

Introduktion til GitHub som projektstyringsværktøj

Hvad er GitHub?

GitHub er en platform, hvor udviklere kan gemme deres kode, samarbejde om projekter og følge ændringer i deres kode over tid. Den er bygget oven på Git, som er et versionsstyringssystem. GitHub tilbyder mange funktioner, der gør det til et kraftfuldt værktøj til projektstyring.

Nøglefunktioner i GitHub:

1. Repositories:

- Et repository (repo) er en mappe, hvor alle filer til dit projekt opbevares. Det inkluderer koden, dokumentation og endda issue tracking.
- Du kan oprette private eller offentlige repos afhængigt af, om du vil dele dit arbejde med hele verden eller holde det begrænset til dit team.

2. Versionsstyring:

- GitHub bruger Git til at spore ændringer i dine filer. Dette betyder, at du kan se, hvem der har ændret hvad og hvornår, og du kan vende tilbage til tidligere versioner af din kode, hvis noget går galt.

3. Branches:

- Branches giver dig mulighed for at arbejde på forskellige funktioner eller fejlrettelser separat fra hovedkoden (main eller master branch). Når dit arbejde på en branch er færdigt, kan det flettes tilbage i hovedkoden.
- Dette muliggør parallel udvikling og mindsker risikoen for at introducere fejl i den stabile version af projektet.

4. Pull Requests:

- Pull requests (PR) bruges til at diskutere og gennemgå ændringer, før de integreres i hovedkoden. Når du opretter en PR, kan teammedlemmer kommentere på ændringerne, foreslå forbedringer og godkende arbejdet.
- PR'er hjælper med at sikre kodekvalitet og fremmer samarbejde.

5. Issues og Projekter:

- GitHub issues er en måde at spore fejl, forbedringer og andre opgaver relateret til dit projekt. Issues kan tildeles teammedlemmer, mærkes med tags og knyttes til milepæle.
- GitHub Projects tilbyder en tavle-lignende visning, hvor du kan organisere issues og PR'er i kolonner, såsom "To Do", "In Progress", og "Done". Dette giver et visuelt overblik over projektets status og hjælper med at styre workflow.

6. Wikis og Dokumentation:

- Hvert repo kan have en wiki, som er ideel til at oprette dokumentation, tutorials og andre ressourcer, som dit team har brug for. God dokumentation er nøglen til at sikre, at alle forstår, hvordan man bruger og bidrager til projektet.

7. CI/CD Integration:

- GitHub Actions giver dig mulighed for at opsætte Continuous Integration og Continuous Deployment (CI/CD) workflows direkte i dit repo. Dette automatiserer test, bygning og implementering af din kode, hvilket sparer tid og reducerer fejl.

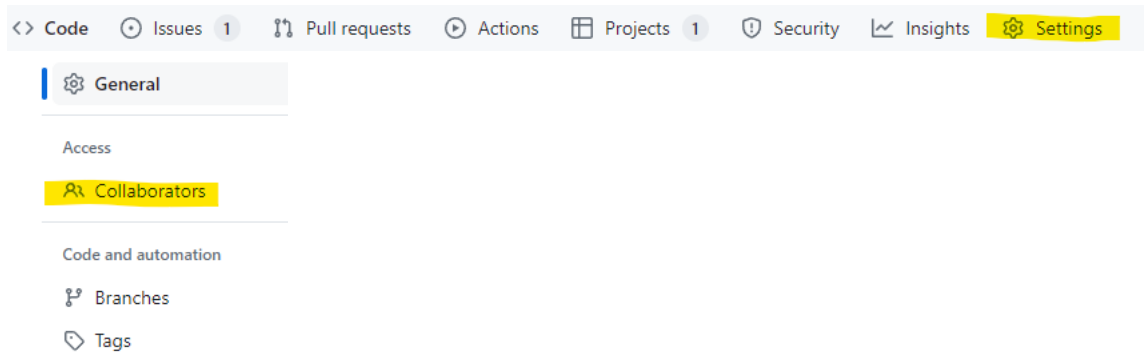
Fordele ved at bruge GitHub til projektstyring:

- **Samarbejde:** GitHub gør det nemt for teams at arbejde sammen på samme kodebase, uanset hvor de befinder sig.
- **Gennemsigtighed:** Alle ændringer er sporet, og det er let at se historikken for hver fil. Dette fremmer ansvarlighed og gennemsigtighed i teamet.
- **Effektivitet:** Med funktioner som branches, PR'er og CI/CD kan du arbejde mere effektivt og levere høj kvalitet kode hurtigere.
- **Centralisering:** Alt relateret til dit projekt - fra kode til dokumentation til opgavestyring - findes ét sted.

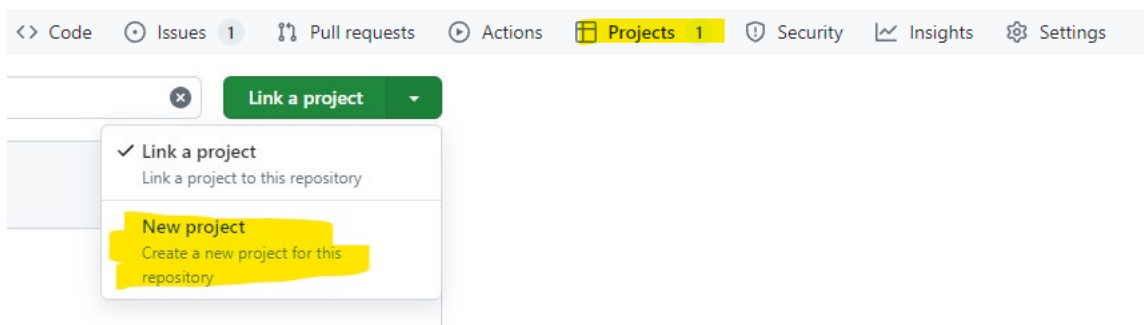
Opsummering: GitHub er mere end bare et sted at gemme din kode. Det er et omfattende værktøj til projektstyring, der understøtter samarbejde, versionsstyring, kodegennemgang, og opgavestyring. Ved at udnytte GitHub's funktioner kan du og dit team arbejde mere effektivt og producere bedre software.

Opstart:

- Opret et nyt vite-projekt i VS code til din opgave.
- Klik på Source i værktøjspanelet til venstre og initialiser et nyt, privat Github repository
- Åbn Github repositoret i browseren
- Klik på Settings og derefter collaborators:



- Tilføj eventuelle gruppemedlemmer + Underviser Anne 😊 (AnneLundM)
- Klik på Projects og derefter Link a projekt-knappen



- Opret et nyt projekt for jeres repository
- Vælg Kanban template og navngiv projektet.

Planlæg før kodning

- Gå til jeres repository på Github og klik på Issues

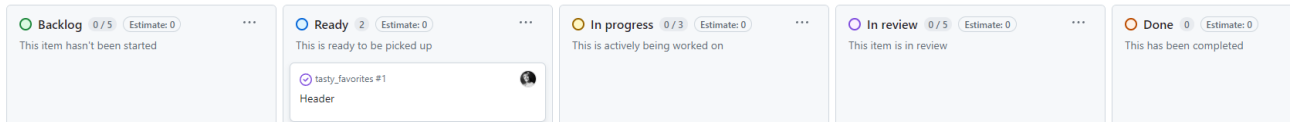


- Læs denne guide til oprettelse af issues: <https://docs.github.com/en/issues/tracking-your-work-with-issues/quickstart>
- Find ud af hvilke issues der skal oprettes, for at kode dit projekt.
- Opret disse issues med udgangspunkt i guiden
- Tilføj evt assignees, projekt og evt. labels til hvert issue

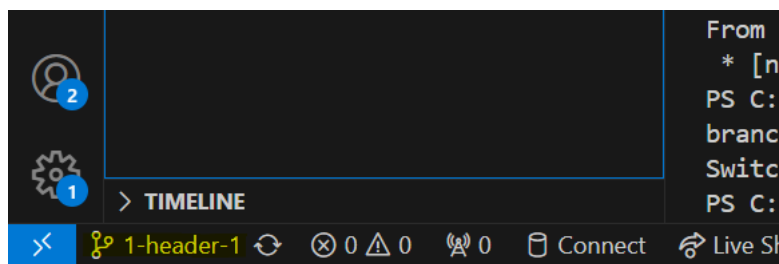
- Gå til Projects og flyt de nye issues fra 'No Status' eller 'Backlog' til 'Ready'. Her skal alle opgaver der er definerede og uddelegerede ligge. (Issues der ikke er uddelegerede skal ligge i 'Backlog'.)

Inden du går i gang med at 'løse dit issue':

- Flyt dit issue/item fra 'Ready' til 'In progress' i dit på Github.

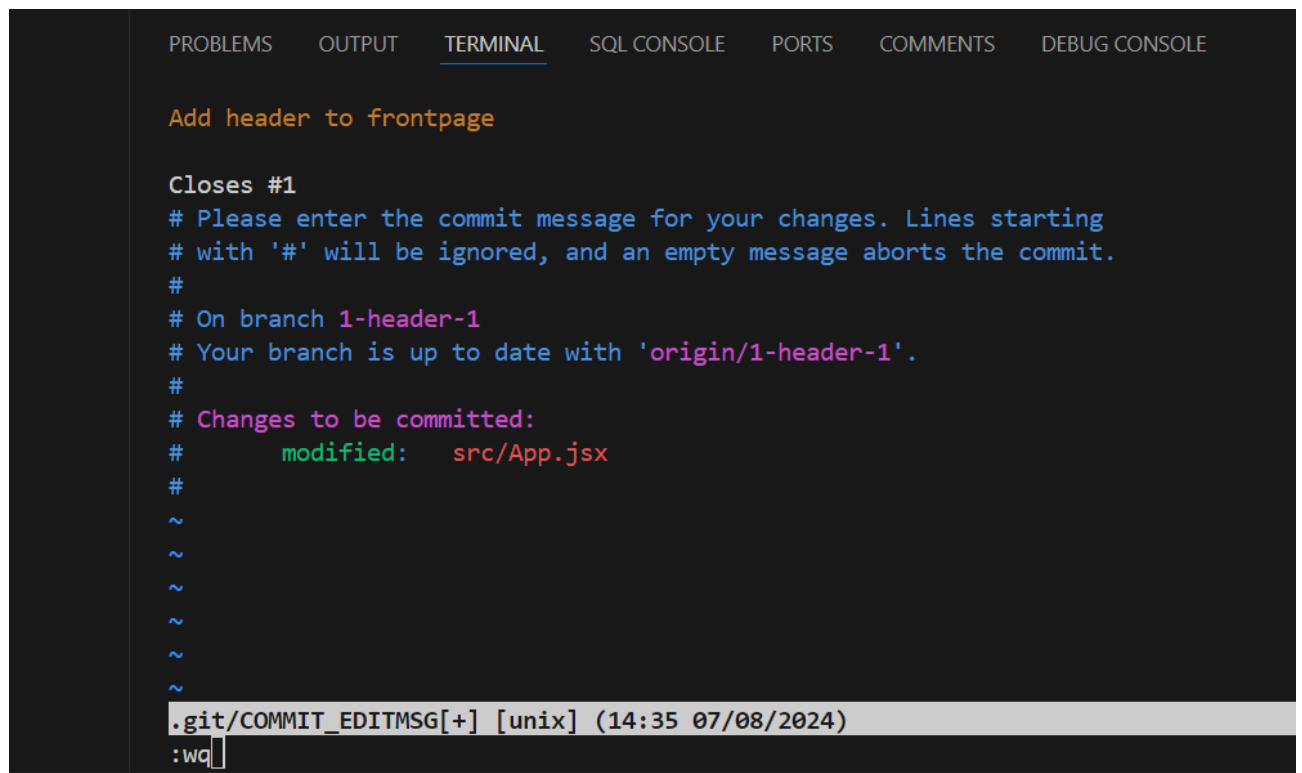


- Opret en branch fra dit issue vha denne guide: <https://docs.github.com/en/issues/tracking-your-work-with-issues/creating-a-branch-for-an-issue>
- Når du står på den nye branch (Kan ses nede i venstre hjørne i VS Code) kan du gå i gang med at kode.



- Når de relevante ændringer er lavet klikkes på Source Control-knappen igen. Her er et overblik over de filer, der er foretaget ændringer i. Gennemgå alle filerne for at være sikker på, at det er de rigtige ændringer der 'pushes' til main (Hovedkoden).
- Åbn terminalen og skriv: **git add** . (Denne kommando 'stager' dine ændringer så de er klar til at blive flettet ind i hovedkoden)
- Igen i terminalen: Skriv **git commit**
- Tryk 'a' på tastaturet for at aktivere redigering.
- Skriv en commit-besked når du har læst følgende vejledning: <https://medium.com/@saeid/10-essential-practices-for-better-git-commits-and-why-they-matter-3cfc420bf53e>

FX:



The screenshot shows a VS Code terminal window with the 'TERMINAL' tab selected. The terminal output is as follows:

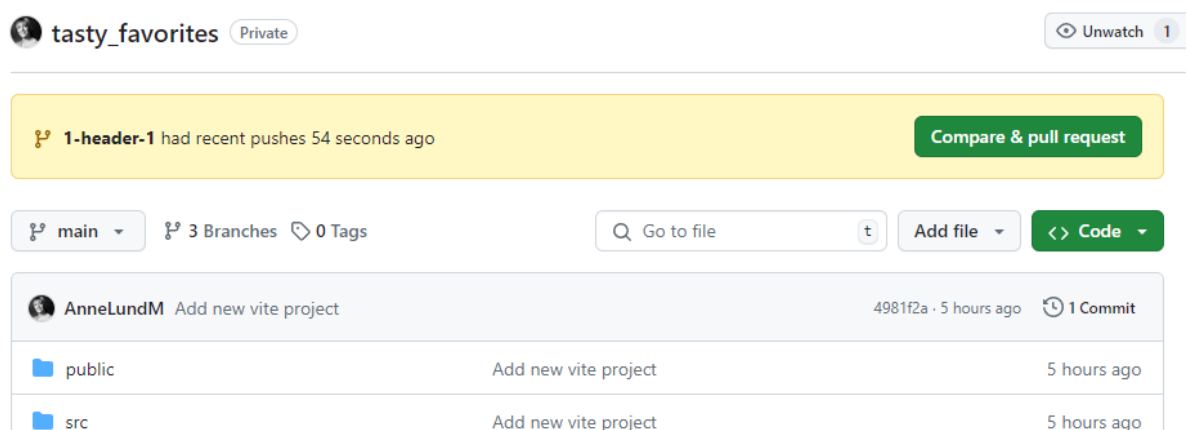
```

Add header to frontpage

Closes #1
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch 1-header-1
# Your branch is up to date with 'origin/1-header-1'.
#
# Changes to be committed:
#   modified:   src/App.jsx
#
~
~
~
~
~
~
~
.git/COMMIT_EDITMSG[+] [unix] (14:35 07/08/2024)
:wq

```

- Afslut ved at trykke Esc efterfulgt af :wq (Står for write and quit)
- Skriv nu i terminalen: git push origin (Nu sendes dit commit til Github hvor du skal lave en pullrequest)
- Åbn repositoriet i browseren. Den skulle gerne se nogenlunde sådan her ud:



- Klik på Compare & pull request

- Dobbelttjek dine ændringer nederst på siden. De røde kodelinjer er dem der bliver fjernet fra hovedkoden når du 'merger' din kode. De grønne er dem der bliver tilføjet.
- Vil du have nogen til at 'reviewe' din kode, kan du tilføje dem under Reviewers. Ellers skal der ikke gøres andet, end at trykkes på 'Create pull request':

Open a pull request

Create a new pull request by comparing changes across two branches. If you need to, you can also [compare across forks](#). [Learn more about diff comparisons here](#).

The screenshot shows the GitHub 'Open a pull request' interface. At the top, it indicates the base branch is 'main' and the compare branch is '1-header-1'. A green checkmark states 'Able to merge. These branches can be automatically merged.' Below this, there's a section to 'Add a title' with the text 'Add header to frontpage'. The 'Add a description' section has a text area with 'Closes #1'. To the right, there are settings for Reviewers (No reviews), Assignees (No one—assign yourself), Labels (None yet), Projects (None yet), and Milestone (No milestone). At the bottom right, there's a 'Development' section with a note about using closing keywords. A green 'Create pull request' button is at the bottom center. A footer note mentions the GitHub Community Guidelines.

- Github tjekker om der evt i mellemtiden er merget noget kode, der konflikter med din. Hvis alt er godt, ser det således ud:

The screenshot shows the GitHub pull request summary. It features three status messages: 1) 'Require approval from specific reviewers before merging' with a link to 'Rulesets' and an 'Add rule' button. 2) 'Continuous integration has not been set up' with a link to 'GitHub Actions' and a note about catching bugs and enforcing style. 3) 'This branch has no conflicts with the base branch' with a green checkmark and a note that merging can be performed automatically. At the bottom, there's a green 'Merge pull request' button and a link to 'open this in GitHub Desktop' or view 'command line instructions'.

- Nu kan der trykkes på den store grønne knap: Merge pull request og confirm pull request og der kommer en boks der sådan her ud:

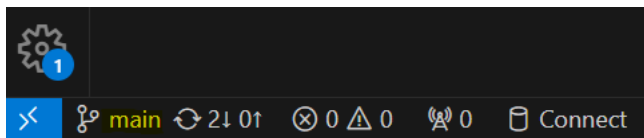


Pull request successfully merged and closed

You're all set—the `1-header-1` branch can be safely deleted.

Delete branch

- Tryk på 'Delete branch'
- Tilbage i VS code's terminal skal der nu 'checkes ud' ud til main således: ***git checkout main***
- Dobbelttjek at det er main-branch du står på:



- 2-tallet der vises her, efterfulgt af en pil ned, betyder at der er ændringer på main som ikke er hentet ned endnu. Det er *vigtigt* at sørge for at koden er up-to-date inden man laver ændringer igen. Derfor skrives nu i terminalen:

git fetch origin

git pull origin

Nu er koden opdateret og der kan igen oprettes en ny branch fra dit næste issue.