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.

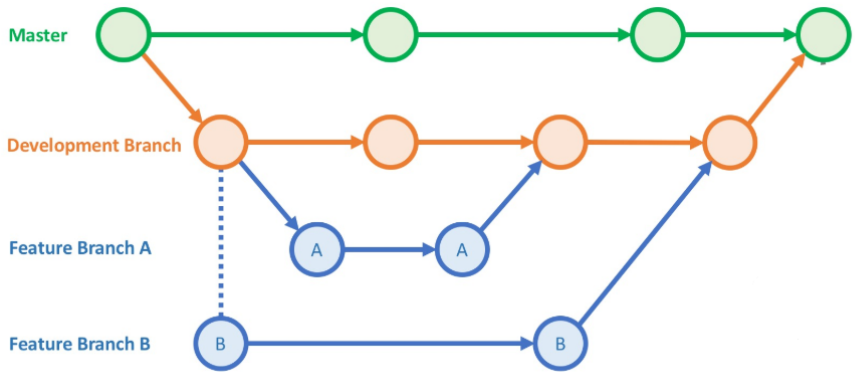

Figure 1: Strukturexempel av ett repo.

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 antigen 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 konflikter, som kan ha uppstått om någon annan har ändrat i development medans du har jobbat på din nya branch.

```
git checkout development
git pull
```

Kommer att ändra till development-branchen och ladda ner alla ändringar som har gjorts. Står det up to date så går det bra att mergea.

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

Finns det ändringar som har gjorts måste du gå tillbaka till din nya branch, och försöka lösa eventuella konflikter. Ett exempel på detta syns nedan.

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.

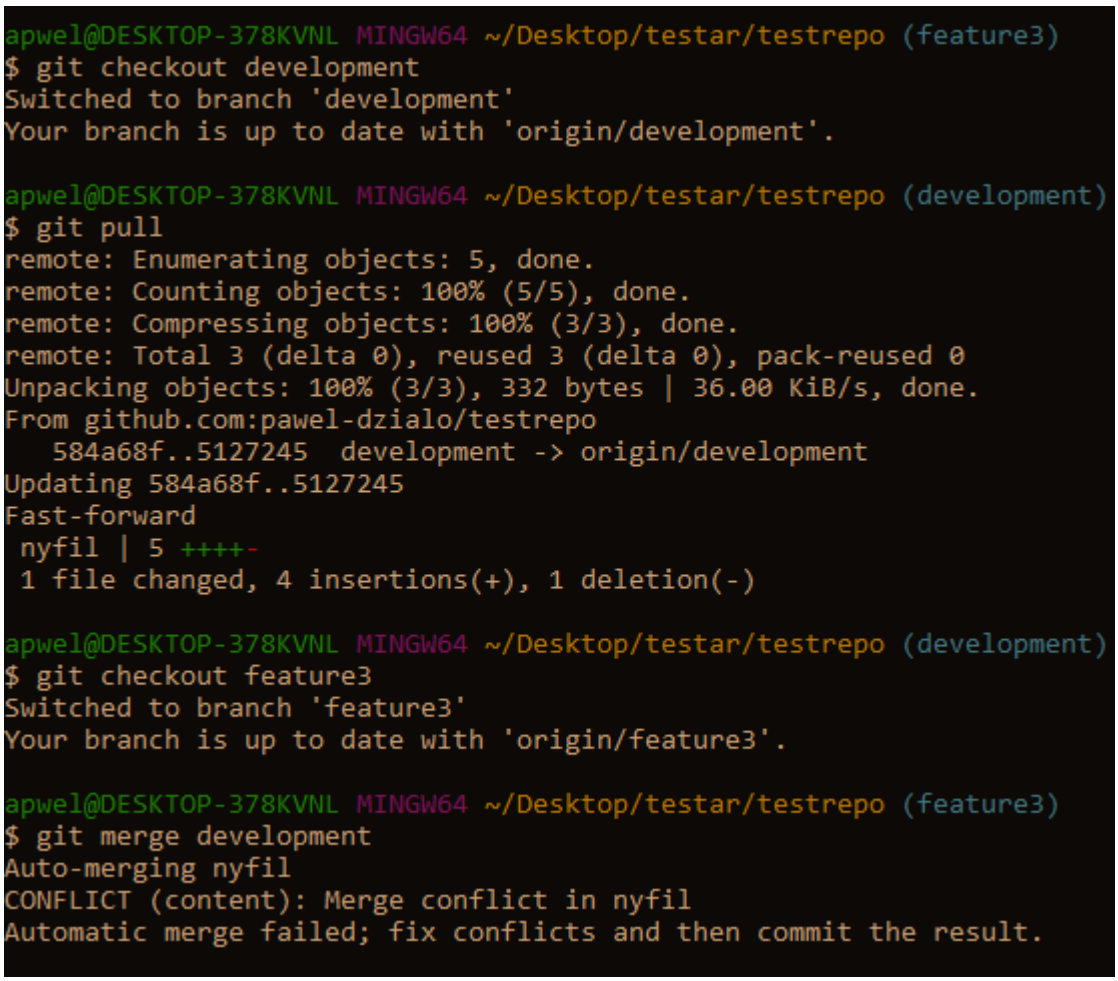

Figure 2: Vid försök till pull av development ser vi att det har gjorts ändringar. Vi byter till vår lokala branch och försöker mergea med development, varpå ett fel uppstår med filen nyfil.

```
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. Nu kan nyfil mergas in i development.