Praktične osnove Git-a

Borna Skračić, pred.

2024

1 Uvod

U ovom dokumentu pokazat će se osnove rada sa sustavom za čuvanje verzija koda, *git*. Za sva pitanja, slobodno se javite na borna.skracic@algebra.hr. Kako biste mogli koristiti *git* alat na Vašem računalu, potrebno je instalirati *git* s poveznice.

2 Osnove Bash-a

Prije početka rada s alatom *Git Bash* (što je preferirani način rada s *gitom*), potrebno je upoznati se s osnovama korištenja Linux Bash naredbenog retka. Iako postoji opcija korištenja *git-a* kroz grafički alat ili IDE, preproučljivo je inicijalno savladati rad s alatom kroz naredbeni redak. Važno je zapamtiti da se u Linux terminlogiji mape (eng. *folders*) nazivaju direktorijima.

2.1 Anatomija naredbi

Sve Linux naredbe izdaju se kroz naredbeni redak Linux ljuske (eng. *shell*). Svaka naredba može se definirati kroz sljedeći predložak:

```
naredba <podnaredba> -<zastavice> [<argumenti>]
```

Naredba je ime alata koji se koristi, primjerice **ls** (skraćeno od *list*, odnosno izlistaj datoteke). Neki alati, poput *git-a* svoje naredbe dijele u podnaredbe koji specificiraju koju funkcionalnost želimo koristiti. Naredbama se također može zadati opcionalni skup zastavica (eng. *flags*) koje postavljaju dodatne opcije prilikom izvršavanja naredbe (primjerice, *git log –graph* će prikazati listu snimki stanja u obliku grafa). Argumenti intuitivno postavljaju vrijednosti koje naredba koristi prilikom izvršavanja. Primjerice naredba:

```
git remote add <ime udaljene poveznice> <url>
```

dodaje udaljenu poveznicu zadanog imena sa zadanim URL-om.

2.2 Često korištene naredbe

Važno je napomenuti da naredbena ljuska uvijek pokazuje neku putanju, koju u daljnjem tekstu smatramo radni direktorij. Slijedi popis često korištenih naredbi i njihovo objašnjene.

```
pwd
```

Prikazuje apsolutnu putanju radnog direktorija.

```
cd <putanja>
```

Promijeni radni direktorij.

```
ls
```

Izlistaj stavke zadanog direktorija (*ls -la* za prikaz svih skrivenih datoteka).

```
cat <putanja>
```

Prikazuje tekstualni sadržaj datoteke na zadanoj putanji.

```
grep <uzorak>
```

Pronalazi zadani uzorak u ulaznom toku.

Poznavanje ovih naredbi je dovoljno kako bismo se snašli u okruženju Linux ljuske, a samim ime i *git-om*.

3 Git

Prije početka produktivnog timskog rada na zajedničkom kodu, potrebno je proučiti *git* naredbe i njihova pravila. Svaka *git* naredba se izdaje kao

```
git [podnaredba]
```

Neke podnaredbe zahtjevaju argumente ili dodatne zastavice (eng. *flags*). U nastavku teksta opisan je osnovni tijek rada i pripadajuće naredbe.

3.1 Repozitoriji

3.1.1 Postavljanje lokalnog repozitorija

Kako bi lokalna mapa postala *git* repozitorij potrebno je u putanji lokalnog direktorija izdati naredbu:

```
git init
```

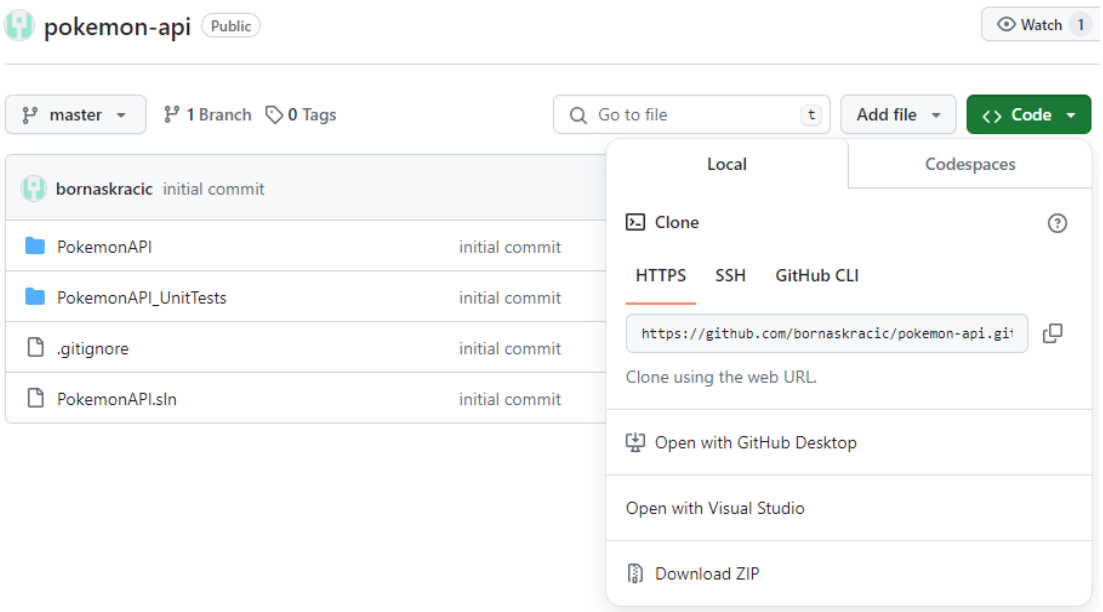
Nakon uspješnog izdavanje naredbe, skriveni **.git** direktoriji s konfiguracijskim stavkama nalazit će se na putanji. **Skriveni .git direktoriji nikad ne modificirate ručno!**

3.1.2 Rad s remote repozitorijem

Ukoliko na servisu za čuvanje verzija koda (primjerice Github) već postoji remote repozitorji, tada je potrebno izdati naredbu:

```
git clone <url>
```

gdje je URL poveznca na remote repozitorij koji se intuitivino može pronaći na stranici repozitorija kako je prikazano na slici.



Po završetku kloniranja, preuzeti direktorij sadržavat će sve datoteke repozitorija, skriveni .git direktoriji i postavljene remote poveznice (što možete lako provjeriti izdavanjem naredbe **git remote -v**)

3.2 Snimke stanja

Kako bi stvorili snimke stanja, potrebno je prvo datoteke dodati u međuspremnik koji nazivamo *staging area*. Te datoteke *git* će svrstati u snimku stanja (eng. *commit*). Ukoliko datoteku želimo dodati u spomenuti međuspremnik, koristimo naredbu:

```
git add <ime_datoteke>
```

Uglavnom ne dodajemo datoteke ručno, već definiramo datoteku naziva **.gitignore** u kojoj navedemo koje datoteke i direktorjie želimo isključiti iz snimke stanja (primjerice *buildove*, konfiguracije, velike datoteke, itd.). Shodno tome, možemo jednostavno zadatni naredbu:

```
git add -A
```

što će dodati sve datoteke koje nisu navedene u **.gitignore** datoteci.

Nakon što su datoteke dodane u *staging area*, možemo vidjeti njihov status korištenjem naredbe:

```
git status
```

Kako bi saznali koje su se promjene nastale u datotekama s obzirom na prijašnji commit, možemo izdati naredbu:

```
git diff
```

Kada smo sigurni da imamo datoteke koje želimo pohraniti u snimku stanja i kojima pratimo promjene (eng. *tracking changes*), izdajemo naredbu:

```
git commit -m '<poruka>'
```

Ta naredba će stvoriti *commit* koji sadrži poruku koja je prosljeđena zastavicom *-m*. Novo stvoreni *commit* nalazi se na lokalnoj inačici repozitorija.

3.3 Sinkronizacija lokalnog i udaljenog repozotorija

3.3.1 Slanje promjena

Ukoliko bismo lokalne promjene htjeli sinkroinizriati s udaljenim (eng. *remote*) repozitorijem (kako bi drugi članovi mogli vidjeti i preuzeti promjene) potrebno je uskladiti lokalnu i udaljenu povijest snimki stanja korištenjem naredbe:

```
git push <ime udaljene poveznice> <ime grane>
```

Prije slanja promjena potrebno je provjeriti konfiguraciju udaljenih poveznica. Listu svih udaljenih poveznica možemo pronaći izdavanjem naredbe:

```
git remote -v
```

Ova naredba ispisat će sva imena udaljenih poveznica i njihove pripadajuće URL-ove. Ukoliko želimo dodati novu *remote* poveznicu, možemo to učiniti na sljedeći način:

```
git remote add <ime> <url>
```

Uobičajno glavnu udaljenu poveznicu nazivamo *origin*, što će *git clone* naredba napraviti automatski za preuzeti repozitorij. URL udaljenog repozitorija možemo saznati na stranicama servisa za čuvanje verzija koda, kako je prikazano na prijašnjoj slici.

3.3.2 Preuzimanje promjena

Kako bismo preuzeli promjene koje se nalaze na udaljenom repozitoriju, potrebno je izdati naredbu:

```
git pull <ime udaljene poveznice> <grana>
```

Podnaredba *pull* će pregledati promjene na udaljenom repozitoriju i integrirati ih lokalno. Postoji inačica podnaredbe, zvana *fetch*:

```
git fetch <ime udaljene poveznice> <grana>
```

koja će provjeriti promjene, **ali ih neće integrirati u loklani repozitorji**.

3.4 Grane

3.4.1 Stvaranje grana

Dobra praksa u radu s *git-om* je rad korištenjem grana (eng. *branch*). Grane nam pomažu u vođenju neovisnih povijesti snimka stanja koje se kasnije mogu integrirati u glavni tijek, odnosno glavnu granu repozitorija (najčešće *master* ili *main* grana). Preporučljivo je u timskom radu, pirilikom razvitka nove značajke (eng. *feature*) da razvojni inžinjer radi na zasebnoj grani i po završetku implementacije, integrira promjene u glavnu granu.
Kako biste vidjeli popis svih grana u repozitorju, možemo izdati naredbu:

```
git branch
```

u čijem će sadržaju jasno biti naznačena trenutna grana. Za stvaranje nove grane potrebno je izdati istu naredbu, uz argument koji predstavlja ime nove grane:

```
git branch <ime grane>
```

Nakon toga, možemo se jednostavno "prebaciti" na novu granu korištenjem naredbe:

```
git checkout <ime grane>
```

3.4.2 Spajanje grana

Kada želimo integrirati promjene iz nove grane u glavnu granu, potrebno je prebaciti se na glavnu granu izdavanjem *git checkout* naredbe. Nastavno, kako bismo spojili (eng. *merge*) sve promjene u glavnu granu, izdajemo naredbu:

```
git merge <ime grane>
```

U ovom koraku često dolazi do konflikata u promjenama koje se nalaze u istim datotekama. Kako biste razrješili konflikte potrebno je pogledati u kojim datotekama se nalaze konflikti. Git će jasno označiti na kojim linijama je došlo do konflikta upotrebom znakova kao u sljedećem primjeru:

```
<<<<<<< HEAD
this is some content to mess with
content to append
=======
totally different content to merge later
>>>>>>> nova-grana
```

Nastale oznake nalaze se u datoteci *primjer.txt* gdje je *git* pronašao konfliktne promjene na istim linijama te je potrebno odlučiti koje od tih promjena (stanja) želimo zadržati.

```
<<<<<<< HEAD
```

označava retke koje se nalaze u snimci stanja na glavnoj grani (u koju se spajaju promjene, u ovom slučaju *master*), dok

```
>>>>>>> nova-grana
```

označava promjene iz grane koju želimo spojiti.

```
=======
```

označava gdje prestaje stanje na glavnoj grani i počinje stanje na novoj grani. Prilikom rješavanja konflikata, odaberemo koju verziju linija želimo, a ostalo izbrišemo. Većina pametnih IDE-a ili sličnih grafičkih alata će nam ponuditi intuitivno sučelje za rješavanje konflikata.
Pri završetku potrebno je snimiti sanje, odnosno napraviti *commit*.

Daljnji rad prati postupak opisan poglavljima 3.2 i 3.3.

3.5 Ostale korisne naredbe

3.5.1 Ispis svih snimki stanja

Kako bismo saznali povijest snimki stanja u repozitoriju, možemo izdati naredbu:

```
git log
```

Inačica naredbe:

```
git log --graph --decorate --all
```

prikazat će povijest stanja u obliku grafa s tekstualnim dekoracijama.

3.5.2 Povratak na prijašnji commit

Najlakši način za povratak na prijašnje stanje jest izdavanje naredbe:

```
git reset --hard <commit id>
```

Identifikator *commit-a* moguće je pronaći u sadržaju ispisa naredbe *git log*. Osim identifikatora moguće je zadati referencu na *commit* u odnosu na trenutni *HEAD* (posljednji *commit* na trenutnoj grani), primjerice:

```
git reset --hard HEAD~2
```

postavlja trenutno stanje repozitorija na dva *commit*-a prije trenutnog.