webgroup

Git-tips

Innehåll

Se vilka filer som ändrats

`git status`

Exempel på utskrift från kommandot:

```
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

      modified:   textfile2.txt

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

      modified:   textfile1.txt

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

      textfile3.txt
```

Förklaring:

`Changes to be committed:` är filer som är tillagda och kommer att commitas i nästa commit.

`Changes not staged for commit:` är filer som har ändrats men inte lagts till och som INTE kommer att commitas i nästa commit.

`Untracked files:` är filer som har ändrats eller skapats som inte har commitats tidigare och inte heller lagts till för att commitats.

Vanliga problem:

Om det står med filer under `Untracked files` som inte ska commitas, exempelvis filer i ./idea mappen, så ska man lägga till dem i `.gitignore` filen. De kommer annars att läggas till i `Changes to be committed` om man kör `git add -A` utan att ha skrivit in dem i `.gitignore`

Om det står med filer under `Changes to be committed:` som inte ska commitas så kan man plocka bort filen genom kommandot `git reset HEAD textfile2.txt` om textfile2.txt inte skulle commitas.

Lägg till filer

`git add -A`, `git add <file>`

Alternativ till `git add -A`:
`git add -A` lägger till alla filer till nästa commit.
`git add --all` samma som ovan.
`git add .` lägger till alla nya och modifierade, men inte borttagna filer.
`git add -u` lägger till alla borttagna och modifierade, men inte nya filer.

Förklaring:
Filer som läggs till med `git add` kommer att commitas i nästa commit.

Vanliga problem:
Om man lägger till en fil man inte vill commita så kan man plocka bort filen genom kommandot `git reset HEAD textfile2.txt` om textfile2.txt inte skulle commitas, om man vill ta bort alla tillagda filer så kör man `git reset HEAD`.

Jobba med branches

Skapa en branch per feature.

git checkout master
git pull
git branch [branch-name]
git checkout [branch-name]

Jobba på den branchen och gör commits till den.

git add [file]
git commit -m "[descriptive message]"

Ta för vana att pusha efter varje commit

git push

När featuren är färdig så vill du uppdatera din feature-branch med det senaste ändringarna från master så att det inte blir några krockar med ändringarna.

git checkout master
git pull
git checkout [branch-name]
git merge master

Här kan det eventuellt bli merge-konflikter som måste lösas innan du fortsätter.
Fortsätt med en **git push** efter att du genomfört din **git merge**

När din feature-branch är uppdaterad med den senaste masterversionen så fortsätter vi med att merga featuren till master branchen.

git checkout master
git pull

Om det tillkommit ändringar sedan du gjorde en **git pull** nyss så måste du göra om de tidigare instruktionerna med **git checkout [branch-name]** och **git merge master**

git merge [branch-name]
git push

När featuren är klar avsluta med att ta bort branchen.
git branch -d [branch-name]

Lösa merge konflikter

Om en **git pull** eller **git merge** resulterar i en konflikt.

> “CONFLICT (content): Merge conflict ...”

Kör först en **git status** för att se i vilka filer det finns konflikter.
Ex:

> You have unmerged paths.
> (fix conflicts and run "git commit")
>
> Unmerged paths:
> (use "git add <file>..." to mark resolution)
>
> both modified: test.txt
>
> no changes added to commit (use "git add" and/or "git commit -a")
> ”

I detta fallet är konflikten i test.txt. Om man öppnar filen så kommer det att finnas en eller flera markeringar som markerar var konflikten är i filen.
Ex.

> <<<<<<< HEAD
> print(“Webgroup är bäst.”)
> =======
> print(“Webgroup is the best!”)
> >>>>>>> master

I detta fallet så är den undre ändringarna som fanns på masterbranchen och det övre är från featurebranchen.
För att lösa konflikten så väljer man ändringen som är korrekt eller tar från båda för att lösa problemet.
Ex.

> print(“Webgroup is the best!”)

När man löst alla konflikter i en fil så skriver man **git add [file]** där [file] i exemplet är test.txt. När man löst konflikterna i alla filer så kör man **git commit** vilket öppnar en texteditor med ett förifyllt commitmeddelande vilket bara är att spara och stänga för att commita. Avsluta med att skriva **git push**

Övriga tips:

Om ett commitmeddelande öppnas i en texteditor kan det ibland vara svårt att veta hur man skriver och sparar i för att ta sig därifrån.
Så om det står GNU nano högst upp i terminalen så är det bara att skriva meddelandet som vanligt och sedan Ctrl + X för att avsluta tryck Y när den frågar om du vill spara och sedan Enter.

Har man däremot lite “otur” så kan man hamna i VI.
För att kunna skriva här så trycker man först på A sedan kan man skriva sitt meddelande.
När man vill spara så trycker man först på ESC sedan skriver man :wq följt av ett Enter för att spara.