Programowanie internetowe

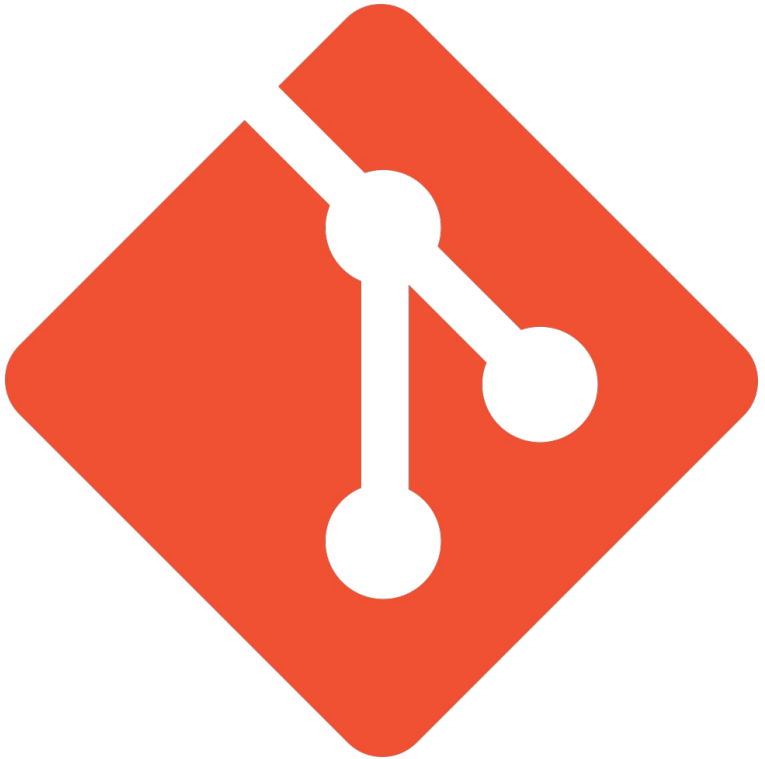
GIT

System kontroli wersji
(the stupid content tracker)

Opracował: inż. Grzegorz Petri

Przegląd zagadnień

- Czym jest GIT, czyli trochę historii
- Cechy systemu
- Idea gałęzi
- Pojęcia i koncepcje
- Zakładanie repozytorium
- Polecenia pracy z Repo



Historia i alternatywy

Co by było gdyby…

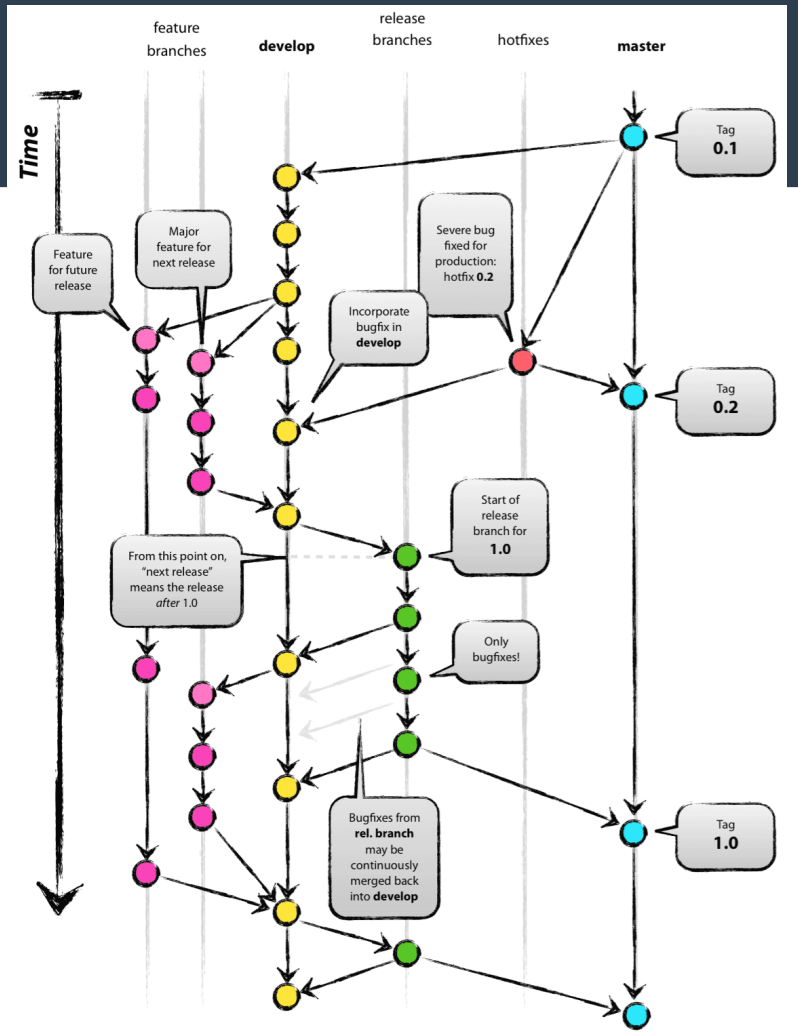
- GIT został stworzony w 2005 roku przez Linusa Torvaldsa dla projektu kernela Linuksa
- Kod Linuksa był rozwijany na własnościowym systemie BitKeeper do 2005 roku, gdy nastąpiła zmiana zasad korzystania z systemu
- Linus szukał alternatywy, ależ żadna nie spełniała jego kryteriów:
  - Patche powinny być stosowane w ciągu **3s**
  - **NIE** stosować podejścia z systemu CVS
  - Wspierać **rozproszony** przepływ pracy
  - Zwiększone **zabezpieczenia** przed uszkodzeniem kodu

Alternatywy

- CVS (Concurent Versions System)
- SVN (subversion system)
- Bazaar
- BitKeeper
- Mercurial
- Rational ClearCase (IBM)

Cechy systemu GIT

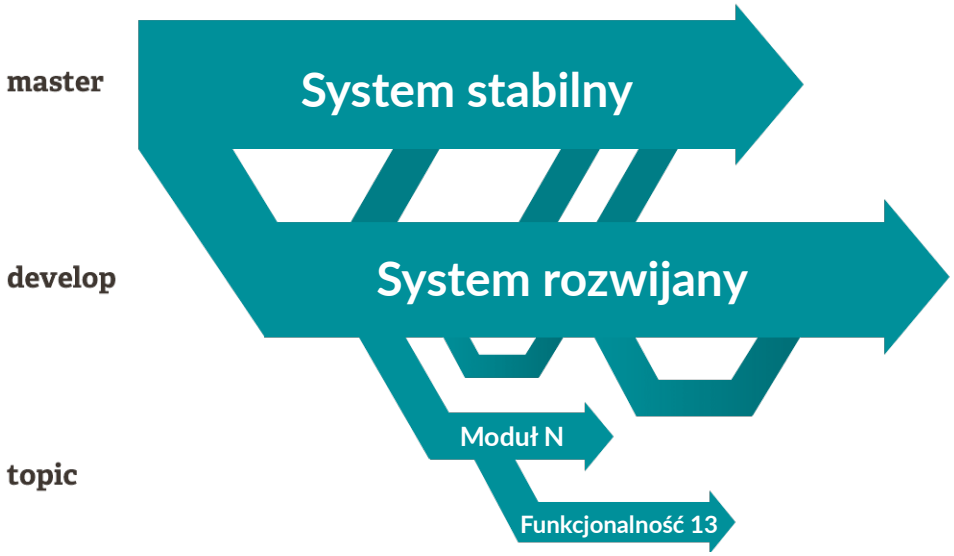
- *Oprogramowanie otwarte i darmowe (GNU/GPLv2)*
- *Wsparcie dla **rozgałęzień** projektu*
- *Praca **off-line** – własna kopia repozytorium*
- ***Współpracuje z istniejącymi protokołami***
  - *HTTP(S), FTP, SSH*
- ***Efektywna praca z dużymi projektami***
  - *jest szybszy niż konkurencyjne rozwiązania*
- ***Snapshoty (wersje) projektu wraz z ich historią:***
  - *system „nie zapamiętuje zmian" przy wykonywaniu rewizji wersji kodu...*
  - *...system wykonuje zrzut CAŁEGO projektu*



Czym jest GIT?

Nowe podejście do pracy

- Bezproblemowe zmiany kontekstu
- Gałęzie/linie kodowe oparte na rolach
- Przepływ pracy oparty na funkcjonalnościach
- Jednorazowe eksperymenty



Witryna projektu: https://git-scm.com/

Słownictwo

- *Working directory (lokalny katalog pracy)*
- *Repo / Repository (repozytorium)*
  - *Local (lokalne repo)*
  - *Remote (zdalne repo)*
- *Origin (adres URL repozytorium)*
- *Snapshot (zrzut stanu projektu - plików)*
  - *Commit (wysłanie / wersja projektu)*
  - *Clone (pobranie projektu z gałęzi master)*
- *Branch (gałąź)*
  - *Master (główna gałąź) / Main / Primary*
  - *<SubBranch> (gałąź wybranej funkcjonalności)*

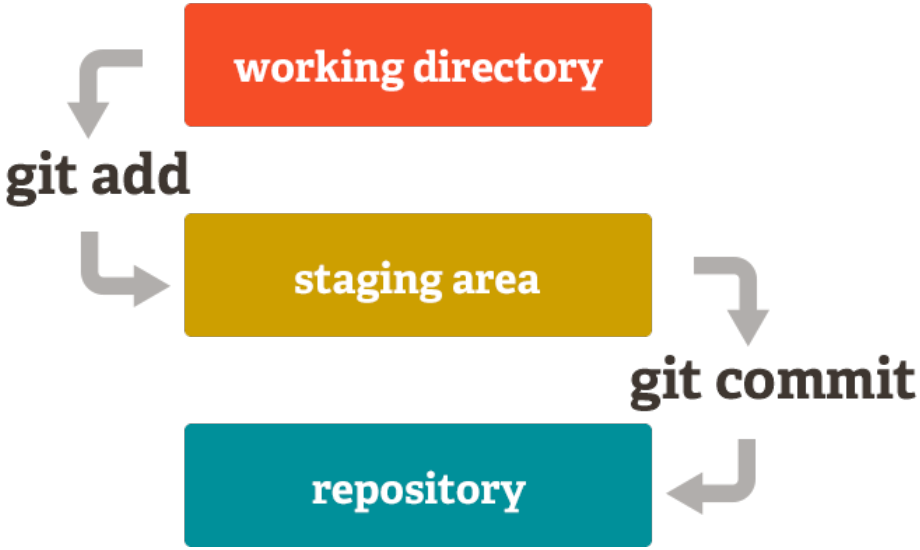
Działania

- Status (wersja, status i historia)
- Branch (listuje lub tworzy gałąź)
- Merge (scal zmiany)
- Push (prześlij do R)
- Pull (pobierz z R)
- Checkout (zmienia gałąź i uaktualnia katalog pracy)
- Reset
- Fetch
- Show
- Diff
- Log
- Stash

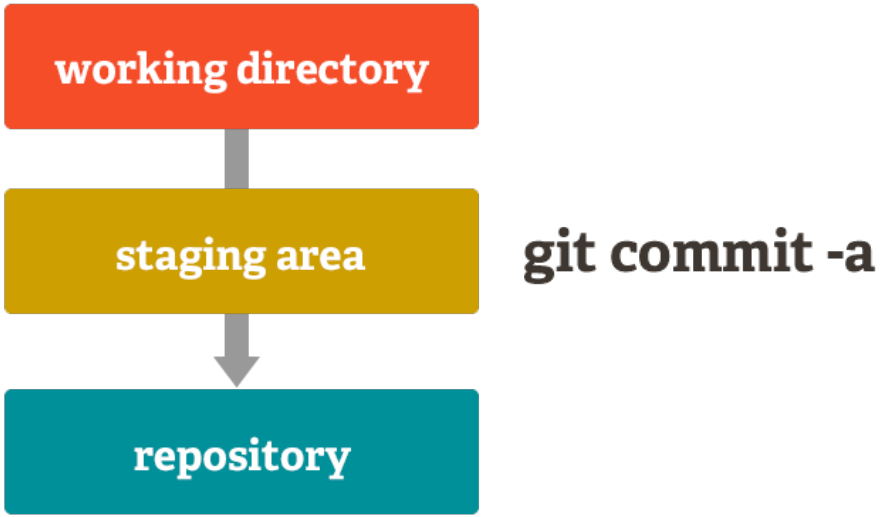
Staging Area

Lista plików do wgrania

Krok po kroku



Szybkie synchro

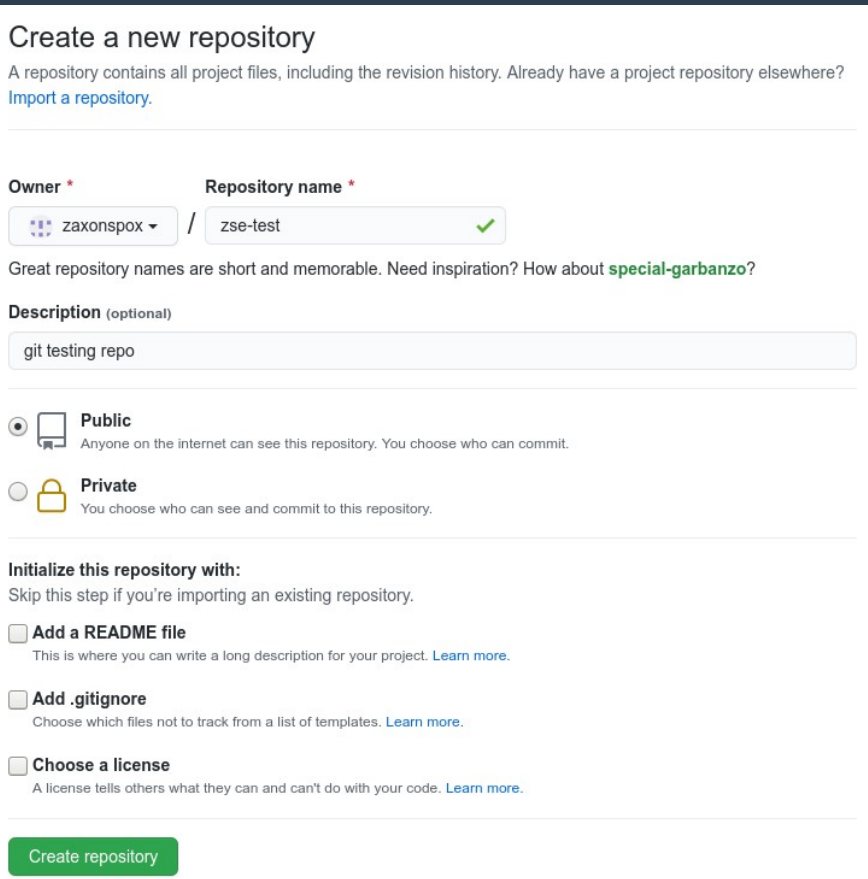


System vs Platforma

GIT ≠ GitHub

System: https://git-scm.com/ Platforma: https://github.com/

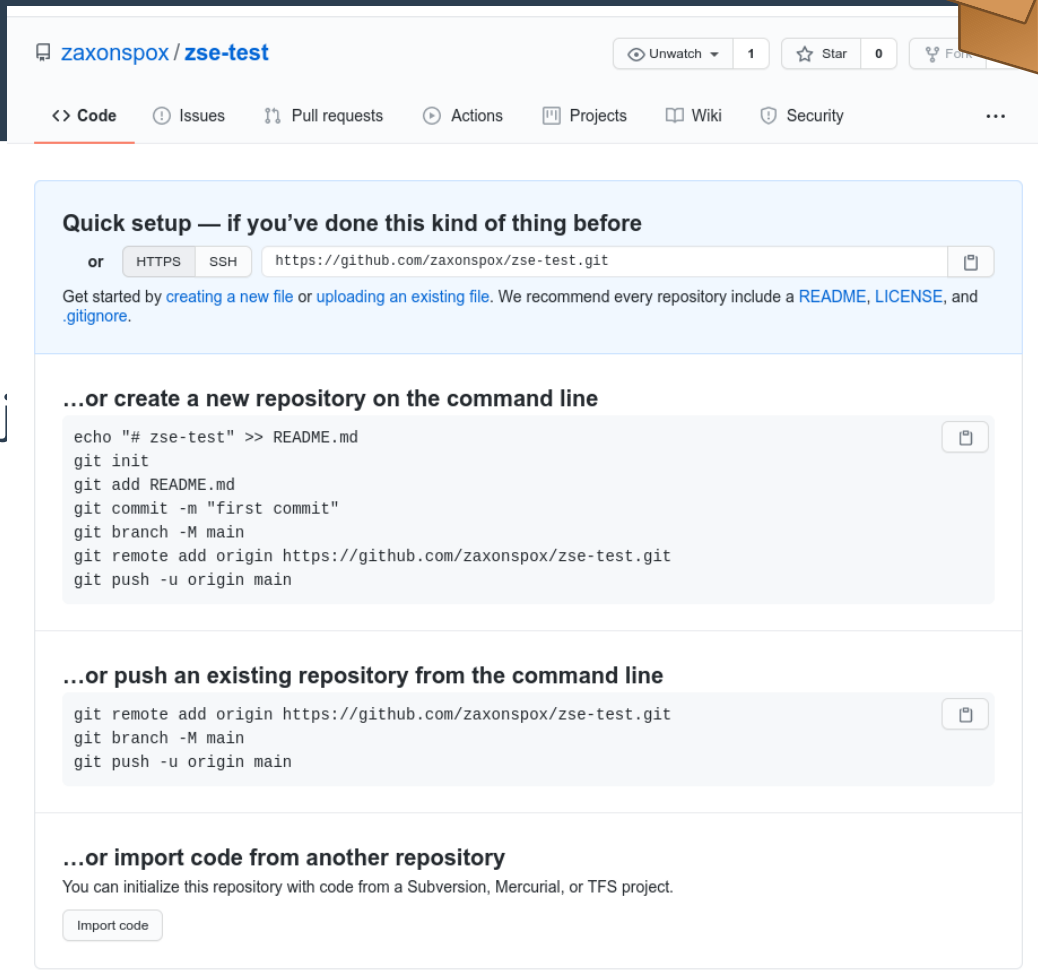
Tworzenie repozytorium



- Nazwa repozytorium
- Opis projektu
- Publiczny / Prywatny
- Tworzenie domyślnych plików informacyjnych (*NIE TWÓRZ ICH !!!*)

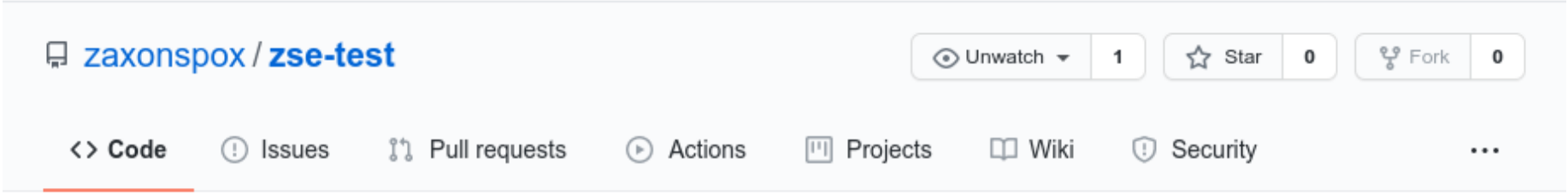
Gotowe repozytorium

- Tworzenie domyślnych plików informacyjnych:
  - README.md – opisujące proj
- Polecenia do rozpoczęcia pracy z projektem:
  - remote add origin – dodanie plików do zdalnego repo
  - wskazanie gałęzi docelowej
  - wypchnięcie zmian



Możliwości repozytorium

- ✔ Śledzenie zgłoszeń błędów
- ✔ Powiadamianie o zmianach
- ✔ Akcje automatyzujące pracę
- ✔ Koordynowanie projektu
- ✔ Dokumentacja w postaci Wiki
- ✔ Określ zasady bezpieczeństwa



Zadania obowiązkowe

1. Załóż konto na platformie GitHub (nazwisko.imie / nazwisko-imie)*
2. Prześlij prowadzącemu adres do konta GitHub
3. Załóż pierwsze repozytorium **Publiczne (KLASA-Nazwisko-IDzadania)****
4. Skonfiguruj informacje o repozytorium (KLASA Nazwisko imię ID-zadania)**
5. Wgrywaj do oddzielnych repozytoriów pliki zadań do oceny

** nazwy konta w stylu: **kroolMaciu$pierwszy** CZY **masterBlaster123** będą skutkować wystawieniem oceny 1*

*** każde Repozytorium ma być **identyfikowalne** względem wykonywanego zadania, np.:*
Zadanie na ocenę: Stworzenie konta oraz repozytorium (ID-zadania podane przez prowadzącego: git-mk-acc)
*Nazwa repozytorium: **1-git-mk-acc***


$ Git questions?