



Faculteit Bedrijf en Organisatie

Titel

Kenzie Coddens

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Olivier Rosseel
Co-promotor:
Raf Boon

Instelling: Aucus

Academiejaar: 2019-2020

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Titel

Kenzie Coddens

Scriptie voorgedragen tot het bekomen van de graad van
professionele bachelor in de toegepaste informatica

Promotor:
Olivieer Rosseel
Co-promotor:
Raf Boon

Instelling: Aucxis

Academiejaar: 2019-2020

Tweede examenperiode

Woord vooraf

Samenvatting

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus.

Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Inhoudsopgave

1	Inleiding	15
1.1	Probleemstelling	15
1.2	Onderzoeksvraag	16
1.3	Onderzoeksdoelstelling	16
1.4	Opzet van deze bachelorproef	16
2	Stand van zaken	17
2.0.1	Software release management for component-based software	17
2.0.2	Challenges and Problems in Release Management Process: A Case Study	19
2.0.3	Methodes and Systems for Software Release Management	21
2.0.4	Continuous Integration and Its Tools	23
2.0.5	DeVops	23

2.1	Amazon AWS	24
2.1.1	Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud	24
2.2	Microsoft Azure	25
2.2.1	Microsoft Azure and cloud computing	25
2.2.2	Microsoft Azure Documentation	26
3	Methodologie	29
4	Conclusie	31
A	Onderzoeksvoorstel	33
A.1	Introductie	33
A.2	Literatuurstudie	34
A.2.1	Conventional Software Testing Vs. Cloud Testing	34
A.2.2	Software Testing Based on Cloud Computing	34
A.2.3	Benchmarking in the Cloud: What It Should, Can, and Cannot Be	35
A.2.4	When to Migrate Software Testing to the Cloud?	35
A.3	Methodologie	35
A.4	Verwachte resultaten	36
A.4.1	Vergelijking van platformen	36
A.4.2	Proof of concept	36
A.4.3	beveiliging experiment	36
A.5	Verwachte conclusies	37
A.5.1	Vergelijking van platformen	37
A.5.2	Proof of concept	37

A.5.3	beveiliging experiment	37
-------	------------------------	----

	Bibliografie	39
--	---------------------	-----------

Lijst van figuren

- 2.1 Figuur uit (van der Hoek & Wolf, 2002). Figuur toont een simpele chart van hoe een software release management systeem opgebouwd kan zijn. 19
- 2.2 Figuur uit (Methodes and systems for software release management, 2005). Flowchart van een niet geautomatiseerde versie beheer procedure. 22
- 2.3 Figuur van (<https://p2zk82o7hr3yb6ge7gzxx4ki-wpengine.netdna-ssl.com/wp-content/uploads/azure-vm-types-comparison-1.jpg>). Chart met alle VM types van azure en hun specifieke code. 27

Lijst van tabellen

1. Inleiding

De inleiding moet de lezer net genoeg informatie verschaffen om het onderwerp te begrijpen en in te zien waarom de onderzoeksvraag de moeite waard is om te onderzoeken. In de inleiding ga je literatuurverwijzingen beperken, zodat de tekst vlot leesbaar blijft. Je kan de inleiding verder onderverdelen in secties als dit de tekst verduidelijkt. Zaken die aan bod kunnen komen in de inleiding (**Polleffiet2011**):

- context, achtergrond
- afbakenen van het onderwerp
- verantwoording van het onderwerp, methodologie
- probleemstelling
- onderzoeksdoelstelling
- onderzoeksvraag
- ...

1.1 Probleemstelling

Uit je probleemstelling moet duidelijk zijn dat je onderzoek een meerwaarde heeft voor een concrete doelgroep. De doelgroep moet goed gedefinieerd en afgeleid zijn. Doelgroepen als “bedrijven,” “KMO’s,” systeembeheerders, enz. zijn nog te vaag. Als je een lijstje kan maken van de personen/organisaties die een meerwaarde zullen vinden in deze bachelorproef (dit is eigenlijk je steekproefkader), dan is dat een indicatie dat de doelgroep goed gedefinieerd is. Dit kan een enkel bedrijf zijn of zelfs één persoon (je co-promotor/opdrachtgever).

1.2 Onderzoeksvraag

Wees zo concreet mogelijk bij het formuleren van je onderzoeksvraag. Een onderzoeksvraag is trouwens iets waar nog niemand op dit moment een antwoord heeft (voor zover je kan nagaan). Het opzoeken van bestaande informatie (bv. “welke tools bestaan er voor deze toepassing?”) is dus geen onderzoeksvraag. Je kan de onderzoeksvraag verder specificeren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

1.3 Onderzoeksdoelstelling

Wat is het beoogde resultaat van je bachelorproef? Wat zijn de criteria voor succes? Beschrijf die zo concreet mogelijk. Gaat het bv. om een proof-of-concept, een prototype, een verslag met aanbevelingen, een vergelijkende studie, enz.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

2.0.1 Software release management for component-based software

Een van de eerste vragen die opdook was: “Hoe wordt software release management nu eigenlijk verwezenlijkt?”. Zijn er bepaalde procedures of praktijken die gevolgd worden om dit in goed banen te lijden. Welke technieken worden er gevolgd? Dit leidde ertoe om in de eerste plaats in de richting van de ITIL-processen te zoeken (ITIL, Information Technology Infrastructure Library). Dit omdat het nauw aansluit bij software release management. De volgende paper (van der Hoek & Wolf, 2002) is een resultaat van de papers bij de zoek term ‘ITIL’.

De paper (van der Hoek & Wolf, 2002) is een rapport, verslag over een jarenlange observatie van software release management praktijken. Deze paper (van der Hoek & Wolf, 2002) is niet meer van de jongste maar bespreekt toch nog een aantal belangrijke kernideeën. De technische kant en de gebruikte software is minder relevant.

De paper (van der Hoek & Wolf, 2002) begint met de problematiek uit te leggen van software release management voor zowel de gebruiker als de ontwikkelaar. In de eerste plaats stelt de paper dat de eindgebruiker altijd het slachtoffer is. Tegenwoordig wordt er veel component gebaseerde software ontwikkelt. Een goed voorbeeld hiervan, zijn Linux packages. Dit maakt het niet altijd gemakkelijk voor de eindgebruiker om de juiste softwarecomponenten te vinden. Laat staan dat het de juiste versies zijn. Voeg daar dan nog aan toe dat sommige softwarebedrijven niet altijd oudere versies aanbieden, dat de eindgebruiker meerdere websites moet afsporen en dat de eindgebruiker ook nog eens verantwoordelijk is voor het installeren en up-to-date houden van al die componenten. Dit is dus ver van een optimale situatie.

Het samenvoegen van verschillende componenten, het uitrollen naar de gebruikers en het updaten ervan, wordt beschreven als software release management. Software release management is enkel verantwoordelijk voor het beheren en het opslaan van de verschillende benodigde componenten en is dus niet bedoelt om de verschillende componenten zelf te compileren of afleiden. Dit zorgt ervoor dat software release management tools compleet platform onafhankelijk kunnen zijn. Het limiteert daardoor wel de functionaliteit. In het bijzonder is de tool niet instaat om aan validatie of versie beheer te doen.

Om aan goede software release management te kunnen doen, is goede documentatie van alle verschillende componenten nodig. De paper stelt een aantal vereisten voor software release management voor. Dit is voor zowel de eindgebruiker als de ontwikkelaar. Deze hebben ze samengesteld uit hun eigen tien jaar lange ervaring. (van der Hoek & Wolf, 2002).

Minimale vereisten voor ontwikkelaars:

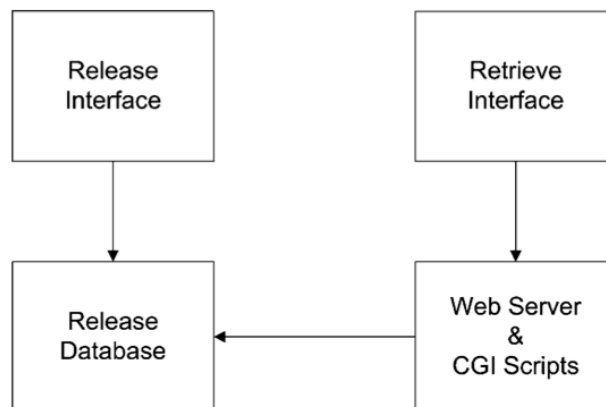
- Afhankelijkheden moeten expliciet zijn en gemakkelijk kunnen worden vastgelegd.
- Releases moeten consistent worden gehouden.
- De reikwijdte van een release moet controleerbaar zijn.
- Het releaseproces zou minimale inspanning aan de kant van de ontwikkelaar moeten inhouden.
- Er moet een geschiedenis van opvragen worden bijgehouden.

Minimale vereisten voor eindgebruiker:

- Beschrijvende informatie moet beschikbaar zijn.
- Er moet transparantie over de locatie worden geboden.
- Een component en zijn afhankelijkheden moeten als één archief kunnen worden opgehaald.
- Software-implementatie hulpmiddelen moeten de software-releasebeheertool kunnen gebruiken als bron voor componenten die moeten worden geïnstalleerd en geconfigureerd.

Hierna geeft de paper (van der Hoek & Wolf, 2002) verder uitleg over een software release management tool die de schrijvers van de paper zelf hebben ontwikkeld. Dit is minder interessant voor de doeleinden van dit onderzoek maar de ideeën achter deze tool kunnen een meerwaarde bieden bij de motivatie waarom software release management zo belangrijk is.

De bedoeling van deze tool is om enerzijds de informatie die gebruikt wordt in het releasebeheer proces te structureren en anderzijds locatietransparantie aan te bieden.



Figuur 2.1: Figuur uit (van der Hoek & Wolf, 2002). Figuur toont een simpele chart van hoe een software release management systeem opgebouwd kan zijn.

De figuur 2.1 illustreert de architectuur van de software release management tool. Deze bestaat uit vier delen. Een logisch gecentraliseerde, maar fysiek gedistribueerde releasedatabase, een interface waarmee ontwikkelaars componenten in de releasedatabase plaatsen, een interface waarmee gebruikers componenten uit de releasedatabase halen en een web-server voor het op afstand toegang krijgen tot de releasedatabase en de componenten.

Deze paper (van der Hoek & Wolf, 2002) stelt dus dat software release management ervoor moet zorgen dat componenten die op verschillende locaties, door verschillende bedrijven ontwikkeld worden, gemakkelijk gecentraliseerd toegankelijk moeten zijn. De bedrijven zelf hebben nog altijd volledige controle in handen van het versie beheer en het groeperen van de verschillende extern benodigde componenten. Tevens is de eindgebruiker nog altijd verantwoordelijk voor de installatie ervan maar dit begint ook minder het geval te zijn aangezien er meer en meer uitrol tools op de markt komen.

2.0.2 Challenges and Problems in Release Management Process: A Case Study

Zoals eerder aangehaald spelen ITIL-achtige procedures een belangrijke rol in software release management. Deze paper (Lahtela & Jantti, 2011) is een korte casestudie met de bedoeling om kort een aantal problemen aan het licht te stellen in verband met software release binnen een bedrijf of organisatie naar een klant toe en hiervoor een oplossing aan te bieden.

De casestudie (Lahtela & Jantti, 2011) beschrijft allereerst een definitie voor release management. “Release management omvat mensen, functies, systemen en activiteiten

om software- en hardware versies effectief te plannen, verpakken, bouwen, testen en implementeren in een productie omgeving.” Lahtela en Jantti (2011) Deze definitie sluit goed aan bij wat we in dit onderzoek trachten te verduidelijken. Ook stelt de studie een redelijk belangrijk algemeen probleem. Helaas is in de praktijk bij veel bedrijven nog geen sprake van een implementatie van ITIL, ISO, enz. procedures. Dit omdat dit zeer moeilijk te implementeren is in reeds bestaande processen. Het hebben van zo een procedures is niet alleen zeer belangrijk om kwalitatief goede software op te leveren maar ook voor de supportafdeling van een bedrijf. Meestal zijn dit de mensen die het meeste te maken krijgen met deze procedures. Hierbij moet wel gezorgd worden dat er duidelijk verschil gemaakt wordt tussen het managen van veranderingen en het release management. Vervolgens beschrijft de studie de vastgestelde problemen en een mogelijk oplossing ervoor. Deze zijn letterlijk overgenomen uit de studie Lahtela en Jantti (2011).

- Er is geen gespecificeerd releasebeheerproces.
 - Het releasebeheerproces moet worden beschreven en gestroomlijnd om ervoor te zorgen dat iedereen in de organisatie het proces kent.
- De rol van releasemanager is onduidelijk.
 - Iemand moet worden genoemd voor de rol van releasemanager. Daarnaast moeten de rollen, toewijzingen en verantwoordelijkheden worden beschreven.
- De klant weet niet wat de release bevat.
 - Meestal worden niet alle aangebrachte veranderingen beschreven of worden deze te technische beschreven waardoor de klant deze niet verstaat. De boodschap is om deze goed te documenteren op een verstaanbare manier.
- De uitgifte distributie snelheid is te hoog.
 - De release-vensters, die het tijdstip bepalen waarop de release in de productie-omgeving van de klant moet worden geïnstalleerd, moeten worden overeengekomen tussen de serviceprovider en de klant, bijvoorbeeld grote releases maandelijks en kleinere releases wekelijks.
- De klant denkt dat de serviceprovider niet alle testgevallen kan testen of inspecteren.
 - Het testproces, dat deels wordt gedaan binnen het release managementproces, moet worden ontwikkeld. Het proces moet worden beschreven en aan product-testers worden geleerd. Daarnaast moeten de testresultaten aan de klant worden voorgelegd.
- Er moeten meer testomgevingen in het testproces zijn.
 - Er moet worden onderzocht of er meer testomgevingen nodig zijn. De ideale situatie is dat er voor elke productieomgeving een identieke testomgeving zou zijn.
- Het verandermanagement van testomgevingen is onvoldoende.
 - Elke wijziging die in een bepaalde testomgeving wordt aangebracht, moet correct worden gedocumenteerd. Op deze manier zijn alle wijzigingen traceerbaar en up-to-date.
- Problemen bij versiebeheer.
 - Alle versies van verschillende producten moeten worden gedocumenteerd in een klant specifieke lijst, die de serviceprovider vertelt welke geïnstalleerde productiesversies de klant heeft.
- De caseorganisatie heeft geen specifieke 'release jury'.

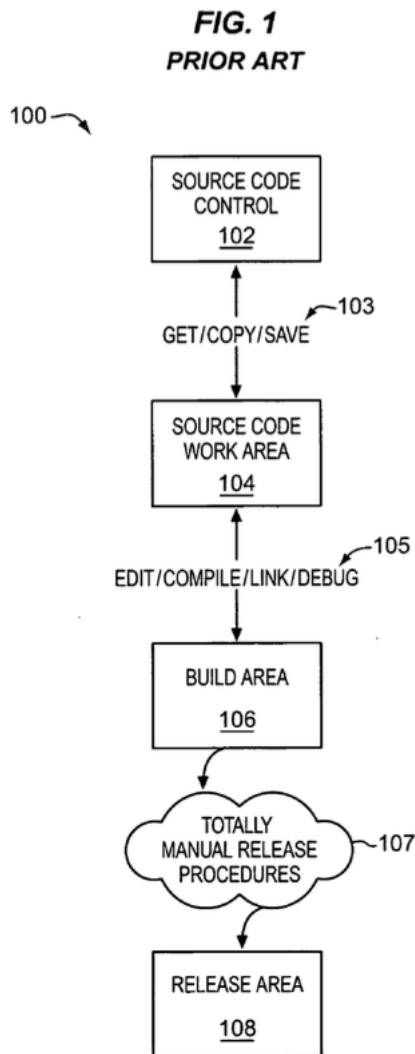
-
- Er is behoefte aan een specifieke jury die releases inspecteert voordat ze in productie worden genomen.

Deze studie (Lahtela & Jantti, 2011) biedt een unieke inkijk op een specifiek geval en biedt oplossingen aan voor bepaalde problemen. Sommige van deze problemen sluiten zeer goed aan bij dit onderzoek. Zo is er nood aan een zeer goede test omgeving zodanig dat er niet nodeloos over en weer moet gereden worden tussen klant en bedrijf. Dit onderzoek zal dan ook deze problematieken in acht nemen en proberen op te lossen in deze use case.

2.0.3 Methodes and Systems for Software Release Management

Het volgende document (Methodes and systems for software release management, 2005) is een patent dat een bepaalde problematiek met normale software release management probeert te verhelpen. Op zich is dit patent niet zo interessant voor dit onderzoek maar de figuren en ideeën waarvoor het bedrijf in kwestie een patent heeft aangevraagd, kunnen wel helpen in het beter begrijpen van software release management en hoe deze wordt toegepast.

Dit document (Methodes and systems for software release management, 2005) begint met zeer algemeen uit te leggen hoe software release management werkt en wat de verschillende stappen zijn die doorlopen worden. Het begint met versie beheer van de software. Waarna deze in een ontwikkelomgeving wordt gebracht. Hierna zal de software meestal naar een test omgeving gaan vooraleer het in productie wordt geplaatst. Al deze verschillende stappen zijn meestal verschillende mensen, in verschillende grote teams, die niets anders dat hun specifieke stap uitvoeren. Zie flowchart 2.2.



Figuur 2.2: Figuur uit (Methodes and systems for software release management, 2005). Flowchart van een niet geautomatiseerde versie beheer procedure.

Het document (Methodes and systems for software release management, 2005) beschrijft echter een aantal problematieken bij het idee uit figuur 1. Zo is dit een zeer manueel proces waarbij verschillende mensen deelnemen. Het is dus zeer gemakkelijk om fouten te maken tijdens een van deze stappen. Bijvoorbeeld een merge conflict. Deze hebben ze proberen te verhelpen door middel van computer geassisteerde release. Zie figuur 2. Ook is er in dit document nagedacht over de verschillende files en methodes om software te compileren en hoe deze beheerd moeten worden. Zo stellen ze gescheiden opslagplaatsen voor, voor inventory files en build files, enz. Zie figuur 3 voor details.

Dit document (Methodes and systems for software release management, 2005) toont dus een nogal algemeen beeld over de software release cycli en wordt meer ter info beschouwd in dit onderzoek.

2.0.4 Continuous Integration and Its Tools

Bij release management hoort continuous integration en continuous development. Het volgende korte artikel (Meyer, 2014) bespreekt kort waarom een organisatie aan continuous integration moet doen en wat voor tools er allemaal bestaan. Ook worden er een aantal aanwijzingen gegeven rond het gebruik van continuous integration.

Het artikel (Meyer, 2014) vertrekt vanuit het standpunt dat een versie beheer tool in ieder softwarebedrijf een gegeven moet zijn. Dit om situaties te vermijden dat het lokaal werkt maar op andere toestellen niet. Ook staat dit toe om volledige geautomatiseerde pipelines te maken die automatisch de code gaat compileren en controleren op fouten door meegeleverde testen of door een uitrol te doen, in een test omgeving. Het artikel gaat verder met te stellen dat het bij iedere organisatie een prioriteit moet zijn om ten allertijden de builds werkende te houden. Dit staat dan niet alleen toe dat er ten allertijden een uitrol kan gedaan worden van werkende code naar een test omgeving, maar ook dat er geen fouten of niet werkende code wordt gepusht door de ontwikkelaars. Het artikel beschrijft verder een aantal tools zoals onder andere Jenkins voor het automatisch uitrollen en testen van nieuwe software.

Dit artikel (Meyer, 2014) is minder belangrijk voor dit onderzoek maar toont wel het belang aan van een build pipeline en een goeie test omgeving voor code werkende te houden en de kwaliteit te behouden.

2.0.5 DeVops

Devops is de overkoepelende term van release management en continuous integration and continuous development. Het is de bedoeling om de verschillende softwareteams onderling te doen communiceren en samenwerken. Dit met doel om beter en sneller software uit te rollen. Het volgende artikel (Ebert, Gallardo, Hernantes & Serrano, 2016) legt in meer detail Devops uit. De 2 luiken van Devops worden besproken alsook waarom Devops belangrijk is. Ook worden veel tools en technieken aangehaald.

Het artikel (Ebert e.a., 2016) begint met uit te leggen wat Devops voor een organisatie kan betekenen. Het stelt dat Devops staat voor een betere samenwerking tussen ontwikkelen, kwaliteit, zekerheid en operaties. Het is lang niet zo dat Devops voor ieder softwareproject kan gebruikt worden, maar het wordt toch aangeraden. Ook vertrekt het artikel uit een gegeven dat er al softwareversie beheer in enige vorm aanwezig is.

Verder stelt het artikel (Ebert e.a., 2016) dat er 2 facetten zijn bij Devops. Enerzijds de build kant van de software en anderzijds het deployment gedeelte van de software.

Builden wordt volgens het artikel hoofdzakelijk op twee manieren gedaan. Aan de ene kant zijn er de traditionele build tools die meestal nog wat handmatig werk vereisen. Om in deze tools een build te doen moet er meestal een xml gemaakt worden met specifieke onderdelen. Dit kan nogal arbeidsintensief zijn. Langs de andere kant zijn er continuous integration procedures. Hierbij wordt getracht om op ieder moment test bare werkende code te hebben. Deze tools zijn een stuk gemakkelijker om te gebruiken en zijn meestal cloud gebaseerd. Dit is omdat het de bedoeling is om op een zeer vlot en sneller tempo updates te kunnen uitrollen. Het artikel beschrijft een tabel Ebert e.a. (2016) waarin een aantal tools met wat rand info beschreven staan.

Het artikel (Ebert e.a., 2016) stelt dat meestal hierna de software getest moet worden zodanig dat kwaliteit kan worden gegarandeerd. Dit moet zo efficiënt en snel mogelijk gebeuren waardoor er dus nood is aan infrastructure as code. Herbruikbare code die snel kan worden uitgevoerd over meerdere platformen. Verschillende automatisatie tools worden door het artikel besproken. De meeste zijn simpel in gebruik en gebruiken yaml of xml voor de test omgevingen te definiëren. Het uitrollen van test omgevingen voor de software wordt meestal in de cloud gedaan of met virtuele machines. Dit is afhankelijk van wat de vereisten zijn van de organisatie.

Ook bespreekt het artikel (Ebert e.a., 2016) een van de nieuwere manieren om software te testen. Ze bespreken het gebruik van microservices in de cloud. Hoofdzakelijk bespreekt het artikel de producten van Amazon AWS. Deze zijn zeer gemakkelijk te automatiseren en gemakkelijk in gebruik.

Dit artikel (Ebert e.a., 2016) is een meer waarde voor dit onderzoek omdat er een hele hoop tools, naast elkaar, kort besproken worden. De nadruk ligt niet op gebruiksvriendelijkheid maar eerder op wat de tools ondersteunen en of dat ze snel bruikbaar zijn voor Devops procedures. Dit artikel bevestigt dat er in de use case van dit onderzoek een nood is aan een goed gebouwde omgeving omdat kwaliteit een nummer een prioriteit geworden is.

2.1 Amazon AWS

2.1.1 Performance Analysis of High Performance Computing Applications on the Amazon Web Services Cloud

Dit onderzoek wil ook de cloud platformen en hun aanbod naast elkaar leggen. Ook wilt dit onderzoek vlug een idee krijgen hoe de verschillende datacenters van de cloud providers samengesteld zijn en hoe de verbindingen hiermee zijn. De volgende paper (Jackson e.a., 2010) is een prestatie test van een Amazon aws datacenter.

De paper (Jackson e.a., 2010) gebruikt synthetische wetenschappelijke testen om de prestatie van een aantal Amazon producten te testen. Een Amazon datacenter is wat raar samengesteld aangezien de eindgebruiker geen enkel idee op voorhand heeft over welke

hardware hij toegewezen krijgt in het datacenter. Bovendien is Amazon benadeelt op vlak van connectiviteit omdat het niet directe lijnen heeft naar de datacenters zoals Microsoft met Azure. Belangrijk is dat alle testen uitgevoerd door de paper, uitgevoerd zijn in hetzelfde datacenter.

In deze paper (Jackson e.a., 2010) testen de onderzoekers vooral of dat een Amazon datacenter geschikt zou zijn voor wetenschappelijk gebruik. Vandaar dat er vooral synthetische wetenschappelijke tools gebruikt worden. De tools zijn zorgvuldig geselecteerd zodanig dat de verschillende aspecten van een datacenter worden getest. Zowel de processor, als RAM, de harde schijven en ook de verbinding met de servers worden getest.

De paper (Jackson e.a., 2010) concludeert dat de prestatie van de systemen wel goed is maar dat het afhankelijk is van welk merk CPU de gebruiker krijgt. Aangezien hier geen controle over is, is het moeilijk om voor dezen redenen Amazon voor rekenintensieve doeleinden aan te raden. Ook is vastgesteld dat de connectie met het datacenter een aanzienlijke limitatie is voor taken die zeer transactie intensief zijn.

Deze paper (Jackson e.a., 2010) is niet zo zeer een meerwaarde voor dit onderzoek. Het artikel is redelijk verouderd en ondertussen is Amazon een van de grotere cloud platformen. Ook bespreekt deze paper niet zo goed de producten van Amazon. Ook is de use case die de paper beschrijft totaal verschillend van de use case die dit onderzoek gebruikt.

2.2 Microsoft Azure

2.2.1 Microsoft Azure and cloud computing

Aangezien binnen de use case van dit onderzoek Microsoft Azure een grote rol speelt, is het belangrijk om een zeer goed idee te vormen van hoe die cloud platform in elkaar zit en wat de verschillende services en producten zijn. Daarom is het volgende hoofdstuk uit een boek (Copeland, Soh, Puca, Manning & Gollob, 2015) redelijk informatief, zeker om ergens te starten.

Het eerste hoofdstuk uit het boek (Copeland e.a., 2015) is vooral bedoelt als een eerste kennismaking met het Azure platform. Azure is eigenlijk ontstaan uit Microsoft hun office 365 aanbiedingen. Microsoft biedt een grote hoeveelheid aan applicaties aan als onderdeel van dit pakket. Omdat die applicaties ergens moeten draaien, is Microsoft begonnen met het bouwen van datacenters om daar hun SaaS (Software as a Service) in te hosten. Al snel beseften ze dat er geld viel te verdienen met het aanbieden van remote services. Het duurde dan ook niet lang vooraleer Microsoft ook begon met IaaS (Infrastructure as a Service) aan te bieden. Dit was toen een unicum volgens het boek (Copeland e.a., 2015), Aangezien tot dan de meeste cloud platformen ontstaan zijn vanuit het verhuren van overschot aan computing power uit een mengelmoe van toestellen uit

een bepaald datacenter. Dit was onder andere een van de conclusies van een verouderd artikel over Amazon aws (Jackson e.a., 2010). Bij Microsoft waren dat speciaal gebouwde datacenters met die services in gedachten. Ook hun PaaS (Platform as a service) wordt kort aangehaald door het boek (Copeland e.a., 2015).

Voorbeelden van Azure Iaas:

- Azure virtual machines
- Azure virtual networks
- Azure virtual networks gateways
- Azure storage solutions
- ...

Voorbeelden van Azure Paas:

- Azure SQL database
- Azure website
- Azure content delivery network
- Azure DeVops
- ...

Verder legt het boek (Copeland e.a., 2015) kort uit wat vanuit Azure gedaan wordt op vlak van privacy en wetgevingen. Dit is minder interessant voor dit onderzoek. Ook gaat het boek kort in op waarom gebruikers, It professionals voor Azure cloud of een ander cloud platform zouden moeten kiezen. Zo haalt het boek (Copeland e.a., 2015) aan dat een cloud platform weinig onderhoud inhoud, minder dan een lokale opstelling. Ook is de eindgebruiker niet verantwoordelijk voor de hardware. Hun datacenter zijn redundant. Dus er is eigenlijk vrij weinig downtime. Ook de aangeboden producten zijn vrij compleet en gemakkelijk onderhoudbaar.

Verder geeft het boek (Copeland e.a., 2015) een korte introductie in het Azure web portaal. Deze heeft dit onderzoek voor kennismaking doeleinden eens doorgelopen. Veel interessants is hier voor dit onderzoek niet uit te melden. Het hoofdstuk is dus een snelle kennismaking met Azure. Dit dient als een basis voor verder onderzoek. Zo gaan we in deze literatuurstudie nog wat dieper in op Azure DeVops en een kleine hands-on. Ook IaaS van Azure wordt nog wat meer uitgediept.


2.2.2 Microsoft Azure Documentation

Zoals eerder aangehaald in deze literatuurstudie, speelt Microsoft Azure een belangrijke rol binnen de use case van dit onderzoek. Het is het vertrek punt voor de vergelijkingen. In het kader van dit onderzoek, is de website van Azure eens uitgeplozen. Dit omdat er een beeld kan gevormd worden over welke Azure services interessant kunnen zijn. Het volgende is een kort verslag. Er worden twee service categorieën aangehaald. Deze zijn


IaaS en PaaS (Infrastructure as a Services en Platform as a Service). Specifiek is er gekeken naar Azure virtual machine, Azure virtual networks, Azure virtual gateways en Azure DevOps. Het laatste is een samenvatting van een onlinecursus van een uur en informatie op Microsoft Docs.

In het kader van software release management en dan vooral de stap om de kwaliteit van software te controleren, is er gekeken of er misschien een mogelijkheid is om deze specifieke infrastructuur eventueel in de cloud te maken. Hiervoor is er een kort concept uitgewerkt. Dit concept beschrijft een hybride infrastructuur (verwijzing) waarbij eigenlijk alle niet use case specifieke toestellen in de cloud zitten. In theorie stond dit toe dat alle infrastructuur dan als code zou kunnen worden gedefinieerd.

Het aanbod qua mogelijkheden voor hardware om een virtuele machine aan te maken op Azure is enorm. Ook in tegenstelling tot de concurrenten wordt er zeer transparant omgesprongen met welke hardware er per optie beschikbaar gesteld wordt. De mogelijkheden verschillen enorm en zijn meestal voor specifieke doeleindes. Ook de prijzen schalen mede met de use cases. Zo zijn er specifieke opties voor machine learning. Of een optie voor machines met enorme hoeveelheden ram en CPU-kracht om met bepaalde databases overweg te kunnen. Ook de goedkopere basis opties zijn redelijk uitgebreid. Al deze opties worden door Azure onderverdeelt in categorieën die door middel van een letter worden aangeduid. Zie figuur 2.3.

 **Azure VM types**

	General Purpose	Compute Optimized	Memory Optimized	Storage Optimized	GPU	High Performance Compute
Type	DC, Av2, Dv2, Dv3, B, Dsv3	Fsv2, F	M, Dv2, G, DSv2, GS, Ev3	Ls	NC, NCv2, ND, BV, NVv2	H
Description	Balanced CPU and memory	High ratio of compute to memory	High ratio of memory to compute	High disk throughput and IO	Specialized with single or multiple NVIDIA GPUs	High memory and compute power – fastest and most powerful
Uses	Testing and dev, small-med databases, low traffic web servers	Medium traffic web servers, network appliances, batch processing, app servers	Relational database services, analytics, and larger caches	Big Data, SQL, NoSQL databases	Compute intensive, graphics-intensive, and visualization workloads	Batch processing, analytics, molecular modeling, and fluid dynamics, low latency RDMA networking

 **ParkMyCloud**

Figuur 2.3: Figuur van (<https://p2zk82o7hr3yb6ge7gzxx4ki-wpengine.netdna-ssl.com/wp-content/uploads/azure-vm-types-comparison-1.jpg>). Chart met alle VM tiers van azure en hun specifieke code.

Verder is Azure de oudere manier van het definiëren van een Azure virtual machine aan het afraden. Hiernaast zijn er ook nog een tal van mogelijkheden op vlak van virtual networking. Het belangrijkste is dat er een mogelijkheid bestaat om een virtueel netwerk

volledig te isoleren van het internet. Dit staat toe dat enkel verkeer dat op dat specifieke netwerk is, aan de virtuele machines kan. Er wordt dan wel meestal een Azure network firewall node voorzien die in dit virtueel netwerk zit, zodanig dat de connectiviteit met de machines ten aller tijden gegarandeerd kan worden. Om dan connectiviteit te hebben met het virtueel netwerk wordt er gebruikgemaakt van een Azure Gateway node. Deze staat een point to point VPN (Virtual Private Network) tunnel toe. Dit is eigenlijk een directe, encrypteerde verbinding met het Azure netwerk. Hiervoor is wel specifieke hardware nodig lokaal in het netwerk. De kost van deze services is eigenlijk bijna niks. Azure werkt immers met een pay-to-run, pay-on-the-go system. Dit wil zeggen dat er dus moet betaald worden voor de aanmaak van de services en daarna voor het verbruik. Microsoft heeft op zijn documentatie portaal tal van stap voor stap handleiding voor het instellen van deze services. Ook concepten voor hybride opstellingen staan hier uitgelegd. Maar dit zou deze literatuurstudie te veel doen afwijken.

Het zou dus in theorie mogelijk moeten zijn om een hybride cloud infrastructuur te voorzien die dynamisch is voor de specifieke te testen projecten. De vraag is of dit handig is in gebruik en niet nodeloos complexiteit toevoegt.

Azure DevOps is een platform van Azure dat organisaties toestaat om bepaalde procedures te definiëren voor het uitrollen van projecten. Het platform staat dan ook integratie toe met andere project follow-up tools van Microsoft. Deze procedures kunnen juist zoals de virtuele machines op Azure via code worden gedefinieerd of via een duidelijk en gemakkelijk te gebruiken GUI. De bedoeling van Azure DevOps is eigenlijk om het oude TFS-systeem te vervangen en een verbeterde interface aan te bieden. Meestal wordt DevOps gebruikt om aan Continuous Integration te doen. Dus er wordt een repository waar de code opstaat gedefinieerd. Hierna bestaat er een mogelijkheid om de code te compileren, automatisch testen te runnen en deze code dan weer door te schuiven naar een volgend stadium. Hier wordt de code dan uitgerold naar een omgeving voor verder kwaliteitscontrole. Het mooie aan dit platform is dat de gebruiker of organisatie niet verplicht is om deze code in een virtuele machine in de cloud te compileren of testen. Er is volledige controle over de procedures.

In het kader van dit onderzoek is dit zeer belangrijk aangezien dit platform op dit moment in gebruik genomen wordt. Er is ook een onlinecursus gevolgd met een basis uitleg en een labo. Dit heeft duidelijk gemaakt dat het mogelijk is om vanuit DevOps op een lokale test omgeving uit te rollen.

3. Methodologie

Etiam pede massa, dapibus vitae, rhoncus in, placerat posuere, odio. Vestibulum luctus commodo lacus. Morbi lacus dui, tempor sed, euismod eget, condimentum at, tortor. Phasellus aliquet odio ac lacus tempor faucibus. Praesent sed sem. Praesent iaculis. Cras rhoncus tellus sed justo ullamcorper sagittis. Donec quis orci. Sed ut tortor quis tellus euismod tincidunt. Suspendisse congue nisl eu elit. Aliquam tortor diam, tempus id, tristique eget, sodales vel, nulla. Praesent tellus mi, condimentum sed, viverra at, consectetur quis, lectus. In auctor vehicula orci. Sed pede sapien, euismod in, suscipit in, pharetra placerat, metus. Vivamus commodo dui non odio. Donec et felis.

Etiam suscipit aliquam arcu. Aliquam sit amet est ac purus bibendum congue. Sed in eros. Morbi non orci. Pellentesque mattis lacinia elit. Fusce molestie velit in ligula. Nullam et orci vitae nibh vulputate auctor. Aliquam eget purus. Nulla auctor wisi sed ipsum. Morbi porttitor tellus ac enim. Fusce ornare. Proin ipsum enim, tincidunt in, ornare venenatis, molestie a, augue. Donec vel pede in lacus sagittis porta. Sed hendrerit ipsum quis nisl. Suspendisse quis massa ac nibh pretium cursus. Sed sodales. Nam eu neque quis pede dignissim ornare. Maecenas eu purus ac urna tincidunt congue.

Donec et nisl id sapien blandit mattis. Aenean dictum odio sit amet risus. Morbi purus. Nulla a est sit amet purus venenatis iaculis. Vivamus viverra purus vel magna. Donec in justo sed odio malesuada dapibus. Nunc ultrices aliquam nunc. Vivamus facilisis pellentesque velit. Nulla nunc velit, vulputate dapibus, vulputate id, mattis ac, justo. Nam mattis elit dapibus purus. Quisque enim risus, congue non, elementum ut, mattis quis, sem. Quisque elit.

Maecenas non massa. Vestibulum pharetra nulla at lorem. Duis quis quam id lacus dapibus interdum. Nulla lorem. Donec ut ante quis dolor bibendum condimentum. Etiam egestas

tortor vitae lacus. Praesent cursus. Mauris bibendum pede at elit. Morbi et felis a lectus interdum facilisis. Sed suscipit gravida turpis. Nulla at lectus. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Praesent nonummy luctus nibh. Proin turpis nunc, congue eu, egestas ut, fringilla at, tellus. In hac habitasse platea dictumst.

Vivamus eu tellus sed tellus consequat suscipit. Nam orci orci, malesuada id, gravida nec, ultricies vitae, erat. Donec risus turpis, luctus sit amet, interdum quis, porta sed, ipsum. Suspendisse condimentum, tortor at egestas posuere, neque metus tempor orci, et tincidunt urna nunc a purus. Sed facilisis blandit tellus. Nunc risus sem, suscipit nec, eleifend quis, cursus quis, libero. Curabitur et dolor. Sed vitae sem. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Maecenas ante. Duis ullamcorper enim. Donec tristique enim eu leo. Nullam molestie elit eu dolor. Nullam bibendum, turpis vitae tristique gravida, quam sapien tempor lectus, quis pretium tellus purus ac quam. Nulla facilisi.

4. Conclusie

Curabitur nunc magna, posuere eget, venenatis eu, vehicula ac, velit. Aenean ornare, massa a accumsan pulvinar, quam lorem laoreet purus, eu sodales magna risus molestie lorem. Nunc erat velit, hendrerit quis, malesuada ut, aliquam vitae, wisi. Sed posuere. Suspendisse ipsum arcu, scelerisque nec, aliquam eu, molestie tincidunt, justo. Phasellus iaculis. Sed posuere lorem non ipsum. Pellentesque dapibus. Suspendisse quam libero, laoreet a, tincidunt eget, consequat at, est. Nullam ut lectus non enim consequat facilisis. Mauris leo. Quisque pede ligula, auctor vel, pellentesque vel, posuere id, turpis. Cras ipsum sem, cursus et, facilisis ut, tempus euismod, quam. Suspendisse tristique dolor eu orci. Mauris mattis. Aenean semper. Vivamus tortor magna, facilisis id, varius mattis, hendrerit in, justo. Integer purus.

Vivamus adipiscing. Curabitur imperdiet tempus turpis. Vivamus sapien dolor, congue venenatis, euismod eget, porta rhoncus, magna. Proin condimentum pretium enim. Fusce fringilla, libero et venenatis facilisis, eros enim cursus arcu, vitae facilisis odio augue vitae orci. Aliquam varius nibh ut odio. Sed condimentum condimentum nunc. Pellentesque eget massa. Pellentesque quis mauris. Donec ut ligula ac pede pulvinar lobortis. Pellentesque euismod. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent elit. Ut laoreet ornare est. Phasellus gravida vulputate nulla. Donec sit amet arcu ut sem tempor malesuada. Praesent hendrerit augue in urna. Proin enim ante, ornare vel, consequat ut, blandit in, justo. Donec felis elit, dignissim sed, sagittis ut, ullamcorper a, nulla. Aenean pharetra vulputate odio.

Quisque enim. Proin velit neque, tristique eu, eleifend eget, vestibulum nec, lacus. Vivamus odio. Duis odio urna, vehicula in, elementum aliquam, aliquet laoreet, tellus. Sed velit. Sed vel mi ac elit aliquet interdum. Etiam sapien neque, convallis et, aliquet vel, auctor non, arcu. Aliquam suscipit aliquam lectus. Proin tincidunt magna sed wisi. Integer blandit

lacus ut lorem. Sed luctus justo sed enim.

Morbi malesuada hendrerit dui. Nunc mauris leo, dapibus sit amet, vestibulum et, commodo id, est. Pellentesque purus. Pellentesque tristique, nunc ac pulvinar adipiscing, justo eros consequat lectus, sit amet posuere lectus neque vel augue. Cras consectetur libero ac eros. Ut eget massa. Fusce sit amet enim eleifend sem dictum auctor. In eget risus luctus wisi convallis pulvinar. Vivamus sapien risus, tempor in, viverra in, aliquet pellentesque, eros. Aliquam euismod libero a sem.

Nunc velit augue, scelerisque dignissim, lobortis et, aliquam in, risus. In eu eros. Vestibulum ante ipsum primis in faucibus orci luctus et ultrices posuere cubilia Curae; Curabitur vulputate elit viverra augue. Mauris fringilla, tortor sit amet malesuada mollis, sapien mi dapibus odio, ac imperdiet ligula enim eget nisl. Quisque vitae pede a pede aliquet suscipit. Phasellus tellus pede, viverra vestibulum, gravida id, laoreet in, justo. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Integer commodo luctus lectus. Mauris justo. Duis varius eros. Sed quam. Cras lacus eros, rutrum eget, varius quis, convallis iaculis, velit. Mauris imperdiet, metus at tristique venenatis, purus neque pellentesque mauris, a ultrices elit lacus nec tortor. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent malesuada. Nam lacus lectus, auctor sit amet, malesuada vel, elementum eget, metus. Duis neque pede, facilisis eget, egestas elementum, nonummy id, neque.

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

Software release management kan op verschillende manieren geïmplementeerd worden. Dit zijn vaak complexe en zeer use case specifieke omgevingen. Ook testen en debuggen van software en infrastructuur zijn belangrijk voor het afleveren van kwalitatieve producten. Een vraag die hierbij opduikt is of dit op lokale infrastructuur moet gebeuren of op cloud platformen die IaaS (Infrastructure as a Service), TaaS (Testing as a Service) of SaaS (Software as a Service) aanbieden.

Der bestaan al tientallen papers over de performantie, flexibiliteit, enz. in een cloud omgeving. Ook over software release management zijn er al talloze papers geschreven. Des ondanks is het toch nog interessant om dit voor een specifieke use case te bekijken. Aucxis heeft al een software release omgeving op Azure. Ze hebben echter geen onderzoek gedaan naar andere cloud platformen of oplossingen. Deze paper is in de eerste plaats bedoelt om voor Aucxis een duidelijk beeld te scheppen over de mogelijkheden.

Deze paper zal proberen in detail duidelijkheid te scheppen over wanneer een cloud platform een goede keuze is en wanneer niet, wat de beste tools zijn, hoe het zit met de gebruiksvriendelijkheid en de prijs. Ook de performantie is niet onbelangrijk. Deze paper zal zich ook afvragen hoe data privacy kan gecontroleerd en geïmplementeerd worden. Op het einde van deze paper zal er een conclusie gemaakt worden over welke optie het best

past binnen de use case van Aucxis.

A.2 Literatuurstudie

A.2.1 Conventional Software Testing Vs. Cloud Testing

Dit artikel (Katherine & Alagarsamy, 2012) snijdt oppervlakkig aan wat pijnpunten kunnen zijn voor testen van software in een cloud platform. Hier gebruiken ze een web applicatie als voorbeeld. Het artikel stelt een aantal punten voor, waarop getest kan worden. Dit zijn de traditionele test cases. Functionaliteit testen, gebruiksvriendelijkheid testen, interface testen, compatibiliteitstesten, performantie testen en tot slot veiligheid en privacy testen. Het artikel stelt een aantal uitdagingen voor bij lokale omgevingen. Dit gaat over de kost, over het onderhoud ervan, hoe eenzijdig een lokale omgeving is en voor ieder project een nieuwe omgeving gebouwd moet worden en het feit dat het geen accurate weergave is van de werkelijke omgevingen waarin de software zal draaien.

Verder legt het artikel kort uit wat voor mogelijke cloud oplossingen er op dat moment zijn. Eerst moet er onderscheid gemaakt worden in hoe een cloud platform uitgerold kan worden. Er bestaat enerzijds de publieke cloud (Google Cloud Platform, AWS, Azure, DigitalOcean) en anderzijds een privé cloud. De privé cloud is een lokale opstelling die beschikbaar is over het internet naar andere gebruikers. Ook bestaat er iets zoals een hybride cloud. Hierna wordt er dan nog onderscheid gemaakt tussen welke services deze platformen kunnen aanbieden. Dit artikel beschrijft er drie. SaaS (service as a Service), PaaS (Platform as a Service), IaaS (Infrastructure as a Service). Het artikel maakt toch een onderscheid van het traditionele testen. Aangezien het in de cloud mogelijk is om de applicatie te testen op load, stress en capaciteit.

Tot slot stelt dit artikel belangrijke uitdagingen aan het licht. Zo is de beveiliging van de data die ontvangen, verstuurd of bewaard worden op een cloud platform belangrijk. Ook zijn er over alle platformen heen weinig tot geen standaarden vastgelegd over zowel de performantie van de systemen als de beschikbaarheid op vlak van aanbod. Het zijn juist deze uitdagingen die belangrijk kunnen zijn voor de onderzoeksvragen.

A.2.2 Software Testing Based on Cloud Computing

In dit artikel (Jun & Meng, 2011) wordt er opnieuw in detail beschreven wat de verschillende platform mogelijkheden zijn zoals IaaS, PaaS, SaaS. Het artikel probeert ook een definitie te geven aan testen op cloud platformen. Tevens geeft het artikel ook een aantal redenen waarom cloud testen een stuk beter zou zijn dan het testen in lokale omgevingen. Deze komen grotendeels overeen met het vorige artikel. In dit artikel wordt er ook besproken dat beveiliging een groot probleem kan zijn. Juist zoals het vorige artikel wordt er gesteld dat het een echte uitdaging is om test datasets in de cloud te gebruiken aangezien deze meestal afkomstig zijn van een klant. Het artikel bespreekt ook een aantal mogelijkheden om met de cloud te verbinden en testomgevingen te configureren. Het

artikel bespreekt vooral virtualisatie.

Dit artikel bevestigt deels het vorige artikel. Het geeft wat detail en inzicht in cloud testen. Dit artikel sluit aan met de onderzoeksvragen en geeft richting in probleemgebieden.

A.2.3 Benchmarking in the Cloud: What It Should, Can, and Cannot Be

Zomaar willekeurig testen of experimenten uitvoeren is meestal geen goed idee. Er is nood aan een goed gedefinieerde methode om deze testen uniform uit te voeren. Dit artikel (Folkerts e.a., 2013) beschrijft in extreem detail hoe een cloud platform het best getest kan worden. Er wordt beschreven wat de valkuilen zijn bij performantietesten van een cloud platform. Zo wordt het testen van een lokale omgeving vergeleken met het testen van een cloud omgeving. Dit is een hele uitdaging aangezien de hardware van een cloud platform meestal verschilt en niet hetzelfde is. Het artikel beschrijft het testen van een cloud platform aan de hand van een aantal use cases. Het artikel gebruikt hiervoor use cases die schaalbaar zijn en in pieken benaderd worden. Ook beschrijft het artikel dat het belangrijk is om goed te definiëren wat er allemaal getest moet worden over de verschillende platformen heen.

Dit artikel biedt een gedetailleerd inzicht in het opstellen van benchmarks voor cloud omgevingen en zal een belangrijke leidraad vormen voor het opstellen van de experimenten.

A.2.4 When to Migrate Software Testing to the Cloud?

Wanneer moet er gedacht worden om naar een cloud omgeving te migreren? Dit artikel (Parveen & Tilley, 2010) beschrijft vanaf wanneer het nuttig is om naar de cloud te migreren. Ook beschrijft het artikel kort wat de ervaring was bij een migratie. Dit artikel is interessant omdat het kort een inzicht geeft in wanneer het nuttig en efficiënt is om naar een cloud te migreren. Dit is interessant omdat dit aansluit bij de probleemstelling of een cloud omgeving voor testen nu zoveel beter kan zijn dan een lokale omgeving.

A.3 Methodologie

Een groot deel van de onderzoeksvragen zullen beantwoord worden door onderzoekswerk en vergelijkingen. Zo zal deze paper in detail bespreken welke cloud platformen er bestaan en wat de mogelijk plannen (tarieven en voor gedefinieerde configuraties) zijn. De verschillende platformen zullen op een duidelijke manier naast elkaar gelegd worden en vergeleken worden. Ook zal er gekeken worden naar software release tools. Op basis hiervan zal er dan een keuze gemaakt worden welke platformen er in aanmerking komen voor een proof of concept.

Ook zal deze paper methodes beschrijven en testen door middel van experimenten wat betreft het behouden van data privacy. Deze paper zal bijvoorbeeld een experiment uitvoeren met een proxy (een tunnel met encryptie naar het datacenter) om de gebruiksvriendelijkheid hiervan te testen.

Na het onderzoek zal er een proof of concept opgezet worden met beste alternatief. Er zal getracht worden om de huidige omgeving van Aucxis zo goed mogelijk te benaderen in functionaliteit. De bedoeling is om dezelfde tools te gebruiken om de omgevingen te monitoren (bijvoorbeeld: een Docker image voor dezelfde configuratie en Telegraf en Grafana voor monitoring). Ook zal er getracht worden om dezelfde testen te gebruiken. Dit alles zal over een bepaalde periode draaiende gehouden worden waarna alle resultaten gebundeld zullen worden. Hierbij zitten ook een aantal subjectieve waarnemingen aangezien er ook onderzoek zal gedaan worden naar gebruiksvriendelijkheid. In deze proof of concept zal er een alternatief platform vergeleken worden met het huidige systeem.

A.4 Verwachte resultaten

A.4.1 Vergelijking van platformen

Er wordt verwacht dat de huidige cloud omgeving vanuit de use case van Aucxis de beste oplossing is, zeker op vlak van gebruiksvriendelijkheid en efficiëntie. Ondanks dit wordt er verwacht dat de alternatieven een even goede oplossing zullen aanbieden. Deze zullen waarschijnlijk niet de meest gebruiksvriendelijkste of goedkoopste oplossingen zijn. Voor de tools voor software release management wordt er verwacht dat er gelijkaardige alternatieven aan de huidige tools vanuit de use case gevonden worden.

A.4.2 Proof of concept

Er wordt verwacht dat er een werkende alternatieve omgeving opgezet zal worden die de huidige functionaliteit benaderd. Er wordt verwacht dat de performantie van dit platform op z'n minst gelijkaardig is aan de huidige use case. Er wordt verwacht dat de gebruiksvriendelijkheid en de kost verbeteren.

A.4.3 beveiliging experiment

Voor dit experiment zijn er gemengde verwachtingen. Vooral op het vlak van tijdrovende configuraties. Er wordt verwacht dat een proxy het meest flexibel is en het meest gebruiksvriendelijk, zeker als het vergeleken wordt met encryptie of andere tools.

A.5 Verwachte conclusies

A.5.1 Vergelijking van platformen

Het is moeilijk om een conclusie te voorspellen over welk cloud platform het beste uit de vergelijkingen zal komen. Ook is het moeilijk te voorspellen wat de beste tools zullen zijn. Er wordt wel verwacht dat er een degelijk alternatief voor de huidige oplossing gevonden zal worden.

A.5.2 Proof of concept

De conclusie zal voor dit experiment zeer duidelijk zijn. Deze zal afhangen van de werking van het alternatief. Is het alternatief sneller, gebruiksvriendelijker, enz. dan zal deze conclusie positief zijn. Anders niet.

A.5.3 beveiliging experiment

Om de data privacy te garanderen wordt er verwacht dat een proxy de beste en meest doelenlijke oplossing zal zijn. Het valt moeilijk te zeggen of andere tools of methodes beter zullen presteren.

Bibliografie

- Barshefsky. (2005). *Methodes and systems for software release management*. 2005/0216486. DUFT SETTER OLLILA & BORNSSEN LLC 2060 BROADWAYSUTE300BOULDER, CO 80302(US).
- Copeland, M., Soh, J., Puca, A., Manning, M. & Gollob, D. (2015, oktober 8). *Microsoft Azure*. Springer-Verlag GmbH. Verkregen van https://www.ebook.de/de/product/25127919/marshall_copeland_julian_soh_anthony_puca_mike_manning_david_gollob_microsoft_azure.html
- Ebert, C., Gallardo, G., Hernantes, J. & Serrano, N. (2016). DevOps. *IEEE Software*, 33(3), 94–100. doi:10.1109/ms.2016.68
- Folkerts, E., Alexandrov, A., Sachs, K., Iosup, A., Markl, V. & Tosun, C. (2013). Benchmarking in the cloud: what it should, can, and cannot be. *Springer-Verlag Berlin Heidelberg*. TPCTC 2012, LNCS 7755.
- Jackson, K. R., Ramakrishnan, L., Muriki, K., Canon, S., Cholia, S., Shalf, J., ... Wright, N. J. (2010). Performance analysis of high performance computing applications on the amazon web services cloud. In *2010 IEEE Second International Conference on Cloud Computing Technology and Science*. IEEE. doi:10.1109/cloudcom.2010.69
- Jun, W. & Meng, F. (2011). Software Testing Based on Cloud Computing. In *2011 International Conference on Internet Computing and Information Services*. IEEE. doi:10.1109/icicis.2011.51
- Katherine, M. & Alagarsamy, D. K. (2012). Conventional software testing vs cloud testing. *International Journal Of Scientific & Engineering Research*, 3(9).
- Lahtela, A. & Jantti, M. (2011). Challenges and problems in release management process: A case study. In *2011 IEEE 2nd International Conference on Software Engineering and Service Science*. IEEE. doi:10.1109/icsess.2011.5982242
- Meyer, M. (2014). Continuous integration and its tools. *IEEE Software*, 31(3), 14–16. doi:10.1109/ms.2014.58

- Parveen, T. & Tilley, S. (2010). When to migrate software testing to the cloud? *Third International Conference on Software Testing, Verification, and Validation Workshops*. DOI 10.1109/ICSTW.2010.77 IEEE. doi:10.1109/ICSTW.2010.77
- van der Hoek, A. & Wolf, A. L. (2002). Software release management for component-based software. *Software: Practice and Experience*, 33(1), 77–98. doi:10.1002/spe.496