

Git-hez tartozó segédlet

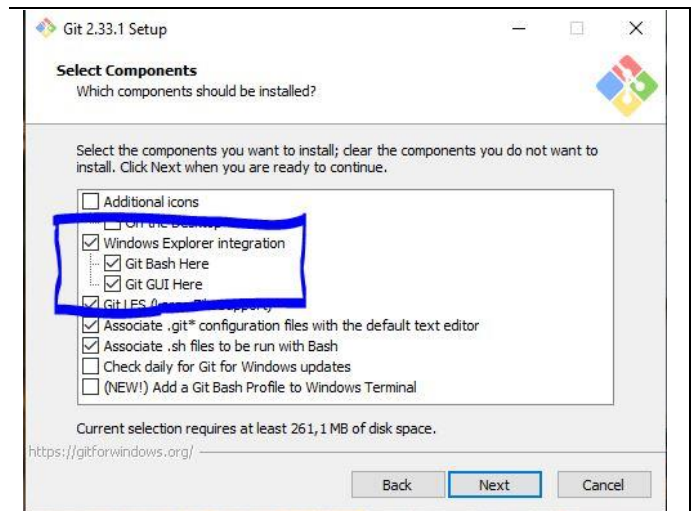
Nos, először is. Amit meg kell tenni az nem más, mint letölteni a Git-et a gépre.

Erről az oldalról tölthetjük le: <https://git-scm.com/downloads>

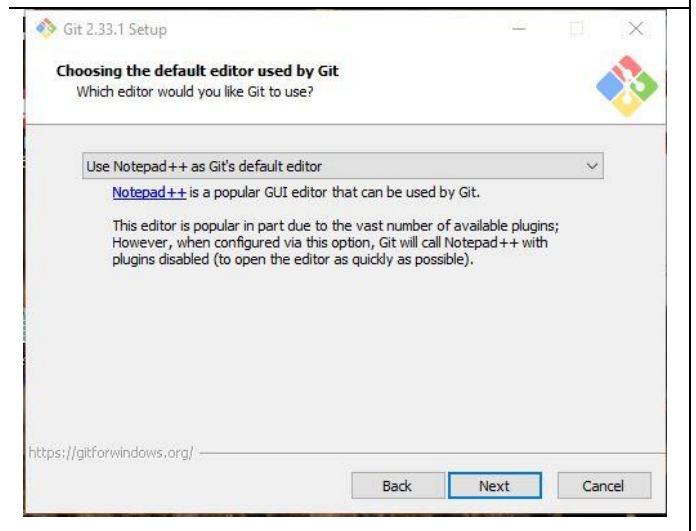
Csak simán a "Download for windows" gomb és automatikusan letölti, amire szükség van (32/64 bit).

Telepítéshez:

➤ Az első dolog, amit le kell csekkolni!



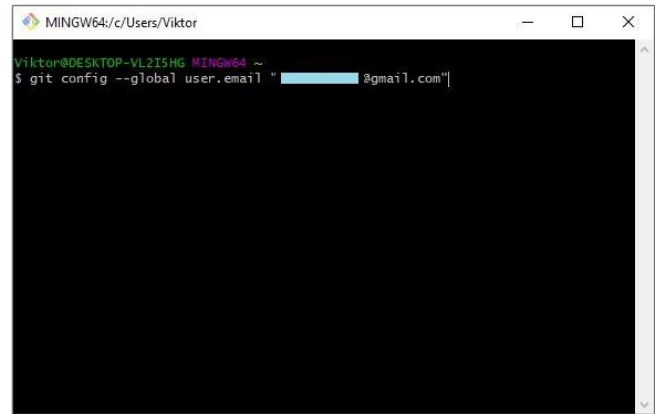
➤ A második kiválasztani az editor-t nekem Notepad++. Persze az, amit éppen szeretsz használni.



➤ A többi meg "next"! Nekünk jelen esetben nem fontos.

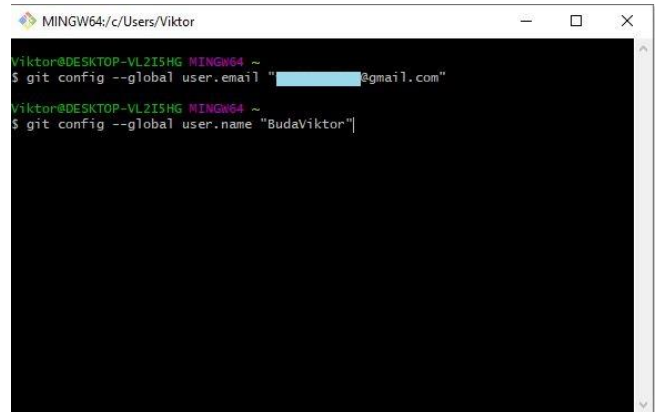
Kezdés Git Bash-ban:

- `git config --global user.email` paranccsal beállítod az email címedet.



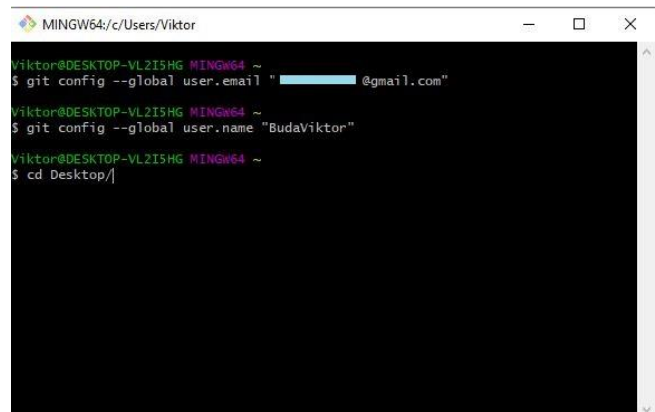
```
MINGW64: c:/Users/Viktor
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.email "[redacted]@gmail.com"
```

- `git config --global user.name` paranccsal pedig a nevedet.



```
MINGW64: c:/Users/Viktor
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.email "[redacted]@gmail.com"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.name "BudaViktor"
```

- Sajnos meg van az rossz tulajdonsága a programnak, hogy minden esetben a `C:\Users\<felhasználó neved>\` mappát veszi alap értelmezetnek. Ezért hogy gyorsan megtaláljam, ahol dolgozom, az asztalra fogok menni.



```
MINGW64: c:/Users/Viktor
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.email "[redacted]@gmail.com"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.name "BudaViktor"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ cd Desktop/
```

- A `git clone https://github.com/buviktor/JMTV.git` paranccsal lehet leklónozni a szervert Repo-t.

```
MINGW64/c/Users/Viktor/Desktop
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.email " [redacted]@gmail.com"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.name "BudaViktor"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ cd Desktop/
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop
$ git clone https://github.com/buviktor/JMTV.git
```

```
MINGW64/c/Users/Viktor/Desktop
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.email " [redacted]@gmail.com"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.name "BudaViktor"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ cd Desktop/
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop
$ git clone https://github.com/buviktor/JMTV.git
Cloning into 'JMTV'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 16 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (2/2), done.
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop
$
```

- Lehetőleg megkérnék mindenkit hogy a `.git` nevű rejtett mappát senki sem módosítsa, mert abból nagy bajok lesznek. Lépünk bele a `cd JMTV/` paranccsal a mappába amit letöltött

```
MINGW64/c/Users/Viktor/Desktop/JMTV
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.email " [redacted]@gmail.com"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.name "BudaViktor"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ cd Desktop/
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop
$ git clone https://github.com/buviktor/JMTV.git
Cloning into 'JMTV'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 16 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (2/2), done.
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop
$ cd JMTV/
```

- következő lépésben beállítjuk a feltöltés és letöltéshez tartozó paramétereket. (lényegében hogy frissíteni tud a saját local Repo-t és a szervert Repo-t is)
- A `git remote add JMTV https://github.com/buviktor/JMTV.git` paranccsal a JMTV megnevezéshez adtuk hozzá a github-os linket, hogy később már ne a linket keljen mindenhová begépelni.

```
MINGW64/c/Users/Viktor/Desktop/JMTV
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.email " [redacted]@gmail.com"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ git config --global user.name "BudaViktor"
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ cd Desktop/
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop
$ git clone https://github.com/buviktor/JMTV.git
Cloning into 'JMTV'...
remote: Enumerating objects: 16, done.
remote: Counting objects: 100% (16/16), done.
remote: Compressing objects: 100% (11/11), done.
remote: Total 16 (delta 2), reused 0 (delta 0), pack-reused 0
Receiving objects: 100% (16/16), done.
Resolving deltas: 100% (2/2), done.
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop
$ cd JMTV/
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git remote add JMTV https://github.com/buviktor/JMTV.git
```

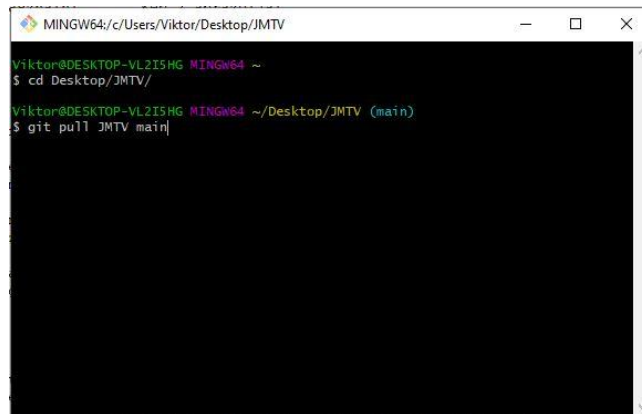
Ezzel kész is vagy, hogy elkezdhesz dolgozni a GitHub-on. Gondolom, felmerült benned az a kérdés miért nem kell a `git init` parancs? Nos, erre egyszerű a válasz, mert mivel leklónoztunk már egy kész Repo-t ezért ő már inicializálva van így felesleges még egyszer ezt megtenni.

Folytatás Git Bash-ban:

Nos, most megmutatom, hogyan és miként lehet egyszerűen fájlokat létrehozni és a már meg lévőket feltölteni. De előbb meg kell, hogy említssem, akár hányszor dolgozni kezdtek van egy parancs, amit jó ha lefuttatok. Mivel sokan dolgoznak egy felületen, és megeshet, hogy valaki módosított valamit egy fájlban, amiben te is dolgozni szeretnél, ha nem kéred le a változtatásokat mielőtt dolgozol, könnyen lehet, hogy ütközés lesz. Erről részletesen fogom majd beszélni, hogy mik ilyenkor a teendők. Reményeim szerint mindenki csak a maga létrehozott fájlokban fog dolgozni egyelőre így nem lehet gondok az ütközésekkel.

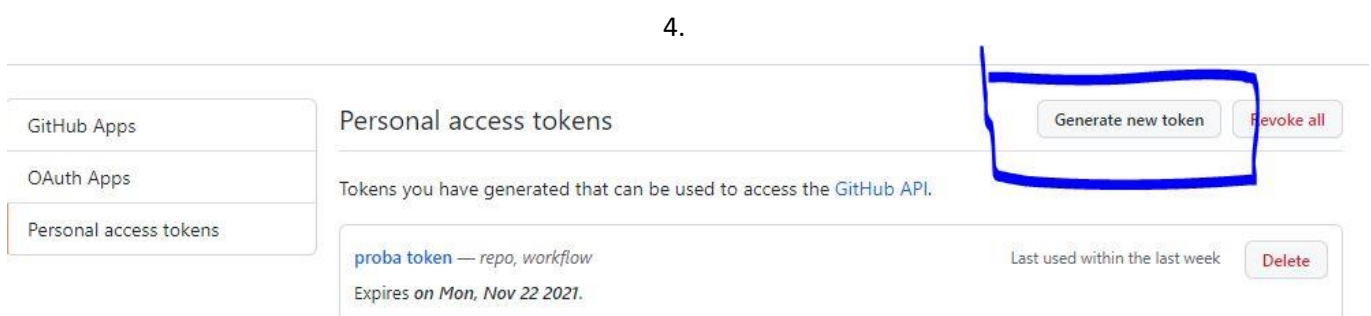
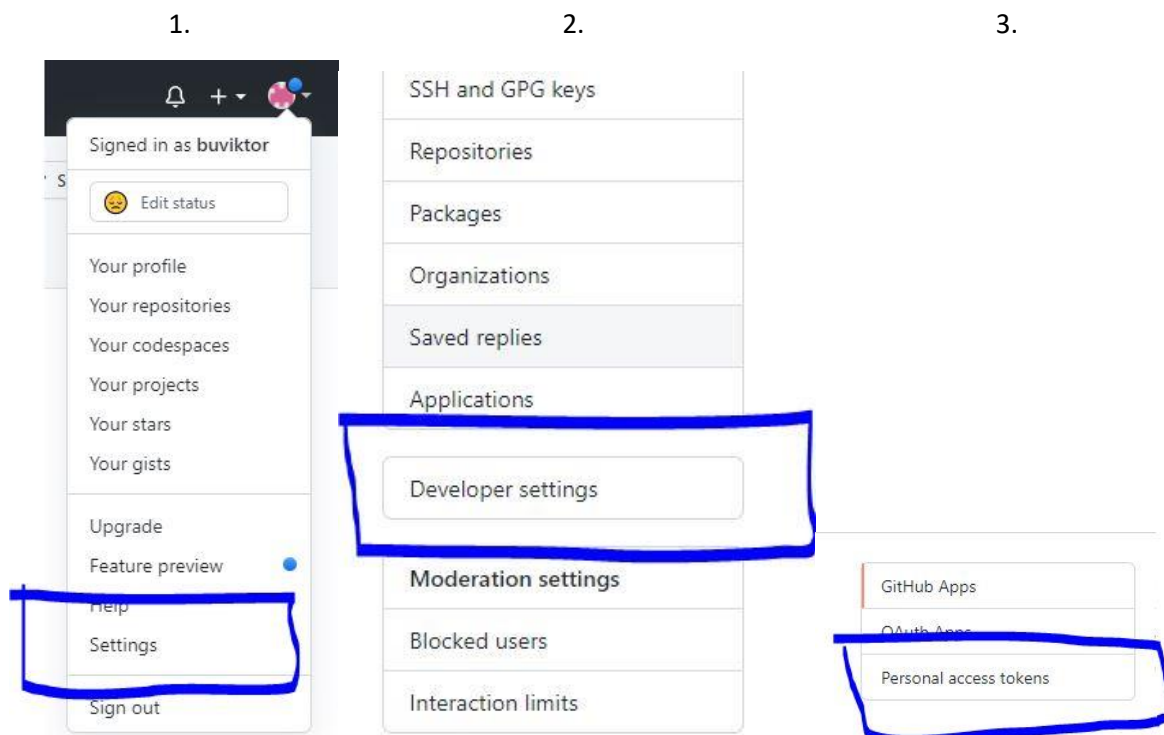
- A parancs, amit le kell futtatni az nem más, mint: `git pull JMTV main`. Frissíti az összes fájlt a server Repo-ból.

Figyelem!! Itt kérni fogja a token azonosítást, amiről már tanárnő is beszélt. Ha esetleg nem akkor a push parancsnál mindenképpen! Lentebb részletezem az igénylését.

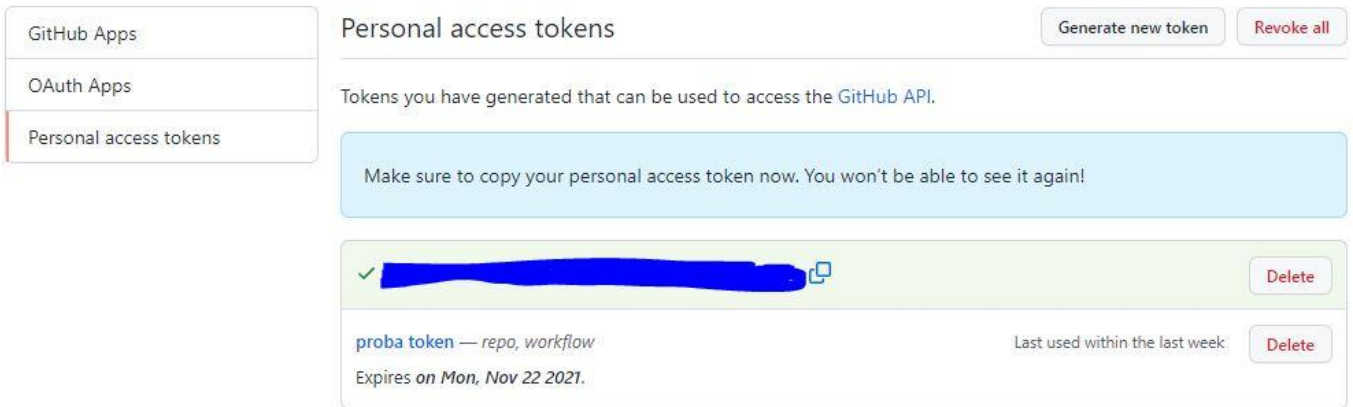


```
MINGW64/c/Users/Viktor/Desktop/JMTV
Viktor@DESKTOP-VL2T5HG MINGW64 ~
$ cd Desktop/JMTV/
Viktor@DESKTOP-VL2T5HG MINGW64 ~/Desktop/JMTV (main)
$ git pull JMTV main
```

- Nos, mi is az a token. Nem más, mint azonosításra használható generált kód.



- Ezután lesz egy pipálás rész. Itt amit elsőnek feltétlenül be kell pipálni az a **repo, workflow** és a **user**!



- És végül megkapod a „kulcsot” az az a token.

- Kérjük le a Repo szerkezetét az **ls** paranccsal. (ez működik a **dir** paranccsal is!)

```
MINGW64/c/Users/Viktor/Desktop/JMTV
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ cd Desktop/JMTV/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git pull JMTV main
From https://github.com/buviktor/JMTV
 * branch      main      -> FETCH_HEAD
 * [new branch] main      -> JMTV/main
Already up to date.

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ ls
Help/  Post-production/  Python/  Web/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ |
```

- Nos, akkor hozzunk létre egy új fájl-t. Elsőnek is belekell menni abba a mappába ahová, szeretnél létrehozni egyet a **cd <mappa neve>** paranccsal.

```
MINGW64/c/Users/Viktor/Desktop/JMTV/Python
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ cd Desktop/JMTV/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git pull JMTV main
From https://github.com/buviktor/JMTV
 * branch      main      -> FETCH_HEAD
 * [new branch] main      -> JMTV/main
Already up to date.

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ ls
Help/  Post-production/  Python/  Web/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ cd Python/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV/Python (main)
$ |
```

- Majd a **touch <fájl név. kiterjesztés>** paranccsal hozzunk is létre egyet.

```
MINGW64/c/Users/Viktor/Desktop/JMTV/Python
Viktor@DESKTOP-VL2I5HG MINGW64 ~
$ cd Desktop/JMTV/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git pull JMTV main
From https://github.com/buviktor/JMTV
 * branch      main      -> FETCH_HEAD
 * [new branch] main      -> JMTV/main
Already up to date.

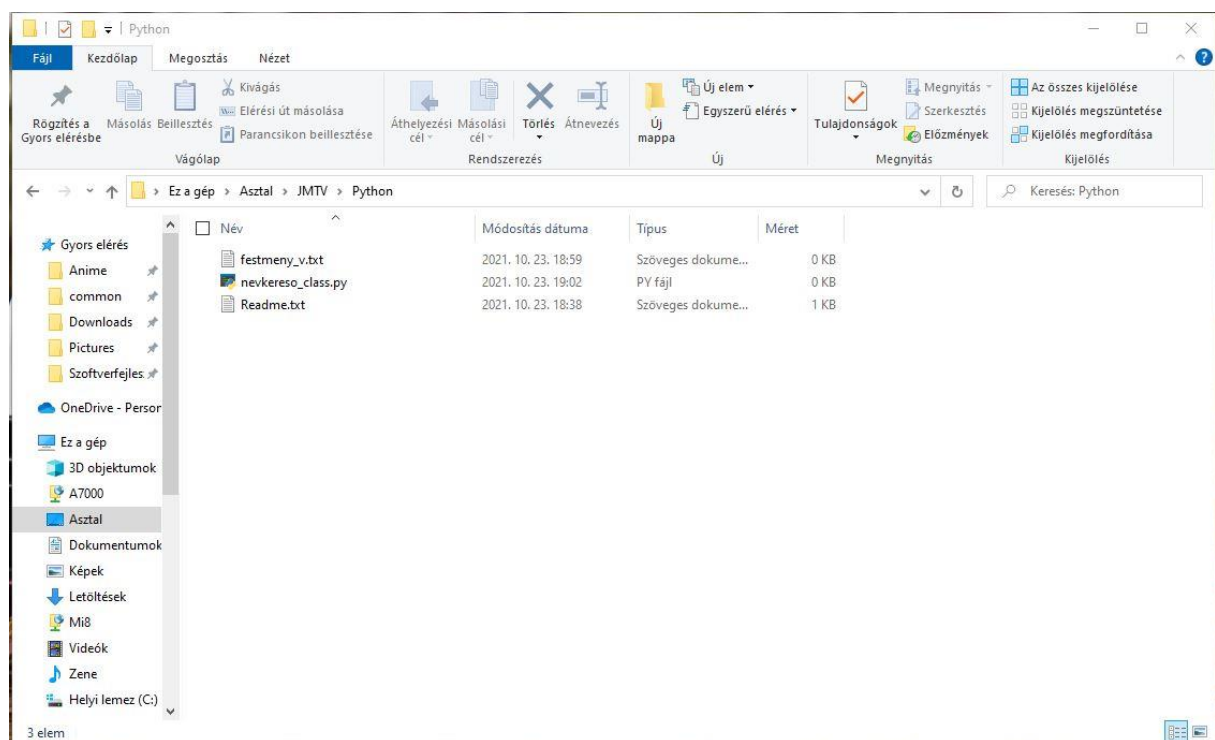
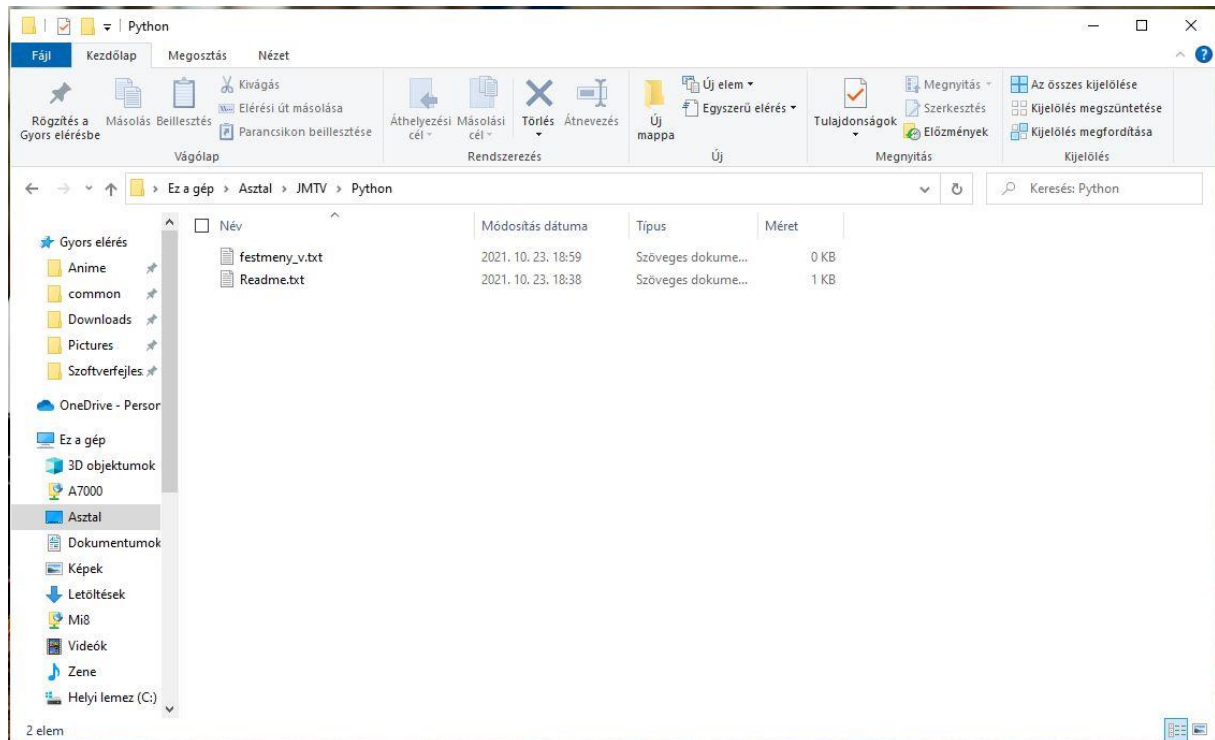
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ ls
Help/  Post-production/  Python/  Web/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ cd Python/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV/Python (main)
$ touch festmeny_v.txt

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV/Python (main)
$ |
```

- A már meglévő fájlokat csak egyszerűen Windows intezőből csak be kell másolni a megfelelő mappában.



- Ha ez megvan, és úgy gondoljuk, hogy kellő mennyiségben javítottunk a fájlokon, ideje feltölteni azt a szerver Repo-ba is. Ezt a következő parancsokkal tudjuk megtenni: elsőnek is vissza kell menni a main részre ez a JMTV mappa. (parancs: `cd ..`), innen könnyebb az eligazodás. Majd lekérjük, hogy mik azok a fájlok, amik nincsenek kezelve a `git status` paranccsal. Itt megfigyelhetjük, hogy megjelent a parancsból létrehozott és a beillesztet fájl is. Valamint leírja a teljes elérési utat ahol található. Következő lépés hogy feltöltsük az úgy nevezett színpadra őket. A parancsa pedig: `git add <fájl>` (jelen esetben van még a fájl előtt a mappa ahol található). Ha ellenőrizzük megint a `git status` paranccsal a következőt kapjuk. Nem kell megijedni, később elfogom magyarázni, hogy pontosan mit és miért látjuk azt, amit. Majd ha mindez megvan commit-olni kell, hogy feltöltés készen legyen, ez lényegében egy kommenttel látja el, amiben megkérné, mindenkit röviden 1-2 szóban foglalja össze mit is csinált. Nem kell regény, mert ha github-ról felmegy valaki és ránéz úgy is látszik mi lett rajta módosítva. A parancsa: `git commit -m "<komment>"`. És az utolsó parancs amivel fel is töltjük a szerver Repo-ba a : `git push JMTV main`.

1.

```
MINGW64/c/Users/Viktor/Desktop/JMTV
$ cd Desktop/JMTV/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git pull JMTV main
From https://github.com/buviktor/JMTV
* branch      main       -> FETCH_HEAD
* [new branch] main       -> JMTV/main
Already up to date.

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ ls
Help/  Post-production/  Python/  Web/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ cd Python/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV/Python (main)
$ touch festmeny_v.txt

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV/Python (main)
$ cd ..

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$
```

2.

```
MINGW64/c/Users/Viktor/Desktop/JMTV
$ ls
Help/  Post-production/  Python/  Web/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ cd Python/

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV/Python (main)
$ touch festmeny_v.txt

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV/Python (main)
$ cd ..

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Python/festmeny_v.txt
    Python/nevkereso_class.py

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

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$
```

3.

```
MINGW64/c/Users/Viktor/Desktop/JMTV
$ touch festmeny_v.txt

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV/Python (main)
$ cd ..

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git status
On branch main
Untracked files:
  (use "git add <file>..." to include in what will be committed)
    Python/festmeny_v.txt
    Python/nevkereso_class.py

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

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git add Python/festmeny_v.txt

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git add Python/nevkereso_class.py

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$
```

4.

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

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git add Python/festmeny_v.txt

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git add Python/nevkereso_class.py

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ get status
bash: get: command not found

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Python/festmeny_v.txt
    new file:   Python/nevkereso_class.py

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$
```

5.

```
MINGW64/c/Users/Viktor/Desktop/JMTV
$ git add Python/nevkereso_class.py

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ get status
bash: get: command not found

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git status
On branch main
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
    new file:   Python/festmeny_v.txt
    new file:   Python/nevkereso_class.py

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git commit -m "Festmeny_v.txt és nevkereso_class.py hozzáadása."
[main c8f846f] Festmeny_v.txt és nevkereso_class.py hozzáadása.
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Python/festmeny_v.txt
 create mode 100644 Python/nevkereso_class.py

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$
```

6.

```
new file:   Python/festmeny_v.txt
new file:   Python/nevkereso_class.py

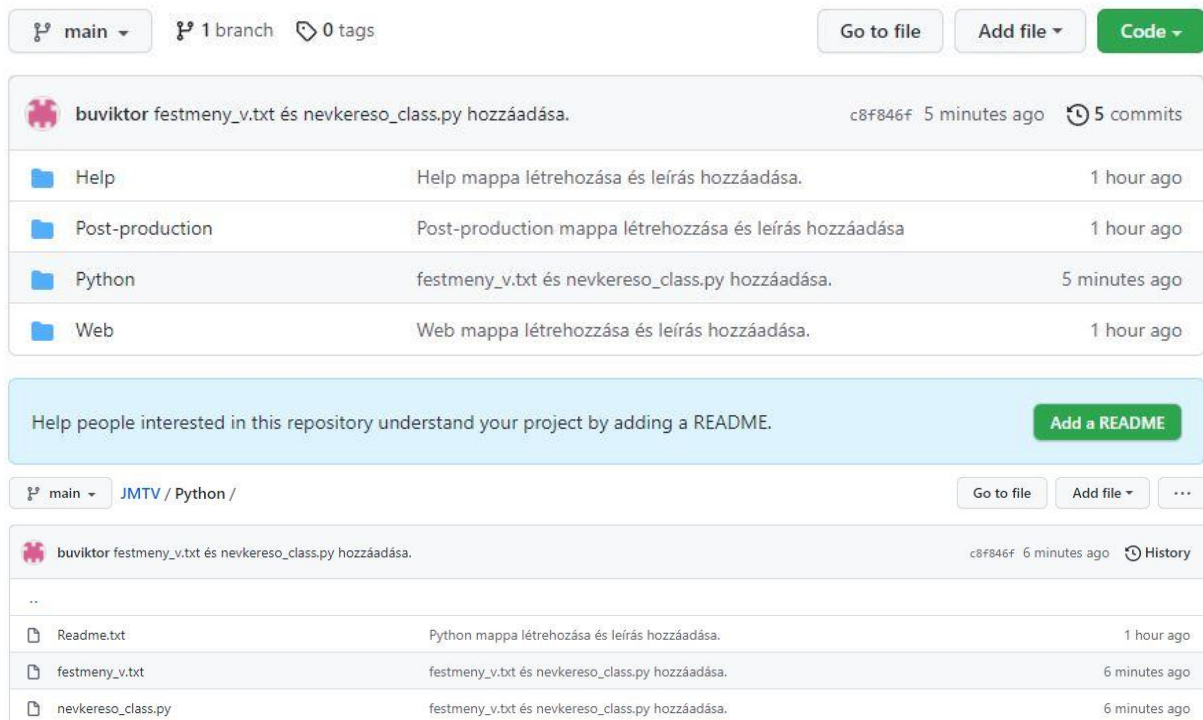
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git commit -m "Festmeny_v.txt és nevkereso_class.py hozzáadása."
[main c8f846f] Festmeny_v.txt és nevkereso_class.py hozzáadása.
 2 files changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 Python/festmeny_v.txt
 create mode 100644 Python/nevkereso_class.py

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$ git push JMTV main
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 4 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (4/4), 478 bytes | 478.00 KiB/s, done.
Total 4 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/buviktor/JMTV.git
   27fc91f..c8f846f  main -> main

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/JMTV (main)
$
```

A képek balról jobbra olvasandók.

- Térjünk át egy kicsit a GitHub-os felületre is. Megmutatom, pontosan mi változik pl.: a commit paranccsal. Arról már volt szó, hogy kommenteket adhatunk egy feltöltéshez. Nos, a probléma vele csak annyi hogy minden feltöltött/módosított fájlhoz ugyan azt a kommentet fűzi.



main 1 branch 0 tags Go to file Add file Code

buviktor festmeny_v.txt és nevkereso_class.py hozzáadása. c8f846f 5 minutes ago 5 commits

File	Commit Message	Time
Help	Help mappa létrehozása és leírás hozzáadása.	1 hour ago
Post-production	Post-production mappa létrehozása és leírás hozzáadása	1 hour ago
Python	festmeny_v.txt és nevkereso_class.py hozzáadása.	5 minutes ago
Web	Web mappa létrehozása és leírás hozzáadása.	1 hour ago

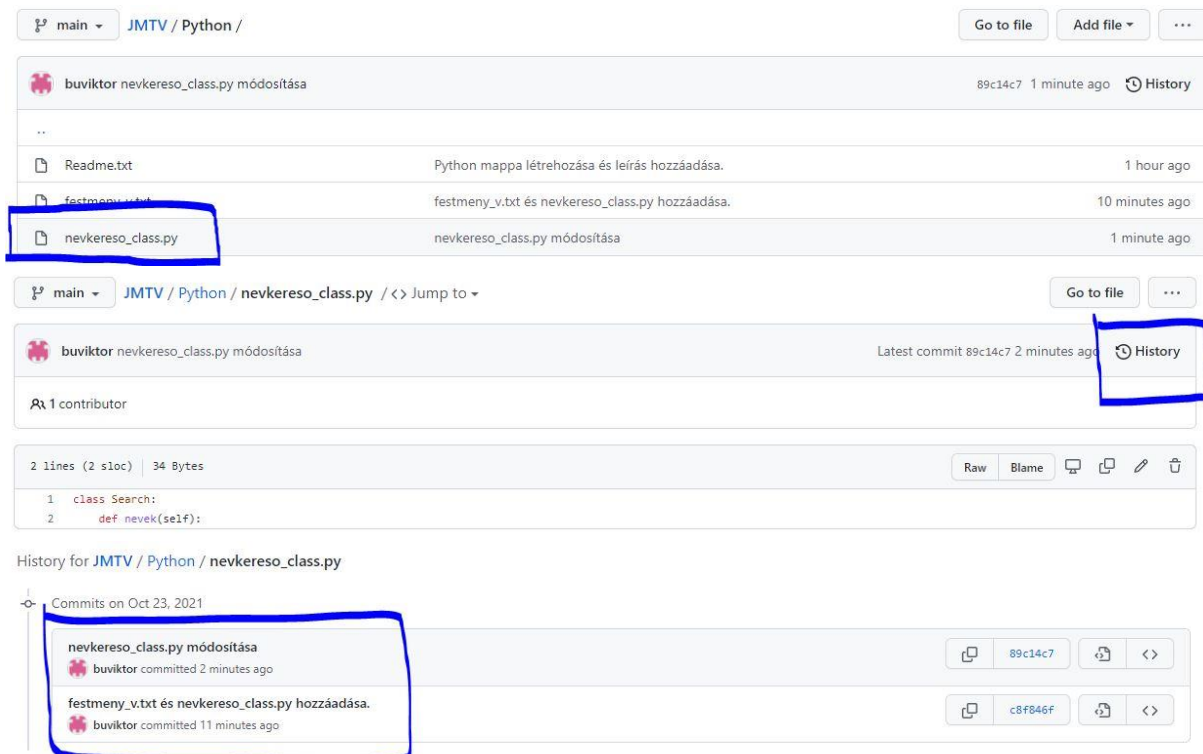
Help people interested in this repository understand your project by adding a README. Add a README

main JMTV / Python / Go to file Add file ...

buviktor festmeny_v.txt és nevkereso_class.py hozzáadása. c8f846f 6 minutes ago History

File	Commit Message	Time
Readme.txt	Python mappa létrehozása és leírás hozzáadása.	1 hour ago
festmeny_v.txt	festmeny_v.txt és nevkereso_class.py hozzáadása.	6 minutes ago
nevkereso_class.py	festmeny_v.txt és nevkereso_class.py hozzáadása.	6 minutes ago

- Hogy miért is kértem, hogy 1-2 szavas legyen csak a komment az a következő miatt:



main JMTV / Python / Go to file Add file ...

buviktor nevkereso_class.py módosítása 89c14c7 1 minute ago History

File	Commit Message	Time
Readme.txt	Python mappa létrehozása és leírás hozzáadása.	1 hour ago
festmeny_v.txt	festmeny_v.txt és nevkereso_class.py hozzáadása.	10 minutes ago
nevkereso_class.py	nevkereso_class.py módosítása	1 minute ago

main JMTV / Python / nevkereso_class.py / <> Jump to Go to file ...

buviktor nevkereso_class.py módosítása Latest commit 89c14c7 2 minutes ago History

1 contributor

2 lines (2 sloc) 34 Bytes Raw Blame

```
1 class Search:
2     def nevek(self):
```

History for JMTV / Python / nevkereso_class.py

Commits on Oct 23, 2021

Commit Message	Commit Hash	Time
nevkereso_class.py módosítása	89c14c7	buviktor committed 2 minutes ago
festmeny_v.txt és nevkereso_class.py hozzáadása.	c8f846f	buviktor committed 11 minutes ago

nevkereso_class.py módosítása

main

buviktor committed 3 minutes ago

1 parent c8f846f commit 89c14c7290cf1cfa8db1e5425ae8a44e02a408b

Showing 1 changed file with 2 additions and 0 deletions.

Split Unified

Python/nevkereso_class.py

```

... @@ -0,0 +1,2 @@
1 + class Search:
2 +     def nevek(self):

```

0 comments on commit 89c14c7

Lock conversation

Write Preview

Leave a comment

Attach files by dragging & dropping, selecting or pasting them.

Comment on this commit

➤ Ugyan is vissza lehet nézni a fájlokon történő módosítást.

Git Bash parancsok:

Ebben részben úgy döntöttem inkább leírok minden parancsot, amit érdemes ismerni. Kibővítve, elmagyarázva, hogy mit miért is csinál. Lássunk is hozzá!

cd .. → Vissza lép egy szinttel/mappával a könyvtár szerkezetben.

cd <mappa> → bele lép egy kiválasztott mappába. Arra kell figyelni csak abban az esetben fog továbblépni, ha szerepel olyan mappa az adott mappában ahol jelenleg vagy. Fontos még megjegyezni, ha tudod az elérési utat mélyebb szintekre/mappákba akkor lehet őket halmozni is. Pl.: cd Desktop/mappa/ .

dir → Kilistázza, a mappa tartalmát ahol tartózkodsz. Windows shell-ben megszokott parancs. Vannak kiegészítő lekérdezései ilyen a **dir -a** :rejtett fájlokat is tartalmazza, **dir -l** :jogokat és létrehozás/módosítás dátumát is tartalmazza.

ls → Szintén kilistázza, a mappa tartalmát ahol tartózkodsz. Linux terminál parancs. Vannak kiegészítő lekérdezései ilyen hasonlóan a Windows-os parancshoz az **ls -a** :rejtett fájlokat is tartalmazza, **ls -l** :jogokat és létrehozás/módosítás dátumát is tartalmazza.

mkdir <mappa neve> → Vele lehet létrehozni mappákat. Fontos megjegyezni, hogy ékezeteknek nincs semmi akadálya.

clear → Ha már túl zsúfolt és átláthatatlan a Bash ablaka, ezzel a paranccsal lehet kitörölni a tartalmát. Viszont a nyilakkal való gyors kitöltés megmarad.

git init → Azt a mappát, amiben tartózkodsz Repository-vá alakítja. Létrehoz benne egy .git rejtett mappát tele rengeteg tartalommal. Ennek a tartalmát idő függvényében egyszer majd részletezem.

git status → A parancs, ami a Repo-hoz tartozó információval szolgál. Itt listázza azokat a fájlokat miket módosítottak, töröltek vagy éppen létrehoztak, de még nincsen a színpadhoz adva és commit-olva.

touch <fájl neve.kiterjesztése> → Létrehozza, az elnevezet és kiterjesztésű üres fájlt.

git add <fájl> → Hozzá adja az adott fájlt a színpadhoz. Ez egy olyan hely ahol sorban állnak a fájlok és várakoznak a feltöltésre (commit-olásra). Innen még vissza lehet őket rakni szerkesztésre, ha még módosítani kívánunk rajta.

git add *.<kiterjesztés> → Minden olyan módosított kiterjesztésű fájlt, amit meghatároztunk neki és abban a mappában van ahol állunk, felrak a színpadra. Pl.: **git add *.txt** : az összes módosított txt fájlt felrakja.

git commit -m "szöveges üzenet" → Véglegesíti a színpadon lévő fájlok verzióját a Repo-ban és mellé egy szöveges üzenetet tesz. Egyetlen gond ezzel az üzenettel, hogy minden fájlhoz ezt az egyet rendeli hozzá. Valamint innen már nem lehet vissza tenni a fájlokat szerkesztésre. Ha módosul valami egy fájlban azt újra színpadra kell majd küldeni mielőtt feltöltve lesznek.

git log → Naplózást listázza ki. Tartalma ki, mikor és mit commit-olt az adott Repo-ban. Mindig legfelülre kerülnek a legutoljára tett változások. Ami még érdekes, hogy minden commit azonosítóját is tartalmazza.

`git remote add <Repo elnevezése> <Repo clone link>` → Hozzá köti a local Repo-hoz egy „szerver” Repo-t. Ahová a helyi fájlokat fel kell tölteni.

`git remote remove <neve>` → Törli/eltávolítja a local Repo-hoz kötött „szerver” Repo-t.

`git push <elnevezett Repo> <Branch neve>` → Minden olyan fájlt ami commit-olva lett feltölt a központi Repo fő vagy mellék könyvtárába (branch-ba). Alapértelmezett Branch név a main elnevezés, ami régen master volt.

`git push -u <elnevezett Repo> <Branch neve>` → Annyiban különbözik az előző parancstól, hogy itt az `-u` kiegészítéssel megjegyeztetjük vele a feltöltési utat. Tehát ha legközelebb fel szeretnénk tölteni a commit-olt fájlokat elég a `git push` parancsot kiadni neki. **Figyelem!!** Ha egy távoli Repo-t hozzá adtunk így a feltöltéshez mindig abba az egy Branch-be fogja a fájlokat feltölteni. Ezt csak abban az esetben célszerű alkalmazni, ha nincsen több Branch létrehozva az adott Repo-hoz.

`git config --global user.email " <regisztrált email cím> "` → Hozzá köti a Github regisztrációt a local Repo-hoz. Lényegében azonosítás miatt van szükség rá, majd későbbiekben a bejelentkezéshez.

`git config --global user.name " <megjelenítendő név> "` → Itt célszerű púpos teve alakban megadni a nevedet és nem nicknevet használni.

`git pull <elnevezett Repo> <Branch neve>` → Minden fájlt frissít/lemásol a hozzá adott távoli Repo Branch-éből (könyvtárából). Természetesen később már elég csak a `git pull` parancs nem kell végig írni, ugyanis magától megjegyzi, hogy honnan szeretné adatokat kérni. Vannak olyan helyzetek, amikor hamarabb lett létrehozva egy local Repo mint egy távoli, azaz vannak olyan fájlok nálad mi nincs a szerveren, de a létre hozásakor még nem létezett a szerver. Ilyenkor hiba kód fogad és ki kell egészíteni a parancsot a következőképpen: `git pull <elnevezett Repo> <Branch neve> --allow -unrelated -histories` .

`git clone <Repo clone link>` → Leklónozza a távoli Repo össze tartalmát. Ez azért jó, mert már egy kész Repo-t kapunk local-nak, így nem kell bajlódni az inicializálással, és netalántán ha módosítottak a konfigurációján. Fontos még megjegyezni, hogy a klónozás a Branch-eket tölti le. Tehát ha te létrehozol egy mappát (pl.:myRepo) a távoli Repo branch neve main, akkor a következőképpen fog megjeleni a könyvtár szerkezet: C:/myRepo/main/(itt lesz a .git mappa).

`git diff <fájl>` → Összehasonlítja a fájlokat az előző commit állapothoz viszonyítva. Ha zöld a módosított tartalom, akkor hozzá lett adva. Ha pedig piros, akkor törölve lett. Ez a parancs csak az éppen módosított fájlokra vonatkozik, amik még nem lettek színpadra küldve!

`git diff HEAD` → Annyiban különbözik az előző parancstól, hogy az összes módosított fájlt kilistázza az előző commit-hoz képest lévő változásaival, ami még nem lett színpadra küldve.

`git diff --staged` → A színpadon lévő fájlok változásait mutatja meg az előző parancshoz hasonlóan. Ha csak egy fájlra vagyunk kíváncsiak, ami a színpadra lett küldve, azt a következő paranccsal nézhetjük meg:

`git diff --cached <fájl>` .

`git checkout -- <fájl>` → Az összes változást törli a fájlban, ami nem volt commit-olva. Ez a parancs a színpadra küldött fájloknál is működik. Nem ajánlatos ezt a parancsot kiadni, mert lényegében teljesen visszaállítja a fájlt az előző commit állapotra.

`git reset <fájl>` → Ha egy fájl a színpadra küldtél, de szeretnél még rajta módosítani úgy, hogy az ne külön verzióba kerüljön, akkor ez a parancs le veszi az adott fájlt a színpadról.

Természetesen rengeteg parancsot tartalmaz maga a program, de első körrel ezekkel fogunk foglalkozni. Az az igazság, hogy én sem néztem, meg a többi parancs miként és hogyan viselkedik.

Konfliktus kezelése:

Utoljára hagytam, mert elvileg nekünk nem lesz rá szükség, de jó ha tudjátok mit kell csinálni egy ilyen helyzetben. Nem sokban különbözik a Github-os kivétel kezelésnél, annyi hogy itt nincs pull request hanem egyből maga a fájlban történik a kezelése. De mutatom képekkel is, hátha úgy egyértelműbb.


- Először is lemásoljuk a fájlokat a *git push* paranccsal.

```
MINGW64/c/Users/Viktor/Desktop/proba
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ git pull origin main
From https://github.com/buviktor/proba
 * branch      main       -> FETCH_HEAD
Already up to date.

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$
```

- Ami a Github-on ezt tartalmazza:

main ▾ proba / readmi.txt Go to file ...

 buviktor Update readmi.txt

Latest commit f1be0df in 2 minutes [History](#)

1 contributor

1 lines (1 sloc) | 7 Bytes

Raw

Blame

1 valami

- A fájl pedig:

readmi.txt - Jegyzetfőb

Fájl Szerkesztés Formátum Nézet Súgó

valami

Abban megegyezhetünk, hogy semmi változás nincs benne. De ha változás van a távoli Repo-ban a következő lesz a helyzet:

- Látjuk, hogy teljesen más a Github-on a fájl tartalma:

main ▾ proba / readmi.txt Go to file ...

 buviktor Update readmi.txt

Latest commit 0803e6d in 4 minutes [History](#)

1 contributor

3 lines (3 sloc) | 68 Bytes

Raw

Blame

1 valami

2 ide került még egy sor!

3 ide meg egy számsor is: 45454545!

- Nézzük meg mi történik, ha változtatunk a local Repo-ban a fájlban és megpróbáljuk feltölteni a távoli Repo-ba:

readmi.txt - Jegyzetfőb

Fájl Szerkesztés Formátum Nézet Súgó

valami

most ez van itt!

meg ez: 1231123!

- Láthatjátok, figyelmeztet bennünket, hogy valami hiba van. Felhívja a figyelmet arra, hogy már módosítva lettek az állományok, amiket felszeretnél tölteni módosítva. Javasolja az újbóli frissítést a legfrissebb fájl verziók beszerzése miatt.

```
MINGW64/c/Users/Viktor/Desktop/proba
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ git pull origin main
From https://github.com/buviktor/proba
 * branch      main       -> FETCH_HEAD
Already up to date.

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ git status
On branch main
Your branch is up to date with 'origin/main'.

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:   readmi.txt

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

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ git add readmi.txt

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ git commit -m "local változás"
[main 9e3d79e] local változás
1 file changed, 2 insertions(+)

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ git push origin main
To https://github.com/buviktor/proba.git
 ! [rejected]        main -> main (fetch first)
error: Failed to push some refs to 'https://github.com/buviktor/proba.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$
```

- Végre hajtva a következőt adta vissza. Frissítette az állományt majd talált 1 eltérést, amit ő megpróbál össze fésülni (Auto-merging), de nem sikerül. Ezért arra kér, hogy javítsuk ki a konfliktus és commit-oljuk az eredményt.

```
MINGW64/c/Users/Viktor/Desktop/proba
To https://github.com/buviktor/proba.git
 ! [rejected]        main -> main (fetch first)
error: Failed to push some refs to 'https://github.com/buviktor/proba.git'
hint: Updates were rejected because the remote contains work that you do
hint: not have locally. This is usually caused by another repository pushing
hint: to the same ref. You may want to first integrate the remote changes
hint: (e.g., 'git pull ...') before pushing again.
hint: See the 'Note about fast-forwards' in 'git push --help' for details.

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ git pull origin main
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 738 bytes | 33.00 KiB/s, done.
From https://github.com/buviktor/proba
 * branch      main       -> FETCH_HEAD
 * f1be0df..0803e6d main   -> origin/main
Auto-merging readmi.txt
CONFLICT (content): Merge conflict in readmi.txt
Automatic merge failed; fix conflicts and then commit the result.

Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main|MERGING)
$
```

Abban mind egyetértünk, hogy Github-on nem változott semmi, hiszen nem engedett feltölteni, de mi van a local fájlal? Egyértelműen jelzi, mi a pontos hiba. Mettől meddig kell javítani a tartalom. Még ami változik a pull request-től az az, hogy itt megjelenik a commit azonosítója is. HEAD a kezdődő sor az azonosító az utolsó.

```
readmi.txt - Jegyzetfőmb
Fájl Szerkesztés Formátum Nézet Súgó
valami
<<<<<<< HEAD
most ez van itt!
meg ez: 1231123!
=====
ide került még egy sor!
ide meg egy számsor is: 45454545!
>>>>>> 0803e6d606593ba0bb0f154679592c2cae876f3b
```

- Most javítsuk ki a hibákat és próbáljuk meg feltölteni a fájlt. Annyit tettem, hogy kivettem a markereket és így már javítva az ütközés. A **git status** parancs alatt lehet látni, van olyan fájl, ami még nincs kezelve.

```
readmi.txt - Jegyzetfőmb
Fájl Szerkesztés Formátum Nézet Súgó
valami

most ez van itt!
meg ez: 1231123!

ide került még egy sor!
ide meg egy számsor is: 45454545!
```

```

MINGW64/c:/Users/Viktor/Desktop/proba
Automatic merge failed; fix conflicts and then commit the result.
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main|MERGING)
$ get status
bash: get: command not found
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main|MERGING)
$ git status
On branch main
Your branch and 'origin/main' have diverged,
and have 1 and 1 different commits each, respectively.
(use "git pull" to merge the remote branch into yours)

You have unmerged paths.
  (fix conflicts and run "git commit")
  (use "git merge --abort" to abort the merge)

Unmerged paths:
  (use "git add <file>..." to mark resolution)
    both modified:   readmi.txt

no changes added to commit (use "git add" and/or "git commit -a")
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main|MERGING)
$ |

```

```

MINGW64/c:/Users/Viktor/Desktop/proba
(use "git add <file>..." to mark resolution)
    both modified:   readmi.txt


no changes added to commit (use "git add" and/or "git commit -a")
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main|MERGING)
$ git add readmi.txt
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main|MERGING)
$ git commit -m "javitva"
[main 2cf990f] javitva
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ git push origin main
Enumerating objects: 10, done.
Counting objects: 100% (10/10), done.
Delta compression using up to 4 threads
Compressing objects: 100% (5/5), done.
Writing objects: 100% (6/6), 746 bytes | 373.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/buviktor/proba.git
 0803e6d..2cf990f  main -> main
Viktor@DESKTOP-VL2I5HG MINGW64 ~/Desktop/proba (main)
$ |

```

Konfliktus kezelése után egyből hiba nélkül föltöltötte a fájlt. Ha esetleg benne hagyd a markereket, akkor is figyelmeztet arra, hogy vannak még olyan fájl, ami nem lett javítva.

➤ De lessünk vissza egy kicsit Github-ra mi is változott. Bekerült szépen minden változtatás.

main ▾
proba / readmi.txt
Go to file
...


buviktor javitva
Latest commit 2cf990f 2 minutes ago
History

1 contributor

8 lines (5 sloc) | 105 Bytes
Raw
Blame
📄
📋
✎
🗑

```

1 valami
2
3 most ez van itt!
4 meg ez: 1231123!
5
6 ide került még egy sor!
7 ide meg egy számsor is: 45454545!
8

```

Nagyjából ennyi lenne egy konfliktus kezelés, ha ütközés van egy adott fájlban.

Köszönöm a figyelmet!