Versie beheer

Zodra je een tijd bezig bent met programmeren, kan het zijn dat je iets meer wensen hebt voor het opslaan van je code.

Misschien heb je eindelijk je code werkend, maar wil je iets anders proberen? Als je nieuwe poging niet handig blijkt, zal het heel moeilijk worden om terug te gaan naar je oude werkende versie.

Misschien wil je op je laptop werken, maar ook thuis op je desktop.

Misschien wil je samenwerken met andere mensen. Hoe krijg je jouw code op hun laptop, en andersom?

Daar hebben we versie beheer voor. Daarmee kan je:

- Meerdere veranderingen in bestanden allemaal in 1x opslaan met een omschrijving
- Terug naar de vorige versie van elk bestand die je zelf kiest
- Met meerdere computers/mensen werken aan hetzelfde project
- Het hele project zorgeloos stuk maken, zonder gevolgen

Git

Om dat allemaal voor je te regelen, heb je software nodig. Git is de populairste applicatie voor versie beheer. Je installeert het op je laptop (of via je code editor) en je kan het meteen gebruiken. Git kan alles wat hierboven vermeld is, en nog veel (veel) meer.

Er komen wel wat nieuwe concepten bij kijken

Repository

Een repository is een opslagplaats. In een repository stop je doorgaans één project - denk b.v. aan alle Java opdrachten, of een repository voor al je HTML opdrachten.

De repository maak je aan in een map (directory), en alles wat daarin staat (ook alle andere mappen!) horen bij het domein van Git.

Branch

In een repository wordt alle code opgeslagen in een `branch`. Vrij vertaald is het een `tak`, en dat is verrassend accuraat. Standaard werk je in de main branch (een `branch` genaamd `main`). Daarin wordt al je code opgeslagen. Waarom in een branch? Om makkelijk te kunnen samen werken.

Als je met iemand gaat samenwerken op 1 project, dan werk je allebei in een kopie van de code. Oftewel je eigen branch; een aftakking van de main branch. Als je klaar bent, en het werkt, ga je jouw aanpassingen weer samenvoegen in de main branch.

Zodra je allebei dezelfde code aanpast kan er wel een probleem ontstaan. Degene die het laatste zijn aanpassingen in de main branch wilt samen voegen, krijgt een conflict. Want zijn aanpassingen zijn op een oudere versie van de code en Git weet niet meer hoe deze samengevoegd moeten worden. Dat moet je dan zelf doen. Dat is een merge conflict.

Commit

In een branch sla je dus alle code op. Zodra je iets werkend hebt op je laptop, waar je blij mee bent, kan je je wijzigingen in 1x opslaan in Git. Alle wijzigingen in welk bestand dan ook, zolang ze in dezelfde git map staan. Dat doe je met een `commit`.

Een commit is dus alle wijzigingen vanaf de vorige opgeslagen versie van de code. Daarbij kan je een naam en een omschrijving toevoegen. Zo kan je makkelijk de vorige versies van code terug vinden.

Al je code tot nu toe is dus niets meer dan een lijst van commits; een serie van wijzigingen op wijzigingen op wijzigingen. Zo kan je heel flexibel terug springen naar de vorige versie van je code, in welk bestand dan ook, en raak je nooit meer iets kwijt.

GitHub

GitHub is het social media platform voor programmeurs. Je hebt een profiel, kan andere programmeurs volgen, je hebt een timeline, etc.

Maar GitHub gaat ook om *samenwerken*. Je mag daar gratis je Git repositories opslaan. Die kan je publiek toegankelijk maken, en dan staan ze op je profiel, maar dat kan ook privé en dan zie je ze alleen zelf.

Je kan je Git repository van je laptop naar GitHub wilt sturen, en ook de laatste wijzigingen van GitHub ophalen richting je laptop. Zo kan je dus heel makkelijk samenwerken met anderen: Je hebt allebei dezelfde repository op je laptop, maar je stuurt je wijzigingen naar GitHub en haalt ze daar ook weer op.

Pull requests (of merge requests)

Het is niet altijd even handig om iedereen in je project zomaar alle code aan te laten passen. Voordat iemands code dus in de main branch wordt samengevoegd, kan je dat laten controleren door andere projectleden; dat noem je een pull request of merge request.

Je maakt een pull request aan op GitHub, om een branch te laten samenvoegen naar een andere branch (de `main`, doorgaans). Alle teamleden kunnen de code controleren, en goedkeuren of comments plaatsen. Daarna kan je met een druk op de knop de code samenvoegen (of *mergen*).