Osnove sustava za čuvanje verzija koda

GIT

Studija slučaja

- 3 programera
  - Iskusni: Ivan i Mislav
  - Pripravnik: Luka
- Priprema – savršena
  - Razgovor s korisnikom
  - Funkcionalna specifikacija
  - Iterativni razvoj
  - Korisnički zahtjevi
  - Podjela zadataka

Studija slučaja

- Svi su na istoj lokaciji, unutar tvrtke
- Korištenje mrežnog diska za spremanje kôda
- Ivan radi zaseban korisnički zahtjev
- Mislav i Luka rade na istom
- Tijek radnje:
  - Ivan i Luka završili i spremili kôd
  - Matija završio i spremio kôd
- PROBLEM?!

Studija slučaja

- Na prezentaciji korisniku – greška u aplikaciji
- Problem: gaženje kôda
- Rješenje: upotreba sustava za čuvanje verzija kôda


git
Name
v1.0
v2.0
v2.1
v2.2
project
project-revised
project-final
project-final-for-real
Name
project
projectt
projecttttttttt

Sustav za čuvanje verzija koda

- VCS – *version control system*
- **Repozitorij** (baza) verzija koda
- Prati **promjene** u projektnim datotekama
- Omogućava
  - Spremanje **snimke stanja** *(snapshot)* u nekom trenutku

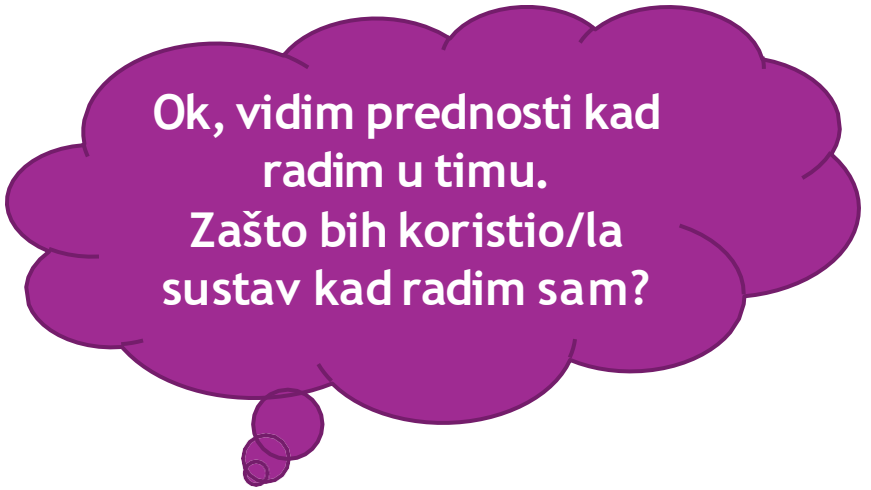
Sustav za čuvanje verzija koda

VCS ne ovisi o:

- Tehnologiji u kojoj razvijamo
- Alatima s kojim radimo (programski jezik, radni okvir, itd.)

Zašto koristiti sustav kontrole verzija?

- Koje su alternative?
  - Korištenje djeljene mape
    - „Ja radim na toj datoteci, ne dirajte”
    - Izuzetno je podložno pogreškama
    - Prije ili kasnije će netko pregaziti tuđe promjene
- Kad se koristi sustav za čuvanje verzija koda
  - Ljudi rade slobodno, istovremeno, bez straha
  - Sustav
    - Upozorava da je došlo do promjene u datoteci
    - Omogućava spajanje različitih verzija



Spremanje verzija

- Dobro je spremati verzije
  - Naporno bez nekog sustava za čuvanje verzija koda
- Scenarij:
  - Kad zaključite da trebate spremiti verziju – spremite cijelu mapu
    - Neovisno o tome u kojim datotekama je došlo to promjene
    - Kako ćete nazivati te mape?
      - Verzija1, Verzija2, Verzija3, …
      - V202004300700, V202004300830, V202004300911, …
    - Ogromna količina nepotrebni podataka
    - Kako ćete pratiti do kojih promjena je došlo na pojedinoj datoteci?
      - Dokumentirate li pažljivo izmjene u *PovijestPromjena.txt?*

- Taj mukotrpni posao, sustav za čuvanje verzija koda radi za vas
- Pamti promjene u pojedinim datotekama
- Sve je lako dostupno kad vam zatreba

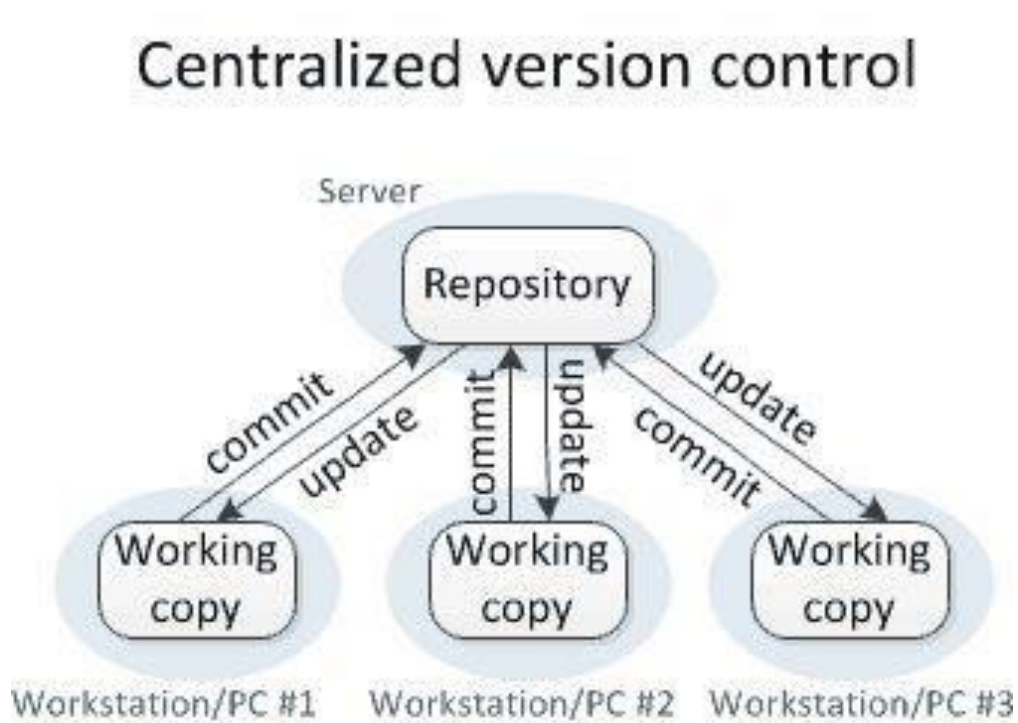
Povratak na neku prethodnu verziju

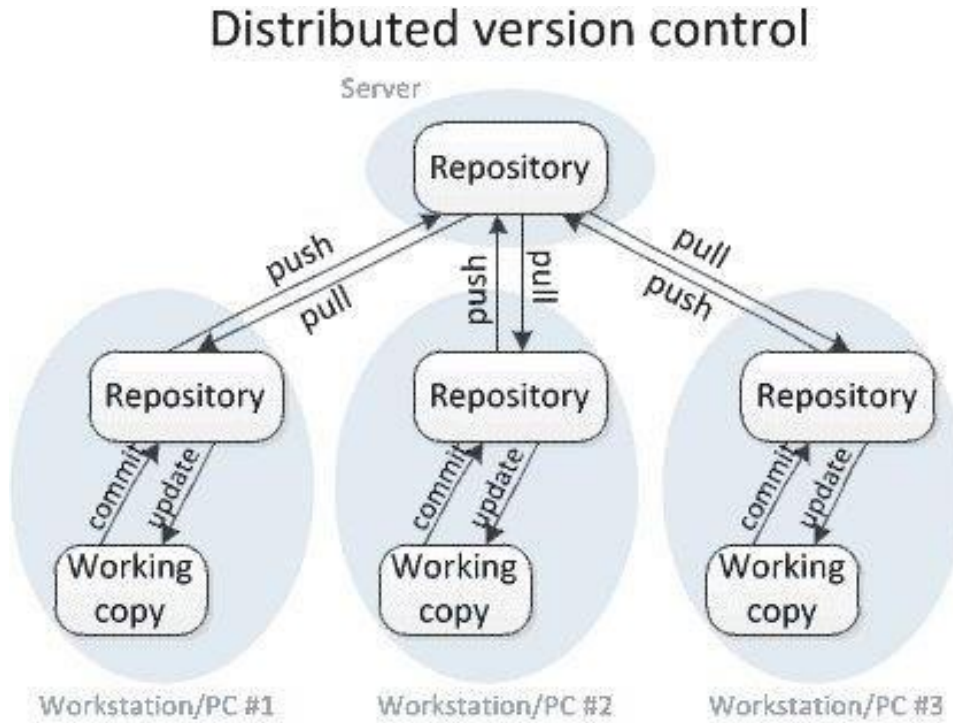
- Omogućava povratak na neku stariju verziju (datoteke ili cijelog projekta)
- *Sve je lako kad … ne možeš zabrljati* ☺
- Ako zaključite da to na čemu ste radili zadnjih par sati treba baciti u smeće, ništa lakše
- Svaki put kad spremite verziju, sustav vas traži unos komentara te izmjene
  - Olakšava razumijevanje nastalih izmjena
- Svaki član ima kod sebe povijest (svoju i cjelokupno)
  - Ako se centralni repozitorij uništi, postoje lokalni iz kojih možemo obnoviti projekt

Zahtjevi za sustav za čuvanje verzija koda

- Poništavanje promjena
- Sigurnosna kopija i vraćanje neke spremljene verzije
- Zajednički rad u timu
  - Spremanje napravljenog
  - Dohvat tuđih promjena
  - Usporedba koda
- Praćenje promjena (tko, kada, zašto)
- Grananje koda

Tipovi sustava za čuvanje verzija koda





SUBVERSION®

mercurial

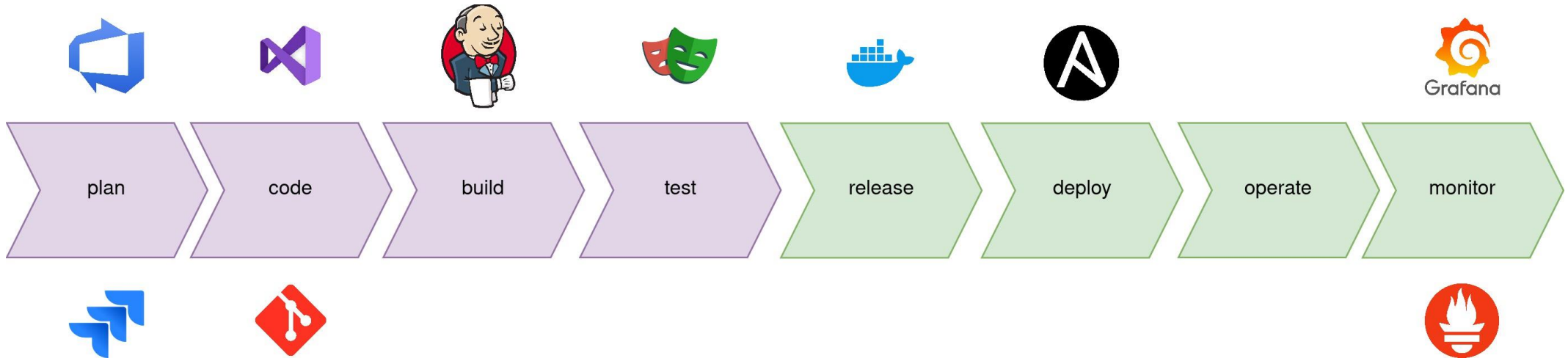
Primjeri sustava za čuvanje verzija koda

SUBVERSION®

mercurial

Primjeri sustava za čuvanje verzija koda

Servisi za čuvanje verzija koda


Grafana
plan
code
build
test
release
deploy
operate
monitor

!=


git

Git

- Distribuirani sustav za čuvanje verzija koda
- Autor: **Linus Torvalds**
- Godina: 2005
- Trenutna verzija: 2.49.0
- Besplatan software otvorenog koda
- Dostupan na Windows/Linux/MacOS

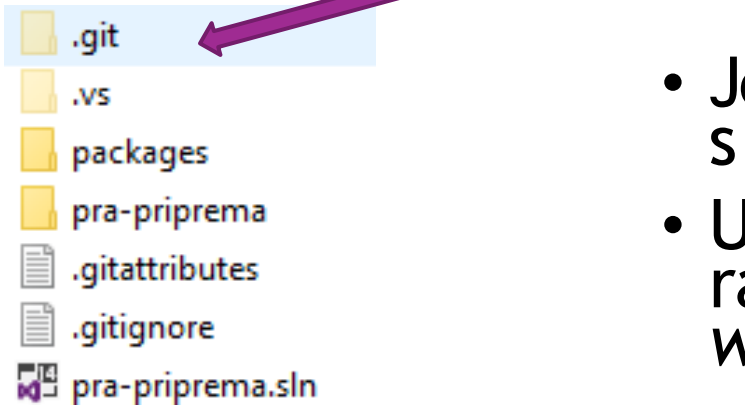
Osnovni koncepti

- Repozitorij
- Snimka stanja *(commit)*

Repozitorij

Što je to?

- „Skladište” ili „Baza” verzija
- Ne samo verzije, već i meta podaci



Lokalni

- Na vašem računalu
- Gdje je fizički taj repozitorij?
  - Skrivena .git mapa u *rootu* vašeg projekta
    - Skrivena je s razlogom ☺
- Jedino vi možete raditi s tim repozitorijem
- U kombinaciji s radnom mapom (eng. *working copy*)

Udaljeni

- Na udaljenom poslužitelju na internetu ili u vašoj lokalnoj mreži
- Samo mapa .git
  - Nema radne mape
- Članovi tima ga koriste za razmjenu promjena

Snimka stanja (commit)

- Snimaju trenutno stanje repozitorija
- Korisni za praćenje tijeka rada na projektu
- Omogućuju povratak na prethodna stanja

Tijek rada

1. rad na projektu

- Dodavanje novih datoteka
- Brisanje postojećih
- Dorade postojećeg koda

2. commit koda

- Tek kad gotova je neka cjelina
- Provjera što ste sve napravili
- Želite li sve to commitati ili samo djelomično?
- Definirati što točno commitate
- Komentar izmjene

3. dohvat tuđih izmjena

- Ako radimo u timu potrebno je periodički dohvaćati tuđe izmjene

Kako možemo početi raditi sa sustavom za čuvanje verzija koda?

Novi

- Projekt koji još nije u sustavu
- Inicijalizirate novi repozitorij za taj projekt

Postojeći

- Ako počinjete raditi na projektu koji je već na nekom udaljenom repozitoriju
- Saznate URL
- „Klonirate” ga na svoje računalo

Stvaranje repoziotirja

- **Lokalno**
  - Inicijalizirati

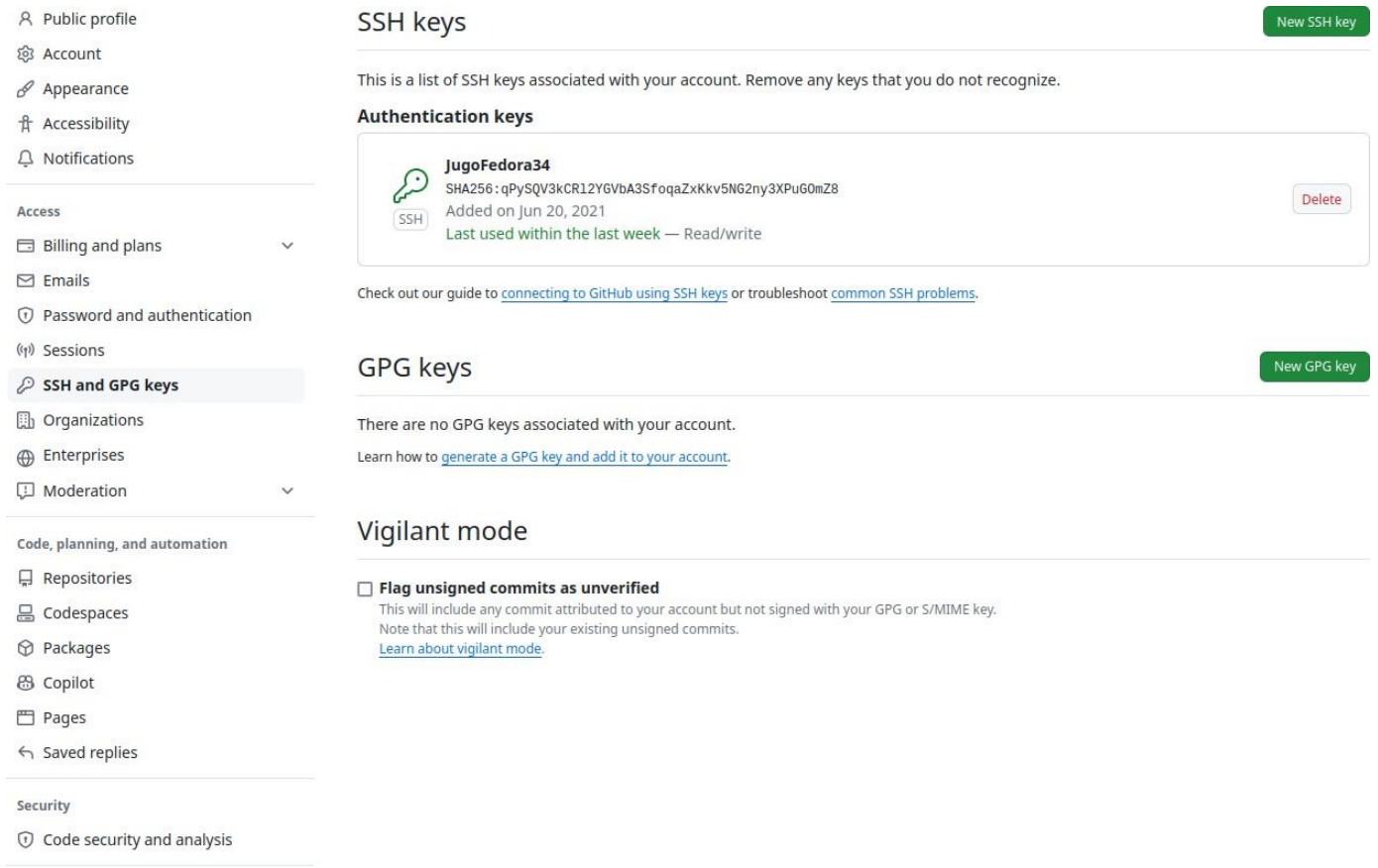
  > git init

- **Udaljeno**
  - Stvoriti repozitorij na servisu za čuvanje verzija koda (npr. GitHub)

  > git remote add <name> <url>

Rad s udaljenim repozitorijima

- HTTPS, koristeći access token
- SSH ključevi



Lista datoteka u .git mapi

```
total 52K
drwxrwxr-x  8 jugo jugo 4.0K Mar 24 23:07 .
drwxrwxr-x  8 jugo jugo 4.0K Mar 25 15:23 ..
drwxrwxr-x  2 jugo jugo 4.0K Mar 22 08:22 branches
-rw-rw-r--  1 jugo jugo   35 Mar 24 23:07 COMMIT_EDITMSG
-rw-rw-r--  1 jugo jugo  205 Mar 22 08:22 config
-rw-rw-r--  1 jugo jugo   73 Mar 22 08:22 description
-rw-rw-r--  1 jugo jugo   23 Mar 22 08:22 HEAD
drwxrwxr-x  2 jugo jugo 4.0K Mar 22 08:22 hooks
-rw-rw-r--  1 jugo jugo 1.1K Mar 24 23:07 index
drwxrwxr-x  2 jugo jugo 4.0K Mar 22 08:22 info
drwxrwxr-x  3 jugo jugo 4.0K Mar 22 08:24 logs
drwxrwxr-x 39 jugo jugo 4.0K Mar 24 23:07 objects
drwxrwxr-x  5 jugo jugo 4.0K Mar 22 08:24 refs
```

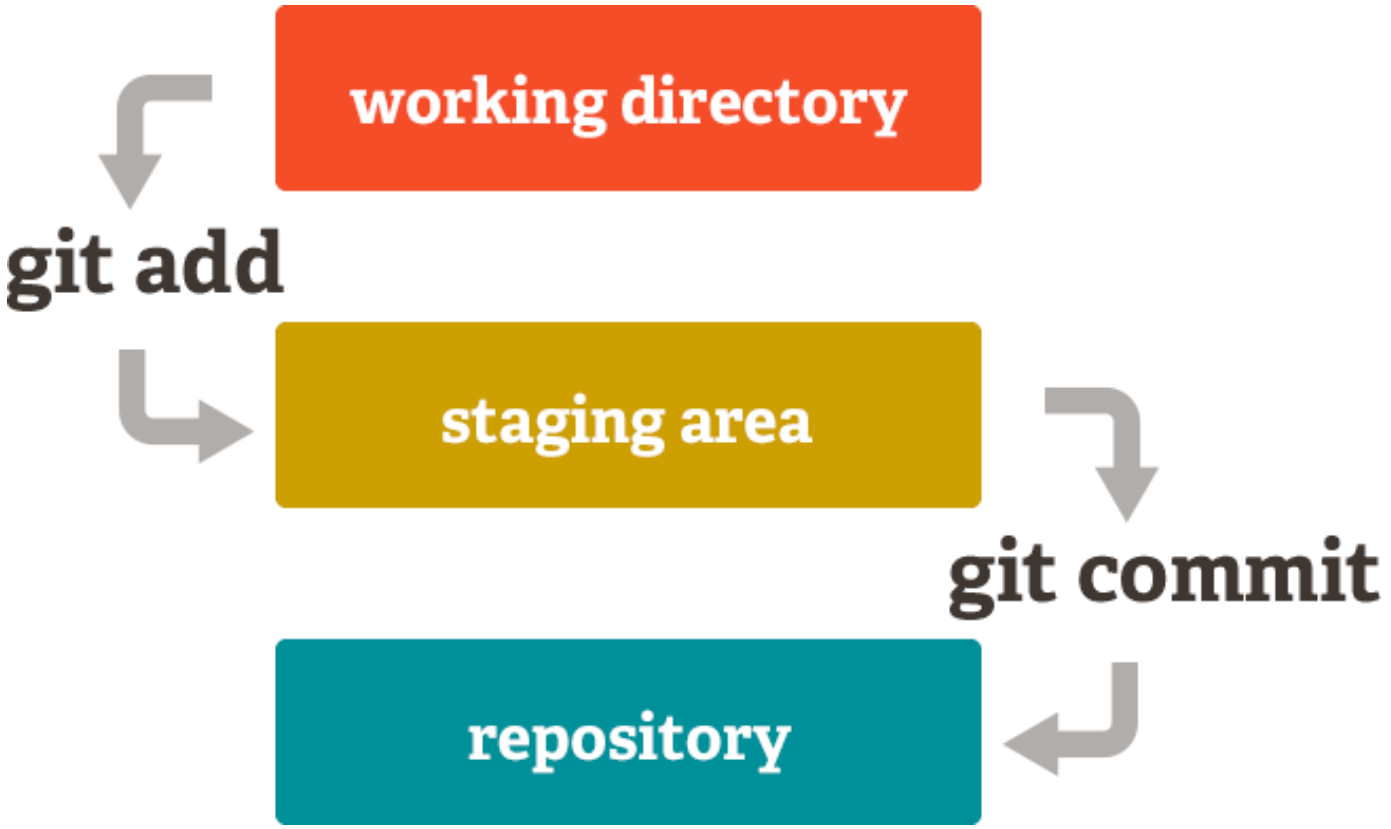
Spremanje promjena

Potrebno je pripremiti **stage:**

> **git add <filename>**

Nakon toga stvaramo **commit:**

> **git commit -m ‘<poruka>’**



Promjene

- Točno znamo tko je kada napravio koju promjenu

```
                    3 usages   bskracic
3/22/24 bskracic  9 class GraphConvolutionalNetwork(torch.nn.Module):
                        bskracic
3/22/24 bskracic 10     def __init__(self, num_node_features, num_classes, hidden_channels, adj_matrix):
3/22/24 bskracic 11         super(GraphConvolutionalNetwork, self).__init__()
3/22/24 bskracic 12         self.adj_matrix = adj_matrix
3/22/24 bskracic 13         self.conv1 = GCNConv(num_node_features, hidden_channels)
3/22/24 bskracic 14         self.lin = Linear(hidden_channels, num_classes)
3/22/24 bskracic 15         self.a = torch.nn.ReLU()
3/22/24 bskracic 16         self.num_classes = num_classes
3/22/24 bskracic 17
```

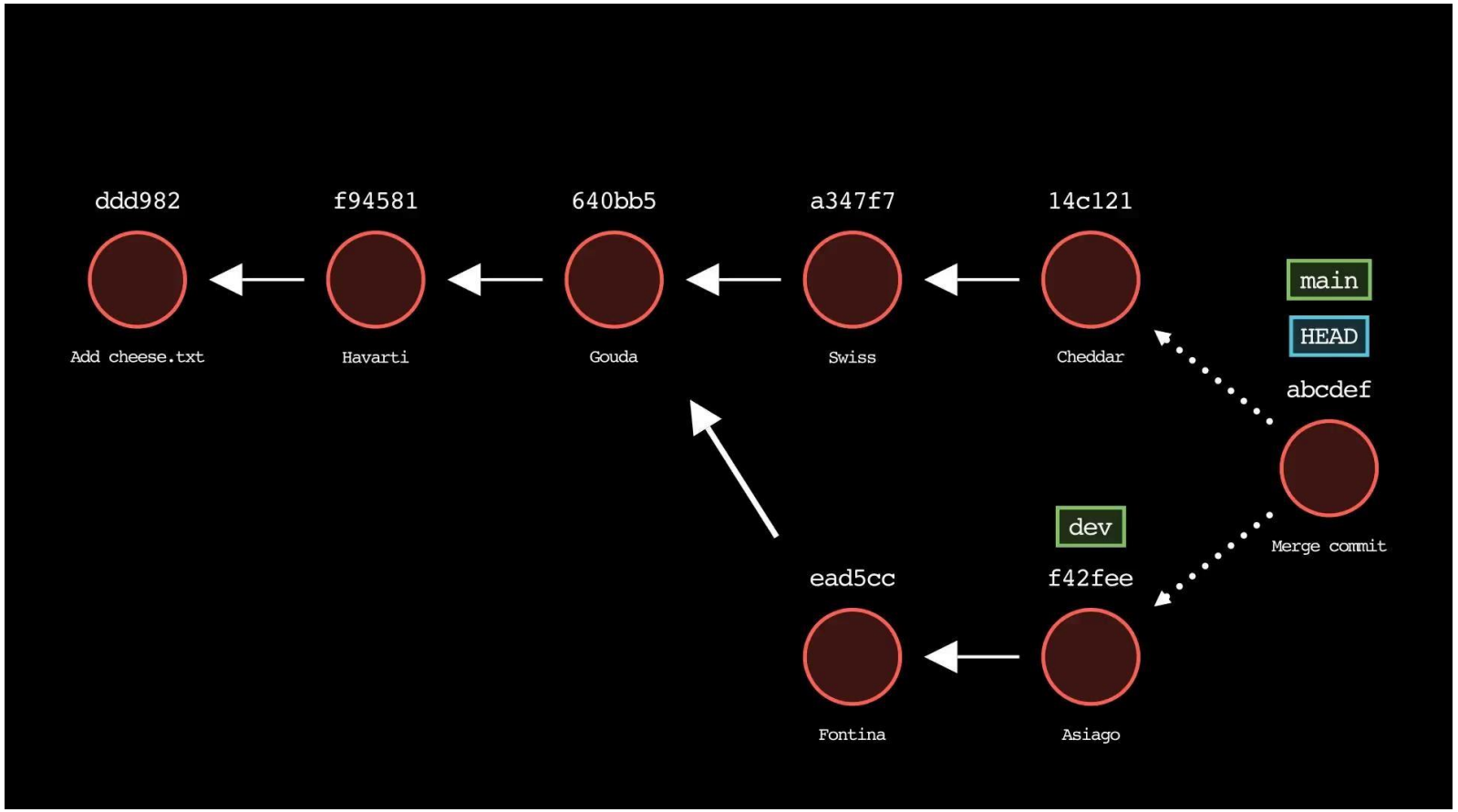
Log

```
commit 2eb699387fbe91a04235db97c9e07508896498ad
Author: bskracic <borna.skracic7@gmail.com>
Date:   Fri Mar 22 14:35:25 2024 +0100

    implemented generate dataset workflow

commit f7655b684c47b8eae18afc4600d39f7c68d3f5a7
Author: bskracic <borna.skracic7@gmail.com>
Date:   Fri Mar 22 08:24:18 2024 +0100

    inital commit
```

Spremanje promjena na udaljenom servisu

- git push <remote_name> <branch_name>

DAG promjena



Preuzimanje promjena

- **Fetch**
  - Naredba *fetch* dohvaća promjene i sprema ih u reference za udaljeni repozitorij i grane u lokalnoj mapi (*refs/remotes/<remote>/*)
  - *fetch* ne integrira promjene u radnoj mapi
- **Pull**
  - Naredba *pull* dohvaća promjene sa udaljenog repozitorija i integrira ih u lokalni repozitoriji
  - Mogući konflikti!

Rad sa postojećim udaljenim repozitorijem

- git clone <url>

Konfiguracija gita

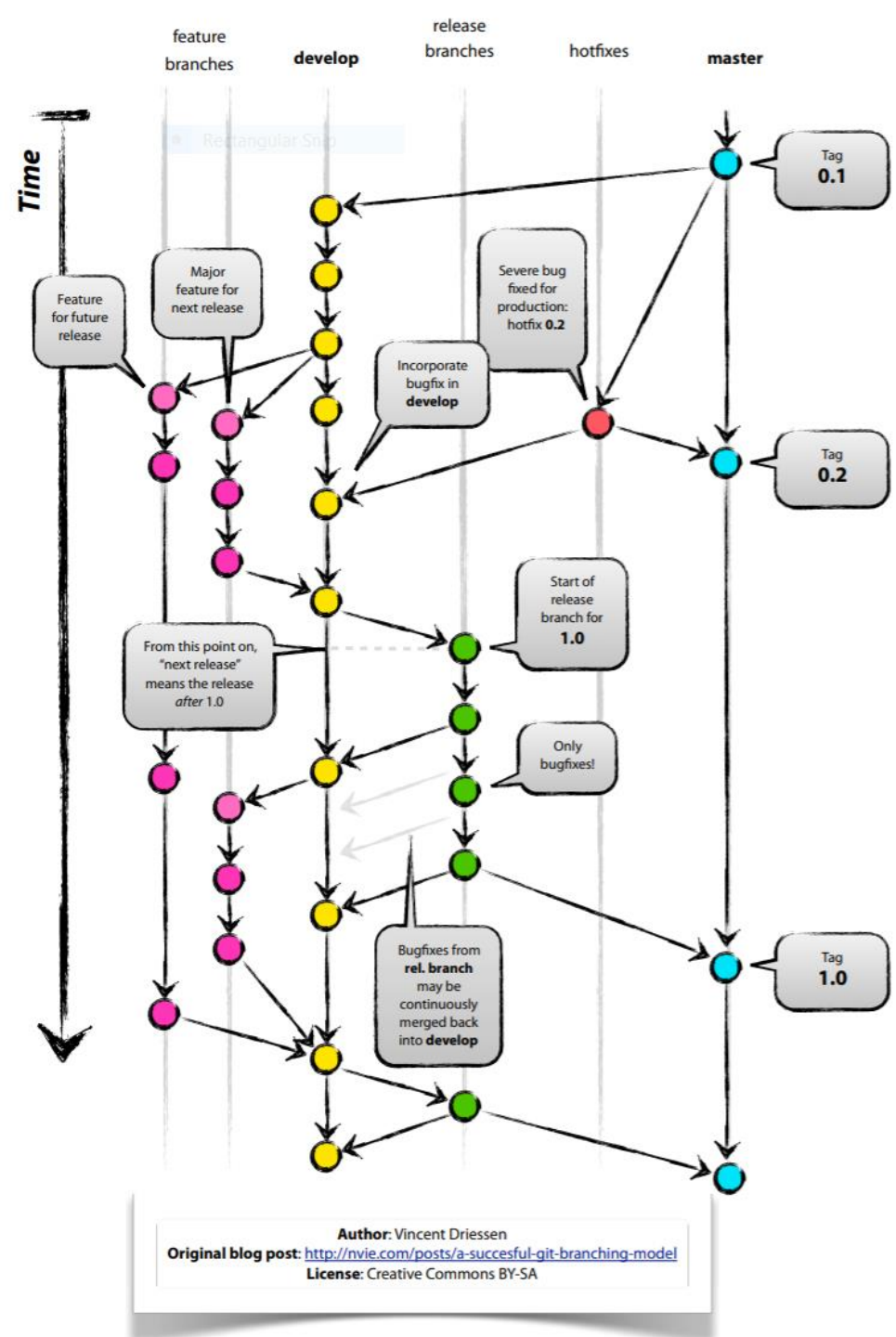
- git config -l

Rad s granama koda

- Glavna grana je **master (main)**
- Možemo stvarati nove grane s novim skupom commitova koji su neovisni o glavnoj grani i drugim granama
- Završetkom rada, uglavnom se grana značajke spaja sa glavnom granom
- Preporučljivo za paralelni rad na više značajki

Git naredbe [obavezno]

- git init
- git log
- git status
- git add <filename>
- git commit –m ‘<message>’
- git push <remote_name> <branch_name>
- git fetch
- git pull <remote_name> <branch_name>
- git clone <url>
- git config <key>=<value>
- git branch
- git checkout <branch_name>



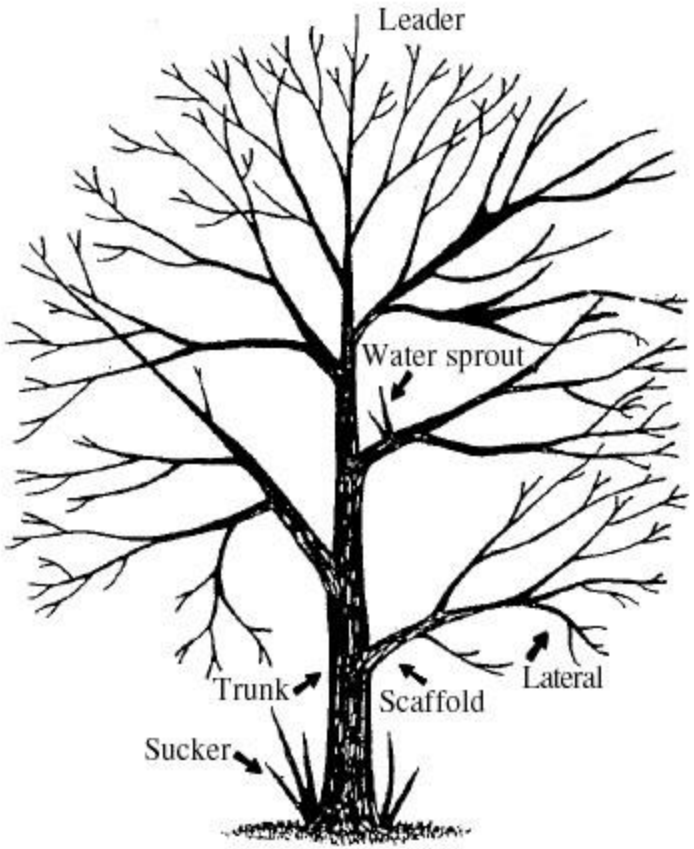
Grananje (eng. *branching*)

Grana (eng. *branch*)

- Pokazivač na snimku stanja (commit)
- Omogućuje i pospješuje paralelni razvoj

- Eksperimentiramo i uvodimo nove promje
  ne utječući na glavno stanje

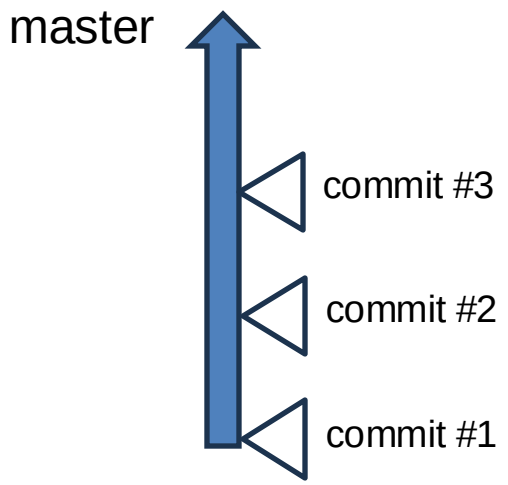
- Grane se mogu ponovno spojiti (eng. **mer**
  nazad (najčešće u main/master granu)



Typical above-ground tree framework

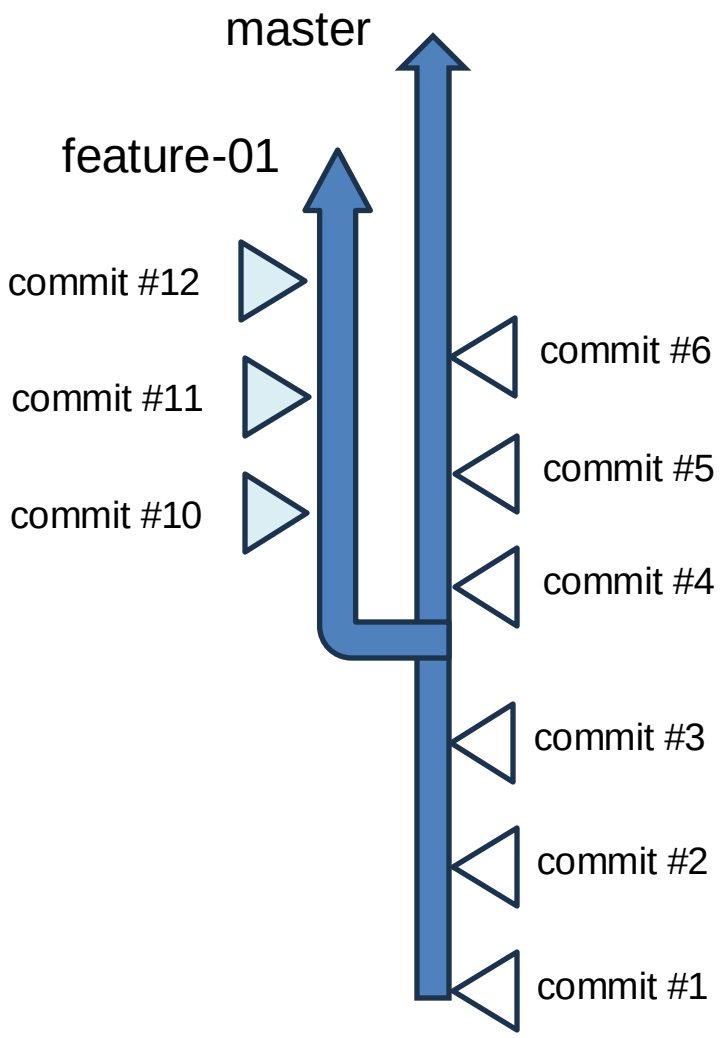
Trunk based development

- Jedna grana (main/master)
  - Nemamo drugih grana
- Sve promjene se direktno pohranjuju na glavnu granu
- Feature toggles
  - Kako odrediti koje promjene se samo testiraju, a koje su u (stabilnoj) produkcijski
- Zahtjeva veliku zrelost tima



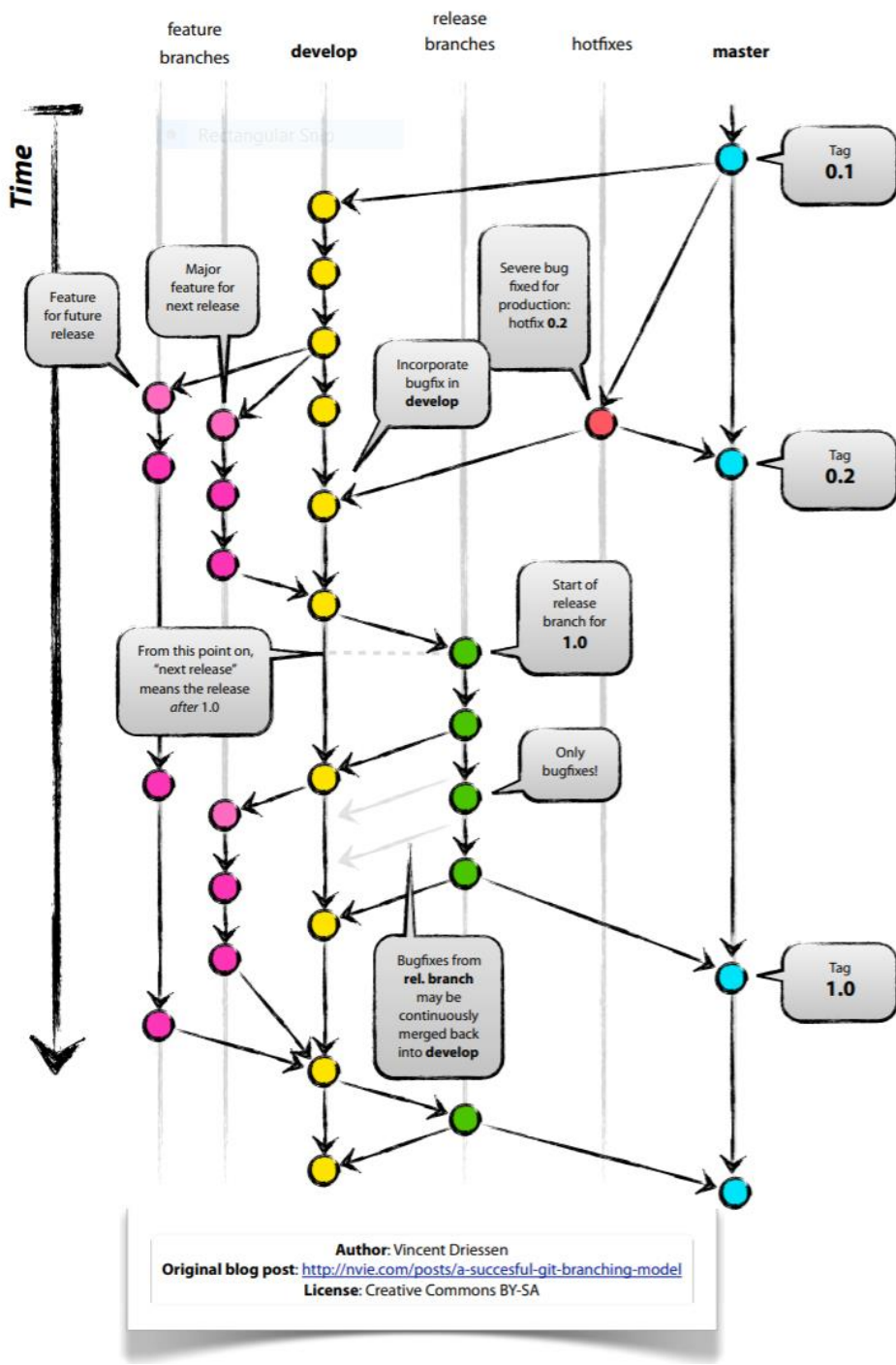
Feature branching

- Jedna glavna grana (main/master)
- Svaka značajka na kojoj radimo ima zasebnu granu
- Značajan utjecaj CD (Continous Delivery)
- Obavezni PR (Pull Request)!

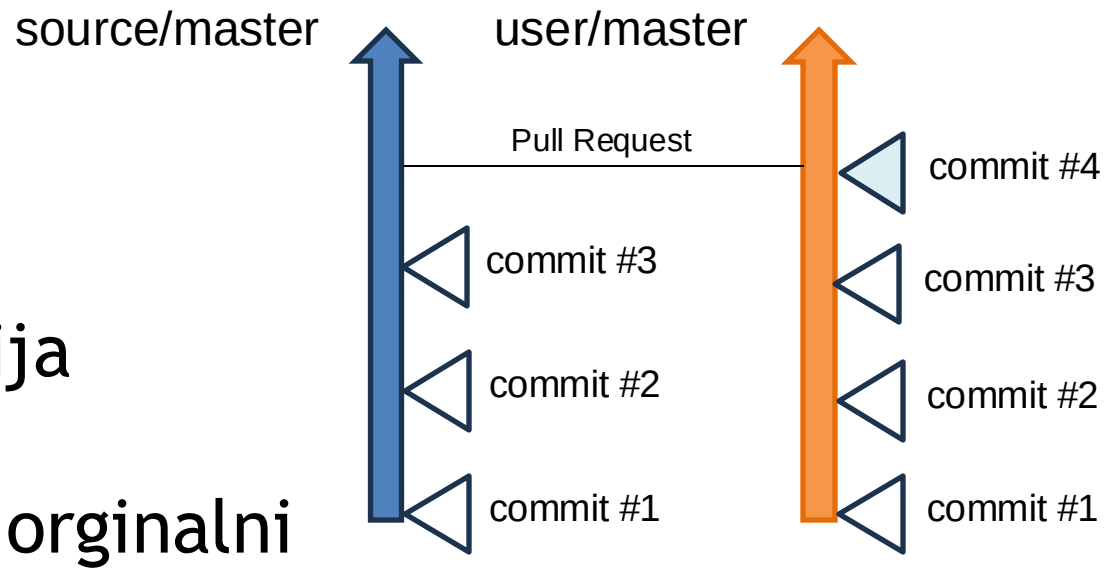


Git Flow

- Sličan kao feature branching
- Postoje najčešće dvije glavne grane
  - produkcijska = **main/master**
  - development = **dev**
- Olakšano praćenje promjena u oba okruženja
- Po potrebi se granaju i feature grane
- Može se proširiti na environmental branching (dev1, dev2, uat, qa, prod, itd.)



Forking strategy



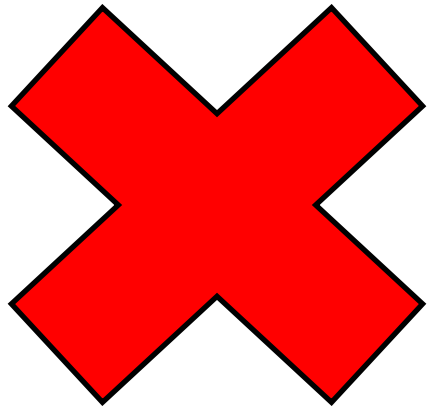
- Kreiramo fork (nova zasebna kopija udaljenog repozitorija)
- Pri završetku rada, radimo PR na orginalni repozitorij s integracijom naših promjena
  - Olakšano je upravljanje ovlastima za push akcije

- Najčešće korišteno u projektima otvorenog koda

Konflikti prilikom spajanja

Nastaju prilikom spajanja sadržaja promijene iz jedne u drugu granu

Potrebno je ih razriješiti prije stvaranja *merge commita*



Skripta

Više detaljnih uputa opisano u dokumentu na IE:
Prakticne-Osnove-Gita.pdf