Praca nad projektem z wykorzystaniem Git

Git śledzi zmiany plików w obrębie konkretnego folderu. Nie ma znaczenia, czy folder ten zawiera kod źródłowy programu komputerowego, rękopis książki czy stronę WWW. Folder, którego zawartość jest kontrolowana przez Git, będziemy nazywali **repozytorium** (ang. *repository*). Repozytoria zawierają specjalny podfolder *.git*, w którym zapisywane są szczegółowe dane o śledzonych plikach

Uruchomienia śledzenia zmian

Cel ćwiczenia
W lokalnym repozytorium tworzymy dokumentację projektu o nazwie Projekt1.
Dokumentacja zawiera 2 rozdziały z plikami opisującymi odpowiednio dane do poszczególnych rozdziałów
schemat katalogu dokumentacji:

```
$ ls -al projekt1/dokumentacja/
total 0
drwxr-xr-x 1 lucyna 197121 0 Oct 25 15:24 ./
drwxr-xr-x 1 lucyna 197121 0 Oct 25 15:24 ../
drwxr-xr-x 1 lucyna 197121 0 Oct 25 15:24 rozdzial1/
drwxr-xr-x 1 lucyna 197121 0 Oct 25 15:24 rozdzial2/
```

Zadaniem naszym jest śledzenie zmian w dokumentacji.
Dajmy możliwość oprogramowaniu Git śledzenia zmian jakie dokonujemy w całym projekcie.
Przechodzimy do katalogu projektu i uruchamiamy śledzenie przez Git zmian dokonywanych w danym projekcie poleceniem **git init**

```
$ cd projekt1/

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1
$ git init
Initialized empty Git repository in C:/Users/lucyna/projekt1/.git/

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$
```

Zauważmy że git generuje nowy katalog ukryty .git jest to jeden katalog na cały projekt w którym zapisują się wszelkie zmiany dokonywane w tym katalogu

Zapisywanie zmian

Git w lokalnym repozytorium działa na 3 obszarach
katalog roboczy – working directory – tutaj znajdują się pliki naszego projektu
przechowalnia – stage area – miejsce do którego tymczasowo wpadają
repozytorium - .git folder

Tworzymy pliki z dokumentacją w poszczególnych rozdziałach dokumentacja1.txt i dokumentacja2.txt
Wydając polecenie **git status** możemy przeglądnąć co zostało zrobione w katalogu projekt1

```
$ git status
On branch master

No commits yet

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

nothing added to commit but untracked files present (use "git add" to track)

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
```

Widzimy że w katalogu zostały dokonane zmiany lecz nie zostały dodane do „przechowalni” i nie są zatwierdzone zmiany
Tak więc aby dodać zmiany do przechowalni wydajemy polecenie git add <file> lub git add . Które doda do przechowalni wszystkie pliki i katalogi które zostały dodane lub zmienione

```
$ git add .

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)
        new file:   dokumentacja/rozdzial1/dokumentacja1.txt
        new file:   dokumentacja/rozdzial2/dokumentacja2.txt
```

Zatwierdzamy bieżący stan operacją zatwierdzenia **git commit** flaga -m pozwala na dodanie komunikatu odnośnie dokonanych zmian

```
$ git commit -m "wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt"
[master (root-commit) eac687e] wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 dokumentacja/rozdzial1/dokumentacja1.txt
 create mode 100644 dokumentacja/rozdzial2/dokumentacja2.txt
```

Edytujemy plik dokumentacja1.txt i wprowadzamy zmiany dodając 2 nowe linijki tekstu, a następnie przekazujemy do przechowalni i zatwierdzamy zmiany w pliku

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   dokumentacja/rozdzial1/dokumentacja1.txt

no changes added to commit (use "git add" and/or "git commit -a")

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git add .

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   dokumentacja/rozdzial1/dokumentacja1.txt


lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git commit -m "wprowadzenie zmiany w  dokumentacja1.txt"
[master 06c2c8c] wprowadzenie zmiany w  dokumentacja1.txt
 1 file changed, 2 insertions(+)
```

Dokonujemy następnej zmiany w dokumentacji uzupełniając plik dokumentacja2.txt
Sprawdzamy status

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   dokumentacja/rozdzial2/dokumentacja2.txt

no changes added to commit (use "git add" and/or "git commit -a")

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$
```

Dodajemy zmiany do poczekalni i zatwierdzamy zmiany

```
$ git add .

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   dokumentacja/rozdzial2/dokumentacja2.txt


lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git commit -m "zmiana zawartości pliku dokumentacja2.txt"
[master fbe7294] zmiana zawartości pliku dokumentacja2.txt
 1 file changed, 2 insertions(+)
```

Praca w przestrzeni roboczej i poczekalni

Polecenie **git clean** służy do usuwania nieśledzonych plików i katalogów
W tym zadaniu tworzymy w przestrzeni roboczej kilka plików i katalogów. Chcemy usuną cześć plików.

```
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        css/
        plik_a
        plik_b
        plik_c

nothing added to commit but untracked files present (use "git add" to track)
```

Uruchamiamy polecenie **git clean** które pozwoli na skasowania plików w przestrzeni roboczej które nie zostały dodane do poczekalni

```
$ git clean
fatal: clean.requireForce defaults to true and neither -i, -n, nor -f given; refusing to
clean
```

Polecenie **git clean** kończy się niepowodzeniem wyłącznie ze względu na środki bezpieczeństwa.
Aby przeglądnąć listę plików do usunięcia wydajemy polecenie git clean z flagą -n a chcąc pokazać też katalogi dodajemy flagę d

```
$ git clean -n
Would remove plik_a
Would remove plik_b
Would remove plik_c

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git clean -nd
Would remove css/
Would remove plik_a
Would remove plik_b
Would remove plik_c
```

Uruchomienie polecenia w trybie interaktywnym pozwala wybrać co chcemy zachować a co kasujemy

```
$ git clean -idf
Would remove the following items:
  css/    plik_a  plik_b  plik_c
*** Commands ***
    1: clean                2: filter by pattern    3: select by numbers
    4: ask each             5: quit                 6: help
What now> 4
Remove css/ [y/N]? N
Remove plik_a [y/N]? y
Remove plik_b [y/N]? y
Remove plik_c [y/N]? N
Removing plik_a
Removing plik_b

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Untracked files:
```

```
  (use "git add <file>..." to include in what will be committed)
        css/
        plik_c

nothing added to commit but untracked files present (use "git add" to track)
```

Dodajemy do kolejki oczekiwania

```
$ git add .

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   css/plik1
        new file:   css/plik2
        new file:   css/plik3
        new file:   plik_c
```

możemy cofnąć plik z kolejki poleceniem **git reset**

```
$ git reset plik_c

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   css/plik1
        new file:   css/plik2
        new file:   css/plik3

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

Robimy teraz przeniesienie z poczekalni do repozytorium tych zmian poleceniem git commit

```
$ git commit -m "wprowadzenie nowego katalogu css z plikami i nowego pliku"
[master 5a47116] wprowadzenie nowego katalogu css z plikami i nowego pliku
 3 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 css/plik1
 create mode 100644 css/plik2
 create mode 100644 css/plik3

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        plik_c

nothing added to commit but untracked files present (use "git add" to track)
```

Widzimy że plik_c jest wycofany z poczekalni

Wprowadzamy zmiany w pliku css/plik1

```
$ git status
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   css/plik1

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

no changes added to commit (use "git add" and/or "git commit -a")

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)

Dodajemy zmodyfikowany plik do poczekalni:
$ git add css/plik1

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   css/plik1
```

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


lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)

Przesuwamy plik z poczekalni do repozytorium

$ git commit -m "wprowadzenie zmiany w css/plik1"
[master 40f1cfc] wprowadzenie zmiany w css/plik1
 1 file changed, 1 insertion(+)

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        plik_c

nothing added to commit but untracked files present (use "git add" to track)
```

Modyfikujemy plik css/plik1 a następnie kasujemy tą modyfikację i przywracamy stan pliku z indeksu gita

```
$ cat css/plik1
asdasdasd
jhsjashdjahsdjkahd

####Modyfikujemy plik

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ vi css/plik1
### wyświetlamy status

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        modified:   css/plik1
        new file:   plik_c

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   css/plik1


lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ cat css/plik1
asdasdasd
jhsjashdjahsdjkahd
ksdjkjdkal`

##### przywracamy plik

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ git checkout -- css/plik1

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
$ cat css/plik1
asdasdasd
jhsjashdjahsdjkahd

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)
```

Zestawienie poleceń :

git init
Inicjowanie repozytorium z możliwością śledzenie zmian i zapisywania ich
git add
polecenia przed zapisaniem zmian w repozytorium
git clean

Usuwa pliki które nie zostały dodane do indeksu Gita
git reset
Przenosi pliki z poczekalni do przestrzeni roboczej. Pełni odwrotną rolę niż git add
git rm / git mv
Usuwa / przenosi pliki w repozytorium Działa tak samo jak rm i mv a jednocześnie dodaje zmiany w poczekalni
git checkout <ścieżka>
Przywraca stan pliku z indeksu gita. W praktyce przywraca wprowadzone zmiany. Polecenie może być wykorzystane tylko wówczas gdy plik istnieje już w indeksie.
Przekazywanie zmian do poczekalni

git commit -m „komentarz”
zapisywanie zmian w repozytorium

Przeglądanie historii

git log

```
$ git log
commit 40f1cfc3de739c9a76c248278d4c08cbcf5d09e3 (HEAD -> master)
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Tue Oct 26 10:10:30 2021 +0200

    wprowadzenie zmiany w css/plik1

commit 5a4711600e815caf2710200883e8621e66af1183
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Tue Oct 26 10:01:18 2021 +0200

    wprowadzenie nowego katalogu css z plikami i nowego pliku

commit fbe729404d85ece34dadd8589955aa03c3c740e5
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Tue Oct 26 09:28:37 2021 +0200

    zmiana zawartości pliku dokumentacja2.txt

commit 06c2c8c874edf9c4d6d5847f1f1f06a106c49422
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Mon Oct 25 16:19:00 2021 +0200

    wprowadzenie zmiany w  dokumentacja1.txt

commit eac687e7e65a66574604ad20ad33bf59dec8c75a
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Mon Oct 25 16:11:42 2021 +0200

    wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt
```

flaga **–oneline** podaje skondensowaną wersje historii

```
 git log --oneline
40f1cfc (HEAD -> master) wprowadzenie zmiany w css/plik1
5a47116 wprowadzenie nowego katalogu css z plikami i nowego pliku
fbe7294 zmiana zawartości pliku dokumentacja2.txt
06c2c8c wprowadzenie zmiany w  dokumentacja1.txt
eac687e wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt
```

flaga **–author=** pozwala pokazać zmiany dokonane przez konkretną osobę

```
$ git log --author="lpyzik"
commit 40f1cfc3de739c9a76c248278d4c08cbcf5d09e3 (HEAD -> master)
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Tue Oct 26 10:10:30 2021 +0200

    wprowadzenie zmiany w css/plik1
```

```
commit 5a4711600e815caf2710200883e8621e66af1183
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Tue Oct 26 10:01:18 2021 +0200

    wprowadzenie nowego katalogu css z plikami i nowego pliku

commit fbe729404d85ece34dadd8589955aa03c3c740e5
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Tue Oct 26 09:28:37 2021 +0200

    zmiana zawartości pliku dokumentacja2.txt

commit 06c2c8c874edf9c4d6d5847f1f1f06a106c49422
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Mon Oct 25 16:19:00 2021 +0200

    wprowadzenie zmiany w  dokumentacja1.txt

commit eac687e7e65a66574604ad20ad33bf59dec8c75a
Author: lpyzik <lpyzik@users.noreply.github.com>
Date:   Mon Oct 25 16:11:42 2021 +0200

    wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt1 (master)

$ git log --author="lpyzik" --oneline
40f1cfc (HEAD -> master) wprowadzenie zmiany w css/plik1
5a47116 wprowadzenie nowego katalogu css z plikami i nowego pliku
fbe7294 zmiana zawartości pliku dokumentacja2.txt
06c2c8c wprowadzenie zmiany w  dokumentacja1.txt
eac687e wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt
```

skrócone wersja historii **git shortlog**

```
$ git shortlog
lpyzik (5):
      wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt
      wprowadzenie zmiany w  dokumentacja1.txt
      zmiana zawartości pliku dokumentacja2.txt
      wprowadzenie nowego katalogu css z plikami i nowego pliku
      wprowadzenie zmiany w css/plik1
```

wyświetlanie zmian w odpowiednim formacie **git log –format=**

```
%h -skrócony hash commita
%an – informacja o autorze
%s – komentarz
%cr – kiedy został dodany commit

$ git log --format="%h %an %s (%cr)"
40f1cfc lpyzik wprowadzenie zmiany w css/plik1 (65 minutes ago)
5a47116 lpyzik wprowadzenie nowego katalogu css z plikami i nowego pliku (74 minutes
ago)
fbe7294 lpyzik zmiana zawartości pliku dokumentacja2.txt (2 hours ago)
06c2c8c lpyzik wprowadzenie zmiany w  dokumentacja1.txt (19 hours ago)
eac687e lpyzik wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt (19 hours ago)
```

Komentarze – jak je pisać

Dobre praktyki pisania komentarzy w repozytorium

1. Podziel komentarz na tytuł oraz treść, oddzielając je pustym wierszem
2. Ogranicz długość tytułu komentarza do 50 znaków
3. Tytuł rozpocznij wielką literą
4. Nie dodawaj kropki na końcu tytułu

5. Używaj trybu rozkazującego w treści tytułu np. Dodaj pole e-mail w formularzu zamiast Nowe pole w formularzu
6. Ogranicz długość wiersza treści do 72 znaków
7. Pisząc treść komentarza odpowiadaj na pytania co i dlaczego. Na pytania jak można odpowiedzieć przeglądając zmiany

ad 1.

```
$ git commit css/plik1
[master f431f8a] Modyfikacja pliku będącego w katalogu css
 1 file changed, 2 insertions(+), 1 deletion(-)
```

otwiera się domyślny edytor i wprowadzamy komentarz podzielony na tytuł i treść oddzielony pustym wierszem

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
Modyfikacja pliku będącego w katalogu css

dotyczy pliku komentarzy do stylów

# On branch master
# Changes to be committed:
#       modified:   css/plik1
#
# Untracked files:
#       plik_c
```

uzyskujemy przez to możliwość wyświetlania samych tytułów

```
$ git shortlog
lpyzik (6):
      wprowadzenie plikow dokumentacja1.txt i dokumentacja2.txt
      wprowadzenie zmiany w  dokumentacja1.txt
      zmiana zawartości pliku dokumentacja2.txt
      wprowadzenie nowego katalogu css z plikami i nowego pliku
      wprowadzenie zmiany w css/plik1
      Modyfikacja pliku będącego w katalogu css
```

Czym jest rozgałęzienie - branch

Rozgałęzienia (ang. branches) służą do zrównoleglania pracy. Pracując w zespole, dzięki branchom, każdy może wprowadzać zmiany, nawet w obrębie tego samym pliku, nie przeszkadzając sobie nawzajem Git przechowuje zmiany w repozytorium w postaci snapshoot-ów każdy snapshoot tworzony jest w momencie przekazywania zmian, czyli jeżeli robimy commit. Rozgałęzienie to po prostu kolejny wskaźnik pokazujący konkretny snapshoot. W którym rozgałęzieniu pracujemy pokazuje wskaźnik HEAD.
W przykładzie rozgałęzienia pracujemy nad dokumentacją poszczególnych rozdziałów osobno a następnie po zakończonej pracy scalamy dane rozgałęzienia. Na serwerze produkcyjnym mamy zawsze aktualną wersję dokumentacji z gałęzi master

tworzymy katalog projektu

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~
$ cd projekt2
```

inicjujemy git-a w projekt2

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2
$ git init
Initialized empty Git repository in C:/Users/lucyna/projekt2/.git/
```

tworzymy plik na głównej gałęzi master

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ touch plik1

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git branch
```

Zapisujemy zmiany w master

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)
        plik1

nothing added to commit but untracked files present (use "git add" to track)

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git add .

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git commit -m "utworzenie plik1 na master"
[master (root-commit) bb6a8fc] utworzenie plik1 na master
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 plik1
```

Tworzymy nowe rozgałęzienie o nazwie nowa

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git branch nowa
```

Przechodzimy do tej gałęzi

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git checkout nowa
Switched to branch 'nowa'
```

Tworzymy nowy plik w tej gałęzi i zapisujemy zmiany

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (nowa)
$ touch plik1_na_branch_nowa

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (nowa)
$ git add .

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (nowa)
$ git status
On branch nowa
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   plik1_na_branch_nowa


lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (nowa)
$ git commit -m "nowy plik na branch nowa"
[nowa 94fc033] nowy plik na branch nowa
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 plik1_na_branch_nowa

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (nowa)
$ git status
On branch nowa
nothing to commit, working tree clean
```

Wyświetlamy zawartość tej gałęzi

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (nowa)
$ ls -al
total 16
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:50 ./
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:47 ../
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:51 .git/
-rw-r--r-- 1 lucyna 197121 0 Oct 26 18:47 plik1
-rw-r--r-- 1 lucyna 197121 0 Oct 26 18:50 plik1_na_branch_nowa
```

Zmieniamy gałąź na master i przeglądamy zawartość

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (nowa)
$ git checkout master
Switched to branch 'master'

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ ls -al
total 16
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:52 ./
```

```
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:47 ../
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:52 .git/
-rw-r--r-- 1 lucyna 197121 0 Oct 26 18:47 plik1
```

Scalamy zawartość rozgałęzienia nowa z master i przeglądamy zawartość

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git merge nowa
Updating bb6a8fc..94fc033
Fast-forward
 plik1_na_branch_nowa | 0
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 plik1_na_branch_nowa

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ ls -al
total 16
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:53 ./
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:47 ../
drwxr-xr-x 1 lucyna 197121 0 Oct 26 18:53 .git/
-rw-r--r-- 1 lucyna 197121 0 Oct 26 18:47 plik1
-rw-r--r-- 1 lucyna 197121 0 Oct 26 18:53 plik1_na_branch_nowa
```

Przeglądamy historię

```
lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git log --graph --decorate --all --oneline
* 94fc033 (HEAD -> master, nowa) nowy plik na branch nowa
* bb6a8fc utworzenie plik1 na master

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$ git log --graph --decorate --all
* commit 94fc033c35d3f89b8f5e2873f32a09d6e862e5cd (HEAD -> master, nowa)
| Author: lpyzik <lpyzik@users.noreply.github.com>
| Date:   Tue Oct 26 18:51:40 2021 +0200
|
|     nowy plik na branch nowa
|
* commit bb6a8fc268eea4566b4dffd9b7586ec6aecc89e3
  Author: lpyzik <lpyzik@users.noreply.github.com>
  Date:   Tue Oct 26 18:49:44 2021 +0200

      utworzenie plik1 na master

lucyna@DESKTOP-EBMHFDU MINGW64 ~/projekt2 (master)
$
```