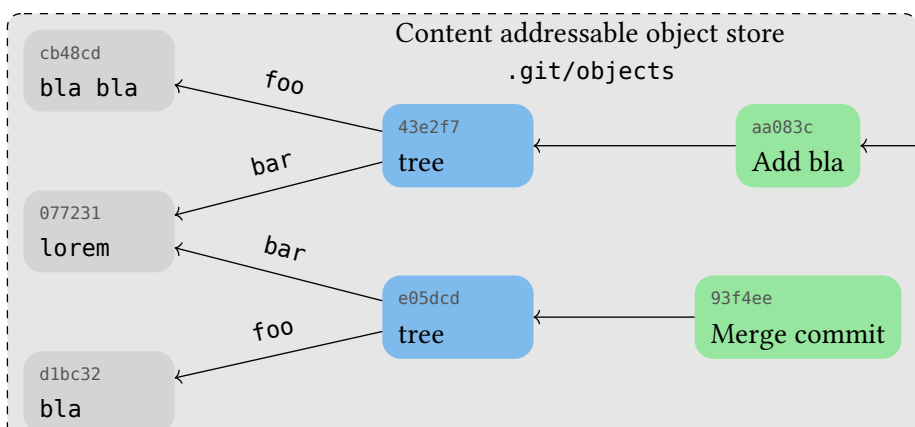
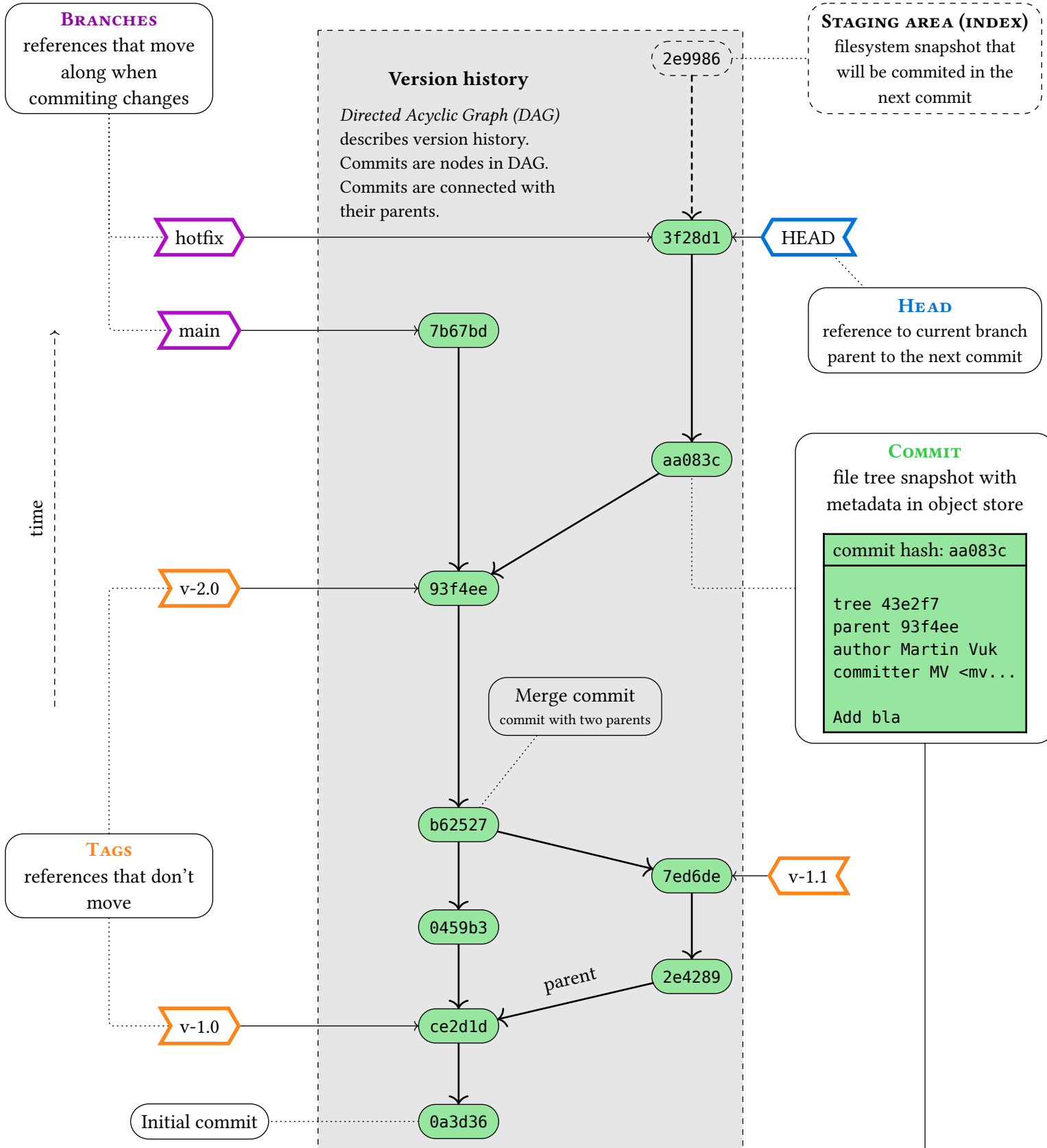




Git za matematike

Naučili se bomo, **kako Git deluje**. Spoznali bomo, da so v ozadju Gita **vsebinsko naslovljiva shramba podatkov, Merklejeva drevesa in usmerjen aciklični graf**.

Cilj: Razumeti *logiko* Gita. Ko razumemo, kaj je v ozadju, lahko operacije, kot so merge, rebase in reset preporsto razložimo s preoblikovanjem grafa in premikanjem kazalcev po grafu.

Čas branja: 30 min

Kaj je Git?

Git je kot **časovni stroj** za datoteke. Uporabniku omogoča, da vidi **pretekle različice** datotek, sprememinja datoteke, **brez skrbi, da bi kaj pokvaril** in datoteke **deli z drugimi**. Git namreč hrani vsebino direktorija z datotekami in celotno zgodovino različic datotek iz preteklosti. Za vsako različico hrani Git zapis o avtorju, datumu in opis sprememb, ki so nastale v primerjavi s predhodno različico.

Sisteme, ki omogočajo hranjenje preteklih različic datotek, imenujemo **sistemi za nadzor različic** (angl. version control system (VCS)) ali **sistemi za upravljanje z izvirno kodo** (angl. Source Code Management (SCM)).

Poleg nadzora različic Git omogoča, da datoteke spreminja več uporabnikov na različnih računalnikih hkrati. Zato je Git **distribuiran sistem za nadzor različic** (angl. Distributed Version Control System (DVCS)).

Git repozitorij

Git repozitorij je direktorij v katerem je poddirektorij `.git`, ki vsebuje vso zgodovino sprememb in ostale podatke, ki jih Git potrebuje.

Vnos: posnetek stanja

Osnovna enota v Gitu je **vnos** (angl. **commit**). Vnos je posnetek stanja zabeleženih datotek v trenutku, ko je bil ustvarjen. Poleg vsebine datotek vsak vnos vsebuje še metapodatke o avtorju, datumu vnosa in opisom sprememb. Vsakemu vnosu je prirejena **zgoščena vrednost vnosa** (angl. **hash**), ki je 40-mestna heksadecimalna vrednost, izračunana s SHA-1, in je natanko določena z vsebino shranjenih datotek in metapodatkov vnosa.

Git obravnava podatke v repozitoriju kot **posnetke stanja** in ne zgolj kot zaporedje **sprememb**. To je ena glavnih razlik med Gitom in predhodnimi sistemi za upravljanje različic (glej [Kaj je Git?](#)).

Vnose in vsebino datotek hrani Git v **skladišču objektov**. Do objektov v skladišču lahko dostopamo, če poznamo njihovo **zgoščeno vrednost**. Objekti, ki jih Git hrani v skladišču so **vnosi, posnetki direktorijev** in **posnetki posameznih datotek**.

zgoščena vrednost: 8dd6d4bdaeff93016bd49474b54a911131759648
tree 65c47feec7465e80492620a48206793e078702e0
parent 16f2994757f1213935b8edb9ae7fee3a8e9ec98d
author MV <mv@example.com> 1765235698 +0100
committer MV <mv@example.com> 1765235698 +0100
Dodaj bla

Tabela 1: Primer vnosa v Gitu. Vnos vsebuje zgoščeno vrednost posnetka direktorija(tree), zgoščeno vrednost starševskega vnosa (parent) in metapodatke. Tudi sam vnos je natančno določen z zgoščeno vrednostjo.

Posnetki direktorijev so v Gitu posebne vrste objekti tipa `tree`. Vsebujejo zgoščene vrednosti in metapodatke o datotekah in direktorijih, ki jih vsebuje.

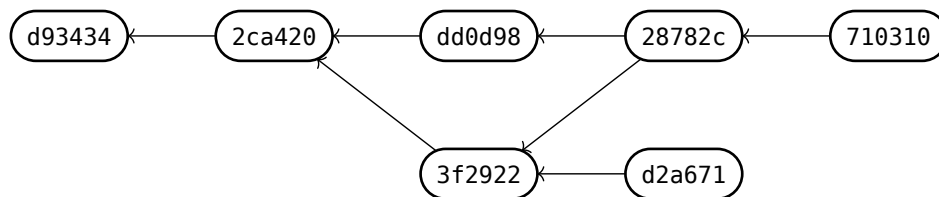
zgoščena vrednost: d934342ca420dd0d9828782c7103103f2922d2a6		
100644 blob	76018072e09c5d31c8c6e3113b8aa0fe625195ca	bar.txt
100644 blob	ba0e162e1c47469e3fe4b393a8bf8c569f302116	foo.txt
040000 tree	3b8bfca88b2cc4127ce5909eb3a7395e8b5f2b6a	podmapa

Tabela 2: Primer posnetka direktorija v Gitu (objekt tipa `tree`). Posnetek vsebuje zgoščene vrednosti datotek in direktorija, ki jih vsebuje. Uporaba zgoščenih vrednosti natančno določa vsebino posnetka direktorija.

Skladišča objektov v Gitu je **skladišče vsebinsko naslovljivih objektov**. Dostop do objekta je mogoč, če poznamo **zgoščeno vrednost** njegove vsebine. Dejansko je referenca na posamezen objekt v Gitu preprosto zgoščena vrednost vsebine tega objekta. Poleg tega deluje zgoščena vrednost vsebine kot kontrolna vsota. To pomeni, da lahko enostavno preverimo verodostojnost vsebine, ki je shranjena v Gitu. Skladišče objektov je v direktoriju `.git/objects`.

Zgodovinski graf sprememb

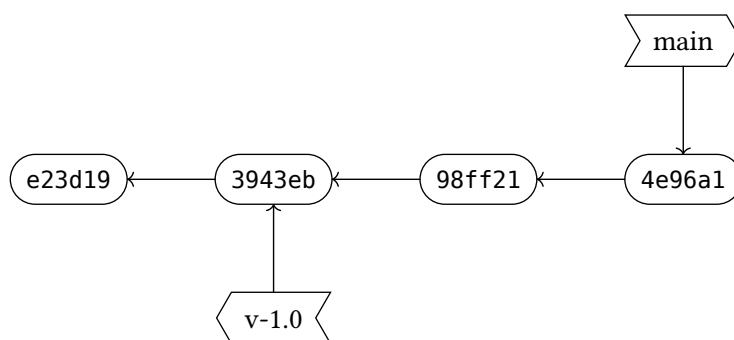
Posamezni vnosi so povezani v **usmerjen acikličen graf** (angl. kratica **DAG**), ki ponazarja zgodovino sprememb. Vsak **vnos** je **vozišče** v grafu in izhaja iz enega ali več starševskih vnosov. Izjema je prvi vnos. **Povezave** v grafu povezujejo vnose z njihovimi starši.



Slika 1: Vnosi v Gitu kot usmerjen graf. Vsak vnos(razen prvega) ima povezavo na vnose iz katerih izhaja.

Reference: veje in značke

Poleg objektov kot so *vnosi*, *posnetki direktorijev* in *posnetki datotek* pozna Git še reference. Reference so kazalci z določenim imenom na posamezen vnos.

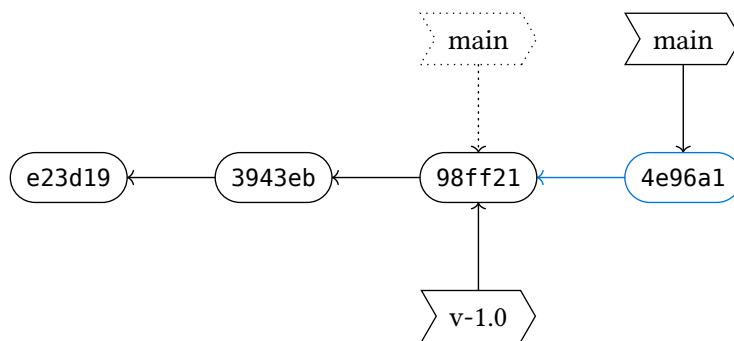


Slika 2: Veja (angl. **branch**) ali značka(angl. **tag**) je preprost kazalec na posamezen vnos(angl. **commit**).

Referenc Git ne hrani v skladišču objektov, temveč posebej v direktoriju `.git/refs`. Zato so reference vezane na posamezen repozitorij in se lahko razlikujejo med različnimi kloni določenega repozitorija.

Veja (angl. **branch**) je posebne vrste referenca, ki se premika, ko dodajamo nove vnose. Vsakič ko ustvarimo nov vnos, se trenutno aktivna veja premakne na novo ustvarjeni vnos.

Značka (angl. **tag**) je referenca, ki je statična in se ne premika več, ko jo enkrat ustvarimo.

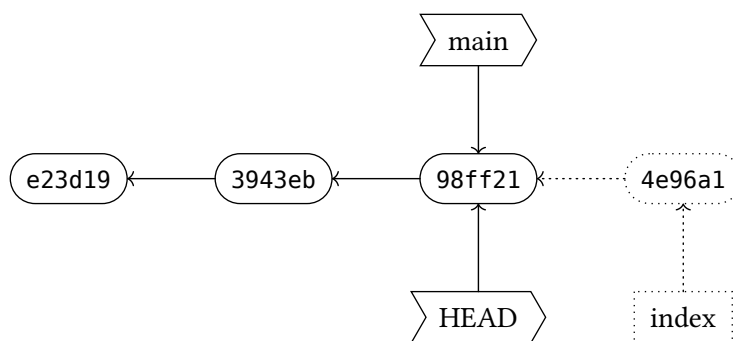


Slika 3: Ko ustvarimo nov vnos, se aktivna veja `main` premakne naprej, značka `v-1.0` pa ostane tam, kjer je bila.

Veje in značke nimajo v Gitu nobenega posebnega pomena, razen tega, da so reference na vnose. Pomen posameznih vej je stvar dogovora med uporabniki. Tako se pogosto uporablja različne veje za različne namene: `main` ali `master` je navadno glavna veja razvoja, veje z imeni `stable`, `production`, `development` in podobno označujejo različne stopnje razvoja programske opreme, veje s predpono `feature-` označujejo razvoj novih funkcionalnosti.

Vse te pomene damo vejam ljudje, ki sodelujemo v nekem Git repozitoriju. Za Git so vse veje in značke zgolj preprosti kazalci na določen vnos.

HEAD je posebna referenca, ki kaže na trenutno aktiven vnos. Vnos, na katerega kaže *HEAD* bo starševski vnos naslednjega vnosa, ki ga bomo dodali.



Slika 4: **HEAD** je referenca na trenutno aktiven vnos/vejo.

Delo z Git

Opis dela z Gitom presega namen tega dokumenta. Zato vas raje preusmerimo na uradno dokumentacijo:

<https://git-scm.com/cheat-sheet>

Povzetek

Samostalniki:

- **Vnos** (angl. **commit**) je posnetek trenutnega stanja projekta, shranjen kot vozlišče v zgodovinskem grafu (DAG), ki vsebuje spremembe datotek ter metapodatke (avtor, čas, sporočilo).
- **Zgoščena vrednost vnosa** (angl. **commit hash**) je 40-mestna heksadecimalna vrednost, izračunana s SHA-1, ki enolično identificira vnos na podlagi njegove vsebine.
- **Veja** (angl. **branch**) je premična oznaka, ki kaže na določen vnos v zgodovini in se samodejno premakne naprej, ko dodajamo nove vnose. Veje omogočajo vzporedne razvojne linije z različnimi spremembami.
- **Oznaka** (angl. **tag**) je statična oznaka, ki trajno kaže na določen vnos. Za razliko od veje se oznaka, nikoli ne premika samodejno, zato se uporablja predvsem za označevanje pomembnih točk v zgodovini, kot so izdaje ali stabilne verzije.

- **Delovna kopija** (angl. **workout copy**) je direktorij v katerem urejamo datoteke, ki jih nato vnesemo v Git. V delovni kopiji imajo na začetku datoteke isto vsebino kot je vsebina trenutno aktivnega vnosa (HEAD). Spremembe, ki jih naaredimo na delovni kopiji lahko zabeležimo v nov vnos.
- **Oddaljen repozitorij** (angl. **remote**) je povezava(url) na oddaljen repozitorij, s katerim izmenjujemo vsebino.

Glagoli (akcije):

- **Checkout** prenese vsebino vnosa v delovno kopijo: `git checkout neka-veja`
- **Commit** ustvari nov vnos: `git commit -m 'Sporočilo'`
- **Add** doda vsebino, ki bo v naslednjem vnosu: `git add dodaj_me.txt`
- **Pull** pobere vsebino iz oddaljenega repozitorija in uskladi lokalno vejo z oddaljeno: `git pull`
- **Push** potisni lokalne vnose na oddaljeni repozitorij in uskladi oddaljeno vejo z lokalno: `git push`
- **Fetch** pobere nove vnose, veje in oznake iz oddaljenega repozitorija: `git fetch`
- **Reset** spremeni kam kaže trenutno izbrana veja: `git reset a239f9e91`
- **Merge** ustvari nov vnos, ki združi dve ločeni veji v eno: `git merge main`
- **Rebase** prestavi vnose v trenutno izbrani veji na izbran vnos: `git rebase main`