Samenwerken met GitHub

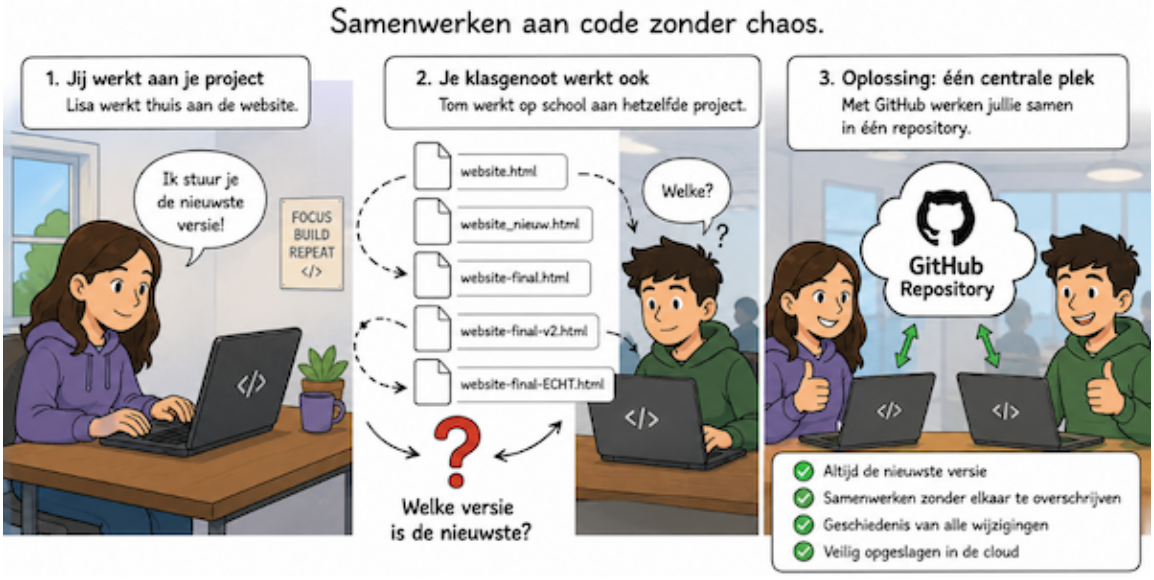
Status niet, af maar een eerste opzetje gegenreerd met NotebookLM/Codex

1. Waarom GitHub? Maak je eerste repository

? Leerdoelen

- Je begrijpt waarom softwareontwikkelaars GitHub gebruiken.
- Je kunt een GitHub-account aanmaken.
- Je kunt een nieuwe repository aanmaken.
- Je begrijpt wat een repository is.

? Situatie uit de praktijk



Stel je voor dat jij en een klasgenoot samen een website maken.

- □□ Lisa werkt thuis.
- □□ Tom werkt op school.

Beiden hebben een laptop.

Ze plaatsen beiden hun bestanden op een gemeenschappelijke drive van school (bijvooebeeld OneDrive).

```
website.html
website_nieuw.html
website_final.html
website_final_v2.html
website_final_echt.html
```

Na een paar dagen weet niemand meer welke versie de nieuwste is. Misschien heeft Lisa een fout opgelost, terwijl Tom ondertussen nieuwe functies hebt toegevoegd. Welke versie moet je nu gebruiken?

Softwareontwikkelaars hebben hiervoor een oplossing bedacht: **Git** en **GitHub**.

? Uitleg

Een **repository** of **repo** is de centrale plek waar alle bestanden van een project worden opgeslagen. Zie het als een online kluis waarin alle teamleden samenwerken.

Elke wijziging wordt opgeslagen. Daardoor kun je altijd zien:

- wie een wijziging heeft gemaakt;
- wanneer deze is gemaakt;
- wat er precies is veranderd;
- en je kunt oude versies terughalen als er iets misgaat.

Vrijwel ieder professioneel softwarebedrijf gebruikt Git en GitHub of een vergelijkbaar systeem.

?? Opdracht – Maak je eerste repository

1. Ga naar **github.com**.
2. Maak een GitHub-account aan als je er nog geen hebt.
3. Klik op **New Repository**.
4. Geef de repository de naam `github-challenge`.
5. Kies voor een **Public Repository**.
6. Vink **Add a README.md** aan.

7. Klik op **Create Repository**.

? Denkvraag

Waarom is een online repository handiger dan bestanden steeds naar elkaar mailen of via Teams versturen?

? Inleveren

1. Een screenshot van je repository op GitHub.
2. De URL van je repository.
3. Het antwoord op de denkvraag in 2 tot 5 zinnen.

2. De repository naar je eigen computer halen (Clone)

? Leerdoelen

- Je begrijpt wat clonen betekent.
- Je kunt een GitHub-repository naar je eigen computer kopiëren.
- Je kunt de repository openen in VS Code.
- Je begrijpt het verschil tussen de online versie en de lokale versie.

? Situatie uit de praktijk

Je repository staat nu online op GitHub. Dat is handig om je project veilig te bewaren en te delen.

Maar je gaat natuurlijk niet de hele tijd op de website van GitHub code typen. Als softwaredeveloper werk je meestal op je eigen laptop in een editor zoals **VS Code**.

Daarom moet je de repository eerst naar je eigen computer halen. Dat noemen we **clonen**.

? Uitleg

Clonen betekent dat je een kopie van de online repository op je eigen computer zet.

Na het clonen heb je twee versies van hetzelfde project:

- de **online versie** op GitHub;
- de **lokale versie** op je eigen computer.

Je werkt meestal in de lokale versie. Later stuur je je wijzigingen weer terug naar GitHub. Dat leer je in de volgende opdracht.

```
GitHub repository
        ↓ clone
Jouw computer / VS Code
```

?? Opdracht – Clone je repository

1. Open je repository `github-challenge` op GitHub.
2. Klik op de groene knop **Code**.
3. Kopieer de HTTPS-link van je repository.
4. Open **VS Code**.
5. Open de terminal in VS Code.
6. Ga in de terminal naar je `htdocs` map.
7. Typ het volgende commando:

```
git clone PLAK_HIER_DE_URL_VAN_JE_REPOSITORY
```

8. Druk op **Enter**.
9. Open daarna de map `github-challenge` in VS Code.
10. Controleer of je het bestand `README.md` ziet.

? Controle

Als het goed is, staat je repository nu op je eigen computer.

Je kunt dit controleren door te kijken of je deze map hebt:

```
htdocs/github-challenge
```

In deze map moet minimaal het bestand `README.md` staan.

? Denkvraag

Wanneer zou je een clone willen maken van een Github Repository, noem tenminste twee situtaties.

? Inleveren

1. Een screenshot van VS Code waarin de map `github-challenge` zichtbaar is.
2. Een screenshot waarin het bestand `README.md` zichtbaar is.
3. Het antwoord op de denkvraag.

3. Je eerste wijziging opslaan (Commit & Push)

? Leerdoelen

- Je begrijpt wat een commit is.
- Je kunt wijzigingen opslaan met Git.
- Je kunt een commit naar GitHub sturen met `git push`.
- Je begrijpt waarom een goede commit message belangrijk is.

? Situatie uit de praktijk

Je hebt de repository naar je eigen computer gekopieerd. Nu kun je bestanden aanpassen.

Maar Git houdt wijzigingen niet automatisch bij. Jij bepaalt zelf wanneer je een belangrijke stap wilt bewaren. Zo kun je altijd terugkijken welke wijzigingen je hebt gemaakt.

Dit werkt een beetje zoals een savepoint in een computerspel. Gaat er later iets mis? Dan kun je altijd terug naar een eerdere versie.

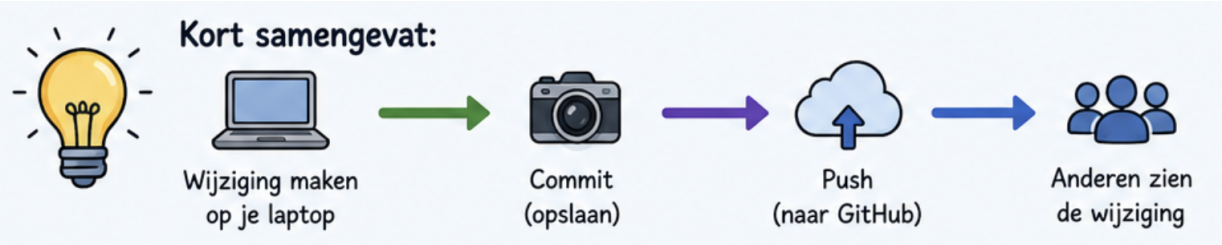
? Uitleg

Een **commit** is een momentopname van je project. Git slaat daarbij op:

- welke bestanden zijn gewijzigd;
- wie de wijziging heeft gemaakt;
- wanneer de wijziging is gemaakt;
- waarom de wijziging is gemaakt (de commit message).

Na een commit staat de wijziging alleen nog op jouw computer. Met **git push** stuur je de commit naar GitHub zodat anderen jouw werk ook kunnen zien.

```
Jouw computer
      │
      │ git commit
      ▼
Nieuwe versie opgeslagen
      │
      │ git push
      ▼
GitHub Repository
```



?? Opdracht – Maak je eerste commit

1. Open de map `github-challenge` in VS Code.
2. Maak een nieuw bestand met de naam `index.html`.
3. Plaats onderstaande code in het bestand:

```
<!DOCTYPE html>
<html>
```

```
<head>
    <title>Mijn eerste GitHub project</title>
</head>
<body>
    <h1>Hallo GitHub!</h1>
</body>
</html>
```

4. Sla het bestand op.
5. Open de terminal.
6. Voer achter elkaar de volgende commando's uit:

```
git add .
git commit -m "Eerste webpagina toegevoegd"
git push
```

7. Open daarna je repository op GitHub.
8. Controleer of `index.html` online zichtbaar is.

? Goede commit messages

Een commit message moet duidelijk beschrijven wat je hebt veranderd.

Minder goed	Goed
Update	Eerste webpagina toegevoegd
Dingen gedaan	Navigatiemenu toegevoegd
Fix	Fout in contactformulier opgelost

? Denkvraag

Waarom is een duidelijke commit message belangrijk wanneer je samenwerkt met andere programmeurs?

? Inleveren

1. Een screenshot van GitHub waarop `index.html` zichtbaar is.
2. Een screenshot van de terminal waarin de commit en push succesvol zijn uitgevoerd.
3. Het antwoord op de denkvraag in 2 tot 5 zinnen.

4. Altijd de nieuwste versie ophalen (Pull)

? Leerdoelen

- Je begrijpt wat `git pull` doet.
- Je kunt wijzigingen van GitHub naar je eigen computer ophalen.
- Je begrijpt waarom je bijna altijd eerst een `git pull` uitvoert voordat je gaat programmeren.

? Situatie uit de praktijk

Stel dat je gisteren aan je project hebt gewerkt. Vandaag open je opnieuw je laptop.

Ondertussen heeft een klasgenoot ook aan hetzelfde project gewerkt en zijn wijzigingen naar GitHub gestuurd.

Jouw computer weet daar nog niets van. Als je nu meteen gaat programmeren, werk je met een oude versie van het project. Dat kan later voor problemen zorgen.

Daarom beginnen softwareontwikkelaars vaak met één simpel commando:

```
git pull
```

? Uitleg

Met **git pull** haal je de nieuwste wijzigingen van GitHub op naar je eigen computer.

Je lokale repository wordt daardoor weer gelijk aan de online repository.

```
GitHub Repository
        |
```

```
        | git pull
        ▼
Jouw computer
```

Onthoud deze gouden regel:

> **Begin iedere programmeersessie met `git pull`.**

?? Opdracht – Haal de nieuwste versie op

1. Open je repository op GitHub in de browser.
2. Open het bestand `README.md`.
3. Klik op het potloodje (**Edit**).
4. Voeg onder de bestaande tekst de volgende regel toe:

```
Dit project is gemaakt tijdens de GitHub Challenge.
```

5. Sla de wijziging op met **Commit changes**.
6. Open nu VS Code.
7. Open de terminal.
8. Voer het volgende commando uit:

```
git pull
```

9. Open `README.md` in VS Code.
10. Controleer of de nieuwe regel nu ook op jouw computer zichtbaar is.

? Controle

Is de nieuwe tekst zichtbaar in `README.md`? Dan is de pull succesvol uitgevoerd.

In de terminal zie je meestal een melding waarin staat welke bestanden zijn bijgewerkt.

? Denkvraag

Waarom is het verstandig om altijd eerst `git pull` uit te voeren voordat je zelf wijzigingen gaat maken?

? Extra uitdaging

Vraag een klasgenoot om ook een regel aan `README.md` toe te voegen.

Voer daarna opnieuw `git pull` uit.

Welke wijziging is erbij gekomen?

? Inleveren

1. Een screenshot van de terminal waarin `git pull` succesvol is uitgevoerd.
2. Een screenshot van `README.md` in VS Code waarin de nieuwe tekst zichtbaar is.
3. Het antwoord op de denkvraag in 2 tot 5 zinnen.

5. Samenwerken aan hetzelfde project (Collaborators)

? Leerdoelen

- Je kunt een klasgenoot uitnodigen als collaborator.
- Je begrijpt wat een collaborator is.
- Je begrijpt waarom softwareontwikkelaars samenwerken via één repository.

? Situatie uit de praktijk

Tot nu toe heb je alleen gewerkt. In de praktijk bouw je software bijna nooit alleen.

Bij een softwarebedrijf werken vaak meerdere programmeurs tegelijkertijd aan hetzelfde project. De één maakt de website, een ander werkt aan de database en weer iemand anders lost fouten op.

Iedereen werkt in dezelfde repository. Daardoor is altijd duidelijk wat de nieuwste versie van het project is.

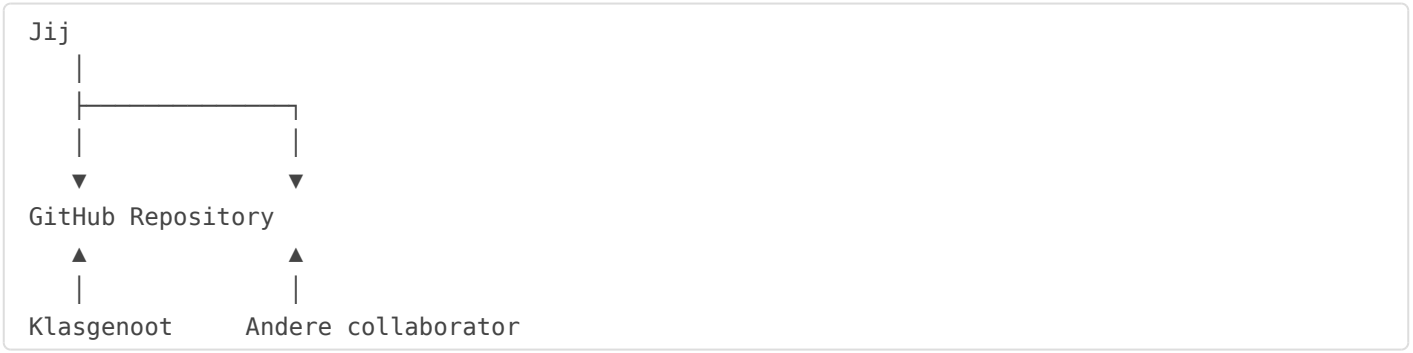
? Uitleg

Een **Collaborator** is iemand die toestemming heeft gekregen om wijzigingen aan jouw repository te maken.

Collaborators kunnen:

- bestanden aanpassen;
- nieuwe bestanden toevoegen;
- commits maken;
- wijzigingen naar GitHub pushen.

Je bepaalt zelf wie toegang krijgt tot jouw repository.



?? Opdracht – Nodig een collaborator uit

1. Open je repository op GitHub.
2. Ga naar **Settings**.
3. Klik op **Collaborators**.
4. Klik op **Add people**.
5. Zoek de GitHub-gebruikersnaam van je klasgenoot.
6. Verstuur de uitnodiging.
7. Je klasgenoot accepteert de uitnodiging.
8. Controleer of je klasgenoot nu als collaborator zichtbaar is.

? Controle

Vraag je klasgenoot om de repository te openen.

Kan hij of zij de repository openen en wijzigingen maken? Dan is de uitnodiging succesvol geaccepteerd.

? Denkvraag

Waarom is het veiliger om alleen vertrouwde personen als collaborator toe te voegen?

? Extra uitdaging

Bekijk de lijst met collaborators.

Kun je ontdekken wie de eigenaar (Owner) van de repository is?

? Inleveren

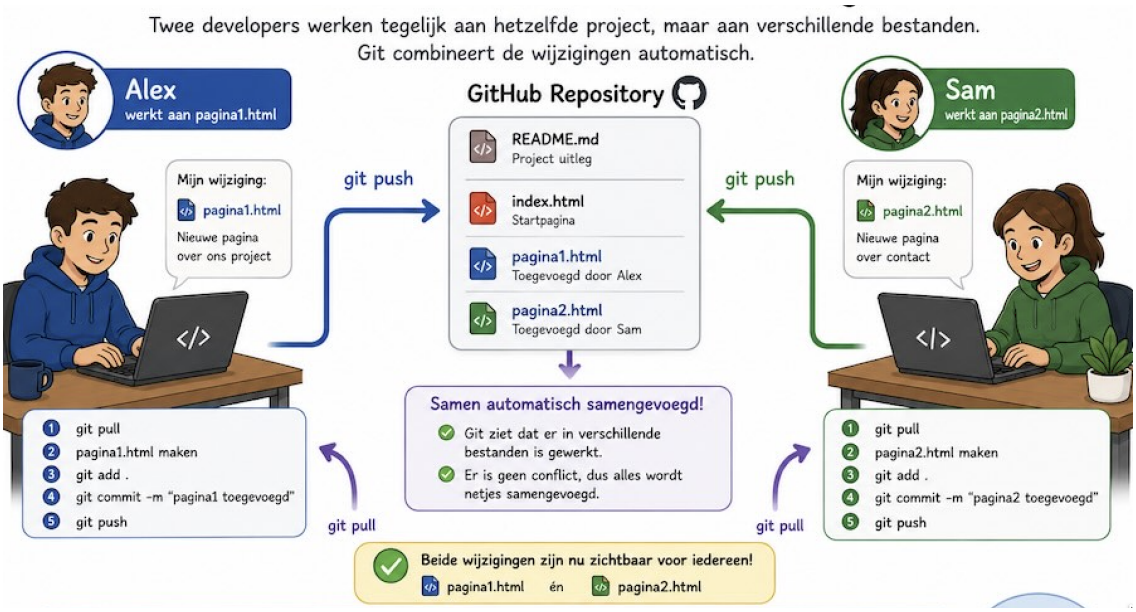
1. Een screenshot van de pagina **Collaborators** waarop je klasgenoot zichtbaar is.
2. Een screenshot van je klasgenoot waarop dezelfde repository geopend is.
3. Het antwoord op de denkvraag in 2 tot 5 zinnen.

6. Tegelijkertijd aan hetzelfde project werken

? Leerdoelen

- Je ervaart hoe meerdere programmeurs tegelijkertijd aan één project kunnen werken.
- Je leert hoe Git wijzigingen van verschillende personen combineert.
- Je oefent met de complete Git-werkwijze: pull, wijzigen, commit en push.

? Situatie uit de praktijk



Je werkt nu samen met een klasgenoot aan dezelfde repository.

Bij een softwarebedrijf werkt bijna nooit één programmeur aan een project. Vaak zijn tientallen ontwikkelaars tegelijkertijd bezig.

Gelukkig hoeven zij niet steeds op elkaar te wachten. Als iedereen aan een ander onderdeel werkt, kan Git de wijzigingen automatisch samenvoegen.

In deze opdracht gaan jullie dat zelf ervaren.

? Uitleg

Wanneer twee programmeurs verschillende bestanden aanpassen, ontstaat er meestal geen probleem.

```
Alex
      |
      ├── pagina1.html
      |
      ▼

GitHub Repository

      ▲
      |
      ├── pagina2.html
      |
Sam
```

Git ziet dat beide programmeurs aan verschillende bestanden hebben gewerkt en voegt de wijzigingen automatisch samen.

?? Opdracht – Werk tegelijkertijd

1. Werk samen met de klasgenoot die collaborator is van jouw repository.
2. Voer allebei eerst `git pull` uit.
3. Jij maakt een bestand `pagina1.html`.
4. Je klasgenoot maakt een bestand `pagina2.html`.
5. Zet in beide bestanden een eenvoudige HTML-pagina met een titel en een koptekst.
6. Voer daarna uit:

```
git add .
git commit -m "Nieuwe pagina toegevoegd"
git push
```

7. Laat daarna ook je klasgenoot zijn of haar wijzigingen committen en pushen.
8. Voer vervolgens opnieuw `git pull` uit.
9. Controleer of je nu zowel `pagina1.html` als `pagina2.html` in je project hebt.

? Controle

Open de projectmap in VS Code.

Als het goed is, zie je nu:

```
README.md
index.html
pagina1.html
pagina2.html
```

Open beide HTML-bestanden in je browser. Werken ze allebei? Dan zijn de wijzigingen succesvol samengevoegd.

? Denkvraag

Waarom ontstond er geen conflict terwijl jullie allebei tegelijkertijd aan hetzelfde project werkten?

? Extra uitdaging

Maak nu allebei nóg een nieuw HTML-bestand, bijvoorbeeld `contact.html` en `over-ons.html`.

Kunnen deze opnieuw zonder problemen worden samengevoegd?

? Inleveren

1. Een screenshot van VS Code waarop zowel `pagina1.html` als `pagina2.html` zichtbaar zijn.
2. Een screenshot van GitHub waarop beide bestanden zichtbaar zijn.
3. Het antwoord op de denkvraag in 2 tot 5 zinnen.

7. Help! Een Merge Conflict ontstaat (Deel 1)

? Leerdoelen

- Je ervaart hoe een merge conflict ontstaat.
- Je begrijpt waarom Git soms een push weigert.
- Je leert dat Git jouw werk beschermt tegen overschrijven.

? Situatie uit de praktijk

Stel dat jij en je klasgenoot tegelijkertijd dezelfde pagina aanpassen.

Jullie veranderen allebei precies dezelfde regel. Daarna probeer je allebei de wijzigingen naar GitHub te sturen.

Wie heeft er dan gelijk?

Git weet dat niet. Daarom weigert Git de tweede push. Zo voorkomt Git dat het werk van iemand anders per ongeluk wordt overschreven.

? Uitleg

Een **Merge Conflict** ontstaat wanneer twee programmeurs dezelfde regel in hetzelfde bestand wijzigen.

Git kan dan niet bepalen welke versie de juiste is.

```
Alex verandert regel 5
        │
        ▼

index.html

        ▲
        │
Sam verandert ook regel 5

 Welke versie moet Git bewaren?
```

In plaats van zomaar één versie te kiezen, stopt Git en vraagt het jou om de juiste keuze te maken.

?? Opdracht – Forceer een merge conflict

1. Voer allebei eerst `git pull` uit.
2. Open `index.html`.
3. Verander allebei dezelfde regel, bijvoorbeeld de tekst in de `<h1>`.
4. Sla het bestand op.
5. Jullie voeren allebei uit:

```
git add .
git commit -m "Titel aangepast"
```

6. Laat één persoon eerst `git push` uitvoeren.
7. De andere persoon probeert daarna ook te pushen.
8. Lees de foutmelding die Git geeft.

? Controle

De tweede programmeur krijgt een melding dat de push is geweigerd.

Git doet dit expres om te voorkomen dat de wijzigingen van de eerste programmeur verloren gaan.

? Denkvraag

Waarom weigert Git de tweede push in plaats van deze automatisch te accepteren?

□□ Wist je dat?

Een merge conflict is geen fout in Git.

Het is juist een veiligheidsmaatregel die voorkomt dat programmeurs elkaars werk kwijtraken.

? Inleveren

1. Een screenshot van de foutmelding in de terminal.
2. Een screenshot waarop zichtbaar is dat de push is geweigerd.
3. Het antwoord op de denkvraag in 2 tot 5 zinnen.

8. Los het merge conflict op (Deel 2)

? Leerdoelen

- Je kunt een merge conflict herkennen.
- Je begrijpt welke wijzigingen van jou zijn en welke van je klasgenoot.
- Je kunt een merge conflict oplossen in VS Code.
- Je kunt de oplossing committen en naar GitHub pushen.

? Situatie uit de praktijk

In de vorige opdracht weigerde Git jouw push. Dat deed Git niet omdat er iets fout was, maar omdat het niet wist welke versie van `index.html` bewaard moest worden.

Nu is het aan jou om die keuze te maken.

Softwareontwikkelaars lossen merge conflicts regelmatig op. Het hoort bij samenwerken aan software.

? Uitleg

Wanneer Git een merge conflict ontdekt, zet het beide versies tijdelijk in hetzelfde bestand.

VS Code laat deze verschillen overzichtelijk zien. Je kunt vervolgens kiezen:

- **Accept Current Change** → behoud alleen jouw wijziging.
- **Accept Incoming Change** → behoud alleen de wijziging van je klasgenoot.
- **Accept Both Changes** → behoud beide wijzigingen.

Nadat je een keuze hebt gemaakt, moet je het bestand opnieuw opslaan, committen en pushen.

```
Conflict gevonden
        │
        ▼
Kies de juiste versie
        │
        ▼
Opslaan
        │
        ▼
git add .
git commit
git push
```

?? Opdracht – Los het conflict op

1. Open `index.html` in VS Code.
2. Bekijk de conflictmarkeringen.
3. Kies **Accept Both Changes**.
4. Controleer of beide teksten nu in het bestand staan.
5. Sla het bestand op.
6. Voer daarna uit:

```
git add .
git commit -m "Merge conflict opgelost"
git push
```

7. Open GitHub en controleer of de nieuwe versie online staat.

? Controle

Als het goed is:

- is de foutmelding verdwenen;
- staan beide wijzigingen in `index.html`;
- is de laatste commit zichtbaar op GitHub.

? Denkvraag

Wanneer zou je *Accept Both Changes* kiezen, en wanneer zou je juist één van de andere opties gebruiken?

? Wist je dat?

Merge conflicts lijken in het begin lastig, maar ervaren softwareontwikkelaars lossen ze regelmatig op. Door vaak kleine commits te maken en regelmatig `git pull` uit te voeren, voorkom je de meeste conflicten.

? Inleveren

1. Een screenshot van VS Code waarop het merge conflict zichtbaar is.
2. Een screenshot nadat het conflict is opgelost.
3. Een screenshot van GitHub waarop de commit `Merge conflict opgelost` zichtbaar is.
4. Het antwoord op de denkvraag in 2 tot 5 zinnen.

9. Veilig experimenteren met Branches

? Leerdoelen

- Je begrijpt wat een branch is.
- Je kunt een nieuwe branch maken.
- Je kunt op een branch werken zonder de hoofdversie te wijzigen.
- Je kunt een branch naar GitHub pushen.

? Situatie uit de praktijk

Stel dat je een compleet nieuw ontwerp voor een website wilt maken.

Wat gebeurt er als het nieuwe ontwerp toch niet mooi blijkt te zijn?

Als je rechtstreeks in de `main`-branch werkt, kan het zijn dat je de werkende website kapot maakt.

Daarom maken softwareontwikkelaars eerst een **branch**. Dat is een kopie van het project waarop je veilig kunt experimenteren.

? Uitleg

Een **branch** is een aparte werkomgeving binnen dezelfde repository.

Je kunt nieuwe ideeën uitproberen zonder dat de hoofdversie verandert.

Pas als alles goed werkt, voeg je de branch samen met de `main`-branch.

```
main
──────────●──────────
           \
            \
nieuwe-layout
            ●──────●
```

Vrijwel iedere professionele softwareontwikkelaar werkt dagelijks met branches.

?? Opdracht – Maak je eerste branch

1. Open de terminal in VS Code.
2. Maak een nieuwe branch met de naam `nieuwe-layout`:

```
git checkout -b nieuwe-layout
```

3. Controleer met `git branch` dat je op de nieuwe branch werkt.
4. Open `index.html`.
5. Verander de achtergrondkleur van de pagina.
6. Voeg eventueel ook een extra koptekst of alinea toe.
7. Voer daarna uit:

```
git add .
git commit -m "Nieuwe layout gemaakt"
git push -u origin nieuwe-layout
```

8. Open GitHub.
9. Controleer of de nieuwe branch zichtbaar is in het branch-overzicht.

? Controle

Controleer op GitHub of je twee branches hebt:

- `main`
- `nieuwe-layout`

Je wijzigingen staan alleen in `nieuwe-layout`. De `main`-branch is nog niet aangepast.

? Denkvraag

Waarom is het veiliger om grote wijzigingen eerst op een branch te maken dan direct op `main`?

? Extra uitdaging

Maak nog een branch met de naam `experiment`.

Voeg hierin een opvallende achtergrondkleur of een extra HTML-element toe.

Schakel daarna terug naar `main` met:

```
git checkout main
```

Zie je dat de wijzigingen uit de branch verdwenen zijn? Dat komt omdat ze nog niet zijn samengevoegd.

? Inleveren

1. Een screenshot van GitHub waarop de branches `main` en `nieuwe-layout` zichtbaar zijn.
2. Een screenshot van de terminal waarin de branch is aangemaakt.
3. Het antwoord op de denkvraag in 2 tot 5 zinnen.

10. Eindopdracht – Samen bouwen met GitHub

? Leerdoelen

- Je past alle Git-vaardigheden uit deze module toe.
- Je werkt succesvol samen met een klasgenoot.
- Je laat zien dat je GitHub kunt gebruiken zoals professionele softwareontwikkelaars dat doen.

? Situatie uit de praktijk

Je bent nu softwareontwikkelaar bij een bedrijf.

Je krijgt samen met een collega de opdracht om een eenvoudige website te bouwen. Jullie werken allebei aan hetzelfde project en gebruiken GitHub om samen te werken.

Gebruik alles wat je in deze module hebt geleerd.

?? Eindopdracht

Werk samen met een klasgenoot en breid jullie website uit.

De website moet minimaal bevatten:

- een homepage (`index.html`);

- minimaal drie extra HTML-pagina's;
- een navigatiemenu tussen de pagina's;
- een eenvoudig CSS-bestand voor de opmaak.

Gebruik tijdens het bouwen GitHub zoals je in deze module hebt geleerd.

? Eisen

- Iedere student maakt minimaal **3 commits**.
- Iedere student voert minimaal **2 keer een git pull** uit.
- Iedere student maakt minimaal **1 branch**.
- Minimaal één branch wordt naar GitHub gepusht.
- Er wordt minimaal één merge conflict veroorzaakt én opgelost.
- Iedere commit heeft een duidelijke commit message.

? Reflectie

1. Waarom is `git pull` meestal het eerste commando dat je uitvoert?
2. Waarom zijn branches handig bij grotere projecten?
3. Hoe helpt GitHub om samen te werken zonder elkaars werk kwijt te raken?
4. Welk onderdeel van Git vond jij het lastigst? Leg uit waarom.

? Inleveren

1. De URL van jullie GitHub-repository.
2. Een screenshot van de commitgeschiedenis.
3. Een screenshot van de branches.
4. Een screenshot van de uiteindelijke website.
5. De antwoorden op de reflectievragen in een PDF.

? Klaar?

Gefeliciteerd! Je hebt de belangrijkste basisvaardigheden van Git en GitHub geleerd:

- ✔ Repository maken
- ✔ Clone
- ✔ Commit
- ✔ Push
- ✔ Pull
- ✔ Samenwerken
- ✔ Merge Conflicts oplossen
- ✔ Branches gebruiken

Dit zijn dezelfde basisvaardigheden die softwareontwikkelaars dagelijks gebruiken.

--

Revision #8
Created 2026-04-09 19:53:02 UTC by Max
Updated 2026-06-25 19:52:10 UTC by Max