Introduktion til git og github

Af Henrik Sterner (hst@nextkbh.dk)

Git er et versionsstyringsværktøj, der gør det muligt at holde styr på ændringer i filer. Det er et meget udbredt værktøj, der bruges af mange udviklere til at holde styr på deres kode. Github er en online tjeneste, der gør det muligt at dele og samarbejde omkring projekter, der er versionsstyret med git.
Til at starte med introduceres git og github med en række øvelser, der skal give en forståelse for, hvordan git og github fungerer. Derefter skal I bruge git og github til at samarbejde omkring et projekt.

Installation af git

Git kan installeres på Windows, Mac og Linux. På Windows og Mac kan git installeres ved at downloade og køre en installationsfil. På Linux kan git installeres ved at køre en kommando i terminalen. Se https://git-scm.com/downloads for at finde installationsfilerne til de forskellige platforme.

Bemærk, at I skal bruge en terminal til at bruge git. Hvis I ikke ved noget om terminaler, så kig i næste slide og ellers spring den over.

Lidt om terminaler - del 1

En terminal er et program, der gør det muligt at skrive kommandoer til computeren. På Windows er det f.eks. programmet “cmd” eller “powershell”. På Mac er det programmet “terminal”. På Linux er det programmet “terminal” eller “konsole”.

I Windows startes terminal ved at trykke på Windows-tasten og skrive “cmd” eller “powershell”. I Mac startes terminal ved at trykke på “cmd” og “space” og skrive “terminal”. I Linux startes terminal ved at trykke på “ctrl”, “alt” og “t”.

Navigere i terminalen

I terminalen kan I navigere rundt i mapper og filer. I kan f.eks. skifte til en anden mappe ved at skrive “cd ”. I kan også se, hvilken mappe I er i ved at skrive “pwd”. I kan også se, hvilke filer og mapper der er i den mappe, I er i ved at skrive “dir” eller “ls”.

Vigtige kommandoer i terminalen

- cd: Skifter til en anden mappe. Eksempel: cd Documents
- cd .. : Skifter til den mappe, der indeholder den mappe, I er i. Eksempel: cd ..
- pwd: Viser, hvilken mappe I er i. Eksempel: pwd
- dir: Viser, hvilke filer og mapper der er i den mappe, I er i. Eksempel: dir
- ls: Viser, hvilke filer og mapper der er i den mappe, I er i. Eksempel: ls
- mkdir: Opretter en mappe. Eksempel: mkdir Documents

Vigtige kommandoer i terminalen - del 2

- rmdir: Sletter en mappe. Eksempel: rmdir Documents
- touch: Opretter en fil. Eksempel: touch hello.py
- rm: Sletter en fil. Eksempel: rm hello.py
- mv: Flytter en fil. Eksempel: mv hello.py Documents
- cp: Kopierer en fil. Eksempel: cp hello.py Documents

Opret en bruger på github

For at kunne bruge github skal I oprette en bruger på www.github.com. Det er gratis at oprette en bruger.

Efter det er vi klar til at komme i gang med at bruge git og github.

Fordele ved at bruge git og github

Git og github har en række fordele, der gør det til et godt værktøj til at samarbejde omkring kode. Nogle af fordelene er:

- Det er nemt at holde styr på ændringer i kode
- Det er nemt at samarbejde omkring kode
- Det er nemt at se, hvem der har lavet hvilke ændringer
- Det er nemt at gå tilbage til en tidligere version af koden
- Det er nemt at se, hvilke ændringer der er lavet siden sidste version
- Det er nemt at se, hvilke ændringer der er lavet siden en bestemt version
- Det er nemt at se, hvilke ændringer der er lavet af en bestemt person

Øvelse 1: Opret et repository på github

Til at starte med skal I oprette et repository på github. Et repository er et sted, hvor I kan gemme jeres kode. I skal oprette et repository til at gemme jeres kode til øvelserne i. I skal også oprette et repository til at gemme jeres kode til projektet i.

1. Log ind på github
2. Klik på plusset øverst til højre og vælg "New repository"
3. Vælg et navn til jeres repository. Det kan f.eks. være "gitintro".
4. Vælg om repositoryet skal være offentligt eller privat. Hvis det er privat, kan kun I se det. Hvis det er offentligt, kan alle se det.
5. Klik på "Create repository"

Tillykke du har nu oprettet et repository på github :-).

Øvelse 2: Opret et repository på din computer

Nu skal I oprette et repository på jeres computer. Det repository skal I bruge til at gemme jeres kode i. I skal oprette et repository til at gemme jeres kode til øvelserne i. I skal også oprette et repository til at gemme jeres kode til projektet i.

1. Åbn en terminal
2. Skift til den mappe, hvor I vil gemme jeres kode. Det kan f.eks. være "Documents" eller "Dokumenter".
3. Opret en mappe til jeres kode. Det kan f.eks. være "gitintro".
4. Skift til den mappe, der indeholder jeres kode. Det kan f.eks. være "gitintro".

Øvelse 2 fortsat: git init og git status

Nu skal I oprette et repository i den mappe, I har oprettet til jeres kode. I skal bruge kommandoen `git init` til at oprette et repository. I skal også bruge kommandoen `git status` til at se status for jeres repository.

5. Kør kommandoen `git init`. Det opretter et repository i den mappe, I er i.
6. Kør kommandoen `git status`. Den viser status for jeres repository. I skulle gerne se noget i stil med:

```
On branch master

No commits yet

nothing to commit (create/copy files and use "git add" to t
```

Øvelse 3: Tilføj en fil til jeres repository

Nu skal I tilføje en fil til jeres repository. I skal bruge kommandoen `git add` til at tilføje filen til jeres repository. I skal også bruge kommandoen `git status` til at se status for jeres repository.

1. Opret en fil i den mappe, I har oprettet til jeres kode. Det kan f.eks. være en fil med navnet "hello.py".
2. Skriv noget kode i filen. Det kan f.eks. være `print("Hello world")`.
3. Kør kommandoen `git status`. I skulle gerne se noget i stil med: `On branch master No commits yet Untracked files: (use "git add <file>..." to include in what will be committed) hello.py nothing added to commit but untracked files present (use "git add" to track)`

Øvelse 3 fortsat: git add og git status

4. Kør kommandoen `git add hello.py`. Det tilføjer filen til jeres repository.
5. Kør kommandoen `git status`. I skulle gerne se, at filen er blevet tilføjet til jeres repository.

Øvelse 3 fortsat: git commit

Nu skal I lave et commit. Et commit er en samling af ændringer, der er lavet i jeres kode. I skal bruge kommandoen `git commit` til at lave et commit. I skal også bruge kommandoen `git status` til at se status for jeres repository.

6. Kør kommandoen `git commit`. Det åbner en editor, hvor I skal skrive en besked til jeres commit. I skal skrive en besked, der beskriver, hvad I har lavet i jeres kode. Det kan f.eks. være "Added hello.py".

Øvelse 3 fortsat: git status

7. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.
8. Kør kommandoen `git log`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet.
9. Kør kommandoen `git log --oneline`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. Hash'en er en unik identifikation af commit'et.

Øvelse 3 fortsat: git graph

10. Kør nu kommandoen `git graph`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.

Øvelse 4: Forbind jeres repository til github

Forbind jeres repository på jeres computer med jeres repository på github. I skal bruge kommandoen `git remote add origin <url>`. I skal også bruge kommandoen `git remote -v` til at se, om jeres repository på jeres computer er forbundet med jeres repository på github. Herunder er et eksempel på, hvordan I kan gøre det:

1. Kør kommandoen `git remote add origin <url>`. Denne kommado forbinder jeres repository på jeres computer med jeres repository på github. I skal erstatte‘ med url'en til jeres repository på github. I skal bruge url'en, der står under "Quick setup" på jeres repository på github.
2. Kør kommandoen ‘git remote -v. Denne kommando viser, om jeres repository på jeres computer er forbundet med jeres repository på github.

Øvelse 5: push jeres fil til github

Nu skal I pushe jeres fil til github. Det betyder, at I skal "uploade" jeres fil til github. I skal bruge kommandoen `git push` til at pushe jeres fil til github. I skal også bruge kommandoen `git status` til at se status for jeres repository.

1. Kør kommandoen `git push`. Denne kommando pushe jeres fil til github. I skal muligvis skrive jeres brugernavn og password til github. I skal også muligvis skrive `git push --set-upstream origin master` i stedet for `git push`.
2. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.
3. Gå ind på jeres repository på github. I skulle gerne se, at jeres fil er blevet tilføjet til jeres repository på github.

Øvelse 6: Lad os lave nogle ændringer i jeres fil

Nu skal I lave nogle ændringer i jeres fil. I skal bruge kommandoen `git status` til at se status for jeres repository. I skal også bruge kommandoen `git diff` til at se, hvilke ændringer I har lavet i jeres fil.

1. Åbn filen i jeres editor. Det kan f.eks. være "hello.py".
2. Lav nogle ændringer i filen. Det kan f.eks. være `print("Hello world")` til `print("Hello world!")`.
3. Kør kommandoen `git status`. I skulle gerne se, at der er nogle ændringer i jeres kode.
4. Kør kommandoen `git diff`. I skulle gerne se, hvilke ændringer I har lavet i jeres kode.
5. Kør kommandoen `git add hello.py`. Denne kommando tilføjer jeres ændringer til jeres repository.

Øvelse 6 fortsat: git status og git diff

6. Kør kommandoen `git status`. I skulle gerne se, at jeres ændringer er blevet tilføjet til jeres repository.
7. Kør kommandoen `git diff`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.
8. Kør kommandoen `git commit`. Denne kommando laver et commit med jeres ændringer. I skal skrive en besked, der beskriver, hvad I har lavet i jeres kode. Det kan f.eks. være "Changed hello.py".
9. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.

Øvelse 7: push jeres ændringer til github

Nu skal I pushe jeres ændringer til github. Det betyder, at I skal "uploade" jeres ændringer til github. I skal bruge kommandoen `git push` til at pushe jeres ændringer til github. I skal også bruge kommandoen `git status` til at se status for jeres repository.

1. Kør kommandoen `git push`. Denne kommando pushe jeres ændringer til github. I skal muligvis skrive jeres brugernavn og password til github. I skal også muligvis skrive `git push --set-upstream origin master` i stedet for `git push`.
2. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.
3. Gå ind på jeres repository på github. I skulle gerne se, at jeres ændringer er blevet tilføjet til jeres repository på github.

Øvelse 7 fortsat: git diff og git log

4. Kør kommandoen `git log --oneline`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.

Opsummering indtil videre

I har lært: - Hvad git og github er - Hvordan I opretter et repository på github - Hvordan I opretter et repository på jeres computer - Hvordan I tilføjer en fil til jeres repository - Hvordan I laver et commit - Hvordan I forbinder jeres repository på jeres computer med jeres repository på github - Hvordan I pusher jeres fil til github - Hvordan I laver ændringer i jeres fil - Hvordan I pusher jeres ændringer til github - Hvordan I ser, hvilke ændringer I har lavet i jeres fil - Hvordan I ser, hvilke ændringer der er lavet siden sidste commit

Vigtige begreber

- Repository: Et sted, hvor I kan gemme jeres kode. Det kan f.eks. være på jeres computer eller på github.
- Workspace: Den mappe, hvor I arbejder med jeres kode
- Index: En liste over ændringer, der skal committes. Det er git init og git add, der tilføjer ændringer til index.
- Commit: En samling af ændringer, der er lavet i jeres kode. Koden i et commit er gemt i et repository. Det er git commit, der laver et commit.
- Push: At uploade jeres kode til github. Det er git push, der pusher jeres kode til github.
- Ændringer: Ændringer, der er lavet i jeres kode. Det er git diff, der viser ændringerne.
- Status: Status for jeres repository. Det er git status, der viser status for jeres repository.
- Log: Log over commits. Det er git log, der viser log over commits.
- Hash: En unik identifikation af et commit. Det er git log, der viser hash'en for et commit.

Øvelse 8: Lav en klon af jeres repository

Nu skal I lave en klon af jeres repository. Det betyder, at I skal kopiere jeres repository fra github til jeres computer. I skal bruge kommandoen `git clone` til at lave en klon af jeres repository. I skal også bruge kommandoen `git status` til at se status for jeres repository.

1. Åbn en terminal
2. Skift til den mappe, hvor I vil gemme jeres kode. Det kan f.eks. være "Documents" eller "Dokumenter".
3. Kør kommandoen `git clone <url>`. Denne kommando kopiere jeres repository fra github til jeres computer. I skal erstatte `<url>` med url'en til jeres repository på github. I skal bruge url'en, der står under "Quick setup" på jeres repository på github.
4. Skift til den mappe, der indeholder jeres repository. Det kan f.eks. være "gitintro".
5. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.

Øvelse 8 fortsat: Lav en klon af jeres repository

6. Kør kommandoen `git log --oneline`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.
7. Kør kommandoen `git graph`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.
8. Kør kommandoen `git remote -v`. I skulle gerne se, at jeres repository på jeres computer er forbundet med jeres repository på github.
9. Kør kommandoen `git remote show origin`. I skulle gerne se, at jeres repository på jeres computer er forbundet med jeres repository på github.

Øvelse 9: Gå tilbage til en tidligere version af jeres kode

Nu skal I gå tilbage til en tidligere version af jeres kode. I skal bruge kommandoen `git checkout` til at gå tilbage til en tidligere version af jeres kode. I skal også bruge kommandoen `git status` til at se status for jeres repository.

1. Kør kommandoen `git log --oneline`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.
2. Kør kommandoen `git checkout <hash>`. Denne kommando går tilbage til en tidligere version af jeres kode. I skal erstatte `<hash>` med hash'en for det commit, I vil gå tilbage til. I skal bruge hash'en, der står under "commit" på jeres repository på github.
3. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.

Øvelse 10: Gå tilbage til den nyeste version af jeres kode

Nu skal I gå tilbage til den nyeste version af jeres kode. I skal bruge kommandoen `git checkout` til at gå tilbage til den nyeste version af jeres kode. I skal også bruge kommandoen `git status` til at se status for jeres repository.

1. Kør kommandoen `git log --oneline`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.
2. Kør kommandoen `git checkout master`. Denne kommando går tilbage til den nyeste version af jeres kode.
3. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.

Øvelse 11: Lav en gren af jeres kode

Nu skal I lave en gren af jeres kode. Det betyder, at I skal lave en kopi af jeres kode. I skal bruge kommandoen `git branch` til at lave en gren af jeres kode. I skal også bruge kommandoen `git branch -v` til at se, hvilke grene der er i jeres repository. I skal også bruge kommandoen `git status` til at se status for jeres repository.

1. Kør kommandoen `git branch <navn>`. Denne kommando laver en gren af jeres kode. I skal erstatte `<navn>` med navnet på jeres gren. Det kan f.eks. være "test".
2. Kør kommandoen `git branch -v`. Denne kommando viser, hvilke grene der er i jeres repository.
3. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.

Øvelse 11 fortsat: Lav en gren af jeres kode

4. Kør kommandoen `git graph`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.
5. Kør kommandoen `git checkout <navn>`. Denne kommando skifter til jeres gren. I skal erstatte `<navn>` med navnet på jeres gren. Det kan f.eks. være "test".

Øvelse 11 fortsat: Lav en gren af jeres kode

6. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.
7. Kør kommandoen `git graph`. I skulle gerne se, at der ikke er nogle commits i jeres gren.
8. Kør kommandoen `git checkout master`. Denne kommando skifter tilbage til master grenen.

Øvelse 12 git merge to grene

Nu skal I merge jeres gren med master grenen. Det betyder, at I skal tilføje jeres ændringer fra jeres gren til master grenen. I skal bruge kommandoen `git merge` til at merge jeres gren med master grenen. I skal også bruge kommandoen `git status` til at se status for jeres repository.

1. Kør kommandoen `git checkout <navn>`. Denne kommando skifter til jeres gren. I skal erstatte `<navn>` med navnet på jeres gren. Det kan f.eks. være "test".
2. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.
3. Kør kommandoen `git graph`. I skulle gerne se, at der ikke er nogle commits i jeres gren.
4. Kør kommandoen `git checkout master`. Denne kommando skifter tilbage til master grenen.
5. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.

Øvelse 12 fortsat: git merge to grene

6. Kør kommandoen `git graph`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.
7. Kør kommandoen `git merge <navn>`. Denne kommando merger jeres gren med master grenen. I skal erstatte `<navn>` med navnet på jeres gren. Det kan f.eks. være "test".
8. Kør kommandoen `git status`. I skulle gerne se, at der ikke er nogle ændringer i jeres kode.
9. Kør kommandoen `git graph`. I skulle gerne se, at der er lavet et commit. I skulle også gerne se, hvem der har lavet commit'et og hvornår det er lavet. I skulle også gerne se, at der er en hash for commit'et. I skulle også gerne se, at der er en pil, der peger på commit'et. Pilen viser, at commit'et er det nyeste commit.

Opsummering: I har lært

- Hvordan I laver en klon af jeres repository
- Hvordan I går tilbage til en tidligere version af jeres kode
- Hvordan I går tilbage til den nyeste version af jeres kode
- Hvordan I laver en gren af jeres kode
- Hvordan I merger to grene
- Hvordan I ser, hvilke grene der er i jeres repository
- Hvordan I skifter mellem grene
- Hvordan I ser, hvilke ændringer der er lavet siden sidste commit
- Hvordan I ser, hvilke ændringer der er lavet siden en bestemt version

Vigtige begreber

- Gren: En kopi af jeres kode. Det er git branch, der laver en gren.
- Master gren: Den gren, der er standard. Det er git branch, der laver en master gren.
- Checkout: At skifte mellem grene. Det er git checkout, der skifter mellem grene.
- Merge: At tilføje ændringer fra en gren til en anden gren. Det er git merge, der merger to grene.
- Diff: Ændringer, der er lavet i en fil. Det er git diff, der viser ændringerne.
- Log: Log over commits. Det er git log, der viser log over commits.
- Hash: En unik identifikation af et commit. Det er git log, der viser hash'en for et commit.
- Pile: Pile, der peger på et commit. Det er git log, der viser pile, der peger på et commit.
- Status: Status for jeres repository. Det er git status, der viser status for jeres repository.

Vigtigste kommandoer i git - del 1

- git init: Opretter et repository
- git add: Tilføjer ændringer til index. Eksempel: git add hello.py
- git commit: Laver et commit. Eksempel: git commit
- git push: Pusher jeres kode til github. Eksempel: git push
- git clone: Laver en klon af jeres repository. Eksempel: git clone
- git pull: Henter ændringer fra et remote repository og merger dem med jeres kode. Eksempel: git pull
- git fetch: Henter ændringer fra et remote repository. Eksempel: git fetch

Til forskel fra pull merger fetch ikke ændringerne med jeres kode.

Vigigste kommandoer i git - del 2

- git checkout: Skifter mellem grene. Eksempel: git checkout
- git merge: Merger to grene. Eksempel: git merge
- git diff: Viser ændringer i en fil. Eksempel: git diff
- git log: Viser log over commits. Eksempel: git log
- git status: Viser status for jeres repository. Eksempel: git status
- git remote: Viser remote repositories. Eksempel: git remote
- git remote add: Tilføjer et remote repository. Eksempel: git remote add origin
- git remote show: Viser et remote repository. Eksempel: git remote show origin
- git remote -v: Viser remote repositories. Eksempel: git remote -v

Vigtigste kommandoer i git - del 3

- git branch: Laver en gren. Eksempel: git branch
- git branch -v: Viser grene. Eksempel: git branch -v
- git graph: Viser en graf over commits. Eksempel: git graph
- git log –oneline: Viser log over commits. Eksempel: git log –oneline
- git log –oneline –graph: Viser en graf over commits. Eksempel: git log –oneline –graph
- git log –oneline –graph –all: Viser en graf over commits. Eksempel: git log –oneline –graph –all
- git log –oneline –graph –all –decorate: Viser en graf over commits. Eksempel: git log –oneline –graph –all –decorate
- git log –oneline –graph –all –decorate –color: Viser en graf over commits. Eksempel: git log –oneline –graph –all –decorate –color

Vigigste kommandoer i git - del 4

- git log –oneline –graph –all –decorate –color –author=: Viser en graf over commits. Eksempel: git log –oneline –graph –all –decorate –color –author=. Det viser kun commits, der er lavet af <navn>.
- git log –oneline –graph –all –decorate –color –author= –since=: Viser en graf over commits. Eksempel: git log –oneline –graph –all –decorate –color –author= –since=. Det viser kun commits, der er lavet af <navn> siden <dato>.

Vigigste kommandoer i git - del 5

- git log –oneline –graph –all –decorate –color –author= –since= – : Viser en graf over commits. Eksempel: git log –oneline –graph –all –decorate –color –author= –since= – . Det viser kun commits, der er lavet af `<navn>` siden `<dato>` i `<fil>`.
- git log –oneline –graph –all –decorate –color –author= –since= – ..: Viser en graf over commits. Eksempel: git log –oneline –graph –all –decorate –color –author= –since= – ... Det viser kun commits, der er lavet af `<navn>` siden `<dato>` i `<fil>..<fil>`.

Git workflow

Git workflow er en måde at bruge git på. Der er mange forskellige måder at bruge git på. Her er et eksempel på et workflow:

1. Lav en gren af master grenen
2. Lav ændringer i koden
3. Tilføj ændringerne til index
4. Lav et commit
5. Push ændringerne til github
6. Lav en pull request
7. Merge pull requesten med master grenen
8. Slet grenen

Brug git til vs code eller Github desktop

I kan også bruge git til vs code eller Github desktop. Det er programmer, der gør det nemt at bruge git. I kan finde vs code på https://code.visualstudio.com/ og Github desktop på https://desktop.github.com/.

Tak for jeres opmærksomhed

Skriv til mig på hst@nextkbh.dk hvis I har spørgsmål eller kommentarer til dette forløb