

DA339A Laboration L1

Syfte

Laborationens syfte är att skapa en arbetsmiljö för genomförande av kommande laborationer och uppgifter. Innehållet omfattar en introduktion till grundläggande navigering i filsystem via Git Bash eller Terminal, användning av keditorn Visual Studio Code samt en introduktion till versionshantering med Git.

Genomgången av Git är framtaget av Johan Holmberg och därefter redigerat för att passa detta kursinnehåll.

Följande tas upp i denna labb:

- Skapa plats för att hantera filer för denna kurs
- GitBash/Terminal
- Keditor
- Git

Redovisning

Laboration L1 ska inte redovisas.

Förberedelser

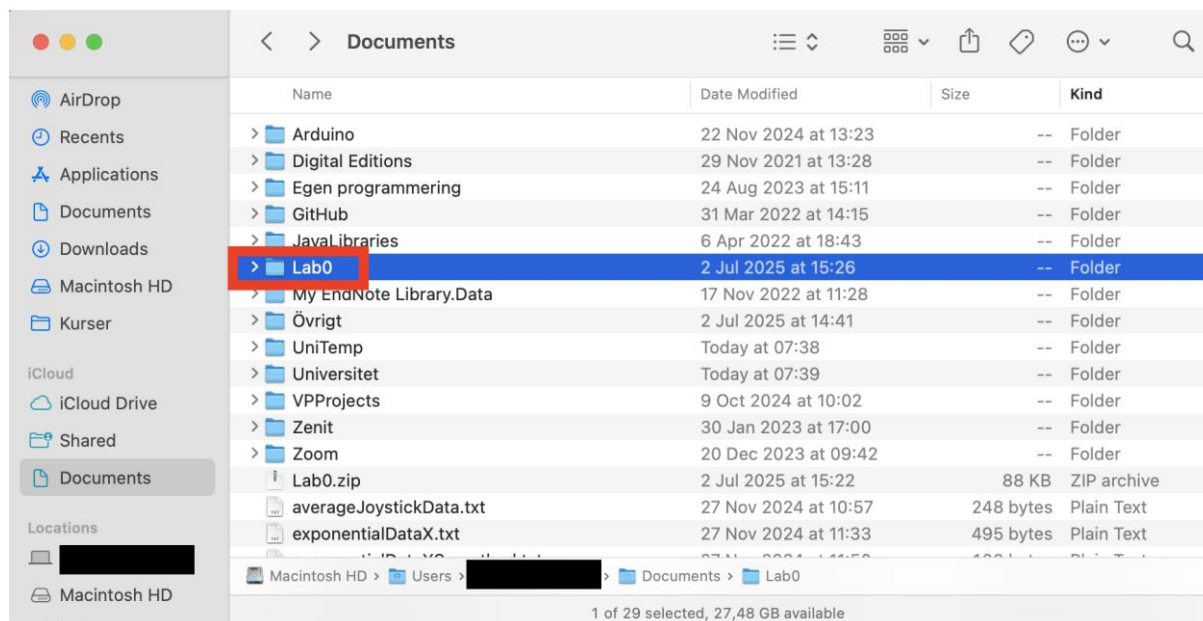
- Genomförandet av Lab 0 om grundläggande kunskaper i filhantering känns svaga.
- Deltagande eller i efterhand läst igenom Föreläsning 1 (F1)
- I förväg nedladdat programmen:
 - GitBash (OBS för Mac kan inbyggda Terminal användas i stället)
 - <https://git-scm.com/downloads/win>
 - GitHub Desktop
 - <https://docs.github.com/en/desktop/overview/getting-started-with-github-desktop>
 - Visual Studio Code
 - <https://code.visualstudio.com/download>

Filhantering

I föregående laboration lärdes det ut hur filer skapas i de olika operativsystemen MacOS och Windows 11. I denna laboration ska de filer som skapades läggas in i en större filstruktur vilket ska ligga till grund för hur all hantering av filer sker för kursen DA339A, men det går även att använda som mall för kommande kurser.

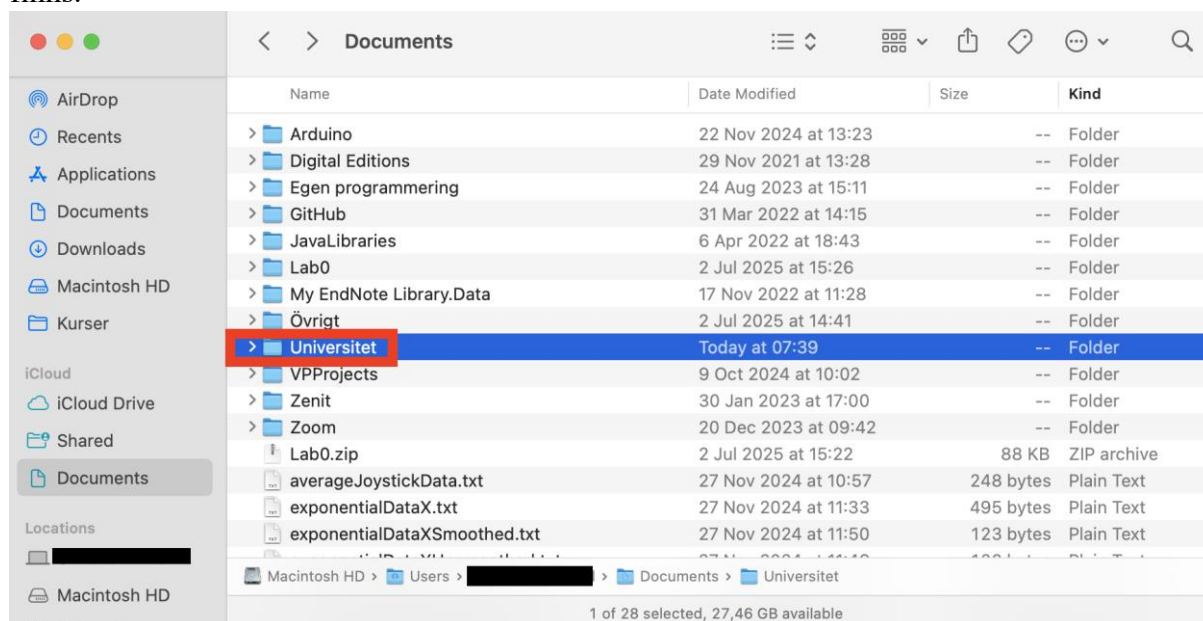
Mac

Börja med att öppna upp *Finder* och gå till *Dokument* vilket visades under Lab 0, väl i Finder kontrollera att katalogen med namnet Lab0 existerar i Dokument.



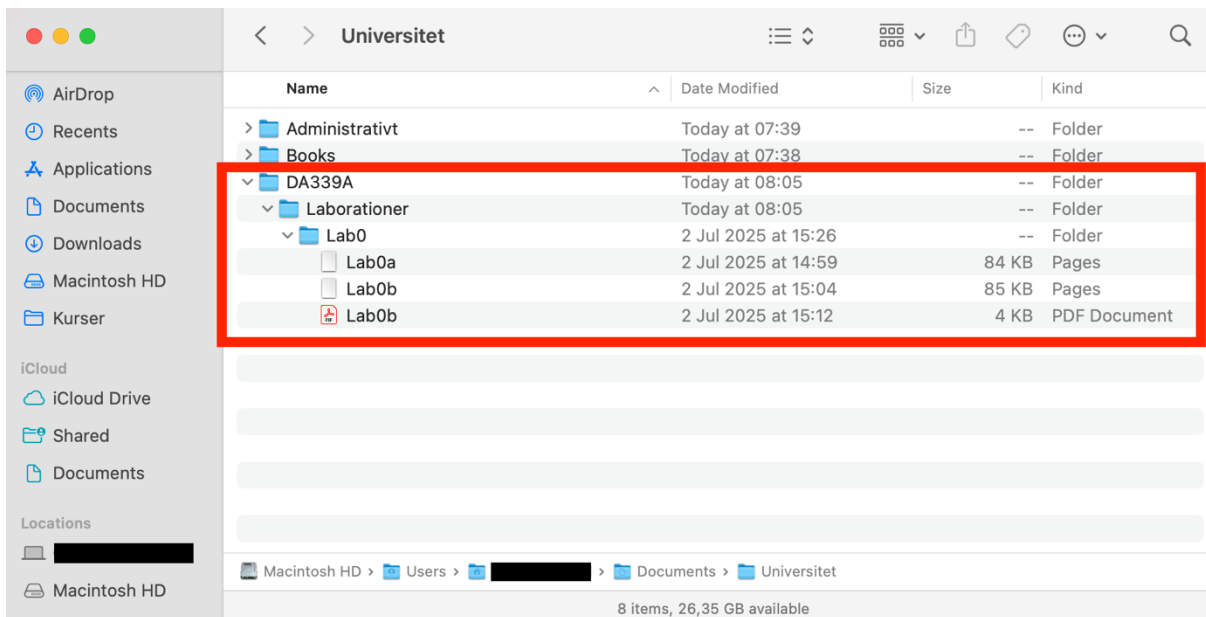
Figur 1: Kontrollera att Lab 0 finns.

I *Dokument* skapa nu en katalog som döps till exempelvis *Universitet*, detta ifall minnet sviker görs genom att högerklicka eller klicka med två fingrar på en plats där varken fil eller katalog finns.



Figur 2: Skapad katalog med namn Universitet.

Öppna nu den nyskapade katalogen, i denna katalog skapas ytterligare en katalog vilket döps exempelvis till *DA339A*, i *DA339A* skapa nu en katalog vilket döps till *Laborationer*. Med katalogen *Laborationer* skapad kan nu *Lab0* vilket ligger i Dokument flyttas in till *Laborationer*. Detta görs lättast genom att antingen dra och släppa katalogen *Lab0* på *Universitet*, alternativt fortsätt att hålla *Lab0* över *Universitet* i några sekunder, därefter hoppar Finder in i *Universitet*. Upprepa detta steg för katalogen *DA339A*, och slutligen kan katalogen *Labv0* nu släppas direkt i på katalogen *Laborationer*. Följs ovan steg bör resultatet se ut som nedan när alla kataloger är expanderade genom att trycka på den lilla pilen till vänster om katalogerna.

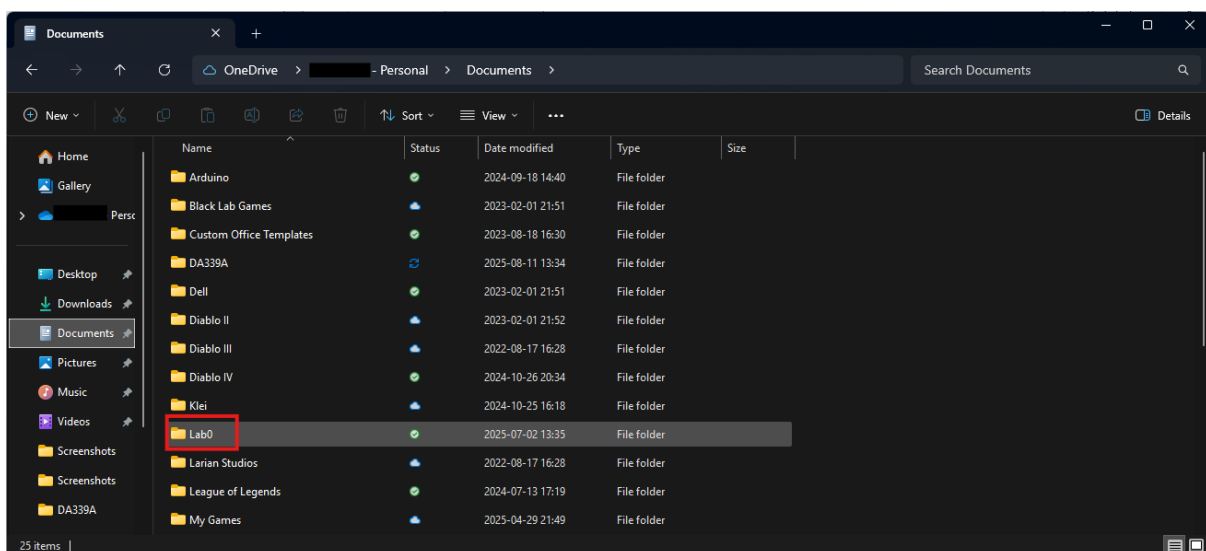


Figur 3: Översiktsbild av den skapade labbstrukturen.

OBS! Skapa inte en till katalog för Lab1 ännu.

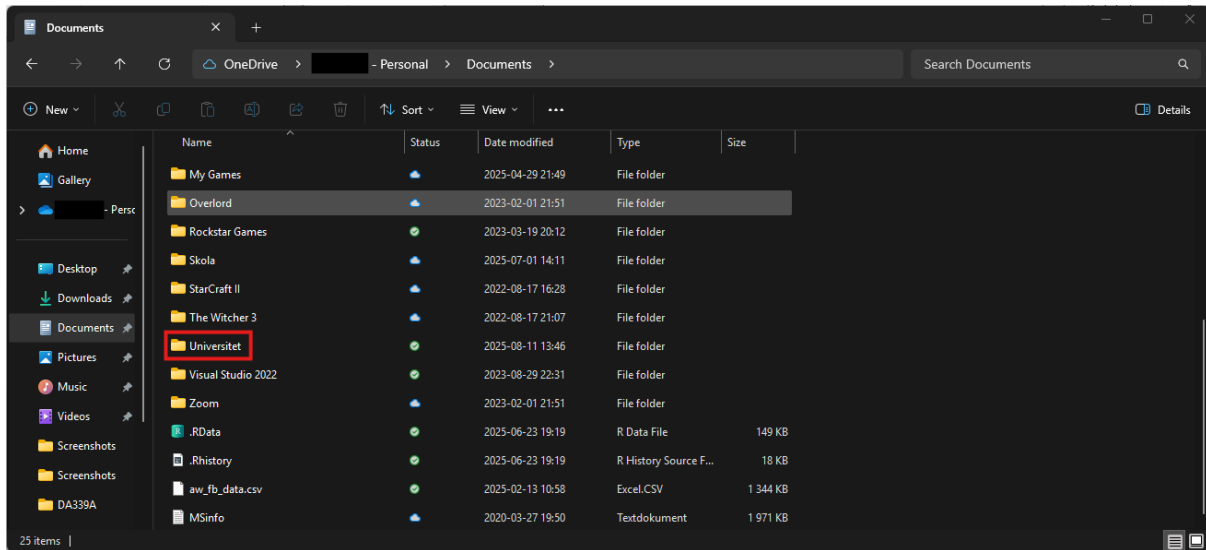
Windows

Börja med att öppna upp Filutforskaren och gå till Dokument, vilket visades under Laboration 0. Väl i Dokument kontrollera att katalogen med namnet *Lab0* existerar i Dokument.



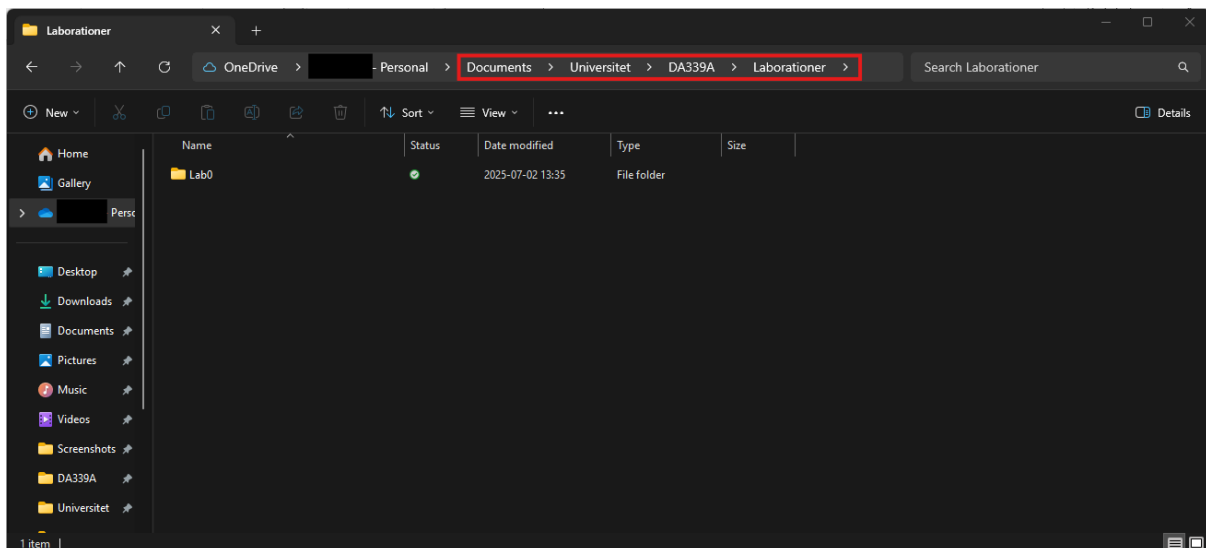
Figur 4: Kontrollera att *Lab0* finns.

I *Dokument* skapa nu en katalog som döps till exempelvis *Universitet*, detta ifall minnet sviker görs genom att antingen högerklicka i ett tomrum i *Dokument* och välja new i menyn som öppnas upp eller trycka på knappen *New* vilket ligger ovanför navigeringsrutan.



Figur 5: Skapad katalog med namn *Universitet*.

Öppna nu den nyskapade katalogen, i denna katalog skapas ytterligare en katalog vilket döps till *DA339A*, i *DA339A* skapa nu en katalog vilket döps till *Laborationer*. Med katalogen *Laborationer* skapad kan nu *Lab0* vilket ligger i *Dokument* flyttas in till *Laborationer*. Detta görs lättast genom att dra och släppa katalogen *Lab0* på *Universitet*, Upprepa detta steg för katalogen *DA339A*, och slutligen kan katalogen *Lab0* nu släppas direkt i på katalogen *Laborationer*. Om allt följs enligt ovan finns numera *Lab0* under *Laborationer* vilket indikeras i den rödmarkerade rutan.



Figur 6: Översiktsbild av den skapade labbstrukturen.

OBS! Skapa inte en till katalog för Lab1 ännu.

Navigering av filsystem

En viktig kunskap inom datavetenskap, vilken återkommer frekvent, är navigering genom verktyg som ofta benämns ”Command Line Interfaces (CLIs)”. Dessa verktyg är användbara då de ger en överblick av filstrukturen i miljöer som vanligtvis saknar stöd för mer avancerade grafiska gränssnitt, exempelvis servrar eller webbservrar.

I denna kurs kommer Git Bash för Windows och Terminalen för Mac att användas, Git Bash liknar Terminalen för Mac väldigt mycket. Det går att ladda ner Git Bash till Mac också, men då macOS bygger på Unix är det fördelaktigt att vänja sig vid Terminalen, eftersom detta ger en vana som senare kan tillämpas i Linux om man någon gång vill göra bytet.

Nedan följer tabeller med olika kommandon att använda i dessa två CLI:er.

Grundläggande kommandon vid navigering och filhantering.

Kommando	Terminal (macOS)	Git Bash (Windows)	Kommentarer
pwd	Visar aktuell katalog	Samma	Samma beteende
ls	Visar filer och kataloger	Samma	macOS använder BSD ls, Git Bash använder GNU ls
ls -la	Visar alla filer, inklusive dolda	Samma	Formateringen kan skilja sig något
cd katalog	Byter till angiven katalog	Samma	Identisk användning
cd ..	Går upp en nivå i katalogstrukturen	Samma	Identisk användning
cd ~	Går till hemkatalog	Samma	Identisk användning
cd /	Går till rotkatalog	Samma	Rotkatalogen skiljer sig mellan systemen
clear	Rensar terminalfönstret	Samma	Användbart för en ren skärm

Skapa och hantera filer samt kataloger

Kommando	Terminal (macOS)	Git Bash (Windows)	Kommentarer
mkdir katalognamn	Skapar en ny katalog	Samma	Fungerar likadant
touch fil.filtyp	Skapar en tom fil	Samma	Användbart vid testning och skript
rm fil.filtyp	Tar bort en fil	Samma	Ingen ångra-funktion, var försiktig
rm -r katalog/	Tar bort en katalog med innehåll	Samma	-r krävs för att ta bort kataloger
cp fil1 fil2	Kopierar fil	Samma	Använd -r för att kopiera kataloger
mv gammal ny	Flyttar eller byter namn på en fil	Samma	Användbart för både flytt och namnbyte

Visa innehållet i filer

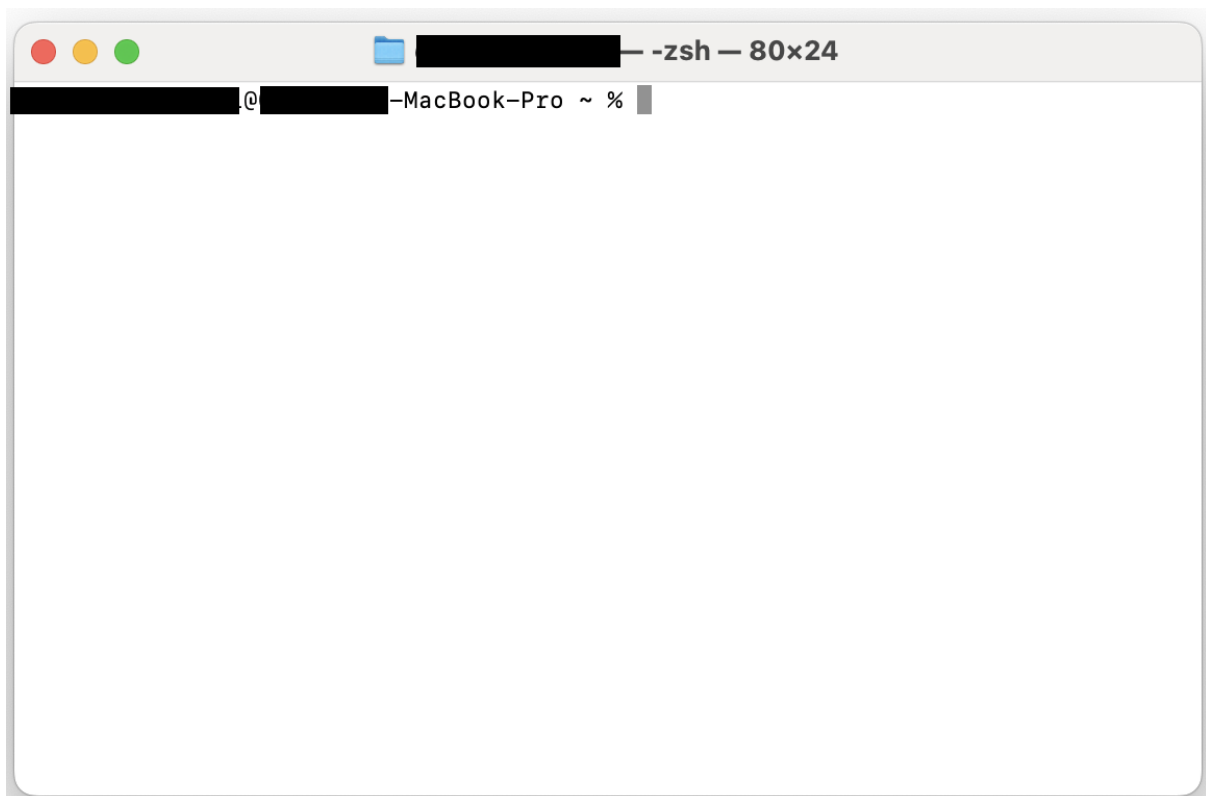
Kommando	Terminal (macOS)	Git Bash (Windows)	Kommentarer
cat fil.txt	Visar innehållet i filen	Samma	Passar bäst för mindre filer
less fil.txt	Visar stora filer med bläddring	Samma	Tryck q för att avsluta
head fil.txt	Visar de första raderna i filen	Samma	Visar som standard 10 rader
tail fil.txt	Visar de sista raderna i filen	Samma	Visar som standard 10 rader

Genvägar och hjälpfunktioner

Kortkommando / Kommando	Terminal (macOS)	Git Bash (Windows)	Kommentarer
cd -	Går till föregående katalog	Samma	Bra för att snabbt växla mellan kataloger
Ctrl + C	Avbryter pågående kommando	Samma	Stoppar kommandot direkt
Ctrl + L	Rensar skärmen	Samma	Samma som clear
Tab (autofyll)	Fyller automatiskt i filnamn	Samma	Snabbar upp arbetet och minskar skrivfel

Navigering i Terminal (Mac)

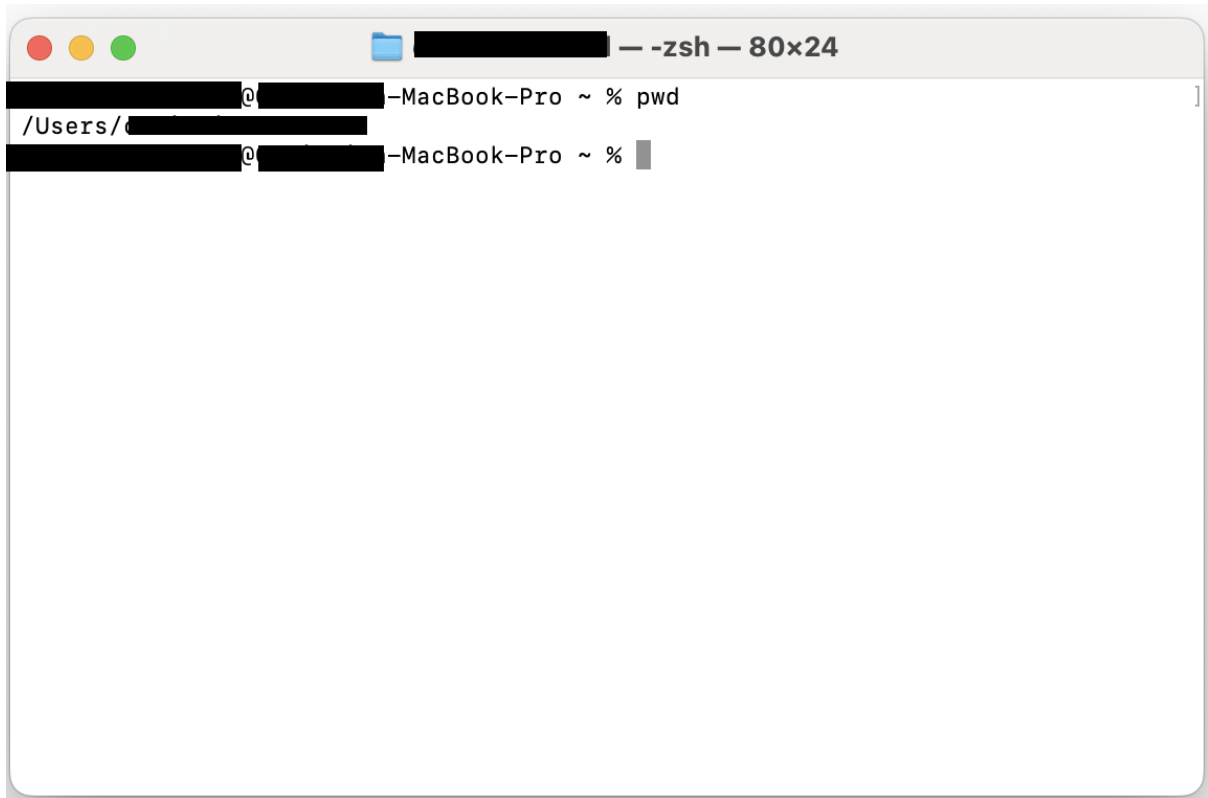
Börja med att öppna terminalen, detta görs genom att söka efter *Terminal* om applikationen inte redan är nålad till Mac Dock. Terminalen öppnas alltid i vad som kallas Hemkatalogen, under denna katalog finns samtliga filer och kataloger tillhörande den inloggade användaren.



Figur 7: Terminal öppnad i hemkatalogen.

När Terminalen öppnas i Hemkatalogen finns det ett stort antal kommandon som kan användas, några av dessa presenterades tidigare i kapitlet. I denna laboration används enklare kommandon, men det är värdefullt att redan nu bekanta sig med fler kommandon än de som behandlas här.

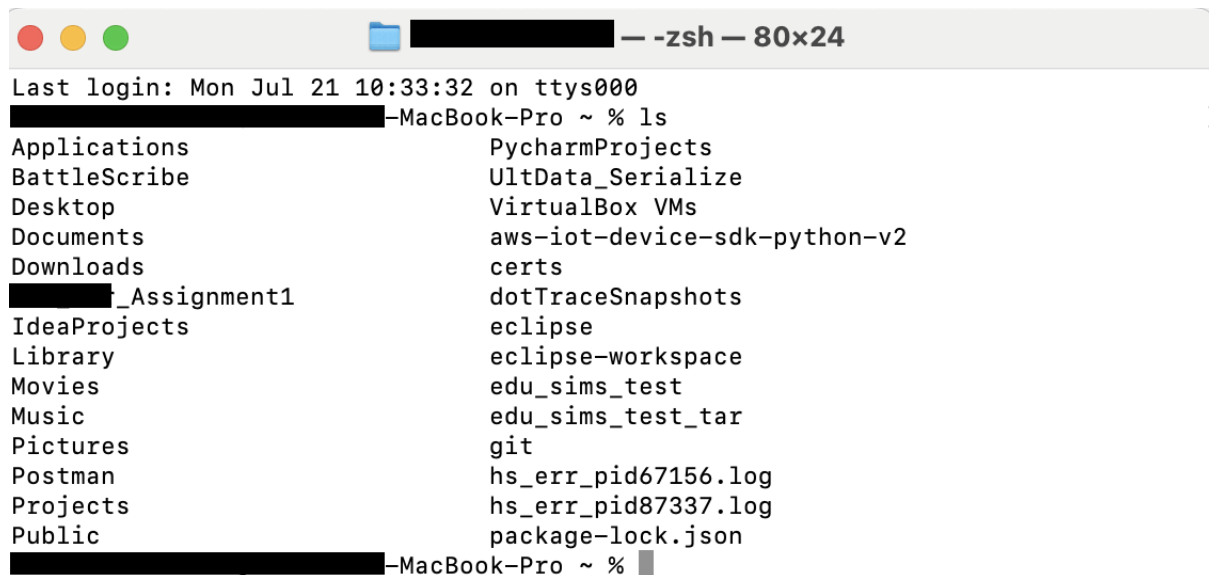
Börja med att skriva kommandot `pwd`. Ta en stund och analysera resultatet. Vad är det som skrivs ut på skärmen när kommandot körs?



Figur 8: Terminalfönstret vilket visar den nuvarande katalogen.

Kommandot visar den nuvarande katalog som Terminalen befinner sig i.

Eftersom Terminalen nyligen startats är denna katalog Hemkatalogen. I denna laboration ska dock större delen av arbetet utföras i katalogen *Dokument*. Detta innebär att den aktuella katalogen i Terminalen behöver ändras dit. Börja därför med att köra kommandot `ls`.



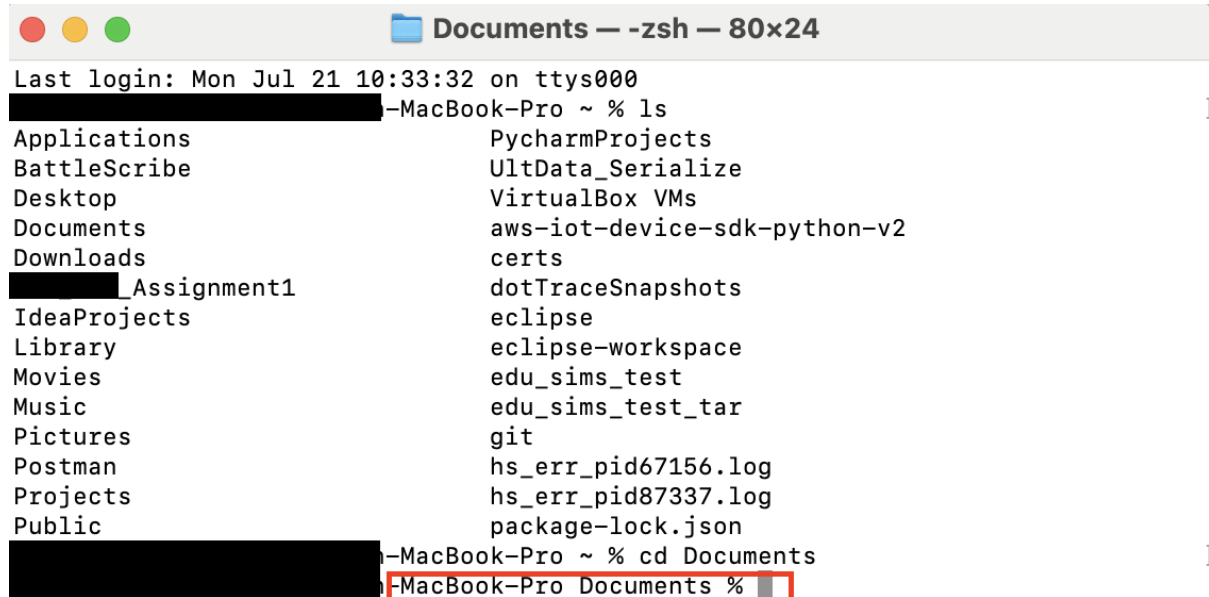
```

Last login: Mon Jul 21 10:33:32 on ttys000
[redacted]-MacBook-Pro ~ % ls
Applications      PycharmProjects
BattleScribe      UltData_Serialize
Desktop           VirtualBox VMs
Documents         aws-iot-device-sdk-python-v2
Downloads         certs
[redacted]_Assignment1 dotTraceSnapshots
IdeaProjects      eclipse
Library           eclipse-workspace
Movies            edu_sims_test
Music             edu_sims_test_tar
Pictures          git
Postman           hs_err_pid67156.log
Projects          hs_err_pid87337.log
Public            package-lock.json
[redacted]-MacBook-Pro ~ %

```

Figur 9: Resultat av att köra kommando `ls`

Genom att köra kommandot `ls` får man en tydlig överblick över var man befinner sig i filsystemet och vilka kataloger som finns tillgängliga att navigera vidare till. Nästa steg är att förflytta sig från Hemkatalogen till den katalog som skapades tidigare, det vill säga *Laborationer* i katalogen *DA339A*. För detta används kommandot `cd`, vilket skrivs följt av ett mellanslag och sedan namnet på den katalog man vill navigera till. I detta fall anges `cd Dokument`.



```

Last login: Mon Jul 21 10:33:32 on ttys000
[redacted]-MacBook-Pro ~ % ls
Applications      PycharmProjects
BattleScribe      UltData_Serialize
Desktop           VirtualBox VMs
Documents         aws-iot-device-sdk-python-v2
Downloads         certs
[redacted]_Assignment1 dotTraceSnapshots
IdeaProjects      eclipse
Library           eclipse-workspace
Movies            edu_sims_test
Music             edu_sims_test_tar
Pictures          git
Postman           hs_err_pid67156.log
Projects          hs_err_pid87337.log
Public            package-lock.json
[redacted]-MacBook-Pro ~ % cd Documents
[redacted]-MacBook-Pro Documents %

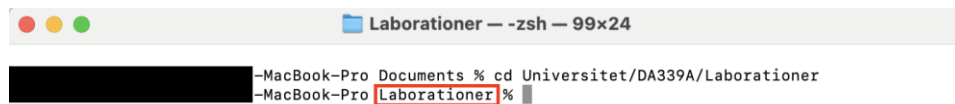
```

Figur 10: Navigering till Dokument med kommando `cd`

Om allt gått som planerat bör Terminalen nu befinna sig i katalogen Dokument och inte längre i Hemkatalogen. Målet är dock att navigera vidare till katalogen *Laborationer* i *DA339A*, som i sin tur ligger under Universitet.

Det är möjligt att upprepa de tidigare stegen och ange varje katalognamn i tur och ordning, men detta kan bli tidskrävande om man behöver navigera genom många nivåer i filsystemet. I stället kan man använda sökvägar för att gå direkt från Dokument till Laborationer.

Återigen används kommandot *cd*, men nu med *snedstreck (/)* för att ange flera kataloger i följd. Skriv helt enkelt det första katalognamnet, följt av /, sedan nästa katalognamn och så vidare, tills den önskade katalogen nås. Försök att navigera till katalogen Laborationer på detta sätt genom att undersöka hur sökvägen för Laborationer ser ut i *Finder*.

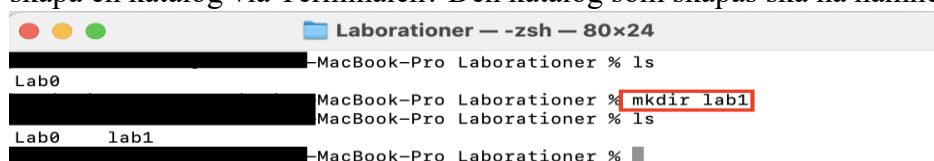


Figur 11: Navigering till Laborationer med kommando *cd*./..

Om detta lyckas bör resultatet bli att Terminalen nu befinner sig i katalogen Laborationer.

Använd kommandot för att lista innehållet i *Laborationer*. Om tidigare laborationer samt denna har följts, finns för närvarande endast katalogen *Lab0* under Laborationer.

Självklart behöver även en katalog skapas för denna laboration, men i stället för att använda något grafiskt gränssnitt ska katalogen skapas via Terminalen. Studera tabellen som introducerades i början av detta avsnitt. Finns det något kommando som kan användas för att skapa en katalog via Terminalen? Den katalog som skapas ska ha namnet *Lab1*.



Figur 12: Katalog skapad med kommando *mkdir*

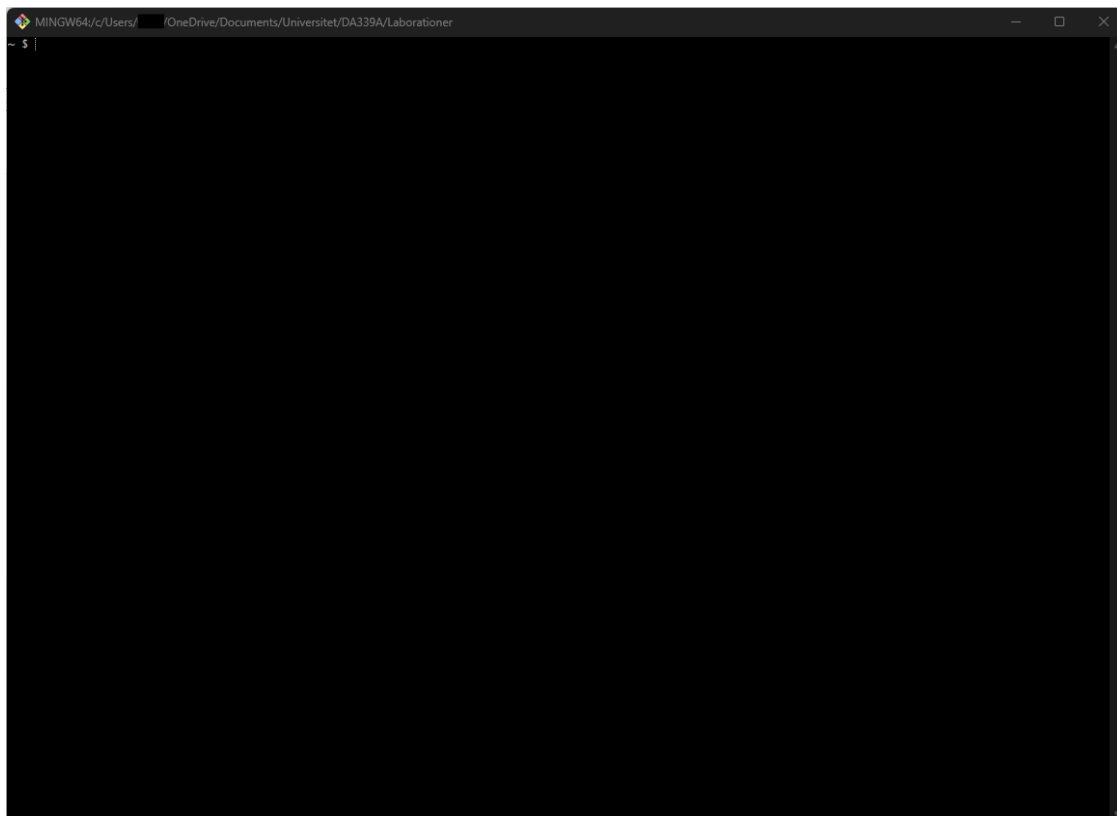
Som bilden visar används kommandot *mkdir* för att skapa nya kataloger.

Navigering i Git Bash (Windows)

Installera Git Bash genom att ladda ner i via länken längst upp i labben. Installeringen av Git Bash kräver inga speciella åtgärder.

När Git Bash är installerat, öppna programmet genom att söka efter det eller klicka på ikonen på skrivbordet om du lagt till den där. Git Bash öppnas i den användarkatalog som tillhör den aktuella användaren på datorn.

OBS! i detta exempel är användarnamnet anonymiserat, men i de flesta fall visas användarens namn till vänster om kommandot i Git Bash.



Figur 13: Git Bash öppnad i hemkatalogen för användaren.

När Git Bash väl öppnas mot hemkatalogen finns det massvis med kommandon vilket kan användas, den tabell som presenterades tidigare i kapitlet innehåller ett antal av dessa. Under denna laboration används enklare kommandon men det är bra att göra sig bekant med flera kommandon än vad som går igenom under denna laboration.

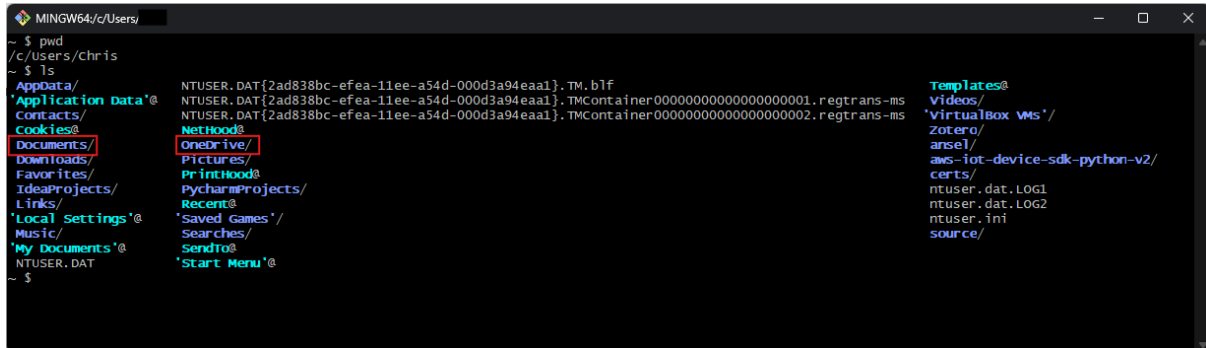
Börja med att skriva kommandot `pwd`, ta en stund och analyser resultatet. Vad är det som skrivs ut på skärmen i samband med att kommandot körs?



Figur 14: Git Bash visar den nuvarande katalogen.

Kommandot visar den nuvarande katalog som Git Bash befinner sig i.

Just nu befinner sig Git Bash i hemkatalogen eftersom programmet nyligen startats. I denna laboration ska dock största mängden av arbetet ske i katalogen Dokument. Detta innebär att den nuvarande katalogen behöver ändras. Börja med att köra kommandot `ls`, ta en stund och analysera resultatet. Vad är det som skrivs ut på skärmen i samband med att kommandot körs?



```

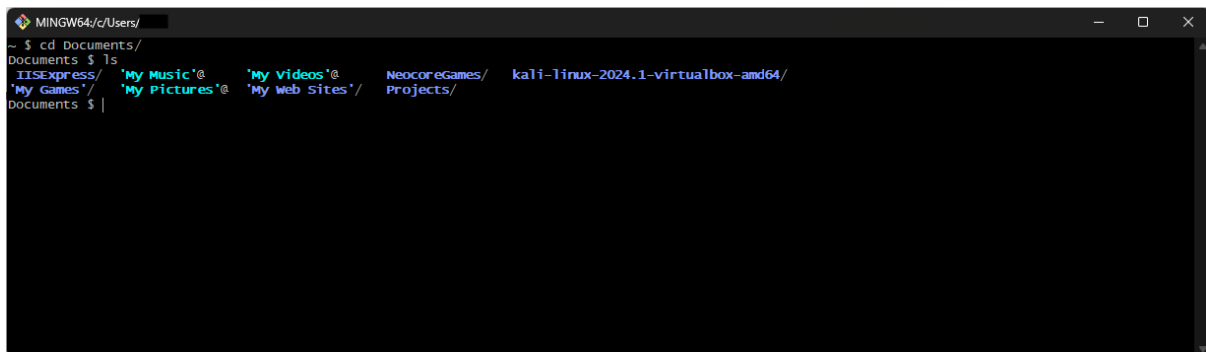
MINGW64/c/Users/chris
~ $ pwd
/c/Users/chris
~ $ ls
AppData/          NTUSER.DAT{2ad838bc-efea-11ee-a54d-000d3a94ea1}.TM.b1f  Templates@
'Application Data'@ NTUSER.DAT{2ad838bc-efea-11ee-a54d-000d3a94ea1}.TMcontainer000000000000000001.regtrans-ms  Videos/
Contacts/         NTUSER.DAT{2ad838bc-efea-11ee-a54d-000d3a94ea1}.TMcontainer000000000000000002.regtrans-ms  'VirtualBox VMs'/
Cookies@          NetHood@          Pictures/          Zotero/
Downloads/        OneDrive/          PrintHood@        'ams-iot-device-sdk-python-v2/'
Favorites/         PycharmProjects/  certs/            ntuser.dat.LOG1
IdeaProjects/      Recent@           'ntuser.dat.LOG2'
Links/             'Saved Games'/    ntuser.ini
'Local Settings'@ Searches/          source/
Music/
'My Documents'@   'Start Menu'@
NTUSER.DAT
~ $

```

Figur 15: Resultat av att köra kommando `ls`

När kommandot `ls` körs får man en tydlig bild av var man befinner sig i filsystemet och vilka mappar som finns tillgängliga att navigera till. I Windows kan katalogen Dokument dock finnas på två olika platser. Den första platsen visas i den markerade röda rutan till vänster, och Dokument finns även under OneDrive, som markerats i den andra röda rutan till höger.

Det enda sättet att vara säker är att navigera till båda platserna och jämföra innehållet. Detta görs med kommandot `cd` (change directory). Börja med att skriva: `cd Dokument`, lista därefter innehållet i Dokument. **OBS! Om katalogen i stället heter Documents, ska detta namn användas i stället.**



```

MINGW64/c/Users/chris
~ $ cd Documents/
Documents $ ls
IISExpress/  'My Music'@  'My Videos'@  NeocoreGames/  kali-linux-2024.1-virtualbox-amd64/
'My Games'/'  'My Pictures'@  'My Web Sites'/'  Projects/
Documents $

```

Figur 16: Innehållet av Dokument under hemkatalogen.

Det är tydligt att de kataloger som arbetats med tidigare inte finns på denna plats, utan OneDrive måste undersökas i hemkatalogen. Just nu befinner sig Git Bash i en annan katalog. Finns det

något kommando som kan användas för att gå tillbaka ett steg? Undersök tabellen ovan och se om ett sådant kommando finns tillgängligt.

OBS! Om de eftersökta katalogerna redan finns i den nuvarande katalogen kan nästa steg antingen hoppas över eller användas som extra övning.

```

MINGW64/c/Users/ /OneDrive/Documents/Universitet/DA339A/Laborationer
~ $ pwd
/c/Users/
~ $ ls
AppData/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TM.b1f Templates
Application Data@ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer000000000000000001.regtrans-ms Videos/
Contacts/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer000000000000000002.regtrans-ms VirtualBox Vms/
Cookies@ NetHood@ Zotero/
Documents/ OneDrive/ ansel/
Downloads/ Pictures/ aws-iot-device-sdk-python-v2/
Favorites/ PrintHood@ certs/
IdeaProjects/ PycharmProjects/ ntuser.dat.LOG1
Links/ Recent@ ntuser.dat.LOG2
Local Settings@ Saved Games/ ntuser.ini
Music/ Searches/ source/
My Documents@ SendTo@
NTUSER.DAT Start Menu@
~ $ cd Documents/
Documents $ ls
IIEExpress/ My Games/ My Music@ My Pictures@ My Videos@ My Web Sites/ NeocoreGames/ Projects/ kali-linux-2024.1-virtualbox-amd64/
Documents $ cd ..
~ $ ls
AppData/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TM.b1f Templates
Application Data@ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer000000000000000001.regtrans-ms Videos/
Contacts/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer000000000000000002.regtrans-ms VirtualBox Vms/
Cookies@ NetHood@ Zotero/
Documents/ OneDrive/ ansel/
Downloads/ Pictures/ aws-iot-device-sdk-python-v2/
Favorites/ PrintHood@ certs/
IdeaProjects/ PycharmProjects/ ntuser.dat.LOG1
Links/ Recent@ ntuser.dat.LOG2
Local Settings@ Saved Games/ ntuser.ini
Music/ Searches/ source/
My Documents@ SendTo@
NTUSER.DAT Start Menu@
~ $

```

Figur 17: Användande av kommando `cd ..` för att backa tillbaka till hemkatalog.

Genom att använda kommandot `cd ..` kan man gå ett steg bakåt i katalogstrukturen åt gången. I detta fall räckte det med att köra `cd ..` en gång för att återgå till hemkatalogen. Det finns även andra kommandon för att navigera tillbaka, om `cd ..` inte användes är det inte fel, det är bara ett sätt att lösa uppgiften på.

Väl tillbaka i hemkatalogen är det dags att utforska katalogen OneDrive. Följ samma steg med `cd` och `ls` för att identifiera om det går att hitta Dokument i OneDrive.

```

MINGW64/c/Users/ /OneDrive/Documents/Universitet/DA339A/Laborationer
~ $ ls
AppData/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TM.b1f Templates
Application Data@ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer000000000000000001.regtrans-ms Videos/
Contacts/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer000000000000000002.regtrans-ms VirtualBox Vms/
Cookies@ NetHood@ Zotero/
Documents/ OneDrive/ ansel/
Downloads/ Pictures/ aws-iot-device-sdk-python-v2/
Favorites/ PrintHood@ certs/
IdeaProjects/ PycharmProjects/ ntuser.dat.LOG1
Links/ Recent@ ntuser.dat.LOG2
Local Settings@ Saved Games/ ntuser.ini
Music/ Searches/ source/
My Documents@ SendTo@
NTUSER.DAT Start Menu@
~ $ cd OneDrive/
OneDrive $ ls
Attachments/ Desktop/ Documents/ Personal Vault.lnk Pictures/ Public/ desktop.ini
OneDrive $ cd Documents/
Documents $ ls
Arduino/ DA339A/ Diablo III/ Larian Studios/ My Games/ Skola/ Universitet/ aw_fb_data.csv
Black Lab Games/ Dell/ Diablo IV/ League of Legends/ Overlord/ StarCraft II/ Visual Studio 2022/ desktop.ini
Custom Office Templates/ Diablo II/ Klei/ MSinfo.txt Rockstar Games/ The Witcher 3/ Zoom/
Documents $

```

Figur 18: Katalogen Universitet hittad under OneDrive/Dokument.

I detta fall kunde katalogen Universitet hittas under Dokument i OneDrive. Nu ska navigation från Dokument till den katalogen vilket skapades tidigare göras, d.v.s. den vilket döptes till *Laborationer* i katalogen *DA339A*.

Det är möjligt att upprepa de tidigare stegen och ange varje katalognamn ett i taget, men detta kan bli tidskrävande om man behöver navigera genom många nivåer i filsystemet. I stället kan man använda sökvägar för att gå direkt från *Dokument* till *Laborationer*.

Återigen används kommandot *cd*, men nu med *snedstreck (/)* för att ange flera kataloger i följd. Skriv helt enkelt det första katalognamnet, följt av */*, sedan nästa katalognamn och så vidare, tills den önskade katalogen nås. Försök att navigera till katalogen *Laborationer* på detta sätt genom att undersöka hur sökvägen för *Laborationer* ser ut i filutforskaren. OBS! inga mellanrum får förekomma i sökvägen.

```

MINGW64/c/Users/ /OneDrive/Documents/Universitet/DA339A/Laborationer
~ $ ls
AppData/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TM.b1f
Application Data@ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer00000000000000000001.regtrans-ms
Contacts/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer00000000000000000002.regtrans-ms
Cookies@ NetHood@
Documents/ OneDrive/
Downloads/ Pictures/
Favorites/ PrintHood@
IdeaProjects/ PycharmProjects/
Links/ Recent@
Local Settings@ Saved Games/
Music/ Searches/
My Documents@ SendTo@
NTUSER.DAT Start Menu@
~ $ cd OneDrive/
OneDrive $ ls
Attachments/ Desktop/ Documents/ Personal Vault.lnk* Pictures/ Public/ desktop.ini
Documents $ ls
Arduino/ DA339A/ Diablo III/ Larian Studios/ My Games/ Skola/ Universitet/ aw_fb_data.csv
Black Lab Games/ Dell/ Diablo IV/ League of Legends/ Overlord/ StarCraft II/ Visual Studio 2022/
Custom Office Templates/ Diablo II/ Klei/ MSInfo.txt Rockstar Games/ The Witcher 3/ Zoom/
Documents $ cd Universitet/DA339A/Laborationer/
Laborationer $ ls
Lab0/
Laborationer $

```

Figur 19: Katalogen *Laborationer* listad.

Som det går att se ovan användes kommandot *cd Universitet/DA339A/Laborationer* i katalogen *Dokument* för att hoppa direkt till katalogen *Laborationer*. Om man vet var man ska och har koll på sökvägen kan man redan från hemkatalogen skriva *cd* följt av hela sökvägen.

Självklart behöver det även skapas en katalog för denna laboration, men i stället för att använda något grafiskt gränssnitt ska katalogen skapas via *Git Bash*. Studera tabellen som introducerades i början av detta avsnitt, finns det något kommando som kan användas för att skapa en katalog via *Git Bash*? Skapa katalogen *Lab1*.

```

MINGW64/c/Users/ /OneDrive/Documents/Universitet/DA339A/Laborationer
~ $ ls
AppData/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TM.b1f
Application Data@ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer00000000000000000001.regtrans-ms
Contacts/ NTUSER.DAT[2ad838bc-efea-11ee-a54d-000d3a94eaa1].TMContainer00000000000000000002.regtrans-ms
Cookies@ NetHood@
Documents/ OneDrive/
Downloads/ Pictures/
Favorites/ PrintHood@
IdeaProjects/ PycharmProjects/
Links/ Recent@
Local Settings@ Saved Games/
Music/ Searches/
My Documents@ SendTo@
NTUSER.DAT Start Menu@
~ $ cd OneDrive/
OneDrive $ ls
Attachments/ Desktop/ Documents/ Personal Vault.lnk* Pictures/ Public/ desktop.ini
Documents $ ls
Arduino/ DA339A/ Diablo III/ Larian Studios/ My Games/ Skola/ Universitet/ aw_fb_data.csv
Black Lab Games/ Dell/ Diablo IV/ League of Legends/ Overlord/ StarCraft II/ Visual Studio 2022/
Custom Office Templates/ Diablo II/ Klei/ MSInfo.txt Rockstar Games/ The Witcher 3/ Zoom/
Documents $ cd Universitet/DA339A/Laborationer/
Laborationer $ ls
Lab0/
Laborationer $ mkdir Lab1
Laborationer $ ls
Lab0/ Lab1/
Laborationer $

```

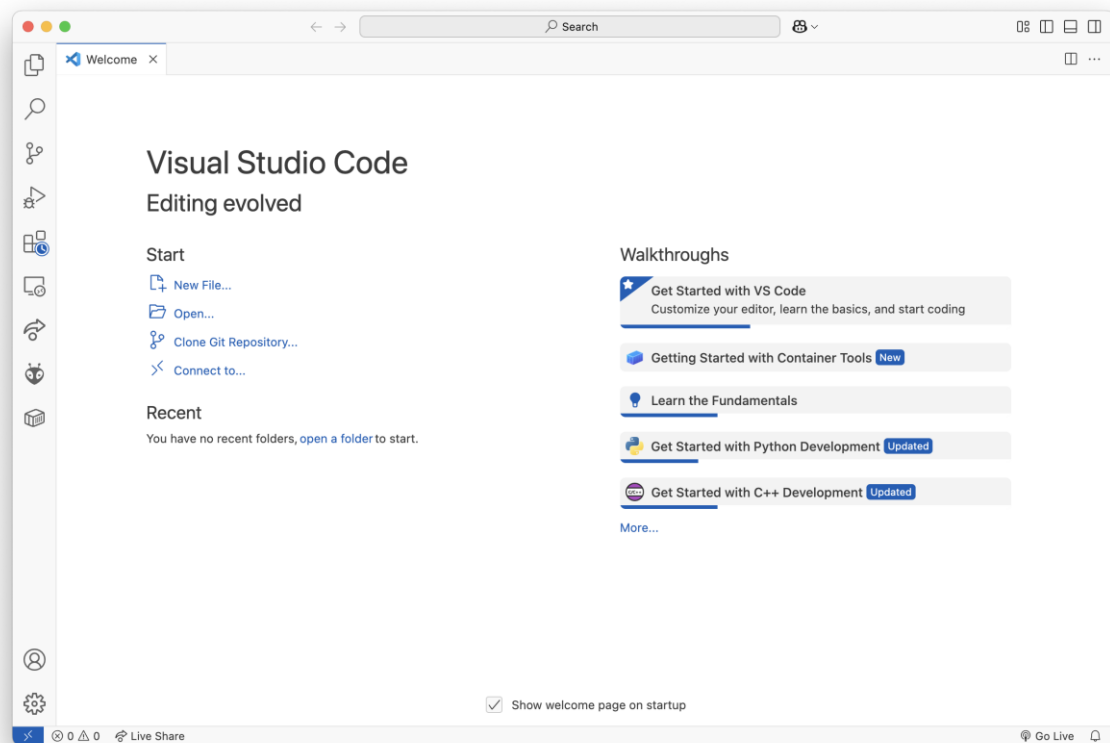
Figur 20: Katalogen *Lab1* skapad med kommando *mkdir* följt av önskat namn.

Som bilden ovan visar används kommandot `mkdir` för att skapa kataloger. I detta fall var det specifikt `mkdir Lab1`

Visual Studio Code

Denna del av laborationen ger en grundläggande genomgång av textredigeraren Visual Studio Code (VSC), som kommer att användas i de kommande labbarna för att skriva exekverbar kod i programmeringsspråket Java.

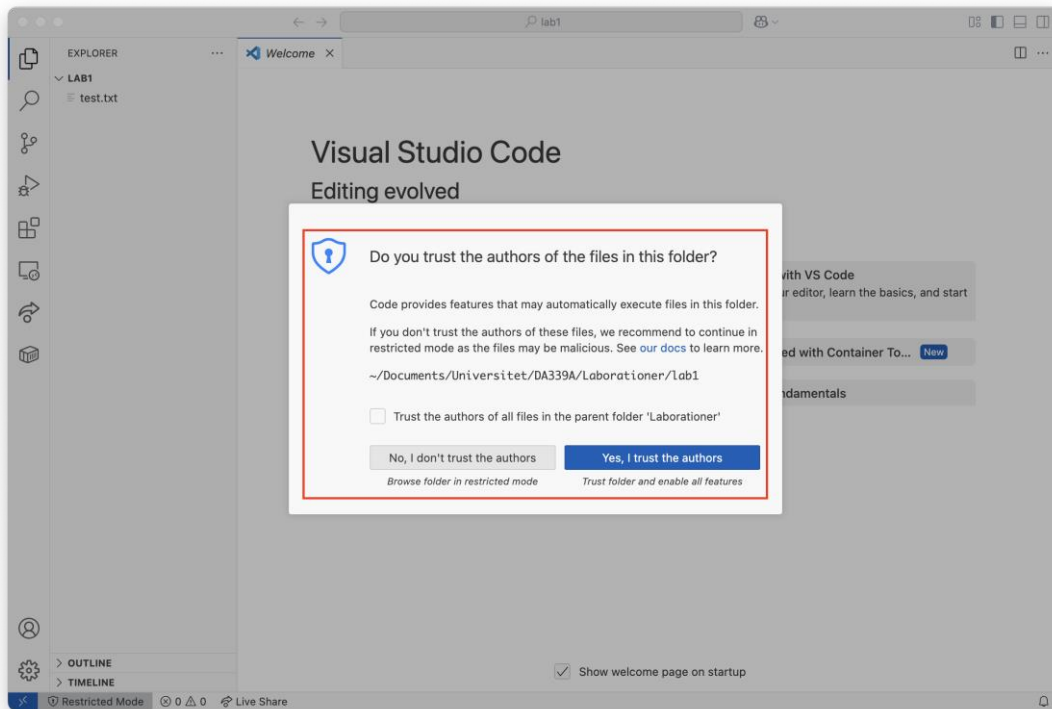
När VSC öppnas syns följande vy. Här går det snabbt att påbörja en ny fil, öppna tidigare skapade filer, använda integration mot Git samt se de filer som senast arbetats med. I VSC kan arbetet ske direkt i en katalog. Detta görs genom att klicka på *Open...* under *Start*. När detta är gjort, navigera till katalogen `Lab1` och öppna den.



Figur 21: Huvudvyn i Visual Studio Code.

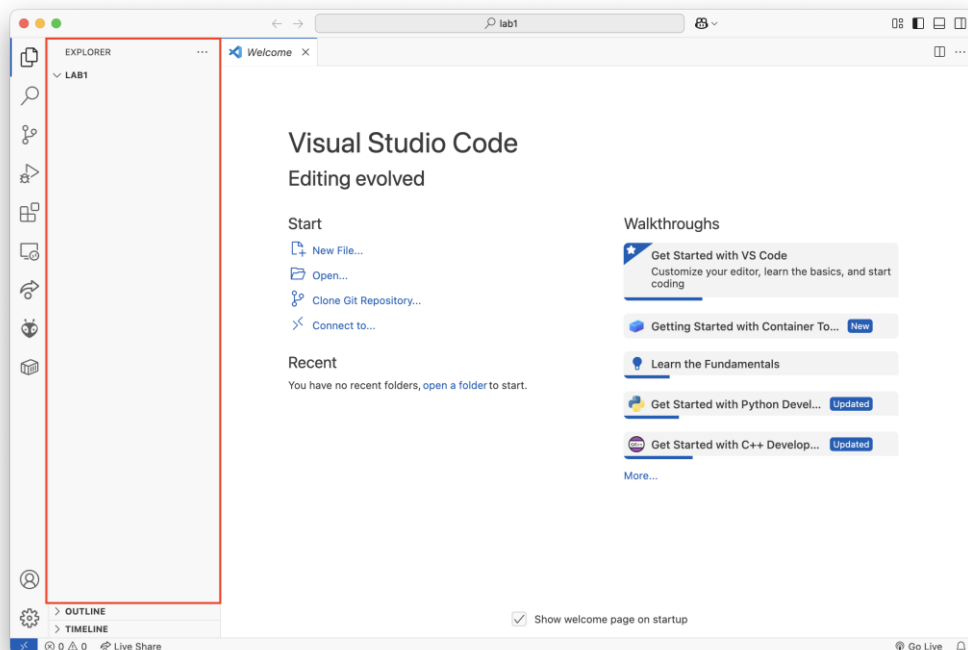
Det är vanligt att ett meddelande dyker upp som frågar om det är säkert att arbeta med filer i den valda katalogen. Det är viktigt att redan nu vänja sig vid att noggrant läsa och granska denna typ av meddelanden. Eftersom användaren i detta fall är skaparen av katalogen kan man ofta

anta att det är säkert att godkänna, men det är alltid bra att vara medveten om vad meddelandet egentligen frågar.



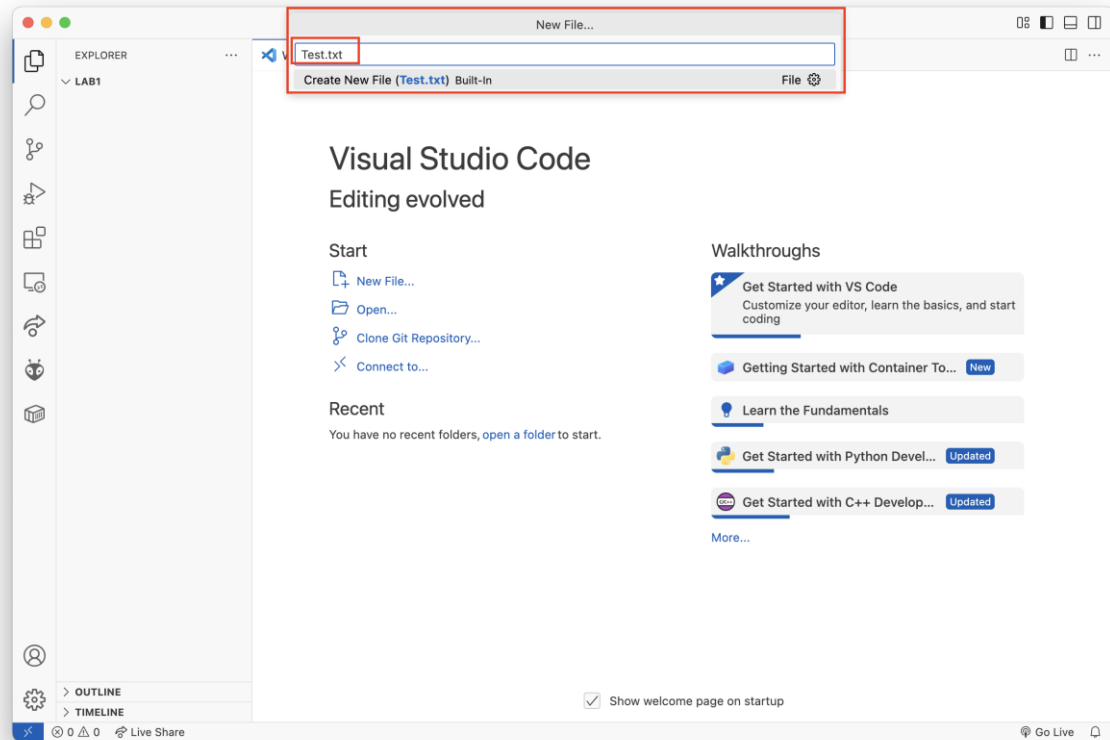
Figur 22: Fråga från VSC ifall författarna kan lita på.

När meddelandet har godkänts öppnar VSC katalogen. Det kanske inte märks omedelbart, men ett nytt fält med namnet Explorer visas. Under detta fält listas alla filer som finns i den öppnade katalogen. I just detta fall kommer fältet att vara tomt, eftersom inga filer ännu finns i katalogen.



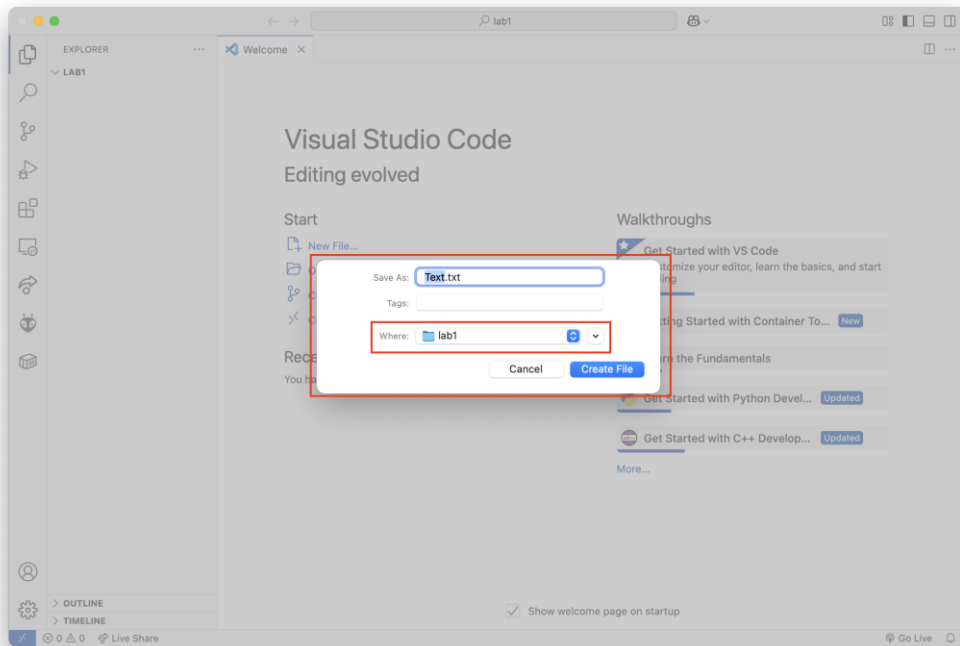
Figur 23: Explorer-rutan öppnad efter använt "Open...".

För att skapa en ny fil behöver man bara klicka på *New File...*, vilket även finns under *Start*. När *New File...* klickas på öppnas en liten ruta ovanför, där *filnamn* och *typ* anges. I detta fall kan det exempelvis skriva *Test.txt*. När namnet har angetts, tryck på *Enter* på tangentbordet.



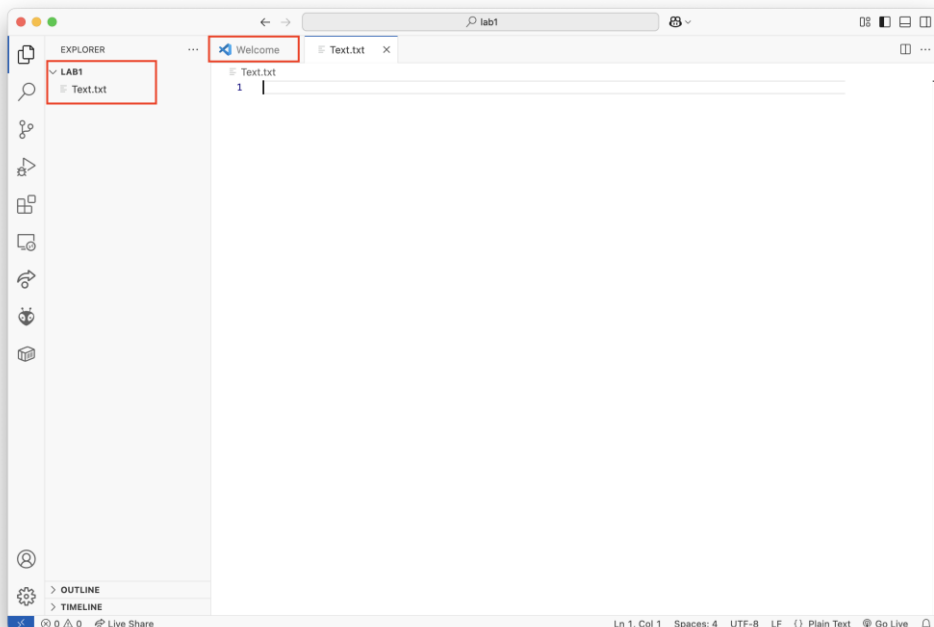
Figur 24: Namn- och typvalet av ny fil.

I den nya rutan som öppnas syns återigen filnamn och filtyp. Det går också att se och ändra platsen där filen ska skapas, men i detta fall klicka enbart på knappen *Create File*.



Figur 25: Skapandet av ny fil.

Nu går den nya filen att se under Lab1 i fältet Explorer. Observera att även den nyskapade filen öppnades direkt och nu är den aktiva filen som arbetas i. Det går också att återgå till fältet som syntes tidigare genom att klicka på fliken Welcome.



Figur 26: Nyskapad fil i VSC.

OBS! att ändringar som görs i ett dokument i VSC aldrig sparas automatiskt om detta inte sker manuellt, fram tills dess att automatisk sparning aktiveras.

Kommandon i Visual Studio Code

Kommando	Kortkommando (Windows/Linux)	Kortkommando (Mac)	Beskrivning
Indentera kod Öppna fil	Shift + alt + F Ctrl + O	Shift + option + F Cmd + O	Indenterar kod automatiskt Öppna en enskild fil
Öppna katalog	Ctrl + K, Ctrl + O	Cmd + K, Cmd + O	Öppna en hel katalog
Spara fil	Ctrl + S	Cmd + S	Spara aktuell fil
Spara alla filer	Ctrl + K, S	Cmd + Option + S	Spara alla öppna filer
Stäng fil	Ctrl + W	Cmd + W	Stäng aktuell flik
Dela redigerare (sida vid sida)	Ctrl + \	Cmd + \	Visa två filer bredvid varandra
Växla mellan flikar	Ctrl + Tab	Cmd + Tab	Byt mellan öppna filer
Sök i fil	Ctrl + F	Cmd + F	Sök efter text i aktuell fil
Ersätt i fil	Ctrl + H	Cmd + H	Ersätt text i aktuell fil
Sök i projekt	Ctrl + Shift + F	Cmd + Shift + F	Sök i alla filer i en katalog
Visa terminal	Ctrl + `	Cmd + `	Öppna/stäng terminal i VS Code
Ny terminal	Ctrl + Shift + `	Cmd + Shift + `	Öppna en ny terminalflik
Kommentera rad	Ctrl + /	Cmd + /	Kommentera/avkommentera vald rad (//)
Flytta rad upp/ner	Alt + ↑ / ↓	Option + ↑ / ↓	Flytta raden uppåt eller nedåt
Duplicera rad	Shift + Alt + ↑ / ↓	Shift + Option + ↑ / ↓	Kopiera raden uppåt eller nedåt
Ångra / Gör om	Ctrl + Z / Y	Cmd + Z / Shift + Cmd + Z	Ångra eller gör om ändring
Visa/dölj sidopanel	Ctrl + B	Cmd + B	Visa eller dölj filutforskaren
Zen-läge (fokuserat läge)	Ctrl + K Z	Cmd + K Z	Fokusera bara på koden (inga paneler)
Kommandopalett	Ctrl + Shift + P	Cmd + Shift + P	Sök bland alla kommandon i VS Code

Versionshantering (Git)

Lokal Git-användning

I denna laboration kommer det att användas lokal Git-hantering med hjälp av Visual Studio Code. Till att börja med behöver git installeras lokalt på datorn, följ nedan guide beroende av operativsystem:

Mac

Det bästa sättet att installera Git på är med verktyget Homebrew, som är en pakethanterare för macOS. Gör följande steg:

1. Öppna upp ett terminalfönster och kör följande kommando: `$ /bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install.sh)"`
2. `$ brew doctor`
3. Homebrew kommer nu att installeras tillsammans med några trevliga utvecklingsverktyg. Låt det köra klart, och kör sedan `$ brew install git`

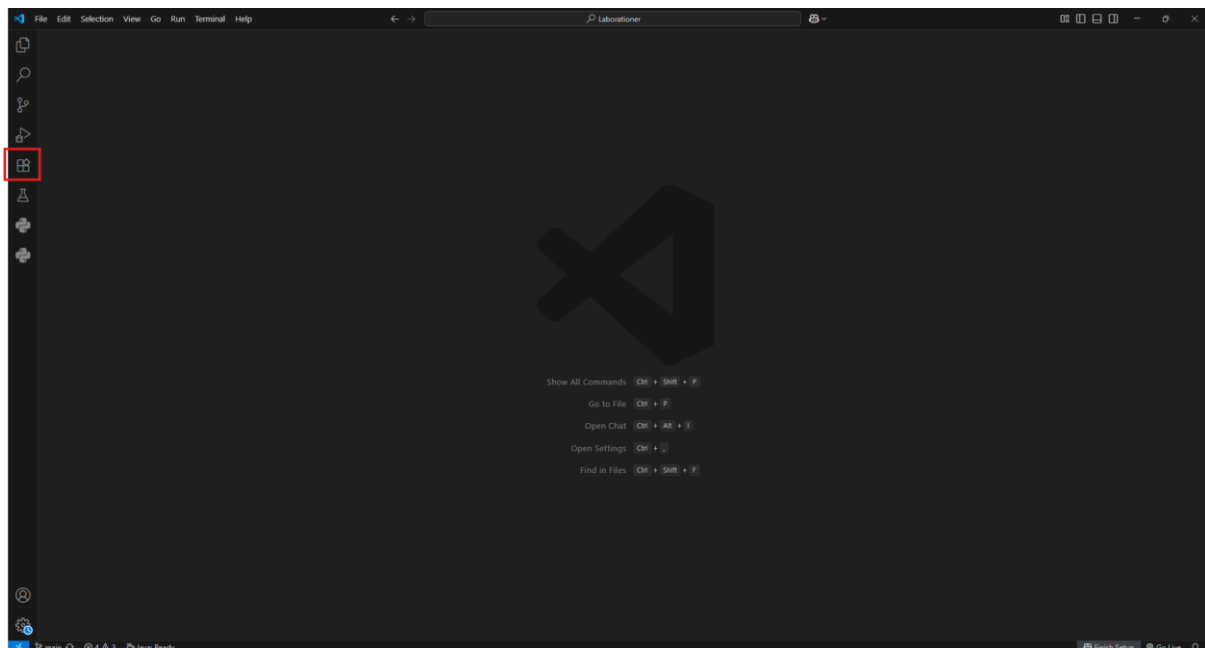
Går det inte att skriva instruktionerna börja med att ta bort dessa tecken \$.

Windows

För Windows är det relativt lätt att sätta upp Git, ladda ned följande fil och kör den sedan enligt instruktioner på skärmen: <https://git-scm.com/downloads/win>

Git History

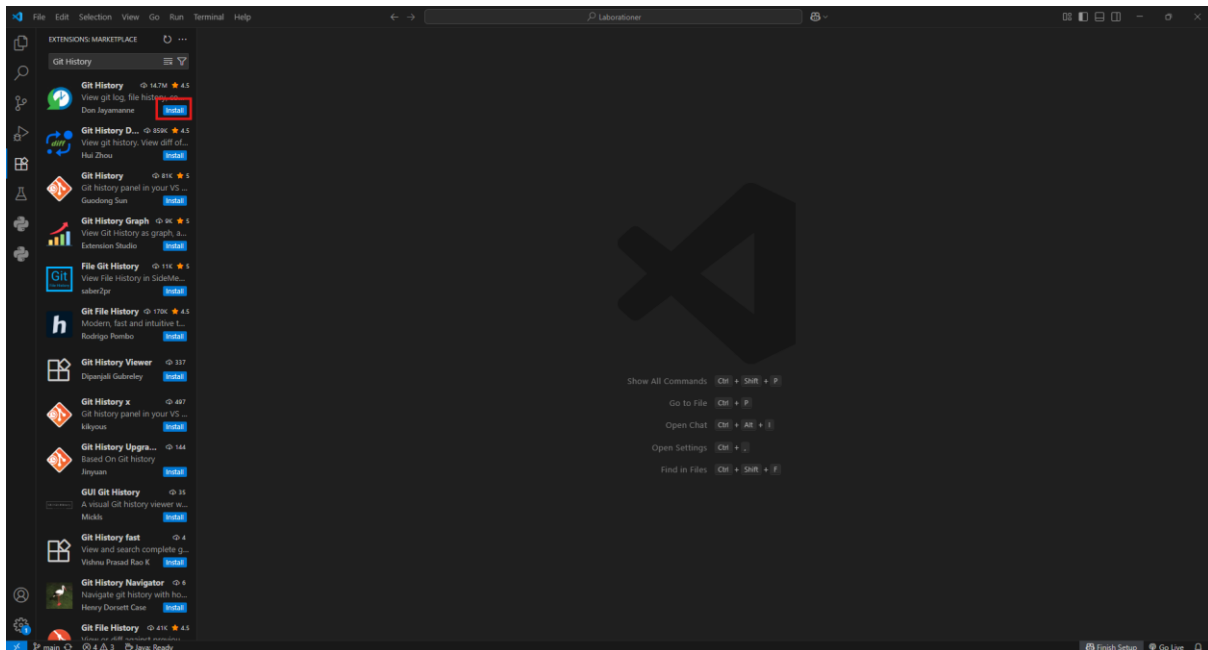
Nästa steg är att lägga till Git History som ett tillägg i VS Code. Börja med att öppna VS Code och klicka därefter på ikonen som ser ut som *tre staplade lådor med en låda ovanför*.



Figur 27: Ikonen för att öppna tillägg.

När denna knapp klickas på öppnas ett nytt fönster. I detta fönster går det att se de redan installerade tilläggen i VS Code samt lägga till nya. Längst upp i fönstret, precis under

Extensions, finns en sökruta. Sök här efter *Git History*. När resultatet visas, klicka på knappen *Install* för att lägga till tillägget.

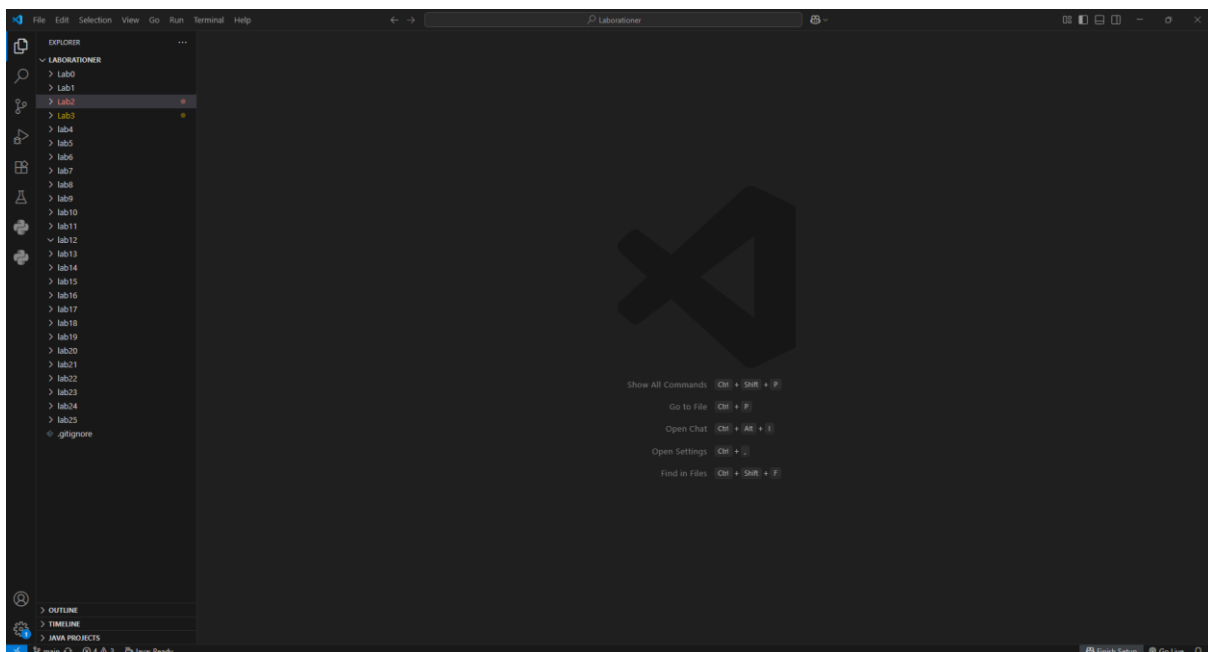


Figur 28: *Git History* tillagt genom tillägg.

Med detta klart går det nu att påbörja nästa del av labben.

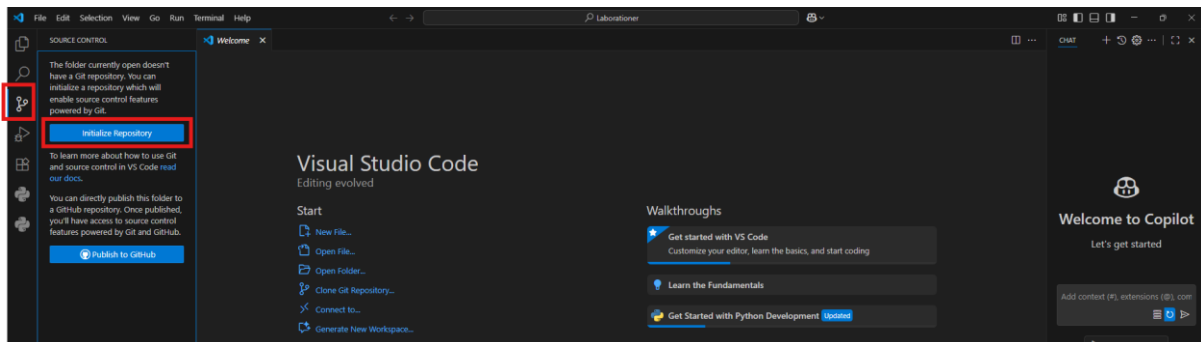
Skapa ett nytt repository

I regel är det en bra idé att samla projekt som hör ihop på ett ställe; i detta fall är det katalogen *Laborationer* i *DA339A*. Börja med att öppna katalogen *Laborationer* i VS Code. I utforskaren till vänster bör alla skapade kataloger för kommande laborationer gå att se.



Figur 29: *Laborationer* öppnad i VS Code (OBS! uppgift för att skapa alla labbar kommer senare).

När detta är gjort ska det första Git-repositoriet (repo) skapas. Detta görs genom att klicka på ikonen för versionshantering (ser ut som tunnelbanelinjer) och därefter klicka på knappen *Initialize Repository*



Figur 30: Knappen för versionshantering tryckt på och initialize repository visas.

Direkt efter detta visas de senaste ändringarna i repot, där alla ändringar som gjorts i filer syns. Eftersom det just nu kan finnas filer i dessa kataloger listas de under *Changes*. Bortse från dessa för tillfället och återgå till projektvyen genom att klicka på *Explorer-ikonen* (ser ut som två dokument på högkant). Hanteringen av ändringar kommer snart att tas upp, men för att få en bättre förståelse ska några ytterligare filer skapas.

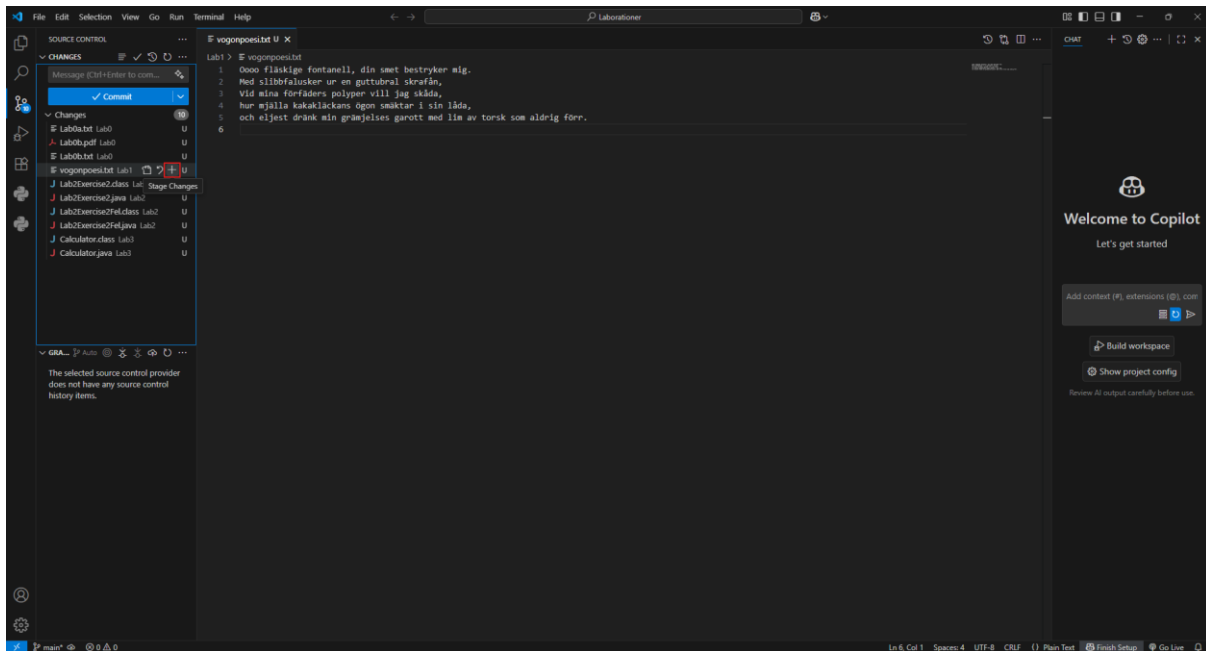
Skapa en fil och versionshantera filen

För att lära lite grundläggande versionshantering ska det först skapas en fil i katalogen *Lab0*, döp denna till *vogonpoesi*, filen ska dessutom vara en textfil.

Taget från en mästare i git är det nu dags att lära ut versionshantering genom dålig poesi, i den nyskapade filen kopiera in följande ”mästerverk”:

*Oooo fläskige fontanell, din smet bestryker mig.
Med slibbfalusker ur en guttubral skrafån,
Vid mina förfäders polyper vill jag skåda,
hur mjälla kakakläckans ögon smäktar i sin låda,
och eljest dränk min grämjelses garott med lim av torsk som aldrig förr.*

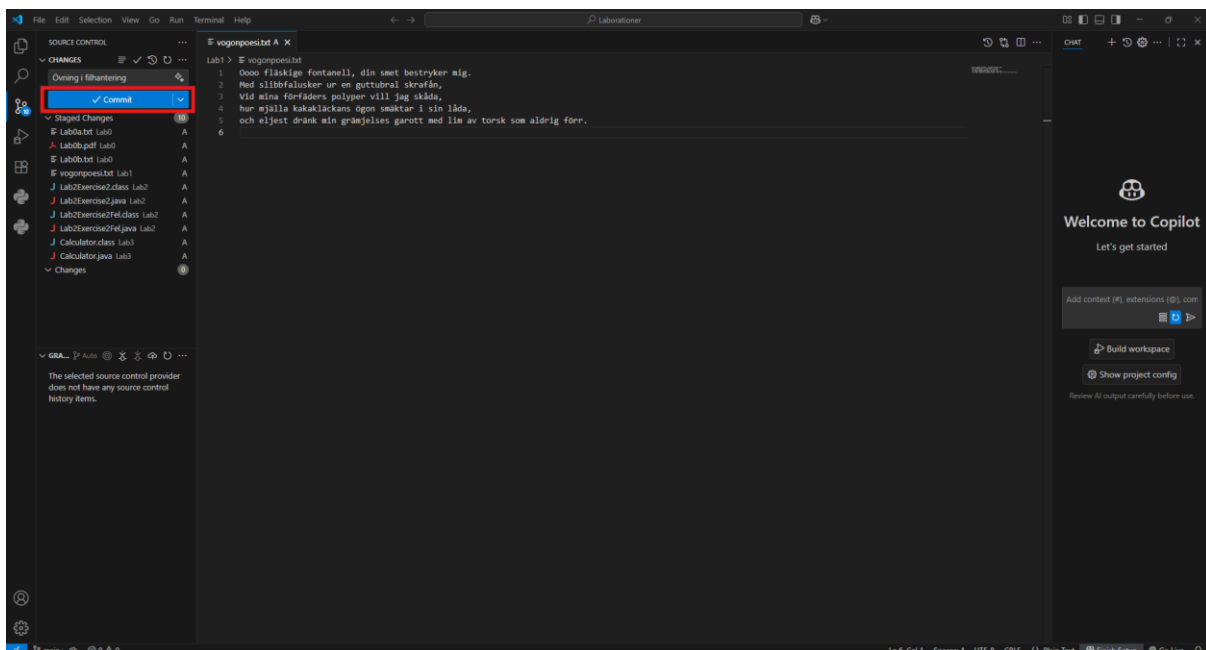
Som Johan själv uttrycker det är vogonpoesi verkligen fruktansvärt. Men med filen skapad gå tillbaka till versionshantering, filen vogonpoesi bör nu synas under *Changes*. För att indikera till Git att filen ska versionhanteras tryck på *plustecknet* som finns utmed *U*. **OBS! Om filen inte syns i Changes, spara i så fall filen manuellt först.**



Figur 31: Tryck på plustecknet utmed en fil för att påbörja versionshantering.

Genom att klicka på *plustecknet* intill *Changes* läggs alla nyligen ändrade filer i katalogen till för versionshantering. När detta är gjort bör flera filer finnas redo att checkas in, vilket kan observeras under *Staged Changes*.

Slutligen ska filerna checkas in, så att de sparas som en revision. Fyll i ett commit-meddelande – tänk på att meddelanden bör vara deskriptiva över det arbete som gjorts, korta och koncisa. Ange ett lämpligt commit-meddelande och klicka sedan på knappen med *en bock och Commit*.



Figur 32: Kort commit-meddelande redo att checkas in till det lokala repot.

Skapa en fil och undvik att versionshantera filen

I vissa fall ska filer inte versionhanteras. Det kan bero på att de är konfigurationsfiler med inloggningsuppgifter, onödiga filer, eller filer som operativsystem eller IDE skapar (exempelvis .DS_Store för macOS eller .iml-filer för IntelliJ).

I detta fall ska det dock skrivas lite poesi igen, men filen ska inte sparas i Git denna gång. Skapa återigen en ny fil i Lab1 och döp den till *boye*, av typen text. Kopiera sedan in följande innehåll:

*Den mätta dagen, den är aldrig störst.
Den bästa dagen är en dag av törst.*

*Nog finns det mål och mening i vår färd -
men det är vägen, som är mödan värd.*

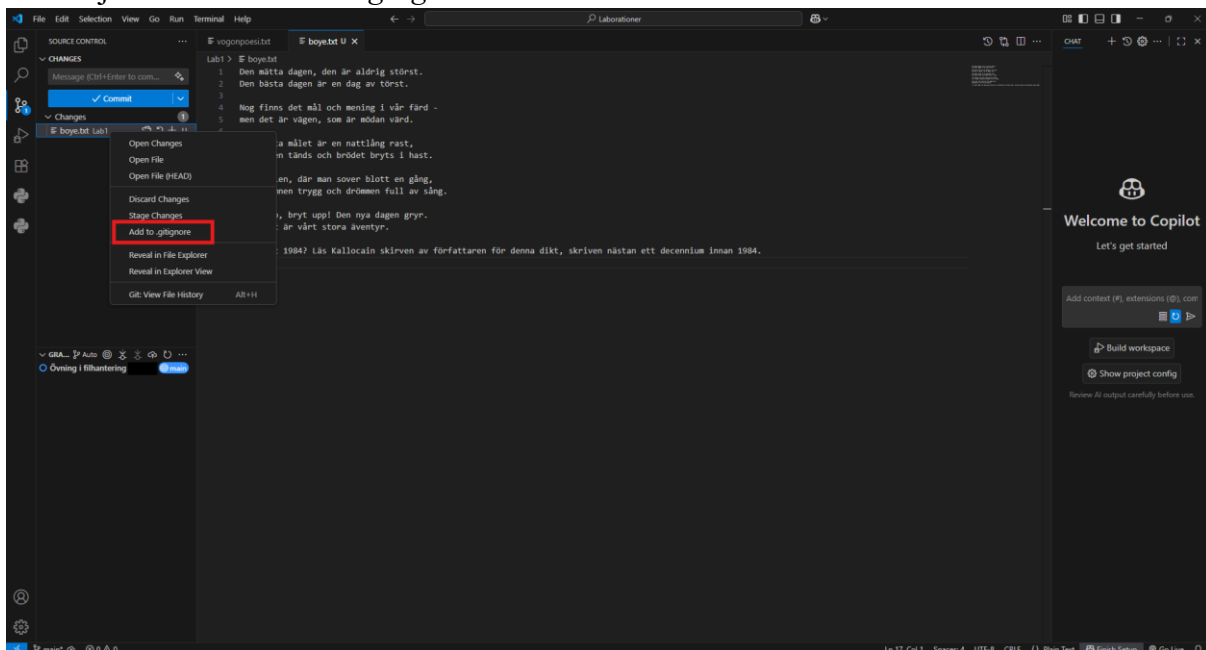
*Det bästa målet är en nattlång rast,
där elden tänds och brödet bryts i hast.*

*På ställen, där man sover blott en gång,
blir sömnen trygg och drömmen full av sång.*

*Bryt upp, bryt upp! Den nya dagen gryr.
Oändligt är vårt stora äventyr.*

*P.S Läst 1984? Läs Kallocain skriven av författaren för denna
dikt, skriven nästan ett decennium innan 1984.*

Som nämnts tidigare är detta inte något som ska versionhanteras. Gå återigen till *versionshantering*, den nyskapade filen bör nu dyka upp under *Changes*. Högerklicka på filen och välj alternativet *Add to .gitignore*.



Figur 33: Nyskapade filen som läggs till .gitignore.

Om filen `.gitignore` öppnas syns att `Lab1/boye.txt` finns listad i filen. *Checka in* den nyskapade filen `.gitignore` på samma sätt som de tidigare filerna.

Filer som inte ska versionhanteras läggs alltså till i filen `”.gitignore”`, vilket fungerar som en instruktion för Git. Alla filer som listas i en `”.gitignore”` flaggas aldrig av Git som nyligen ändrade, eftersom Git inte bryr sig om dessa filer i versionshanteringen.

Uppdatera en fil

Det är nu skapade tre filer, en som versionhanteras, en som ignoreras och en som håller reda på vilka filer som ignoreras. Nu ska filen **vogonpoesi.txt** ändras. Öppna därför filen och lägg till följande rad som sista rad i texten:

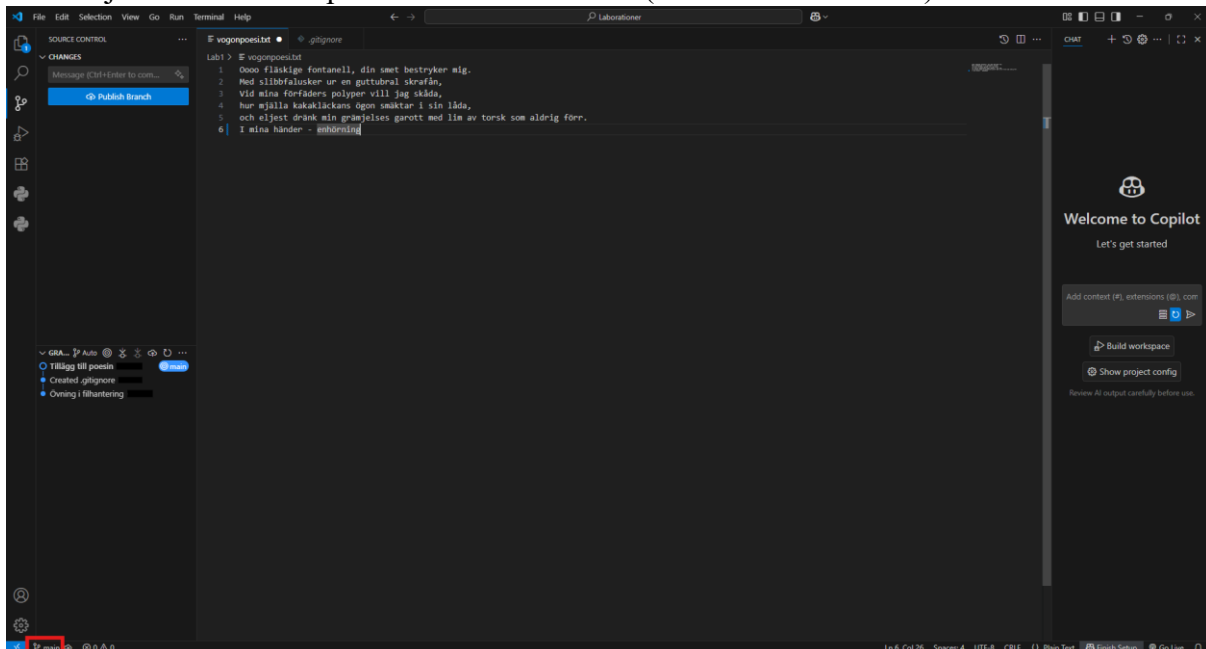
I mina händer – enhörning

Det är viktigt att notera att det egentligen inte spelar någon roll vilken ändring som görs här, så länge dokumentet förändras. Exempelvis genom att ta bort en rad, lägga till en rad eller ändra på en rad.

Gå återigen till versionshantering. Filerna `vogonpoesi.txt` bör nu dyka upp under *Changes*. *Checka in* förändringen på samma sätt som tidigare, med ett lämpligt commit-meddelande.

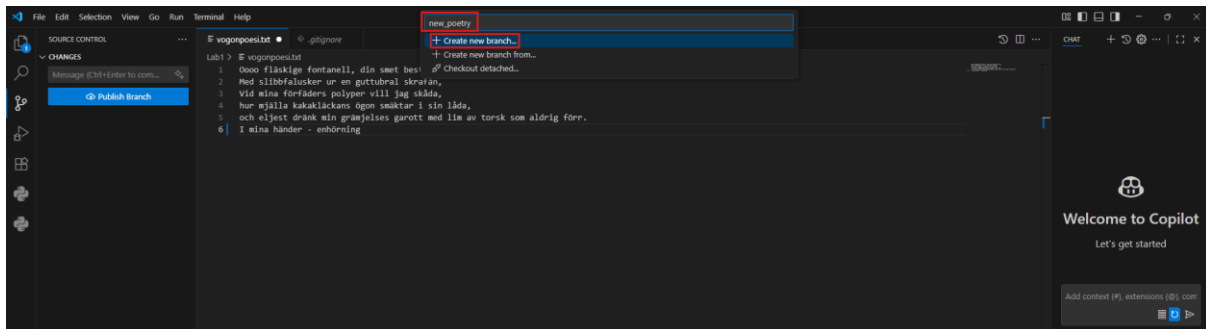
Jobba med grenar

Git är ett utmärkt verktyg för att hantera parallellt arbete. När detta görs kallas det att man grenar eller skapar branches. Den kommande delen fokuserar på hur arbetet i två grenar går till. Börja med att klicka på `main` nere till vänster (kan även heta `master`).



Figur 34: Grenen `main` inringad nere till vänster.

I rutan som dyker upp längst upp, ange ett lämpligt namn på den nya grenen och klicka sedan på *Create New Branch...*. Observera att den nuvarande grenen automatiskt bytte till den nyskapade.

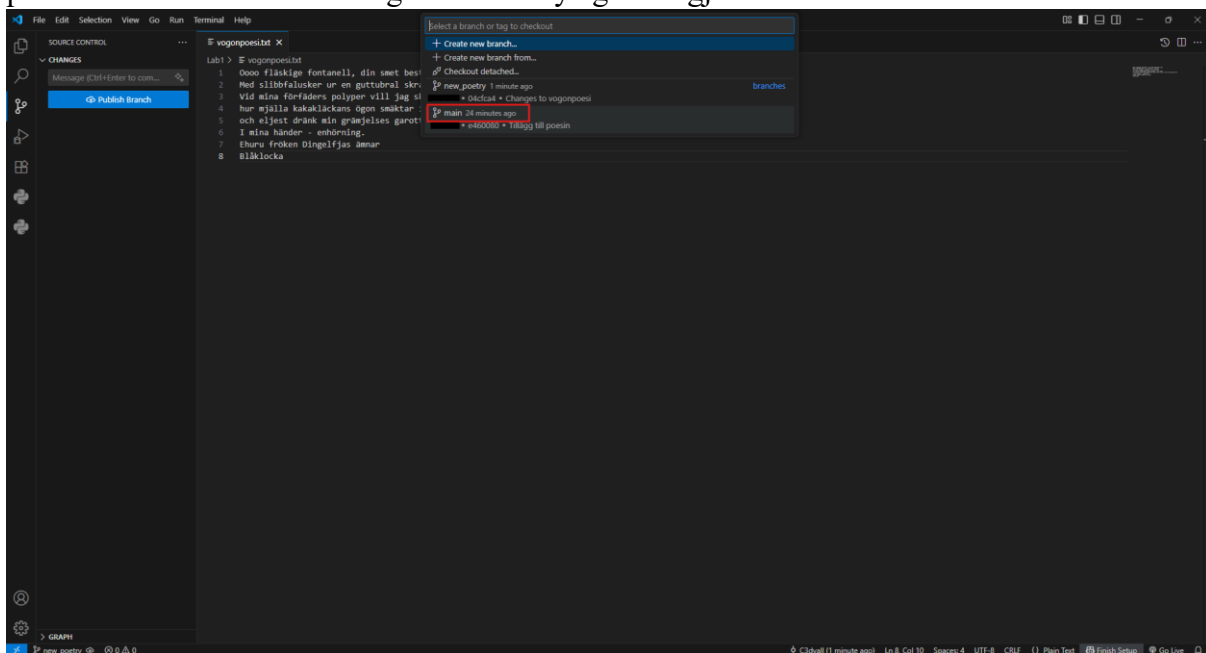


Figur 35: Namngivning och skapande av ny gren.

Med två grenar skapade, öppna återigen *vagonpoesi.txt* och kontrollera att det är den nyskapade grenen som arbetas i. Därefter lägg till ytterligare text:

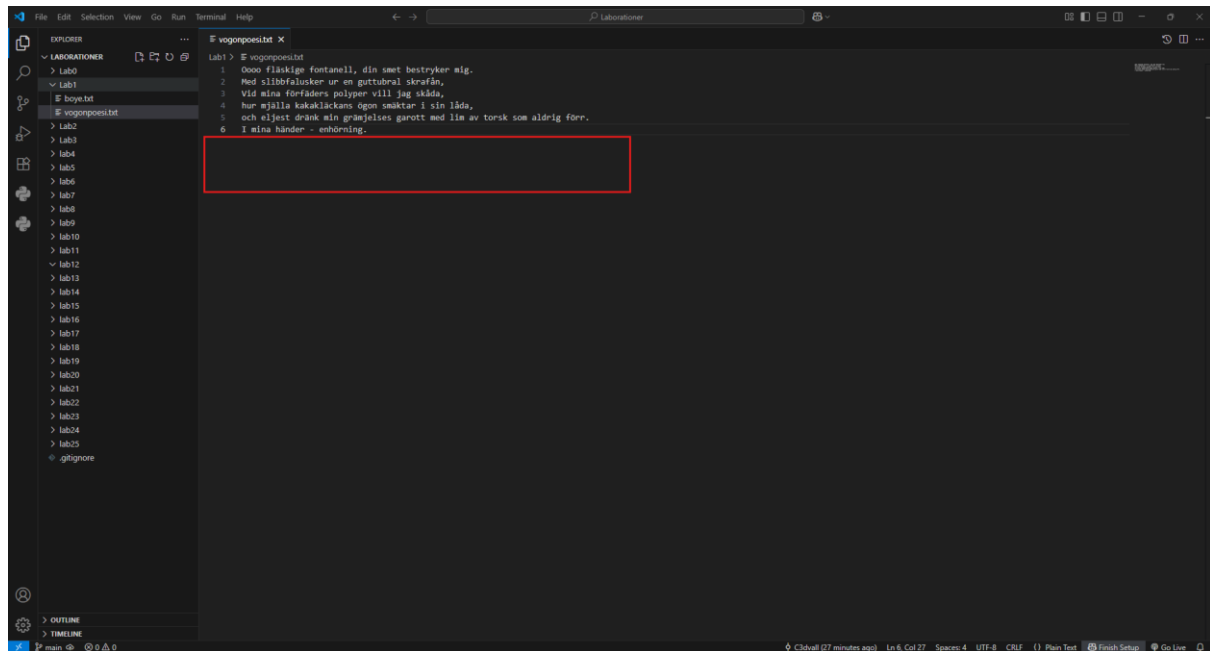
*Ehuru fröken Dingelfjas ämnar
Blålocka*

Gå till *versionshantering* och lägg till filen genom att klicka på *plustecknet*. Ange därefter ett lämpligt *commit-meddelande* och *checka in* filen. Växla sedan tillbaka till grenen *main/master* på samma sätt som när växlingen till den nya grenen gjordes.



Figur 36: Går tillbaka till grenen main.

Öppna filen *vagonpoesi.txt*, går det att observera något konstigt i denna fil?



Figur 37: De två nyligen tillagda raderna saknas.

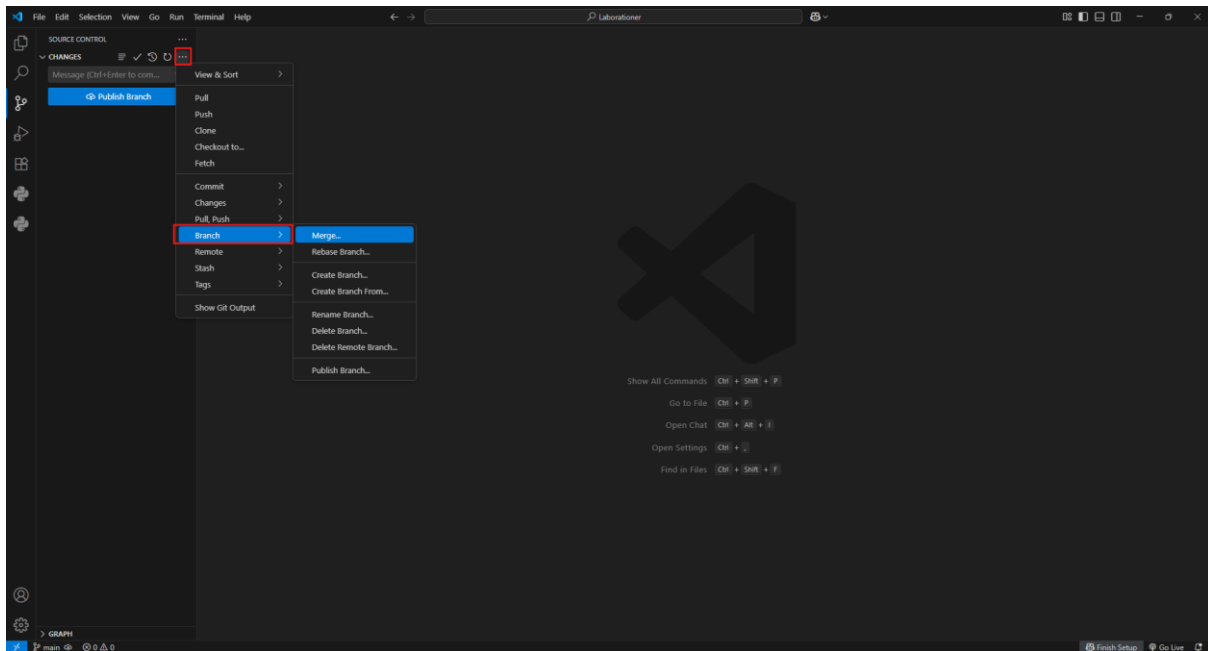
Mycket riktigt saknas de rader som precis lades till, men detta är väntat. De finns fortfarande, men enbart i den tidigare skapade grenen, i detta fall `new_poetry`. Detta ska självklart lösas, men innan dess, öppna *vagonpoesi.txt* i grenen *main/master* och byt ut punkten på första raden mot ett utropstecken. Spara filen och checka därefter in den enligt den praxis som beskrivits.

Slå samman två grenar

Tidigare märktes att text saknades i *vagonpoesi.txt* i grenen *main/master*, samtidigt som det senaste utropstecknet endast finns i texten på *main/master*. Detta ska nu lösas genom att utföra en så kallad *merge*, vilket innebär att två grenar slås samman.

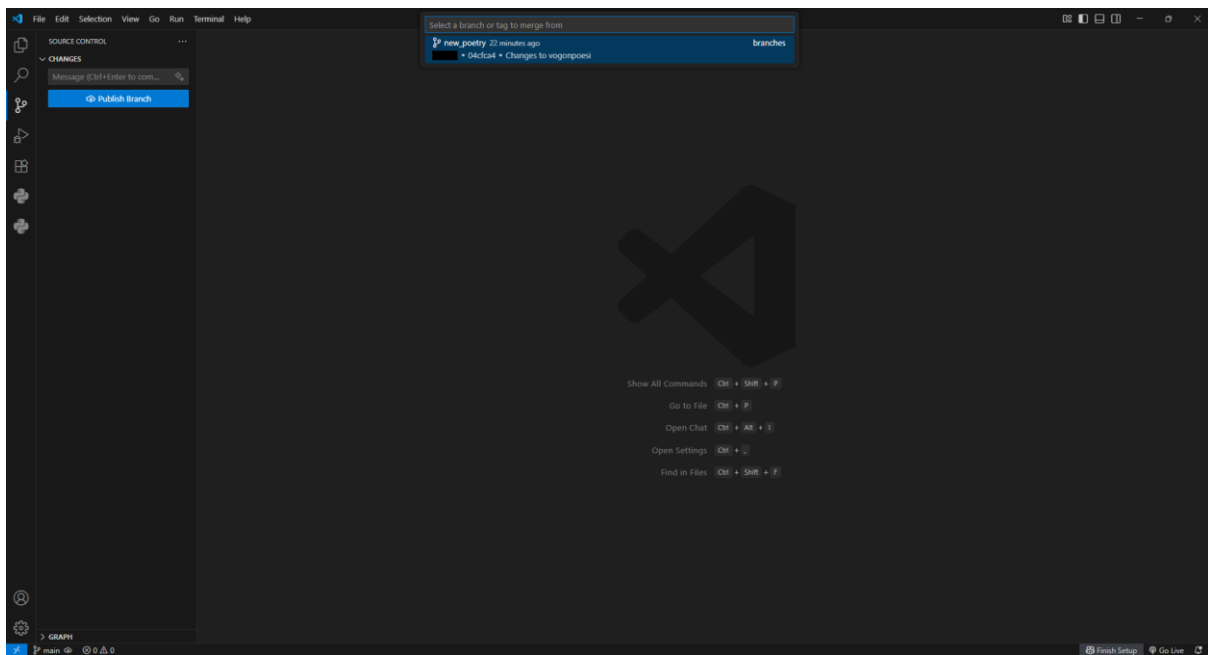
Börja med att säkerställa att VS Code befinner sig i grenen *main/master*. Gå sedan till *versionshantering*, klicka på de tre punkterna intill *Changes*, välj *Branch* och därefter *Merge...*

.



Figur 38: Välja att utföra en merge.

I nästa ruta, välj den andra grenen som skapades tidigare.

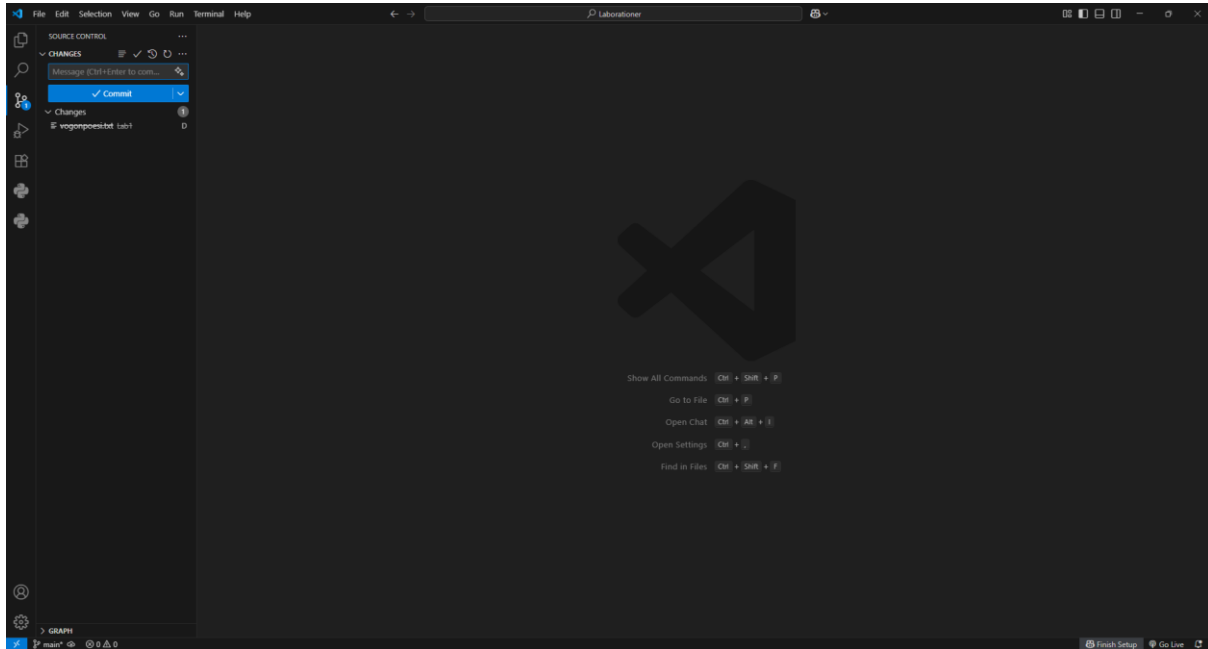


Figur 39: Väljer new_poetry.

Observera nu att filen *vogonpoesi.txt* innehåller allt, både det som lades till i grenen new_poetry och utropstecknet som lades till precis innan mergingen gjordes.

Ta bort en fil

Eftersom dikten vagonpoesi inte är något som bör behållas, ska den slutligen tas bort. *Börja med att radera filen.* Gå sedan återigen till *versionshantering*, det bör nu dyka upp en logg för vagonpoesi.txt, men observera att filen nu är *överstruken*. Detta indikerar att filen har tagits bort.

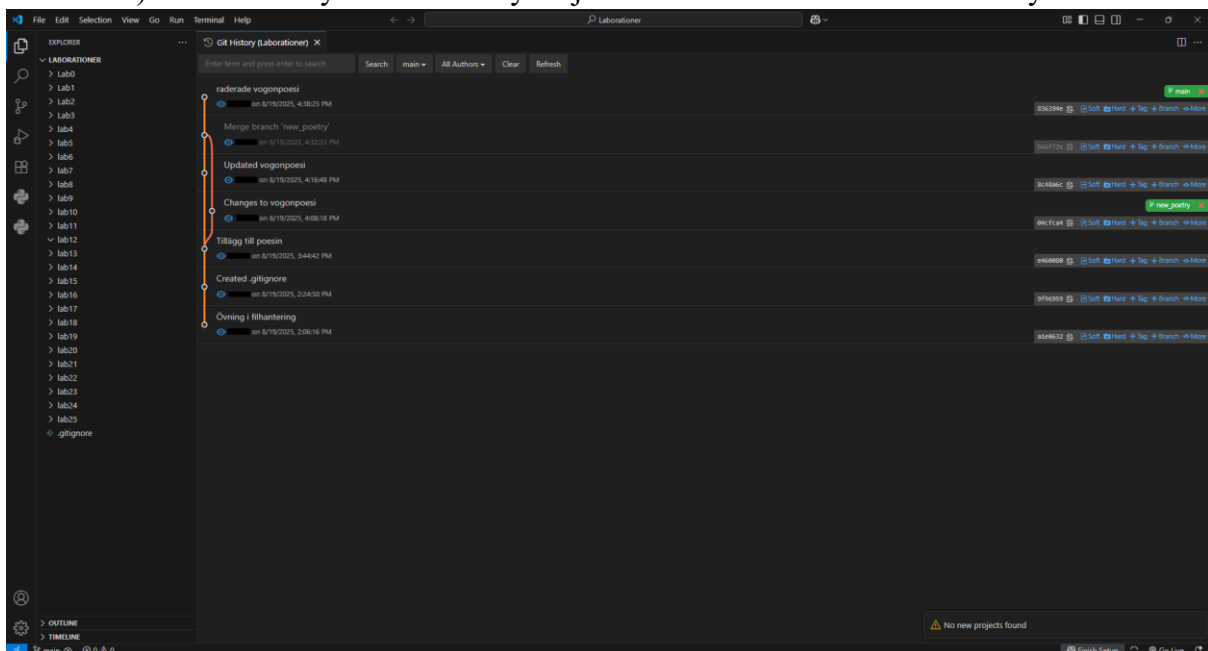


Figur 40: vagonpoesi.txt är raderad.

Återigen checka in detta enligt den praxis som har satts upp.

Summering

Alla steg som gåtts igenom i denna laboration kan verka svåra att komma ihåg, eftersom det var en hel del. Lyckligtvis behöver man inte minnas detta på egen hand, eftersom Git hanterar det. Genom att högerklicka i filträdet över laborationerna (utan att välja någon specifik fil eller laboration) visas en meny. I denna meny väljs alternativet Git: View File History.



GitHub

Fortsätter i Laboration 2.

Uppgifter

Terminal/Git Bash

Alla övningar ska göras helt i Git Bash/Terminal

1. Skapa en katalog för alla kurser som läses just nu under katalogen *Universitet*
2. I katalogen *Laborationer* i *DA339A*, skapa kataloger för kommande labbar *Lab2 till och med Lab25*
3. Skapa en textfil (.txt) i katalogen *Lab1*
4. Öppna den nyskapade filen, lägg till en kort text och skriv sedan ut innehållet
5. Radera den skapade textfilen
6. Skapa en katalog i *Lab1* och ta sedan bort den
7. Kopiera en fil antingen i *Lab0* eller *Lab1*
8. Flytta en fil från *Lab0* till *Lab1*