Afstudeerverslag

Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low cost cloud oplossing te creëren bij i3D

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Opdrachtgever: i3D
Opdrachtgever bedrijf: Rick Sloot
Bedrijfsmentor: Rick Sloot
Plaats: Rotterdam
Datum: 19-01-2016
Versie: 1.0

Versiebeheer

Versie	Datum	Wijzigingen
0.1	19-01-2016	Layout opgezet
0.2	31-03-2016	Opmaak en spelling gecontroleerd, conceptversie voor school
0.3	28-04-2016	Onderdelen toegevoegd. Spelling en grammatica gecontroleerd. Tweede conceptversie voor school.
1.0	30-05-2016	Onderdelen toegevoegd. Spelling en grammatica gecontroleerd. Definitieve versie.

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

Voorwoord

Voor u treft u de scriptie met als onderwerp de afstudeeropdracht: “Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low cost cloud oplossing te creëren bij i3D”. Het is een onderzoek uitgevoerd in opdracht van i3D naar de mogelijkheid om winst te maken met een low-cost cloud oplossing op basis van OpenStack en verouderde hardware. Het onderzoek is uitgevoerd naar aanleiding van mijn afstuderen aan de Haagse Hogeschool te Delft.

De reden waarom ik de opdracht van de opdrachtgever heb aangenomen is voor een groot deel het werken met open source oplossingen. Ook het voorrecht om te mogen werken in een datacenter is natuurlijk meegenomen in de keuze.

Allereerst wil ik het gehele bedrijf i3D en in het bijzonder Rick Sloot bedanken dat mij deze opdracht is toevertrouwd, ook de ruimte die ik kreeg om de opdracht uit te voeren heb ik altijd als zeer prettig ervaren.
Rick Sloot, de opdrachtgever binnen het project, heeft mij altijd goed geholpen mochten er vragen zijn.

Tenslotte wil ik de volgende personen bedanken:

- Jan Dirk Schagen en Marinus Maris – voor de goede begeleiding en ondersteuning vanuit school.
- De collega's binnen i3D – voor een gezellige werkomgeving waar ik me direct thuis voelde.
- OpenStack community – Voor het beantwoorden van mijn vragen over OpenStack.
- Daniëlle van Eik – Voor ondersteuning en motivatie

Inhoudsopgave

Samenvatting

Dit document beschrijft de uitwerking en het proces van het project “Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low cost cloud oplossing te creëren bij i3D”. De opdracht is uitgevoerd door Jeroen Holtkamp, student aan de Haagse Hogeschool te Delft. De opdracht is, zoals ook beschreven in de titel, in opdracht van het bedrijf i3D, gevestigd te Rotterdam.
Het bedrijf i3D heeft beschikking over een eigen datacenter, en met een datacenter komt ook de beschikking over veel servers. Na een aantal jaar zijn deze servers “oud' en kunnen niet meer meedraaien in de core business van i3D, deze betreft namelijk game hosting, colocatie en hosting voor bedrijven. Voor aanvang van dit project waren er veel servers zo oud dat ze niet meer effectief en winstgevend ingezet konden worden in de huidige bedrijfsprocessen van i3D.
Op hetzelfde moment zijn er ook voorstellen gedaan om naast de huidige high end cloud op basis van Azure, de “Red cloud”, een low-cost cloud oplossing neer te zetten, dit zou makkelijker zijn voor klanten die minder geld overhebben voor hosting. Een eerder onderzoek naar een platform voor de low-cost oplossing wees uit dat OpenStack het aangewezen platform voor zo'n oplossing zou zijn.

Als opdracht voor dit project gold dat er een proof of concept moest worden opgezet die kon aantonen dat er winst met een low-cost cloud oplossing te maken zou zijn. Het proof of concept moest op basis van de oude hardware en OpenStack worden opgezet. Voor aanvang van het project werd er met de opdrachtgever afgesproken dat er ook naar Ceph gekeken zou worden. Ceph is een storage oplossing op basis van object storage.
In een literatuuronderzoek is eerst onderzoek gedaan naar OpenStack en Ceph en de werking van deze platforms. Uit dit onderzoek kwam naar voren dat OpenStack een Cloudplatform is die bestaat uit meerdere modulair opgebouwde componenten, de zogenaamde services. Er kunnen (virtuele of fysieke) nodes worden opgezet om één of meerdere OpenStack componenten te hosten. De OpenStack componenten communiceren met elkaar via RPC-calls op een messagequeue en via de REST-API waarmee de componenten elkaar commando's kunnen geven.
Ook kwam uit het onderzoek naar voren dat Ceph een storage-oplossing is die block storage, object storage en filesystem storage aan kan bieden terwijl al deze storage oplossingen object storage als platform hebben.
De oude stukken hardware waren aanwezig in het datacenter van i3D. Bij oplevering van deze hardware bleek echter dat ze allemaal verschillende waren uitgerust, door middel van een inventarisatie en optimalisatie zijn de servers klaar gemaakt voor het draaien van OpenStack.

Met de opdrachtgever van het project zijn systeemeisen overlegd waarvan de performance van OpenStack de belangrijkste waren, de uitwerking die OpenStack op de oude servers heeft was namelijk niet duidelijk.
Om de uitwerking van OpenStack op de oude servers te zien en de performance eisen te testen is in de architectuurfase van het project alvast een proefopstelling opgezet. Uit de performance tests kwam naar voren dat de servers OpenStack gemakkelijk aankonden. Eerder afgesproken plannen om de meest kritieke bottleneck in het proof of concept redundant uit te voeren zijn niet doorgegaan vanwege tijdsgebrek.

Op basis van de ontwerpen uit de infrastructuurfase van het project is het proof of concept opgezet. Ceph is niet opgezet als storage backend in het proof of concept met als reden dat het een zeer project betreft en OpenStack moet ook al opgezet worden. Naast Ceph heeft OpenStack ook goede storage-oplossingen, nl: Cinder block storage en Swift object storage. Met deze OpenStack-native storage componenten is het proof of concept opgezet.

In de testfase van het project zijn de systeemeisen die de opdrachtgever aan het project had getest op functionaliteit in het proof of concept. Er werd ook geëist van het systeem dat het winst zou moeten maken, een project moet namelijk zakelijk te verantwoorden zijn. Met een winstberekening is gemeten of er winst gemaakt zou kunnen worden met een low-cost cloud oplossing op basis van het proof of concept. De uitkomst van de winstberekening zag er op basis van een minimale productie-omgeving niet rooskleurig uit; er zou verlies gedraaid worden. Echter bij het groeien van de cloud naar een omvang die een heel rack in beslag zou nemen was er een kleine winst waar te nemen. Met het groeien van de cloud zou deze dus winstgevend kunnen worden.

Verklarende woordenlijst / symbolenlijst

Actor = Binnen OpenStack wordt met een actor een user of een usergroup bedoeld. Een role kan toegewezen worden aan (assigned to) een actor.

Broker (AMQP) = Een stuk software die het AMQP protocol mogelijk maakt. De broker accepteert verbindingen met clients (software die gebruik wil maken van de queue). De exchange en de queue maken deel uit van de broker.

Ceph monitor = De monitoring service houdt maps bij van verschillende onderdelen van Ceph. De status van het cluster wordt opgeslagen samen met een map van de OSD's, placement groups en voor het CRUSH algoritme.

Compute node = Een device in een OpenStack omgeving die zorgt voor: processorkracht (CPU), geheugen, netwerk en opslag om instanties te kunnen draaien.

Controller node = Een device in een OpenStack cloud omgeving die voor aansturing zorgt.

Core componenten = De onderdelen van OpenStack die de basis van de cloud vormen. Hier kunnen nog optionele componenten aan worden toegevoegd.

Consumer (AMQP) = Een applicatie die de berichten uit de queue ontvangt.

CRUSH (Ceph) = Een algoritme van Ceph die berekend hoe data het beste (efficiënste) opgeslagen en opgehaald kan worden op de data locaties. Het algoritme probeert SPOF, performance bottlenecks en limieten aan de schaalbaarheid te voorkomen.

DAS / Direct- Attached Storage = Een storage device die direct is aangesloten op het systeem, bijvoorbeeld een HDD die direct aangesloten zit via SATA.

Ephemeral storage = Een opslagsoort waarbij de opslag “verdwijnt” als de draaiende instantie wordt opgezegd.

Exchange (AMQP) = Onderdeel van de broker. Binnen de exchange worden messages ontvangen en doorgestuurd over de queue. Dit doorsturen wordt gedaan met verschillende routing manieren, nl: direct exchange, topic exchange, fanout exchange en headers exchange. De exchange is ook wel te vergelijken met een postbode.

Flavor = Een flavor binnen OpenStack is een blauwdruk van de virtuele resources die gekozen kunnen worden voor een instance.

Floating IP = Een IP-adres die “zweeft”. Binnen OpenStack kunnen deze floating IP-adressen op commando worden gekoppeld en ontkoppeld van instances en virtuele “hardware” zoals virtuele routers.

IOPS = Staat voor input output operations per second. Het is een meetpunt waarmee opslagmedia kunnen worden gemeten.

HT-technologie = Hyper threading: één core gedraagt zich als twee threads. Het besturingssysteem ziet een verdubbeling in losse processors. Zo wordt een quad-core processor voor een besturingssysteem gezien als acht losse processoren I.P.V. vier.

LDAP = Lightweight Directory Access Protocol. Met LDAP wordt beschreven hoe informatie binnen directory services benaderd kunnen worden.

Optionele componenten = De onderdelen van OpenStack die niet vereist zijn in de basis van de cloud. Ze kunnen wel de functionaliteit van OpenStack uitbreiden.

NIC = Network interface controller: een fysieke netwerk interface in een device.

NOC = Staat voor Network Operations Center. Het is een plek (in een bedrijf) waar het netwerk in de gaten wordt gehouden. Bij i3D is het ook een afdeling binnen het bedrijf.

Node = Een device die deel uitmaakt van een bepaald netwerk. In dit project wordt met een OpenStack node een fysieke server bedoeld.

OSD (Ceph) = Staat voor Object Storage Device. De term “OSD” is een fysieke of logische storage node. Echter wordt met “OSD” meestal de daemon bedoeld. De correcte term moet dan wel “Ceph OSD” zijn, en staat dan voor Object Storage Daemon. De daemon is verantwoordelijk voor de opslag van de objectdata en de toegang naar deze objecten.

Overcommit = De verhouding waarin er “oversell” voorkomt. Een fysieke core kan dus meerdere keren worden verhuurd. Dit geld tevens voor het RAM-geheugen. Als een systeem bijvoorbeeld 2 fysieke cores bevat zonder HT en de overcommit is 200% dan kunnen er 4 virtuele cores worden uitgedeeld aan klanten. Hier moet echter wel bij worden bedacht dat niet meer dan het maximum fysieke cores en fysiek aantal GB RAM aan één klant kan worden uitgedeeld.

Persistent storage = Een opslagsoort waarbij de opslag altijd toegankelijk (uitgaand van 100% availability) is. De draaiende instanties hebben geen effect op deze manier van opslag.

POSIX = Staat voor Portable Operating-System Interface. Een verzameling van standaarden die voor compatibiliteit (een eenheid) zorgen tussen verschillende programmeerinterfaces.

Publisher (AMQP) = Een applicatie die berichten over de queue stuurt.

Queue (AMQP) = Hier worden de messages gebufferd en overheen gestuurd naar clients. Het is te vergelijken met een brievenbus.

RADOS = Staat voor Reliable, Autonomic Distributed Object Store. Het Ceph storage cluster is opgezet volgens RADOS. RADOS bestaat uit Ceph OSD's en Ceph monitors.

Service account = Een “user” account waarmee de verschillende OpenStack services kunnen communiceren met Keystone.

Target = Domains en tenants kunnen binnen OpenStack allebei worden gezien als targets omdat aan beide roles kunnen worden toegewezen (assigned on).

Telemetry = Een proces waarbij data wordt verzameld van “moeilijk te bereiken” plekken. In OpenStack betekent dit dat de acties die gebruikers uitvoeren in de cloud worden verzameld en opgeslagen.

Tenant = Een “project” of groepering waarbinnen resources bestaan zoals: gebruikers, instances en (virtuele) netwerken. Een tenant is niet hetzelfde als een group.

1. Inleiding

i3D.net (de opdrachtgever) is een managed hosting provider die is gevestigd in Rotterdam. Klanten die worden bediend komen vooral uit de sectoren: gaming, overheid, onderwijs, gezondheidszorg, sport, web-shop,
hosting en print media. Om al deze hosting diensten uit te breiden is er een aantal jaar geleden besloten binnen i3D om een cloud platform op te zetten: de Red Cloud. Dit platform is gebaseerd op Microsoft Azure.
Ten tijde van het opzetten van de Red Cloud zijn er ook veel servers binnen het datacenter “afgeschreven”, deze servers konden door hun leeftijd niet meer worden ingezet voor de core business van i3D. Op dat moment is het idee opgekomen om naast de Red Cloud ook een ander cloud platform op te zetten op basis van open source software. In een voorgaand onderzoek is gebleken dat het cloudplatform “OpenStack” het beste gebruikt kon worden voor de intenties die i3D had met de alternatieve (low-cost) cloud oplossing.

In dit project is het de bedoeling dat er wordt aangetoond of en hoe OpenStack kan draaien op de oudere hardware die i3D niet meer in kan zetten voor core processen. Omdat i3D een onderneming is moet de oplossing ook winst maken, in een winstberekening moet dit worden aangetoond. Het eindproduct van het project moet een proof of concept van een low-cost cloud oplossing op basis van OpenStack worden.

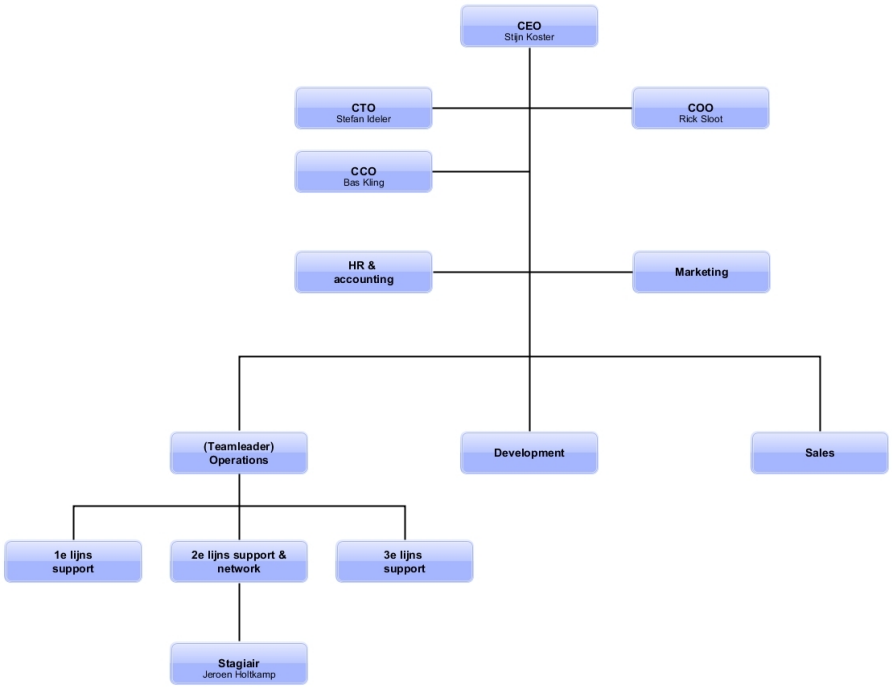
Om het doel van het project te bereiken is er onderzoek gedaan naar OpenStack en de hardware die i3D aanbiedt voor de opdracht. Naast OpenStack als onderwerp heeft de opdrachtgever ook de eis gesteld dat de storage-oplossing “Ceph” onderzocht zou worden.
Na het literatuuronderzoek is het proof of concept ook daadwerkelijk ontworpen door het maken van globale en gedetailleerde ontwerpen. Aan de hand van de ontwerpen is uiteindelijk het proof of concept opgezet.

Wel zijn er binnen het project zijn er een aantal veranderingen opgetreden t.o.v. het afstudeerplan.
In het plan stond een voorbeeld van een vraag die gesteld moest worden in het project, nl: “Kan de storage/dienst worden uitgebreid (en hiermee ook eventuele winst) met opensource software? En met welke software kan er eventueel worden uitgebreid?”.
Deze vraag is binnen dit project niet van toepassing en is door een communicatiefout in het plan terecht gekomen. De opdrachtgever wilde met het project weten of de dienst uitgebreid kon worden met Ceph. Er is dus geen onderzoek gedaan naar componenten waar OpenStack mee uitgebreid kan worden.

Een tweede verandering is de vraag over SDN, nl: “Kan een netwerk van de servers worden gebruikt als SDN? Hierbij moet wel worden gekeken of er nog aanvullende apparaten benodigd zijn als de mogelijkheid bestaat.”. Uiteraard is een onderzoek naar SDN met OpenStack zeer leerzaam en leuk. Echter bleek bij het begin van de stage dat dit geen vraag was die de opdrachtgever beantwoord wilde hebben. Het plan is geschreven aan de hand van het “sollicitatiegesprek” bij i3D. Hier zijn een aantal suggesties gegeven voor een project. De suggestie van SDN hoorde echter niet bij dit project.

2. Organisatie

i3D.net is een managed hosting provider die is gevestigd in Rotterdam. Klanten die worden bediend komen vooral uit de sectoren: gaming, overheid, onderwijs, gezondheidszorg, sport, web-shop, hosting en print media. Wereldwijd bestaat i3D uit 16 datacenters waar 31.000 klanten gebruik van maken. In het datacenter in Rotterdam zijn 40 mensen werkzaam.
Het bedrijf is relatief nieuw (2004). Ook de groei is explosief te noemen: i3D.net heeft al enkele prijzen gewonnen waaronder voor het snelstgroeiende winstgevende bedrijf in Zuid-Holland. Tevens is i3D.net al vier jaar op een rij opgenomen in de Deloitte Fast 500 EMEA, als snelgroeiend technologie bedrijf.



Afbeelding 1: Het organogram van i3D.

Het verloop van het project heb ik doorgebracht op de afdeling van de 1e en 2e lijns (game) support. Hier is de werkplek gevestigd. Op drukke dagen kan de support afdeling wel chaotisch zijn. Hierdoor kan het soms zijn dat de concentratie op het werk iets verminderd was. Over het algemeen was hier geen sprake van. Op de game support afdeling zijn ten alle tijden minimaal twee personen aanwezig. Gemiddeld zitten er 3 tot 5 personen op deze afdeling.
In de loop van het project is er een proef of concept opgezet. Deze was te benaderen via het netwerk zodat ik op mijn werkplek de opstelling kon instellen en configureren. Echter was het voor het fysieke gedeelte van de opbouw nodig om naar het datacenter te lopen. Dit datacenter is gelegen achter het “kantoor”.

In afbeelding 1 is te zien dat ik ingedeeld ben bij “network” of NOC, fysiek zit ik echter bij game support. Dit komt door het feit dat er bij NOC geen werkplekken meer over waren.

3. Opdrachtomschrijving

In dit hoofdstuk wordt het probleemdomein van de opdracht beschreven. Het probleemdomein is grotendeels in beeld gebracht door middel van een interview met de opdrachtgever. Dit interview is opgenomen in bijlage IV.

3.1 Probleemstelling

Naast de high end Red cloud op basis van Microsoft Azure heeft i3D ook behoefte aan een laag geprijsde cloud oplossing. Er zullen genoeg klanten zijn die wat minder eisen stellen aan hun hosting en/of er minder geld voor over hebben. Aan de andere kant zijn er veel oudere servers binnen i3D die op dit moment niet commercieel ingezet (kunnen) worden. Dat komt voornamelijk door het ontbreken van de juiste platforms voor deze servers. Ook ontbreekt de juiste toepassing nog. In een eerder onderzoek is er onderzocht wat voor open source cloud platform het beste zou passen binnen i3D. De conclusie van dit onderzoek was dat OpenStack de beste keuze was om te implementeren binnen i3D.

Als deze twee problemen en het voorgaande onderzoek bij elkaar op worden geteld dan komen de hoofdpunten waar dit project om draait tevoorschijn:

- I3D heeft naast alle high-end hosting oplossingen ook behoefte aan een low-end cloud oplossing op basis van OpenStack.
- De toegewezen oude servers moeten dienen als platform.
- Er is geen kennis van de eventuele winst die met zo'n soort cloud oplossing gemaakt kan worden.

De problemen hierbij zijn:

- De specificaties van de hardware zijn niet bekend.
- Er is niet genoeg kennis van OpenStack binnen i3D aanwezig.
- Er is geen kennis van Ceph aanwezig binnen i3D.
- Voor de afstudeerder is het interne netwerk van i3D niet bekend.
- Er is geen kennis van de eventuele winst die met zo'n soort cloud oplossing gemaakt kan worden.

Ook heeft de opdrachtgever graag dat specifiek wordt gekeken naar een storage oplossing genaamd: “Ceph”. Deze storage oplossing zou dan eventueel gebruikt kunnen worden in het proof of concept en in de uiteindelijke cloud oplossing.

3.2 Doelstelling

De doelstelling van de opdracht is het verwezenlijken van een IaaS proof of concept gebaseerd op het OpenStack platform. Het proof of concept moet met zo min mogelijk kosten worden opgezet. Bij elke keuze moeten de kosten in gedachten worden gehouden. Als hardware moeten de oude servers worden gebruikt die i3D aanlevert voor de opdracht.
Om te bewijzen dat de oplossing wel of niet winstgevend is zal er een winstberekening gemaakt moeten worden. Aan de hand van de winstberekening kan aan het eind van de opdracht worden afgelezen of er mogelijk winst zit in de oplossing.
Het proof of concept moet na het eindigen van het project reproduceerbaar zijn. Om hier aan te voldoen zal er een handleiding geschreven worden. Deze handleiding beschrijft alle acties die tot het proof of concept hebben geleid.

3.3 Resultaten

Als dit project met goed gevolg is afgerond dan is het duidelijk of er een low-cost cloud oplossing op basis van OpenStack en de oude servers van i3D opgezet kan worden. Op basis van de winstberekening die is gemaakt in het project is dan af te

leiden hoeveel winst er ongeveer gemaakt kan worden.
I3D kan aan de hand van deze resultaten en het proof of concept beslissen of zij de cloud oplossing op gaan zetten of niet. Als de cloud oplossing wel opgezet gaat worden op basis van de oude servers dan kan i3D veel geld besparen omdat ze dan geen nieuwe servers nodig hebben voor de oplossing en de oude servers kunnen weer worden ingezet met een duidelijk doel I.P.V. dat de servers worden verkocht tegen zeer lage prijzen.

4. Aanpak

In dit hoofdstuk wordt de aanpak beschreven waarmee het project is doorlopen. In het plan van aanpak (bijlage IX) staan alle onderdelen van de aanpak beschreven. In dit hoofdstuk zijn de onderzoeksvragen beschreven samen met de verantwoording van de projectmethode en de project planning.

4.1 Onderzoeksvragen

Hier onder worden de onderzoeksvragen beschreven. Uit de onderzoeksvragen kan de aanpak van het project worden afgeleid. De hoofdvraag die gesteld wordt binnen dit project is: “Is het mogelijk om een winstgevende low-cost cloud oplossing op te zetten op basis van OpenStack en de oude servers van i3D”.

4.1.1 Hardware specificaties

Dat de specificaties van de hardware niet bekend zijn is een cruciaal probleem. Dit probleem strekt zich uit van de interne hardware van de servers tot de bekabeling in het rack.
De volgende hardwarespecificaties moeten bekend worden: De bekabeling, de specificaties van de aangeleverde servers (CPU, max geheugen, etc), de specificaties van de switches en de aantallen van de verschillende stukken hardware. De vragen die gesteld moeten worden bij het onderzoek naar de aanwezige hardware is:

- Welke stukken hardware krijg ik opgeleverd?
- Welke specificaties hebben deze stukken hardware?
- Hoe zijn deze stukken hardware georganiseerd en gepositioneerd binnen i3D/het datacenter?

4.1.2 OpenStack kennis

Binnen i3D niet genoeg kennis aanwezig van OpenStack. Voor dit project heeft er een project gedraaid waarbij is gekeken welke cloud oplossing het beste bij i3D zou passen. Tevens is bij het voorgaande project een minimale opstelling op basis van OpenStack opgezet. De persoon die het voorgaande project heeft uitgevoerd geldt binnen mijn project als expert op het gebied van OpenStack.
Het is echter van groot belang dat OpenStack wordt onderzocht. Met dit onderzoek kunnen eisen die i3D stelt theoretisch worden getoetst aan de literatuur: misschien zijn sommige eisen onredelijk of helemaal niet haalbaar. Het is ook van belang duidelijkheid te verkrijgen over de werking van OpenStack. Tevens moet het jargon dat bij OpenStack hoort duidelijk zijn. De vragen die gesteld moeten worden bij het onderzoek naar OpenStack zijn:

- Wat is OpenStack?
  - Uit welke onderdelen bestaat OpenStack?
  - Waar dienen de onderdelen, waar OpenStack uit bestaat, voor?
- Hoe werkt OpenStack en hoe wordt het opgezet?
- Wat zijn de (systeem)eisen die OpenStack stelt aan de infrastructuur?
- Waar kunnen eventuele knelpunten (in de performance) voorkomen bij het groeien van de cloud?

4.1.3 Ceph kennis

Het storage platform Ceph is qua kennis ook nog niet bekend bij i3D. Het is net als OpenStack wel bekeken door i3D. De opdrachtgever vond Ceph zo veel potentie hebben dat daar ook een onderzoek naar uitgevoerd moet worden. In dit onderzoek moet ook worden gekeken naar de werking van de storage oplossing.

- Wat is Ceph?
- Hoe werkt Ceph?
- Wat zijn de (systeem)eisen die Ceph stelt aan de infrastructuur?
- Hoe is Ceph te koppelen met OpenStack?
- Is Ceph van toegevoegde waarde voor OpenStack binnen dit project?

4.1.4 Interne netwerk

Het interne netwerk van i3D is bij de afstudeerder niet bekend. Dit is echter wel van belang om te onderzoeken in wat voor omgeving het proof of concept zou moeten werken. Niet het gehele i3D netwerk hoeft duidelijk te zijn. Wat wel duidelijk moet zijn is:

- Hoe ziet de directe omgeving van het proof of concept er uit?
- Wat zijn de protocollen op het i3D netwerk waar het proof of concept mee te maken krijgt?
- Wat zijn de belangrijke overige faciliteiten op het i3D netwerk waar het proof of concept mee te maken krijgt?

4.1.5 Winst

De mogelijke winst die gemaakt kan worden met zo'n soort oplossing is niet bekend. Om servers commercieel in te zetten is het belangrijk om te weten of er ook winst gemaakt kan worden. Deze winst hangt van veel factoren of “variabelen” af. Sommige factoren kunnen pas effectief worden onderzocht op het moment dat het proof of concept functioneel is.
Met de winstberekening is er meer zekerheid over de levensvatbaarheid, een opvolgend project kan met meer zekerheid zakelijk gerechtvaardigd worden.

De vragen die hierbij gesteld worden zijn:

- Welke variabelen zijn van toepassing op het proof of concept?
- Kan er theoretisch winst worden gemaakt met deze oplossing?

4.1.6 Projectgrenzen

Binnen i3D is lang niet alle kennis van OpenStack aanwezig. Het is echter wel duidelijk dat OpenStack een zeer groot project is. Er kan dus nooit binnen de tijd van een aantal maanden een proof of concept worden opgezet die de gehele low-cost cloud oplossing verantwoord. In samenspraak met de opdrachtgever zijn er projectgrenzen gesteld aan de omvang van de opdracht. Oplossingen voor de eisen die i3D stelt zullen uiteraard wel worden verantwoord. Tevens worden er aanbevelingen gedaan mocht een oplossing niet (volledig) geïmplementeerd kunnen worden.

4.2 Verantwoording methode

Binnen een project moet er altijd volgens een methode worden gewerkt om het overzicht te behouden. In de eerste fase van het project is dan ook een methode gekozen. Eerst is er een aantal methodieken uitgekozen die mogelijk gebruikt zouden kunnen worden. Vervolgens zijn deze methodieken langs het project gelegd en gekeken wat het project nodig had.

Uit de voorselectie zijn de volgende methodieken gekomen:

- PRINCE2 [1]
- Waterval (overlappend: “sashimi”) [2]
- ASI [3]
- PMBok [4]

PRINCE2

PRINCE2 staat voor PRojects IN Controlled Environment, het is een procesmatige methodiek voor projectmanagement. Het is een projectmethode die in veel soorten omgevingen kan worden gebruikt, het is echter het meest van toepassing in ICT-projecten. PRINCE2 omvat naast projectmanagement ook het management van mensen en middelen. Er bestaan binnen een PRINCE2 project zeven principes waar ten alle tijden aan voldaan dient te worden. Als niet aan alle zeven principes wordt voldaan dan kan er niet worden gesproken van een PRINCE2 project. Er bestaan binnen PRINCE2 zeven thema's die uiteen worden gezet in zeven processen. De zeven processen lopen van project start-up (SU) tot project einde (CP).

Waterval/sashimi

De watervalmethode is een methode die is afgeleid van de cascades in een waterval, het is een lineaire ontwikkelmethodiek. Aan het eind van elke fase worden er vooraf gedefinieerde producten opgeleverd. In de originele watervalmethode zullen de fases en producten goed moeten worden gepland, want er kan niet meer worden teruggekeerd naar een eerdere fase zoals in een iteratieve methode. Het voordeel van deze aanpak is dat de voortgang goed gevolgd kan worden. De fasering in de watervalmethode betreft: definitie, ontwerp, gedetailleerd ontwerp, ontwikkeling, test, invoeren en onderhoud.
Om te voorkomen dat er een fout wordt ontdekt die is gemaakt in een voorgaande fase en vervolgens het gehele project over moet doen kan er een overlap worden gebruikt. De watervalmethode met een overlap ingebouwd heet het Sashimi model. Met de overlap van de fases kan er dus een fase terug worden gegaan bij het ontdekken van een fout.

ASI

ASI is een lineaire ontwikkelmethodiek voor technische infrastructuren. Binnen ASI worden vier fases onderkent: Definitiefase, Architectuurfase, Ontwerpfase en ontwikkelfase. Na het afsluiten van een fase onder ASI kan er niet worden teruggekeerd naar een afgesloten fase, net als in de watervalmethode. Wel is er sprake van iteratief gedag binnen de fases.

PMBok

PMBok staat voor Project Management Body of Knowledge, het is een procesmatige methodiek voor projectmanagement. PMBoK is een framework in het PMP (Project Management Professional) programma. Vooral in Amerika is het een veelgebruikte methode. Het is een framework met best practices. De best practices zijn opgemaakt aan de hand van vroegere projecten die gefaald of succesvol waren. Project managers kunnen binnen hun project kiezen welke best practices het meest geschikt zijn. Er worden echter geen activiteiten opgelegd of voorgesteld.
De PMBoK gaat uit van vijf processen: initiatie, planning, uitvoering, beheer en afsluiting. Elke fase van het project doorloopt deze processen. PMBoK wordt gebruikt bij projecten die zijn opgezet om een uniek product of dienst op te leveren. Het grote nadeel van PMBoK is dat het een framework is en niets voorschrijft. Als project manager moet je zelf de best practices uitzoeken en gebruiken. Dit vereist een hoog niveau aan kennis en ervaring vanaf het begin van het project.

Tabel 1: De kenmerken van verschillende methodieken zijn vergeleken

	Beschikbare literatuur	Toegankelijkheid	Flexibiliteit in de methode	Project toepassing	Schaalbaarheid	Totaal (M=25)
PRINCE2	4	2	4	3	4	17
Waterval/ sashimi	4	5	3	4	4	20
ASI	2	4	2	3	4	15
PMBok	4	2	4	2	3	15

In de vergelijking hierboven worden een aantal kenmerken tussen de methodieken vergeleken. De betekenis van de verschillende kenmerken is het volgende:

Beschikbare literatuur

Om in te lezen over de gebruikte methode zal er literatuur aanwezig moeten zijn die geraadpleegd kan worden. Bij PRINCE2 en PMBok is dit voornamelijk literatuur waar voor betaald moet worden. Uiteraard kunnen er boeken worden geleend bij de bibliotheek.
De waterval/sashimi methodiek is goed verantwoord op internet. Hier zijn veel bronnen over te vinden. Als laatste is de ASI-methode zeer slecht tot niet te vinden op het internet. Het ASI-rapport is de enige literatuur die er te vinden is.

Toegankelijkheid

De toegankelijkheid van een methode is hoe gemakkelijk deze methode eigen gemaakt kan worden. Uiteraard mag dit geen groot obstakel zijn. Echter wordt een project een stuk lastiger als het onderzoek naar de methodiek een lange tijd in beslag neemt.
De PRINCE2 methode is in Nederland binnen IT bedrijven vaak gebruikt. In een aantal stages is deze methode dus ook wel langs gekomen. Echter kost het inlezen in deze methode enige tijd, zeker omdat er geen PINO project van gemaakt mag worden.
De methodiek PMBok is van origine Amerikaans en in Nederland nauwelijks gebruikt in kleinere bedrijven. Tevens moet voor deze methode veel onderzoek worden verricht. Eigenlijk moet er veel verstand en ervaring opgebouwd zijn alvorens het te gebruiken. Dit is zeker zo omdat er zelf best practices moeten worden gekozen door de project managers.
De watervalmethode is zeer toegankelijk omdat er hard wordt voorgeschreven wat wanneer gedaan moet worden. Dit zelfde geldt tevens voor de ASI-methode.

Flexibiliteit in de methode

De flexibiliteit van de methode is in dit project nodig aangezien er een proof of concept wordt ontwikkeld. Uiteraard wordt voor het bouwen van het proof of concept alles goed onderzocht. Er kunnen echter altijd bij het bouwen of bij het testen fouten omhoog komen.
In de ASI methode zit het uitwerken en eventuele testen in één fase. Binnen de fase kan er altijd worden teruggevallen bij fouten. Als er echter fouten omhoogkomen die een fase eerder zijn gemaakt dan kan er niet terug worden gekeerd.
Bij de watervalmethode volgens het sashimimodel kan door de overlap in de fases wel worden teruggekeerd om eventuele fouten en wijzigingen aan te pakken.

Project toepassing

Uiteraard is het belangrijk dat de methode van toepassing is voor het project. Projectmethoden zoals PRINCE2 en PMBok zijn al snel van toepassing voor een project. Dit komt doordat er beschreven wordt hoe het project gemanaged moet worden en niet wat en hoe het product gemaakt moet worden.
De toepassing van de waterval methode is het ontwikkelen van een product, meestal software. Aangezien in dit project een proof of concept wordt ontwikkeld is deze methode van toepassing. Ook staan de eisen in het project vast en wordt het project uitgevoerd door één persoon.
De ASI-methode is een methode voor het ontwikkelen van een technische infrastructuur. Binnen dit project is wel gekozen om een netwerk/infrastructuur te ontwikkelen. De ontwikkeling van een technische infrastructuur binnen ASI loopt echter veel verder. Het beheer en support van de infrastructuur moet ook volledig worden besproken. Dit is echter geen onderdeel van het project. Tevens moet er binnen deze methode gezocht worden naar leveranciers van de middelen, maar deze zijn allemaal bekend. Tenslotte is het zo dat de ASI methode een watervalmethode is zonder overlap in de fases.

Schaalbaarheid

Met de schaalbaarheid van een methode wordt bedoeld hoe gemakkelijk deze methode afgeschaald kan worden. Aangezien het afstudeertraject alleen gedaan wordt en het projectteam klein is moet de methode hier ook op uitgerust zijn.
Zo zijn PRINCE2 en PMBok redelijk schaalbaar aangezien er gekozen kan worden voor een minimalistische aanpak. Echter is zelfs de minimalistische aanpak van deze methodieken een uitnodiging voor veel onnodige documentatie. Dit komt omdat het projectmethoden zijn, en in dit project zijn de enige leden de projectleider en de opdrachtgever.
De waterval- en ASI methoden zijn ontwikkelmethoden. Deze methodieken zijn meer gericht op het ontwikkelen van een

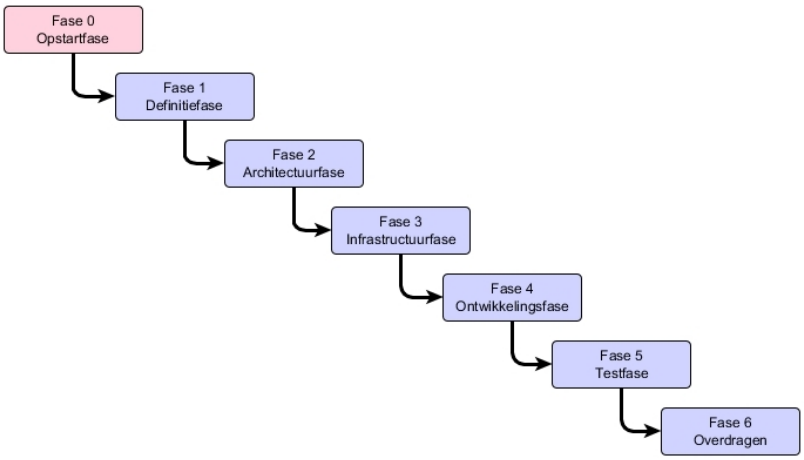
object. Uiteraard is er in deze methodieken projectbeheersing ingebouwd. Deze beheersing is echter wel zo dat er weinig tot geen onnodige documentatie gedaan hoeft te worden.

Methodiek keuze

Uiteindelijk is door het afwegen van verschillende voor- en nadelen van de projectmethodieken de waterval- of sashimimethode naar voren gekomen als beste keus. Uiteraard wordt in het project de waterval met overlap of sashimi gebruikt.

4.3 Fasering

In afbeelding 2 is te zien hoe in het project de fasering is ingedeeld. Toch wordt er gedeeltelijk afgeweken van de gebruikelijke indeling voor de watervalmethodiek. De reden van afwijken zit in de laatste twee fases van de watervalfasering. Dit zijn de uitrol en onderhoud van het product. In dit project is uitrol en onderhoud van het proof of concept echter niet van belang. Dit komt omdat er een proof of concept wordt aangeleverd: deze hoeft niet bedrijfsbreed te worden uitgerold en er hoeft ook geen verder onderhoud aan te worden gepleegd.



Afbeelding 2: De gebruikte fasering volgens de sashimimethode.

Aan het eind van elke fase zal er een faseovergang plaatsvinden. Een faseovergang betekent dat er een gesprek met de opdrachtgever gemaakt wordt. In dit gesprek wordt besproken welke producten zijn gemaakt, wat de volgende fase zal inhouden en wat de volgende fase zal opleveren. Het eind van de fase is dus een beslismoment (GO of NO GO). Bij een GO zal worden gestart met de volgende fase. Bij een NO GO zullen eerst de knelpunten in die fase verholpen moeten worden.

Dit project wordt uitgevoerd volgens de waterval/sashimimethode. Dit betekent dat er alles aan wordt gedaan om in een fase de producten juist op te leveren. Blijkt er echter in een vorige fase iets te zijn misgegaan is er nog de mogelijkheid om terug te keren om dit probleem aan te pakken en verder te gaan. Met deze oplossing hoeft niet het gehele project of een gehele fase over gedaan te worden.

Opstartfase

In de opstartfase worden de voorbereidende werkzaamheden uitgevoerd om het project opgestart te krijgen. Deze werkzaamheden bestaan uit het opzetten van de werkplek en het uitzoeken van een projectmethode.

Definitiefase

In de definitiefase moet het project gedefinieerd worden. Het gaat dan dus om het vormgeven en het helder krijgen van het project. In een literatuuronderzoek wordt OpenStack, Ceph en de beschikbare hardware onderzocht. Voor het plan van aanpak worden de problemen, doelstellingen, voorwaarden en andere onderwerpen uitgezocht om een duidelijk beeld te krijgen van het project.
Ook worden in deze fase de eisen achterhaald bij de opdrachtgever. Dit zal gebeuren in één van de wekelijkse gesprekken. Om beter te kunnen integreren in het bedrijf zal in deze fase ook een bedrijfsoriëntatie worden geschreven. Met deze oriëntatie kan een beter beeld gevormd worden van het bedrijf.

Architectuurfase

In de architectuurfase moet er een globaal ontwerp worden gemaakt voor het proof of concept. Om sommige kritische gedeeltes van OpenStack te kunnen testen zal in deze fase ook een proefopstelling worden opgezet die vooral bedoeld is om de performance van OpenStack op de hardware te onderzoeken. Er zullen performance tests nodig zijn om te zien of de eisen worden behaald met de architectuur. Aan de hand van deze tests kan er een architectuurontwerp worden gecreëerd waar alle eisen kunnen worden verantwoord. Na de test kan de meest kritieke bottleneck worden aangepakt in het ontwerp en later in de bouwfase.
Ook dienen deze performance tests om te zien of OpenStack überhaupt wel draait op de servers.
Aan het eind van de fase wordt het globale ontwerp besproken met de opdrachtgever. Op zijn beurt kan de opdrachtgever een GO of een NO GO geven voor de volgende fase.

Infrastructuurfase

In de infrastructuurfase zal het proof of concept gedetailleerd ontworpen moeten worden (infrastructuurontwerp). Het ontwerp moet in deze fase zo worden uitgewerkt dat er in de ontwikkelfase aan de hand van het ontwerp een werkend product opgezet kan worden.

Ontwikkelfase

In de ontwikkelfase wordt het proof of concept opgebouwd volgens de ontwerpen. Om het proof of concept later weer reproduceerbaar te maken zal hier ook een handleiding worden geschreven. Deze handleiding beschrijft precies wat er is gedaan om het proof of concept op te zetten.
Ook wordt in deze fase het testplan geschreven. In dit plan wordt uit de doeken gedaan op welke manier de systeemeisen getest moeten worden.

Testfase

In de testfase wordt het proof of concept op die punten getest die in de initiatiefase als systeemeisen waren opgegeven. Zo kan er achter worden gekomen of het systeem daadwerkelijk voldoet aan de vooraf gestelde eisen.
Ook wordt in de testfase aan de hand van de testresultaten een gedetailleerde winstberekening gemaakt. Met de uitkomst van de testen zijn er variabelen in te vullen in de winstberekening die eerder nog niet voorhanden waren.
In de testfase kunnen ook nog wijzigingen aan het systeem boven komen drijven. Als zich er problemen voordoen bij het testen kunnen deze opgelost worden door terug te gaan naar de ontwikkelfase. Als de problemen opgelost zijn kan er weer worden getest. De handleiding die is beschreven in de ontwikkelfase zal op dit moment ook aangepast moeten worden.

Overdragen

In de laatste fase van het project worden alle documenten en producten veilig overgedragen aan de opdrachtgever en de school.

4.3 Op te leveren producten

De producten die opgeleverd dienen te worden binnen het project zijn:

- Plan van aanpak
- Onderzoeksrapport
- Gedetailleerde winst- en kostenberekening
- Interviewverslag/meetingverslag

- Handleiding
- Globaal ontwerp (architectuur)
- Gedetailleerd ontwerp (infrastructuur)
- Testplan / testrapport
- Proof of concept

4.4. Risicoanalyse

Tabel 2: De risico"s van het project.

Risico	Kans	Ernst risico	Maatregel(en)
Collega / specialist heeft geen tijd.	Klein	Middel	Op tijd afspraken inplannen, een andere afspraak inplannen of een andere collega vragen.
Achterlopen op schema	Middel	Middel	Planning goed in de gaten houden, prioriteiten stellen, de planning aanpassen en de marge aanspreken.
Gebrek aan kennis	Middel	Klein	Genoeg kennis opdoen in de definitiefase, hulp inroepen van collega's.
Langdurige ziekte	Klein	Groot	Het project stopzetten en hervatten als daar de mogelijkheid toe is.
Te weinig informatie beschikbaar.	Klein	Middel	Een specialist raadplegen.
School geeft geen akkoord bij controle van de producten.	Klein	Middel	Samen met school en de opdrachtgever een oplossing bedenken.
Verlies van gegevens.	Middel	Middel	Een goede back-up policy instellen. Zo zal er gebruik worden gemaakt van Google drive en een memory stick als back-up
Verlies van proof of concept of wijzigingen aan proof of concept.	Klein	Groot	Elke dag in een log en de handleiding bijhouden wat er aan het proof of concept is gedaan. De activiteiten die verricht moeten worden om op dat punt te komen bijhouden.

4.5 Kwaliteitsborging

Vanuit i3D

Om kwaliteit binnen het project te waarborgen zijn er standaard meetings ingepland. Deze meetings vinden één of twee keer per twee weken plaats. In de meetings wordt vooral de voortgang, de beslissingen en problemen besproken. Bij problemen en vragen kunnen collega's of de opdrachtgever altijd om hulp worden gevraagd.
Tevens vind aan het eind van elke faseovergang een gesprek met de opdrachtgever plaats. In dit gesprek kan de opdrachtgever aangeven of hij de fase goedkeurt (GO) of niet (NO GO).

Vanuit school

Om de kwaliteit van het project te garanderen vinden er ook een aantal contactmomenten plaats tussen school en de afstudeerder. Eerst vind een bedrijfsbezoek plaats. In dit bedrijfsbezoek kan de afstudeerder nog enige vragen stellen. Een aantal weken daarna stuurt de afstudeerder een voortgangsverslag naar school. Hier kunnen opmerkingen en vragen van school uit door komen.
Ook kan er bij vragen contact worden opgenomen met de begeleiders van school.

4.6 Communicatie

Naar de examinatoren op school kan er communicatie plaatsvinden via de mail en via de telefoon. In eerste instantie wordt er alleen naar school gemaild voor de standaard afspraken. Uiteindelijk kan er ook nog contact plaatsvinden in "real life". Zo moet er een bedrijfsbezoek plaatsvinden door de eerste examinator. Dit bezoek is echter wel op initiatief van de afstudeerder. Ook de andere standaard contactmomenten vinden via de mail plaats.

De communicatie naar het bedrijf zal vooral plaatsvinden via de begeleider. Er is de mogelijkheid om één of twee keer per twee weken een meeting in te plannen. Hier kunnen vragen worden gesteld en overlegd worden over de opdracht. Tussendoor kunnen er ook contactmomenten plaatsvinden met de begeleider/opdrachtgever. Deze momenten zijn echter niet ingepland en er kan dus niet van uit worden gegaan dat er ook een contactmoment kan of zal plaatsvinden.
Binnen het bedrijf kan er altijd een vraag worden gesteld aan de andere medewerkers. Er loopt veel ervaring en expertise rond. De kans is dus aanwezig dat de vraag beantwoord kan worden.

4.7. Kosten en baten

Kosten:

Aan het project zijn voor i3D niet veel kosten verbonden. De kosten die wel bestaan zijn:

- Tijd voor begeleiding en meetings
- De bezette werkplek
- Een onkostenvergoeding

De tijd die Rick Sloot kwijt is aan meetings met de afstudeerder kunnen redelijk worden ingeschat. In de berekening wordt er van uit gegaan dat er elke week een meeting plaatsvindt. Ook zal in de berekening worden meegenomen dat de afstudeerder eerder is begonnen met de stage.
Een gemiddelde meeting is een uur lang. Er worden twintig meetings afgesproken. Als daar de korte vragen die tussendoor worden gesteld aan worden toegevoegd komt er nog zo'n twee uur bij.

*45 min * 20 meetings + 2 uur = 17 uur meeting.*

De bezette werkplek is een open plek. Hier zit dus niemand op het moment van de stage. Wel kost deze plek energie.

In de tijdberekening wordt de extra tijd apart opgeteld. De extra tijd is verkregen door eerder dan de uiterlijke begindatum aan het project te beginnen. De tijd dat de afstudeerder in het bedrijf werkzaam is:

*((8 uur per dag * 5 dagen per week) * 17 weken = 680 uur) + ((8 uur per dag * 5 dagen per week) * 4 weken = 160 uur) = 840 uur.*

Voor de gemaakte uren krijgt de afstudeerder een maandelijkse onkosten- en reisvergoeding.

Baten:

De baten die i3D terugkrijgt bij een succesvol project:

- Antwoord of en hoe OpenStack opgezet kan worden op oude servers.
- Theoretische winstberekening van de voorgestelde opstelling.
- Een afgerond documentonderzoek over OpenStack en afhankelijke onderdelen, hiermee kan de nodige kennis worden opgedaan door de lezers.
- Aanbevelingen over het verbeteren van het proof of concept.

4.4 Planning

In eerste instantie is een globale planning gemaakt voor het project. Bij het opzetten van het plan van aanpak is deze planning aangemaakt. In bijlage XIII is deze planning te zien. Door de initiatiefase heen is een gedetailleerde planning opgezet.

5. Systeemeisen

Binnen dit project is Rick Sloot de enige belanghebbende. Andere medewerkers hebben geen invloed op de eisen aan het proof of concept. Dit komt omdat i3D (met Rick Sloot als contactpunt) wil testen of OpenStack onder bepaalde voorwaarden winstgevend kan draaien.
Binnen een eventueel volgend project kan OpenStack wel opgezet worden als productiecloud. Hier zouden de andere medewerkers dan wel hun behoeften kunnen opgeven.

De behoeften voor het proof of concept zijn middels een semi-gestructureerd interview met Rick Sloot naar voren gekomen. Dit interview is opgenomen in bijlage IV. Er is gekozen voor een semi-gestructureerd interview omdat van tevoren nog niet duidelijk is welke vragen er gesteld moeten worden: een gestructureerd interview is dus niet makkelijk op te zetten, een open interview te wijd: Er moet wel een afkadering zijn voor het onderwerp en er zal enige sturing aan het interview gegeven moeten worden.
De behoeften die de belanghebbende aan het proof of concept heeft gesteld staan aangegeven in bijlage I.

5.1 Niet-functionele systeemeisen

Tabel 3: De systeemeisen binnen het project.

Categorie	Systeemeis
Availability	~~Redundantie moet worden aangebracht op de meest kritieke plek in het proof of concept.~~ (geschrapt)
Back-up	Er moet een back-up van gebruikersdata gemaakt kunnen worden binnen het proof of concept.
Back-up	Er moet een back-up gemaakt kunnen worden van de services/componenten binnen het proof of concept.
Back-up	Binnen het proof of concept moeten snapshots van de VM's gemaakt kunnen worden.
Disaster recovery	In het uiteindelijke proof of concept mag de storage bij schijfuitval (van 1 schijf in het proof of concept) geen dataverlies lijden.
Documentation	Na het project moet de opstelling/het proof of concept opnieuw opgezet kunnen worden.
Network	Het proof of concept moet een alternatief voor SNAT en DNAT hebben om "klanten" publieke IP-adressen op te laten vragen.
Network	Er moet een mogelijkheid zijn om externe IP-adressen te kunnen bestellen voor gebruik om instances.
Open source	Binnen het project moet er gebruik zijn gemaakt van OpenStack.
Overig	Er moet in het proof of concept ingelogd kunen worden op een control panel.
Overig	In het proof of concept met het verbruik van “klanten” gemeten kunnen worden.
Performance	Het netwerkverkeer van en naar de storage mag niet zo hoog worden dat de verbinding wordt dichtgetrokken. Tijdens normaal (uitgezonderd piekverkeer) gebruik mag het niet hoger worden dan 50% van de lijn.
Performance	Het netwerkverkeer over heel het netwerk mag onder normale omstandigheden (uitgezonderd piekverkeer) niet over 50% van de lijn komen.
Performance	In het proof of concept mag de slowdown op de VM's niet hoger worden dan 200% bij normaal gebruik (uitgezonderd piekverkeer).
Platform compatibility	Er moet op het proof of concept één soort operating system (één distributie) kunnen worden gehost.
Price	De uiteindelijke winst (of verlies) moet inzichtelijk worden.
Price	De uiteindelijke abonnementskosten voor klanten moeten lager zijn dan de primaire cloud oplossing (Red cloud).

Resource constraints	De apparatuur die zich in het rack bevindt moet in de testfase onder de 32 ampére stroomsterkte en 1gbit netwerktraffic blijven.
Scalability	Er moet in het proof of concept de mogelijkheid zijn om hun resources te kunnen schalen.
Scalability	Er moet moet in het proof of concept de mogelijkheid zijn om extra storage te kunnen "bestellen".
Security	In het proof of concept mogen onbevoegde verbindingen vanaf het i3D netwerk niet in een van de interne netwerken komen.
Security	De Linux installaties en de OpenStack componenten op de fysieke nodes moeten geüpdatet kunnen worden.

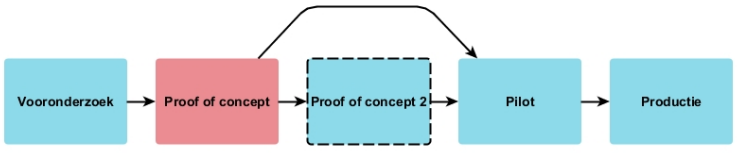
5.3 Grenzen aan het systeem

Niet het gehele systeem zal afgedekt kunnen worden met de systeemeisen. Dit komt omdat het gaat om een proof of concept waar sommige kanten van het systeem minder prioriteit hebben dan andere delen. Er moet dus een afkadering worden gebruikt. De onderstaande opsomming laat zien welke onderdelen minder prioriteiten hebben en daarom niet of nauwelijks zijn ingevoerd in het proof of concept.

De onderbelichte kanten in het proof of concept vanwege afkadering zijn:

- Availability: De availability is in een cloudomgeving van groot belang, echter vanwege afkadering van het project was er gekozen om alleen het meest kritieke punt (bottleneck) in de performance dubbel uit te voeren, en dus een hogere availability te geven.
- Capacity/capaciteit: Het is voor het proof of concept niet van belang om een capacity requirement door te voeren. De benodigde systeemcapaciteit (nodes) voor het proof of concept is al berekend. Een basale capaciteitsplanning zal worden gemaakt aan de hand van de testen op de proof of concept. Aan de hand van de performance moet de capaciteit (groei) worden ingecalculeerd. De capaciteit wordt meegenomen in de winstberekening. Ook is de capaciteit van rack(s), voeding, netwerkapparatuur, etc voldoende aanwezig voor het uitbreiden van het proof of concept.
- Extensibility/Uitbreidbaarheid: Het enige punt van uitbreidbaarheid dat duidelijk had moeten worden is de uitbreiding met Ceph. Ceph wordt echter niet gebruikt in dit proof of concept. De extensibility van OpenStack is al zeer hoog door de modulaire opbouw van de software.
- Support: Deze eis is in een proof of concept niet van belang.
- Live migratie van de VM's: dit is een belangrijke feature om te ondersteunen in een cloud omgeving. Echter in het proof of concept hoeft het niet ondersteund te worden.
- (Uitgebreide) Neutron functionaliteiten en SDN die Neutron kan aanbieden aan “klanten” hoeven niet te worden gebruikt.
- Efficiency: Er hoeft in het proof of concept niet te worden gekeken naar de efficiëntie. De efficiëntie zou een stuk verbeterd kunnen worden door het automatisch in-en uitschakelen van de servers. Dit in-en uitschakelen zou kunnen gebeuren door de interne ILO-port op de servers.
- Maintainability: De onderhoudbaarheid van het systeem is niet van belang in het proof of concept.

In afbeelding 3 is te zien hoe de verdere ontwikkeling van de low-cost cloud er uit zou kunnen zien: het rode vierkant is dit project.



Afbeelding 3: Het overzicht van de verschillende "versies" binnen de ontwikkeling. In dit project wordt het proof of concept gemaakt: in deze afbeelding weergegeven in het rood.

6. Definitie

Het literatuuronderzoek heeft een analyse/onderzoek document opgeleverd. Dit document is te vinden in bijlage VIII: “onderzoeksverslag”. In bijlage II staan de behoeften en eisen die zijn achterhaald bij de belanghebbende in het project.

6.1 OpenStack

OpenStack is een open source platform voor IaaS cloud computing. Met OpenStack is binnen een datacenter grote hoeveelheden rekeneenheden en opslag te managen. Dit alles kan grafisch aan worden gestuurd door een dashboard en door middel van commando's via de REST[5].
Het project is opgezet door Rackspace hosting en NASA[6]. Op dit moment wordt het project gemanaged door de OpenStack foundation. Achter OpenStack schuilt een zeer actieve en grote community. Tevens dragen veel grote ondernemingen bij aan de opbouw van OpenStack. Een aantal namen van bedrijven zijn: Intel, IBM, Redhat en Hewlett packard.

Op het moment van afstuderen is Liberty[7] de laatste stabiele release van OpenStack. Vandaar dat deze versie is gekozen om het proof of concept mee op te zetten.

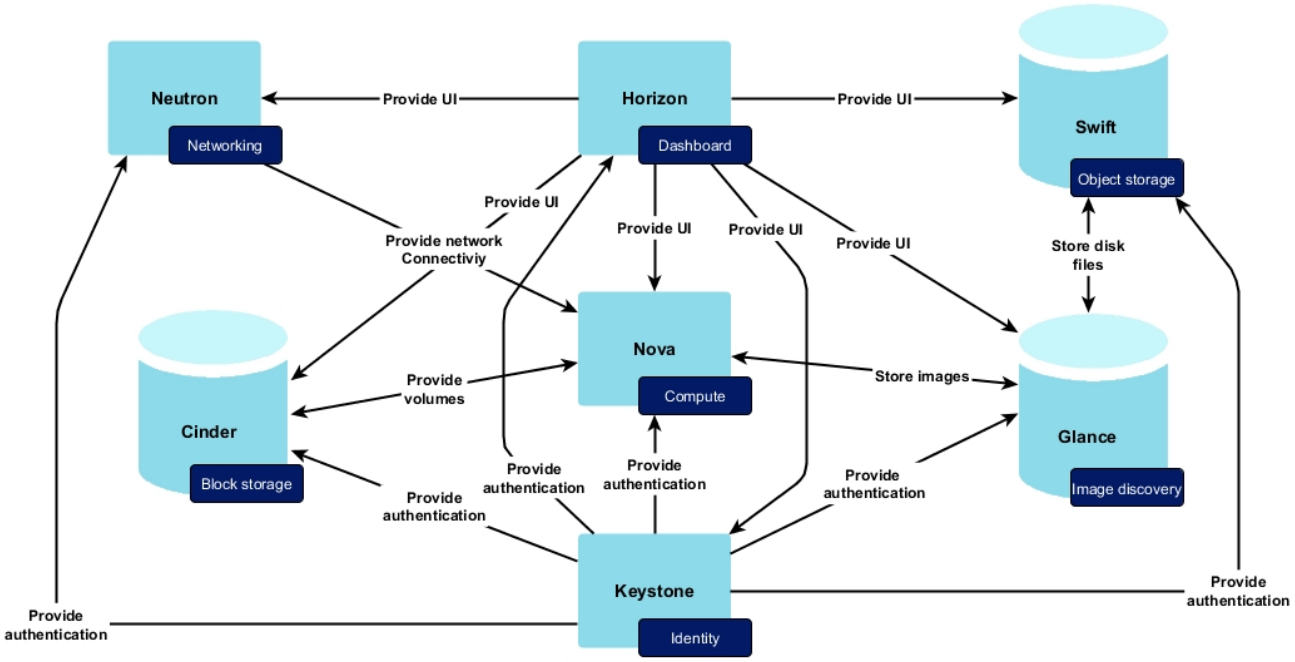
Afbeelding 4: Liberty: de twaalfde release van OpenStack.

Het platform bestaat uit een groot aantal verschillende onderdelen of componenten. Elk onderdeel kan worden opgezet om OpenStack uit te breiden en te beheren. Het betekent dat het niet noodzakelijk is om alle onderdelen op te zetten. De componenten die in vrijwel elke OpenStack omgeving belangrijk of onmisbaar zijn om te hebben draaien zijn de core componenten. In tabel 3 worden alle componenten (tot februari 2016) van OpenStack uitgelicht. In afbeelding 5 zijn tevens de OpenStack-componenten met hun onderlinge verhoudingen afgebeeld.

Tabel 3: De componenten binnen OpenStack en hun werking. De “core services” zijn dikgedrukt[8].

Naam component	**Functionaliteit**
Nova	Compute service: wordt gebruikt voor het hosten en onderhouden van de virtuele machines. Huisvest de hypervisor, het zorgt er voor dat gebruikers virtuele machines kunnen aanmaken.
Swift	Object storage: Het wordt gebruikt voor het opslaan en weer ophalen van dataobjecten. Swift object storage wordt gezien als betrouwbare redundante opslag.
Glance	Dit component wordt gebruikt voor de discovery, uploaden en downloaden van images en metadata. De gebruikers van OpenStack kunnen met deze services images die ze willen gebruiken in hun cloud discoveren, uploaden of downloaden. Op basis van deze images kunnen instances worden geboot op een compute node.
Horizon	Dashboard: Deze service verzorgd een web-based portal waarmee de verschillende componenten aangestuurd kunnen worden door admin users.
Keystone	Identity service: Dit is een centraal geregelde service waar alle users in staan opgeslagen. Het kan worden gebruikt bij authenticatie en autorisatie bij verschillende OpenStack services.
Neutron	Networking: Dit is het onderdeel die de cloud instances netwerkmogelijkheden geeft. Het is een component waarmee het netwerk rondom OpenStack software defined gemanaged kan worden. VLAN's, IP-adressen naar verschillende VM's en firewall management behoren onder andere tot de mogelijkheden.
Cinder	Block storage: Dit onderdeel verzorgt de persistent block storage voor de draaiende instanties.
Ceilometer	Meetservice: Met dit component kan het gebruik van de cloudservices worden gemeten. Met deze waarden kunnen de statistieken, benchmarks en eindafrekeningen worden bepaald.
Heat	Maakt OpenStack automatisch schaalbaar.
Trove	Database as a service: Met deze service kunnen klanten een relationele database gebruiken binnen OpenStack.

Sahara	Met deze service kunnen makkelijker data intensieve programma's bovenop OpenStack worden uitgevoerd.
Ironic	Met dit component kunnen bare metal machines worden gehost in plaats van virtuele machines.
Searchlight	Deze service zorgt voor indexing en zoekmogelijkheden door de OpenStack resources.
Designate	DNS as a Service.
Zaqar	Messaging service
Barbican	Security service: Met deze service kan opslag worden beveiligd, secrets (bv. passwords) worden gemanaged, sleutels worden gemanaged en certificaten worden gemanaged.
Manila	Met deze service kunnen persistent file-shares worden gemanaged



Afbeelding 5: De "core componenten" en hun verhoudingen. Horizon is geen core component maar wel benodigd[9].

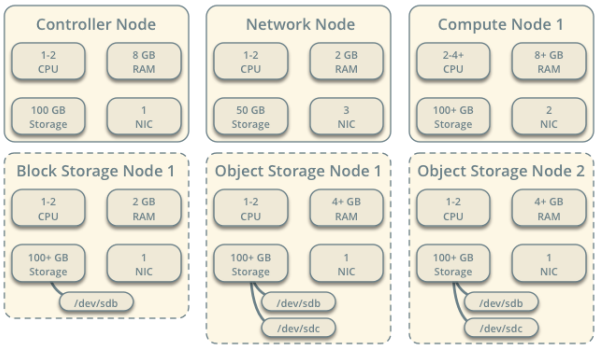
6.1.1 OpenStack nodes (servers)

Binnen een OpenStack opstelling zijn een aantal verschillende soorten nodes te onderscheiden. Nodes zijn de hosts in een OpenStack cloud die verschillende services onderbrengen.
De nodes zijn: controller node, compute node, storage node (block storage, object storage) en network node[10].
Tevens kunnen ondersteunende diensten worden ondergebracht op één of meerdere extra nodes. Voorbeelden van extra diensten kunnen zijn: NTP, SSH jumpbox, ect. In afbeelding 6 is een overzicht van de minimale hardware om OpenStack normaal te draaien afgebeeld.

Volgens de medewerkers van OpenStack is de minimale OpenStack installatie: "*The simplest architecture you can build upon for Compute has a single cloud controller and multiple compute nodes*[11]". Deze minimale opstelling is echter niet geschikt om in te zetten als productiecloud.

Controller node: Herbergt het merendeel van de services.
De controller bevat de componenten: Horizon, Keystone, Glance, Ceilometer. Tevens kan de controller node de volgende services bevatten: NTP-service, message queue en de databases voor de services.
De activiteiten van de controller node(s) liggen vooral bij de doorvoer en afhandeling van data die bestemd is voor

verschillende services. Op deze nodes is per service een database actief die informatie over users en instances bijhoudt. Een message queue zorgt er voor dat de OpenStack services via remote procedure calls met elkaar kunnen communiceren, deze message queue is ook onderdeel van de controller[13].
Bij uitbreiding van het aantal users en/of van het aantal services is het vooral van belang dat de server een goede processor met veel kernen heeft en een goede netwerk doorvoer. De goede netwerk doorvoer is van belang omdat de controller veel verschillende services en calls moet afhandelen[14].
De controller node is echter een node die samengestelde services bevat. OpenStack raad aan om bij de opbouw van een proof of concept te starten met een controller node: "*A cloud controller is just a conceptual simplification. In the real world, you design an architecture for your cloud controller that enables high availability so that if any node fails, another can take over the required tasks. In reality, cloud controller tasks are spread out across more than a single node.*"[15] Na de opbouw van het proof of concept zou als een volgend project de services op de controller node verspreid kunnen worden over meerdere andere nodes: de controller node is dus niet voor elk project een vereiste.



Afbeelding 6: De minimale hardware die nodig is om OpenStack te kunnen draaien[12].

Compute node: Deze nodes verzorgen de hypervisors en zorgt daarmee dat virtuele machines kunnen worden geboot. Één compute node heeft ook één hypervisor, er kunnen meerdere VM's of instances draaien op één compute node. Ook kunnen de nodes worden gezien als CPU- en memorypools. Een compute node kan lokale storage aanbieden aan instances. Ze bevatten een networking agent die de virtuele machines kan koppelen met virtuele netwerken[16].
De compute nodes hebben over het algemeen de hoogste eisen aan hardware, deze hoge eisen komen van het virtueel draaien van instances. De load op de compute node wordt bepaald door de manier dat de instances geschaald zijn, de overcommit van een compute node en de activiteit van de users op die node[17].

Block storage node: Block storage nodes kunnen binnen OpenStack worden gebruikt voor het toewijzen van nieuwe opslag aan draaiende instances. OpenStack gebruikers kunnen op deze manier makkelijk een nieuw volume toewijzen aan de draaiende virtuele machine[17].

Object storage/proxy node: De object storage nodes kunnen worden gebruikt voor het opslaan van over het algemeen grotere files, zoals: Images, back-ups en archieven als objecten[18]. Bij een object storage oplossing vormen servers een storage backend waarbij data als objecten opgeslagen kunnen worden[19].
Bij block storage en object storage is het aan te raden om na te denken over een dedicated datalijn tussen compute en storage. Opslag kan veel verkeer genereren en het is meestal niet wenselijk dat andere gebruikers hier last van hebben. Ook hebben beide opslag methoden ruime hoeveelheden aan storage capaciteit nodig als er geen gebruik wordt gemaakt van een backend[17].

Network node: Als er gebruik wordt gemaakt van een network node dan betekent dit automatisch dat het OpenStack Neutron component wordt gebruikt. Tenants kunnen zelf virtuele netwerk resources regelen. De tenants kunnen private netwerken met eigen IP-adressen instellen binnen hun IaaS. Het neutron component kent in de Liberty versie van OpenStack ook al vele plug-ins. Met deze plug-ins is het mogelijk dat de tenants zelf QoS, firewalls, VPN's en intrusion detection instellen[20].

Een OpenStack omgeving kan ook functioneren zonder network node. Binnen zo'n instelling zullen de netwerk functionaliteiten minimaal zijn en is er ook geen sprake van gevirtualiseerde netwerkonderdelen voor tenants. De netwerk functionaliteiten worden dan verzorgt door de compute nodes zelf. De service wordt legacy of Nova networking genoemd[20].

Network design: Vanuit OpenStack wordt aangeraden om minimaal de volgende netwerken te implementeren[21]: 1. Een publiek netwerk (internetverbinding), 2. Een verbinding tussen de compute nodes en de network nodes zodat de klanten kunnen verbinden (tunnel), 3. Een verbinding tussen de controllers en alle nodes (dus ook de network nodes), hiermee kan onder andere de messagequeue communiceren en kunnen de fysieke nodes worden gemanaged.

6.2 Operating system

OpenStack kan worden geïnstalleerd op vrijwel alle Linux distributies, en zelfs op Windows. Bij veel operating systems zullen de packages die nodig zijn voor OpenStack zelf moeten worden gecompileerd. Nu zijn er ook een aantal distributies die standaard de OpenStack (repositories) packages verstrekken[22].
De operating systems die OpenStack zelf aanbeveelt en die ook allemaal OpenStack packages verstrekken zijn: Ubuntu 14.04 LTS, Red Hat Enterprise Linux 7, CentOS 7, SUSE Linux Enterprise Server 12 en Debian 8[22].

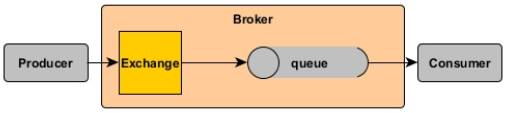
Bij de software selectie moet er rekening worden gehouden met de ondersteunde hypervisors in de gebruikte distributie. De verschillende hypervisors worden door OpenStack in groepen ingedeeld. Aan de groepen kan worden afgeleid hoe intens de hypervisor drivers worden getest in combinatie met OpenStack.
OpenStack onderscheidt twee groepen: groep A en groep B. Groep A: deze drivers worden volledig getest en ondersteund. De enige driver die staat ingedeeld in deze groep is voor KVM en QEMU.
Groep B: deze drivers worden niet volledig getest. De hypervisor drivers die OpenStack in deze groep heeft ingedeeld zijn voor: XenServer, vSphere en Hyper-V[23].

6.3 Database

Binnen OpenStack wordt veel gebruik gemaakt van databases. Nu ondersteunt OpenStack veel verschillende soorten databases. De meest gebruikte databases die OpenStack ondersteunt zijn: MariaDB, MySQL en PostgreSQL[24].

6.4 Message queue

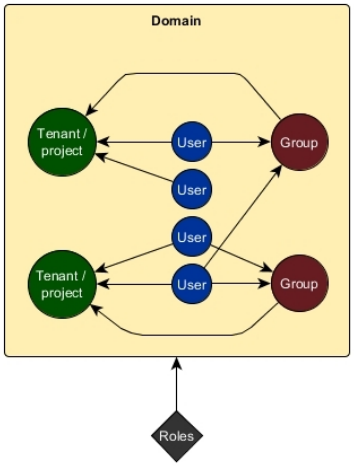
In OpenStack is een messagequeue vereist om de cloud te laten functioneren. Een messagequeue kan worden gezien als een stuk middleware die het mogelijk maakt om verschillende processen, applicaties of systemen met elkaar te kunnen laten praten ongeacht het ontwerp van de processen, applicaties en systemen, dit gebeurd via remote procedure calls[25].



Afbeelding 7: De werking van AMQP[28].

Binnen OpenStack wordt een messagequeue gebruikt om de verschillende componenten met elkaar te laten communiceren. Omdat de componenten zijn geschreven door verschillende programmeurs is de messagequeue nodig om via RPC's de componenten van verschillende programmeurs toch met elkaar te laten communiceren. Tevens is het zo dat door het gebruik van een messagequeue OpenStack modulair wordt[25].

OpenStack gebruikt RabbitMQ als broker om het Advanced Message Queuing Protocol (AMQP) te implementeren[26]. AMQP is een protocol die het gemixt gebruik van verschillende messaging systems mogelijk maakt. Hiermee wordt de interoperabiliteit van de message queue vergroot.
Het AMQP werkt met brokers, dit zijn zogezegd de message “servers”[27]. In afbeelding 7 is de werking van de messagequeue te zien.



Afbeelding 8: Roles zijn globaal uniek. Tenants, users en groups zijn onderdeel van een domain.

6.5 OpenStack identity (Keystone)

Aangezien een cloud een “verzamelplaats” van users en data is heeft OpenStack een grote behoefte aan een identity service. Het component die de identity service mogelijk maakt onder OpenStack heet

Keystone.
Keystone zorgt voor authenticatie, autorisatie en een service catalogus. De andere services die OpenStack aanbiedt moeten samenwerken met Keystone om authenticatie en autorisatie mogelijk te maken[28], dit betekend dat op het moment dat een user een request stuurt naar een OpenStack service dit component eerst bij Keystone moet controleren of de gebruiker deze request mag uitvoeren.

6.5.1 Entiteiten

Binnen OpenStack/Keystone bestaan de entiteiten users, groups, tentants (ofwel projects), roles en domains[29]. Users zijn net als in andere omgevingen digitale representaties van personen, systemen of services. Deze users kunnen verschillende roles aannemen.
Een role zorgt voor het toewijzen van rechten en privileges aan users. Met deze rechten kunnen users specifieke taken uitvoeren.
Een tentant is te vergelijken met een groep of een project. Het wordt gebruikt om resources en users van elkaar te scheiden. Een tenant is te vergelijken met een bedrijfsonderdeel.
Een domain is te vergelijken met een onderneming. Tussen verschillende domains kunnen de users en user groups elkaar ook niet zien. Tevens kunnen er in verschillende domains ook dezelfde namen worden gebruikt voor users, groups en tenants. Formeel gezien is een domain een collectie van specifieke users, groups en tenants[29]. Een overzicht van de OpenStack entiteiten is te zien op afbeelding 8.

6.6 OpenStack image service (Glance)

De Glance service is vanaf de tweede officiële OpenStack release aanwezig. Het component zorgt voor discovery van images, registratie van images en het ophalen van images[30]. Met Glance kan een user dus een snapshot van een virtual machine of een schijf maken, en vervolgens zal Glance deze opslaan in de bij Glance geregistreerde backend. Ook kunnen er instances worden geboot van de opgeslagen images in de Glance-store.

6.7 OpenStack compute (Nova)

OpenStack Nova kan worden gezien als de core van OpenStack. Dit component zorgt voor toegang tot (compute) resources. Het component houdt grote aantallen VM's bij. Tevens managed de Nova service de draaiende VM's[30].

6.8 OpenStack networking (Neutron)

Met Neutron kan er gebruik worden gemaakt van virtuele netwerken en hebben users ook de mogelijkheid tot het gebruik van uitgebreide services zoals LbaaS (load balancing), VPN en FwaaS (Firewall)[31]. De Neutron service beheerd ook de IP-adressen binnen OpenStack.
Met Neutron bestaat iedere VM en tenant in een virtueel netwerk. Dit virtuele netwerk is een layer 2 netwerk(segment), vergelijkbaar met VLAN's bij fysieke netwerken. Ieder segment is dus een broadcast domain. Binnen de virtuele netwerken bestaan subnetten (lokale range, class A, B of C). Elk virtueel netwerksegment heeft een subnet toegewezen gekregen. Elk VM heeft een (virtuele) port. Met deze port worden VM's in een subnet gehangen. Een port kan worden gezien als een aansluitpunt aan een virtuele switch. De VM wordt aan de port aangesloten en de port zit aangesloten aan het virtuele netwerk(segment)[31].

6.9 OpenStack block storage (Cinder)

Het OpenStack project biedt storage aan in verschillende vormen. Zo zijn er ephemeral, file level, block en object storage. De

block storage wordt aangeboden door het Cinder project binnen OpenStack. Bij block storage worden er blocks aangeboden aan de klanten/VM´s. Een block wordt door de OpenStack VM's gezien als een storage device (een extra harde schijf) of direct attached storage. Een block kan op hetzelfde moment ook maar aan één instantie zijn gekoppeld[32].
De verbinding tussen Nova en Cinder kan op basis van iSCSI, Fibrechannel of network file system zijn. Het backend van Cinder kan uit veel verschillende formaten bestaan, waaronder: LVM en NFS.[33]

6.10 OpenStack object storage (Swift)

OpenStack heeft een “native” component voor object storage, nl: Swift. Swift is een component die schaalbaarheid en redundantie uitstraalt. Het object storage systeem is zo opgezet dat het voor lange tijd grote BLOB's data (objecten) kan opslaan. Deze objecten zijn redundant opgeslagen binnen de nodes. Mocht er een node uitvallen dan repliceert Swift de data van de uitgevallen node naar een nieuwe node[32].

Swift kan gebruikt worden als “reliable storage”, zo kan Cinder back-ups maken met Swift als backend en kunnen users grote bestanden uploaden naar Swift.[32] De bestanden binnen Swift worden gezien als object, deze objecten hebben allemaal een uniek ID. Aan de hand van dit ID kan een user zijn object weer ophalen of verwijderen. De object storage wordt aangesproken met HTTP-commands, zoals: GET en PUT[34].

Binnen Swift bestaat er een hiërarchie boven de objecten, nl: Accounts → Containers → Objects[34].
Accounts zijn de bovenste laag van hiërarchie. Een Swift account komt overeen met de betreffende tenant in OpenStack. Onder de accounts komen containers, containers kunnen worden gezien als “namespaces” binnen een Swift account. Er kunnen “oneindig veel” containers worden aangemaakt binnen een account. Binnen de containers kunnen ook ACL's worden aangemaakt om toegang tot objecten te ontzeggen.
Tenslotte komen er onder de containers de objecten. Binnen de standaard instellingen van OpenStack is de maximale grootte van een object 5GB.
Bij de “directory” indeling binnen Swift komt deze hiërarchie ook naar voren, de opbouw van een path is: /[Swift versie]/[account]/[container]/[object].

6.11 OpenStack Benchmarking

Rally is een benchmarking tool binnen OpenStack. Het kan realistische load genereren om zo een actieve cloud te simuleren. Met deze load kan performance en scalability worden getest[35].
Als pakket is Rally eigenlijk alleen een “schil” bovenop Tempest: de officiele test suite van OpenStack. Echter door de complexiteit van Tempest werd dit niet vaak gebruikt. Met Rally als schil is Tempest veel beter “bestuurbaar”[35]. Met Rally kan OpenStack gedeployed, geverifieerd en gebenchmarkt worden.

6.12 OpenStack telemetry

Bij cloud diensten is het van belang dat verleende services kunnen worden gemeten. Verleende service (In bijvoorbeeld CPU-kracht/aantal cores of tijd) zijn immers de onderdelen die geld in het laatje brengen.
Binnen OpenStack zijn er verschillende projecten die elk een gedeelte van telemetry op zich nemen. Zo is er Ceilometer: Dit project verzorgt het meten van het verbruikt binnen de verschillende OpenStack componenten[36]. Hierna is er nog het AODH-project. Dit project is een uitbreiding op Ceilometer: het zorgt voor een “alarm” op het moment dat er aan vooraf gestelde criteria zijn voldaan. Een voorbeeld kan zijn dat een gebruiker over zijn verbruik heen gaat[36].

6.13 Ceph

Bij aanvang van het project is er expliciet gevraagd om samen met OpenStack ook Ceph te onderzoeken. Ceph is een

storage backend voor cloud oplossingen. Eigenlijk is het een alles in één storage oplossing, Ceph is namelijk zo opgebouwd dat het block storage, object storage en file system storage bij elkaar in één store kan aanbieden.
Echter op de overgang van de definitiefase en de architectuurfase is in samenspraak met de opdrachtgever overeen gekomen dat Ceph in dit proof of concept niet gebruikt gaat worden. De keuze hiervoor is gemaakt vanwege de grootte van het Ceph project, het is een project op zich om Ceph op te zetten.
In de definitiefase is Ceph wel onderzocht. Uit dit onderzoek is gebleken dat Ceph een veelbelovend platform is voor i3D. Dit komt omdat alle storage oplossingen in één ondergebracht kunnen worden. Ook zorgt Ceph er zelf voor dat bottlenecks niet of zo min mogelijk voorkomen.

In hoofdstuk 3 van het onderzoeksverslag (bijlage VIII) kan het verslag van het onderzoek naar Ceph worden gevonden. Ook staat in het onderzoeksverslag de vergelijking tussen de OpenStack storage oplossingen en Ceph.

6.13.1 Verschil OpenStack storage oplossingen en Ceph storage

Het opzetten van de combinatie OpenStack en Ceph wordt goed ondersteund en zelf aangemoedigd door beide partijen[37]. Tevens wordt deze setup in veel bronnen besproken.
Volgens de developers van Ceph is er een hoge availability voor de storage te behalen en het is te gebruiken op low-end systemen[38].
Het nadeel zou echter wel zijn dat bij het groeien van de OpenStack cloud ook het gebruik van Ceph groeit, en hiermee ook de load. Het is geen rare gedachte dat na een korte tijd meer performance vereist is van de oude servers. Als de servers dan al maximaal geüpgraded zijn zou er over moeten worden gestapt op andere servers. Dit houd in dat er een grote investering vereist is om performance drops tegen te gaan.

Aan de andere kant biedt OpenStack ook oplossingen voor storage, nl: Swift en Cinder. Als deze oplossingen worden gecombineerd ontstaat er ook een krachtige storage oplossing. Zo kan Swift binnen een Openstack cloud bijvoorbeeld worden gebruikt voor het opslaan van grote datasets zoals: snapshots en Cinder volume back-ups.
De Cinder service zorgt voor toewijzen van block storage aan de OpenStack instances en kan instances zelfs laten booten van block volumes.
De Cinder- en Swift componenten kunnen beide ook “lokaal” worden opgezet. Bij swift betekend lokaal opzetten dat er een LVM backend wordt gebruikt. Bij Swift betekend lokaal opzetten dat er geen 3th-party backend wordt gebruikt, maar een lokaal storage network wordt opgezet.

Tabel 4: Vergelijking tussen verschillende storage componenten. Binnen OpenStack worden Swift en Cinder meestal samen gebruikt.

Storage component	Object storage	Block storage	File level storage	Store snapshots	Store Cinder volume back-ups
Ceph	**V**	**V**	**V**	**V**	**V**
Swift	**V**			**V**	**V**
Cinder		**V**			
Local file system			**V**	**V**	

Ceph en OpenStack Cinder kunnen niet worden vergeleken. Dit komt door het feit dat Cinder een besturende eenheid is die met een interne storage backend (LVM) werkt of een externe 3th-party backend nodig heeft om te functioneren, Ceph is daarentegen een storage backend.

OpenStack Swift en Ceph kunnen echter wel worden vergeleken. Het eerste punt wat opvalt is performance. Uit een performance vergelijking blijkt dat Swift last kan krijgen van een bottleneck: de proxy server. In Swift wordt minimaal één proxy server gebruikt om de data te verdelen over de nodes. Veel load zal dus komen te liggen op deze proxy server.
Bij Ceph is het zo geregeld dat het CRUSH algoritme bottlenecks probeert te voorkomen met slimme load distibution[38]. Hierbij moet wel worden bedacht dat bij het groeien Ceph al snel meer resources nodig heeft, services zoals OSD, monitoring en CRUSH moeten de omgeving constant in de gaten houden.

Ceph maakt gebruik van OSD-daemons om opslag aan te sturen, Ceph raad aan om per fysieke schijf één OSD daemon in te stellen[39]. Swift maakt gebruik van een “ring”. Deze ring is bij iedere node bekend en op die manier weten de proxy nodes waar de verschillende object storage nodes en volumes te vinden zijn.

Het aantal servers dat nodig is voor een minimale productieomgeving is ook verschillend tussen Swift en Ceph. Bij Ceph wordt er aangeraden om binnen een minimaal productiecluster negen fysieke servers neer te zetten, de berekening is hierbij: 3 monitoring nodes + 5 storage nodes + 1 RGW (RADOS GateWay)[39].
Bij Swift wordt een minimum aantal servers van zes aangeraden, de berekening hierbij is als volgt: 1 object proxy server + 5 object storage servers[40]. Hier moet wel een kanttekening worden geplaatst dat er bij Swift minimaal nog een node bij moet, namelijk de proxy servers die redundant moet worden uitgevoerd, Als de proxy node uitvalt dan kan niemand meer bij de Swift service.

Als het storage cluster over meerdere locaties wordt verdeeld kan Ceph volgens Christian Huebner zeer ernstige nadelen vertonen:

“...Ceph writes only synchronously and requires a quorum of writes to return successfully. With those issues in mind, let’s imagine a cluster with two regions, separated by a thousand miles, 100ms latency, and a fairly slow network connection. Let’s further imagine we are writing two copies into the local region and two more to the remote region. Now the quorum of our four copies is three, which means the write request is not going to return before at least one remote copy is written. It also means that even a small write will be delayed by 0.2 seconds, and larger writes are going to be seriously hampered by the throughput restriction.
On the other hand, Swift in the same two-region architecture will be able to write locally first and then replicate to the remote region over a period of time due to the eventual consistency design. Swift also requires a write quorum, but the write_affinity setting can configure the cluster to force a quorum of writes to the local region, so after the local writes are finished the write returns a success status.”[41]

Als de Ceph uit meerdere regions gaat bestaan kan de consistency een probleem gaan vormen. Daar tegenover kan Swift het af met eventual consistency bij een setup over meerdere regions.

Volgens Christian Huebner kan binnen Ceph de dataveiligheid in het gedrang komen omdat de RADOS clients op hetzelfde netwerk draaien als het niet geëncrypte replicatieverkeer[41].

Uiteraard is het van belang bij de keuze tussen de storage oplossingen om te kijken naar de use case: een low-cost cloud oplossing op oude servers. De toekomste klanten die de OpenStack cloud service aanvragen weten bij het tekenen van het SLA dat de service op sommige punten kwalitatief minder kan zijn dan de high-end Red cloud binnen i3D. Zo kan de performance minder zijn omdat er wordt gewerkt met 1Gb/s lijnen (I.P.V. 10Gb/s bij de Red cloud) en met minder krachtige servers. De scalability van de storage zal ook niet zo hoog zijn als in de Red cloud omdat de servers maar vier HDD's kunnen huisvesten.
Omdat er in dit project gebruik gemaakt moet worden van servers met lagere performance zal er gekozen worden voor OpenStack Swift. De redenen hiervoor zijn:

- Om een minimale productieomgeving op te zetten met Swift zijn er minder servers nodig dan met Ceph, nl: 7 tegen 9.
- Ceph wordt al snel “hongerig naar resources”, dit komt door de services die Ceph als een consistent bottleneck voorkomend geheel laten werken. Deze services moeten constant de omgeving in de gaten houden. Bij swift is dit minder het geval, veel load komt bij de object proxy servers te liggen. Als de performance te laag wordt kunnen dus eerst de proxy servers geüpgraded worden.
- De eventual consistency kan een voordeel worden als er gebruik gemaakt gaat worden van meerdere regions binnen de OpenStack cloud, nl: betere performance, er hoeft geen reply te worden afgewacht.

6.14 Hardware inventarisatie

Om het proof of concept op te zetten is oude hardware opgeleverd die niet meer werd gebruikt in het datacenter.
De initiële hardware oplevering voor mijn opdracht bestaat uit de volgende stukken hardware:

- 44 HP Proliant DL120 G6
- 2 Dell Powerconnect 5548
- Server rack
- 250GB harde schijven

Er was een inventarisatie nodig omdat alle servers verschillende setups hadden. Tevens is het volgens de OpenStack organisatie van belang voor een zo foutloos mogelijke opbouw om de hardware zo gelijk mogelijk te houden. Veel servers waren zelfs niet uitgerust geheugen of harde schijven.

6.14.1 Servers

Het proof of concept dient te worden opgezet met de HP Proliant DL120 G6. In de oplevering zijn er 44 van deze servers aan mijn project toegewezen. Alle servers zijn met de inventarisatie geoptimaliseerd, dit betekent dat de servers uitgerust zijn met 8 GB RAM en een harde schijf van 250GB. De processor die in de server zat bleef behouden.
De nummers van de servers die opgeleverd zijn voor het proof of concept zitten in een range van oplopende nummers, nl: Server 1986 ~ Server 2000 en Server 4404 ~ Server 4405.

6.14.2 Processors

Al deze servers bevatte een Intel Xeon X3430. Deze processor is een non-HT 4 cores processor met 2,4Ghz en Intel VT-x virtualisatie acceleratie.

Bij het virtualiseren op een Nova compute node kunnen fysieke processor cores worden geprovisioneerd. Hoe meer cores een server bevat hoe meer er uitgedeeld (en overgeprovisioneerd) kunnen worden. De X3430 processor bevat vier cores zonder HT-technologie (dubbele threads). Het besturingssysteem op deze nodes ziet dus vier losse processors.
Met Intel VT-x bieden de processors hardwarematige ondersteuning bij virtualisatie. Er is met VT-x een redelijke snelheidswinst te behalen over volledig softwarematig virtualiseren[42].

6.14.3 Hard disks

De DL 120 G6 servers hebben zelf vier drive bays aan de voorkant van het device[43]. Hier kunnen 3,5 inch harde schijven in worden gefaciliteerd.
Het uitrusten van de servers is gedaan in de inventarisatie.
Na de inventarisatie is gebleken dat er al 28 harde schijven met een capaciteit van 250 GB in het rack aanwezig waren. Als er meer harde schijven nodig waren dan zou dit gevraagd kunnen worden aan de magazijnbeheerder. Er mochten zo veel harde schijven worden gebruikt als nodig. Alle harde schijven hebben een SATA 300 interface en een rotatiesnelheid van 7200RPM.

6.14.4 RAM

Omdat het bij een cloud omgeving draait om virtualisatie is de RAM-grootte belangrijk. De maximale grootte voor een DL120 server die volledig is uitgerust met unbuffered DIMM's is 8GB (4 x 2GB). Met Buffered DIMM's kan er tot 16GB worden gebruikt[43], deze zijn echter niet beschikbaar, daardoor zijn de servers met 8GB RAM uitgerust.

6.14.5 Switches

Om het proof of concept op te zetten is er één switch opgeleverd. Deze switch is van het type Dell Powerconnect 5548. De switch bezit 48 poorten met gigabit snelheid (1000BASE-T).
De switch is managed en ondersteund alle belangrijke niet-proprietary protocollen, zoals: IEEE 802.1Q, STP en IGMP. Tevens kan met het IEEE 802.1p protocol op elke port prioriteit worden aangegeven. Hiermee kan de QoS worden gemanaged[44]. Als blijkt dat één switch te weinig ruimte biedt kan een tweede switch worden geïnstalleerd. Er moet echter in eerste instantie van uit worden gegaan dat er een enkele switch is.

Afbeelding 9: Een Dell Powerconnect 5548

6.14.6 Rack

De hardware was bij de oplevering allemaal gemount in een dedicated rack. Dit rack is aanwezig in het datacenter van i3D. Specificaties van de racks zijn onder andere:

- Geïntegreerde koeling van de vloer naar boven.
- Slot op de rackdeur.
- 32 ampére stroomsterkte kan worden geleverd.
- 1 Gb netwerksnelheid.

6.14.7 Het interne netwerk van i3D

Binnen dit project is het ook van belang dat het interne netwerk van i3D is onderzocht. Om een OpenStack installatie te laten draaien moet er namelijk ook duidelijk zijn hoe het directe netwerk er uitziet waar de installatie in komt. Naar het interne netwerk van i3D is wel onderzoek gedaan, echter zal dit niet uitgebreid worden gerapporteerd aangezien hier gevoelige informatie bij zit. Het volledige interne netwerk is ook verder niet heel belangrijk om in het verslag te belichten. Wel worden hieronder de punten beschreven waar op moet worden gelet bij het opzetten van het OpenStack proof of concept.

- Voor het project is een range van globale IPv4-adressen gereserveerd.
- Binnen i3D worden VLAN's gebruikt binnen het netwerk. Dit betekent dat het OpenStack project één of meer eigen VLAN's krijgt. Er is in ieder geval één VLAN die het project aansluit op het openbare netwerk van i3D (toegang naar “buiten”).
- Op het fysieke vlak lopen er in de toegewezen suite naar elk rack een 10Gb-verbinding via “top-of-rack” verbindingen.

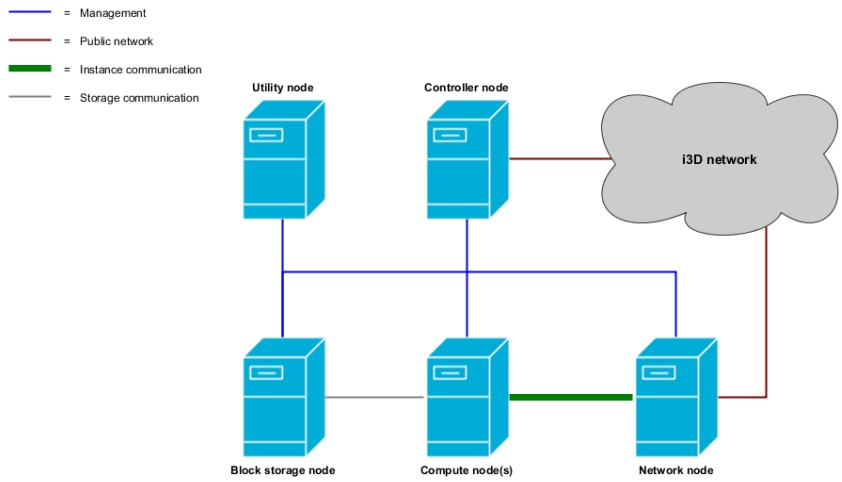
7. Architectuur

In dit hoofdstuk wordt het proces beschreven van de architectuurfase. In het bijlage X: “architectuurontwerp” is het volledige proces uit deze fase beschreven. In dit ontwerp staat het ontwerp en de manieren hoe is voldaan aan de gestelde eisen. Tevens is er een proefopstelling opgezet in deze fase. Met de proefopstelling kon worden gekeken hoe de performance was van OpenStack op de oude servers (in een basis PoC opstelling: geen dubbele nodes). Na de creatie van de proefopstelling kon naast de performance ook direct de schaalbaarheid worden getest tot op zeker niveau. Aan de hand van de uitkomst kon een architectuur worden ontworpen.

7.1 Proefopstelling opzetten

In de architectuurfase moeten de gestelde systeemeisen kunnen worden verantwoord. Dit is de reden dat er is gekozen om de performance te testen op een proefopstellen alvorens het architectuurontwerp te maken, de performance is namelijk een kritiek punt binnen de architectuur. Zeker omdat er wordt gewerkt met oudere hardware is het testen van de performance voordat het architectuurontwerp wordt gemaakt van belang. Zonder deze proefopstelling is er geen enkel beeld over hoe een minimale OpenStack installatie zich houdt op de hardware.
Het was in het begin van de architectuurfase ook de bedoeling om naast de performance ook duidelijkheid te scheppen over de meest kritieke bottleneck binnen de opstelling: deze zou redundant uitgevoerd moeten worden in het proof of concept. Om uit te vinden waar zich de meest kritieke bottleneck bevond had dit ook moeten worden getest in de architectuurfase, zo zou de redundante oplossing voor de bottleneck worden meegenomen in de ontwerpen.
Echter door tijdgebrek (zie subhoofdstuk 6.4) is het inbouwen van redundantie op de plek van de meest kritieke bottleneck geschrapt. De tests op de proefopstelling is echter wel blijven staat zodat ik nog de performance van de machines kon testen alvorens de ontwerpen te maken.

In deze proefopstelling en het proof of concept zal er voor gekozen worden om alle OpenStack nodes op fysieke servers zonder virtualisatielaag te installeren, dit omwille van de lagere performance van de systemen en het aantal systemen die opgeleverd worden.



Afbeelding 10: De proefopstelling waarbij de controller- en compute nodes opgezet worden.

Er is gekozen voor een opstelling zoals die te zien is op afbeelding 10. De keuze voor deze opstelling is gemaakt omdat dit volgens OpenStack de minimale proof of concept opstelling is.
Het probleem met een proefopstelling opbouwen voor OpenStack is dat veel componenten in elkaar steken en van elkaar afhankelijk zijn. Om een werkende geheel te krijgen, zelfs voor een proefopstelling, zal de OpenStack opstelling zoals op afbeelding 20 moeten worden opgebouwd. Dit is de minimale opstelling van OpenStack als er niet wordt gekeken naar one- of two-server setups.

De block storage node hoeft niet op te worden genomen in een minimale opstelling, dit is in de proefopstelling echter wel gedaan omdat de performance-eisen getest worden, hier is de block storage node bij nodig. Voor de utility node geldt ook dat hij niet vereist is in een minimale OpenStack omgeving, deze node wordt echter gebruikt om OpenStack Rally (de testsuite) te draaien.

- Controller node
- Compute node
- Network node
- Block storage node
- Utility node

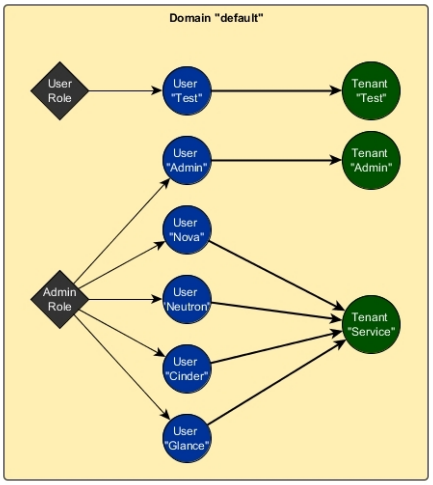
Deze proefopstelling is opgezet aan de hand van de Liberty OpenStack guide voor Ubuntu 14.04. Dit is een beschrijving hoe een basis opstelling van OpenStack opgezet zou kunnen worden. Alle handelingen die zijn gedaan om aan de proefopstelling te komen zijn ook direct vastgelegd in de handleiding. Deze handleiding is te vinden in bijlage VI.

7.1.1 Passwords

De passwords die worden gebruikt in het proof of concept zijn opgeslagen in Keepass. Het genereren van de wachtwoorden is gebeurd door het uitvoeren van het volgende command:

< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-20};echo;

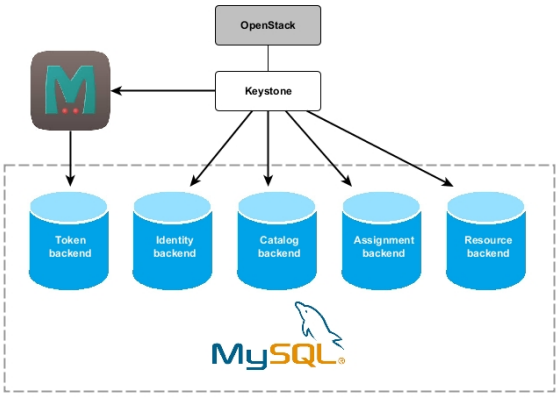
De lengte van de passwords wordt voor dit proof of concept gehouden op 20 karakters. Deze lengte is gekozen omdat volgens NIST na het twintigste karakter de entropie veel minder toeneemt[40]. Om de entropie nog groter te maken zijn er letters, hoofdletters en getallen gebruikt bij het genereren van de wachtwoorden



Afbeelding 11: De entiteiten die aangemaakt zijn in de proefopstelling.

7.1.2 Keystone

De controller node wordt eerst uitgerust met de keystone service. De Keystone service is bedoeld voor de authenticatie, autorisatie en service catalogus binnen OpenStack. De entiteiten die aangemaakt zijn binnen Keystone zijn te zien op afbeelding 11.
Deze service regelt ook het identity management van de OpenStack services. De Keystone service is echter ook vrij belangrijk bij de performance van OpenStack: er kan een vertraging optreden als Keystone niet snel genoeg de aanvragen kan behandelen. Om dit “probleem” voor te zijn wordt er gebruik gemaakt van “memcached”.



Afbeelding 12: De backends die zijn gebruikt binnen Keystone. Het token backend gebruikt Memcached als caching.

Memcached is een stuk software dat een cache in het geheugen van een node maakt. Deze cache zorgt voor het opvangen van de load naar een database. Memcached is bedoeld om dynamisch web verkeer te versnellen, dit is dus ideaal voor Keystone. De grootte van de cache is vooraf ingesteld. De data die door memcached gaat wordt afgehandeld met rotatie. Als er een nieuw stuk data in de cache wordt gezet als deze al vol is dan zal de oudste data worden overschreven. De data wordt opgeslagen in zogenaamde “slabs”. Deze slabs kunnen variëren in grootte.

Voor dit proof of concept is als Keystone identity backend de SQL-database gekozen. Vooral door het feit dat het eindproduct van dit project een proof of concept is. Tevens bestaat er nog geen LDAP server (en bijkomende active directory). Op afbeelding 12 staan de verschillende gebruikte Keystone backends met hun gekozen platform afgebeeld.

7.1.3 Glance

Als tweede wordt de image service genaamd “Glance” geïnstalleerd, deze zorgt voor registratie van de images binnen OpenStack. De service weet waar de images staan en zorgt ook voor het transport van en naar compute nodes.

De Glance service heeft een storage backend nodig om de images in op te slaan. Als backend kan een Swift cluster worden gekozen of een backend van Swift (zoals Ceph). In de proefopstelling wordt Swift echter nog niet opgezet omdat geen vereist gedeelte is in OpenStack.
Bij het ontbreken van de object storage is er gebruik gemaakt van interne Glance storage. De interne storage voor Glance is de directory */var/lib/glance/images* op de controller node.

7.1.4 Nova

Als derde service wordt de Nova compute service geïnstalleerd. Deze service houdt de draaiende images bij en huisvest ook de hypervisor.

Als hypervisor binnen OpenStack kunnen verschillende varianten worden gekozen. Er is binnen deze opstelling gekozen voor KVM aangezien deze hypervisor het beste wordt getest bij OpenStack (zie subhoofdstuk 5.2). Naast KVM had er ook gekozen kunnen worden voor QEMU, beide worden aangestuurd door de “libvirt” virtualisatie API.
De QEMU hypervisor is beter om te gebruiken I.C.M. oudere hardware die geen hardwarematige virtualisatie ondersteunt. Om deze reden is het niet verstandig om QEMU als hypervisor te kiezen in een productieomgeving. KVM daarentegen ondersteunt wel hardwarematig virtualisatie en heeft hiermee ook een performance voordeel t.o.v. QEMU op de gebruikte servers. Om deze reden is er gekozen voor KVM als hypervisor.

7.1.5 Neutron

De vierde service die is geïnstalleerd betreft Neutron, de networking service. Met Neutron wordt netwerk connectiviteit aangeboden aan de VM's. Het idee bij Neutron is dat er virtuele netwerken kunnen worden opgezet waar VM's aangehangen kunnen worden. Dit koppelen gaat via een virtuele switch (Open vSwitch).

Klanten moeten binnen het proof of concept ook de mogelijkheid hebben om IP-adressen op te vragen waarmee ze naar het i3D netwerk (internet) kunnen communiceren, dit gaat door middel van floating IP's in OpenStack. Floating IP's zijn echter zo ingesteld dat het een lokaal IP-adres betreft binnen OpenStack: Een NAT-oplossing dus. Aangezien er geen SNAT en DNAT gebruikt mogen worden binnen dit proof of concept en er voldoende vrije globale IP-adressen voor handen zijn binnen i3D wordt er on-to-one NAT gebruikt. In deze NAT oplossing wordt er één globaal IP-adres geconverteerd naar één intern IP-adres.

7.1.6 Cinder

Om de diskspace op te zetten op de block storage node voor instances moet in het geval van dit proof of concept LVM worden gebruikt. Omdat er in de Block storage nodes RAID-kaarten worden gebruikt is er een uitdaging. Deze uitdaging is het volgende:

Met een hardware RAID in RAID 10 waarbij de vier drives tot één grote drive worden gemaakt is er geen plek meer voor lege drives. De maximale drive capaciteit in een DL120 is namelijk vier. Om Cinder op te zetten is er een leeg stuk ruimte op een drive nodig. Aangezien er RAID wordt gebruikt moet de LVM partitie “naast” het OS worden geïnstalleerd.
Dit kan op een aantal manieren:

- Met een loopback device.
- Op de zelfde schijf in een andere partitie.

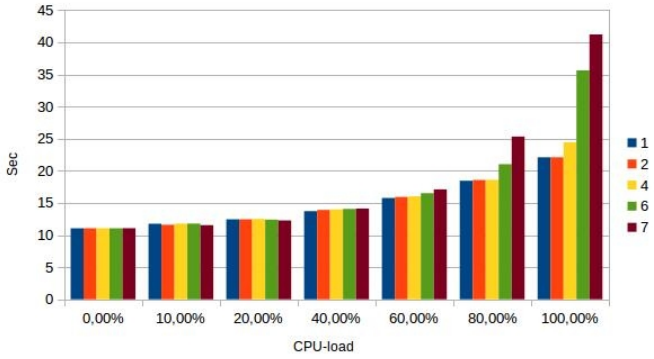
Met een loopback device wordt er binnen het OS een virtueel device gemaakt. Het besturingssysteem ziet dit virtuele device als een echte fysieke schijf. Met dit device kan Cinder worden opgezet. Echter zal de performance hier onder lijden. Dit komt door het “emuleren” van de schijf. Het pad van de data volgt dan Instance → Block storage node → OS op block storage node → Loopback device.
De betere oplossing is dus de installatie van de LVM-partitie naast het OS. Hier hoeft de data niet eerst te worden omgezet naar een virtuele schijf in het OS. Het pad van de data die hier wordt gevolgd is: instance → Block storage node → LVM partitie.

7.2 Tests

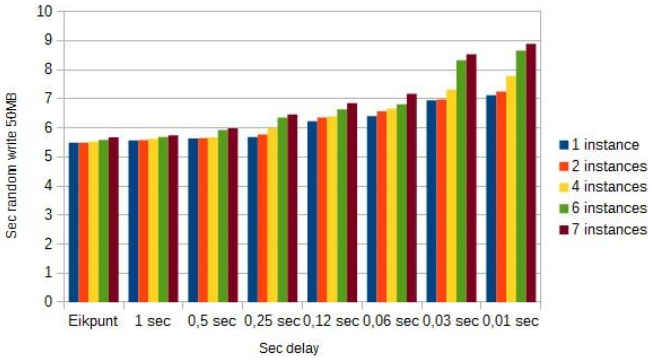
Na het opzetten van de proefopstelling kon de performance worden getest aan de hand van de performance systeemeisen. De systeemeisen staan beschreven in hoofdstuk 5. Uiteraard is er voordat het testen begon een testplan geschreven, deze is te vinden in bijlage XII.

7.2.1 Compute node tests

Allereerst is er een test opgesteld om de volgende eis te testen: “In het proof of concept mag de slowdown op de VM's niet hoger worden dan 200% bij normaal gebruik (uitgezonderd piekverkeer)”. Bij deze eis gaat het vooral om het punt dat de gebruikers geen last van elkaar mogen krijgen op hun instances. Deze eis is in het leven geroepen door de opdrachtgever omdat de hele opstelling draait op oude systemen. Op het moment dat er meerdere personen gebruik gaan maken van één systeem kunnen er performance drops te zien zijn.
Het gaat bij deze eis om de compute node op zichzelf: kan de hardware de load aan? Als deze eis negatief wordt getest dan zal dit inhouden dat de overcommit waarde op de compute node omlaag moet worden gebracht. Dit houd direct in dat er minder klanten op een compute node kunnen, oftewel: minder inkomsten per compute node. Dit is een direct link met de winstberekening.

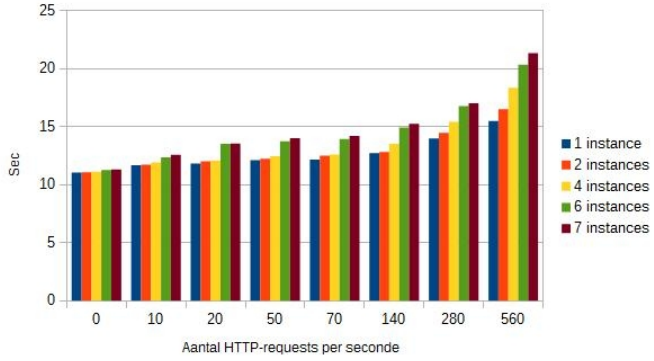


Afbeelding 13: De uitslag van de CPU-load test.



Afbeelding 14: De uitslag van de HDD-load test.

Om deze overcommit te kunnen berekenen en de eis te testen zijn er tests opgezet die CPU-noise, HDD-load en een webserver nabootsen. Voor de CPU-test betekent dit dat er op X aantal instances CPU-load wordt gezet en op een willekeurige instance (op dezelfde node) wordt gemeten wat de slowdown is. Voor de HDD-test is er getest door random data te schrijven naar een bestand op de instances. De webserver test is uitgevoerd door met “Apache AB” HTTP-requests van een instance naar een webserver op dezelfde instance te sturen. Bij de webserver test zijn er naast HTTP-requests ook database request gegenereerd. Omdat op een webserver niet elke HTTP-request meteen betekent dat direct ook de database wordt aangesproken is deze verhouding om 5:1 gehouden: 5 HTTP-requests betekent dat er 1 database request wordt verstuurd.



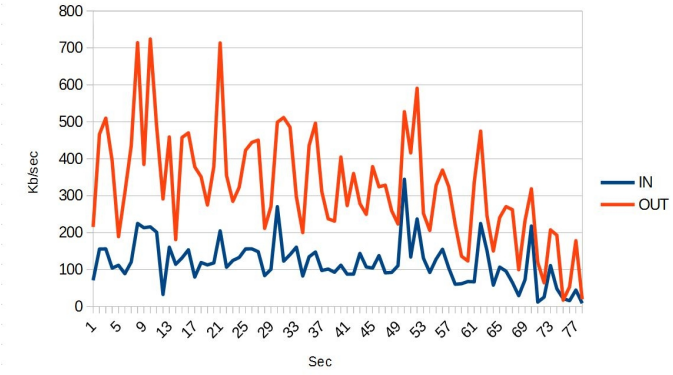
Afbeelding 15: De uitslag van de webserver test.

De uitslagen van deze tests moesten aangeven of noise op de instances aanvaardbaar bleef. De eisen zijn gecontroleerd door in alle tests een een ijkpunt (nulpunt) te nemen en vanaf hier de eis te controleren. Bij de CPU-test, HDD-test en webserver test waren de ijkpunten de eerste rij in de grafieken (afb. 23, 24 en 25).
In deze tests mochten de waarde onder “normale load” niet boven de 200% slowdown komen.
Bij de CPU-test is “normale load” maximaal 60%. Bij de HDD-load test kon niet gemakkelijk een normale load waarde worden gevonden. Bij de HDD-load test mag de gehele test niet boven 200% van het ijkpunt komen.
Bij de webserver test is een normale load voor een kleine webserver 140 HTTP-requests/sec. In afbeeldingen 13, 14 en 15 zijn de uitslagen van de compute node tests te zien.
Uit de grafieken is op te maken dat alle compute node tests positief waren, in elke test is de “normale load” waarde immers niet 200% hoger dan het nulpunt. De conclusie bij deze tests is dus dat kleine/middelgrote webservers op de instances kunnen draaien zonder andere instances te hinderen op de compute node.

7.2.2 Bandwidth usage tests

Naast de compute node tests moest ook de bandwidth usage van de lijn worden getest. Dit moest worden gedaan om de volgende performance systeemeisen positief of negatief te kunnen testen:

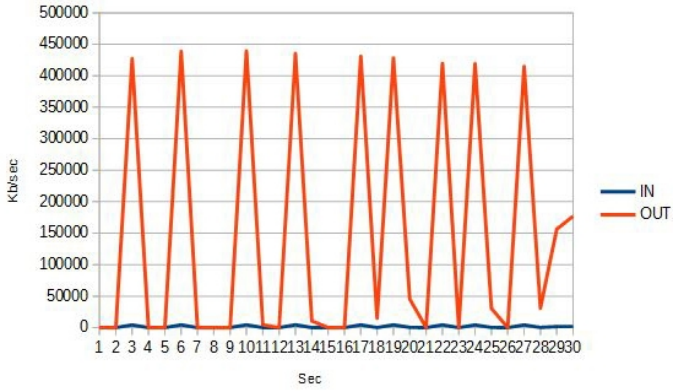
- Het netwerkverkeer over heel het netwerk mag onder normale omstandigheden (uitgezonderd piekverkeer) niet over 50% van de lijn komen.
- Het netwerkverkeer van en naar de storage mag niet zo hoog worden dat de verbinding wordt dichtgetrokken. Tijdens normaal (uitgezonderd piekverkeer) gebruik mag het niet hoger worden dan 50% van de lijn.



Afbeelding 16: De uitslag van de verbinding load test.

Bij de bovenstaande systeemeis werd OpenStack Rally ingezet om de test op te zetten. Met Rally kan er een realistische user load worden gegenereerd binnen OpenStack. Er kan bijvoorbeeld met 3 users in 2 tenants 100 keer een user list worden aangevraagd.
Door een test op te zetten waarbij alle acties die users zouden kunnen uitvoeren tegelijkertijd uit te voeren kan een realistische maximum load worden opgezet binnen OpenStack. In de test is er gemeten interface van de controller node op het management netwerk. Dit is gedaan omdat hier alle requests aankomen. Op dit punt kan dus ook worden gemeten wat het bandbreedteverbruik is.

Er is load gegenereerd door OpenStack Rally activiteiten te laten uitvoeren die users van de cloud ook zouden uitvoeren. In afbeelding 16 is te zien hoe het bandbreedteverbruik zich heeft gehouden tijden de load generatie. Het uitgangspunt van de test is dat normaal verkeer niet boven 50% van de lijn uitkomt. De bandbreedte van de lijn betreft 1Gb. Het maximale verbruik op de lijn tijdens de test was ongeveer 750Kb/s. Dit betekent dat de test positief was.
De conclusie bij deze test is dat het verkeer op het management netwerk minimaal met 50000% (1mb/s → 500Mb/s) kan groeien alvorens de helft van de lijn te bezetten.



Afbeelding 17: De uitslag van de storage load test.

Vervolgens moest ook de andere performance systeemeis worden getest. Hier moest de verbinding tussen de compute nodes en de block storage node worden getest. Bij normaal gebruik zou deze verbinding niet boven de 50% bandwidth

usage mogen komen.
De test is opgezet door Cinder block volumes aan te maken op verschillende instances. Door load te genereren op deze volumes kan een maximale load worden behaald. Aan de hand van deze resultaten kan worden gezien hoe de bandwidth usage van de connectie tussen de compute nodes en de block storage node is.
Eerst zijn de Cinder volumes gekoppeld aan de instances. Vervolgens is er door random data te schrijven naar deze volumes load gegenereerd.
De testresultaten zijn zo dat het totale verbruik onder 50% van de totale bandbreedte van de lijn blijft: De test is dus geslaagd. Op afbeelding 17 is de uitslag van deze test te zien.
De conclusie bij deze test is dat wanneer het gemiddelde bandbreedte verbruik over de gehele test wordt genomen het bandbreedteverbruik niet hoog is (rond de 100Mb/s gemiddeld in 30 seconden testen). OpenStack verstuurd zijn informatie dus met pieken naar de cache van de Cinder node, die op zijn beurt de informatie naar het RAID-array schrijft.

7.4 Aanpassing aan het project vanwege achterstand

In de architectuurfase van het project werd duidelijk dat het project iets achterliep op de planning. Hier moest iets aan worden gedaan. In samenspraak met de opdrachtgever is er besloten om het inbouwen van redundantie voor de meest kritieke bottleneck te laten vervallen. Het nadeel was echter wel dat dit effect pas duidelijk werd in de bouwfase. In de architectuurfase zijn de performance tests opgenomen om te testen wat de meest kritieke bottleneck zou vormen in de proefopstelling. Aan de hand van deze tests kon hier actie op worden ondernomen in de bouwfase. Omdat er geen actie meer ondernomen hoefde te worden op de meest kritieke bottleneck bleven de performancetests in de architectuurfase wel bestaan, er is namelijk nog wel gekeken of de systemen de OpenStack omgeving aan konden. Dit bekijken of de systemen OpenStack aankonden kon worden gecombineerd met het testen van de performance systeemeisen op de proefopstelling.

8. Infrastructuurfase

In dit hoofdstuk wordt de output van de infrastructuurfase besproken. Deze fase volgt op de architectuurfase. De infrastructuurfase is in het leven geroepen om de globale ontwerpen uit de architectuurfase om te zetten in gedetailleerde en “direct te implementeren” ontwerpen. In bijlage XI wordt het infrastructuurontwerp uit deze fase beschreven.

In de architectuurfase was er een proefopstelling opgezet om de performance systeemeisen te kunnen testen alvorens de architectuur te ontwerpen. Van deze proefopstelling was een groot deel opnieuw te gebruiken voor het proof of concept.

In het infrastructuur ontwerp is dus rekening gehouden met de proefopstelling die er al stond. Op deze manier was de moeite die in het opzetten van de proefopstelling was gestoken niet voor niets. In de proefopstelling was al rekening gehouden met de IP-nummering, portnummering, VLAN's, serverbenaming, hardware in de nodes, et cetera. Bij het ontwerpen van de infrastructuur is er informatie toegevoegd aan de nummeringen, benamingen, planningen, etc die er al lagen van de proefopstelling.

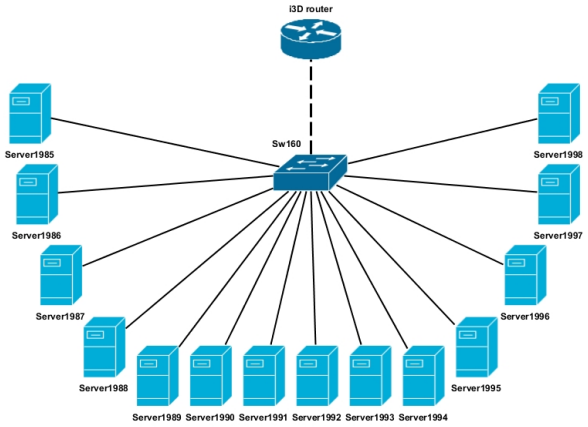
8.1 Netwerk toplogie

In dit subhoofdstuk wordt de netwerktopologie gedetailleerd weergeven. Aan de technische netwerktopologie kan niet zo veel informatie ontleent worden. Dit komt door het feit dat alle servers op één switch zitten aangesloten terwijl alle servers wel meerdere verbindingen hebben. De meeste servers zitten ook in meerdere netwerken. De technische netwerktopologie is te zien in afbeelding 18. In deze afbeelding staan alleen alle verbindingen met de switch aangegeven, een lijn van server naar switch kan echter ook meerdere connecties voorstellen. De verbinding tussen de switch en de router is een top of rack verbinding van 10Gb/s.

Met deze reden is er ook een wat meer concreet ontwerp gemaakt waar duidelijk staat aangegeven in welk netwerk elke verbinding zich bevindt, welke interface voor de betreffende verbinding wordt gebruikt en de IP-adressen per verbinding. Deze variant is te zien in afbeelding 19. Onderaan naast de network node staat ook een kader met een IP-adres range. Deze range geeft aan welke IP-adressen gebruikt kunnen worden als Floating IP binnen OpenStack. De network node deelt deze floating IP-adressen zelf uit op verzoek.

8.1.1 Systemen en hardware

Alle servers die gebruikt zijn in het ontwerp zijn allemaal van het type HP DL120 G6. In deze servers is 8GB unbuffered RAM aanwezig. Tevens hebben alle servers de beschikking over 1 processor van het type Intel Xeon X3430. Het is een quad core processor zonder hyper threading met de cores draaiend op 2,4Ghz. De servers hebben de beschikking over 2 ingebouwde NIC's.
Door het lage aantal netwerkinterfaces die standaard aanwezig zijn op de DL120 servers is er gekozen om bij de volgende nodes het aantal interfaces met twee uit te breiden met een uitbreidingskaart: Network node en de compute nodes. Deze uitbreidings kaart is van Broadcom en van het type “D43042 PCIe”



Afbeelding 18: De technische netwerktopologie.

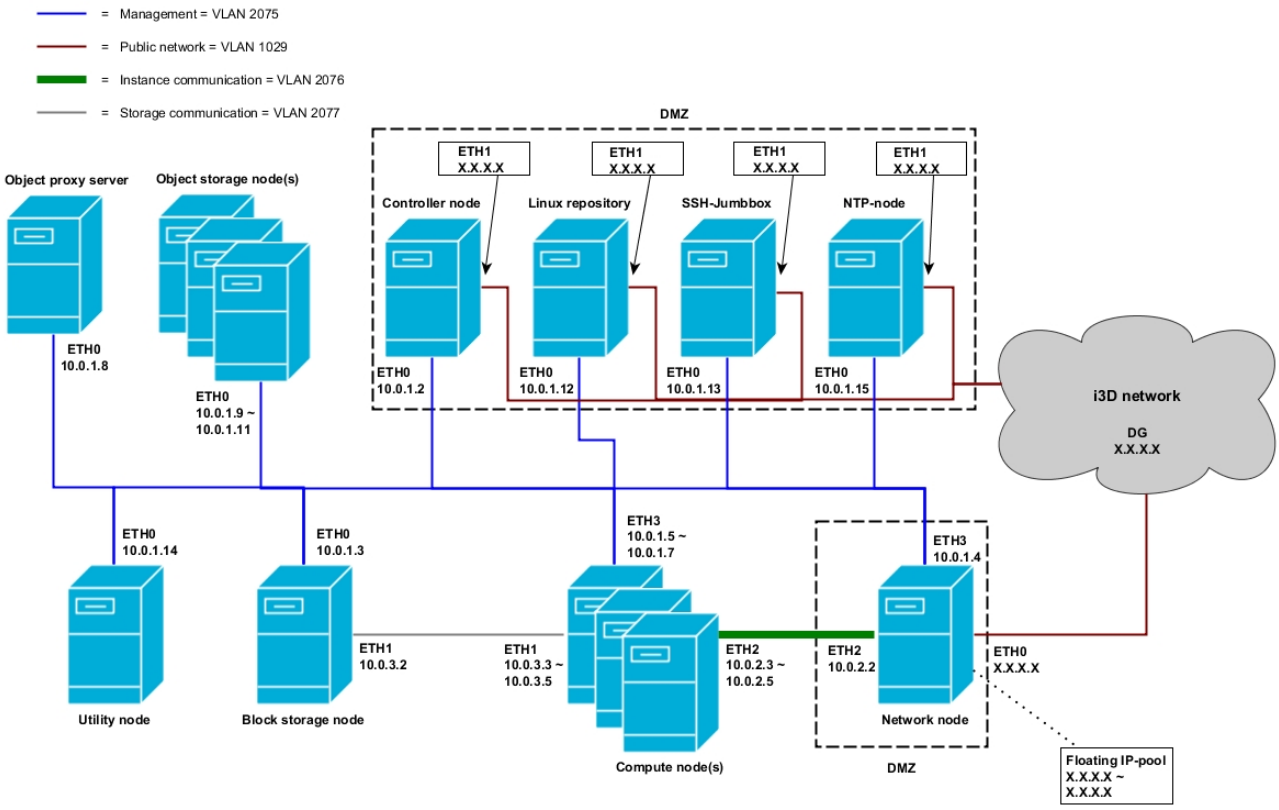
De block storage node is uitgebreid met een RAID-kaart om te voldoen aan een systeemeis. Deze RAID-kaart biedt aan vier schijven connectiviteit. De vier schijven moeten in RAID10 worden gezet. De RAID-kaart is van het type “HP Smart Array P410/256 2-ports Int PCIe x8 SAS Controller”

De switch die gebruikt wordt in het ontwerp is van het type Dell Powerconnect 5548. Het is een 48 ports switch. Elke port is

een gigabit verbinding. Tevens wordt er enkel gebruik gemaakt van CAT5e bekabeling. Deze koper bekabeling kan een snelheid van 1Gb/sec behalen.
Alle nodes die worden opgenomen in het proof of concept draaien allemaal op een fysieke servers. Er wordt geen gebruik gemaakt van virtualisatie voor de platforms van de nodes. Deze keuze is gemaakt om de performance zo hoog mogelijk te houden.

8.1.2 IP-planning en switch indeling

Op de centrale switch in dit project moeten alle porten in verschillende VLAN's worden ingedeeld. Omdat de netwerken in het proof of concept allemaal intern zijn behalve het i3D-netwerk is er gekozen voor makkelijke IP-reeksen: het management netwerk heeft de reeks 10.0.1.X/24, de instance-tunnel heeft de IP-reeks 10.0.2.X/24 en het (block) storage communication netwerk heeft de reeks 10.0.3.x/24. De VLAN-nummers 2075, 2076 en 2077 zijn gekozen omdat het de eerst voorkomende vrije VLAN's waren binnen het netwerk van i3D.
Om met het i3D-netwerk te verbinden en floating-IP's uit te kunnen delen aan instances is er een /26 pool van IPv4-adressen beschikbaar gemaakt aan het project. Het VLAN-nummer van deze pool is 1029 en is gekozen omdat dit het eerst voorkomende vrije VLAN-nummer was. Het VLAN 1029 is het OpenStack VLAN op het i3D-netwerk.



Afbeelding 19: De netwerktopologie van het proof of concept.

8.1.3 Harddisk management

De nodes/servers binnen OpenStack hebben de volgende aantallen HDD's aan boord:

Tabel 6: Het aantal HDD's in de verschillende servers.

Node	Aantal nodes	Aantal HDD's	Filesystem	Opslagruimte HDD's
Controller node	1	1	EXT2(boot partitie), EXT3	250GB
Compute node	3	1		250GB
Network node	1	1		250GB
Block storage node	1	4 (In RAID10)		250GB
Utility node	1	1		250GB
Swift proxy server	1	1		250GB
Swift storage node	3	4		250GB
SSH-jumpbox	1	1		250GB
Repository node	1	1		250GB
NTP-node	1	1		250GB

8.2 Applicaties en configuratie

De applicaties die worden gebruikt bij dit project maken voornamelijk deel uit van OpenStack. Bij het opzetten van het proof of concept gaat er echter niet van alle OpenStack-componenten gebruik worden gemaakt, dit komt omdat er veel OpenStack-componenten bestaan, en het worden er steeds meer. In tabel 7 is te lezen welke nodes welke services/componenten gebruiken in het proof of concept. Een uitgebreid overzicht van de services en de configuratie hier van is te vinden in bijlage XI (Infrastructuurontwerp).

Tabel 7: De verschillende nodes binnen het PoC met de services die zij huisvesten.

Node	Welke services
Controller node	MySQL, NoSQL, RabbitMQ, Ceilometer-central, Glance, Keystone, Apache2, Memcached, API-endpoints, Horizon
Compute node(s)	Nova, KVM, Ceilometer-meter
Network node	Neutron, OpenvSwitch
Cinder node	Cinder, iSCSI, LVM
Object storage node(s)	Swift-container, Swift-accounts, Swift-objects, XFS
Object proxy server	Swift-proxy, Memcached
SSH-jumpbox	SSH
NTP-node	NTP
Repository node	APT-mirror, FTP-server
Utility node	OpenStack Rally

Naast deze componenten zijn er ook applicaties nodig waar de OpenStack componenten van af hangen. Als eerste is er het OS die op de nodes moet komen te draaien. Het OS is voor het gehele proof of concept vastgesteld op Ubuntu 14.04 server edition. Er is gekozen voor deze Linux distibutie omdat het overgrote deel van de OpenStack installaties uitgevoerd worden op Ubuntu[46], fouten en bugs zullen dan ook veel sneller worden gevonden en er is betere ondersteuning. Naast de voorgaande onderbouwing van de keuze is er ook gekozen voor Ubuntu omdat ik, de afstudeerder, hier verreweg de meeste ervaring mee heb opgedaan binnen en buiten de opleiding.

Om data op te slaan hebben alle componenten een database nodig. De database die gebruikt wordt om componenten data in te laten opslaan is MySQL. De keuze is gevallen op MySQL omdat deze database variant een standaard is binnen i3D. De OpenStack developers raden zelf MySQL of MariaDB aan[47], beide zijn echter vrijwel identiek.

Echter wordt ook een NoSQL-database geïnstalleerd. NoSQL is nodig om OpenStack-Ceilometer functioneel te krijgen. De OpenStack developers raden NoSQL[48] aan bij het opzetten van Ceilometer omdat NoSQL door zijn opbouw een hogere performance kan behalen van MySQL.

Het onderdeel van OpenStack waar de autorisatie en authenticatie wordt verzorgt, Keystone, is afhankelijk van een Apache2 server. Om deze reden wordt er ook een Apache2 server geïnstalleerd.

Ook is het onmisbaar in een OpenStack omgeving om een functionerende messagequeue te hebben ingericht.
Als AMQP-messagequeue is er gekozen voor RabbitMQ omdat dit de standaard messagequeue binnen Ubuntu en OpenStack is: *"OpenStack Oslo RPC uses RabbitMQ by default..."*[49]. Hier wordt uitgegaan van het principe van "hoe meer personen de software gebruiken hoe beter de ondersteuning is".

8.2.1 Non-OpenStack nodes (in het DMZ)

Naast de OpenStack nodes moeten er in het proof of concept ook een aantal extra nodes worden opgenomen. Deze nodes worden in het DMZ geplaatst (zie hoofdstuk 7.4: beveiligingsmaatregelen).

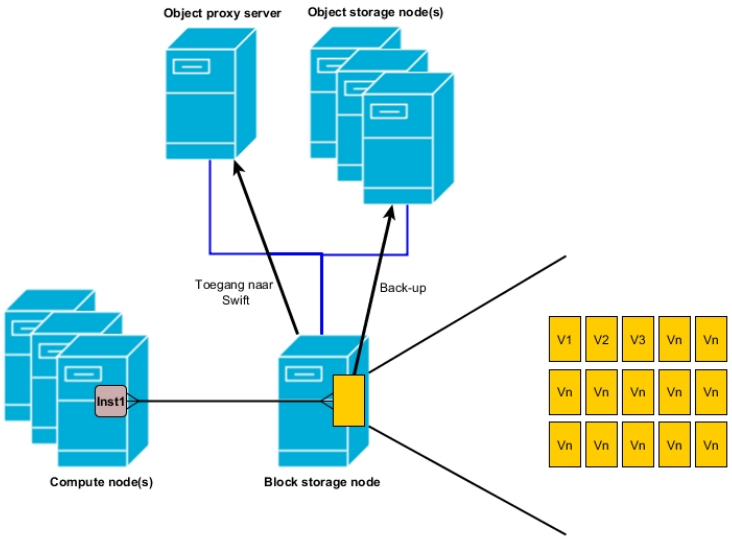
De eerste extra node is de NTP-node: deze node moet kunnen synchroniseren met de NTP-server van i3D. Deze node is in het leven geroepen zodat de controller node niet hoeft te synchroniseren met NTP, het vormt dus een soort tussenlaag.
Op deze node is de service "ntp" geïnstalleerd en ingesteld als NTP-server. De NTP-node maakt verbinding en synchroniseert met de interne NTP-server van i3D.

De tweede extra node is de repository node: deze node vormt een lokale Linux repository zodat alle nodes in het proof of concept lokaal kunnen updaten. Met het packet "apt-mirror" wordt de repository opgezet en de node gesynchroniseerd met een bestaande mirror. Via "proftp" kunnen de nodes op het management netwerk hun updates ophalen bij de repository node.
Alle systemen zijn opgezet met Ubuntu 14.04 LTS server edition 64-bit: dit betekent dat er op de repository node ook alleen de packets voor deze release en de packets voor de OpenStack Liberty release aanwezig hoeven te zijn.

Als laatste is er de SSH-jumpbox: via deze node kunnen alle nodes bereikt worden via het management netwerk. De SSH-jumpbox is in het leven geroepen om niet het gehele management netwerk open te moeten laten om het te kunnen managen via SSH.
Op de jumpbox hoeven geen packages geïnstalleerd te worden. Dit komt omdat de SSH-service standaard geïnstalleerd staat op Ubuntu 14.04.
Wel moeten er op deze node SSH-keys worden geactiveerd, dit wordt gedaan door "SSH authorized_keys". De SSH-keys moeten worden geactiveerd voor de "admin1997" user. Na het instellen van de key authenticatie moet de authenticatie via username/password in SSH worden uitgeschakeld.

8.3 Back-up plan

In dit hoofdstuk worden twee verschillende soorten back-up beschreven. Als eerste is er de back-up voor klanten. Als service in de achtergrond moet er een back-up worden gedraaid. In geval van calamiteit bij de gebruiker kan deze back-up worden aangesproken.



Afbeelding 20: Het uitvoeren van een incremental non-disruptive back-up binnen OpenStack.

Aan de andere kant is er ook de systeem back-up. Dit is de back-up van de OpenStack services zelf. Voor de systeem back-up is een vaste procedure die OpenStack voorschrijft.

8.3.1 Klant back-up

Bij gesprekken met de opdrachtgever is naar voren gekomen dat hij aangetoond wil hebben dat er wel back-up kan plaatsvinden voor de opslag van users. De back-up moet echter “achter de schermen” gebeuren. In een cloud oplossing die uit dit PoC voortkomt zijn de gebruikers zelf verantwoordelijk voor hun data. Er moet echter wel van de data van elke gebruiker met X tijd interval een back-up worden gedraaid. Op het moment dat de gebruiker zijn of haar data kwijt is door uitval en zelf geen back-up heeft kan er tegen een vergoeding aanspraak worden gemaakt op de back-up uit de cloud zelf.

Door het gebruik van “cinder backup” zijn er back-ups mogelijk van de Cinder block storage naar de Swift object storage. De Swift storage is een opslag gebaseerd op objecten. Van nature is Swift (en iedere object storage) ook high available. Dit komt omdat er standaard twee replica's worden gemaakt van alle data: de data bestaat drie keer binnen Swift.

De data van users op de blocks zal incremental non-disruptive worden geback-upt. Dit betekent dat de services door blijven draaien: een live back-up van de Cinder volumes. De back-ups kunnen worden gepland op een rustig moment, bijvoorbeeld 's nachts. Om de week of om een aantal dagen kunnen de back-ups worden gedraaid. Op afbeelding 20 is te zien hoe de incremental non-disruptive back-up wordt gecreëerd.

8.3.2 Systeem back-up

Ook is er een back-up plan voor de systeembestanden van OpenStack. Er kan een back-up worden gemaakt van alle OpenStack-componenten door de files van de betreffende services samen met dumps van de MySQL en NoSQL databases in een map te plaatsen. De bestanden van de betreffende services kunnen worden gevonden in de mappen: /etc/[servicenaam], /var/lib/[servicenaam] en /var/log/[servicenaam].
Vervolgens moet de map met alle service bestanden en database dumps worden ingepakt als TAR bestand en opgeslagen worden op een veilige plaats. In het proof of concept is de “veilige plaats” de Swift storage.

8.4 Beveiligingsmaatregelen

Om te zorgen dat de security systeemeis tot uitvoering wordt gebracht zijn er beveiligingsmaatregelen nodig. De eerste maatregelen zijn al te zien in de netwerk topologie (afbeelding 29). In dit subhoofdstuk worden alle beveiligingsmaatregelen besproken die uitgevoerd moeten worden bij het opzetten van het proof of concept.

Er moet gebruik gemaakt worden van een DMZ. Alle onderdelen binnen de infrastructuur die verbinding moeten maken met het i3D netwerk staan in de DMZ. Alle nodes in de infrastructuur, zeker de nodes in de DMZ, hebben een Iptables firewall die alle porten dichtzet behalve de porten die nodig zijn. Door het implementeren van een DMZ wordt er een extra compartiment gevormd om de interne netwerken van het proof of concept. In subhoofdstukken 7.4.6 ~ 7.4.10 worden de verschillende nodes besproken die in het DMZ staan.

8.4.1 Users en passwords

Op alle fysieke nodes zijn er twee users aanwezig, dit zijn de root user en een admin user. De admin user is de user die gebruikt wordt voor alle administrator werkzaamheden. De naam van de admin user komt overeen met de server waar hij actief op is, bijvoorbeeld: de admin op server1988 heet “admin1988”. De root user wordt niet gebruikt. De admin users kunnen door middel van SUDO hun privileges liften. De admin users moeten ook deel uitmaken van de admin group.
De wachtwoorden worden opgeslagen in een Keepass database. Tevens zijn de passwords die worden gebruikt binnen dit PoC 20 karakters lang. De passwords worden gegenereerd met:
< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-20};echo;

8.4.2 Updates

Alle fysieke nodes in het netwerk kunnen steeds worden voorzien van de laatste updates door de repository node (subhoofdstuk 7.4.3). Hierdoor blijft de omgeving up-to-date.

8.4.3 VLAN's en VLAN-tunneling

Op afbeelding 19 (de netwerk topologie) is er ook te zien dat er VLAN's worden gebruikt in de opbouw van het proof of concept. Het gebruik van deze VLAN's heeft tot gevolg dat de verschillende netwerken onderling niet kunnen communiceren (geen broadcast-domain). Communicatie uit het ene netwerk kan zo niet naar een ander netwerk binnen OpenStack, dit zorgt voor een betere performance en beveiliging. De verschillende netwerken zijn: management, public, block storage en VM-tunnel.
De virtuele netwerken binnen OpenStack worden gescheiden door GRE-tunnels. OpenStack maakt tunnels aan voor alle tenants die actief zijn binnen de cloud. De tunnels lopen naar alle compute nodes waar de betreffende tenant actief is. Het verkeer wordt verstuurd met een segmentation-ID. Tussen de tunnels is er geen uitwisseling van communicatie mogelijk. De GRE-tunnels worden door OpenStack zelf opgezet, er moet wel worden aangegeven wat de tunnel-ID-range is (1:1000).

8.4.4 HTTPS en XSS-protectie voor OpenStack Horizon

Om aanvallen op OpenStack Horizon en op de gebruikers te voorkomen wordt er gebruik gemaakt van HTTPS in plaats van HTTP voor OpenStack Horizon (dashboard). Bij het invoeren van de HTTPS verbinding wordt ook direct gebruik gemaakt van HSTS. HSTS kan worden ingevoerd met het omzetten van één parameter. Met HSTS wordt er afgedwongen dat een gebruiker HTTPS gebruikt, zo kunnen aanvallen op basis van HTTP niet of veel minder makkelijk uit worden gevoerd.
De aanvallen waar tegen wordt beveiligd met deze maatregelen zijn in ieder geval “cookie hijacking” en “downgrade attacks”. Met cookie hijacking kan de sessie van de gebruiker worden overgenomen door een aanvaller. De aanvaller kan vervolgens toegang krijgen tot het account van de gebruiker. Met downgrade attacks kan een oudere versie van een protocol worden gebruikt terwijl het nieuwere protocol actief is (HTTP I.P.V. HTTPS). Deze downgrade attack wordt voorkomen door HSTS.

8.4.5 Firewalls en Security groups

Er zijn binnen het proof of concept een aantal manieren waarmee de toegang van het netwerkverkeer geregeld kan worden. Voor de fysieke machines zijn er de Iptables firewalls en voor de OpenStack instances zijn er de “security groups”.
De firewalls moeten alle porten afsluiten behalve de porten die worden gebruikt. De security groups werken op dezelfde manier als een IP-tables firewall, alleen nu virtueel en op basis van apart in te stellen OpenStack “security groups”. Standaard staan alle ports in de security groups dicht.

Op het i3D netwerk van het proof of concept zijn de controller node, repository node, NTP-node en de SSH-jump node aangesloten. Voor de porten die open staan op het i3D-netwerk moeten extra beveiligingsmaatregelen worden ingevoerd met Iptables. Met “extra maatregelen” wordt bedoeld dat de rules zo specifiek mogelijk worden gemaakt doormiddel van source- en destination IP(-range), source- en destinations ports, gebruikte interface en states bevatten. De default policy in de Iptables-rules is DROP (niet doorlaten).

8.4.6 Controller node

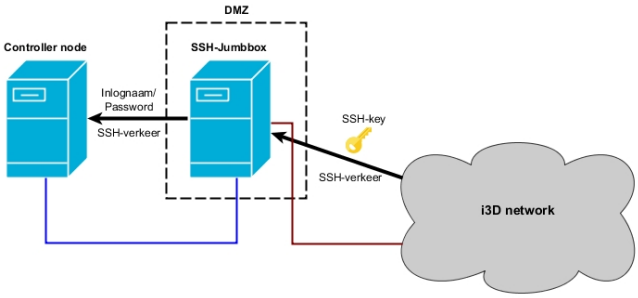
De controller node is het onderdeel wat de (meeste) communicatie verzorgt in OpenStack. De enige communicatie die van het i3D netwerk komt gaat via port 443, de HTTPS port. De beveiliging van OpenStack Horizon wordt besproken in subhoofdstuk 7.4.4.

8.4.7 Repository node

De repository server kan worden gezien als een soort “proxy server” voor Linux packages. In plaats van dat de nodes op de interne netwerken een connectie naar het internet nodig hebben om updates en packets binnen te halen wordt er intern een repository opgezet. Deze repository node zorgt voor een tussenlaag omdat hij is opgenomen in het DMZ.
De repository node update zijn packets uit de i3D-repository.

8.4.8 SSH-jump node

De SSH-jump node is nodig om SSH-verkeer naar het interne netwerk mogelijk te maken voor beheerders van de cloud. De reden dat er een extra node wordt gebruikt voor de connectie in plaats van directe verbinding met het interne netwerk is veiligheid.
Zoals in de configuratie van de SSH-jump node is vermeld wordt de SSH-verbinding naar deze node opgezet door een SSH-key. Een SSH-key is bijna onkraakbaar en zorgt voor een veel betere beveiliging dan inlognaam/password. Vanaf de jump node kan er naar interne nodes wel met inlognaam/passwords worden ingelogd. Op afbeelding 21 is te zien hoe er ingelogd kan worden met SSH binnen het PoC.



Afbeelding 21: De SSH-jump node functioneert als "tussenlaag" in de SSH-communicatie met het interne netwerk.

8.4.9 NTP-node

De NTP-node is in de DMZ opgenomen omdat deze service ook actief moet zijn binnen het netwerk. De systemen hebben allemaal de juiste tijd nodig om bijvoorbeeld in de logs de goede tijd aan te geven.
Vaak synchroniseert NTP via een internet verbinding. Vrijwel alle nodes hebben geen verbinding met het internet, vandaar dat er intern een NTP-node wordt opgenomen. Deze server wordt dedicated NTP-server gemaakt aangezien er NTP-aanvallen zouden kunnen voorkomen. Deze aanvallen op NTP hebben op deze manier minder uitwerking op de werking van de cloud dan wanneer de NTP-service op de controller node zou draaien.
De NTP-server synchroniseert alleen maar naar de interne NTP-server van i3D. Het netwerk van i3D wordt vanuit het PoC-netwerk gezien als extern. Door het feit dat de OpenStack NTP-server alleen synchroniseert met de NTP-server van i3D wordt dit opgenomen in de configuratie van de NTP-service. In het configuratiebestand van de NTP-service wordt aangegeven dat er enkel moet worden gesynchroniseerd met de NTP-server van i3D.

Om NTP-reflexion en aplification attacks te voorkomen moet de service up-to-date zijn. Dit betekent dat het versienummer hoger moet zijn dan v4.2.7p26. Om er zeker van te zijn dat deze zwakheid in NTP te misbruiken is zal de monitor in NTP uitgezet moeten worden. Hiermee is geen monlist meer op te vragen op de NTP-server. Een amplification attack in NTP richt zich namelijk op de monlist.

8.4.10 Network node

De network node bevind zich ook in het DMZ. Bij deze node liggen de zaken echter net iets anders. Op de externe interface van de network node is een virtuele bridge aangemaakt. Via deze bridge kan verkeer van het internet bij de instances komen. Een voorbeeld zou kunnen zijn dat een bezoeker van een website via de network node naar de desbetreffende webserver gerouteerd wordt. Deze webserver wordt gehost op een instance in de cloud.
Verkeer van en naar de fysieke interface van de network node is niet meer mogelijk.

8.5. QoS

Binnen het proof of concept worden een aantal QoS-limieten ingesteld om zo nog beter te voldoen aan bepaalde systeemeisen. Tevens kan met de juiste QoS-instellingen de performance een betere balans krijgen. Klanten krijgen hierbij minder kans om de performance scheef te trekken en voor het noisy neighbour effect te zorgen.

8.5.1 Limits

In OpenStack kunnen extra flavors worden aangemaakt voor de klanten. De klanten kunnen aan de hand van deze flavors nieuwe instances opzetten. Standaard krijgt een flavor aantal vcores, aantal GB geheugen en aantal GB storage mee. Met de flavors kunnen ook extra specs worden meegeven, de zogenaamde “limits”.

In het proof of concept wordt alleen gewerkt met de flavor die ook gebruikt wordt in de winstberekening. Dit is de reden waarom er maar één flavor wordt besproken in dit hoofdstuk.

CPU-weighting

Met de Nova limits kan een vorm van CPU-weighting worden ingesteld. Met deze weights kan er voor worden gezorgd dat iedereen op de node zijn of haar eigen deel krijgt van de CPU-kracht. De CPU-weight kan worden meegegeven met de flavors. Aan de hand van het CPU-share nummer wat elke instance in een domain heeft kan CPU-time worden berekend. De CPU-weight wordt ingesteld op 10.

8.5.2 Cinder QoS

Met de Cinder QoS instellingen kan de load op de Cinder node worden gebalanceerd voor meerdere instances. Er moet vanuit worden gegaan dat de instances niet constant load genereren op de Cinder node. Echter voor de hoge pieken in het gebruik waardoor andere users eerder performance slowdowns krijgen moet er een peak/cap worden ingesteld.

Er wordt vanuit gegaan dat de HDD's in de RAID-opstelling 75 MB/s ~ 100 MB/s doen. Dit is het gemiddelde voor standaard 7200 RPM HDD's. Nu staan de disks in een RAID 10 opstelling. Dit betekent dus dat er performance slowdown optreed omdat de RAID-controller een mirror moet bijhouden. Echter betekent die ook dat de snelheid/performance met ongeveer 30% ~ 50% toeneemt in stipe (RAID 0). De snelheid van de schijven komt nu neer op ongeveer 130MB/s ~ 150 MB/s.
Er wordt een cap gezet op 25% van de totale performance van de block storage node HDD's. Deze waarde is gekozen omdat de instances nog genoeg performance naar de HDD's moeten houden maar aan de andere kant moet de cap op de lees- en schrijfsnelheid wel zo laag zijn dat instances andere instances zo min mogelijk performance slowdowns bezorgen. Bij een cap op 50% van de performance zou er maar één instance “zijn deel” (50%) vol kunnen trekken en blijft er voor de andere instances niks over. Als er dan nog een instance komt die ook hoge load genereert is er alsnog sprake van slowdowns voor de andere gebruikers. Bij 33% kunnen 3 personen hoge load genereren, dit is alsnog vrij weinig. Daarom is er gekozen voor een cap op 25%. Een cap op 25% op het totaal van 130 MB/s betekent alsnog een peak snelheid van ongeveer 33 MB/s.

Met een snel testje is gekeken wat de snelheid van de RAID-opstelling is. Dit testje is alleen bedoeld om te zien of mijn berekening ongeveer klopte. Hieronder staan de uitkomsten van het testje:

```
#sudo hdparm -Tt /dev/sda

Timing cached reads:   17912 MB in  2.00 seconds = 8964.72 MB/sec
Timing buffered disk reads: 216 MB in  2.1241 seconds =  101.69 MB/sec

#dd if=/dev/zero of=/tmp/output bs=4k count=100000; rm -f /tmp/output

4096000000 bytes (4.0 GB) copied, 25.0695 s, 159 MB/s
dd if=/dev/zero of=/tmp/output bs=4k count=100000
```

De read_bytes_sec parameter krijgt een waarde van 34603008 B/s.

De write_bytes_sec parameter krijgt een waarde van 34603008 B/s.

8.5.3 Neutron QoS

Binnen het Neutron component kan de QoS voor de instances worden ingesteld op de virtuele porten. Dit betekent dat de aansluiting van een instance op een virtueel netwerk afgeknepen kan worden met deze rules.
Er zal eerst een policy aan moeten worden gemaakt. Deze policy krijgt dan de naam “QoS-Neutron”. Vervolgens krijgt de policy de volgende rule:

```
QoS-Neutron
--max-kbps 40000
--max-burst-kbps 4000
```

De rule heeft dezelfde naam als de policy. De bandwidth die met deze rule wordt gegeven is 40Mb/s. Vervolgens moet deze policy worden toegepast op elke virtuele port. Deze waarde is verkregen door te overleggen met een collega binnen i3D over wat een webserver ongeveer maximaal nodig heeft aan bandbreedte. Deze waarde kan makkelijker aan worden gepast voor andere flavors.

8.5.4 Oversell

Door het toepassen van oversell of overcommit kunnen fysieke resources meerdere keren worden verkocht als virtuele resources. De overcommit moet in het proof of concept komen te liggen op: 2:1 voor vcores en 1:1 voor RAM.
De HP DL120 G6 servers hebben fysiek een CPU met 4 cores en 8GB RAM. Met de overcommit betekent dit dat er 8 vcores en 8GB RAM verdeeld kan worden over verschillende instances.

9. Ontwikkeling

In de ontwikkelfase wordt het gedetailleerde ontwerp uit de infrastructuurfase opgebouwd tot een proof of concept. In bijlage VI staat de handleiding. Deze handleiding beschrijft alle acties die uitgevoerd moeten worden om weer te komen tot hetzelfde proof of concept. Met deze handleiding wordt het proof of concept reproduceerbaar.
Uiteraard zijn in deze fase fouten opgekomen bij de opbouw. Deze fouten en de bijbehorende oplossingen zijn te lezen in het evaluatie hoofdstuk (subhoofdstuk 11.4).

9.1 Testplan

In de ontwikkelingsfase van het project is er ook een testplan gecreëerd. Aan de hand van dit testplan zijn de systeemeisen en de “groei variabele” voor de winstberekening in de testfase getest. Het testplan voor de testfase is te vinden in bijlage XII.

9.1.1 Systeemeisen testen

Het testplan beschrijft alle tests die zijn uitgevoerd om aan te tonen dat de systeemeisen correct zijn geïmplementeerd. In tabel 8 is een overzicht te zien van de geteste systeemeisen en de wijze waarop ze zijn getest.

Tabel 8: De geteste systeemeisen en de wijze waarop ze getest zijn.

Naam test	Systeemeis	Wijze van testen
Redundante storage	In het uiteindelijke proof of concept mag de storage bij schijfuitval (van 1 schijf in het proof of concept) geen dataverlies lijden.	Een HDD verwijderen uit het RAID10-array van de block storage node. Naderhand een lege schijf terugplaatsen en laten rebuilden.
Schaalbare resources	De "klanten" moeten in het proof of concept de mogelijkheid hebben om hun resources te kunnen schalen.	Snapshot maken van de draaiende instance. Verwijderen van de instance. Opzetten van de snapshot op basis van een nieuwe flavor.
Storage bestellen	De "klanten" moeten in het proof of concept de mogelijkheid hebben om extra storage te kunnen "bestellen" (schalen/resource provisioning).	Block volume aanmaken in de Cinder service. Block volume koppelen aan de draaiende instance. Gekoppelde block volume partitioneren en vervolgens belasten met het schrijven van random data.
Hosten van een virtueel OS	Er moet op het proof of concept één soort operating system (één distributie) kunnen worden gehost.	Booten van een instance op basis van Ubuntu 14.04 cloud. Koppelen van een floating IP. Inloggen op de instance via SSH.
Floating IP's	– Het proof of concept moet een alternatief voor SNAT en DNAT hebben om "klanten" publieke IP-adressen op te laten vragen. – Mensen moeten externe IP-adressen kunnen bestellen om hun instances te kunnen verbinden met internet en i3D netwerken.	Floating IP koppelen aan een draaiende instance. Via SSH inloggen op de instance. Connectiviteit van de instance met het internet testen D.M.V. pings en downloaden van een test bestand.
Klantdata back-up	– Er moet een back-up van de data van de gebruikers gemaakt kunnen worden. – Binnen het proof of concept moeten snapshots van de VM's gemaakt kunnen worden.	Een non-disruptive back-up en incremental-non-disruptive back-up maken van een gekoppeld block volume met de Cinder back-up driver. Vervolgens moeten de back-ups worden gerecovered. Swift wordt gebruikt als backend. Ook een snapshot maken van een draaiende instance en daarna recoveren.
OpenStack systeemback-up	Er moet een back-up gemaakt kunnen worden van de services binnen OpenStack.	Back-up maken van de OpenStack service bestanden en databases. Swift gebruiken als backend voor het back-up bestand. Vervolgens de database back-ups recoveren.
Control panel	Klanten moeten kunnen inloggen op het	Inloggen in het grafische control panel van OpenStack

	OpenStack control panel.	“Horizon”. Alle tabs in het grafische overzicht bekijken.
Update	De Linux installaties en de OpenStack componenten op de fysieke nodes moeten geüpdatet kunnen worden.	De mirror zelf laten updaten. Elke node laten updaten van de lokale mirror. Controleren of de updates daadwerkelijk doorgevoerd zijn op de OpenStack nodes.
Verbruik meten	Het verbruik van de klanten in OpenStack moet gemeten kunnen worden.	Een OpenStack user een actie laten uitvoeren (downloaden image) en controleren of Ceilometer dit heeft gezien en opgeslagen.
Management network security	In het proof of concept mogen onbevoegde verbindingen vanaf het i3D netwerk niet in een van de interne netwerken komen.	Portscans uitvoeren op de DMZ-nodes en controleren op open poorten. Proberen in te loggen op de SSH-jump node zonder key (alleen inlognaam/password). Via HTTP proberen in te loggen op de controller node, dit mag niet, alleen HTTPS is toegestaan.

9.1.1 Eisen die niet kunnen worden getest

In het testplan zijn een aantal systeemeisen niet opgenomen. De volgende eisen kunnen niet getest worden:

- Na het project moet de opstelling/het proof of concept opnieuw opgezet kunnen worden.
- De uiteindelijke winstberekening, in de testfase, moet positief zijn (er moet winst gemaakt kunnen worden).
- De uiteindelijke abonnementskosten voor klanten moeten lager zijn dan de primaire cloud oplossing.
- Binnen het project moet er gebruik zijn gemaakt van OpenStack.

De redenen waarom deze bovenstaande eisen niet zijn getest zijn divers. De twee eisen over de winstberekening konden pas worden uitgewerkt na het uitbrengen van het testrapport: er zijn namelijk variabelen in de winstberekening afhankelijk van de tests. Voor de andere eisen geldt dat ze wel verlangt werden in het proof of concept maar niet als zodanig te testen waren.
De documentatie eis was afhankelijk van het schrijven van de handleiding. Een handleiding is niet te testen. De eis dat er OpenStack gebruikt moet worden in het proof of concept is onnodig om te testen.

9.1.2 Tests voor de winstberekening

Naast het testen van de systeemeisen zijn er ook een aantal testen opgenomen in het testplan waarmee de limiet van verschillende nodes werd getest. Eigenlijk was het de bedoeling om deze tests al mee te nemen in de architectuurfase tests. Omdat er geen redundantie meer ingebouwd hoefde te worden op de plek van de meest-kritieke bottleneck was de reden om deze tests in de architectuurfase uit te voeren weg. Het nadeel was wel dat er geen groei waar te nemen was op de meest kritieke bottleneck binnen OpenStack en daardoor de winstberekening minder nauwkeurig zou kunnen berekenen.
In de winstberekening is het van belang dat de groei bekend is. De “groei” betekent “om de hoeveel klanten moet er een nieuwe X node ingezet worden om de performance slowdowns niet hoger te laten oplopen dan 200%”. Aan de hand van dit gegeven kan berekend worden hoeveel nodes er nodig zouden zijn in een grotere OpenStack omgeving dan het proof of concept.
Om dit te berekenen moesten de block storage node, network node en de controller node worden getest. De test staat ook beschreven in het testplan (bijlage XII).
De tests bestaan uit het genereren van “realistische” load op de betreffende node en controleren met OpenStack Rally wat de performance slowdown zou zijn. Tevens moet de betreffende fysieke node gemonitord worden op CPU-gebruik (processen), geheugengebruik en bij de Cinder node ook het schijfgebruik.

10. Testfase

De testfase is in het leven geroepen om alle tests in het testplan ook daadwerkelijk uit te voeren. In dit hoofdstuk worden de testresultaten weergeven. In tabel 8 van het vorige hoofdstuk staan de testplannen beschreven. Het testrapport is te vinden in bijlage XII. Naast het testen van de systeemeisen wordt het stroomverbruik van de verschillende stukken hardware gemeten, deze resultaten moeten ook worden meegenomen in de winstberekening.

10.1 Testresultaten

In de subhoofdstukken hier onder worden de uitgevoerde tests en de testresultaten besproken. De meeste tests zijn functietests. In deze functietests worden de acties nagebootst hoe ze normaal in een OpenStack omgeving uitgevoerd zouden worden. Als er bij het uitvoeren van deze functietests geen fouten op komen dan is de test geslaagd.

10.1.1 Redundante storage

Deze test stond in het teken van de redundante block storage. Hier moest worden aangetoond dat de Cinder block storage nog steeds door zou draaien bij het uitvallen van één HDD. De test is uitgevoerd door een HDD te verwijderen terwijl het systeem draaide. Na het verwijderen is het array gemonitord met het programma “*hpssacli*”, dit programma gaf een missende/falende schijf aan. Vervolgens is er een lege schijf in het lege slot terug gestopt. Het *hpssacli* programma gaf aan dat de recovery van de schijf was gestart.
Na een tijdje is er nog eens met het HP-programma gecontroleerd, nu gaf het aan dat het array weer in orde was. Dit betekende dat bij het falen van één schijf de node gewoon door kon draaien.
De test was geslaagd aangezien er geen fouten op zijn gekomen tijdens de test en het array aan het eind van de test weer functioneel was. De conclusie bij deze test is dat er met een DL120 G6 server met HP p410 RAID-kaart een RAID-array is op te zetten waar één schijf kan falen en dit bij tijdig ingrijpen niet zorgt voor dataverlies.

10.1.2 Schaalbare resources

In deze test moest er worden aangetoond dat het mogelijk is om een instance te schalen binnen het proof of concept. Binnen OpenStack kan er worden geschaald door een snapshot te maken van een instance en vervolgens een nieuwe instance te maken op basis van de snapshot en een nieuwe flavor.
Na het schalen van de instance werd er met de commando's *#top* en *#lsblk* nog gecontroleerd of de resources van de nieuwe flavor ook daadwerkelijk toegewezen waren aan de instance; dit was het geval.
Bij de test hebben zich verder geen fouten voorgedaan. Tevens is de instance goed opgeschaald. Dit betekend dat de test positief is. De conclusie bij deze test is dat het in het proof of concept op basis van OpenStack Liberty mogelijk is om instances te kunnen schalen en voorzien van meer/minder resources.

10.1.3 Storage bestellen

In deze tests was het de bedoeling om aan te tonen dat gebruikers storage zouden kunnen bestellen in het proof of concept. Met Cinder werd een block volume aangemaakt en deze vervolgens is vervolgens gekoppeld met een draaiende instance. Het gekoppelde volume werd gepartitioneerd en voorzien van een ext3 bestandssysteem. Het volume werd uiteindelijk beschreven met een bestand waar random data in gezet werd. De gehele test kwamen er geen fouten op en werd de eis aangetoond.
Bij het schrijven van de random data zijn er geen fouten omhoog gekomen. Dit betekend dat de test positief is. De conclusie van deze test is dat het in het proof of concept op basis van OpenStack Liberty mogelijk is om instances te kunnen voorzien van block volumes en hier data op te schrijven.

10.1.4 Hosten van een virtueel OS

In deze test was het van belang dat er werd aangetoond dat het proof of concept in staat was een virtueel operating system te hosten in de vorm van een instance. Om deze eis aan te tonen is er een Ubuntu 14.04 image in de Glance-store gezet. Op basis van de Ubuntu 14.04 image is er een instance gemaakt.
Ook was het van belang dat er toegang tot de instance kon worden verkregen, dus er is een floating-IP aangemaakt en aan de instance toegewezen. Met SSH kon er toegang worden gekregen tot de nieuw aangemaakte instance.
Heel de test zijn er geen fouten opgekomen.
Bij deze test zijn geen fouten opgetreden daarom is deze test positief. De conclusie van deze test is dat het in het proof of concept op basis van OpenStack Liberty mogelijk is om instances te kunnen hosten op basis van een Linux gebaseerd OS.

10.1.5 Floating-IP

In deze test moest worden aangetoond dat gebruikers van de omgeving IP-adressen konden bestellen voor hun instances. Aan het begin van de test stond er al een instance op basis van Ubuntu 14.04 cloud klaar. Eerst is er een floating-IP aangevraagd bij de Neutron service. Vervolgens is het floating-IP gekoppeld met de instance.
Door via SSH in te loggen op de instance kon het floating-IP nog iets verder worden getest. Het feit dat via SSH in te loggen was gaf al aan dat het floating-IP functioneerde. Vervolgens is de instance ook nog geüpdatet, er is gepingt naar een aantal adressen en er is een testbestand gedownload naar de instance.
De gehele test is er geen foutmelding opgekomen.
Alle testonderdelen zijn zonder fout uitgevoerd. Dit betekend dat de test positief is. De conclusie bij deze test is dat het in het proof of concept op basis van OpenStack Liberty mogelijk is om instances te kunnen voorzien van een IP-adres waarmee ze met een extern netwerk kunnen verbinden zonder gebruik te maken van SNAT en DNAT.

10.1.6 Klantdata back-up

In deze test moest er worden aangetoond dat er back-ups gemaakt konden worden van gebruikersdata. De data van gebruikers staat in het proof of concept in Cinder block volumes.
Er zijn back-ups te maken van Non-bootable volumes door de Cinder-backup driver: deze maakt een back-up van een volume en plaatst deze in het aangewezen backend, in dit geval het Swift backend. Er is een non-disruptive (terwijl de instance en het volume live staan) back-up gemaakt van het gekoppelde volume. Hierna is gecontroleerd of hij daadwerkelijk in de Swift-store stond en verwijderd uit de Cinder-store. Vervolgens is deze back-up weer gerecovered naar Cinder en gecontroleerd met het commando *#MD5sum* of het bestand wat aanwezig was op het volume niet veranderd was. Het bestand was niet veranderd
Hierna is er weer een back-up gemaakt net als hierboven beschreven, nu werd er echter een incremental-non-disruptive back-up gemaakt. Het is namelijk ook belangrijk dat er incremental back-ups gemaakt kunnen worden, zo is er schijfruimte te besparen en wordt er minder bandbreedte gebruikt bij het maken van de back-ups. Ook werd na het recoveren van deze back-up met *#MD5sum* gecontroleerd of het bestand dat aanwezig was op het volume niet was veranderd. Het bestand was niet veranderd.
Als laatste werd ook getest of het maken van een non-disruptive back-up van een bootable-volume met draaiende instance goed werkte. Om dit te testen werd er een Nova-snapshot gemaakt van de instance. Om te controleren of het restoren van deze snapshot functioneel was werd eerst het oude Cinder volume verwijderd en een nieuwe aangemaakt. Hierna is er een nieuwe instance opgezet op het Cinder volume op basis van de snapshot. Door een floating-IP te koppelen met de instance en vervolgens via SSH in te loggen is gecontroleerd of de instance ook echt actief was en functioneerde.
Alle commando's en acties in deze test zijn uitgevoerd door OpenStack zonder een foutmelding. Tevens komen de checksums overeen en functioneerde de instance na de recovery naar behoren.
De klant back-uptest is positief. De conclusie bij deze test is dat het mogelijk is om een back-up te maken van live instances en hun gekoppelde (live) block volumes binnen het proof of concept op basis van OpenStack Liberty.

10.1.7 OpenStack systeemback-up

In deze test moest er worden aangetoond dat er ook een back-up te maken was van de OpenStack-services. Om te

voorkomen dat de volledige cloud omgeving opnieuw opgezet moet worden bij uitval van componenten zijn deze back-ups belangrijk. De ontwikkelaars van OpenStack raden aan om een handmatige (of via een script) back-up te maken van de bestanden van alle OpenStack services.
Aan het begin van de test is een directory aangemaakt waar de bestanden van alle verschillende OpenStack-services in geplaatst moesten worden. Alle databases zijn gedumpt naar verschillende bestanden en in de back-up directory verplaatst. Ook de bestanden (configuratie, logs, etc) van de verschillende services zijn gekopieerd naar de back-up directory. Vervolgens is de back-up directory ingepakt als TAR bestand en geüpload naar de Swift-store.
Hierna is de mogelijkheid om de databases te recoveren getest door het TAR bestand weer uit de Swift store terug te halen en de database dumps te recoveren.
Gedurende de hele test heeft OpenStack geen fout gegeven en de volledige back-up procedure verliep goed, de test is dus positief. De conclusie bij deze test is dat het OpenStack systeem naar een back-up geschreven kan worden en de databases kunnen zonder problemen weer gerecovered worden.

10.1.8 Control panel

In deze test moest de functionaliteit van het OpenStack control panel worden aangetoond. Door in te loggen op het proof of concept via een browser kon het dashboard worden opgevraagd. Vervolgens is de functionaliteit van het dashboard vastgesteld door elke tab in het overzicht aan te klikken.
Hier kwamen geen fouten op; de test was dus geslaagd. De conclusie bij deze test is dat de opstelling op basis van OpenStack Liberty ook met een grafische omgeving aangestuurd kan worden en dit functioneert in het proof of concept.

10.1.9 Update

In deze test moet de functionaliteit van de lokale repository worden aangetoond. Eerst werd de lokale mirror zelf geüpdatet aan de hand van de i3D mirror. Vervolgens zijn alle nodes in het proof of concept geüpdatet aan de hand van de lokale mirror. Na het updaten van de nodes zijn er steekproefsgewijs een aantal packages gecontroleerd: de versienummers van deze packages op de nodes moesten hetzelfde zijn als de versienummers van de packages op de i3D-mirror. Alle versienummers van de packages op de nodes kwamen overeen met die op de i3D-mirror.
Omdat er in de test geen fouten zijn opgekomen bij het uitvoeren van de update commando's en alle versienummers klopte is de test geslaagd. De conclusie bij deze test is dat de nodes in het proof of concept succesvol kunnen updaten vanaf een lokale repository die opgenomen in in de OpenStack omgeving.

10.1.10 Verbruik meten

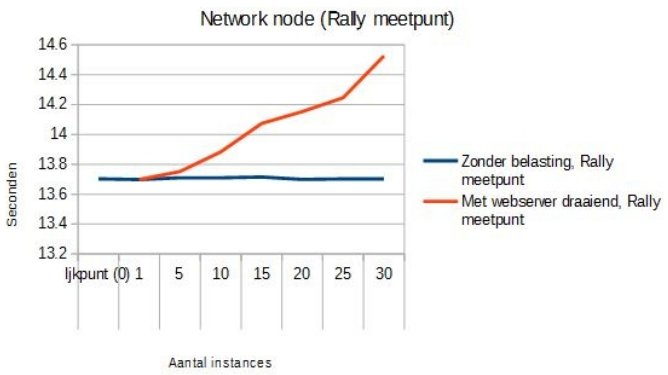
Deze test was gewijd aan het meten van verbruik die gebruikers veroorzaken. Het verbruik kan in OpenStack worden gemeten met het component “Ceilometer”. Er zijn twee soorten verbruik gemeten: image download en netwerk verbruik.
Het eerste soort verbruik is getest door een image uit de Glance-store te downloaden naar */dev/null*. Na het downloaden is er met Ceilometer gekeken of het gelogd was: *#ceilometer statistics -m image.download -p 60*. De output van het commando was het samengestelde verbruik van de “image.download meter”. In deze output stond de datum van uitvoeren, deze actie was netjes gelogd.
Voor het tweede soort verbruik was er een instance opgezet met een floating-IP gekoppeld. Door deze instance het commando's *#apt-get update* en *#apt-get dist-upgrade* te laten uitvoeren zou er netwerkverbuik worden gegenereerd. Met Ceilometer is er gekeken of dit verbruik was gelogd met *#ceilomter sample-list -m network.incoming.bytes*. De output van het commando was een lijst met twee entries: het verkeer gegenereerd door *#apt-get update* en het verkeer gegenereerd door het commando *#apt-get dist-upgrade*. Deze actie was ook netjes gelogd.
Dit betekend dat de test posities is verlopen. De conclusie bij deze test is dat het verbruik die OpenStack gebruikers genereren gemeten en gelogd kan worden met Ceilometer op een opstelling op basis van OpenStack Liberty.

10.1.11 Performance groei

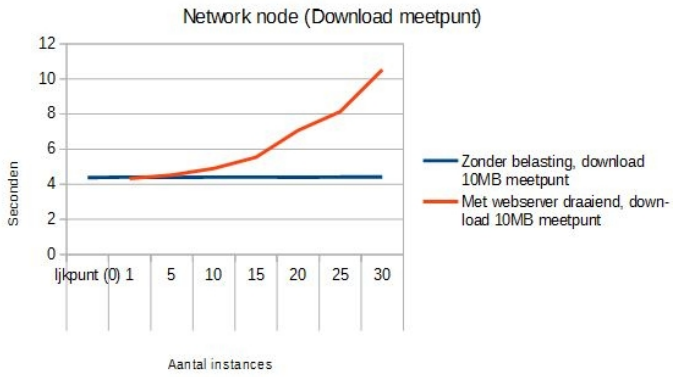
Deze test is anders dan andere tests; hier moet worden gemeten bij hoeveel instances een OpenStack node dubbel moet

worden uitgevoerd. Bij deze tests werd het omslagpunt van 200% performance slowdown aangehouden. De uitkomsten van deze tests moest worden verwerkt in de winstberekening.
Als eerste werd de controller node getest. Op deze node is load gegenereerd door met Apache AB HTTP-requests te versturen van een naar de instances en met MySQL database-writes uit te voeren op de instances.
Door met OpenStack Rally de “create-and-delete-user.json” benchmark uit te voeren kon de performance van de controller node gekwantificeerd worden. Het nulpunt (Rally laten draaien zonder instances) van de test viel op 19,6728 seconde. Een slowdown van 200% zou inhouden dat het aantal instances dat zorgt voor een slowdown van het Rally meetpunt van 39,3456 seconde het aantal instances is wat de controller node aankan. Het omslagpunt van de controller node kwam bij 16 instances. De uitslag van deze test is te zien op afbeelding 22. De conclusie bij deze test is dat het maximaal aantal instances die de controller node aankan voordat hij 200% vertraagd 16 instances bedraagt.

Bij de test van de block storage node is er ook realistische load gegenereerd op de instances door AB en MySQL. Het meetpunt werd gecreëerd door een instance 1000MB aan random data naar een Cinder volume te laten wegschrijven. Het nulpunt van deze test viel op 9,859 seconde. Een slowdown van 200% zou inhouden dat het omslagpunt lag op 19,718 seconde. Wel gold in deze test een limiet van 20 instances omdat er per Cinder node niet meer storage aangeboden kon worden (bij 20GB ruimte per instance).
De realistische load op de 20 instances kreeg de slowdown tot 10.751 seconden. Deze node moet dus dubbel worden uitgevoerd bij maximaal 20 instances vanwege de schijfruimte. De uitslag van deze test is te zien op afbeelding 33.
In deze test is wel een vergelijking opgezet (die niet meetelt in de test): er is ook volledige belasting uitgevoerd door de instances op de Cinder node (de gele lijn op afbeelding 23). Deze belasting is uitgevoerd door alle instances tegelijk random data naar de Cinder node te laten schrijven. Hier is goed te zien dat bij volledige belasting na 10~11instances de performance dramatisch achteruit gaat.
De conclusie bij deze test is dat afhankelijk van de opslagruimte die de block storage node bevat en de instances toebedeeld krijgen het omslagpunt wordt bepaald, de performance die de Cinder node moet bieden vormt geen bottleneck bij normaal (niet constant volledige load) gebruik en bij gebruik van 250GB HDD's.



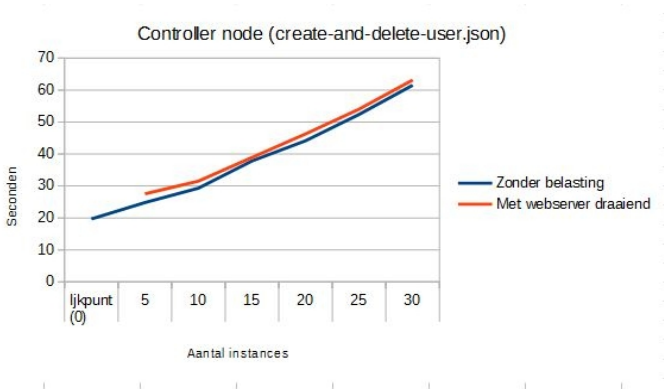
Afbeelding 24: De uitkomst van de performance/capacity test van de network node (Rally meetpunt).



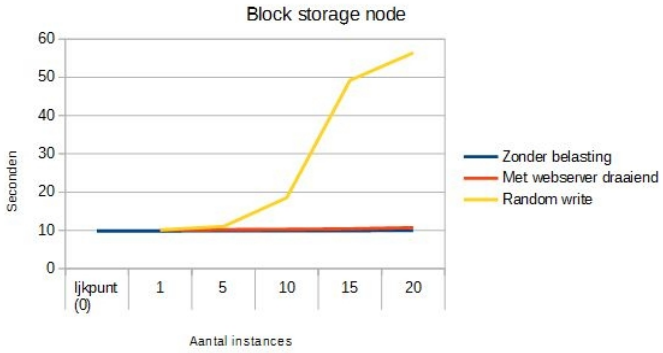
Afbeelding 25: De uitkomst van de performance/capacity test van de network node (download meetpunt).

Bij de test van de network node is de realistische load ook gegenereerd door instances Apache AB en MySQL-writes te laten uitvoeren. Er waren in deze test twee meetpunten: het eerste meetpunt was om de resources van de network node te testen en werd gemaakt door Rally de “create-and-delete-networks.json” benchmark te laten uitvoeren (nulpunt 13,703 seconde, omslagpunt 27,406 seconde), het tweede meetpunt werd gevormd door een testbestand van 10MB van een mirror te downloaden (nulpunt 4,377 seconde, omslagpunt 8,754 seconde). Het meetpunt die bij het minste aantal instances 200% slowdown laat zien gaf het eindtotaal aan. Uiteindelijk was het download meetpunt degene die als eerste 200% slowdown aangaf, dit was bij 26 instances. De uitslagen van de beide meetpunten is te zien op afbeelding 24 en 25. De conclusie bij

deze test is dat de belasting van de network node zelf niet snel een bottleneck zal vormen. De bottleneck die gevormd wordt in de network node is de datadoorvoer over zijn netwerkinterface, en dat gebeurd bij 26 actief netwerk genererende instances.



Afbeelding 22: De uitkomst van de performance/capacity test van de controller node.



Afbeelding 23: De uitkomst van de performance/capacity test van de block storage node.

10.1.12 Management network security

Als laatste tests werd de security van het management network onder de loep genomen. Ongewenste personen van buiten de opstelling mogen niet in het management netwerk van de opstelling komen. Om dit te controleren zijn de “toegangswegen” naar de interne netwerken, de DMZ-nodes, gecontroleerd.
Allereerst is met portscans gekeken of de porten van de DMZ-nodes op de i3D-netwerk interfaces goed ingesteld waren. De enige porten die hier mochten openstaan waren de porten die gebruikt werden voor de dienst van de DMZ-node. De SSH-jump node mocht port 50385 (SSH) open hebben staan, de controller node mocht port 443 (HTTPS) open hebben staan en de NTP-node mocht port 123 (NTP) open hebben staan.
Door het uitvoeren van drie verschillende portscans (TCP SYN, TCP connect en FYN) met Nmap zijn de open porten achterhaald. De uitkomst was precies zoals werd verwacht: SSH-jump node – alleen port 50385 open, controller node – alleen port 443 open, NTP-node – alleen port 123 open. Dit onderdeel van de test was geslaagd.
Het volgende onderdeel was het testen van de SSH-jump node en de controller node. De SSH-jump node is getest door te proberen in te loggen via normale inlognaam/password combinatie, dit zou echter niet meer mogelijk mogen zijn. Bij deze manier van inloggen gaf Putty een fout, het inloggen was niet mogelijk. Hierna is er ingelogd met een SSH-key als authenticatie, dat was wel mogelijk en er moest een passphrase worden opgegeven. Dit was precies zoals het onderdeel zou moeten werken.
Tenslotte is de controller node gecontroleerd door verbinding te maken via HTTP. Het dashboard mag deze HTTP-verbinding niet accepteren omdat er alleen verbinding gemaakt mag worden via HTTPS. De HTTP-verbinding gaf een error dat er geen gebruik gemaakt mocht worden van HTTP.
Tenslotte is er ook gecontroleerd of er ingelogd kon worden via HTTPS: deze verbinding werd door dashboard geaccepteerd.
De uitkomsten van de tests waren allemaal zoals ze verwacht werden en beschreven werden in het testplan. De test is positief. De conclusie bij deze test is dat de toegang voor onbevoegden op het management netwerk in ieder geval heel moeilijk te verkrijgen is.

10.2 Stroomverbruik hardware

Het stroomverbruik van de servers moest ook worden berekend om mee te nemen in het proof of concept. In de berekening werd uitgegaan van 3 niveau's: idle, 50% belasting en 100% belasting. Voor deze test is de utility node genomen. Als

meetpunt van het aantal ampère werd de meter in het rack gebruikt. Het nadeel van dit meetpunt was echter wel dat het heel het rack meet en een getal geeft met 1 decimaal.
Als load generator is het programma *Stress-ng* gebruikt, met dit programma is op commando load te genereren. Er is eerst gemeten met de utility node uitgeschakeld. Hierna werd er gemeten met de utility node aan en zonder belasting, 50% belasting met Stress-ng en 100% belasting met Stress-ng.
Met de utility node uitgeschakeld gaf de meter 4,87A aan. Met de utility node op idle gaf de meter 5,15A aan. De utility node gaf 5,34A en 5,42 aan op 50% load en 100% load.
De berekening die het stroomverbruik zou berekenen is: (stroomverbruik node met belasting – stroomverbruik uitgeschakeld). Idle had de node 0,28A verbruik, met 50% load had de node 0,47A verbruik, met 100% load had de node 0,55A verbruik. Deze uitkomsten geven de waarden in ampère terwijl in de winstberekening gerekend wordt met watt. De waarden in ampère moeten worden vermenigvuldigd met 230(volt) om tot het wattage te komen. Idle was het wattage 69W, met 50% load was het wattage 119,6W, met 100% load was het wattage 149,5W.
De uitkomst van de test voor het stroomverbuik van de nodes was op deze manier nog niet compleet. Deze servers is namelijk opgestart met een HDD. Van de uitkomst van 149W moet de waarde van 9,1225W van één HDD worden afgehaald zoals uitgelegd staat in onderstaand stuk tekst.

Naast de servers is ook het stroomverbuik van losse HDD's achterhaald door online benchmarks te zoeken[50][51][52][53]. Er waren een aantal verschillende soorten HDD's aanwezig in het proof of concept. Van alle soorten HDD's is het stoomverbruik achterhaald. Vervolgens is het gemiddelde stroomverbruik genomen van alle verschillende soorten HDD's en hun state (idle en read/write). Het gemiddelde stroomverbruik was 9,1225W.

11. Winstberekening

De winstberekening is een van de belangrijkste documenten die moet worden opgeleverd binnen het project. Het doel van de winstberekening is om uit te vinden of er winst gemaakt kan worden met een low-cost cloud oplossing op basis van het proof of concept (eindproduct van dit project). De winstberekening kan worden gevonden in bijlage V.

De winstberekening bevat 3 onderdelen: de kosten, de baten en de winstberekening.

11.1 Kosten

In het kostenonderdeel van de winstberekening worden de kosten van hardware, onderdelen, verbruik en huur.
Het bovenste gedeelte van de berekening geeft aan welke hardware er gebruikt wordt voor het proof of concept en in een cloud oplossing op basis van het proof of concept.
De kolom “aanschaf” geeft de dagwaarde van het onderdeel aan. Om achter de dagwaarde van een onderdeel te komen is er op verschillende websites gekeken waar deze onderdelen nog tweedehands aan werden geboden, dat is ook de reden waarom deze waardes zo laag liggen. Bij deze aanschafwaardes gaat het om de dagwaarde die een product heeft, de berekening aan de hand van deze waardes volgt echter pas in de winstberekening.
Verder staan ook enkele benodigde specificaties in het hardware gedeelte. Het aantal cores en aantal GB geheugen van de servers is nodig in onderstaande berekeningen.
De verbruiksvariabelen van de servers zijn achterhaald door te testen (subhoofdstuk 10.2), de verbruiksvariabelen van de switches en HDD's zijn achterhaald door documentonderzoek. De uitkomsten van de verbruikstest van de nodes was in Ampère, deze uitkomsten zijn vermenigvuldigd met het aantal volt van het stroomnet (230V) om aan het aantal watt te komen.

Het volgende onderdeel zijn de vaste lasten. Dit onderdeel spreekt voor zich: i3D heeft kosten aan stroom, dataverbruik en rackhuur. De vaste lasten worden in dit onderdeel afgedrukt zodat ze gebruikt kunnen worden in de volgende berekeningen.

Hierna komt het overzicht van de OpenStack nodes. Dit onderdeel bestaat uit vijf delen: het aantal OpenStack nodes en switches, de netwerkinterfaces per node, de instances per node, het aantal HDD's per node en de stroomkosten die de nodes met zich meebrengen.
De waardes in de kolom “aantal nodes” worden gebruikt in verdere berekeningen, zoals in de berekening van de stroomkosten.
De waardes in de kolom “netwerk interfaces per node” worden gebruikt om de groei van het aantal switches te kunnen berekenen. Een switch heeft 48 poorten, de berekening is dan het totale aantal netwerkinterfaces van alle nodes delen door 48 en dan afronden naar boven, dat is het aantal switches.
De waardes in de kolommen “instances per node” en “totaal instances per node” worden gebruikt bij het berekenen van de groei. De uitkomsten van de performance groei test worden dan ook hier gebruikt. Aan de hand van het aantal instances die actief zijn op de compute node kan worden berekend of er een ander type node bijgeplaatst moet worden in de berekening. Als het totaal aantal instances die actief kunnen zijn op de: network node, controller node, block storage node of object storage node lager uitkomt dan het aantal instances op de compute node moet er van dat type node een worden bijgeplaatst. Een voorbeeld: er staan in de opstelling 8 compute nodes (= max. 56 instances, want 7 per compute node) en er staan 2 network nodes (kan max 52 instances aan) dan moet er een network node worden bijgeplaatst in het overzicht.
De waardes in de kolommen “aantal HDD's per node” en “aantal HDD's totaal” worden gebruikt in verdere berekeningen, zoals in het verbuik van de nodes.
De laatste vier kolommen gaan over stroomverbruik van de servers en switches, dit is het verbruik van In de kolom “Verbruik” staat het gemiddelde verbruik van de servers en switches, deze waardes worden vervolgens in KiloWatt per jaar berekend en hierna in KiloWatt per maand, hier is ook het verbruik van de HDD's bij opgeteld (aantal HDD's per soort node + gemiddeld verbruik node). De kilowatt per maand wordt vervolgens door middel van de stroomkosten per Kwh omgezet in stroomkosten per maand. De uitkomst helemaal rechts onderin is de uitkomst van de stroomkostenberekening.

Alle nodes in het kostenvlak zijn minimaal dubbel uitgevoerd. Dit is gedaan omdat een productieomgeving van OpenStack

altijd minimaal 1 keer redundant de nodes uitgevoerd heeft staan.

Het kosten onderdeel bevat echter veel verschillende eenheden, zoals: € per maand, € per Mb, Kwh. Door de ruime verscheidenheid aan eenheden is het van belang dat alle kosten naar één enkele eenheid worden gebracht, dat gebeurd in het derde onderdeel: “de berekening”.

11.2 Baten

Het volgende onderdeel is het baten onderdeel. Hier worden de baten berekend aan de hand van een standaard prijs voor een instance. Er is gekozen om in de berekening één OpenStack flavor aan te houden waar het hele baten onderdeel op is gebaseerd. Wat deze standaard flavor inhoudt is te zien in de rij achter “standaard flavor”, dit is: 1 virtuele CPU core, 1GB geheugen, 20GB block storage en 30GB object storage. De prijs is samen met de opdrachtgever afgestemd en is €12,50.
Onder de kop “klanten per compute node” wordt vervolgens de berekening gemaakt hoeveel klanten met de huidige overprovisie op één compute node geplaatst kunnen worden op basis van de standaard flavor.
De kolom “fysiek” geeft de fysieke resources van de compute node aan. De kolom “overprovisie” geeft de oversell van de virtuele resources op de compute node aan. De kolom “virtueel” geeft de virtuele resources aan. Uiteindelijk wordt in de kolom “aantal klanten” de berekening gemaakt hoeveel klanten er op een compute node geplaatst kunnen worden. In deze berekening wordt echter wel rekening gehouden met 1GB geheugen en ½ core voor de hypervisor van de node.
Tenslotte kan er aan de hand van de standaardprijs en het maximaal aantal klanten per compute node worden berekend wat er verdient kan worden met één compute node.

11.3 Winstberekening

In het derde onderdeel, de winstberekening, worden alle benodigde variabelen omgerekend tot euro's per maand. De kosten van van de hardware dagwaarde wordt hier neergezet als euro per maand, dit kan worden gezien als huur per maand over de stukken hardware die i3D zou moeten betalen waarbij de “afbetalingsperiode” een jaar bedraagt.
In deze berekening blijft de rack huur een vaste last aangezien de racks niet afbetaald zullen gaan worden. De berekening van de rack huur wordt gedaan op basis unit bezetting. Een rack bestaat uit 48 U's hoogte (er passen 48 stukken hardware met 1 U hoogte in). De rack huur is dus op basis van het aantal U's wat de cloud oplossing in beslag neemt. De berekening is dus: rack huur per maand delen door aantal U's van het rack en deze uitkomst vermenigvuldigen met het aantal stukken hardware (elk stuk hardware neemt 1U in beslag).

Uiteindelijk kan onderaan de winstberekening worden gezien of de oplossing winstgevend zal gaan worden of niet. Er is een berekening gemaakt voor een minimale productieopstelling en een berekening voor een productieopstelling die een heel rack in beslag neemt.

Onder de minimale productie-omgeving wordt verstaan: 2 network nodes, 3 compute nodes, 2 controller nodes, 2 block storage nodes, 1 NTP-nodes, 1 repository node, 1 SSH-jump node, 1 utility node, 3 object proxy servers en 2 objects proxy nodes. In deze omgeving zijn alle aansturende nodes dubbel uitgevoerd zodat ze een failover hebben. Het aantal compute nodes en object storage nodes is overgenomen uit het proof of concept.

De productieopstelling die een heel rack in beslag neemt komt op basis van de uitkomsten uit de testfase uit op: 5 network nodes, 17 compute nodes, 8 controller nodes, 3 block storage nodes, 1 NTP-node, 1 respository node, 1 SSH-jump node, 1 utility node, 5 object storage nodes, 2 object proxy servers en 4 switches.

De minimale productieomgeving draait volgens de winstberekening nog verlies, echter als er wordt opgeschaald naar een omgeving die een heel rack in beslag neemt kan er winst worden gemaakt. De gehele winstberekening is gebaseerd op het sunny day scenario dat iedereen dezelfde flavor kiest en dat de compute nodes vrijwel direct vol zitten (de hardware staat niet onnodig aan). Als deze opstelling echt uitgevoerd zou worden dan zou de winst toch iets lager uitkomen dan in de berekening.

12. Evaluatie

In dit hoofdstuk word het project geëvalueerd. In het subhoofdstuk “resultaten” wordt de uitkomst van het project, oftewel de conclusie besproken. Verder worden het proces, de verantwoording van de beroepstaken en de aanbevelingen aan i3D besproken.

12.1 Resultaten

Na het uitvoeren van het project is de conclusie dat er winst gemaakt kan worden op basis van een low-cost cloud oplossing op basis van OpenStack en de oude DL120 G6 servers. De uitkomst van de winstberekening op basis van de minimale productieomgeving gaf nog een verlies aan waar de winstberekening op basis van een geheel gevuld rack een winst aangaf. Er is dus winst te maken alleen moet de omgeving dan wel in het groot worden opgezet, hoe meer de opstelling groeit hoe meer potentie er is om winst te maken.

12.2 Proces

Het project is uitgevoerd volgens de sashimi waterval methode. Deze methode houdt de structuur van de normale watervalmethode aan met een overlap in de fases: er kan dus worden teruggekeerd naar een eerdere fase.
In het project heb ik een keer terug moeten keren naar een eerdere fase omdat er nog iets moest worden aangepast. Dit vond plaats in de ontwikkelingsfase. Bij het opbouwen van het proof of concept kwam ik er achter dat het topologie-ontwerp fouten bevatte, deze heb ik moeten verbeteren. Tijdens de oplevering van het infrastructuurontwerp aan de opdrachtgever was dit ook niet naar voren gekomen.

12.2.1 definitiefase

In de definitiefasefase van het project is de meeste tijd gaan zitten van alle fases in het project. Eerst was er de keuze voor een projectmethodiek. Door het feit dat ik op school vrij weinig ervaring heb opgedaan met verschillende methodieken maakte dat ik me er vrij ver in moest gaan verdiepen. De PRINCE2-methode had ik op een vorige stage gebruikt, daar had ik al wat ervaring mee. Ook met de waterval/sashimi-methode had ik door de opleiding heen ervaring opgedaan.

Na het schrijven van het plan van aanpak en de literatuurstudie te hebben opgepakt bleek dat OpenStack een vrij fors platform is. Er kan gemakkelijk een aantal weken worden gestoken in het onderzoeken van OpenStack heb ik gemerkt. Hier was al deels op gerekend, echter bleef ik bij een aantal onderwerpen in het onderzoek naar OpenStack hangen: zo was het bijvoorbeeld niet duidelijk hoe OpenStack nu precies zijn netwerkfunctionaliteiten regelt, hoe de interne netwerkscheidingen werken en hoe het floating-IP's toewijst.
Na een aantal dagen te hebben vastgezeten met deze vragen heb ik Chris Zonneveld gevraagd om hulp, hij had namelijk OpenStack al eens deels opgezet. Door een gesprek met Chris en Arjen was alles zo goed als opgehelderd en kon ik weer verder met het onderzoek.
Naast het onderzoek naar OpenStack is er ook onderzoek uitgevoerd naar Ceph in de definitiefase. Na een tijdje bleek ook Ceph een groot project te zijn waar je weken zo niet maanden onderzoek voor nodig hebt. De reden dat Ceph niet verder is meegenomen in het project is dan ook wel duidelijk: niet genoeg tijd om Ceph ook te onderzoeken en op te zetten. Bij de vergelijking van de OpenStack-native componenten met Ceph bleek dat de OpenStack componenten wellicht nog een betere keus zouden zijn op basis van de use case.
De definitiefase nam uiteindelijk wat meer tijd dan gepland.

De definitiefase had twee documenten als output: plan van aanpak en onderzoeksverslag.

Het plan van aanpak is geschreven om een duidelijk beeld te scheppen over de aanpak van het project en hiermee te peilen of de opdrachtgever akkoord is met de voorgestelde aanpak. Ook is er in het plan van aanpak opgenomen hoe het project

gemanaged zal gaan worden (bijvoorbeeld: wanneer worden er meetings gehouden en wat zijn de risico's en de oplossingen op deze risico's).
Aan de hand van de planning, die ook onderdeel is van het plan van aanpak, is in het project goed ingeschat wanneer ik voor of juist achter op schema liep. Mede door de planning heb ik met een afwijkingsplanning kunnen voorkomen dat het project uit zou lopen.

Het onderzoeksverslag is geschreven om het (literatuur)onderzoek naar OpenStack, Ceph en de opgeleverde hardware inzichtelijk te maken en antwoord te geven op deelvragen. Van OpenStack zijn alle relevante onderdelen besproken. Met dit literatuuronderzoek is de benodigde kennis opgedaan om OpenStack op te zetten en de opgeleverde hardware zo in te stellen dat deze het meest geoptimaliseerd liep.

12.2.2 Architectuurfase

Aan het begin van de architectuurfase was het nog de bedoeling dat door afkadering van de opdracht alleen de meest kritieke bottleneck redundant uitgevoerd zou moeten worden. Om te testen welke node de bottleneck was is er een proefopstelling opgezet om op te testen. Dit was ook de voornaamste reden waarom er een testonderdeel in de architectuurfase was opgenomen.
Door de vele fouten die opkwamen bij het opzetten van de proefopstelling was zo veel tijd verloren dat er ook geen sprake meer kon zijn van redundantie inbouwen op één plek. Het testonderdeel in de architectuurfase heb ik wel gewoon door laten gaan zodat in ieder geval de performance eisen getest konden worden. Aan de hand van de uitkomsten was het mogelijk om verbeteringen op performance vlak door te voeren in het infrastructuurontwerp.
De architectuurfase was door de vele fouten in de proefopstelling veel later afgerond dan gepland. Door dit achterlopen is er voor gekozen om een afwijkingsplanning te maken en de redundantie niet in te bouwen, zo zou er in de infrastructuurfase en ontwikkelfase ingelopen kunnen worden op de planning.

De architectuurfase had het architectuurontwerp als output. Omdat in de architectuurfase ook is gekozen om testen uit te voeren op een proefopstelling had de architectuurfase ook de proefopstelling en een gedeeltelijk testplan/testrapport als output.

In het architectuurontwerp is uiteindelijk beschreven wat het globale ontwerp van het proof of concept zou zijn. Aan de hand van het onderzoek van de proefopstelling in de architectuurfase zijn de nodige keuzes gemaakt in het architectuurontwerp op het vlak van performance. Alle systeemeisen zijn verantwoord door ten minste één oplossing met onderbouwing.

12.2.3 Infrastructuurfase

In de infrastructuurfase in het document infrastructuurontwerp gemaakt. Deze fase kon ik vrij snel en gemakkelijk doorlopen omdat er weinig complicaties om de hoek kwamen kijken bij het maken van het ontwerp. Aan het eind van de infrastructuurfase liep ik nog maar een klein stukje achter op de planning, dit kwam door het afwijkingsplan.

De infrastructuurfase had het infrastructuurontwerp als output. In dit infrastructuurontwerp is het architectuurontwerp op zo'n manier uitgewerkt dat het proof of concept aan de hand van dit ontwerp opgezet kon worden. Het opzetten van dit ontwerp was snel gebeurd waardoor ik weer iets inliep op de planning.

12.2.4 Ontwikkelfase

In de ontwikkelfase is het proof of concept daadwerkelijk opgebouwd. In de architectuurfase had ik al een proefopstelling opgezet, deze kon ik hier grotendeels verder ontwikkelen tot proof of concept. Ik had namelijk al rekening gehouden met het feit dat ik de proefopstelling door zou ontwikkelen dus waren de IP-adressen, ports, VLAN's en andere details zo goed als mogelijk vooruit gepland.

Wel kwam ik er in deze fase achter dat ik een fout had gemaakt in mijn infrastructuurontwerp, het topologie-ontwerp bevatte fouten. In dit geval was ik dus erg blij dat ik had gekozen voor het sashimimodel, zo kon ik namelijk een fase terug om de fout op te lossen. Als ik de normale watervalmethode, de ASI-methode of een andere starre lineaire ontwikkelmethode had

gebruikt zou ik door deze fout weer opnieuw moeten beginnen.

Bij het opzetten van het proof of concept kwamen uiteraard wel fouten en problemen omhoog. De grootste fouten waren allebei met OpenStack Swift. Bij de ene fout wilde de service niet opstarten, uiteindelijk bleek het systeem te vallen over een regel commentaar. De andere fout kwam voor bij het opzetten van de Swift-ring, deze wilde de object storage node niet verbinden. Na het opnieuw instellen van de ring en alle server een hard reboot te hebben gegeven waren de problemen opgelost. Beide fouten namen wel weer veel tijd in beslag. De manier hoe ik met deze problemen om ben gegaan is, als je door het bomen het bos niet meer kan zien moet stoppen en de volgende dag met een frisse blik de problemen oplossen. Deze problemen zijn allemaal beschreven in bijlage VII.

Uiteindelijk liep ik aan het eind van de ontwikkelfase iets voor op mijn afwijkingsplanning. De documenten en producten die de ontwikkelfase als output had waren: ongeteste proof of concept, handleiding, testplan.
Zoals verwacht kon ik inlopen op de planning in de infrastructuurfase en de ontwikkelfase, dit betekende dat ik in de testfase zelf een stukje op de planning voorliep.

De handleiding is geschreven om het proof of concept die wordt gecreëerd in dit project reproduceerbaar te maken. Tijdens het opzetten van de proefopstelling en tijdens het opzetten van het proof of concept is elke actie die is uitgevoerd om het systeem op te zetten gedocumenteerd. Naderhand kan OpenStack Liberty op basis van DL120 G6 servers dus weer opnieuw en identiek worden opgezet. Ook kan de handleiding worden gebruik om weer te geven hoe de testomgeving in het testplan er uit ziet. Aangezien veel tests het gehele proof of concept vereisen kan er worden verwezen naar de handleiding I.P.V. in het testplan opnieuw het hele proof of concept te beschrijven.
Het schrijven van het testplan zorgde wel voor enig afremmen van de snelheid waarmee de proefopstelling en proof of concept werden opgezet. Als er elke keer een stukje moet worden gedocumenteerd op het moment dat er een actie op de opstelling is uitgevoerd kan er nooit echt vaart worden gemaakt bij het opzetten.

Het testplan is geschreven omdat er in dit project getest moest worden. Bij een test is het van belang dat er van tevoren wordt bepaalt waarom er wordt getest en wat het doel is. Bij elke test heb ik geprobeerd om zo goed mogelijk weer te geven hoe de test uitgevoerd zou worden, wat het doel was, wat er van verwacht werd en wat de voorwaarden waren waarbij de test geslaagd zou zijn.

12.2.5 Testfase

Als laatste noemenswaardige fase was er de testfase. In deze fase heb ik de systeemeisen en performance groei (performancegrenzen van de nodes opzoeken) getest aan de hand van het testplan dat in de ontwikkelfase was gecreëerd. In de testfase zijn geen fouten of afwijkingen uit de tests naar voren gekomen en alle tests waren succesvol.
Aan het eind van deze fase liep ik iets voor op de afwijkingsplanning.
De documenten en producten die de testfase als output had waren: getest proof of concept en het testrapport.

Het testrapport is in hetzelfde document opgenomen als het testplan. De reden dat het testverslag is geschreven is zodat de uitkomsten van de test inzichtelijk zouden worden, om eventuele fouten tijdens de tests inzichtelijk te maken en om aan te geven of de systeemeisen succesvol getest waren.
Al met al is het testen en het schrijven van het testrapport vlot gegaan. Aan de hand van het testplan kon worden afgeleidt hoe de test opgezet moest worden en hoe er getest moest worden. Tijdens het testen was het eigenlijk alleen zaak om de test uit te voeren en de uitkomsten te documenteren.

12.3 Beroepstaken

Binnen het project was het de bedoeling om ook een aantal competenties of beroepstaken aan te tonen. Deze beroepstaken moeten verworven zijn in de studie en bij het afstuderen moeten ze gebruikt en aangetoond kunnen worden. Bij dit project is getracht om de volgende vier competenties aan te tonen:

- A1 – analyseren van het probleemdomein.

- C9 – Ontwerpen van een infrastructuur.
- D18 – Testen van een infrastructuur.
- G1 Praktische aspecten hanteren in (internationale) projecten.

De eerste beroepstaak, het analyseren van het probleemdomein, heb ik aangetoond door aan het begin van het project rond de tafel te gaan zitten met de opdrachtgever en het probleemdomein boven tafel te krijgen. Het probleemdomein is beschreven in het plan van aanpak. De taal die aangehouden is bij het analyseren van het probleemdomein is gewoon in het Nederlands.

De tweede beroepstaak, ontwerpen van een infrastructuur, is gestart met het achterhalen van de behoeften die de opdrachtgever had aan het proof of concept. De eisen waren vrijwel hetzelfde als de behoeften van de opdrachtgever aangezien de opdrachtgever de enige belanghebbende met inspraak in het project was.
Aan de hand van de opgestelde systeemeisen is er een (globaal) architectuurontwerp gemaakt. In dit architectuurontwerp zijn ook allerlei alternatieven benoemd bij de oplossingen van de systeemeisen.
De performance was een kritisch gedeelte in dit project daarom is er voor gekozen om de performance systeemeisen te testen op een basale proefopstelling. Met de uitkomsten uit deze test is er in de infrastructuurfase nog bij kunnen sturen op het vlak van de performance. Een voorbeeld van een onderdeel dat is bijgestuurd is de overprovisie van instance op de compute nodes, het maximaal aantal instances met de standaard flavor op een compute node was bijgesteld van 8 naar 7.
In de infrastructuurfase is vervolgens een gedetailleerde topologie en gedetailleerd ontwerp gemaakt op basis van het architectuurontwerp. Aan de hand van het infrastructuurontwerp is het proof of concept opgebouwd.

De derde beroepstaak, testen van een infrastructuur, is al gestart in de architectuurfase van het project aangezien daar al de performance eisen werden getest. Voor deze performance eisen is dus speciaal de proefopstelling opgebouwd. De overige eisen zijn allemaal in de testfase aan tests onderworpen.
Voor de start van alle tests is er een testplan opgesteld. In dit testplan werd voor elke test beschreven: welke eisen er werden getest, de manier van meten, uitkomsten die de test positief of negatief laten uitslaan, meting waarbij de test positief is en de vragen die met de test beantwoord worden.
In de testfase zijn alle, in het testplan beschreven, tests uitgevoerd. Het testrapport is in hetzelfde document opgenomen als het testplan, zo zit alles mooi bij elkaar. Het testverslag bevat: resultaten van de tests, verslag van de tests en de conclusies die daarbij getrokken kunnen worden. De documentatie die de testopstelling reproduceerbaar maakt in beschreven in de handleiding, vrijwel alle tests worden namelijk op het hele proof of concept uitgevoerd.

De laatste beroepstaak, praktische aspecten hanteren in (internationale) projecten, is aangetoond door het opstellen van het plan van aanpak met de volgende onderwerpen: de opdrachtomschrijving, de project aanpak en de risico's.
De aanpak is niet opgenomen in de opdracht omschrijving omdat deze al heel goed is omschreven in de inleiding van het document, als ik ook nog een aanleiding zou schrijven vormen er dubbele stukken tekst.

12.4 tegengekomen problemen

Bij het opzetten van de OpenStack omgeving zijn er veel fouten en problemen opgekomen. Deze problemen moesten goed worden opgelost. De problemen en de manier hoe deze zijn opgelost kunnen teruggevonden worden in bijlage VII.

12.5 Aanbevelingen

Tijdens het uitvoeren van het project zijn er een aantal aanbevelingen opgekomen die in een opvolgend project voor i3D van pas zouden kunnen komen.
Als eerste moet elke node minimaal dubbel uitgevoerd zijn in een productieopstelling op basis van OpenStack. Het mooie is dat redundantie in OpenStack ook meteen betekend dat er gebruik gemaakt kan worden van load balancing, het verdelen van de belasting over meerdere nodes. Het invoeren van meer redundantie bij de OpenStack nodes gebeurd met “pace maker”.

Een andere aanbeveling is het opzetten van offsite-back-ups. Op deze manier worden de back-ups niet op dezelfde plek opgeslagen als de andere data, en is het ook minder voor de hand liggend dat bij een ramp naast de data ook de back-ups van de data weg is.

Een hele belangrijke aanbeveling voor als i3D verder wil met het OpenStack project is het werken met snellere machines. Uit het project is gebleken dat OpenStack wel draait maar er kan veel meer uitgehaald worden bij gebruik van krachtigere machines. Een groot nadeel is de performance/stroomverbruik verhouding van de oude DL120 G6 servers. Met deze oude servers wordt relatief veel meer stroom verbruik dan met machines die een stuk moderner zijn. Ook is het aantal instances die gehost kunnen worden vanaf een compute node in het proof of concept zeer laag.
Door alleen al gebruik te maken van even oude DL160 G6 servers kan waarschijnlijk het dubbele aantal instances worden gehost op de compute nodes, dit komt omdat de DL160's 2 processoren en het dubbele aantal GB's aan geheugen aankunnen T.O.V. de DL120's.

Het opzetten van nieuwe OpenStack nodes heb ik in het proof of concept allemaal handmatig gedaan, het is echter een stuk gemakkelijker en ten zeerste aan te raden om een automation applicatie zoals Puppet of Chef op te zetten om nieuwe nodes uit te rollen.

Als laatste wordt de aanbeveling gedaan om de OpenStack componenten meer te verspreiden over de verschillende nodes, vooral de controller node bevat vrij veel services. Natuurlijk kan de controller net als in het proof of concept worden opgezet maar het is verstandiger om de services die veel belasting genereren een dedicated node te geven. Het is bijvoorbeeld verstandig om de databases waar de verschillende servies van afhankelijk zijn een eigen (redundant uitgevoerde) node te geven. Ook is het verstandig als er gebruik gemaakt gaat worden van metering om Ceilometer een eigen node te geven, deze service genereert zeer veel belasting.

Literatuurlijst

[1] Bert Hedeman, Gabor Vis van Heemst, Hans Fredriksz (2009). “Projectmanagement op basis van PRINCE2 editie 2009”. Eerste druk. Zaltbommel: Van Haren uitgevers. 2009.

[2] Sashimi-model. “sashimi-model”. Available: http://www.waterfall-model.com/sashimi-waterfall-model/. [Accessed: 13-01-2016].

[3] Ngi. “ASI-rapport: ontwerp en ontwikkeling van systeem-/netwerkinfrastructuren”. Ngi. Nijkerk. 1995.

[4] PMI Netherlands Chapter (2009). “A Guide to the Project Management Body of Knowledge (PMBoK Guide) 4th Edition - Nederlandstalige uitgave”. Vierde druk. Zaltbommel: Van Haren uitgevers. 2009.

[5] OpenStack. “overview of the OpenStack project”. Available: http://www.openstack.org/software/. [Accessed: 27-01-2016].

[6] OpenStack. “OpenStack community”. Available: http://www.openstack.org/community/. [Accessed: 27-01-2016].

[7] OpenStack. “OpenStack releases”. Available: http://www.openstack.org/software/liberty/. [Accessed: 28-01-2016].

[8] OpenStack. “OpenStack components”. Available: http://www.openstack.org/software/. [Accessed: 28-01-2016].

[9] Sharone Zitzman. “OpenStack components”. Available: http://getcloudify.org/img/blog/openstackdiagram.jpg. [Accessed: 01-02-2016].

[10] OpenStack. “OpenStack nodes”. Available: http://docs.openstack.org/openstack-ops/content/compute_nodes.html. [Accessed: 01-02-2016].

[11] OpenStack. “OpenStack example architecture”. Available: http://docs.openstack.org/openstack-ops/content/example_architecture.html. [Accessed: 01-02-2016].

[12] OpenStack. “example architecture with neutron networking hw”. Available: http://docs.openstack.org/juno/install-guide/install/apt/content/ch_overview.html#example-architecture-with-neutron-networking-hw. [Accessed: 01-02-2016]

[13] OpenStack. “Cloud controller design”. Available: http://docs.openstack.org/openstack-ops/content/cloud_controller_design.html. [Accessed: 01-02-2016].

[14] OpenStack. “Example architecture”. Available: http://docs.openstack.org/liberty/install-guide-ubuntu/overview.html#example-architecture. [Accessed: 02-02-2016].

[15] OpenStack. “Designing for cloud controllers and cloud management”. Available: http://docs.openstack.org/openstack-ops/content/cloud_controller_design.html. [Accessed: 02-02-2016].

[16] OpenStack. “Example architecture”. Available: http://docs.openstack.org/liberty/install-guide-ubuntu/overview.html#example-architecture. [Accessed: 02-02-2016].

[17] OpenStack, OpenStack operations guide. OpenStack foundation, 2014. [E-book] Available: http://docs.openstack.org/openstack-ops/openstack-ops-manual.pdf [Accessed: 03-02-2016].

[18] OpenStack. “OpenStack storage decisions”. Available: http://docs.openstack.org/openstack-ops/content/storage_decision.html. [Accessed:03-02-2016].

[19] Yadin Porter De Leon, Tony Piscopo. "Object Storage versus Block Storage: Understanding the Technology Differences". Available: http://www.druva.com/blog/object-storage-versus-block-storage-understanding-technology-differences. [Accessed: 03-02-2016].

[20] OpenStack. “OpenStack network design”. Available: http://docs.openstack.org/openstack-ops/content/network_design.html. [Accessed: 03-02-2016].

[21] OpenStack. “Technical considerations”. Available: http://docs.openstack.org/arch-design/generalpurpose-technical-considerations.html#designing-network-resources. [Accessed 04-02-2016]

[22] OpenStack. “How to get OpenStack”. Available: https://wiki.openstack.org/wiki/Get_OpenStack. [Accessed: 04-02-2016].

[23] OpenStack. “Hypervisor support matrix”. Available: https://wiki.openstack.org/wiki/HypervisorSupportMatrix. [Accessed: 04-02-2016].

[24] OpenStack. “OpenStack SQL-databases”. Available: http://docs.openstack.org/liberty/install-guide-ubuntu/environment-sql-database.html. [Accessed: 04-02-2016].

[25] OpenStack. “OpenStack messagequeue”. Available: http://docs.openstack.org/developer/nova/rpc.html. [Accessed: 05-02-2016].

[26] OpenStack. “AMQP and Nova”. Available: http://docs.openstack.org/developer/nova/rpc.html. [Accessed 05-02-2016].

[27] O.S. Tezer . “AMQP message queue”. Available: https://www.digitalocean.com/community/tutorials/an-advanced-message-queuing-protocol-amqp-walkthrough. [Accessed: 05-02-2016].

[28] Peter Ledbrook. “Understanding AMQP, the protocol used by RabbitMQ”. Available: http://blog.springsource.com/wp-content/uploads/2010/06/rabbit-basics.png. [Accessed: 05-02-2016].

[29] Steve Martinelli, et al., Identity, Authentication, and Access Management in OpenStack . Sebastopol: O’Reilly Media, Inc., 2015.

[30] Ken Pepple, Deploying OpenStack. Sebastopol: O’Reilly Media, Inc., 2011.

[31] James Denton, Learning OpenStack Networking (Neutron), 2014. [E-book] Available: https://www.safaribooksonline.com/library/view/learning-openstack-networking/9781783983308/index.html [Accessed: 08-02-2016].

[32] OpenStack. “OpenStack storage decisions”. Available: http://docs.openstack.org/openstack-ops/content/storage_decision.html. [Accessed: 08-02-2016].

[33] OpenStack. “Cinder support matrix”. Available: https://wiki.openstack.org/wiki/CinderSupportMatrix. [Accessed: 09-02-2016].

[34] Joe Arnold. “OpenStack Swift”. Available: https://www.safaribooksonline.com/library/view/openstack-swift/9781491903841/. [Accessed: 10-02-2016].

[35] OpenStack. “OpenStack Rally”. Available: http://rally.readthedocs.io/en/latest/overview.html. [Accessed: 21-03-2016].

[36] OpenStack. “OpenStack telemetry”. Available: https://wiki.openstack.org/wiki/Telemetry. [Accessed:21-03-2016].

[37] OpenStack. “Ceph: The De Facto Storage Backend for OpenStack”. Available: https://www.openstack.org/summit/openstack-summit-hong-kong-2013/session-videos/presentation/ceph-the-de-facto-storage-backend-for-openstack. [Accessed: 15-02-2016]

[38] Ceph. “Hardware Recommendations”. Available: http://docs.ceph.com/docs/hammer/start/hardware-recommendations/. [Accessed: 22-02-2016]

[39] Ceph. “Zero To Hero Guide : : For CEPH CLUSTER PLANNING”. Available: http://ceph.com/planet/zero-to-hero-guide-for-ceph-cluster-planning/. [Accessed: 22-02-2016].

[40] OpenStack. “Before you begin”. Available: http://docs.openstack.org/icehouse/install-guide/install/apt/content/before-you-begin-swift-install.html. [Accessed: 23-02-2016].

[41] Christian Huebner. “Ceph vs Swift – An Architect’s Perspective”. Available: https://www.mirantis.com/blog/cephvsswiftarchitectsperspective/. [accessed 24-02-2016]

[42] Intel. “Intel Xeon processor X3430 specifications”. Available: http://ark.intel.com/nl/products/42927/Intel-Xeon-Processor-X3430-8M-Cache-2_40-Ghz. [accessed 23-02-2016]

[43] HP (14-11-2011). “Dl120 G6 manual”. Available: http://images10.newegg.com/UploadFilesForNewegg/itemintelligence/HP/13504_na1400508550568.pdf. [accessed 25-02-2016]

[44] Dell (2013). “Powerconnect 5548 specifications”. Available: http://www.dell.com/downloads/global/products/pwcnt/en/powerconnect-5500-series-spec.pdf. [accessed 25-02-2016]

[45] NIST (2004). “Electronic authentication guideline”. Available: “http://wayback.archive.org/web/20040712152833/http://csrc.nist.gov/publications/nistpubs/800-63/SP800-63v6_3_3.pdf. [accessed 09-03-2016]

[46] Zdnet. “OpenStack's top operating system: Ubuntu Linux”. Available: http://www.zdnet.com/article/openstacks-top-operating-system-ubuntu-linux/. [Accessed: 04-02-2016].

[47] OpenStack. “Database back end considerations”. Available: http://docs.openstack.org/security-guide/databases/database-backend-considerations.html. [Accessed: 08-02-2016].

[48] OpenStack. Choosing a database backend”. Available: http://docs.openstack.org/developer/ceilometer/install/dbreco.html. [Accessed: 08-02-2016].

[49] OpenStack. “Configure the Oslo RPC messaging system”. Available: http://docs.openstack.org/kilo/config-reference/content/configuring-rpc.html. [Accessed: 09-02-2016].

[50] Tech report. “Western digital re3 benchmark”. Available: http://techreport.com/review/15588/western-digital-re3-hard-drive/9. [Accessed: 18-05-2016].

[51] Tech report. “Seagate VB0250eaver benachmark”. Available: http://techreport.com/review/16472/seagate-barracuda-7200-12-hard-drive/11 . [Accessed: 18-05-2016].

[52] Tech report. “Seagate Barracuda ES.2 benchmark”. Available: http://techreport.com/review/13732/seagate-barracuda-es-2-hard-drive/14 . [Accessed: 18-05-2016].

[53] Tech report. “Seagate Barracuda ES benchmark”. Available: http://techreport.com/review/13732/seagate-barracuda-es-2-hard-drive/15 . [Accessed: 18-05-2016].

Bijlage I: Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf (*structuur niet wijzigen***)**

Afstudeerblok: 2016-1.1 (start uiterlijk 8 februari 2016)
Startdatum uitvoering afstudeeropdracht:
Inleverdatum afstudeerdossier volgens jaarrooster: 3 juni 2016

Studentnummer: 09021000
Achternaam: dhr Holtkamp
Voorletters: FJ
Roepnaam: Jeroen
Adres: Prins Hendrikstraat 60
Postcode: 2291ES
Woonplaats: Wateringen
Telefoonnummer: 0174296047
Mobiel nummer: 0612233622
Privé emailadres: fj.holtkamp@gmail.com

Opleiding: Technische informatica
Locatie: Delft
Variant: voltijd

Naam studieloopbaanbegeleider: dhr M. Maris
Naam begeleidend examinator: dhr J.D. Schagen
Naam tweede examinator: dhr M. Maris

Naam bedrijf: I3D.net
Afdeling bedrijf:
Bezoekadres bedrijf: Van Nelleweg 1
Postcode bezoekadres: 3044BC Rotterdam
Postbusnummer:
Postcode postbusnummer:
Plaats: Rotterdam
Telefoon bedrijf: +31 (0)10 890 00 70
Telefax bedrijf:
Internetsite bedrijf: http://www.i3d.nl/

Achternaam opdrachtgever: dhr Sloot
Voorletters opdrachtgever: R
Titulatuur opdrachtgever:
Functie opdrachtgever: COO
Doorkiesnummer opdrachtgever:
Email opdrachtgever: ricksloot@i3d.nl

Achternaam bedrijfsmentor: dhr Sloot
Voorletters bedrijfsmentor: R
Titulatuur bedrijfsmentor:

Functie bedrijfsmentor: COO
Doorkiesnummer bedrijfsmentor:
Email bedrijfsmentor: ricksloot@i3d.nl

NB: bedrijfsmentor mag dezelfde zijn als de opdrachtgever

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:

Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low cost cloud oplossing te creëren bij i3D.

Opdrachtomschrijving

1. Bedrijf

i3D.net is een managed hosting provider die is gevestigd in Rotterdam. Klanten die worden bediend komen vooral uit de sectoren: gaming, overheid, onderwijs, gezondheidszorg, sport, web-shop, hosting en print media. Wereldwijd bestaat i3D uit 16 datacenters waar 31.000 klanten gebruik van maken. In het datacenter in Rotterdam zijn 40 mensen werkzaam.
Het bedrijf is relatief nieuw (2004). Ook de groei is explosief te noemen: i3D.net heeft al enkele prijzen gewonnen waaronder voor het snelstgroeiende winstgevende bedrijf in Zuid-Holland. Tevens is i3D.net al vier jaar op een rij opgenomen in de Deloitte Fast 500 EMEA, als snel-groeiend technologie bedrijf.
De opdracht zal worden gedaan op de afdeling datacenter / operations.

2. Probleemstelling

i3D.net heeft momenteel een Cloud oplossing op basis van Azure Pack. Het doel van i3D.net is daar een low-costs variant naast te gaan zetten met als doel de oude hardware te gebruiken die reeds in bezit is en eerder zijn ingezet voor gameservers en dedicated servers.

Binnen het datacenter van i3D.net is een groot aantal servers te oud geworden om te kunnen meedraaien voor de primaire activiteiten. Binnen de primaire activiteiten zijn de servers ingezet als gameserver of als dedicated server voor klanten. Het gaat over grote aantallen servers die bij elkaar bij aanschaf veel geld hebben gekost. I3D.net is om die reden op zoek naar een mogelijkheid om de servers voor secundaire doeleinden te gebruiken. Ook worden deze oudere servers met de tijd steeds minder waard. Nu de servers wellicht nog enige waarde hebben kan er worden gekeken of er een oplossing te vinden is voor de servers of dat het beter zou zijn ze te verkopen.
In een eerder onderzoek is onderzocht of de opensource cloud oplossing “Openstack” werkend gekregen kan worden op deze oude servers. Het gebruiken van de servers om openstack te draaien kan een oplossing zijn om niet direct alle apparatuur compleet af te hoeven schrijven, of met flinke afschrijving te verkopen. De conclusie van dit onderzoek was dat openstack kan werken op de servers. Met de conclusie van dit onderzoek is er voor i3D.net een plan opgekomen waar de oude servers ingezet zouden kunnen worden als low-cost variant cloud oplossing. Als cloud oplossing worden nu servers op basis van Azure packs aangeboden. De low-cost cloud oplossing zou dan opgezet moeten worden met Openstack (en eventuele uitbreidingen om meer cloud services te kunnen aanbieden).

Uiteraard zijn er nog veel meer vragen die gesteld moeten worden voordat de cloud oplossing

gecreëerd kan worden (analyse):

- Kan er theoretisch gezien winst worden gemaakt met de low cost cloud oplossing? Hierbij moet bedacht worden dat de servers al ouder zijn en beschikken over minder performance. In een cloud oplossing is het toch vaak of altijd zo dat de performance een deel van de service bepaald. Als de servers niet voldoende blijken te zijn om winst op te leveren dan zal er een onderzoek plaats moeten vinden om servers te vinden waar dit wel mee bereikt kan worden.
  Ook de stroomkosten spelen in dit plaatje een aanzienlijk aandeel, deze zullen ook onderzocht moeten worden.
- Kan de storage/dienst worden uitgebreid (en hiermee ook eventuele winst) met opensource software? En met welke software kan er eventueel worden uitgebreid?
- Kan een netwerk van de servers worden gebruikt als SDN? Hierbij moet wel worden gekeken of er nog aanvullende apparaten benodigd zijn als de mogelijkheid bestaat.

3. Doelstelling van de afstudeeropdracht

De uiteindelijke doelstelling van de opdracht is een proof of concept van een low-cost cloud oplossing/cloud storage. Deze cloud oplossing moet draaien op open source software en op de oude servers die niet meer ingezet kunnen worden als primaire servers. Tevens moeten de kosten en opbrengsten van de servers inzichtelijk zijn gemaakt.

Er kan een mogelijkheid bestaan dat de servers zo verouderd zijn dat ze niet meer (winstgevend) ingezet kunnen worden in een low cost opstelling. In dit geval zal er een alternatief moeten worden gevonden. Hier moet dan worden gekeken welke specs een winstgevend geheel zullen vormen en welke servers hiervoor aangeschaft zullen moeten worden.

Na de analyse zal er ook een proof of concept van de low cost cloud oplossing (storage) neergezet moeten worden. Voor het bouwen van het proof of concept zal er eerst onderzocht moeten worden hoe dit gedaan kan worden. Vragen hierbij zullen zijn:

- Hoe zijn de servers te koppelen?
- Kunnen resources worden gedeeld?
- Hoe is de storage werkende te krijgen zodat het gebruik maakt van alle servers?
- Hoe kan Openstack gekoppeld worden met Ceph RADOS. Hoe krijgen we dit voor elkaar? (Ceph is een goed schaalbaar, open source en gedistribueerd storage system.)

Om het proof of concept te bouwen moet er een cluster van servers worden gebouwd. Om aan te tonen dat het proof of concept wel of niet werkt wordt er een cluster van een rack met servers gebouwd i.p.v. direct alle servers.
Omdat het verband tussen performance en prijs in dit onderzoek een groot punt is zal ook gekeken moeten worden naar verschillende soorten disks in de aangeboden oplossing (SSD, SCSI, SATA).

Tijdens en na het bouwen van het proof of concept zal de opstelling ook moeten worden getest. Redundantie is in de storage zeer belangrijk. Tevens is de performance van de opstelling (verschillende soorten schijven) van groot belang, deze zal extra aandacht krijgen in de test.

4. Resultaat

Met de conclusie van de opdracht weet i3D.net of er een low-cost oplossing opgezet kan worden met de oude servers. Ook zal duidelijk zijn hoe deze oplossing tot stand is gekomen. Moet er

bijvoorbeeld nieuwe apparatuur worden toegevoegd om het proof of concept werkende te krijgen? De kosten en opbrengsten van de oude servers is op dit moment ook inzichtelijk. Met deze informatie kan i3D beslissen om ook echt een low-cost cloud oplossing neer te zetten of alle oude servers te verkopen.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

- Werkplek opzetten en voorbereidende werkzaamheden (1 dag)
- Verdiepen in de projectmethode (4 dagen)
- Eisen vaststellen (2 dagen)
- Plan van aanpak opstellen (4 dagen)
  - Risicoanalyse schrijven
- Analyse (14 dagen)
  - Beschikbare documentatie zoeken over de servers (kosten, performance, etc)
  - Onderzoek doen naar de technische aspecten van de servers
  - Onderzoek doen naar theoretische kosten en opbrengsten van het proof of concept
  - Onderzoeken of en de storage op het cluster van servers ook gebundeld kan worden met opensource software (bijvoorbeeld Ceph)
  - Onderzoeken of SDN toegepast kan worden in een netwerk van de oude servers
  - Onderzoeken hoe het proof of concept op te zetten is
- Engineers interviewen voor de opbouw van het cluster (1 dag)
- Servers (cluster) opbouwen (15 dagen)
- Testplan schrijven (3 dagen)
- Servers (cluster) testen (5 dagen)
- Testrapport schrijven (2 dagen)
- Handleiding schrijven om proof of concept reproduceerbaar te maken (4 dagen)
- Afstudeerverslag schrijven (15 dagen)

6. Op te leveren (tussen)producten

- Plan van aanpak
- Analyserapport
- Interviewverslag
- Handleiding
- Testplan / testrapport
- Proof of concept

7. Te demonstreren competenties en wijze waarop

A1 – Analyseren van het probleemdomein

Voordat het plan van aanpak kan worden geschreven is het van belang dat het probleemdomein van de afstudeeropdracht goed duidelijk is. Binnen de opdracht zal het probleemdomein worden geanalyseerd.

Met behulp van netwerkschema's en andere middelen om te verduidelijken zal het probleemdomein worden beschreven.

C9 - Ontwerpen van een infrastructuur

In de opdracht is het de bedoeling om een cluster van servers te maken die samen kunnen werken. De samenwerkende servers moeten dan een clous oplossing bieden. Voordat het cluster gemaakt kan worden moet het eerst worden ontworpen.
In het ontwerp wordt opgenomen:

- Een topologie
- IP adresplan
- Configuratie van de servers en de software
- De geïnstalleerde software
- De gebruikte hardware in de servers (bijvoorbeeld welke schrijven (performance)).
- Kosten- en batenanalyse

D18 Testen van een infrastructuur

Het testen van het proof of concept is zeer wenselijk. Van het storage cluster is voornamelijk van belang dat de redundantie (failover) wordt getest en de performance van de verschillende opslagmedia.
Eerst zal er een testplan worden geschreven. Hoe gaat de test er uitzien en wat moet er worden getest? De testopstelling is in meer of mindere mate al aanwezig als zijnde het proof of concept. Deze opstelling zal wel extra worden toegelicht als dit noodzakelijk is.
Na de uitvoer van de geplande metingen zal hier een testrapport over worden opgesteld. Dit testrapport zal dus ook de conclusies bevatten die antwoord geven op de aanleiding van de test.

G1 Praktische aspecten hanteren in (internationale) projecten

Om de praktische aspecten van het project te hanteren zullen deze in het plan van aanpak worden beschreven. Het gaat hier om:

- Aanleiding
- Probleemstelling
- Doelstelling
- (Beoogde) resultaten
- Opdrachtomschrijving
- Aanpak
- Risicoanalyse

Al de bovenstaande punten zullen in detail worden uitgewerkt

Bijlage II: Behoeften en eisen

Behoeften en Eisen

Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low-cost cloud oplossing te creëren bij i3D

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Stagebedrijf: i3D
Plaats: Rotterdam
Datum: 25-01-2016

Versiebeheer

Versie	Datum	Wijziging
0.1	25-01-2016	Eerste versie, Layout
1.0	30-03-2016	Uiteindelijke versie

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

1. Behoeften van de belanghebbende

Binnen dit hoofdstuk worden de achterhaalde behoeften van de belanghebbende beschreven. Dit project heeft maar één belanghebbende, nl: Rick Sloot. Hij is dus de enige die behoeften en harde eisen kan geven die moeten gelden voor het proof of concept. Dit komt omdat het een afgeschermd project is. Andere medewerkers binnen i3D hebben geen direct belang bij dit project.
Andere medewerkers kunnen in een volgend project wel belanghebbend worden. Dit gebeurd als het een project wordt waar een invoering als productiecloud het doel is.

Niet het gehele systeem zal afgedekt kunnen worden met de systeemeisen. Dit komt omdat het gaat om een proof of concept waar sommige delen van het systeem niet onderzocht hoeven worden. De reden hiervoor is de tijd die voor het project beschikbaar is. Ook vind i3D sommige kanten van het proof of concept op dit moment niet belangrijk om duidelijk te krijgen.

Er is met Rick Sloot een semi-gestructureerd interview gehouden. Hier zijn de behoeften aan het proof of concept naar voren gekomen.
Hier onder staan de behoeften gescheiden door categorie waar ze in vallen. Op de plekken waar er gesproken wordt over klanten wordt bedoeld: "de klanten die gebruik maken van de oplossing als deze in productie is genomen". De "klanten" zijn dus in principe denkbeeldige personen.

Kosten

- Het systeem moet een goedkoop alternatief worden voor de Red Cloud van i3D.
- De (theoretische) kosten aan het proof of concept moeten zo laag mogelijk blijven. Alle kosten die extra worden overwogen moeten worden besproken met de opdrachtgever.
- De theoretische winst van de cloud oplossing moet inzichtelijk worden.
- Alle hardware moet uit hergebruik komen. Er staat al vast welke hardware voor dit proof of concept moet worden gebruikt. Dit betekend ook dat er alleen gebruik kan worden gemaakt van HDD's.

Availability

- De availability van het proof of concept is minder belangrijk dan in een traditionele cloudomgeving. De klanten die de uiteindelijke oplossingen gaan gebruiken krijgen goed gecommuniceerd dat ze geen hoge uptime moeten verwachten. Dit zal ook in de SLA staan dan. Een failover/redundantie hoeft alleen te worden aangebracht op de meest kritieke plek in het proof of concept.

Disaster recovery

- De datastorage (backend) zal (een vorm van) redundantie moeten ondersteunen.

Platform

- Het proof of concept moet worden opgezet op basis van OpenStack.

Schaalbaarheid

- De klanten moeten storage kunnen bijbestellen (opschalen).
- De klanten moeten hun resources kunnen schalen (tot maximaal de grootte van de fysieke machine).

Netwerk

- Liever geen gebruik maken van SNAT en DNAT, omdat sommige services niet goed werken en er een overschot aan ports komt.
- De klanten zouden externe IP-adressen moeten kunnen aanvragen voor hun instances.

Documentatie

- Het proof of concept moet naderhand weer hetzelfde opgezet kunnen worden (gerepliceerd).

Performance

- Een instance moet nog werkbaar zijn als de helft van de cores op de compute node in gebruik zijn.
- Het netwerkverkeer van en naar de storage oplossingen moeten niet de verbinding "dichttrekken".
- Het netwerkverkeer over elk netwerk binnen OpenStack mag niet storend worden voor andere gebruikers qua performance.

Security

- Er moet een standaard level aan security ingevoerd zijn. Onbevoegden van buitenaf mogen niet binnen OpenStack komen.

Back-up

- Gebruikerdata en de OpenStack services moeten geback-upt kunnen worden.

Algemeen

- Klanten moeten kunnen ingeloggen via een/het OpenStack control panel.
- Aan het eind moet een functionerende omgeving staan.
- Het resource gebruik van het proof of concept moet onder de limiet van het rack blijven (32 amp)
- Er moet minimaal 1 instance kunnen worden gehost. Deze instance moet van een zelf gekozen distributie zijn.
- Klanten moet op het systeem VM's kunnen draaien en hier snapshots van kunnen maken.
- Het verbruik van klanten moet gemeten kunnen worden.

2. Eisen

In dit hoofdstuk zijn de eisen aan het proof of concept beschreven. De eisen zijn afgeleid van de behoeften van de belanghebbende. De meeste behoeften waren al zo opgesteld dat ze niet omgezet hoefde te worden om er eisen van te maken. Omdat er maar één belanghebbende binnen dit project aanwezig is zullen er ook geen afwegingen tussen conflicterende behoeften gemaakt hoeven worden.

Binnen dit project zijn er minder niet-functionele eisen aanwezig dan wanneer het eindproduct een volledige productie infrastructuur zou zijn. Het eindproduct betreft nu een proof of concept.

De initiële eisen die worden gesteld aan de infrastructuur/het proof of concept zijn allemaal naar voren gekomen in meetings met de opdrachtgever. De haalbaarheid van de eisen is getoetst met de aanwezige literatuur over OpenStack. Binnen de definitiefase zijn de eisen scherpgesteld aan de hand van de OpenStack literatuur en bij het onderzoek naar de toegewezen hardware.

Categorie	Systeemeis
Availability	~~Redundantie moet worden aangebracht op de meest kritieke plek in het proof of concept.~~ (geschrapt)
Back-up	Er moet een back-up van gebruikersdata gemaakt kunnen worden binnen het proof of concept.
Back-up	Er moet een back-up gemaakt kunnen worden van de services/componenten binnen het proof of concept.
Back-up	Binnen het proof of concept moeten snapshots van de VM's gemaakt kunnen worden.
Disaster recovery	In het uiteindelijke proof of concept mag de storage bij schijfuitval (van 1 schijf in het proof of concept) geen dataverlies lijden.
Documentation	Na het project moet de opstelling/het proof of concept opnieuw opgezet kunnen worden.
Network	Het proof of concept moet een alternatief voor SNAT en DNAT hebben om "klanten" publieke IP-adressen op te laten vragen.
Network	Er moet een mogelijkheid zijn om externe IP-adressen te kunnen bestellen voor gebruik om instances.
Open source	Binnen het project moet er gebruik zijn gemaakt van OpenStack.
Overig	Er moet in het proof of concept ingelogd kunnen worden op een control panel.
Overig	In het proof of concept met het verbruik van “klanten” gemeten kunnen worden.
Performance	Het netwerkverkeer van en naar de storage mag niet zo hoog worden dat de verbinding wordt dichtgetrokken. Tijdens normaal (uitgezonderd piekverkeer) gebruik mag het niet hoger worden dan 50% van de lijn.
Performance	Het netwerkverkeer over heel het netwerk mag onder normale omstandigheden (uitgezonderd piekverkeer) niet over 50% van de lijn komen.
Performance	In het proof of concept mag de slowdown op de VM's niet hoger worden dan 200% bij normaal gebruik (uitgezonderd piekverkeer).
Platform compatibility	Er moet op het proof of concept één soort operating system (één distributie) kunnen worden gehost.
Price	De uiteindelijke winst (of verlies) moet inzichtelijk worden.
Price	De uiteindelijke abonnementskosten voor klanten moeten lager zijn dan de primaire cloud oplossing (Red cloud).
Resource constraints	De apparatuur die zich in het rack bevindt moet in de testfase onder de 32 ampére stroomsterkte en 1gbit netwerktraffic blijven.
Scalability	Er moet in het proof of concept de mogelijkheid zijn om hun resources te kunnen schalen.
Scalability	Er moet moet in het proof of concept de mogelijkheid zijn om extra storage te kunnen "bestellen".
Security	In het proof of concept mogen onbevoegde verbindingen vanaf het i3D netwerk niet in een van de interne netwerken komen.
Security	De Linux installaties en de OpenStack componenten op de fysieke nodes moeten geüpdatet kunnen worden.

Bijlage III: Bedrijfsoriëntatie

Bedrijfsoriëntatie

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Stagebedrijf: i3D
Plaats: Rotterdam
Datum: 26-01-2016

Versiebeheer

Versie	Datum	Wijziging
0.1	26-01-2016	Conceptversie bedrijfsoriëntatie
1.0	01-02-2016	Definitieve versie

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

Inleiding

In dit document wordt de uitgevoerde bedrijfsoriëntatie beschreven. Met het uitvoeren van deze oriëntatie is een breder beeld gevormt over de organisatie i3D.
Het uiteindelijke document heeft een brede beschrijving van i3D. Onderwerpen die in de oriëntatie naar voren komen zijn:

- De omgeving van de organisatie
- De doelstellingen van de organisatie
- De werkprocessen binnen i3D
- De technologie binnen i3D
- De organisatiecultuur van i3D
- De organisatiestructuur van i3D
- Mijn rol als IT'er / mijn plaats in de organisatie

Inhoudsopgave

1. Omgeving

De omgeving van een bedrijf is zeer belangrijk. De onderneming zal de omgeving goed in de gaten moeten houden om ongewenste bewegingen uit te verminderen en positieve bewegingen te omarmen.

1.1 Markt

Het bedrijf i3D is actief op de hosting markt. In de hosting markt biedt i3D game de volgende diensten aan:

- Gamehosting met ondersteuning
- Webhosting
- Cloud services (IaaS)
- Colocatie
- Dedicated servers

De klantgroepen die te onderscheiden zijn in de markt zijn:

- Klanten voor een game server (gamers).
- Uitgevers van games die hun game willen hosten bij i3D.
- Bedrijven: voor colocatie, dedicated servers, cloud services en webhosting.
- Particulieren: voor webhosting en dedicated servers.

De bovenstaande opsommingen geven al aan dat i3D alleen actief is in de hosting markt. Echter wel op verschillende niveau's van de hosting markt: van webhosting tot colocatie en van cloud services tot gamehosting. I3D is erg geïntegreerd wat betreft de diversiteit aan markten waar i3D zich op bevindt. Als er wordt gekeken naar de diversiteit aan aanbod binnen de hosting markt is i3D vrij divers.

In de hosting markt is cloud hosting nog vrij actueel. Ineens wil ieder bedrijf over naar "de cloud". I3D heeft zelf al een cloud service draaien. Andere actuele onderwerpen zijn er niet. Binnen de gamehosting zullen de servertechnieken actueel moeten blijven. Alleen zo kunnen de nieuwste games ook worden gehost.

De markt van i3D is in conclusie vrij geïntegreerd. Dit komt voor het grootste deel doordat i3D vrijwel alleen op de hosting markt aanwezig is.

1.2 Stabiliteit

De stabiliteit van de bedrijfsomgeving is gemiddeld te noemen. Het eerste punt waardoor de stabiliteit verminderd is verandering door politieke invloeden. Aangezien de organisatie bezig is met cloud services en hosting hebben invloeden van de politiek greep op de stabiliteit. Er worden nog wel eens nieuwe wetten gemaakt om vooral privacy en security van (hosting)klanten meer te garanderen.

Nieuwe concurrenten zijn in de hosting scene wel voorkomend. Vooral kleine aanbieders van webhosting komen er nog wel bij als concurrenten. Op dit moment is de cloud nog een vrij actueel onderwerp. Er komen ook nog steeds nieuwe bedrijven bij die cloud services aanbieden.

Binnen de IT is het vrij snel zo dat er dynamisch ingespeeld moet worden op nieuwe technologiën. De snelheid bij het verouderen van technieken ligt vrij hoog. Echter is het niet zo dat er een specifiek onderdeel binnen i3D is waar de technologische vernieuwing hoger ligt dan in de rest van i3D.

De behoefte van afnemers is binnen i3D wel een groot gedeelte waar flexibel ingespongen moet worden. Aangezien het om hosting services gaat zijn er veel bedrijven die vaak andere eisen stellen aan hun hosting dienst. Colocatie, cloud services en webhosting zijn nog relatief stabiel.

Game hosting is, zeker met het brede aanbod, erg dynamisch. Er komen vrij vaak nieuwe games uit die gehost (kunnen) worden door i3D.

1.3 Competitie

Het eerste onderwerp waar de competitie door kan worden bepaald is de concurrentie. Concurrentie is in de hosting scene vaak voorkomend. Er zijn vrij veel hosting bedrijven actief vandaag de dag en er komen er steeds bij. Vooral het hosten van websites wordt vaak gedaan. Als er wordt gepraat over colocation concurrenten dan wordt het lijstje al een stuk korter. Dit komt doordat er een datacenter nodig is voor colocatie. Niet veel bedrijven kunnen dat bieden. Uiteraard kunnen bedrijven wel een contract sluiten met een datacenter om hun klanten onder te brengen in het datacenter.
Het zelfde geldt voor dedicated servers: daar is ook veel apparatuur voor nodig. Echter is het makkelijker voor concurrenten om dedicated servers aan te bieden dan een gehele colocatie. Voor de cloud oplossingen geldt ook hetzelfde.
Als het om game hosting gaat dan heeft i3D wel concurrenten, echter worden deze een stuk minder concurrerend als er bedacht wordt dat i3D de allernieuwste games mag en kan hosten. Dit wordt dan aangevraagd door de ontwikkelaars/uitgevers van de games. Ook host i3D een heel breed spectrum aan games. I3D is één van de enige in Nederland die hosting in zo'n vorm kan aanbieden.

Binnen de omgeving van i3D zijn weinig mensen aangesloten bij een vakbond. Ook is het zo dat het in de IT niet tot veel onrust komt (wat tot tussenkomst van een vakbond leidt) zoals in de zorg.
De overheid heeft weinig invloed op de omgeving van i3D. Uiteraard zijn er altijd de standaard wetgeving waar aan moet worden voldaan: hierbij kan worden gedacht aan privacy en beveiliging.

Wat betreft milieugroeperingen heeft i3D weinig competitie. Wel kan worden bedacht dat in de toekomst de omgeving wat vijandiger wordt door milieugroeperingen. Door de hoge energieafname van het datacenter kunnen milieuvoorstanders in opstand komen.

I3D heeft met het datacenter een hoge behoefte aan middelen. De beschikbaarheid van die middelen is op dit moment goed te noemen. In de toekomst kunnen middelen minder goed verkrijgbaar worden door schaarste, bijvoorbeeld door een milieuramp. Door de hoge behoefte aan middelen wordt de omgeving van i3D wel vijandelijker.

In conclusie is de omgeving van i3D middel competatief, dit komt vooral door de behoefte aan middelen en de aanwezig concurrentie.

2. Organisatiedoelen

Door doelen te stellen kan een bedrijf bestaan. Een organisatie is een doelgericht samenwerkingsverband. In het bedrijf werken allerlei personen die ook weer op zichzelf verschillende doelen nastreven. Zo kunnen medewerkers doelstellingen hebben die een positief effect hebben op de organisatie en er zijn medewerkers die doelstellingen hebben die een negatief effect hebben op de organisatie.

Om in de maatschappelijke behoefte kwalitatieve hosting te voorzien is i3D opgericht. De missie die i3D heeft geformuleerd is als volgt:

"i3D.net leverd volledig ontwikkelde hosting oplossingen aan bedrijven over heel de wereld. Dit doet i3D als een managed hosting provider met een wereldwijde netwerkdekking. Om langdurende- en goede relaties op te bouwen en te onderhouden met klanten brengt i3D de meest innovatieve en marktleidende hostingtechnologiën naar de markt van vandaag. Door aanhoudende focus op kwaliteit kunnen we de beste resultaten/prestaties aanbieden. Als een innovatief bedrijf met veel kennis aan boord kunnen we onze klanten ook de beste resultaten laten behalen."

De grootste bedrijfsdoelstelling binnen i3D is: “grote uitbreiding op hosting vlak op internationaal gebied, maar ook in Nederland.”
Verdere doelstelling zijn nog niet actief geformuleerd of goedgekeurd.

3. De werkprocessen

De primaire processen binnen i3D houden in:

- Game helpdesk
- Game uitgevers ondersteunen
- Game servers onderhouden/opzetten
- Technische helpdesk
- Technisch onderhoud en servers opzetten

De afdeling waar ik nu werkzaam ben is in het network team. De belangrijkste taken van het network team zijn technisch onderhoud, servers opzetten en technisch onderhoud.

Het werkproces "technische helpdesk" omgeeft de werkzaamheden communiceren met klanten die problemen ondervinden met hun host en de problemen oplossen die binnen zijn gekomen en in een ticket staan.
Het werkproces "technische helpdesk" is in de communicatie met de klanten zeer voorspelbaar. Bij de afhandeling van een call wordt in principe elke keer volgens een vast patroon gewerkt. Echter zijn de problemen die de klanten aangeven een stuk minder voorspelbaar en kan hier ook niet gemakkelijk een vaste procedure voor worden aangehouden.
De moeilijkheidsgraad van de communicatie naar klanten is zeer laag. Wel is het zo dat er in het Engels moet kunnen worden gecommuniceerd. Voor de rest hoeft er weinig mentale inspanning te worden geleverd om deze werkzaamheden tot een goed te brengen. Als de tickets zijn gemaakt en de problemen moeten worden opgelost is er echter meer kennis van zaken nodig. Vaak zijn de problemen vrij complex en weinig voorkomend. Om deze problemen op te lossen is het dus wel een vereiste dat de medewerkers zijn opgeleid om dit te kunnen uitvoeren, ze hebben een hoog kennisniveau van het betreffende probleemgebied nodig. Denk hierbij aan onderwerpen als: complexe netwerkproblemen en virtualisatieproblemen.

Het werkproces "servers opzetten en technisch onderhoud" omgeeft de werkzaamheden om nieuwe servers klaar te maken voor een klant en servers te onderhouden voor een klant. Binnen dit werkproces kan er gedacht worden aan werkzaamheden zoals: RAM bijprikken, HDD vervangen, nieuwe servers ophangen in een rack. Voorkomende problemen oplossen die opkomen bij opzetten en onderhoud.

Het werkproces "servers opzetten en technisch onderhoud" kan in het begin onvoorspelbaar zijn. Uiteraard kunnen medewerkers worden getraind om de meeste problemen die zich voordoen tijdens het onderhoud van de apparatuur. Echter blijven er altijd problemen bestaan die niet van tevoren te zien waren. Naast die onvoorzienbare problemen is dit werkproces vrij voorspelbaar, zeker bij herhaaldelijk onderhoud van dezelfde soorten apparatuur. Voor de moeilijkheidsgraad geldt eigenlijk hetzelfde plaatje. Als er tijdens het onderhoud een onbekende fout opspringt zal deze wel verholpen moeten worden. Hier is meer mentale inspanning voor nodig, maar gebeurd niet vaak. Bij technisch onderhoud is het vaak zo dat er bij aanvang al volledig duidelijk is wat er moet gebeuren en hoe het moet gebeuren: hier is dus weinig mentale inspanning nodig.

4. De technologie

Technologie, dat is in een IT-bedrijf niet weg te denken. Zonder technologie bestaat er namelijk geen IT. Technologie helpt bij of zorgt voor de transformatie van grondstof/beginsituatie naar diensten of goederen. Binnen i3D zijn het voornamelijk diensten. De technologie (computersystemen, infrastructuren, servers, etc) helpt of is cruciaal bij het aanbieden van diensten aan klanten. Hier kan de aanvraag of behoefte van de klant worden gezien als "grondstof" of beginsituatie.
De technologie binnen i3D is vrij hightech te noemen. Zo staan er in het datacenter bijvoorbeeld directe uplinks

De cloud is een vrij actuele technologische ontwikkeling die relevant is voor i3D. Aan de ene kant is het een bedreiging voor i3D. Klanten kunnen de cloud zien als een betere vorm van hosting en daarom overstappen naar een bedrijf die wel (of betere) cloud diensten aanbiedt.
Aan de andere kant is de cloud een kans die i3D heeft aangegrepen. Om meer klanten te trekken en huidige klanten te behouden is er een IaaS cloud ontwikkeld op basis van Microsoft Azure met de naam "Red cloud".

Een andere actuele technologische ontwikkeling is Software Defined Networking of SDN. Het is een manier om fysieke devices in een infrastructuur te kunnen uitvoeren als virtueel apparaat. De invoering van SDN zal een hogere complexiteit van het netwerk betekenen en hoge kosten bij invoering.

Aspecten van de technologie		Kennisniveau	
		Laag	**Hoog**
Technologie	**Complex**		**X**
	Eenvoudig		

Afbeelding 1: In dit kwadrant is i3D gepositioneerd.

5. De organisatiecultuur

I3D is een relatief klein bedrijf als er wordt gekeken naar het aantal medewerkers. De lijnen kunnen daarom ook vrij kort zijn. Dat is echter niet altijd het geval.
De personen die snel carrière maken binnen i3D zijn vooral de mensen met veel ervaring in de richting waarin ze werken. Ook moeten deze personen zichzelf goed kunnen uitdrukken. Bij een hogere functie komen namelijk veel besprekingen kijken. De rolmodellen binnen i3D zijn Stijn Koster en Rick Sloot. Beide personen zitten in het hoge management. Stijn Koster is de oprichter en baas van i3D. Rick Sloot is de COO binnen i3D.

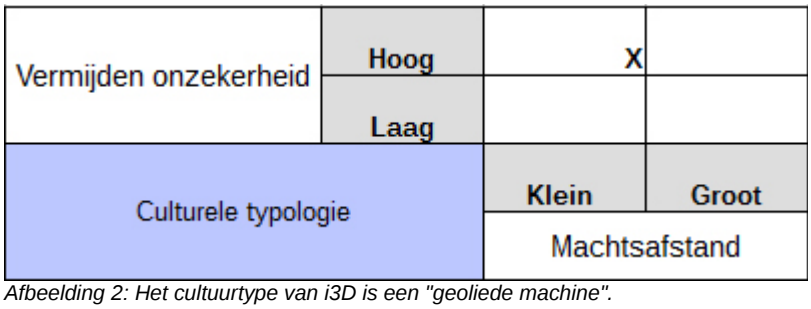
Binnen i3D zijn er een aantal status-symbolen die aangeven waar i3D voor staat. Het eerste status-symbool is de kleding. Er wordt geacht dat iedereen die bij i3D werkt ook de betreffende kledingstukken aantrekt. Zo bestaan er overhemden, truien en jassen. Op de kledingstukken staat het logo van i3D afgebeeld. In het logo van i3D staat ook "performance hosting" hier kan dus wel uit opgemaakt worden waar i3D voor staat.
Het gebouw waar i3D in is gevestigd is de oude Van Nelle fabriek in Rotterdam. Dit is al een status symbool op zich. I3D zit in dit gebouw gevestigd omdat er grote ruimtes aanwezig zijn (voor het datacenter) en er zijn vrij directe data-en electriciteitverbindingen aan te leggen vanwege de locatie t.o.v. de havens.
De inrichting bij de netwerk- en gaming afdeling is gericht op de games die gehost worden of gehost zijn door i3D.

Een ander gedragspatroon is de openheid en informaliteit van de medewerkers onderling. Uiteraard merk ik dit alleen op bij de afdelingen "gameondersteuning" en "netwerk". Dat komt omdat ik heel de dag rond deze afdelingen aanwezig ben. Echter komen er ook mensen van andere afdelingen die onderling ook informeel en open communiceren.
Vergaderen is vrijwel altijd een formele aangelegenheid, zo ook bij i3D. Het vergaderen wordt eigenlijk alleen gedaan door de personen die wat hoger in de “piramide” van het bedrijf staan. Vaak zijn dit vergaderingen tussen de CEO, COO, CTO, een belangrijk lid van de betreffende afdeling en klanten. De klanten zijn nog wel eens uitgevers van games en belangrijk voor i3D.

De normen en waarden van de medewerkers in i3D is vrij goed te onderscheiden. Een belangrijke norm is vriendelijkheid en een professionele werkhouding tegenover klanten. Alle klanten moeten gelijk worden behandeld. In de dagelijkse gang van zaken is dat goed te merken, zeker als er klanten aan de telefoon komen of in de livechat verschijnen.

Een belangrijk norm, ook voor stagiaires, is het blijven als er iets af moet. Het gaat dan om werkzaamheden die de primaire processen van het bedrijf aangaan en waarbij makkelijk geholpen kan worden. De opdrachtgever die heeft wel gezegd dat hij stagiaires niet kan dwingen om te blijven, het is echter wel leerzaam.
Het is dan ook vrij logisch dat collega's boos worden als personen weg gaan als nog werk ligt, zeker vlak voor de oplevering van een grote gamehosting.

Om de organisatiecultuur binnen i3D te kunnen beschrijven is het schema over de cultuurtypen van Hofstede gebruikt.



Afbeelding 2: Het cultuurtype van i3D is een "geoliede machine".

De cultuur van de organisatie binnen i3D is procesgericht te noemen. Op het gebied van onderhouden en

inrichten van de servers worden er zo min mogelijk risico's genomen. Dat past ook wel bij een groot datacenter: daar mijdt men risico's, bijvoorbeeld vanwege de afspraken in de SLA's. De toegewezen werkzaamheden, meestal veel, moeten af, dat is het belangrijkste. Er kan wel eens verschil in werk zitten door een raar probleem die binnenkomt bij de helpdesk. Over het algemeen is het veel van hetzelfde.

Het bovenstaande geeft al gedeeltelijk aan dat i3D taakgericht is. Er heerst een relatief sterke druk om de werkzaamheden af te krijgen. Anders zou er geklaagd kunnen gaan worden van hogerhand of vanuit een klant. Het welzijn van medewerkers is gedeeltelijk belangrijk, echter het op tijd afhebben van kwalitatief werk is belangrijker. Dat is gedeeltelijk te wijten aan het grote gedeelte mannen (afwezigheid van vrouwen) bij i3D. De individuen die hoog in de “piramide” staan bij i3D nemen belangrijke beslissingen. Er wordt weinig in een groep beslist.

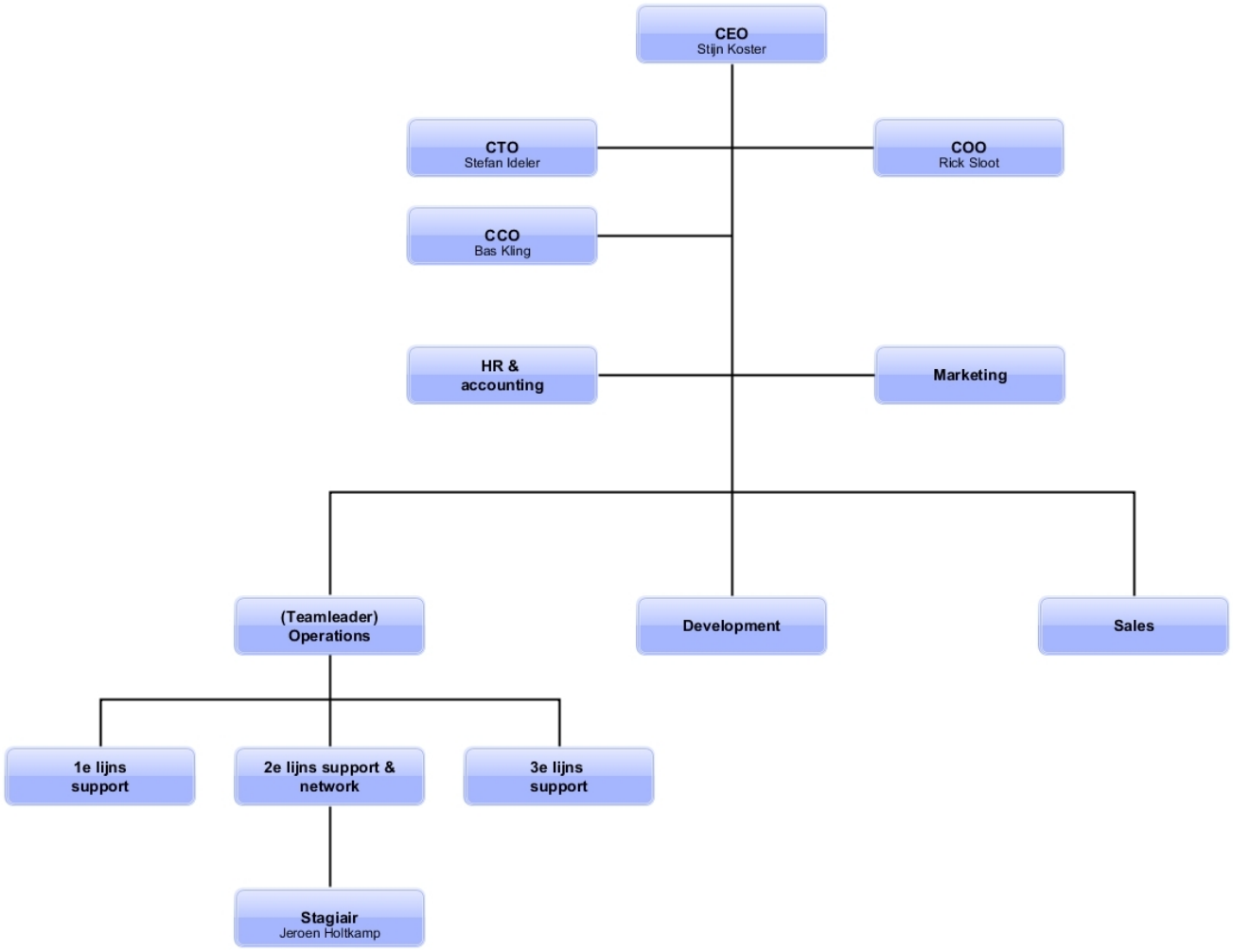
De medewerkers binnen i3D zijn deels organisatie-gebonden en deels professioneel-gebonden. Ze identificeren zich wel met de organisatie i3D. Echter worden mensen niet alleen op basis van schoolachtergrond opgenomen. Ook is de geschiktheid voor het werk een belangrijk onderdeel. Zo werken er ook schoolverlaters bij i3D. Zij hebben weinig schoolachtergrond (diploma) maar ze zijn absoluut geschikt voor het werk. Er wordt veel gesproken over de privé-levens van iedereen. Het zou daarom ook gedeeltelijk zo kunnen zijn dat normen en waarden die binnen i3D gelden tevens bij de medewerkers thuis gelden.

De openheid bij i3D is vrij groot desondanks de zwijgplicht die er geldt. Er worden in het bedrijf nog al eens bedrijfsgeheimen besproken. Deze geheimen kunnen ook voortkomen uit geheim van een klant. Nieuwkomers krijgen ook deze informatie te horen. Wel wordt er duidelijk gemaakt dat het om geheime informatie gaat. Tevens is iedere medewerker open over zijn of haar achtergrond en leefsituatie. Nieuwe medewerkers zullen zich hier snel thuis voelen. Dat komt gedeeltelijk ook door de informele instelling.

De controle binnen i3D is vrij gebalanceerd. Aan de ene kant is de afdeling waar ik zit. De dagelijkse gang van zaken is vrij informeel. Ook als er gekeken wordt naar postbestellingen en onderdelen is men niet kostenbewust en heerst er geen strakke controle. Een voorbeeld hiervan is een binnengekomen pakket dat door “iemand” is besteld en wat bijna kwijtraakt, opeens vindt iemand het pakket weer. Aan de andere kant is daar het management van i3D. Zij houden zich strak aan vergadertijden en praten vrij serieus over het werk. Toch denk ik dat de strakke controle de overhand zou moeten hebben aangezien i3D een leverancier van relatief risicovolle diensten is.

Binnen i3D is de oriëntatie gericht op de klanten en wat minder op de procedures. Dat komt omdat klanttevredenheid belangrijker is dat strikte regels. Echter zit hier ook een grens aan aangezien de procedures belangrijk zijn. Met deze procedures kan een wat strakkere bedrijfsvoering gegarandeerd worden. Er zit dus een grens aan de klantgerichtheid.

6. De oganisatiestructuur



Afbeelding 3: Het organogram van i3D.

In afbeelding 3 is een overzicht te vinden van de afdelingen binnen i3D. Zelf ben ik ingedeeld bij de afdeling "network", al zit ik fysiek op de afdeling "1e en 2e lijns support".
Er is binnen i3D een hiërarchie te onderscheiden. Deze is wat platter dan bij grotere ondernemingen. Ook de bestuursleden zijn zeker niet onbereikbaar qua directe communicatie, ze zijn echter wel vaak onbereikbaar door drukte.
Binnen de organisatie heerst een informele sfeer, dit is bijvoorbeeld te merken aan opmerkingen die gemaakt worden tegen elkaar en iedereen spreekt elkaar aan bij zijn voornaam. Naar de buitenwereld toe zal de piramide vorm van i3D meer tot uiting komen. Dit kan bijvoorbeeld worden gezien aan het feit dat de bestuursleden in pak lopen. Lager in de hiërarchie hebben de medewerkers kleding aan met i3D logo.
In de organisatie i3D is een platte lijnorganisatie te herkennen. Iedere afdeling heeft een afdelingshoofd. De afdelingshoofden sturen de medewerkers in hun afdeling aan en luisteren op hun beurt weer naar het management.
De afwezigheid van veel lagen in de hiërarchie heeft een groot voordeel: veel minder bureaucratie. Alle beslissingen die het management neemt hoeven maar één stap te nemen voordat ze bij de medewerkers komen. Sommige beslissingen zullen zelfs van het management naar de medewerkers doorgegeven worden.
Een belangrijke reden waarom er voor deze organisatiestructuur is gekozen is de duidelijkheid. In deze

organisatievorm is het zeer duidelijk wie voor wat verantwoordelijk is. De bevoegdheden zijn ook duidelijk.
Een duidelijk nadeel van deze structuur is de druk op de managers: zij hebben veel verantwoordelijkheid en veel taken.

7. Mijn rol als TI'er binnen het bedrijf

De werkzaamheden die ik uitvoer bij i3D dragen bij aan de bedrijfsprocessen. In het kort houden de werkzaamheden in dat er een proof of concept opgezet moet worden. Met dit proof of concept moet worden aangetoont dat er een low-cost cloud oplossing opgezet kan worden met behulp van OpenStack en Ceph.
Mijn werkzaamheden hebben vooral te maken met de bedrijfsprocessen omtrent technische onderhoud en servers opzetten. Er is echter geen direct verband aangezien het gaat om een proof of concept. Er wordt niet een product opgezet dat ook direct als product verkocht kan worden. Het is meer een afgezonderd proces wat indirect, in de toekomst, te maken zou kunnen krijgen met deze bedrijfsprocessen.
Wel kan er door de conclusie van de opdracht een nieuwe klantenkring worden aangetrokken. Er zijn genoeg mensen die niet veel geld overhebben voor hosting en toch een hostingdienst willen afnemen.

De werkzaamheden die uitgevoerd worden door heel de opdracht hebben vooral te maken met:

- Networking
- Testen
- Cloud computing
- Onderzoek uitvoeren
- Een project juist doorlopen

Zo zijn de beroepstaken “ontwerpen van een infrastructuur” en “testen van een infrastructuur” aan te tonen met dit project. Binnen het project moet er toch een netwerk/infrastructuur worden opgezet om het proof of concept tot uiting te laten komen. Voordat het netwerk kan worden opgezet zal het dus eerst moeten worden ontworpen. Alvorens het netwerk te ontwerpen moeten eerst alle ontwerp parameters en eisen duidelijk zijn.
Tegen het einde van het project moet het proof of concept nog worden getest. Met het juist testen kan de beroepstaak “testen van een infrastructuur” worden aangetoont.
Ook zijn de vaardigheden “onderzoeken” en “project doorlopen” vrij cruciaal in dit project. Voordat er ook maar iets kan worden gemaakt of ontworpen moeten de betreffende onderwerpen/vragen worden onderzocht. Dit om de juiste kennis te krijgen om het project op een juiste manier te kunnen doorlopen.
De kennis over netwerktechnologie uit de opleiding komt goed van pas in dit project. Met het ontwerpen en opzetten van een netwerk zal er toch kennis van netwerken nodig zijn.

Bijlage IV: Interviewverslagen

Interview 1

Geïnterviewde:	Rick Sloot als RS
Functie binnen i3D:	CEO
Functie binnen project:	Opdrachtgever
Locatie:	Van Nelleweg 1, Rotterdam
Datum:	15-01-2016
Tijdstip:	13.00
Interviewer:	Jeroen Holtkamp als JH
Soort interview:	Open
Onderwerp:	Probleemdomein in beeld krijgen

JH: Goedemiddag Rick.

RS: Goedemiddag Jeroen.

JH: Zo, in het eerste interview wilde ik het graag over het probleemdomein van de opdracht hebben. Om te kunnen bepalen hoe het project precies opgezet gaat worden en wat ik in de volgende fasen ga doen zal ik eerst moeten weten wat de problemen precies zijn.

RS: Okee, duidelijk.

JH: Het interview is open. Dit betekent dat ik een aantal onderwerpen heb opgesteld. Binnen deze onderwerpen kunnen we vrij communiceren en kan ik doorvragen.

RS: Kom maar op dan.

JH: Het eerste onderwerp is het initiële probleem. Wat heeft er voor gezorgd dat deze opdracht tot stand kwam, de aanleiding zogezegd?

RS: Eigenlijk hebben we meerdere zogenaamde “problemen”. De eerste is het ontbreken van een goedkoop alternatief voor onze bestaande Red Cloud omgeving binnen i3D.

JH: Wat is de reden dat er een goedkoper alternatief zou moeten komen?

RS: Nou, sommige klanten vragen aan ons of er al een goedkoper alternatief bestaat voor de Red cloud. Meestal is het zo dat ze niet zulke hoge eisen stellen aan de oplossing, maar wel de prijs betalen voor deze kwaliteit.

JH: Dat is duidelijk, wat is het volgende probleem?

RS: Het volgende probleem is het overschot aan oude servers. We draaien veel servers in het datacenter. Na een bepaalde tijd zijn de servers niet goed meer in te zetten voor de primaire taken binnen i3D. Deze servers zijn dan “over”. De servers hebben dan geen oplossing om commercieel ingezet te worden. Om het nog beter te zeggen: “Er is geen technologie in het bedrijf aanwezig om de servers in grote volumes weg te zetten”.

JH: Okee, en er waren nog meer aanleidingen voor het project?

RS: Nee, deze twee dat zijn de aanleidingen voor het project.

JH: Ik heb wel al gehoord dat er al onderzoek is gedaan?

RS: Ja, dat klopt. Vorig jaar is er onderzoek gedaan om te bekijken welk cloudplatform wij het beste zouden kunnen invoeren binnen i3D. Hier kwam toen OpenStack uit. Dat is ook de reden dat daar in ieder geval geen onderzoek meer naar gedaan hoeft te worden. Het platform staat dus vast: dat is OpenStack.

JH: En de opdracht is ontstaan door deze twee aanleidingen bij elkaar op te tellen?

RS: Ja, dat is waar. I3D heeft behoefte aan een low-cost cloud oplossing voor minder eisende klanten plus er zijn servers over. Dat is natuurlijk een mooie gelegenheid om te kijken of dat een oplossing is om de oude servers commercieel in te zetten.

JH: En zal er dan ook gekeken moeten worden naar andere onderdelen naast OpenStack?

RS: Wij hebben even gekeken naar OpenStack en het blijkt dat voor storage relatief vaak gebruik wordt gemaakt van een project met de naam Ceph. Wij hebben graag dat hier ook naar gekeken wordt. Misschien is dat ook wel wat om te implementeren in de opstelling.

JH: Okee. Zijn dat alle aanleidingen?

RS: Ja, dat zijn ze allemaal.

JH: Ik denk dat ik al een aantal problemen die hier uit voorkomen kan raden. Toch ga ik het vragen. Wat zijn de problemen hierbij?

RS: Ten eerste heb je het probleem dat er nog weinig kennis is van OpenStack binnen i3D.

JH: En om een low-cost cloud oplossing op te zetten op basis van OpenStack heb je wel kennis van OpenStack nodig.

RS: Ja, dat lijkt me wel handig.

JH: Hetzelfde zal dan ook gelden voor Ceph?

RS: Ja, dat klopt, hier is ook nog geen onderzoek naar gedaan.

JH: Okee, en wat is het volgende probleem?

RS: Er is natuurlijk het probleem nog dat we niet weten of we wel winst kunnen maken met deze oplossing. Dat is wel belangrijk als je iets commercieel in gaat zetten

JH: Ja, precies. Het zal dan wel een theoretische berekening worden want het eindproduct is een proof of concept.

RS: Dat klopt. De winst kan niet worden “in het echt” worden gemeten zonder dat het in productie wordt genomen.

JH: Ook kan ik me voorstellen dat het een probleem is dat i3D niet weet of de oude servers zo'n oplossing wel aan kan.

RS: Nee, precies, maar dat is juist een punt om uit te zoeken met het project.

JH: Ja, inderdaad, dat snap ik. Is het al wel duidelijk welke apparatuur er beschikbaar is voor de opdracht?

RS: Half, maar dit zou je even moeten nakijken. De specificaties van de apparatuur zou je dan ook moeten uitzoeken. Die weten wij natuurlijk ook niet uit ons hoofd.

JH: haha, nee, dat snap ik. En het interne netwerk zou ik dan ook wel moeten onderzoeken om te zien in welke omgeving de opstelling komt te staan?

RS: Ja, maar over het netwerk dat kun je overleggen met netwerk.

JH: Okee, en zijn er nog meer problemen?

RS: Nee, ik geloof dat dat het wel is.

JH: Wat gebeurd er eigenlijk met de servers als er geen winst gemaakt kan worden met de oplossing?

RS: We kunnen de servers eventueel nog verkopen. Hier krijgen we weinig meer voor. Ook kunnen we een andere toepassing gaan bedenken voor de servers. Maar er is wel steeds meer ruimte nodig in het datacenter voor nieuwere servers.

JH: Okee. En wat zijn de gewenste veranderingen die deze problemen moeten oplossen voor i3D?

RS: We willen dat er een opstelling staat aan het eind waarmee we wat beter kunnen zien of OpenStack kan draaien op de oude apparatuur en hoe OpenStack werkt. Deze opstelling zal dus niet alle functionaliteiten hoeven te bevatten die we uiteindelijk in de low-cost cloud oplossing verlangen.

JH: Een proof of concept dus?

RS: Ja, dat klopt.

JH: Op een later tijdstip kunnen we dan ook de grenzen stellen die voldoen bij het proof of concept?

RS: Yes.

JH: Om te verifiëren wat de mogelijke winst van de oplossing zou zijn moet er ook een berekening van de theoretische winst worden gemaakt?

RS: Ja, dat is waar.

JH: Aan gezien het een proof of concept wordt denk ik dat het ondersteunende documentatie nodig heeft zodat het naderhand weer net zo opgebouwd kan worden?

RS: Op een later tijdstip moeten wij het proof of concept zelf ook weer op kunnen bouwen net zoals jij dat in dit project hebt gedaan.

JH: En het proof of concept moet worden opgezet op de toegewezen oude hardware?

RS: Ja, zo kun je het wel samenvatten.

JH: Okee, en zijn er nog verder gewenste veranderingen?

RS: Nee, ik geloof dat dit het wel is.

JH: Even nog de vraag: Wie zijn de belanghebbenden van het project?

RS: Eigenlijk is heel i3D belanghebbend. Alleen zijn de andere medewerkers binnen i3D afgeschermd van het project. Ik ben de contactpersoon van heel i3D.

JH: Ehm...Dan geloof ik dat we wel zo'n beetje alles hebben besproken. Dan wil ik je graag bedanken voor de vrijgemaakte tijd.

Interview 2

Geïnterviewde:	Rick Sloot als RS
Functie binnen i3D:	CEO
Functie binnen project:	Opdrachtgever
Locatie:	Van Nelleweg 1, Rotterdam
Datum:	18-01-2016
Tijdstip:	13.00
Interviewer:	Jeroen Holtkamp als JH
Soort interview:	semigestructureerd
Onderwerp:	algemeen beeld krijgen en projectgrenzen
Optioneel commentaar:	minder tijd voor het gesprek, het is geen vooraf ingeplande meeting

JH: Goedemiddag.

RS: Goedemiddag Jeroen. Jij had een aantal vragen voor mij?

JH: Ja dat klopt. Om een beter beeld te kunnen vormen en een beetje op de goede weg te komen heb ik een aantal vragen..

RS: Okee, kom maar op dan.

JH: Wat zijn de projectgrenzen?

RS: Ten eerste is het belangrijk dat de functionaliteit van het systeem basaal wordt gehouden. Dit betekent dat er bijvoorbeeld maar één image op hoeft te draaien. Het systeem aan het eind hoeft niet uitgebreidt te zijn, maar hij moet wel stabiel draaien.

JH: Okee, zijn er nog meer projectgrenzen?

RS: Uiteraard. Zo denk ik dat het belangrijk is dat je niet te veel de diepte in gaat met routing protocollen. Natuurlijk moeten deze worden beschreven, maar er is zo veel informatie op te doen in de richting van routing dat je je er niet té veel in moet verdiepen. Je zult dan waarschijnlijk niet genoeg tijd hebben voor de hoofdopdracht.

JH: Nee, dat lijkt me inderdaad niet zo slim. Hoe zit het eigenlijk met onderdelen in het netwerk die ik niet standaard toegewezen krijg?

RS: Ook is het zo dat je in je netwerk misschien onderdelen nodig zult hebben die we je niet hebben toegewezen aan het begin van de opdracht. In dit geval moet je altijd in stock producten nemen. Het liefst zelfs gebruikte producten.

JH: Het “huismerk” zogezegd?

RS: Haha, ja precies! Op die manier heb je ook niet de tijdverspilling van onderdelen selecteren.

JH: Zitten er nog grenzen aan het rack waar de onderdelen in geplaatst worden?

RS: Nou, de grenzen van het rack zijn 32 ampére stroomsterkte en 1 Gbit aan netwerksnelheid. In principe is dit voor het project niet heel belangrijk, want daar kom je niet aan.

JH: Okee, dat is duidelijk. In het vorige interview hadden we het over onderzoek naar Ceph. Is het zo dat Ceph alleen onderzocht moet worden?

RS: Wat bedoel je precies?

JH: Nou, ik gok dat naar Ceph nog meer alternatieven bestaan. Nu is het zo dat Ceph onderzocht en eventueel geïmplementeerd moet worden. Het is echter wel een harde eis dat Ceph onderzocht moet worden.

RS: Ja, dat klopt. Wij hebben bij het kijken naar OpenStack ook gezien dat er een storage oplossing genaamd Ceph bestaat. Graag willen we dat je alleen hier op richt en op de componenten binnen OpenStack. Alternatieven van Ceph hoeven we niet te weten binnen dit project. We wilen alleen weten of Ceph iets is voor ons.

JH: Okee, zijn er nog meer projectgrenzen?

RS: Ik geloof dat dat ze wel waren. Wellicht schiet mij er later nog eentje te binnen. Ik moet wel bijna weer weg.

JH: Okee, even snel nog. Is het al duidelijk welke hardware ik toegewezen krijg?

RS: Ehm...ik dacht de DL120. In het rack passen er wel een stuk of veertig. Je moet zelf even kijken, ik denk niet dat je ze allemaal nodig gaat hebben in je opstelling. Je mag er alles mee doen wat je wilt. We moeten wel even kijken welk rack we je toe kunnen wijzen.

JH: Dat is duidelijk. Even een vraag over de meetings. Die kunnen we elke week of om de week plannen?

RS: Dat is waar. Je moet wel even van tevoren doorgeven dat je een meeting wil hebben. Het liefst met een agenda uitnodiging. Heel vaak is het ook zo dat je andere collega's hier ook veel over weten. Met hen kun je dus ook overleggen.

JH: Okee. Dan nog snel een vraag over de berekening van de winst. Aan het eind moet er een gehele winstberekening liggen?

RS: Dat is waar, we willen weten of er winst gemaakt kan worden met deze oplossing.

JH: Is er een standaard voor winstberekening voor zoiets?

RS: Je zou het even aan de personen kunnen vragen die met die met de financiën bezig zijn. Op zich denk ik ook wel dat je er uitkomt als je er logisch over nadenkt.

JH: Het lijkt me ook niet heel lastig, maar ik vraag me af of er een standaard voor was. Eerst variabele bedenken die van toepassing zijn bij de berekening. Dan de waardes invullen en berekenen?

RS: Dat lijkt me inderdaad een goede methode. Je kunt in het begin van je project ook al een basis berekening maken met variabelen die je al kan verzinnen en zo opzoeken.

JH: Dan in een latere fase de berekening afmaken als ik de andere variabelen kan invullen.

RS: Precies! Maar ik heb weer andere verplichtingen. Voor de volgende meeting moet je dus even een invite versturen.

JH: Okee.

Interview 3

Geïnterviewde:	Rick Sloot als “RS”
Functie binnen i3D:	CEO
Functie binnen project:	Opdrachtgever
Locatie:	Van Nelleweg 1, Rotterdam
Datum:	22-01-2016
Tijdstip:	13.00
Interviewer:	Jeroen Holtkamp als “JH”
Soort interview:	semigestructureerd
Onderwerp:	Systeemeisen

JH: Hallo Rick.

RS: Goedemiddag Jeroen.

JH: Nou, voor vandaag had ik in gedachte de systeemeisen helder te krijgen. We hadden het er tussendoor al even over gehad.

RS: Ja, dat klopt. Hoe gaan we het bespreken?

JH: Ik wil graag de systeemeisen bespreken. Ik denk dat het verstandig is om dit te doen aan de hand van de niet-functionele eisen. Dan komen we vanzelf op de juiste onderwerpen. Eigenlijk is het binnen een project zo dat er belanghebbenden zijn die behoeften hebben aan het systeem. Deze behoeften kunnen dan in eisen worden omgezet. Aangezien dit project maar één belanghebbende bevat zullen we alle eisen met z'n tweëen moeten bespreken.

RS: Wat bedoel je precies met het bespreken aan de hand van niet-functionele eisen?

JH: Ik zeg dan bijvoorbeeld: “ik wil graag de eisen aan de availability bespreken”. Een ander voorbeeld is eisen aan de security. Op die manier komen we op de goede gesprekspunten. Als er aan het eind nog eisen over zijn die besproken moeten worden dan kan dat nog.

RS: Okee, dat is goed.

JH: Aan de hand van de probleemstelling en doelstelling heb ik zelf al een aantal behoeften kunnen “raden”. Nu wil ik graag kijken of die kloppen. De eerste is uiteraard scalability aangezien het om een cloud oplossing gaat. Zelf had ik bedacht dat de eis zou moeten zijn: “Resources moeten schaalbaar zijn binnen de oplossing”.

RS: Uiteraard, schaalbaarheid is natuurlijk het key element van cloud computing. Ik zou graag willen dat klanten hun systemen kunnen schalen. Natuurlijk werken we met oude hardware dus is de schaalbaarheid gering. Ook wil ik dat klanten hun storage kunnen schalen. Ze moeten zogezegd extra storage kunnen “bestellen”.

JH: Okee, was dat alles omtrent scalability?

RS: Ik geloof het wel.

JH: Dan wil ik het graag hebben over de availability van het systeem.

RS: De availability van de low-cost cloud oplossing is minder belangrijk. De klanten weten dat ze minder betalen en de availability zal dan ook minder zijn. Het zal minder erg zijn als een klant offline is dan in onze Red cloud. Ik wil wel dat er redundantie is ingebouwd in de storage.
Maar goed, dit gaat om een proof of concept die op wordt gezet door één persoon in een paar maanden. Het liefst wil ik een oplossing waar alle onderdelen een goede failover hebben, maar dat kan niet in dit project.

Daarom wil ik alleen een failover op de meest kritieke plek in het proof of concept.

JH: Okee, de volgende vraag zou in de richting van disaster recovery zijn. De storage oplossing(en) moeten dus redundantie bevatten?

RS: Ja, dat klopt. Dan gaat de (belangrijke) data van klanten minder makkelijk verloren. Deze harde eis is het zelfde als die voor de availability.

JH: Dan is de volgende back-up.

RS: Hmm. In de uiteindelijke oplossing is het zo dat er niet actief back-ups worden aangeboden. Wel is het zo dat achter de schermen back-ups moeten worden gemaakt. Als er echt iets mis gaat dan kan de back-up terug worden gehaald. Dan denk ik dat één keer per week een back-up wel goed is aangezien het een low cost oplossing wordt. Klanten mogen echter wel de mogelijkheid hebben om back-ups te maken. Als er maar een mogelijkheid bestaat om een back-up te maken van de klant data.
Ook wil ik graag hebben dat er een back-up gemaakt kan worden van de systeemdata van OpenStack. De plaatsing van de back-ups maakt niet zo veel uit voor dit proof of concept. Ik wil gewoon dat de back-ups gemaakt kunnen worden.

JH: Juist. Was dat back-up?

RS: Ja, ik geloof dat dat alles voor back-up was.

JH: Okee, dan een belangrijke: de kosten.

RS: Ja, dat is inderdaad een belangrijk punt voor dit project. Het grootste deel van de behoeften over de kosten was al boven tafel geloof ik. Ten eerste moet het een goedkoop alternatief worden voor onze Red cloud.

JH: Dat betekent dus dat in de winstberekening de theoretische abonnementskosten voor de klant lager moeten liggen dan de kosten bij de Red cloud, lijkt me?

RS: Dat klopt. Dan kom ik ook nog op een ander punt. De hardware uitbreidingen in het systeem. Deze moeten ook worden meegewogen in de winstberekening als extra kosten. Bij het gebruik van uitbreidingen wil ik graag dat je het met mij overlegt. Het liefst zo min mogelijk uitbreidingen gebruiken.

JH: Dat is duidelijk.

RS: Ook wil ik de theoretische winst weten aan het eind van het project. Maar goed, dat is een onderdeel van de winstberekening.

JH: Okee, waren dat de behoeften in de richting van winst/cost?

RS: Ja.

JH: De volgende is dan de capaciteit van het systeem.

RS: Daar heb ik geen behoeften over denk ik.

JH: Er zijn inderdaad nog honderden systemen over. Aan de capaciteit van de server is dus al voldaan. En een capacity planning? Moet ik die hebben?

RS: Wat betreft een capacity planning, die hoef je niet te hebben.

JH: De extensibility van het proof of concept, zijn hier nog behoeften op te geven?

RS: Nee, daar heb ik niets over. Nou ja, het moet uitbreidbaar zijn met Ceph.

JH: Ja, inderdaad. OpenStack is op zichzelf al zeer extensible door de modulaire opzet.

JH: Okee, en de maintainability?

RS: Die hoeft niet terug te komen in het proof of concept.

JH: Ja, de schijven moeten hot-swappable zijn, maar dat is een feit, geen eis. Ook heb ik al gezien dat ze dat zijn, dus dat komt wel goed.

JH: Dan wil ik vragen of je behoeften hebt aan de opbouw van het netwerk, oftewel network.

RS: Ehm, niet direct geloof ik. O, misschien heb ik wel iets. Ik vind het belangrijk dat klanten IP-adressen kunnen bestellen voor hun instances. Met deze IP-adressen kunnen ze dan op het i3D netwerk. En ook wil ik graag dat je geen SNAT of DNAT gebruikt in je oplossing. Hier hebben we slechte ervaringen mee. Dan was dat hem voor network.

JH: Haha, okee. Dan nog een belangrijke: De performance.

RS: Daar kunnen we denk ik niet veel over zeggen.

JH: ja, precies, we weten niet wat de systemen doen.

RS: Maar ik wil wel niet dat de verbindingen tussen de nodes boven de 50% komt. We weten al dat je gigabit verbindingen hebt. Dit betekent dus: niet boven de 500Mb/s. Maar dit zal je denk ik ook niet halen. In onze andere cloud oplossing blijft het er ook ver onder.

JH: nou, dat denk ik ook niet dan.

RS: De verschillende klanten moeten ook geen last van elkaar krijgen op hun VM's. De VM's moeten werkbaar blijven.

JH: De noisy neighbour?

RS: Ja, precies.

JH: Maar hoe definiëren we “werkbaar”?

RS: De vertraging/slowdowns op de VM's mogen niet hoger dan 200% zijn. Het is dus zo dat er iemand hoog verbruik heeft op zijn VM een ander hier geen last van krijgt.

RS: Ja, precies. Meer weet ik niet te zeggen van de performance. De low-cost cloud oplossing is ook niet bedoeld als high performance cloud. Net als bij de availability weten de mensen dat het wel eens minder snel kan zijn.

JH: Okee, mooi. De volgende eis is open source. Op zich is dat een heel voor de hand liggende eis. De software die gebruikt wordt voor het opzetten is OpenStack en eventueel Ceph. Deze stukken software zijn open source.

RS: Dat klopt wel ja.

JH: Wat ook meteen opkomt is natuurlijk de wetgeving of compliance eis. Vooral omdat het een cloud oplossing is.

RS: Op het gebied van wetgeving hoeft er weinig aandacht geschonken te worden aan de opstelling. Wij gaan er nu gewoon van uit dat het alleen draait in Nederland.

JH: Dat is duidelijk. Hoe zit het eigenlijk met de efficiency?

RS: Als de systemen niet gebruikt worden dan staan ze uit om kosten te sparen natuurlijk. Je zou wel uit kunnen zoeken of er een mogelijkheid in OpenStack bestaat om automatisch in- en uit te schakelen. De systemen die nu actief zijn binnen i3D staan aan met een overhead van 150%, de rest is uit. Er zijn ook technieken om automatisch de servers aan te zetten als ze uit staan. Hier zou je even naar kunnen kijken. Toch denk ik niet dat dit mogelijk is op die oude servers. Maar echt nodig vind ik dit nu niet.

JH: Platform compatibility, zijn hier nog eisen voor? Is er bijvoorbeeld de eis om een Linux distributie én een Windows product te kunnen draaien?

RS: Op zich is het wel goed om te kijken of een versie van Windows werkt op OpenStack. Alleen zullen de toekomstige klanten hier weinig gebruik van maken. Dat komt door de licentiekosten. In de Azure cloud zijn de licentiekosten al verrekend met de Azure software. Als een Windows image op OpenStack draait zullen personen de bijkomende licentiekosten moeten betalen. Dan wordt het minder low cost uiteraard. Daarom voel ik geen behoefte om dit een eis te maken.

JH: En de documentation eis. Zo ga ik er van uit dat er een volledige handleiding moet worden geschreven over hoe het proof of concept is opgezet?

RS: Uiteraard.

JH: Zijn er aan het rack, de server of het netwerk nog resource constraints?

RS: De enige die ik kan bedenken zijn dat je maximaal 32 ampére stroomsterkte in het rack hebt en maximaal 1Gbit netwerksnelheid. Over die 32 amps ga je niet heen.

JH: In het eerste gesprek over de opdracht werd er gesproken over het testen met SSD's en HDD's in de OpenStack oplossing. Is het echt zo dat ik een deel van de opstelling met SSD's moet gaan uitrusten?

RS: Nee, eigenlijk willen we dat je alleen HDD's gebruikt. In deze oplossing zijn SSD's veel te duur, dan blijft het geen low-cost cloud oplossing meer.

JH: Dan heb ik denk ik wel zo'n beetje mijn lijstje gehad. Zijn er nog meer eisen waar het systeem aan moet voldoen?

RS: Mwa, er moet wel minimaal één instance kunnen worden gehost op de omgeving. Deze instance mag van een zelf gekozen distibutie zijn. Ook wil ik nog dat het verbruik van klanten gemeten kan worden.

JH: Waarmee de uiteindelijke rekening gemaakt kan worden?

RS: Precies ja!

JH: Dan hebben we alle behoeften wel zo'n beetje?

RS: Zo denk ik wel dat we alle eisen aan het systeem rond hebben.

JH: Okee, bedankt voor je tijd!

Bijlage V: Winstberekening

Kosten

Hardware

Soort	Merk / Type	Aanschaf (tweedehands)	Afschrijving per jaar	Aantal cores	Hoeveelheid RAM (GB)	Min. verbruik (Watt)	gem. Verbruik (Watt)	50% load verbruik (Watt)	Max. verbruik (Watt)
Server	HP Proliant DL120 G6	€ 10,00	€ 0,00	4	8	61,5125	113,18625	110,4775	140,3775
Switch	Dell Powerconnect 5548	€ 15,00	€ 0,00			50	85		120
Uitbreidingskaart netwerk	Broadcom D43042 PCIe	€ 10,00	€ 0,00						
RAID-kaart	HP Smart Array P410	€ 20,00	€ 0,00						
Hard drive	250GB (verschillende merken)	€ 5,00	€ 0,00			7,4875	9,1225		

Vaste lasten

Soort	Inkoop	per [eenheid]
Stroom	€ 0,19	KwH
Dataverbruik	€ 0,50	Mbit
Rack huur	€ 350,00	Maand

Nodes

Type node/server	Aantal	Tot. Netw.int. Per node	Instances per node	Totaal instances per node	Aantal HDD's per node (MAX 4)	Aantal HDD's totaal	Verbruik (Watt)	Per jaar (24/7) (KW)	Per maand (24/7) (KW)	Str.kosten per maand
Network node	2	8	26	52	1	2	122,30875	2142,8493	178,570775	€ 33,93
Compute node	3	12	7	21	1	3	122,30875	3214,27395	267,8561625	€ 50,89
Controller node	2	6	16	32	1	2	122,30875	2142,8493	178,570775	€ 33,93
Block storage node	2	6	40	80	4	8	149,67625	2622,3279	218,527325	€ 41,52
NTP-node	1	3			1	1	122,30875	1071,42465	89,2853875	€ 16,96
Repository node	1	3			1	1	122,30875	1071,42465	89,2853875	€ 16,96
SSH-jump node	1	3			1	1	122,30875	1071,42465	89,2853875	€ 16,96
Utility node	1	2			1	1	122,30875	1071,42465	89,2853875	€ 16,96
Object storage nodes	3	6	25	75	4	12	149,67625	3933,49185	327,7909875	€ 62,28
Proxy server	2	4			1	2	122,30875	2142,8493	178,570775	€ 33,93
Totaal Nodes	18	53			16	33		20484,3402	1707,02835	€ 324,34
Switches	2						85	1489,2	124,1	€ 23,58
Totaal systemen	20	53			16	33		21973,5402	1831,12835	**€ 347,91**

Baten verwacht

Constanten

	vcore	vmem (GB)	Opslagruimte (block) (GB)	Opslagruimte (object) (GB)
Standaard flavor	1	1	20	30

Prijzen

Standaard flavor instance	€ 12,50

Klanten per compute node

	Fysiek	Overprovisie	virtueel	Aantal klanten
Cores	4	2	7	7
Geheugen (GB)	8	1	7	7
			Klanten per compute node:	7
			Opbrengst per compute node:	€ 87,50
			Opbrengst totaal:	**€ 262,50**

Winstberekening

Basisopstelling OpenStack winstberekening

Stroomkosten per maand	€ 347,91
Huur RAID-kaarten per maand	€ 3,33
Huur Netwerkkaarten per maand	€ 4,17
Huur servers per maand	€ 15,00
Huur HDD's per maand	€ 13,75
Rack huur per maand	€ 145,83
Subtotaal	€ 530,00
Opbrengst per compute node	€ 262,50
Totaal	**€ 267,50-**

DE HAAGSE HOGESCHOOL

Kosten

Hardware

Soort	Merk / Type	Aanschaf (tweedehands)	Afschrijving per jaar	Aantal cores	Hoeveelheid RAM (GB)	Min. verbruik (Watt)	gem. Verbruik (Watt)	50% load verbruik (Watt)	Max. verbruik (Watt)
Server	HP Proliant DL120 G6	€ 10,00	€ 0,00	4	8	61,5125	113,18625	110,4775	140,3775
Switch	Dell Powerconnect 5548	€ 15,00	€ 0,00			50	85		120
Uitbreidingskaart netwerk	Broadcom D43042 PCIe	€ 10,00	€ 0,00						
RAID-kaart	HP Smart Array P410	€ 20,00	€ 0,00						
Hard drive	250GB (verschillende merken)	€ 5,00	€ 0,00			7,4875	9,1225		

Vaste lasten

Soort	Inkoop	per [eenheid]
Stroom	€ 0,19	KwH
Dataverbruik	€ 0,50	Mbit
Rack huur	€ 350,00	Maand

Nodes

Type node/server	Aantal	Tot. Netw.int. Per node	Instances per node	Totaal instances per node	Aantal HDD's per node (MAX 4)	Aantal HDD's totaal	Verbruik (Watt)	Per jaar (24/7) (KW)	Per maand (24/7) (KW)	Str.kosten per maand
Network node	5	20	26	130	1	5	122,30875	5357,12325	446,4269375	€ 84,82
Compute node	17	68	7	119	1	17	122,30875	18214,21905	1517,8515875	€ 288,39
Controller node	8	24	16	128	1	8	122,30875	8571,3972	714,2831	€ 135,71
Block storage node	3	9	60	180	4	12	149,67625	3933,49185	327,7909875	€ 62,28
NTP-node	1	3			1	1	122,30875	1071,42465	89,2853875	€ 16,96
Repository node	1	3			1	1	122,30875	1071,42465	89,2853875	€ 16,96
SSH-jump node	1	3			1	1	122,30875	1071,42465	89,2853875	€ 16,96
Utility node	1	2			1	1	122,30875	1071,42465	89,2853875	€ 16,96
Object storage nodes	5	10	25	125	4	20	149,67625	6555,81975	546,3183125	€ 103,80
Proxy server	2	4			1	2	122,30875	2142,8493	178,570775	€ 33,93
Totaal Nodes	44	146			16	68		49060,599	4088,38325	€ 776,79
Switches	4						85	2978,4	248,2	€ 47,16
Totaal systemen	48	146			16	68		52038,999	4336,58325	**€ 823,95**

Baten verwacht

Constanten

	vcore	vmem (GB)	Opslagruimte (block) (GB)	Opslagruimte (object) (GB)
Standaard flavor	1	1	20	30

Prijzen

Standaard flavor instance	€ 12,50

Klanten per compute node

	Fysiek	Overprovisie	virtueel	Aantal klanten
Cores	4	2	7	7
Geheugen (GB)	8	1	7	7
			Klanten per compute node:	7
			Opbrengst per compute node:	€ 87,50
			Opbrengst totaal:	**€ 1.487,50**

Winstberekening

Basisopstelling OpenStack winstberekening

Stroomkosten per maand	€ 823,95
Huur RAID-kaarten per maand	€ 5,00
Huur Netwerkkaarten per maand	€ 18,33
Huur servers per maand	€ 36,67
Huur HDD's per maand	€ 28,33
Rack huur per maand	€ 350,00
Subtotaal	€ 1.262,28
Opbrengst per compute node	€ 1.487,50
Totaal	€ 225,22

i3D.NET PERFORMANCE HOSTING

Bijlage VI: Handleiding

Handleiding OpenStack proof of concept

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Stagebedrijf: i3D
Plaats: Rotterdam
Datum: 29-02-2016

Versiebeheer

Versie	Datum	Wijziging
0.1	29-02-2016	Layout opgezet
0.2	25-03-2016	Eerste conceptversie
0.3	05-04-2016	Spel- en grammaticafouten verbeterd. Opmaak verbeterd.
1.0	20-05-2016	Definitieve versie

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

Inleiding

In dit document worden de acties beschreven die uitgevoerd moeten worden om tot hetzelfde proof of concept te komen als in dit project. Met deze handleiding wordt het proof of concept dus reproduceerbaar. Alle hoofdstukken stellen een OpenStack component voor. In het laatste hoofdstuk wordt verder ingegaan op het opzetten van de testomgevingen.

In het document worden verschillende manieren gebruikt om aanpassingen, code, input en dergelijke aan te geven. Hier onder wordt uitgelegd hoe al deze verschillende stijlen worden weergeven.

Een stuk code wordt in de tekst *schuingedrukt* weergegeven, zoals:

restrict -4 default kod notrap nomodify nopeer noquery
restrict -6 default kod notrap nomodify nopeer noquery

Een commando wordt in de tekst schuingedrukt en met een hekje weergeven, zoals:

#echo "ABCDEF"

Ook kunnen er variabelen worden gebruikt in een commando, zoals een IP-adres. Deze waarden zijn afhankelijk van de opstelling. Variabelen worden weergegeven tussen blokhaakjes gevolgd door de waarde die is gebruikt in het proof of concept, zoals:

bind-address = [IP adres van controller management netwerk interface][123.456.789.0]

Er zijn momenten dat het aanroepen van een commando een volgende actie vereist. Een voorbeeld hiervan is het aanroepen van *#mysql -u root -p*. Direct na dit commando wordt het password gevraagd. Binnen deze handleiding wordt dit weergeven met twee hekjes, zoals:

#mysql -u root -p
##[password van de mysql root user]

Als er in een stuk code iets een specifiek stuk moet worden aangepast zal eerst de oude waarde worden weergeven met de tekst "*VOOR:*" er voor. De nieuwe waarde die er hoort te staan zal worden weergeven met "*NA:*" er voor, zoals:

VOOR: *admin_token = ADMIN_TOKEN*
NA: *admin_token = [password]*

De inhoud van nieuwe scripts die moeten worden aangemaakt zullen worden weergeven in het `courier lettertype` en schuingedrukt.

i3D.NET PERFORMANCE HOSTING

DE HAAGSE HOGESCHOOL

Inhoudsopgave

1. Voorbereidingen en benodigdheden

1.1 Hardware

De hardware voor dit proof of concept is vastgesteld op:

- 10 Servers: HP proliant DL120 G6.
- 2 Switches: Dell powerconnect 5548.

1.2 Software

De software voor dit proof of concept is vastgesteld op:

- Ubuntu 14.04 LTS server edition (geen GUI).
- OpenStack componenten uit de cloud repository van Ubuntu.
- Putty: om een SSH-verbinding naar de server te kunnen realiseren.

1.3 Passwords

De password die worden gebruikt in het proof of concept zijn opgeslagen in Keepass. Het genereren van de wachtwoorden is gebeurd door het uitvoeren van het volgende command:

< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-20};echo;

Met dit commando wordt er een random waarde opgevraagd met urandom. Vervolgens wordt er met tc een filter overheen gegooit zodat de output alleen de opgegeven tekens bevat (A~Z, a~z, 0~9). Deze output wordt vervolgens afgekapt op de opgegeven lengte met head. De lengte van de passwords wordt voor dit proof of concept gehouden op 20 karakters.

1.4 OS installeren

Om de installaties van Ubuntu 14.04 LTS op de servers te krijgen is er gebruik gemaakt van het control panel. Vanuit dit panel is met genoeg privileges een server te installeren met Ubuntu 14.04 LTS. Bij dit proces wordt gevraagd om:

- Hostname
- Password
- Partitionering

1.4.1 Partitionering

Als eerste is de partitionering van de controller node(s) bepaald. De controller nodes hebben genoeg aan een enkele relatief kleine harde schijf. Er is op deze nodes geen behoefte aan opslag voor cloudusers en instances. De partitionering voor de **controller nodes** is als volgt gepland:

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	65536 MB	ext3	Belangrijk dat er goed gelogd kan blijven worden. Om in het proof of concept Glance images te kunnen uploaden/downloaden.
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering voor de **compute nodes** is als volgt gepland:

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	65536 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering voor de **network nodes** is als volgt gepland:

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	65536 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering voor de **block storage nodes** is als volgt gepland:

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	65536 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize
Geen	Geen	Rest (~42GB)	Geen	Deze lege, ongepartitioneerde ruimte is bedoeld voor de

				LVM ruimte voor Cinder.

De partitionering voor de **Utility node** is als volgt gepland:

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering van de Swift proxy server is als volgt:

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	8192 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering van de Swift storage servers is als volgt:

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	8192 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

Na het accepteren van de instellingen zal de installatie van Ubuntu 14.04 LTS automatisch worden doorlopen. Nadat de installatie succesvol is afgerond kan er een SSH verbinding worden opgezet naar de server. Bij het creëren van het proof of concept is er gekozen voor PuTTY als SSH client.

1.5 Users

Bij de installatie van elke node moet de root user allereerst een ander password opgegeven krijgen. Dit password komt van het commando die is gedefinieerd in hoofdstuk 1.3.

Het password van de root user wordt gewijzigd met:

#passwd [password]

Nadat de root user een ander password heeft gekregen moet er een administrator user aan worden gemaakt voor die node. De benaming van de administrators is hetzelfde als de naam van de server. Zo krijgt bijvoorbeeld *server1986* de user *administrator1986*. Om de user aan te maken wordt het volgende commando uitgevoerd:

#adduser [username]

Users moeten ook enige permissies hebben en lid zijn van enige groepen om normaal te kunnen functioneren. Ten eerste moeten de admin gebruikers (bijvoorbeeld admin1986) logs kunnen lezen. Dit kan door ze lid te maken van de *adm* groep. Dit wordt gedaan door:

#usermod -a -G adm [administrator username]

Nu kan worden gecontroleerd of de admin lid is van de groep met:

#groups [administrator username]

1.6 OpenStack repository

Om de packages die benodigd zijn voor OpenStack te kunnen bemachtigen moet er een repository worden toegevoegd. Dit moet worden uitgevoerd op elke fysieke OpenStack node. Dit kan worden gedaan met de volgende commando's:

 # sudo apt-get install software-properties-common
 # sudo add-apt-repository cloud-archive:liberty

1.7 update en OpenStack client

Het OS en de toegevoegde OpenStack repository moeten worden geüpdated naar de laatste versies. Dit kan worden gedaan met het volgende commando:

#sudo apt-get update
#sudo apt-get dist-upgrade

Na het updaten en upgraden kan de laatste versie van de OpenStack client worden gedownload. Dit wordt gedaan met:

#sudo apt-get install python-openstackclient

2. Controller node opzetten

De eerste node die opgezet hoort te worden is de controller node. Er wordt in deze handleiding van uit gegaan dat op deze node nu al besturingssysteem aanwezig is (hoofdstuk 1.4) en een root user en admin user bestaan. Deze users moeten een gewijzigd password hebben volgens hoofdstuk 1.5. Tevens moet de OpenStack repository toegevoegd zijn en het OS geüpdatet (hoofdstuk 1.6 en 1.7).

2.1 Database

Binnen OpenStack is het van belang dat er een database wordt opgezet. In deze databases wordt informatie opgeslagen over alle componenten/services. Er is besloten om vanwege bedrijfsstandaard een MySQL-database op te zetten.

2.1.1 MySQL

Eerst moeten de packages worden gedownload. Dit wordt gedaan met:

#apt-get install mysql-server python-pymysql

Direct bij de installatie moet een root password worden ingegeven. Deze moet worden gegenereerd met het in hoofdstuk 1 besproken commando om wachtwoorden te genereren.
Hierna moet er een config bestand worden gemaakt voor de database. Het bestand moet worden aangemaakt in de directory */etc/mysql/conf.d/*. Het bestand moet *mysqld_openstack.cnf* worden genoemd. De volgende lijnen met code moet in het bestand komen:

[mysqld]
Set to Management IP
bind-address = [IP-adres van controller management netwerk interface][10.0.1.2]
default-storage-engine = innodb
innodb_file_per_table
collation-server = utf8_general_ci
init-connect = 'SET NAMES utf8'
character-set-server = utf8

Het bind address is een IP-adres die de controller node gekregen heeft om aangesloten te zijn op het management netwerk.

Om de MySQL-database te beveiligen kan het *mysql_secure_installation* script worden gedraaid. De instellingen die dit script doorvoert kunnen ook handmatig worden aangepast.
Eerst vraagt het script of het root password gewijzigd moet worden. Aangezien er al een root password is gemaakt is dit niet meer nodig.
Hierna vraagt het script of de anonymous user toegang moet hebben. In principe maakt dit niet zo veel uit aangezien dit een proof of concept betreft. De anonymous user wordt toch geweigerd in het proof of concept.
Vervolgens vraagt het script of de root user in de MySQL-database geweigerd moet worden als hij op afstand toegang probeert te krijgen. Als er wijzigingen aan de database gemaakt moeten worden zal dit via SSH gebeuren. Deze optie moet dus ingesteld worden.
Hierna vraagt het script of de “test” database verwijderd moet worden. Aangezien iedereen toegang heeft tot de test database kan dit een beveiligingsrisico zijn. Deze database moet dus worden verwijderd.
De entries die nu overblijven in de user database zijn: root@localhost, root@172.0.0.1, root@::1 en

debian-sys-maint@localhost.

2.1.2 NoSQL

Om de telemetry (Ceilometer) service te installeren en te gebruiken moet er naast een MySQL-database ook een NoSQL-database worden geïnstalleerd.

Eerst moeten de packages worden geïnstalleerd. Dit wordt gedaan met:

#apt-get install mongodb-server mongodb-clients python-pymongo

Vervolgens moet de config file worden gewijzigd. De wijzing die moet plaatsvinden is het aanpassen van het IP-adres waar de NoSQL-database op benaderd kan worden. Om dit te bereiken moet de volgende waarde worden veranderd:

bind_ip = [IP-adres van controller management netwerk interface][10.0.1.2]

Hierna moet de NoSQL-service worden herstart. Dit wordt gedaan met:

#service mongodb restart

2.2 Message queue

Om binnen OpenStack communicatie tussen de verschillende componenten mogelijk te maken moet er een message queue worden opgezet. Er wordt gebruik gemaakt van RabbitMQ.
Eerst moeten de packages worden gedownload. Dit wordt gedaan met:

#apt-get install rabbitmq-server

Vervolgens moet er een nieuwe user voor RabbitMQ aangemaakt worden genaamd “openstack”. Dit kan worden gedaan met:

#rabbitmqctl add_user openstack [Password voor de openstack user]

Ten slotte moet de nieuwe RabbitMQ “openstack” user de juiste permissies krijgen. Uiteraard moet de openstack user read, write en configuration access krijgen, dit is volledige access. Dit wordt gedaan met:

#rabbitmqctl set_permissions openstack "." ".*" ".*"*

3. Controller node configuratie

In dit hoofdstuk wordt het configureren van de controller node besproken. De controller node is een node met uiteenlopende OpenStack services bij elkaar gepakt.

3.1 Keystone identity service

Om de keystone service op te zetten moet er eerst een database worden aangemaakt. Deze database wordt aangemaakt op de MySQL-server op de controller node. Dit kan worden gedaan met:

#mysql -u root -p
##[password van de mysql root user]
#CREATE DATABASE keystone;

Uiteraard moet de keystone database volledig bereikbaar zijn voor de keystone user. Dit wordt gedaan met:

```
#GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'localhost' IDENTIFIED BY '[keystone database password]';
#GRANT ALL PRIVILEGES ON keystone.* TO 'keystone'@'%' IDENTIFIED BY '[keystone database password]';
```

De laatste voorbereiding die getroffen moet worden is het automatisch laten starten van de Keystone service na de installatie uitschakelen. Dit kan worden gedaan met een override script:

#echo “manual” > /etc/init/keystone.override

Na het aanmaken van dit script kunnen de Keystone packages worden geïnstalleerd. Dit wordt gedaan met:

#apt-get install keystone apache2 libapache2-mod-wsgi memcached python-memcache

3.1.1 Keystone configuration file

Binnen Keystone moet een standaard admin_token worden aangemaakt voordat de API gebruikt kan worden. Er wordt door OpenStack wel aangeraden om het admin token uit te zetten voor een productie cloud. Dit admin token kan ingegeven worden in de keystone configuratie. De locatie van deze file is */etc/keystone/keystone.conf/*.
Het admin token kan worden aangemaakt door onder de *[default]* kop het volgende te veranderen:

VOOR: *admin_token = ADMIN_TOKEN*
NA: *admin_token = [password]*

In het zelfde configuratie bestand moeten ook de volgende variabelen worden aangepast:
De database die keystone moet gebruiken. Dit staat onder de *[database]* kop:

VOOR: *#connection = <None>*
NA: *connection = mysql+pymysql://keystone:[Keystone database password]@controller/keystone*

De memcache installatie die keystone moet gebruiken. Dit staat onder de *[memcache]* kop.

VOOR: *#servers = localhost:11211*
NA: *servers = localhost:11211*

Het token backend moet weten welk backend er moet worden gebruikt. Dit staat onder de *[token]* kop.

VOOR: *#provider = uuid*
#driver = sql
NA: *provider = uuid*
driver = memcache

Er is binnen Keystone ook behoefte aan het revoken van tokens. Dit wordt opgeslagen in een SQL-database.

VOOR: *#driver = sql*
NA: *driver = sql*

Om uitgebreide logging op te zetten met Keystone zal de verbose variabele op true gezet moeten worden. Dit kan handig zijn bij troubleshooting.

VOOR: *#verbose = true*
NA: *verbose = true*

Dat waren alle aanpassingen aan het keystone.conf bestand. De Keystone database moet nog ingericht worden. Dit gebeurt met het volgende commando:

#su -s /bin/sh -c "keystone-manage db_sync" keystone

3.1.2 Apache server

Om de Keystone server actief te krijgen moet ook de Apache server ingesteld worden. Als eerste moet de Apache installatie weten waar de Keystone installatie te bereiken is. Dit moet worden aangepast in het */etc/apache2/apache2.conf* bestand. Binnen de *global configuration* kop moet er een nieuwe variabele worden aangemaakt met het IP-adres of hostname van de controller node:

ServerName [IP-adres van de controller]

Vervolgens moet er een nieuwe bestand worden aangemaakt binnen */etc/apache2/sites-available/*. Het bestand moet de naam *wsqi-keystone.conf* heten. De inhoud van dit bestand moet het volgende worden:

```
Listen 5000
Listen 35357

<VirtualHost *:5000>
    WSGIDaemonProcess keystone-public processes=5 threads=1 user=keystone
group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-public
    WSGIScriptAlias / /usr/bin/keystone-wsgi-public
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    <IfVersion >= 2.4>
      ErrorLogFormat "%{cu}t %M"
```

```
    </IfVersion>
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        <IfVersion >= 2.4>
            Require all granted
        </IfVersion>
        <IfVersion < 2.4>
            Order allow,deny
            Allow from all
        </IfVersion>
    </Directory>
</VirtualHost>

<VirtualHost *:35357>
    WSGIDaemonProcess keystone-admin processes=5 threads=1 user=keystone
group=keystone display-name=%{GROUP}
    WSGIProcessGroup keystone-admin
    WSGIScriptAlias / /usr/bin/keystone-wsgi-admin
    WSGIApplicationGroup %{GLOBAL}
    WSGIPassAuthorization On
    <IfVersion >= 2.4>
      ErrorLogFormat "%{cu}t %M"
    </IfVersion>
    ErrorLog /var/log/apache2/keystone.log
    CustomLog /var/log/apache2/keystone_access.log combined

    <Directory /usr/bin>
        <IfVersion >= 2.4>
            Require all granted
        </IfVersion>
        <IfVersion < 2.4>
            Order allow,deny
            Allow from all
        </IfVersion>
    </Directory>
</VirtualHost>
```

Om deze virtual hosts actief te laten draaien moeten ze nog enabled worden:

#ln -s /etc/apache2/sites-available/wsgi-keystone.conf /etc/apache2/sites-enabled

Nu moet de service alleen nog opnieuw worden opgestart:

#service apache2 restart

3.1.3 Service entity

Om gebruik te maken van OpenStack moet er eerst een tijdelijk token worden aangemaakt. Dit token heet het *ADMIN_TOKEN*. Hiermee kan OpenStack worden geconfigureerd. Dit token worden geset met:

#export OS_TOKEN=[Keystone token]

#export OS_URL=http://[management IP-adres van de controller][10.0.1.2]:35357/v3
#export OS_IDENTITY_API_VERSION=3

Om de catalogus voor Keystone aan te maken wordt het volgende commando gebruikt:

#openstack service create –-name keystone –-description “OpenStack Identity” identity

In de catalogus moeten verschillende endpoints worden aangemaakt zodat de services weten waar de andere services te vinden zijn:

#openstack endpoint create –-region RegionOne identity public http://[management IP-adres van de controller][10.0.1.2]:5000/v2.0
#openstack endpoint create –-region RegionOne identity internal http://[management IP-adres van de controller][10.0.1.2]:5000/v2.0
#openstack endpoint create –-region RegionOne identity admin http://[management IP-adres van de controller][10.0.1.2]:35357/v2.0

3.1.4 Users, roles en tenants

Om de omgeving te kunnen testen is er al een domain aanwezig binnen OpenStack, genaamd: “default”. Om te testen wordt hier een admin tenant (project) aangemaakt. Dit kan met het volgende commando:

#openstack project create –-domain default –-description “admin project” admin

Binnen het default domain moeten er ook users bestaan. Om te beginnen kan de user “admin” worden aangemaakt. Dit kan met het volgende commando:

#openstack user create –-domain default –-password-prompt admin

Bij de admin user hoort ook een role. Deze kan worden aangemaakt met het volgende commando:

#openstack role create admin

Deze role moet hierna ook worden toegevoegd aan de admin tenant en de admin user:

#openstack role add –-project admin –-user admin admin

Om ook de werking van normale (unprivileged) users te testen wordt er een test tenant en user aangemaakt Deze test user krijgt ook een role toegewezen gekregen:

#openstack project create –-domain default –-description “test Project” test
#openstack user create –-domain default –-password-prompt test
#openstack role create user
#openstack role add –-project test –-user test user

Om binnen een domain alle services een user te geven wordt er ook een service tenant aangemaakt.

#openstack project create --domain default --description "Service Project" service

3.1.5 Functietest

Na het opzetten van Keystone moet het tijdelijke admin_token worden verwijderd voor de veiligheid. Dit wordt gedaan door het bestand *keystone-paste.ini* aan te passen in de directory */etc/keystone/*. Onder de koppen [pipeline:public_api], [pipeline:admin_api] en [pipeline:api_v3] moet de code "*admin_token_auth*" worden verwijderd.

[pipeline:public_api]
VOOR: *pipeline = sizelimit url_normalize request_id build_auth_context token_auth admin_token_auth json_body ec2_extension user_crud_extension public_service*
NA: *pipeline = sizelimit url_normalize request_id build_auth_context token_auth json_body ec2_extension user_crud_extension public_service*

[pipeline:admin_api]
VOOR: *pipeline = sizelimit url_normalize request_id build_auth_context token_auth admin_token_auth json_body ec2_extension user_crud_extension admin_service*
NA: *pipeline = sizelimit url_normalize request_id build_auth_context token_auth json_body ec2_extension user_crud_extension admin_service*

[pipeline:api_v3]
VOOR: *pipeline = sizelimit url_normalize request_id build_auth_context token_auth admin_token_auth json_body ec2_extension_v3 s3_extension simple_cert_extension revoke_extension federation_extension oauth1_extension endpoint_filter_extension service_v3*
NA: *pipeline = sizelimit url_normalize request_id build_auth_context token_auth json_body ec2_extension_v3 s3_extension simple_cert_extension revoke_extension federation_extension oauth1_extension endpoint_filter_extension service_v3*

De tijdelijke environment variabelen OS_TOKEN en OS_URL kunnen nu weer worden uitgezet:

unset OS_TOKEN OS_URL

Hierna kan er een token worden aangevraagd met de admin en test user. Deze aanvraag kan later worden gedaan in horizon. Echter kan het nu ook commandline. Hier onder wordt respectievelijk het commando gegeven om als admin een token aan te vragen en om als test een token aan te vragen:

#openstack --os-auth-url http://[controller node IP-adres][10.0.1.2]:35357/v3 --os-project-domain-id default --os-user-domain-id default --os-project-name admin --os-username admin --os-auth-type password token issue

#openstack --os-auth-url http://[controller node IP-adres][10.0.1.2]:5000/v3 --os-project-domain-id default --os-user-domain-id default --os-project-name test --os-username test --os-auth-type password token issue

3.2 Glance image service

Eerst moet er een Glance database worden aangemaakt voor de Glance service:

#mysql -u root -p
##[password van de mysql root user]
CREATE DATABASE glance;

De Glance user moet toegang hebben tot de Glance-database. Dit wordt gedaan door de volgende commando's:

#GRANT ALL PRIVILEGES ON glance. TO 'glance'@'localhost' IDENTIFIED BY '[glance database password]';*
#GRANT ALL PRIVILEGES ON glance. TO 'glance'@'%' IDENTIFIED BY '[glance database password]';*

Om de services te managen moeten de admin credentials worden geset:

export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=admin
export OS_TENANT_NAME=admin
export OS_USERNAME=admin
export OS_PASSWORD=[admin password]
export OS_AUTH_URL=http://[management IP-adres van de controller]:35357/v3
export OS_IDENTITY_API_VERSION=3

De bovenstaande codes kunnen beter in een script worden gezet. Dit script krijgt dan de naam “admin-openrc.sh”. Om dit script uit te voeren wordt het volgende commando gebruikt:

#source admin-openrc.sh

3.2.1 Users, entities en endpoints

Om de glance service te laten runnen is een user nodig speciaal voor die service. In dit geval is het de user “*glance*”. Dit wordt gedaan met het commando:

#openstack user create --domain default --password-prompt glance
##[password die de glance user moet krijgen]
##[password die de glance user moet krijgen, nogmaals]

De nieuw aangemaakte user “*glance*” moet de role van admin toegewezen krijgen. Dit is het geval omdat deze service overal toegang toe moet hebben. De user wordt lid gemaakt van de tenant “*service*”. Dit wordt gedaan met:

#openstack role add –-project service –-user glance admin

Nu moet er een service entity worden aangemaakt:

#openstack service create –-name glance –-description “OpenStack Image service” image

Hierna moeten de API-endpoints worden aangemaakt in de Keystone catalogus:

#openstack endpoint create –-region RegionOne image public http://10.0.1.2:9292
#openstack endpoint create –-region RegionOne image internal http://10.0.1.2:9292
#openstack endpoint create –-region RegionOne image admin http://10.0.1.2:9292

3.2.2 installatie Glance

Nadat de database, user en andere entiteiten zijn aangemaakt kan Glance worden geïnstalleerd. Dit wordt gedaan met:

#apt-get install glance python-glanceclient

3.2.3 Glance glance-api.conf configuration file

Nu moet het configuratiebestand */etc/glance/glance-api.conf* worden gewijzigd.
Onder de kop [DEFAULT] moet de variabele *notification_driver* worden aangepast:

VOOR: *#notification_driver =*
NA: *notification_driver = noop*

Onder de [database] kop moet de manier worden beschreven hoe glance bij de database kan komen:

VOOR: *sqlite_db = /var/lib/glance/glance.sqlite*
NA: *#sqlite_db = /var/lib/glance/glance.sqlite*

VOOR: *#connection = <None>*
NA: *connection = mysql+pymysql://glance:[Glance database password]@[Controller node IP-adres] [10.0.1.2]/glance*

VOOR: *#auth_uri = <None>*
NA: *auth_uri = http://10.0.1.2:5000*
auth_url = http://10.0.1.2:35357

Direct onder de regel met “auth_uri” moeten de volgende regels code worden toegevoegd:

auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = [password van Glance user in Keystone]

Onder de [paste_deploy] kop moet het volgende worden aangepast:

VOOR: *#flavor = <None>*
NA: *flavor = keystone*

Onder de kop [glance_store] moeten de variabelen worden gegeven die zorgen voor de storage binnen Glance:

VOOR: *#default_store = file*
NA: *default_store = file*

VOOR: *filesystem_store_datadir = <None>*
NA: *filesystem_store_datadir = /var/lib/glance/images/*

3.2.4 Glance glance-registry.conf configuration file

Hierna moet het bestand */etc/glance/glance-registry.conf* worden aangepast.
Onder de kop [DEFAULT] moet de variabele *notification_driver* worden aangepast:

VOOR: *#notification_driver =*
NA: notification_driver = noop

Onder de [database] kop moet de manier worden beschreven hoe Glance bij de database kan komen:

VOOR: *sqlite_db = /var/lib/glance/glance.sqlite*
NA: *#sqlite_db = /var/lib/glance/glance.sqlite*

VOOR: *#connection = <None>*
NA: *connection = mysql+pymysql://glance:[Glance database password]@[Controller node IP-adres] [10.0.1.2]/glance*

VOOR: *#auth_uri = <None>*
NA: *auth_uri = http://10.0.1.2:5000*
auth_url = http://10.0.1.2:35357

Direct onder de regel met “auth_uri” moeten de volgende regels code worden toegevoegd:

auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = glance
password = [password van Glance user in Keystone]

Onder de [paste_deploy] kop moet het volgende worden aangepast:

VOOR: *#flavor = <None>*
NA: *flavor = keystone*

3.2.5 Glance service herstarten

Nu moet de Glance database worden ingedeeld. Dit gebeurt met het volgende commando:

#su -s /bin/sh -c “glance-manage db_sync” glance

Tenslotte moet de Glance service worden herstart. Er bestaan twee processen die een herstart nodig hebben, nl:

glance-api en *glance-registry*. Dit gebeurt met het volgende commando:

#service glance-registry restart
#service glance-api restart

3.2.6 Functietest

Om de Glance service te testen moet er naast de globale variabelen die al geet zijn nog een globale variabele worden geset, nl:

#export OS_IMAGE_API_VERSION=2

De bovenstaande environment variabele kan ook worden toegevoegd aan het admin-openrc.sh script:

#echo “export OS_IMAGE_API_VERSION=2” | tee -a admin-openrc.sh demo-openrc.sh

Ook is er een image nodig om de Glance image service te testen:

#wget http://download.cirros-cloud.net/0.3.4/cirros-0.3.4-x86_64-disk.img

De image is nu gedownload in de huidige directory. Deze image moet geüpload worden naar de glance service. De service zal de images nu nog uploaden naar locale controller directory */var/lib/glance/images/*. Dit wordt gedaan met het volgende commando:

#glance image-create --name "cirros" --file cirros-0.3.4-x86_64-disk.img --disk-format qcow2 --container-format bare --visibility public –progress

De geüploade image kan nu worden bekeken in een lijst die glance kan geven:

#glance image-list

Om ook een “normale” image te kunnen gebruiker moet er ook Ubuntu 14.04 cloud worden geüpload in Glance. Deze image kan worden verkregen met het volgende commando:

#wget http://cloud-images.ubuntu.com/releases/14.04.3/release/ubuntu-14.04-server-cloudimg-amd64-disk1.img -P /

Met dit comando wordt de 64-bit versie van Ubuntu 14.04 cloud gedownload. Om het gedownloade bestand in de Glance store te krijgen moet het volgende commando worden uitgevoerd:

#glance image-create --name "Ubuntu_1404_cloud" --file ubuntu-14.04-server-cloudimg-amd64-disk1.img --disk-format qcow2 --container-format bare --visibility public

In de Glance-store is de nieuwe image nu te zien met:

#glance image-list

3.3 Iptables instellen

```
#!/bin/sh
```

```
IPT=/sbin/iptables

## FLUSH
$IPT -F
$IPT -X
$IPT -t nat -F
$IPT -t nat -X
$IPT -t mangle -F
$IPT -t mangle -X

### HTTPS ###

#$IPT -A INPUT -i eth1 -p tcp -d [IP-adres van de controller node op het i3D-
netwerk] --dport 443 -m state --state NEW,ESTABLISHED -j ACCEPT
#$IPT -A OUTPUT -o eth1 -p tcp -s [IP-adres van de controller node op het i3D-
netwerk] --sport 443 -m state --state ESTABLISHED -j ACCEPT

### NTP intern ###

$IPT -A OUTPUT -o eth0 -p tcp -d 10.0.1.15 --dport 123 -m state --state
NEW,ESTABLISHED -j ACCEPT
$IPT -A INPUT -i eth0 -p tcp -s 10.0.1.15 --dport 123 -m state --state ESTABLISHED
-j ACCEPT

### FTP intern ###

-A INPUT -i eth0 -p tcp -m tcp --dport 21 -s 10.0.1.0/24 -m state --state
ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 20 -s 10.0.1.0/24 -m state --state
ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --sport 1024:65535 --dport 20:65535 -s 10.0.1.0/24
-m state --state ESTABLISHED -j ACCEPT

-A OUTPUT -o eth0 -p tcp -m tcp --dport 21 -d 10.0.1.0/24 -m state --state
NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 20 -d 10.0.1.0/24 -m state --state
ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --sport 1024:65535 --dport 20:65535 -d 10.0.1.0/24
-m state --state ESTABLISHED,RELATED -j ACCEPT

### MySQL intern###

iptables -A INPUT -i eth0 -p tcp –-dport 3306 -s 10.0.1.0/24 -m state --state
NEW,ESTABLISHED -j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 3306 -m state --state NEW,ESTABLISHED -j
ACCEPT

### Keystone public ###

iptables -A INPUT -i eth0 -p tcp –-dport 5000 -m state --state NEW,ESTABLISHED -j
```

```
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 5000 -m state --state NEW,ESTABLISHED -j
ACCEPT

### AMQP intern###

iptables -A INPUT -i eth0 -p tcp –-dport 5672 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 5672 -m state --state NEW,ESTABLISHED -j
ACCEPT

### VNC-proxy browser ###

iptables -A INPUT -i eth0 -p tcp –-dport 6080 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 6080 -m state --state NEW,ESTABLISHED -j
ACCEPT

### Nova-API ###

iptables -A INPUT -i eth0 -p tcp –-dport 8773 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 8773 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A INPUT -i eth0 -p tcp –-dport 8775 -s -m state --state NEW,ESTABLISHED
-j ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 8775 -m state --state NEW,ESTABLISHED -j
ACCEPT

### Nova-endpoint ###

iptables -A INPUT -i eth0 -p tcp –-dport 8774 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 8774 -m state --state NEW,ESTABLISHED -j
ACCEPT

### Cinder ###

iptables -A INPUT -i eth0 -p tcp –-dport 8776 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 8776 -m state --state NEW,ESTABLISHED -j
ACCEPT

### Ceilometer ###

iptables -A INPUT -i eth0 -p tcp –-dport 8777 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 8777 -m state --state NEW,ESTABLISHED -j
ACCEPT
```

```
### Glance-registry ###

iptables -A INPUT -i eth0 -p tcp –-dport 9191 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 9191 -m state --state NEW,ESTABLISHED -j
ACCEPT

### Glance-API ###

iptables -A INPUT -i eth0 -p tcp –-dport 9292 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 9292 -m state --state NEW,ESTABLISHED -j
ACCEPT

### Neutron ###

iptables -A INPUT -i eth0 -p tcp –-dport 9696 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 9696 -m state --state NEW,ESTABLISHED -j
ACCEPT

### Keystone admin ###

iptables -A INPUT -i eth0 -p tcp –-dport 35357 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 35357 -m state --state NEW,ESTABLISHED
-j ACCEPT

### SSH intern###

iptables -A INPUT -i eth0 -p tcp --dport 50385 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 50385 -m state --state ESTABLISHED -j
ACCEPT

### default policy DROP

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
```

3. Nova compute service

Bij de setup van de Nova compute node moeten sommige componenten op de controller node komen en sommige componenten op de compute node komen.

3.1 Controller node installatie

Op de controller node moet eerst een Nova database worden aangemaakt voor de compute service:

#mysql -u root -p
##[password van de MySQL root]
#CREATE DATABASE nova
#GRANT ALL PRIVILEGES ON nova. TO 'nova'@'localhost' IDENTIFIED BY '[nova database password]';*
#GRANT ALL PRIVILEGES ON nova. TO 'nova'@'%' IDENTIFIED BY '[nova database password]';*

Voor de zekerheid moeten de admin credentials worden geladen. Dit is niet nodig als deze al geladen staan:

#source admin-openrc.sh

Nu de admin credentials weer geload zijn kunnen OpenStack commando's worden uitgevoerd. Eerst moet er een nova user worden aangemaakt:

#openstack user create –-domain default –-password-prompt nova
##[password die de nova user moet krijgen]
##[password die de nova user moet krijgen, nogmaals]

Hierna moet de role worden gemaakt voor de Nova user:

#openstack role add –-project service –-user nova admin

Er kan worden gecontroleerd of de Nova user is toegevoegd:

#openstack role list --user nova --project service

De Nova service moet ook worden aangemaakt:

#openstack service create –-name nova –-description “Nova compute” compute

Nu moeten de endpoints voor de Nova service worden gemaakt:

#openstack endpoint create --region RegionOne compute public http://[Management controller IP-adres]
[10.0.1.2]:8774/v2/%\(tenant_id\)s
#openstack endpoint create --region RegionOne compute internal http://[Management controller IP-adres]
[10.0.1.2]:8774/v2/%\(tenant_id\)s
#openstack endpoint create --region RegionOne compute admin http://[Management controller IP-adres]
[10.0.1.2]:8774/v2/%\(tenant_id\)s

Op dit moment kunnen de Nova packages worden gedownload voor op de controller node. Dit wordt gedaan met het volgende commando:

#apt-get install nova-api nova-cert nova-conductor nova-consoleauth nova-novncproxy nova-scheduler python-novaclient

De configuratie kan worden aangepast in de */etc/nova/nova.conf* file. Sommige stukken code moeten worden toegevoegd en sommige stukken code moeten gewijzigd worden.

Onder de [DEFAULT] kop moeten de volgende variabelen worden aangepast:

De volgende code moet worden toegevoegd aan het configuratiebestand (/etc/nova/nova.conf):

```
rpc_backend = rabbit
auth_strategy = keystone
my_ip = 10.0.1.2
#network_api_class = nova.network.neutronv2.api.API
#security_group_api = neutron
#linuxnet_interface_driver =
#nova.network.linux_net.NeutronLinuxBridgeInterfaceDriver
#firewall_driver = nova.virt.firewall.NoopFirewallDriver

[database]
connection = mysql+pymysql://nova:[Nova database password]@[controller management
IP-adres][10.0.1.2]/nova

[oslo_messaging_rabbit]
rabbit_host = [controller management IP-adres][10.0.1.2]
rabbit_userid = openstack
rabbit_password = [password RabbitMQ openstack account]

[keystone_authtoken]
auth_uri = http://[Controller management IP-adres][10.0.1.2]:5000
auth_url = http://[Controller management IP-adres][10.0.1.2]:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = nova
password = [nova user password]

[glance]
host = [Controller management IP-adres]

[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Hierna moet de database worden ingedeeld:

su -s /bin/sh -c “nova-manage db sync” nova

Nu moeten alle verschillende services van Nova worden herstart:

#service nova-api restart
#service nova-cert restart
#service nova-consoleauth restart
#service nova-scheduler restart
#service nova-conductor restart
#service nova-novncproxy restart

3.1.1 Overcommit

In het /etc/nova/nova.conf bestand op de controller node moeten een tweetal regels code worden opgenomen om de overcommit in te stellen.

```
cpu_allocation_ratio = 2
ram_allocation_ratio = 1
```

3.2 Compute node installatie

Om een aparte compute node op te zetten moet er uiteraard een besturingssysteem aanwezig zijn op de node. Dat moet Ubuntu14.04 LTS server zijn. Net als bij de andere nodes moet ook op deze node de OpenStack repository worden toegevoegd:

#apt-get install software-properties-common
#add-apt-repository cloud-archive:liberty
#apt-get update
#apt-get dist-upgrade
#apt-get install nova-compute sysfsutils

Nu moet het bestand *nova.conf* in de directory */etc/nova/* worden gewijzigd:

```
rpc_backend = rabbit
auth_strategy = keystone
my_ip = [IP-adres van de compute node][10.0.1.5]

[vnc]
vnc_enabled = True
vncserver_listen = 0.0.0.0
vncserver_proxyclient_address = $my_ip
novncproxy_base_url = http://[IP-adres van de controller node, managementnet]
[10.0.1.2]:6080/vnc_auto.html

[keystone_authtoken]
auth_uri = http://[IP-adres van de management interface van de controller node]
[10.0.1.2]:5000
auth_url = http://[IP-adres van de management interface van de controller node]
[10.0.1.2]:35357
auth_plugin = password
project_domain_id = default
```

```
user_domain_id = default
project_name = service
username = nova
password = [Password van de Nova user in Keystone]

[glance]
host = [IP-adres van de management interface van de controller node][10.0.1.2]

[oslo_messaging_rabbit]
rabbit_host = [IP-adres van de management interface van de controller node]
[10.0.1.2]
rabbit_userid = openstack
rabbit_password = [Password van de openstack user voor RabbitMQ]

[oslo_concurrency]
lock_path = /var/lib/nova/tmp
```

Vervolgens moet de nova-compute service worden herstart. Dit wordt gedaan met het volgende commando:

#service nova-compute restart

3.3 Nieuwe flavor aanmaken

In OpenStack worden flavors gebruikt om aan te geven hoeveel resources een instance toe krijgt gewezen. Om naast Cirros test images ook grotere instances te kunnen booten moet er een tussenmaat worden aangemaakt. Het aanmaken van een flavor moet op de controller node gebeuren. Dit wordt gedaan met het commando:

#openstack flavor create [naam][m1.small_small] auto [MB RAM][1024] [GB storage][20] [Aantal vCPU][1]

3.3.1 Limits

Bij het aanmaken van de flavors zijn de extra specs nog niet meegegeven. De extra specs geven limieten aan voor de instances die booten op basis van de betreffende flavor. In het proof of concept moeten er een aantal extra specs in wordne gesteld om betere QoS te bieden aan de instances.

De eerste is *cpu_shares*. Om deze extra spec aan te maken moet het volgende commando worden gebruikt:

#nova flavor-key [FlavorID/naam][m1.small_small] set quota:cpu_shares=[waarde][10]

3.4 Functietest

Om te controleren of alle Nova processen up zijn kan het volgende commando worden gebruikt:

#source /etc/admin-openrc.sh
#nova list

In het overzicht die OpenStack teruggeeft moeten alle Nova services staan en ze moeten allemaal “up” staan. De “host” van het nova-compute proces moet de compute host weergeven. Mocht het nova-compute proces niet in de lijst staan is er iets aan de hand met messagequeue.

Ook zijn er commando's om de instances te resetten, rebooten en starten. Deze commando's worden gebruikt als er een instance actief is:

nova reset-state –-active [instanceID]
nova reboot [instanceID]
nova start [instanceID]

4. Neutron network service

4.1 Controller node configureren

#mysql -u root -p
##[password van de MySQL root]
#CREATE DATABASE neutron
#GRANT ALL PRIVILEGES ON neutron. TO 'neutron'@'localhost' IDENTIFIED BY '[neutron database password]';*
#GRANT ALL PRIVILEGES ON neutron. TO 'neutron'@'%' IDENTIFIED BY '[neutron database password]';*

Voor de zekerheid moeten de admin credentials worden geladen. Dit is niet nodig als deze al geladen staan:

#source admin-openrc.sh

Nu de admin credentials weer geload zijn kunnen OpenStack commando's worden uitgevoerd. Eerst moet er een neutron user worden aangemaakt:

#openstack user create –-domain default –-password-prompt neutron
##[password die de neutron user moet krijgen]
##[password die de neutron user moet krijgen, nogmaals]

Hierna moet de role worden gemaakt voor de Neutron user:

#openstack role add –-project service –-user neutron admin

Er kan worden gecontroleerd of de Neutron user is toegevoegd:

#openstack role list --user neutron --project service

De Neutron service moet ook worden aangemaakt:

#openstack service create –-name neutron –-description “Neutron network” network

Hierna moeten de service-endpoints worden aangemaakt voor de Neutron service:

#openstack endpoint create --region RegionOne network public http://10.0.1.2:9696
#openstack endpoint create --region RegionOne network internal http://10.0.1.2:9696
#openstack endpoint create --region RegionOne network admin http://10.0.1.2:9696

Vervolgens moeten de benodigde packets voor Neutron worden geïnstalleerd:

#apt-get install neutron-server neutron-plugin-ml2 python-neutronclient

Nu moet het */etc/neutron/neutron.conf* bestand worden gewijzigd. Deze eerste variabelen staan onder de [DEFAULT] kop:

VOOR: *#rpc_backend = rabbit*
#auth_strategy = keystone

#core plugin = ml2
#service_plugins = router
#allow_overlapping_ips = False
#notify_nova_on_port_status_changes = True
#notify_nova_on_port_data_changes = True
#nova_url =

NA: *rpc_backend = rabbit*
auth_strategy = keystone
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
notify_nova_on_port_status_changes = True
notify_nova_on_port_data_changes = True
nova_url = http://[IP-adres van de controller node op het management netwerk][10.0.1.2]:8774

Deze variabelen onder de [oslo_messaging_rabbit] kop moeten worden gewijzigd:

VOOR: *#rabbit_host = localhost*
#rabbit_userid = guest
#rabbit_password = guest

NA: *rabbit_host = [IP adres van de controller op het management netwerk][10.0.1.2]*
rabbit_userid = openstack
rabbit_password = [openstack RabbitMQ password]

Onder de [database] kop moet de volgende variabele worden aangepast:

VOOR: *connection = sqlite:////var/lib/neutron/neutron.sqlite*
NA: *connection = mysql+pymysql://neutron:[password van de neutron database]@[IP-adres van de controller node op het management netwerk][10.0.1.2]/neutron*

Onder de [keystone_authtoken] kop moeten de volgende variabelen worden geplaatst en vervangen:

VOOR: *auth_uri = http://127.0.0.1:35357/v2.0/*
identity_uri = http://127.0.0.1:5000
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%
NA: *auth_uri = http://10.0.1.2:5000*
auth_url = http://10.0.1.2:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = [password van de Neutron user in Keystone]
admin_tenant_name = %SERVICE_TENANT_NAME%
admin_user = %SERVICE_USER%
admin_password = %SERVICE_PASSWORD%

Onder de [nova] moeten de volgende veranderingen plaatsvinden:

VOOR: *#auth_plugin =*
NA: *auth_url = http://10.0.1.2:35357*
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = nova
password = [password van de nova user in Keystone]

Om instellingen over de ML2 plugin in te stellen moet het */etc/neutron/plugins/ml2/ml2_conf.ini* bestand worden aangepast:

Onder de [ml2] kop moeten de volgende wijzigingen worden gemaakt:

VOOR: *type_drivers = local,flat,vlan,gre,vxlan,geneve*
tenant_network_types = local
meganism_drivers =
NA: *type_drivers = flat,vlan,gre,vxlan*
tenant_network_types = gre
meganism_drivers = openvswitch

Onder de [ml2_type_gre] kop moeten de volgende variabelen worden gewijzigd:

VOOR: *tunnel_id_ranges =*
NA: *tunnel_id_ranges = 1:1000*

Onder de [securitygroup] kop moeten de volgende variabelen worden gewijzigd:

VOOR: *#enable_security_group = True*
#enable_ipset = True
NA: *enable_security_group = True*
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

In het */etc/nova/nova.conf* bestand moeten de volgende wijzigingen plaatsvinden:
Onder de [DEFAULT] kop moeten de volgende variabelen worden toegevoegd:

VOOR:
NA: *network_api_class = nova.network.neutronv2.api.API*
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver

Er moet een nieuwe kop aan worden gemaakt, nl: [neutron]. Hier moeten de volgende variabelen in worden gezet:

VOOR:
NA: [neutron]
url = http://[IP-adres van de controller node op het management netwerk][10.0.1.2]:9696
auth_strategy = keystone
auth_url = http://[IP-adres van de controller node op het management netwerk][10.0.1.2]:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
region_name = RegionOne
project_name = service
username = neutron
password = [password van de neutron user binnen Keystone]

4.2 Network node installeren

Eerst moet de nieuwe OpenStack Liberty repository worden toegevoegd:

#apt-get install software-properties-common
#add-apt-repository cloud-archive:liberty
#apt-get update
#apt-get dist-upgrade

In het bestand */etc/sysctl.conf* moeten een aantal variabelen worden veranderd. Sysctl.conf wordt gebruikt voor netwerkfunctionaliteit/netwerkbeveiliging:

VOOR: *#net.ipv4.ip_forward=1*
#net.ipv4.conf.default.rp_filter=1
#net.ipv4.conf.all.rp_filter=1
NA: *net.ipv4.ip_forward=1*
net.ipv4.conf.default.rp_filter=0
net.ipv4.conf.all.rp_filter=0

Vervolgens moeten de hiervoor aangebrachte wijzigingen worden geladen:

sysctl -p

De packets voor de network node moeten worden geïnstalleerd:

#apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent neutron-l3-agent neutron-dhcp-agent neutron-metadata-agent

4.2.1 /etc/neutron/neutron.conf bestand

Nu moet het /etc/neutron/neutron.conf bestand worden gewijzigd. Eerst moet de database connectie worden uitgeschakeld omdat de network node geen directe verbinding heeft met de database:

VOOR: *connection = sqlite:////var/lib/neutron/neutron.sqlite*
NA: *#connection = sqlite:////var/lib/neutron/neutron.sqlite*

Onder de [DEFAULT] kop moet het rpc_backend worden geactiveerd:

VOOR: *#rpc_backend = rabbit*
#core_plugin = ml2
#service_plugins =
#allow_overlapping_ips = False
#auth_strategy = keystone
NA: *rpc_backend = rabbit*
core_plugin = ml2
service_plugins = router
allow_overlapping_ips = True
auth_strategy = keystone

Doordat het rpc_backend op rabbit is gezet moeten de variabelen onder de [oslo_messaging_rabbit] worden aangepast:

VOOR: *#rabbit_host = localhost*
#rabbit_userid = guest
#rabbit_password = guest
NA: *rabbit_host = 10.0.1.2*
rabbit_userid = openstack
rabbit_password = [Openstack user voor RabbitMQ]

Nu moeten de variabelen onder de [keystone_authtoken] kop worden aangepast. De *identity_uri* variabele moet worden weggecomment. In plaats van de *identity_uri* variabele wordt de *auth_url* variabele worden gebruikt.

VOOR: *auth_uri = http://127.0.0.1:35357/v2*
identity_uri = http://127.0.0.1:5000
NA: *auth_uri = http://10.0.1.2:5000*
#identity_uri = http://127.0.0.1:5000
auth_url = http://10.0.1.2:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = [password van de neutron user in de Keystone service]

4.2.2 /etc/neutron/plugins/ml2/ml2_conf.ini bestand

Hierna moet het */etc/neutron/plugins/ml2/ml2_conf.ini* bestand worden gewijzigd. Onder de [ml2] kop moeten de volgende variabelen worden veranderd:

VOOR: *type_drivers = local,flat,vlan,gre,vxlan,geneve*
tenant_network_types = local
meganism_drivers =
NA: *type_drivers = flat,vlan,gre,vxlan*
tenant_network_types = gre
meganism_drivers = openvswitch

Onder de [ml2_type_flat] kop moeten de volgende variabelen worden gewijzigd:

VOOR: #flat_networks =
NA: flat_networks = external

Onder de [ml2_type_gre] kop moeten de volgende variabelen worden gewijzigd:

VOOR: *tunnel_id_ranges =*
NA: *tunnel_id_ranges = 1:1000*

Onder de [securitygroup] kop moeten de volgende variabelen worden gewijzigd:

VOOR: *#enable_security_group = True*
#enable_ipset = True
NA: *enable_security_group = True*
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

Tenslotte moeten de [ovs] en [agent] koppen met de volgende variabelen worden toegevoegd:

```
[ovs]
local_ip = [IP adres van het tunnel netwerk aan de network node kant][10.0.2.2]
bridge_mappings = external:br-ex

[agent]
tunnel_types = gre
```

4.2.3 /etc/neutron/l3_agent.ini bestand

Onder de [DEFAULT] kop moeten de volgende variabelen worden gewijzigd:

VOOR: *#interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver*
#external_network_bridge = br_ex
#router_delete_namespaces = True
NA: *interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver*
external_network_bridge =
router_delete_namespaces = True

4.2.4 /etc/neutron/dhcp_agent.ini bestand

Onder de [DEFAULT] kop moeten de volgende variabelen worden gewijzigd:

VOOR: *#interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver*
#dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
#dhcp_delete_namespaces = True
NA: *interface_driver = neutron.agent.linux.interface.OVSInterfaceDriver*
dhcp_driver = neutron.agent.linux.dhcp.Dnsmasq
dhcp_delete_namespaces = True

4.2.5 /etc/neutron/metadata_agent.ini

Voordat het bestand word gewijzigd moet eerst een nieuw password worden aangemaakt voor de metadata proxy. Een password moet worden aangemaakt volgens hoofdstuk 1.3.

VOOR: *auth_url = http://localhost:5000/v2.0*
auth_region = RegionOne
 # nova_metadata_ip = 127.0.0.1
#metadata_proxy_shared_secret =
NA: *auth_uri = http://10.0.1.2:5000*
auth_url = http://10.0.1.2:35357
auth_region = RegionOne
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = [password van de neutron user in de Keystone service]
nova_metadata_ip = [IP adres van de controller node in het management netwerk][10.0.1.2]
metadata_proxy_shared_secret = [password voor de metadata proxy]

4.2.6 /etc/nova/nova.conf bestand op de controller node

Onder de [neutron] kop moeten de volgende variabelen worden toegevoegd:

```
service_metadata_proxy = True
metadata_proxy_shared_secret = [password voor de metadata proxy]
```

4.2.7 OpenvSwitch service

Eerst moet voor de zekerheid de service openvswitch worden herstart:

#service openvswitch-switch restart

Vervolgens moet er een externe bridge worden aangemaakt voor de network node:

#ovs-vsctl add_br br-ex

Hierna moet de externe interface port van de server worden aangesloten op de virtuele switch:

#ovs-vsctl add-port br-ex [Netwerkinterface van de network node waar het i3D netwerk op aangesloten zit][eth2]

4.2.8 Services restarten

Hierna moeten de service worden herstart:

#service neutron-plugin-openvswitch-agent restart
#service neutron-l3-agent restart
#service neutron-dhcp-agent restart

#service neutron-metadata-agent restart

4.2.9 functietest

Met het volgende commando kan een lijst met neutron services worden opgevraagd:

#neutron agent-list

De services die op de network node draaien moeten op dit moment reactie terug geven. Er moet dus een lijst geprint worden met vier services er in, nl: OpenvSwitch, Metadata agent, DHCP agent en L3 agent.

4.3 Compute node configureren

In het bestand /etc/sysctl.conf moeten een aantal variabelen worden veranderd, tevens moeten de laatste twee variabelen worden toegevoegd:

VOOR: *net.ipv4.conf.all.rp_filter=1*
net.ipv4.conf.default.rp_filter=1
NA: *net.ipv4.conf.all.rp_filter=0*
net.ipv4.conf.default.rp_filter=0
net.bridge.bridge-nf-call-iptables=1
net.bridge.bridge-nf-call-ip6tables=1

Om de wijzigingen in het /etc/sysctl.conf van kracht te laten worden moet het volgende commando worden uitgevoerd:

#sysctl -p

Op de compute node moeten de volgende packages worden geïnstalleerd:

apt-get install neutron-plugin-ml2 neutron-plugin-openvswitch-agent

4.3.1 /etc/neutron/neutron.conf bestand

Onder de [DEFAULT] kop moeten de volgende wijzigingen worden aangebracht:

VOOR: *#rpc_backend = rabbit*
#core_plugin = ml2
#service_plugins =
#Allow_overlapping_ips = False
#auth_strategy = keystone
NA: *rcp_backend = rabbit*
core_plugin = ml2
service_plugins = router
Allow_overlapping_ips = True
auth_strategy = keystone

De connection variabele onder de [database] kop moet worden weggecomment omdat ook de compute node de neutron data niet direct naar de database schrijft:

VOOR: *connection = sqlite:////var/lib/neutron/neutron.sqlite*
NA: #connection = sqlite:////var/lib/neutron/neutron.sqlite

Doordat het rpc_backend op rabbit is gezet moeten de variabelen onder de [oslo_messaging_rabbit] worden aangepast:

VOOR: *#rabbit_host = localhost*
#rabbit_userid = guest
#rabbit_password = guest
NA: *rabbit_host = [IP adres van de controller node management netwerk][10.0.1.2]*
rabbit_userid = openstack
rabbit_password = [Openstack user voor RabbitMQ]

Nu moeten de variabelen onder de [keystone_authtoken] kop worden aangepast. De *identity_uri* variabele moet worden weggecomment. In plaats van de *identity_uri* variabele wordt de *auth_url* variabele worden gebruikt.

VOOR: *auth_uri = http://127.0.0.1:35357/v2*
identity_uri = http://127.0.0.1:5000
NA: *auth_uri = http://[IP adres van de controller node management netwerk][10.0.1.2]:5000*
#identity_uri = http://127.0.0.1:5000
auth_url = http://[IP adres van de controller node management netwerk][10.0.1.2]:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = neutron
password = [password van de neutron user in de Keystone service]

4.3.2 /etc/neutron/plugins/ml2/ml2_conf.ini bestand

Hierna moet het */etc/neutron/plugins/ml2/ml2_conf.ini* bestand worden gewijzigd. Onder de [ml2] kop moeten de volgende variabelen worden veranderd:

VOOR: *type_drivers = local,flat,vlan,gre,vxlan,geneve*
tenant_network_types = local
meganism_drivers =
NA: *type_drivers = flat,vlan,gre,vxlan*
tenant_network_types = gre
meganism_drivers = openvswitch

Onder de [ml2_type_gre] kop moeten de volgende variabelen worden gewijzigd:

VOOR: *tunnel_id_ranges =*
NA: *tunnel_id_ranges = 1:1000*

Onder de [securitygroup] kop moeten de volgende variabelen worden gewijzigd:

VOOR: *#enable_security_group = True*

#enable_ipset = True

NA: *enable_security_group = True*
enable_ipset = True
firewall_driver = neutron.agent.linux.iptables_firewall.OVSHybridIptablesFirewallDriver

Tenslotte moeten de [ovs] en [agent] koppen met de volgende variabelen worden toegevoegd:

```
[ovs]
local_ip = [IP-adres van de VM tunnel aan de compute kant][10.0.2.3]

[agent]
tunnel_types = gre
```

4.3.3 /etc/nova/nova.conf bestand

Onder de [DEFAULT] kop moeten de volgende variabelen worden toegevoegd:

```
network_api_class = nova.network.neutronv2.api.API
security_group_api = neutron
linuxnet_interface_driver = nova.network.linux_net.LinuxOVSInterfaceDriver
firewall_driver = nova.virt.firewall.NoopFirewallDriver
```

Er moet een [neutron] kop worden gemaakt. Hieronder moeten de volgende variabelen worden aangemaakt:

```
url = http://[IP-adres van de controller node op het management netwerk]
[10.0.1.2]:9696
auth_strategy = keystone
admin_auth_url = http://[IP-adres van de controller node op het management netwerk]
[10.0.1.2]:35357
admin_tenant_name = service
admin_username = neutron
admin_password = [password van de neutron user in de Keystone service]
```

4.3.4 Functietest

De functionaliteit van de geconfigureerde compute node kan worden achterhaald met het volgende commando:

#neutron agent-list

In de uitgeprinte lijst moeten nu de vijf geconfigureerde services staan, nl: Metadata agent op de network node, DHCP agent op de network node, Open vSwitch op de compute node, Open vSwitch op de network node en L3 agent op de network node.

4.4 Extern netwerk

De externe bridge die is gemaakt moet nog een koppeling binnen OpenStack krijgen. Hier moet een virtueel netwerk worden aangemaakt. Als dit virtuele netwerk is aangemaakt ziet het er als volgt uit: virt_ext_network → virt_ext_bridge → extern netwerk.
Om dit vertuele netwerk aan te maken moet het volgende commando worden gebruikt:

neutron net-create ext-net --router:external --provider:physical_network external --provider:network_type flat

Vervolgens moet dit virtuele netwerk worden ingericht met een subnet. Hier wordt ook de range van floating IP's aangegeven:

neutron subnet-create ext-net [Netwerkadres van het i3D netwerk in CIDR notatie][X.X.X.X/26] --name ext-subnet --allocation-pool start=[start van de floating IP range][X.X.X.X],end=[einde van de floating IP range] [X.X.X.X] --disable-dhcp --gateway [gateway op het i3D netwerk][X.X.X.X]

4.5 Virtueel tenant netwerk

Om de test user (in de test tenant) naar het i3D netwerk te kunnen laten communiceren moet er nog een virtueel netwerk worden aangemaakt. Dit virtuele netwerk is bedoeld voor de tenant “test” die eerder binnen Keystone is aangemaakt.

Eerst moeten de credentials van de test-user worden geladen i.p.v. de admin user:

source [pad naar test-openrc.sh][/etc/test-openrc.sh]

Hierna kan de test user het netwerk aanmaken:

neutron net-create test-net

Ook de test user moet het virtuele netwerk een subnet geven:

neutron subnet-create test-net [Netwerkadres van het virtuele netwerk in CIDR notatie][192.168.1.0/24] --name test-subnet --dns-nameserver [Adres van DNS server][8.8.4.4] --gateway [Gateway op het tenant netwerk] [192.168.1.1]

Hierna moet het interne virtuele netwerk van de test user/tenant worden gekoppeld aan het i3D netwerk. Dit wordt gedaan met een virtuele router.
Eerst moet de router worden aangemaakt:

neutron router-create test-router

Hierna moet de router worden gekoppeld:

neutron router-interface-add test-router test-subnet

Hierna moet de router nog als gateway worden ingesteld:

neutron router-gateway-set test-router ext-net

4.5.1 Neutron-QoS

In Neutron zijn een aantal QoS-functies die aangemaakt moeten worden. Het gaat dan om elke virtuele port waar klant instances op communiceren.

#neutron qos-policy-create throttled
#neutron qos-bandwidth-limit-rule-create throttled –-max-kbps 10000 –-max-burst-kbps 10000

Om een port te updaten met de nieuwe QoS rules moet het volgende commando worden inegegeven:

neutron port-update [port ID] –-no-qos-policy

4.5.2 Functietest

Nu kan worden bevestigd of het interne virtuele netwerk te benaderen is door vanaf een fysieke machine het floating IP-adres te pingen. In deze handleiding is gebruik gemaakt van de X.X.X.X ~ X.X.X.X range. Dit betekend dat OpenStack de virtuele router op het test tenant netwerk het eerst mogelijke IP-adres geeft. Vanaf een fysieke machine moet dit IP-adres dus te pingen zijn.

Ook is er op dit moment te controleren of er een image kan booten binnen OpenStack. Om deze test uit te voeren moet er eerst een keypair worden aangemaakt:

#source [pad naar test-openrc.sh][/etc/test-openrc.sh]
#nova keypair-add test-key
#nova keypair-list

De uitvoer van het laatste bovenstaande commando geeft een lijst met het keypair. Naast deze lijst moeten ook andere lijsten worden opgevraagd om een instance te kunnen booten:

#nova flavor-list
#nova image-list
#neutron net-list
#nova secgroup-list

In de uitvoer van de bovenstaande commando's staan variabelen die moeten worden gebruikt bij het aanmaken van een instance. De instance wordt aangemaakt met het volgende commando:

#nova boot --flavor m1.tiny --image [image naam uit de nova image-list][cirros] --nic net-id=[ID van het test netwerk] --security-group default --key-name [Naam van het keypair][test-key] [naam die de instance moet krijgen][test-instance1]

Om de instance hierna nog een floating-IP toe te wijzen voor i3D netwerk toegang moeten de volgende commando's worden uitgevoerd:

#neutron floatingip-create public

Het floating-IP-adres dat is teruggestuurd vanuit Neutron moet hierna worden gekoppeld met de instance:

#nova floating-ip-associate [instance naam] [floating-IP]

Nu moet er een port worden opgezet in OpenStack. Het is port 22 om SSH te gebruiken.

nova secgroup-add-rule [Naam van de security group][default] tcp 22 22 0.0.0.0/0

5. Horizon

In dit hoofdstuk wordt het dashboard genaamd “horizon” geïnstalleerd.

Eerst moeten de packages worden geïnstalleerd:

#apt-get install openstack-dashboard

5.1 /etc/openstack-dashboard/local_settings.py bestand

In dit bestand moet eerst de volgende code worden vervangen:

VOOR: *OPENSTACK_HOST = “”*
NA: *OPENSTACK_HOST = “[IP-adres van de controller node op het management netwerk][10.0.1.2]”*

Om alle hosts toegang te geven naar Horizon moet de volgende code worden vervangen:

VOOR: *ALLOWED_HOSTS = '*'*
NA: *ALLOWED_HOSTS = ['*',]*

Hierna moet worden gecontroleerd of de volgende regels code kloppen:

```
CACHES = {
    'default': {
         'BACKEND': 'django.core.cache.backends.memcached.MemcachedCache',
         'LOCATION': '127.0.0.1:11211',
    }
}
```

Ook moet er worden ingesteld wat de default role is voor gebruikers die aangemaakt worden via het dashboardL

VOOR: *OPENSTACK_KEYSTONE_DEFAULT_ROLE = “”*
NA: *OPENSTACK_KEYSTONE_DEFAULT_ROLE = “user”*

Het multi-domain model moet worden ondersteund:

VOOR: *#OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = False*
NA: *OPENSTACK_KEYSTONE_MULTIDOMAIN_SUPPORT = True*

Hierna moeten de API-versies correct worden ingesteld:

```
OPENSTACK_API_VERSIONS = {
    "identity": 3,
    "volume": 2,
}
```

Hierna moet de Apache server worden herstart:

#service apache2 restart

6. Cinder block storage

Op de Cinder block storage node moet eerst een Cinder database worden aangemaakt voor de storage service:

#mysql -u root -p
##[password van de MySQL root]
#CREATE DATABASE cinder
#GRANT ALL PRIVILEGES ON cinder. TO 'cinder'@'localhost' IDENTIFIED BY '[Cinder database password]';*
#GRANT ALL PRIVILEGES ON cinder. TO 'cinder'@'%' IDENTIFIED BY '[Cinder database password]';*

Voor de zekerheid moeten de admin credentials worden geladen. Dit is niet nodig als deze al geladen staan:

#source admin-openrc.sh

Nu de admin credentials weer geload zijn kunnen OpenStack commando's worden uitgevoerd. Eerst moet er een Cinder user worden aangemaakt:

#openstack user create –-domain default –-password-prompt cinder
##[password die de Cinder user moet krijgen]
##[password die de Cinder user moet krijgen, nogmaals]

Hierna moet de role worden gemaakt voor de Cinder user:

#openstack role add –-project service –-user cinder admin

Er kan worden gecontroleerd of de Cinder user is toegevoegd:

#openstack role list --user cinder --project service

De Cinder service moet ook worden aangemaakt:

#openstack service create –-name cinder –-description “OpenStack Block Storage” volume
#openstack service create –-name cinderv2 –-description “OpenStack Block Storage” volumev2

Hierna moeten de endpoints worden gecreëerd:

#openstack endpoint create --region RegionOne volume internal http://10.0.1.2:8776/v1/%\(tenant_id\)s
#openstack endpoint create --region RegionOne volume iadmin http://10.0.1.2:8776/v1/%\(tenant_id\)s
#openstack endpoint create --region RegionOne volume public http://10.0.1.2:8776/v1/%\(tenant_id\)s

#openstack endpoint create --region RegionOne volumev2 internal http://10.0.1.2:8776/v2/%\(tenant_id\)s
#openstack endpoint create --region RegionOne volumev2 iadmin http://10.0.1.2:8776/v2/%\(tenant_id\)s
#openstack endpoint create --region RegionOne volumev2 public http://10.0.1.2:8776/v2/%\(tenant_id\)s

Nu moeten de packages worden geïnstalleerd:

#apt-get install cinder-api cinder-scheduler python-cinderclient

6.1 /etc/cinder/cinder.conf op de controller node

Na het installeren van de packages kan het */etc/cinder/cinder.conf* bestand worden aangepast. Alle onderstaande waardes moeten aan het bestand worden toegevoegd:

```
[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_confg = /etc/cinder/api-paste.ini
iscsi_helper = tgtadm
volume_name_template = volume-%s
volume_group = cinder-volumes
verbose = True
auth_strategy = keystone
state_path = /var/lib/cinder
lock_path = /var/lock/cinder
volumes_dir = /var/lib/cinder/volumes
rpc_backend = rabbit
my_ip = [IP-adres van de controller node op het management network][10.0.1.2]

[database]
connection = mysql+pymysql://cinder:[Password van de Cinder user in de Cinder
database]@[IP-adres van de controller node op het management network]/cinder

[oslo_messaging_rabbit]
rabbit_host = [IP-adres van de controller node op het management network]
rabbit_userid = openstack
rabbit_password = [Password van de RabbitMQ openstack user]

[keystone_authtoken]
auth_uri = http://[IP-adres van de controller node op het management network]:5000
auth_url = http://[IP-adres van de controller node op het management
network]:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = cinder
password = [Password van de Cinder user in Keystone]

[oslo_concurrency]
lock_path = /var/lib/cinder/tmp
```

6.2 /etc/nova/nova.conf op de controller node

In het nova.conf bestand hoeft alleen aagegeven te worden welke region Cinder gebruikt in de opstelling. Dit wordt gedaan door de volgende regels toe te voegen aan dit bestand:

```
[cinder]
os_region_name = RegionOne
```

Hierna moeten de betrokken services op de controller node worden herstart:

#service nova-api restart
#service cinder-scheduler restart
#service cinder-api restart

6.3 Partitionering block storage node

In de proefopstelling is er gebruik gemaakt van een hardwarematige RAID10. Nu worden de vier schijven in de DL120 g6 gezien als één schijf. Dit betekend nu wel dat er voor Cinder geen lege schijven op te zetten zijn met LVM. Deze LVM partitie moet dan op “dezelfde schijf (RAID10)” als het OS worden opgezet. De partitie is opgezet volgens het overzicht in hoofdstuk 1.4.1.

Voor de proefopstelling is er met de i3D deploy feature voor klanten een partitionering opgezet. Echter kan er met de feature niet gekozen worden om een gedeelte van de ruimte leeg te laten zonder mountpoint en filesystem.
Nu kan er een installatie worden gedaan van een live-USB-stick of CD en hierin de partities goed zetten. Ook kan er wel gedeployed worden met de i3D deployment feature. Dan moet hierna wel gewerkt worden met parted in Ubuntu rescue mode.

Voor de proefopstelling is gekozen om toch te deployen. Na het deployen bleek dat de overige ruimte toegewezen was aan de laatste (swap) partitie in de lijst.
Hierna is geboot in Ubuntu recovery mode. In deze modus is de swap partitie resized naar 8192MB. De overige ruimte is aangemaakt als logische partitie zonder filesystem. Deze acties zijn gedaan met de volgende commando's:

Eerst kijken hoe de partities staan met:

parted -l

Nu de swap partitie waar alle ruimte aan is toegewezen verwijderen of resizen:

parted
rm [nummer van de partitie]

of

parted
resize [nummer van de partitie] [startpunt in KB, MB of GB] [Eindpunt in KB, MB of GB]

Hierna moet er opnieuw worden geboot met: *quit*, en dan *reboot*.
Hierna moet opnieuw worden geboot in recovery modus. Als de swap partitie is verwijderd moet er een nieuw (kleinere) worden aangemaakt. Als er is geresized kan er direct een logisch volume worden aangemaakt:
Om een nieuwe swap partitie aan te maken worden de volgende commando's uitgevoerd:

parted
mkpartfs primary linux-swap [startpunt in KB, MB of GB] [Eindpunt in KB, MB of GB]

Hierna moet er opnieuw worden geboot met: quit, en dan reboot.

Nu kan de lege ruimte worden gepartitioneerd:

parted
mkpart logical [startpunt in KB, MB of GB] [Eindpunt in KB, MB of GB]

6.4 block storage node installeren

Eerst moet de OpenStack repository worden toegewezen:

#apt-get install software-properties-common
#add-apt-repository cloud-archive:liberty
#apt-get update
#apt-get dist-upgrade

Eerst moet LVM worden geïnstalleerd en geconfigureerd om het te gebruiken met Cinder:

#apt-get install lvm2

Hierna moet er een fysiek volume worden aangemaakt:

#pvcreate [mount van de LVM space][/dev/sda5]

Ook moet de volume-group worden aangemaakt voor LVM:

#vgcreate [naam van de volume-group][cinder-volumes] [Mount van de LVM space][/dev/sda5]

Met deze filter wordt de toegang naar SDA opengezet. Naar de resterende drives en partities wordt geen toegang verleent. Hier is a accept en r is reject. Dit moet worden aangepast in /etc/lvm/lvm.conf.

VOOR: *filter = ["a/.*/"]*
NA: *filter = ["a/sda/", "r/.*/"]*

Hierna moeten de nodige packages worden geïnstalleerd:

#apt-get install cinder-volume python-mysqldb

6.4.1 /etc/cinder/cinder.conf bestand

In het /etc/cinder/cinder.conf bestand moeten de volgende regels code worden toegevoegd:

```
[DEFAULT]
rootwrap_config = /etc/cinder/rootwrap.conf
api_paste_confg = /etc/cinder/api-paste.ini
state_path = /var/lib/cinder
rpc_backend = rabbit
auth_strategy = keystone
my_ip = [IP-adres van de storage node op het management netwerk][10.0.1.3]
enabled_backends = lvm
glance_host = [IP-adres van de Controller node op het management netwerk]
```

```
[10.0.1.2]
enable_v1_api = True
enable_v2_api = True
osapi_volume_listen = 0.0.0.0
osapi_volume_listen_port = 8776
glance_host = [IP-adres van de Controller node op het management netwerk]
[10.0.1.2]
glance_port = 9292
notification_driver = cinder.openstack.common.notifier.rpc_notifier
scheduler_driver = cinder.scheduler.filter_scheduler.FilterScheduler

[database]
connection = mysql+pymysql://cinder:[Password van de cinder user in de
database]@[IP-adres van de controller node op het management netwerk]
[10.0.1.2]/cinder

[oslo_messaging_rabbit]
rabbit_host = [IP-adres van de Controller node op het management netwerk]
[10.0.1.2]
rabbit_userid = openstack
rabbit_password = [Password van de Rabbit MQ OpenStack user]

[keystone_authtoken]
auth_uri = http://[IP-adres van de Controller node op het management netwerk]
[10.0.1.2]:5000
auth_url = http://[IP-adres van de Controller node op het management netwerk]
[10.0.1.2]:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = cinder
password = [Password van de Cinder user in Keystone]

[lvm]
volume_driver = cinder.volume.drivers.lvm.LVMVolumeDriver
volume_group = cinder-volumes
iscsi_protocol = iscsi
iscsi_helper = tgtadm

[oslo_concurrency]
lock_path = /var/lib/cinder/tmp
```

Hierna moeten de services worden herstart:

```
#service tgt restart
#service cinder-volume restart
```

6.5 Cinder QoS

In Cinder moet er ook een vorm van QoS worden opgenomen. Dit kan worden gedaan door het uitvoeren van de volgende commando's:

#cinder qos-create throttled read_bytes_sec=34603008 write_bytes_sec=34603008
#cinder type-create throttled
#cinder qos-associate [QoS_ID] [CInder_type_ID]

Hierna moet bij het aanmaken van elk Cinder-volume ook de *volume-type* parameter worden meegegeven. Het commando is als volgt:

#cinder create --display-name [volumenaam] --volume-type throttled [grootte in GB][1]

6.6 Functietest

Eerst moet worden getest of de Cinder-services online zijn (op de controller node):

#source [pad naar admin-openrc.sh][/etc/admin-openrc.sh]
#cinder service-list

De uitkomst van dit commando moet een tabel zijn met de Cinder-scheduler op de controller node en Cinder-volume op de block storage node ([naam van de block storage node][server1987]@lvm).

Om te testen of de services werken moeten eerst de credentials van de test users worden geladen in de environment variablen. De reden hier van is dat de admin user en de test user in een andere tenant zitten. Ze kunnen elkaars block volumes dus niet “zien”:

#source [pad naar test-openrc.sh][/etc/test-openrc.sh]

6.6.1 Cinder-volume aanmaken en koppelen

Hierna moet het volume worden aangemaakt op de block storage node:

#cinder create –-display-name [naam van het volume][testvolume1] [grootte van het volume in GB][1]

De uitkomst van dit commando is een tabel met informatie over het volume. Hierna moet het volume ook in de Cinder-listing staan:

#cinder list

Hierna moet het block volume gekoppeld worden met de instance:

#nova volume-attach test-instance1 d48db218-66be-49ce-b1ca-df95dcb6982a

Hierna kan er worden gecontroleerd of het volume is gekoppeld met:

#nova volume-list

Als er nu wordt verbonden naar de instance met SSH dan moet het block volume te zien zijn. Voor de test is er een Cirros instance actief. Op deze instance kan worden gecontroleerd of het volume correct is gekoppeld met

fdisk:

#sudo fdisk -l
#sudo fdisk [disk][vdb]

In het fdisk commandprompt moeten de volgende commando's worden ingegeven:

#n
#p
#2048
#2097151
#w

In de normale command prompt moet dan worden ingegeven:

#mkfs.ext3 /dev/[disk][vdb1]

Hierna kan de disk worden gemount:

#mkdir /media/disk1
#mount /dev/vdb1 /media/disk1

Met het volgende commando kan worden gecontroleerd of het volume goed is gepartitioneerd, geformatteerd en gemount:

#lsblk

6.6.2 Een instance booten van een Cinder volume

Op dit moment kan er een instance worden geboot vanaf een Cinder-volume in plaats van de lokale Nova storage.

Er moet eerst een image aanwezig zijn in de Glance-store. Dit wordt voorgedaan in subhoofdstuk 3.2.6. Vervolgens kan met deze image een bootable volume gemaakt worden:

#openstack volume create --image [imageID uit de Glance-store] --size [grootte in GB] --description "[omschrijving van het volume]" [naam van het volume]

Eerst moet er een nieuw keypait aan worden gemaakt met:

#nova keypair-add [Keypair naam]

Hierna moeten een aantal lists worden opgevraagd om de nodige informatie te weten te komen:

#neutron net-list
#nova flavor-list
#nova keypair-list

#openstack server create --flavor [Flavor-type][m1.small_small] --key-name [Key-naam][bootable] --volume

[VolumeID van het net aangemaakt bootable volume] --nic net-id=[ID van het virtuele interne netwerk] --security-group default [Naam van de instance][bootable]

Na dit commando moet het volume als “bootable = true” staan en de instance moet aanwezig staan in de nova listing:

#cinder list
#nova list

6.6.3 Het RAID-array monitoren op afstand

Bij het opzetten van de Cinder is er gebruik gemaakt van een RAID-array in RAID 10 opstelling. Als de opstelling staat dan is het RAID-array ook actief. Om fouten met het RAID-array te achterhalen en te monitoren moet een stukje software van HP zelf geïnstalleerd worden op de block storage server.
Het stukje software waar het hier om gaat is “hpssacli”, dit programma moet eerst worden opgehaald van de servers van HP.
Nadat het programma is gedownload moet het worden gekopieerd naar de block storage node. Dit kan met SCP of met een memory stick, O.I.D. Als het stuk software op de block storage node staat moet het worden geïnstalleerd met:

#dpkg -i hpssacli-2.0-16.0_amd64.deb

Als de software succesvol is geïnstalleerd kan de status van het RAID-array worden opgevraagd met het commando:

#hpssacli
#ctrl all show config

7. Rally OpenStack benchmarking

Om de omgeving te kunnen testen op performance wordt Rally gebruikt. Met Rally kan een OpenStack omgeving worden getest met een benchmark.

7.1 Installatie Rally

De installatie van Rally gebeurt op een losstaande node: de “utility node”. Deze node moet wel aangesloten zijn op het interne management netwerk.

Eerst moeten de nodige packages worden geïnstalleerd:

#apt-get install libssl-dev libffi-dev python-dev libxml2-dev libxslt1-dev libpq-dev git python-pip

Hierna kan het installatiescript worden aangeroepen:

#wget -q -O- https://raw.githubusercontent.com/openstack/rally/master/install_rally.sh | bash

Vervolgens moet er ook op deze node een *admin-openrc.sh* bestand worden aangemaakt. In dit bestand staat de informatie omtrent de admin user. Met dit bestand kan Rally de OpenStack opstelling benaderen als admin user.
Nu moet Rally nog worden vertelt dat het deze crentials mag gebruiken:

#rally deployment create --fromenv –name=[naam van de deployment][Openstack test1]

Met deze credentials heeft Rally een deployment gemaakt. Het deployment waar Rally het over heeft is een soort plattegrond van OpenStack.
Nu kan er worden gecontroleerd of Rally allee services “ziet”. De output van dit commando moet een lijst zijn met alle services in OpenStack. Elke service zou “available” moeten aangeven. Dit kan met het commando:

#rally deployment check

De rally installatie kan worden gemanaged met het *rally.conf* bestand in de */etc/* map. De verdere bestanden die Rally gebruikt staan in *~/samples/*. Het volledige pad naar de rally files mag zelf worden gedefinieerd. Standaard is dit pad: */usr/share/rally/samples/*.

8. Ceilometer

Eerst moet de dedicated database worden aangemaakt voor Ceilometer. Het betreft een noSQL-database:

#mongo --host controller --eval '
#db = db.getSiblingDB("ceilometer");
#db.addUser({user: "ceilometer",
#pwd: "[Password voor de ceilometer user",
#roles: ["readWrite", "dbAdmin"]})'

Vervolgens moet de service user worden aangemaakt:

#openstack user create --domain default --password-prompt ceilometer
##Password voor ceilometer ingeven
##Password voor Ceilometer nogmaals ingeven

Hierna moeten de role, service entity en de endpoints worden aangemaakt:

#openstack role add --project service --user ceilometer admin

#openstack service create --name ceilometer --description "Telemetry" metering

#openstack endpoint create --region RegionOne metering public http://10.0.1.2:8777
#openstack endpoint create --region RegionOne metering internal http://10.0.1.2:8777
#openstack endpoint create --region RegionOne metering admin http://10.0.1.2:8777

Tenslotte moeten de nodige pakketten worden geïnstalleerd:

#apt-get install ceilometer-api ceilometer-collector ceilometer-agent-central ceilometer-agent-notification ceilometer-alarm-evaluator ceilometer-alarm-notifier python-ceilometerclient

8.1 /etc/ceilometer/ceilometer.conf bestand

Onder de [database] kop:

VOOR: #connection = <NONE>
NA: connection = mongodb://ceilometer:[Ceilometer database password]@[IP-adres van de controller node op het manegement netwerk][10.0.1.2]:27017/ceilometer

onder de [DEFAULT] kop

VOOR: #rpc_backend = rabbit
NA: rpc_backend = rabbit
auth_strategy = keystone

Onder de [oslo_messaging_rabbit] kop

[oslo_messaging_rabbit]
rabbit_host = [IP-adres van de controller node op het management network]

rabbit_userid = openstack
rabbit_password = [Password van de RabbitMQ openstack user]

Onder de [keystone_authtoken] kop

auth_uri = http://[IP-adres van de controller node op het manegement netwerk][10.0.1.2]:5000
auth_url = http://[IP-adres van de controller node op het manegement netwerk][10.0.1.2]:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = ceilometer
password = [ceilometer password]

Nieuw stukje code die moet worden aangemaakt onderaan het config bestand.

```
[service_credentials]
os_auth_url = http://[IP-adres van de controller node in het management netwerk]
[10.0.1.2]:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = [ceilometer password]
os_endpoint_type = internalURL
os_region_name = RegionOne
```

8.2 meters aanmaken

8.2.1 /etc/glance/glance-api.conf en /etc/glance/glance-registry.conf

onder de [DEFAULT] kop

VOOR: #rpc_backend = rabbit
NA: rpc_backend = rabbit
auth_strategy = keystone

Onder de [oslo_messaging_rabbit] kop

[oslo_messaging_rabbit]
rabbit_host = [IP-adres van de controller node op het management network]
rabbit_userid = openstack
rabbit_password = [Password van de RabbitMQ openstack user]

8.2.2 Op de compute node(s)

Op de compute nodes moeten meters worden aangemaakt. Eerst moet het nodige pakket worden gedownload:

#apt-get install ceilometer-agent-compute

/etc/ceilometer/ceilometer.conf bestand

onder de [DEFAULT] kop

VOOR: #rpc_backend = rabbit
NA: rpc_backend = rabbit
auth_strategy = keystone

Onder de [oslo_messaging_rabbit] kop

[oslo_messaging_rabbit]
rabbit_host = [IP-adres van de controller node op het management network]
rabbit_userid = openstack
rabbit_password = [Password van de RabbitMQ openstack user]
Onder de [keystone_authtoken] kop

auth_uri = http://[IP-adres van de controller node op het manegement netwerk][10.0.1.2]:5000
auth_url = http://[IP-adres van de controller node op het manegement netwerk][10.0.1.2]:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = ceilometer
password = [ceilometer password]

Er moet een nieuwe kop *[service credentials]* worden aangemaakt.

```
[service_credentials]
os_auth_url = http://[IP-adres van de controller node in het management netwerk]
[10.0.1.2]:5000/v2.0
os_username = ceilometer
os_tenant_name = service
os_password = [ceilometer password]
os_endpoint_type = internalURL
os_region_name = RegionOne
```

9. Swift object storage

Bij het aanmaken van de Swift opstelling moeten eerst de juiste instellingen aan Keystone worden meegegeven. Eerst moet er een service user worden aangemaakt en de admin role worden toegewezen:

#openstack user create --domain default --project service --password [Password voor de swift user in Keystone] swift
#openstack role add --project service --user swift admin

Hierna moet de Swift service worden aangemaakt:

openstack service create --name swift --description "OpenStack Object Storage" object-store

Vervolgens moeten de endpoints voor Swift worden aangegegeven in Keystone:

#openstack endpoint create --region RegionOne object-store public http://[IP-adres van de Swift proxy server]:8080/v1/AUTH_%\(tenant_id\)s
[10.0.1.8]
#openstack endpoint create --region RegionOne object-store internal http://[IP-adres van de Swift proxy server]:8080/v1/AUTH_%\(tenant_id\)s
[10.0.1.8]
#openstack endpoint create --region RegionOne object-store admin http://[IP-adres van de Swift proxy server]:8080/v1/AUTH_%\(tenant_id\)s
[10.0.1.8]

9.1 Proxy node installatie

Eerst moet er een nieuwe server worden ingerichte met Ubuntu 14.04 LTS en moet de OpenStack repository worden toegewezen:

#apt-get install software-properties-common
#add-apt-repository cloud-archive:liberty
#apt-get update
#apt-get dist-upgrade

Hierna moeten de nodige packages worden geïnstalleerd:

#apt-get install swift swift-proxy python-swiftclient python-keystonemiddleware memcached

Nu moet er een nieuwe directory worden aangemaakt voor de Swift files:

#mkdir /etc/swift

9.1.1 /etc/swift/proxy-server.conf bestand

met een text-editor moet er een nieuw bestand worden aangemaakt en direct worden geopend:

#/etc/swift/proxy-server.conf

In dit bestand moet de volgende tekst worden geplaatst:

[DEFAULT]

```
bind_ip = 0.0.0.0
bind_port = 8080
user = swift

[pipeline:main]
pipeline = healthcheck cache authtoken keystoneauth proxy-server

[app:proxy-server]
use = egg:swift#proxy
allow_account_management = true
account_autocreate = true

[filter:authtoken]
paste.filter_factory = keystonemiddleware.auth_token:filter_factory
auth_uri = http://[IP-adres van de controller node op het management netwerk]
[10.0.1.2]:5000
auth_url = http://[IP-adres van de controller node op het management netwerk]
[10.0.1.2]:35357
auth_plugin = password
project_domain_id = default
user_domain_id = default
project_name = service
username = swift
password = [Password van de swift service in Keystone]
delay_auth_decision = true

[filter:keystoneauth]
use = egg:swift#keystoneauth
operator_roles = admin,swiftoperator

[filter:healthcheck]
use = egg:swift#healthcheck

[filter:cache]
use = egg:swift#memcache

[filter:catch_errors]
use = egg:swift#catch_errors
```

9.1.2 /etc/swift/swift.conf

Hierna moet er nog een nieuw bestand worden aangemaakt en worden geopend met de text editor, nl: /etc/swift/swift.conf:

#nano /etc/swift/swift.conf

In dit bestand moet de volgende code worden gezet:

```
[swift-hash]
swift_hash_path_suffix = [random secret]
```

```
swift_hash_path_prefix = [random secret]
```

Nu moet deze directory “swift” als owner krijgen:

#chown -R swift. /etc/swift

9.1.3 Swift ring

Hierna moet de “ring” worden aangemaakt:

#swift-ring-builder /etc/swift/account.builder create 12 3 1
#swift-ring-builder /etc/swift/container.builder create 12 3 1
#swift-ring-builder /etc/swift/object.builder create 12 3 1
#swift-ring-builder /etc/swift/account.builder add r0z0-[IP-adres van obj.storage node 1][10.0.1.9]:6002/device0 100
#swift-ring-builder /etc/swift/container.builder add r0z0-[IP-adres van obj.storage node 1][10.0.1.9]:6001/device0 100
#swift-ring-builder /etc/swift/object.builder add r0z0-[IP-adres van obj.storage node 1][10.0.1.9]:6000/device0 100
#swift-ring-builder /etc/swift/account.builder add r1z1-[IP-adres van obj.storage node 1][10.0.1.10]:6002/device1 100
#swift-ring-builder /etc/swift/container.builder add r1z1-[IP-adres van obj.storage node 1] [10.0.1.10]:6001/device1 100
#swift-ring-builder /etc/swift/object.builder add r1z1-[IP-adres van obj.storage node 1][10.0.1.10]:6000/device1 100
#swift-ring-builder /etc/swift/account.builder add r2z2-[IP-adres van obj.storage node 1][10.0.1.11]:6002/device2 100
#swift-ring-builder /etc/swift/container.builder add r2z2-[IP-adres van obj.storage node 1] [10.0.1.11]:6001/device2 100
#swift-ring-builder /etc/swift/object.builder add r2z2-[IP-adres van obj.storage node 1][10.0.1.11]:6000/device2 100
#swift-ring-builder /etc/swift/account.builder rebalance
#swift-ring-builder /etc/swift/container.builder rebalance
#swift-ring-builder /etc/swift/object.builder rebalance

De .gz files die zijn aangemaakt bij het aanmaken van de ring moeten de “swift” user als owner krijgen:

#chown swift. /etc/swift/.gz*

Nu kan de swift proxy server worden gestart:

#initctl start swift-proxy

9.2 Object storage nodes (* 3)

De object sotrage nodes worden aangemaakt en geconfigureerd met de onderstaande handelingen. Omdat alle configuratie gelijk is, enkel een aantal variabelen zijn anders. Door deze reden is de configuratie maar één keer beschreven. De configuraite moet echter wel op allee dire de object storage servers uitgevoerd worden. Storage node 1 = device0, storage node 2 = device1, storage node 3 = device2.

Eerst moet er een nieuwe server worden ingerichte met Ubuntu 14.04 LTS en moet de OpenStack repository

worden toegewezen:

#apt-get install software-properties-common
#add-apt-repository cloud-archive:liberty
#apt-get update
#apt-get dist-upgrade

Hierna moeten de nodige packages worden geïnstalleerd:

#apt-get install swift swift-account swift-container swift-object xfsprogs

Nu moet de tweede harde schijf in het systeem worden omgezet naar het XFS filesystem:

#mkfs.xfs -i size=1024 -s size=4096 /dev/sdb1

Hierna moet er een mount directory worden aangemaakt, de tweede HDD mounten op deze directory en de owner van de mount directory naar “swift” zetten:

#mkdir -p /srv/node/[devicenaam][device0]
#mount -o noatime,nodiratime,nobarrier /dev/sdb1 /srv/node/[devicenaam][device0]
#chown -R swift. /srv/node

Eerst moet het /etc/fstab bestand worden geopend:

#nano /etc/fstab

Nu moet in de fstab file de mountpoint worden aangemaakt, aan het eind van het bestand:

/dev/sdb1 /srv/node/*[devicenaam][device0]* xfs noatime,nodiratime,nobarrier 0 0

Vanaf de proxy-server moeten de ring bestanden naar de storage nodes worden verstuurd:

#scp /etc/swift/*.gz [IP-adres van de object storage node][10.0.1.9]:/etc/swift/

De ring files moeten nog “swift” als owner krijgen:

#chown swift. /etc/swift/.gz*

Ook op de object storage nodes moet het swift.conf bestand worden aangemaakt:

#nano /etc/swift/swift.conf

In dit bestand moet dezelfde waarde komen als het swift.conf bestand heeft op de proxy server:

```
[swift-hash]
swift_hash_path_suffix = [random secret, zelfde als op de proxy node]
swift_hash_path_prefix = [random secret, zelfde als op de proxy node]
```

Vervolgens moet rsync worden geconfigureerd, er moet een nieuw bestand worden aangemaakt:

#nano /etc/rsyncd.conf

Hier moet de volgende tekst in komen te staan:

```
pid file = /var/run/rsyncd.pid
log file = /var/log/rsyncd.log
uid = swift
gid = swift
address = [IP-adres van de object storage node][10.0.1.9]

[account]
path            = /srv/node
read only       = false
write only      = no
list            = yes
incoming chmod  = 0644
outgoing chmod  = 0644
max connections = 25
lock file =     /var/lock/account.lock

[container]
path            = /srv/node
read only       = false
write only      = no
list            = yes
incoming chmod  = 0644
outgoing chmod  = 0644
max connections = 25
lock file =     /var/lock/container.lock

[object]
path            = /srv/node
read only       = false
write only      = no
list            = yes
incoming chmod  = 0644
outgoing chmod  = 0644
max connections = 25
lock file =     /var/lock/object.lock

[swift_server]
path            = /etc/swift
read only       = true
write only      = no
list            = yes
incoming chmod  = 0644
outgoing chmod  = 0644
max connections = 5
lock file =     /var/lock/swift_server.lock
```

Nu moet rsync worden geactiveerd, eerst moet het bestand */etc/default/rsync* worden geopend:

#nano /etc/default/rsync

VOOR: *RSYNC_ENABLE=false*
NA: *RSYNC_ENABLE=true*

Hierna moet de rsync service worden gestart:

#/etc/init.d/rsync start

Tenslotte moeten de service worden herstart. Omdat het veel services zijn kan dit het beste gebeuren met een loop:

#for ringtype in account container object; do initctl restart swift-$ringtype; for service in replicator updater auditor; do if [$ringtype != 'account'] || [$service != 'updater']; then initctl restart swift-$ringtype-$service; fi; done; done

9.3 Functietest

De Swift service kan worden getest door objecten aan te maken in Swift. Deze acties worden uitgevoerd vanaf een client machine, dit kan ook vanaf de controller node.
Eerst moet de admin user actief zijn:

#source /etc/admin-openrc.sh

Eerst kan met met het volgende commando worden gekeken of de service en proxy server werkt en actief is:

#swift stat

Dan moet er eerst een container worden aangemaakt, met list kan worden gecontroleerd of het ook daadwerkelijk is aangemaakt:

#swift post [naam van de container][test1]
#swift list

Nu kan er een bestand worden geüpload naar deze container, met list kan worden gekeken of het bestand daadwerkelijk in de container aanwezig is:

#swift upload [naam van de container][test1] [naam van het random bestand][test.txt]
#swift list test1

Nu kan het bestand weer worden gedownload naar de client.

#swift download test1 test.txt

Hierna kan het bestand en de container weer worden verwijderd met:

#swift delete test1 test.txt
#swift delete test1

9.4 Cinder backup driver opzetten

Om de Cinder block storage backups te kunnen laten maken naar Swift moet de backup_driver worden opgezet en geconfigureerd.

Eerst moet op de block storage node het cinder-backup pakket worden geïnstalleerd:

#apt-get install cinder-backup

Op de block storage node moet het bestand /etc/cinder/cinder.conf worden aangepast. De volgende waarde moet worden toegevoegd onder de [DEFAULT] kop:

#backup_driver = cinder.backup.drivers.swift

Op de controller node moet het bestand /etc/cinder/cinder.conf worden aangepast. De volgende waarde moet worden toegevoegd onder de [DEFAULT] kop:

#backup_driver = cinder.backup.drivers.swift

Op de block storage node moeten de service worden herstart:

#service tgt restart
#service cinder-volume restart
#service cinder-backup restart

Op de controller node moet de cinder-scheduler service worden herstart:

#service cinder-scheduler restart

9.4.1 Functietest

Om de backup-driver te kunnen testen moeten eerst de cinder-volumes duidelijk zijn:

#cinder list

Met deze lijst kan worden bepaald welk volume een backup nodig heeft. Een live of non-disruptive backup wordt uitgevoerd met:

#cinder backup-create --force [VolumeID]

Met het volgende commando kan worden gecontroleerd of de backup wordt of is gemaakt:

#cinder backup-list

Als fouten in de backup zo ver mogelijk uitgesloten moeten zijn dan moet het volume eerst worden afgekoppeld. Vervolgens moet de betreffende instance (waar de volume aangekoppeld zit) uitgeschakeld worden:

#nova stop [instanceID of instance name]

Vervolgens moet het volume worden afgekoppeld:

#nova volume-detach [instanceID] [volumeID]

Nu kan de backup van de volume worden gemaakt:

#cinder backup-create [volumeID]

Met het volgende commando kan worden gecontroleerd of de backup wordt of is gemaakt:

#cinder backup-list

9.5 Glance image driver opzetten

Om voor Glance een Swift-backend op te zetten zijn nog een aantal stappen nodig. Hier moeten een aantal wijzigingen in het */etc/glance/glance-api.conf* bestand op de controller node aan worden gebracht.

Alle wijzigingen moet plaatsvinden onder de *glance_store* kop.
VOOR: default_store = file
#stores =
filesystem_store_datadir = /var/lib/glance/images/
#swift_store_auth_address =
#swift_store_user =
#swift_store_key =
#swift_store_create_container_on_put = False
NA: default_store = swift
stores = glance.store.swift.Store
#filesystem_store_datadir = /var/lib/glance/images/
swift_store_auth_address = http://10.0.1.2:35357/v2.0/
swift_store_user = service:glance
swift_store_key = [Password van de Glance user in Keystone]
swift_store_create_container_on_put = True

Na deze veranderingen moeten de *glance-api* en *glance-registry* service worden herstart. Dit gebeurd met:

#service glance-api restart
#service glance-registry restart

9.5.1 Functietest

Om de Swift backend voor Glance uit te testen moeten eerst de admin credentials worden geladen:

#source [pad naar het admin credentials bestand]

Nu moet er een image te uploaden zijn naar de Glance service (met Swift als backend):

#glance image-create –-name “Ubuntu_1404_cloud” –-disk-format=qcow2 –-container-format=bare –-visibility

public –-progress

Met het volgende commando kan worden gecontroleerd of de image in de Glance store staat:

#glance image-list

Nu moet er worden gecontroleerd of de image ook in de Swift backend aanwezig is. Hiervoor moet eerst een *openrc* credentials bestand worden aangemaakt voor de Glance service user aangezien dit de enige user is die de images in de backend kan zien.

Er moet dus een *openrc* bestand worden aangemaakt. Dit bestand moet de naam "*glance_user-openrc.sh*" krijgen. Dit bestand moet de volgende inhoud krijgen:

```
export OS_PROJECT_DOMAIN_ID=default
export OS_USER_DOMAIN_ID=default
export OS_PROJECT_NAME=service
export OS_TENANT_NAME=service
export OS_USERNAME=glance
export OS_PASSWORD=[password van de Glance user in Keystone]
export OS_AUTH_URL=http://[IP-adres van de controller node op het management
netwerk][10.0.1.2]:35357/v3
export OS_IDENTITY_API_VERSION=3
export OS_AUTH_VERSION=3
```

Door dit bestand te sources worden de environment variablen zo geset dat de Glance user in Swift kan kijken:

#source [pad naar het openrc bestand]/glance_user-openrc.sh

Vervolgens moet er in Swift de list worden opgevraagd:

#swift list

Als output moet er "glance" in de listing worden teruggegeven. Met het volgende commando kan er in de glance-container worden gekeken:

#swift list glance

De output van dit commando moet één of een aantal objects zijn.

10. SSH-jumpbox opzetten

In dit hoofdstuk wordt er uitgelegd hoe de SSH-jumpbox is opgezet. Als eerste is er een lege node nodig. Deze node moeten worden voorzien van Ubuntu 14.04 server edition.

Eerst moet de node worden geüpdatet:

#apt-get update
#apt-get dist-upgrade

Vervolgens moet er een user worden aangemaakt. Dit gaat volgens de beschrijving in hoofdstuk 1.5.

Hierna moet er met Puttygen een key worden aangemaakt. Deze key gaat gebruikt worden als SSH-key.
In het “Puttygen” programma moet er eerst op de knop “generate” worden geklikt. Puttygen zal nu een key genereren.
Na het genereren moet er een secret worden ingevoerd in de invoervelden “key passphrase” en “ Confrm passphrase”. Deze secret moet worden opgeslagen in de password manager.
Vervolgens moeten de private- en public key worden opgeslagen door te klikken op de knoppen “Save public key” en “Save private key”.
De Key die verschijnt in het venster bovenin Puttygen moet worden gekopieerd.

Op de SSH-jump node moet er eerst een directory worden aangemaakt en de rechten van deze directory moeten worden aangepast:

#mkdir ~/.ssh
#chmod 0700 ~/.ssh

Vervolgens moet het *~/.ssh/authorized_keys* bestand worden aangemaakt en de rechten moeten worden aangepast:

#touch ~/.ssh/authorized_keys
#chmod 0644 ~/.ssh/authorized_keys

Nu moet het *authorized_keys* bestand worden geopend en de gekopieerde tekst uit Puttygen moet hier in worden gekopieerd:

#nano ~/.ssh/authorized_keys

In putty moet vervolgens de key worden ingeladen en worden gekoppeld aan een verbinding. Onder session moet het IP-adres van de SSH-jumpbox worden ingegeven. Bij “saved sessions” kan de naam van deze sessie worden ingegeven. Onder “data” moet de Auto-login username van de user op de SSH-jumpbox worden ingegeven. Onder SSH → Auth moet de private key worden ingeladen. Dit gaat over de private key die is opgeslagen met Puttygen. Tenslotte moet de sessie worden opgeslagen met Session → Save. Nu kan de sessie worden opgezet met Session → Open.

Hierna moeten de Username/Password logins via SSH worden uitgeschakeld. Eerst moet het bestand */etc/ssh/sshd_config* worden geopend:

#nano /etc/ssh/sshd_config

In dit bestand moeten de volgende waarden worden aangepast:

VOOR: *#PasswordAuthentication yes*
NA: *PasswordAuthentication no*

VOOR: *UsePAM yes*
NA: *UsePAM no*

Tenslotte moet de SSH-service worden herstart om de veranderingen van kracht te laten gaan:

#service ssh restart

10.1 Iptables instellen

```
#!/bin/sh
IPT=/sbin/iptables

## FLUSH
$IPT -F
$IPT -X
$IPT -t nat -F
$IPT -t nat -X
$IPT -t mangle -F
$IPT -t mangle -X

### SSH extern ###

iptables -A INPUT -i eth1 -p tcp --dport 50385 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth1 -p tcp --sport 50385 -m state --state ESTABLISHED -j
ACCEPT

### NTP intern ###

$IPT -A OUTPUT -o eth0 -p tcp -d 10.0.1.15 --dport 123 -m state --state
NEW,ESTABLISHED -j ACCEPT
$IPT -A INPUT -i eth0 -p tcp -s 10.0.1.15 --dport 123 -m state --state ESTABLISHED
-j ACCEPT

### FTP intern ###

-A INPUT -i eth0 -p tcp -m tcp --dport 21 -s 10.0.1.0/24 -m state --state
ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 20 -s 10.0.1.0/24 -m state --state
ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --sport 1024:65535 --dport 20:65535 -s 10.0.1.0/24
-m state --state ESTABLISHED -j ACCEPT

-A OUTPUT -o eth0 -p tcp -m tcp --dport 21 -d 10.0.1.0/24 -m state --state
NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 20 -d 10.0.1.0/24 -m state --state
```

```
ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --sport 1024:65535 --dport 20:65535 -d 10.0.1.0/24
-m state --state ESTABLISHED,RELATED -j ACCEPT

### SSH intern input ###

iptables -A INPUT -i eth0 -p tcp --dport 50385 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 50385 -m state --state ESTABLISHED -j
ACCEPT

### SSH intern jump ###

iptables -A OUTPUT -o eth0 -p tcp --sport 50385 -m state --state NEW,ESTABLISHED
-j ACCEPT
iptables -A INPUT -i eth0 -p tcp --dport 50385 -m state --state ESTABLISHED -j
ACCEPT

### default policy DROP

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
```

11. Linux repository node

In dit hoofdstuk wordt de procedure uitgelicht waarmee de repository node is opgezet in het proof of concept. Deze node moeten worden voorzien van Ubuntu 14.04 server edition.

Eerst moet de node worden geüpdatet:

#apt-get update
#apt-get dist-upgrade

Vervolgens moet er een user worden aangemaakt. Dit gaat volgens de beschrijving in hoofdstuk 1.5.

Op deze node zijn twee pakketten van belang, nl: *proftp* en *apt-mirror*. Deze pakketten moeten worden dan ook worden gedownload:

#apt-get install apt-mirror proftpd-basic

In de installatie van proftp wordt gevraagt hoe de installatie moet worden geconfigureerd. Hier moet voor “standalone” worden gekozen.

11.1 APT-mirror

Met APT-mirror kan een eigen repository node worden aangemaakt. Dit wordt gedaan aan de hand van een al bestaande mirror.

In het */etc/apt/mirror.list* bestand moet het pad worden opgegeven waar de packets worden neergezet. Tevens moet de source mirror worden opgegeven:

VOOR: set base_path /
NA: set base_path /var/spool/apt-mirror

VOOR: *deb http://archive.ubuntu.com/ubuntu trusty main restricted universe multiverse*
deb http://archive.ubuntu.com/ubuntu trusty-security main restricted universe multiverse
deb http://archive.ubuntu.com/ubuntu trusty-updates main restricted universe multiverse
deb http://archive.ubuntu.com/ubuntu trusty-proposed main restricted universe multiverse
deb http://archive.ubuntu.com/ubuntu trusty-backports main restricted universe multiverse

deb-src http://archive.ubuntu.com/ubuntu trusty main restricted universe multiverse
deb-src http://archive.ubuntu.com/ubuntu trusty-security main restricted universe multiverse
deb-src http://archive.ubuntu.com/ubuntu trusty-updates main restricted universe multiverse
deb-src http://archive.ubuntu.com/ubuntu trusty-proposed main restricted universe multiverse
deb-src http://archive.ubuntu.com/ubuntu trusty-backports main restricted universe multiverse
NA: *deb http://mirror.i3d.net/pub/ubuntu trusty main restricted universe multiverse*
deb http://mirror.i3d.net/pub/ubuntu trusty-security main restricted universe multiverse
deb http://mirror.i3d.net/pub/ubuntu trusty-updates main restricted universe multiverse
deb http://mirror.i3d.net/pub/ubuntu trusty-proposed main restricted universe multiverse
deb http://mirror.i3d.net/pub/ubuntu trusty-backports main restricted universe multiverse
deb http://ubuntu-cloud.archive.canonical.com/ubuntu trusty-updates/liberty main

deb http://mirror.i3d.net/pub/ubuntu trusty main restricted universe multiverse
deb http://mirror.i3d.net/pub/ubuntu trusty-security main restricted universe multiverse
deb-src http://mirror.i3d.net/pub/ubuntu trusty-updates main restricted universe multiverse
deb-src http://mirror.i3d.net/pub/ubuntu trusty-proposed main restricted universe multiverse
deb-src http://mirror.i3d.net/pub/ubuntu trusty-backports main restricted universe multiverse

Eerst moet de directory worden aangemaakt die is opgegeven in het */etc/apt/mirror.list* bestand:

#mkdir -p /var/spool/apt-mirror

Vervolgens kan de lokale mirror worden gesynchroniseerd met de opgegeven mirror. Dit kan een aantal uren duren:

#apt-mirror

Ook moet er een automatische synchronisatie worden uitgevoerd om de lokale mirror up-to-date te houden. In het proof of concept wordt dit 1 uur 's nachts uitgevoerd. Dit kan worden uitgevoerd met:

#crontab -e
##editor keuze
#2

Als laatste regel code moet de volgende tekst worden ingeplakt:

```
0 1 * * * /usr/bin/apt-mirror >> /var/log/apt-mirror/apt-mirror.log
```

Om problemen te voorkomen moet het log bestand nog worden aangemaakt:

#touch /var/log/apt-mirror/apt-mirror.log

11.2 ProFTP

Om alle (interne) gebruikers toegang te kunnen verschaffen naar de repository node met FTP moet er een configuratiebestand worden aangemaakt:

#nano /etc/proftpd/conf.d/anonymous.conf

In dit bestand moet de volgende code worden geplakt:

```
<Anonymous ~ftp>
   User ftp
   Group nogroup
   UserAlias  anonymous ftp
   RequireValidShell off
#  MaxClients 10
       <Directory *>
            <Limit WRITE>
                 DenyAll
```

```
        </Limit>
    </Directory>
</Anonymous>
```

Vervolgens moet de mirror worden gemount op de FTP-service:

#mount --bind /var/spool/apt-mirror/mirror/mirror.i3d.net/ /srv/ftp/

Bij een reboot moet de mount op de ProFTP-server ook weer worden opgepakt. Dit wordt gedaan met:

#update-rc.d proftpd enable

Vervolgens moet het /etc/rc.local bestand worden geopend:

#nano /etc/rc.local

Boven de code “*exit 0*” moet de volgende code worden geplakt:

```
sleep 5
sudo mount --bind  /var/spool/apt-mirror/mirror/mirror.i3d.net/ /srv/ftp/
```

Tenslotte moet de FTP-service worden herstart:

#service proftpd restart

11.3 Iptables instellen

```
#!/bin/sh
IPT=/sbin/iptables

## FLUSH
$IPT -F
$IPT -X
$IPT -t nat -F
$IPT -t nat -X
$IPT -t mangle -F
$IPT -t mangle -X

### HTTP ###

#$IPT -A OUTPUT -o eth1 -p tcp -s [IP-adres van de repository node op het i3D-
netwerk] -d 213.163.76.200 --sport 80 -m state --state NEW,ESTABLISHED -j ACCEPT
#$IPT -A INPUT -i eth1 -p tcp -s 213.163.76.200 -d [IP-adres van de repository
node op het i3D-netwerk] --dport 80 -m state --state ESTABLISHED -j ACCEPT

### DNS 1 ###

$IPT -A OUTPUT -o eth1 -p udp -d 213.163.76.185 --dport 53  -m state --state
NEW,ESTABLISHED -j ACCEPT
```

```
$IPT -A INPUT -i eth1 -p udp -s 213.163.76.185 --sport 53  -m state --state
ESTABLISHED -j ACCEPT
$IPT -A OUTPUT -o eth1 -p tcp -d 213.163.76.185 --dport 53 -m state --state
NEW,ESTABLISHED -j ACCEPT
$IPT -A INPUT -i eth1 -p tcp -s 213.163.76.185 --sport 53 -m state --state
ESTABLISHED -j ACCEPT

### DNS 2 ###

$IPT -A OUTPUT -o eth1 -p udp -d 91.198.152.196 --dport 53  -m state --state
NEW,ESTABLISHED -j ACCEPT
$IPT -A INPUT -i eth1 -p udp -s 91.198.152.196 --sport 53  -m state --state
ESTABLISHED -j ACCEPT
$IPT -A OUTPUT -o eth1 -p tcp -d 91.198.152.196 --dport 53 -m state --state
NEW,ESTABLISHED -j ACCEPT
$IPT -A INPUT -i eth1 -p tcp -s 91.198.152.196 --sport 53 -m state --state
ESTABLISHED -j ACCEPT

### NTP intern ###

$IPT -A OUTPUT -o eth0 -p tcp -d 10.0.1.15 --dport 123 -m state --state
NEW,ESTABLISHED -j ACCEPT
$IPT -A INPUT -i eth0 -p tcp -s 10.0.1.15 --dport 123 -m state --state ESTABLISHED
-j ACCEPT

### FTP intern ###

-A INPUT -i eth0 -p tcp -m tcp --dport 21 -s 10.0.1.0/24 -m state --state
NEW,ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 20 -s 10.0.1.0/24 -m state --state
ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --sport 1024:65535 --dport 20:65535 -s 10.0.1.0/24
-m state --state ESTABLISHED,RELATED -j ACCEPT

-A OUTPUT -o eth0 -p tcp -m tcp --dport 21 -d 10.0.1.0/24 -m state --state
ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 20 -d 10.0.1.0/24 -m state --state
ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --sport 1024:65535 --dport 20:65535 -d 10.0.1.0/24
-m state --state ESTABLISHED -j ACCEPT

### SSH intern###

iptables -A INPUT -i eth0 -p tcp --dport 50385 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 50385 -m state --state ESTABLISHED -j
ACCEPT

### default policy DROP
```

```
$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
```

11.4 Functietest

Om de werking van de lokale mirror te testen moet een node worden aangepast zodat hij update en upgrade vanaf de lokale repository kan uitvoeren. In het /etc/apt/source.list bestand van een random node (in het voorbeeld de utility node) moeten de bestaande regels code worden weggecomment. De volgende regels code moeten vervolgens worden ingeplakt:

```
deb ftp://10.0.1.12/pub/ubuntu trusty main restricted
deb-src  ftp://10.0.1.12/pub/ubuntu trusty main restricted

deb  ftp://10.0.1.12/pub/ubuntu trusty-updates main restricted
deb-src  ftp://10.0.1.12/pub/ubuntu trusty-updates main restricted

deb  ftp://10.0.1.12/pub/ubuntu trusty universe
deb-src  ftp://10.0.1.12/pub/ubuntu trusty universe
deb  ftp://10.0.1.12/pub/ubuntu trusty-updates universe
deb-src  ftp://10.0.1.12/pub/ubuntu trusty-updates universe

deb  ftp://10.0.1.12/pub/ubuntu trusty multiverse
deb-src  ftp://10.0.1.12/pub/ubuntu trusty multiverse
deb  ftp://10.0.1.12/pub/ubuntu trusty-updates multiverse
deb-src  ftp://10.0.1.12/pub/ubuntu trusty-updates multiverse

deb  ftp://10.0.1.12/pub/ubuntu trusty-backports main restricted universe
multiverse
deb-src  ftp://10.0.1.12/pub/ubuntu trusty-backports main restricted universe
multiverse
```

Na het opslaan van het bestand zouden de volgende commando's moeten werken:

```
#apt-get update
#apt-get upgrade
```

12. NTP server

Binnen dit project is er voor gekozen om een dedicated NTP-node te maken. Om een NTP-server te installeren moet eerst het NTP-pakket geïnstalleerd worden:

#sudo apt-get install ntp

Om de NTP-server te configureren moet eerst het ***ntp.conf*** script worden aangepast. Dit wordt gedaan door:

#sudo nano /etc/ntp.conf

In dit script moeten de lijst met servers worden vervangen door het IP-adres van de interne NTP-server van i3D. De lijnen code die moeten worden vervangen beginnen allemaal met *server.* Deze code moet worden vervangen door:

server 213.163.76.185 iburst
server 31.3.104.60

Bij uitval van de connectie naar de time servers moet de lokale NTP-server goed blijven lopen. Onderaan het bestand moeten de volgende regels code worden meegenomen:

```
server 127.127.1.0
fudge 127.127.1.0 stratum 10
```

Vervolgens moeten de service afgeschermd worden:

```
### default restrict
restrict default kod nomodify nopeer notrap

### Exception IP's ###
restrict 10.0.1.0 mask 255.255.255.0 notrap nomodify noquery
restrict 213.163.76.185 mask 255.255.255.255 nomodify notrap noquery
```

Hierna moet de NTP-service worden herstart:

#sudo service ntp restart

De server is nu te controleren met:

#ntpq -p

Nu kunnen alle nodes in het management netwerk hun NTP-source op het IP-adres van de NTP-node zetten.

12.1 NTP-server beschermen tegen amplification attacks

Om de NTP-server te beschermen tegen amplification attacks zal de monitor uitgezet moeten worden. Als eerste moet de service uitgeschakeld en disabled moeten worden:

#service ntp stop
#update-rc.d -f ntpd remove

Vervolgens moet het bestand */etc/ntp.conf* worden geopend. In dit configuratie bestand moet helemaal onderaan de volgende regel code worden neergezet:

```
disable monitor
```

Na het opslaan van dit bestand moet de NTP-service weer worden herstart:

#service ntp restart

12.2 Iptables instellen

Er moet een script worden aangemaakt om Iptables makkelijk te kunnen configureren. Dit aanmaken en openen met nano gebeurd met:

#touch iptables_rules.sh
#chmod 700 iptables_rules.sh
#nano iptables_rules.sh

Voor deze server moeten de volgende regels code in het script komen te staan:

```
#!/bin/sh
IPT=/sbin/iptables

## FLUSH
$IPT -F
$IPT -X
$IPT -t nat -F
$IPT -t nat -X
$IPT -t mangle -F
$IPT -t mangle -X

### NTP-server sync ###

$IPT -A OUTPUT -o eth1 -p tcp -s [IP-adres van de NTP-node op het i3D-netwerk] -d
213.163.76.185 --sport 123 -m state --state NEW,ESTABLISHED -j ACCEPT
$IPT -A INPUT -i eth1 -p tcp -s 213.163.76.185 -d [IP-adres van de NTP-node op het
i3D-netwerk] --dport 123 -m state --state ESTABLISHED -j ACCEPT

### NTP-client listen ###

$IPT -A INPUT -i eth0 -p tcp -s 10.0.1.0/24 -d 10.0.1.15 --dport 123 -m state
--state NEW,ESTABLISHED -j ACCEPT
$IPT -A OUTPUT -o eth0 -p tcp -s 10.0.1.15 -d 10.0.1.0/24 --sport 123 -m state
--state ESTABLISHED -j ACCEPT

### FTP intern ###
```

```
-A INPUT -i eth0 -p tcp -m tcp --dport 21 -s 10.0.1.0/24 -m state --state
ESTABLISHED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --dport 20 -s 10.0.1.0/24 -m state --state
ESTABLISHED,RELATED -j ACCEPT
-A INPUT -i eth0 -p tcp -m tcp --sport 1024:65535 --dport 20:65535 -s 10.0.1.0/24
-m state --state ESTABLISHED -j ACCEPT

-A OUTPUT -o eth0 -p tcp -m tcp --dport 21 -d 10.0.1.0/24 -m state --state
NEW,ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --dport 20 -d 10.0.1.0/24 -m state --state
ESTABLISHED -j ACCEPT
-A OUTPUT -o eth0 -p tcp -m tcp --sport 1024:65535 --dport 20:65535 -d 10.0.1.0/24
-m state --state ESTABLISHED,RELATED -j ACCEPT

### SSH intern ###

iptables -A INPUT -i eth0 -p tcp --dport 50385 -m state --state NEW,ESTABLISHED -j
ACCEPT
iptables -A OUTPUT -o eth0 -p tcp --sport 50385 -m state --state ESTABLISHED -j
ACCEPT

### default policy DROP

$IPT -P INPUT DROP
$IPT -P OUTPUT DROP
$IPT -P FORWARD DROP
```

Na het maken van dit script zal het nog moeten worden uitgevoerd:

#./iptables_rules.sh

Met het volgende commando kan er worden gecontroleerd of de firewall goed staat geconfigureerd:

#iptables -L

13. Tests opzetten

13.1 Compute node tests

In deze test worden de compute nodes los getest op performance van CPU, RAM en diskIO.
Uiteraard moet er voor aanvang van het opzetten van de test een werkende OpenStack omgeving staan.

13.1.1 Ubuntu cloud instance aanmaken

De test wordt uitgevoerd met eerst 1 instance, hierna 2 instances, hierna 4 instances, vervolgens 6 instances en tenslotte 8 instances.
In dit sub hoofdstuk wordt uitgelegd hoe een Ubuntu 14.04 cloud instance aangemaakt kan worden.

Eerst een keypair voor de instance maken:

#nova keypair-add [naam van de key][test-key]
#nova keypair-list

De output van de key moet even worden opgeslagen in een tekstbestand. Vervolgens moeten de volgende commando's uit worden gevoerd om de nodig informatie op te vragen:

#nova flavor-list
#nova image-list
#neutron net-list
#nova secgroup-list

Met deze informatie kan het volgende commando uitgevoerd worden om een instance aan te maken:

#nova boot --flavor m1.small_small --image [image naam uit de nova image-list][ubuntucloud] --nic net-id=[ID van het test netwerk] --security-group default --key-name [Naam van het keypair][test-key] [naam die de instance moet krijgen][test-instance1]

#neutron floatingip-create [publiek netwerk][ext-net]
#nova floating-ip-associate private-instance [floating-IP]

Nu moet er een port worden opgezet in OpenStack. Het is port 22 om SSH te gebruiken.

nova secgroup-add-rule [Naam van de security group][default] tcp 22 22 0.0.0.0/0

Vervolgens kan er worden ingelogd naar de instance. Dit moet echter wel worden gedaan met een SSH-key. In het project is dit gedaan met Putty. De output uit het tekstbestand moet worden omgezet naar een putty-key. Dit wordt gedaan door het tekstbestand in te laden in “Putty Key generator”. Er kan optioneel nog een passphrase worden ingegeven. Als dit is ingesteld moet op “Save private key” worden geklikt.
In putty moet onder session het IP-adres van de instance worden ingegeven. Bij “saved sessions” kan de naam van deze sessie worden ingegeven. Onder “data” moet de Auto-login username van “*ubuntu*”. Onder SSH → Auth moet de private key worden ingeladen. Nu kan de sessie worden opgezet.

In de instance moet de MTU-size nog worden aangepast van 1500 naar 1400:

#sudo ip link set mtu 1400 dev eth0

13.1.2 CPU-benchmark opzetten (stress-ng)

Als CPU-load generator wordt Stress-ng gebruikt. Deze load generator moet op elke instance worden geïnstalleerd. Omdat er op de Ubuntu cloud instance geen CC of GCC aanwezig is moet het packet “build-essential” worden geïnstalleerd. Tevens moet het packet “make” worden geïnstalleerd. Door het feit dat stress-ng niet uit de Ubuntu repository gehaald kan worden moet deze zelf worden opgebouwd. Dit kan worden gedaan met:

#sudo apt-get install build-essential
#sudo apt-get install make
#http://kernel.ubuntu.com/~cking/tarballs/stress-ng/stress-ng-0.03.13.tar.gz
#tar zxf stress-ng-0.03.13.tar.gz
#cd stress-ng-0.03.13
#make

Het commando moet hiena uiteraard wel lokaal worden aangeroepen. Dit wordt gedaan met

./stress-ng -paramameters

Onderstaand voorbeeld wordt gebruikt om 1 CPU core te laten cyclen, voor de duur van 40 seconden met een CPU-load van 100%:

./stress-ng -c 1 -t 40 -l 100

13.1.2 CPU-benchmark opzetten (sysbench)

Om de performance van de CPU te kunnen meter wordt het programma “sysbench” uitgevoerd. Met het CPU-test gedeelte van sysbench wordt de tijd gemeten voordat het programma klaar is met het berekenen van X priemgetallen.
Om sysbench te installeren moet het volgende commando worden gebruikt:

#apt-get install sysbench

13.2 Connection test opzetten

Eerst moet er op de controller node een meettool worden aangemaakt. De usage van de dataverbinding moet worden gemeten. Dit gebeurd met het programma *ifstat*. Eerst moet *ifstat* worden geïnstalleerd:

#apt-get install ifstat

Om te testen met Rally moet eerst de role “member” worden aangemaakt binnen Keystone:

#openstack role create member

13.3 Storage connection test opzetten

Allereerst moet er een instance actief zijn op de compute node. Aan deze instance moet het volume worden gekoppeld. Het opzetten van de instance gaat volgens de methode uit subhoofdstuk 11.1.1.

Hierna moet er een volume worden aangemaakt op de block storage node:

#cinder create –-display-name [naam van het volume][testvolume1] [grootte van het volume in GB][1]

Hierna moet het block volume gekoppeld worden met de instance:

#nova volume-attach [instancenaame][test-instance1] [VolumeID][d48db218-66be-49ce-b1ca-df95dcb6982a]

Op de Ubuntu image moet het volume nu worden gepartitioneerd:

#sudo fdisk -l
#sudo fdisk [disk][vdb]

In het fdisk commandprompt moeten de volgende commando's worden ingegeven:

#n
#p
#1024
#2097151
#w

In de normale command prompt moet dan worden ingegeven:

#mkfs.ext3 /dev/[disk][vdb1]

Hierna kan de disk worden gemount:

#mkdir /media/disk1
#mount /dev/vdb1 /media/disk1

Bijlage VII: Tegengekomen problemen

In deze bijlage worden de problemen beschreven die opkwamen bij het opzetten van de OpenStack omgeving.

1. Invalid credentials

Bij het opzetten van de communicatie tussen Glance en Keystone is er een fout opgetreden. Deze fout kwam boven tafel bij het testen van deze verbinding. Bij de test was het de bedoeling om met Glance een image te uploaden naar de */var/lib/glance/images/* directory van de controller node. Echter kwam hier een foutmelding naar voren, nl: *"Invalid OpenStack Identity Credentials"*.
Het probleem bij deze foutmelding is het aantal problemen wat hier aan ten grondslag kan liggen. Eerst is er gekeken naar de logs. In de logs werd er aangegeven dat de Glance service geen token stuurde naar de Keystone service.
Om uit te sluiten dat het een verbindingsprobleem was werd eerst gekeken naar de ports waar Apache op moet luisteren. In eerdere versies luisterde de Keystone service zelf op de ports. Apache en Keystone kunnen conflicteren als de Keystone service per ongeluk is gestart. Apache luisterde op de juiste ports. Dit was dus niet het probleem.
Vervolgens is er gekeken naar de environment variabeles die moeten worden geset voor de admin user. Hier was ook alles goed.
Hierna is er gekeken naar de instellingen in de "glance-api.conf" en "glance-registry.conf" bestanden. Hier stond ook alles juist ingesteld.
Het probleem moest dus ergens anders zitten. De communicatie naar de Keystone server wordt gemaakt met de Glance user. Deze user is opnieuw aangemaakt, en heeft opnieuw de admin role toegewezen gekregen. De fout kan hier zijn dat het password van de Glance user niet overeenkomt met het password wat Keystone verwacht. Dit was het ook niet.
Uiteindelijk bleek het te gaan om een endpoint die niet bleek te bestaan. De endpoints waren bekeken met het commando *openstack endpoint list*. Hier kwam alleen het public endpoint naar voren. Hier moesten ook de internal en admin endpoints naar voren komen. Deze zijn aangemaakt. Vervolgens werkte het uploaden van de images wel.

2. ARP probleem

Bij het opzetten van de networking service kon de Neutron Open vSwitch agent op de compute node niet gezien worden. Hier trad het probleem op dat de Open vSwitch instance geen verbinding kon maken via het instance communication network, dit is het netwerk waar over de VM's communiceren naar de buitenwereld en naar de network node.
Eerst werden de configuratie files nagekeken op typefouten. Deze stonden er niet in.
Hierna werd de fysieke verbinding gecontroleerd. Hier kwam naar voren dat er niet kon worden gepingt over de lijn. Na het bekijken van de firewall/iptables instellingen en het tracen van het netwerk verkeer kwam ik er achter dat de ARP-requests nooit terug aankwamen. De network node stuurde dan wel een ARP-request naar de compute node, deze ontving het ook, hij stuurde iets terug (of probeerde het in ieder geval), maar dit kwam nooit aan. Na zeer lang troubleshooten en raadplegen van experts zijn de compute node en de network node opnieuw geïnstalleerd.

3. MAC-probleem

Na het opnieuw installeren van de network- en compute nodes kwamen er opnieuw fouten boven. Bij het opnieuw installeren van Linux zijn direct de netwerk insteekkaarten geïnstalleerd. De netwerkkaarten zijn voor het uitbreiden van de twee aanwezige network interfaces naar vier.
Bij het testen van de verbindingen tussen de controller- compute en network nodes kwam naar boven dat het niet werkte. Ook hier is weer meteen naar de iptables gekeken. Na het troubleshooten van de switch (VLAN-problemen) en de servers kwam ik er achter dat de interfaces in de server verwisselt waren. Dit idee kwam in mij op toen ik zag dat de MAC adressen in de switch helemaal niet overeen kwamen met mijn planning en Linux. De uitleg hierbij is dat Linux bij het koppelen van MAC-adressen aan interfaces altijd begint met het laagste MAC-adres. Toen het insteekkaartje er bij werd gezet zijn deze MAC-INT koppelingen dus verwisseld.

Op dit moment werd bijgehouden tot wanneer de verbinding tussen de compute- en network node zou blijven werken. Op deze manier kon worden nagegaan wanneer het ARP-probleem op zou treden en waar de fout zat. Echter bij het opnieuw opzetten van de compute node en de network node kwam er geen ARP-fout meer naar boven. Er kon dus ook geen fout meer worden getraceerd. Wel is het kans groot dat het probleem heeft gezeten bij het koppelen van de externe bridge aan het externe fysieke netwerk. Zeer waarschijnlijk heeft die bridge een conflict veroorzaak op het tunnel netwerk.

4. Unexpected API error (HTTP 500)

Bij het testen van de Nova service kwam er een “unexpected API error” op. Bij het testen werd er een Nova instance aangemaakt. Hiermee zouden alle services tot op dat punt samen moeten werken (Keystone, Glance, Nova en Neutron). Het vervelende van een “unexpected API error” is echter dat niet duidelijk is wat de fout is. Gelukkig wordt er op zo'n moment we gelogd wat de laatste paar acties van OpenStack zijn. Hier kwam echter ook geen duidelijk antwoord uit.

Aan het begin heb ik dus de logs gecontroleerd: hier kwam niet veel zinnigs uit, behalve het component waar het fout ging. Hierna heb ik de configuratie van het component (Nova) bekeken. Deze configuratie file is de *nova.conf* file op de controller node. Hier stond echter alles goed.
Hierna heb ik online naar de fout gezocht. Het bleek dat er veel mensen last van hadden, dit zou kunnen wijzen op een bug. Er waren al mensen die beweerden dat het lag aan een foute authenticatie (verkeerd password). Na de password te hebben gecontroleerd kon ik vrij zeker zeggen dat dat het niet was.
Vervolgens was er een persoon die aandroeg om te controleren of “/v2.0” was toegevoegd aan de “admin_auth_url” in het nova configuratie bestand. Dit hielp ook niet.

Tenslotte heb ik de logs nog verder terug gelezen. Hier stond ergens verstopt dat sommige instellingen in het *nova.conf* bestand “deprecated” waren, verlopen dus. Het ging hier om de “admin_auth_url” en aanhangende variabelen. Volgens de OpenStack documentatie moesten dit de “auth_plugin” en aanhangende variabelen worden. Na het aanpassen van deze waarden in het nova.conf bestand kan ik wel succesvol een instance aanmaken.

5. Partities op de block storage node

Bij het opzetten van de Cinder node moest er een groot stuk van het volume gepartitineerd worden maar zonder filesystem blijven. Dit was nodig om LVM op te zetten.
In de opbouw van het proof of concept en de proefopstelling zijn alle OS'en op de fysieke nodes geïnstalleerd door middel van het klantoverzicht. In het klantoverzicht van i3D kunnen instances worden gedeployed op servers. Voor het deployen moeten ook de partities worden ingegeven. Deze partities worden bij het deployen automatisch aangemaakt op het OS. Nu was het echter zo dat er geen partitie aangemaakt kon worden zonder filesystem. Dit bleek uit het proberen om dit toch voor elkaar te krijgen. In het overzicht waar de partities worden

aangemaakt moeten filesystem, grootte, mountpoint en naam worden aangegeven. Om de volledige HDD te gebruiken kan in plaats van een partitiegrootte ook “rest” worden ingegeven. Met deze waarde wordt de rest van de schijf aan de betreffende partitie meegegeven. Door geen rest waarde in te geven zou er eventueel een lege ruimte moeten ontstaan op de schijf.
Na het uitvoeren van de deployment bleek echter dat er geen lege ruimte was ontstaan. De laatste partitie in het overzicht (swap) was uitgesmeerd alsof deze de restwaarde ingegeven had gekregen. Er bestond nu dus een swap partitie van ruim 400GB. Na nog een aantal pogingen om dit goed te krijgen is er besloten om de swap partitie te resizen.

Dit resizen kan uitgevoerd worden met partitie programma's met een GUI. Het is van belang dat er wordt geboot van een bootable medium, en niet de HDD. Op deze manier zouden de partities niet worden gemount en zouden deze gemakkelijker aangepast moeten kunnen worden. Het opstarten van een GUI partitie manager via een bootable USB-stick was echter niet mogelijk. De server was niet in staat om “gparted” of “partitionmagic” op te starten. Het leek er op alsof de grafische kaart in de server dit niet pakte.

Vervolgens ging ik op de commandline tour. Door in Ubuntu in te loggen onder rescue mode waren de partities nog niet of nauwelijks in gebruik. Met parted is de swap partitie verwijderd en direct opnieuw aangemaakt met de juiste grootte. Na een herstart van het OS moest er weer in rescue mode worden ingelogd. Nu werd er met parted een nieuwe partitie aangemaakt op de resterende ruimte. Vervolgens kan er wel een LVM partitie van worden gemaakt.

6. Instance kan niet naar internet verbinden

Bij het opzetten van de tests in de architectuurfase kwam er naar voren dat de instances (Ubuntu 14.04 cloud) niet konden verbinden met het internet. Eerst werd er onderzocht of dit misschien aan de firewall op de Linux instance kon liggen. Deze is geflushed. Hierna werkte het nog niet.
Hierna werd er gekeken of het aan de security groups in OpenStack zou kunnen liggen. De ports stonden hier echter goed. Dit was het dus ook niet.
Het pingen van een URL op het internet werkte wel, het was dus niet zo dat er helemaal geen connectiviteit was. Ook was het mogelijk om naar deze instance een SSH-verbinding op te zetten.
Het probleem trad op bij het downloaden van een packet en het updaten van de repository. Hier zou het probleem dat eventueel gevonden moeten worden. Bij het connecten met speedtest liep hij elke keer vast bij:

```
"Connecting to speedtest.dal01.softlayer.com (speedtest.dal01.softlayer.com)|
74.86.116.210|:80... connected.
HTTP request sent, awaiting response..."
```

Met het Linux programma “tcpdump” werd er eerst gekeken naar de pakketjes die verstuurd werden. Hieruit bleek dat de pakketjes wel werden verstuurd maar eventueel niet konden terugkeren.
Ondertussen had ik op het OpenStack forum een vraag over dit probleem geplaatst. Een gebruiker liet mij weten dat het eventueel met de MTU-size van de GRE-tunnel te maken had. Dit zou een vaker voorkomend probleem zijn.
De standaard MTU-size van de instance stond op 1500. De maximum MTU-size staat ook op 1500. Bij het vormen van een GRE-tunnel moet er ook rekening worden gehouden met de GRE-header. Deze header moet nog van de MTU-size van 1500 worden afgehaald. Na het aanpassen van de MTU-size van 1500 naar 1454 werkte de verbinding ineens goed.

7. Fout bij het starten van Swift-proxy

Bij het opzetten van de Swift proxy node kwam er een fout op bij het herstarten van de swift-proxy service. De fout kwam op bij het herstarten van de Swift-proxy met initctl:

“initctl: Job failed to start”

echter heeft de proxy service geen dedicated log file. Er moest dus eerst worden gezocht waar de logs opgeslagen werden. Dit bleek in syslog te zijn. In de syslog stond de volgende melding:

“init: swift-proxy pre-start process (3551) terminated with status 1”

Met deze melding was niet zo veel aan te vangen. Hierna is er gezocht in het swift-proxy.conf bestand naar eventuele typefouten en configuratiefouten. Deze bleken er echter niet te staan. Na veel zoeken in de OpenStack fora bleek dat de service ook gestart kon worden met “swift-proxy-server” dit zou wel een goede foutmelding genereren. Bij het starten met *swift-proxy-server /etcswift/proxy-server.conf*. Gaf de service de volgende melding:

“Error trying to load config from /etc/swift/proxy-server.conf: File contains no section headers.
file: /etc/swift/proxy-server.conf, line: 1
' # create new\n'”

Dit was een duidelijke foutmelding. Bij het kijken in het swift-proxy.conf bestand bleek op line 1 de volgende comment te staan:

##proxy-server

Hier bleek de service problemen mee te hebben. Na het verwijderen van van deze lijn code startte de service direct.

8. Fout met de Swift-ring

Bij de functietest van OpenStack Swift was het de bedoeling om met het commando *#swift put [naam]* een swift container aan te kunnen maken binnen Swift. Er kwam echter een fout op:

```
Account GET failed: http://10.0.1.10:8080/v1/AUTH_39c7abff851e46a9add203a6a8e00e4a?
format=json 503 Internal Server Error  [first 60 chars of response]
<html><h1>Service Unavailable</h1><p>The server is currently
```

Aan deze fout was niet zo veel te zien. Het enige wat de service als feedback geeft was dat de service niet bereikbaar is. Om wat dieper in het probleem te duiken moest de logs worden geraadpleegd. De Swift logs staan niet op de controller node maar op de object proxy node in het bestand /var/log/syslog. In de log kon het volgende worden gevonden:

```
proxy-server: ERROR Insufficient Storage 10.0.1.10:6002/sdb1 (txn:
tx48d9c4a4a0614a2699104-0054aa5c6d)
```

In commando's die daarvoor gegeven waren aan Swift was niet te zien dat de service fouten gaf. Zo was er met het *#swift stat* commando de statistieken bekeken van Swift. Hier kwam geen fout op. Ook met het commando *#swift list* werd er geen fout gegeven. Hier kwam helemaal niks terug, maar dat is goed, want er was ook nog niks aangemaakt binnen Swift.

De foutmelding was vreemd aangezien alle drie de object storage nodes waren uitgerust met genoeg schijfruimte. Bij de functietest werd alleen geprobeerd om een tekstbestand van een aantal byte te uploaden naar Swift. Er moest dus iets anders aan de hand zijn.
Allereerst werd het swift configuratiebestand op de object proxy node opnieuw bekeken op type- of configuratiefouten. Deze stonden hier echter niet in.
Bij het zoeken op het OpenStack forum kwam ik nog een gebruiker tegen die dezelfde foutmelding had gekregen. Er stond in deze thread ook een eventuele oplossing bij. Het probleem zou kunnen zijn dat het paden naar de object storage nodes niet goed staan in de object-server.conf-, container-server.conf- en account-server.conf bestanden. Deze bestanden staan op elke object storage node.

In deze bestanden zouden de volgende parameter verkeerd staan: *swift_dir en devices*. De waarden die deze parameters zouden moeten hebben volgens de thread waren respectievelijk: */etc/swift* en */srv/nodes*.
Na het aanpassen van deze waardes herstartte ik alle Swift-services op de object storage nodes. Na de herstart bleek het probleem nog steeds te bestaan. Tevens kwam ik door een ander thread er achter dat deze paden niet meegestuurd hoeven te worden aangezien de paden in een ander configuratiebestand als ingesteld staan.

Een volgend thread op het forum gaf aan dat er ook een fout in de Swift-ring zou kunnen zitten. Bij de installatie van Swift wordt de ring aangemaakt op de object proxy node. Alle nodes in de Swift installatie moeten bekend zijn met de ring, dus de bestanden die de ring een geheel maken moeten naar elke node worden gekopieerd. Dit is in de installatie gedaan met SCP. De ring is aangemaakt en vervolgens zijn de nodige bestanden verstuurd van de object proxy node naar de object storage nodes.
Om er zeker van te zijn dat de ring goed zou zijn is deze eerst verwijderd en vervolgens opnieuw aangemaakt. Dit gebeurd met het *swift-ring-builder remove* en *swift-ring-builder add* commando. De ring bestanden werden ook nu weer naar de object storage nodes verstuurd met SCP. Hierna zijn alle Swift services herstart op de object proxy node en de object storage nodes. Ook hierna bleef het probleem bestaan.

Na nog een aantal configuraties geprobeerd te hebben besloot ik om de gehele Swift installatie (alle fysieke nodes) een hard reboot te geven. Na het opstarten van de service probeerde ik opnieuw het *#swift put* commando. Nu werkte het echter wel.

De fout zal waarschijnlijk hebben gezeten in een onderliggende service die een volledige reboot nodig had. Of er was een onderdeel die was vastgelopen en “los kwam” door de reboot. Uiteindelijk was het dus wel te verklaren waarom de *#swift stat* en *#swift list* commando's goed uitgevoerd werden: deze hadden geen verbinding nodig van de object proxy server naar de object proxy nodes.

Bijlage VIII: Onderzoeksverslag

Onderzoeksverslag

Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low-cost cloud oplossing te creëren bij i3D

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Stagebedrijf: i3D
Plaats: Rotterdam
Datum: 01-02-2016

Versiebeheer

Versie	Datum	Wijziging
0.1	01-02-2016	Layout opgezet
0.2	15-02-2016	Eerste concept
0.3	25-02-2016	Tweede versie, spelfouten verbeterd en onderdelen toegevoegd.
0.4	05-04-2016	Derde versie, spelfouten verbeterd en onderdelen uit de architectuurfase toegevoegd.
0.5	17-04-2016	Vierde versie, spelfouten verbeterd en onderdelen uit de infrastructuurfase toegevoegd.
1.0	22-05-2016	Laatste spelling- en grammaticacontrole. Het document is klaar voor oplevering.

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

Inleiding

In dit document worden de resultaten van het literatuuronderzoek en fysieke onderzoek besproken. Dit onderzoek volgt als het ware op de vragen uit de probleemstelling. Voor het grootste gedeelte heeft het literatuuronderzoek in de richting van OpenStack gewezen. Bij het opzetten van een proof of concept op basis van OpenStack is kennis van OpenStack van cruciaal belang.
In het eerste gedeelte van het document wordt de cloud in het algemeen beschreven. Al snel wordt deze basiskennis van de cloud opgevolgd door een gedeelte over OpenStack. In dit gedeelte wordt vooral onderzocht wat OpenStack precies is, uit welke onderdelen het bestaat en welke eisen het stelt aan systemen (theoretisch). Omdat de opdrachtgever speciaal heeft gevraagd om ook Ceph onder de loep te nemen en bij geschikt bevinden te gebruiken is dat pakket ook onderzocht. Net als bij het onderzoek naar OpenStack wordt bij Ceph ook gekeken wat het is, welke eisen het aan systemen stelt, en uit welke onderdelen het bestaat. Tevens is de koppeling tussen OpenStack en Ceph onderzocht en de verschillen en overeenkomsten tussen de twee.

Echter was er ook een wat meer fysiek onderzoek nodig. Er waren systemen opgeleverd voor het project en deze moesten worden geïnventariseerd. Aan het eind van het document is deze inventarisatie besproken samen met het vraagstuk hoe de directe netwerk omgeving van de opstelling er uit ziet.

Inhoudsopgave

Verklarende woordenlijst / symbolenlijst

Actor = Binnen OpenStack wordt met een actor een user of een usergroup bedoeld. Een role kan toegewezen worden aan (assigned to) een actor.

Broker (AMQP) = Een stuk software die het AMQP protocol mogelijk maakt. De broker accepteert verbindingen met clients (software die gebruik wil maken van de queue). De exchange en de queue maken deel uit van de broker.

Ceph monitor = De monitoring service houdt maps bij van verschillende onderdelen van Ceph. De status van het cluster wordt opgeslagen samen met een map van de OSD's, placement groups en voor het CRUSH algoritme.

Compute node = Een device in een OpenStack omgeving die zorgt voor: processorkracht (CPU), geheugen, netwerk en opslag om instanties te kunnen draaien.

Controller node = Een device in een OpenStack cloud omgeving die voor aansturing zorgt.

Core componenten = De onderdelen van OpenStack die de basis van de cloud vormen. Hier kunnen nog optionele componenten aan worden toegevoegd.

Consumer (AMQP) = Een applicatie die de berichten uit de queue ontvangt.

CRUSH (Ceph) = Een algoritme van Ceph die berekend hoe data het beste (efficiënste) opgeslagen en opgehaald kan worden op de data locaties. Het algoritme probeert SPOF, performance bottlenecks en limieten aan de schaalbaarheid te voorkomen.

DAS / Direct- Attached Storage = Een storage device die direct is aangesloten op het systeem, bijvoorbeeld een HDD die direct aangesloten zit via SATA.

Ephemeral storage = Een opslagsoort waarbij de opslag “verdwijnt” als de draaiende instantie wordt opgezegd.

Exchange (AMQP) = Onderdeel van de broker. Binnen de exchange worden messages ontvangen en doorgestuurd over de queue. Dit doorsturen wordt gedaan met verschillende routing manieren, nl: direct exchange, topic exchange, fanout exchange en headers exchange. De exchange is ook wel te vergelijken met een postbode.

Flavor = Een flavor binnen OpenStack is een blauwdruk van de virtuele resources die gekozen kunnen worden voor een instance.

Floating IP = Een IP-adres die “zweeft”. Binnen OpenStack kunnen deze floating IP-adressen op commando worden gekoppeld en ontkoppeld van instances en virtuele “hardware” zoals virtuele routers.

IOPS = Staat voor input output operations per second. Het is een meetpunt waarmee opslagmedia kunnen worden gemeten.

HT-technologie = Hyper threading: één core gedraagt zich als twee threads. Het besturingssysteem ziet een verdubbeling in losse processors. Zo wordt een quad-core processor voor een besturingssysteem gezien als acht losse processoren I.P.V. vier.

LDAP = Lightweight Directory Access Protocol. Met LDAP wordt beschreven hoe informatie binnen directory services benaderd kunnen worden.

Optionele componenten = De onderdelen van OpenStack die niet vereist zijn in de basis van de cloud. Ze kunnen wel de functionaliteit van OpenStack uitbreiden.

NIC = Network interface controller: een fysieke netwerk interface in een device.

NOC = Staat voor Network Operations Center. Het is een plek (in een bedrijf) waar het netwerk in de gaten wordt gehouden. Bij i3D is het ook een afdeling binnen het bedrijf.

Node = Een device die deel uitmaakt van een bepaald netwerk.

OSD (Ceph) = Staat voor Object Storage Device. De term “OSD” is een fysieke of logische storage node. Echter wordt met “OSD” meestal de daemon bedoeld. De correcte term moet dan wel “Ceph OSD” zijn, en staat dan voor Object Storage Daemon. De daemon is verantwoordelijk voor de opslag van de objectdata en de toegang naar deze objecten.

Overcommit = De verhouding waarin er “oversell” voorkomt. Een fysieke core kan dus meerdere keren worden verhuurd. Dit geld tevens voor het RAM-geheugen. Als een systeem bijvoorbeeld 2 fysieke cores bevat zonder HT en de overcommit is 200% dan kunnen er 4 virtuele cores worden uitgedeeld aan klanten. Hier moet echter wel bij worden bedacht dat niet meer dan het maximum fysieke cores en fysiek aantal GB RAM aan één klant kan worden uitgedeeld.

Persistent storage = Een opslagsoort waarbij de opslag altijd toegankelijk (uitgaand van 100% availability) is. De draaiende instanties hebben geen effect op deze manier van opslag.

POSIX = Staat voor Portable Operating-System Interface. Een verzameling van standaarden die voor compatibiliteit (een eenheid) zorgen tussen verschillende programmeerinterfaces.

Publisher (AMQP) = Een applicatie die berichten over de queue stuurt.

Queue (AMQP) = Hier worden de messages gebufferd en overheen gestuurd naar clients. Het is te vergelijken met een brievenbus.

RADOS = Staat voor Reliable, Autonomic Distributed Object Store. Het Ceph storage cluster is opgezet volgens RADOS. RADOS bestaat uit Ceph OSD's en Ceph monitors.

Service account = Een “user” account waarmee de verschillende OpenStack services kunnen communiceren met Keystone.

Target = Domains en tenants kunnen binnen OpenStack allebei worden gezien als targets omdat aan beide roles kunnen worden toegewezen (assigned on).

Telemetry = Een proces waarbij data wordt verzameld van “moeilijk te bereiken” plekken. In OpenStack betekend dit dat de acties die gebruikers uitvoeren in de cloud worden verzameld en opgeslagen.

Tenant = Een “project” waarbinnen resources bestaan zoals: gebruikers, instances en (virtuele) netwerken.

1. OpenStack

OpenStack is een open source platform voor IaaS cloud computing. Met OpenStack is binnen een datacenter grote hoeveelheden rekeneenheden en opslag te managen. Dit alles kan grafisch aan worden gestuurd door een dashboard en door middel van commando's via de REST[1].
Het project is opgezet door Rackspace hosting en NASA. Op dit moment wordt het project gemanaged door de OpenStack foundation. Achter OpenStack schuilt een zeer actieve en grote community. Tevens dragen veel grote ondernemingen bij aan de opbouw van OpenStack. Een aantal namen van bedrijven zijn: Intel, IBM, Redhat en Hewlett packard[2].
De updates aan OpenStack worden rond zes maanden durende cycli gepland. In tabel 1 worden de updates aan OpenStack weergeven.

Afbeelding 3: In dit project wordt OpenStack gebruikt.

Release naam	Release datum	Inbegrepen componenten
Austin	21 oktober 2010	Nova, Swift
Bexar	3 februari 2011	Nova, Glance Swift
Cactus	15 april 2011	Nova, Glance Swift
Diablo	22 september 2011	Nova, Glance Swift
Essex	5 april 2012	Nova, Glance, Swift, Horizon, Keystone
Folsom	27 september 2012	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Grizzli	4 april 2013	Nova, Glance, Swift, Horizon, Keystone, Quantum, Cinder
Havana	17 oktober 2013	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Ceilometer, Heat
Icehouse	17 april 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Ceilometer, Heat, Trove
Juno	16 oktober 2014	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Ceilometer, Heat, Trove, Sahara
Kilo	30 april 2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Ceilometer, Heat, Trove, Sahara, Ironic
Liberty	15 oktober 2015	Nova, Glance, Swift, Horizon, Keystone, Neutron, Cinder, Ceilometer, Heat, Trove, Sahara, Ironic, Searchlight, Designate, Zaqar, Barbican, Manila
Mitaka	4~8 april 2016	-

Tabel 1: De releases van OpenStack[3].

Op het moment van afstuderen is Liberty[4] de laatste stabiele release van OpenStack. Vandaar dat deze versie wordt gekozen voor het project.

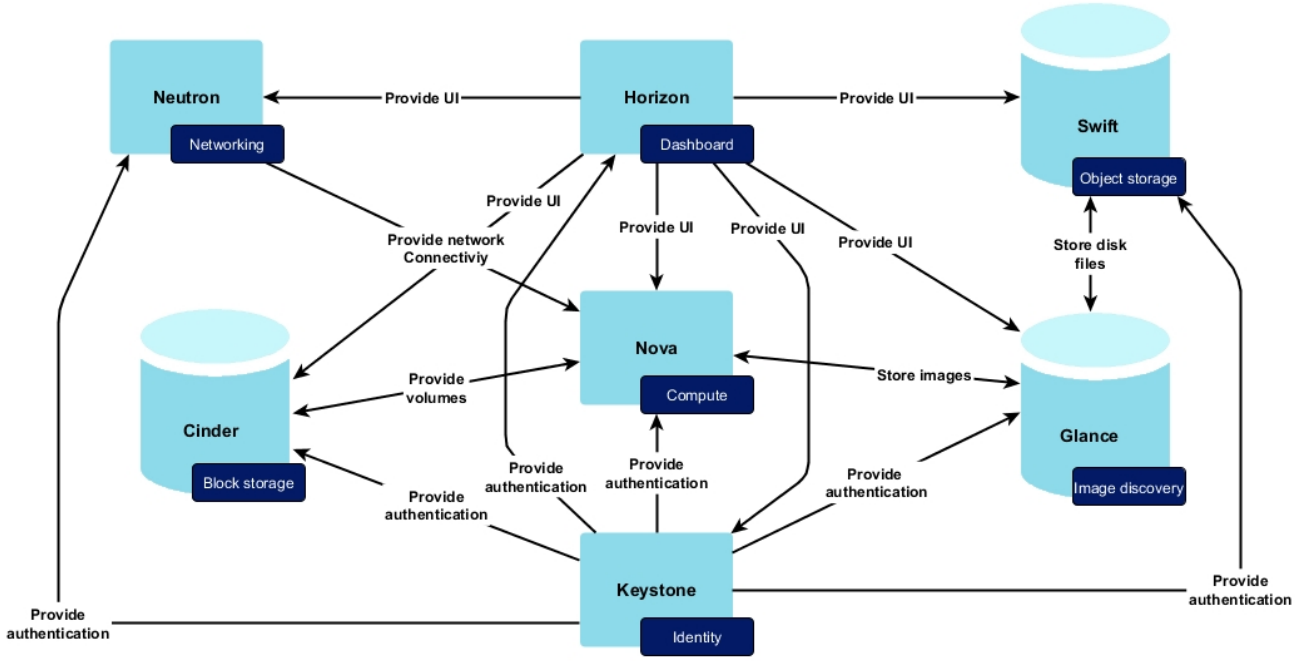
Afbeelding 4: Liberty is de twaalfde release van OpenStack.

Het platform bestaat uit een groot aantal verschillende onderdelen of componenten. Elk onderdeel kan worden opgezet om

OpenStack uit te breiden en te beheren. Het betekend dat het niet noodzakelijk is om alle onderdelen op te zetten. De componenten die in vrijwel elke OpenStack omgeving belangrijk of onmisbaar zijn om te hebben draaien zijn de core componenten. In tabel 2 worden alle (tot februari 2016) componenten van OpenStack uitgelicht. Tevens worden in afbeelding 4 de core componenten van OpenStack uitgelicht met hun onderlinge verhoudingen.

Tabel 2: De componenten binnen OpenStack en hun werking. De “core services” zijn dikgedrukt[5].

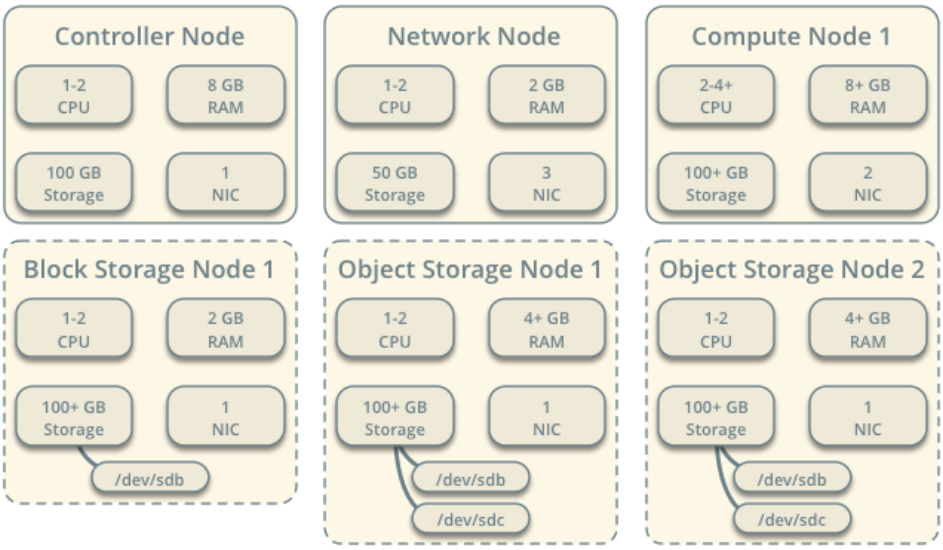
Naam component	Functionaliteit
Nova	Compute service: wordt gebruikt voor het hosten en onderhouden van de virtuele machines. Het zorgt er voor dat gebruikers virtuele machines kunnen aanmaken.
Swift	Object storage: Het wordt gebruikt voor het opslaan en weer ophalen van dataobjecten.
Glance	Dit component wordt gebruikt voor de discovery en upload van images en metadata. De gebruikers van OpenStack kunnen met deze services images die ze willen gebruiken in hun cloud discoveren of uploaden.
Horizon	Dashboard: Deze service verzorgd een web-based portal waarmee de verschillende componenten aangestuurd kunnen worden.
Keystone	Identity service: Dit is een centraal geregelde service waar alle users in staan opgeslagen. Het kan worden gebruikt bij authenticatie en authorisatie bij verschillende OpenStack services.
Neutron	Networking: Het is een component waarmee het netwerk rondom OpenStack software defined gemanaged kan worden. VLAN's, IP-adressen naar verschillende VM's en firewall management behoren onder andere tot de mogelijkheden.
Cinder	Block storage: Dit onderdeel verzorgt de persistent block storage voor de draaiende instanties.
Ceilometer	Meetservice: Met dit component kan het gebruik van de cloudservices worden gemeten. Met deze waarden kunnen de statistieken, benchmarks en eindafrekeningen worden bepaald.
Heat	Maakt OpenStack automatisch schaalbaar.
Trove	Database as a service: Met deze service kunnen klanten een relationele database gebruiken binnen OpenStack.
Sahara	Met deze service kunnen makkelijker data intensieve programma's bovenop OpenStack worden uitgevoerd.
Ironic	Met dit component kunnen bare metal machines worden gehost in plaats van virtuele machines.
Searchlight	Deze service zorgt voor indexing en zoekmogelijkheden door de OpenStack resources.
Designate	DNS as a Service.
Zaqar	Messaging service
Barbican	Security service: Met deze service kan opslag worden beveiligd, secrets (bv. passwords) worden gemanaged, sleutels worden gemanaged en certificaten worden gemanaged.
Manila	Met deze service kunnen persistent file-shares worden gemanaged



Afbeelding 5: De "core componenten" en hun verhoudingen. Horizon is geen core component maar wel benodigd[6].

1.1 OpenStack nodes (servers)

Binnen een OpenStack opstelling zijn een aantal verschillende soorten nodes te onderscheiden. Nodes zijn de hosts in een OpenStack cloud die verschillende services onderbrengen.
De nodes zijn: controller node, compute node, storage node (block storage, object storage) en network node[7]. Tevens kunnen ondersteunende diensten worden ondergebracht op één of meerdere extra nodes. Voorbeelden van extra diensten kunnen zijn: NTP, SSH jumpbox, ect. In afbeelding 4 is een overzicht van de minimale hardware om OpenStack normaal te draaien afgebeeld.



Afbeelding 4: De minimale hardware die nodig is om OpenStack te kunnen draaien[9].

Volgens de medewerkers van OpenStack is de minimale OpenStack installatie: "*The simplest architecture you can build upon for Compute has a single cloud controller and multiple compute nodes*[8]". Deze minimale opstelling is echter niet geschikt om in te zetten als productiecloud.

1.1.1 Controller node

De controller herbergt het merendeel van de services en bevat de componenten: Horizon, Keystone, Glance, Ceilometer. Tevens kan de controller node de volgende services bevatten: NTP-service, message queue en de databases voor de services.
De activiteiten van de controller node(s) liggen vooral bij de doorvoer en afhandeling van data die bestemd is voor verschillende services. Op deze nodes is per service een database actief die informatie over users en instances bijhoudt. Een message queue zorgt er voor dat de OpenStack services via remote procedure calls met elkaar kunnen communiceren, deze message queue is ook onderdeel van de controller[10].
Bij uitbreiding van het aantal users en/of van het aantal services is het vooral van belang dat de server een goede processor met veel kernen heeft en een goede netwerk doorvoer. Het aantal kernen is van belang omdat er wordt aangeraden om per Nova-API (compute) één kern vrij te hebben. De goede netwerk doorvoer is van belang omdat de controller veel verschillende services en calls moet afhandelen[11].
De controller node is echter een node die samengestelde services bevat. OpenStack raad aan om bij de opbouw van een proof of concept te starten met een controller node: "*A cloud controller is just a conceptual simplification. In the real world, you design an architecture for your cloud controller that enables high availability so that if any node fails, another can take over the required tasks. In reality, cloud controller tasks are spread out across more than a single node.*"[12] Na de opbouw van het proof of concept zou als een volgend project de services op de controller node verspreid kunnen worden over meerdere andere nodes: de controller node is dus niet voor elk project een vereiste.

1.1.2 Compute node

Deze nodes verzorgen de hypervisors en zorgt daarmee dat de virtuele machines draaien. Één compute heeft ook één hypervisor, er kunnen meerdere VM's of instances draaien op één compute node. Ook kunnen de

nodes worden gezien als CPU- en memorypools. Tevens kan een compute node lokale storage aanbieden aan instances. Ze bevatten een networking agent die de virtuele machines kan koppelen met virtuele netwerken[13]. De compute node hebben over het algemeen de hoogste eisen aan hardware. De hoogste systeemeisen van deze nodes komen van het virtueel draaien van instances. De load op de compute node wordt bepaald door de manier dat de instances geschaald zijn, de overcommit van een compute node en de activiteit van de users op die node[14].

1.1.3 Block storage node

Block storage nodes kunnen binnen OpenStack worden gebruikt voor het toewijzen van nieuwe opslag aan draaiende instances. OpenStack gebruikers kunnen op deze manier makkelijk een nieuw volume toewijzen aan de draaiende virtuele machine[14].
De block storage nodes bevatten de disks die block storage mogelijk maken voor de virtuele machines.

1.1.4 Object storage/proxy node

De object storage nodes kunnen worden gebruikt voor het opslaan van over het algemeen grotere files, zoals: Images, back-ups en archieven als objecten[15]. Bij een object storage oplossing vormen servers een storage backend waarbij data als objecten opgeslagen kunnen worden[16].
Bij block storage en object storage is het aan te raden om na te denken over een dedicated datalijn tussen compute en storage. Opslag kan veel verkeer genereren en het is meestal niet wenselijk dat andere gebruikers hier last van hebben. Ook hebben beide opslag methoden ruime hoeveelheden aan storage capaciteit nodig als er geen gebruik wordt gemaakt van een backend[14].

1.1.5 Network node

Als er gebruik wordt gemaakt van een network node dan betekent dit automatisch dat het OpenStack Neutron component wordt gebruikt. Tenants kunnen zelf virtuele netwerk resources regelen. De tenants kunnen private netwerken met eigen IP-adressen instellen binnen hun IaaS. Het neutron component kent in de Liberty versie van OpenStack ook al vele plug-ins. Met deze plug-ins is het mogelijk dat de tenants zelf QoS, firewalls, VPN's en intrusion detection instellen[17].

Een OpenStack omgeving kan ook functioneren zonder network node. Binnen zo'n instelling zullen de netwerk functionaliteiten minimaal zijn en is er ook geen sprake van gevirtualiseerde netwerkonderdelen voor tenants. De netwerk functionaliteiten worden dan verzorgt door de compute nodes zelf. De service wordt legacy of Nova networking genoemd[17].

Network design: Vanuit OpenStack wordt aangeraden om minimaal de volgende netwerken te implementeren[18]: 1. Een publiek netwerk (internetverbinding), 2. Een verbinding tussen de compute nodes en de network nodes zodat de klanten kunnen verbinden (tunnel), 3. Een verbinding tussen de controllers en alle nodes (dus ook de network nodes), hiermee kan onder andere de messagequeue communiceren en kunnen de fysieke nodes worden gemanaged.

1.2 Operating system

OpenStack kan worden geïnstalleerd op vrijwel alle Linux distributies, en zelfs op Windows. Bij veel operating systems zullen de packages die nodig zijn voor OpenStack zelf moeten worden gecompileerd. Nu zijn er ook een aantal distributies die standaard de OpenStack (repositories) packages verstrekken[19].
De operating systems die OpenStack zelf aanbeveelt en die ook allemaal OpenStack packages verstrekken zijn:

Ubuntu 14.04 LTS, Red Hat Enterprise Linux 7, CentOS 7, SUSE Linux Enterprise Server 12 en Debian 8[19].

Bij de software selectie moet er rekening worden gehouden met de ondersteunde hypervisors in de gebruikte distributie. De verschillende hypervisors worden door OpenStack in groepen ingedeeld. Aan de groepen kan worden afgeleid hoe intens de hypervisor drivers worden getest in combinatie met OpenStack.
Er bestaat een groep A: deze drivers worden volledig getest en ondersteund. De enige driver die staat ingedeeld in deze groep is voor KVM en QEMU.
Ook is er een groep B: deze drivers worden niet volledig getest. De hypervisor drivers die OpenStack in deze groep heeft ingedeeld zijn voor: XenServer, vSphere en Hyper-V[20].

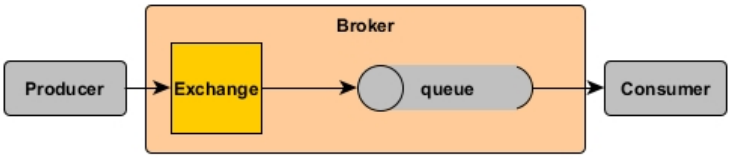
1.3 Database

Binnen OpenStack wordt veel gebruik gemaakt van databases. Nu ondersteunt OpenStack veel verschillende soorten databases. De meest gebruikte databases die OpenStack ondersteunt zijn: MariaDB, MySQL en PostgreSQL[21].

1.4 Message queue

In OpenStack is een messagequeue vereist om de cloud te laten functioneren. Een messagequeue kan worden gezien als een stuk middleware die het mogelijk maakt om verschillende processen, applicaties of systemen met elkaar te kunnen laten praten ongeacht het ontwerp van de processen, applicaties en systemen, dit gebeurd via remote procedure calls[22].
Binnen OpenStack wordt een messagequeue gebruikt om de verschillende componenten met elkaar te laten communiceren. De componenten zijn geschreven door verschillende programmeurs. Met de messagequeue kunnen de componenten van verschillende programmeurs toch met elkaar communiceren. Tevens is het zo dat door het gebruik van een messagequeue OpenStack modulair wordt[22].

OpenStack gebruikt RabbitMQ als broker om het Advanced Message Queuing Protocol (AMQP) te implementeren[23]. AMQP is een protocol die het gemixt gebruik van verschillende messaging systems mogelijk maakt. Hiermee wordt de interoperabiliteit van de message queue vergroot.
Het AMQP werkt met brokers, dit zijn zogezegd de message “servers”[24].
De exchange is een onderdeel van de broker. Binnen de exchange worden aan de hand van bindings de messages op de queue gezet. Bindings zijn de regels die gebruikt worden bij het distribueren van messages[23].



Afbeelding 5: De globale werking van AMQP[25].

Met de queue worden berichten van publishers naar comsumers verstuurd. Ook zorgt de queue voor buffering van messages.Deze queues worden ook wel eens “oneindige buffers” genoemd. De publisher produceert het bericht en de consumer vangt het bericht op en verwerkt het.

De exchange kan de messages op verschillende manieren routeren naar de queues[23], nl:

- Direct exchange: de messages worden bij de queues afgeleverd aan de hand van routing keys. Aan een routing key kan de exchange zien waar de messages naartoe moeten.
- Fanout exchange: Kan worden vergeleken met een broadcast. De exchange stuurt de message naar alle queues die hij kent.
- Topic exchange: Hier stuurt de exchange de messages op basis van het publisher-subscriber principe.

Hier worden routing keys gebruikt en de queues zijn op een specifieke manier gekoppeld (binded) aan de exchange.
- Headers exchange: De messages worden aan de hand van verschillende headers gerouteerd.

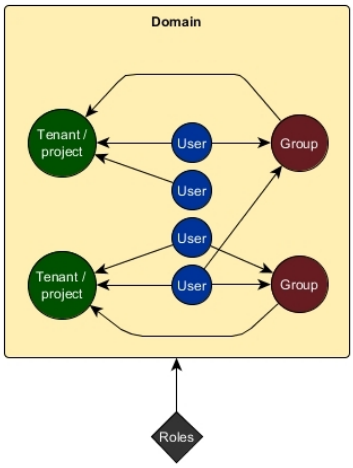
1.5 OpenStack identity (Keystone)

Aangezien een cloud een “verzamelplaats” van users en data is heeft OpenStack een grote behoefte aan een identity service. Het component die de identity service mogelijk maakt onder OpenStack heet Keystone. Keystone zorgt voor authenticatie, autorisatie en een service catalogus. De andere services die OpenStack aanbiedt moeten samenwerken met Keystone om authenticatie en autorisatie mogelijk te maken[26], dit betekend dat op het moment dat een user een request stuurt naar een OpenStack service dit component eerst bij Keystone moet controleren of de gebruiker deze request mag uitvoeren.

Keystone moet ook weten welke services aanwezig zijn in de betreffende OpenStack installatie. Om dit bij te houden heeft Keystone een service catalog. Om deze catalogus bij te houden moeten alle gebruikte services binnen OpenStack geregisteerd worden. In het service catalog staan de verschillende URL's van de OpenStack services, zogenaamde endpoints. De endpoints bestaan uit een public URL, internal URL en admin URL.[26] Simpel gezegd zorgt de Keystone service catalog voor routeren van users en applicaties naar de goede OpenStack services.

1.5.1 Entiteiten

Binnen OpenStack/Keystone bestaan de entiteiten users, groups, tentants (ofwel projects), roles en domains[26]. Users zijn net als in andere omgevingen digitale representaties van personen, systemen of services. Deze users kunnen verschillende roles aannemen.



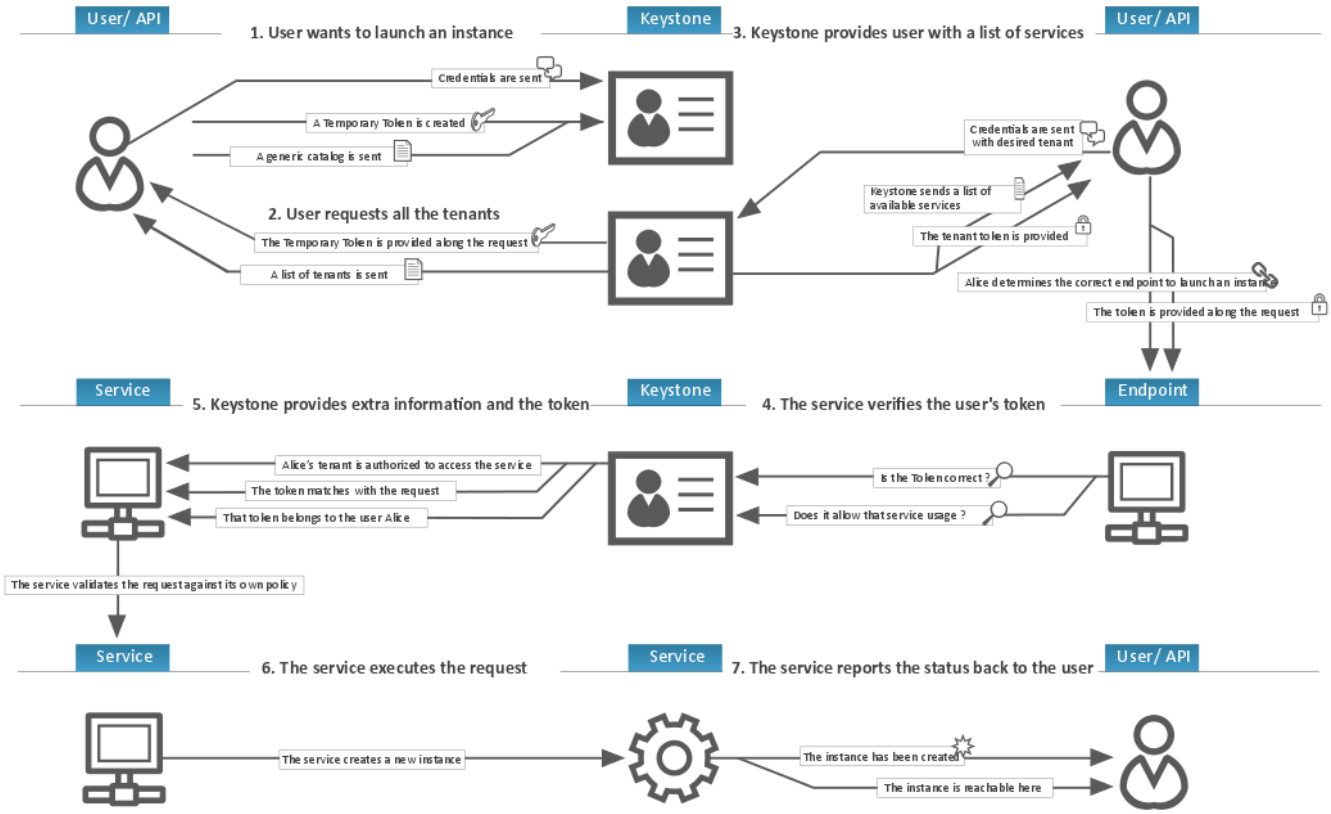
Afbeelding 6: Roles zijn globaal uniek. Tenants, users en groups zijn onderdeel van een domain.

Een role zorgt voor het toewijzen van rechten en privileges aan users. Met deze rechten kunnen users een specifieke taak uitvoeren. Roles zijn globaal aanwezig binnen de OpenStack cloud, ze zitten dus niet in een domain.
Bij de toewijzing van een role zijn er drie onderdelen van belang, nl: De actor (user of group), de role zelf en het target (tenant of domain)[26].

Een tentant is te vergelijken met een groep of een project. Het wordt gebruikt om resources en users van elkaar te scheiden. Users kunnen door middel van roles toegang gegeven worden tot één of meerdere tenants. Als er op de meest simpele manier gekeken wordt naar Keystone dan kan er worden gezegd dat het een registratie is waar wordt bijgehouden wie of wat toegang heeft tot deze projecten[26].
Een domain is te vergelijken met een onderneming. Tussen verschillende domains kunnen de users en user groups elkaar ook niet zien. Tevens kunnen er in verschillende domains ook dezelfde namen worden gebruikt voor users, groups en tenants[26]. Formeel gezien is een domain een collectie van specifieke users, groups en tenants.

1.5.2 OpenStack authenticatie

De authenticatie binnen OpenStack is schematisch weergeven in afbeelding 7.



Afbeelding 7: Authenticatie binnen OpenStack[27].

2.5.3 Keystone backends

Met Keystone kunnen verschillende backends worden gebruikt[26].

- Identity backend. Keystone slaat de identiteiten in deze backend op. De identiteiten zijn dus de users en groups: de actors zogezegd. Op basis van SQL of LDAP.
- Assignment backend. Hier worden de roles en role-assignments opgeslagen. Op basis van SQL.
- Token backend. OpenStack. Users kunnen ook niet verder werken als de tokens niet continu worden gecontroleerd door Keystone. Op basis van SQL, en er kan ook aanvullend gebruik worden gemaakt van Memcached voor meer performance.
- Catalog backend. Hier worden de endpoints van alle OpenStack services opgeslagen. Dit is van belang zodat Keystone weet waar alle componenten te vinden zijn. Dit backend is een SQL-backend.
- Resource backend. In dit backend wordt de data opgeslagen over de verschillende tenants en domains die aanwezig zijn binnen OpenStack. Deze backend is een SQL-backend.

2.6 OpenStack image service (Glance)

De Glance service is vanaf de tweede officiële OpenStack release aanwezig. Het component zorgt voor discovery van images, registratie van images en het ophalen van images.[28] Met Glance kan een user dus een snapshot van een virtual machine of een schijf maken, en vervolgens zal Glance deze opslaan in de backend. Gebruikers van de cloud kunnen images (en snapshots) uploaden naar de Glance backend. Deze images

kunnen private of public toegankelijk zijn. Vervolgens kunnen de opgeslagen images door de user worden gebruikt om VM's mee op te starten in OpenStack compute[28]. Glance kan dus worden gezien als een repository voor images binnen OpenStack.

2.7 OpenStack compute (Nova)

OpenStack Nova kan worden gezien als de core van OpenStack. Dit component zorgt voor toegang tot (compute) resources. Het component houdt grote aantallen VM's bij. Tevens managed de Nova service de draaiende VM's[28].
Binnen Nova is een scheduler aanwezig die de load verdeel over de verschillende compute nodes, het component bepaalt waar (op welke node) een VM het beste kan komen te draaien.[29] Het proces voor het uitkiezen van de meest geschikte node bestaat uit twee stappen. De eerste stap is filteren: hier wordt gekeken welke node het minst is beladen of er wordt random een host uitgekozen.[29] De tweede stap is weighting: Omdat er verschillende soorten servers kunnen worden gebruikt voor compute nodes wordt er in deze stap gekeken naar wat de user vraagt en welke server dit kan bieden. Servers krijgen weights toegewezen bij het opzetten. Server met lagere performance krijgen een hogere weight[29].

2.8 OpenStack networking

Met Neutron kan er gebruik worden gemaakt van virtuele netwerken en hebben users ook de mogelijkheid tot het gebruik van uitgebreide services zoals LbaaS (load balancing), VPN en FwaaS (Firewall)[30]. De Neutron service beheerd ook de IP-adressen binnen OpenStack.
Met Neutron bestaat iedere VM en tenant in een virtueel netwerk. Dit virtuele netwerk is een layer 2 netwerk(segment), vergelijkbaar met VLAN's bij fysieke netwerken. Ieder segment is dus een broadcast domain. Binnen de virtuele netwerken bestaan subnetten (lokale range, class A, B of C). Elk virtueel netwerksegment heeft een subnet toegewezen gekregen. Elk VM heeft een (virtuele) port. Met deze port worden VM's in een subnet gehangen. Een port kan worden gezien als een aansluitpunt aan een virtuele switch. De VM wordt aan de port aangesloten en de port zit aangesloten aan het virtuele netwerk(segment)[30].

2.9 OpenStack block storage

Het OpenStack project biedt storage aan in verschillende vormen. Zo zijn er ephemeral, block en object storage. De block storage wordt aangeboden door het Cinder project binnen OpenStack. Bij block storage worden er block volumes aangeboden aan de klanten/VM´s. Een block wordt door de OpenStack VM's gezien als een storage device (een extra harde schijf) of direct attached storage. Een block kan op hetzelfde moment ook maar aan één instantie zijn gekoppeld[31].
De verbinding tussen Nova en Cinder kan op basis van iSCSI, Fibrechannel of network file system zijn. Het backend van Cinder kan uit veel verschillende formaten bestaan, waaronder: LVM en NFS.[32]

2.10 OpenStack object storage

OpenStack heeft een “native” component voor object storage, nl: Swift. Swift is een component die schaalbaarheid en redundantie uitstraalt. Het object storage systeem is zo opgezet dat het voor lange tijd grote BLOB's data (objecten) kan opslaan. Deze objecten zijn redundant opgeslagen binnen de nodes. Mocht er een node uitvallen dan repliceert Swift de data van de uitgevallen node naar een nieuwe node[32].

Swift kan gebruikt worden als “reliable storage”. Zo kan Cinder back-ups maken met Swift als backend. Ook kunnen users grote bestanden uploaden naar Swift.[32] De bestanden binnen Swift worden gezien als object. Objecten hebben allemaal een uniek ID. Aan de hand van dit ID kan een user zijn object weer ophalen of

verwijderen. De object storage wordt aangesproken met HTTP-commands, zoals: GET en PUT[33].

Binnen Swift bestaat er een hiërarchie boven de objecten, nl: Accounts → Containers → Objects[33].
Accounts zijn de bovenste laag van hiërarchie. Een Swift account komt overeen met de betreffende tenant in OpenStack.
Onder de accounts komen containers. Containers kunnen worden gezien als “namespaces” binnen een Swift account. Er kunnen “oneindig veel” containers worden aangemaakt binnen een account. Binnen de containers kunnen ook ACL's worden aangemaakt om toegang tot objecten te ontzeggen.
Tenslotte komen er onder de containers de objecten. Binnen de standaard instellingen van OpenStack is de maximale grootte van een object 5GB.
Bij de “directory” indeling binnen Swift komt deze hiërarchie ook naar voren, de opbouw van een path is: /[Swift versie]/[account]/[container]/[object].

2.11 OpenStack Benchmarking

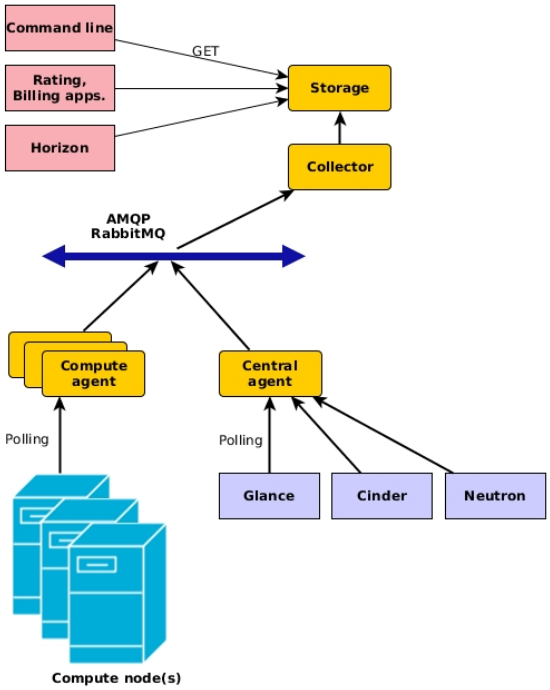
Rally is een benchmarking tool binnen OpenStack. Het kan realistische load genereren om zo een actieve cloud te simuleren. Met deze load kan performance en scalability worden getest[34].
Als pakket is Rally eigenlijk alleen een “schil” bovenop Tempest: de officiele test suite van OpenStack. Echter door de complexiteit van Tempest werd dit niet vaak gebruikt. Met Rally als schil is Tempest veel beter “bestuurbaar”[34]. Met Rally kan OpenStack gedeployed, geverifieerd en gebenchmarkt worden.

2.12 OpenStack telemetry

Bij cloud diensten is het van belang dat verleende services kunnen worden gemeten. Verleende service (In bijvoorbeeld CPU-kracht/aantal cores of tijd) zijn immers de onderdelen die geld in het laatje brengen.
Binnen OpenStack zijn er verschillende projecten die elk een gedeelte van telemetry op zich nemen. Zo is er Ceilometer: Dit project verzorgt het meten van het verbruikt binnen de verschillende OpenStack componenten[35]. Hierna is er nog het AODH-project. Dit project is een uitbreiding op Ceilometer: het zorgt voor een “alarm” op het moment dat er aan vooraf gestelde criteria zijn voldaan. Een voorbeeld kan zijn dat een gebruiker over zijn verbruik heen gaat[35].

Het metering onderdeel wordt door Ceilometer op zich genomen: het verbruik van klanten wordt gemeten. Rating is het omzetten van verbruiksdata in bruikbare data (variabelen) voor een rekening. Billing is het omzetten van meerdere verbruiksvariabelen in één prijs onderaan een rekening.
De rating en billing onderdelen kunnen door verschillende projecten uitgevoerd worden. Een aantal voorbeelden van rating en billing programma's zijn:

- Cloudkitty (een rating system)
- Cyclops (3th-party rating en billing)



Afbeelding 8: De architectuur van Ceilometer.

2.13 OpenStack QoS , limits en quota

Op OpenStack is QoS ook te implementeren. De QoS is te implementeren door op elke service apart in te stellen. Zo is QoS in Neutron bijvoorbeeld in te stellen door neutron-qos commando's te gebruiken.

2.13.1 Flavor extra specs

Een stuk gemakkelijker dan QoS in elke service apart te implementeren is de mogelijkheid om QoS op te nemen in de instance flavors. Bij het aanmaken van een flavor moet de naam, aantal vcores, aantal GB RAM en aantal GB storage worden aangegeven. Naast deze parameters kunnen er extra parameters genaamd “extra specs” worden toegewezen aan een flavor.[36]
De extra specs zijn opties die de QoS van een instance gebaseerd op de betreffende flavor bepalen. Zo zijn er extra specs in te stellen voor:

- CPU
- Memory
- Disk I/O
- Disk I/O of bytes per seconde
- Network bandwidth

Deze limits, zoals OpenStack de extra specs ook noemt, zijn bedoeld voor libvirt, de virtualisatiemanager. Met deze extra specs worden de hypervisor en Nova aangestuurd. Alle extra specs worden dus uitgevoerd op de compute node waar de betreffende instance op draait[37]. De limits voor de CPU worden uitgevoerd met Cgroups. De limits voor de lokale storage worden uitgevoerd met DiskIOLimits. De limits voor netwerkbandbreedte worden uitgevoerd met tc[37].

2.13.2 Cinder QoS

Om de Cinder volumes van QoS te voorzien kunnen de cinder-qos commando's worden gebruikt. De QoS-opties zijn hetzelfde als de opties voor “disk_tuning” in de disk-limits[38]. De opties zijn:

- total_bytes_sec: Totaal aantal bytes die de instance per seconde in totaal kan lezen/schrijven.
- read_bytes_sec: Aantal bytes per seconde die de instance kan lezen van het block device.
- write_bytes_sec: Aantal bytes per seconde die de instance naar het block device kan schrijven.
- total_iops_sec: Totaal I/O operaties per seconde die de instance kan uitvoeren.
- read_iops_sec: Aantal I/O read operaties per seconde die de instance kan uitvoeren op het block device.
- write_iops_sec: Aantal I/O write operaties per seconde die de instance kan uitvoeren op het block device.

2.13.3 Neutron Qos

De QoS in OpenStack networking, Neutron, is een apart opgezet project. In de QoS van Neutron bestaan twee “nieuwe” entiteiten, nl: “Policies” en “Rules”[39]. De “rules” zijn andere entiteiten dan de “rules” uit Keystone. Een policy is een collectie rules die gekoppeld kan worden aan een virtuele port of een heel virtueel netwerk binnen Neutron. De rules zijn de QoS-restricties waaruit policies zijn opgebouwd.
In de rules zijn alleen de “max-kbps” en “max-burst-kbps” aan te geven voor inbound traffic en outbound

traffic[39]. De Neutron service zet de rules om in restricties op de OpenvSwitch virtuele switch.

2.13.5 Oversell

Het oversellen of overcommitten van compute nodes is binnen cloud omgevingen vrij normaal. Door het feit dat één server plek kan bieden aan meerdere klanten kunnen resources ook meerdere keren worden verkocht. De resources worden op dat moment dus gedeeld. Overcommit in de cloud is een balans vinden tussen winst aan de ene kant en performance aan de andere kant. Een enkele instance in OpenStack kan echter niet meer vcores hebben dan het fysieke aantal CPU-cores[40].
In OpenStack is er al standaard een overcommit ratio ingesteld. Deze verhouding ligt op 16:1 voor vcores en 1,5:1 voor RAM.

3. Ceph

Ceph is onderzocht omdat dit werd gevraagd vanuit i3D. Uiteraard zijn er nog meer storage backends voor OpenStack, zoals: Gluster en Sheepdog. Deze andere oplossingen zijn niet onderzocht omdat dit niet binnen de scope van het project valt (projectgrens).

3.2 Wat is Ceph

Ceph is een storage backend voor cloud oplossingen: een alles in één storage oplossing op een platform van object storage. Ceph is zo opgebouwd dat het block storage, object storage en file system storage bij elkaar kan aanbieden. Deze verschillende soorten storage kunnen tegelijk worden aangeboden aan veel verschillende hosts en/of instanties[41].

Deze services worden aangeboden door drie componenten:

- Ceph OSD
- Monitor
- MDS

De object storage device (OSD) daemon zorgt voor de opslag, replicatie, recovery en herbalaceren van data over de storage nodes. Ook kijkt deze daemon of de andere replicatie nodes nog heartbeat hebben. Mocht dit niet het geval zijn dan kan er direct worden gestart met het repliceren van de node naar een andere functioneren de node. Binnen Ceph zijn drie replicatienodes (1 datanode en 2 replicatie) standaard, het kan echter ook met twee (1 datanode en 1 replicatie)[42].

De monitoring service houdt maps bij van verschillende onderdelen van Ceph. De status van het cluster wordt opgeslagen samen met een map van de OSD's, placement groups en voor het CRUSH algoritme[42].

De Ceph metadata server (MDS) slaat metadata op om het Ceph filesystem te ondersteunen. Met de metadata kunnen standaard commando's worden uitgevoerd op een instance (die op Ceph FS draait) zonder hoge load. Als bijvoorbeeld het commando *ls* uitgevoerd wordt zal dit zonder de metadata enorme load genereren. Dit komt door de manier van opslag (object storage) die het storage cluster gebruikt. De metadata server wordt ook gebruikt om een directory en filestructuur op het object storage backend te leggen[42].

3.3 Ceph vereisten

Het Ceph project is zo opgebouwd dat het normaal kan draaien met ongespecialiseerde hardware. Dit geldt uiteraard wel bij lage tot middelmatige load[43].

Het OSD-proces heeft minimaal een dual-core processor nodig maar quad core is uiteraard beter. Het berekenen met CRUSH, data repliceren en een kopie bijhouden van het cluster map vergt allemaal veel processor kracht.
De OSD's hebben met normale bezigheden niet veel RAM nodig. Ceph raadt zelf 500MB per instantie van de OSD daemon. Het is echter wel zo dat de OSD daemons tijdens recovery veel meer geheugen nodig hebben, dit kan oplopen tot 1GB RAM per 1TB storage per daemon[43].

De monitoring service heeft een veel minder krachtige processor nodig. Het bijhouden van de maps heeft niet veel processorkracht nodig. Een dual core processor kan hier makkelijk volstaan. Voor het geheugen wordt 1GB

RAM per daemon aangeraden: de data van de monitoring service moet snel kunnen worden verspreid. De hoeveelheid storage die de monitoring service nodig heeft ligt rond de 10GB per daemon[43].

De metadata service heeft behoefte aan krachtige processoren: ze moeten namelijk dynamisch hun load redistribueren. Ceph adviseerd hier minimaal een krachtige quad-core processor, maar liever krachtiger.
Net als de monitoring service moet de metadata service zijn data snel kunnnen verspreiden. Ook hier wordt 1GB per daemon aangeraden, liever meer. De metadata service heeft voor storage maar 1MB per daemon nodig[43].

Het gebruik van VLAN's tussen de servers wordt door Ceph aangeraden[43]. Hiermee kunnen de fysieke verbindingen worden afgeschermd van elkaar wat de secirity enigszins verbeterd. Hier is echter wel vereist dat de NIC's en de switches VLAN's (IEEE 802.1q) ondersteunen.

3.3.1 OS platform

Voor Ceph kan als platform worden gekozen uit een aantal Linux besturingssystemen. Deze betreffen: CentOS, Debian, Fedora, Red hat en Ubuntu[44]. Voor de laatste release (Infernalis) zijn CentOS en Ubuntu volledig getest in combinatie met Ceph, dit is te zien in tabel 3.

Tabel 3: De Linux releases waarmee Ceph is getest voor de laatste Ceph release (Infernalis). B = basic unit testen, I = installation en functionality testen, C = Uitgebreide functionality en stress testen[44].

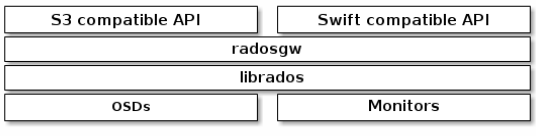
Distribution	Release	Kernel	Testing
CentOS	7	Linux-3.10.0	B, I, C
Debian	8.0	Linux-3.16.0	B, I
Fedora	22	Linux-3.14.0	B, I
RHEL	7	Linux-3.10.0	B, I
Ubuntu	14.04	Linux-3.13.0	B, I, C

3.4 Vergelijking Ceph storage en OpenStack storage

Ceph storage en OpenStack block storage (Cinder) zijn niet te vergelijken. Ceph is namelijk een storage backend voor OpenStack en Cinder is een component die een backend nodig heeft. Dit betekend dat een Ceph storage backend aan ëën of meerdere OpenStack block storage nodes gehangen kan worden.
OpenStack Swift en Ceph kunnen wel worden vergeleken. Beide projecten bieden object storage clusters.

3.4.1 Ceph storage oplossingen

Alle Ceph storage oplossingen zijn gebaseerd op het RADOS storage system. RADOS is een synoniem voor het storage cluster[45]. Het storage cluster bestaat uit de OSD service(s)/daemon(s) en de monitoring service(s)/daemon(s) [45]. Het minimale aantal nodes voor het storage cluster is drie, nl: 1 monitoring node en 2 OSD nodes (voor replicatie) [45].



Afbeelding 8: De Ceph object gateway: om objects op te slaan in een Ceph backend[47].

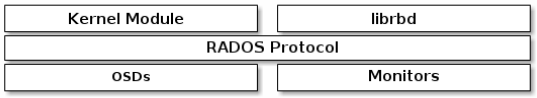
Object storage

Bij object storage gebeurd data uitwisseling via de Ceph Object Gateway[46]. Aan deze gateway kan een Swift API worden aangesloten. Als onderlaag wordt gebruik gemaakt van Ceph OSD's en de monitoring service. Een

overzicht van de Ceph object gateway staat in afbeelding 8.

Block storage

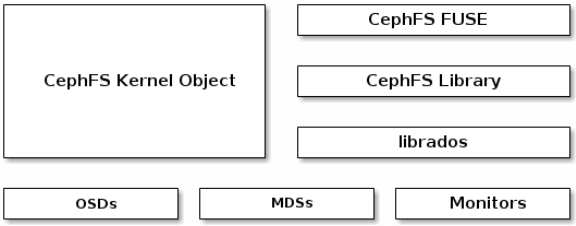
Block storage kan worden gebruikt in Ceph via het RADOS protocol. Via de kernel of via de library *librbd* kan dit protocol worden bereikt[48]. OpenStack Cinder kan het Ceph block device direct benaderen. Vanuit OpenStack Nova wordt er een verbinding gelegd tussen een instance (hypervisor) en het Ceph block device[48]. Een overzicht van deze verbinding is te zien in afbeelding 9.



Afbeelding 9: Het Ceph storage cluster[49].

Ceph filesystem

Ceph heeft ook een eigen filesystem. Het filesystem gebruikt hetzelfde storage cluster als het block device en object storage, tevens kan het worden gebruikt voor bijvoorbeeld legacy applicaties[50]. Bij gebruik van het filesystem moet wel een metadata server worden geïnstalleerd. Deze metadata server zorgt voor het opslaan van directory structuren en andere structuren om het Ceph filesystem POSIX-compatible te maken. De metadata is nodig omdat Ceph alle data als objecten opslaat in het cluster[50]. Een overzicht van deze verbinding is te zien in afbeelding 10.



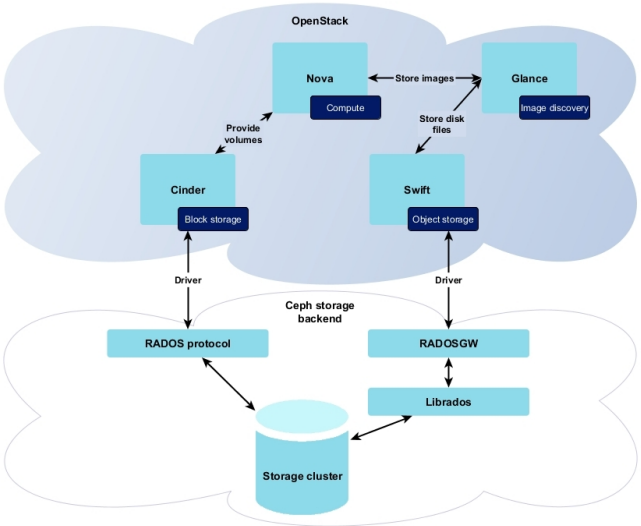
Afbeelding 10: Het Ceph filesystem[51].

Het filesystem is echter nog wel in experimentele staat. Er wordt aangeraden om het nog niet in te zetten voor productie omgevingen[50].

In afbeelding 11 is te zien hoe Ceph aan de OpenStack nodes gekoppeld kan worden.

3.4.3 Verschil storage oplossingen

Het opzetten van de combinatie OpenStack en Ceph wordt goed ondersteund en zelf aangemoedigd door beide partijen[52]. Tevens wordt deze setup in veel bronnen besproken.
Volgens de developers van Ceph is er een hoge availability voor de storage te behalen en het is te gebruiken op low-end systemen[53].
Het nadeel zou echter wel zijn dat bij het groeien van de OpenStack cloud ook het gebruik van Ceph groeit, en hiermee ook de load. Het is geen rare gedachte dat na een korte tijd meer performance vereist is van de oude servers. Als de servers dan al maximaal geüpgraded zijn zou er over moeten worden gestapt op andere servers. Dit houd in dat er een grote investering vereist is op performance drops tegen te gaan.



Afbeelding 11: OpenStack met Ceph storage backend.

Aan de andere kant biedt OpenStack ook oplossingen voor storage, nl: Swift en Cinder. Als deze oplossingen worden gecombineerd ontstaat er ook een krachtige storage solution. Zo kan Swift binnen een Openstack cloud bijvoorbeeld worden gebruikt voor het opslaan van grote datasets zoals: snapshots en Cinder volume back-ups. De Cinder service zorgt voor toewijzen van block storage aan de OpenStack instances en kan instances zelfs laten booten van block volumes.
De Cinder- en Swift componenten kunnen beide ook “lokaal” worden opgezet. Bij swift betekend lokaal opzetten dat er een LVM backend wordt gebruikt. Bij Swift betekend lokaal opzetten dat er geen 3th-party backend wordt gebruikt, maar een lokaal storage network wordt opgezet.

Storage component	Object storage	Block storage	File level storage	Store snapshots	Store Cinder volume back-ups
Ceph	V	V	V	V	V
Swift	V			V	V
Cinder		V			
Local file system			V	V	

Tabel 4: Vergelijking tussen verschillende storage componenten. Binnen OpenStack worden Swift en Cinder meestal samen gebruikt.

Ceph en OpenStack Cinder kunnen niet worden vergeleken. Dit komt door het feit dat Cinder een besturende eenheid is die met een interne storage backend (LVM) werkt of een externe 3th-party backend nodig heeft om te functioneren, Ceph is daarentegen een storage backend.

OpenStack Swift en Ceph kunnen echter wel worden vergeleken. Het eerste punt wat opvalt is performance. Uit een performance vergelijking blijkt dat Swift last kan krijgen van een bottleneck: de proxy server. In Swift wordt minimaal één proxy server gebruikt om de data te verdelen over de nodes. Veel load zal dus komen te liggen op deze proxy server.
Bij Ceph is het zo geregeld dat het CRUSH algoritme bottlenecks probeert te voorkomen met slimme load distibution[6]. Hierbij moet wel worden bedacht dat bij het groeien Ceph al snel meer resources nodig heeft, services zoals OSD, monitoring en CRUSH moeten de omgeving constant in de gaten houden.

Ceph maakt gebruik van OSD-daemons om opslag aan te sturen, Ceph raad aan om per fysieke schijf één OSD daemon in te stellen[54]. Swift maakt gebruik van een “ring”. Deze ring is bij iedere node bekend en op die manier weten de proxy nodes waar de verschillende object storage nodes en volumes te vinden zijn.

Het aantal servers dat nodig is voor een minimale productieomgeving is ook verschillend tussen Swift en Ceph. Bij Ceph wordt er aangeraden om binnen een minimaal productiecluster negen fysieke servers neer te zetten, de berekening is hierbij: 3 monitoring nodes + 5 storage nodes + 1 RGW (RADOS GateWay)[54].
Bij Swift wordt een minimum aantal servers van zes aangeraden, de berekening hierbij is als volgt: 1 object proxy server + 5 object storage servers[55]. Hier moet wel een kanttekening worden geplaatst dat er bij Swift minimaal nog een node bij moet, namelijk de proxy servers die redundant moet worden uitgevoerd, Als de proxy node uitvalt dan kan niemand meer bij de Swift service.

Als het storage cluster over meerdere locaties wordt verdeeld kan Ceph volgens Christian Huebner zeer ernstige nadelen vertonen:

> *...Ceph writes only synchronously and requires a quorum of writes to return successfully.*
> *With those issues in mind, let’s imagine a cluster with two regions, separated by a thousand miles, 100ms latency, and a fairly slow network connection. Let’s further imagine we are writing two copies into*

> *the local region and two more to the remote region. Now the quorum of our four copies is three, which means the write request is not going to return before at least one remote copy is written. It also means that even a small write will be delayed by 0.2 seconds, and larger writes are going to be seriously hampered by the throughput restriction.*
>
> *On the other hand, Swift in the same two-region architecture will be able to write locally first and then replicate to the remote region over a period of time due to the eventual consistency design. Swift also requires a write quorum, but the write_affinity setting can configure the cluster to force a quorum of writes to the local region, so after the local writes are finished the write returns a success status.[56]*

Als de Ceph uit meerdere regions gaat bestaan kan de consistency een probleem gaan vormen. Daar tegenover kan Swift het af met eventual consistency bij een setup over meerdere regions.

Volgens Christian Huebner kan binnen Ceph de dataveiligheid in het gedrang komen omdat de RADOS clients op hetzelfde netwerk draaien als het niet geëncrypte replicatieverkeer[56].

Uiteraard is het van belang bij de keuze tussen de storage oplossingen om te kijken naar de use case: een low-cost cloud oplossing op oude servers. De toekomste klanten die de OpenStack cloud service aanvragen weten bij het tekenen van het SLA dat de service op sommige punten kwalitatief minder kan zijn dan de high-end Red cloud binnen i3D. Zo kan de performance minder zijn omdat er wordt gewerkt met 1Gb/s lijnen (I.P.V. 10Gb/s bij de Red cloud) en met minder krachtige servers. De scalability van de storage zal ook niet zo hoog zijn als in de Red cloud omdat de servers maar vier HDD's kunnen huisvesten.
Omdat er in dit project gebruik gemaakt moet worden van servers met lagere performance zal er gekozen worden voor OpenStack Swift. De redenen hiervoor zijn:

- Om een minimale productieomgeving op te zetten met Swift zijn er minder servers nodig dan met Ceph, nl: 7 tegen 9.
- Ceph wordt al snel “hongerig naar resources”, dit komt door de services die Ceph als een consistent bottleneck voorkomend geheel laten werken. Deze services moeten constant de omgeving in de gaten houden. Bij swift is dit minder het geval, veel load komt bij de object proxy servers te liggen. Als de performance te laag wordt kunnen dus eerst de proxy servers geüpgraded worden.
- De eventual consistency kan een voordeel worden als er gebruik gemaakt gaat worden van meerdere regions binnen de OpenStack cloud, nl: betere performance, er hoeft geen reply te worden afgewacht.

4 Hardware inventarisatie

Om het proof of concept op te zetten is oude hardware opgeleverd die niet meer werd gebruikt in het datacenter. De initiële hardware oplevering voor mijn opdracht bestaat uit de volgende stukken hardware:

- 44 HP Proliant DL120 G6
- 2 Dell Powerconnect 5548
- Server rack
- 250GB harde schijven

Er was een inventarisatie nodig omdat alle servers verschillende setups hadden. Tevens is het volgens de OpenStack organisatie van belang voor een zo foutloos mogelijke opbouw om de hardware zo gelijk mogelijk te houden. Veel servers waren zelfs niet uitgerust geheugen of harde schijven.

4.1 Servers

Het proof of concept dient te worden opgezet met de HP Proliant DL120 G6. In de oplevering zijn er 44 van deze servers aan mijn project toegewezen. Alle servers zijn met de inventarisatie geoptimaliseerd, dit betekent dat de servers uitgerust zijn met 8 GB RAM en een harde schijf van 250GB. De processor die in de server zat bleef behouden.
De nummers van de servers die opgeleverd zijn voor het proof of concept zitten in een range van oplopende nummers, nl: Server 1986 ~ Server 2000 en Server 4404 ~ Server 4405.

4.2 Processors

Al deze servers bevatte een Intel Xeon X3430. Deze processor is een non-HT 4 cores processor met 2,4Ghz en Intel VT-x virtualisatie acceleratie.

Bij het virtualiseren op een Nova compute node kunnen fysieke processor cores worden geprovisioneerd. Hoe meer cores een server bevat hoe meer er uitgedeeld (en overgeprovisioneerd) kunnen worden. De X3430 processor bevat vier cores zonder HT-technologie (dubbele threads). Het besturingssysteem op deze nodes ziet dus vier losse processors.
Met Intel VT-x bieden de processors hardwarematige ondersteuning bij virtualisatie. Er is met VT-x een redelijke snelheidswinst te behalen over volledig softwarematig virtualiseren[57].

4.3 Hard disks

De DL 120 G6 servers hebben zelf vier drive bays aan de voorkant van het device[58]. Hier kunnen 3,5 inch harde schijven in worden gefaciliteerd.
Het uitrusten van de servers is gedaan in de inventarisatie.
Na de inventarisatie is gebleken dat er al 28 harde schijven met een capaciteit van 250 GB in het rack aanwezig waren. Als er meer harde schijven nodig waren dan zou dit gevraagd kunnen worden aan de magazijnbeheerder. Er mochten zo veel harde schijven worden gebruikt als nodig. Alle harde schijven hebben een SATA interface en een rotatiesnelheid van 7200RPM.

4.4 RAM

Omdat het bij een cloud omgeving draait om virtualisatie is de RAM-grootte belangrijk. De maximale grootte voor

een DL120 server die volledig is uitgerust met unbuffered DIMM's is 8GB (4 x 2GB). Met Buffered DIMM's kan er tot 16GB worden gebruikt[58], deze zijn echter niet beschikbaar, daardoor zijn de servers met 8GB RAM uitgerust.

4.5 Switches

Om het proof of concept op te zetten is er één switch opgeleverd. Deze switch is van het type Dell Powerconnect 5548. De switch bezit 48 poorten met gigabit snelheid (1000BASE-T).
De switch is managed en ondersteund alle belangrijke niet-proprietary protocollen, zoals: IEEE 802.1Q, STP en IGMP. Tevens kan met het IEEE 802.1p protocol op elke port prioriteit worden aangegeven. Hiermee kan de QoS worden gemanaged[59]. Als blijkt dat één switch te weinig ruimte biedt kan een tweede switch worden geïnstalleerd. Er moet echter in eerste instantie van uit worden gegaan dat er een enkele switch is.

Afbeelding 12: Een Dell Powerconnect 5548

4.6 Rack

De hardware was bij de oplevering allemaal gemount in een dedicated rack. Dit rack is aanwezig in het datacenter van i3D. Specificaties van de racks zijn onder andere:

- Geïntegreerde koeling van de vloer naar boven.
- Slot op de rackdeur.
- 32 ampére stroomsterkte kan worden geleverd.
- 1 Gb netwerksnelheid.

4.7 Het interne netwerk van i3D

Binnen dit project is het ook van belang dat het interne netwerk van i3D is onderzocht. Om een OpenStack installatie te laten draaien moet er namelijk ook duidelijk zijn hoe het directe netwerk er uitziet waar de installatie in komt. Naar het interne netwerk van i3D is wel onderzoek gedaan, echter zal dit niet uitgebreid worden gerapporteerd aangezien dit nadelig kan werken voor i3D. Het volledige interne netwerk is ook verder niet heel belangrijk om in het verslag te belichten. Wel worden hieronder de punten beschreven waar op moet worden gelet bij het opzetten van het OpenStack proof of concept.

- Voor het project is een range van globale IPv4-adressen gereserveerd. Deze range zal verder worden gespecificeerd in het infrastructuurontwerp.
- Binnen i3D worden VLAN's gebruikt binnen het netwerk. Dit betekent dat het OpenStack project één of meer eigen VLAN's krijgt. Er is in ieder geval één VLAN die het project aansluit op het openbare netwerk van i3D (toegang naar “buiten”).
- Op het fysieke vlak lopen er in de toegewezen suite naar elk rack een 10Gb-verbinding via “top-of-rack” verbindingen.

Bronvermelding

[1] OpenStack. “overview of the OpenStack project”. Available: http://www.openstack.org/software/. [Accessed: 27-01-2016].

[2] OpenStack. “OpenStack community”. Available: http://www.openstack.org/community/. [Accessed: 27-01-2016].

[3] OpenStack. “OpenStack releases”. Available: http://releases.openstack.org/. [Accessed 28-01-2016].

[4] OpenStack. “OpenStack releases”. Available: http://www.openstack.org/software/liberty/. [Accessed: 28-01-2016].

[5] OpenStack. “OpenStack components”. Available: http://www.openstack.org/software/. [Accessed: 28-01-2016].

[6] Sharone Zitzman. “OpenStack components”. Available: http://getcloudify.org/img/blog/openstackdiagram.jpg. [Accessed: 01-02-2016].

[7] OpenStack. “OpenStack nodes”. Available: http://docs.openstack.org/openstack-ops/content/compute_nodes.html. [Accessed: 01-02-2016].

[8] OpenStack. “OpenStack example architecture”. Available: http://docs.openstack.org/openstack-ops/content/example_architecture.html. [Accessed: 01-02-2016].

[9] OpenStack. “example architecture with neutron networking hw”. Available: http://docs.openstack.org/juno/install-guide/install/apt/content/ch_overview.html#example-architecture-with-neutron-networking-hw. [Accessed: 01-02-2016]

[10] OpenStack. “Cloud controller design”. Available: http://docs.openstack.org/openstack-ops/content/cloud_controller_design.html. [Accessed: 01-02-2016].

[11] OpenStack. “Example architecture”. Available: http://docs.openstack.org/liberty/install-guide-ubuntu/overview.html#example-architecture. [Accessed: 02-02-2016].

[12] OpenStack. “Designing for cloud controllers and cloud management”. Available: http://docs.openstack.org/openstack-ops/content/cloud_controller_design.html. [Accessed: 02-02-2016].

[13] OpenStack. “Example architecture”. Available: http://docs.openstack.org/liberty/install-guide-ubuntu/overview.html#example-architecture. [Accessed: 02-02-2016].

[14] OpenStack, OpenStack operations guide. OpenStack foundation, 2014. [E-book] Available: http://docs.openstack.org/openstack-ops/openstack-ops-manual.pdf [Accessed: 03-02-2016].

[15] OpenStack. “OpenStack storage decisions”. Available: http://docs.openstack.org/openstack-ops/content/storage_decision.html. [Accessed:03-02-2016].

[16] Yadin Porter De Leon, Tony Piscopo. "Object Storage versus Block Storage: Understanding the Technology Differences". Available: http://www.druva.com/blog/object-storage-versus-block-storage-understanding-technology-differences. [Accessed: 03-02-2016].

[17] OpenStack. “OpenStack network design”. Available: http://docs.openstack.org/openstack-ops/content/network_design.html. [Accessed: 03-02-2016].

[18] OpenStack. “Technical considerations”. Available: http://docs.openstack.org/arch-design/generalpurpose-technical-considerations.html#designing-network-resources. [Accessed 04-02-2016]

[19] OpenStack. “How to get OpenStack”. Available: https://wiki.openstack.org/wiki/Get_OpenStack. [Accessed: 04-02-2016].

[20] OpenStack. “Hypervisor support matrix”. Available: https://wiki.openstack.org/wiki/HypervisorSupportMatrix. [Accessed: 04-02-2016].

[21] OpenStack. “OpenStack SQL-databases”. Available: http://docs.openstack.org/liberty/install-guide-ubuntu/environment-sql-database.html. [Accessed: 04-02-2016].

[22] OpenStack. “OpenStack messagequeue”. Available: http://docs.openstack.org/developer/nova/rpc.html. [Accessed: 05-02-2016].

[23] OpenStack. “AMQP and Nova”. Available: http://docs.openstack.org/developer/nova/rpc.html. [Accessed 05-02-2016].

[24] O.S. Tezer . “AMQP message queue”. Available: https://www.digitalocean.com/community/tutorials/an-advanced-message-queuing-protocol-amqp-walkthrough. [Accessed: 05-02-2016].

[25] Peter Ledbrook. “Understanding AMQP, the protocol used by RabbitMQ”. Available: http://blog.springsource.com/wp-content/uploads/2010/06/rabbit-basics.png. [Accessed: 05-02-2016].

[26] Steve Martinelli, et al., Identity, Authentication, and Access Management in OpenStack . Sebastopol: O’Reilly Media, Inc., 2015.

[27] OpenStack. “The Keystone identity manager”. Available: http://docs.openstack.org/kilo/install-guide/install/apt/content/figures/2/figures/SCH_5002_V00_NUAC-Keystone.png. [05-02-2016]

[28] Ken Pepple, Deploying OpenStack. Sebastopol: O’Reilly Media, Inc., 2011.

[29] OpenStack. “OpenStack configuration reference: scheduling”. Available: http://docs.openstack.org/liberty/config-reference/content/section_compute-scheduler.html. [05-02-2016]

[30] James Denton, Learning OpenStack Networking (Neutron), 2014. [E-book] Available: https://www.safaribooksonline.com/library/view/learning-openstack-networking/9781783983308/index.html [Accessed: 08-02-2016].

[31] OpenStack. “OpenStack storage decisions”. Available: http://docs.openstack.org/openstack-ops/content/storage_decision.html. [Accessed: 08-02-2016].

[32] OpenStack. “Cinder support matrix”. Available: https://wiki.openstack.org/wiki/CinderSupportMatrix. [Accessed: 09-02-2016].

[33] Joe Arnold. “OpenStack Swift”. Available: https://www.safaribooksonline.com/library/view/openstack-swift/9781491903841/. [Accessed: 10-02-2016].

[35] OpenStack. “OpenStack Rally”. Available: http://rally.readthedocs.io/en/latest/overview.html. [Accessed: 21-03-2016].

[36] OpenStack. “OpenStack telemetry”. Available: https://wiki.openstack.org/wiki/Telemetry. [Accessed:21-03-2016].

[37] OpenStack. “Instance resource quota”. Available: https://wiki.openstack.org/wiki/InstanceResourceQuota. [Accessed: 21-03-2016].

[38] Sebastien Han. “OpenStack, Ceph RBD and QoS”. Available: https://www.sebastien-han.fr/blog/2013/12/23/openstack-ceph-rbd-and-qos/. [Accessed: 22-03-2016].

[39] Livnat Peer, Moshe Levi, Irena Berezovsky. “QoS Neutron N00bie”. Available: https://www.openstack.org/assets/Uploads/QoS-Neutron-N00bie.pdf. [Accessed: 23-03-2016].

[40] OpenStack. “OpenStack operations guide: compute nodes”. Available: http://docs.openstack.org/openstack-ops/content/compute_nodes.html. [Accessed: 23-03-2016].

[41] Ceph. “Ceph storage”. Available: http://ceph.com/ceph-storage/. [Accessed: 10-02-2016]

[42] Ceph. “Intro to Ceph”. Available: http://docs.ceph.com/docs/master/start/intro/. [Accessed: 10-02-2016]

[43] Ceph. “Hardware recommendations”. Available: http://docs.ceph.com/docs/master/start/hardware-recommendations/. [Accessed: 10-02-2016]

[44] Ceph. “OS recommendations”. Available: http://docs.ceph.com/docs/master/start/os-recommendations/. [Accessed: 10-02-2016]

[45] Ceph. “Ceph storage cluster”. Available: http://docs.ceph.com/docs/master/rados/. [Accessed: 10-02-2016]

[46] Ceph. “Ceph object gateway”. Available: http://docs.ceph.com/docs/master/radosgw/. [Accessed: 11-02-2016]

[47] Ceph. “Ceph object gateway components”. Available: http://docs.ceph.com/docs/master/_images/ditaa-50d12451eb76c5c72c4574b08f0320b39a42e5f1.png. [Accessed: 11-02-2016]

[48] Ceph. “Ceph block device”. Available: http://docs.ceph.com/docs/master/rbd/rbd/. [Accessed: 11-02-2016]

[49] Ceph. “Ceph block device components”. Available: http://docs.ceph.com/docs/master/_images/ditaa-

dc9f80d771b55f2daa5cbbfdb2dd0d3e6dfc17c0.png. [Accessed: 12-02-2016]

[50] Ceph. “Ceph filesystem”. Available: http://docs.ceph.com/docs/master/cephfs/. [Accessed: 12-02-2016]

[51] Ceph. “Ceph filesystem components”. Available: http://docs.ceph.com/docs/master/_images/ditaa-b5a320fc160057a1a7da010b4215489fa66de242.png. [Accessed: 12-02-2016]

[52] OpenStack. “Ceph: The De Facto Storage Backend for OpenStack”. Available: https://www.openstack.org/summit/openstack-summit-hong-kong-2013/session-videos/presentation/ceph-the-de-facto-storage-backend-for-openstack. [Accessed: 12-02-2016]

[53] Ceph. “Hardware Recommendations”. Available: http://docs.ceph.com/docs/hammer/start/hardware-recommendations/. [Accessed: 22-02-2016]

[54] Ceph. “Zero To Hero Guide : : For CEPH CLUSTER PLANNING”. Available: http://ceph.com/planet/zero-to-hero-guide-for-ceph-cluster-planning/. [Accessed: 22-02-2016].

[55] OpenStack. “Before you begin”. Available: http://docs.openstack.org/icehouse/install-guide/install/apt/content/before-you-begin-swift-install.html. [Accessed: 23-02-2016].

[56] Christian Huebner. “Ceph vs Swift – An Architect’s Perspective”. Available: https://www.mirantis.com/blog/cephvsswiftarchitectsperspective/. [accessed 24-02-2016]

[57] Intel. “Intel Xeon processor X3430 specifications”. Available: http://ark.intel.com/nl/products/42927/Intel-Xeon-Processor-X3430-8M-Cache-2_40-Ghz. [accessed 23-02-2016]

[58] HP (14-11-2011). “Dl120 G6 manual”. Available: http://images10.newegg.com/UploadFilesForNewegg/itemintelligence/HP/13504_na1400508550568.pdf. [accessed 25-02-2016]

[59] Dell (2013). “Powerconnect 5548 specifications”. Available: http://www.dell.com/downloads/global/products/pwcnt/en/powerconnect-5500-series-spec.pdf. [accessed 25-02-2016]

Bijlage IX: Plan van aanpak

Plan van aanpak

Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low-cost cloud oplossing te creëren bij i3D

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Stagebedrijf: i3D
Plaats: Rotterdam

Datum: 18-01-2016

Versiebeheer

Versie	Datum	Wijziging
0.1	18-01-2016	Layout PvA
0.2	22-01-2016	Conceptversie PvA
0.3	03-03-2016	Spelling en grammatica bijgewerkt.
1.0	30-03-2016	Document klaar gemaakt voor oplevering, uiteindelijke versie

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

Inhoudsopgave

1. Opdrachtomschrijving

1.1 Opdrachtgever, opdrachtnemer en begeleiders

Opdrachtgever: Rick Sloot, i3D
Opdrachtnemer: Jeroen Holtkamp
Eerste examinator: Jan Dirk Schagen
Tweede examinator: Marinus Maris

1.2 Belanghebbenden

Binnen het project is Rick Sloot de enige belanghebbende die behoeften en eisen op kan geven voor de opstelling. Dit komt omdat het een afgeschermd project is. Andere medewerkers binnen i3D hebben geen belang bij dit project.
Andere medewerkers kunnen in een volgend project wel belanghebbend worden. Dit gebeurd als het een project wordt waar een invoering als productiecloud het doel is.

1.3 Probleemstelling

Naast de high end Red cloud op basis van Microsoft Azure heeft i3D ook behoefte aan een laag geprijsde cloud oplossing. Er zullen genoeg klanten zijn die wat minder eisen stellen aan hun hosting en/of er minder geld voor over hebben. Aan de andere kant zijn er veel oudere servers binnen i3D die op dit moment niet commercieel ingezet (kunnen) worden. Dat komt voornamelijk door het ontbreken van de juiste platforms voor deze servers. Ook ontbreekt de juiste toepassing nog. In een eerder onderzoek is er onderzocht wat voor open source cloud platform het beste zou passen binnen i3D. De conclusie van dit onderzoek was dat OpenStack de beste keuze was om te implementeren binnen i3D.

Als deze twee problemen en het voorgaande onderzoek bij elkaar op worden geteld dan komen de hoofdpunten waar dit project om draait tevoorschijn:

- I3D heeft naast alle high-end hosting oplossingen ook behoefte aan een low-end cloud oplossing op basis van OpenStack.
- De toegewezen oude servers moeten dienen als platform.
- Er is geen kennis van de eventuele winst die met zo'n soort cloud oplossing gemaakt kan worden.

De problemen hierbij zijn:

- De specificaties van de hardware zijn niet bekend.
- Er is niet genoeg kennis van OpenStack binnen i3D aanwezig.
- Er is geen kennis van Ceph aanwezig binnen i3D.
- Voor de afstudeerder is het interne netwerk van i3D niet bekend.
- Er is geen kennis van de eventuele winst die met zo'n soort cloud oplossing gemaakt kan worden.

Ook heeft de opdrachtgever graag dat specifiek wordt gekeken naar een storage oplossing genaamd: “Ceph”. Deze storage oplossing zou dan eventueel gebruikt kunnen worden in het proof of concept en in de uiteindelijke cloud oplossing.

1.4 Doelstelling

De doelstelling van de opdracht is het verwezenlijken van een IaaS proof of concept gebaseerd op het OpenStack platform. Het proof of concept moet met zo min mogelijk kosten worden opgezet. Bij elke keuze moeten de kosten in gedachten worden gehouden. Als hardware moeten de oude servers worden gebruikt die i3D aanlevert voor de opdracht.
Om te bewijzen dat de oplossing wel of niet winstgevend is zal er een winstberekening gemaakt moeten worden. Aan de hand van de winstberekening kan aan het eind van de opdracht worden afgelezen of er mogelijk winst zit in de oplossing.
Het proof of concept moet na het eindigen van het project reproduceerbaar zijn. Om hier aan te voldoen zal er een handleiding geschreven worden. Deze handleiding beschrijft alle acties die tot het proof of concept hebben geleid.

2. Aanpak

In dit hoofdstuk wordt de aanpak beschreven waarmee het project is doorlopen. In het plan van aanpak staan alle onderdelen van de aanpak beschreven. In dit hoofdstuk zijn de onderzoeksvragen beschreven samen met de verantwoording van de projectmethode en de project planning.

2.1 Onderzoeksvragen

Hier onder worden de onderzoeksvragen beschreven. Uit de onderzoeksvragen kan de aanpak van het project worden afgeleid.

2.1.1 Hardware specificaties

Dat de specificaties van de hardware niet bekend zijn is een cruciaal probleem. Dit probleem strekt zich uit van de interne hardware van de servers tot de bekabeling in het rack.
De volgende hardwarespecificaties moeten bekend worden: De bekabeling, de specificaties van de aangeleverde servers (CPU, max geheugen, etc), de specificaties van de switches en de aantallen van de verschillende stukken hardware. De vragen die gesteld moeten worden bij het onderzoek naar de aanwezige hardware is:

- Welke stukken hardware krijg ik opgeleverd?
- Welke specificaties hebben deze stukken hardware?
- Hoe zijn deze stukken hardware georganiseerd en gepositioneerd binnen i3D/het datacenter?

2.1.2 OpenStack kennis

Binnen i3D niet genoeg kennis aanwezig van OpenStack. Voor dit project heeft er een project gedraaid waarbij is gekeken welke cloud oplossing het beste bij i3D zou passen. Tevens is bij het voorgaande project een minimale opstelling op basis van OpenStack opgezet. De persoon die het voorgaande project heeft uitgevoerd geldt binnen mijn project als expert op het gebied van OpenStack.
Het is echter van groot belang dat OpenStack wordt onderzocht. Met dit onderzoek kunnen eisen die i3D stelt theoretisch worden getoetst aan de literatuur: misschien zijn sommige eisen onredelijk of helemaal niet haalbaar. Natuurlijk is het ook van belang dat het duidelijk is hoe OpenStack werkt. Tevens moet het jargon dat bij OpenStack hoort duidelijk zijn. De vragen die gesteld moeten worden bij het onderzoek naar OpenStack zijn:

- Wat is OpenStack?
  - Uit welke onderdelen bestaat OpenStack?
  - Waar dienen de onderdelen, waar OpenStack uit bestaat, voor?
  - Hoe functioneren de onderdelen waar OpenStack uit bestaat?
- Hoe werkt OpenStack en hoe wordt het opgezet?
- Wat zijn de (systeem)eisen die OpenStack stelt aan de infrastructuur?
- Waar kunnen eventuele knelpunten (in de performance) voorkomen bij het groeien van de cloud?

2.1.3 Ceph kennis

Het storage platform Ceph is qua kennis ook nog niet bekend bij i3D. Het is net als OpenStack wel bekeken door i3D. De opdrachtgever vond Ceph zo veel potentie hebben dat daar ook een onderzoek naar uitgevoerd moet worden. In dit onderzoek moet ook worden gekeken naar de werking van de storage oplossing.

- Wat is Ceph?
- Hoe werkt Ceph?
- Wat zijn de (systeem)eisen die Ceph stelt aan de infrastructuur?
- Hoe is Ceph te koppelen met OpenStack?
- Is Ceph van toegevoegde waarde voor OpenStack binnen dit project?

2.1.4 Interne netwerk

Het interne netwerk van i3D is bij de afstudeerder niet bekend. Dit is echter wel van belang om te onderzoeken in wat voor omgeving het proof of concept zou moeten werken. Niet het gehele i3D netwerk hoeft duidelijk te zijn. Wat wel duidelijk moet zijn is:

- Hoe ziet de directe omgeving van het proof of concept er uit?
- Wat zijn de protocollen op het i3D netwerk waar het proof of concept mee te maken krijgt?
- Wat zijn de belangrijke overige faciliteiten op het i3D netwerk waar het proof of concept mee te maken krijgt?

2.1.5 Winst

De mogelijke winst die gemaakt kan worden met zo'n soort oplossing is niet bekend. Om servers commercieel in te zetten is het belangrijk om te weten of er ook winst gemaakt kan worden. Deze winst hangt van veel factoren of “variabelen” af. Sommige factoren kunnen pas effectief worden onderzocht op het moment dat het proof of concept functioneel is.
Met de winstberekening is er meer zekerheid over de levensvatbaarheid, een opvolgend project kan met meer zekerheid zakelijk gerechtvaardigd worden.

De vragen die hierbij gesteld worden zijn:

- Welke variabelen zijn van toepassing op het proof of concept?
- Kan er theoretisch winst worden gemaakt met deze oplossing?

2.1.6 Projectgrenzen

Binnen i3D is lang niet alle kennis van OpenStack aanwezig. Het is echter wel duidelijk dat OpenStack een zeer groot project is. Er kan dus nooit binnen de tijd van een aantal maanden een proof of concept worden opgezet die de gehele low-cost cloud oplossing verantwoord. In samenspraak met de opdrachtgever zijn er projectgrenzen gesteld aan de omvang van de opdracht. Oplossingen voor de eisen die i3D stelt zullen uiteraard wel worden verantwoord. Tevens worden er aanbevelingen gedaan mocht een oplossing niet (volledig) geïmplementeerd kunnen worden.

2.2 Op te leveren producten

Binnen het project zijn een aantal producten die opgeleverd dienen te worden:

- Plan van aanpak
- Analyse/onderzoeksrapport
- Gedetailleerde winst- en kostenberekening

- Interviewverslag/meetingverslag
- Handleiding
- Globaal ontwerp (architectuur)
- Gedetailleerd ontwerp (infrastructuur)
- Testplan / testrapport
- Proof of concept

2.3 Projectgrenzen

Aan het project worden grenzen gesteld om overschrijding van de planning te voorkomen. Met de grenzen wordt ook voorkomen dat er onderzoek en werk wordt verricht die niet nodig is voor het project.
Bij i3D kan het eens voorkomen dat er werkzaamheden voorkomen die niet binnen de scope van het project vallen. Die werkzaamheden kunnen worden aangepakt om extra ervaring met de techniek en met de omgeving binnen i3D op te doen. Er valt dan te denken aan meehelpen met bekabelen van servers/racks bij een grote oplevering.

De eerste grens is de functionaliteit van het proof of concept. Deze mag basaal worden gehouden. Zo hoeft er bij het aantonen van de werking van het proof of concept maar één image functioneel te zijn. Ook is het zo dat er maar één soort operating system functioneel hoeft te zijn.

De tweede grens gaat over het netwerk. Uiteraard moeten de gebruikte routing- en switching protocollen worden beschreven. Echter is het niet de bedoeling om veel te veel de diepte in te gaan over deze onderwerpen.

De derde grens gaat over de opbouw van het netwerk. Bij behoefte aan apparatuur die op dat moment nog niet is toegewezen moet het "huismerk" van i3D.net/SmartDC worden aangehouden of oude onderdelen uit de voorraad worden gebruikt. Er hoeft dus geen vergelijking tussen verschillende soorten devices te worden uitgevoerd om te zien welke het beste zou zijn in het netwerk.

De vierde grens gaat over onderzoek naar alternatieven van Ceph. Vanuit i3D is gevraagd om specifiek te richten op Ceph als eventuele (alternatieve) opslagoplossing. Dit houd in dat er geen onderzoek gedaan hoeft te worden naar alternatieven voor Ceph, buiten OpenStack storage oplossingen om.

De mogelijkheid bestaat dat sommige onderdelen voor de pilot/productie versie van de OpenStack cloud (mogelijk volgend project) te groot zullen zijn om te implementeren in het proof of concept. Als dit voorkomt dan wordt er een aanbeveling geschreven over dit betreffende component.

2.4 Randvoorwaarden

Om het project goed te kunnen doorlopen worden er een aantal voorwaarden gesteld waar i3D aan moet voldoen. Zo zal er een werkplek beschikbaar moeten zijn voor de afstudeerder. Tevens moet er een plek in het datacenter worden toegewezen die dedicated is voor deze opdracht. De plek moet ook duidelijk gereserveerd staan om te voorkomen dat de servers worden verwijderd of herbruikt.

2.5 Middelen

De middelen waarmee wordt gewerkt binnen de opdracht zijn de volgende:

Literatuur:

- Internetbronnen
- Website: “Official OpenStack website”. https://www.openstack.org/
- Website: “Official Ceph website”. http://ceph.com/
- Boek: Jeroen Horlings (2011). “Cloud computing”. Den Haag: Sdu uitgevers.
- Boek: Bert Hedeman, Gabor Vis van Heemst, Hans Fredriksz (2009). “Projectmanagement op basis van PRINCE2 editie 2009”. Zaltbommel: Van Haren uitgevers.
- Boek: PMI Netherlands Chapter (2009). “A Guide to the Project Management Body of Knowledge (PMBoK Guide) 4th Edition - Nederlandstalige uitgave”. Zaltbommel: Van Haren uitgevers.
- Document: “ASI-rapport”

Hardware:

- Werkplek: PC met Windows 10 en laptop met Linux Mint
- 40 servers met interne opslag
- één switch (op aanvraag twee switches)
- Eventueel extra apparatuur
- Memory stick voor back-up documenten

Software:

- yEd: voor het maken van diagrammen
- Google drive: online documenten opslag
- LibreOffice

3. Methode

Binnen een project moet er altijd volgens een methode worden gewerkt om het overzicht te behouden. In de eerste fase van het project is dan ook een methode gekozen. Eerst is er een aantal methodieken uitgekozen die mogelijk gebruikt zouden kunnen worden. Vervolgens zijn deze methodieken langs het project gelegd en gekeken wat het project nodig had.

Uit de voorselectie zijn de volgende methodieken gekomen:

- PRINCE2 [1]
- Waterval (overlappend: “sashimi”) [2]
- ASI [3]
- PMBok [4]

PRINCE2

PRINCE2 staat voor PRojects IN Controlled Environment, het is een procesmatige methodiek voor projectmanagement. Het is een projectmethode die in veel soorten omgevingen kan worden gebruikt, het is echter het meest van toepassing in ICT-projecten. PRINCE2 omvat naast projectmanagement ook het management van mensen en middelen. Er bestaan binnen een PRINCE2 project zeven principes waar ten alle tijden aan voldaan dient te worden. Als niet aan alle zeven principes wordt voldaan dan kan er niet worden gesproken van een PRINCE2 project. Er bestaan binnen PRINCE2 zeven thema's die uiteen worden gezet in zeven processen. De zeven processen lopen van project start-up (SU) tot project einde (CP).

Waterval/sashimi

De watervalmethode is een methode die is afgeleid van de cascades in een waterval, het is een lineaire ontwikkelmethodiek. Aan het eind van elke fase worden er vooraf gedefinieerde producten opgeleverd. In de originele watervalmethode zullen de fases en producten goed moeten worden gepland, want er kan niet meer worden teruggekeerd naar een eerdere fase zoals in een iteratieve methode. Het voordeel van deze aanpak is dat de voortgang goed gevolgd kan worden. De fasering in de watervalmethode betreft: definitie, ontwerp, gedetailleerd ontwerp, ontwikkeling, test, invoeren en onderhoud.
Om te voorkomen dat er een fout wordt ontdekt die is gemaakt in een voorgaande fase en vervolgens het gehele project over moet doen kan er een overlap worden gebruikt. De watervalmethode met een overlap ingebouwd heet het Sashimi model. Met de overlap van de fases kan er dus een fase terug worden gegaan bij het ontdekken van een fout.

ASI

ASI is een lineaire ontwikkelmethodiek voor technische infrastructuren. Binnen ASI worden vier fases onderkent: Definitiefase, Architectuurfase, Ontwerpfase en ontwikkelfase. Na het afsluiten van een fase onder ASI kan er niet worden teruggekeerd naar een afgesloten fase, net als in de watervalmethode. Wel is er sprake van iteratief gedag binnen de fases.

PMBok

PMBok staat voor Project Management Body of Knowledge, het is een procesmatige methodiek voor projectmanagement. PMBoK is een framework in het PMP (Project Management Professional) programma.

Vooral in Amerika is het een veelgebruikte methode. Het is een framework met best practices. De best practices zijn opgemaakt aan de hand van vroegere projecten die gefaald of succesvol waren. Project managers kunnen binnen hun project kiezen welke best practices het meest geschikt zijn. Er worden echter geen activiteiten opgelegd of voorgesteld.
De PMBoK gaat uit van vijf processen: initiatie, planning, uitvoering, beheer en afsluiting. Elke fase van het project doorloopt deze processen. PMBoK wordt gebruikt bij projecten die zijn opgezet om een uniek product of dienst op te leveren. Het grote nadeel van PMBoK is dat het een framework is en niets voorschrijft. Als project manager moet je zelf de best practices uitzoeken en gebruiken. Dit vereist een hoog niveau aan kennis en ervaring vanaf het begin van het project.

Tabel 1: De kenmerken van verschillende methodieken zijn vergeleken

	Beschikbare literatuur	**Toegankelijkheid**	**Flexibiliteit in de methode**	**Project toepassing**	**Schaalbaarheid**	**Totaal (M=25)**
PRINCE2	4	2	4	3	4	17
Waterval/ sashimi	4	5	3	4	4	20
ASI	2	4	2	3	4	15
PMBok	4	2	4	2	3	15

In de vergelijking hierboven worden een aantal kenmerken tussen de methodieken vergeleken. De betekenis van de verschillende kenmerken is het volgende:

Beschikbare literatuur

Om in te lezen over de gebruikte methode zal er literatuur aanwezig moeten zijn die geraadpleegd kan worden. Bij PRINCE2 en PMBok is dit voornamelijk literatuur waar voor betaald moet worden. Uiteraard kunnen er boeken worden geleend bij de bibliotheek.
De waterval/sashimi methodiek is goed verantwoord op internet. Hier zijn veel bronnen over te vinden. Als laatste is de ASI-methode zeer slecht tot niet te vinden op het internet. Het ASI-rapport is de enige literatuur die er te vinden is.

Toegankelijkheid

De toegankelijkheid van een methode is hoe gemakkelijk deze methode eigen gemaakt kan worden. Uiteraard mag dit geen groot obstakel zijn. Echter wordt een project een stuk lastiger als het onderzoek naar de methodiek een lange tijd in beslag neemt.
De PRINCE2 methode is in Nederland binnen IT bedrijven vaak gebruikt. Na een aantal stages is deze methode dus ook wel langs gekomen. Echter kost het inlezen in deze methode enige tijd, zeker als er geen PINO project van gemaakt mag worden.
De methodiek PMBok is van origine Amerikaans en in Nederland nauwelijks gebruikt in kleinere bedrijven. Tevens moet voor deze methode veel onderzoek worden verricht. Eigenlijk moet er veel verstand en ervaring opgebouwd zijn alvorens het te gebruiken. Dit is zeker zo omdat er zelf best practices moeten worden gekozen door de project managers.
De watervalmethode is zeer toegankelijk omdat er hard wordt voorgeschreven wat wanneer gedaan moet worden. Dit zelfde geldt tevens voor de ASI-methode.

Flexibiliteit in de methode

De flexibiliteit van de methode is in dit project nodig aangezien er een proof of concept wordt ontwikkeld.

Uiteraard wordt voor het bouwen van het proof of concept alles goed onderzocht. Er kunnen echter altijd bij het bouwen of bij het testen fouten omhoog komen.
In de ASI methode zit het uitwerken en eventuele testen in één fase. Binnen de fase kan er altijd worden teruggevallen bij fouten. Als er echter fouten omhoogkomen die een fase eerder zijn gemaakt dan kan er niet terug worden gekeerd.
Bij de watervalmethode volgens het sashimimodel kan door de overlap in de fases wel worden teruggekeerd om eventuele fouten en wijzigingen aan te pakken.

Project toepassing

Uiteraard is het belangrijk dat de methode van toepassing is voor het project. Projectmethoden zoals PRINCE2 en PMBok zijn al snel van toepassing voor een project. Dit komt doordat er beschreven wordt hoe het project gemanaged moet worden en niet wat en hoe het product gemaakt moet worden.
De toepassing van de waterval methode is het ontwikkelen van een product, meestal software. Aangezien in dit project een proof of concept wordt ontwikkeld is deze methode van toepassing. Ook staan de eisen in het project vast en wordt het project uitgevoerd door één persoon.
De ASI-methode is een methode voor het ontwikkelen van een technische infrastructuur. Binnen dit project is wel gekozen om een netwerk/infrastructuur te ontwikkelen. De ontwikkeling van een technische infrastructuur binnen ASI loopt echter veel verder. Het beheer en support van de infrastructuur moet ook volledig worden besproken. Dit is echter geen onderdeel van het project. Tevens moet er binnen deze methode gezocht worden naar leveranciers van de middelen, maar deze zijn allemaal bekend. Tenslotte is het zo dat de ASI methode een watervalmethode is zonder overlap in de fases.

Schaalbaarheid

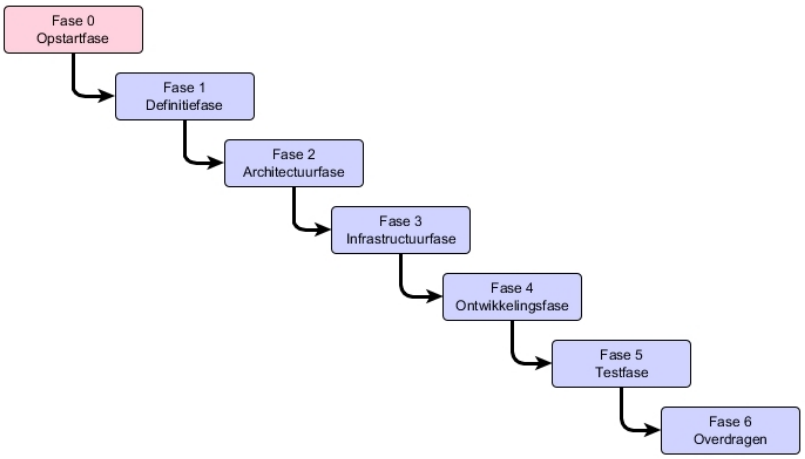
Met de schaalbaarheid van een methode wordt bedoeld hoe gemakkelijk deze methode afgeschaald kan worden. Aangezien het afstudeertraject alleen gedaan wordt en het projectteam klein is moet de methode hier ook op uitgerust zijn.
Zo zijn PRINCE2 en PMBok redelijk schaalbaar aangezien er gekozen kan worden voor een minimalistische aanpak. Echter is zelfs de minimalistische aanpak van deze methodieken een uitnodiging voor veel onnodige documentatie. Dit komt omdat het projectmethoden zijn, en in dit project zijn de enige leden de projectleider, de opdrachtgever en enige stakeholders.
De waterval- en ASI methoden zijn ontwikkelmethoden. Deze methodieken zijn meer gericht op het ontwikkelen van een object. Uiteraard is er in deze methodieken projectbeheersing ingebouwd. Deze beheersing is echter wel zo dat er weinig tot geen onnodige documentatie gedaan hoeft te worden.

Methodiek keuze

Uiteindelijk is door het afwegen van verschillende voor- en nadelen van de projectmethodieken de waterval- of sashimimethode naar voren gekomen als beste keus. Uiteraard wordt in het project de waterval met overlap of sashimi gebruikt.

3.1 Fasering

In afbeelding 1 is te zien hoe in het project de fasering is ingedeeld. Toch wordt er gedeeltelijk afgeweken van de gebruikelijke indeling voor de watervalmethodiek. De reden van afwijken zit in de laatste twee fases van de watervalfasering. Dit zijn de uitrol en onderhoud van het product. In dit project is uitrol en onderhoud van het proof of concept echter niet van belang. Dit komt omdat er een proof of concept wordt aangeleverd: deze hoeft niet bedrijfsbreed te worden uitgerold en er hoeft ook geen verder onderhoud aan te worden gepleegd.



Afbeelding 1: De gebruikte fasering volgens de sashimimethode.

Aan het eind van elke fase zal er een faseovergang plaatsvinden. Een faseovergang betekent dat er een gesprek met de opdrachtgever gemaakt wordt. In dit gesprek wordt besproken welke producten zijn gemaakt, wat de volgende fase zal inhouden en wat de volgende fase zal opleveren. Het eind van de fase is dus een beslismoment (GO of NO GO). Bij een GO zal worden gestart met de volgende fase. Bij een NO GO zullen eerst de knelpunten in die fase verholpen moeten worden.

Dit project wordt uitgevoerd volgens de waterval/sashimimethode. Dit betekent dat er alles aan wordt gedaan om in een fase de producten juist op te leveren. Blijkt er echter in een vorige fase iets te zijn misgegaan is er nog de mogelijkheid om terug te keren om dit probleem aan te pakken en verder te gaan. Met deze oplossing hoeft niet het gehele project of een gehele fase over gedaan te worden.

Opstartfase

In de opstartfase worden de voorbereidende werkzaamheden uitgevoerd om het project opgestart te krijgen. Deze werkzaamheden bestaan uit het opzetten van de werkplek en het uitzoeken van een projectmethode.

Definitiefase

In de definitiefase moet het project gedefinieerd worden. Het gaat dan dus om het vormgeven en het helder krijgen van het project. Voor het plan van aanpak worden de problemen, doelstellingen, voorwaarden en andere onderwerpen uitgezocht om een duidelijk beeld te krijgen van het project.
Ook worden in deze fase de eisen achterhaald bij de opdrachtgever. Dit zal gebeuren in één van de wekelijkse gesprekken.
Om beter te kunnen integreren in het bedrijf zal in deze fase ook een bedrijfsoriëntatie worden geschreven. Met deze oriëntatie kan een beter beeld gevormd worden van het bedrijf.

Architectuurfase

In de architectuurfase moet er een globaal ontwerp worden gemaakt voor het proof of concept. Om sommige kritische gedeeltes van OpenStack te kunnen testen zal in deze fase ook een proefopstelling worden opgezet

die vooral bedoeld is om de performance van OpenStack op de hardware te onderzoeken. Er zullen performance tests nodig zijn om te zien of de eisen worden behaald met de architectuur. Aan de hand van deze tests kan er een architectuurontwerp worden gecreëerd waar alle eisen kunnen worden verantwoord. Na de test kan de meest kritieke bottleneck worden aangepakt in het ontwerp en later in de bouwfase.
Ook dienen deze performance tests om te zien of OpenStack überhaupt wel draait op de servers.
Aan het eind van de fase wordt het globale ontwerp besproken met de opdrachtgever. Op zijn beurt kan de opdrachtgever een GO of een NO GO geven voor de volgende fase.

Ontwerpfase

In de ontwerpfase zal het proof of concept gedetailleerd ontworpen moeten worden (infrastructuurontwerp). Het ontwerp moet in deze fase zo worden uitgewerkt dat er in de ontwikkelfase zo een werkend product opgezet kan worden.

Ontwikkelfase

In de ontwikkelfase wordt het proof of concept opgebouwd volgens de ontwerpen. Om het proof of concept later weer reproduceerbaar te maken zal hier ook een handleiding worden geschreven. Deze handleiding beschrijft precies wat er is gedaan om het proof of concept op te zetten.
Ook wordt in deze fase het testplan geschreven. In dit plan wordt uit de doeken gedaan op welke manier de systeemeisen getest moeten worden.

Testfase

In de testfase wordt het proof of concept op die punten getest die in de initiatiefase als systeemeisen waren opgegeven. Zo kan er achter worden gekomen of het systeem daadwerkelijk voldoet aan de vooraf gestelde eisen.
Ook wordt in de testfase aan de hand van de testresultaten een gedetailleerde winstberekening gemaakt. Met de uitkomst van de testen zijn er variabelen in te vullen in de winstberekening die eerder nog niet voorhanden waren.
In de testfase kunnen ook nog wijzigingen aan het systeem boven komen drijven. Als zich er problemen voordoen bij het testen kunnen deze opgelost worden door terug te gaan naar de ontwikkelfase. Als de problemen opgelost zijn kan er weer worden getest. De handleiding die is beschreven in de ontwikkelfase zal op dit moment ook aangepast moeten worden.

Overdragen

In de laatste fase van het project worden alle documenten en producten veilig overgedragen aan de opdrachtgever en de school.

4. Risico

Tabel 2: De risico"s binnen het project.

Risico	Kans	Ernst risico	Maatregel(en)
Collega / specialist heeft geen tijd.	Klein	Middel	Op tijd afspraken inplannen, een andere afspraak inplannen of een andere collega vragen.
Achterlopen op schema	Middel	Middel	Planning goed in de gaten houden, prioriteiten stellen, de planning aanpassen en de marge aanspreken.
Gebrek aan kennis	Middel	Klein	Genoeg kennis opdoen in de definitiefase, hulp inroepen

			van collega's.
Langdurige ziekte	Klein	Groot	Het project stopzetten en hervatten als daar de mogelijkheid toe is.
Te weinig informatie beschikbaar.	Klein	Middel	Een specialist raadplegen.
School geeft geen akkoord bij controle van de producten.	Klein	Middel	Samen met school en de opdrachtgever een oplossing bedenken.
Verlies van gegevens.	Middel	Middel	Een goede back-up policy instellen. Zo zal er gebruik worden gemaakt van Google drive en een memory stick als back-up
Verlies van proof of concept of wijzigingen aan proof of concept.	Klein	Groot	Elke dag in een log bijhouden wat er aan het proof of concept is gedaan. De activiteiten die verricht moeten worden om op dat punt te komen bijhouden.

5. Kwaliteitsborging

Vanuit i3D

Om kwaliteit binnen het project te waarborgen zijn er standaard meetings ingepland. Deze meetings vinden één of twee keer per twee weken plaats. Het kan ondersteuning bieden zodat de kwaliteit van het product hoog blijft. Bij problemen en vragen kunnen collega's of de opdrachtgever altijd om hulp worden gevraagd.
Tevens vind aan het eind van elke faseovergang een gesprek met de opdrachtgever plaats. In dit gesprek kan de opdrachtgever aangeven of hij de fase goedkeurt (GO) of niet (NO GO).

Vanuit school

Om de kwaliteit van het project te garanderen vinden er ook een aantal contactmomenten plaats tussen school en de afstudeerder. Eerst vind een bedrijfsbezoek plaats. In dit bedrijfsbezoek kan de afstudeerder nog enige vragen stellen. Een aantal weken daarna stuurt de afstudeerder een voortgangsverslag naar school. Hier kunnen opmerkingen en vragen van school uit door komen.
Ook kan er bij vragen contact worden opgenomen met de begeleiders van school.

6. Communicatie

Naar de examinatoren op school kan er communicatie plaatsvinden via de mail en via de telefoon. In eerste instantie wordt er alleen naar school gemaild voor de standaard afspraken. Uiteindelijk kan er ook nog contact plaatsvinden in "real life".
Zo moet er een bedrijfsbezoek plaatsvinden door de eerste examinator. Dit bezoek is echter wel op initiatief van de afstudeerder. Ook de andere standaard contactmomenten vinden via de mail plaats.

De communicatie naar het bedrijf zal vooral plaatsvinden via de begeleider. Er is de mogelijkheid om één of twee keer per twee weken een meeting in te plannen. Hier kunnen vragen worden gesteld en overlegd worden over de opdracht. Tussendoor kunnen er ook contactmomenten plaatsvinden met de begeleider/opdrachtgever. Deze momenten zijn echter niet ingepland en er kan dus niet van uit worden gegaan dat er ook een contactmoment kan of zal plaatsvinden.
Binnen het bedrijf kan er altijd een vraag worden gesteld aan de andere medewerkers. Er loopt veel ervaring en expertise rond. De kans is dus aanwezig dat de vraag beantwoord kan worden.

6. Kosten en baten

Kosten:

Aan het project zijn voor i3D niet veel kosten verbonden. De kosten die wel bestaan zijn:

- Tijd voor begeleiding en meetings
- De bezette werkplek
- De onkostenvergoeding

De tijd die Rick Sloot kwijt is aan meetings met de afstudeerder kunnen redelijk worden ingeschat. In de berekening wordt er van uit gegaan dat er elke week een meeting plaatsvindt. Ook zal in de berekening worden meegenomen dat de afstudeerder eerder is begonnen met de stage.
Een gemiddelde meeting is een uur lang. Er worden (met eerder beginnen) twintig meetings afgesproken. Als daar de korte vragen aan worden toegevoegd komt er nog zo'n twee uur bij.

*45 min * 20 meetings + 2 uur = 17 uur meeting.*

De bezette werkplek is een open plek. Hier zit dus niemand op het moment van de stage. Wel kost deze plek energie.

In de tijdberekening wordt de extra tijd apart opgeteld. De extra tijd is verkregen door eerder dan de uiterlijke begindatum aan het project te beginnen. De tijd dat de afstudeerder in het bedrijf werkzaam is:

*((8 uur per dag * 5 dagen per week) * 17 weken = 680 uur) + ((8 uur per dag * 5 dagen per week) * 4 weken = 160 uur) = 840 uur.*

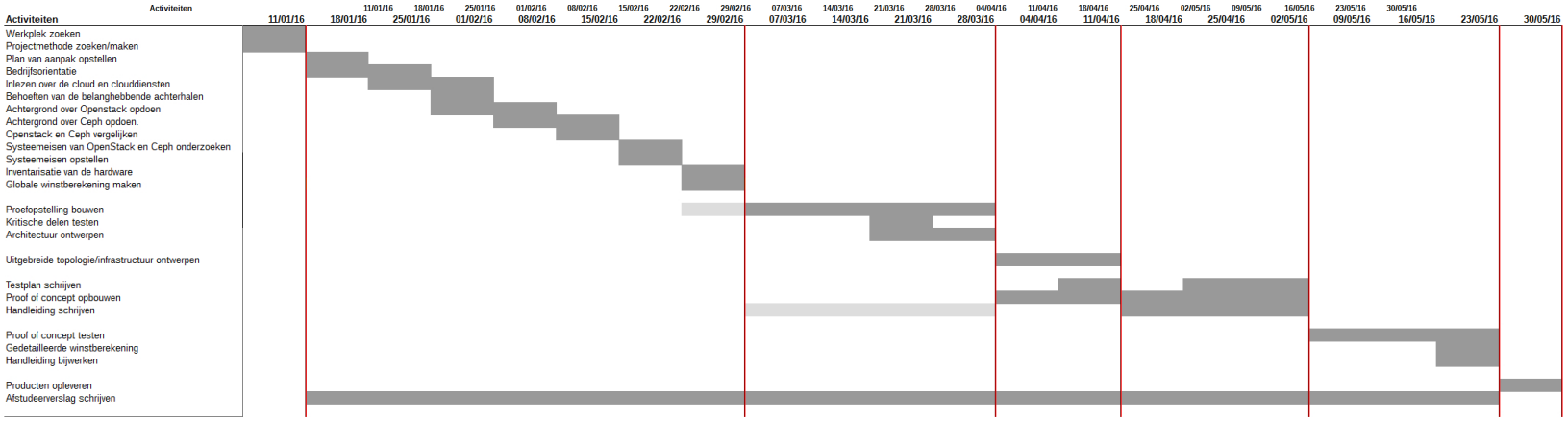
Voor de gemaakte uren krijgt de afstudeerder een maandelijkse onkosten- en reisvergoeding.

Baten:

De baten die i3D terugkrijgt bij een succesvol project:

- Antwoord of en hoe OpenStack opgezet kan worden op oude servers.
- Theoretische winstberekening van de voorgestelde opstelling.
- Een afgerond documentonderzoek over OpenStack en afhankelijke onderdelen, hiermee kan de nodige kennis worden opgedaan door de lezers.
- Aanbevelingen over het verbeteren van het proof of concept.

7. Planning



Afbeelding 2: De planning van het project.

Bijlage X: Architectuurontwerp

Architectuurontwerp

Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low-cost cloud oplossing te creëren bij i3D

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Stagebedrijf: i3D
Plaats: Rotterdam

Datum: 26-03-2016

Versiebeheer

Versie	Datum	Wijziging
0.1	26-03-2016	Layout opzetten
1.0	05-04-2016	Document opleveren

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

Inleiding

Dit document is gemaakt in de loop van de architectuurfase van het project. Het beschrijft het architectuurontwerp en de gemaakte keuzes die leidde tot dit ontwerp.

In eerste instantie was het de bedoeling om in de architectuurfase een proefopstelling op te bouwen. Aan de hand van deze opstelling zouden dan de performance eisen getest kunnen worden. Samen met de performance eisen kon dan ook de “limiet” van verschillende nodes in de omgeving worden getest. De node die de meest kritieke bottleneck qua performance bleek te zijn zou dan redundant worden uitgevoerd. In de testfase van het project zou de redundante node in zijn gebouwd en kon deze meegenomen worden in de test. Aan de verschillen in de performance kon dan vrij betrouwbaar worden berekend om de hoeveel klanten er een nieuwe node zou moeten komen. Dit is belangrijk voor de winstberekening.
Echter door tijdsgebrek is er met de opdrachtgever afgesproken dat de meest kritieke bottleneck niet hoeft te worden ingebouwd. Wel is de opbouw van de proefopstelling bewaard gebleven in deze fase. Alle performance eisen zijn ook getest op deze opstelling.

Inhoudsopgave

Verklarende woordenlijst / symbolenlijst

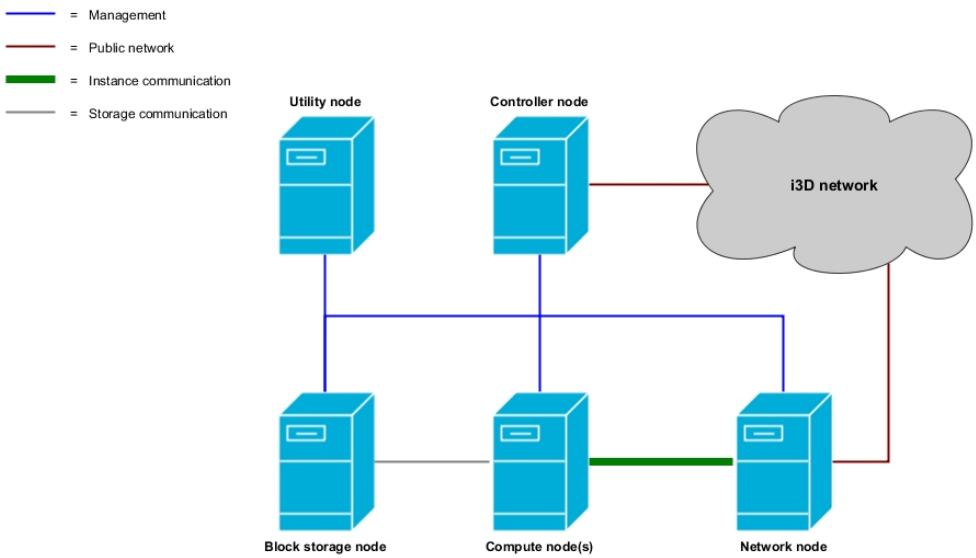
LVM = staat voor Logical Volume Management

Flavor = Een flavor binnen OpenStack is een blauwdruk van de virtuele resources die gekozen kunnen worden voor een instance.

Floating IP = Een IP-adres die “zweeft”. Binnen OpenStack kunnen deze floating IP-adressen op commando worden gekoppeld en ontkoppeld van instances en virtuele “hardware” zoals virtuele routers.

1. Proefopstelling

In de architectuurfase is het van belang dat het ontwerp van de architectuur duidelijk wordt. Aan de hand van de opgegeven en achterhaalde systeemeisen kan een ontwerp worden gemaakt. Het is echter wel de bedoeling dat het ontwerp ook aan de eisen voldoet.



Afbeelding 1: De proefopstelling waarbij de controller-, compute-, network- en block storage nodes(s) worden opgezet.

In de architectuurfase moeten de gestelde systeemeisen kunnen worden verantwoord. Dit is de reden dat er is gekozen om de performance te testen op een proefopstellen alvorens het architectuurontwerp te maken, de performance is namelijk een kritiek punt binnen de architectuur. Zeker omdat er wordt gewerkt met oudere hardware is het testen van de performance voordat het architectuurontwerp wordt gemaakt van belang. Zonder deze proefopstelling is er geen enkel beeld over hoe een minimale OpenStack installatie zich houdt op de hardware.
Het was in het begin van de architectuurfase ook de bedoeling om naast de performance ook duidelijkheid te scheppen over de meest kritieke bottleneck binnen de opstelling: deze zou redundant uitgevoerd moeten worden in het proof of concept. Om uit te vinden waar zich de meest kritieke bottleneck bevond had dit ook moeten worden getest in de architectuurfase, zo zou de redundante oplossing voor de bottleneck worden meegenomen in de ontwerpen. De redundantie op de plek van de meestkritieke bottleneck is door tijdsgebrek echter geschrapt.

In deze proefopstelling en het proof of concept zal er voor gekozen worden om alle OpenStack nodes op fysieke servers zonder virtualisatielaag te installeren, dit omwille van de lagere performance van de systemen en het aantal systemen wat opgeleverd wordt.

Er is gekozen voor een opstelling zoals die te zien is op afbeelding 1. De keuze voor deze opstelling is gemaakt omdat dit volgens OpenStack de minimale proof of concept opstelling is.
Het probleem met een proefopstelling opbouwen voor OpenStack is dat veel componenten in elkaar steken en van elkaar afhankelijk zijn. Om een werkende geheel te krijgen, zelfs voor een proefopstelling, zal de OpenStack opstelling zoals op afbeelding 20 moeten worden opgebouwd. Dit is de minimale opstelling van OpenStack als er niet wordt gekeken naar one- of two-server setups.
De block storage node hoeft niet op te worden genomen in een minimale opstelling, dit is in de proefopstelling echter wel gedaan omdat de performance-eisen getest worden, hier is de block storage node bij nodig. Voor de

utility node geldt ook dat hij niet vereist is in een minimale OpenStack omgeving, deze node wordt echter gebruikt om OpenStack Rally (de testsuite) te draaien.

- Controller node
- Compute node
- Network node
- Block storage node
- Utility node

1.1 Ontwerpkeuzes

Binnen dit subhoofdstuk worden de verschillende ontwerpkeuzes toegelicht die zijn gemaakt bij het opzetten van de proefopstelling. Sommige, zo niet de meeste, onderdelen van de proefopstelling kunnen worden meegenomen om het uiteindelijke proof of concept te creëren, daarom worden de ontwerpkeuzes toegelicht.

1.1.1 Besturingssysteem

Voor dit project is gekozen voor de distributie Ubuntu 14.04 LTS om OpenStack op te zetten. De voornaamste redenen hiervoor zijn eigenlijk de grote groep (OpenStack)gebruikers en mijn eigen ervaring met het OS. Volgens Mark Shuttleworth heeft 55% van de Openstack gebruikers gekozen voor Ubuntu [1]. Als er veel gebruikers zijn dan zijn er ook veel mensen die eventueel voor support kunnen zorgen. En zullen er potentieel minder fouten voorkomen.
Zelf heb ik veel meer ervaring met Ubuntu dan met de andere operating systems. Bij het gebruik van Ubuntu is er ook veel minder behoefte aan inwerken op het OS.

1.1.2 Database

Binnen OpenStack is een database noodzakelijk. Deze database wordt gebruikt voor het opslaan van cruciale data door de OpenStack services. Er zijn verschillende databases te gebruiken onder OpenStack zoals: MySQL, PostgreSQL en MariaDB.

Uiteindelijk is de keuze gevallen op de MySQL database. Dit komt omdat dit binnen i3D de standaard is voor databases. Voor de werking en performance zijn de verschillen tussen deze databases niet van groot belang.

1.1.3 Passwords

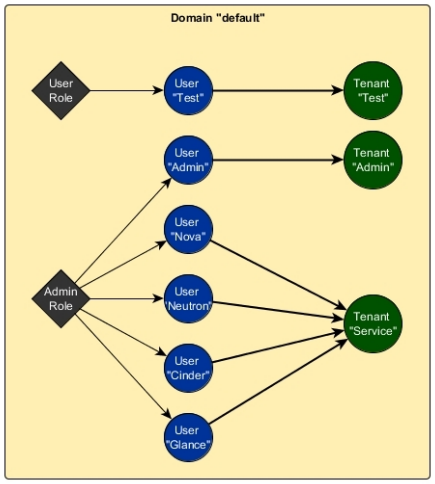
De passwords die worden gebruikt in het proof of concept zijn opgeslagen in Keepass. Het genereren van de wachtwoorden is gebeurd door het uitvoeren van het volgende commando:

< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-20};echo;

1.1.4 Keystone

De controller node wordt eerst uitgerust met de keystone service. De Keystone service is bedoeld voor de authenticatie, autorisatie en service catalogus binnen OpenStack. De entiteiten die aangemaakt zijn binnen Keystone zijn te zien op afbeelding 2.
Deze service regelt ook het identity management van de OpenStack services. De Keystone service is echter ook vrij belangrijk bij de performance van OpenStack: er kan een vertraging optreden als Keystone niet snel genoeg de aanvragen kan behandelen. Om dit “probleem” voor te zijn wordt er gebruik gemaakt van “memcached”.

Memcached is een stuk software dat een cache in het geheugen van een node maakt. Deze cache zorgt voor het opvangen van de load naar een database. Memcached is bedoeld om dynamisch web verkeer te versnellen, dit is dus ideaal voor Keystone. De grootte van de cache is vooraf ingesteld. De data die door memcached gaat wordt afgehandeld met rotatie. Als er een nieuw stuk data in de cache wordt gezet als deze a vol is dan zal de oudste data worden overschreven. De data wordt opgeslagen in zogenaamde “slabs”. Deze slabs kunnen variëren in grootte.



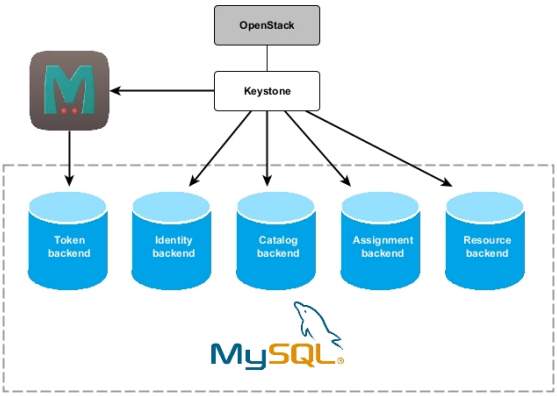
Afbeelding 2: De entiteiten die aangemaakt zijn in de proefopstelling.

Voor dit proof of concept is als Keystone identity backend de SQL-database gekozen. Vooral door het feit dat het eindproduct van dit project een proof of concept is. Tevens bestaat er nog geen LDAP server (en bijkomende active directory).
Voor het token backend is gekozen voor het inzetten van memcached. Hiermee kan de algehele performance van OpenStack op niveau worden gehouden (minder kans op performance drops). Dit is zeker slim in een infrastructuur op basis van oudere hardware. Daar kan niet standaard uit worden gegaan van hoge database performance.

Om de configuratie van de Keystone service op te zetten is grotendeels de guide van OpenStack zelf gebruikt. Ook is er in deze guide een hoofdstuk gewijd aan het testen van de service. Hier wordt getest of de admin user en de test user tokens kunnen aanmaken.

1.1.5 Glance

Als tweede wordt de image service genaamd “Glance” geïnstalleerd. Glance zorgt voor registratie van de images binnen OpenStack. De service weet waar de images staan en zorgt ook voor het transport van en naar compute nodes.



Afbeelding 3: De backends die zijn gebruikt binnen Keystone. Het token backend gebruikt Memcached als caching.

De Glance service heeft een storage backend nodig om de images in op te slaan. Als backend kan een Swift cluster worden gekozen of een backend van Swift (zoals Ceph). In de proefopstelling wordt Swift echter nog niet opgezet omdat geen vereist gedeelte is in OpenStack.
Bij het ontbreken van de object storage is er gebruik gemaakt van interne Glance storage. De interne storage voor Glance is de directory */var/lib/glance/images* op de controller node.

1.1.6 Nova

Als derde service wordt de Nova compute service geïnstalleerd. De Nova service is de belangrijkste service in OpenStack en ook de eerste service die aanwezig was in OpenStack. Deze service houdt de draaiende images bij en zorgt voor aansturing voor de images.

Als hypervisor binnen OpenStack kunnen verschillende varianten worden gekozen. Er is binnen deze opstelling

gekozen voor KVM aangezien deze hypervisor het beste wordt getest bij OpenStack. Naast KVM had er ook gekozen kunnen worden voor QEMU, beide worden aangestuurd door de “libvirt” virtualisatie API.
De QEMU hypervisor is beter voor oudere hardware die geen hardwarematige virtualisatie ondersteunt. Om deze reden is het niet verstandig om QEMU als hypervisor te kiezen in een productieomgeving. KVM daarentegen ondersteund wel hardwarematig virtualisatie en heeft hiermee ook een performance voordeel t.o.v. QEMU. Om deze reden is er gekozen voor KVM.

1.1.7 Neutron

De vierde service die is geïnstalleerd betreft Neutron, de networking service. Met Neutron wordt netwerkconnectiviteit aangeboden aan de VM's. Het idee bij Neutron is dat er virtuele netwerken kunnen worden opgezet waar VM's aangehangen kunnen worden. Dit koppelen gaat via een virtuele switch (Open vSwitch). Tevens kunnen er SDN-services worden aangeboden met Neutron.

Klanten moeten binnen het proof of concept ook de mogelijkheid hebben om IP-adressen op te vragen waarmee ze naar het i3D netwerk (internet) kunnen communiceren. Dit gaat door middel van floating IP's in OpenStack. Floating IP's zijn echter zo ingesteld dat het een lokaal IP-adres betreft binnen OpenStack: Een NAT-oplossing dus. Aangezien er geen SNAT en DNAT gebruikt mogen worden binnen dit proof of concept en er voldoende vrije globale IP-adressen voor handen zijn binnen i3D wordt er on-to-one NAT gebruikt. In deze NAT oplossing wordt er één globaal IP-adres geconverteerd naar één intern IP-adres.

1.1.8 Cinder

Om de diskspace op te zetten op de block storage node voor instances moet in het geval van dit proof of concept LVM worden gebruikt. Omdat er in de Block storage nodes RAID-kaarten worden gebruikt is er een uitdaging. Deze uitdaging is het volgende:

Met een hardware RAID in RAID 10 waarbij de vier drives tot één grote drive worden gemaakt is er geen plek meer voor lege drives. De maximale drive capaciteit in een DL120 is namelijk vier. Om Cinder op te zetten is er een leeg stuk ruimte op een drive nodig. Aangezien er RAID wordt gebruikt moet de LVM partitie “naast” het OS worden geïnstalleerd.
Dit kan op een aantal manieren:

- Met een loopback device.
- Op de zelfde schijf in een andere partitie.

Met een loopback device wordt er binnen het OS een virtueel device gemaakt. Het besturingssysteem ziet dit virtuele device als een echte fysieke schijf. Met dit device kan Cinder worden opgezet. Echter zal de performance hier onder lijden. Dit komt door het “emuleren” van de schijf. Het pad van de data volgt dan Instance → Block storage node → OS op block storage node → Loopback device.
De betere oplossing is dus de installatie van de LVM-partitie naast het OS. Hier hoeft de data niet eerst te worden omgezet naar een virtuele schijf in het OS. Het pad van de data die hier wordt gevolgd is: instance → Block storage node → LVM partitie.

1.1.9 Rally

De Rally installatie moet worden geïnstalleerd op een aparte node genaamd de “utility node”. Dit is in verband met het spreiden van de load. Er kan ook gekozen worden op Rally te installeren op de controller node. Rally kan echter makkelijk op een aparte node en is alleen bedoeld om te testen, het is geen functioneel onderdeel.

Op deze manier kan de controller node een stukje worden ontzien.

1.1.10 Tests

De tests die volgen uit de opbouw van de proefopstelling worden beschreven in bijlage XII. Bij het testen van de proefopstelling worden de kritieke punten getest. Deze punten moeten worden getest voordat sommige eisen in het architectuurontwerp verantwoord kunnen worden. De kritieke punten die worden getest zijn de punten van de performance.

2. Eisen

In dit hoofdstuk worden de achterhaalde systeemeisen genoemd. Aan deze eisen worden oplossingen gekoppeld, eventueel met een alternatief voor de oplossing. Tevens worden de systeemgrenzen genoemd. De systeemgrenzen zijn in het leven geroepen om te noemen welke niet-functionele eisen niet in het systeem zijn opgenomen en waarom niet. Ook wordt er aangegeven welk gedeelte van deze niet-functionele eisen er wel wordt ingevoerd.

2.1 Hoe wordt er aan de eisen voldaan?

2.1.1 Storage mag geen dataverlies lijden bij uitval van één disk

Eis:

In het uiteindelijke proof of concept mag de permanente storage bij schijfuitval (van 1 schijf in het proof of concept) geen dataverlies lijden.

Oplossing:

Permanente storage wordt in deze architectuur aangeboden door Cinder. Dit is block storage op een aparte storage node. Voor het verdere verloop van het project wordt Ceph niet meer gebruikt en er wordt geen onderzoek gedaan naar andere storage back-ends (projectgrens) voor Cinder. Door deze reden wordt er gebruik gemaakt van interne/lokale LVM storage in de block storage node. Er wordt ook maar van één block storage node gebruik gemaakt. Met één node kan al worden aangetoond dat block storage werkt binnen OpenStack. Ook kan hiermee aan de eis worden voldaan.

Om deze lokale storage in de block storage node toch redundant te maken is er gekozen voor een RAID oplossing in de node. Deze RAID-oplossing wordt geïmplementeerd door middel van een hardware RAID kaart. Er is gekozen voor een hardware RAID kaart omdat deze voor de DL120 G6 in overvloed aanwezig zijn binnen i3D. Tevens worden de schijven in de node dan hot-swappable. Ook is van deze lower-performance servers alle performance nodig.
Een andere oplossing had software RAID kunnen zijn. Dit alternatief is echter processor intensief: er kan minder processorkracht worden gegeven aan de core activity van de node. Ook wordt de hot-swap functionaliteit niet of nauwelijks ondersteunt.

De fysieke server heeft vier disk bays.
In deze storage node wordt een RAID-configuratie (RAID10) gedraaid. Er kan worden gekozen tussen een aantal RAID configuraties, nl: RAID1, RAID0, RAID10, RAID5. Uiteraard zijn er meer RAID-opstellingen, er moet echter van uit worden gegaan de er maar vier schijven in de servers passen. Bij zulke kleine schijf aantallen zijn de andere RAID-oplossingen weinig functioneel.
Eigenlijk kunnen RAID0 en RAID1 direct worden weggestreept omdat “the best of both worlds” in RAID10 zit. Tevens is het zo dat RAID0 geen redundantie ondersteund en wel goede performance biedt. RAID1 biedt zeer gode redundantie maar heeft geen performance voordelen.
Er is voor RAID 10 gekozen omdat dit zorgt voor meer performance tijdens runtime en ook tijdens recovery dan RAID 5. In RAID 10 is er ook een minder kans op data loss omdat tijdens disk fail sneller een nieuwe schijf kan worden gerecovered. Het nadeel is echter wel dat er meer disks nodig zijn.

Alternatief:

Voor deze eis is er geen alternatief te verzinnen die zou passen in dit project. Een goed alternatief zou zijn om de block storage te installeren binnen Ceph. Dit komt omdat Ceph automatisch redundant is door zijn opzet (object storage).
Tevens kan er bedacht worden dat de Cinder block storage nodes dan maar redundant worden uitgevoerd om aan deze eis te voldoen. Het is echter zo dat Cinder block storage nodes niet redundant uit te voeren zijn. Als er redundantie moet ontstaan binnen Cinder moet dit van het backend komen.

2.1.2 Documentatie

Eis:

Na het project moet de opstelling/het proof of concept opnieuw opgezet kunnen worden.

Oplossing:

Aan deze eis kan vrij gemakkelijk worden voldaan. Dit kan door het schrijven van een handleiding tijdens de opbouw en het testen van het proof of concept.
Als het proof of concept naderhand weer opnieuw moet worden opgebouwd kan dit met de handleiding.

Alternatief:

Bij deze eis is geen alternatief beschikbaar.

2.1.3 Winstberekening

Eisen:

- De uiteindelijke winstberekening, in de testfase, moet positief zijn (er moet winst gemaakt kunnen worden).
- De uiteindelijke abonnementskosten voor klanten moeten lager zijn dan de primaire cloud oplossing.

Oplossing:

Aan de uitkomst van deze eisen kan niet veel worden gedaan. Bij het opzetten van het proof of concept kan er aandacht worden besteed aan het de (theoretische) kosten die worden gemaakt. Ook moet uit de globale winstberekening (in de definitiefase) al een positief resultaat zijn gekomen.
De eis kan wel worden getest in de testfase door het afmaken van de winstberekening. De tweede eis komt voort uit de eerste eis.

Alternatief:

Bij deze eis is geen alternatief beschikbaar.

2.1.4 Schaalbaarheid

Eis:

De "klanten" moeten in het proof of concept de mogelijkheid hebben om hun resources te kunnen schalen.

Oplossing:

Als het boot volume zich fysiek bevindt op de compute node dan kan dit uit worden gevoerd door de volgende acties:

- Een snapshot maken van het volume.
- Hun Cinder volume van hun instance af te koppelen.
- Een instance maken met de juiste instellingen (flavor) op basis van de snapshot.
- Hun Cinder volume koppelen aan deze nieuwe instance.

Deze oplossing draait dus op het idee dat er binnen OpenStack nieuwe instances kunnen worden gecreëerd van snapshots. Om dit te laten werken moeten er snapshots kunnen worden gemaakt en moet de Cinder service actief zijn.

Alternatief:

Er kan gebruik worden gemaakt van het “Nova-resize” commando. Met dit commando kan er een nieuwe flavor worden gekozen voor de instance. Hierna zal OpenStack de instance naar zijn nieuwe locatie transporteren op basis van de aanwezige filters. De oude instance blijft nu nog wel actief, deze moet hierna nog worden verwijderd. Voor dit alternatief is het noodzakelijk dat er meerdere compute nodes aanwezig zijn.

2.1.5 Storage bestellen

Eis:

De "klanten" moeten in het proof of concept de mogelijkheid hebben om extra storage te kunnen "bestellen" (schalen/resource provisioning).

Oplossing:

Aan deze eis kan worden voldaan door het opzetten van een oplossing met block storage. Binnen OpenStack is het Cinder project voor block storage. Klanten kunnen extra persistente storage toewijzen aan hun instance met block storage[2].
Voor deze eis is het opzetten van minimaal één Cinder block storage node noodzakelijk.

Alternatief:

Als alternatief op block storage kan ook de fysieke storage op de compute node zelf worden gebruikt. Het gevaar hier is echter dat deze storage niet redundant is uitgevoerd. Als er meer lokale storage nodig is kan er een “Nova-resize” worden overwogen. Er kan worden gekozen voor een beter passende flavor.
Het nadeel is hier echter wel bij dat er downtime bij het nova-resize commando komt kijken.
Voor dit alternatief is het noodzakelijk dat er meerdere compute nodes aanwezig zijn.

2.1.6 Performance

Eis:

Het netwerkverkeer van en naar de storage mag niet zo hoog worden dat de verbinding wordt dichtgetrokken.

Tijdens normaal (uitgezonderd piekverkeer) gebruik mag het niet hoger worden dan 50% van de lijn.

Oplossing:

Aan de hand van de tests in de architectuurfase is te zeggen dat het netwerkverkeer deze limiet niet haalt. Om het netwerkverkeer, vanwege de hoge pieken, nog iets naar beneden bij te sturen moet er QoS worden ingevoerd op de Cinder node. Deze QoS-rule(s) worden uitgewerkt in de infrastructuurfase.

Alternatief:

N.V.T.

Eis:

Het netwerkverkeer over heel het netwerk mag onder normale omstandigheden (uitgezonderd piekverkeer) niet over 50% van de lijn komen.

Oplossing:

Uit de tests in de architectuurfase blijkt dat dit niet het geval is.

Alternatief:

N.V.T.

Eis:

In het proof of concept mag de slowdown op de VM's niet hoger worden dan 200% bij normaal gebruik (uitgezonderd piekverkeer).

Oplossing:

Uit de tests in de architectuurfase blijkt dat dit niet het geval is. Wel moet er in de infrastructuurfase rules worden ingevoerd voor de overcommit binnen OpenStack Nova. Standaard staat de overcommit op 16 voor de vCPU en 1,5 voor RAM. Dot moet naar 2 voor vCPU en 1 voor het RAM.

Alternatief:

N.V.T.

2.1.7 Security

Eis:

In het proof of concept mogen onbevoegde verbindingen vanaf het i3D netwerk niet in een van de interne netwerken komen.

Oplossing:

Om deze verbindingen te voorkomen zal er een DMZ worden opgezet. Binnen dit DMZ zijn de nodes/servers actief die interne verbindingen en externe verbindingen moeten hebben. De controller(s), NTP-server, SSH-jumpbox en Linux-repository staan hier in.

Er wordt in dit ontwerp gekozen voor een interne losse NTP-server om geen NTP-verbinding naar het internet open te hebben staan. Binnen het i3D-netwerk is er een NTP-server aanwezig waarmee de OpenStack-NTP-server kan synchroniseren. De keuze om de NTP-server niet te implementeren op de controller is gedaan om performance issues zo veel mogelijk te voorkomen. Op het moment dat de NTP-service wel op de controller actief is en er vind een NTP-reflection attack plaats zal dit dus zijn weerslag hebben op de gehele cloud. Dit komt omdat de controller voor een groot deel verantwoordelijk is voor de performance van de cloud.

Ook is er in het ontwerp gekozen voor een SSH-jumpbox. Deze jumpbox is bedoeld om met SSH op het management netwerk van de OpenStack oplossing te kunnen komen. De jumpbox heeft wel een globaal IP-adres. Dit is echter een zwakte punt in de beveiliging van de oplossing, maar wel noodzakelijk. Om deze jumpbox goed te beveiligen worden er Iptables rules ingesteld op deze systemen. In deze rules komt bijvoorbeeld de mogelijkheid voor om alleen verkeer op port 22 naar de jumpbox door te laten die van enkele IP-adressen komt (whitelist).

De Linux repository server wordt aangemaakt zodat alle fysieke nodes kunnen updaten en upgraden over het management netwerk. Deze repo-server heeft echter wel een directe verbinding met het internet nodig zodat hij zelf up-to-date kan blijven.

De network nodes staan ook deels in het DMZ. Dit komt omdat de network nodes naast virtuele netwerken ook de verbinding verzorgen voor de instances. Via externe IP-adressen kunnen instances verbinding maken met i3D netwerken. Zo kunnen er bijvoorbeeld klanten op een webserver komen die gehost is binnen de OpenStack cloud.

Om deze oplossing op te zetten moet er ten eerste sprake zijn van compartimenten in het netwerk. Deze kunnen worden ingebouwd door verschillende netwerken te maken. Zo moet er een netwerk komen voor management en API-verkeer, communicatie met de storage, een tunnel netwerk tussen de compute nodes en network node(s) en verbindingen met het i3D netwerk.
Vervolgens moeten de benodigde servers aanwezig zijn: NTP, SSH-jumpbox, repository server. Ook moeten er Iptables-rules worden aangemaakt voor de benodigde ontzegging van toegang. Alle servers in OpenStack moeten goede toegangsbeveiliging hebben, maar die in het DMZ zeker. Het gaat dan over sterke wachtwoorden en goed ingestelde firewalls.

Alternatief:

Er is voor deze eis geen alternatief beschikbaar.

2.1.8 Platform compatibility

Eis:

Er moet op het proof of concept één soort operating system (één distributie) kunnen worden gehost.

Oplossing:

Er moet een OpenStack opstelling worden gebouwd. Op de compute node kan dan een Linux distributie worden gehost. Er zal worden gekozen voor CirrOS als test OS. In een later stadium zullen er Ubuntu cloud images op gehost worden.

Alternatief:

Er is geen alternatief beschikbaar voor deze eis.

2.1.9 Opensource

Eis:

Binnen het project moet er gebruik zijn gemaakt van OpenStack.

Oplossing:

Het proof of concept wordt opgebouwd op basis van OpenStack.

Alternatief:

Voor deze eis is geen alternatief beschikbaar.

2.1.10 Resource constraints

Eis: De apparatuur die zich in het rack bevindt moet in de testfase onder de 32 ampére stroomsterkte en 1gbit netwerktraffic blijven.

Oplossing:

De opstelling opbouwen. De resource constraints worden niet overschreden aangezien er maar enkele servers worden gebruikt voor het proof of concept. Het rack is gebouwd op volledige load.

Alternatief:

Er is geen alternatief beschikbaar voor deze eis.

2.1.11 Network

Eis:

Het proof of concept moet een alternatief voor SNAT en DNAT hebben om "klanten" publieke IP-adressen op te laten vragen.

Oplossing:

Voor deze eis kan een oplossing genaamd “floating IP” worden gebruikt. Binnen OpenStack is het normaal dat er zogenaamde floating IP's worden gebruikt i.c.m. Neutron networking service. Om geen last te hebben van SNAT en DNAT wordt er gebruik gemaakt van one-to-one NAT. Elk extern IP-adres wordt een op een omgezet naar een intern IP-adres. Op deze manier vormen de vele porten die gebruikt moeten worden geen probleem.

Om deze oplossing te kunnen gebruiken moet Neutron networking service worden gebruikt binnen het proof of concept. Binnen deze networking service moet er gebruik gemaakt worden van one-to-one NAT met floating-IP's.

Alternatief:

Om het gebruik van een NAT-oplossing te omzeilen (dus ook one-to-one NAT) kan er gebruik worden gemaakt van legacy networking. Bij legacy networking krijgt de Nova service controle over het netwerk. Elke compute node heeft dan dus zijn eigen netwerk controle. Op deze manier kan er voor elke instance een direct global-IP adres worden gebruikt.

Eis:

Mensen moeten externe IP-adressen kunnen bestellen om hun instances te kunnen verbinden met internet en i3D netwerken.

Oplossing:

Binnen OpenStack kunnen floating IP-adressen worden uitgedeeld uit een pool die Neutron bijhoud. Deze pool bestaat uit globale IP-adressen.
Om deze oplossing te implementeren moet er gebruik worden gemaakt van Neutron met floating IP-adressen.

Alternatief:

Voor deze eis is er geen alternatief beschikbaar.

2.1.12 Back-up

Eis:

Er moet een back-up van de data van de gebruikers gemaakt kunnen worden.

Oplossing:

Vanaf de Cinder block storage kunnen er back-ups worden gemaakt naar de Swift object storage service. Deze back-ups worden gemaakt met de Cinder back-up driver.
Om deze oplossing te implementeren moet er een Swift object storage service worden opgezet.

Alternatief:

De klanten kunnen zelf een back-up maken van hun data. Dit kunnen ze doen door gebruik te maken van een 3th-party back-up oplossing. In de uiteindelijke OpenStack cloud oplossing is het namelijk ook de bedoeling dat klanten merendeel voor hun eigen data zorgen. De back-up wordt dan alleen gemaakt om klanten van dienst te zijn bij disaster recovery.

Eis:

Er moet een back-up gemaakt kunnen worden van de services binnen OpenStack.

Oplossing:

Er kan een back-up worden gemaakt van de services binnen OpenStack zonder hier iets extra's voor te implementeren. Alle services kunnen worden geback-upt door de nodige bestanden en databases op te slaan.

Alternatief:

Er is voor deze eis geen alternatief beschikbaar.

2.1.13 Overig

Eis:

Binnen het proof of concept moeten snapshots van de VM's gemaakt kunnen worden.

Oplossing:

Snapshots kunnen altijd worden gemaakt binnen OpenStack. De compute service maakt de snapshots en ze worden opgeslagen door de Glance image service.

Alternatief:

Er is geen alternatief beschikbaar voor deze eis.

Eis:

Klanten moeten kunnen inloggen op het OpenStack control panel.

Oplossing:

Binnen OpenStack is er een grafisch control panel beschikbaar genaamd Horizon. Dit is een dashboard waar klanten kunnen inloggen met hun credentials en acties kunnen uitvoeren.
Deze oplossing kan worden ingebouwd door de OpenStack oplossing uit te rusten met de Horizon service.

Alternatief:

Voor deze eis is geen alternatief beschikbaar.

Eis:

De Linux installaties en de OpenStack componenten op de fysieke nodes moeten geüpdatet kunnen worden

Oplossing:

Om deze eis te kunnen waarmaken moet er een lokale Linux repository worden ingebouwd in het netwerk.

Alternatief:

Er kan in plaats van een linux repository ook een cache of proxy server worden opgezet in het interne netwerk.

Met een cache server worden alle aangevraagde packages gecached. Alle packages die toch niet worden gebruikt hoeven dan ook niet aanwezig te zijn op de schijf van de lokale repository.

Eis:

Het verbruik van de klanten moet gemeten kunnen worden.

Oplossing:

Het verbruik van klanten kan worden gemeten met het OpenStack project genaamd Ceilometer. Aan de hand van Ceilometer zouden ook direct rekeningen kunnen worden gemaakt voor de klanten.

Alternatief:

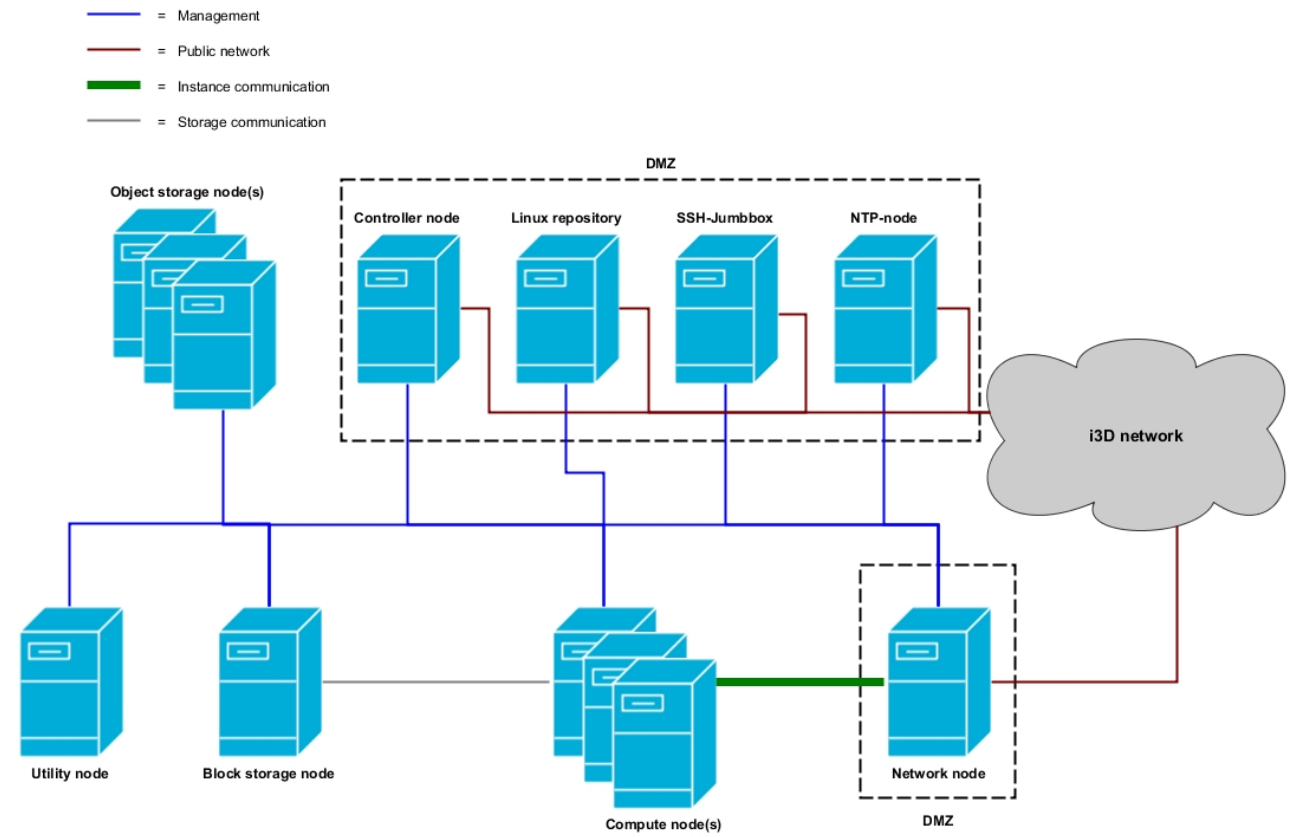
Er is voor deze eis geen alternatief beschikbaar.

2.1.14 Constanten

Bij het ontwerpen en opzetten van het proof concept zijn de volgende punten in gedachte gehouden:

- De servers die gebruikt worden zijn allemaal hetzelfde. Er mogen geen andere servers worden gebruikt, enkel servers van het type DL120 G6.
- De keuzes die worden gemaakt binnen de architectuur moeten zoveel mogelijk berusten om het behouden van lage kosten. Echter mag bij goede onderbouwing wel voor een oplossing worden gekozen die hogere kosten inhouden, hier moet dan echter wel de kwaliteit van het proof of concept mee verbeteren.

3. De architectuur



Afbeelding 5: De architectuur van het proof of concept zonder object storage.

3.1 Netwerken

In het netwerk ontwerp die is weergegeven in afbeelding 5.

3.2 Beschrijving architectuur

In de bovenstaande afbeelding is de architectuur te zien die voor het proof of concept gepland staat. Allereerst vallen de vele kleuren verbindingen op. De verschillende kleuren geven de verschillende netwerken aan waar de connecties zich in bevinden. De legenda links bovenin de afbeelding geeft aan in welk netwerk de connectie zich bevindt. Binnen OpenStack wordt aangeraden om minimaal drie netwerken door te voeren, nl: Het management netwerk, Het tunnel netwerk en het publieke netwerk. Hiernaast is er ook een verbinding tussen de compute nodes en de block storage node te vinden, dit is het netwerk voor de storage communicatie. Fysiek zijn alle connecties aangesloten op één switch. Op deze switch zijn de connecties gescheiden door VLAN's.

Om de OpenStack opstelling zo gesloten mogelijk te houden is er gebruik gemaakt van een DMZ. Alle nodes die met het i3D netwerk zijn verbonden staan in het DMZ. Op deze manier kunnen door middel van Iptables-rules alleen de juiste verbindingen worden binnen gelaten.

Bronvermelding

[1] Steven J. Vaughan-Nichols (16-03-2014). “OpenStack's top operating system: Ubuntu Linux“. ZDnet. Available: http://www.zdnet.com/article/openstacks-top-operating-system-ubuntu-linux/. [Accessed: 09-02-2016]

[2] OpenStack (03-03-2016). “block storage”. Available: http://docs.openstack.org/liberty/install-guide-ubuntu/common/get_started_block_storage.html. OpenStack. [03-03-2016]

Bijlage XI: Infrastructuurontwerp

Infrastructuurontwerp

Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low-cost cloud oplossing te creëren bij i3D

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Stagebedrijf: i3D
Plaats: Rotterdam
Datum: 06-04-2016

Versiebeheer

Versie	Datum	Wijziging
0.1	06-04-2016	Eerste layout
0.2	16-04-2016	Conceptversie
1.0	18-04-2016	Opleveren document

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

Inleiding

Na het afronden van de architectuurfase is er in de infrastructuur fase ruimte om de globale ontwerpen gedetailleerd uit te werken. In de architectuurfase zijn er allerlei (globale) ontwerpen gecreëerd van de uiteindelijke omgeving.
In de infrastructuurfase zijn deze globale ontwerpen zo gedetailleerd uitgewerkt dat er aan de hand van de ontwerpen de betreffende omgeving opgezet kan worden.

Een kosten en baten analyse is te vinden in bijlage V. De configuratie van de onderdelen in het PoC wordt in dit document besproken. Echter is de gedetailleerde configuratie van de OpenStack componenten en overige nodes tezamen met de testopzet te vinden in bijlage VI.

Inhoudsopgave

Verklarende woordenlijst

NIC = Network interface controller, dit is een interface om een systeem te laten verbinden met een netwerk.

Tenant = Een “project” waarbinnen resources bestaan zoals: gebruikers, instances en (virtuele) netwerken.

Region = Een afscheiding tussen de compute services voor verschillende klanten. Met regions kunnen verschillende “discrete sections” worden gecreëerd.

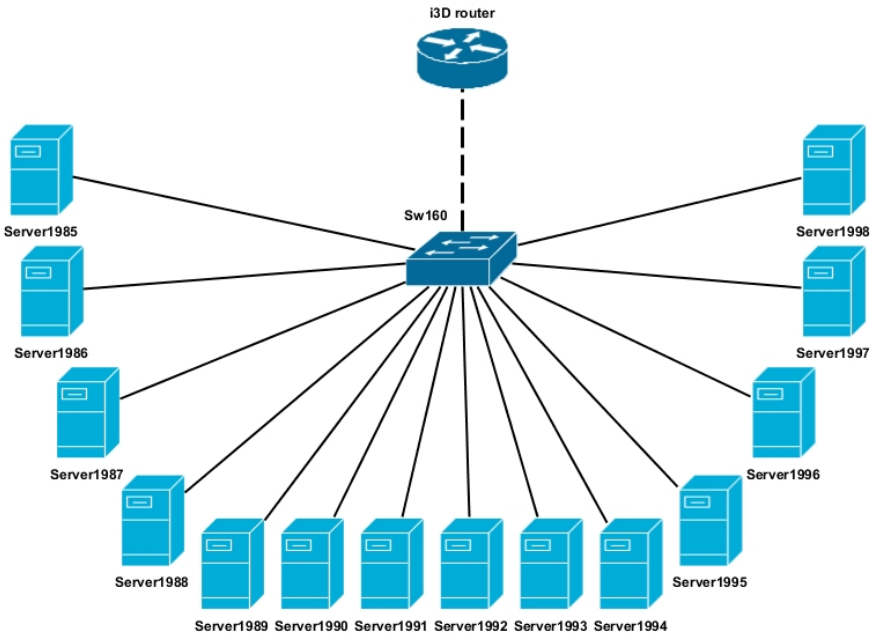
Incremental back-up = Een vorm van back-uppen waarbij er eerst een full back-up wordt gemaakt de volgende back-ups bevatten alleen de veranderingen.

Non-disruptive back-up = Een vorm van back-up creatie waarbij de instance live blijft. Er wordt een kopie van de instance gemaakt en deze wordt geback-upt.

DMZ = Demilitarized zone. Een onderdeel van een (bedrijfs)netwerk waar componenten in staan die verbonden zijn met het veilige (trusted) network en met het “onveilige” (untrusted) openbare netwerk.

1 Netwerk toplogie

In dit subhoofdstuk wordt de netwerktopologie gedetailleerd weergeven. Aan de technische netwerktopologie kan niet zo veel informatie ontleent worden. Dit komt door het feit dat alle servers op één switch zitten aangesloten terwijl alle servers wel meerdere verbindingen hebben. De meeste servers zitten ook in meerdere netwerken. De technische netwerktopologie is te zien in afbeelding 1. In deze afbeelding staan alleen alle verbindingen met de switch aangegeven, een lijn van server naar switch kan echter ook meerdere connecties voorstellen. De verbinding tussen de switch en de router is een top of rack verbinding van 10Gb/s.



Afbeelding 1: De technische netwerktopologie.

Met deze reden is er ook een wat meer concreet ontwerp gemaakt waar duidelijk staat aangegeven in welk netwerk elke verbinding zich bevindt, welke interface voor de betreffende verbinding wordt gebruikt en de IP-adressen per verbinding. Deze variant is te zien in afbeelding 2. Onderaan naast de network node staat ook een kader met een IP-adres range. Deze range geeft aan welke IP-adressen gebruikt kunnen worden als Floating IP binnen OpenStack. De network node deelt deze floating IP-adressen zelf uit op verzoek.

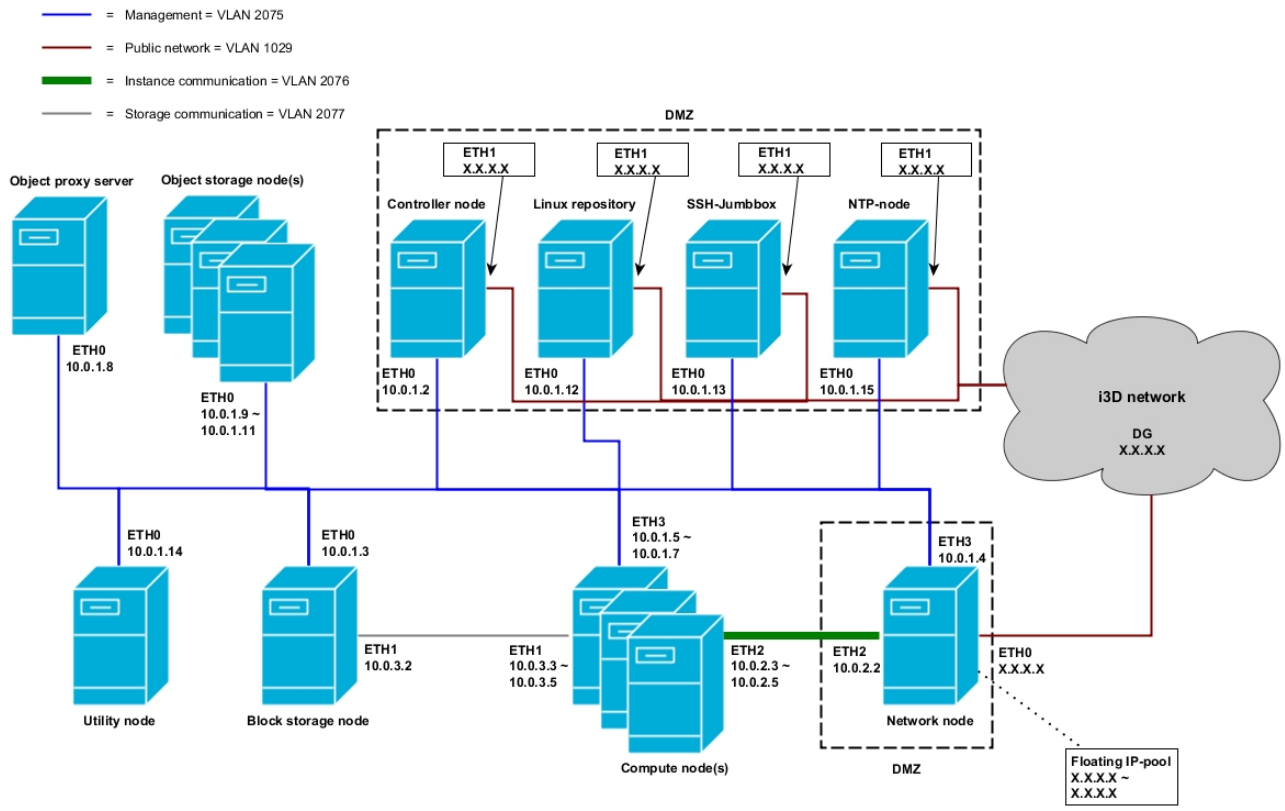
1.1 Systemen en hardware

Alle servers die gebruikt zijn in het ontwerp zijn allemaal van het type HP DL120 G6. In deze servers is 8GB unbuffered RAM aanwezig. Tevens hebben alle servers de beschikking over 1 processor van het type Intel Xeon X3430. Het is een quad core processor zonder hyper threading met de cores draaiend op 2,4Ghz. De servers hebben de beschikking over 2 ingebouwde NIC's.
Door het lage aantal netwerkinterfaces die standaard aanwezig zijn op de DL120 servers is er gekozen om bij de volgende nodes het aantal interfaces met twee uit te breiden met een uitbreidingskaart: Network node en de compute nodes. Deze uitbreidings kaart is van Broadcom en van het type “D43042 PCIe”
De block storage node is uitgebreid met een RAID-kaart om te voldoen aan een systeemeis. Deze RAID-kaart biedt aan vier schijven connectiviteit. De vier schijven moeten in RAID10 worden gezet. De RAID-kaart is van het type “HP Smart Array P410/256 2-ports Int PCIe x8 SAS Controller”

De switch die gebruikt wordt in het ontwerp is van het type Dell Powerconnect 5548. Het is een 48 ports switch. Elke port is een gigabit verbinding. Tevens wordt er enkel gebruik gemaakt van CAT5e bekabeling. Deze koper bekabeling kan een snelheid van 1Gb/sec behalen.

1.2 IP-planning en switch indeling

Op de centrale switch in dit project moeten alle porten in verschillende VLAN's worden ingedeeld. Omdat de netwerken in het proof of concept allemaal intern zijn behalve het i3D-netwerk is er gekozen voor makkelijke IP-reeksen: het management netwerk heeft de reeks 10.0.1.X/24, de instance-tunnel heeft de IP-reeks 10.0.2.X/24 en het (block) storage communication netwerk heeft de reeks 10.0.3.x/24. De VLAN-nummers 2075, 2076 en 2077 zijn gekozen omdat het de eerst voorkomende vrije VLAN's waren binnen het netwerk van i3D.
Om met het i3D-netwerk te verbinden en floating-IP's uit te kunnen delen aan instances is er een /26 pool van IPv4-adressen beschikbaar gemaakt aan het project. Het VLAN-nummer van deze pool is 1029 en is gekozen omdat dit het eerst voorkomende vrije VLAN-nummer was. Het VLAN 1029 is het OpenStack VLAN op het i3D-netwerk.



Afbeelding 2: De netwerktopologie van het proof of concept.

1.3 Harddisk management

De nodes/servers binnen OpenStack hebben de volgende aantallen HDD's aan boord:

Tabel 1: Het aantal HDD's in de verschillende servers.

Node	Aantal HDD's	Opslagruimte HDD's
Controller node	1	250GB
Compute node	1	250GB
Network node	1	250GB
Block storage node	4 (In RAID10)	250GB
Utility node	1	250GB
Swift proxy server	1	250GB
Swift storage node	4	250GB
SSH-jumpbox	1	250GB
Repository node	1	250GB
NTP-node	1	250GB

2. Applicaties en configuratie

De applicaties die worden gebruikt bij dit project maken voornamelijk deel uit van OpenStack. Zo kan OpenStack zelf gezien worden als een grote applicatie, echter klopt dit niet helemaal want OpenStack bestaat uit veel kleine componenten/applicaties. Bij het opzetten van het proof of concept gaat er niet van alle componenten gebruik worden gemaakt. De volgende OpenStack componenten worden wel gebruikt:

- OpenStack-Keystone (authenticatie en autorisatie)
- OpenStack-Glance (image-service)
- OpenStack-Nova (compute-service)
- OpenStack-Neutron (networking)
- OpenStack-Horizon (Grafisch dashboard)
- OpenStack-Cinder (Block storage)
- OpenStack-Swift (Object storage)
- OpenStack-Ceilometer (Metering)
- OpenStack-Rally (Benchmarking)

Naast deze componenten zijn er ook applicaties nodig waar de OpenStack componenten van af hangen. Als eerste is er het OS die op de nodes moet komen te draaien. Het OS is voor het gehele proof of concept vastgesteld op Ubuntu 14.04 server edition.
Om data op te slaan hebben alle componenten een database nodig. De database die gebruikt wordt om componenten data in te laten opslaan is MySQL. Echter wordt er ook NoSQL geïnstalleerd. NoSQL is nodig om OpenStack-Ceilometer functioneel te krijgen.
Het onderdeel van OpenStack waar de autorisatie en authenticatie wordt verzorgt, Keystone, is afhankelijk van een Apache2 server. Daarom wordt er ook een Apache2 server geïnstalleerd.
Ook is het onmisbaar in een OpenStack omgeving om een functionerende messagequeue te hebben ingericht. Als AMQP-messagequeue is er gekozen voor RabbitMQ.

Tabel 2: De verschillende nodes binnen het PoC met de services die zij huisvesten.

Node	Welke services	
Controller node	MySQL, NoSQL, RabbitMQ, Ceilometer, Glance, Keystone, API-endpoints, Horizon	
Compute node	Nova, Ceilometer	
Network node	Neutron	
Cinder node	Cinder	
Object storage node	Swift	
Proxy server	Swift	
SSH-jumpbox	SSH	
NTP-node	NTP	
Repository node		

2.1 Keystone

Afhankelijkheden:
- MySQL
- Apache2
- Memcache

Installatie locatie: Controller node

Nodige packages (Controller node):
- keystone
- apache2-utilslibapache2-mod-wsgi
- memcached
- python-memcache

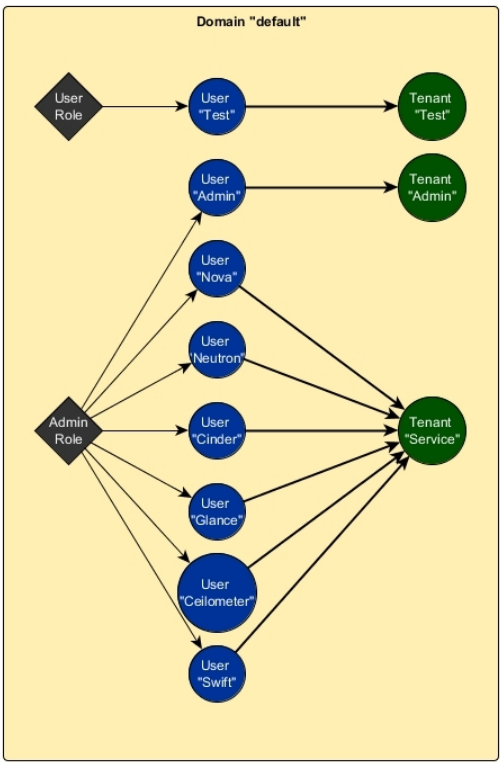
Endpoints:
- Keystone public: X.X.X.X:5000 (public IP van de controller)
- Keystone internal: 10.0.1.2:5000 (Internal IP van de controller)
- Keystone admin: 10.0.1.2:35357 (Internal IP van de controller)

Users, roles, tenants en domains (zie afbeelding 3):

Roles:
- admin
- user

Users:
- admin: role admin, tenant admin
- test: role user, tenant test
- glance: role admin, tenant service
- nova: role admin, tenant service
- neutron: role admin, tenant service
- cinder: role admin, tenant service
- ceilometer: role admin, tenant service

Tenants:
- service
- test
- admin



Afbeelding 3: De entiteiten die in het proof of concept aanwezig moeten zijn.

2.1.1 MySQL

Er moet in MySQL een nieuwe database worden aangemaakt. De naam van de database is “keystone”. Deze database moet volledig toegankelijk zijn voor de user “keystone”.

2.1.2 Apache2

De Apache2 service wordt gebruikt als ontvangstpunt en zendpunt van HTTP-requests en replies binnen OpenStack. Deze service moet gaan luisteren op port 5000 en 35357.

2.1.3 Memcached

De Memcache service wordt gebruikt als cache voor het Keystone token-backend. Deze service moet gaan luisteren op port 11211.

2.2 Glance

Afhankelijkheden:
- MySQL

Installatie locatie: Controller node

Nodige packages (Controller node):
- glance
- python-glanceclient

Endpoints:
- Glance public: X.X.X.X:9292 (public IP van de controller)
- Glance internal: 10.0.1.2:9292 (Internal IP van de controller)
- Glance admin: 10.0.1.2:9292 (Internal IP van de controller)

2.2.1 MySQL

Er moet in MySQL een nieuwe database worden aangemaakt. De naam van de database is “glance”. Deze database moet volledig toegankelijk zijn voor de user “glance”.

2.3 Nova

Afhankelijkheden:
- MySQL
- KVM (Compute node)

Installatie locatie: Controller node, Compute node(s)

Nodige packages (Controller node):
- nova-api
- nova-cert
- nova-conductor
- nova-consoleauth
- nova-novncproxy
- nova-scheduler
- python-novaclient

Nodige packages (Compute node)

- nova-compute
- sysfsutils

Endpoints:

- Nova public: X.X.X.X:8774 (public IP van de controller)
- Nova internal: 10.0.1.2:8774 (Internal IP van de controller)
- Nova admin: 10.0.1.2:8774 (Internal IP van de controller)

2.3.1 MySQL

Er moet in MySQL een nieuwe database worden aangemaakt. De naam van de database is “nova”. Deze database moet volledig toegankelijk zijn voor de user “nova”.

2.3.2 KVM

KVM wordt geïnstalleerd op het moment dat nova-compute wordt geïnstalleerd.

2.3.3 Flavors

Nova heeft al een standaard lijstje met flavors. Echter moet een worden toegevoegd in het proof of concept, nl: “m1.smaller 6 1024 20 1” → naam, ID, GB RAM, GB block storage, VCPU.

2.4 Neutron

De Neutron service zorgt binnen OpenStack voor netwerk functionaliteit aan instances en het zorgt voor SDN functionaliteit.
De service moet worden geïnstalleerd op een aparte network node. Compute nodes moeten wel packages krijgen om virtuele switches op te zetten met openvswitch. Tussen de network node en de compute nodes moet een GRE-tunnel worden opgezet om “virtuele” VLAN's te creëren. Met OpenvSwitch worden de nodige virtuele bridges gecreëerd, hiermee kunnen instances aan netwerken en virtuele netwerken worden gekoppeld.

Afhankelijkheden:

- MySQL (Controller node)
- GRE(-tunnel)
- OpenvSwitch

Installatie locatie: Controller node, Compute node(s), Network node

Nodige packages (Controller node):

- neutron-server
- neutron-plugin-ml2
- python-neutronclient

Nodige packages (Compute node):

- neutron-plugin-ml2
- neutron-plugin-openvswitch-agent

Nodige packages (Network node):

- neutron-plugin-ml2
- neutron-plugin-openvswitch-agent
- neutron-l3-agent
- neutron-dhcp-agent
- neutron-metadata-agent

Endpoints:

- Neutron public: X.X.X.X:9696 (public IP van de controller)
- Neutron internal: 10.0.1.2:9696 (Internal IP van de controller)
- Neutron admin: 10.0.1.2:9696 (Internal IP van de controller)

2.4.1 MySQL

Er moet in MySQL een nieuwe database worden aangemaakt. De naam van de database is “neutron”. Deze database moet volledig toegankelijk zijn voor de user “neutron”.

2.4.2 GRE-tunnel

Om tunnels op te zetten met GRE moet er eerst een range van ID's worden opgegeven die GRE kan uitgeven. Met deze ID's worden de nodes die aangesloten zijn op het tunnel netwerk geïdentificeerd. Op basis van deze ID's worden de packets naar de goede destination gestuurd. De tunnel range moet zijn: 1:1000.

2.4.3 OpenvSwitch

Met OpenvSwitch moeten een bridge handmatig worden aangemaakt. Deze bridge is:

- br-ex: De bridge moet “aangesloten” worden op de interface die de network node aan het i3D netwerk verbindt.

Tevens moet aan ML2 (Layer2 plugin) worden meegegeven wat de IP-adressen zijn van de interfaces op het tunnel netwerk. Deze adressen zijn: 10.0.2.2, 10.0.2.3, 10.0.2.4, 10.0.2.5.

2.4.4 Floating IP's

Om Neutron floating IP-adressen te kunnen laten uitdelen moet er eerst een virtueel subnet worden aangemaakt binnen OpenStack Neutron. Dit subnet moet onderdeel zijn van een virtueel netwerk die gekoppeld zit aan de br-ex bridge.
Het virtuele i3D netwerk is flat. De floating IP-range die het “ext-net” subnet krijgt is: X.X.X.X/26 ~ X.X.X.X/26.

2.5 Horizon

Met horizon wordt er een OpenStack dashboard geïnstalleerd. Standaard worden alle hosts (*['*',]*) toegelaten tot de dashboard service. Om de service meer responsief te maken wordt er gebruik gemaakt van memcached. Dit is de memcached installatie op de controller node.

Afhankelijkheden:

- Memcached (controller node)

Installatie locatie: Controller node

Nodige packages (Controller node):
- openstack-dashboard

2.6 Cinder

Cinder is de service die block storage mogelijk maakt. Alle blocks/volumes worden vanuit Cinder aangeboden binnen de volume group: “cinder-volumes”. Op de block storage node is de opzet zo dat er met een hardware RAID-controller een RAID10-opstelling wordt neergezet. Het RAID-array wordt voorzien van de besturende eenheid (OS en services) en de lege ongepartitioneerde ruimte moet als LVM-partitie worden geformatteerd.

Afhankelijkheden:
- MySQL
- iSCSI

Installatie locatie: Controller node, Block storage node

Nodige packages (Controller node):
- cinder-api
- cinder-scheduler
- python-cinderclient

Nodige packages (Block storage node)
- lvm2
- cinder-volume
- python-mysqldb
- cinder-backup

Endpoints:
- Cinder public: X.X.X.X:8776/v1/ (public IP van de controller)
- Cinder internal: 10.0.1.2:8776/v1/ (Internal IP van de controller)
- Cinder admin: 10.0.1.2:8776/v1/ (Internal IP van de controller)
- Cinder publicv2: X.X.X.X:8776/v2/ (public IP van de controller)
- Cinder internalv2: 10.0.1.2:8776/v2/ (Internal IP van de controller)
- Cinder adminv2: 10.0.1.2:8776/v2/ (Internal IP van de controller)

2.6.1 MySQL

Er moet in MySQL een nieuwe database worden aangemaakt. De naam van de database is “cinder”. Deze database moet volledig toegankelijk zijn voor de user “cinder”.

2.6.2 Cinder node

De Cinder block storage node wordt uitgerust met een RAID-kaart. De vier schijven die in de node aanwezig zijn worden tot één groot geheel gemaakt met RAID10. Op deze RAID10 opstelling komt het OS te staan volgens de partitie indeling in hoofdstuk 2.x. Het lege gedeelte dat op de schijf overblijft moet worden opgezet naar LVM. De LVM partitie komt in de volume-group “cinder-volumes” te staan.

2.6.3 iSCSI

De verbinding tussen de block storage node en de compute nodes volgt het iSCSI protocol. Als iSCSI-target-framework wordt TGT gebruikt.

2.7 Rally

Met Rally kunnen er tests en benchmarking worden uitgevoerd op de OpenStack cloud.

Afhankelijkheden: N.V.T.

Installatie locatie: Utility node

Nodige packages (Controller node):

- libssl-dev
- libffi-dev
- python-dev
- libxml2-dev
- libxslt1-dev
- libpq-dev
- git
- python-pip

2.8 Ceilometer

Cinder is de service die block storage mogelijk maakt. Alle blocks/volumes worden vanuit Cinder aangeboden binnen de volume group: “cinder-volumes”.

Afhankelijkheden:

- NoSQL (Controller node)

Installatie locatie: Controller node, Block storage node, Object storage node, block storage node, proxy server, network node

Nodige packages (Controller node):

- ceilometer-api
- ceilometer-collector
- ceilometer-agent-central
- ceilometer-agent-notification
- ceilometer-alarm-evaluator
- ceilometer-alarm-notifier
- python-ceilometerclient

Nodige packages (Compute nodes)

- ceilometer-agent-compute

Endpoints:

- Ceilometer public: X.X.X.X:8777 (public IP van de controller)
- Ceilometer internal: 10.0.1.2:8777 (Internal IP van de controller)
- Ceilometer admin: 10.0.1.2:8777 (Internal IP van de controller)

2.8.1 NoSQL

Er wordt voor het Ceilometer component gebruik gemaakt van een NoSQL-database. Deze database luistert op port 27017. Als database user moet “ceilometer” aangemaakt worden.

2.9 Swift

Met Swift kan object storage op worden gezet. Er is gekozen voor een opzet met één object proxy node en drie object storage nodes. De account-service, object-service en container-service komen op elke object storage node te staan.
Elke object storage node is uitgerust met vier HDD's. Op de eerste schijf staat de besturende eenheid (OS en de Swift-services). De overige drie schijven, in elke object storage node, zijn leeg en moeten geformatteerd worden als XFS-filesystem.

Afhankelijkheden:
NVT

Installatie locatie: Proxy server, object storage nodes

Nodige packages (Proxy server):

- swift
- swift-proxy
- python-swiftclient
- python-keystonemiddleware
- memcached

Nodige packages (Object storage nodes):

- swift-account
- swift-container
- swift-object
- xfsprogs

Endpoints:

- Swift public: X.X.X.X:8080 (public IP van de proxy server)
- Swift internal: 10.0.1.8:8080 (Internal IP van de proxy server)
- Swift admin: 10.0.1.8:8080 (Internal IP van de proxy server)

2.10 NTP-node

De NTP-node die zich in het netwerk bevindt moet kunnen synchroniseren met de NTP-server van i3D.
Op deze node is de service “ntp” geïnstalleerd. De NTP-node maakt verbinding en synchroniseert met de interne NTP-server van i3D.

2.11 Repository node

De repository node wordt opgezet met de packets “apt-mirror” en “proftp”. Met “apt-mirror” wordt de repository opgezet en de node gesynchroniseerd met een bestaande mirror. Via “proftp” kunnen de nodes op het management netwerk hun updates ophalen bij de repository node.
Alle systemen zijn opgezet met Ubuntu 14.04 LTS server edition 64-bit. Dit betekend dat er op de repository node ook alleen de packets voor deze release aanwezig hoeven te zijn.

2.12 SSH-jumpbox

Via de jumpbox kunnen alle nodes bereikt worden via het management netwerk. Om niet het gehele management netwerk open te moeten laten om het te kunnen managen via SSH wordt er een SSH-jumpbox gemaakt. Dit zorgt tevens voor een verbeterde security.
Op de jumpbox hoeven geen packages geïnstalleerd te worden. Dit komt omdat de SSH-service standaard geïnstalleerd staat op Ubuntu 14.04.
Wel moeten er op deze node SSH-keys worden geactiveerd, dit wordt gedaan door “SSH authorized_keys”. De SSH-keys moeten worden geactiveerd voor de “admin1997” user. Na het instellen van de key authenticatie moet de authenticatie via username/password in SSH worden uitgeschakeld.

2.13 HDD-partitionering nodes

De partitionering voor de controller nodes is als volgt gepland:

Tabel 3: de partitionering van de HDD in de controller node.

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	65536 MB	ext3	Belangrijk dat er goed gelogd kan blijven worden. Om in het proof of concept Glance images te kunnen uploaden/downloaden.
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering voor de compute nodes is als volgt gepland:

Tabel 4: de partitionering van de HDD in de compute node.

Device	Mount	Size	Filesystem	Comment

/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	65536 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering voor de network nodes is als volgt gepland:

Tabel 5: de partitionering van de HDD in de network node.

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	65536 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering voor de block storage nodes is als volgt gepland:

Tabel 6: de partitionering van de HDD in de Cinder node.

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	65536 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize
Geen	Geen	Rest (~420GB)	Geen	Deze lege, ongepartitioneerde ruimte is bedoeld voor de LVM partitie van Cinder.

De partitionering voor de Utility node is als volgt gepland:

Tabel 7: de partitionering van de HDD in de utility node.

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering van de Swift proxy server is als volgt:

Tabel 8: de partitionering van de HDD in de Swift proxy server.

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	8192 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

De partitionering van de Swift storage servers is als volgt:

Tabel 9: de partitionering van de HDD in de Swift storage node.

Device	Mount	Size	Filesystem	Comment
/dev/sda1	/boot	2048 MB	ext2	
/dev/sda2	/	Rest	ext3	
/dev/sda3	/usr	8192 MB	ext3	
/dev/sda4	/tmp	2048 MB	ext3	
/dev/sda5	/var	8192 MB	ext3	
/dev/sda6	Swap	8192 MB	ext3	100% RAMsize

2.14 Configuratie

De gehele OpenStack omgeving wordt geïnstalleerd in de region: “RegionOne”.

2.14.1 De users per node

Tabel 10: De users per node.

username	Username	Node	Functie node
Admin1986	root	server1986	Controller node
Admin1987	root	server1987	Block storage node
Admin1988	root	server1988	Network node
Admin1989	root	server1989	Compute node
Admin1990	root	server1990	Compute node
Admin1991	root	server1991	Compute node
Admin1992	root	server1992	Proxy server
Admin1993	root	server1993	Object storage node
Admin1994	root	server1994	Object storage node
Admin1995	root	server1995	Object storage node
Admin1996	root	server1996	Repository node
Admin1997	root	server1997	SSH-jump node

Admin1998	root	server1998	NTP-node

3. Back-up plan

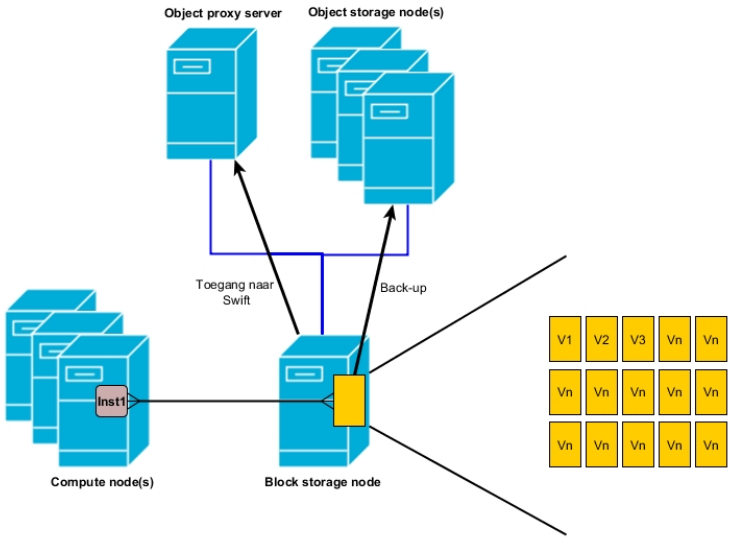
In dit hoofdstuk worden twee verschillende soorten back-up beschreven. Als eerste is er de back-up voor klanten. Als service in de achtergrond moet er een back-up worden gedraaid. In geval van calamiteit bij de gebruiker kan deze back-up worden aangesproken.
Aan de andere kant is er ook de systeem back-up. Dit is de back-up van de OpenStack services zelf. Voor de systeem back-up is een vaste procedure die OpenStack voorschrijft.

3.1 Klant back-up

Het back-up plan voor klanten binnen dit project is niet van zeer groot belang. Het punt is dat de use case voor een eventuele cloud gebaseerd op het eindproduct van dit project een low-cost oplossing is. Bij gesprekken met de opdrachtgever is naar voren gekomen dat hij aangetoond wil hebben dat er wel back-up kan plaatsvinden voor de opslag van users. De back-up moet echter “achter de schermen” gebeuren. In een eventuele cloud oplossing die uit dit PoC komt zijn de gebruikers zelf verantwoordelijk voor hun data. Er moet echter wel van de data van elke gebruiker met X tijd interval een back-up worden gedraaid. Op het moment dat de gebruiker zijn of haar data kwijt is door uitval en zelf geen back-up heeft kan er tegen een vergoeding aanspraak worden gemaakt op de back-up uit de cloud zelf.

Het back-up plan is als volgt:

Door het gebruik van “cinder backup” zijn er back-ups mogelijk van de Cinder block storage naar de Swift object storage. De Swift storage is een opslag gebaseerd op objecten. Van nature is Swift (en iedere object storage) ook high available. Dit komt omdat er standaard twee replica's worden gemaakt van alle data: de data bestaat drie keer binnen Swift.



Afbeelding 4: Het uitvoeren van een incremental non-disruptive back-up binnen OpenStack.

De data van users op de blocks zal non-disruptive worden geback-upt. Dit betekend dat de services door blijven draaien: een live back-up van de Cinder volumes. De back-ups kunnen worden gepland op een rustig moment, bijvoorbeeld 's nachts. Om de week of om een aantal dagen kunnen de back-ups worden gedraaid.

Het gaat hierbij om incremental non-disruptive back-ups. De eerste keer dat er een back-up van een volume wordt gedraaid zal dit een full back-up zijn. Hierna worden er constant incremental back-ups gedraaid. Op afbeelding 4 is te zien hoe deze back-up wordt gemaakt.

Klanten kunnen hiernaast ook zelf een back-up van hun data maken. Er zijn genoeg pakketten op de markt om dat doel te verwezenlijken.

3.2 Systeem back-up

Ook is er een back-up plan voor de systeembestanden van OpenStack. Er kan een back-up worden gemaakt van alle OpenStack-componenten (tabel 11) door de files van de betreffende services samen met dumps van de MySQL en NoSQL databases in een map te plaatsen. De bestanden van de betreffende services kunnen worden gevonden in de mappen: /etc/[servicenaam], /var/lib/[servicenaam] en /var/log/[servicenaam]. Vervolgens moet de map met alle service bestanden en database dumps worden ingepakt als TAR bestand en opgeslagen worden op een veilige plaats. In het proof of concept is de “veilige plaats” de Swift storage.

Tabel 11: De bestanden die moeten worden geback-upt bij een systeemback-up.

Node	Component/service	Directories
Controller	Nova	/etc/nova /var/log/nova /var/lib/nova
Controller	Glance	/etc/glance /var/log/glance /var/lib/glance/image-cache
Controller	Keystone	/etc/keystone /var/log/keystone /var/lib/keystone
Controller	Cinder	/etc/cinder /var/log/cinder /var/lib/cinder
Controller	Neutron	/etc/neutron /var/log/neutron /var/lib/neutron
Controller	Horizon	/etc/openstack-dashboard
Controller	Ceilometer	/etc/ceilometer
Controller	Iptables	/etc/iptables_contr
Compute 1	Nova	/etc/nova /var/log/nova /var/lib/nova/buckets /var/lib/nova/CA /var/lib/nova/images /var/lib/nova/keys /var/lib/nova/networks /var/lib/nova/tmp
Compute 2	Nova	/etc/nova /var/log/nova /var/lib/nova/buckets /var/lib/nova/CA /var/lib/nova/images /var/lib/nova/keys /var/lib/nova/networks /var/lib/nova/tmp
Compute 3	Nova	/etc/nova /var/log/nova /var/lib/nova/buckets /var/lib/nova/CA /var/lib/nova/images /var/lib/nova/keys /var/lib/nova/networks /var/lib/nova/tmp
Block storage node	Cinder	/etc/cinder /var/log/cinder /var/lib/cinder
Object proxy-server	Swift	/etc/swift
Object storage server 1	Swift	/etc/swift

		/etc/fstab /etc/rsyncd.conf /etc/default/rsync
Object storage server 2	Swift	/etc/swift /etc/fstab /etc/rsyncd.conf /etc/default/rsync
Object storage server 3	Swift	/etc/swift /etc/fstab /etc/rsyncd.conf /etc/default/rsync
NTP-node	NTP	/etc/ntp.conf /etc/iptables_ntp
SSH-jumpnode	SSH	/etc/ssh ~/.ssh/authorized_keys /etc/iptables_ntp
Repository node	Repository	/etc/apt/mirror.list /etc/proftpd /etc/iptables_ntp

Ook moet er een back-up worden gemaakt van de databases. In tabel 8 is te zien welke databases dit betreft en welk commando er moet worden gebruikt.

Tabel 12: De databases die moeten worden geback-upt.

database	**Command**
cinder	mysql -u root -p cinder < cinder.sql
glance	mysql -u root -p glance < glance.sql
keystone	mysql -u root -p keystone < keystone.sql
neutron	mysql -u root-p neutron < neutron.sql
nova	mysql -u root -p nova < nova.sql
Ceilometer	Mongorestore –h 10.0.1.2 Ceilometer_dump

4. Beveiligingsmaatregelen

Om te zorgen dat de security systeemeis tot uitvoering wordt gebracht zijn er beveiligingsmaatregelen nodig. De eerste maatregelen zijn al te zien in de netwerk topologie (afbeelding 2). In dit subhoofdstuk worden alle beveiligingsmaatregelen besproken die uitgevoerd moeten worden bij het opzetten van het proof of concept.

Er moet gebruik gemaakt worden van een DMZ. Alle onderdelen binnen de infrastructuur die verbinding moeten maken met het i3D netwerk staan in de DMZ. Alle nodes in de infrastructuur, zeker de nodes in de DMZ, hebben een Iptables firewall die alle porten dichtzet behalve de porten die nodig zijn. Door het implementeren van een DMZ wordt er een extra compartiment gevormd om de interne netwerken van het proof of concept. In subhoofdstukken 4.1.2 ~ 4.6 worden de verschillende nodes besproken die in het DMZ staan.

4.1 Users en passwords

Op de meeste fysieke nodes zijn er twee users aanwezig. Dit zijn de root user en een admin user. De admin user is de user die gebruikt wordt voor alle werkzaamheden. De naam van de admin user komt overeen met de server waar hij actief op is, bijvoorbeeld: de admin op server1988 heet “admin1988”. De root user wordt niet gebruikt. De admin users kunnen door middel van SUDO hun privileges liften. De admin users moeten ook deel uitmaken van de admin group.
De wachtwoorden worden opgeslagen in een Keepass database. Tevens zijn de passwords die worden gebruikt binnen dit PoC 20 karakters lang. De passwords worden gegenereerd met:
< /dev/urandom tr -dc _A-Z-a-z-0-9 | head -c${1:-20};echo;

4.2 Updates

Alle fysieke nodes in het netwerk kunnen steeds worden voorzien van de laatste updates door de repository node. Hierdoor blijft de omgeving up-to-date.

4.3 VLAN's en VLAN-tunneling

Op afbeelding 3 is er ook te zien dat er VLAN's worden gebruikt in de opbouw van het proof of concept. Het gebruik van deze VLAN's heeft tot gevolg dat de verschillende netwerken onderling niet kunnen communiceren (geen broadcast-domain onderling). Communicatie uit het ene netwerk kan zo niet naar een ander netwerk binnen OpenStack, dit zorgt voor een betere performance en beveiliging. Het gaat hier om de afscheiding van de fysieke netwerken. De verschillende netwerken zijn: management, public, block storage en VM-tunnel.
De virtuele netwerken binnen OpenStack worden gescheiden door GRE-tunnels. OpenStack maakt tunnels aan voor alle tenants die actief zijn binnen de cloud. De tunnels lopen naar alle compute nodes waar de betreffende tenant actief is. Het verkeer wordt verstuurd met een segmentation-ID. Tussen de tunnels is er geen uitwisseling van communicatie mogelijk. De GRE-tunnels worden door OpenStack zelf opgezet, er moet wel worden aangegeven wat de tunnel-ID-range is (1:1000).

4.4 HTTPS en XSS-protectie voor OpenStack Horizon

Om sommige aanvallen op OpenStack Horizon en op de gebruikers te voorkomen wordt er gebruik gemaakt van HTTPS in plaats van HTTP voor OpenStack Horizon (dashboard). Bij het invoeren van de HTTPS verbinding wordt ook direct gebruik gemaakt van HSTS. HSTS kan worden ingevoerd met het omzetten van één parameter. Met HSTS wordt er afgedwongen dat een gebruiker HTTPS gebruikt.
De aanvallen waar tegen wordt beveiligd met deze maatregelen zijn in ieder geval “cookie hijacking” en

“downgrade attacks”. Met cookie hijacking kan de sessie van de gebruiker worden overgenomen door een aanvaller. De aanvaller kan vervolgens toegang krijgen tot het account van de gebruiker. Met downgrade attacks kan een oudere versie van een protocol worden gebruikt terwijl het nieuwere protocol actief is (HTTP I.P.V. HTTPS). Deze downgrade attack wordt voorkomen door HSTS.

4.5 Firewalls en Security groups

Er zijn binnen het proof of concept een aantal manieren waarmee de toegang van het netwerkverkeer geregeld kan worden. Voor de fysieke machines zijn er de Iptables firewalls, opgenomen in het Ubuntu OS.
De firewalls moeten alle porten afsluiten behalve de porten die worden gebruikt.
Voor toegangscontrole naar de instances zijn er de OpenStack security groups. De security groups werken op dezelfde manier als een IP-tables firewall, alleen nu virtueel. Standaard staan alle port naar de instances dicht.

Op het i3D netwerk van het proof of concept zijn de controller node, repository node, NTP-node en de SSH-jump node aangesloten. Voor de porten die open staan op het i3D netwerk moeten extra beveiligingsmaatregelen worden ingevoerd met Iptables. De Iptables staan op de fysieke nodes.

Wat betreft de nodes op het management netwerk, deze moeten wel Iptables rules krijgen. Deze rules zijn echter niet opgenomen in dit ontwerp omdat alle ingaande en uitgaande porten volledig open staan. Dit betekend dat voor elke porten die open moeten staan rules worden aangemaakt zonder extra parameters, zoals:

De default policy is tevens DROP op de interne nodes. Voor alle (interne en DMZ) nodes zijn er standaard rules nodig naast de specifieke rules.

De firewalls (Iptables) worden zo ingesteld dat al het verkeer dat nodig is door kan. Al het overige verkeer wordt standaard gereject.

4.6 Controller node

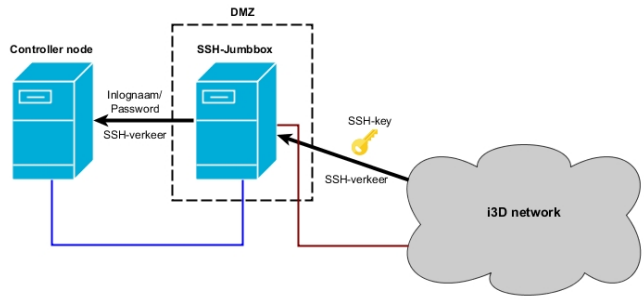
De controller node is het onderdeel wat de (meeste) communicatie verzorgt in OpenStack. De enige communicatie die van het i3D netwerk komt gaat via port 443, de HTTPS port. De Iptables rules aangaande port 443 worden besproken in subhoofdstuk 4.1.5. OpenStack Horizon wordt besproken in subhoofdstuk 4.1.4.

4.7 Repository node

De repository server kan worden gezien als een soort “proxy server”. In plaats van dat de nodes op de interne netwerken een connectie naar het internet nodig hebben om updates en packets binnen te halen wordt er intern een repository opgezet. Deze repository node zorgt voor een tussenlaag.
De repository node update zijn packets van de repository van i3D.

4.8 SSH-jump node

De SSH-jump node is nodig om SSH-verkeer naar het interne netwerk mogelijk te maken voor beheerders van de cloud. De reden dat er een extra node wordt gebruikt voor de connectie in plaats van directe verbinding met het interne netwerk is



Afbeelding 5: De SSH-jump node functioneert als "tussenlaag" in de SSH-communicatie met het interne netwerk.

veiligheid.
Zoals in de configuratie van de SSH-jump node is vermeld wordt de SSH-verbinding naar deze node opgezet door een SSH-key. Een SSH-key is bijna onkraakbaar en zorgt voor een veel betere beveiliging dan inlognaam/password. Vanaf de jump node kan er naar interne nodes wel met inlognaam/passwords worden ingelogd. Op afbeelding 5 is te zien hoe er ingelogd kan worden met SSH binnen het PoC.

4.9 NTP-node

Met NTP kunnen systemen hun datum en tijd afstemmen op een atoomklok. De NTP-server waar mee wordt verbonden en gesynchroniseerd synchroniseert zelf met een andere NTP-server.
De NTP-node is in de DMZ opgenomen omdat deze service ook actief moet zijn binnen het netwerk. De systemen hebben allemaal de juiste tijd nodig om bijvoorbeeld in de logs de goede tijd aan te geven.
Normaal wordt er gesynchroniseerd via een internet verbinding. In dit netwerk is dat niet mogelijk, vandaar dat er intern een NTP-node wordt opgenomen. Deze server wordt dedicated aan NTP gemaakt aangezien er NTP-aanvallen zouden kunnen voorkomen. Deze aanvallen hebben op deze manier minder uitwerking op de werking van de cloud dan wanneer de NTP-service op de controller zou draaien.
De NTP-server synchroniseert alleen maar naar de interne NTP-server van i3D. Het netwerk van i3D wordt vanuit het PoC-netwerk gezien als extern. Door het feit dat de OpenStack NTP-server alleen synchroniseert met de NTP-server van i3D wordt dit opgenomen in de configuratie van de NTP-service. In het configuratiebestand van de NTP-service wordt aangegeven dat er enkel moet worden gesynchroniseerd met de NTP-server van i3D.

Om NTP-reflexion en aplification attacks te voorkomen moet de service up-to-date zijn. Dit betekend dat het versienummer hoger moet zijn dan v4.2.7p26. Om er zeker van te zijn dat deze zwakheid in NTP te misbruiken is zal de monitor in NTP uitgezet moeten worden. Hiermee is geen monlist meer op te vragen op de NTP-server. Een amplification attack in NTP richt zich namelijk op de monlist.

4.10 Network node

De network node bevind zich ook in het DMZ. Bij deze node liggen de zaken echter net iets anders. Op de externe interface van de network node is een virtuele bridge aangemaakt. Via deze bridge kan verkeer van het internet bij de instances komen. Een voorbeeld zou kunnen zijn dat een bezoeker van een website via de network node naar de desbetreffende webserver gerouteerd wordt. Deze webserver wordt gehost op een instance in de cloud.

5. QoS

Binnen het proof of concept worden een aantal QoS-limieten ingesteld om zo nog beter te voldoen aan bepaalde systeemeisen. Tevens kan met de juiste QoS-instellingen de performance een betere balans krijgen. Klanten krijgen hierbij minder kans om de performance scheef te trekken en voor het noisy neighbour effect te zorgen.

5.1 Limits

In OpenStack kunnen extra flavors worden aangemaakt voor de klanten. De klanten kunnen aan de hand van deze flavors nieuwe instances opzetten. Standaard krijgt een flavor aantal vcores, aantal GB geheugen en aantal GB storage mee. Met de flavors kunnen ook extra specs worden meegeven, de zogenaamde “limits”.

In het proof of concept wordt alleen gewerkt met de flavor die ook gebruikt wordt in de winstberekening. Dit is de reden waarom er maar één flavor wordt besproken in dit hoofdstuk.

5.1.1 CPU-weighting

Met de Nova limits kan een vorm van CPU-weighting worden ingesteld. Met deze weights kan er voor worden gezorgd dat iedereen op de node zijn of haar eigen deel krijgt van de CPU-kracht. De CPU-weight kan worden meegegeven met de flavors. Aan de hand van het CPU-share nummer wat elke instance in een domain heeft kan CPU-time worden berekend. Deze vorm van QoS wordt ingezet om het noisy neighbour effect nog verder te kunnen onderdrukken op de compute node.

De *CPU-share* moet worden ingesteld op 10.

5.2 Cinder QoS

Met de Cinder QoS instellingen kan de load op de Cinder node worden gebalanceerd voor meerdere instances. Er moet vanuit worden gegaan dat de instances niet constant load genereren op de Cinder node. Echter voor de hoge pieken in het gebruik waardoor andere users eerder performance slowdowns krijgen moet er een peak/cap worden ingesteld.

Er wordt vanuit gegaan dat de HDD's in de RAID-opstelling 75 MB/s ~ 100 MB/s doen. Dit is het gemiddelde voor standaard 7200 RPM HDD's. Nu staan de disks in een RAID 10 opstelling. Dit betekend dus dat er performance slowdown optreed omdat de RAID-controller een mirror moet bijhouden. Echter betekend die ook dat de snelheid/performance met ongeveer 30% ~ 50% toeneemt in stipe (RAID 0). De snelheid van de schijven komt nu neer op ongeveer 130MB/s ~ 150 MB/s.
Er wordt een cap gezet op 25% van de totale performance van de block storage node HDD's. Deze waarde is gekozen omdat de instances nog genoeg performance naar de HDD's moeten houden maar aan de andere kant moet de cap op de lees- en schrijfsnelheid wel zo laag zijn dat instances andere instances zo min mogelijk performance slowdowns bezorgen.
Bij een cap op 50% van de performance zou er maar één instance “zijn deel” (50%) vol kunnen trekken en blijft er voor de andere instances niks over. Als er dan nog een instance komt die ook hoge load genereert is er alsnog sprake van slowdowns voor de andere gebruikers. Bij 33% kunnen 3 personen hoge load genereren, dit is alsnog vrij weinig. Daarom is er gekozen voor een cap op 25%. Een cap op 25% op het totaal van 130 MB/s betekend alsnog een peak snelheid van ongeveer 33 MB/s.

Met een snel testje is gekeken wat de snelheid van de RAID-opstelling is. Dit testje is alleen bedoeld om te zien of mijn berekening ongeveer klopte, en dat deed het wel:

```
#sudo hdparm -Tt /dev/sda

Timing cached reads:   17912 MB in  2.00 seconds = 8964.72 MB/sec
Timing buffered disk reads: 216 MB in  2.1241 seconds =  101.69 MB/sec

#dd if=/dev/zero of=/tmp/output bs=4k count=100000; rm -f /tmp/output

4096000000 bytes (4.0 GB) copied, 25.0695 s, 159 MB/s
dd if=/dev/zero of=/tmp/output bs=4k count=100000
```

De read_bytes_sec parameter krijgt een waarde van 34603008 B/s.
De write_bytes_sec parameter krijgt een waarde van 34603008 B/s.

5.3 Neutron QoS

Binnen het Neutron component kan de QoS voor de instances worden ingesteld op de virtuele porten. Dit betekend dat de aansluiting van een instance op een virtueel netwerk afgeknepen kan worden met deze rules. Er zal eerst een policy aan moeten worden gemaakt. Deze policy krijgt dan de naam “QoS-Neutron”. Vervolgens krijgt de policy de volgende rule:

```
QoS-Neutron
--max-kbps 10000
--max-burst-kbps 1000
```

De rule heeft dezelfde naam als de policy. De bandwidth die met deze rule wordt gegeven is 10Mb/s. Vervolgens moet deze policy worden toegepast op elke virtuele port. Deze QoS-rule wordt ingezet omdat het binnen een cloud omgeving handig is om personen een vaste netwerksnelheid te geven. Zonder deze vorm van QoS kan één klant de instance-tunnel al voltrekken aangezien de verbinding dan best-effort is.

5.4 Oversell

Door het toepassen van oversell of overcommit kunnen fysieke resources meerdere keren worden verkocht als virtuele resources. De overcommit moet in het proof of concept komen te liggen op: 2:1 voor vcores en 1:1 voor RAM.
De HP DL120 G6 servers hebben fysiek een CPU met 4 cores en 8GB RAM. Met de overcommit betekend dit dat er 8 vcores en 8GB RAM verdeeld kan worden over verschillende instances.

Bijlage XII: Testverslag

Testverslag

Onderzoek uitvoeren naar hergebruik van oude datacenter servers om een low-cost cloud oplossing te creëren bij i3D

Auteur: Jeroen Holtkamp
Instelling: Haagse hogeschool
Stagebedrijf: i3D
Plaats: Rotterdam
Datum: 27-03-2016

Versiebeheer

Versie	Datum	Wijziging
0.1	28-03-2016	Eerste layout
0.2	04-04-2016	Testplan en testrapportage voor de proefopstelling toegevoegd. Grammatica- en spelfouten verbeterd.
1.0	22-05-2016	

Documentdistributie

Naam	Rol
Jan Dirk Schagen	Eerste examinator
Marinus Maris	Tweede examinator, SLB'er
Rick Sloot	Opdrachtgever
Rick Sloot	Bedrijfsmentor

Inleiding

In dit document worden de tests beschreven die uitgevoerd zijn binnen het project. De tests zijn van belang omdat ze aantonen dat de systeemeisn van de opdrachtgever ook daadwerkelijk functioneel zijn.

De tests hebben plaatsgevonden in twee fases: de architectuurfase en de testfase. In de architectuurfase zijn alleen de performance systeemeisen getest. De performance eisen zijn eerder getest dan alle andere eisen omdat de performance eisen belangrijk zijn in het proof of concept. Op basis van de uitkomsten van de tests konden nog aanpassingen plaatsvinden in de architectuur en infrastructuurfases.
Omdat de tests in twee fases plaats hebben gevonden is de opbouw van dit document ook aangepast aan de twee verschillende fases. Ook is dit document het testplan en testrapport in één, eerst worden de testomgevingen besproken, vervolgens de testaanpak, tenslotte komen de testresultaten ter sprake.

i3D.NET PERFORMANCE HOSTING

DE HAAGSE HOGESCHOOL

Inhoudsopgave

Verklarende woordenlijst

Flavor = Een flavor binnen OpenStack is een blauwdruk van de virtuele resources die gekozen kunnen worden voor een instance.

QoS = Quality of Service. Hiermee wordt de prioriteit van een service uitgedrukt.

Rate limiter = Een vorm van QoS waarbij een service wordt beperkt.

1. Opdrachtformulering

In dit hoofdstuk wordt de opdracht tot testen beschreven.

1.1 Opdrachtgever en opdrachtnemer

Binnen het project zijn de volgende opdrachtgever en opdrachtnemer te onderscheiden:

Opdrachtgever: Rick Sloot, i3D
Opdrachtnemer: Jeroen Holtkamp

1.2 Doelstelling van de test

De tests binnen dit project vinden in twee fases plaats:

Eerst zijn er de tests in de architectuurfase. Deze tests vinden plaats om de performance van de proefopstelling te achterhalen. De performance van de omgeving is binnen dit project het belangrijkste punt. Dit komt doordat er gewerkt wordt met oude systemen waar niet precies duidelijk van is wat OpenStack doet op dat platform.

De tweede testfase in het project komt voor in de “testfase”. In deze fase zullen alle systeemeisen worden getest. In deze fase is het ook zo dat de OpenStack omgeving is uitgebouwd van proefopstelling in een proof of concept. De proefopstelling was om de performance te kunnen testen voor het bouwen. Het proof of concept is het eindproduct van het project. De doelstellingen bij het testen van de systeemeisen is het achterhalen of het systeem voldoet aan de vooraf gestelde eisen.
In de testfase wordt er naast het testen van de systeemeisen ook de OpenStack omgeving getest op performance bottlenecks binnen het systeem. De uitkomsten van de testen voor de bottlenecks kunnen worden meegenomen in de winstberekening. Een voorbeeld van een testresultaat die in de winstberekening meegenomen kan worden zou zijn: “Bij 20 klanten is er een nieuwe controller node nodig om niet te veel performance issues te krijgen”.

1.3 Documentopbouw

De opbouw van dit document is als volgt:

- Omschrijving van het document.
- Omschrijving van de testomgevingen (test op de proefopstelling en tests van de systeemeisen in de testfase).
- Testplannen voor de beide testfases (test op de proefopstelling en tests van de systeemeisen in de testfase).
- Testrapport voor de beide testfases (test op de proefopstelling en tests van de systeemeisen in de testfase).

Zoals uit bovenstaande punten blijkt is dit document dus bedoeld als testplan en als testrapport. Het testplan en testresultaten uit de architectuurfase komen er in. Tevens wordt dit document bijgewerkt met het testplan en de testresultaten in de testfase van het project. Op deze manier zijn alle plannen en resultaten van uitgevoerde tests in het project bij elkaar gebonden.

2. Testomgeving architectuurfase

In dit hoofdstuk staat beschreven hoe de omgeving er uit ziet waar de testen plaatsvinden. Hoe deze omgevingen op zijn gezet staat beschreven in de handleiding. De omgevingen betreffen de proefopstelling en het proof of concept. Met de handleiding (bijlage VI, afstudeerverslag) zijn deze opstellingen reproduceerbaar. In de handleiding staat beschreven hoe de opstellingen zijn opgebouwd. In het hoofdstuk “tests opzetten” van de handleiding staat ook beschreven hoe de tools moeten worden opgezet.

2.1 Compute node los testen

De eerste testsessie die uitgevoerd worden op de proefopstelling zijn de tests die de compute node los testen. Met deze test wordt ook duidelijk wat de eventuele overlast voor de neighbours is op de compute nodes, deze mag namelijk niet hoger dan 200% worden. De compute node moet los worden getest om uit te vinden wat de performance van de losse oude systemen is.

In deze sessie vinden veel verschillende tests plaats op de compute node, nl:

- CPU-test
- HDD-test
- web-server test

Ook vinden in deze fase tests plaats op de gehele proefopstelling:

- Load tests met OpenStack rally om de performance systeemeisen te testen. De connecties mogen niet te snel vollopen.

2.1.1 CPU-test

Omgeving: Een compute node met 7 actieve Ubuntu 14.04 cloud instances.

Omschrijving: De eerste test is de CPU-test. Met deze test moet worden getest of instances hinder ondervinden van elkaar bij normale CPU-load. Tevens wordt hier ook met lage en hoge load getest om te zien hoe de performance wordt verminderd bij hoge load van de neighbours.

Eis die wordt getest: “In het proof of concept mag de slowdown op de VM's niet hoger worden dan 200% bij normaal gebruik (uitgezonderd piekverkeer)”.

Tools:
- Stress-ng
- Sysbench
- md5sum

Voorbereiding:

Alle instances op de compute node moeten benaderbaar zijn via SSH. Ook moet ook op alle instances S*tress-ng* worden geïnstalleerd. Op één van de zeven instances moet naast S*tress-ng* ook *Sysbench* worden geïnstalleerd. Ook is MD5sum nodg, deze is echter al aanwezig op een Ubuntu installatie.
Om de performance te kunnen meten met *MD5sum* moet er een bestand aanwezig zijn met random data. Dit

bestand wordt gebruikt om een hash uit te rekenen. Het aanmaken van dit bestand op de instance waar ook *Sysbench* aanwezig is kan met het volgende commando:

head -c 100M </dev/urandom >/home/ubuntu/rand.random

2.1.2 HDD-test

Omgeving: Een compute node met 7 actieve Ubuntu 14.04 cloud instances.

Omschrijving: De tweede test is de HDD-test. Met deze test moet worden getest of instances hinder ondervinden van elkaar bij normale HDD-load. Tevens wordt hier ook met lage en hoge load getest om te zien hoe de performance wordt verminderd bij hoge load van de neighbours. Eerst wordt er getest op de lokale schijf in de compute node. Ook wordt de performance slowdown van de schijven in de block storage node op dezelfde manier getest.

Eis die wordt getest: “In het proof of concept mag de slowdown op de VM's niet hoger worden dan 200% bij normaal gebruik (uitgezonderd piekverkeer)”.

Tools: -

Voorbereiding:

Alle instances op de compute node moeten benaderbaar zijn via SSH. Op de meetinstance moet er een bestand worden aangemaakt waar de random data naartoe gestuurd kan worden. Deze wordt random.rnd genoemd.
Om de Cinder storage te testen moeten de instances allemaal uitgerust worden met een block volume van 1GB.

Voor de test die de Cinder storage test zijn nog een aantal extra acties vereist. Zo moeten in Cinder de volumes worden aangemaakt. Vervolgens moeten de volumes worden gekoppeld met de betreffende instances. Op de instances moet tenslotte worden gezorgd voor een partitionering van het volume.

2.1.3 Webserver test

Omgeving: Een compute node met 7 actieve Ubuntu 14.04 cloud instances.

Omschrijving: De derde test is de webserver test. Met deze test moet worden getest of instances hinder ondervinden van elkaar bij realistische webserver load. Tevens wordt hier ook met lage en hoge load getest om te zien hoe de performance wordt verminderd bij hoge load van de neighbours.

Eis die wordt getest: “In het proof of concept mag de slowdown op de VM's niet hoger worden dan 200% bij normaal gebruik (uitgezonderd piekverkeer)”.

Tools:

- MySQL
- Apache2
- Apache2 AB
- Sysbench

Voorbereiding:

Alle instances op de compute node moeten benaderbaar zijn via SSH. Op alle instances moet een *MySQL server* worden geïnstalleerd.
Binnen de *MySQL-server* moet een database genaamd “test” worden aangemaakt. In de database “test” moet een tabel genaamd “test” worden aangemaakt. Deze tabel bestaat uit de rijen A, B en C. Alle drie de rijen zijn van het type VARCHAR(100).
Ook moet *Apache2* worden geïnstalleerd op de instances. Naast de *Apache2* installaties moet ook nog het packet apache2-utils worden geïnstalleerd om gebruikt te kunnen maken van *Apache2 AB*.
Om de load te meten moet op de meetinstance *Sysbench* worden geïnstalleerd.

2.2 Dataverbindingen tussen de nodes testen

In deze test zal er worden getest hoe de dataverbindingen zich verhouden in een actieve OpenStack omgeving. Het is in deze test van belang dat het gericht is op de datalijnen exclusief de block storage lijnen. De block storage lijnen worden in de volgende test (een andere eis) individueel getest.

Omgeving: De volledige OpenStack proefopstelling.

Omschrijving: Bij deze test wordt er met OpenStack Rally een realistische load gegenereerd op de proefopstelling. Door te meten bij de netwerkinterface op het managementnetwerk van de controller node kan gezien worden hoe vol de verbindingen worden getrokken.
Het is wel van belang te melden bij deze test dat Glance verkeer wordt gezien als piekverkeer. Hier is voor gekozen omdat Glance volgens de documentatie de dataverbinding helemaal vol zal trekken en hier geen QoS voor kan worden ingesteld.

Eis die wordt getest: “Het netwerkverkeer over heel het netwerk mag onder normale omstandigheden (uitgezonderd piekverkeer) niet over 50% van de lijn komen.”.

Tools:

- OpenStack Rally op een individuele node.
- ifstat

Voorbereiding:

Rally moet worden geïnstalleerd op een individuele node. Dit wordt uitgelegd in de handleiding in het hoofdstuk “tests opzetten”. Na het opzetten van Rally moet ook de admin “openRC” file worden gekopieerd van de controller node naar de node waar Rally op draait. In dit geval gaat het om het bestand “*admin_openrc.sh*”. Tenslotte moet dit openRC bestand worden gesourced worden om ook op de utility node (node waar Rally op staat) admin privileges te verkrijgen.

2.3 Dataverbinding naar de (block) storage testen

In deze test zal er worden getest hoe de dataverbindingen zich verhouden in een actieve OpenStack omgeving. Het is in deze test van belang dat het gericht is op de datalijnen exclusief de block storage lijnen. De block storage lijnen worden in de volgende test (een andere eis) individueel getest.

Omgeving: De verbinding tussen compute en Cinder block storage.

Omschrijving: Bij deze test wordt er met *head -c 100M </dev/urandom >/home/ubuntu/rand.random* hoge load

gegenereerd naar de block storage node. Door te meten bij de netwerkinterface op de storage node kan gezien worden hoe vol de storage verbinding wordt getrokken.

Eis die wordt getest: “Het netwerkverkeer van en naar de storage mag niet zo hoog worden dat de verbinding wordt dichtgetrokken. Tijdens normaal (uitgezonderd piekverkeer) gebruik mag het niet hoger worden dan 50% van de lijn.”.

Tools:

- ifstat

Voorbereiding:
Om deze test voor te bereiden moeten de instances die gebruikt gaan worden voor deze test worden voorzien van een Cinder block device. Deze devices hoeven niet zo groot te zijn, ze mogen 1~2 GB zijn.

3. Testomgeving testfase

Dit subhoofdstuk wordt gewijd aan het beschrijven van de testomgevingen die gebruikt moeten worden in de testfase. Het is onderdeel van het testplan en is daarom geschreven in de ontwikkelingsfase. De acties die uitgevoerd zijn om het proof of concept op te bouwen zijn reproduceerbaar door de handleiding (bijlage VI, afstudeerverslag).

3.1 Redundante storage

Deze test zal er op gericht zijn om te controleren of de Cinder block storage redundant is uitgevoerd en zich ook zo gedraagt.

Omgeving: De Cinder block storage node.

Omschrijving: Bij deze test zal er moeten worden gekeken of de storage goed blijft werken bij uitval van één disk.

Eis die wordt getest: “In het uiteindelijke proof of concept mag de storage bij schijfuitval (van 1 schijf in het proof of concept) geen dataverlies lijden.”

Tools:

- -

Voorbereiding:

Er moet een lege 250GB 7200RPM harde schijf voorhanden zijn.

3.2 Schaalbare resources

In deze test moet worden aangetoond dat de resources van instances kunnen schalen. Dit moet kunnen gebeuren op het commando van een gebruiker.

Omgeving: Het volledige proof of concept.

Omschrijving: In een cloud is het van belang dat klanten hun VM's kunnen up- en downscalen. Bij deze test zal het schalen gebeuren door eerst een snapshot te maken van de gekoppelde storage. Vervolgens moet de gekoppelde storage afgekoppeld worden. Hierna er een nieuwe instance gemaakt worden op basis van een nieuwe flavor (up- of downscalen). Tenslotte wordt de Cinder storage weer aangekoppeld.

Eis die wordt getest: “De "klanten" moeten in het proof of concept de mogelijkheid hebben om hun resources te kunnen schalen.”

Tools:

- -

Voorbereiding:

In het proof of concept moet een instance actief zijn met minimaal één gekoppeld Cinder volume. Deze instance moet gecreëerd zijn op basis van Ubuntu 14.04 cloud.

3.3 Storage bestellen

Het schalen van storage of het “bestellen” van extra storage moet worden aangetoond met deze test. Het bestellen van de instances moet kunnen gebeuren op commando van een gebruiker.

Omgeving: Het volledige proof of concept.

Omschrijving: Een eis van de opdrachtgever was dat de gebruikers in de OpenStack cloud extra storage konden bestellen. In deze test zal er een extra Cinder volume moeten worden aangemaakt. Vervolgens moet dit volume worden gekoppeld aan de betreffende instance.

Eis die wordt getest: “De "klanten" moeten in het proof of concept de mogelijkheid hebben om extra storage te kunnen "bestellen" (schalen/resource provisioning).”

Tools:

- dd
- lsblk

Voorbereiding:

In het proof of concept moet een instance actief zijn zonder Cinder volumes gekoppeld. Deze instance moet gecreëerd zijn op basis van Ubuntu 14.04 cloud.

3.4 Hosten van een virtueel OS

Een systeemeis aan het proof of concept is dat een instance van in ieder geval één distributie kan worden gehost in OpenStack. Bij het testen is de CirrOS image al gehost. De CirrOS image is zo ver uitgekleed dat er verder niet veel mee gedaan kan worden, hierdoor is de image wel heel klein. Bij deze test moet er echter een “normaal” OS worden gehost om aan te tonen dat dit mogelijk is.

Omgeving: Het volledige proof of concept.

Omschrijving: Bij deze test moet er een instance kunnen worden gehost in de OpenStack omgeving. De instance mag van een zelf gekozen distributie zijn.

Eis die wordt getest: “Er moet op het proof of concept één soort operating system (één distributie) kunnen worden gehost.”

Tools:

- Putty
- Puttygen

Voorbereiding:

Om de test voor te bereiden moet eerst de gebruikte image (Ubuntu 14.04 cloud) worden gedownload. Dit gebeurd met het commando:

#wget http://cloud-images.ubuntu.com/releases/14.04.3/release/ubuntu-14.04-server-cloudimg-amd64-disk1.img -P /home/admin1986

Vervolgens moet de gedownloade image opgenomen worden in de Glance service. Er kan handmatig een image worden toegevoegd aan de Glance service met het commando:

#glance image-create --name "Ubuntu_1404_cloud" --file /etc/ubuntu-14.04-server-cloudimg-amd64-disk1.img --disk-format qcow2 --container-format bare --visibility public

Ook moet er een virtueel netwerk aangemaakt zijn in Neutron. In dit virtuele netwerk moet er een subnet zijn aangemaakt waar de gebruikte user toegang toe heeft.
Ook moet de default security group zo staan ingesteld dat verkeer op port 22 naar de instance kan komen. Er moet een keypair zijn aangemaakt met de naam “instance-key”. Er moet tenslotte ook een floating-IP zijn aangemaakt met Neutron die gebruikt kan worden in de test.

3.5 Floating-IP's

In deze test moet aangetoond en getest worden dat in ieder geval één instance naar het externe netwerk (internet) kan verbinden. In het huidige proof of concept kan dit door gebruik te maken van floating IP's.

Omgeving: Network node, compute node, controller node.

Omschrijving: Het gebruik en het correct functioneren van de geïmplementeerde functie om instances met het externe netwerk te verbinden: floating-IP's moet worden getest in deze test.

Eis(en) die wordt getest:

- Het proof of concept moet een alternatief voor SNAT en DNAT hebben om "klanten" publieke IP-adressen op te laten vragen.
- Mensen moeten externe IP-adressen kunnen bestellen om hun instances te kunnen verbinden met internet en externe netwerken.

Tools:

- Aptitude
- Ping
- wGET

Voorbereiding:

Om deze test voor te bereiden moet er een kale instance van Ubuntu 14.04 cloud actief draaien binnen de testomgeving.

3.6 Klantdata back-up

Binnen het proof of concept is het van belang dat er wordt aangetoond dat er de mogelijkheid is om back-ups te maken van klantdata.

Omgeving: Het volledige proof of concept

Omschrijving: Met deze test wordt de back-up oplossing voor klantdata getest. Met deze oplossing worden alle volumes van de klanten van de Cinder node naar de Swift storage gekopieerd. Dit wordt gedaan door de Cinder back-up driver. In de test zal er een back-up van de live instance gemaakt moeten worden met de Nova snapshot functie. Na het maken van een back-up moet de back-up ook weer worden gerecovered. Aan het eind van de test moet het volume weer volledig functioneel zijn en zonder fouten werken.

Eisen die worden getest:

- “Er moet een back-up van de data van de gebruikers gemaakt kunnen worden.”
- “Binnen het proof of concept moeten snapshots van de VM's gemaakt kunnen worden.”

Tools:

- dd
- md5sum

Voorbereiding:

In de testomgeving moet er al een Ubuntu 14.04 cloud instance actief zijn. Aan deze instance moet een floating-IP zijn gekoppeld. Ook moet er een Cinder volume aan de instance gekoppeld zijn. Het bootvolume van de gebruikte instance moet ook in Cinder worden gehost.
Het volume die gekoppeld zit aan de instance moet een grootte hebben van 1GB. Met Fdisk moet het volume worden gepartitioneerd en gefotmatteerd naar het EXT3 filesystem. Met dd moet er een random file van 1020MB worden aangemaakt op deze instance. Door gebruik te maken van md5sum moet van dit random bestand een MD5-checksum worden gecreëerd. Deze checksum moet apart (ander systeem) worden opgeslagen. Met deze opstelling is er op de gekoppelde schijf een vorm van integriteitscontrole uit te voeren.
Helaas is het niet mogelijk om een integriteitscontrole uit te voeren op het boot volume omdat het een live back-up betreft. In een live back-up veranderd de omgeving veel te snel om een controle met een checksum uit te voeren.

3.7 OpenStack systeemback-up

Ook van de OpenStack-componenten moet een back-up gemaakt kunnen worden. Bij calamiteiten of bij falen van een component moeten deze gerecovered kunnen worden. Met deze test wordt ook gecontroleerd of de gekozen oplossing voor de systeemback-up eis volstaat.

Omgeving: Het volledige proof of concept

Omschrijving: In deze test wordt gecontroleerd of de oplossing voor de systeemback-up voldoet. Het is de bedoeling dat er een back-up wordt gemaakt van alle componenten waar het proof of concept uit bestaat. Vervolgens moeten de onderdelen ook weer worden gerestored. Hierna moeten alle onderdelen nog steeds functioneel zijn en goed werken. De OpenStack componenten worden wel één voor één behandeld.

Eis die wordt getest: “Er moet een back-up gemaakt kunnen worden van de services binnen OpenStack.”

Tools: N.V.T.

Voorbereiding:

Voor deze test is weinig voorbereidend werk nodig. De omgeving moet wel in volledig functionerende staat zijn. Ook moet er een container genaamd “backup” actief zijn in Swift.

3.8 Control panel

Omgeving: Het volledige proof of concept

Omschrijving: In OpenStack moet de mogelijkheid bestaan om gebruik te maken van een grafisch control panel. Met deze test wordt de functionaliteit van het OpenStack-dashboard getest. Met een test user wordt getest of het dashboard naar behoren functioneerd.

Eis die wordt getest: “Klanten moeten kunnen inloggen op het OpenStack control panel.”

Tools: N.V.T.

Voorbereiding:

Voor aanvang van de test moet er een test user worden aangemaakt. Deze user moet de user role toegewezen krijgen en in het default domain worden geplaatst.

3.9 Update

Omgeving: Het volledige proof of concept

Omschrijving: Om de nodes en de OpenStack installatie op-to-date te houden moeten deze onderdelen geüpdatet kunnen worden. Binnen het proof of concept is een repository server aanwezig. Met deze test moet dus worden gekeken of de nodes ook daadwerkelijk kunnen updaten via de repository node.

Eis die wordt getest: “De Linux installaties en de OpenStack componenten op de fysieke nodes moeten geüpdatet kunnen worden.”

Tools: N.V.T.

Voorbereiding:

De Iptables moeten goed ingesteld staan. Ook moeten de FTP-verbindingen van de nodes naar de repository node goed staan ingesteld.

3.10 Verbruik meten

Omgeving: Het volledige proof of concept

Omschrijving: De opdrachtgever vond het ook belangrijk dat het verbruik van users in de cloud bijgehouden zou worden. Met OpenStack Ceilometer is het verbruik bij te houden. In deze test moet worden aangetoont dat Ceilometer werkt en de acties van gebruikers logt.

Eis die wordt getest: “Het verbruik van de klanten in OpenStack moet gemeten kunnen worden.”

Tools: N.V.T.

Voorbereiding:

Een instance opzetten op basis van de m1.small_small flavor en Ubuntu 14.04. Er moet tevens een floating-IP gekoppeld worden aan deze instance.

3.11 Performance groei

In deze tests moet er worden aangetoont op welk punt in de groei van de cloud aansturende nodes bijgeplaatst moeten worden. Dit zijn dan ook geen tests die positief of negatief als uitkomst hebben, maar er is een getal als uitkomst. Het getal die de tests als uitkomst hebben is het aantal instances die de huidige aansturende nodes aankunnen voordat er een nieuwe aanstuderende node bijgeplaatss moet worden. Onder aansturende nodes verstaan we: controller node, network node en Block storage node.
De uitkomst van de tests, de, groei, is belangrijk in de winstberekening. Er moet worden berekend bij welk aantal instances er een nieuwe aansturende node geplaatst moet worden, zo kan de groei van de aansturende nodes worden berekend, dit zijn variabelen waar voor getest moet worden.

Omgeving: Het volledige proof of concept

Omschrijving: Voor de winstberekening zijn de variabelen van de groei in de aansturende nodes belangrijk. Met deze tests wordt duidelijk bij welk aantal instances er een nieuwe controller node, network node, block storage node en object proxy node nodig is qua performance.

Eis die wordt getest: NVT, de uitkomsten zijn variabelen in de winstberekening.

Tools: N.V.T.

Voorbereiding:

Als voorbereiding voor deze tests moeten er eerst 21 key-pairs(3 * 7 = maximale belasting compute nodes met m1.small_small flavor) en bijbehorende Putty-keys worden aangemaakt. Ook moeten er alvast 21 bootable Cinder-volumes worden aangemaakt met het *#openstack volume create* commando. Tevens moeten er 21 floating-IP's worden aangemaakt.

3.12 Management network security

In deze test moet worden aangetoont dat ongewenste verbinding van buiten de interne netwerken niet in het interne netwerk kunnen komen.

Omgeving: Het volledige proof of concept.

Omschrijving: Deze systeemeis is ontstaan uit de initiële systeemeis: “Alle normale beveiligingsmaatregelen

moeten zijn geconfigureerd”. De huidige systeemeis is er voor in de plaats gezet aangezien de voorgaande eis niet te testen en zeer onduidelijk zou zijn.
Bij deze test moet worden gekeken naar de beveiligingsmaatregelen en of ze goed zijn geïmplementeerd. De test wordt gericht op de servers in het DMZ, alle andere servers staan door middel VLAN's gescheiden van het i3D netwerk dus hoeven niet te worden getest.

Eis die wordt getest: In het proof of concept mogen onbevoegde verbindingen vanaf het externe netwerk niet in een van de interne netwerken komen.

Tools:

- Nmap

Voorbereiding:
Als voorbereiding is er niets uit te voeren. Het huidige proof of concept moet worden getest.

4. Testaanpak architectuurfase

In dit subhoofdstuk wordt besproken hoe de tests in de architectuurfase worden aangepakt. Samen met de aanpak worden ook de voorwaarden waarbij de tests geslaagd zijn beproken, en de acties die moeten wordne uitgevoerd wanneer de test niet positief wordt getest worden ook besproken.

4.1 Compute node los testen

Om de onderstaande eis te testen zijn er verschillende tests nodig. Zo moet de CPU en de storage worden getest. Ook moet er een test worden uitgevoerd met realistische belasting om te zien of de instances onder elkaar meer dan 200% performance slowdown veroorzaken. In alle tests mag de slowdown niet boven de 200% komen bij normale belasting. Per test zal ook worden aangegeven wat voor die test “normale” belasting is.

4.1.1 CPU-test

Eerst vind er een test plaats die de CPU-noise onder de instances in kaart brengt en hiermee de eis test.
De bedoeling is om een belastingstest uit te voeren waarbij de belasting steeds wordt opgevoerd. De belasting wordt zelfs boven de grens van “normale” belasting getild omdat hiermee te zien is wanneer eventuele noise wel boven de grens van de eis uitkomt. Onder normale belasting van de CPU verstaan we 60%. De load van 60% is gekozen na een vraag van medewerkers van de netwerkafdeling van i3D. Dit betekend dat de CPU-slowdown op het meetpunt niet meer dan 200% mag worden terwijl alle andere instances op de compute node tot 60% CPU-load hebben.

Een opmerking bij dit testen is wek dat de instances die niet meedoen aan de test uitstaan. Instances die aan zouden staan terwijl ze niet meedoen aan de test veroorzaken dan ongewenste load.

Er wordt begonnen met één instance die aan wordt gezet. Op deze instance staat *Stress-ng* én *Sysbench*. Met *Stress-ng* kan een CPU-load van N% worden gegenereerd. *Sysbench* is de applicatie die de meting doet. Door de berekening van priemgetallen kan worden gekeken hoe veel de CPU gebruikt kan worden door die instance.

Eerst wordt er een ijkpunt gemaakt. Dit wordt gedaan door *Sysbench* te laten draaien zonder dat *Stress-ng* actief is. Op deze manier kan worden gekeken hoe lang *Sysbench* nodig heeft voor de berekeningen zonder dat er load actief is.
Vervolgens Moet *Stress-ng* op een CPU-load van 10% worden gezet. Tijdens het beladen van de CPU door *Stress-ng* moet nu *Sysbench* gedraaid worden. Hiervan kan worden afgeleid hoe lang *Sysbench* nodig heeft om de priemgetallen uit te rekenen terwijl er een load actief is. Dit gaat door tot 100%. De load zal worden gegenereerd in de volgende stappen: 0%(nulpunt/ijkpunt), 10%, 20%, 40%, 60%, 80% en 100%.

De commando's voor *Stress-ng* en *Sysbench* zijn als volgt (op de plek van N moet de load worden ingevuld):

#stress-ng --cpu 1 -t 60 -l N
#sysbench --num-threads=1 --test=cpu --cpu-max-prime=10000 run

Hierna moet er een instance bij worden aangezet. Nu moet er een nieuw ijkpunt worden aangemaakt. Dit ijkpunt is om te zien hoe lang *Sysbench* zijn berekeningen uitvoert met twee instances op idle. Vervolgens worden dezelfde stappen in load van *Stress-ng* doorlopen maar nu op twee instances. Elke keer wordt er met *Sysbench* gemeten op 1 instance.
Op deze manier moeten alle instances worden aangezet er wordt de load elke keer opgevoerd. De instance aantallen die moeten worden getest zijn: 1, 2, 4, 6, 7.

De volgende tabel zal er dan uitkomen:

	0% (ijkpunt)	10%	20%	40%	60%	80%	100%
1 instrance							
2 instances							
4 instances							
6 instances							
7 instances							

Tabel1: De tabel die gevuld wordt met testresutaten in het testrapport.

De Sysbench test wordt per test-onderdeel drie keer gedraaid. Van deze drie keer wordt een gemiddelde genomen. Deze keus is gemaakt om fluctuaties in de testresultaten op te vangen.

Om De testresultaten te ondersteunen wordt dezelfde test nog een keer doorlopen. Deze keer wordt in plaats van Sysbench MD5sum gebruikt als meetapplicatie. Met MD5sum wordt er een MD5-hash gemaakt van het test bestand. Om een meetbaar resultaat te krijgen wordt dit commando uitgevoerd in combinatie met het time commando van Linux.
Dit time commando geeft de tijd terug hoe lang het commando nodig had om uit te voeren. Als eerste komt hier de clock time uit. Dit is de tijd zoals op een klok. Als tweede en derde geeft het time commando user- en systemtime terug. Dit is de tijd die de uitvoer heeft besteed in de kernel en buiten de kernel. Als deze twee bij elkaar worden geteld dan is de uitkomst de tijd die het commando binnen en buiten de kernel nodig heeft gehad op uit te voeren.
Deze test is qua opzet het zelfde als de vorige test alleen is de meetapplicatie anders. Het commando die wordt gebruikt bij het meten is:

#Time md5sum randomfile.rnd

4.1.2 HDD-test

Omdat er in de compute maar één harde schijf zit is het handig om te zien of dit de instances kan vertragen. De uitkomst rekend echter niet heel zwaar mee aangezien in een vervolgtraject in de opbouw van OpenStack zeer waarschijnlijk gebruik gemaakt gaat worden van een groot storage backend. Normale load is bij deze test moeilijk te definiëren. Echter zal een normale load onder de 100KB data per 0,01 sec blijven.

Deze test is in opbouw hetzelfde als de CPU-test. Er zijn weer zeven instances waar een load op wordt uitgevoerd en één van die instances is ook aangewezen als meetpunt.

De load wordt gegenereerd met een script. Het script om de lokale HDD te testen heeft de volgende inhoud:

```
#!/bin/bash

while true
do
        sudo head -c 100K </dev/urandom >/home/ubuntu/rand.random
        sleep X
done
```

Het script om de block storage te testen heeft de volgende inhoud:

```
#!/bin/bash

while true
do
        sudo head -c 100K </dev/urandom >/media/disk2/testfile
        sleep 0.01
done
```

Met het script wordt er om de X seconden een reeks van 100KB random data naar een bestand geschreven. Door deze X-variabele aan te passen kan een drukker systeem gesimuleerd worden. De X zal de volgende waardes hebben: 1 sec, 0,5 sec, 0,25 sec, 0,12 sec, 0,06 sec, 0,03 sec en 0,01 sec.

Op het meetpunt wordt bij de lokale opslag test gemeten met het volgende commando:

#time head -c 50M </dev/urandom >/home/ubuntu/random_file

Op het meetpunt wordt bij de Cinder opslag test gemeten met het volgende commando:

#time head -c 50M </dev/urandom >/media/disk2/testfile

Met dit commando wordt er een reeks van 50MB data naar een bestand weggeschreven. Met het “time” commando wordt de tijd gemeten hoe lang het commando nodig heeft om uit te voeren.
Op deze manier kan gemeten worden hoe lang het schrijven op de HDD duurt in een drukke omgeving.

4.1.3 Webserver test

Met deze test wordt er een realistische load gegenereerd op de nodes. Er wordt ingeschat door de opdrachtgever dat de meeste instances gebruikt gaan worden als webserver of soortgelijke oplossingen. Om een load te genereren die realistisch is moet er een webserver gesimuleerd worden.
Omdat bij een webserver vrijwel altijd een database is geïnstalleerd zal dit ook gebeuren in de testopstelling. Normale load op een kleine webserver valt ergens in de range van 10 ~ 100 HTTP-requests per seconde.

Met Apache2 AB kunnen webservers gebenchmarkt worden. In een script kan dit commando zo worden opgezet zodat het een X aantal requests naar een Apache webserver wegstuurt. Om dit aantal requests te reguleren (per sec) wordt ook een sleep commando gegeven.

```
#!/bin/bash

while true
do
        ab -n 56 -c 1 http://109.200.204.43/
        sleep 0.1
done
```

Om ook een SQL-database te simuleren bij deze HTTP-requests wordt er per 5 HTTP-requests een SQL-INSERT uitgevoerd. De SQL-simulatie wordt uitgevoerd met het volgende script:

```
#!/bin/bash

while true
do
        echo "insert into test (a,b,c) values ('data','data','data');" | mysql -u
       root -ptoor test
        sleep 0.009
        echo "delete from test where a = 'data';" | mysql -u root -ptoor test
        sleep 0.009
done
```

Met het script wordt om de X seconden een SQL-INSERT of een SQL-DELETE commando gegeven. Dit aantal seconden is 5 keer lager dan de webserver.
Met Sysbench zal er een meetpunt worden opgezet.

De uitkomst van alle drie de tests (CPU, HDD en webserver) moeten aangeven dat de performance slowdown voor andere instances op die compute node niet hoger mogen worden dan 200% bij normaal gebruik. Als de tests hier onder blijven dan zijn de tests geslaagd en is de eis positief getest.
Als er negatief wordt getest dan zal de overcommit op de compute node lager gezet moeten worden.

4.2 Dataverbindingen tussen de nodes testen

In deze tests is het de bedoeling om de verbinding tussen de nodes te testen aan de hand van de volgende eis:

“Het netwerkverkeer over heel het netwerk mag onder normale omstandigheden (uitgezonderd piekverkeer) niet over 50% van de lijn komen.”

In deze test is het dus de bedoeling dat het systeem wordt getest op de systeemeis. Om de eis te kunnen testen moet er eerst een realistische load worden gegenereerd. Het is echter wel zo dat het maximum moet worden getest aan de load die users kunnen veroorzaken op de dataverbindingen. Alle users bij elkaar mogen de load op de dataverbindingen niet boven de 50% heen trekken.

Met OpenStack Rally kunnen verschillende scenario's worden gesimuleerd. Een scenario zou kunnen zijn: Het aanmaken van tien instances en hierna deze instances weer verwijderen. Bij deze test worden alle scenario's die users kunnen veroorzaken binnen hun activiteit in de cloud tegelijk uitgevoerd.

Deze activiteiten zijn:

- Create and delete users: 100 requests totaal
- Create and delete roles: 100 requests totaal
- Create and list keypairs: 10 keer door 6 users = 60 requests totaal
- Create and list networks: 100 keer door 9 users = 900 requests totaal
- List flavors: 10 keer door 6 users = 60 requests totaal
- Create and delete images: 10 keer door 6 users = 60 requests totaal

In dit lijstje komt het scenario “Boot and delete”. Dit komt omdat het management volledig wordt volgetrokken door Glance. Er is geen QoS of rate limiter voor Glance als de images van de controller worden gehaald.

Het nadeel van Rally is dat de scenario's eigenlijk niet tegelijk uitgevoerd kunnen worden. Echter kunnen de

scenario's wel tegelijk worden uitgevoerd als elk scenario apart wordt gestart.

In de test is ~1Gb/s het maximum van de lijn. Om de test te laten falen zal de load boven de 500Mb/s moeten komen. Als de load onder de 500Mb/s blijft is de test geslaagd. De load wordt gemonitord op de controller aansluiting op het management netwerk. Er wordt op deze aansluiting gemonitord aangezien de controller de aansturende eenheid is in de OpenStack omgeving.
Op de controller moet het volgende commando worden uitgevoerd:

#ifstat -n -b -i eth0

Met de utility node moeten meerdere sessies worden opgezet. Op deze node moeten de volgende commando's worden uitgevoerd:

#rally --verbose task start /usr/share/rally/samples/tasks/scenarios/keystone/create-and-list-users.json
#rally --verbose task start /usr/share/rally/samples/tasks/scenarios/keystone/create-and-delete-roles.json
#rally --verbose task start /usr/share/rally/samples/tasks/scenarios/nova/create-and-list-keypairs.json
#rally --verbose task start /usr/share/rally/samples/tasks/scenarios/nova/list-flavors.json

Als de usage van de netwerkverbindingen tussen de nodes niet boven de 50% uitkomt is de test geslaagd.
Als er negatief wordt getest zal er moeten worden gekeken naar oplossingen om de verbinding af te knijpen voor verschillende instances.

4.3 Dataverbinding naar de (block) storage testen

In deze test moet er worden gekeken of het netwerkverbruik op de connectie tussen compute- en block storage node boven de 50% van de totale bandbreedte uitkomt.
Dit kan worden getest door instances uit te rusten met een cinder block volume en hier load op te genereren.

Verschillende instances krijgen block volumes gekoppeld. Op deze instances wordt er vervolgens met *head -c 500M </dev/urandom >[pad naar block device]* load gegenereerd naar de Cinder service.
Op deze manier kan worden getest of het verbruik op deze verbinding boven de 50% van de totale bandbreedte uitkomt.

Als de usage van de storage verbinding niet boven de 50% van de totale bandbreedte uitkomt is de test geslaagd.
Als er negatief wordt getest zal moeten worden gekeken naar oplossingen om de verbinding van de instances te kunnen afknijpen, eventueel met QoS.

5. Testaanpak testfase

In dit subhoofdstuk wordt besproken hoe de tests in de testfase worden aangepakt. Samen met de aanpak worden ook de voorwaarden waarbij de tests geslaagd zijn beproken, en de acties die moeten wordne uitgevoerd wanneer de test niet positief wordt getest worden ook besproken.

5.1 Redundante storage

In deze test moet worden aangetoond dat de Cinder block storage functioneel blijft bij het uitvallen van één disk. Dit is theoretisch het geval aangezien de RAID-controller in de server is ingesteld op RAID 10. Bij het uitvallen van een disk zou de RAID-array gewoon door moeten blijven draaien.
Aangezien de server en de RAID-controller hot-pluggable zijn kan een disk fault worden gesimuleerd door het verwijderen van een schijf. Bij het verwijderen van een schijf moet de controller eerst zeggen dat hij de betreffende schijf niet meer detecteert. Vervolgens moet de controller de nieuwe schijf weer opnieuw gaan opbouwen als deze wordt aangesloten.

Als de verwijderde schijf aan het eind van de test geen fouten geeft in het overzicht van de controller en er naar het array toe kan worden geschreven van een instance af is de test positief getest.
Als er negatief wordt getest dan moet er eerst opnieuw worden getest met een andere RAID-controller en andere schijven.

5.2 schaalbare resources

De mogelijkheid om resources te kunnen schalen is het onderwerp van deze test. In OpenStack gebeurd het up- en downscalen met het wisselen van flavors.

Als eerste moet er een snapshot worden gemaakt van de betreffende instance. Het creëren van een snapshot moet gebeuren met het commando *#nova image create [instanceID] [instancenaam]*.
Hierna moeten alle Cinder volume die gekoppeld staan met de betreffende instance worden afgekoppeld. Dit afkoppelen gebeurd met het commando *#nova volume-detach [instanceID] [volumeID]*. Op dit moment draait alleen de kale instance nog op de compute node.
Nu kan er een nieuwe instance worden gestart op basis van een andere flavor en de gecreëerde snapshot (image). Dit opbouwen moet met het volgende commando:

#nova boot --flavor [nieuwe flavor] --image [image naam uit de nova image-list] --nic net-id=[ID van het netwerk] --security-group default --key-name [Naam van het keypair] [naam die de instance moet krijgen]

Hierna moeten de Cinder volumes weer kunnen worden aangekoppeld.
Uiteindelijk moet de instance zonder problemen kunnen worden opgestart en aangeven dat de virtuele resources inderdaad geschaald zijn. Het controleren van de virtuele resources moet gebeuren in de instance zelf en op de controller node. In de instance kan er worden gecontroleerd op virtuele resources door het uitvoeren en analyseren van het commando *#top* (RAM) en *#df -Th* (HDD). Op de controller node kan worden gecontroleerd of het schalen is gelukt met de volgende commando's: *#nova show [instanceID]* en *#nova flavor-list*.

De test is positief als er bij het schalen zich geen fouten voordoen. Tevens moeten de services en de instance na het schalen aangeven dat de resources inderdaad zo groot zijn als de flavor.
Als er negatief wordt getest dan moet de fout worden achterhaald. Niet goed schalen volgens deze methode geeft namelijk een duidelijke fout in het systeem aan.

5.3 Storage bestellen

Deze test staat in het teken van het bestellen van extra storage. Het toewijzen van extra storage binnen OpenStack gebeurd met ondersteuning van Cinder.
De test wordt uitgevoerd door eerst een kale Ubuntu 14.04 cloud instance actief te zetten. Vervolgens moet er met Cinder een volume worden aangemaakt, dit gebeurd met:

#cinder create –-display-name [naam van het volume] [grootte van het volume in GB]

Hierna moet het volume worden gekoppeld aan de betreffende instance. Dit moet met het commando:

#nova volume-attach [instanceID] [volumeID]

Na het koppelen moet er worden gecontroleerd of het volume echt gekoppeld is in de instance. Dit moet staan in de */dev/* directory als VD*. Ook moet er met Fdisk een partitie en een filesystem aangemaakt kunnen worden op de schijf. Na het partitioneren moet er met *dd* random data naar een random bestand op dit nieuwe volume kunnen worden geschreven.

De test is positief als er bij het schalen zich geen fouten voordoen. Tevens moet het partitioneren van en schrijven naar het volume zonder fouten verlopen.
Als de test negatief is zal de fout moeten worden achterhaald en opgelost. Deze functie moet goed werken binnen OpenStack. Als het niet goed werkt zit er zeer waarschijnlijk een fout binnen Cinder.

5.4 Hosten van een virtueel OS

Deze test is op zich vrij kort als alles zonder fout werkt. In de voorbereiding op de test is de image al klaargezet in de Glance-store, er is een Neutron virtueel netwerk en subnet aangemaakt, er is een keypair aangemaakt die gebruikt kan worden met de image en de default security group is zo ingesteld dat port 22 (SSH) doorgelaten wordt naar de instance. Ook is er een floating-IP aangemaakt.

De test wordt gestart met het maken van een instance uit de Ubuntu 14.04 cloud image in de Glance-store. Dit wordt gedaan met het *#nova boot* commando. Vervolgens moet er worden gecontroleerd of de actie succesvol is uitgevoerd met het *#nova list* commando. Dit commando moet de gecreëerde image weergeven.

Om de test te vervolgen moet er hierna een floating-IP worden gekoppeld aan de gemaakte instance. Deze floating-IP moet worden gekoppeld met het *#nova floating-ip-associate* commando.

Om te controleren of de instance op dit moment ook echt actief is moet er met Putty worden ingelogd via SSH naar de betreffende instance. Er moet connectie worden gemaakt met het floating-IP aangezien deze gekoppeld zit met de instance.

Als er connectie gemaakt kan worden met de instance zonder dat er fouten opkomen is de test geslaagd: er kan op dat moment succesvol een instance van een zelfgekozen distributie worden gehost in het proof of concept.
Als de test negatief is zal de fout moeten worden achterhaald en opgelost. Hierna moet de test opnieuw worden uitgevoerd.

5.5 Floating-IP's

Met deze test moet worden aangetoond dat instances met het externe netwerk verbonden kunnen worden door middel van floating-IP's. De gehele test mag OpenStack geen fouten geven.

Als eerste moet er een floating-IP worden aangemaakt met het *#neutron floatingip-create* commando. Vervolgens moet de Ubuntu 14.04 cloud instance die al klaar stond worden voorzien van het betreffende floating-IP. Dit moet gebeuren met het *#nova floating-ip-associate* commando. Hierna moet de test opnieuw worden uitgevoerd.

Hierna moet er worden ingelogd op de instance via SSH. Na het inloggen kan de verbinding via het floating IP worden getest met het *#apt-get update && apt-get dist-upgrade* commando. Tevens moet er gepingt kunnen worden naar een adres die zich bevindt op het internet: http://www.google.com. Met wGET moet er een test bestand op te halen zijn van: http://www.colocenter.nl/speedtest/100mb.bin.

Deze test alleen als positief getest worden gezien als het aanmaken van het floating-IP, het koppelen van het floating-IP, het updaten met aptitude, het pingen en het ophalen van een testbestand met wGET zonder fouten is verlopen.
Als de test negatief is zal de fout moeten worden achterhaald en opgelost. De fout zit dan waarschijnlijk in Neutron. Hierna moet de test opnieuw worden uitgevoerd.

5.6 Klantdata back-up

Bij deze test wordt de mogelijkheid om met Cinder een back-up naar Swift te maken onder de loep genomen. Deze oplossing is gekozen om te voldoen aan de klantdata back-up eis. In deze test wordt er gekeken of deze oplossing ook goed werkt.

Als voorbereiding op de test staat er al een instance klaar. Deze instance heeft zijn boot schijf en één gekoppelde schijf gehost staan op de block storage node. Tevens staat er op het gekoppelde volume een bestand van 1020MB groot. Over het random bestand van 1020MB is een md5sum uitgevoerd. Hier is later in de test een vorm van integriteitscontrole mee uit te voeren.

Om de test te starten moet er eerst gecontroleerd worden of er ingelogd kan worden op de instance. Vervolgens moet er gecontroleerd worden of het volume daadwerkelijk is gekoppeld met het commando:

#cinder list

Hierna moet er een back-up worden gemaakt van het gekoppelde Cinder volume. Ook moet er direct gecontroleerd worden of de back-up wordt gemaakt. De commando's wat gegeven moet worden om dit uit te voeren is:

#cinder backup-create –-force [ID van het gekoppelde volume]
#cinder backup-list

Ook moet er worden gecontroleerd of de back-up ook daadwerkelijk in de Swift-store staat. In het overzicht moeten rond de 22 objecten met hetzelfde ID staan als de back-up. Dit moet met het commando:

#swift list volumebackups

Vervolgens moet er ook een back-up worden gemaakt van het boot volume. Om een back-up te maken van een instance moet er echter een snapshot worden gemaakt. Voor het maken van de snapshot moet er in de instance nog een sync commando worden uitgevoerd. Dit wordt gedaan om de “dirty buffers”weg te schrijven. Daarom moeten de volgende commando's worden getypt:

#sync (op de instance)
#nova image-create [instanceID] [Naam van de nieuw te maken snapshot]

Hierna moet worden gecontroleerd of de snapshot in de Glance-store voorkomt:

#glance image-list

Vervolgens moeten de beide volumes verwijderd worden, dit gebeurd met de commando's:

#nova stop [instanceID]
#nova volume-detach [instanceID] [volumeID]
#cinder delete [volumeID]
#nova delete [instanceID]
#cinder delete [volumeID van het volume waar de instance op draaide]

Na deze commando's moet het *#cinder volume-list* commando en het *#nova list* commando lege lijsten terug geven.

Vervolgens moeten de volumes weer worden gerestored uit de back-up en snapshot. Eerst moet de snapshot worden gerestored:

#openstack volume create --[SnapshotID] --size 20 backup_test
#openstack server create --flavor m1.small --key-name backup_test_key --volume [VolumeID] --nic net-id=[ID van het interne netwerk] --security-group default backup_test_instance
#nova floating-ip-associate [instanceID] [floatingIP-adres]

Hierna moet de instance actief staan in de nova listing *#nova list*. Nu moet de volume-back-up worden gerestored. Tevens moet het volume worden gekoppeld met de instance. Dit moet met het volgende commando's gebeuren:

#cinder create –-display_name backup_volume1
#cinder backup-restore –-volume [volumeID] [back_upID]
#nova volume-attach [instanceID] [volumeID]

Hierna moet er worden ingelogd naar de instance via een SSH-verbinding met Putty. Het inloggen moet goed en zonder fouten verlopen. Als er is ingelogd moet het volume worden gemount op de instance. Dit wordt gedaan met de commando's (op de instance):

#sudo mkdir /media/drive
#mount /dev/vdb /media/drive
#cd /media/drive
#ls

De output van het *#ls* commando moet “lost+found” en “randomfile” zijn. Over het bestand “randomfile” moet de

MD5sum worden uitgevoerd:

#md5sum randomfile

De uitkomst (hash) moet hetzelfde zijn als de hash die is gegenereerd als voorbereiding op de test.

Als tweede en laatste gedeelte van de test moet er nog een incremental non-disruptive back-up worden gemaakt van het gekoppelde volume.

Om deze back-up uit te voeren moet het random bestand op het gekoppelde volume eerst worden voorzien van nieuwe random data. Na het voorzien van de nieuwe random data moet er ook een MD5 checksum worden uitgevoerd over dit bestand. Dit moet met het commando's:

#dd if=/dev/urandom of=/media/drive/randomfile bs=1020M count=1
#md5sum randomfile

Met het volgende commando's moet hierna een incremental back-up van het gekoppelde volume worden gemaakt en direct gecontroleerd:

#cinder backup-create –-incremental –-force [volumeID]
#cinder backup-list

Vervolgens moet het volume worden verwijderd:

#nova volume-detach [instanceID] [volumeID]
#cinder delete [volumeID]

Na het verwijderen moet er een nieuw target-device worden aangemaakt en de incremental back-up moet naar dit target device worden weggeschreven:

#cinder create –-display-name backup_volume1 1
#cinder backup-restore –-volume [target_volumeID] [backupID]
#nova volume-attach [instanceID] [volumeID]

Op de instance waar het gekoppelde volume aan zit gekoppeld moet het volume eerst worden gemount voordat het random bestand getest kan worden met de checksum:

#mount /dev/sdb1 /media/drive1

Hierna moet met md5sum de checksum van het random bestand worden genomen:

#md5sum /media/drive1/randomfile

De checksum moet overeenkomen met de checksum die eerder is gemaakt.

De test is positief als alle uitgevoerd acties zonder foutmeldingen van OpenStack worden uitgevoerd en er daadwerkelijk back-ups zijn gemaakt en gerecovered. Ook moeten de checksums van het gekoppelde volume in de normale back-up en de incremental back-up met elkaar overeenkomen.

Als de test negatief is dan zal de voorkomende fout onderzocht en verholpen moeten worden. Hierna moet de test opnieuw worden uitgevoerd.

5.7 OpenStack systeemback-up

In deze test worden de verschillende databases, de configuratie bestanden en de overige bestanden handmatig gekopieerd, dat is de manier om een back-up te maken van de OpenStack services. De bestanden van de services en de service-databases worden gekopieerd naar een lokale map op de controller voor deze test. Vervolgens wordt deze map als TAR-bestand ingepakt en geüpload naar Swift.
Hierna zullen de databases weer worden gerecovered. Alle acties moeten kunnen worden uitgevoerd zonder dat er foutmeldingen opkomen en het systeem instabiel wordt.
De files en directories worden niet gerecovered aangezien dit overbodig is: de bestanden worden dan eerst uit hun directories gekopieerd en vervolgens weer terug geplaatst, dat toont verder niets aan. Als de files en directories worden gebackupt en vervolgens gerecovered staan de indentieke bestanden op deze plek terug, daarom wordt er alleen een back-up gemaakt van de files en directories en ze worden niet gerecovered.

Allereerst moet de lokale map worden aangemaakt worden op de controller node:

#mkdir /var/lib/backup_dir

Hierna moet er een back-up worden gemaakt van de component-databases en Ceilometer backend-database. Dit gebeurd met de commando's:

#mysqldump -u root -p –-opt nova > /var/lib/backup_dir/DB/nova.sql
#mysqldump -u root -p –-opt cinder > /var/lib/backup_dir/DB/cinder.sql
#mysqldump -u root -p –-opt glance > /var/lib/backup_dir/DB/glance.sql
#mysqldump -u root -p –-opt keystone > /var/lib/backup_dir/DB/keystone.sql
#mysqldump -u root -p –-opt neutron > /var/lib/backup_dir/DB/neutron.sql
#mongodump –-username ceilometer –-password [MongoDB password]–-out /var/lib/backup_dir/Ceilometer_dump

Om de test voot te zetten moet er ook een back-up worden gemaakt van alle OpenStack-componenten. Hier onder staat een tabel met daarin de nodes en directories waar een back-up van gemaakt moet worden. Dit betekend dat de files en directories gekopieerd moeten worden naar de directory */var/lib/backup-dir/[naam van de node/[naam van de service]* op de controller node. Als de files op andere nodes staan dan moeten ze met SCP naar de controller node worden gekopieerd. Al deze files en directories worden gekopieerd om te testen of het kopiëren goed gaat en om uiteindelijk een realistisch back-up bestand te krijgen. Er moet een back-up worden gemaakt van de volgende directories en files:

Tabel 2: De files en directories waar een back-up van gemaakt moet worden gemaakt.

Node	Component/service	Directories
Controller	Nova	/etc/nova /var/log/nova /var/lib/nova
Controller	Glance	/etc/glance /var/log/glance /var/lib/glance/image-cache
Controller	Keystone	/etc/keystone /var/log/keystone /var/lib/keystone

Controller	Cinder	/etc/cinder /var/log/cinder /var/lib/cinder
Controller	Neutron	/etc/neutron /var/log/neutron /var/lib/neutron
Controller	Horizon	/etc/openstack-dashboard
Controller	Ceilometer	/etc/ceilometer
Controller	Iptables	/etc/iptables_contr
Compute 1	Nova	/etc/nova /var/log/nova /var/lib/nova/buckets /var/lib/nova/CA /var/lib/nova/images /var/lib/nova/keys /var/lib/nova/networks /var/lib/nova/tmp
Compute 2	Nova	/etc/nova /var/log/nova /var/lib/nova/buckets /var/lib/nova/CA /var/lib/nova/images /var/lib/nova/keys /var/lib/nova/networks /var/lib/nova/tmp
Compute 3	Nova	/etc/nova /var/log/nova /var/lib/nova/buckets /var/lib/nova/CA /var/lib/nova/images /var/lib/nova/keys /var/lib/nova/networks /var/lib/nova/tmp
Block storage node	Cinder	/etc/cinder /var/log/cinder /var/lib/cinder
Object proxy-server	Swift	/etc/swift
Object storage server 1	Swift	/etc/swift /etc/fstab /etc/rsyncd.conf /etc/default/rsync
Object storage server 2	Swift	/etc/swift /etc/fstab /etc/rsyncd.conf /etc/default/rsync
Object storage server 3	Swift	/etc/swift /etc/fstab /etc/rsyncd.conf /etc/default/rsync
NTP-node	NTP	/etc/ntp.conf /etc/iptables_ntp
SSH-jumpnode	SSH	/etc/ssh ~/.ssh/authorized_keys /etc/iptables_ntp
Repository node	Repository	/etc/apt/mirror.list /etc/proftpd /etc/iptables_ntp

De /var/lib/backup_dir directory heeft hierna dus de volgende layout:

- Controller
  - Nova
  - Glance
  - Keystone
  - Cinder
  - Neutron
  - Horizon
  - Ceilometer
  - Iptables
- Compute_1
  - Nova
- Compute_2
  - Nova
- Compute_3
  - Nova
- Block
  - Cinder
- Object_proxy
  - Swift
- Object_store_1
  - Swift
- Object_store_2
  - Swift
- Object_store_3
  - Swift
- NTP
  - NTP
- SSH
  - SSH
- Repository
  - Repository
- DB

Als de databases, files en directories allemaal opgenomen zijn in de */var/lib/backup_dir/* dan moet de inhoud ingepakt worden in een enkel bestand. Dit moet gedaan worden met het volgende commando:

#tar -zcvf backup_[datum].tar.gz /var/lib/backup_dir

Vervolgens moet het back-up bestand in de Swift backend worden geplaatst en worden gecontroleerd of dit daadwerkelijk is gebeurd. In plaats van Swift kunnen er ook andere backend gebruikt kunnen worden. In deze test wordt gebruik gemaakt van Swift.

#swift upload backup backup_datum.tar.gz
#swift list backup

Op dat moment is de volledige OpenStack-back-up geüpload naar de Swift back-end. Hierna moet de recovery van de databases nog worden getest.

Eerst moet de back-up directory van de controller worden verwijderd. Vervolgens moet de back-up weer uit de Swift-storage worden gehaald en uit worden gepakt:

#swift download backup backup_datum.tar.gz
#tar -C /var/lib/backup_dir -xvf backup_datum.tar.gz

Hierna moeten de databases van deze services een voor een worden gerestored. Eerst worden de services uitgeschakeld en vervolgens de betreffende service in de database gerestored, deze commando's en te stoppen services zijn te vinden in tabel 3. Na het restoren van een service-database moeten de betreffende services weer worden gestart, ook hier mogen geen fouten worden aangegeven. Ook in de logs van de betreffende service mogen bij het starten geen fouten opkomen.
Tenslotte moet de service worden gecontroleerd door een actie uit te voeren. De acties die uitgevoerd moeten worden om de services te testen staan ook in tabel 3.

Tabel 3: de services die gestopt moeten worden samen met de commando's die uitgevoerd moeten worden om de databases te restoren.

Te stoppen services	Restore database	Command	Uit te voeren actie
cinder-api cinder-scheduler	cinder	mysql -u root -p cinder < cinder.sql	cinder create volume nova volume-attach
glance-api glance-registry	glance	mysql -u root -p glance < glance.sql	glance image-list glance image-create
keystone	keystone	mysql -u root -p keystone < keystone.sql	create user create keypair
neutron-server	neutron	mysql -u root-p neutron < neutron.sql	create floating-ip
nova-api nova-cert nova-conductor nova-consoleauth nova-novncproxy nova-scheduler	nova	mysql -u root -p nova < nova.sql	boot instance associate-floating-ip
Ceilometer-agent-central ceilometer-agent-notification ceilometer-alarm-evaluator ceilometer-alarm-notifier ceilometer-api ceilometer-collector	Ceilometer	Mongorestore –h 10.0.1.2 Ceilometer_dump	ceilometer meter-list ceilometer statistics

Als alle acties in deze test uitgevoerd kunnen worden zonder dat OpenStack een fout geeft is de test geslaagd. Als de test niet is geslaagd zal er gekeken moeten worden waar de fout vandaan kwam en deze oplossen. Vervolgens moet de test dan opnieuw worden gestart.

5.8 Control panel

In deze test moet worden aangetoond dat gebruikers van de cloud kunnen inloggen in een grafisch control panel. Bij het opzetten van het proof of concept is OpenStack Horizon opgezet, dit is het control panel binnen OpenStack. Met deze test moet dus worden gekeken of OpenStack Horizon goed functioneert en normale users bij dit dashboard kunnen komen.

Aan het begin van de test moet worden gecontroleerd of een normale user in kan loggen in het control panel. In de voorbereiding voor aanvang van de test is er een user aangemaakt genaamd “test”. Deze user heeft de role “user”, zit in de tenant “test” en bevindt zich in het domain “default”.
Het inloggen gebeurd door het aanroepen van de controller in een browser. Deze test wordt uitgevoerd vanaf mijn laptop. In de browser moet Horizon worden opgevraagd met URL: “*https://10.0.1.2/horizon*'.
In het volgende login scherm komen velden te staan waar de login gegevens ingevuld moeten worden. Tijdens of na het inloggen mogen geen fouten opkomen, dit kan duiden op verkeerde instellingen van Keystone of Horizon.

Na het inloggen moeten er overzichten van alle OpenStack-services op het scherm verschijnen. Alle tabs moeten worden aangeklikt. Geen van deze tabs mag en foutmelding geven.
Hierna moet weer worden uitgelogd. Ook hier mag geen foutmelding opkomen.

De test is positief als er geen foutmeldingen opkomen bij het testen. Als de test negatief is dan moeten de fouten worden achterhaald en afgehandeld, hierna moet er dan opnieuw worden gestart met de test.

5.9 Update

Met deze test moet worden aangetoond dat alle nodes die aangesloten zitten op het management netwerk kunnen updaten via de repository node. De nodes die op het management-netwerk zitten aangesloten en moeten kunnen updaten van de lokale repository zijn: utility node, controller node, block storage node, network node, compute nodes, object proxy node, object storage nodes, SSH-jump node, NTP-node.

Als eerste is het van belang dat de repository node zelf de updates aan de mirror zonder foutmeldingen binnenhaalt. Het updaten van de repository moet aangeroepen worden met het *#apt-mirror* commando. Vervolgens moet op elke node een update van de packages worden uitgevoerd. Met *#apt-get update* kan worden gecontroleerd of de betreffende node zijn package-list goed kan binnenhalen. Vervolgens moet met *#apt-get dist-upgrade* worden gecontroleerd of alle packages op de package-list die een nieuwere versie hebben dan de packages op het systeem correct kunnen worden opgehaald vanaf de lokale repository. Bij deze test is het ook van belang dat de OpenStack packages worden geüpdatet.

Om te controleren dat de packages ook echt zijn geüpdatet moeten een aantal packages worden gekozen waarop wordt gecontroleerd of ze echt geüpdatet zijn, dit is een steekproef. Ook moeten er een aantal packages uit de OpenStack repository worden gecontroleerd op dezelfde manier.
De versies van de gekozen packages moeten eerst worden gecontroleerd bij de repository waar ze oorspronkelijk vandaan komen (in het proof of concept zijn dit de i3d-repository en de ubuntu-cloud-repository). De gecontroleerde packages moeten versienummers hebben die overeenkomen met de versienummers van de packages die geüpdatet zijn op de nodes. De packages die gecontroleerd worden zijn:

- network-manager (i3d-repository)
- linux-generic (i3d-repository)
- openssh-client (i3d-repository)
- nova-common (ubuntu-cloud-repository) (alleen op de controller node)

Om de versienummers van de bovenstaande packages te controleren moet er gebruik worden gemaakt van het *#apt-cache policy* commando op de repository node. Met dit commando wordt er gekeken welke update candidates er zijn voor de betreffende package.
Als de updates geen fouten genereren en de versienummers na het updaten komen overeen is de test

geslaagd, de nodes kunnen dan updaten van de lokale repository.

5.10 Verbruik meten

In het proof of concept is het van belang dat het verbruik van “klanten” gemeten kan worden. In OpenStack wordt voor dit doel Ceilometer gebruikt. Deze test is eigenlijk best simpel: er moet een actie worden uitgevoerd door een user en Ceilometer moet deze actie loggen.
De actie die moet worden uitgevoerd in deze test is het downloaden van een image uit de Glance-store naar de lokale opslag op de controller node. Dit gebeurd met het commando:

#glance image-download "[imageID]" > /dev/null

Vervolgens moet er worden gecontroleerd of Ceilometer deze actie heeft gelogd. Dit gebeurd met het commando:

#ceilometer statistics -m image.download -p 60

In het overzicht wat commando als output geeft moet er een nieuwe entry zijn aangemaakt waar de “period start” cel dezelfde waarde heeft als het tijdstip waarop het commando is aangeroepen.

Vervolgens moet ook het vebruik worden gemeten van een instance die netwerkverkeer genereerd. Een goede manier om netwerkverkeer te genereren is het update van de instance. Op dit moment moet er worden ingelogd op de instance die is klaargezet voor de test. De instance is op basis van Ubuntu 14.04 cloud en heeft een floating-IP gekoppeld.
Er moet dus een update worden uitgevoerd op de instance met:

#apt-get update
#apt-get dist-upgrade -y

Vervolgens moet het gegenereerde verbruik te zien zijn in Ceilometer. Dit verbruik kan worden gecontroleerd met het commando:

#ceilometer sample-list -m network.incoming.bytes

Dit commando moet een uitput geven met de timestamp van het moment dat de update commando's waren aangeroepen, de instance die de aanroep deed, de meter die het verbruik heeft gemeten en het daadwerkelijke verbruik.

Als beide scenario's zijn uitgevoerd zonder fouten te geven en een output geven zoals is voorspeld dan is de test geslaagd. Als de test niet slaagd dan moet er worden gekeken naar de fouten die er voorkomen. Deze fouten moeten vervolgens worden achterhaald en verholpen. Vervolgens moet de test opnieuw worden worden uitgevoerd.

5.11 Performance groei

In deze test moet de groei van de aansturende nodes inzichtelijk worden. Om deze groei inzichtelijk te maken moet de controller node, network node, block storage node en object proxy node worden getest op hun performance, dit zijn de zogenaamde *aansturende nodes*. De uitkomst van deze tests moet een getal zijn die

een aantal instances aangeeft waarbij er een nieuwe aansturende node bij moet worden geplaatst. De output van deze tests geldt ook direct als input voor de winstberekening. Als uitkomst van de test geldt dus geen positief/negatief.

Aan het begin van de test worden alle vier de aansturende nodes aan een nultest onderworpen. De nultest geeft een tijdsduur als uitkomst. Het moment waarop er groei moet plaatsvinden in de OpenStack omgeving is als de aansturende nodes over de 200% slowdown heen gaan. Op het moment dat de testresultaten van de nultest met 200% zijn toegenomen dan is het aantal instances waar in die stap mee getest is de uitkomst van de test. Om achter de groei te komen zal er worden getest in stappen van vijf instances. De nieuw op te zetten instances moeten een Ubuntu 14.04 cloud image als basis hebben en ze moeten worden opgezet op bootable Cinder disks.
Alle nieuwe instances moeten een Apache2 server opzetten en een MySQL-database. In het hoofdstuk “tests opzetten” van de handleiding wordt uitgelegd hoe deze servers op moeten worden gezet.

Met Apache2 AB moet de webserver load worden gegenereerd. In een script kan dit commando zo worden opgezet zodat het een X aantal requests naar een Apache webserver wegstuurt. Om dit aantal requests te reguleren (per sec) wordt ook een sleep commando gegeven in het script (webserver.sh):

```
#!/bin/bash

while true
do
        ab -n 7 -c 1 http://[IP-adres van de instance]/
        sleep 0.05
done
```

Om ook een SQL-database te simuleren bij deze HTTP-requests wordt er per 5 HTTP-requests een SQL-INSERT uitgevoerd. De SQL-simulatie wordt uitgevoerd met het volgende script (DB.sh):

```
#!/bin/bash

while true
do
        echo "insert into test (a,b,c) values ('data','data','data');" | mysql -u
       root -ptoor test
        sleep 0.035
        echo "delete from test where a = 'data';" | mysql -u root -ptoor test
        sleep 0.035
done
```

Met het script wordt om de X seconden een SQL-INSERT of een SQL-DELETE commando gegeven. Dit aantal seconden is 5 keer lager dan de webserver.
Met Sysbench zal er een meetpunt worden opgezet.

Bij deze tests wordt expres geen gebruik gemaakt van OpenStack Rally benchmarking om load te genereren op instances, dit omdat er van het standpunt uit is gegaan dat die OpenStack commando's vrijwel nooit worden aangeroepen in een kleinere OpenStack cloud omgeving terwijl alle klant instances actief draaien: Dit zal dus geen realistische load genereren.

5.11.1 Controller node

Als eerste moet de controller node aan een test worden onderworpen. Om het nulpunt te bepalen bij de test zal er met OpenStack rally load moeten worden gegenereerd op de controller node. Met de Rally-test genaamd: “create-and-delete-user.json” zal het nulpunt worden gemeten. De instelling van de Rally-test moeten default zijn: 100 keer een user aanmaken en weer verwijderen. De tijd dat de test nodig heeft om uit te voeren, terwijl alle instances uit staan, is het nulpunt.
Het commando wat moet worden gebruikt om OpenStack Rally de benchmark te laten uitvoeren is:

#rally task start /usr/share/rally/samples/tasks/scenarios/keystone/create-and-delete-user.json

Alle stappen (5 instances) worden getest terwijl de instances wel aan staan maar geen taak hebben(idle), en met load van een webserver en MySQL database. De load van de webservers wordt gehouden op die van een kleine webserver: 140 HTTP requests/sec en 28 MySQL writes per seconde. Alle stappen worden gemeten met de OpenStack Rally test “create-and-delete-user.json”. Ook zullen de resources op de controller node in de gaten worden gehouden met het commando #top. De opbouw van de webservers is hetzelfde als de opbouw van de webserver test in de architectuurfase.
Met de volgende commando's worden de webserver en MySQL load generaties aangezet:

#./webserver.sh
#./DB.sh

5.11.2 Block storage node

Hierna wordt de block storage node getest. Om in deze test het nulpunt te bepalen zal er random data moeten worden geschreven naar een bestand op het volume waar de instance ook staat, terwijl er maar één instance aan staat (de test instance). De load wordt net als in de controller test gegenereerd door load van AB en MySQL-writes, doormiddel van de scripts (webserver.sh en DB.sh) die al zijn aangemaakt: dit is de realistische load.
Om ook te testen wat er gebeurd bij volledige belasting (constant alle instances laten schrijven naar het Cinder volume) wordt er per stap ook random data geschreven op de Cinder volumes van de instances. Op elke instance moet het volgende commando worden uitgevoerd om een random bestand aan te maken, deze goede rechten te geven en random data naar dit bestand te schrijven:

#touch random && chmod 755 random && time head -c 1000M </dev/urandom >random

Met het volgende commando wordt op de test instance een random bestand aangemaakt, de rechten goed gezet en random data naar het bestand geschreven:

#touch random && chmod 755 random && time head -c 1000M </dev/urandom >random

Bij het eind van deze test wordt gekeken naar de uitkomst van de realistische load, de maximale load doormiddel van random data schrijven wordt aan de test toegevoegd als vergelijking.

Met de volgende commando's worden de webserver en MySQL load generaties aangezet:

#./webserver.sh
#./DB.sh

Ook wordt de IO op de block storage node in de gaten gehouden met het commando *#iotop*.

Een opmerking bij deze test is dat het maximale aantal test instances op 20 ligt. Dit komt doordat elk Cinder-volume 20GB is (net als in de winstberekening) en op de block storage node ligt de totale schijfruimte op:

((250GB + 250GB + 250GB + 250GB) / 2) – (2GB + 65GB + 8GB) = 425GB rumte voor block volumes

Het totaal aantal instance komt nu uit op:

425GB / 20GB per instance = 21 Instances.

Echter wordt het totaal aantal van 20 instances aangehouden omdat dit beter in de test past. Ook is er op dan nog wat marge over op de Cinder node.

5.11.3 Network node

Vervolgens wordt de network node getest. In deze test zijn er twee meetpunten. Het eerst meetpunt is de “create-and-delete-networks.json” benchmark in OpenStack Rally en het tweede meetpunt is het downloaden van een testbestand van http://mirror.nl.leaseweb.net/speedtest/100mb.bin: dit is een mirror waar je speedtest mee kunt uitvoeren, vrij stabiel in snelheid dus.
Het benchmark meetpunt wordt gemeten vanaf de utility node en is gericht op de resources van de network node. Het download meetpunt wordt gemeten vanaf een instance en is gericht op de bandbreedte die de network node doorlaat. Het is binnen deze test zo dat het meetpunt dat eerder over de 200% performance slowdown heen gaat het aantal instances aangeeft die de network node maximaal aan kan.
De nulpunten worden in deze test vastgesteld door de Rally benchmark zonder actieve instances te laten draaien en te downloaden uit de speedtest zonder actrieve instances.

Het OpenStack rally meetpunt wordt gedraaid door het uitvoeren van het volgende commando:

#rally task start /usr/share/rally/samples/tasks/scenarios/neutron/create-and-delete-networks.json.

Het download meetpunt wordt opgezet door het uitvoeren van het volgende commando:

#time wget http://mirror.nl.leaseweb.net/speedtest/10mb.bin

De load wordt in deze test ook gegenereerd door het uitvoeren van AB en MySQL op de instances. Met de volgende commando's worden de webserver en MySQL load generaties aangezet:

#./webserver.sh
#./DB.sh

5.12 Management network security

In deze test moeten de aansluitingen van het proof of concept met het i3D netwerk worden getest op security, deze machines, de DMZ-nodes, zijn namelijk het doorgeefluik tussen het i3D-netwerk en de interne netwerken. De DMZ-nodes bestaan uit de controller node, NTP-node, SSH-jump node en de repository node.

Om te testen of portscans nog mogelijk zijn op de servers die in het DMZ staan worden er eerst verschillende

Nmap portscan uitgevoerd. De verschillende scans die worden uitgevoerd op de DMZ-nodes zijn: TCP SYN scan, TCP connect scan (of UDP connect scan), FYN scan. Wel zal er worden aangegeven in Nmap dat de hele range van porten wordt gescand, ook de private port range (49152 ~ 65535).
Deze portscans mogen alleen als uitkomst geven dat de porten openstaan van de, aan het i3D netwerk, aangeboden services. Dit betreft: controller node: port 443, NTP node: port 123, SSH-jump node: port 50385. De repository node mag geen port 80 als reactie op de portscan geven aangezien deze port alleen geopend hoeft te zijn op het moment dat er een apt-mirror update wordt gedaan.6 Verder mogen er aan de kant van het i3D-netwerk geen open porten te scannen zijn.

De verschillende commando's die gegeven moeten worden voor de nmap portscans zijn:

#nmap X.X.X.X -p 1-65535 (= TCP SYN scan)
#nmap -sT X.X.X.X -p 1-65535 (= TCP connect scan)
#nmap -sU X.X.X.X -p 1-65535 (= UDP connect scan)
#nmap -sF X.X.X.X -p 1-65535 (= FYN scan)

De DMZ-nodes moeten ook afzonderlijk worden getest om te zien of “hun services” afgeschermd zijn.

Als eerste zal de SSH-jump node worden getest, hier is het van belang dat er alleen ingelogd kan worden met de juiste SSH-key. Er zullen dus pogingen ondernomen moeten worden om in te loggen zonder SSH-key, dus met inlognaam/wachtwoord combinatie. Er moet worden getest of er ingelogd kan worden als “admin1997” en met “root”, dit zijn de enige twee normale users op het systeem. Tevens moet de server bij het inloggen met de betreffende SSH-key vragen om een “passphrase”.

Als tweede zal er worden gecontroleerd of er nog via HTTP ingelogd kan worden naar het dashboard op de controller node. Het inloggen via HTTP op het dashboard is uitgezet en via HSTS. Als er ingelogd wordt op het dashboard via HTTP dan moet deze service melden dat dit niet kan of een “connection timed out” foutmelding geven, aangezien er geen verbinding gemaakt kan worden via port 80.

De NTP-node en repository node hoeven niet te worden getest met een connectietest aangezien de NTP-server alleen connectie zoekt met de time-servers van i3D. De repository node opent zijn port 80 alleen op het moment dat hij een downloadverzoek heeft aan de i3D-mirror.

6. Testrapportages architectuurfase

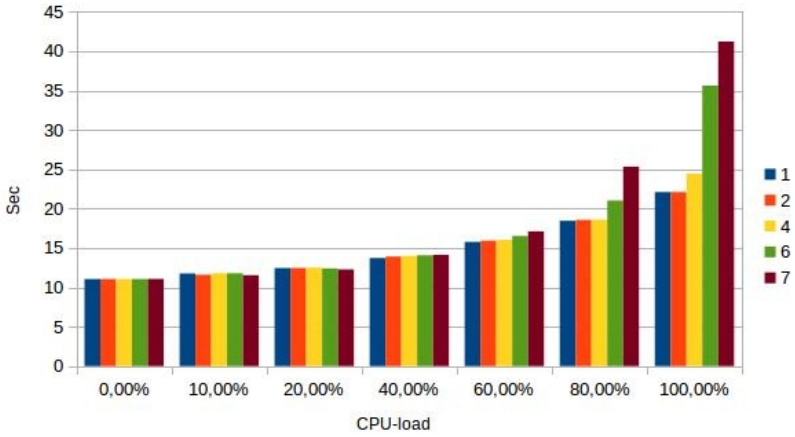
In dit subhoofdstuk worden de testresultaten besproken van de tests die zijn gehouden in de architectuurfase.

6.1 Compute node los testen

Bij het testen van de losse compute nodes ging het om de rauwe performance van de compute nodes zelf in de OpenStack omgeving. Deze performance is getest door de CPU en de HDD performance onder de loep te nemen. Tevens is hiernaast ook een test uitgevoerd waar een webserver werd nagebootst op de instances.

6.1.1 CPU-test

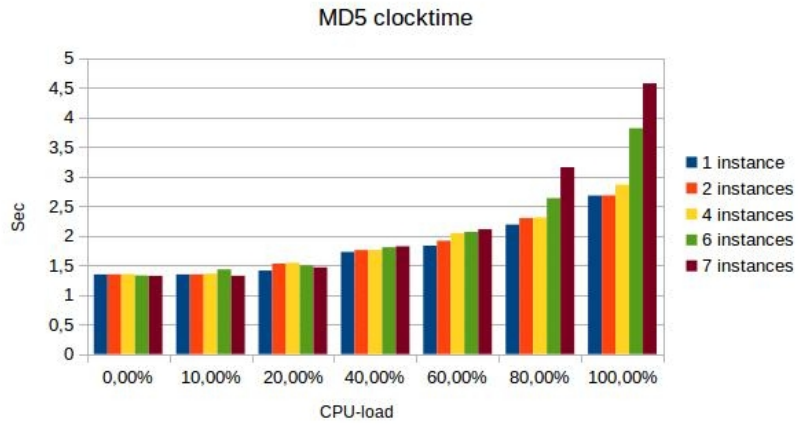
In de eerste test werd er load gegenereerd met *Stress-ng*, de meting werd op één instance verricht met *Sysbench*. In afbeelding 1 zijn de testresultaten te zien.



Afbeelding 6: De testresultaten van de CPU-load test gemeten met Sysbench.

Bij 0.00% staan de ijkpunten aangegeven. De load waardes tot de 60% load mogen niet hoger worden dan 200% van het ijkpunt.
Zoals op de grafiek te zien is blijft de load hier ruimschoots onder.
Verder is uit de resultaten af te leiden dat pas bij constante belasting op 80% van de CPU met volledige instance bezetting de performance slowdowns boven de 200% van het ijkpunt komen, de compute nodes kunnen qua processorskracht de instances dus makkelijk aan.
Deze test is dus positief.

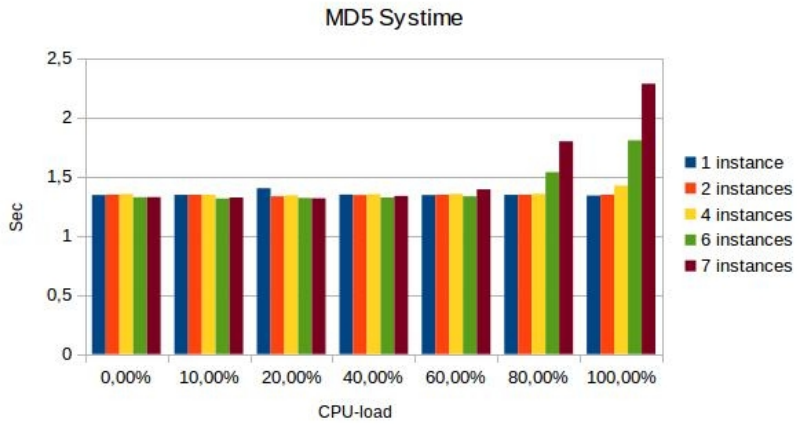
Er is ook op 80% en 100% CPU-load getest. Dit is gedaan om te zien hoe de performance zich houdt als de andere instances op de compute node zeer hoge load genereren. Vooral bij de 100% is goed te zien dat vooral met volledige load de performance inzakt. Voor de rest is te zien dat de quad-core processor die aanwezig is in de DL120 G6 servers de CPU-load goed verdeeld.



Afbeelding 7: De testresultaten van de CPU-load test gemeten met MD5sum: clocktime.

Ook is de CPU nog test met een ander meettool namelijk MD5sum. Door MD5sum te combineren met het *time* commando binnen Linux kan de tijd worden opgemeten hoe lang het duurt voordat een MD5-hash is berekend terwijl de andere instances op de compute node verschillende CPU-loads genereren.

In afbeelding 2 zijn de resultaten te zien van de CPU-test gemeten met MD5sum terwijl er gekeken wordt naar de “clock time”. De clock time is de tijd die het uitvoeren van het commando nodig heeft volgende een echte klok. Ook hier is weer het zelfde resultaat te zien als in de test waar gemeten wordt met *Sysbench*.

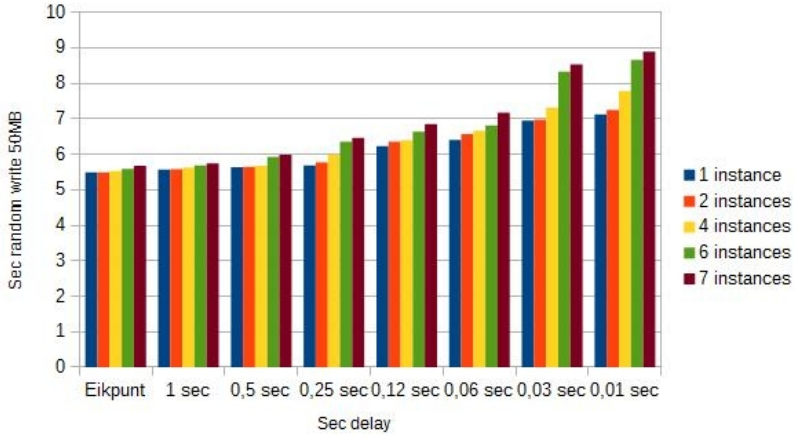


Afbeelding 8: De testresultaten van de CPU-load test gemeten met MD5sum:systime.

Als laatste testsessie in de CPU-test wordt er weer gemeten met MD5sum, echter is er nu gekeken naar de “system time”. De uitslag is te zien in afbeelding 3. De system time geeft aan hoe lang het systeem over het uitvoeren van het commando heeft gedaan.
Ook hier is weer hetzelfde resultaat als in de voorgaande CPU-tests waar te nemen.

6.1.2 HDD-test

In de tweede test werd de harde schijf getest die lokaal in de compute node aanwezig is. In afbeelding 4 is te zien wat de testresultaten zijn bij de test op de lokale HDD. Op afbeelding 5 is te zien wat de testresultaten zijn bij de test op de Cinder block storage node.



Afbeelding 9: De testresultaten van de HDD-load read/write test gemeten met het time commando

De test werd opgezet door alle instances die aanwezig waren op de compute node random data te laten schrijven naar een bestand met de commando's:

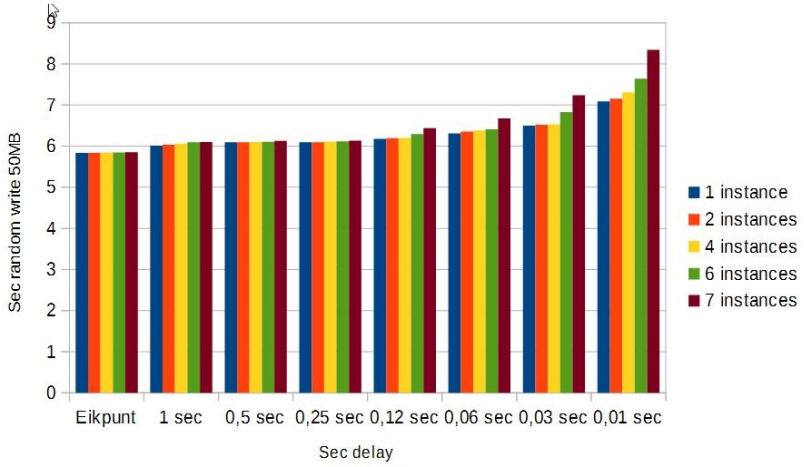
```
#sudo head -c 100K
</dev/urandom
>/home/ubuntu/rand.random

#sudo head -c 100K
</dev/urandom >/media/drive2/rand.random
```

Om de load wat realistischer te maken zijn de instances zo ingesteld dat ze niet constant een stream zouden schrijven naar de schijf. In plaats hier van werd er een bestand van 100KB om de X aantal seconden geschreven. Als er een constante stream geschreven zou worden dan zou het commando direct een groot deel van de lijn naar de HDD opeisen.

In de testresultaten van de lokale storage is te zien dat de performance niet zeer dramatisch omhoog schiet. Tevens is over heel de range te zien dat de testresultaten niet meer dan 200% van het ijkpunt worden. De conclusie bij deze twee tests is dat een compute node op basis van een DL120 G6 server qua HDD (SATA 300

7200 RPM) makkelijk 7 instances en normale HDD-belasting aankan. Deze test is dus ook positief.

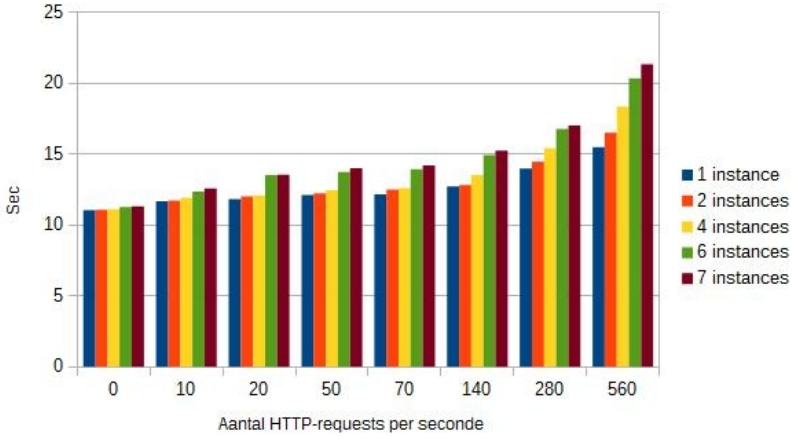
In de testresultaten van de Cinder storage is ook te zien dat de performance niet snel achteruit gaat. Uiteindelijk is de performance van de Cinder storage net iets hoger dan de lokale opslag. Dit is echter wel te verklaren aangezien er in de Cinder storage gebruik wordt gemaakt van RAID10. In deze grafiek is ook te zien dat de testresultaten niet meer dan 200% slowdown geven over de gehele range van de test. De test is dus positief.



Afbeelding 5: De testresultaten van de HDD-load read/write test op de CInder storage.

6.1.3 Webserver test

Als laatste testsessie in de “compute node los testen” test is de webserver test uitgevoerd. Deze test was nogal complex in het opzetten. Op elke instance werd een scipt opgezet om met het *AB* commando load te genereren naar de Apache server die ook aanwezig was op datzelfde instance. Uiteindelijk waren er dus 7 Apache servers actief. Elke instance stuurde met AB requests naar zijn eigen Apache server.
Naast de webserver waren er op de instances ook scripts aanwezig waarmee een database load gegenereerd kon worden. Dit was gedaan omdat een webserver vrijwel altijd een database nodig heeft.
Er is getest met het aantal HTTP-requests per seconde. Hier was altijd de database load gekoppeld die vijf maal minder was dan het aantal HTTP-req/s.
Bij 50 HTTP-req/s waren er dus 50 / 5 = 10 database aanvragen per seconde. Het aantal database aanvragen per seconde is lager gehouden dan het aantal HTTP-req/s omdat nooit elk HTTP-request ook een database request wordt.



Afbeelding 6: De testresultaten van de webserver test gemeten met Sysbench.

In afbeelding 6 zijn de testresultaten te zien. Het ijkpunt is vastgesteld rond de 11 seconde. Bij het meten werd die niet snel veel meer. Normale load van een kleine webserver bedraagt ongeveer maximaal 140 HTTP-req/s. In de grafiek is te zien dat bij 140 HTTP-req/s het aantal seconde op het meetpunt niet meer dan 200% is toegenomen, de test is dus positief. Zelfs bij een belasting van 560 HTTP-req/s (en 112 DB-requests) op 7 instances is de performance slowdown nog geen 200%. De conclusie bij deze twee tests is dat een compute node op basis van een DL120 G6 server qua HDD (SATA 300 7200 RPM) makkelijk 7 instances en kleine webservers aankan, de webservers kunnen zelfs een stukje groeien, dan kan de node het nog steeds aan zonder 200% slowdown.

Uit de grafieken is op te maken dat alle compute node tests positief waren, in elke test is de “normale load”

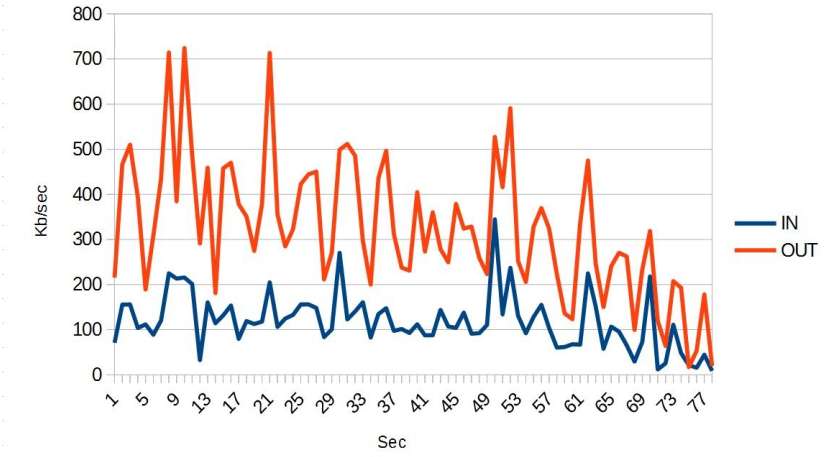
waarde immers niet 200% hoger dan het nulpunt. De conclusie bij deze tests is dus dat kleine/middelgrote webservers op de instances kunnen draaien zonder andere instances te hinderen op de compute node.

6.2 Dataverbindingen tussen de nodes testen

In deze test zijn de dataverbindingen getest binnen OpenStack. Met deze test kon de performance eis: “Het netwerkverkeer over heel het netwerk mag onder normale omstandigheden (uitgezonderd piekverkeer) niet over 50% van de lijn komen.” positief of negatief worden getest.

In de test is er gemeten bij de aansluiting van de controller node op het management netwerk van OpenStack. Dit is gedaan omdat hier alle requests aankomen. Op dit punt kan dus ook worden gemeten wat het bandbreedteverbruik is.

Er is load gegenereerd door OpenStack Rally activiteiten te laten uitvoeren die users van de cloud ook zouden uitvoeren. In afbeelding 7 is te zien hoe het bandbreedteverbruik zich heeft gehouden tijden de loadgeneratie. Het uitgangspunt van de test is dat normaal verkeer niet boven 50% van de lijn uitkomt.
De bandbreedte van de lijn betreft 1Gb. Het maximale verbruik op de lijn tijdens de test was ongeveer 750Kb/s. De resultaten laten zien dat het management netwerk van OpenStack makkelijk zeer veel groei aankan, het verbruik kan ruim 500 keer meer worden wil het piekverbruik de helft van de lijn bandbreedte aantikken. Dit betekend dat de test positief was.
De conclusie bij deze test is dat het verkeer op het management netwerk minimaal met 50000% (1mb/s → 500Mb/s) kan groeien alvorens de helft van de lijn te bezetten.

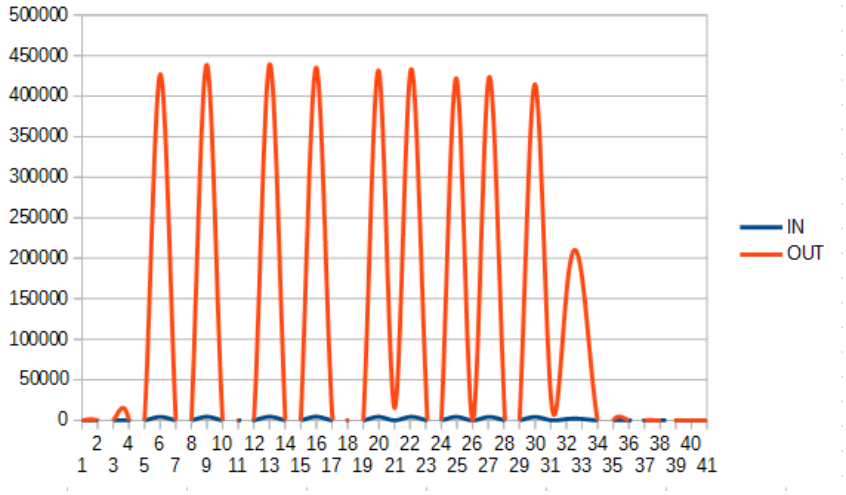


Afbeelding 7: De testresultaten van de dataverbinding test.

6.3 Dataverbinding naar de (block) storage testen

In deze test is de dataverbinding tussen de block storage node en de compute nodes getest. De eis hierbij was: “Het netwerkverkeer van en naar de storage mag niet zo hoog worden dat de verbinding wordt dichtgetrokken. Tijdens normaal (uitgezonderd piekverkeer) gebruik mag het niet hoger worden dan 50% van de lijn.”
Er is bij deze test gemeten bij de aansluiting van de block storage node op het management netwerk. Dit is gedaan omdat op deze node de requests en het verkeer aangaande Cinder aankomen. Op dit punt is het bandbreedteverbruik van de block storage goed te meten.

Er is load gegenereerd door zeven instances tegelijk random data te laten schrijven naar een file op de Cinder storage. Op afbeelding 8 is goed te zien dat de instances de dataverbinding goed gebruiken. Het gebruik bij de test blijft ook ruim onder de helft van de totale bandbreedte van de lijn.
Uit de resultaten is af te leiden dat data met pieken wordt verstuurd naar de ontvanger. Deze pieken kunnen hoger uitkomen dan de helft van de bandbreedte, maar uiteindelijk is de meeste tijd de lijn vrij rustig, dit betekend dat met deze bandbreedte de opstelling nog veel groter geschaald kan worden.
De testresultaten zijn zo dat ze onder 50% van de totale bandbreedte van de lijn blijven. De test is dus geslaagd.



Afbeelding 8: De testresultaten van de storage verbinding test.

7. Testrapportage testfase

In dit subhoofdstuk worden de testresultaten besproken van de tests die zijn gehouden in de testfase.

7.1 Redundante storage

In deze test is de failover van het RAID-array in de Cinder node getest. Om de test te beginnen is er eerst gecontroleerd met het hpssacli programma of het RAID-array nog in goede conditie was. Deze controle is uitgevoerd met de commando's:

#hpssacli
#ctrl all show config

Op afbeelding 9 is de output van van deze commando's te zien. Het RAID-array was in goed staat aan het begin van de test.

```
logicaldrive 1 (465.6 GB, RAID 1+0, OK)

physicaldrive 2I:1:1 (port 2I:box 1:bay 1, SATA, 250 GB, OK)
physicaldrive 2I:1:2 (port 2I:box 1:bay 2, SATA, 250 GB, OK)
physicaldrive 2I:1:3 (port 2I:box 1:bay 3, SATA, 250.0 GB, OK)
physicaldrive 2I:1:4 (port 2I:box 1:bay 4, SATA, 250 GB, OK)
```

Afbeelding 9: Het overzicht van het RAID-array, de logische schijf en de fysieke schijven zijn in orde.

Na deze controle is de harde schijf uit de server verwijderd. Na het verwijderen is opnieuw de controle met hpssacli uitgevoerd, deze gaf aan bij de eerste schijf dat hij “missing” was. De hele opstelling draaide wel door.

Vervolgens is de lege HDD in het slot van de Cinder node ingevoerd. De controle met het hpssacli programma gaf aan dat de schijf in de “recovering” state was. Het array was de schijf dus opnieuw aan het opbouwen.
Na een aantal keer gecontroleerd te hebben met hpssacli was het recoveren afgelopen, het hpssacli programma gaf op dat moment aan dat het RAID-array weer in orde was zoals op afbeelding 9.

De test was geslaagd aangezien er geen fouten op zijn gekomen tijdens de test en het array aan het eind van de test weer functioneel was. De conclusie bij deze test is dat er met een DL120 G6 server met HP p410 RAID-kaart een RAID-array (RAID10) is op te zetten waar één schijf kan falen en dit bij tijdig ingrijpen niet zorgt voor dataverlies.

7.2 Schaalbare resources

In deze test is de mogelijkheid om resources te kunnen schalen getest. Dit is gedaan om de volgende systeemeis te kunnen waarborgen: “De "klanten" moeten in het proof of concept de mogelijkheid hebben om hun resources te kunnen schalen”.

Als eerste is er een snapshot gemaakt van de instance. Met deze actie wordt er van de instance een nieuwe image gemaakt. Dit gebeurd met

nova image-create [Instancenaam][test-instance] [te creëren imagenaam][scale_test]

Echter, omdat het een live-snapshot is moet voor het aanroepen van dir commando eerst een vorm van data-synchronisatie uitgevoerd worden.

#sync
#fsfreeze -f /mnt

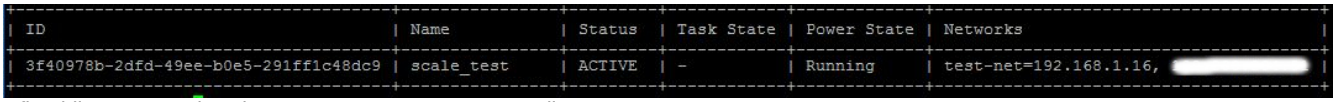
Om met de test door te gaan zijn alle gekoppelde Cinder-volumes ontkoppeld:

#nova volume-detach [instanceID] [VolumeID]

Met het volgende commando is vervolgens een instance op basis van een nieuwe flavor en een nieuwe image worden gestart:

#nova boot –-flavor m1.medium –-image scale_test –-nic net-id=[ID] security-group default key-name scale_test scale_test
#neutron floatingip_create ext-net
#nova floating-ip-associate scale_test [FloatingIP]

Tot dit moment is de test voorspoedig verlopen. Dit is te zien in het instance overzicht (afbeelding 9).



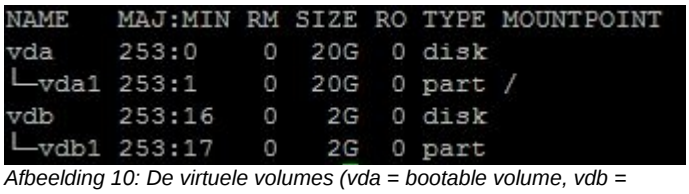
Afbeelding 9: De actieve instance, opgeroepen met nova list.

Vervolgens is het Cinder volume aan de nieuwe instance aangekoppeld met:

#nova volume-attach scale_test [VolumeID]

Op dit punt stond de instance weer online en in dezelfde staat als de oude instance. Het enige verschil was dus de geschaalde resources.

Hierna werden nog een aantal testjes uitgevoerd om te zien of de resources ook goed werden herkend in de instance en de Nova-service.
Het *#top* commando in de instance geeft terug dat de instance beschikte over 2049900KB RAM. Dit betekend dat het aantal GB RAM overeenkomt met de opgegeven flavor.
Met het commando *#lsblk* kunnen de virtuele volume's worden opgevraagd samen met de hoeveelheid storage die ze aanbieden, dit is te zien op afbeelding 10. De nieuwe flavor biedt 20 GB storage aan.



Afbeelding 10: De virtuele volumes (vda = bootable volume, vdb = gekoppeld Cinder volume).

Op dit moment was het aangetoond dat de scaling was gelukt, in ieder geval voor de instance zelf. Om er helemaal zeker van te zien moest de Nova-service worden geraadpleegd.
Op afbeelding 11 is de output te zien van het *#nova show* commando. Hier is te zien dat de nieuwe instance is gekoppeld aan de nieuwe flavor (upscaling). Op afbeelding 12 is te zien wat de flavors inhouden, waaronder ook “m1.small”.

Property	Value
OS-DCF:diskConfig	MANUAL
OS-EXT-AZ:availability_zone	nova
OS-EXT-SRV-ATTR:host	server1990
OS-EXT-SRV-ATTR:hypervisor_hostname	server1990.dedi.i3d.net
OS-EXT-SRV-ATTR:instance_name	instance-0000008a
OS-EXT-STS:power_state	1
OS-EXT-STS:task_state	-
OS-EXT-STS:vm_state	active
OS-SRV-USG:launched_at	2016-05-02T07:58:55.000000
OS-SRV-USG:terminated_at	-
accessIPv4	
accessIPv6	
config_drive	
created	2016-05-02T07:59:14Z
flavor	m1.small (2)
hostId	bb3407e77704d88d65cbe57351b2925586193540491 20b01dcfd360e
id	3f40978b-2dfd-49ee-b0e5-291ff1c48dc9
image	scale_test (061b7018-3d24-41cf-ad17-ed79a8faccbf)
key_name	scale_test2
metadata	{}
name	scale_test
os-extended-volumes:volumes_attached	[{"id": "9459c574-d288-4467-b3ad-32e1c7274d5d"}]
progress	0
security_groups	default
status	ACTIVE
tenant_id	c3f2d1c2bb6a4a93a8a4e92369206663
test-net network	192.168.1.16, [redacted]
updated	2016-05-02T08:51:18Z
user_id	9830508ee3a346b3a19dfa3f4675db9a

Afbeelding 11: De output van het nova show commando. Ook hier is te zien dat de nieuwe flavor is gekoppeld aan de instance.

ID	Name	Memory_MB	Disk	Ephemeral	Swap	VCPUs	RXTX_Factor	Is_Public
1	m1.tiny	512	1	0		1	1.0	True
2	m1.small	2048	20	0		2	1.0	True
3	m1.medium	4096	40	0		2	1.0	True
4	m1.large	8192	80	0		4	1.0	True
5	m1.xlarge	16384	160	0		8	1.0	True
b8b8fb7c-f15d-45de-b317-6316b4ab8863	m1.small_small	1024	10	0		1	1.0	True

Afbeelding 12: De verschillende flavors. "m1.small_small" is handmatig aangemaakt. De overige flavors zijn default.

Bij de test hebben zich verder geen fouten voorgedaan. Tevens is de instance goed opgeschaald. Dit betekend dat de test positief is. De conclusie bij deze test is dat het in het proof of concept op basis van OpenStack Liberty mogelijk is om instances te kunnen schalen en voorzien van meer/minder resources.

7.3 Storage bestellen

Met deze test moest worden aangetoond dat “klanten” in OpenStack ook extra storage zouden kunnen bestellen, met ander woorden: schaalbare storage. In OpenStack is extra storage op te vragen bij de Cinder-service.
Bij de aanvang van de test stond er al een kale Ubuntu 14.04 cloud instance klaar. Deze was opgezet met:

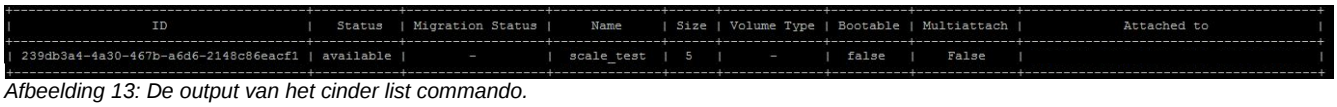
#nova boot –-flavor m1.small –-image scale_test –-nic net-id=[ID] security-group default key-name scale_test

scale_test
#neutron floatingip_create ext-net
#nova floating-ip-associate scale_test [FloatingIP]

Na de start van de test is er een nieuw Cinder-volume aangemaakt. Dit is gedaan met:

#cinder create –-display-name scale_test 5

Ter controle is de cinder list opgevraagd. Hier stond ook de nieuwe volume in. Dit is te zien op afbeelding 13.

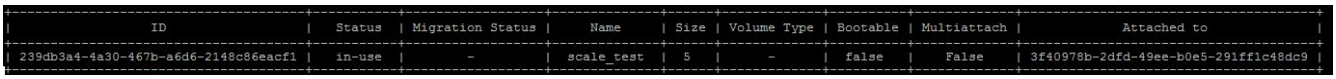


ID	Status	Migration Status	Name	Size	Volume Type	Bootable	Multiattach	Attached to
239db3a4-4a30-467b-a6d6-2148c86eacf1	available	-	scale_test	5	-	false	False	

Afbeelding 13: De output van het cinder list commando.

Vervolgens is het volume gekoppeld met de instance, het gevolg is te zien op afbeelding 14. Dit is gedaan met:

#nova volume-attach scale_test [VolumeID]



ID	Status	Migration Status	Name	Size	Volume Type	Bootable	Multiattach	Attached to
239db3a4-4a30-467b-a6d6-2148c86eacf1	in-use	-	scale_test	5	-	false	False	3f40978b-2dfd-49ee-b0e5-291ff1c48dc9

Afbeelding 14: De output van Cinder na het koppelen van het volume.

Cinder gaf aan dat de instance is gekoppeld aan het device */dev/vdb*. Door lokaal op de instance het commando *#lsblk* uit te voeren is gecontroleerd of het volume ook echt was gekoppeld aan */dev/vdb*. De output van het *#lsblk* commando is te zien op afbeelding 15.

Vervolgens is met Fdisk het volume gepartitioneerd en voorzien van een ext3 filesystem. Deze actie is uitgevoerd zodat er een functionaliteitstest op het volume uitgevoerd kon worden. Op afbeelding 16 is de output van het *#lsblk* commando te zien na het partitioneren en uitrusten met een filesystem.

```
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda    253:0    0  20G  0 disk
└─vda1 253:1    0  20G  0 part /
vdb    253:32   0   5G  0 disk
```

Afbeelding 15: De output van het #lsblk commando voor de aanmaak van de partitie.

Hierna is het nieuwe volume getest door random data naar een random bestand op het volume te sturen. De grootte van dit bestand werd gehouden op 64MB*64=4096MB. Dit is gedaan met het commando:

#dd if=/dev/urandom of=/media/scale_test/random bs=64M count=64

```
NAME   MAJ:MIN RM SIZE RO TYPE MOUNTPOINT
vda    253:0    0  20G  0 disk
└─vda1 253:1    0  20G  0 part /
vdb    253:32   0   5G  0 disk
└─vdb1 253:33   0   5G  0 part /media/scale_test
```

Afbeelding 16: De output van het #lsblk commando na de aanmaak van de partitie en filesystem.

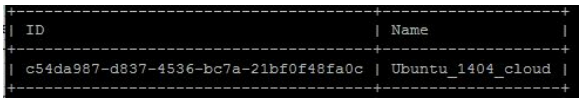
Bij het schrijven van de random data is er geen errorcode omhoog gekomen. Dit betekend dat de test positief is. De conclusie bij deze test is dat het in het proof of concept op basis van OpenStack Liberty mogelijk is om instances te kunnen voorzien van block volumes.

7.4 Hosten van een virtueel OS

Het doel van deze test was het aantonen dat er instances van in ieder geval één distributie konden draaien in

het proof of concept.
Bij de start van de test was de Ubuntu 14.04 cloud image al gelist in de Glance-store. Dit was te zien met het *#glance image-list* commando (afbeelding 17).

#nova boot –-flavor m1.small –-image scale_test –-nic net-id=[ID] security-group default key-name instance_key Ubuntu_1404_cloud



Afbeelding 17: De Ubuntu 14.04 cloud image in de Glance-store.

Of de instance daadwerkelijk is aangemaakt is gecontroleerd met het *#nova list* commando, dit is te zien op afbeelding 18.

Afbeelding 18: De output van het nova list commando na het aanmaken van de instance.

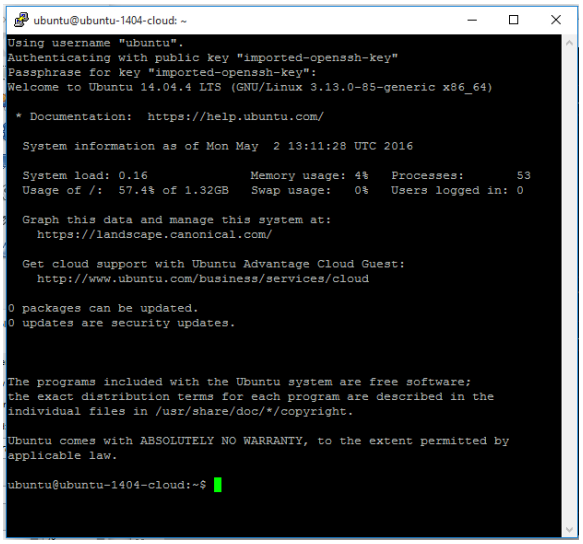
OpenStack gaf op dit punt in ieder geval al aan dat de instance geen error meegaf. Vervolgens moest er eerst een floating IP gekoppeld worden aan de instance:

#nova floating-ip-associate Ubuntu_1404_cloud [floating IP]

Om toegang te verkrijgen tot de instance moest er een SSH-verbinding worden aangemaakt naar de instance. Dit werd gedaan met Putty. De enige manier van inloggen naar de Ubuntu instance is via een SSH-key. Met deze reden is eerst met Puttygen een private key gegenereerd op basis van het OpenStack keypair. De SSH-verbinding naar de instance met Putty kon hierna worden opgezet.

Op afbeelding 19 is te zien hoe het welkomstscherm van de Ubuntu 14.04 cloud instance er uit ziet nadat er is ingelogd.

Bij deze test zijn geen fouten opgetreden daarom is deze test positief. De conclusie bij deze test is dat het in het proof of concept op basis van OpenStack Liberty mogelijk is om instances te kunnen hosten op basis van een Linux OS.



Afbeelding 19: Het welkomstscherm van de Ubuntu 14.04 cloud instance.

7.5 Floating-IP

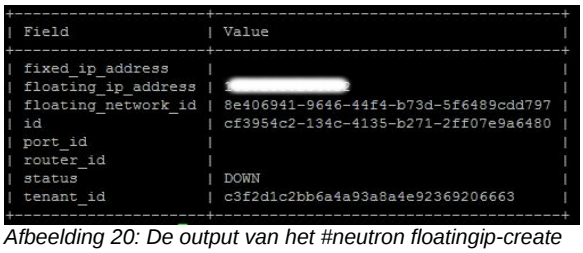
In deze test moet worden aangetoond en getest of het aanmaken en gebruiken van floating-IP's functioneel is binnen het proof of concept.

Aan het begin van de test staat er al een kale instance van Ubuntu 14.04 cloud klaar. Deze instance moet worden voorzien van een floating-IP. Eerst moet hier voor een floating-IP worden aangemaakt. Het floating-IP wordt gemaakt met het commando:

#neutron floatingip-create ext-net

Als output van dit commando komt er een tabel omhoog die informatie over het gemaakt floating-IP. Deze tabel is te zien op afbeelding 20.
Met het volgende commando is het floating-IP gekoppeld met de instance die al klaar stond:

#nova floating-ip-associate float_test XXX.XXX.XXX.XXX



Field	Value
fixed_ip_address	
floating_ip_address	[redacted]
floating_network_id	8e406941-9646-44f4-b73d-5f6489cdd797
id	cf3954c2-134c-4135-b271-2ff07e9a6480
port_id	
router_id	
status	DOWN
tenant_id	c3f2d1c2bb6a4a93a8a4e92369206663

Afbeelding 20: De output van het #neutron floatingip-create commando.

Dit commando geeft geen output. Om deze instance toch te controleren op gekoppelde floating-IP is het *#nova list* commando gebruikt. In dit overzicht stond het floating-IP ook beschreven. Het koppelen van het floating-IP aan de instance was op dat moment gelukt en zonder fouten verlopen.

Om de connectiviteit van het gekoppelde floating-IP te testen is met Putty een SSH-verbinding gemaakt naar de betreffende instance. Het verbinden en inloggen bij de instance was ook succesvol verlopen.

Om de verbinding via het floating-IP-adres verder te testen is de instance geüpdatet met het commando:

#apt-get update && apt-get dist-upgrade

Ook dit commando werd zonder fouten uitgevoerd: de package lists zijn zonder fouten geüpdatet en alle betreffende packages zijn zonder fout geüpdatet.

Vervolgens is het adres www.google.com gepingt vanaf de instance. Dit lukte ook zonder fouten: er kwamen ping replies terug naar de instance.

Als laatste is er met wGET een testbestand opgehaald van het adres http://www.colocenter.nl/speedtest/100mb.bin. Ook dit testbestand is zonder fout gedownload van de opgegeven URL.

Alle testonderdelen zijn zonder fout uitgevoerd. Dit betekend dat deze test positief is. De conclusie bij deze test is dat het in het proof of concept op basis van OpenStack Liberty mogelijk is om instances te kunnen voorzien van een IP-adres waarmee ze met een extern netwerk kunnen verbinden.

7.6 Klantdata back-up

Voor de start van de test moest er nog even worden gecontroleerd of alle testonderdelen goed stonden. Er werd met Putty via SSH ingelogd op de instance: Deze stond aan en werkte. Vervolgens werd er met het *#cinder list* commando opgevraagd welke volumes er gekoppeld waren. Ook het “backup_test1” volume was in de lijst aanwezig en was teven gekoppeld met de juiste “Ubuntu_1404_cloud” instance.

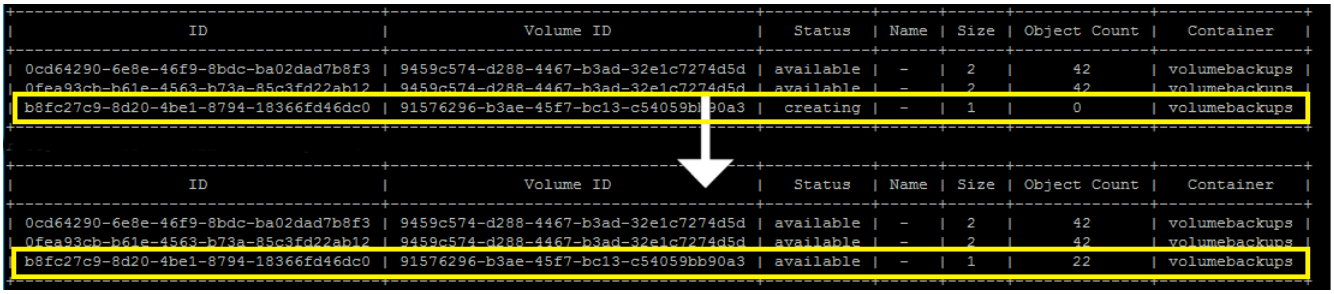
Hierna kon de echte test beginnen. Als eerste werd er een non-disruptive back-up van het gekoppelde volume gemaakt en direct gecontroleerd met de commando's:

#cinder backup-create –-force [volumeID van het gekoppelde volume]
#cinder backup-list

Op afbeelding 21 is het verloop van de back-up in het *#cinder backup-list* overzicht te zien. Hierna is er gecontroleerd of de back-up in de Swift-store staat. Dit is gedaan met het commando:

#swift list volumebackups

De output van dit commando liet 22 objecten met hetzelfde ID als de back-up zien. Dit betekende dat de back-up daadwerkelijk in de Swift-backend stond.



Afbeelding 21: De output van het #cinder backup-list commando. Boven: bezig met de back-up, onder: de back-up is klaar.

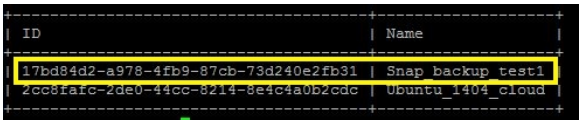
Vervolgens is er een snapshot gemaakt van het boot-volume van de instance in Cinder. Voordat de snapshot werd gemaakt is er nog het *#sync* commando gegeven op de instance. Dit werd gedaan om “dirty buffers” weg te schrijven.

#sync
#nova image-create [instanceID] [Naam van de nieuwe snapshot]

Na dit commando moest de nieuwe snapshot als image in de Glance lijst staan. Dit is gecontroleerd met het commando:

#glance image-list

De output van de glance image list was als verwacht: De snapshot werd weergeven als nieuwe image in de Glance-store. De output van het *#glance image-list* commando is te zien in afbeelding 22.



Afbeelding 22: De output van het #glance image-list commando. De snapshot is opgenomen in de Glance-store.

Hierna zijn de instance en het gekoppelde volume verwijderd:

#nova stop [instanceID]
#nova volume-detach [instanceID] [volumeID]
#cinder delete [volumeID]
#nova delete [instanceID]
#cinder delete [volumeID van het volume waar de instance op draaide]

Als output gaven de *#cinder volume-list* en *#nova list* commando's lege lijsten terug. Dit was verwacht, de instance en het gekoppelde volume waren immers verwijderd.

Hierna zijn de volumes weer opgezet om de restore-functie te testen. Er werd begonnen met het aanmaken van het volume en de instance:

#openstack volume create –snapshot [SnapshotID] --size 20 backup_test

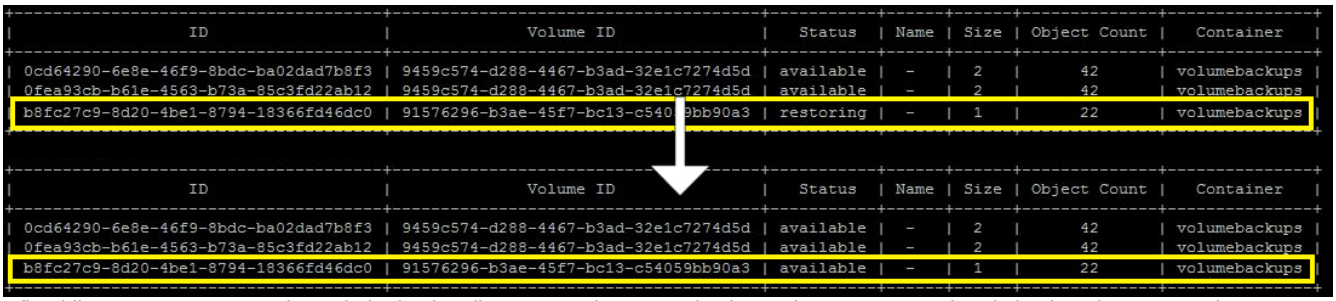
#openstack server create --flavor m1.small --key-name backup_test_key --volume [VolumeID] --nic net-id=[ID van het interne netwerk] --security-group default backup_test_instance
#nova floating-ip-associate [instanceID] [floatingIP-adres]

Bij het uitvoeren van deze commando's kwamen geen fouten naar boven. Met het *#nova list* commando is gecontroleerd of de instance ook echt draaide, dit was het geval.

Om het volume te kunnen restoren moet er eerst een target volume zijn. Hierna kan de backup gerestored worden op de target. Deze acties zijn uitgevoerd met de volgende commando's:

#cinder create –-display-name backup_volume1 1
#cinder backup-restore –-volume [volumeID] [back_upID]

Het restore-process is te zien in afbeelding 23.

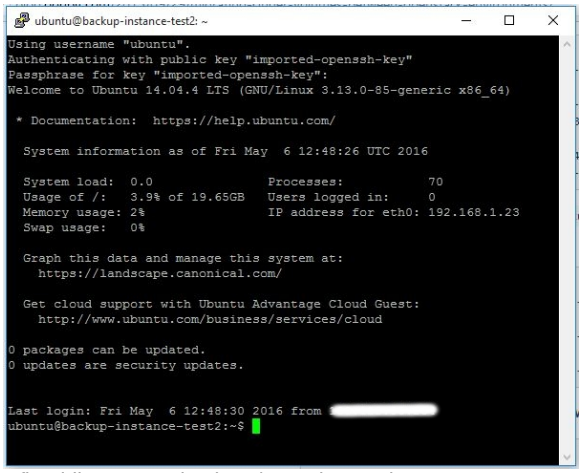


Afbeelding 23: De output van het #cinder backup-list commando. Boven: bezig met het restoren, onder: de back-up is gerestored.

Op dit moment kon het volume worden gekoppeld met de instance. Dit gebeurde met het commando:

#nova volume-attach [instanceID] [volumeID]

Nu was alles weer in oude staat (zoals voor de back-up en restore). Er moest nog worden ingelogd op de instance om het gekoppelde volume te controleren en te kijken of er ingelogd kon worden, Dit deed ik met Putty via een SSH-verbinding.



Afbeelding 24: De back-up/snapshot werkt!

Het inloggen naar de instance functioneerde naar behoren. Dit is te zien op afbeelding 24. Tevens ging het mounten van het gekoppelde volume naar behoren.
Om integriteitsproblemen op te merken is aan het begin van de test een MD5sum gedaan over het random bestand op het gekoppelde volume. Hier kwam de volgende waarde uit:

64b772af6933cc96ac647c07f9358437

Aan het eind van de test is er weer een MD5sum gedaan over het random bestand nadat er een back-up is gemaakt van het volume en deze is gerestored. Hier kwam de volgende waarde uit:

64b772af6933cc96ac647c07f9358437

Het random bestand op het gekoppelde volume is dus gelijk gebleven.

Dit betekend dat het eerste gedeelte van de test positief was. Het tweede gedeelte van de test bestand uit het uitvoeren van een incremental non-disruptive back-up van het gekoppelde volume.

Allereerst is er een nieuwe full back-up gemaakt van het gekoppelde volume:

#cinder backup-create –-force [VolumeID]

Vervolgens is het random bestand op het gekoppelde volume voorzien van nieuwe random data. Het bestand is door MD5sum gehaald. De uitkomst van het MD5sum commando was:

0f420e96ed3bb5cb17cc082ed6e74287

Hierna is er een incremental back-up gemaakt van het gekoppelde volume:

#cinder backup-create –-incremental –-force [volumeID]

Met het *#cinder backup-list* commando is gecontroleerd of de back-up echt aangemaakt werd. Dit is te zien op afbeelding 25. Er werd een nieuwe back-up aangemaakt die de wijzigingen in de full-back-up aangeeft. Aangezien vrijwel heel het volume is veranderd is het increment ook groot.

Afbeelding 25: De output van het #cinder backup-list commando. Boven: bezig met het maken van de back-up, onder: de back-up is gemaakt.

Om de recovery van het volume te testen moet het gekoppelde volume eerst worden afgekoppeld en verwijderd:

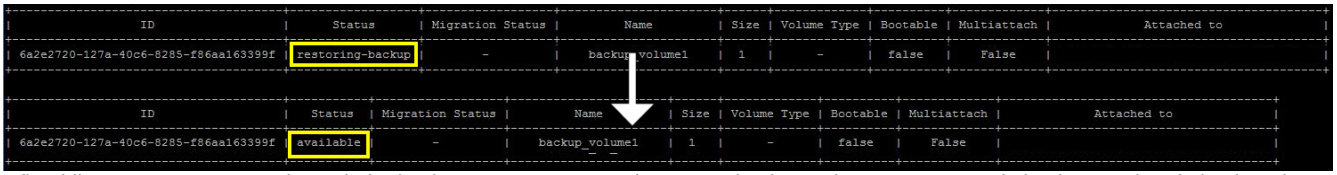
#nova volume-detach [instanceID] [volumeID]
#cinder delete [volumeID]

Ook nu moet er eerst een target device worden aangemaakt:

#cinder create –-display-name backup_volume1 1

Hierna is de incremental back-up van het gekoppelde volume gerecovered naar het target volume:

#cinder backup-restore –-volume [target_volumeID] [backupID]



Afbeelding 26: De output van het #cinder backup-restore commando. Boven: bezig met het restoren van de backup, onder: de back-up is gerestored.

De restore van het gekoppelde volume is zonder problemen verlopen. Dit is ook te zien op afbeelding 26.
Om een checksum op het bestand op het gekoppelde volume uit te voeren moet het volume eerst gemount worden. Op de instance worden hiervoor het commando uitgevoerd:

#mount /dev/sdb1 /media/drive1

Vervolgens kan met md5sum de checksum van het random bestand worden genomen. Deze checksum moet hetzelfde zijn als de checksum die genoteerd is voor de incremental back-up. Dit commando moet op de instance uitgevoerd worden:

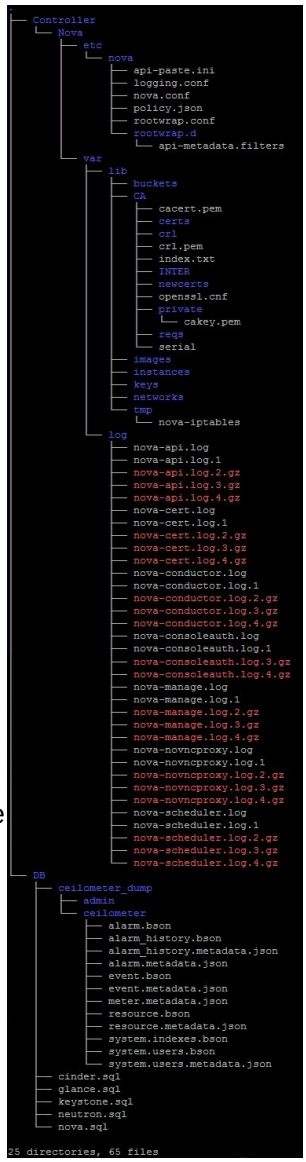
#md5sum /media/drive1/randomfile

De uitkomst van dit commando was:

0f420e96ed3bb5cb17cc082ed6e74287

Deze uitkomst komt overeen met de checksum van voor de incremental back-up. Dit betekend dat het random bestand niet is veranderd bij de back-up.

Alle commando's en acties in deze test zijn uitgevoerd door OpenStack zonder een foutmelding. Tevens komen de checksums overeen en functioneerde de instance na de rocovery naar behoren.
De klant back-uptest is positief. De conclusie bij deze test is dat het mogelijk is om live instances en hun gekoppelde (live) block volumes te back-uppen binnen het proof of concept op basis van OpenStack Liberty.



Afbeelding 27: directoryoverzicht van de backup map tijdens de test.

7.7 OpenStack systeemback-up

Eerst is de map gemaakt waar alle files worden verzameld:

#mkdir /var/lib/backup_dir
#mkdir /var/lib/backup_dir/DB

In deze directory zijn eerst de verschillende service-databases gebackupt:

#mysqldump -u root -p –-opt nova > /var/lib/backup_dir/DB/nova.sql
#mysqldump -u root -p –-opt cinder > /var/lib/backup_dir/DB/cinder.sql
#mysqldump -u root -p –-opt glance > /var/lib/backup_dir/DB/glance.sql
#mysqldump -u root -p –-opt keystone > /var/lib/backup_dir/DB/keystone.sql

#mysqldump -u root -p –-opt neutron > /var/lib/backup_dir/DB/neutron.sql
#mongodump –-username ceilometer –-password [MongoDB password] -h 10.0.1.2 –-out /var/lib/backup_dir/

De back-ups van de databases creëren verliep zonder problemen: er kwamen geen foutmeldingen op.
Vervolgens zijn alle files en directories van de OpenStack services gekopieerd naar de back-up directory, ook dit verliep zonder problemen, er waren geen lees of schrijffouten. Op afbeelding 27 is de output van het *#tree* commando te vinden, dit commando is uitgevoerd na het maken van de database back-ups en de nova back-ups.

Na het “inruimen van de back-up directory ging de test door met het inpakken van deze directory en deze vervolgens opslaan in de Swift store:

#tar -zcvf backup_[datum].tar.gz /var/lib/backup_dir
#swift upload backup backup_[datum].tar.gz
#swift list backup

Het list commando gaf terug dat het ingepakte back-up bestand in de Swift-store stond. Dit gedeelte van de test verliep ook zonder problemen. Tot dit punt was de back-up procedure getest. Hierna werd de recovery van de databases getest.

Om de recovery te testen is eerst het ingepakte bestand en de back-up directory verwijderd. Vervolgens is het ingepakte back-up bestand gedownload en uitgepakt:

#rm -r /var/lib/backup_dir
#rm /var/lib/backup_[datum].tar.gz
#swift download backup backup_[datum].tar.gz
#tar -C /var/lib/backup_dir -xvf backup_[datum].tar.gz

Nu stonden de bestanden weer uitgepakt in de */var/lib/backup_dir* directory.

Eerst is de Nova-service gerecovered: alle Nova-services op de controller zijn uitgeschakeld en de database back-up is teruggeplaatst met *#mysql -u root -p cinder < cinder.sql*. Om te testen of de back-up van de Nova-service goed is verlopen werd hierna een instance geboot met het *#boot instance* commando en een floating-IP gekoppeld met *#associate-floating-ip*.
De recovery van de Nova database werd zonder fouten uitgevoerd en de twee commando's werden ook zonder problemen uitgevoerd.
Vervolgens zijn alle andere services (Cinder, Glance, Neutron, Keystone en Ceilometer) gerecovered en getest volgens de testaanpak in het testplan. Geen van de database-recoveries en tests die hier op volgde gaf een problemen of foutmeldingen.

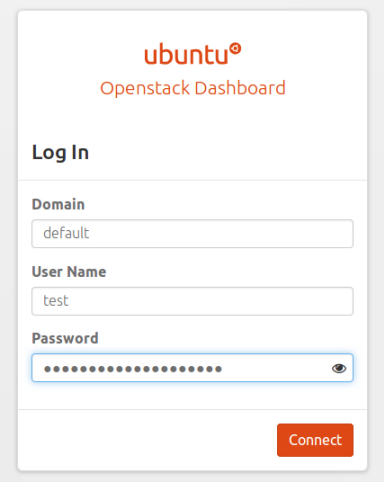
Gedurende de hele test heeft OpenStack geen fout gegeven en de volledige back-up procedure verliep goed, de test is dus positief. De conclusie bij deze test is dat dat het OpenStack systeem naar een back-up geschreven kan worden en de databases kunnen zonder problemen weer gerecovered worden.

7.8 Control panel

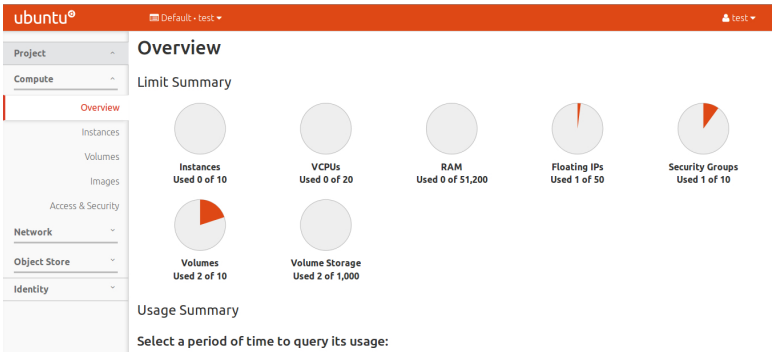
In het begin van de test is het OpenStack dashboard opgevraagd in de browser op de gebruikte laptop. Zonder foutmelding werd het loginscherm van OpenStack Horizon weergeven, dit is te zien op afbeelding 28.
De credentials van de test user werden ingevoerd in dit overzicht en zonder fouten kwam het dashboard

tevoorschijn.
Op afbeelding 29 is dit dashboard te zien. Links bovenin het scherm staat dat het overzicht bij tenant test / domain default hoort, dit klopt. Rechts bovenin staat dat het overzicht bij user test hoort, dit is ook juist.

Hierna zijn alle tabbladen van het overzicht doorgeklikt om te zien of dit allemaal functioneerde. Bij het klikken zijn er geen fouten opgekomen. Dit betekend dat de test positief is.
De conclusie bij deze test is dat de opstelling op basis van OpenStack Liberty ook met een grafische omgeving aangestuurd kan worden en dit functioneert in het proof of concept.



Afbeelding 28: Het loginscherm van OpenStack Horizon.



Afbeelding 29: Het dashboard na inloggen van de test user.

7.9 Update

Binnen deze test is de functionaliteit van de lokale repository server getest. Als eerst is de functie van repository server zelf getest. Dit is gedaan door het *#apt-mirror* commando uit te voeren. Dit commando werd zonder fouten uitgevoerd en heeft de nodige packages geüpdatet.

Vervolgens is op alle nodes de commando's *#apt-get update* en *#apt-get dist-upgrade* uitgevoerd. Op geen enkele node mochten deze update commando's fouten genereren. Op geen enkele node hebben deze commando's fouten gegenereerd. Op elke node werden de package-lists en de packages zelf netjes gedownload en geïnstalleerd. Als sources was dit de lokal repository node, dit is te zien op afbeelding 30.

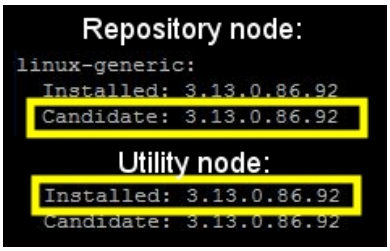
```
Hit ftp://10.0.1.12 trusty-updates/main amd64 Packages
Hit ftp://10.0.1.12 trusty-updates/restricted amd64 Packages
```
*Afbeelding 30: De package-listst werden correct gedownload***.**

Als steekproef is hierna naar een aantal packages op de nodes gekeken. De versienummers van de packages op de nodes moesten overeenkomen met de versienummers van de packages op de oorspronkelijke mirror (waar de lokale mirror zijn packages vandaan haalt). Op het moment van testen waren de versienummers:

- linux-generic (i3d-repository) *3.13.0.86.92*
- openssh-client (i3d-repository) *1:6.6p1-2ubuntu2.7*

- nova-common (ubuntu-cloud-repository) (alleen op de controller node) *2:12.0.2-0ubuntu1~cloud0*

De eerste drie packages uit de bovenstaande opsomming zijn gecontroleerd op elke node. De *nova-common* package is alleen gecontroleerd op de controller node.
De bovenstaande packages kwamen op de nodes qua versienummer allemaal overeen met de packages op de oorspronkelijke mirror (mirror.i3d.net). Op afbeelding 31 is een voorbeeld te zien: het versienummer van het package *linux-generic* op de oorspronkelijke mirror komt hier overeen met het versienummer van de package op de utility node.



Afbeelding 31: De versienummers van de linux-generic package komen overeen.

Omdat er in de test geen fouten zijn opgekomen bij het uitvoeren van de update commando's en alle versienummers klopte is de test geslaagd. De conclusie bij deze test is dat de nodes in het proof of concept succesvol kunnen updaten vanaf een lokale repository die opgenomen is in de OpenStack omgeving.

7.10 Verbruik meten

In deze test was het van belang dat het verbruik van cloud-gebruikers gemeten kon worden. Om te voldoen aan deze eis is een OpenStack component genaamd “Ceilometer” geïnstalleerd. Door het uitvoeren van een aantal commando's wordt in deze functionaliteitstest gekeken of Ceilometer ook daadwerkelijk het verbruik van gebruikers meet.

Eerst is er gekeken of Ceilometer het downloaden van een image registreert. Met het volgende commando is er eerst een image uit de Glance-store gedownload naar de controller node en direct weggegooid:

#glance image-download "[imageID]" > /dev/null

Vervolgens is er in Ceilometer gecontroleerd of deze actie is gelogd:

#ceilometer statistics -m image.download -p 60

De uitkomst is te zien op afbeelding 32.

Max	Min	Avg	Sum	Count	Duration	Duration Start	Duration End
259392000.0	259392000.0	259392000.0	259392000.0	1	0.0	2016-05-16T09:02:32.719000	2016-05-16T09:02:32.719000

Afbeelding 32: Het verbruik na het downloaden van een image.

Vervolgens is ook getest of Ceilometer het bandbreedteverbruik van een instance bij zou kunnen houden. Eerst is er een instance gestart op basis van Ubuntu 14.04 cloud en er is een floating-IP gekoppeld met deze instance. Vervolgens is op deze instance een update uitgevoerd. Hierna was er in Ceilometer te zien dat de instance bandbreedte en hoeveel bandbreedte de instance verbruikt had. Dit is bekeken met het commando:

#ceilometer sample-list -m network.incoming.bytes

De uitkomst van het bovenstaande commando is te zien op afbeelding 33.

Resource ID	Name	Type	Volume	Unit	Timestamp
instance-00000094-2dd41c5f-6cbf-403e-9df6-2cc7ea8c959d-tapfdafcde9-7f	network.incoming.bytes	cumulative	32782375.0	B	2016-05-16T09:22:32.638000
instance-00000094-2dd41c5f-6cbf-403e-9df6-2cc7ea8c959d-tapfdafcde9-7f	network.incoming.bytes	cumulative	24816.0	B	2016-05-16T09:12:32.620000

Afbeelding 33: Het verbruik van een instance na het updaten.

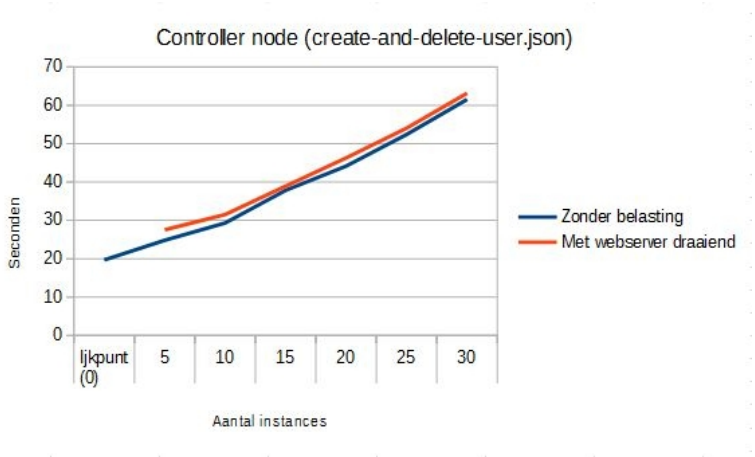
De twee uitgevoerde scenario's om te zien of Ceilometer het verbruik goed zou meten waren goed uitgevoerd en er zijn tijdens de test geen fouten opgekomen. De test is daarom succesvol verlopen.
De conclusie bij deze test is dat het verbruik die OpenStack gebruikers genereren gemeten en gelogd kan worden met Ceilometer op een opstelling op basis van OpenStack Liberty.

7.11 Performance groei

In deze test was het van belang dat het punt waarop de aansturende nodes 200% slowdown geven bekend werd.

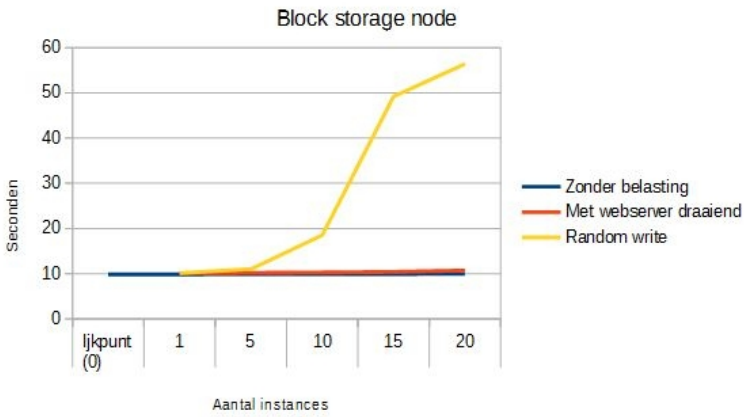
7.11.1 Controller node

Het omslagpunt van 200% slowdown werd bij de controller node gevonden door realistische load te genereren op de compute nodes met AB en MySQL-writes. Eerst werd het ijkpunt/nulpunt vastgesteld. Het nulpunt werd vastgesteld door OpenStack Rally vanaf de utility node users aan te laten maken en direct weer te laten verwijderen. Hier zou een relatief stabiele meting uitkomen. Als ijkpunt werd de “load duration” van de create-and-delete-users-benchmark genomen.
Hierna werden er elke keer instances geboot in stappen van vijf instances. Elke keer als er vijf nieuwe instances aan waren gezet is er opnieuw getest met het meetpunt. Daarna werd de realistische load gegenereerd op de compute nodes en opnieuw gemeten.



Afbeelding 34: De uitkomst van de performance/capacity test van de controller node.

Op afbeelding 34 is de uitslag van de controller node test te zien: Het nulpunt was 19.6728 seconde. Op het punt dat het meetpunt 39.3456 aan zou geven was het omslagpunt bereikt, in de grafiek betekend dit het snijpunt als de rode lijn op 39.3456 seconden staat: dit gebeurde op het punt dat 16 instances load aan het genereren waren.
De conclusie bij deze test is dat het maximaal aantal instances die de controller node aankan voordat hij 200% vertraagd 16 instances bedraagt.



Afbeelding 35: De uitkomst van de performance/capacity test van de block storage node.

7.11.2 Block storage node

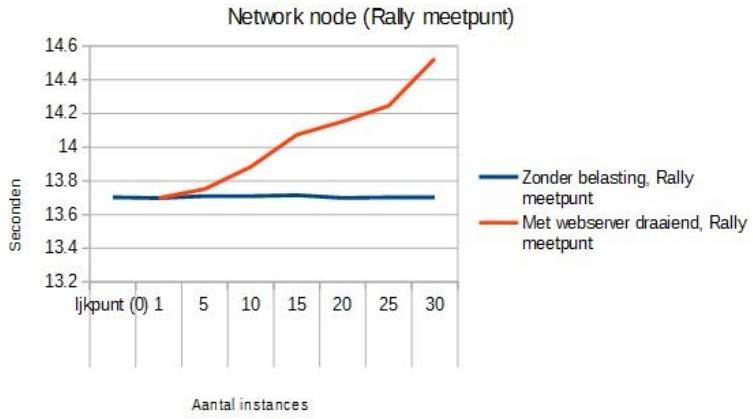
Bij de test op de block storage node werd ook een realistische load gegenereerd. Deze load werd gemaakt door MySQL database writes te laten doen, net als in een webserver. Het meetpunt werd gecreëerd door de tijd te meten die het kost om een bestand op de Cinder node te vullen met 1000MB random data. Als realistische load werd er weer door Apache2 en MySQL load gegenereerd. Tevens is er als vergelijking ook maximale load uitgevoerd met alle instances, dit is gedaan door middel van het schrijven van random data.

Op afbeelding 35 is de uitslag van de block storage node test te zien: het nulpunt bij deze test was 9.859 seconde. Dit betekend dat als de tijdsduur in de test met de webserver-load boven de 19.718 seconden komt dit het maximale aantal instances voor deze node betreft.
In deze test telt de gele lijn (Random write) echter niet mee aangezien deze alleen ter vergelijking, dit is namelijk maximale belasting van de node. Er wordt dus naar de rode lijn gekeken in de test.
Tevens geldt er in deze test dat 20 instances het maximale aantal instances betreft, dit is de limiet aan de interne storage in de block storage node.
Er wordt gekeken naar het punt dat de rode lijn over de 19.718 seconden heen gaat, dit gebeurd echter niet. Om deze reden wordt het maximaal aantal instances in de block storage node aangehouden: dit is 20 instances.
De conclusie bij deze test is dat afhankelijk van de opslagruimte die de block storage node bevat en de instances toebedeeld krijgen het omslagpunt wordt bepaald, de performance die de Cinder node moet bieden vormt geen bottleneck bij normaal (niet constant volledige load) gebruik en bij gebruik van 250GB HDD's.

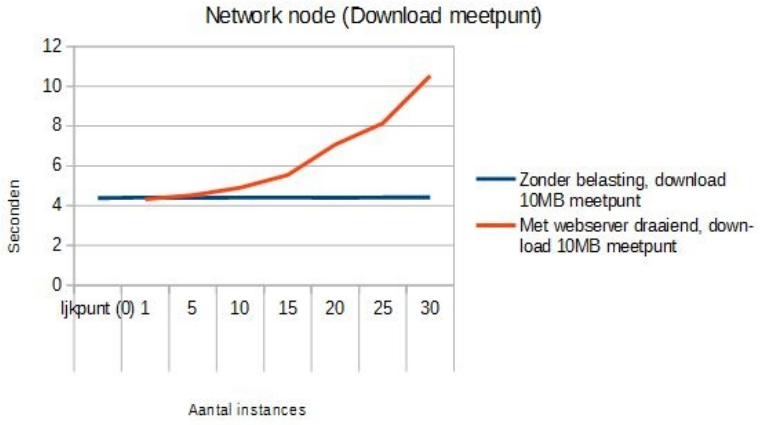
7.11.3 Network node

Bij deze test werd ook een realistische load gegenereerd door middel van AB en MySQL writes. De meetpunten werden gemaakt door het uitvoeren van de “*create-and-delete-networks.json*” benchmark met OpenStack Rally en het downloaden van een bestand van 10MB van de mirror *http://mirror.nl.leaseweb.net/speedtest/10mb.bin*.

Op afbeelding 36 en afbeelding 37 is te zien wat de uitslag van de network node tests is. Bij deze tests is het van belang dat de test die als eerste een 200% slowdown laat zien het maximale aantal instances voor de network node aangeeft.
Het nulpunt bij deze test was voor het Rally meetpunt 13.703 seconden. Op het punt dat het meetpunt 27.406 seconde zou aangeven was dit het maxmaal aantal instances voor de network node. In deze test komt het meetpunt niet bij de 27.406 seconden, er wordt dus naar de volgende test (bandwidth) gekeken.
Het nulpunt voor het download meetpunt was 4.377 seconden. Op het punt dat het meetpunt 8.754 seconden zou aangeven was dat het maximaal aantal instances voor de network node. Het snijpunt van 8.754 seconden wordt bereikt bij 26 instances.
De conclusie bij deze test is dat de belasting van de network node zelf niet snel een bottleneck zal vormen. De bottleneck die gevormd wordt in de network node is de datadoorvoer over zijn netwerkkaart.



Afbeelding 36: De uitkomst van de performance/capacity test van de network node (meetpunt Rally).



Afbeelding 37: De uitkomst van de performance/capacity test van de network node (meetpunt download).

7.12 Management netwerk security

De security van de DMZ-nodes is in deze test gecontroleerd op mogelijkheden om in het interne netwerk te komen. Eerst zijn er vier soorten portscans uitgevoerd op deze vier nodes om te zien of er nog open ports waren waar een service op aan het luisteren was. Er werd verwacht van de nodes dat er een aantal ports open waren omdat deze nodig zijn voor de draaiende service, deze ports waren: controller node port 443, SSH-jump node port 50385, NTP-node port 123. De repository node moet helemaal geen open port terug geven.
In afbeeldingen 38 t/m 41 is te zien dat de out van deze portscans precies overeen komt met de voorspelling.

Hierna is gecontroleerd of er ingelogd kan worden naar de SSH-jump node zonder private key, dit zou inhouden dat de beveiliging veel minder goed is. Met Putty is geprobeerd om in te loggen op de SSH-jump node door gebruik te maken van de username en password methode. De pogingen om op deze manier in te loggen als “user1997” en “root” hadden allebei dezelfde uitwerking: Putty gaf aan dat deze methode niet ondersteund werd. Op afbeelding 42 is te zien dat Putty de inlogpoging door middel van inlognaam en password blokkeert.

Op het moment dat er ingelogd werd naar de SSH-jump node via SSH met de juiste key vroeg Putty ook om de passphrase die bij deze key zou horen. Dit betekend dat dit onderdeel van de test positief is.

Afbeelding 42: Het inloggen op de SSH-jump node door middel van inlognaam en password wordt geblokkeerd.

Als laatste is de controller node gecontroleerd door verbinding te maken via HTTP. Het dashboard mag deze HTTP-verbinding niet accepteren omdat er alleen verbinding gemaakt mag worden via HTTPS. De HTTP-verbinding gaf een error dat er geen gebruik gemaakt mocht worden van HTTP.
Tenslotte is er ook gecontroleerd of er ingelogd kon worden via HTTPS: deze verbinding werd door dashboard geaccepteerd.

De uitkomsten van de tests waren allemaal zoals ze verwacht werden en beschreven werden in het testplan. De test is positief.
De conclusie bij deze test is dat alle toegangspoorten naar het managementnetwerk zodanig beveiligd zijn dat binnenkomen in het interne netwerk voor eventuele aanvallers een heel stuk moeilijker is geworden.

```
Host is up (0.000024s latency).
Not shown: 65534 closed ports
PORT      STATE SERVICE
443/tcp   open  https
```
Afbeelding 38: De output van de TCP SYN portscan op de controller node.

```
Host is up (0.00020s latency).
Not shown: 65534 closed ports
PORT      STATE SERVICE
50385/tcp open  unknown
```
Afbeelding 40: De output van de TCP SYN portscan op de SSH-jump node.

```
Host is up (0.00014s latency).
All 65535 scanned ports on hosted-by.i3d.net ([redacted]) are closed
MAC Address: 78:E3:B5:FD:80:17 (Hewlett-Packard Company)

Nmap done: 1 IP address (1 host up) scanned in 38.42 seconds
```
Afbeelding 39: De output van de TCP SYN portscan op de repository node.

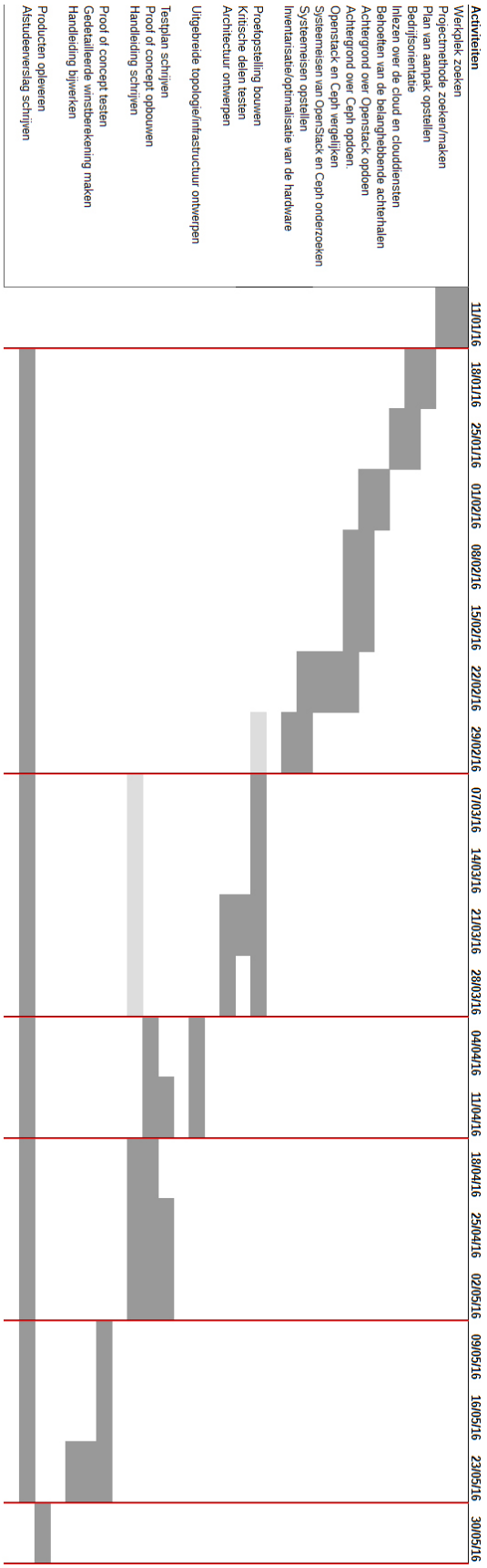
```
Not shown: 65534 closed ports
PORT      STATE SERVICE
123/tcp   open  ntp

Nmap done: 1 IP address (1 host up) scanned in 8.02 seconds
```
Afbeelding 41: De output van de TCP SYN portscan op de SSH-jump node.

Bijlage XIII: Planning

i3D.NET PERFORMANCE HOSTING

DE HAAGSE HOGESCHOOL

Globale planning:



Afwijkingsplan: