



# Opleiding Applicatie Ontwikkelaar

## Leerlijn Documentatie Versiebeheer & Scrum

### Projecten doen

*Domein D t/m G Level 1*

*Auteur: Erik Mast, Aminah Balfaqih*

*Datum: 19-4-2018*

# Inhoudsopgave

Overzicht .....	4
Voorkennis.....	4
Materialen .....	4
Bronnen .....	4
Instructies .....	4
Einddoelen.....	4
Beschrijving .....	5
Doelen .....	5
Beoordeling .....	5
Versiebeheer .....	6
Inleiding .....	6
GIT voor beginners .....	6
Installatie .....	6
Personaliseren van de installatie.....	7
Je eerste project .....	8
Samenvatting.....	9
Oefening 1: Uitbreiden van je website.....	9
Als er iets niet goed gaat .....	10
Oefening 2: Vergis je eens.....	10
Online bewaren van je werk.....	11
Werken in de online repository.....	13
Oefening 3: Aanpassingen online zetten.....	13
Samenvatting.....	13
Starten op een andere computer .....	14
Oefening 4: Een andere werkplek .....	14
Samenvatting Git Flow .....	15
De laatste loodjes: markeren van een bruikbare versie .....	15
Wat is scrum? .....	16
Een beknopte omschrijving .....	16
Scrum Opdracht.....	17
Opdracht omschrijving .....	17



Eindopdracht .....	19
Nawoord .....	20
Bronnen .....	21
Git .....	21
Scrum .....	21

## Overzicht

Level: Domein D t/m G Level 1  
Duur: Onbepaald  
Methode: Scrum

### Voorkennis

Je moet een product kunnen maken in de taal Java of PHP.

### Materialen

- Je laptop met;
- Editor, bijvoorbeeld Kladblok, Notepad++, Atom of een andere IDE
- Webbrowser, bijvoorbeeld Internet Explorer, Firefox, Chrome
- Account bij Github: <https://github.com/> of een andere Git webprovider

### Bronnen

- Zie bijlage Bronnen

### Instructies

- Je eigen code-repository maken
- Goede gewoontes ontwikkelen
- Deze module bevat ook een hoofdstuk voor de Scrum leerlijn.
- Het is aan te bevelen om deze module op volgorde te bestuderen.

### Einddoelen

- In het vervolg kun je je code bewaren met Git en op GitHub, in alle volgende projecten zal je Git gebruiken.
- Na deze module kun je zelf een backlog maken (functionaliteit beschrijven) en opdelen in userstories, en deze



## Beschrijving

In deze module leer je hoe je je eigen werk veilig kunt stellen. Denk aan per ongeluk bewaren van code die niet werkt, of een laptop die kapot gaat.

Ook ga je leren hoe je op een simpele manier Scrum kunt gebruiken voor een project.

## Doelen

Je kunt straks je eigen code veiligstellen, en op de online repository laten zien dat je dat goed doet. Na deze module zal je ook in staat zijn om scrum uit te voeren op een simpele manier.

## Beoordeling

Deze module wordt beoordeeld aan de hand van een programmeeropdracht, waarbij je kunt laten zien dat je de kennis kunt gebruiken en beheerst, zowel op het gebied van Versiebeheer als Scrum

Na voltooiing van de eindopdracht worden alle domeinen (D t/m G) in een keer afgetekend. Voor zover van toepassing wordt een cijfer toegekend voor alle domeinen.

# Versiebeheer

## Inleiding

Als je aan het programmeren bent is de verleiding groot om alles lekker op je eigen computer te houden. Normaal gesproken gaat dat goed, maar wat als je per ongeluk een heel slecht stukje code bewaart over goede code van gisteren? Of je harde schijf gaat stuk vlak voordat je je opdracht zou willen inleveren? Voor ons als professionals in opleiding staat het stom als we op deze manier een opdracht om zeep helpen. En in de realiteit zal je opdrachtgever dit onacceptabel vinden.

Maar gelukkig gaan we leren hoe je je eigen versies kunt bijhouden zonder problemen. Als bijkomend voordeel heb je dat je altijd een kopie hebt van je werk, en hooguit een beetje werk kwijt bent, mocht er iets vreselijks gebeuren.

## GIT voor beginners

Het antwoord op al dit soort problemen, en meer: GIT.

GIT is een versiebeheer programma. Dat betekent: als je een bestand aanpast en indient bij je repository (bibliotheek) dan zal niet alleen de nieuwe versie bewaard worden, maar ook het oude bestand (en de versie daarvoor etc.).

Even een voorbeeld: je hebt een html met een beetje tekst erin. Vervolgens besluit je dat je andere tekst erin wilt hebben. Dus je vervangt de tekst en bewaart het bestand. Als je dat bestand nu indient bij je repository, zal de wijziging opgeslagen worden, en je oude bestand wordt doorgeschoven naar een soort archief. Bovenop de stapel komt het laatste bestand, en daaronder dus het oude bestand.

## Installatie

Om GIT te kunnen gebruiken moet je wat software installeren in de volgorde zoals hieronder staat.

- GIT
- TortoiseGit (pc) of GitKraken (Multiplatform)

**De links vind je onderaan bij de bronnen.**

### Wat als je werkt op een Mac?

Git kun je het prettigst installeren via Homebrew:

[https://brew.sh/index\\_nl.html](https://brew.sh/index_nl.html)

En daar na in de console deze opdracht uitvoeren: **brew install git**

### Wil je een Git Gui?

Veel Mac gui clients vind je op:

<https://git-scm.com/download/gui/mac>

GitKraken is een goede optie.

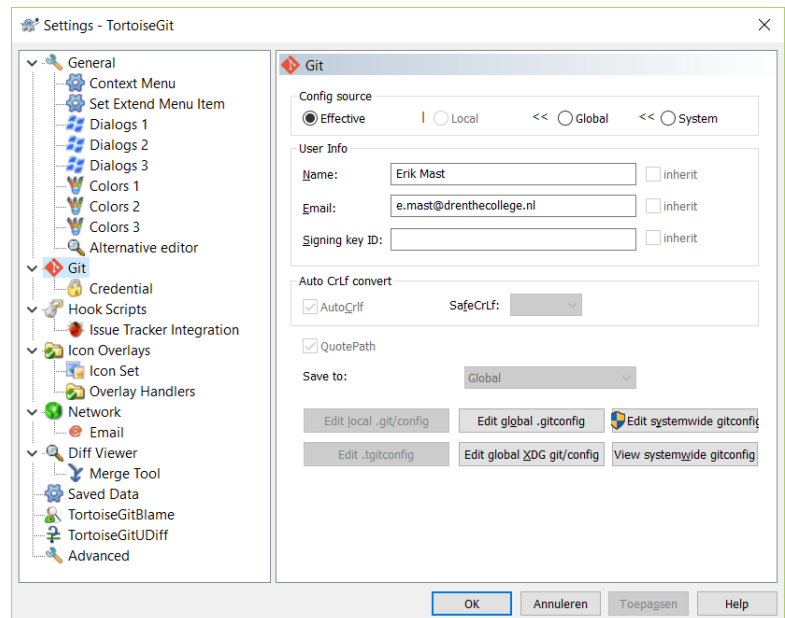
## Personaliseren van de installatie

Een van de dingen die je moet doen is het configureren van je naam en emailadres voor Git. Op dit moment is het het meest praktisch als je daarvoor je dc-email account gebruikt.

### In TortoiseGit

Klik rechts op een map, en kies TortoiseGit->Settings. Je krijgt een scherm zoals rechts, en daar vul je het vakje "Name" en "Email" in.

**Natuurlijk vul je je eigen naam en email in!**



### Vanuit een opdrachtprompt

Open een Opdrachtprompt (die kun je vinden door cmd te zoeken in Windows) en voer de volgende regels uit. **Vul natuurlijk wel je eigen naam en emailadres in!**

```
git config --global user.name "Jouw naam"
```

```
git config --global user.email jouwemailadres@student.drenthecollege.nl
```

## Je eerste project

Allereerst gaan we nu een project onder Git brengen. Om voorzichtig te zijn zullen we niet een bestaand project gebruiken maar iets nieuws bouwen.

### Starten met Git

Eerst maak je een nieuwe werkmapij (working directory). In mijn geval is dat:

C:\xampp\htdocs\opdrachten\E1\GitTest

Ik ga een PHP-website maken, daarom maak ik op deze plek een map. Als je een andere webserver gebruikt komt die map natuurlijk op een andere plek.

Vervolgens ga ik met de verkenner in de map staan, klik rechts op het lege vak en kies "Git Create repository here"

Dit zorgt ervoor dat je begint met een verse repository.

### Wat als je de "Git Create..." optie niet kunt vinden?

Je hebt dan waarschijnlijk niet de GitDesktop geïnstalleerd. Dat is je eigen keus. Een alternatief is om in de Commandprompt deze opdrachten te geven:

```
cd C:\xampp\htdocs\opdrachten\E1\GitTest
git init .
```

De punt betekent overigens "huidige map"

### Wat als je werkt op een Mac?

Als je GitKraken hebt geïnstalleerd heb je waarschijnlijk vergelijkbare opties. En anders kun je in de console naar je map gaan en de laatste van de twee opdrachten ook uitvoeren.

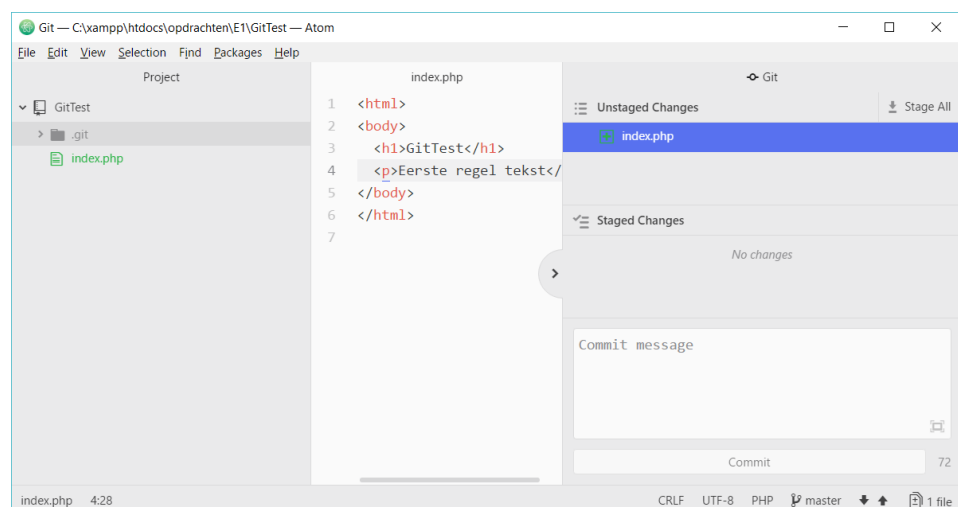
## Eerste bestand in de map maken

Open Atom, en controleer even of je de GitHub package geïnstalleerd hebt (als dat nog niet zo is, doe dat **nu** even).

Vervolgens open je de folder (map) waarin je net je nieuwe repository hebt gemaakt. Maak in die folder een nieuw bestand en werk een standaard simpele html uit. Bewaar dit bestand als "index.php"

Daarna zul je dit zien.

Zie je het rechter tabblad niet, klik even op je map GitTest links en kies daarna Packages->GitHub->Toggle Git Tab (sneltoets Ctrl-Shift-9)





**DOMEIN D T/M G LEVEL 1: PROJECTEN DOEN**

De groene tekst van “index.php” betekent dat je een nieuw bestand hebt, wat je nog niet ingediend hebt bij je repository (commit), rechts zie een tabblad met Git, waarin staat dat je bestand nog niet gestaged is (aangemeld om in te dienen, in TortoiseGit heet dat “unversioned”).

Vervolgens kun je in Atom je bestand “stage”n (In TortoiseGit: add) of toevoegen aan de lijst.

Tot slot kun je het bestand indienen (commit), nadat je er zinvol commentaar bij gezet hebt natuurlijk. Daarna is dat bestand officieel onder versiecontrole van Git.

Aan de linker kant zal je zien dat dat bestand nu zwart geworden is (dat betekent: in de repository, sindsdien niets aan gewijzigd)

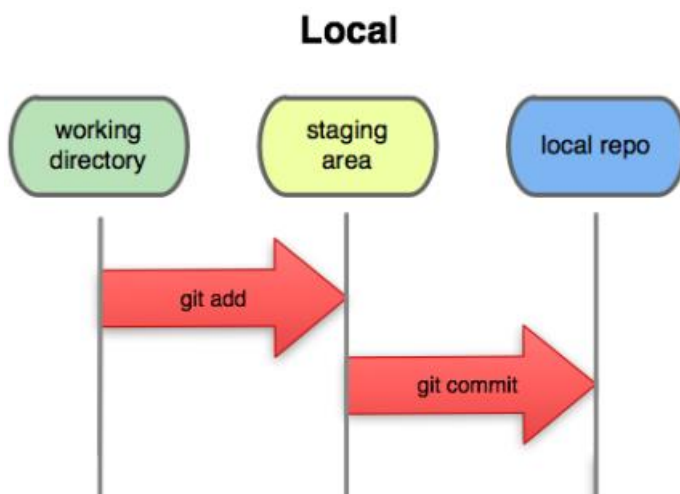
### Tekst aanpassen

Type nu nog eens een regel erbij en bewaar het geheel weer. Je zult zien dat de naam van je bestand oranje wordt. Dat betekent: aangepast sinds de laatste keer dat het bestand is ingediend.

Om deze wijzigingen toe te voegen aan je repository, kun je weer: stage (betekent nu klaarzetten) en commit doen. Commit in TortoiseGit doet dit in één keer.

### Samenvatting

Tot slot, in ons geval ziet de flow er dus zo uit:



### Oefening 1: Uitbreiden van je website

Maak nog minimaal vier html pagina's en een css (waarin je simpele opmaak doet: kleuren en lettergrootte bijvoorbeeld, maak er geen week werk van) en probeer vooral het proces goed te begrijpen.

Als hierover vragen zijn, stel ze vooral!

## Als er iets niet goed gaat

Stel je voor dat je een stuk code hebt ingediend, en dan kom je daarna erachter dat er iets niet goed is. Bijvoorbeeld je code werkt niet meer.

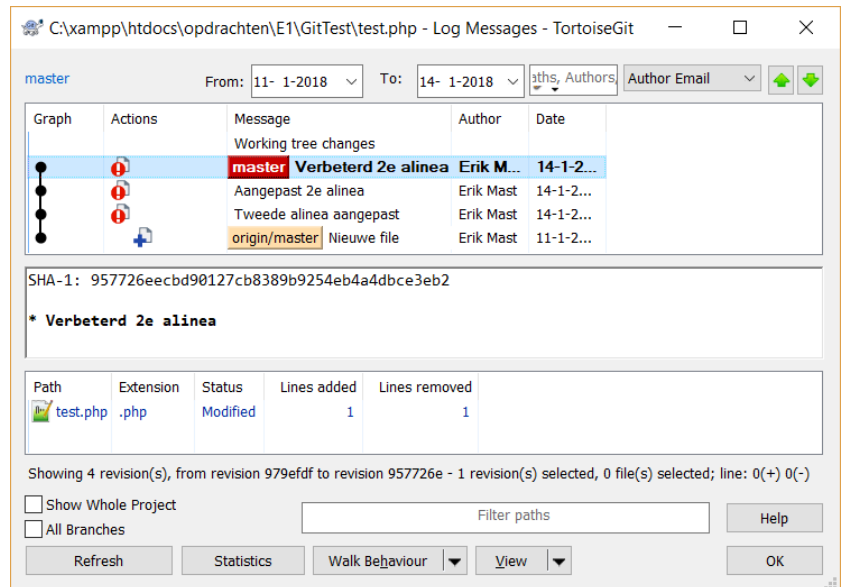
Je kunt dan met TortoiseGit dit aanpassen: klik rechts op het bestand, kies TortoiseGit>Show Log.

Dan krijg je een scherm te zien, waarin je laatste wijziging terug kunt draaien:

En als je daarna met rechts klikt op de regel die hier blauw is, kun je kiezen voor:

### Revert change by this commit

Dat zal de laatste commit ongedaan maken en dat wordt weer als nieuwe commit toegevoegd.



Dit is verwarrend. Bedenk je dat als je aanpassing doet en indient, en daarna nog een aanpassing doet, zal je die ook moeten indienen. En omdat iets wat je ongedaan maakt ook weer een aanpassing is, zal je die ook moeten indienen.

## Oefening 2: Vergis je eens

Ga met de code uit oefening 1 verder. Voeg nog twee pagina's toe en maak expres een vergissing, die je daarna terugdraait. En dan indient (commit) en vervolgens nog een paar aanpassingen doet.

## Online bewaren van je werk

Tot nu toe heb je steeds op je eigen computer gewerkt, en heb je misschien het gevoel dat GIT een soort extra handeling is, waar je niet veel voordeel van hebt. Het voordeel ontstaat eigenlijk als je bijvoorbeeld een fout maakt die je terug wil draaien, of als je eens wilt kijken hoe je het een paar versies geleden hebt gedaan. Dit kun je allemaal zien in TortoiseGit>Show Log of in GitDesktop.

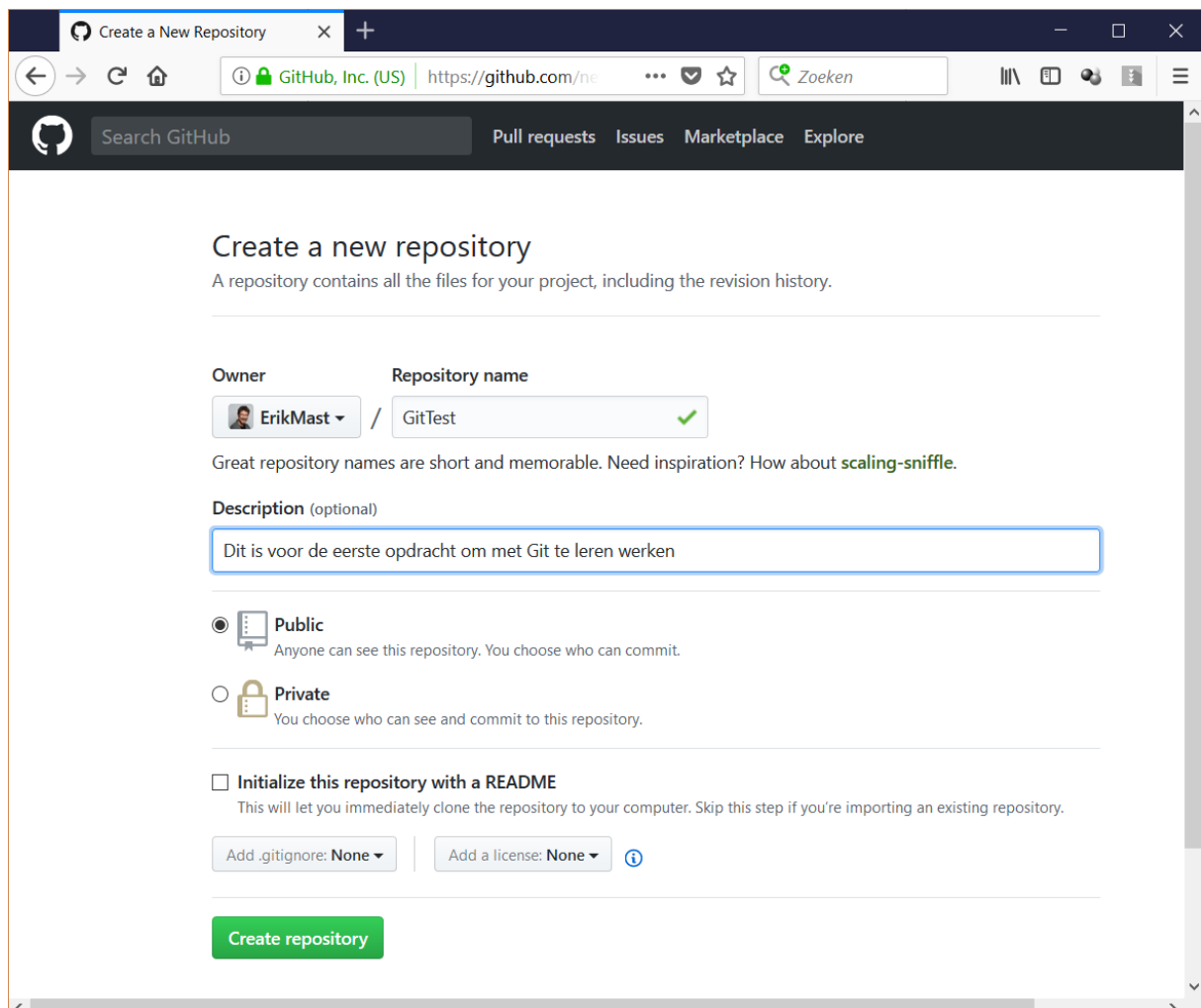
Maar: je code staat nog niet online en loopt nog steeds gevaar als je harddisk stuk gaat.

## Bewaren op GitHub

GitHub is een website die goed samenwerkt met GIT. Maar voor je voordeel hebt hiervan, moet je een account maken.

Dus ga naar **GitHub.com** en maak je eigen account aan. Doe dit met je emailadres van Drenthe College, want straks gaat dat nog handig worden. Vul ook je naam etc. in, dat maakt het zoeken voor anderen gemakkelijker.

Vervolgens maak je eerste repository aan zoals in het plaatje hieronder. Vul de naam in en de Description en druk op "Create Repository"



Create a New Repository

Search GitHub Pull requests Issues Marketplace Explore

### Create a new repository

A repository contains all the files for your project, including the revision history.

Owner: ErikMast / Repository name: GitTest ✓

Great repository names are short and memorable. Need inspiration? How about [scaling-sniffle](#).

Description (optional): Dit is voor de eerste opdracht om met Git te leren werken

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

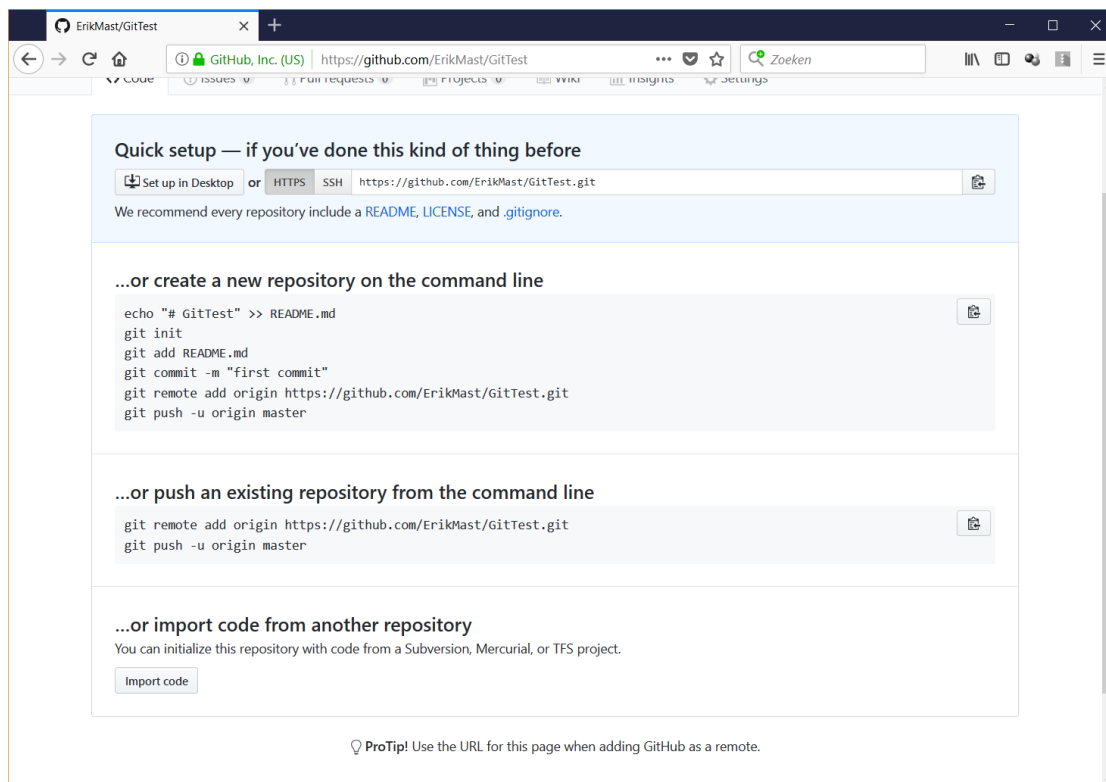
☐ Private  
You choose who can see and commit to this repository.

☐ 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

Daarna kom je in dit scherm:



Rechts van het vakje waar in mijn geval <https://github.com/ErikMast/GitTest.git> staat, staat een knop en als je daarop klikt komt de inhoud van dat vakje op je klembord te staan.

## Code online zetten met TortoiseGit

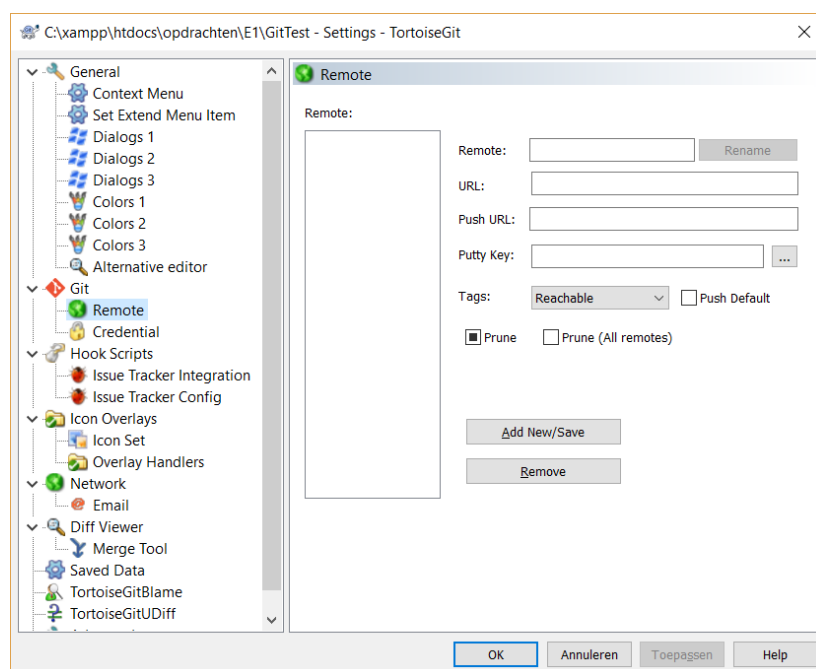
Klik rechts op de map waar je code uit oefening 1 en 2 staat, en kies TortoiseGIT>Settings

Het scherm wat hiernaast staat zal verschijnen en je kiest GIT>Remote.

In het URL-vakje klik je rechts en dan kies je voor plakken, en dan wordt de link naar je repository ingevuld.

Als je op OK drukt vraagt Git of hij fetch moet doen, het antwoord is ja, want dan klopt de administratie weer.

Tot slot kies je voor TortoiseGIT>Push, en die zorgt dat alle commits van jouw project naar GitHub gaan. Daar kun je dat ook zien als je wilt.



## Code online zetten met een commandprompt

Als je geen TortoiseGit hebt kun je het ook zo doen:

Ga naar de map waar je project staat, in mijn geval is dat:

```
C:\xampp\htdocs\opdrachten\E1\GitTest
```

en dan voor je deze twee opdrachten uit, natuurlijk vul je wel je eigen URL in die je op het klembord hebt staan.

```
git remote add origin https://github.com/ErikMast/GitTest.git  
git push -u origin master
```

En tot slot kun je weer gaan kijken op GitHub.

## Werken in de online repository

Als je dit allemaal gedaan hebt kun je al je code ook online bewaren wat handig is als je bijvoorbeeld op twee computers werkt (of in de volgende module met iemand samenwerkt)

Het proces van indienen krijgt een extra opdracht: na elke commit moet je ook **pushen** (opsturen naar de server). In Atom kun je dat doen door rechtsonder op het pijltje omhoog te klikken.

Het is verstandig om altijd te pushen vlak voor je stopt met werken, dan kun je namelijk de volgende keer gemakkelijk weer verder.

Als je begint met werken, moet je dan alleen maar **pull** te doen en je kunt verder. Nu nog alleen op je computer, maar na het hoofdstuk “Starten op een andere computer” ook op je computer thuis (als dat een andere is)

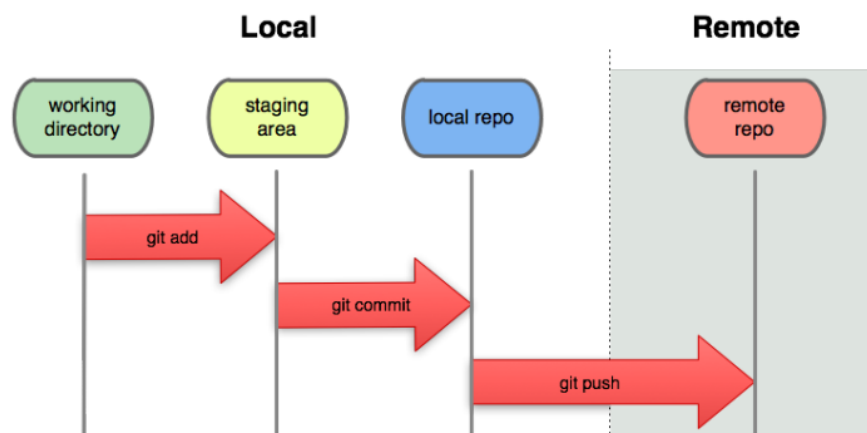
## Oefening 3: Aanpassingen online zetten

Voeg twee bestanden toe aan je project uit oefening 1 en 2, en dien deze ook in samen met aanpassingen in andere bestanden (je zult immers ook links toevoegen naar die nieuwe bestanden).

Doe dat per bestand zodat je het proces ook hier weer goed onder de knie krijgt.

## Samenvatting

Dus nu ziet onze workflow er als volgt uit:



## Starten op een andere computer

Er komt een moment dat je eens op een andere computer wilt of moet werken. Dan doorloop je het proces van installeren van Git en TortoiseGit voor die computer, en misschien ga je ook nog een soort ontwikkelomgeving installeren op die computer.

Uiteindelijk heb je alles klaar staan, en hoeft je alleen maar je code op te halen. Dit proces gaan we nu even nadoen op je eigen computer.

## TortoiseGit

Met de verkenner ga je in naar bijvoorbeeld C:\xampp\htdocs\opdrachten\E1.

Je klikt rechts op Git Clone. Als URL vul je in:

`https://github.com/ErikMast/GitTest.git`

En als Directory vul je in

`C:\xampp\htdocs\opdrachten\E1\GitTest2`

Daarna druk je op "OK" en dan wordt de hele geschiedenis van de repository opgehaald en in GitTest2 neergezet. Daarna kun je daar dus ook werken.

## Commandprompt

Op de commandprompt doe je deze opdracht:

```
git clone https://github.com/ErikMast/GitTest.git GitTest2
```

Dit betekent dat je een kloon maakt van de hele geschiedenis van de repository, de link is weer je link naar je GitHub repository, en het laatste woord is de nieuwe map waar het in komt.

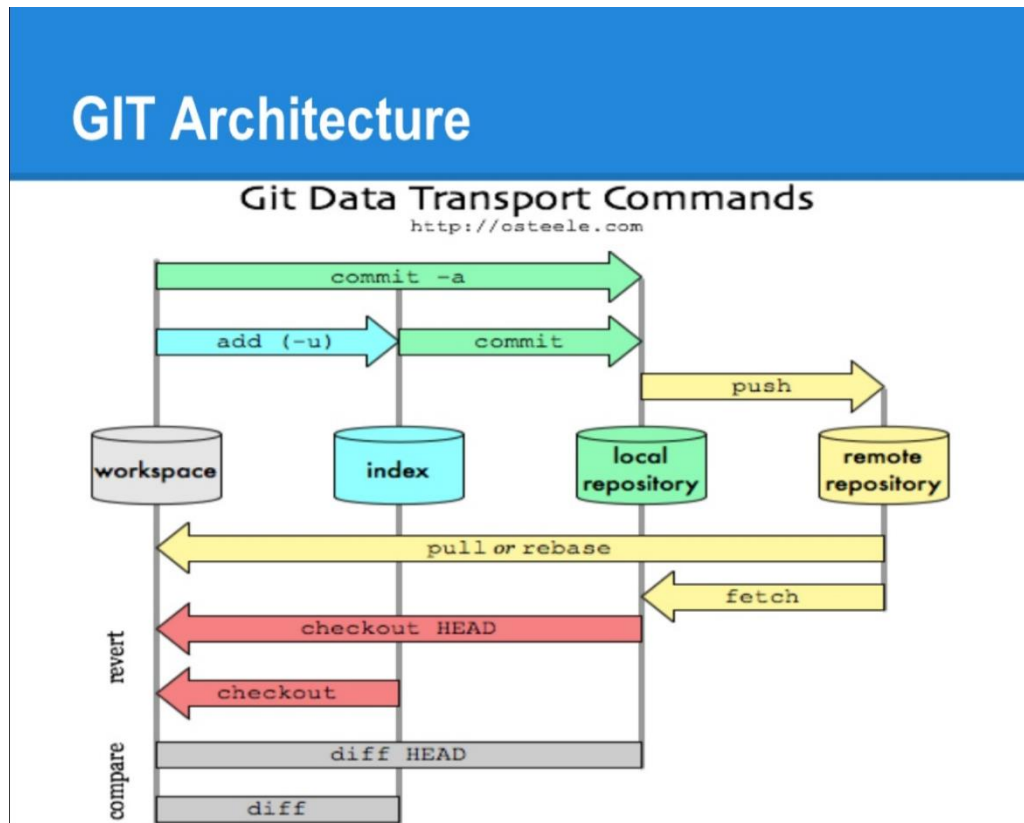
In mijn geval staat hij dus in dezelfde map als GitTest.

## Oefening 4: Een andere werkplek

Ga nu in de GitTest2 map een paar aanpassingen doen, en stuur die ook op naar de server. Zet in het commentaar erbij dat je dat vanaf een andere plek doet, dat maakt het inzichtelijk voor het proces.

## Samenvatting Git Flow

Zo ziet onze workflow er nu uit. Een aantal opdrachten van Git zijn niet behandeld, die je zal pas gebruiken als je het beter snapt en er wat langer mee gewerkt hebt.



## De laatste loodjes: markeren van een bruikbare versie

In de eindopdracht gaat gevraagd worden om elke cyclus van ontwikkeling (oftewel "Bruikbare tussenversie") van je opdracht te markeren met een Tag. Dat kun je doen op de commandline in de map van je project.

```
git tag -a v1.4 -m "my version 1.4"
```

Of met TortoiseGit met het commando "Create Tag"

Deze tags kun je terugvinden in de folder waarin je werkt in de log van TortoiseGit.

## Wat is scrum?

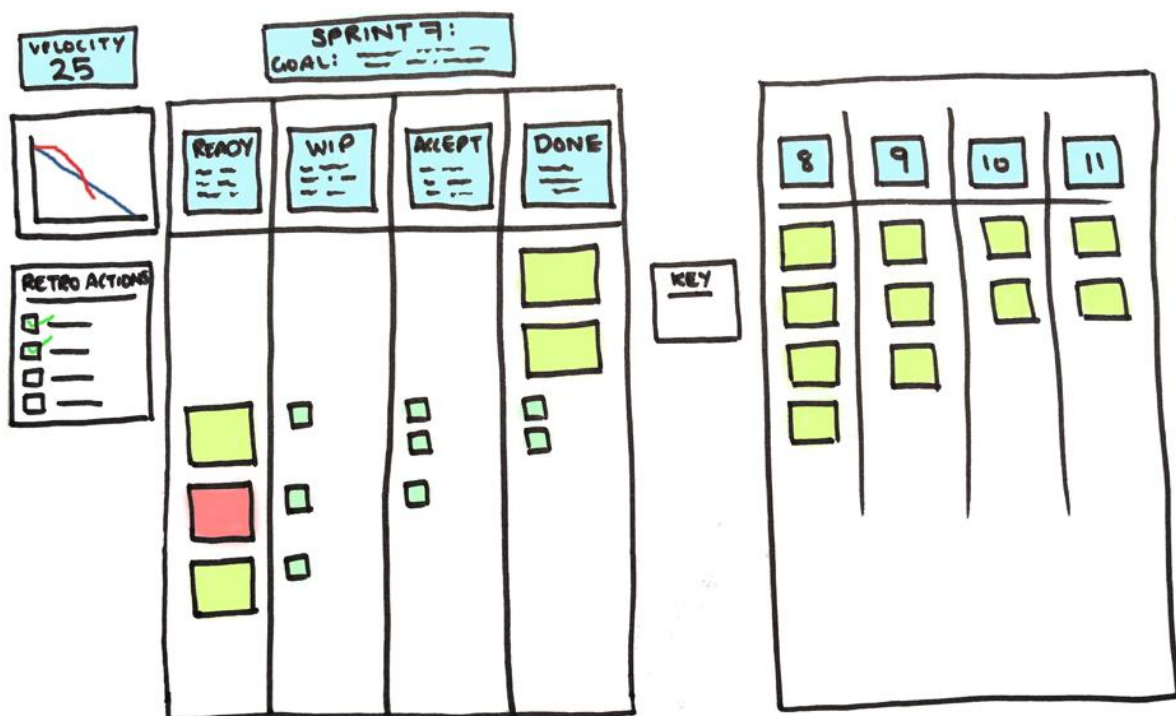
### Een beknopte omschrijving

Scrum is een methodiek die je kunt gebruiken om projecten uit te voeren. In de IT wereld wordt scrum al intensief gebruikt. Het geeft duidelijkheid over wat er gedaan moet worden en de opdrachtgever is altijd op de hoogte. De projectgroep werkt in kleine stapjes (sprints) en test tussendoor zijn product. De groep gaat dan ook pas verder als een onderdeel afgesloten is en de groep toe is aan de volgende stap. Hiervoor wordt een definition of done geschreven, zodat voor iedereen duidelijk is wanneer is werkelijk afgerond is.

De projectgroep levert deelproducten op. Dit kan snel gaan en er kan op elk moment bijgestuurd worden.

Scrum maakt samenwerking helder en transparant.

Bij scrum wordt er gebruik gemaakt van een scrumboard.



Hierop staan alle taken die uitgevoerd moeten worden en ook door wie deze uitgevoerd wordt. Elke taak heeft een bepaalde status, nl. afgerond, work in progress, acceptatie en afgerond.

Als alle onderdelen het gebied "done" bereikt heeft dan is de sprint voorbij en kan er een nieuwe sprint gepland worden.



## Scrum Opdracht

*Om level 1 van scrum te kunnen halen is het van belang dat je voor jezelf een scrum board bij gaat houden. Hiervoor kan je github gebruiken. Bij het behalen van dit leven heb je automatisch level 1 van samenwerking ook behaald.*

### Opdracht omschrijving

Jouw opdracht voor dit level is het maken van een forum. De volgende eisen zijn van toepassing:

- Je moet een onderwerp kunnen aanmaken
- Andere moeten op dit onderwerp kunnen reageren
- Je moet hoofdthema's hebben (bijvoorbeeld games, lifestyle etc)
- Elke onderwerp moet onder een thema geplaatst worden

### Opdracht A

Zorg dat je eerst duidelijk hebt wat er moet gebeuren en schrijf dit op. Dit wordt jouw product backlog. Let erop dat als je iets moet bestuderen dat dit ook een onderdeel is van je activiteiten. Schrijf dit in een document en noem het document product backlog.

Uitleg: <http://scrumguide.nl/product-backlog/>

### Opdracht B

Maak daarna user stories van jouw product backlog. En bepaal de definition of ready.

Uitleg:

<http://scrumguide.nl/user-story/>

<http://www.fourpoints.nl/blog/de-definition-of-ready-helps-scrumteams-voorwaarden-te-stellen>

### Opdracht C

Geef prioriteiten aan in de backlog. Maak daarna groepjes van de werkzaamheden. Groepeer onderdelen die iets met elkaar te maken hebben. Deze groepjes zijn epics. Maak nu een sprint backlog voor jouw eerste sprint.

Uitleg: <http://scrumguide.nl/sprint-backlog/>



### Opdracht D

Bepaal daarna voor elk onderdeel wanneer iets echt afgerond is. Dit wordt de definition of done.

Uitleg: <http://scrumguide.nl/definition-of-done/>

### Opdracht E

Zet nu alles voor jouw sprint op een digitaal scrumboard. En houd je vorderingen bij via dit bord.

Uitleg: <http://scrumguide.nl/scrumbord/>

## Eindopdracht

Je gaat het Forum wat je bedacht hebt nu ook bouwen in HTML en CSS. Dit ga je doen met Scrum (Je gaat alle taken die je ingevoerd hebt in GitHub gebruiken), en onder versiebeheer met Git (waarmee je alle aanpassingen gaat bijhouden in GitHub). Benodigde materialen kun je downloaden vanaf Magister.

Criteria zijn :

- Een forum **zoals je hebt gedefinieerd** in de opdrachten A t/m E. Uitgewerkt in HTML en CSS (dus alleen zoals het er uit zou moeten zien in combinatie met het onderstaande ontwerp)
- Een Scrumboard waaraan gezien kan worden dat de taken opgedeeld zijn (geweest) en een simpele Scrum cyclus hebben doorlopen.
- Alle taken zijn opgedeeld in minimaal 2 sprints, je kunt dit ook laten zien d.m.v. een tag in GitHub. Als je die tag gezet hebt, laat je ook kort het forum aan de docent zien.
- Zinnige commentaren bij het committen van de code.
- **Niet** aan het eind het eindproduct committen zonder tussenstappen.
- Er wordt regelmatig HTML/CSS code gecommit. Het ontwikkelproces moet inzichtelijk zijn.





## Nawoord

In het vervolg ga je al je projecten doen met Git. Oefening is noodzakelijk, en werken met Git is iets wat je waarschijnlijk heel lang gaat doen. Als je deze module hebt afgerond, wordt ook niet geaccepteerd dat je je code niet bijhoudt met Git, daar zullen minpunten voor worden afgetrokken.

Ook verwachten we een basale Scrum opzet bij elk volgend project. Dus dat je voor elk project de stappen van backlog, userstories, en prioritering ervan doorloopt.

## Bronnen

### Git

- ✓ **Download van Git**  
<https://git-scm.com/downloads>
- ✓ **Download van TortoiseGit (PC)**  
<https://tortoisegit.org/download/>
- ✓ **Download van GitKraken (Multiplatform)**  
<https://www.gitkraken.com/>
- ✓ **Gource: tool om je commits te bekijken als film**  
<http://gource.io/>
- ✓ **GitHub: een online service waar je je code kunt beheren**  
<https://github.com/>

### Scrum

- ✓ <http://scrumguide.nl/product-backlog/>
- ✓ <http://scrumguide.nl/user-story/>
- ✓ <http://www.fourpoints.nl/blog/de-definition-of-ready-helpt-scrumteams-voorwaarden-te-stellen>
- ✓ <http://scrumguide.nl/sprint-backlog/>
- ✓ <http://scrumguide.nl/definition-of-done/>
- ✓ <http://scrumguide.nl/scrumbord/>