Komma i gång med git

Git är ett versionhanteringssystem (VCS, Version Control System) som håller reda på och spårar ändringar som sker i filer. Git är inte enbart till för oss utvecklare utan går även att använda för vanliga filer som vi ofta måste ändra i och vi behöver hålla reda på ändringar som sker och har skett.

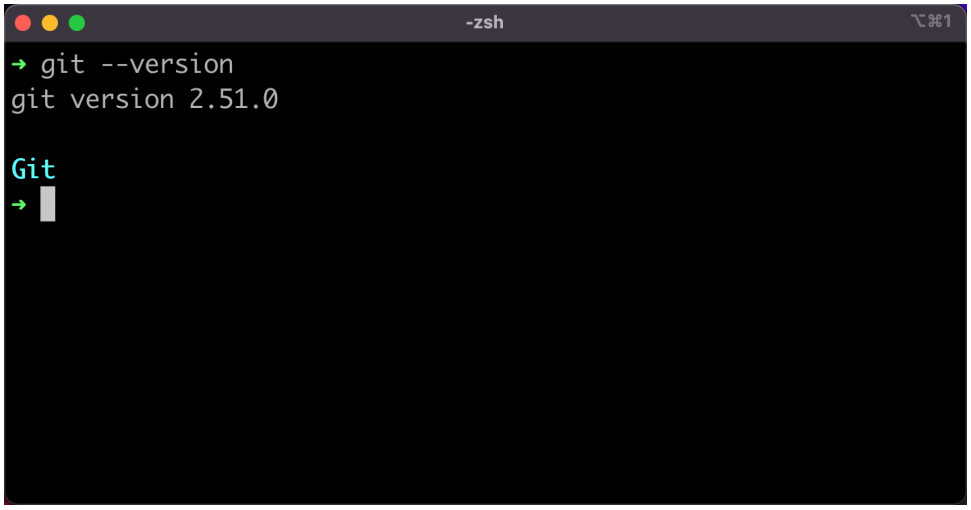
För oss utvecklare innebär det att vi får ett enkelt sätt att hantera ändringar som vi gör i våra kodfiler som även enkelt låter oss att ångra ändringar som vi gjort. Vi kanske gör ändringar i koden som i sin tur leder till att något inte längre fungerar och vi behöver enkelt hitta alla ställen som vi har ändrat på utan att gå igenom hela kodmassan. Om vi har vår kod hanterad av git kan vi enkelt backa(ångra) de nya ändringarna som vi gjort tillbaka till senast fungerande version.

Git låter oss även att skapa nya grenar(branches) för att testa idéer som vi inte vill lansera ännu. Vi kan även med hjälp av git samarbeta med andra utvecklare och dela med oss av kod utan att vi påverkar varandras kod. När vi är klara så kan vi enkelt slå samman ändringarna till en kodmassa.

Installera git

Först och främst måste vi ha git installerat. Om det inte är installerat så gå till https://git-scm.com/downloads och ladda ner en version som är anpassat för operativsystemet och följ installationsanvisningarna. Installationsanvisningarna är olika för Windows, Linux samt MacOS.

När installationen är klar så öppna git's egna bash terminal och skriv in följande kommando i terminalfönstret



Om version numret returneras så är git installerat korrekt och vi kan gå vidare till nästa steg.

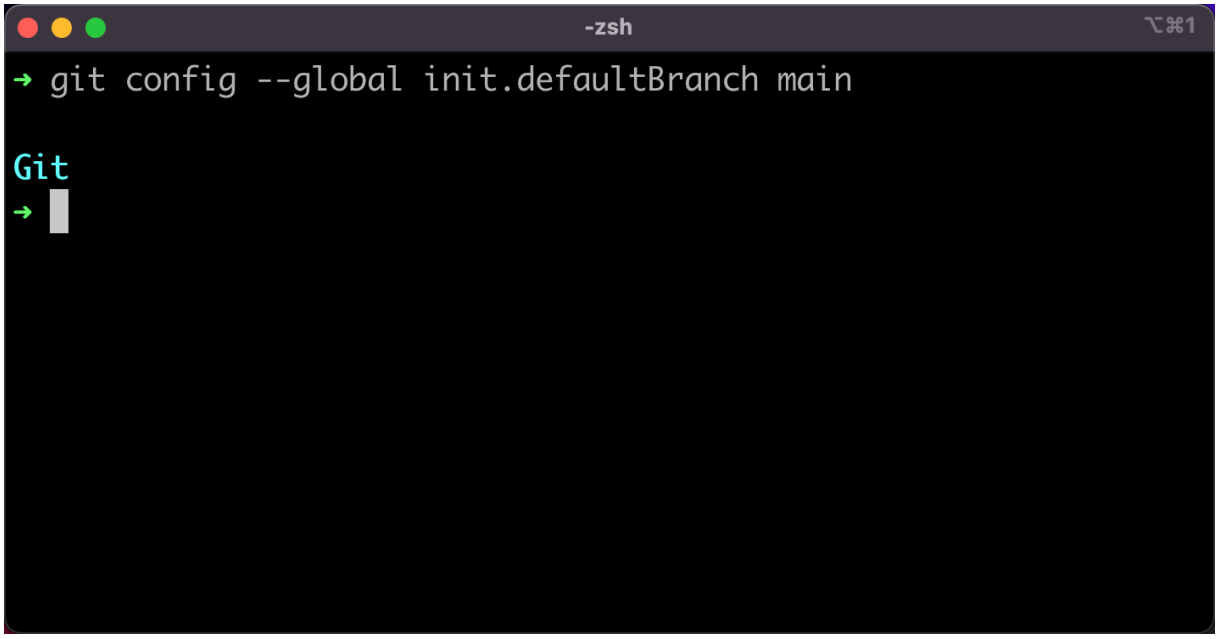
Konfigurera git

Innan vi börjar använda git behöver vi göra några globala inställningar så att vi inte får problem när vi börjar checka in kod i git och framöver till GitHub. Följande inställningar behövs göra.

1. Se till att huvudgrenen (main branch) får namnet main. Det finns en risk att installationen har givit huvudgrenen namnet master, vilket är det gamla och föråldrade namnet vi förr gav den.
2. Registrera vårt användarnamn, detta kan vara vad som helst men bör matcha namnet som vi kommer att registrera på GitHub när vi ska börja publicera kod i molnet.
3. Registrera vår e-postadress som även denna måste vara den e-postadress som vi kommer att registrera på GitHub senare.
4. Ställa in VS Code som vår standard editor för git.

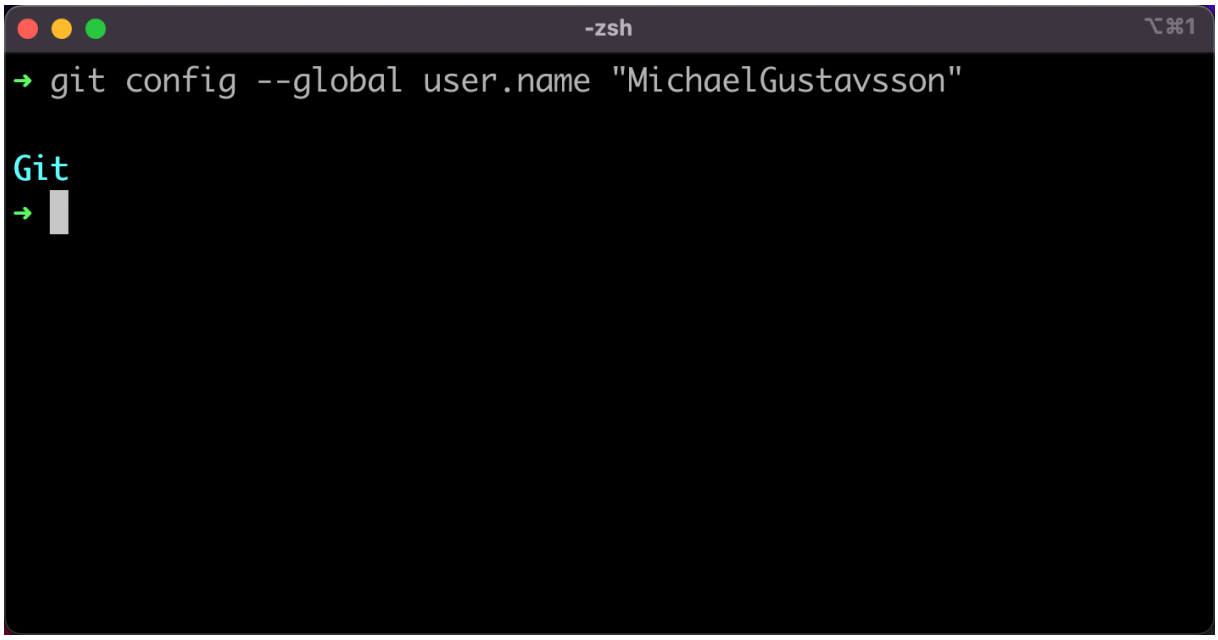
Steg 1.

Öppna upp ett bash terminalfönster om det inte redan är öppet och skriv in följande kommando och tryck på Enter.



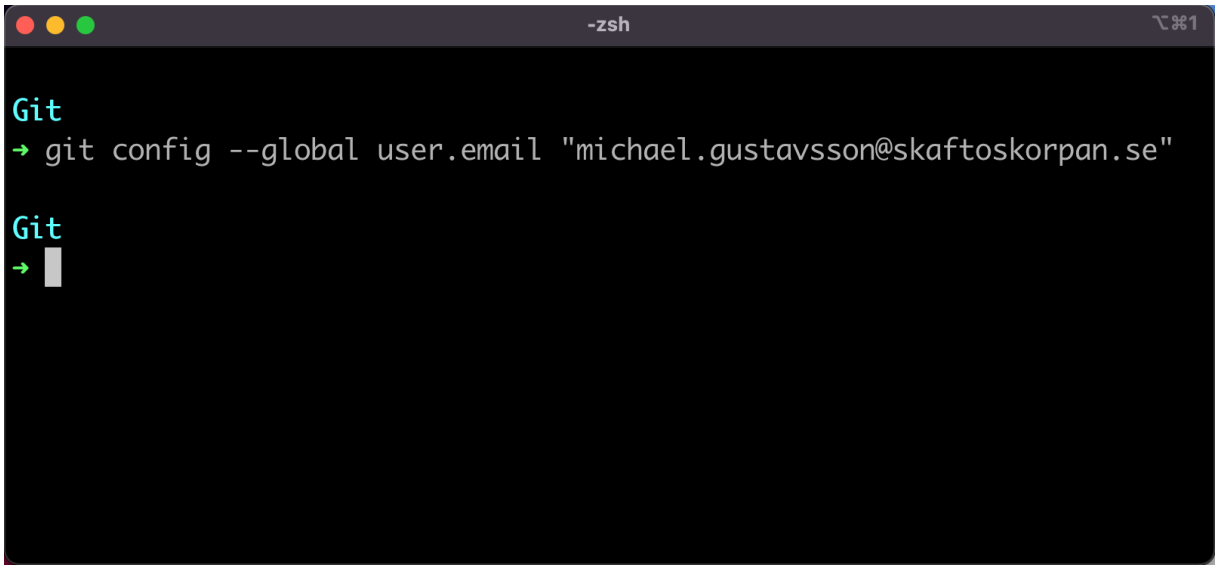
Steg 2.

I samma terminalfönster skriv in följande kommando, men byt ut mitt namn mot ert användarnamn och tryck på Enter.



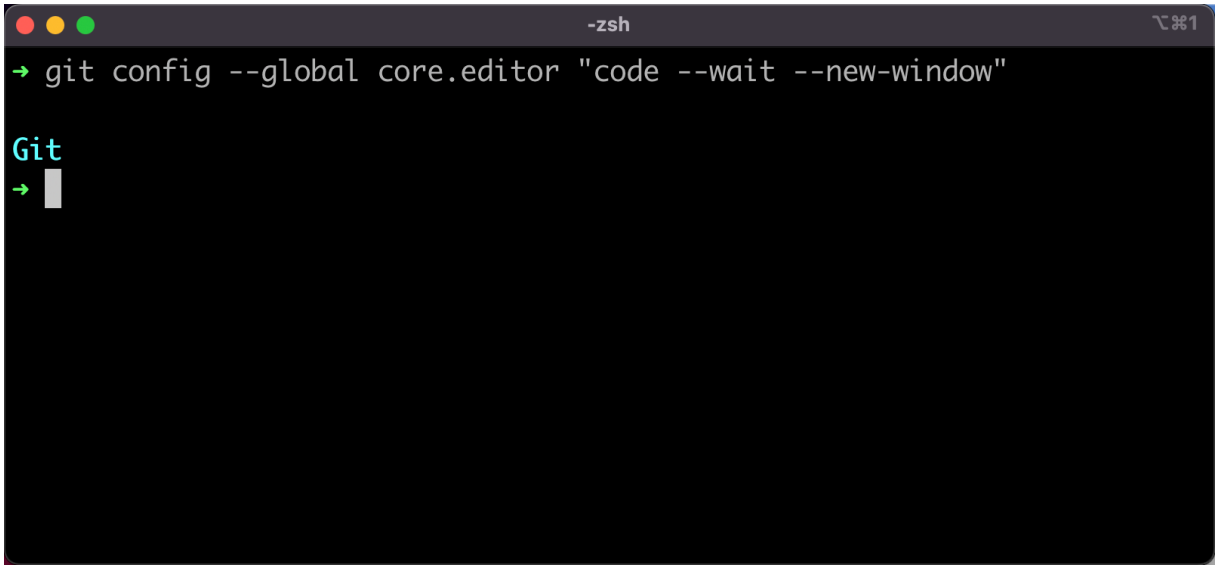
Steg 3.

Än en gång i samma terminal fönster skriv följande kommando och byt ut min e-postadress till er egen e-postadress och tryck på Enter.



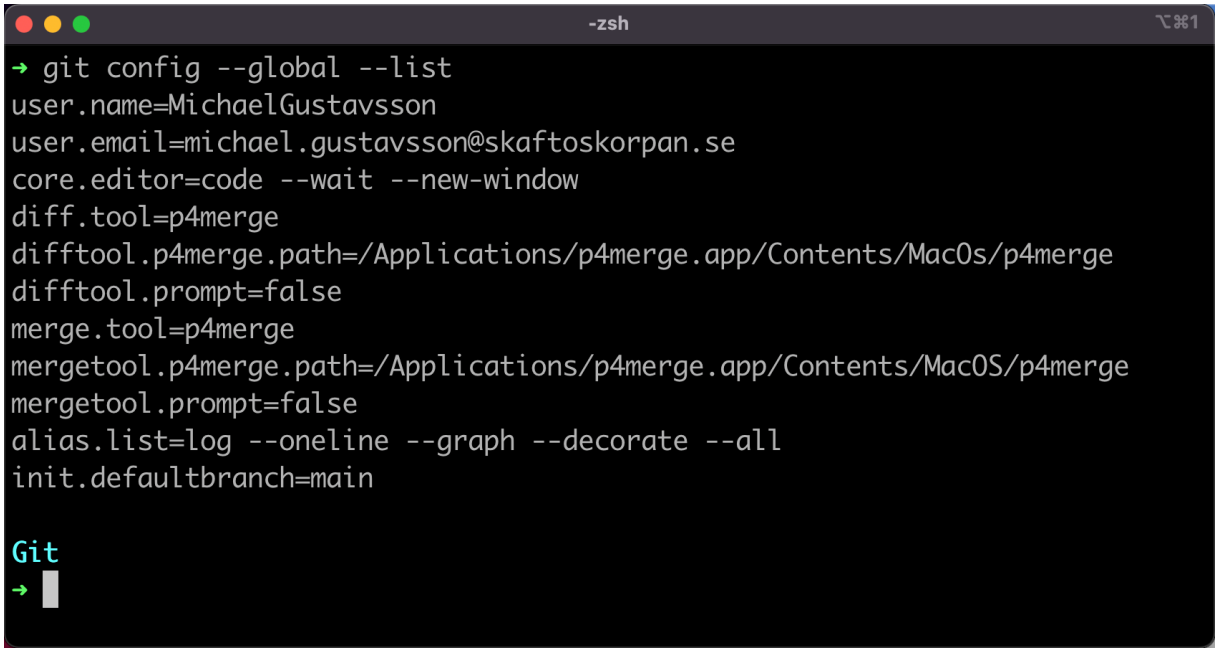
Steg 4.

Sista steget i konfigurationen är att ställa in VS Code som vår redigerare för git. Om vi inte gör detta vår vi git standard redigerar som är gamla hederliga Vim som inte är speciellt kul att arbeta med 😂.



Bekräfta konfigurationen

Än en gång i terminalfönstret skriv in följande kommando och tryck på Enter.

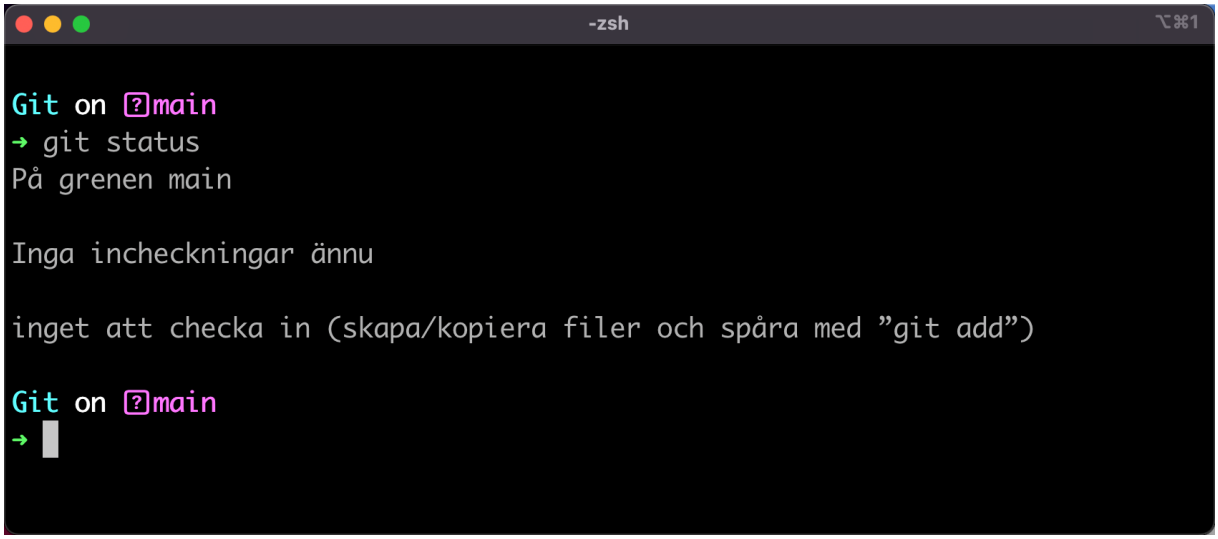


Sätta upp vårt första git repo

I terminal fönster skapa en mapp/katalog som ni vill använda för följande exempel. I mitt fall har jag skapat en mapp **Git**. Navigera in i den nyligen skapade mappen med kommandot cd <<mappnamn>>.

Skriv nu in följande kommando och tryck sedan på Enter.

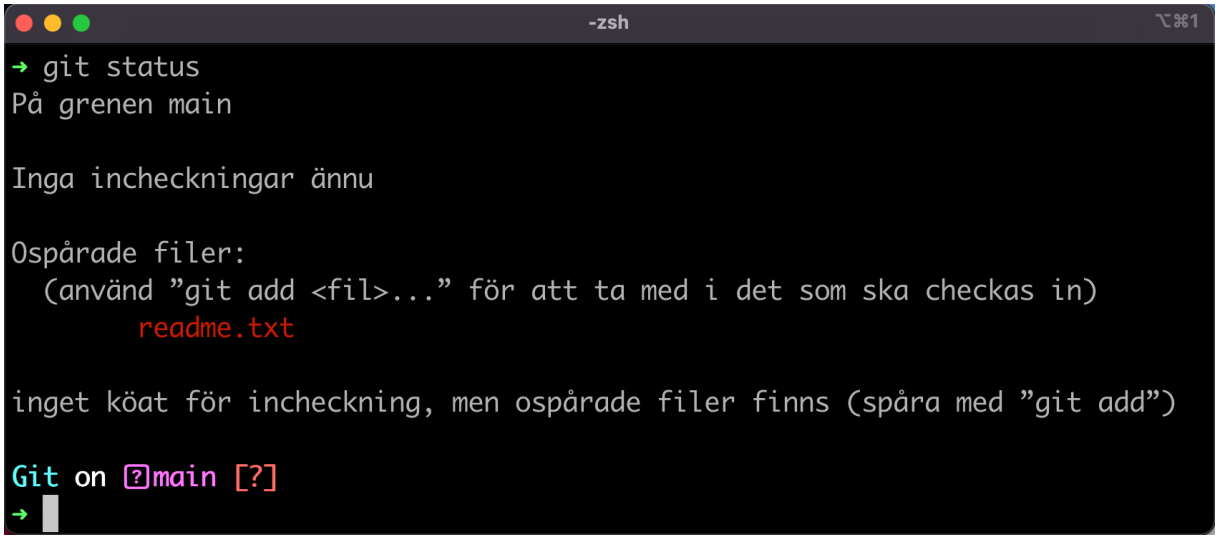
Vi har nu skapat vårt första git repo. Vi kan skriva in följande kommando för att bekräfta att vi har ett tom repo och att vi är på grenen main.



Arbeta med git

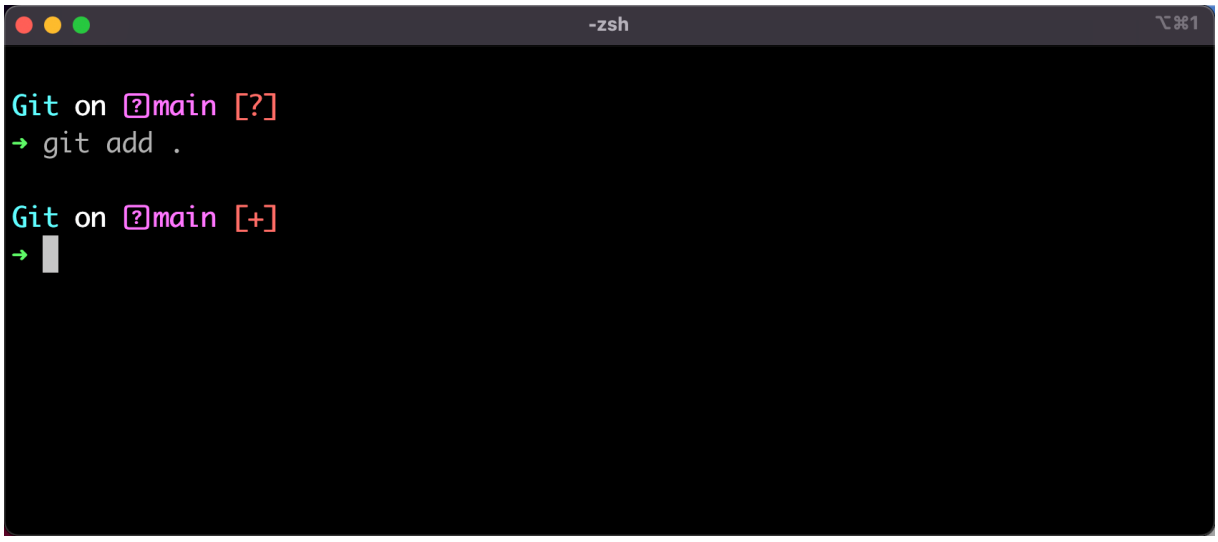
Nu när vi har ett nytt repo låt oss testa det. I terminal fönstret skapa en fil med kommandot touch. Jag skapar en fil med namnet readme.txt.

Om vi nu skriver in kommandot git status i terminalfönstret så ser vi följande resultat.

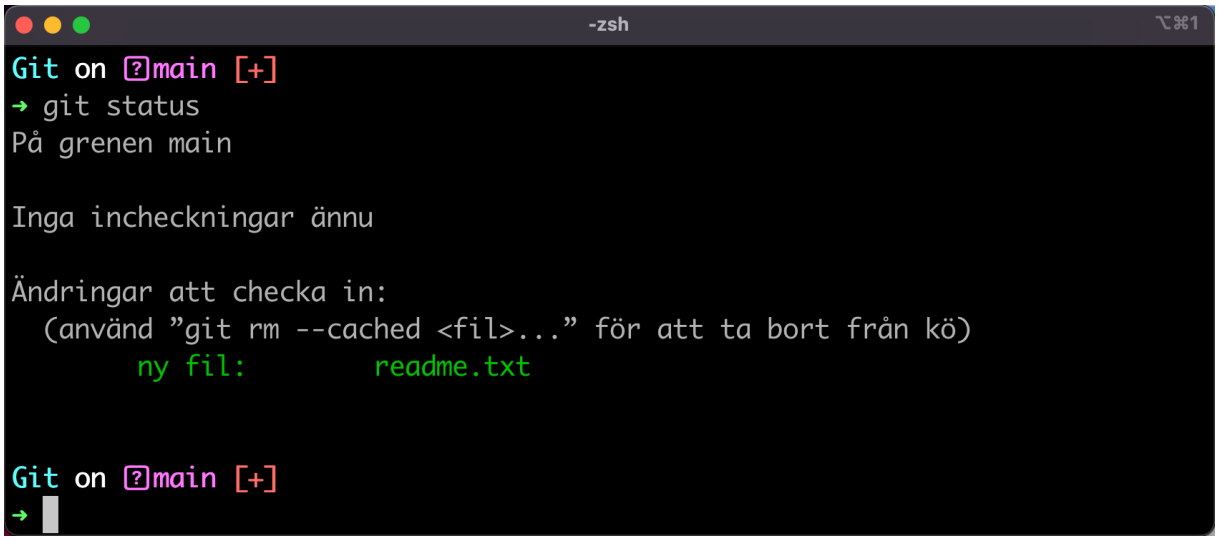


Vi kan nu se att vi har ospårade filer i vårt fall readme.txt vilket syns i rött i terminalfönstret. Detta betyder att git är medveten om filen readme.txt men inte ännu har tagit filen under sina vingar för att spåra ändringar som görs i filen.

För att tala om för git att git ska ansvara för filen måste vi göra en så kallad staging av filen. Detta gör vi genom att skriva in kommandot ***git add .*** i terminalfönstret.



Punkten i kommandot ta hand om ALLA filer som finns i katalogen. Vill man enbart lägga till en enstaka fil kan vi skriva git add <<filnamn>>. Om vi nu ännu en gång skriver kommandot git status i terminalfönstret får vi följande resultat.

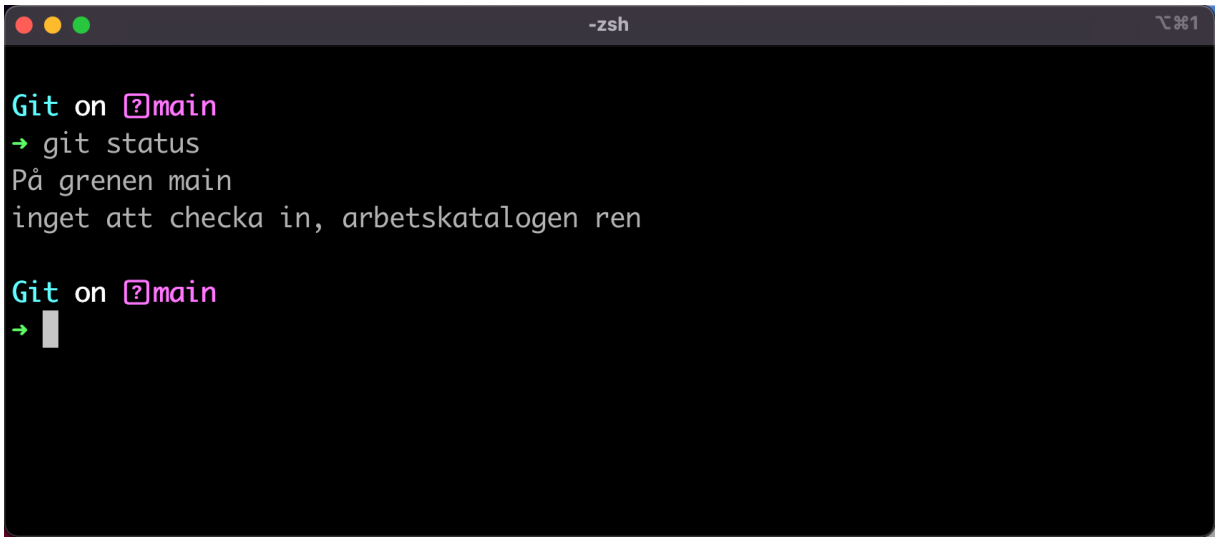


Vad vi nu ser är att git har lagt till filen och aktiverat spårning av ändringar.

Nu kan vi checka in filen i git's repo genom att skriva in följande kommando ***git commit -m "Initial Commit"***.

Commit är kommandot som talar om för git att nu är det ok att lagra/spara ändringarna till till repot. Flaggan -m används för att ge ett namn som gör att vi kan spåra ändringen. Meddelande är valfritt men det är viktigt att meddelandet på något sätt beskriver vad som skett med filen.

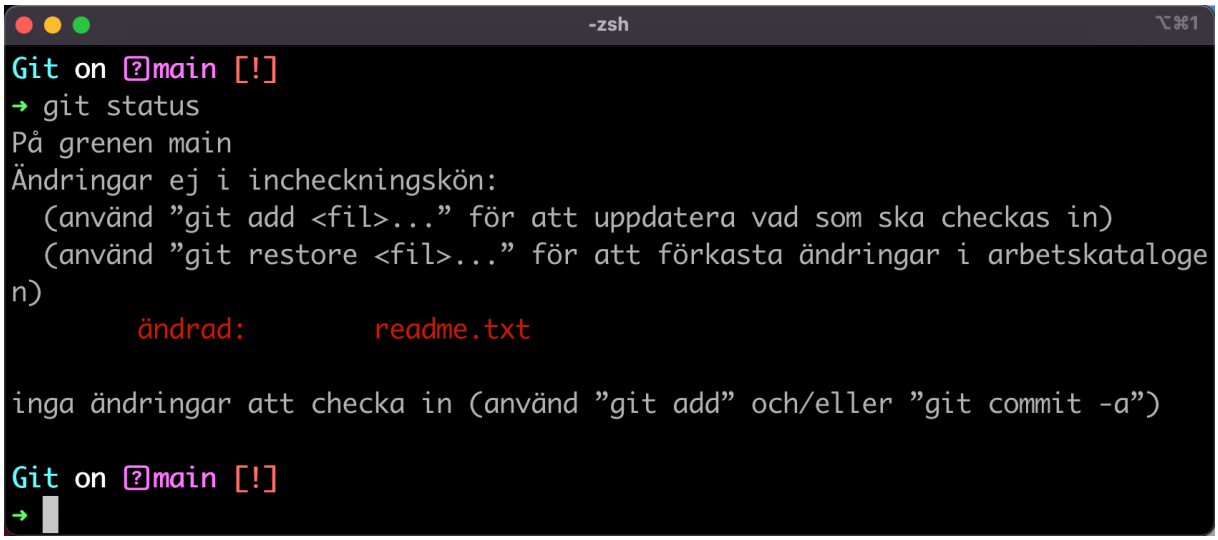
Om vi nu skriver in kommandot *git status* igen kan vi se att allt är i sin ordning och att det inte finns något som inte är omhändertaget 🎉.



Hantera förändringar

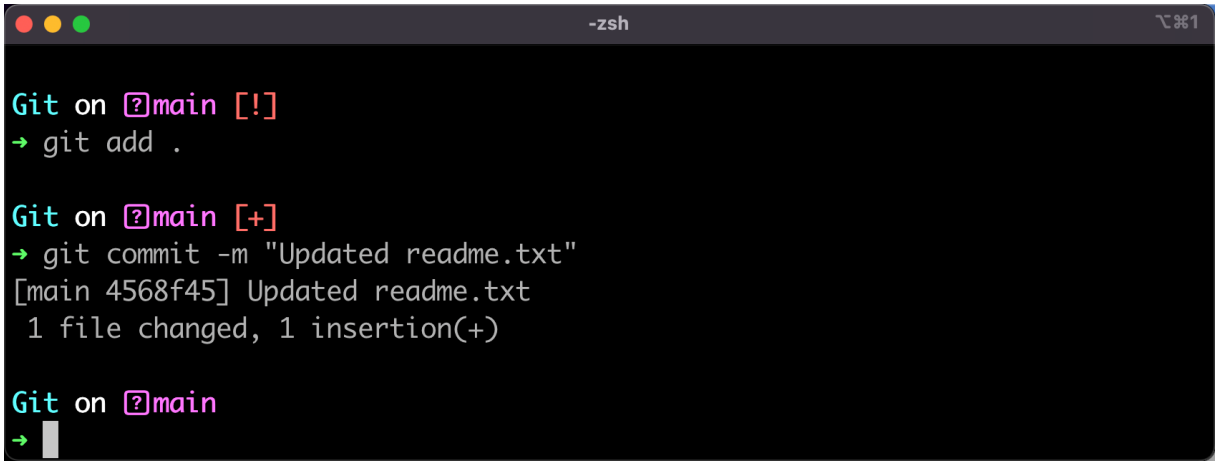
Öppna upp readme.txt filen i valfri texteditor och gör några ändringar i dokumentet. Spara dokumentet och stäng textredigeraren och gå tillbaka till terminalfönstret.

I terminalfönstret skriv in kommandot *git status*.



Vi kan nu se att git indikerar åt oss att filen readme.txt är ändrad. Vi behöver nu bara göra om samma sak igen. Det vill säga lägga till ändringarna i kön för incheckning.

Så vi skriver in kommandot git add . igen. Efter det skriver vi in kommandot git commit -m ”Updated readme.txt” för att checka in ändringen.



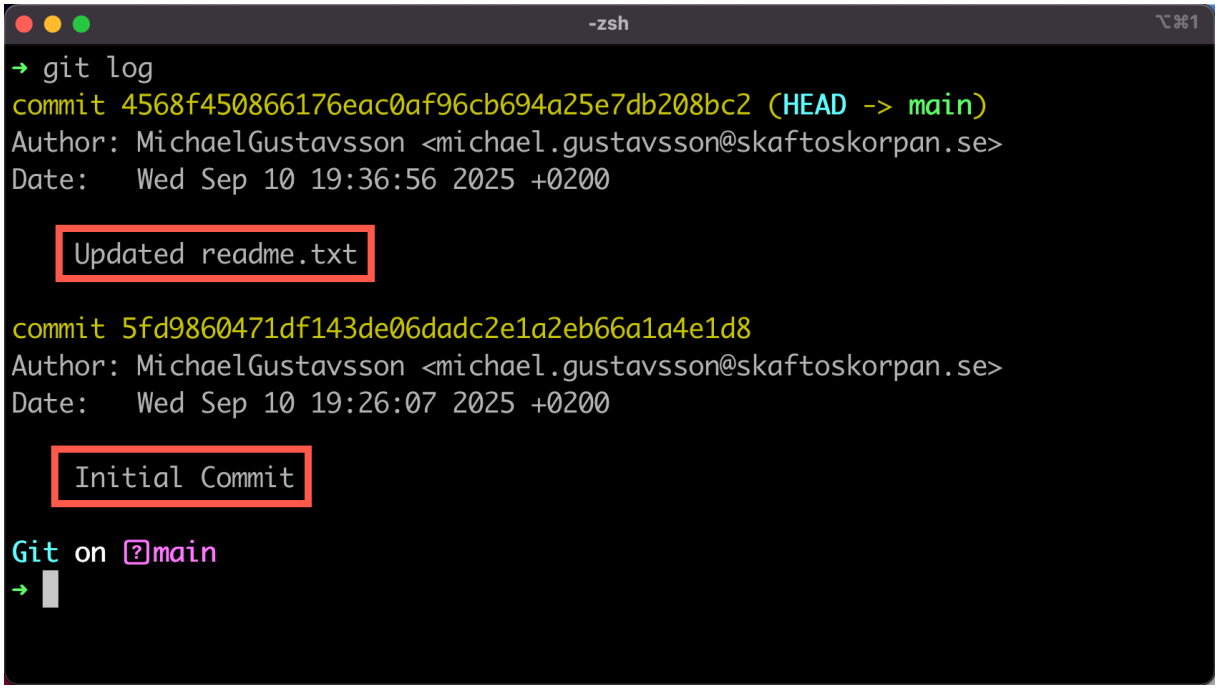
OBSERVERA!

Det finns genvägar för att både gör add och commit i ett kommando, men det kommer jag till lite senare.

Avslutningsvis så kör vi kommandot *git status* igen. Vi kan se att allt är ok och allt är i repot.

Git historik

Vi kan kontrollera vad som har hänt över en tid i vårt repo genom att använda ett annat kommando i git. Låt oss testa följande kommando *git log* i terminalfönstret.



Vi ser våra båda ändringar i utskriften. Längst ner ser vi vår initiala incheckning och överst/först ser vi den senaste incheckningen. Vi ser även vem som har gjort det och vilken datum och tid det inträffade.

Git kommandon

Kommando	Beskrivning
git init	Skapar ett git repo
git add .	Lägg till filer i repot
git commit -m	Sparar ner förändringarna till repot
git log	Listar historik
git status	Visar om vi har placerat filer, filerna i Gits repo.
git config	Används för att sätta upp i vilket namn och vilken e-postadress som ska användas för att spåra vem som gjort vad. Används primärt tillsammans med GitHub.
git branch	Listar alla grenar som är skapade för git repot
git checkout -b <branch-name>	Skapar och byter till en ny gren
git checkout <branch-name>	Byter till en befintlig gren
git merge <source-branch>	slår samman ändringar ifrån <source-branch> till aktuell gren.
git config –global –list	Visar alla inställningar i git som är gjorda globalt(central) som kommer att påverka varje nytt git repo
git config –global init.defaultBranch main	Sätt namnet på huvudgren i config så att alla nya git repo får samma huvudgren.
git config –global –edit	Öppnar upp den globala config edit för ändringar
git config –edit	Öppnar upp den lokala config för ändringar
git config –unset user.name	Ta bort vald inställning lokalg
git config –global –unset user.name	Ta bort vald inställning globalt
git config –global core.editor “code –wait –new-window”	Ställer in VS Code som vår standardeditor för git.
git config –global user.name “Användarnamn”	Ställer in användarnamn i git
git config –global user.email “e-post address”	Ställer in e-post adress i git