



**SMART**FACTOR

## BootCamp: zajęcia 3

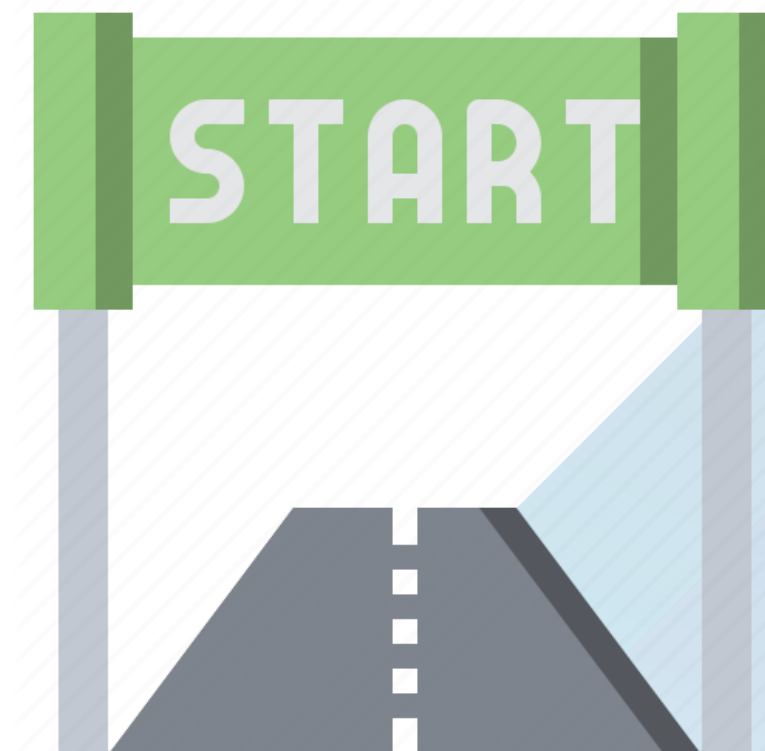
„Nie zawsze potrafię przewidywać,  
ale potrafię **kłaść podwaliny**.  
Bo **przyszłość** jest czymś,  
co się **buduje**”

Antoine de Saint-Exupery



# Spis treści

1. Po co nam system kontroli wersji?
2. GIT – instrukcja step by step
3. GIT – schemat działania
4. Zadania do samodzielnego rozwiązania
5. Przydatne linki



# Po co nam system kontroli wersji?

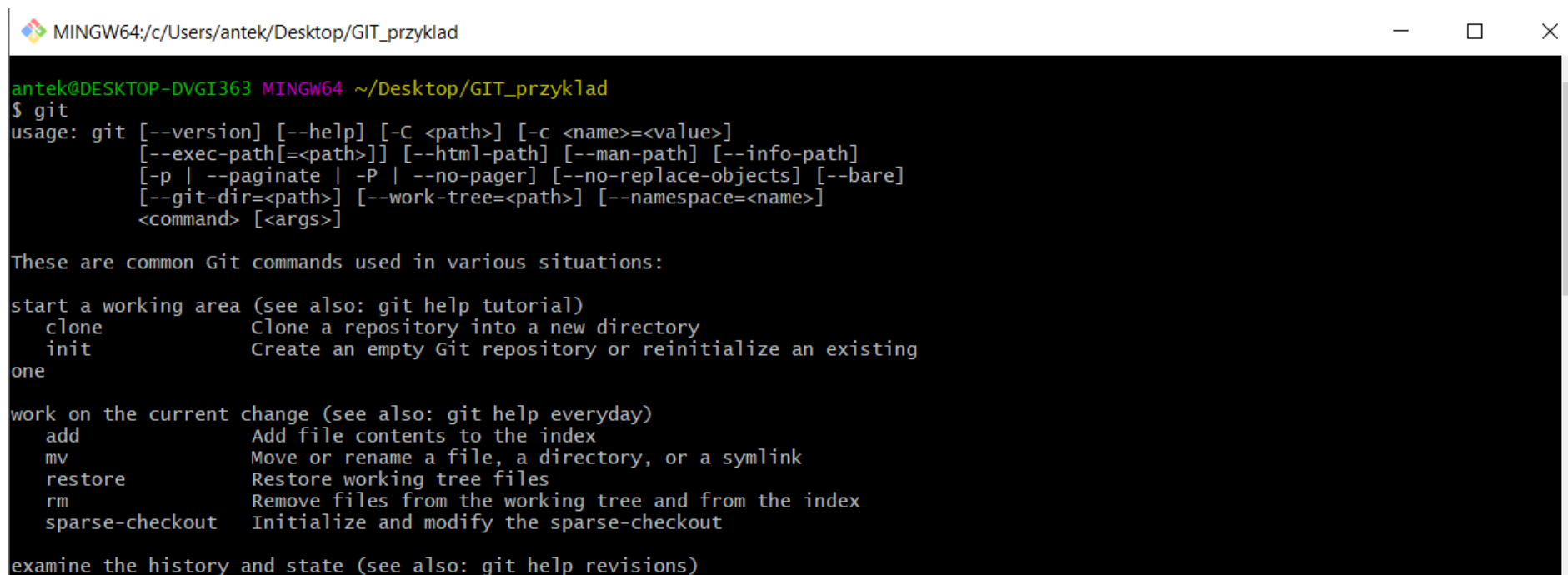
- Co to jest VCS? (*Version Control System*)
- Do czego służy VCS?
  - I. Przegląd historii zmian, przykład: [Commits · Leaflet/Leaflet \(github.com\)](https://github.com/Leaflet/Leaflet/commits)
  - II. Informacja o użytkownikach, którzy tych zmian dokonali
  - III. Możliwość powrotu do dowolnej, poprzedniej wersji pliku/projektu
  - IV. Kontrola pracy zespołowej
- Przydatność VCS podczas pracy w zespole – projekt musi aktualny na każdym komputerze używanym do pracy
- Repozytorium jako *backup*
- Inne narzędzia do kontroli wersji [Comparison of version-control software – Wikipedia](https://en.wikipedia.org/wiki/Comparison_of_version-control_software)
- Tworzenie własnego *portfolio* w kontekście przyszłych rozmów o pracę
- Rozgałęzianie kodu (*branching*) – możliwy na przykład przy użyciu GITa. Daje możliwość tworzenia nowych funkcji aplikacji niezależnie od siebie



Podgląd zmian wykonanych w kodzie

# GIT – instrukcja step by step cz. 1

- **Utwórz konto w serwisie [GitHub](#)**
- **Zainstaluj [Git - Downloads \(git-scm.com\)](#) Tutorial instalacji: [\[Kurs Gita w praktyce\] Jak zainstalować Gita? 🖥️ cz.2 \(#10\) - YouTube](#)**
- **Utwórz pusty folder projektowy na dysku lokalnym**
- **W pustym folderze naciśnij prawym przyciskiem myszy i uruchom Git Bash (Rys.1)**
- **W otwartym oknie konsoli wpisz polecenie `git` aby wyświetlić listę dostępnych poleceń (Rys. 2.)**



```
MINGW64/c:/Users/antek/Desktop/GIT_przyklad
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad
$ git
usage: git [--version] [--help] [-C <path>] [-c <name>=<value>]
      [--exec-path[=<path>]] [--html-path] [--man-path] [--info-path]
      [-p | --paginate | -P | --no-pager] [--no-replace-objects] [--bare]
      [--git-dir=<path>] [--work-tree=<path>] [--namespace=<name>]
      <command> [<args>]

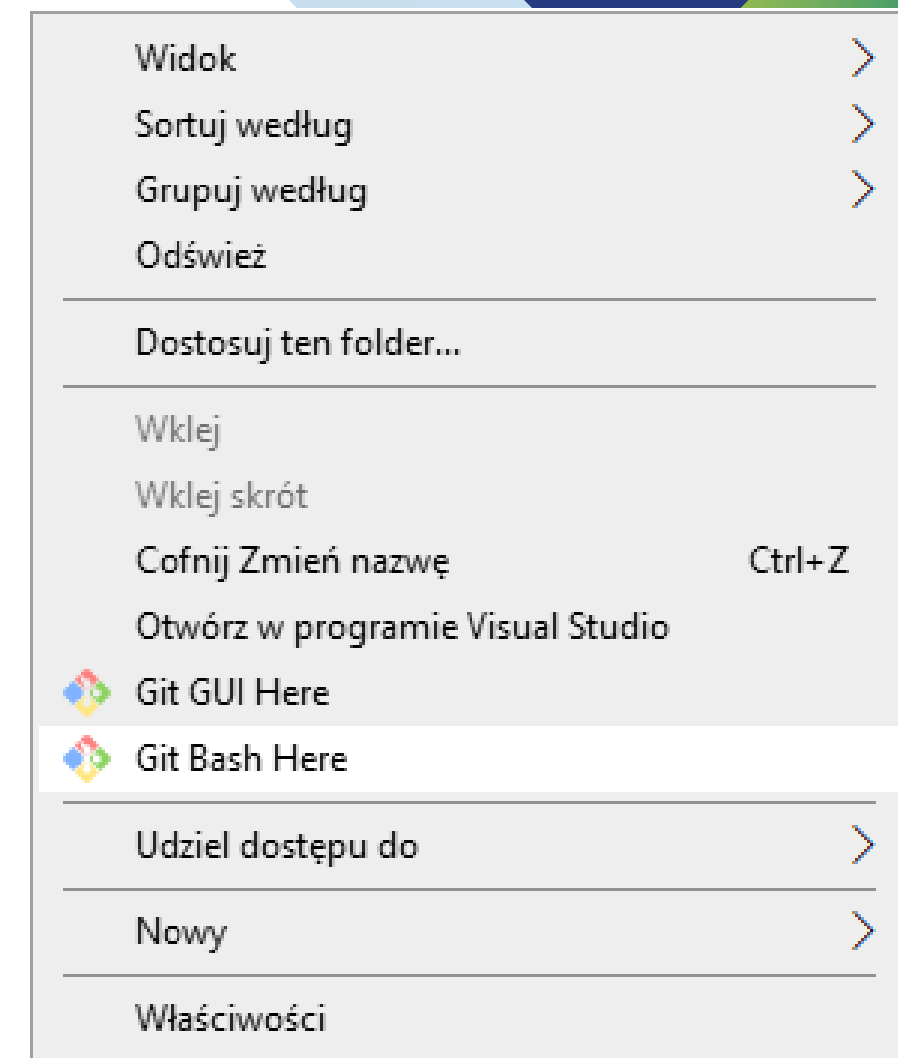
These are common Git commands used in various situations:


start a working area (see also: git help tutorial)
  clone      Clone a repository into a new directory
  init       Create an empty Git repository or reinitialize an existing
one

work on the current change (see also: git help everyday)
  add        Add file contents to the index
  mv         Move or rename a file, a directory, or a symlink
  restore    Restore working tree files
  rm         Remove files from the working tree and from the index
  sparse-checkout  Initialize and modify the sparse-checkout

examine the history and state (see also: git help revisions)
```

Rys. 2. Okno komend po wpisaniu polecenia `git`



Rys. 1. Uruchomienie Git Bash

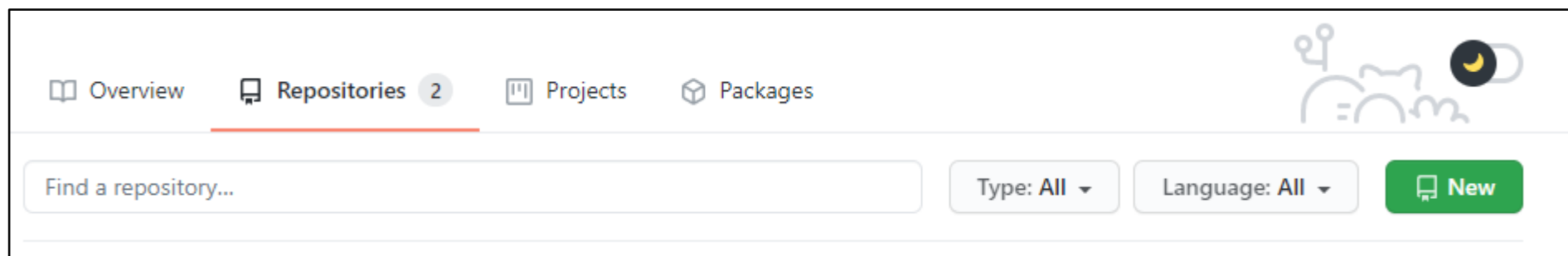
# GIT – instrukcja step by step cz. 2

- Zainicjuj repozytorium GIT wpisując komendę `git init` (Rys. 3.)
- Stwórz repozytorium na stronie [GitHub](#) po zalogowaniu (Rys. 4a, 4b)

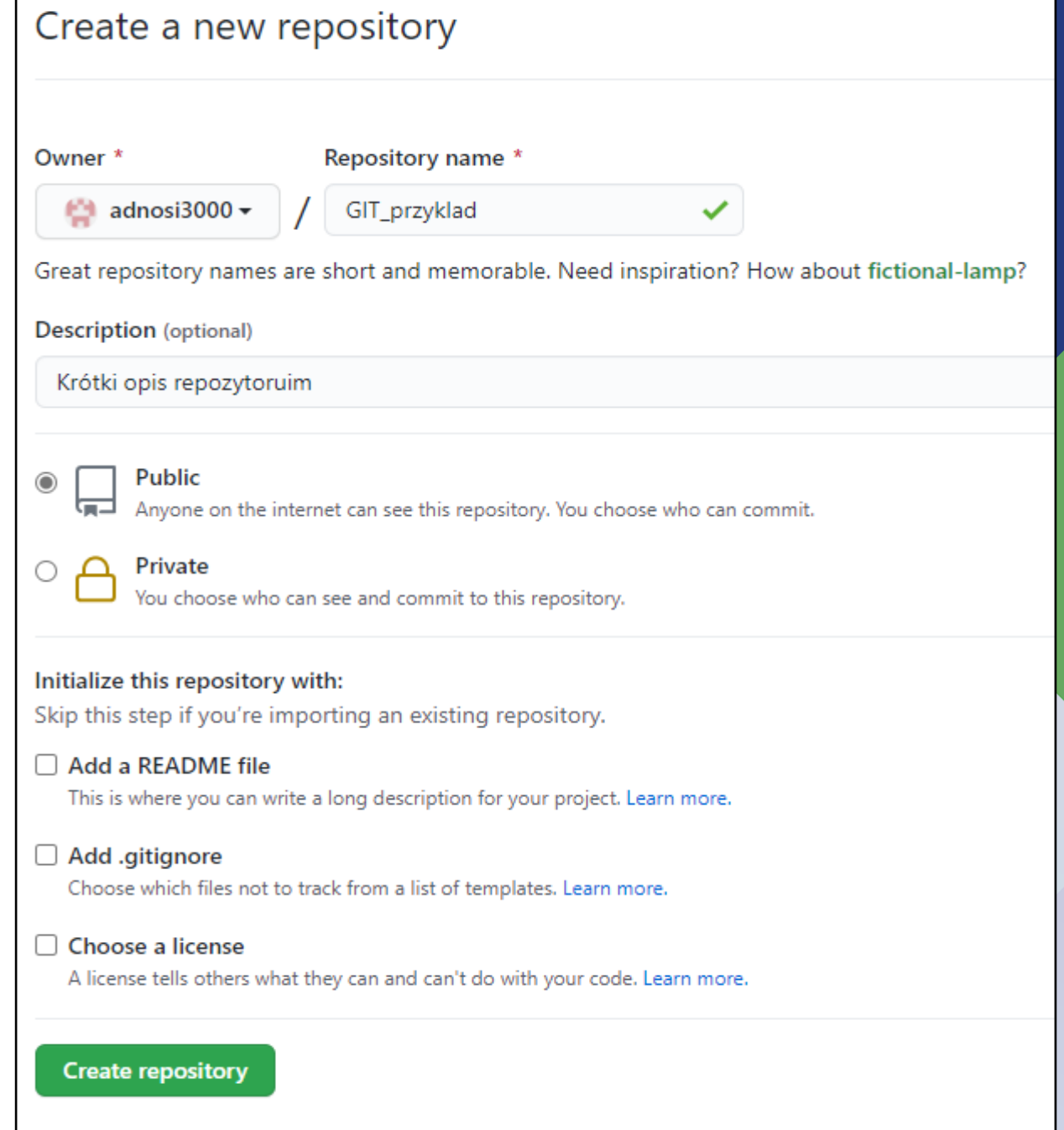
```
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad
$ git init
Initialized empty Git repository in C:/Users/antek/Desktop/GIT_przyklad/.git/

antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ |
```

Rys. 3. Fragment okna komend po wpisaniu polecenia `git init`




Rys. 4a. Przycisk do tworzenia nowego repozytorium

The screenshot shows the 'Create a new repository' form on GitHub. The title is 'Create a new repository'. Below the title, there are two input fields: 'Owner' and 'Repository name'. The 'Owner' field contains 'adnosi3000' and the 'Repository name' field contains 'GIT\_przyklad', which is marked with a green checkmark. Below these fields, there is a note: 'Great repository names are short and memorable. Need inspiration? How about fictional-lamp?'. There is also a 'Description (optional)' text area with the placeholder text 'Krótki opis repozytorium'. Below the description, there are two radio button options: 'Public' (selected) and 'Private'. The 'Public' option has a subtext: 'Anyone on the internet can see this repository. You choose who can commit.' The 'Private' option has a subtext: 'You choose who can see and commit to this repository.' Below these options, there is a section titled 'Initialize this repository with:' with the instruction 'Skip this step if you're importing an existing repository.' There are three checkboxes: 'Add a README file' (with subtext 'This is where you can write a long description for your project. Learn more.'), 'Add .gitignore' (with subtext 'Choose which files not to track from a list of templates. Learn more.'), and 'Choose a license' (with subtext 'A license tells others what they can and can't do with your code. Learn more.'). At the bottom of the form is a green button labeled 'Create repository'.

Rys. 4b. Formularz do tworzenia nowego repozytorium

# GIT – instrukcja step by step cz. 3


### Quick setup — if you've done this kind of thing before

 Set up in Desktop

 or 

HTTPS


SSH



Get started by [creating a new file](#) or [uploading an existing file](#). We recommend every repository include a [README](#), [LICENSE](#), and [.gitignore](#).


### ...or create a new repository on the command line

```
echo "# GIT_przyklad" >> README.md
git init
git add README.md
git commit -m "first commit"
git branch -M main
git remote add origin https://github.com/adnosi3000/GIT_przyklad.git
git push -u origin main
```



### ...or push an existing repository from the command line

```
git remote add origin https://github.com/adnosi3000/GIT_przyklad.git
git branch -M main
git push -u origin main
```



### ...or import code from another repository

You can initialize this repository with code from a Subversion, Mercurial, or TFS project.

Import code

Rys. 5. Pomoc GitHub'a po utworzeniu repozytorium, zniknie po dodaniu pierwszego pliku

# GIT – instrukcja step by step cz. 4

- Aby podejrzeć status projektu wpisz do konsoli komendę `git status` (Rys. 6.)
- Stwórz plik w formacie `.txt` i zapisz we wskazanym wcześniej folderze projektowym (Rys. 7.)

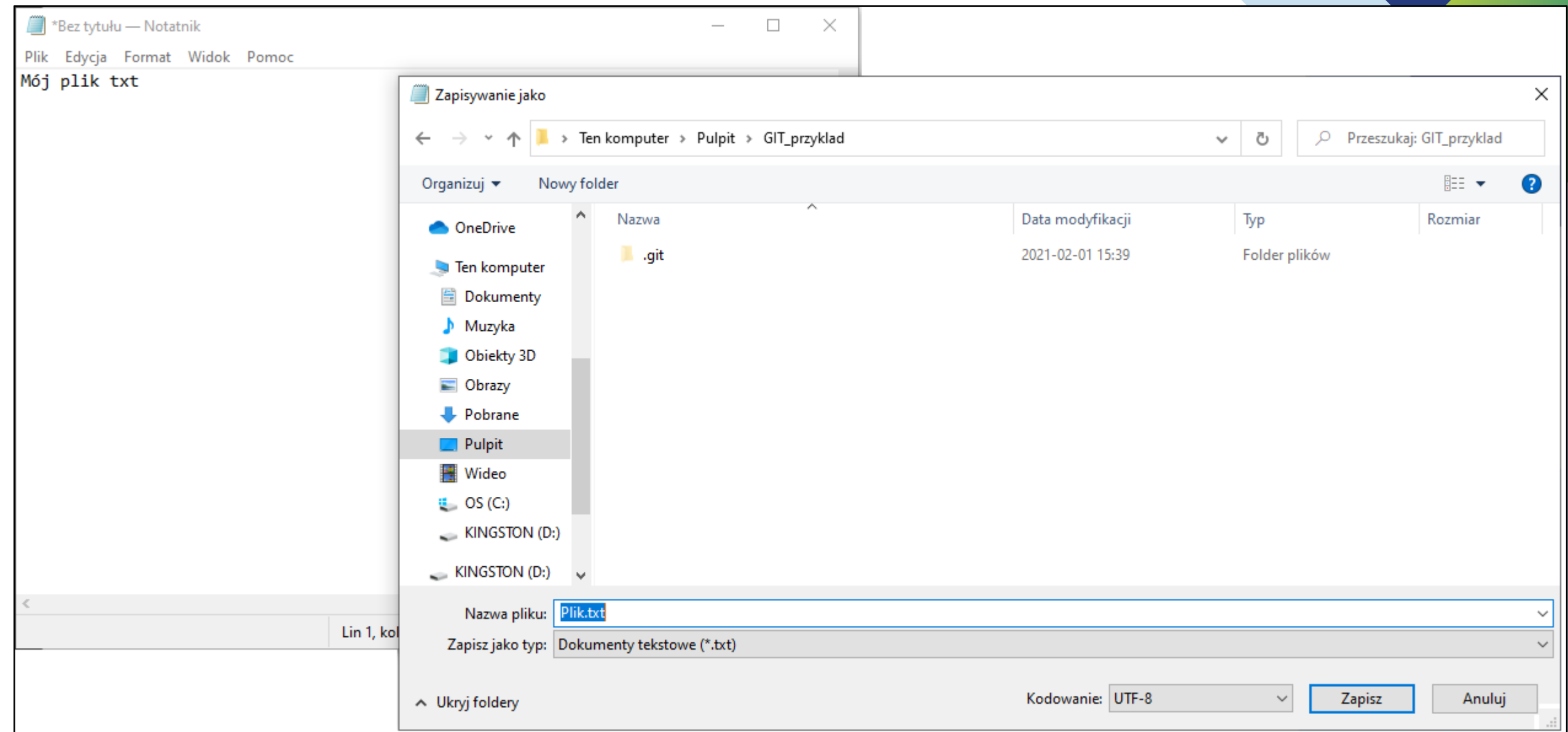
```
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ git status
On branch main

No commits yet

nothing to commit (create/copy files and use "git add" to track)

antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ |
```

Rys. 6. Fragment okna komend po wpisaniu polecenia `git status`



Rys. 7. Utworzenie i zapisanie pustego pliku `txt`



## GIT – instrukcja step by step cz. 5

- Użyj komendy `git status` jeszcze raz. Jakie widzisz zmiany? (Rys. 8.)
- Dodaj pliki do repozytorium za pomocą komendy `git add -A` (flaga `-A` sprawia, że dodajemy od razu wszystkie pliki, które znajdują się w folderze). Ponownie sprawdź status projektu komendą `git status` (Rys. 9.).

```
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ git status
On branch main

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        Plik.txt

nothing added to commit but untracked files present (use "git add" to track)
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ |
```

Rys. 8. Fragment okna komend po wpisaniu polecenia `git status`

```
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ git add -A

antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ git status
On branch main

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   Plik.txt

antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ |
```

Rys. 9. Fragment okna komend po wpisaniu polecenia `git add -A` oraz `git status`



# GIT – instrukcja step by step cz. 6

- Przygotuj „paczkę zmian” do wysłania na serwer za pomocą komendy `git commit -m 'message'` (w miejsce message wpisz krótki komentarz do zmian) (Rys. 10.)
- **Tylko za pierwszym razem:** użyj komendy `git remote add origin https://github.com/YourRepoAdres` aby wskazać adres Twojego repozytorium (dostępny do skopiowania na serwisie GitHub) (Rys. 11.)

```
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ git commit -m 'Tresc krotkiej wiadomosci o zmianach w plikach'
[main (root-commit) ace8f7b] Tresc krotkiej wiadomosci o zmianach w plikach
1 file changed, 1 insertion(+)
create mode 100644 Plik.txt

antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ |
```

Rys. 10. Fragment okna komend po wpisaniu polecenia `git commit -m 'Tresc krotkiej wiadomosci o zmianach w plikach'`

```
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ git remote add origin https://github.com/adnosi3000/GIT_przyklad.git

antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ |
```

Rys. 11. Fragment okna komend po wpisaniu polecenia `git remote add origin https://github.com/adnosi3000/GIT_przyklad.git`

# GIT – instrukcja step by step cz. 7

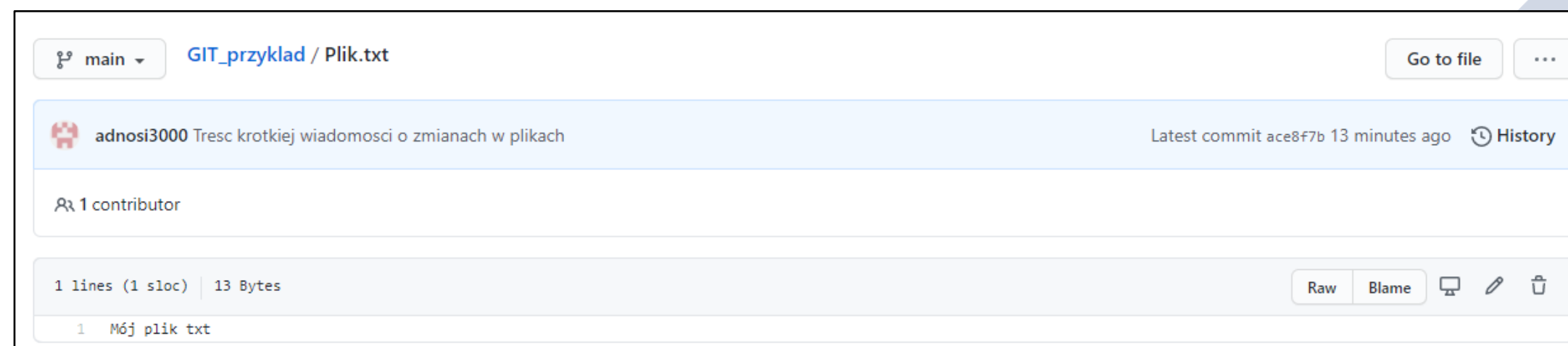
- **Tylko za pierwszym razem:** użyj komendy `git branch -M main`
- Prześlij przygotowaną „paczkę plików” do repozytorium za pomocą komendy `git push -u origin main` (kolor czerwony: **tylko za pierwszym razem**). Konieczne będzie utworzenie tokena!!! [Creating a personal access token - GitHub Docs](#). W okno logowania wklej swój wygenerowany token. (Rys. 12.)
- Odśwież stronę repozytorium w serwisie GitHub, Twoje pliki powinny być widoczne (Rys. 13.)
- Aby pobrać pliki używamy komendy `git pull`

```
antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ git branch -M main

antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$ git push -u origin main
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Writing objects: 100% (3/3), 255 bytes | 85.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/adnosi3000/GIT_przyklad.git
 * [new branch]      main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.

antek@DESKTOP-DVGI363 MINGW64 ~/Desktop/GIT_przyklad (main)
$
```

Rys. 12. Fragment okna komend po wpisaniu polecenia `git branch -M main` oraz `git push -u origin main`



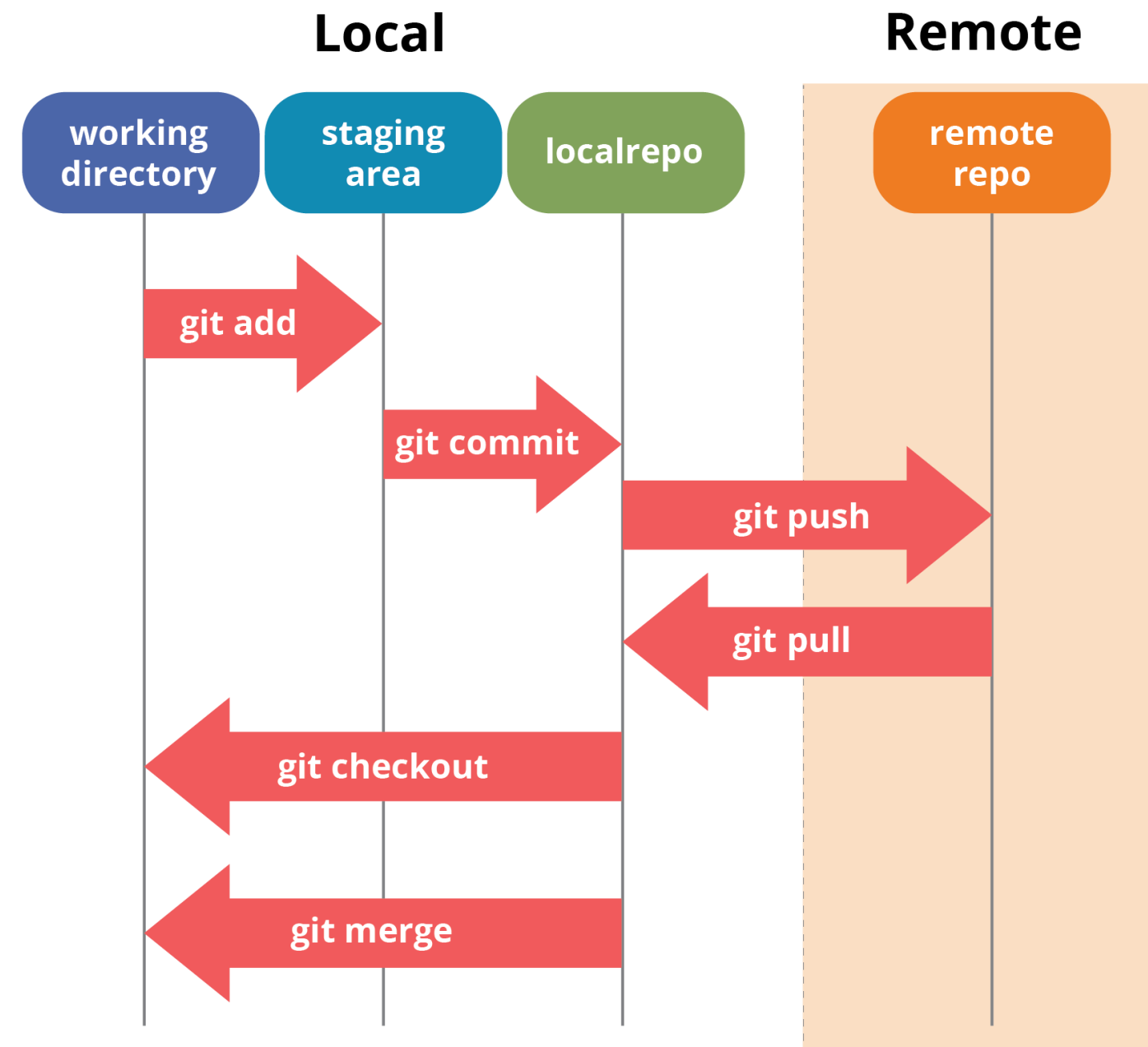
Rys. 13. Plik txt widoczny w repozytorium

# Miejsce na notatki

A large, empty rectangular area with a dashed blue border, intended for taking notes.



# GIT – schemat działania

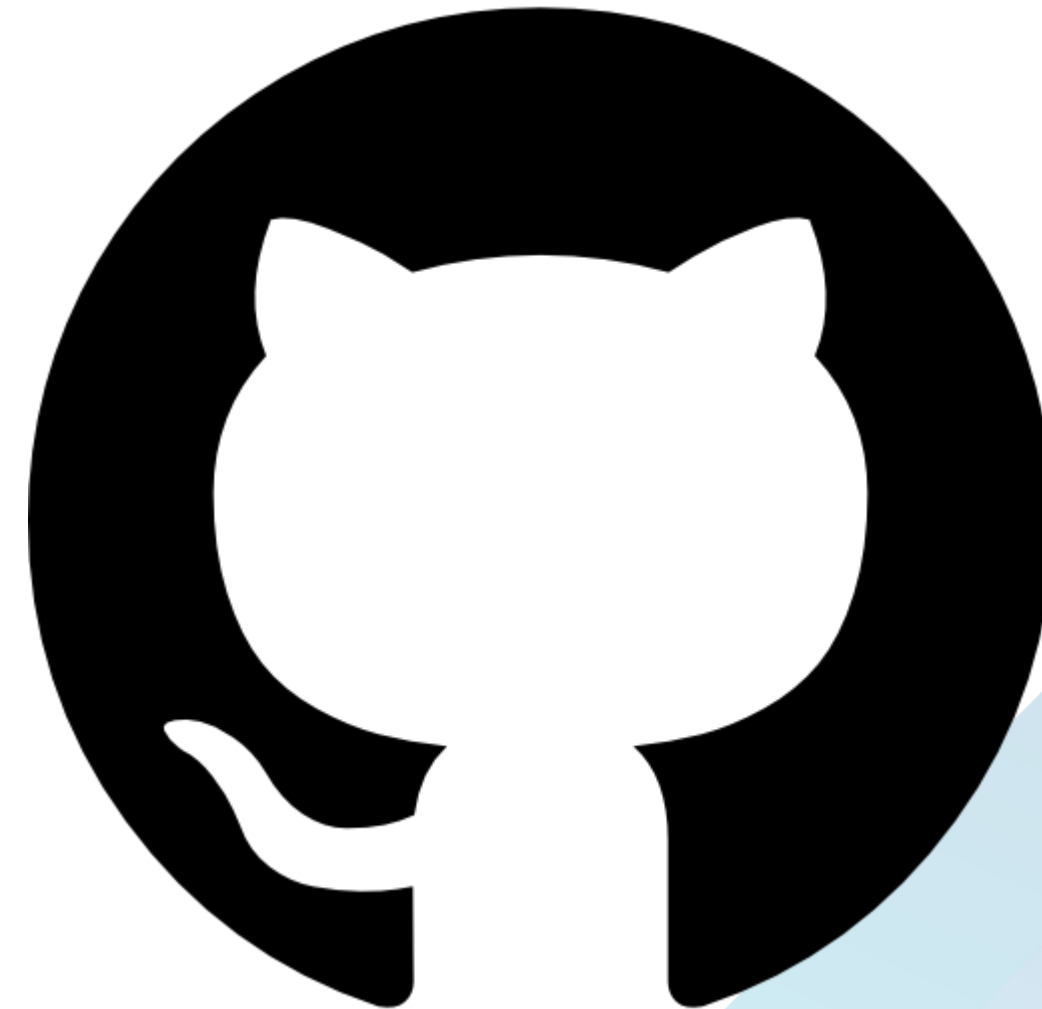


Rys. 14. Schemat działania git, źródło: [3]

# Zadania do samodzielnego rozwiązania

## Zadanie 1

Stwórz nowe repozytorium w taki sam sposób, jak było to zaprezentowane na zajęciach. Dodaj do niego kilka plików o różnych rozszerzeniach: .py .txt .jpg; Zwróć uwagę na podgląd i możliwość edytowania plików w serwisie GitHub. Dokonaj edycji pliku on-line a następnie pobierz go do lokalnego katalogu za pomocą polecenia `git pull`. Znow dokonaj edycji pliku (tym razem lokalnie) i przełącz zaktualizowany plik do repozytorium.



## Przydatne linki

- [1] [GitHub](#)
- [2] [Git - Downloads \(git-scm.com\)](#)
- [3] [Git Tutorial | Commands And Operations In Git | Edureka](#)
- [4] [Learn Git In 15 Minutes – YouTube](#)
- [5] [Creating a personal access token - GitHub Docs](#)
- [6] [\[Kurs Gita w praktyce\] Jak zainstalować Gita? 🖥️ cz.2 \(#10\) – YouTube](#)
- [7] [Comparison of version-control software – Wikipedia](#)
- [8] [\[Kurs Gita w praktyce\] Czym jest Git i po co Ci GitHub? 🖥️ cz.1 \(#9\) – YouTube](#)
- [9] [#git Czym jest Git? - Kurs gita po polsku #1/12 - YouTube](#)





# SMARTFACTOR



+48 798 622 487



ul. Poselska 29  
03-931 Warszawa



mail@smartfactor.pl



www.smartfactor.pl