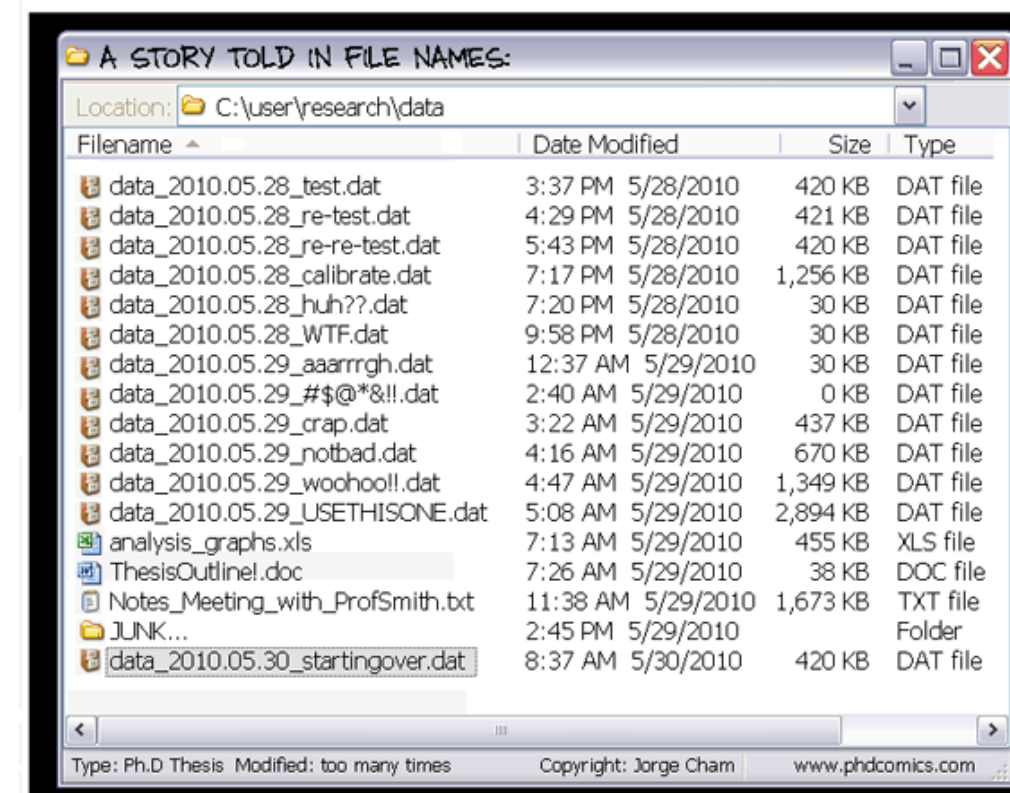
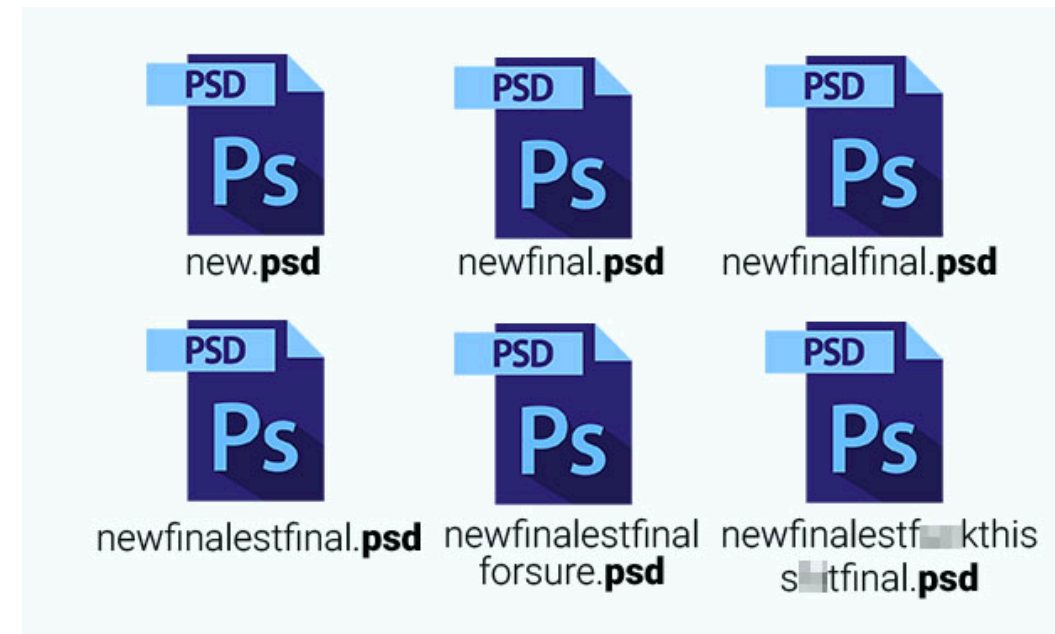


Varför versionshantering?

- Hur delar ni på filer i ett grupparbete nu? Dropbox? Server? Vad finns det för för- och nackdelar med det?
- Äganderätt - "Rör inte main.java just nu, jag är inne och ändrar i den."
- Kunna utveckla större ändringar parallellt
- Backup - Kunna återställa tidigare versioner
- Spåra ändringar

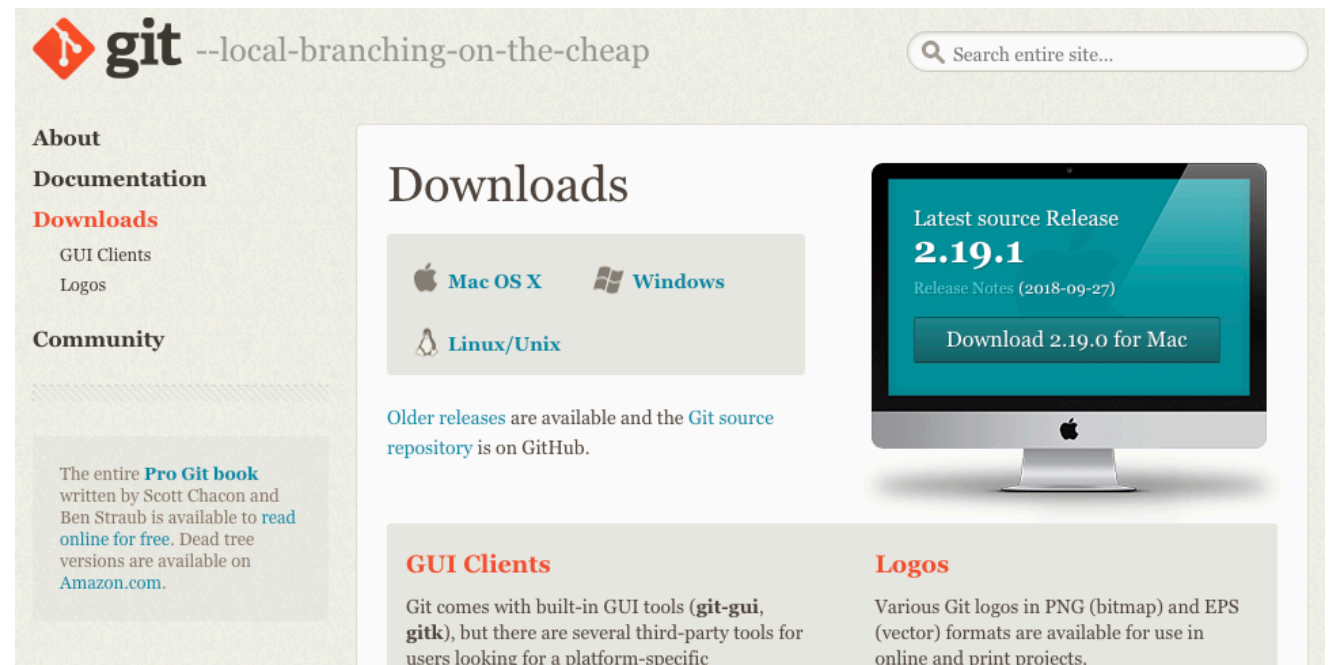


Varför GIT?

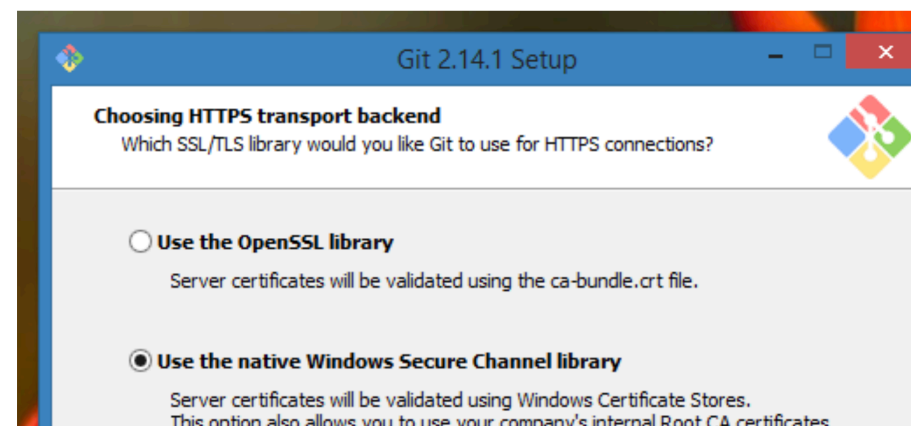
- Lätt att använda - när man kommer över inlärningskurvan
- Distribuerat - behövs ingen central server
- Extremt vanligt/efterfrågat

Getting started

- *Uppgift:* Installera GIT
<https://git-scm.com/downloads>
 - Alt win: <https://gitforwindows.org/>
 - <https://tutesfornewdevelopers.wordpress.com/2017/08/27/git-windows/>
 - Alt mac: <https://brew.sh>
`brew install git`
- I ett senare skede kommer vi att använda oss av tjänsten *github* och för att förenkla en del inför dagens övningar ska vi redan nu skaffa ett konto där.
- *Uppgift:* Skapa ett konto på <https://github.com>

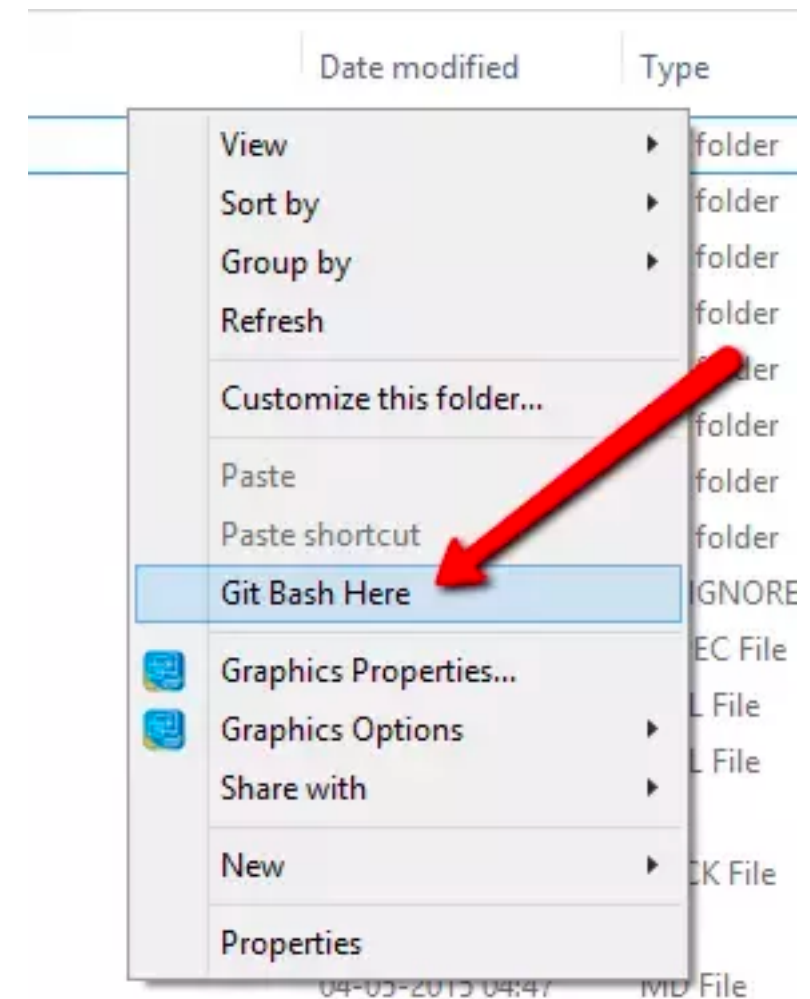


For **Choosing HTTPS transport backend**, choose **Use the native Windows Secure Channel Library**. This is especially helpful if you are a new student or developer and may not know how SSH works. Let's stay out of the complexity of OpenSSL for git.



Getting started

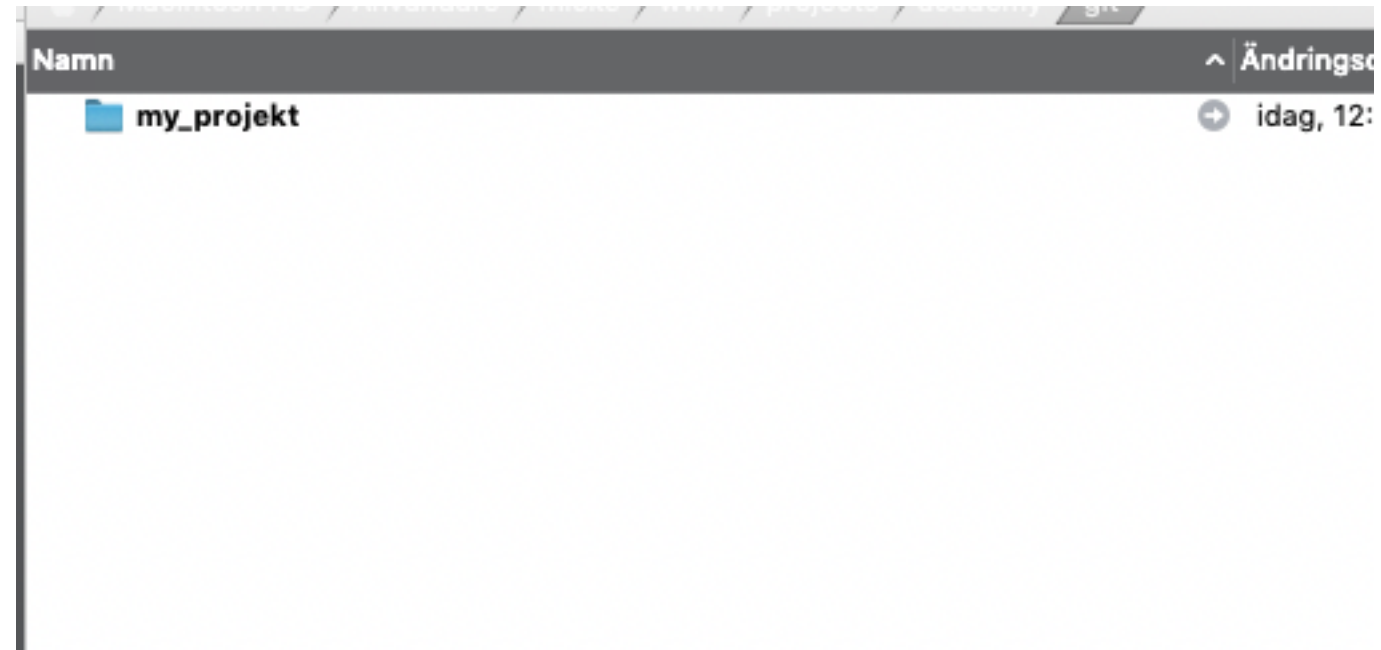
- Lättaste sättet att använda git i terminal-läge i Windows är att navigera till den mapp man vill använda för sina projekt och använda `git bash` därifrån genom att högerklicka.
- För mac navigerar man till rätt mapp med terminalen.
- Bra terminalprogram:
<https://www.terminal2.com/>



<https://gitforwindows.org/>

Skapa ditt första repo

- *Uppgift:* Skapa en mapp för ditt projekt.
- Undvik mellanslag och ”konstiga” tecken.
- Gå till mappen i terminalen (*git bash*) och kör:
`git init`



```
1. micke@Mikaels-MacBook-Pro: ~/www/projects/academy/git/my_projekt (zsh)
Last login: Thu Oct 18 12:15:45 on ttys004
➔ ~ cd /Users/micke/www/projects/academy/git/my_projekt
➔ my_projekt git init
Initialized empty Git repository in /Users/micke/www/projects/academy/git/my_projekt/.git/
➔ my_projekt git:(master) |
```

Vad innehåller repot?

- I git bash (liksom i linux/mac) listar man filer med kommandot

`ls`

- Man kan ange parametrar till `ls`, såsom `l` (lång listning/detaljer) och `a` (alla filer, även dolda).

`ls -la`

```
1. micke@Mikaels-MacBook-Pro: ~/www/pr
➔ my_projekt git:(master) ✗ ls
README.md      my_projekt.iml test.txt
➔ my_projekt git:(master) ✗ ls -l
total 16
-rw-r--r--  1 micke  staff   39 Oct 19 01:51 README.md
-rw-r--r--  1 micke  staff  335 Oct 19 01:39 my_projekt.iml
-rw-r--r--  1 micke  staff   0 Oct 18 22:03 test.txt
➔ my_projekt git:(master) ✗ ls -la
total 16
drwxr-xr-x  7 micke  staff  224 Oct 19 01:51 .
drwxr-xr-x  5 micke  staff  160 Oct 19 00:50 ..
drwxr-xr-x 14 micke  staff  448 Oct 20 16:35 .git
drwxr-xr-x  6 micke  staff  192 Oct 19 01:52 .idea
-rw-r--r--  1 micke  staff   39 Oct 19 01:51 README.md
-rw-r--r--  1 micke  staff  335 Oct 19 01:39 my_projekt.iml
-rw-r--r--  1 micke  staff   0 Oct 18 22:03 test.txt
➔ my_projekt git:(master) ✗
```

Var är mitt repo?

- Mappen som innehåller din .git-mapp är ditt repo.
- Ett ganska vanligt misstag man gör i början är att skapa ett repo som inte är tomt, t ex att göra om hela sin Dokument-mapp till ett repo. Det enda man behöver göra då är att ta bort sin .git-mapp och börja om.

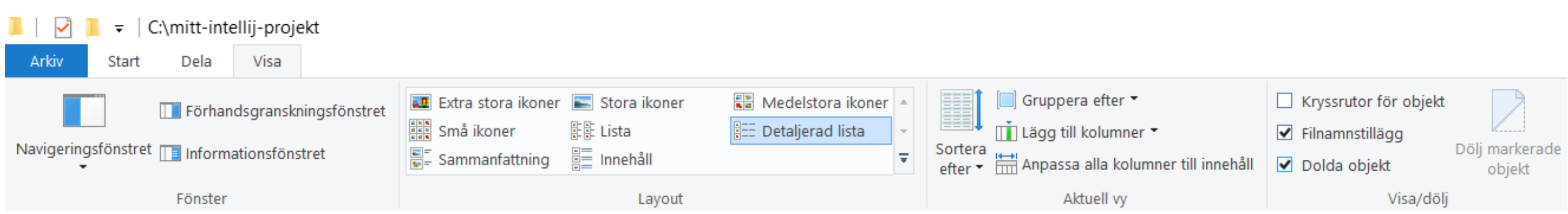
Vad är ett lämpligt repo?

- Det är lämpligt att ha ett repo per projekt.
- En del nybörjare lägger flera projekt i samma repo för att man vill samla ihop alla projekt, men det gör det svårare att hålla reda på projekten och att dela de projekt man vill dela.

Vad innehåller repot?

- Allt som hör till git sparas i mappen `.git`.
- Om den inte syns i Utforskaren kan ni ställa in visningsalternativ i Windows.
- För vardagligt arbete behöver man inte röra något i den här mappen.

```
1. micke@Mikaels-MacBook-Pro: ~/www/projects/academy/git/my_projekt (zsh)
➔ my_projekt git:(master) ls -la
total 0
drwxr-xr-x  3 micke  staff   96 Oct 18 21:43 .
drwxr-xr-x  3 micke  staff   96 Oct 18 12:22 ..
drwxr-xr-x  9 micke  staff  288 Oct 18 21:47 .git
➔ my_projekt git:(master) ls -la .git
total 24
drwxr-xr-x  9 micke  staff  288 Oct 18 21:47 .
drwxr-xr-x  3 micke  staff   96 Oct 18 21:43 ..
-rw-r--r--  1 micke  staff   23 Oct 18 21:43 HEAD
-rw-r--r--  1 micke  staff  137 Oct 18 21:43 config
-rw-r--r--  1 micke  staff   73 Oct 18 21:43 description
drwxr-xr-x 13 micke  staff  416 Oct 18 21:43 hooks
drwxr-xr-x  3 micke  staff   96 Oct 18 21:43 info
drwxr-xr-x  4 micke  staff  128 Oct 18 21:43 objects
drwxr-xr-x  4 micke  staff  128 Oct 18 21:43 refs
➔ my_projekt git:(master)
```



Konfigurera git

- För att git ska veta vem som gör vad måste du ange vem du är.

```
$ git config --global user.name "John Doe"
```

```
$ git config --global user.email johndoe@example.com
```

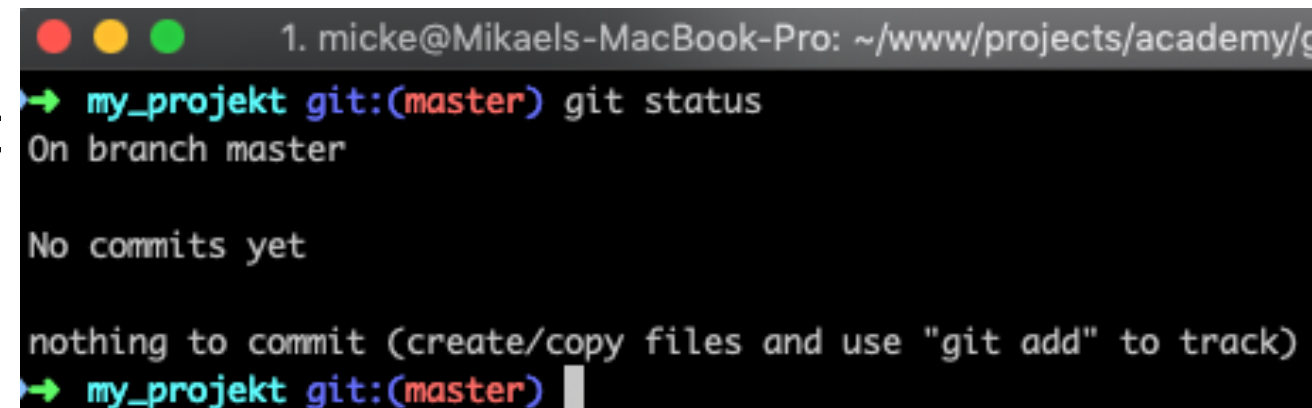
- Använd samma mailadress som du har använt för ditt konto på GitHub.
- Du kan ange vilken editor du vill använda. Troligen får du en fråga om vilken editor du vill använda under installationen.

```
$ git config --global core.editor "'C:\Program Files  
(x86)\Microsoft VS Code\code.exe' -w"
```

<https://git-scm.com/book/en/v2/Getting-Started-First-Time-Git-Setup>

Nu då?

- *Uppgift:* Kör `git status`.
- Vad säger informationen?
 - En *branch* är lite som en kopia av mappen. Vi kommer att prata mer om brancher vid ett senare tillfälle.
 - *No commits yet* innebär att vi inte har sparat några ändringar i repot än.
 - *Nothing to commit* innebär att det inte finns ändringar att spara.

A terminal window with a dark background and light-colored text. The title bar shows three colored window control buttons (red, yellow, green) and the text '1. micke@Mikaels-MacBook-Pro: ~/www/projects/academy/g'. The terminal content shows a prompt followed by the command 'git status'. The output is: 'On branch master', 'No commits yet', and 'nothing to commit (create/copy files and use "git add" to track)'. A second prompt is visible at the bottom with the command 'git status' again.

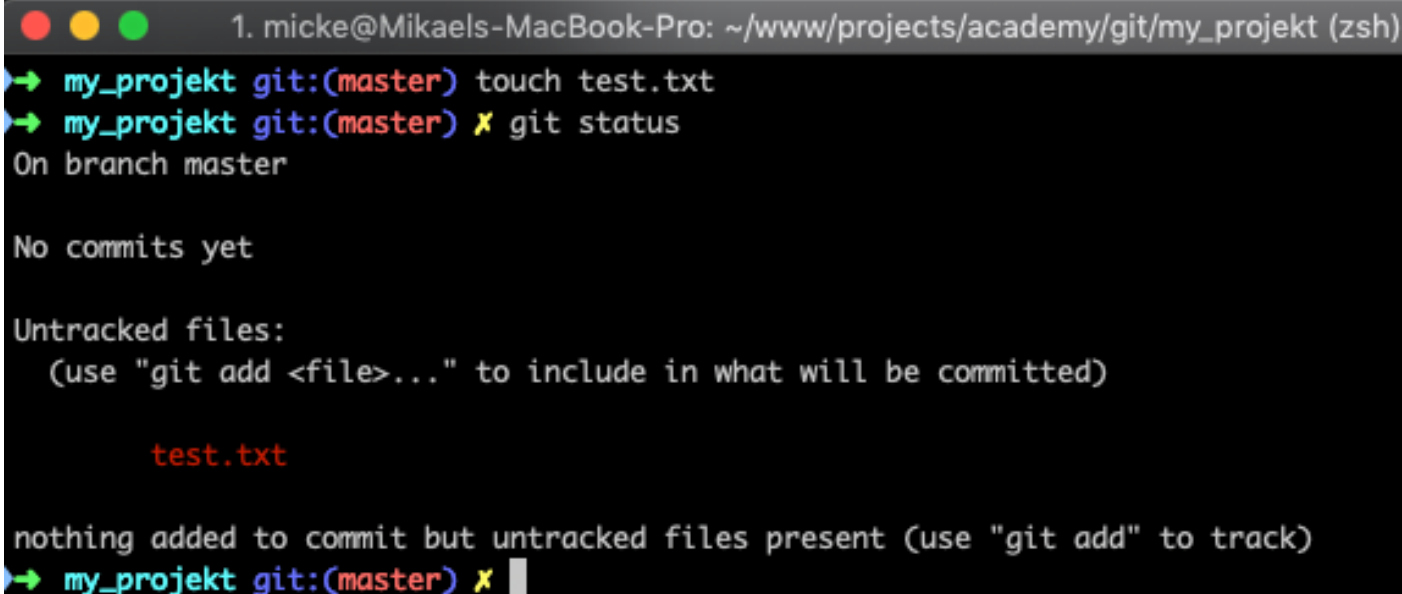
```
1. micke@Mikaels-MacBook-Pro: ~/www/projects/academy/g
→ my_projekt git:(master) git status
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to track)
→ my_projekt git:(master) 
```

Lägg till en fil

- Tänk ditt repo som en vanlig mapp.
- *Uppgift:* Lägg till en fil som kan innehålla text.
- Kör `git status` igen.
Vad har ändrats?
- *Untracked files* innebär att vi har filer i repot som git inte håller reda på.

A terminal window with a dark background and light-colored text. The title bar shows '1. micke@Mikaels-MacBook-Pro: ~/www/projects/academy/git/my_projekt (zsh)'. The terminal shows the following commands and output:

```
→ my_projekt git:(master) touch test.txt
→ my_projekt git:(master) ✗ git status
On branch master

No commits yet

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    test.txt

nothing added to commit but untracked files present (use "git add" to track)
→ my_projekt git:(master) ✗
```

Lägg till en fil

- *Uppgift:* Använd `git add` för att git ska börja hålla koll på filen.
- Bara för att vi har sagt åt git att hålla koll på om filen ändras innebär det inte att git *sparar* ändringarna.

```
nothing added to commit but untracked files present (use "git add" to track)
➔ my_projekt git:(master) ✕ git add test.txt
➔ my_projekt git:(master) ✕ git status
On branch master

No commits yet

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

        new file:   test.txt

➔ my_projekt git:(master) ✕
```

Committa ändringar

- *Uppgift:* För att spara ändringarna gör vi en commit:
`git commit -m 'Message'`
- Det vi gör är att säga åt git att spara ändringarna i repot tillsammans med ett meddelande.
- Om allt har gått bra är ändringarna sparade nu!

```
➡ my_projekt git:(master) ✕ git commit -m 'My first commit'
[master (root-commit) a6a6222] My first commit
1 file changed, 0 insertions(+), 0 deletions(-)
create mode 100644 test.txt
➡ my_projekt git:(master) █
```

Lägga till text-fil

- I det här fallet skulle vi lägga till en fil som kan innehålla text.
- Git kan hålla reda på att både text- och binärfiler (bilder, Word-dokument osv) har ändrats, men det kan inte visa oss skillnader i binärfiler.
- Git bryr sig inte om tomma mappar.

Remote

- Man kan kлона sitt repo till andra datorer. Det innebär att man har ett gemensamt repo, men som kan befinna sig i olika versioner på olika datorer.
- Git fungerar jättebra lokalt, men det finns fördelar med att ha repot på flera ställen.
 - Off site-backup.
 - Lättare att samarbeta.
 - Du kan få tillgång till repot vilken dator du än sitter vid.

Leverantörer

- Det finns flera leverantörer som låter en sätta upp git-repon hos dem. Det finns även lösningar man kan sätta upp själv om man har en webbserver.

- <https://github.com/>

- <http://bitbucket.org>

- <https://gitlab.com/>





GitHub

- Github är en av de största leverantörerna.
<https://github.com/>
- Du kan lägga upp obegränsat med publika repon. Ett publikt repo kan alla se allt innehåll i.
- Mot betalning kan du även lägga upp privata repon. I ett privat repo kan ägaren bestämma vem som får se innehållet.
- En github-profil kan ibland vara mer effektiv än ett CV.

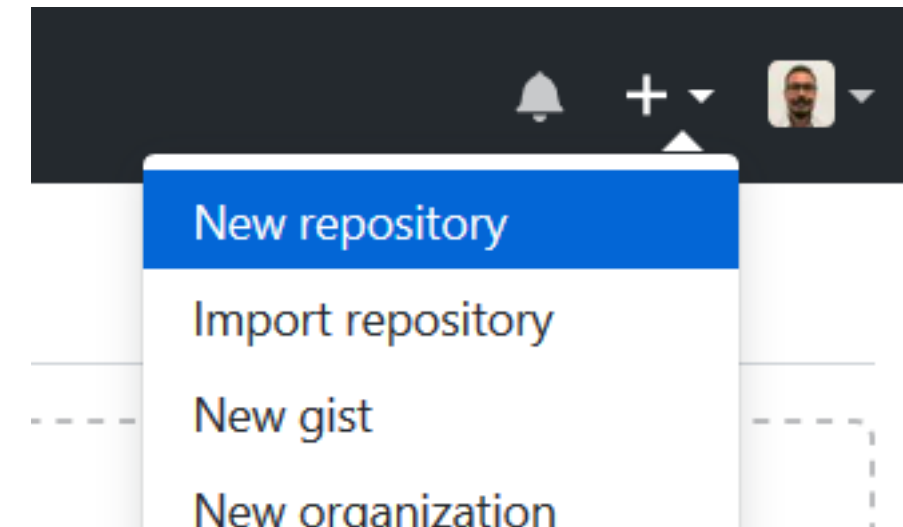
Koppla ihop repor

- I den kommande övningen ska vi koppla ihop ett existerande lokalt repo med ett existerande remote repo.
- Det vanligaste/enklaste tillvägagångssättet är annars att man skapar sitt repo remote på t ex GitHub först och sedan klonar det, men vi kan hamna i situationer där man behöver kunna koppla ihop existerande repor.



GitHub

- *Uppgift:* Logga in / registrera dig och skapa ett repo.



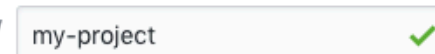
Create a new repository

A repository contains all the files for your project, including the revision history.

Owner



Repository name



Great repository names are short and memorable. Need inspiration? How about [vigilant-memory](#).

Description (optional)



Public

Anyone can see this repository. You choose who can commit.



Private

You choose who can see and commit to this repository.



Initialize this repository with a README

This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None**

Add a license: **None**



Create repository



GitHub

- Nu har du två orelaterade repon, ett lokalt och ett på GitHub.
- *Uppgift:* Kopiera sökvägen till ditt repo. Nu kan du koppla ihop ditt lokala repo med ditt remote-repo.

The screenshot shows the GitHub interface for a repository named 'my-project'. At the top, there are navigation tabs: Code, Issues (0), Pull requests (0), Projects (0), Wiki, Insights, and Settings. Below these, there's a description field with a placeholder 'description, website, or topics provided.' and an 'Edit' button. The repository statistics show 1 commit, 1 branch, 0 releases, and 1 contributor. A dropdown menu is open for 'ch: master', showing options like 'New pull request', 'Create new file', 'Upload files', 'Find file', and 'Clone or download'. The 'Clone or download' button is highlighted with a red box. Below this, a modal window titled 'Clone with HTTPS' is open, showing the URL 'https://github.com/emmio-micke/my-prc' and a copy icon. The modal also has buttons for 'Open in Desktop' and 'Download ZIP'.



GitHub

- *Uppgift:* Lägg till ditt remote-repo till ditt lokala repo i terminalen:

```
git remote add origin https://github.com/  
[red box]<username>/[blue box]<your-repo>.git
```

- Nu har vi lagt till vårt remote repo, men våra ändringar är inte sparade där.

```
➔ my_projekt git:(master) git remote add origin https://github.com/[red box]emmio-micke[blue box]my-project.git  
➔ my_projekt git:(master) |
```




GitHub

- *Uppgift:* För att hämta ändringarna som är gjorda på vårt remote-repo använder vi: `git pull`
- Oups! Vi har inte pratat om brancher än, men git förstår inte om vi vill att branchen *master* ska följa branschen *master* **på remote**. Vi kan ange att vi vill det.
`git branch --set-upstream-to=origin/master master`

```
1. micke@Mikaels-MacBook-Pro: ~/www/projects/academy/git/my_proje
➔ my_projekt git:(master) git pull
warning: no common commits
remote: Enumerating objects: 3, done.
remote: Counting objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), done.
From https://github.com/emmio-micke/my-project
* [new branch]      master      -> origin/master
There is no tracking information for the current branch.
Please specify which branch you want to merge with.
See git-pull(1) for details.

    git pull <remote> <branch>

If you wish to set tracking information for this branch you can do so with:

    git branch --set-upstream-to=origin/<branch> master
➔ my_projekt git:(master) █
```

```
➔ my_projekt git:(master) git branch --set-upstream-to=origin/master master
Branch 'master' set up to track remote branch 'master' from 'origin'.
➔ my_projekt git:(master) █
```

Main

- Tidigare har det varit standard att låta "huvud-branchen" heta *master*.
- Alldeles nyligen tog GitHub beslutet att döpa om huvud-branchen till *main*.



GitHub

- *Uppgift:* Nytt försök:
`git pull`
- *Uppgift:* Oups! Det här innebär att git inte ser att det finns någon gemensam bas mellan vårt lokala repo och vårt remote-repo. Det finns inga filer i det ena repot som även finns i det andra, så git tycker inte att dessa repon verkar ha med varandra att göra. Vi kan fixa det.
`git pull --allow-unrelated-histories`

```
1. micke@Mikaels-MacBook-Pro: ~/v  
→ my_projekt git:(master) git pull  
fatal: refusing to merge unrelated histories  
→ my_projekt git:(master) █
```



GitHub

- Vad har hänt nu?
- Git kan slå ihop ändringarna i de båda repona. Det kallas för *merge*.
- Som standard kommer Git att automatiskt att vilja göra en commit för att spara ändringarna i mergen. Vi har inte angett något commit-meddelande, så git öppnar en texteditor så vi kan ange det där. Git föreslår också ett meddelande åt oss.

```
1. git pull --allow-unrelated-histories (vim)
Merge branch 'master' of https://github.com/emmio-micke/my-project

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
~
~
```



GitHub

- Vilken editor man får upp beror på hur ens inställningar ser ut.
- Vi tittade tidigare på hur man kan ställa in t ex Visual Studio Code som standardeditor.
- Har man inte valt något får man troligen vim som editor.
- Vim är en kompetent editor med en något hög inlärningströskel.
- Det kan löna sig att lära sig baskommandon i vim eftersom det är vanligt på servrar där man inte har tillgång till en desktop-miljö.



```
1. git pull --allow-unrelated-histories (vim)
Merge branch 'master' of https://github.com/emmio-micke/my-project

# Please enter a commit message to explain why this merge is necessary,
# especially if it merges an updated upstream into a topic branch.
#
# Lines starting with '#' will be ignored, and an empty message aborts
# the commit.
~
~
~
```

- *Uppgift:* För att ändra text i vim:
 - Använd piltangenterna för att placera markören.
 - Tryck `<i>` för att hamna i *insert mode*.
 - Gör ändringarna och återgå till normalläget med `<esc>`.
- För att spara filen och avsluta vim:
`:wq`



GitHub

- *Uppgift:* Allt ser ut att ha gått bra! Låt oss kolla hur mappen ser ut nu.

```
ls -la
```

- Vi har fått in README.md-filen som vi hade i vårt remote-repo.

```
➔ my_projekt git:(master) git pull --allow-unrelated-histories
Merge made by the 'recursive' strategy.
 README.md | 1 +
 1 file changed, 1 insertion(+)
 create mode 100644 README.md
➔ my_projekt git:(master) █
```

```
➔ my_projekt git:(master) ls -la
total 8
drwxr-xr-x  5 micke  staff  160 Oct 19 00:17 .
drwxr-xr-x  3 micke  staff   96 Oct 18 12:22 ..
drwxr-xr-x 14 micke  staff  448 Oct 19 00:28 .git
-rw-r--r--  1 micke  staff   12 Oct 19 00:17 README.md
-rw-r--r--  1 micke  staff    0 Oct 18 22:03 test.txt
➔ my_projekt git:(master) █
```



GitHub

- *Uppgift:* Vi kollar statusen på vårt repo.

```
git status
```

- Git jämför statusen på vårt repo med statusen på vårt remote-repo och ser att vårt lokala repo har två stycken commits som inte finns på vårt remote-repo. (*Ahead by 2 commits.*)
- Om det hade funnits commits på vårt remote-repo som inte funnits i vårt lokala repo hade det stått att vår branch *is behind* vårt remote-repo.

```
➤ my_projekt git:(master) git status
On branch master
Your branch is ahead of 'origin/master' by 2 commits.
  (use "git push" to publish your local commits)

nothing to commit, working tree clean
➤ my_projekt git:(master) █
```





GitHub



- *Uppgift:* Vi kan ladda upp våra commits till vårt remote-repo.
`git push`
- Nu är statusen på vårt remote-repo uppdaterad.


```
➔ my_projekt git:(master) git push
Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 8 threads.
Compressing objects: 100% (3/3), done.
Writing objects: 100% (5/5), 522 bytes | 522.00 KiB/s, done.
Total 5 (delta 0), reused 0 (delta 0)
To https://github.com/emmio-micke/my-project.git
    f9f0f68..1c1f1a0  master -> master
➔ my_projekt git:(master)
```

3 commits 1 branch

Branch: master New pull request

 **emmio-micke** Merge branch 'master' of <https://github.com/emmio-micke/my-project>

 README.md	Initial commit
 test.txt	My first commit

 [README.md](#)

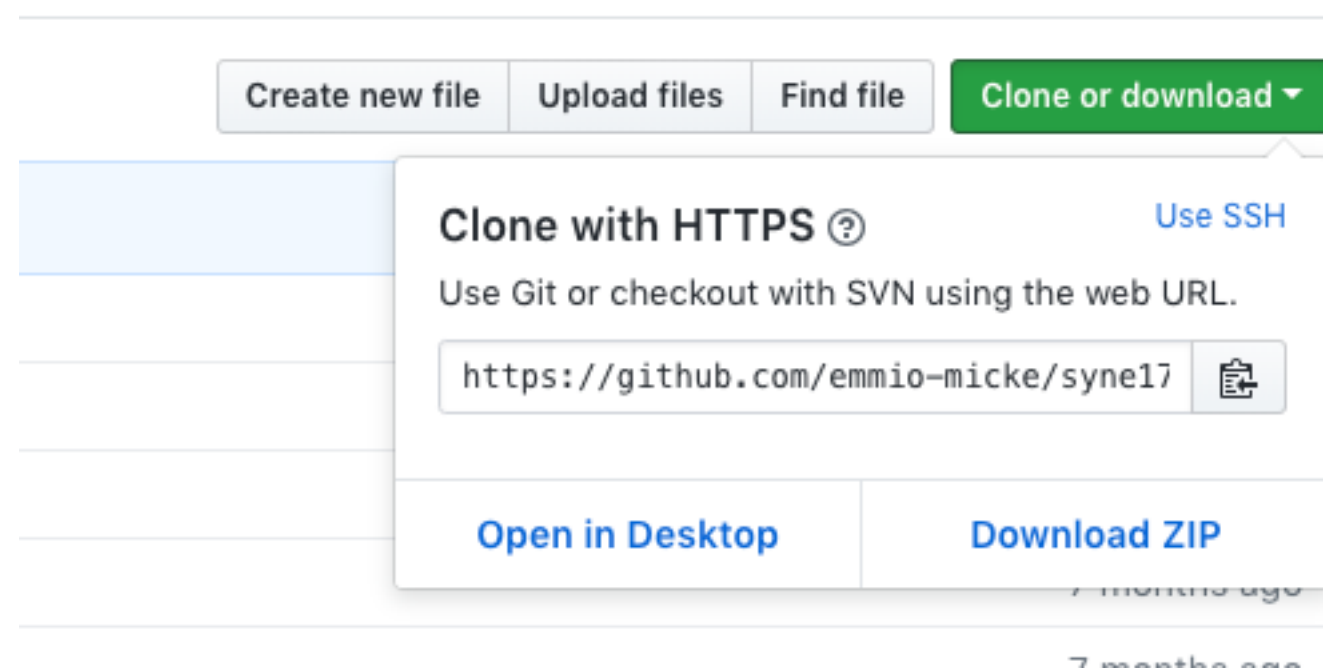
Klona repo

- Det vanligaste sättet att börja samarbeta med ett existerande repo är att klonas det.
- *Uppgift:* Skapa **inte** en ny projekt-mapp utan navigera till mappen ovanför ditt förra projekt.

```
➤➔ my_projekt git:(master) cd ..
➤➔ git ls -la
total 0
drwxr-xr-x  3 micke  staff   96 Oct 18 12:22 .
drwxr-xr-x  3 micke  staff   96 Oct 18 12:18 ..
drwxr-xr-x  5 micke  staff  160 Oct 19 00:17 my_projekt
➤➔ git
```

Klona repo

- *Uppgift:* Skaffa adressen till din bänkgrannes repo och klonar det.
`git clone https://github.com/<user>/<repo-name>.git`
- Git kommer att hämta repot till en mapp med samma namn som repot. Om repot heter syne17.git kommer mappen att heta syne17.
- Om man vill döpa projektet till något annat kan man ange mappens namn som argument.
`git clone https://github.com/<user>/<repo-name>.git projekt2`

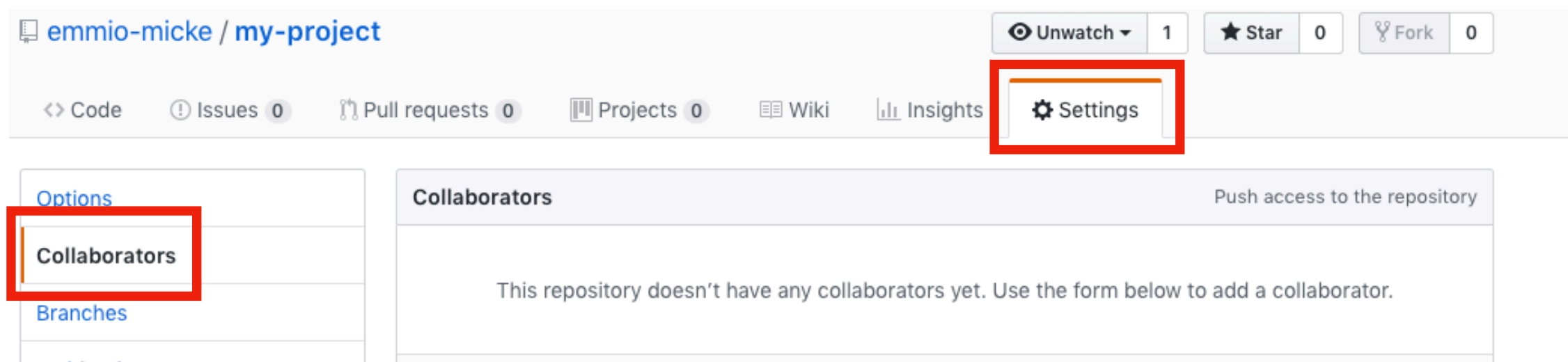


```
→ git git clone https://github.com/emmio-micke/syne17.git
Cloning into 'syne17'..
remote: Enumerating objects: 389, done.
remote: Total 389 (delta 0), reused 0 (delta 0), pack-reused 389
Receiving objects: 100% (389/389), 3.86 MiB | 3.03 MiB/s, done.
Resolving deltas: 100% (183/183), done.
→ git
```

```
→ git git clone https://github.com/emmio-micke/syne17.git projekt2
Cloning into 'projekt2'..
remote: Enumerating objects: 389, done.
remote: Total 389 (delta 0), reused 0 (delta 0), pack-reused 389
Receiving objects: 100% (389/389), 3.86 MiB | 3.02 MiB/s, done.
Resolving deltas: 100% (183/183), done.
→ git
```

Collaborators

- *Uppgift:* Gör en ändring eller lägg till en fil i ditt klonade repo.
- Committa ändringen och pusha. Vad händer?
- Eftersom du inte äger repot har du inte skrivrättigheter till det.
- Användarrättigheter hanteras olika i olika system. (Github, bitbucket osv.)
- I Github kan man lägga till collaborators, konton som ska kunna skriva ändringar till ens repo.
- *Uppgift:* Lägg till din kollega som collaborator. Testa att det funkar.




Adding files

- Varför måste jag lägga till ändrade filer varje gång? Varför kan jag inte bara committa?
- För att du ska kunna välja vilka ändringar du vill spara. Det är inte säkert att du vill committa alla filer du har gjort ändringar i.

Lista commits

- För att lista commit-historiken kan vi använda `git log`.
- Bläddra upp och ner med piltangenterna, avsluta med `q`.
- Varje commit har en *hash*, en slags id.



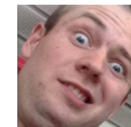
```
commit 74e6d22b5cf909bad744d72fc2ac2991bdc7e02b (HEAD -> master, origin/  
Merge: 3e469a8 5f5a35d  
Author: Mikael Olsson <mikael.olsson@emmio.se>  
Date: Tue Jul 3 10:56:27 2018 +0200  
  
Merge branch 'paula'  
  
# Conflicts:  
#     final project/repo.txt  
  
commit 5f5a35d1acc8caa055eef128288465dd9c5c9096 (origin/paula)  
Author: paulazhao <32543902+paulazhao@users.noreply.github.com>  
Date: Fri Jun 29 16:36:55 2018 +0200  
  
repoCommit  
  
commit 3e469a822909ed02e758ba41c811cbfd6d9a39de (origin/Christer)  
Merge: 6f0bb4a 39d750d  
Author: Linghult <j.linghult@gmail.com>  
Date: Sun Apr 8 19:31:36 2018 +0200  
  
Merge branch 'master' of https://github.com/emmio-micke/syne17  
  
commit 39d750d072b1f640d201aeef6e3d67169e56fd06  
:
```

Commit

- Hur ofta ska man committa?
 - Beror lite på men det är nästan aldrig en nackdel att committa ofta, kanske upp till ett par gånger i timmen.
- Hur ser ett bra commit-meddelande ut?
 - Det beskriver ändringen, vilket problem man har försökt lösa, ev issue-id.
 - **Bra:** #123 Fixes the issue with rendering problems.
 - **Mindre bra:** Worked on stuff.

Commits from last night

Humorsite. Går igenom commit-loggar från publika repon och tar med alla meddelanden som har svärord eller liknande i sig.



MartinGarenfeld
04/02/18 10:24 AM

`some lda shit`



coltonon
04/02/18 10:06 AM

`Added full project, code is still shit though`



blunt1337
04/02/18 9:05 AM

`Removed this mini css shit`



SpicySnail
04/02/18 8:56 AM

`fuck me`



Poojawa
04/02/18 8:46 AM

`wew lad. fuck off with this already`



Geekyew
04/02/18 8:45 AM

`Great googly moogly it's all gone to shit`



bcarbong
04/02/18 8:44 AM

`recursive insertion sort working, hell fucking yeah`

<http://www.commitlogsfromlastnight.com/>

Pull

- Hur ofta ska man göra `git pull`?
- Hyfsat ofta. Man vill ha sina kollegors ändringar för att undvika mergekonflikter och nya buggar. Det kan vara bra att göra en `git pull` innan du gör en `git push`.

Stash

- Om du har gjort ändringar i filer och inte sparat/committat dem kommer git att protestera om du försöker göra en `git pull`.
- Det beror på att om du skulle göra en git pull skulle alla sparade ändringar skrivas över.
- Du kan committa ditt arbete för att spara det.
- Om du inte vill det av någon anledning kan du istället stasha dina ändringar. Det innebär att spara undan alla ändringar för att senare kunna re-applya dem igen.
`git stash`
- Applya genom
`git stash pop`
 - <https://git-scm.com/docs/git-stash>

Speciella filer

- Ibland vill man inte att vissa filer ska hanteras av git. Det kan t ex vara config-filer, filer som innehåller lösenord osv.
- I git kan man skapa filen `.gitignore` som innehåller regler för vilka filer som ska ignoreras.
- Det kan vara specifika filer, mappar, allt i en mapp med vissa undantag osv.

.gitignore i Windows

- Windows gillar inte filer som heter `.nånting`. För Windows ser det ut som att filen inte har något filnamn.
- Du kan skapa filen i git bash, sedan kan du öppna den med valfri editor.
`touch .gitignore`

.gitignore

```
$ git status
```

```
[...]
```

```
# Untracked files:
```

```
[...]
```

```
#
```

```
Documentation/foo.html
```

```
#
```

```
Documentation/gitignore.html
```

```
#
```

```
file.o
```

```
#
```

```
lib.a
```

```
#
```

```
src/internal.o
```

```
[...]
```

.gitignore

```
# ignore objects and archives,  
# anywhere in the tree.  
*.[oa]
```

```
Documentation/foo.html  
Documentation/gitignore.html  
file.o  
lib.a  
src/internal.o
```

```
# ignore generated html files,  
*.html
```

```
# except foo.html which is  
# maintained by hand  
!foo.html
```

```
$ git status  
[...]  
# Untracked files:  
[...]  
#       Documentation/foo.html  
[...]
```

.gitignore - tips

- Git bryr sig inte om tomma mappar. Om man ignorerar allt innehåll i en mapp så kommer själva mappen inte heller att komma med i repot.
- Man kan skapa en tom fil, kalla den t ex *empty*, och be git att ignorera allt i mappen förutom den filen.

```
images/*  
!images/empty
```


.gitignore

- På gitignore.io kan man få färdiga filer beroende på vilka behov/ide:er/miljöer man har.
- *Uppgift:* Gå in på gitignore.io och skapa en fil. Spara filen till ditt repo, committa, pusha och testa att den fungerar.



Merge

- *Uppgift:* Samarbeta med en kompis, använd samma repo.
- Gör ändringar i olika filer på varsin dator.
- Committa och pulla/pusha. Vad händer?
- Git kan automatiskt merga många ändringar, det såg vi förut när vi lade till vårt remote-repo. I det här fallet kommer git att göra en merge och committa den, vi behöver bara bidra med ett meddelande.

Merge-konflikt

- *Uppgift:* Samarbeta med en kompis, använd samma repo.
- Gör ändringar i samma fil på varsin dator.
- Committa och pulla/pusha. Vad händer?
- Om git inte automatiskt kan merga ihop era ändringar kommer ni att få en merge-konflikt.

Lösa en merge-conflict

- *Uppgift:* Börja med att ta reda på vilka filer som har en merge-konflikt, om du inte redan vet.
`git status`
- Öppna filen i valfri editor och öppna filen med merge-konflikten.
- Konflikten kommer att visas med speciella markörer.
- Redigera filen för att spara de ändringar du vill ha kvar och spara filen.

```
$ git status
# On branch branch-b
# You have unmerged paths.
#   (fix conflicts and run "git commit")
#
# Unmerged paths:
#   (use "git add ..." to mark resolution)
#
# both modified:   styleguide.md
#
no changes added to commit (use "git add" and/or "git commit -a")
```

```
If you have questions, please
<<<<<<< HEAD
open an issue
=====
ask your question in IRC.
>>>>>>> branch-a
```

Lösa en merge-conflict

- *Uppgift:* Lägg till dina ändringar.

```
git add .
```

- **Committa och pusha.**

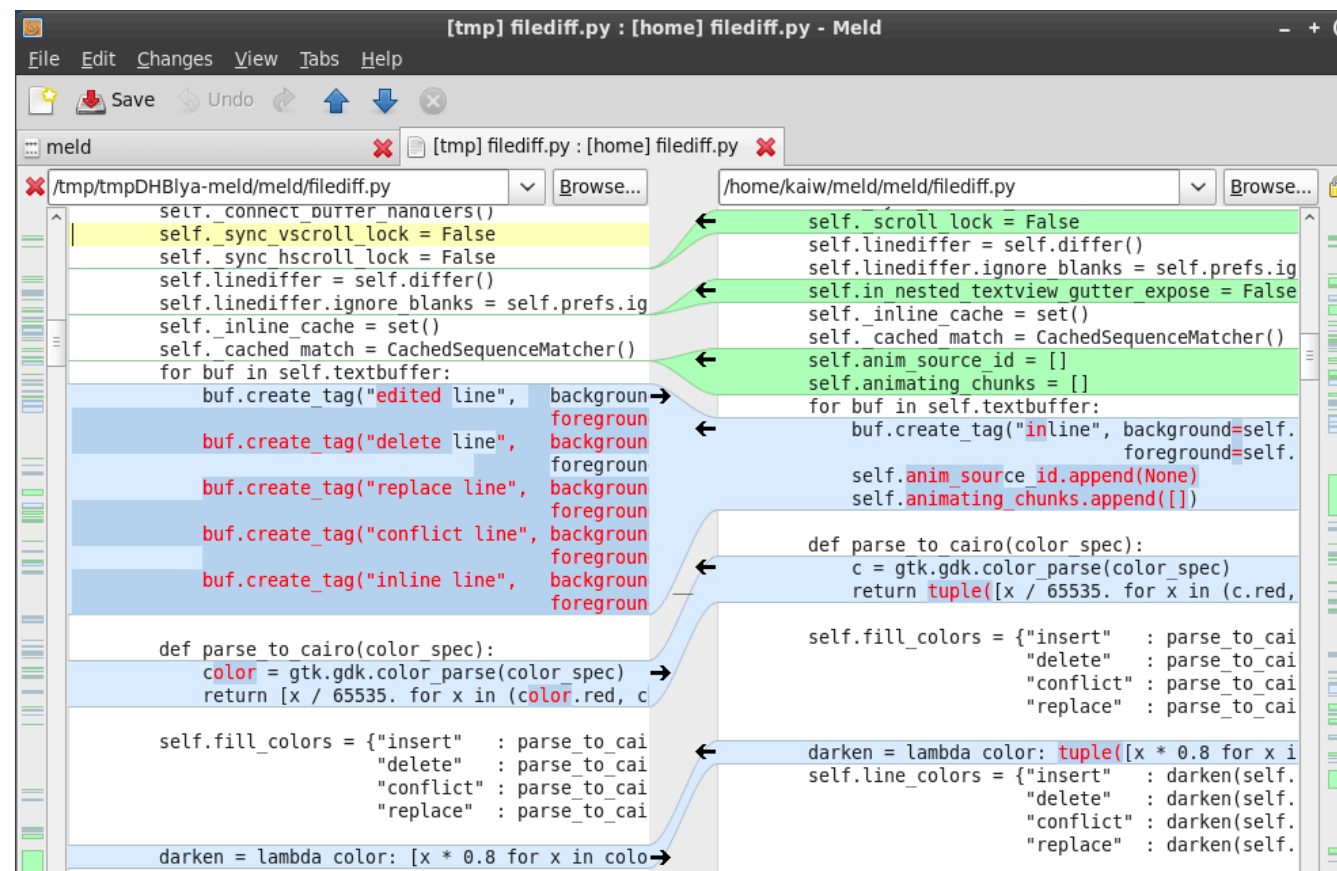
```
git commit -m "Resolved merge conflict by  
incorporating both suggestions."
```

```
git push
```

- <https://help.github.com/articles/resolving-a-merge-conflict-using-the-command-line/>

Diff-verktyg

Vi öppnade filen i en editor och letade efter markörerna, men det finns program och editorer som gör detta på ett bättre sätt åt oss, nämligen diff-verktyg. Det kan vara värt att hitta ett bra diff-verktyg och lära sig hur det fungerar.



Git blame

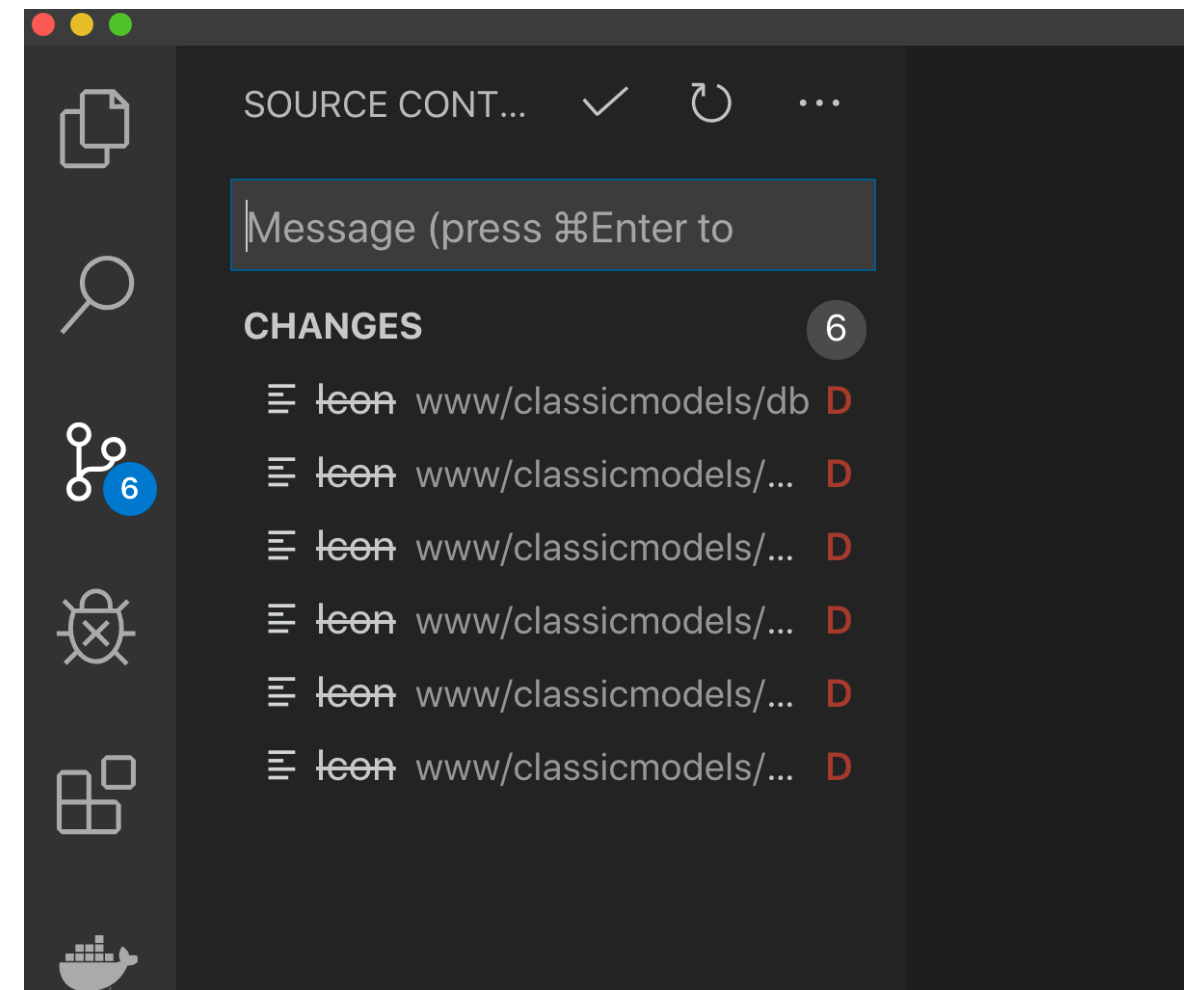
- Man kan se vem som har ändrat en specifik fil.
`git blame <file>`
- Precis som i `git log` navigerar man med piltangenterna och avslutar med `q`.

```
➔ final project git:(master) ✗ ls -la
total 280
drwxr-xr-x  5 micke  staff   160 Oct 19 00:50 .
drwxr-xr-x 10 micke  staff   320 Oct 19 00:53 ..
-rw-r--r--  1 micke  staff 132701 Oct 19 00:50 Slutprojekt.pdf
-rw-r--r--  1 micke  staff  1770 Oct 19 00:50 queries.txt
-rw-r--r--  1 micke  staff  1027 Oct 19 00:50 repo.txt
➔ final project git:(master) ✗ git blame repo.txt
➔ final project git:(master) ✗
```

```
1. git blame repo.txt (less)
74e6d22b (Mikael Olsson      2018-07-03 10:56:27 +0200 1) # Klistra in en länk i https-form till ditt repo k
14164acf (Ida Nilsson       2018-03-29 14:34:29 +0200 2) https://github.com/emmio-micke/syne17-slutprojekt.
14164acf (Ida Nilsson       2018-03-29 14:34:29 +0200 3)
7e8b3c2d (Ida Nilsson       2018-03-29 14:55:38 +0200 4) https://github.com/
1f2f0793 (Linghult         2018-03-29 14:32:43 +0200 5) https://github.com/
7e8b3c2d (Ida Nilsson       2018-03-29 14:55:38 +0200 6) https://github.com/
a64f0936 (Ida Nilsson       2018-03-29 14:56:48 +0200 7) https://github.com/
814dd31a (jennybacklund    2018-03-29 14:47:44 +0200 8) https://github.com/
e939618b (veronika.borup   2018-03-29 14:46:00 +0200 9) https://github.com/
2008235b (veronika.borup   2018-03-29 14:48:45 +0200 10) https://github.com/
2008235b (veronika.borup   2018-03-29 14:48:45 +0200 11) https://github.com/
7c50d7f6 (luddepantzar     2018-03-29 15:42:15 +0200 12) https://github.com/
```

Visual Studio Code

- VS Code har rätt bra stöd för att använda Git direkt.
- Det är bra att kunna använda git i terminalen eftersom man sällan har tillgång till desktop-program på en server, men i dagligt arbete är det enklare att använda VS Codes inbyggda eller ett annat, fristående verktyg.



Git GUI

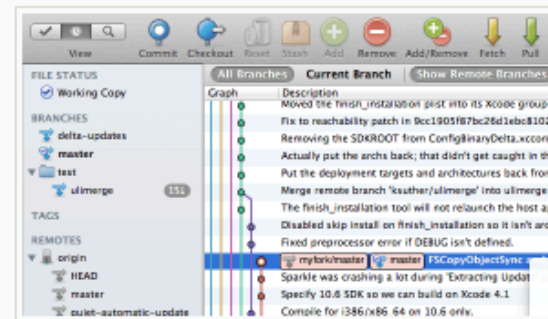
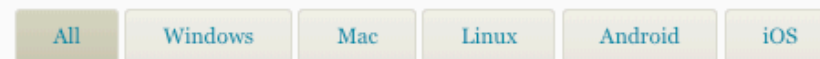
- Om du inte gillar VS Codes inbyggda git-funktioner och inte gillar terminalen finns det många alternativ. Personligen gillar jag SourceTree, även om jag har gått över till att använda det inbyggda i VS Code för det mesta.

- <https://git-scm.com/downloads/guis>

GUI Clients

Git comes with built-in GUI tools for committing ([git-gui](#)) and browsing ([gitk](#)), but there are several third-party tools for users looking for platform-specific experience.

If you want to add another GUI tool to this list, just [follow the instructions](#).

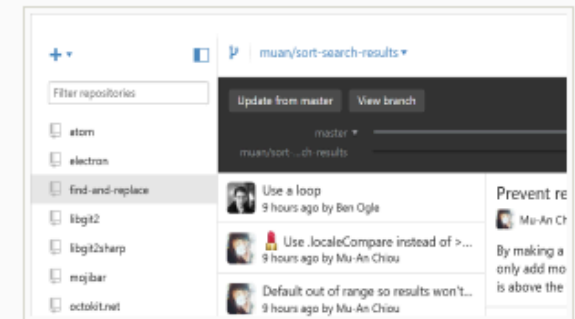


SourceTree

Platforms: Mac, Windows

Price: Free

License: Proprietary

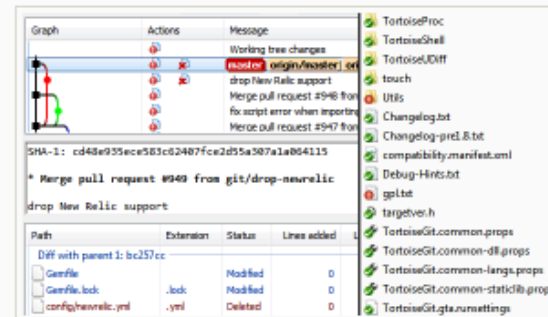


GitHub Desktop

Platforms: Mac, Windows

Price: Free

License: MIT

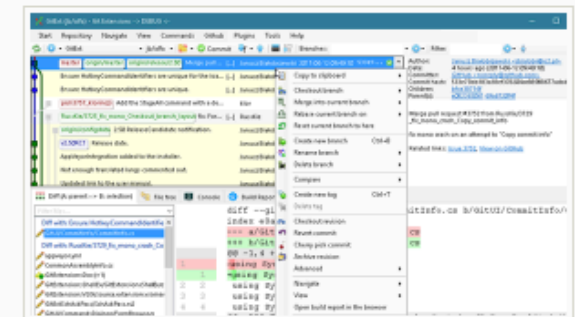


TortoiseGit

Platforms: Windows

Price: Free

License: GNU GPL



Git Extensions

Platforms: Linux, Mac, Windows

Price: Free

License: GNU GPL

Sammanfattning

- Git är ett distribuerat versionshanteringssystem.
 - <https://github.com>
 - Arbetsflöde
 - Status
 - Add
 - Commit
 - (Pull)
 - Push
 - Merge-konflikter
 - .gitignore
- `git init`
 - `git clone`
 - `git status`
 - `git add`
 - `git commit`
 - `git push`
 - `git pull`
 - `git stash`

Läs mer

- <https://git-scm.com/docs/>
- <https://www.atlassian.com/git/tutorials/>
- <https://sethrobertson.github.io/GitBestPractices/>