

## Użytkowanie systemów kontroli wersji.

Tworzenie wszelakiego oprogramowania od dłuższego czasu stało się niezwykle wymagające. Obecnie istnieje niewiele aplikacji/programów, których kod może zamknąć się w kilkuset liniach kodu. Chociaż małe aplikacje nadal mogą być rozwijane przez pojedyncze osoby to jednak coraz częściej w projektach biorą udział co najmniej zespoły dwuosobowe; przy większych rozwiązaniach liczba ta sięga nawet kilkudziesięciu osób pracujących tylko przy jednym z modułów całego projektu.

Rozwijanie w tych warunkach kodu byłoby co najmniej karkołomne z kilku powodów. Najbardziej prozaicznym jest oczywiście dostęp do pojedynczego pliku z kodem. W danej chwili tylko jedna osoba miałaby możliwość edycji/pisania w danym pliku. Innym problemem byłaby możliwość testowania dodawanej przez programistę funkcjonalności – musiałby on czekać aż pozostałe osoby udostępnią mu aktualne postępy w ich kodzie by on mógł zweryfikować własny kod. Do tego dochodzą ustalenia odpowiedzialności za kod, kto dodał niedziałające rozwiązanie itp. Wisienką na torcie jest brak możliwości cofnięcia zmian – uczestnicy projektu sami musieliby zadbać o robienie zrzutu poprzednich wersji rozwiązania by w razie problemów móc wrócić do działającej wersji.

Powyżej wymienione kłopoty powinny niemal każdego przekonać, że nawet najprostszy projekt wymaga dodatkowego, dedykowanego narzędzia służącego do zarządzania kodem. Narzędzia, do którego dostęp będą mieli wszyscy pracujący przy kodzie, pozwalającego na wgląd w starsze wersje plików źródłowych/binarnych, pozwalającego na natychmiastowe aktualizacje oraz testy pisanego kodu. Obecnie na rynku jest kilka rozwiązań, które pozwalają na kontrolowanie kodu rozwijanego projektu (zresztą nie tylko kodu, a niemal wszystkich plików). Coraz częściej jednak wybieranym narzędzie jest git (akronim nie posiada jednoznacznego rozwinięcia).

Git został zaprojektowany przez Linusa Torvaldsa na potrzeby rozwoju jądra systemu Linux. Początkowo wykorzystywany był jedynie w gronie programistów tego systemu oraz programistów tworzących narzędzia do systemów Unix/Unix-like. Jednak jego prostota, łatwość użytkowania oraz niemal zerowe wymagania sprawiły, że coraz chętniej zaczęli po niego sięgać także „więksi”, jak firmy motoryzacyjne (np. Volvo, Toyota), firmy zajmujące się oprogramowaniem (np. Microsoft) oraz administratorzy sieci firmowych, w których przechowywane są różnego rodzaju dokumenty (nie będące kodem). Obecnie narzędzie to niemal dominuje rynek narzędzi kontroli wersji, wypierając pozostałe rozwiązania (SVN, bezpośrednia „konkurencja”, posiada już ok. 20% udziału w rynku, chociaż jeszcze 4 lata temu miała ok 80%).

Co spowodowało, że git stał się tak popularny? Przede wszystkim niezależność – każdy członek projektu posiada na swoim komputerze kopię centralnego repozytorium, którym może dowolnie zarządzać – tworzyć zapisy, cofać wersje, weryfikować zawartość plików itp. Opcja ta jest szczególnie ważna w przypadku gdy pracujemy bez dostępu do sieci – pozostałe rozwiązania nie dają nam w takim wypadku jakiegokolwiek możliwości do ingerowania w repozytorium.

Drugą ważną sprawą jest możliwość rozgałęziania projektów. Mianowicie każdy członek projektu może pracować nad własną wersją aplikacji w niezależnej przestrzeni od pozostałych uczestników. Gałąź projektu może bowiem posiadać zupełnie inne wersje plików (nawet posiadać ich więcej/mniej), które później można włączać do innych gałęzi (bądź włączyć zawartość pozostałych gałęzi do naszych). Ponadto użytkownicy mogą mieć kilka niezależnych gałęzi, nad którymi pracują!

Trzecim argumentem na plus dla git jest system zapisywania plików. Konkurencyjne rozwiązania tworzą bowiem każdorazowo kopię każdego pliku projektu do dodawanej rewizji (zapisu w projekcie). W git natomiast zapisywane są jedynie fragmenty zmian. Dzięki temu repozytoria git są znacznie mniejsze, bardziej elastyczne i pozwalają na szybkie przemieszczanie się pomiędzy wersjami (rewizjami) projektu.

Oczywiście narzędzie posiada pewne niedoskonałości, jednak są one niemal zerowe. Jedną z niedogodności może być np. pozorne skomplikowanie użytkowania narzędzia. Jednak z czasem okaże się, że w większości wypadków potrzebne nam będą jedynie podstawowe funkcje.

Git, przez swoją budowę oraz licencje (zupełnie darmowy) jest bardzo dobrze opisany na stronie głównej projektu. W języku polskim jest dostępne starsze wydanie książki na temat jego użytkowania; można również zapoznać się z darmową wersją najnowszej książki (jednak jest ona w całości po angielsku, niemiecku, francusku). Ponieważ obie pozycje w pełni wyczerpują temat działania i użytkowania systemu git, poniższy materiał skupi się na wyborze zewnętrznego serwera repozytorium oraz, dla bardziej zaawansowanych użytkowników, o utworzeniu własnego serwera repozytorium. Ponadto zostanie przedstawione narzędzie znacznie ułatwiające pracę z repozytorium.

Odnośniki do materiałów:

<https://git-scm.com> - strona główna projektu

<https://git-scm.com/book/pl/v1> – polska wersja wydanej w 2009 roku książki na temat git

<https://git-scm.com/book/en/v2> – angielska wersja wydanej w 2014 roku książki na temat git;

większość tematyki pokrywa się z pierwszym wydaniem, opisane zostały natomiast nowe rozwiązania i funkcje dostępne w 2 wersji git.

## 1. Wybór serwera repozytorium.

Git jako narzędzie nie potrzebuje serwera. Można go używać lokalnie, poprzez utworzenie repozytorium w katalogu zawierającym pliki/podkatalogi, które mają stanowić repozytorium. Takie rozwiązanie będzie całkowite zgodne ze sztuką zwłaszcza w przypadku, gdy nad projektem pracujemy sami, do tego zawsze na tym samym komputerze/nośniku danych. Sprawa nieco się skomplikuje jeżeli nad kodem pracujemy na różnych komputerach, do tego chcemy co jakiś czas udostępnić go osobom trzecim (np. drugiemu programiście, wykładowcy, znajomym). Wtedy mamy do dyspozycji następujące rozwiązania:

- kopiować ze sobą folder repozytorium
- przesyłać go na swój dysk wirtualny/pocztę
- utworzyć własny serwer repozytorium
- założyć konto w odpowiedniej usłudze

Ostatnia możliwość jest najprostsza dla osób, które w prosty sposób chcą cieszyć się przenośnością swoich repozytoriów.

W sieci znajdziemy co najmniej kilkanaście serwisów oferujących możliwość założenia kont, na których możemy „powiesić” nasze przyszłe projekty. Niektóre z nich są płatne, niektóre zupełnie darmowe. Jedne oferują dodatkowe narzędzia, inne jedynie przestrzeń. To, którego operatora wybierzemy zależy tylko od nas i naszych potrzeb. W poniższym materiale opisane zostanie założenie konta oraz zarządzanie projektem w serwisie gitlab.com. Wybór tego serwisu został uwarunkowany następującymi czynnikami:

- jeden z niewielu o otwartym źródle (dostępny jest kod źródłowy jego wszystkich narzędzi, jak i samego serwisu)
- zezwala na nielimitowaną ilość projektów (w zasadzie do 10.000 projektów na użytkownika; prócz tego możemy dołączać do dowolnej ilości projektów)
- zezwala na dodanie nielimitowanej ilości użytkowników do projektów prywatnych; nie ogranicza także zakładania i wykorzystywania projektów prywatnych
- nie limituje miejsca na nasze projekty
- działa z większością narzędzi kontroli wersji na rynku

Aby założyć konto na GitLab przechodzimy pod adres

<https://gitlab.com>

about.gitlab.com

GitLab

Download Features Pricing Community Explore Blog Sign In Sign Up

#AmplifyYourCode on the GitLab World Tour! See where we'll be!

Tools for modern developers

GitLab unifies issues, code review, CI and CD into a single UI

Sign Up Learn More

Klikamy Gign up (niebieska ramka)

gitlab.com/users/sign\_in



## GitLab.com

GitLab.com offers free unlimited (private) repositories and unlimited collaborators, please sign up or in on the right.

- [Explore projects on GitLab.com](#) (no login needed)
- [More information about GitLab.com](#)
- [GitLab.com Support Forum](#)

By signing up for and by signing in to this service you accept our:

- [Privacy policy](#)
- [GitLab.com Terms](#).


Sign in Register

Name  
Tezcatlipoca

Username  
Username is available.





Email  
.

Password  
Minimum length is 8 characters

Nie jestem robotem  reCAPTCHA  
Prywatność · Warunki

Register

Didn't receive a confirmation email? [Request a new one.](#)

Sign in with    

Podajemy nazwę użytkownika, adres poczty oraz hasło; możemy także autoryzować się kontem z Google, Twitter, github bądź bitbucket (dwa ostatnie to podobne serwisy do gitlab).

**INFORMACJA:** Chociaż brzmi niezwykle wygodnie, lepiej nie logować się strony gitlab przy użyciu konta z innej strony. Proszę pamiętać, że strony te mogą uzyskiwać dostęp do informacji na temat konta utworzonego w ten sposób co niekoniecznie musi być nam na rękę!

Po wykonaniu tych czynności zostaniemy poinformowani o przesłaniu na nasz adres informacji potwierdzającej założenie konta w portalu gitlab.com. Strona poinformuje nas o oczekiwaniu na potwierdzenie informacji...



# Almost there...

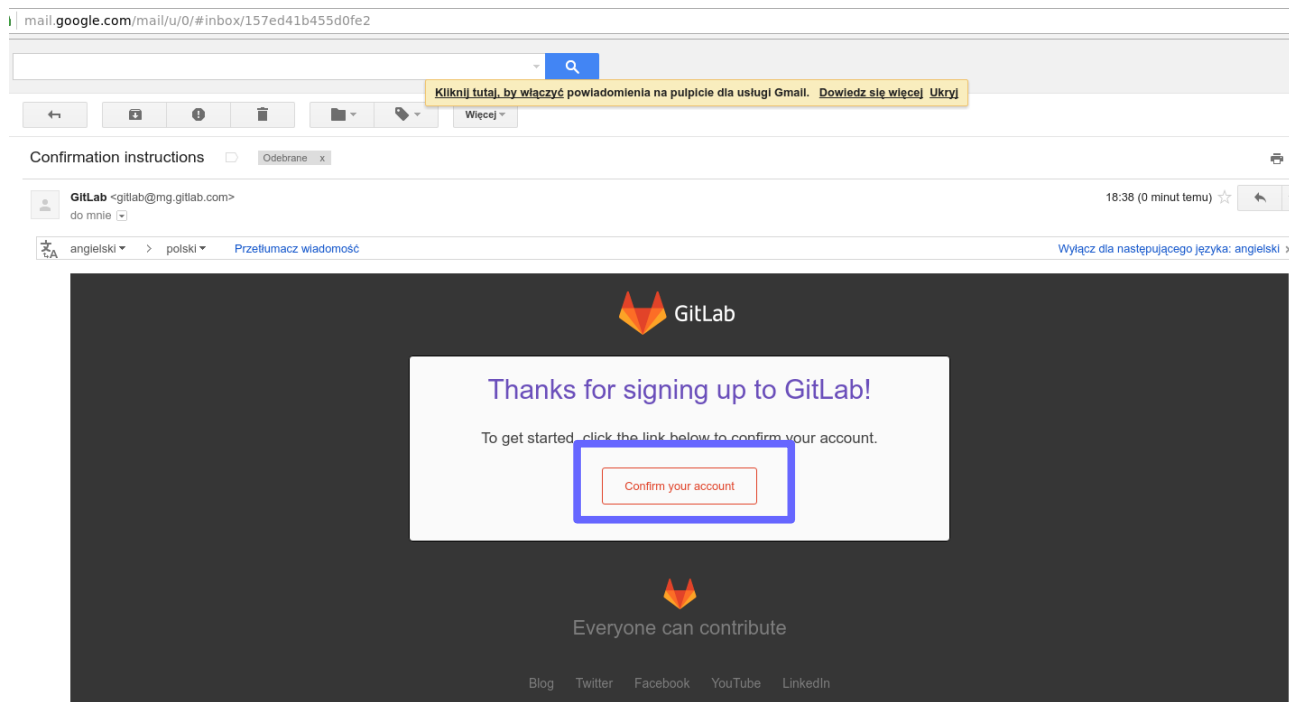
Please check your email to confirm your account

No confirmation email received? Please check your spam folder or

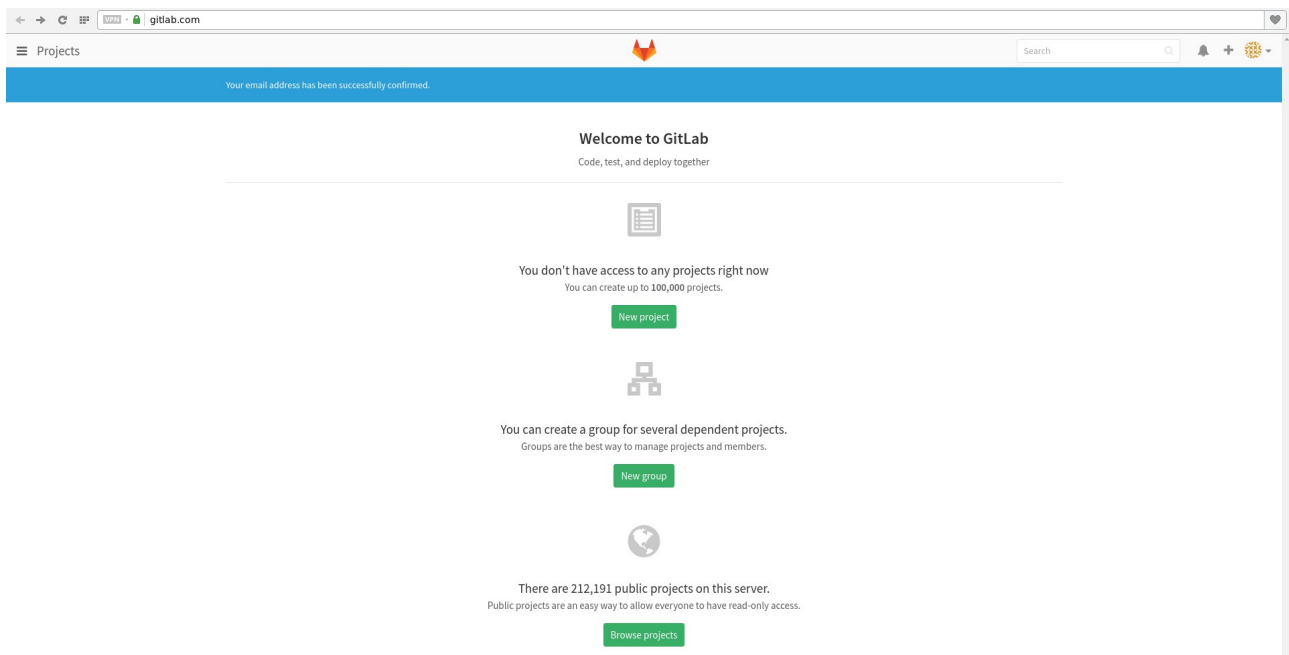
[Request new confirmation email](#)

[Explore](#) [Help](#) [About GitLab](#)

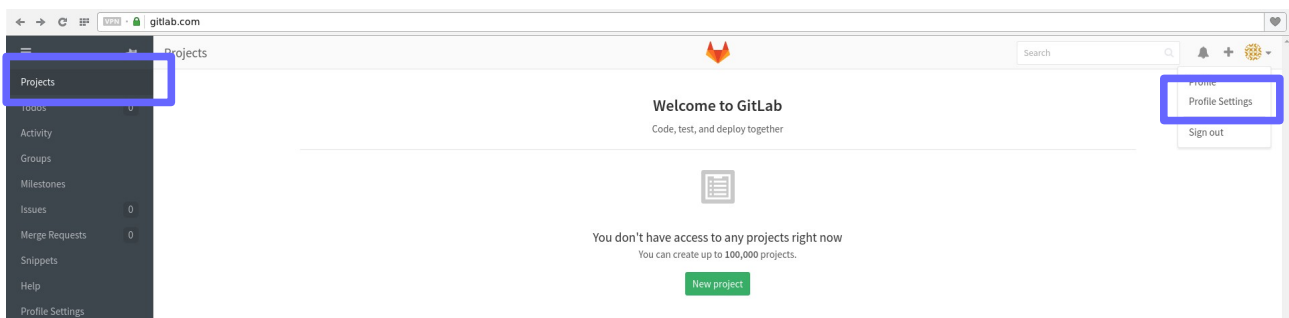
Poczta z prośbą o potwierdzenie wygląda tak jak na zrzucie poniżej. Wystarczy kliknąć w odpowiedni przycisk (niebieska ramka):



Zaraz po zaakceptowaniu przejdziemy na stronę główną naszego konta w portalu. Wyświetlone zostaną nam podstawowe możliwości – założenie projektu, grup bądź możliwość przejrzania publicznych projektów.




Nas jednak aktualnie powinna zainteresować możliwość edycji profilu (wejście do opcji zaznaczone niebieskim prostokątem):



---


---



[Profile](#) [Account](#) [Applications](#) [Access Tokens](#) [Emails](#) [Password](#) [Notifications](#) [SSH Keys](#) [Preferences](#) [Audit Log](#)

---

**Public Avatar**  
You can upload an avatar here or change it at [gravatar.com](https://gravatar.com)



**Upload new avatar**  
   
The maximum file size allowed is 200KB.

---

**Main settings**  
This information will appear on your profile.

**Name**  
  
Enter your name, so people you know can recognize you.

**Email**  
  
We also use email for avatar detection if no avatar is uploaded.

**Public email**  
  
This email will be displayed on your public profile.

**Skype**

**LinkedIn**

**Twitter**

**Website**


**Location**

**Organization**

**Bio**  
  
Tell us about yourself in fewer than 250 characters.

---

Na pierwszej stronie mamy naszą nazwę użytkownika oraz możliwość ustawienia: obrazka reprezentującego nasz profil, ustawienie adresu poczty (bądź zmianę aktualnego na nowy), ustawienie widoczności adresu (domyślnie nie jest pokazywany), czy też ustawienie pozostały opcji kontaktowych (domyślnie puste).



Profile **Account** Applications Access Tokens Emails Password Notifications SSH Keys Preferences Audit Log

---

**Private Token**

Your private token is used to access application resources without authentication.

**Private token**

It can be used for atom feeds or the API. Keep it secret!

[Reset private token](#)

---

**Two-Factor Authentication**

Increase your account's security by enabling Two-Factor Authentication (2FA).

Status: Disabled

[Enable Two-Factor Authentication](#)


---


**Social sign-in**


Activate signin with one of the following services


**Connected Accounts**

Click on icon to activate signin with one of the following services

 [Connect](#)

 [Connect](#)

 [Connect](#)

 [Connect](#)

---

**Change username**

Changing your username will change path to all personal projects!

**Path**

Current path: https://gitlab.com/t/...

[Update username](#)

---

**Remove account**

Deleting an account has the following effects:


- All user content like authored issues, snippets, comments will be removed

[Delete account](#)

Zakładka Account pozwala na ustawienie tzw. prywatnego żetonu (Private Token). Żeton można wykorzystywać jako autoryzację naszego konta w programach zarządzających wersjami oprogramowania. Dzięki temu nie musimy podawać ciągle naszego hasła. Numer ten powinien pozostawać tajemnicą! Jeżeli mamy podejrzenie, że został on nam skradziony zalecane jest wygenerowanie go na nowo (trzeba będzie na nowo podać go we wszystkich programach, w których został on wpisany).

Drugą ważną opcją jest możliwość autoryzacji dwuetapowej. Sprowadza się ona do dodatkowego potwierdzenia naszej tożsamości, np. poprzez zewnętrzną aplikację dostępną na nasz telefon. Włączenie tej opcji zwiększa bezpieczeństwo naszego konta, jednak może także nieco skomplikować do niego dostęp.

W kolejnej linii mamy możliwość sparowania naszego konta z innymi portalami/usługami. Ponadto w tej zakładce możemy spróbować zmienić nazwę konta bądź je usunąć.



Profile Account **Applications** Access Tokens Emails Password Notifications SSH Keys Preferences Audit Log

Search

---

**Applications**

Manage applications that can use GitLab as an OAuth provider, and applications that you've authorized to use your account.

**Add new application**

**Name**

**Redirect URI**

Use one line per URI

[Save application](#)

**Your applications (0)**

You don't have any applications

**Authorized applications (0)**

You don't have any authorized applications

Zakładka Applications można dodać zewnętrzne aplikacje zarządzające kontem na gitlab. Opcja przydatna w chwili, gdy posiadamy konta na innych serwisach tego typu i chcemy poprzez tamte serwisy zarządzać kontem gitlab.

**Personal Access Tokens**

You can generate a personal access token for each application you use that needs access to the GitLab API.

You can also use personal access tokens to authenticate against Git over HTTP. They are the only accepted password when you have Two-Factor Authentication (2FA) enabled.

**Add a Personal Access Token**

Pick a name for the application, and we'll give you a unique token.

Name

Expires at

Create Personal Access Token

**Active Personal Access Tokens (0)**

You don't have any active tokens yet.

**Inactive Personal Access Tokens (0)**

There are no inactive tokens.

Jeżeli chcemy nadać jakiejś zewnętrznej aplikacji możliwość ingerencji w konto gitlab poprzez jego API (rozwijamy np. własną aplikację pozwalającą na zarządzanie kontem) to tutaj możemy wygenerować żetony dostępowe. Generowane żetony mogą mieć swój czas życia (datę wygaśnięcia). Jeżeli nie będziemy pisać takie aplikacji to zakładka jest dla nas zbędna.

**Emails**

Control emails linked to your account

**Add email address**

Email

Add email address

**Linked emails (1)**

- Your Primary Email will be used for avatar detection and web based operations, such as edits and merges.
- Your Notification Email will be used for account notifications.
- Your Public Email will be displayed on your public profile.
- All email addresses will be used to identify your commits.

tlatocateculti@gmail.com Primary Email Notification Email

Do konta gitlab można przypisać więcej niż jeden adres poczty. Opcja ta jest przydatna w chwili, gdy chcemy na różne konta otrzymywać różnego rodzaju informacje, np. o zaproszeniach do projektów, o zgłoszonych błędach do naszych projektów itd.

Profile Account Applications Access Tokens Emails **Password** Notifications SSH Keys Preferences Audit Log

### Password

After a successful password update, you will be redirected to the login page where you can log in with your new password.

#### Change your password or recover your current one

Current password

You must provide your current password in order to change it.

New password

Password confirmation

[Save password](#) [I forgot my password](#)

Następna zakładka pozwala na zmianę hasła do konta; jeżeli nie pamiętamy obecnego hasła to z tej zakładki możemy je odzyskać.

Profile Account Applications Access Tokens Emails Password **Notifications** SSH Keys Preferences Audit Log

### Notifications

You can specify notification level per group or per project.

By default, all projects and groups will use the global notifications setting.

#### Global notification settings

Notification email

Global notification level

Participate

Groups (0)

Projects (0)

To specify the notification level per project of a group you belong to, you need to visit project page and change notification level there.

Profile Account Applications Access Tokens Emails Password Notifications **SSH Keys** Preferences Audit Log

### SSH Keys

SSH keys allow you to establish a secure connection between your computer and GitLab.

#### Add an SSH key

Before you can add an SSH key you need to [generate it](#).

Key

Don't paste the private part of the SSH key. Paste the public part, which is usually contained in the file '~/.ssh/id\_rsa.pub' and begins with 'ssh-rsa'.


Title

[Add key](#)

Your SSH keys (0)

There are no SSH keys with access to your account.

Ta zakładka pozwala na dodanie klucza SSH do naszego konta. Dzięki niemu będziemy mogli korzystać z klonowania/uaktualniania/pobierania projektów bez podawania hasła do konta. Opcja działa jedynie z połączeniami SSH (nie działa dla połączeń HTTPS).




Search


[Profile](#)
[Account](#)
[Applications](#)
[Access Tokens](#)
[Emails](#)
[Password](#)
[Notifications](#)
[SSH Keys](#)
[Preferences](#)
[Audit Log](#)

### Application theme


This setting allows you to customize the appearance of the site, e.g. the sidebar.




Graphite




Charcoal




Green



Gray



Violet



Blue

### Syntax highlighting theme

This setting allow you to customize the appearance of the syntax.

```
1 class DoctypesCont
2 def index
3   @doctypes = Dc
4 end
5
6 def show
```

White

```
1 class DoctypesCont
2   def index
3     @doctypes = Dc
4   end
5
6   def show
```

Dark

```
1 class DoctypesCont
2   def index
3     @doctypes = Dc
4   end
5
6   def show
```

Solarized Light

```
1 class DoctypesCont
2   def index
3     @doctypes = Dc
4   end
5
6   def show
```

Solarized Dark

```
1 class DoctypesCont
2   def index
3     @doctypes = Dc
4   end
5
6   def show
```

Monokai

### Behavior

This setting allows you to customize the behavior of the system layout and default views.

**Layout width**

Fixed ▼

Choose between fixed (max. 1200px) and fluid (100%) application layout.

**Default Dashboard (?)**

Your Projects (default) ▼

**Project view (?)**


Readme (default) ▼

Choose what content you want to see on a project's home page.

[Save changes](#)

Przedostatnia zakładka pozwala na zmianę wyglądu poszczególnych podstron gitlab. Zmiany dokonane w tej zakładce są przeważnie kosmetyczne i nie mają wpływu na mechanikę naszych repozytoriów.

Po dokonaniu stosownych zmian (jeżeli w ogóle jakichkolwiek dokonaliśmy) możemy przejść do utworzenia pierwszego projektu. W tym celu przechodzimy na stronę główną naszego konta bądź klikamy opcję Projects z menu po lewej stronie ekranu.




Search

## Welcome to GitLab


Code, test, and deploy together

---



You don't have access to any projects right now  
You can create up to 100,000 projects.

[New project](#)



Search

---

**New project**  
Create or Import your project from popular Git services

**Project path**

**Project name**

Want to house several dependent projects under the same namespace? [Create a group](#)

**Import project from**

**Project description (optional)**


**Visibility Level (?)**

**Private**  
Project access must be granted explicitly to each user.

**Internal**  
The project can be cloned by any logged in user.

**Public**  
The project can be cloned without any authentication.

Podajemy nazwę naszego projektu (niebieska ramka) oraz ustalamy dostępność naszego projektu (domyślnie jest on prywatny – tylko osoby zaproszone będą miały do niego dostęp). Po kliknięciu Create project nasz projekt zostanie utworzony i będziemy mogli zacząć z niego korzystać




This project Search

---

Project Activity Pipelines Registry Issues Merge Requests Wiki Snippets

Project "testzakproject" was successfully created.

You won't be able to pull or push project code via SSH until you [add an SSH key](#) to your profile. [Don't show again](#) | [Remind later](#)

  
 testzakproject

**The repository for this project is empty**

If you already have files you can push them using command line instructions below. Otherwise you can start with adding a [README](#), a [LICENSE](#), or a [gitignore](#) to this project. You will need to be owner or have the master permission level for the initial push, as the master branch is automatically protected.

**Command line instructions**

**Git global setup**

```
git config --global user.name " "
git config --global user.email " "
```

**Create a new repository**

```
git clone https://gitlab.com/
cd testzakproject
touch README.md
git add README.md
git commit -m "add README"
git push -u origin master
```

**Existing folder or Git repository**

```
cd existing_folder
git init
git remote add origin https://gitlab.com/
git add .
git commit
git push -u origin master
```

Gitlab wyświetli nam podstawowe informacje na temat poleceń, jakich możemy używać w naszym systemie celem sklonowania, edycji, zapisywania oraz eksportowania projektu na serwer. W kolejnych dwóch punktach opisane zostanie korzystanie z narzędzia git w systemie Linux oraz Windows.

## 2. Korzystanie z repozytorium w systemach Unix/Unix-like.

Niemalże wszystkie dystrybucje systemy Linux są odgórnie wyposażone w narzędzie git. Jeżeli nie to zapewne można je znaleźć w repozytorium pakietów używanej dystrybucji. Przykładowo dodanie git w systemach rodziny Debian:

```
sudo apt-get install git
```

I już możemy korzystać z dobrodziejstw opisywanego rozwiązania!

INFORMACJA: Pomimo możliwości wykonania poszczególnych kroków w środowisku graficznym poniżej zostanie przedstawiona wersja w linii poleceń. Linia poleceń jest bowiem znacznie szybsza niż klikanie myszą po folderach oraz wykorzystanie komend może odbyć się poprzez kopiuj i wklej w każdej systemowej konsoli!

Klonowany projekt sam utworzy dla siebie nowy katalog w naszym systemie plików. Zaleca się jednak trzymać swoje projekty w jednym katalogu (bądź kilku, w zależności od preferencji). Utworzenie katalogu następuje poprzez polecenie:

```
mkdir <nazwa_katalogu>
```

Przykładowo utworzenie katalogu projekty w katalogu domowym naszego użytkownika:

```
mkdir projekty
```

Następnie przechodzimy do tego katalogu:

```
cd projekty
```

Teraz, w celu wykorzystania naszego utworzonego już projektu wydajemy polecenie:

```
git clone https://gitlab.com/tlatocateculi/testzakproject.git
```

Oczywiście wpisujemy SWOJE DANE (widoczne po utworzeniu projektu) – powyższy link jest składową nazwy naszego konta oraz nazwą utworzonego projektu.

Tak będzie wyglądała „rozmowa” z serwerem projektu:

```
tez@desktop-pc ~/zakprojekt $ git clone https://gitlab.com/tlatocateculi/testzakproject.git
Cloning into 'testzakproject'...
p11-kit: couldn't load module: /usr/lib/x86_64-linux-gnu/pkcs11/gnome-keyring-pkcs11.so: /usr/lib/x86_64-linux-gnu/pkcs11/gnome-keyring-pkcs11.so: cannot open shared object file: No such file or directory
Username for 'https://gitlab.com': tlatocateculi
Password for 'https://tlatocateculi@gitlab.com':
warning: You appear to have cloned an empty repository.
Checking connectivity... done.
```

Teraz przechodzimy do katalogu naszego zaimportowanego projektu:

```
cd testzakproject
```

Obecnie w katalogu nic nie ma. Utwórzmy zatem jakikolwiek plik – np. index.html. Można wykorzystać dowolny edytor tekstowy – w tym przypadku wykorzystamy kate:

```
kate index.html
```

Teraz wpisujemy przykładowy kod, np.:

```
<html>
  <head>
  </head>
  <body>
    <p>Strona główna w projekcie git!</p>
  </body>
</html>
```

Możemy zapisać plik.

Nowe pliki nie dodają się same do naszego repozytorium. Musimy je dodawać „ręcznie”. Jeżeli chcemy dodać pojedyncze pliki możemy robić za pomocą tej komendy:

```
git add <nazwy_plików_oddzielone_spacjami>
```

Można tak także dodawać katalogi. Innym sposobem jest dodawanie wszystkiego co znajduje się w katalogu repozytorium:

```
git add .
```

Powyższa komenda doda wszystko co znajduje się w katalogu do repozytorium. Od tego momentu dodane elementy będą śledzone pod kątem dokonanych w nich zmian. Teraz trzeba zatwierdzić zmiany w dodanych elementach. W tym celu posłużymy się poniższą komendą:

```
git commit -m 'Krótki komentarz dokonanych zmian, uwag na temat obecnego stanu projektu itp.'
```

Tym samym aktualny stan wszystkich plików repozytorium został zapisany. Od tego momentu wszelkie nowe zmiany w plikach nie będą miały wpływu na sporządzoną migawkę aktualnej zawartości elementów. W każdej chwili można dokonać przeglądu plików, porównać z aktualną zawartością bądź cofnąć zmiany do określonego stanu pliku.

INFORMACJA: Pliki można także usuwać. W tym wypadku, prócz fizycznego usunięcia z dysku trzeba je także usunąć z repozytorium. W tym celu należy użyć polecenia:

```
git rm <nazwa_pliku/ów>
```

Ostatnią ważną sprawą jest możliwość wyeksportowania zmian w repozytorium na nasz serwer. Dokonuje się tego komendą:

```
git push origin master
```

```

tez@desktop-pc ~/zakprojekt/testzakproject $ git push origin master
pll-kit: couldn't load module: /usr/lib/x86_64-linux-gnu/pkcs11/gnome-keyring-pk
cs11.so: /usr/lib/x86_64-linux-gnu/pkcs11/gnome-keyring-pkcs11.so: cannot open s
hared object file: No such file or directory
Username for 'https://gitlab.com': ██████████
Password for 'https://██████████@gitlab.com':
Counting objects: 3, done.
Delta compression using up to 4 threads.
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 283 bytes | 0 bytes/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://gitlab.com/t██████████git
 * [new branch]      master -> master

```

W odpowiedzi otrzymamy komunikaty jak na powyższy zrzucie (tak jak w przypadku pobierania repozytorium zostaniemy poproszeni o podanie loginu oraz hasła). Gotowe. Zmiany można podziwiać na naszym serwerze.

### 3. Korzystanie z repozytorium na systemie Windows.

Git nie jest tak łatwy w obsłudze w systemie Windows. Dopiero Windows 10 w najnowszej wersji wprowadza możliwość wykorzystywania powłoki systemu Linux.

Fani gita stworzyli natomiast graficzne narzędzia pozwalające na użytkowanie repozytorium także pod Windows. Jednym z takich narzędzi jest Tortoise – chyba najpopularniejsze narzędzie do kontroli systemów zarządzania wersjami oprogramowania.

Oprogramowanie można pobrać z tej strony:  
<https://tortoisegit.org/download/>

The screenshot shows the TortoiseGit website's download page. The page title is "Download - TortoiseGit" and the URL is "https://tortoisegit.org/download/". The page features the TortoiseGit logo and navigation links for "About", "Download", "Support", and "Contribute". The main content area is titled "Download" and states "The current stable version is: 2.3.0". It provides links to "release notes" and an "FAQ: System prerequisites and installation". There are two sections for downloading the installer: "for 32-bit OS" and "for 64-bit OS", each with a "Download" link and file size information. Below this, there is a "Donate" section and a "Language Packs" section. The "Language Packs" section includes a table with columns for Language, Code, Completeness, 32 Bit, and 64 Bit.

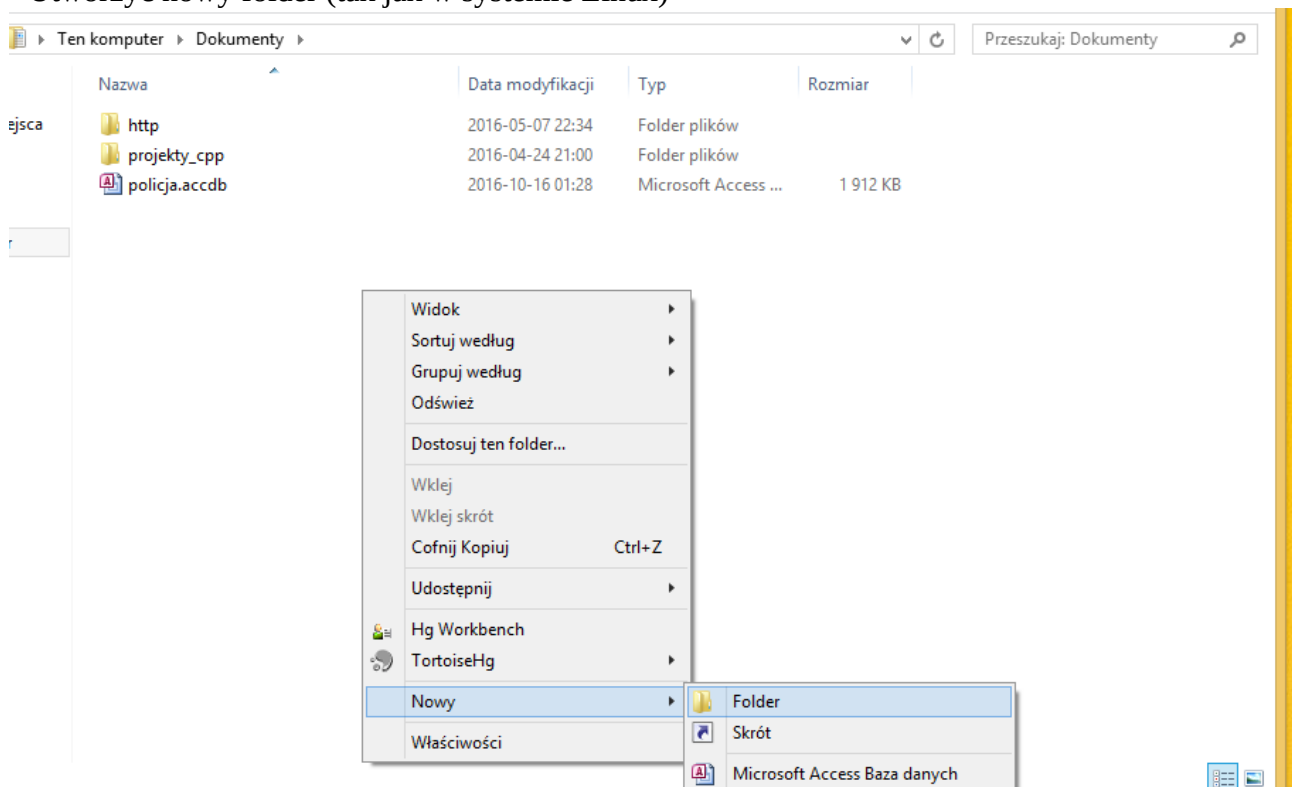
Language	Code	Completeness	32 Bit	64 Bit
Bulgarian	bg	56%	<a href="#">Setup</a>	<a href="#">Setup</a>
Catalan	ca	87%	<a href="#">Setup</a>	<a href="#">Setup</a>
Chinese, simplified	zh_CN	100%	<a href="#">Setup</a>	<a href="#">Setup</a>

Instalacja polega na klikaniu „Dalej” (można edytować listę składników do instalacji, jednak nie jest to wymagane). Po instalacji narzędzie wymaga posiadania narzędzi Git dla Windows. Można je pobrać poprzez stronę <https://git-for-windows.github.io/>

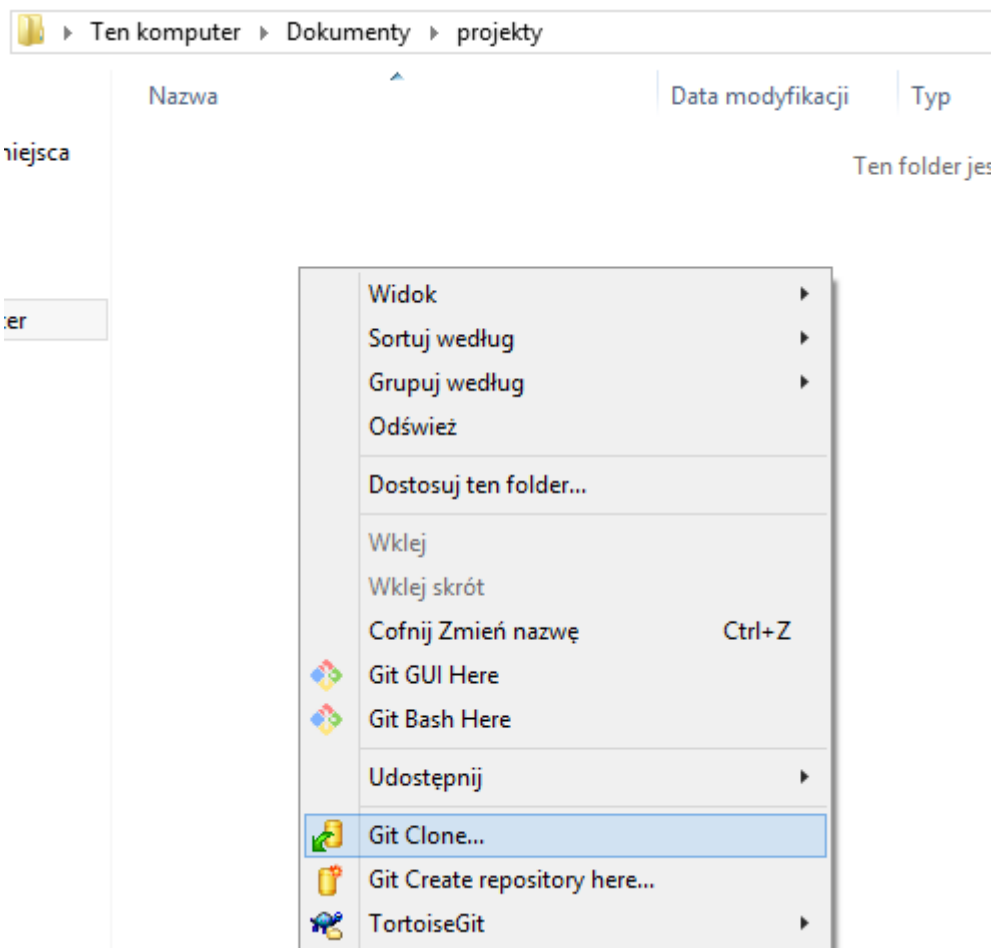
Podobnie jak to miało miejsce z Tortoise instalację można wykonać dokonując domyślnych wyborów. Teraz wszystko powinno działać.

Wystarczy:

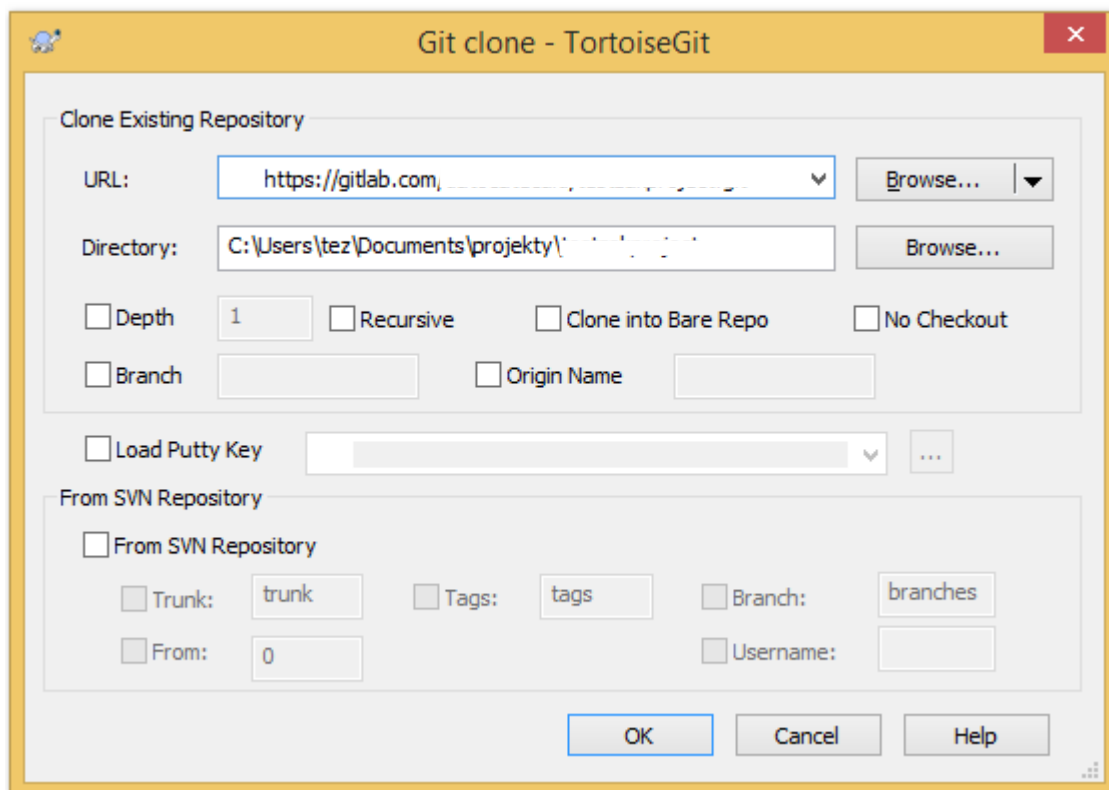
- Utworzyć nowy folder (tak jak w systemie Linux)



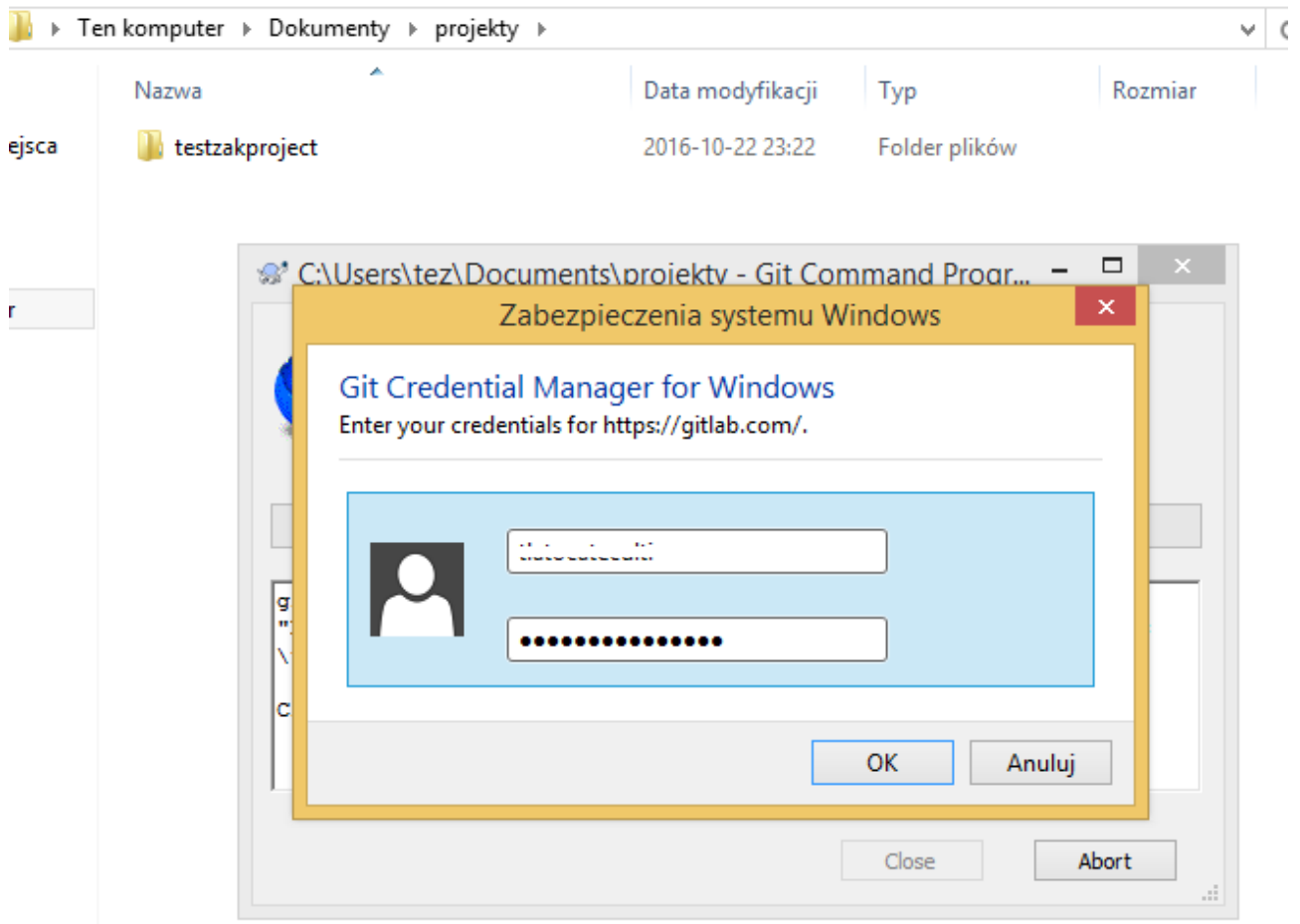
- W folderze kliknąć prawym przyciskiem myszy i wybrać opcję Clone



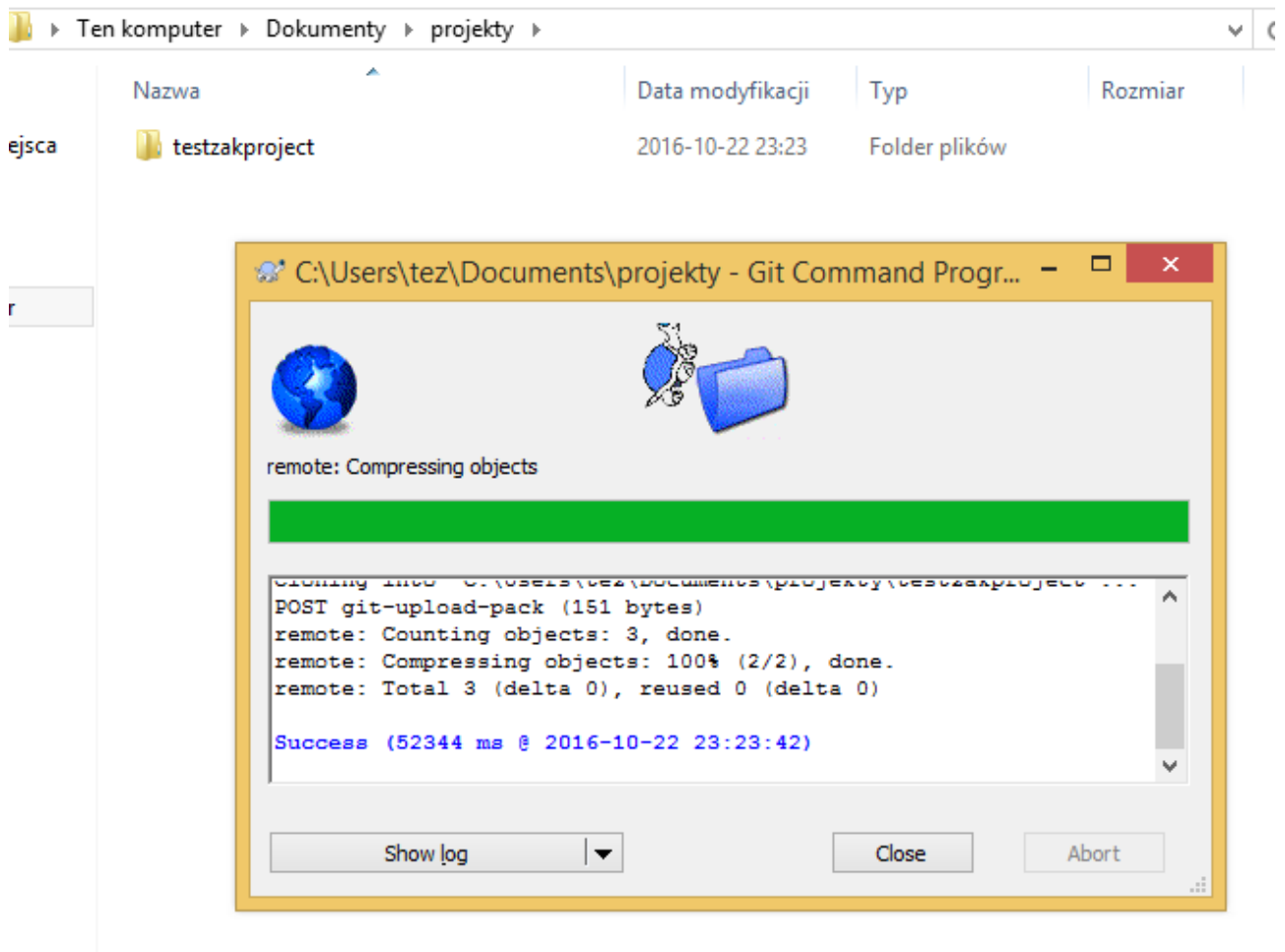
- W nowym oknie kopiujemy adres do naszego repozytorium:



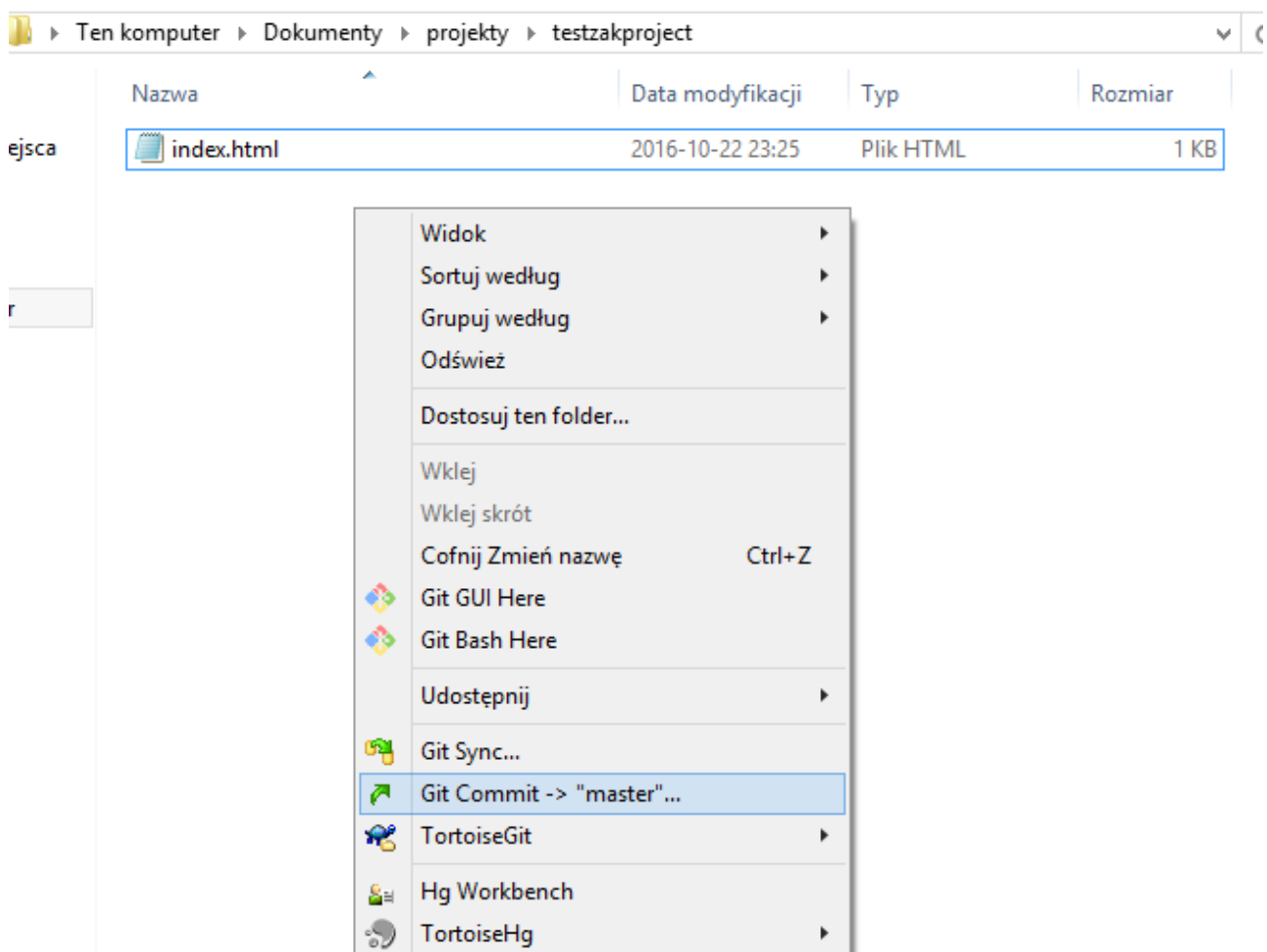
- Autoryzujemy się



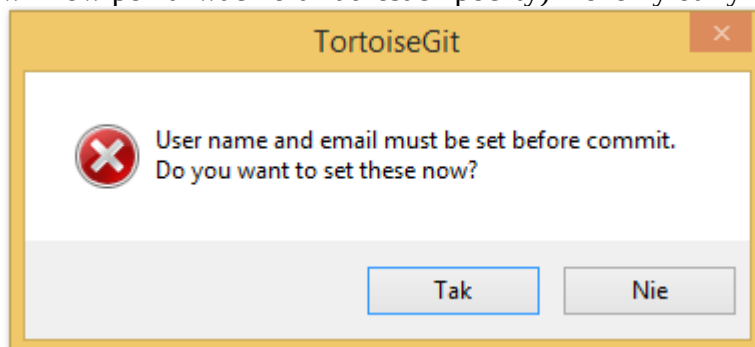
- Gotowe; nasz projekt został właśnie zaimportowany (otrzymujemy stosowną informację):



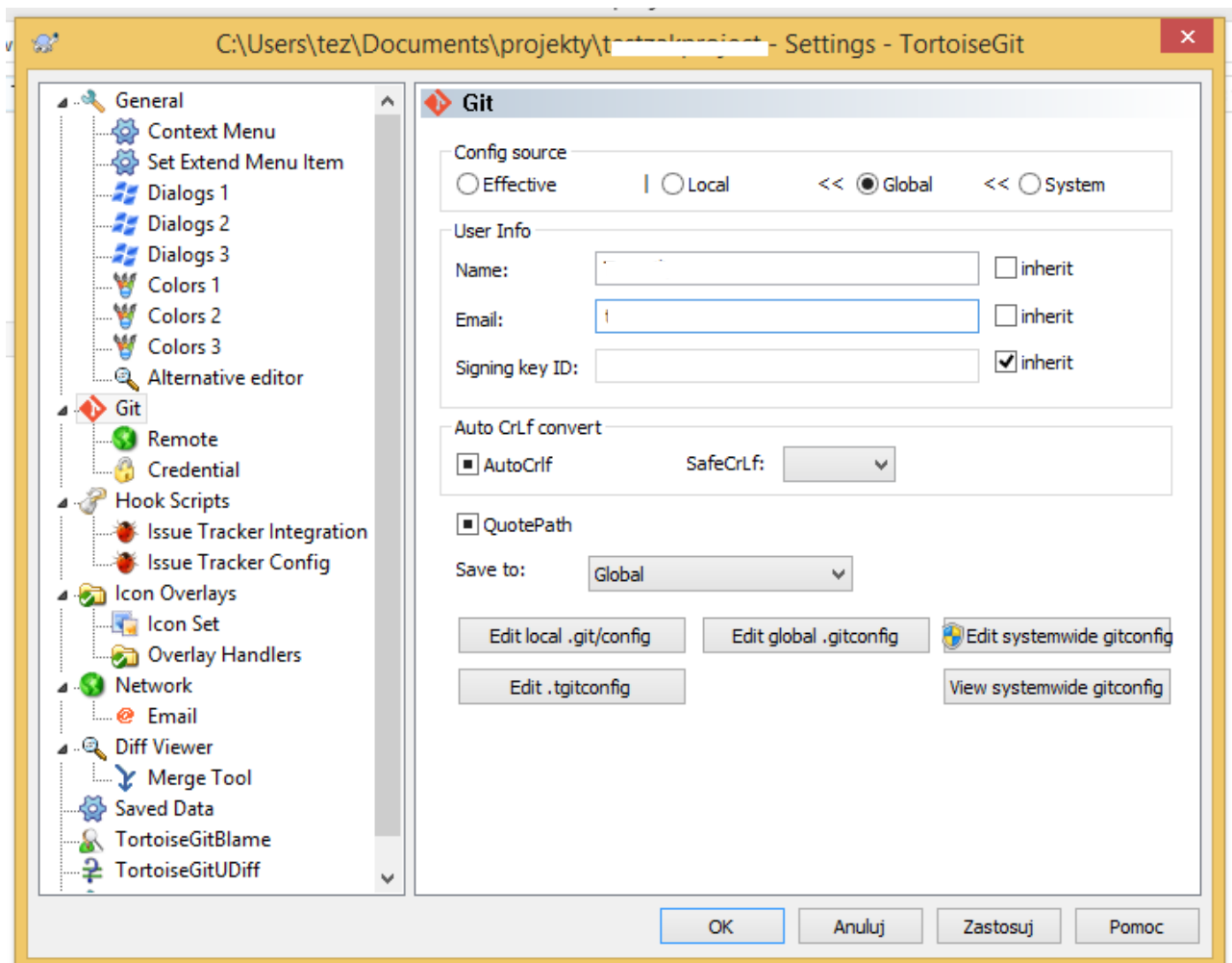
Teraz możemy edytować pliki znajdujące się w naszym projekcie tak jak chcemy i lubimy (przed dowolne narzędzia). Jeżeli będziemy chcieli dokonać zapisu repozytorium robimy to będąc w katalogu projektu; klikamy w dowolnym miejscu prawym przyciskiem myszy i wybieramy opcję Git Commit → „Master”...



Jeżeli nie dokonaliśmy jeszcze ustawień kim jesteśmy dla repozytorium (proszę pamiętać, że git rozróżnia użytkowników po nazwach oraz adresach poczty) możemy otrzymać taki oto komunikat:

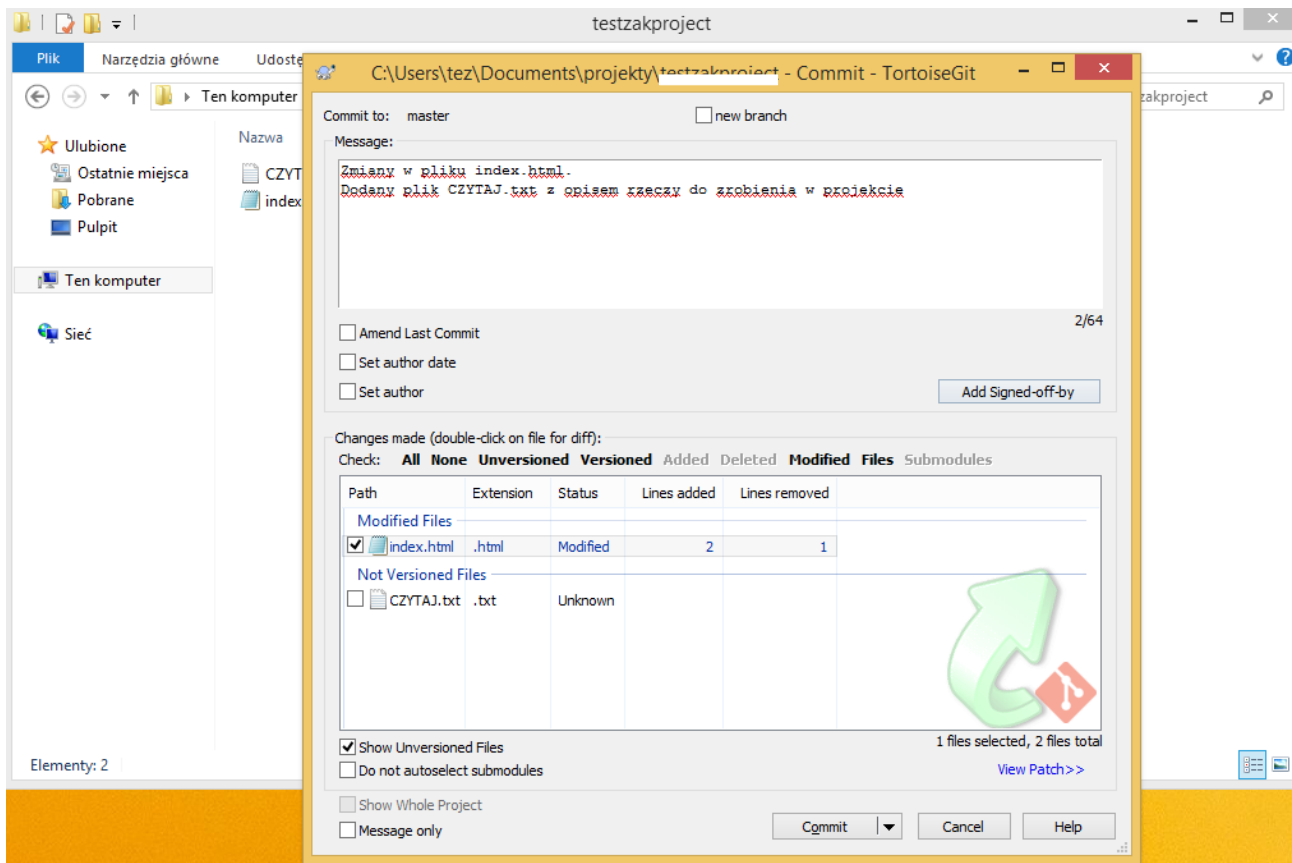


Odpowiadamy twierdząco, po czym dokonujemy zmian w odpowiednich polach (zaznaczone czerwoną ramką):

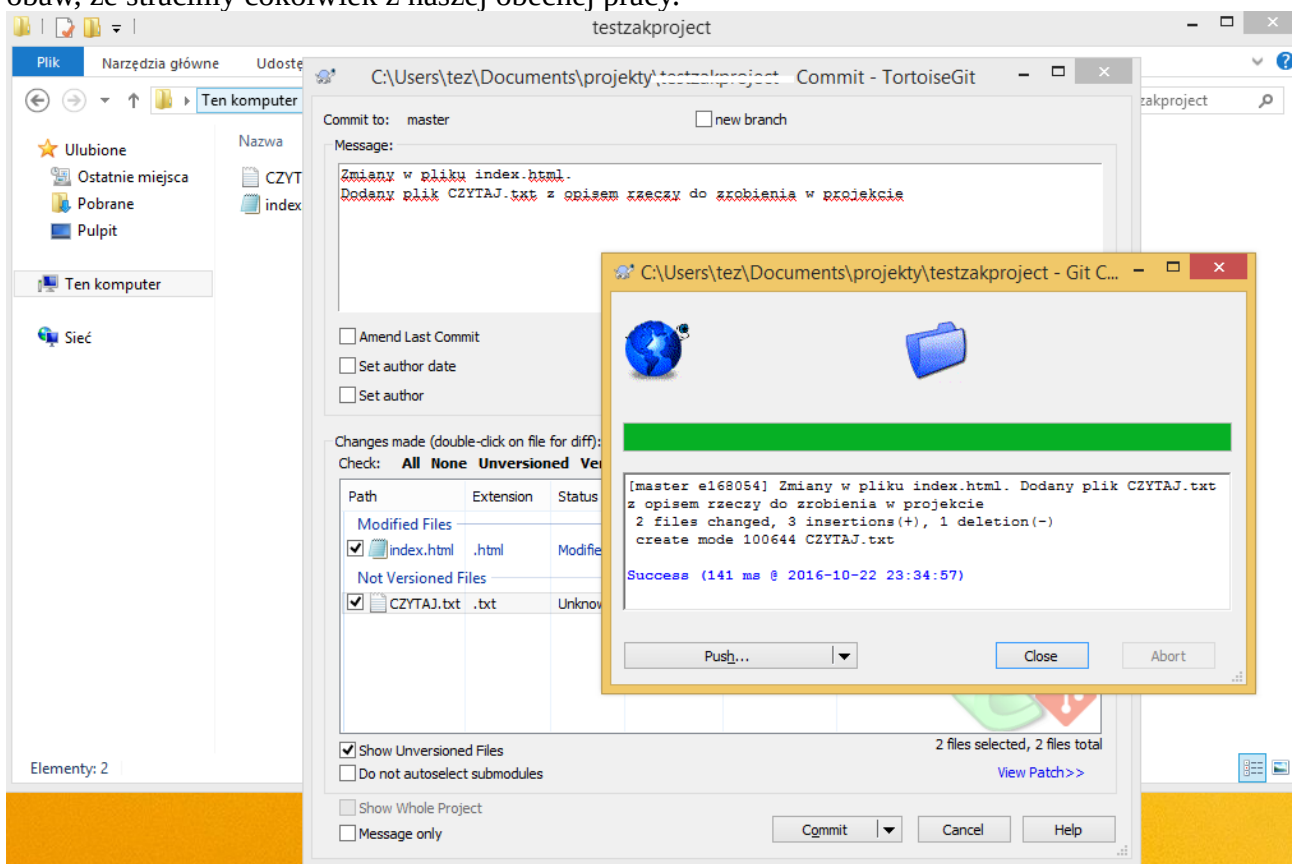


Proszę zauważyć, że narzędzie posiada znacznie więcej opcji. Można je dowolnie edytować, w zależności od swoich potrzeb (aktualny materiał nie będzie się nimi zajmować).

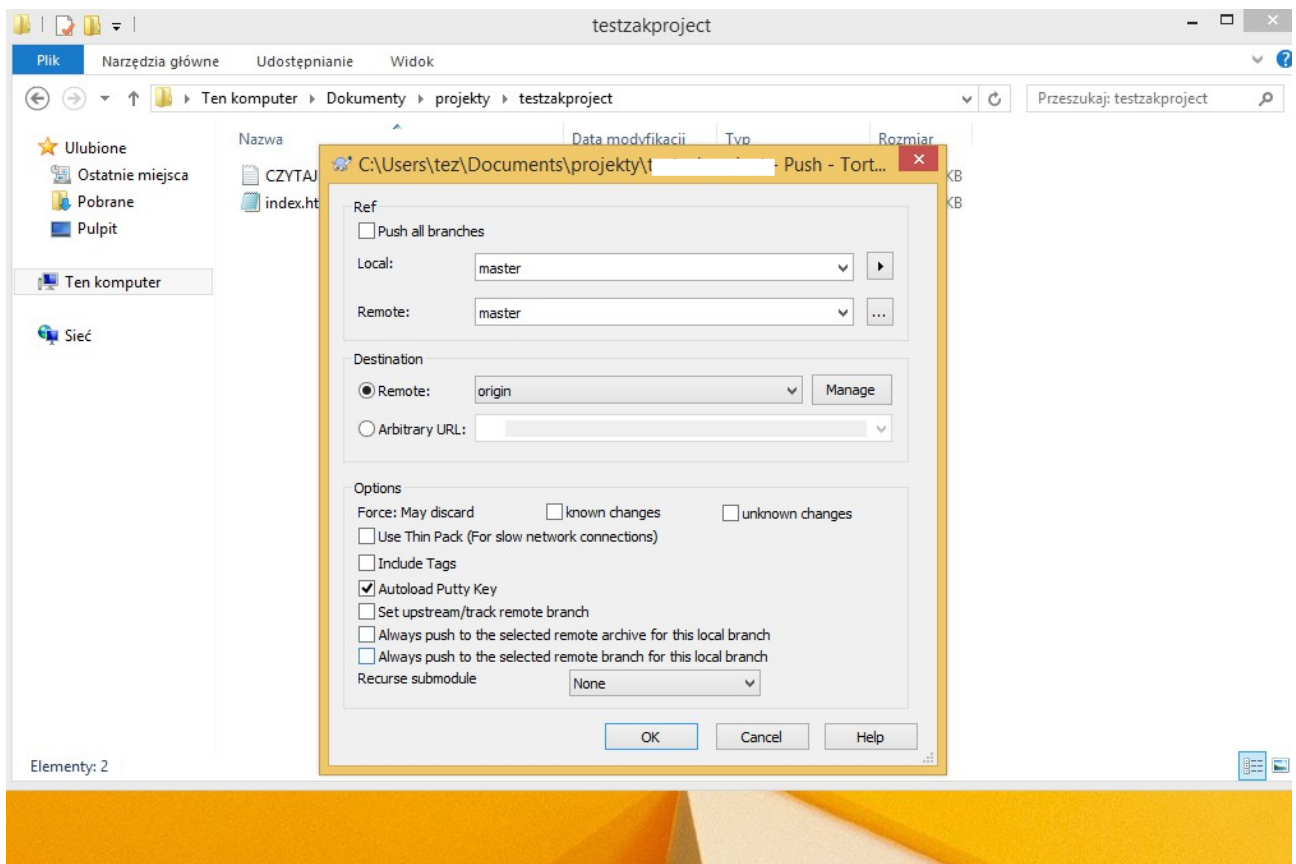
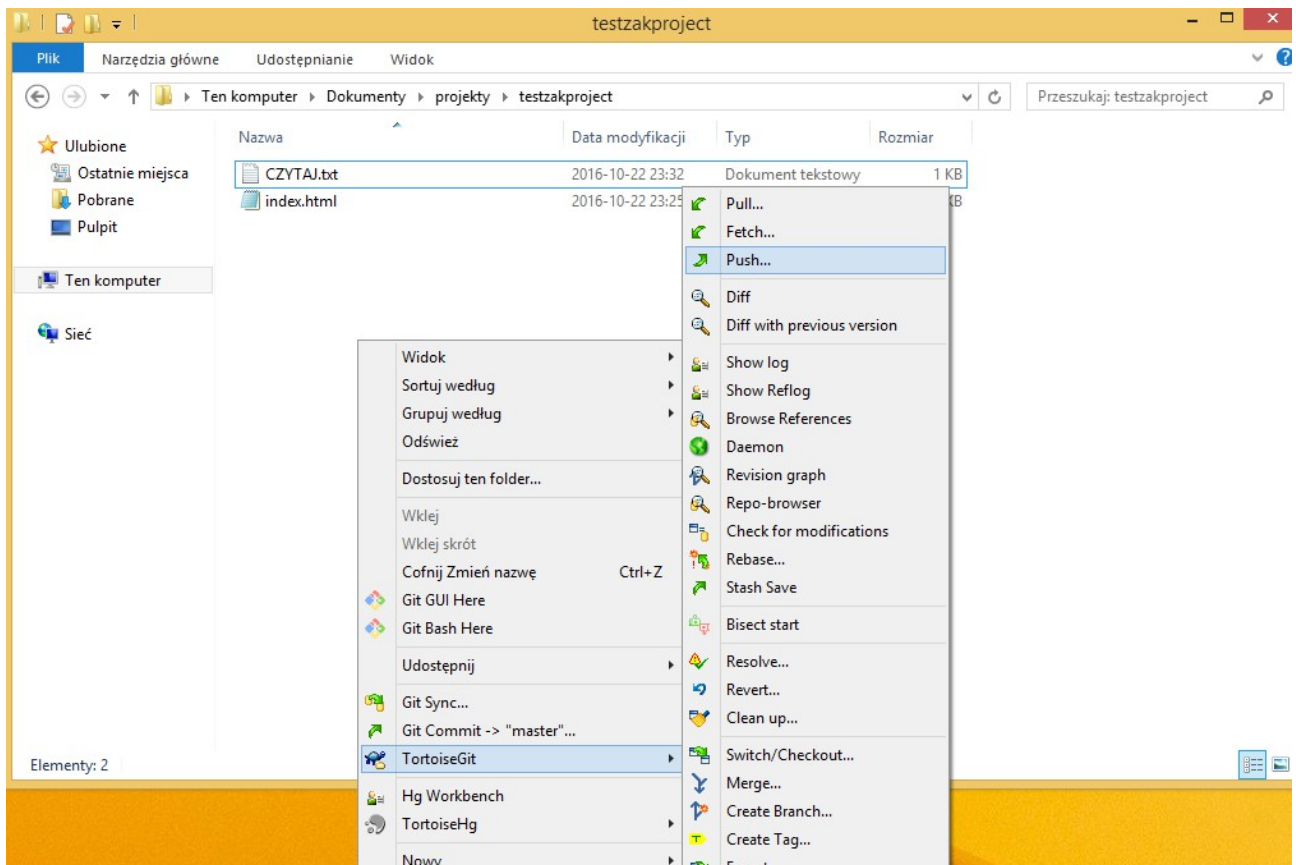
Gdy klikniemy OK przejdziemy do okna zatwierdzenia zmian w projekcie. Proszę zwrócić uwagę na plik CZYTAJ.txt – jest to plik dodany aktualnie, jeszcze nie dodany do repozytorium. Możemy go do niego dołączyć poprzez zaznaczenie go.



Klikając Commit akceptujemy zmiany. Możemy bezpiecznie dalej modyfikować pliki projektu bez obaw, że stracimy cokolwiek z naszej obecnej pracy.



Jeżeli chcemy wysłać naszą pracę na serwer to korzystamy z opcji Push w menu kontekstowym, jak na rzucie poniżej:



Projekt został właśnie wysłany na serwer.

## WAŻNA INFORMACJA!!!

Przedstawione opcje narzędzia z poziomu konsoli, jak i interfejsu graficznego nie przedstawiają ich pełnych możliwości! Proszę zauważyć, że obie wersje (konsolowa i graficzna) posiadają, prócz prostego zapisywania/eksportowania projektu na serwer, także możliwość dociągnięcia zmian do projektu (opcja Pull), przebudowy (rebase), sprawdzenie zmian (diff) i wiele, wiele innych przydatnych poleceń. Niniejszy materiał nie będzie się jednak nimi zajmował – szczegóły dodatkowych poleceń można odnaleźć we wspomnianych wcześniej podręcznikach.

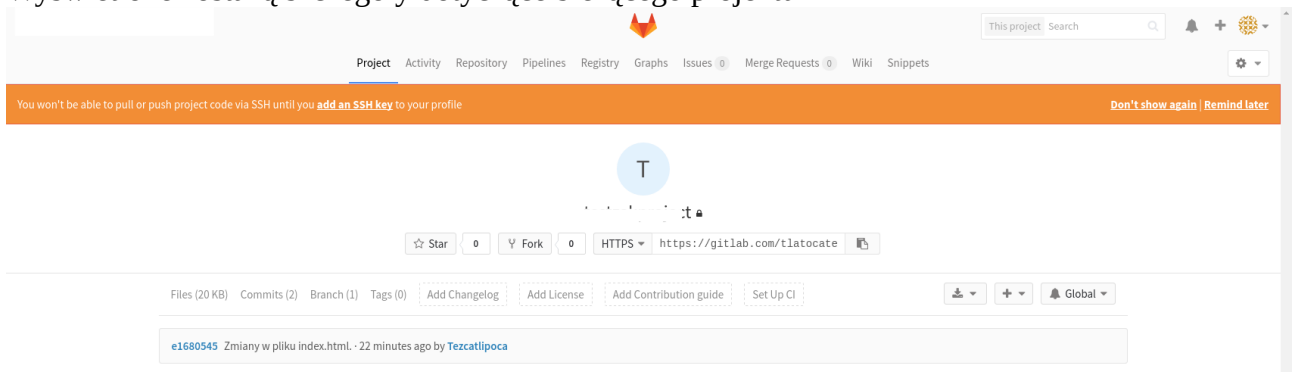
### 4. Korzystanie z projektu poprzez stronę WWW.

Ponieważ posiadamy już projekt, który posiada przynajmniej jedną zapisaną migawkę, możemy sprawdzić jak prezentuje się on po stronie WWW. Aby zobaczyć listę naszych projektów (obecnie jest jeden) należy kliknąć zakładkę Projekts.



Z listy klikamy na nasz projekt (czerwona ramka)

Wyświetlone zostaną szczegóły dotyczące bieżącego projektu

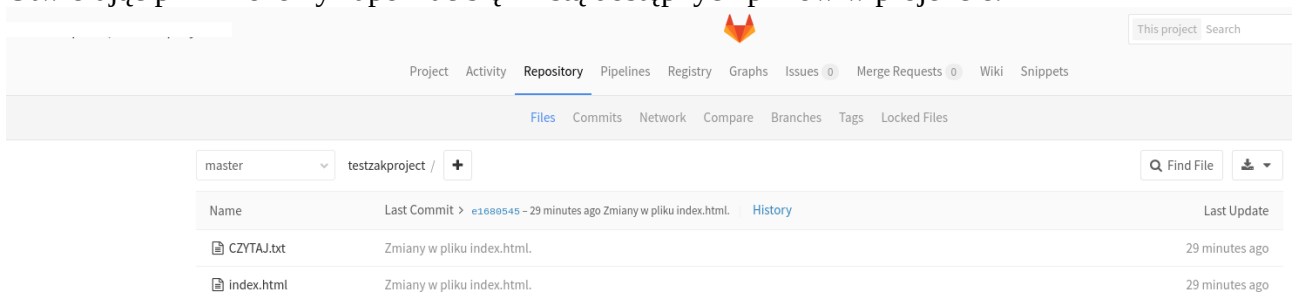


Z tego ekranu możemy oznaczyć projekt gwiazdką (ulubiony/wyróżniony), rozwidlić go na nowy projekt (Fork; opcji tej używa się by zacząć NOWY projekt na bazie aktualnie przeglądanego), bądź skopiować jego adres celem klonowania go/udostępnienia innym osobom. Ponadto możemy przeglądać pliki wchodzące w skład projektu (Files), aktualne migawki (Commits), gałęzie (Branch), etykiety (Tags).

Pozostałe opcje to możliwość dodania informacji o zmianach w repozytorium, dodanie informacji o licencji projektu (szczególnie ważne przy publicznych projektach) czy też dodanie informacji o autorach/pomocnikach w projekcie. Ponadto dodatkowe opcje pozwalają skopiować aktualny stan repozytorium do spakowanego pliku (zip/tar/trg.gz) i/lub dołączyć nową uwagę/spostrzeżenie/plik/gałąź.

Poniżej opcji gitlab podaje nam informacje o ostatniej modyfikacji w projekcie.

Otwierając pliki możemy zapoznać się z listą dostępnych plików w projekcie.

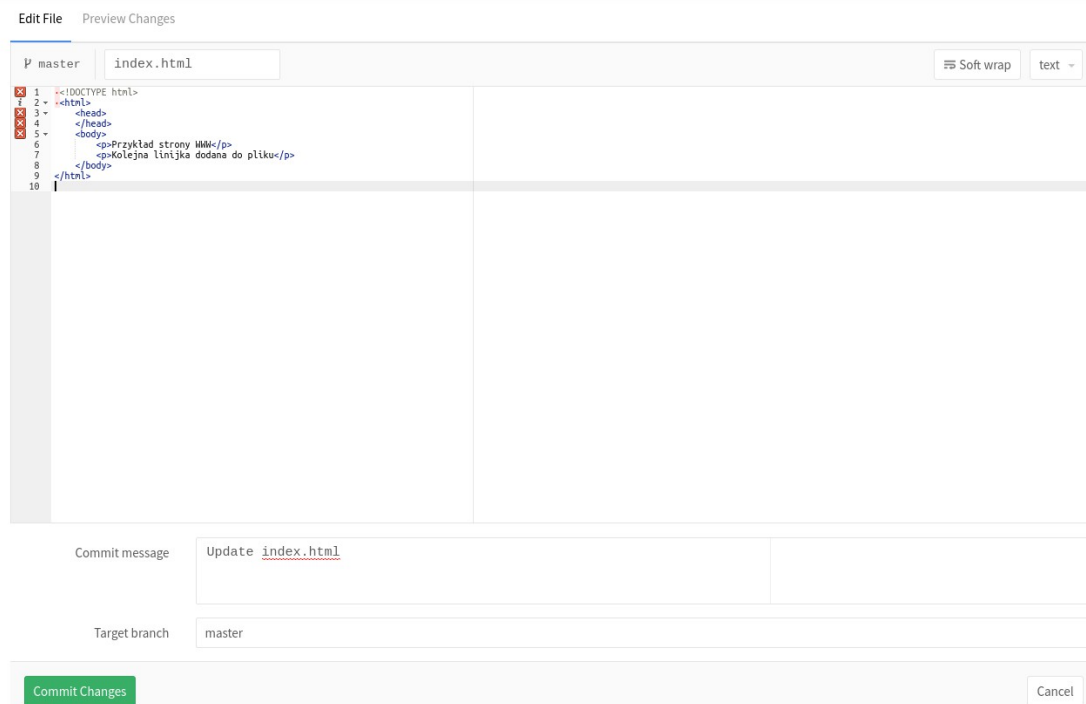


Po wybraniu któregoś z plików widzimy jego zawartość, a także mamy możliwość edytowania go, podmiany, zablokowania czy skasowania.



```
1 <html>
2   <head>
3 </head>
4   <body>
5     <p>Przykład strony WWW</p>
6     <p>Kolejna linijka dodana do pliku</p>
7   </body>
8 </html>
```

## Edycja pliku index.html:



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4 </head>
5 <body>
6 <p>Przykład strony WWW</p>
7 <p>Kolejna linijka dodana do pliku</p>
8 </body>
9 </html>
10
```

Commit message: Update [index.html](#)

Target branch: master

Commit Changes Cancel

Na dole mamy możliwość edycji komentarza do zmiany. Po kliknięciu zielonego przycisku zmiana w pliku zostanie dokonana, natomiast repozytorium otrzyma nowe zatwierdzenie (migawkę).

Zmiany poprzez stronę WWW nie są może najlepszą opcją, czasami jednak mogą stać się nieocenioną pomocą w pracy z repozytorium.

## 5. Uwagi końcowe

Przedstawione w niniejszym materiale informacje na temat repozytoriów są jedynie informacjami wstępnymi. Należy mieć na uwadze, że narzędzie to ma znacznie szersze zastosowanie. Sam materiał należy traktować jedynie jako wstęp do użytkowania narzędzia.

## MATERIAŁY DODATKOWE:

<https://blog.profitbricks.com/top-source-code-repository-hosts/>  
<https://git.wiki.kernel.org/index.php/GitHosting>