PROFESSIONALISEREN — UNIT 1

Versiebeheer

Leer hoe software developers hun bestanden organiseren en de verschillende versies van hun code beheren — van een nette mappenstructuur tot werken met Git en GitHub.

Week 1

Bestanden, folders en een mappenstructuur

Leerdoel: je weet wat bestanden en folders zijn, en je hebt een duidelijke mappenstructuur voor je schoolwerk gemaakt

Waar gaat deze week over?

Je gaat de komende jaren met heel veel bestanden werken: terwijl je aan het coderen bent, maar ook gewoon voor je schoolwerk. Een duidelijke **mappenstructuur** helpt je om gemaakt werk niet kwijt te raken en maakt het makkelijker om back-ups te maken.

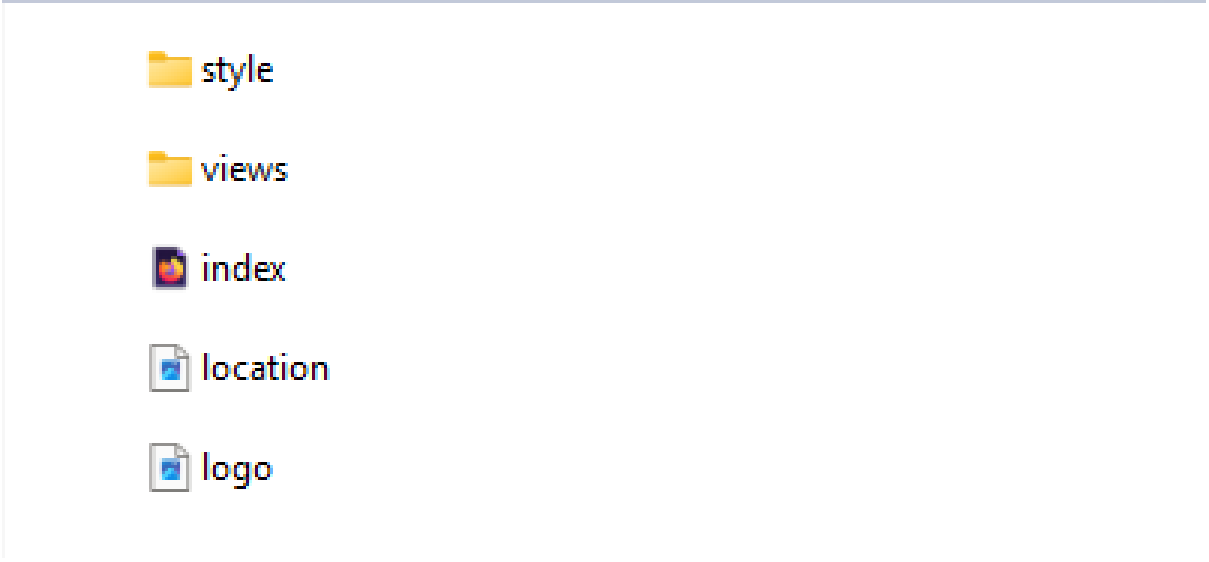
Voor de lessen van de volgende weken is het belangrijk dat je vlot met bestanden en folders kunt werken. Daarom starten we daarmee.

Aan het einde van deze week weet je wat bestanden en bestandsextensies zijn, wat folders zijn, heb je een eigen mappenstructuur voor je schoolwerk gemaakt, en weet je hoe je OneDrive gebruikt om je werk veilig te bewaren.

1.1 Bestanden

Een bestand is 'iets' op je laptop, computer of MacBook. Voorbeelden zijn een Word-document, een game-*executable* waarmee je een spel start, of een foto.

Als jij straks als software developer websites maakt, werk je ook met bestanden. Een website bestaat namelijk uit verschillende soorten bestanden die naar elkaar verwijzen.



1.2 Bestandsextensies

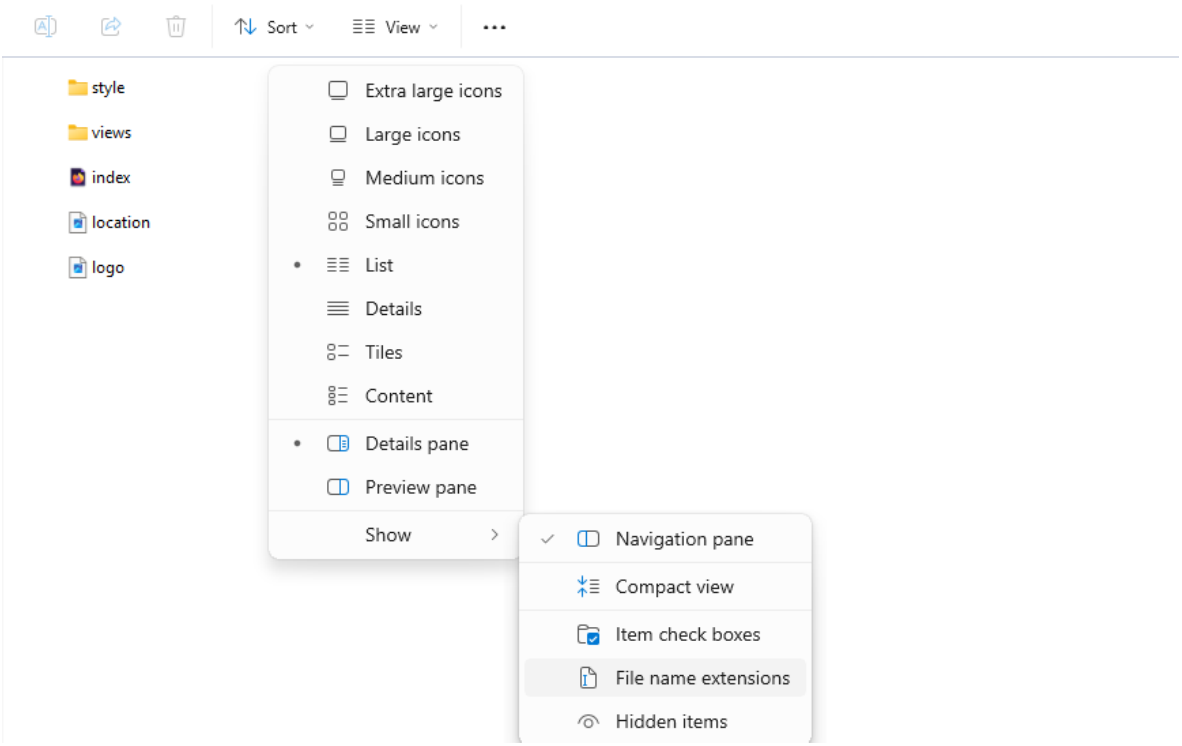
Elk bestand heeft een **extensie**. Dankzij die extensie begrijpen apparaten wat voor soort bestand iets is. De extensie staat achter de punt in de bestandsnaam, bijvoorbeeld `index.html` of `logo.png`.

Een paar veelvoorkomende extensies:

1.3 Bestandsextensies tonen

Standaard verbergt Windows de extensies. Voor software developers is het juist handig om ze te zien — dan weet je precies wat voor soort bestand iets is.

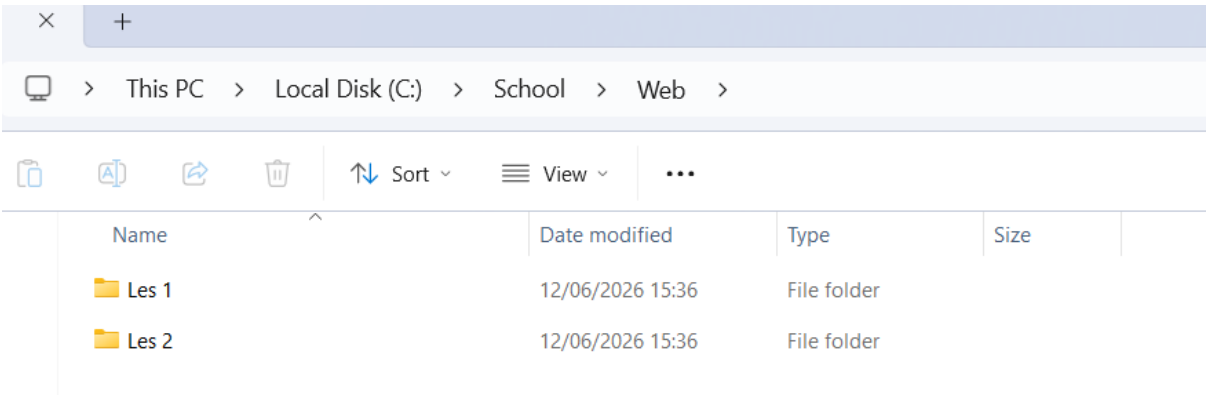
Open je **File Explorer** en zet onder de tab **View** de optie **File name extensions** aan.

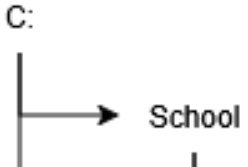


1.4 Mappen en een mappenstructuur

Op je laptop werk je met verschillende **mappen** (ook wel *folders* genoemd) waarin je bestanden zet. Je kunt ook mappen aanmaken binnen andere mappen. Die opbouw noemen we een **mappenstructuur**.

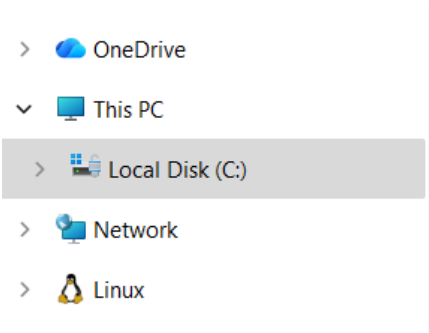
Terwijl je software ontwikkelt is het belangrijk dat je weet op welke plaats bestanden opgeslagen worden, en in welke folder dat gebeurt.



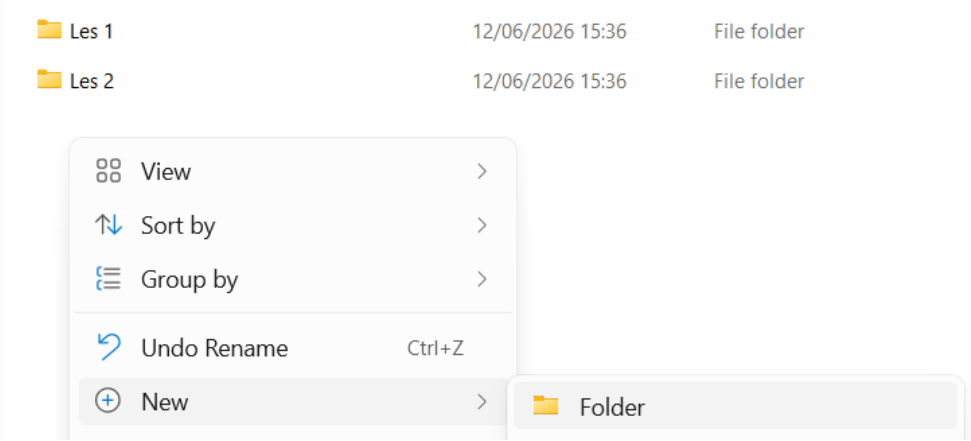


1.5 Nieuwe mappen maken

Maak een nieuwe folder aan op je `C:`- of `D:`-schijf. Selecteer eerst de schijf in je File Explorer.



Klik daarna met de **rechtermuisknop** in de File Explorer om een nieuwe folder aan te maken.

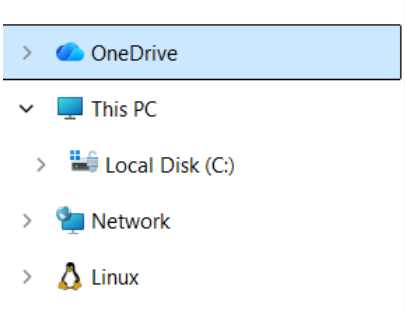


1.6 OneDrive

Op Windows-laptops heb je toegang tot **OneDrive**. OneDrive is een plaats om bestanden en folders op het internet op te slaan.

We raden je op school aan om OneDrive te gebruiken, zodat je gemaakte bestanden nooit kwijtraken — ook niet als je laptop kapotgaat. Je bestanden staan dan namelijk nog steeds op het internet.

Je kunt je OneDrive selecteren door deze aan te klikken in je File Explorer. Let op dat je ingelogd moet zijn met een Microsoft-account.



1.7 Folders en bestanden tijdens je schooltijd

Terwijl je de opleiding software developer volgt, gaan je docenten ervan uit dat je bestanden zoals Word-documenten op je OneDrive opslaat, zodat je ze niet kwijtraakt.

Let op: voor **code** geldt het tegenovergestelde. Code sla je juist *niet* op via OneDrive — sommige applicaties werken dan namelijk niet goed. Voor het opslaan van code gebruiken we als developers een techniek die **Git** heet. Daarover leer je vanaf week 2 meer.

1.8 Onthoud voor nu het volgende

- Sla **code** niet op via OneDrive.
- Sla **andere bestanden** (zoals documenten) juist wél op via OneDrive.

Volgende week leer je hoe je je code op de juiste manier opslaat met Git.

Oefeningen

Bestandsextensies tonen

Standaard verbergt Windows de bestandsextensies. Als developer wil je ze juist zien.

Wat ga je doen?

1. Open je **File Explorer**.
2. Ga naar de tab **View** (Beeld).
3. Zet de optie **File name extensions** (Bestandsnaamextensies) aan.

Klaar als…

Je ziet nu achter elke bestandsnaam de extensie staan, bijvoorbeeld `document.docx` of `foto.png`.

Bestanden op je laptop verkennen

Kijk eens rond op je eigen laptop. Welke bestandsextensies kom je tegen, en wat betekenen ze?

Wat ga je doen?

Zoek minstens **vijf verschillende bestandsextensies** op je laptop en noteer per extensie wat voor soort bestand het is. Vul de tabel hieronder in (neem hem over in een document).

Tip: kom je een extensie tegen die je niet kent? Zoek hem op via een zoekmachine — zo leer je je laptop steeds beter kennen.

Een mappenstructuur maken voor je schoolwerk

Je gaat nu zelf een mappenstructuur maken voor je schoolwerk, zodat je gemaakt werk niet kwijtraakt.

Wat ga je doen?

1. Maak een nieuwe folder op je laptop met de naam `school`. Klik hiervoor met de **rechtermuisknop** in je File Explorer en kies *Nieuwe map*.
2. Maak **binnen** de `school`-folder een aparte folder voor elk van je vakken en/of projecten. De namen moeten vooral voor jou duidelijk zijn.
3. Zorg dat het werk dat je voor elk vak maakt op de juiste plaats wordt opgeslagen. Heb je al bestanden die op een andere plek staan? Verplaats die dan nu naar de juiste folder.

Klaar als…

Je hebt een `school`-folder met daarin een nette onderverdeling per vak, en je bestaande schoolwerk staat op de juiste plek.

Je mappenstructuur naar OneDrive zetten

Verplaats (of kopieer) je gemaakte mappenstructuur naar **OneDrive**, zodat je werk veilig staat — ook als je laptop kapotgaat.

Wat ga je doen?

1. Selecteer alle bestanden en folders die je wilt kopiëren of verplaatsen (sleep met je cursor om een selectie te maken).
2. Druk daarna op een van deze toetscombinaties:
3. Ga naar je **OneDrive** in de File Explorer en druk op `CTRL + V` om de bestanden daar neer te zetten.

Onthoud: documenten en schoolwerk zet je wél op OneDrive. **Code** zet je er juist *niet* op — dat doe je vanaf volgende week met Git.

Klaar als…

Je `school`-mappenstructuur staat (ook) op je OneDrive.

Jouw mappenstructuur voor school

Inleveropdracht Week 1

Je hebt deze week geleerd hoe je je schoolwerk overzichtelijk bewaart. Nu laat je zien dat je dat zelf kunt. Een nette mappenstructuur is de eerste stap in versiebeheer: je weet altijd waar je werk staat en raakt niets kwijt.

Maak een mappenstructuur voor je schoolwerk met een hoofdfolder `school` en daarin een aparte folder per vak en/of project. Zorg dat de structuur (ook) op je OneDrive staat. Maak hier screenshots van en schrijf een korte uitleg van je indeling. Lever je werk in via **Itslearning**, waar je docent het nakijkt en je feedback geeft.

In te leveren

- Een screenshot van je 'school'-folder met daarin een onderverdeling per vak/project
- Een screenshot waaruit blijkt dat de structuur (ook) op OneDrive staat
- Een korte uitleg (paar zinnen) van je gekozen indeling

Beoordelingscriteria

• Er is een hoofdfolder 'school' aangemaakt (2 pt)
• Binnen 'school' staat een aparte folder per vak en/of project (3 pt)
• De mappen hebben duidelijke, logische namen (2 pt)
• De mappenstructuur staat (ook) op OneDrive (2 pt)
• De korte uitleg beschrijft waarom de indeling logisch is (1 pt)

Tips

- Bedenk eerst welke vakken en projecten je hebt voordat je mappen aanmaakt.
- Zet bestandsextensies aan, dan zie je beter wat er in je mappen staat.
- Maak je screenshots met de Snipping Tool (Windows-toets + Shift + S).

Week 2

Git en GitHub

Leerdoel: je weet wat Git en GitHub zijn, en je kunt een repository forken en clonen naar je eigen laptop

Waar gaat deze week over?

In week 1 heb je geleerd wat bestanden zijn en hoe je ze opslaat. Wat je nog niet weet, is hoe wij als developers de verschillende **versies** van onze code bijhouden en bewaren. Dat ga je nu leren.

Aan het einde van deze week weet je wat Git en GitHub zijn, weet je wat een repository en een fork zijn, en heb je voor het eerst code van het internet naar je eigen laptop gekopieerd (*clonen*).

2.1 Over Git

Software developers gebruiken meestal een techniek met de naam **Git** om de verschillende versies van hun code te beheren.

Met Git kun je je code opslaan en aanpassen, maar je kunt ook **terugkijken** naar vorige versies en precies bijhouden wanneer welke wijziging is gemaakt. Dat is voor developers erg nuttig.

Eerlijk is eerlijk: Git is niet de makkelijkste techniek om mee te starten. Daarom begin je er meteen aan het begin van je opleiding mee, zodat je het de rest van je studie kunt gebruiken.

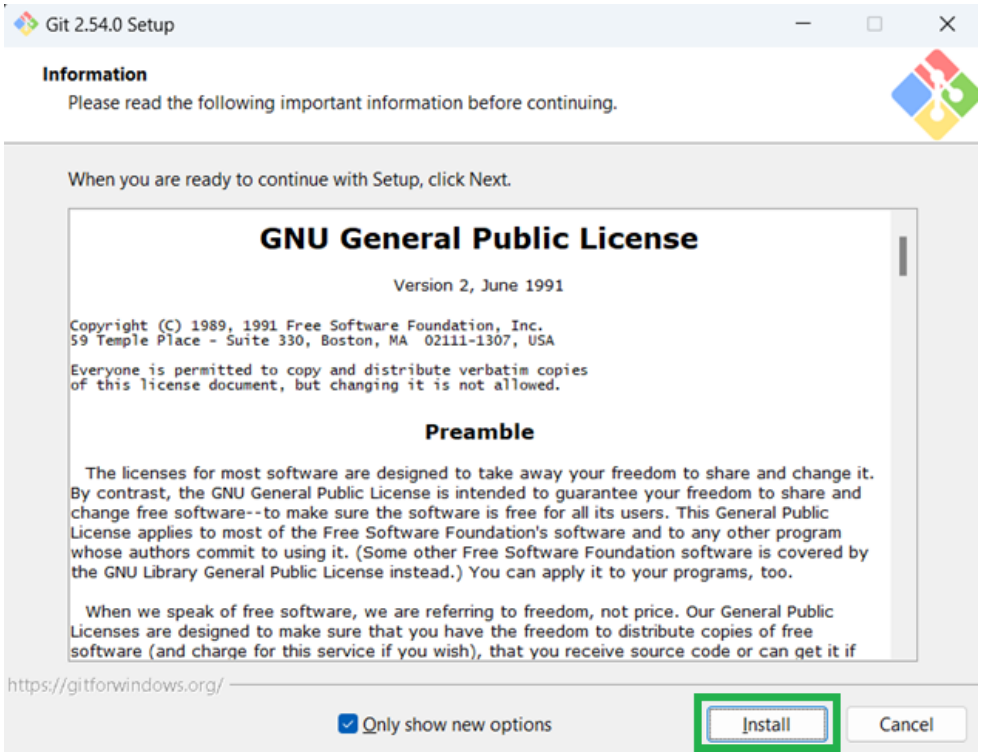
2.2 Git installeren

Voordat je Git kunt gebruiken, moet je het installeren op je laptop.

Ga naar git-scm.com/install/windows en download de nieuwste versie van Git.

Nadat je de gedownloade *executable* uitvoert, zie je het installatiescherm. Klik op installeren

en sluit de installer niet voordat deze klaar is.



2.3 Een GitHub-account maken

Je hebt nu Git geïnstalleerd. Daarmee kan je laptop code online zetten en beheren. Maar je hebt ook een plaats nodig om die code écht op te slaan. Daarvoor gebruiken wij **GitHub**.

Er bestaan ook andere plaatsen om code op te slaan met Git, maar GitHub is degene die wij gebruiken.

Ga naar github.com en maak een eigen account. Let daarbij op het volgende:

Je account blijft **altijd** toegankelijk. Een goede optie als je van plan bent zelf projecten te starten die je wilt bewaren.

Je schoolmail wordt een tijd na je afstuderen verwijderd, waardoor je account ontoegankelijk kan worden. Handig als je school en privé gescheiden wilt houden.

Onthoud goed: je gebruikersnaam, wachtwoord én het gebruikte e-mailadres. Je hebt deze de rest van je opleiding nog nodig. Je gebruikersnaam kun je later wijzigen, je gekozen e-mailadres niet.

Na het aanmaken krijg je een validatie-e-mail. Klik op de link daarin om je account te activeren.

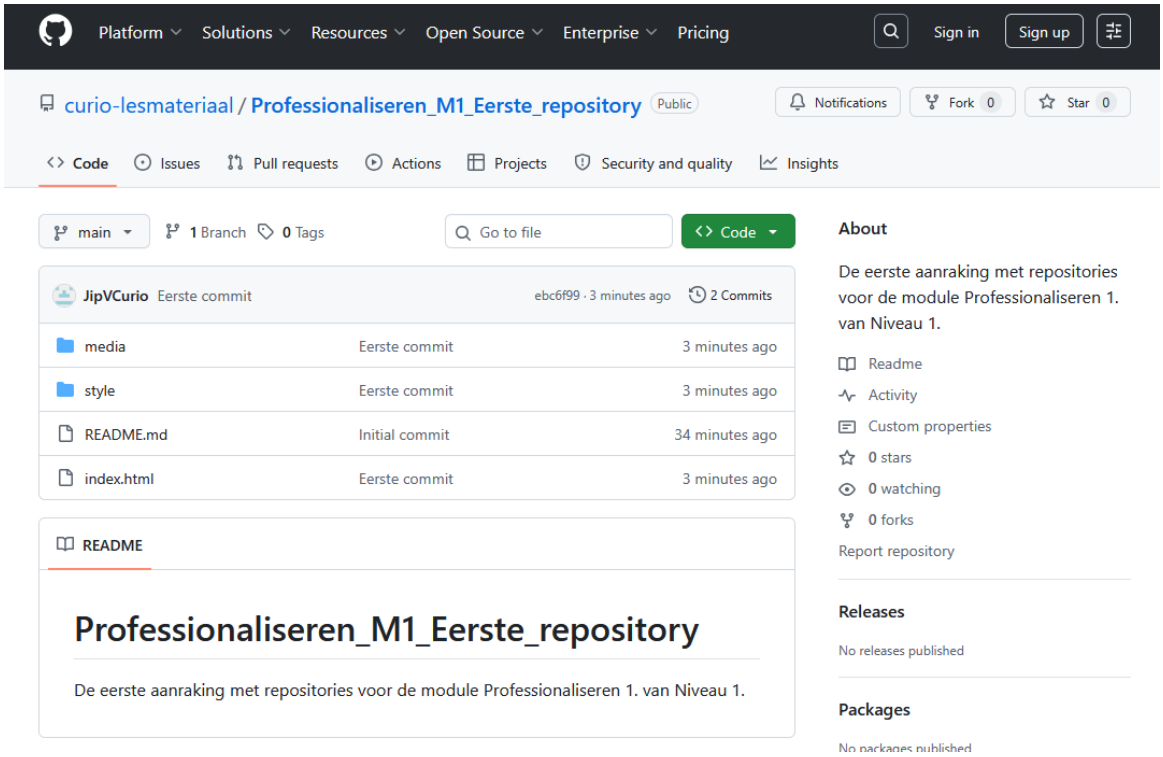
2.4 Repositories

Alle code op GitHub wordt opgeslagen in een **repository**. Vertaal je 'repository' naar het

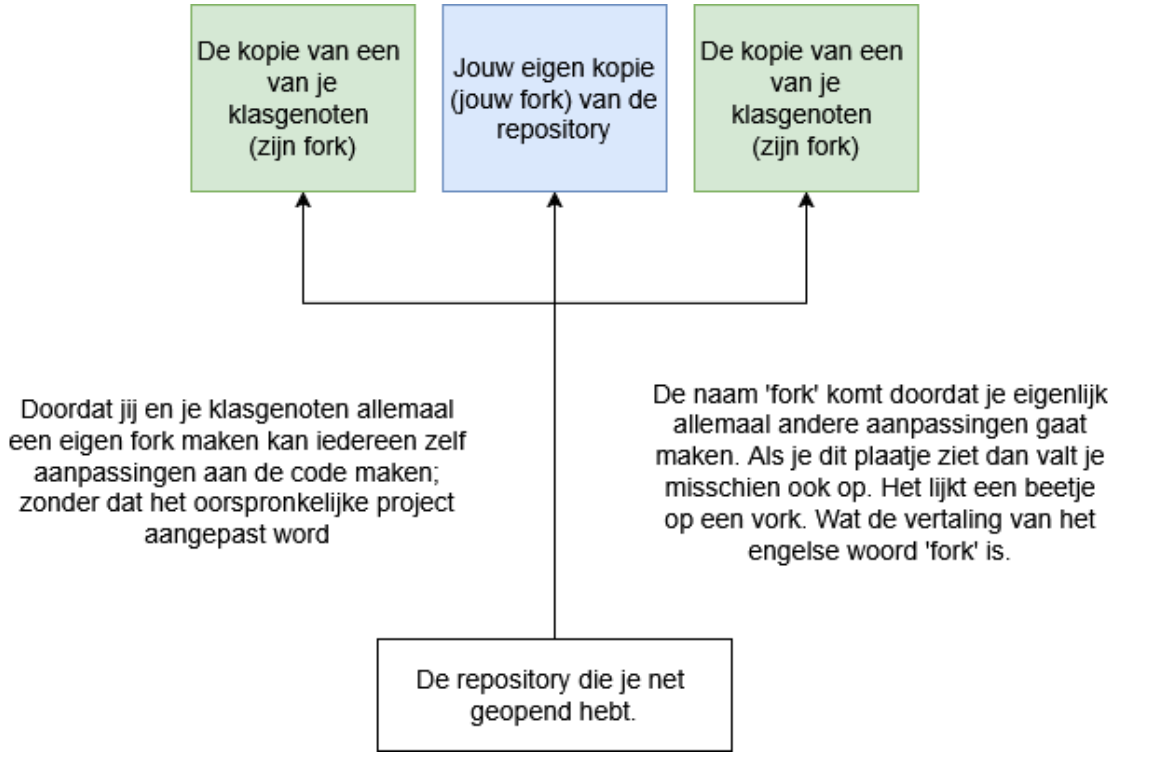
Nederlands, dan krijg je het woord *opslagplaats*. En dat is precies wat het is: een opslagplaats voor code!

2.5 Forks van repositories

Voordat je zelf een repository maakt, laten we je eerst ervaren waarom repositories nuttig zijn. Zorg dat je ingelogd bent op GitHub en ga naar deze repository:github.com/curio-lesmateriaal/Professionaliseren_M1_Eerste_repository

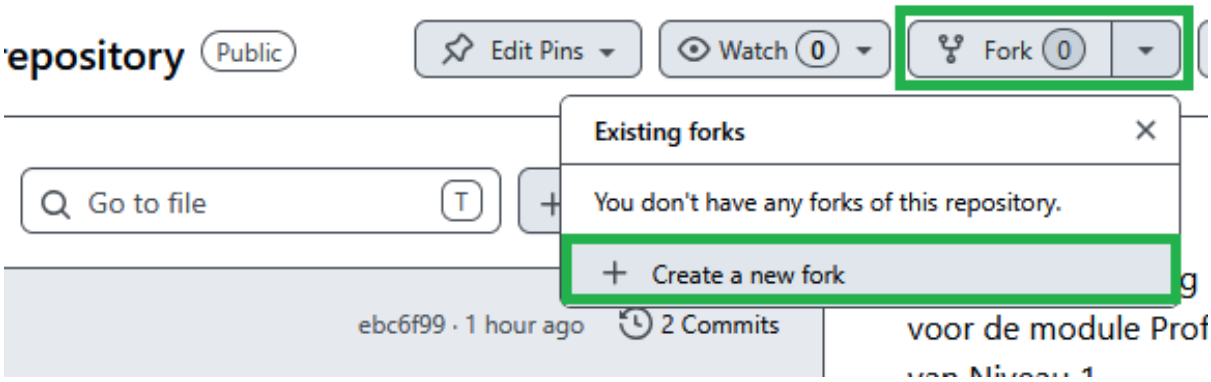


Je gaat zo een **fork** van deze repository maken. Een fork maken betekent dat je een **eigen kopie** van een repository maakt waar je zelf op door kunt werken — zonder dat je de originele repository verandert.

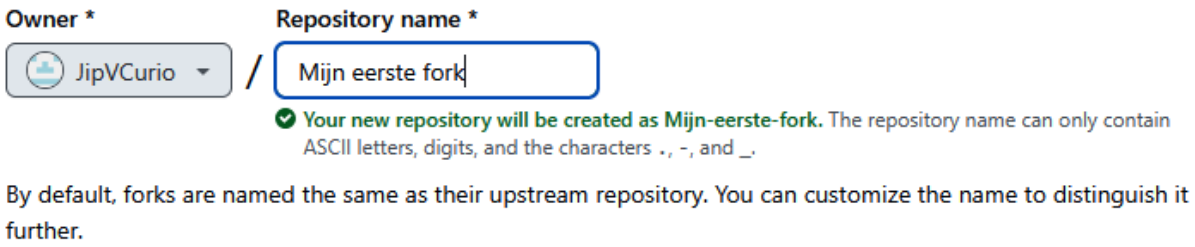


2.6 Een bestaande repository forken

Nu je weet wat een fork is, ga je er zelf één maken van de voorbeeldrepository. Zorg dat de repository openstaat in je browser en klik op de **Fork**-knop.



Geef je repository de naam **mijn eerste fork**. Je ziet dat de naam automatisch `mijn-eerste-fork` wordt, met jouw gebruikersnaam ervoor. Dit is namelijk jouw eigen kopie.



Klik daarna op **Create fork**. Er wordt nu een kopie gemaakt en aan jouw GitHub-account gekoppeld. Daarna word je doorgestuurd naar je eigen kopie — dat zie je linksbovenin je scherm.

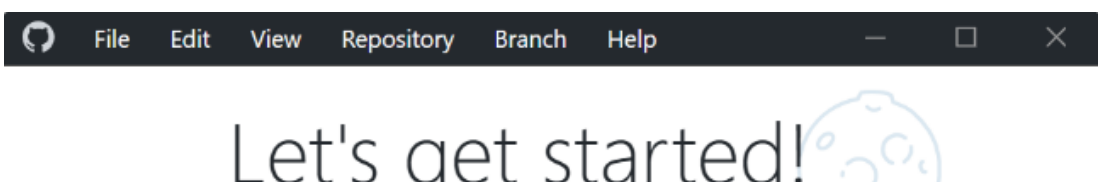
Je weet nu hoe je een kopie maakt van een bestaande repository.

2.7 Een Git-tool kiezen

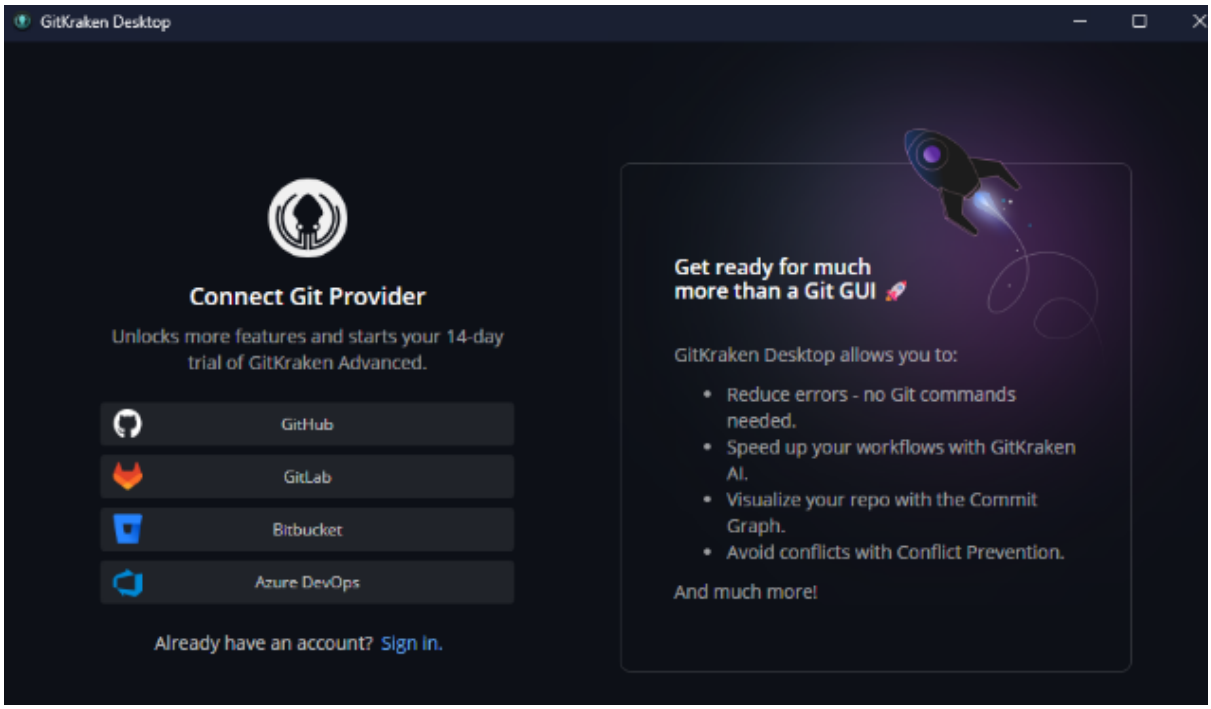
Je hebt een fork gemaakt op GitHub. In de volgende stap leer je hoe je die code naar je eigen laptop kopieert. Daarvoor heb je een **programma** nodig.

Vroeger deden we dit via de command prompt (dat kan nog steeds!), maar dan moet je veel commando's onthouden. Daarom raden we je aan een hulpmiddel te kiezen. Een paar voorbeelden:

De tool die door GitHub zelf is gemaakt. Goed startpunt voor beginners.desktop.github.com/download



Wordt gebruikt door een aantal grote bedrijven. Let op: de gratis versie werkt alleen met *publieke* repositories.gitkraken.com/git-client

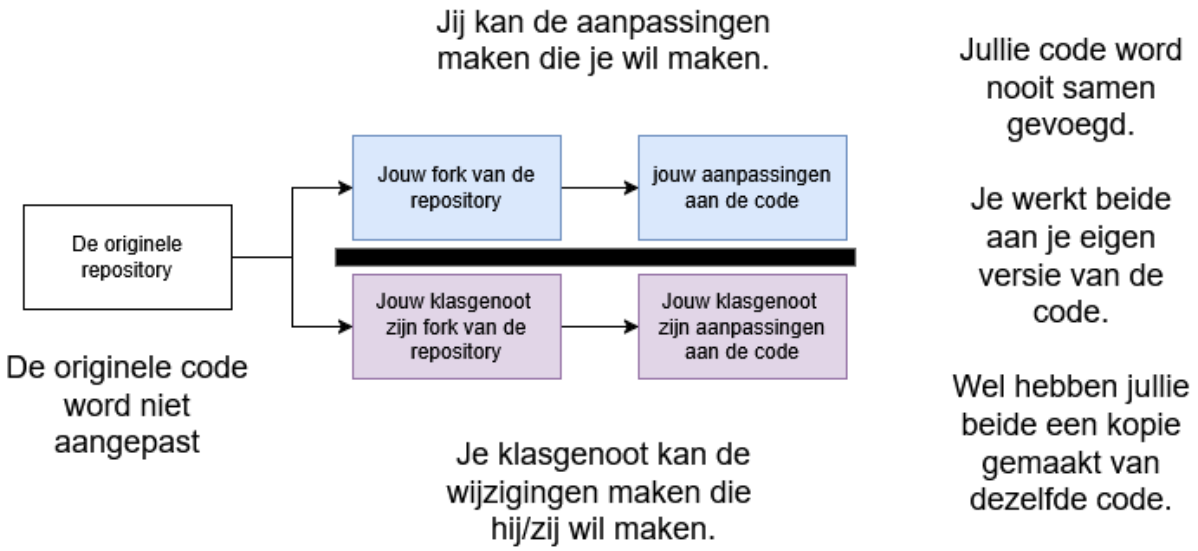


Meer voorbeelden vind je op git-scm.com/tools/guis. Kies en installeer in elk geval één tool waarmee jij met Git gaat werken. Je mag later altijd nog wisselen naar een tool die je fijner vindt.

De meeste Git-tools zijn gratis, maar niet allemaal. Sommige hebben betaalde functies — maar alles wat je tijdens je studie nodig hebt, kun je gratis blijven doen.

2.8 Het clonen van een repository

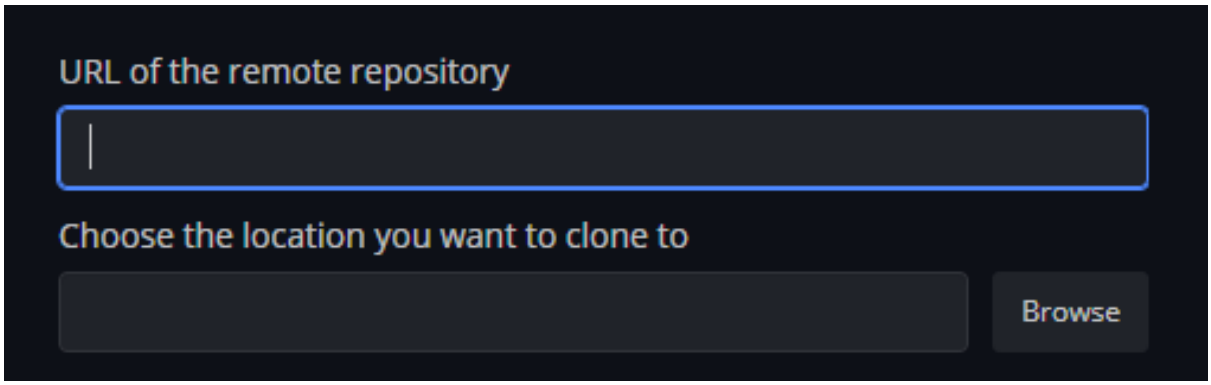
Je hebt nu een tool gekozen. Met een fork heb je een eigen kopie van een online repository. Om er ook echt wijzigingen aan te maken, moet je de code eerst naar je laptop kopiëren. Dat kopiëren noemen we **clonen**.

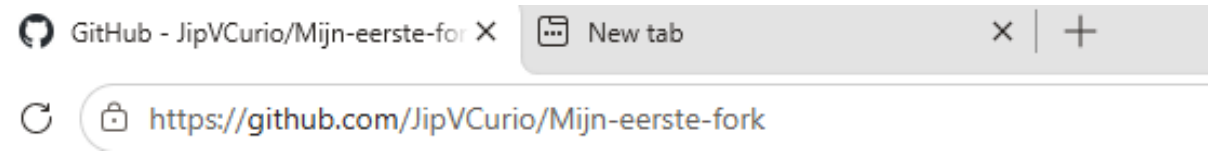


2.9 Je repository clonen naar je laptop

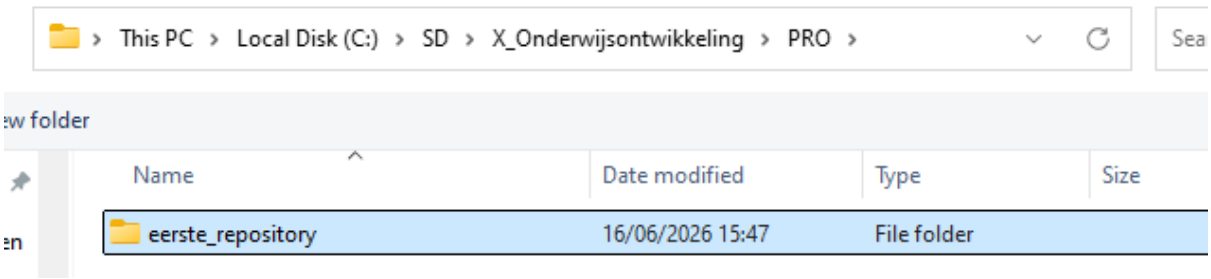
Zoek in je gekozen tool naar de optie om een repository te **clonen**. Mogelijk moet je eerst inloggen met je GitHub-gegevens. Om te clonen vul je twee dingen in:

1. De remote repository — de URL waar je online repository staat. Die kopieer je op GitHub met `CTRL + C`.





2. De locatie om naartoe te clonen — de (lege!) folder op je laptop waar de code terecht moet komen. Gebruik de *browse*-knop om een folder te kiezen.

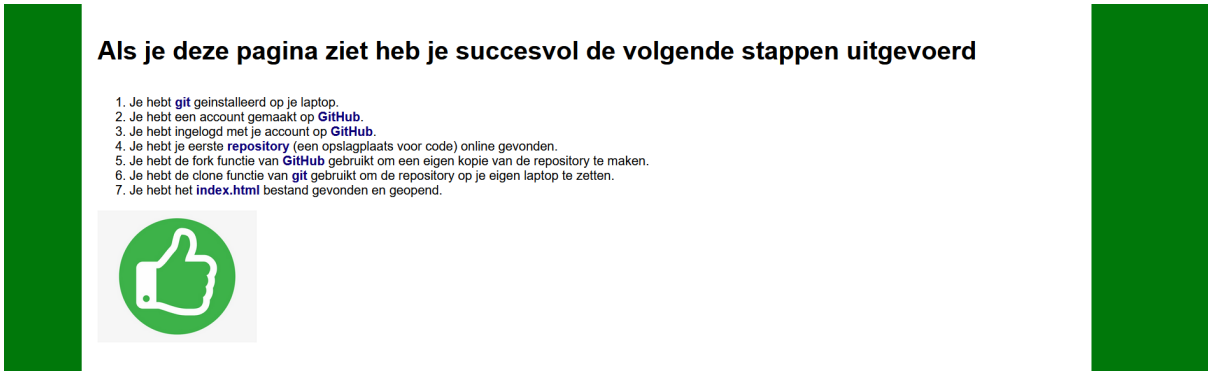


Tip: geef de folder een duidelijke naam als `eerste_repository` en zet hem op de juiste plaats in je mappenstructuur uit week 1.

Druk daarna op de **clone**-knop. Je hebt nu de online code naar je laptop gekopieerd.

2.10 Controleren of het clonen gelukt is

Open op je laptop de folder die je als clone-locatie koos. Daarin zou een `index.html`-bestand moeten staan. Dubbelklik erop om het te openen. Als alles goed ging, zie je nu deze website:



2.11 Volgende week

Je hebt deze week verschillende Git-technieken gebruikt om een kopie van een online repository op je eigen laptop te zetten. Volgende week leer je hoe je **zelf** code op een repository zet.

Oefeningen

Git installeren

Voordat je met Git kunt werken, installeer je het op je laptop.

Wat ga je doen?

1. Ga naar git-scm.com/install/windows.
2. Download de **nieuwste versie** van Git.
3. Voer de gedownloade *executable* uit en doorloop de installatie. Sluit de installer niet voordat deze klaar is.

Klaar als…

Git is volledig geïnstalleerd op je laptop.

Een GitHub-account maken

Om code online op te slaan heb je een GitHub-account nodig.

Wat ga je doen?

1. Ga naar github.com.
2. Kies bewust welk e-mailadres je gebruikt:
3. Maak je account aan en activeer het via de validatie-e-mail die je ontvangt.

Onthoud goed: je gebruikersnaam, wachtwoord en gebruikte e-mailadres. Je hebt deze de rest van je opleiding nodig.

Klaar als…

Je hebt een geactiveerd GitHub-account en bent ingelogd.

Een bestaande repository forken

Je maakt nu je eerste **fork**: een eigen kopie van een bestaande repository.

Wat ga je doen?

1. Zorg dat je ingelogd bent op GitHub.
2. Ga naar github.com/curio-lesmateriaal/Professionaliseren_M1_Eerste_repository.
3. Klik op de **Fork**-knop.
4. Geef je fork de naam **mijn eerste fork** (wordt automatisch `mijn-eerste-fork`).
5. Klik op **Create fork**.

Klaar als…

Je wordt doorgestuurd naar je eigen kopie van de repository. Linksbovenin staat jouw gebruikersnaam vóór de repositorynaam.

Een Git-tool kiezen en installeren

Om code van GitHub naar je laptop te halen, heb je een Git-tool nodig.

Wat ga je doen?

1. Kies een tool waarmee jij met Git gaat werken. Voorbeelden:
2. Installeer de gekozen tool.

We dagen je uit om meerdere tools te proberen en door te gaan met degene die jij het fijnste vindt. Wisselen mag altijd nog.

Klaar als…

Je hebt één Git-tool geïnstalleerd en kunt deze openen.

Je fork clonen naar je laptop

Nu ga je de code van je fork **clonen** (kopiëren) naar je eigen laptop.

Wat ga je doen?

1. Open je Git-tool en zoek de optie om een repository te **clonen** (log eventueel eerst in).
2. Vul de **remote repository** in: de URL van *jouw* fork (kopieer deze op GitHub met `CTRL + C`).
3. Kies een **lege locatie** op je laptop om naartoe te clonen. Geef de folder een duidelijke naam zoals `eerste_repository` en zet hem op de juiste plek in je mappenstructuur.
4. Druk op de **clone**-knop.

Klaar als…

De code staat in de gekozen folder op je laptop.

Controleren of het clonen gelukt is

Controleer of het clonen goed is gegaan.

Wat ga je doen?

1. Open in je File Explorer de folder waarnaar je hebt gecloond.
2. Zoek het bestand `index.html`.
3. Dubbelklik erop om het in je browser te openen.

Klaar als…

Je ziet de website van de voorbeeldrepository in je browser. Lukt dit niet? Vraag je docent om hulp — het is belangrijk dat Git goed werkt voor de rest van je opleiding.

Je eerste fork en clone

Inleveropdracht Week 2

Deze week heb je kennisgemaakt met Git en GitHub. Nu laat je zien dat je een repository kunt forken en clonen. Dit is de basis waarmee developers samen aan code werken.

Maak een fork van de repository Professionaliseren_M1_Eerste_repository, clone die fork naar je laptop, en open de `index.html` in je browser. Maak hier screenshots van. Lever je werk in via **Itslearning**, waar je docent het nakijkt en je feedback geeft.

In te leveren

- Een screenshot van je fork op GitHub (met jouw gebruikersnaam in de repositorynaam)
- Een screenshot van de gecloonde folder op je laptop
- Een screenshot van de geopende index.html in je browser

Beoordelingscriteria

• Git is geïnstalleerd en er is een werkend GitHub-account (2 pt)
• Er is een fork gemaakt van de voorbeeldrepository (3 pt)
• De fork is succesvol gecloond naar een nette locatie op de laptop (3 pt)
• De index.html opent en toont de website in de browser (2 pt)

Tips

- Zet de gecloonde folder op een logische plek in je mappenstructuur uit week 1.
- Lukt het clonen niet? Controleer of je de juiste repository-URL gebruikt en of de doelmap leeg is.
- Vraag je docent om hulp als Git niet werkt — dat is belangrijk voor de rest van je opleiding.

Week 3

Code beheren via Git

Leerdoel: je kunt een eigen repository maken, code committen met een logische titel en pushen naar GitHub

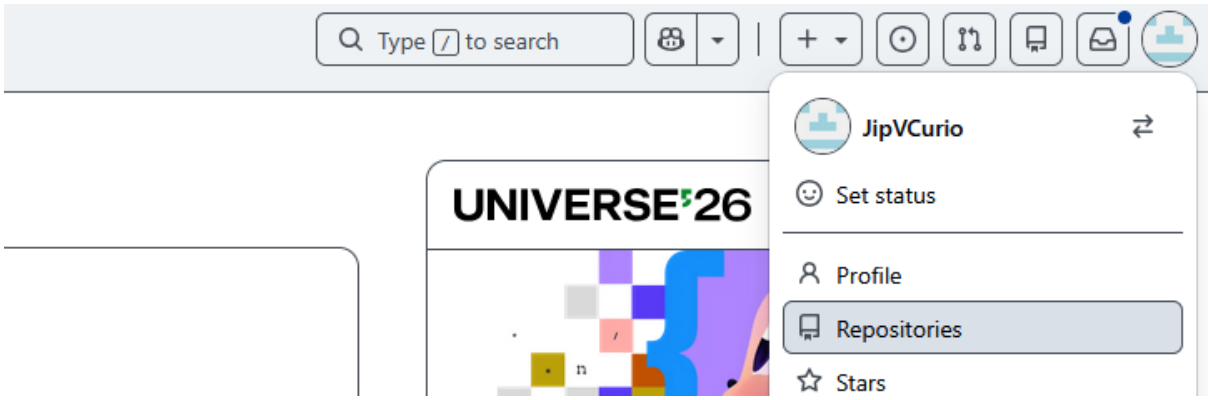
Waar gaat deze week over?

Vorige week leerde je hoe je code uit een online repository naar je laptop kopieert (*clonen*). Deze week draai je het om: je maakt **zelf** een repository, schrijft eigen code en zet die online.

Aan het einde van deze week weet je hoe je een eigen repository maakt, wat een commit is, en hoe je je code met een push naar GitHub stuurt. Ook kun je je commitgeschiedenis terugkijken.

3.1 Je eigen (lege) repository maken

1. Ga naar github.com, log in en ga naar **Repositories**.

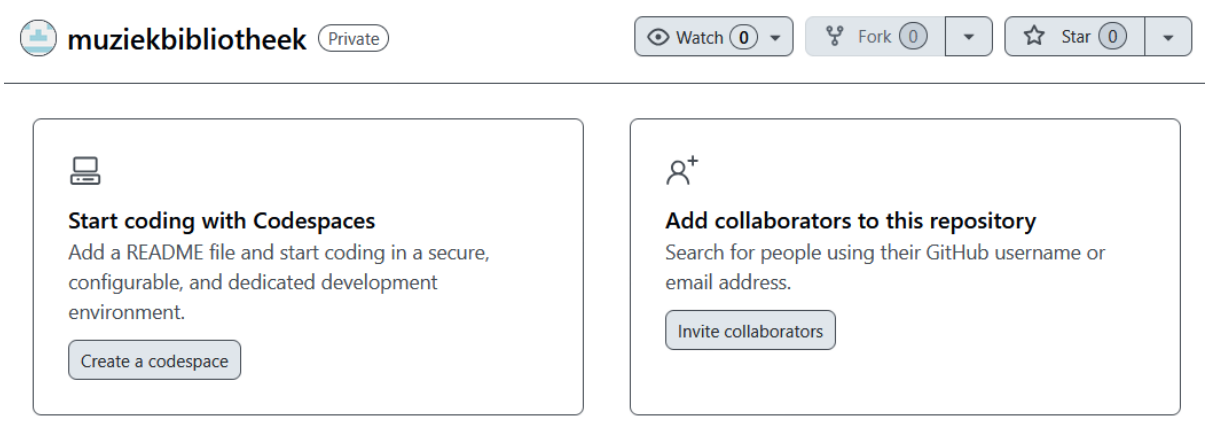


1. Klik op **New**.

1. Vul de volgende gegevens in:
2. Klik op **Create repository**.

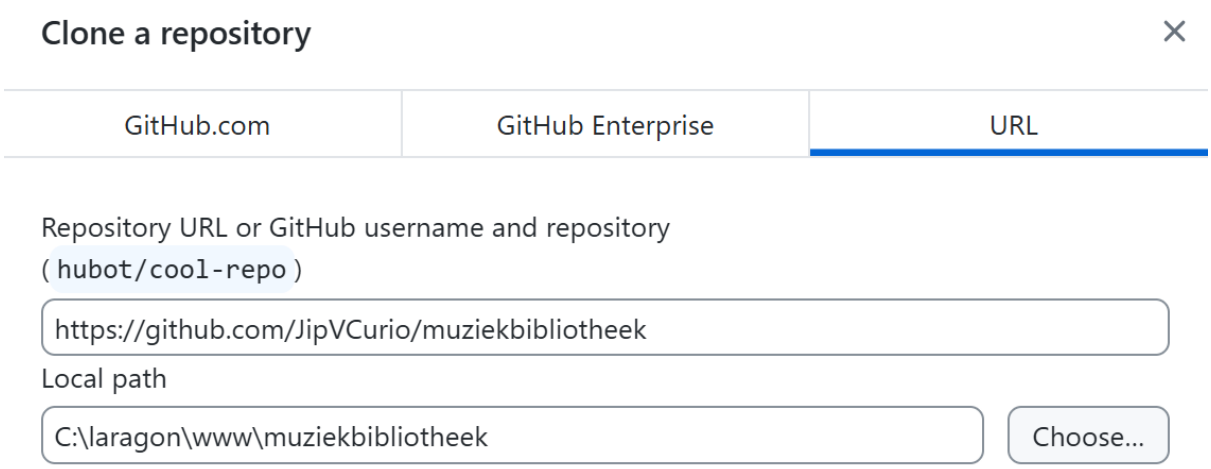
3.2 Controleren of het maken gelukt is

Als alles goed ging, zie je nu een 'lege' repository:



3.3 Je repository clonen naar je laptop

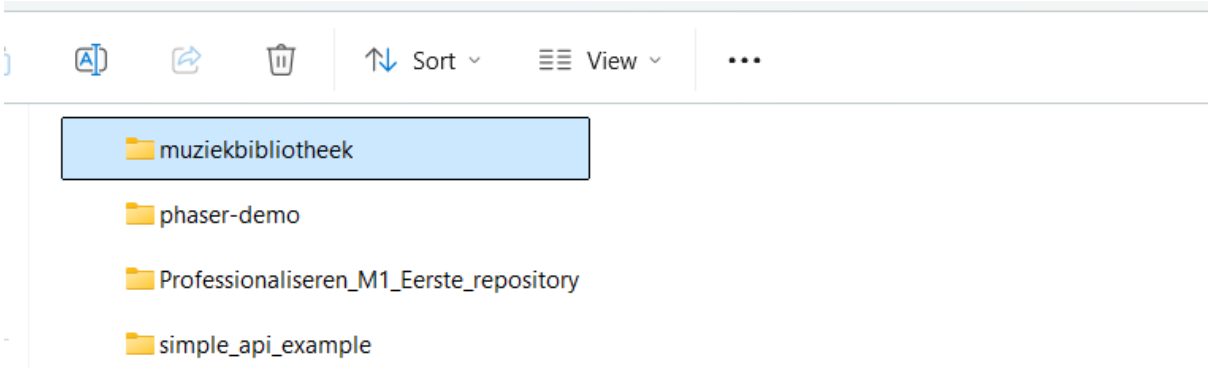
Voordat je code kunt toevoegen, moet je de repository eerst **clonen** naar je laptop. Gebruik daarvoor de tool die je vorige week koos (GitHub Desktop, GitKraken of iets anders).



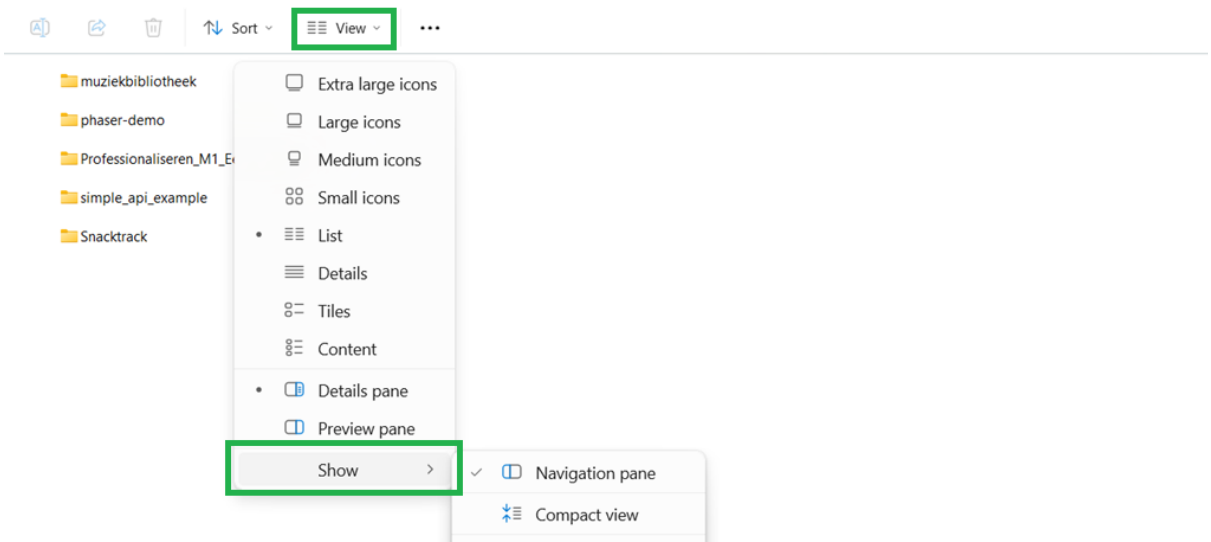
Jij hebt een **andere** repository-URL en local path dan in het voorbeeld. Controleer of de local path verwijst naar de plek waar jij je code wilt opslaan.

3.4 Controleren of het clonen gelukt is

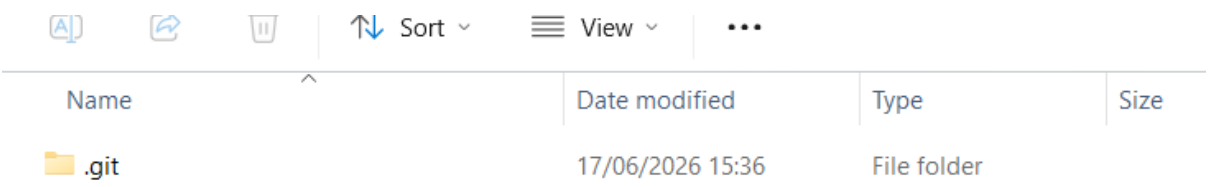
Navigeer in je File Explorer naar de folder waar je naartoe cloonde. Controleer of er een folder `muziekbibliotheek` is.



Open de folder en **zet verborgen bestanden aan** (als dat nog niet aanstaat).



Als het clonen goed ging, zie je nu een verborgen folder met de naam `.git`.



De `.git`-folder bevat informatie over je repository. **Zonder deze folder werkt Git niet.** Verwijder hem niet en pas de bestanden erin niet aan.

3.5 De muziekbibliotheek maken

Gebruik de vaardigheden die je tijdens de webles hebt geleerd om een eenvoudige website na te maken. Vervang `{jouw naam}` en `{titel}` door je eigen naam en 3 muzieknummers waar je graag naar luistert (of de eerste 3 die in je opkomen).

De muziekbibliotheek van {jouw naam}

{muziek titel 1}

{muziek titel 2}

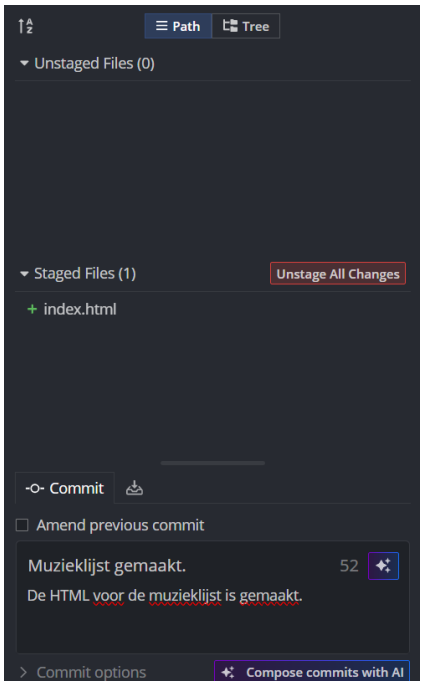
{muziek titel 3}

Sla het bestand op in je `muziekbibliotheek`-folder.

3.6 Je eerste commit maken

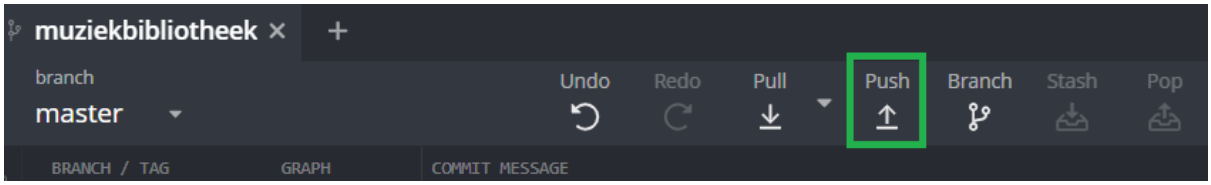
Voordat je je code naar de repository kunt sturen, maak je eerst een **commit**. Een commit is een moment waarop je de code opslaat zoals die op dat moment is.

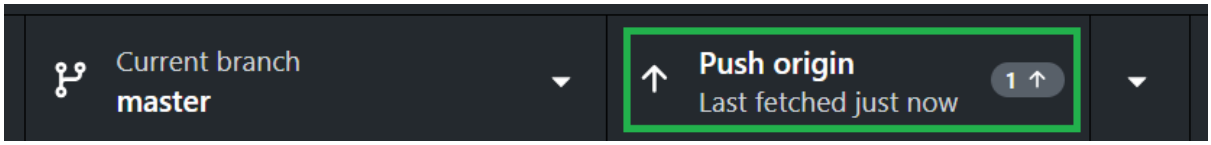
Het is belangrijk dat je in de **titel** van je commit beschrijft wat je gedaan hebt. Zo weten je toekomstige teamgenoten welke aanpassingen jij hebt gemaakt.



3.7 Je commit pushen naar de repository

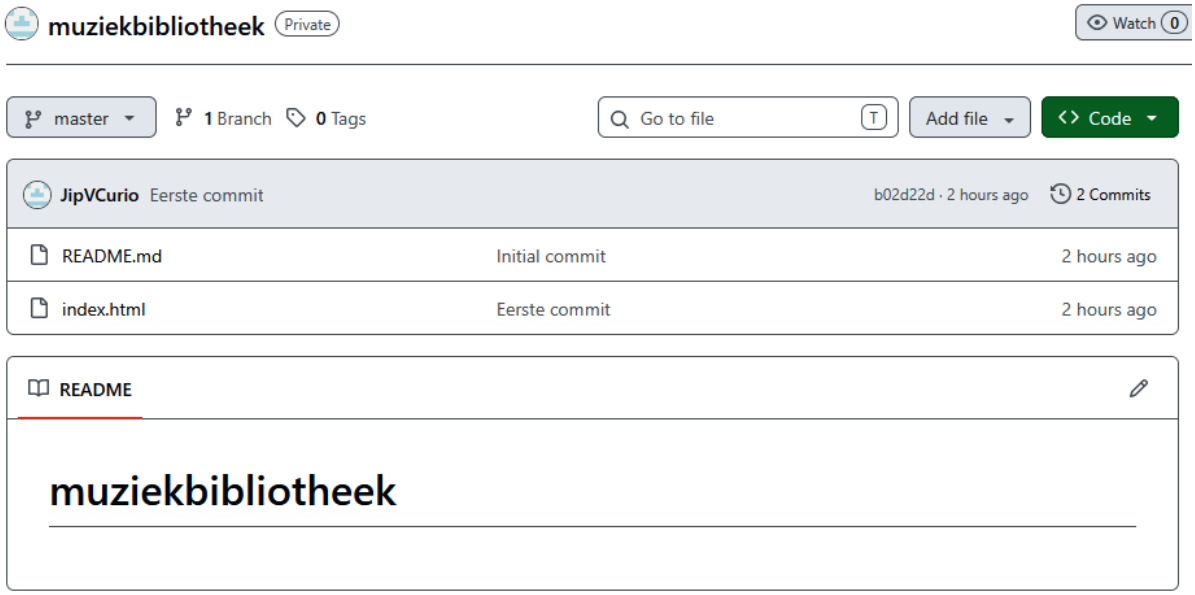
Een commit wordt eerst opgeslagen op je **eigen laptop**. Om de commit (en de code erin) naar je online repository te sturen, gebruik je een **push**.





3.8 Controleren of je commit verstuurd is

Ga op github.com naar de repository die je in stap 3.1 maakte. Als je push gelukt is, zie je nu je code online staan:



Je code staat nu op GitHub! Daardoor kun je op elke laptop verder werken — en kunnen jij en een klasgenoot later samen aan dezelfde code werken.

Zie je je code nog niet op GitHub? Vraag je docent om hulp. Het is voor de rest van je opleiding belangrijk dat GitHub goed werkt.

3.9 De regels van goede commits

Je weet nu hoe je code op een GitHub-repository zet. Onthoud de belangrijkste regel:

- Geef elke commit een **logische, beschrijvende titel** van wat je hebt veranderd

(bijvoorbeeld *"artiesten toegevoegd"*).

- Commit op **logische momenten**: telkens als je iets afgerond hebt.
- Vergeet na het committen niet te **pushen**, anders staat je code nog niet online.

3.10 Nog een wijziging: artiesten toevoegen

Oefen het commit-en-push-proces nog een keer door voor elk nummer ook de **artiest** toe te voegen. Maak daarna een nieuwe commit met een logische titel (bijvoorbeeld *"artiesten toegevoegd"*) en push deze naar je repository.

De muziekbibliotheek van {jouw naam}

{muziek titel 1}

{muziek titel 2}

{muziek titel 3}

De artiesten

{artiest 1}

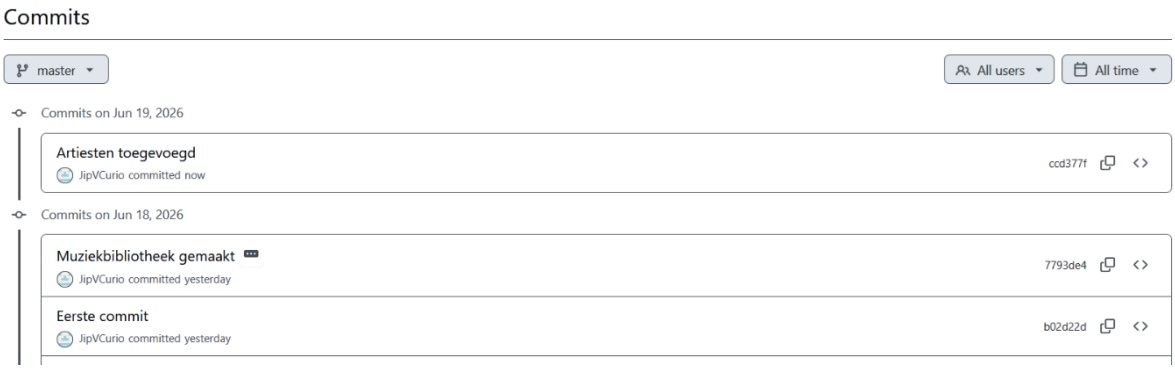
{artiest 2}

{artiest 3}

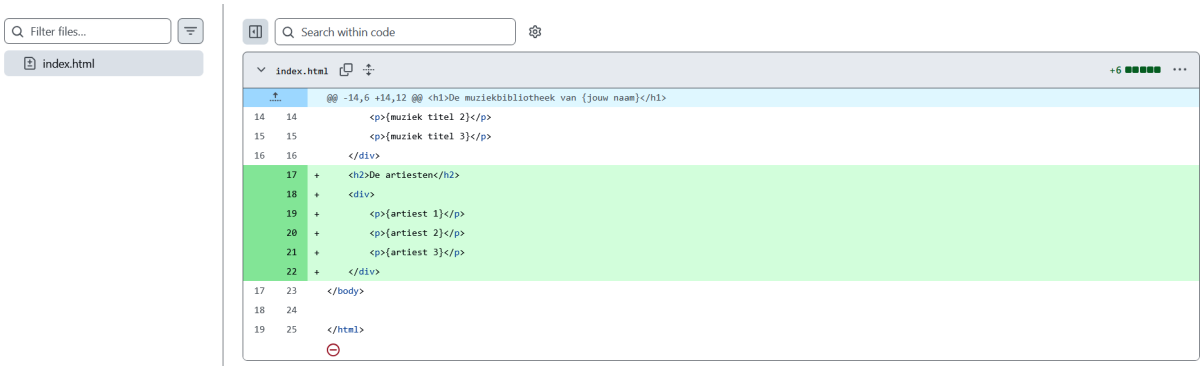
3.11 Je commitgeschiedenis controleren

Een voordeel van Git is dat je precies kunt zien wanneer welke code is gepusht. Klik in je GitHub-repository op **commits**.

Je ziet nu de geschiedenis van je commits.



Klik op je laatste commit. Een **groene** box geeft aan dat code is toegevoegd; een **rode** box dat code is weggehaald.



3.12 Volgende stappen

Vandaag heb je voor het eerst eigen code naar GitHub gestuurd. Die code is nu altijd voor je beschikbaar en je kunt terugkijken welke aanpassingen wanneer zijn gemaakt. In een volgende module leer je hoe je Git gebruikt om écht samen te werken met klasgenoten in dezelfde repository.

Oefeningen

Je eigen repository maken

Je maakt nu zelf een lege repository op GitHub.

Wat ga je doen?

1. Ga naar github.com, log in en ga naar **Repositories**.
2. Klik op **New**.
3. Vul in:
4. Klik op **Create repository**.

Klaar als…

Je ziet een lege `muziekbibliotheek`-repository op GitHub.

Je repository clonen en controleren

Clone je nieuwe repository naar je laptop en controleer of het gelukt is.

Wat ga je doen?

1. Clone de `muziekbibliotheek`-repository met je Git-tool naar een nette locatie in je mappenstructuur.
2. Open de gecloonde folder in je File Explorer.
3. **Zet verborgen bestanden aan** als dat nog niet aanstaat.
4. Controleer of er een verborgen folder `.git` in staat.

Verwijder de `.git`-folder nooit en pas de inhoud ervan niet aan — zonder deze folder werkt Git niet.

Klaar als…

Je ziet de `.git`-folder in je gecloonde `muziekbibliotheek`-folder.

De muziekbibliotheek-website maken

Maak een eenvoudige website voor je muziekbibliotheek.

Wat ga je doen?

1. Gebruik de HTML-vaardigheden uit de webles om een eenvoudige pagina te maken.
2. Vervang `{jouw naam}` door je eigen naam en voeg **3 muzieknummers** toe waar je graag naar luistert.
3. Sla het bestand op **in je `muziekbibliotheek`-folder** (de gecloonde folder).

Klaar als…

Er staat een HTML-bestand met jouw naam en 3 nummers in je `muziekbibliotheek`-folder.

Je eerste commit maken en pushen

Zet je code online met een commit en een push.

Wat ga je doen?

1. Maak in je Git-tool een **commit** van je gemaakte website.
2. Geef de commit een **logische titel**, bijvoorbeeld *"muziekbibliotheek toegevoegd"*.
3. **Push** je commit naar de GitHub-repository.
4. Ga naar je repository op GitHub en controleer of je code er nu staat.

Een commit staat eerst alleen op je laptop. Pas na een **push** staat je code online.

Klaar als…

Je ziet je code op GitHub in de `muziekbibliotheek`-repository.

Je tweede commit — artiesten toevoegen

Je oefent het commit-en-push-proces nog een keer.

Wat ga je doen?

1. Voeg in je website voor **elk nummer ook de artiest** toe.
2. Maak een nieuwe **commit** met een logische titel zoals *"artiesten toegevoegd"*.
3. **Push** je commit naar de GitHub-repository.

Klaar als…

Je website toont per nummer ook de artiest, en de wijziging staat online op GitHub.

Je commitgeschiedenis bekijken

Bekijk de geschiedenis van je commits op GitHub.

Wat ga je doen?

1. Open je `muziekbibliotheek`-repository op GitHub.
2. Klik op **commits**.
3. Bekijk de lijst met commits die je hebt gemaakt.
4. Klik op je laatste commit. Bekijk de **groene** (toegevoegde) en eventueel **rode** (verwijderde) regels.

Klaar als…

Je ziet minstens twee commits met logische titels en kunt per commit zien welke code is veranderd.

Je eigen muziekbibliotheek op GitHub

Inleveropdracht Week 3

Deze week heb je geleerd hoe je zelf code beheert met Git: een repository maken, committen en pushen. Nu breng je dat samen in één eindproduct.

Maak een private repository `muziekbibliotheek`, clone die naar je laptop en bouw een eenvoudige website met jouw naam en 3 nummers (inclusief artiest). Maak minstens twee commits met logische titels en push je code naar GitHub. Lever de repository-URL en je screenshots in via **Itslearning**, waar je docent je werk nakijkt en feedback geeft.

In te leveren

- De URL van je (private) muziekbibliotheek-repository op GitHub
- Een screenshot van je commitgeschiedenis met minstens twee commits
- Een screenshot van je website in de browser

Beoordelingscriteria

• Er is een eigen (private) repository 'muziekbibliotheek' gemaakt en gecloond (2 pt)
• De website toont jouw naam en minstens 3 nummers met artiest (2 pt)
• Er zijn minstens twee commits met logische, beschrijvende titels (3 pt)
• De code is gepusht en staat zichtbaar online op GitHub (3 pt)

Tips

- Commit op logische momenten en geef elke commit een duidelijke titel.
- Vergeet niet te pushen — anders staat je code nog niet online.
- Voeg je docent eventueel toe als collaborator als dat gevraagd wordt voor het nakijken.

Week 4

Samenwerken via GitHub

Leerdoel: je kunt een klasgenoot toevoegen als collaborator en samen code committen, pushen en een merge conflict oplossen

Waar gaat deze week over?

Tot nu toe heb je altijd alleen aan jouw eigen repository gewerkt. Maar software developers werken vrijwel altijd **samen** — meerdere mensen aan dezelfde code. Deze week leer je hoe dat werkt via GitHub.

Aan het einde van deze week weet je wat een collaborator is, kun je iemand toegang geven tot jouw repository, werk je voor het eerst samen aan één gedeelde codebase en weet je hoe je een merge conflict herkent en oplost.

4.1 Wat is een collaborator?

Een **collaborator** is iemand die schrijfrechten heeft op jouw repository. Dat betekent dat die persoon code kan pushen naar jouw repository — net alsof het de zijne of hare is.

Dit is anders dan een **fork**. Bij een fork maakt iemand een eigen kopie van jouw repository. Bij een collaborator werken jullie allebei aan **dezelfde** repository.

- **Fork** — je maakt een eigen kopie; wijzigingen komen niet automatisch terug in het origineel.
- **Collaborator** — je werkt samen in één repository; alle commits zijn direct zichtbaar voor iedereen.

4.2 Een collaborator uitnodigen

Je voegt een collaborator toe via de instellingen van je repository op GitHub.

1. Open je repository op github.com.
2. Klik op **Settings** (rechtsboven in de repository).
3. Ga in het linkermenu naar **Collaborators**.
4. Klik op **Add people**.
5. Zoek op de GitHub-gebruikersnaam of het e-mailadres van je klasgenoot.
6. Klik op **Add [naam] to this repository**.

Je klasgenoot ontvangt nu een uitnodiging — via e-mail én als melding op GitHub.

4.3 De uitnodiging accepteren

De uitgenodigde persoon moet de uitnodiging eerst accepteren voordat hij of zij toegang heeft.

1. Open de e-mail van GitHub met het onderwerp *"[naam] has invited you to collaborate"*.
2. Klik op **View invitation** en daarna op **Accept invitation**.

Of via GitHub zelf: klik op het belletje (notificaties) rechtsboven en zoek de uitnodiging.

Na het accepteren kan de collaborator de repository clonen en code pushen.

Let op: clone de repository van je klasgenoot naar een **nieuwe, aparte folder** op je laptop. Zet hem niet in een bestaande Git-repository.

4.4 De samenwerkworkflow

Zodra jullie allebei toegang hebben tot de repository, is er één gouden vuistregel:

Pull !’ aanpassen !’ commit !’ push

Begin elke werksessie met een **pull**. Zo haal je de nieuwste versie van de code op en voorkom je problemen.

Waarom is dit zo belangrijk? Als jouw klasgenoot een commit heeft gepusht terwijl jij aan het werken was, wijkt jouw lokale versie af van de versie op GitHub. Als je dan probeert te pushen, weigert GitHub dat — jij bent namelijk *achter* op de online versie.

4.5 Wijzigingen ophalen met een pull

Een **pull** haalt de nieuwste commits van GitHub naar je laptop. Je doet dit in je Git-tool.

Klik op de **Pull**-knop in de werkbalk bovenaan GitKraken.

Klik op **Fetch origin** (of **Pull origin** als er nieuwe commits zijn) in de werkbalk bovenaan GitHub Desktop.

Na een succesvolle pull zie je de commits van je klasgenoot in je commitgeschiedenis staan.

4.6 Wat is een merge conflict?

Soms lukt een pull niet zomaar. Dat gebeurt als jullie allebei **dezelfde regel** in **hetzelfde bestand** hebben aangepast. Git weet dan niet welke versie hij moet bewaren — dit heet een **merge conflict**.

Git markeert het conflict direct in het bestand:

```
<<<<<<< HEAD
<title>Muziekbibliotheek van Anna</title>
=======
<title>Muziekbibliotheek van Bo</title>
>>>>>>> origin/main
```

- Het stuk tussen `<<<<<<< HEAD` en `=======` is **jouw versie** (lokaal).
- Het stuk tussen `=======` en `>>>>>>>` is **de versie van GitHub** (van je klasgenoot).

Een merge conflict is geen fout — het is Git die om jouw hulp vraagt. Je lost het op door zelf te kiezen welke versie je wilt bewaren.

4.7 Een merge conflict oplossen

Het oplossen van een merge conflict gaat in vijf stappen.

1. **Open het bestand** met het conflict in je code-editor (bijv. VS Code).
2. **Zoek de conflictmarkers** (`<<<<<<<`, `=======`, `>>>>>>>`).
3. **Kies welke versie je wilt houden** — of combineer ze als dat logisch is.
4. **Verwijder alle markers** (de regels met `<<<<<<<`, `=======` en `>>>>>>>`).
5. **Sla het bestand op**, maak een nieuwe **commit** en **push**.

GitKraken markeert bestanden met een conflict met een oranje uitroepteken. Klik op het bestand om het te openen in de ingebouwde conflictoplosser.

GitHub Desktop toont een melding dat er conflicten zijn. Klik op **Open in Visual Studio Code** (of je eigen editor) om het conflict handmatig op te lossen.

Na het oplossen commit je het resultaat met een logische titel zoals *"merge conflict opgelost"* en push je naar GitHub.

4.8 Onthoud voor nu het volgende

- **Voeg je klasgenoot toe als collaborator** via Settings !’ Collaborators.
- **Pull altijd eerst** voordat je begint met werken.
- **Bij een merge conflict**: open het bestand, kies de juiste versie, verwijder de markers en commit + push.

Volgende week leer je meer over branches — een techniek om nog veiliger samen te werken zonder elkaars werk te verstoren.

Oefeningen

Je klasgenoot uitnodigen als collaborator

Zoek een klasgenoot om mee samen te werken. Jullie gaan dit en de volgende oefeningen samen uitvoeren.

Werk je in een **drietal**? Dan nodigt de derde persoon één van jullie uit voor zijn of haar repository, en wordt zelf bij één van jullie uitgenodigd.

Wat ga je doen?

1. Spreek met je klasgenoot af wie wie uitnodigt (of doe het allebei over en weer).
2. Open jouw `muziekbibliotheek`-repository op github.com.
3. Ga naar **Settings !' Collaborators !' Add people**
4. Zoek de GitHub-gebruikersnaam van je klasgenoot.
5. Stuur de uitnodiging.

Klaar als…

Je klasgenoot heeft de uitnodiging ontvangen (verschijnt als *Pending* in je collaborators-lijst).

De uitnodiging accepteren en repository clonen

Accepteer de uitnodiging van je klasgenoot en haal de code naar je laptop.

Wat ga je doen?

1. Open de uitnodigings-e-mail van GitHub (*"[naam] has invited you to collaborate"*) en klik op **View invitation**, of open GitHub en klik op het belletje rechtsboven.
2. Klik op **Accept invitation**.
3. Clone de repository van je klasgenoot naar je laptop — in een **nieuwe, aparte folder** (niet in een bestaande repository).

Geef de folder een duidelijke naam, zoals `muziekbibliotheek-[naam-klasgenoot]`, zodat je hem niet verwart met je eigen repository.

Klaar als…

De code van je klasgenoot staat op jouw laptop en je ziet zijn of haar bestanden in de folder.

Jij voegt een nummer toe aan de gedeelde repository

Je werkt nu als collaborator in de repository van je klasgenoot. Voeg een nummer toe aan zijn of haar muziekbibliotheek.

Wat ga je doen?

1. Open de repository van je klasgenoot in je code-editor.
2. Doe eerst een **pull** om zeker te zijn dat je de nieuwste versie hebt.
3. Voeg in `index.html` een nieuw nummer toe aan de lijst — kies zelf een nummer en artiest.
4. Maak een **commit** met een logische titel, bijvoorbeeld *"nummer toegevoegd door [jouw naam]"*.

5. **Push** je commit naar GitHub.

Klaar als…

Je klasgenoot kan op GitHub jouw commit zien staan in de commitgeschiedenis van zijn of haar repository.

Wijzigingen ophalen en jij voegt ook iets toe

Nu draai je het om: je klasgenoot heeft iets toegevoegd aan jouw repository. Haal die wijzigingen op en voeg zelf ook nog iets toe.

Wat ga je doen?

1. Open **jouw eigen** `muziekbibliotheek`-repository in je Git-tool.
2. Doe een **pull** — je ziet nu de commit van je klasgenoot binnenkomen.
3. Open `index.html` in je editor en voeg zelf nog een nummer toe.
4. Maak een **commit** met een logische titel, bijvoorbeeld *"extra nummer toegevoegd"*.
5. **Push** je commit naar GitHub.

Klaar als…

De commitgeschiedenis van jouw repository toont commits van jou én van je klasgenoot.

Een merge conflict veroorzaken en oplossen

Jullie gaan bewust een merge conflict veroorzaken, zodat je leert hoe het voelt en hoe je het oplost.

Voer deze stappen **in volgorde** uit. Wacht op het teken van je klasgenoot voordat je verdergaat.

Wat ga je doen?

Stap 1 — Beide studenten tegelijk (niet pullen!)

Spreek af wie **student A** (eigenaar van de repository) en wie **student B** (de collaborator) is. Jullie werken voor deze oefening allebei in de repository van student A.

Student B: zorg dat je de repository van student A al gecloond hebt naar je laptop (zie oefening 3). Heb je dat nog niet gedaan, doe dat dan eerst.

Beide studenten openen nu **tegelijk** het bestand `index.html` van de repository van student A in hun eigen editor — zonder eerst te pullen. Zoek allebei de titeltag bovenaan in het bestand. Het is belangrijk dat jullie **dezelfde regel** aanpassen, maar met een **andere tekst**, zodat Git straks niet weet welke versie hij moet bewaren:

- **Student A** verandert de titel naar: `<title>Muziekbibliotheek van [naam A]</title>`
- **Student B** verandert de titel naar: `<title>Muziekbibliotheek van [naam B]</title>`

Sla allebei het bestand op en ga dan pas verder met stap 2. **Pull niet tussendoor.**

Stap 2 — Student A pusht als eerste

Student A maakt een commit (*"titel aangepast"*) en pusht. Dit lukt.

Stap 3 — Student B probeert te pushen

Student B maakt ook een commit en probeert te pushen. Dit mislukt — GitHub weigert omdat student B achterloopt op de online versie.

Stap 4 — Student B pullt en lost het conflict op

Student B doet een pull. Git kan de twee versies niet automatisch samenvoegen en meldt een merge conflict. Open `index.html` in je editor. Je ziet nu de conflictmarkers in het bestand staan. Alles tussen `HEAD` en `=======` is jouw lokale versie; alles tussen `=======` en `origin/main` is de versie van student A die al online stond.

Bespreek met je klasgenoot welke titel jullie willen houden (of maak er een combinatie van). Verwijder daarna alle drie de markerregels en de versie die je niet wilt houden, zodat er alleen een gewone, correcte titeltag overblijft. Sla het bestand op.

Stap 5 — Student B commit en pusht de oplossing

1. Maak een commit met de titel *"merge conflict opgelost"*.
2. Push naar GitHub.

Klaar als…

Het conflict is opgelost, de commit staat online en beide studenten zien in de commitgeschiedenis dat er een merge conflict is opgelost.

Commitgeschiedenis van de samenwerking bekijken

Bekijk de commitgeschiedenis van de gedeelde repository en controleer of de samenwerking zichtbaar is.

Wat ga je doen?

1. Open de `muziekbibliotheek`-repository op GitHub.
2. Klik op **commits**.
3. Scroll door de geschiedenis en controleer:

Klaar als…

Je ziet minstens vier commits, van twee verschillende GitHub-gebruikers, waaronder de commit van het opgeloste merge conflict.

Samenwerken aan de muziekbibliotheek

Inleveropdracht Week 4

Deze week heb je geleerd hoe je samenwerkt aan één repository. Je weet hoe je een collaborator uitnodigt, hoe de samenwerkworkflow werkt en hoe je een merge conflict oplost.

Werk samen met je klasgenoot aan de `muziekbibliotheek`-repository. Voeg elkaar toe als collaborator, laat allebei minstens één commit zien en los gezamenlijk een merge conflict op. Lever je screenshots en de repository-URL in via **Itslearning**.

In te leveren

- De URL van de gedeelde muziekbibliotheek-repository op GitHub
- Een screenshot van de collaborators-pagina (Settings !’ Collaborators) waarop beide namen zichtbaar zijn
- Een screenshot van de commitgeschiedenis waarop commits van twee verschillende gebruikers zichtbaar zijn
- Een screenshot van de commit "merge conflict opgelost" (of de commit details waaruit de oplossing blijkt)

Beoordelingscriteria

• Een klasgenoot is correct uitgenodigd en heeft de uitnodiging geaccepteerd (2 pt)
• Beide studenten hebben minstens één commit gemaakt met een logische, beschrijvende titel (3 pt)
• Er is een merge conflict opgetreden en correct opgelost (commit "merge conflict opgelost" zichtbaar) (3 pt)
• De definitieve code staat gepusht en is zichtbaar op GitHub (2 pt)

Tips

- Spreek vooraf af wie eigenaar is van de repository die jullie voor de inleveropdracht gebruiken.
- Vergeet niet te pullen voordat je begint met werken — zo voorkom je onnodige merge conflicts.
- Bij het oplossen van een merge conflict: verwijder altijd alle drie de markerregels (<<<<<<<, =======, >>>>>>>).