HO
GENT

Faculteit Bedrijf en Organisatie

High Availability oplossingen voor PostgreSQL: een vergelijkende studie en proof of concept

Elias Ameye

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

Promotor:
Thomas Aelbrecht
Co-promotor:
Ruben Demey

Instelling: —

Academiejaar: 2020-2021

Tweede examenperiode

Faculteit Bedrijf en Organisatie

High Availability oplossingen voor PostgreSQL: een vergelijkende studie en proof of concept

Elias Ameye

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

Promotor:
Thomas Aelbrecht
Co-promotor:
Ruben Demey

Instelling: —

Academiejaar: 2020-2021

Tweede examenperiode

Woord vooraf

In dit onderzoek zal ik het hebben over High Availability oplossingen en eht belang ervan in PostgreSQL clusters. Met de Coronapandemie die nog overal aanwezig is, is er een grote stijging in het digitale gebruik. Online winkelen heeft een enorme groei gekend. Koerierbedrijven hebben overuren moeten draaien om pakjes en post te brengen bij de mensen thuis. Technologie bedrijven kennen een extra druk omdat er meer beroep wordt gedaan op IT-services. Deze digitale (r)evolutie toont ons dat beschikbaarheid van diensten zeker heel relevant is. Stel je voor dat een server van Zalando, door software problemen, uitvalt. Alle aankopen van het laatste uur zijn niet doorgekomen. Een financieel drama. De oplossing? Een standby server die inspringt in geval van downtime. Resultaat? Geen downtime, geen financieel drama, geen geknoei met corrupte data. Met dit voorbeeld wil ik het belang aantonen van High Availability in clusters. Stel, er loopt iets mis, kan het probleem snel opgelost worden, zonder dat de klant of het bedrijf er iets van nadelige ervaringen aan overhoudt.

Ik wil graag Thomas Aelbrecht, mijn promotor, bedanken voor de goede begeleiding en de duidelijke feedback. Ongeveer tweewekelijks kwamen we samen om eens te overlopen hoever ik zat. Hierdoor gaf ik mijzelf telkens een deadline tegen wanneer ik bepaalde zaken verricht wou hebben.

Ook wil ik Ruben Demey, co-promotor, bedanken om bij de vragen die ik had, duidelijke antwoorden te geven waardoor ik telkens een stap dichter was bij het einde.

Tot slot wil ik ook nog mijn vriendin bedanken voor de steun en toeverlaat.

Samenvatting

Lorem ipsum dolor sit amet, consectetuer adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetuer 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, consectetuer 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

Lijst van figuren

Lijst van tabellen

1. Inleiding

1.1 Probleemstelling

Deze vergelijkende studie kan een meerwaarde bieden aan bedrijven die werken met een kleine of grote PostgreSQL cluster waarin zij High Availability willen implementeren. Het kan ook een meerwaarde zijn voor bedrijven die al High Availability implementaties hebben in hun cluster, maar die een frisse blik nodig hebben, of willen upgraden naar een meer hedendaagse oplossing. Hiermee is dus vooral de focus gericht op systeem- en netwerkbeheerders die dagelijks bezig zijn met het onderhouden en/of beheren van servers, meer specifiek database servers.

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 specifiëren in deelvragen. Bv. als je onderzoek gaat over performantiemetingen, dan

1.3 Onderzoeksdoelstelling

Het beoogde resultaat van deze bachelorproef is om uit de vergelijkende studie één oplossing voor te stellen die kan gebruikt worden voor implementatie van een High Available

PostgreSQL cluster. Hieraan gekoppeld zal ook een eerste poging tot een proof-of-concept getoond waarin deze oplossing geïmplementeerd zit.

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. Hierin zal ik mij vooral focussen op High Availability, PostgreSQL, Puppet, clustering en de reeds aanwezige oplossingen voor High Availability.

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

Dit onderzoek is een vergelijkende studie tussen verschillende PostgreSQL oplossingen. In dit hoofdstuk zullen al verschillende oplossingen aan bod komen. Hiermee zal worden gekeken naar de nadruk die gelegd wordt bij elke oplossing. Op deze manier kan gekeken worden naar welke elementen regelmatig terugkomen, en dus welke we kunnen gebruiken om verschillende oplossingen met elkaar te vergelijken.

2.1 PosgreSQL

Databank

Wikipedia omschrijft een databank als een georganiseerde collectie van data, opgeslagen in en toegankelijk vanuit een computersysteem (https://en.wikipedia.org/wiki/Database: :text=A%20database%2

Relationele databank

Een relationele databank is een type databank waarin gebruik wordt gemaakt van een structuur die het mogelijk maakt om gegevens te identificeren en te benaderen in relatie tot een ander deeltje data in diezelfde databank. Deze gegevens worden vaak georganiseerd in tabellen. Deze tabellen kunnen honderden, duizenden, miljoenen rijen en kolommen aan data hebben. Een kolom kent vaak ook een specifiek gegevenstype. Deze gegevenstypes kunnen getallen (integers), woorden (strings) of andere soorten bevatten. (https://www.codecademy.com/articles/what-is-rdbms-sql: :text=A%20relational%20database%20is%20a,dat

Object-georiënteerde databank

Een objectgeoriënteerde database (OODBMS) is een type databank die zich baseert op objectgeoriënteerd programmeren (OOP). De gegevens worden hier voorgesteld en opgeslagen in de vorm van objecten. OODBMS worden ook objectdatabases of objectgeoriënteerde databasemanagementsystemen genoemd (https://www.c-sharpcorner.com/article/what-are-object-oriented-databases-and-their-advantages2/: :text=An

SQL

SQL (structured query language) is de eigen taal specifiek ontwikkelt voor interactie met databanken. Een databank modelleert entiteiten uit het echte leven en slaat deze op in tabellen. Via SQL is het mogelijk om de gegevens in deze tabellen te manipuleren. (https://www.datacamp.com/community/tutorials/what-is-sql?utm$_source$ = $adwords_ppcutm_campaignid = 898687156utm_adgroupid = 48947256715utm_device = cutm_keyword = utm_matchtype = butm_network = gutm_adpostion = utm_creative = 229765585183utm_targetid = dsa-429603003980utm_looc_interest_ms = utm_looc_physical_ms = 1001208gclid = Cj0KCQjwse-DBhC7ARIsAI8YcWJOxNwWfRDfjloPnAcswV6QsafwgHI0hqa-gQFH5fj59JLPM04NOxMaAmud$

Object-relationele databank

Een object-relationele database (ORD / ORDBMS) is een samenstelling uit zowel een relationele database (RDB / RDBMS), als een object-georiënteerde database (OOD / OODBMS). Samen ondersteunt het de basiscomponenten van elk objectgeoriënteerd databasemodel in zijn schema's en gebruikte querytaal, zoals klassen, overerving en objecten. Het bevat aspecten en kenmerken van bovenstaande genoemde modellen. Zo wordt het relationele duidelijk in de manier van opslaan van gegevens. Deze worden opgeslagen in een traditionele database en worden dan met behulp van SQL query's gemanipuleerd en benaderd. Aan de andere kant is ook het objectgeoriënteerde gedeelte merkbaar, namelijk dat de database beschouwd wordt als een objectopslag. Kort gezegd is één van de voornaamste doelstellingen van een objecte-relationele database, het dichten van de kloof tussen relationele en objectgeoriënteerde modelleringstechnieken en conceptuele datamodelleringstechnieken zoals daar zijn het entiteit-relatiediagram (ERD) en object-relationeel mappen (ORM) (https://www.techopedia.com/definition/8714/object-relational-database-ord). Klasses, Overerving, Types, Functies zijn kenmerken van de object-relationele database en zullen de basisconcepten vormen voor PostgreSQL. Een klasse is een verzameling van gegevenstypes die bij eenzelfde soort iets horen. Bijvoorbeeld een klasse CD kan als kenmerken hebben: Titel, zanger, Datum uitgave, aantal liedjes... . Overerving is wanneer een klasse bepaalde kenmerken overerft, krijgt van een superklasse. Bijvoorbeeld een klasse Olifant erft van de klasse Zoogdier kenmerken. Hierin is de klasse Olifant een specialisatie van de klasse Zoogdier. Een type is hierboven al eens genoemd geweest. Dit gaat over de verschillende soorten data die er zijn. Getallen, woorden, objecten zijn hier voorbeelden van. Functies zijn een reeks SQL-statements die een specifieke taak uitvoeren. Functies bevorderen de herbruikbaarheid van code.

PostgreSQL

PostgreSQL is een open source systeem dat zich toelegt op het beheer van object-relationele databases. Het heeft meer dan 30 jaar actieve ontwikkeling en heeft een sterke reputatie op

vlak van betrouwbaarheid, robuustheid van functies en prestaties (https://www.postgresql.org/). PostgreSQL biedt een uitgebreide set van functionaliteiten die een hoge mate van customisatie mogelijk maakt binnen het systeem. Dit gaat van data administratie, beveiliging, tot backup en herstel . PostgreSQL wordt regelmatig bijgewerkt door de PostgreSQL Global Development Group en bijdragers uit de community. Deze community ondersteunt zichzelf en zijn gebruikers door het aanbieden van online educatieve bronnen en communicatiekanalen, zoals daar zijn PostgreSQL wiki, online forums en officiële documentatie. Er zijn ook bedrijven die commerciële support bieden aan een prijs (https://nethosting.com/mysql-vs-postgresql-2019-showdown/). Volgens DB-Engines is PostgreSQL de vierde database die vandaag de dag het meest gebruikt wordt en de tweede meest gebruikte open source database, na MySQL (https://db-engines.com/en/ranking). DB-Engines verklaarde PostgreSQL in 2017, 2018 en 2020 het DBMS (Database management system) van het jaar (https://db-engines.com/en/system/PostgreSQL). PostgreSQL biedt veel mogelijkheden om ontwikkelaars te helpen bij het bouwen van applicaties, om beheerders te helpen bij het beschermen van data-integriteit en het bouwen van fouttolerante omgevingen. Het helpt ook bij het beheren van data, hoe groot of hoe klein de dataset ook is. PostgreSQL voldoet sinds september 2020 aan 170 van de 179 verplichte functies voor SQL:2016 Core conformiteit. Schaalbaarheid valt ook toe te schrijven aan PostgreSQL, dit zowel in de hoeveelheid data die het kan beheren, als in het aantal gelijktijdige gebruikers dat het kan accomoderen. Er zijn actieve PostgreSQL clusters in productie omgevingen die terabytes aan data beheren, en gespecialiseerde systemen die petabytes beheren (https://www.postgresql.org/about/).

Postgres speelt in op de bovenvernoemde vier basisconcepten van een object-relationele databank zodat gebruikers het systeem makkelijk kunnen uitbreiden. Deze vier kenmerken, naast nog andere functies maken van Postgres een object-relationele database. Bovenstaande vermelde kenmerken zouden doen blijken dat Postgres voornamelijk een object-georiënteerde database is, maar de ondersteuning van de traditionele relationele databases, toont duidelijk aan dat, ondanks de object-georiënteerde kenmerken, Postgres stevig verankerd is in de relationele database wereld (https://www.postgresql.org/docs/6.3/c0101.htm).

2.2 High Availability

Het doel van High Availability architectuur is ervoor te zorgen dat een server, website of applicatie verschillende vraagbelastingen en verschillende soorten storingen kan verdragen. En dit met de minst mogelijke downtime. Door gebruik te maken van best practices die zijn ontworpen om hoge beschikbaarheid te garanderen, helpt dit volgens ServersAustralia om in een organisatie maximale productiviteit en betrouwbaarheid te bereiken. (https://www.serversaustralia.com.au/resources/blog/what-is-high-availability-ha-and-do-i-need-it/: :text=The

High Availability is het vermogen van een systeem om continu operationeel te blijven te zijn gedurende een wenselijk lange tijd. Men kan Availability meten ten opzicht van 100% operationeel, als in, nooit uitvallen. Beschikbaarheid wordt vaak uitgedrukt als een percentage van uptime in een bepaald jaar op basis van de SLA's, Service Level Agreements (https://www.enterprisedb.com/blog/what-does-database-high-availability-

really-mean). Vaak duidt men deze norm aan als de Five 9's, namelijk 99,999% beschikbaarheid (https://searchdatacenter.techtarget.com/definition/high-availability). High availability impliceert dat delen van een systeem volledig zijn getest en dat er voorzieningen zijn voor storingen/failures in de vorm van redundante componenten. Servers kunnen worden ingesteld om in geval van nood de verantwoordelijkheden over te dragen aan een externe server, in een back-up proces. Hier spreekt men dan van failover (https://www.techopedia.com/definition/1021/high-availability-ha).

Belangrijke principes van High Availability zijn:

1. **Het elimineren van single point of failure**: Toevoeging van redundantie zorgt er voor zodat het falen van een onderdeel in het systeem niet leidt tot het volledige falen van een geheel systeem.

2. **Betrouwbare cross-over**: In een redundant systeem wordt het kruispunt zelf een single point of failure. Fouttolerante systemen moeten voorzien in een betrouwbaar crossover- of automatisch omschakelingsmechanisme om storingen te voorkomen.

3. **Storingdetectie**: Als bovenstaande principes proactief bewaakt worden, dan zal een gebruik misschien nooit een systeemstoring zien. Postgres biedt de bouwstenen om bovenstaande principes volledig uit te werken zodat er op deze manier High Availability verzekerd kan worden.

Bij het elimineren van single points of failure ondersteunt Postgres de volgende fysieke stand-by's:

1. **Cold Standby**: Dit is een back-up server die beschikt over back-ups en alle nodige WAL-bestanden voor herstel. WAL is de afkorting voor Write Ahead Log. Het logt elke transactie die uitgevoerd wordt op een database voordat het wordt uitgevoerd. Een Cold Standby systeem is geen operationeel systeem, maar het kan wel beschikbaar worden gemaakt als dat nodig is. Voornamelijk worden dan backup servers en WAL bestanden gebruikt voor het maken van een nieuwe PostgreSQL node als onderdeel van disaster recovery.

2. **Warm Standby**: Hierin draait Postgres in herstelmodus en ontvangt updates door gebruik te maken van gearchiveerde logbestanden of door gebruik te maken van log shipping replicatie van Postgres. Log shipping is een proces waarbij de back-up van transactielogbestanden op een primaire database wordt geautomatiseerd en vervolgens op een standby server wordt hersteld (https://docs.microsoft.com/en-us/sql/database-engine/log-shipping/about-log-shipping-sql-server?view=sql-server-ver15). In deze modus aanvaardt Postgres geen verbindingen en queries.

3. **Hot Standby**: Ook bij Hot Standby draait Postgres in herstelmodus en ontvangt het updates door gebruik te maken van gearchiveerde logbestanden of door gebruik te maken van log shipping van Postgres. Het verschil met Warm Standby is dat in deze herstelmodus Postgres hier wel verbindingen ondersteunt en read-only queries.

Bovenstaande voorbeelden zijn mogelijkheden die kunnen helpen bij het elimineren van

single points of failure. Afhankelijk van het overeengekomen niveau van beschikbaarheid, kunnen gebruikers voor een van de bovenstaande kiezen.

In geval van een volledige uitval van een systeem is geografische redundantie algemeen zeer wenselijk. Op deze manier worden servers verdeeld over meerdere locaties verdeeld over de wereld. Bij downtime door een natuurramp bijvoorbeeld zijn standby servers op meerdere fysieke (ongetroffen) locaties beschikbaar om in te vallen. Dit type van redundantie kan zeer duur uitdraaien, waarbij het een verstandige beslissing kan zijn om te kiezen voor een gehoste oplossing, waarbij de provider datacenters heeft over heel de wereld.

Voor betrouwbare cross-over biedt EDB een technologie genaamd EDB Postgres Failover Manager (EFM). Dit maakt automatische failover van de Postgres master node naar een standby node mogelijk in geval van een software- of hardwarefout op de master. EFM maakt gebruik van JGroups, die een betrouwbare, gedistribueerde en redundante infrastructuur biedt zonder een single point of failure. EDB Postgres Failover Manager kan ook gebruikt worden voor de detectie van storingen. Het bewaakt de server continu en zal storingen op verschillende niveaus detecteren. Het is ook capabel om om failover uit te voeren van de master node naar één van de replica nodes om het systeem beschikbaar te maken voor het accepteren van databaseverbindingen en queries. Wanneer EFM goed geconfigureerd is, kan het storingen detecteren en direct failover uitvoeren.

Verlies van service kunnen we in twee categoriëen opdelen. Geplande uitval of downtime en ongeplande uitval of downtime. Geplande downtime is vaak het gevolg van onderhoudsactiviteiten. Dit kan zijn door een softwarepatches die een herstart van het systeem of van de database vereist. In het algemeen is deze uitval niet onverwachts en zal deze uitval geen grootschalige gevolgen hebben. Een ongeplande downtime is vaak het resultaat van een of andere fysieke gebeurtenis, zoals hardware- of softwarestoring, of een anomalie in de omgeving. Stroomuitval, defecte CPU- of RAM-componenten (of eventueel andere hardwarecomponenten), netwerkstoringen, inbreuken op de beveiliging, of diverse defecten in toepassingen, middleware en besturingssystemen resulteren bijvoorbeeld in ongeplande uitval. In geval van (on)geplande downtime kan EFM helpen om de downtime zoveel mogelijk te minimaliseren. Voor een geplande downtime kan een gebruiker bijvoorbeeld eerst alle standby nodes patchen en EFM gebruiken om over te schakelen voordat de master node gepatcht wordt. Bij een ongeplande downtime kan EFM ervoor zorgen dat de storingen gedetecteerd worden en failover uitvoeren naar de juiste standby node, om dan deze node de nieuwe master node te maken. EFM zal na dit proces er ook voor zorgen dat de oude master node niet terugkomt om een split-brain situatie te voorkomen. Split-brain duidt op de inconsistenties in beschikbaarheid en data. Hierdoor ontstaan er twee afzonderlijke datasets met overlap. (https://www.enterprisedb.com/blog/what-does-database-high-availability-really-mean).

Databasereplicatie is het proces waarin we gegevens gaan kopiëren van een database naar één of meerdere replica's. Dit om de toegankelijkheid van gegevens en fouttolerantie te verbeteren. In de context van replicatie gebruikt men vaak ook de termen actief-actief en actief-passief. Raima Database Manager (RDM) ondersteunt beide technieken. Bij Raima wordt actieve replicatie gewoon replicatie genoemd en passieve replicatie spiegelen

(mirroring). Spiegelen zal resulteren in identieke replica's zoals de originele database, terwijl replicatie zal resulteren in replica's die niet identiek zijn aan de originele database. Deze replica's zullen alle records bevatten die van de originele database zijn overgebracht, maar de fysieke organisatie van de records in de databasebestanden (of in het geheugen) kan verschillen. Om terug te komen op de termen actief-actief en actief-passief zullen die vaker verwijzen naar andere concepten dan dewelke we juist omschreven hebben (replicatie en spiegelen). Actieve-actieve replicatie betekent replicatie in twee richtingen van gegevens tussen twee databases die beide actief worden bijgewerkt. Actieve-passieve replicatie betekent replicatie in één richting van een actief bijgewerkte master node naar een slave node die niet wordt bijgewerkt, behalve door het replicatieproces. Hier verwijst men soms ook naar master-slave replicatie. In RDM is replicatie altijd actief-passief (https://raima.com/rdme-high-availability-database/).

Load Balancing is ook een manier om High Availability te waarborgen. Het doel van een load balancer is om toepassingen en/of netwerkverkeer te verdelen over meerdere servers en componenten. Het zal binnenkomende verzoeken routeren naar verschillende servers. Hiermeer wil het optionele prestaties en betrouwbaarheid verbeteren. Enkele voorbeelden van load balancing is Round Robin die ervoor zorgt dat de verzoeken van de load balancer naar de eerste server gaan in de rij. De verzoeken gaan deze rij af, tot hij op het einde komt, waarna hij terug van het eerste element in de rij begint. Een tweede manier van load balancing is Least Connection. Hierbij zal er gekozen worden om gebruik te maken van de server met het minst aantal actieve verbindingen. Load balancers spelen een rol bij het tot stand brengen van een infrastructuur met High Availability, maar het hebben van een load balancer staat niet garantie voor het hebben van High Availability. Door redundantie te implementeren voor de load balancer zelf, kan deze geelimineerd worden als een single point of failure. (https://phoenixnap.com/blog/what-is-high-availability)

2.3 Cluster oplossingen

Hier nog bepaalde oplossingen beschrijven die ik in mijn voorstel heb staan, zoals Patroni, pgpool, pg pool II,...

Andere die ik ben tegengekomen in literatuurstudie al: Raima (RDM) EDB Postgres Failover Manager EMF

Kort Lijstje met clustering oplossingen voor postgres: 1. PgCluster 2. pgpool-I 3. Pgpool-II 4. slony 5. Bucardo 6. Londiste 7. Mammoth 8. rubyrep 9. pg_shard 10 pglogical 11. Postgres-XL 12. Citus

Van elk zal ik dan kijken of ze open source zijn (normaal gezien zijn ze dit allemaal al, maar het kan geen kwaad om dit nog eens op te zoeken), welke focus ze leggen...

2.4 Puppet

Puppet CTO Deepak Giridharagopal zei dat in het kielzog van de economische neergang als gevolg van de COVID-19 pandemie, meer IT-teams zwaar zullen moeten vertrouwen op automatisering. De meeste IT-teams zullen ofwel even groot blijven of worden ingekrompen. De IT-omgeving zal echter steeds complexer worden. De enige manier om IT-teams in staat te stellen meer te doen met minder is het automatiseren van meer routinetaken (https://devops.com/puppet-brings-orchestration-to-it-automation/).

Puppet is een cross-platform client-server gebaseerde toepassing die wordt gebruikt voor configuratiebeheer. Het behandelt de software en zijn configuraties op meerdere servers. Er zijn hierbij twee versies beschikbaar. De ene is open-source, de andere is een betalende, commerciële versie. Het werkt op zowel Linux als op Windows. Het gebruikt een declaratieve aanpak om updates, installaties en andere taken te automatiseren. De software kan systemen configureren met behulp van bestanden die manifesten worden genoemd. Een manifest bevat instructies voor een groep of type server(s) die wordt/worden beheerd(https://www.liquidweb.com/kb/what-is-puppet-and-what-role-does-it-play-in-devops/).

Wat is configuratiebeheer nu juist? Configuratiebeheer onderhoudt en bepaalt productkenmerken door fysieke en functionele attributen, ontwerp, vereisten en operationele informatie op te slaan gedurende de levenscyclus van een server.

Puppet maakt gebruik van de beschrijvende programmeertaal Ruby. Ruby is een dynamische, open source programmeertaal met de nadruk op eenvoud en productiviteit (https://www.ruby-lang.org/en/).

Vroeger werden software en systemen door systeembeheerders manueel opgezet en geconfigureerd. Maar toen het te beheren aantal servers snel toenam, moest er gezocht worden naar een manier om die processen te automatiseren, om dan zo tijd te besparen en de nauwkeurigheid te vergroten. Puppet is uit deze zoektocht ontstaan.

Puppet werkt aan de hand van een eenvoudig client/server architectuur workflow proces. Hierin bestaat er een master server die alle informatie bevat over de configuraties van de verschillende nodes aanwezig. Het slaat deze configuraties op in manifestbestanden op een een centrale server, genaamd de Puppet master, en voert deze manifesten uit op de remote client servers genaamd agents (https://www.liquidweb.com/kb/what-is-puppet-and-what-role-does-it-play-in-devops/).

3. Methodologie

1. Literatuurstudie

2. Opstelling schifting en keuze 2 oplossingen

3. Vergelijkende studie van 2 oplossingen waarin ik deze 2 oplossing volledig uitleg en waarom deze juist gekozen zijn geweest.

4. Keuze van 1 oplossing die ik zal voorleggen als ultieme oplossing, deze oplossing zal ook gebruikt worden voor de proof-of-concept

5. Proof-of-concept

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, consectetuer 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 consectetuer 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

Database high availability (HA) staat voor de garantie van het behouden van gegevens in geval er zich een defect of storing voordoet aan de databank server. Een storing aan een databank server kan te wijten zijn aan verschillende factoren. Voorbeelden hiervan zijn het verlies van netwerkconnectie en een defect in de software of hardware van de databankserver. Ook menselijke factoren en omgevingsfactoren moeten in rekening genomen worden. Voorbeelden hiervan zijn een menselijke vergissing en een wijziging in temperatuur. Investeren in high availability geeft meer zekerheid over de beschikbaarheid van data en biedt verschillende mogelijkheden voor failover en systeembescherming (**IBM1**). Met behulp van clusters kan er één actieve en een of meerdere standby instanties van de databank server zijn. Een cluster is een groep van servers en computers die samenwerken met elkaar alsof het één systeem is. Deze standby instanties zullen, in het ideale geval, dezelfde gegevens bevatten als de actieve server (**BDQ**). Wanneer dan een actieve server faalt, kan een standby instantie inspringen waardoor dataverlies en server downtime gereduceerd worden.

A.2 State of the art

Singer spreekt over high availability (HA) clustering als een groep van servers die applicaties en services ondersteunen die op een betrouwbare manier gebruikt kunnen worden met een minimaal aantal aantal downtimes. Hij bespreekt de cluster architectuur en wat de best practice is voor high availability (HA) binnen een cluster. De conclusie die hier getrokken wordt, is dat het primaire doel van een high availability (HA) systeem het voorkomen en elimineren van alle single points of failure zijn. Dit systeem moet beschikken over meerdere geteste actieplannen. Dit zodat ze in geval van storing, verstoring en defect in dienstverlening direct, gepast en onafhankelijk kunnen reageren. Zorgvuldige planning + betrouwbare implementatiemethoden + stabiele softwareplatforms + degelijke hardware-infrastructuur + vlotte technische operaties + voorzichtige managementdoelstellingen + consistente databeveiliging + voorspelbare redundantiesystemen + robuuste back-upoplossingen + meerdere herstelopties = 100% uptime (**Singer2020**). Ook Jevtic heeft het over veel van de punten die hierboven reeds zijn aangehaald. Jevtic spreekt over een highly available architectuur waarin meerdere componenten samenwerken om een ononderbroken service gedurende een bepaalde periode te garanderen. Dit omvat ook de reactietijd op verzoeken van gebruikers. Jevtic kenmerkt een highly available (HA) infrastructuur aan de hand van: 1. Hardware redundantie; 2. Software en applicatie redundantie; 3. Gegevens redundantie; 4. Elimineren van storingspunten (**Jevtic2018**). Akhtar heeft vier van de meest gebruikte database high availability (HA) oplossingen opgelijst. Deze vier zijn "PgPool-II", "PostgreSQL Automatic Failover (PAF)", "RepMgr [Replication Manager] ", en "Patroni"(**Akhtar2020**). Akhtar vergelijkt deze verschillende oplossingen kort met elkaar. Akhtar definieert high availability (HA) als niet alleen de continuïteit van een bepaalde service, maar volgens hem gaat high availability (HA) ook over het vermogen van een systeem om een (hogere) werkdruk te kunnen schalen en te beheren. Dit systeem moet volgens Akhtar de gemiddelde werkdruk, maar ook de piekmomenten aankunnen. Aldus Andersen zijn de top drie open-source databanken van 2019, in volgorde van top 3, MySQL met 31.7%, PostgreSQL met 13.4% en MongoDB met 12.2% (**Anderson2020**) van het totaal aantal open-source databank gebruikers.

A.3 Methodologie

In de eerste fase van het onderzoek zal er een vergelijkende studie gebeuren over de huidige, database high availability (HA) oplossingen. Deze verschillende tools/oplossingen zullen dan met elkaar vergeleken worden. Hierbij wordt gekeken naar welke elementen er allemaal (meermaals) voorkomen. Hiervan komt er een lijst die gebruikt zal worden om te schiften tussen de verschillende oplossingen. Op deze manier zal er dan een oplossing gekozen worden. Via deze methode wordt er gekeken om maximum 3 verschillende oplossingen uit te werken, waarbij één oplossing gebruikt zal worden bij de proof of concept. In de tweede fase van het onderzoek wordt de focus gelegd op het opzetten van de PostgreSQL (pgSQL) cluster als proof of concept. PostgreSQL is een open-source, object-relationeel databank systeem (**PostgreSQL2020**). Bij Inuits, een Belgisch open-source bedrijf met verschillende vestigingen in Europa, merken ze een stijging in de vraag

naar het PostgreSQL verhaal. Deze zal vooraf gegaan worden door een literatuurstudie over PostgreSQL (pgSQL). Aan de hand hiervan zal er gewerkt worden aan het opbouwen van de PostgreSQL (pgSQL) cluster. Vooraleer dit geautomatiseerd wordt, zal de opbouw manueel verlopen. De opbouw zal gebeuren via virtuele machines (VirtualBox) waarop Linux-distributies staan. In het onderzoek zal dus gebruik gemaakt worden van Linux-servers. De keuze van Linux-distributie zal onderbouwd worden in dit onderzoek. De opbouw van de cluster zal telkens grondig gedocumenteerd worden. Alle commando's zullen hierbij overlopen worden. Hierna zal er een inleidende literatuurstudie zijn over Puppet en zal er aansluitend via Puppet gewerkt worden om deze PostgreSQL (pgSQL) cluster te reproduceren. Ook hier zal alles grondig gedocumenteerd worden. Na het opzetten van de PostgreSQL (pgSQL) cluster zal er getest worden of de gebruikte database high availability (HA) oplossing functioneel is. Deze testen zullen uitgebreid beschreven worden.

A.4 Verwachte resultaten

Uit het onderzoek zal blijken dat verschillende database high availability (HA) oplossingen mogelijk zijn binnen een PostgreSQL (pgSQL) cluster. De opbouw van deze cluster zal gebeuren aan de hand van virtuele machines. Wanneer de virtuele machine, waarop de PostgreSQL server staat, uitvalt, zal er een standby instantie van deze server het werk van de actieve, uitgevallen server overnemen. Hierdoor zal er geen downtime of dataverlies zijn. De data zal beschikbaar en onverstoord blijven.

Uit dit onderzoek zullen ook best practices volgen die gehanteerd kunnen worden om op die manier het risico op verlies van gegevens te verminderen. De kans om offline te zijn zal lager liggen met een high available systeem.

Door gebruik te maken van Puppet zal de reproduceerbaarheid van de high available (HA) PostgreSQL (pgSQL) cluster zeer eenvoudig en snel moeten verlopen.

A.5 Verwachte conclusies

Uit het onderzoek zal blijken dat database high availability (HA) een blijvend topic is waar voldoende aandacht aan besteed moet worden in kleine en grote bedrijven. Zonder de implementatie van een high available (HA) architectuur kan een storing of downtime van de databank (SQL) server grote gevolgen hebben op een bedrijf/organisatie. Gevolgen zoals verlies van vertrouwen bij klanten, verlies van inkomen, verlies van informatie. Door middel van hardware-, software- en gegevensredundantie en het elimineren van mogelijke storingspunten zal er high availability gegarandeerd worden. De kost en tijd die geïnvesteerd moet worden in het onderhouden van een high available systeem zal lager liggen dan de kost en tijd die geïnvesteerd moet worden in geval van een downtime. Met de proof of concept wordt dan aangetoond dat database high availability (HA) eenvoudig te implementeren valt in een PostgreSQL (pgSQL) cluster.