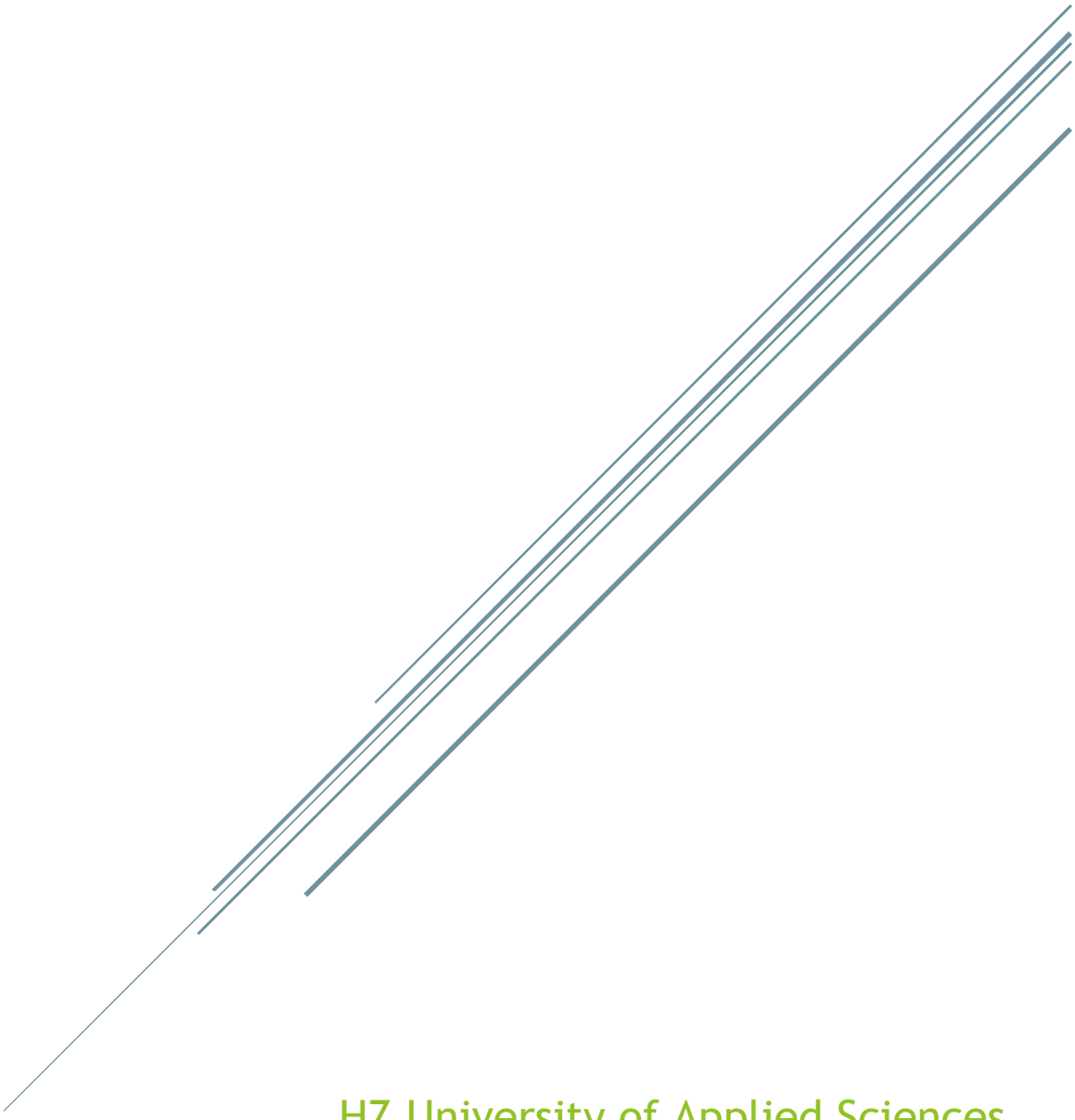
GIT VERSIEBEHEER BIJ DE HZ UNIVERSITY OF APPLIED SCIENCES

Bastiaan van Mol - Onderzoeksrapport



HZ University of Applied Sciences
VCC

GIT versiebeheer bij de HZ University of Applied Sciences

Bastiaan van Mol - Onderzoeksrapport

Plaats van uitgave:	Zaamslag
Datum:	20-04-2015
Studentnummer:	0061826
Studiejaar:	2014/2015
Semester:	2
Studieonderdeel:	VCC
Begeleidend docent:	Wouter Everse, Anton Bil
Beoordelende student:	Bastiaan van Mol
Versie:	1.0

Samenvatting

In het kader van de Vrije Compositie Cursus opdracht aan de HZ University of Applied Sciences is door de student een onderzoek uitgevoerd naar de mogelijkheden voor een GIT versiebeheer systeem binnen de onderwijsinstelling. Voor dit onderzoek is er gekeken naar mogelijke systemen die dit kunnen verwezenlijken en verschillende workflows die passen bij de gestelde randvoorwaarden. Uit het onderzoek is gebleken dat een GIT versiebeheer systeem binnen de onderwijsinstelling een mogelijkheid is als er gebruik wordt gemaakt van de student-teacher workflow in combinatie met GitLab als systeem. Hiermee is het mogelijk om studenten en docenten te scheiden in groepen en projecten te koppelen aan klassen. Tevens is beveiliging hierbij geregeld door middel van ingebouwde authenticatie. Ook het pushen en pullen naar de instantie is een mogelijkheid.

Inhoudsopgave

1 Inleiding

1.1 Aanleiding

Op de ICT opleiding van de HZ University of Applied Sciences wordt er gebruik gemaakt van een versiebeheer systeem voor de studenten alsmede de docenten. Met dit versiebeheer systeem kan eenvoudig, en met meerdere personen tegelijk, gewerkt worden aan een stuk software of programma. Het huidige versiebeheer programma dat gebruikt wordt, SVN, wordt al een dagje ouder en er zijn nieuwe spelers op de markt die nieuwe kansen bieden voor de opleiding op het gebied van versiebeheer.

Naar aanleiding van de bovenstaande situatie is er een onderzoek gestart naar het gebruik van een nieuw versiebeheer systeem, namelijk Git. Bij dit onderzoek ligt de nadruk op het vervangen van het huidige versiebeheer systeem met die van Git zonder dat de randvoorwaarden verloren gaan. Dit verslag is het directe resultaat van dit uitgevoerde onderzoek.

1.2 Onderzoeksvraag

Om duidelijkheid te scheppen omtrent de definitie van het onderzoek is er een centrale onderzoeksvraag geformuleerd, deze luidt:

Hoe kan de ICT opleiding van de HZ University of Applied Sciences het best gebruik maken van het versiebeheer systeem Git?

Deze onderzoeksvraag is onderverdeeld in deelvragen die de basis vormen voor de activiteiten binnen dit project, deze luiden:

- *Wat zijn de grote verschillen tussen Git en SVN?*
- *Wat zijn de belangrijke randvoorwaarden die de opleiding stelt aan het systeem?*
- *Welke Git opzet past het beste bij deze randvoorwaarden?*
- *Welke software is er beschikbaar om deze opzet te realiseren?*
- *Wat zijn de stappen en functionaliteiten van deze Git opzet en hoe werkt deze precies?*

De onderzoeksvraag alsmede de opgestelde deelvragen zullen beantwoord worden aan de hand van verscheidene literatuuronderzoeken, desk-researches en experimenten. Hoe deze precies zijn opgeteld is terug te vinden in Hoofdstuk 3 – Methode.

Dit onderzoeksrapport heeft als doel om de lezer te informeren over het doorlopen onderzoek traject alsmede een inzicht te geven in de toepassing van Git en hoe deze het best binnen de ICT opleiding van de HZ University of Applied Sciences geïntegreerd kan worden. Dit rapport beschrijft het doorlopen traject en hoe de resultaten gewonnen zijn.

1.3 Opzet

Dit rapport is op een chronologische volgorde opgebouwd om een zo goed mogelijk beeld te vormen van het doorlopen traject. Het begint met een inleiding waar de aanleiding van het onderzoek staat beschreven alsmede de centrale vraag en deelvragen. Hierop volgend wordt het theoretisch kader uitgewerkt met daarin een uitleg over de belangrijke begrippen die in de scope vallen van het onderzoek. In het hoofdstuk daarna wordt de methode beschreven die gebruikt is om tot de beantwoording van de onderzoeksvraag te komen, gevolgd door de resultaten van het onderzoek. Deze resultaten zullen per deelvraag worden behandeld. Als laatste volgt er een conclusie met een aanbeveling voor de opdrachtgever. In deze conclusie wordt tevens de onderzoeksvraag beantwoord en teruggeblikt op het gehele onderzoek.

1.4 Functie

Met behulp van dit onderzoeksrapport krijgt de opdrachtgever een inzicht in hoe Git precies werkt en hoe dit invulling kan geven aan de, door de opleiding gestelde, randvoorwaarden van het versiebeheer systeem. Deze resultaten kunnen gebruikt worden om in de toekomst over te stappen naar een Git versiebeheer systeem.

2 Theoretisch kader

Dit hoofdstuk bevat een beschrijving van het theoretisch kader. Hier worden begrippen uitgelegd die van belang zijn binnen de scope van de onderzoeksvraag/deelvragen alsmede het gehele onderzoek.

2.1 Zoekplan

Voorafgaand aan het literatuuronderzoek is er een zoekplan opgesteld welke de zoekstrategie van de onderzoeker beslaat. In dit zoekplan staan de eisen waar de bronnen op zijn minst aan moeten voldoen. Daarnaast wordt ook de zoekmethode beschreven alsmede de kernwoorden waar op gezocht is.

De eisen die gesteld zijn aan de bronnen zijn:

- De bron mag niet ouder zijn dan 5 jaar

De typen bronnen die gebruikt zijn:

- Internet artikelen
- Boeken
- Online documenten/Wetenschappelijke artikelen
- Internetfora

De kernwoorden komen voort uit de aanleiding van het project. Een aantal van deze woorden zijn de volgende:

- Git
- Versiebeheer
- Git Workflows

Mochten er geen goede resultaten voort komen uit de bovenstaande kernwoorden, dan wordt er gebruik gemaakt van de Engelse variant van het woord, of synoniemen.

2.2 Git

GIT is een software broncode versiebeheer systeem dat gebruikt kan worden door ontwikkelaars om snel en efficiënt, tegelijkertijd, aan een stuk software/code te werken. In deze zin lijkt Git heel erg op andere versiebeheer programma's als SVN en CVS, echter het grote verschil is dat Git gedistribueerd werkt. Dit betekend dat je niet een checkout doet op de laatste versie van de code, maar de gehele repository kopieert (cloned) naar een lokale werk omgeving. Het grote voordeel hiervan is dat iedereen een eigen backup heeft van de globale server waar het project op staat. Hierdoor kan er aan de code gewerkt worden zonder dat er een internet verbinding nodig is. De aanpassingen worden namelijk lokaal opgeslagen en vervolgens gepusht naar de remote repository. (Git-scm.com, z.d.)

2.2.1 Belangrijke termen

Hieronder worden een aantal belangrijke termen beschreven die vaak voor komen wanneer er over Git gepraat wordt, zo ook in dit document. (GitWiki, z.d.)

Master
De hoofd repository waar de ontwikkelaars aan werken en waar de integratie van alle verschillende versies plaats vindt.

Clone
Een clone is een kopie van een bestaande Git repository, normaal gesproken van een remote location naar een eigen lokale omgeving.

Commit
Het opsturen van bestanden naar een repository, in dit geval de lokale repository. Bij andere versiebeheer systemen wordt dit ook wel de check-in genoemd.

Fetch of pull
Fetch is het ophalen van de laatste versie van de master repository, ook wel de update functie genoemd bij overige versiebeheer systemen. Het verschil tussen fetch en pull is hier dat fetch alleen de laatste versie ophaalt, pull haalt deze op en merged deze samen met lokale repository.

Push
Deze term wordt gebruikt om de code op te sturen naar de remote repository.

Remote
Dit zijn locaties van een eigen repository op een centrale server, dus niet op een lokale omgeving.

Head
De head is een referentie naar de node waar de ontwikkelomgeving van de repository op dat moment naar verwijst, dit hoeft dus niet altijd de laatste versie te zijn.

Branch
De branch kan vergeleken worden met een branch in de andere versiebeheer systemen, echter in Git is het niet meer dan een label gegeven aan een node. Het is geen fysieke kopie van de bestanden zoals in de andere systemen.

2.3 SVN

SVN is, net zoals GIT, een versiebeheer systeem voor ontwikkelaars. Het grote verschil echter met GIT is dat SVN niet distributed is. Bij SVN doe je dus een checkout op de laatst geüpdatet code op de repository. Het voordeel hiervan is wel dat iedereen snel de laatste versie van de code beschikbaar heeft, echter het committen met meerdere ontwikkelaars tegelijk kan zorgen voor verschillende conflicten. (Apache.com, z.d.)

3 Methode

Dit hoofdstuk beschrijft de verschillende gebruikte methoden gedurende het onderzoek. Per deelvraag volgt er een uitleg over de gebruikte methode en waarom deze relevant is voor het behalen van het juiste resultaat.

3.1 Wat zijn de grote verschillen van Git tussen SVN?

De eerste deelvraag van dit onderzoek beslaat het onderzoeken van de grote verschillen tussen Git en, het huidig gebruikte versiebeheer systeem, SVN. Om deze te bepalen wordt er gebruik gemaakt van een desk-research-methode. Dit wil zeggen dat er op internet gezocht gaat worden naar relevante en betrouwbare bronnen.

Om de reikwijdte van dit desk-research-onderzoek te beperken worden de functionaliteiten van beide versiebeheer systemen opgezocht en naast elkaar gelegd. Hieruit worden vervolgens de verschillen gehaald en bepaald of deze relevant zijn voor het onderzoek.

3.2 Wat zijn de belangrijke randvoorwaarden die de opleiding stelt aan het systeem?

De tweede deelvraag omvangt een onderzoek naar de belangrijke randvoorwaarden die door de opleiding gesteld zijn aan het systeem. Om dit te achterhalen wordt er een afspraak gemaakt met de opdrachtgever. Hierbij wordt, naast het bespreken van de algehele opdracht, ook aandacht besteed aan de randvoorwaarden en functionaliteiten van het versiebeheer systeem. De resultaten van dit gesprek leggen de basis voor het bepalen van de juiste opzet van Git specifiek voor de opleiding.

3.3 Welke Git opzet past het beste bij deze randvoorwaarden?

Voor de derde deelvraag wordt er een onderzoek uitgevoerd waarbij er gekeken wordt naar verschillende Git opzetten (workflows) die beschikbaar zijn. Omdat niet alle workflows zullen voldoen aan de randvoorwaarden van de opleiding, zal een tweetal worden gekozen en specifieker worden onderzocht naar de gestelde randvoorwaarden. Voor dit onderzoek wordt er gebruik gemaakt van een desk-research-onderzoek.

3.4 Welke software is er beschikbaar om deze opzet te realiseren?

De vierde deelvraag beslaat het invullen van de gekozen opzet uit de vorige deelvraag aan de hand van specifieke software die uiteindelijk hulp kunnen bieden om de gewenste opzet in te richten. Hierbij wordt er gezocht naar software die aansluit bij de gestelde randvoorwaarden en de mogelijkheid biedt om aan de gekozen Git opzet te voldoen.

Voor het uiteindelijk kiezen van het geschikte software pakket zullen een aantal criteria worden opgesteld. Deze komen wederom voort uit de randvoorwaarden en de gekozen Git opzet. De systemen worden vervolgens aan deze randvoorwaarden getoetst en een uiteindelijke score zal bepalen welk pakket het beste uit de test komt. Dit zal gebeuren aan de hand van een literatuuronderzoek.

3.5 Wat zijn de stappen en functionaliteiten van deze Git opzet en hoe werkt deze precies?

Voor de laatste deelvraag zal de gekozen Git opzet in het gekozen software pakket getracht gerealiseerd worden. Hierbij zal een uitleg worden gegeven van hoe dit precies in zijn werk gaat, hoe gebruikers toegevoegd kunnen worden, hoe deze gekoppeld kunnen worden aan projecten en hoe uiteindelijk code gepusht kan worden naar deze projecten. Deze stappen worden uitgewerkt in een handleiding die meegegeven zal worden bij de opzet.

4 Resultaten

In dit hoofdstuk worden de resultaten van het onderzoek behandeld. Per deelvraag wordt er gekeken naar de resultaten die zijn vergaard gedurende het onderzoek.

4.1 Verschillen GIT & SVN

In de eerste deelvraag wordt er gekeken naar de grote verschillen tussen GIT en SVN en wat voor een impact deze hebben op de probleemsituatie die de HZ University of Applied Sciences nu heeft. De vraag betreffende deze deelvraag luidt als volgt:

Wat zijn de grote verschillen van Git tussen SVN?

4.1.1 Grote verschillen

Tussen GIT en SVN zijn een aantal verschillen te zien. In de onderstaande paragrafen zijn deze verschillen kort beschreven en geanalyseerd in de context van de probleemstelling.

Distributed

Het eerste, en meteen ook grootste verschil, tussen beide systemen is het feit dat GIT een distributed systeem is en SVN niet. Dit concept is al langere tijd in de running, maar vereist toch nog enige uitleg.

Beide systemen hebben een gecentraliseerde repository en/of server waar het project op staat. Waarbij SVN gebaseerd is om direct van deze repository het project te halen en hier aan te werken, is bij GIT het principe om eerst een kopie van deze repository op te slaan in een eigen omgeving. Dit wordt ook wel de clone repository genoemd. Dit distributed concept heeft hele grote voordelen wanneer er in teams wordt gewerkt. (Siddique, 20-09-2010)

Het hebben van een distributed versiebeheer systeem heeft een aantal grote voordelen. Vanwege het feit dat de gebruiker altijd een eigen cloned repository heeft, is er geen communicatie met de server vereist om verder te werken. Dit kan bij de SVN versie ook, maar niet met zoveel details als GIT heeft. Zo kan er met GIT ook in de eigen omgeving clones, branches e.d. worden aangemaakt ter ondersteuning van het programmeren. (Git-scm.com, z.d.)

Metadata

Het tweede verschil is dat GIT zijn gegevens en bestanden opslaat als metadata. Dit heeft als groot voordeel dat wanneer er een clone wordt gemaakt va de centrale repository, alles uit deze repository wordt meegenomen naar de lokale repository. Dit beslaat niet alleen het project, maar ook de branches, version history en tags. (Siddique, 20-09-2010)

Het grote voordeel hiervan is dat voor iedereen in het project direct duidelijk is waar aan gewerkt wordt en wat er al gedaan is, zelfs in de lokale omgeving.

Globale revision

Omdat GIT en SVN zo anders zijn opgezet, zijn er ook een aantal features van SVN die gemist worden in GIT. Een van deze is dat GIT geen globaal revisie heeft. Hiermee wordt bedoeld dat er geen laatste snapshot van het project beschikbaar is. Dit omdat er met ontzettend veel branches gewerkt kan worden en het lastig te bepalen is wat de laatste versie van het project is. (Siddique, 20-09-2010)

Dit heeft als nadeel dat het lastig te zien is wat de laatste versie is van het project en wat de uiteindelijk ingeleverde versie is van de student. Versienummers worden namelijk ook niet weergeven in GIT.

4.1.2 Voor- en nadelen

De bovenstaande verschillen leggen de basis tot overweging voor het wisselen van versiebeheer systeem betreffende de probleemsituatie. Zoals gezegd hebben beide systemen zo hun voor- en nadelen.

Uit de verschillen kan al globaal worden opgemaakt dat GIT voornamelijk gebruikt wordt voor middelmatige tot grote projecten waar veel man aan werkt. Iedere programmeur heeft zijn eigen versie van het project beschikbaar en kan daar veranderingen aan maken naar wens zonder dat het de andere projectleden schaadt. Bij SVN heeft dit een grote impact aangezien alle veranderingen direct naar de centrale repository gepusht worden. Dit is iets waar de HZ zeker rekening mee moet houden, aangezien de gehele opzet heel anders werkt.

4.2 Randvoorwaarden

De tweede deelvraag bevat de randvoorwaarden waaraan het GIT versiebeheer systeem, volgens de beheerder en gebruiker, moet voldoen. Deze randvoorwaarden zijn opgesteld aan de hand van een meeting met de opdrachtgever. De deelvraag voor dit onderdeel luidt:

Wat zijn de belangrijke randvoorwaarden die de opleiding stelt aan het systeem?

4.2.1 Wensen

Tijdens het samenzitten met de opdrachtgever zijn de volgende wensen en/of randvoorwaarden naar voren gekomen. Hier moet rekening mee gehouden worden wanneer er gezocht gaat worden naar een vervangend GIT systeem.

Eenvoudig beheersbaar

Volgens de opdrachtgever moet de GIT omgeving eenvoudig beheersbaar te zijn. Dit wil zeggen dat er zonder al te veel moeite repositories kunnen worden aangemaakt en worden beheerd wanneer nodig. Iets wat hiermee zou kunnen helpen is bijvoorbeeld het hebben van een user interface.

Elke student zijn eigen omgeving

Omdat het gaat om een opleiding is het niet gewenst dat studenten de revisies van zijn/haar medestudenten kan inzien. De omgeving van de student moet daarom dus ook alleen voor hem/haar beschikbaar zijn.

Passend binnen huidige omgeving

Het huidige SVN versiebeheer systeem draait op een Ubuntu 14.4 server. Dit is de centrale server binnen de HBO-ICT opleiding van de HZ waarop projecten van de studenten wordt opgeslagen. Hierop zal dus ook de GIT omgeving op moeten draaien.

4.3 GIT Opzet

De derde deelvraag beslaat het opzoeken van een passende GIT workflow/opzet voor de randvoorwaarden die de opdrachtgever gesteld heeft aan het systeem. Hiervoor is een literatuuronderzoek uitgevoerd. De deelvraag voor dit gedeelte luidt als volgt:

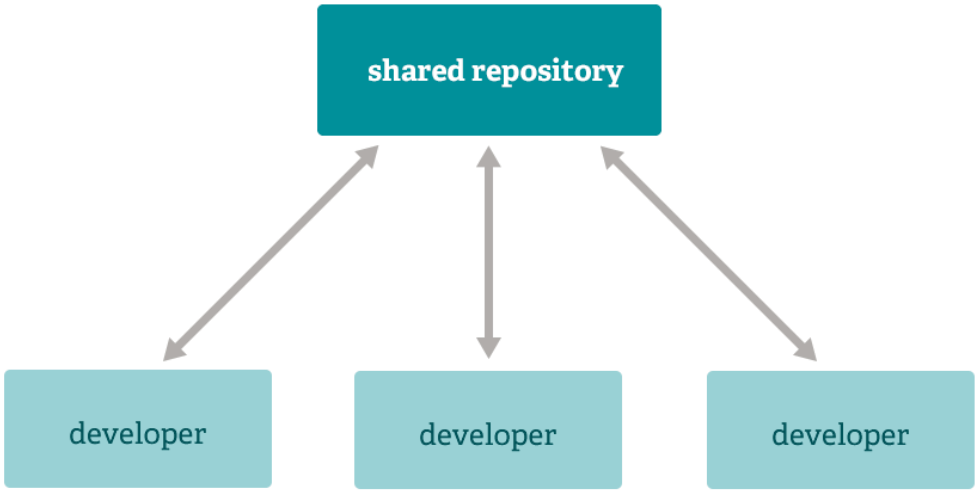
Welke Git opzet past het beste bij deze randvoorwaarden?

4.3.1 Workflows

Doordat GIT op een gedistribueerde manier omgaat met haar versiebeheer, is het mogelijk om deze op een eindeloos aantal manieren in te vullen. In het kader van het onderzoek is er gezocht naar een tweetal oplossingen die van toepassing zullen zijn in de werkomgeving van de HZ.

Subversion-Style Workflow

Zoals de naam bij deze werkwijze al doet vermoeden, wordt er bij de eerste workflow gebruik gemaakt van een oplossing die veel gemeen heeft van de subversion (SVN) versiebeheer omgeving. Hierbij is er één gezamenlijke repository waarin de ontwikkelaars hun deel van de code naar kunnen pushen. Daarbij zit GIT zo in elkaar dat er niet gepusht kan worden wanneer er met een verouderde versie van de shared repository wordt gewerkt, zo is de kans op conflicten erg laag. (Git-scm.com, z.d.)



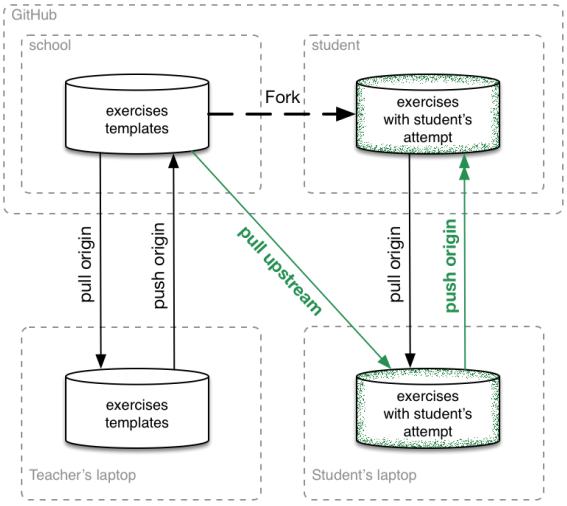
Figuur 4.1 – Subversion-Style Workflow

Deze workflow wordt tevens al gebruikt in de huidige SVN omgeving en werkt naar behoren voor de onderwijsinstelling. Een van de grote voordelen voor de HZ is dat iedereen zijn eigen repository heeft waarin hij/zij zijn code kan pushen. Dit zorgt er tevens voor dat alleen hij/zij toegang heeft tot deze repository waardoor ook andere studenten hier geen code vanaf kunnen halen. (Git-scm.com, z.d.)

Het nadeel van het gebruik van de SVN workflow is dat het niet het volledig potentieel van GIT ondersteund. De huidige workflow blijf hetzelfde en er wordt vrij weinig tot geen gebruik gemaakt van de branch en tag technieken van GIT. (Git-scm.com, z.d.)

Student-Teacher Workflow

De tweede oplossing is de student-teacher workflow, oftewel een workflow die gebruikt kan worden in scholen. Hierbij wordt de opdracht eerst door de docent op een template repository geplaatst. De studenten kunnen deze dan vervolgens pullen naar hun eigen werkomgeving, bijvoorbeeld op hun laptop, en hier vervolgens aan werken. Wanneer de student klaar is kan hij/zij deze pushen naar een eigen forked repository waar alleen zijn/haar antwoorden staan. De docent kan dan vervolgens deze antwoorden weer inzien en nakijken. (Saunier, 21-04-2014)



Figuur 4.2 – Student-teacher workflow

Het grote voordeel van de student-teacher workflow is dat elke student zijn eigen forked repository heeft binnen een project. Tevens staat alle informatie over de opdracht in de exercise template waardoor direct duidelijk is wat er gedaan moet worden als deze gepullt is naar de student zijn eigen omgeving. De docent kan tevens via een dashboard zien hoe het staat met de opdrachten van de studenten en wie er wel wat heeft ingeleverd en wie niet. (Saunier, 21-04-2014)

Het nadeel hier is dat het lastig te waarborgen is dat studenten alleen de code van hun eigen fork inzien. Voor het beveiligen van de forks moet goed worden nagedacht aangezien dit een groot struikelblok is. (Saunier, 21-04-2014)

4.3.2 Keuze

Kijkend naar de bovenstaande workflows kan geconcludeerd worden dat beide zo zijn voor- en nadelen hebben. Aan het eind van de rit is de teacher-student workflow aantrekkelijker voor de HZ vanwege het feit dat deze de functionaliteiten van GIT goed implementeert. Het gebruik van de SVN workflow biedt geen verdere voordelen ten opzichte van de huidige situatie, waardoor het onderzoek hiervoor ophoud. Voor de student-teacher workflow wordt in de volgende deelvraag gezocht naar een passend systeem.

4.4 GIT Software

De vierde deelvraag beslaat het onderzoeken van verschillende software pakketten die de mogelijkheid bieden om een Git server op te zetten en te beheren. Voor deze deelvraag is een literatuuronderzoek uitgevoerd. De deelvraag luidt als volgt:

Welke software is er beschikbaar om deze opzet te realiseren?

4.4.1 Criteria

Om de software te toetsen aan de gestelde eisen is er gebruik gemaakt van een selectie aan criteria. Deze criteria zijn voort gekomen uit de randvoorwaarden alsmede de gekozen Git workflow uit de voorgaande deelvraag. De criteria luiden als volgt:

- **User authentication**
  Of er een mogelijkheid bestaat tot user authentication om gebruikers toegang te verlenen aan

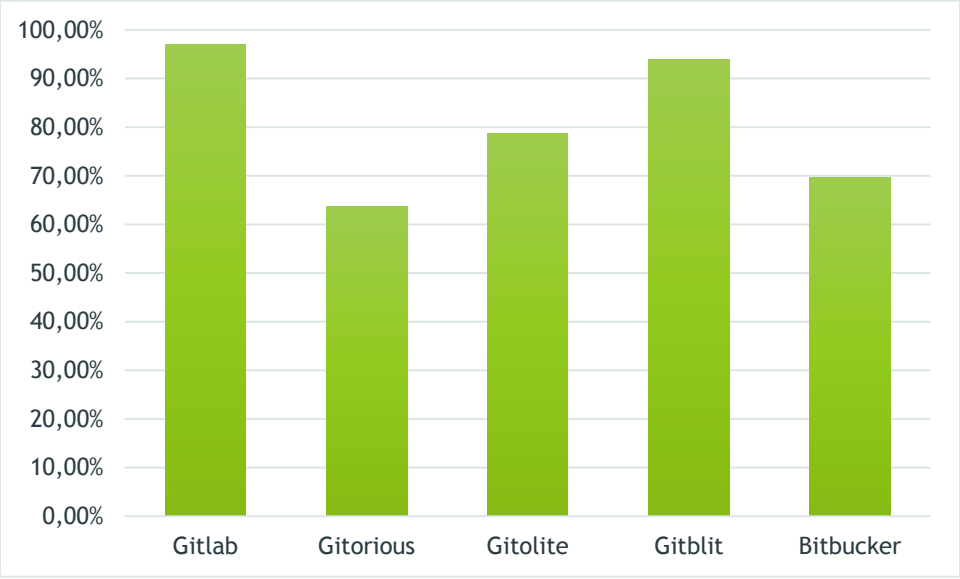
bepaalde projecten en/of mappen. Dit is belangrijk voor het veilig houden van de omgeving en het niet kunnen kopiëren van andere studenten.

- **Repository forking**
  Bij de Git workflow wordt er gebruik gemaakt van repository forking, waarbij de student zijn opgaven inlevert in zijn fork van de master repository. De mogelijkheid om dit te doen moet dus wel bestaan.
- **Webhooks**
  Mocht het gewenst zijn om eventuele andere web toepassingen te koppelen aan de Git omgeving, dan kan dit gedaan worden door middel van webhooks. Dit speelt bijvoorbeeld een rol bij het toevoegen van een user interface of een dashboard.
- **User interface**
  Het hebben van een user interface voor het beheren van de omgeving helpt enorm om het beheer eenvoudig te houden. Dit zorgt dus ook voor pluspunten mocht de software hier aan voldoen.
- **Huidige omgeving**
  De software moet uiteindelijk in de huidige omgeving draaien, het is dus belangrijk dat dit ook mogelijk is.

Elk van deze criteria heeft een weging toegekend gekregen en is getoetst op een score van 1 t/m 3.

4.4.2 Uitkomst

De uitkomst van het Git software onderzoek is in de volgende grafiek te zien. Hierbij komt naar voren dat Gitlab de winnaar is, hij scoort namelijk nagenoeg perfect op alle criteria. Gitblit is een goede tweede kandidaat met maar een punt minder dat Gitlab. De overige software pakketten voldeden of niet aan een criteria, of waren betaald. De resultaten van het onderzoek zijn terug te vinden in bijlage A.



Figuur 4.3 – Uitkomst GIT systemen onderzoek

Voor het opzetten van de Git omgeving in de volgende deelvraag is dus voor Gitlab gekozen, omdat deze het best uit de test kwam.

4.5 Realisatie

De vijfde, en tevens laatste, deelvraag beslaat het onderzoek naar de realisatie van de Gitlab instantie in een virtuele omgeving waarin de gekozen workflow geïmplementeerd wordt. Voor deze deelvraag is een experiment uitgevoerd. De deelvraag luidt als volgt:

Wat zijn de stappen en functionaliteiten van deze Git opzet en hoe werkt deze precies?

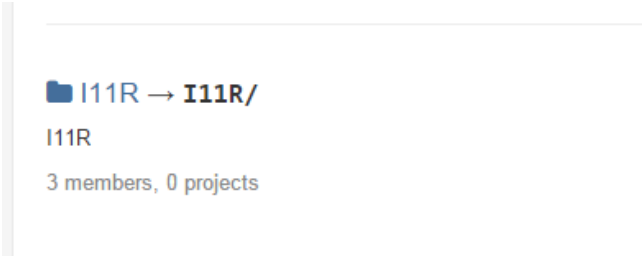
4.5.1 Opzet

Voor de realisatie van de Gitlab omgeving is gebruik gemaakt van een Virtuele Machine. Deze machine komt overeen met de omgeving die gebruikt wordt binnen de HZ University of Applied Sciences zodat de resultaten zo accuraat mogelijk zijn.

Voor Gitlab is er gebruik gemaakt van een Gitlab Virtual Machine installatie, dit is een omgeving die al is opgezet door Gitlab zelf om zo direct aan de slag te kunnen met de software. Dit zorgt er tevens voor dat alle instellingen al direct goed staan. (Bitnami.com, z.d.)

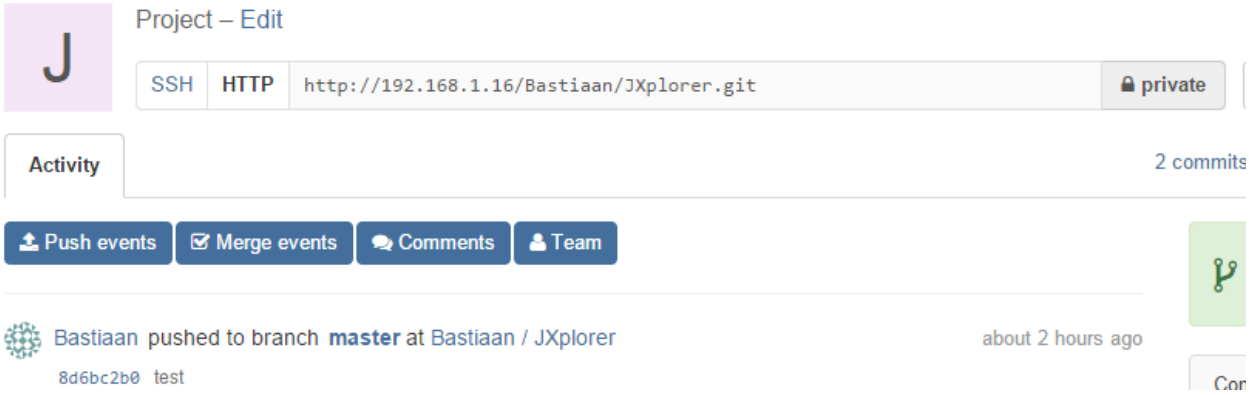
4.5.2 Uitkomst

In de Gitlab omgeving is de student-teacher workflow, zoals beschreven staat in deelvraag 3, uitgewerkt. Hiervoor is een handleiding geschreven die teruggevonden kan worden in bijlage 2. In deze handleiding wordt er dieper ingegaan op de relevante functionaliteiten van de Gitlab omgeving. Denk hierbij aan bijvoorbeeld het toevoegen van klassen met studenten, zie figuur 4.4:



Figuur 4.4 - Een klas in GitLab

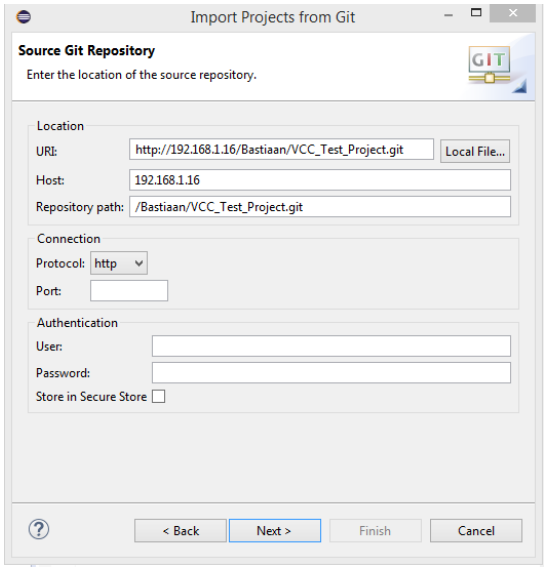
Ook is er gekeken naar het aanmaken van de projecten en hoe de studenten hier hun eigen code aan kunnen toevoegen. Een voorbeeld van zo'n dergelijk project gezien vanaf de student is hieronder te zien:



Figuur 4.5 - Het persoonlijke dashboard van de student bij een project.

Als laatste is er specifiek gekeken hoe projecten naar de Gitlab instantie gepusht en gepullt kunnen worden aan de hand van de ontwikkelomgeving Eclipse. Deze omgeving wordt op de HZ vaak gebruikt

door studenten. In de onderstaande afbeelding is te zien hoe een project gepullt kan worden van de Gitlab instantie:



Figuur 4.6 – Het pullen van een project met Eclipse

Meer uitleg over de functionaliteiten is terug te vinden in Bijlage 2.
De virtuele machine instantie kan via de volgende link gedownload worden: https://db.tt/u1vQG98v

5 Discussie

In dit hoofdstuk wordt er gekeken naar de vergaarde resultaten en de gebruikte methode en hoe deze elkaar beïnvloed hebben.

5.1 Verschillen GIT & SVN

Voor het onderzoek naar de verschillen tussen Git & SVN is gebruik gemaakt van een literatuuronderzoek. Hierbij kwam naar voren dat er een aantal grote verschillen tussen de systemen zitten die aan de ene kant voordelig zijn, maar aan de andere kant ook zeker nadelig. Aan de hand van dit literatuuronderzoek is er gezocht naar specifieke bronnen die dieper ingingen op de functionaliteiten van beide versiebeheer systemen. Deze functionaliteiten zijn vervolgens met elkaar vergeleken. Ook is er gezocht naar de grote verschillen en gekeken of deze overeen kwamen met de gevonden resultaten. Als resultaat is er dan ook een lijst met grote verschillen gekomen die relevant zijn voor de probleemsituatie, hierdoor kan de gekozen methode als succesvol beschouwt worden.

5.2 Randvoorwaarden

Voor deelvraag twee is er gebruik gemaakt van de interview methode waarbij er samengezeten is met de opdrachtgever om de gehele opdracht te bespreken. Hierbij kwam naar voren dat het eenvoudig beheren van de omgeving een belangrijk punt is. Daarnaast moet er gekeken worden naar de beveiliging van de repositories zodat studenten niet bij elkaar kunnen kijken. Ook het feit dat de Gitlab instantie in de huidige omgeving moet kunnen draaien is een belangrijk criteria. Door samen te zitten met de opdrachtgever zijn deze voorwaarden wat dieper besproken en is een betere lijst kunnen maken met relevante voorwaarden. Daarnaast heeft de eigen ervaring ook een rol gespeeld, aangezien er zelf ook gebruik is gemaakt van de omgeving maar dan vanuit het oogpunt van de student.

5.3 GIT Opzet

Het onderzoeken naar een geschikte GIT opzet besloeg de focus van deelvraag 3. Hiervoor is een literatuuronderzoek uitgevoerd waarbij er gezocht is naar verschillende opzetten/workflows die passen bij de gestelde randvoorwaarden uit deelvraag twee. Veel van de gevonden workflows zijn echter gericht op een team van veel programmeurs, wat bij de HZ geen optie is. Hierdoor zijn er maar een tweetal workflows gevonden die echt aansluiten bij de randvoorwaarden. Vanwege deze reden zijn er ook geen criteria opgesteld waarop de workflows getoetst zijn, wat achteraf gezien wel beter gedaan had kunnen worden. Vervolgens is er gekeken naar de voor- en nadelen van de workflows en is er een keuze gemaakt welke het beste past bij de HZ. Uiteindelijk is hieruit de student-teacher workflow gekomen. De gekozen methode sluit goed aan bij de informatievraag naar bestaande workflows die gebruikt kunnen worden. Met een geschikte workflow als resultaat kan deze methode ook als succesvol beschouwd worden.

5.4 Git Software

De vierde deelvraag beslaat het onderzoeken van een geschikt softwarepakket waarmee de workflow gerealiseerd en beheerd kan worden in de huidige omgeving. Hiervoor is wederom een literatuuronderzoek uitgevoerd naar beschikbare systemen. Deze systemen zijn vervolgens getoetst aan de hand van criteria die zijn opgesteld uit de randvoorwaarden. Elk criteria heeft een weging gekregen en de systemen zijn vervolgens beoordeeld aan de hand van gevonden literatuur. Hieruit kwam vervolgens naar voren dat Gitlab de beste omgeving is voor de eisen en wensen van de HZ. Hoewel het antwoord goed onderbouwd is had een experiment bij elk software pakket een beter resultaat geleverd, echter was dit niet te voltooien in verband met tijd en tevens ook kennis. Desondanks kan de gebruikte methode als succesvol beschouwd worden.

5.5 Realisatie

De laatste deelvraag gaat over de realisatie van de gekozen workflow in het gekozen software pakket. Voor dit onderzoek is een experiment uitgevoerd waarbij de aspecten in een virtuele omgeving zijn geïnstalleerd en zijn getest. Voor de Gitlab instantie is gebruik gemaakt van een bestaande Virtual Machine instantie, dit omdat de onderzoeker het niet voor elkaar kreeg om een compleet nieuwe installatie werkend te krijgen. Uiteindelijk is er gekozen om dus een bestaande instantie te gebruiken en hier de workflow op te testen. Deze workflow is uiteindelijk succesvol geïmplementeerd waardoor er studenten, docenten, klassen en projecten aangemaakt kunnen worden. Tevens is het mogelijk om te pushen en pullen naar de omgeving met daarbij de gewenste authenticatie. Ondanks het feit dat de installatie niet geheel vlekkeloos verliep, zijn de functionaliteiten wel allemaal doorgevoerd. Daardoor kan ook deze methode als succes beschouwd worden.

6 Conclusie

Dit laatste hoofdstuk, de conclusie, bevat een antwoord op de hoofdvraag en een advies aan de opdrachtgever over hoe nu verder gegaan kan worden met de gevonden resultaten.

6.1 Centrale onderzoeksvraag

De resultaten van het uitgevoerde zijn van groot belang voor het beantwoorden van de centrale onderzoeksvraag. Deze onderzoeksvraag luidt:

Hoe kan de ICT opleiding van de HZ University of Applied Sciences het best gebruik maken van het versiebeheer systeem Git?

Zoals in het onderzoek naar voren is gekomen kan de ICT opleiding van de HZ University of Applied Sciences het best gebruik maken van de student-teacher workflow in combinatie met het Gitlab software pakket als zijnde het versiebeheer systeem. Ondanks dat er grote verschillen zitten tussen Git en SVN, het huidige versiebeheer systeem, is de realisatie van een soortgelijke omgeving toch een oplossing. De student-teacher workflow stelt de beheerder in staat om complete klassen in een keer toe te voegen aan projecten en deze projecten ook te beheren. Daarnaast is het beheren van al deze projecten eenvoudig gemaakt door middel van het Gitlab dashboard, waarin alle aspecten van het versiebeheer systeem beheerd en gecontroleerd kunnen worden. De realisatie heeft aangetoond dat deze opzet ook daadwerkelijk werkt en een mogelijke oplossing biedt voor de probleemsituatie.

6.2 Aanbeveling

Ondanks dat het onderzoek heeft aangetoond dat het mogelijk is om binnen de ICT opleiding van de HZ University of Applied Sciences een Git omgeving te gebruiken, wordt het toch nog aanbevolen om hier een goede overweging in te maken. Het huidige versiebeheer systeem werkt nog naar behoren en voldoet prima aan de voorwaarden die gesteld worden. Daarnaast is de migratie naar een nieuw versiebeheer systeem een flinke klus, vooral als alle bestanden bewaard dienen te blijven. Het grote nadeel van Git ten opzichte van SVN is dat Git zich voornamelijk focust op grote projecten met grote groepen gebruikers. Hierdoor komen de functionaliteiten van Git pas echt goed tot zijn recht. Wanneer er met kleine groepen gebruikers gewerkt wordt, zijn deze functionaliteiten al snel overbodig, denk aan het maken van branches of tags.

Mocht de opleiding er toch voor kiezen om voor een Git omgeving te gaan, dan is de aanbeveling om nog eens goed te kijken naar de authenticatie rechten en het beheer van de gehele omgeving. Gitlab biedt hiervoor uitgebreide oplossingen echter het testen hiervan is niet volledig uitgevoerd.

7 Literatuurlijst

Saunier, S. (21-04-2014) *Practical Example of using git in a school*. Opgehaald van http://sebastien.saunier.me/blog/2014/04/21/practical-example-of-using-git-in-a-school.html

Git-scm.com (z.d.) Distributed. Opgehaald van http://git-scm.com/about/distributed

Git-scm.com (z.d.) Branching and merging. Opgehaald van http://git-scm.com/about/branching-and-merging

Chacon, S. & Straub, Ben. (z.d.) Git and Other Systems - Git as a Client. Opgehaald van http://git-scm.com/book/en/v2/Git-and-Other-Systems-Git-as-a-Client

Chacon, S. & Straub, Ben. (z.d.) Git Basics – Getting a Git Repository. Opgehaald van http://git-scm.com/book/en/v2/Git-Basics-Getting-a-Git-Repository

Bitnami.com (z.d.) GitLab Virtual Machines. Opgehaald van https://bitnami.com/stack/gitlab/virtual-machine

Siddique (20-09-2010) *5 Fundamental differences between GIT & SVN*. Opgehaald van http://boxysystems.com/index.php/5-fundamental-differences-between-git-svn/

Gitlab.com (z.d.) Gitlab – Features. Opgehaald van https://about.gitlab.com/features/

Gitlab.org (z.d.) GitLab Community Edition. Opgehaald van https://gitlab.com/gitlab-org/gitlab-ce

v.d. Ploeg, N. (23-08-2013) How To Set Up GitLab As Your Very Own Private GitHub Clone. Opgehaald van https://www.digitalocean.com/community/tutorials/how-to-set-up-gitlab-as-your-very-own-private-github-clone

Hethey, J. M. (2013) *GitLab Repository Management.* Opgehaald van https://www.safaribooksonline.com/library/view/gitlab-repository-management/9781783281794/cover.html

Gitorious.com (z.d.) *Products – Local Install.* Opgehaald van http://www.gitorious.com/local_install/

Gitorious.com (z.d.) *Features.* Opgehaald van http://www.gitorious.com/#features

Vaughan, T. (22-10-2009) *Fork From Gitorious?* Opgehaald van https://groups.google.com/forum/#!topic/github/-5tJlQjt1Dk

Gitolite.com (z.d.) *Gitolite Overview.* Opgehaald van http://gitolite.com/gitolite/overview.html

Gitolite.com (z.d.) *Using hooks.* Opgehaald van http://gitolite.com/gitolite/g2/hooks.html

Ellingwood, J. (15-10-2013). *How To Use Gitolite to Control Access to a Git Server on an Ubuntu 12.04 VPS.* Opgehaald van https://www.digitalocean.com/community/tutorials/how-to-use-gitolite-to-control-access-to-a-git-server-on-an-ubuntu-12-04-vps

Sitaramc (02-11-2014). *Gitolite*. Opgehaald van https://github.com/sitaramc/gitolite

Gitblit.com (z.d.) *Screenshots.* Opgehaald van http://gitblit.com/screenshots.html

Gitblit.com (z.d.) *Features.* Opgehaald van http://gitblit.com/features.html

Atlassian.com (z.d.) *Add Users, Set Permissions, and Review Account Plans.* Opgehaald van https://confluence.atlassian.com/display/BITBUCKET/Add+Users,+Set+Permissions,+and+Review+Account+Plans

Atlassian.com (z.d.) *Forking a Repository. Opgehaald van* https://confluence.atlassian.com/display/BITBUCKET/Forking+a+Repository

Atlassian.com (z.d.) *Manage Bitbucket hooks. Opgehaald van* https://confluence.atlassian.com/display/BITBUCKET/Manage+Bitbucket+hooks

Bitbucket.org (z.d.) *Features.* Opgehaald van https://bitbucket.org/#features

GitWiki (z.d.) *Tips And Tricks*. Opgehaald van https://git.wiki.kernel.org/index.php/TipsAndTricks

Apache.com (z.d.) *Apache Subersion*. Opgehaald van https://subversion.apache.org/

8 Bijlage A – Git software onderzoek

GIT Server	CRITERIA: WEGING	User authentication Score 3	Repository forking Score 2
GITLAB		Mogelijkheid tot vijf verschillende user groepen van administrator tot en met guest. Elk met hun eigen account. User groepen worden toegekend bij projecten.	Mogelijkheid binnen de gratis versie van GitLab. De fork laat de gebruiker een repostiory aanmaken op zijn namespace binnen het project.
Final Score: 32	SCORE	3	3
GITORIOUS		Voor projecten en groepen kunnen access levels worden ingesteld. Dit per gebruiker of per groep.	Forking is niet beschikbaar in gitorious, hiervoor moet direct een nieuwe repo voor worden gemaakt.
Final Score: 21	SCORE	3	1
GITOLITE		Met Gitlolite kan specifieke branch en tag access worden geregeld voor de projecten. Multi users zijn dus ook ondersteund.	Het maken van forks is een optie, dit dient wel in de shell te gebeuren.
Final Score: 26	SCORE	3	2
GITBLIT		Er zijn zes verschillende niveaus voor toegang tot de bestanden voor de gebruikers Per project komen hier nog eens vier restrictie opties bij.	Er is een optionele functie beschikbaar voor het forken van een repository naar een lokale namespace.
Final Score: 31	SCORE	3	3
BITBUCKET		Bitbucket heeft user access op branch en tag niveau. Het toevoegen van extra users kost echter wel geld.	Bij het forken van een repository zijn vele verschil-lende opties nog mogelijk, zoals de access levels van de gebruikers.
Final Score: 23	SCORE	2	3

Webhooks Score 1	User Interface Score 2	Huidige omgeving Score 3
Het hebben van een webhook is deels aanwezig in de gratis versie. Voor meer functies moet de enterprise edition gekocht worden.	GitLab heeft een zeer uitgebreide user interface voor zowel de administrator alsmede de student. Veel informatie kan teruggevonden worden in de interface.	Het installeren van de GitLab instantie is toepasbaar in de huidige omgeving die gebruikt wordt op de HZ.
2	3	3
Gitorious heeft een extra optie voor het integreren van webhooks voor het toevoegen van extra web functies als dashboards.	Er is een web interface aanwezig maar deze beslaat alleen het inzien van de merge requests en een wiki over het project.	Gitorious is overgekocht door GitLab en wordt niet verder ondersteund na het eind van mei 2015, iedereen wordt gevraagd over te stappen naar Gitlab
3	2	1
Het koppelen van web hooks aan de Gitolite server is geen probleem, dit dient echter wel in de bestanden zelf aangepast te worden.	Er is geen specifieke user interface beschikbaar voor gitolite, er zijn wel een aantal zelgemaakte interfaces, maar deze gaan niet heel diep in op het managen van de repos.	Het is mogelijk om de gitolite git server te installeren op de huidige omgeving van de HZ. Tevens zijn er veel mogelijkheden tot aanpasen van de instantie.
2	1	3
Er is een hook script beschikbaar welke geregeld kan worden per repository. Veel informatie is hier niet over beschikbaar.	Er is een zeer uitgebreide interface aanwezig waarin alle opties geregeld kunnen worden en dashboards gezien kunnen worden van de projecten.	Het installeren van de gitlib instantie in de huidige omgeving is een mogelijkheid.
1	3	3
Bitbucker biedt de mogelijk-heid tot hooks, echter alleen een specifiek aantal die ze zelf ondersteunen.	Er is een web interface aanwezig om de status van de projecten in te zien. Ook in de client is dit mogelijk.	Bitbucket voor 5 users is gratis meer users vereist een betaalde versie. Dit is geen optie voor de HZ wegens kosten Verdere installatie in de omgeving is wel mogelijk.
2	3	1

9 Bijlage B – Handleiding Gitlab

Installatie

Downloadlink van de Virtual Machine instantie: https://db.tt/u1vQG98v

Bijgevoegd is een instantie van de Gitlab Virtual Machine. Om deze te installeren dient deze allereerst geïmporteerd te worden in een Oracle VM VirtualBox omgeving. Klik in dit programma op Bestand -> Appliance Importeren. Selecteer hierbij het juiste Virtual Machine bestand en druk op importeren. Alle instellingen staan al goed. Start vervolgens de Gitlab Virtual Machine (dit duurt een tijdje).

Web interface

Voor toegang tot de web interface van Gitlab dient er naar de url gegaan te worden die bij het opstarten van de virtual machine getoond wordt. De url welke gegeven wordt is:

http://192.168.1.16/

De inloggegevens als administrator zijn:

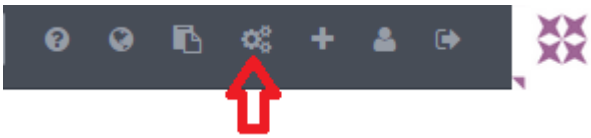
Username: user@example.com
Wachtwoord: bitnami1

Eenmaal ingelogd is het dashboard te zien van de administrator gebruiker.

Toevoegen gebruikers

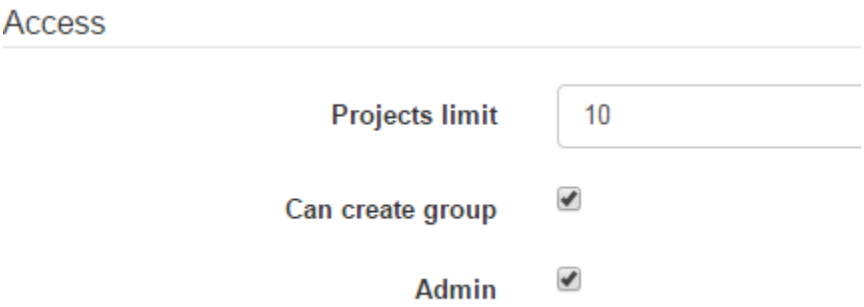
Binnen het kader van de gekozen workflow zijn er een tweetal gebruikers, de docenten en de studenten. Hieronder wordt kort toegelicht hoe deze aangemaakt kunnen worden.

Als eerste dient er gegaan te worden naar het administratie panel van de administrator. Deze kan gevonden worden rechtsboven in de menubalk. De drie tandwieltjes is de knop die gebruikt dient te worden:



Vervolgens opent het administratie scherm zich en is er aan de linkerkant in het menu een kopje “Users” te vinden. Deze knop opent het user menu waarin alle gebruikers beheert kunnen worden. Voor het toevoegen van een nieuwe gebruiker drukt men op “New User”, wat rechts van boven in het scherm gevonden kan worden. Dit opent het new user scherm.

Voor het toevoegen van de docenten en de studenten dient dit formulier ingevuld te worden. Zaken als social media kunnen ook toegevoegd worden, echter is dit niet van belang voor de HZ. Het grote verschil tussen de docenten en studenten zit hem in het Access menu.

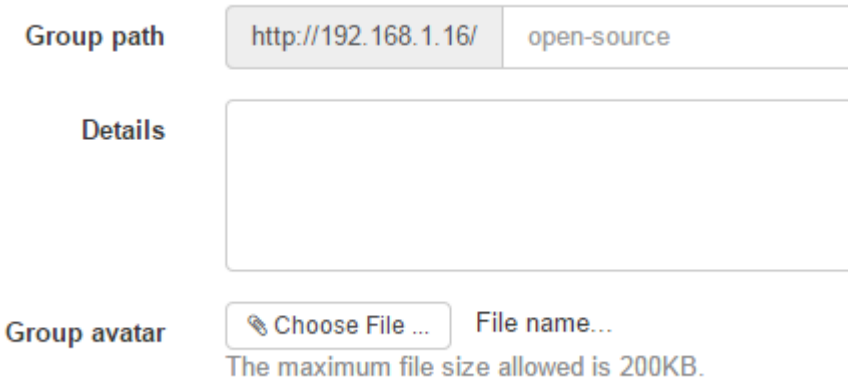


Voor de docenten dienen beide hokjes (Can create group & Admin) aangevinkt te worden. Hierdoor kunnen docenten projecten en groepen aanmaken en beheren. Ook het projecten limiet kan verhoogt worden. Wanneer een docent of student is aangemaakt krijgt hij/zij een mail met daarin de inloggegevens. Vervolgens kunnen zij zelf inloggen in de web interface en hun projecten beheren.

Toevoegen klassen

Om gebruikers eenvoudig te beheren en toe te voegen aan projecten kunnen er groepen worden aangemaakt. Deze fungeren als klassen/studiejaren van de opleiding. Om deze groepen aan te maken dient men wederom te gaan naar het admin menu:

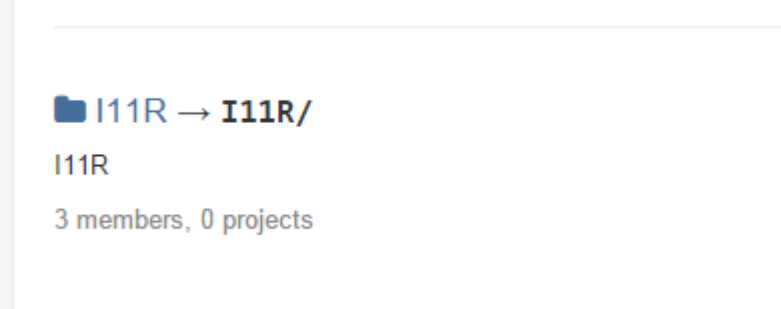
Voor het aanmaken van een nieuwe groep drukt met op de “New Group” knop rechts bovenin de pagina. Hierdoor wordt het nieuwe groep venster geopend.



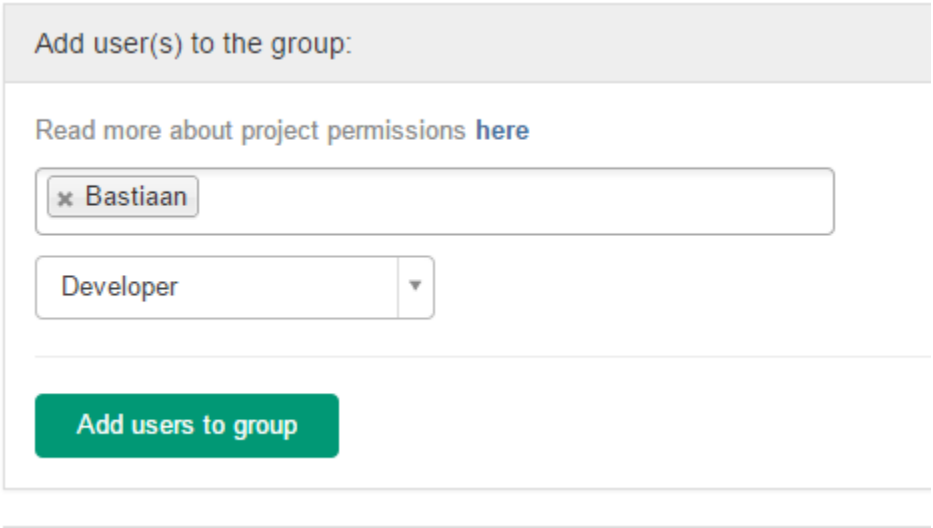
De group path bepaald de naam van de groep, bijvoorbeeld I11R. Dit is het pad waar de projecten van de groepen in worden opgeslagen. Meer details kunnen gegeven worden over de groep al is dit niet per se nodig. Ook een afbeelding kan toegevoegd worden.

Toevoegen gebruikers aan klas

Na het aanmaken van de klas is deze nog wel leeg. Om studenten toe te voegen aan een klas gaat men naar het “group” scherm. In dit scherm staat een overzicht van alle aangemaakte groepen. Hier is er een optie om de groep te bewerken en te verwijderen, dit is echter niet nodig tenzij de naam van de groep aangepast moet worden.



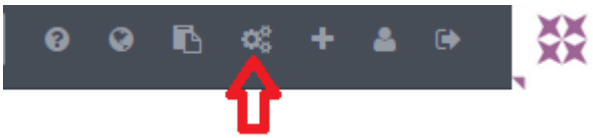
Voor het toevoegen van gebruikers klikt men op de naam van de groep, in het geval van de afbeelding hierboven is dit I11R. In het volgende scherm kan informatie gevonden worden over de groep en de gebruikers die er in zitten. Het toevoegen van de studenten kan aan de rechter kant van het scherm:



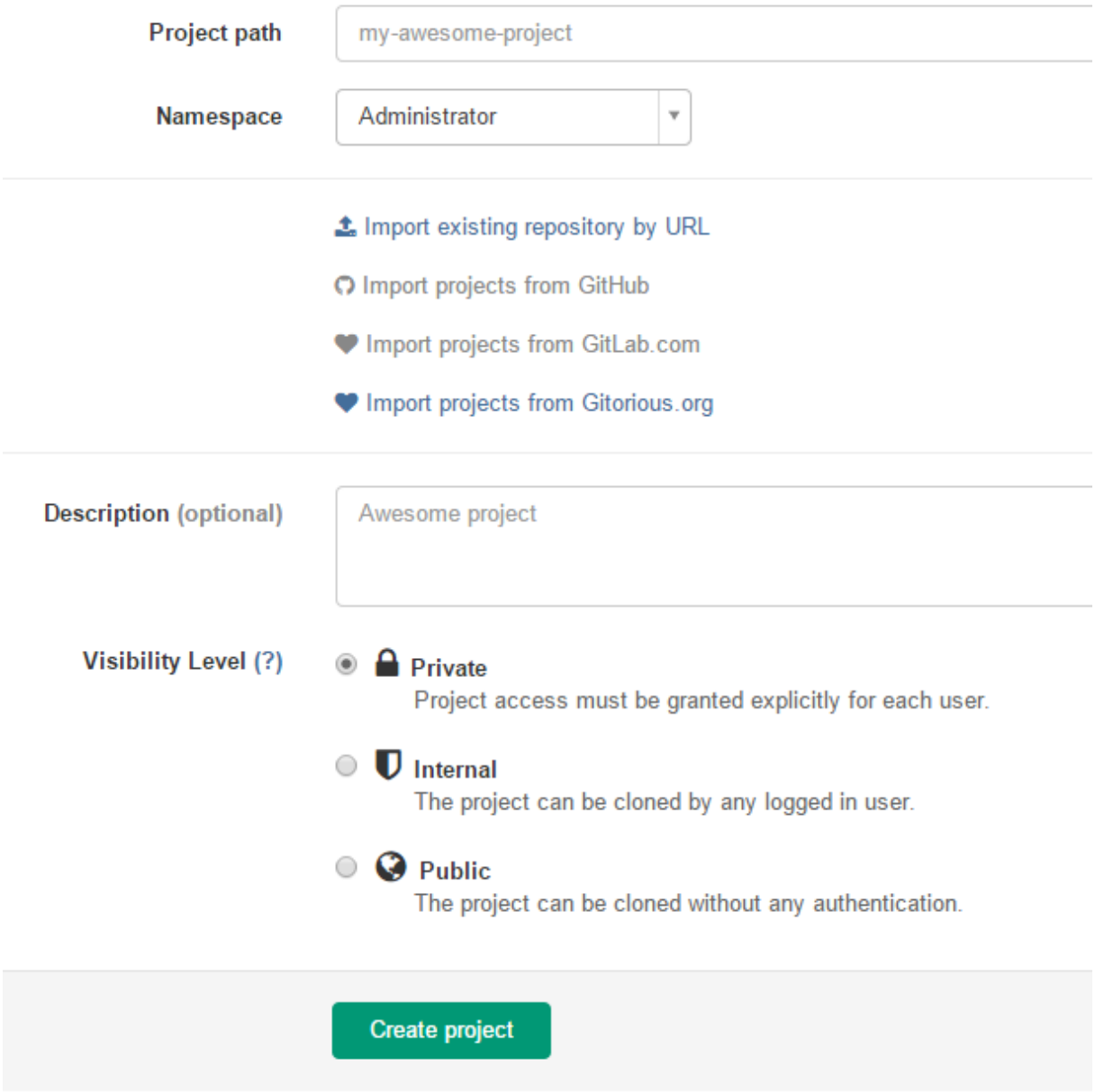
In de bovenste zoekbalk kan gezocht worden naar aangemaakte studenten/docenten. Wanneer er op een van de namen geklikt wordt dan wordt deze aan het lijstje toegevoegd. Vervolgens kan de rol van de studenten gekozen worden. Belangrijk hier is dat de studenten de rol als Developer krijgen en de docenten de rol als Administrator/Owner. Dit vanwege de beveiligingsrechten voor het ophalen van andermans projecten. De administrator kan namelijk alle projecten ophalen, de developers alleen de master repository en hun eigen forks.

Aanmaken projecten

Het aanmaken van de projecten is een van de belangrijkste onderdelen van het versiebeheersysteem. Om de projecten aan te maken moet men weer eerst naar het admin paneel gaan door middel van de onderstaande knop.



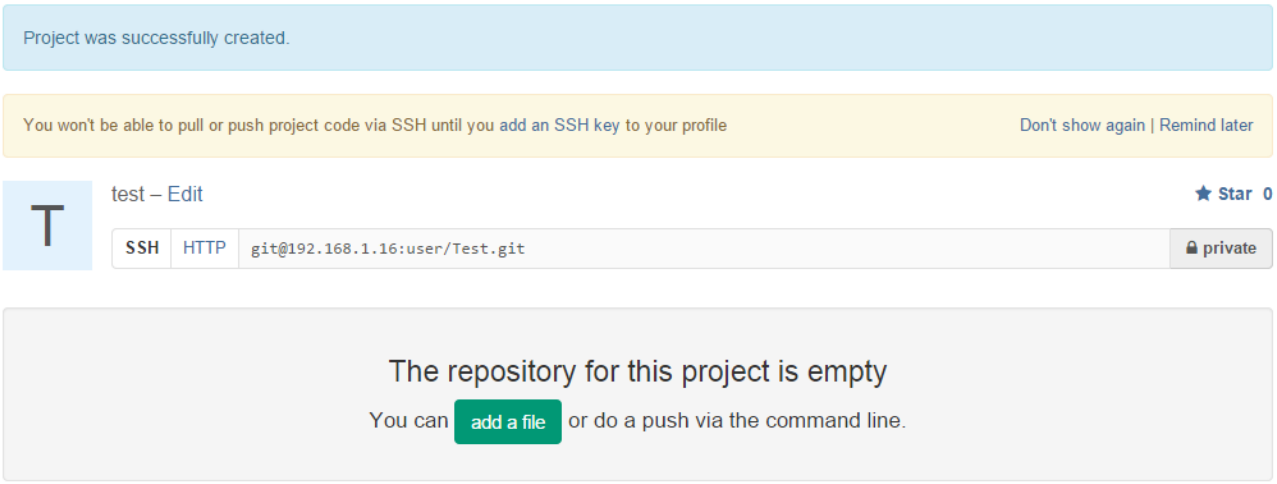
Vervolgens dient er aan de linkerkant van de pagina op de “Projects” knop gedrukt te worden. Dit zorgt er voor dat het project overview scherm getoond wordt. (De lijst met projecten kan heel groot zijn, dit vanwege het feit dat ook forked projecten in de lijst terug te vinden zijn.) In dit overzicht kan er onder andere gezocht worden op projectnaam en groepen. Helemaal rechts van boven op de pagina staat een knop met “New Project”. Klik hierop voor het aanmaken van een nieuw project.



Wanneer op de knop is gedrukt voor het aanmaken van een nieuw project, dan is het bovenstaande formulier te zien. Hierin kunnen de details van het project ingevuld worden. Het project path is de

naam van het project en vormt een belangrijk aspect van de link naar het project. De namespace bepaald de toegang tot het project, in het bovenstaande voorbeeld staat alleen de administrator als namespace, niemand anders dan hem heeft dan toegang. Als namespace kan ook een groep worden gekozen, bijvoorbeeld I11R. Op deze manier worden alle leden van de groep toegevoegd aan het project. Er zijn tevens nog opties voor het importeren van project van bijvoorbeeld GitHub. Hiervoor is een connectie nodig met het desbetreffende platform. Mocht dit gewenst zijn dan kan hier meer over gevonden worden bij de account pagina van de gebruiker. Het derde punt is het commentaar dat toegevoegd kan worden, dit kan een korte beschrijving bevatten van het project. Als laatste kan de visibility aangepast worden, het best kan deze op private worden gelaten. Hierdoor heeft alleen de groep mensen die expliciet aangegeven staat toegang tot het project.

Na het aanmaken van het project komt men op het dashboard van het project. Hierin staat globaal wat informatie over het zojuist aangemaakte project, zoals de naam en de status. Tevens is de SSH alsmede http link gegeven voor het project. Door middel van deze link kunnen er bestanden van het project gepullt worden en naar het project gepusht worden. Ook is er een mogelijkheid om een file aan te maken. Deze file kan bijvoorbeeld gebruikt worden voor de opdrachtomschrijving. Deze kunnen de studenten vervolgens uit het project halen. Het aanmaken van de file spreekt voor zichzelf, de instructies staan op het scherm.



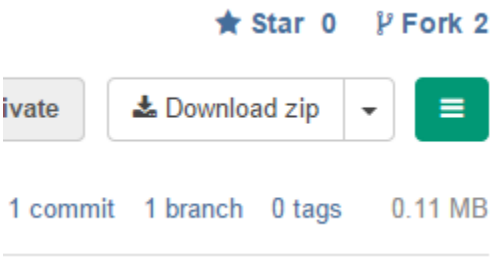
Bovenaan het scherm valt ook te lezen dat de mogelijkheid tot pull en push met SSH niet mogelijk is. Dit komt omdat er geen SSH key gekoppeld is aan het account. Dit is een mogelijkheid om te doen voor elk account, echter doet de applicatie het ook goed met http.

Project forking (student account)

De volgende uitleg gaat over het forken van het project in een eigen repository van de student. Tot deze repository hebben alleen de student en de administrator toegang. Overige studenten in dezelfde groep kunnen de uitwerkingen niet inzien. Voor het inloggen van de student dient de administrator eerst uit te loggen. Dit kan door de uiterst rechtse knop van boven in het menu (waar ook de admin knop staat). Vervolgens dient er in te loggen met een account van een student, hiervoor kunnen de volgende inlog gegevens gebruikt worden:

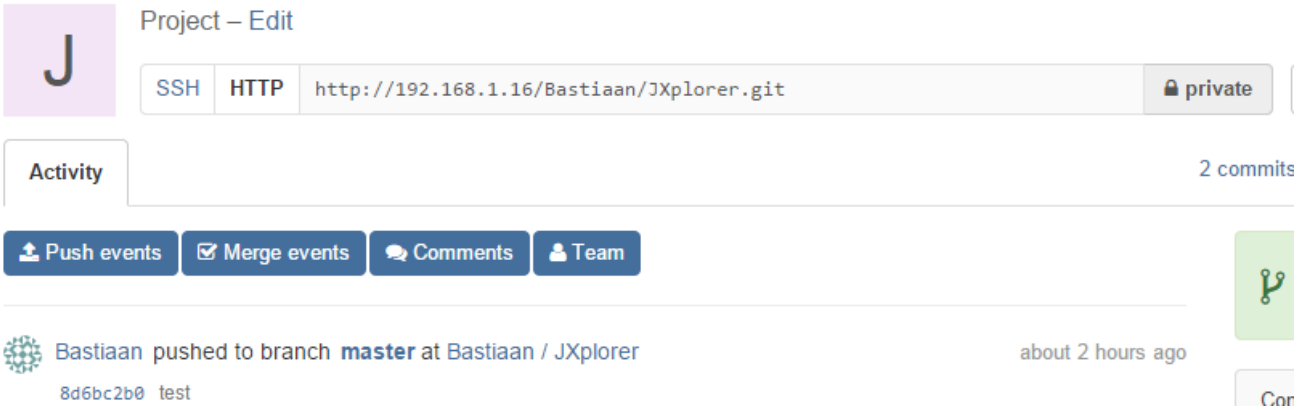
Username: Bastiaan
Wachtwoord: testtest

Wanneer er is ingelogd is direct het dashboard te zien van de student (zie ook direct dat er minder opties zijn). Als de student zijn betrokken projecten in wilt zien, dient er in het linker menu op “Projects” gedrukt te worden. Hierdoor komt de lijst met projecten naar voren. Als er vervolgens op een project gedrukt wordt is het dashboard te zien van het project. Deze komt vrijwel overeen als die van de administrator. Aan de rechterkant van het scherm zijn een aantal opties te kiezen:



Klik hierbij op de “Fork” tekst voor het maken van een forked repository. Als er nog geen forked repository is voor het project is er een grote knop te zien met de naam van de student erop. Als hierop gedrukt wordt dan wordt er een forked repository aangemaakt. Dit is een exacte clone van de master respository.

Wanneer de forked repository is aangemaakt is het volgende scherm te zien:

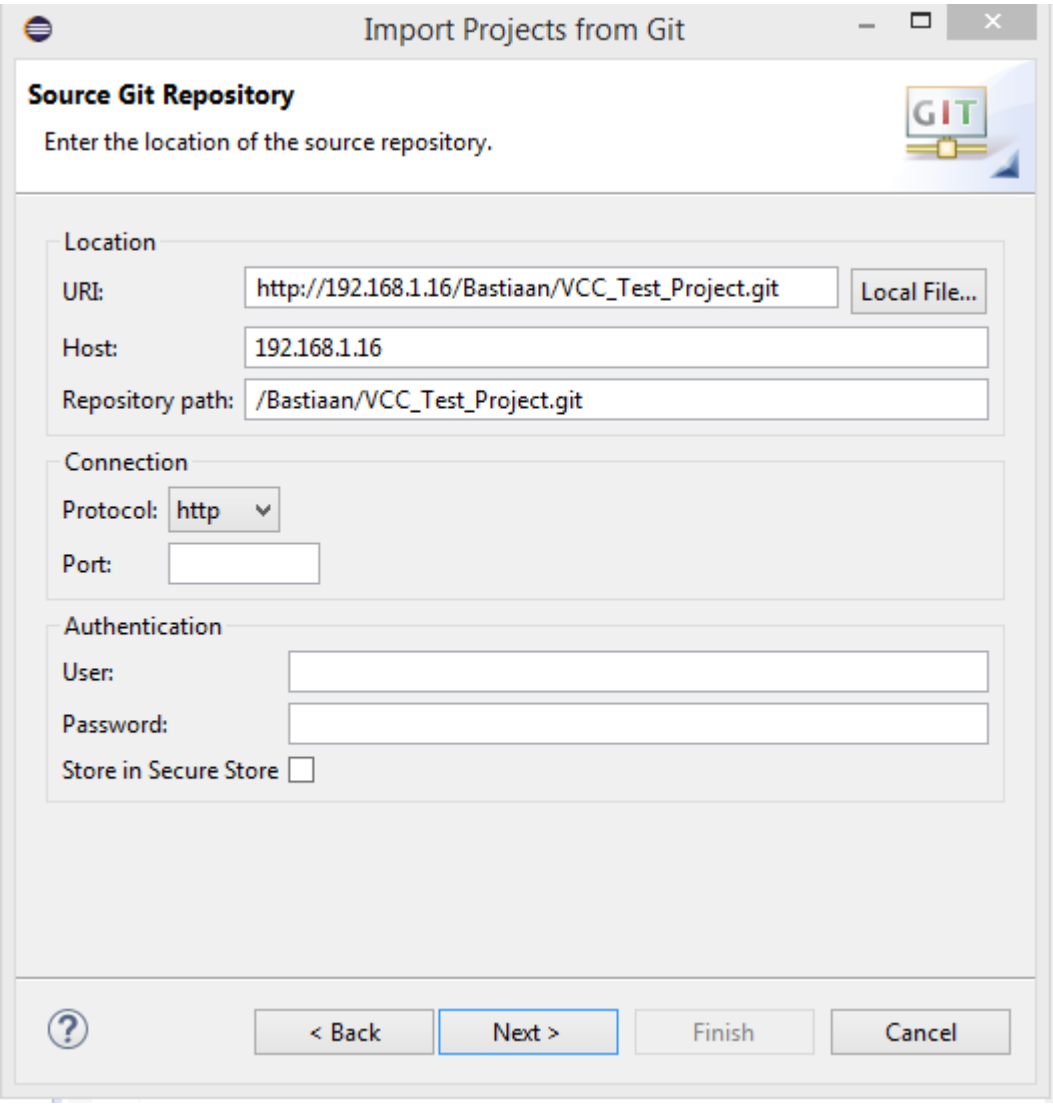


Doordat dit een andere repository is, heeft deze ook een andere link om projecten te pushen en pullen. Deze link, in de bovenstaande afbeelding, kan bijvoorbeeld gebruikt worden in Eclipse. Hierover in het komende onderdeel meer uitleg.

Pull door middel van Eclipse

Om gebruik te maken van het GIT versiebeheer binnen Eclipse, dient er eerst de EGit plugin voor Eclipse gedownload te worden. Door middel van deze plugin kan de gebruiker een connectie maken met een Git repository en via daar zijn projecten delen.

Om een eigen project van de fork te halen, moet men het project importeren. Dit gebeurt door middel van de import optie (rechtermuisklik in de package explorer -> import). In het volgende menu, kies Git -> Projects from Git. Vervolgens wordt er gevraagd of het gaat om een bestaande lokale repository of om een URI. Kies in dit geval de URI. Het volgende scherm komt naar voren:



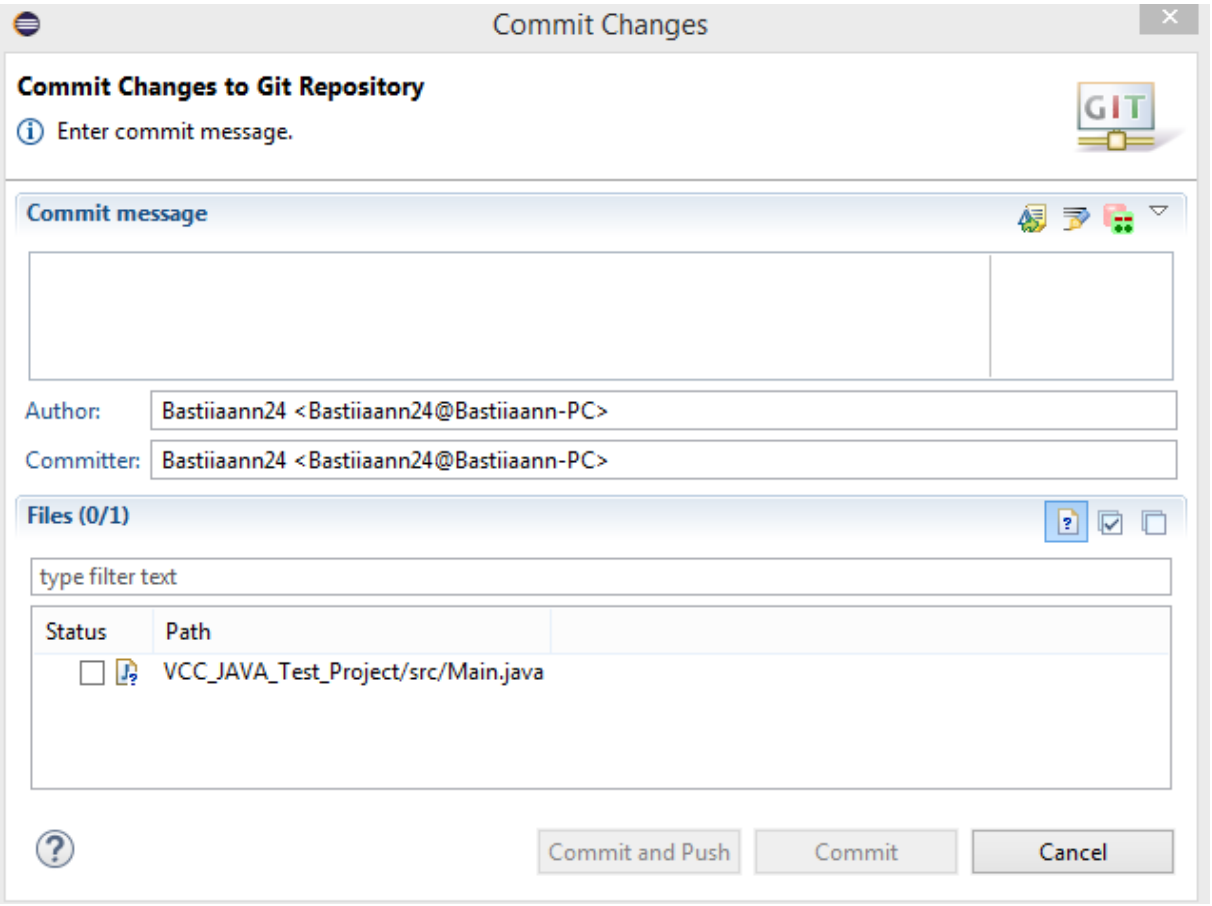
In de Location opties, vul de locatie in van de forked repository. Dit is de http repository die op het dashboard van de forked repository te vinden is. Vul vervolgens ook de host en het repository path in, beide zijn uit de URI te halen. Het connection type moet op http staan als dit gebruikt wordt, ssh is ook een mogelijkheid als het ondersteund wordt door de server. Bij authentication worden de gegevens van de gebruiker ingevuld. Dit zijn dezelfde inlog gegevens als die van het dashboard.

Als alles is ingevuld en erop next wordt gedrukt, dan wordt er gevraagd welke branch van de repository gepullt moet worden. In ons geval is er alleen een master branch aanwezig, selecteer deze. Vervolgens wordt er gevraagd waar het project opgeslagen moet worden op de lokale schijf, kies hiervoor een geschikte plaats en druk op next. De rest van de stappen gaan over het importeren van het project en wat voor instellingen hieraan verbonden zijn. Wanneer al deze stappen doorlopen zijn en op finish is gedrukt, dan is het project te vinden in de package explorer van Eclipse.

Push door middel van Eclipse

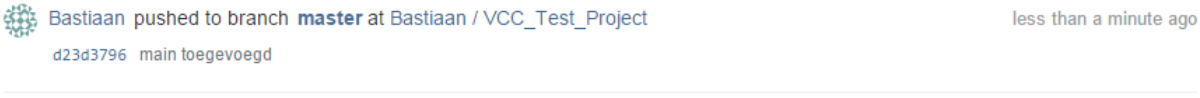
Het laatste onderdeel van deze handleiding is het pushen door middel van Eclipse. Zojuist is een project al van de Git server afgehaald en klaar gemaakt voor gebruik. Wanneer er aan het project gewerkt wordt en een versie opgeleverd moet worden, dan is het wel noodzakelijk dat er ook gepusht kan worden naar de server.

Voor het pushen zijn twee mogelijkheden, namelijk pushen naar de lokale repository en pushen naar de Git server. Beide gaan via dezelfde methode in Eclipse. Wanneer er een commit geplaatst moet worden, drukt men met rechtermuisklik op het project, Team -> Commit. Dit opent het commit scherm:



In het bovenste vak kan het bericht van de commit worden ingevuld, dit is een informatieve tekst over wat er precies is aangepast. Vervolgens klikt men in de Files tab alle bestanden aan die gecommit dienen te worden. Wanneer alles goed is krijgt men de mogelijkheid om te committen of te committen en pushen. Kort gezegd zorgt commit alleen er voor dat de aanpassingen naar de lokale repository worden gestuurd. De commit en push zorgt er dus voor dat de lokale repository gesynchroniseerd wordt met de Git server. Kies in dit geval voor de Commit en Push.

Om te pushen moeten de inlog gegevens van de gebruiker ingevuld te worden. Vul deze in en Eclipse start met het pushen van de aanpassingen naar de Git server. Als alles is goed gegaan komt er een scherm naar voren met daarin de resultaten. In het Gitlab dashboard is vervolgens deze push ook te zien:


Bastiaan pushed to branch master at Bastiaan / VCC_Test_Project
less than a minute ago
d23d3796 main toegevoegd