

Release Plan Groep 5

Teun Willems, Jules Kreutzer

13 februari 2018

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.

2

The screenshot shows the GitHub repository settings for a repository. The 'Default branch' section is at the top, with a dropdown menu set to 'Development' and an 'Update' button. Below this is the 'Protected branches' section, which includes a description of protected branches and a 'Choose a branch...' dropdown. Two branches are listed: 'master' and 'Testing', each with an 'Edit' button. At the bottom of the 'Protected branches' section are 'Previous' and 'Next' navigation buttons.

Default branch

The default branch is considered the "base" branch in your repository, against which all pull requests and code commits are automatically made, unless you specify a different branch.

Development • Update

Protected branches

Protect branches to disable force pushing, prevent branches from being deleted, and optionally require status checks before merging. New to protected branches? [Learn more](#).

Choose a branch... ▾

master	Edit
Testing	Edit

Previous Next

Figuur 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.

Branch protection for **Testing**

- ☒ **Protect this branch**
Disables force-pushes to this branch and prevents it from being deleted.
- ☒ **Require pull request reviews before merging**
When enabled, all commits must be made to a non-protected branch and submitted via a pull request with at least one approved review and no changes requested before it can be merged into **Testing**.
- ☒ **Dismiss stale pull request approvals when new commits are pushed**
New reviewable commits pushed to a branch will dismiss pull request review approvals.
- ☐ **Require review from Code Owners**
Require an approved review in pull requests including files with a designated code owner.
- ☐ **Require status checks to pass before merging**
Choose which status checks must pass before branches can be merged into **Testing**. When enabled, commits must first be pushed to another branch, then merged or pushed directly to **Testing** after status checks have passed.
- ☒ **Include administrators**
Enforce all configured restrictions for administrators.

Save changes

Figuur 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 doorgezet 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 gecommit. 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

3 Continuous Integration

4 Testing

5 Versioning

Semantic Versioning?

6 Continuous Delivery