

# Les 4 - Versiebeheer

[« Vorige les](#)[» Volgende les](#)

## Huiswerk

Programmeren voor Beginners - Les 4

Resultaat: 10,0

[Bekijk huiswerk](#)

Je hebt 1 vraagmoment. 1 moment over.

[Ga naar stel een vraag](#)

## Inhoudsopgave

- 4.1 Inleiding
- 4.2 Wat is versiebeheer?
- 4.3 Werken met Git
- 4.4 Werken met GitHub
- 4.5 Versiebeheer in de praktijk
- 4.6 Samenvatting
- 4.7 Vraagmoment
- 4.8 Oefenopgaven
- 4.9 Huiswerkopgaven

### 4.1 Inleiding

In de vorige les gaven we aan dat u vanaf nu kunt beginnen met het echte werk: het onder de knie krijgen van een programmeertaal. Toch is er nog een onderwerp dat we voor die tijd willen behandelen: versiebeheer. Door uw code op te slaan in 'versies' kunt u fouten gemakkelijker terugdraaien. Ook als u uw software ontwikkelt in een richting waar u later spijt van krijgt, kunt u teruggrijpen naar een eerdere versie.



Ook voor programmeurs is versiebeheer van groot belang.

## Leerdoelen

Aan het einde van deze les weet u:

- wat versiebeheer is;
- wat een versiebeheersysteem (VCS) is;
- wat Git is en hoe u het installeert;
- wat een *repository* is;
- hoe u uw werk kunt terugbrengen in een eerdere staat;
- hoe u werkt met GitHub;
- welke werkwijze een programmeur hanteert.



Enkele afbeeldingen in deze les zijn moeilijk leesbaar in de printversie van deze les. Wilt u een afbeelding uitvergroten? Bekijk dan de digitale versie van de les op Plaza. Klik op de afbeeldingen om ze te vergroten.

## 4.2 Wat is versiebeheer?

Het is u waarschijnlijk weleens overkomen: per ongeluk werken in een oude versie, de verkeerde versies doorsturen of – erger nog – bestanden kwijtraken. Dit kunt u voorkomen met **versiebeheer**. Dit is het bijhouden van de verschillende fasen van een bestand: van aanmaak en aanvulling tot aan de definitieve versie.

Bij versiebeheer stelt u documenten op één centrale plek digitaal beschikbaar. Ook houdt u alle wijzigingen bij, waardoor u op een later moment altijd een bepaalde versie kunt inzien.

Versiebeheer is vooral prettig als u met meerdere mensen aan één bestand werkt. Goed versiebeheer heeft namelijk grote voordelen:

- Alle bestanden zijn altijd beschikbaar.
- Informatie wordt sneller gevonden.
- Voor iedereen is duidelijk welk bestand het meest recent.
- Werken in oude of verkeerde versies is verleden tijd.
- U kunt zien wie welke wijzigingen wanneer heeft gedaan. Als u projectbeheerder bent, kunt u wijzigingen ook goed- of afkeuren.
- Er is minder opslagruimte nodig. (Bestanden staan nog maar op één plek.)
- Informatie kan gemakkelijker gedeeld worden.

- U kunt gemakkelijk terugrijpen naar een eerdere versie, als de huidige versie niet voldoet.

Werkt u met Microsoft Word? Dan heeft u al met versiebeheer gewerkt, bewust of onbewust:

- U werkte bewust met versiebeheer door meerdere versies van bestanden op te slaan en/of te werken met de functie "Wijzigingen bijhouden".
- Op de achtergrond maakte het programma automatische back-ups van uw bestand, waardoor u altijd kon terugrijpen naar een eerdere versie. Onbewust was er dus ook sprake van versiebeheer.

Als programmeur zou u natuurlijk ook kunnen werken met meerdere versies van hetzelfde bestand. Programmeurs hebben echter software ontwikkeld om versiebeheer gemakkelijker en overzichtelijker te houden. Zo'n programma noemen we ook wel een **versiebeheersysteem**, of kortweg VCS (*Version Control System*).

Het grote verschil tussen deze software en het versiebeheer door *Microsoft Word*, is dat *Microsoft Word* alleen de veranderingen van een specifiek document bijhoudt, terwijl de software voor programmeurs alle veranderingen bijhoudt binnen een map op de harde schijf (plus alle submappen). Ook als u bestanden en/of mappen toevoegt of verwijdert, wordt dat dus geregistreerd en kan de handeling ongedaan worden gemaakt.

In de programmeerwereld zijn verschillende populaire versiebeheersystemen verkrijgbaar, zoals *Bitbucket*, *Mercurial*, *Perforce* en *SVN*. Het populairste systeem is echter *Git*. Op dit systeem gaan we in de volgende paragraaf dieper in.

## 4.3 Werken met Git

*Git* is een opensourceapplicatie die werd ontwikkeld door informaticus Linus Torvalds, vooral bekend als de 'vader' van het besturingssysteem *Linux* (na Windows en Mac het populairste besturingssysteem voor computers).

De makers van *Linux* werkten lange tijd met het versiebeheersysteem *BitKeeper*. Deze software was lange tijd gratis beschikbaar, totdat *BitMover* (het bedrijf achter *BitKeeper*) besloot om het programma niet langer voor vrij gebruik aan te bieden. Torvalds en zijn medewerkers besloten daarom om een eigen VCS te ontwikkelen: *Git*.



*Git* werd ontwikkeld door de ontwikkelaars van *Linux*.

Torvalds stelde diverse eisen aan de nieuwe VCS. Zo moest het programma onder meer snel zijn, maar ook simpel te gebruiken. Daarnaast moest het een **gedistribueerd versiebeheersysteem** worden. Dit betekent dat de volledige opslag voor iedereen afzonderlijk te downloaden moet zijn. In de opslag worden alle versies van het bestand bewaard, in plaats van alleen de laatste versie. Wanneer een server niet bereikbaar is, kan de opslag van iedere persoon worden opgehaald en hersteld.



Naast gedistribueerd versiebeheersysteem kennen we centrale versiebeheersystemen, die bestaan uit één centrale server die extern is opgeslagen. Hier worden alle versies van bestanden opgeslagen en kunnen meerdere mensen met hun eigen computer bestanden vanaf downloaden. Een nadeel hiervan is dat, als deze server niet bereikbaar is, niemand meer bij de bestanden kan. Wanneer er geen back-up is gemaakt, is ook echt alles weg.

De hiervoor genoemde 'volledige opslag' noemen we ook wel een **repository** (of kortweg 'repo'). Deze map bevat alle bestanden van een project. In de repository wordt ook de revisiegeschiedenis van elk bestand opgeslagen. U kunt een repository individueel beheren, maar u kunt hem ook delen met anderen.

In deze paragraaf leggen we u uit wat u moet doen om het versiebeheersysteem *Git* te kunnen gebruiken.

## 4.3.1 Git installeren

Allereerst moet u *Git* installeren op uw computer. Afhankelijk van uw besturingssysteem vindt u hier voor instructies in paragraaf 4.3.1.1 (voor Windows) of 4.3.1.2 (voor MacOS).

### 4.3.1.1 Windows

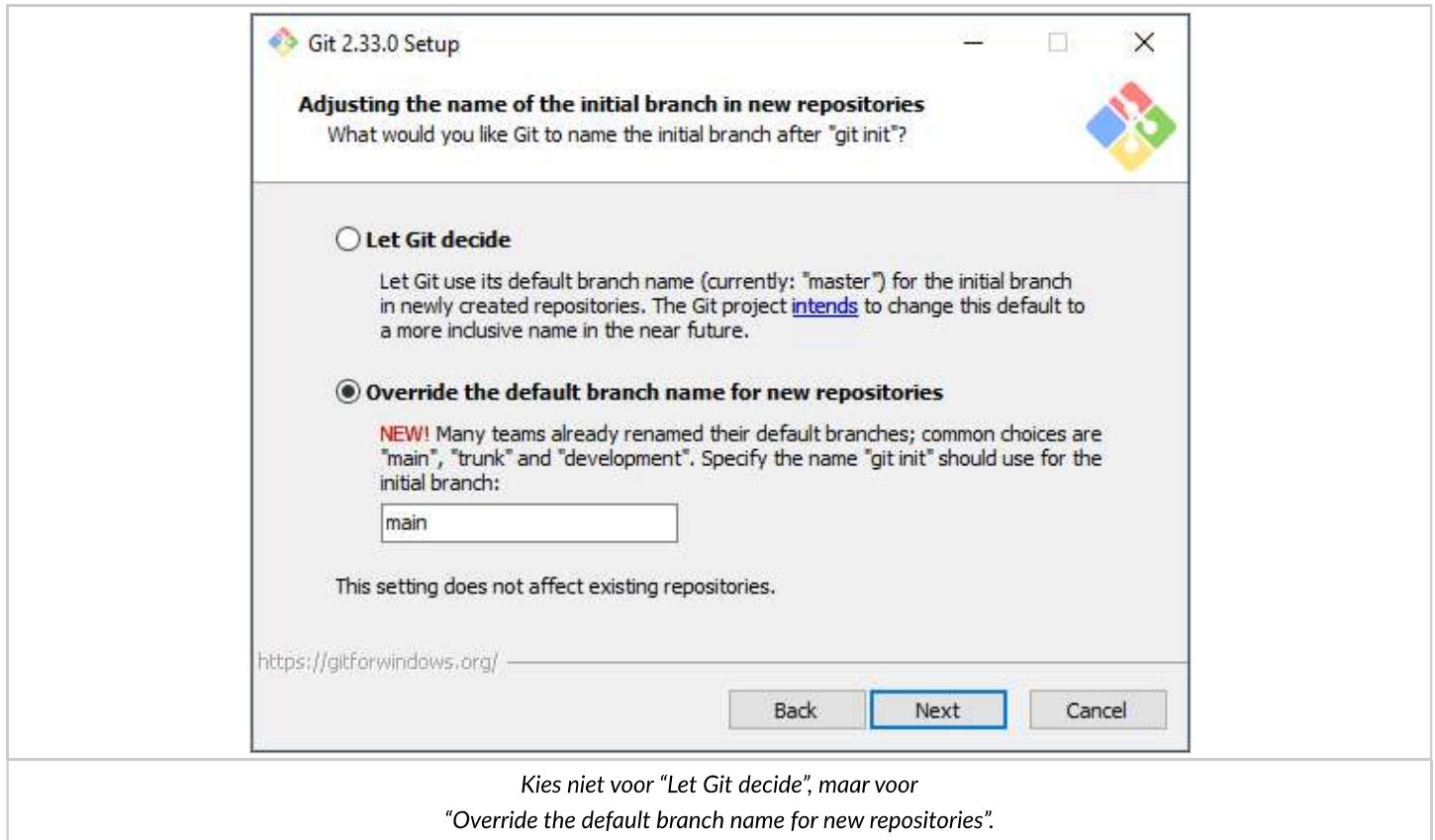
Wilt u *Git* installeren op een Windows-computer, dan leest u in deze subparagraaf hoe u te werk moet gaan.

- Ga naar [www.git-scm.com/downloads](http://www.git-scm.com/downloads).
- Klik op "Windows".



The screenshot shows the official Git website at [www.git-scm.com](http://www.git-scm.com). The main navigation menu includes links for About, Documentation, Downloads (which is highlighted in red), GUI Clients, Logos, and Community. On the right side, there's a search bar and a large image of a computer monitor displaying a teal screen. The central content area is titled "Downloads" and features four download links: "macOS", "Windows" (which is also highlighted in red), "Linux/Unix", and "Older releases are available and the Git source repository is on GitHub.". Below these are sections for "GUI Clients" (with a link to "View GUI Clients") and "Logos" (with a link to "View Logos"). At the bottom left, there's a link to "About this site" and a note about patches and comments. At the bottom right, it says "Git is a member of Software Freedom Conservancy". A callout box at the bottom of the page says "Klik op 'Windows'".

- Wacht tot het downloaden is voltooid.
- Open het installatiebestand.
- Doorloop de installatiewizard door in ieder venster op "Next" te klikken, **behalve het venster "Adjusting the name of the initial branch in new repositories"**. Hier kiest u niet voor "Let Git decide", maar voor de andere optie.



Kies niet voor "Let Git decide", maar voor  
"Override the default branch name for new repositories".

- Klik op "Install" om de installatie te voltooien.
- Haal het vinkje voor "Release View Notes" weg.
- Klik op "Finish".
- Start uw computer opnieuw op.



Het kan zijn dat het installatieproces van toekomstige versies van Git iets afwijkt van deze beschrijving. Krijgt u schermen te zien die niet in deze uitleg of in de video worden getoond en twijfelt u hierdoor over de te nemen stappen? Maak dan gebruik van het vraagmoment. Uw docent helpt u graag.

Na de herstart kunt u op de volgende manier controleren of Git op de juiste manier is geïnstalleerd:

- Open VS Code.
- Klik met de muis in de *Terminal*, achter uw huidige positie op de harde schijf (bijvoorbeeld C:\Users\MSI of C:\Users\Alex\_).



Ziet u de terminal niet meer? Klik dan in de *Menu Bar* achtereenvolgens op "Terminal" en "New Terminal". Ziet u uw positie op de harde schijf niet verschijnen? Blader dan terug naar de vorige les om de voorkeursconsole in te stellen.

- Typ git --version en druk op Enter.

Als u Git goed geïnstalleerd heeft, ziet u de tekst Git [versienummer] in het venster verschijnen:

Microsoft Windows [Version 10.0.19043.1165]  
(c) Microsoft Corporation. Alle rechten voorbehouden.

```
C:\Users\MSI\Documents\NHA-prg>git --version
git version 2.33.0.windows.1
```

```
C:\Users\MSI\Documents\NHA-prg>
```



Ga naar Plaza en bekijk een video over deze uitleg.

### 4.3.1.2 MacOS

Wilt u Git op een Mac-computer installeren, dan leest u in deze subparagraaf hoe u te werk gaat.

- Ga naar [www.git-scm.com/downloads](https://www.git-scm.com/downloads).
- Klik op "macOS".

The screenshot shows the official Git website ([git-scm.com/](https://git-scm.com/)). The main navigation bar includes links for About, Documentation, Downloads (which is currently selected), and Community. On the right side, there's a search bar and a large image of a computer monitor displaying a teal screen. The 'Downloads' section features three buttons: 'macOS' (highlighted with a red box), 'Windows', and 'Linux/Unix'. Below these buttons, text indicates that older releases and source code are available on GitHub. The 'macOS' section contains links for 'GUI Clients' (with a note about built-in tools like git-gui and gitk) and 'Logos' (with a note about various formats available). The 'Git via Git' section provides instructions for cloning the repository and browsing the web interface.

*Klik op "macOS".*

- Download de *Binary Installer*.

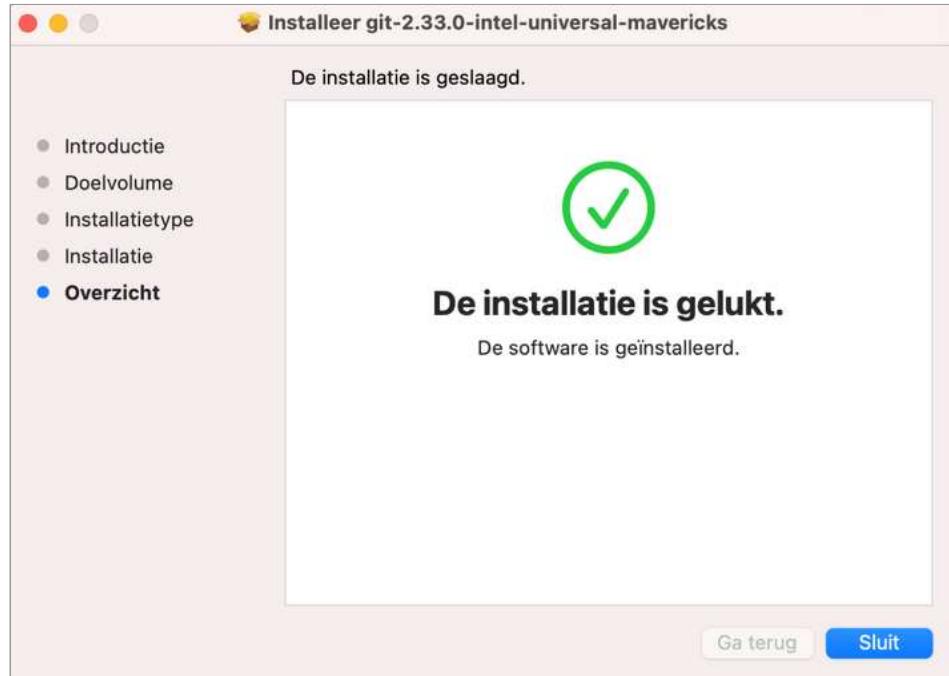
Wacht tot het downloaden is voltooid.

- Dubbelklik op het installatiebestand om de toepassing te installeren.
- Doorloop de installatiewizard. U mag akkoord gaan met de standaardinstellingen.



Het kan zijn dat het installatieproces van toekomstige versies van Git iets afwijkt van deze beschrijving. Krijgt u schermen te zien die niet in deze uitleg of in de video worden getoond en twijfelt u hierdoor over de te nemen stappen? Maak dan gebruik van het vraagmoment. Uw docent helpt u graag.

U krijgt een melding als de installatie is gelukt:



- Klik op "Sluit" om de wizard te sluiten.
- Start uw computer opnieuw op.

Na de herstart kunt u op de volgende manier controleren of Git op de juiste manier is geïnstalleerd:

- Open VS Code.
- Open de Terminal.
- Typ `git --version` en druk op Enter.

Als u Git goed heeft geïnstalleerd, ziet u de tekst `Git [versienummer]` in het venster verschijnen:

The screenshot shows the VS Code interface with the 'TERMINAL' tab selected. The terminal window displays the following text:  
PROBLEMS    OUTPUT    DEBUG CONSOLE    TERMINAL  
  
The default interactive shell is now zsh.  
To update your account to use zsh, please run `chsh -s /bin/zsh`.  
For more details, please visit <https://support.apple.com/kb/HT208050>.  
MBP-van-admin:~ admin\$ git --version  
git version 2.33.0  
MBP-van-admin:~ admin\$ █

#### 4.3.2 Map aanmaken en openen

Na de installatie van Git moet u iets doen wat u in de vorige les al leerde: een map aanmaken en openen in VS Code.

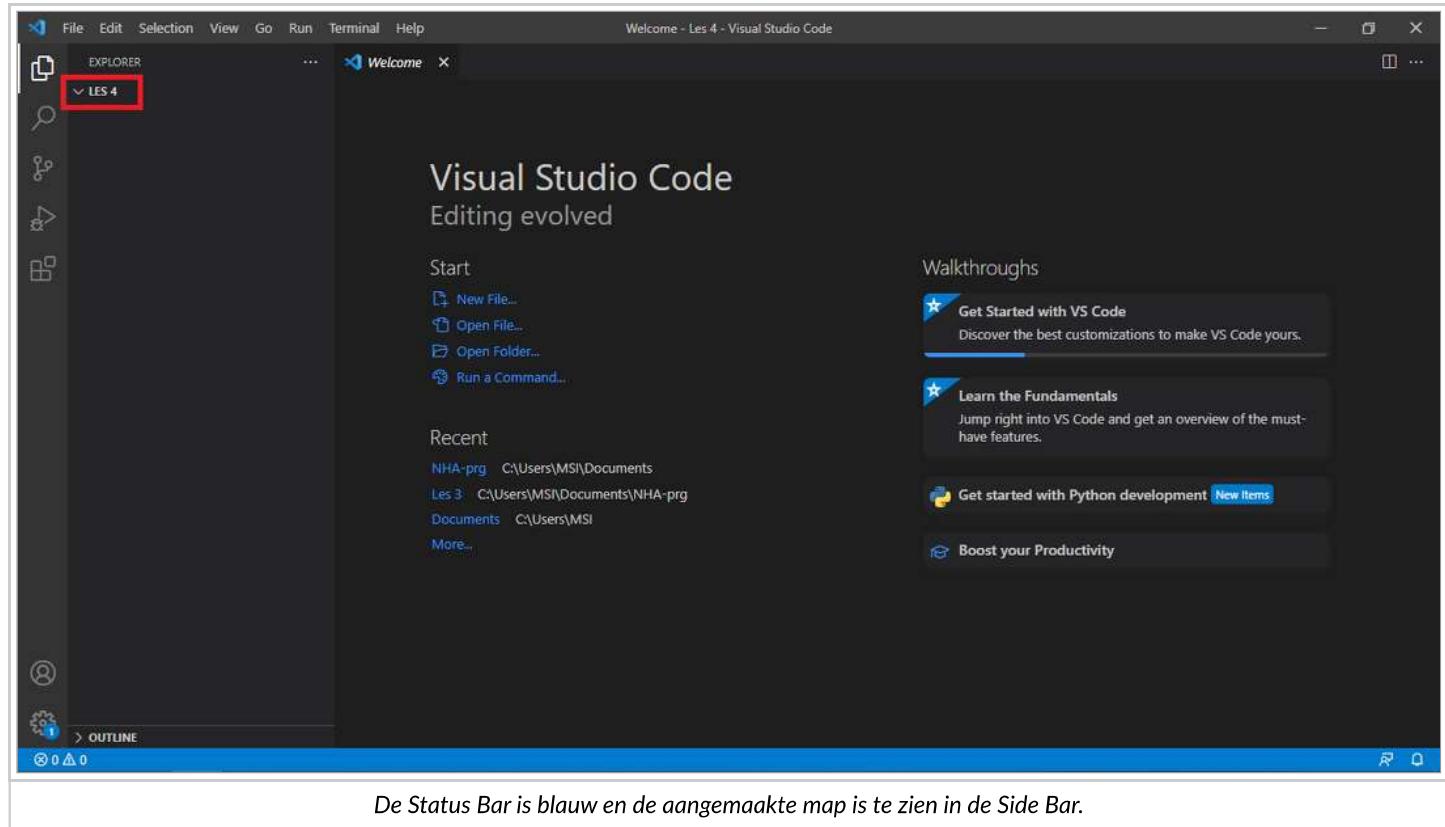
- Open VS Code.
- Open een map die u eerder aanmaakte, bijvoorbeeld NHA-prg).



Weet u het nog? Een map openen doet u in de *Menu Bar* op "File" te klikken, vervolgens te klikken op "Open Folder" en tot slot de map te selecteren.

- Klik op de knop "New Folder".
- Geef de map een naam (bijvoorbeeld Les 4).
- Klik in de *Menu Bar* op "File".
- Klik op "Open Folder".
- Selecteer de map die u zojuist aanmaakte.

Doordat u een map heeft geopend, is de *Status Bar* onder in beeld blauw. Daarnaast ziet u in de *Explorer* (in de *Side Bar*) de map 'Les 4' staan.



### 4.3.3 Map initialiseren

De volgende stap is het initialiseren van de map. 'Initialiseren' houdt in dat u een opslagmedium gaat voorbereiden voor ontvangst van gegevens. In ons geval gaan we een map (met submappen) voorbereiden voor het gebruik van *Git*.

- Klik in de *Menu Bar* op "Terminal".
- Klik op "New Terminal".
- Typ git init en druk op Enter.

De *Terminal* zal nu ongeveer het volgende bericht geven:

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 4>git init
Initialized empty Git repository in C:\Users\MSI\Documents\NHA-prg\Les 4/.git/

C:\Users\MSI\Documents\NHA-prg\Les 4>
```

Het exacte bericht is afhankelijk van uw positie op de harde schijf. In ieder geval zal uw bericht de tekst `Initialized empty Git repository` bevatten. Dit betekent dat de map die u eerder aanmaakte, is veranderd in een *repository*. Daardoor zullen alle wijzigingen die u vanaf nu doorvoert, bewaard blijven. (En kunnen ze weer ongedaan worden gemaakt, indien u dat wenst.)

#### 4.3.4 Uzelf kenbaar maken als gebruiker

U moet uzelf ook nog kenbaar maken als gebruiker aan het Git-systeem.

Dit is eenmalig en hoeft u bij het initialiseren van toekomstige mappen niet meer te doen. Zou u deze stap overslaan, dan zal *Git* straks niet werken.

- Ga naar een nieuwe regel in de *Terminal*.
- Typ `git config --global user.email "voorbeeld@mail.com"` (de tekst in rood vervangt u uiteraard door uw eigen e-mailadres).
- Druk op Enter.

U zult, in tegenstelling tot eerdere opdrachten, geen boodschap terugkrijgen.

- Typ `git config --global user.name "naam"` (de tekst in rood vervangt u uiteraard door uw eigen voor- en achternaam).
- Druk op Enter.

Ook nu zult u geen boodschap terugkrijgen.

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE
Microsoft Windows [Version 10.0.19043.1237]
(c) Microsoft Corporation. Alle rechten voorbehouden.

C:\Users\MSI\Documents\NHA-prg\Les 4>git init
Initialized empty Git repository in C:\Users\MSI\Documents\NHA-prg\Les 4/.git/

C:\Users\MSI\Documents\NHA-prg\Les 4>git config --global user.email "        @nha.nl"
C:\Users\MSI\Documents\NHA-prg\Les 4>git config --global user.name "Test NHA"

C:\Users\MSI\Documents\NHA-prg\Les 4>
```

*Maak uzelf kenbaar als gebruiker.*

#### 4.3.5 Bestand aanmaken

De volgende stap is het aanmaken van een bestand.

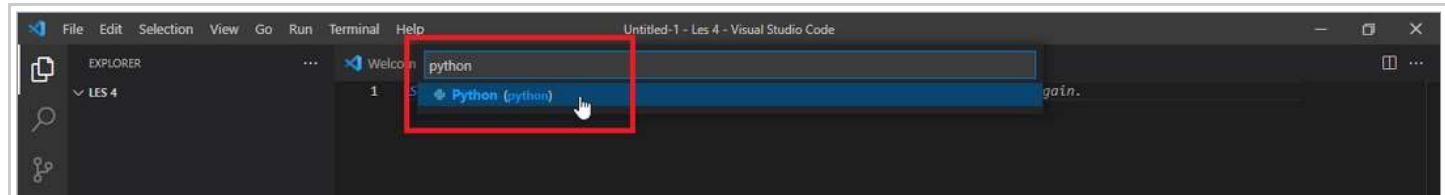
- Klik in de Menu Bar op “File”.
- Klik op “New File”.
- Klik in het document dat verschijnt, op “Select a language”.

```
File Edit Selection View Go Run Terminal Help
Untitled-1 - Les 4 - Visual Studio Code
EXPLORER ... Welcome E Untitled-1 X
1 |Select a language| to get started. Start typing to dismiss, or don't show this again.

Klik op "Select a language".
```

- Typ ‘Python’.

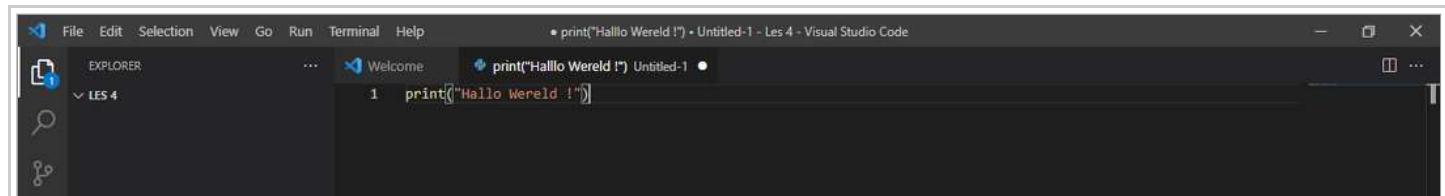
- Klik op “Python (python)”.  
◦



## Selecteer de taal Python.

- Zet de cursor in de eerste regel van de editor.
  - Typ de volgende code:

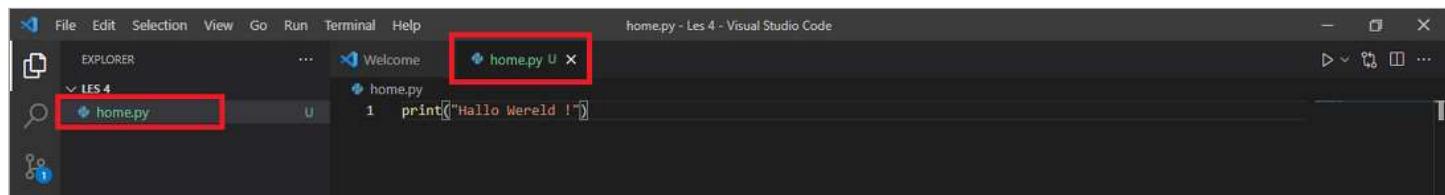
```
print("Hallo Wereld !")
```



*Typ een regel code.*

- Klik in de Menu Bar op "File".
  - Klik op "Save As...".
  - Sla het bestand op onder de naam 'home'.

Dit bestand zal verschijnen in de *Explorer* (in de *Side Bar*). Ook het tabblad krijgt de naam 'home.py':



#### 4.3.6 Veranderingen ongedaan maken

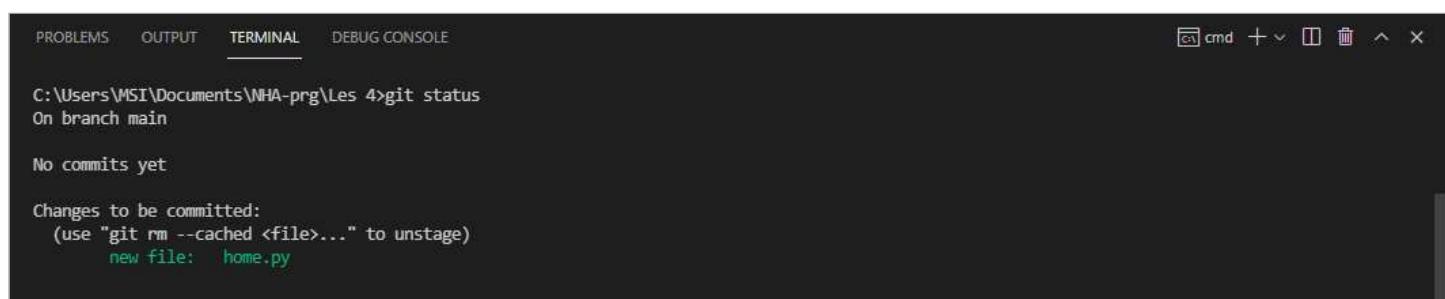
In deze subparagraaf laten we u zien u hoe u uw wijzigingen ongedaan kunt maken.

- Ga naar een nieuwe regel in de *Terminal*.
  - Typ `git add .` en druk op Enter. (Vergeet de `.` niet!)

Met dit commando voegt u bestanden toe die u veranderde, toevoegde of verwijderde. In uw geval niet zo heel spannend: u maakte slechts één bestand aan met de naam 'home.py' en dit bestand bevat slechts één regel code.

- Ga naar een nieuwe regel in de Terminal.
  - Typ `git status` en druk op Enter.

In de *Terminal* verschijnt nu een lijst met bestanden die u veranderde, toegevoegde en/of verwijderde:



En inderdaad: er is sprake van één nieuw bestand, genaamd 'home.py'.

- Zet de cursor in de tweede regel van de editor (van het bestand 'home.py').

- Typ de volgende code:

```
print("Van deze regel krijg ik later spijt")
```

- Klik in de *Menu Bar* op "File".
- Klik op "Save".
- Ga naar een nieuwe regel in de *Terminal*.
- Typ `git restore .` en druk op Enter. (Vergeet de `.` niet!)
- Kijk naar de code in de editor.

Als het goed is, ziet u nog maar één regel code. De tweede regel is verdwenen.



Dit is slechts een manier om de versies van onze software te beheren. Later zult u nog kennismaken met andere manieren, zoals `git reset`.

### 4.3.7 Definitieve versie maken

Tot slot maken we in deze subparagraaf een definitieve versie van het bestand.

- Ga naar een nieuwe regel in de *Terminal*.
- Typ `git add .` en druk op Enter. (Vergeet de `.` niet!)
- Typ `git commit -m "Dit is de eerste versie"` en druk op Enter.

Met `commit` geeft u aan dat u de wijzigingen definitief wilt opslaan. De `-m` geeft u de mogelijkheid om aan deze versie een bericht toe te voegen.

U krijgt ongeveer het volgende bericht terug in de *Terminal*:

```
C:\Users\MSI\Documents\NHA-prg\Les 4>git restore .
C:\Users\MSI\Documents\NHA-prg\Les 4>git add .
C:\Users\MSI\Documents\NHA-prg\Les 4>git commit -m "Dit is de eerste versie"
[main (root-commit) 27a95ee] Dit is de eerste versie
 1 file changed, 1 insertion(+)
   create mode 100644 home.py
```

U heeft nu een (harde) versie gecreëerd. Wat er nu verder ook gebeurt, u kunt altijd terugkeren naar deze versie. Dat zullen we in de volgende les doen.

## 4.4 Werken met GitHub

U heeft op uw computer *Git* geïnstalleerd en dit dient als uw lokale versiebeheersysteem. In deze paragraaf gaat u een versiebeheersysteem in de cloud opzetten: *GitHub*. Dit systeem kan enerzijds dienen als back-up, maar maakt het ook mogelijk om vanaf welke werkplek dan ook aan uw code te kunnen werken (mits u een internetverbinding heeft). Daarnaast maakt dit het samenwerken aan een code gemakkelijker.



*GitHub* is een versiebeheersysteem in de cloud.

In deze paragraaf laten we u zien hoe u een *GitHub*-account aanmaakt en hoe u een bestand kunt uploaden.

We beginnen met het aanmaken van een account:

- Ga naar [www.github.com](https://www.github.com).

Op deze pagina kunt u inloggen (rechtsboven), maar u kunt er ook een nieuw account aanmaken. We gaan ervan uit dat u nog geen account heeft, dus we kiezen voor dat laatste.

- Vul uw e-mailadres in.
- Klik op "Sign up for Github".

The screenshot shows the GitHub homepage with a dark blue background. At the top, there's a navigation bar with links like 'Why GitHub?', 'Team', 'Enterprise', 'Explore', 'Marketplace', and 'Pricing'. On the right side of the header are 'Search GitHub', 'Sign in', and a 'Sign up' button. The main headline reads 'Where the world builds software' in large white text. Below it, a sub-headline says 'Millions of developers and companies build, ship, and maintain their software on GitHub—the largest and most advanced development platform in the world.' A large globe graphic is on the right. In the center, there's a red-bordered box around the 'Email address' input field and the 'Sign up for GitHub' button. Below this, there are four statistics: '65+ million Developers', '3+ million Organizations', '200+ million Repositories', and '72% Fortune 50'. At the bottom right, there's a cartoon character of a person in a space suit standing on a small patch of green grass.

Vul uw e-mailadres in en klik op "Sign up for GitHub".

- Bevestig uw e-mailadres.
- Vul een wachtwoord in.
- Kies een gebruikersnaam.
- Geef aan of u productupdates en aankondelingen wilt ontvangen.
- Los een simpele puzzel op om te bewijzen dat u geen robot bent.
- Klik op "Create account".

U ontvangt nu een e-mail van *GitHub*, met daarin een *launch code*.

- Open de e-mail.
- Voer de code in op de webpagina om uw e-mailadres te verifiëren.

Daarna is uw account klaar voor gebruik. U ontvangt een welkomstmail en wordt doorgestuurd naar de landingspagina van uw profiel.

- Klik in het linkermenu op "Create repository".

The screenshot shows the GitHub homepage. At the top left, there's a search bar and navigation links for 'Pull requests', 'Issues', 'Marketplace', and 'Explore'. A prominent green button labeled 'Create repository' is highlighted with a red box. To the right, there's a 'Learn Git and GitHub without any code!' section with a 'Read the guide' button, another green 'Create repository' button, and a 'Start a project' button. Below this is a 'Recent activity' section with a note about tracking actions across GitHub. Further down are sections for introducing yourself (with a sample README file) and discovering interesting projects/people. At the bottom, there's footer information including copyright (© 2021 GitHub, Inc.) and links to 'Blog', 'About', 'Shop', 'Contact GitHub', 'API', 'Training', 'Status', 'Privacy', 'Docs', 'Security', and 'Pricing'.

Klik op "Create repository".

- Vul als "Repository name" de naam **ijssalon** in. (U zult later snappen waarom we voor deze naam kiezen.)

De overige instellingen veranderen we nu niet.

- Klik op "Create repository".

The screenshot shows the 'Create a new repository' form. At the top, it says 'Create a new repository' and provides a note about what a repository contains. The 'Owner' dropdown is set to 'testNHA' and the 'Repository name' field contains 'ijssalon'. Below this, there's a note about great repository names being short and memorable, with a suggestion for 'sturdy-octo-parakeet'. There's also an optional 'Description (optional)' field with a placeholder. Under 'Initialize this repository with:', there are three options: 'Public' (selected), 'Private', and 'Add a README file'. The 'Public' option has a note about anyone on the internet seeing the repository. The 'Private' option has a note about choosing who can see and commit. Below these are three checkboxes for 'Add a README file', 'Add .gitignore', and 'Choose a license', each with its own descriptive note. At the bottom is a large green 'Create repository' button. The footer includes standard GitHub links like 'Blog', 'About', 'Shop', 'Contact GitHub', 'Pricing', 'API', 'Training', 'Status', 'Privacy', 'Docs', 'Security', and 'About'.

Vul een repository name in en klik op "Create repository".

U heeft uw eerste *repository* aangemaakt. Nadat u op de groene knop heeft gedrukt, komt u terecht op de *repository page* van uw nieuwe pagina. Hier vindt u de verschillende mogelijkheden die *GitHub* u biedt. **Verlaat deze pagina nog niet:** deze pagina bevat informatie die u later in deze les nog nodig heeft.

Als laatste actie zullen we de *online repository* koppelen aan de map die we eerder aanmaakten en initialiseerden voor *Git*.



Heeft u de map in *VS Code* per ongeluk gesloten? Voer dan de volgende stappen uit om terug te komen in de juiste map:

- Open *VS Code*.
- Klik in de *Menu Bar* op “File”.
- Klik op “Open Folder”.
- Zoek naar de map die u eerder in deze les aanmaakte.

- Klik in de *Menu Bar* op “Terminal”.
- Klik op “New Terminal”.
- Ga terug naar de *GitHub*-webpagina (die we bewust open lieten staan).

Ergens halverwege de pagina ziet u deze zin staan:

... or push an existing repository from the command line

De drie regels die eronder staan, zijn de regels die u nodig heeft. Die zullen er ongeveer als volgt uitzien:

- Ga terug naar *VS Code*.
- Ga naar een nieuwe regel in de *Terminal*.
- Plak de eerste regel in de *Terminal*.
- Druk op Enter.

Met deze eerste opdrachtregel maken we een connectie aan tussen de map die we in *Git* geopend hebben en de *repository* die u heeft aangemaakt in *GitHub* (ook wel de **remote repository** genoemd).

- Plak de tweede regel in de *Terminal*.
- Druk op Enter.

Elke repository heeft een standaard *branch* nodig. De *branch* waarin wij gaan werken, heeft de standaardnaam ‘main’. Dat geven we aan met deze tweede opdrachtregel.

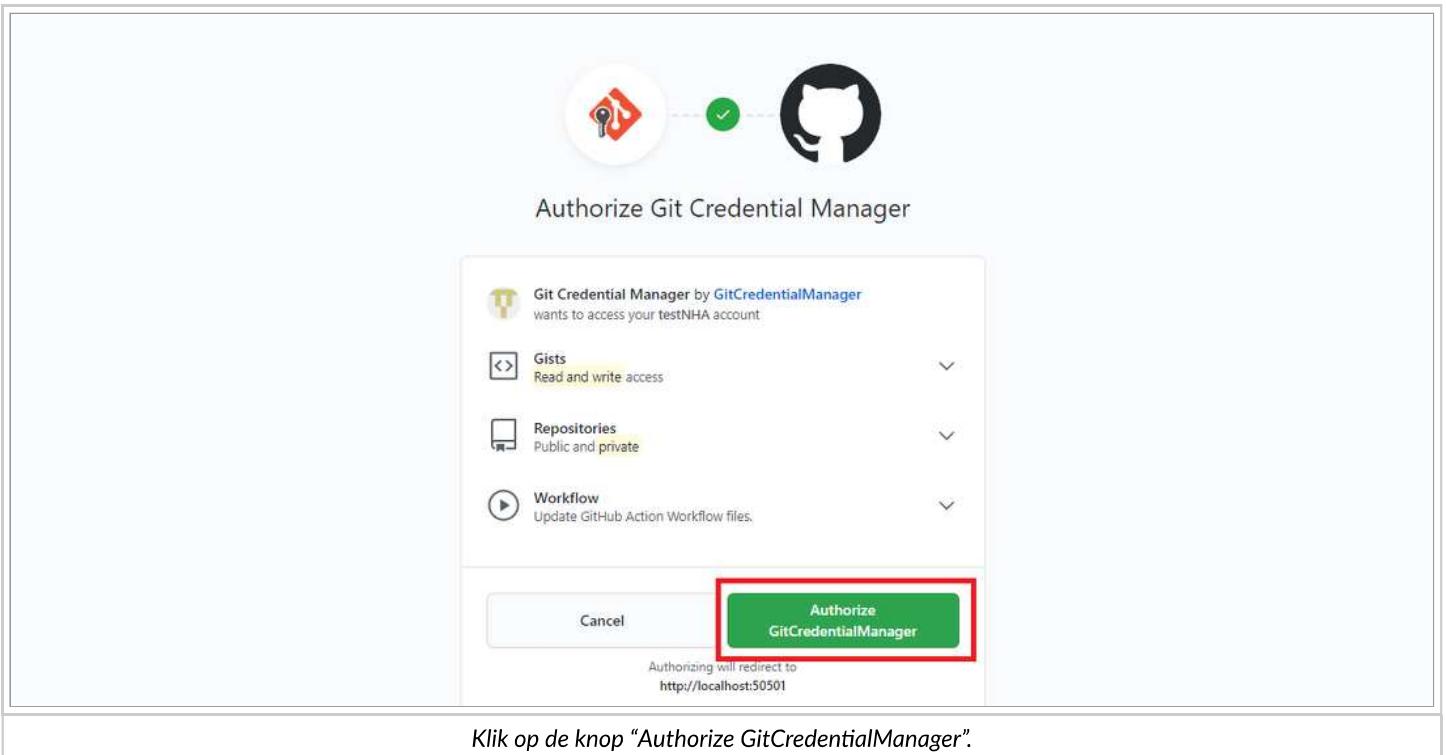
- Plak de derde regel in de *Terminal*.
- Druk op Enter.

Met dit commando uploaden we alle lokale code naar de *online repository*.

Bij het uitvoeren van de laatste opdrachtregel zal u gevraagd worden in te loggen met uw *GitHub*-gebruikersnaam en wachtwoord:

```
C:\Users\MSI\Documents\NHA-prg\Les 4>git push -u origin main
info: please complete authentication in your browser...
```

- Ga terug naar de *GitHub*-webpagina.
- Voer uw gebruikersnaam en wachtwoord in.
- Klik op de knop “Authorize Git CredentialManager”.



Klik op de knop "Authorize GitCredentialManager".

- Ga terug naar VS Code.

VS Code heeft nu toegang tot uw GitHub-account. Dat is te zien aan de boodschap in de Terminal, die er ongeveer zo uitziet:

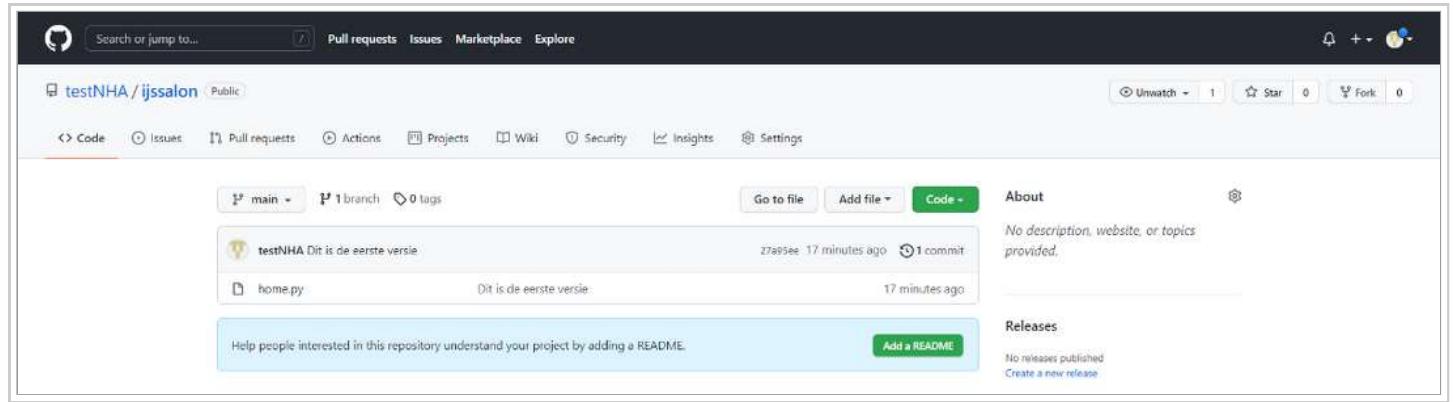
```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

Enumerating objects: 6, done.
Counting objects: 100% (6/6), done.
Delta compression using up to 2 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (6/6), 524 bytes | 52.00 KiB/s, done.
Total 6 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/testNHA/ijssalon.git
 * [new branch] main -> main
Branch 'main' set up to track remote branch 'main' from 'origin'.
```

Dit betekent dat u uw lokale project vanaf nu kunt synchroniseren met de repository in de cloud.

- Ga terug naar de GitHub-webpagina.
- Bekijk de repository 'ijssalon'.

U zult zien dat deze niet meer leeg is, maar dat deze de pagina 'home.py' bevat! (Hiervoor moet u mogelijk eerst nog op 'ijssalon' klikken op de GitHub-pagina.)



Het bestand 'home.py' staat nu ook in de online repository.

## 4.5 Versiebeheer in de praktijk

Tot slot doorlopen we in deze paragraaf het proces in de praktijk. We gaan achtereenvolgens het volgende doen:

1. Code toevoegen
2. Veranderingen toevoegen aan de Git-updatelijst (commando `git add`)
3. Alle veranderingen uit de Git-updatelijst vastleggen in een versie (commando `git commit`)
4. Nieuwe versie uploaden naar GitHub (commando `git push`)

Als u uzelf deze manier van werken aanleert, heeft u zowel lokaal als in de cloud een back-up van uw werk. Mocht er iets misgaan, dan kunt u altijd terug naar een vorige versie.

### 4.5.1 Code toevoegen

Om veranderingen aan de Git-updatelijst te kunnen toevoegen, moeten er natuurlijk eerst veranderingen zijn. We gaan daarom allereerst code toevoegen.

- Open VS Code.
- Open het bestand 'home.py'.
- Voeg een tweede regel code toe:

```
print("Welkom in onze ijssalon")
```

- Klik in de Menu Bar op "File".
- Klik op "Save".



Heeft u het al door? We bouwen in deze cursus een webshop voor een ijssalon!

### 4.5.2 Git add

We gaan nu de code toevoegen aan de lijst met wijzigingen, die uiteindelijk in een versie vervat zullen worden.

- Ga naar een nieuwe regel in de Terminal.
- Typ `git add .` en druk op Enter. (Vergeet de `.` niet!)

### 4.5.3 Git commit

Weet u het nog? Met `commit` geeft u aan dat u de wijzigingen definitief wilt opslaan. Dat is wat we nu gaan doen met de lijst met wijzigingen (in ons geval: één wijziging).

- Ga naar een nieuwe regel in de Terminal.
- Typ `git commit -m "1 regel toegevoegd"` en druk op Enter.

### 4.5.4 Git push

Tenslotte gaan we de nieuwe versie toevoegen aan de kopie in de cloud. Door middel van het commando `git push` uploaden we de lokale repository naar GitHub.

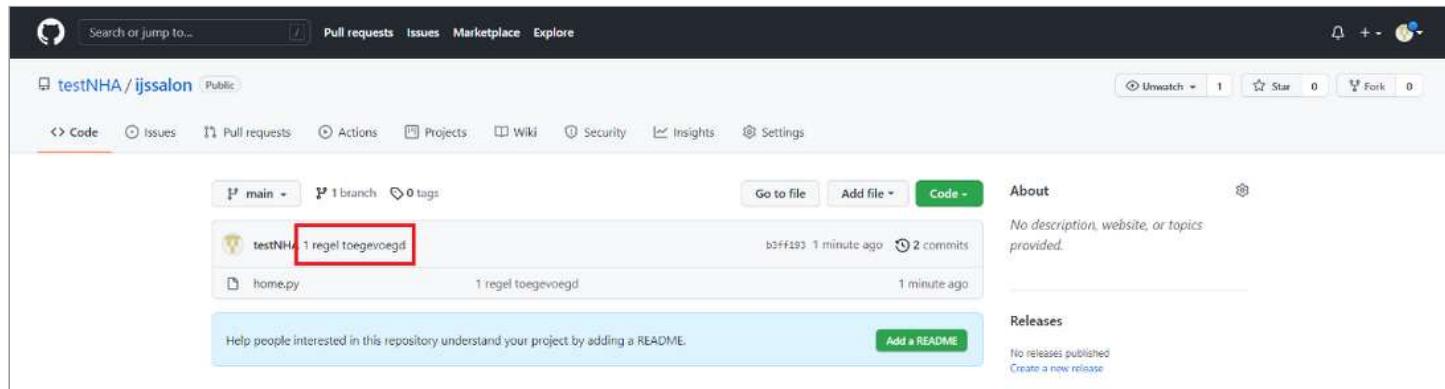
- Ga naar een nieuwe regel in de Terminal.
- Typ `git push` en druk op Enter.

Het kan iets langer duren dat u gewend bent, maar uiteindelijk krijgt u een reactie als deze te zien:

```
PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

C:\Users\MSI\Documents\NHA-prg\Les 4>git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 2 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 296 bytes | 59.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0), pack-reused 0
To https://github.com/testNHA/ijssalon.git
  27a95ee..b3ff193 main -> main
```

Als alles goed is gegaan, ziet u de verandering terug op GitHub:

A screenshot of a GitHub repository page for 'testNHA/ijssalon'. The page shows a single commit from user 'b3ff193' made 1 minute ago. The commit message is '1 regel toegevoegd'. This message is highlighted with a red rectangular box. Below the commit, there is a note: 'Help people interested in this repository understand your project by adding a README.' followed by a green 'Add a README' button.

(Mogelijk moet u de pagina verversen voordat u de verandering ziet.)

## 4.6 Samenvatting

Voordat we echt gaan programmeren, moest er nog een onderwerp worden behandeld: versiebeheer. Een belangrijk onderwerp, want door u bewust bezig te houden met versiebeheer, vermindert u het aantal misverstanden en fouten.

Versiebeheer is het bijhouden van de verschillende fases van een bestand: van aanmaak en aanvulling tot aan de definitieve versie. In de programmeerwereld zijn verschillende populaire versiebeheersystemen verkrijgbaar. Het populairste systeem is *Git*. Dit gedistribueerd versiebeheersysteem heeft u in deze les geïnstalleerd op uw computer.

Daarnaast lieten we zien hoe u onder meer een map verandert in een *repository* en wijzigingen ongedaan maakt (of juist definitief opslaat) met behulp van *Git*.

Behalve met het lokale systeem *Git* werkten we in deze les ook met een versiebeheersysteem in de cloud: *GitHub*. U maakte een account, creëerde een online *repository* en koppelde deze aan de map die u eerder aanmaakte in *Git*.

Tot slot doorliepen we de manier van werken die u zichzelf zou moeten aanleren. Hierbij zijn drie *Git*-commando's van belang:

git add .	Veranderingen toevoegen aan de <i>Git</i> -updatelijst
git commit -m "uw bericht"	Alle veranderingen uit de <i>Git</i> -updatelijst vastleggen in een versie
git push	Nieuwe versie uploaden naar <i>GitHub</i>

*GitHub* heeft zoveel functionaliteiten en mogelijkheden dat u als beginner wellicht door de bomen het bos niet meer ziet. Kijk gerust rond, maar laat u niet gek maken. De voor u belangrijkste functie hebben we behandeld in deze les. Deze functie bestuurt u vanuit *VS Code*.

We hebben in de voorgaande vier lessen een lange aanloop genomen naar het werkelijke programmeren. Het duurt even, maar dan heeft u ook wat: u komt zeer beslagen ten ijs!

## 4.7 Vraagmoment

Heeft u vragen over deze les? Dan kunt u die stellen via een vraagmoment.

U herkent een vraagmoment aan het spreekballonnetje dat op Plaza in het overzicht "Lessen" bij een les staat. Door op deze spreekballoon te klikken, komt u op een nieuwe pagina, waar u uw vraag in het tekstvak kunt typen.



Het tekstvak mag maximaal vijfhonderd tekens bevatten. Zorg er dus voor, dat u uw vraag/vragen bondig formuleert.

Nadat u op "Verstuur vraag" heeft geklikt, wordt uw vraag automatisch naar uw docent verstuurd. Zodra uw docent de vraag heeft beantwoord, verschijnt er op uw dashboard bij de opleiding een spreekballoon.

U kunt per les één keer vragen stellen. Verzamel dus uw vragen over de les, voordat u deze instuurt. Als u gebruik heeft gemaakt van het vraagmoment, verdwijnt het spreekballonnetje bij de les.

## 4.8 Oefenopgaven

De volgende opgaven werkt u als oefenopgaven voor uzelf uit. De uitwerkingen van deze opgaven kunt u meteen zelf controleren.

Wat is versiebeheer?

[Bekijk antwoord](#)

Welke twee dingen doet u als u zich bezighoudt met versiebeheer?

[Bekijk antwoord](#)

Wat is het grote verschil tussen versiebeheersystemen voor programmeurs en het versiebeheersysteem van een programma als Microsoft Word?

[Bekijk antwoord](#)

Git is een gedistribueerd versiebeheersysteem. Wat betekent dit?

[Bekijk antwoord](#)

Wat is een repository?

[Bekijk antwoord](#)

Met welk commando voegt u wijzigingen toe aan de Git-updatelijst?

[Bekijk antwoord](#)

Wat geeft u aan met het commando git commit?

[Bekijk antwoord](#)

Wat is het grote verschil tussen Git en GitHub?

→ Bekijk antwoord

Met welk commando uploadt u uw lokale repository naar GitHub?

→ Bekijk antwoord

## 4.9 Huiswerkopgaven

Maak deze huiswerkopgaven en stuur ze via Plaza naar uw docent. Deze opgaven worden nagekeken en voorzien van een cijfer. Mocht u een onvoldoende cijfer behalen, dan krijgt u een herkansing.

1. Er bestaat niet alleen het commando `git push`, maar ook het commando `git pull`. Zoek uit wat dit laatste commando doet en omschrijf dit in uw eigen woorden. (Probeer het ook zelf eens uit!)
2. Ook is er het commando `git clone`. Zoek uit wat dit commando doet en omschrijf dit in uw eigen woorden. (Probeer het ook zelf eens uit!)
3. Schrijf een commando waarmee u alle wijzigingen in de Git-updatelijst definitief opslaat. Zorg ervoor dat de versie de boodschap `Bugs opgelost` meekrijgt.
4. Voeg de volgende regel code toe aan het bestand 'home.py':

```
print("Under construction")
```

Voeg deze verandering toe aan de Git-update lijst, leg alle veranderingen uit deze lijst vast in een versie en upload deze versie naar GitHub. Maak een printscreen van uw scherm (inclusief Terminal) in VS Code en een printscreen van de GitHub-pagina waarop deze verandering is terug te zien.

Hoe maak ik een printscreen?

- Als Windows-gebruiker drukt u hiervoor op de printscreenknoop op uw toetsenbord. Plak de afbeelding vervolgens in het programma `Paint` (of een vergelijkbaar programma) om deze op te slaan als afbeelding of plak de afbeelding in `Microsoft Word`.



- Als Mac-gebruiker drukt u tegelijkertijd de toetsenbordcombinatie CMD (ofwel Apple-toets) + SHIFT + 3 en laat ze weer los. De printscreen wordt dan automatisch op uw desktop opgeslagen.