

# Lite smått om Git

pawdzi-7

September 2022

## 1 Uppbyggnad av ett git-projekt

I figur 1 kan ni se olika branches av ett repo. Vi har branchen "master" som kan ses som en huvudbranch där all vår fungerande kod ligger. Det hade varit dumt att inte alltid ha en fungerande kopia ifall något går sönder, alltså görs det inga förändringar i den direkt. Istället har vi branches som i figuren heter "development", "feature branch A", "feature branch B". Vill vi införa en ändring i koden gör vi det alltså i en sådan feature branch. Därefter mergar vi våra ändringar med development branchen, och först när development branchen är testad och fungerande kan vi börja fundera på att merga med vår master branch. Allt detta kan kännas omständigt, men det är viktigt att det upprätthålls för att folk ska kunna koda parallellt, och för att undvika en situation där något slutar fungera och vi helt plötsligt inte har en fungerande kodbas alls.

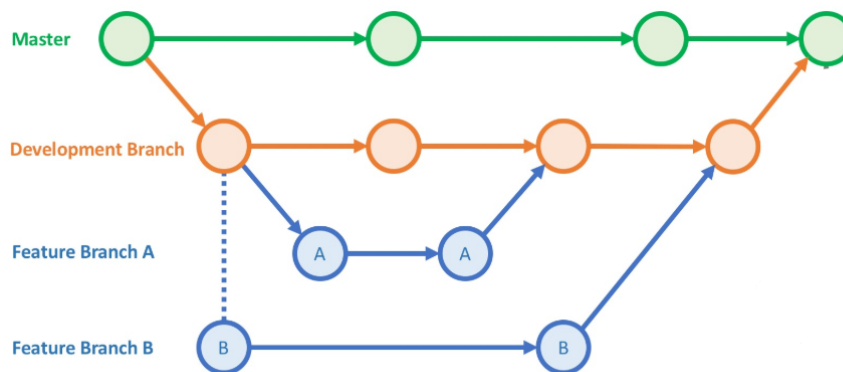


Figure 1: Strukturexempel av ett repo.

## 2 En kort tutorial.

För att alla ska kunna testa på hur git fungerar har jag skapat ett test-repo. Är du med i rapportrepon så har jag redan bjudit in dig, och är du inte det än får

du kontakta mig så löser jag både tillgång till rapportrepon och testrepon.

Här utgår jag från att du använder antingen Linux eller Git Bash under Windows. Under Mac OS tror jag att det ska fungera på ungefär samma sätt, men jag lovar inget. Har du Windows men inte Git Bash, och vill skaffa det så finns det hundratals guider redan, så det är bara att Googla och lösa det.

Repon hittar du under

```
https://github.com/pawel-dzialo/testrepo
```

För att ladda ner ett repo öppnar du en git klient, vare sig Git Bash eller ett terminalfönster på Linux och skriver

```
git clone [SSH URL]
```

SSH URL hittar du på sidan för repot under den gröna knappen "Code". Välj sedan SSH och kopiera adressen.

När repon är clonad skriver du

```
cd [Namn på repot]
```

För att komma in i mappen. I vårt fall är namnet på repot testrepo.

Testrepot är uppbyggt enligt en struktur som liknar den i figur 1. Det finns alltså en branch som heter master och en branch som heter development. Som du kan se på repots githubsida, så finns det ändringar gjorda i development branchen som inte har mergats med master, nämligen filerna nyfil, och nyfil2. För att ladda ner den nyaste versionen av development branchen skriver du

```
git checkout development
```

Nu arbetar du i development branchen istället för master. Säg att du nu vill göra en egen ändring, exempelvis lägga till en ny fil. Först skapar du en ny branch

```
git branch [namn på din branch]
```

Ett lämpligt namn för en branch är exempelvis namnet på funktionen som du vill lägga till. För att byta till din nyskapade branch skriver du

```
git checkout [namn på din branch]
```

Nu kan du göra ändringar. Detta simulerar vi genom att du exempelvis skapar en fil. När du har gjort detta kan du lägga till din fil i branchen genom

```
git add [namn på filen]
```

Därefter skapar du en commit och pushar direkt till din nyskapade branch.

```
git commit -m "en kort beskrivning av ändringen"
git push
```

Nu ligger din ändring med i din nyskapade branch. Innan du försöker slå ihop den med development bör du kolla ifall det inte finns några ändringar som har gjorts i development medans du har jobbat på din ändring.

```
git checkout development
git pull
```

Kommer att ändra till development-branchen och ladda ner alla ändringar som har gjorts.

```
git checkout [namnet på din branch]
git merge development
```

Detta försöker att mergea din nyskapade branch med development. Finns det inga ändringar som krockar med dina så har Finns det ändringar som har gjorts som krockar med dina, kommer detta att visa ett fel, ett exempel på detta finns i figurerna längst ner.

Slutligen har jag låst master branchen så att det inte går att bara ändra i den. Istället får vi, när vi känner oss nöjda med hur development branchen ser ut, skapa en pull request. Denna måste godkännas av minst en person för att gå igenom och slås ihop med master. Pull requesten kan skapas genom att gå in på repots gitsida, gå in på branchen development och klicka på Compare & Pull Request.

```

apwel@DESKTOP-378KVNL MINGW64 ~/Desktop/testar/testrepo (feature3)
$ git checkout development
Switched to branch 'development'
Your branch is up to date with 'origin/development'.

apwel@DESKTOP-378KVNL MINGW64 ~/Desktop/testar/testrepo (development)
$ git pull
remote: Enumerating objects: 5, done.
remote: Counting objects: 100% (5/5), done.
remote: Compressing objects: 100% (3/3), done.
remote: Total 3 (delta 0), reused 3 (delta 0), pack-reused 0
Unpacking objects: 100% (3/3), 332 bytes | 36.00 KiB/s, done.
From github.com:pawel-dzialo/testrepo
   584a68f..5127245  development -> origin/development
Updating 584a68f..5127245
Fast-forward
 nyfil | 5 ++++
 1 file changed, 4 insertions(+), 1 deletion(-)

apwel@DESKTOP-378KVNL MINGW64 ~/Desktop/testar/testrepo (development)
$ git checkout feature3
Switched to branch 'feature3'
Your branch is up to date with 'origin/feature3'.

apwel@DESKTOP-378KVNL MINGW64 ~/Desktop/testar/testrepo (feature3)
$ git merge development
Auto-merging nyfil
CONFLICT (content): Merge conflict in nyfil
Automatic merge failed; fix conflicts and then commit the result.

```

Figure 2: Vid försök till pull av development ser vi att det har gjorts ändringar. Vi byter till vår lokala branch och försöker mergea med development, varpå ett fel uppstår med filen nyfil.

```

1  def function():
2      while True:
3          print("Hejhej")
4  <<<<<<< HEAD
5
6  def newFunction(i):
7      return i-1
8  =====
9
10 def newFunction(i):
11     return i+1
12 >>>>>> development
13

```

Figure 3: Filen öppnas lokalt, och vi kan se att ändringen i development har varit att en funktion returnerar i+1 och lokalt returnerar den i-1

```

1  def function():
2      while True:
3          print("Hejhej")
4
5  def newFunction(i):
6      return i+1
7

```

Figure 4: Efter diskussion med personen som har gjort ändringen kommer vi överens om att i+1 är den korrekta lösningen.

```
apwel@DESKTOP-378KVNL MINGW64 ~/Desktop/testar/testrepo (feature3|MERGING)
$ git add nyfil

apwel@DESKTOP-378KVNL MINGW64 ~/Desktop/testar/testrepo (feature3|MERGING)
$ git commit -m "Resolved conflict"
[feature3 49f9746] Resolved conflict

apwel@DESKTOP-378KVNL MINGW64 ~/Desktop/testar/testrepo (feature3)
$ git push
Enumerating objects: 7, done.
Counting objects: 100% (7/7), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 332 bytes | 332.00 KiB/s, done.
Total 3 (delta 1), reused 0 (delta 0), pack-reused 0
remote: Resolving deltas: 100% (1/1), completed with 1 local object.
To github.com:pawel-dzialo/testrepo.git
   7c78d2b..49f9746  feature3 -> feature3

apwel@DESKTOP-378KVNL MINGW64 ~/Desktop/testar/testrepo (feature3)
$ git merge development
Already up to date.
```

Figure 5: När nyfil har ändrats för att lösa konflikten kan vi åter pusha den till vår nya branch. Nu borde det gå bra att mergea.