Versiebeheer hand-out 4

In de vorige hand-out is er een uitleg geweest over hoe branching gedaan kan worden en hoe de basis 3 branchingstructuur werkt. In deze hand-out gaat het over het tegenovergesteld van branching: Mergen en pullen. Dit zijn de onderwerpen:

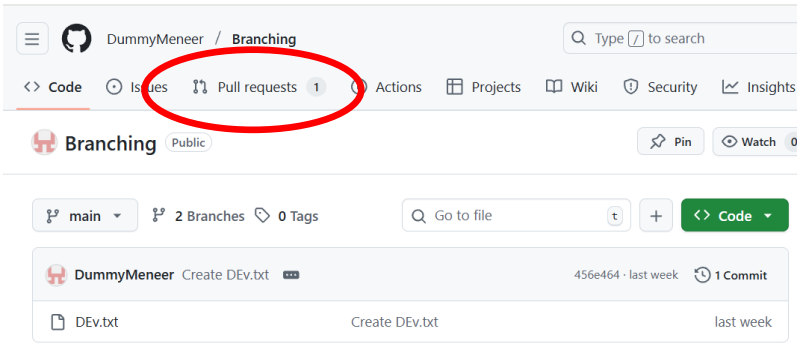
- Mergen en Pullen
- Pull requests
- Rebase
- Squash
- Merge conflicts

Inhoudsopgave

Opdracht 1a: Merge 1 branch terug

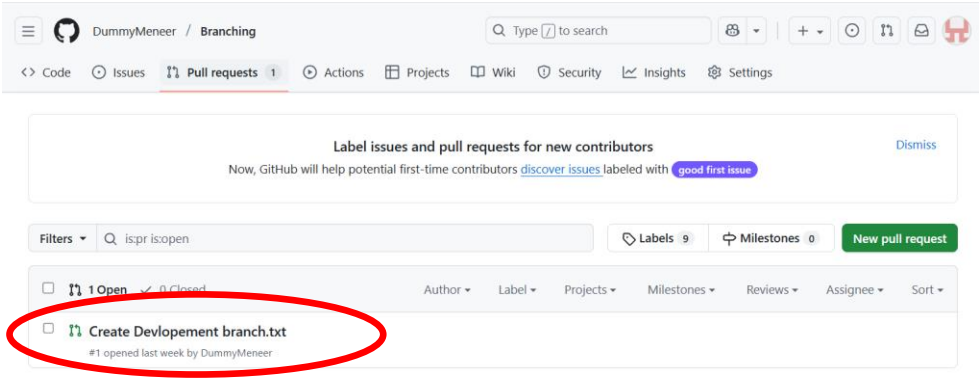
Mergen is het tegenovergestelde van branchen. Deze actie laat mensen toe om informatie of code van een branch over te zetten naar een andere branch. Dit kan gedaan worden om meerdere redenen maar de meest voorkomende reden is omdat er een stuk code ergens anders nodig is om voortgang te maken bij het project. Er zijn twee merges die te vinden zijn in GitHub, eentje is al gegenereerd voor je als je een branch aan maakt. Deze merge staat klaar en kan op ieder moment gepakt worden om de branch weer terug te zetten naar de branch waar die vandaan komt.

Om een branch te mergen via de gegenereerde merge ga je naar je code overzicht en ga je naar de pull requests. Dit tabblad kan je vinden bovenin naast issues.

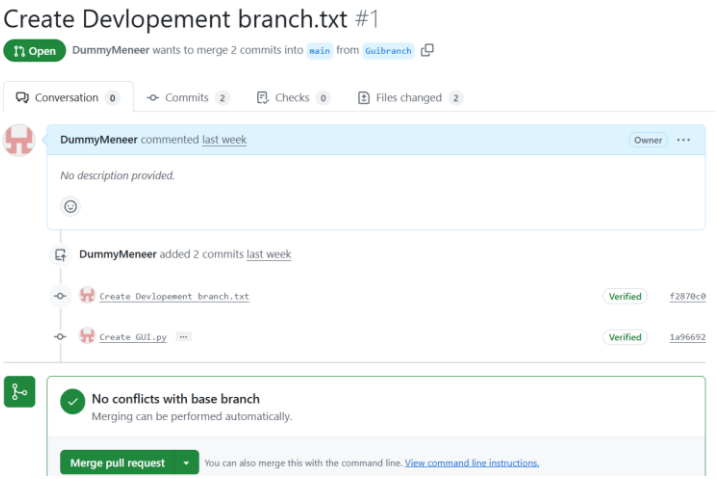


In dit overzicht staan alle pull requests (gangbaar ook genoemd merge requests)

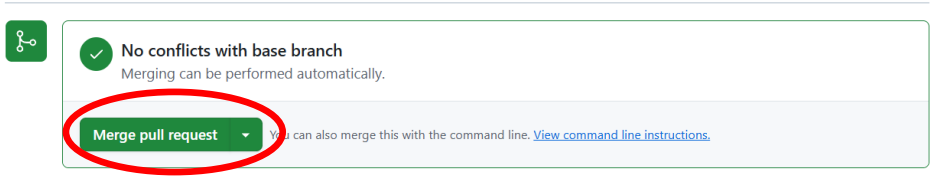
In dit overzicht zien we dan ook de commit die is gemaakt tijdens het aanmaken van de Guibranch.



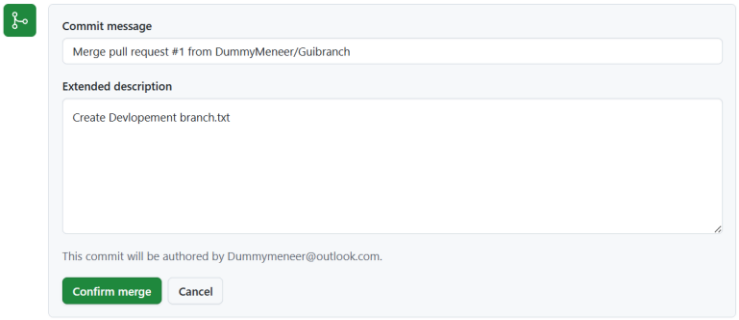
Als we hier op klikken krijgen we de details van deze merge te zien. Hier staan de beschrijvingen van de merge, welke commits het mee neemt en of de merge in orde is. Verder hieronder is ook een chat gebied waar groepsleden of jijzelf notities kunnen achterlaten voor elkaar of discussies kunnen houden.



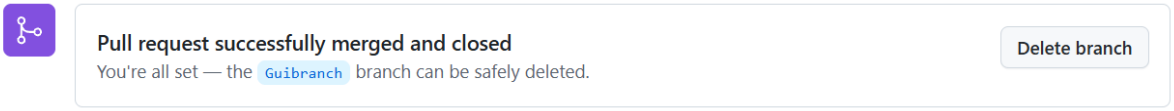
Zoals te zien in de vorige afbeelding is er een groen gebied met “no conflicts with base branch” dit betekent dat de code die we naar de branch willen mergen geen problemen veroorzaakt en GitHub laat ons toe om het uit te voeren. We klikken dan op de groene knop met Merge pull request.



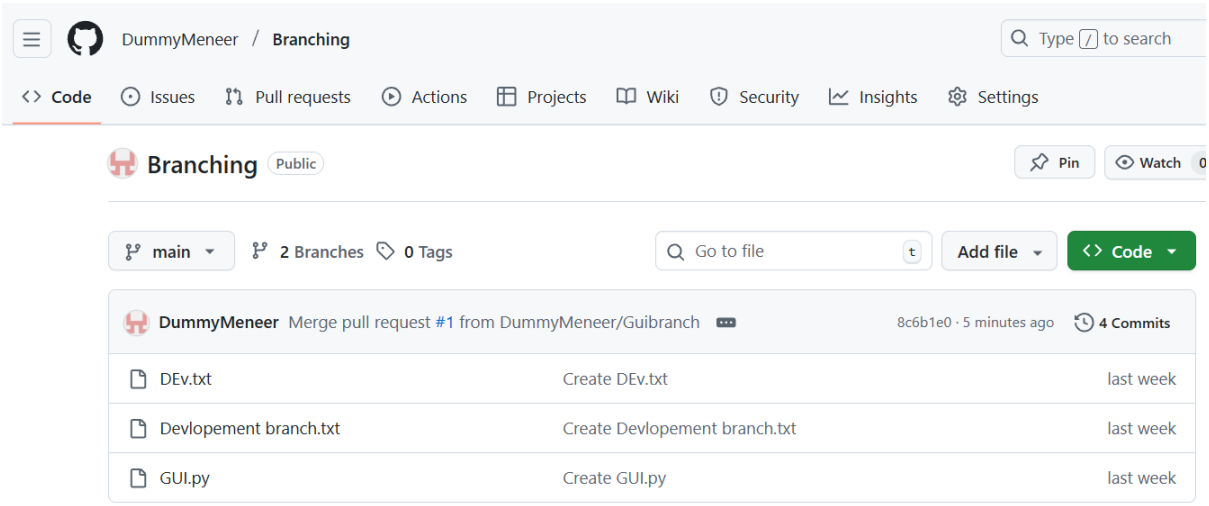
Dit opent een commit veld. Hier vul je indien dat niet is gegenereerd een gepaste naam voor de commit en een beschrijving van wat er wordt gemergd. Dit voorkomt dat je later moet terugzoeken naar welke commits/functies hierin zaten. Vervolgens druk je op confirm merge.



Het kan zijn dat GitHub je vraagt of je de branch wil deleten. Dit is een permanente actie en verwijderd alle andere info. Dit kan zeer nuttig zijn als je een squash wil uitvoeren, meer hierover in opdracht 4. Voor nu negeren we dit.



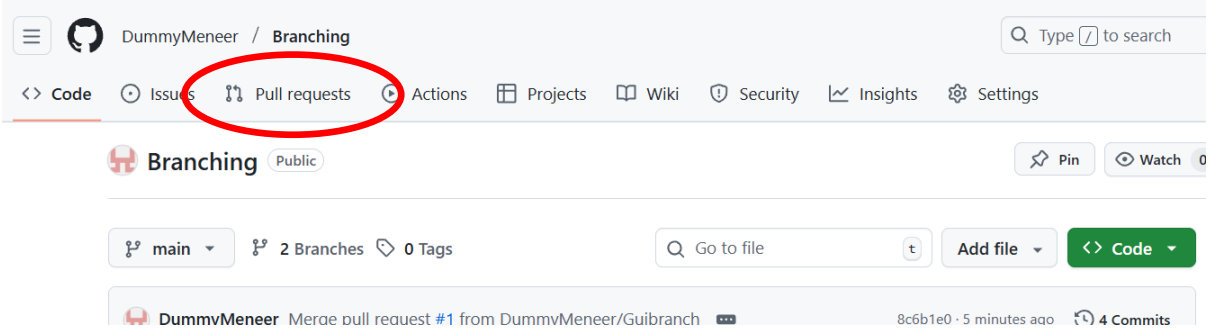
Als we nu teruggaan naar het code overzicht dan zien we dat de bestanden van de branch nu ook op de main staan.



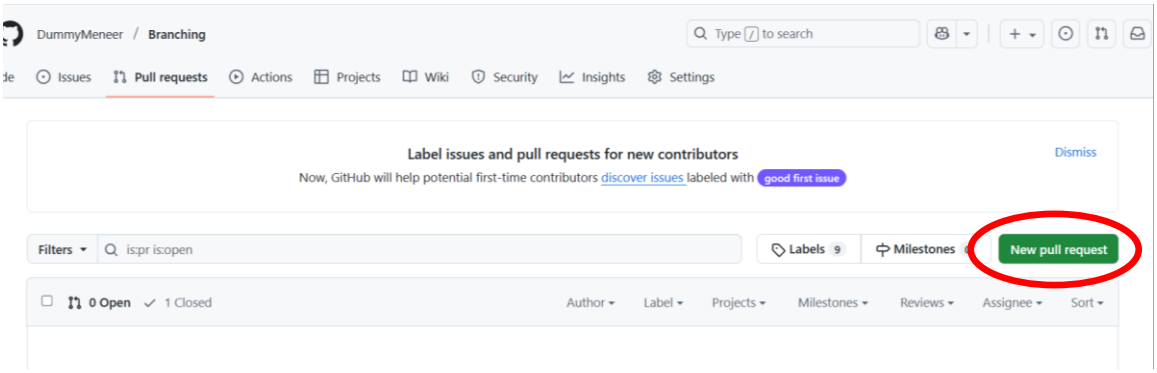
Probeer in een test repository een branch te maken met daarin een commit en deze te mergen met de main aan de hand van de automatisch gemaakte merge.

Opdracht 1b: Pull requests

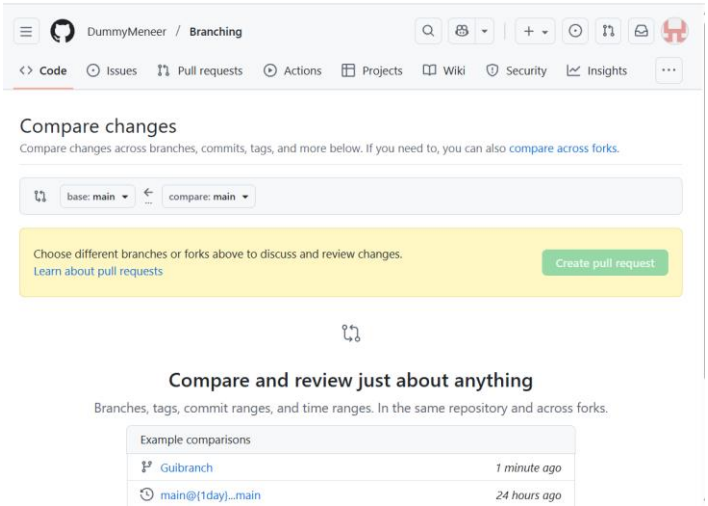
Het kan voorkomen dat je vaker moet mergen met een branch, misschien omdat je een paar extra toevoegingen hebt gemaakt of omdat je een bug wilde repareren. Als je dan al de gegenereerde merge hebt gebruikt dan moet er een nieuwe komen. Hiervoor gaan we naar de pull requests tab om naar het overzicht te gaan.

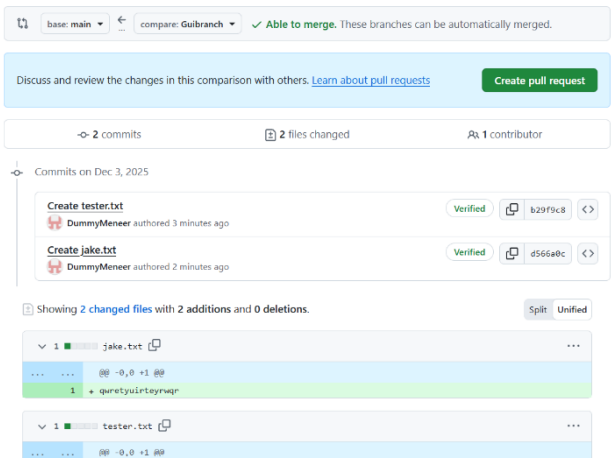


In het overzicht gaan we naar “New pull request” en klikken hierop.



Dit opent het menu voor de pull requests. Hierin zien we een setje dropdowns met daarin de branches. De branch waar je naartoe merget is de eerste en de tweede is de branch waarin je code hebt staan die gemergt moet worden met de rest. Standaard zal GitHub hier in beide velden Main zetten. Dit is uiteraard niet de bedoeling, want we willen iets van de Guitbranch hebben. We klikken op de tweede dropdown en kiezen de juiste branch.

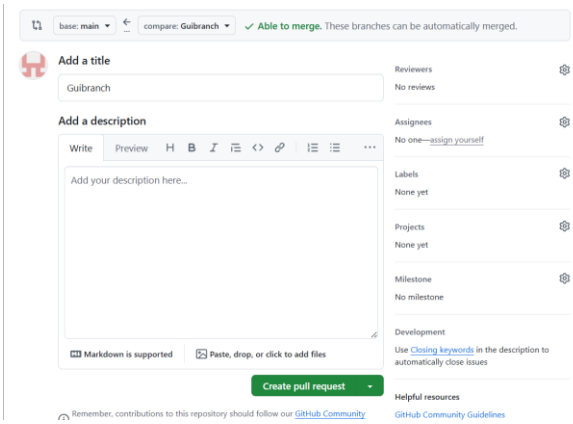


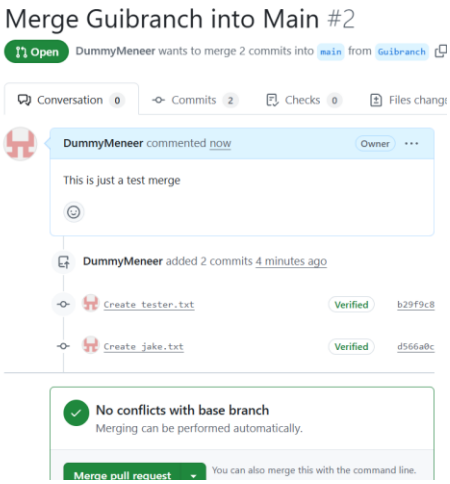


GitHub geeft voor ons dan netjes aan dat de Guibranch 2 commits bevat met 2 nieuwe bestanden. Als dat klopt kan er op de “Create pull request” geklikt worden.

Er wordt dan gevraagd om de naam en beschrijving van de merge en deze is vaak niet genereerd. Vul deze in volgens je commit regels en druk op “Create pull request”

Dit opent het overzicht ervan en laat je toe de merge door te zetten of om nog een bericht achter te laten voor degene die het moet goedkeuren.

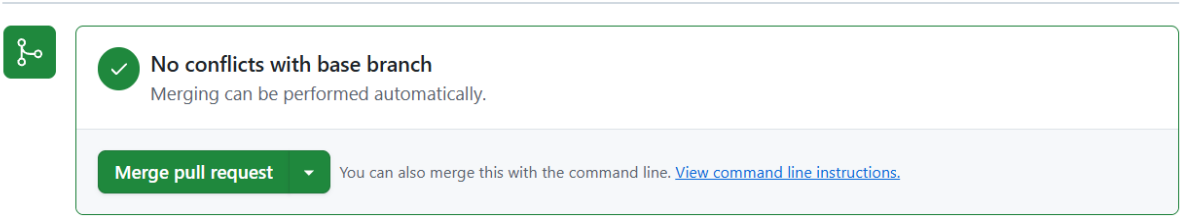




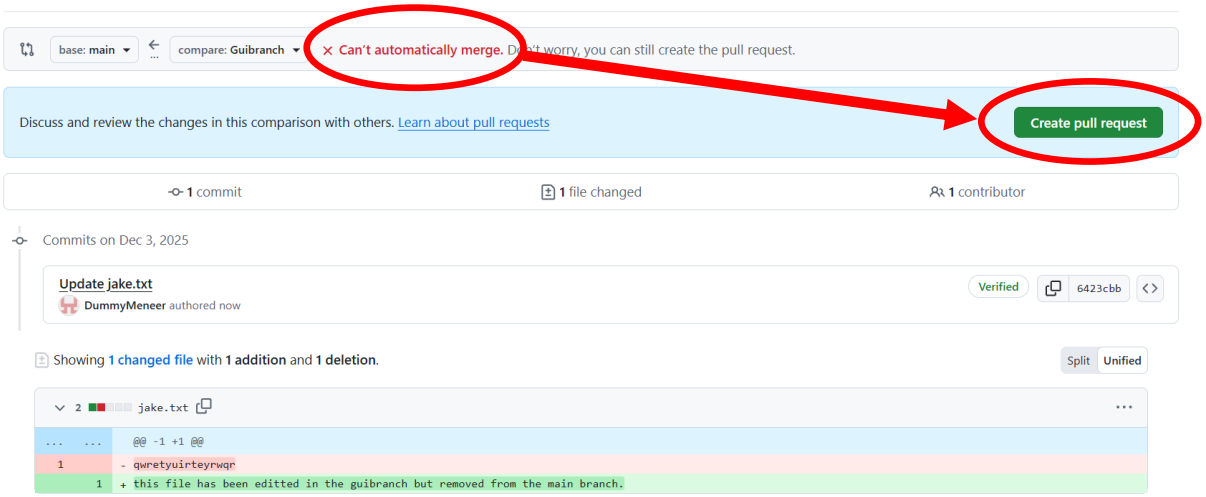
Probeer nu zelf een pull request te maken zonder een nieuwe branch aan te maken.

Opdracht 2: Merge conflicts

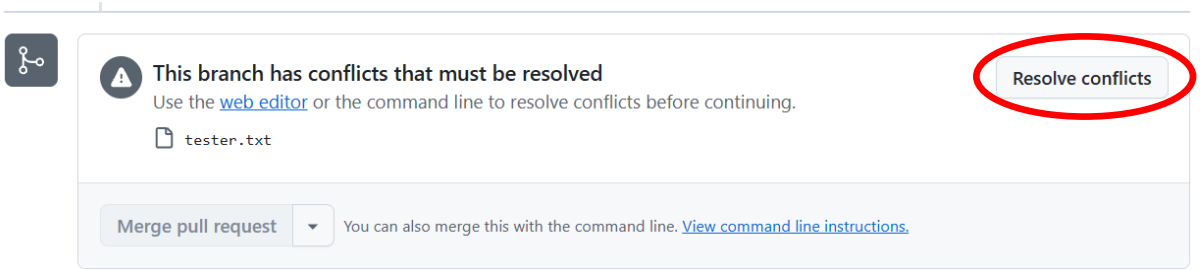
Merge conflicts zijn situaties waarin je een versie van een bestand met een andere versie wil combineren maar deze spreken elkaar tegen. In kleine groepen of projecten zal dit niet veel voorkomen. Het is wel belangrijk om hier mee om te leren gaan. Zoals we hebben gezien bij de twee merges hiervoor laat GitHub ons zien of een merge in orde is. Als er geen conflict is geeft het een groen vinkje zoals hieronder.



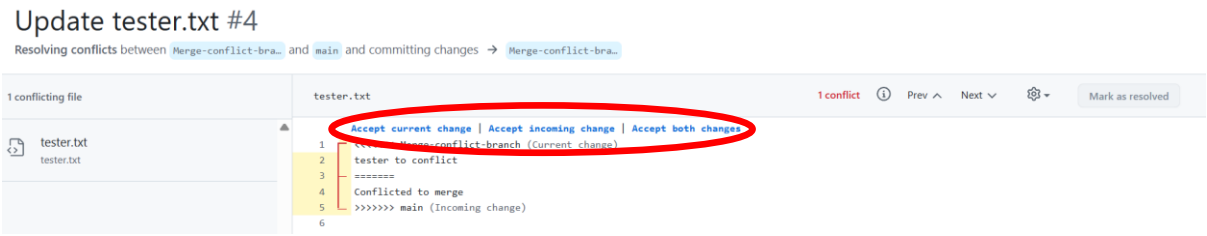
Het kan zijn dat bij het maken van een merge request dat GitHub een bericht geeft dat de merge niet gelijk afgehandeld kan worden. Zoals te zien is hieronder. Dit betekent dat er mogelijk een merge conflict is ontstaan. Je kan dan alsnog de pull request maken maar mogelijk nog niet afronden.



Als er wel een merge conflict is dan is dat te zien aan een grijs waarschuwingsteken. Dit kan zijn omdat er op beide branches aan hetzelfde bestand is gewerkt en nu dus twee versies heeft die van elkaar verschillen. Om dit probleem op te lossen klik je op Resolve conflicts.

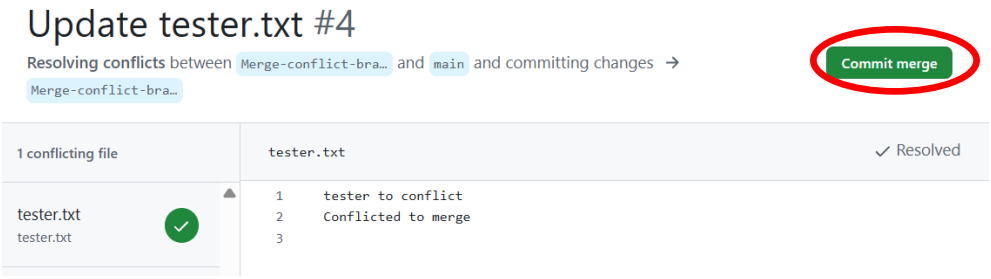


Dit opent een venster met daarin een overzicht van conflicterende bestanden (links) en de code die conflicterend is. GitHub geeft dan aan welke opties beschikbaar zijn als oplossing: hou de code van de huidige branch, hou de code van de mergende branch of voeg ze beide samen. Liggend aan de afspraken van de groep is de ene oplossing beter dan de ander. In dit geval gebruiken we beide.

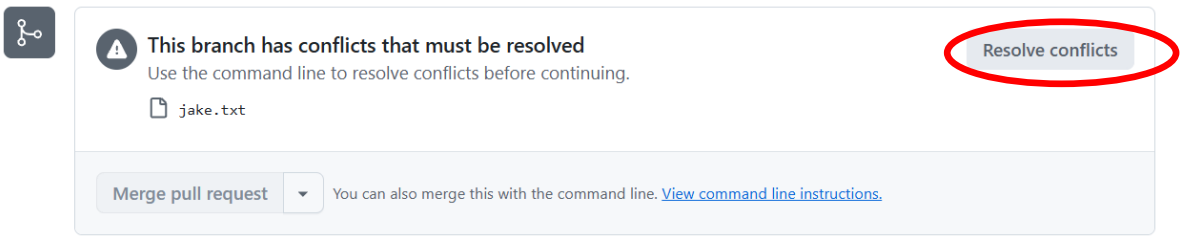


Nadat we hierop hebben geklikt zien we de rode lijn verdwijnen en ziet de code er vervolgens zo uit. We kunnen als alles afgehandeld is dan het conflict markeren als resolved aan de rechterzijde. Dit geeft aan bij GitHub dat dit bestand klaar is voor de merge. Mochten er meer bestanden links staan moet dit gedaan worden bij ieder bestand waar een probleem zich in bevindt.

Als alles opgelost is kan de merge gecommit worden. Er zal een knop oplichten met commit merge.



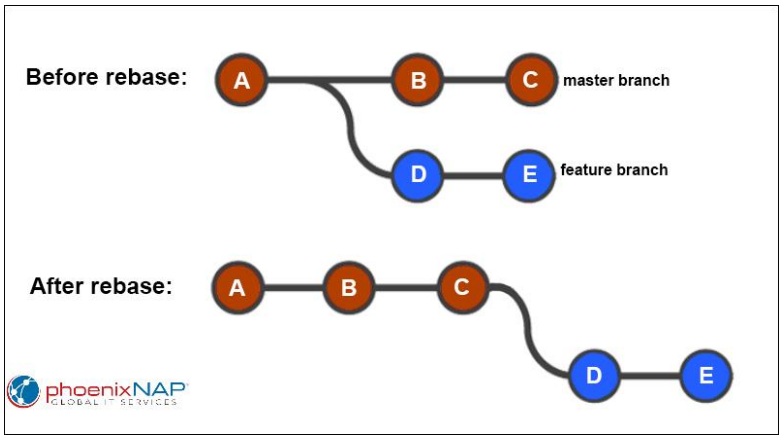
Het kan zijn dat een merge conflict te complex is voor GitHub om op te lossen. Dan is de knop van resolve conflicts niet klikbaar. Deze zal vaak voorkomen als je in een branch een bestand aanpast terwijl in de andere branch het bestand verwijderd is. Dit kan opgelost worden via de Command line of door het bestand opnieuw aan te maken in de branch waar deze mist.



Probeer nu zelf een merge conflict te maken door op twee branches hetzelfde document aan te passen met iets anders en de twee branches te mergen.

Opdracht 3a: Rebase branches

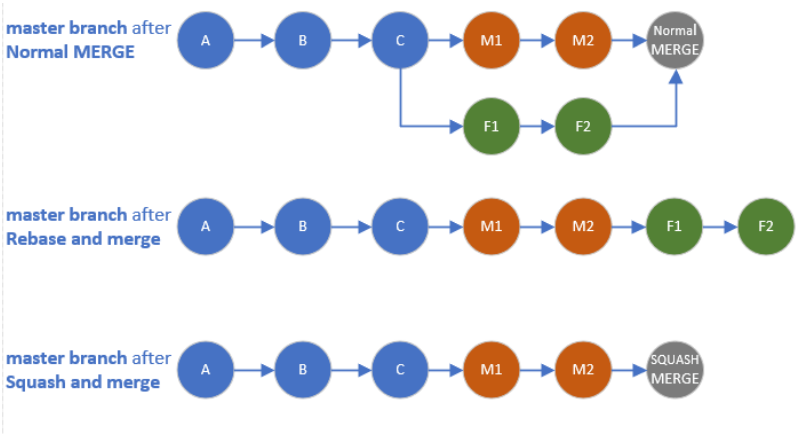
Er zijn nog twee andere merge soorten die belangrijk kunnen zijn om te begrijpen. De eerste is de rebase. Het kan zijn dat de development branch een paar updates heeft die jouw feature branch kan gebruiken. Dan is het verstandig om een merge uit te voeren van de development branch naar jouw feature branch. Op deze wijze heb je de meest recente versie van de code en voorkom je dubbel werk. Deze merge heeft de naam rebase gekregen. Omdat de base van de feature branch code opnieuw wordt gedefinieerd. Hieronder een afbeelding ter visualisatie.



Bron afbeelding

Opdracht 3b: Squash branch

De tweede merge heeft de squash merge, dit zorgt ervoor dat alles wat er tegelijk gebeurd is op een feature branch wordt samengevoegd tot 1 merge commit. Binnen GitHub kan dit bereikt worden door een feature branch te mergen met de development branch en dan de feature branch te verwijderen. Dit kan voor zeer grote projecten nuttig zijn aangezien de hoeveelheid branches anders uit de hand loopt. Voor kleine projecten, zeker voor je portofolio is dit afgeraden aangezien werkwijze een bonus punt kan zijn voor sollicitaties.



Bron afbeelding

Opdracht 3c: Merge vs rebase vs squash

In theorie zijn alle drie de merges even belangrijk voor de situaties waarvoor ze gebruikt worden. Ga voor jezelf na welke je wel en niet zou gebruiken in je eigen projecten.

Opdracht 4: Merge benamingen

Merges benoem je vaak zoals GitHub ze aanmaakt, en anders Merge “Branchnaam” naar/to “Origine branchnaam”. Bijvoorbeeld: “Merge Login to UI” of “Merge fogBranch to GameMapBranch”

De details en redenen zet je vervolgens in de beschrijving van de merge. Deze beschrijving is zeer noodzakelijk aangezien in sommige projecten een verantwoording nodig is om een merge uit te voeren en deze wordt dan hier uitgelegd.

Voeg Merge benamingen toe aan je lijstje van consistente benamingen.

Opdracht 5: Merge afspraken

Een belangrijk aspect van mergen is dat er goede afspraken over gemaakt worden. Op de main staat vaak software die op dat moment al live draait, dit betekent dat als je een stuk code ernaartoe mergt en die niet werkt, je live programma’s onderuit kan halen. Dit is niet gewenst

Om deze reden merge je NOOIT code die defect is naar een vitale branch, ergo niet de Main of Dev branch.

Wanneer Merge je naar **binnen** bij de basis 3 structuur:

- Als je feature getest is en goed werkt mag deze naar de dev branch
- Als de Dev branch een werkende en gecontroleerde versie bevat van het product mag deze naar de Main branch

Wanneer merge je naar **buiten** bij de basis 3 branch:

- Als je een feature branch wil updaten met recentere project code
- Als je tussen feature branches code nodig hebt. Dit wordt bij grote projecten afgeraden, dit kan leiden tot spaghetti beheer.

Wanneer Merge je **NIET**:

- Als er iets in de code kapot is of voor gebroken code kan zorgen.
- Als er code mist of niet afgemaakt is.
- Als je geen toestemming hiervoor hebt van de groep/teamleider/manager

Afspraken maken kan problemen voorkomen en orde bewaren in een project. Deze afspraken worden dan bewaakt door de gehele groep of door een aangestelde bewaker.

Voorbeelden van afspraken:

- Geen merges maken tot iedereen de code heeft gecontroleerd
- Geen merges maken tot het onderdeel klaar is en werkt
- Alleen X mag merges maken
- Merges mogen gemaakt worden naar de work branch zonder overleg maar de dev branch vereist goedkeuring van een ander groepslid.

- Nooit mergen naar de master tot het einde van het project.

Deze voorbeelden kunnen voor sommige projecten of groepen werken en niet voor anderen.

Ga met je groepje zitten en spreek af wanneer je een merge uitvoert en hoe merges worden uitgevoerd. Hou er rekening mee dat iedereen minstens 1 merge moet maken.

Week 4 Eindopdracht

Ga een groepje maken van 3 tot max 4 mensen en maak een keuze voor een projectje. Maak de repo aan maar ga nog geen code schrijven of posten. Alleen de beschrijving van de inhoud van de repo en voeg de mensen toe die toegang horen te hebben. Volgende week mogen jullie starten na de les over projects.

Volgende week:

Volgende week worden de volgende onderwerpen behandeld:

- GitHub projecten
- Insights
- Issues

De weken erna zijn niet noodzakelijk voor het halen van het vak maar wel streng aangeraden aangezien er nuttige informatie wordt gedeeld. Er is geen actieve werkvorm van week 6 t/m week 8. Als je voor jezelf wilt werken dan mag dat, geef dit van tevoren aan bij de docent.