DevOps
2022.09.06

Opracował: Mikołaj Ziółek
ziolek.mikosz@protonmail.com

Git Essentials 2
Basic commands 2
git init 2
git add 2
git commit 3
git log 3
git status 4
git diff 5
Git Architecture 5
Hash Values & HEAD 5
Hash Value 5
H-E-A-D 7

Git Essentials

Basic commands

```
git init
```

*Po zainstalowaniu Git'a i utworzeniu naszego folderu projektowego możemy zainicjalizować nasze repozytorium przy pomocy komendy **git init**.*

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn
$ git init
Initialized empty Git repository in D:/Projects/devop-learn/.git/
```

*Po wykonaniu komendy w folderze projektu zostanie utworzony folder '**.git'**. To w nim odbywa się cała magia **śledzenia zmian**. Tutaj Git przechowuje wszystko, co wie o naszym projekcie.*

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ ls -la
total 8
drwxr-xr-x 1 ziole 197609 0 Sep  6 23:58 ./
drwxr-xr-x 1 ziole 197609 0 Sep  6 23:57 ../
drwxr-xr-x 1 ziole 197609 0 Sep  6 23:58 .git/
```

```
git add
```

*Aby śledzić zmiany, należy najpierw jakichś dokonać. W tym przypadku w folderze projektu umieszczę tę właśnie notatkę i użyję komendy '**git add .'** aby dodać wszystkie (w tym przypadku ten jeden plik) pliki z aktualnego katalogu, które nie są jeszcze śledzone.*

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git add .
```

git commit

Po dodaniu zmian z katalogu, Git jeszcze faktycznie ich nie śledzi, aby to osiągnąć, należy wywołać komendę ***git commit*** *z dodatkową flagą* ***-m "i naszym komentarzem"****.*

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git commit -m "[Initial Commit] Added my learning notes"
[master (root-commit) 7dfde50] [Initial Commit] Added my learning notes
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 DevOps_Notes.pdf
```

git log

Komenda ***git log*** *wylistuje wszystkie commit'y i informację o nich, które zostały utworzone w ramach naszego projektu.*

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git log
commit 7dfde5067f9e1c7d246e9eec1d35791054e238c3 (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date:   Wed Sep 7 00:34:20 2022 +0200

    [Initial Commit] Added my learning notes
```

Idąc od góry, widzimy:

commit - *Unikalny numer identyfikujący nasz commit.*

Author - *Autor commit, pobrany z pliku konfiguracyjnego .gitconfig.*

Date - *Data utworzenia commit.*

message - *Komentarz dodany przez autora w trakcie tworzenia commit komendą git commit -m "komentarz"*

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git log --since=2022-09-01 --until=2022-09-31 --author=mikosz --grep="Init"
commit 7dfde5067f9e1c7d246e9eec1d35791054e238c3
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date:   Wed Sep 7 00:34:20 2022 +0200

    [Initial Commit] Added my learning notes
```

```
git status
```

*Zadaniem komendy **git status** jest wylistowanie wykrytych zmian wewnątrz projektu.*

Przykładowe przypadki użycia:

Brak zmian:

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
nothing to commit, working tree clean
```

Wykrycie nowego pliku:

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Untracked files:
  (use "git add <file>..." to include in what will be committed)
        new_file.txt.txt

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

Zarejestrowanie śledzenia nowego pliku:

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git add new_file.txt.txt

ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   new_file.txt.txt
```

Zarejestrowanie usunięcia nowego pliku:

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   new_file.txt.txt

Changes not staged for commit:
  (use "git add/rm <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        deleted:    new_file.txt.txt
```

git diff

*Podczas wprowadzania zmian w wielu plikach (już istniejących w naszym repozytorium), czasem możemy zapomnieć, lub nawet nie spodziewać się, że jakiś plik zawiera w sobie nowe elementy. W takich chwilach dobrze jest posłużyć się komendą **git diff,** która wylistuje nam istniejące modyfikacje.*

*W tym przykładzie dodałem do repozytorium plik **file.txt**, po czym go zmodyfikowałem:*

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git status
On branch master
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   file.txt

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:   file.txt
```

*Komendą **git diff** mogę podejrzeć co zostało zmienione:*

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git diff
diff --git a/file.txt b/file.txt
index f925f18..0cd5375 100644
--- a/file.txt
+++ b/file.txt
@@ -1 +1 @@
-Hello, this is a file.
\ No newline at end of file
+Hello, this is a modified file.
\ No newline at end of file
```

Jak widać kolorem czerwonym został zaznaczony fragment, który uległ zmianie i kolorem zielonym jak wygląda teraz.

*Domyślnie **git diff** pokaże nam zmiany istniejące pomiędzy **working directory** a **staging area** (więcej o nich później), czyli pomiędzy naszym lokalnym miejscem roboczym a obszarem, w którym git przechowuje poprzednią wersję pliku. Czyli po użyciu komendy **git add file.txt** nie zobaczymy już zmian.*

*W przypadku gdybyśmy chcieli zobaczyć zmiany na już dodanych plikach skorzystamy z komendy **git diff --staged**.*

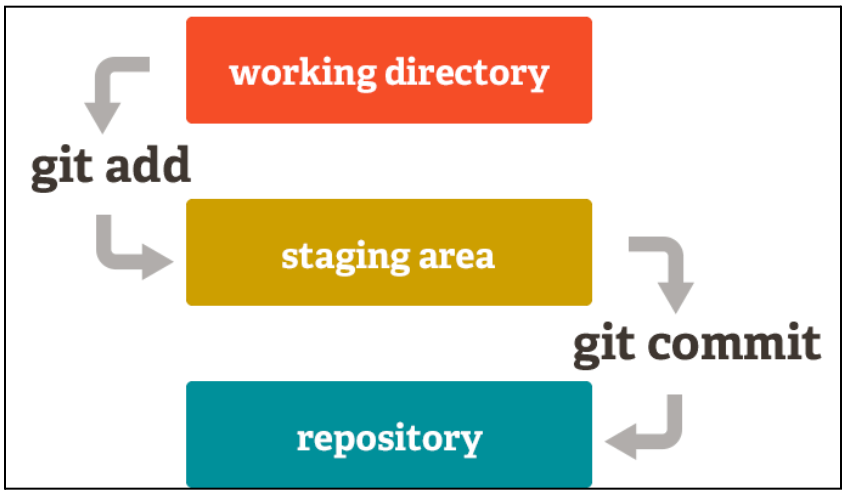
```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git add file.txt

ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn (master)
$ git diff --staged
diff --git a/file.txt b/file.txt
new file mode 100644
index 0000000..3aae3e9
--- /dev/null
+++ b/file.txt
@@ -0,0 +1 @@
+Hello, this is a modified and staged file.
\ No newline at end of file
```

*W tym przypadku ujrzymy zmiany pomiędzy **repository** a **staging area** (tak jakby poziom wyżej).*

Git Architecture

*Git używa tak zwanej **architektury trzech drzew (three-tree architecture)**.*

*To oznacza, że (tak jak inne systemy kontroli) posiada repozytorium i strefę roboczą **i** dodatkowo **staging index** czyli miejsce, do którego odkładane są nasze **commit'y**.*



Dzięki temu możemy dokonać zmian w wielu różnych plikach w naszym obszarze roboczym, a następnie utworzyć zestaw z kilku niezależnych commitów.

Hash Values & HEAD

Hash Value

Git generuje sumę kontrolną (checksum) dla każdego zbioru zmian.
*Jest to wartość **HASH** - liczba generowana przez pobranie danych i wprowadzenie ich do algorytmu matematycznego.*

W folderze .git wewnątrz katalogu naszego projektu znajdziemy plik 'HEAD' - znajdziemy w nim informację o tym na jaką wartość hash, w tym momencie, wskazuje HEAD.

Przy pomocy komendy 'git log' możemy sprawdzić czy faktycznie tak jest.

```
ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/.git (GIT_DIR!)
$ cat HEAD
ref: refs/heads/master

ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/.git (GIT_DIR!)
$ cat refs/heads/master
e4d974efe63ed09495b02a00a68643c8536f0d76

ziole@DESKTOP-3AASAME MINGW64 /d/Projects/devop-learn/.git (GIT_DIR!)
$ git log
commit e4d974efe63ed09495b02a00a68643c8536f0d76 (HEAD -> master)
Author: mikosz08 <46966211+mikosz08@users.noreply.github.com>
Date:   Thu Sep 8 02:23:55 2022 +0200

    Adds new topic - 'Git Architecture'
```

Jak widać wartość w pliku HEAD wskazuję na hash zawarty w moim ostatnim commit'cie.

H-E-A-D

*Jest to zmienna nazywana wskaźnikiem lub referencja do konkretnego **commit'a**.*
HEAD zawsze wskazuje na 'czubek' naszego branch'a (o branch'ach później) w naszym repozytorium. To miejsce w którym zakończyliśmy nasze ostatnie działania.