

M05 PRACA ZDALNA

L05 Wysyłanie zmian

(Tak się rodzą Wielkie Rzeczy ;))

Maciej Aniserowicz

Aktualnie...

nasze "bare" repo nie posiada niczego: ani commita, ani gałęzi

Bare demo



[serwer repo]

1. git branch
2. gitk --all

Nic, pusto! Trzeba je uzupełnić...

Push

Push, baby, push!

Wysłanie zmian to komenda "push" - **wypchnięcie** commitów

Push



[repo 1]

1. git push

The current branch master has no upstream branch.

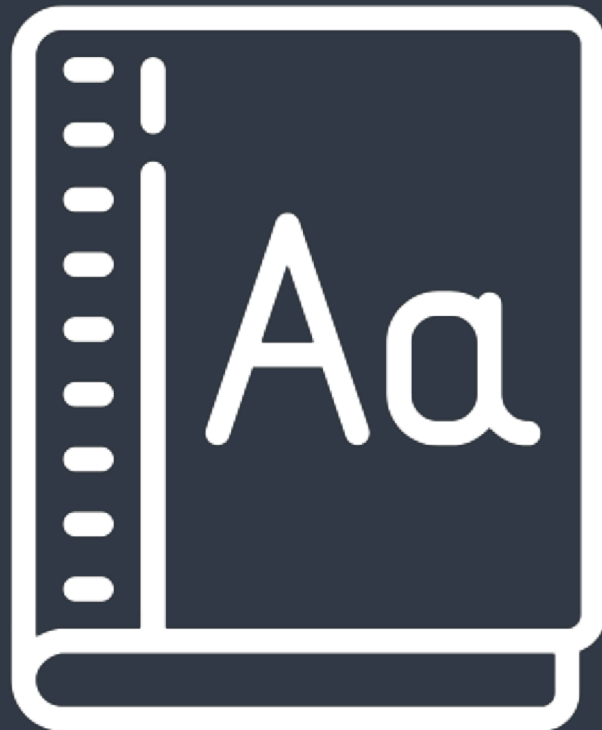
Upstream branch

Odpowiednik lokalnej gałęzi w zdalnym repozytorium.

Git śledzi zmiany w jednej i drugiej, wskazując zmiany między nimi.

Operacje push/pull domyślnie wiedzą, której zdalnej gałęzi domyślnie użyć.

Nazwy mogą być różne! Ważne jest mapowanie w konfiguracji Gita.



Push push **jeszcze raz!**

- 
1. `git push origin master -u`


"new branch" & "master set up to track remote branch"

Od teraz: *"git push"* bez tłumaczenia dokąd zmiany mają być wysłane.

2. `git remote show origin:`

"master pushes to master"

Dość tych **masterów!** Inne nazwy gałęzi

- 
1. `git checkout -b my_experiment_branch`
 2. `git push origin -u my_experiment_branch:aniserowicz_#44`
 3. `git branch -a`

Przydatne gdy gałęzie wspólnym repo są traktowane jako "work in progress".

"Zdalne" gałęzie mogą zawierać np. ID realizowanego zadania i nazwisko.

Dwukropek

Oddziela lokalną nazwę od nazwy "zdalnej":

my_experiment_branch:aniserowicz_#44

[lokalna] : [zdalna]

Skasowanie zdalnej gałęzi

To także **push**!

Z dwukropkowym rebusem:

Pchnij NIC do gałęzi <x>

Skasowanie zdalnej gałęzi

- 
1. `git push origin :aniserowicz_#44`

Alternatywnie (mniej lannersko)

1. `git push origin --delete aniserowicz_#44`

BEST PRACTICE

Zastanów się 3x **przed**
skasowaniem zdalnej gałęzi!

Czy **na pewno** nie jest
potrzebna/używana?



Backup

Backup?

Czy nasz "push" wykonał pełny "backup"?

Zobaczmy...

Backup?



[bare repo]


1. `gitk --all`

Nie ma tagów!

Nie byłoby też gałęzi innych niż "*master*"!

To **żaden backup...**

Backup-repo

- 
1. `mkdir backup_repo.git`
 2. `cd backup_repo.git`
 3. `git init --bare`

Na przykład leżące na Dropboxie, jak już wspominaliśmy.

Link do backup-repo




[repo 1]

1. `git remote add backup ../backup_repo.git`
2. `git remote`

mamy 2 remotes!

jedno "*origin*", drugie "*backup*"

Jak WYKONAĆ backup?

- 
1. `git branch feature-branch`

nowa gałąź (w praktyce będziesz ich mieć wiele)

2. `git push backup --mirror`

wysyła WSZYSTKO

Uwaga: to nie jest zastępstwo "prawdziwego" backupu! Bardziej na potrzeby jednego programisty.

Usługi oferujące hosting Gita muszą zadbać o coś bardziej "pro".

Pushing "modes"

Aktualne zachowanie to tzw. *"simple pushing mode"*


Simple mode robi *"push"* do *upstream branch*.

(polski język piękna język ;))

Można zmienić to zachowanie: poczytaj o konfiguracji `push.default`

(na tym etapie nie radzę)

Remote rename

- 
1. `git remote rename backup bak`
 2. `git branch -a`

// Side note

Dlatego tak kocham Gita: można się bawić do woli bez obaw, że się cokolwiek zepsuje.

(prawie) Wszystko jest odwracalne.

Podsumowanie

- git push
- git push -u
- git push --delete
 - git push <remote> :<remote_branch>
- git remote rename
- git push --mirror



ZADANIE

- ✓ wyślij zmiany do utworzonego wcześniej "bare repo", konfigurując jednocześnie "upstream branch"
- ✓ stwórz kolejne "bare repository" i dodaj je jako remote "backup"
- ✓ wyślij wszystkie zmiany do remote "backup"
- ✓ zmień nazwę tego remote

