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. Początkowy 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 jakiekolwiek 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

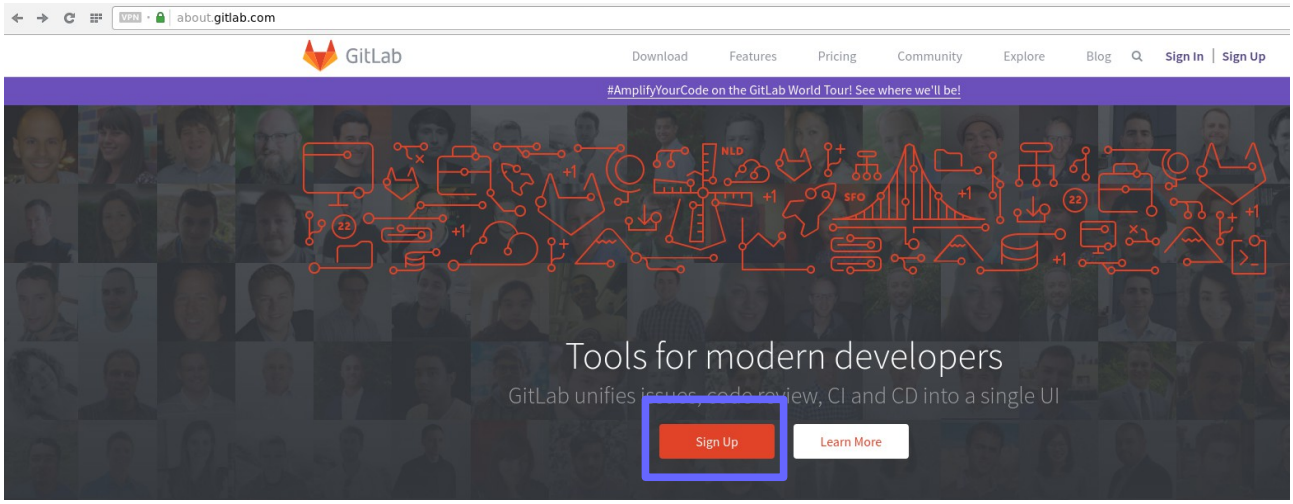
Ostatnia możliwość jest najprostsza dla osób, które w prosty sposób chcą cieszyć się przenośnością swoich repozytoriów.

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:

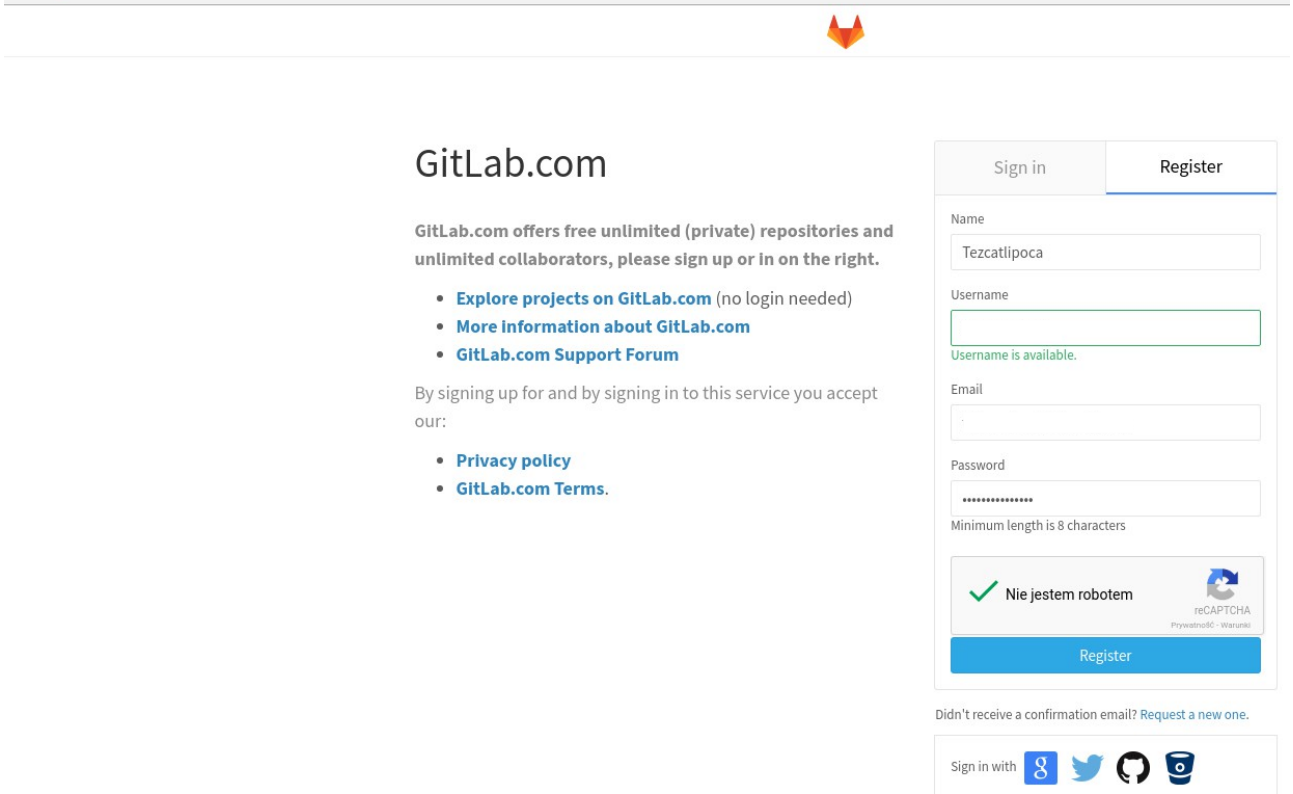
- jeden z niewielu o otwartym źródle (dostępny jest kod źródłowy jego wszystkich narzędzi, jak i samego serwisu)
- zezwala na nielimitowaną ilość projektów (w zasadzie do 10.000 projektów na użytkownika; prócz tego możemy dołączać do dowolnej ilości projektów)
- zezwala na dodanie nielimitowanej ilości 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



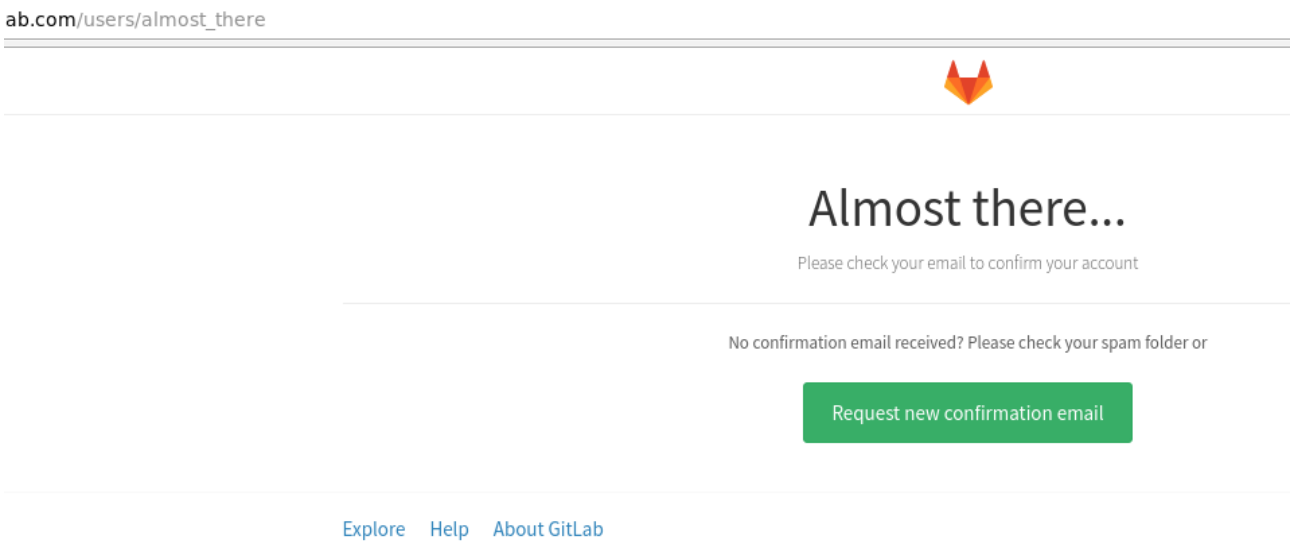
Klikamy Gign up (niebieska ramka)



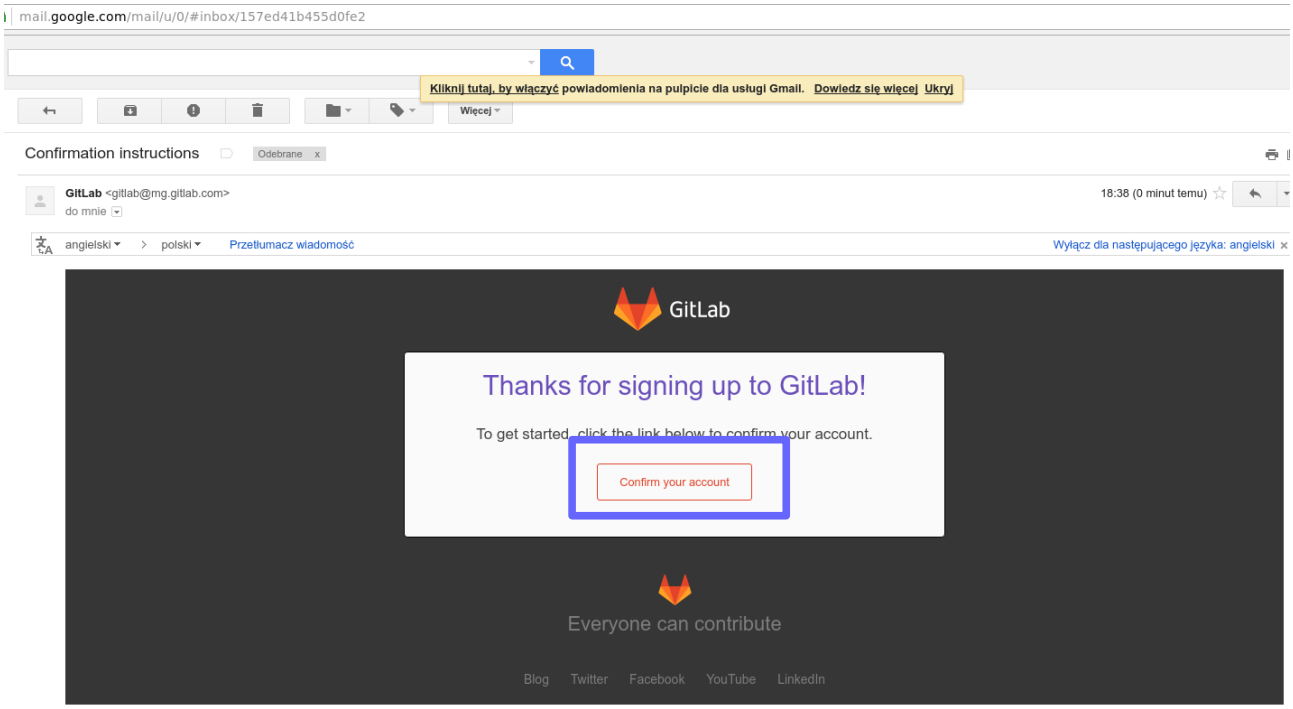
Podajemy nazwę użytkownika, adres poczty oraz hasło; możemy także autoryzować się kontem z Google, Twitter, github bądź bitbucket (dwa ostanie to podobne serwisy do gitlab).

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!

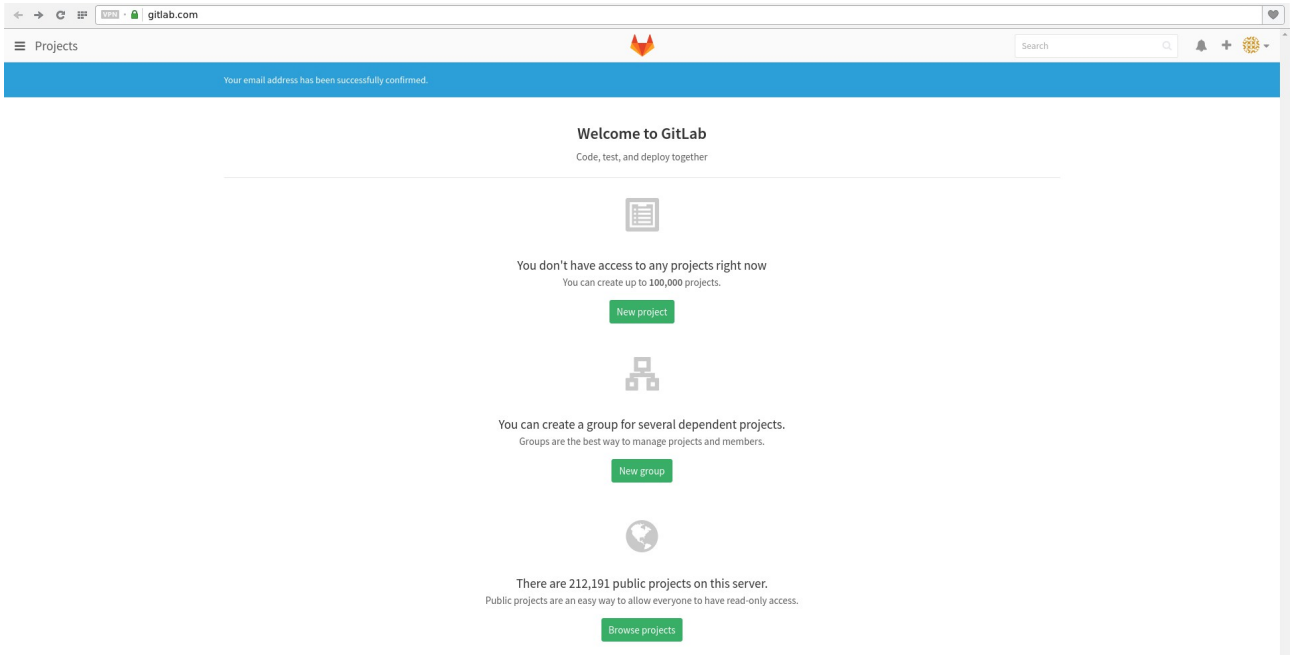
Po wykonaniu tych czynności zostaniemy poinformowani o przesłaniu na nasz adres informacji potwierdzającej założenie konta w portalu gitlab.com. Strona poinformuje nas o oczekiwaniu na potwierdzenie informacji...



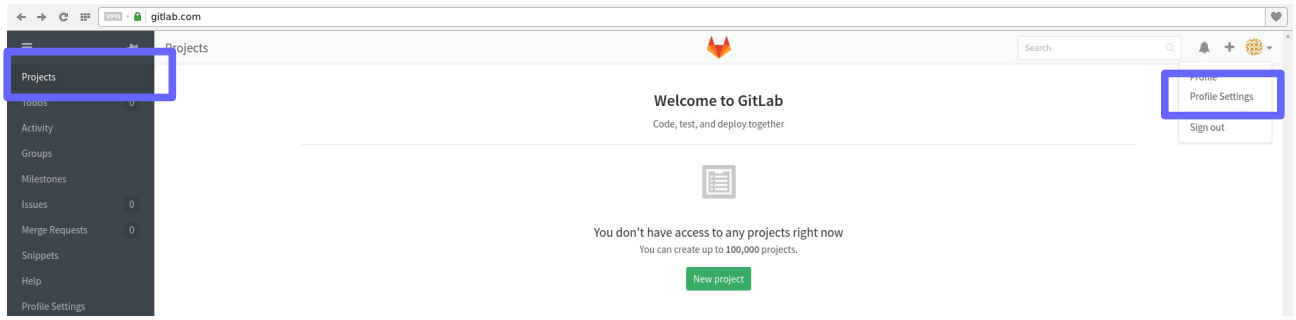
Poczta z prośbą o potwierdzenie wygląda tak jak na zrzucie poniżej. Wystarczy kliknąć w odpowiedni przycisk (niebieska ramka):

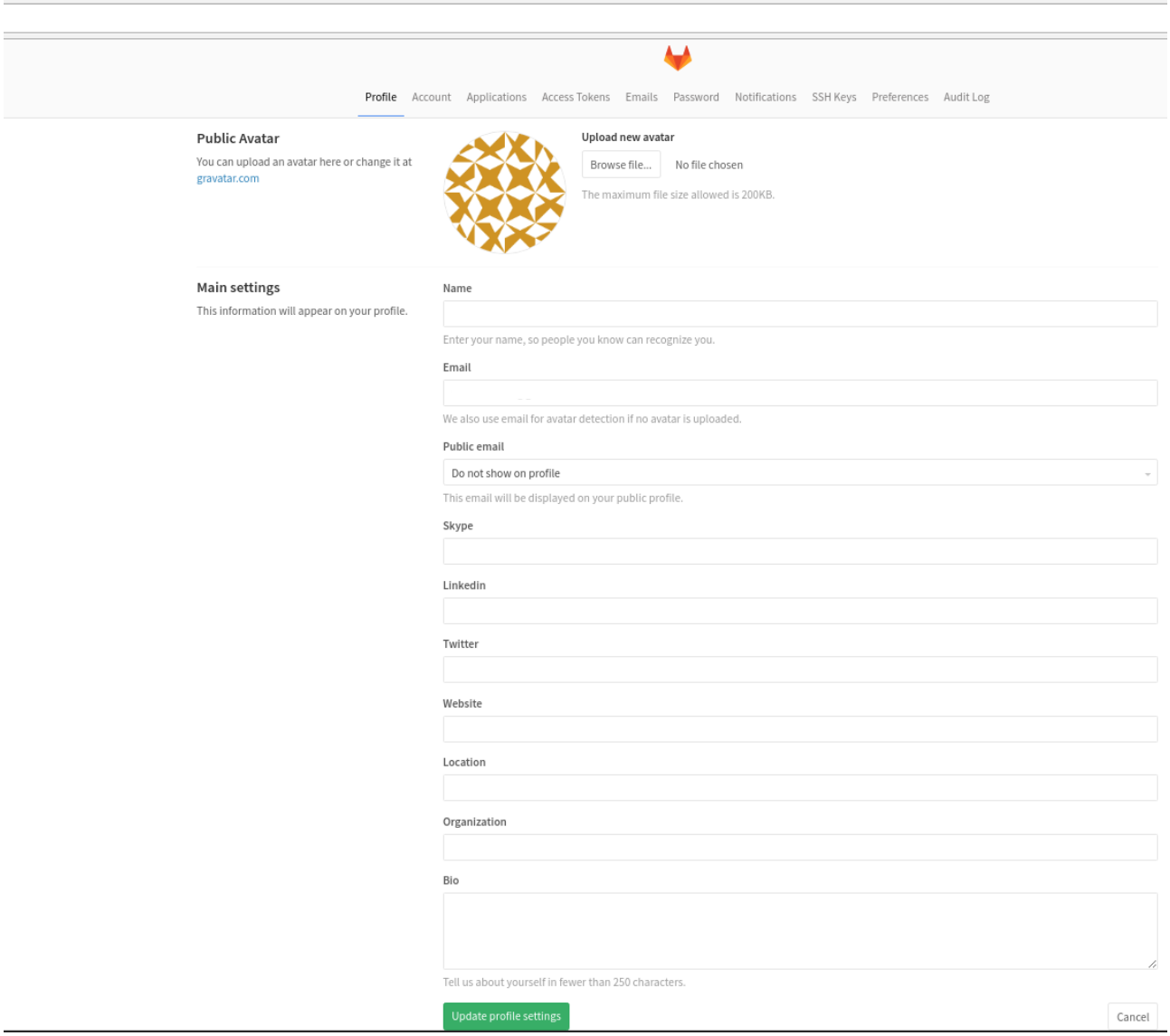


Zaraz po zaakceptowaniu przejdziemy na stronę główną naszego konta w portalu. Wyświetlone zostaną nam podstawowe możliwości – założenie projektu, grup bądź możliwość przejrzenia publicznych projektów.

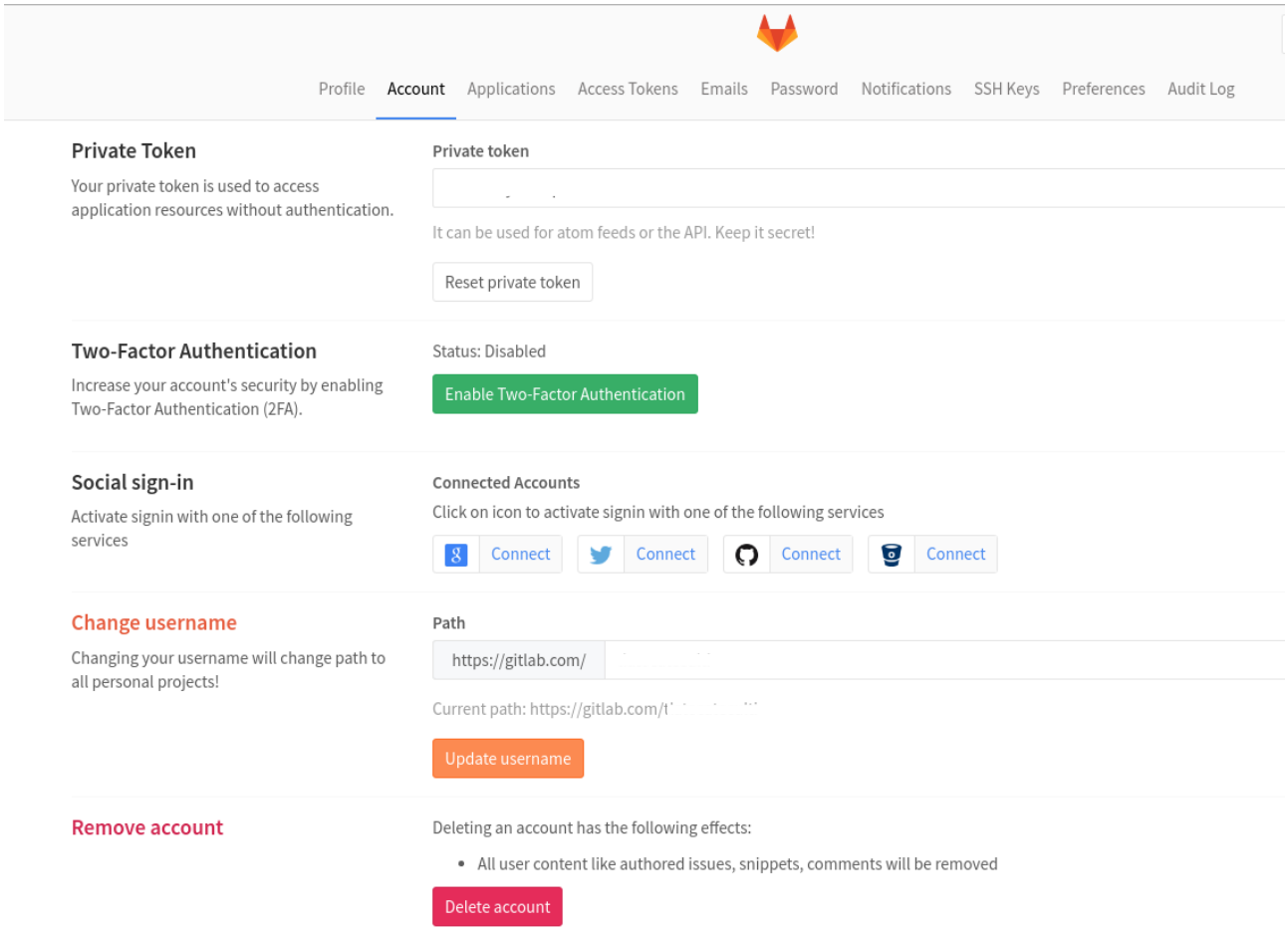


Nas jednak aktualnie powinna zainteresować możliwość edycji profilu (wejście do opcji zaznaczone niebieskim prostokątem):

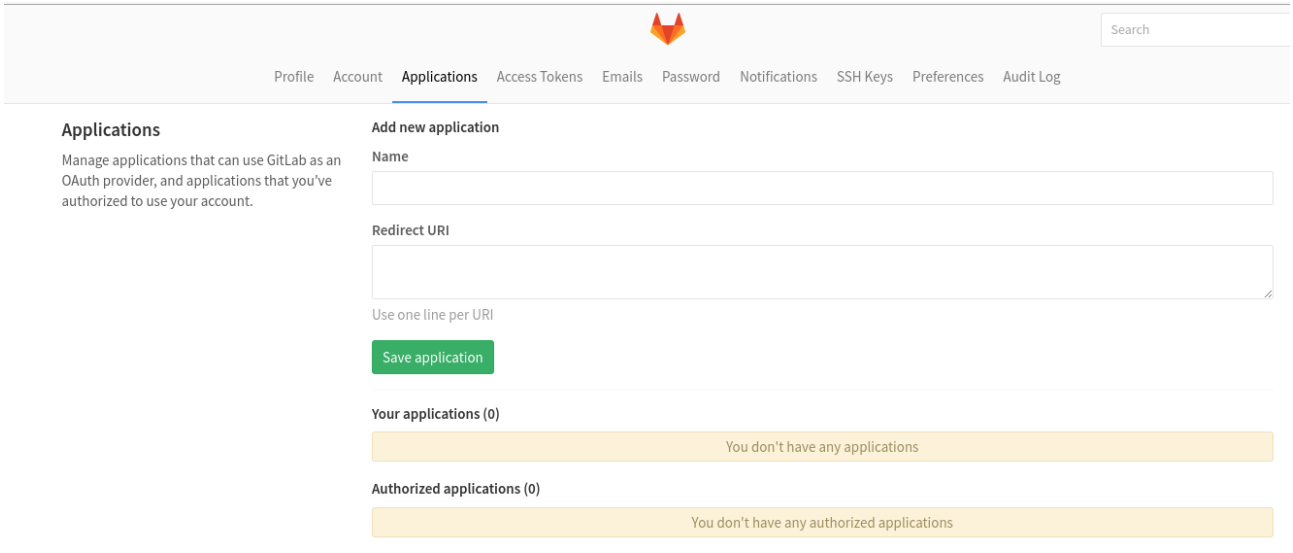




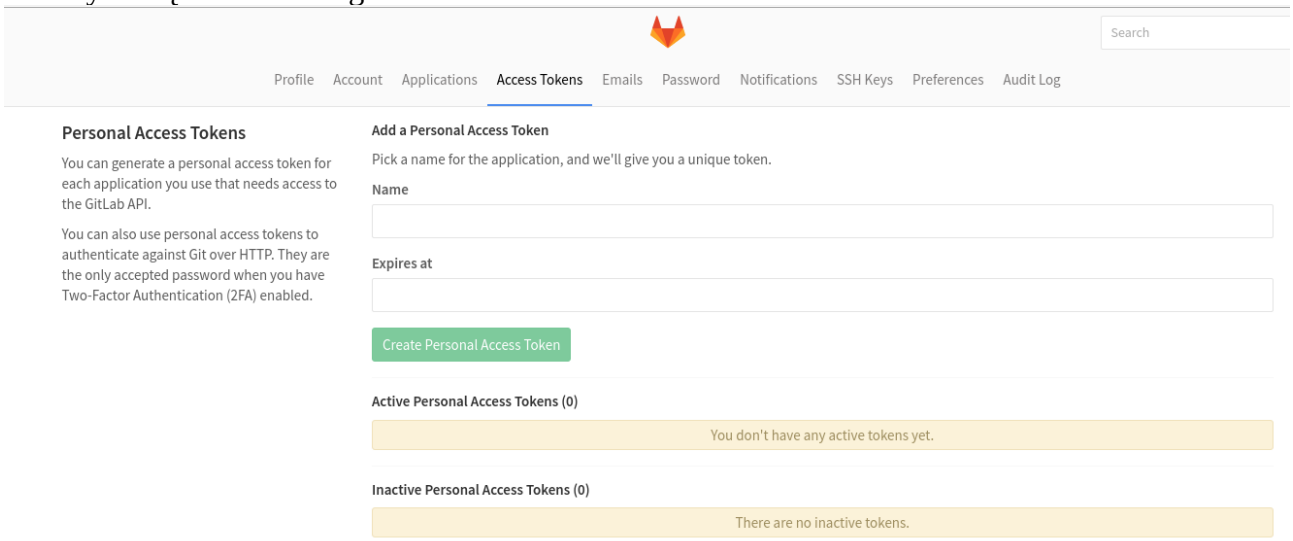
Na pierwszej stronie mamy naszą nazwę użytkownika oraz możliwość ustawienia: obrazka reprezentującego nasz profil, ustawienie adresu poczty (bądź zmianę aktualnego na nowy), ustawienie widoczności adresu (domyślnie nie jest pokazywany), czy też ustawienie pozostały opcji kontaktowych (domyślnie puste).



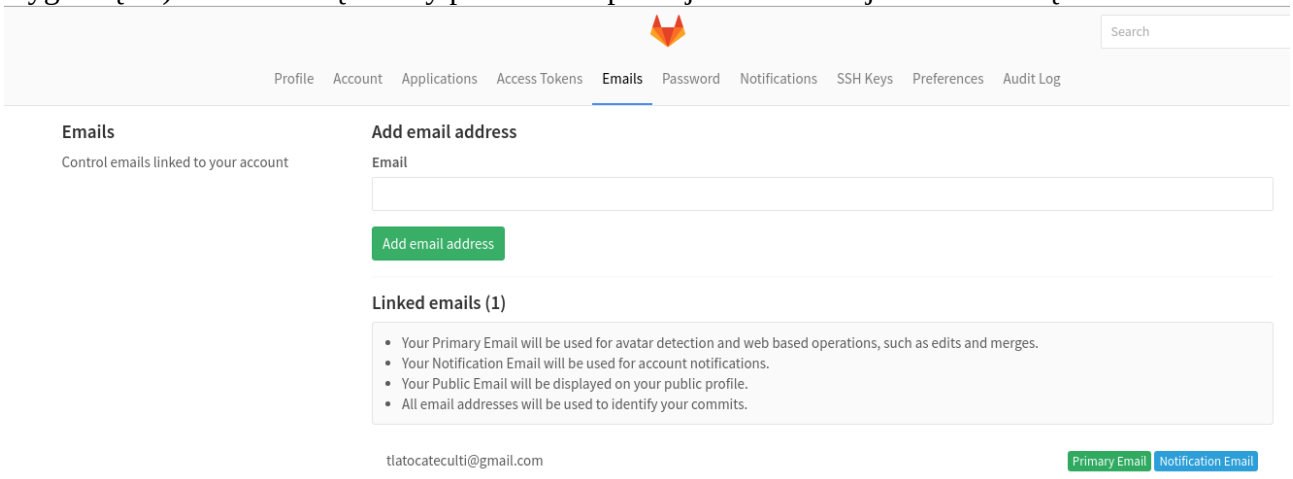
Zakładka Account pozwala na ustawienie tzw. prywatnego żetonu (Private Token). Żeton można wykorzystywać jako autoryzację naszego konta w programach zarządzających wersjami oprogramowania. Dzięki temu nie musimy podawać ciągle naszego hasła. Numer ten powinien pozostawać tajemnicą! Jeżeli mamy podejrzenie, 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.



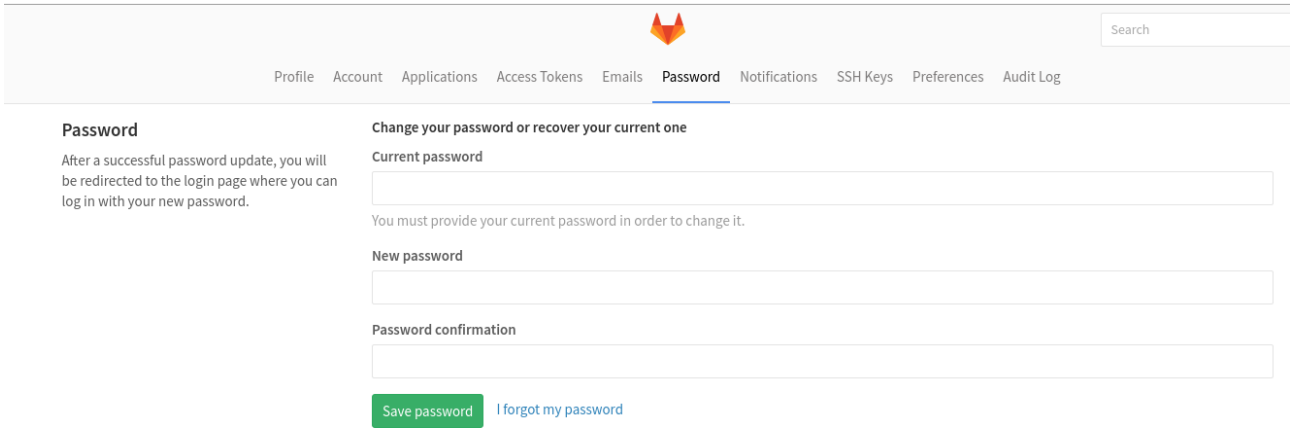
Zakładka Applications można dodać zewnętrzne aplikacje zarządzające kontem na gitlab. Opcja przydatna w chwili, gdy posiadamy konta na innych serwisach tego typu i chcemy poprzez tamte serwisy zarządzać kontem gitlab.



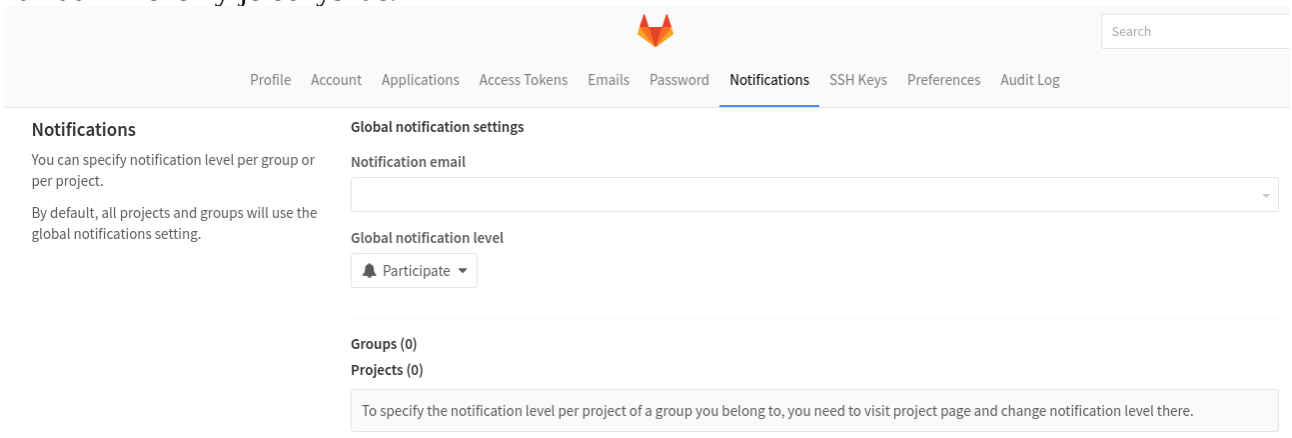
Jeżeli chcemy nadać jakieś zewnętrznej aplikacji możliwość ingerencji w konto gitlab poprzez jego API (rozwijamy np. własną aplikację pozwalającą na zarządzanie kontem) to tutaj możemy wygenerować żetony dostępowe. Generowane żetony mogą mieć swój czas życia (datę wygaśnięcia). Jeżeli nie będziemy pisać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

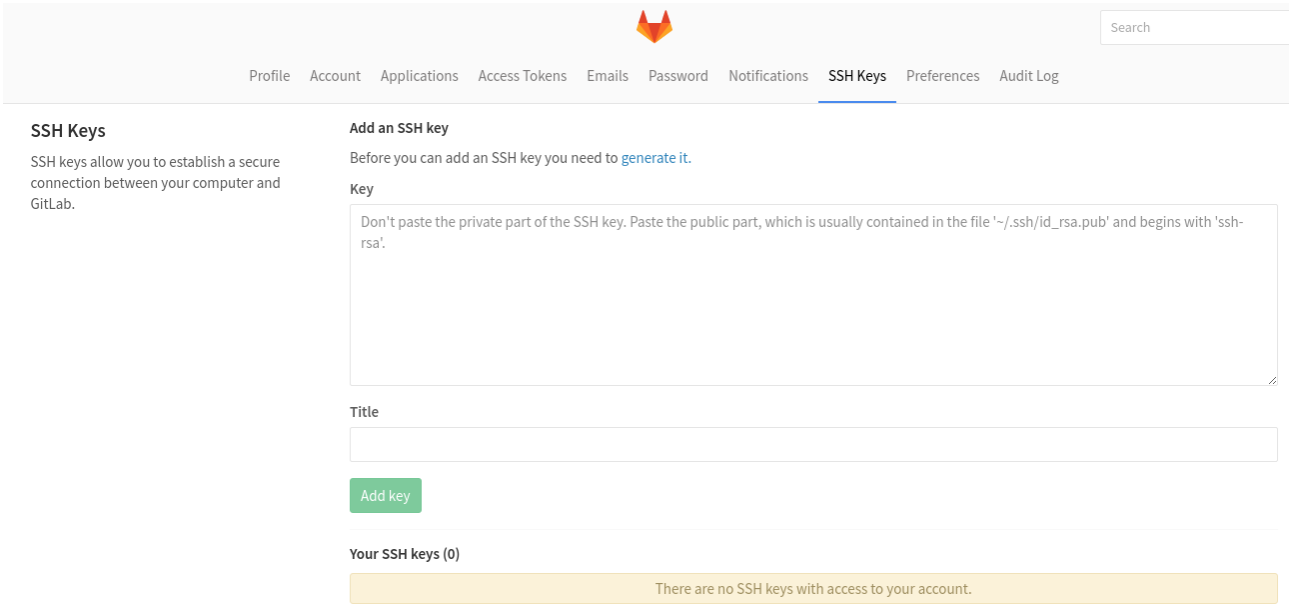


Do konta gitlab można przypisać więcej niż jeden adres poczty. Opcja ta jest przydatna w chwili, gdy chcemy na różne konta otrzymywać różnego rodzaju informacje, np. o zaproszeniach do projektów, o zgłoszonych błędach do naszych projektów itd.

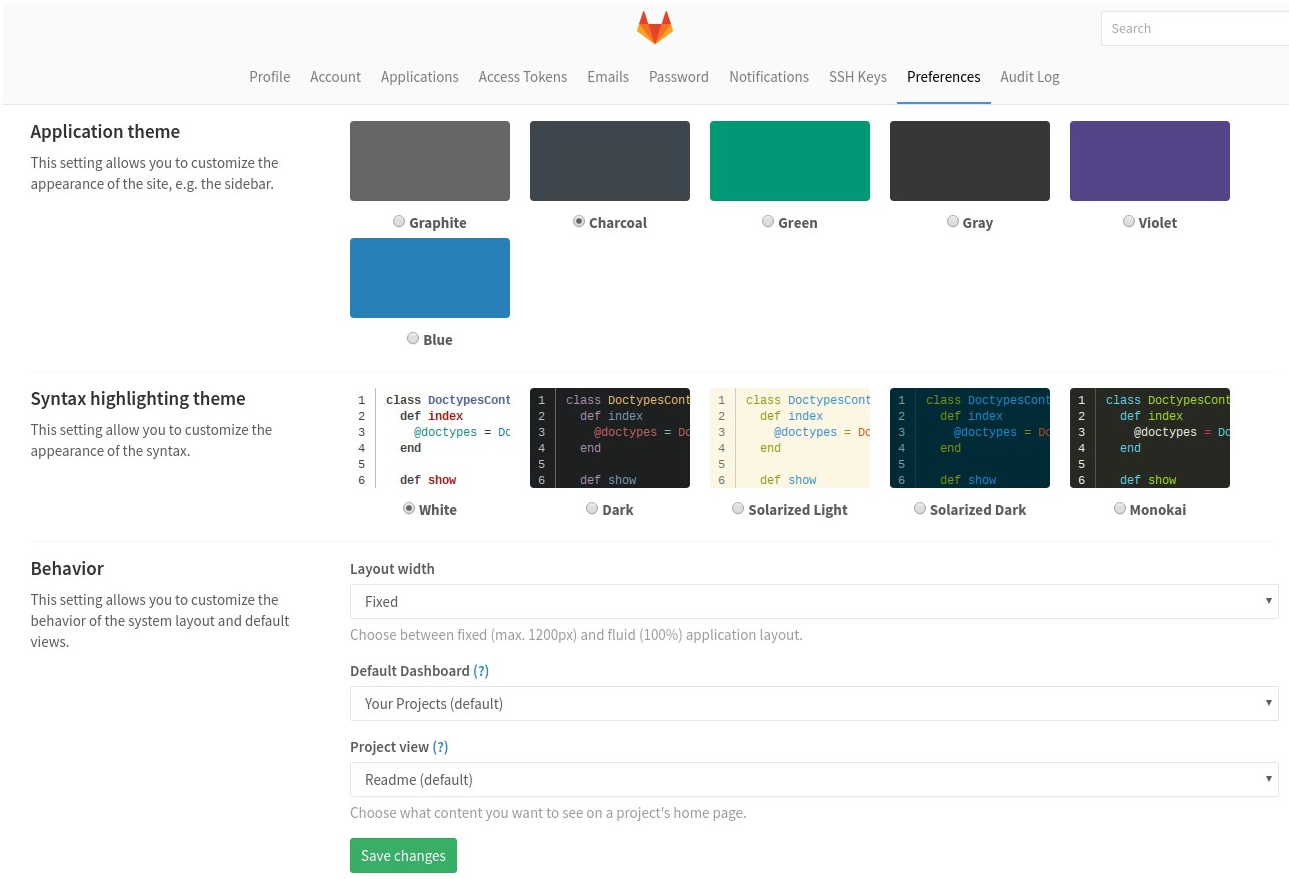


Następna zakładka pozwala na zmianę hasła do konta; jeżeli nie pamiętamy obecnego hasła to z tej zakładki możemy je odzyskać.



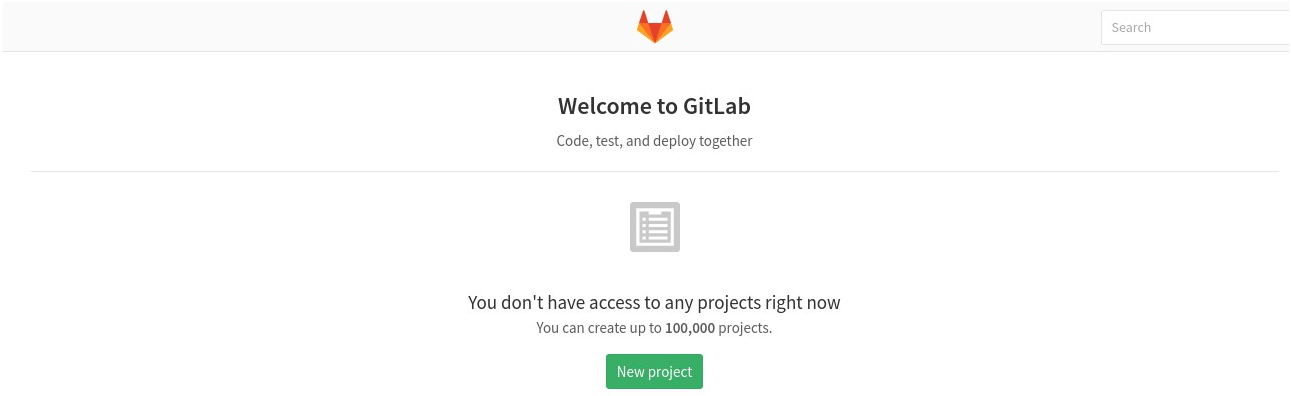


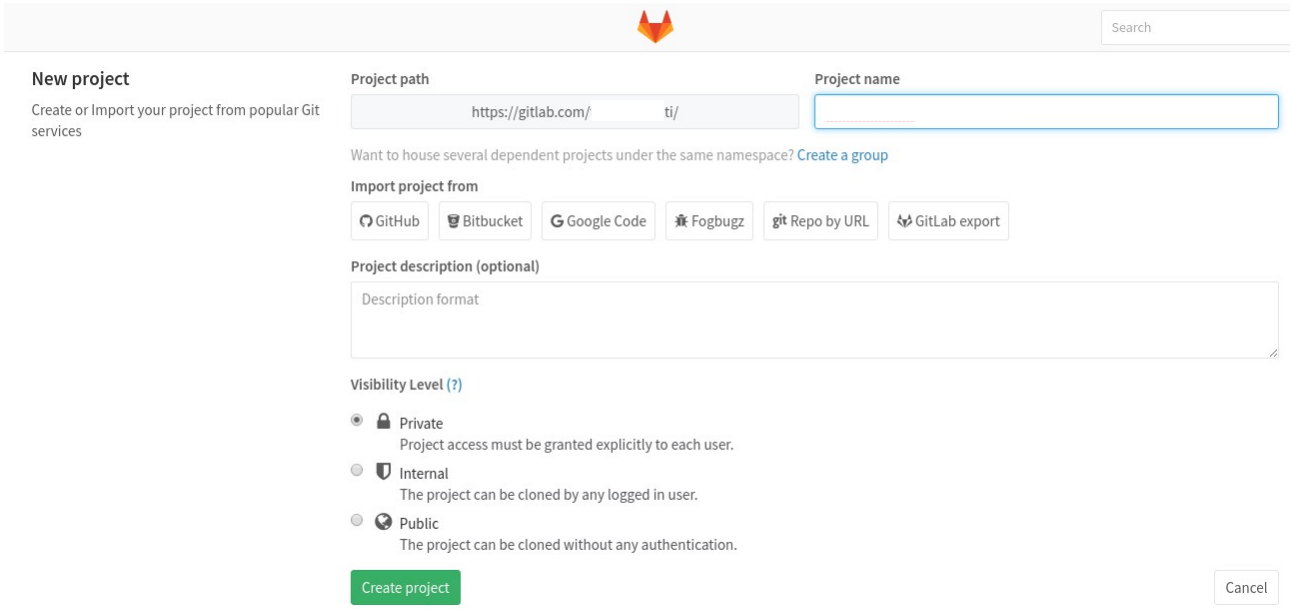
Ta zakładka pozwala na dodanie klucza SSH do naszego konta. Dzięki niemu będziemy mogli korzystać z klonowania/uaktualniania/pobierania projektów bez podawania hasła do konta. Opcja działa jedynie z połączeniami SSH (nie działa dla połączeń HTTPS).



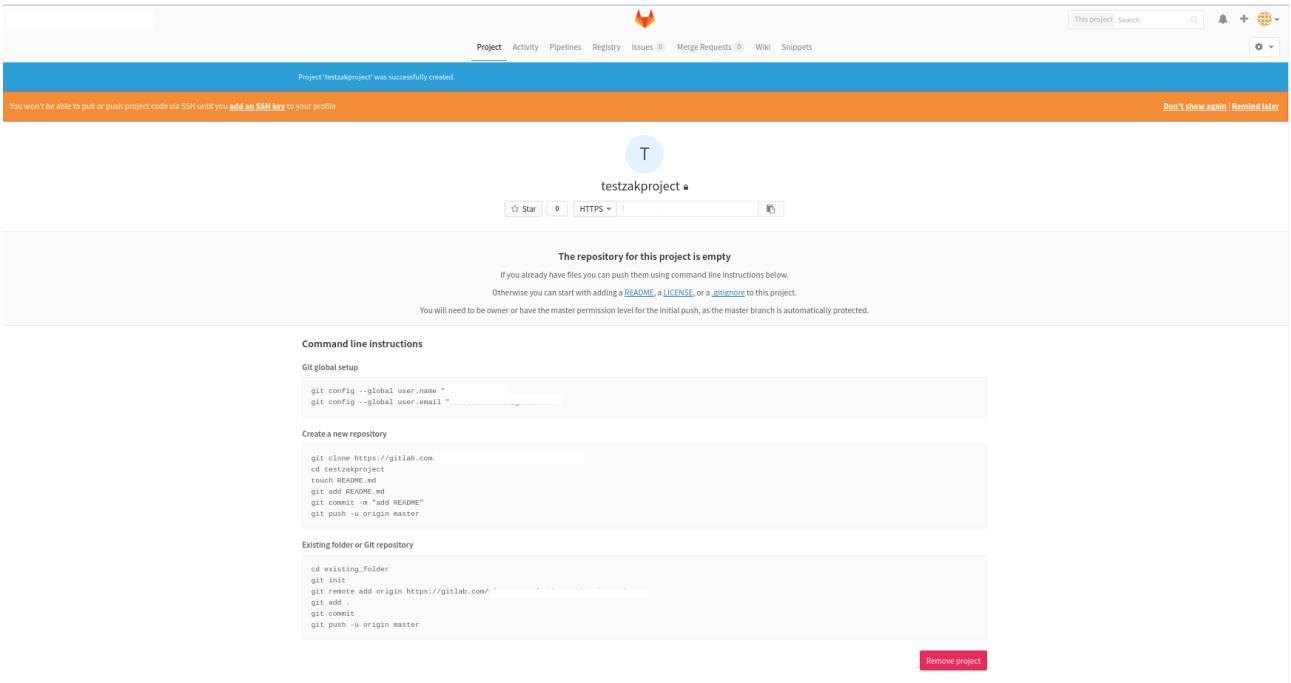
Przedostatnia zakładka pozwala na zmianę wyglądu poszczególnych podstron gitlab. Zmiany dokonane w tej zakładce są przeważnie kosmetyczne i nie mają wpływu na mechanikę naszych repozytoriów.

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.





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



Gitlab wyswietli 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 użytkowanej 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/tlatocateculti/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:

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
p11-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': [redacted]
Password for 'https://[redacted]@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/[redacted]git
 * [new branch]      master -> master
```

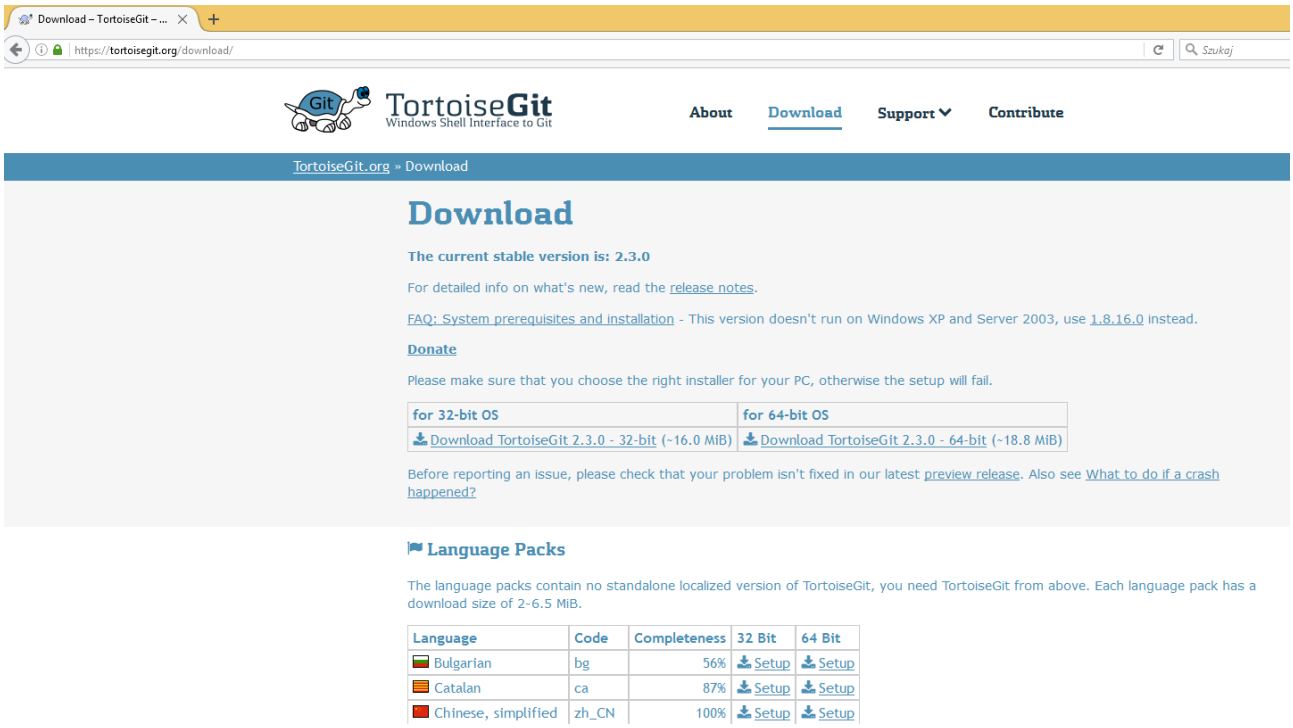
W odpowiedzi otrzymamy komunikaty jak na powyższy zrzucie (tak jak w przypadku pobierania repozytorium zostaniemy poproszeni o podanie loginu oraz hasła). Gotowe. Zmiany można podziwiać na naszym serwerze.

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/

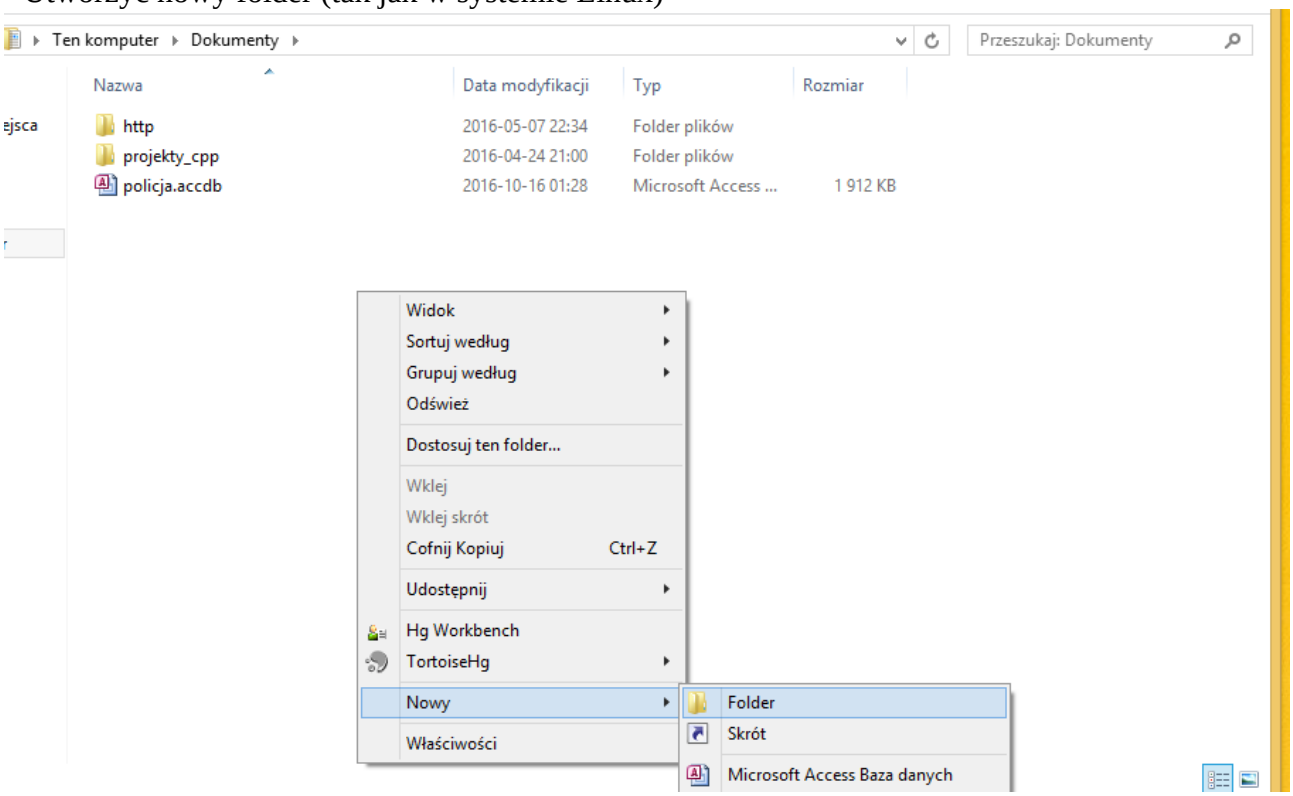


Instalacja polega na klikaniu „Dalej” (można edytować listę składników do instalacji, jednak nie jest to wymagane). Po instalacji narzędzie wymaga posiadania narzędzi Git dla Windows. Można je pobrać poprzez stronę https://git-for-windows.github.io/

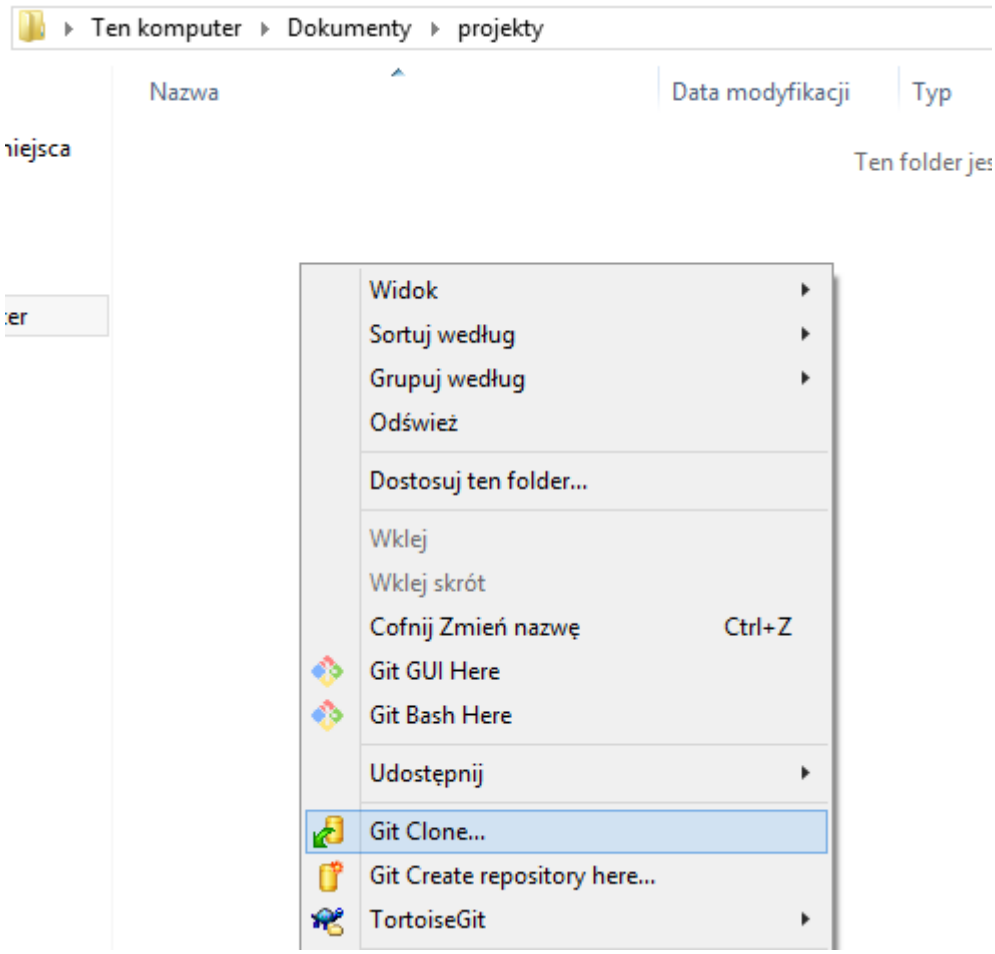
Podobnie jak to miało miejsce z Tortoise instalację można wykonać dokonując domyślnych wyborów. Teraz wszystko powinno działać.

Wystarczy:

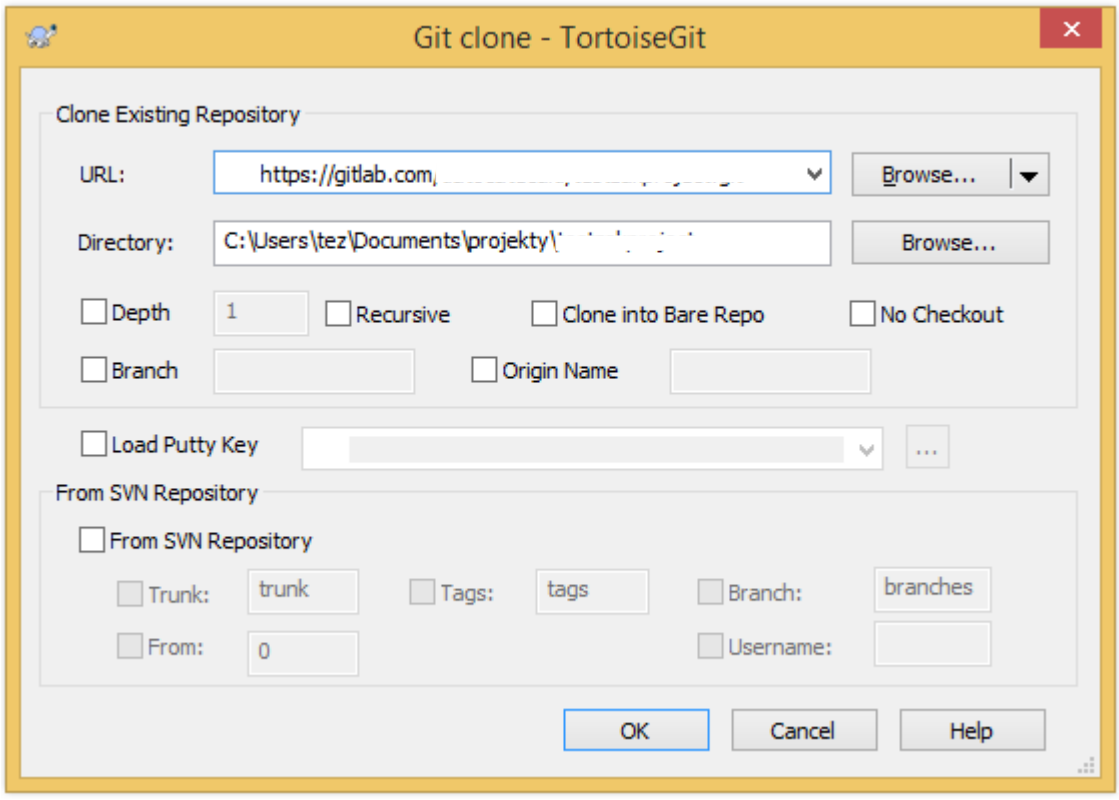
- Utworzyć nowy folder (tak jak w systemie Linux)



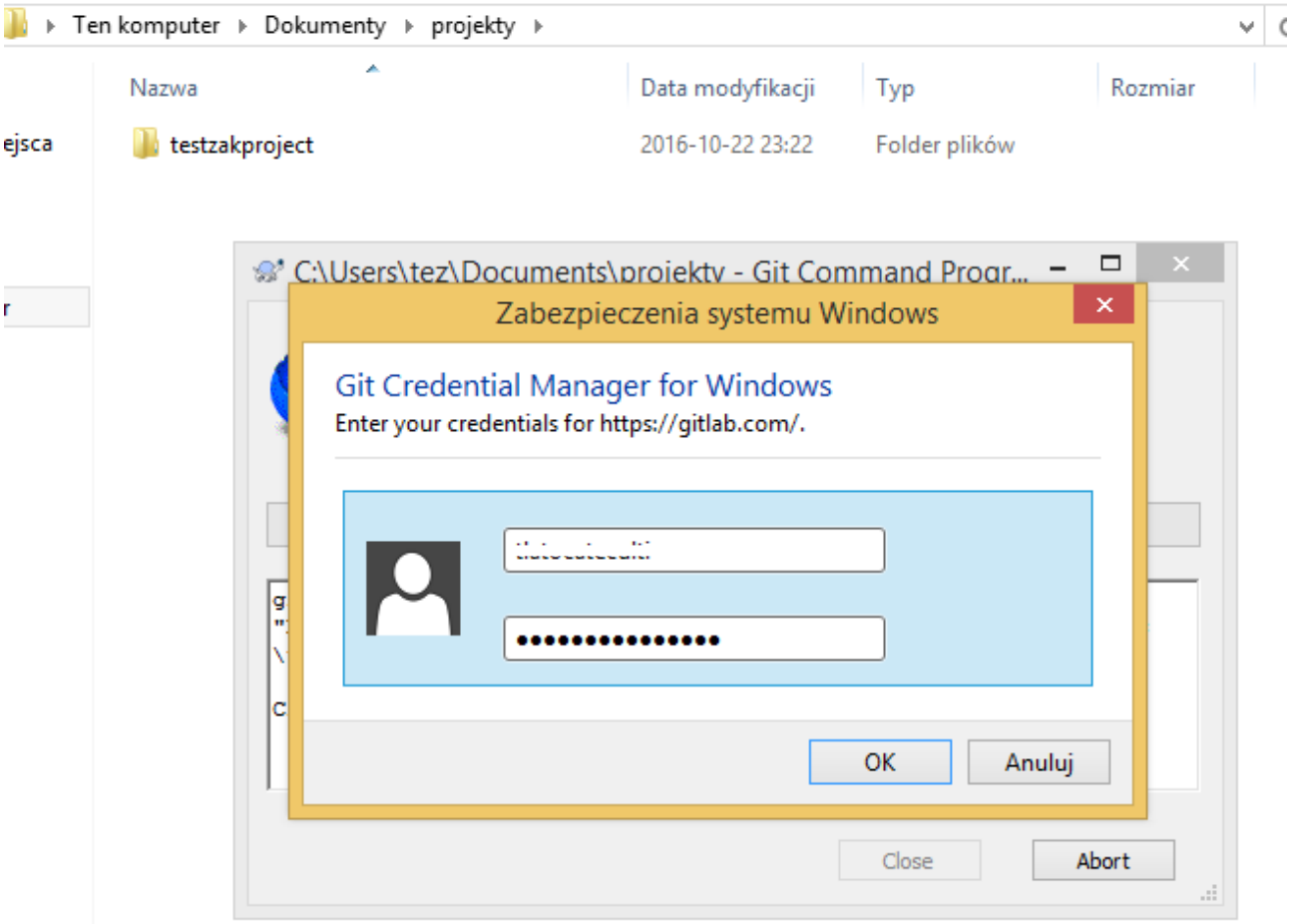
- W folderze kliknąć prawym przyciskiem myszy i wybrać opcję Clone



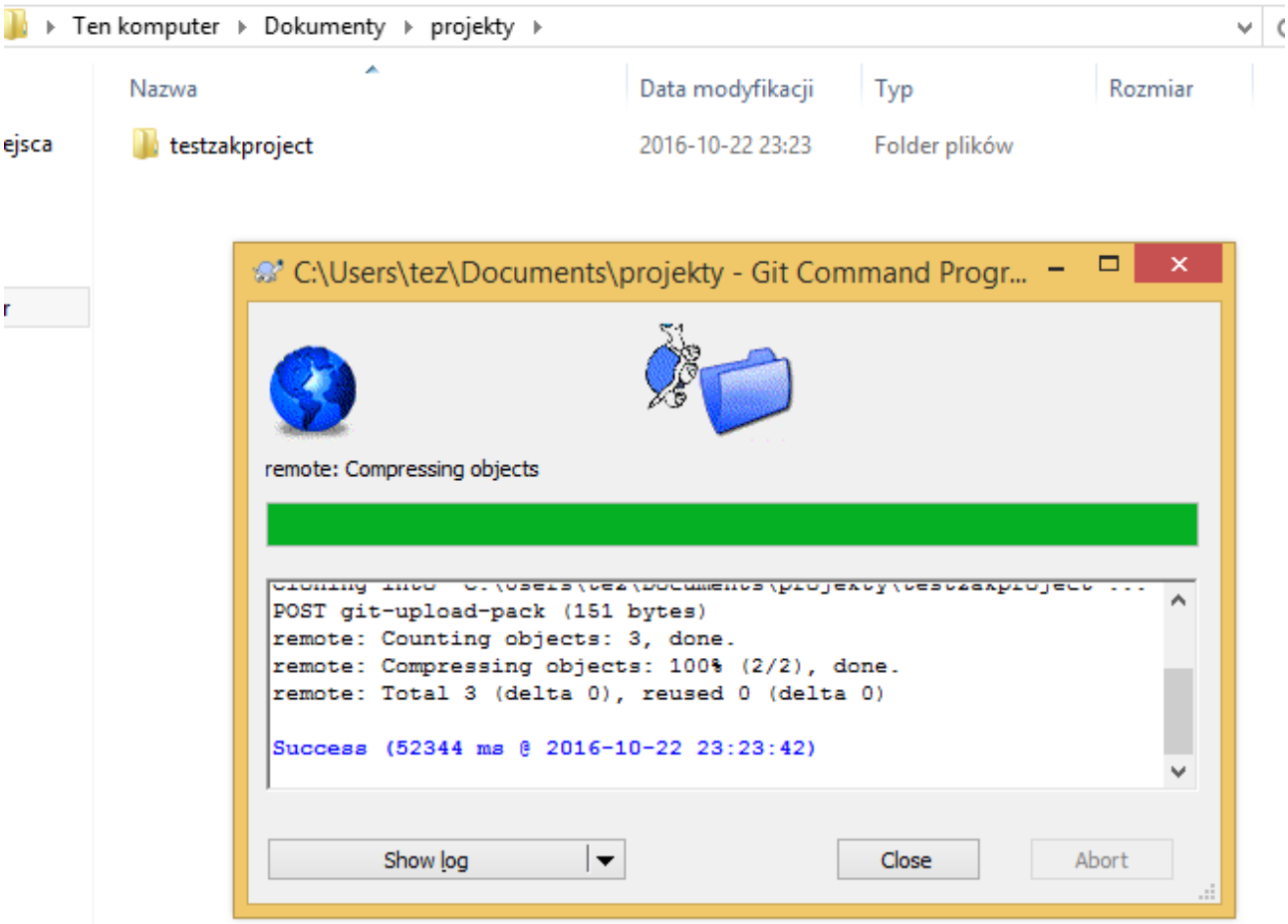
- W nowym oknie kopiujemy adres do naszego repozytorium:



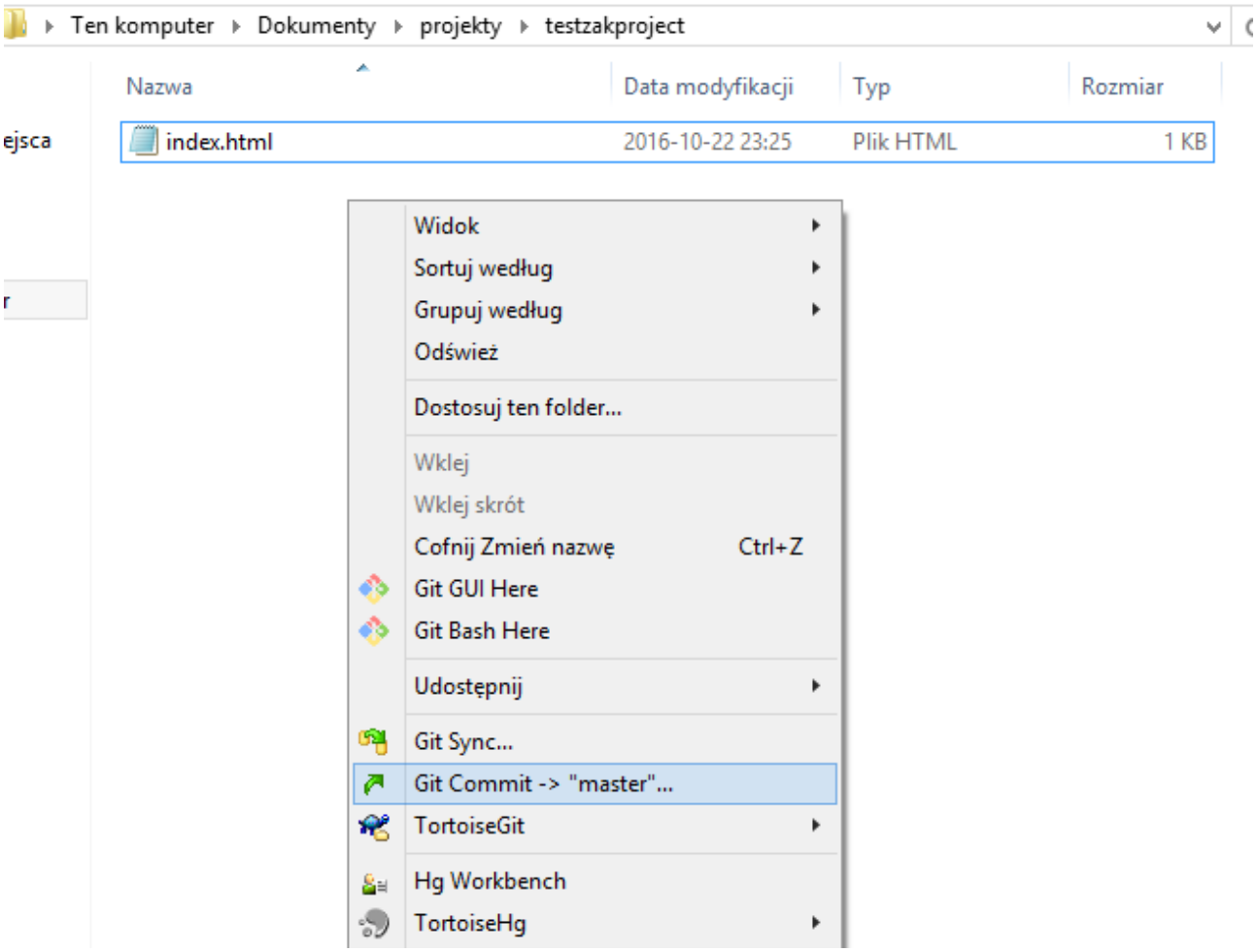
- Autoryzujemy się



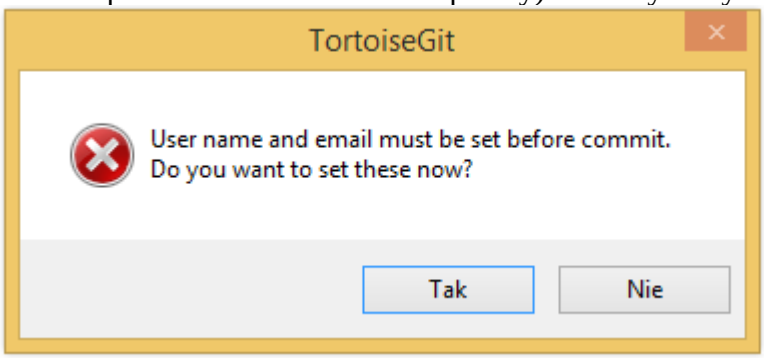
- Gotowe; nasz projekt został właśnie zaimportowany (otrzymujemy stosowną informację):



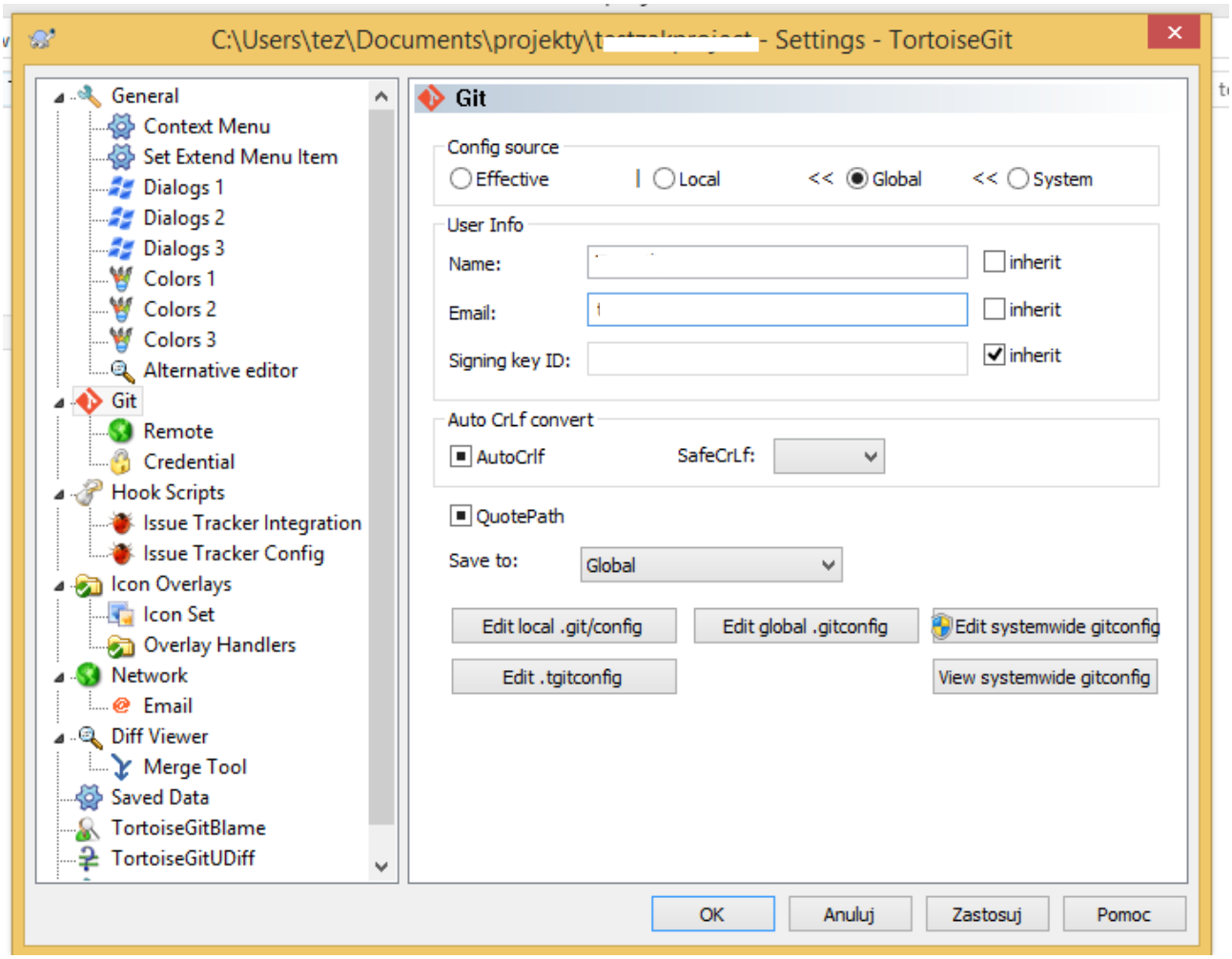
Teraz możemy edytować pliki znajdujące się w naszym projekcie tak jak chcemy i lubimy (przed dowolne narzędzia). Jeżeli będziemy chcieli dokonać zapisu repozytorium robimy to będąc w katalogu projektu; klikamy w dowolnym miejscu prawym przyciskiem myszy i wybieramy opcję Git Commit → „Master”…



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:

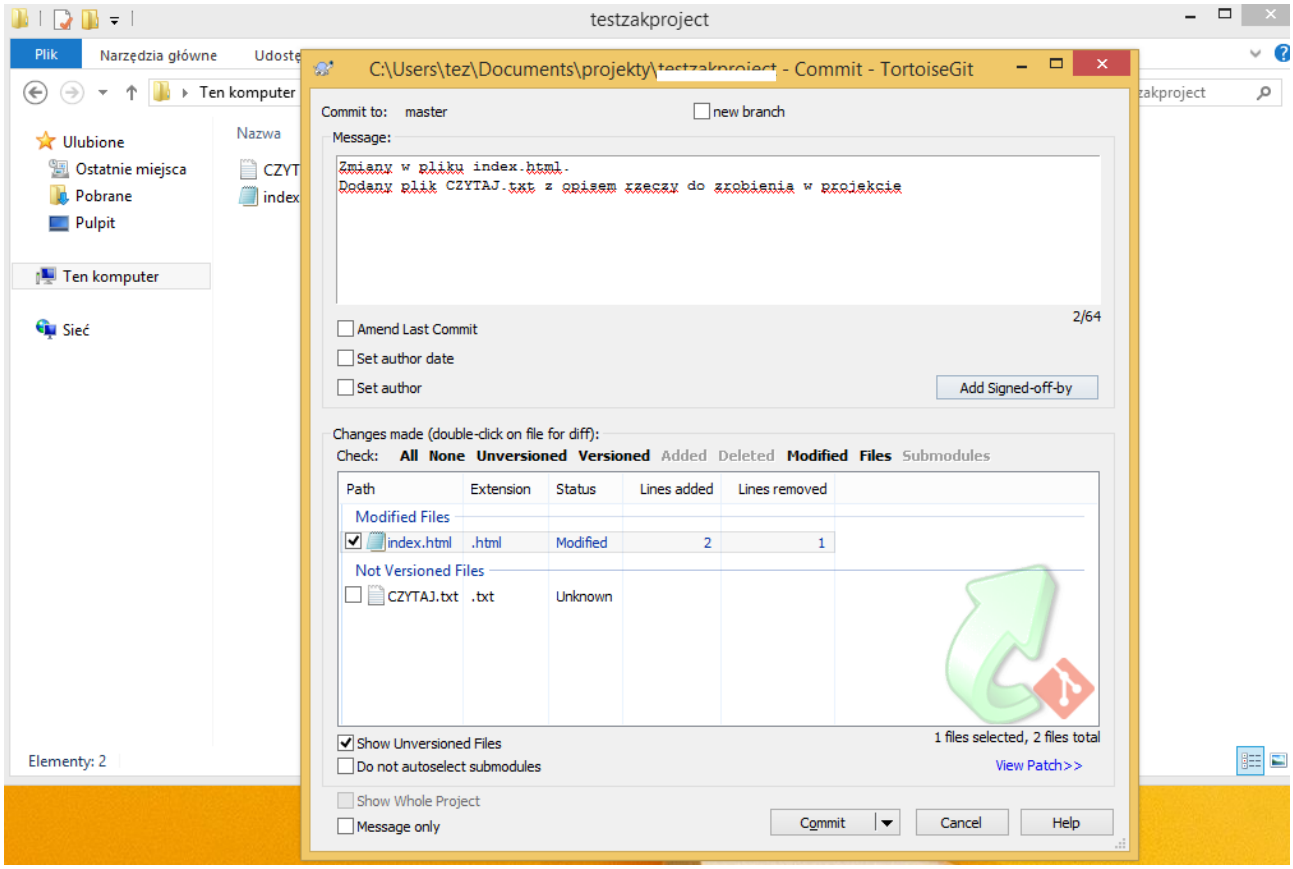


Odpowiadamy twierdząco, po czym dokonujemy zmian w odpowiednich polach (zaznaczone czerwoną ramką):

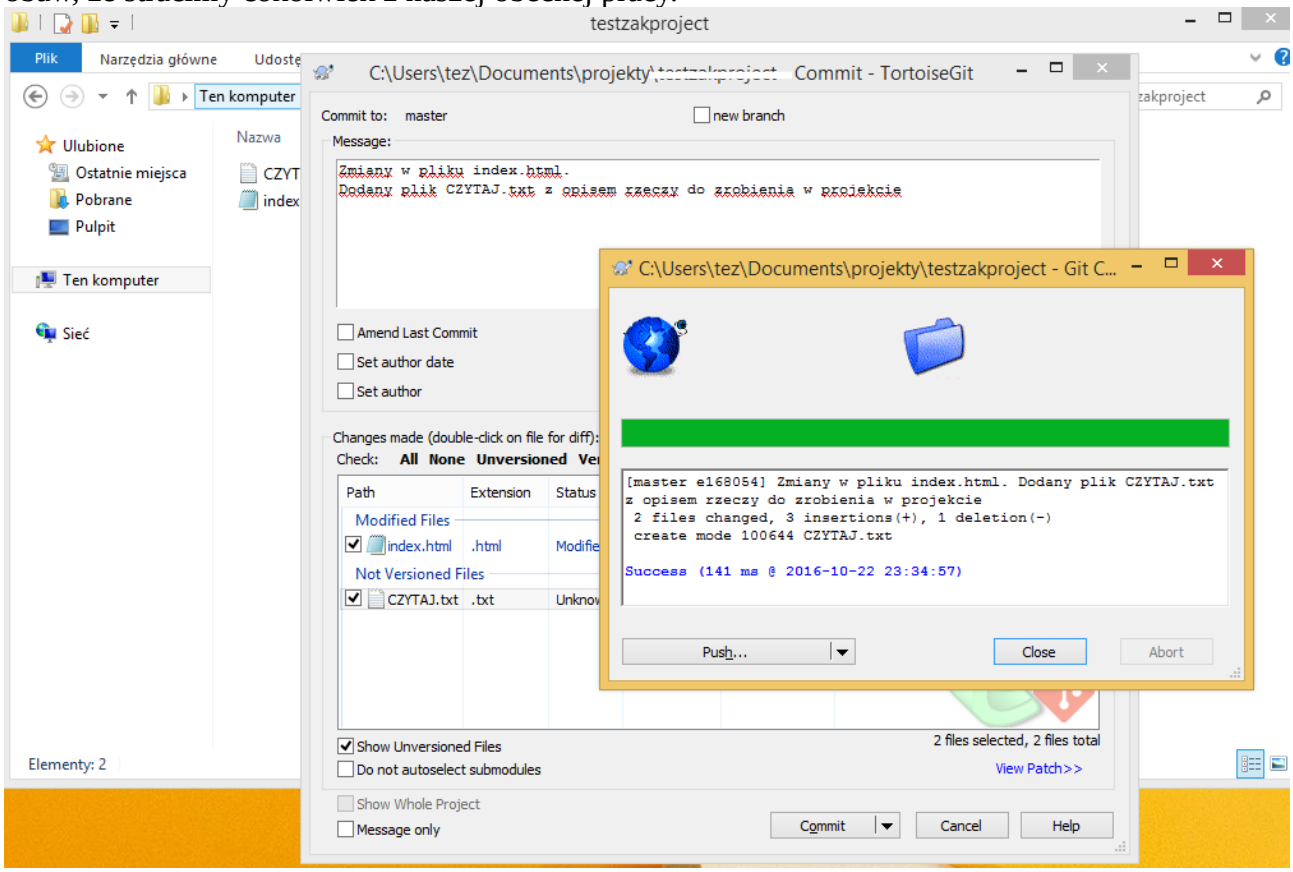


Proszę zauważyć, że narzędzie posiada znacznie więcej opcji. Można je dowolnie edytować, w zależności od swoich potrzeb (aktualny materiał nie będzie się nimi zajmować).

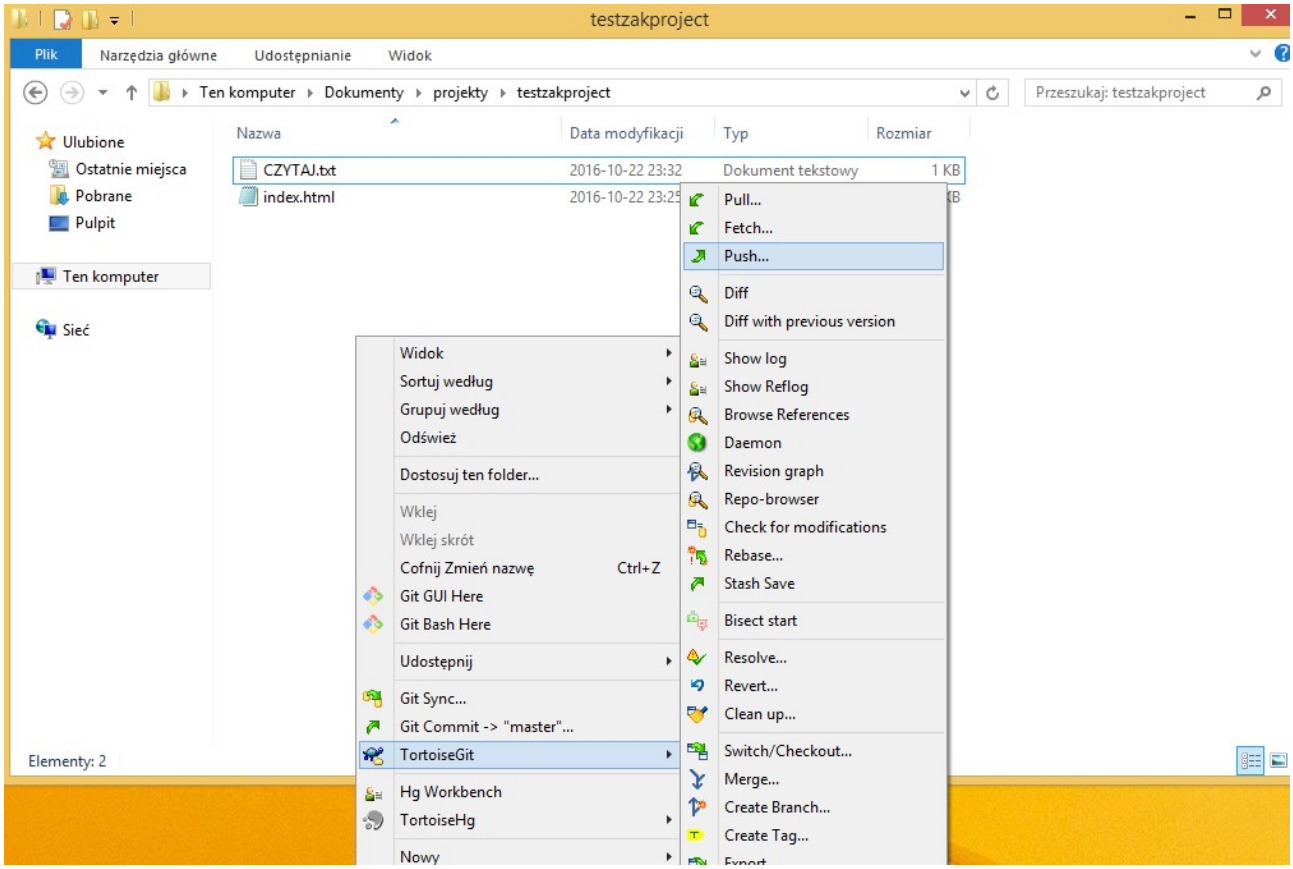
Gdy klikniemy OK przejdziemy do okna zatwierdzenia zmian w projekcie. Proszę zwrócić uwagę na plik CZYTAJ.txt – jest to plik dodany aktualnie, jeszcze nie dodany do repozytorium. Możemy go do niego dołączyć poprzez zaznaczenie go.

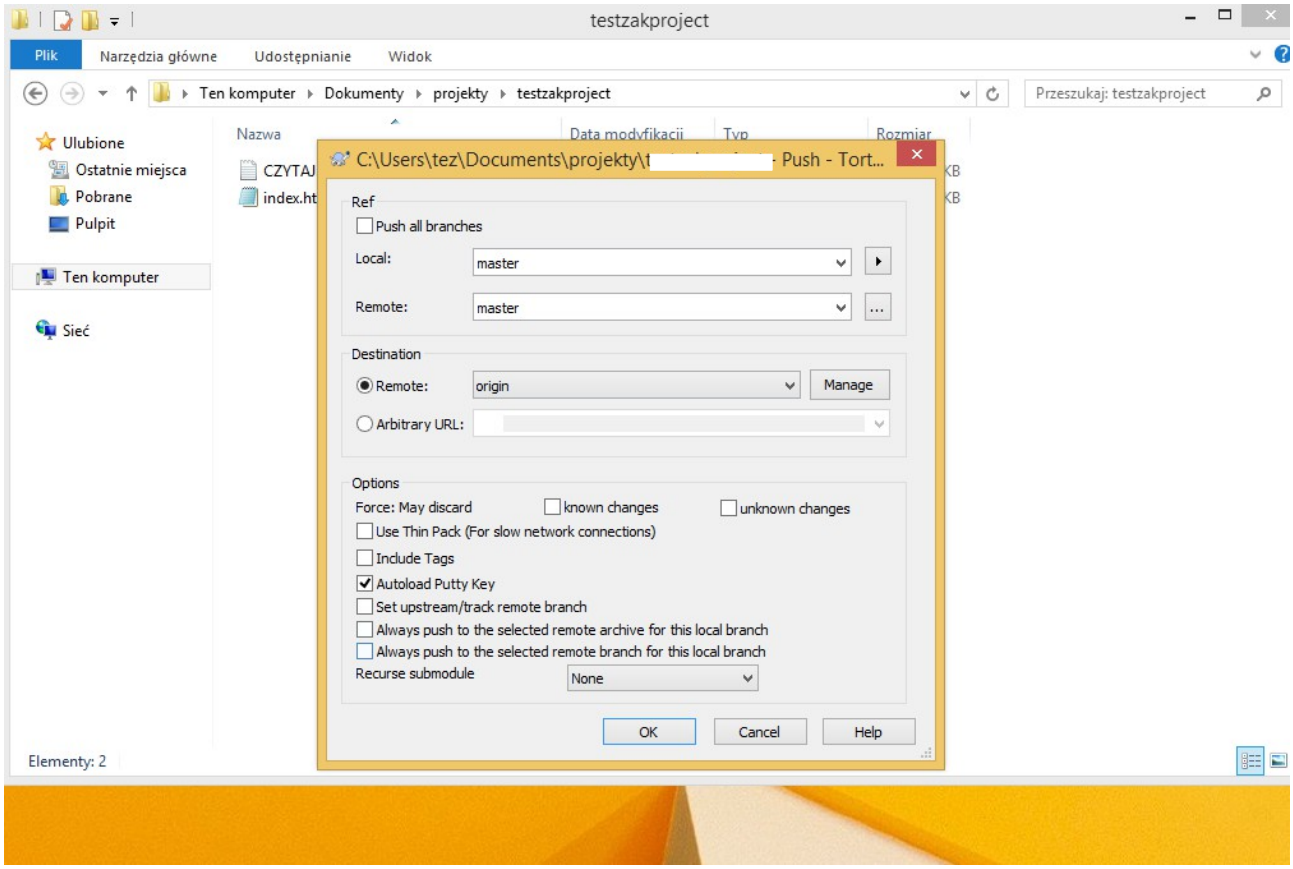


Klikając Commit akceptujemy zmiany. Możemy bezpiecznie dalej modyfikować pliki projektu bez obaw, że stracimy cokolwiek z naszej obecnej pracy.



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





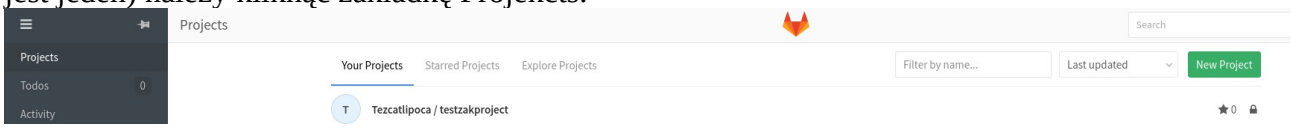
Projekt został właśnie wysłany na serwer.

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.

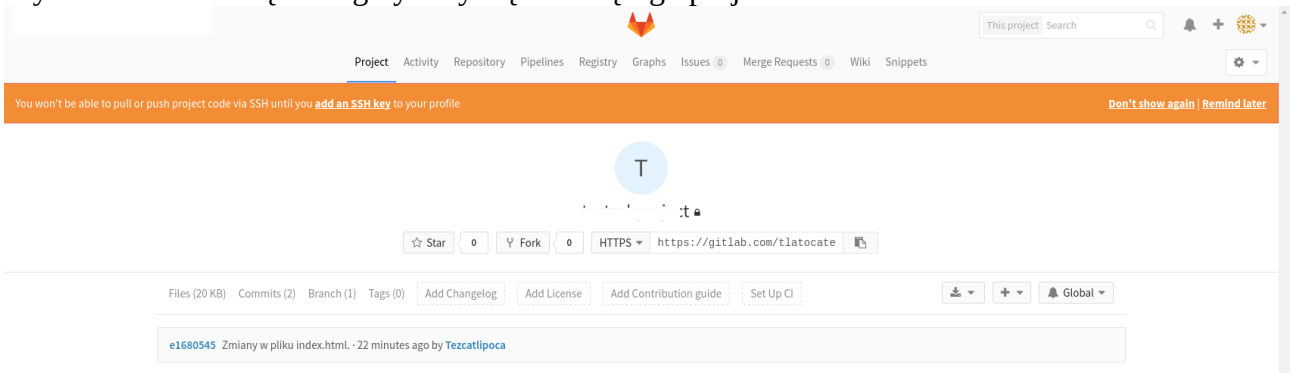
4. Korzystanie z projektu poprzez stronę WWW.

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 Projekcts.



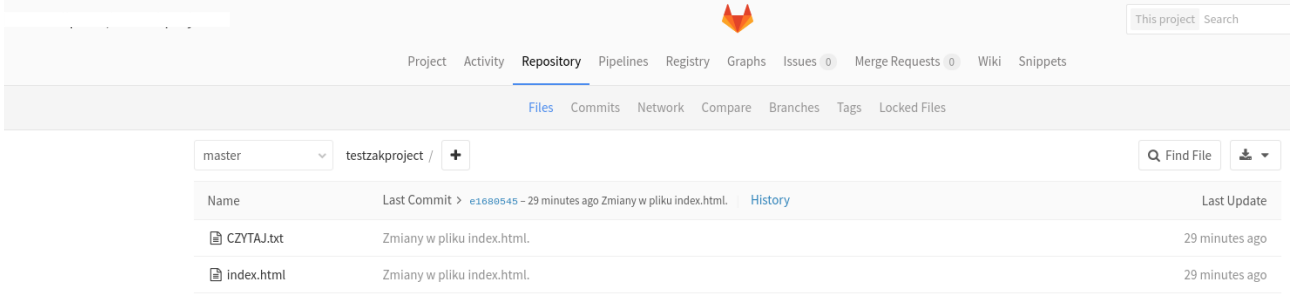
Z listy klikamy na nasz projekt (czerwona ramka)

Wyświetlone zostaną szczegóły dotyczące bieżącego projektu

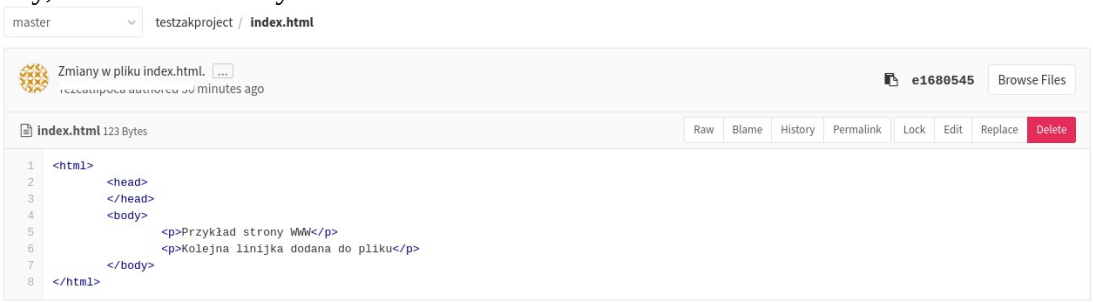


Z tego ekranu możemy oznaczyć projekt gwiazdką (ulubiony/wyróżniony), rozwidlić go na nowy projekt (Fork; opcji tej używa się by zacząć NOWY projekt na bazie aktualnie przeglądanego), bądź skopiować jego adres celem klonowania go/udostępnienia innym osobom. Ponadto możemy przeglądać pliki wchodzące w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

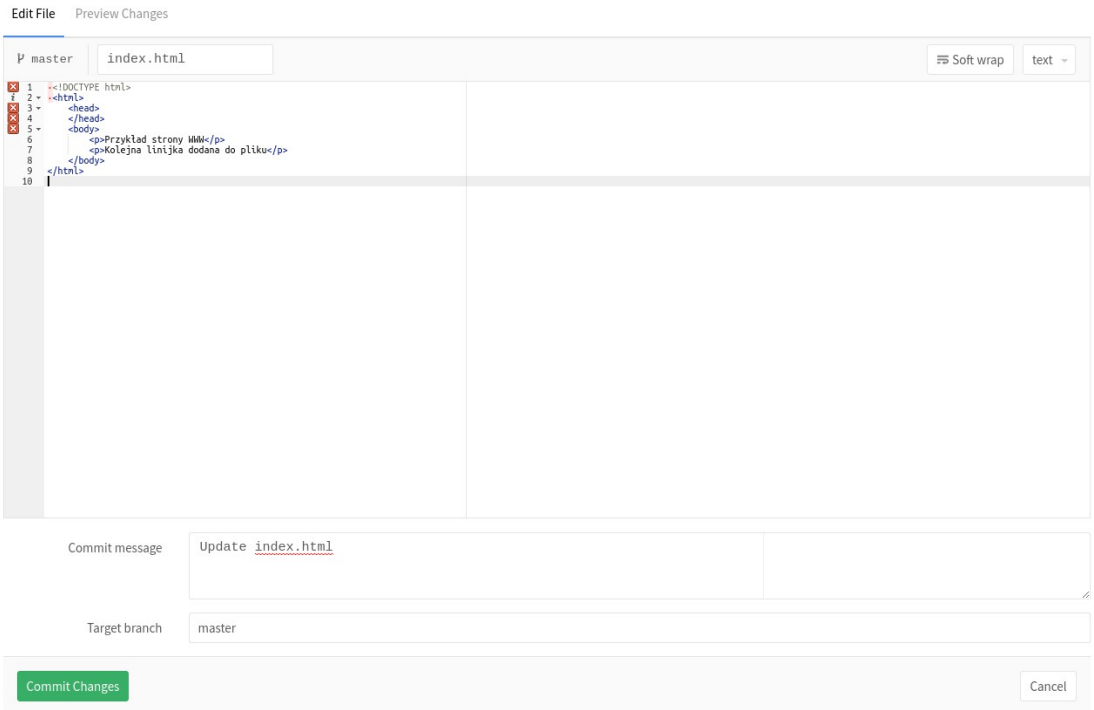
Otwierając pliki możemy zapoznać się z listą dostępnych plików w projekcie.



Po wybraniu któregokolwiek pliku widzimy jego zawartość, a także mamy możliwość edytowania go, podmiany, zablokowania czy skasowania.



Edycja pliku index.html:



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