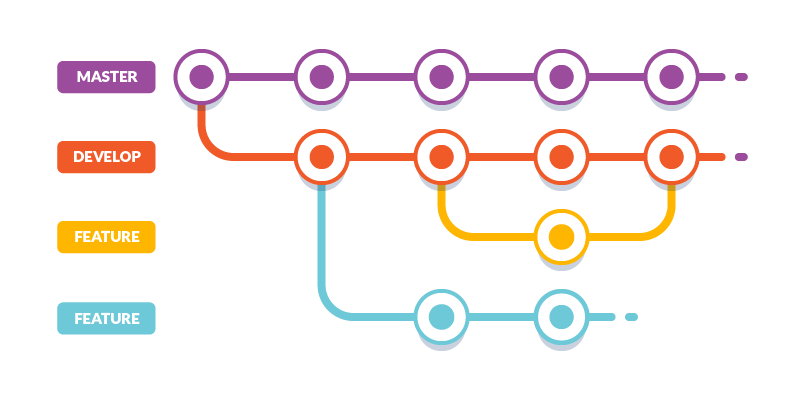
23-9-2022

Muijsenberg, Reno R.F.

Fontys

Git Flow

Research



Inhoud

[Aanleiding 2](#_Toc118724044)

[Hoofdvraag 3](#_Toc118724045)

[Deelvragen 3](#_Toc118724046)

[Wat is Git flow 4](#_Toc118724047)

[Branches 4](#_Toc118724048)

[Main 4](#_Toc118724049)

[Develop 4](#_Toc118724050)

[Feature 4](#_Toc118724051)

[Release 4](#_Toc118724052)

[Hotfix 4](#_Toc118724053)

[Wat zijn andere Git strategieën 5](#_Toc118724054)

[GitHub flow 5](#_Toc118724055)

[Branches 5](#_Toc118724056)

[Gitlab Flow 6](#_Toc118724057)

[Branches 6](#_Toc118724058)

[Wat zijn de nadelen van deze strategieën 7](#_Toc118724059)

[Nadelen Git flow 7](#_Toc118724060)

[Nadelen Github flow 7](#_Toc118724061)

[Nadelen Gitlab flow 7](#_Toc118724062)

[Hoe werkt Git flow in een groepsproject 8](#_Toc118724063)

[Git Flow toepassen 9](#_Toc118724064)

[Conclusie 10](#_Toc118724065)

[Bronnen 11](#_Toc118724066)

# Aanleiding

Dit document bevat een onderzoek wat moest doen voor school. Ik heb gekozen om mijn onderzoek te laten gaan over Git flow omdat wij op school en in de werk wereld bijna overal git wordt gebruikt. Het is dan ook handig om bepaalde best practices te kennen zodat je altijd goed een keuze kunt maken over wat het beste bij jouw project past.

# Hoofdvraag

* Hoe pas ik Git flow toe in mijn eigen project.

# Deelvragen

* Wat is Git flow
* Wat zijn andere git strategieën
* Wat zijn de nadelen van deze strategieën
* Hoe werkt Git flow in een groepsproject

# Wat is Git flow

Git Flow is de oudste strategie voor het beheren van je code met git. Het zorgt ervoor dat je je code goed kan onderhouden op bepaalde niveaus. Het wordt veel gebruikt door open source projecten zodat ze goed code die gepusht wordt kunnen monitoren.

## Branches

In deze strategie heb je twee hoofd branches: de main branch en de develop branch. Verder heb je nog een aantal kleinere branches die hier ook mee gebruikt worden zoals: feature, release en hotfix.

### Main

De main branch is bedoeld als stabiel release dit is dan ook de branch waar geen fouten op mogen staan. Vaak wordt vanaf deze branch direct deployed naar de live server.

### Develop

Je werkt op de develop branch wanneer dit werk op een stabiel punt komt wordt het gemerged in main branch met een release nummer.

### Feature

De feature branch wordt gebruikt voor wordt gemaakt vanaf de develop branch voor een kleine feature en wordt wanneer af weer terug gemerged in develop branch.

### Release

De release branch is bedoeld voor wanneer je de code op het punt komt dat het gereleased kan worden deze branch wordt dan gebruikt voor het klaarmaken/ laatste bug fixen voordat het in de main branch wordt gemerged.

### Hotfix

De hotfix branch wordt gebruikt voor het oplossen van urgente problemen. Deze branch wordt gemaakt vanaf de main branch en hier ook rechtstreek in terug gemerged.

# Wat zijn andere Git strategieën

## GitHub flow

Github flow is een makkelijkere versie van Git flow en is bedacht met deployment als hoofdgedachten. Deze strategie wordt dan ook door Github zelf gebruikt om hun site te developen. Deze strategie is makkelijk om te snappen en wordt vaak gebruikt door kleinere development teams.

### Branches

Bij Github flow heb je maar twee branches de main en feature branches.

#### Main

De main branch is bedoeld als stabiel release dus deze branch zou altijd in staat moeten zijn om rechtstreeks deployed te worden naar een live omgeving.

#### Feature

Wanneer deze branch gemaakt wordt krijgt hij een descriptieve naam waaraan te zien is wat er op deze branch gemaakt wordt. De branch wordt vanaf main gemaakt en word vervolgen weer terug gemerged in main door middel van een pull request.

## Gitlab Flow

Gitlab flow is de laatst bedachte branch strategie. Gitlab flow is bedacht voor het te verbeteren van de nadelen van de andere twee strategieën maar dit maakt het niet per definitie beter. Gitlab flow probeert de complexiteit van git flow te verminderen en de functionaliteit van github flow te verbeteren.

### Branches

Met deze strategie heb je drie branches: een production branch, de main branch en feature branches.

#### Production

De production branch wordt gebruikt als live branch deze branch mogen dan ook geen bugs bevatten omdat dit meteen naar een live omgeving gezet wordt.

#### Main

De main branch of ook wel de develop branch in deze strategie wordt gebruikt als de main branch op deze branch wordt alles klaar gemaakt voordat het in production (live) gezet kan worden.

#### Feature

Op deze branch worden losse feature gemaakt deze branch wordt gemaakt vanaf main en weer terug in main gemerged.

# Wat zijn de nadelen van deze strategieën

## Nadelen Git flow

Een van de nadelen van deze werkwijze is dat wanneer een feature te lang op zijn/haar eigen feature branch bevind je merge conflicten kan krijgen omdat het niet meer synchroon loopt met de reset van het project. Verder kan deze methode voor projecten die continuouse intergration hebben een slomere oplossing zijn.

## Nadelen Github flow

Wanneer er veel features tegelijk worden gemaakt kan het aantal pull request snel opstappelen, dit kan er vaak voor zorgen dat er dan ook merge conflicten komen op de main branch. Wanneer er dan zoveel pull request zijn en elke feature moet gedeployed worden kan het voorkomen dat je heel lang bent het wachten tot jouw feature uiteindelijk op live staat. Dit kan natuurlijk invloed hebben op deadlines.

## Nadelen Gitlab flow

Deze strategie is een zeer flexibele strategie waardoor het voor elk bedrijf net anders kan zijn daarom is het verstandig dat er altijd gedocumenteerd wordt hoe jouw team precies te werk gaat met de strategie. Er zijn ook een aantal optionele branches die in deze strategie gebruikt worden maar deze halen de complexiteit omhoog en hangt er dus vanaf of dit goed werkt met jouw team.

# Git Flow toepassen

Om Git flow direct toe te passen in je repository heb je of de op Windows apparaten [Git for Windows](https://git-scm.com/download/win) nodig. Nadat je dit geïnstalleerd hebt kunt je in je repository “git flow init” runnen via de CMD. Deze command maakt voor jou de benodigde branches aan om git flow te gebruiken. Wanneer je “git flow init” gebruikt vraagt hij je naar de naam van de branches die jij wilt gaan gebruiken.

Text

Description automatically generated

#### Feature branch

Wanneer je dan aan een feature wilt gaan beginnen gebruik je de git flow command “git flow feature start {feature\_branch}” en wanneer je dan klaar bent met een feature merge je de feature branch weer in de develop branch met de command “git flow feature finish {feature\_branch}”.

#### Release branch

Wanneer de develop branch genoeg features heeft gekregen en klaar is om te releasen of de release datum is eraan het komen wordt er een release branch gemaakt. Als de branch gemaakt is wordt er een release cycle gemaakt dus kunnen er na dit punt geen nieuwe features toegevoegd worden allen nog maar bug fixes en documentatie gerelateerde veranderingen. Wanneer de release branch dan klaar is voor live te komen wordt deze branch met een versie nummer in main gemerged en hierna wordt release weer terug gemerged in de develop branch. De release branch wordt gemaakt met het command “git flow release start {versie nummer}” en als de release branch klaar is wordt deze gemerged met het command “git flow release finish {versie nummer}”.

#### Hotfix branch

De hotfix branch wordt gebruikt om snel nog patches naar de release branch te pushen. Dit zorgt ervoor dat je team aan bugs kan werken zonder dat er een blokkade is voor andere team leden. Je maakt een hotfix branch aan door middel van “git flow hotfix start {hotfix\_branch}”. Wanneer de hotfix klaar is wordt deze gemerged in zowel main als develop met de command “git flow hotfix finish {hotfix\_branch}”

# Hoe werkt Git flow in een groepsproject

In de proftaak hebben we ook besloten om Git flow te gebruiken. We hebben bij iedereen die meewerkt aan het project de stappen uitgevoerd om Git flow te initialiseren. Nadat we dit gedaan hadden kon iedereen via de console de git flow command gebruiken om bijvoorbeeld een nieuwe feature aan te maken. Maar na een tijdje liepen we tegen een struikelblok aan dit was namelijk dat wanneer je `git flow feature finish {feature\_name} ` runt hij automatisch terug merged in de dev branch. Maar aangezien de dev branch branch protection had ging dit fout en merged hij dit alleen lokaal. We hadden dus een keuze om te maken.

* Keuze 1: Schakel de branch protection uit, dit is natuurlijk niet optimaal want dit vergroot de kans op fouten.
* Keuze 2: Maak iedereen owner van de repo maar ook hier vergroot dit de kans op fouten
* Keuze 3: Zoek een andere manier of dit kan zonder de kans op fouten te vergroten.

Wij hebben natuurlijk gekozen voor optie 3. We zijn op het internet gaan zoeken naar een oplossing waarmee we de kans op fouten niet vergroten en dat we de werking van git flow in takt houden. We kwamen er op uit dat we in plaats van `git flow feature finish` de `git flow feature publish` command gebruiken. Dit zorgt ervoor dat de branch met de veranderingen naar de origin wordt gepusht. Dan kun je vanaf daar een nieuw pull request maken om met dev te merge. Het enige wat er dan nog moest gebeuren is de branch op je lokale machine te verwijderen.

# Conclusie

# Bronnen

<https://rewind.com/blog/git-branching-strategies-explained/>

<https://docs.gitlab.com/ee/topics/gitlab_flow.html>

<https://nira.com/gitlab-workflow/>

<https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow>