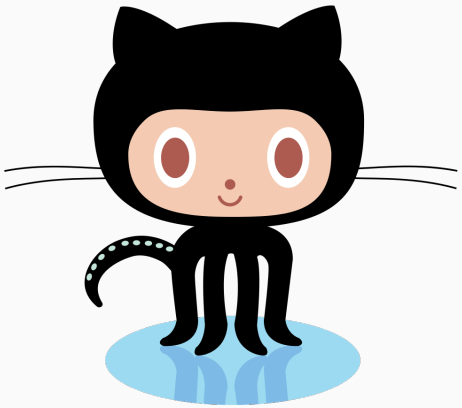
Git 101

GitGoing



Alexander Alsing
Magnus Aagaard Sørensen

10. marts 2016

Aarhus School of Engineering

Introduktion

Hvorfor git?

Et projekt kræver deling af filer

Hvorfor git?

Et projekt kræver deling af filer

- BlackBoard
  - Men hvem kan finde ud af det?

Hvorfor git?

Et projekt kræver deling af filer

- BlackBoard
  - Men hvem kan finde ud af det?
- Den gamle skole
  - Facebook
  - Email

Hvorfor git?

Et projekt kræver deling af filer

- BlackBoard
  - Men hvem kan finde ud af det?
- Den gamle skole
  - Facebook
  - Email
- De nye klienter
  - Dropbox
  - Google Drive

Hvorfor git?

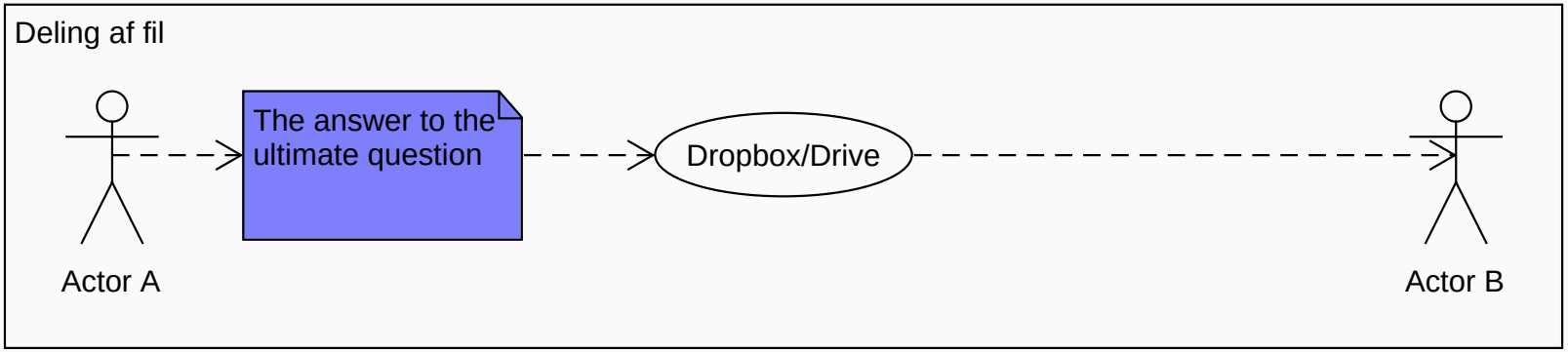
Et projekt kræver deling af filer

- BlackBoard
  - Men hvem kan finde ud af det?
- Den gamle skole
  - Facebook
  - Email
- De nye klienter
  - Dropbox
  - Google Drive

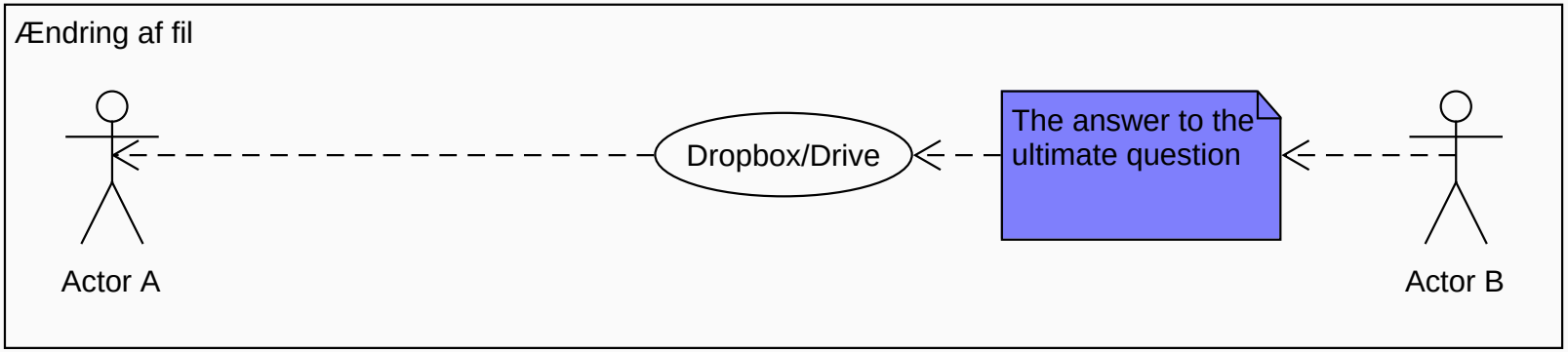
Problemer med

- Kollisioner når flere tilgår den samme fil
- Svært at holde effektivt styr på tidligere versioner

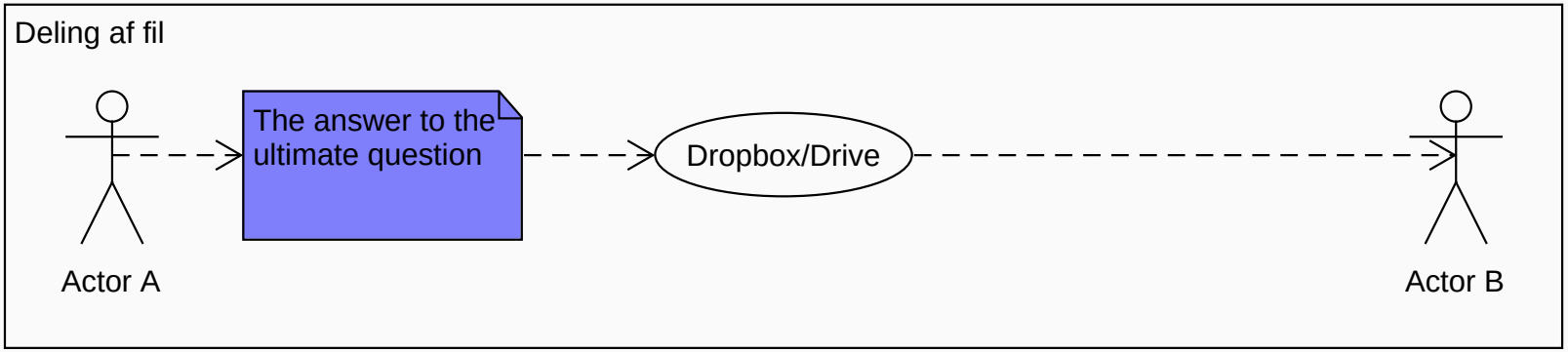
Problem med centralisering



Problem med centralisering



Problem med centralisering



Hvad sker der nu?

Problem med centralisering

Løsningen

GIT!

Hvad er git?

Git er et decentraliseret versionstyringssystem

Hvad er git?

Git er et decentraliseret versionstyringssystem

- git er værktøjet

Hvad er git?

Git er et decentraliseret versionstyringssystem

- git er værktøjet
- GitHub er den centrale sky

Hvad er en terminal?

Hvorfor terminal?

- Mangel på processorkraft
- Brug for et simpelt interface

Hvad er en terminal?

Hvorfor terminal?

- Mangel på processorkraft
- Brug for et simpelt interface

Hvorfor bruger vi den i dag?

- Kommandoer kan kun fortolkes på en måde
- Fjerne ”ligegyldig” information
- Ressourcer

Basal brug af terminal

Navigation

Basal brug af terminal

Navigation

- `cd`
  - `cd EnMappe/DerLigger/I/En/Mappe`
  - Tap completion

Basal brug af terminal

Navigation

- `cd`
  - `cd EnMappe/DerLigger/I/En/Mappe`
  - Tap completion
- `ls`

Basal brug af terminal

Navigation

- `cd`
  - `cd EnMappe/DerLigger/I/En/Mappe`
  - Tap completion
- `ls`
- `~`
  - `cd ~`
  - `cd C:\Users \DinBruger`

GitGoing

Lokal kopi (git clone)

Opret et lokalt repository som en kopi af projektet.

```
git clone URL
```

Laver en ny mappe med samme navn som projektet

Prøv selv!

`https://github.com/Quanalogy/GitGoing.git`

Se status (git status)

Se hvilke ændringer der vil komme med i næste revision

```
git status
```

Se status (git status)

Se hvilke ændringer der vil komme med i næste revision

```
git status
```

Fortæller om hver enkelt fil bliver tilføjet, modificeret eller slettet

Tilføj filer (git add)

`git add`, simplificeret sagt, tilføjer filer til projektet

- git holder kun øje med add'ed filer

Tilføj filer (git add)

`git add`, simplificeret sagt, tilføjer filer til projektet

- git holder kun øje med add'ed filer
- Tilføj nye filer med add

  ```
  git add [FILE1] [FILE2] ... [FILE_N]
  ```

Tilføj filer (git add)

`git add`, simplificeret sagt, tilføjer filer til projektet

- git holder kun øje med add'ed filer
- Tilføj nye filer med add

  ```
  git add [FILE1] [FILE2] ... [FILE_N]
  ```
- Gælder også for at tilføje eksisterende filer, der er modificeret siden sidst
- `git status`!

Fjerne/slet filer (git rm)

- Slet filer med rm

  `git rm (--cached) [FILE]`

  - `--cached` keyword bruges hvis den skal have fjernet tracked status, men man ønsker at beholde filen

- Kan også gøres via almindelig fil manager
- Brug `git status` for at se hvad der har ændret sig
- Skal du blot flytte eller omdøbe en fil, brug `git mv`

  `git mv [FILE_OLD_LOCATION] [FILE_NEW_LOCATION]`

Fjerne/slet filer (git rm)

- Slet filer med rm

  `git rm (--cached) [FILE]`

  - `--cached` keyword bruges hvis den skal have fjernet tracked status, men man ønsker at beholde filen

- Kan også gøres via almindelig fil manager
- Brug `git status` for at se hvad der har ændret sig
- Skal du blot flytte eller omdøbe en fil, brug `git mv`

  `git mv [FILE_OLD_LOCATION] [FILE_NEW_LOCATION]`

  `git mv [FILE_OLD_NAME] [FILE_NEW_NAME]`

Oprettelse af en ny revision (git commit)

Gem listen af ændringer som vist i status med en beskrivende tekst

- Gøres med

```
git commit -m "Besked"
```

Oprettelse af en ny revision (git commit)

Gem listen af ændringer som vist i status med en beskrivende tekst

- Gøres med

  ```
  git commit -m "Besked"
  ```

- Sørg for **ALTID** at skrive ordentlige beskeder, det gør arbejdet lettere senere hen

Oprettelse af en ny revision (git commit)

Gem listen af ændringer som vist i status med en beskrivende tekst

- Gøres med

  ```
  git commit -m "Besked"
  ```

- Sørg for **ALTID** at skrive ordentlige beskeder, det gør arbejdet lettere senere hen
- Glemmer man '-m' ryger man ind i vim, det ønsker man som regel ikke. Brug ':wq' for at slippe ud igen.
- Brug `git status`!

gitignore

- git har mulighed for altid at udelade filer fra dine projekter
- Smart, ingen ide i at smide ligegyldige filer frem og tilbage

gitignore

- git har mulighed for altid at udelade filer fra dine projekter
- Smart, ingen ide i at smide ligegyldige filer frem og tilbage
- GitHub kan lave det for dig!

GitHub

- GitHub samler projektet fra alle de lokale REPOs

GitHub

- GitHub samler projektet fra alle de lokale REPOs
- Fordele som studerende

GitHub

- GitHub samler projektet fra alle de lokale REPOs
- Fordele som studerende
- Nu til GitHub og dit (måske) første projekt på GitHub!

git push

Smider staged ændringer op på fælles REPO

git push

Smider staged ændringer op på fælles REPO

- `git push <remote> [branch]`

  Vil typisk være origin der er din remote - hvis du ikke har ændret på standard config

git push

Smider staged ændringer op på fælles REPO

- `git push <remote> [branch]`

  Vil typisk være origin der er din remote - hvis du ikke har ændret på standard config
- `-u` keyword (set upstream)

git pull

Henter andres ændringer ned

git pull

Henter andres ændringer ned

- `git pull <remote> [branch]`
- `<remote>` vil igen typisk være origin

Branches

Nu bliver det smart!

- Del projektet op i forskellige områder - Semesterprojekt 1:
  - Motorstyring
  - Lysstyring
  - Refleksdetektor
  - Lyd
  - Testing
  - W/E
- Branchens navn står altid i git bash i blå parentes efter stien

Branches: ny branch

Kommando for at lave en ny branch og skifte til den

```
git checkout -b [newBranchName]
```

Branches: ny branch

Kommando for at lave en ny branch og skifte til den

```
git checkout -b [newBranchName]
```

Prøv selv :)

Branches: skift branch

Skift branch uden `-b` keyword

```
git checkout [branchName]
```

Branches: skift branch

Skift branch uden `-b` keyword

```
git checkout [branchName]
```

Prøv selv at skifte til master branch og tilbage til branchen fra før

Branches: merge branch

Har du testet ændringer som er klar til master branch kan du merge dem ind i master branchen

Branches: merge branch

Har du testet ændringer som er klar til master branch kan du merge dem ind i master branchen

- Start med at skifte over til branchen som skal merges til (master)

Branches: merge branch

Har du testet ændringer som er klar til master branch kan du merge dem ind i master branchen

- Start med at skifte over til branchen som skal merges til (master)
- Herefter merges der over - branch skal være den branch som merges fra

  ```
  git merge [branch]
  ```

git log

Git log giver dig et overblik over alle snapshots (ringe i træet) der er lavet i løbet af projektet

- Når du commit'er
- Når du merger to branches (Da det er et commit)

?

Slides, cheatsheet etc. er tilgængelig på
`https://github.com/`