

Git: a legvégső visszavonás gomb

PCLinuxOS Magazine – 2016. szeptember

Írta: Peter Kelly (critter)

A legmodernebb számítógépes alkalmazások, mint a szövegszerkesztők és grafikai tervezők bírnak valamilyen eljárással, ami lehetővé teszi a felhasználónak a továbbiakban szükségtelennek számító művelet visszavonását. Ám az alkalmazás bezárásával általában ez a lehetőség elveszik. Hacsak nem mentettél, a régi változat örökre elveszett.

A programozók már régen rájöttek, hogy ez egy nem kívánatos helyzet és elhatározták, hogy tesznek valamit ez ügyben. Az eredmény verziókontroll-rendszer néven ismert és számos ilyen rendszer jelent meg eltérő képességekkel, összetettséggel és a többihez képest egy-egy megoldásában elkerülhetetlenül hátránnyal.

Többségünk nem fejlesztő, aki egyszerre dolgozik több szoftver projekten együtt más fejlesztőkkel, így számunkra ezek a rendszerek olyan teremtmények, amik több erőfeszítést követelnek, mint amennyit megtakarítanak.

Git



Ugyanakkor van egy rendszer, ami más, a neve Git. A Git kitűnő pedigivel rendelkezik, és emellett néhány határozott előnnyel:

- Linus Torvalds fejlesztette ki, a Linux kernel fejlesztése ellenőrzésének támogatására;
- elképesztően stabil, hiszen nagyon régóta, 2005 óta létezik, valójában régebb óta létezik, mint maga a PCLinuxOS Magazine;
- nagy képességű, de olyanok is használhatják, akiknek szerényebbek az elvárásai és csak néhányat kezelnek a sok, elérhető parancs közül;
- a működtetése nem kívánja meg a kliens-szerver felállást;
- alkalmas egyének számára bármely projekthez, nem csak kódolóknak;
- noha eredetileg a Linux-közösség számára készült, a Git elérhető a legtöbb, népszerű operációs rendszer alatt, közte Mac, Windows és Solaris alatt is.

Nos, mi a Git? „Elosztott verziókontroll-rendszer”-ként lehet leírni, de ez ne vezessen félre. A leírás megmondja, hogy a Git mire képes, és nem azt, hogy mire használható. A Git maga egy parancssori eszköz, de a széleskörű alkalmazásának köszönhetően (a legnépszerűbb verziókontroll-rendszerre vált mára), számos grafikus kezelőfelület van, ami megédesítheti az életedet és megőrzi egészségedet.

Nekem való a Git?

Jóllehet, arra tervezték, hogy sok felhasználó egy projekten dolgozzon vele, egyének is élvezhetik a rendszer előnyeit. A munkám zömét a fő számítógépemen készítem, azután bizonyos dolgokban a laptopomon folytatom a munkát. A laptop hálózaton keresztül tud csatlakozni a fő számítógéphez, de egyetlen laptoptal, vagy asztali géppel is élvezhetjük a Git előnyeit.

Hogy neked miképpen lehet hasznos, függ attól, hogy mit csinálsz, de hadd mutassak néhány egyszerű példát.

Van egy nagy zenei fájlgyűjteményed, amit CD-kről olvastál be, egy adott CD-ről minden szám egy könyvtárban. Néhány CD válogatást tartalmaz és szeretnéd a számokat előadó szerint újrendezni. Ha a gyűjteményed több ezer számból áll, akkor ez egy elég bonyolult feladat. Ha később úgy döntesz, hogy az eredeti sorrend sokkal jobb volt, akkor elég komoly feladat elé nézel, hacsak nem készítettél mentést és most két gyűjteményed is van. Azonban, ha először csináltál egy Git tárolót, amihez csupán három egyszerű parancs kell, akkor a kétfajta rendezési sorrend közötti váltás csak néhány egyszerű parancs. Ez még elég gyors is, hiszen az aktuális zenei adatokat nem kell újraírni, csak a helyük sorrendjét kell átrendezni.

A Git használata

A Git-et parancssoros programnak tervezték, de ma már számos grafikus alkalmazás áll rendelkezésre. A grafikus „felületek” nagy segítséget jelentenek a projekt előrehaladásának és a végrehajtott változtatások megjelenítésében. Használhatjuk a projektünk irányítására, de a Git igazi ereje a parancssorban van. Tehát bemutatom mindkettőt.

Git: a legvégső visszavonás gomb

Borzasztóan sok parancs és opció tartozik a Git-hez, de projektirányításra és időszakonkénti pillanatfelvételekre, amikhez bármikor visszatérhetsz, csak nagyon kevés kell.

A bemutató projektben létrehozok egy Git-tárolót, hozzáadok néhány projektfájlt és a kezdeti szakaszban készítek egy pillanatfelvételt a projektről (a Git-szlangban ezeket a felvételeket commit-oknak hívják, mivel a legfrissebb munkádat hozzáfűzöd a projekt aktuális szakaszához). A következő szakaszban további fájlokat adok hozzá, néhányat módosítok és törölök pár eredeti fájlt, majd készítek egy újabb felvételt. Végül visszatérek az eredeti felvételhez, ami helyreállítja a lemeztől törölt fájlokat (igen, a fájlokat tényleg töröltük, de a Git képes újraépíteni a fájlokat a tárolóban rögzített adatokból). Hogy mindezt végrehajtsam, csak ezek a parancsok kellene:

init, add, commit, rm és revert.

A bemutató projekt magyarázata során sok más parancsot is kell használnom, amik a tároló kezeléshez hasznos információkat jelenítenek meg. Céljuk segíteni a tisztánlátást, hogy mit és miért csinálunk, de nincs feltétlen szükség rájuk. Van még néhány további parancs, amit alkalmanként használok, de a legtöbb egyfelhasználós munkához ennyi elég és ez csak töredéke annak, ami rendelkezésre áll.

Indítás

A Git alapból nem települ a PCLinuxOS alatt, de a tárolóban elérhető. Indítsd el a Synaptic-ot, frissíts, jelöld ki az összes frissítést és alkalmazd, hogy a rendszered teljesen naprakész állapotban legyen. Ezután görgess le a Git-hez, jelöld ki telepítésre és alkalmazd. Néhány további, a Git-hez szükséges fájlt is magával húz. Engedélyezd! Most már van telepített Git. Az első használat előtt meg kell adni néhány információt. Nem öncélúan, hanem így közlöd a Git-tel, hogy ki, mit csinált. A Git-nek kell a neved és e-mail címed, de mivel mi nem közösködünk másokkal, ez tetszés szerint elhagyható. Én ezt a parancsot adtam ki:

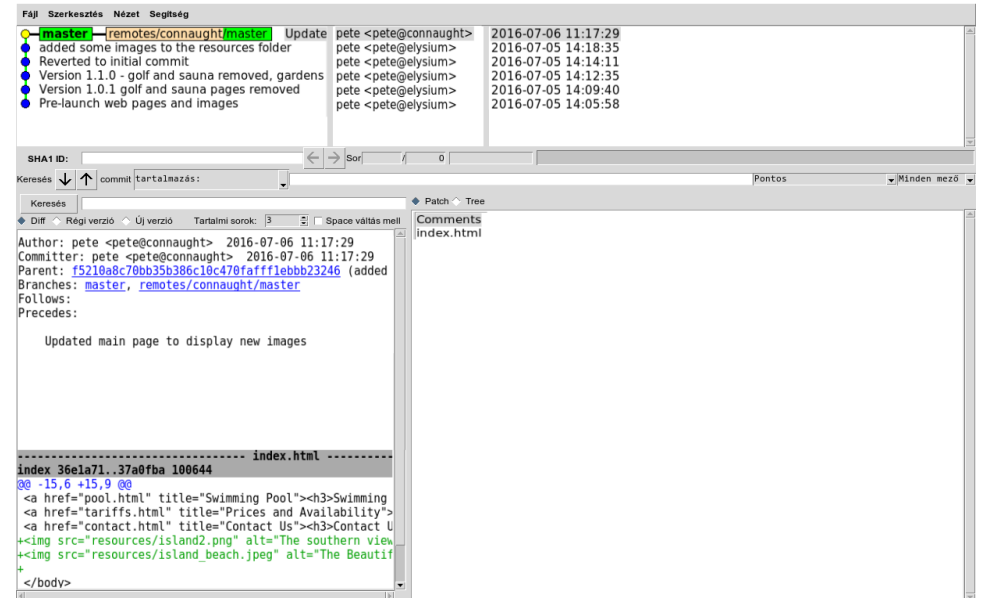
```
git config --global user.name "pete"
git config --global user.email "pete@connaught"
```

A „connaught” az e-mail címben jelen esetben az a host név, amit a számítógépnek adtam. A Git kész a használatra.

A Git használatának bemutatására készítek egy egyszerű weblapot néhány további lappal, amikre hivatkozni lehet. Nem feltétlenül kell weblap. Bármilyen dokumentum lehet, vagy dokumentumok sora, amik időről időre változhatnak.

Lehet akár programkód is! Nincs szükség web-, vagy HTML-ismeretekre a megértéshez.

A Git használata nem bonyolult, de egy kicsit más, mint más alkalmazásoké. A képernyőkép, ami néhány, a bemutatóban használt parancs kimenetét mutatja, kicsit zsúfoltnak tűnik, de majd világos lesz.



Ne aggódj, a Git majdnem mindent elvégez helyetted.

Az első dolog, amit csinálnunk kell, egy könyvtár a projekt fájljainak tárolására, ezt „sunshine island”-nak nevezzük el, ezután belépünk a könyvtárba. Ehhez nyiss terminált és menj oda, ahol a projektet tárolni szeretnéd, majd írd be ezt a két parancsot:

```
mkdir sunshine_island
cd sunshine_island
```

Most már a projektkönyvtárba kerülünk, amire eztán „munkakönyvtár”-ként hivatkozom. Teljesen üres, tehát közölnünk kell a Git-tel, hogy ez a „Git tároló” (repository) és gondoskodnunk kell arról, hogy a Git figyelje ezt a könyvtárat. A beállításához használt parancs a git init.

```
git init
Initialized empty Git repository in
/home/pete/sunshine_island/.git/
```

Ezzel létrehoz egy .git nevű rejtett könyvtárszerkezetet az aktuális könyvtárban, ami tartalmazza az összes, a Git számára a fájlok létrehozásának, változtatásának, vagy törlésének követéséhez szükséges információt. A továbbiakban erre „Git könyvtár”-ként hivatkozom. Belenézhetsz a könyvtárba, de ne változtass meg semmit, legalábbis addig, amíg úgy nem érzed, hogy tudod mit csinálsz. Ezután adnunk kell néhány fájlt a Git-nek, hogy figyelje.

Előbb egy kis elmélet.



A Git tároló jelenleg egy üres váz. Amikor fájlokat adunk hozzá a munkakönyvtárhoz, a Git-nek meg kell mondanunk, hogy azok közül a fájlok közül melyek követendők és melyek nem. A fájl, vagy -típus tiltásához létre kell hozni egy .gitignore fájl és abban felsorolni. Pl. néhány szerkesztő mentést készít a fájl nevével, amit hullámvonal vezet be „~afájloom.txt”. Ha ebbe a fájlba beírod ~*, akkor az összes ilyen fájl a Git teljesen negligálja.

A Git az add parancsot használja hivatkozás elhelyezésére a tárolási (holding), vagy feldolgozási (staging) területen lévő fájlokra, amiket később „commit”-olunk a tárolóba.

Amikor a fájlt először „commit”-oljuk, a fájl összes adata a tárolóba bekerül egy referenciakulccsal. A kulcs egy SHA-1 ellenőrzőösszeg néven ismert 40 hexadecimális elemből álló egyedi karakterlánc, amit nem biztonsági okokból, hanem az egyedisége miatt használunk. A fájlnevekkel ellentétben egy SHA-1 kulcs nagy valószínűséggel nem ismétlődik meg ebben a világban, így a Git pontosan tudja, hogy melyik adata hivatkozunk. Erről nem kell túl sokat tudnunk a Git használatához.

Ha egy követésre kijelölt fájl bárhogy megváltozik, a Git tudni fog róla és képes azt jelezni. Ezután ha a fájlt a „staging” területhez adod (add), akkor a csak a megváltozott adatokat rögzíti. Mihelyst a megváltozott fájlokat „commit”-olják a tárolóba, az adatváltozások bekerülnek a Git adatbázisába és hivatkozáshoz új ellenőrzőösszeget kap. Ebből az adatbázisból a Git képes bármilyen fájlösszeállítást egy korábbi „commit”-olt állapotba visszaállítani – ahogy azt később látni fogjuk.

A bemutató projekt

Az elképzelt helyzet

A Sunshine Island egy üdülőhely brosúrája nyaraláshoz, amit te készítesz. Nem kell tudnod, hogy milyen látogatókat vonzhat, vagy milyen létesítmények lehetnek, vagyis ez leginkább „puhatolózó” oldal, amit gyorsan kell megváltoztatni, ha a lehetséges érdeklődők mást vártak. Amennyiben rossz irányba mozdultál, akkor képesnek kell lenned az eredeti felálláshoz visszatérni. Emellett jogi elvárás, hogy feljegyzést kell vezetned arról, hogy mit, mikor és kinek ajánlottál.

Először olyan üdülőre gondolunk, ami luxus elhelyezést, golf-pályát, koktélbárt, szaunát, úszómedencét kínál – a teljes paletta.

Egyelőre a weblap leírása valahogy így néz ki:

- Fő oldal – képi hivatkozásokkal a látványosságokra;
- Szállás oldal – képek és a szobák leírásai;
- Golfpálya oldal – képek és leírás az egyes lyukakról;
- Koktélbár oldal – ismertető a bár lehetőségeiről;
- Szauna oldal – egyszerű lap, az elérhető szolgáltatások felsorolásával;
- Úszoda oldal – rengeteg kép az uszodában szórakozó emberekről;
- Tarifák oldal – a szigeten elérhető szolgáltatások és hozzá tartozó árak;
- Kapcsolatok oldal – foglalás információ, köszönet-nyilvánítás, vagy panasz;

A Git létrehozott munkakönyvtárban most már megcsinálhatod a szükséges weblapokat.

Az első visszajelzés kedvezőtlen. A golfpálya látogatóit zavarni fogja, ha látják a majmokat, amint golflabdákat csennek el. A szauna is rossz ötlet, mivel a hely trópuson van és egy forró szauna itt nem népszerű. A többi nem olyan rossz, de változtatni kell. A döntés az, hogy a golfpályát le kell venni és helyette buja trópusi kertet kell bemutatni, paradicsommadarakkal és kedves, szelíd állatokkal. A szauna helyére egy légkondicionált könyvtár kerül, ahol a vendégek pihenhettek, ha a víz túlságosan meleg lenne a számukra.

Az eredeti felálláshoz képest a következő változásokat kell végrehajtani.

- A Fő oldal – a hivatkozások szerkesztése: az új helyek beazonosítása és ugrás;
- A Golfpálya és a Szauna oldalak szükségtelenek és el kell távolítani;
- Egy új oldalt kell beilleszteni, ami kiemeli a trópusi kertek látványosságait.
- Egy, a könyvtárat bemutató új oldalt kell beilleszteni.

Git: a legvégső visszavonás gomb

A Git használata a projektben

A weblap fájljait elkészítettük és a munkakönyvtárban helyeztük el, egy források nevű könyvtárral együtt, ami weblapon használt képeket, videoklipeket és hangfájlokat tartalmazza. A könyvtár kilistázva így néz ki:

```
ls
accom.html  golf.html  lounge.html  resources/  tariffs.html  contact.html  index.html
pool.html  sauna.html
```

A Git rendelkezik egy status paranccsal a tárolók pillanatnyi állapotának jelzésére.

```
git status
On branch master
```

```
Initial commit
```

```
Untracked files:
```

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

```
accom.html
contact.html
golf.html
index.html
lounge.html
pool.html
resources/
sauna.html
tariffs.html
```

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

Nagyon kevés információ jelenik meg itt.

A kimenet felsorolja azokat a fájlokat, amiket a Git ismer és nem kapott utasítást az ignorálásukra. Az összes fájlunk pirossal szerepel a listában, mint nem követett. A forráskönyvtár törtjellel zárul, jelezvén, hogy könyvtár. A Git segítőkészen jelzi, hogy „git add” paranccsal kell a fájlok követését elindítani. Mivel azt akarjuk, hogy a Git ezeket a fájlokat kövesse, használhatjuk a „*” dzsókart (pont is használható - „git add .”) az összes fájlnev helyett.

```
git add *
```

Most ismét adjuk ki a status parancsot.

```
git status
```

```
On branch master
```

```
Initial commit
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file: accom.html
new file: contact.html
new file: golf.html
new file: index.html
new file: lounge.html
new file: pool.html
new file: resources/golf.jpg
new file: resources/island-hotel.jpg
new file: resources/island.jpg
new file: resources/lounge.jpg
new file: resources/pool.jpg
new file: resources/sauna.jpg
new file: sauna.html
new file: tariffs.html
```

A status parancs most kilistázza az összes fájlt, benne a forráskönyvtár fájljait mint újakat és zölden jeleníti meg azokat.

Az „Initial commit” megjegyzés közli velünk, hogy még semmi sincs a tárolóban és bármely fájl, amit commit-olunk kiinduló commit lesz. Ezek a fájlok bekerültek a staging területre, és rájuk mondjuk, hogy stage-geltek (staged). Innen vagy kivehetjük (unstaged), vagy commit-olhatjuk a tárolóba.

Amikor commit-oljuk a fájlokat egy commit üzenet (commit message) összeállítását fogja kérni, ami leírja a commit-olás okát. A Git az üzenet nélkül nem lép tovább. Használhatjuk az -m opciót parancssorban az üzenet összeállításához, vagy a Git nyit egy szerkesztőt, ami az üzenetünket mutatja, és kurzort a szövegbevitelre készen. Az alap szerkesztő a vim, de a Git beállító fájljában megváltoztatható úgy, ahogy az e-mail címünket is megadtuk. Itt az -m opciót fogom használni.

```
git commit -m 'Pre-launch web pages and images'
```

```
[master (root-commit) f81a260] Version 1.0.0 Pre-launch web pages
and images
```

```
14 files changed, 90 insertions(+)
create mode 100755 accom.html
create mode 100755 contact.html
create mode 100755 golf.html
```


Git: a legvégső visszavonás gomb

```
create mode 100755 index.html
create mode 100755 lounge.html
create mode 100755 lounge.html
create mode 100755 pool.html
create mode 100755 resources/golf.jpg
create mode 100755 resources/hotel.jpg
create mode 100755 resources/island.jpg
create mode 100755 resources/lounge.jpg
create mode 100755 resources/pool.jpg
create mode 100755 resources/sauna.jpg
create mode 100755 sauna.html
create mode 100755 tariffs.html
```

git status

```
On branch master
nothing to commit, working directory clean
```

Most már van munkatárolónk és bármikor visszatérhetünk a jelenlegi helyzethez.

A Git még vezet naplót is, ami valahogy így néz ki:

git log

```
commit f81a260f3e2a5fe52d12e594f25881f84e5e6022
Author: pete <pete@connaught>
Date: Tue Jul 5 13:24:22 2016 +0100
```

Version 1.0.0 Pre-launch web pages and images

Az SHA-1 referencia barna színben jelenik meg, és hogy mi változtattunk, mikor és miért. Ezt minden commit-olás esetén megismétlődik. Az SHA-1 referenciával lehet az adott pillanatképet elérni, de az első néhány karakter, 6, vagy 7 általában elegendő, miként azt a commit parancs kimenetén is láthatod.

Ezen munkakönyvtár eléréshez három parancsot használtunk:

```
git init
git add *
git commit -m 'Version 1.0.0 Pre-launch web pages and images'
```

Ez minden. Volt néhány parancs, amivel a Git-et az elején beállítottuk és alkalmaztuk még a status és log parancsot is, hogy némi információt nyerjünk, de csak ez a három parancs kellett a tároló létrehozásához. Valóban ilyen könnyű.

A projekt módosítása

A tesztközönség nem volt túlságosan lenyűgözve a javasolt üdülönktől, ezért kell némi változtatás.

- Két új fájlt készítettünk a munkakönyvtárban: gardens.html és library.html.
- Az index.html, azaz a fő oldalt a változások tükrözésére szerkesztettük.
- Néhány új képet kellett a forráskönyvtárhoz hozzáadni.
- Két fájlt törölni kellett a munkakönyvtárból és a Git által követett fájlok listájából.

A fájlok merevlemezeiről törlésére (nyugi, a Git csinálta ezt) és a Git általi követés leállítására a Git „rm” parancsát alkalmaztuk, hogy kézben tartsuk a „munkakönyvtárban” zajló eseményeket.

```
git rm golf.html sauna.html
rm 'golf.html'
rm 'sauna.html'
```

```
git status
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)
```

```
deleted:    golf.html
deleted:    sauna.html
```

Végrehajthatjuk ezeket a változtatásokat és egy új SHA-1 referencia készül a projektben ezekre a változásokra hivatkozásul. Vedd észre, hogy semmi sem történik addig, amíg a commit parancsot ki nem adjuk.

```
git commit -m 'Version 1.0.1 golf and sauna pages removed'
```

```
[master 9638a4b] Version 1.0.1 golf and sauna pages removed
2 files changed, 20 deletions(-)
delete mode 100755 golf.html
delete mode 100755 sauna.html
```

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

Git: a legvégső visszavonás gomb

```
gardens.html
library.html
resources/gardens.jpg
resources/library.jpg
```

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

Az új fájlokat hozzá kell adni a munkakönyvtárhoz és azután a Git által követett fájlok listájához. A fájlok elválasztása egy szóközzel történik.

```
git add gardens.html library.html resources/gardens.jpg resources/library.jpg
```

```
git status
```

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

```
new file:   gardens.html
new file:   library.html
new file:   resources/gardens.jpg
new file:   resources/library.jpg
```

Négy fájl kész a commit-oláshoz. A fő oldal index.html-jét is módosítani kell az előtt, mielőtt commit-olnánk.

```
git status
```

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

```
new file:   gardens.html
new file:   library.html
new file:   resources/gardens.jpg
new file:   resources/library.jpg
```

Changes not staged for commit:

(use "git add <file>..." to update what will be committed)

(use "git checkout -- <file>..." to discard changes in working directory)

```
modified:  index.html
```

Most hozzáadjuk ezt a fájlt a staging területhez.

```
git add index.html
```

```
git status
```

On branch master

Changes to be committed:

(use "git reset HEAD <file>..." to unstage)

```
new file:   gardens.html
modified:   index.html
new file:   library.html
new file:   resources/gardens.jpg
new file:   resources/library.jpg
```

Láthatjuk, hogy most már öt fájl vár készen a commit-olásra, tehát commit-olunk.

```
git commit -m 'Version 1.1.0 - golf and sauna removed, gardens and library added. Main page updated'
```

```
[master 33d5349] Version 1.1.0 - golf and sauna removed, gardens
and library added. Main page updated
5 files changed, 22 insertions(+), 2 deletions(-)
create mode 100755 gardens.html
create mode 100755 library.html
create mode 100755 resources/gardens.jpg
create mode 100755 resources/library.jpg
```

Most már három pillanatképünk van a programunkról, amik bármikor visszaállíthatóak: a jelenlegi verzió 1.1.0; a kezdő változat 1.0.0 és a közbenső állapot, amit akkor mentettünk, amikor a golf és a szauna oldalt eltávolítottuk. Noha néhány fájlt töröltünk, helyreállíthatóak. Ez úgy hangzik, mintha a Git tároló nagyon hamar hatalmasra duzzadhatna, de valójában a Git nagyon gazdaságos, mivel nem fájl szinten dolgozik, hanem a változásokat a fájlok adatai között tartja nyilván.

Visszatérés az előző kiadáshoz

Épphogy nekiláttál az árlista oldalának frissítéséhez, egy új masszázsközpont oldalát írnád a könyvtár lecserélésére, mikor telefonon hívnak. Mr. Big van a vonalban, a fizetést osztó ember és dühös. „Nem ezzel bíztam meg” harsogja. „Szeretem a golfpályát. Rakd vissza, ahogy volt, különben kirúglak!”

Anyám! Néhányat töröltünk azokból a fájlokból.

```
ls
```

```
accom.html gardens.html library.html pool.html tariffs.html
contact.html index.html lounge.html resources/
```

Nincsenek ott a golf, vagy a szauna fájljai.

Git: a legvégső visszavonás gomb

Ez nem gond, könnyen vissza (vagy előre) ugorhatunk az időben az SHA-1 kulcs segítségével. A Git naplójában látjuk, hogy az eredeti commit kulcsa f81a260 kezdetű. A Git a jelenlegi pozíciódat HEAD-ként jelöli a commit-fában, tehát csak vissza kell lépni a HEAD-ből oda, ahová csak akarsz menni. A visszapörgetés az f81a260 és a HEAD között egy lépésben történik, anélkül hogy a folyamat minden egyes állomásánál üzenetben kérdezne, ehhez a Git revert parancsát --no-commit jelzővel kell használni. Ha már a kívánt helyen vagyunk, commit-olhadjuk ezt az állapotot a Git-nek.

```
git revert --no-commit f81a260..HEAD
```

```
git commit -m 'Reverted to initial commit'
```

```
[master f5210a8] Reverted to initial commit
5 files changed, 4 insertions(+), 4 deletions(-)
rename gardens.html => golf.html (79%)
delete mode 100755 resources/gardens.jpg
delete mode 100755 resources/library.jpg
rename library.html => sauna.html (82%)
```

Most listázva a munkakönyvtár fájljait ezt kapjuk:

```
ls
accom.html golf.html lounge.html resources/ tariffs.html
contact.html index.html pool.html sauna.html
```

A golf- és a szaunafájlok visszatértek, úgy tűnik, megtarthatod a munkádat. És, amikor már az ember megnyugodott, újból végrehajthatod a változtatásaidat néhány paranccsal.

Katasztrófaelhárítás

A Git sosem veszíti el, rakja át, vagy rongálja meg a projekted követett fájljait, de mindig megvan a veszélye a tároló buta (?) felülírásnak, vagy törlésnek, illetve a merevlemez meghibásodásának. A Git-et többes tárolórendszerhez tervezték, így minden felhasználónak lehet saját példánya a teljes tárolóról. Ezen másolatok bármelyikéből a teljes projekt és az időben összes változása újra előállítható. Katasztrófa elkerülésére célszerű másolatot készíteni máshol, másik könyvtárban, másik partíción, vagy másik meghajtón, másik számítógépen, miként az asztali gép, laptop összeállításommal tettem.

Az új hely a laptopom home könyvtára lesz, innen hozzáférék az asztali gépemhez. Neked csak annyi kell, hogy legyen hol létrehozni a tárolót és

ahonnan elérheted az eredeti tárolót. Az „elysium” host-nevű laptopom home könyvtárából, az asztali gépet az /mnt/connaught-home-ba csatolva kiadom:

```
git clone /mnt/connaught-home/pete/sunshine_island
```

```
Cloning into 'sunshine_island'...
done.
```

```
cd sunshine_island
```

```
ls
resources/      contact.html    index.html      pool.html       tariffs.html
accom.html      golf.html       lounge.html     sauna.html
```

A klónozás folyamat kifejezetten gyors és most már dolgozhatok a projekten bármely gépen.

A dolgok szinkronban tartása

Magától értetődő, hogy a projektről két példányt bírva és egymástól függetlenül dolgozva rajta kétségbeejtő kavardást okozhat a projekt eltérő fázisban lévő részeiben. Éppen ez az, aminek az elkerülésére tervezték és ebben a dologban kiemelkedően jó. A Git képes a világban meglévő több ezer tároló közötti különbségeket kezelni a projekt fejlesztésének több évén átívelően és képes visszaállítani bármely korábbi commit állapotot. Ehhez képest a mi egyszerű egyfelhasználós, két helyszínes felállásunk egy sima séta a parkban.

A „connaught” asztali gépemén néhány képfájl hozzáadtam a forráskönyvtárhoz és ezután ellenőriztem a tároló állapotát.

```
git status
```

```
On branch master
```

```
Untracked files:
```

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

```
resources/gardens.jpg
resources/island2.png
resources/island_beach.jpeg
resources/library.jpg
```

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

Hozzáadtam ezeket a fájlokat a tárolóhoz, majd ismét ellenőriztem a státuszt.

Git: a legvégső visszavonás gomb

```
git add *
```

```
git status
```

```
On branch master
```

```
Changes to be committed:
```

```
(use "git reset HEAD <file>..." to unstage)
```

```
new file:   resources/gardens.jpg
new file:   resources/island2.png
new file:   resources/island_beach.jpeg
new file:   resources/library.jpg
```

Az eredménnyel elégedett lévén elhatároztam, hogy commit-ot csináljak. Ezeket a fájlokat később adom hozzá a HTML lapok forráskódjához.

```
git commit -m 'added some images to the resources folder'
```

```
[master f5210a8] added some images to the resources folder
4 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 resources/gardens.jpg
create mode 100644 resources/island2.png
create mode 100644 resources/island_beach.jpeg
create mode 100644 resources/library.jpg
```

A Git mutatja az új bevittelt (plusz a többieket):

```
git log
```

```
commit f5210a8c70bb35b386c10c470afff1ebbb23246
Author: pete <pete@connaught>
Date: Tue Jul 5 13:50:42 2016 +0100
    added some images to the resources folder
```

Később eldöntöttem, hogy az „elysium” laptopommal dolgozom be a képeket a HTML fájlba. A laptopon lévő tárolót az asztaliról „klónoztam”, tehát a tároló az asztalra hivatkozik „eredetiként” (origin). Ez az alapállás. Ahhoz, hogy a laptopról bármilyen változást az asztalra átvigyek a pull parancsot kell használnom.

```
git pull origin
```

```
remote: Counting objects: 16, done.
remote: Compressing objects: 100% (16/16), done.
remote: Total 16 (delta 5), reused 0 (delta 0)
Unpacking objects: 100% (16/16), done.
From /mnt/connaught-home/pete/sunshine_island
f81a260..f5210a8 master -> origin/master
```

```
Updating f81a260..f5210a8
```

```
Fast-forward
```

```
resources/gardens.jpg      | Bin 0 -> 29671 bytes
resources/island2.png     | Bin 0 -> 88053 bytes
resources/island_beach.jpeg | Bin 0 -> 6790 bytes
resources/library.jpg     | Bin 0 -> 280028 bytes
4 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 resources/gardens.jpg
create mode 100644 resources/island2.png
create mode 100644 resources/island_beach.jpeg
create mode 100644 resources/library.jpg
```

A laptop tárolója most pontosan olyan állapotban van, mint amilyenben az asztali.

Helyi és távoli tárolók

A tárolók közötti információtovábbításra a Git-ben három parancs van: fetch, pull és push. A fetch a változásokat a távoli tárolóból kapja. A pull, ahogy láttuk, veszi a változásokat és beteszi a helyi tárolóba, a push egyszerűen a pull ellentéte.

A Git „remote” (távoli) parancsa kilistázza a távoli tárolókat és a -v opcióval sokkal részletesebb kimenetet ad. A laptop klónozott tárolója automatikusan „origin” (eredet) nevet ad az elődjének, de connught-ra – asztali – visszatérve, meg kell adnunk a laptop teljes elérési címét (URL), hogy ugyanezt a műveletet végrehajthassuk. Ez nem kényelmes és hibát okozhat. Ezért a Git lehetővé teszi számunkra alias használatát a távoli helyekkel kapcsolatban a „remote add” paranccsal.

```
git remote add elysium /mnt/elysium-home/pete/sunshine_island
```

```
git remote
elysium
```

```
git remote -v
```

```
elysium      /mnt/elysium-home/pete/sunshine_island (fetch)
elysium      /mnt/elysium-home/pete/sunshine_island (push)
```

A felállításunkban egy URL van minkét irányban az átadásra. Ismét használhatjuk a git pull parancsot, ahogy korábban, ám az elysiumot connaught-ra cserélve.

A laptopon elvégezve némi munkát, most szeretném az asztalit frissíteni a munka folytatásához, azért lehúzó (pull) a változásokat. Ám ez alkalommal, meg kell jelölnöm „master”-t az ág (branch) a húzáshoz (az ágról hamarosan).

```
git pull elysium master
```

```
From /mnt/elysium-home/pete/sunshine_island
```

Git: a legvégső visszavonás gomb

* branch master -> FETCH_HEAD

Updating f5210a8..bdf2a88

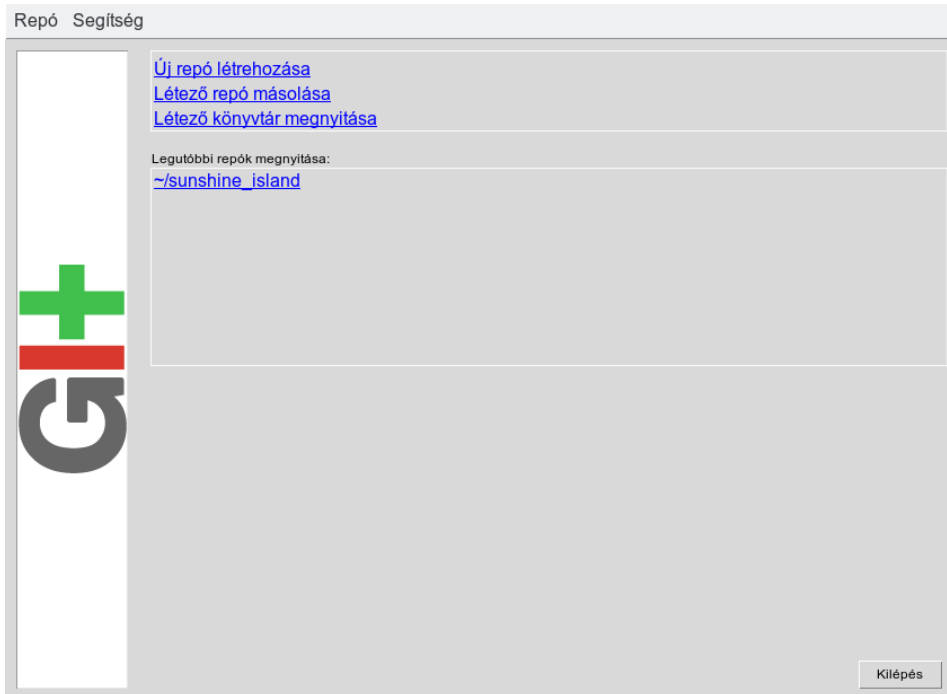
Fast-forward

index.html | 3 +++

1 file changed, 3 insertions(+)

Git gui – egy grafikus kezelőfelület

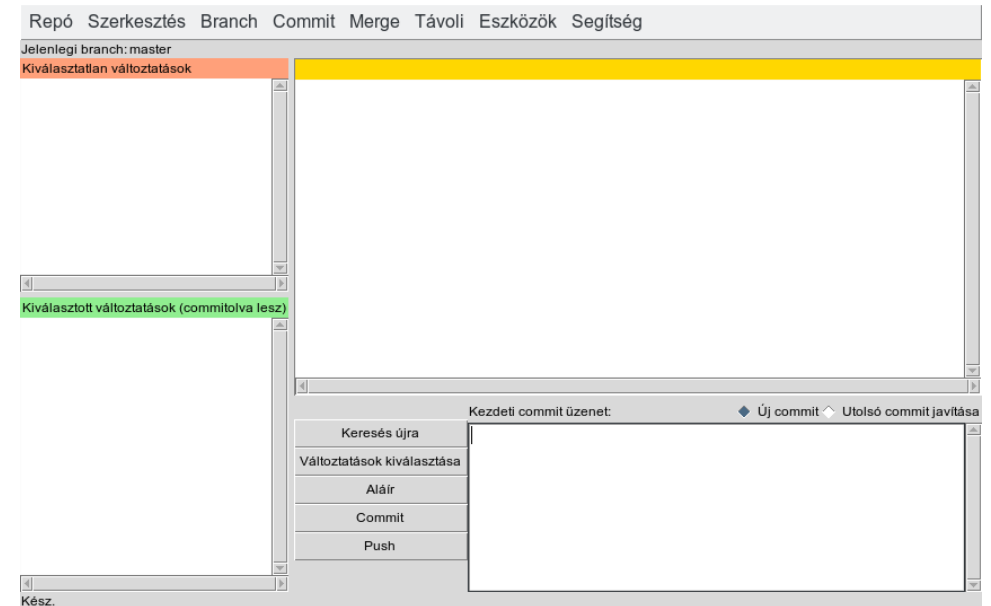
Számos grafikus felület érhető el a Git-hez, de a git gui (két szó), amit itt be fogok mutatni, része a Git telepítőjének. A PCLinuxOS tárolóiban továbbiak is vannak, mint a Qgit és a Gigggle. Az Eclipse IDE (integrált fejlesztőrendszer) szintén támogatja a Git-et, mint egy önálló „perspektívát”. Az Eclipse is elérhető a PCLinuxOS tárolóiban.



A git gui történetesen egy commit-, branch-készítő és összevonó eszköz, és hozzáfér egy másik, gitk nevű eszközhöz is, ami az előzményeket jeleníti meg grafikusán. Ezekkel az eszközökkel majdnem mindet, ami kell, elvégezhetünk.

Az alkalmazást terminálban, vagy Alt+F2 futtató ablakban git gui-t beírva, vagy az asztal fő menüjébe készített bejegyzés segítségével indíthatod. Alapbeállítás szerint ez nem készül el.

Az opciók eléggé magától értetődőek, és mivel korábban már használtam, a sunshine-island tárolót a tároló előzmények rész felajánlja. A sunshine_island-ot kiválasztva ezt a nem túl tartalmas ablakot kapom.



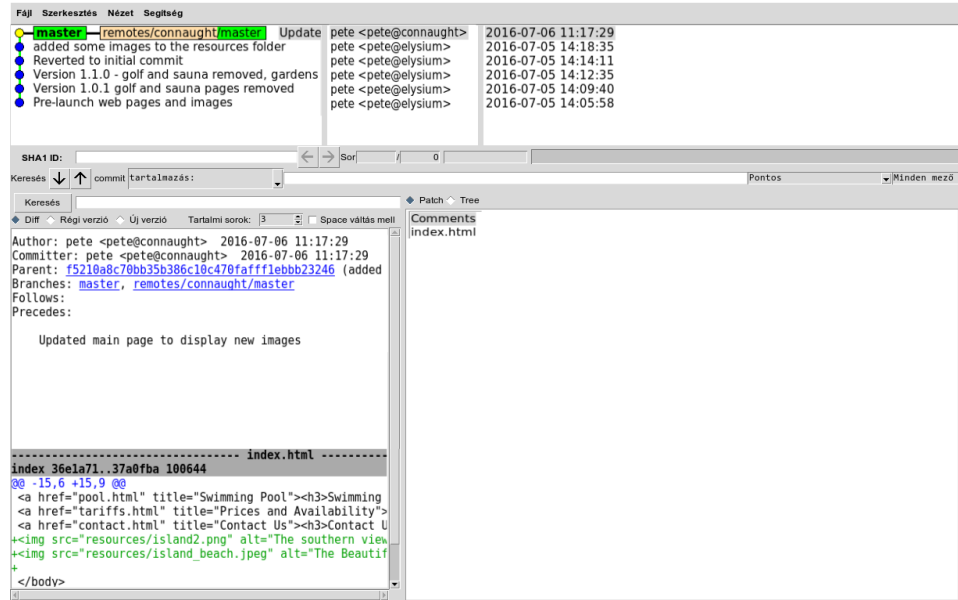
Pár perct a menüopciók tanulmányozására szánva elég jó képet kaphatsz arról, hogy az eszköz mire jó, és a parancssori terminálunk opciói közül sokat felismerhetsz. A bemutató projektünk előző szakaszában a Nagy Ember utasított az eredeti elképzelésekhez történő visszatérésre, amit megtettél és ezt követően hozzáadtál néhány képet. A szíved mélyén tudod, hogy a Nagy Ember hamarosan hasonlóan fog vélekedni, mint te, ezért elhatároztad, hogy a jelenlegi mellett elkezdesz dolgozni egy alternatív változaton is. Ennek érdekében készítesz egy leágazást (branch).

Vedd úgy, hogy egy ág egy elágazás a projektéd útvonalán. Mint az igazi utakon az elágazások, az egyes branch-ek eltérő irányba visznek el, még ha valahol a jövőben ismét összefuthatnak.



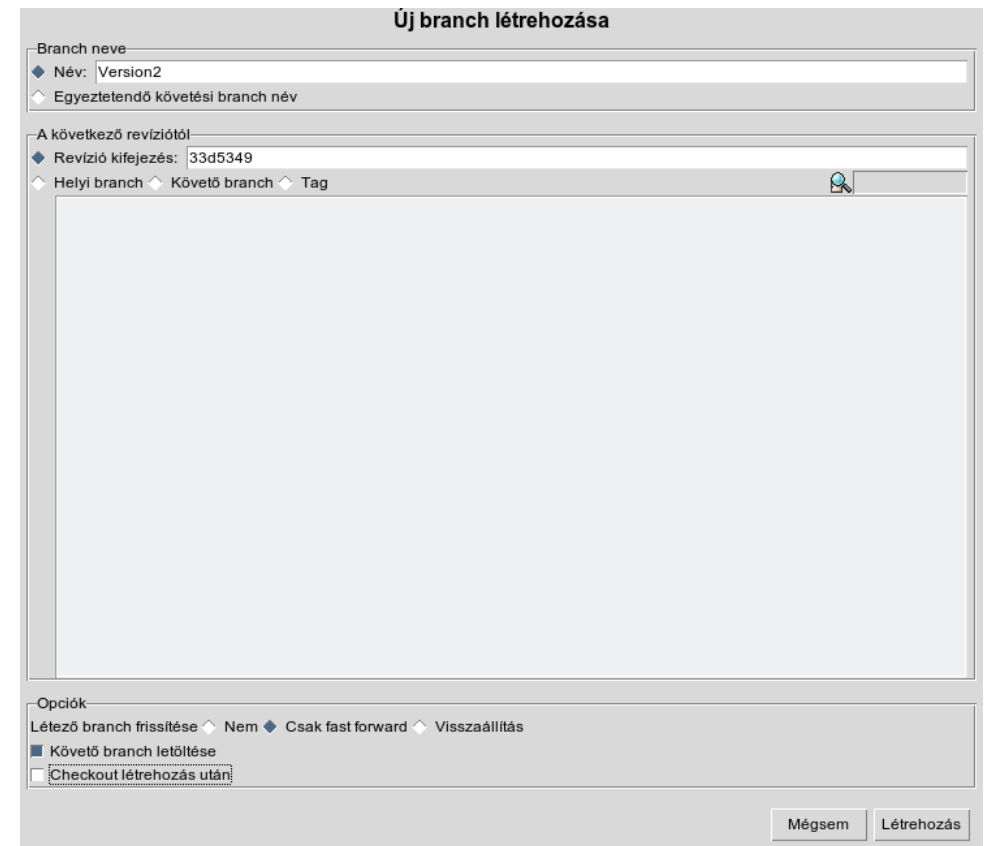
Git: a legvégső visszavonás gomb

Grafikus környezetben sokkal egyszerűbben bemutatható a tároló és a brach-konceptió, és ezért halasztottam idáig a branch-ek elmagyarázását. A menüből válaszd a Repó → Az összes branch történetének vizualizálása. Valami ilyesmit fogsz kapni.



The screenshot shows a Git GUI interface. At the top, there's a commit history list with columns for commit type (e.g., Update, Reverted), author (pete <pete@connaught>), and date (2016-07-06 11:17:29). Below this, the SHA1 ID is shown as 'commit'. The main area displays a diff for 'index.html', showing the commit message 'Updated main page to display new images' and the corresponding HTML code changes, including links to 'pool.html', 'tariffs.html', and 'contact.html', and image tags for 'island2.png' and 'island_beach.jpeg'.

Ez a lap sokkal érdekesebb. Az ablakban balra fent láthatjuk mind a helyi, mind a távoli tárolókat és a commit összes előzményét. Zölddel jelölve az egyes tárolók jelenlegi branch-ei, ami egyben a master is e pillanatban, mivel ez a Git által alapból létrehozott és még nem csináltunk másikat. Bármely sorra kattintva az előzményekben, az egyes panelek tartalma megváltozik, az adott commit-nak megfelelően. A jobb alsó panel fölött két rádiókapcsoló található, a Patch és a Tree. A Patch mutatja az előzményekben adott ponthoz tartozó változásokat, a Tree a tárolóban abban a munknakönyvtárban található fájlokat. Egy fájlra kattintva bal oldalt megjelenik annak a fájlnak a tartalma (a képek itt nem jelennek meg, csak a QGit-nél, ami egy nagyszerű tulajdonság). Kattints a „Version 1.1.0...” kezdetű commit-üzenet sorára. Ez az a pillanatkép, amivel az új branch-ünket indítani akarjuk. Másold le, vagy jegyezd fel az SHA-1 első 6-7 karakterét. Zárd be az ablakot és a git gui-ban válaszd a Branch → Létrehozás-t.



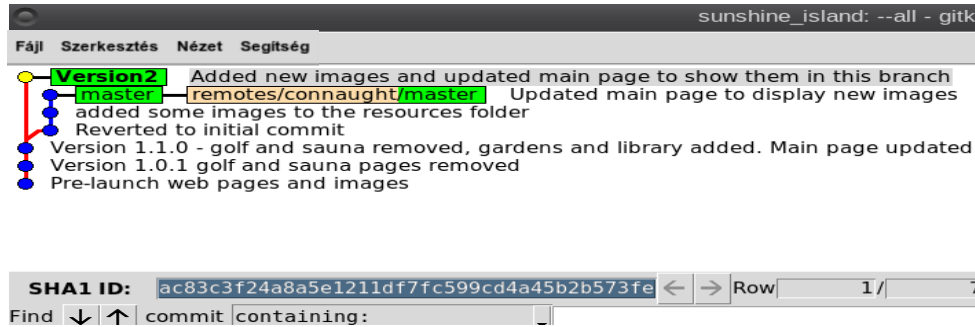
The screenshot shows the 'Új branch létrehozása' (Create new branch) dialog box. It has a title bar and several sections: 'Branch neve' (Branch name) with a text field containing 'Version2' and a dropdown for 'Egyeztetendő követési branch név' (Branch to track); 'A következő revíziótól' (From which revision) with a dropdown for 'Revízió kifejezés: 33d5349' (Revision specifier); and 'Helyi branch' (Local branch) and 'Követő branch' (Tracking branch) dropdowns. At the bottom, there are 'Opciók' (Options) including 'Létező branch frissítése' (Update existing branch) with radio buttons for 'Nem' (No), 'Csak fast forward' (Only fast forward), and 'Visszaállítás' (Reset), and a checked 'Követő branch letöltése' (Fetch tracking branch) option. There is also an unchecked 'Checkout létrehozás után' (Checkout after creation) option. 'Mégsem' (Cancel) and 'Létrehozás' (Create) buttons are at the bottom right.



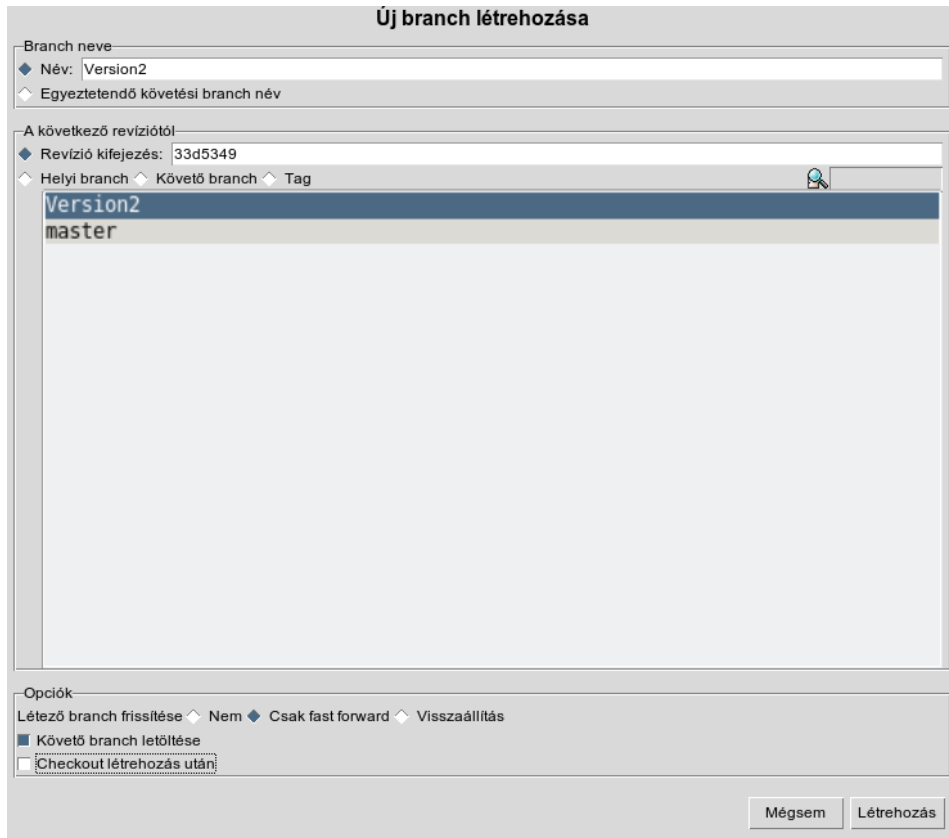
A megjelenő párbeszédben adj nevet az új branch-nek, a „Revízió kifejezés”-re kattintva írd be az SHA-1 referenciát, majd kapcsold ki a Checkout létrehozás után”-t. Rendesen bekapcsolva szoktam hagyni, de a bemutató kedvéért kikapcsolom.

Git: a legvégső visszavonás gomb

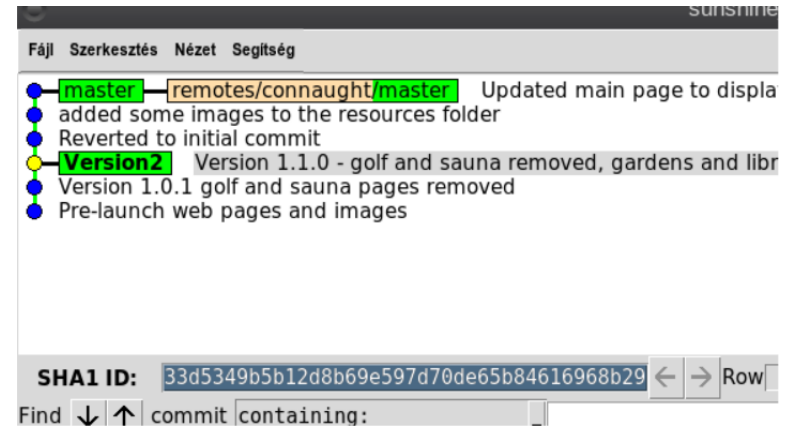
Kattints a Létrehozásra. Mintha semmi sem történt volna, de az „Összes branch történetének vizualizálása” ablakot hívva láthatjuk az új ágat a helyi tároló kiválasztott revíziós pontjánál.



Zárd be ezt az ablakot és a git gui menüből válaszd a „Branch → Checkout”-ot (ellenőrzés), válaszd ki a version2 branch-et , kattints a checkout-ra és ismét nyisd meg a vizualizáló ablakot.

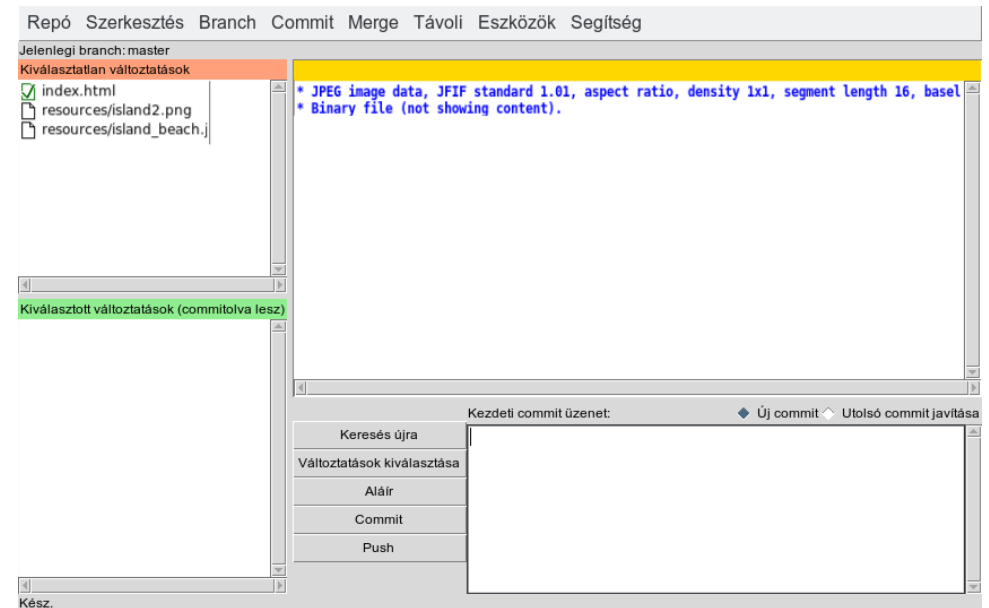


A version2 branch most félkövén jelenik meg jelezvén, hogy most az aktuális branch-ben van, és az általunk választott revíziós helyzetben van. A távoli tároló a master branch-ben marad. Fontos annak a belátása, hogy a checkout parancsra a kívánalmak szerint megváltozik a munkakönyvtár tartalma.



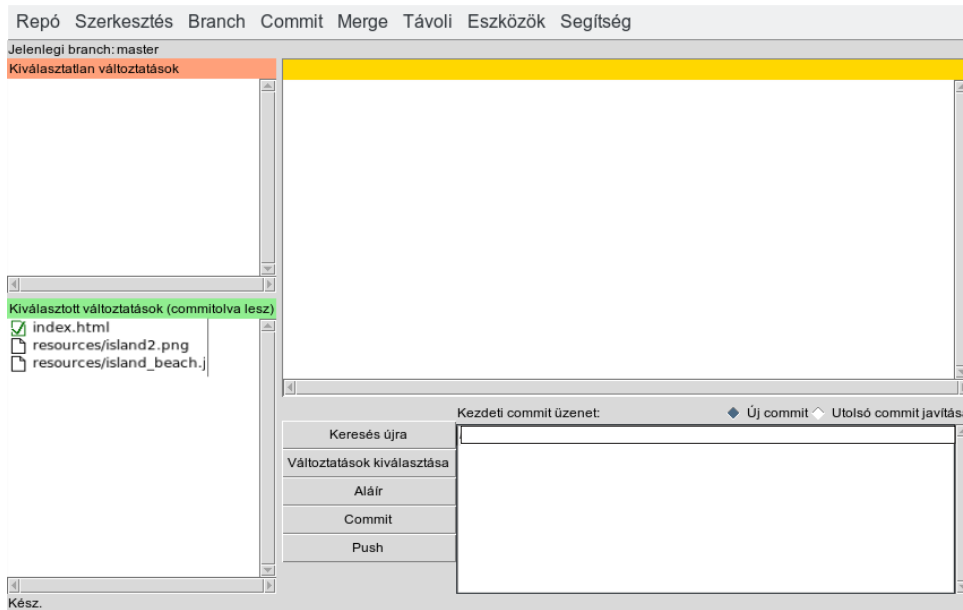
Most végrehajthatunk némi változtatást, amik csak a version2 branch-ben jelennek meg, miközben a megosztott weblapon található, jelenleg használt master branch nem változik.

Ismét töltsd be az új képeket a forráskönyvtárba és szerkeszd meg a fő oldalt úgy, hogy mutassa azokat. A git gui ablakában nyomj F5-öt a képernyő frissítésére és a változások unstaged-ként (közé nem tett) jelennek meg.

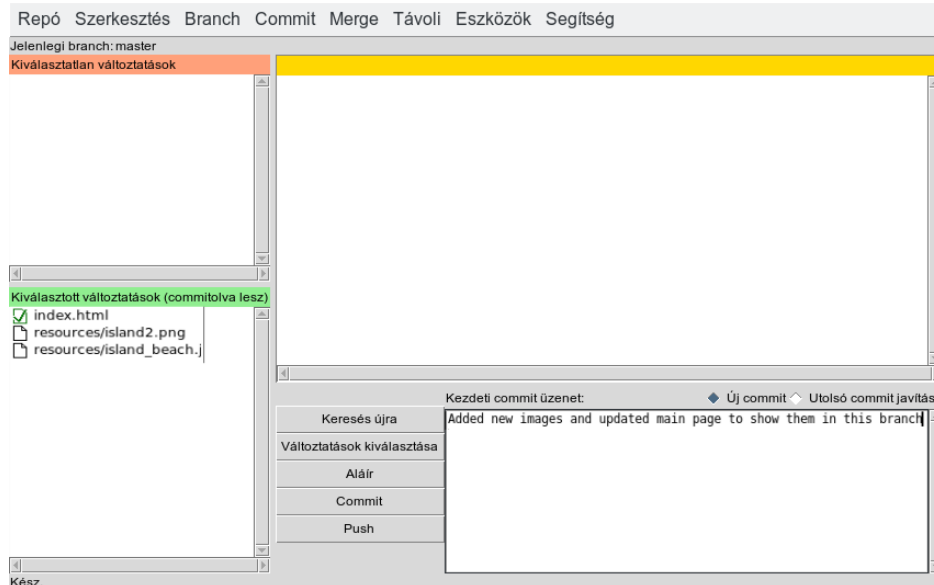


Git: a legvégső visszavonás gomb

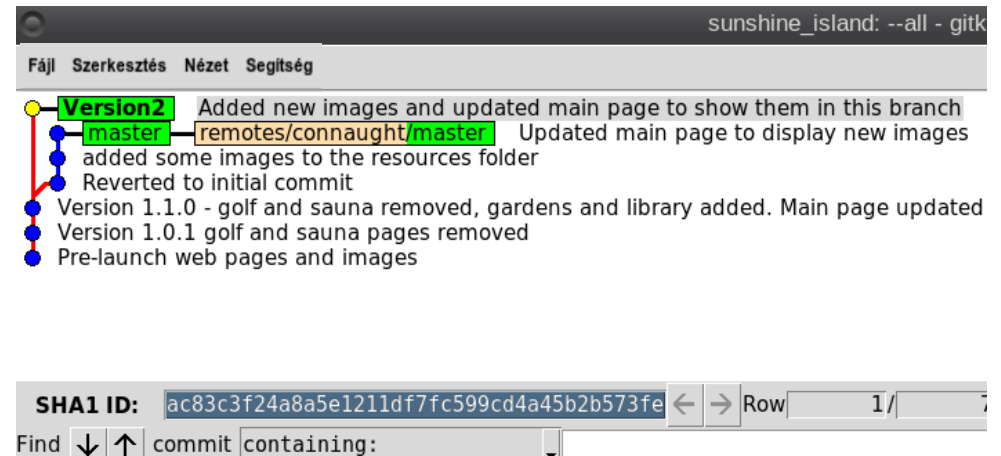
Jelöld ki a megváltozott fájlokat és válaszd a Commit → Változtatások kiválasztása, hogy commit-olásra kirakjuk, ezzel a bal alsó panelbe kerülnek.



Írj egy megfelelő commit-üzenetet a jobb alsó négyzetbe és válaszd a „Commit → A kiválasztott commitolása”-t.

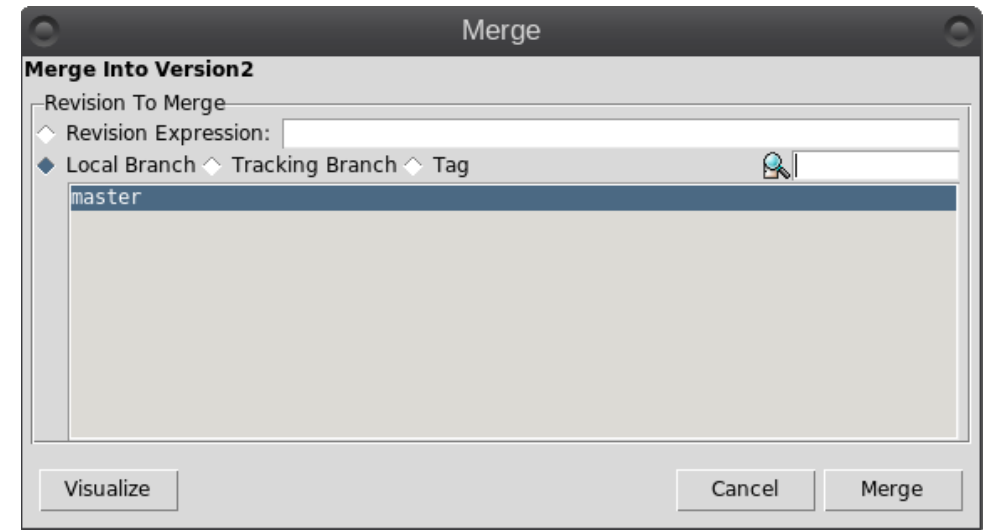


A vizualizációs ablakban most már tisztán látszik a branch és a commit hatása.



Továbbdolgozhatunk ezen a branch-en, de ha változtatni kell a most megjelenített weblapon, egyszerűen kijelöljük a master branch-t. Emellett a változásokat le is húzhatjuk (pull) a távoli tárolóról, vagy a git gui Távoli → Push opciójával áttölthetjük a távoli tárolóba.

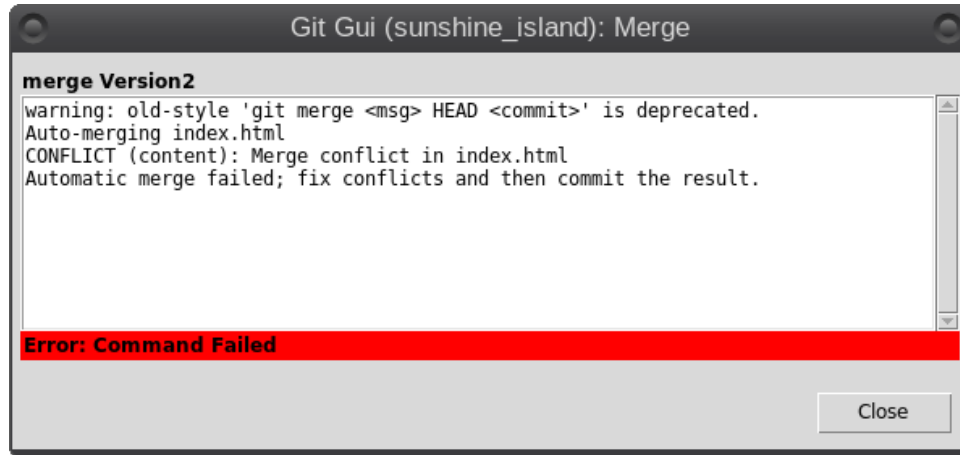
Adott esetben szükség lehet a két ág összevonására, hogy az azokban végrehajtott változtatásokat érvényesítsd. Könnyen megteheted a git gui Merge menüopciójával.



A megjelenő párbeszédben az aktuális branch már kiválasztott és csak azt kell megjelölni, hogy mely ágot olvasztod bele. Van még egy nagyon hasznos Vizualizálás gomb, ami lehetővé teszi az eredmény megtekintését, még mielőtt ténylegesen végrehajtaná az összevonást.

Git: a legvégső visszavonás gomb

A bemutató projektünkben keveset nyerünk az ágak összevonásából, de mindenképpen megcsináljuk.



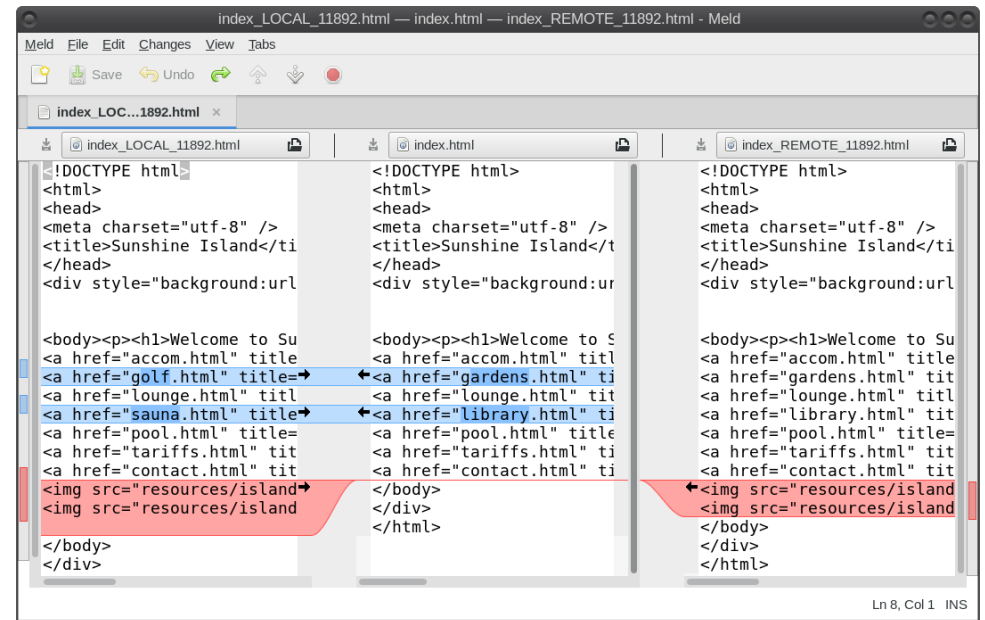
Hú ha! Nem sikerült, de a helyzet koránt sem olyan rossz, mint elsőre tűnik. A figyelmeztetés annyit jelez, hogy ütközés van az index.html fájlban és kézzel kell megoldanunk a gondot, majd a javított fájlt commit-oljuk a merge véglegesítésére. A Git rendelkezik egy mergetool-nak nevezett eszközzel, ami a Linux diff parancsához hasonló számos eszköz egyikét használja az ütközések megjelenítésére. A személyes kedvencem ezek közül a meld, ami a PCLinuxOS tárolójában megtalálható. Hozzáadhatod a .gitconfig fájlodhoz ennek a két sornak a megváltoztatásával, vagy beírásával:

`merge.tool=meld` – *Megj.: létre kell hozni egy [merge] fejezetet és oda beírni ezeket*
`diff.tool=meld`

A Branch menüben „Checkout...”-old a master branch-et, hogy az a jelenlegi legyen, majd add ki ezt a parancsot:

`git mergetool`

Ezt az ablakot jeleníti meg, vagy ha nem a meld-et használod, akkor valami hasonlót.

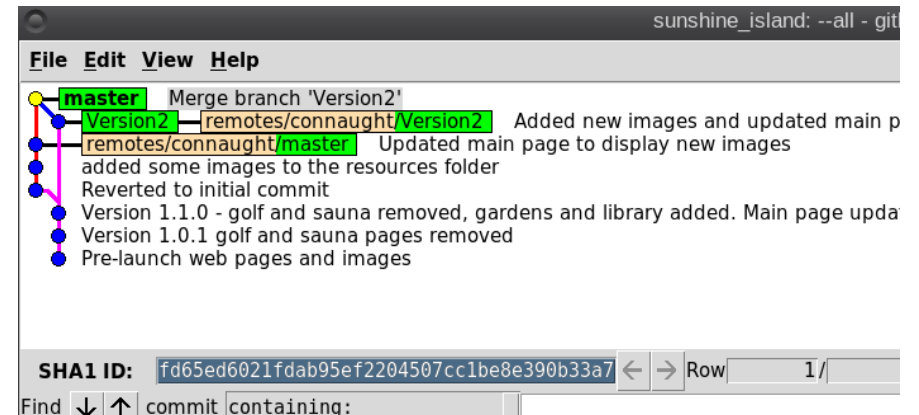


Világosan jelzi az ütközéseket és a középső panelt szerkesztve meghatározhatod, mi kerüljön az összevonásba, majd commit-tal tedd teljessé az összevonást.

Merging:
`index.html`

Normal merge conflict for 'index.html':
{local}: modified file
{remote}: modified file

A végső commit végrehajtása után az időbeni áttekintés valahogy így néz ki:



A Git nagyon hatékony eszköz és én csak nagyon az alapokat mutattam be a képességeiből, bár sokaknak ennyi is elég. Ha egyszer megismered az alapokat és párszor használtad már a Git-et, könnyű lesz dolgozni vele. Ez a cikk a git használatával készült, ami sokat segített a demo projekt képeinek és terminál kimeneteinek szinkronizálásában.

Számos módon végrehajthattam volna a bemutató során végzett műveleteket és a Git-ínyencek valószínűleg megvetnek majd a minimalista megközelítem miatt, de működik és én így csinálom. A felvázolt lépések nem alkalmasak többfelhasználós együttműködési környezetben, mivel nem foglalkozik mások bedolgozásával, de egyedüli felhasználó esetén jól működik.

Ha többet akarsz megtudni, akkor a [git weblapján](#) található dokumentáció elég jó és tartalmaz egy hivatkozást a Scott Charon és Ben Straub által írt „Pro Git” könyvre. Creative Commons licenc alapján készült, online olvasható és nagyon átfogó, de kezdőknek is használható.



PCLinuxOS-Cloud secure private simple-to-use
Sign up TODAY! pclosusers.com/services-signup.php



CHIMPBOX
The chimpbox packs a punch. Zero noise, small footprint and low power usage.
<http://chimpbox.us>



Setup Error

Microsoft Windows has encountered an unrecoverable error. Please reboot and install PCLinuxOS.

OK



The PCLinuxOS Magazine Special Editions!

Windows Migration Guide September 2013

Enlightenment Special Edition MAY 2011

The NEW PCLinuxOS Magazine Fall 2010

The KDE 4 SC Special Edition

GTK Lightweight Desktops: Xfce & LXDE Special Edition November 2010

The NEW PCLinuxOS Magazine

Command Line Interface Intro Special Edition October, 2010

Openbox Special Edition March, 2012

Get Your Free Copies Today!