Adatbányászat a Gyakorlatban

Kuknyó Dániel

2024/25/1

1. Gyakorlat: Verziókezelés

1 GitHub

Regisztráció a GitHub-ra

www.github.com

Git telepítése

Windows

Git Bash (parancssoros Git) letöltése: https://git-scm.com/downloads
Asztali kliens: https://desktop.github.com/

Linux

A Linux terminálon a következő parancsot kell végrehajtani a `gh` csomag telepítéséhez.

```
(type -p wget >/dev/null || (sudo apt update && sudo apt-get
    install wget -y)) \
&& sudo mkdir -p -m 755 /etc/apt/keyrings \
&& wget -qO- https://cli.github.com/packages/githubcli-archive-
    keyring.gpg | sudo tee /etc/apt/keyrings/githubcli-archive-
    keyring.gpg > /dev/null \
&& sudo chmod go+r /etc/apt/keyrings/githubcli-archive-keyring.gpg
    \
&& echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/
    keyrings/githubcli-archive-keyring.gpg] https://cli.github.com/
    packages stable main" | sudo tee /etc/apt/sources.list.d/github
    -cli.list > /dev/null \
&& sudo apt update \
&& sudo apt install gh -y
```

Konfigurálás

Parancssorban

Név hozzáadása:

```
git config --global user.name "Gipsz Jakab"
```

E-mail cím hozzáadása

```
git config --global user.email "gipsz.jakab@gmail.com"
```

Asztali kliensen

`File` / `Options` / `Accounts` / `Sign in` / Itt hitelesíteni kell a Githubot a böngészővel

2 Git alapok

Új tárhely létrehozása

Parancssorban

```
$ git init
$ git add *.c
$ git add LICENSE
$ git commit -m 'Initial project version'
```

Asztali kliensen

`File` / `New repository`

1. **Name**: A tárhely neve
2. **Description**: A tárhely rövid leírása
3. **Local path**: A tárhely mappája a helyi számítógépen
4. **Git ignore**: A gitignore-hoz adott fájlok nem lesznek feltöltve a tárhelyre. Itt érdemes kiválasztani az adott programnyelvhez tartozó ignore konfigurációt. A `.gitignore` a tárhely gyökerében lévő rejtett, szöveges állomány, amelynek a tartalmában szereplő fájlokat a Git a verziókezelésből ki fogja hagyni. Ilyenek lehetnek a nagyméretű állományok vagy személyes információt, titkos adatokat, lokális konfigurációt tartalmazó fájlok.
5. **License**: Milyen másolhatósági / továbbadhatósági szabályok vonatkoznak a tárhelyre. Ezt legtöbb esetben elhagyhatjuk.

Gitignore szerkesztése

Belépés a tárhely gyökerébe:

```
cd c/Users/Jakab/GitHub/desktop-tutorial
```

Új ignore fájl létrehozása:

```
nano .gitignore
```

Ezután minden sorban egy ignorált fált lehet megadni, vagy a * karakterrel egy általános definíciót megadni:

```
videos/big_video.mp4
passwords/*
*.zip
*.rar
*.jpg
```

Mentés és kilépés a szerkesztőből:

```
ctrl + o
ctrl + x
```

Ezek a definíciók példa jellegűek. A valósan ignorált fájlok és típusok tárhelytől függően változhatnak.

Létező tárhely klónozása

Egy tárolót a `git clone <url>` paranccsal lehetséges klónozni. Például, a libgit2 nevű Git linkelhető könyvtár klónozásának parancsa:

```
git clone https://github.com/libgit2/libgit2
```

3 Fejlesztési ágak kezelése

Fájlok státuszának ellenőrzése

A fő eszköz, amellyel meg lehet határozni, hogy mely fájlok milyen állapotban vannak, a `git status` parancs. Ha közvetlenül egy klónozás után fut ez a parancs, az output hasonló lesz:

```
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
nothing to commit, working tree clean
```

Ha például egy README fájl hozzáadása futtatódik a parancs, a kimenet:

```
$ echo 'My Project' > README
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Untracked files:
(use "git add <file>..." to include in what will be committed)
README
nothing added to commit but untracked files present (use "git add"
    to track)
```

Fájlok indexelése

Új fájlok esetén

Amikor egy új fájl létrejön a munkakönyvtárban, az alapértelmezés szerint nem kerül be a Git verziókezelésébe. A `git add <filename>` parancs segítségével hozzáadható ez az új fájl az indexhez. Ez azt jelenti, hogy a fájl készen áll arra, hogy a következő commit során bekerüljön a verziókezelésbe.

```
$ git add README
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
(use "git restore --staged <file>..." to unstage)
new file:
 README
```

Létező, módosított fájlok esetén

Amikor egy már verziókezelés alatt álló fájl módosításra kerül a munkakönyvtárban, a változtatások alapértelmezés szerint nem kerülnek azonnal az indexbe. A `git add <filename>` parancs segítségével a módosított fájl hozzáadható az indexhez. Ez azt jelenti, hogy a fájlban történt változtatások készen állnak arra, hogy a következő commit során bekerüljenek a verziókezelésbe.

```
$ git add CONTRIBUTING.md
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)
new file:
 README
modified:
 CONTRIBUTING.md
```

Fájlok eltávolítása az indexből

Egy fájl Gitből való eltávolításához a követett fájlok közül kell eltávolítani, majd commitolni kell a változtatásokat. Ezt a `git rm` parancs képes elvégezni. Ez a munkamappából is eltávolítja a fájlt.

```
$ git rm PROJECTS.md
rm 'PROJECTS.md'
$ git status
On branch master
Your branch is up-to-date with 'origin/master'.
Changes to be committed:
(use "git reset HEAD <file>..." to unstage)
deleted:
 PROJECTS.md
```

Egy indexelt fájl törlése az indexből

Ez a parancs nem fogja eltávolítani a fájlt a munkamappából, csak a Git követést fogja leállítani. Ez a változtatás commit és push esetén a távoli tárhelyről törölni fogja a fájlt.

```
$ git reset HEAD CONTRIBUTING.md
Unstaged changes after reset:
M
 CONTRIBUTING.md
$ git status
On branch master
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:
 CONTRIBUTING.md
```

Változtatások commitolása

Commit során a változtatások elmentődnek egy lokális pillanatképbe. A pillanatképet a távoli tárhelyre `push` során lehetséges feltölteni.

```
$ git commit -m "Story 182: fix benchmarks for speed"
[master 463dc4f] Story 182: fix benchmarks for speed
2 files changed, 2 insertions(+)
create mode 100644 README
```

A `-m` paraméter kötelező, és a commit üzenetet (message) adja meg.

4 Távoli tárhelyek

Változtatások lekérdezése

A `fetch` parancs lekérdezi a változtatott fájlok listáját. Ezután letöltődnek azok az ágak és commitok, amelyek még nincsenek meg a lokális tárhelyen.

```
$ git fetch origin
```

Változtatások befésülése

A `fetch` parancs által letöltött változtatások befésülését a `git pull` parancs képes elvégezni. Ez a lokális fájlrenszerben frissíti a fájlokat a távolira:

```
git pull origin main
```

Ebben a példában az `origin` távoli tároló `main` fejlesztési ágáról történik a letöltés.

Új fejlesztési ág létrehozása

A `git branch <branchname>` parancs lokálisan létrehoz egy új fejlesztési ágat. Egyenértékű vele a `checkout` parancs a `-b` argumentummal.

```
$ git checkout testing
```

vagy

```
$ git checkout -b testing
```

Az aktuálisan megjelölt ágat a `HEAD` mutató tartja számon. Ezt bármelyik pillanatban le lehet kérdezni a `git log` segítségével:

```
$ git log --oneline --decorate
f30ab (HEAD -> master, testing) Add feature #32 - ability to add
    new formats to the
central interface
34ac2 Fix bug #1328 - stack overflow under certain conditions
98ca9 Initial commit
```

Váltás ágak között

A váltás a `git checkout` paranccsal történik:

```
$ git checkout testing
```

Ez a parancs a `HEAD` mutatót átmozgatja a `testing` ágra. A Git ágak közötti

váltásra szabályok érvényesek, nem lehet minden esetben megtenni, és némelyik esetben pedig összeolvasztási konfliktusokhoz vezethet.

- **Tiszta munkakönyvtár**: Ágak közötti váltás csak akkor lehetséges, ha a munkakönyvtár tiszta, azaz nincsenek nem commitolt változtatások. Ha vannak módosítások, azokat commitolni vagy stashelni kell a váltás előtt.
- **Stash használata**: Ha a változtatások nem kívánatosak a jelenlegi ágban, de nem szeretnének commitolni, a git stash parancs használható. Ez ideiglenesen elmenti a változtatásokat, így lehetővé teszi az ágak közötti váltást.
- **Konfliktusok elkerülése**: Ha egy másik ágra történő váltás során olyan fájlok vannak módosítva, amelyek a célágban is módosultak, merge konfliktusok léphetnek fel. Ezeket a konfliktusokat manuálisan kell megoldani a váltás után.
- **Követetlen fájlok**: Az ágak közötti váltás nem érinti az untracked státuszban lévő (nem verziókezelt) fájlokat. Ezek a fájlok megmaradnak a munkakönyvtárban.

Stash

A `git stash` parancs ideiglenesen elmenti a munkakönyvtárban lévő módosításokat anélkül, hogy commitolni kellene őket. Ez hasznos lehet, ha váltani kell egy másik ágra, de a jelenlegi módosításokat még nem szeretnénk commitolni. A `git stash` parancs elmenti a változtatásokat egy "stash" nevű területre, ahonnan később vissza lehet őket állítani. Módosítások stashelése:

```
$ git stash -m "Temporary changes"
```

Ezután biztonságosal lehet váltani a `main` ágra:

```
$ git checkout main
```

A stashelt változtatások megtekintése:

```
$ git stash list
stash@{0}: On feature-branch: Temporary changes
```

Változtatások kivétele a stashből:

```
$ git stash pop stash@{0}
```

Összefésülések

Ágak összefésülése a `git merge` paranccsal lehetséges. Ebben az esetben a fejlesztési előzmények a két ág esetén eltérnek. Mivel a commit ág nem közvetlen őse a célágnak, a Git egy háromágú összefésülést végez a két pillanatkép és a közös ős segítségével.

```
$ git checkout master
Switched to branch 'master'
$ git merge iss53
Merge made by the 'recursive' strategy.
index.html |
 1 +
1 file changed, 1 insertion(+)
```