Git & Github

Git is een krachtig en populair versiebeheersysteem (VCS) dat ontwikkelaars helpt bij het beheren van wijzigingen in de code en het verbeteren van de samenwerking tussen teamleden.
GitHub is een webgebaseerde service die Git-hosting biedt voor het beheer van softwareprojecten en om met anderen samen te werken aan projecten.

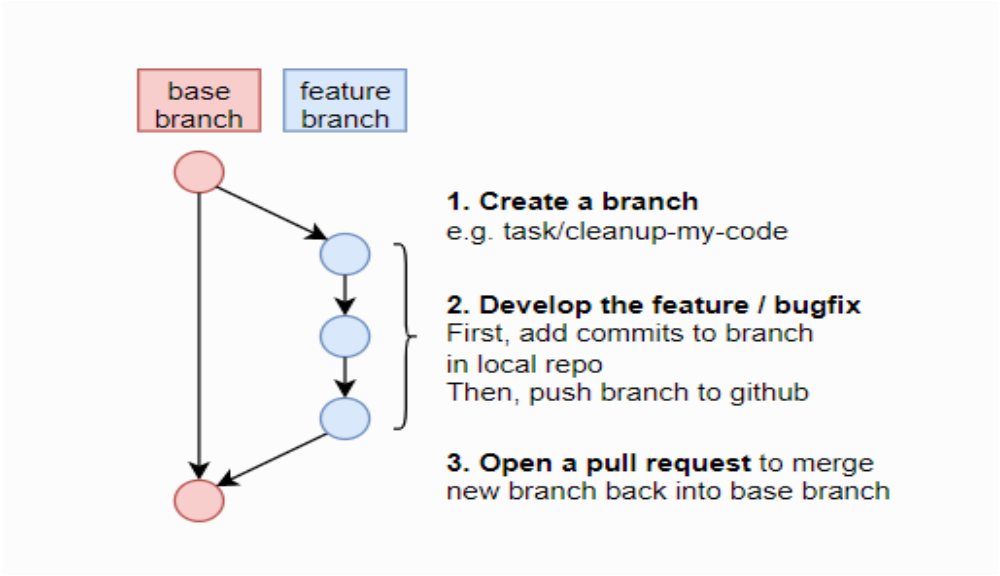
Waarom

Git gebruiken we om verschillende redenen:

1. **Versiebeheer**: Git stelt ons in staat om wijzigingen in de code te beheren en een geschiedenis bij te houden van wie welke wijzigingen heeft aangebracht en wanneer. Dit maakt het gemakkelijker om bugs op te sporen en problemen op te lossen.
2. **Samenwerking**: Git maakt het gemakkelijk voor meerdere ontwikkelaars om samen te werken aan dezelfde code, waardoor ze efficiënter kunnen werken en elkaars werk kunnen bekijken en beoordelen. Dit helpt bij het voorkomen van conflicten in de code en maakt het gemakkelijker om wijzigingen in de code te integreren.
3. **Branching en merging**: Git maakt het gemakkelijk om te werken met meerdere branches van de code, waardoor ontwikkelaars kunnen werken aan verschillende functies zonder de hoofdversie van de code te verstoren. Wanneer deze functies gereed zijn, kunnen ze eenvoudig worden samengevoegd met de hoofdversie van de code.
4. **Back-up**: Git slaat alle wijzigingen in de code op, zelfs als deze per ongeluk zijn verwijderd. Hierdoor kan verloren code gemakkelijk worden hersteld.
5. **Continuïteit** van ontwikkeling: Git stelt ons in staat om de codebase continu te verbeteren en te evolueren. Ontwikkelaars kunnen nieuwe functies en verbeteringen aanbrengen zonder dat dit de bestaande functionaliteit van de code beïnvloedt.

Belangrijke begrippen

- Een **repository** is een centrale opslagplaats waarin alle bestanden en alle geschiedenis van een project worden opgeslagen.
- In de centrale repository staat de hoofdcode, ook wel de "**master branch**" genoemd.
- Een **branch** is een afsplitsing van de "master branch", waarin ontwikkelaars kunnen werken zonder de hoofdcode te beïnvloeden. Hierdoor kunnen ontwikkelaars afzonderlijk aan verschillende functies werken zonder elkaar te beïnvloeden. Er zijn de volgende soorten branches:
    - **Main branch** - de hoofdbranch, die de meest recente stabiele versie van de code bevat. Dit is de branch die in productie zal worden gebruikt. Ook wel master branch of base branch genaamd.
    - **Feature branch** - een branch die wordt gebruikt voor het toevoegen van nieuwe functies aan de code. Nadat de functie is voltooid, wordt de feature branch samengevoegd met de hoofdbranch.
    - **Bugfix branch** - een branch die wordt gebruikt voor het oplossen van bugs in de code. Nadat de bug is opgelost, wordt de bugfix branch samengevoegd met de hoofdbranch.
    - **Release branch** - een branch die wordt gebruikt voor het voorbereiden van een nieuwe release van de code. Dit wordt gedaan om alle nieuwe functies en bugfixes te testen voordat ze worden samengevoegd met de hoofdbranch.
- Wanneer de code in een branch is voltooid en getest, kan deze worden samengevoegd met de hoofdcode door middel van een "**merge**".



Werken met GIT als Se'er

1. **Pullen**: haal **eerst** de meest recente versie van de code op: Voordat je begint met werken aan de code, moet je ervoor zorgen dat je de meest recente versie van de code hebt opgehaald van de hoofdbranch.
   Als je *nog niet beschikt* over een lokale kopie van het project, is het nodig om te **clonen** om een lokale kopie van de code te maken. Clonen houdt in dat je een volledige kopie van de repository maakt, inclusief alle branches en commit-geschiedenis. Dit is handig als je vanaf het begin aan het project wilt werken en de complete geschiedenis nodig hebt om een volledig beeld te hebben van de ontwikkeling van de code.
2. Controleer of er **conflicten** zijn: Controleer of er conflicten zijn tussen de wijzigingen die je (eventueel) al hebt gemaakt en de wijzigingen van anderen die je ophaalt door te pullen.
3. Maak een **nieuwe (feature) branch** aan de wijzigingen in de code aan te brengen.
4. Maak **wijzigingen** in de code op je eigen feature branch.
5. Voeg wijzigingen toe aan de **staging area.** Een stage is een tijdelijke opslagplaats voor wijzigingen die je wilt committen naar de repository. Het stelt je in staat om selectief te kiezen welke wijzigingen je wilt opnemen in de volgende commit.
6. **Commit** je wijziging(en). Een commit in Git is een actie waarbij de huidige wijzigingen in het codebestand worden vastgelegd in de lokale repository. Hierbij wordt een korte beschrijving gegeven van de wijzigingen die zijn aangebracht in de code. Een commit creëert als het ware een nieuw punt in de geschiedenis van de code, waarbij alle wijzigingen die tot dat punt zijn aangebracht worden vastgelegd.
7. **Push** de wijzigingen naar de hoofdbranch om deze beschikbaar te maken voor anderen.
8. Doe een **pull request**. Deze doe je nadat je wijzigingen hebt gecommit en gepusht naar de feature branch en je klaar bent om deze te integreren met de hoofdbranch. Je vraagt dan aan anderen om jouw wijzigingen te bekijken en te beoordelen voordat ze worden samengevoegd met de hoofdbranch