Best practices om continuous integration en continuous delivery uit te rollen in een SAPUI5 webapplicatie op SAP Cloud Platform

Onderzoeksvoorstel Bachelorproef 2018-2019

Renaat Haleydt[1]

Samenvatting
Met deze studie is het doel de best practices uit te schrijven om een Continuous Integration / Continuous Delivery pipeline te implementeren in een SAPUI5 webapplicatie op SAP Cloud Platform. De applicatie maakt gebruik van SAP HANA, NodeJS en HTML5. Er bestaan verschillende tools om een CI/CD pipeline te maken die elk voor- en nadelen hebben. Amista beseft dat er nood is om wijzigingen sneller te implementeren en CI/CD kan daar een oplossing voor bieden. Het is belangrijk dat dit onderzocht wordt, omdat dit binnen SAP een nieuw concept is. Er is geen specifieke handleiding voor de technologieën die ze bij Amista gebruiken. Een voorbeeldapplicatie - die de technologieën (SAP HANA, NodeJS en HTML5) gebruikt - zal helpen om in deze studie de best practices te onderzoeken die Amista op weg te helpen om de meest geschikte CI/CD pipeline te implementeren. De verschillende tools worden kort toegelicht en de voor- en nadelen van de verschillende tools worden besproken. Deze studie is de gemiste koppeling tussen de theorie die je op internet vindt en een praktijkvoorbeeld van een SAPUI5 webapplicatie op een SAP Cloud Platform. De conclusie is dat Amista er meer voordelen dan nadelen uithaalt wanneer ze investeert in een CI/CD pipeline en dat er gebruik zal gemaakt worden van Git, Jenkins en Nexus.

Sleutelwoorden
Onderzoeksdomein. Continuous Integration — Continuous Delivery — SAP Cloud Platform — Webapplicatieontwikkeling

Co-promotor
Pieter-Jan Deraedt[2] (Amista)

Contact: [1] renaat.haleydt.y0066@student.hogent.be; [2] pieter-jan.deraedt@amista.be;

Inhoudsopgave

1. Introductie

Betere, redundante projecten opleveren, dit is waar ieder bedrijf naar streeft. Amista is een consultancy bedrijf dat op zoek gaat, samen met hun klanten, naar oplossingen binnen de wereld van SAP. Bij Amista weten ze heel goed waar de noden van hun klanten liggen en zouden hier dan ook graag op inspelen. Klanten willen namelijk de gewenste veranderingen onmiddellijk te zien krijgen, de wachttijd op een oplevering die ze willen doorvoeren moet minimaal blijven. Meestal werkt een heel team aan de oplevering van een project, dit leidt tot verschillende versies van de code. Men weet niet precies op welk punt de code 'echt werkt'. Continuous Integration en Continuous Delivery kan een hulpmiddel zijn om bij elke push werkende software te hebben. Er wordt namelijk gecontroleerd of de code voldoet aan criteria om het als 'werkende' te beschouwen (de code wordt groen verklaard).

Vandaag de dag verwachten klanten dat het programma blijft werken eens een oplevering doorgevoerd wordt (dit wordt vaak 0 downtime genoemd in het vakjargon). Een manier om deze nood op te vangen is het implementeren van Continuous Deployment.

Deze studie biedt een weg aan bedrijven die denken om een CI/CD pipeline te integreren op SAP Cloud Platform om te slagen in hun opdracht. Het gaat specifiek over SAPUI5 webapplicaties die gebruik maken van SAP HANA, NodeJS en HTML5. De best practices om zo een systeem uit te rollen worden besproken, alsook wat de voor- en nadelen zijn van bepaalde frameworks/tools.

Onderzoeksvragen:

- Wat zijn de voor- en nadelen van een CI/CD pipeline te integreren?
- Is het mogelijk om op een eenvoudige manier een Continuous Integration en Continuous Delivery pipeline te implementeren voor de ontwikkelingen van een SAPUI5 applicatie op SAP Cloud Platform?
- Hoe kunnen we deze implementatie tot een succes

brengen?
- Welke tools moeten we gebruiken om een CI/CD pipeline op SAP Cloud Platform te implementeren en wat zijn de voor- en nadelen hierbij?

2. State-of-the-art

SAP Cloud Platform

SAP Cloud Platform is een Platform-as-a-service (PaaS), een platform dat - door hardware en software samen te brengen - applicaties overal toegankelijk maakt en samenbrengt tot 1 platform (IT, 2018). SAP Cloud Platform is een development en deployment platform dat ook de hand reikt aan verschillende technologieën: Internet of Things, big data, Artificial Intelligence enzovoort. Het is een platform dat zowel on-premise als cloud technologieën samen kan brengen, die je kan uitbreiden en zelf kan ontwikkelen. Het haalt zijn kracht vooral uit de perfecte integratie van andere SAP software die je ook nog eens kan uitbreiden.

Continuous Integration Dit is een manier om software te maken waar de focus ligt bij de teamleden (Fowler, 2006). Zij worden verwacht hun geschreven code op regelmatige basis te integreren met de master applicatie. Op die manier gaat de code door een molen die een geautomatiseerde build zal uitvoeren en kijkt of de code slaagt voor Unit tests. Een best practice is om te testen in een kopie van de productieomgeving, zo heb je weinig risico op ongelukken. Deze techniek van implementeren brengt bepaalde voordelen met zich mee: er zijn minder problemen om de code te integreren op de master applicatie waardoor men sneller kan voldoen aan de eisen van de klant.

Continuous Delivery De code wordt getest de testen slagen wordt de code afgeleverd in een formaat dat klaar is om te deployen. Bij deze stap is er een menselijke keuze nodig om de software naar de klant of gebruiker te brengen (Fowler, 2013).

Continuous Deployment

Dit is een manier om software uit te sturen en als klaar te beschouwen. Het is een mogelijkheid om alle soorten wijzigingen - inclusief nieuwe functies, configuratiewijzigingen, bugfixes en experimenten - op een veilige, snelle en automatische manier bij de klant of gebruiker te brengen. Deze manier van deployen is makkelijker om te voldoen aan de wijzigingen die moeten doorgevoerd worden. Er komt geen grote release bij te pas waar iedereen bang afwacht of de update stand houdt. Door telkens kleine veranderingen op een veilige manier door te voeren zorgt men ervoor dat de applicatie veel minder kans heeft op falen (Claps, Svensson & Aurum, 2015). Het is aan te raden dat wanneer je Continuous Deployment wil implementeren in je project, je eerst Continuous Integration en Continuous Delivery op punt moet zetten. Ook is het belangrijk om een goede flow van versie control te hebben binnen het team en je moet gebruik maken van een automated deploy script die je aan de build hangt .

CI/CD integreren op SAP Cloud Platform

SAP Cloud Platform biedt de mogelijkheid om verschillende omgevingen op te stellen waarin je kan werken als developer. Het vergt enige vereisten om te voldoen aan de regels van Continuous Integration (Kramer, 2018):

- Hou alles goed bij via een version control systeem
- Automatiseer de build
- Zorg ervoor dat tijdens de build er Unit testen lopen
- Het team moet op regelmatige basis commits uitvoeren
- Elke verandering moet gebuild worden
- Als er errors tevoorschijn komen tijdens de build moeten die opgelost worden
- De build moet uitgetest worden op een kopie van de productieomgeving
- Automatiseer de deployment

Eens deze regels toegepast zijn kunnen we spreken van een CI implementatie. Vaak wordt CI in combinatie gebracht met Continuous Delivery. Om dit in een vloeiende lijn te laten gaan spreekt men van een CI/CD pipeline.

CI/CD pipeline volgens SAP

Een programmeur schrijft nieuwe code voor een verandering die de klant wil uitvoeren. Idealiter zou dit - voor het mergen naar de masterapplication - eens door een voter build moeten gaan, waar automatische test aanwezig zijn die kijken of de code geen problemen zou geven als je die zou mergen met de master. Een laaste stap voor de code naar de master gemerged wordt, is het toepassen van code reviews door collega developers (het 4-ogen principe). Na het samenvoegen wordt automatisch de CI-build geactiveerd. De code gaat door de automatische tests. Eens de testen slagen worden de wijzigingen geïntegreerd op de master.

Dan komt de Continuous Delivery fase, waar-

bij de code nog eens door een test systeem gaat. Deze fase gebeurt allemaal automatisch, maar er kunnen ook manueel testen uitgevoerd worden. Eens de code door deze fase raakt is ze klaar om te deployen. Bij Continuous Deployment worden de wijzigingen dus automatisch naar buiten gebracht (Kramer, 2018).

Tools die gebruikt kunnen worden om een CI/CD pipeline te implementeren

Er bestaan verschillende source code repositories waar je de versies van je code kan beheren. GitHub, Git, GitLab, Bitbucket zijn voorbeelden van zo een tools. Build schedulers zorgen ervoor dat de procedures worden samengesteld en dat de builds worden getriggerd. Voorbeelden hiervan zijn: Jenkins, Travis CI, GitLab-CI en Bamboo. Sonatype Nexus en Archiva zijn voorbeelden van tools die gebruikt worden als repository manager, deze houden bij wijze van spreken de code bij die klaar is om te deployen.

3. Methodologie

Eerst worden de voor- en nadelen van het integreren van een CI/CD pipeline in een SAPUI5 applicatie uitgeschreven. Deze studie zal de voor- en nadelen van de verschillende tools onderzoeken op vlak van configureerbaarheid met SAP en de tools die Amista gebruikt, documentatie die te vinden is online en de kostprijs. Er wordt een voorbeeldapplicatie gemaakt die zal helpen bij het uitschrijven van de best practices om een CI/CD pipeline uit te werken aan de hand van de tools die gekozen werden. De voorbeeldapplicatie wordt een omgeving waar het mogelijk is om kleine deeltjes code te wijzigen en die dan testen of ze klaar zijn om te deployen.

4. Verwachte resultaten

Uit onderzoek zal blijken dat de nadelen niet zullen opwegen tegen de vele voordelen die een CI/CD pipeline te bieden heeft. Amista maakt reeds gebruik van Git als versiebeheersysteem, dit zal ongetwijfeld doorwegen op de keuze. Ook het feit dat Git open source en dus helemaal gratis te gebruiken is heeft zijn voordelen. GitHub, GitLab en Bitbucket zijn allemaal een online repository management systeem om Git projecten te beheren. GitLab is open source, maar er bestaat een formule waar een onderneming moet betalen voor de diensten en server beheer. Als een onderneming gebruik wil maken van GitHub en Bitbucket zal ze geld op tafel moeten leggen. Wanneer we kijken naar de samenwerking met build schedulers, zien we dat de online repository management systemen hun eigen tool hebben. Zo heeft GitLab een eigen gemaakte Continuous Integration tool, GitLab CI genaamd. Deze tool is gratis te gebruiken tot 2000 minuten per maand (GitLab, 2019). Wanneer er gebruik gemaakt wordt van Bitbucket zal hoogst waarschijnlijk Bamboo gebruiken, deze twee tools komen van hetzelfde bedrijf en werken bijgevolg naadloos samen. Wanneer men GitHub gebruikt is het mogelijk om Travis CI te implementeren. Jenkins heeft dan weer plug-ins ter beschikking waarbij de samenwerking met de bovenstaande online repository management systemen verzekerd is.

Als we kijken naar de build schedulers zien we dat Jenkins een grote community heeft, het brengt geen kosten met zich mee (want het is open source) en biedt vele plug-ins aan om combinatie met andere tools makkelijk te laten verlopen. Sonatype Nexus biedt de mogelijkheid aan om te kiezen tussen de open source versie of de professionele versie die minstens $10/maand kost (Sonatype, 2019). Het verschil zit hem in de support die Sonatype biedt (OBrien, 2010). Apache Archiva is ook open source, maar daar is minder informatie over te vinden. De community is niet zo groot als bij Nexus.

In dit onderzoek wordt er ook een koppeling van de gevonden theorie aan een klein praktijkvoorbeeld gemaakt. Dit aan de hand van best practices om een Continuous Integration/Continuous Delivery pipeline op te zetten en een soort gids om die best practices toe te passen op de omgeving waar Amista mee werkt.

5. Verwachte conclusies

Als onderneming heeft het zeker voordelen om een CI/CD pipeline te integreren. Gaande van onmiddellijke feedback op geschreven code van developers, betere implementatie, 0 downtime. Als nadeel verwacht ik dat kost groot zal zijn. De integratie en onderhoud van zo een infrastructuur brengt heel wat aanpassingen met zich mee die geld kosten. Er moeten ook veel test geschreven worden om een goede pipeline te bekomen. Ik denk dat het mogelijk is dat bedrijven, mits een goede handleiding van best practices bij de hand, een succesvolle CI/CD pipeline zullen integreren.

Volgens mij zullen Git, Jenkins en Nexus gebruikt worden om e pipeline te maken. Rekening houdend met de kosten en de configureerbaarheid met de tools die Amista gebruikt en beschikbare documentatie.

Referenties

Claps, G. G., Svensson, R. B. & Aurum, A. (2015). On the journey to continuous deployment: Technical and social challenges along the way. *Information and software technology*, *57*, 21–31.

Fowler, M. (2006). Continuous Integration. Verkregen van http://www.dccia.ua.es/dccia/inf/asignaturas/MADS/2013-14/lecturas/10_Fowler_Continuous_Integration.pdf

Fowler, M. (2013). Continuous Delivery. Verkregen van https://martinfowler.com/bliki/ContinuousDelivery.html

GitLab. (2019). GitLab pricing. Verkregen van https://about.gitlab.com/pricing/

IT, S. G. (2018). Overview — SAP Cloud Platform. Verkregen van https://cloudplatform.sap.com/index.html

Kramer, W. (2018). Continuous Integration (CI) Best Practices with SAP, CI/CD Practices. Verkregen van https://developers.sap.com/tutorials/ci-best-practices-ci-cd.html

OBrien, T. (2010). Nexus Open Source or Professional: Which One is Right for You? Verkregen van https://blog.sonatype.com/2010/01/nexus-open-source-or-professional-which-one-is-right-for-you/

Sonatype. (2019). Nexus product pricing. Verkregen van https://www.sonatype.com/nexus-product-pricing