SCM STRATEGIE

B2

AUTEUR: GEORDI TRAPMAN
EIGENAAR: GEORDI TRAPMAN

16/02/2018
V1

INHOUDSOPGAVE

INLEIDING

In dit document staan de afspraken en procedures beschreven m.b.t. het softwareconfiguratiebeheer tijdens het project voor de ontwikkeling van de website voor herstelcirkels. De branch-strategie is grotendeels geïnspireerd door een blogpost van Vincent Driessen. Vragen kunnen gesteld worden aan Geordi Trapman (softwareconfiguratiebeheerder SCM).

BRANCHING

MASTER

Deze branch bevat enkel code waarop genoeg vertrouwd kan worden om foutloos te functioneren in een productie omgeving. Deze **branch** is uniek en loopt oneindig door.

- Alle **commits** hebben een tag
- Mag alleen gemerged worden vanaf **release-*** en **hotfix-*** branch
- Bij merges wordt er een pull-request aangemaakt. Deze wordt door een andere developer en de SCM nagekeken en goedgekeurd.

DEVELOPMENT

Deze **branch** wordt gebruikt om losse features en bugfixes te integreren in één codebasis. Hij is uniek en loopt oneindig door.

- Bevat bugfixes als commits
- Wordt niet op gewerkt voor functionaliteit
- Mag alleen gemerged worden vanaf **release-***, **hotfix-*** en **feature-*** branches

FEATURE-*

Deze **branch** wordt gebruikt om losse functionaliteiten te ontwikkelen. Deze heeft de naam van de functionaliteit of issue zoals deze geregistreerd staat in Taiga (bijvoorbeeld feature-datepicker).

- Wordt 1 keer gepulled vanaf **development**
- Wordt 1 keer, no ff, gemerged met **development** of gesloten
- Kan eventueel lokaal worden opgeslagen als er maar een ontwikkelaar aan werkt(dit heeft geen voorkeur)

RELEASE-*

Deze **branch** wordt gebruikt om te werken naar een release. Deze heeft als naam, het versienummer van de release (bijvoorbeeld release-1.0). Deze **branch** wordt in overleg met het projectteam aangemaakt en bevat alle functionaliteiten zoals afgesproken in de releaseplanning.

- Wordt 1 keer gepulled vanaf **development**
- Bevat bugfixes als commits
- Bugfixes worden ook naar **development** gezet

HOTFIX-*

Deze **branch** wordt gebruikt om problemen op te lossen die zich voortdoen bij gereleasede code (**commits** op **master**). Deze wordt dus enkel aangemaakt als er problemen door alle testen heen zijn gekomen en problemen veroorzaken binnen productie. Meerdere bugfixes kunnen in een hotfix **branche** worden meegenomen. De naam van deze branch is incrementeel op een decimaal punt (bijvoorbeeld hotfix-1.1).

- Wordt 1 keer gepulled van master
- Bevat bugfixes als commits
- Zodra de problemen zijn opgelost wordt de branch gemerged met **master** en **development** en gesloten

AFSPRAKEN

ISSUES

Issues worden gebruikt om bugs of gewenste verbeteringen/toevoegingen te registreren. De titel en omschrijving bevatten nuttige en gedetailleerde informatie, in het Nederlands. Er worden labels toegevoegd die slaan op het **issue** om zoeken en toewijzing gemakkelijker te maken. Aan **issues** wordt gewerkt binnen GitHub.

COMMITS

Bij een **commit** wordt een relevante titel gevoegd die slaat op de wijzigingen die erin zijn doorgevoerd. Alle tekst bij het **commit** bericht zal in het Nederlands zijn.

TAGS/RELEASES

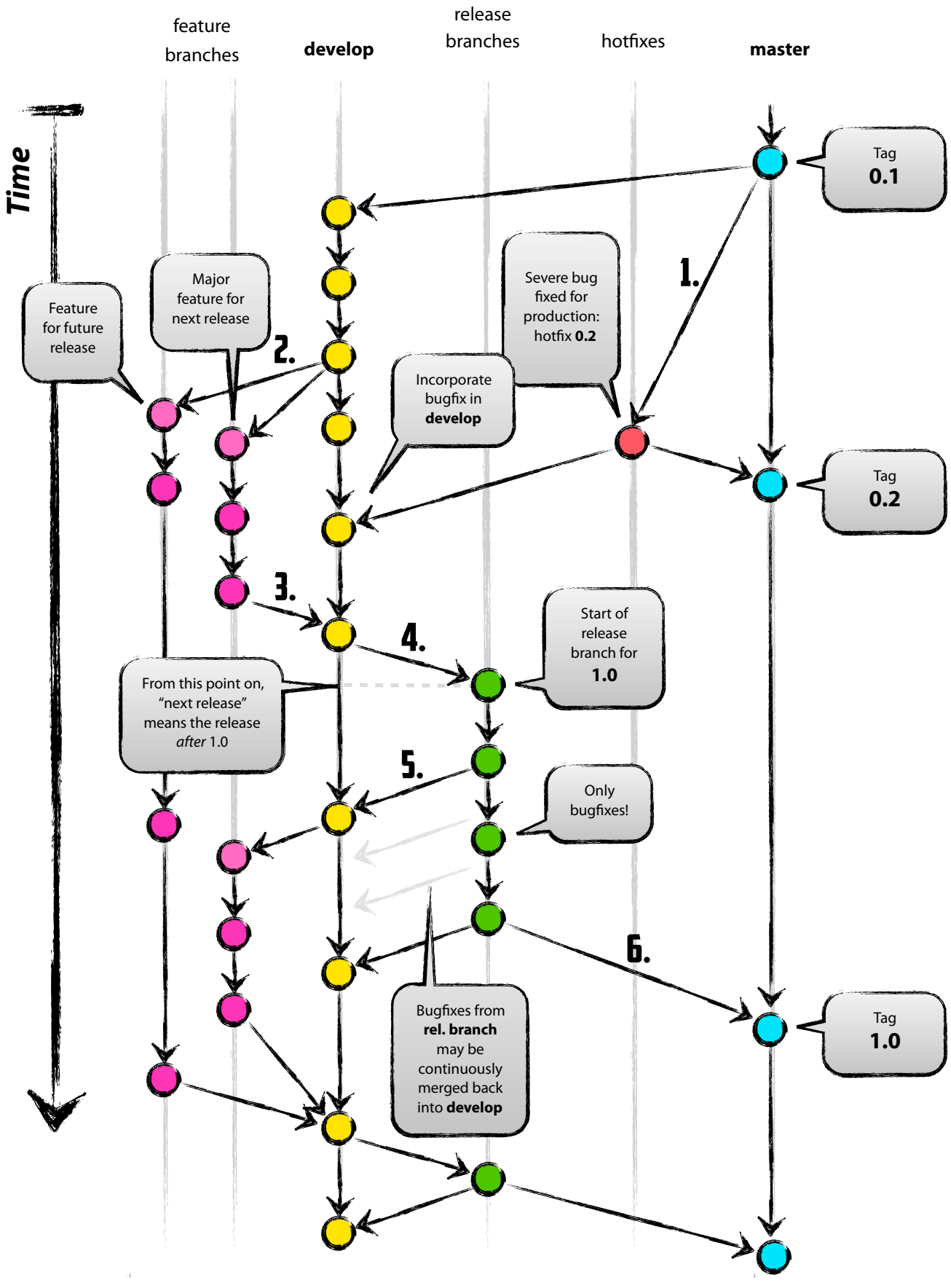
Alle **commits** naar de master **branch** zullen een versienummer krijgen door middel van een **tag**. Bij een versie waarbij de functionaliteiten worden uitgebreid wordt er compleet nieuwe versie neergezet (bijvoorbeeld: v1.0 > v2.0). Bij herstellen van bugs wordt het cijfer achter de komma opgehoogd (bijvoorbeeld: v1.1 > v1.2).

PULL-REQUEST

Zodra een **feature** voltooid is, kan deze er een **pull-request** worden gemaakt. Dit wordt gedaan via de GitHub website. Een **pull-request** moet worden goedgekeurs door 1 andere developen voordat deze toegelaten wordt. Het is aan te raden de **development-branch** te mergen in de **feature-branch** voor het starten van een **pull-request**. De criteria voor het goedkeuren van een pull-request zijn als volgt:

- Er zijn geen merge conflicts in de **pull-request**
- De code is gecontrolleerd op het volgen van de coding guidelines
- De pull-request omvat geen code die niet gerelateerd is aan de **feature**

GIT COMMANDS



1. CREER HOTFIX BRANCH

```
$ git checkout -b hotfix-1.2.1 master
Switched to a new branch “hotfix-1.2.1”
```

2. CREER FEATURE-* BRANCH

```
$ git checkout -b feature-datepicker develop
Switched to a new branch “feature-datepicker”
```

3. MERGE DEVELOPMENT > FEATURE

```
$ git checkout feature-datepicker
Switched to branch ‘feature-datepicker’
$ git merge develop
Updating ea1b82a..05e9557
(Summary of changes)
$ git push
```

4. CREER RELEASE BRANCH

```
$ git checkout -b release-1.2 develop
Switched to a new branch “release-1.2”
```

5. MERGE RELEASE > DEVELOP

```
$ git checkout develop
Switched to branch ‘develop’
$ git merge --no-ff release-1.2
Merge made by recursive.
(Summary of changes)
```

6. RELEASE BRANCH GEREEDMAKEN

```
$ git checkout master
Switched to branch ‘master’
$ git merge --no-ff release-1.2
Merge made by recursive.
(Summary of changes)
$ git tag -a 1.2
$ git checkout develop
Switched to branch ‘develop’
$ git merge --no-ff release-1.2
Merge made by recursive.
(Summary of changes)
$ git branch -d release-1.2
Deleted branch release-1.2 (was ff452fe).
```