

Git

Continuous Integration Basics

8	WERKEN MET REMOTE REPOSITORIES	3
8.1	GitHub	3
8.2	Alternatieven	4
9	GIT REMOTE REPOSITORIES MET GITHUB	5
9.1	Werken met remotes	5
9.2	Laat je remotes zien	5
9.3	Remote repositories toevoegen	6
9.4	Wijzigingen doorvoeren naar de remote repository	11
9.5	Een bestaande lokale repository toevoegen aan een remote repository	15
9.6	Clonen van een remote repository	16
9.7	Remote repositories verwijderen	17
9.8	Lokale repository synchroniseren met je remote repository	17
9.9	Samenwerken met meerdere personen in één repository	19
9.10	Een aantal commando's	20

8 WERKEN MET REMOTE REPOSITORIES

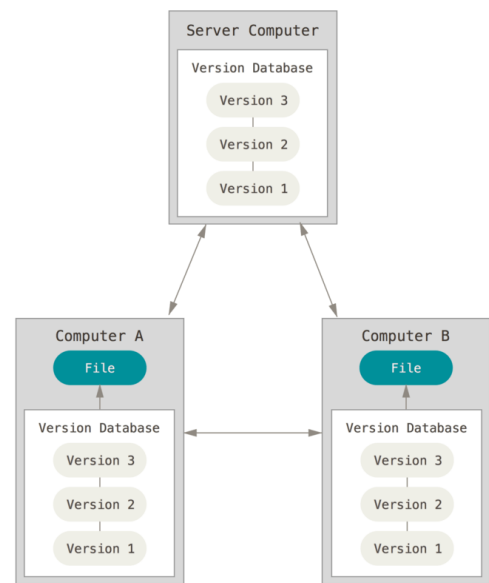
De manier van werken die we tot hertoe gezien hebben is een prima start. Echter zal het enkel de wijzigingen bijhouden op de computer waar je mee aan de slag bent. Omdat git een **Distributed Version Control System** is zou het zonde zijn als we enkel lokaal blijven werken en niet verder ingaan op de kracht hiervan.

In een Distributed Version Control System zullen de clients niet enkel gaan kijken naar de nieuwste momentopnames van de bestanden. Integendeel, ze zullen steeds het volledige plaatje gaan weerspiegelen, inclusief de volledige geschiedenis.

Met andere woorden als een server er tussen uit valt dan kunnen elk van de client repositories die via deze server samenwerkten simpelweg opnieuw naar een server gekopieerd worden om een 'restore' (herstelpunt) te bekomen.

Elke kloon is dus effectief een volledige* back-up van alle gegevens.

* Uiteraard enkel tot en met het laatste moment waarop de client gesynchroniseerd heeft met de online repository.



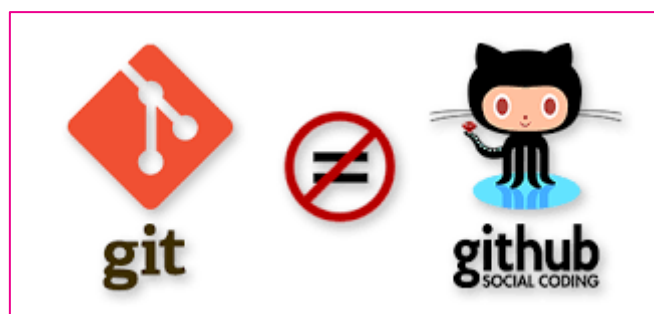
8.1 GITHUB

Binnen deze lessenreeks zullen we gebruik maken van GitHub. Github is wereldwijd een van de populairste platformen gebouwd rondom git. Github voorziet ons van alle mogelijkheden die git ons ter beschikking stelt met daarenboven nog eens heel wat bijkomende functionaliteit. Een ideaal platform als keuze voor onze remote repositories.

Met andere woorden (*programmeer gerelateerd*) de locatie waar we onze code zullen stockeren zodat we op meerdere plaatsen of met meerdere developers aan onze code kunnen werken.

Let op!

Zoals hierboven reeds vermeld is Github een **platform** waar je git repositories kan plaatsen. Git en Github hebben voor de rest niets met elkaar te maken. Github is gebouwd omheen de git functionaliteit.



8.2 ALTERNATIEVEN

Github is uiteraard niet de enige oplossing. Er zijn tal van alternatieven beschikbaar:

- Bitbucket
- GitLab
- DevHub
- Beanstalk
- SourceForge
- Apache Allura
- Cloud Source (Google)
- AWS CodeCommit
- GitKraken

Bron: [9 Github Alternatives for Source Code and Version Control](#)

Onderstaand een vergelijking van enkele aanbieders:

Aanbieder	Public Repositories	Private Repositories
Visual Studio Online	-	Aantal ongelimiteerd Gratis voor 5 gebruikers
GitHub	Aantal ongelimiteerd Gratis	Aantal ongelimiteerd Gratis (t.e.m. 3 collaborators)
BitBucket (Atlassian)	Aantal ongelimiteerd Gratis <i>(Academic subscription geeft je meerdere build minutes)</i>	Gratis tot 5 gebruikers Betaland vanaf 6 gebruikers

9 GIT REMOTE REPOSITORIES MET GITHUB

9.1 WERKEN MET REMOTES

Om samen te kunnen werken aan eender welk project, moet je uiteraard kennis hebben over hoe je de remote repositories moet beheren. Remote repositories zijn versies van je project, die worden beheerd op het internet (*bijvoorbeeld GitHub*) of ergens op een netwerk. Je kunt er gerust meerdere hebben, waarvan over het algemeen elke online repository lees- en schrijfbaar is voor jezelf en/of je teamleden en mogelijk publiek toegankelijk (*via leesrechten*) .

Samenwerken met anderen houdt in dat je deze remote repositories kunt beheren en data kunt pushen en pullen op het moment dat je werk moet delen. Remote repositories beheren houdt ook in weten hoe je ze moet toevoegen, ongeldige repositories moet verwijderen, meerdere remote branches moet beheren en ze als tracked of niet kunt definiëren, en meer. In dit onderdeel van de cursus zullen we deze remote-beheer vaardigheden behandelen.

9.2 LAAT JE REMOTES ZIEN

Om te zien welke remote servers je geconfigureerd hebt, kun je het `git remote` commando uitvoeren. Het laat de namen (*alias**) zien van iedere **remote** die je geconfigureerd hebt. Als je als startpunt een repository gecloned hebt, dan zul je op z'n minst de **origin** zien. Dat is standaard de naam die Git aan de server geeft waarvan je gecloned hebt.

Alvorens er remote repositories getoond kunnen worden zullen we deze moeten gaan toevoegen. Hoe we dit doen bespreken we in het volgende onderdeel.

* De locatie van een remote repository is vaak een relatief lange url. De **alias** zorgt ervoor dat we deze locatie zelf een naam kunnen toewijzen. **Origin** is de alias die standaard door git gebruikers toegewezen wordt aan de locatie van de remote repository.



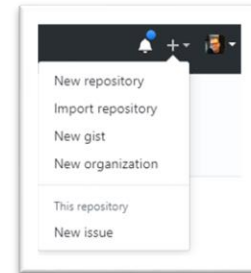
9-1 <https://unito.io/blog/beginners-guide-to-github/>

9.3 REMOTE REPOSITORIES TOEVOEGEN

Om een nieuw Git remote repository als een makkelijk te refereren alias toe te voegen, voer je `git remote add [alias] [url]` uit.

Aangezien we nog geen remote repository hebben zullen we eerst een eigen remote repository gaan aanmaken in GitHub:

- Surf naar GitHub
- Log in met je gegevens
- Klik rechts bovenaan op het plusteken en kies New repository
- Op het volgend scherm vul je de Repository naam in (in dit geval **EersteKennismaking**)
- Vul eventueel een description in
- Plaats je repository **private**, zo is deze enkel zichtbaar voor jezelf. Indien deze public staat is je repository zichtbaar voor iedereen.
- Vink "Initialize this repository with a README" **niet** aan.
- Add .gitignore mag op **None** blijven staan voor deze kennismaking.
- Klik daarna op de knop **Create Repository** om je remote repository te maken.
- Zie screenshot:



Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository.](#)

Owner

Repository name *

Drieze ▾

/

EersteKennismaking ✓

Great repository names are short and memorable. Need inspiration? How about **congenial-umbrella**?

Description (optional)

☐ Public

Anyone can see this repository. You choose who can commit.

☒ Private

You choose who can see and commit to this repository.

☐ Initialize this repository with a README

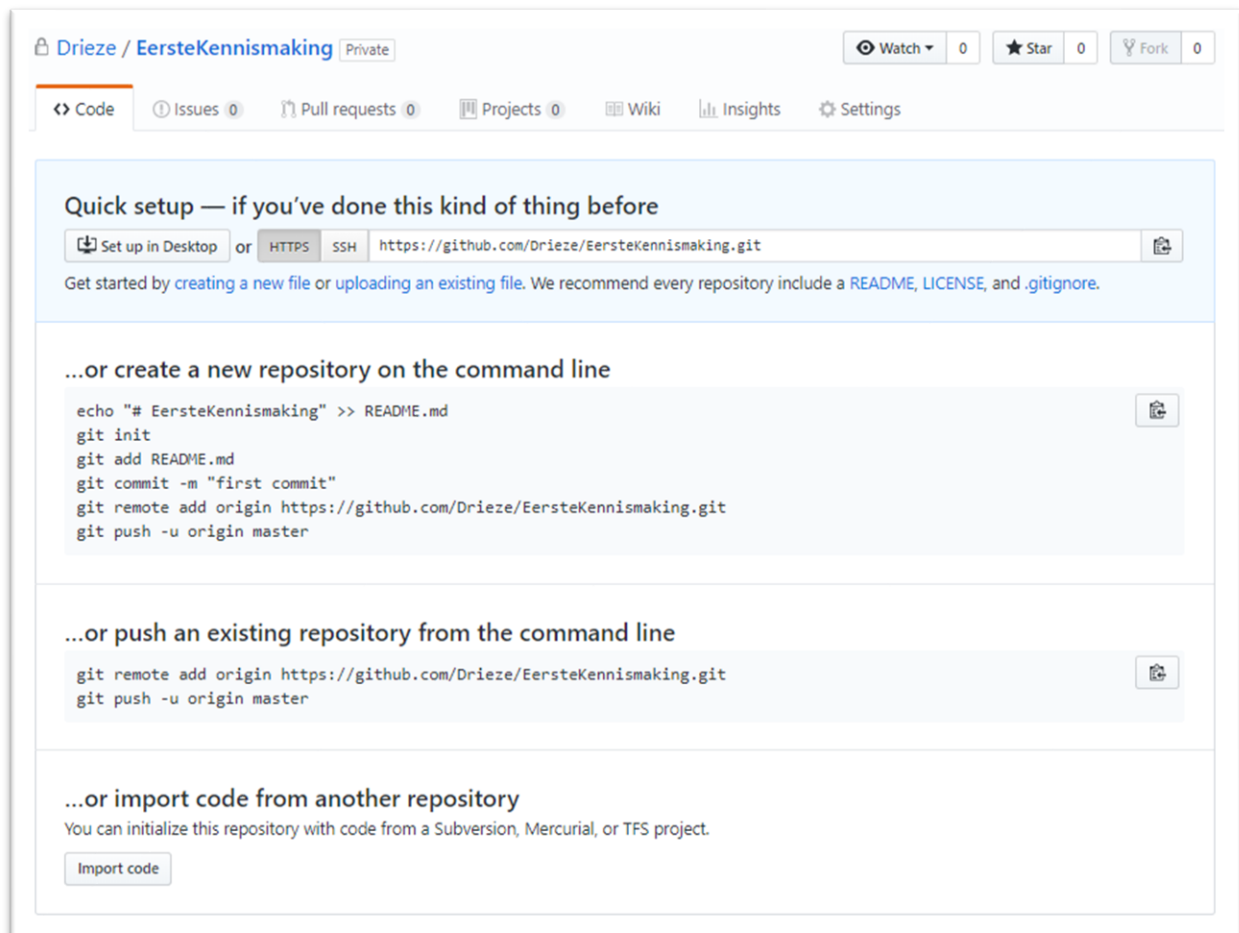
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: None ▾

Add a license: None ▾ ⓘ

Create repository

Onze online (remote) repository is nu gemaakt en we zien nu volgend scherm:

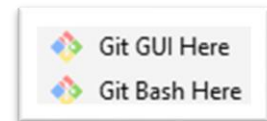


We krijgen vier mogelijkheden om deze remote repository te gebruiken:

- **Quick setup**
Voor de ervaren "Gitters", deze kunnen meteen aan de slag met de url die meegegeven werd.
- **Create a new repository on the command line**
Deze mogelijkheid gebruiken we wanneer we nog een lokale repository moeten maken en deze later willen syncen (pushen) met onze remote repository.
- **Push an existing repository from the command line**
Wanneer we reeds in het bezit zijn van een lokale repository en deze willen toevoegen aan een remote repository kunnen we met deze twee git bash commando's aan de slag.
- **Import code from another repository**
Mochten we remote repositories hebben in andere Git platformen zoals Subversion, Mercurial of TFS kunnen we aan de slag door op de knop "Import code" te klikken. In deze module gaan we hier niet mee aan de slag.

Wij zullen voor dit voorbeeld kiezen voor de tweede optie (Create a new repository on the command line).

- Maak op je bureaublad (of een andere locatie) een folder aan, en noem deze EersteKennismaking
- Dubbelklik op deze folder
- Klik met de rechtermuistoets op de lege folder en kies **Git Bash here**
Dit opent het Bash venster.

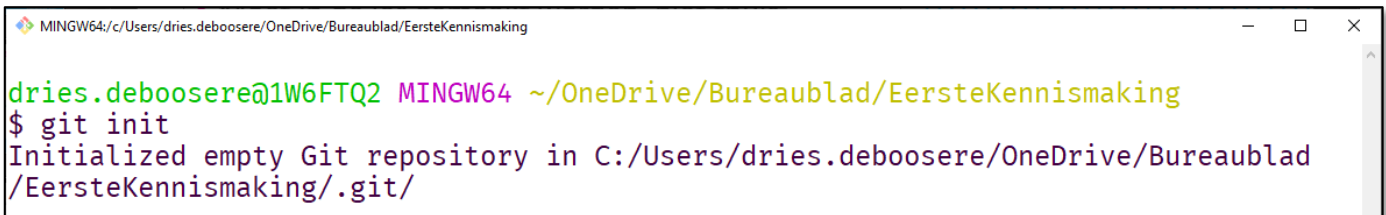


GitHub geeft ons de nodige info om ons op weg te zetten om een nieuwe repository te creëren:

```
echo "# EersteKennismaking" >> README.md
git init
git add README.md
git commit -m "First commit"
git remote add origin https://github.com/Drieze/EersteKennismaking.git
git push -u origin master
```

In ons geval zullen we starten vanaf de stap **git init**. Zoals we reeds in het vorige hoofdstuk hebben gezien is dit het commando om een (lokale) git repository aan te maken.

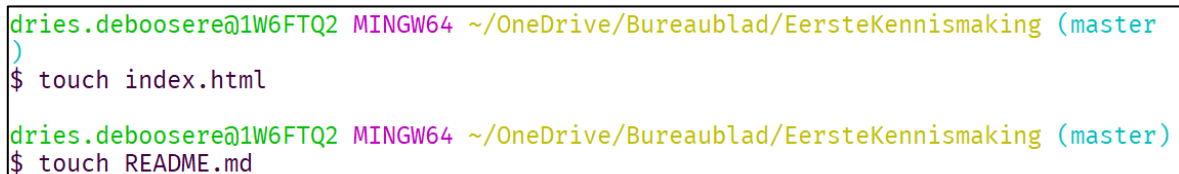
- In het Bash venster tik je dus het **git init** commando in:



```
MINGW64:/c:/Users/dries.deboosere/OneDrive/Bureaublad/EersteKennismaking
dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking
$ git init
Initialized empty Git repository in C:/Users/dries.deboosere/OneDrive/Bureaublad/EersteKennismaking/.git/
```

- Voeg in je folder EersteKennismaking twee bestanden toe, je kan dit doen via het **touch** commando:

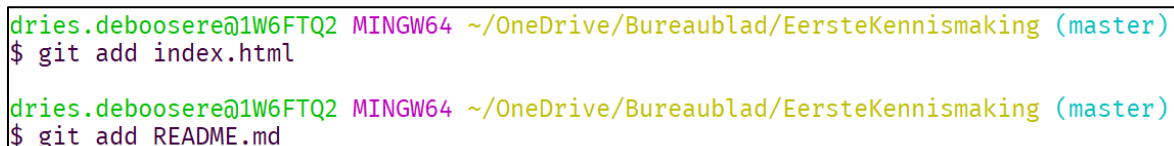
```
touch index.html
touch readme.md
```



```
dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ touch index.html

dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ touch README.md
```

Deze bestanden dienen we toe te voegen in onze **staging area** zodat deze bestanden meegenomen worden in onze volgende commit. Vorig hoofdstuk hebben we gezien dat we dit kunnen doen via het **git add [bestandsnaam]** commando of **git add ***



```
dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git add index.html

dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git add README.md
```


- Doordat beide bestanden opgenomen zijn in onze staging area zullen beide bestanden nu toegevoegd worden met onze nieuwe (nog te maken) commit.

Deze commit zullen we nu maken. Tik in je console het volgende commando in:

```
git commit -m "First commit"
```

We geven enkel een titel mee, geen body in onze commit.

```
dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git commit -m "First commit"
[master (root-commit) 9eeb8f3] First commit
2 files changed, 0 insertions(+), 0 deletions(-)
create mode 100644 README.md
create mode 100644 index.html
```

- Onze commit is nu gemaakt in onze lokale repository maar de bedoeling is natuurlijk dat deze repository online geplaatst wordt in onze remote repository in GitHub. Om dit te doen gebruiken we het volgende commando

```
git remote add origin https://github.com/JouwGitHubHandle/EersteKennismaking.git
```

Bij jullie zal de URL er anders uitzien, kopieer dus de URL die jullie verkregen hebben van GitHub.

Door dit commando uit te voeren zal je lokale repository gekoppeld worden met de remote repository.

```
dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git remote add origin https://github.com/Drieze/EersteKennismaking.git
```

We zien in dit commando het woord **origin** staan, hiermee geven we de verkorte naam van onze remote repository aan. Dit is ook de standaard naam die Git aan de server geeft als je een repository gekloond hebt van ergens anders (*komt later nog aan bod in deze cursus*).

Wij zullen onze Git remote repository dan ook altijd **origin** als verkorte naam geven.

- Wanneer we kijken op GitHub.com zien we dat onze bestanden nog niet te vinden zijn in de online/remote repository op GitHub. We hebben met het vorige commando er enkel maar voor gezorgd dat er een remote repository werd toegevoegd aan onze lokale repository. Om onze lokale repository te syncen met onze remote repository moeten we volgend commando uitvoeren:

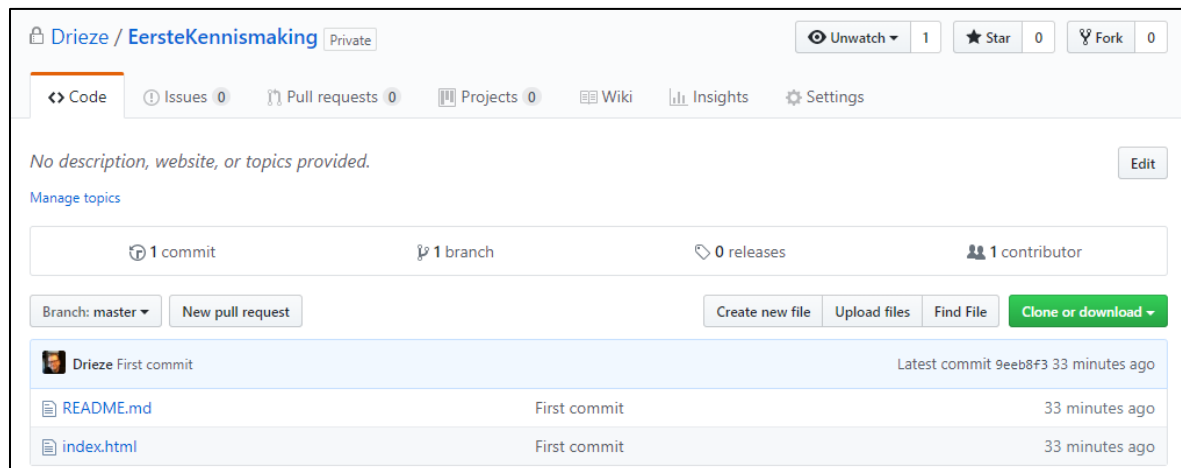
```
git push -u origin master
```

Met het git push commando zeggen we dat alle wijzigingen op onze lokale repository mogen doorgevoerd worden naar onze remote repository. De **-u** flag wil zeggen dat we een tracking reference toevoegen naar de upstream server waarnaartoe we zullen pushen. Door deze flag wordt er een link gelegd in onze lokale repository die de opgegeven remote samen met de opgegeven branch zal gaan linken aan elkaar. Hierdoor kunnen we in alle volgende stappen gebruik maken van de kortere syntax **git push** om onze wijzigingen door te voeren.

Origin is de verkorte naam van onze online Git repository en **master** betekent dat we zullen pushen naar onze **master branch** (*later meer over het branchen*).

```
dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git push -u origin master
Enumerating objects: 3, done.
Counting objects: 100% (3/3), done.
Delta compression using up to 8 threads
Compressing objects: 100% (2/2), done.
Writing objects: 100% (3/3), 230 bytes | 230.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Drieze/EersteKennismaking.git
 * [new branch]      master -> master
Branch 'master' set up to track remote branch 'master' from 'origin'.
```

- Als we nu teruggaan naar onze remote repository op GitHub zien we dat de wijzigingen (de twee nieuwe bestanden) nu wel te vinden zijn in onze GitHub repository.



We zien rechts van onze twee bestanden de titel van de laatste commit waarmee deze bestanden zijn gewijzigd (in ons geval aangemaakt) en we zien ook hoelang dit geleden is.

We zien ook dat er tot op heden **1 commit** is gemaakt en dat er **1 branch** aanwezig is voor deze repository.

- Wanneer we nu het git commando `git status` uitvoeren zien we dat onze lokale repository up to date is met onze remote repository:

```
dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git status
On branch master
Your branch is up to date with 'origin/master'.

nothing to commit, working tree clean
```

9.4 WIJZIGINGEN DOORVOEREN NAAR DE REMOTE REPOSITORY

Tot nu toe hebben we enkel een lege README.md en een lege index.html file staan in onze lokale en remote repository.

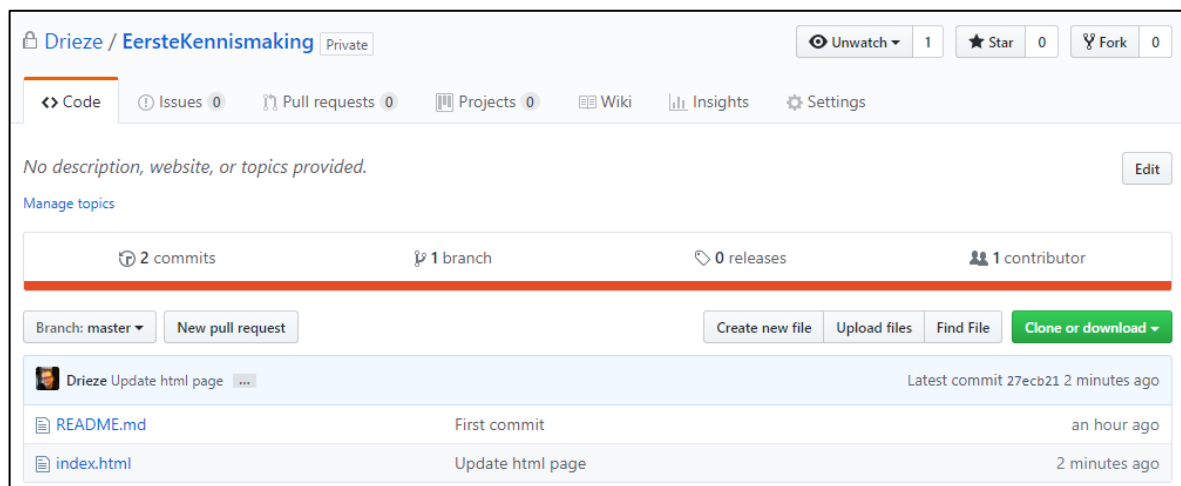
We gaan nu ons index.html bestand wat aanpassen. Voeg onderstaande HTML code toe in het html-bestand op je lokale repository. Met andere woorden, voeg dit toe aan het index.html bestand in de folder EersteKennismaking op jouw computer.

```
<!DOCTYPE html>
<html lang="nl">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <meta http-equiv="X-UA-Compatible" content="ie=edge">
  <title>Eerste kennismaking met GitHub</title>
</head>
<body>
  <h1>GitHub</h1>
  <p>Een plaats voor online repositories</p>
</body>
</html>
```

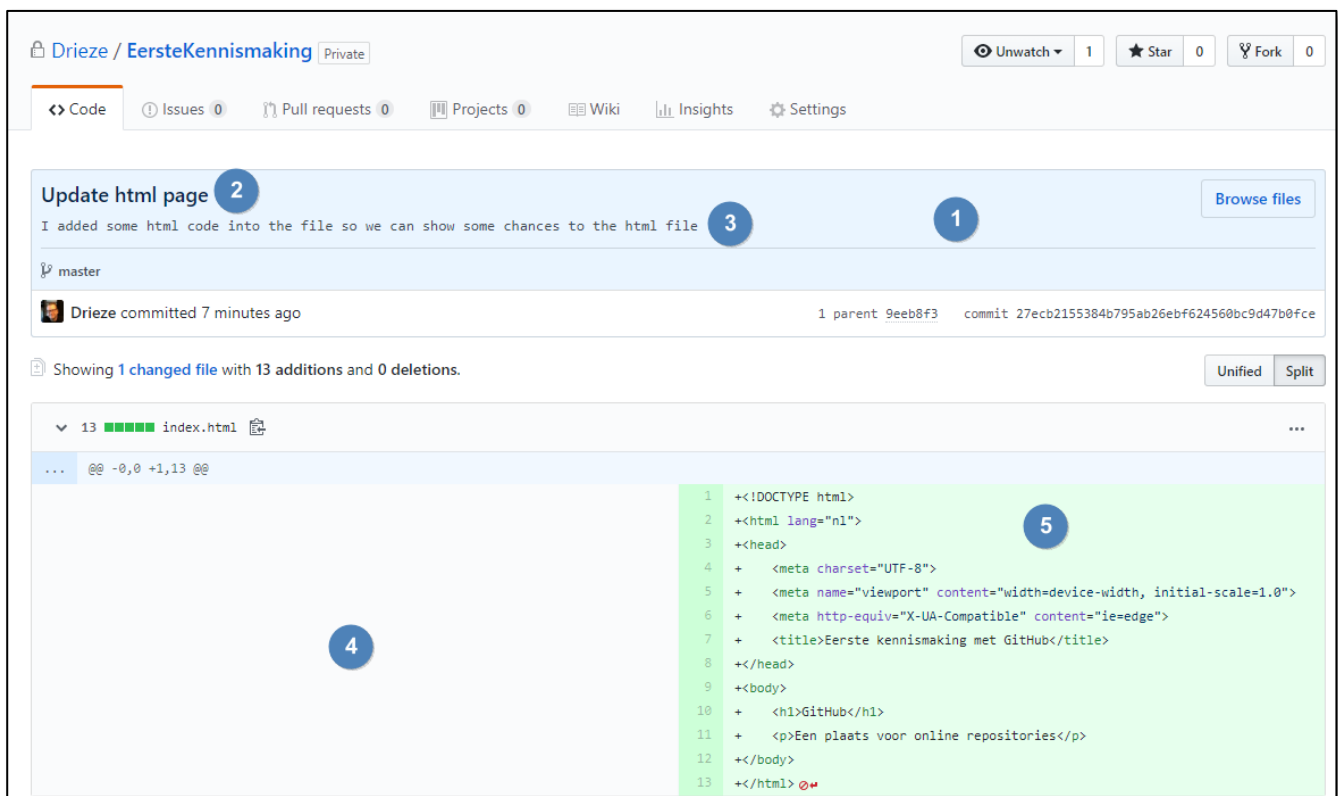
Met het `git status` commando zien we dat Git ziet dat er wijzigingen zijn gebeurd aan index.html maar dat deze wijzigingen nog niet zijn opgenomen in de staging area. Om deze code te pushen naar onze remote repository moeten we nu volgende stappen doorlopen:

- `git add index.html`
Het index.html bestand toevoegen aan de staging area
- `git status`
Even controleren of ons bestand nu wel opgenomen is in de staging area
- `git commit -m "Update html page"`
Met deze commit geven we aan dat we het wijzigingen hebben aangebracht. Deze commit heeft een title en een body meegekregen.
- `git push`
Met dit commando pushen we onze wijzigingen van onze lokale repository door naar onze remote repository

- Wanneer we dit controleren op GitHub zien we dat onze wijzigingen effectief zijn doorgevoerd:



- Wanneer we nu klikken op de titel van onze commit (Update html page) zien we op het volgende scherm volgende zaken:
 - a. Commit details
 - b. Commit titel
 - c. Commit body *(laten we voorlopig even uit de scope van onze cursus)*
 - d. Inhoud index.html bestand **voor** de commit
 - e. Inhoud index.html bestand **na** de commit



- We voeren in onze index.html bestand volgende wijzigingen door in onze lokale repository:

```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Tweede kennismaking met GitHub</title>
  </head>
  <body>
    <h1>GitHub</h1>
    <p>
      GitHub is een populaire website waarop software kan geplaatst worden. GitHub is
      gebouwd rond het Git-versiebeheersysteem, waardoor GitHub alle mogelijkheden van Git
      en eigen toevoegingen aanbiedt.
    </p>
  </body>
</html>
```

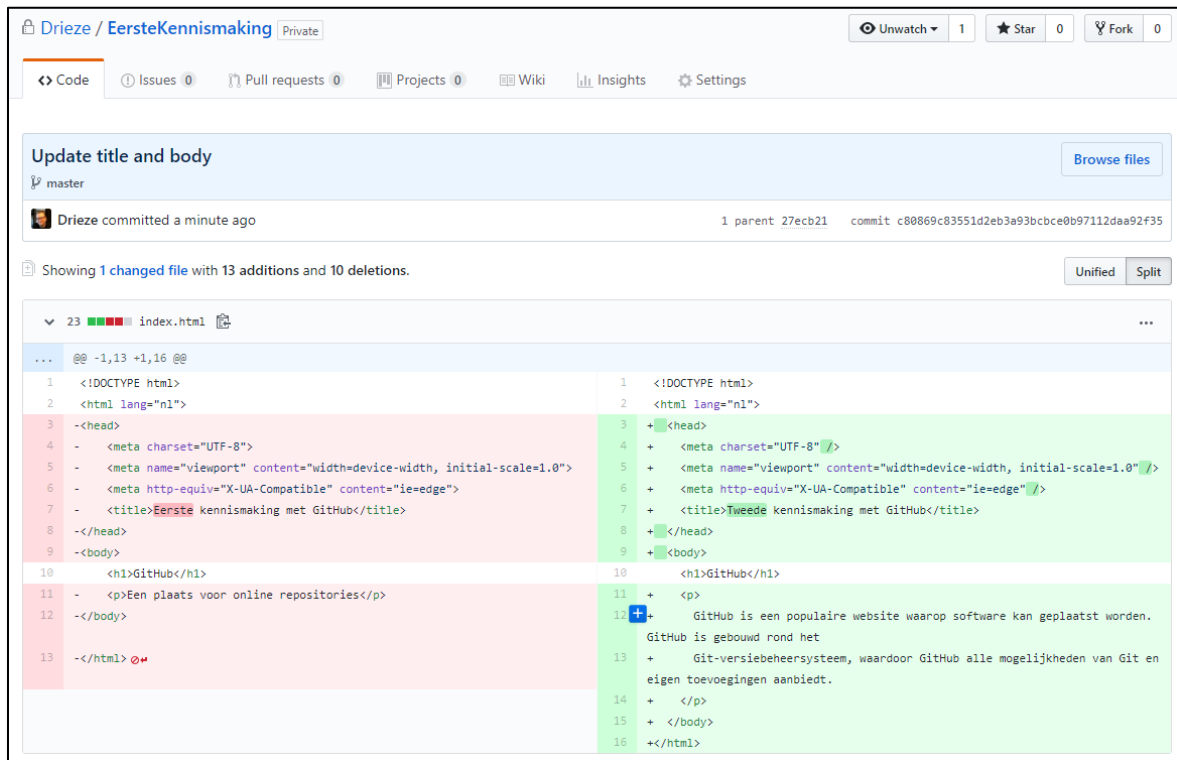
- We voegen dit bestand opnieuw toe aan de staging area, we doen een commit en we pushen de wijzigingen naar onze remote repository via de ondertussen gekende commando's.

```
dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git add index.html

dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git commit -m "Update title and body"
[master c80869c] Update title and body
1 file changed, 16 insertions(+), 13 deletions(-)
rewrite index.html (66%)

dries.deboosere@1W6FTQ2 MINGW64 ~/OneDrive/Bureaublad/EersteKennismaking (master)
$ git push
Enumerating objects: 5, done.
Counting objects: 100% (5/5), done.
Delta compression using up to 8 threads
Compressing objects: 100% (3/3), done.
Writing objects: 100% (3/3), 612 bytes | 612.00 KiB/s, done.
Total 3 (delta 0), reused 0 (delta 0)
To https://github.com/Drieze/EersteKennismaking.git
27ecb21..c80869c master -> master
```

- We klikken op onze commit details in GitHub en zien nu volgende scherm:

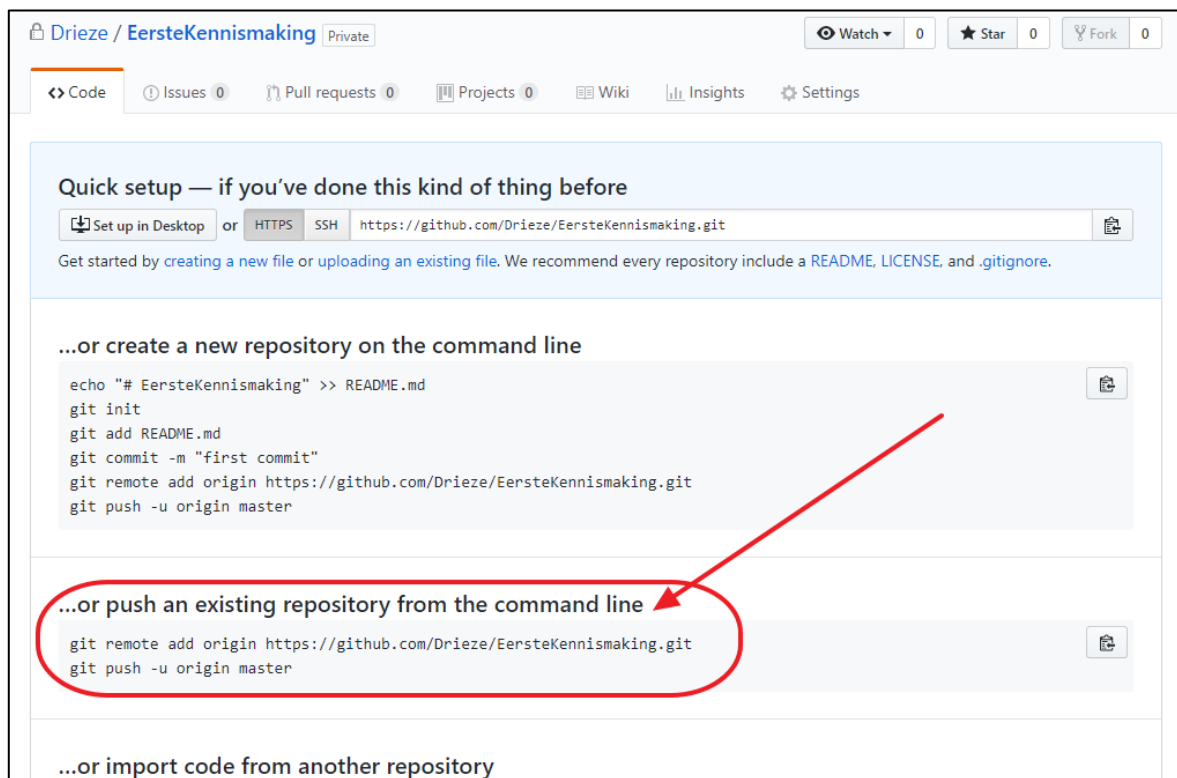


We zien opnieuw de toestand van ons bestand **voor** de commit en de toestand **na** de commit. In dit geval heeft Git 13 regels code toevoegingen gedaan en 10 regels code verwijderd.

9.5 EEN BESTAANDE LOKALE REPOSITORY TOEVOEGEN AAN EEN REMOTE REPOSITORY

In het vorige hoofdstuk hebben we eerst een lokale git repos aangemaakt, bestanden aangemaakt, bestanden gewijzigd en deze dan gepushed naar onze remote repository.

Bij het aanmaken van een remote repository in GitHub krijgen we ook de mogelijkheid om een bestaande lokale repository te koppelen aan een remote repository:



Indien we dus reeds een lokale repository hadden met de nodige commits, maar die nog niet gekoppeld was met een remote repository kunnen we dit doen via de beide commando's die we reeds gezien hebben:

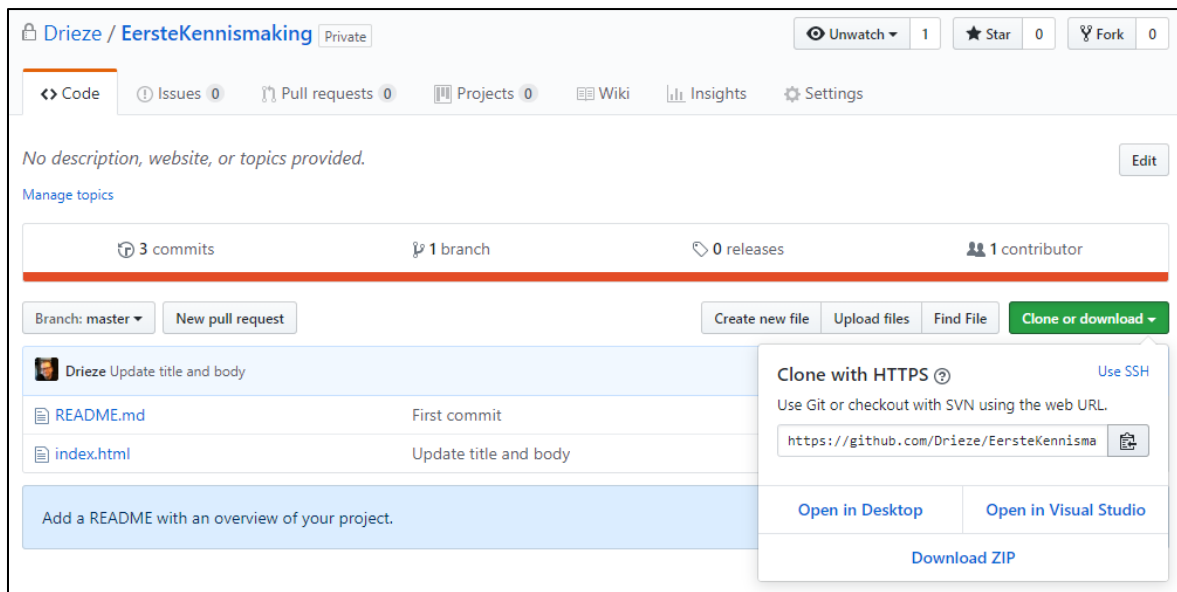
```
git remote add origin https://github.com/JouwGitHubHandle/EersteKennismaking.git
git push -u origin master
```

9.6 CLONEN VAN EEN REMOTE REPOSITORY

Onze remote repository staat nu op GitHub. We kunnen deze nu op elke pc waarop we werken gaan downloaden om er verder aan te werken.

Om dit te demonstreren gooien we onze gemaakte folder EersteKennismaking in de prullenmand van onze computer.

We gaan naar onze remote repository in GitHub en klikken op de groene knop Clone or download en kopiëren de URL die in het tekstvak staat (druk op de knop met het klembord symbool om de URL te kopiëren):



Voor dit voorbeeld wens ik deze repository EersteKennismaking opnieuw op mijn bureaublad te plaatsen, maar de plaats waar jij deze repository wenst te plaatsen speelt in principe geen rol. Klik rechts op mijn bureaublad en kies voor Bash here en in Bash geef ik volgend commando in:

```
git clone https://github.com/JouwGitHubHandle/EersteKennismaking.git
```

Hierdoor wordt er een folder EersteKennismaking (*de naam van de repository*) gemaakt op mijn bureaublad met alle bestanden die te vinden zijn in de online/remote repository.

We hebben dus terug een lokale repository waarmee we op onze computer kunnen verder werken.

Na het maken van de nodige wijzigingen kunnen we de alles opnieuw pushen naar onze remote.

9.7 REMOTE REPOSITORIES VERWIJDEREN

Mocht je het nodig achten om een remote repository te verwijderen dan kan dit via het volgende commando:

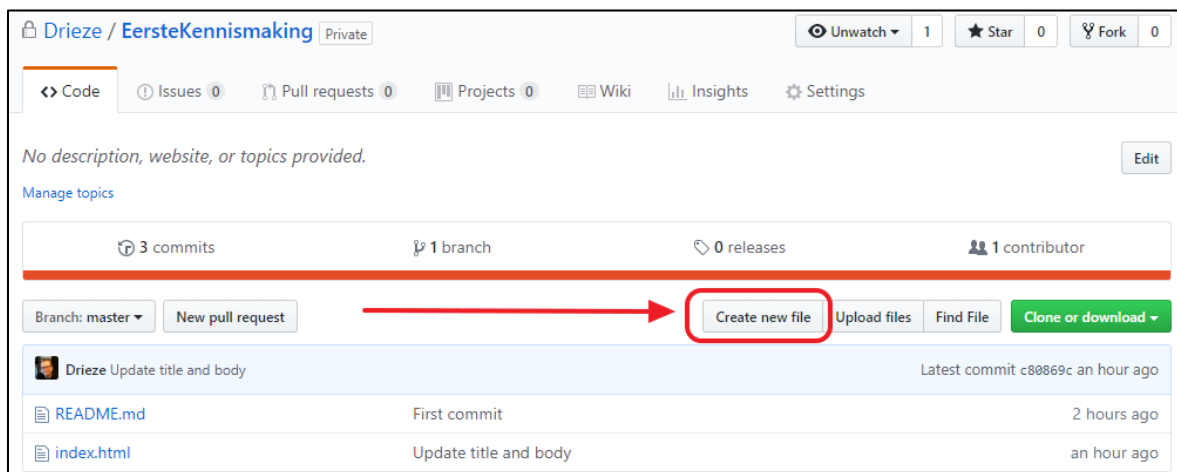
```
git remote rm origin
```

Eventueel vervang je origin door de naam die jij hebt gegeven. **Rm** in het commando staat voor **remove**.

9.8 LOKALE REPOSITORY SYNCHRONISEREN MET JE REMOTE REPOSITORY

Wanneer je samenwerkt met meerdere personen aan één repository kan het zijn dat er wijzigingen zijn doorgevoerd op de remote repository die je nog niet hebt staan op je lokale repository. Om de wijzigingen van de remote repository binnen te halen in je lokale repository kunnen we gebruik maken van het `git pull` commando.

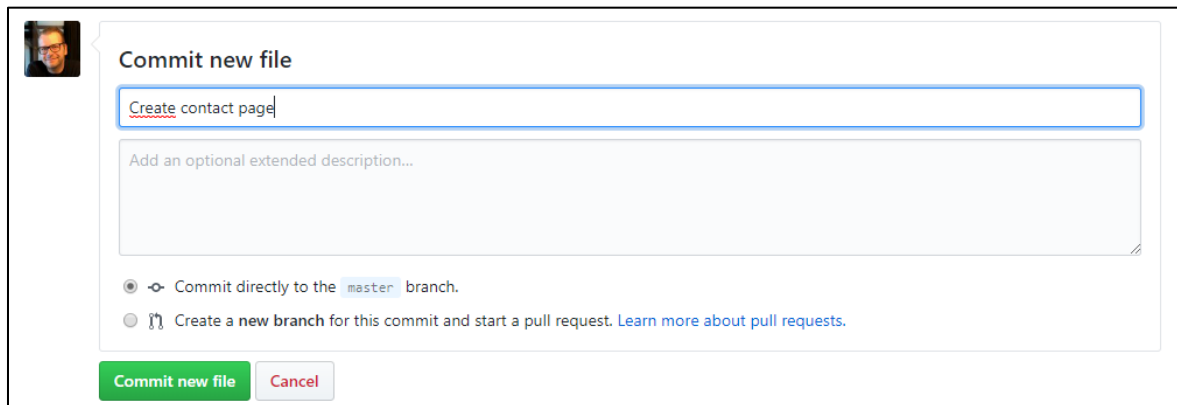
Om dit te simuleren zullen we aan inloggen in GitHub en naar onze repository gaan en klikken op de knop Create new file:



We geven dit bestand de naam **contact.html** en plakken onderstaande HTML code in dit bestand:

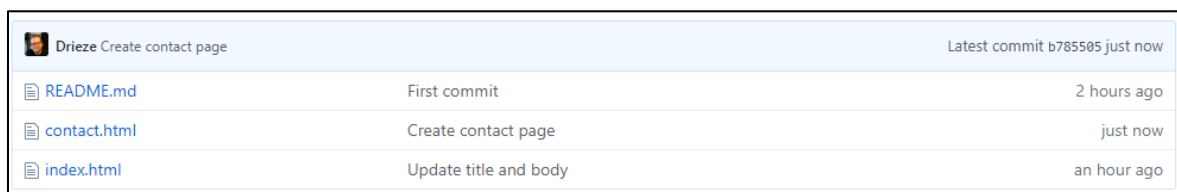
```
<!DOCTYPE html>
<html lang="nl">
  <head>
    <meta charset="UTF-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1.0" />
    <meta http-equiv="X-UA-Compatible" content="ie=edge" />
    <title>Contactpagina met GitHub aangemaakt</title>
  </head>
  <body>
    <h1>Contact</h1>
    <p>
      Deze pagina is in GitHub aangemaakt.
    </p>
  </body>
</html>
```

Onderaan de pagina geven we een titel mee met de commit. We geven als titel “Create contact page”. De description laten we leeg:



We klikken op de knop “Commit new file”.

Onze remote repository ziet er nu zo uit:



Drieze Create contact page		Latest commit b785505 just now
README.md	First commit	2 hours ago
contact.html	Create contact page	just now
index.html	Update title and body	an hour ago

We hebben dit bestand nu zelf aangemaakt via GitHub, maar laten we er eventjes van uitgaan dat dit bestand aangemaakt werd door een andere persoon. Deze persoon heeft **contact.html** aangemaakt in zijn lokale repository en via **git push** zijn de wijzigingen van deze persoon doorgevoerd naar onze remote repository.

Aangezien wij deze wijzigingen nog niet hebben staan in onze lokale repository gaan we er voor zorgen dat onze lokale repository up to date is met de remote repository.

We doen dit via het commando **git pull**. Dit commando zal er voor zorgen dat onze lokale repository up to date is met de remote repository.

Mocht dit niet onmiddellijk lukken dan zal je wat meer informatie moeten meegeven met het commando: **git pull origin master**. Hiermee geven we aan dat we willen synchroniseren met de origin server en met de master branch.

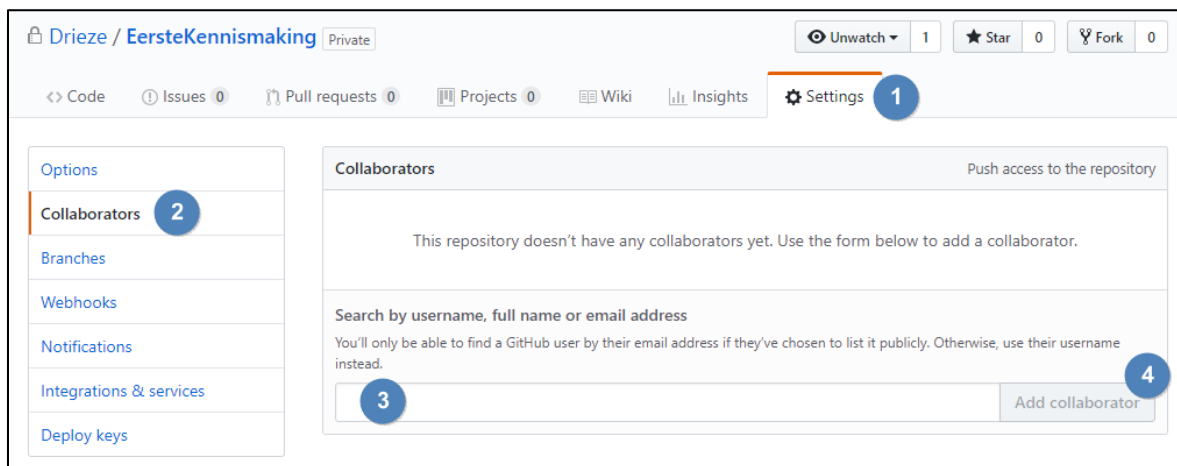
Nu zien we dat ook het **contact.html** bestand in onze lokale repository staat (en dus te vinden is onze map EersteKennismaking)

Het pull commando is een combinatie van een **fetch** (ophalen) en een **merge** (samenvoegen) met je actuele code.

9.9 SAMENWERKEN MET MEERDERE PERSONEN IN ÉÉN REPOSITORY

In de meeste gevallen zal er met een team samengewerkt worden in één repository. Om dit mogelijk te maken kan de eigenaar van de remote repository collaborators gaan toevoegen. Men kan personen toevoegen via volgende manier:

1. Klik op settings
2. Klik op collaborators
3. Tik de username, volledige naam of emailadres van je teamgenoten in
4. Voeg ze toe



Vergeet niet om duidelijke afspraken te maken onderling om zo weinig mogelijk problemen en conflicten tegen te komen!

9.10 EEN AANTAL COMMANDO'S

Commando	Beschrijving
<code>git remote add origin Locatie</code>	Voegt een verwijzing naar een remote repository toe
<code>git push -u origin master</code>	Stuurt je lokale commits (huidige branch) naar de remote repository gelinkt aan de alias origin. De -u optie zorgt er tevens voor dat je huidige branch gelinkt wordt aan de naam van de branch die je meegeeft in dit commando. Dit stelt je in staat om het git push commando hierna te gebruiken zonder verwijzing naar waar je wil pushen. (na de eerste push met de -u optie uiteraard)
<code>git push</code>	Doorvoer van wijziging naar online repository (na een commit)
<code>git pull</code>	Haalt de laatste state op van de remote
<code>git clone Locatie</code>	Maakt een kloon van een repository in een nieuwe map

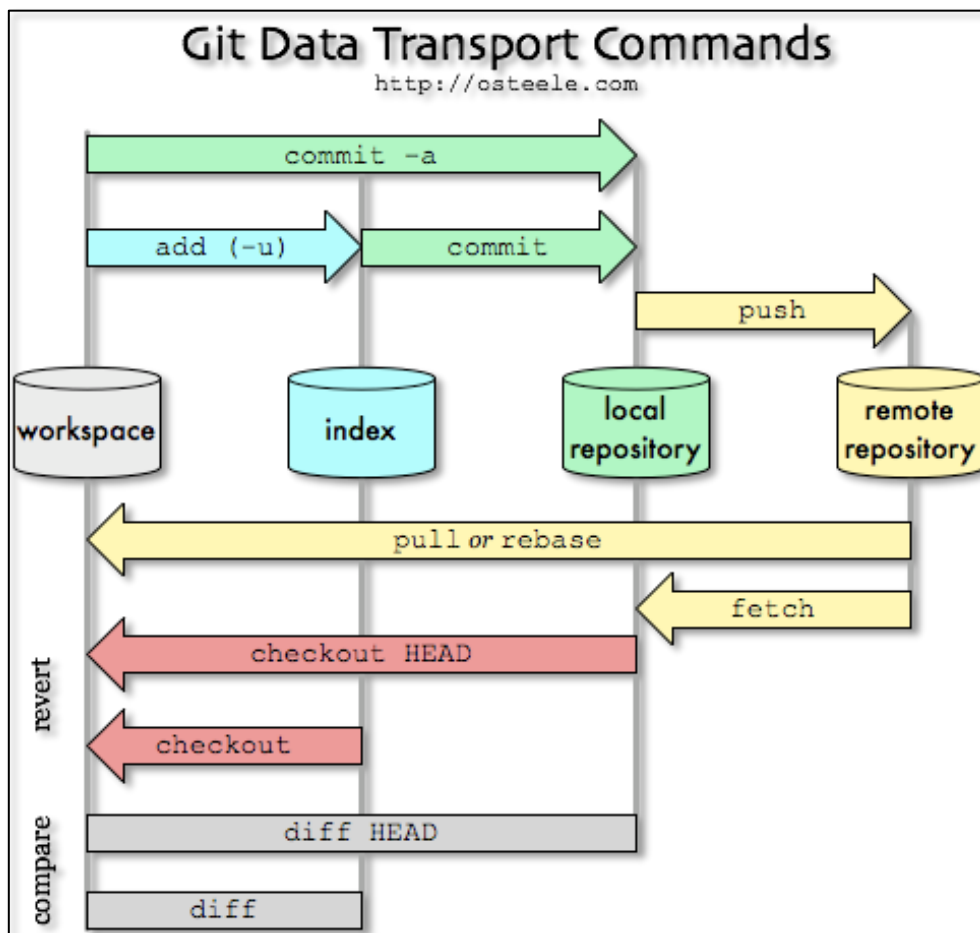
Een overzicht van de basiscommando's vind je op:

<https://confluence.atlassian.com/bitbucketserver/basic-git-commands-776639767.html>

Volgend filmpje geeft enkele extra mogelijkheden:

https://www.youtube.com/watch?v=SWYqp7iY_Tc

- Onderstaand een figuur die grafisch de basisfunctionaliteiten weergeeft:



Wil je zelf even experimenteren : <https://git-school.github.io/visualizing-git/>

