**Waarom GIT gebruiken?**

- Graag willen we de veranderingen in de code van een bepaalde applicatie bijhouden. Als er dan iets misgaat kunnen de veranderingen worden teruggedraaid.

- Er kan gewerkt worden aan verschillende versies. Denk bijvoorbeeld aan een werkend stukje software waaraan je een nieuwe feature wil toevoegen. Dan is het praktisch om dit in een losgekoppelde ruimte te doen, zodat de werkende code blijft bestaan.

- Het is een makkelijke manier om in een team software te ontwikkelen. Er is één plek waar de geschreven code staat opgeslagen (de repository). Verschillende ontwikkelaars kunnen daar code aan toevoegen.

**Git Bash**

**-** Dit is de command line interface van windows om Git te gebruiken.

**De opbouw van Git commands**

<programma> <flags> <data>

- Met het eerste commando sturen we een programma aan.

- Met een flag sturen we bepaald gedrag aan.

- Met data geven we input mee.

**Commands**

- “pwd” 🡪 Present working directory. Laat zien waar we ons op dit moment bevinden.

- “ls” 🡪 List. Laat zien welke bestanden er allemaal in de huidige locatie staan.

- “mkdir” + mapnaam 🡪 Make directory. Maak een nieuwe map aan. Typ de naam van de nieuwe map achter mkdir.

- “cd” + locatie 🡪 Chance directory. Verplaats jezelf naar een nieuwe locatie.

- “cd’ + .. 🡪 Change directory. Verplaats jezelf een map terug.

- “git” 🡪 Een help document. Je krijgt alle arguments te zien die je met een git command kan meesturen.

- “git init” 🡪 Maakt van de huidige locatie een git repository.

- “rm” 🡪 Remove.

- “touch” + test.txt 🡪 Maakt een tekst bestand aan. Je kan hiermee ook andere bestanden aanmaken.

**Commands met flags**

- “ls -a” 🡪 List met verborgen bestanden. Laat alle bestanden incl. verborgen bestanden zien.

**Shortkeys**

- CTRL + L 🡪 Leegt het scherm van de command line.

**De manual command**

Het is mogelijk om de “man” command the gebruiken om te zien welke flags je er aan mee kan geven. “man” staat in dit geval voor manual.

**Git repository**

Een Git repository is een verzameling van alle bestanden die we toegevoegd hebben / in Git hebben staan plus alle veranderingen / versies. Je kan dus ook heen en weer springen door de verschillende versies in je repository.

**Bestanden toevoegen aan repository**

- Manier 1: de volledige bestandsnaam uitschrijven: “git add test.txt”.

- Manier 2: alleen een punt schrijven. Dit voegt alle bestanden in de Git repository toe, ook in de mapjes naar beneden.

**Git commit**

Nadat een bestand is toegevoegd aan de repository moet deze nog worden ge-commit. Na het toevoegen van een bestand is deze ge-staged, oftewel klaar om ge- commit te worden. Met het committen van een bestand moet er altijd een bestand worden meegegeven. Ook moet je aangeven wie de auteur is van het bestand of wie de aanpassingen heeft gemaakt. Als je niet aan de juiste voorwaarden voldoet kan je in de VI terrecht komen, een tekst editor in de command line.

Als je uit de VI wil komen moet je op escape drukken. Zorg dat je onderaan in het scherm “:wq” kan typen.

Wat is nou precies het doel van eerst een bestand adden en daarna pas committen? Met de git status command kan je zien welke bestanden er klaarstaan (deze zijn ge-staged). Je krijgt dus nog een kans om je bestanden even te checken voordat je ze echt daadwerkelijk commit. Wanneer er iets gecommit wordt is het heel lastig om de veranderingen nog aan te passen.

Om een commit te maken voeg je ook een flag toe namelijk -am (add message). Hiermee voeg ik een bericht toe / een beschrijven wat deze commit precies inhoud. Als het bestand al ge-add is dan hoef je alleen -m te gebruiken.

**Branches**

- Vroeger werd de main-branch altijd Master genoemd. Vanwege de nare interpretatie van de naam Master worden veel main-branch nu Main genoemd. Ik heb Git ook op die manier geïnstalleerd.

- Als je een nieuwe feature wil toevoegen aan een werkend stukje software kan je een nieuwe branch aanmaken. Dit doe je door middel van git checkout -b “NaamVanBranch”. De -b flag staat voor een nieuwe branch.

- Als ik nu het bestand aanpas en commit in de nieuwe branch en ik ga weer terug naar de main branch. Dan zal ik deze veranderingen niet zien als ik het bestand open.

**Merging branches**

Wanneer je een werkend stukje code wil toevoegen aan de main branch moet je de twee branches mergen. Dit doet je altijd op een “trekkende” manier. Je gaat dus naar de branche toe waar je de verandering “naar toe wil trekken”.

Met de command “git merge testbranch” kan ik de main- en testbranch met elkaar mergen.

**Good practices**

- Laat aan je collega’s weten wanneer je branches gaat mergen. Als meerder collega’s op hetzelfde moment branches gaan mergen dan kunnen er hele grote merge conflicten ontstaan.

**GitHub**

Github is niets anders dan een plek waar je je git repository kan hosten. GitHub is wel één van de meest gebruikte platformen maar het is zeker niet de enige!

**Github repository inladen**

1. Ga naar Github.com en zoek de juiste repository op.

2. Klik op de groene code download knop, selecteer HTTPS en kopieer de link.

3. Zorg dat je in Git Bash niet in een andere repository zit.

4. Schrijf de command: git clone LINK-VAN-REPOSITORY.

\* Handig om te weten: CTRL + V werkt niet in Git Bash. Je moet op shift insert drukken om te plakken.

**Git Ignore file**

Met Git Ignore kan je bepaalde bestanden niet meenemen in je git repository. Met MAC maakt de finder bijvoorbeeld een .ds-store nogiets bestand aan. Deze wil je niet continu uploaden.