



samen sterk voor werk



Deze cursus is eigendom van VDAB Competentiecentra ©

Peoplesoftcode:

Wettelijk depot:

versie: 13/3/2015



## INHOUD

1	Inleiding .....	4
1.1	Doelstelling.....	4
1.2	Versiebeheersysteem.....	4
1.3	Repository .....	4
1.3.1	Private repository – public repository .....	4
1.3.2	Push - pull .....	4
1.3.3	Een dag van een ontwikkelaar met git (van links naar rechts) .....	5
1.3.4	De inhoud van de repository .....	5
1.4	Git installeren .....	5
1.5	Git gebruiken .....	5
1.6	Algemene instellingen .....	5
1.7	Help .....	5
2	Basishandelingen in de repository.....	6
2.1	De repository maken.....	6
2.2	De status van de repository .....	6
2.3	Een bestand toevoegen aan de repository .....	6
2.3.1	Het bestand toevoegen aan de staging area .....	7
2.3.2	De staging area committen in de repository .....	7
2.3.3	Samenvatting .....	7
2.4	Meerdere bestanden toevoegen aan de staging area .....	7
2.5	Een directory toevoegen aan de staging area.....	8
2.6	Alle bestanden toevoegen aan de staging area .....	8
2.7	Een bestand wijzigen.....	8
2.8	Een bestand hernoemen .....	8
2.9	Een bestand verwijderen .....	9
2.10	Bestanden en directories negeren .....	9
3	Een historiek van het project.....	10
4	Aliassen .....	11
5	Een vorige versie van het project terug inzien.....	12
5.1	Bestanden in de huidige versie overschrijven vanuit een vorige versie .....	12
5.2	Tag .....	13
6	Branches.....	14
6.1	Een branch maken.....	14

6.2	Een overzicht van de branches .....	14
6.3	Een branch verwijderen .....	14
6.4	Een branch activeren .....	14
6.5	Branches mergen .....	15
6.6	Merge conflicten .....	15
6.6.1	Een wijziging zonder conflict .....	15
6.6.2	Een wijziging met een conflict .....	16
7	GitHub: een public repository .....	17
7.1	Een public repository aanmaken .....	17
7.2	De public repository kenbaar maken .....	17
7.3	Push .....	17
7.4	Clone .....	17
7.5	Fetch .....	18
7.6	Pull .....	18
7.7	Conflict .....	18
8	Git grafisch .....	20
8.1	Bestanden toevoegen aan de staging area .....	20
8.2	De staging area committen .....	20
8.3	Een bestandswijziging committen .....	20
8.4	Een branch maken .....	20
8.5	Een branch actief maken .....	20
8.6	Een branch mergen .....	21
8.7	Een public repository kenbaar maken .....	21
8.8	Push .....	21
8.9	Clone .....	21
8.10	Pull .....	21
9	Eclipse integratie .....	22
9.1	Het project maken .....	22
9.2	Een repository maken in het project .....	22
9.3	De staging area .....	22
9.4	Lijstje van de commits .....	23
9.5	Checkout .....	23
9.6	Branches .....	23
9.7	Push – Pull .....	24
10	NetBeans integratie .....	25

---

10.1	Het project maken.....	25
10.2	Een repository maken in het project.....	25
10.3	De staging area .....	25
10.4	Lijstje van de commits .....	25
10.5	Checkout.....	25
10.6	Branches .....	25
10.7	Push – Pull .....	26
11	Visual Studio integratie .....	27
11.1	Het project maken.....	27
11.2	Een repository maken in het project.....	27
11.3	Algemene instellingen .....	27
11.4	De staging area .....	27
11.5	Lijstje van de commits .....	27
11.6	Branches .....	27
11.7	Push – Pull .....	28

# 1 Inleiding

## 1.1 Doelstelling

Je leert in deze cursus werken met GIT, een versiebeheersysteem.

## 1.2 Versiebeheersysteem

Een project is een applicatie die je maakt.

Je houdt in een versiebeheersysteem alle wijzigingen bij van project sources.

Het Engelse woord voor versiebeheersysteem is version control system (VCS).

Je kan vorige versies van je project terugvinden in het versiebeheersysteem.

Je kan zo een verkeerde source aanpassing ongedaan maken.

Je werkt meestal niet alleen aan een project, maar met een team.

Meerdere teamleden bewaren wijzigingen van sources in het versiebeheersysteem.

Als twee leden dezelfde source wijzigen en proberen te bewaren in het versiebeheersysteem, zorgt het versiebeheersysteem ervoor dat de wijzigingen van het ene lid niet de wijzigingen van het andere lid overschrijven. Je ziet verder in de cursus hoe dit gebeurt.

Er bestaan meerdere versiebeheersystemen: Git, Subversion, Mercurial, ...

Je leert in deze cursus werken met Git.


## 1.3 Repository

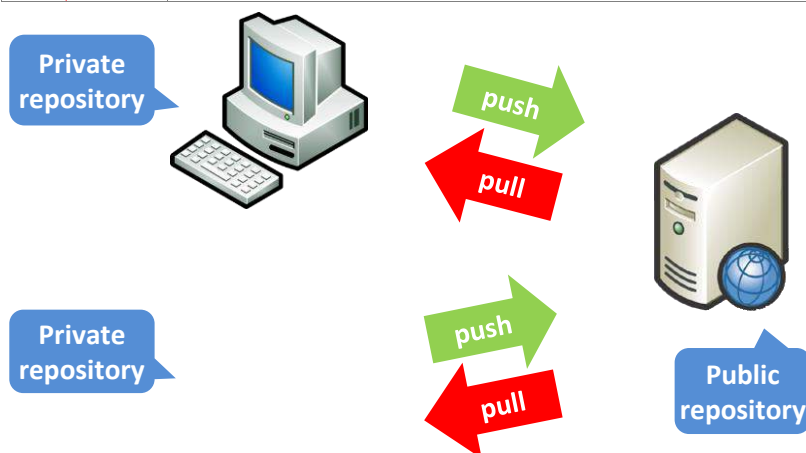
De repository is de plaats waar Git versies van je project bijhoudt.

### 1.3.1 Private repository – public repository

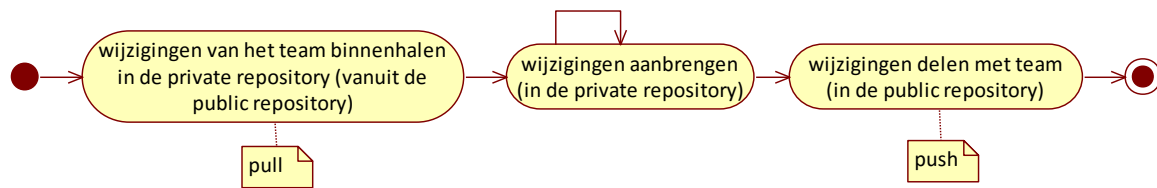
- Private repository De repository op de individuele PC van een teamlid
- Public repository De repository op een server die alle teamleden delen

### 1.3.2 Push - pull

	Een teamlid bewaart gewijzigde sources initieel in zijn private repository. Achteraf deelt hij deze wijzigingen met de andere teamleden: hij brengt de wijzigingen van zijn private repository over naar de public repository.
	Het teamlid brengt op regelmatige tijdstippen de wijzigingen van andere teamleden over van de public repository naar zijn eigen private repository



### 1.3.3 Een dag van een ontwikkelaar met git (van links naar rechts)



### 1.3.4 De inhoud van de repository

De repository bevat alle onderdelen, waarmee je het project vanaf nul opbouwt

- Sources (.java, .php, .cs, .js, ...)
- CSS bestanden
- Afbeeldingen
- Configuratiebestanden
- Documentatie

De repository bevat geen tijdelijke bestanden (.tmp) of bestanden die je kan maken op basis van andere bestanden. De repository bevat bijvoorbeeld geen gecompileerde bestanden in de repository: je kan die aanmaken op basis van de sources.

## 1.4 Git installeren

Je downloadt het Git installatieprogramma vanaf <http://git-scm.com/download/win>

Je voert dit installatieprogramma uit. Als je op de stap Adjusting your PATH environment komt, kies je Run Git from the Windows Command Prompt

Bij de andere keuzes moet je soms Linux opdrachten tikken in een Linux emulator.

## 1.5 Git gebruiken

- Je kan Git gebruiken door opdrachten in te tikken op de command prompt.
- Je kan Git ook gebruiken met een GUI tool, meegeleverd met Git
- Git is ook geïntegreerd in veel IDE's (Eclipse, NetBeans, Visual Studio, ...)

Je leert in deze cursus de verschillende werkwijzen kennen.

## 1.6 Algemene instellingen

Ieder teamlid moet eenmalig twee instellingen wijzigen in zijn Git configuratie

Je voert daartoe volgende opdrachten uit aan de command prompt

```
git config --global user.name "TikHierJeNaam"
```

```
git config --global user.email "TikHierJeEmailAdres"
```

## 1.7 Help

Je vraagt van een opdracht help door deze opdracht te volgen met **--help**.

Je vraagt bijvoorbeeld help over de opdracht **config** met **git config --help**

## 2 Basishandelingen in de repository

### 2.1 De repository maken

Ieder teamlid heeft op zijn computer een directory met het project. Deze directory heet de “working directory”.

Het teamlid maakt eenmalig een private repository in deze directory. Git bewaart in deze repository alle projectwijzigingen.

1. Je maakt een working directory voor een voorbeeldproject “recepten”  
`md c:\recepten`
2. Je plaatst je in deze directory  
`cd c:\recepten`
3. Je maakt de repository met de opdracht  
`git init`

Git maakte een verborgen directory `.git` in de working directory.

Git gebruikt deze verborgen directory als repository.

### 2.2 De status van de repository

Je vraagt de status van de repository met

`git status`

Je ziet in de statusmeldingen **nothing to commit**.

Committer betekent source wijzigingen bewaren in de repository.

Gezien het huidige project nog geen sources bevat, is er ook niets te committer.

### 2.3 Een bestand toevoegen aan de repository

Je maakt in de directory `c:\recepten` het bestand `readme.txt` (bvb. met NotePad)

**Dit project is een website met lekkere recepten.  
Per recept worden ingrediënten en werkwijze vermeld.**

Je voegt in twee stappen een bestand toe aan de repository

1. Je voegt de source toe aan de staging area (een voorbereidende ruimte)
2. Je commit de staging area naar de repository

Je vraagt eerst de status van de repository

`git status`

Je ziet de melding **nothing added to commit but untracked files present**

Untracked files zijn bestanden die zich nog niet in de staging area bevinden.

Gezien de staging area leeg is zie je ook **nothing added to commit**

Je ziet **readme.txt** vermeld bij **Untracked files**.

**readme.txt**

bevindt zich in de  
working directory:



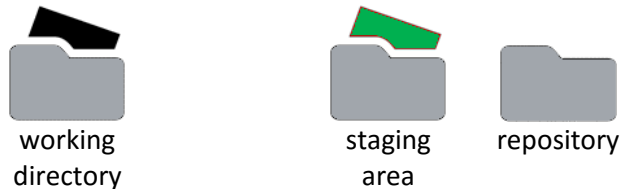


### 2.3.1 Het bestand toevoegen aan de staging area

1. Je voegt **readme.txt** toe aan de staging area  
`git add readme.txt`
2. Je vraagt de status van de repository  
`git status`  
Je ziet bij **new file** het bestand dat zich in de staging area bevindt: **readme.txt**

**readme.txt**

bevindt zich nu ook in de staging area:



### 2.3.2 De staging area committen in de repository

1. Je commit de staging area in de repository met de opdracht **commit**.  
Je geeft als verplichte **-m** parameter een reden mee voor deze commit.  
`git commit -m "eerste commit"`  
Je ziet in de meldingen **1 file changed, 2 insertions(+)**  
Dit betekent: de repository bevat één bestandswijziging, bestaande uit 2 regels
2. Je vraagt daarna de status van de repository.  
`git status`  
Je ziet in de meldingen **nothing to commit, working directory clean**  
Dit betekent dat alle bestandswijzigingen bewaard zijn in de repository

**readme.txt**

bevindt zich nu ook in de repository:



### 2.3.3 Samenvatting



## 2.4 Meerdere bestanden toevoegen aan de staging area

1. Je kopieert **index.html**, **pannenkoeken.html**, **oliebollen.html** en **wafels.html** van het oefenmateriaal naar de directory **c:\recepten**
2. Je vraagt de status van de repository  
`git status`  
De HTML bestanden staan vermeld als **untracked**: ze bevinden zich nog niet in de staging area.
3. Je voegt deze bestanden toe aan de staging area  
`git add *.html`
4. Je commit de staging area naar de repository  
`git commit -m "met HTML bestanden"`

## 2.5 Een directory toevoegen aan de staging area

1. Je maakt in `c:\recepten` een directory `images`
2. Je kopieert `oliebollen.jpg`, `pannenkoeken.jpg` en `wafels.jpg` van het oefenmateriaal in deze directory
3. Je vraagt de status van de repository  
`git status`  
De directory `images` is vermeld als untracked
4. Je voegt deze directory (met zijn bestanden) toe aan de staging area  
`git add images`
5. Je commit de staging area naar de repository  
`git commit -m "met images"`

## 2.6 Alle bestanden toevoegen aan de staging area

1. Je kopieert `favicon.ico` van het oefenmateriaal naar `c:\recepten`
2. Je voegt alle bestanden, die zich nog niet in de staging area bevinden, toe aan de staging area.  
Git neemt hierbij ook bestanden mee uit subdirectory's.  
`git add .`
3. Je commit de staging area naar de repository  
`git commit -m "met favicon.ico"`

## 2.7 Een bestand wijzigen

1. Je wijzigt in `readme.txt` de laatste zin naar  
**Per recept worden ingrediënten, foto en werkwijze vermeld.**
2. Je vraagt de status van de repository  
`git status`  
`readme.txt` is vermeld als modified
3. Je voegt `readme.txt` toe aan de staging area  
`git add readme.txt` of `git add .`
4. Je commit de staging area naar de repository  
`git commit -m "readme.txt aangepast"`  
Je ziet in de meldingen **1 file changed, 1 insertion(+), 1 deletion(-)**

Git heeft in de repository bewaard dat één bestand gewijzigd werd.

Een oude regel is uit het bestand verwijderd (**1 deletion**)

en een nieuwe (vervangende) regel is aan het bestand toegevoegd (**1 insertion**).

## 2.8 Een bestand hernoemen

Je hernoemt een bestand in de working directory met de opdracht `mv`.

Deze hernoeming komt op dat moment ook in de staging area terecht.

1. Je hernoemt `favicon.ico` naar `icon.ico`  
`git mv favicon.ico icon.ico`
2. Je vraagt de status van de repository  
`git status`  
Je ziet **renamed: favicon.ico -> icon.ico**
3. Je commit de staging area naar de repository  
`git commit -m "favicon.ico hernoemd naar icon.ico"`

## 2.9 Een bestand verwijderen

Je verwijdert een bestand in de working directory met de opdracht `rm`. Deze verwijdering komt op dat moment ook in de staging area terecht

1. Je verwijdert `icon.ico`  
`git rm icon.ico`
2. Je vraagt de status van de repository  
`git status`  
Je ziet **deleted:**     `icon.ico`
3. Je commit de staging area naar de repository  
`git commit -m "icon.ico verwijderd"`

## 2.10 Bestanden en directories negeren

Je voegt met de handige opdracht `git add .` alle bestanden, die zich nog niet in de staging area bevinden, toe aan de staging area.

Het is hierbij vervelend dat bestanden in de staging area terechtkomen die je niet in de repository wil, zoals tijdelijke bestanden (`*.tmp`).

Je verhindert dit door dit soort bestanden te vermelden in een bestand `.gitignore`. Git houdt geen rekening met de bestanden vermeld in dit bestand.

1. Je kopieert `.gitignore` naar `c:\recepten`  
Dit bestand heeft volgende inhoud

```
*.tmp  
*.temp
```

2. Je kopieert `tijdelijk.tmp` van het oefenmateriaal naar `c:\recepten`
3. Je vraagt de status van de repository  
`git status`  
Je ziet dat **.gitignore** wél vermeld is, **tijdelijk.tmp** niet.
4. Je voegt `.gitignore` toe aan de staging area  
`git add .`
5. Je commit de staging area naar de repository  
`git commit -m ".gitignore toegevoegd"`

### 3 Een historiek van het project

Git heeft opdrachten waarmee je de historiek (commits) van het project opvraagt.

Je ziet de historiek met de opdracht **log**

**git log**

Je ziet per commit

- de unieke identifier van de commit (bvb. `cf295ceb5f540247394f84ac0f2599915905e18d`)
- de naam en het email adres van de persoon die de commit uitvoerde
- de datum en de tijd van de commit
- de reden van de commit

Je verlaat dit overzicht met de toets **q**

Je ziet een verkort historiek overzicht met de opdracht

**git log --pretty=oneline**

Je ziet per commit één regel met

- de unieke identifier van de commit (bvb. `cf295ceb5f540247394f84ac0f2599915905e18d`)
- de reden van de commit

Je kan ook meegeven dat je niet alle commits wil zien

- **git log --pretty=oneline --max-count=3**  
Je ziet enkel de laatste drie commits.
- **git log --pretty=oneline --since="1 hour ago"**  
Je ziet enkel de commits van het laatste uur.
- **git log --pretty=oneline --until="1 hour ago"**  
Je ziet alle commits, behalve die van het laatste uur.
- **git log --pretty=oneline --author="Joe Dalton"**  
Je ziet enkel de commits van de person Joe Dalton.

Je kan de lay-out van één lijn in de historiek personaliseren

**git log --pretty=format:"%h %ad %s (%an)" --date=short**

- **%h**  
Dit is de afgekorte hash (unieke identifier) van de commit
- **%ad**  
Dit is de datum van de commit
- **%s**  
Dit is de reden van de commit
- **%an**  
Dit is de naam van de persoon die de commit uitvoerde
- **--date=short**  
Je geeft aan dat Git de datum in korte notatie moet tonen.

## 4 Aliassen

Sommige opdrachten vragen veel tikwerk.

Je kan een lange opdracht associëren met een korte alias.

Je voert daarna de lange opdracht uit door de korte alias in te tikken.

Je associeert de lange opdracht

```
git log --pretty=format:"%h %ad %s (%an)" --date=short
```

met de korte alias

**historiek**

```
git config --global alias.historiek "log --pretty=format:'%h %ad %s (%an)' --date=short"
```

Je kan de lange opdracht vanaf nu uitvoeren met

```
git historiek
```

## 5 Een vorige versie van het project terug inzien

Je kan een vorige versie (commit) van het project terug *inzien* met de opdracht **checkout**. De laatste versie van je project blijft hierbij *ongewijzigd*.

Je geeft de korte hash mee van de commit die je wil inzien

1. Je bekijkt de korte hashes van alle commits met **git historiek**
2. Je tikt **git checkout**, een spatie en de korte hash van de commit met als reden **readme.txt aangepast**.  
Het resultaat is bvb. dat je het bestand **favicon.ico** terug ziet.
3. Je keert terug van de commit die je hebt ingezien naar de meest recente commit **git checkout master**  
Het resultaat is bvb. dat je het bestand **favicon.ico** niet meer ziet.

### 5.1 Bestanden in de huidige versie overschrijven vanuit een vorige versie

Je kan bestanden uit de huidige versie overschrijven met hun inhoud uit een vorige versie met de opdracht **checkout**.

Je geeft de korte hash mee van de commit die de vorige versie bevat en je geeft de bestandsnaam mee van het bestand dat je naar de huidige commit wil overbrengen. In plaats van een bestandsnaam kan je ook een wildcard gebruiken als het over *meerdere* bestanden gaat.

1. Je bekijkt de korte hashes van alle commits met **git historiek**
2. Je tikt **git checkout**, een spatie, de korte hash van de commit met als reden **met favicon.ico**, een spatie en **readme.txt**
3. Je tikt **git status**  
Je ziet **modified:   readme.txt**
4. Als je **readme.txt** inzielt, zie je dat die de inhoud heeft van de commit met als reden **met favicon.ico**
5. Je hebt nu twee mogelijkheden.
  - a. Je wil de huidige versie van **readme.txt** opnemen in de huidige commit (wat we hier niet gaan doen)
    - i. Je tikt **git add readme.txt**
    - ii. Je tikt **git commit -m "readme.txt overschreven door vorige versie"**
  - b. Je wil **readme.txt** terug zoals hij was voor **git checkout ... readme.txt** (wat we hier wel gaan doen)
    - i. Je tikt **git checkout head readme.txt**

## 5.2 Tag

Een korte hash is niet gemakkelijk te onthouden en in te tikken.

Je lost dit op door een commit uit de repository een bijnaam (tag) te geven.

Je doet dit met de opdracht **tag**.

1. Je bekijkt eerst de korte hashes van de commits met **git historiek**
2. Je tikt nu **git tag metfavicon.ico** en een spatie en daarna de korte hash van de commit met als reden **readme.txt aangepast**. In dit voorbeeld is **metfavicon.ico** de tag die je aan deze commit geeft.
3. Je kan nu deze tag gebruiken in de opdracht **checkout**  
**git checkout metfavicon.ico**
4. Je plaatst daarna de meest recente commit terug in de working directory  
**git checkout master**
5. Je ziet lijst van de tags met de opdracht  
**git tag**
6. Je ziet in de historiek van je project de tag van een commit met de optie %d  
**git log --pretty=format:"%h %ad %s (%an) %d" --date=short**

## 6 Branches

Je kan met branches tegelijk aan meerdere versies van je project werken.

Je kan werken aan een nieuwe versie 1.1, terwijl je tegelijk de huidige versie 1.0 onderhoudt.

Deze twee parallelle versies heten branches.

Je kan op een bepaald moment branches mergen.

Je fusioneert dan de wijzigingen van de ene branch met de wijzigingen van de andere branch.

Je maakt een branch

- voor experimentele veranderingen (bvb. een algoritme performanter maken).
- voor nieuwe features
- om fouten te corrigeren

Er is altijd een default branch aanwezig. Deze heeft de naam **master**.

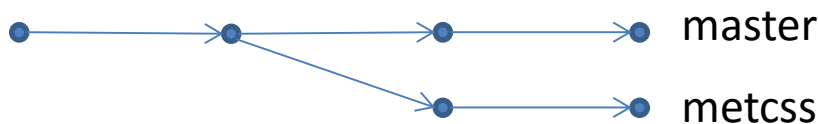
Alle handelingen die je tot nu in het project deed, gebeurden in deze branch.

Als je **git status** intikt, zie je in de meldingen **On branch master**.

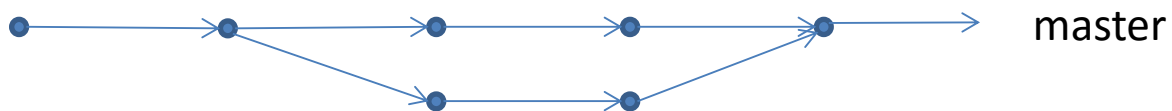
Je zal als voorbeeld een branch **metcss** maken.

Je past in die branch CSS toe op **oliebollen.html**, **pannenkoeken.html** en **wafels.html**.

Je maakt tegelijk in de branch **master** het bestand **mattentaart.html** en je wijzigd **index.html**



Op het einde merge je de branch **metcss** met de branch **master**.



### 6.1 Een branch maken

Je maakt een branch met de opdracht **branch**, gevolgd door de branch naam.

```
git branch metcss
```

```
git branch eentweedebranch
```

### 6.2 Een overzicht van de branches

Je vraagt een overzicht van de branches met de opdracht **branch**.

Git toont een sterretje bij de actieve branch (de branch waarin je momenteel werkt).

```
git branch
```

### 6.3 Een branch verwijderen

Je verwijdert een branch met de opdracht **branch -d**, gevolgd door de branch naam.

```
git branch -d eentweedebranch
```

### 6.4 Een branch activeren

Je activeert een branch met de opdracht **checkout**, gevolgd door de branch naam.

1. Je activeert de branch **metcss** actief

```
git checkout metcss
```

2. Je ziet dat deze branch de actieve is met **git branch**

3. Je kopieert alle oefenbestanden uit de directory **metcss** naar **c:\recepten**

4. Je ziet met **git status** dat Git drie gewijzigde bestanden en één nieuw bestand in de working directory detecteert.



5. Je voegt deze bestanden toe aan de staging area  
`git add .`
6. Je ziet met `git status` dat deze bestanden zich nu bevinden in de staging area
7. Je commit de staging area naar de repository  
`git commit -m "met css"`
8. Je maakt de branch master actief  
`git checkout master`
9. De working directory bevat nu bijvoorbeeld `default.css` niet meer  
Dit bestand bevindt zich in de niet-actieve branch `metcss`.
10. Je kopieert `mattentaart.html` van het oefenmateriaal naar `c:\recepten`
11. Je ziet met `git status` dat Git één nieuw bestand detecteert
12. Je voegt deze bestanden toe aan de staging area  
`git add .`
13. Je ziet met `git status` dat deze bestanden zich nu bevinden in de staging area
14. Je commit de staging area naar de repository  
`git commit -m "met mattentaart"`

## 6.5 Branches mergen

Je merget (fusioneert) de wijzigingen van een bepaalde branch in de actieve branch met de opdracht `merge`, gevolgd door de naam van de bepaalde branch.

Je merget de branch `metcss` in de actieve branch (`master`)  
`git merge metcss`

De working directory bevat nu bijvoorbeeld `default.css`.

Git heeft dit bestand van de branch `metcss` gemerged in de actieve branch.

## 6.6 Merge conflicten

Het gebeurt dat je eenzelfde bestand in twee branches wijzigt.

Wanneer je deze branches merget kan je een merge conflict krijgen.

### 6.6.1 Een wijziging zonder conflict

1. Je maakt de branch `metcss` actief  
`git checkout metcss`
2. Je voegt in `index.html`, na `</title>`, volgende regel toe  
`<link rel="stylesheet" href="default.css">`
3. Je slaat het bestand op.
4. Je voegt het bestand toe aan de staging area  
`git add .`
5. Je commit de staging area naar de repository  
`git commit -m "index.html met css"`
6. Je maakt de branch `master` actief  
`git checkout master`
7. Je voegt in `index.html`, voor `</ul>`, volgende regel toe  
`<li><a href="mattentaart.html">Mattentaart</a></li>`
8. Je slaat het bestand op.
9. Je voegt het toe aan de staging area  
`git add .`
10. Je commit de staging area naar de repository  
`git commit -m "index.html verwijst naar mattentaart.html"`
11. Je merget de branch `metcss` in de actieve branch (`master`)  
`git merge metcss`
12. Het bestand `index.html` bevat nu beide regels.

### 6.6.2 Een wijziging met een conflict

Git kan bij een merge soms niet zelf beslissen hoe de wijzigingen op een bestand door te voeren. Je hebt dan een merge conflict, dat jij moet oplossen.

1. Je maakt een branch **werkwijzecorrigeren**  
`git branch werkwijzecorrigeren`
2. Je maakt deze branch actief  
`git checkout werkwijzecorrigeren`
3. Je wijzigt in **oliebollen.html**, de regel **Bestrooi met bloedsuiker en serveer.** naar **Bestrooi met veel bloedsuiker en serveer.**
4. Je slaat het bestand op.
5. Je voegt het bestand toe aan de staging area  
`git add .`
6. Je commit de staging area naar de repository  
`git commit -m "oliebollen met veel bloedsuiker"`
7. Je maakt de branch **master** actief  
`git checkout master`
8. Je wijzigt in **oliebollen.html**, de regel **Bestrooi met bloedsuiker en serveer.** naar **Bestrooi met weinig bloedsuiker en serveer.**
9. Je slaat het bestand op.
10. Je voegt het toe aan de staging area  
`git add .`
11. Je commit de staging area naar de repository  
`git commit -m "oliebollen met weinig bloedsuiker"`
12. Je merget de branch **werkwijzecorrigeren** in de actieve branch (**master**)  
`git merge werkwijzecorrigeren`
13. Je krijgt foutmeldingen  
**CONFLICT (content): Merge conflict in oliebollen.html**  
**Automatic merge failed; fix conflicts and then commit the result.**
14. Je ziet in het conflicterende bestand **oliebollen.html** volgende regels op de plaats van de conflicterende wijziging  

```
<<<<<< HEAD
Bestrooi met weinig bloedsuiker en serveer.</li>
=====
Bestrooi met veel bloedsuiker en serveer.</li>
>>>>>> werkwijzecorrigeren
```

Je ziet tussen **<<<<<< HEAD** en **=====** de wijziging uit de huidige branch (**master**).  
 Je ziet tussen **=====** en **>>>>>> werkwijzecorrigeren** de wijziging uit de branch die je wil mergen (**werkwijzecorrigeren**).
15. Je lost het conflict op door deze regels te wijzigen naar **Bestrooi met bloedsuiker naar keuze en serveer.</li>**
16. Je slaat het bestand op.
17. Je voegt het bestand toe aan de staging area  
`git add .`
18. Je commit de staging area naar de repository  
`git commit -m "oliebollen met bloedsuiker naar keuze"`

## 7 GitHub: een public repository

Je kan zelf een public repository maken op één van je servers.

Je maakt als alternatief in deze cursus een public repository op de gratis Git server GitHub.

Je maakt een GitHub account op <https://github.com/>.

### 7.1 Een public repository aanmaken

1. Je ziet bovenaan de pagina je user name.
2. Je klikt daarnaast op  en je kiest **New repository**.  
Je maakt zo een public repository.
3. Je tikt bij **Repository name** de naam van de public repository: **recepten**
4. Je klikt op de knop 
5. Je ziet een pagina met de URL van de public repository:  
**<https://github.com/JeAccountNaam/recepten.git>**

### 7.2 De public repository kenbaar maken

Je maakt in je private repository de public repository kenbaar onder een bijnaam met de opdracht **remote add** gevolgd door de bijnaam, gevolgd door de URL van de public repository.

Je maakt de public repository op de URL **<https://github.com/JeAccountNaam/recepten.git>** kenbaar onder de bijnaam **origin**

**git remote add origin <https://github.com/JeAccountNaam/recepten.git>**



Opgepast, deze URL is hoofdlettergevoelig  
en je moet deze opdracht uitvoeren in een directory die al een git repository heeft.

Je kan een lijstje van de bijnamen opvragen met de opdracht

**git remote**

### 7.3 Push

Je zet de private repository over naar de public repository met de opdracht **push**, daarna de bijnaam van de public repository, daarna de naam van een branch in de private repository.

1. Je zet de private repository branch **master** over naar de public repository **origin**  
**git push origin master**
2. Je tikt de username en paswoord van je GitHub account.
3. GitHub zet daarna de private repository over naar de public repository.
4. Je klikt daarna op je user name in de website van Git.
5. Je ziet een pagina met een lijst **Popular repositories** en daarin de repository **recepten**.
6. Je klikt deze repository aan en je ziet de inhoud van deze repository.  
Je ziet ook de inhoud van het bestand **readme.txt**

### 7.4 Clone

Als nieuw teamlid moet je één keer de public repository overzetten naar een nieuwe private repository. Je doet dit met de opdracht **clone**, gevolgd door de URL van de public repository.

Je sluit het command prompt venster om dit te proberen en je verwijdert **c:\recepten**

Je tikt volgende opdrachten in de command prompt



```
cd c:\
git clone https://github.com/JeAccountNaam/recepten.git
```

Git maakt een directory **recepten** in de huidige directory (/)  
en plaatst in die directory een kopie van de public repository.

## 7.5 Fetch

Een teamlid brengt de wijzigingen van andere leden in de public repository over naar zijn private repository met de opdracht **fetch**, gevolgd door de bijnaam van de public repository.

Je wijzigt eerst in de public repository **default.css** via de website van GitHub.

1. Je klikt op **default.css** en je kiest 
2. Je wijzigt **#EF503C** naar **Chocolate**
3. Je tikt **achtergrondkleur Chocolate** onder **Commit changes**
4. Je klikt op 

Je tikt volgende opdrachten in de command prompt

```
cd c:\recepten
git fetch origin
```

Git plaatst de wijzigingen niet onmiddellijk in de huidige branch (**master**) van de private repository, maar in een branch **origin/master**.

Je merget de wijzigingen met de huidige branch: **git merge origin/master**

Je ziet dat git één bestand wijzigde (**default.css**).



## 7.6 Pull

Je kan de twee opdrachten uit het vorig puntje (**git fetch origin** en **git merge origin/master**) combineren met één opdracht: **git pull origin master**

## 7.7 Conflict

Als een bestand tegelijk in de public repository én in de private repository wijzigde, krijg je een conflict bij de opdracht **pull**, vergelijkbaar met een merge conflict.

Je wijzigt in de public repository **default.css** via de website van GitHub.

1. Je klikt op **default.css** en je kiest 
2. Je wijzigt **Chocolate** naar **Olive**
3. Je tikt **achtergrondkleur Olive** onder **Commit changes**
4. Je klikt op 
1. Je wijzigt in de private repository **c:\recepten** in het bestand **default.css** de kleur **Chocolate** naar **Orange**.
2. Je plaatst dit bestand in de staging area  
**git add .**
3. Je commit de staging area naar de repository  
**git commit -m "achtergrondkleur orange"**
4. Je zet de wijzigingen van de public repository over naar de private repository  
**git pull origin master**
5. Je ziet een foutmelding  
**CONFLICT (content): Merge conflict in default.css**
6. Je ziet in **default.css** de wijziging in de private branch,  
daarna de wijziging in de public branch  

```
<<<<<<< HEAD
    background-color: Orange;
=====
    background-color: Olive;
>>>>>>> e71d78894b7ba5edc71d535794fb7103109871e1
```
7. Je lost het conflict op door deze regels te wijzigen naar  
**background-color: Orange;**
8. Je slaat het bestand op.
9. Je plaatst dit bestand in de staging area  
**git add .**

10. Je commit de staging area naar de repository

`git commit -m "na achtergrondkleur conflict"`



Opmerking

Met de hyperlink [Graphs, Network](#) zie je een mooie grafische historiek van je project.



Opmerking

Naast GitHub kan je ook een repository maken op bijvoorbeeld **bitbucket.org**



Opmerking

Veel cursisten stellen een project, waaraan ze werken gedurende een opleiding, ter beschikking met een public repository.

Hun project heeft op de PC in het opleidingscentrum een private repository.

Op het einde van een opleiding dag pushen ze die naar de public repository.

Thuis clonen ze een nieuw project van de public repository naar hun PC.

Wijzigingen van een bestaand project pullen ze in de private repository.

## 8 Git grafisch

Je leert in dit hoofdstuk werken met de GUI meegeleverd met Git.

Je kopieert eerst **mattentaart.jpg** van het oefenmateriaal naar **c:\recepten\images**.

Je klikt in de Windows File Explorer met de rechtermuisknop op de map **recepten** en je kiest de opdracht **Git Gui**

### 8.1 Bestanden toevoegen aan de staging area

1. Je ziet **images/mattentaart.jpg** bij **Unstaged Changes**
2. Je klikt op de knop **Stage Changed**
3. Je ziet **images/mattentaart.jpg** nu bij **Staged Changes**

### 8.2 De staging area committen

Je tikt met **mattentaart.jpg** bij **Commit Message** en je klikt op de knop **Commit**

### 8.3 Een bestandswijziging committen

1. Je laat de Git GUI tool openstaan.
2. Je wijzigt **mattentaart.html**.  
Je tikt na `</h1>` ``
3. Je keert terug naar de Git GUI tool.
4. Je klikt op de knop **Rescan**. Je ziet **mattentaart.html** bij **Unstaged Changes**
5. Je klikt op de knop **Stage Changed** en je antwoord bevestigend.  
Je ziet **mattentaart.html** nu bij **Staged Changes**
6. Je tikt met **mattentaart.jpg** verwijzing bij **Commit Message** en je klikt op **Commit**

### 8.4 Een branch maken

1. Je kiest in het menu **Branch** de opdracht **Create**.
2. Je tikt bij **Name** de naam van de nieuwe branch: **metcopyright** en je kiest **Create**  
De melding **Checked out 'metcopyright'** onder in het venster toont dat deze nieuwe branch direct de actieve branch is.
3. Je wijzigt **index.html**. Je tikt na `</nav>`  
`<footer><small>&copy; Copyright Git oefening</small></footer>`
4. Je keert terug naar de Git GUI tool.
5. Je klikt op de knop **Rescan**. Je ziet **index.html** bij **Unstaged Changes**
6. Je klikt op de knop **Stage Changed**  
Je ziet **index.html** nu bij **Staged Changes**
7. Je tikt met **copyright** bij **Commit Message** en je klikt op de knop **Commit**

### 8.5 Een branch actief maken

1. Je kiest in het menu **Branch** de opdracht **Checkout**
2. Je kiest **master** in de lijst en je klikt op de knop **Checkout**
3. Je wijzigt **index.html**. Je tikt na `</nav>`  
`<footer><small>&copy; Copyright de Zweedse kok</small></footer>`
4. Je keert terug naar de Git GUI tool
5. Je klikt op de knop **Rescan**. Je ziet **index.html** bij **Unstaged Changes**
6. Je klikt op de knop **Stage Changed**. Je ziet **index.html** nu bij **Staged Changes**
7. Je tikt met **de zweedse kok** bij **Commit Message** en je klikt op de knop **Commit**

## 8.6 Een branch mergen

1. Je kiest in het menu **Merge** de opdracht **Local Merge**
2. Je kiest **metcopyright** in de lijst en je klikt op de knop **Merge**  
Je krijgt een foutbericht met een merge conflict. Je sluit dit bericht
3. Je verwijdert in **index.html** de regels <<<<<< **HEAD** tot en met =====  
en de regel >>>>>> **metcopyright**
4. Je slaat deze wijzigingen op
5. Je keert terug naar de Git GUI tool
6. Je kiest in het menu **Commit** de opdracht **Stage to Commit** en je bevestigt  
Je ziet **index.html** nu bij **Staged Changes**
7. Je tikt "na **copyright conflict**" bij **Merge Commit Message** en je klikt op **Commit**

## 8.7 Een public repository kenbaar maken

1. Je kiest in het menu **Remote** de opdracht **Add**
2. Je tikt bij **Name** de bijnaam van de public repository: **remote**
3. Je tikt bij **Location** de URL van de public repository:  
**<https://github.com/JeAccountNaam/recepten.git>**
4. Je kiest **Do Nothing Else Now** bij **Further Action**

## 8.8 Push

1. Je kiest in het menu **Remote** de opdracht **Push**
2. Je kiest **remote** bij **Remote** en je klikt op de knop **Push**
3. Je tikt de username en het paswoord van je GitHub account

## 8.9 Clone

1. Je sluit het venster met de command prompt en je sluit Git GUI
2. Je verwijdert de directory **c:\recepten**
3. Je tikt volgende opdrachten in aan de command prompt  
**cd \**  
**git gui**
4. Je kiest **Clone Existing Repository**
5. Je tikt bij **Source Location** de URL van de public repository  
**<https://github.com/JeAccountNaam/recepten.git>**
6. Je tikt **recepten** bij **Target directory**
7. Je klikt op de knop **Clone**

## 8.10 Pull

1. Je wijzigt via de GitHub website in de public repository in **default.css**  
**Orange** naar **LightSalmon**
2. Je keert terug naar de Git GUI tool
3. De Git GUI tool heeft geen **pull** opdracht.  
Je moet dus pullen met de combinatie van **fetch** en **merge**
4. Je kiest in het menu **Remote** de opdracht **Fetch from** en daarna **origin**
5. Je kiest daarna in het menu **Merge** de opdracht **Local Merge**  
Je kiest **origin/master** in de lijst en je klikt op de knop **Merge**

## 9 Eclipse integratie

Je neemt dit hoofdstuk door als je regelmatig Eclipse gebruikt. Deze cursus gebruikt Eclipse Mars.

### 9.1 Het project maken

Je maakt in Eclipse een Java console project

(Je integreert Git in andere projecttypes op dezelfde manier)

1. Je kiest in het menu **File** de opdracht **New, Project, Java Project**
2. Je kiest **Next**
3. Je tikt **GitCursus** bij **Project Name** en je kiest **Finish**

### 9.2 Een repository maken in het project

1. Je klikt in de **Package Explorer** met de rechtermuisknop op je project
2. Je kiest de opdracht **Team, Share Project**
3. Je plaatst een vinkje bij **Use or create repository in parent folder of project**
4. Je selecteert het project **GitCursus**
5. Je kiest **Create Repository** en je kiest **Finish**

### 9.3 De staging area

Je kiest in het menu **Window** de opdracht **Show View, Other, Git, Git Staging** en **OK**

Je ziet een venster met onder andere **Unstaged Changes** en **Staged Changes**

Je voegt een source file toe aan het project

1. Je klikt in de **Package Explorer** met de rechtermuisknop op je project
2. Je kiest **New, Class**
3. Je tikt **be.vdab** bij **Package**
4. Je tikt **Main** bij **Name**
5. Je plaatst een vinkje bij **public static void main(String[] args)**
6. Je kiest **Finish**

Je ziet onderaan **Main.java** vermeld bij **Unstaged changes**

Je ziet ook al een bestand **.gitignore**

Je plaatst deze bestanden in de staging area

1. Je selecteert **Main.java** en **.gitignore** in **Unstaged changes**
2. Je sleept beide bestanden naar **Staged changes**

Je commit de staging area

1. Je tikt **eerste commit** bij **Commit Message**
2. Je klikt op de knop **Commit**

Je tikt in **Main.java** onder de regel **// TODO ... volgende regel**

```
System.out.println("Hallo");
```

Je slaat het bestand op.

Je ziet onderaan **Main.java** terug vermeld bij **Unstaged changes**

Je kan op een verkorte manier de unstaged changes toevoegen aan de staging area en de staging area committen

1. Je klikt in de **Package Explorer** met de rechtermuisknop op je project
2. Je kiest **Team, Commit**
3. Je tikt **hallo** bij **Commit Message**
4. Je klikt op de knop **Commit**



## 9.4 Lijstje van de commits

1. Je kiest in het menu **Window** de opdracht **Show View, Other, Git, Git Repositories**
2. Je klikt in het venster **Git Repositories** met de rechtermuisknop op **GitCursus**
3. Je kiest de opdracht **Show In, Git Reflog**
4. Je ziet een venster met de twee commits van dit project

## 9.5 Checkout

Je doet een checkout van de commit met de commit message **eerste commit**

1. Je klikt in de **Package Explorer** met de rechtermuisknop op het project
2. Je kiest de opdracht **Replace With, Commit**
3. Je kiest **OK** bij de waarschuwing
4. Je kiest in de lijst de commit met de message **eerste commit**
5. Je klikt op de knop **OK**

**GitCursus.java** bevat de regel **System.out.println("Hallo")** niet meer.

Je doet op dezelfde manier een checkout van de commit met de message **hallo**

## 9.6 Branches

Je maakt een branch **intfrans**

1. Je kiest in het venster **Git Repositories GitCursus, Branches, Local**
2. Je klikt met de rechtermuisknop op **master** en je kiest **Create Branch**
3. Je tikt **intfrans** bij **Branch name**
4. Je kiest **Finish**

Deze branch wordt onmiddellijk de actieve branch.

Je ziet dit in het venster **Git Repositories**: er staat een vinkje bij deze branch.

1. Je wijzigt in **Main.java** **Hallo** naar **Bonjour**
2. Je slaat deze source op
3. Je sleept in het venster **Git Staging Main.java** naar **Staged Changes**
4. Je tikt **bonjour** bij **Commit Message**
5. Je klikt op de knop **Commit**

Je maakt de branch **master** terug actief

1. Je dubbelklikt in het venster **Git Repositories master** (binnen **Branches, Local**)
2. Je klikt op de knop **OK**

Je ziet de versie van **Main.java** zoals ze in deze branch werd gecommitt.

Je merget de branch **intfrans** in de branch **master**

1. Je klikt in het venster **Git Repositories** met de rechtermuisknop op **master**
2. Je kiest de opdracht **Merge**
3. Je kiest de branch **intfrans**
4. Je klikt op de knop **Merge**
5. Je klikt op de knop **OK**

Je leert nu met een merge conflict oplossen

1. Je activeert de branch **intfrans**  
en je wijzigt in **Main.java** **Bonjour** naar **Bonjour tout le monde**
2. Je commit deze wijziging
3. Je activeert de branch **master** en je wijzigt in **Main.java** **Bonjour** naar **Hallo iedereen**
4. Je commit deze wijziging
5. Je merget de branch **intfrans** in de branch **master**
6. Je klikt op de knop **OK** in het venster met als eerste regel **Result Conflicting**
7. Je ziet in **Main.java** de wijziging in de branch **master**,  
gevolgd door de conflicterende wijziging in de branch **intfrans**
8. Je corrigeert **Main.java: System.out.println("Hallo iedereen");**

9. Je commit deze correctie

## 9.7 Push – Pull

Je maakt op de website van GitHub een repository **gitcursuseclipse**

Je maakt deze public repository kenbaar in Eclipse

1. Je klikt in het venster **Git Repositories** met de rechtermuisknop op het onderdeel **Remotes**
2. Je kiest de opdracht **Create Remote**
3. Je laat **origin** staan bij **Remote Name** en je klikt op **OK**
4. Je klik op de knop **Change** bij **Url** en je tikt de URL van de public repository bij **URI**.
5. Je vult **User** en **Password** in, vinkt **Store in Secure store** aan en je kiest **Finish, Save**

Je pusht de private repository naar de public repository

1. Je klikt in het venster **Git Repositories** met de rechtermuisknop op **master**
2. Je kiest **Push Branch, Next** en **Finish**

Je wijzigt via de website van GitHub in **Main.java** iedereen naar **allemaal**

Je fetcht de public repository in de private repository

1. Je opent in het venster **Git Repositories** het onderdeel **Remotes**
2. Je klikt met de rechtermuisknop op **origin**
3. Je kiest de opdracht **Configure Fetch**
4. Je klikt op de knop **Add**
5. Je tikt **m** bij **Source**  
Eclipse zoekt in de public repositories welke branches met **m** beginnen en stelt de branch **master [branch]** voor. Je klikt op dit voorstel.  
**Source** wijzigt naar **refs/heads/master**
6. Je klikt op de knop **Next**
7. Je ziet **refs/remotes/origin/master** bij **Destination**  
Dit is de private repository branch waarin Eclipse de public repository merget
8. Je klikt op de knop **Finish**.
9. Je klikt op de knop **Save**
10. Je klikt in het venster **Git Repositories** met de rechtermuisknop op **origin**
11. Je kiest de opdracht **Fetch**
12. Je klikt op de knop **OK**
13. Je klikt in het venster **Git Repositories** het onderdeel **Branches** open
14. Je klikt daarbinnen het onderdeel **Remote Tracking** open
15. Je klikt met de rechtermuisknop op **origin/master**
16. Je kiest de opdracht **Merge**  
Je merget hiermee deze branch in de actieve Local branch: **master**

## 10 NetBeans integratie

Je neemt dit hoofdstuk door als regelmatig NetBeans gebruikt. Deze cursus gebruikt NetBeans 8.0.1

### 10.1 Het project maken

Je maakt in NetBeans een Java console project.

(Je integreert Git in andere projecttypes op dezelfde manier.)

1. Je kiest in het menu **File** de opdracht **New Project, Java, Java Application**
2. Je kiest **Next**
3. Je tikt **GitCursus** bij **Project Name** en je kiest **Finish**

### 10.2 Een repository maken in het project

Je kiest in het menu **Team** de opdracht **Git, Initialize Repository** en je klikt op **OK**

### 10.3 De staging area

Als je in het tabblad **Projects** de muisaanwijzer laat rusten op **GitCursus.java**, zie je **Added**.

Dit betekent dat deze source zich automatisch in de staging area bevindt.

Je commit de staging area naar de repository

1. Je klikt in het menu **Team** de opdracht **Commit**
2. Je tikt **eerste commit** bij **Commit Message**
3. Je klikt op de knop **Commit**

Je tikt in **GitCursus.java** onder de regel `// TODO ... volgende regel`

```
System.out.println("Hallo");
```

Je slaat het bestand op.

Als je in het tabblad **Projects** de muisaanwijzer laat rusten op **GitCursus.java** zie je **Modified**.

Dit betekent dat de source gewijzigd is in de working directory, en bewaard is in de staging area.

Je commit de staging area met de commit message **hallo**

### 10.4 Lijstje van de commits

1. Je kiest in het menu **Team** de opdracht **Show History**

### 10.5 Checkout

Je doet een checkout van de commit met de commit message **eerste commit**

1. Je kiest in het menu **Team** de opdracht **Checkout, Checkout Revision**
2. Je klikt op de knop **Select**
3. Je kiest links **Branches, Local, Master**
4. Je kiest rechts de commit met de message **eerste commit**
5. Je klikt op de knop **Select**
6. Je klikt op de knop **Checkout**

**GitCursus.java** bevat de regel `System.out.println("Hallo")` niet meer.

Je doet op dezelfde manier een checkout van de commit met de message **hallo**

### 10.6 Branches

Je maakt een branch **intfrans**

1. Je kiest in het menu **Team** de opdracht **Branch/Tag, Create Branch**
2. Je tikt **intfrans** bij **Branch Name**
3. Je plaatst een vinkje bij **Checkout Created Branch**
4. Je kiest **Create**

Deze branch is nu de actieve branch.

1. Je wijzigt in **GitCursus.java** **Hallo** naar **Bonjour**
2. Je slaat deze source op
3. Je kiest in het menu **Team** de opdracht **Commit**
4. Je tikt **bonjour** bij **Commit Message**
5. Je klikt op de knop **Commit**

Je maakt de branch **master** terug actief

1. Je kiest in het menu **Team** de opdracht **Branch/Tag, Switch to Branch**
2. Je kiest **master** bij **Branch**
3. Je klikt op de knop **Switch**

Je ziet de versie van **GitCursus.java** zoals ze in deze branch werd gecommit.

Je merget de branch **intfrans** in de branch **master**

1. Je kiest in het menu **Team** de opdracht **Branch/Tag, Merge Revision**
2. Je klikt op de knop **Select**
3. Je kiest links **Branches, Local, intfrans**
4. Je klikt op de knop **Select**
5. Je klikt op de knop **Merge**

Je leert nu om te gaan met een merge conflict

1. Je activeert de branch **intfrans**  
en je wijzigt in **GitCursus.java** de tekst **Bonjour** naar **Bonjour tout le monde**
2. Je commit deze wijziging
3. Je activeert de branch **master**  
en je wijzigt in **GitCursus.java** de tekst **Bonjour** naar **Hallo iedereen**
4. Je commit deze wijziging
5. Je merget de branch **intfrans** in de branch **master**
6. Je ziet een venster met de titel **Resolve Conflicts**. Je kiest **Resolve**
7. Je ziet links **GitCursus.java** zoals gewijzigd in de branch **master**  
Je ziet rechts **GitCursus.java** zoals gewijzigd in de branch **intfrans**
8. Je prefereert de wijziging in de branch **master** met de knop **Accept**
9. Je klikt onder in het venster op de knop **OK**
10. Je slaat het bestand op
11. Je commit de staging area

## 10.7 Push – Pull

Je maakt op de website van GitHub een repository **gitcursusnetbeans**

Je pusht de private repository naar de public repository

1. Je kiest in het menu **Team** de opdracht **Remote, Push**
2. Je tikt bij **Repository URL** de URL van de public repository
3. Je tikt de username en het paswoord van je GitHub account
4. Je klikt op de knop **Next**
5. Je plaatst een vinkje bij de branch **master**
6. Je klikt op de knop **Next**
7. Je klikt op de knop **Finish**

Je wijzigt via de website van GitHub in **Main.java** **iedereen** naar **allemaal**

Je pullt de public repository in de private repository

1. Je kiest in het menu **Team** de opdracht **Remote, Pull**
2. Je klikt op de knop **Next** en je klikt op de knop **Finish**

## 11 Visual Studio integratie

Je neemt dit hoofdstuk door als je regelmatig Visual Studio 2013 gebruikt.

### 11.1 Het project maken


Je maakt in Visual Studio een C# console project.

1. Je kiest in het menu **File** de opdracht **New, Project, Visual C#, Console Application**
2. Je tikt **GitCursus** bij **Project Name** en je kiest **OK**

### 11.2 Een repository maken in het project

1. Je klikt in de **Solution Explorer** met de rechtermuisknop op de solution
2. Je kiest de opdracht **Add Solution to Source Control**

### 11.3 Algemene instellingen

1. Je kiest in het menu **View** de opdracht **Team Explorer**
2. Je klikt in de **Team Explorer** op  en je kiest **Settings**
3. Je kiest **Settings**
4. Je kiest **Global Settings**
5. Je kan je **User Name** en **Email Address** instellen


### 11.4 De staging area

Je tikt in **Program.cs** onder de derde accolade `Console.WriteLine("Hallo");`

Je ziet in de **Solution Explorer** een rood vinkje voor **Program.cs**.

Dit betekent dat deze source gewijzigd is in de working directory, en bewaard is in de staging area.

Je commit de staging area naar de repository

1. Je klikt in de **Team Explorer** op  en je kiest **Changes**
2. Je tikt **eerste commit** bij **Enter a commit message**
3. Je klikt op de knop **Commit All**

Je commit de staging area met de commit message **hallo**

### 11.5 Lijstje van de commits

1. Je klikt in de **Team Explorer** op  en je kiest **Branches**
2. Je klikt onder **Unpublished Branches** met de rechtermuisknop op **master**
3. Je kiest de opdracht **View History**

### 11.6 Branches


Je maakt een branch **intfrans**

1. Je klikt in de **Team Explorer** op  en je kiest **Branches**
2. Je klikt met de rechtermuisknop op **master** en je kiest **New Local Branch From**
3. Je tikt **intfrans** bij **Enter a branch name**
4. Je kiest **Create Branch**

Deze branch is nu de actieve branch.


1. Je wijzigt in **Program.cs** **Hallo** naar **Bonjour**
2. Je commit de staging area met de commit message **bonjour**

Je maakt de branch **master** terug actief

1. Je klikt in de **Team Explorer** op  en je kiest **Branches**
2. Je dubbelklikt **master**

Je ziet de versie van **Program.cs** zoals ze in deze branch werd gecommitt.

Je merget de branch **intfrans** in de branch **master**

1. Je klikt in de **Team Explorer** op  en je kiest **Branches**
2. Je kiest **Merge**
3. Je kiest **intfrans** bij **Merge from branch**
4. Je kiest **master** bij **Into current branch**
5. Je kiest **Merge**


Je leert nu een merge conflict oplossen

1. Je activeert de branch **intfrans** en je wijzigt in **Program.cs**  
**Bonjour** naar **Bonjour tout le monde**
2. Je commit deze wijziging
3. Je activeert de branch **master** en je wijzigt in **Program.cs** **Bonjour** naar **Hallo iedereen**
4. Je commit deze wijziging
5. Je merget de branch **intfrans** in de branch **master**
6. Je kiest **Conflicts:1**
7. Je kiest **Program.cs**
8. Je kiest **Compare Files**
9. Je ziet links **Program.cs** zoals gewijzigd in de branch **intfrans**  
Je ziet rechts **Program.cs** zoals gewijzigd in de branch **master**
10. Je prefereert de wijziging in de branch **master**  
met **Keep Target** bij **master** in de **Team Explorer**.
11. Je kiest **Commit Merge**
12. Je tikt **conflict opgelost** bij **Enter a commit message**
13. Je klikt op de knop **Commit All**

## 11.7 Push – Pull


Je maakt op de website van GitHub een repository **gitcursusvisualstudio**

Je pusht de private repository naar de public repository

1. Je klikt in de **Team Explorer** op  en je kiest **Sync**
2. Je kiest **Publish Git Repo** onder **Publish to Remote Repository**
3. Je tikt bij **Enter the URL of an empty Git repo** de URL van de public repository
4. Je kiest **Publish**

Je wijzigt via de website van GitHub in **Program.cs** **iedereen** naar **allemaal**

Je pullt de public repository in de private repository

1. Je klikt in de **Team Explorer** op  en je kiest **Sync**
2. Je kiest **Pull**

## COLOFON

Domeinexpertisemanager	Jean Smits
Moduleverantwoordelijke	
Auteurs	Hans Desmet
Versie	22/8/2016
Codes	Peoplesoftcode: Wettelijk depot:

### Omschrijving module-inhoud

Abstract	Doelgroep	Opleiding informatica
	Aanpak	Zelfstudie
	Doelstelling	Git kunnen gebruiken
Trefwoorden		Git
Bronnen/meer info		