Git branches

Werken met branches

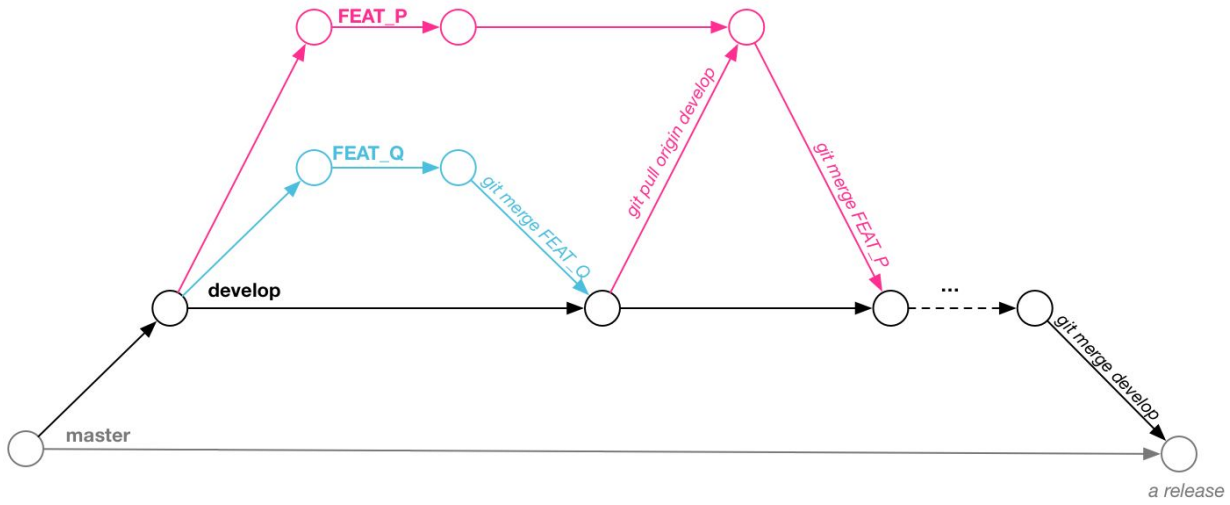
In git kun je werken in losse *'branches*'. Dit stelt je in staat om afzonderlijk aan verschillende delen van je code te werken, zonder dat wijzigingen elkaar in de weg zitten.

Wanneer je een nieuwe git repository aanmaakt, bevind je je in de *master* branch. De wijzigingen die je vervolgens commit worden aan deze branch toegevoegd. Vanuit een branch kun je een nieuwe branch aanmaken. De wijzigingen die je in deze nieuwe branch commit bevinden zich enkel in deze branch en niet in de master branch. Wanneer je klaar bent met het werken in een branch, kun je deze code toevoegen aan (mergen met) je hoofd-branch.

In projecten wordt de *master* branch vaak enkel gebruikt voor releases van het project, bijvoorbeeld om een werkende versie te kunnen testen met gebruikers. Naast het werken in de master branch, wordt een extra branch aangemaakt genaamd '*develop*'. Deze blijft het hele project bestaan.

Vanuit de *develop* branch worden losse branches aangemaakt. In deze losse branches worden features toegevoegd, wordt code anders gestructureerd, worden bugs gefixt… etc. Het is nuttig om een duidelijke naamgeving te hanteren voor branch-namen, zodat je de inhoud (wijzigingen /toevoegingen e.d.) goed kunt afleiden a.d.h.v. de naam van een branch. Een development team maakt vaak afspraken over de naamgeving van branches. Zo gebeurt het toevoegen van nieuwe functionaliteiten vaak in een branch genaamd *FEAT_<naam_functionaliteit>* en bijvoorbeeld een bugfix in een branch genaamd *BUGFIX_<issue_code>*.

In de onderstaande afbeelding zie je hoe er gewerkt kan worden vanuit de *develop* branch in plaats van de master branch.

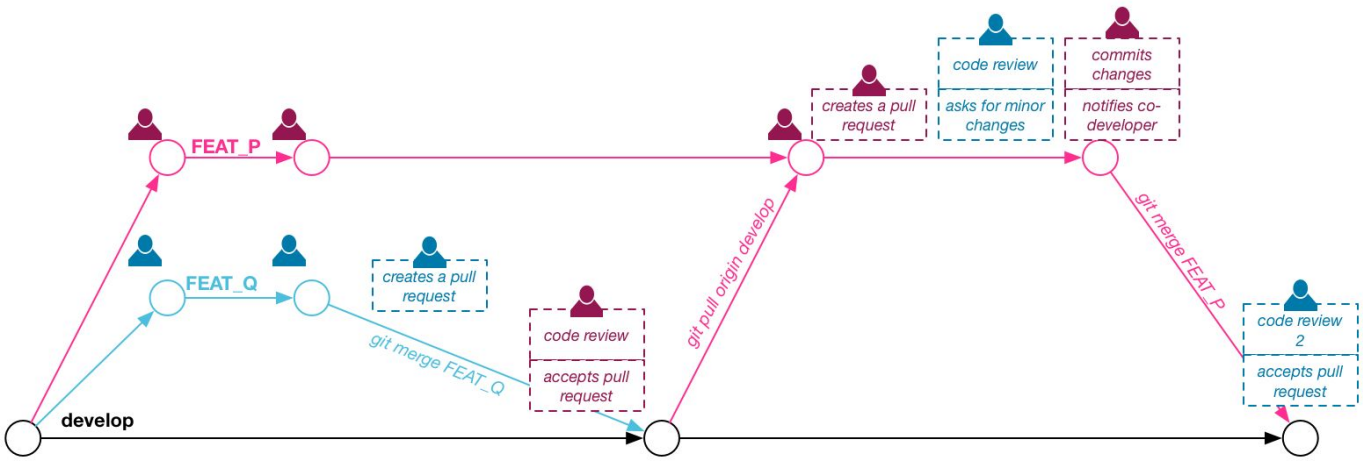


Wanneer je in een losse branch hebt gewerkt aan een afgebakende functionaliteit en wanneer dit onderdeel klaar en getest is, ben je klaar om de wijzigingen in deze branch toe te voegen aan de *develop* branch. Dit heet **merge**. Dit kun je doen vanuit de *command line*, maar ook in de remote

repository d.m.v. een zogenaamde '*merge request*' (op github heet dit een '*pull request*'). Het voordeel van het maken van een *merge request* is dat je een teamgenoot kunt vragen om online de wijzigingen eerst te reviewen, een zogenaamde *code review*. Het kan zijn dat je teamgenoot je naar aanleiding van een code review vraagt om nog iets aan te passen, voordat hij/zij de *merge request* wilt accepteren. Wanneer bijvoorbeeld een stukje code niet geheel duidelijk is, kan hij/zij vragen om hier een aantal comments aan toe te voegen ter verduidelijking/verantwoording van de gemaakte keuzes. De communicatie hierover kan via de online repository verlopen, het kan later in een project nuttig zijn om hierbij te kunnen.

Wanneer je een *merge request* wilt aanmaken terwijl de develop branch in de tussentijd is aangepast, is het belangrijk om zelf de nieuwe code uit de develop branch je eigen branch binnen te halen (`$ git pull origin develop`). Het kan namelijk zijn dat er zich anders *merge conflicts* voordoen. Deze *merge conflicts* kom je al tegen bij het binnenhalen van de nieuwe code uit de *develop* branch. Omdat jij het beste op de hoogte bent van de wijzigingen in jouw branch, is het efficiënter wanneer jij de *merge conflicts* oplost, dan wanneer je dat overlaat aan je teamgenoot.

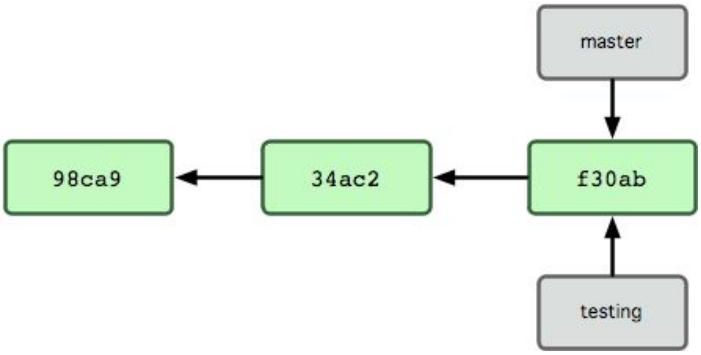
Ook in een project waarin je alleen werkt zonder anderen is het prettig om te werken in losse branches. Je kunt zo werken aan verschillende functionaliteiten zonder dat de afzonderlijke wijzigingen elkaar in de weg zitten. Je kunt dan tussendoor eenvoudig switchen tussen branches. Het gebruik maken van online *merge requests* is ook prettig in een solo-project. Dit creëert namelijk een moment waarop je kritisch naar je eigen wijzigingen kan kijken, in plaats van aan te nemen 'dat het wel goed zal zijn'. Zo kun je onder andere beter voorkomen dat er slordige code in je project terecht komt.



[1] bron: https://git-scm.com/book/nl/v1/Branchen-in-Git-Wat-is-een-branch

Relevante git commands

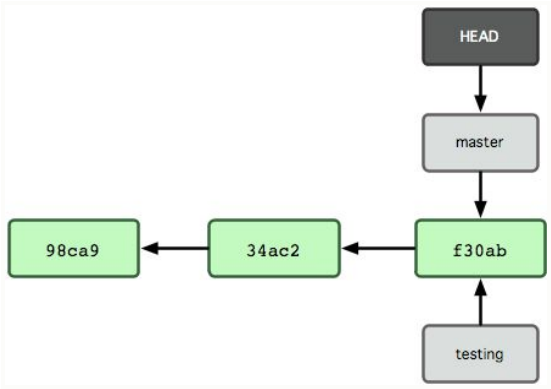
Het aanmaken van een nieuwe branch.
Let op: deze wordt aangemaakt vanuit de branch waar je je op dat moment in bevindt. Zie ook de afbeelding[1] hiernaast, waarin vanuit de master branch een nieuwe branch wordt aangemaakt.

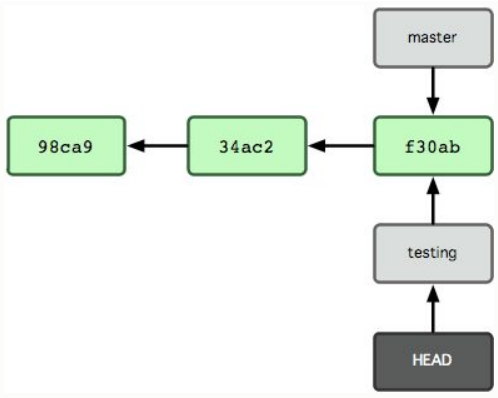


`$ git branch <branch-name>`
Maakt een nieuwe git branch aan genaamd <branch-name>.

Het uitchecken van een branch.
Git wijst met een verwijzing naar een branch, deze verwijzing wordt de HEAD genoemd. Wanneer je een nieuwe branch hebt aangemaakt, wijst je HEAD nog steeds naar dezelfde branch als daarvoor. Je 'bevindt' je dus nog niet in de nieuwe branch. Om de HEAD naar de nieuwe branch te laten wijzen moet je deze nieuwe branch zelf 'uitchecken'. Zie ook de onderstaande afbeeldingen[1]. Op de linker afbeelding wijst de HEAD nog naar de master branch. Op de rechter afbeelding wijst de HEAD naar de nieuwe branch.

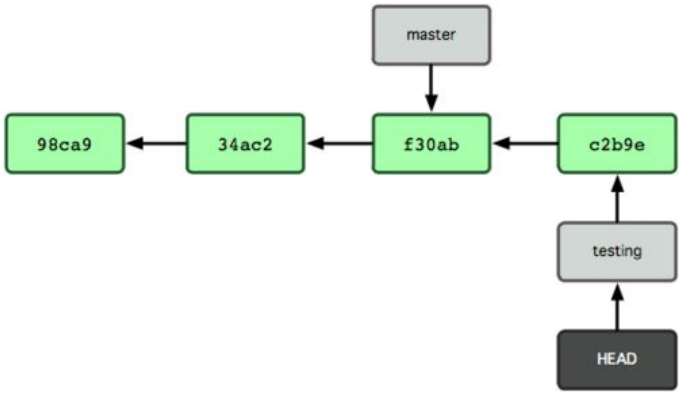
`$ git checkout <branch-name>`
Verplaatst de HEAD verwijzing naar een andere branch.





[1] bron: https://git-scm.com/book/nl/v1/Branchen-in-Git-Wat-is-een-branch

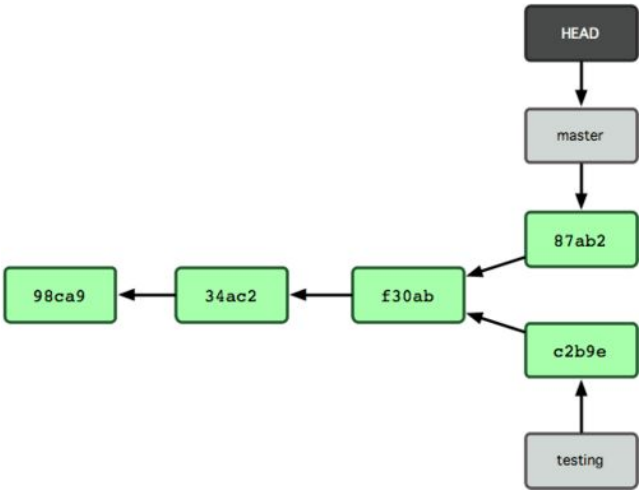
Voor het aanmaken van een branch en het direct uitchecken hiervan kun je de volgende command gebruiken.



`$ git checkout -b <branch-name>`
Maakt een nieuwe git branch aan genaamd <branch-name> en verplaatst direct de HEAD verwijzing naar deze nieuwe branch.

Wanneer je daarna wijzigingen in/toevoegingen aan de code commit, gaat de HEAD mee vooruit in deze branch, zie de afbeelding[1] rechtsboven.

Wanneer je daarna de master branch uitcheckt (`$ git checkout master`) is deze wijziging niet te zien, omdat deze is uitgevoerd en gecommit in de '*nieuwe branch*'.
Hetzelfde geldt voor een wijziging en commit in de master branch, deze is niet aanwezig in de '*nieuwe branch*'. Zie de afbeelding[1] hiernaast.



<u>Bestaande branches tonen</u>
Met de onderstaande commands kun je de branches tonen. Bij het tonen van lokale branches staat er een * voor de branch waar op dat moment de HEAD heen wijst.

`$ git branch`
Toont de lokale branches.

`$ git branch -r`
Toont de remote branches.

`$ git branch -a`
Toont zowel de lokale als remote branches.

<u>Branches mergen</u>
Wanneer je klaar bent met het werken in een losse branch ben je klaar om je branch te mergen. Dit kun je doen d.m.v. een *merge request*, zie de uitleg op pagina 1 en 2.

[1] bron: https://git-scm.com/book/nl/v1/Branchen-in-Git-Wat-is-een-branch

Naast het mergen via een *merge request*, kun je ook de *merge* command gebruiken in de command line. Check dan eerst de branch uit waar je naartoe wil mergen, bijvoorbeeld de *develop* branch.

`$ git merge <branch-name>`
Merged de lokale branch genaamd <branch-name> in de huidige branch.

Bestaande branches verwijderen
Wanneer je klaar bent met een branch kun je deze verwijderen. Let op: om een lokale branch te verwijderen, moet je je buiten die branch bevinden. De HEAD moet dan naar een andere branch wijzen.

`$ git branch -d <branch-name>`
Verwijdert de lokale branch genaamd <branch-name>.

`$ git push origin --delete <branch-name>`
Verwijdert de remote branch genaamd <branch-name>.

Indien je meer wilt weten over git branches, kun je kijken op de onderstaande site, deze bevat een duidelijke uitleg over git branches:
https://git-scm.com/book/nl/v1/Branchen-in-Git-Wat-is-een-branch.

[1] bron: https://git-scm.com/book/nl/v1/Branchen-in-Git-Wat-is-een-branch