

Sistemi za kontrolu verzija

Alati za timski rad - SVN, Git

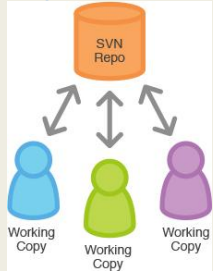
Sistemi za kontrolu verzija

- Koriste se kada je potrebno pratiti verzije projekta i kada više programera radi na projektu
- Postoji repozitorijum u kom se nalaze datoteke i fascikle koje čine projekat
- Svaka izmena koja se sačuva na repozitorijumu (zvana `commit`) pored samih izmena čuva i informacije o tome ko je napravio izmenu, kad je izmena sačuvana i šta je tačno izmena
- Dostupan je istorijat izmena projekta na repozitorijumu, te je moguće pristupiti ranijim verzijama projekta i svim relevantnim podacima
- Repozitorijumi nude mogućnost formiranja više grana projekata, tako da izmene projekta na jednoj grani ne utiču na ostale grane, što pruža podršku eksperimentisanju i nezavisnom razvoju delova sistema

Sistemi za kontrolu verzija

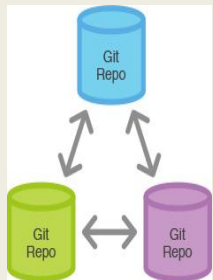
- Dva popularna sistema:

- SVN



- centralizovan sistem za kontrolu verzija, glavni repozitorijum se nalazi na centralnom serveru, dok ostali čvorovi imaju samo preuzetu verziju projekta
- commit čuva izmene u odnosu na prethodni commit
- alati: TortoiseSVN, Subclipse,...

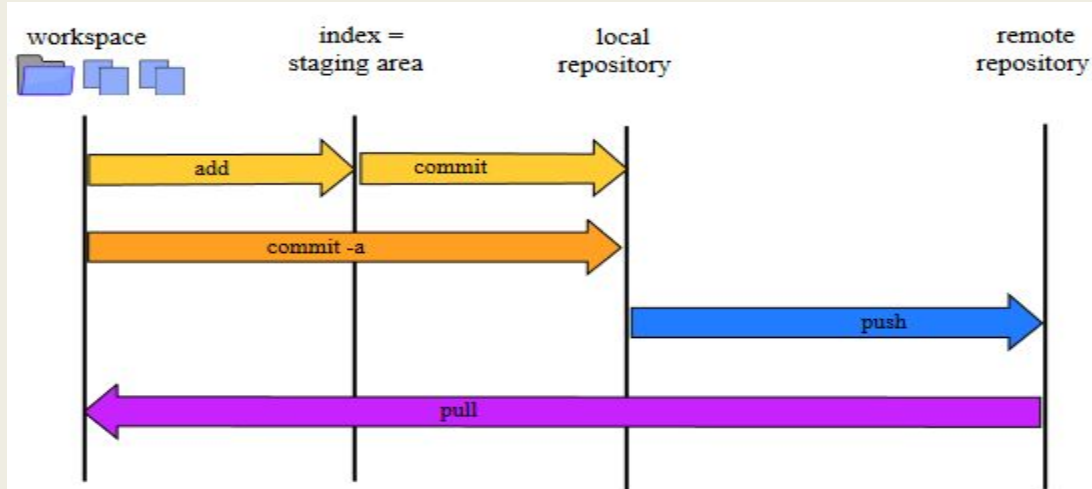
- Git



- distribuiran sistem za kontrolu verzija, gde svako poseduje ceo repozitorijum kod sebe
- svaki commit čuva cele izmenjene datoteke (snapshot), a ne samo izmene
- alati: Git Bash, SourceTree,...

Git

- Iako svako u timu sadrži ceo, suštinski isti repozitorijum, jedan repozitorijum se može smatrati posebnim (`origin`), a to je onaj koji se nalazi na udaljenom serveru (najčešće GitHub, GitLab, BitBucket, interni git server firme,...)
- Čuvanje izmena se vrši u dve faze, postavljanje na *staging area*, gde se commit može pripremiti i sam commit
- Korisno za logičko odvajanje izmena u više commit-a
- Jedan commit treba biti moguće opisati kroz rečenicu ili dve
- Težiti ka davanju prefiksa commit-u (Add, Fix, itd.) kako bi se ukazalo na tip izmene

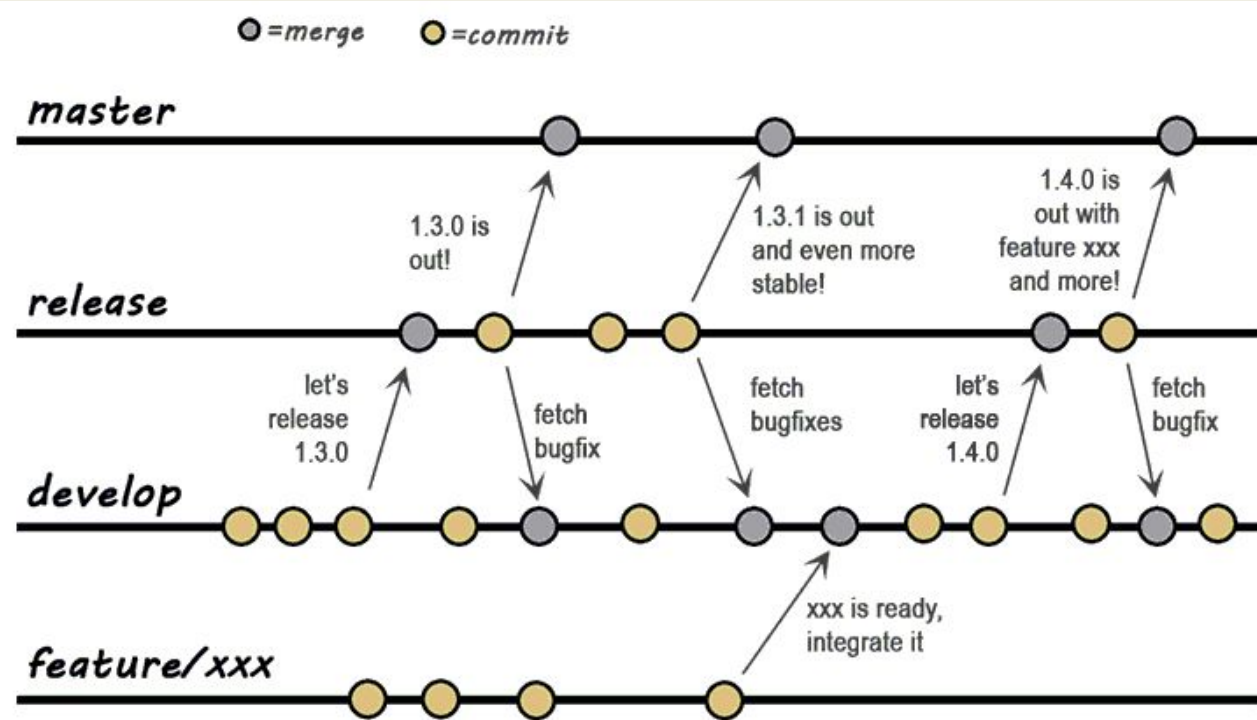


Git: Operacije

- Najčešće operacije (`git <command>`):
 - `init <project name>` - pravi nov lokalni repozitorijum u tekućem direktorijumu
 - `clone <repository url>` - preuzimanje celog repozitorijuma
 - `add <file name>` - stavljanje promene datoteke na staging area
 - `rm <file name>` - brisanje datoteke iz sistema i postavljanje na staging area
 - `commit -m "<message>"` - čuvanje izmena koje su postavljene na staging area u lokalni repozitorijum
 - `status` - prikaz stanja datoteka koje su ažurirane, a nisu `commit`-ovane
 - `diff <file name>` - prikazivanje izmene datoteke u odnosu na verziju koja je poslednje preuzeta sa repozitorijuma
 - `checkout <file name>` - odbacivanje promena nad datotekom
 - `revert <commit>` - odbacuje izmene napravljene u datom `commit-u`

Git: Grane

- Omogućavaju izolovan rad na različitim komponentama sistema, i pruža dobru podršku testiranja ideja
- Razvijamo komponentu i kad smo zaokružili celinu spajamo (`merge`) kod komponente sa glavnom granom
- Preporuka je da master grana ima samo `merge`, a ne `commit` čvorove
- HEAD pokazivač pokazuje na kom čvoru smo



Git: Operacije sa granama

- Najčešće operacije (`git <command>`):
 - `branch` - lista sve grane koje postoje u repozitorijumu (karakter `*` stoji pored grane na kojoj se trenutno nalazimo (HEAD))
 - `branch <branch name>` - stvara novu granu sa datim imenom
 - `branch -d <branch name>` - briše granu sa datim imenom
 - `checkout <branch name>` - pomera HEAD pokazivač na datu granu i menja sadržaj fajl sistema tako da oslika stanje repozitorijuma u datom čvoru
 - `merge <branch>` - kombinuje istorijat (`commit`-ove) navedene grane u trenutnu granu (prilikom ove operacije mogu nastati konflikti)

Git: Scenario upotreba grana

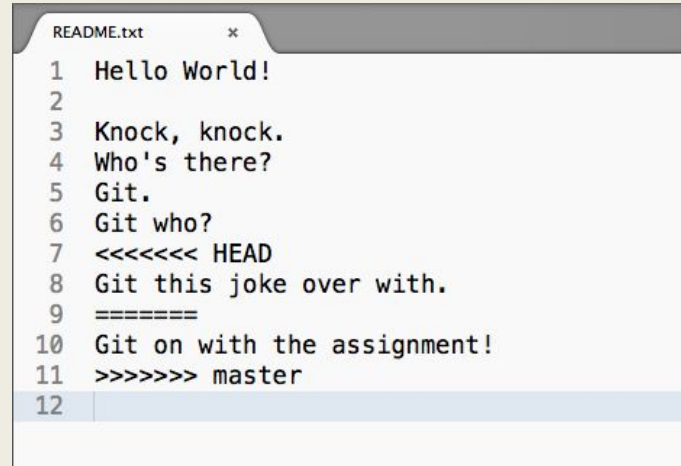
- Razvijate web sajt za evidenciju biljaka koji je postavljen na 'development' grani
- Formirate granu 'tree' (`branch tree`) gde ćete razviti komponente za unos, izmenu, pregled i brisanje drveća (create, read, update, delete - CRUD)
- Radite na toj grani i u jednom trenutku vam jave da postoji problem sa web sajtom
- Čuvate ili odbacujete izmene koje ste napravili i prebacujete se na 'development' granu (`checkout development`), i kreirate novu granu 'hotfix' (`branch hotfix`)
- Kad ste uradili ispravku i namestili sve da radi vraćate se na 'development' granu (`checkout development`) i spajate izmene sa 'hotfix' grane u 'development' granu (`merge hotfix`)
- Obrišete 'hotfix' granu (`branch -d hotfix`) i vratite se na granu 'tree' (`checkout tree`) i nastavljate razvoj komponente
- Napomena: Izmene koje je 'hotfix' doneo na 'development' se ne nalaze u 'tree'

Git: Merge

- Merge je trivijalan u slučaju 'development' i 'hotfix' grane, gde se samo pripaja sadržaj jedne grane drugoj
- Problem nastaje kada se desi divergencija grana, kao na primer kada bi hteli da spojimo 'tree' i 'development' granu
- Dešava se spajanje preko tri čvora, gde se posmatraju poslednji čvorovi grana koje se spajaju i čvor u kom su se grane razišle, i rezultat tog spajanja je nov čvor
- Ukoliko se desi da se u toku rada obe grane izmenile iste delove istih datoteka dolazi do konflikta

Konflikti

- Konflikti nisu specifični za git i svi sistemi za kontrolu verzija imaju svoj način razrešavanja konflikta
- Izvršavanjem `git status` dobijamo informacije o datotekama koje su u konfliktu
- Kad je datoteka u konfliktu izgeneriše se tekst koji obeležava koje linije koda pripadaju grani na kojoj smo sad, a koje grani sa koje `merge`-ujemo
- Na slici se pojavio konflikt kad su dve strane u isto vreme izmenile liniju ispod “Git who?” i sistem je izmenio README.txt datoteku tako da uključi tekst iz obe verzije
- Nakon što se izabere tekst koji želi da se sačuva potrebno je dodati datoteku na staging area putem `git add`, što razrešava konflikt



```
README.txt
1 Hello World!
2
3 Knock, knock.
4 Who's there?
5 Git.
6 Git who?
7 <<<<<<< HEAD
8 Git this joke over with.
9 =====
10 Git on with the assignment!
11 >>>>>> master
12
```

Git: Operacije sa udaljenim repozitorijumom

- Najčešće operacije (`git <command>`):
 - `remote` - prikaz liste repozitorijuma koji su vezani za dati projekat
 - `remote add <alias> <url>` - dodavanje veze sa repozitorijumom pod aliasom
 - `remote rm <alias>` - uklanjanje veze između repozitorijuma i projekta
 - `branch -a` - sve grane vezane za projekat, i one sa udaljenih repozitorijuma
 - `fetch <repository alias/url>` - preuzima ceo istorijat repozitorijuma i stvara/ažurira remote grane
 - `pull <repository alias/url>` - izvršava `git fetch <repository alias>` i odmah zatim `git merge` komandu, tako da komanda `git pull` preuzima sadržaj repozitorijuma i spaja podatke sa njega u lokalnu granu na kojoj se nalazimo
 - `push <repository alias/url> <branch>` - šalje izmene sa grane na repozitorijum
- <https://git-scm.com/book/en/v2> (prva 3 poglavlja za osnove)

Konflikti - nastavak

- Dva korisnika inicijalno povuku istu verziju datoteke (`git pull`), nakon čega oba korisnika izmene datoteku i sačuvaju izmene u lokalni repozitorijum
- Oba korisnika izvrše `git push` naredbu da pošalju izmene na udaljeni repozitorijum:
 - prvi `push` prolazi bez problema i čuva se izmena na repozitorijumu;
 - drugi `push` zahteva od korisnika da preuzme najnoviju verziju udaljenog repozitorijuma pre nego što može da izvrši `push`;
 - ukoliko u ovom trenutku izvrši operaciju `pull`, drugi korisnik dobija konflikt
 - konflikt se razrešava na isti način kao kod `merge`-a lokalnih grana
 - preporuka je raditi `git fetch` i onda `git merge` kako bi imali veću kontrolu nad procesom i veću svest o tome šta se zapravo izmenilo

Angular biblioteke

Internacionalizacija, UI Bootstrap

Internacionalizacija

- Internacionalizacija (I18n) predstavlja proces razvoja aplikacije tako da sadrži elemente koji mogu prikazivati sadržaj prilagođen lokalitetu
- Lokalizacija (L10n) predstavlja proces adaptiranja delova aplikacije za neki konkretan lokalitet (način na koji se određeni tekst prikazuje, način na koji ispisujemo datum i vreme, zatim valuta u kojoj je cena proizvoda prikazana i sl.)
- Angularovi filteri za valutu, datum i brojeve prilagođavaju ispis lokalitetu
- Potrebno je preuzeti i uključiti u projekat skriptu za traženi lokalitet
- <https://github.com/angular/bower-angular-i18n>
- Nakon toga, sve što je potrebno je uneti zavisnost za `ngLocale` i filteri će biti prilagođeni navedenom lokalitetu
- Nema načina da se dinamički menja lokalitet bez osvežavanja stranice
- Ne postoji servis/način da se lokalizuje tekstualni sadržaj stranice

Dinamička promena lokaliteta

- Biblioteka `angular-dynamic-locale` omogućava dinamičku promenu lokaliteta i instalira se putem komande `bower install angular-dynamic-locale`
- Potrebno je uključiti skripte u `index.html` i označiti zavisnost za `tmh.dynamicLocale` modul
- U kontroleru se označava zavisnost za `tmhDynamicLocale` i poziva se nad servisom metoda `tmhDynamicLocale.set(<ime lokaliteta>)`
- Modul podrazumevano traži skripte za lokalitete po šablonu `angular/i18n/angular-locale_{{locale}}.js`, ali je moguće konfigurisati `tmhDynamicLocaleProvider` putem `localeLocationPattern(<šablon putanje>)`

Prevođenje teksta

- Tekstualan sadržaj web stranice se može podeliti u dve grupe:
 - Statički tekst - labele, naslovi, tekst koji je deo korisničkog interfejsa
 - Dinamički tekst - podaci koji se dobavljaju sa servera ili udaljenih servisa
- `angular-translate` je modul koji podržava lokalizaciju teksta, asinhrono učitavanje podataka za i18n, dinamičku promenu lokaliteta, zatim filtere, servise i direktive za podršku lokalizacije sadržaja (`bower install angular-translate`)
- Uključiti modul u projekat i zatim sav statički tekst iz HTML-a zameniti ključevima koji će za vreme izvršavanja aplikacije da budu zamenjeni lokalizovanim tekstom
- Definišemo objekat za određen lokalitet koji sadrži parove ključ - prevod i prosledimo ga `$translateProvider.translations(<ime lokaliteta>, <objekat>)`
- U HTML-u stavljamo filter `translate` na ključ ili direktivu na roditeljski element ključa
- Putem `$translate.use(<ime lokaliteta>)` postavljamo lokalitet

primer 3

UI Bootstrap

- Bootstrap komponente napisane u AngularJS-u
- Pruža korisne komponente za GUI koje se ne moraju od nule pisati već se samo uključe i koriste
- Instalacija:
 - `bower install angular-bootstrap`
 - uključiti ga u `index.html`
 - registrovati ga kao zavisnost (`angular.module('app', ['ui.bootstrap'])`);
- Harmonika, odabirač datuma, padajući meni, komponenta za straničenje, modalni dialog, tabovi, itd.
- <http://angular-ui.github.io/bootstrap/>

ngOptions

- Atributska direktiva koja se postavlja u `select` tag kako bi se dinamički generisali `option` tagovi (slično kao `ngRepeat`, ali `ngOptions` ima bolje performanse, i veću fleksibilnost)
- Koristan za formiranje combobox-a sa objektima (a ne sa samo stringovima)
- Opciono se može staviti prazan `option` tag (`value=""`) u `select` tag za null vrednost
- Postoji više načina formiranja `ngOptions` izraza:
 - `<label> for <value> in <array>`
 - `ng-options="place.name for place in places"`
 - Labela označava šta da bude ispis, dok je value iterator i vrednost modela
 - `<select> as <label> for <value> in <array>`
 - `ng-options="place.zip as place.name for place in places"`
 - Select je vrednost modela, dok value predstavlja samo iterator

Track by

- `ngRepeat` u svom osnovnom obliku ne dozvoljava duplikate u nizu
- Track by može staviti na kraj `ngRepeat` i `ngOptions` izraza i omogućava:
 - Ubacivanje duplikata `ng-repeat="n in [1,2,2,3,3] track by $index"`
 - Bolje performanse kad na track by stavimo neki jedinstven atribut elementa niza, npr. `id` `ng-repeat="employee in employees track by employee.id"`
- `<value> in <array> track by <trackExpression>`
- Napomena: Kod `ngOptions` ne treba koristiti `select` i `track by` izraz u isto vreme