

# Release Plan Groep 5

Teun Willems, Jules Kreutzer

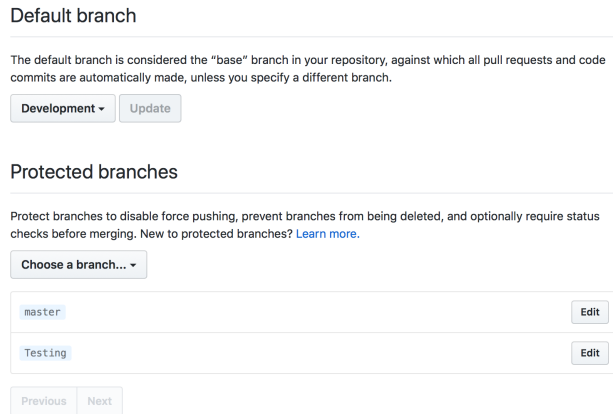
17 februari 2018

# Inhoudsopgave

- 1 Version Control
  - 1.1 Branches . . . . .
  - 1.1.1 Master . . . . .
  - 1.1.2 Testing . . . . .
  - 1.1.3 Development . . . . .
  - 1.1.4 Features . . . . .
- 2 Code Style
- 3 Continuous Integration
- 4 Testing
- 5 Versioning
- 6 Continuous Delivery

# 1 Version Control

Door middel van version control kunnen meerdere ontwikkelaars aan dezelfde software werken. Tijdens JEA6 zal Git (GitHub) gebruikt worden voor version control. Er is hiervoor gekozen omdat hiermee reeds de meeste ervaring is opgedaan.



The screenshot shows the 'Default branch' and 'Protected branches' settings in a GitHub repository. Under 'Default branch', 'Development' is selected as the default branch, with an 'Update' button. Below, the 'Protected branches' section is active, showing a list of protected branches: 'master' and 'Testing'. Each branch has an 'Edit' button. At the bottom, there are 'Previous' and 'Next' navigation buttons.

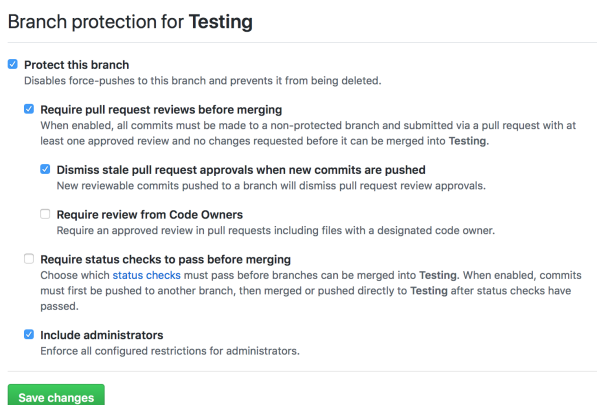
Figuur 1.1: Overzicht van default branch en protected branches

## 1.1 Branches

Voor de repository waar de code van het vak JEA6 wordt bijgehouden, zijn verschillende branches aangemaakt. Wat het nu is van deze verschillende branches, staat in de volgende alinea's uitgelegd.

### 1.1.1 Master

De code die beschikbaar is in de master-branch, is de code van de software versie die in productie gebruikt wordt. Nieuwe functies of bug fixes mogen dan ook niet rechtstreeks naar deze branch gecommitt worden. Deze branch is tevens protected. Dit wilt zeggen dat vanuit een git programma (UI of CLI) niet rechtstreeks naar deze branch gecommitt kan worden. Dit zorgt ervoor dat een ontwikkelaar niet per ongeluk naar deze branch kan committen.



The screenshot shows the 'Branch protection for Testing' settings. It lists several protection rules: 'Protect this branch' (checked), 'Require pull request reviews before merging' (checked), 'Dismiss stale pull request approvals when new commits are pushed' (checked), 'Require review from Code Owners' (unchecked), 'Require status checks to pass before merging' (unchecked), and 'Include administrators' (checked). A green 'Save changes' button is at the bottom.

Figuur 1.2: Overzicht van protectie wat ingesteld is op branch

### 1.1.2 Testing

De Testing-branch zal gebruikt worden om nieuwe functionaliteit die ontwikkeld zijn in de development branch te testen. Wanneer de functionaliteit in de testing branch geaccepteerd wordt, zal deze worden doorgezeten naar de master branch zodat de verbeteringen en/of nieuwe wijzigingen beschikbaar zijn voor alle gebruikers op de productie omgeving De Testing-branch heeft dezelfde setup als

de Master-branch. Dit wilt zeggen dat het op deze branch ook niet mogelijk is om rechtstreeks code te committen.

### 1.1.3 Development

De Development-branch bevat nieuwe functionaliteit of verbeteringen wat ontwikkeld zijn. Deze branch is vrij toegankelijk voor de ontwikkelaar om eventueel kleine(re) verbeteringen door te voeren. Het is als nog de bedoeling dat nieuwe functionaliteit niet rechtstreeks op deze branch wordt gecommitt. Mocht een ontwikkelaar nieuwe functionaliteit of grotere verbeteringen willen ontwikkelen, kan hiervoor een zogenaamde Feature-branch worden aangemaakt. Meer informatie hierover is in de volgende alinea te lezen

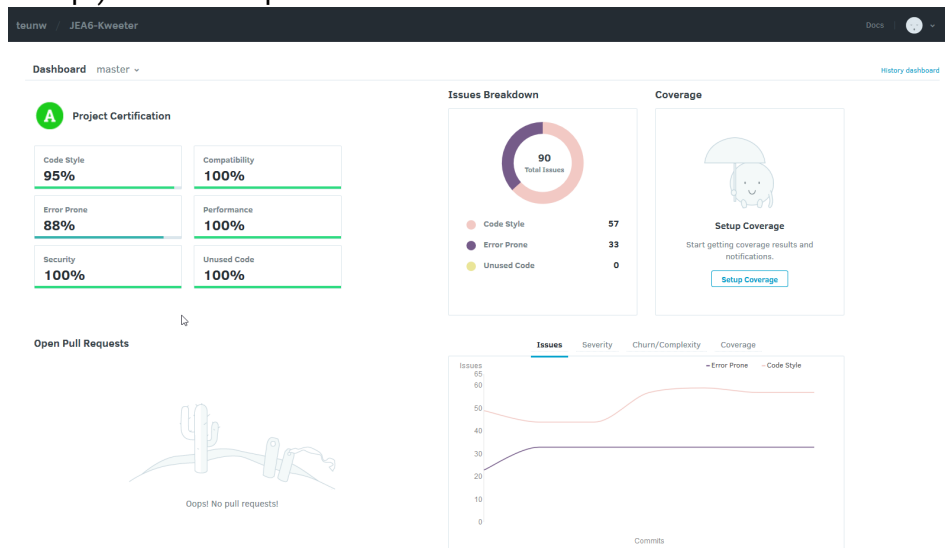
### 1.1.4 Features

De Features-branch is geen branch die direct aangemaakt is op Github. Dit komt doordat "Features" een overkoepelende term is. Wanneer een ontwikkelaar nieuwe functionaliteit of verbeteringen wilt ontwikkelen, dan zal hiervoor een nieuwe branch moeten worden aangemaakt, bijvoorbeeld "Dev-exampleFeature". Aan de hand van de prefix "Dev-" weet iedereen dat dit een branch betreft waar nieuwe functionaliteit of verbeteringen op worden ontwikkeld. Het gedeelte achter de prefix moet dan ook beschrijven aan welke functionaliteit of verbeteringen worden gewerkt.

Wanneer de ontwikkelaar het implementeren heeft voltooid, zal dit eerst getest moeten worden op zijn eigen, lokale omgeving. Wanneer die nieuwe functionaliteit werkt met de rest van de ontwikkelde software, kan de ontwikkelaar zijn branch, bijvoorbeeld "Dev-exampleFeature" gaan samenvoegen met de Development-branch. Hiervoor maakt de ontwikkelaar een merge request aan wat ervoor zorgt dat diene nieuwe code wordt samengevoegd op de Development-branch.

## 2 Code Style






Voor de code style worden de standaard code style settings van IntelliJ IDEA gebruikt. De code style wordt ook gecontroleerd door Codacy, een code analyse tool. Codacy controleerd je Java (en Javascript) codebase op verschillende errors.



Ook kan er gecontroleerd worden op beveiligingsproblemen in je applicatie, stel dat je bijvoorbeeld SQL-query parameters in je queries hebt staan.

### 3 Continuous Integration

Continuous is geregeld door TravisCI, voor open source projecten is TravisCI gratis te gebruiken. Omdat het project met Gradle is gebouwd is het makkelijk om CI ondersteuning toe te voegen aan de backend. Het commando "gradle test" draait alle tests in het project, als er iets fout gaat is dit ook meteen in Github zichtbaar.

Dev ci (#1) ...	Verified	 0d53d68	
 teunw committed 4 days ago ✓			
Fixed CI		 967ecdd	
 teunw committed 4 days ago ✗			

## 4 Testing

## 5 Versioning



## 6 Continuous Delivery