Napredni GIT koncepti za developere

Za developera učenje rada sa Gitom se nekako podrazumeva. Rad sa lokalnim i udaljenim repozitorijumima, unos promena i upravljanje granama: ovo su neke osnove Gita koje bi trebalo da znate. Ali pored ovih osnova, Git pruža i neke naprednije komande koje će vam kao developeru definitivno olakšati život. Oni mogu poboljšati vašu produktivnost i rešiti neke ne tako neuobičajene probleme sa kojima se susrećete kada sarađujete na projektu sa drugim developerima.

Ako ste želeli da postanete profesionalac u radu sa Gitom, ovo su sigurno komande koje treba da znate.

U današnjem tekstu ćemo vas provesti kroz nekoliko naprednih Git koncepata. Od `git bisect` do `git rebase`. Oni vam mogu uštedeti mnogo vremena koje provodite pretražujući Stack Overflow, kada dođe do problema sa kontrolom verzija.

Brzo pronađite greške pomoću `git bisect`

Pretpostavimo da repozitorijum na kojem radite ima nekoliko neuspešnih testova zbog greške. Niko još nije uspeo da odredi uzrok ili commit koji ga je pokrenuo. Pronalaženje lošeg commit-a je posebno teško jer su i stotine drugih već napravljene u spremištu nakon poslednjeg radnog commit-ovanja.

Git bisect vam omogućava da brzo pronađete commit koje je stvorio grešku u ovakvim situacijama. Kada obezbedite poznato „dobar“ i „loš“ commit, git bisect koristi binarnu pretragu da bi u najkraćem mogućem roku došao do prvog problematičnog commit-a.

Pokretanje operacije presecanja počinje sa komandom:

```
git bisect start
```

Možete identifikovati referencu dobrog commit-a, gde je sve funkcionisalo kako se očekivalo, i lošeg commit-a, gde test ne uspeva, tako što ćete pregledati istoriju commit-ovanja koristeći `''git log''`. Morate da obezbedite ova dva commit-a da biste ih podelili koristeći sledeće dve komande.

```
git bisect good <good-commit-ref>
git bisect bad <bad-commit-ref>
```

Čim pokrenete `''git bisect bad''`, on proverava commit-ovanje na sredini između dobrih i loših commit-a, što vam omogućava da ga pregledate. Ako testovi i dalje ne uspevaju u proverenom commit-u, možete da obavestite bisect tako što ćete pokrenuti `''git bisect bad''`. S druge strane, ako je to dobar commit, pokrenite `''git bisect good''`.

Nakon ovog koraka, bisect ponovo proverava srednji commit između ažuriranih dobrih i loših commit-a tako da možete da pregledate kod i pokrenete testove da utvrdite da li je dobar ili loš. Nastavlja ovaj proces sve dok ne dođe do prvog lošeg commit-a.

Možete da zaustavite operaciju presecanja i pređete na konačno urezivanje koristeći sledeću komandu

```
git bisect reset
```

Izaberite i primenite individualne commit-e pomoću `git cherry-pick`

Iako se to ne dešava često, ponekad naiđemo na scenarije u kojima želimo da izaberemo određeni commit iz druge grane u repozitorijumu i primenimo ga na onu na kojoj radimo.

Na primer, možda ćete želeti da izaberete promene koje ste greškom uneli u drugu granu i primenite ih na željenu granu. Ili biste možda želeli da primenite rešenje koje je neko drugi pružio na granu na kojoj radite bez pisanja koda od nule. Ako koristite `''git merge''` u ovakvim situacijama, on može primeniti promene koje nisu relevantne za vašu trenutnu granu.
Stoga, u takvim slučajevima, `git cherry-pick` može biti komanda koju treba da koristite.
Kada izvršavate ovu komandu, prvo bi trebalo da preuzmete referencu na commit koji želite da izaberete korišćenjem logova. Zatim pogledajte granu na kojoj želite da primenite izabrani commit.

```
git checkout feature-1
```

Sada možete da primenite commit pomoću:

```
git cherry-pick <commit-ref>
```

Ako želite da samo postavite izmene u ovu granu bez commit-ovanja, trebalo bi da koristite:

```
git cherry-pick <commit-ref> --no-commit
```

Važno je napomenuti da korišćenje ove komande možda nije najbolji način delovanja u nekim slučajevima. U zavisnosti od toga kako koristite `git cherry-pick`, on može da kreira duplikate commit-a u vašoj grani. Commit-ovanje koje primenite takođe može da uvede nove bagove i greške u vašu granu. Dakle, ako rešite da koristite `cherry-pick` commit-e u vašem Git-u, uverite se da ste pre toga isprobali sve druge opcije.

Promenite bazu grane pomoću `git rebase`

Obično koristimo `''git merge''` komandu da kombinujemo izmene napravljene u dve grane repozitorijuma. Ako pogledamo istoriju commit-ovanja repozitorijuma nakon operacije spajanja, biće zbunjujuće pratiti preplitanja commit-a iz dve grane.

Šta ako želimo da olakšamo praćenje istorije commit-ovanja našeg projekta tako što ćemo istoriju grane spajanja linearno uskladiti sa drugom? U tom slučaju možemo da koristimo ''git rebase''.
U nekoliko reči, odgovornost komande rebase je promena baznog urezivanja grane. Da vidimo na nekom konkretnom primeru šta to zapravo znači.

Recimo da imate dve grane u svom spremištu: main i feature. Nakon kreiranja feature grane, vaš tim je dodao commit-e u obe ove grane. Sada želite da spojite granu feature sa main granom pre objavljivanja. U ovom trenutku, možete pokrenuti ''git rebase'' na grani feature pre nego što je spojite pomoću ''git merge''. On menja osnovu grane funkcije do poslednjeg commit-ovanja u vašoj main grani kao što je prikazano na slici ispod.

Komanda rebase daje vašoj feature grani linearno kompatibilnu istoriju sa main granom, za razliku od ranije. Sada, ako pokrenete ''git merge'', istorija commit-ovanja main grane će biti manje zbunjujuća i lakša za praćenje.
Korišćenje ove komande je prilično jednostavno. Prvo pogledajte granu koju želite da ponovo bazirate. Onda pokrenite ovo..

```
git rebase <base-branch-name>
```

Međutim, ''git rebase'' je komanda koju treba da koristite sa oprezom jer ona ponovo piše (gazi) istoriju grane. Tokom procesa možete izgubiti korisne informacije sačuvane u istoriji grane. Takođe može imati negativan uticaj ako radite u grani sa mnogo saradnika. Dakle, najbolje je ograničiti upotrebu na lične grane osim ako niste 100% na istoj stranici sa svim članovima vašeg tima.

Lako izaberite fajlove za commit-ovanje pomoću `git add -p`

''git add'' komanda je jedan od osnovnih koraka u toku rada sa Git-om. On postavlja fajlove u direktorijum projekta koje treba commit-ovati. Koristimo različite verzije ove komande u zavisnosti od konkretne prilike.

''git add <file-path>'' za postavljanje fajlova na određenoj putanji

''git add .'' za postavljanje izmenjenih i novih fajlova

Ali ako želite da pregledate promene u svakoj datoteci deo po deo (kao zakrpe) pre nego što odlučite da li da ih postavite ili ne, možete da koristite komandu ''git add -p''. Prikazuje izmene napravljene u fajlu u vidu pojedinačnih delova i traži od vas da izaberete da li da postavite svaku od njih ili ne.

Evo izbora koje Git pruža za svaki deo. Trebalo bi da otkucate odgovarajuće slovo kada se od vas zatraži da potvrdite svoju odluku.

y: postavi ovaj deo

n: ne postavljaj ovaj deo

q: quit; ne postavljaj ovaj deo ili bilo koji od preostalih delova

a: postavi ovaj deo i sve kasnije delove u fajlu

d: ne postavljaj ovaj deo ili bilo koji od kasnijih delova u fajlu

e: ručno editujte trenutni deo

Promenite svoju poslednju commit poruku pomoću `git commit –amend`

Sigurno ste se zatekli u situaciji da želite da promenite svoj poslednji commit jer ste pogrešili ili ste zaboravili da dodate nešto važno? ''git commit --amend'' vam omogućava da uradite upravo to.

Sa ovom komandom, možete promeniti svoju poslednju commit poruku ili kombinovati trenutno izvedene promene sa prethodnim commit-ovanjem bez kreiranja potpuno novog commit-a. Ovaj korak ne menja samo prethodno commit-ovanje; on zamenjuje commit potpuno novim.

Hajde da vidimo kako da koristimo ovu naredbu za izmenu commit-a.

Ako želite da promenite poruku:

```
git commit --amend -m <commit message>
```

Ako želite da kombinujete postepeno izmene, prvo dodajte izmenjene datoteke koristeći `''git add''`.
Zatim izmenite poslednji commit na sledeći način:

```
git commit --amend --no-edit
```

Ako ste već push-ovali poslednji commit u udaljei repozitorijum, morali biste da koristite komandu `''git push --force-vith-lease''` za push-ovanje modifikovanog commit-a.
Opet, pošto ova operacija izmene menja istoriju commit-ovanja, trebalo bi da pazite kada i gde je koristite.

Redovno ažurirajte svoju lokalnu Git instalaciju

Ako kao developer radite u Git-u važno je da redovno ažurirate svoju lokalnu Git instalaciju. Tim pre što je otkriveno nekoliko sigurnosnih propusta u ovom popularnom softveru.

Najgori od dva propusta (CVE-2022-24765) potencijalno omogućava napadaču da izvrši određene komande. Najviše su u riziku developeri koji koriste Git za Windows ili Git na multi-user računarima. Kod multi-user računara, zlonamerni napadač može da kreira `.git` direktorijum na deljenoj lokaciji iznad žrtvinog tekućeg radnog direktorijuma. Na primer, u Windows-u, napadač može da kreira `C:\.git\config`, što bi rezultiralo time da sve git invokacije koje se dese van repozitorijuma, pročitaju njegove konfiguracione vrednosti.
Imajući u vidu da neke konfiguracione promenljive (kao recimo `core.fsmonitor`) iniciraju izvršavanje arbitrarnih komandi, to može da dovede do toga da one budu izvršene i tokom rada na deljenom računaru.

Developerima se savetuje da ažuriraju svoje sisteme na Git v2.35.2 da bi se zaštitili od potencijalnih napada. Developeri koji koriste Git na Linux-u ili macOS, su takođe pogođeni propustom pod nazivom CVE-2022-24765, doduše u manjoj meri. U svakom slučaju ažuriranje je svakako neophodno.

Druga ranjivost (CVE-2022-24767) je ograničena samo na Git za Windows uninstaller. Kao i kod prethodne ranjivosti, određeni nivo pristupa je pretpostavka za realizaciju potencijalnog napada.

Napadi se zasnivaju na postavljanju malicioznog `.dll` fajla na ciljanom sistemu.
Imajući u vidu da ova ranjivost predstavlja određeni rizik, korisnici se savetuju da ažuriraju svoju lokalnu Git instalaciju – Git za Windows v2.35.2.