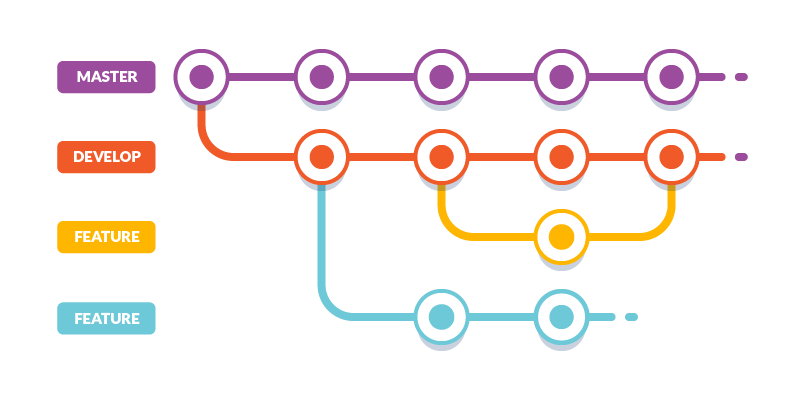
23-9-2022

Muijsenberg, Reno R.F.

Fontys

Git Flow

Research



Inhoud

[Aanleiding 2](#_Toc116297819)

[Verschillende strategieën 3](#_Toc116297820)

[Git Flow 3](#_Toc116297821)

[Branches 3](#_Toc116297822)

[Nadelen 3](#_Toc116297823)

[GitHub flow 4](#_Toc116297824)

[Branches 4](#_Toc116297825)

[Nadelen 4](#_Toc116297826)

[Gitlab Flow 5](#_Toc116297827)

[Branches 5](#_Toc116297828)

[Nadelen 5](#_Toc116297829)

[Git Flow toepassen: 6](#_Toc116297830)

[Bronnen 7](#_Toc116297831)

# 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.

# Verschillende strategieën

## 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.

### Nadelen

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.

## 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.

### Nadelen

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.

## 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.

### Nadelen

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}”.

# 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>