Git - distribuirani sistem za kontrolu verzije

Preporuka za čitanje o Git-u - besplatna knjiga Pro Git Book

Motivacija

- Sistematski rešava jedan od klasičnih problema “juče je radilo, danas ne”
- U mnogome olakšava rad u grupi
- Gotovo svaka firma, istraživački centar, čak i akademske zajednice ga aktivno koriste (na neki način moglo bi se reći da je sada poznavanje Git-a “opšte znanje” u IT sektoru)

Uvod

Git je trenutno najpopularniji i najrasprostranjeniji sistem za kontrolu verzija. Namenjen je praćenju **promena** izvornog koda tokom procesa razvoja softvera. Razvijen je sa idejom podrške koordinaciji tima programera (konkretno je Linus Torvalds napravio kako bi olakšao razvoj Linux kernela), i članovi tima mogu da obavljaju razvoj, testiranje, otklanjaju greške i dodaju nove delove znajući da u bilo kom trenutku može biti rekonstruisana bilo koja verzija koda u proizvoljnom trenutku.

Osobe koje učestvuju u razvoju mogu na osnovu uvida u projektnu istoriju da saznaju:

- Koje su promene unete?
- Ko ih je uneo?
- Kada su promene nastale?
- Zašto su promene bile neophodne?

Načini korišćenja Git-a

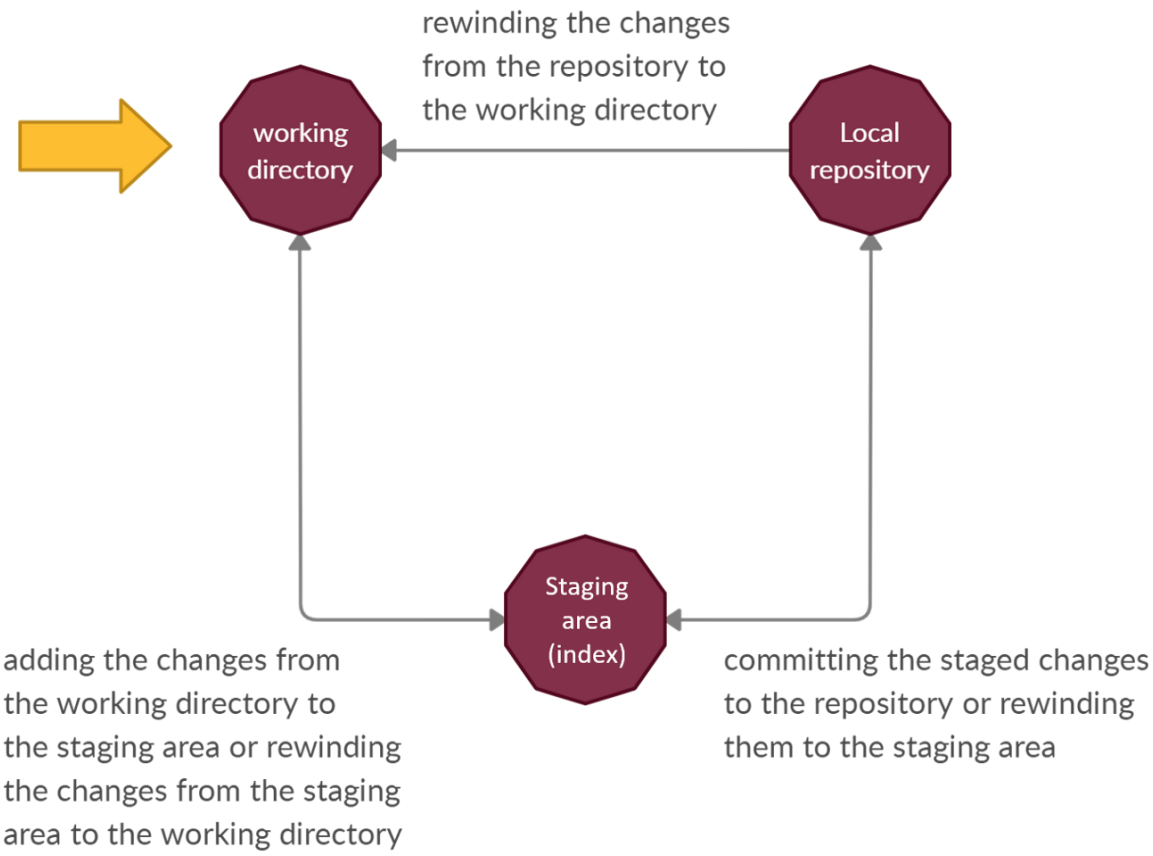
- Pomoću komandne linije - na vežbama ćemo koristiti Git Bash
- Pomoću grafičkog interfejsa - Osnovni je GitGui - olakšava pregled, a neki kompleksniji su Git Kraken ili Tortoise Git

Preporuka je da se **prvo savlada korišćenje iz terminala**, kako bi imali neki osećaj šta se dešava kada se koriste grafički alati.

Git je **distribuirani sistem** za kontrolu verzija - to znači da **ne postoji jedan glavni server** ili sa kojim mora da se komunicira tokom korišćenja. Svaka instanca repozitorijuma je nezavisna i sadrži punu istoriju.
Prednost je ta što je moguće raditi u potpunosti bez mreže ili bilo kakvim kontaktom sa ostalim članovima tima, a samo po potrebi ostvarivati vezu i preuzimati ili postavljati izmene na server ili druge instance.
Praksa je, ako više ljudi radi na projektu, da postoji jedna instanca repozitorijuma kojoj svi imaju pristup (najčešće se naziva **origin**) i GitHub je trenutno najpopularnija platforma.

Git koncepti i proces rada

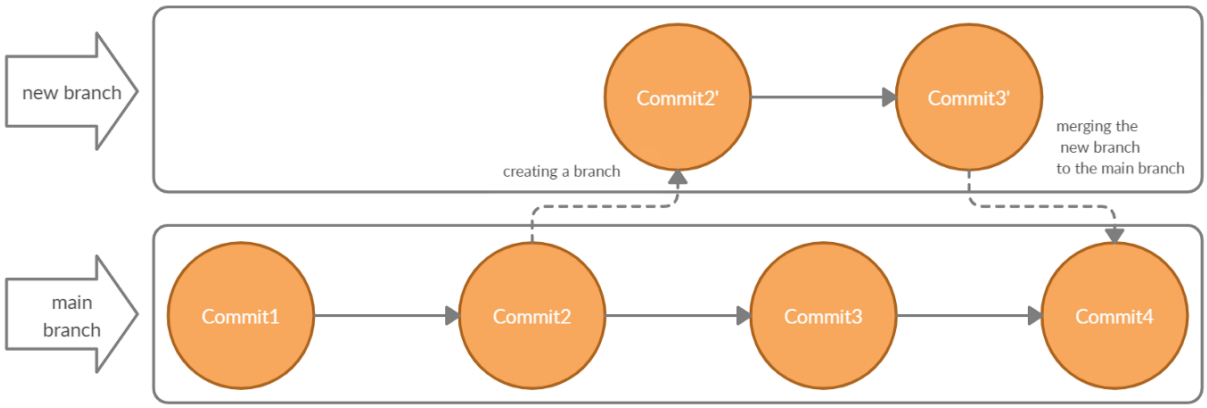
Kao što je prethodno rečeno, Git je sistem koji prati izmene tokom razvoja projekta. Dokumenta, nezavisno od njihovog tipa, mogu da se nalaze u **tri sekcije** kada je u direktorijumu inicijalizovan repozitorijum.



Na slici iznad su prikazana te tri sekcije.
Sve što se nalazi u **radnom okruženju** je nadgledano od strane Git-a. Svaka izmena (dodavanje, uklanjanje ili menjanje nekog dokumenta) će automatski biti detektovana.

Iako će Git pratiti sve izmene, na korisniku je da te izmene prvo prebaci u **pripremnu zonu,** a nakon toga i konačno postavi (**commit-**uje) u lokalni repozitorijum.

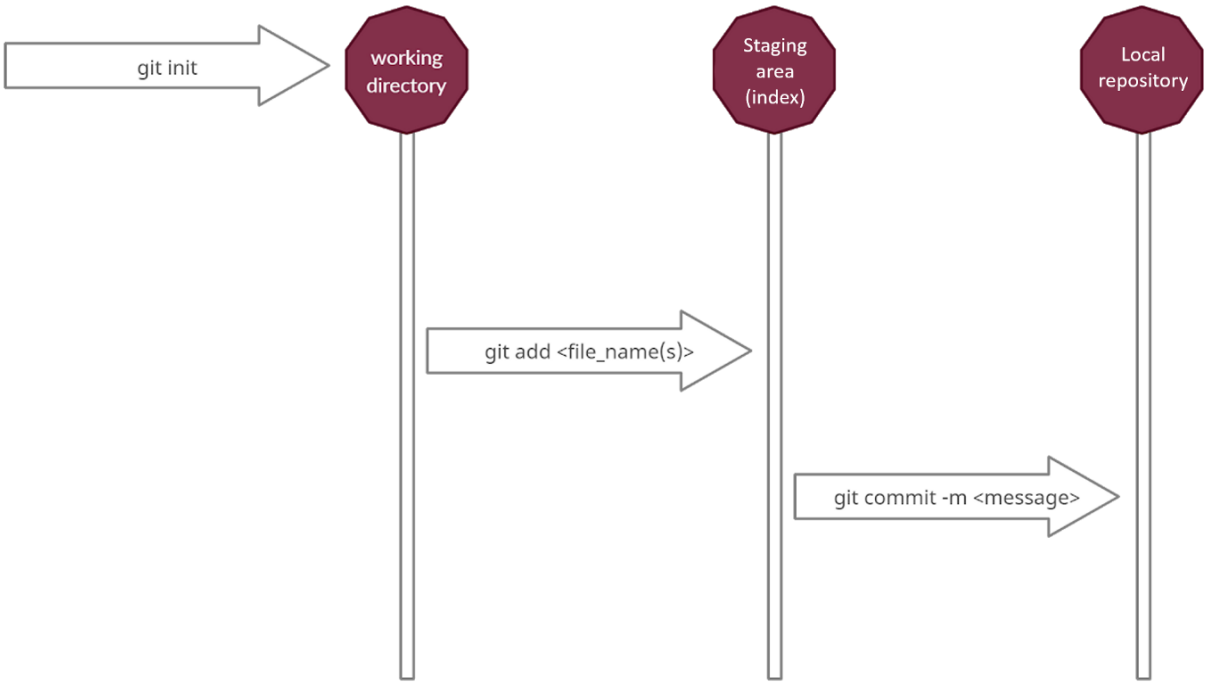
Istorija jednog repozitorijuma se ogleda u nizu **commit**-ova, ilustrovano na slici ispod:



Još jedan bitan koncept u Git-u je mogućnost grananja (kreiranja **branch-**eva).

Ovo omogućava paralelni rad na više funkcionalnosti u isto vreme. Jednom kada se funkcionalnost u potpunosti ispuni, najčešće se izmene sa određene grane spoje (**merge)** u glavnu granu.

Primer inicijalizacije i commit-ovanja



Osnovne Git komande

Tok rada se obično odvija u tri faze koje se sukcesivno smenjuju:

1. Korisnik modifikuje sadržaj datoteka u radnom direktorijumu
2. Korisnik selektivno označava one datoteke koje trebaju biti postavljene u repozitorijum, čime se one prebacuju u pripremnu zonu (**staging)**
3. Obavlja se operacija postavljanja (**commit)**, čime se preuzimaju datoteke iz pripremne zone i permanentno smeštaju u Git repozitorijum

git init

Komanda ***git init*** inicijalizuje novi repozitorijum u tekućem direktorijumu i počinje sa njegovim praćenjem. Na ovaj nacin se dodaje skriveni direktorijum (.git) u kojem je smeštena interna baza podataka neophodna za praćenje verzije

Primer: `git init` -> pozvano u direktorijumu koji želimo da postane git repozitorijum

git clone

Komanda ***git clone*** kreira lokalnu kopiju postojećeg repozitorijuma sa servera. Klonirani repozitorijum sadrži sve izvorne datoteke, zajedno sa istorijom promena.

Primer: `git clone https://github.com/me/repo.git`

git status

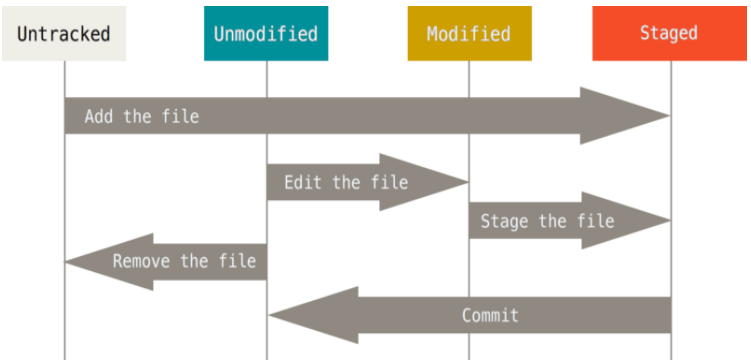
Svaka datoteka u radnom direktorijumu može se nalaziti u jednom od dva stanja:

- Nepraćena datoteka ***(untracked)***
- Praćena datoteka ***(tracked)***

Praćena datoteka je ona o čijim promenama Git vodi računa. Ona može biti u nekom od sledećih stanja:

- Neizmenjena (***unmodified)*** - fajl čiji sadržaj nije menjan od poslednjeg postavljanja
- Izmenjena (***modified)*** - datoteka čiji sadržaj jeste promenjen
- Pripremljena za postavljanje ***(staged)*** - prilikom sledećeg postavljanja, biće postavljene samo one datoteke koje se nalaze u ovom stanju, tj. koje se nalaze u pripremnoj zoni

Napomena: moguće je dodavati i pojedinačne delove datoteke, ne mora cela



Komanda ***git status*** služi za proveru stanja datoteka u radnom direktorijumu

git add

Komanda ***git add*** označava izmenjene datoteke za postavljanje, čime se prebacuju u pripremnu zonu.

```
git add main.c
```

git commit

Komanda ***git commit*** snima promene iz pripremne zone u repozitorijum, čime se kompletira kreiranje nove verzije. Uz svaki ***commit*** je potrebno napisati i komentar koji opisuje svrhu postavljenih izmena.

```
git commit -m “Popravlja grešku u računu.”
```

git branch

Komanda bez parametara izlistava trenutno postojeće grane i zvezdicom označava na kojoj se trenutno nalazi glava (HEAD) repozitorijuma.

```
git branch nova_grana
```

Kreira novu granu razvoja koda pod zadatim imenom.

git checkout

Prelazi na željenu granu ili određeni commit.

```
git checkout nova_grana
```

git merge

Spaja promene neke druge grane u granu na kojoj se trenutno nalazimo.

git pull

Komanda ***git pull*** ažurira sadržaj lokalnog repozitorijuma sadržajem sa udaljenog servera.

git push

Komanda ***git push*** ažurira repozitorijum na udaljenom serveru promenama napravljenim u lokalu

Na sledećem linku se nalazi Git Cheat Sheet sa kratkim objašnjenjima komandi.