Versjonskontroll - Git og Github

Agenda

- Versjonskontroll: Hva og hvorfor
- Source Code Management og Git
- Github
- Git-funksjonalitet
- Anbefalt arbeidsflyt (team og hver enkelt utvikler)

Vi har et fundamentalt problem...

- ALLE GJØR FEIL...!
  - Sletter feil fil
  - Sletter feil kode
  - Finner ikke frem etter opprydning (mappestruktur, navngivning, osv.)
  - «Det funka jo tidligere ...!»
  - Glemmer hvorfor vi gjorde noe på en spesifikk måte
  - En tiltenkt fix gjør problemet verre
  - ...
- Gang frekvens av feil med antall personer i teamet...

If I Could Turn Back Time ...

- Når det skjer noe feil ønsker vi ofte å kunne gå tilbake
  - «Jeg vil bare at det skal funke igjen...»
  - «Jeg skulle ønske jeg gjorde det annerledes...»
  - «Hvordan fikset jeg dette sist?»
  - «Hvordan endte egentlig jeg opp her...?»

- Løsningen: Versjonskontroll

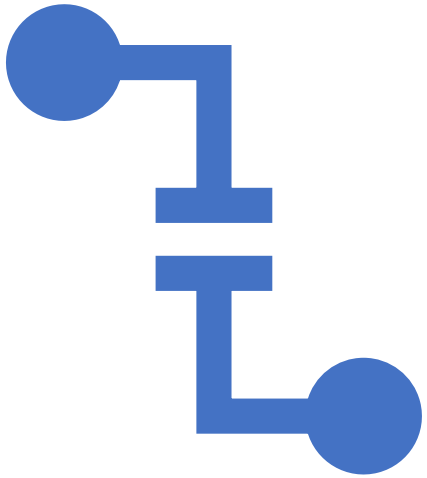
Versjonskontroll

- Lar oss spore:
  - HVA som er endret
  - NÅR det ble endret
  - HVEM som endret
  - HVORFOR det ble endret (viktig!)
- Vi kan revertere til tidligere versjoner
- Altså: En backup i form av en tidslinje med endringer

Source Code Management

- Kildekoden er et sentralt bruksområde for versjonskontroll
  - Merk at versjonskontroll kan gjelde mer

- Source Code Management Systemer
  - Fungerer som en sentral og felles lagringsplass for prosjektet
    - Kildekode, filer, config-filer, biblioteker, verktøy, osv.
    - Er i praksis en mappe med filer
    - Kalles typisk et «repository» (forkortes repo)
  - Støtter parallelt arbeid mellom flere utviklere



Source Code Management

- Fire generelle features:
  - Code transfer
    - Uviklere kan laste ned filene for arbeid, og laste dem opp når de er ferdig
  - Version storage and retrieval
    - Tidligere versjoner av prosjektfiler lagres og kan hentes
  - Version information
    - Informasjon om de forskjellige versjonene lagres og kan hentes
  - Branching and merging
    - Utviklere kan lage branches for å jobbe parallelt uten å forstyrre andre. Branches kan merges for å slå sammen arbeidet til ett

(Sommerville, 2021)

Git

- Source Code Management Systemer har eksistert siden 1970-tallet
- Centralized Source Code Management var dominant i mange år
  - Utviklere check-er filer ut og inn
  - En advarsel blir gitt hvis en fil allerede er check-et ut av noen andre
  - Filer må lastes ned over nettet (avhengig av tilgjengelighet)

- Men: I 2005 ble Git utviklet av Linus Thorvalds
  - Distributed Source Code Management
  - Utviklere kopierer ikke bare filer, men laster ned hele «repositoriet» til sin PC
  - Utviklere kan jobbe uten tilgang til nettet (sikrere og raskere)
  - Det er tryggere å eksperimentere lokalt
  - Main/Master branch blir introdusert

(Sommerville, 2021)

Github

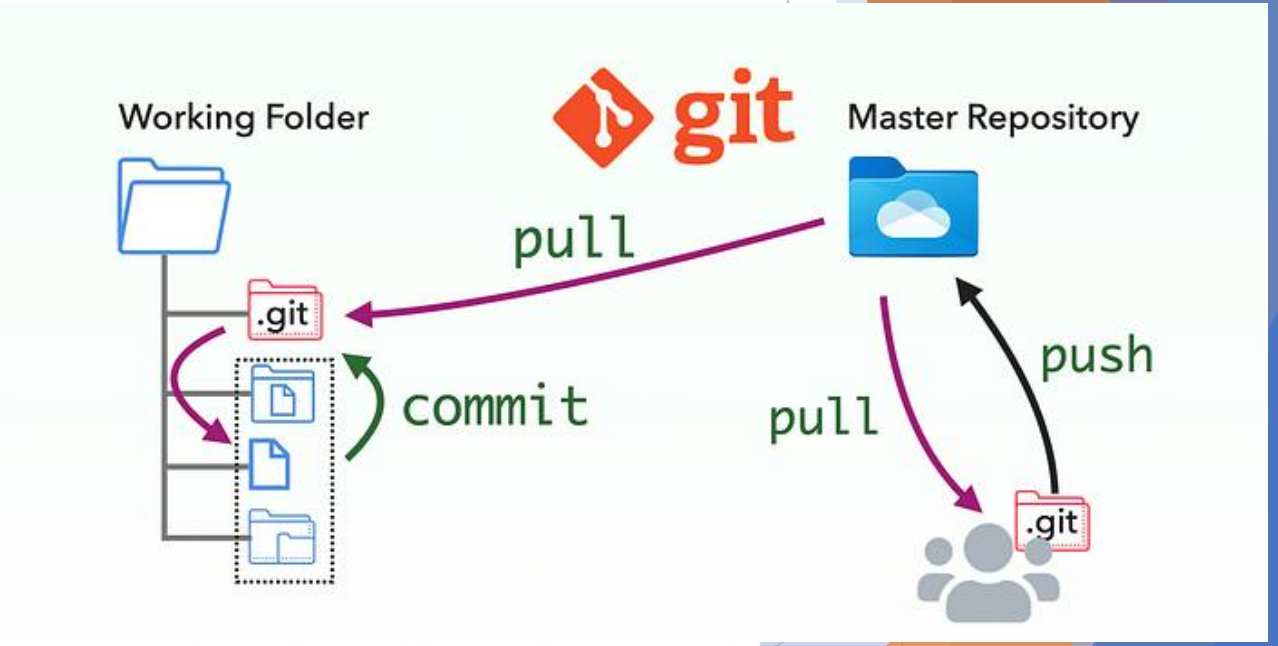
- GitHub er en av flere tjenester for Git-hosting
  - Hoster repos og gir verktøy for administrasjon
    - Tilgang, oversikt, samarbeid ...
  - Gratis
  - Utviklere kan jobbe på mange så mange repos de ønsker
  - Man kan lage sin egen versjon av et eksisterende prosjekt ved å "fork"-e
    - Typisk at noen lager en egentilpasset versjon av et open source prosjekt
  - Funker også som et alternativ til enkeltpersoners bruk av skylagringstjenester
    - Og er bedre egnet til utviklingsprosjekter (større kontroll)

Bruk av GitHub og Git

- Github: https://github.com/
- Hvert gruppemedlem SKAL benytte en Github-konto
- Ett gruppemedlem bør lage et **public** repo for prosjektet
  - Foreslått navn: SET_Gruppe_[deres gruppenummer]
  - Inviter andre gruppemedlemmer som collaborators
  - Dette vil være "prosjektmappen" deres
  - Må være public slik at vi som veileder også kan se

Git-funksjonalitet

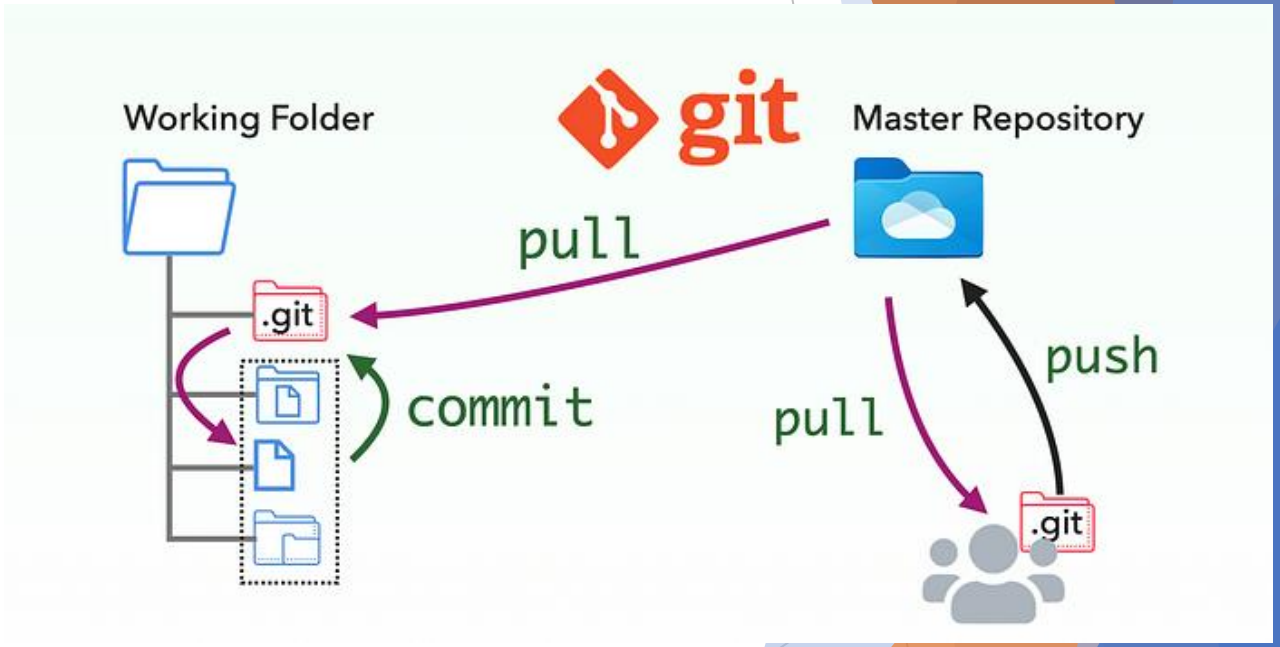
- Git har en rekke funksjonalitet som er viktig å forstå for å bruke det
- I utgangspunktet er dette CLI kommandoer som benyttet gjennom terminal
  - https://ohshitgit.com/
- … men det finnes i dag mange GUI-applikasjoner som gjør bruk av Git mer intuitivt
  - GitKraken
  - Github Desktop
  - Integrert i IntelliJ IDEA
  - Git GUI
  - Osv.



https://zhiminzhan.medium.com/10-minutes-guide-to-git-version-control-for-testers-f58e059bb5e7

Git-funksjonalitet - Kloning

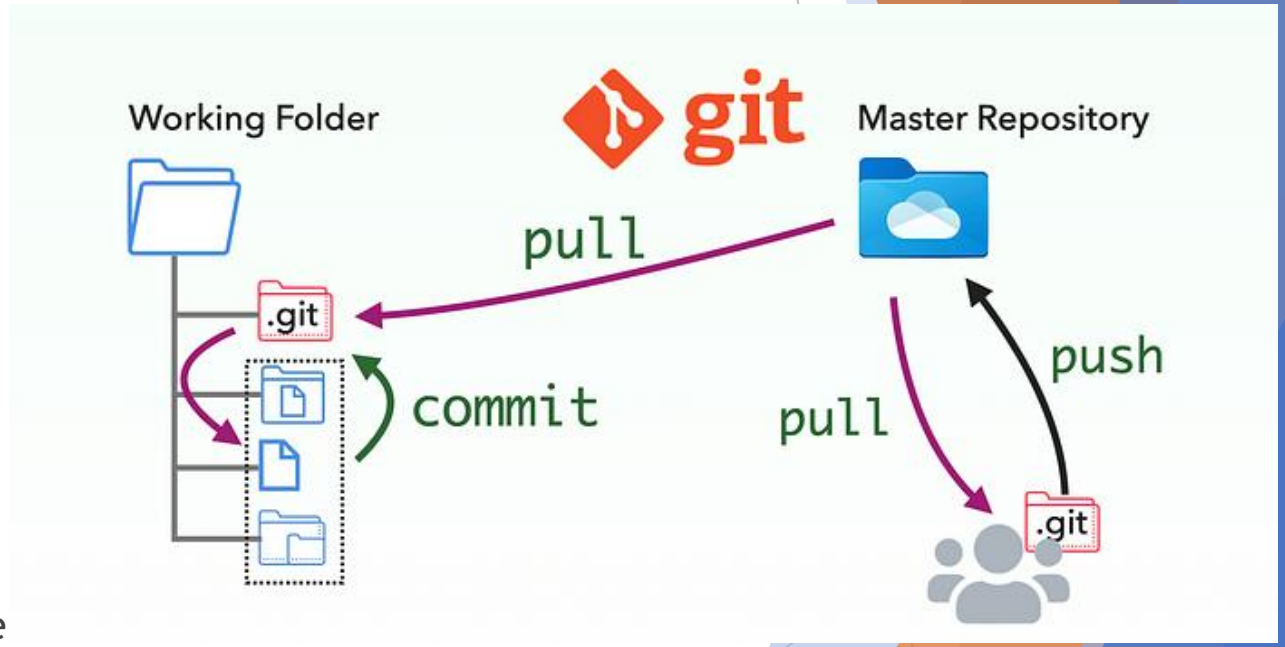
- Klone: Å lage en lokal kopi av et remote repo
  - Lager en mappe med samme innhold som remote repo
  - Merk at den lokale kopien er separat fra remote repo-et og det må synkroniseres med hverandre



https://zhiminzhan.medium.com/10-minutes-guide-to-git-version-control-for-testers-f58e059bb5e7

Git-funksjonalitet - Commit

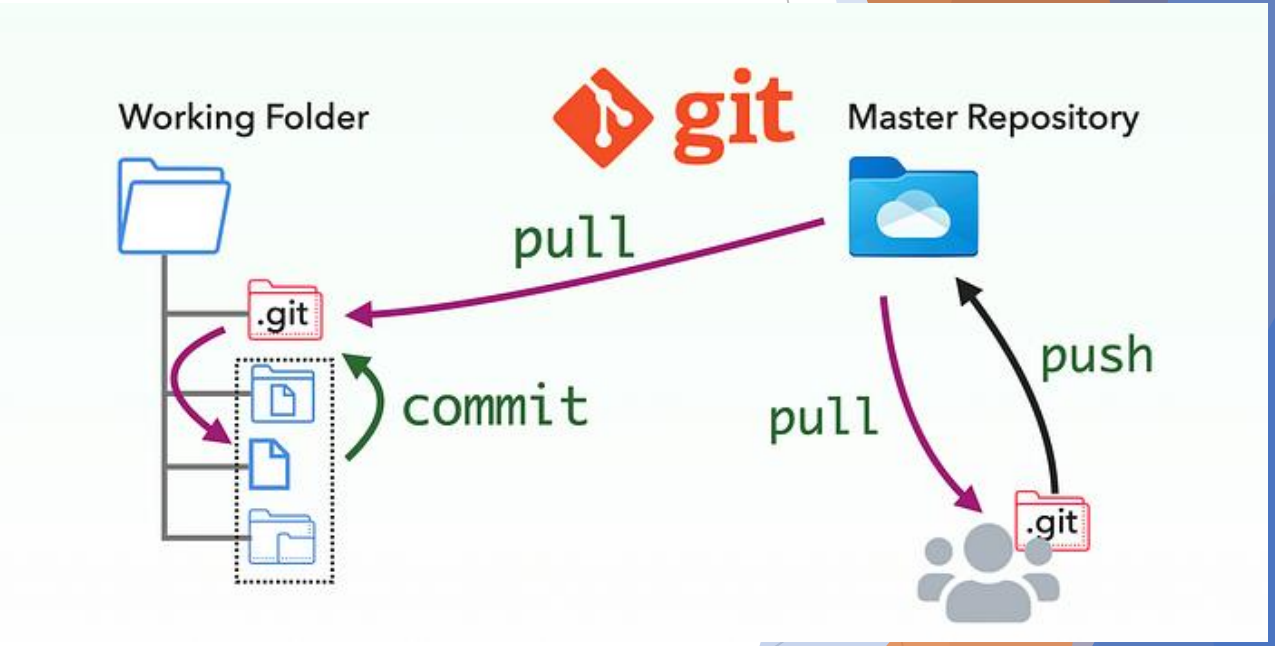
- Når man gjør endringer på de lokale filene blir ikke disse automatisk lagret i versjonskontroll-systemet
- Vi må "committe" endringer for å lage en ny "versjon" av prosjektet som vi kan observere eller gjenopprette senere
  - Commit: Vi deklarerer en endring til versjonskontroll-systemet
- En commit består av to ting
  - Add/stage - Vi må spesifisere hvilke endrede filer vi vil inkludere i commit-en
  - Selve commit-en – Gis et navn og evt. en beskrivelse
- Merk at en commit i et lokalt repo ikke automatisk følger til remote repo-et...



https://zhiminzhan.medium.com/10-minutes-guide-to-git-version-control-for-testers-f58e059bb5e7

Git-funksjonalitet - Push

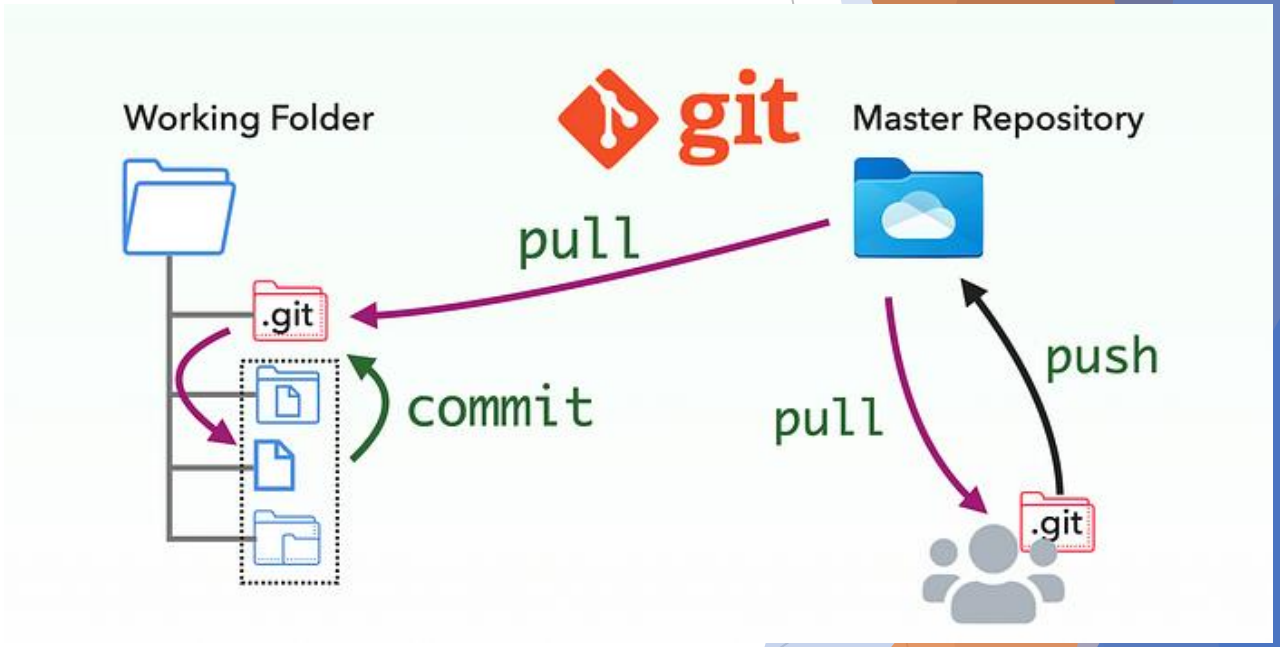
- For å oppdatere remote repo-et med lokale endringer (commits) må vi "pushe" disse
  - Push: Å sende endringene til remote repo
  - Konseptuelt: Vi deler våre endringer med resten av utviklerene



https://zhiminzhan.medium.com/10-minutes-guide-to-git-version-control-for-testers-f58e059bb5e7

Git-funksjonalitet – Fetch og Pull

- Det lokale repoet kan fort ende opp med å bli utdatert
  - Andre utviklere pusher sine commits
  - Du har selv pushet endringer, men fra en annen maskin
- Fetch – Ser etter endringer fra remote repo-et, uten å oppdatere det lokale
  - Mange Git GUI-applikasjoner automatiserer dette
- Pull – Oppdaterer det lokale repo-et med endringer fra remote repo-et
  - Vi henter endringene
  - Kan føre til konflikter



https://zhiminzhan.medium.com/10-minutes-guide-to-git-version-control-for-testers-f58e059bb5e7

Eksempel – Github-repo, Clone, Commit, Push og Pull