# Teamoverstijgende afspraken technologie BB8 bv.

v0.1
06-04-2022

# Git - Versiebeheer

## Gebruikte dienst

De meeste groepen gebruiken GitHub en een groep gebruikt Bitbucket, deze werken praktisch hetzelfde. Enkele groepen zitten vast aan GitHub vanuit hun opdrachtgever.

Vanuit de scrum masters is gekozen om JIRA te gebruiken als planningsprogramma. JIRA werkt goed samen met Bitbucket, je kunt hier bijvoorbeeld branches aan taken koppelen en dit automatiseren met het planningsbord.

## Workflow

Groepen die veel codebase schrijven gaan werken met een model op basis van A successful Git branching model van Vincent Driessen. Dit model werkt met een aantal standaard branches.

- **master (main)**. Deze branch is altijd klaar voor productie, alles wat op deze branch staat is ook goedgekeurd door de opdrachtgever. Deze branch is ook beschermd, hier kan niet zomaar op gepusht worden.

- **acceptance**. Deze branch is bedoeld als review branch, wanneer je een demo hebt voor je opdrachtgever zul je deze branch laten zien. Wanneer de opdrachtgever akkoord is met deze branch merge je hem naar main en maak je een release aan.

- **develop**.  Op deze branch wordt tijdens de sprint gewerkt, vanaf deze branch worden alle feature branches aangemaakt, en alle feature branches worden d.m.v. pull requests weer in develop gemerged.

- **feature**. Als je aan taak wordt gekoppeld maak je een feature branch aan vanaf develop. Je werkt aan deze feature branch totdat de feature af is. Wanneer de feature af is maak je een pull request aan en deze wordt dan nagekeken door iemand die niet aan deze feature heeft gewerkt.

  Hier zou je eventueel ook testen en pipelines aan toevoegen. Zo weet je dat de code die gepusht is ook zeker werkt.

## A successful Git branching model & Data Science

De graad waarop bepaalde opdrachten bestaat uit Data Science binnen de BB8 opdrachten van dit jaar  is behoorlijk variabel. Data Science projecten gebruiken (in de eerste fases) vaak meer *REPL*-omgevingen zoals Jupyter Notebooks dan “traditionele” codebases. Als men de fases van het CRISP-DM model gebruikt als categorisatie van verschillende tijdsinvesteringen binnen het Data-Science process, leert de werkelijkheid dat vaak het

merendeel van de tijd geïnvesteerd wordt in de constante iteratie tussen *Data Understanding & Data Preparation*.
Het onderzoeken van de data, het visualisaties hiervan creëren, en eventuele inzichten hiervan toepassen om de data te bewerken voor bruikbaarheid kan gebeuren in teamverband, maar gebeurt ook vaak op persoonlijker niveau. Ook zullen veel pogingen om bepaalde bewerkingen op de data te doen zinloos, en daarmee mogelijk ook niet meegenomen worden in het eindproduct.
Vanwege het tijdelijke, persoonlijke of verkennende karakter dat Jupyter Notebooks kunnen hebben vanwege al deze feiten, kan het een onnodige tijdsinvestering zijn om deze altijd door een systeem van code-reviews heen te sluizen. Daarom is het mogelijk voor Data-Science teams om met expliciet vastgestelde procedures hier bewust vanaf te wijken, door deze Notebooks bijvoorbeeld direct te ontwikkelen in de main-branch.

Vaak is bij de vroege stadia van Data-Science projecten een eventuele geschreven codebase ook ondersteunend voor het onderzoeksproces. Hierom zal code eerder in combinatie met Jupyter Notebooks ontwikkeld worden, en is de delta tussen de acceptance en main branches van “A successful Git branching model” vooral een zinloze extra stap. Daarom kan (tot een eventuele production-stage bereikt wordt) expliciet gekozen worden om feature branches direct op de main-branch aan te sluiten.

Voor het succesvol gebruik van Git voor het ontwikkelen van Jupyter Notebooks is het ook belangrijk om bepaalde procedures aan het houden om frequente, lastig op te lossen merge-conflicten te vermijden. Jupyter-Notebooks worden opgeslagen als JSON-bestanden, met daarin geordende informatie over de code-cellen, de tekst-cellen, en eventuele outputs. De informatie over de inhoud van code-cellen en tekst-cellen is redelijk onveranderlijk, tenzij de code/tekst natuurlijk wordt aangepast. Git kan wat betreft merges hier ook redelijk goed mee omgaan. De opgeslagen informatie over outputs is wat dynamischer, en heeft de neiging vaak tussen sessies en/of gebruikers te veranderen. Dit veroorzaakt wanneer er op Git 2 tijdlijnen ontstaan in de aanpassingen van de Jupyter Notebook, of het nou door branching of door oude lokale kopieën is, vrij betrouwbaar zo-goed-als-onoplosbare merge conflicts. Een simpele maatregel om dit te voorkomen is **altijd legen van alle cell-outputs voor het maken van een commit**, behalve wanneer de Jupyter Notebook in kwestie een definitieve versie is, of er geen kans is dat er een 2-tijdlijnen situatie ontstaat.

Wanneer men effectief versiebeheer op Jupyter Notebooks wilt waarborgen, is het gebruik van de software *ReviewNB* aangeraden. Deze software is de integreren met zowel Github als Bitbucket, en staat de gebruiker makkelijk in staat de verschillen van Jupyter Notebooks in Notebook formaat, in plaats van. metadata-formaat, voor onder anderen commits en pull requests te bekijken.

# Planningsbord

Hoewel de beslissing van een planningsbord vooral een beslissing van de scrum masters van BB8-B.V. is, is een eventuele integratie met de online Git-omgevingen relevant om te benoemen. Veel aanbieders van online Git-omgevingen als Github of Atlassian (bitbucket) bieden veel interfuncitonaliteit tussen deze 2 diensten. De specifieke mogelijkheden wat betreft integraties tussen planningsbord en online Git-omgeving zijn eventuele mooie toevoegingen, maar niet leidend in de keuzes voor welke leverancier er voor deze diensten wordt gekozen, en voor nu zijn er geen voorschriften in het combineren van deze 2 vanuit dit document.

Goed om te noemen is dat er niet alleen integraties binnen de bedrijven van mogelijk zijn, maar (eventueel met 3e-partij software) ook soms tussen de softwares van verschillende bedrijven, bijvoorbeeld tussen Jira (planningsbord) en Github (Git).

# Cloud data storage

# Documentatie

Alle groepen werken met Python, we houden hierbij de Google Python Style Guide aan. Ook liggen er in het Turing lab nog enkele boeken over clean code, deze kunnen ook gebruikt worden.