Komendy GIT

**Wersja GITa**

git --version

**Dokumentacja**

Dokumentacja komend Gita:

git add --help

git branch --help

git config --help

git init --help

**Konfiguracja**

Lista wszystkich opcji konfiguracyjnych:

git config -l

Ustalenie wartości wybranej opcji konfiguracyjnej:

git config --global user.name „Imie Nazwisko”

git config --global user.mail [you@example.com](mailto:you@example.com)

Globalny plik konfiguracyjny użytkownika:

C:\Users\nazwa-użytkownika\.gitconfig

Plik konfiguracyjny dotyczący konkretnego repozytorium:

.git\config

**Inicjalizacja repozytorium**

Inicjalizacja nowego repozytorium zwykłego w bieżącym folderze:

git init

Inicjalizacja nowego repozytorium zwykłego w podanym folderze:

git init scieżka/do/folderu

Inicjalizacja nowego repozytorium surowego w bieżącym folderze:

git init --bare

Inicjalizacja nowego repozytorium surowego w podanym folderze:

git init --bare scieżka/do/folderu

**Klonowanie repozytorium**

Klonowanie repozytorium do folderu o nazwie identycznej jak repozytorium:

git clone adres-repozytorium

Klonowanie repozytorium do bieżącego folderu:

git clone adres-repozytorium .

Klonowanie repozytorium do dowolnego folderu:

git clone adres-repozytorium ścieżka/do/folderu

**Informacje o repozytorium**

Lista uczestników projektu:

git shortlog -s -n

Lista uczestników projektu:

git shortlog -s -n | wc -l

Liczba plików w repozytorium

find . -type f -print | grep -v -E ‘/\.git/’ | wc -l

Liczba rewizji zawartych w repozytorium:

git log --pretty=oneline | wc -l

**Historia projektu**

Sprawdzanie historii projektu:

git log

Skrócona historia projektu:

git log --pretty=oneline

git log --oneline

Skrócona historia zawierająca zminimalizowane identyfikatory SHA-1

git log --abbrev-commit --abbrev=4 --pretty=oneline

Historia zawierająca trzy ostatnie rewizje:

git log -3

Historia zawierająca rewizje podanego użytkownika:

git log --author=użytkownik

Historia zawierająca rewizję z podanego okresu:

git log --since=”2012-01-01” --until=”2012-01-15”

Formatowanie wydruku historii według dowolnych reguł:

git log --pretty=format:”%h %cd”

Lista ostatnich 10 rewizji w skróconym formacie prezentowana graficznie:

git log --pretty=oneline --abbrev-commit --abbrev=4 -10 --graph

**Interfejs graficzny**

Uruchomienie programu Git GUI:

git gui

Uruchomienie programu do wizualnej prezentacji zawartości repozytorium:

gitk

**Przywracanie obszaru roboczego do wybranej rewizji**

Przywrócenie stanu plików w obszarze roboczym do postaci z ostatniej rewizji (polecenia nie usuwają nowych plików)

git reset --hard

git checkout -f

Przywrócenie stanu plików w obszarze roboczym do postaci z podanej rewizji (rewizje późniejsze są usuwane z historii projektu):

git reset --hard [SHA-1]

Przywrócenie stanu plików w obszarze roboczym do postaci z podanej rewizji(po tej operacji repozytorium znajduję się w stanie detached head W tym stanie nie powiino się tworzyć nowych plików):

git checkout -f [SHA-1]

**Stan repozytorium**

Informacje o stanie repozytorium:

git status

skrócona informacja o stanie repozytorium:

git status -s

**Uproszczony model tworzenia rewizji**

Tworzenie rewizji zawierającej bieżący stan wszystkich plików(pierwsza wersja):

git add -A

git commit -m „komunikat …”

Tworzenie rewizji zawierającej bieżący stan wszystkich plików(druga wersja):

git add .

git commit -a -m „komunikat …”

**Gałęzie**

Sprawdzanie listy wszystkich gałęzi oraz gałęzi bieżącej:

git branch

Przejście na wybraną gałąź:

git checkout [nazwagałęzi]

np.

git checkout master

git checkout dev

**Usuwanie folderów .git**

Usunięcie wszystkich folderów .git zawartych w folderze bieżącym i jego podfolderach:

find . -name .git -type d -exec rm -fr {} \;

**Tworzenie rewizji(zatwierdzanie zmian)**

Po wydaniu komendy:

git commit

uruchamiany jest edytor tekstu, w którym należy wprowadzić opis rewizji(domyślnie vi).

Opis rewizji możemy podać jako wartość parametru -m:

git commit -m „komunikat…”

Parametr -a powoduje automatyczną indeksację(przed wykonaniem rewizji) plików zmodyfikowanych oraz usuniętych:

git commit -a

**Zmiana stanu pliku**

Indeksacja pliku:

git add [nazwa pliku]

Indeksacja wszystkich nowych plików:

git add .

Indeksacja wszystkich plików(nowych, zmodyfikowanych, usuniętych); równoważne wersje:

git add --all .

git add --adl

git add -A .

git add -A

Indeksacja plików zmodyfikowanych I usuniętych:

git add --updated .

git add -u .

Usunięcie pliku:

git rm [nazwa-pliku]

Indeksacja usuwania pliku(plik pozostanie w obszarze roboczym):

git rm --cached [nazwa-pliku]

Usunięcie pliku, nawet jeśli nie jest on aktualny:

git rm -f nazwa-pliku

Zmiana nazwy pliku:

git mv stara-nazwa nowa-nazwa

**Tworzenie plików**

echo a > a.txt

echo „Lorem …..” > lorem.txt

**Składnia poleceń Gita**

Specjalny parametr -- oddziela parametry od ścieżek:

git add -- --all

Powyższe polecenie indeksuje plik o nazwie --all.

**Ignorowanie plików**

Plik konfiguracyjny podlegający zapisowi w repozytorium:

.gitignore

Plik konfiguracyjny niepodlegający zapisywaniu w repozytorium:

.git/info/exclude

Globalny plik .gitignore (niezapisywany w repozytoriach):

ścieżka\do\dowolnego\folderu\.gitingnore

Ustalenie nazwy globalnego pliku .gitconfig:

git config --global core.excudesfile ścieżka\do\dowolnego\folderu\.gitignore

Powyższa instrukcja tworzy wpis w globalnym pliku konfiguracyjnym:

[core]

excludesfile = scieżka\\do\\dowolnego\\folderu\\.gitignore

Przykładowa zawartość pliku .gitignore

#foldery

.idea

nbpoject

tmp

#pliki

\*.[oa]

\*~

\*.exe

**Znaczniki**

Tworzenie znacznika opisanego dotyczącego ostatniej rewizji:

git tag -a NAZWA -m KOMENTARZ

np.

git tag -a v1.2.3 -m „Wydanie wersji 1.2.3”

Tworzenie znacznika opisanego dotyczącego dowolnej rewizji:

git tag -a NAZWA -m KOMENTARZ SHA-1

np.

git tag -a v4.5.6 -m „Wydanie ver. 4.5.6” aabbccdd

Tworzenie znacznika lekkiego dotyczącego ostatniej rewizji:

git tag NAZWA

Tworzenie znacznika lekkiego dotyczącego dowolnej rewizji:

git tag NAZWA SHA-1

Usuwanie dowolnego znacznika:

git tag -d NAZWA

Sprawdzenie dostępnych znaczników:

git tag

Sprawdzenie dostępnych znaczników wraz z datami utworzenia:

git log --tags --simplify-by-deoration --pretty=”format:%ai %d”

Posortowana lista znaczników wraz z datami utworzenia:

git log --tags --simplify-by-decoration --pretty=”format:%ai %d” | sort

Lista dostępnych znaczików opisanych:

git describe

Lista wszystkich dostępnych znaczników:

git describe --tags

Wyświetlanie szczegółowych danych znacznika:

git show -s v2.0.10

Zapisywanie rewizji oznaczonej znacznikiem v0.1.2 do skompresowanego archiwum:

git archive --format=zip --output=../project-0.1.2.zip v0.1.2

git archive --format=zip v0.1.2 > ../project-0.1.2.zip

**Dziennik reflog**

Przeglądanie całego dziennika reflog:

git reflog

Przeglądanie dziennika reflog gałęzi:

git reflog nazwa-gałęzi:

np.

git reflog dev

Czyszczenie dziennika reflog:

git reflog expire --all --expire=now

**Identyfikator rewizji**

Ustalenie pełnego SHA-1 rewizji:

git rev-parse identyfikator

np.:

git rev-parse v0.1.2

git rev-parse HEAD

git rev-parse master~2

git rev-parse [abc@{3.day.ago}](mailto:abc@%7b3.day.ago%7d)

Lista rewizji osiągalnych z podanej rewizji:

git rev-list id

np.

git rev-list HEAD

**Modyfikowanie historii projektu**

Usuwanie ostatnich rewizji :

git reset --hard X

Modyfikowanie ostatniej rewizji :

git commit --amend -a -m „…”

Łączenie rewizji:

git rebase -I X

Usuwanie zmian wprowadzonych przez rewizję:

gir revert --no-edit X

Odzyskiwanie pliku z wybranej rewizji:

git checkout -f X nazwa-pliku

**Operowanie Gałęziami**

Sprawdzanie bieżącej gałęzi

git branch

git status -sb

Tworzenie gałęzi wskazującej na bieżącą rewizję:

git branch nazwa

Tworzenie gałęzi wskazującej na dowolną rewizję:

git branch nazwa SHA-1

przełączanie gałęzi:

git checkout nazwa

Tworzenie i przełączanie gałęzi(nowa gałąź wskazuje bieżącą rewizję):

git checkout -b nazwa

Tworzenie i przełączanie gałęzi(nowa gałąź wskazuje dowolną rewizję):

git checkout -b nazwa SHA-1

Przełączanie gałęzi w repozytorium surowym:

git symbolic-ref HEAD refs/heads/nazwa-gałęzi/

Lista gałęzi zawartych w bieżącej gałęzi:

git branch --merged

Lista gałęzi rozłączonych z bieżącą gałęzią:

git branch --no-merged

Lista gałęzi zawartych w dowolnej gałęzi:

git branch --merged SHA-1

Lista gałęzi rozłącznych z dowolną gałęzią:

git branch --no-merged SHA-1

Usuwanie gałęzi zawartych w dowolnej gałęzi występującej w repozytorium:

git branch -d nazwa-gałęzi

Usuwanie gałęzi rozłącznych:

git branch -D nazwa-gałęzi

Zmiana nazwy gałęzi:

git branch -m stara-nazwa nowa-nazwa

Zmiana nazwy gałęzi z wymuszeniem usunięcia istniejącej gałęzi o podanej nazwie:

git branch -M stara-nazwa nowa-nazwa

Sprawdzanie różnic pomiędzy gałęziami:

git diff --name-status gałąź-a..gałąź-b

**Optymalizacja repozytorium**

Parametry usuwania zgubionych rewizji:

git config gc.pruneexpire now

git config gc.reflogexpireunreachable now

Usuwanie zgubionych rewizji:

git prune

Optymalizacja repozytorium:

git gc

**Łączenie gałęzi(merge)**

Łączenie gałęzi :

git merge nazwa-gałęzi

Łączenie wielu gałęzi:

git merge a b c

Usuwanie łączenia gałęzi wykonanego operacją git merge:

git reset --hard HEAD~

**Łączenie gałęzi(rebase)**

Łączenie gałęzi

git rebase nazwa-gałęzi

Usuwanie łączenia gałęzi wykonanego operacją git rebase(w oparciu o dziennik reflog):

git reflog

git reset --hard HEAD@{n}

Anulowanie nieudanej operacji git rebase:

git rebase --abort

**Ręczne klonowanie repozytorium**

git init

git remote and origin adres-klonowanego-repozytorium

git fetch --no-tags origin master:refs/remotes/origin/master

git branch --set-upstream master origin/master

git reset --hard HEAD

**Repozytoria zdalne**

Definiowanie adresu repozytorium zdalnego:

git remote add nazwa adres

Listowanie adresów repozytoriów zdalnych:

git remote -v

Usuwanie adresu repozytorium zdalnego:

git remote rm nazwa

**Gałęzie śledzone**

Sprawdzanie gałęzi śledzonych dla gałęzi X:

git config --get branch.X.remote

git config --get branch.X.merge

Ustalanie gałęzi śledzonej:

git branch --set-upstream galaz-lokalna repozytorium-zdalne/gałąź-zdalna

Równoważny sposób definiowania gałęzi śledzonej master(należy wydać oba polecenia):

git config branch.master.remote origin

git config branch.master.merge refs/heads/master

**Listowanie gałęzi**

Gałęzie lokalne:

git branch

Gałęzie zdalne:

git branch -r

Wszystkie gałęzie:

git branch -a

**Synchronizowanie gałęzi**

Pobieranie gałęzi zdalnej

git fetch nazwa-repozytorium-zdalnego nazwa-gałęzi-zdalnej

np.

git fetch origin master

Pobieranie gałęzi śledzonej dla gałęzi bieżącej:

git fetch

Pobieranie i łączenie gałęzi zdalnej:

git merge repozytorium-zdalne/gałąź-zdalna

np.

git merge origin/master

Wykonywanie operacji git merge dla gałęzi śledzonej

git merge

Pobieranie gałęzi zdalnej, łącznie z gałęzią bieżącą i przywracanie obszaru roboczego:

git pull nazwa-repozytorium-zdalnego nazwa-galezi-zdalnej

Wykonywanie operacji git pull dla gałęzi śledzonej:

git pull

Przesyłanie zawartości bieżącej gałęzi do repozytorium zdalnego:

git push nazwa-repozytorium-zdalnego nazwa-gałęzi-zdalnej

np.

git push my abc

Przesyłanie gałęzi ze zmianą nazwy:

git push nazwa-repozytorium-zdalnego nazwa-gałęzi-lokalnej:nazwa-gałęzi-zdalnej

np.

git push my lokalna-lorem:zdalna-ipsum

Przesyłanie gałęzi bieżącej do repozytorium zdalnego(wydane w gałęzi xyz prześle gałąź xyz):

git push nazwa-repozytorium-zdalnego HEAD

Przesyłanie gałęzi bieżącej do gałęzi śledzonej:

git push

Przesyłanie gałęzi bieżącej i wiązanie gałęzi zdalnej z lokalną:

git push -u nazwa-repozytorium-zdalnego nazwa gałęzi-zdalnej

Przesyłanie gałęzi lokalnej z nadpisaniem historii:

git push -f

git push -f nazwa-repozytorium-zdalnego nazwa-gałęzi-zdalnej

git push -f nazwa-repozytorium-zdalnego nazwa-gałęzi-lokalnej:nazwa-gałęzi-zdalnej

Przełączanie na gałąź zdalną:

git checkout nazwa-gałęzi-zdalnej

Usuwanie gałęzi zdalnej:

git push repozytorium-zdalne :nazwa-gałęzi-zdalnej

**Zabezpieczanie repozytorium**

Zabezpieczanie przed utratą rewizji

git config receive.denyDeletes true

git config receive.denyNonFastForwards true