Afstudeerverslag

Het ontwikkelen van een monitoringsysteem

Student:	Wan Long Cheung
Student nummer:	11035463
Opleidingsinstituut:	De Haagse Hogeschool Johanna Westerdijkplein 75 2521 EN Den Haag
Academie:	ICT & Media
Opleiding:	Informatica
Datum:	3-juni-2015
Versie:	1.0
Eerste examinatoren:	E.M. van Doorn
Tweede examinator:	G.M. Tuk
Opdrachtgever:	ENOVATION Rivium Westlaan 1 2909 LD Capelle a/d IJssel
Bedrijfsbegeleider:	Albert van ’t Hart

Referaat

Afstudeerverslag, Wan Long Cheung, Het ontwikkelen van een monitoringsysteem, De Haagse Hogeschool, ENOVATION, februari 2015.

De afstudeeropdracht is uitgevoerd in de periode van februari 2015 t/m juni 2015 bij het bedrijf ENOVATION te Capelle a/d IJssel. Deze afstudeerverslag beschrijft het proces rondom de afstudeeropdracht. De opdracht is een onderdeel van de opleiding Informatica aan De Haagse Hogeschool.

Descriptoren:

- Docker
- Influxdb
- Grafana
- Consul
- Java
- Spring Framework
- Scrum

Voorwoord

Deze afstudeerverslag is het resultaat van de uitgevoerde afstudeeropdracht “Het ontwikkelen van een monitoringsysteem” bij ENOVATION in de periode van februari 2015 t/m juni 2015. Met dit verslag wil ik aantonen dat ik op een professionele wijze heb gewerkt aan mijn afstudeeropdracht.

Ik wil graag ENOVATION bedanken voor het beschikbaar stellen van de afstudeeropdracht en Albert van ’t Hart bedanken voor de begeleiding gedurende mijn afstudeerperiode. Ook wil ik alle collega’s van ENOVATION bedanken die mij hebben geholpen en het gezellig hebben gemaakt bij het uitvoeren van de afstudeeropdracht. Daarnaast wil ik ook mij studieloopbaan begeleider Gerda in ’t Veld bedanken voor al haar steun en advies gedurende de opleiding. Tenslotte wil ik E.M. van Doorn en G.M. Tuk bedanken voor de begeleiding vanuit De Haagse Hogeschool.

Wan Long Cheung
Capelle a/d IJssel, juni 2015.

Inhoudsopgave

Documenteigenschappen

Historie

Versie	Datum	Auteur	Wijziging
0.1	12-02-2015	Wan Long Cheung	Opzet
0.2	10-02-2015	Wan Long Cheung	Uitgevoerde sprints beschrijven
0.3	12-03-2015	Wan Long Cheung	Eerste opleveringsmoment
0.4	15-03-2015	Wan Long Cheung	Feedback verwerken
0.5	10-04-2015	Wan Long Cheung	Uitgevoerde sprints beschrijven
0.6	21-04-2015	Wan Long Cheung	Tweede opleveringsmoment
0.7	04-05-2015	Wan Long Cheung	Uitgevoerde sprints beschrijven
0.8	10-05-2015	Wan Long Cheung	Tussentijds oplevering
0.9	22-05-2015	Wan Long Cheung	Feedback verwerken van tussentijds assessment
1.0	03-06-2015	Wan Long Cheung	Definitief maken

Inleiding

Voor mijn opleiding Informatica op De Haagse Hogeschool te Den Haag wordt er in de laatste fase een afstudeeropdracht uitgevoerd, waarin moet blijken dat ik op een bepaald niveau kan werken en denken. In de periode van februari 2015 t/m juni 2015 heb ik een afstudeeropdracht uitgevoerd bij het bedrijf ENOVATION te Capelle a/d IJssel. De afstudeeropdracht houdt in dat er een monitoringsysteem ontwikkeld wordt dat aansluit op de nieuwe architectuur met microservices.

Dit afstudeerverslag is ingedeeld in drie delen. In het eerste deel van het verslag wordt de projectachtergrond beschreven met nadere toelichting op de algemene informatie van het bedrijf, de aanleiding tot het schrijven van de afstudeeropdracht en de te verwachtte resultaten.

In het tweede deel van het verslag beschrijf ik de uitvoering van het proces. Hierin beschrijf ik per sprint waaraan ik heb gewerkt, welke problemen er zijn en welke keuzes ik heb genomen. Er worden voorbeelden gegeven om problemen duidelijker te beschrijven, gevolgd door een nadere toelichting van de gemaakte keuzes.

In het derde deel van het verslag wordt er geëvalueerd op de uitvoering van het proces en de opgeleverde producten. In de evaluatie beschrijf ik de sterke en verbeterpunten van de uitvoering en welke stap ik anders zou doen voor de volgende keer. Daarnaast wordt er ook geëvalueerd op het beroepstaken. In de bijlage is er een woordenlijst opgenomen, waarin de technische begrippen worden uitgelegd. Daarnaast zijn de opgeleverde documenten tijdens het project in de bijlage te vinden.

Deel I. Project achtergrond

In dit deel van het afstudeerverslag beschrijf ik de achtergrondinformatie m.b.t. het afstudeerproject. Hierbij wordt het bedrijf beschreven, de afdeling waar het afstudeerproject wordt uitgevoerd en de werkwijze van ENOVATION. Daarnaast wordt ook de aanleiding van het project beschreven.

1.1 Bedrijf

ENOVATION is een internationaal ICT-bedrijf, dat actief is in de zorg, en transport & logistieke sector. Het bedrijf zorgt voornamelijk voor de uitwisseling van elektronische berichten tussen mensen, organisaties en systemen. De vestiging bij Capelle aan de IJssel bestaat ongeveer uit 100 werknemers en verwacht de komende jaar een groei van 10 procent.

Deze afstudeeropdracht is uitgevoerd binnen het bedrijf ENOVATION bij de afdeling Product Development. De ontwikkelafdeling is verdeeld in een aantal groepen. Deze zijn: Platform, LSPConnect, ZorgMail en Luna. Hieronder volgt een beschrijving van de teams binnen de ontwikkelafdeling en de producten.

Platform

In deze team wordt er gewerkt aan generieke oplossingen voor de producten van ENOVATION. De systemen van ENOVATION hebben op een aantal onderdelen na overlap met elkaar, omdat er geen standaard componenten zijn die ingezet kan worden. Hiervoor worden componenten ontwikkeld, zodat deze ingezet kunnen voor meerdere systemen en tijd wordt bespaard in het ontwikkelproces. Het ontwikkelen van een monitoringsysteem wordt een standaard component, daarom wordt het afstudeerproject binnen deze team uitgevoerd.

LSPConnect

Het LSP (Landelijk Schakelpunt) maakt het voor zorgverleners mogelijk om actuele medische gegevens van een patiënt te kunnen opvragen. Hiermee kunnen de juiste behandelingen gegeven worden aan patiënten. Met LSPConnect worden bestaande systemen geïntegreerd met het LSP. Dit is nodig, omdat er veel eisen zijn rondom de beveiliging en beschikbaarheid van de digitale informatie (medische gegevens). In deze groep wordt het ontwikkelen van LSPConnect voortgezet.

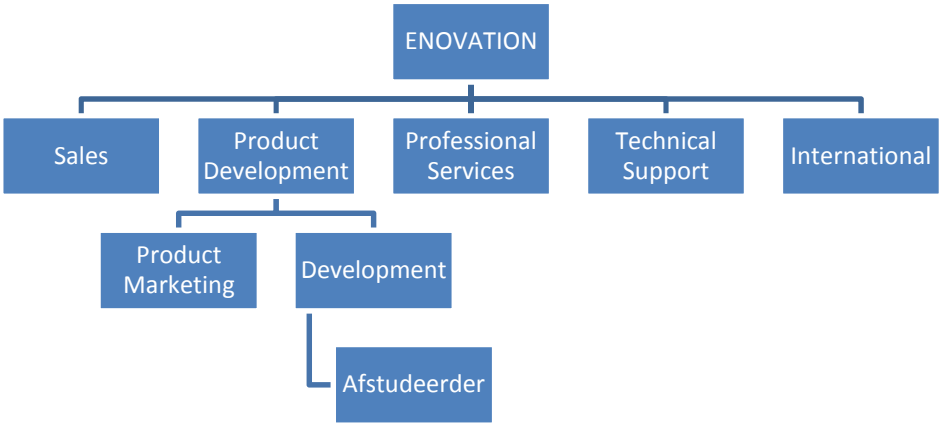
ZorgMail

ZorgMail is een systeem dat bestaat uit componenten die zorgverleners ondersteunen bij het digitaal uitwisselen van gegevens. Het is een systeem dat gebruikt wordt binnen alle disciplines van de zorg en is geconfigureerd om altijd beschikbaar te zijn. Een grootdeel van alle elektronische berichten in de zorg wordt via ZorgMail verzonden. Alle ontwikkelingen rondom ZorgMail wordt binnen deze team uitgevoerd.

Luna

In dit team worden maatwerk software ontwikkeld, maar tegenwoordige worden de applicaties alleen nog onderhouden. De focus van deze team ligt momenteel bij het migreren van de Rijkswaterstaat mail systeem. Daarnaast houdt het team zich ook bezig met add-ons voor ZorgMail.

Organisatie structuur



1.2 Werkwijze

De producten van ENOVATION worden met de software ontwikkelmethodiek Scrum ontwikkeld. De sprints hebben een duur van drie weken. Gedurende een sprint werkt een team van ongeveer vier medewerkers aan het product. Om de teamleden en teamleider (meestal de Scrum Master maar niet altijd) op de hoogte te houden is er dagelijks een meeting rondom de Scrumboard. Aan het einde van elk sprint wordt er een korte demo gegeven aan de belanghebbende. Afhankelijk van de omstandigheden worden voorbereidingen aan het einde of begin van een sprint uitgevoerd. Door te werken met korte sprints kan er gemakkelijk ingespeeld worden op de verandering rondom het project.

1.3 Probleemstelling

ENOVATION beschikt over een eigen hostingcenter voor hosting van systemen van ENOVATION en hosting van systemen van derden zoals Rijkswaterstaat, Politie Haaglanden, Stichting Beurshal en diverse informatiesystemen voor de zorg. Huidige systemen van ENOVATION hebben veelal een traditionele monolithisch opzet, waarbij deze vaak op een vast aantal servers draaien. ENOVATION beschikt over een standaard monitoring omgeving (Nagios), maar de aansluiting van nieuwe software op deze omgeving kost veel tijd en is niet gestandaardiseerd. Om een koppeling te maken met de omgeving worden handmatige handelingen uitgevoerd en maatwerk software gemaakt. ENOVATION gaat in de komende jaren migreren van een architectuur met monolithische systemen naar een systeem dat bestaat uit microservices. Met het huidige monitorsysteem is het monitoren van microservices een veel werk eisende activiteit. Er kunnen meerdere instanties van een microservice gedraaid worden om de belasting te kunnen verdelen over de servers. In de toekomst zullen er dus een groot aantal microservices actief zijn. Daarnaast zal de deployment van deze microservices ervoor zorgen dat deze niet (altijd) meer op een vast aantal servers draaien. Het kan dus zijn dat er na de deployment een microservice op twee services wordt gehost en het niet bekend is op welke servers de microservice is. Tenslotte is geen standaard set van data dat van een microservice verzameld kan worden voor monitoring.

1.4 Doelstelling

ENOVATION wil voor het monitoren van systemen een standaard monitoringsysteem inzetten die inzetbaar is voor alle systemen dat bestaat uit microservices. Hierbij gaat het om het opslaan van diverse gegevens. Enkele voorbeelden hiervan zijn het processor capaciteit gebruik, geheugen gebruik en een aantal diverse foutmeldingen. Deze gegevens worden op een later tijdstip of real-time opgehaald en op een dashboard getoond.

1.5 Verwachte resultaat

Een monitoringsysteem (prototype) dat aansluit op nieuwe wensen en eisen van ENOVATION en welke gemakkelijk inzetbaar is bij de ontwikkeling van nieuwe systemen op basis van microservices. Het monitoringsysteem bestaat uit componenten dat metrics (meetgegevens) verzamelt, opslaat en toont. Hiermee kan een systeembeheerder gemakkelijk de systemen monitoren en wordt er tijd bespaard in zowel het beheer- als ontwikkelproces.

1.6 Microservices

ENOVATION wil in de komende tijd migreren naar een architectuur met microservices bij het ontwikkelen van applicaties. Het monitoringsysteem dient hier rekening mee te houden, omdat het een van de eisen zijn bij het project. Een definitie van microservices (Newman, 2015):

> *"Microservices are small, autonomous services that work together. Small, and focused on doing one thing well."* [1]
>
> *-Sam Newman-*

[1] Bron – Newman, S. (2015). *Building Microservices.* O'Reilly. (Hoofdstuk 1 Pagina 2)

Microservices zijn kleine services die met elkaar samenwerken. Bij elk microservice ligt de focus on een specifieke taak. Een voorbeeld van een microservice is een service dat alleen de accounts gegevens van gebruikers beheerd. Om inzicht te krijgen over microservices worden de voor- en nadelen beschreven.

Technologie

Een van de belangrijkste voordelen van microservices is dat er met de diverse technologie gewerkt kan worden. Zo kan er voor elke component een ander techniek toegepast worden dat beter is om een bepaalde functies uit te voeren. Met microservices is het eenvoudig om de nieuwe technieken uit te proberen, omdat niet de hele applicatie aangepast hoeft te worden. Er kan gekozen worden om een simpel component dat weinig impact heeft voor het systeem, als het niet goed functioneert, te ontwikkelen met de nieuwe techniek. Bij een monolithische applicatie is niet eenvoudig om nieuwe technieken toe te passen zonder dat het een grote impact heeft op het bestaande systeem. Het veranderen van programmeer taal of Framework kan dus niet eenvoudig worden toegepast.

Schaalbaarheid

Bij een monolithische architectuur is het alleen mogelijk om het systeem in zijn geheel op te schalen. Als een onderdeel van het systeem veel resources gebruikt en deze geen apart component is dan is de enige oplossing het hele systeem opschalen. Bij een architectuur met microservices is het eenvoudig om een microservice op te schalen, want het is een component van een systeem. Het opschalen van een microservice kan gedaan worden door twee instantie van de service te draaien.

Deployment

Een ander voordeel van microservices is dat het afzonderlijk gedeployed kan worden bij een kleine aanpassing in een component. Een aanpassing in een component betekent dus dat alleen dat component opnieuw gedeployed moet worden. Bij een monolithische applicatie is het nodig dat de gehele applicatie wordt gedeployed bij een kleine aanpassing.

Monitoring

Het monitoren van een applicatie met een microservices architectuur is complex, want het bestaat uit meerder componenten en er kunnen meerder instanties van een component gehost worden. Om de monitoring goed te kunnen uitvoeren is het nodig dat er metrics van alle instanties worden bij gehouden. Bij een applicatie met een monolithische architectuur is het eenvoudig om de metrics te verzamelen en te tonen, want er is maar een verzameling van metrics. Bij microservices ligt het anders, want van elk instantie is er een verzameling van metrics. Er is dan veel informatie en het wordt heel snel onoverzichtelijk, maar door de metrics te aggregeren kan het overzichtelijker worden. Daarnaast is het vinden van de oorzaak van een foutmelding niet eenvoudig, want er kan een cascade ontstaat, waar een fout van de ene microservices veroorzaakt wordt door een ander microservices. Bij een applicatie met een monolithische architectuur is eenvoudiger te achterhalen dan met microservices, omdat het bestaat uit een component.

Deel II. Uitvoering

In het afstudeerproject is er gewerkt met sprints vanaf het begin. Dit is gedaan om de voortgang van het project te kunnen monitoren en de deliverables kunnen hierdoor geëvalueerd worden door een ander. De deliverables kunnen dus naast prototypes ook documentatie zijn. De sprints hebben een tijdsduur van drie weken. Voor het inplannen van de sprint doel is er aan het begin van de sprint een moment met de opdrachtgever ingepland. De daily stand up meeting wordt gehouden met het platformteam, omdat het monitoringsysteem een onderdeel wordt van het platform.

In dit deel van het afstudeerverslag worden de uitgevoerde werkzaamheden en de keuzes die gemaakt zijn per sprint beschreven.

2.1 Sprint 1 – Voorbereiden voor het ontwikkeltraject

Het doel van deze sprint is het maken van een aanpak, waarin duidelijk vermeld staat op welk manier ik het project ga uitvoeren. Daarnaast geeft het ook inzicht over de beschikbare tijd voor het uitvoeren van de afstudeeropdracht. Een tweede doel van deze sprint is het uitvoeren van een onderzoek over systeem monitoring binnen ENOVATION, zodat het duidelijk wordt waar rekening mee gehouden moet worden het te ontwikkelen monitoringsysteem.

2.1.1 Plan van aanpak

Methodiek

Voor dit project is er gekozen om met een Agile methodiek te werken, omdat de veranderende eisen van de stakeholders hiermee gemakkelijk op te vangen is. De stakeholders hebben een idee van welke functionaliteiten het te ontwikkelen systeem moet hebben, maar deze kunnen in het verloop van tijd nog gaan veranderen. Door van prototype naar prototype te werken moet het duidelijk geworden welke oplossing het beste bij ENOVATION past.

In dit project is er gekozen om met SCRUM te werken, omdat het een bekende methodiek is voor mij en ENOVATION. Hierdoor is het niet nodig om de methodiek te onderzoeken. Daarnaast zijn voldoende contactmomenten met de opdrachtgever om de voortgang van het afstudeerproject te bespreken, zodat het in de goede richting gestuurd kan worden. Ook heb ik goede ervaring opgedaan met SCRUM in mijn derdejaars stage. Hierdoor heb ik meer vertrouwen in SCRUM.

Er wordt gewerkt met een aangepaste SCRUM methodiek, omdat niet alle activiteiten toepasbaar zijn binnen het project en volgens de officiële SCRUM beschrijving (scrumguides, 2015) moet een SCRUM team minimaal uit drie teamleden bestaan. Er worden opnieuw userstories verzameld aan het begin van elke sprint, want na het presenteren van de demo kan de vooraf verzamelde userstories en prioriteit daarvan veranderen.

Scope

Het ontwikkelen van een monitoringsysteem kan breed getrokken worden. Om een afgerond product te kunnen opleveren aan het einde, wordt er in de eerste instantie gefocust op het kunnen verzamelen, opslaan en tonen van metrics. Het ontwikkelen van alarmeringen valt buiten de scope.

Risico factoren

Voor dit project is er een inschatting gemaakt van factoren die de voortgang van het project kunnen belemmeren. De eerste factor is leren werken met een time series database. In mijn opleiding heb ik alleen ervaring opgedaan met een relationele database. Om dit risico te verkleinen wordt er tijd besteed om te leren werken met een time serie database.

De tweede factor is het leren om gaan met Spring Framework, want ook hier heb ik geen ervaring opgedaan. Om dit risico te verkleinen wordt er tijd besteed om de basis van Spring Framework te leren kennen. Dit wordt gedaan door de tutorials te volgen, het boek van Spring Framework te lezen en bestaande projecten te bekijken.

Planning

Er is een keuze gemaakt om een sprint duur van drie weken te houden, omdat de sprints in het bedrijf ook een duur hebben van drie weken. In de afstudeerperiode zijn er dan vijf sprints gedefinieerd met twee weken voor uitlooptijd. Er is een planning gemaakt waarbij er in de eerste sprint voorbereidende werkzaamheden uitgevoerd worden om goed te kunnen starten met het ontwikkelen van een monitoringsysteem. Een van de voorbereidende werkzaamheden is het maken van een plan van aanpak. In sprint twee t/m vijf staat gepland om uitvoerende werkzaamheden te doen zoals ontwikkelen en testen (Bijlage C Planning).

(In de externe bijlage is het volledige plan van aanpak te vinden.)

2.1.2 Onderzoek

Om het onderzoek goed te kunnen verrichten is er als eerst een onderzoeksplan gemaakt. Met het onderzoeksplan wordt het duidelijk voor zowel de onderzoeker zelf en de opdrachtgever waar naar onderzoek gedaan wordt. In het onderzoeksplan staat onder andere de hoofdvraag met daarbij de deelvragen van het onderzoek en de manier waarop informaties worden verzameld. In de externe bijlage is het volledige onderzoeksrapport te vinden. De belangrijkste onderdelen worden hieronder toegelicht.

In dit project is er onderzoek gedaan naar het huidige monitorsysteem van ENOVATION. De hoofdvraag die hierbij hoort is:

Waar moet er rekening mee gehouden worden bij het ontwikkelen van een monitoringsysteem?

Voor het kunnen beantwoorden van de hoofdvraag worden de volgende deelvragen gesteld:

- Wat wordt er verstaan onder systeem monitoring?
- Hoe ziet de huidige situatie van systeem monitoring eruit?
- Hoe ziet de ontwikkelstraat van ENOVATION eruit?
- Waar wil ENOVATION naar toe met het te ontwikkelen monitoringsysteem?

Voor het beantwoorden van de deelvragen zijn er diverse methoden toegepast, waar onder literatuur onderzoek en interviews.

Samenvatting onderzoek

Het doel van een monitoringsysteem is om metrics van applicaties te kunnen verzamelen. Aan de hand van de gegevens kunnen inschatting gemaakt worden, zodat er gepaste handelingen worden uitvoert om de service te kunnen optimaliseren. Een voorbeeld hiervan is door op tijd een tweede instantie van een service te draaien om de belasting aan te kunnen. In de huidige situatie van ENOVATION is het monitoringsysteem verouderd en niet flexibel. Er wordt veel handmatige handelingen uitgevoerd om de applicaties en servers te kunnen monitoren. In de komende tijd wil ENOVATION over stappen naar een architectuur met microservices. Met het huidige monitoringsysteem wordt het monitoren van microservices een zware taak, omdat een microservice dynamisch en in een groot aantal is. ENOVATION wil een nieuw monitoringsysteem hebben, waarbij het monitoren van microservers gemakkelijk uitgevoerd kan worden en het bestaande monitoringsysteem geheel kan vervangen. Om een nieuw monitoringsysteem te kunnen ontwikkelen wordt er rekening gehouden met de ontwikkelstraat van ENOVATION. Het monitoringsysteem wordt uiteindelijk een platform component dat gemakkelijk inzetbaar is voor alle applicaties. Het monitoringsysteem kan ingedeeld worden in drie fases. Binnen de afstudeeropdracht wordt de focust gelegd op eerste fase en dat is het kunnen opslaan en tonen van metrics.

Deelvraag 1: Wat wordt er verstaan onder systeem monitoring?

Onder systeem monitoring wordt er verstaan een systeem continue in de gaten houden, waarbij meetwaardes worden verzameld om te kunnen anticiperen in de toekomst. Meest bekende meetwaardes bij een systeem monitor zijn processor kracht en geheugen gebruik.

Deelvraag 2: Hoe ziet de huidige situatie van systeem monitoring eruit?

Het huidige monitor systeem van ENOVATION bestaat uit diverse componenten. Deze zijn Nagios, Monitor Demon, Statistic Collector, RRD files en Cacti. In het huidige monitoringsysteem worden er veel onderhoud en handmatige handelingen uitgevoerd om de systemen te kunnen monitoren. In de toekomst wil ENOVATION migreren naar een architectuur met microservices. Hierdoor is het huidige systeem niet geschikt voor de toekomst.

Deelvraag 3: Hoe ziet de ontwikkelstraat van ENOVATION eruit?

ENOVATION ontwikkeld voornamelijk applicaties op de Java platform met Spring Framework. Er worden diverse tools gebruikt om het ontwikkeltraject te ondersteunen. Verder wordt er gewerkt met de software methodiek SCRUM.

Deelvraag 4: Waar wil ENOVATION naar toe met het te ontwikkelen monitoringsysteem?

ENOVATION wil voor veel voorkomende onderdelen in de applicaties componenten ontwikkelen, want deze kunnen opnieuw gebruikt worden in de applicaties. Daarnaast is het gebruiken van componenten efficiënter voor het ontwikkeltraject, want er wordt tijd bespaard. Hierdoor kan er gefocust worden op het ontwikkelen van nieuwe functionaliteiten.

Hoofdvraag: Waar moet er rekening mee gehouden worden bij het ontwikkelen van een monitoringsysteem?

Bij het ontwikkelen van het monitoringsysteem moet er rekening mee gehouden worden met microservices, want ENOVATION wil in de komende tijd migreren naar een architectuur met microservices. Wegens de beperkte tijd dat er aan het project besteedt kan worden is de focus in de eerste instantie om metrics te kunnen opslaan en te tonen. Het is belangrijk dat metrics op een eenvoudig manier verzameld kunnen worden, want anders kost het veel tijd om dit mogelijk te maken en dan is het geen verbetering meer t.o.v. het huidig systeem.

2.1.3 Interviews

Allereerst was het belangrijk om de stakeholders van het project te identificeren, want zij hebben belang bij het te ontwikkelen monitoringsysteem. De stakeholder van het project zijn de ontwikkelaars, architecten en beheerders. De ontwikkelaars hebben het huidige monitoringsysteem ontwikkeld en zullen in de toekomst het nieuwe monitoringsysteem implementeren. De architecten hebben een visie over het te ontwikkelen monitoringsysteem. De beheerders zijn de eindgebruikers van het monitoringsysteem en dus belangrijk voor het project. Er is gekozen om openinterviews te houden, omdat er doorgevraagd kan worden om bepaalde onderdelen helder te krijgen. Daarnaast zijn er ook een aantal vragen voorbereid om de huidige situatie helder te krijgen.

Ontwikkelaar

Om de huidige situatie te verkennen is een ontwikkelaar geïnterviewd die verstand heeft van het huidige monitoringsysteem. Voor de interviews zijn er van te voren een aantal vragen voorbereid om het interview in de goede richting te kunnen sturen. Ook is er gekeken naar de tools en pakketten die gebruikt werden in de huidige situatie. Uit het interview is gebleken dat het huidige monitoringsysteem niet eenvoudig aanpasbaar is. Voor een simpel wijziging zoals het aanpassen van het dashboard (interface voor metrics) moet er veel werk gedaan worden.

De vragen die voorbereid zijn voor de ontwikkelaar zijn:

1. *Waaruit bestaat het huidige monitoringsysteem?*
2. *Hoe werkt het huidige monitoringsysteem?*
3. *Wat moet er gedaan worden om een applicatie te kunnen monitoren?*

Architect/Opdrachtgever

De architecten zijn geïnterviewd om de samenhang van het te ontwikkelen monitorsysteem en de applicaties te verduidelijken. Hieruit is gebleken waaraan het monitoringsysteem moet voldoen en waar rekening gehouden moet worden. Een van de belangrijke punten was dat het monitorsysteem moet kunnen omgaan met microservices, want in de komende tijd wordt er gemigreerd naar een architectuur met microservices.

Beheerder

De beheerders weten welke gegevens verzameld en opgeslagen moeten worden. Hiervoor waren ook een aantal vragen voorbereid om het helder te krijgen over de gegevens die met het huidige monitoringsysteem verzameld waren en welke functionaliteiten belangrijk zijn voor een monitoringsysteem. Hieruit is gebleken dat basis metrics zoals CPU load, Memory en Heap in de eerste fase het belangrijkste zijn. Daarnaast is alarmeringen ook belangrijk, maar deze valt buiten de scope van het project.

De vragen die voorbereid zijn voor de beheerder zijn:

1. *Hoe wordt het huidige* monitoringsysteem *gebruikt?*
2. *Welke metrics moeten er verzameld worden?*
3. *Welke functionaliteiten van zijn belangrijk bij een monitoringsysteem?*

De userstories zijn allemaal in Jira (online scrumboard) geplaats. Wegens het grote aantal userstories worden alleen de belangrijkste userstories opgenomen in het verslag. In de userstories staat beschreven waaraan er ontwikkeld wordt, waaraan het moet voldoen en welke taken er zijn. De prioriteiten van de userstories zijn bepaald met de MosCoW methode.

2.2 Sprint 2 – Metrics opslaan en tonen

In de voorgaande sprint was het duidelijk geworden welke metrics belangrijk zijn bij een monitoringsysteem (bijlage D Enkele userstories). Deze sprint is bedoeld voor het kunnen verzamelen van metrics. Om de metrics te kunnen verzamelen moet er allereest gezocht worden naar bestaande oplossingen binnen het Spring Framework[2] en Spring Boot[3], want de applicaties zijn hiermee ontwikkeld. De gevonden oplossingen zijn vervolgens toegepast in een testproject om de mogelijkheden te bestuderen. In dit hoofdstuk wordt aandacht besteed aan de gemaakte keuzes rondom het verzamelen van metrics.

2.2.1 Metrics types

Een monitoringsysteem verzamelt metrics om de status van een applicatie om te kunnen monitoren. Metrics zijn gegevens die vertellen wat de prestatie is van een applicatie. Enkele voorbeelden van metrics zijn: Memory, CPU average en http status codes (zie bijlage E). Aan de hand van de metrics kunnen geplande handelingen uitgevoerd worden. Een voorbeeld hiervan is: de CPU average van de afgelopen tijd analyseren om een rustig moment van een dag te kunnen kiezen voor het uitvoeren van een update in de productieomgeving.

Metrics hebben een type en elke type is bedoeld voor het opslaan van een ander soort waarde. De meest eenvoudig types zijn Gauge en Counter. Daarnaast zijn er ook geavanceerde types zoals Meters en Histogram. Deze types zijn gedefinieerd in de Spring Boot Actuator en Coda Hale's Metrics libraries. Met de libraries wordt het verzamelen van gegevens van de applicatie een stuk eenvoudiger gemaakt. Hieronder worden de Metrics type beschreven.

Gauge

Een Gauge is het meest eenvoudige type dat gebruikt kan worden voor het bijhouden van een waarde. Gauge wordt gebruik voor het opslaan van één waarde van een moment. Een voorbeeld waar Gauge voor gebruik kan worden is het bijhouden van memory gebruik.

Counter

Bij een counter wordt een teller bijgehouden en het teller zal in de meeste gevallen alleen oplopen. Dit wordt meestal toegepast voor het bijhouden van het aantal keren dat iets aangeroepen is. De waarde van de counter zal in de meeste gevallen alleen maar stijgen en niet dalen. Een voorbeeld waar Counter wordt toegepast is het bijhouden van aantal login pogingen.

Meter

Met *meter* wordt de frequentie van een event bijgehouden, daarbij worden het gemiddelde per 1-, 5- en 15-minuten berekend vanaf het moment dat de applicatie actief is. Wanneer de applicatie herstart dan begint de *meter* weer bij nul. Het gemiddelde kan ook anders berekend worden, door alleen een *counter* bij te houden en met behulp van aggregaat functies van Influxdb. Enkele voorbeelden van aggregaat functies zijn: count, min, max en mean. Met aggregaat functies kunnen waardes berekend worden van een set data. Het nadeel van aggregaat functies is dat het uitvoeren van de query langer kan gaan duren, omdat er berekeningen gedaan worden. Vergeleken met *meter* worden de waardes van te voren berekend en vervolgens opgeslagen in de database.

[2] Spring Framework – Een Framework voor het Java platform
[3] Spring Boot – Autoconfiguratie voor Spring Framework

De belasting van het berekenen van de waarde ligt dus aan de kan van de applicatie bij het gebruiken van *meter*. Het voordeel van aggregaat functies is dat er alleen berekeningen worden uitgevoerd als de query binnen komt. Bij *meter* worden de berekeningen voor het opslaan in de database al berekend en neemt dus meer opslag ruimte in beslag.

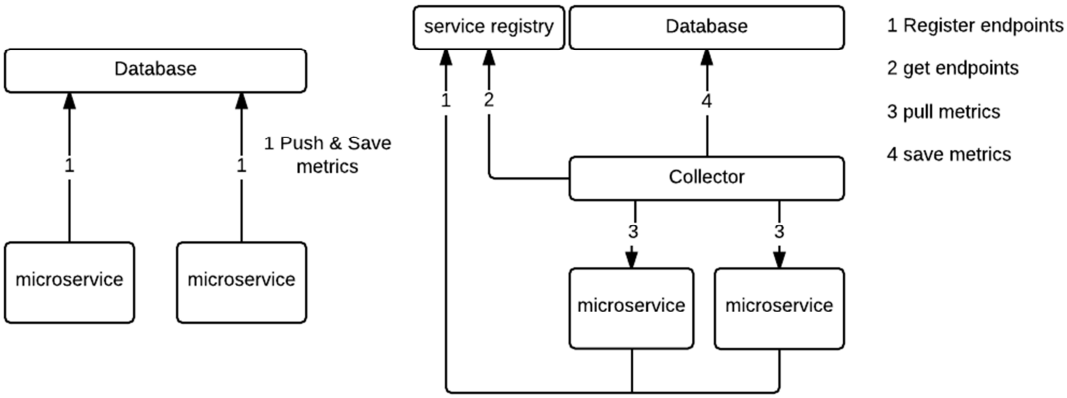
Histogram

Bij een histogram wordt de verdeling van de meetwaardes berekend. De volgende waardes worden berekend aan de hand van de verzamelde data: minimum, maximum, mediaan, gemiddelde en diverse percentiel. Ook hier kunnen de waardes met behulp van aggregaat functies van Influxdb berekend worden in combinatie met *counter.*

2.2.2 Mechanisme

Er moet rekening gehouden worden met de nieuwe situatie om de metrics te kunnen verzamelen, want dit heeft invloed op het ontwerp van het monitoringsysteem. In de huidige situatie wordt de deployment van applicaties naar de productieomgeving op een standaard manier gedaan, waarbij van te voren wordt bepaald op welke node[4] de applicatie wordt gehost. In de komende tijd wordt er gemigreerd naar een architectuur met microservices, waarbij er gebruik gemaakt zal worden van een tool (Docker wordt in het volgende hoofdstuk beschreven). Met de tool wordt de deployment van een applicatie vereenvoudigd. Uiteindelijk zal een systeem tijdens de deployment bepalen waar een applicatie wordt gehost.

De metrics van een applicatie kunnen verzameld worden met het push en pull mechanisme. In figuur 1 een schets van beide mechanisme met daarbij de nodige stappen.



Figuur 1 push (links) & pull (rechts)

Een push mechanisme is eenvoudig, want de applicatie stuurt een set van data naar een database. Een onderdeel van de applicatie is dan verantwoordelijk om dit mogelijk te maken. Het is niet nodig om te weten waar de applicatie draait en alleen de endpoint van de database moet bekend zijn in applicatie. Bij een pull mechanisme is het minder eenvoudig, want de gegevens moeten opgehaald worden door een ander component, de Collector. De verantwoordelijkheid voor het verzamelen van metrics ligt bij een pull mechanisme buiten de applicatie. De Collector moet eerst de endpoints van de applicaties weten om de metrics te kunnen verzamelen.

[4] Node - een machine waarop applicaties worden gehost

Hiervoor is er een Service Registry, waar alle applicaties zich aanmelden. De Collector haalt de endpoints van de applicaties op bij de Service Registry en kan vervolgens de metrics van de applicaties ophalen en opslaan in de database.
Om inzicht te krijgen over welke mechanisme het beste past in het project worden de voordelen en nadelen van beide mechanisme in de tabel hieronder getoond.

Push mechanisme	
Voordeel	Nadeel
Applicatie is zelf verantwoordelijk voor het beschikbaar stellen van de gegevens.	Gegevens worden op een bepaalde frequentie opgeslagen. Er is dus altijd een vertraging afhankelijk van de frequentie.
	Configuratie nodig in de applicatie

Pull mechanisme	
Voordeel	Nadeel
De functionaliteiten zijn gescheiden.	Er zijn meerder componenten nodig, want het moet bekend worden waar de applicatie draait (Service Registry) en waar de gegevens verzameld en opgeslagen kunnen worden (collector).

Voor het verzamelen en opslaan van de gegevens wordt het push mechanisme toegepast met een testproject, omdat het eenvoudiger is dan het pull mechanisme waar meerdere componenten voor nodig zijn. Hierdoor kan er sneller een prototype ontwikkeld worden. De keuze voor het pull en push mechanisme kan in de komende sprints veranderen, want ENOVATION is bezig met de ontwikkelingen van microservices.

Met de push mechanisme wordt er gewerkt aan een Spring Boot Starter dat metrics van de microservice verzameld en naar de database push. Een Spring Boot Starter is vergelijkbaar met een library dat toegevoegd kan worden bij een Spring Boot applicatie.

2.2.3 Doel van metrics

Metrics worden verzameld en opgeslagen in een database. Van sommige metrics is het alleen nodig dat er van de afgelopen zeven dagen wordt bijgehouden en van andere over een langere periode. Dit hangt af van met welke doel van de metrics worden verzameld. Aan de hand van voorbeelden wordt de doel van verzamelen van metrics duidelijk gemaakt.

Waarschuwend doeleinde

Een applicatie heeft een connection pool van maximaal tien connecties. Het monitoren van hoeveel connecties er open staan op een bepaald moment kan inzicht geven over de performance van de applicatie. Wanneer er over een lange periode de aantal open connecties boven een waarde blijft kan er geconcludeerd worden dat de connecties niet goed wordt afgesloten na gebruik. Dus wanneer het aantal connecties voor een periode boven de zeven is, dan zijn er nog maar drie connecties over. Op een gegeven moment kan dit ten koste gaan van de performance van de applicatie en zelfs leiden tot een crash. In dit geval is het bijhouden van de gegevens alleen belangrijk voor een korte periode.

Analytisch doeleinde

Van een web applicatie wordt het aantal login pogingen bijgehouden. Als de metrics bijvoorbeeld van een half jaar beschikbaar zijn, kan er geanalyseerd worden op welke dagen de web applicatie het meest gebruik wordt. Aan de hand hiervan kan er bijvoorbeeld gepland worden op welke dag een update doorgevoerd kan worden.

2.2.4 Time series Database

De metrics worden verzameld en moeten opgeslagen worden in een database, zodat er terug gekeken kan worden naar het verleden. Voor het opslaan van metrics is er een speciaal soort database beschikbaar namelijk een time serie database. Voor dit project wordt Influxdb gebruikt, omdat ENOVATION dit heeft aanbevolen. Voor het afstudeerproject is het niet belangrijk welke time serie database er gebruik wordt, omdat de opdracht gaat over het verzamelen van metrics niet het selecteren een time serie database.

ENOVATION heeft voor Influxdb gekozen, omdat het ten eerste een opensource project is met een grote community. Vergeleken met een ander time serie database zoals KDB+ waar licentie kosten aan vast zitten, is het niet gewenst. Met een grote community geeft het meer vertrouwen dat er verder aan het product wordt ontwikkeld. Daarnaast heeft Influxdb geen external dependencies nodig en kan na de installatie direct gebruikt worden. Andere time series database zoals Druid hebben external dependencies en dit is niet gewenst, omdat er meer software geïnstalleerd en onderhouden moeten worden. Een voorbeeld hiervan is dat Druid een MySQL variant gebruikt voor het opslaan van metadata. Wanneer Druid overstapt naar een ander database voor het opslaan van metadata dan moet bij een update van Druid deze ook aangepast worden.

2.2.5 Influxdb

Influxdb is een time serie database die gespecialiseerd is in het opslaan van time series. Een time serie is vergelijkbaar met een tabel in een relationele database. Het heeft altijd de kolommen time en sequence number, waarbij de kolom time geïndexeerd is. Dit zorgt ervoor dat het uitvoeren van een query met de optie “where time is …” sneller uitgevoerd kan worden. Met sequence number wordt de volgorde bepaald als er meerdere rows zijn met dezelfde waarde in time. Naast de kolommen time en sequence number zijn meestal een of meerdere kolommen voor de meetwaardes.

De naam van een time serie heeft een formaat van *<tagName>.<tagValue>.<measurement>.* Een voorbeeld hiervan is: *Server.1.response*. In de naam van de time serie moet aangegeven worden wat de data van de time series inhouden, omdat er geen ander plek is om dit aan te geven. Binnen een database zijn er dus veel time series, waarbij elke time serie een verzameling van data met tijd is. Het aantal time series dat bij een microservice hoort is afhankelijk van de functionaliteit en het aantal instanties daarvan. Er kunnen meer metrics verzameld worden van een microservice met veel functionaliteiten dan een microservice met minder functionaliteiten. Daarnaast heeft het aantal instanties van een microservice ook invloed op het totaal aantal time series, want er is gekozen om de gegevens van elk instantie in nieuwe time series op te slaan. In het volgende paragrafen wordt beschreven waarom hiervoor gekozen is.

Metrics opslaan

Er kunnen meerder instanties van een microservice gehost worden en van elk instantie zijn er metrics te verzamelen. Met het volgende voorbeeld wordt het duidelijk gemaakt, waarom het beter is om metrics van elke instantie apart op te slaan. Van een applicatie worden er twee instanties gehost, genaamd web1 en web2. Bij elk instantie wordt een counter van een methode bijgehouden en daarvan zijn de volgende gegevens bekend, zie tabel 1.

counter.web1

	t1	t2	t3	t4	t5	t6	t7
Web1	2		4		6		8

+

counter.web2

	t1	t2	t3	t4	t5	t6	t7
Web2		2		4		0	

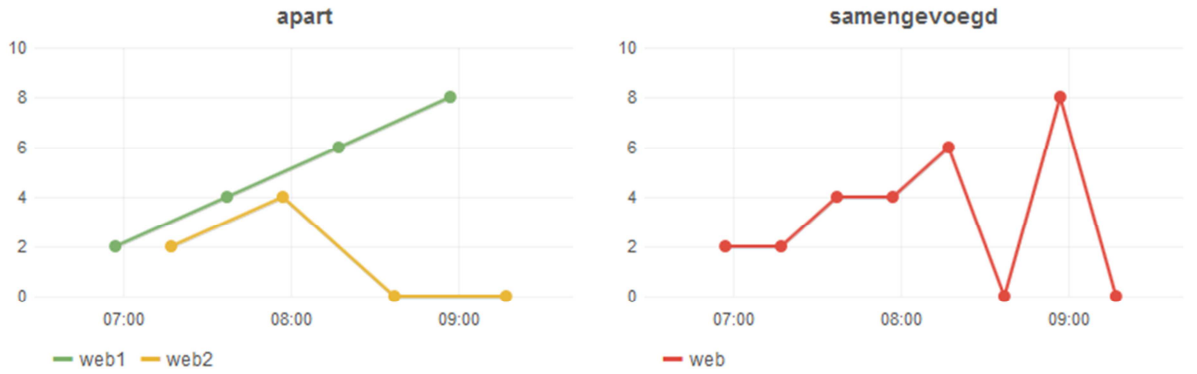
=

samengevoegd

	t1	t2	t3	t4	t5	t6	t7
Web	2	2	4	4	6	0	8

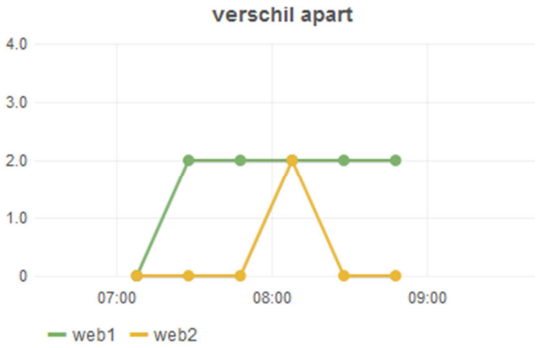
Tabel 1 gegevens van het voorbeeld

Hieronder worden de waardes weer gegeven in een grafiek. In figuur 2 links is apart te zien wat de counter is van web1 en web2. In de grafiek is te zien dat web1 een constante groei heeft en dat web2 op een gegeven moment stopt met groeien en daalt. In figuur 2 rechts is de grafiek weergegeven wanneer de waardes in een time series worden opgeslagen. Het probleem is dat de dip (zie figuur 2 rechts) een verkeerd beeld geeft, want de counter van de methode stijgt in het verloop van tijd en daalt niet. Wanneer de counter daalt, is er dus wat aan de hand met de applicatie. Dezelfde metrics van andere instanties opslaan in een time serie veroorzaakt dus een probleem, want de metrics lopen door elkaar heen en geeft een verkeerde beeld. Dit kan opgelost worden door een kolom te voegen voor het bijhouden van de instantie naam, maar dit wordt in de volgende paragraaf beschreven.



Figuur 2 de gegevens in een grafiek (link apart, rechts samen)

Een counter op een methode zou in verloop van tijd stijgen. Het visualiseren van de gegevens zou dus in een stijgende lijn resulteren in een grafiek (figuur 2 groene lijn), maar dit is niet wat een beheerder wil zien. Een beheerder wil zien hoe vaak iets aangeroepen is op een bepaald tijdstip. Om dit mogelijk te maken wordt het verschil van een punt ten opzichte van het vorige punt berekend met de aggregaat functie difference van Influxdb, zie figuur 3.



Figuur 3 het verschil t.o.v. het vorige punt

Performance

Dezelfde metrics van twee instanties kunnen in een time serie opgeslagen worden, maar dit heeft invloed op de performance. Met het volgende voorbeeld wordt het duidelijk, waarom de performance hieronder gaat lijden als de metrics van dezelfde soort van meerdere instanties in een time series worden opgeslagen. Van een applicatie worden er twee instanties gehost, waar een counter zit op de methode createUser. De gegevens worden zowel gecombineerd in een time serie als in aparte time series opgeslagen. De gecombineerde time serie heeft een extra kolom om het onderscheid te kunnen maken van welke instantie de record is, zie tabel 2.

instance1.CreateUser

time	value
t1	20
t3	40
t5	60
t7	80

+

instance2.CreateUser

time	value
t2	20
t4	40
t6	60
t8	80

=

combine.CreateUser

time	value	instance
t1	20	1
t2	20	2
t3	40	1
t4	40	2
t5	60	1
t6	60	2
t7	80	1
t8	80	2

Tabel 2 links en midden apart en rechts gecombineerd

Bij het uitvoeren van een select query wordt het voordeel van apart opslaan duidelijk. Een select query uitvoeren om de CreateUser van instance1 te tonen duurt langer bij een gecombineerde time serie dan bij een aparte time serie, want er worden meer vergelijking uitgevoerd. De query voor de gecombineerde time serie ziet er als volgt uit: "*Select * from combine.CreateUser where instance=1".* Bij de aparte time serie ziet de query er als volgt uit: "*Select * from instance1.CreateUser"*. De query duurt bij de *combine.CreateUser* langer dan bij *instance1.CreateUser*, want een full range scan wordt uitgevoerd om alleen de record van instance 1 op te halen. Het uitlezen bij de time serie instance1.CreateUser is dus sneller.

Het opslaan van gegevens in een time serie met meerdere instanties veroorzaakt een probleem en kan opgelost worden door een extra kolom toe te voegen. Een betere oplossing is door de gegevens van elke instantie apart op te slaan, hierdoor kan de database beter presteren bij het uitvoeren van een select query.

Testen in de praktijk

In de vorige paragraaf is beschreven dat de database beter presteert als de metrics van twee instanties niet worden gecombineerd in een time serie, want alleen de kolom time is geïndexeerd. Volgens de theorie (Tré, 2007) zorgt het gebruiken van een index dat het ophalen van gegevens sneller is. Om de performance winst te kunnen meten heb ik een test opgezet waarbij de database twee time series bevatten. In de ene time serie zijn er metrics van twee instanties met een extra kolom om herkomst van de data te kunnen achter halen. De andere time serie bevat alleen de metrics van één instantie.

In de test wordt een query uitgevoerd voor het ophalen van data van zeven dagen geleden van applicatie1. Aan de hand van de respons time kan er geconcludeerd worden of er daadwerkelijk een performance winst is. Voor het uitvoeren van de query is er gekozen om het met de command line tool “cURL” uit te voeren, omdat dit de meest eenvoudigste manier is en de data wordt niet gevisualiseerd (wat tijd kost). In de database is er test data toegevoegd met een interval van 10 seconden gedurende zeven dagen. In tabel 3 en 4 is een klein deel van de data te zien. In tabel 5 zijn de response time weergeven bij het uitvoeren van een query dat de gegevens ophaalt van de afgelopen zeven dagen van hostName1.

De volgende queries zijn gebruikt: Select * from single
Select * from combine where hostName=’hostname1’

In tabel 5 is duidelijk te zien dat er een groot verschil is in de response time van de twee queries. Het apart opslaan van de metrics zorgt dus dat het op het ophalen sneller gebruikt dan wanneer het gecombineerd is.

time	sequence_number	value
1433235123666	42392030001	63.199381129717935
1433235113666	42392040001	4.993254893693511
1433235103666	42392050001	56.14172115931908
1433235093666	42392060001	45.7411082396271

Tabel 3 Apart (single)

time	sequence_number	value	hostName
1429280010120	8975640001	55.677627651708995	hostname2
1429280010119	8975650001	55.677627651708995	hostname1
1429280000120	8975660001	47.49095156235412	hostname2
1429280000119	8975670001	47.49095156235412	hostname1

Tabel 4 Gecombineerd (combine)

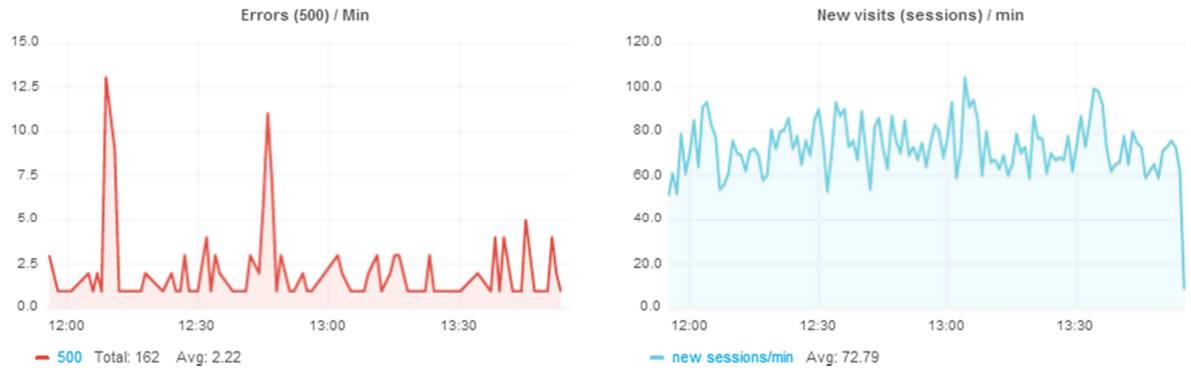
Poging	Apart	Gecombineerd	Factor
1	1.256	2.558	2.03
2	1.258	2.905	2.30
3	1.245	2.547	2.04
4	1.354	2.493	1.84
5	1.378	2.494	1.80

Tabel 5 Response tijden (in seconden)

2.2.6 Grafana

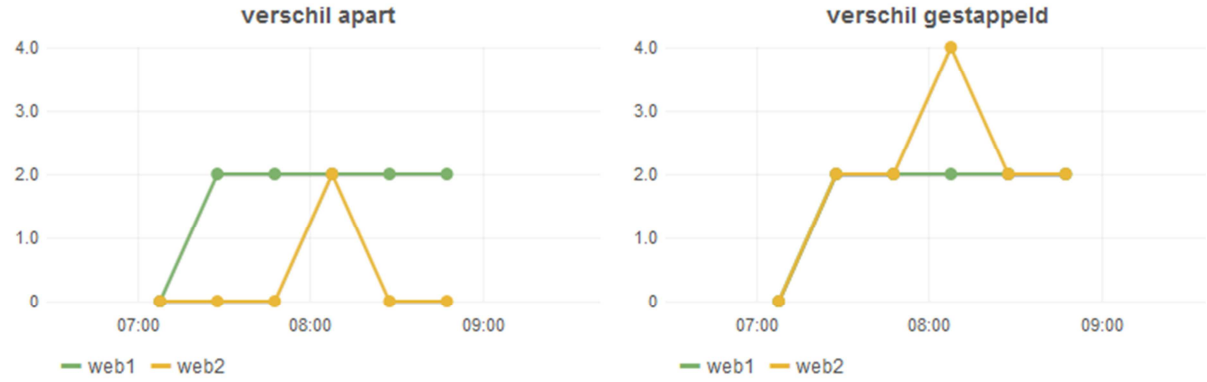
Na het verzamelen en opslaan van metrics is de volgende stap het visualiseren van de data. In de documentatie van Influxdb is er een verwijzing naar Grafana voor het tonen van metrics. Na het uitproberen van de live demo omgeving van Grafana blijkt dat het een zeer geschikte tool is met uitgebreide functionaliteiten. Daarnaast heeft ENOVATION ook aangeraden om Grafana te gebruiken voor het maken dashboards, wat tevens de doorslaggevende reden is geweest om deze applicatie te gebruiken.

Grafana maakt het mogelijk om de gegevens van Influxdb te kunnen tonen op een dashboard, een pagina met grafieken over een applicatie, zie figuur 4. Het is gemakkelijk om een dashboard te configureren, omdat alle configureerwerk via de interface wordt gedaan. Om de metrics te kunnen tonen worden queries geconfigureerd. De vormgeving van de grafieken kunnen eenvoudig aangepast worden via de user interface.



Figuur 4 Voorbeelden van grafieken in Grafana[5]

In de voorgaande paragraaf heb ik beschreven over het verschil tonen t.o.v. het vorige punt, waarbij er twee instanties van een applicatie in een grafiek worden getoond zie figuur 5 links. Voor een beheerder is het handiger als de punten van de twee instanties bij elkaar opgeteld worden om het totaal beeld te kunnen zien. Door de optie *stack* aan te vinken in Grafana kan het gedaan worden, zie resultaat figuur 6 rechts.



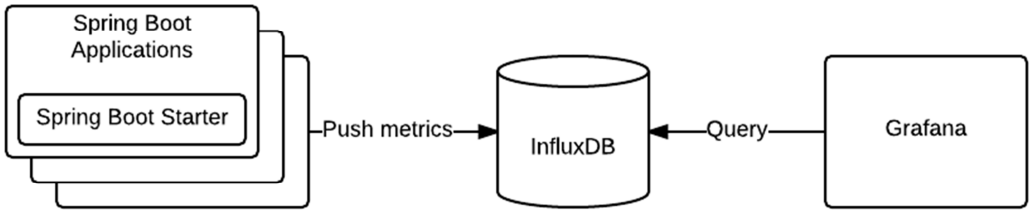
Figuur 5 links apart en rechts gestapeld

[5] Bron – http://grafana.org/

In een grafiek kunnen meerdere queries toegevoegd worden. Dit wordt gebruikt om het verschil van de waardes te kunnen zien. Een voorbeeld hiervan is de http status codes zoals: 200 ok, 400 bad request en 404 not found. Deze worden allemaal in een aparte timer serie opgeslagen. Het is dus handig als de time series die bij elkaar horen in een grafiek getoond kunnen worden. Het toevoegen van een query is eenvoudig, want Grafana heeft een hulpmiddel bij het maken van een query. Het nadeel hiervan is dat een time series pas zichtbaar is wanneer het minimaal een keer is voorgekomen. Het is dus alleen mogelijk om query toe te voegen van een time serie dat bestaat. Een ander oplossing om meerdere time series te tonen in een grafiek is door reguliere expressie te gebruiken in een query. In een grafiek waar de http status codes van een Rest Call methode worden getoond is beter om reguliere expressie te gebruiken in de query, omdat er meerdere time series geselecteerd kunnen worden. Wanneer er een nieuwe time series erbij komt die voldoet aan de reguliere expressie dan wordt deze ook geselecteerd en getoond in de grafiek. Bij een normale query kan het pas toegevoegd worden als het een keer voorgekomen is en dit vereist handmatige handelingen.

2.2.7 Resultaat

Aan het einde van de sprint is er een Spring Boot Starter[6] (prototype) opgeleverd, waarbij het mogelijk is om metrics van Spring Boot applicatie op te slaan in de database. De Spring Boot Starter is in een eenvoudig test project geïmplementeerd om metrics te kunnen verzamelen. Er is een keuze gemaakt om alle metrics op te slaan dat met Spring Boot Actuator opgehaald kan worden. In de bijlage E is een voorbeeld van metrics te zien in JSON formaat. Na het opslaan van metrics kan het vervolgens gevisualiseerd worden met Grafana. In figuur 6 is een schets te zien van de huidige situatie.



Figuur 6 schets huidige ontwerp

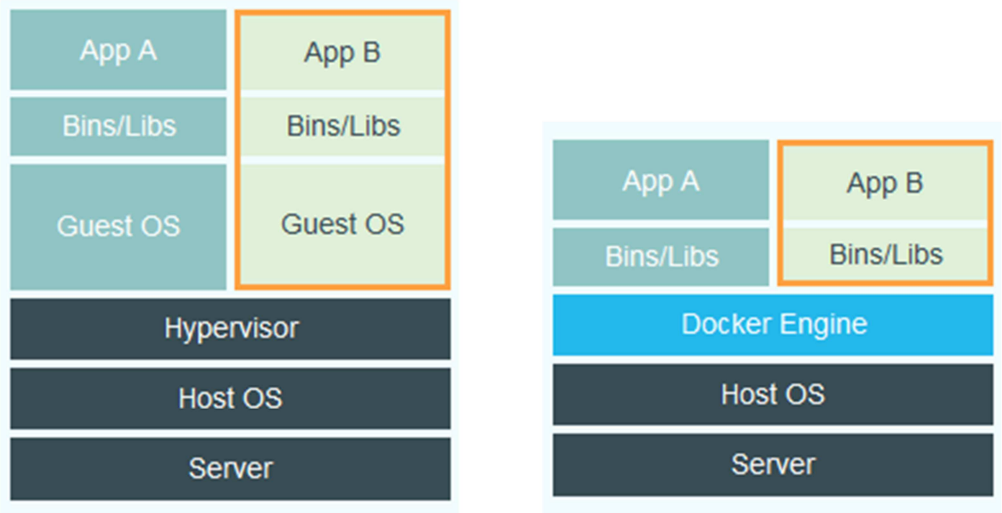
[6] Spring Boot Starter – Een verzameling van dependencies voor specifieke functionaliteiten en configuraties

2.3 Sprint 3 - Omgaan met meerdere instanties

Deze sprint is bedoeld om de monitor component (Spring boot starter) te implementeren in een onderdeel van een bestaand systeem. Hieruit moet blijken welke functionaliteiten toegevoegd moet worden, zodat de spring boot starter geschikt is voor productie. Tijdens de ontwikkeling is er nagedacht over de daadwerkelijke werking met microservices. Dit heeft geleid tot een andere aanpak in het project.

2.3.1 Docker

ENOVATION gebruikt Docker voor het draaien van nieuwe applicaties. Met Docker kunnen applicaties geïsoleerd draaien in hun eigen container op de Docker Engine. Een container bevat meestal een applicatie met al zijn dependencies. Het voordeel van een container is dat er meerdere gemaakt en geïsoleerd gestart kunnen worden. Zo is er geen conflict tussen de dependencies van de applicaties. Het is dus mogelijk dat bij de ene container php 5 draait terwijl een andere container php 6 draait. Docker is vergelijkbaar met Virtual Machine maar heeft minder overlap, omdat de Guest OS niet nodig is, zie figuur 7.



Figuur 7 links Virtual Machine, rechts Docker[7]

Container levensduur

Het maken en starten van een container kan binnen enkele seconden gedaan worden. De bedoeling is dat de containers worden vervangen als er een nieuwe versie van de applicatie komt. Na het aanmaken van een container kan deze verder niet meer aangepast worden. De container wordt dus niet meer geupdate maar vervangen.

Data volume

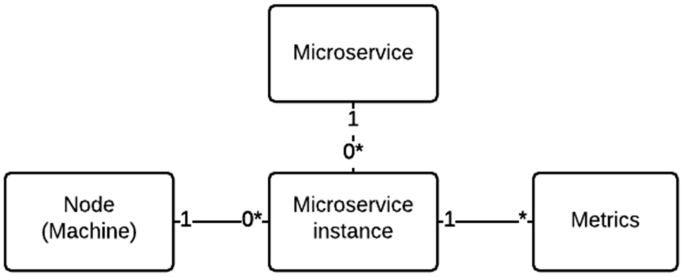
Docker houdt standaard de data in de container wanneer er geen parameters worden meegegeven bij het opstarten van een container. Het verwijderen van de container zorgt er ook voor dat de data verwijderd wordt. Dit is niet handig als de data van een database wordt verwijderd. Om dit op te lossen is de documentatie van Docker bestudeerd om een oplossing te vinden om data buiten de container op te slaan. Er zijn twee oplossingen gevonden om dit probleem op te lossen. De eerste oplossing is door een data container te gebruiken, een lege container met alleen data. Deze kan dan vervolgens gekoppeld worden aan een container.

[7] Bron - https://www.docker.com/whatisdocker/

De tweede oplossing is door gebruik te maken van een folder van de Host en deze te koppelen met de container. De data wordt dan in de gekoppelde folder opgeslagen. Er is gekozen om de folder van Host te koppelen met de container, want de data staat dan lost van Docker. Daarnaast is de kans op het per ongeluk verwijderen van een data container groot, want een data container kan eenvoudig verwijderd worden.

2.3.2 Naamconventie

Van een applicatie kunnen er meerder instanties gedraaid worden. Dit wordt gedaan wegens de schaalbaar en beschikbaarheid van de applicatie. Van elk instantie worden er metrics opgeslagen in de database en elke instantie wordt gehost op een node.



Figuur 8 Metricsobject Model

Wegens performance redenen wordt per instantie per variabel in een time serie opgeslagen (zie sprint 2). Om dit te kunnen doen moeten de time series een unieke naam hebben. Een voorbeeld van een time serie naam is:

SampleApplication.counter.status.200.metrics.count

De bedoeling is dat de naam van de time serie uniek wordt, zodat het te traceren van welke instantie de metrics zijn. Een mogelijke oplossing is door een instantienummer te voegen wat als volg uit ziet:

*SampleApplication.**instance1**.counter.status.200.metrics.count*

*SampleApplication.**instance2**.counter.status.200.metrics.count*

Met de genoemde oplossing is het niet mogelijk om te achter halen op welke node de instantie wordt gehost. Als de metrics aantonen dat de CPU de afgelopen periode zwaar belast is dan is het belangrijk om te weten op welke node de instantie draait, zodat er handelingen uitgevoerd kunnen worden op de node. Bijvoorbeeld het verplaatsen van de instantie naar een ander node waar de CPU minder wordt belast. Naast het willen weten van welke instantie de metrics komen is ook de locatie waarop de applicatie draait belangrijk. Er wordt dus gezocht naar een naam dat uniek is, aangeeft van welke instantie het komt en aangeeft op welke node de instantie draait.

Met IP adres en port nummer

Er is gekeken naar hoe bestaande monitoringsystemen dit probleem aanpakken. Een van de oplossingen is het gebruiken van IP adres gecombineerd met de port nummer wat als volg uitziet in het project:

***1.2.3.4:80**.SampleApplication.counter.status.200.metrics.count*

***1.2.3.4:81**.SampleApplication.counter.status.200.metrics.count*

Dit is een goede oplossing, want met de IP adres kan er achterhaald worden waar de machine is en met het portnummer kan er onderscheid gemaakt worden als de instanties op dezelfdemachine draait. Het nadeel is dat de IP adres kan veranderen als de instantie wordt herstart.

Dit zorgt ervoor dat er nieuwe time series worden aangemaakt. Op ten duur zijn er een groot aantal time series die niet meer gebruikt worden en geen gegevens bevatten, deze kunnen dan verwijderd worden.

Met node en container naam

Een ander oplossing is door de combinatie van node en container naam te gebruiken, want dit geeft meer controle op de naamgeving en voldoet aan de gestelde eisen. Een container naam is altijd uniek, want binnen Docker kan er geen container aangemaakt worden als de naam al in gebruik is. Voorbeelden van node en container naam combinatie zijn:

Node1.SampleService.*SampleApplication.counter.status.200.metrics.count*

Node2.SampleService.*SampleApplication.counter.status.200.metrics.count*

Met deze oplossing is het direct aan de naam te zien op welke node en van welke instantie de gegevens komen. Daarnaast kan een time serie opnieuw gebruikt worden en dus blijven het aantal time series beperkt. Het nadeel hiervan is dat de time series oude en nieuwe metrics bevatten. Om het te kunnen onderscheiden kan er bij gehouden worden wanneer een container gestopt en gestart is.

Om de time series van een unieke naam te voorzien wordt de oplossing toegepast met node en container naam, omdat het gemakkelijker te onderhouden is. Met IP adres en port nummer komen er op ten duur een groot aantal time series in de database terecht. Het probleem hierbij is dat het een zoek werk is om te achter halen welke time series van de nieuwe en actieve containers zijn. Wanneer er in Grafana een query in het dashboard aangepast wordt, is het vinden van de juist time series lastig, omdat er een groot aantal time series zijn. Met node en container naam blijf het aantal time series beperkt, omdat de time series opnieuw gebruikt worden. Hierdoor bevatten de time series metrics van oude container en nieuwe container. Het aanpassen van een dashboard in Grafana is dus ook eenvoudiger en in sommige gevallen niet nodig.

2.3.3 Verandering van mechanisme

Er is verder nagedacht over de werking van microservices bij het zoeken naar een geschikte naam conventie voor de time series. Dit heeft er tot geleid dat metrics niet meer met het push mechanisme wordt verzameld maar met het pull mechanisme. Hieronder wordt beschreven waarom er van mechanisme is veranderd.

Ten eerste zit de verantwoordelijkheid niet meer in de applicaties maar in een andere service. Dit beïnvloedt de deployment van het monitoringsysteem. Met het push mechanisme zit er een component in de applicatie dat gemonitord wordt. Bij het updaten van een nieuwe versie van de component heeft het effect op alles applicaties dat gemonitord wordt en dit is niet gewenst, omdat het veel werk vereist om alle applicatie te moeten updaten met de nieuwe component. Een voorbeeld daarvan is dat de applicatie opnieuw getest moet worden met de nieuwe component. Daarnaast zit ook de configuratie niet meer aan de kan van de gemonitorde applicaties, maar bij de services dat verantwoordelijk is voor het verzamelen van metrics. Dit zorgt ervoor dat er op een plek alle configuraties gedaan kan worden.

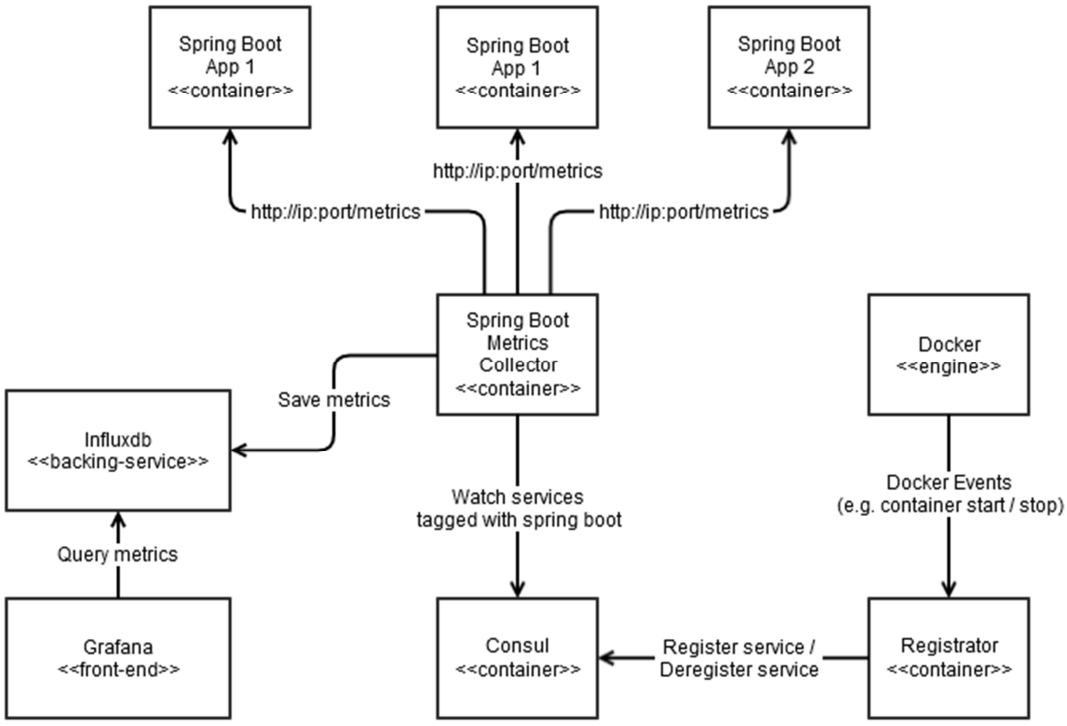
Ten tweede heeft een microservice in principe een specifieke functionaliteit. Door het pull mechanisme toe te passen sluit het beter aan met de microservices architectuur. Hierdoor zijn de functionaliteiten gescheiden en is het gemakkelijker om de monitor component aan te passen.

Tenslotte is het met het push mechanisme niet eenvoudig te achter halen waar de metrics vandaan komen, omdat er geen componenten zijn die het registeren. Hiervoor moet er bij het aanmaken van een container aangegeven worden welke instantie het is en op welk node het draait. Het ip adress en portnummer combinatie kan niet toegepast worden, omdat deze pas bekend is wanneer de container actief is en als de container eenmaal actief is kan deze niet meer aangepast worden.

2.3.4 Het nieuwe ontwerp

Een nieuwe ontwerp is nodig, omdat er een keuze gemaakt is om van het push mechanisme over te stappen naar het pull mechanisme. Het nieuwe ontwerp is te zien in figuur 9 en oude ontwerp in figuur 6 . Om het pull mechanisme te kunnen realiseren zijn er meer componenten nodig dan bij het push mechanisme. Het nieuwe ontwerp bestaat uit de volgende componenten:

- Spring Boot Metrics Collector: verantwoordelijk voor het verzamelen en opslaan van metrics. Deze wordt ontwikkeld in de komende sprints.
- Registrator: een tool dat Docker container events registreert bij een Service Registry
- Consul: een Service Registry, hier worden alle services geregistreerd
- Influxdb: een time serie database
- Grafana: web front, het tonen van metrics op een dashboard



Figuur 9 Schets van nieuwe situatie met pull mechanisme

De Spring Boot Metrics Collector verzamelt metrics van Spring Boot Applicaties. Om dit te kunnen doen moet het bekend zijn wat de endpoints zijn. De Spring Boot applicaties draaien in een Docker container en wanneer een container stop of start moet de Spring Boot Metrics Collector hiervan op de hoogte zijn.

Met Registrator worden de events van een container gemonitord en geregistreerd bij een Service Registry. De Service Registries die Registrator ondersteunt zijn Consul en Skydns. ENOVATION heeft gekozen voor Consul als Service Registry, daarom wordt Consul gebruikt bij het nieuwe ontwerp. Een van de functionaliteit die Consul biedt is een uitgebreide health check waar gecontroleerd wordt op heartbeat[8] en meer. Daarnaast heeft het ook een userinterface waar een overzicht van alle geregistreerde services getoond worden. Bij Consul worden dus de endpoints van applicaties dus geregistreerd. De Spring Boot Metrics Collector haalt de endpoints op bij Consul en kan vervolgens de metrics ophalen bij de Spring Boot applicatie en opslaan in de Influxdb.

2.3.5 Resultaat

Metrics van meerdere instanties kunnen opgeslagen worden, waarbij de namen van de time series aangeven waar de metrics vandaan komen. Er is van push mechanisme naar pull mechanisme veranderd voor het verzamelen van metrics. Hiervoor is de Spring Boot Metrics Collector ontwikkeld dat verantwoordelijk is voor het verzamelen en opslaan van de metrics. De Spring Boot Metrics Collector bevat aan het einde van de sprint de kern functionaliteiten zoals het kunnen verzamelen en opslaan van metrics, maar op een aantal plekken kan het op een andere manier geïmplementeerd worden. In de volgende sprint wordt er gewerkt aan een betere oplossing.

[8] Heartbeat – op interval de applicatie op de hoogte houden dat het actief is

2.4 Sprint 4 - Spring Boot Metrics Collector verder uitwerken

Deze sprint heeft als doel om de Spring Boot Metrics Collector verder uit te werken. Er is als eerste gekeken naar het verzamelen van metrics op container niveau. Daarnaast zijn bestaande functies geoptimaliseerd, zodat het beter functioneert. Vervolgens is er gewerkt aan om de Spring Boot Metrics Collector in een Docker container te draaien. Tenslotte zijn er unit test en integratietest gemaakt voor de Spring Boot Metrics Collector.

2.4.1 cAdvisor

Metrics kunnen op verschillende niveaus verzameld worden. In de vorige sprints ben ik voornamelijk bezig geweest met metrics op applicatie niveau. In deze sprint heb ik gekeken naar metrics op container niveau. ENOVATION heeft mij aanbevolen om cAdvisor te bekijken, want met cAdvisor kunnen er metrics op container niveau verzameld worden. Daarnaast biedt het ook ondersteuning voor Influxdb, wat aansluit met wat er gebruik wordt. Enkele voorbeelden van metrics op container zijn: aantal bytes verzonden en ontvangen, geheugen en processor gebruik.

Er is gekeken naar de manier waarop de metrics in de database worden opgeslagen. Daaruit is gebleken dat de metrics van alle containers in een time serie worden opgeslagen, waarbij een extra kolom wordt gebruikt om de herkomst van de metrics bij te houden. Dit is niet efficiënt (zie sprint 2) wanneer de metrics worden opgehaald met een query, want alleen de kolom time is geïndexeerd. Een query van Influxdb heeft meestal een conditie met time voor het selecteren van metrics van afgelopen uren, dagen, weken enz. Een andere conditie zoals: where container_name =”name” is dus minder effectief en duurt langer bij het uitvoeren daarvan, omdat een range scan wordt uitgevoerd op een grote verzameling van data. De condities kunnen ook gecombineerd worden, maar er wordt dan nog steeds een range scan uitgevoerd. Ik heb mijn opdrachtgever wegens de performance geadviseerd om cAdvisor niet samen te gebruiken met Influxdb, maar cAdvisor wel apart te gebruiken om de container real time te monitoren.

2.4.2 Consul blocking queries

Om de metrics van een Spring Boot applicatie te kunnen ophalen moet de endpoint van applicatie bekend zijn. Bij het starten van een container wordt de endpoint geregistreerd bij Consul (Service Registry). De Spring Boot Metrics Collector haalt de endpoints van Spring Boot Applicaties op bij Consul en kan vervolgens de metrics van applicaties ophalen.

In de vorige sprint is het ophalen van de endpoint zo geïmplementeerd dat het elk minuut alle endpoints bij Consul worden opgehaald. Dit is niet efficiënt, want het gebeurt niet vaak dat er een instantie van een Spring Boot Applicatie stopt of start op dag basis. Hierdoor worden er onnodige netwerkverkeer en CPU belasting gegenereerd. Om dit efficiënter te maken worden blocking queries toegepast. Bij een blocking query kunnen er parameters meegeven worden, deze zijn index en wait. Hiermee kan er aangegeven worden welke index de aanvrager nu heeft en hoe lang het blijft wachten op een verandering. Wanneer de meegegeven index gelijk is aan de index van Consul dan wordt er gewacht tot de wait voorbij is. Als er tussen de wait een verandering plaats vindt dan krijgt de aanvrager een antwoord terug en bij geen verandering krijgt de aanvrager een antwoord terug na de wait. Wanneer de meegegeven index niet gelijk is aan de index van Consul, dan krijgt de aanvrager direct een antwoord terug. Door blocking queries toe te passen is het efficiënter dan op interval alle endpoints ophalen, want de aanvrager krijgt alleen een antwoord terug als er een verandering heeft plaats gevonden. Dit bespaart onnodige netwerkverkeer en CPU belasting.

Concurrency

Er wordt een blocking query gebruikt bij het ophalen van endpoints van Spring Boot applicaties. Bij een blocking query wordt de parameter wait meegegeven. Dit zorgt ervoor dat er na de aangegeven tijd een antwoord terug komt als er geen veranderingen zijn. Afhankelijk van de parameter wait kan het dus lang of korte duren. Om dit af te handelen wordt het uitvoeren van een blocking query in een apart thread gedaan, zodat de applicatie niet op een antwoord van de blocking query hoeft te wachten. Hierdoor zijn er twee threads die dezelfde variabele benaderen. De ene thread wil schrijven terwijl de ander het wil lezen. Om te voorkomen dat de (read) thread met een corrupte variabele werkt, wordt er een kopie gemaakt.

2.4.3 Dockerizing

De Spring Boot Metrics Collector kan de metrics van Spring Boot applicaties verzamelen en opslaan in de database. Vervolgens kunnen de metrics getoond worden met Grafana. De volgende stap is ervoor zorgen dat de applicatie op Docker gedraaid kan worden. ENOVATION gebruikt Docker voor het draaien van applicaties. Om een applicatie te kunnen draaien met Docker moet er een Docker Image gemaakt worden. Met een Docker Images kan er vervolgens Docker containers aangemaakt worden. In een Docker container draait de applicatie geïsoleerd van andere containers.

Er zijn een aantal manieren om een Docker Image te maken. De eerste manier is door een Dockerfile te maken. Een Dockerfile is een tekst document waarin alle commando's staan gedefinieerd voor het maken van een Docker Image. De Dockerfile kan vervolgens uitgevoerd worden met het commando: "docker build" op de terminal (computer terminal). De tweede manier om een Docker Image te maken is door gebruik te maken van de build tool Maven. Er is een Maven plug-in die een Docker Image kan maken en ook een Docker Container kan starten en stoppen tijden het uitvoeren van testen, wat handig is voor een integratietest. Er is gekozen om de Docker Images is maken via de build tool Maven, omdat het beter aansluit bij de build-omgeving van ENOVATION. Hierdoor kunnen de handelingen geautomatiseerd worden met de build tool Maven.

2.4.4 Unit test en integratietest

Er zijn unit test gemaakt voor de kern functionaliteiten van de Spring Boot Metrics Collector, om er zeker van te zijn dat de code goed functioneert. Naast het maken van unit test is er ook een integratietest gemaakt om de Spring Boot Metrics Collector te testen met een echte database. Voor de integratietest is er gebruik gemaakt van de Docker Maven plug-in. Hiermee kan er Docker container met Influxdb aangemaakt worden bij het uitvoeren van integratietest. Bij het uitvoeren van de integratie test wordt er gecontroleerd of de Spring Boot Metrics Collector metrics in de database opslaan. Daarnaast is er een Jenkins[9] Job aangemaakt dat diverse taken uitvoert wanneer er code worden geüpload naar de repository, zodat de handelingen geautomatiseerd worden. Een van de taken die uitgevoerd wordt door Jenkins is het uitvoeren van alle testen.

2.4.5 Resultaat

Voor het ophalen van services bij Consul wordt blocking query toegepast. Dit is een efficiënter manier dan op interval de services op te halen. Daarnaast zijn er testen gemaakt om de code te testen. Aan het einde van deze sprint is er een prototype opgeleverd dat in Docker gehost kan worden.

[9] Jenkins – Continuous integration server, een server die het project bouwt en andere taken uitvoert.

2.5 Sprint 5 – Refactor code en Systeemtest ontwerpen

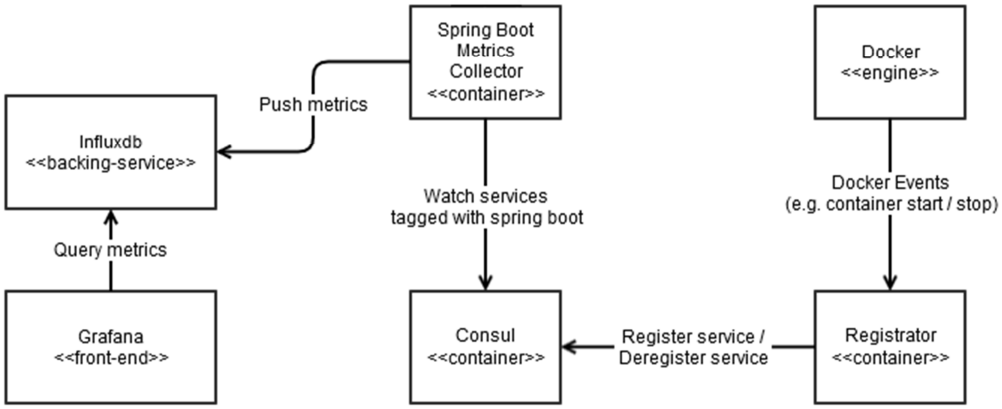
Het doel van deze sprint is om de code van Spring Boot Metrics Collector te refactoren, zodat het gemakkelijker is om nieuwe functionaliteiten toe te voegen en bestaande code opnieuw te gebruiken. Daarnaast wordt ook de systeemtest gemaakt en uitgevoerd.

2.5.1 Refactor code

De Spring Boot Metrics Collector kan metrics van Spring Boot applicaties verzamelen en opslaan in de database. Het is geïmplementeerd met Influxdb als database en Consul als Service Registry. Het zou in de toekomst kunnen voorkomen dat er een ander database wordt gebruikt. Een voorbeeld van een ander time serie database is bijvoorbeeld Druid. Om gemakkelijker van implementatie te kunnen wisselen is er een interface gemaakt voor de Influxdb. Daarnaast is er ook een interface gemaakt voor Consul, want ook deze kan in de toekomst veranderen, zie bijlage F klassendiagram.

2.5.2 Systeemtest

Het monitoringsysteem bestaat uit een aantal componenten, deze zijn: Metrics Collector, Influxdb en Grafana. Daarnaast is het monitoringsysteem afhankelijk van Consul, want Consul wordt gebruikt voor het bijhouden van de endpoints van de Docker containers. Consul is een infrastructuur component dat ook gebruik wordt voor andere applicaties. Het monitoringsysteem is niet afhankelijk van Registrator, maar het wordt wel gebruikt bij het uitvoeren van systeemtest, omdat services dan automatisch geregistreerd worden bij Consul. Zonder Registrator moet een service handmatig toegevoegd worden bij consul en hierdoor kunnen er fouten ontstaat. In figuur 10 is een eenvoudig weergave te zien van wat de testbasis is. Het uitvoeren van de systeemtest wordt over meerder servers gedaan in de testomgeving, omdat het dan overeenkomt met de productieomgeving, zie bijlage G Architectuur Productieomgeving.



Figuur 10 testbasis

Voor het uitvoeren van systeemtest is er gekozen voor exploratory testing, omdat er meer getest kan worden dan de hiervoor bedachte logische testgevallen. Er kunnen nieuwe test scenario's ontdekt worden waar van te voren niet aan gedacht is. Deze kunnen dan ook direct uitgevoerd worden. Daarnaast is het doel om het systeem in zijn geheel te testen, waarbij de focus ligt op de happyflow[10]. Naast de happyflow is het ook belangrijk dat de badflow[11] wordt getest. Met kennis van het gedrag van het systeem kan dit gemakkelijker uitgevoerd worden.

[10] Happyflow – onder normale omstandigheden
[11] Badflow – situaties die normale in normale omstandigheden niet voorkomen

In een van de testcases worden de metrics gecontroleerd. Voor een eenvoudig Spring Boot applicatie zijn er al meer dan 20 metrics. Er worden maar drie metrics gekozen waarover controles worden uitgevoerd, want van drie specifieke metrics zijn de karakteristieke eigenschappen bekend. Hierdoor is het gemakkelijker te controleren. Daarnaast is de verwachting dat als er bij een aantal metrics goed zijn dat de overige metrics ook goed zijn.

2.5.3 Logische test gevallen

De volgende testcases zijn bedacht voor het uitvoeren van exploratory testing. Met als beginpunt dat alle componenten van het monitoringsysteem gestart zijn met correct ingevulde parameters. Wanneer alle componenten van het monitoringsysteem gestart zijn dan kunnen de testcases uitgevoerd worden.

Testcase	Actie	Verwacht uitkomst
1	Controleer de /metrics endpoint van de Spring Boot Metrics Collector.	Een JSON output met metrics
2	Controleer de endpoint van consul. */v1/catalog/node/<nodename>* Gebruik voor nodename de naam van de node waar de Spring Boot Metrics Collector.	Een JSON output met services. Een van de services is van Spring Boot Metrics Collector.
3	Run een tweede instantie van Spring Boot Metrics Collector. Stop vervolgens na een korte periode(10 sec) de tweede instantie.	In de log van de eerste Spring Boot Metrics Collector is te zien dat de services zijn verandert, twee keer.
5	Controleer de database endpoint van Influx door in te loggen op de userinterface.	Na het inloggen ben je op de pagina waar een lijst van database worden getoond.
6	Controleer of metrics van de Spring Boot Metrics Collector worden opgeslagen door de volgende query uit te voeren op de database: list series	Een lijst met time series wordt getoond op de pagina.
7	Gebruik de userinterface van Influxdb om een query uit te voeren waar de metric counter.status.200.metrics.count wordt opgehaald	De value van de counter stijgt in de verloop van de tijd en een stijgende lijn is te zien in de grafiek.
8	Maak in Grafana een grafiek met de volgende metrics: -status.200.metrics.count -heap -heap.used	 -Een stijgende lijn in de grafiek. -Een stil liggende lijn. -Een zig zag lijn.

2.5.4 Automatisering

Voor het opzetten van het monitoringsysteem is het vereist dat er een aantal componenten worden gestart. Hiervoor is er samen met een ander ontwikkelaar een script gemaakt, waarbij de handelingen worden geautomatiseerd. De script is gemaakt, zodat het monitoringsysteem eenvoudig op te zetten is in een ander omgeving. Hierdoor zijn de omgevingen op dezelfde manier geconfigureerd bij het uitvoeren van de script.

Bij het uitvoeren van de script is er een bug gevonden dat te maken heeft met de volgorde waarin de componenten worden gestart. Door kleine aanpassingen te maken in de Spring Boot Metrics collector is het probleem opgelost. Hierna was het mogelijk om het systeemtest uit te voeren.

2.5.5 Fysieke testgevallen

Testcase	Actie	Verwacht uitkomst	Komt overeen
1	Controleer de /metrics endpoint van de Spring Boot Metrics Collector.	Een JSON output met metrics	**Ja**
2	Controleer de endpoint van consul. */v1/catalog/node/<nodename>* Gebruik voor nodename de naam van de node waar de Spring Boot Metrics Collector.	Een JSON output met services. Een van de services is van Spring Boot Metrics Collector.	**Ja**
3	Run een tweede instantie van Spring Boot Metrics Collector. Stop vervolgens na een korte periode(10 sec) de tweede instantie.	In de log van de eerste Spring Boot Metrics Collector is te zien dat de services zijn verandert, twee keer.	**Ja**
5	Controleer de database endpoint van Influxdb door in te loggen op de userinterface van Influxdb.	Na het inloggen ben je op de pagina waar een lijst van database worden getoond.	**Ja**
6	Controleer of metrics van de Spring Boot Metrics Collector worden opgeslagen door de volgende query uit te voeren op de database: list series	Een lijst met time series wordt getoond op de pagina.	**Ja**
7	De userinterface van Influxdb gebruiken om een query uit te voeren waar de metric counter.status.200.metrics.count wordt opgehaald.	De waarde van count stijgt in de verloop van de tijd en een stijgende lijn in de grafiek.	**Ja**
8	Maak in Grafana een grafiek met de volgende metrics: -status.200.metrics.count -heap -heap.used	 -Een stijgende lijn in de grafiek. -Een stil liggende lijn. -Een zig zag lijn.	**Ja**

2.5.5 Resultaat

Er is een kleine bug gevonden bij het opzetten van de testomgeving. Na het oplossen van de bug kon de systeemtest uitgevoerd worden. Uit de systeemtest is gebleken dat de al vooraf bedachte logische testgevallen geslaagd zijn. Er zijn verder geen bugs meer gevonden, maar wel een opmerking over de naam van de node. Deze wordt gebruikt bij de naamgeving van de time series. De naam dat gebruikt wordt voor de node is een gegenereerde naam en het moet in de script aangepast worden, zodat het gemakkelijker aan de naam te zien welke node het is. Naast het uitvoeren van de systeemtest is er ook een dashboard gemaakt voor Spring Boot applicaties die in de demo omgeving gehost worden. Al met al functioneert het monitoringsysteem naar behoren en kan in gebruik genomen worden.

Deel III. Evaluatie

In dit deel van het verslag wordt er geëvalueerd op de product, proces en de beroepstaken.

3.1 Productevaluatie

Aan het einde van het afstudeerproject heb ik een systeem kunnen opleveren dat voldoet aan de wensen van ENOVATION. Het systeem bestaan uit drie onderdelen namelijk, Spring Boot Metrics Collector, Influxdb en Grafana. Spring Boot Metrics collector is ontwikkeld in de afgelopen sprints en is verantwoordelijk voor het verzamelen van metrics van Spring boot applicaties. De metrics worden vervolgens opgeslagen in Influxdb. Hierna kunnen de metrics op de dashboard van Grafana getoond worden.

Ik ben zeer tevreden met het opgeleverde systeem, want het is een mooi afgerond product dat metrics kan opslaan en tonen. Door een Spring Boot starter toe te voegen aan een Spring Boot applicatie kunnen metrics van de applicatie verzameld worden. Het is dus eenvoudig om metrics te kunnen verzamelen van Spring Boot applicaties.

3.2 Procesevaluatie

In de eerste sprint is gefocust op het maken van een plan van aanpak en het uitvoeren van een onderzoek. Bij het maken van een plan van aanpak is er een planning gemaakt voor de afstudeer periode, waarin vermeld staat hoeveel sprints er beschikbaar zijn voor het gehele project. Ook zijn er een aantal risicofactoren gevonden die de voortgang van het project kunnen belemmeren. Een van de risico's was werken met een Time Serie database, omdat ik voorheen alleen met relationele databases heb gewerkt en geen ervaring heb met Time Serie databases. Om dit risico te verkleinen is aan het begin aandacht besteed aan het leren werken met een Time serie database. Door de uitgebreide documentatie en diverse voorbeeld projecten online was het niet lastig geweest om te leren werken met een Time Serie database.

Daarna heb ik een onderzoek uitgevoerd naar factoren waar er rekening gehouden mee moet worden bij het ontwikkelen van een monitoringsysteem. Dit heeft ervoor gezorgd dat er een applicatie ontwikkeld is dat geschikt om applicaties met microservices architectuur te kunnen monitoren. Daarnaast is er tijd besteed om met Influxdb en Grafana te werken. Hierdoor konden er in de volgende sprints gefocust worden op ontwikkelwerkzaamheden.

De sprints waarin de ontwikkelwerkzaamheden zijn uitgevoerd zijn goed verlopen, want aan het einde van elk sprint is er een werkend prototype geleverd die tevens gedemonstreerd kon worden. Ondanks dat de applicatie in Java ontwikkeld is, heb ik het project toch als uitdagend en leerzaam ervaren. De reden hiervoor is dat het de eerste keer is geweest dat ik met Spring Framework heb gewerkt. Met behulp van collega's en het boek over Spring Framework heb ik de basis van Spring Framework leren kennen en hetgeen wat ik geleerd heb kunnen toepassen in het project.

De keuze om met de software ontwikkelmethodiek SCRUM te werken is een goede keuze geweest, want, hierdoor was het gemakkelijk om te gaan met de verandering van mechanisme in sprint drie. Daarnaast zijn er door de daily stand up meeting voldoende contact momenten geweest met de opdrachtgever om de voortgang te bespreken.

Voor de volgende keer zou ik weer op dezelfde manier te werk gaan, omdat het project in het geheel goed verlopen is en de opdrachtgever tevreden is met het opgeleverde monitoringsysteem. Wel zou ik meer tijd besteden bij het uitdenken van de toepassing van de applicatie, zodat er in één keer een goede keuze gemaakt kan worden over de manier waarop de metrics verzameld worden.

3.3 Beroepstaken evaluatie

In het afstudeerplan heb ik aangegeven op welke beroepstaken er gefocust wordt bij het uitvoeren van de afstudeeropdracht. In deze paragraaf worden de beroepstaken geëvalueerd.

Nummer	Naam	Niveau
1.4	Uitvoeren analyse door definitie van requirements	Lastig + zelfstandig (3)
2.1	Opstellen gegevensmodel van database	Lastig + zelfstandig (3)
2.2	Ontwerpen, bouwen en bevragen van een database	Lastig + zelfstandig (3)
3.3	Bouwen applicatie	Complex + zelfstandig (4)

1.4 Uitvoeren analyse door definitie van requirements (Niveau 3)
Voor dit project heb ik een aantal stakeholders geïnterviewd voor het verzamelen van userstories. Na de demonstratie van het prototype zijn er nieuwe userstories bij gekomen. Dit heeft ervoor gezorgd dat de prioriteiten van voorgaande userstories ook verandert zijn. Ook zorgde de verandering van mechanisme in sprint drie voor nieuwe userstories. Een deel van de bestaande userstories kwamen hierdoor dus te vervallen.

Ik heb naar mijn idee voldoende aandacht besteed bij het verzamelen van userstories. Daarbij ben ik actief bezig geweest bij het uitdenken van de functionaliteiten en de bruikbaarheid van de applicatie.

2.1 Opstellen gegevensmodel van database (Niveau 3)
In het begin van het project is er gewerkt aan de implementatie die het mogelijk maakt om metrics van een instantie van een applicatie op te kunnen slaan. Hierna is gewerkt aan de functionaliteit om metrics van meerder instanties van een applicatie op te kunnen slaan. Om mogelijk te maken is er gezocht naar mogelijke oplossingen met Influxdb. De implementatie die het mogelijk maakt om de metrics op te slaan is in Influxdb is uitgevoerd in sprint 2 en 3.

Deze beroepstaak is niet op de aangegeven niveau behaald, omdat de gegevensmodel van de time series vergelijkbaar zijn met eenvoudige en losse tabellen.

2.2 Ontwerpen, bouwen en bevragen van een database (Niveau 3)
Er is in dit project gewerkt met Influxdb een Time Serie Database voor het opslaan van metrics. Er is rekening gehouden met de manier waarop Influxdb gehost wordt. Daarnaast is er ook rekening gehouden met de naamgeving van de time series, want deze dient per instantie van een applicatie uniek te zijn, zie sprint 3.

Ik heb deze beroeps taak behaald, omdat ik met een time serie database werk. Er is nagedacht over het kunnen opslaan van metrics van meerdere instanties. Hiervoor was het nodig geweest om een unieke naam te verzinnen voor de time series. Door een goede naam conventie te implementeren is er minder onderhoud vereist voor de database, want de time series worden opnieuw gebruikt. Tenslotte is er gekeken naar de performance van Influxdb wanneer de time series van meerder instantie gecombineerd worden. Hieruit is gebleken dat het ophalen van metrics sneller wordt uitgevoerd als de metrics van ander instanties niet gecombineerd worden.

3.3 bouwen applicatie (Niveau 4)
Om de metrics van Spring Boot Applicaties te kunnen verzamelen en opslaan is de Spring boot Metrics Collector ontwikkeld, zie figuur 9 waar een schets van het hele landschap wordt getoond. Om de Spring Boot Collector te kunnen ontwikkelen is er tijd besteed om de basis van Spring Framework te leren kennen, want de applicaties binnen ENOVATION worden hiermee ontwikkeld. De Spring Boot Collector communiceert met Consul om de metrics te kunnen verzamelen van Spring Boot Applicaties. Hiervoor was het nodig geweest om de API van Consul te bestuderen. Daarnaast is er vanaf begin met Docker gewerkt, zodat er geen grootverschil is tussen de ontwikkelomgeving en productieomgeving.
Deze beroepstaak is op het aangegeven niveau behaald, omdat er een Spring Boot Applicatie ontwikkeld is in een complexe omgeving waar gebruik gemaakt wordt van Spring Framework en Docker. Daarnaast is er ook rekening gehouden met toekomstige verandering, zodat het gebruiken van ander database en service registry gemakkelijker geïmplementeerd kan worden.

Bijlagen

Bijlage A - Literatuurlijst

Boeken en pdf:

Ashish Sarin, J. S. (2014). *Getting started with spring framework.*

Newman, S. (2015). *Building Microservices.* O'Reilly.

Tré, G. D. (2007). *Principes van databases.* Pearson Education.

scrumguides. (2015, 4 24). Opgehaald van http://www.scrumguides.org/docs/scrumguide/v1/Scrum-Guide-NL.pdf

Websites:

Docker userguide. (2015, 2 maart). Geraadpleegd van https://docs.docker.com/userguide/

Influxdb. (2015, 2 maart). Geraadpleegd van http://influxdb.com/docs/v0.8/introduction/getting_started.html

Grafana. (2015, 20 maart). Geraadpleegd van http://docs.grafana.org/

Spring Boot. (2015, 12 maart). Geraadpleegd van http://docs.spring.io/spring-boot/docs/current/reference/htmlsingle/

Consul Service Discovery with Docker. (2015, 14 april). Geraadpleegd van http://progrium.com/blog/2014/08/20/consul-service-discovery-with-docker/

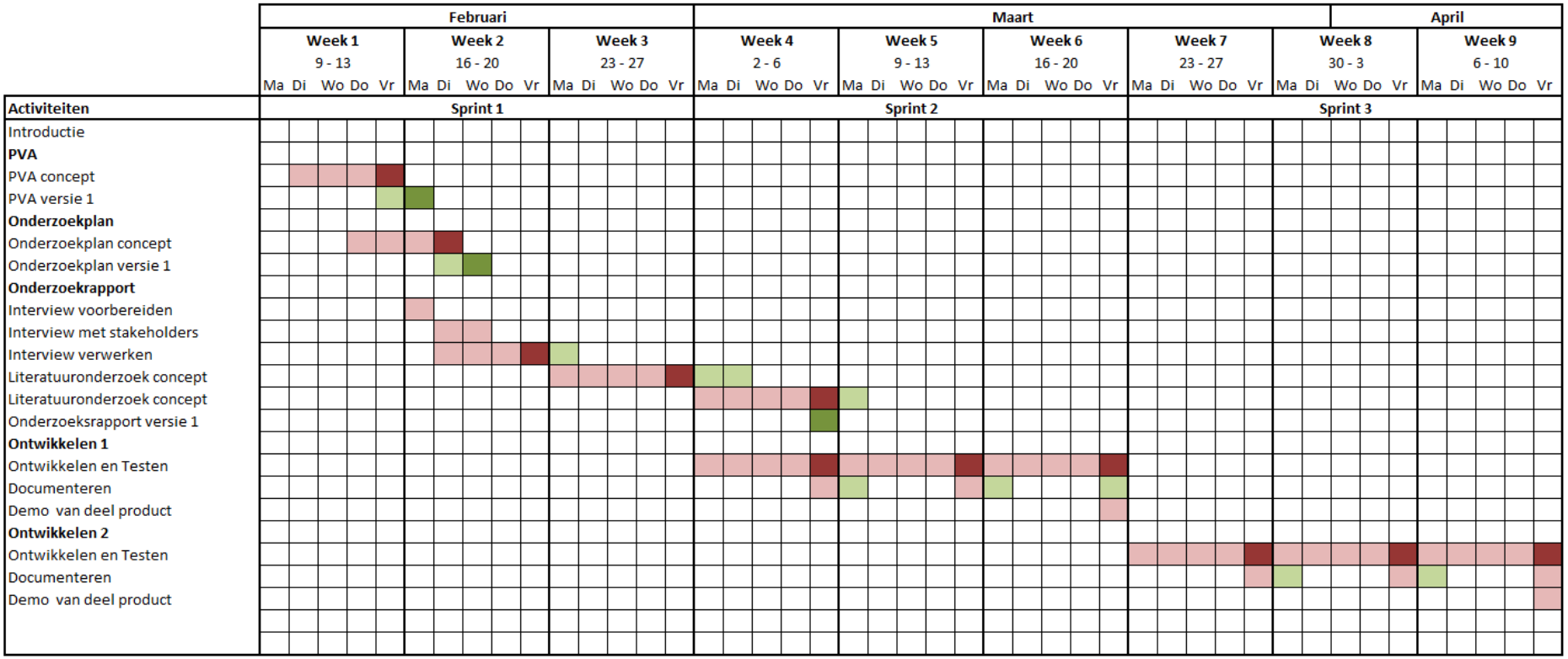
Automatic Docker Service Announcement with Registrator. (2015, 21 april). Geraadpleegd van http://progrium.com/blog/2014/09/10/automatic-docker-service-announcement-with-registrator/

Bijlage B - Woordenlijst

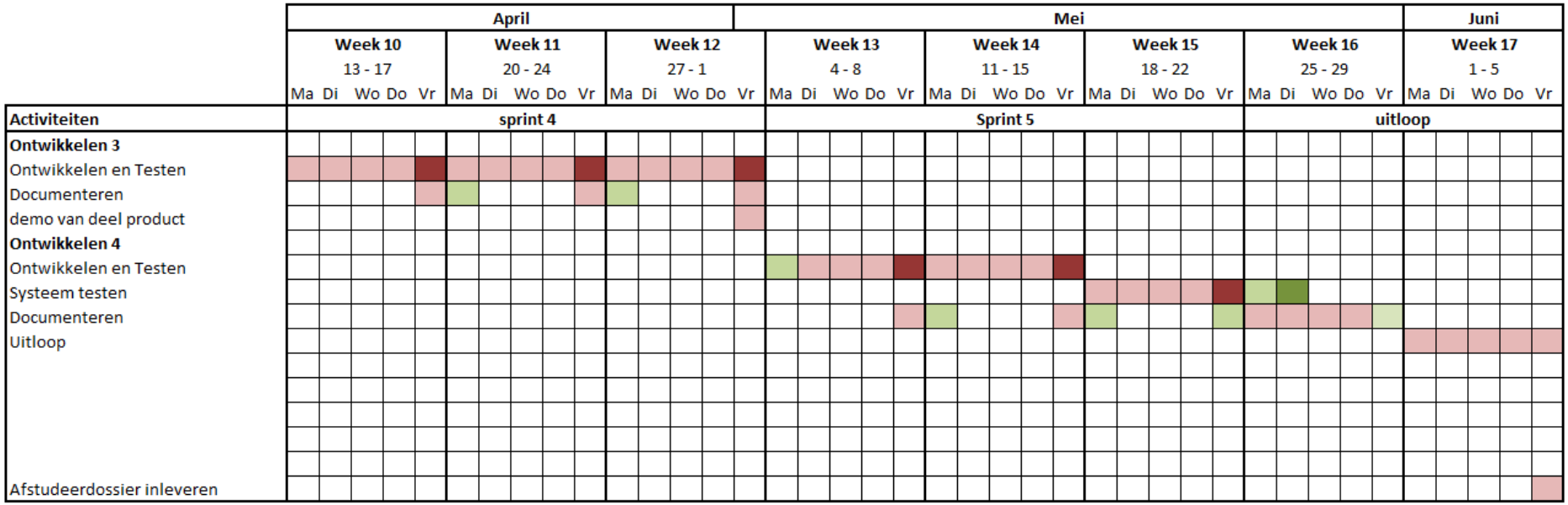
Naam	beschrijving
Metrics	Meetgegevens die verzameld worden van een applicatie. Enkele voorbeelden hiervan zijn memory en CPU gebruik.
Grafana	Een web front end voor het maken van dashboard waar metrics gevisualiseerd worden in grafieken.
Influxdb	Een speciaal database dat geoptimaliseerd is voor het opslaan van time series.
Docker	Een tool om applicaties in een container te kunnen draaien, waarbij de containers geïsoleerd zijn.
Consul	Een Service register, waar de endpoints van applicaties worden geregistreerd.
Registrator	Een tool dat Docker container events registreert bij een Service register.
Node	Een machine waarop applicaties worden gehost.
Instantie	Een applicatie kan bestaan uit meerdere instanties. Met instantie wordt bedoeld een van de instanties van de applicatie.
Spring Framework	Een applicatie Framework voor de Java platform.
Spring Boot	Autoconfiguratie voor Spring Framework.

Bijlage C - Planning

Deel 1



Deel 2



25% examinator komt op bezoek
45% voortgang bespreken
60% concept afstudeerverslag
80% tussentijdse assessment

Periode naar feedback moment
Feedback moment
Feedback verwerken
Oplever moment

5 juni inleveren tussen 10.30 en 12.00
3 exemplaren hardcopy
een keer digitaal
eigen exemplaar meenemen

Bijlage D - Enkele userstories

US 01 Het kunnen verzamelen van metrics **Punten - 3** **Prioriteit - Must**	
User story	Als beheerder wil ik metrics van een Spring Boot Applicatie kunnen verzamelen, zodat de status van de applicatie bekend wordt.
Acceptatie criteria	1. Metrics van Spring Boot Applicatie kunnen verzamelen. 2. De verzamelde metrics kunnen printen op console (scherm). 3. Basis metrics zoals: CPU load, Memory en Heap moeten verzameld kunnen worden.
Taken	1. Bestudeer de API van Spring Boot Actuator om de metrics op te halen van Spring Boot Applicaties. 2. Benader de metrics endpoint van een Spring Boot applicatie (hardcode). 3. Haal de metrics op met een configureerbaar interval tijd. 4. Print de metrics uit op console (scherm). 5. Maak een readme.md file aan met de belangrijkst commando's bij dit project.

US 02 Endpoints van Spring Boot applicaties ophalen bij Consul (Service Registry) **Punten - 3** **Prioriteit - Must**	
User story	De endpoints van Spring Boot applicaties moeten opgehaald worden bij Consul, omdat alle endpoints van Spring Boot applicaties bij Consul geregisterd worden.
Acceptatie criteria	1. De endpoints van de applicaties worden ophalen bij Consul. 2. Endpoints worden opgehaald m.b.v. blocking query.
Taken	1. Bestudeer de API van Consul. 2. Implementeer Consul voor het ophalen van endpoints.

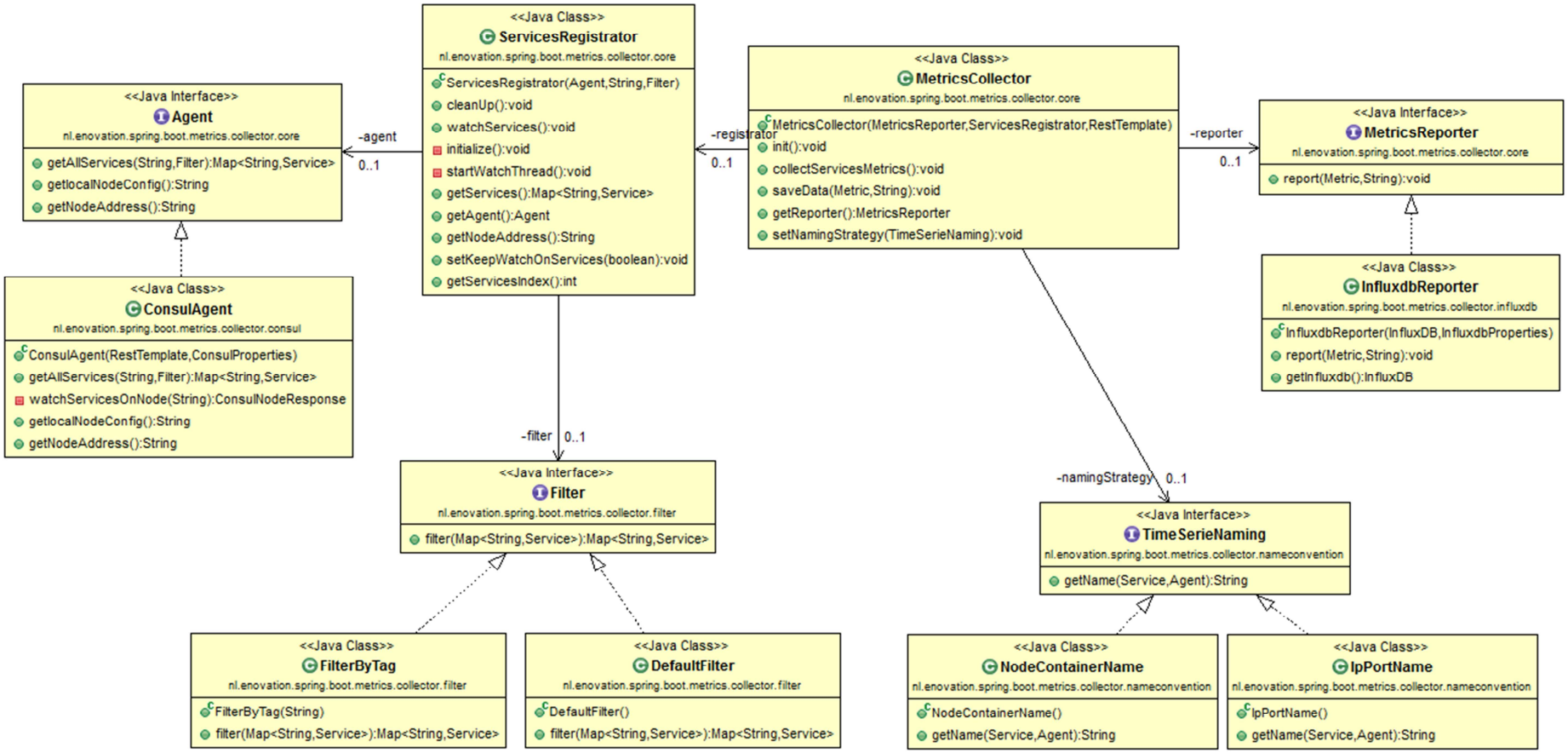
US 03 De verzamelde metrics opslaan in Influxdb (time series database) **Punten - 3** **Prioriteit - Must**	
User story	Als beheerder wil ik dat de verzamelde metrics opgeslagen worden in Influxdb, zodat de geschiedenis wordt bewaard.
Acceptatie criteria	1. De metrics worden opgeslagen in Influxdb.
Taken	1. Installeer Influxdb in Docker. 2. Ontwerp gegevens structuur. 3. Implementeer Influxdb voor het opslaan van metrics

US 04 De metrics tonen in Grafana (web front end) **Punten - 2** **Prioriteit - Must**	
User story	Als beheerder wil ik dat metrics in database getoond worden in Grafana, omdat het visualiseren van de metrics in een grafiek een beter beeld geeft.
Acceptatie criteria	1. Een dashboard waar de metrics getoond worden.
Taken	1. Installeer Grafana in Docker. 2. Maak een dashboard waar de metrics getoond worden.

Bijlage E - Metrics van een eenvoudige applicatie in JSON formaat

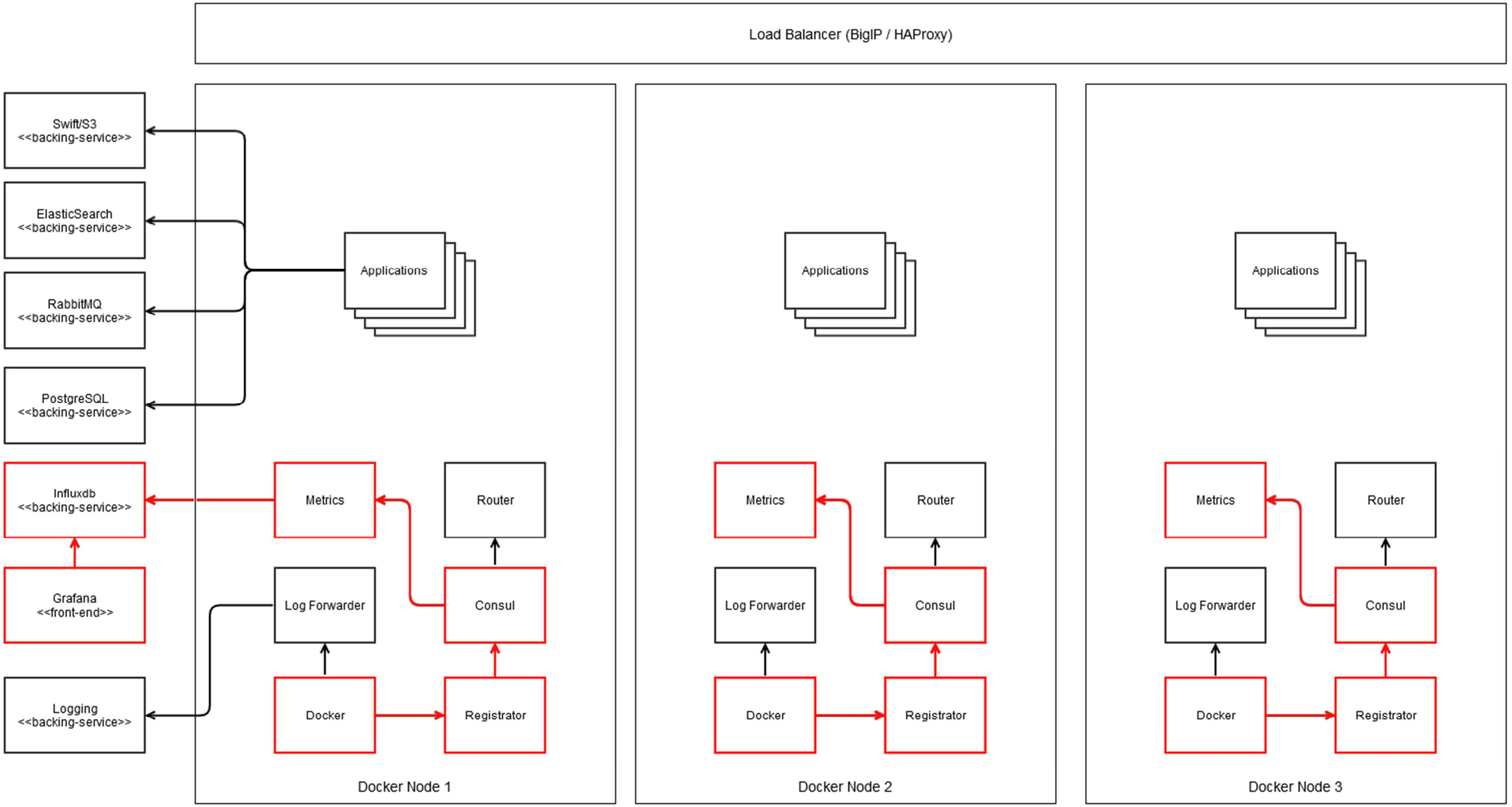

```
{
    "mem": 215040,
    "mem.free": 143753,
    "processors": 4,
    "uptime": 1850679,
    "instance.uptime": 1842951,
    "systemload.average": 0.01,
    "heap.committed": 215040,
    "heap.init": 32768,
    "heap.used": 71286,
    "heap": 458752,
    "threads.peak": 21,
    "threads.daemon": 19,
    "threads": 21,
    "classes": 6674,
    "classes.loaded": 6674,
    "classes.unloaded": 0,
    "gc.ps_scavenge.count": 17,
    "gc.ps_scavenge.time": 176,
    "gc.ps_marksweep.count": 2,
    "gc.ps_marksweep.time": 227,
    "httpsessions.max": -1,
    "httpsessions.active": 0,
    "datasource.primary.active": 0,
    "datasource.primary.usage": 0,
    "counter.status.200.metrics": 369,
    "counter.status.200.star-star.favicon.ico": 1,
    "gauge.response.metrics": 4,
    "gauge.response.star-star.favicon.ico": 15
}
```

Bijlage F - Klassendiagram



Klassendiagram van de Spring Boot Metrics Collector

Bijlage G - Architectuur Productieomgeving met microservices



De rood gemarkeerde blokken zijn onderdelen van het monitoringsysteem en worden gebruikt bij het uitvoeren van de systeemtest.

Externe Bijlage

Bijlage H - Plan van aanpak

Bijlage I - Onderzoeksrapport

Plan van aanpak

Ontwikkelen van een monitoringsysteem

Naam:	Wan Long Cheung	Opdrachtgever:	Albert van ‘t Hart
Datum:	16-02-2015	Versie:	1

Inhoudsopgave

Het bedrijf

ENOVATION is een internationaal ICT-bedrijf, dat actief is in de zorg en transport & logistiek sector. Het bedrijf zorgt voornamelijk voor de uitwisseling van elektronische berichten tussen mensen, organisaties en systemen. De vestiging bij Capelle aan de IJssel bestaat ongeveer uit 100 werknemers en verwacht de komende jaar een groei van 10 procent.

Het project

Probleemstelling

ENOVATION beschikt over een eigen hostingcenter voor hosting van systemen van ENOVATION en hosting van systemen van derden zoals Rijkswaterstaat, Politie Haaglanden, Stichting Beurshal en diverse informatiesystemen voor de zorg. Huidige systemen van ENOVATION hebben veelal een traditionele monolithisch opzet, waarbij deze vaak op een vast aantal servers draaien. ENOVATION beschikt over een standaard monitoring omgeving (Nagios), maar de aansluiting van nieuwe software op deze omgeving kost veel tijd en is niet gestandaardiseerd. Om een koppeling te maken met de omgeving worden handmatige handelingen uitgevoerd en maatwerk software gemaakt. ENOVATION gaat in de komende jaren migreren van een architectuur met monolithische systemen naar een systeem dat bestaat uit microservices. De deployment van deze microservices zal ervoor zorgen dat deze niet (altijd) meer op een vast aantal servers draaien. Voor het monitoren van de systemen worden er diverse meetgegevens bijgehouden die per systeem kan verschillen.

Doelstelling van de afstudeeropdracht

ENOVATION wil voor het monitoren van systemen een standaard monitoringsysteem, die inzetbaar is voor alle systemen. Hierbij gaat het om het opslaan van diverse gegevens. Enkele voorbeelden hiervan zijn de processor capaciteit gebruik, geheugen gebruik en een aantal diverse foutmeldingen. Deze gegevens worden op een later tijdstip of realtime opgehaald en op een dashboard getoond.

Resultaat

Een monitoringsysteem (prototype) dat aansluit op de nieuwe wensen en eisen van ENOVATION en welke gemakkelijk inzetbaar is bij de ontwikkeling van nieuwe systemen op basis van microservices. Het monitoringsysteem bestaat uit componenten dat meetgegevens verzamelt en toont. Hiermee kan een systeembeheerder gemakkelijk de systemen monitoren en wordt er tijd bespaard in zowel het beheer- als ontwikkelproces.

Scope

Het ontwikkelen van een monitor systeem kan breed getrokken worden. Om een afgerond product te kunnen opleveren aan het einde, wordt er in de eerste instantie gefocust op het automatische kunnen verzamelen en het tonen van meetgegevens. Het tonen van de meetgegevens wordt pas uitgevoerd wanneer het verzamelen van de meetgegevens naar wensen werkt. Daarnaast valt het ontwikkelen van alarmeringen buiten de scope.

Risicofactoren

Bij het analyseren van het project heb ik een aantal risicofactoren gevonden die de voortgang van het project kunnen belemmeren. Om dit te kunnen voorkomen is er voor elk risicofactor een oplossing bedacht. Hieronder worden de risicofactoren korte beschreven met een oplossing om dit te kunnen vermijden en de kans op optreden te kunnen verkleinen.

Kennis van Timeserie Database

In dit project worden meetgegevens van een applicatie verzameld en opgeslagen in een Timeserie Database. Het opslaan van de gegevens kan een mogelijk risico zijn, want ik heb alleen ervaring opgedaan met een relationele Database.

Werken met een Timeserie Database is dus voor mij nieuw, daarom wordt er in het onderzoek tijd besteed voor het analyseren van de werking van een Timeserie Database. Daarnaast is het niet bekend in welke vormen de meetgegevens beschikbaar zijn en ondersteund worden in een Timeserie Database. Ook wordt er tijd besteed in het onderzoek om een standaard vorm van de meetgegevens te kunnen definiëren.

Kennis van Spring Framework

ENOVATION ontwikkeld applicaties in JAVA met Spring Framework. Voor het ontwikkelen van een nieuwe monitoringsysteem is het noodzakelijk dat er rekening gehouden wordt met Spring Framework, want onderdelen van het monitoringsysteem moeten ook ontwikkeld worden in JAVA met Spring Framework.

Om te leren werken met Spring Framework worden tutorials gevolgd en andere middelen gebruikt om de basis het Framework te leren kennen.

Werken met een aangepaste SCRUM

Voor dit project is er gekozen om een Agile methodiek toe te passen, want hiermee kan er gemakkelijk ingespeeld worden op de verandering van requirements. Daarnaast kan er ook meer controles uitgeoefend worden, doordat de voortgang dagelijks wordt doorgenomen.

In dit project is er gekozen om met SCRUM te werken, omdat het een bekende methodiek is voor mij en ENOVATION. Hierdoor is het niet nodig om de methodiek te onderzoeken. Daarnaast zijn voldoende contactmomenten met de opdrachtgever om de voortgang van het afstudeerproject te bespreken, zodat het in de goede richting gestuurd kan worden. Ook heb ik goede ervaring opgedaan met SCRUM in mijn derdejaars stage. Hierdoor heb ik meer vertrouwen in SCRUM.

In dit project wordt er gewerkt met een aangepaste SCRUM methodiek, omdat niet alle activiteiten toepasbaar zijn binnen het project en volgens de officiële SCRUM beschrijving moet een SCRUM team minimaal uit drie teamleden bestaan. In vergelijking met een ander Agile methodiek zoals XP is SCRUM beter voor dit project, omdat er activiteiten in het methodiek zitten dat beter aansluit met de beschikbare middelen van het bedrijf. Verder wordt hieronder toegelicht hoe SCRUM wordt toegepast voor dit project.

Sprints

Er is afgesproken dat een sprint een duur heeft van drie weken, zodat het synchroon loopt met het andere project binnen ENOVATION. De demo kan dan op hetzelfde moment gepresenteerd worden. Om de voortgang van het project te kunnen monitoren wordt er vanaf het begin gewerkt met sprints, waarbij de deliverables niet een werkend prototype is maar documenten. Na elk sprint wordt er dus documenten opgeleverd en wanneer het ontwikkeltraject begint worden er prototypes opgeleverd.

Userstories

Voor het verzamelen van Userstories worden stakeholders geïnterviewd. De stakeholders zijn alle belanghebbende van het nieuwe monitoring systeem. Hierna worden Userstories ingevoerd in JIRA en deze worden geprioriteerd met de stagebegeleider (product owner). Vervolgens worden de Userstories ingepland voor een sprint, ook wel sprint planning genoemd. Wanneer het bekend is welke Userstories worden uitgevoerd in de sprint worden de Userstories opgesplitst in taken. Er wordt per dag minimaal een taak afgerond, want een taak moet zo opgedeeld zijn dat het in een werkdag uitgevoerd kan worden.

SCRUM board

Voor het monitoren van de voortgang van het project wordt een SCRUM board gebruikt. De SCRUM board wordt zowel fysiek op de muur met papier en magneet als met de tool JIRA (digitaal) bij gehouden. Op de SCRUM board staan de Userstories die voor de sprint zijn ingepland. De SCRUM board is verdeeld in drie kolommen namelijk: ‘To Do’, ‘In progress’ en ‘Done’. In de kolom ‘To Do’ staan de Userstories die uitgevoerd worden in de sprint. In de kolom ‘In progress’ staan de taken waar men mee bezig is. In de kolom ‘Done’ staan de Userstories en taken die uitgewerkt zijn. Voor het verifiëren van een Userstory wordt een verify taak gemaakt en deze wordt dan uitgevoerd door de product owner.

Daily Standup Meeting

De Daily Standup Meeting wordt gehouden rond om de fysieke SCRUM board met het platformteam dat zich bezig met het ontwikkelen van componenten voor het platform van ENOVATION. Het afstudeerproject wordt een onderdeel van het platform en daarom is het gebruikelijk dat de Daily Standup meeting SCRUM meeting samen gehouden wordt.

Sprint Review en Retrospective

Sprint review wordt aan het einde van elk sprint uitgevoerd. Hierin wordt het verloop van de sprint doorgenomen, een demo geven van het prototype en het opnemen van feedback. Vervolgens wordt de sprint retrospective uitgevoerd met het platformteam, omdat zoals eerder vermeld behoord het monitorsysteem tot een onderdeel van het platform.

Activiteiten en tussen producten

Plan van aanpak

In het begin van het project wordt een plan van aanpak gemaakt, om te opdrachtgever een beeld te geven van hoe het project wordt uitgevoerd.

Onderzoekplan & Onderzoeksrapport

Voor het ontwikkelen van een monitoringsysteem wordt er eerst onderzoek gedaan, omdat er rekening gehouden moet worden met de nieuwe architectuur waar ENOVATION naar gaat migreren. Daarnaast zijn er zijn veel tools beschikbare die gebruikt kunnen worden voor het te ontwikkelen monitoringsysteem.

In het onderzoeksplan wordt er bepaald naar waar onderzoek gedaan. In het onderzoek staat het opdoen van de nodige kennis centraal voor het ontwikkelen van het monitoringsysteem.

Productbacklog

Een document met Userstories dat verzameld wordt voor het te ontwikkelen monitoringsysteem. De Userstories worden verzameld door stakeholder van het monitoringsysteem te interviewen.

Sprintbacklog

De uitgevoerd werkzaamheden worden hierin beschreven met de feedback van sprint retrospective.

Prototype

Als het ontwikkeltraject van start gaat, kan er een aan het einde van de sprint een prototype opgeleverd worden.

Testplan & Testrapport

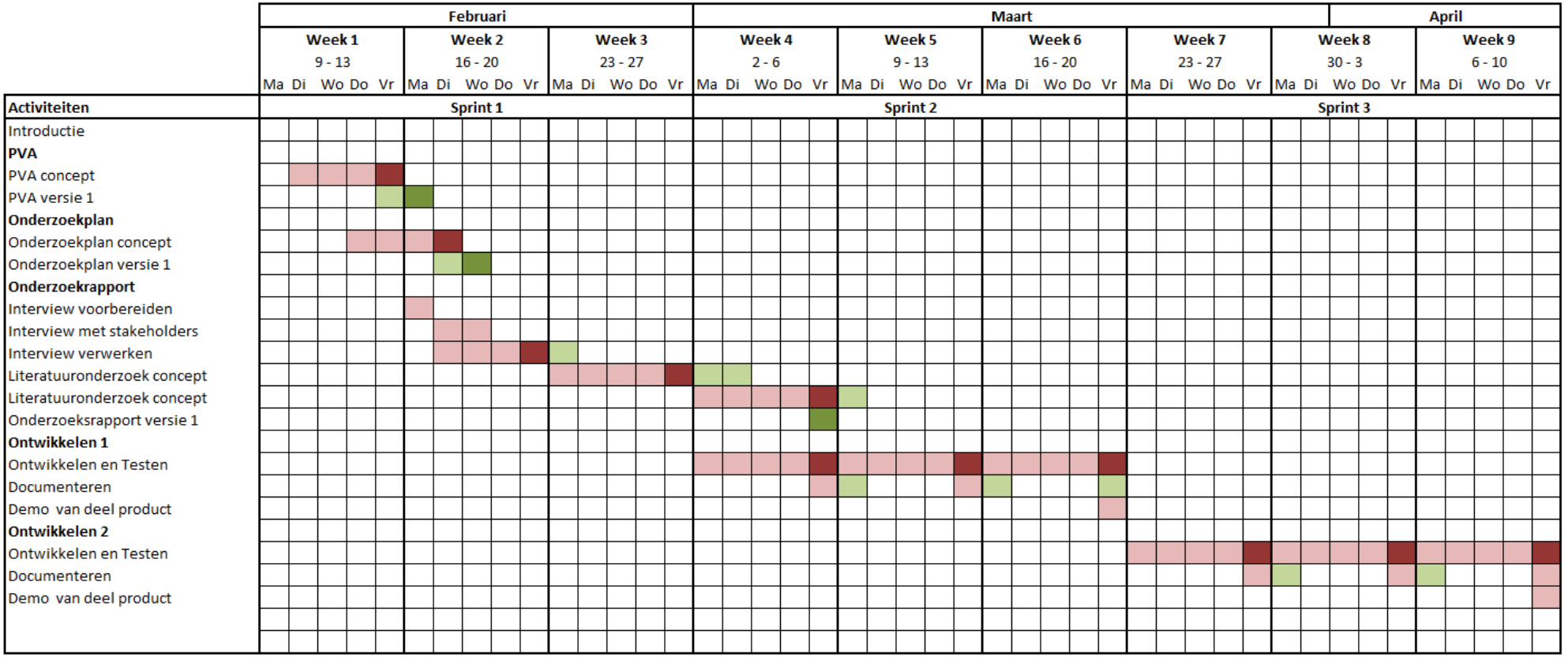
Voor het test van het gehele systeem wordt een systeemtest uitgevoerd. Vooraf wordt er bepaald wat er getest wordt en wanneer het goed gekeurd kan worden, dit staat in de Testplan. De resultaten worden vervolgens gedocumenteerd in het Testrapport.

Wijze van rapporteren

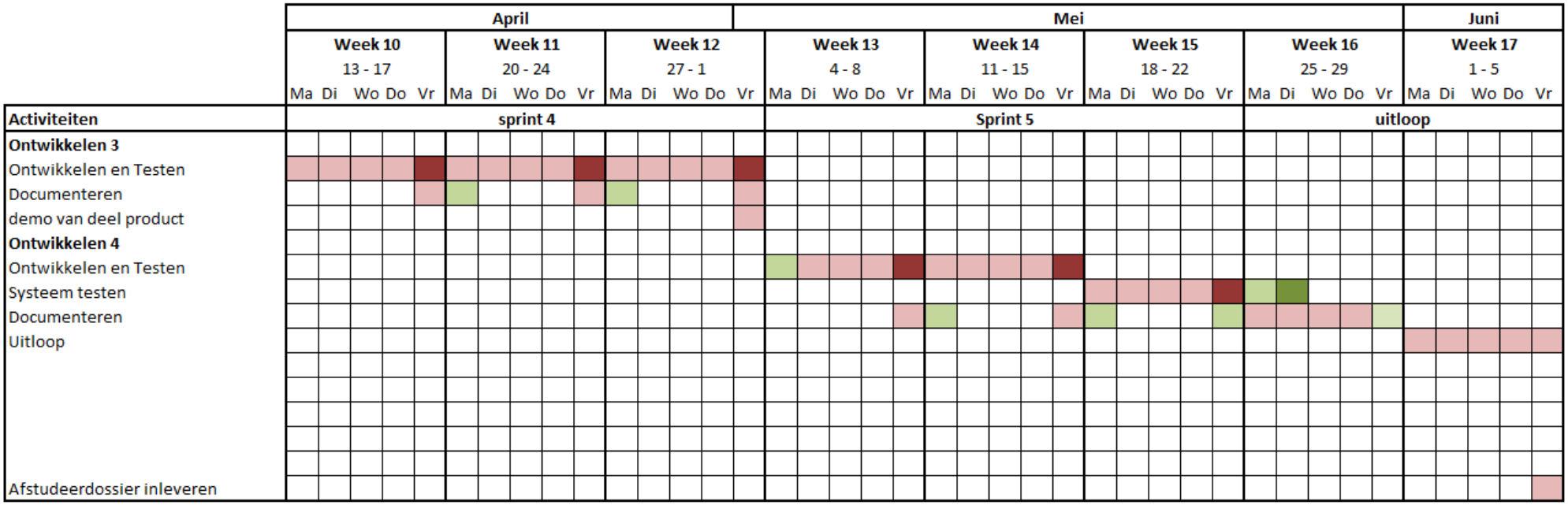
De opdrachtgever wordt op een aantal manieren op de hoogte gehouden over de voortgang van het project. Allereerst is er een onderdeel van SCRUM de ‘Daily Standup Meeting’ hierin wordt er kort verteld over de uitgevoerde activiteiten en waaraan gewerkt wordt. Daarnaast wordt aan het einde van elk week de uitgevoerde werkzaamheden doorgenomen. Tenslotte een review moment aan het einde van elk sprint waar het tussen product toegelicht kan worden.

Planning

Deel 1 (Z.o.z. deel 2)



Deel 2



25% examinator komt op bezoek
45% voortgang bespreken
60% concept afstudeerverslag
80% tussentijdse assessment

Periode naar feedback moment
Feedback moment
Feedback verwerken
Oplever moment

5 juni inleveren tussen 10.30 en 12.00
3 exemplaren hardcopy
een keer digitaal
eigen exemplaar meenemen

Onderzoeksrapport

Het ontwikkelen van een monitoringsysteem

Naam:	Wan Long Cheung	Opdrachtgever:	Albert van ‘t Hart
Datum:	05-04-2015	Versie:	1.0

Inhoudsopgave

Documenteigenschappen

Historie

VERSIE	DATUM	VERANDERING	OPDRACHTGEVER	AUTEUR
0.1	03-03-2015	Concept	Albert van ’t Hart	Wan Long Cheung
0.2	12-03-2015	Concept	Albert van ’t Hart	Wan Long Cheung
0.3	23-03-2015	Concept	Albert van ’t Hart	Wan Long Cheung
1.0	05-04-2015	Final	Albert van ’t Hart	Wan Long Cheung

Samenvatting

Het doel van een monitoringsysteem is om metrics van applicaties te kunnen verzamelen. Aan de hand van de gegevens kunnen inschatting gemaakt worden, zodat er gepaste handelingen worden uitvoert om de service te kunnen optimaliseren. In de huidige situatie van ENOVATION is het monitoringsysteem verouderd en niet flexibel. Er wordt veel handmatige handelingen uitgevoerd om de applicaties en servers te kunnen monitoren. In de komende tijd wil ENOVATION over stappen naar een architectuur met microservices. Met het huidige monitoringsysteem wordt het monitoren van microservices een zware taak, omdat een microservice dynamisch en in een groot aantal is. ENOVATION wil een nieuw monitoringsysteem hebben, waarbij het monitoren van microservers gemakkelijk uitgevoerd kan worden en het bestaande monitoringsysteem geheel kan vervangen. Om een nieuw monitoringsysteem te kunnen ontwikkelen wordt er rekening gehouden met de ontwikkelstraat van ENOVATION. Het monitoringsysteem wordt uiteindelijk een platform component dat gemakkelijk inzetbaar is voor alle applicaties. Het monitoringsysteem kan ingedeeld worden in fases. Binnen de afstudeeropdracht wordt de focust op de eerste instantie gelegd op fase een, het kunnen opslaan en tonen van de gegevens.

Deel I: opzet onderzoek

In dit deel van het rapport wordt het opzet van het onderzoek beschreven. Voor het beantwoorden van de hoofdvraag worden er deelvragen opgesteld. Per deelvraag wordt er toegelicht hoe deze beantwoord worden.

1.1 Inleiding

ENOVATION is een bedrijf dat constant aan het groeien is, zo ook de systemen van ENOVATION. Systemen krijgen steeds meer functionaliteiten en worden complexer. Om de systemen goed te kunnen blijven monitoren is het nodig dat de monitor systeem ook met de tijd mee groeit. Het huidige monitor systeem is niet geschikt voor de toekomst en dus wordt er gewerkt aan een nieuwe monitor systeem. ENOVATION heeft een beeld bij het ontwikkelen van de nieuwe monitor systeem. Om dit te kunnen realiseren wordt er onderzoek gedaan naar systeem monitoring binnen ENOVATION. In dit onderzoeksproject wordt de volgende hoofdvraag gesteld:

Waar moet er rekening mee gehouden worden bij het ontwikkelen van een monitoringsysteem?

Bij deze onderzoeksvraag zijn de volgende deelvragen opgesteld:

- Wat wordt er verstaan onder systeem monitoring?
- Hoe ziet de huidige situatie van systeem monitoring eruit?
- Hoe ziet de ontwikkelstraat van ENOVATION eruit?
- Waar moet er rekening mee gehouden worden bij het ontwikkelen van een monitoringsysteem?

1.2 Methode

Om de deelvragen te kunnen beantwoorden worden er per vraag een methode vastgesteld om deze te kunnen beantwoorden.

Wat wordt er verstaan onder systeem monitoring?

Om deze vraag te beantwoorden, wordt een kort literatuuronderzoek gedaan. Hierin worden toepassing van monitoring verder toegelicht. Ook wordt er gezocht naar een algemene definitie van systeem monitoring.

Hoe ziet de huidige situatie van systeem monitoring eruit?

Om deze vraag te beantwoorden, worden interviews gehouden met applicatie-beheerders en ontwikkelaars van het huidige monitoringsysteem. Daarnaast worden de problemen van de huidige situatie beschreven.

Hoe ziet de ontwikkelstraat van ENOVATION eruit?

Om deze vraag te beantwoorden, worden interviews gehouden met softwareontwikkelaars van ENOVATION, omdat zij dagelijks bezig zijn in de ontwikkelstraat en dus het meeste kennis daarover hebben.

Waar wil ENOVATION naar toe met het te ontwikkelen monitoringsysteem?

Om deze vraag te beantwoorden, worden interviews gehouden met solution-architecten van ENOVATION, omdat het nieuwe beeld van systeem monitoring door hun bedacht zijn.

Deel II: Resultaten

In dit deel van het rapport worden de deelvragen beantwoord. Per deelvraag worden de resultaten verder toegelicht.

2.1 Wat wordt er verstaan onder monitoringsysteem?

In dit hoofdstuk wordt er als eerst een aantal voorbeelden beschreven waar monitoring wordt toegepast. Vervolgens worden de algemene definities die binnen de context horen uitgelegd en ten slotte een beschrijving van een bekend systeem monitoring in de Windows omgeving.

2.1.1 Toepassing

In de zorgsector worden patiënten gemonitord waarbij meetwaardes zoals hartslag, bloeddruk en zuurstofgehalte continue gemeten en getoond worden. Hierdoor worden het medische personeel op de hoogte gehouden van de toestand van de patiënt. Geavanceerde monitoring hebben daarnaast ook alarmeringen wanneer het toestand van de patiënt dreigend wordt.

In de ICT-sector worden systemen en applicaties gemonitord. Dit wordt gedaan om op tijd te kunnen handelen, zodat de services optimaal kunnen blijven functioneren. Daarnaast wordt het ook gebruikt om de oorzaak van het probleem te kunnen achter halen. Voor het monitoren van systemen worden systemen zoals Nagios toegepast.

In de particulierensector wordt monitoring ook toegepast door de opkomende trend van het gebruiken van “wearable tech divices” zoals “smart wristband” en “smart watch”. Deze apparaten bevatten diverse sensoren die activiteiten van de gebruiker kan meten. Enkele voorbeelden hiervan zijn het meten van de hartslag, lichaamstemperatuur en het aantal gelopen stappen. Na het verzamelen van de meetgegevens kunnen deze getoond worden op een mobile app.

Kortom monitoring komt in diverse sectoren voor. Het is dus belangrijk om bepaalde gegevens te kunnen bijhouden, zodat men deze kan analyseren. De genoemde voorbeelden hierboven hebben allemaal een ding gemeen en dat is het verloop van een bepaalde meetwaarde kunnen bekijken en aan de hand daarvan handelen. Dit is de kern van monitoring.

Voor de meest passende definitie van monitoring is het volgende gevonden (Ensie, 2015):

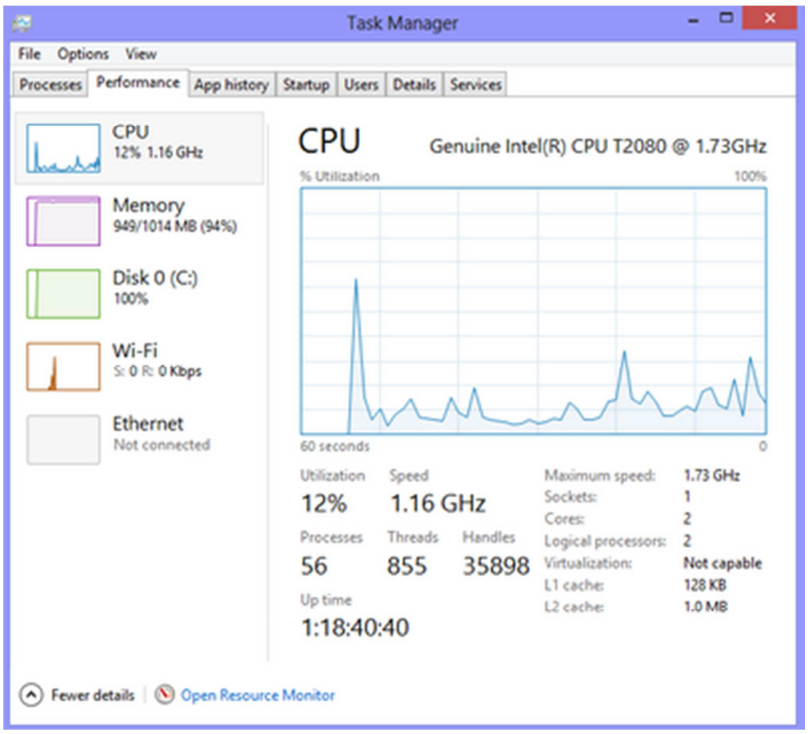
> *Monitoren is het continu in de gaten houden van (complexe) processen en onderwerpen. Aan de hand van monitoren kan het proces worden aanpast en een nieuwe richting worden geven. Monitoren wordt gedaan om het beste resultaat te bewerkstelligen.*

Voor een algemene beschrijving van monitoringsysteem is de volgende definitie gevonden (Encyclo, 2015):

> *Aan de hand waarvan bepaalde effecten van het gevoerde (ruimtelijke) beleid systematisch opgevolgd en in kaart gebracht kunnen worden (bijvoorbeeld verkeersdruk of woningbestand). Aan de hand van deze gegevens kan het beleid worden geëvalueerd en zo nodig worden bijgestuurd.*

2.2.2 Windows Task Manager

Een voorbeeld van monitoring is de Windows Task Manager. Deze is op de Windows omgeving te vinden door op de volgende toetsen combinatie in te drukken: “Crtl + Alt + Esc” en vervolgens op “Start Task Manager”. In de afbeelding hieronder is te zien dat de CPU (processor kracht) gebruik van de afgelopen minuut getoond wordt in een grafiek.



Figuur 1 Windows Task Manager in Windows 8

2.2 Hoe ziet de huidige situatie van systeem monitoring eruit?

Om duidelijk te maken hoe het huidige monitor systeem van EVONVATION werkt, worden eerst de tools beschreven die gebruikt worden in de huidige situatie. Vervolgens wordt geschetst en beschreven hoe het huidige landschap van systeem monitor eruit ziet. Daarna wordt een beschrijving gegeven van de problemen die in de huidige situatie voorkomen. Ten slotte wordt er beschreven in welke fase de huidige monitoring zich bevindt volgens het model van James Turnbull.

In het huidige monitoringsysteem worden de volgende tools gebruikt; Nagios, RRDTool en Cacti. Deze worden hieronder beschreven.

2.2.1Nagios

Een geavanceerd monitoringsysteem dat naast het monitoren van systemen, netwerken en infrastructuren ook notificaties kan versturen indien er incidenten optreden. Nagios wordt voornamelijk gebruikt voor het controleren of een systeem online is door om een bepaalde tijd een bericht te sturen naar het systeem dat gemonitord wordt. Daarnaast voert Nagios ook op gepland tijdstip andere controles uit. Voor het configureren van de controles is er een configuratie file nodig waar commando's van Nagios ingevoerd kunnen worden.

2.2.2 RRDTool

Een RRD is een Database waar gewerkt wordt met een vaste hoeveelheid data opslagcapaciteit. Een RRD beperkt zich tot het alleen opslaan van numerieke waardes en wordt voornamelijk gebruikt voor het opslaan van meetwaardes. Het idee achter RDD is dat de opslagruimte hergebruik wordt wanneer er geen lege opslag ruimte beschikbaar is. Aan de hand van het volgende voorbeeld wordt het duidelijk gemaakt hoe een RRD werkt. Een RRD kan gezien worden als een klok waar achter elk getal een waarde wordt opgeslagen. Wanneer een waarde wordt opgeslagen wijst de wijzer naar het volgende getal waar een nieuwe waarde wordt opgeslagen en deze cyclus wordt continue herhaald. Op een geven moment wijst de wijzer naar een getal waar al een waarde wordt opgeslagen, deze waarde wordt overschreven door de nieuwe waarde. Op deze manier vervangt de nieuwste waarde en oudste waarde wanneer er geen opslag ruimte meer beschikbaar is. Er kan dus tot op een bepaalde hoogte waardes opgeslagen worden, voordat er waardes worden overschreven. De capaciteit van de database zal altijd dezelfde blijven en het is niet nodig om de database van tijd tot tijd op te schonen. Een RRDTool werkt met RRDs en maakt het mogelijk om waardes op te slaan en op te vragen.

2.2.3 Cacti

Een tool dat data ophaalt uit een RRD en deze vervolgens grafische kan weergeven. Met de tool kunnen er diverse dashboards geconfigureerd worden, waarbij de waardes ook nog eens op diverse manieren gevisualiseerd kunnen worden met behulp van templates. Deze kunnen overigens eenvoudig aangepast worden.

2.2.4 Maatwerksoftware

De tools die gebruikt worden voor het monitoren van de systemen vereisen dat er maatwerksoftware ontwikkeld moet worden om specifiek onderdelen van de systemen te kunnen monitoren. Voor Nagios is de ENOVATION Monitor Demon ontwikkeld die er voor zorgt dat de systemen op andere functionaliteiten gemonitord kan worden naast de health checks. Een van de functies die de ENOVATION Monitor Demon uitvoert is het controleren of het mail systeem nog goed functioneert. Dit wordt gedaan door een mail te versturen en vervolgens te controleren of de mail is ontvangen. Voor zulke controles worden er maatwerksoftware ontwikkeld in het ENOVATION Monitor Demon. De functie van ENOVATION Monitor Demon is dus het uitvoeren van controles en deze terug koppelen met Nagios.

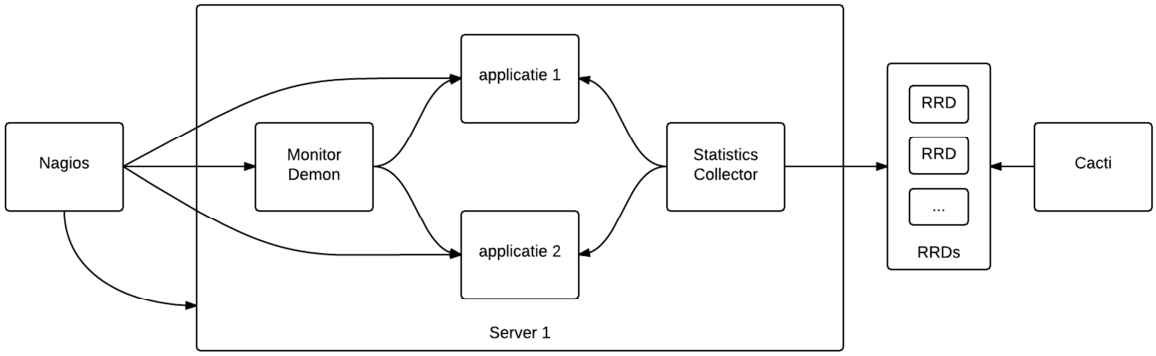
Daarnaast is de ENOVATION Statistics Collector ontwikkeld voor de integratie met de RRDTool, zodat meetgegevens van systemen verzameld en opgeslagen kan worden. Met de ENOVATION Statistics Collector worden specifieke meetwaardes verzameld. Enkele voorbeelden hiervan zijn: memory heap, aantal threads en het aantal database connecties. Voor het monitoren van andere waardes moet er aanpassing gemaakt worden in de software.

Zowel de ENOVATION Monitor Demon als de ENOVATION Statistics Collector worden op machines gehost waar de systemen worden gemonitord.

2.2.5 Het monitorlandschap

In figuur 2 is te zien dat er twee applicatie gehost worden op een server. Naast de applicaties worden ook maatwerksoftware (Monitor Demon en Statistics Collector) op de server gehost die het monitoring uitbreiden. Nagios monitort de server en de applicaties. Daarnaast wordt er met behulp van de Monitor Demon andere controles uitgevoerd op de applicaties. Met de Statistics Collector worden meetgegevens van de applicaties verzameld en opgeslagen in een RDD die vervolgens in een diagram getoond wordt in Cacti.

Alle systemen van ENOVATION worden gemonitord met Nagios maar niet van alle systemen worden de meetgegevens verzameld.



Figuur 2 Schets van huidige situatie van systeem monitoring

2.2.6 De problemen bij meetgegevens verzamelen

In de huidige situatie zijn er een aantal problemen bij het verzamelen van meetgegevens. Ten eerste wordt er per meetwaarde per applicatie handmatige handelingen vereist om de gegevens te kunnen opslaan en tonen. Voor het opslaan van het geheugen en processor gebruik van een applicatie worden er handmatig twee aparte RRD aangemaakt per applicatie. Voor de situatie in figuur 2 worden dus vier RRD's aangemaakt voor het opslaan van twee meetwaardes van twee applicaties. Voor het weergeven van de meetgegevens wordt er bij Cacti geconfigureerd waar de data opgeslagen zijn.

Ten tweede is de huidige manier van systeem monitoring niet geschikt voor de toekomst, want ENOVATION wil in de komende jaren migreren naar een architectuur met microservers. Hierdoor zullen het aantal RRD's met een groot aantal toenemen met het huidige systeem, het beheren van de RRD's wordt dus een zware taak. Daarnaast moeten deze ook handmatige configureert worden in Cacti.

Ten slotte moet er maatwerksoftware ontwikkeld en onderhouden worden om de meet gegevens te kunnen verzamelen, want hiervoor is de Statistics Collector ontwikkeld.

2.2.7 Monitoring model

In het artikel (A Monitoring Maturity Model, 2015) van James Turnbull wordt er een model beschreven over monitoring. In het model wordt monitoring verdeeld in drie fases, deze zijn: "Manual", "Reactive" en "Proactive". Hieronder worden de fasen verder toegelicht.

Manual - In deze fase van monitoring ligt de focus op het kunnen leveren van een service en er wordt weinig aandacht besteed aan de kwaliteit hiervan. Het monitoren wordt vaak handmatig uitgevoerd, waarbij eenvoudige controles worden verricht.

Reactive - Monitoring wordt in deze fase grotendeels geautomatiseerd. Hierbij worden tools zoals Nagios gebruikt voor het verzamelen van meetgegevens. Enkele voorbeelden van meetgegevens zijn: processor en geheugen gebruik. Daarnaast worden er meldingen verstuurd om de beheerder op de hoogte te houden over de ontwikkelingen.

Proactive - Het monitoren gebeurt geheel automatisch, waarbij standaard problemen automatisch worden opgelost met standaard oplossingen. De focus in deze fase ligt op het kunnen leveren van kwaliteit naast de beschikbaarheid.

Volgens het model van James Turnbull bevindt ENOVATION met monitoring nu is de fase van "Reactive", want op een aantal punten komt het over een met de beschrijving van "Reactive" monitoring. Verder wil ENOVATION uiteindelijk over gaan naar "Proactive" fase maar dat wordt verder behandeld in een ander deelvraag.

2.3 Hoe ziet de ontwikkelstraat van ENOVATION eruit?

2.3.1 Methodiek

Binnen ENOVATION wordt de software methodiek Scrum toegepast. De sprints hebben een duur van drie weken. Projecten in het platform team hebben een groter draagvlak en daarom wordt het tussenproduct gedemonstreerd aan een grotere groep. Bij andere teams binnen de ontwikkelafdeling worden de demo's meestal gegeven aan een kleinere groep vergeleken met het platform team. De meeste activiteiten binnen Scrum worden toegepast in de praktijk. Wel worden sommige activiteiten op detail niveau anders uitgevoerd dan de definities in de Scrum guide.

2.3.2 Tools

Om het software ontwikkeltraject te kunnen optimaliseren worden tools gebruikt. De meeste tools zijn opensource, op een enkele na. Hieronder worden de tools die gebruikt worden toegelicht.

Atlassian (Jira, Stash, Confluence)

Atlassian biedt een aantal tools om het ontwikkeltraject te kunnen optimaliseren. Daarnaast zijn de tools van Atlassian met elkaar geïntegreerd. Hierdoor kunnen onderdelen met elkaar gelinkt worden. Een van de producten die Atlassian biedt is Jira, een online Scrum Board. De voortgang van het project kan hiermee online gemonitord worden. Voor het centraal opslaan van de code wordt Stash gebruikt, een online opslagruimte voor code. Ten slotte wordt alles met betrekking tot het project gedocumenteerd in Confluence. Een voorbeeld hiervan is het stappenplan om een Stash repository goed te kunnen configureren.

Git

Git is een versie beheer systeem voor code, waarbij diverse versies van een applicatie bijgehouden kunnen worden. Voor het uitwerken van een userstory wordt allereerst een Branch van de Master gemaakt. In de nieuwe Branch wordt vervolgens de userstory uitgewerkt. Na het uitwerken van de userstory wordt de nieuwe functionaliteit in de Branch geverifieerd door een ander. Ten slotte wordt de Branch samengevoegd met de Master als het geverifieerd is.

Java tools

ENOVATION heeft voornamelijk applicaties ontwikkeld op het Java platform met Spring Framework. Tools die hierbij zijn gebruikt zijn:

Spring Boot

In Spring Framework wordt er veel gewerkt met configuraties bestanden in xml formaat. Met Spring Boot wordt het ontwikkelen van een applicatie vereenvoudigd, doordat er minder configuratie nodig zijn. Daarnaast heeft Spring Boot een aantal standaard componenten dat gebruikt kunnen worden.

Maven

Binnen een applicatie zijn er een aantal dependencies[1] waarmee de applicatie is ontwikkeld. Voor het beheren van de dependencies en builds[2] van het project wordt Maven gebruikt. Hierdoor hebben de ontwikkelaars altijd dezelfde dependencies lokaal op PC staan en worden nieuwe dependencies gedownload als deze niet lokaal aanwezig is. Daarnaast kunnen acties geautomatiseerd worden bij het builden van het project. Vaak moet het project ingepakt worden en vervolgens verplaatst worden naar een specifiek map. Met Maven kunnen deze handelingen geautomatiseerd worden.

Jenkins

Een continuous integration system ontwikkeld in Java. Het is een systeem dat onder andere Apache Maven gebaseerde project kan uitvoeren. Hiermee kan er geconfigureerd worden wanneer specifieke acties worden uitgevoerd zoals het uitvoeren van alle testen en build starten. Ook kunnen acties getriggerd worden wanneer een ontwikkelaar zijn code commit.

2.3.3 OTAP omgeving

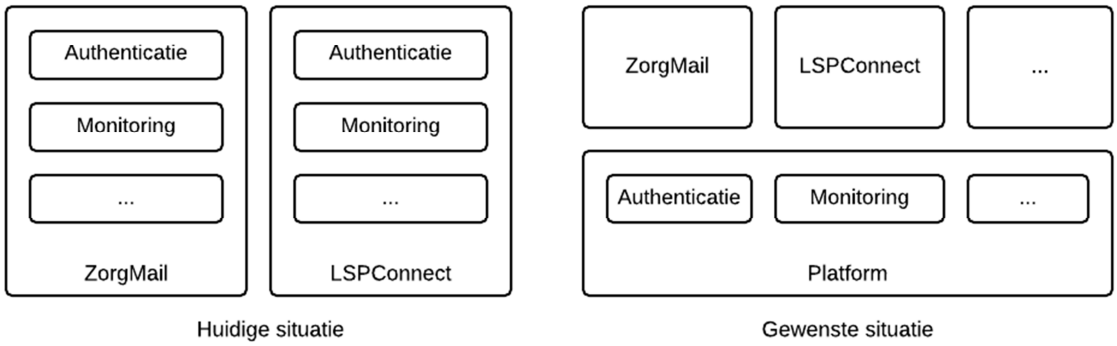
In ENOVATION wordt er gebruik gemaakt van een OTAP omgeving, wat staat voor ontwikkel, test, acceptatie en productie omgeving. In elke omgeving ligt de focus ergens anders. In de ontwikkel omgeving wordt er voornamelijk gefocust op het ontwikkelen van nieuwe functionaliteiten. Het product wordt daarna in de test omgeving geplaatst. In het test omgeving wordt het product getest en geverifieerd door een tester. Vervolgens komt het product in de acceptatie omgeving waarbij de productowner het product verifieerd. Ten slotte wordt het product in het productie omgeving geplaats wanneer de productowner het geaccepteerd heeft.

[1] Dependencies - Afhankelijkheden van een project, meestal wordt hiermee bedoeld welke component
[2] Build - Het project in zijn geheel inpakken tot een file, meestal een Jar file of War file

2.4 Waar wil ENOVATION naar toe met het te ontwikkelen monitoringsysteem?

De applicaties van ENOVATION heeft in de huidige situatie op een aantal plekken overlap, omdat er geen standaard componenten zijn die ingezet kunnen worden voor de applicaties. Voorbeelden hiervan zijn de accountregistratie en login, deze onderdelen komen voor in alle applicaties. Voor het authenticatie onderdeel wordt er een standaard component ontwikkeld in het platform team. Voor het monitoren van de applicaties is er ook geen standaard component dat ingezet kan worden. Het te ontwikkelen monitor systeem wordt uiteindelijk een component in het platform.

In figuur 3 wordt de huidige en gewenste situatie weergegeven. Het ontwikkelen van standaard componenten heeft een aantal voordelen. Ten eerste kunnen componenten opnieuw gebruikt worden voor andere applicaties, waardoor het ontwikkeltraject efficiënter wordt. Ten tweede kan er gefocust worden op het ontwikkelen van nieuwe functionaliteiten, omdat bestaande functionaliteiten al ontwikkeld is in de componenten. Ten slotte wordt het onderhouden van de code vereenvoudigd, want aanpassingen zijn alleen nodig op een plek en dat is in de component zelf. Zonder componenten zouden aanpassingen op meerdere plekken uitgevoerd moeten worden, wat tijdrovend kan zijn.



Figuur 3 schets van huidige en gewenste situatie

2.4.1 Tools

Hieronder wordt beschreven welke tools ENOVATION wil gebruiken voor de deployen[3] van applicaties en het ontwikkelen van het monitoringsysteem. Deze tools worden als uitgangspunten gebruikt bij het ontwikkelen van het monitoringsysteem.

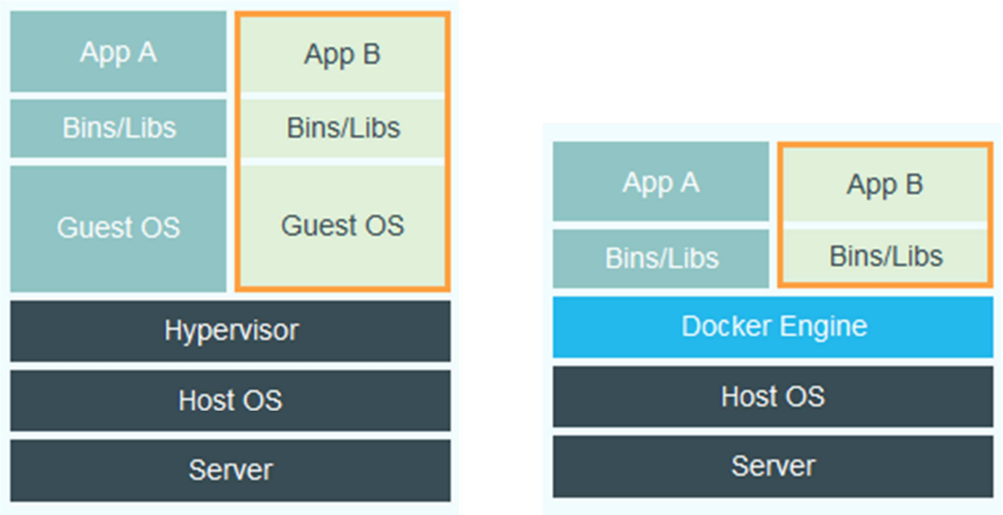
Docker

In de huidige situatie is de deployen van een applicatie een zware taak, want er zijn veel afhankelijkheden. Enkele voorbeelden hiervan zijn:

- Het besturingssysteem op de productieomgeving is anders
- Op de productieomgeving is er een andere versie van Java geïnstalleerd
- Op de productieomgeving wordt een applicatie gehost met een oude versie van een library

Dit zijn de problemen die vaak voorkomen in de praktijk. Met de tool (Docker, 2015) kunnen de problemen opgelost worden. Docker is een tool waarmee applicaties gemakkelijker deployed kunnen worden. Het idee er achter is dat applicaties een eigen container krijgen met daarin alle afhankelijkheden erin. De inhoud van een container kan geconfigureerd naar eigen wensen. De container wordt vervolgens geïsoleerd gehost op de Docker Engine. Op deze manier hebben de containers geen invloed op elkaar en dus maakt het niet uit wat voor afhankelijkheden er zijn van de andere applicaties.

Docker is vergelijkbaar met Virtual Machine, maar met een Virtual Machine komt een besturingssysteem erbij en dat kost neemt schrijfruimte in beslag dan een Docker container. In figuur 4 wordt weergegeven wat het grote verschil is tussen Virtual Machine en Docker.



Figuur 4 links Virtual Machines, rechts Docker

[3] Deploy – applicatie in productie omgeving zetten

Time Serie Database(TSDB)

In de huidige situatie worden meetgegevens opgeslagen in een RRD, maar dit is een verouderd techniek. Voor het te ontwikkelen monitoringsysteem wil ENOVATION een database gebruiken dat geoptimaliseerd is in het opslaan van time serie data, want de meetgegevens die verzameld worden is gebaseerd op een moment opname. ENOVATION heeft een klein vooronderzoek gedaan naar TSDB en daaruit is gebleken dat InfluxDB een geschikt tool is dat gebruikt kan worden bij het ontwikkelen van het monitoringsysteem. De tool (InfluxDB, 2015) is een opensource Time serie database zich gespecialiseerd in het opslaan van metrieke gegevens voor analytisch einddoel.

Dashboard

Voor het gebruiken van een TSDB is er een vervangende tool nodig om de metrieke gegevens te kunnen tonen, want in de huidige situatie wordt Cacti gebruikt dat alleen RRD ondersteund. In het vooronderzoek naar TSDB is een tool (Grafana, 2015) gevonden dat goed aansluit met InfluxDB. Grafana is geavanceerde Dashboard, waarbij metrieke gegevens in een grafiek getoond kunnen worden.

2.4.2 Het beeld van systeemmonitoring

Hieronder wordt beschreven wat het monitoringsysteem uiteindelijk moet kunnen. Het uitvoeren van de opdracht wordt binnen een beperkte tijd gedaan, daarom wordt het project ingedeeld in fases. Bij het uitvoeren van de afstudeeropdracht wordt er in de eerste instantie gefocust op fase 1: het kunnen verzamelen van meetgegevens en deze vervolgens tonen op één dashboard.

Fase1

Allereerst moet het monitoringsysteem meetgegevens kunnen verzamelen en opslaan zonder dat er handmatig handelingen vereist wordt, want in de huidige situatie wordt er veel handmatige handelingen geëist en dat is tijdrovend. Voor het verzamelen van de meetwaardes moet er een standaard set komen dat overal toepasbaar is. Voor het kunnen monitoren van specifieke meet waardes moet er slechts een kleine aanpassing in de code gedaan worden. Er wordt gestreefd naar conventie over configuratie. Hierdoor wordt er minder onderhoud vereist van het monitoringsysteem. Na het kunnen opslaan van de gegevens moet het getoond worden op een dashboard.

Fase2

In fase wordt er gefocust op het kunnen versturen van meldingen en alarmeringen, zodat beheerders op de hoogte gehouden worden wanneer het dreigt mist te gaan. Een voorbeeld hiervan is wanneer de aantal database connecties blijft stijgen en daarna niet daalt, dan is er een vermoeden dat de database connectie niet goed wordt afgesloten. Op den duur zou het de performance van het systeem kunnen belemmeren.

Fase3

In de laatste fase moet het systeem proactief kunnen reageren, zodat bepaalde handelingen geautomatiseerd worden. Een voorbeeld hiervan is het automatische kunnen opschalen van een applicatie om de belasting te kunnen verspreiden over meerdere instanties.

Deel III: Conclusie

Aan de hand van het onderzoek kan er antwoord gegeven worden op de deel vragen. Met de deelvragen kan er vervolgens een antwoord worden gegeven op de hoofdvraag. Hieronder wordt eerst antwoord gegeven op de deelvragen en vervolgens een antwoordt op het hoofd vraag.

Deelvraag 1: Wat wordt er verstaan onder systeem monitoring?

Onder systeem monitoring wordt er verstaan een systeem continue in de gaten houden, waarbij meetwaardes worden verzameld om te kunnen anticiperen in de toekomst. Meest bekende meetgegevens bij systeem monitor zijn processor kracht en geheugen gebruik.

Deelvraag 2: Hoe ziet de huidige situatie van systeem monitoring eruit?

Het huidige monitor systeem van ENOVATION bestaat uit diverse componenten. Deze zijn Nagios, Monitor Demon, Statistic Collector, RRD, Cacti. In het huidige monitor systeem worden er veel onderhoud en handmatige handelingen vereist om de systemen te kunnen monitoren. In de toekomst wil ENOVATION migreren naar een architectuur met micro servers. Hierdoor is het huidige systeem niet geschikt voor de toekomst.

Deelvraag 3: Hoe ziet de ontwikkelstraat van ENOVATION eruit?

ENOVATION ontwikkelt voornamelijk applicaties op de Java platform met Spring Framework. Er worden diverse tools gebruikt om het ontwikkeltraject te ondersteunen. Verder wordt er gewerkt met de software methodiek SCRUM.

Deelvraag 4: Waar wil ENOVATION naar toe met het te ontwikkelen monitoringsysteem?

ENOVATION wil voor veel voorkomende onderdelen in de applicaties componenten ontwikkelen, want deze kunnen opnieuw gebruikt worden in de applicaties. Daarnaast is het gebruiken van componenten efficiënter voor het ontwikkeltraject, want er wordt tijd bespaard. Hierdoor kan er gefocust worden op het ontwikkelen van nieuwe functionaliteiten.

Hoofdvraag: Waar moet er rekening mee gehouden worden bij het ontwikkelen van een monitoringsysteem?

Bij het ontwikkelen van het monitoringsysteem moet er rekening mee gehouden worden met microservices, want ENOVATION wil in de komende tijd migreren naar een architectuur met microservices. Wegens de beperkte tijd dat er aan het project besteedt kan worden is de focus in de eerste instantie om metrics te kunnen opslaan en te tonen. Het is belangrijk dat metrics op een eenvoudig manier verzameld kunnen worden, want anders kost het veel tijd om dit mogelijk te maken en dan is het geen verbetering meer t.o.v. het huidig systeem.

Bijlagen

Bijlage A - Literatuurlijst

A Monitoring Maturity Model. (2015, Februari 20). Opgehaald van kartar: http://www.kartar.net/2015/01/a-monitoring-maturity-model/

Docker. (2015, February 20). Opgehaald van Docker.com: https://www.docker.com/

Encyclo. (2015, Februari 17). *Encyclo*. Opgehaald van encyclo.nl: http://www.encyclo.nl/begrip/Monitoringsysteem

Ensie. (2015, Februari 18). *ensie*. Opgehaald van ensie.nl: https://www.ensie.nl/bart-hellinga/monitoren

grafana. (2015, February 24). Opgehaald van http://grafana.org/: http://grafana.org/

InfluxDB. (2015, February 24). Opgehaald van InfluxDB.com: http://influxdb.com/