

## Tutorial: Github Basis – terminal.

(Begin met de terminal te openen. Als je deze niet vindt kan je op 'cmd+spatie' drukken voor 'terminal' in spotlight te zoeken.)

### Basis terminalcommands:

`cd` → (Change Directory) Gaat terug naar het hoofdoverzicht van de computer

`ls` → (List) Laat je zien wat er bevindt in de map waarin je je bevindt.

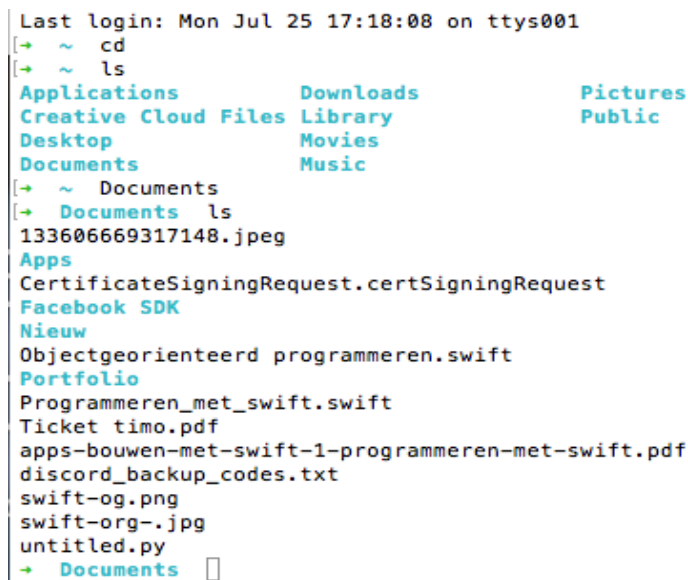
Cmd + K → Leeg je scherm van je terminal.(Je blijft in dezelfde map.)

Om te navigeren naar een volgende map hoeft je enkel de naam van de map in te geven. Bv: 'Documents' zie afbeelding 1.

Speel hier al eens wat mee zodat je bekend geraakt in terminal. Open verschillende mappen door hun naam in te geven. Ga terug naar de hoofdmap door 'cd' in te geven, en kijk in verschillende mappen wat er in staat door 'ls' in te geven.

(Hulp: bij het ingeven van je map naam kan je ook na enkele letters op de tab button drukken. Deze vult je woordt automatisch aan als het bestaat.)

Eens je in een map zit, dan staat dit ook aangegeven.



```
Last login: Mon Jul 25 17:18:08 on ttys001
[+] ~ cd
[+] ~ ls
Applications      Downloads      Pictures
Creative Cloud Files Library      Public
Desktop           Movies
Documents          Music
[+] ~ Documents
[+] Documents ls
133606669317148.jpeg
Apps
CertificateSigningRequest.certSigningRequest
Facebook SDK
Nieuw
Objectgeoriënteerd programmeren.swift
Portfolio
Programmeren_met_swift.swift
Ticket timo.pdf
apps-bouwen-met-swift-1-programmeren-met-swift.pdf
discord_backup_codes.txt
swift-og.png
swift-org-.jpg
untitled.py
[+] Documents
```

Afbeelding 1

### Nieuwe mappen & files zelf aanmaken:

`mkdir` → (Make Directory) laat je een nieuwe map aanmaken op je computer, in de map waar jij je bevindt met je terminal. (afbeelding 2.)

`touch` → Laat je een nieuwe file aanmaken. Bv: een textfile (afbeelding 3.)

```

[→ Documents mkdir gitvandaag ]
[→ Documents ls ]
133606669317148.jpeg
Apps
CertificateSigningRequest.certSigningRequest
Facebook SDK
Nieuw
Objectgeoriënteerd programmeren.swift
Portfolio
Programmeren_met_swift.swift
Ticket timo.pdf
apps-bouwen-met-swift-1-programmeren-met-swift.pdf
discord_backup_codes.txt
gitvandaag
swift-og.png
swift-org-.jpg
untitled.py
[→ Documents ]

```

Afbeelding 2

```

[→ Documents gitvandaag ]
[→ gitvandaag touch nieuwtextfiletje.txt ]
[→ gitvandaag ls ]
nieuwtextfiletje.txt

```

Afbeelding 3

### Basis git via terminal:

*git init* → Als je een map hebt dat nog niet geïnitieerd is voor github. Achteraf komt er bij deze map in je terminal 'git' bij te staan. (Afbeelding 4)

Normaal klik je dit automatisch aan bij het maken van een xcode project. Maar in sommige gevallen is dit niet gebeurt of ben je het misschien vergeten, dan kan dit handig zijn. (Afbeelding 4.)

```

[→ gitvandaag git init ]
Initialized empty Git repository in /Users/Stefan/Documents/gitvandaag/.git/
[→ gitvandaag git:(master) * ]

```

Afbeelding 4

*git status* → Dit laat de status van je map zien, rode files wilt zeggen dat deze niet opgeslagen kunnen worden in je git repository en dat dit filedje niet nagekeken wordt op vernieuwingen. Om dit te bekomen doe je: 'git add'

*git add* → bv.: git add nieuwtextfiletje.txt. Nu wordt het textfiletje toegevoegd om nagekeken te worden of er wijzigingen in gebeuren. Als je kijkt naar je status ('git status') dan zie je hieronder dat het bestand groen is geworden, en dus klaar is om naar github gestuurd te worden. (Afbeelding 5) (git add -A zorgt ervoor dat je al je files add in 1 keer, dit wordt vaker gebruikt tijdens het coderen)

```

[→ gitvandaag git:(master) * git status ]
On branch master

Initial commit

Untracked files:
  (use "git add <file>..." to include in what will be committed)

    nieuwtextfiletje.txt

nothing added to commit but untracked files present (use "git add" to track)
[→ gitvandaag git:(master) * git add nieuwtextfiletje.txt ]
[→ gitvandaag git:(master) * git status ]
On branch master

Initial commit

Changes to be committed:
  (use "git rm --cached <file>..." to unstage)

    new file:   nieuwtextfiletje.txt

```

Afbeelding 5

*git commit -m "Je persoonlijk bericht" →*

Een commit is een soort 'timestamp' het laat achteraf op github zien wanneer je juist een verandering hebt gedaan en welke. Deze verandering moet je natuurlijk wel zelf ingeven en de 'timestamp' is er pas vanaf dat jij je file of project/map commit. Bv.: `git commit -m "ik heb net een nieuwe file aangemaakt"`. Als je nu achteraf weer naar je status gaat zien, dan geeft hij aan dat alles in orde is.(afbeelding 6)

```
[→ gitvandaag git:(master) ✕ git commit -m "Ik heb net een nieuwe file aangemaakt"]
t"
[master (root-commit) be8aa29] Ik heb net een nieuwe file aangemaakt
 1 file changed, 0 insertions(+), 0 deletions(-)
 create mode 100644 nieuwtextfiletje.txt
[→ gitvandaag git:(master) git status ]
On branch master
nothing to commit, working directory clean
```

Afbeelding 6

Probeer elke keer wanneer je een wijziging hebt gedaan (zeker een goede wijziging bv.: nieuwe feature voor je app) te commiten!

*git log →*

Laat een soort logboek zien van je 'git' dus je kan al je commit's zien met hun tekst en info. Om terug uit je log te gaan indien nodig druk je op de 'q' toets.

```
commit be8aa29c7320d3de519641368b2cac34032021c
Author: AdamsDevelopment <adams.development@hotmail.com>
Date:   Mon Jul 25 18:14:35 2016 +0200

    Ik heb net een nieuwe file aangemaakt
(END)
```

Afbeelding 7

*git checkout →*

Stel dat je terug wilt naar een oude commit, kan je via checkout dit doen op volgende manier: `git checkout be8aa29`. Dit zijn de eerste 7 cijfers van mijn commit zoals in afbeelding 7.

Vb. je maakt een file aan 'file1.docx', deze commit je en daarna maak je een nieuwe file aan 'file2.docx' en deze commit je ook. Stel je wilt terug naar de eerste commit gaan zodat alles na die eerste commit terug verdwijnt. Dan je gebruik je `git checkout` en de eerste 7 cijfers van de commit naar waar je terug wilt. Deze 7 cijfers/letters vindt je terug in je 'git log'.

## Aanmaken van branches en waarom:

*git checkout -b →*

Dankzij `git checkout -b` kan je nieuwe branches maken. bv.: `git checkout -b new branch`. Standaard heb je een master branch. Maar als je deze branch maak verwissel je automatisch naar je nieuwe branch die je net hebt aangemaakt met de naam 'new branch'.

*git checkout →*

Met `git checkout` kan je wisselen van branch. Wil je terug naar de master branch dan geef je in: `git checkout master`. (zie afbeelding 8)

```

[→ gitvandaag git:(master) git checkout -b new-branch ]
Switched to a new branch 'new-branch'
[→ gitvandaag git:(new-branch) git branch ]
master
* new-branch
[→ gitvandaag git:(new-branch) git checkout master ]
Switched to branch 'master'
→ gitvandaag git:(master) █

```

Afbeelding 8

## Waarom branches?

VB1: Stel je werkt met 2 personen aan 1 app en een derde Senior iOS Developer die alles controleert wat jullie schrijven van code. Dan werk je met branches. De Master branch, developer 1 branch, developer 2 branch. Als developer 1 een nieuwe functie heeft toegevoegd, dan commit en pusht hij deze naar de 'developer 1 branch'. De als de senior developer dit heeft nagekeken en goedgekeurd, dan merged hij deze branch in de Master branch. De branch die fysiek gaat uitkomen (de volledige app).

VB2: Stel je werkt aan een bestaande app. Deze heeft al ergens een Master branch als er goed aan gewerkt is. Maar nu moet er opnieuw aan gewerkt worden om nieuwe functies toe te voegen en bugs(fouten) op te lossen. Dan heb je weer 3 branches. Master branch, bugfix branch, functie branch. Als er in de bug fix branch een bug is opgelost. Kan deze weer gemerged worden met de master branch.

## Github voor iOS:

Maak een account op [www.github.com](https://www.github.com). (Hier ga ik vanuit dat iedereen een account kan aanmaken.) Vervolgens gaan we een ssh key aanmaken.

1. Ga naar volgende site <https://help.github.com/articles/generating-an-ssh-key/>
2. Klik op de 'Generating a new SSH key and adding it to the ssh-agent' en volg deze stappen
3. Daarna ga je terug naar de site en klik je op 'Adding a new SSH key to your GitHub account' en volg weer de stappen.

Het doel van de SSH key is om jou specifieke computer te linken met de server van GitHub.

Nu gaan we beginnen met ons iOS Project op github te plaatsen aan de hand van een voorbeeld:

### je Xcode project aanmaken:

1. Maak een nieuw Xcode project aan.
2. Kies voor Single View Application en klik 'Next'.
3. Geef de nodige gegevens (Product Name, ...) in en klik 'Next'. (mijn project noemt githubtest)
4. Kies waar je het wilt opslaan en zorg dat je onderin 'Create Git repository on My Mac' aanvinkt. (Dit is niet nodig als je je project zet in een map die je al geïnitialiseerd hebt voor git.)
5. Voeg nu als test enkele dingen in je storyboard toe. bv.: button, imageView, ...

### Een github repository aanmaken:

1. Ga naar de homepage van github en klik nadat je bent ingelogd op 'New repository'.
2. Kies voor public (private heeft een prijskaartje) geef een 'repository name' op en vink de 'initialize this repository with a README' en klik 'Create repository'. De andere dingen zijn voorlopig niet van belang. (mijn repository noemt githubtest)

Een README file is enkel om wat uitleg etc. te geven over je app.

3. Vervolgens kom je op de homepage van je repository. Nu moet je deze nog linken aan je Xcode project.

### Repository linken aan je Xcode Project:

1. Op de hoofdpagina van je nieuwe repository klik je op clone or download en kopieer je de SSH key.
2. Open dan je terminal en navigeer naar je xcode project map.
3. Als je nog niets hebt veranderd aan je nieuw project en je kijkt naar je 'git status' dan zie je dat alles in orde is. Heb je al wel iets veranderd zoals in het vb hierboven. Dan zie je het volgende als je je 'git status' checkt. (Afbeelding 9)

```
→ Documents githubtest ]
→ githubtest git:(master) x git status ]
On branch master
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git checkout -- <file>..." to discard changes in working directory)

        modified:   githubtest/Base.lproj/Main.storyboard

no changes added to commit (use "git add" and/or "git commit -a")
→ githubtest git:(master) x █
```

Afbeelding 9

4. Als je nu in je terminal test git remote -v, zou je normaal al je remote repository's moeten zien. Aangezien we er nog geen hebben aangemaakt. (daarvoor hoort de SSH key) zal er niets gebeuren.
5. Typ nu in terminal git remote add origin en vervolgens je ssh key. (zie afbeelding 10) git remote add is de command die je gebruikt voor een remote toe te voegen. Origin is gewoon de naam die je aan deze remote geeft. Meest standaard wordt origin gebruikt.
6. Als je dan terug kijkt met het command 'git remote -v' dan zie je dat er een remote is toegevoegd.

```
→ githubtest git:(master) x git remote -v ]
→ githubtest git:(master) x git remote add origin git@github.com:AdamsDevelopment/githubtest.git ]
→ githubtest git:(master) x git remote -v ]
origin git@github.com:AdamsDevelopment/githubtest.git (fetch)
origin git@github.com:AdamsDevelopment/githubtest.git (push)
→ githubtest git:(master) x █
```

Afbeelding 10

7. Vervolgens typ je in terminal 'git add -A'. Dit wil zeggen voeg alle nieuwe files/gewijzigde files toe.
8. Om te controleren kijken we nog eens naar onze 'git status'. Deze zegt nu dat onze files die in afbeelding 9 rood zijn, klaar zijn om gecommit te worden.
9. Nu typen we git commit -m om onze files te committen. Hier moet je een bericht geven en best van wat je hebt gedaan. BV.: 'git commit -m "Buttons op storyboard geplaatst"'

```
→ githubtest git:(master) x git add -A ]
→ githubtest git:(master) x git status ]
On branch master
Changes to be committed:
  (use "git reset HEAD <file>..." to unstage)

        modified:   githubtest/Base.lproj/Main.storyboard

→ githubtest git:(master) x git commit -m "Buttons op storyboard geplaatst" ]
[master 34300d2] Buttons op storyboard geplaatst
1 file changed, 10 insertions(+), 3 deletions(-)
→ githubtest git:(master) x █
```

Afbeelding 11

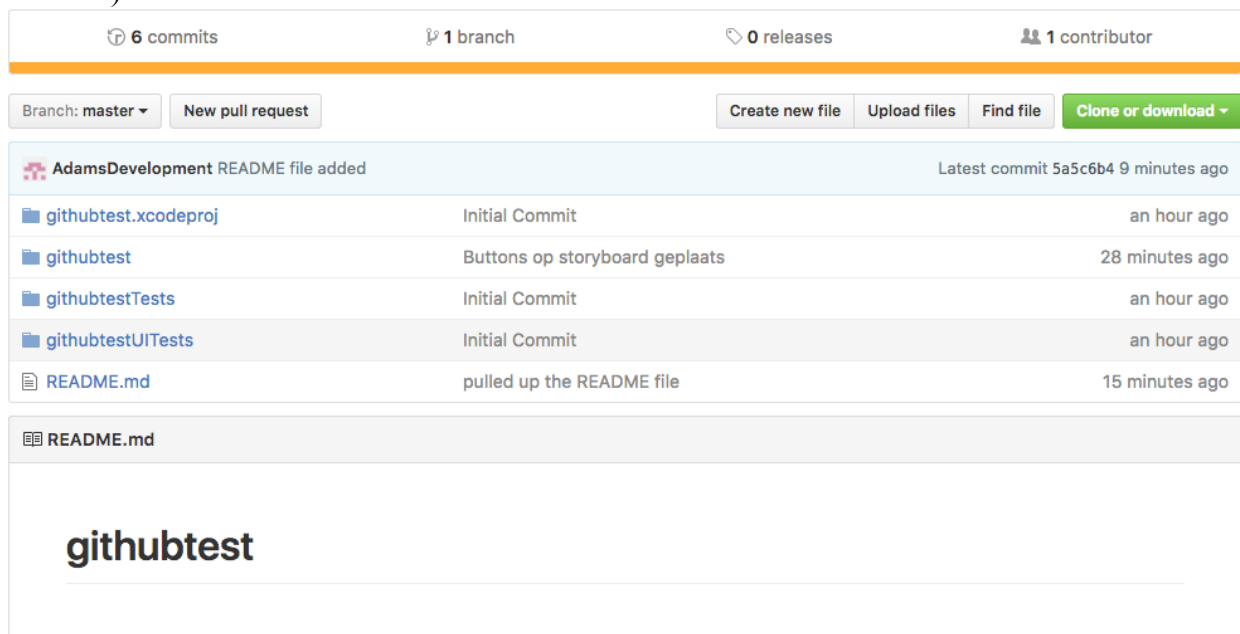
10. Als je nu je git status checkt, krijg je 'working directory clean' te zien. Alles is nu in orde om je project volledig op github te zetten.

Als je nu in je xcode project map 'ls' typt in terminal. Dan zie je dat de readme file er niet bij staat. Terwijl deze wel op github staat en bij dit project hoort. Deze willen we eerst nog mee in onze map zetten:

1. Met het command 'git pull origin master' kan je het readme filedje eraf halen. (origin is de remote die we zonet hebben aangemaakt. En master is de branch waarin we ons bevinden en bovendien op deze moment de enige branch die gemaakt is in ons project).

Soms kan je in een 'vim' terecht komen. Dit is een bestand geopend in de terminal. Bv het readme fileetje. Als je dit wil sluiten typ je :x en druk op de 'Enter' knop. Als je dit wilt testen kan je in je xcode project map, vanaf dat je de readme file van github hebt gehaald, 'vim README.md' dan zal het readme fileetje openen in je terminal en kan je deze hier wijzigen.

2. Check nu nog eens je git status en zorg ervoor dat deze komt te staan op 'working directory clean'
3. Als dit klaar is gaan we alles terug naar github pushen met volgend command 'git push origin master'. Als je nu je github repository vernieuwd zie je alles er op staan. (afbeelding 12)



Afbeelding 12

### Nu gaan we een nieuwe branch aanmaken:

Stel dat jij als developer instaat voor in dit project de imageViews toe te voegen. Dan maken we voor jou een nieuwe branch met imageViews.

1. Gebruik de command 'git checkout -b imageViews' om een branch aan te maken met de naam 'imageViews'. Je zal automatisch verplaatsen naar deze branch.
2. Als je nu in je xcode project een imageview toevoegd op je storyboard. Terug je git status checkt en je gewijzigde files terug add en commit, dan kan je met volgende command ze pushen naar de nieuwe branch, 'git push origin imageViews'.

3. Vernieuw je github repository nu en je zal zien dat er bovenin 2 branches zijn komen te staan. Wat ook opvalt is dat in je master branch die normaal nog open staat niets verandert is. Maar ga je zien naar de 'imageViews' Branch dan staat daar het hele project weer op met de nieuwe commit.