Opstart

# Kom godt i gang

For at kunne bruge git og github er der en række ting der skal på plads først.

1. Have installeret GIT -> <https://git-scm.com>
2. Have en konto på GITHUB -> [www.github.com](http://www.github.com)

HUSK første gang git skal bruges skal følgende gøres, hvis ikke så bliver man spurgt når man prøver at lave et commit. $ git config --global user.name "John Doe"

$ git config --global user.email johndoe@example.com

Husk første gang man pusher til et repo (efter et clone/init) så skal følgende gøres for at ens git ”tracker” remote:

$ git push –set-upstream <remote> <branch>

(git push –set-upstream origin master)

Der kan forekomme problemmer med at, git ikke ser remote og lokals historik som værende relaterede, hvis dette forekommer under opsætningen af et projekt, så kan følgende bruges:

$ git pull --allow-unrelated-histories

Teori om Git & Github

# Git

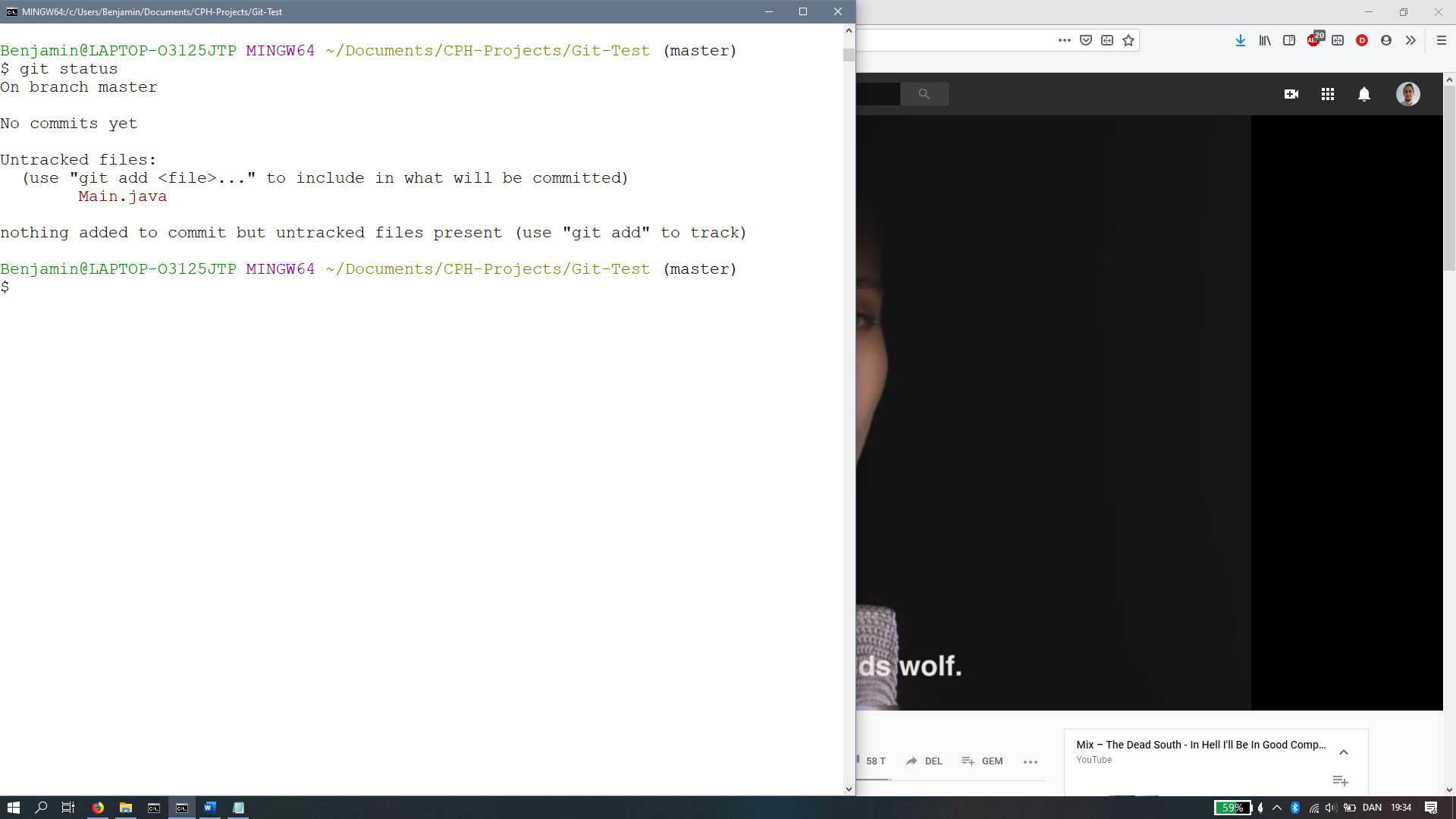
Git gør det muligt at dele kode med alle gruppens medlemmer, og sikre at der ikke bliver tabt vigtige elementer ved deling. Ved brug af Google Drive / OneDrive, skal man manuelt indsætte nye metoder og klasser og holde styr på at der ikke er noget der bliver fjernet eller ændret med noget forkert. Git hjælper med denne proces og håndterer meget automatisk.

Når vi arbejder med git så er ’commit’ et vigtigt term, man er nød til at forstå for at kunne udnytte git korrekt. I bruger vi ’commits’ til at gemme versioner af vores filer. Det vil sige at når vi danner et commit så gemmer vi den filernes tilstand.

Git arbejder med filer, ved at kategoriserer dem i følgende kategorier.

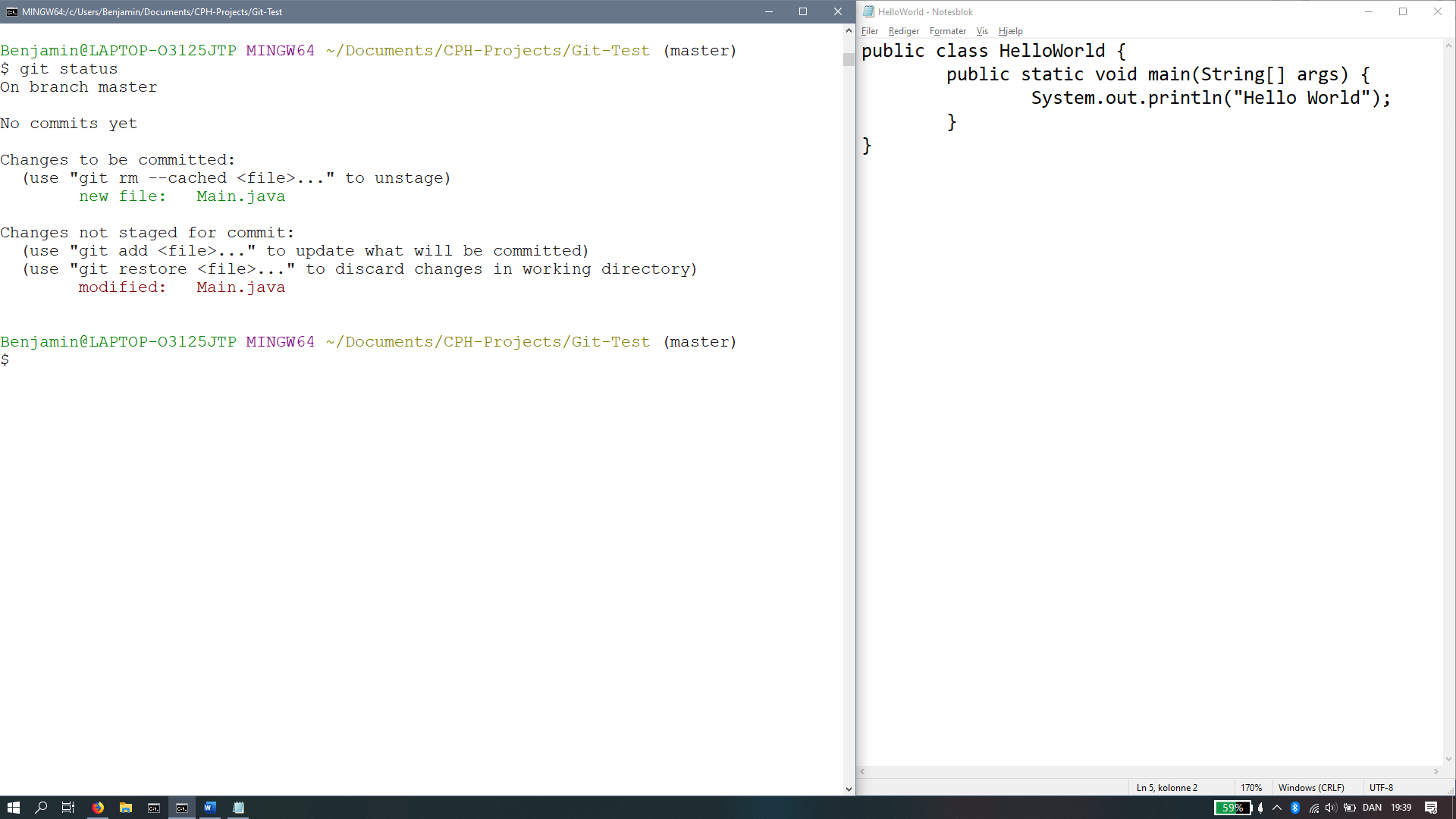
## ’Untracted files’

Disse filer bliver ikke holdt øje med af Git, det vil sige de er altså ikke inkluderet i din versionsstyring og vil derfor ikke blive taget med når git skal gemme en ’version’



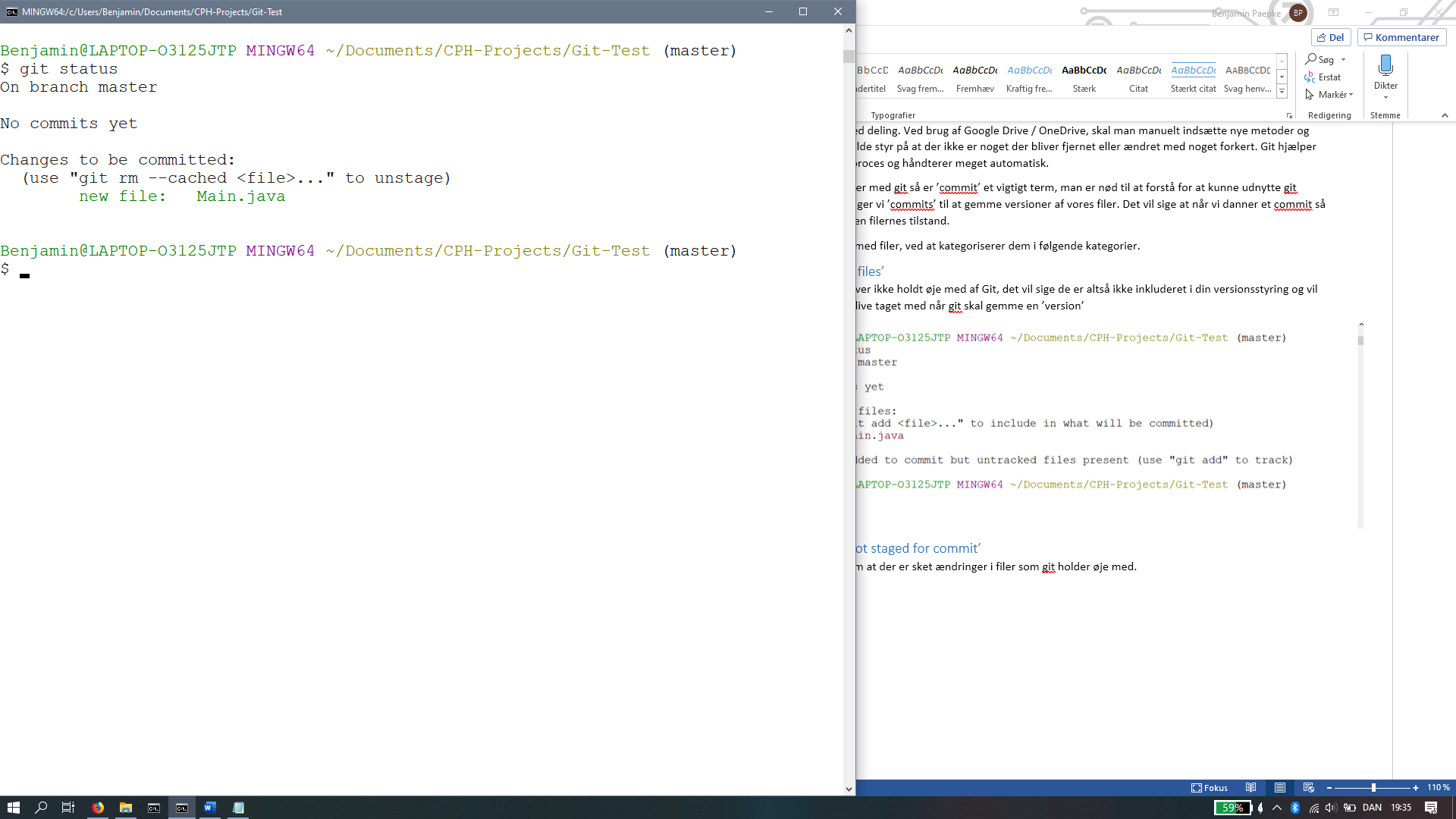
## ‘Changes not staged for commit’

Her er tale om at der er sket ændringer i filer som git holder øje med.

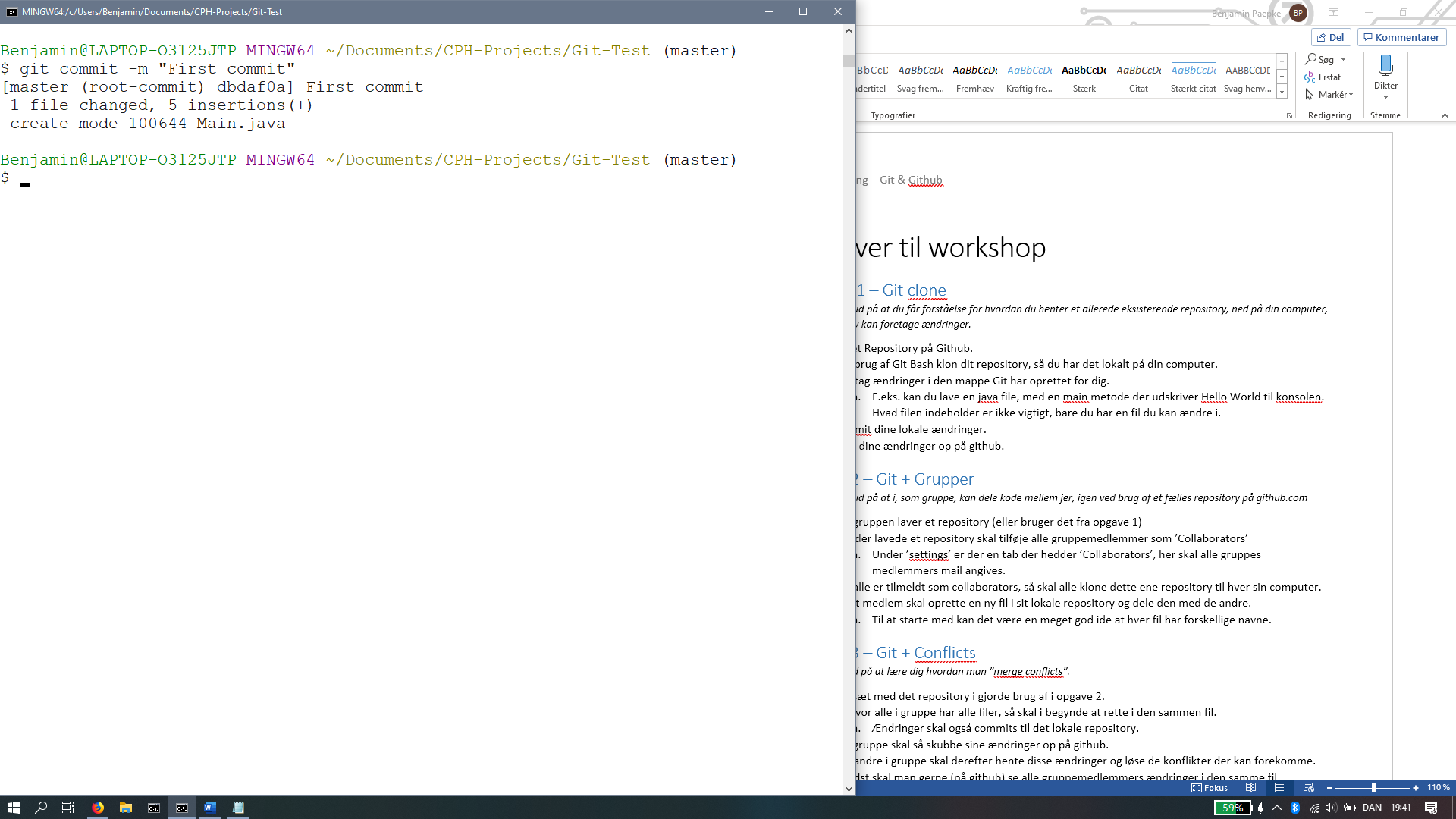


## ’Changes to be commited’

Her er der tale om at følgende filer er ændret siden sidste commit, men at de er inkluderet i det næste kommende commit.



## Commit



# GitHub

Github står i bund og grund kun for at let at kunne dele sin versionsstyret kode.

Det vil sige at vi bruger github som host for vores kode.

Opgaver til workshop

# Opgaver 1 – Git clone

*Opgaven går ud på at du får forståelse for hvordan du henter et allerede eksisterende repository, ned på din computer, hvorfra du selv kan foretage ændringer.*

1. Lav et Repository på Github.
2. Ved brug af Git Bash klon dit repository, så du har det lokalt på din computer.
3. Foretag ændringer i den mappe Git har oprettet for dig.
   1. F.eks. kan du lave en java file, med en main metode der udskriver Hello World til konsolen. Hvad filen indeholder er ikke vigtigt, bare du har en fil du kan ændre i.
4. Commit dine lokale ændringer.
5. Skub dine ændringer op på github.

# Opgave 2 – Git + Grupper

*Opgaven går ud på at i, som gruppe, kan dele kode mellem jer, igen ved brug af et fælles repository på github.com*

1. En i gruppen laver et repository (eller bruger det fra opgave 1)
2. Den der lavede et repository skal tilføje alle gruppemedlemmer som ’Collaborators’
   1. Under ’settings’ er der en tab der hedder ’Collaborators’, her skal alle gruppes medlemmers mail angives.
3. Når alle er tilmeldt som collaborators, så skal alle klone dette ene repository til hver sin computer.
4. Hvert medlem skal oprette en ny fil i sit lokale repository og dele den med de andre.
   1. Til at starte med kan det være en meget god ide at hver fil har forskellige navne.

# Opgave 3 – Git + Conflicts

*Opgave går ud på at lære dig hvordan man ”merge conflicts”.*

1. Fortsæt med det repository i gjorde brug af i opgave 2.
2. Nu hvor alle i gruppe har alle filer, så skal i begynde at rette i den sammen fil.
   1. Ændringer skal også commits til det lokale repository.
3. En i gruppe skal så skubbe sine ændringer op på github.
4. Alle andre i gruppe skal derefter hente disse ændringer og løse de konflikter der kan forekomme.
5. Til sidst skal man gerne (på github) se alle gruppemedlemmers ændringer i den samme fil.

# Opgave 4 – Git Init – EKSTRA OPGAVE (!!)

*Opgave går ud på at bruge en lokal kodebase som startpunkt, og derved skubbe al data i den ønskede mappe op på github uden at skulle klone et repository.*

1. Lav et repository på github.
2. Lav en mappe med navnet Test-Projekt, eller find på noget selv, og lav en fil i mappen som hedder HelloWorld.java
3. Fyld filen ud med en klasse, af samme navn, og lav samtidig en main metode
4. Initialiser mappen til brug med git.
   1. Git init, initialiserer den mappe du står i.
5. Skub nu filen på dit github repository.
   1. Når du har lave et commit og kalder git push, så skal du blot følge vejledningen git selv giver.