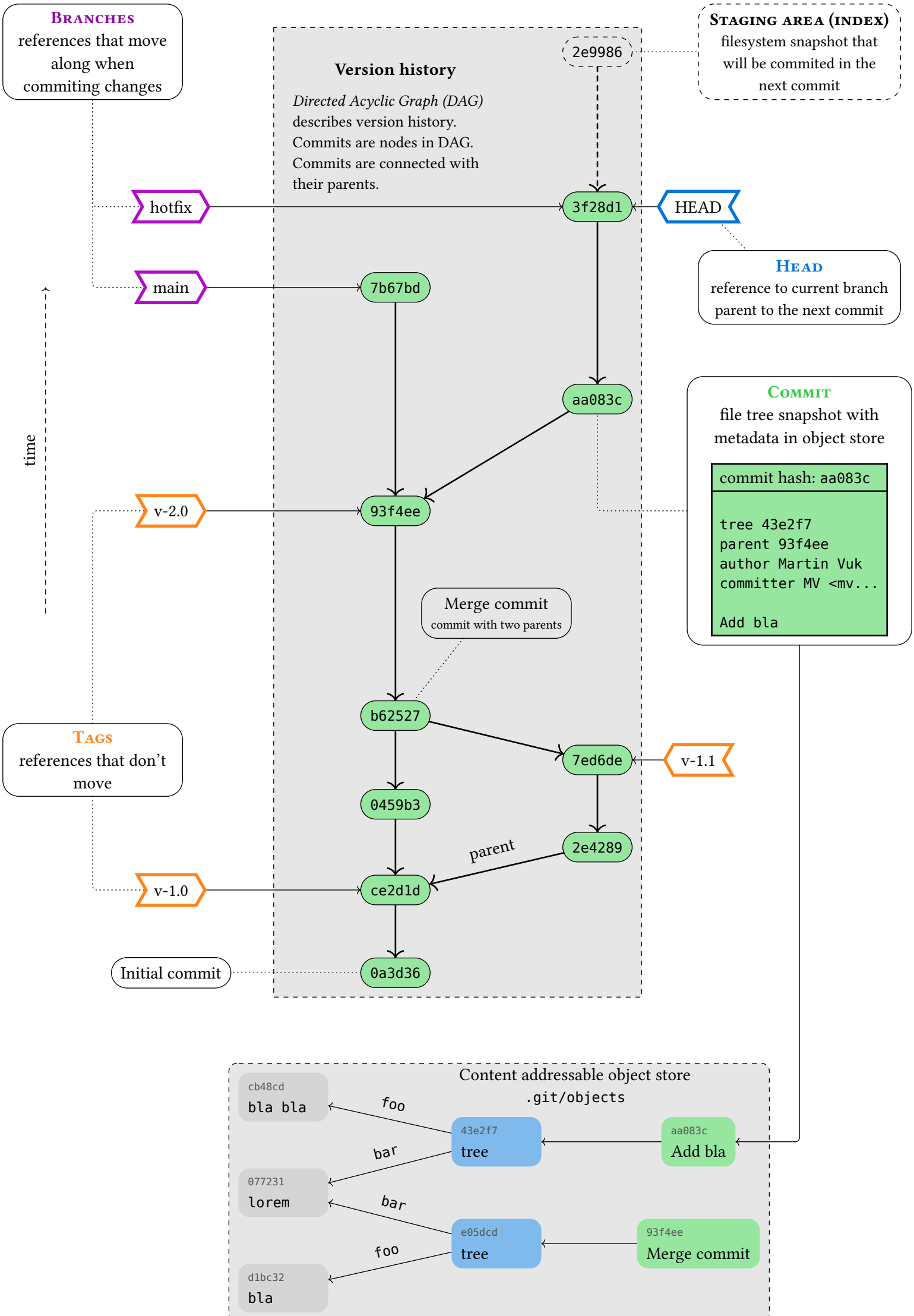

Branches
references that move along when commiting changes
Staging area (index)
filesystem snapshot that will be commited in the next commit
Version history
Directed Acyclic Graph (DAG) describes version history. Commits are nodes in DAG. Commits are connected with their parents.
2e9986
hotfix
3f28d1
HEAD
Head
reference to current branch parent to the next commit
main
7b67bd
aa083c
Commit
file tree snapshot with metadata in object store
commit hash: aa083c
tree 43e2f7
parent 93f4ee
author Martin Vuk
committer MV <mv...
Add bla
time
v-2.0
93f4ee
Merge commit
commit with two parents
b62527
Tags
references that don't move
7ed6de
v-1.1
0459b3
2e4289
parent
v-1.0
ce2d1d
Initial commit
0a3d36
Content addressable object store
.git/objects
cb48cd
bla bla
foo
43e2f7
tree
aa083c
Add bla
bar
077231
lorem
bar
e05dcd
tree
93f4ee
Merge commit
foo
d1bc32
bla

Git za matematike

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

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

1. 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**. Poleg časovnega stroja je Git **razpršeno skladišče datotek**. Omogoča, da datoteke **hkrati ureja več uporabnikov** na različnih računalnikih in kasneje spremembe **združi**.

Git 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. Vse te imformacije dajejo podroben pregled nad zgodovino sprememb.

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

Poleg nadzora različic Git omogoča hkratno spreminjanje datotek več uporabnikov na različnih računalnikih. Zato je Git distribuiran sistem za nadzor različic (angl. Distributed Version Control System (DVCS)).

V nadaljevanju bomo obravnavali naslednje teme:

- *Podatkovno skladišče:* Kako Git uporablja zgoščevalno funkcijo in Merklejeva drevesa za hranjenje posnetkov vsebine direktorija.
- *Zgodovina sprememb:* Kako zgodovino predstavimo z usmerjenim acikličnim grafom, v katerem so vozlišča različice in ki povezuje različice z njihovimi neposrednimi predhodniki.
- *Reference*: Kako preproste reference (kazalci) na vsebino omogočajo bliskovito preklaplanje med različicami in preprečijo popolno zmešnjavo, ko več ljudi hkrati spreminja iste datoteke.

2. Podatkovno skladišče

2.1. Git repozitorij

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

2.2. 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.

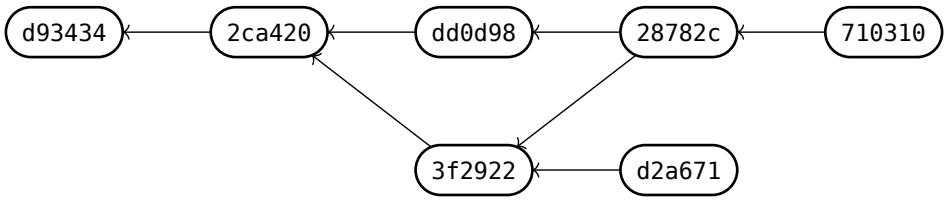
zgoščena vrednost: `d934342ca420dd0d9828782c7103103f2922d2a6`		
`100644`	`blob 76018072e09c5d31c8c6e3113b8aa0fe625195ca`	`bar.txt`
`100644`	`blob ba0e162e1c47469e3fe4b393a8bf8c569f302116`	`foo.txt`
`040000`	`tree 3b8bfca88b2cc4127ce5909eb3a7395e8b5f2b6a`	`podmapa`

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. To pomeni, da je referenca na posamezen objekt v Gitu preprosto zgoščena vrednost(angl. hash) vsebine tega objekta. Po drugi strani je vsebina objekta določena z njegovo zgoščeno vrednostjo. To pomeni, da lahko enostavno preverimo verodostojnost vsebine, ki je shranjena v Gitu. Git hrani skladišče objektov v direktoriju `.git/objects`.

3. Zgodovinski graf sprememb

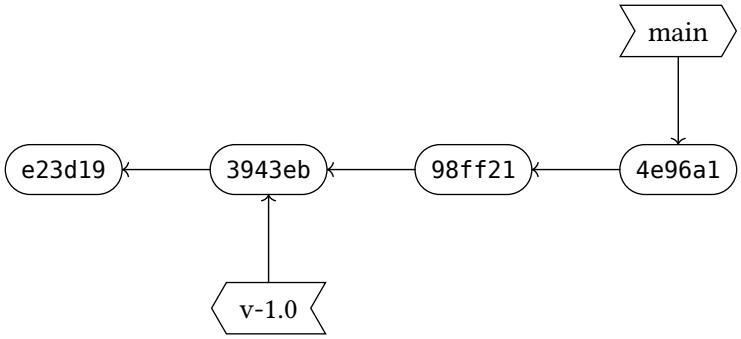
Posamezni vnosi so povezani v **usmerjen acikličen graf (DAG)**, ki predstavlja zgodovino sprememb. Vsak **vnos** je **vozlišče** v grafu. Vsak vnos izhaja iz enega ali več starševskih vnosov. Izjema je prvi vnos. **Povezave** v grafu povezujejo vnose z njihovimi starši.



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

4. Kazalci: 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.

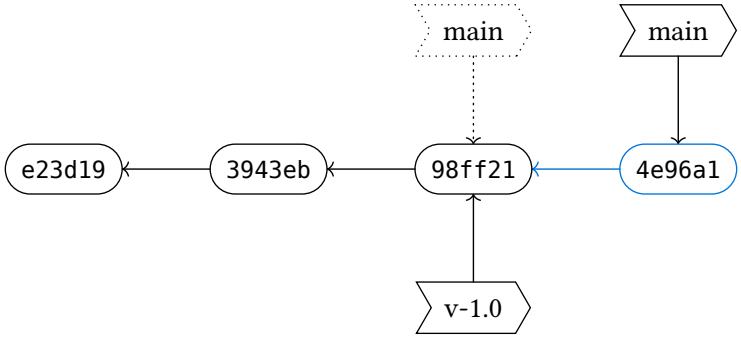


Slika 2: Veja (angl. branch) ali značka(angl. tag) je preprost kazalec na posamezen vnos(angl. commit).

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.

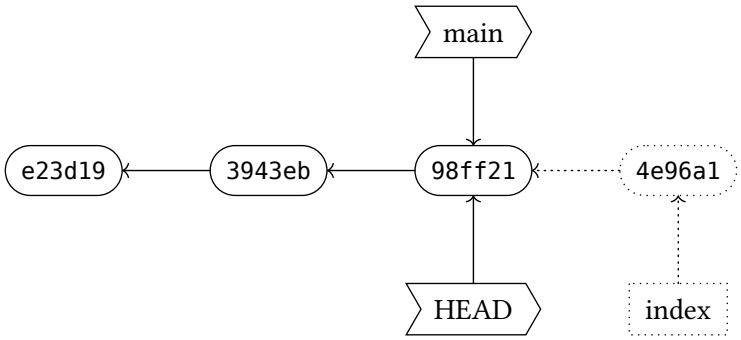


Slika 3: Ko ustvarimo nov vnos, se aktivna veja `main` premakne naprej, značka `v-1.0` pa ostane tam, kjer je bila.

Veje in značke nimajo v Gitu nobenega posebnega pomena, razen tega, da so reference na vnose. Pomen posamenznih 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 naslednjeg vnosa, ki ga bomo dodali.



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

4.1. Delo z Git

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

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

4.2. 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):

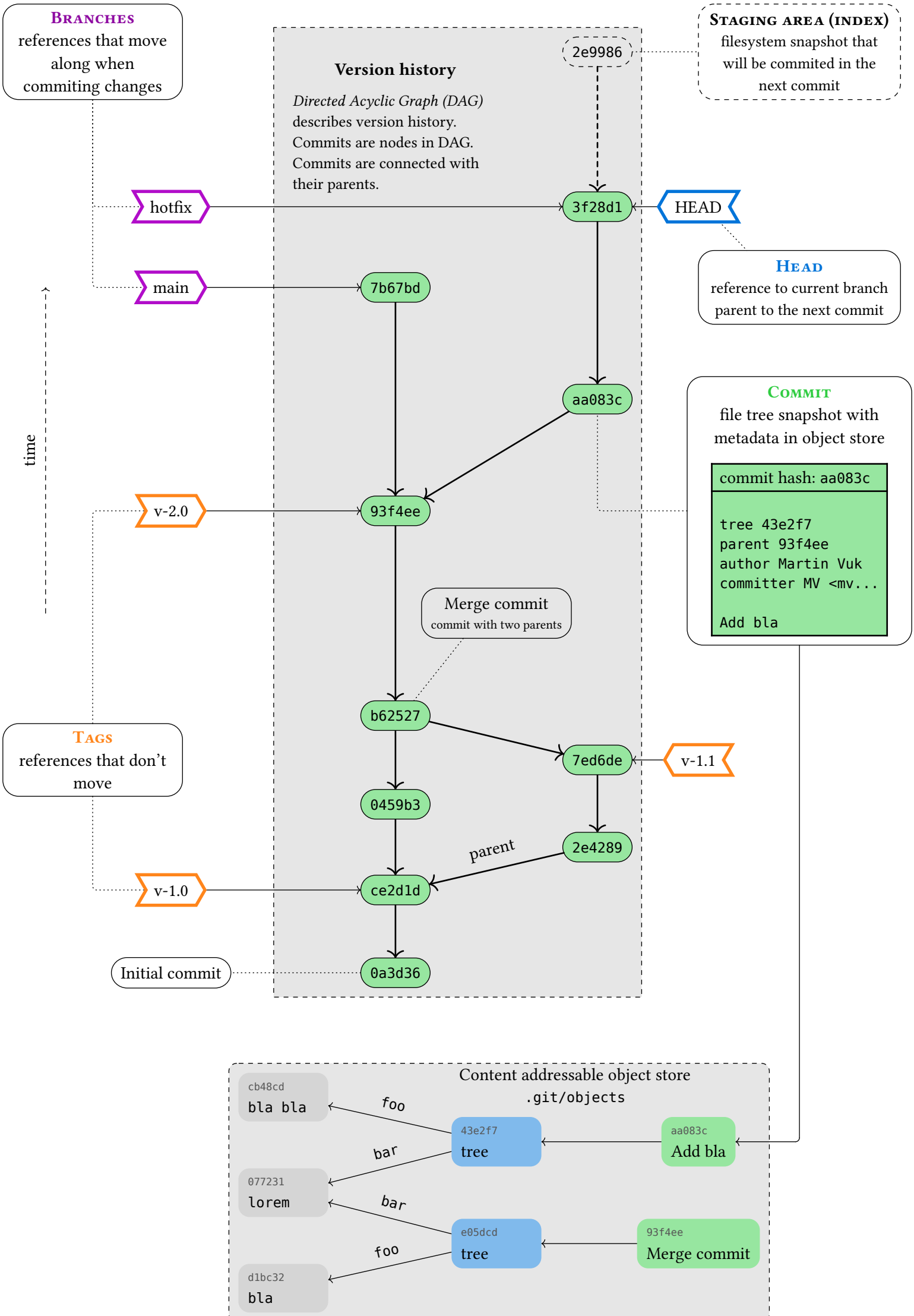
- **Checkout** prenese vsebino vnosa v delovno kopijo: `git checkout neka-veja`
- **Commit** ustvari nov vnos: `git commit -m 'Sporočilo'`
- **Add** doda vsebino, ki bo v naslednjem vnosu: `git add dodaj_me.txt`
- **Pull** poberi vsebino iz oddaljenega repozitorija in uskladi lokalno vejo z oddaljeno: `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`

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

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

Pri pripravi dokumenta sem uporabil Gemini 3. Vse odgovore sem preveril in uredil po svoje.

Sledi še skica, ki povzame vse komponente Git repozitorija.


Branches
references that move along when commiting changes
Staging area (index)
filesystem snapshot that will be commited in the next commit
Version history
Directed Acyclic Graph (DAG) describes version history. Commits are nodes in DAG. Commits are connected with their parents.
2e9986
hotfix
3f28d1
HEAD
Head
reference to current branch parent to the next commit
main
7b67bd
aa083c
Commit
file tree snapshot with metadata in object store
commit hash: aa083c
tree 43e2f7
parent 93f4ee
author Martin Vuk
committer MV <mv...
Add bla
time
v-2.0
93f4ee
Merge commit
commit with two parents
b62527
Tags
references that don't move
7ed6de
v-1.1
0459b3
2e4289
parent
v-1.0
ce2d1d
Initial commit
0a3d36
Content addressable object store
.git/objects
cb48cd
bla bla
foo
43e2f7
tree
aa083c
Add bla
bar
077231
lorem
bar
e05dcd
tree
93f4ee
Merge commit
foo
d1bc32
bla