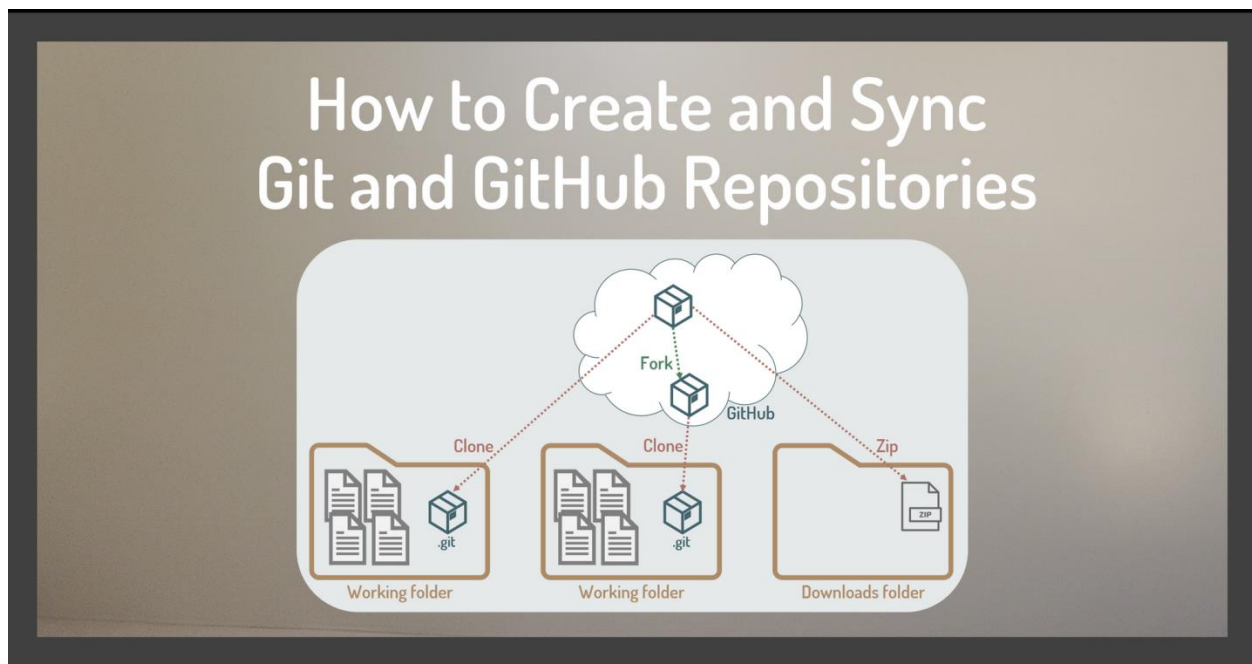


Archivácia súborov a dokumentov prostredníctvom VCS Git a GitHub

<https://www.freecodecamp.org/news/create-and-sync-git-and-github-repositories/>

Pri práci na programovaní je nevyhnutnou súčasťou vašich každodenných činností využívanie existujúcich kódov ktoré už niekto vytvoril alebo používanie dokumentov a súborov ktoré súvisia s touto činnosťou. Tak isto je veľmi často potrebné naše dokumenty a súbory zdieľať s niekým iným a archivovať si ich aby boli prístupné nielen lokálne ale aj vzdialene z ľubovoľného miesta a v ľubovoľnom čase. Na tieto účely sa veľmi dobre hodí veľmi rozšírený tzv. **VCS (Version Control Systems)** nazývané Git resp GitHub¹.



Tento systém používa vo svojej architektúre úložisko (**repository**) pričom sa rozlišuje tzv. lokálne úložisko Git a tzv. vzdialené úložisko GitHub.

- Zatiaľ čo s **lokálnym úložiskom** pracujeme na svojom počítači, notebooku, tablete alebo mobile tak tu pracujeme na vlastnej kópii súboru alebo projektu. Zmeny tu vykonávame a testujeme nezávisle od ostatných ešte pred tým, ako ich odošlete do vzdialeného úložiska, aby ich tu mali prístupní ostatní ľudia, ktorí ich takto môžu prevziať, skontrolovať, použiť a ev. zlúčiť do celku.
- **Vzdialené úložisko** používate na ukladanie mimo našej vlastnej lokality a na zdieľanie súborov a projektu s ostatnými osobami v tíme resp. vo verejnosti.
- Ak chceme aby na oboch úložiskách boli uložené rovnaké dokumenty a súbory musíme tieto dva úložiská najprv vytvoriť a potom podľa potreby synchronizovať

¹ <https://git-scm.com/>

V praxi rozlišujeme dva scenáre ako pracovať s takýmto VCS:

- Prvým je že **vytvoríme najprv lokálne úložisko**: Najprv vytvoríte lokálny archív a odošlete svoje súbory do tohto úložiska. Vzdialené úložisko si vytvoríte neskôr. Keď už obe úložiská existujú, môžeme ich podľa nášho uváženia synchronizovať. Tento scenár je bežný pre prípad, ak sme projekt začali sami a až neskôr ho budeme zdieľať s ostatnými členmi tímu alebo verejnosťou.
- Druhým je že **vytvoríme najprv prepojenie na vzdialené úložisko**: Pri tomto scenári už existuje vzdialený archív a my chcete so súbormi ktoré sa tu nachádzajú pracovať. Toto vzdialené úložisko následne použijeme na vytvorenie svojho lokálneho úložiska, aby ste tu mohli lokálne používať stiahnuté súbory a vykonávať na nich zmeny resp. ich používať a testovať. Aj v tomto prípade keď už obe úložiská existujú, môžeme ich podľa nášho uváženia navzájom synchronizovať. Treba spomenúť že tento scenár je bežný, ak od začiatku pracujeme na tímovom alebo tzv. open source² projekte, ktorý má už napr. na GitHub vytvorené existujúce úložisko.

V ďalšom texte si zodpovieme nasledovné otázky:

1. [Čo znamenajú Git a GitHub úložisko?](#)
2. [Ako nainštalovať Git a GitHub ?](#)
3. [Scenár 1: Vytvorenie lokálneho úložiska ako prvého](#)
4. [Ako vytvoriť lokálne úložisko ?](#)
5. [Ako odovzdať súbory do lokálneho úložiska ?](#)
6. [Ako vytvoriť vzdialené úložisko GitHub ?](#)
7. [Ako povedať Gitu o GitHub ?](#)
8. [Ako synchronizovať lokálne a vzdialené úložiská ?](#)
9. [Scenár 2: Vytvorenie vzdialeného úložiska ako prvého](#)
10. [Ako klonovať úložisko ?](#)
11. [Zhrnutie záverom](#)

Skôr než sa budeme zaoberať našimi dvomi scenármi vysvetlíme si niektoré pojmy.

² <https://opensource.com/resources/what-open-source>

Čo znamenajú pojmy Git a GitHub a úložisko ?

Úložisko (angl. **repository**), alebo skrátené „repo“, ukladá a sleduje verzie vašich súborov a projektov . Keď vykonáte zmeny v týchto súboroch, vložíte (alebo skopírujete) tieto súbory do úložiska na úschovu. Úložisko uchováva zoznam všetkých vašich potvrdených (angl. **commit**-nutých) zmien, ktoré takto evidujú tzv. **históriu potvrdzovania** (angl. **commit history**) .

Git je populárna a široko používaná program pre správu verzií (VCS) pri ktorej používa na vytváranie a prácu s verziami **repository** (úložiská). Git beží lokálne na vašom počítači a Git si môžete [tu](#) stiahnuť, nainštalovať a používať na akejkolvek platforme a bez akýchkoľvek nákladov a poplatkov. S Git-om si vytvoríte vaše lokálne úložisko v niektorom z vašich pracovných priestorov ktorý používate pre odkladanie vašich súborov alebo vášho projektu. Git si potom v tomto priestore ukladá aj históriu zmien na jednotlivých súboroch a všetky údaje ktoré sa viažu k súborom odovzdávaným do tohto priestoru resp. adresára (angl. file folder). Ako veľmi názorná forma úvodu do problematiky Git-u vám dobre poslúži aj vzhľadnutie tohto videa: https://youtu.be/hfOeWgWp_E

GitHub je na rozdiel od Git-u webová stránka, ktorá hostuje vzdialené úložiská na internete. Základný účet na GitHub-e je tiež zadarmo. GitHub sa používa na vytvorenie vzdialeného úložiska pre verejný alebo regulovaný prístup záujemcov k súborom a projektom z ľubovoľného miesta a v ľubovoľnom čase. Pomocou vzdialeného úložiska môžete ukladať svoje súbory a kódy mimo vášho pracoviska, spolupracovať s ostatnými, pracovať na školských, firemných alebo open source projektoch a predviesť svoje portfólio potenciálnym zamestnávateľom. Obdobne ako veľmi názorná forma zoznámenia sa s problematikou GitHub-u vám veľmi dobre poslúži aj vzhľadnutie tohto videa: <https://youtu.be/Uf2LLF7UKMw>

Pred tým však ako prejdeme k našim scenárom musíme si nainštalovať Git a vytvoriť účet na GitHub !

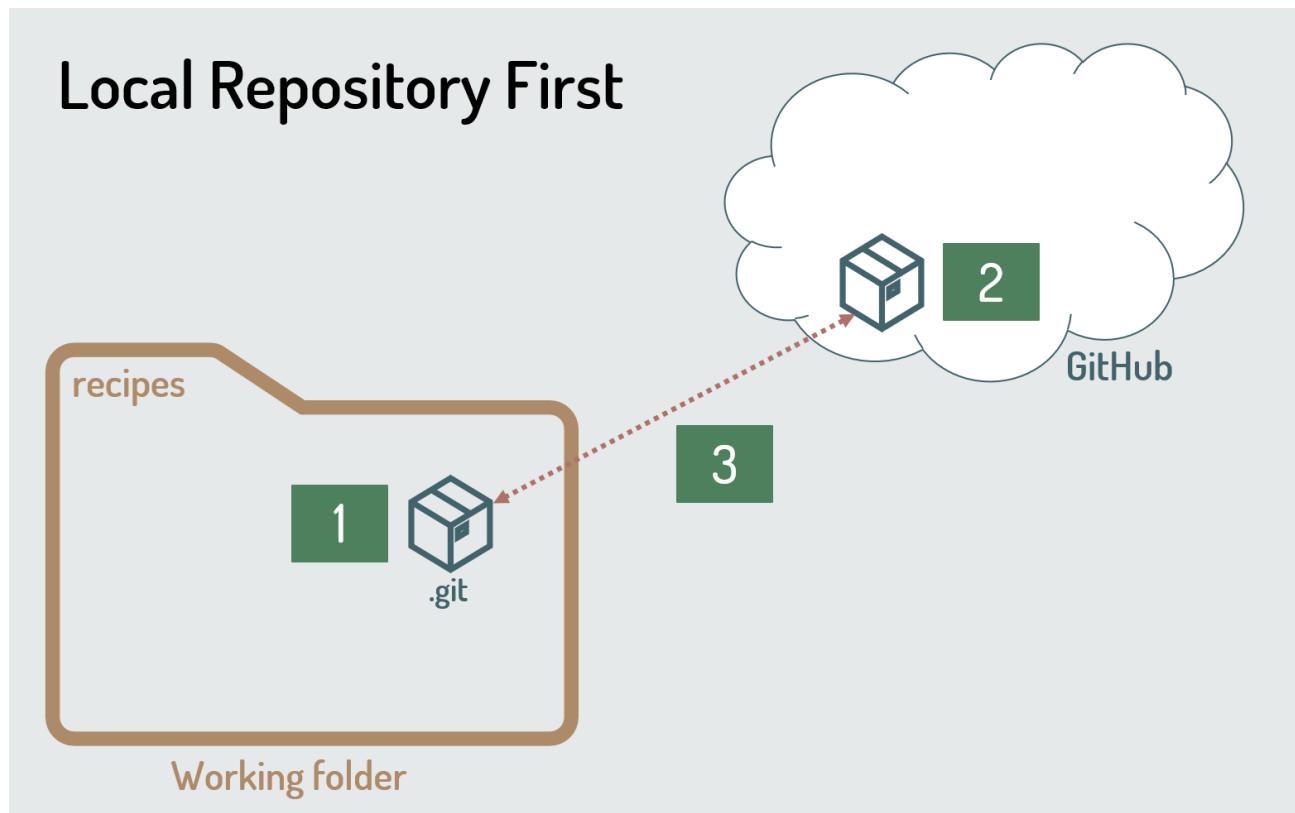
Ako nainštalovať Git a GitHub ?

Ak už máte tieto nástroje nainštalované, pokojne preskočte na ďalšiu časť, alebo si prečítajte tento článok aj bez toho aby tieto nástroje ihneď použili pri kódovaní alebo vytváraní nejakého súboru. Ak by ste sa chceli prepracovať k zvládnutiu týchto nástrojov pomocou príkladov uvedených v tomto návode, je potrebné vykonať niekoľko prípravných krokov. Najprv si do svojho počítača **nainštalujte Git**, ak ho tam ešte nemáte. Podrobné pokyny na inštaláciu Gitu nájdete v druhej časti tohto videa: <https://youtu.be/Xzy-hSdNGOI> Potom si **vytvorte na GitHub-e účet**, ak ho tam ešte nemáte. Toto video poskytuje podrobné pokyny na vytvorenie účtu GitHub: <https://youtu.be/GrWL62j3gTU> Nakoniec **vytvorte pracovný priestor** v počítači (ja som ho pomenoval **recipes** a vytvoril som ho vo Foldry Documents). Potom si napríklad v tomto priestore **vytvorte dva textové súbory** s názvami: **file1.txt** a **file2.txt**. Aj keď tu pre demonštrovanie postupu používame iba textové súbory, tak proces synchronizácie ktorý tu bude popisovaný je rovnaký s akýmkoľvek iným typom súboru. Teraz už sme pripravení zoznámiť sa s našimi dvoma scenármi v poradí najprv lokálne úložisko a potom vzdialené.

Scenár 1: Vytvorenie lokálneho úložiska ako prvého

Povedzme, že si chcete napr. vytvoriť webovú stránku receptov (alebo nejakú aplikáciu) na zhromažďovanie a správu receptov. Vytvoríte si local repository, aby sme tu mohli jednoducho sledovať zmeny v súboroch nášho projektu. Neskôr sa rozhodneme, že chceme si vytvoriť aj remote repository na uchovávanie kópií vašich súborov mimo nášho počítača a na zdieľanie nášho projektu

s ostatnými ľuďmi. Keď budeme mať vzdialený archív, budeme chcieť, aby bol náš lokálny archív (local repository) synchronizovaný s týmto vzdialeným úložiskom a vytvárala sa tu takto jeho aktualizácia. Tento scenár je znázornený na obrázku 1:



Obrázok 1. Vytvorenie lokálneho úložiska ako prvého

Jednotlivé čísla na obrázku 1 znamenajú nasledovné :

1. Najprv si vytvoríme miestne úložisko ktoré je tu zobrazené ako krabica.
2. Vzdialené úložisko si vytvoríme niekedy neskôr pomocou GitHub.
3. Tieto dva archívy budú navzájom synchronizované.

Prejdime si tieto kroky podrobne.

1. Ako vytvoriť lokálne úložisko ?

Keďže už sme si vopred pripravili pracovný priečinok **recipes** a vytvorili dva súbory, ktoré sú súčasťou nášho projektu alebo ich jednoducho chceme archivovať môžeme pristúpiť k vytvoreniu **local repository**. Za týmto účelom potrebujeme otvoriť terminalové okno nejakého príkazového interpretera (Git Bash, PoweShell, CMD CoMmanD Prompt atď.) a pomocou príkazov (dir, cd .., cd folder atď.)³ prejsť do nášho pracovného priečinka **recipes**. Potom inicializujeme vytvorenie nového lokálneho úložiska Git pomocou nasledujúceho príkazu:

```
git init
```

³ <https://www.digitalcitizen.life/command-prompt-how-use-basic-commands/>

Tento príkaz inicializuje nové úložisko Git v aktuálnom priečinku. Proces inicializácie vytvorí v priečinku nášho projektu **recipes** priečinok **.git**, v ktorom budú uložené súbory a údaje týkajúce sa nášho local repository.

Tým že tento priečinok začína bodkou (**.git**) ide o priečinok ktorý je deklarovaný ako skrytý. V systéme Windows na zobrazenie takýchto priečinkov vo File Explorer-y (Prehliadači súborov) je potrebné v hlavičke pri voľbe View začiarknúť voľbu, Hidden items. Keď sme sa presvedčili že máme vytvorené **local repository** môžeme prísť k ďalšiemu kroku a odovzdať naše dva súbory (**file1.txt** a **file2.txt**) do tohto úložiska.

2. Ako odovzdať súbory do lokálneho úložiska ?

Spravidla do local repository posielame naše súbory vždy vtedy keď vytvárame nové súbory, meníme názvy a hlavne obsahy existujúcich súborov alebo odstraňujeme súbory. Vytvorenie local repository zaisťuje to, že úložisko na základe našich podnetov vie sledovať aktuálny stav nášho projektu alebo súborov. Odovzdanie súborov do lokálneho úložiska vyžaduje dva kroky:

1. Pridanie súborov - add

2. Prepojenie suborov - commit

Najprv musíme súbory do pracovnej oblasti Git pridať príkazom:

```
git add
```

Gitu týmto povieme, ktoré súbory chceme zahrnúť do commit-u a pridať ich tým do local repository. Príkazom add ich pridávame iba do tzv. prípravnej oblasti (**staging area**) a staging nám ešte umežňuje pred vlastným uložením selektívne si vybrať, ktoré zmeny chceme zahrnúť do prepojenia-commitu.

Použitím **.** (bodky) pridáme do staging oblasti všetky súbory nachádzajúce sa v pracovnom priečinku **recipes** (a jeho podpriečinkoch). Ak chceme pridať iba konkrétne súbory, je potrebné ich namiesto bodky menovite uviesť. V druhom kroku už príkazom commit presunieme pripravené súbory do našej local repository:

```
git commit -m "Komentár čo sa so súborom robilo"
```

Tento príkaz už odovzdá súbory z pracovnej oblasti (staging area) Git-u do nášho local repository.

Parameter **-m** slúži ako informácia pre commit ze za nim nasleduje text ktorý v sebe obsahuje správu (message) o tom čo sa súborom urobilo. Tato správa má príjemcovi a čitateľovi tohto súboru povedať jasnú informáciu popisujúcu zmeny, ktoré sa so súborom vykonali aby na to mohol čitateľ nadviazať napr. pri riešení svojej čiastkovej úlohy. Ďalšie informácie o definovaní dobre formulovaných správ pre commit nájdete aj v tomto videu: <https://youtu.be/9UImPCMZ4tc> Po týchto krokoch sa na našom termináli napr. CMD zobrazí nasledujúce:

```
Command Prompt
C:\Users\Deborah\Documents\recipes>git init
Initialized empty Git repository in C:/Users/Deborah/Documents/recipes/.git/

C:\Users\Deborah\Documents\recipes>git add .

C:\Users\Deborah\Documents\recipes>git commit -m "initial commit"
[main (root-commit) e7e51ba] initial commit
2 files changed, 2 insertions(+)
create mode 100644 file1.txt
create mode 100644 file2.txt

C:\Users\Deborah\Documents\recipes>
```

Obrázok 2. Vytvorenie lokálneho úložiska a nasledné uskutočnenie commit-u súborov projektu.

Príkaz `init` nám ako reakciu zobrazil správu ktorá nám hovorí, že vytvoril prázdne úložisko Git v adresári ktorého cestu zobrazí tiež. Príkaz `add` nám žiadny výstup neposkytuje a príkaz `commit` zobrazí vetvu (`branch`), na ktorej sa práve pri zavoľaní príkazu `commit` nachádzame (tomto príklade je to `main`). Tiež je tu niekoľko niekoľko prvých znakov identifikátora operácie `id` a správa ktorá bola odovzdaná `commitu`. Potom nasleduje zobrazenie zoznamu zmenených súborov. V tomto príklade boli zmenené 2 súbory a oba boli do local repository vložené ako novo vytvorené súbory. Číslo za `create mode` označuje typ súboru a povolenia. Napr. číslo `100644` znamená, že ide o bežné súbory, nie priečinky, a ich vlastník má povolenia na ich čítanie a zápis. Keď už máme vytvorené lokálne úložisko a v ňom aj nejaké súbory je čas na vytvorenie nášho vzdialeného úložiska.


3. Ako vytvoriť vzdialené úložisko GitHub ?

V nasledujúcej časti vytvoríme **remote repository** na GitHub pomocou GitHub-u. Prejdeme pri tom na webovú stránku GitHub ktorá sa nachádza na webovej adrese [https:// www.github.com](https://www.github.com) kde sa prihlásime. Ak nemáme ešte na GitHub účet, pozrite si video <https://youtu.be/GrWL62j3gTU> v časti „Ako nainštalovať Git a GitHub“ kde nájdeme kroky ktoré treba vykonať na vytvorenie účtu. Po prihlásení sa na GitHub resp. po vytvorení účtu na môžeme prístup k inicializácii a vytvoreniu nášho úložiska na GitHub. Za tým účelom je vhodné si pozrieť video: <https://youtu.be/QuCdgrYph98> Po prihlásení sa na GitHub budete presmerovaní na váš osobný informačný panel ktorý poskytuje informácie o vašich úložiskách a projektoch. Ak je to prvýkrát, čo si na GitHub-e vytvárame úložisko, bude vám poskytnuté tlačidlo **Create Repository** ktoré vám s vytvorením remote repository pomôže. Ak ste už GitHub používali, namiesto toho uvidíte tlačidlo **New**. V oboch prípadoch kliknutím na tlačidlo prejdete na **Create a new repository** stránku, ako je znázornené na obrázku 3.

Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner *

 DeborahK ▾

Repository name *

/ recipes ✓

Great repository names are short and memorable. Need inspiration? How about **studious-enigma**?

Description (optional)

☒  **Public**

Anyone on the internet can see this repository. You choose who can commit.

☐  **Private**

You choose who can see and commit to this repository.

Initialize this repository with:

Skip this step if you're importing an existing repository.

☒ **Add a README file**

This is where you can write a long description for your project. [Learn more](#).

Add .gitignore

Choose which files not to track from a list of templates. [Learn more](#).

.gitignore template: **None** ▾

Choose a license

A license tells others what they can and can't do with your code. [Learn more](#).

License: **MIT License** ▾

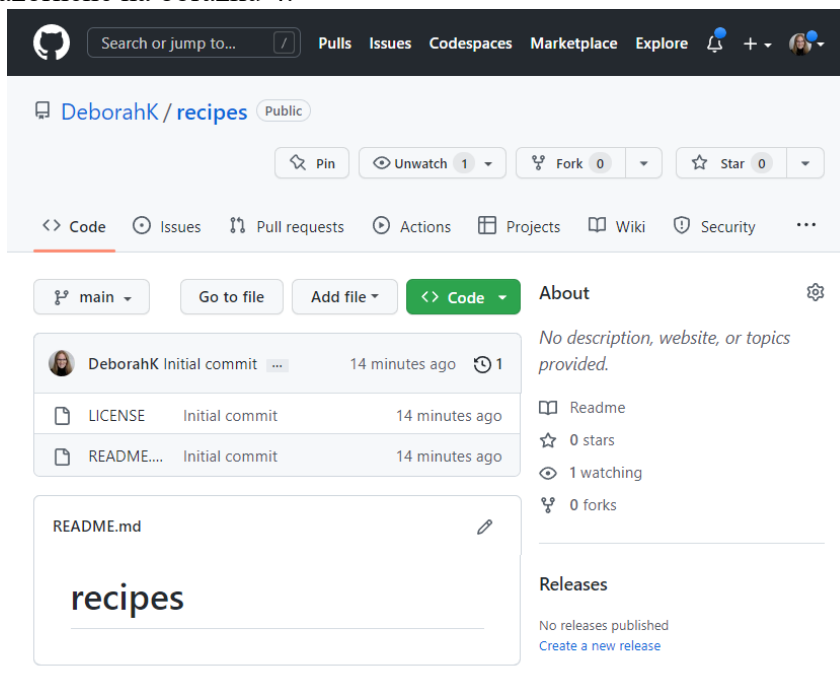
 You are creating a public repository in your personal account.

Create repository



Obrázok 3. Vytvorenie nového vzdialeného úložiska pomocou GitHub.

Začínáme so zadaním názvu úložiska. Názov úložiska by mal byť krátky, ale mal by popisovať váš projekt. Musí byť jedinečný v rámci vášho účtu GitHub t.j. nebal by už byť použitý. GitHub tento názov skontroluje za nás. Všeobecné konvencie pre názvy repozitárov odporúčajú používať malé písmená. Ak existuje viacero slov, použite pomlčky medzi slovami, ako napríklad kniha_receptov. Naše úložisko sme nazvali **recipes**. A zelená šípka vedľa názvu znamená, že je jedinečný v rámci tohto účtu GitHub. Priloženie popisu tohto účtu ako napr. k čomu je používaný je voliteľné. Takto môžete v prípade potreby zadať ďalšie informácie o úložisku. Ďalej máme možnosť vybrať voľbu či bude vytvorené úložisko verejné alebo súkromné. Ak vytvoríte verejné úložisko, môže ho vidieť ktokoľvek na internete. Ale iba spolupracovníci (angl. **collaborators**), ktorých si vy vyberiete, sa môžu k tomu repository pripojiť (commit-ovať). Nemusíte sa teda obávať, že by vaše súbory cudzinci menili. Ak vytvoríte súkromný archív, bude ostatným ľuďom s výnimkou vami vybraných spolupracovníkov nedostupný. Pokiaľ sa na vaše súbory resp. kód nevzťahuje autorský zákon, nie je vlastníctvom nejakej inštitúcie alebo si ho jednoducho chcete nechať iba pre seba, traba zvážiť použitie verejného repository. Následne máte možnosť vytvoriť súbor `readme.md` ktorý má poslanie poskytnúť návštevníkom vášho úložiska predstavu o tom, na čo toto úložisko slúži, a pokyny na jeho používanie. Tento súbor je vytváraný vo formáte Markdown a ak ho chcete pridať začiariknite príslušné políčko. Šablóna `.gitignore` slúži na to aby ste sem vložili súbory z vášho pracovného priestoru ktoré nechcete zahrnúť do archivácie a sledovania verzií Git. Je to napr. v prípade keď ide o súbory s citlivými údajmi alebo iba dočasne použité rozsiahle súbory ktoré možno kedykoľvek jednoducho znova vygenerovať. Pre náš výukový projekt ktorý je veľmi jednoduchý šablónu `.gitignore` nepotrebujeme. Nasleduje výber licencie. Licencia umožňuje ostatným vývojárom poskytnúť informáciu, čo môžu robiť s kódom alebo súbormi ktoré sa nachádzajú vo vašom repo. Napríklad, či ich môžu voľne používať pre svoje vlastné projekty. Pokiaľ chcete použiť túto voľbu GitHub vám poskytne webovú stránku, ktorá vám pomôže pri ďalšom rozhodovaní. Kliknutím na [learn more](#) pod **Choose a license** získate možnosť prejsť na stránku dokumentácie GitHub-u kde nájdete ďalšie informácie. Na tejto stránke nájdete odkaz na [choosealicense.com](#) kde nájdete pomoc pri výbere licencie, ktorá má pre váš projekt zmysel. Pre naše úložisko **receptov** zvolíme licenciu MIT ktorá je jednoduchá a tolerantná. Stlačením tlačítka **Create repository** ukončíme vytvorenie nového úložiska a zatvoríme túto kartu v prehliadači. Následne GitHub poskytne súhrn údajov, takže si túto akciu môžete skontrolovať a vytvorené repo sa ukáže v podobe ako je znázornené na obrázku 4.

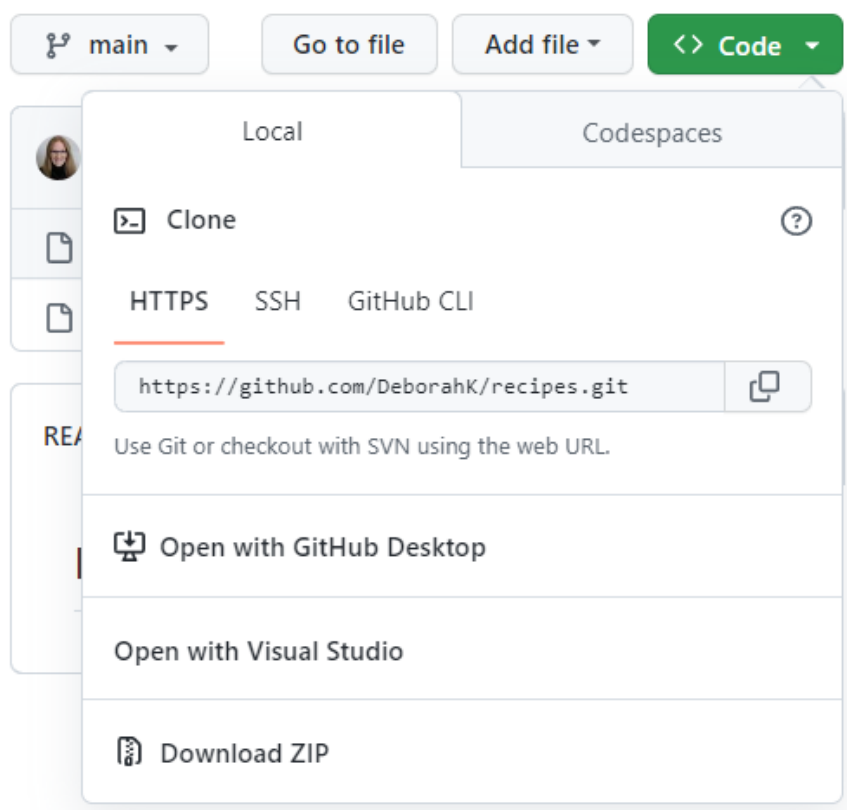


Obrázok 4. Naše vzdialené úložisko GitHub.

Ako vidieť na obrázku 4, GitHub automaticky vykonal náš prvý commit! Toto pripojenie spočíva v začlenení dvoch súborov, ktoré sme označili pri vytváraní remote repository: náš licenčný súbor a readme.md. Nastavila sa správa commitu "Počiatočný commit". A GitHub zobrazí obsah nášho súboru readme.md. V tomto okamihu náš local repository obsahuje naše dva textové súbory file1.txt a file2.txt a náš remote repository obsahuje našu licenciu a súbor readme.md. Teraz budeme chcieť, aby sa tieto dva úložiská zhodovali t.j. **zosynchronizovali**.

Ako povedať Gitu o GitHub ?

Skôr ako budete môcť synchronizovať lokálne a vzdialené úložiská, musíte medzi nimi vytvoriť spojenie. V podstate musíte svojmu lokálnemu úložisku povedať, kde má nájsť vzdialené úložisko. Vzdialené úložisko pritom identifikujeme podľa jeho URL. Na obrázku 4 si všimnime v hornej časti na pravo zelené **Code**. Kliknutím na toto tlačidlo zobrazíte podrobnosti o úložisku (obrázok 5).



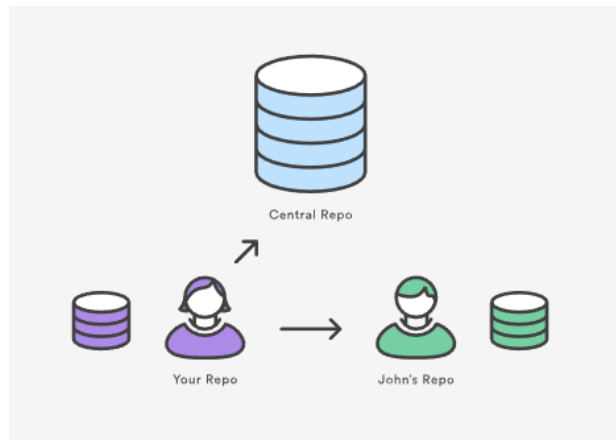
Obrázok 5. Nájdenie adresy URL úložiska GitHub.

Vyberíme kartu HTTPS, a kliknutím na ikonu **copy** skopíruje adresu URL tohto úložiska GitHub. Vratime sa späť do okna príkazového interpreta napr Git shell-u ktoré sa nachádza na našom lokálnom počítači a **vzdialené úložisko prepojíme s lokálnym úložiskom** pomocou príkazu:

```
git remote add recipes-gh https://github.com/DeborahK/recipes.git
```

Potom sa presvedčíme či bola vzdialená konektivita vykonaná a s akými prezývkami príkazom:

```
git remote
```



Obrázok 6. Vzdialené pripojenia nášho repo do centrálneho repo a do repa niekoho iného

Príkaz **remote add**⁴ pridá remote repository na zadanú adresu URL. A umožňuje vám priradiť pre toto úložisko prezývku, čím pri odkazovaní sa na remote repository nemusíme zadávať adresu URL. Bežná prezývka vzdialeného úložiska je **origin**, ale tento názov môže byť trochu mätúci, najmä vtedy ak vzdialené úložisko nie je origin-om projektu ale má odlišný názov a cestu repository ako je to v prípade názvu a cesty repository lokálneho. Kvôli prehľadnosti je v takomto prípade vhodné ako prezývku vzdialeného úložiska používať názov lokálneho úložiska s doplnkom **-gh** (informácia pre GitHub), čím si ho ľahko zapamätáme. Pokojne je však možné použitie akéhokoľvek mena a ide o to aby sme si prezývku zapamätali. Samotný príkaz **remote** zobrazí zoznam vzdialených prezývok remote repository, ktoré vaše local repository pozná. Takto to vyzerá:

```

C:\Users\Deborah\Documents\recipes>git remote add recipes-gh https://github.com/DeborahK/recipes.git
C:\Users\Deborah\Documents\recipes>git remote
recipes-gh
C:\Users\Deborah\Documents\recipes>_

```

Obrázok 7. Vytvorenie pripojenia k vzdialenému úložisku.

Naše local repository Git-u teraz vie, kde má nájsť jeho pridružené remote repository. Ďalším krokom bude synchronizácia týchto repository, aby sa ich história commit-ov zhodovala.

Ako synchronizovať lokálne a vzdialené úložiská ?

Aby sme synchronizovali naše lokálne a vzdialené úložiská, najprv načítame históriu zmien zo vzdialeného úložiska a pomocou príkazu **pull** ich pridáme do nášho lokálneho úložiska. Potom zase naše lokálne histórie zmien presunieme do vzdialeného úložiska pomocou príkazu **push**. Ak by sme však radšej synchronizovali úložiská pomocou VS-Code, pozrite si toto video:

<https://youtu.be/5vYPLUMP6dg>

Z ktorého kód nájdete tu: <https://github.com/DeborahK/petcafe-html>

4

<https://www.atlassian.com/git/tutorials/syncing#:~:text=The%20git%20remote%20command%20lets,direct%20links%20into%20other%20repositories.>

Ak chceme **stiahnuť** súbory z **remote repository** sa do nášho **local repository** príkaz Git-tu bude:

```
git pull recipes-gh main --allow-unrelated-histories
```

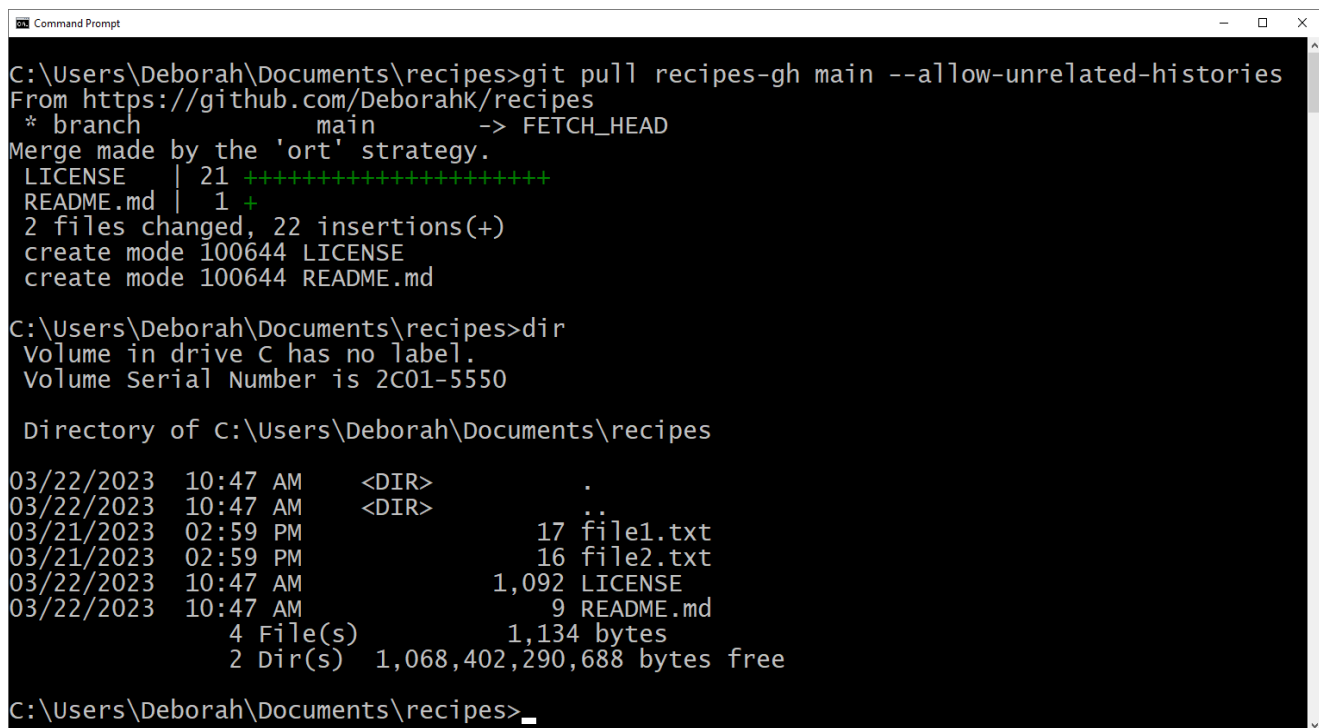
Príkaz **pull** vyžaduje prezývku remote repository (v našom prípade recipes-gh) a názov banch (vetvy). Keďže sme nevytvorili žiadne ďalšie vetvy, špecifikujeme primárnu vetvu (**main**).

Voľba **--allow-unrelated-histories** sa vyžaduje pri prvom stiahnutí, pretože chceme zlúčiť dva archívy ktoré boli vytvorené samostatne a momentálne nezdediajú súvisiacu históriu.

Po prvom použití príkazu **pull** už git nevyžaduje tento príznak a možno používať zápis :

```
git pull recipes-gh main
```

V oboch prípadoch príkaz **pull** pomocou zadanej prezývky (**recipes-gh**) vyhľadá príslušné URL pre remote repository. Následne získa históriu commit-ov a ďalšie údaje zo zadanej vetvy (main) vzdialeného archívu, ktorý sa vo vetve lokálneho archívu nenachádza. Prevzaté údaje zlúči s údajmi uloženými v lokálnom úložisku. Ak sa vyskytnú nejaké konflikty kým Git zmeny zlúči, budete ich musieť vyriešiť manuálne. Nakoniec Git aktualizuje lokálny pracovný priečinok so všetkými súbormi z posledného commit-u. Pomocou príkazu **dir** v systéme Windows alebo príkazu ls v systéme Mac môžeme zistiť či už náš local repository teraz má naše pôvodné súbory plus súbory licencie a readme.md z nášho remote repository tak ako je znázornené na obrázku 8.



```
Command Prompt
C:\Users\Deborah\Documents\recipes>git pull recipes-gh main --allow-unrelated-histories
From https://github.com/DeborahK/recipes
* branch      main      -> FETCH_HEAD
Merge made by the 'ort' strategy.
 LICENSE      | 21 ++++++
 README.md    | 1 +
 2 files changed, 22 insertions(+)
 create mode 100644 LICENSE
 create mode 100644 README.md

C:\Users\Deborah\Documents\recipes>dir
Volume in drive C has no label.
Volume Serial Number is 2C01-5550

Directory of C:\Users\Deborah\Documents\recipes

03/22/2023  10:47 AM    <DIR>          .
03/22/2023  10:47 AM    <DIR>          ..
03/21/2023  02:59 PM             17 file1.txt
03/21/2023  02:59 PM             16 file2.txt
03/22/2023  10:47 AM           1,092 LICENSE
03/22/2023  10:47 AM              9 README.md
               4 File(s)              1,134 bytes
               2 Dir(s)  1,068,402,290,688 bytes free

C:\Users\Deborah\Documents\recipes>_
```

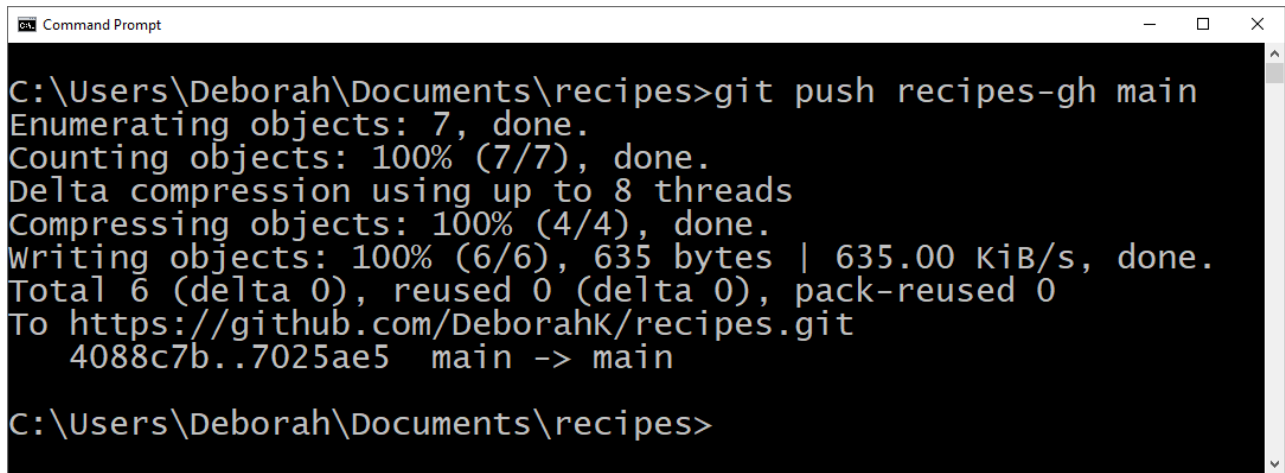
Obrázok 8. Zobrazenie histórie zmien z remote repository (vzdialeného úložiska).

Ak je tomu tak, tak teraz naše lokálne úložisko obsahuje celú históriu commitov z úložiska vzdialeného.

Históriu zmien z nášho lokálneho úložiska presunieme do nášho vzdialeného úložiska pomocou príkazu **push** nasledovne :

```
git push recipes-gh main
```

Príkaz **push** vyžaduje názov remote repository (**recipes-gh**) a názov vetvy (branch). Keďže sme nevytvorili žiadne ďalšie vetvy, špecifikujeme primárnu vetvu (**main**). Tento príkaz zlúči históriu lokálnych zmien do vzdialeného archívu. Ak sa vyskytnú nejaké konflikty, budete ich musieť vyriešiť manuálne, až potom Git zmeny zlúči. Výsledok tejto operácie je vidieť na obrázku 9:

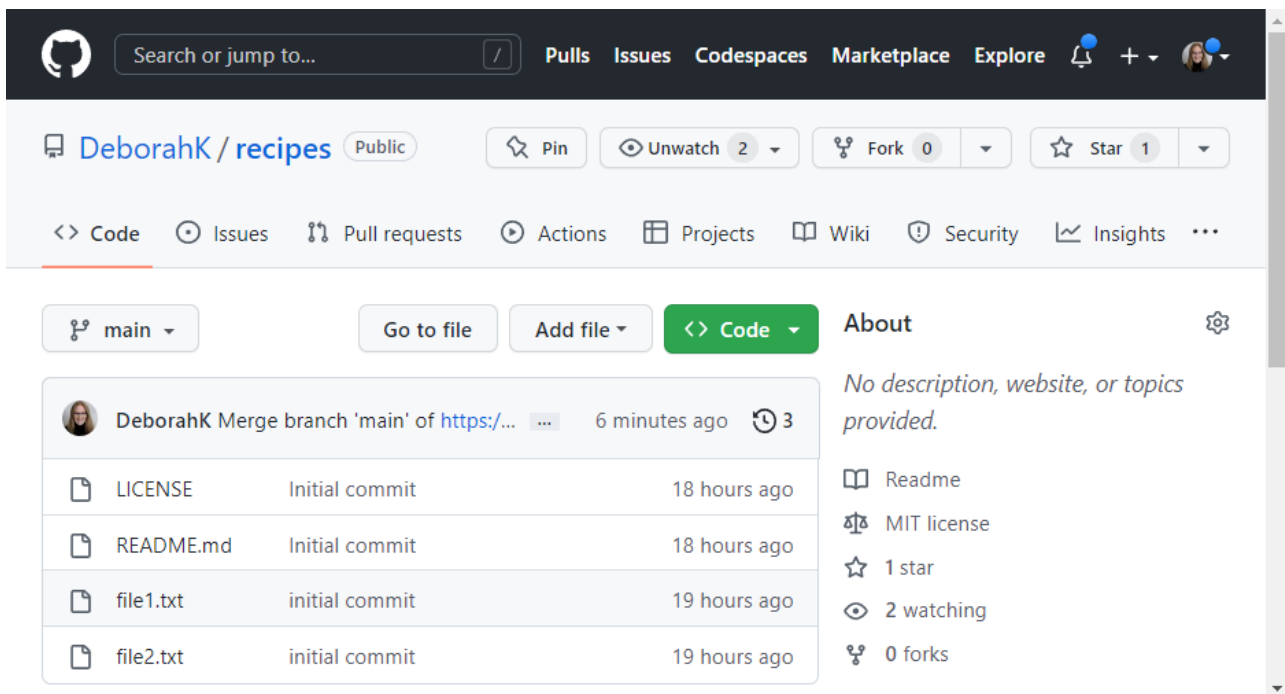


```
C:\Users\Deborah\Documents\recipes>git push recipes-gh main
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (4/4), done.
Writing objects: 100% (6/6), 635 bytes | 635.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/DeborahK/recipes.git
 4088c7b..7025ae5  main -> main

C:\Users\Deborah\Documents\recipes>
```

Obrázok 9. Prenesenie histórie lokálnych zmien do vzdialeného archívu.

Ak sa chceme uistiť, že naše vzdialené úložisko obsahuje všetky naše súbory, pozrieme si vzdialené úložisko na GitHub-e. Na obrázku 10 vidíme, že súbory z nášho lokálneho úložiska sú teraz aj v našom vzdialenom úložisku a naše **úložiská sú synchronizované!**



Obrázok 10. Naše vzdialené úložisko so súbormi z nášho lokálneho úložiska.

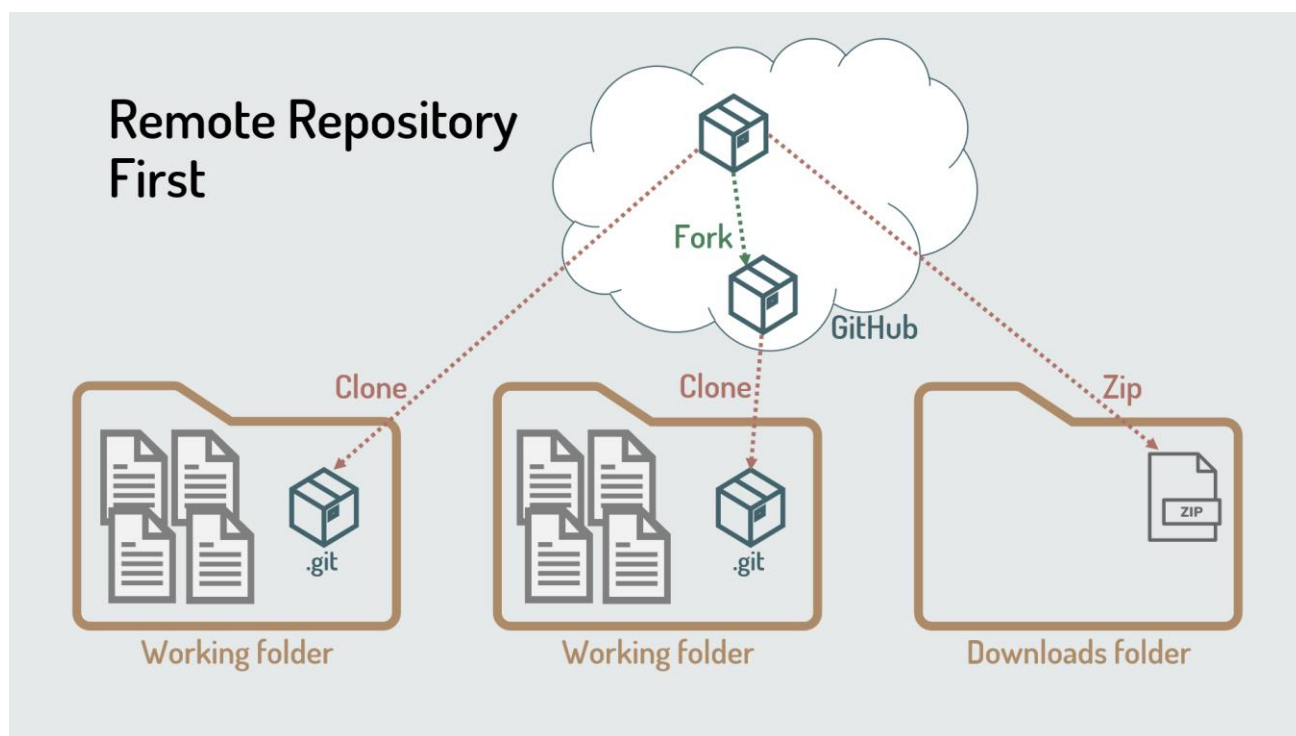
Stručne povedané, **synchronizácia vašich local a remote repository** zahŕňa stiahnutie všetkých zmien zo vzdialeného úložiska do vášho úložiska lokálneho, vyriešenie konfliktov a následné prenesenie vašich lokálnych zmien späť do vzdialeného úložiska.

Toto všetko čím sme sa doposiaľ zaoberali platilo pre prvý scenár. Vytvorili sme si lokálne úložisko z našich existujúcich súborov. Potom sme si vytvorili vzdialené úložisko, nastavili pripojenie k

tomuto vzdialenému úložisku a synchronizovali naše súbory. Následne sme opakovali príkazy **pull** a **push** čím sme uskutočňovali synchronizáciu týchto úložísk.

Scenár 2: Vytvorenie vzdialeného úložiska ako prvého

Pri našom druhom scenári vychádzame z toho že už vzdialené úložisko existuje. Bude to napr. prípad ak sa pripojíme, ak sa pripojíte k tímu alebo niekomu (napr. učiteľovi), ktorý už má uložené existujúce súbory na GitHub-e. Alebo ak budeme napr. pracovať na projekte s otvoreným zdrojovým kódom, alebo ak ste na GitHub-e našli nejaký kód resp. súbory, ktoré by ste chceli použiť ako štartovaciu čiaru pre svoj vlastný projekt. Ak chcete **vytvoriť svoj local repository z existujúceho remote repository** a stahovať odtiaľ súbory, existujú tri základné spôsoby, ako to urobiť vid'. obrázok 11.



Obrázok 11. Získanie súborov zo vzdialeného úložiska.

Na ľavej strane obrázku 11 máme naznačenú jednu možnosť: používa sa vtedy **ak vlastníte alebo máte prístupové práva na úpravu vzdialeného úložiska**. Aby ste v takomto prípade vytvorili svoje lokálne úložisko môžete **vzdialené úložisko naklonovať**. Potom už stačí len priamo synchronizovať zmeny medzi dvoma úložiskami.

Druhá možnosť je zobrazená v strede obrázku 11: Platí pre prípad ak **nevlastníte vzdialený repozitár alebo nemáte prístupové práva na jeho úpravu**. Vtedy sa použije ako prvé GitHub na rozvetvenie (**fork**) repozitára. Tým sa vo vašom účte GitHub-u vytvorí kópia vzdialeného úložiska. Následne aby ste vytvorili svoje lokálne úložisko už **naklonujte iba svoju kópiu**. Robí sa to najmä vtedy, ak pracujete na projekte s otvoreným prístupom k súborom a zdrojovým kódom alebo ak napr. ako východiskový bod pre váš projekt chcete použiť súbor či kód ktorý sa nachádza v úložisku niekoho iného.

Tretia možnosť je zobrazená vpravo na obrázku 11 – a vychádza z toho že je možnosť stiahnuť si zip súboru zo vzdialeného úložiska. Táto možnosť však na rozdiel od prvých dvoch nie je založená na vytvorení lokálneho úložiska. Preto ju používame hlavne vtedy ak sa chcete iba pozrieť na obsah

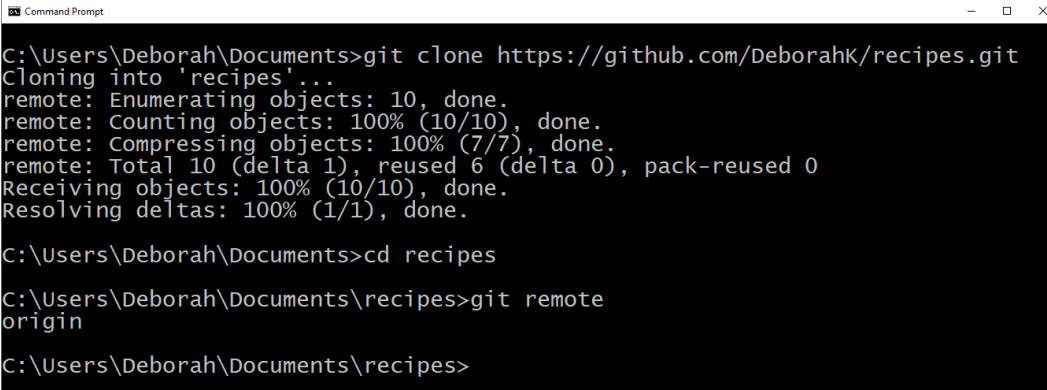
súbore alebo kódu ktorý sa nachádza na nejakom verejnom úložisku GitHub-u a neplánujete v ňom sledovať zmeny. Na uskutočnenie tejto možnosti nám stačí aj vzdialený archív, ktorý nemá priradený žiadny náš lokálny archív. Môžete použiť napr. použiť vzdialený archív **recipes**, ktorý sme už vytvorili pri prvej možnosti. Ak by ste pre tento prípad ale radšej použili úložisko niekoho iného, je potrebné ho najskôr fork-ovať (viď. druhá možnosť). Keď už máte vytvorené vzdialené úložisko, treba sa uistiť, že s ním nie je spojené žiadne lokálne úložisko. Ak ale použijete **recipes** ako vzdialený archív, je potrebné pri tejto možnosti odstrániť priečinok **recipes** z vášho lokálneho systému. Tým sa odstráni aj váš lokálny archív a teraz ste pripravení vytvoriť nové lokálne úložisko zo vzdialeného úložiska nachádzajúceho sa na GitHub-e.

Ako klonovať úložisko ?

Ak chcete **vytvoriť local repository z remote repository** nachádzajúceho sa na GitHub-e, použijeme príkaz **clone**. Ak by ste radšej uprednostnili klonovanie úložiska pomocou aplikácie GitHub Desktop, pozrite si toto video: <https://youtu.be/GpqIr0y2rvc> Na **klonovanie pomocou príkazu Git** potrebujeme najskôr URL vzdialeného úložiska. Potom prejdite na GitHub, prihláste sa a uvidíte svoj osobný informačný panel. Nájdite úložisko, ktoré chcete klonovať. Kliknutím na zelené tlačidlo **Code** zobrazíte podrobnosti o úložisku, tak ako je znázornené na obrázku 5. Kliknutím na tlačidlo copy vedľa adresy URL si ju skopírujete do clipboard-u⁵. Potom otvorte terminál alebo príkazový riadok v počítači a prejdite do priečinka, v ktorom chcete vytvoriť pracovný priečinok vášho projektu. Ak prejdeme do nášho priečinka **Documents** tak už budeme pripravení na klonovanie tohto folder-u príkazom :

```
git clone https://github.com/DeborahK/recipes.git
```

Príkaz **clone** vyžaduje URL remote repository a túto adresu URL sme práve skopírovali z GitHub-u, takže ju iba vložíme ako súčasť tohto príkazu. Keď bude príkaz vykonaný, najprv sa vytvorí pracovný priečinok pre úložisko pomocou názvu vzdialeného úložiska z adresy URL, ktorá je v tomto prípade **recipes**. Potom skopíruje históriu commitov zo vzdialeného úložiska na poskytnutú adresu URL. Proces klonovania tiež automaticky prepojí nové lokálne úložisko s jeho vzdialeným úložiskom. A vzdialenému úložisku priradí prezývku **origin**. Na základe toho môžeme odkazovať na vzdialené úložisko namiesto jej adresy URL pomocou tejto prezývky. Nakoniec proces klonovania skopíruje všetky súbory z posledného commit-u do lokálneho pracovného priečinka. Potom už môžeme pri lokálnej práci na projekte pridávať, upravovať alebo odstraňovať súbory iba v tomto pracovnom priečinku. Ak prejdeme nadol do nového priečinka **recipes** tak na zobrazenie prepojeného vzdialeného úložiska použijeme známy príkaz **remote**. Na obrázku 12 je vidieť, že tento priečinok je skutočne prezývaný ako **origin**.



```
Command Prompt
C:\Users\Deborah\Documents>git clone https://github.com/DeborahK/recipes.git
Cloning into 'recipes'...
remote: Enumerating objects: 10, done.
remote: Counting objects: 100% (10/10), done.
remote: Compressing objects: 100% (7/7), done.
remote: Total 10 (delta 1), reused 6 (delta 0), pack-reused 0
Receiving objects: 100% (10/10), done.
Resolving deltas: 100% (1/1), done.

C:\Users\Deborah\Documents>cd recipes

C:\Users\Deborah\Documents\recipes>git remote
origin

C:\Users\Deborah\Documents\recipes>
```

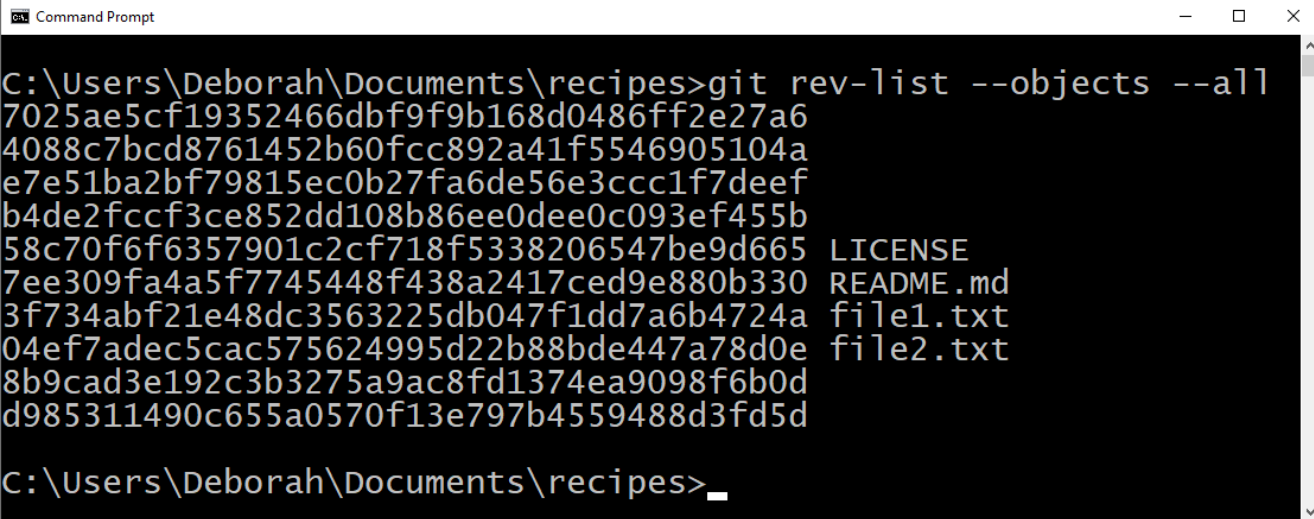
Obrázok 12. Klonovanie úložiska.

⁵ <https://tech.sme.sk/c/578871/co-je-to-schranka-clipboard.html>

Všimnite si výstup operácie klonovania na obrázku 12. Git vymenuje (alebo nájde) 10 objektov. Vo vzdialenom úložisku však máme iba štyri súbory **recipes**. Prečo potom nachádza 10 predmetov? Je to preto, že proces klonovania kopíruje celú históriu commit-ov a nielen aktuálne súbory. Na zobrazenie zoznamu objektov použijeme príkaz **rev-list** s príznakmi **--objects** a **--all**

git rev-list --objects --all

Obrázok 13 zobrazuje výstup tohto príkazu:



```
C:\Users\Deborah\Documents\recipes>git rev-list --objects --all
7025ae5cf19352466dbf9f9b168d0486ff2e27a6
4088c7bcd8761452b60fcc892a41f5546905104a
e7e51ba2bf79815ec0b27fa6de56e3ccc1f7deef
b4de2fccf3ce852dd108b86ee0dee0c093ef455b
58c70f6f6357901c2cf718f5338206547be9d665 LICENSE
7ee309fa4a5f7745448f438a2417ced9e880b330 README.md
3f734abf21e48dc3563225db047f1dd7a6b4724a file1.txt
04ef7adec5cac575624995d22b88bde447a78d0e file2.txt
8b9cad3e192c3b3275a9ac8fd1374ea9098f6b0d
d985311490c655a0570f13e797b4559488d3fd5d

C:\Users\Deborah\Documents\recipes>
```

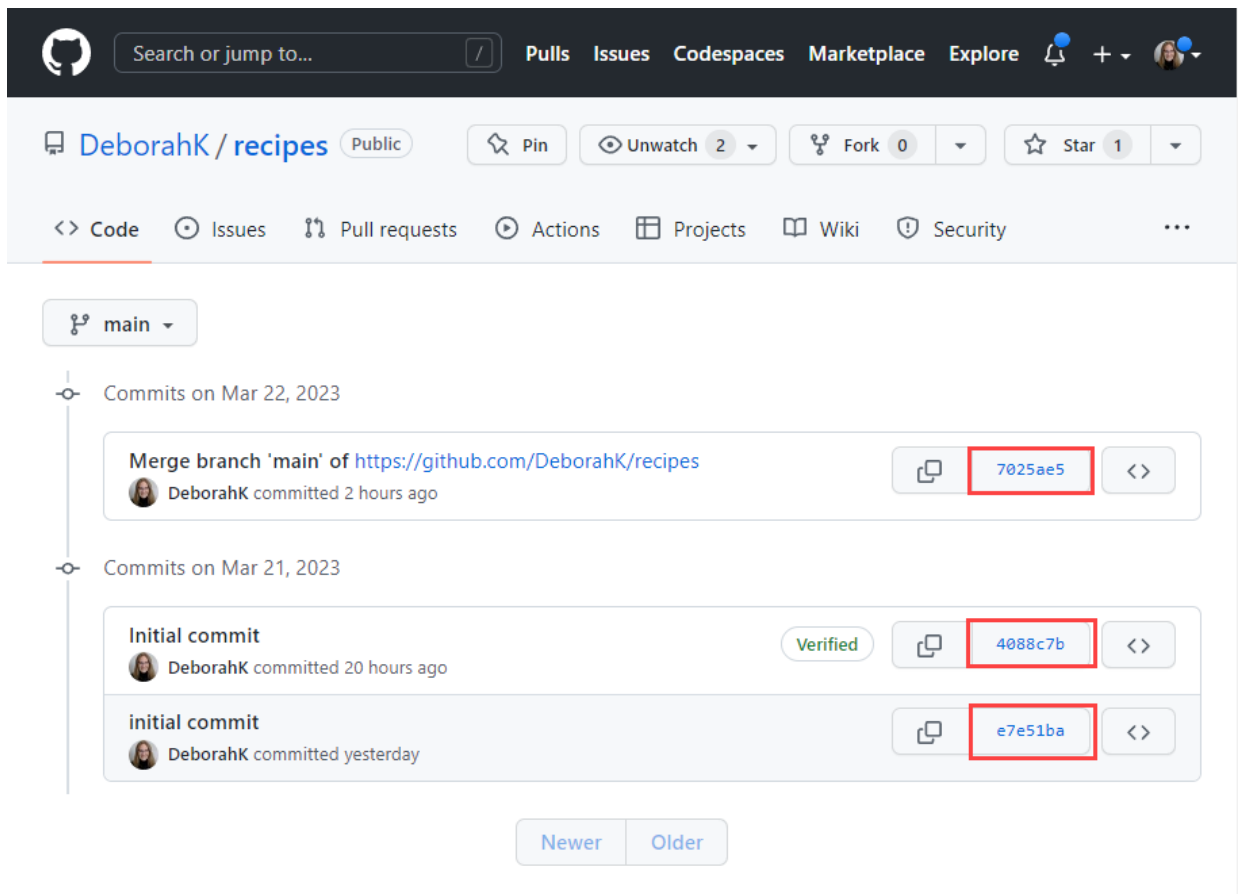
Obrázok 13. Zoznam objektov v úložisku.

Každý z vyššie uvedených objektov je zobrazený ako identifikátor SHA alebo Secure Hash Algorithm⁶. SHA sa tiež používa ako id (identifikátor) na jedinečnú identifikáciu objektov v našom úložisku. Prvé tri objekty v tomto zozname (objekty 1-3) sú tri commit-y vykonané v tomto úložisku. Pri pohľade na zoznam potvrdení v GitHub (obrázok 14) sa pri každom potvrdení zobrazuje niekoľko prvých znakov svojho SHA. Tieto sa zhodujú s prvými tromi objektmi:

- Potvrdenie zlúčenia, keď sme zlúčili dva úložiská.
- GitHub **initial commit** prešiel do vzdialeného úložiska a vytvoril súbory **LICENSE** a **README.md**.
- **initial commit** hovorí že miestneho úložiska sme sa commit-ovali súbory file1.txt a file2.txt.

Objekty nachádzajúce sa v strede zoznamu znázornenom na obrázku 13 (t.j. objekty 5-8) sú štyri commit-ované súbory. Zvyšné tri objekty (t.j. objekty 4, 9, 10) sú stromové objekty ktoré reprezentujú stav priečinka vrátane súborov a podpriečinkov ktoré sú v tomto priečinku. Pritom sa zachováva hierarchia priečinka.

⁶ <https://sk.theastrologypage.com/secure-hash-algorithm>



Obrázok 14. História commitov kedy každý je identifikovaný niekoľkými prvými hodnotami SHA.

Pomocou **dir** príkazu v systéme Windows alebo **ls** príkazu v systéme Mac uvidíme, že náš pracovný priečinok teraz obsahuje všetky súbory z nášho vzdialeného úložiska.

```

C:\Users\Deborah\Documents\recipes>dir
Volume in drive C has no label.
Volume Serial Number is 2C01-5550

Directory of C:\Users\Deborah\Documents\recipes

03/22/2023  12:35 PM    <DIR>          .
03/22/2023  12:35 PM    <DIR>          ..
03/22/2023  12:35 PM                17 file1.txt
03/22/2023  12:35 PM                16 file2.txt
03/22/2023  12:35 PM            1,092 LICENSE
03/22/2023  12:35 PM                9 README.md
               4 File(s)              1,134 bytes
               2 Dir(s)  1,068,328,906,752 bytes free

C:\Users\Deborah\Documents\recipes>_

```

Obrázok 14. Adresár súborov v našom pracovnom priečinku.

Po tom čo sme naklonovali vzdialené úložisko, naše lokálne a vzdialené úložiská sa budú zhodovať. Aby sme ich udržali v synchronizácii, používame **pull** a **push** ako sme videli v časti „[Ako synchronizovať miestne a vzdialené úložiská](#)“ vyššie v tomto článku. Všimnite si však, že ak pracujeme s remote repository, ktoré nevlastníme, asi nebudeme mať prístup na priame ukladanie zmien do tohto úložiska. Ak by sme to však chceli budeme musieť majiteľa priečinka požiadať o túto činnosť. Žiadosť na stiahnutie alebo uskutočnenie iných úkonov informuje vlastníkov alebo iných prispievateľov, že ste do ich repository vložili zmeny a požadujete, aby boli tieto zmeny skontrolované a stiahnuté alebo začlenené do ich vzdialeného úložiska. Ďalšie informácie o žiadostiach ohľadom stiahnutia a ukladania zmenených súborov do cudzieho priečinka nájdete v tomto videu:

https://youtu.be/v_A8O3cpDyM

Zhrnutie záverom

V prvom scenári sme mali existujúci súbor alebo vytvorený kód a pre ne sme mali vytvorené lokálne úložisko. Až následne sme vytvorili vzdialené úložisko, aby sme si tu uchovali, sprístupňovali a archivovali kópie týchto súborov a kódov programu aj mimo lokality v našom počítači.

V druhom scenári bolo už vytvorené vzdialené úložisko v GitHub-e a my sme mali už spolu s jeho vlastníkom, našim tímom a rôznymi prispievateľmi umožnený prístup ku zdrojom ktoré sa tam nachádzajú. Potom sme toto vzdialené úložisko naklonovali, preto aby sme si vytvorili z neho vytvorili naše lokálne úložisko ktoré je s týmto vzdialeným úložiskom previazané. Týmto spôsobom môžeme pracovať na našej vlastnej lokálnej kópii súboru, kódu resp. projektu a vykonávať na nich testy, pokusy a zmeny nezávisle od toho čo sa deje na iných úložiskách. To sa deje pred tým, ako tieto zmeny prenesieme do centrálného vzdialeného úložiska, aby k nim mali prístup aj ostatní ľudia a mohli ich študovať, používať, kontrolovať, pridávať do svojich riešení a zlučovať do spoločného projektu.

Je jedno či už najskôr vytvoríte local repository alebo najprv vytvoríte remote repository na GitHub-e, podstatne je to že tieto úložiská môžeme synchronizovať s príkazmi Git-u **pull** a **push** až potom keď budú obe tieto úložiská fyzicky vytvorené. Porozumenie problematiky local repository a remote repository je dôležité nie len ich udržiavanie v synchronizácii, ale aj v prípade tímovej práce na spoločných projektoch a situáciách kedy je potrebné od vlastníka vzdialeného úložiska požiadať o prístup k tomuto úložisku. Ďalšie podrobnosti o učení Git a GitHub nájdete v tomto kurze:

<https://youtu.be/pICJdbC7j0Q>