



Kurs Front-End **Developer**
Systemy kontroli wersji - GIT

SYSTEMY KONTROLI WERSJI

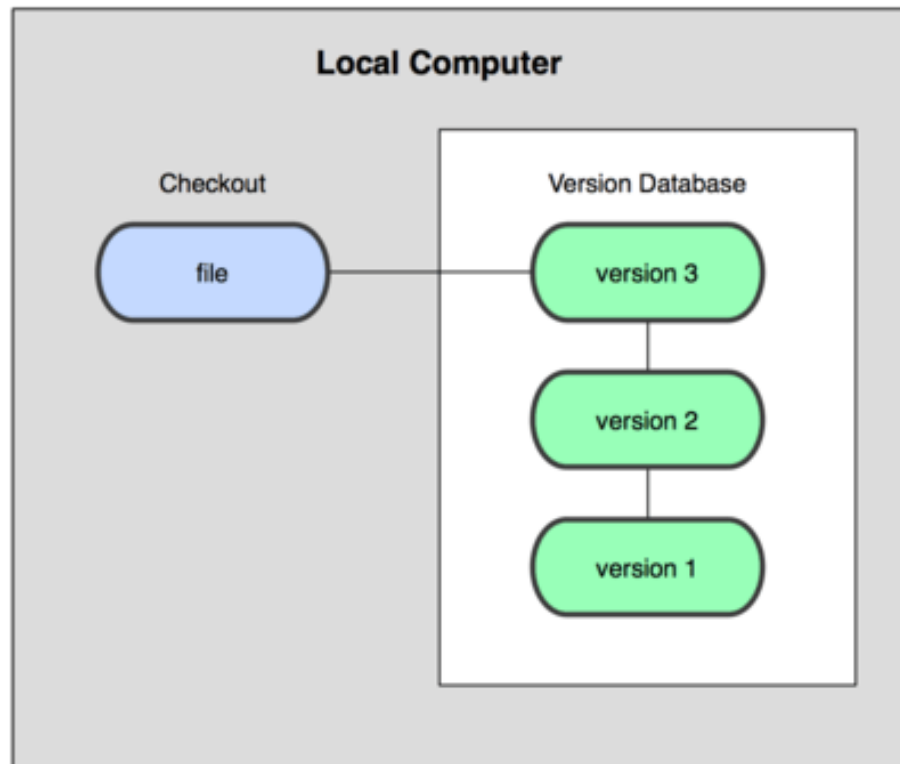
Systemy kontroli wersji to narzędzie pozwalające na zapisywanie i śledzenie zmian w kodzie projektu.

W każdym momencie programista może mieć wgląd w historię swojej pracy i w razie potrzeby do niej powrócić.

Ułatwiają także pracę nad projektem w kilkuosobowych zespołach

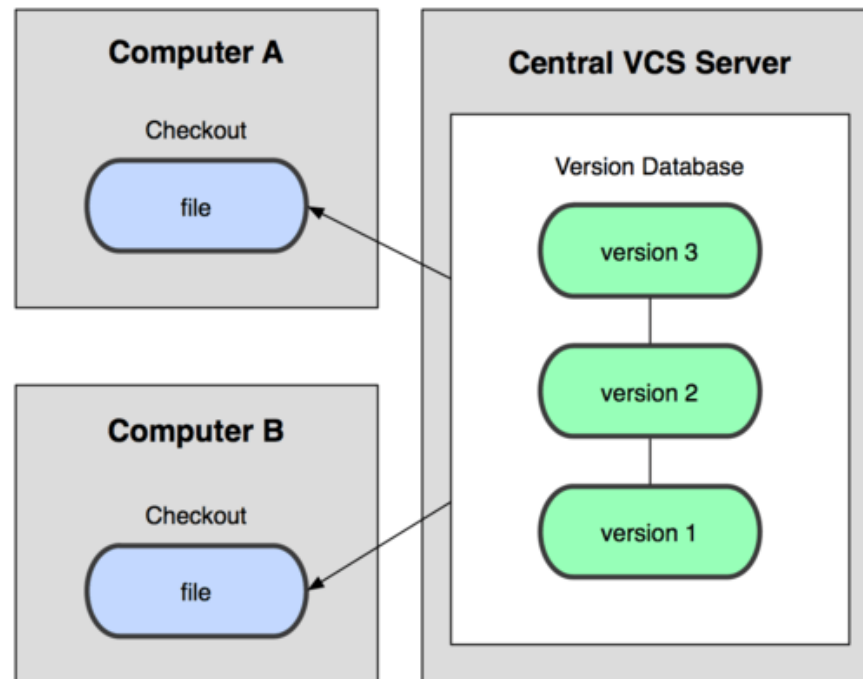
LOKALNE SYSTEMY KONTROLI WERSJI

działające na jednej fizycznej maszynie



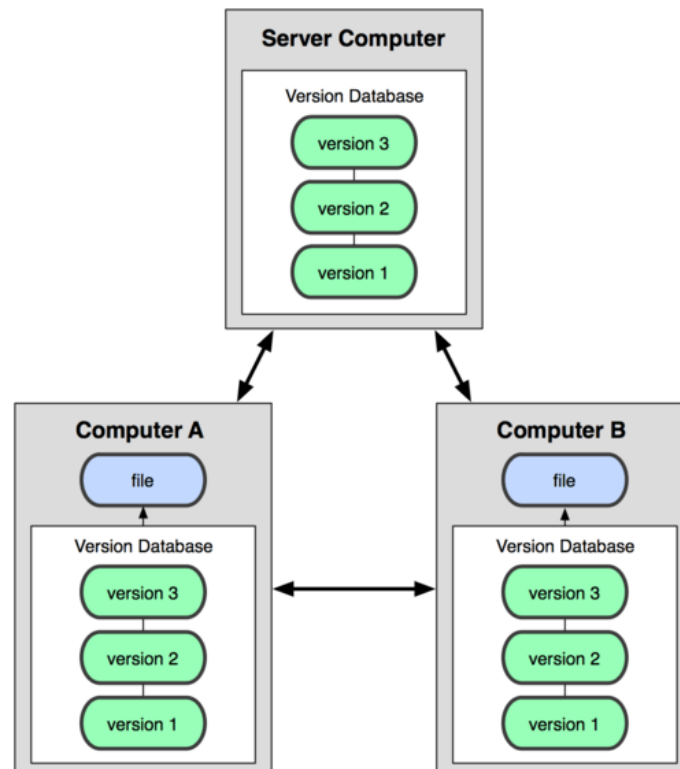
SCENTRALIZOWANE SYSTEMY KONTROLI WERSJI

jeden serwer zawierający wszystkie pliki projektu oraz klienci, którzy mogą pobierać ich najnowszą wersję



ROZPROSZONE SYSTEMY KONTROLI WERSJI

Repozytoria przechowywane lokalnie i na serwerze



DLACZEGO GIT?



- A. Git jest obecnie najpopularniejszym systemem kontroli wersji.
- B. Istnieje kilka darmowych repozytoriów ([GitHub](#), [Bitbucket](#)).
- C. Łatwa instalacja i konfiguracja na różnych platformach

INSTALACJA GITA

Jeśli posiadasz komputer z zainstalowanym systemem Windows
pobierz plik instalacyjny

<https://github.com/git-for-windows/git/releases/>

Pobieramy aktualną, stabilną wersję Git for Windows

INSTALACJA GITA

Jeśli posiadasz komputer z systemem OSX to pobierz plik instalacyjny

<https://sourceforge.net/projects/git-osx-installer/>

Jeśli posiadasz komputer z zainstalowanym systemem Linux (dla dystrybucji opartych na Debianie - np. Ubuntu)

Wpisz w konsoli komendę

apt-get install git

KONFIGURACJA GITA

W celu korzystania z GITA niezbędna jest jego podstawowa konfiguracja, która pozwala identyfikować użytkownika, który pracuje z danym repozytorium.

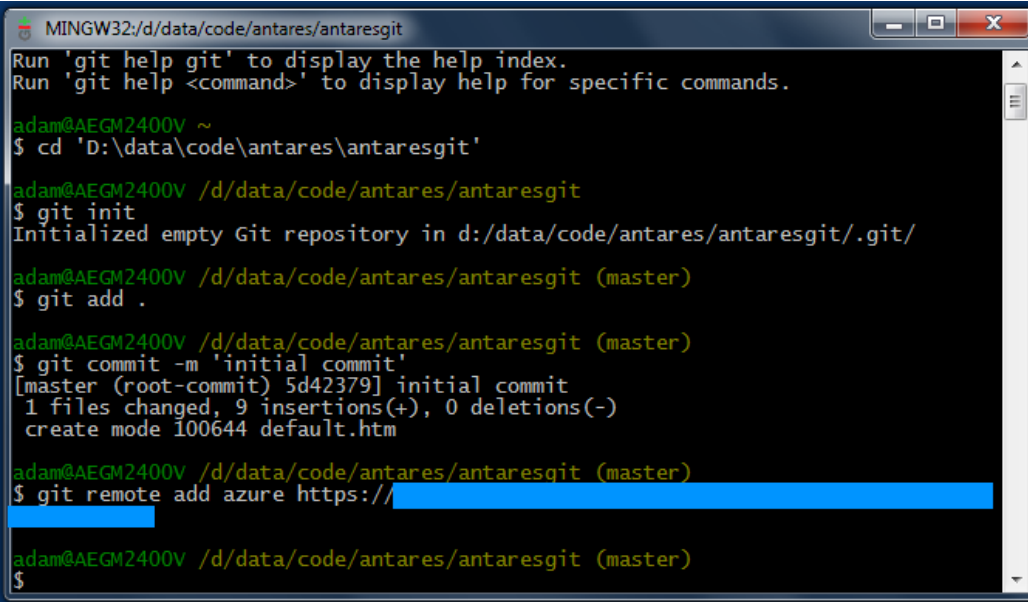
Wszystkie działania konfiguracyjne prowadzimy poprzez wpisywanie odpowiednich komend w konsoli.

Jeśli mamy już zainstalowanego GITA sprawdzamy konfigurację komendą

```
git config --list
```

KONFIGURACJA GITA

Jeśli jesteś użytkownikiem Windowsa, to po zainstalowaniu GIta będziesz miał dostęp do programu Git Bash, który będzie dostępny w Menu Start w folderze Git.



```
MINGW32:/d/data/code/antares/antaresgit
Run 'git help git' to display the help index.
Run 'git help <command>' to display help for specific commands.

adam@AEGM2400V ~
$ cd 'D:\data\code\antares\antaresgit'

adam@AEGM2400V /d/data/code/antares/antaresgit
$ git init
Initialized empty Git repository in d:/data/code/antares/antaresgit/.git/

adam@AEGM2400V /d/data/code/antares/antaresgit (master)
$ git add .

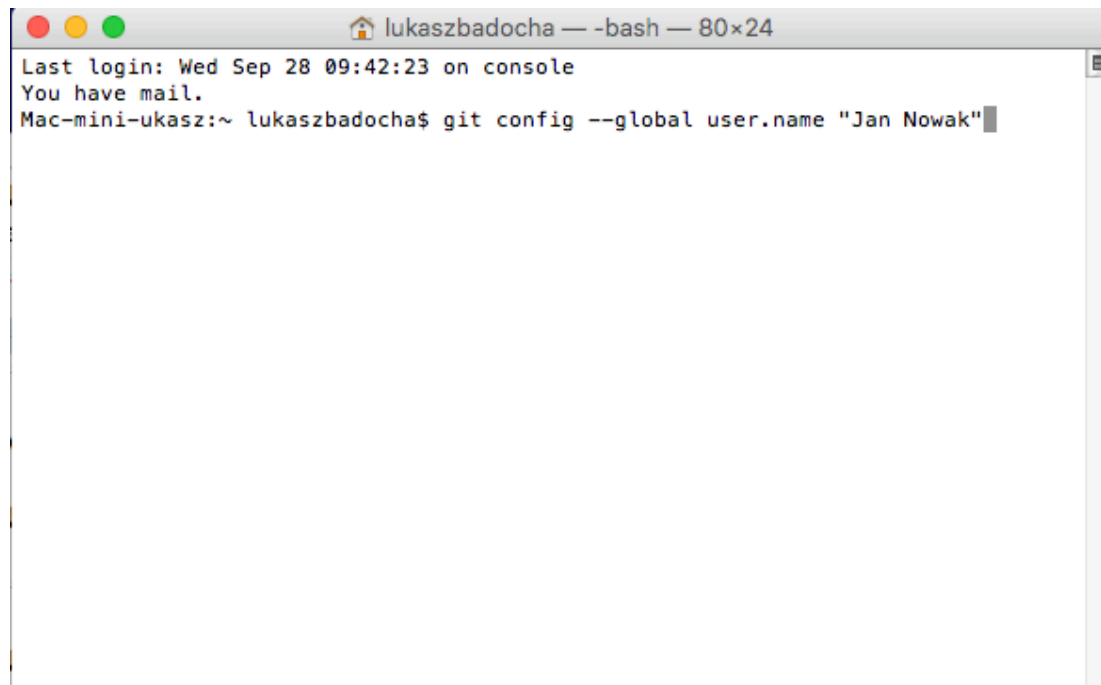
adam@AEGM2400V /d/data/code/antares/antaresgit (master)
$ git commit -m 'initial commit'
[master (root-commit) 5d42379] initial commit
1 files changed, 9 insertions(+), 0 deletions(-)
create mode 100644 default.htm

adam@AEGM2400V /d/data/code/antares/antaresgit (master)
$ git remote add azure https://[redacted]

adam@AEGM2400V /d/data/code/antares/antaresgit (master)
$
```

KONFIGURACJA GITA

Użytkownicy systemów Unixowych (OSX, Linux) mają domyślnie zainstalowany terminal w swoich systemach operacyjnych

A screenshot of a macOS terminal window. The title bar shows the user 'lukaszbadocha' and the shell '-bash' with a window size of '80x24'. The terminal output shows a successful login on September 28th, a notification about mail, and the execution of the command 'git config --global user.name "Jan Nowak"'.

```
lukaszbadocha — -bash — 80x24
Last login: Wed Sep 28 09:42:23 on console
You have mail.
Mac-mini-ukasz:~ lukaszbadocha$ git config --global user.name "Jan Nowak"
```

KONFIGURACJA GITA

Na początek musimy skonfigurować użytkownika GITA. Robimy to wpisując w terminalu następujące komendy. **Każdą komendę zatwierdzamy przyciskiem Enter.**

```
git config --global user.name "Jan Nowak"  
git config --global user.email jannowak@example.com
```

W miejsce imienia i nazwiska oraz adresu wpisujemy własne dane.

Należy również skonfigurować domyślny edytor dla GITA

```
git config --global core.editor brackets
```

Na tym kończymy podstawową konfigurację. Nie zamykaj konsoli, ponieważ w dalszej części wprowadzenia będziemy z niej korzystać.

PODSTAWOWE KOMENDY W KONSOLI

Aby poruszać się w strukturze katalogów poprzez użycie konsoli powinieneś znać podstawowe komendy

`cd nazwa_folderu` – Wejdź do folderu o danej nazwie

`cd ..` – Wróć do folderu nadrzędnego

`ls` – pokaż zawartość folderu

`ls -l` – listuje zawartość folderu z dodatkowymi opcjami

`ls -a` – pokazuje ukryte pliki

`cd ~` - Wróć do folderu głównego (root)

`mkdir` – tworzy nowy katalog

`touch nazwa_pliku` – tworzy nowy plik

PODSTAWOWE KOMENDY W KONSOLI

Aby poruszać się w strukturze katalogów poprzez użycie konsoli powinienś znać podstawowe komendy

rm -r nazwa_folderu – Usun folder

rm nazwa_pliku – Usun plik o danej nazwie

pwd – pokaż ścieżkę do aktualnego folderu

cp ścieżka_do_pliku/folderu ścieżka_do_folderu – Kopiuj plik/folder do wskazanego folderu

mv ścieżka_do_pliku/folderu ścieżka_do_folderu – Przenieś plik/folder do wskazanego folderu – służy również do zmiany nazwy pliku lub folderu

WARSZTATY ☺ KOMEND W KONSOLI

Zadanie z konsoli / termiala

1. Przejdź do pulpitu
2. Utwórz na pulpicie folder “konsola” (**mkdir**)
3. W folderze “konsola” stwórz dwa foldery “pierwszy” i “drugi”
4. Przejdź do folderu “pierwszy” i stwórz w nim plik index.html (**touch**)
5. Sprawdź zawartość folderu (**ls**)
6. Skopiuj plik z folderu “pierwszy” do folderu “drugi” (**cp**)
7. Przejdź do folderu drugi i sprawdź, czy jest tam skopiowany plik
8. Usuń plik z folderu “pierwszy” (**rm**)
9. Usuń cały folder “pierwszy” (**rm -r**)
10. Zmień nazwę folderu z “drugi” na “pierwszy” (**mv**)

UŻYWAMY REPOZYTORIUM GIT 😊

- logujemy się / zakładamy konto na konto na [GitHub](#)
- tworzymy nowe repozytorium - [New Repository](#)
- Nadajemy nazwę [username.github.io](#) oraz dodajemy opis ([username](#) to Twoja nazwa użytkownika GitHub)

UŻYWAMY REPOZYTORIUM GIT 😊

- Otwieramy konsolę i na pulpicie tworzymy folder o nazwie **repo-git**
- Przechodzimy do folderu repo-git (**cd repo-git**)
- klonujemy repozytorium - git clone <https://github.com/username/username.github.io.git> (username to Twoja nazwa użytkownika GitHub)
- wchodzimy do folderu **cd nazwa folderu**

UŻYWAMY REPOZYTORIUM GIT 😊

- otwieramy folder z repozytorium w [Brackets](#)
- tworzymy plik [index.html](#)
- dodajemy treść - tworzymy dokument HTML z jednym paragrafem

UŻYWAMY REPOZYTORIUM GIT 😊

WAŻNE:

PRZED commitem **ZAWSZE DODAJEMY PLIKI DO ŚLEDZENIA** np. `git add *`

- Sprawdzamy czy index.html jest w folderze – komenda `ls`
- Sprawdzamy status repozytorium – komenda `git status`
- Dodajemy plik do śledzenia – komenda `git add *`
- Ponownie sprawdzamy status repozytorium – komenda `git status`

UŻYWAMY REPOZYTORIUM GIT 😊

- robimy pierwszy commit - `git commit -m 'init'`
- Wysyłamy pliki do zdalnego repozytorium – komenda `git push`
- Modyfikujemy plik dodając drugi paragraf
- Sprawdzamy różnicę od ostatniego commita – komenda `git diff`

UŻYWAMY REPOZYTORIUM GIT 😊

- Podmieniamy URL zdalnego repozytorium z HTTPS na SSH

```
git remote set-url origin git@github.com:<Username>/<Project>.git
```

- Robimy push do repozytorium już bez hasła

```
git push origin master
```

UŻYWAMY REPOZYTORIUM GIT 😊

- Tworzymy nową gałąź - `git checkout -b newbranch`
- Wysyłamy nową gałąź do zdalnego repozytorium – komenda `git push origin newbranch`
- Dodajemy index.html do śledzenia w nowej gałęzi - `git add *` i robimy commit
- Porównujemy obie gałędzie – komenda `git diff master..newbranch`

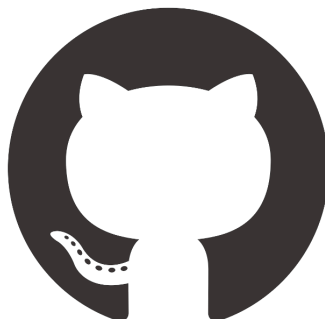
UŻYWAMY REPOZYTORIUM GIT 😊

- Przełączamy się do głównej gałęzi - **git checkout master**
- w pliku index.html znikną zmiany dodane po commit
- Modyfikujemy plik dodając nowy paragraf (pod pierwszym paragrafem), dodajemy plik do śledzenia (**git add ***) i robimy commit
- Porównujemy obie gałędzie – komenda **git diff master..newbranch**
- Łączymy pliki z gałęzi – komenda **git merge master newbranch**

UŻYWAMY REPOZYTORIUM GIT 😊

- Pojawia się konflikt, który ręcznie usuwamy
- Ponownie dokonujemy commita poprawionego pliku
- `git commit -m "merged master with newbranch"`
- Wysyłamy pliki do repozytorium zdalnego – `git push`
- Podgląd projektu możesz zobaczyć na GitHubPages
<http://username.github.io>

GITHUB DESKTOP



Zainstaluj program GitHub Desktop. Posłuży on do wysyłania i pobierania kodu Twoich projektów z GitHuba oraz do śledzenia zmian na poszczególnych etapach pracy - <https://desktop.github.com/>

Dodaj repozytorium do GitHub Desktop – File-> New Repository -> Add – wskaż katalog z Twoim projektem

WARSZTATY ☺ Git Hub Pages

GitHub Pages

1. Stwórz swoje konto **GitHub Pages**
2. W folderze głównym repozytorium GitHub Pages stwórz plik **index.html** o treści **<p>Moje Portfolio</p>** - ćwiczenie robimy już w pełnej strukturze HTML5
3. W folderze głównym repozytorium GitHub Pages stwórz folder **zadania-domowe**
4. W folderze **zadania-domowe** repozytorium GitHub Pages stwórz podfoldery **zadanie-1**, **zadanie-2**, **zadanie-3**
5. W każdym ze stworzonych w punkcie 4. podfolderze stwórz plik **index.html** z treścią **<p>Zadanie \$</p>**, gdzie **\$** to numer kolejnego zadania – ćwiczenie robimy w pełnej strukturze HTML5

WARSZTATY ☺ Git Hub Pages

GitHub Pages – c.d.

6. Zrób **commit**, a potem zrób **push** do GitHub
7. Pobierz link do **Twojego repozytorium** GitHub Pages i zobacz, czy zmiany się wgrały
8. Pobierz **link do Twojej głównej strony na GitHub Pages** i wklej do przeglądarki – sprawdź, czy widzisz ją w przeglądarce
9. Pobierz **linki do twoich stron GitHub Pages w podfolderach z zadaniami** i wklej je do przeglądarki – sprawdź, czy widzisz je w przeglądarce



Akademia 108

<https://akademia108.pl>