

Kurs Front-End Developer  
GIT

# WPROWADZENIE DO GIT

GIT jest systemem kontroli wersji, czyli narzędziem, które pozwala na śledzenie historii zmian w plikach projektu. Dzięki posiadaniu historii mamy dostęp do wersji projektu z każdego etapu jego powstawania. Jest to niezwykle istotne przy zespołowej pracy, ponieważ GIT pozwala na równoległe pisanie kodu projektu i zapobiega wzajemnemu nadpisywaniu plików. Dzięki przechowywaniu plików na zewnętrznym serwerze mamy pewność, że nie utracimy naszego projektu w przypadku awarii naszego komputera. Mamy też dostęp do niego z każdego urządzenia, które ma skonfigurowane środowisko GIT.

# WPROWADZENIE DO GIT

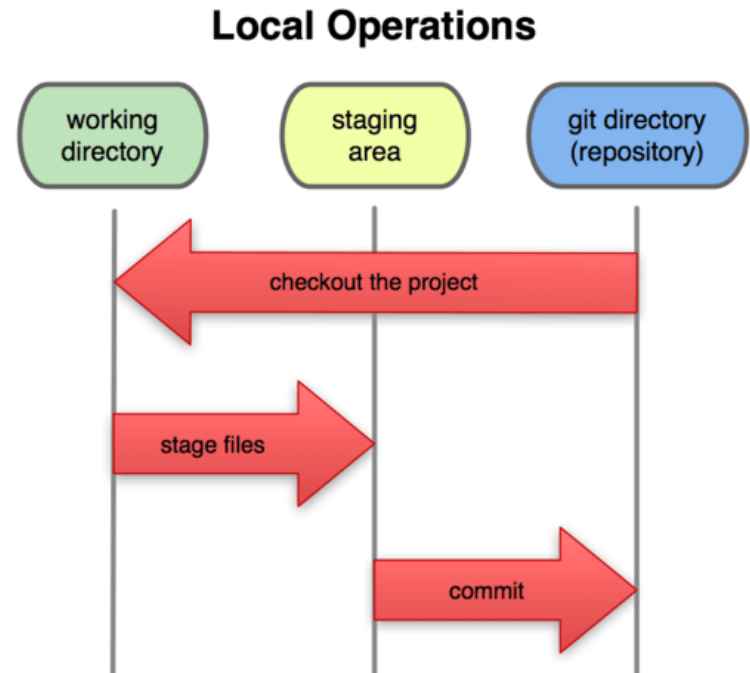
Pracując nad projektem, który zlokalizowany jest w folderze na naszym komputerze możemy stworzyć w nim tak zwane repozytorium GITa, które będzie śledziło zmiany i pozwalało je zapisać.

Tworząc takie repozytorium będziemy mówić o tym, że pliki naszego projektu znajdują się w trzech stanach: śledzony, zmodyfikowany i zatwierdzony.

- Śledzony – dodanie pliku do śledzenie oznacza, że GIT będzie monitorował na bieżąco, czy w pliku są dokonywane zmiany (komenda **git add**)
- Zmodyfikowany – ten stan ma plik, który jest śledzony i zostały w nim dokonane zmiany, ale zmiany nie zostały zatwierdzone i zapisane do bazy danych GIT – taki plik można porównać z zatwierdzoną wersją pliku i zobaczyć jakie są między nimi różnice
- Zatwierdzony - plik i dane w nim zawarte zostały zapisane w bazie danych GIT (komenda **git commit**)

# WPROWADZENIE DO GIT

Mówiąc inaczej, gdy damy polecenie GITowi aby śledził plik, to na bieżąco sprawdza on, czy plik jest modyfikowany. W momencie dokonania zmian w pliku przechodzi on do statusu zmieniony i trafia do przechowalni, w której możemy zobaczyć zmiany jakie zostały dokonane w stosunku do wcześniejszej wersji. Gdy chcemy zatwierdzić wszystkie zmiany, to używając komendy *commit* zapisujemy ją w lokalnej bazie danych GIT.



# INSTALACJA GITA

Jeśli posiadasz komputer z zainstalowanym systemem Windows  
pobierz plik instalacyjny

<https://github.com/git-for-windows/git/releases/download/v2.10.0.windows.1/Git-2.10.0-32-bit.exe> - dla 32-bitowej wersji systemu Windows

<https://github.com/git-for-windows/git/releases/download/v2.10.0.windows.1/Git-2.10.0-64-bit.exe> - dla 64-bitowej wersji systemu 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**

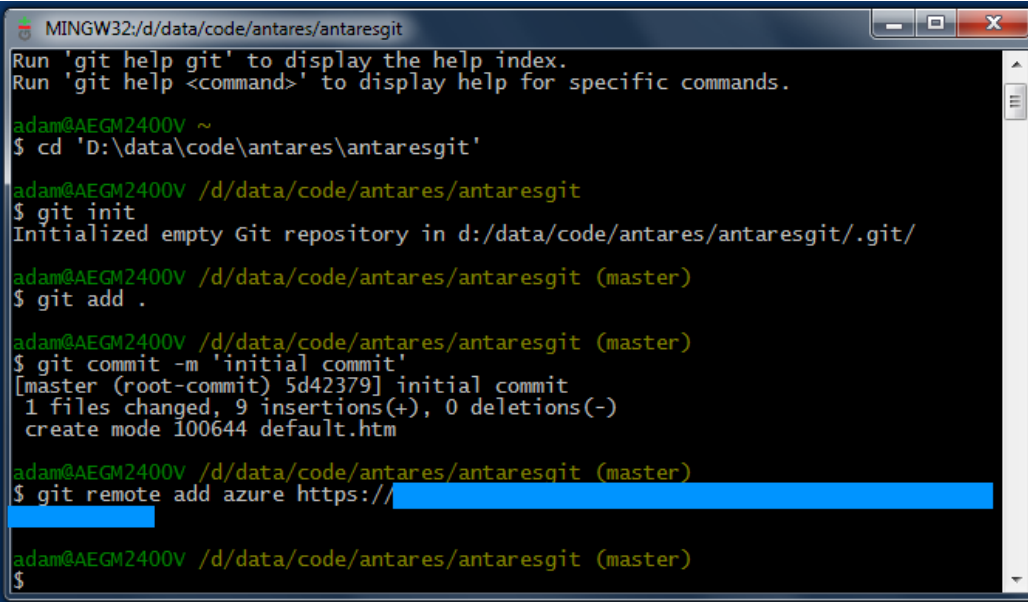
# KONFIGURACJA GITA

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.

# 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

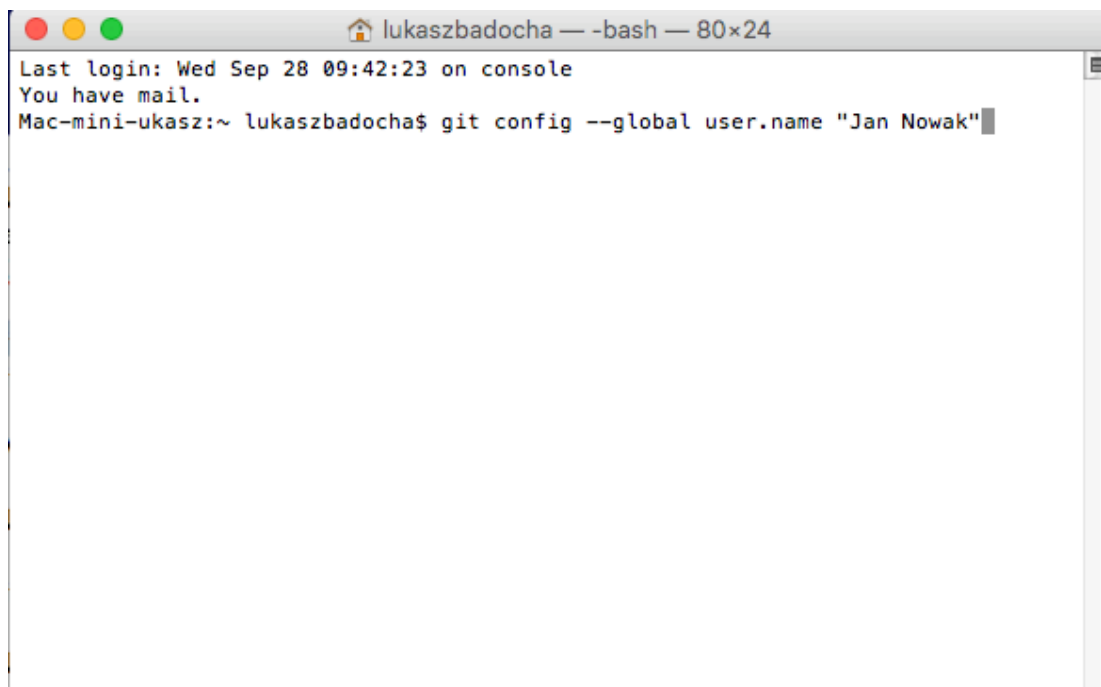
adam@AEGM2400V /d/data/code/antares/antaresgit (master)
$ git remote add azure https://[redacted]

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.

# ZDALNE REPOZYTORIUM GITA

Pliki naszego projektu możemy przechowywać w zdalnym repozytorium. Na potrzeby kursu będziemy korzystać z repozytorium oferowanego przez serwis GitHub <https://github.com/>.

Pierwszym zadaniem będzie stworzenie konta na serwisie.

Szczegółowe informacje na temat działania GitHuba można znaleźć pod tym linkiem <https://guides.github.com/activities/hello-world/>

Na potrzeby wprowadzenia pokażemy w jaki sposób stworzyć nowy projekt na GitHubie, jak stworzyć lokalne repozytorium, jak połączyć lokalne repozytorium ze zdalnym repozytorium na GitHubie oraz jak wysłać stworzone przez nas pliki do zdalnego repozytorium.

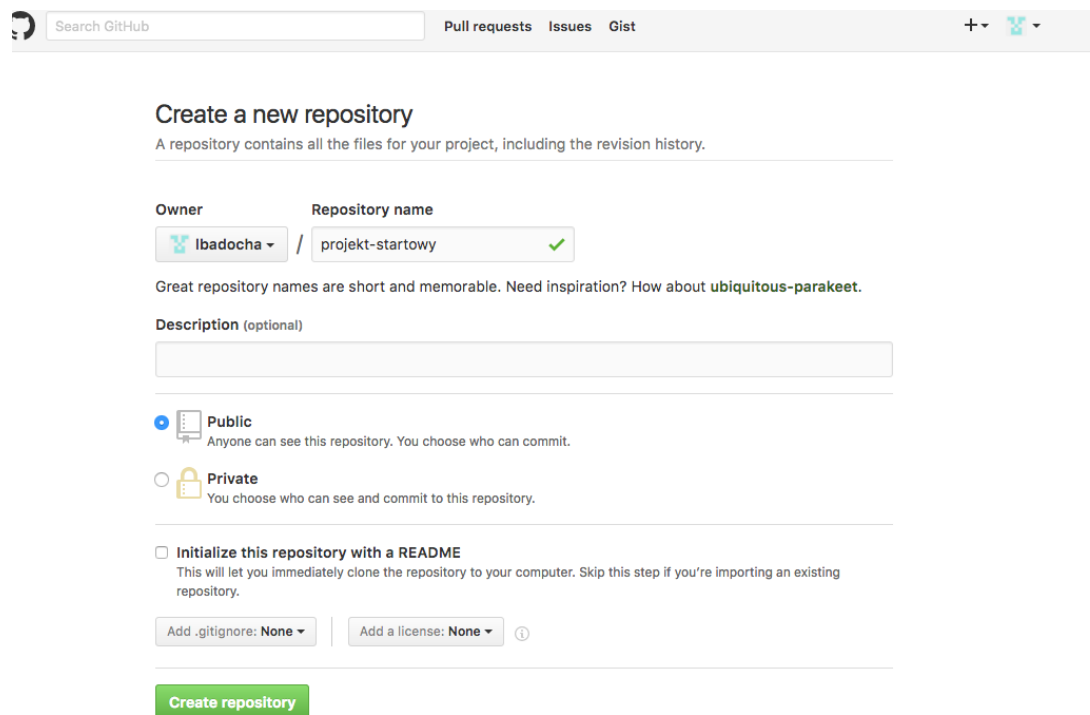
# NOWY PROJEKT GIT

W serwisie GitHub tworzymy nowy projekt, klikając przycisk “Start a project”.



# NOWY PROJEKT GIT

Nadajemy nazwę naszemu projektowi - na przykład “projekt-startowy” i klikamy przycisk “Create repository”



Search GitHub Pull requests Issues Gist + -

## Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: lbadocha / Repository name: projekt-startowy ✓

Great repository names are short and memorable. Need inspiration? How about [ubiquitous-parakeet](#).

Description (optional)

☒ **Public**  
Anyone can see this repository. You choose who can commit.

☐ **Private**  
You choose who can see and commit to this repository.

☐ **Initialize this repository with a README**  
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None** ⓘ

**Create repository**

# NOWY PROJEKT GIT

Następnie tworzymy folder na naszym komputerze. Na potrzeby wprowadzenia utworzymy folder na Pulpicie o nazwie “projekt-startowy”.

W naszej konsoli musimy przejść do odpowiedniego folderu wpisując komendę

```
cd ~/Desktop/projekt-startowy
```

(Aby lepiej zrozumieć działanie podstawowych komend w terminalu zapoznaj się z tym materiałem [https://tutorial.djangogirls.org/pl/intro\\_to\\_command\\_line/](https://tutorial.djangogirls.org/pl/intro_to_command_line/) )

W tym folderze będziemy tworzyli repozytorium lokalne dla naszego projektu startowego.

Otwieramy ten folder w edytorze tekstu Brackets i tworzymy w nim plik index.html

# NOWY PROJEKT GIT

W terminalu wykonujemy następujące polecenia.

**git init** - inicjuje lokalne repozytorium

**git add \*** - dodaje do śledzenia pliki i foldery z naszego katalogu (w naszym przypadku będzie to plik index.html)

**git commit -m 'rozpoczęcie projektu'** - zatwierdzamy wersję plików w lokalnym repozytorium - w nawiasach

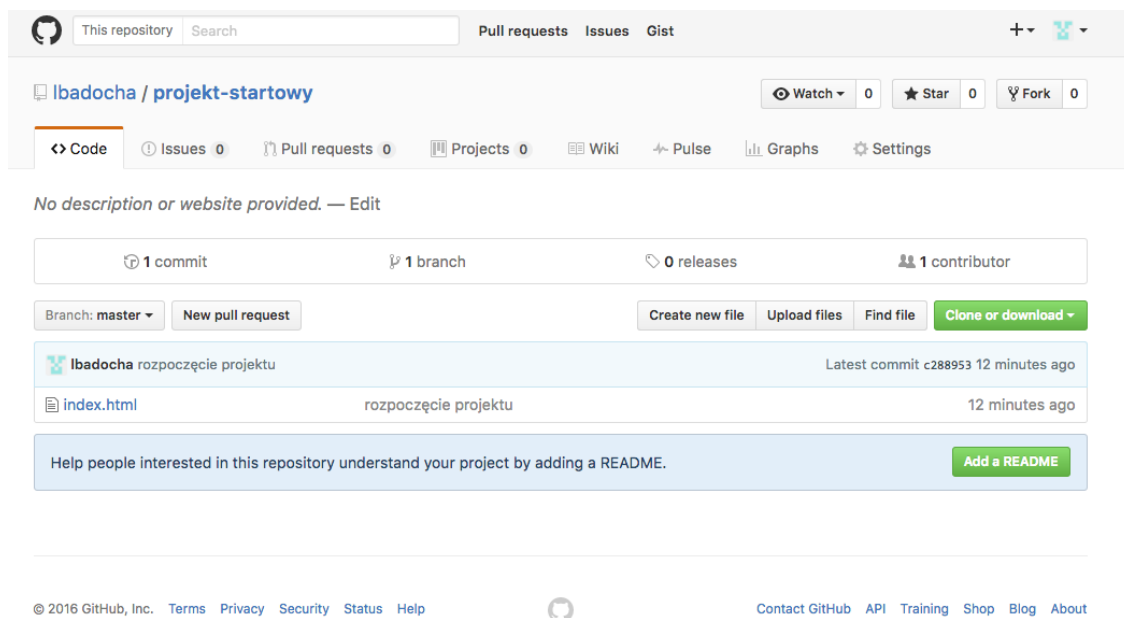
**git remote add origin https://github.com/**

**twoja\_nazwa\_uzytkownika\_github/projekt-startowy.git** - tworzymy powiązanie lokalnego repozytorium z repozytorium zdalnym

**git push -u origin master** - wysyłamy stan lokalnego repozytorium do repozytorium zdalnego

# NOWY PROJEKT GIT

Po wejściu na stronę GitHuba i wybraniu repozytorium projektu powinieneś widzieć, że Twój plik został wysłany





# NOWY PROJEKT GIT

Przeedytuj plik index.html dodając do niego tekst 'test' i zapisz go.

Wpisz w konsoli

**git commit -am 'pierwsza zmiana'** - zatwierdza zmianę w lokalnym repozytorium i dodaje komentarz

**git push** - wysyła zmiany do repozytorium zdalnego

Aktualną wersję pliku oraz historię zmian możesz prześledzić na swoim profilu na GitHub

# PODSTAWY UŻYWANIA GITA

Miałeś już okazję zobaczyć, jak stworzyć repozytorium oraz wysłać pliki do zdalnego repozytorium. Zapoznaj się z podstawowymi komendami GITA i sposobem ich działania

**git config** - konfiguracja ustawień GITA

**git config *opcja*** - pokazuje konfigurację danej opcji (np. git config user.name)

**git config --list** - pokazuje konfigurację wszystkich opcji danego repozytorium

# PODSTAWY UŻYWANIA GITA

**git init** - tworzy nowe repozytorium plików

**git clone źródło katalog** - klonuje repozytorium ze wskazanego źródła do katalogu (np. `git clone https://github.com/twoja_nazwa_uzytkownika_github/projekt-startowy.git` - sklonuje repozytorium ze zdalnego serwera do katalogu w którym się aktualnie znajdujemy)

**git add nazwa\_pliku** - dodanie pliku do śledzenia (komenda `git add *` doda do śledzenia wszystkie nowe pliki w folderze z repozytorium)

# PODSTAWY UŻYWANIA GITA

**git commit** - zapamiętanie wszystkich śledzonych plików w lokalnym repozytorium (opcja -m pozwala dodać komentarz, co jest zalecane, aby wiedzieć jaki element uległ zmianie w danym commicie - np. git commit -m 'dodanie menu strony')

**git status** - wyświetla informacje o aktualnym stanie repozytorium

**git diff** - komenda wyświetla porównanie różnic plików znajdujących się w poczekalni z plikami zatwierdzonymi w lokalnym repozytorium

# PODSTAWY UŻYWANIA GITA

`git rm nazwa_pliku` - usuwa plik z repozytorium

`git mv przenosimy_z przenosimy_do` – komenda służąca do przeniesienia pliku

`git reset HEAD nazwa_pliku` – usuwa niezatwierdzony plik z poczekalni

`git checkout -- nazwa_pliku` - przywraca ostatnią zapamiętaną wersję pliku z lokalnego repozytorium - wszystkie zmiany w pliku zostaną usunięte

# PODSTAWY UŻYWANIA GITA

**git log** - pokazuje wszystkie zatwierdzone zmiany w lokalnym repozytorium

**git revert *nazwa\_commita*** - usuwa wszystkie zmiany, które zostały dokonane po danym commicie i tworzy nowy commit

**git reset *nazwa\_commita*** - przechodzi do wskazanego commita niszcząc wszystkie wcześniejsze - nie zmienia stanu plików, które możemy ponownie commitować

**git remote** - pokazuje nazwy zdalnych serwerów z naszym repozytorium (**git remote -v** pokazuje url zdalnego repozytorium)

# PODSTAWY UŻYWANIA GITA

`git remote add nazwa_zdalnego_repozytorium url_repozytorium` - łączy lokalne repozytorium z repozytorium zdalnym

`git fetch nazwa_zdalnego_repozytorium` - pobranie zdalnego repozytorium

`git pull` - pobranie plików ze zdalnego repozytorium i ich scalenie z aktualnymi plikami w repozytorium lokalnym

`git push nazwa_zdalnego_repozytorium` - wysłanie stanu repozytorium lokalnego do repozytorium zdalnego

# PODSTAWY UŻYWANIA GITA

To tylko część możliwości jakie daje GIT. Zachęcamy do zapoznania się informacjami dostępnymi w oficjalnym poradniku GITa

<https://git-scm.com/book/pl/v1/Pierwsze-kroki>

Na początek zapoznaj się szczególnie z rozdziałami 1. 2. 3. oraz 5.

Pełna dokumentacja GITa dostępna jest pod linkiem

<https://git-scm.com/documentation>





Akademia 108  
ul. Mostowa 6/13  
31-061 Kraków