

**Akademia Górniczo-Hutnicza  
im. Stanisława Staszica w Krakowie**

**Wydział Inżynierii Mechanicznej i Robotyki**



Instrukcja do laboratorium

Programowanie Obiektowe  
I Inżynieria Oprogramowania

**1a**

**Narzędzia stosowane na zajęciach  
Wprowadzenie do GIT**

**Wersja 1.0**

Spis treści

1	Wstęp .....	3
2	Git – tworzenie lokalnego repozytorium .....	3
2.1	Repozytorium z projektami tworzonymi podczas zajęć - Konsola .....	3
3	Dodatkowe opcje .....	8
3.1	Ignorowanie plików .....	9
3.2	Przegląd logów i historii zmian .....	9
3.3	Poprzednie wersje .....	9
3.4	Przeglądanie aktualnych zmian .....	<b>Błąd! Nie zdefiniowano zakładki.</b>
3.5	Przeglądanie aktualnych zmian .....	9

Spis zadań

Zadanie 1	Komendy Unix.....	3
Zadanie 2	git add .....	6
Zadanie 3	katalog, plik.....	8
Zadanie 4	usuwanie plików z przechowalni .....	8
Zadanie 5	commity (migawki, zatwierdzenia).....	8
Zadanie 5	kolejne repozytorium .....	8
Zadanie 5	git ignore .....	9
Zadanie 5	git log .....	9
Zadanie 5	git show.....	9
Zadanie 5	git diff.....	9

## 1 Wstęp

Kurs inżynierii oprogramowania obejmuje pracę w środowisku kontroli wersji GitLab. W tej instrukcji zamieszczono podstawy działania środowiska, sposób utworzenia lokalnego repozytorium, a także konta na zdalnym serwerze repozytoriów. Instrukcja przeznaczona jest na 2 zajęcia.

Uwaga! Instrukcje obejmujące tematykę git publikowane są w wersji 1.0. Jeżeli znajdziecie Państwo w nich błędy lub coś będzie niejasne i pojawią się sugestie co do poprawy proszę o kontakt na e-mail [ambrozin@agh.edu.pl](mailto:ambrozin@agh.edu.pl).

Instrukcje przygotowano w oparciu o książkę Mariot Tsioara „Git i GitHub – Kontrola wersji, zarządzanie projektami i zasady pracy zespołowej” Helion 2022.

## 2 Git – tworzenie lokalnego repozytorium

### Zadanie 1 Komendy Unix

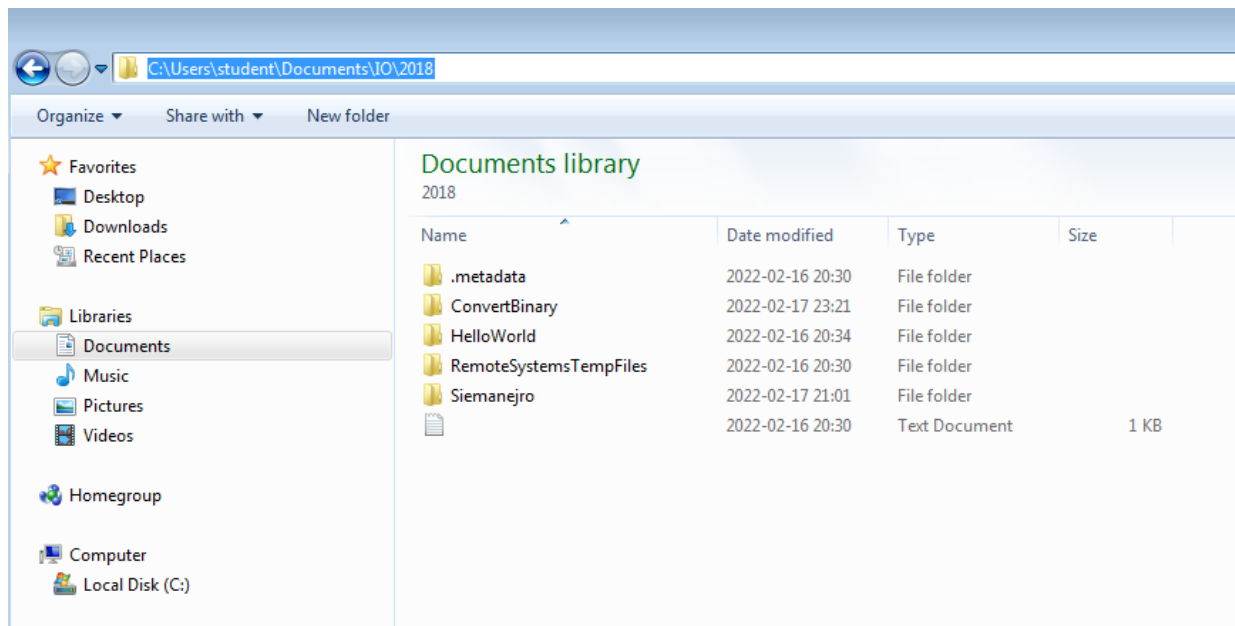
Przypomnij sobie najważniejsze komendy unixowe służące do nawigacji, tworzeniu i usuwaniu katalogów.

Tablica poniżej przedstawia najważniejsze komendy potrzebne do tworzenia repozytoriów.

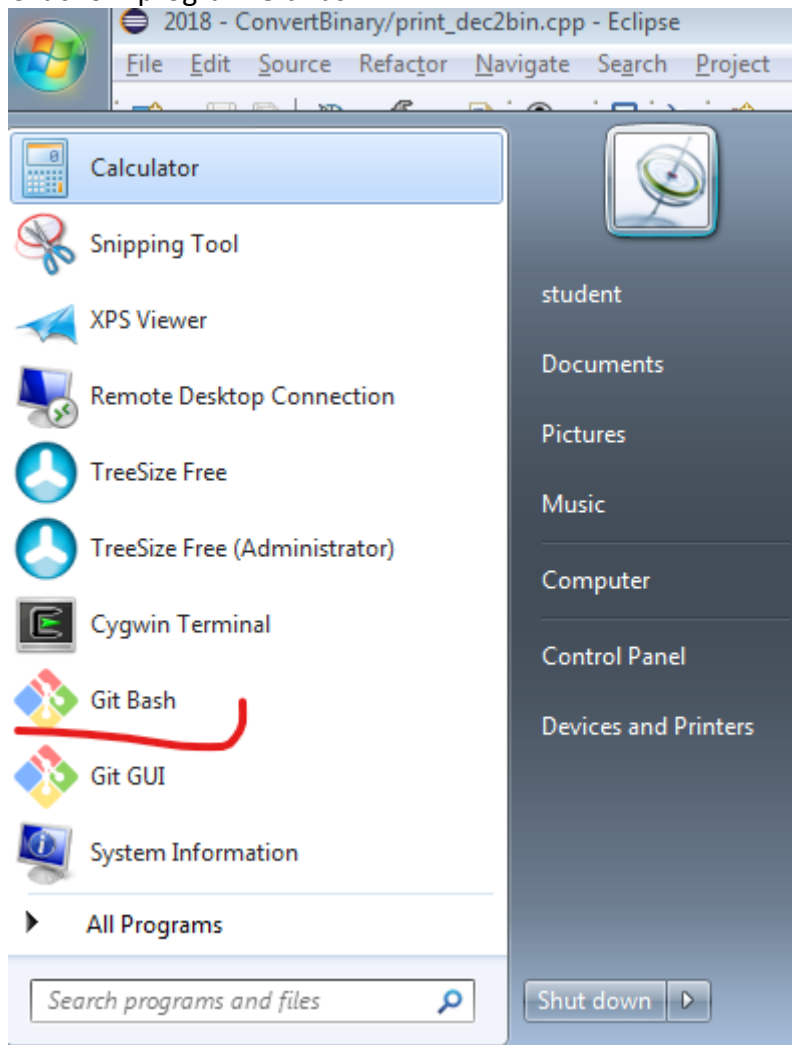
Komenda	Znaczenie
<b>pwd</b>	Obecny katalog
<b>cd</b>	Zmień katalog
<b>ls</b>	Wypisz zawartość katalogu
<b>git init</b>	Utworzenie lokalnego repozytorium
<b>git status</b>	Informacje o stanie repozytorium
<b>git add [folder]</b>	Dodanie folderu do indeksowania przez gita
<b>git commit -m „[comment]”</b>	Przesłanie obecnego stanu do repozytorium. Koniecznie dodaj komentarz!

### 2.1 Repozytorium z projektami tworzonymi podczas zajęć - Konsola

Nawiguj za pomocą eksploratora windows do miejsca, w którym zostały zapisane pliki projektów C++.



Uruchom program Git Bash



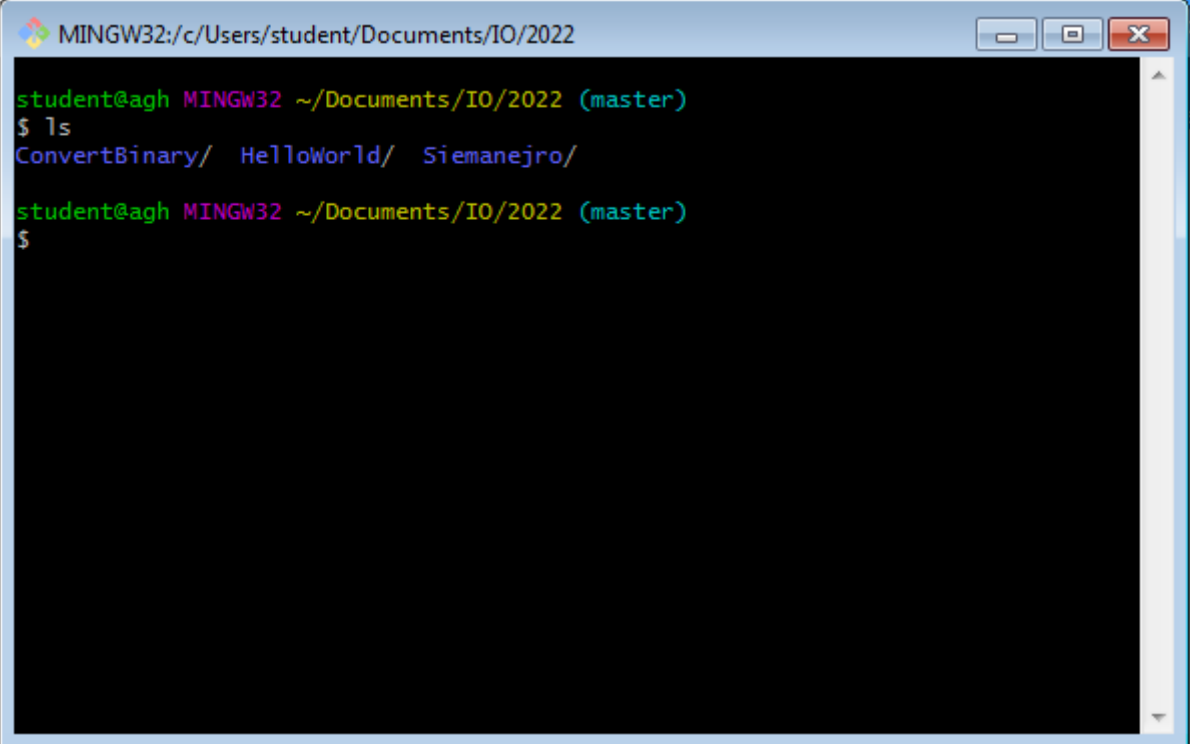
Skonfiguruj Gita swoimi danymi użytkownika

```
student@agh MINGW32 ~/Documents/IO/2022 (main)
$ git config --global user.name "Lukasz Ambrozinski"

student@agh MINGW32 ~/Documents/IO/2022 (main)
$ git config --global user.email "ambrozin@agh.edu.pl"

6 student@agh MINGW32 ~/Documents/IO/2022 (main)
$ |
```

W uruchomionym oknie w linii komend zmień katalog na ten zawierający pliki projektów.



```
MINGW32:/c/Users/student/Documents/IO/2022

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ ls
ConvertBinary/  HelloWorld/  Siemanejro/

student@agh MINGW32 ~/Documents/IO/2022 (master)
$
```

Wywołaj komendę git init

```
MINGW32:/c/Users/student/Documents/IO/2022

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ ls
ConvertBinary/ HelloWorld/ Siemanejro/

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ git init
Initialized empty Git repository in C:/Users/student/Documents/IO/2022/.git/

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ ls
ConvertBinary/ HelloWorld/ Siemanejro/

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ ls -a
./ ../ .git/ ConvertBinary/ HelloWorld/ Siemanejro/

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ |
```

Zwróć uwagę, że w katalogu docelowym pojawił się ukryty katalog .git

Dodaj katalogi do przechowalni – polecenie git add.

```
MINGW32:/c/Users/student/Documents/IO/2022

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ git add HelloWorld/
warning: LF will be replaced by CRLF in HelloWorld/.gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in HelloWorld/src/HelloWorld.cpp.
The file will have its original line endings in your working directory

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ git add .
warning: LF will be replaced by CRLF in ConvertBinary/.gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in Siemanejro/.gitignore.
The file will have its original line endings in your working directory
warning: LF will be replaced by CRLF in Siemanejro/src/Siemanejro.cpp.
The file will have its original line endings in your working directory

student@agh MINGW32 ~/Documents/IO/2022 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
```

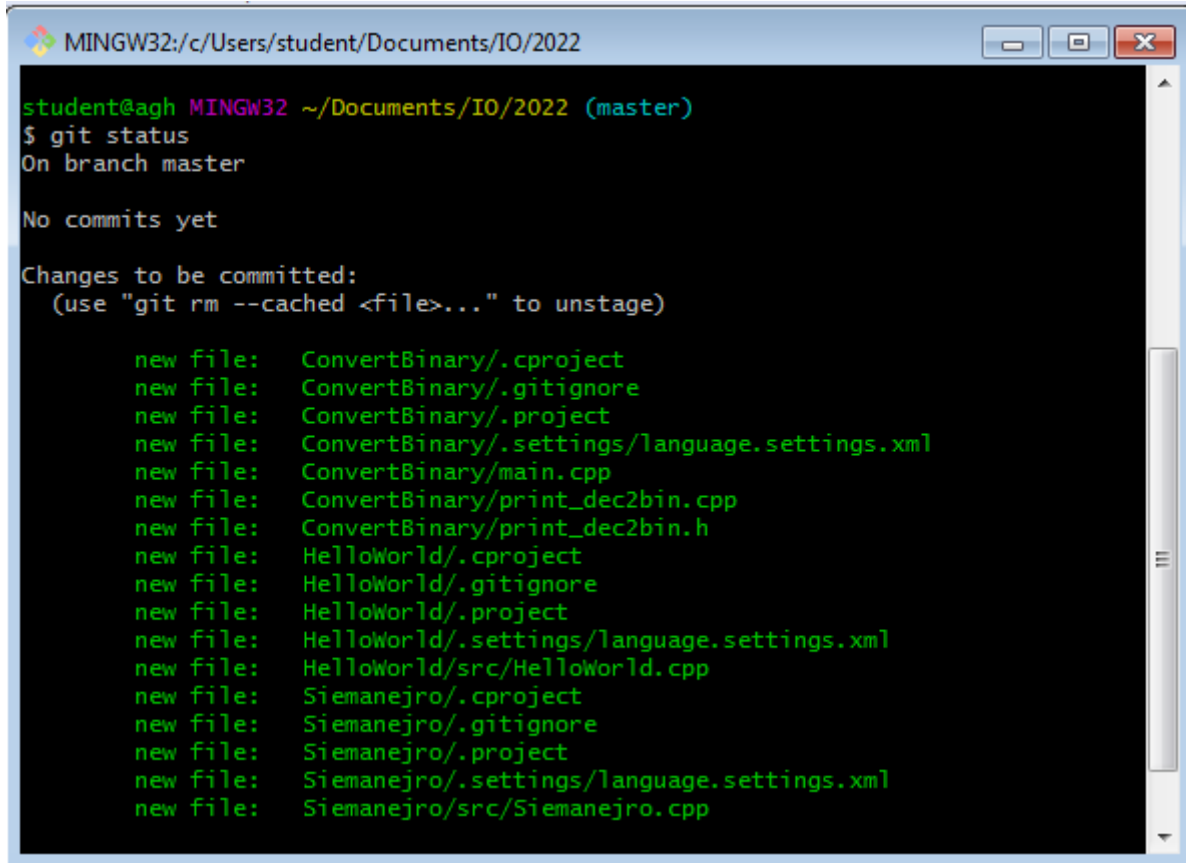
#### Zadanie 2 git add

Opisz różnice pomiędzy komendą:  
git add NazwaFolderu

a

```
git add .
```

Wykonaj komendę git status

A screenshot of a terminal window titled 'MINGW32:/c/Users/student/Documents/IO/2022'. The terminal shows the output of the 'git status' command. It indicates that the user is on the 'master' branch and that there are no commits yet. A list of 15 new files is shown, including project files for 'ConvertBinary', 'HelloWorld', and 'Siemanejro' directories, as well as their respective settings and source files.

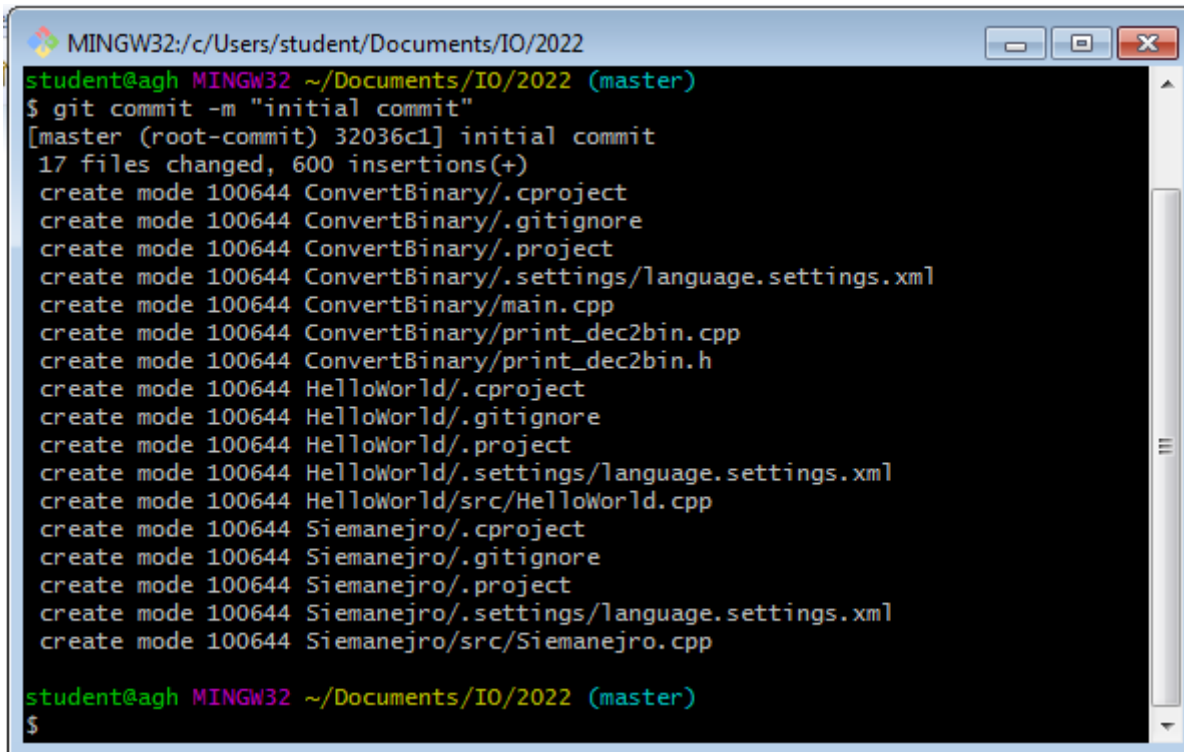
```
student@agh MINGW32 ~/Documents/IO/2022 (master)
$ git status
On branch master

No commits yet

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

    new file:   ConvertBinary/.cproject
    new file:   ConvertBinary/.gitignore
    new file:   ConvertBinary/.project
    new file:   ConvertBinary/.settings/language.settings.xml
    new file:   ConvertBinary/main.cpp
    new file:   ConvertBinary/print_dec2bin.cpp
    new file:   ConvertBinary/print_dec2bin.h
    new file:   HelloWorld/.cproject
    new file:   HelloWorld/.gitignore
    new file:   HelloWorld/.project
    new file:   HelloWorld/.settings/language.settings.xml
    new file:   HelloWorld/src/HelloWorld.cpp
    new file:   Siemanejro/.cproject
    new file:   Siemanejro/.gitignore
    new file:   Siemanejro/.project
    new file:   Siemanejro/.settings/language.settings.xml
    new file:   Siemanejro/src/Siemanejro.cpp
```

Następnie wykonaj komendę git commit -m „komentarz ”



```
MINGW32:/c/Users/student/Documents/IO/2022
student@agh MINGW32 ~/Documents/IO/2022 (master)
$ git commit -m "initial commit"
[master (root-commit) 32036c1] initial commit
17 files changed, 600 insertions(+)
create mode 100644 ConvertBinary/.cproject
create mode 100644 ConvertBinary/.gitignore
create mode 100644 ConvertBinary/.project
create mode 100644 ConvertBinary/.settings/language.settings.xml
create mode 100644 ConvertBinary/main.cpp
create mode 100644 ConvertBinary/print_dec2bin.cpp
create mode 100644 ConvertBinary/print_dec2bin.h
create mode 100644 HelloWorld/.cproject
create mode 100644 HelloWorld/.gitignore
create mode 100644 HelloWorld/.project
create mode 100644 HelloWorld/.settings/language.settings.xml
create mode 100644 HelloWorld/src/HelloWorld.cpp
create mode 100644 Siemanejro/.cproject
create mode 100644 Siemanejro/.gitignore
create mode 100644 Siemanejro/.project
create mode 100644 Siemanejro/.settings/language.settings.xml
create mode 100644 Siemanejro/src/Siemanejro.cpp
student@agh MINGW32 ~/Documents/IO/2022 (master)
$
```

### Zadanie 3 katalog, plik

W katalogu roboczym utwórz nowy katalog (użyj poleceń z poziomu konsoli).  
W katalogu umieść plik readme.md. Następnie wykonaj operację  
git add oraz git commit (pomiędzy wywołaniami tych poleceń wywołaj git status).

### Zadanie 4 usuwanie plików z przechowalni

W usuń plik readme.md z przechowalni. Zastosuj polecenie  
Git rm -- cached readme.md  
Sprawdź stan poleceniem git status.

### Zadanie 5 commity (migawki, zatwierdzenia)

Utwórz commit (migawkę) aktualnego stanu katalogu. Wykorzystaj polecenie git commit

## 3 Dodatkowe opcje

### Zadanie 6 kolejne repozytorium

Zanim przejdziemy dalej wykonaj następujące czynności:

- Utwórz nowe repozytorium
- Utwórz pliki TODO.txt i zapisz w nim tekst
- Wyślij plik TODO.txt do przechowalni
- **Zatwierdź** projekt i wprowadź opis **commita**



- Utwórz dwa dodatkowe pliki DONE.txt oraz WORKING.txt
- Wyślij je do przechowalni, a następnie zatwierdź
- Zmień nazwę WORKING.txt na IN Progress.txt
- Dodaj tekst do DONE.txt
- Sprawdź stan katalogu
- Wyślij plik IN Progress.txt i DONE.txt do przechowalni
- Wycofaj plik DONE.txt z przechowalni
- Zatwierdź projekt
- Sprawdź stan katalogu

### 3.1 Ignorowanie plików

Niektóre pliki nie chcemy aby były śledzone przez gita. Te pliki wpisywane są do pliku o nazwie .gitignore.

#### Zadanie 7 git ignore

W bieżącym repozytorium utwórz plik „PRIVATE.txt”  
Utwórz plik .gitignore i umieść w nim nazwę pliku PRIVATE.txt  
Sprawdź polecenie git status

Dodaj plik .gitignore do przechowalni, następnie wykonaj commit  
Sprawdź git status

### 3.2 Przegląd logów i historii zmian

#### Zadanie 8 git log

Wywołaj komendę git log - zaobserwuj jej działanie

### 3.3 Poprzednie wersje

Aby zobaczyć poprzednie wersje wystarczy użyć polecenia git show

#### Zadanie 9 git show

Wywołaj komendę git show - zaobserwuj jej działanie

### 3.4 Przeglądanie aktualnych zmian

#### Zadanie 10 git diff

Zmodyfikuj jeden z plików tekstowych w katalogu. Wywołaj komendę git diff nazwa.pliku - zaobserwuj jej działanie

## **4 Commity**

Uwaga konkurs! Jak przetłumaczyć na j. polski commit w kontekście gita? W instrukcji będzie się przewijać tłumaczenie „zatwierdź”, a obraz pliku utworzony tym sposobem nazywany jest migawką.

### **4.1 Trzy stany plików w repozytorium**