Versionshantering

med git

David Everlöf, 21/10 - 2020

Vem är jag?

- AFRY (ÅF)
- Utvecklar appar till iPhone
- Swedbanks iPhone-app

Upplägg

- Interaktiva delar
- Repetition
- Övningar

Idag lär vi oss

- Versionshantering
- Git
- GitHub
- Kollaborera
- Fråga!

Vad är versionshantering?

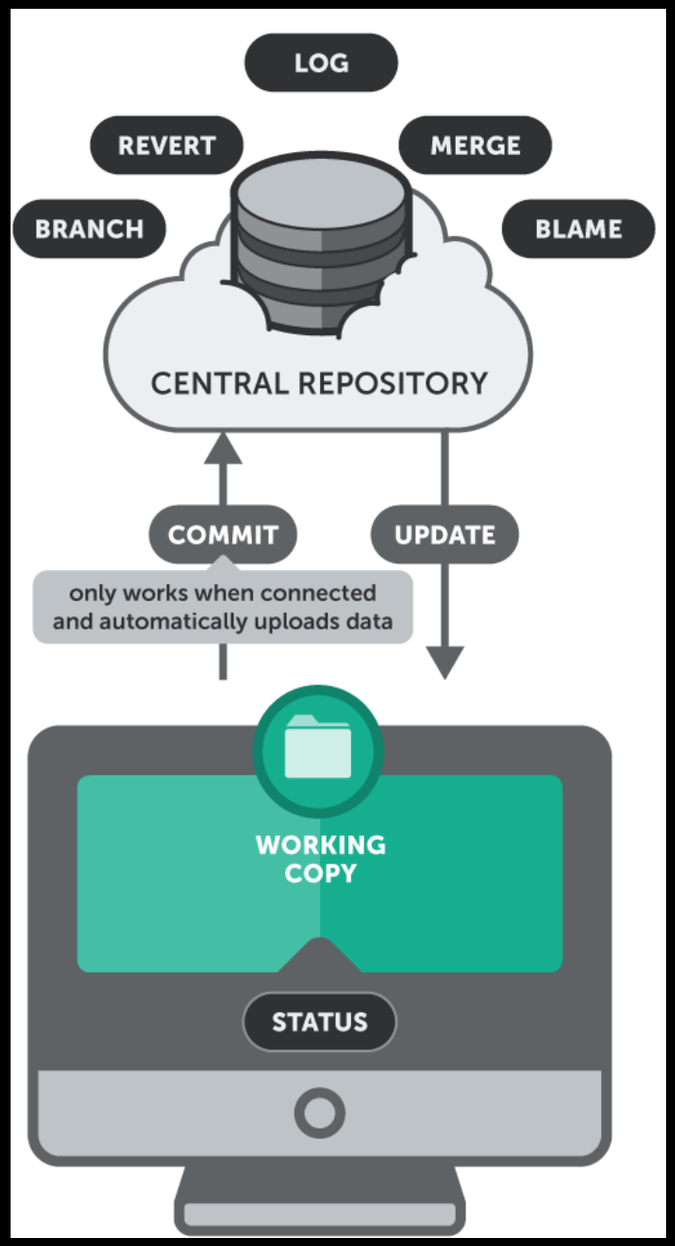
- Spara ändringar av filer över tid
- Återställa filer till tidigare versioner
- Koordinera arbetet mellan flera utvecklare
- Vilken kod finns i vilken version

Vad är ett repository?

- All historik för ett projekt/komponent
- Repo

Två grupper av verktyg

Centraliserade



Bildkälla: https://scriptcrunch.com/295/

Vad är Git?

- Git är ett verktyg för versionshantering
- Väletablerat
- Tjänster

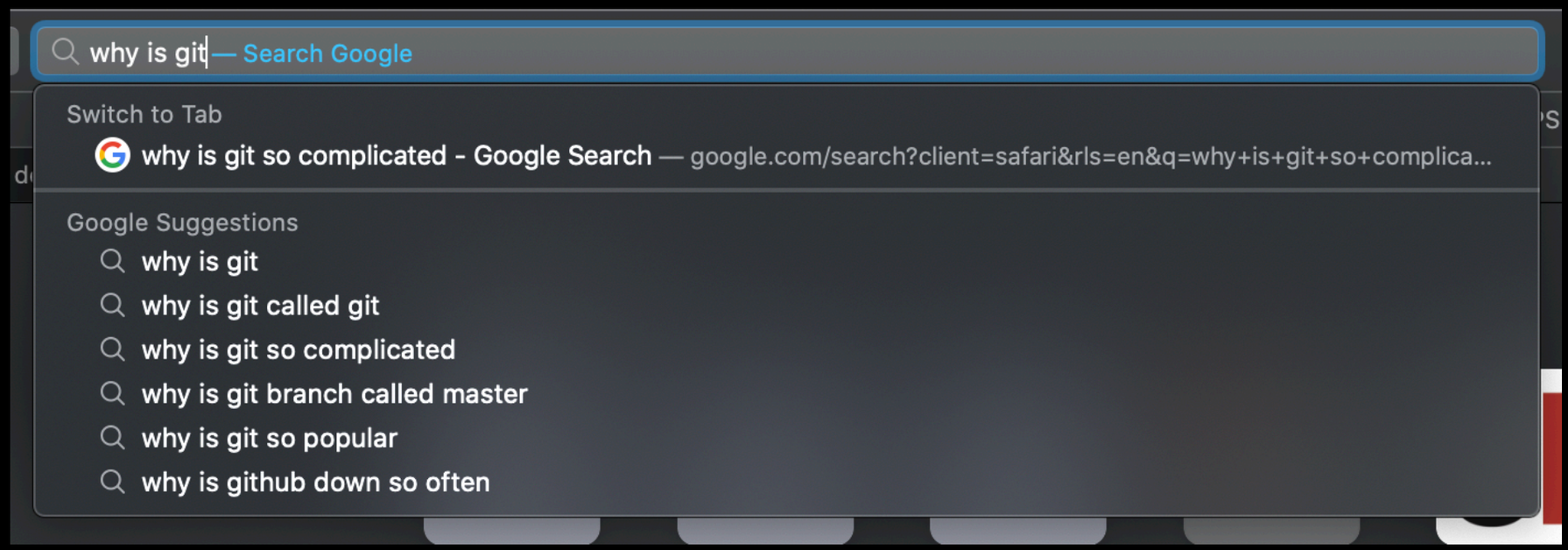
Verktyg

- Finns en uppsjö med GUI-verktyg
- SourceTree
- CLI (Command Line Interface)

Kort historia

- Linuxkärnan
- Email 1991-2002
- BitKeeper 2002
- 2005 BitKeeper ville ha betalt
- Linus Torvalds, Linux-skaparen tog fram git
- 2016 bytte BitKeeper ut sin betalningsmodell mot en OpenSource-modell
- https://github.com/git/git/commits/master?after=d4a392452e292ff924e79ec8458611c0f679d6d4+60739&branch=master

Git


why is git — Search Google
Switch to Tab
why is git so complicated - Google Search — google.com/search?client=safari&rls=en&q=why+is+git+so+complica...
Google Suggestions
why is git
why is git called git
why is git so complicated
why is git branch called master
why is git so popular
why is github down so often

Home

PUBLIC

Stack Overflow

Tags

Users

FIND A JOB

Jobs

Companies

TEAMS What's this?

Free 30 Day Trial

to whether to enable it. – Yimin Rong Mar 20 '18 at 1:45

14 @YiminRong That can be done with Git's `alias` feature: git-scm.com/book/en/v2/Git-Basics-Git-Aliases – Edric Oct 5 '18 at 14:50

8 @RomainValeri - Same way undo works everywhere else. – Yimin Rong Mar 25 at 12:35

8 @YiminRong Not buying it. People would still fumble and undo things not to be undone. But more importantly, `git reflog` is already close to what you describe, but gives the user more control on what's to be (un)done. But please, no, "undo" does not work the same everywhere, and people would *expect* many different things for the feature to achieve. Undo last commit? Undo last action? If last action was a push, undo how exactly, (reset and push) or (revert and push)? – RomainValeri Mar 25 at 13:23

show 7 more comments

86 Answers

Active Oldest Votes

Undo a commit and redo

23539

```
$ git commit -m "Something terribly misguided"              # (1)
$ git reset HEAD~                                           # (2)
<< edit files as necessary >>                               # (3)
$ git add ...                                               # (4)
$ git commit -c ORIG_HEAD                                   # (5)
```

1. This is what you want to undo.
2. This does nothing to your working tree (the state of your files on disk), but undoes the commit and leaves the changes you committed unstaged (so they'll appear as "Changes not staged for commit" in `git status`, so you'll need to add them again before committing). If you *only* want to *add* more changes to the previous commit, or change the commit message[1], you could use `git reset --soft HEAD~` instead, which is like `git`

Commit

- “Spara” utvalda filer
- När man skapar ett nytt repo finns det inga commits, dvs det finns ingen historik
- Alla commits som skapas bildar en graf, som representerar historiken för ett repo
- Lättare att se visualiserat

Commit C
Fil1

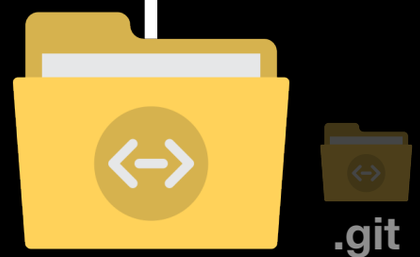
Radera Fil2

Commit B
Fil1
Fil2

Modifiera Fil1

Commit A
Fil1
Fil2

Lägg till Fil1
Lägg till Fil2



Repo

`git init`

Working tree

Fil 1

Fil 2

Fil 3

Staging area
(Index)

Fil 1

Fil 2

Fil 3

Historik

Commit B
Fil 1
Fil 2
Fil 3

Commit A
Fil 1
Fil 2
Fil 3

.git

Vi provar!

Interaktiv del 1

Working tree

Staging area
(Index)

History

att-handla.txt

“att-handla.txt” = untracked

Working tree

Staging area
(Index)

History

att-handla.txt

att-handla.txt

Commit A
att-handla.txt

git add att-handla.txt

git commit -m “Lade till ‘att-handla.txt’

Vad ingår i en commit?

- **Meddelande - Varför gjordes denna förändring?**
- **Filträd / mappstruktur - vilka filer och mappar ingår i commiten?**
- Författare - vem skapade commiten?
- Tidpunkt - när skapades commiten?
- Får unik identifierare (“SHA-summa”)
  - T.ex. 3cf5a7ef2f029ae4f26e13886bcbd05e08e330b8

Övning 1

Vår första commit

Commit A

att-handla.txt

(Tips! Skapa ett lokalt repo via: New > Local Repository)

Nu behöver vi köpa fler saker och göra en “att göra” lista!

Interaktiv del 2

Working tree

Staging area
(Index)

History

att-handla.txt

att-göra.txt

att-handla.txt

git diff

Commit A
att-handla.txt

Working tree

att-handla.txt

att-göra.txt

Staging area
(Index)

att-handla.txt

att-göra.txt

History

Commit B
att-handla.txt
att-göra.txt

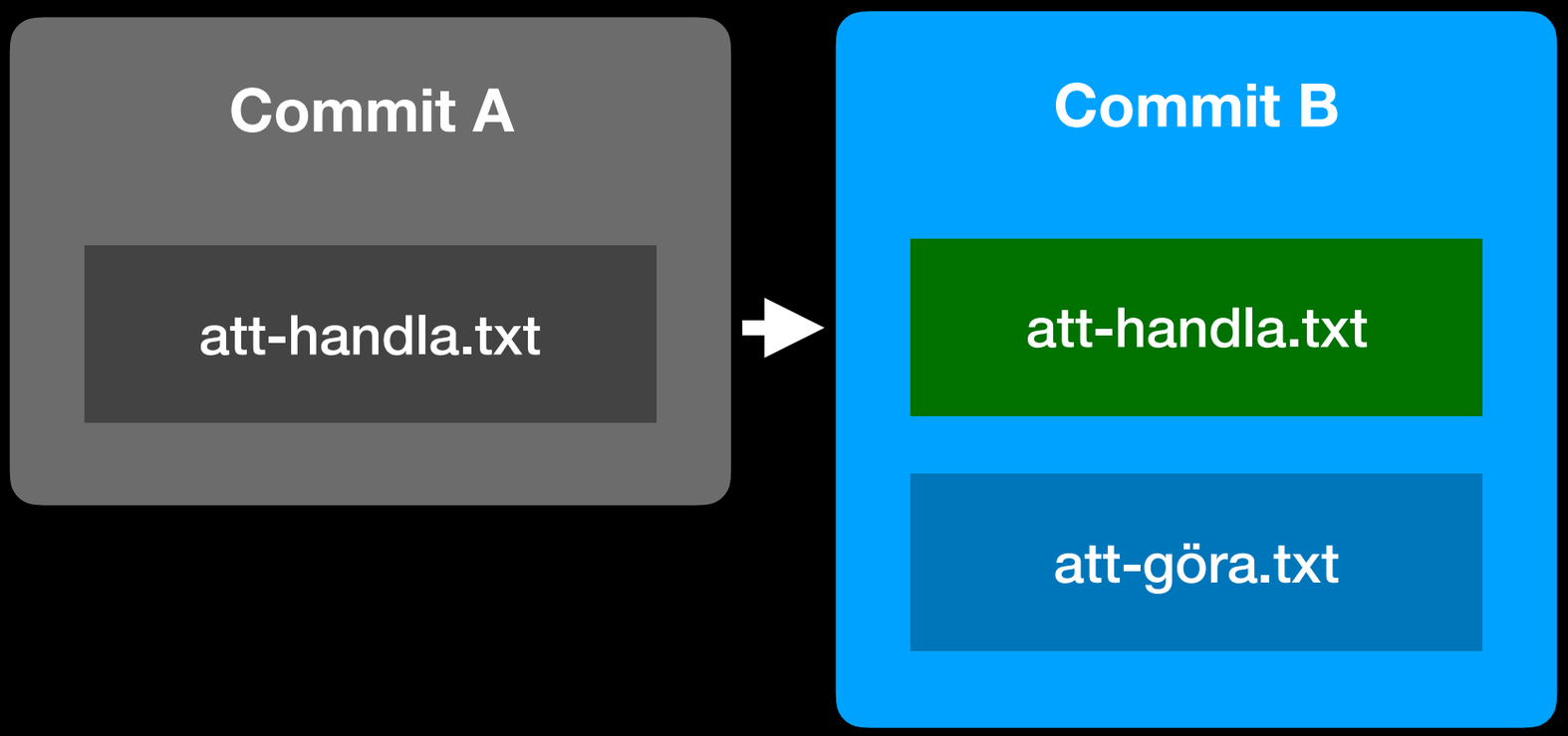
Commit A
att-handla.txt

git add att-handla.txt
git add att-göra.txt

git commit -m “Skapa ‘att-göra.txt’ + fler inköp”

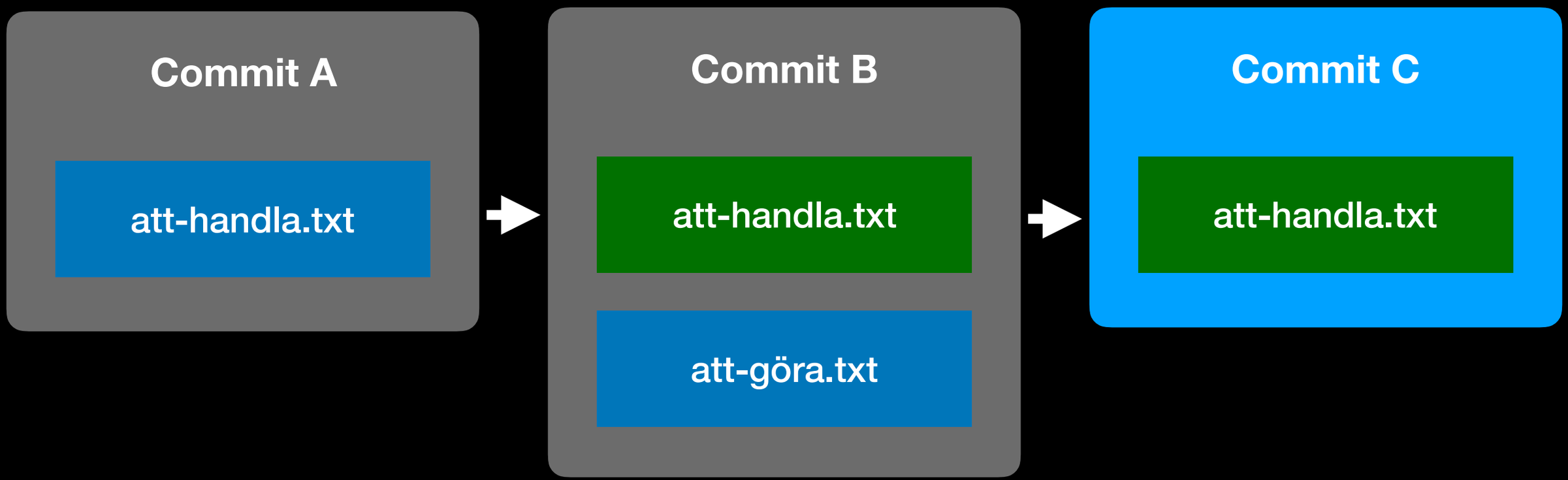
Övning 2

Lägg till en fil och modifiera en befintlig



Övning 3

Radera vår ‘att-göra.txt’ fil



Ångra en ändring i vårt working tree

Working tree

Staging area
(Index)

History

att-handla.txt

att-handla.txt

Commit C
att-handla.txt

Commit B
att-handla.txt
att-göra.txt

Commit A
att-handla.txt

git restore att-handla.txt
(git checkout -- att-handla.txt)

Ångra en ändring i vårt working tree

Interaktiv del 3

Ångra en “stagad” fil

Working tree

Staging area
(Index)

History

att-handla.txt

att-handla.txt

Commit C
att-handla.txt

Commit B
att-handla.txt
att-göra.txt

git diff --staged

Commit A
att-handla.txt

git add att-handla.txt

git restore --staged att-handla.txt
(git reset HEAD att-handla.txt)

Ångra en “stagad” fil

Interaktiv del 4

Kort sammanfattning

- git add
- git status
- git commit
- git log
- git diff
- git restore (checkout)
- git restore --staged (reset)

.gitignore

- Låt git ignorera vissa filer
- Deklarera vilka filer/mappar vi ej är intresserade av
- Måste commitas
- Ex. log-filer, bygg-artefakt, “hemligheter”

Övning 4

1. Vilken typ av filer verkar rimligt att lägga i .gitignore när man utvecklar C# i Visual Studio?
2. Vilken typ av filer bör man absolut inte lägga i .gitignore?

3. Skapa ett nytt projekt i Visual Studio - välj att VS skall använda git för versionshantering samt att VS skall skapa en .gitignore-fil. Inspektera .gitignore filen som skapas. *Ser ni några överraskningar? Stämmer den bra med svaren i fråga ett och två?*

Övning 5

Working tree

Fil_A

.gitignore

out.xml

out.dll

Staging area (Index)

Fil_A

.gitignore

History

.gitignore

```
*.dll
```

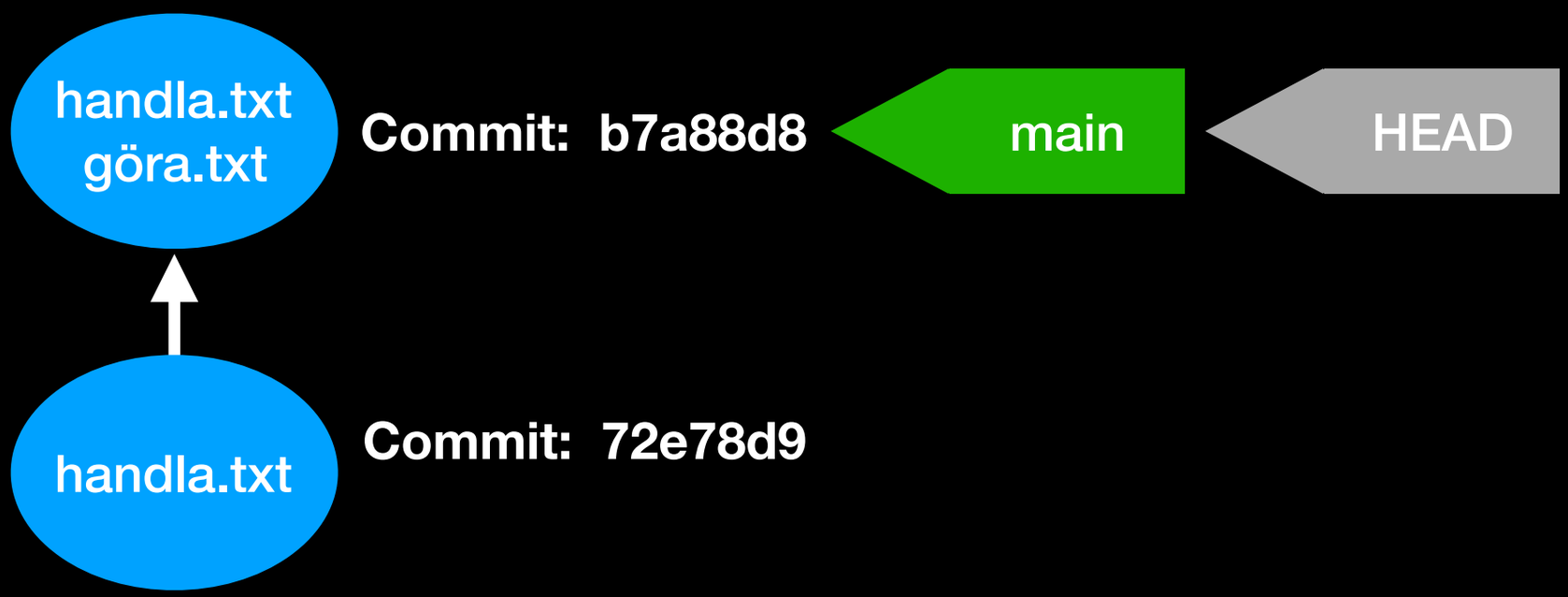
Vad anser git statusen vara för vårt git-repo i scenariot ovan?

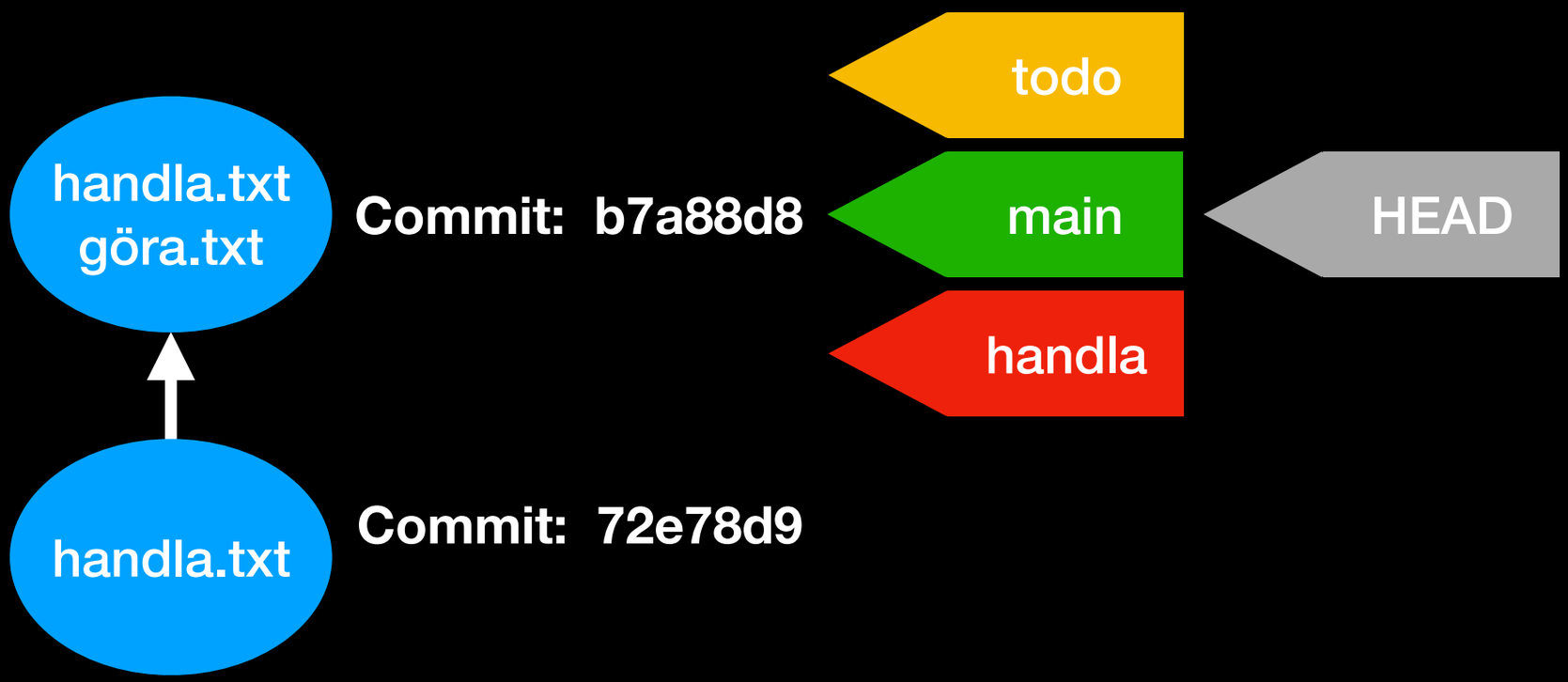
Branch

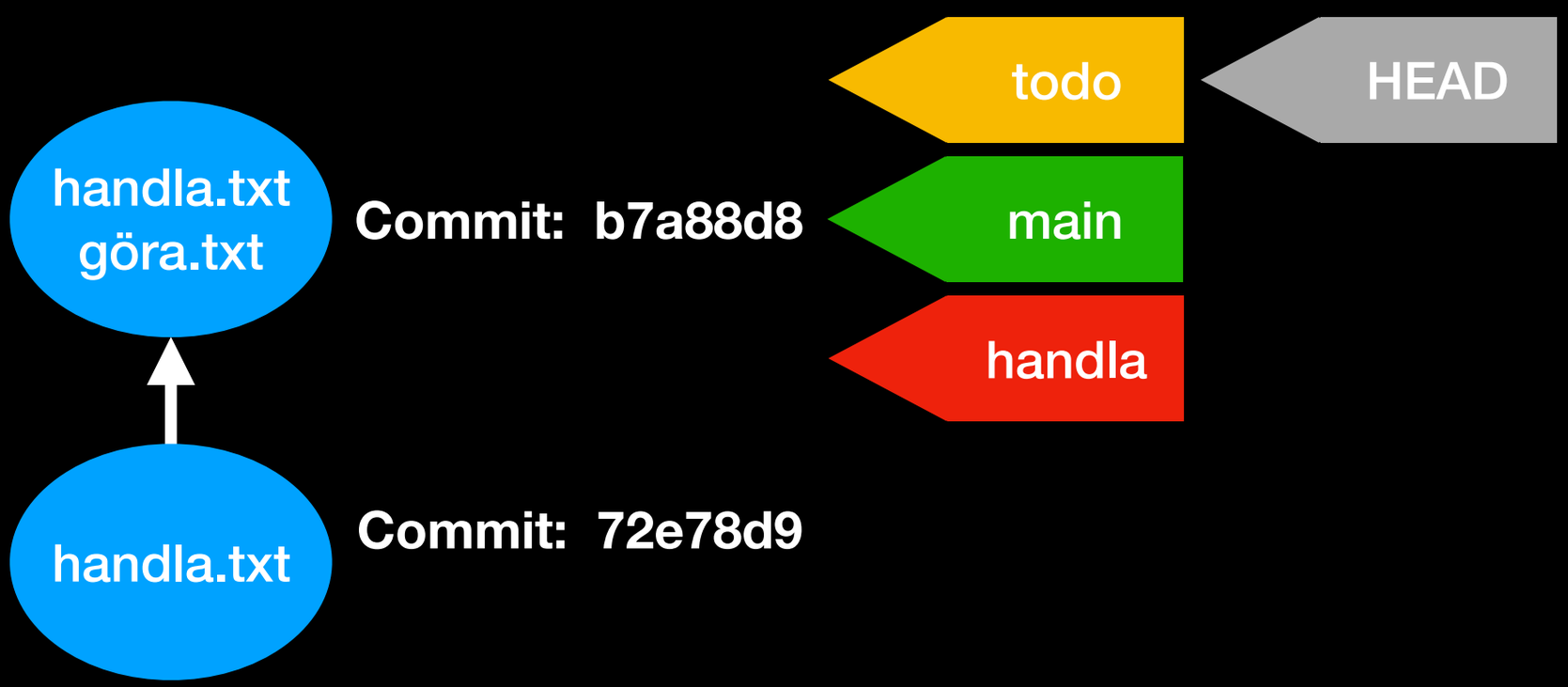
- Jobba på flera funktioner parallellt
- En branch är egentligen bara en namn på en SHA-summa
- Flyttas med en commit
- Dela med sig
- main (master)
- HEAD

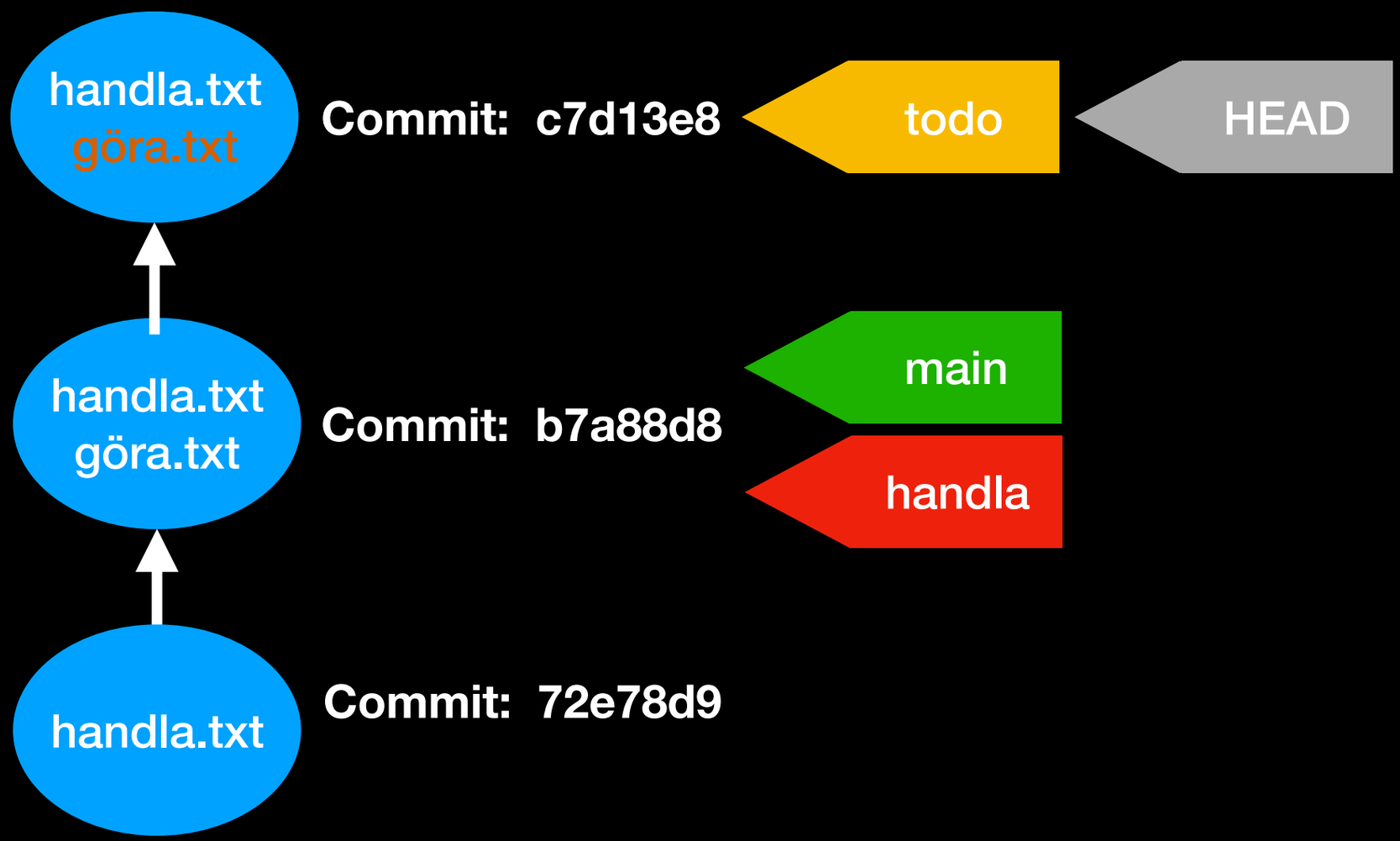
Branch - kommando

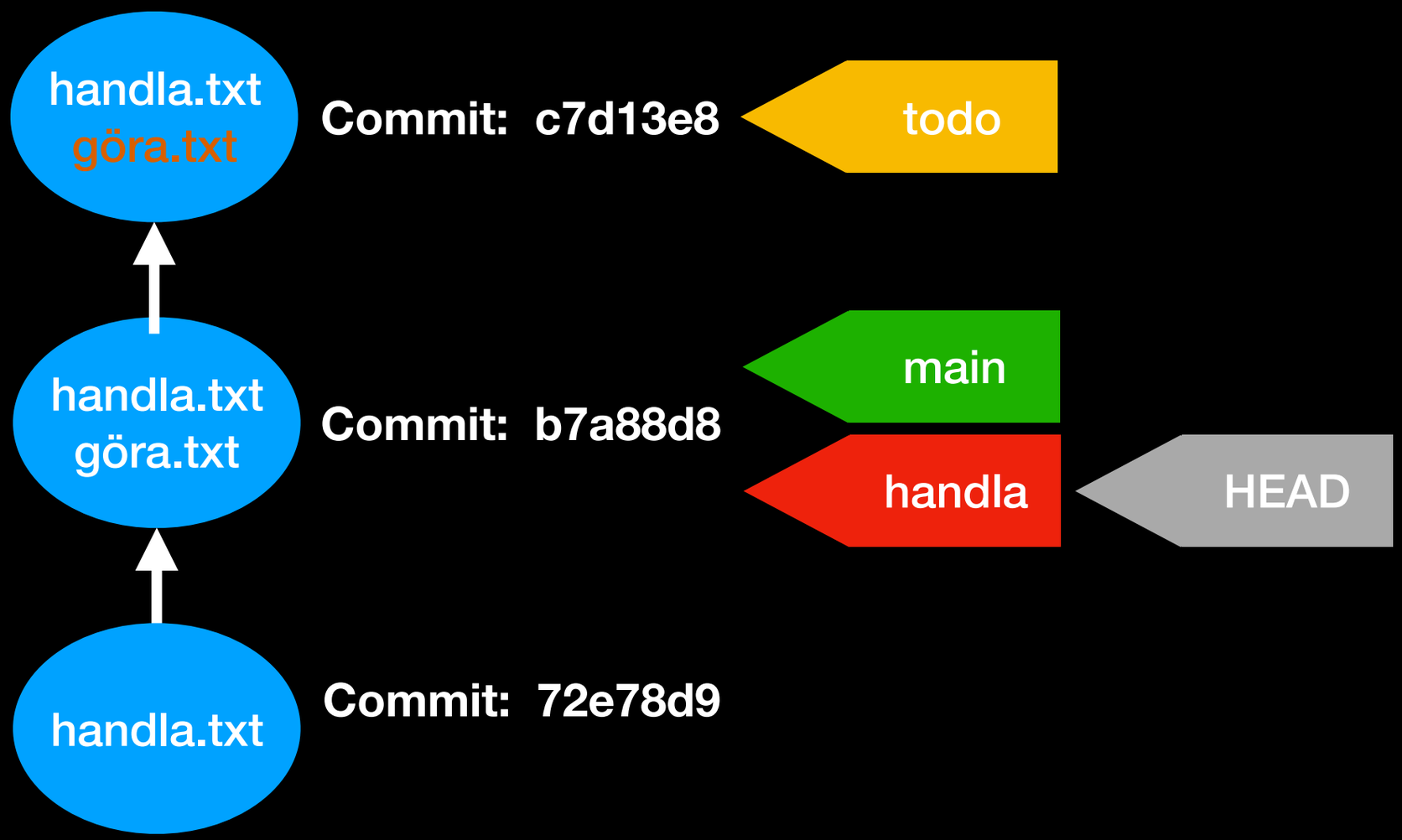
- git branch - list
- git branch feature - skapa branch (från HEAD)
- git checkout [branch] - checka ut
- git config --global alias.slog 'log --all --decorate --oneline --graph’

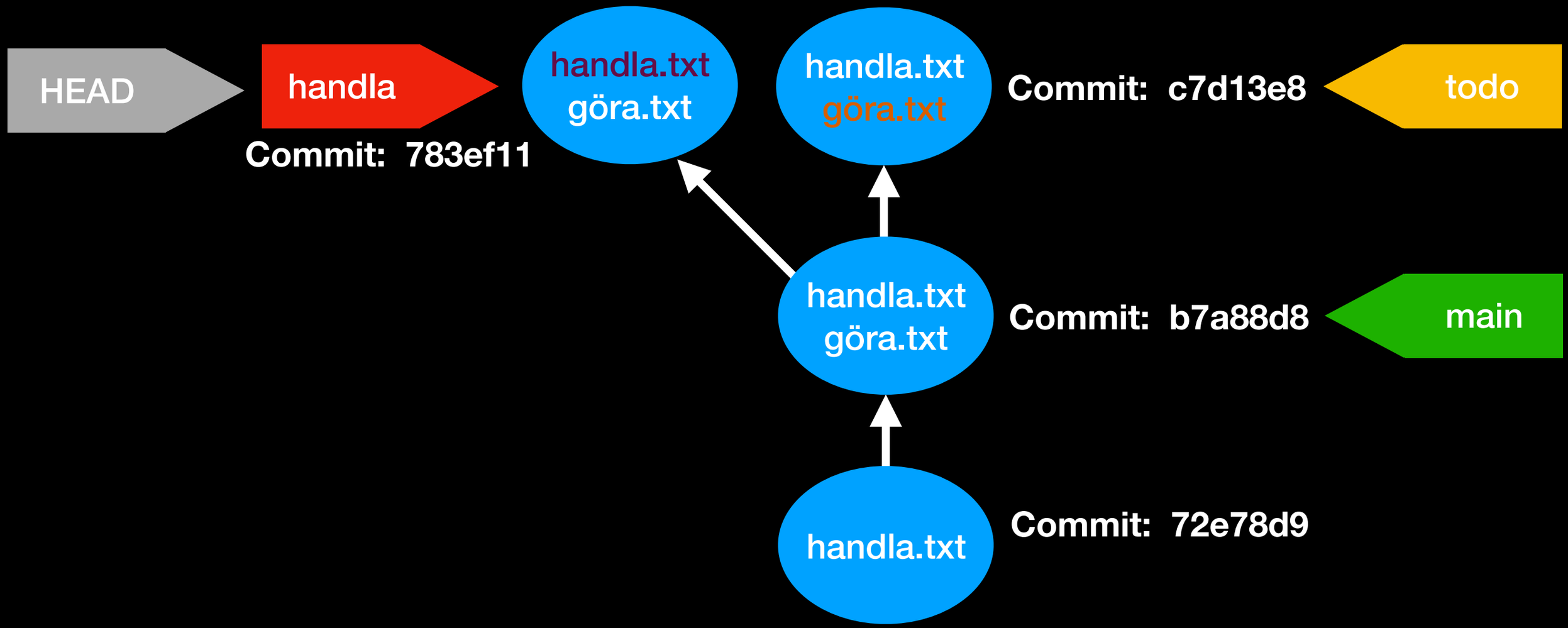

handla.txt
göra.txt
Commit: b7a88d8
main
HEAD
handla.txt
Commit: 72e78d9


handla.txt
göra.txt
Commit: b7a88d8
todo
main
HEAD
handla
handla.txt
Commit: 72e78d9


handla.txt
göra.txt
Commit: b7a88d8
handla.txt
Commit: 72e78d9
todo
HEAD
main
handla


handla.txt
göra.txt
Commit: c7d13e8
todo
HEAD
handla.txt
göra.txt
Commit: b7a88d8
main
handla
handla.txt
Commit: 72e78d9


handla.txt
göra.txt
Commit: c7d13e8
todo
handla.txt
göra.txt
Commit: b7a88d8
main
handla
HEAD
handla.txt
Commit: 72e78d9


HEAD
handla
Commit: 783ef11
handla.txt
göra.txt
handla.txt
göra.txt
Commit: c7d13e8
todo
handla.txt
göra.txt
Commit: b7a88d8
main
handla.txt
Commit: 72e78d9

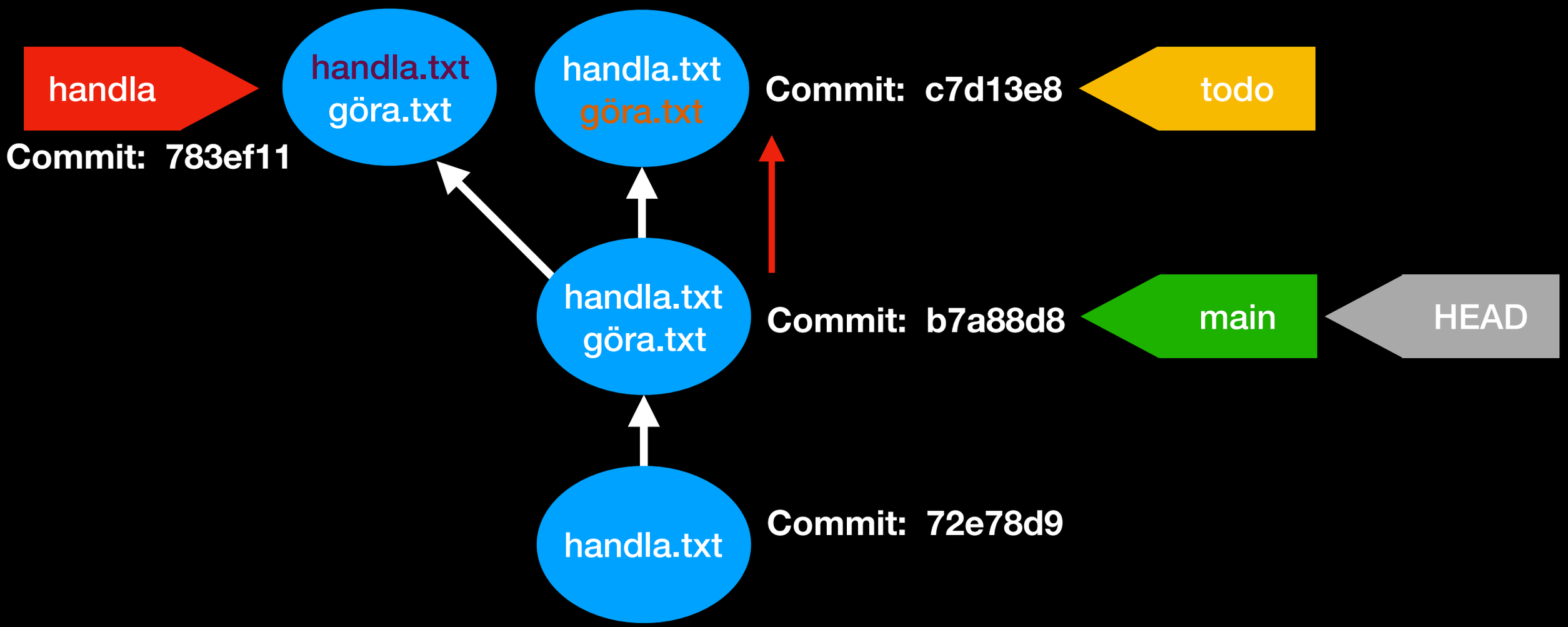
Vi skapar brancher!

Interaktiv del 5

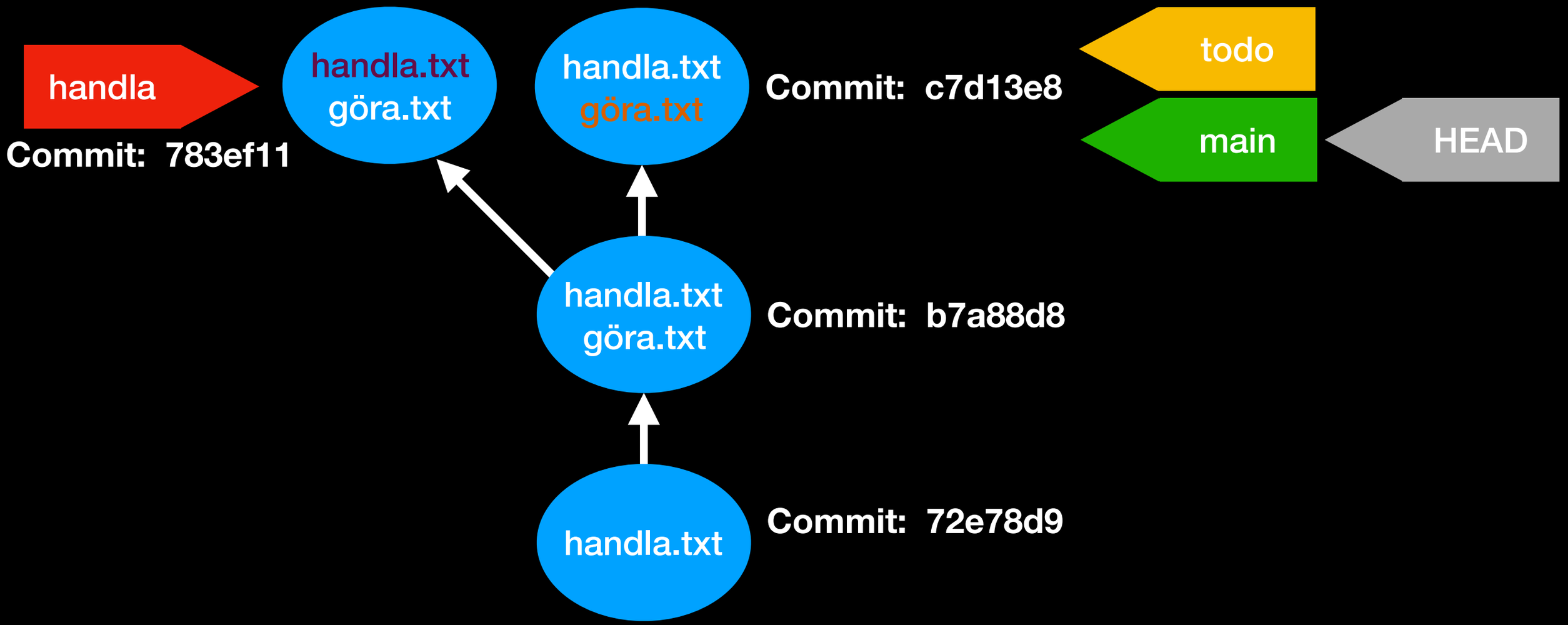
Merge

- Slå ihop förändringar
- Två typer av merge:
  - Fast-forward merge
  - 3-way-merge - merge-commit
- git branch --merged

Fast forward



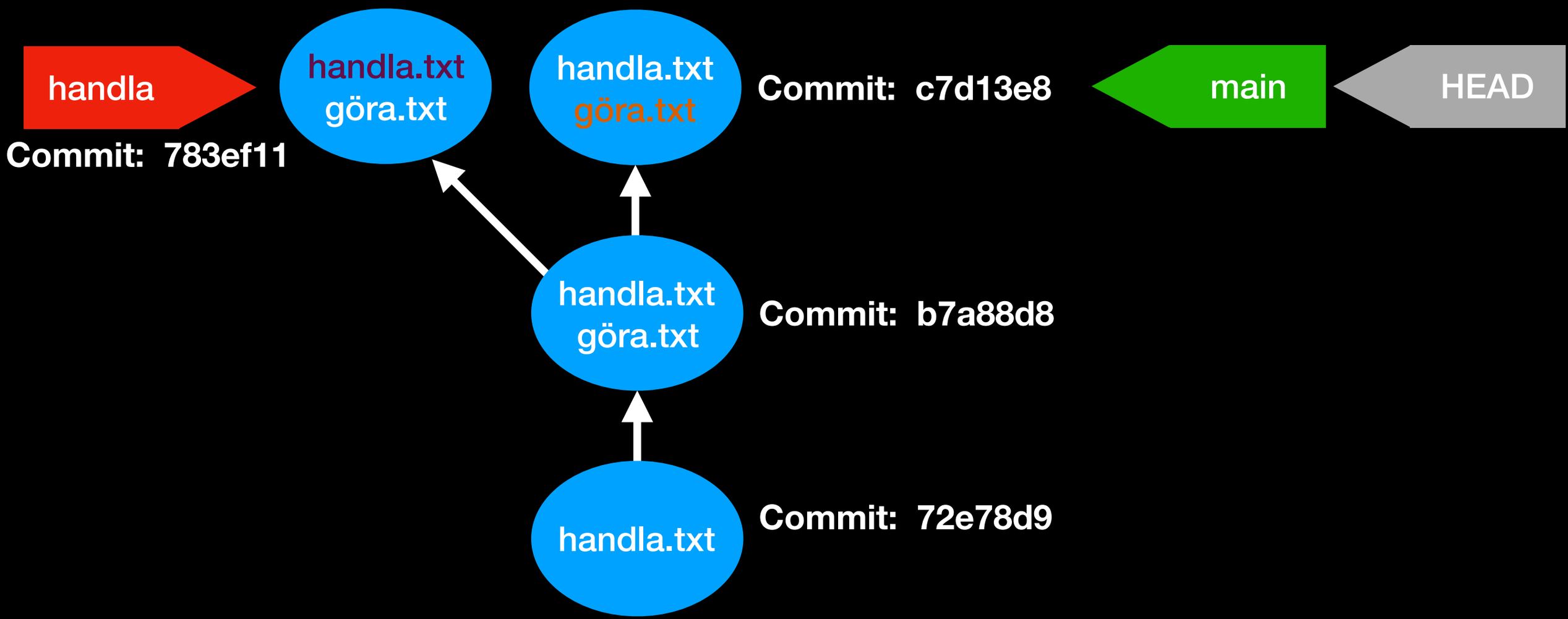
Fast forward



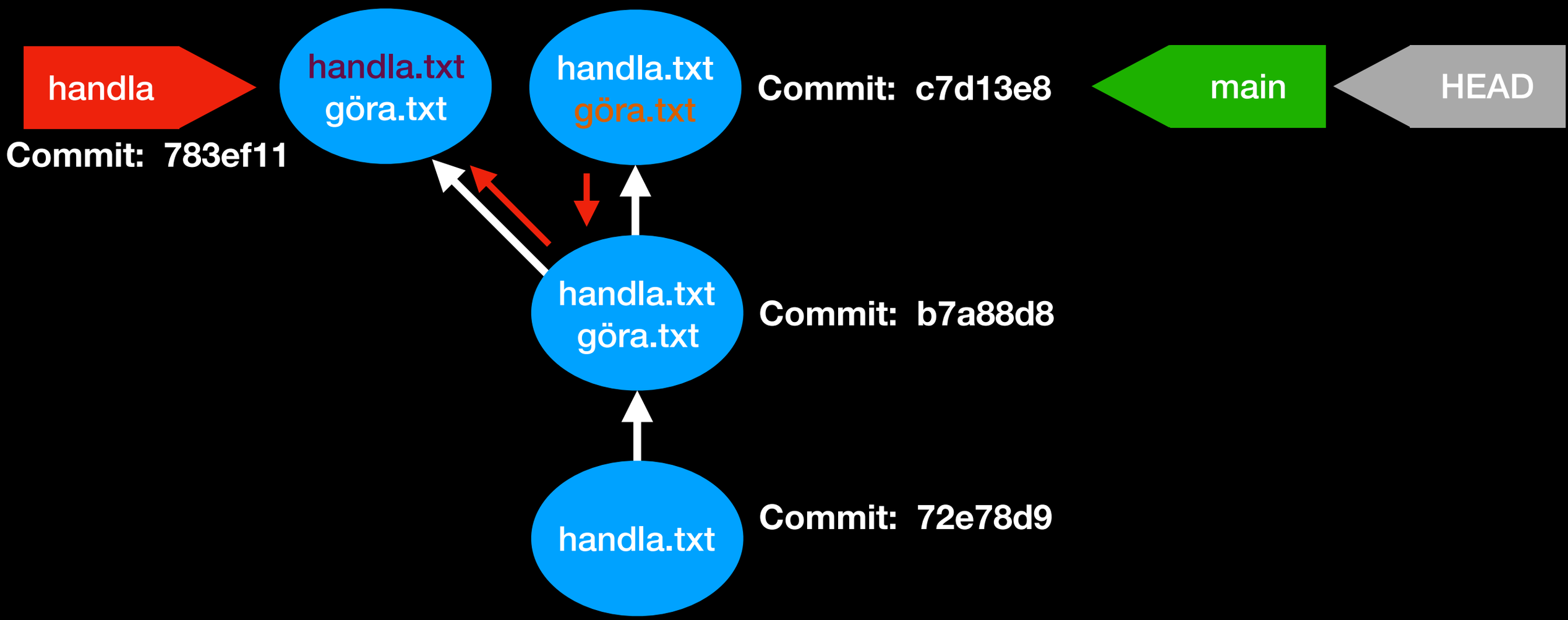
Vi mergar ‘todo’!

Interaktiv del 6

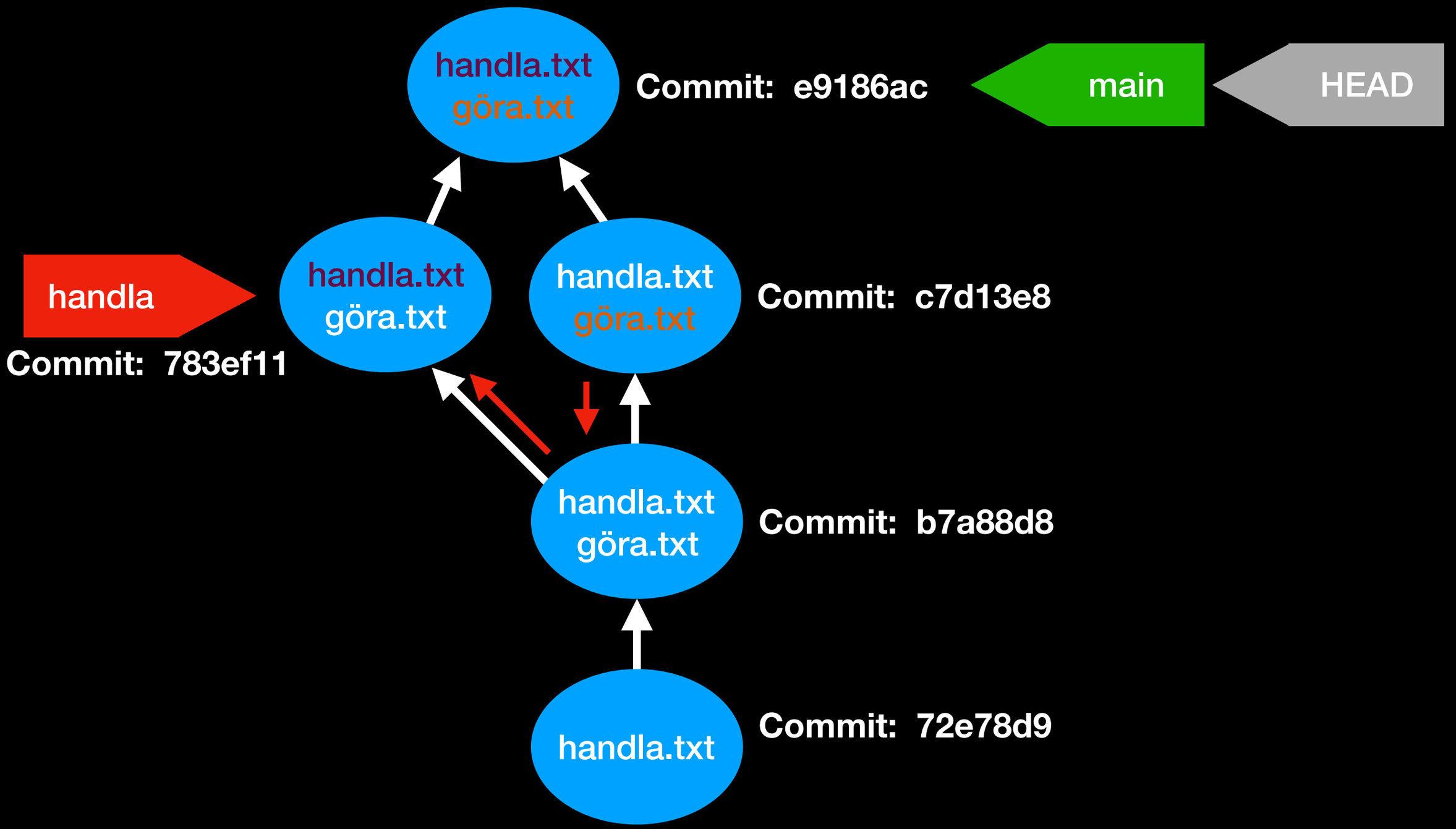
3-way-merge


handla
Commit: 783ef11
handla.txt
göra.txt
handla.txt
göra.txt
Commit: c7d13e8
main
HEAD
handla.txt
göra.txt
Commit: b7a88d8
handla.txt
Commit: 72e78d9

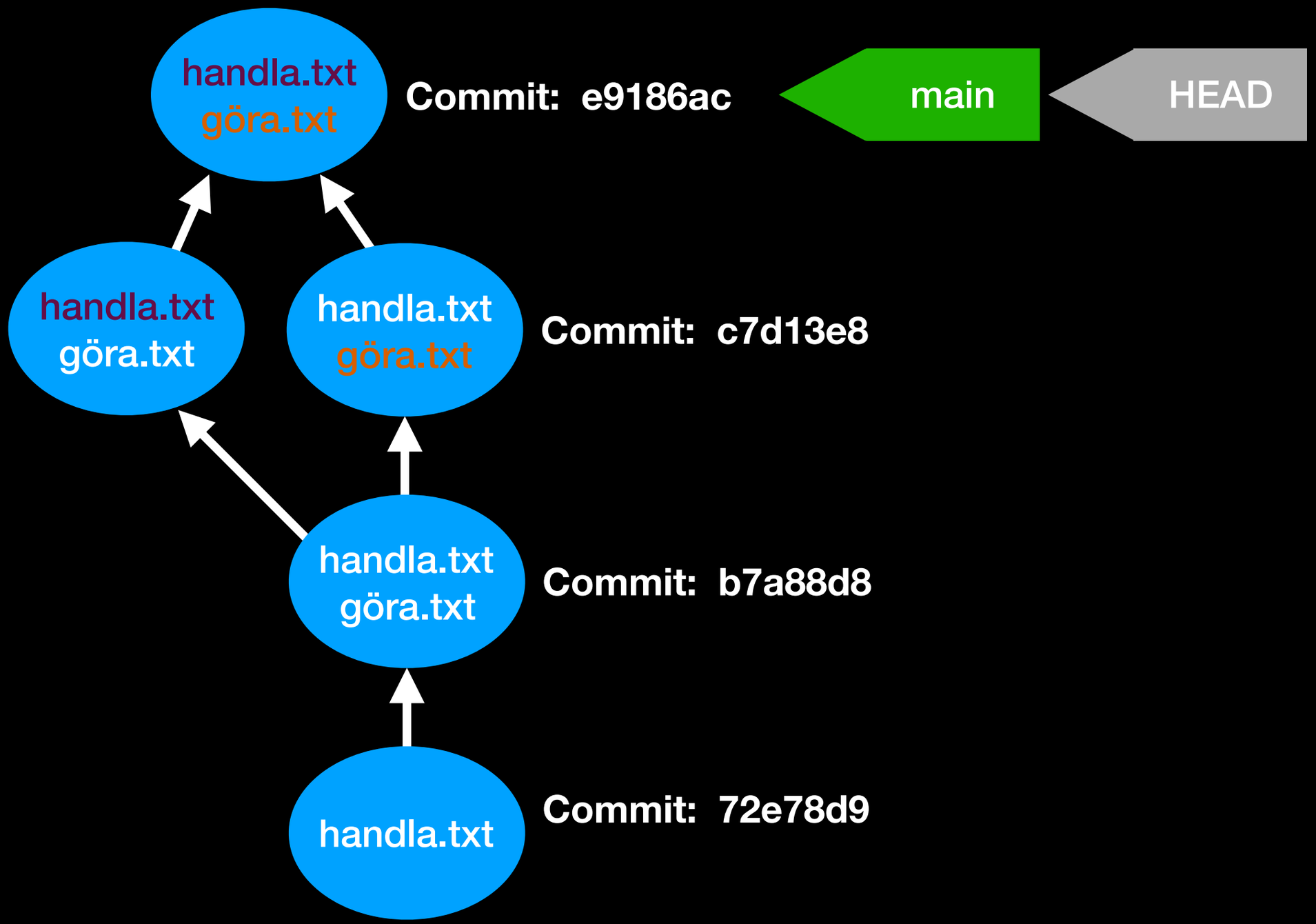
3-way-merge


handla
Commit: 783ef11
handla.txt
göra.txt
handla.txt
göra.txt
Commit: c7d13e8
main
HEAD
handla.txt
göra.txt
Commit: b7a88d8
handla.txt
Commit: 72e78d9

3-way-merge


handla.txt
göra.txt
Commit: e9186ac
main
HEAD
handla
Commit: 783ef11
handla.txt
göra.txt
handla.txt
göra.txt
Commit: c7d13e8
handla.txt
göra.txt
Commit: b7a88d8
handla.txt
Commit: 72e78d9

3-way-merge


handla.txt
göra.txt
Commit: e9186ac
main
HEAD
handla.txt
göra.txt
handla.txt
göra.txt
Commit: c7d13e8
handla.txt
göra.txt
Commit: b7a88d8
handla.txt
Commit: 72e78d9

Vi mergar ‘handla’!

Interaktiv del 7

Tag

- Som branch - men statisk
- git tag [tag-namn]
- git tag

Övning 6

SSH-nyckel

1. Skapa ett ssh-nyckelpar - ssh-keygen i “git bash”
2. Lägg in din ***publika*** SSH-nyckel (*id_rsa.pub*) på github
3. Testa ‘ssh -T git@github.com' skall säga *Hi UserX! You've successfully authenticated, but GitHub does not provide shell access.*

Navigera via:

User > Settings > SSH and GPG keys > New SSH key

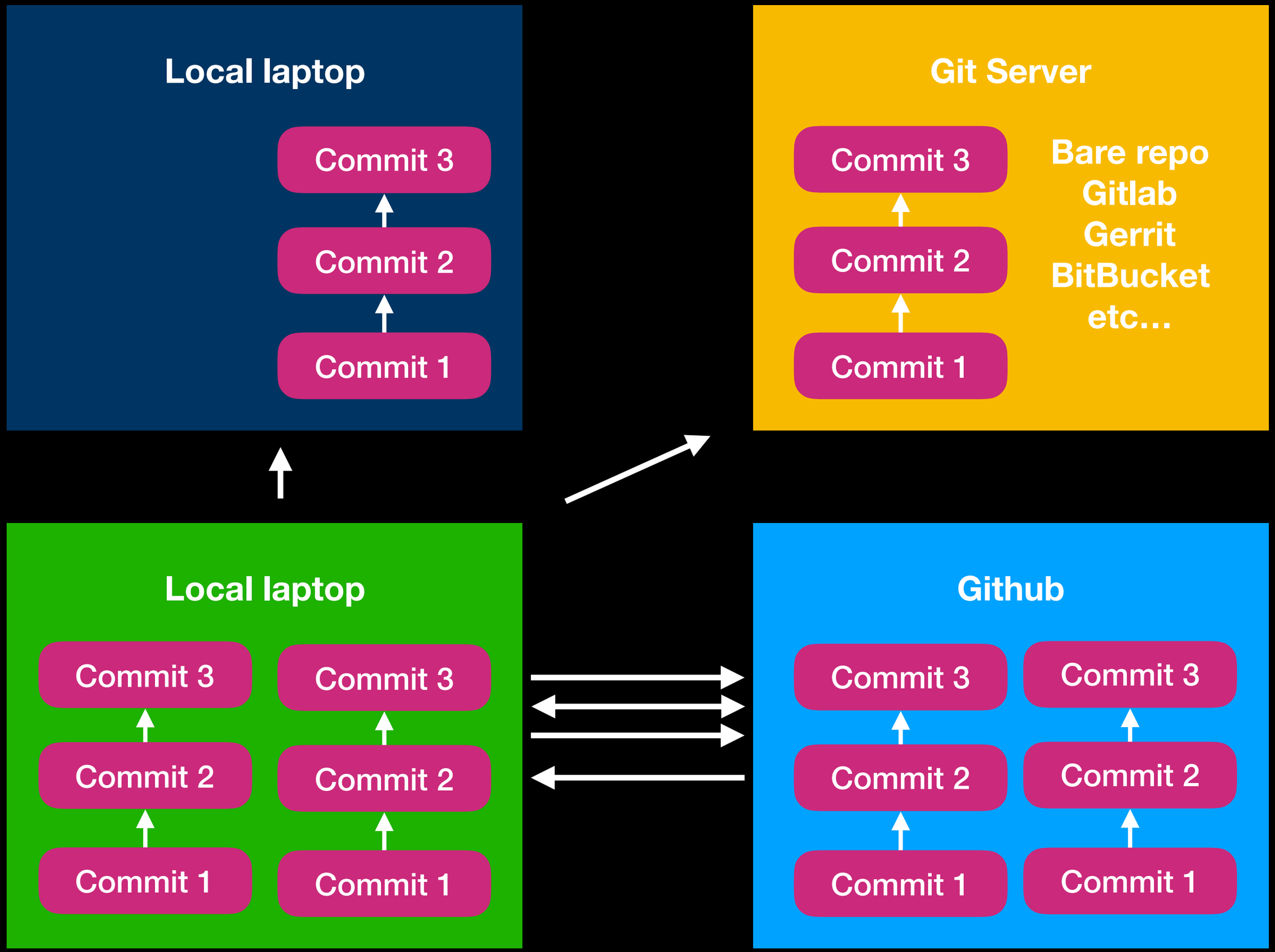
Eller surfa in direkt på:

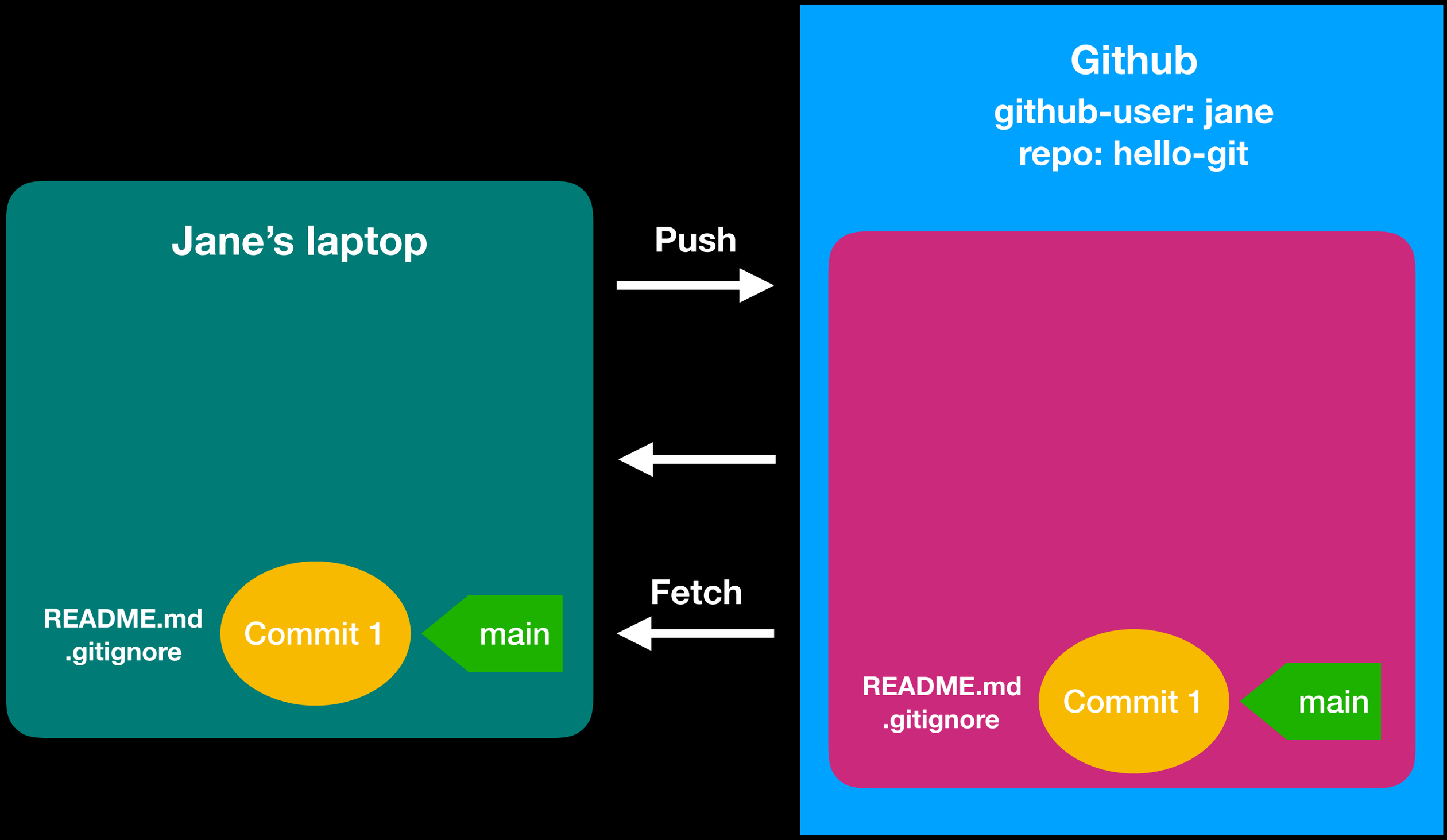
https://github.com/settings/keys

https://support.atlassian.com/bitbucket-cloud/docs/set-up-an-ssh-key/#SetupanSSHkey-ssh3

Remotes

- Ett repo “någon annanstans”
- Som en genväg till en URL: git@github.com:user/repo.git
- Github, Gitlab, BitBucket eller liknande
- Public / Private


Local laptop
Commit 3
Commit 2
Commit 1
Git Server
Commit 3
Commit 2
Commit 1
Bare repo
Gitlab
Gerrit
BitBucket
etc…
Local laptop
Commit 3
Commit 2
Commit 1
Commit 3
Commit 2
Commit 1
Github
Commit 3
Commit 2
Commit 1
Commit 3
Commit 2
Commit 1



git clone git@github.com:jane/hello-git.git

Kort sammanfattning

- Clone - skapa lokalt repo, med remote ‘origin’
- Push - skicka upp ändringar till remote
- Fetch - hämta hem ändringar från remote
- (Merge) - slå ihopa två brancher
- Pull - fetch + merge

Övning 7

Remote

1. Skapa ett repo på github.com - namnge det ‘hello-git’
2. Skapa med ‘README.md’ + .gitignore för VS
3. Klona ned projektet till er dator
4. Modifiera README.md på github och commita ändringen
5. Hämta hem förändringen från github (fetch)
6. Slå ihop förändringen från github’s main-branch med den lokala main-branchen (merge)

Övning 8

History/blame

1. Klona repot: git@github.com:everlof/treasure.git

2. Vid ett tillfälle glömdes en hemlig produktionsnyckel i koden. Vad är produktionsnyckel?

3. Klona repot: git@github.com:everlof/pet-store.git

4. Vem var det som döpte fågeln till ‘Gene’? Försök att lösa denna enbart genom att använda Visual Studio.

Övning 9

Jobba i par tillsammans mot en remote på GitHub

1. Gå ihop i grupper om 2
2. En person skapar ett repo och ger skriv access till den andra gruppmedlemmen
3. Simulera att ni jobba på en kodbas och hämta varandras ändringar och pusha upp nya
4. Försök att framkalla en konflikt (i git) och lös den

Övning 10

Jobba i par tillsammans via två remote på GitHub (fork + pull request)

1. Gå ihop i grupper om 2
2. En person skapar ett repo
3. Andra personen fork-ar repot den första personen skapat
4. Jobba i på andra personens repo och pusha upp arbetet till forken
5. Skapa en ‘Pull Requests’ med arbetet från fork’en till orginal-repot

Övning 11

Konflikt

1. I vilka scenarier kan det tänkas bli en konflikt när man mergar?

2. Sätt upp ett scenario där en conflict inträffar och försök lös konflikten. Lättast är nog att göra det med två brancher lokalt, men ni kan också prova göra det genom att modifiera koden på GitHub direkt.

Övning 12

Ändra i en commit

Tänk på att när en commit ändras , så får den alltid ett nytt “id”

1. Hur ändrar man commit-meddelande på den senaste commit man gjorde?

2. Om man ändrar commit-meddelandet på en commit som man redan har pushat upp till en remote kan man få problem - varför?

3. Modifiera innehållet i en commit och lägg till ändringen till “samma” commit

Övning 13

Ändra innehåll i en commit

Tänk på att när en commit ändras , så får den alltid ett nytt “id”

1. Läs på om “git reset” (`git help reset` eller https://git-scm.com/docs/git-reset). Fokusera på de fallen där man använder --hard och —soft.

2. Använd “git reset --hard” för att flytta tillbaka branch-pekaren till en tidigare commit. Tips: skapa en ny branch för detta så att `master` stannar kvar på den senaste commiten.

3. Använd “git reset --soft” för att gå tillbaka till en tidigare commit, men behåller ert working-tree från nuvarande commit.

https://git-scm.com/book/en/v2/Git-Tools-Reset-Demystified

Övning 14

Fördjupning

1. Vad betyder det när man är i “detatched head”-state?

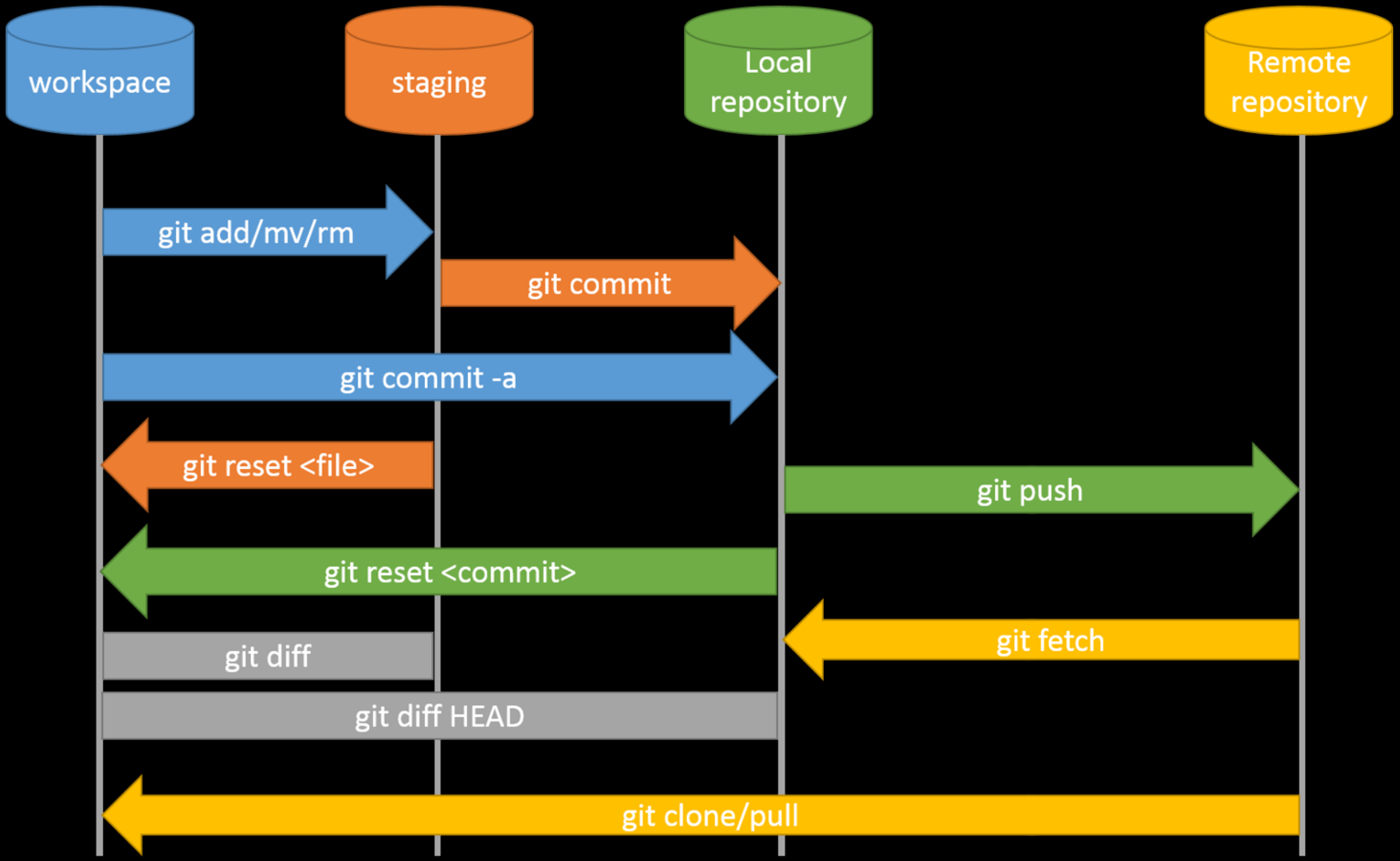
2. Vad är kan man göra för att ta sig ur “detatched head”-state?

3. Varför kan det anses vara “farligt” att vara i “detatched head”-state?

Länkar

- https://github.com/everlof/git-course
- Cheatsheet: https://training.github.com/downloads/github-git-cheat-sheet/
- Bra intro git-graph, working tree, staging area & history: https://www.youtube.com/watch?v=uR6G2v_WsRA
- Brancher & merge: https://www.youtube.com/watch?v=FyAAIHHClqI
- Remotes: https://www.youtube.com/watch?v=Gg4bLk8cGNo
- Pro Git (Bok): https://git-scm.com/book/en/v2
- http://git-school.github.io/visualizing-git/

Fågelperspektiv



Källa: https://alinex.gitlab.io/env/git/