

|  |  |
| --- | --- |
| **Auteur:**  Naam: Hakan Uzun  Studentnummer: 1573838  **Bedrijf:**  MaxServ B.V.  **Opleiding:**  Systeembeheer Duaal 2013 - 2014 | **Onderwerp:**  Afstudeerscriptie (TCICT-AFSTUD-12)  **Datum:**  Breda, 29 februari 2014  **Versie:**  1.0 |



Server Deployment en Beheer Automatisering

Afstudeerscriptie

Afstudeerscriptie

Server Deployment en Beheer Automatisering

**Auteur:**

|  |  |
| --- | --- |
| Naam: | Hakan Uzun |
| Studentnummer: | 1573838 |
| Datum: | 29 februari 2014 |

**Opleiding:**

|  |  |
| --- | --- |
| Onderwijsinstelling: | Hogeschool Utrecht - Centrum voor Natuur en Techniek |
| Opleidingsrichting: | HBO Systeembeheer |
| Adres: | Nijenoord 1  3552 AS Utrecht |
| Telefoon: | 088 481 82 83 |

**Organisatie:**

|  |  |
| --- | --- |
| Naam Bedrijf: | MaxServ B.V. |
| Adres: | Productiestraat 4  4283 JM Breda |
| Telefoon: | 076-711 52 22 |

# 1 - Algemeen

## 1.1 versiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Datum | Status | Opmerkingen |
| 0.1 | 28-02-2014 | Concept | Eerste opzet scriptie |
| 0.5 | 29-02-2014 | Concept | Document is verder uitgewerkt |
| 0.6 | 15-05-2014 | Concept | Alle opmerkingen van de begeleider zijn erin verwerkt |
| 0.7 | 19-05-2014 | Concept | Opzet functioneel en technisch ontwerp gemaakt. |
| 0.8 | 22-05-2014 | Concept | Conclusie en aanbevelingen en Management samenvatting verwerkt. |
| 0.9 | 23-05-2014 | Concept Final | Bronnen bijgewerkt en inleiding goed gezet. |
| 0.9.1 | 25-05-2014 | Concept Final | Aanpassingen en aanvullingen gedaan die zijn teruggekomen. |
| 1.0 | 29-05-2014 | Final | Spellingscontrole gedaan en onderdelen geredigeerd |

## 1.2 Distributiebeheer

|  |  |  |  |
| --- | --- | --- | --- |
| Versie | Datum | Ontvanger | Opmerkingen |
| 0.5 | 10-05-2014 | Begeleider, L. van Raaij | Status bekijken |
| 0.6 | 15-05-2014 | Hakan Uzun | Verwerken van opmerkingen Begeleider |
| 0.9 | 23-05-2014 | Alf Moens | Opsturen concept volledige scriptie voor opmerkingen en eventuele aanvullingen |
| 0.9.1 | 25-05-2014 | Hakan Uzun | Verwerken opmerkingen Docent Begeleider. |
| 1.0 | 29-05-2014 | Alf Moens | Opsturen Final |

# Voorwoord

Deze scriptie is geschreven door Hakan Uzun in het kader van het afstudeerproject in opdracht van Hogeschool Utrecht. Ik heb de opleiding HBO Systeembeheer gevolgd aan de faculteit Natuur en Techniek in Utrecht.

Met veel plezier en inzet heb ik aan deze scriptie gewerkt. Deze opdracht is tot stand gekomen nadat mijn eerste opdracht was afgekeurd omdat het een BI-tintje (Bedrijfskundige Informatica) had. Deze opdracht heb ik te danken aan mijn bedrijfsbegeleider Luuk van Raaij; hij heeft deze opdracht aan mij voorgesteld en het leek mij een leuk en vooral een innovatief onderwerp voor mij en MaxServ. De samenwerking met Luuk van Raaij is prettig verlopen.

Verder gaat mijn dank uit naar Alf Moens voor de begeleiding gedurende de afstudeerperiode. En natuurlijk kan ik mijn collega Remco Overdijk niet vergeten, omdat ik in de afgelopen periode veel van hem heb geleerd. Daarnaast bedank ik ook mijn familie en vrienden die ik in de afgelopen maanden bezig heb gehouden met wanneer ik klaar zou zijn met afstuderen.

Hakan Uzun

Breda, 23-05-2014

# Managementsamenvatting

De afdeling systeembeheer/engineering van MaxServ B.V. is verantwoordelijk voor het technische beheer van de ICT-Infrastructuur. Hieronder valt ook het installeren van development servers waarop de diensten en services die MaxServ B.V. levert worden ontwikkeld. Zoals in elk IT-bedrijf is zorgvuldige documentatie en standaardisatie van servers belangrijk. Dit voorkomt namelijk dat er kennis over de servers verloren gaat. Alle servers hebben een standaard installatie met modules die het TYPO3-systeem omschrijft. De praktijk wijst echter uit dat er vooral veel configuratieverschillen in de servers zitten die niet voldoen aan de standaarden.

**Uit het bovenstaande probleem is de volgende opdracht ontstaan:**  
“Ontwerp, documenteer en implementeer een server deploymentomgeving waarmee de systeembeheerafdeling servers kan voorzien van een standaard configuratie. Naast de inrichting dient er ook bepaald te worden hoe de oplossing in beheer wordt genomen en welke stappen er nodig zijn om de oplossing te implementeren.”

Om de bovenstaande vraag te beantwoorden, is het SDBA-project opgezet. Server Deployment en Beheer Automatisering oftewel SDBA, is toegepast om het serverbeheerproces efficiënter te maken voor de systeembeheerders. Om dit proces te vergemakkelijken wordt het basisidee achter SDBA gebruikt: het traditionele handwerk zoveel mogelijk automatiseren.

Om deze automatisering te kunnen realiseren is een pakketselectie doorlopen waar het pakket Puppet uit is gekomen. De functie van Puppet binnen de organisatie van Maxserv wordt het automatisch installeren en beheren van de servers. Om tot het eindresultaat te komen is er tijdens de ontwerpfase rekening gehouden met de opdrachtgever en is er een PoC (Proof of Concept) uitgevoerd. Daarnaast zijn er nog bash scripts ontwikkeld die de systeembeheerder helpen met de initiële installatie van de server.

**Resultaat**

Tijdens de PoC is een positief resultaat naar voren gekomen. Het kost een systeembeheerder nu met behulp van Puppet in plaats van vier uur, in combinatie met finetunen, maar tien tot vijftien minuten om een dergelijke server op te leveren. Dat wil zeggen dat waar we in vier uur voorheen één server opleverden, we nu bijna zestien servers kunnen opleveren en ervoor zorgen dat de developers kunnen beginnen aan ontwikkeling. Zoals hierboven wordt genoemd, is niet alleen de tijd verbeterd. Security is ook naar een hoger niveau getild. Developers hebben alleen specifieke toegang op plaatsen/services binnen de server. Ook worden de firewallregels centraal geregeld en wordt alles geblokkeerd zodat alleen IP-adressen worden toegelaten die gespecificeerd zijn in de node configuratie of in de core configuratie.

Daarnaast heeft het product alle eisen die MaxServ heeft gesteld gedekt en gaat de afdeling systeembeheer het systeem verder intern doorontwikkelen.

Inhoudsopgave

[1 - Algemeen 2](#_Toc389163984)

[1.1 versiebeheer 2](#_Toc389163985)

[1.2 Distributiebeheer 2](#_Toc389163986)

[Voorwoord 3](#_Toc389163987)

[Managementsamenvatting 4](#_Toc389163988)

[2 - Inleiding 7](#_Toc389163989)

[3 - Projectcontext 8](#_Toc389163990)

[4 - Probleemstelling en opdrachtomschrijving 9](#_Toc389163991)

[4.1 - Huidige situatie en Probleemstelling 9](#_Toc389163992)

[4.2 - De opdracht 9](#_Toc389163993)

[4.3 - Randvoorwaarden 10](#_Toc389163994)

[4.5 - Op te leveren producten 10](#_Toc389163995)

[4.6 - Aanpak 11](#_Toc389163996)

[5 - Server Deployment en Beheer Automatisering (SDBA) 12](#_Toc389163997)

[5.1 - Wat is SDBA? 12](#_Toc389163998)

[5.2 - Waarom SDBA? 13](#_Toc389163999)

[5.3 - Hoe werkt SDBA? 13](#_Toc389164000)

[6 - Onderzoekstechniek en Pakketselectiemethode 16](#_Toc389164001)

[6.1 - Berenschot-methode 16](#_Toc389164002)

[6.2 - MoSCoW 17](#_Toc389164003)

[7 - Pakketselectie 17](#_Toc389164004)

[7.1 - Vooronderzoek 17](#_Toc389164005)

[7.2 - Weging eisen en wensen 18](#_Toc389164006)

[7.3 - Longlist 19](#_Toc389164007)

[7.4 - Shortlist 20](#_Toc389164008)

[7.5 - Pakketkeuze 21](#_Toc389164009)

[7.5.1 CFEngine 22](#_Toc389164010)

[7.5.2 Puppet 23](#_Toc389164011)

[7.5.3 Chef 23](#_Toc389164012)

[7.6 Conclusie pakketselectie 24](#_Toc389164013)

[8 - Ontwerp 25](#_Toc389164014)

[8.1 - Functioneel ontwerp 25](#_Toc389164015)

[8.1.2 - Analyse van eisen uit de organisatie 25](#_Toc389164016)

[8.1.3 - Gebruikers- en informatie-eisen 26](#_Toc389164017)

[8.1.4 - Eisen aan Puppetsoftware 27](#_Toc389164018)

[8.1.5 - Eisen aan de workflow en dataflow van SDBA 27](#_Toc389164019)

[8.1.6 - Analyse transitiefase 29](#_Toc389164020)

[8.2 - Technisch ontwerp 30](#_Toc389164021)

[8.2.1 - Algemene informatie 30](#_Toc389164022)

[8.2.3 - Serverrollen 30](#_Toc389164023)

[8.2.4 - Ontwerp 31](#_Toc389164024)

[8.2.4.1 - Software 31](#_Toc389164025)

[8.2.4.1.1 - Role: Core 32](#_Toc389164026)

[8.2.4.1.2 - Role: Webserver 33](#_Toc389164027)

[8.2.5 - Hardware 34](#_Toc389164028)

[8.2.6 - Uptime & Monitoring 35](#_Toc389164029)

[8.2.6.1 - Uptime 35](#_Toc389164030)

[8.2.6.2 - Monitoring 35](#_Toc389164031)

[8.2.6 - Ontwerp inrichting server 36](#_Toc389164032)

[8.2.7 - Installatie op Puppetcliënt en aanmelding op master 38](#_Toc389164033)

[8.3 - Implementatieplan en beschrijving 39](#_Toc389164034)

[8.3.1 - Beschrijving 39](#_Toc389164035)

[8.3.1.1 - Controleren huidige omgeving 39](#_Toc389164036)

[8.3.1.2 - Ontwerpeisen verzamelen 39](#_Toc389164037)

[8.3.1.3 - Bouwen van de Puppet Server 39](#_Toc389164038)

[8.3.1.4 - Testen van de server 40](#_Toc389164039)

[8.3.1.5 - Testen van profiles 40](#_Toc389164040)

[9 - Proof of Concept 41](#_Toc389164041)

[9.1 - Testplan 41](#_Toc389164042)

[9.1.1 - Installatie Puppetserver 41](#_Toc389164043)

[9.1.2 - Deployen server + Aanmelden 41](#_Toc389164044)

[9.1.3 - Deployen Core role 41](#_Toc389164045)

[9.1.4 - Deployen Webserver role 42](#_Toc389164046)

[9.2 - Testresultaten 43](#_Toc389164047)

[9.2.1 - Installatie Puppetserver 43](#_Toc389164048)

[9.2.1.1 - Testresultaat - installatie Puppet Master 43](#_Toc389164049)

[9.2.2 - Deployen server + Aanmelden 44](#_Toc389164050)

[9.2.3 - Deployen Core role 45](#_Toc389164051)

[9.2.4 - Deployen Webserver role 46](#_Toc389164052)

[9.3 - Conclusie PoC 47](#_Toc389164053)

[10 - Conclusie en aanbevelingen 48](#_Toc389164054)

[10.1 - Conclusie 48](#_Toc389164055)

[10.2 - Aanbevelingen 49](#_Toc389164056)

[11 - Evaluatie 49](#_Toc389164057)

[12 - Afkortingen 50](#_Toc389164058)

[13 - Literatuurlijst 50](#_Toc389164059)

[14 - Overzicht gebruikte afbeeldingen en tabellen 51](#_Toc389164060)

[Bijlage 1 - Ubuntu Aanmeldscript 52](#_Toc389164061)

[Bijlage 2 - Redhat Family Aanmeldscript 53](#_Toc389164062)

[Bijlage 3 - core.pp 54](#_Toc389164063)

[Bijlage 4 - Plan van Aanpak 56](#_Toc389164064)

# 2 - Inleiding

Efficiëntie is tegenwoordig een veel gebruikte term. Ook binnen MaxServ op de afdeling systeembeheer is het een hedendaags gebruikte term. Efficiëntie neemt vele voordelen met zich mee, bijvoorbeeld tijdsbesparing en vooral geldbesparing.

Efficiëntie komt niet uit zichzelf, daarom moet een bedrijfsproces geoptimaliseerd of geautomatiseerd worden. MaxServ kampte met verschillende serverconfiguraties, wat niet efficiënt was voor de afdeling systeembeheer. Dit was de aanleiding voor dit document, genaamd “Server Deployment en Beheer Automatisering” (verder SDBA).

Het project vindt plaats binnen MaxServ B.V. op de afdeling systeembeheer.

Hieronder wordt de opbouw van deze scriptie beschreven.

Als eerste beschrijft hoofdstuk drie de projectcontext. Hoofdstuk vier beschrijft de probleemstelling en de opdrachtomschrijving die naar voren zijn gekomen tijdens het observeren van het proces en het voeren van gesprekken met systeembeheerders. Hoofdstuk vijf beschrijft waarom het SDBA-project is opgestart. Vervolgens beschrijft hoofdstuk zes welke onderzoekstechnieken er zijn gebruikt en de pakketselectiemethode die is doorlopen. Daarna wordt de daadwerkelijke pakketselectie uitgevoerd in hoofdstuk zeven en wordt er in hoofdstuk acht het ontwerp beschreven. Hier kunnen we denken aan een functioneel ontwerp en een technisch ontwerp met daarbij een Implementatieplan.   
Binnen hoofdstuk negen wordt de PoC uitgevoerd die bestaat uit het testplan en de testresultaten die naar voren zijn gekomen. Hierbij kunnen we meteen kijken of de oplossing daadwerkelijk succes heeft opgeleverd. Dit wordt in hoofdstuk tien beschreven.

# 3 - Projectcontext

MaxServ B.V. is in het jaar 2007 opgericht met diverse diensten en services die op abonnementsbasis worden vermarkt. De dienstverlening bestaat onder andere uit webhosting, domeinregistraties, diverse SaaS-oplossingen (Software as a Service) en webonderhoudscontracten.

MaxServ B.V. is vooral gericht op de ontwikkeling van TYPO3 -sites. TYPO3 is een Enterprise graded CMS-systeem voor middelgrote en grote websites. TYPO3 kan door gebruik van plug-ins en op maat gemaakte elementen volledig naar wens worden uitgebreid.

Een aantal grote organisaties die gebruikmaken van TYPO3 zijn CDA, EO, CNV en diverse universiteiten (zoals Eindhoven, Delft en Erasmus).

Gezien de stijgende vraag naar ontwikkelservers op de afdeling development en het stijgende aantal projecten, blijkt informatie en standaardisatie noodzakelijk. Door deze toename is het voor systeembeheer een inefficiënte en tijdrovende bezigheid geworden om per project extra servers te installeren en te beheren.

Het SDBA-project richt zich op het efficiënter maken van beheer- en installatietijd. Hoe het systeem ingericht gaat worden en de efficiëntiegraad verhoogd gaat worden, hangt van veel factoren af: de nieuwe servers, de bestaande servers, de updates die op servers doorgevoerd moeten worden, de configuraties van alle servers, enzovoort.

Er is op dit moment binnen de organisatie geen eenvoudige manier om al deze aspecten te adresseren voor tot een grote efficiëntieverbetering. Het SDBA-project moet leiden tot een verbetering van de efficiëntie in dit proces.

# 4 - Probleemstelling en opdrachtomschrijving

In dit hoofdstuk wordt de huidige situatie beschreven met daarbij de opdracht en de op te leveren producten met de aanpak.

## 4.1 - Huidige situatie en Probleemstelling

De afdeling systeembeheer/engineering van MaxServ B.V. is verantwoordelijk voor het technische beheer van de ICT-infrastructuur. Hieronder valt ook het installeren van development servers waarop de diensten en services die MaxServ B.V. levert worden ontwikkeld.

Zoals in elk IT-bedrijf is zorgvuldige documentatie en standaardisatie van servers belangrijk, want dit voorkomt dat er kennis over de servers verloren gaat. Alle servers hebben een standaard installatie met modules die het TYPO3-systeem omschrijft. De praktijk wijst echter uit dat er vooral veel configuratieverschillen, vanwege de handmatige configuraties, in de servers zitten die niet voldoen aan de standaarden. Dit betekent dat zowel het documenteren als beheren inefficiënt wordt, wat leidt tot ongewenste gevolgen zoals:

1. Ontwikkelaars krijgen problemen met servers en raken tijd kwijt omdat ze “constant” aan systeembeheer moeten vragen om bepaalde modules te installeren of te bewerken, waardoor ze zich niet kunnen richten op programmeren en ontwikkelen.
2. Systeembeheerders raken tijd kwijt vanwege het onderzoeken van verschillende serverconfiguraties tijdens het updaten/installeren van deze systemen.
3. Er treden conflicten op vanwege versieverschillen binnen de servers en het niet gebruikmaken van een standaard voor gedefinieerde configuratiesets.
4. Vertraging door toename van het aantal incidenten welke voortkomen uit punten één en twee.

De ICT-infrastructuur levert ter ondersteuning van de dagelijkse werkzaamheden diensten aan de afdeling development. Elke server die niet voldoet aan de standaard vormt een serieuze bedreiging voor veiligheid en betrouwbaarheid van het productie/developmentproces binnen een project.

## 4.2 - De opdracht

Het doel van het project is om een advies uit te brengen aan de opdrachtgever op basis van de volgende opdrachtomschrijving:

|  |
| --- |
| “Ontwerp, documenteer en implementeer een server deployment omgeving waarmee de systeembeheerafdeling servers kan voorzien van een standaard configuratie. Naast de inrichting dient ook bepaald te worden hoe de oplossing in beheer wordt genomen en welke stappen er nodig zijn om de oplossing te implementeren.” |

**Deelvragen:**

1. Welke deployment software is geschikt om Linux servers volledig te deployen?
2. Wordt het open source of commercial software?
3. Welke deployment software bevat de meeste configuratiemogelijkheden?
4. Welke deployment software is het makkelijkste te beheren?
5. Hoe kunnen systeembeheerders de gekozen software beheren?
6. Hoe kunnen de systeembeheerders de gedeployde servers het beste documenteren?

## 4.3 - Randvoorwaarden

MaxServ heeft de volgende randvoorwaarden gesteld:

* het moet de beheertaken verminderen of vergemakkelijken;
* het moet webservers in een standaard kunnen configureren;
* het moet de mogelijkheid bieden alle servers vanuit een centrale machine te beheren;
* het moet patches kunnen uitrollen;
* het moet de firewall kunnen configureren;
* het moet gebruikersaccounts kunnen managen.

## 4.5 - Op te leveren producten

Het onderzoek moet de volgende resultaten opleveren:

* Ontwerpadvies:
  + Functioneel Ontwerp
    - In het functioneel ontwerp wordt beschreven wat het systeem aan functionaliteiten moet bieden.
    - Een beschrijving van de functionele eisen, wensen en randvoorwaarden voor het deployment systeem.
  + Pakketselectie
    - De beschrijving van de selectiecriteria (kwaliteits- en leverancierscriteria en functionele en technische criteria) waaraan het deploymentpakket moet voldoen;
    - Een overzicht van de pakketten en selectiecriteria in een selectiematrix;
    - Een beschrijving van het geselecteerde pakket uit de pakketselectie.
  + Technisch ontwerp
    - Het technisch ontwerp beschrijft hoe de eisen en wensen uit het functionele ontwerp technisch gerealiseerd zullen worden.
  + Implementatieplan
    - Advies over de stappen die doorlopen moeten worden om een succesvolle invoering van de nieuwe processen en procedures.
* Proof of Concept **(PoC)**
  + Een beschrijving van de geteste functionaliteiten, uitgewerkt in een testplan;
  + Een beschrijving van de uiteindelijke testresultaten.

## 4.6 - Aanpak

Het project is in verschillende fasen verdeeld, zodat het project beheersbaar blijft.

**Fase 1 Onderzoek:**

1. Onderzoek theorie en modellen;
2. Interviews afnemen met betrokkenen;
3. De huidige situatie inventariseren met de bijbehorende problematiek;
4. Beschrijven probleemanalyse;
5. Alle eisen en wensen inventariseren voor het te ontwerpen systeem.

**Fase 2 Ontwerp:**

1. Selectie van software (pakketselectie);
2. Opstellen functioneel ontwerp (FO);
3. Opstellen technisch ontwerp (TO);
4. Inrichten van het systeem (testopstelling);
5. Opstellen van een implementatieplan.

**Fase 3 Proof of Concept:**

1. Opstellen van het testplan;
2. Testen van een deployment en toepasbaarheid van het systeem;
3. Opstellen van de testresultaten.

**Fase 4 Presentatie en verdediging:**

Het presenteren en verdedigen van de onderzoeksresultaten.

**Fase 5 Afsluiting:**

Afsluiting van het gehele project.

# 5 - Server Deployment en Beheer Automatisering (SDBA)

In dit hoofdstuk wordt het project SDBA beschreven en wordt uitgelegd wat het project houdt.

## 5.1 - Wat is SDBA?

Server Deployment en Beheer Automatisering, oftewel SDBA, heeft als doel om het serverbeheersproces efficiënter te maken voor de systeembeheerders. Omdat er verschillende besturingssystemen aanwezig zijn op het platform, wat ook wel eens een heterogene serveromgeving wordt genoemd, is het installeren/beheren van die omgeving geen gemakkelijk proces.

Het doel van SDBA is om het traditionele handwerk zoveel mogelijk te automatiseren en dit proces voor de systeembeheerders van MaxServ makkelijker te maken. Het traditionele handwerk kost veel tijd en versiebeheer van configuratiebestanden is moeilijk bij te houden. Daarnaast wordt het uitrollen van elke server anders uitgevoerd, wat een niet standaard configuratie als resultaat heeft.

De initiële installaties zullen altijd door systeembeheerders uitgevoerd worden. Denk hierbij aan het aanmaken van nieuwe servers waarop alleen basisinstallaties neergezet worden en die vervolgens op een centrale server worden aangemeld. Nadat de initiële installaties zijn uitgevoerd, is het de bedoeling dat de Nodes (servers) die met het softwarepakket van de SDBA gekoppeld worden en altijd een standaard configuratie ophalen welke gedownload worden vanuit een centrale server. Zodra de nodes de configuraties vanuit de centrale server hebben ontvangen, weten de nodes (servers) wat voor rol ze moeten vervullen. Vervolgens voeren de nodes de gehele configuratie uit en worden de de modules die zij nodig hebben automatisch geïnstalleerd. Hierdoor zal alles altijd aan de hand van een gedefinieerde standaard worden geconfigureerd.

Nadat de servers zijn uitgerold en geïnstalleerd, moet SDBA ook de beheertaken van systeembeheerders overnemen. Denk hierbij aan het automatisch softwarematig updaten van de servers, het updaten van firewall rules, het toevoegen van virtual hosts binnen Apache, enzovoorts.

De initiële deploy van de nodes wordt via een bestaand controlepaneel gedaan. Of dit handmatig moet blijven gebeuren of geautomatiseerd gaat worden, zal in een later stadium van het project blijken.

**Hoofdtaken die het SDBA-project zou kunnen overnemen:**

* configuratiemanagement
  + packagemanagement
    - Bijvoorbeeld: geautomatiseerde installatie van webservers.
  + Gebruikersaccountmanagement  
    (SSH Keys beheren die toegang hebben tot de server met verschillende accounts)
  + firewallmanagement.
* discovery en monitoring
* rapporteren.

**Het SDBA-project gaat het mogelijk maken om de beheer- en onderhoudstaken te automatiseren, waardoor de volgende doelstellingen bereikt worden:**

* het verminderen van de beheer- en onderhoudskosten;
* het verminderen van de ad hoc changes en het centraliseren van het configuratiemanagement;
* het verminderen van de beheer- en onderhoudstijd (Mean time to repair);
* het verminderen van routinewerkzaamheden;
* het centraliseren van het configuratiemanagement.

## 5.2 - Waarom SDBA?

Veel bedrijven willen hun IT-processen verbeteren en automatiseren. Het project SDBA gaat mogelijkhedenbieden om beheer- en onderhoudstaken routinematig te automatiseren. Hierdoor kan de beheer- en onderhoudstijd verminderd worden waardoor de beheerders meer tijd hebben voor andere werkzaamheden.

Hieronder worden de voordelen van SDBA genoemd:

* De veranderingen die gemaakt worden, worden geregistreerd. Dit kan op een later tijdstip gebruikt worden om bijvoorbeeld fouten op te sporen.
* De onderlinge verbanden tussen software wordt vastgelegd. Hierdoor is bijvoorbeeld makkelijker te achterhalen welke modules er aangepast moeten worden om een fout te herstellen.
* Tijdsbesparing op installatie en onderhoud.

## 5.3 - Hoe werkt SDBA?

De deployment van alle nodes wordt gedaan via OnApp. Dit is de software waarop onze Cloudomgeving draait. OnApp is te vergelijken met de software VMWare waar virtuele servers op gedraaid kunnen worden.

|  |
| --- |
| OnApp software enables Infrastructure-as-a-Service for hosts, telcos and other service providers. With OnApp in your datacenter you can use commodity hardware to sell public & private cloud services, dedicated servers, Virtual Private Servers, CDN, DNS, storage and much more, through one fully automated control panel. |

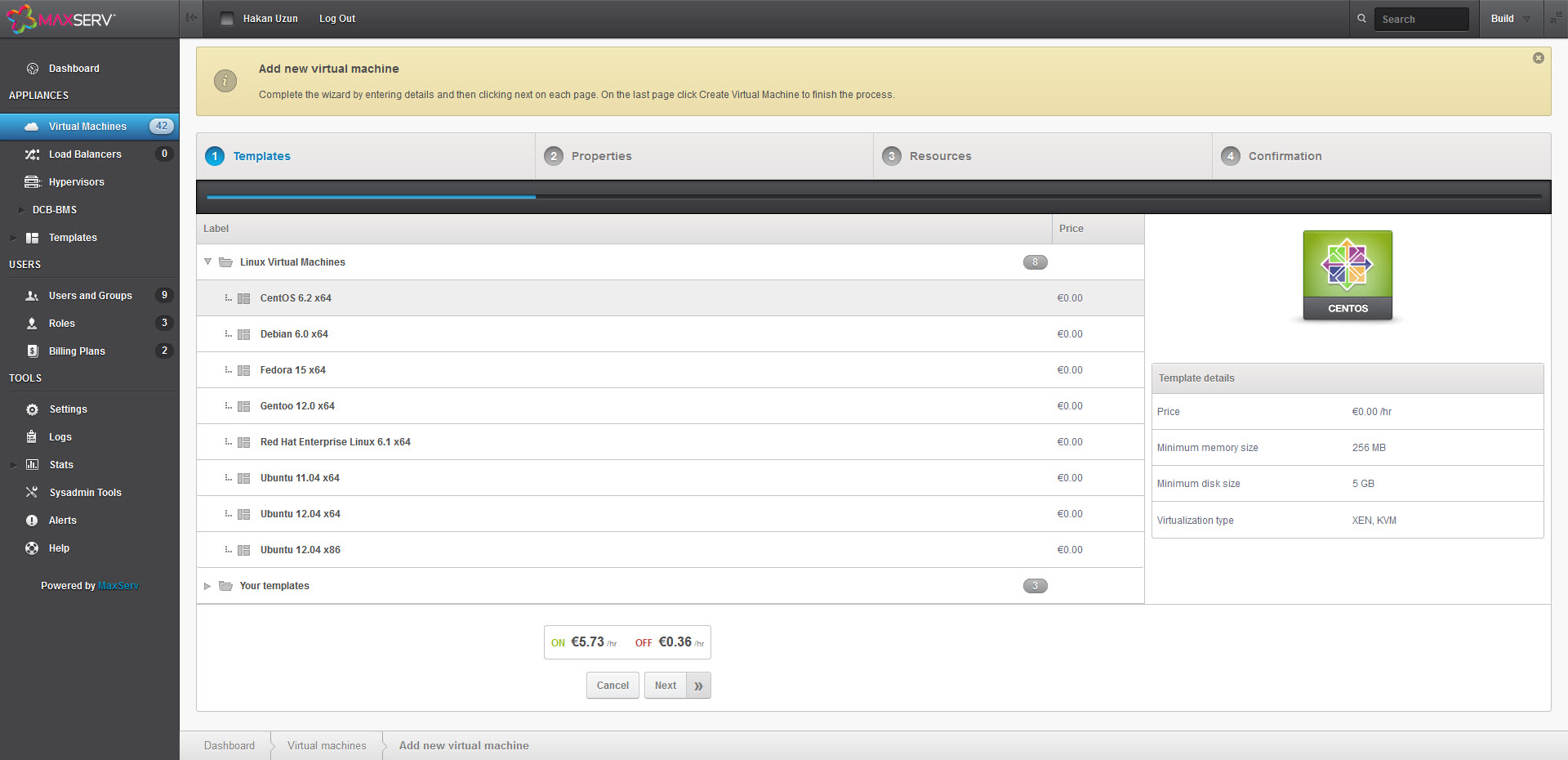
Tabel 1 - Hoe OnApp zichzelf omschrijft[[1]](#footnote-2)

De OnApp Cloud is opgebouwd uit verschillende onderdelen:

* **OnApp controller**
  + Hierop staat de web interface waarin alles geregeld kan worden. Hier kunnen bijvoorbeeld individuele machines geïnstalleerd worden en resources kunnen via de controller vergroot of verkleind worden. Daarnaast kun je via het controlepaneel consolesessies opstarten zodat je direct op de machine werkt i.p.v. via SSH.
* **Hypervisor**
  + Op deze hardware worden de virtuele machines gehost en geeft deze machines toegang tot de resources die aanwezig zijn (bijvoorbeeld geheugen, CPU, netwerktoegang).
* **Data store**
  + Hier wordt alles wat met storage te maken heeft geregeld. Dat wil zeggen dat de disk van de hypervisor hier worden opgeslagen. Dit gebeurt door middel van een SAN[[2]](#footnote-3). Ook worden van deze disks automatische backups gemaakt op een aparte backupserver.

SDBA bestaat uit twee verschillende hoofdonderdelen.

Het proces begint eerst met een server deploy en zoals eerder al aangegeven is het nog onbekend of dit een handmatig proces blijft of een geautomatiseerd proces gaat worden.



Afbeelding 1- OnApp controlepaneel manueel OS selectie

Als de server is aangemaakt, wordt er een geautomatiseerde installatie gedaan via de CM-server waardoor een gestandaardiseerde installatie wordt uitgevoerd.

CM is gebaseerd op het Client-servermodel. De communicatie tussen client en de server gaat via het http-protocol. De communicatie tussen de client en server wordt beveiligd en encrypted door een SSL-certificaat. De client krijgt het certificaat zodra hij zich registreert op de CM-server. Tijdens de registratie vraagt de client een certificaat aan op de CM-server die eerst handmatig geautoriseerd moet worden. Zodra een systeembeheerder de server heeft geautoriseerd en de authenticatie met de client(s) in orde is bevonden, stuurt de CM-server het configuratiebestand naar de client(s). Zodra deze is ontvangen, begint de cliënt met de configuratie van de server. De client(s) en de server houden beide een hash bij om configuratiechanges bij te houden.

Als er een misconfiguratie wordt gedaan, blijft de client nog draaien omdat hij automatisch het ge-cachte configuratiebestand gebruikt dat eerder van de server afgehaald/gestuurd. De gemaakte configuratiebestanden worden opgeslagen in een branch. Een branch bestaat uit verschillende configuraties die worden gebruikt door de CM-server. Op de client draait een translation agent die de gemaakte configuratiebestanden vertaalt naar het opgegeven profiel. Dit profiel kan bestaan uit verschillende Linuxdistributies zoals Ubuntu, CentOS enz. De agent die op de client(s) draait controleert of er iets geconfigureerd moet worden of niet.



Afbeelding 2 - Illustratie CM-proces

# 6 - Onderzoekstechniek en Pakketselectiemethode

Voor het onderzoek wordt gebruikgemaakt van verschillende methoden.

Voor de informatieverzameling wordt tijdens het onderzoek gebruikgemaakt van semi-gestructureerde interviews, ook wel kwalitatief onderzoek genoemd. De gespreksonderwerpen liggen vast, maar de vraagstelling niet. Hierdoor kan de interviewer doorvragen op de antwoorden van de geïnterviewde.

Vervolgens wordt er gebruikgemaakt van de Berenschot-methode[1]; deze zal gedeeltelijk worden toegepast om het pakketselectietraject op een verantwoorde wijze te doorlopen. De Berenschot-methode zorgt voor een gestructureerde aanpak en pakketkeuze, die in overeenstemming is met de eisen en wensen van de stuurgroep. Er wordt ook gebruikgemaakt van MoSCoW[1]. Dit betreft een methode om prioriteiten aan te geven. Deze methode zal toegepast worden om de eisen en wensen tijdens de selectie- en PoC-fase te prioriteren.

## 6.1 - Berenschot-methode

De Berenschot-methode[1] zorgt voor een gestructureerde aanpak en pakketkeuze. De Berenschot-methode omvat 11 stappen, waarvan de eerste vijf selectiestappen zijn en de laatste zes vervolgstappen. Omdat het huidige project niet alle stappen van de Berenschot-methode zal omvatten, is hieronder een selectie gemaakt van de vier stappen die in dit project gebruikt zullen worden.

1. Vooronderzoek
   1. Het opstellen van een lijst met criteria waaraan het pakket moet voldoen zodat pakketten die niet aan deze criteria voldoen in latere stappen afvallen.
2. Aanbod pakketten scannen (Longlist)
   1. Een algemene lijst met alle mogelijke bruikbare pakketten of leveranciers. De pakketten in de longlist worden gedetailleerd onderzocht en tegen de criteria aangehouden van stap één om tot een shortlist te komen (stap

Afbeelding 3 - Berenshot-methode

drie).

1. Voorselectie (Shortlist)
   1. De shortlist bevat alle pakketkandidaten die geselecteerd en geadviseerd kunnen worden. Het verschil tussen stap twee en stap drie is dat de pakketten die in stap twee aan de eisen/wensen voldoen, meegenomen zijn naar stap drie. Dus de lijst is een stuk korter (er zijn ondertussen dus al pakketten afgevallen). In stap drie wordt elk pakket nogmaals grondig bestudeerd op beschikbare features en functies die vervolgens met elkaar vergeleken worden. Vervolgens worden bepaalde scores uitgedeeld aan de hand van punten die worden gegeven aan de gestelde eisen. Uiteindelijk blijven er ongeveer drie pakketten over waaruit gekozen kan worden.
2. Keuze Pakket
   1. Het uiteindelijke pakket dat uit de shortlistselectie is voortgekomen.

## 6.2 - MoSCoW

MoSCoW is een Ezelsbruggetje voor een techniek die wordt gebruikt om prioriteiten aan te geven.

Hieronder is aangegeven wat de letters M, S, C en W betekenen, de kleine letters “o” hebben geen representatieve betekenis.

|  |  |  |
| --- | --- | --- |
| M | Must have | Dit zijn specificaties/onderdelen die vereist zijn. |
| S | Should Have | Een gewenste eis maar een workaround is goed |
| C | Could Have | Een eis die alleen aan bod hoeft te komen als er tijd genoeg is |
| W | Won’t Have | Een eis die nu overbodig is, maar in de toekomst interessant kan zijn. |

Tabel 2 - Betekenis MoSC0w

MoSCoW wordt toegepast tijdens de pakketselectie. De eisen en wensen worden opgedeeld in een soortgelijke tabel zoals hierboven. Vervolgens worden aan de eisen en wensen cijfers gekoppeld die worden gegradeerd door de systeembeheer afdeling van MaxServ.

# 7 - Pakketselectie

Omdat niet alle wensen en eisen even belangrijk zijn, is het nodig om een verschil in “zwaarte” van een wens of eis te maken. Om dit te doen is het nodig om een waarde van belangrijkheid aan een eis/wens toe te kennen. Dit is gedaan aan de hand van een vooronderzoek waarin vanzelf naar voren komt waar de nadruk op ligt.

## 7.1 - Vooronderzoek

In paragraaf 4.3 had MaxServ al een aantal randvoorwaarden meegegeven die hieronder nogmaals zijn weergegeven.

* Het pakket moet de beheertaken verminderen of vergemakkelijken.
* Het pakket moet webservers in één standaard kunnen configureren.
* Het pakket moet de mogelijkheid bieden alle servers vanuit een centrale machine te beheren.
* Het pakket moet patches kunnen uitrollen.
* Het pakket moet de firewall kunnen configureren.
* Het pakket moet gebruikersaccounts kunnen managen.

De gegeven randvoorwaarden waren al behoorlijk specifiek, maar om in de pakketselectie tot betere resultaten te komen, is het verstandig om deze lijst uit te breiden met eisen en wensen. Hiervoor zijn observaties gedaan en gesprekken aangegaan met systeembeheerders.

De observaties zijn gedaan tijdens de dagelijkse werkzaamheden die uitgevoerd werden. Hier is gekeken naar hoe de afdeling systeembeheer te werk ging als er bijvoorbeeld een vraag binnenkwam over het uitvoeren van een verandering op de server. Tijdens de werkzaamheden zijn gesprekken aangegaan met de uitvoerenden van de opdracht(en). Daarnaast heb ik zelf ook opdrachten uitgevoerd en gekeken waar de knelpunten zaten.

Daaruit zijn de volgende extra eisen en wensen naar voren gekomen.

* Het pakket moet Linux ondersteunen.
* Het pakket moet makkelijk uit te breiden zijn met modules en software.
* Het pakket moet makkelijk te onderhouden zijn.
* Het pakket moet goede communityondersteuning hebben.
* Het pakket moet interoperabel zijn; de mogelijkheid hebben om makkelijk met andere systemen te kunnen samen werken.

De profielbeschrijving van MaxServ geeft eigenlijk al een antwoord op de vraag die gesteld wordt. Hierin staat dat MaxServ altijd pleit voor opensource. Maar omdat er een keuze gemaakt moet worden tussen commercieel of opensource, is hierover met het management gepraat. Kort samengevat werd er als antwoord gegeven, zoals verwacht, dat “het pakket opensource [moet] zijn”.

## 7.2 - Weging eisen en wensen

Niet alle wensen en eisen zijn even belangrijk. Om een verschil in de ‘zwaarte’ van een wens of eis te maken, wordt de MoSCoW-methode uitgevoerd en is samen met de werknemers van de afdeling systeembeheer aan de Must haves en Should haves een waarde van belangrijkheid toegekend.

De werknemers van de afdeling systeembeheer hebben de onderstaande tabel ingevuld. Vervolgens is alles naast elkaar gelegd en zijn de scores ingevuld. Met deze democratische onderscheiding kunnen we later in het proces bepalen of een pakket voldoet aan de wensen van de organisatie en is er ook een gemiddelde weging per eis/wens bepaald.

|  |  |  |
| --- | --- | --- |
|  | Eis/wens | Gemiddelde weging |
| Must Have | Het pakket moet de beheertaken verminderen of vergemakkelijken. | 10 |
| Het pakket moet webservers in één standaard kunnen configureren. | 10 |
| Het pakket moet de mogelijkheid bieden alle servers vanuit een centrale machine te beheren. | 10 |
| Het pakket moet patches kunnen uitrollen. | 6 |
| Het pakket moet de firewall kunnen configureren. | 9 |
| Het pakket moet gebruikersaccounts kunnen managen. | 9 |
| Het pakket moet Linux ondersteunen | 10 |
| Het pakket moet makkelijk uit te breiden zijn met modules en software. | 9 |
| Het pakket moet makkelijk te onderhouden zijn. | 10 |
| Het pakket moet goede communityondersteuning hebben. | 6 |
| Het pakket moet opensource zijn. | 10 |
| Het pakket moet interoperabel zijn; de mogelijkheid hebben om makkelijk met andere systemen samen te kunnen werken. | 9 |
| Should Have | Alvast aanmaken van de benodigde TYPO3-mappen met de juiste rechten | 5 |
| Could Have | Automatische installatie van TYPO3 | - |
| Integrator die kiest welke modules hij nodig heeft in een dashboard zodat de gekozen software automatisch de gekozen modules plaatst/installeert | - |
| Implementatie van het pakket | - |

Tabel 3 - MoSCoW inclusief weging

De Must haves hebben de hoogste score. Deze zijn immers het belangrijkste voor systeembeheer. In Should Have zit een onderdeel dat de afdeling development had doorgegeven. Omdat dit een redelijk eenvoudige taak is en de afdeling systeembeheer daar zelf een lage cijfer aan heeft toegekend, wordt het wel genoemd maar is de “implementatie” van deze eis niet per se nodig.

## 7.3 - Longlist

In het vooronderzoek zijn eisen en wensen opgesteld waaraan de pakketten moeten voldoen.

De lijst die hieronder is opgesteld, bevat alle mogelijke en beschikbare pakketten. Bij nadere bestudering blijven er van de achttien pakketten nog maar vijf over deze zijn niet gekleurd. De reden is dat afgewezen pakketten niet meer ontwikkeld worden, bijna niet meer ontwikkeld worden of dat de specificaties nauwelijks of niet overeenkomen met de gestelde eisen.

Legenda:

|  |  |
| --- | --- |
|  | Zeer verouderde pakket of gestopt met ontwikkeling. |
|  | Voldoet niet aan de eisen. |

|  |  |  |
| --- | --- | --- |
| # | Pakket | Stable Release |
| 01 | Ansible | 2014 |
| 02 | Bcfg2 | 2013 |
| 03 | CFEngine | 2013 |
| 04 | Cdist | 2012 |
| 05 | Chef | 2014 |
| 06 | ISconf | 2006 |
| 07 | Juju (Ubuntu Only) | 2013 |
| 08 | OCS Inventory NG | 2011 |
| 09 | PIKT | 2007 |
| 10 | Puppet | 2013 |
| 11 | Quattor | 2014 |
| 12 | Radmind | 2008 |
| 13 | Rex | 2013 |
| 14 | Rundeck | 2014 |
| 15 | SmartFrog | 2009 |
| 16 | Salt | 2014 |
| 17 | Spacewalk | 2013 |
| 18 | STAF | 2012 |

Tabel 4 - Pakketselectie Longlist

De pakketten die doorgaan naar de shortlist krijgen een weging van de gestelde eisen die in [paragraaf 7.2](#_7.2_-_Weging) zijn vastgesteld.

## 7.4 - Shortlist

De shortlist heeft als doelstelling om te filteren en de pakketselectie te verkleinen voor een makkelijkere selectie.

Per pakket is er gekeken naar de specifieke eisen die de maker stelt en de handleidingen die zijn vrijgegeven. Daarbij is gekeken naar de learning curve die systeembeheerders moeten doorlopen en naar of de implementatie “makkelijk” is uit te voeren. Om tot een juiste keuze te komen, is per must have een maximaal te geven cijfer gedefinieerd, zie [paragraaf 7.2](#_7.2_-_Weging). Als er bijvoorbeeld een onderdeel wat minder dan maximaal is, kan er een 9 van de 10 punten gegeven worden of 3 van 6 punten.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Score criteria | Slecht | Middel | Goed | Beste |
| 0-44 | 45-66 | 67-88 | 89-108 |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | | | | | | Pakketten | | | | | |
|  | | | | | P1 Ansible | | | | | |  |
|  | | | | P2 Bcfg2 | | | | | | | P1 |
|  | | | P3 CFEngine | | | | | | | P2 |
|  | | P4 Chef | | | | | | | P3 |
|  | P5 Puppet | | | | | | | P4 |
| Eisen en wensen | | | | | | | P5 |
| Het pakket moet de beheertaken verminderen of vergemakkelijken. | | | | | | | 10 | 10 | 9 | 0 | 0 |
| Het pakket moet webservers in één standaard kunnen configureren. | | | | | | | 10 | 10 | 10 | 10 | 10 |
| Het pakket moet de mogelijkheid bieden alle servers vanuit een centrale machine te beheren. | | | | | | | 10 | 10 | 10 | 10 | 10 |
| Het pakket moet patches kunnen uitrollen. | | | | | | | 6 | 6 | 6 | 0 | 0 |
| Het pakket moet de firewall kunnen configureren. | | | | | | | 9 | 9 | 9 | 9 | 9 |
| Het pakket moet gebruikersaccounts kunnen managen. | | | | | | | 9 | 9 | 9 | 0 | 9 |
| Het pakket moet Linux ondersteunen. | | | | | | | 10 | 10 | 10 | 10 | 10 |
| Het pakket moet makkelijk uit te breiden zijn met modules en software. | | | | | | | 9 | 9 | 9 | 9 | 9 |
| Het pakket moet makkelijk te onderhouden zijn. | | | | | | | 10 | 10 | 10 | 0 | 0 |
| Het pakket moet goede communityondersteuning hebben. | | | | | | | 6 | 6 | 6 | 0 | 0 |
| Het pakket moet opensource zijn. | | | | | | | 10 | 10 | 10 | 10 | 10 |
| Het pakket moet interoperabel zijn; de mogelijkheid hebben om makkelijk met andere systemen samen te kunnen werken. | | | | | | | 9 | 9 | 9 | 0 | 0 |
| Totaal | | | | | | | **108** | **108** | **107** | **58** | **67** |

Tabel 5 - Pakketselectie Shortlist

De pakketten met de hoogste scores worden in de pakketkeuze genoemd. In de volgende paragraaf worden de drie pakketten die het beste scoorden beschreven.

## 7.5 - Pakketkeuze

Het is geen wonder dat de drie pakketten CFEngine, Chef en Puppet zo dicht bij elkaar zitten qua score. De drie pakketten hebben een verleden. Hieronder wordt in het kort per pakket beschreven waarom het een geschikt pakket zou zijn voor MaxServ.

Maar eerst worden de soorten configuratietalen die de pakketten gebruiken uitgelegd, omdat die in de volgende paragrafen naar voren zullen komen.

Alle pakketten die hieronder genoemd worden, gebruiken een eigen domain-specific-language, oftewel DSL, voor het configureren van de servers. Een van de grote voordelen van het gebruiken van hun eigen DSL, is dat het makkelijker te programmeren is dan een standaard programmeertaal, zoals shellscripts.

De systeembeheerders hoeven met DSL dus geen complete code te schrijven, maar alleen stukjes code te onthouden (expressions) dat ze kunnen toepassen. Het voordeel hiervan is dat het voor een systeembeheerder makkelijker is om te leren en toe te passen.

### 7.5.1 CFEngine

CFEngine is van de drie het oudste pakket . CFEngine versie 1 is ontwikkeld in 1993. Het was initieel ontwikkeld om een groep werkstations op de afdeling Theoretische Fysica binnen Oslo University in Norwegen te automatiseren. Naturlijk heeft het niet stil gestaan en zijn de ontwikkelingen omtrent CFEngine doorgegaan. In 1998 was er volgens CFEngine ontevredenheid over de mate van inzicht in het gebied. Daarom is er bij Oslo University College een nieuwe researchprogramma opgestart voor systeembeheerders en dit leidde tot een grote herschrijving van de code waardoor CFEngine 2 is ontstaan. Vervolgens is er ongeveer vijf jaar na dato weer een herschrijving gedaan waardoor CFEngine 3, de huidige versie, is ontstaan.

CFEngine is gebaseerd op het client-serverprincipe zoals bijna elk soortgelijk pakket. Het pakket maakt gebruik van een declaratieve configuratietaal die de configuratie van server vergemakkelijkt. Op elke server draait een agent die met de centrale CFEngine-master communiceert en de configuratiebestanden downloadt.

In de vorige versies van CFEngine werd op een aantal functies gericht. Deze functies waren netwerkconfiguratie (interface), de bewerking van bestanden, het maken van symlinks en permissies van bestanden. Na de re-write van het pakket is dit in CFEngine 3 uitgebreid naar een volledig configuratiemanagementpakket.

CFEnine is gebaseerd op de scriptingtaal C. C is wel een taal die minder geheugen verbruikt, maar heeft wel een hogere leercurve dan andere talen. Dat is in dit geval een minpunt voor systeembeheerders met gelimiteerde programmeerskills.

Een voorbeeld van CFEnginetaal:

|  |
| --- |
| body common control  {  bundlesequence => { "install\_packages" };  inputs => { "libraries/cfengine\_stdlib.cf" };  }  bundle agent install\_packages  {  vars:  "desired\_packages"  slist => { # list of packages we want  "apache2",  };  packages:  "$(desired\_packages)" # operate on listed packages  package\_policy => "add", # What to do with packages: install them.  package\_method => generic;  } |

### 7.5.2 Puppet

De reden dat Puppets score zo dichtbij de score van CFEngine zit, is dat de core van Puppets net zo werkt als CFEngine. Dit komt doordat het programma is gemaakt door een van de powerusers van CFEnine 2. Hij uitte zijn onvrede over CNFEngine 2 uit door in 2005 Puppet op te richten.

Puppet is ontworpen als een bibliotheek waarmee de server gebruikers naar eigen wens kunnen configureren. Puppet is volledig geschreven in Ruby, maar heeft naast die taal ook zijn eigen DSL-taal. Die taal is ontworpen om het toegankelijker te maken voor systeembeheerders met gelimiteerde programmeerkennis. Evenals CFEngine werkt Puppet met het client-serverprincipe.

Een voorbeeld van Puppettaal:

|  |
| --- |
| class apache-server {    # determine the apache-server package based on the operatingsystem fact  $apache\_server = $operatingsystem ? {  Redhat => "httpd",  default => "apache2",  }    package { "$apache\_server":  ensure => "present",  alias => "apache-server",  }    file { "/var/www/html/index.html":  source => "puppet:///index.html",  require => Package["apache-server"],  }    } |

### 7.5.3 Chef

Ook CHEF is ontstaan uit onvrede van Puppet. Voorheen heette CHEF Opscode, maar omdat Opscode een zwakke marketingnaam zou zijn, is het softwarepakket omgedoopt tot CHEF. Chef werkt net als de andere twee pakketten op basis van het client-serverprincipe.

Chef is geschreven in Ruby. Net zoals Puppet, bevat Chef configuratiebestanden die handelingen bevat die uitgevoerd moeten worden. Zoals de naam laat zien is de gedachte erachter dat de gebruikers van het pakket hun configuratie schrijven als een recept.

|  |
| --- |
| package "apache2" do  case node[:platform]  when "centos"  package\_name "httpd"  when "debian","ubuntu"  package\_name "apache2"  when "arch"  package\_name "apache"  end  action :install  end |

## 7.6 Conclusie pakketselectie

Puppet, Chef en CFengine zijn allemaal spelers op de configuratiemanagementmarkt. Natuurlijk is de keuze die gemaakt gaat worden een belangrijke keuze. Het pakket gaat een belangrijke rol spelen binnen MaxServ omdat het een installatie- en beheerproces moet overnemen die normaal systembeheerders handmatig uitvoeren. Daarnaast gaat het om een geautomatiseerd systeem dat waarschijnlijk meerdere jaren gebruikt gaat worden en dus ook meerderjarig mee moet gaan.

Na het vergelijken van de pakketten en door te kijken naar de infrastructuur waarmee het moet werken, is de keuze gevallen op Puppet. Puppet heeft een grote installatiegroep waartoe grote bedrijven behoren zoals CERN, NASA, Intel en Atlassian [6]. Omdat de gebruikersgroep zo groot is, zijn er vele bijdragers die meedoen met het ontwikkelen van Puppet ook zijn third-partysystemen ontwikkeld die samenwerken met Puppet. Dit betekent “automatisch” dat Puppet een blijver is op de markt omdat de ontwikkelingen altijd zullen doorgaan. Dit is geen beschreven eis maar een bedrijf zijnde wil immers een pakket gebruiken dat geüpdatet blijft.

Wat ook meetelt, is de keuze van configuratietalen. Chef gebruikt bijvoorbeeld een extensie van de Rubytaal. Al is Ruby een goede taal, het is niet ontwikkeld als configuratiemanager. Waarom alleen Ruby geen goede taal is, is omdat systeembeheerders met een dergelijk pakket moeten werken. Systeembeheerders hebben niet altijd kennis van programmeren, waardoor de leercurve eventuele trainingskosten omhoog brengt. Dit is niet altijd wenselijk. De keuze Puppet is ook in Ruby geschreven, maar de systeembeheerder die het gaat implementeren heeft de keuze uit twee talen, namelijk Ruby en hun eigen makkelijkere DSL. Hun eigen DSL heeft een lagere leercurve en elke systeembeheerder die minimale uitleg/training krijgt, kan ermee overweg.

**Nogmaals alle voordelen van Puppet op een rij:**

* Het is makkelijk om te leren vanwege de custom DSL-laag
  + Makkelijk leesbaar
  + Eenvoudig te leren.
* Het wordt gebruikt door grote bedrijven, dus ontwikkeling is altijd gegarandeerd.
* Het kan uitgebreid worden met modules en plug-ins.
* Het heeft een groot scala aan literatuur en handleidingen.

**Puppet voldoet aan de gestelde eisen van MaxServ B.V.:**

* Het pakket moet webservers in één standaard kunnen configureren.
* Het pakket moet de mogelijkheid bieden alle servers vanuit een centrale machine te beheren.
* Het pakket moet patches kunnen uitrollen.
* Het pakket moet de firewall kunnen configureren.
* Het pakket moet gebruikersaccounts kunnen managen.
* Het pakket moet Linux ondersteunen.
* Het pakket moet makkelijk uit te breiden zijn met modules en software.
* Het pakket moet makkelijk te onderhouden zijn.
* Het pakket moet goede communityondersteuning hebben.
* Het pakket moet opensource zijn.
* Het pakket moet interoperabel zijn; de mogelijkheid hebben om makkelijk met andere systemen samen te kunnen werken.

De pakketselectie heeft een duidelijk resultaat opgeleverd. Het pakket past goed in de organisatie en in de organisatiestructuur. Verwacht wordt dat de efficiëntie van beheer- en onderhoudstaken omhoog zal gaan en er minder tijd nodig is voor de kleine onderhouds- en beheertaken.

# 8 - Ontwerp

Het ontwerp heeft als doel een detailspecificatie te leveren van het te leveren product. Het ontwerp bestaat uit twee onderdelen, namelijk het functionele ontwerp en het technische ontwerp. Beide ontwerpen zijn specifiek gemaakt voor MaxServ. Echter, ze kunnen ook algemeen gebruikt worden als er IP-adressen veranderd worden.

## 8.1 - Functioneel ontwerp

Na de pakketselectie is voor de nieuwe software die gebruik gaat worden een functioneel ontwerp nodig. Dit is nodig om alles zo vloeiend en efficiënt mogelijk te kunnen uitrollen/implementeren. In dit ontwerp staat een blauwdruk beschreven voor het nieuwe SDBA-systeem

### 8.1.2 - Analyse van eisen uit de organisatie

Om een goed beeld te krijgen van de eisen die aan het nieuwe platform gesteld worden, is een onderverdeling in een aantal lagen gemaakt. Het zijn zes lagen die ieder een functie hebben in de beveiliging van het platform. Elke laag, is ook betrokken bij het functioneren van het platform. Daaruit kunnen we concluderen dat er mogelijkerwijs ook organisatorische wijzigingen noodzakelijk kunnen zijn voor een adequate werking voor het functioneren en beveiligen.

Om verantwoord beheer van het systeem te kunnen waarborgen, dienen de volgende zes stappen vervuld te worden.

|  |  |  |
| --- | --- | --- |
| # | Laag | Omschrijving |
| 1 | Autorisatiebeheer | Binnen het platform komen veel gegevens met een vertrouwelijk karakter. Denk hierbij bijvoorbeeld aan MYSQL-gegevens. Het op de juiste wijze uitdelen en intrekken van autorisatie is dus van cruciaal belang. Dit geldt zowel voor de SSH public keys op het platform als voor de firewall rules. |
| 2 | Configuratiebeheer | Hoewel Puppet is gemaakt om dit onderdeel te verzorgen, zijn er een aantal redenen waarom configuratiebeheer hier van belang is. Het gaat namelijk niet om de configuraties op de verschillende servers, maar om de configuraties die uitgerold gaan worden op de verschillende servers. Het is raadzaam om dit gecontroleerd te houden. Als er bijvoorbeeld iets verkeerd wordt geconfigureerd of als een verkeerde (externe) gebruiker toegang heeft tot de servers, kan herleid worden wie dit geautoriseerd/ingesteld heeft. |
| 3 | Incidentbeheer | Aan de Puppetservers worden bepaalde eisen gesteld met betrekking tot beschikbaarheid. Mocht zich onverhoopt een storing voordoen dan is het belangrijk om procedures te hebben om deze storing te kunnen verhelpen. |
| 4 | Wijzigingsbeheer | Configuratiewijzingen kunnen een storing veroorzaken. Daarom is het van belang dat het wordt bijgehouden zodat er teruggevallen kan worden op een noodprocedure of een roll-back van de configuratie. |
| 5 | Operationeel beheer | Omdat het op een virtueel platform komt te draaien, heeft het systeem zelf alleen softwarematig beheer nodig. Het platform daaronder (dus het cloudplatform) heeft daarentegen wel periodiek onderhoud nodig. Maar omdat het cloudplatform waarop het gaat draaien al bestaat. En daarvoor al periodiek onderhoudsprocedures bestaan, gaat het hier alleen over het SDBA-project. Denk hierbij bijvoorbeeld aan security updates of bug fixes in de OS/software die gedraaid wordt. Logging en monitoring zijn hierbij zeer belangrijk. |
| 6 | Beveiligingsbeheer | Beveiliging is een van de meest belangrijke onderdelen. De wereld van leaks en hacks staat niet stil. Elke dag komen er nieuwe dreigingen bij. Daarom worden op dit systeem gepaste maatregelen genomen om de beveiliging te garanderen. Als organisatie moet je op deze dreigingen kunnen anticiperen. Daarom is besloten om alles voor de buitenwereld te blokkeren en ip-adressen van de mensen die onze developmentomgevingen willen benaderen selectief toe te voegen. |

Tabel 6 – Zes-stappen-beheer van systeem

### 8.1.3 - Gebruikers- en informatie-eisen

Een medewerker die de server gaat gebruiken moet de benodigde informatie, zoals wachtwoorden, kunnen bereiken. Zonder deze informatie kunnen de medewerkers die de server willen gebruiken niet functioneren. Het is daarom belangrijk om dit goed te regelen en er de onderstaande eisen aan te hangen:

* Informatie moet toegankelijk zijn, dat wil zeggen dat de informatie gedocumenteerd moet worden in de ticket die is toegewezen om de server te deployen.
* Informatie moet betrouwbaar zijn; we willen immers dat de medewerkers niet hoeven te zoeken naar juiste gegevens.
* De gegevens moeten up to date gehouden worden, zoals in tabel vijf #6 is beschreven. Dit moet gedaan worden zodat bijvoorbeeld het IP adres van een externe consultant of programmeur die niet meer aan het project werkt, uit de firewall van de desbetreffende project server wordt gehaald. Zo kan MaxServ de veiligheid waarborgen en toegang tot de development servers gecontroleerd houden.

**Hieronder is opgesteld welke eisen er aan de gebruikers gesteld worden:**

|  |  |
| --- | --- |
| # | Eis |
| 1 | Roottoegang op de servers is voor normale (development)medewerkers te allen tijde verboden. Zij hebben slechts toegang tot shell, via de beschikbaar gestelde www-data-accounts. Alleen de beheerders van Puppet hebben roottoegang. |
| 2 | Het toevoegen van SSH keys op de server of IP-adressen in de firewall moet te allen tijde door een geautoriseerd persoon gedaan worden. Dit kan vanuit het management of vanuit een van de desbetreffende projectmanagers zijn. Als de medewerker thuis wil werken dan wordt op aanvraag zijn private IP toegevoegd op de server waar hij op moet werken. |
| 3 | Aanpassingen aan de modules die uitgerold kunnen worden, moeten eerst gereviewd worden voordat ze op productie worden uitgerold. Dit moet gedaan worden door een tweede beheerder die er verstand van heeft en de configuratiecode kan lezen. |

### 8.1.4 - Eisen aan Puppetsoftware

SDBA moet natuurlijk ergens op draaien. Eerder was al voorbij gekomen dat het op een virtuele cloudomgeving gaat draaien. Natuurlijk heeft het gekozen pakket een cloudserver nodig waar het op gaat draaien en moet de server aan een aantal eisen voldoen. Hieronder volgt een overzicht van de belangrijkste eisen:

|  |  |  |
| --- | --- | --- |
| # | Eis | Omschrijving |
| 1 | Een 64 bit operating system | * Ubuntu 12.04 LTS * CentOS 6 * Debian 6, 7 |
| 2 | Modules die op de masterserver moeten komen | * Benodigde pakketten worden automatisch geïnstalleerd door de Puppet repository.  **Informatie voor het instellen van de repo:**<http://docs.puppetlabs.com/guides/puppetlabs_package_repositories.html> * Foreman GUI voor Puppet |
| 4 | Modules die de masterserver moet beheren | * **Apache2/HTTPD** * **MySQL** * **Firewall** * **SSH Keys** |
| 5 | Bestaande servers | * Er moet een core profile komen welke alleen de benodigde bestanden op de server installeert en ze gereedmaakt voor een complete overgang naar Puppet. |

Tabel 7 - Eisen SDBA

### 8.1.5 - Eisen aan de workflow en dataflow van SDBA

Puppet is gebouwd als een client-servermodel met een centrale server en agents die op verschillende servers draaien.

Een van de keuzes in Puppets applicatie-architectuur is dat clientnodes geen toegang krijgen tot de modules op de masterserver. In plaats van toegang te verlenen, worden deze modules in een configuratie samengesteld en geleverd aan de nodes.

**De werking zoals hierboven is omschreven biedt meerdere voordelen:**

1. Het principe ‘hoe minder privileges, hoe beter’. Een server weet precies hoe zijn configuratie eruit ziet en hoe hij geconfigureerd moet worden, maar de server weet niet hoe de andere servers geconfigureerd zijn.
2. Rechten kunnen compleet gescheiden worden om de configuratie te regelen, waaronder bijvoorbeeld ook de toegang tot de “centrale” dataopslag.
3. Omdat de hosts weten wat ze nodig hebben, kunnen zij zichzelf volledig configureren zonder verbinding te hoeven maken met een centrale server. Dit betekent dat de server altijd hetzelfde blijft, ondanks dat de masterserver bijvoorbeeld down is.

**De workflow is relatief eenvoudig;**

1. Maak een server aan op de OnApp cloud platform.
2. Installeer de Puppetagent en zorg dat hij zich aanmeldt op de server.
3. De Puppetagent vraagt een certificaat aan op de masterserver aan, die geautoriseerd moet worden door systeembeheer.
4. De Puppetagent verzamelt informatie over de host waarop het draait en geeft dit door aan de masterserver.
5. De masterserver geeft de configuratiebestand met modules door aan de node.
6. De Puppetagent op de node rolt de configuratie lokaal uit waardoor de lokale configuratie veranderd wordt. Vervolgens rapporteert hij terug aan de Puppet masterserver.

Afbeelding 4 - Workflow Puppet

De dataflow van de Puppet Master naar de agent is een voorgedefinieerd proces. De agent heeft toegang tot zijn eigen systeeminformatie, de configuratie en elk rapport dat het genereert. De master heeft kopieën van deze gegevens en heeft toegang tot alle Puppetmodules en alle diensten die nodig kunnen zijn om de configuraties samen te stellen. Standaard zal de agent de master om de 30 minuten controleren op veranderingen.

**De belangrijkste datatypes zijn:**

* Facts:
  + Systeemgegevens die op elke machine verzameld en gebruikt worden om configuraties samen te stellen.
* Manifest:
  + Bestanden met Puppetcode; deze bestanden worden meestal verzameld in een zogenaamde module.
* Catalog:
  + Hierin is de configuratie waar het systeem aan moet voldoen opgenomen.
* Report:
  + Hierin staan alle gebeurtenissen (bijvoorbeeld errors of waarschuwingen) die tijdens het configureren van een bepaalde catalogus naar voren komen vanuit de agent.



Afbeelding 5 - Dataflow Puppet

### 8.1.6 - Analyse transitiefase

Op dit moment wordt alles met de hand gedaan. Er zijn her en der wat scripts gemaakt die wel voor wat kleine automatiseringen zorgen. Daarom is door het management van MaxServ besloten om Puppet direct op alle systemen in te zetten.

Om het nieuwe platform te kunnen integreren, zijn er een aantal zaken die we op een rijtje moeten zetten:

* De huidige servers kunnen niet direct geconverteerd worden naar Puppet.
* Er wordt een staging server aangemaakt en vervolgens wordt daarop getest hoe het werkt.
* De huidige servers worden eerst voorzien van een core profile.
* Nieuwe servers worden wel direct via Puppet geconfigureerd.

De bovenstaande feiten zijn noodzakelijk om tot een goede implementatie te komen. Ze zijn bedacht onder het mom van “better safe then sorry”, wat ook past in de huidige strategie van de afdeling systeembeheer. Ook is de transitie op deze wijze makkelijker door te voeren en zal het minder complex zijn. De gevolgen zijn immers in verschillende scenario’s al bekend dankzij de staging server.

## 8.2 - Technisch ontwerp

Binnen het technisch ontwerp zullen we aandacht besteden aan onder meer:

* Ontwerp
* Benodigde software (modules)
* Benodigde server (Specs)
* Beveiliging

### 8.2.1 - Algemene informatie

Puppet wordt meestal ingezet als een eenvoudig client-servermodel. De hoofdserver vanwaaruit alle configuraties worden geserveerd, wordt de Puppet Master genoemd. De Puppetclients die op alle nodes (servers) gaan draaien worden agents genoemd. De agents zullen ervoor zorgen dat er verbinding gemaakt wordt met de Puppet Master.

De Puppet Master heeft een server-service draaien die alle configuratie voor de omgeving bevat. De Puppetagent maakt verbinding met de Puppet Master door middel van een ge-encrypte omgeving met standaard SSL en downloadt (ook wel Pull genoemd) de configuratie van de Puppet Master. Wat belangrijk is om te onthouden, is dat Puppet alleen maar configuraties wijzigt als deze nog niet zijn gewijzigd. Het bovenstaande wordt ook wel configuration run genoemd. Dat wil zeggen als een handmatig geïnstalleerde server wordt aangemeld, alle configuraties daarop wordt gewijzigd.

### 8.2.3 - Serverrollen

In Puppet kunnen we modules bundelen tot een class. Dat wil zeggen dat we niet alle modules los hoeven te definiëren in de node profile. De classes kunnen uitgebreid worden naarmate er meerdere losstaande typen servers komen. Deze classes kunnen fungeren als een serverrol die opgegeven kan worden als in een node profile.

* Core server - Binnen deze serverrole (class) zijn alleen de basis werksets opgenomen. De core server kan gebruikt worden voor de reeds bestaande servers die wel in Puppet moeten komen, maar nog niet rijp zijn voor de volledige implementatie.
* Webserver - Binnen de webserverrole (class) wordt de gehele developmentserver opgebouwd. Dit profiel bevat alle essentiële software die nodig is om een webserver op te kunnen zetten waar het CMS-systeem TYPO3 onder kan draaien. De class hoeft dus maar één keer aan de node profile toegevoegd te worden.

### 8.2.4 - Ontwerp

Deze paragraaf bevat een functiebeschrijving van de componenten welke benodigd zijn voor het opbouwen van de server. Daarbij wordt gedetailleerde technische informatie gegeven over de onderdelen die in het functioneel ontwerp zijn beschreven.

#### 8.2.4.1 - Software

Puppet biedt ontelbare modules aan voor het opbouwen van een server. Uit al deze modules zullen er maar een aantal gebruikt worden. De modules die geïmplementeerd worden, zijn ook al beschreven in het functionele ontwerp echter worden ze in deze paragraaf meer gedetailleerd omschreven.

Voor de opbouw van het Puppetplatform wordt de volgende software gebruikt:

|  |  |
| --- | --- |
| Type | Omschrijving |
| Operating System | Ubuntu 12.04 LTS |
| Server | Puppet Software   * Hiera: hiermee wordt het configureren van Puppet makkelijker gemaakt. Zo hoeft er geen volledig Manifest geschreven te worden en kunnen sysops YAML files gebruiken om hun wensen door te geven. |
| **GIT**  GIT is een versiebeheer controlesysteem die bestanden bijhoudt die zijn gewijzigd. Zo worden configuratiebestanden die op de Puppet Master zijn uitgerold bijgehouden. |
| **Foreman**  Hierin kunnen de statistieken van de hosts die draaien op Puppet, bekeken worden. Met Foreman heeft de sysop een grafische interface waarop gekeken kan worden of er fouten zijn ontstaan tijdens het installeren/configureren van de provisioned server. |

Tabel 8 - Software globaal

##### 8.2.4.1.1 - Role: Core

In deze role zijn de modules die de sysops dagelijks gebruiken opgenomen.



Afbeelding 6 – Overzicht core modules

|  |  |
| --- | --- |
| Module | Omschrijving |
| Wget | Wget is een tool voor het ophalen van bestanden via http, HTTPS en FTP. Het is een tool die via de commandline wordt aangeroepen en dus ook makkelijk in scripts te implementeren is. |
| Curl | Curl is om documenten/bestanden te verkrijgen van en naar een server. Curl ondersteunt een bulk aan protocollen zoals http, HTTPS, FTP, GOPHER, DICT, TELNET, LDAP of FILE. Net als wget wordt Curl aangeroepen via de commandline, dus is het makkelijker te implementeren in scripts. |
| Vim | Vim is een afkorting van Vi Improved. Het is een uitbreiding op Vi, een van de belangrijkste editors in de Linuxwereld. |
| Midnight Commander | Midnight Commander zal voor velen een bekende naam zijn. Het is een visual filemanager in een niet-grafische omgeving. Door middel van Midnight Commander kunnen bestanden en mappen eenvoudig worden gekopieerd of verplaatst. Dit is een veel gebruikte tool bij Linuxsysteembeheerders. |
| Multitail | In Multitail kunnen meerdere bestanden gelijktijdig geopend worden, terwijl het originele tailcommando maar één bestand kan openen. Multitail is daarnaast ook handig voor het openen van logbestanden omdat het interactief geladen wordt en regels real-time worden getoond. Het is dus eigenlijk een vervanger van watch. |
| Htop | Htop is een geavanceerde procesviewer. Het is vergelijkbaar met top, maar Htop heeft een aantal extra functies die het nog beter en overzichtelijker maken. Hierdoor is het een onmisbare tool geworden voor Linuxsysteembeheerders. |
| Firewall | De Firewallregels die hier getoond worden, vloeien voort uit paragraaf 8.1.3. Hier zijn specifieke veiligheidseisen gesteld omtrent toegangsregels. De onderstaande Firewallregels zijn samen met systeembeheer bepaald:   * Toegang van buiten moet te allen tijde geblokkeerd worden. * Kantoor-IP moet altijd toegang hebben tot alles: 83.80.22.236 * SNMP & SSH moeten altijd monitoring hebben. Whitelist: 83.80.22.237 * Zabbix Monitoring moet altijd toegang hebben. Whitelist 84.53.103.85 * Poorten 8140 en 61613 moeten altijd toegang hebben tot Puppet. Deze moeten wel open blijven en niet IP-gebonden zijn. Stel dat Puppet een IP change krijgt, dan is het cruciaal dat de master aan de agents zijn nieuwe IP kan doorgeven. |

Tabel 9 - Core profile software

##### 8.2.4.1.2 - Role: Webserver

In deze role zullen de modules opgenomen worden die ervoor zorgen dat het CMS TYPO3 juist kan draaien op een webserver die gedeployed wordt.



Afbeelding 7 - Webserver Role

|  |  |
| --- | --- |
| Module | Omschrijving |
| Apache | * Apache moet OTAP variabele automatisch meekrijgen in zijn configuratie. Die worden vanuit de node profiel gegeven.   + otap: o (ontwikkeling)   + otap: p (productie voor in de toekomst) * Automatische configuratie van Vhost en activering daarvan. * De module moet zowel commando’s van Debian familysystemen als Redhatsystemen ondersteunen.   + **Debian Family:** packages: ['apache2', 'apache2-mpm-prefork', 'libapache2-mod-php5', 'php5-cli', 'php5', 'php5-gd', 'php5-curl', 'php5-mcrypt']   + **Redhat Family:**   packages: ['httpd', 'php', 'php-gd', 'php-mbstring', 'php-soap', 'php-xml'] |
| Mysql | * De collatie moet altijd UTF-8 zijn. * Database moet automatisch aangemaakt worden met de gegevens die worden opgegeven in de node profile. * De mysql config moet beïnvloedbaar zijn vanuit de node profile. |
| SSH Keys | * De public keys van alle developers moeten op de servers worden toegevoegd zodat ze toegang hebben tot het www-data-account. * De public keys van alle beheerders zodat deze toegang hebben tot root als www-data. |

Tabel 10 - Webserver software

### 8.2.5 - Hardware

De Puppet Master gaat op de cloudomgeving draaien. Zo kan de server niet alleen uitgebreid worden qua specificaties maar maakt het het back-uppen van de gehele server ook makkelijker. De servers die aangemaakt worden op het cloudplatform van MaxServ worden altijd dagelijks geback-upt die zo teruggerold kunnen worden. Daarnaast kan bijvoorbeeld na een volledige crash, een nieuwe server opgebouwd worden uit de geback-upte images. Dit zorgt ervoor dat de down time drastisch wordt verlaagd.

In de documentatie van Puppet staan de volgende adviesgegevens:

|  |
| --- |
| The puppet master role should be installed on a robust, dedicated server.  Minimum requirements: 2 processor cores, 1 GB RAM, and very accurate timekeeping.  Recommended requirements: Physical hardware, or VMware or Xen or KVM virtual server, with 2-4 processor cores and 4 GB RAM. Performance will vary, but this configuration can generally manage approximately 1,000 agent nodes. |

Tabel 11 – Hardware-eisen volgens Puppet [[3]](#footnote-4)

De eisen die hierboven worden genoemd vormen slechts een indicatie. Om in ieder geval aan de basiseisen te voldoen en om de server genoeg capaciteit en power te geven zijn de specificaties iets opgeschroefd (afbeelding 8). Echter, de specificaties kunnen altijd worden aangepast doordat het op een virtuele omgeving draait.



Afbeelding 8 - Hardwarespecificaties

### 8.2.6 - Uptime & Monitoring

Bij een nieuw platform moet ook rekening gehouden worden met monitoring en uptime. Daar is in deze scriptie nog niet over gesproken, maar het is wel een belangrijk aspect in het technische ontwerp.

#### 8.2.6.1 - Uptime

In gesprek met de afdeling systeembeheer en het management is besloten dat er rekening gehouden moet worden met de volgende eisen tijdens het ontwerp van de Puppet Master:

* De Puppet Master moet een betrouwbare server worden. Daarom is er een doel gesteld van een jaarlijkse beschikbaarheid van minimaal 98%.
* De MTTR (Mean Till Repair) mag binnen kantooruren niet boven de 2 uur uitkomen.

#### 8.2.6.2 - Monitoring

Binnen MaxServ wordt gebruikgemaakt van het monitoringpakket Zabbix. Zabbix is monitoringsoftware die checkt op uptime en die statistieken bijhoudt, zoals bijvoorbeeld de serverload. De servers die gemonitord worden, worden bijgehouden door middel van SNMP.

Ook kunnen we met behulp van Zabbix om de 10 minuten een test laten uitvoeren die controleert op specifieke poorten. Met de uitkomst van deze testresultaten kunnen we vanuit Zabbix een periodiek overzicht krijgen met uptimestatistieken van de server.

**Poorten die gemonitord dienen te worden:**

|  |  |
| --- | --- |
| Poort | Omschrijving |
| 8140 | Puppet Master poort |
| 22 | SSH-poort |
| 443 | HTTPS-poort (Foreman Dashboard) |

### 8.2.6 - Ontwerp inrichting server

In afbeelding 9 zien we een weergave van het design van de Puppet Master. De keuze van het ontwerp is niet standaard vloeit voort uit een platform met hoge beschikbaarheid en performance te bouwen. De extra onderdeel die erbij is gekomen is Memcached. Verder in dit hoofdstuk wordt uitgelegd wat de modules betekenen.



Afbeelding 9 - Indeling Server

De server is opgebouwd uit twee fictieve lagen. De standaard serverlaag en de Ruby/Rack. De standaard serverlaag bestaat uit de apache webserver in combinatie met Passenger en Memcached. Apache2 is een bekende webserver binnen de Linuxwereld om webapplicaties op te draaien. Om Ruby/Rack-applicaties binnen een Apache2 te kunnen draaien (waar Puppet en Foreman op gebaseerd zijn, de fictieve laag 2), is er een communicatieservice genaamd Passenger voor geplaatst. Passenger is een module voor apache die ook wel mod\_rails of mod\_rack wordt genoemd. Memcached is een caching mechanisme dat gebruikt wordt om de snelheid van Foreman te optimaliseren. Memcached slaat dynamische aanvragen op in de RAM om zo vraag naar “externe” databases/API’s te verkleinen en alles sneller te serveren. Vervolgens hebben we in diezelfde laag nog MYSQL en GPG. MYSQL wordt gebruikt door Foreman om zijn data wat hij heeft gekregen van Puppet op te slaan. Denk hierbij aan de reports en facts. GPG beveiligd de secrets die in Puppet zijn geplaatst; denk hierbij aan MYSQL-wachtwoorden.

In de tweede laag bevinden zich de applicaties Puppet Master, Foreman en Hiera. Laag 2 is de Ruby/Rack stack waarop beide applicaties draaien. De Foreman proxy fungeert als een portal. Hierin kunnen elementen aangeroepen worden zoals de Puppet CA (om certificaten te signen). Foreman is een dashboard waarin alle statistieken en nodes worden weergegeven. Alles wat hierin aangepast wordt, heeft invloed op de Puppet Master. Deze zijn immers met elkaar verbonden.

Eerder in de pakketselectie was al aangegeven dat er een extra keuze was in taalniveau. Hieradata (DSL van Puppet) is hier de tweede keuze; het is makkelijk leesbaar voor systeembeheerders die geen programmeerkennis hebben.

De redenen waarom voor Hiera gekozen is:

* Makkelijker om nodes te configureren met standaard data omdat je overrides kan maken in modules.
* Makkelijker her te gebruiken code.
* Makkelijk om gemaakte modules online te zetten zonder gevoelige data te verwijderen omdat statische bedrijfsdata niet in de module is verwerkt vanwege die overrides.

**Voorbeeld Hieracode:**

databases:

XXXX:

username: "XXXX"

password: "XXX"

vhosts:

hakan.scriptie.voorbeeld:

path: "/var/www/scriptie"

ssl: yes

certificateName: "wildcard-uzunlar-nl"

typo3: yes

**Voorbeeld standaard Puppetcode:**

apache::vhost { ' hakan.scriptie.voorbeeld:

port => '443',

docroot => '/var/www/scriptie',

ssl => true,

ssl\_cert => '/etc/ssl/ wildcard-uzunlar-nl’,

ssl\_key => '/etc/ssl/ wildcard-uzunlar-nl,

}

mysql::db { 'XXXX':

user => 'XXXX',

password => 'XXX',

host => 'localhost',

grant => ['ALL'],

}

### 8.2.7 - Installatie op Puppetcliënt en aanmelding op master

De Puppetclient moet aangemeld worden op de master voordat begonnen wordt met de installatie. Dat moet gebeuren nadat de server is aangemaakt via OnApp. De standaardprocedure van Puppet beschrijft dat dit handmatig moet gebeuren.

Om dit in een keer goed te doen, heb ik samen met mijn collega een bash script gemaakt. De scripts controleren of Puppet al aanwezig is of niet. Zo niet, voegt hij de software repository van Puppet toe en begint de server met de installatie van de puppetclient. Nadat hij klaar is met installeren, voegt hij in de hostfile van de desbetreffende Node (server) de IP entries en hostnames van de Puppet Master toe.

Nadat het gehele script is uitgevoerd, moet het certificaat op de master erver gesigneerd worden. Dit kan gedaan worden met het commando “puppet cert sign <nodename>“

Beide scripts zijn uit te voeren met ./debian.sh of ./redhat.sh

|  |
| --- |
| **Debian Family servers** |
| #!/bin/bash  if pgrep puppet > /dev/null 2>&1; then  echo puppet is already running, not good, bye.;  ……  sed -i 's/START=no/START=yes/g' /etc/default/puppet  /etc/init.d/puppet start  **Volledig script is te vinden in bijlage 1** |

|  |
| --- |
| **Redhat Family servers** |
| #!/bin/bash  if pgrep puppet > /dev/null 2>&1; then  echo puppet is already running, not good, bye.;  exit 1;  …  rpm -ivh http://yum.puppetlabs.com/el/${VERSION}/products/i386/puppetlabs-release-${VERSION}-${MINOR}.noarch.rpm  yum clean all  yum install puppet -y  puppet agent --waitforcert 60 --test  /etc/init.d/puppet start  **Volledig script is te vinden in bijlage 2** |

## 8.3 - Implementatieplan en beschrijving

Het implementatieplan bestaat uit verschillende fases. Er zal met het ontwerp worden gestart in een testomgeving (PoC, oftwel de Proof of Concept). Deze omgeving zal de functionaliteiten hebben die de productieomgeving in MaxServ ook zal hebben.

De fases waaruit de implementatie zal bestaan, zijn hieronder weergegeven:

|  |  |
| --- | --- |
| # | Faseomschrijving |
| 1 | Controleren huidige omgeving |
| 2 | Ontwerpeisen verzamelen |
| 3 | Bouwen van de Puppet Server |
| 4 | Testen van de server |
| 5 | Testen van Roles op servers |

Tabel 12 - Implementatieplan

### 8.3.1 - Beschrijving

#### 8.3.1.1 - Controleren huidige omgeving

* Ubuntu X64 Webserver
* CentOS X64 Webserver

In de huidige omgeving draaien alle server in een autonome stand. Er is geen deploymentmechanisme dat de omgeving installeerd of beheerd. Alle servers die nu draaien, zijn allemaal met de hand geconfigureerd en worden ook met de hand beheerd.

#### 8.3.1.2 - Ontwerpeisen verzamelen

* **Een virtual server met:**
  + 4 CPU
  + 8GB Ram
  + 300GB
* **Software op de server:**
  + Puppet
  + Foreman
  + Hiera
  + Ruby/Rack
  + GPG
  + MySQL
  + Apache
    - Passenger
  + Memcached

#### 8.3.1.3 - Bouwen van de Puppet Server

* Aanmaken virtuele server in OnApp
  + Ubuntu X64
* Installatie Puppet (Installeert automatisch Ruby)
  + Opbouwen benodigde modules.
* Installatie Foreman (Foreman installeert automatisch Apache2)
* Installatie Memcached
* Installatie GIT

#### 8.3.1.4 - Testen van de server

* Aanmaken profile nieuwe server en pushen naar GIT
* Installatie nieuwe development server
* Overzetting van een bestaande development server naar de core profile om te testen
  + Back-up!
* Testen connectiviteit en installatie.

#### 8.3.1.5 - Testen van profiles

* Aanmelden server op Puppet
* Aanmaken Node Profile
* Controleren via Foreman op succesrate en verloop.

# 9 - Proof of Concept

Het volledige technische ontwerp is aan de hand van hoofdstuk zeven en hoofdstuk acht uitgevoerd.

Als eerste is een server deployed met de besturingssysteem Ubuntu 14.04 LTS (Long Term Support). Vervolgens zijn de software elementen die zijn beschreven geïnstalleerd en geoptimaliseerd, zoals in de bronnen [11] Turnbull, James, and Jeffrey McCune. 2011. Pro Puppet, [12] Labs, P., Docs: Installing Puppet en [13] Foreman, The Forman :: Manual zijn beschreven. Na de installatie van de server zijn de, waardoor de server is geïnstalleerd en operationeel is in de testomgeving.

De omgeving bestaat uit twee scenario’s, namelijk de core server en de webserver. Beide worden aan de hand van een specifiek testplan getest.

## 9.1 - Testplan

### 9.1.1 - Installatie Puppetserver

|  |  |  |
| --- | --- | --- |
| # | Benodigdheden | Resultaat/Commentaar |
|  | Aanmaken server in OnApp |  |
|  | Installatie Puppet Master |  |
|  | Installatie Foreman |  |

### 9.1.2 - Deployen server + Aanmelden

|  |  |  |
| --- | --- | --- |
| # | Benodigdheden | Resultaat/Commentaar |
|  | Aanmaken server in OnApp |  |
|  | Runnen van Puppetagent scriptinstallatie |  |
|  | Certificaat signeren op de Puppet Master |  |

### 9.1.3 - Deployen Core role

|  |  |  |
| --- | --- | --- |
| # | Benodigdheden | Resultaat/Commentaar |
|  | Aanmaken node profile met de class  maxserv::roles::core |  |
|  | Runnen van puppet agent -t[[4]](#footnote-5) |  |
|  | **Bekijken of basisonderdelen zijn deployed:** |  |
|  | Wget  Curl  Vim  Midnightcommander  Multitail  Htop |  |
|  | Firewall + Rules (Iptables -L -n) |  |

### 9.1.4 - Deployen Webserver role

|  |  |  |
| --- | --- | --- |
| # | Benodigdheden | Resultaat/Commentaar |
|  | Aanmaken node profile met de classes   * maxserv::roles::webserver * maxserv::roles::core |  |
|  | Runnen van puppet agent -t |  |
|  | **Verifiëren of de onderdelen zijn geïnstalleerd:** |  |
|  | **Debian Family:** packages: ['apache2', 'apache2-mpm-prefork', 'libapache2-mod-php5', 'php5-cli', 'php5', 'php5-gd', 'php5-curl', 'php5-mcrypt'] |  |
|  | **Redhat Family:**  packages: ['httpd', 'php', 'php-gd', 'php-mbstring', 'php-soap', 'php-xml'] |  |
|  | **MYSQL** |  |
|  | Wget  Curl  Vim  Midnightcommander  Multitail  Htop |  |
|  | Firewall + Rules (Iptables -L -n) |  |

# 

## 9.2 - Testresultaten

De tests zijn uitgevoerd op de meest gebruikte development OS namelijk Ubuntu 12.x tot 14.x.

### 9.2.1 - Installatie Puppetserver

|  |  |  |
| --- | --- | --- |
| # | Benodigdheden | Resultaat/Commentaar |
|  | Aanmaken server in OnApp | Algemeen bekende proces dat succesvol is afgerond. |
|  | Installatie Puppet Master | Resultaat:  \* Starting Puppet Master [ OK ] |
|  | Installatie Modules | Modules zijn succesvol geïnstalleerd. |
|  | Installatie Foreman | Installatie Foreman succesvol. |

#### 9.2.1.1 - Testresultaat - installatie Puppet Master

Voor de installatie zijn de volgende stappen gevolgd:

**Toevoegen van de Puppet repository:**

|  |
| --- |
| $ wget https://apt.puppetlabs.com/puppetlabs-release-trusty.deb  $ dpkg -i puppetlabs-release-trusty.deb  $ apt-get update |

**Installatie Puppet Master en Passenger:**

|  |
| --- |
| $ apt-get install puppetmaster-passenger $ apt-get install puppetmaster |

**Installatie modules voor role webserver:**

|  |
| --- |
| **puppet module install puppetlabs-mysql** /etc/puppet/environments/production/modules  └─┬ puppetlabs-mysql (v2.2.3)  └── puppetlabs-stdlib (v4.1.0) [/etc/puppet/environments/common]  **puppet module install puppetlabs-apache**  /etc/puppet/environments/production/modules  └─┬ puppetlabs-apache (v1.0.1)  ├── puppetlabs-concat (v1.0.2) [/etc/puppet/environments/common]  └── puppetlabs-stdlib (v4.1.0) [/etc/puppet/environments/common] |

**Create role file:   
/etc/puppet/environments/production/modules/maxserv/manifests/roles/Core.pp**

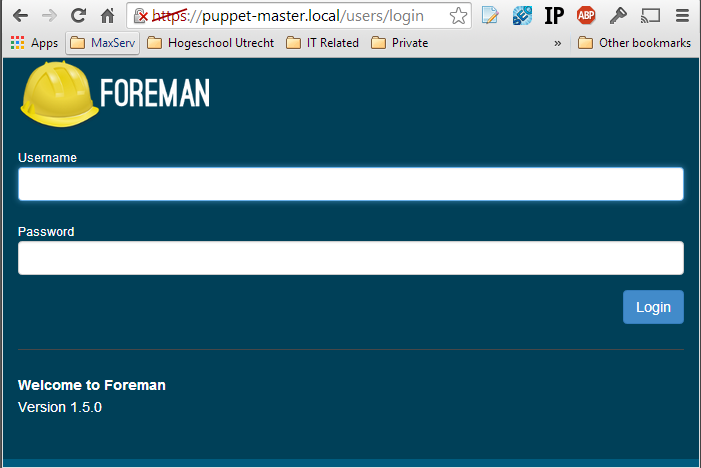
|  |
| --- |
| Complete script te zien in bijlage 3 |

**Create role file:  
/etc/puppet/environments/production/modules/maxserv/manifests/roles/Core webserver.pp:**

|  |
| --- |
| class maxserv::roles::webserver {  include maxserv::roles::core  include maxserv::services::apache  include maxserv::services::mysql  } |

**Installatie Foreman:**

|  |
| --- |
| echo "deb http://deb.theforeman.org/ trusty 1.5" > /etc/apt/sources.list.d/foreman.list  echo "deb http://deb.theforeman.org/ plugins 1.5" >> /etc/apt/sources.list.d/foreman.list  wget -q http://deb.theforeman.org/foreman.asc -O- | apt-key add -  apt-get update && apt-get install foreman-installer  **$ foreman-installer** Preparing installation Done  Success!  \* Foreman is running at https://puppet-master.local  Default credentials are 'admin:changeme'  \* Foreman Proxy is running at https://puppet-master.local:8443  \* Puppetmaster is running at port 8140  The full log is at /var/log/foreman-installer/foreman-installer.log |



### 9.2.2 - Deployen server + Aanmelden

|  |  |  |
| --- | --- | --- |
| # | Benodigdheden | Resultaat/Commentaar |
|  | Aanmaken server in OnApp | Algemeen bekend proces dat succesvol is afgerond |
|  | Runnen van Puppetagent scriptinstallatie | Certificaat succesvol aangevraagd |
|  | Certificaat signeren op de Puppet Master | Certificaat succesvol gesigneerd |

Server agent geeft een error 400 terug omdat er geen node profile is aangemaakt, maar dit wordt in de volgende stappen uitgevoerd.

### 9.2.3 - Deployen Core role

|  |  |  |
| --- | --- | --- |
| # | Benodigdheden | Resultaat/Commentaar |
|  | Aanmaken node profile met de class  maxserv::roles::core |  |
|  | Runnen van puppet agent -t |  |
|  | **Bekijken of basisonderdelen zijn deployed:** |  |
|  | Wget  Curl  Vim  Midnightcommander  Multitail  Htop | Onderdelen zijn geïnstalleerd |
|  | Firewall + Rules (Iptables -L -n) | Firewall Rules zijn doorgevoerd |

**Aanmaken node profile met de class maxserv::roles::core:**

|  |
| --- |
| **Puppet-client.local.yaml**  ---  otap: P  classes:  - maxserv::roles::core |

**Firewall Check:**

|  |
| --- |
| **Iptables -L -n** **target prot opt source destination**  ACCEPT icmp -- 0.0.0.0/0 0.0.0.0/0 /\* 000 accept all icmp \*/  ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 /\* 001 accept all to lo interface \*/  ACCEPT all -- 0.0.0.0/0 0.0.0.0/0 /\* 002 accept related established rules **\*/ state** RELATED,ESTABLISHED  ACCEPT tcp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 8140 /\* 005 Puppe Traffic \*/  ACCEPT tcp -- 83.80.22.236 0.0.0.0/0 multiport dports 22 /\* 300 SSH inbound for MaxServ Office \*/  ACCEPT udp -- 0.0.0.0/0 0.0.0.0/0 multiport dports 123 /\* 301 NTP port IN \*/  ACCEPT udp -- 83.80.22.237 0.0.0.0/0 multiport dports 161 /\* 304 SNMP Monitoring IN \*/  ACCEPT tcp -- 83.80.22.237 0.0.0.0/0 multiport dports 22 /\* 304 SSH Monitoring IN \*/  ACCEPT tcp -- 84.53.103.85 0.0.0.0/0 multiport dports 10050,10051 /\* 304 Zabbix Monitoring IN \*/  DROP all -- 0.0.0.0/0 0.0.0.0/0 /\* 999 drop all \*/ |

### 9.2.4 - Deployen Webserver role

|  |  |  |
| --- | --- | --- |
| # | Benodigdheden | Resultaat/Commentaar |
|  | Aanmaken node profile met de classes   * maxserv::roles::webserver * maxserv::roles::core | Het aanmaken van de profile was makkelijk. Omdat alle items in de module al zijn gedefinieerd, hoeft in de node profile alleen aangegeven te worden hoe alles moet heten. |
|  | Runnen van puppet agent -t | Puppetagent succesvol gedraaid |
|  | **Verifiëren of de onderdelen zijn geïnstalleerd:** |  |
|  | **Debian Family apache2ctl -M & php -m:** packages: ['apache2', 'apache2-mpm-prefork', 'libapache2-mod-php5', 'php5-cli', 'php5', 'php5-gd', 'php5-curl', 'php5-mcrypt'] | Alle modules zijn succesvol geïnstalleerd en werken correct.  root@derm:~# apache2 -v  Server version: Apache/2.2.22 (Ubuntu) |
|  | **MYSQL** | root@derm:~# mysql  Welcome to the MySQL monitor. Commands end with ; or \g.  Your MySQL connection id is 12824  Server version: 5.5.37-0ubuntu0.12.04.1 (Ubuntu) |
|  | Wget  Curl  Vim  Midnightcommander  Multitail  Htop | Onderdelen zijn geïnstalleerd. |
|  | Firewall + Rules (Iptables -L -n) | Iptables zijn ook goed; MaxServ heeft alleen toegang. Externe toegang is geblokkeerd. |

Aanmaken node profile met de classes (een daadwerkelijke development server is aangemaakt):

|  |
| --- |
| ---  otap: O  classes:  - maxserv::roles::core  - maxserv::roles::webserver  databases:  derm\_dev:  username: "dermdb"  password: "XXXXXX"  vhosts:  derm.dev.maxserv.com:  path: "/var/www/derm-appeal"  typo3: yes |

## 9.3 - Conclusie PoC

Wanneer we alle testresultaten in ogenschouw nemen, kan het volgende geconcludeerd worden:

* De Puppet installatie is zoals ze zelf aangeven in hun handleiding vrijwel direct te gebruiken. Het moeilijke aan de installatie is echter wel de structuur die aangemaakt moet worden, vooral in combinatie met Hiera.
* Het uitrollen van de verschillende rollen op de servers is met succes getest. De nodes installeren de gevraagde elementen die zijn gedefinieerd. De installatie is een korte tijd gedaan wat met handmatige installatie niet in te halen is.
* De beste werkwijze en makkelijkste werkwijze voor systeembeheer is om alle bestanden die Puppet Master gebruikt door middel van GIT op de Puppet Master uit te rollen.

Al met al heeft het pakket gewerkt en is mijn doel voor MaxServ om een goed werkend pakket op te leveren bereikt.

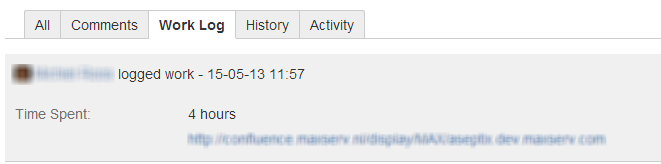
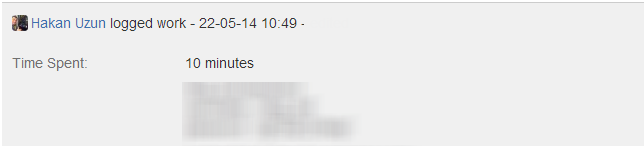
# 10 - Conclusie en aanbevelingen

## 10.1 - Conclusie

Voor MaxServ heeft dit onderzoek duidelijk resultaat opgeleverd. Het doel was om een zo efficiënt mogelijk product op te leveren om serverinstallaties uit te voeren. Het gekozen pakket, Puppet, is een zeer efficiënt product als het gaat om installaties en deployments. Wel moeten de servers handmatig aangemaakt worden in onApp maar dat vertraagt het proces niet.

De uitdaging tijdens het invoeren van het systeem was de overlast voor de developers zoveel mogelijk te verlagen en de continuïteit te waarborgen. Hieronder viel ook het verlagen van de down time tijdens het overgaan van de server naar een Puppet controlled omgeving.

Waar de meeste tijd bespaard wordt, is de installatie van een nieuwe server. In plaats van vier uur om een installatie in combinatie met finetunen uit te voeren, kost het systeembeheer nu tien tot vijftien minuten om een dergelijke server op te leveren (zie Afbeelding 10). Dat wil zeggen dat we in de vier uur die het voorheen kostte om een server op te leveren nu bijna zestien servers kunnen opleveren en kunnen beginnen aan ontwikkeling.



Afbeelding 10 - Situatie voor en na de implementatie

Hieruit kan geconcludeerd worden dat het pakket ook de overgebleven eis dekt die MaxServ heeft gesteld, namelijk:

* het pakket moet de beheertaken verminderen of vergemakkelijken.

Ook de security is naar een hoger niveau getild. Developers hebben alleen toegang tot plaatsen/functies van de server waar ze moeten zijn. Als zij expliciete roottoegang willen, moeten zij dit aangeven bij systeembeheer die hun toegang kan geven tot de rest van de server. Ook worden de firewallregels centraal geregeld. Alles wordt geblokkeerd en alleen IP-adressen die gespecificeerd zijn in de nodeconfiguratie of in de coreconfiguratie worden toegelaten. Door het toepassen van de firewall policy worden twee vliegen in een klap geslagen. De eerste is natuurlijk de security die hierboven is genoemd, maar daarnaast ook een stuk beveiliging voor zoekmachine-indexering zodat de development websites bijvoorbeeld niet in de Google zoekresultaten worden opgenomen. Voorheen werd wel eens vergeten om een robots.txt in de siteroot te plaatsen en werd het developmentdomein opgenomen in de searchindex van Google. Nu is dat verleden tijd omdat de crawlers ook geblokkeerd worden. Die kunnen nu niet meer bij de developmentservers komen.

Kortom, MaxServ en vooral de systeembeheerders gaan veel aan dit pakket hebben. Niet alleen qua kosten maar ook qua beheergemak.

## 10.2 - Aanbevelingen

Hieronder worden de aanbevelingen aan MaxServ opgesomd:

* Maak altijd gebruik van de GIT repo die is ingericht bij het aanmaken van nieuwe modules of node configuratiebestanden en werk nooit rechtstreeks op de Puppet Master server. Zo kan er bij problemen snel geschakeld worden en bestanden die problemen veroorzaken teruggerold worden.
* Tijdens de PoC is het wel eens voorgekomen dat de servers bij een configuratiefout niet meer juist werkten (denk bijvoorbeeld aan een Firewall rule). Controleer dus altijd de modules en configuratiebestanden die aangemaakt worden in een staging omgeving. Zo voorkom je problemen en is deployment van de wijziging 100% gecontroleerd.
* Zorg altijd voor up to date public keys van de systeembeheerders. Hierdoor voorkom je ongeoorloofd root toegang tot systemen. (voorbeeld: één systeembeheerder die wilt programmeren wat op de development afdeling komt te zitten)
* Blijf binnen het pakket ontwikkelen met nieuwe elementen/modules en breid het uit. Binnen het pakket is the sky the limit.

# 11 - Evaluatie

Het uitvoeren van dit project in combinatie met de dagelijkse werkzaamheden hebben geleid tot een zeer drukke maar vooral een leerzame periode.

Ik ben al ruim 6 jaar werkzaam in het bedrijf en het blijft toch lastig om alles binnen je eigen organisatie vanuit een onpartijdig perspectief te bekijken. Maar gelukkig is het mij met de hulp van mijn begeleider en een frisse kijk op nieuwe technologie die MaxServ naar een hoger niveau moet tillen het toch gelukt om mijzelf onpartijdig op te stellen.

Met behulp van de planning en wat extra overuren is het gelukt om de gestelde deadline te halen en een werkend product af te geven dat gebruikt kan worden in de productie en zelfs al gebruikt wordt voor het maken van developmentservers. De afdeling systeembeheer van MaxServ B.V. heeft mijn product zeer enthousiast ontvangen. Daarnaast wil ik betrokken blijven bij het product en ga ik mijzelf hard maken voor en voortborduren op de aanbeveling die ik als laatst heb gegeven in hoofdstuk 10, paragraaf 10.1:

* Blijf binnen het pakket ontwikkelen met nieuwe elementen/modules en breid het uit. Binnen het pakket is the sky the limit.

Ik hoop mijn opleiding af te ronden nu ik mijn opdracht succesvol heb afgerond.

Hakan Uzun

Giessen, 23 mei 2014

# 12 - Afkortingen

PoC - Proof of Concept

SDBA - Server Deployment Automatisering

MoSCoW - M= Must haves, S=Should haves, C=Could haves, W=Won’t haves

GIT - Een versiebeheer systeem. (http://www.git-scm.com/)

# 13 - Literatuurlijst

1. Cuppen, J.P.H.M, and Bernard Cuppen. *IT in bedrijf: veranderingsanalyse, WfM en pakketselectie*. Den Haag: Academic Service, 2010.
2. Grit, R. *Projectmanagement: projectmatig werken in de praktijk*. Groningen [etc.]: Noordhoff, 2011.
3. Jan Dik Zegel. *Van probleem naar prestatie: afstuderen in de HBO-praktijk*. Hilversum: Op de i, 2011.
4. Nathans, Hanna. *Adviseren als tweede beroep: resultaat bereiken als adviseur*. Deventer: Kluwer, 2005.
5. Steehouder, M.F, and Maruschka Gijsen. *Leren communiceren*. Groningen [etc.]: Wolters-Noordhoff, 2006.
6. Puppetlabs, Puppetlabs.com, <http://puppetlabs.com/about/customers>,  
   laatst geraadpleegd: 23-04-2014
7. CFEngine, cfengine.com, <https://cfengine.com/product/what-is-cfengine/>
8. Laatst geraadpleegd: 23-04-2014
9. CA technologies, ca.com, <http://www.ca.com/~/media/Files/ProductBriefs/ca-configuration-automation-12-5-ps.pdf>,  
   laatst geraadpleegd: 10-03-2014
10. Delaet, T., Joosen, W., & Vanbrabant, B. (2010, November). A survey of system configuration tools. In *Proceedings of the 24th Large Installations Systems Administration (LISA) conference (San Jose, CA, USA, 11/2010 2010), Usenix Association, Usenix Association*, <https://www.usenix.org/legacy/event/lisa10/tech/full_papers/Delaet.pdf>

Laatst geraadpleegd: 10-04-2014

1. Turnbull, James, and Jeffrey McCune. 2011. *Pro Puppet Cover Title. - “Maximize and Customize Puppet’s Capabilities for Your Environment.”* [Berkeley, Calif.]: Apress.
2. Labs, P., Docs: Installing Puppet, <http://docs.puppetlabs.com/guides/install_puppet/pre_install.html>  
   Laatst geraadpleegd: 22-05-2014
3. Foreman, The Forman :: Manual, <http://theforeman.org/manuals/1.5/index.html>  
   Laatst geraadpleegd: 22-05-2014
4. Bill Driscoll (09/06/2009) Special Issue: IT Automation: Server Configuration Management <http://www.networkcomputing.com/wireless/229607346>  
   Laatst geraadpleegd: 15-04-2014
5. The Modern Network, How Puppet Mastery Can Make the Network More Manageable for Federal Agencies, <http://themodernnetwork.com/government/how-puppet-mastery-can-make-the-network-more-manageable-for-federal-agencies/>  
   Laatst Geraadpleegd: 25-04-2014
6. Steve Wexler, (22/09/2011), Puppet Seeks Wider IT automation Market,  
   http://www.networkcomputing.com/private-cloud/231601936

Laatst geraadpleegd: 25-04-2014

1. Puppet Configuration Management Tool ,SparkSupport.com,  
   <http://www.sparksupport.com/blog/puppet-configuration-management-tool>  
   Laatst geraadpleegd: 26-04-2014

# 14 - Overzicht gebruikte afbeeldingen en tabellen

[Afbeelding 1- OnApp controlepaneel manueel OS selectie 14](#_Toc389163784)

[Afbeelding 2 - Illustratie CM-proces 15](#_Toc389163785)

[Afbeelding 3 - Berenshot-methode 16](https://d.docs.live.net/7d8763345f4c27f5/Hogeschool%20Utrecht/Huidige%20Projecten/Afstuderen/Scriptie/Scriptie_Server_deployment_automatisering_Hakan_Uzun_Final_v1.0.docx#_Toc389163786)

[Afbeelding 4 - Workflow Puppet 28](#_Toc389163787)

[Afbeelding 5 - Dataflow Puppet 29](#_Toc389163788)

[Afbeelding 6 – Overzicht core modules 32](#_Toc389163789)

[Afbeelding 7 - Webserver Role 33](#_Toc389163790)

[Afbeelding 8 - Hardwarespecificaties 34](#_Toc389163791)

[Afbeelding 9 - Indeling Server 36](#_Toc389163792)

[Afbeelding 10 - Situatie voor en na de implementatie 48](#_Toc389163793)

[Tabel 1 - Hoe OnApp zichzelf omschrijft 13](#_Toc389163794)

[Tabel 2 - Betekenis MoSC0w 17](#_Toc389163795)

[Tabel 3 - MoSCoW inclusief weging 18](#_Toc389163796)

[Tabel 4 - Pakketselectie Longlist 19](#_Toc389163797)

[Tabel 5 - Pakketselectie Shortlist 20](#_Toc389163798)

[Tabel 6 – Zes-stappen-beheer van systeem 26](#_Toc389163799)

[Tabel 7 - Eisen SDBA 27](#_Toc389163800)

[Tabel 8 - Software globaal 31](#_Toc389163801)

[Tabel 9 - Core profile software 32](#_Toc389163802)

[Tabel 10 - Webserver software 33](#_Toc389163803)

[Tabel 11 – Hardware-eisen volgens Puppet 34](#_Toc389163804)

[Tabel 12 - Implementatieplan 39](#_Toc389163805)

# Bijlage 1 - Ubuntu Aanmeldscript

|  |
| --- |
| #!/bin/bash  if pgrep puppet > /dev/null 2>&1; then  echo puppet is already running, not good, bye.;  exit 1;  else  echo puppet is not running, good.;  if dpkg -l puppet >/dev/null 2>&1; then  echo puppet is already installed, not good, bye.;  exit 1;  else  echo puppet is not yet installed, good.;  fi  fi  if command -v wget > /dev/null 2>&1; then  echo wget is already installed;  else  echo installing prerequisite wget;  apt-get install wget;  fi  if command -v lsb\_release > /dev/null 2>&1; then  echo "installing for:";  lsb\_release -a  else  apt-get install lsb-release -y  fi  wget http://apt.puppetlabs.com/puppetlabs-release-`lsb\_release -sc`.deb  dpkg -i puppetlabs-release-`lsb\_release -sc`.deb  apt-get update  apt-get install puppet -y  echo "192.168.23.129 puppet-master.local puppet-master" >> /etc/hosts;  echo puppet master added to hosts file;    puppet agent --waitforcert 60 --test  sed -i 's/START=no/START=yes/g' /etc/default/puppet  /etc/init.d/puppet start |

# Bijlage 2 - Redhat Family Aanmeldscript

|  |
| --- |
| #!/bin/bash  if pgrep puppet > /dev/null 2>&1; then  echo puppet is already running, not good, bye.;  exit 1;  else  echo puppet is not running, good.;  if rpm -qa | grep puppet > /dev/null 2>&1; then  echo puppet is already installed, not good, bye.;  exit 1;  else  echo puppet is not yet installed, good.;  fi  fi  MINOR=7  if command -v lsb\_release > /dev/null 2>&1; then  echo "installing for:";  lsb\_release -a;  VERSION=`lsb\_release -sr | cut -d. -f1`;  MINOR=`lsb\_release -sr | cut -d. -f2`;  else  if [ ! -f /etc/system-release-cpe ]; then  echo "Cannot determine OS release/version, not good.";  exit 1;  else  echo "Installing for: `cat /etc/system-release-cpe`";  VERSION=`cat /etc/system-release-cpe | cut -d: -f5`  fi    rpm -ivh http://yum.puppetlabs.com/el/${VERSION}/products/i386/puppetlabs-release-${VERSION}-${MINOR}.noarch.rpm  yum clean all  yum install puppet -y  echo "192.168.23.129 puppet-master.local puppet-master" >> /etc/hosts;  echo puppet master added to hosts file;  puppet agent --waitforcert 60 --test  /etc/init.d/puppet start |

# Bijlage 3 - core.pp

|  |
| --- |
| class maxserv::roles::core {  class { 'maxserv::firewall': }  #Ensure the puppetmaster is present in /etc/hosts  file {'/etc/hosts':  ensure => present  } ->  file\_line {'puppetmaster':  ensure => present,  path => '/etc/hosts',  line => '192.168.23.129 puppet-master.local puppet-master puppet'  }  #Install some packages our sysops use every day  include maxserv::packages::wget  include maxserv::packages::curl  include maxserv::packages::git  include maxserv::packages::vim  include maxserv::packages::midnightcommander  #The multitail and htop packages are custom built on CentOS,  #So we'll have to include the GCC toolchain to make them  if $osfamily == 'RedHat' {  class { 'maxserv::packages::toolchain': }  package { 'redhat-lsb':  ensure => latest  }  }  include maxserv::packages::multitail  include maxserv::packages::htop  #Include required core services  include maxserv::services::monitoring  include maxserv::services::ntp  include maxserv::services::puppetagent  include maxserv::services::ssh  } |

**Pakketten die zijn defined in core:**

**Wget.pp**

|  |
| --- |
| class maxserv::packages::wget {  package { 'wget':  ensure => latest  }  } |

**Curl.pp**

|  |
| --- |
| class maxserv::packages::curl {  package { 'curl':  ensure => latest  }  } |

**Vim:**

|  |
| --- |
| class maxserv::packages::vim {  case $osfamily {  'RedHat': {  $packages = ['ctags', 'vim-enhanced']  }  'Debian': {  $packages = ['exuberant-ctags', 'vim']  }  default: { fail("Unrecognized operating system for Vim") }  }  package { $packages:  ensure => latest  }  } |

**MC (Midnight Commander): mc.pp**

|  |
| --- |
| class maxserv::packages::midnightcommander {  package { 'mc':  ensure => 'installed'  }  } |

# Bijlage 4 - Plan van Aanpak

1. http://onapp.com/platform/ [↑](#footnote-ref-2)
2. Storage area network: Een SAN is een server op het netwerk die dedicated is aan dataopslag. [↑](#footnote-ref-3)
3. http://puppetlabs.com/puppet/requirements [↑](#footnote-ref-4)
4. Puppet agent -t: Dit is de commando, die ervoor zorgt dat de server contact opneemt met de Puppet Master. En vervolgens zijn configuratie download. [↑](#footnote-ref-5)