

Git og Versjonskontroll: Bygg en Enkel HTML-Side Lokalt

Introduksjon

I denne oppgaven skal du lære å bruke Git for versjonskontroll mens du bygger en enkel HTML-side i Visual Studio Code (VS Code). **Alt arbeid skjer lokalt på din egen maskin** – ingen synkronisering til GitHub eller eksterne servere. Vi fokuserer på å lage commits, opprette alternative versjoner (branches), og navigere mellom dem for å forstå hvordan Git lar deg eksperimentere uten å ødelegge hovedarbeidet.

oppgaven tar ca. 45 minutter og består av 5 utviklingssteg. HTML-koden er svært enkel – poenget er å spore endringer, gå tilbake i tid, og jobbe på alternative versjoner. Du kan forhåndsvisne siden ved å høyreklikke på [index.html](#) og velge "Open with Live Server" (installer extension hvis nødvendig).

Forutsetninger:

- VS Code installert med Git integrert (hvis ikke, installer Git fra git-scm.com og restart VS Code).
- Åpne en ny mappe i VS Code for prosjektet ditt (File > Open Folder).
- Åpne terminalen i VS Code: Trykk Ctrl+` (backtick) eller velg Terminal > New Terminal.

Hva er Git? Git er et verktøy for versjonskontroll. Det lar deg spore endringer i filer, lage "snapshots" (commits) av arbeidet ditt, og gå tilbake i tid. Branches lar deg lage alternative versjoner og bytte mellom dem.

Steg 0: Oppsett (5 min)

1. Opprett en ny fil i VS Code: Høyreklikk i Explorer-panelet > New File > Kall den [index.html](#).
2. Åpne terminalen og initialiser Git-repoet:
 - Kommando: `git init`
 - **Hva gjør den?** Oppretter et nytt Git-repository lokalt i mappen din. Dette setter opp Git til å spore filer her.
3. Sjekk status:
 - Kommando: `git status`
 - **Hva gjør den?** Viser hvilke filer som er endret, staged (klar for commit), eller ikke tracket.

Nå er du klar til å starte!

Steg 1: Grunnleggende HTML-struktur (5 min)

1. Skriv denne enkle HTML-koden i [index.html](#):

```
<!DOCTYPE html>
<html lang="no">
<head>
  <title>Min Første Nettside</title>
</head>
<body>
```

```
</body>
</html>
```

2. Lagre filen (Ctrl+S).

3. I terminalen:

- `git add index.html`
 - **Hva gjør den?** Legger filen til "staging area" – sier til Git at denne endringen skal inkluderes i neste commit.
- `git commit -m "Initial HTML-struktur"`
 - **Hva gjør den?** Lager et lokalt snapshot (commit) av endringene. `-m` legger til en melding som beskriver hva du gjorde.

4. Sjekk historikk:

- `git log --oneline`
 - **Hva gjør den?** Viser en kort liste over commits, med ID og meldinger. Bruk dette for å se progresjonen.

Forhåndsvis siden – den er tom, men gyldig!

Steg 2: Legg til overskrift (5 min)

1. Endre `<body>`-delen til:

```
<body>
  <h1>Velkommen til Min Side!</h1>
</body>
```

2. Lagre.

3. I terminalen:

- `git status` (sjekk endringer).
- `git add index.html`
- `git commit -m "Lagt til overskrift"`

4. `git log --oneline` – se at du har to commits nå.

Forhåndsvis: Du ser en stor overskrift.

Steg 3: Legg til paragraf (5 min)

1. Legg til en paragraf under `<h1>`:

```
<body>
  <h1>Velkommen til Min Side!</h1>
  <p>Dette er en enkel nettside laget med HTML og Git.</p>
</body>
```

2. Lagre.

3. I terminalen:

- `git add index.html`
- `git commit -m "Lagt til paragraf"`

4. `git log --oneline` – tre commits.

Forhåndsvis: Nå med tekst.

Steg 4: Legg til enkel styling (5 min)

1. Legg til en `<style>`-tag i `<head>`:

```
<head>
  <title>Min Første Nettside</title>
  <style>
    body { background-color: lightblue; }
    h1 { color: navy; }
  </style>
</head>
```

2. Lagre.

3. I terminalen:

- `git add index.html`
- `git commit -m "Lagt til enkel CSS-styling"`

4. `git log --oneline` – fire commits.

Forhåndsvis: Bakgrunn og farge endret.

Steg 5: Legg til lenke (5 min)

1. Legg til en lenke under paragrafen:

```
<body>
  <h1>Velkommen til Min Side!</h1>
  <p>Dette er en enkel nettside laget med HTML og Git.</p>
  <a href="https://www.example.com">Besøk Example</a>
</body>
```

2. Lagre.

3. I terminalen:

- `git add index.html`
- `git commit -m "Lagt til lenke"`

4. `git log --oneline` – fem commits. Dette er din hovedlinje (main branch).

Fokus på Alternative Versjoner og Navigering (10 min)

Nå skal vi fokusere på å lage alternative versjoner (branches) og navigere mellom dem. Dette viser hvordan Git lar deg eksperimentere trygt.

1. Sjekk hvilke branches du har:

- `git branch`
 - **Hva gjør den?** Lister alle lokale branches. Du ser `* main` (du er på main).

2. Finn commit-ID fra Steg 3 (bruk `git log --oneline` – f.eks. "abc123 Lagt til paragraf").

3. Opprett en ny branch fra den commiten:

- `git checkout -b alternativ-versjon <commit-ID>` (erstatt med den korte ID-en, f.eks. abc123).
 - **Hva gjør den?** `-b` oppretter en ny branch kalt "alternativ-versjon". `checkout` bytter til den commiten. Filene dine endres automatisk til versjonen fra Steg 3.

4. Endre `<p>`-teksten i `index.html` til: `<p>Dette er en ALTERNATIV versjon uten styling eller lenke.</p>`.

5. Lagre.

6. I terminalen:

- `git add index.html`
- `git commit -m "Endret tekst i alternativ branch"`

7. `git log --oneline` – se historikken i denne branchen (den deler historie med main opp til Steg 3).

Navigering mellom versjoner:

- Bytt tilbake til main:
 - `git checkout main`
 - **Hva gjør den?** Bytter til "main" branch. Filene går tilbake til Steg 5-versjonen (med styling og lenke).
- Bytt til alternativ-versjon igjen:
 - `git checkout alternativ-versjon`
 - Filene endres til din alternative versjon.
- Se forskjellene mellom branches:
 - `git diff main alternativ-versjon`
 - **Hva gjør den?** Viser linje-for-linje-forskjeller mellom de to branches.
- Forhåndsvis i begge: Bytt branch og oppdater Live Server for å se endringene.

Eksperimenter: Opprett en ny branch fra main (`git checkout -b ny-feature`) og legg til noe nytt, f.eks. en ekstra paragraf. Commit, og bytt frem og tilbake.

Hvis du vil kombinere branches (valgfritt, hvis tid):

- Fra main: `git merge alternativ-versjon`
 - **Hva gjør den?** Slår sammen endringer fra alternativ-versjon inn i main. Løs konflikter hvis nødvendig (rediger filen og commit).

Avslutning og Tips (5 min)

- Du har nå bygget en side i 5 steg og lært å navigere alternative versjoner lokalt!

- Vanlige kommandoer oppsummert:
 - `git init`: Start repo.
 - `git add <fil>`: Stage endringer.
 - `git commit -m "melding"`: Lag snapshot.
 - `git status`: Sjekk status.
 - `git log --oneline`: Se historikk.
 - `git branch`: Se branches.
 - `git checkout <branch/commit-ID>`: Bytt versjon/branch.
 - `git diff <branch1> <branch2>`: Se forskjeller.
- Feilsøking: Hvis du er "detached" (ikke på branch), bruk `git checkout main`.
- Neste steg: Senere kan du legge til remote (f.eks. GitHub) med `git remote add` og `git push`.

Prøv å lage flere branches og naviger!