

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...



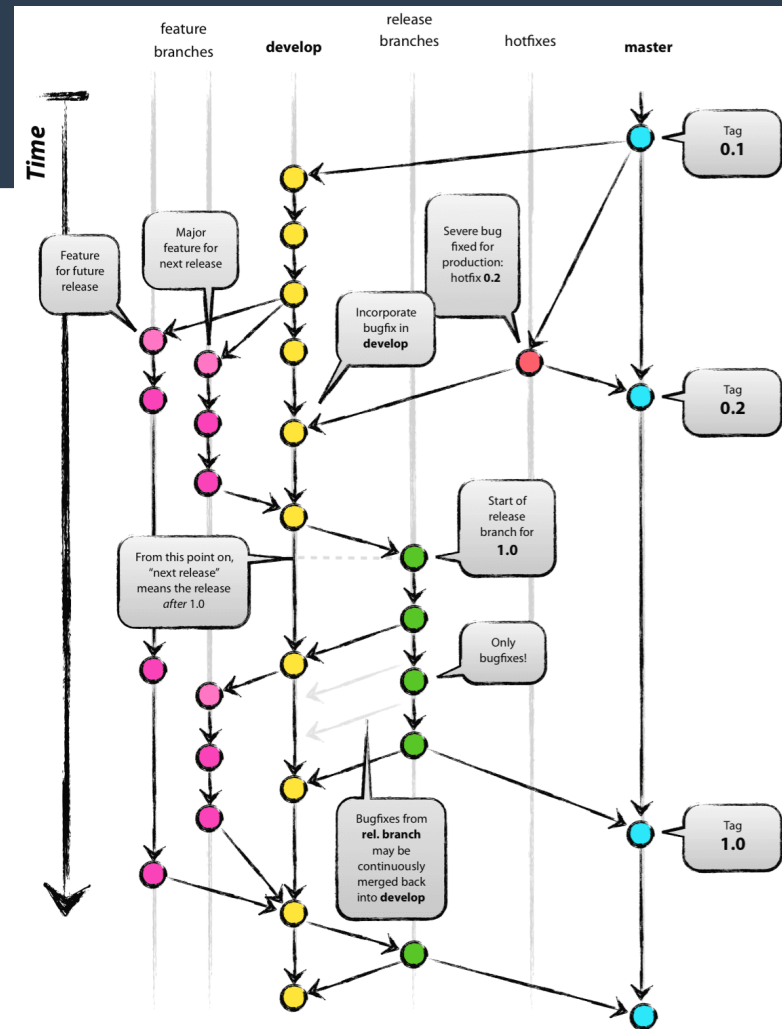
- GIT został stworzony w 2005 roku przez Linusa Torvaldsa dla projektu kernela Linuksa
- Kod Linuksa był rozwijany na własnościowym systemie BitKeeper do 2005 roku, gdy nastąpiła zmiana zasad korzystania z systemu
- Linus szukał alternatywy, ależ żadna nie spełniała jego kryteriów:
 - Patche powinny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

Alternatywy

- CVS (Concurrent 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 rzut CAŁEGO projektu

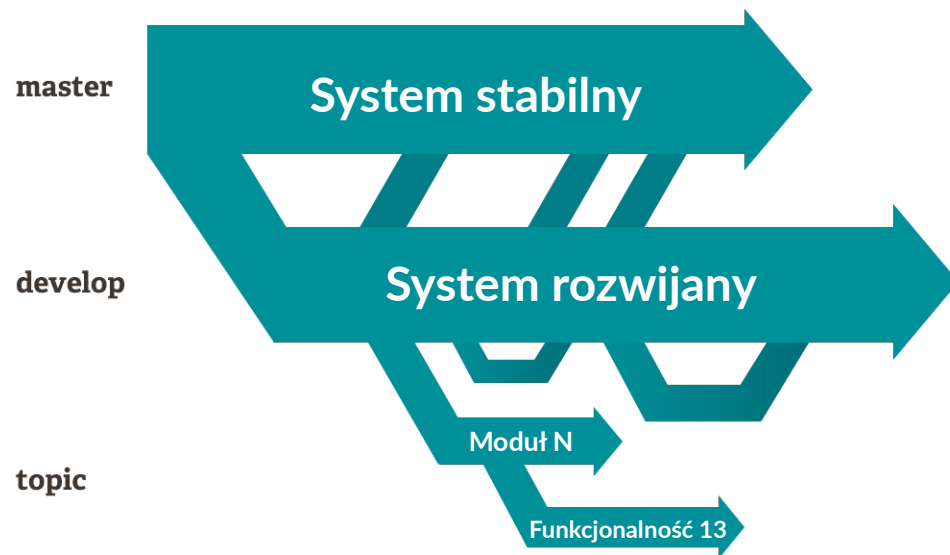


Czym jest GIT?

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/>



- *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)*

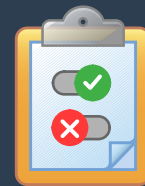
Działania

- *Status (wersja, status i historia)*
- *Branch (listuje lub tworzy gałąź)*
- *Merge (scal zmiany)*
- *Push (prześlij do R)*
- *Pull (pobierz z R)*

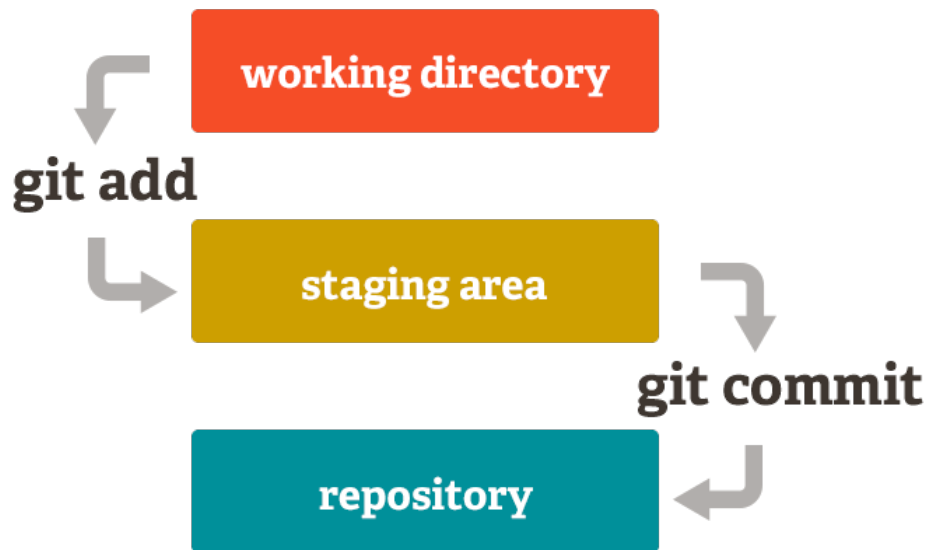
- *Checkout (zmienia gałąź i uaktualnia katalog pracy)*
- *Reset*
- *Fetch*
- *Show*
- *Diff*
- *Log*
- *Stash*

Staging Area

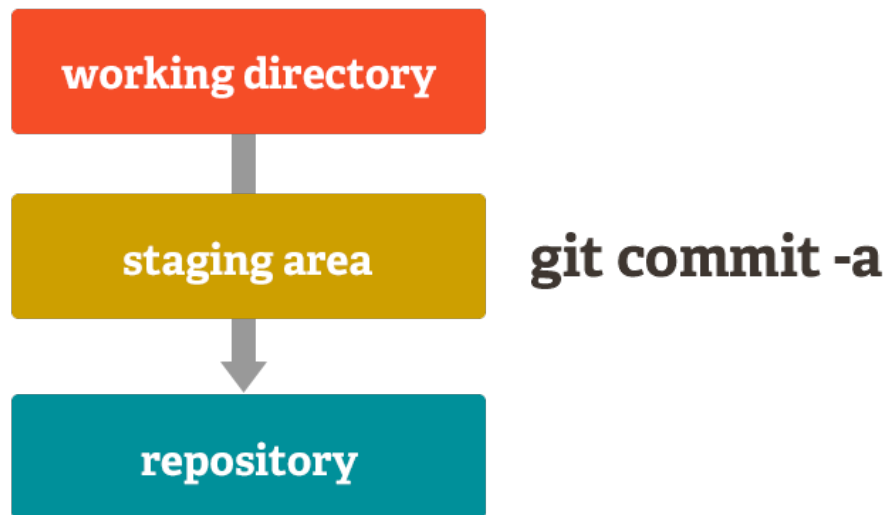
Lista plików do wgrania



Krok po kroku



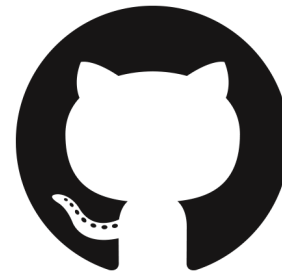
Szybkie synchro



System vs Platforma



GIT \neq GitHub



System: <https://git-scm.com/>

Platforma: <https://github.com/>

Tworzenie repozytorium



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner *

zaxonspox

Repository name *

zse-test

Great repository names are short and memorable. Need inspiration? How about [special-garbanzo?](#)

Description (optional)

git testing repo

☒ **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐ **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☐ **Add a README file**

This is where you can write a long description for your project. [Learn more.](#)

☐ **Add .gitignore**

Choose which files not to track from a list of templates. [Learn more.](#)

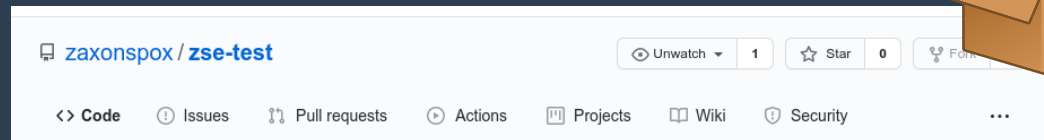
☐ **Choose a license**

A license tells others what they can and can't do with your code. [Learn more.](#)

Create repository

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

Gotowe repozytorium



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

Quick setup — if you've done this kind of thing before

or ☐ HTTPS ☐ SSH

Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).

...or create a new repository on the command line

```
echo "# zse-test" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/zaxonspox/zse-test.git
git push -u origin main
```

...or push an existing repository from the command line

```
git remote add origin https://github.com/zaxonspox/zse-test.git
git branch -M main
git push -u origin main
```

...or import code from another repository

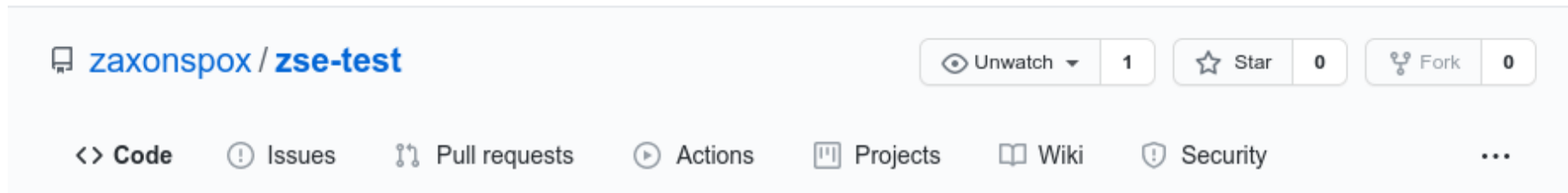
You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

[Import code](#)

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?