Pierwsze repozytorium: od pustego projektu do pierwszego pusha

Poziom: podstawy **Dotyczy: Gitea 1.27, Git dla Windows i Linux** **Czas: ~40 min**

Ta strona prowadzi osobę, która nigdy nie używała Gita, od pierwszego repozytorium w organizacji zespołu Oli do zaakceptowanego pull requesta. Po drodze tłumaczymy każde nowe pojęcie osobno, z analogią.

Skróty użyte na tej stronie (5)

Skrót	Co znaczy
CA	Certificate Authority: urząd certyfikacji, który podpisuje certyfikaty
HTTPS	HTTP zabezpieczony szyfrowaniem TLS
SSH	Secure Shell: szyfrowane zdalne logowanie do serwera
SSL	Secure Sockets Layer: dawna nazwa szyfrowania, dziś formalnie TLS
TLS	Transport Layer Security: szyfrowanie połączeń, np. w HTTPS

Fiszka

TYLKO ODCZYT

Utworzenie klucza SSH na Windowsie: ssh-keygen

```
ssh-keygen -t ed25519 -C "tomek.nowak@zielona.example"
```

Klucz publiczny i prywatny trafiają domyślnie do `C:\Users\tomek.nowak\.ssh\`. Zawartość pliku `id_ed25519.pub` wklejamy potem w ustawieniach konta Gitea, w sekcji SSH Keys.

□□ **SSH** = Secure Shell: szyfrowane zdalne logowanie do serwera · **ssh-keygen** = wygeneruj parę kluczy SSH albo sprawdź odcisk klucza

□□ TYLKO ODCZYT

Sklonowanie repozytorium przez SSH: git clone

```
git clone git@git.zielona.example:zielona/cennik.git
```

Pobiera pełną kopię repozytorium razem z całą historią commitów na dysk lokalny, do nowego katalogu o nazwie repozytorium.

□□ **SSH** = Secure Shell: szyfrowane zdalne logowanie do serwera · **git** = system kontroli wersji (nazwa to brytyjski slang: nieznośny typ)

□□ TYLKO ODCZYT

Zapisanie zmiany do śledzenia: git add

```
git add cennik.xlsx
```

Oznacza plik jako gotowy do zapisania w kolejnym commicie. Bez tego kroku Git widzi zmianę na dysku, ale jeszcze jej nie śledzi.

□□ **git** = system kontroli wersji (nazwa to brytyjski slang: nieznośny typ)

□□ ZMIENIA SYSTEM

Zatwierdzenie zmiany: git commit

```
git commit -m "Aktualizacja cen kawy na czwarty kwartal"
```

Zapisuje oznaczone zmiany jako nowy, nazwany punkt w historii projektu, razem z autorem i znacznikiem czasu.

□□ **git** = system kontroli wersji (nazwa to brytyjski slang: nieznośny typ)

□□ ZMIENIA SYSTEM

Wysłanie zmian na serwer: git push

```
git push origin nazwa-galezi
```

Przesyła commity zapisane lokalnie do repozytorium na `git.zielona.example`, żeby stały się widoczne dla reszty zespołu.

git = system kontroli wersji (nazwa to brytyjski slang: nieznośny typ)

TYLKO ODCZYT

Pobranie zmian z serwera: git pull

```
git pull origin main
```

Pobiera i od razu dołącza do lokalnej kopii commity, które ktoś inny wysłał na serwer od czasu ostatniego pobrania.

git = system kontroli wersji (nazwa to brytyjski slang: nieznośny typ)

TYLKO ODCZYT

Utworzenie nowej gałęzi roboczej: git checkout -b

```
git checkout -b nazwa-galezi
```

Tworzy nową gałąź jako kopię bieżącego stanu i od razu na nią przełącza, żeby dalsze commity trafiały tam, a nie na `main`.

git = system kontroli wersji (nazwa to brytyjski slang: nieznośny typ)

TYLKO ODCZYT

Przełączenie Gita na magazyn certyfikatów Windows: git config --global http.sslbackend

```
git config --global http.sslbackend schannel
```

Naprawia błąd certyfikatu SSL przy operacjach przez HTTPS na Windowsie, opisany niżej w sekcji o częstych błędach.

SSL = Secure Sockets Layer: dawna nazwa szyfrowania, dziś formalnie TLS · **HTTPS** = HTTP zabezpieczony szyfrowaniem TLS · **git** = system kontroli wersji (nazwa to brytyjski slang: nieznośny

typ)

Z życia wzięte

Z ŻYCIA WZIĘTE

Tomek klonuje repozytorium `cennik` na swój nowy laptop z Windows i przy pierwszym `git clone` przez HTTPS dostaje czerwony komunikat: `SSL certificate problem: unable to get local issuer certificate`. Przegląda tę samą stronę `git.zielona.example` w przeglądarce bez żadnego ostrzeżenia o certyfikacie, więc przez chwilę podejrzewa awarię serwera, zanim Ola pokazuje mu jedną komendę z sekcji Częste błędy.

Jak to działa

Rozwiń: jak to działa pod spodem

Pojęcie: repozytorium. Katalog, w którym Git śledzi każdą zmianę plików, razem z pełną historią, kto i kiedy ją wprowadził. **Analogia:** segregator z każdą wersją dokumentu spiętą osobno i podpisaną datą, zamiast jednego pliku nadpisywanego w kółko.

Pojęcie: commit. Zapisany w historii punkt, opisujący dokładnie, co się zmieniło względem poprzedniego punktu, kto to zrobił i kiedy. **Analogia:** zapis stanu gry: w każdej chwili można wrócić do dowolnego wcześniejszego zapisu, nawet jeśli od tamtej pory wydarzyło się dużo więcej.

Pojęcie: gałąź (branch). Równoległa linia commitów, zaczynająca się od wspólnego punktu z inną gałęzią, na przykład `main`. **Analogia:** kalka nałożona na oryginalny szkic: rysujemy na niej dowolne poprawki, a oryginał pod spodem zostaje nietknięty, dopóki nie zdecydujemy się przenieść poprawek na stałe.

Pojęcie: push. Wysłanie commitów zapisanych lokalnie do repozytorium na serwerze. **Analogia:** wgranie zaktualizowanego pliku na wspólny dysk sieciowy, żeby inni widzieli nową wersję.

Pojęcie: pull. Pobranie i dołączenie do lokalnej kopii commitów, które ktoś inny wcześniej wysłał na serwer. **Analogia:** odświeżenie widoku wspólnego dysku, żeby zobaczyć to, co dopisali inni od naszej ostatniej wizyty.

Pojęcie: klon (clone). Pierwsze pobranie repozytorium z serwera na dysk lokalny, razem z całą jego historią, nie tylko bieżącym stanem plików. **Analogia:** kserokopia całego segregatora ze wszystkimi wcześniejszymi wersjami dokumentów, nie tylko ostatniej kartki.

Pojęcie: organizacja. Kontener grupujący repozytoria i zespoły w Gitea, niezależny od kont osobistych poszczególnych osób. W Zielonej repozytoria zespołu programistów leżą w organizacji `zielona`, a nie na koncie osobistym Oli: odejście Oli z firmy nie zabiera ze sobą repozytoriów zespołu.

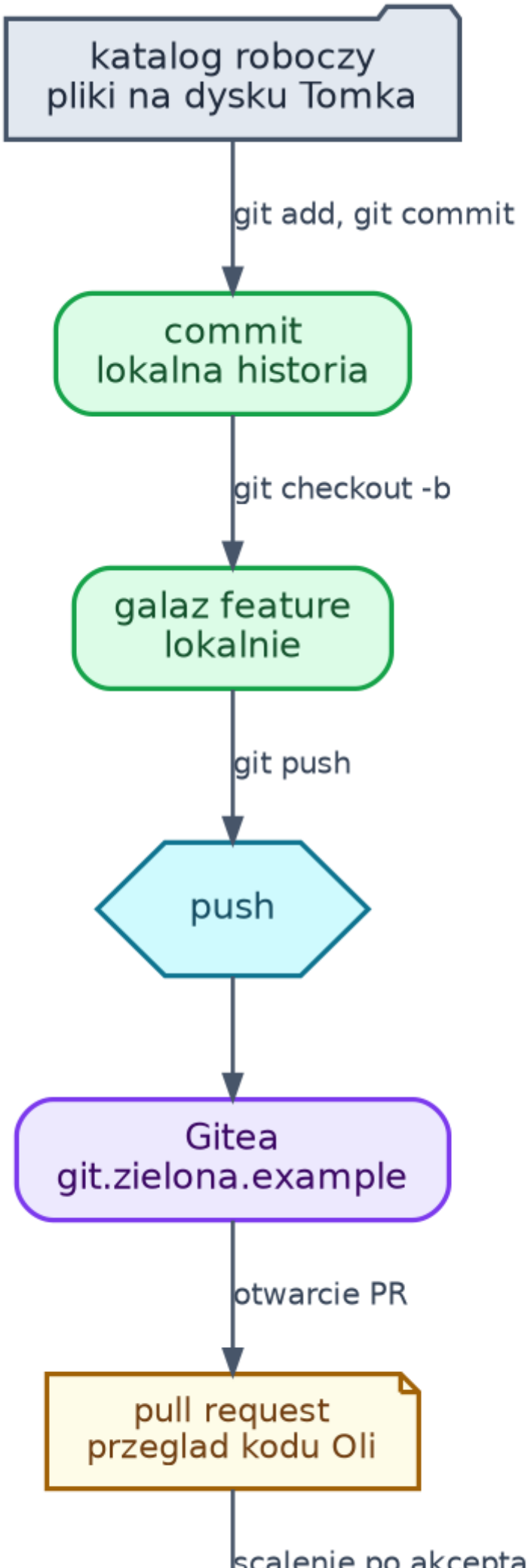
Pojęcie: pull request. Prośba o przeniesienie commitów z jednej gałęzi (na przykład gałęzi Tomka) do drugiej (na przykład `main`), razem z widokiem różnic gotowym do przejrzenia. **Analogia:** podanie z naniesionymi poprawkami, które przełożony przegląda i albo akceptuje w całości, albo odsyła z komentarzem, zanim trafi do oficjalnej dokumentacji.

Pojęcie: przegląd kodu (code review). Etap pull requesta, w którym inna osoba z zespołu, zwykle Ola jako prowadząca, czyta zaproponowane zmiany linijka po linijce, zostawia komentarze i albo prosi o poprawki, albo zatwierdza scalenie.

Pojęcie: scalenie (merge). Ostateczne przeniesienie zaakceptowanych commitów z gałęzi pull requesta do gałęzi docelowej, zwykle `main`. Od tego momentu zmiana staje się częścią głównej linii projektu.

Pojęcie: ochrona gałęzi (branch protection). Reguła w ustawieniach repozytorium blokująca bezpośredni push na wskazaną gałąź, zwykle `main`: każda zmiana musi przejść

przez pull request i przegląd kodu, zamiast trafiać na główną gałąź od razu, bez niczyjej wiedzy.


katalog roboczy
pliki na dysku Tomka
git add, git commit
commit
lokalna historia
git checkout -b
galaz feature
lokalnie
git push
push
Gitea
git.zielona.example
otwarcie PR
pull request
przeglad kodu Oli
scalenie po akceptacji

Przepływ pracy w Gitea: od katalogu roboczego przez commit i push do pull requesta i scalenia z main (kliknięcie otwiera pełny rozmiar)

Praca przez SSH albo przez HTTPS z tokenem

Do codziennej pracy Tomek wybiera jeden z dwóch sposobów łączenia się z serwerem. Klucz SSH, dodany raz w ustawieniach konta, działa bez podawania hasła przy każdej operacji `git clone`/`push`/`pull`. Token dostępu osobistego, generowany w ustawieniach konta (Settings → Applications → Generate New Token), działa jak hasło używane zamiast właściwego hasła konta, z zakresem uprawnień węższym niż całe konto: przydaje się w skryptach automatyzujących i tam, gdzie klucza SSH nie da się użyć.

Pierwszy push z Windowsa i z Linuksa

Sam protokół Git działa identycznie na obu systemach: różni się tylko powłoka, w której wpisujemy komendy (PowerShell albo Git Bash na Windowsie, bash na Linuksie) i domyślny mechanizm weryfikacji certyfikatów, opisany niżej. Kolejność kroków zostaje ta sama: `git clone`, edycja pliku, `git add`, `git commit`, `git checkout -b` na nową gałąź, `git push origin nazwa-galezi`, a na końcu pull request w interfejsie Gitea.

Błąd certyfikatu SSL przy operacjach Git na Windowsie

Objaw. Operacja `git clone`/`push`/`pull` po HTTPS na stacji Windows kończy się błędem `SSL certificate problem: unable to get local issuer certificate`, mimo że certyfikat serwera Gitea jest poprawny i zaufany przez przeglądarkę na tej samej stacji.

Przyczyna. Domyślny mechanizm weryfikacji certyfikatów w Gicie na Windowsie korzysta z własnego magazynu zaufanych certyfikatów, dostarczanego razem z Gitem (backend oparty o OpenSSL), zamiast z systemowego magazynu certyfikatów Windows. Certyfikat zaufany przez system i przeglądarkę nie musi być automatycznie zaufany przez ten wbudowany magazyn Gita.

Naprawa. Przełączamy globalnie backend kryptograficzny Gita na `schannel`, czyli natywny mechanizm Windows:

```
git config --global http.sslbackend schannel
```

Od tej zmiany Git korzysta z tego samego magazynu zaufanych certyfikatów co reszta systemu operacyjnego, więc certyfikat wewnętrznego urzędu certyfikacji rozpoznawany przez system jest rozpoznawany też przez Gita. Alternatywą, gdy z jakiegoś powodu nie chcemy zmieniać backendu globalnie dla wszystkich repozytoriów, jest wskazanie Gitowi wprost pliku z certyfikatem firmowego CA:

```
git config --global http.sslCAInfo "C:\CA\ca-zielona.crt"
```

> □□ Klonowanie przez SSH, opisane wyżej, w ogóle nie korzysta z tego mechanizmu weryfikacji certyfikatów TLS: jeśli błąd certyfikatu przeszkadza, a firmowy CA jeszcze nie trafił do zaufanych źródeł na stacjach, SSH bywa szybszym obejściem na czas naprawy tego pierwszego.

Częste błędy

Objaw	Przyczyna	Co robimy
`SSL certificate problem: unable to get local issuer certificate` na Windowsie	Git na Windowsie nie ufa firmowemu CA we własnym magazynie certyfikatów	`git config --global http.sslbackend schannel` albo `http.sslCAInfo` ze ścieżką do certyfikatu CA
`remote: Repository not found` mimo poprawnego adresu	literówka w nazwie organizacji albo repozytorium, albo brak uprawnień do niego	sprawdzamy dokładny adres w interfejsie Gitea, sekcja Clone
`! [remote rejected] main -> main (protected branch hook declined)`	próba pushu bezpośrednio na chronioną gałąź `main`	tworzymy gałąź roboczą i wysyłamy zmiany przez pull request
Konflikt przy `git pull` albo scaleniu pull requesta	ta sama linia pliku zmieniona równolegle na dwóch gałęziach	Git zaznacza konfliktujące fragmenty w pliku, poprawiamy ręcznie, `git add` , `git commit` kończący scalenie

□□ Ćwiczenia

1. Użytkownik na stacji Windows zgłasza błąd `unable to get local issuer certificate` przy `git push` do wewnętrznego serwera Gitea, mimo że przeglądarka na tej samej stacji otwiera adres serwera bez ostrzeżenia o certyfikacie. Wyjaśniamy przyczynę rozbieżności.

Odpowiedź

Przeglądarka korzysta z systemowego magazynu zaufanych certyfikatów Windows, w którym certyfikat wewnętrznego urzędu certyfikacji jest już zaufany. Git na Windowsie domyślnie korzysta z własnego, wbudowanego magazynu certyfikatów niezależnego od systemowego, więc certyfikat zaufany przez system nie musi być automatycznie zaufany przez Gita, dopóki nie przełączymy backendu na `schannel`.

1. Tomek pracuje na gałęzi `dodanie-rabatow` i próbuje wysłać zmiany komendą `git push origin main`. Wyjaśniamy, co się prawdopodobnie stanie i jak powinna wyglądać poprawna komenda.

Odpowiedź

Jeśli `main` jest gałęzią chronioną, push zostanie odrzucony niezależnie od nazwy podanej po `origin`, bo dotyczy gałęzi docelowej `main`, a nie gałęzi, na której faktycznie pracuje Tomek. Poprawna komenda to `git push origin dodanie-rabatow`, a przeniesienie zmian na `main` odbywa się przez pull request, nie przez bezpośredni push.

Powiązane: Logowanie kontem z Active Directory · Instalacja Gitea w Dockerze · Gitea Actions: runner i pierwszy workflow

Źródła: docs.gitea.com, rozdziały „Basic Usage" i „Command Line" · `git help config` (sekcje `http.sslbackend` i `http.sslCAInfo`)

Revision #33
Created 2026-09-27 10:32:08 UTC by Tomasz Konieczny
Updated 2026-10-26 14:17:32 UTC by Tomasz Konieczny