git introduktion
Del 2 av 4

https://github.com/Rnqst

Roland Ronquist

git-kursen schema

- 9:00 – 9:45
- Rast
- 10:00 – 10:45
- Rast
- 11:00 – 12:00

git SCM – Source Code Management

- Versionshantering / revisionshantering
- Håller reda på hur koden utvecklas över tiden
- Vad levererades till vem? Närdå?
- Sparar plats genom att bara lagra förändringar

Vad gör ett versionshanteringssystem?

Termer

- Master, main, trunk, stam, rot
- Gren, branch, fork….
- Revision, version

Bilden på föregående sida från:

https://git-scm.com/about

Varför är git så bra?

- Varje användare har en helt egen kopia av koden
- Alla kopior kan innehålla alla versioner
- Alla kopior kan vara jämlika, en server är inte nödvändig

Missa inte boken

- Läs boken Pro Git som du antingen kan beställa som en tryckt bok eller ladda hem
- https://git-scm.com/book/en/v2
- Den finns översatt till 30-talet språk
- Dock är inte alla översättningar klara

Var är vi nu?

- Alla har konton på github
- Alla har GitHub Desktop

Git från kommandoraden

Den som behöver styra git mer precist använder ofta git från kommandoraden.

Om du vill kan du installera från:

https://git-scm.com/downloads

Detta är inte nödvändigt för kursen här.

Frågorna från igår

Tänk på följande frågor:

Din gamla dator gick sönder men du hade checkat in all din kod på github innan. Har du då tappat information när du tittar på koden i din nya dator efter att ha checkat ut den där igen?

Frågorna från igår

Om vi fortsätter frågan med att även github kraschar efter att du lyckats hämta ut koden till din nya dator. Vad finns det för möjlighet att då kopiera din kod till dina kollegor om github ligger nere?

Om det går att kopiera till kollegorna, blir deras kopior då lika kompletta som om de tankat sakerna direkt från github?

Frågorna från igår

- Varför lägger så många upp sina program på github?
- Är det inte bättre att sälja programmen och bli rik?

Ett litet projekt

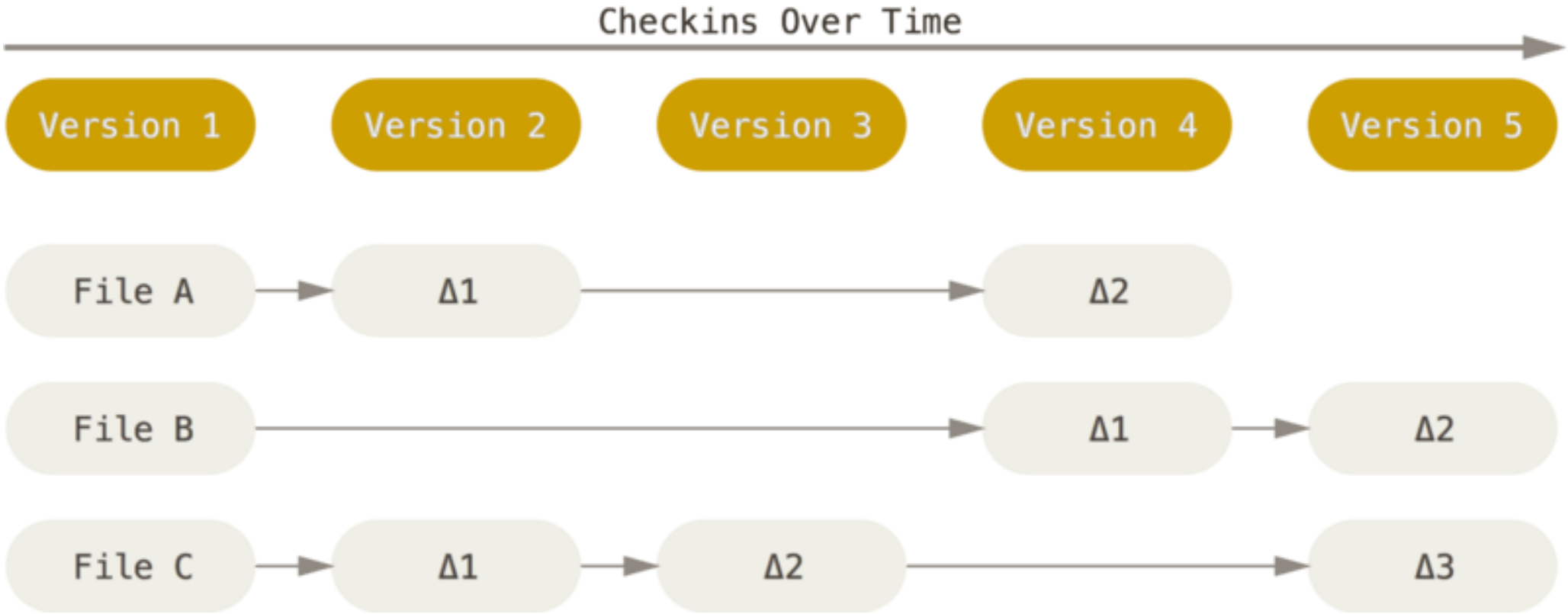

Bild från boken Pro Git

Ett litet projekt

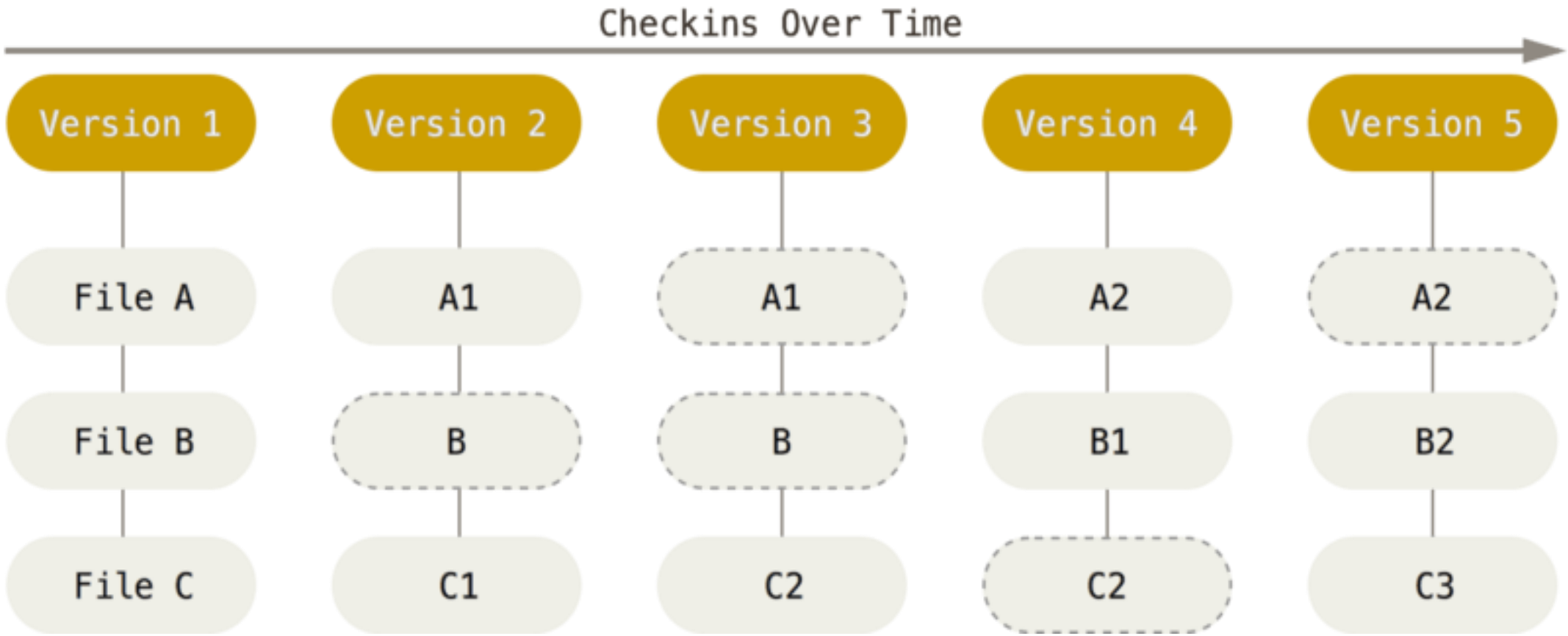

Bild från Pro Git

Ett litet projekt

Termer:

- Sparad eller commited
- Ändrad eller modified
- Preparerad eller staged

Ett litet projekt

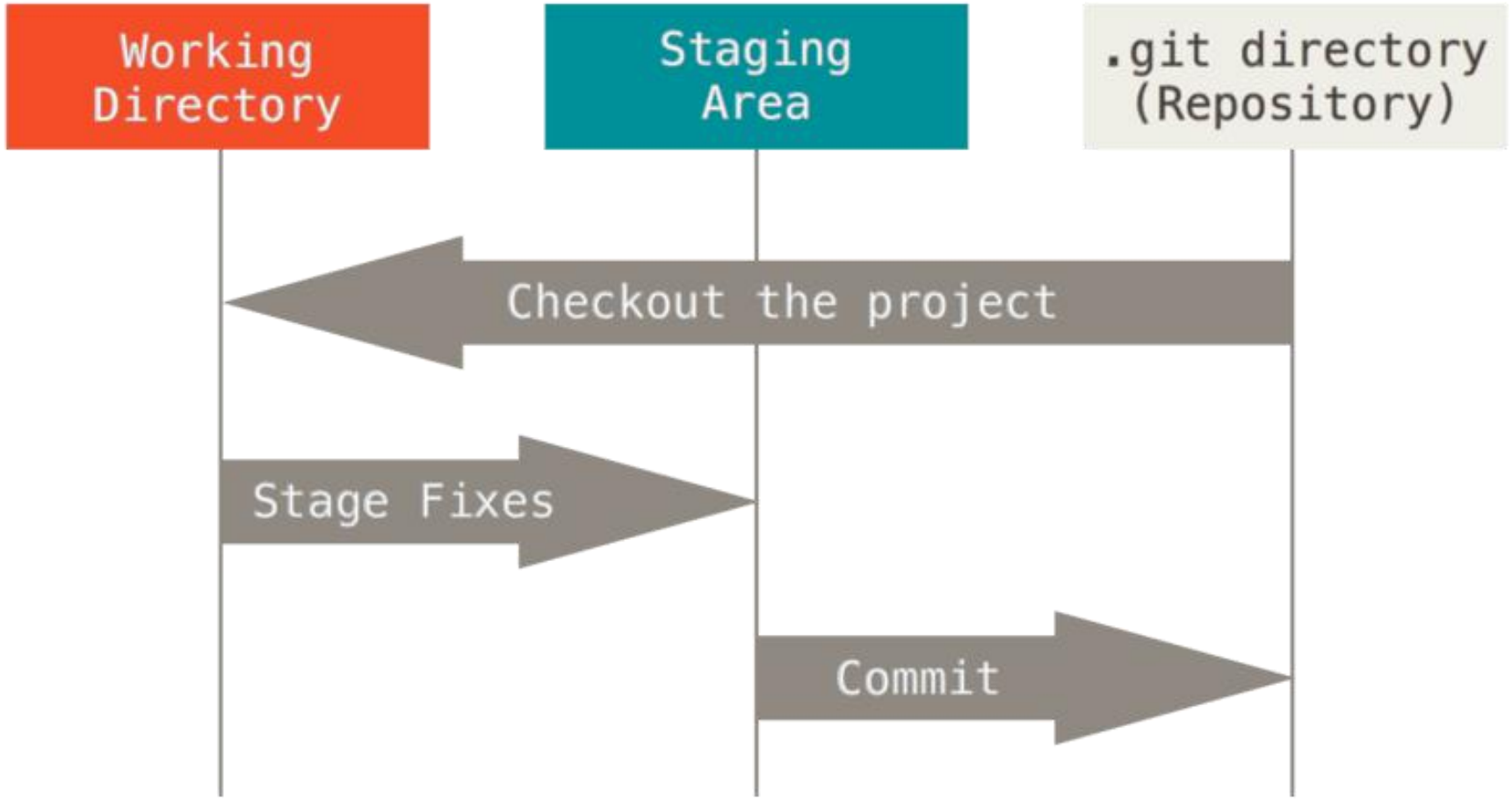

Bild från Pro Git

Ett litet projekt

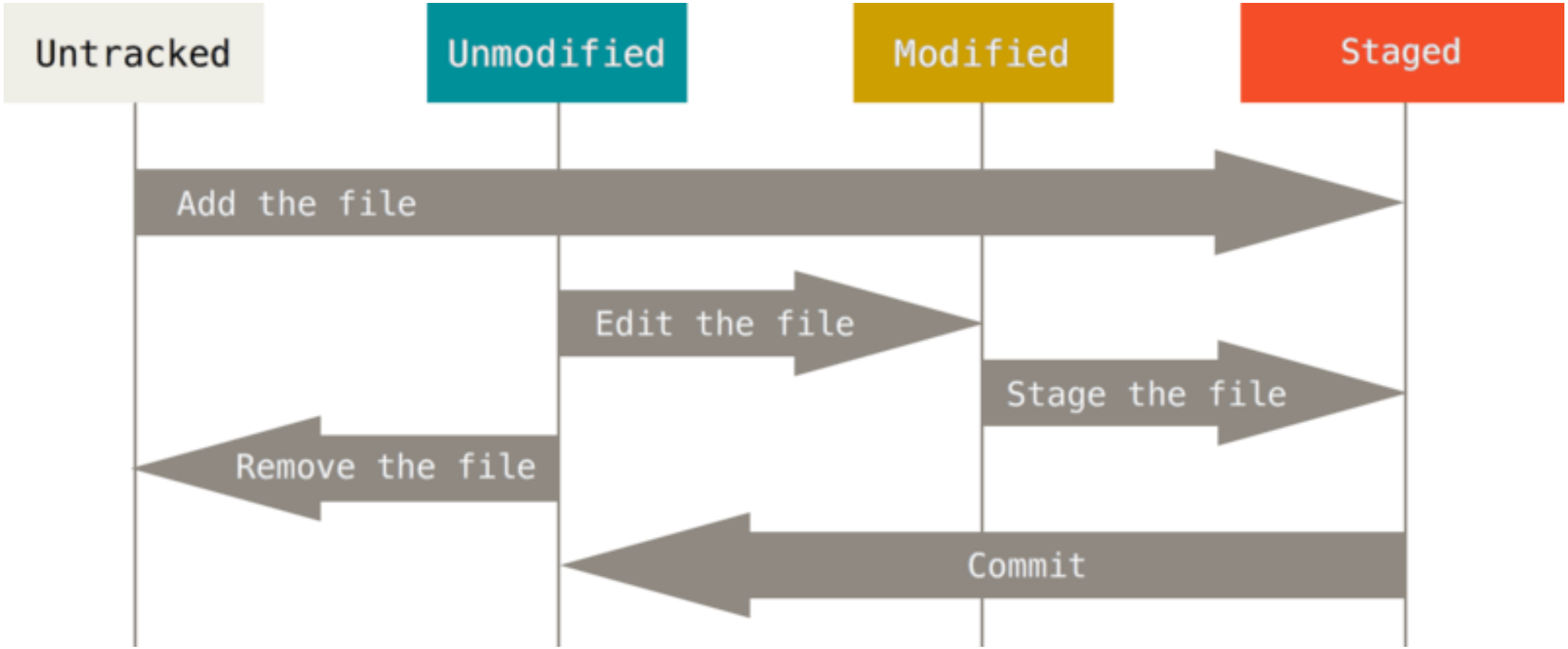

Bild från Pro Git

Att utestänga filer

`.gitignore`

Genom att lägga till filer och filändelser i den filen så hamnar dessa filer inte i dit git-arkiv

Att skapa en fil vars namn börjar med en punkt . kan kräva lite extraarbete i windows. I linux betyder en punkt . i början på namnet att filen är dold.

Att utestänga filer

Exempel från Pro Git:

```
# ignorera alla .a filer
*.a

# men spåra lib.a, även om du ignorerar .a filer
enligt regeln ovan
!lib.a

# ignorera enbart TODO-filen i den aktuella katalogen
/TODO
```

Att utestänga filer

Exempel från Pro Git:

```
# ignorera alla filer i kataloger som heter build
build/

# ignorera doc/notes.txt, men inte i
# doc/server/arch.txt
doc/*.txt

# ignorera alla .pdf-filer i doc-katalogen och någon
# av dess underkataloger
doc/**/*.pdf
```

Utforska ditt repo

`git status`

Visar läget just nu

`git log`

Visar utvecklingen över tiden

(fördjupning: `git diff…`)

Hur gör du i GitHub Desktop?

Lägg till en fil

I GitHub Desktop är det bara att placera en ny fil i en katalog som ingår i ditt repo så läggs den till automatiskt (varning: `.gitignore`).

git add <filnamn>

Lägger till en fil manuellt

Hur skapar vi ett nytt repo?

I GitHub Desktop kan du antingen köra `CTRL-N` eller använda menyerna: **File > New Repository**

Med kommandon kör du:

```
git init
```

Då skapas de filer som git behövs för att hålla rätt på filerna i den katalog du står i och underliggande filträd.

Knyta ihop påsen, göra klart

`git commit`

När du arbetat färdigt med filerna så körs commit för att lägga till dina uppdateringar (från prepareringsytan, staging area, till lokalt repo).

`git push`

Laddar sedan upp ditt repo till en server.

GitHub Desktop, hur gör du där?

Övningar

Vi jobbar i desktop-tutorial

- Skapa en fil, **MinFil.txt**, skriv in minst fem rader text.
- Skriv in en förklarande kommentar
- Kör **commit**

Övningar

Vi jobbar i desktop-tutorial

- Ändra rad tre i filen **MinFil.txt**.
- Skriv in en förklarande kommentar
- Kör **commit**
- Ladda upp filerna med **push**
- Hur ser det ut i din webbläsare på **github.com**?

Övningar

Vi jobbar i desktop-tutorial

- Skapa en ny gren, döpt den till **laboration**
- Vad händer då? Finns filerna kvar?
- Lägg till en ny fil
- Lägg till två nya rader i mitten av filen **MinFil.txt**
- Kör commit och push, vettiga kommentarer…
- Vad har hänt på github nu?

Fördjupningsövning

Vi jobbar i desktop-tutorial, i grenen **laboration**

Du vill kopiera ändringarna i din gren till stammen (**main**).

Hur gör du det? Finns det flera sätt?

Läs **Pro Git**.

Fördjupningsövning

Det här är för dem som vill lära sig git-kommandon, har du svårt att hänga med i hur du använder GitHub Desktop, vänta med denna uppgiften till nästa lektion.

Hur gör du för att lägga upp ett nytt projekt i git?

Tänk dig att du redan har en katalog med några filer och utgå från den.

Vilka kommandon behöver du köra för att allt skall hamna i git lokalt på din maskin? Vad behöver du ytterligare göra för att allt även skall hamna på github?

Läs **Pro Git**.

Självstudier

Vad menas med att en fil är ”**ospårad**” eller ”**untracked**”?

Vad gör **git clean**?

Vad är en ”**prepareringsyta**” eller ”**stage area**”?

Hur kan du ställa in så att git inte bryr sig om vissa filer?

Självstudier

1. Jobba i desktop-tutorial med GitHub Desktop. Gör en gren, branch, lägg till ett par filer där. Hur för du då tillbaka dina nya filer till stammen, main?

2. Pröva att göra en gren, gå till main och editera och commita en fil där som även finns i stammen. Vad händer nu när du för tillbaka den nya grenen till stammen?

3. Gör som under två, men innan du för tillbaka grenen till stammen så går du till stammen och editerar stammens version av den filen du editerade i grenen, men ändra den på ett annat sätt än du gjorde i grenen. Gå sedan till grenen och försök föra tillbaks den till stammen, vad händer då?

Självstudier

Läs kapitel 5 om ”Distributed Git” i boken Pro Git.