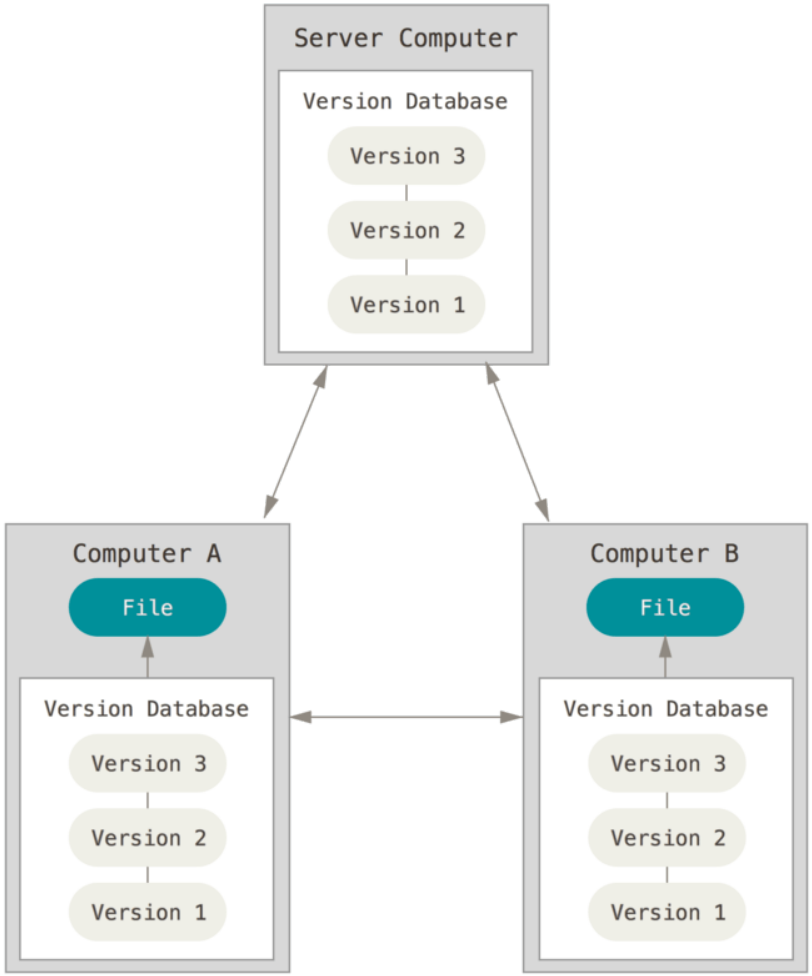
https://git-scm.com/

Działanie scentralizowanego systemu kontroli wersji



https://git-scm.com/

Działanie rozproszonego systemu kontroli wersji



https://git-scm.com/

Charakterystyka Gita

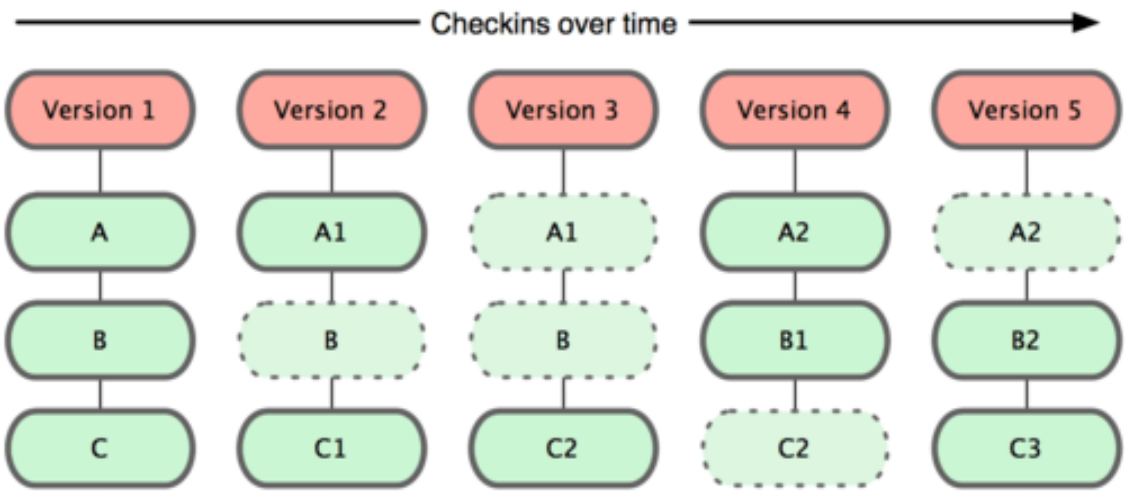
Opis

Git jest rozproszonym systemem kontroli wersji stworzonym przez Linusa Torvaldsa (twórca jądra Linux). Korzysta z niego m.in. jQuery, Perl, Qt, Ruby on Rails oraz KDE.

https://git-scm.com/

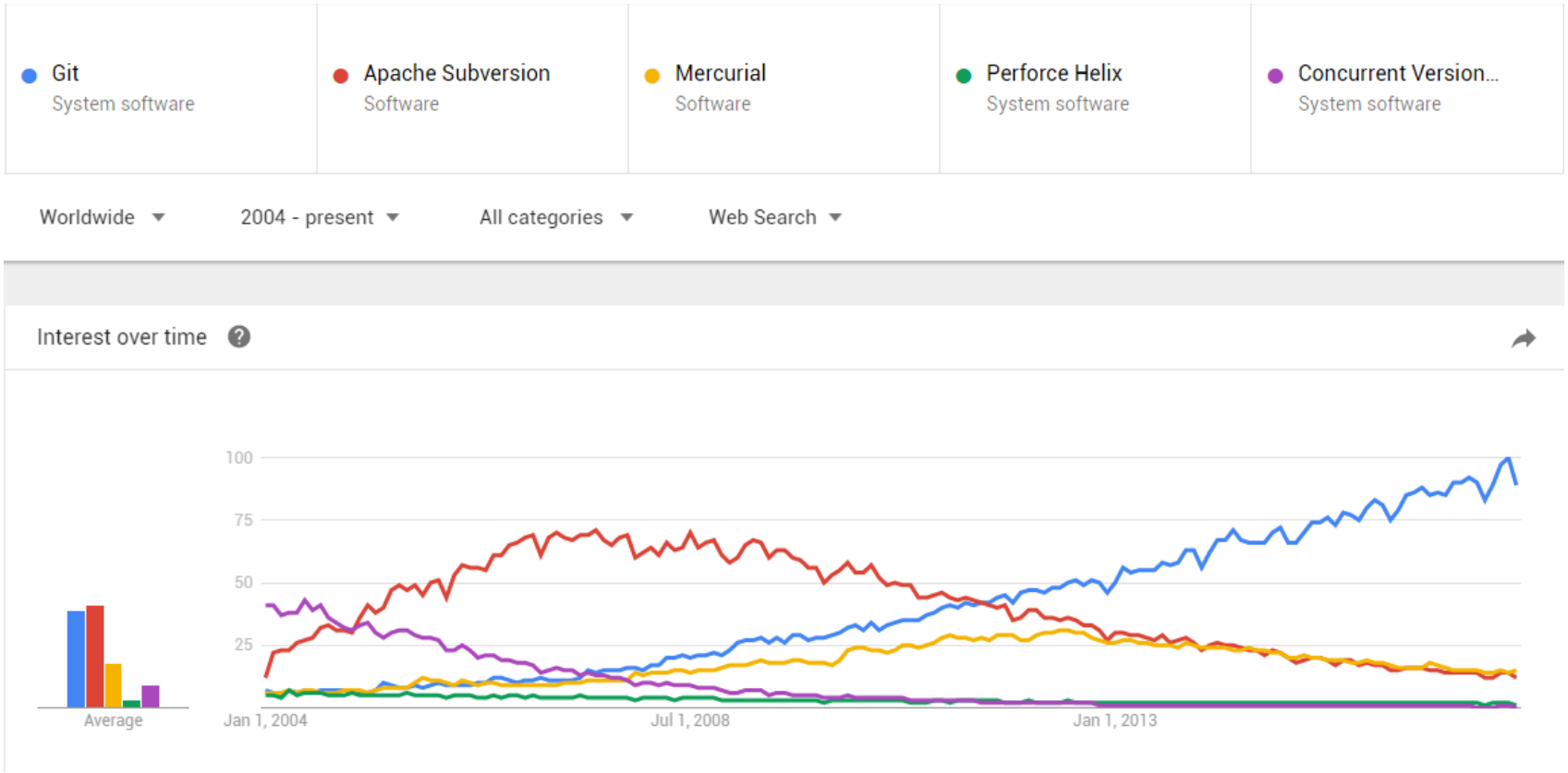
Sposób przechowywania plików w Git'cie

Git traktuje dane jak zestaw migawek systemu plików - w momencie zapisania stanu projektu tworzony jest obraz zawierający informacje o wyglądzie wszystkich plików w danych momencie, zawierający referencję do danej migawki.



https://git-scm.com/

Popularność Gita



https://trends.google.com/

GitHub

Opis

GitHub jest hostingowym serwisem przeznaczonym do przechowywania projektów programistycznych korzystających z systemu kontroli wersji Git. Udostępnia on darmowy hosting dla programów open source oraz płatne repozytoria prywatne.

https://revive-adserver.com/

GitHub dla studentów

Learn to ship software like a pro

Like 41K Tweet

There's no substitute for hands-on experience, but for most students, real world tools can be cost prohibitive. That's why we created the GitHub Student Developer Pack with some of our partners and friends: to give students free access to the best developer tools in one place so they can learn by doing.

Get your pack

https://education.github.com/pack

Waffle - zarządzanie projektami z GitHuba

Opis

Waffle jest darmowym systemem zarządzania zadaniami w pracy zespołowej silnie zintegrowanym z serwisem GitHub. Ułatwia on prowadzenie projektów zgodnie ze zwinnymi metodykami.

https://waffle.io/

Bitbucket

Opis

Bitbucket jest hostingowym serwisem przeznaczonym do przechowywania projektów programistycznych korzystających z systemu kontroli wersji Mercurial lub Git. Udostępnia on darmowe repozytoria prywatne (dla maksymalnie 5 kont użytkowników).

https://bitbucket.org/

Jira - zarządzanie projektami z Bitbucket

Opis

Jira jest zamkniętym oprogramowaniem firmy Atlassian (producent Bitbucketa) służącym do śledzenia błędów oraz zarządzania projektami, zgodnie z metodykami zwinnymi. Aplikacja jest zintegrowana z Bitbucketem.

https://jira.atlassian.com/

Podstawy terminologii GitHuba

Repozytorium

Repozytorium (ang. repository) to miejsce, w którym przechowywane są wszystkie wersje wszystkich plików źródłowych naszego projektu.

Wykonanie kopii

Fork umożliwia wykonanie kopii repozytorium.

Lokalne zatwierdzenie zmian

Commit to proces lokalnego zatwierdzenia modyfikacji. Powinien zawierać informacje dotyczące zmian.

Zdalne zatwierdzenie zmian

Push to proces wysłania zmian do innego repozytorium (zdalne zatwierdzenie modyfikacji).

Podstawy terminologii GitHuba cz. 2

Pobranie danych

Pull lub fetch oznacza proces pobrania zmian z innego repozytorium.

Gałęzie

Branch umożliwia utworzenie nowej wersji wewnątrz repozytorium (branche pozwalają na prace wielu użytkownik równocześnie, bez nadpisywania zmian – po zakończeniu pracy łączy się zmiany z kodem pozostałych i rozwiązuje konflikty).

Łączenie kodu

Merge umożliwia połączenie wielu zmian z różnych źródeł, które może spowodować wystąpienie sprzecznych modyfikacji wymagających ręcznej naprawy (merge conflict).

Podstawy terminologii GitHuba cz. 3

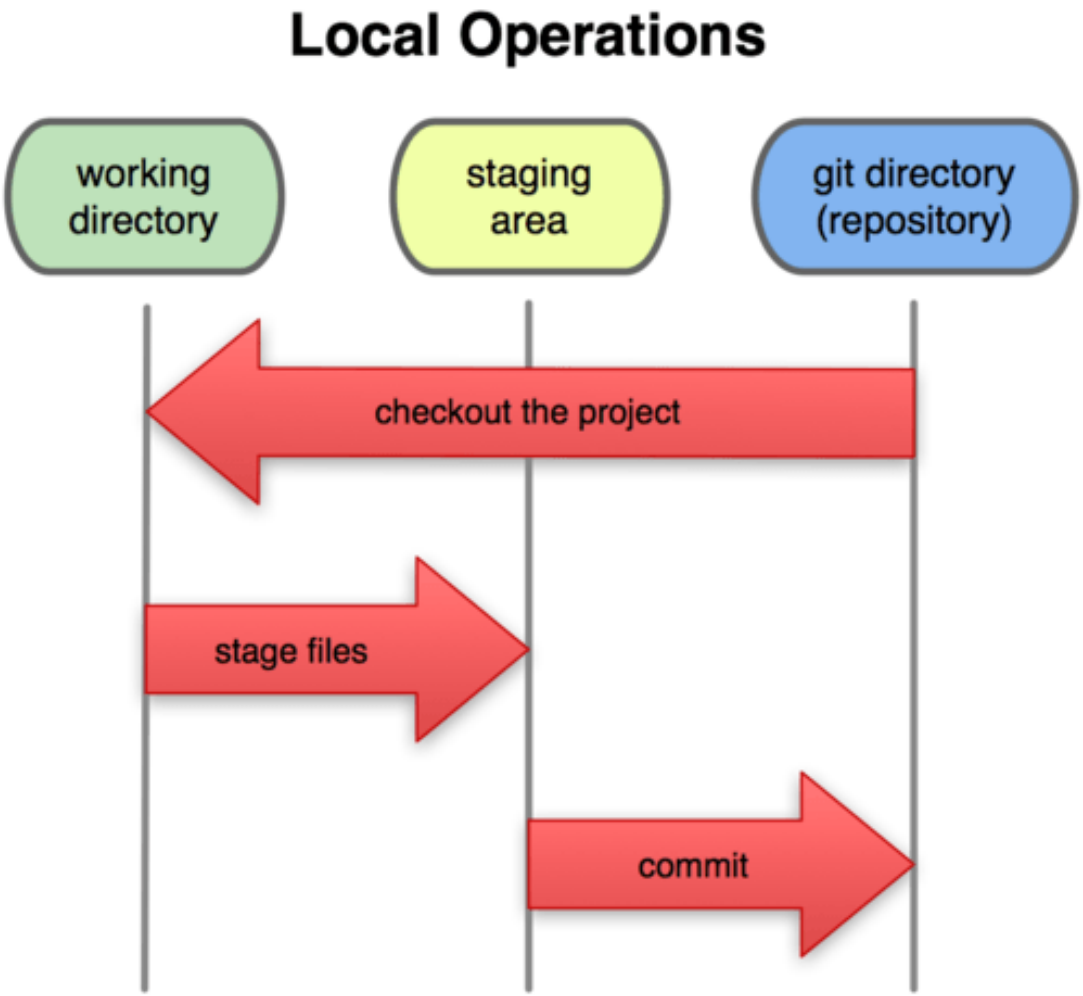
Wykonanie migawki pliku

Add umożliwia wykonanie migawki pliku.

Przełączenie między gałęziami

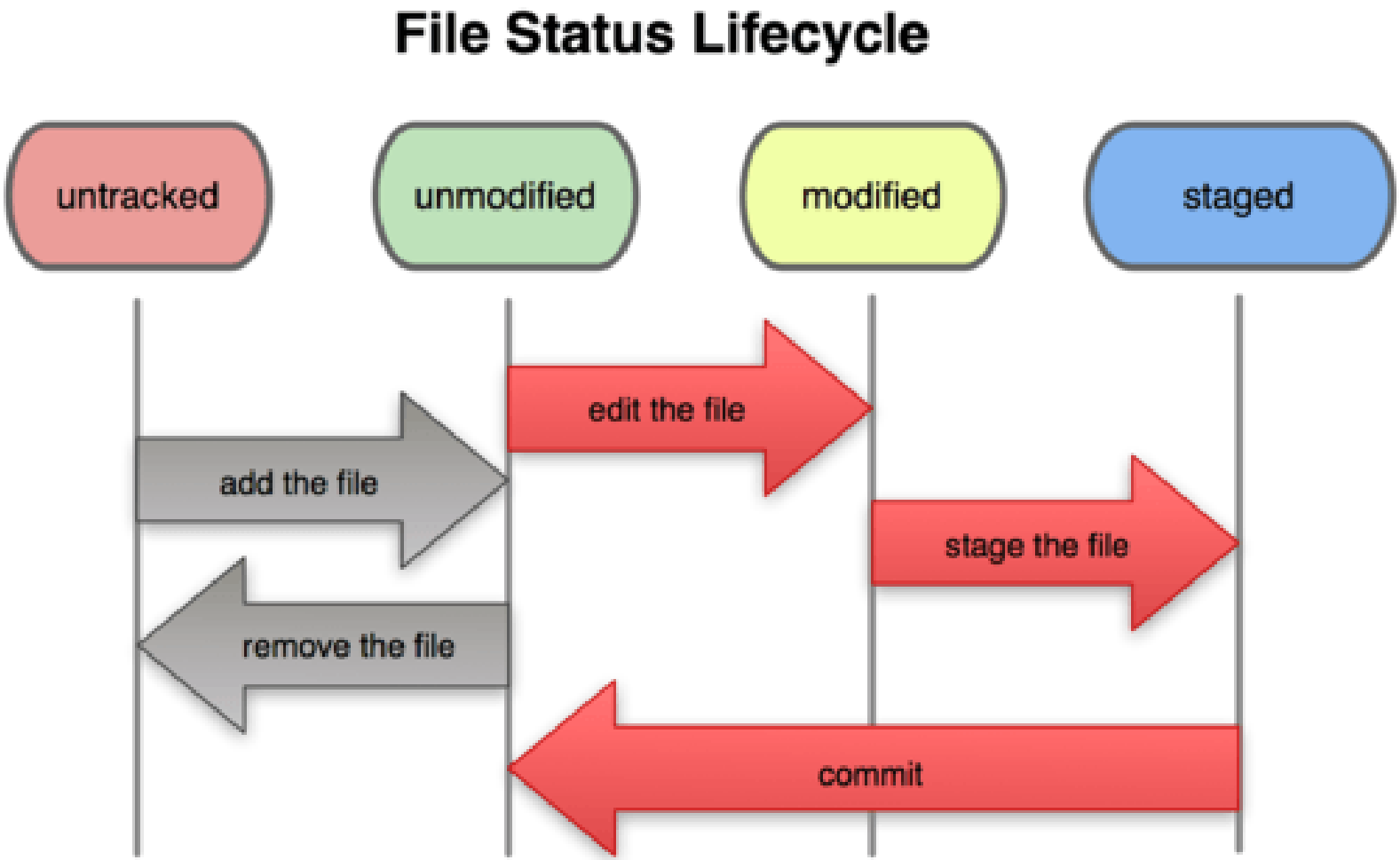
Checkout pozwala na przełączenie między gałęziami (branches).

Schemat działania - etap lokalny



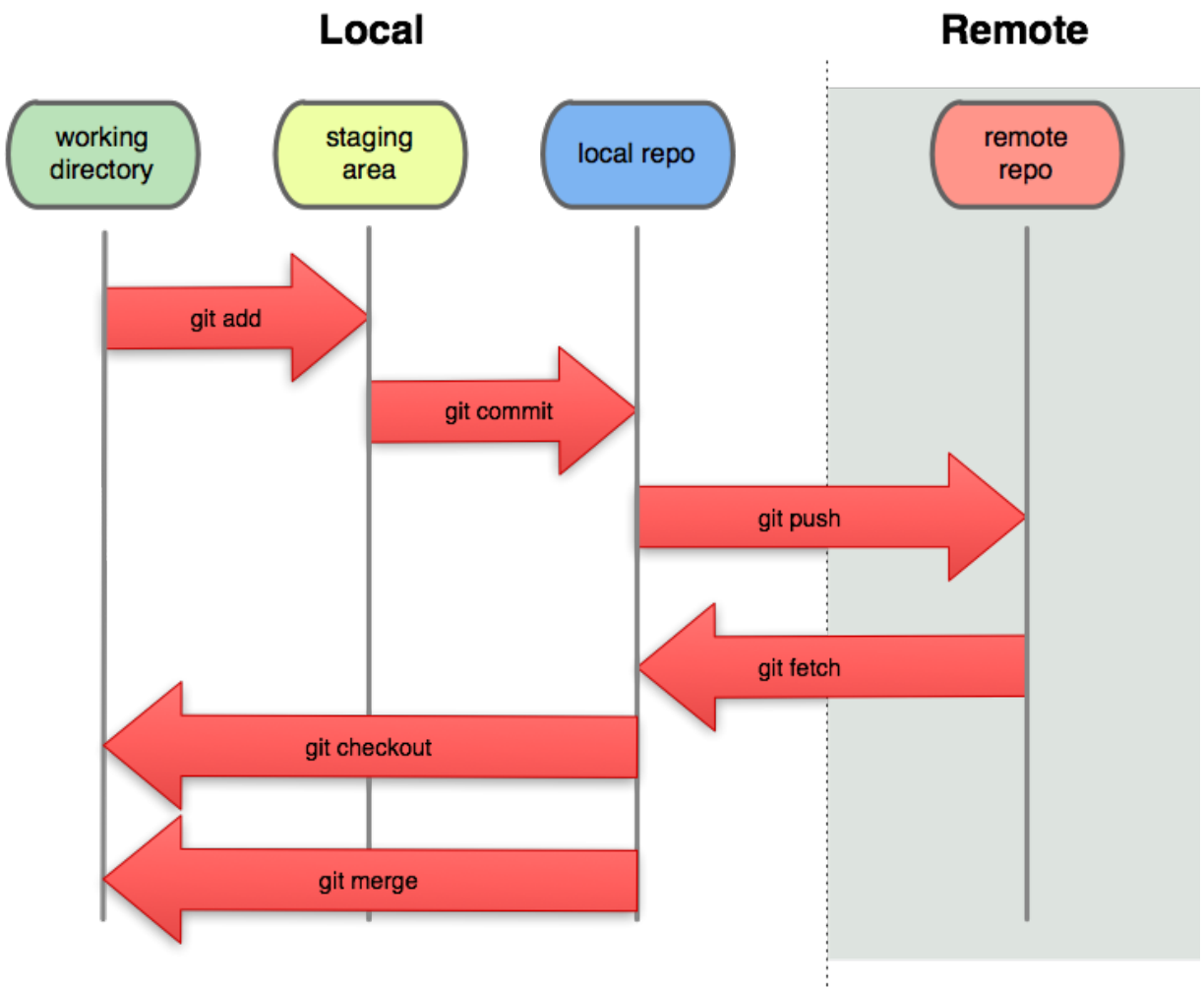
https://git-scm.com/

Schemat działania - życie pliku



https://git-scm.com/

Schemat działania - etap zdalny



http://bartoz.no-ip.org/

Ściąga

GitHub
GIT CHEAT SHEET.
https://services.github.com/on-demand/downloads/github-git-cheat-sheet.pdf

Zadania

1. Załóż konto na `https://github.com`

Zadania

1. Załóż konto na `https://github.com`
2. Pobierz i zainstaluj GitHub Desktop `https://desktop.github.com`

Zadania

1. Załóż konto na `https://github.com`
2. Pobierz i zainstaluj GitHub Desktop `https://desktop.github.com`
3. Wykonaj dołączony do programu tutorial.

Zadania

1. Załóż konto na `https://github.com`
2. Pobierz i zainstaluj GitHub Desktop `https://desktop.github.com`
3. Wykonaj dołączony do programu tutorial.
4. Niech lider założy repozytorium dla zadania projektowego i doda wszystkich członków zespołu.

Zadania

1. Załóż konto na `https://github.com`
2. Pobierz i zainstaluj GitHub Desktop `https://desktop.github.com`
3. Wykonaj dołączony do programu tutorial.
4. Niech lider założy repozytorium dla zadania projektowego i doda wszystkich członków zespołu.
5. Powiążcie projekt z `https://waffle.io`

Git Shell

> „Można zrobić coś w 20 sekund przez GUI, ale kodem wygląda kozacko.”
>
> — Artur Z., programista

Przykład w Git Shell

```
~\Documents\GitHub [master +1 ~0 -0 ~]> cd ..
~\Documents> cd projekt
~\Documents\projekt> git init
Initialized empty Git repository in C:/Users/pc/Documents/projekt/.git/
~\Documents\projekt [master]> git config --global user.name "Prowadzacy"
~\Documents\projekt [master]> git config --global user.email krzysztof@
~\Documents\projekt [master]> git remote add projekt https://github.com/projekt
~\Documents\projekt [master]> git pull projekt
remote: Counting objects: 6, done.
remote: Compressing objects: 100% (4/4), done.
remote: Total 6 (delta 1), reused 5 (delta 0), pack-reused 0
Unpacking objects: 100% (6/6), done.
From https://github.com/projekt
 * [new branch]      czx        -> projekt/czx
 * [new branch]      master     -> projekt/master
You asked to pull from the remote 'projekt', but did not specify
a branch. Because this is not the default configured remote
for your current branch, you must specify a branch on the command line.
~\Documents\projekt [master]> git pull projekt czx:master
From https://github.com/projekt
 * [new branch]      czx        -> master
~\Documents\projekt [master]> git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        nowy.txt

nothing added to commit but untracked files present (use "git add" to track)
~\Documents\projekt [master +1 ~0 -0 !]> git add nowy.txt
~\Documents\projekt [master +1 ~0 -0 ~]> git commit -m "Dodano plik nowy"
[master 3a23b36] Dodano plik nowy
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 nowy.txt
~\Documents\projekt [master]> git push --set-upstream projekt master
Counting objects: 2, done.
Delta compression using up to 8 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (2/2), 241 bytes | 0 bytes/s, done.
Total 2 (delta 1), reused 0 (delta 0)
remote: Resolving deltas: 100% (1/1), completed with 1 local objects.
Branch master set up to track remote branch master from projekt.
To https://github.com/projekt
   5b0d7ca..3a23b36  master -> master
```

Dziękuję za uwagę