Research Project Toegepaste Informatica

Datadog (stock symbol: DDOG), 2021. © 2023 CompaniesLogo.com

Voorbereiden op *scalability*: Datadog monitoring van WeaveWorks Sock Shop

SNB 3

Mateo Lemmens

Tomas Soors

Victor Beckx

Rasmus Leseberg

Academiejaar 2022-2023

Abstract

Datadog is een monitoringservice voor *cloud-scale* applicaties, die monitoring van servers, databases en andere functies aanbiedt via een *SaaS*-gebaseerd platform. Als integratie van de WeaveWorks Sock Shop kan Datadog relevante data monitoren, die door middel van bepaalde *metrics* en alerts de webshop voorbereidt op vlak van *scalability*. Deze paper onderzoekt features, integraties en alertsystemen van Datadog, met als doel om een overzicht van bruikbare data te scheppen aan de hand van verschillende userscenario's. De analyse onderzoekt de correlaties tussen de *metrics* van data, gebruikt gesimuleerde userscenario's om bepaalde microservices onder druk te zetten, en kijkt naar alertsystemen die relevante waarschuwingen weergeven.

De leidende onderzoeksvraag klinkt als volgt: "Welke *metrics* kunnen via Datadog worden verwerkt om een website/Sock Shop voor te bereiden op vlak van *scalability*?". Meer specifiek, deze onderzoeksvraag zet dus de integraties, *metric* types en alertsystemen van Datadog uiteen, met ondersteuning van gerelateerde deelvragen.

Het grootste deel van de research berust op een technisch onderzoek dat loadtesting scripts zal gebruiken om verschillende userscenario's te simuleren, met als doel om de microservices van de Sock Shop te targeten en te stresstesten. Vervolgens focust de research zich ook op het correct gebruik en instellen van alertsystemen, zodat de microservices vroegtijdig kunnen communiceren dat die onder druk staan. Door te onderzoeken welke *KPIs* van belang zijn vanuit de alertsystemen is het mogelijk om de Sock Shop voor te bereiden op grotere bezoekersaantallen. Het belang hiervan is dat een groot aantal bezoekers de beschikbaarheid en functionaliteit van de webshop niet mag beperken.

Uiteindelijk blijkt uit het onderzoek dat het gebruik van loadtestingscripts, in samenwerking met de Datadog integraties en monitoringtypes, de Sock Shop kan voorbereiden op een *scalability* scenario. Met betrekking tot de vooropgestelde KPI's die ondervonden zijn door meerdere Locust-testen op de Sock Shop, zou het door middel van Datadog alerts mogelijk zijn om de Sock Shop voor te bereiden om bij te schalen.

Inhoudsopgave

Lijst van gebruikte figuren

Lijst van gebruikte tabellen

Lijst van gebruikte afkortingen

Afkorting	Definitie
SaaS	Software as a Service
KPI	Key Performance Indicators
KP	Key Performance
API	Application Programming Interface
RAM	Random Access Memory
CPU	Central Processing Unit
OS	Operating System
CI/CD	Continuous Integration/Continuous Delivery
HPA	HorizontalPodAutoScaler
IoT	Internet of Things
UI	User Interface
VM	Virtual Machine
DB	Database

1 Onderzoeksvraag en hypothese

1.1 Hoofdvraag en deelvragen

Deze researchpaper onderzoekt welke *metrics* Datadog verwerkt om een microservice-gebaseerde webshop voor te bereiden op vlak van *scalability*. De onderzoeksvraag is:

"Welke metrics *kunnen via Datadog worden verwerkt om een website/Sock Shop voor te bereiden op vlak van* scalability?*"*

Om deze onderzoeksvraag te beantwoorden, zijn er verschillende deelvragen opgesteld die zich richten op de aparte features van Datadog:

1. *Integrations*: "Welke integraties biedt Datadog aan in verband met de Sock Shop voor het verzamelen van *metrics*, en welke features van deze integraties zijn relevant?

2. Dashboards: "Welke *metrics* zijn van belang in verband met *scalability*, en hoe kunnen deze verzamelde *metrics* gecorreleerd worden binnen Datadog?"

3. Monitoring: "Welke combinatie van monitortypes binnen Datadog kunnen ingesteld worden ter voorbereiding van *scalability*?"

Deze deelvragen vormen de leidraad van het onderzoek en zullen tijdens het technisch onderzoek ondersteund worden aan de hand van toegepaste userscenario's.

1.2 Hypothese

De stelling van dit onderzoek is dat correct alerts instellen de WeaveWorks Sock Shop voorbereidt op een *scalability* scenario. De Datadog integraties verzamelen de belangrijkste *KPI's* van specifieke containers, en de monitoringtypes bieden de mogelijkheid aan om vroegtijdig te reageren. Met behulp van een overzichtelijk dashboard is het voor de gebruiker mogelijk om de microservices apart te analyseren en kritische alerts op de juiste plek te tonen.

Met behulp van loadtestingscripts is het mogelijk om de microservices tot hun limiet te belasten, en daardoor de *KPI's* op de juiste hoeveelheid in te stellen. Door herhalende loadtesting scripts uit te voeren is het mogelijk om gemiddelde *KPI*-waarden te bepalen, en deze te gebruiken om de Sock Shop voor te bereiden om te schalen. De loadtesting scripts zijn gebaseerd op realistische userscenario's die specifieke functies van de microservices targeten, en de scripts zorgen voor geïsoleerde testomgevingen om de microservices onder druk te zetten.

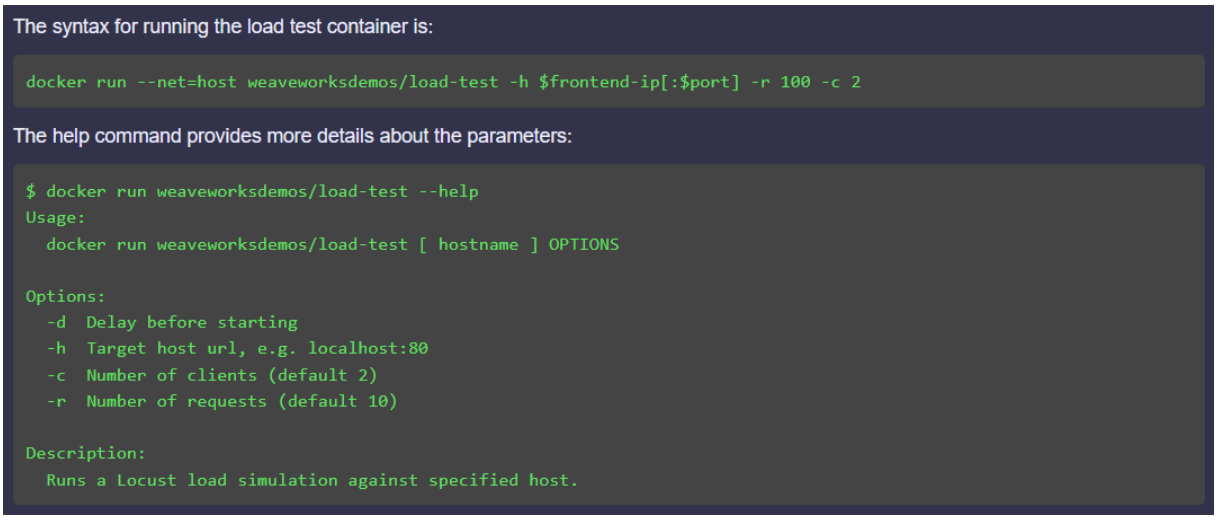
Uiteindelijk is het door de resultaten van het technisch onderzoek mogelijk te bepalen precies wanneer het tijd is om voor de Sock Shop te schalen voor hogere bezoekersaantallen.

2 Onderzoeksmethode

De onderzoeksmethode voor deze researchpaper is gebaseerd op een technisch onderzoek wat gebruik maakt van loadtesting scripts. Het Datadog-platform zal de verzamelde data van de tests tonen in een dashboard, om zo bepaalde *KPI's* als alerts vast te leggen.

Voor de containers of microservices waarvoor integraties bestaan worden deze ook gebruikt. Alle containers krijgen bijvoorbeeld de Docker-integratie, de databases krijgen de MongoDB en MySQL integratie, de 'front-end' (WeaveWorks) microservice krijgt de NodeJs-integratie en de 'edge-router' microservice krijgt de Traefik-integratie.

Gerichte loadtesting scripts zullen zoals vermeld de microservices van de webapplicatie onder druk zetten. Deze scripts zijn gebaseerd op realistische userscenario's, die gericht bezoekersverkeer simuleren en specifieke functies van de webshop targeten. Scripts zorgen ervoor dat deze testen gemakkelijk en consistent zijn in uitvoering. Een voorbeeld van een loadtesting container wat de 'front-end' microservice belast is te zien in Figuur 1.



Figuur 1: Voorbeeld van een loadtestscript [1]

Het voorbeeld in Figuur 1 maakt gebruik van een WeaveWorks-container als loadtest om de 'front-end' microservice te belasten, maar om bepaalde functies van de webshop te isoleren zijn zelfgeschreven scripts waarschijnlijk eerder van toepassing. Het doel zal zijn om de kritische functies van de webshop te isoleren, zoals bestellingen plaatsen, inloggen, de cataloog van producten bekijken, zodat de limieten van de kritische infrastructuur vastgelegd kunnen worden.

Het volgende hoofdstuk zal de integraties en monitoringtypes van Datadog uiteenzetten, met als introductie een dieper inzicht tot microservices, containers en schaalbaarheid.

3 Literatuurstudie

De hoofdvraag is gefocust op *scalability* binnen een webapplicatie context, waar de Sock Shop webapplicatie uit containers bestaat die microservices draaien. *Scalability* heeft daardoor een specifieke context en definitie. Dit hoofdstuk bespreekt kort wat de inhoud van deze context is, en zet microservices, containers, *scalability*, en diverse Datadog-functies uiteen.

3.1 Microservices en containers

Een microserviceframework bestaat uit microservices die in dit geval binnen containers draaien. Dit creëert een systeem dat de knelpunten van een centrale database vermijdt. Het maakt ook *CI/CD*-pipelines voor applicaties mogelijk en moderniseert de *technology stack* van de applicatie. [2]

Containers zijn een lichtgewicht, efficiënte oplossing voor applicaties om tussen omgevingen te communiceren en onafhankelijk te werken. Alles wat nodig is (behalve het gedeelde OS op de server) om de applicatie uit te voeren is verpakt in het containerobject. Dit houdt de code, runtime, systeemafhankelijke tools, library's en andere *dependencies* in. Containers zijn een standaardmanier om microservices te implementeren, omdat de microservice als containerimage verpakt is. Elke service bestaat binnen een aparte container, en schalen gebeurt op basis van gewijzigde containerinstanties.

Microservices daarentegen zijn een populaire optie voor het moderniseren van applicaties. Enerzijds zijn ze kostenbesparend, waardoor ze ideaal zijn voor kleinere bedrijven. Anderzijds kunnen microservices ook cloudnative zijn, zoals de Sock Shop in dit geval, waardoor ruimte wordt bespaard die normaal nodig is voor *on-premise* systemen. Wat nog crucialer is, microservices omzeilen de problemen van monolithische applicaties. De problemen van monolithische applicaties zijn typisch:

- Ze zijn moeilijk en langzaam te onderhouden en te testen.
- Het repareren van een functie met bugs of onderhoud betekent downtime voor de hele applicatie.
- Moeite met het beheersen van verschillende programmeertalen.

Toch is schaalbaarheid het belangrijkste voordeel van microservices in vergelijking met monolithische applicaties; de capaciteit van microservices kan worden geconfigureerd om dynamisch te schalen afhankelijk van navraag [3].

3.2 Wat is microservice *scalability*?

Containerservers schalen (*scalen)* wanneer hun systeemresources (CPU, RAM, netwerkbandbreedte, enz.) worden verhoogd om te voldoen aan de navraag die het systeem vereist. Containers kunnen hun resources reguleren op twee manieren:

1. Verticaal schalen: de applicatie capaciteit vergroten door de capaciteit van de servers te vergroten, virtueel of fysiek. Voorkeur ligt hier bij *stateful* apps, omdat deze vereisen dat de klantinformatie tussen sessies moet worden bewaard om te werken.

2. Horizontaal schalen: het aantal serverinstanties vergroten om de vraag te beheren. Voorkeur voor *stateless* applicaties omdat deze geen klantgegevens opslaan.

Stateless en *stateful* protocollen verschillen van elkaar. Een *stateless* systeem stuurt een *request* naar de server en stuurt het antwoord (of de status) van de server terug zonder enige informatie op te slaan. Aan de andere kant verwachten *stateful* systemen een antwoord, volgen deze informatie op, en verzenden het verzoek opnieuw als er geen antwoord wordt ontvangen [4].

In het geval van horizontaal schalen is er ook het idee van *downscaling*, namelijk het verminderen van serverinstanties wanneer de navraag daalt. Op deze manier kan schalen ook de kosteneffectiviteit van microservices verbeteren [3].

Vervolgens zullen bepaalde integraties van Datadog besproken worden om meer inzicht te bieden in de opbouw van het technische onderzoek.

3.3 Deelvraag 1: *Integrations*

Integrations: *"Welke integraties biedt Datadog aan in verband met de Sock Shop voor het verzamelen van* metrics*, en welke features van deze integraties zijn relevant?"*

Om gegevens correct te kunnen monitoren en het type van *metrics* te bepalen, is het belangrijk om eerst te begrijpen hoe Datadog gegevens verzamelt. Daarnaast is het ook belangrijk om te begrijpen uit welke onderdelen de Sock Shop is opgebouwd om zo de juiste integratie features te bepalen.

3.3.1 Wat zijn integraties?

Integraties zijn de fundamentele bouwstenen voor de verzameling van data binnen Datadog. Integraties bestaan uit verschillende tools en features die de mogelijkheid bieden om een bepaalde service te meten en gegevens hierover te verzamelen. Binnen Datadog bestaan er meer dan 600 verschillende integraties, beschikbaar voor diverse services zoals: Docker, Apache, en MySQL onder andere.

Datadog biedt deze integraties aan op drie verschillende manieren. De meest voorkomende manier is een integratie die via de Datadog agent draait. Een ontwikkelaar heeft ook de mogelijkheid om binnen het Datadog platform een integratie te ontwikkelen op maat van hun applicatie.

De verzamelde gegevens noemen we ook wel logs of *metrics*. Naast data verzamelen is het ook de verantwoordelijkheid van de integraties, samen met de Datadog agent, om de data door te sturen naar een centrale plaats. Deze centrale plaats is het online Datadog platform [5].

3.3.2 De Datadog agent

De Datadog agent is een open source applicatie die de basis vormt om gegevens te loggen over een bepaald systeem. De software is online beschikbaar via het Datadog platform en de broncode is gepubliceerd op GitHub [6].

De agent wordt geconfigureerd op ieder individueel systeem waar een verzameling van informatie op moet gebeuren, en waarvan de informatie beschikbaar gesteld moet zijn op het online Datadog platform. De agent kan op diverse besturingssystemen draaien (Windows, Linux, macOS), en is met andere woorden de brug tussen enerzijds het systeem, en anderzijds de onlineomgeving van Datadog. Daarnaast vormt deze ook de basis voor de installatie en configuratie van de Datadog integraties. Zonder de agent kunnen bepaalde integraties niet werken. Ten slotte gaat de agent via de Datadog *API* de verzamelde informatie die van de integraties komen doorsturen naar een centrale plaats, namelijk het online Datadog platform [7] [8].

De kracht van de Datadog agent is dat er binnen een omgeving meerdere agents tegelijk op diverse systemen kunnen draaien. Ze bieden een ondersteuning aan de integraties, en ze sturen allemaal informatie door naar een centrale plaats.

3.3.3 Relevante integraties voor de Sock Shop

Via de Datadog agent is het mogelijk om systeemrelevante gegevens te meten, bijvoorbeeld RAM en CPU gebruik. Deze zijn ook in het online platform opgenomen en kunnen eventueel relevant zijn om een *scalability* scenario voor te bereiden.

De Sock Shop is hoofdzakelijk opgebouwd uit drie elementen. Er zijn twee databases: MongoDB en MySQL, de rest van de services draaien allemaal in verschillende Docker containers. Aangezien dat MongoDB en MySQL op verschillende servers draaien gaat er op de agent van die servers enkel de integratie voor dat typen database geconfigureerd worden. Alle andere servers, waar de verschillende services van de Sock Shop onderverdeeld zijn over Docker containers, krijgen de Datadog Docker integratie geïnstalleerd.

Bepaalde services van de Sock Shop zouden ook met behulp van meer specifieke integraties gemonitord kunnen worden, bijvoorbeeld zou de NodeJs integratie voor de ‘front-end’ service relevant kunnen zijn, of de Traefik integratie voor de ‘edge-router’ service.

3.4 Deelvraag 2: Dashboards

Dashboards: *“Welke* metrics *zijn van belang in verband met* scalability, *en hoe kunnen deze verzamelde* metrics *gecorreleerd worden binnen Datadog?”*

Dashboards zijn handig om *KP metrics* te visualiseren en te volgen. Er zijn een aantal features die het opzetten van een dashboard mogelijk maken, zoals Widgets, *Querying, Functions, Template Variables,* en *API* [9]. In dit onderzoek komen voornamelijk Widgets aan board, wat bouwstenen zijn om een visueel dashboard op te bouwen.

Dashboards hebben een raster lay-out, wat gemakkelijk met drag-and-drop te navigeren is, en bieden ook aanvullende opties om bepaalde datacollectie te debuggen. Realtime vervolg van de draaiende microservices en hun alerts is het uiteindelijke doel van dit onderzoek, om ervoor te zorgen dat de Sock Shop op tijd geschaald kan worden in het geval van toenemend bezoekersverkeer [9].

3.4.1 Welke metric types zijn er?

Elke *metric* die bij Datadog wordt ingediend, moet een type hebben. Het type van een *metric* beïnvloedt hoe de *metric*-waarden worden weergegeven wanneer ze worden opgevraagd, en wat de grafische mogelijkheden binnen Datadog zijn. Het type *metric* wordt weergegeven in het detailzijpaneel voor de gegeven *metric* op de 'Metrics Summary' pagina [10].

De volgende *metric* types zijn door Datadog geaccepteerd:

- COUNT
- RATE
- GAUGE
- DISTRIBUTION

Welke type van *metric* toepasselijk zal zijn hangt van de microservice af, en wordt dieper onderzocht tijdens het technisch onderzoek. Het volgende stuk zal de monitoringtypes van Datadog uiteenzetten.

3.5 Deelvraag 3: Monitoring

Monitoring: *"Welke combinatie van monitortypes binnen Datadog kunnen ingesteld worden ter voorbereiding van* scalability*?"*

3.5.1 Wat zijn monitors binnen Datadog?

Zoals eerder vermeld in dit onderzoek is het verzamelen van gegevens via de integraties van Datadog, en het correleren van deze gegevens met de dashboards van Datadog van groot belang voor monitoring en *scalability*. Het is echter niet voldoende om alleen gegevens op een centrale locatie te verzamelen. Het is van belang om te weten wanneer er verdachte of kritieke veranderingen optreden in deze gegevens. Hiervoor biedt Datadog de functie 'Monitors' aan.

Deze monitors controleren actief de verzamelde gegevens, de mogelijke integraties die van toepassing zijn op de systemen, netwerkeindpunten en meer [11]. Gebruikers kunnen monitors aanmaken om specifieke waarschuwingen te creëren. Dit kan door gebruik te maken van beschikbare monitortypes die Datadog aanbiedt en gepersonaliseerde *KPIs* die voor de gebruiker van belang zijn. Alle monitors die zijn aangemaakt worden weergegeven onder 'Manage Monitors' en kunnen daar worden gemaakt, verwijderd, gekopieerd, gedempt of aangepast.

Automatisch ingestelde notificaties voor aangemaakte monitors kunnen de gebruiker of het team op de hoogte stellen wanneer er een verdachte of kritieke verandering optreedt in de verzamelde gegevens. Dit kan via e-mail, Slack, of andere integraties van Datadog. Deze notificaties worden alleen verstuurd wanneer een ingestelde *KPI* wordt overschreden. De notificatie kan diverse informatie bevatten, zoals details over de overschreden *KPI* en eventuele screenshots. De mobiele Datadog applicatie biedt ook de mogelijkheid om deze monitors op te volgen [11].

3.5.2 Welke monitor types biedt Datadog aan?

Monitors kunnen ingesteld worden met een bepaald monitortype, waarvan het type monitor afhankelijk is van de waarschuwingssoort die de gebruiker of het team wil ontvangen. In dit onderdeel worden verschillende monitortypes besproken met de meeste relevantie tot *scalability*.

De hostmonitortype gaat bijvoorbeeld controleren of de geconnecteerde host via Datadog agent data aan het versturen is naar het Datadog platform. Hierdoor kan een alert onmiddellijke notificaties versturen naar de gebruiker of het team, als één of meerdere hosts geen data meer verstuurt naar de Datadog server [12]. Dit wil dan zeggen dat één of meerdere hosts uitgevallen zijn door overbelaste services. Op tijd reageren op monitors zorgt voor een snelle responstijd en helpt om de Sock Shop voor te kunnen bereiden op vlak van *scalability*.

Verder is er de *metricmonitortype*. Dit type monitor gaat zoals alle monitortypes via de Datadog agent informatie verzamelen. Hier gaat het specifiek over het verzamelen van een continue stroom van *metrics.* Dit is handig voor het monitoren van drempelwaarden of KPI's. De monitor gaat de gebruiker alarmeren wanneer een bepaalde drempelwaarde of KPI die ingesteld is via het *metricmonitortype* is overschreden. Dit kan helpen voor het op tijd bij schalen van services zodat deze niet overbelast raken.[12]

Het volgende monitortype interessant voor *scalability* is het *anomalymonitortype*. Dit monitortype gaat automatische detecteren wanneer er een *metric* zich anders gedraagt dan vroeger met in het achterhoofd welke dag het is, welk uur het op die dag het is en verschillende trends. Dit is handig voor het monitoren van groeiende *metrics.* Bijvoorbeeld voor het monitoren van een stabiele groei van websitegebruikers. Door het continu groeien van gebruikers gaat de KPI verouderd zijn en hier kan deze monitor bij helpen. Deze monitor geeft in het begin een slecht resultaat, maar met tijd geeft deze monitor mooie KPI's weer om deze dan te gebruiken binnen het *metricmonitortype*.[12]

Om de Sock Shop zo precies mogelijk te monitoren is de Datadog RUM (*Real User Monitoring*) feature ideaal, omdat deze Datadog feature vanzelf de meest relevante *metrics* samenstelt op basis van usersessies die opgenomen worden van Datadog.

3.5.3 Datadog RUM

Real User Monitoring (RUM) geeft *end-to-end* inzicht in de realtime activiteit van individuele gebruikers. Het gegenereerde dashboard heeft op verschillende vlaktes meerwaarde voor de Sock Shop, namelijk kunnen de volgende aspecten onder de loep worden genomen [13]:

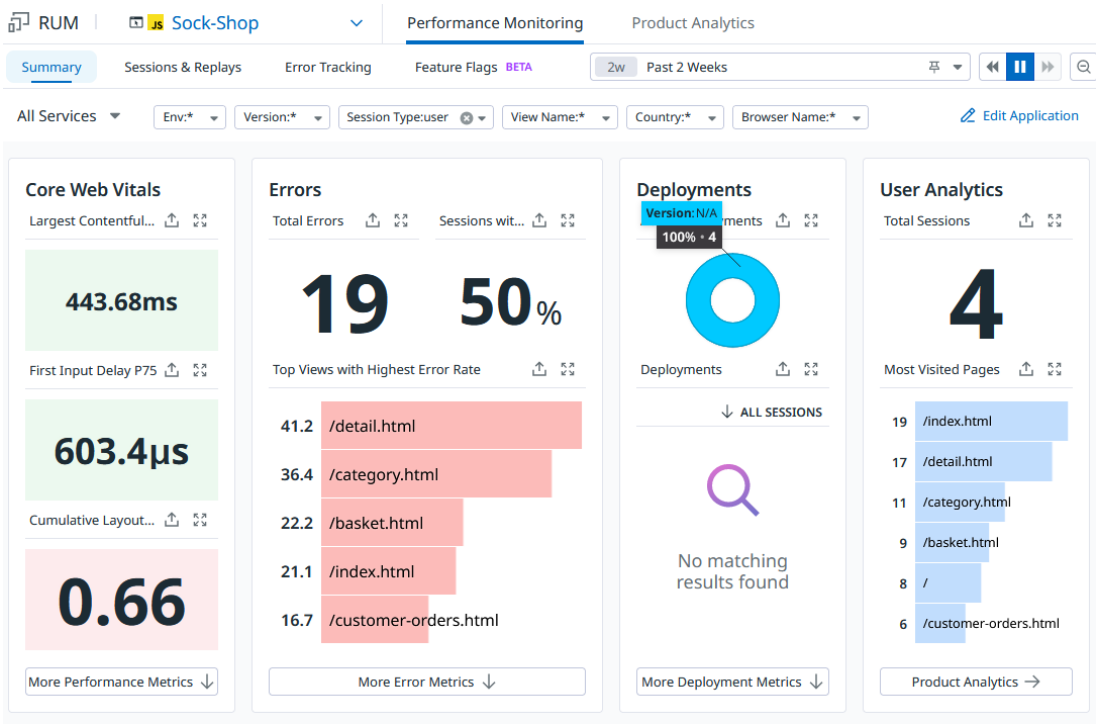
- Performance: De prestatie van webpagina's door gebruikersacties en uitgevoerde netwerk-*requests*, hoe performant de front-end code in het geheel is.

- Error Management: 4XX's, 5XX's en andere belangrijke fouten opsporen terwijl die gebeuren. Dit kan ook door versieverschillen heen vervolgd worden.

- *Analytics*/*Usage*: Inzicht tot wie de applicatie gebruikt (land, apparaat, OS), hoe de applicatie gebruikt wordt, en welke *trajectories* de gebruiker volgen.

- Support: Informatie over individuele gebruikerssessies kunnen helpen om problemen op te lossen.

Een RUM dashboard geeft deze informatie in een gebruiksvriendelijke wijze weer:



Figuur 2: RUM dashboard van Datadog

Op deze plek zou het in principe mogelijk zijn om alerts in te stellen voor bepaalde KPIs, zodat de Sock Shop bij kan schalen als het nodig zou zijn. Het volgende onderzoek zal deze KPI's vastleggen.

4 Uitvoering

De uitvoering bestaat uit twee delen. Het eerste deel bevat zich met een Sock Shop set-up zonder een Kubernetes toepassing, en het tweede deel vergelijkt de resultaten met een Sock Shop set-up waarbij Kubernetes K3s wel gebruikt wordt. De loadtesting scripts die tijdens het onderzoek zijn gebruikt, zijn met Locust uitgevoerd.

4.1 Loadtesting scripts

Om de loadtesten voor de Sock Shop uit te voeren zijn er verschillende loadtesting-tools gebruikt. Hieronder staat een beschrijving van twee tools die zijn geïmplementeerd in de Jenkins-pipeline om de tests uit te voeren. Beide tools zijn open source, waardoor ze soms wat moeilijker te implementeren zijn maar wel veiliger en algemeen beter ontwikkeld.

4.1.1 Locust

Locust is een loadtestingtool die het mogelijk maakt om gebruikersgedrag na te bootsen en te implementeren aan de hand van Python-code. Hiermee kan een tester eenvoudig testscenario's opstellen voor specifieke toepassingen, zoals in ons geval de Sock Shop. Er is een mogelijkheid om de testen direct via de *command line* uit te voeren. Op deze manier zal locust een lokale webserver opzetten, waar verschillende parameters, zoals het aantal gebruikers, ingesteld kunnen worden. Tijdens het uitvoeren van de tests zijn realtime gegevens zoals errors beschikbaar in een automatisch aangemaakte UI. Na afloop genereert locust automatisch diverse grafieken en rapporten, die in verschillende formaten, zoals CSV beschikbaar zijn om te downloaden.

Dit is een screenshot van de header van de gebruikte locust klasse:

```
class WebsiteUser(HttpUser):
    wait_time = between(1, 10)
    host = "http://10.128.102.2"

    random_priority = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

    @task(20)
    def index(self):
        with self.client.get(
            "/index.html",
            catch_response=True) as resp:
            pass
```

Figuur 3: Locust WebsiteUser klasse header

Een locust loadtestingscript is opgedeeld in meerdere taken, waarvan de taak ‘index’ te zien is in Figuur 3. Deze opgelijste taken behoren tot een specifieke klasse, in ons geval de WebsiteUser klasse, en kunnen ingesteld worden voor specifieke doeleinden.

Binnen locust zijn er diverse variabelen die gebruikt zijn om een zo realistisch mogelijk userscenario te bekomen. Hieronder staat een korte beschrijving van diegene die in het onderzoek een invloedrijke rol spelen:

- wait_time: Deze indiceert een random tijd tussen het uitvoeren van 2 taken.
- Taak gewicht: bijvoorbeeld zoals gezien in Figuur 3, "@task(20)".

Iedere taak heeft een nummer tussen haakjes. Dit wijst op het gewicht van een taak. Locust kiest de prioriteit van een taak aan de hand van het getal tussen de haakjes. Bij bepaalde taken, zoals de homepage bezoeken, ligt de prioriteit bij 20 omdat dat de meest bezochte pagina zal zijn. Bij andere taken zal de prioriteit random gekozen worden, met een getal tussen de 1 en 20. Op die manier zal elke gebruiker een unieke *trajectory* hebben. Het volledige loadtesting script is te zien in bijlage A. Om de testen niet alleen via *command line* uit te voeren komt Taurus in beeld, een handige open source testingtool.

4.1.2 Taurus

Taurus is een open source testtool die ondersteuning biedt voor het uitwerken van testscenario's. Taurus zelf kan overweg met diverse open source testingmodules zoals locust, en selenium onder anderen. Via taurus kan een tester echt gebruikersgedrag nabootsen en *real-world* scenario's uitbouwen.

Om Taurus te configureren is er een yaml configuratiebestand die de testen van een of meerder testtools aanroept. Daarna is er de mogelijkheid diverse parameters in te stellen. Enkele van de gebruikte staan hieronder toegelicht (zie variabelen in paragraaf 4.2), omdat deze van groot belang zijn voor het verloop van de testen. Ook het uitvoeren van Taurus moet via de *command line*, taurus zelf geeft een low level dashboard met test gegevens weer. Eens de testen klaar zijn genereerd het diverse rapporten die in een sub map.

In het project is BlazeMeter samen met Taurus geïmplementeerd. BlazeMeter is een tool wat samenwerkt met Taurus en na uitvoering van de taurus-scenario's de gegevens verzamelt en in realtime dashboards beschikbaar stelt. Het voordeel aan BlazeMeter is dat de resultaten in een webinterface zichtbaar zijn, een verkort voorbeeld daarvan is te zien in bijlage C. Daarnaast kan er ook een koppeling aan een account gebeuren met een API-token om duidelijke overzichten van alle testen te krijgen. Dit geeft ook het voordeel dat resultaten makkelijk met anderen te delen zijn. Enkele waarden spelen een invloedrijke rol bij de Taurus configuratie, zoals 'hold-for', 'ramp-up', en '*concurrency*', en het volledige configuratiebestand is te zien in bijlage B.

```
execution:
- executor: locust
  concurrency: 1500
  hold-for: 10m
  ramp-up: 1m
  scenario: locust
```

Figuur 4: Taurus configuratie

4.2 Pre-Kubernetes

Pre-Kubernetes bevond de Sock Shop zich in een stadium waarin het draaide op het vSphere-platform zonder enige container-*orchestration* tool. In deze context was elke service toegewezen aan een specifieke server, waarbij onderlinge connectiviteit werd bewerkstelligd via het lokale netwerk van de servers. Dit werkmodel was ook afhankelijk van een externe controller-VM, die verantwoordelijk was voor het uitvoeren van testscripts en gerelateerde taken.

Voor het Pre-kubernetes gedeelte zijn de variabelen als volgt:

4.2.1 Overzicht variabelen

Overzicht variabelen		
Controle/Constante	***Afhankelijk***	***Onafhankelijk***
Locust script	*Throughput* (gem. hits/s)	Aantal gebruikers (totaal)
wait_time (1,10)	Responstijd (gem. ms)	
hold-for: 10m	90% Responstijd (gem. ms)	
ramp-up: 1m	Bandbreedte (gem. MiB/s)	

Tabel 1: Overzicht variabelen pre-K3s

Het loadtesting script, en de instelling daarvan, blijft gedurende het onderzoek constant. Het aantal gebruikers neemt tijdens het onderzoek toe, om zo de reactie te zien bij de afhankelijke variabelen.

De ramp-up variabele is het gemakkelijkste te verwoorden als de stijging van de gebruikers binnen een bepaalde tijd. Als de ramp-up een minuut zou zijn, zouden het vooropgestelde aantal gebruikers geload worden op de Sock Shop binnen een minuut. Tijdens het onderzoek werd er ook geëxperimenteerd met langere ramp-up waarden, om te kijken of een ramp-up van een minuut de Sock Shop onnodig onder stress zet, maar dat was niet het geval. Bij een ramp-up van een uur was de initiële reactie soortgelijk als bij een ramp-up van een minuut (met dit aantal gebruikers).

Ten tweede is er de hold-for variabele. De naam verwijst naar de duur van de test, en de mate waarin deze de gebruikers ‘vasthoudt’. Om een realistisch resultaat te krijgen is het beter om de testen langer te laten lopen, te werken met piekuren en om de ramp-up te wijzigen naar afwisselende vormen. Dat valt helaas buiten het onderzoek en kan een eventuele toepassing zijn voor toekomstig onderzoek.

Ten laatste is er de wait_time variabele die de gebruikers gaat laten wachten tussen de een en tien seconden voor dat deze een nieuwe request gaan uitvoeren op de Sock Shop. Dit is om het zo realistisch mogelijk op een echte user te laten lijken.

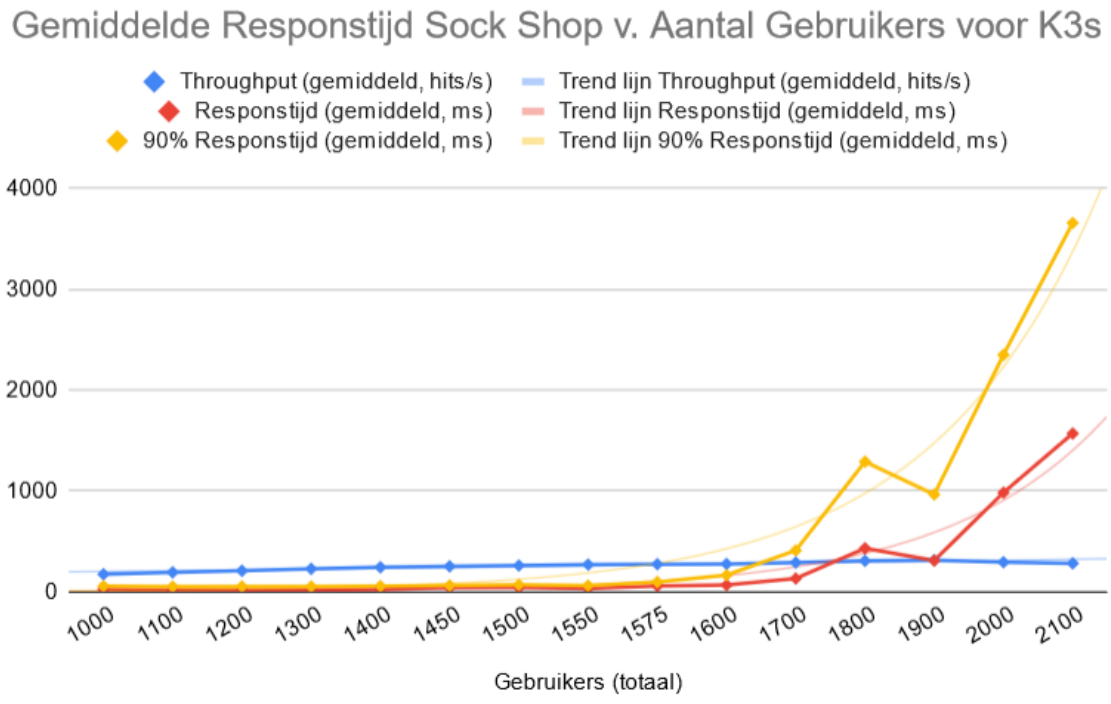
Op de volgende pagina zijn de testresultaten te zien met een gebruikersaantal tussen de 1000 en 2100 gelijktijdige gebruikers.

4.2.2 Resultaten pre-K3s toepassing

Gebruikers (totaal)	*Throughput* (gemiddeld, hits/s)	Responstijd (gemiddeld, ms)	90% Responstijd (gemiddeld, ms)	Bandbreedte (gemiddeld MiB/s)
1000	172.71	23	55	1.19
1100	191.33	13	50	1.69
1200	207.76	15	51	1.76
1300	226.27	16	51	1.99
1400	242.19	23	55	1.71
1450	251.08	42	62	1.67
1500	259.59	41	68	1.85
1550	268.54	32	60	2.33
1575	271.93	56	95	1.58
1600	274.23	66	164	2.42
1700	288.75	130	409	2.24
1800	306.71	430	1290	2.49
1900	312.15	309	964	2.41
2000	292.31	983	2350	2.58
2100	280.49	1570	3660	2.24

Tabel 2: Resultaten loadtesting pre-K3s

4.2.3 Grafische weergave resultaten



Figuur 5: Grafische weergave van de resultaten pre-K3s

De afhankelijke variabelen die door het aantal gebruikers veranderen zijn *throughput*, algemene responstijd, gemiddeld van 90% van de responstijd, en bandbreedte. De *throughput* van de website is het aantal hits/s de Sock Shop aankan. Dus het aantal *requests* dat er per seconde verstuurd wordt.

Vanuit de testen is duidelijk te zien wanneer de responstijd exponentieel begint te stijgen. Dit gebeurt rond de 1700-1800 gebruikers, met ongeveer 300 *requests* per seconde. De responstijd die 90% van de gebruikers ervaren is in dit onderzoek eerder belangrijk, omdat de resterende 10% meestal onrealistische waardes teruggeven. Belangrijk te noteren is wel dat deze waardes afhankelijk zijn van het type loadtest en de infrastructuur. De beperkingen van het onderzoek worden in de evaluatie in detail besproken.

Verder is het interessant te vermelden dat de *throughput* rond de 1900 gebruikers weer begint te dalen, omdat de responstijd een punt heeft bereikt waar de respons zo traag is, dat dit een algemene vertraging veroorzaakt bij het aantal *requests*. De bandbreedte stijgt ook met mate van het aantal gebruikers, maar dat is vanzelfsprekend. De bandbreedte is wel in *Mebibyte*/s gemeten, maar dat verschilt weinig van megabyte.

4.2.4 KPI's pre-K3s

De KPI-waarden die vanuit dit onderzoek vastgelegd zijn berusten op deze specifieke loadtest en de uitvoering daarvan. Met andere soorten loadtests zullen de resultaten verschillen, maar dat valt buiten de scope van het onderzoek. De volgende KPI's zijn vastgelegd op bepaalde punten waar *scaling* nodig zou zijn om responstijden hoger dan 0.1 seconde te vermijden.

- Aantal gebruikers: ~ 1700
- Gemiddelde *throughput:* ~ 300 hits/seconde

Het is wel belangrijk om te vermelden dat de waardes vanuit de responstijd zullen verschillen met de echte ervaring van de users. Het respons van een GET-request van de server zal altijd sneller zijn dan het inladen van een index pagina bijvoorbeeld, waarbij meerdere elementen en figuren ook moeten laden.

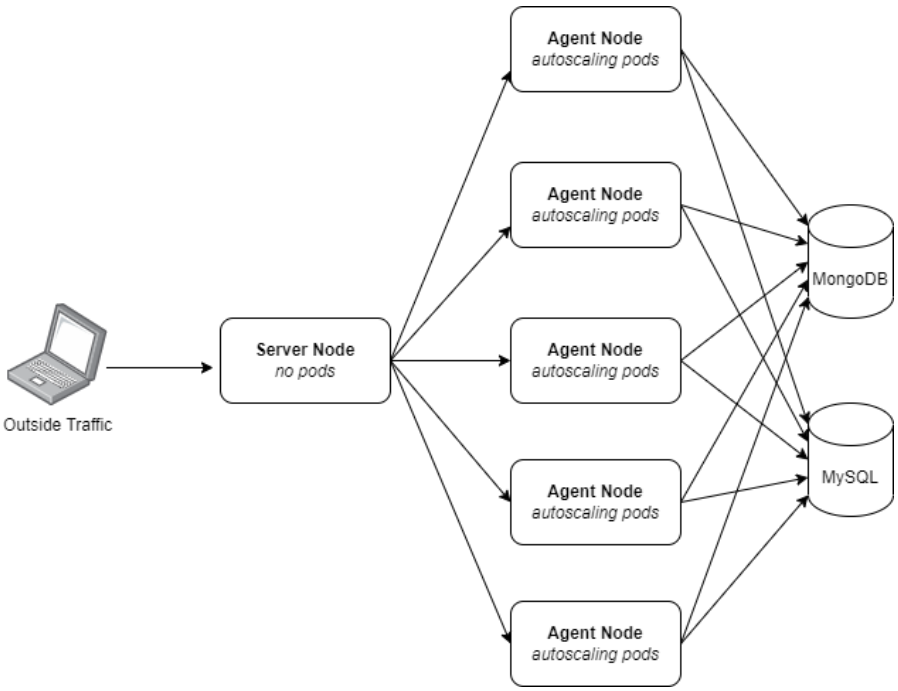
4.2.5 Conclusie Pre-Kubernetes

In de conventionele vSphere-opstelling waar geen container-*orchestration* wordt gebruikt, kan de Sock Shop ongeveer 1575 gebruikers verwerken met een responstijd die < 0.1s ligt. Belangrijk is wel te noteren dat deze waardes afhankelijk zijn van de aard van loadtest en infrastructuur. Zou de infrastructuur en het realtime gebruik van echte users niet verschillen van de loadtest, dan kunnen de KPI-waardes die boven vermeld staan als Datadog alerts ingesteld worden. Het is dus wel mogelijk om aan de hand van loadtesting scripts KPI-waardes vast te leggen, die in Datadog kunnen worden gebruikt voor alerts. Het volgende stuk van het onderzoek maakt gebruik van een K3s-cluster in de hoop dat de K3s-cluster meer bezoekers aankan onder dezelfde omstandigheden.

4.3 Post-Kubernetes

K3s is een *highly available*, gecertificeerde Kubernetes-distributie die is ontworpen voor productieworkloads op resource-beperkte locaties of binnen IoT-apparaten [14]. Voor het tweede gedeelte van het onderzoek werd K3s gebruikt om een basiscluster op te zetten, om op die manier de loadtestingresultaten voor en na K3s te kunnen vergelijken.

Het onderzoek stuurt het inkomende verkeer van de loadtest direct naar de *endpoint* van de K3s-cluster, zonder dat het nog door een *loadbalancer* moet. De ServiceLB die automatisch in een aangemaakte K3s-cluster aanwezig is, zorgt dat de *pods* verdeeld zijn, en om bij te *scalen* als het nodig zou zijn. Figuur 6 geeft de gebruikte architectuur weer van het post-K3s onderzoek.



Figuur 6: K3s Architectuur

4.3.1 *Autoscaling pods*

Om de cluster op weg te helpen met grotere workloads zijn HorizontalPodAutoScaler (HPA) *pods* een deel van de cluster. HPA-*pods* updaten automatisch de workload resource (zoals een *deployment*) met het doel om automatisch bij te schalen als de huidige *nodes* de workload moeilijk aan zouden kunnen. De HPA-technologie is een Kubernetes API-resource en werkt als een controller die binnen het Kubernetes control-*plane* ageert, die ook periodiek de schaal van de target aanpast [15].

Zoals eerder vermeld in de literatuurstudie is dit een geval van horizontaal schalen, waarbij nieuwe *pods* bij komen om te helpen met de workload. De K3s-cluster maakt gebruik van HPA-*pods* voor de 'front-end'-*pod* binnen de *namespace*, de user-*pod*, en voor de cart-*pod*.

```
student@k3s:~/SockShop/microservices-demo/deploy/kubernetes/autoscaling$ cat front-end-hsc.yaml
---
apiVersion: autoscaling/v1
kind: HorizontalPodAutoscaler
metadata:
  name: front-end
  namespace: sock-shop
spec:
  scaleTargetRef:
    apiVersion: apps/v1beta1
    kind: Deployment
    name: front-end

  minReplicas: 1
  maxReplicas: 4
  targetCPUUtilizationPercentage: 50
```

Figuur 7: Autoscaling van 'front-end' pod

Het aantal maxReplicas is op 4 ingesteld voor de HPA's, wat afgestemd is op de ruimtebeperkingen van de VM's binnen de vSphere omgeving. De trigger waardoor een HPA-*pod* aangemaakt wordt is een CPU verbruik van de target-*pod* wat boven de 50% stijgt. Voor het post-kubernetes gedeelte van dit onderzoek zijn de variabelen als volgt:

4.3.2 Overzicht variabelen

Overzicht variabelen		
Controle/Constante	***Afhankelijk***	***Onafhankelijk***
Locust script	*Throughput* (gem. hits/s)	Aantal gebruikers (totaal)
wait_time (1,10)	Responstijd (gem. ms)	ramp-up
hold-for: 15m	90% Responstijd (gem. ms)	
	Bandbreedte (gem. MiB/s)	

Tabel 3: Overzicht variabelen onderzoek post-K3s

Enkele beïnvloedende factoren zullen ook verandert moeten worden gedurende het onderzoek, omdat de resultaten anders een te grote foutmarge zullen oplopen. Tijdens het pre-K3s onderzoek werd er al na gekeken of een langere ramp-up time andere resultaten zou geven, maar dat was niet het geval. In het post-K3s onderzoek is het aantal gelijktijdige gebruikers hoger, waarbij een te korte ramp-up time wel een te grote shock voor het systeem zou zijn. Daarom is ervoor gekozen om de ramp-up time op de volgende manier te justeren afhankelijk van het aantal gebruikers:

- Tussen 1500 - 2750 gebruikers: ramp-up van 10 minuten
- Tussen 3000 - 4000 gebruikers: ramp-up van 20 minuten
- Tussen 4250 - 5000 gebruikers: ramp-up van 30 minuten

Deze veranderende factoren hebben een kleine, maar merkbare invloed op de resultaten, omdat de uitvoeringstijd van de loadtesten langer zal duren. Daardoor zijn er verschillen in de gemiddelde waardes die bij *throughput*, responstijd, en bandbreedte terugkomen, maar het is een aanpassing die in dit geval noodzakelijk is om een grotere foutmarge te vermijden.

4.3.3 Resultaten post-K3s

Gebruikers (totaal)	*Throughput* (gemiddeld, hits/s)	Responstijd (gemiddeld, ms)	90% Responstijd (gemiddeld, ms)	Bandbreedte (gemiddeld MiB/s)
1500	204.82	7	11	1.75
1750	238.76	8	13	1.94
2000	273.3	10	14	1.94
2250	307	10	13	2.49
2500	340.89	15	19	3.15
2750	375.58	13	16	3.64
3000	388.52	23	41	3.07
3250	418.66	44	126	3.49
3500	447.39	78	230	4.17
3750	468.54	197	445	3.41
4000	468.92	560	1002	4.83
4250	453.15	710	1020	4
4500	456.88	983	1100	4.15
4750	465.67	1230	1130	4.07
5000	462.28	1560	1450	4.31

Tabel 4: Resultaten loadtesting post-K3s

4.3.4 Grafische weergave resultaten

Figuur 8: Grafische weergave resultaten post-K3s

Net zoals bij de loadtestingresultaten voor de K3s-implementatie is bij deze resultaten ook een specifiek punt waarbij de gemiddelde responstijd exponentieel begint te stijgen. Dit is rond de 3000-3250 gebruikers, die gemiddeld ongeveer 400 *requests* per seconde maken. Dit wordt verder ondersteund door de waardes van de *throughput*, die na 3750 gebruikers een plateau heeft bereikt, en zelfs begint te dalen wanneer het aantal gebruikers te hoog is om een acceptabele responstijd te garanderen (< 1000ms). Interessant is wel te vermelden dat de gemiddelde responstijd < 3000 gebruikers bijna 5x zo snel is als bij de pre-K3s setup.

Het effect van de veranderede ramp-up time is duidelijk te zien tussen 4000 en 5000 gebruikers, waar de graad van de stijging minder hoog is in vergelijking met < 4000 gebruikers, maar de trendlijn van de gemiddelde responstijd blijft wel redelijk constant.

HPA's zijn bij deze resultaten ook een deel geweest van het cluster, dus de cluster heeft zelf zo veel mogelijk bijgeschaald om het aantal gebruikers tegemoet te komen. Afgezien van het grote aantal beperkingen (die in de Evaluatie grondig zullen worden besproken), is het mogelijk om bepaalde KPI-waarden vast te leggen vanuit dit onderzoek.

4.3.5 KPIs post-K3s

De KPI-waarden die vanuit dit onderzoek vastgelegd zijn berusten op deze specifieke loadtest en de uitvoering daarvan. Met andere soorten loadtests zullen de resultaten verschillen, maar dat valt buiten de scope van het onderzoek. De volgende KPI's zijn vastgelegd op bepaalde punten waar *scaling* nodig zou zijn om responstijden hoger dan 0.1 seconde te vermijden.

- Aantal gebruikers: ~ 3100
- Gemiddelde *throughput:* ~ 400 hits/seconde

In het volgende stuk zal het onderzoek geëvalueerd worden, waarbij de vermelde beperkingen ook besproken zullen worden.

4.3.6 Conclusie post-K3s

Afgezien van het grote aantal beperkingen is het mogelijk om aan te duiden wanneer de Sock Shop moet beginnen bij te schalen. Alerts kunnen in Datadog ingesteld worden, bijvoorbeeld in Datadog RUM op het aantal gelijktijdige usersessies, wanneer meer dan 3100 gelijktijdige gebruikers actief zijn in de Sock Shop. Hetzelfde geldt voor het aantal *requests* per seconde. Wanneer die boven de 400 hits/seconde stijgen, kan de Sock Shop nieuwe pods bijmaken om opnieuw horizontaal bij te schalen.

Deze resultaten zijn afhankelijk van deze soort van loadtest, met een lineaire stijging in gebruikers, waarbij het aantal HPA's beperkt is tot de infrastructuur van de cluster. In een andere omgeving, met een ander soort loadtest, zouden de resultaten verschillen. Deze resultaten en KPI's zijn dus alleen als absolute waardes aan te nemen als de huidige infrastructuur onveranderd blijft, en als de gebruikers zich soortgelijk gedragen in realtime. Op de volgende pagina worden de beperkingen in detail behandeld.

5 Evaluatie

5.1 Beperkingen

Zoals eerder besproken waren er verschillende beperkingen verbonden aan dit onderzoek. Het aantal resources ter beschikking waren niet representatief van een standard webshop ten eerste. Aansluitend zijn er ook andere tools die een betere toepassing hebben in deze situatie.

Beperkingen op het vlak van omgeving en resources komen neer op de vSphere en de vooropgestelde *source* code. De vSphere had een beperkte opslag van 300 GB en een maximum van 30 GB RAM-geheugen. Dit zorgde voor een gebrek aan disk ruimte, wat uiteindelijk een effect kan hebben op de infrastructuur met of zonder K3s-cluster. Een cluster waarbij de *nodes* bijna 100% diskgebruik hebben bereikt kan minder makkelijk data wegschrijven, wat een effect heeft op de performantie van de K3s-cluster.

Op vlak van de gebruikte software en tools zijn er beperkingen bij het testen en monitoren van de cluster of de servers in de vSphere. WeaveWorks heeft zelf Locust aangeraden als testplatform [16]. Maar Locust testen samen met Datadog RUM is niet ideaal omdat Datadog RUM alle gebruikers als een user-sessie groepeert, omdat die van dezelfde 'locatie' komen. Dit is niet ideaal als het aantal sessies van een test belangrijk is. Daarnaast is het moeilijk om met Locust de gewilde en realistische testsituaties te simuleren, omdat verschillende testvormen moeilijk in te stellen zijn. K6s zou een betere optie zijn voor toekomstige onderzoeken, omdat Datadog een K6s integratie heeft, waarbij ramp-up met plateaus kunnen worden ingesteld, net zoals herhalingen, en andere *custom* testvormen. Verder is het ook zeer belangrijk nogmaals te vermelden dat de aard van loadtest met de ingestelde ramp-up de resultaten volledig beïnvloedt, met als gevolg dat andere soorten loadtests waarschijnlijk andere resultaten zouden genereren. Bij de uitvoering is de ramp-up constant en lineair, wat geen realistisch gebruikersgedrag zou zijn normaal, en het is dus aan te nemen dat de resultaten anders zouden zijn met andere soorten ramp-ups.

De architectuur van het K3s cluster heeft ook een effect op de resultaten. In een grotere K3s cluster met meer HPA-*pods* waren de resultaten anders geweest in vergelijking met de gebruikte infrastructuur. Clusters zijn in realistische gevallen eerder groter en meer flexibel. In dit onderzoek is deze redelijk beperkt en heeft deze een beperkt aantal workers en maar een *control-plane.* Dit is een beperking met betrekking tot hoe realistisch het onderzoek is.

Het is dus belangrijk om de resultaten van deze uitvoering als een momentopname in een zeer specifieke omgeving te beschouwen.

Andere invloedrijke factoren die in het onderzoek een rol speelden zijn factoren zoals netwerkperformance en bandbreedte van het PXL-netwerk, serverperformance van de Ubuntu Servers, DB-performance van de MongoDB en MySQL DB's, *caching* instellingen van de Traefik edge-router service, en user gedrag natuurlijk.

Afgezien van het grote aantal beperkingen is het mogelijk om de hypothese te bespreken.

5.2 Bespreking van de Hypothese

Rekening houdend met deze beperkingen, kan er geconcludeerd worden dat de hypothese gedeeltelijk juist was en bevestigd werd door de verkregen resultaten. De hypothese stelde dat het correct instellen van alerts de WeaveWorks Sock Shop voor zou kunnen bereiden op een *scalability* scenario.

Het uitgevoerde onderzoek toonde aan dat door gebruik te maken van Datadog-integraties en monitoringtypes, belangrijke KPI's van de Sock Shop verzameld konden worden en analyse mogelijk was. Dit bood de mogelijkheid om vroegtijdig te reageren. Ook al was Datadog RUM geen succesvolle optie voor het instellen van alerts, zouden de algemene dashboards van Datadog wel gebruikt kunnen worden om alerts in te stellen.

Bovendien werd aangetoond dat het gebruik van loadtestingscripts effectief was om de microservices tot hun limiet te belasten en zo de juiste KPI's in te stellen. Door herhaald loadtestingscripts uit te voeren, konden gemiddelde KPI-waarden worden bepaald, die gebruikt werden om de Sock Shop voor te bereiden op *scalability*. Deze loadtestingscripts waren gebaseerd op realistische gebruikersscenario's die specifieke functies van de microservices targetten, en ze zorgden voor geïsoleerde testomgevingen om de microservices onder druk te zetten.

Uiteindelijk maakte de evaluatie van de verkregen resultaten het mogelijk om precies te bepalen wanneer het tijd is om (in deze context) de Sock Shop te schalen voor hogere bezoekersaantallen.

6 Conclusie

Uit het onderzoek blijkt dat het gebruik van loadtestingscripts in samenwerking met de Datadog integraties en monitoringtypes, de Sock Shop kan voorbereiden op een *scalability* scenario. Door middel van Locust loadtestingscripts is het mogelijk om de limieten van de kritische infrastructuur aan te duiden. Aan de hand van de resultaten is het dus mogelijk om bepaalde KPI-waardes in te stellen als Datadog alerts, om zo de Sock Shop voor te bereiden op *scalability.*

De leidende onderzoeksvraag was: “Welke *metrics* kunnen via Datadog worden verwerkt om een website/Sock Shop voor te bereiden op vlak van *scalability*?”.

Vanuit de resultaten van het onderzoek bleek dat het aantal gebruikers en de gemiddelde *throughput*-waarde als de meest relevante *metrics* vooraan staan. Door gebruik te maken van verschillende monitoringtools is het in principe mogelijk om de Sock Shop voor te bereiden om te schalen, als deze KPI-waarden ingesteld zijn als alerts.

Het gebruik van Kubernetes zorgt ervoor dat er een snellere en betere werking is van de microservices waaruit de Sock Shop is opgebouwd. Hierdoor is deze horizontaal schaalbaar. Ook uit de gemiddelde waardes blijkt dat kubernetes de Sock Shop veel sneller en efficiënter laat werken.

Wel is het nogmaals belangrijk om te vermelden dat het aantal beperkingen bij het onderzoek vrij hoog is. Niet alleen de factoren die oncontroleerbaar waren, zoals de netwerkperformance of de performantie van de Ubuntu servers speelden een rol. De hele K3s-infrastructuur, de aard van loadtesten, het gedrag van de gebruikers binnen de loadtesten, dat zijn allemaal factoren die tot andere resultaten zouden voeren in een ander onderzoek. Daardoor is het instellen van alerts met KPI-waardes uit het huidige onderzoek wel relevant voor een Sock Shop omgeving zoals diegene van dit onderzoek, maar minder relevant voor een algemene webshop met een andere infrastructuur en echte gebruikers.

Bibliografie

[1] „Load test," Sock Shop, 2017. [Online]. Available: https://microservices-demo.github.io/docs/load-test.html. [Geopend 16 April 2023].

[2] „Microservices And Containers," AVI Networks, [Online]. Available: https://avinetworks.com/what-are-microservices-and-containers/. [Geopend 16 April 2023].

[3] „Microservices Scalability as a Business Issue," Optimus Information, 29 June 2022. [Online]. Available: https://www.optimusinfo.com/microservices-scalability-as-a-business-issue/. [Geopend 15 April 2023].

[4] C. BasuMallick, „Stateful vs. Stateless: Understanding the Key Differences," spiceworks, 20 September 2022. [Online]. Available: https://www.spiceworks.com/tech/cloud/articles/stateful-vs-stateless/. [Geopend 16 April 2023].

[5] „Introduction to Integrations," Datadog, [Online]. Available: https://docs.datadoghq.com/getting_started/integrations/. [Geopend 14 April 2023].

[6] Sagar, „Getting Started with the Datadog Agent on Ubuntu Linux," Talearning, 31 March 2022. [Online]. Available: https://adamtheautomator.com/datadog-agent/. [Geopend 14 April 2023].

[7] „Getting Started with the Agent," Datadog, [Online]. Available: https://docs.datadoghq.com/getting_started/agent/. [Geopend 14 April 2023].

[8] „Datadog Agent," Cribl, [Online]. Available: https://docs.cribl.io/shared/sources-datadog-agent/#:~:text=Datadog%20Agent%20is%20open%2Dsource,or%20containers%20on%20the%20host. [Geopend 14 April 2023].

[9] „Dashboards," Datadog, [Online]. Available: https://docs.datadoghq.com/dashboards/. [Geopend 16 April 2023].

[10] „Metri Types," Datadog, [Online]. Available: https://docs.datadoghq.com/metrics/types/?tab=count. [Geopend 16 April 2023].

[11] „Monitor Types," Datadog, [Online]. Available: https://docs.datadoghq.com/monitors/types/. [Geopend 15 April 2023].

[12] „Alerting," Datadog, [Online]. Available: https://docs.datadoghq.com/monitors/. [Geopend 15 April 2023].

[13] „RUM & Session Replay," Datadog, [Online]. Available: https://docs.datadoghq.com/real_user_monitoring/. [Geopend 17 May 2023].

[14] „Why Use K3s," K3S, [Online]. Available: https://k3s.io/. [Geopend 27 May 2023].

[15] „Horizontal Pod Autoscaling," Kubernetes, [Online]. Available: https://kubernetes.io/docs/tasks/run-application/horizontal-pod-autoscale/. [Geopend 27 May 2023].

[16] „Load tests Weaveworks," Weaveworks , [Online]. Available: https://microservices-demo.github.io/docs/load-test.html.

Bijlagen

A. Locust Loadtestingscripts

```
from locust import HttpUser, between, task, run_single_user
import random


class WebsiteUser(HttpUser):
    wait_time = between(1, 10)
    host = "http://10.128.102.2"

    random_priority = [1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20]

    @task(20)
    def index(self):
        with self.client.get(
            "/index.html",
            catch_response=True) as resp:
            pass

    @task(random.choice(random_priority))
    def basket(self):
        with self.client.get(
            "/basket.html",
            catch_response=True) as resp:
            pass

    @task(random.choice(random_priority))
    def basket(self):
        with self.client.get(
            "/card",
            catch_response=True) as resp:
            pass

    @task(random.choice(random_priority))
    def basket(self):
        with self.client.get(
            "/orders",
            catch_response=True) as resp:
            pass

    @task(random.choice(random_priority))
    def basket(self):
        with self.client.get(
            "/address",
            catch_response=True) as resp:
            pass
```

```
    @task(random.choice(random_priority))
    def customer(self):
        with self.client.get(
            "/customers/57a98d98e4b00679b4a830b2",
            catch_response=True) as resp:
            pass









    @task(15)
    def product(self):
        catalogue = ["03fef6ac-1896-4ce8-bd69-b798f85c6e0b", "3395a43e-2d88-
40de-b95f-e00e1502085b", "510a0d7e-8e83-4193-b483-e27e09ddc34d", "808a2de1-
1aaa-4c25-a9b9-6612e8f29a38", "819e1fbf-8b7e-4f6d-811f-693534916a8b",
"837ab141-399e-4c1f-9abc-bace40296bac", "a0a4f044-b040-410d-8ead-
4de0446aec7e", "d3588630-ad8e-49df-bbd7-3167f7efb246", "zzz4f044-b040-410d-
8ead-4de0446aec7e"]
        random_product = random.choice(catalogue)
        product_url = "/detail.html?id=" +  random_product
        with self.client.get(
            product_url,
            catch_response=True) as resp:
            pass


    @task(random.choice(random_priority))
    def category(self):
        with self.client.get(
            "/category.html",
            catch_response=True) as resp:
            pass


    task(random.choice(random_priority))
    def customer_orders(self):
        with self.client.get(
            "/customer-orders.html",
            catch_response=True) as resp:
            pass


    @task(random.choice(random_priority))
    def category_pages(self):
        random_page = random.randint(1,3)
```

```
        with self.client.get(
            "/category.html?page=" + str(random_page) + "&size=" +
str(random_page * 2),
            catch_response=True) as resp:
            pass


    @task(random.choice(random_priority))
    def category_types(self):
        random_colour = ["sport", "black", "geek", "magic", "skin", "blue",
"red", "brown", "green", "formal", "toes", "smelly", "large", "small"]
        tags = set()
        while len(tags) < 3:
            tags.add(random.choice(random_colour))
        tags_str = "%2C".join(tags)
        with self.client.get("/category.html?tags=" + tags_str,
catch_response=True) as resp:
            pass



if __name__ == "__main__":
    run_single_user(WebsiteUser)
```

B. Taurus Configuratie

```
---
reporting:
- module: blazemeter

modules:
  blazemeter:
    token: <API token>
    upload-artifacts: true
    public-report: true
    browser-open: none
    send-interval: 5s

execution:
- executor: locust
  concurrency: 1500
  hold-for: 10m
  ramp-up: 1m
  scenario: locust

scenarios:
  locust:
    script: "../loadtesting/run_single_user.py"
    interpreter: python3
```

C. Verkorte BlazeMeter Report

BlazeMeter

LOAD TEST REPORT

Jenkins Build 263

Report created by Rasmus Leseberg.
Date of Run: Thu, 05/25/2023 - 20:59
Duration: 36 minutes

DESCRIPTIVE SUMMARY / CONCLUSIONS

Filtered By:

Scenarios: **All**

Locations: **All**

This report is displayed in milliseconds (ms)

Average Throughput	Avg. Response Time	90% Response Time	Error Rate
468.9	**560**	**1002**	**0.17**
Hits/s	Milliseconds	Milliseconds	%

TOP 5 SLOWEST RESPONSES (BY AVG. RESPONSE TIME)

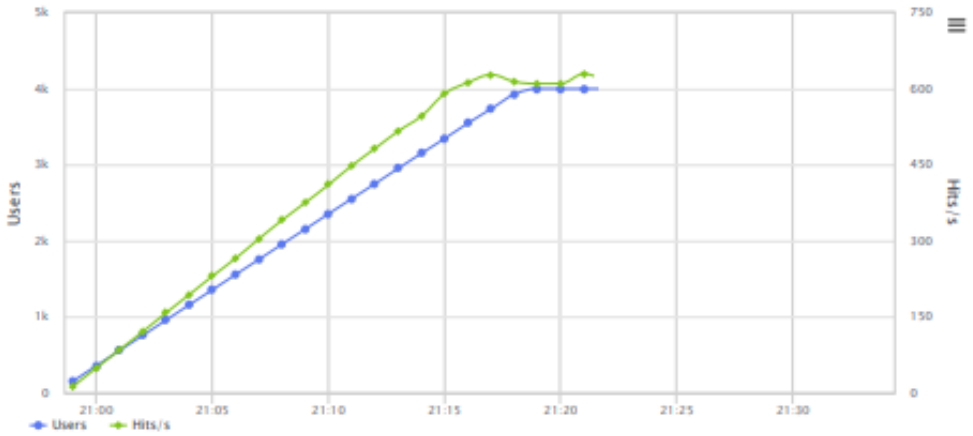
Request	# Samples	Avg Time	90% Time	Max Time
/category.html?tags=...	107	879 ms	1801 ms	7300 ms
/category.html?tags=...	957	686 ms	1301 ms	8092 ms
/category.html?tags=...	632	683 ms	1095 ms	8568 ms
/category.html?tags=...	616	637 ms	1075 ms	8936 ms
/category.html?tags=...	1216	634 ms	1184 ms	7388 ms

TOP 5 ERRORS

Error Code	Error Message	% of executions	Count
500	Remote end closed connection without response	0.15%	1468
500	[Errno 104] Connection reset by peer	0.02%	203
500	[Errno 111] Connection refused	0.00%	4
500	Internal Server Error for url: /address	0.00%	2
500	Internal Server Error for url: /customers /57a98d98e4b00679b4a830b2	0.00%	1

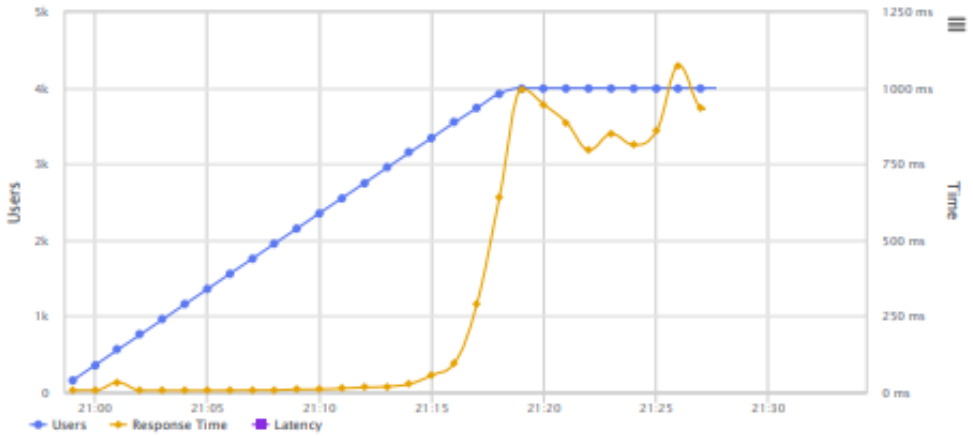
TIMELINE GRAPHS

Concurrent Users and Hits/s



TIMELINE GRAPHS

Concurrent Users and Response Time



7 Reflectieverslagen

Door: Rasmus Leseberg

Tijdens het tweede semester van 2TIN was de Sock Shop de kernfocus van het Research Project. Het project was onderverdeeld in vier sprints, waarbij de 'Research Paper' een onderzoekende rol speelde als aanvulling tot het project.

Het hele Research Project is uiterst leerrijk geweest, ik heb waardevolle technische vaardigheden leren kennen die in de toekomst waarschijnlijk goed van pas zullen komen. De inhoud van de sprints was op een manier verdeeld waardoor inhoud van andere vakken zoals Automation, Systems Advanced II, of Big Data (MongoDB) goed van pas kwam. De Research Paper zelf is ook een waardevolle oefeningen geweest i.v.m. de bachelorproef die volgend jaar te wachten staat. Ik ben dit soort schrijfopdrachten gewend in het Engels, maar dankzij de taallector heb ik ook haar waardevolle feedback bij deze Research Paper kunnen implementeren.

De teamdynamiek was constructief en productief voor het grootste gedeelte, wat een positieve verrassing was ten opzichte van eerdere groepsprojecten. Teamgenoten hebben elkaar uit de knoop kunnen helpen waar nodig, en samen hebben we op drie van de vier sprintgoals de sprintgoal behaald. Een voorzet wat ik wel mee zal nemen voor toekomstige groepsopdrachten is het beter communiceren van gebrek aan teamwerk. Ik ga meestal liever de confrontatie niet aan, en neem aan dat de onbalans aan gepresteerde uren wel vanzelf spreekt als motivatie voor verandering, maar ik ga dat beter verbaliseren in toekomstige groepsopdrachten. Als ik niet expliciet vermeld dat de last te groot is, dan zouden mijn teamgenoten het soms ook niet kunnen weten.

Verder nam het Research Project persoonlijk ook te veel tijd in beslag waardoor andere vakken hebben geleden, dus ik zal mijn tijdmanagement proberen te verbeteren bij het volgende project.

Door: Tomas Soors

Wij als tweedejaars studenten in de opleiding toegepaste informatica, kregen de kans om deel te nemen aan het researchproject. Hierbij was de opdracht om de website “Sock Shop” te *deployen* door middel van verschillende tools en hier dan *high availability* op toe te passen. Daarnaast was er een *paper* die is onderbouwd door een literatuurstudie en een technisch onderzoek.

De taken in dit project zijn uiteenlopend, iedereen in elke groep heeft wel aan alles kunnen werken. Iedereen kreeg *hands-on* experience met de verschillende onderdelen en kon onderzoeken naar verschillende oplossingen voor de problemen die het project met zich meebracht.

Dit werd onderbouwd door het Scrumprincipe met verschillende sprints, 4 om precies te zijn. Bij elke sprint werden er taken vastgelegd die in die tijdspannen idealiter afgewerkt moeten zijn. Bij elke sprint vonden er verschillende *meetings* plaats die ieder groepslid een beter beeld gaf van wat de rest voor zijn rekening nam. Om ten slotte een *retrospective* te doen bij elke sprint om verbeterpunten te achterhalen op groepsvlak.

Op vlak van de paper werd er een literatuurstudie gedaan op basis van het technisch onderzoek dat werd ondernomen tijdens deze sprints. Bij de 3de sprint werd er gekeken naar de performance van de Sock Shop voor de k3s *cluster* geïmplementeerd was. Om tijdens de 4de en laatste sprint te onderzoeken wat de performance is van de Sock Shop na de implementatie van de k3s *cluster*. Door middel van de vergelijking tussen de resultaten van de 2 situaties werd er vergeleken met de vondsten van de literatuurstudie en werd er een conclusie getrokken. Zoals eerder besproken waren er ook beperkingen betrokken bij deze paper, maar de conclusie was getrokken rekening houdende met deze beperkingen.

Vergeleken met de gestelde hypothese waren deze resultaten in lijn met de verwachtingen die waren gesteld. We kunnen de Sock Shop schalen naar de meest efficiënte grote. De meldingen die binnenkomen op onze monitoring tool Datadog zijn relevant voor deze schaling en maken het gemakkelijker om te bepalen wanneer deze zou moeten plaatsvinden.

Voor mij waren deze laatste weken leerrijk. Op praktisch vlak en werken in een team. De teams waren zo opgesteld dat er verschillende persoonlijkheden in elk team aanwezig waren en dat heb ik wel gemerkt. Wat soms wel frustrerend was, maar daarnaast ook een leerzame ervaring. Het samenwerken was fijn en vlot. De praktische leerstof gaf mij een beter beeld over de toepassingen in het werkveld, dan heb ik het over de verschillende tools die aan bod kwamen of de beperkingen die sommige toepassingen met zich meebrengen. De communicatie tussen de verschillende teamgenoten liep soms stroever, maar kwam uiteindelijk wel meer tot bloei. Daarnaast kreeg iedereen van het team de kans om zijn deel te representeren en te verdedigen.

Deze ervaring heeft mij doen realiseren dat in teams samenwerken niet altijd gemakkelijk is. Er zijn veel verschillende karakters waarmee iemand te maken kan krijgen en het is belangrijk om te praten over dingen die iemand dwarszitten en hier proberen een oplossing voor te zoeken. De technologieën die aan bod zijn gekomen zullen ook zeker terug voorkomen in de toekomst, en dit geeft mij een voorsprong die ik zeker ga benutten als deze terug aan bod komen.

Door: Mateo Lemmens

In het tweede jaar toegepaste informatica hebben we voor het researchproject een volledige *deployment* van de Sock Shop opgezet, een uitwerking gedaan wat betreft *high availability.* Hierbij hebben we een paper geschreven die de verschillen uitlegt tussen een pre en post kubernetes uitwerking.

Het moeilijkste obstakel was de eerste sprint zowel technische als project gericht, technisch was het een hele uitdaging om de Sock Shop via Docker zonder *compose* op diverse machines te *deployen*. Ook op vlak van agile scrum werken was een zeer leerrijk proces. Op het begin was de werking met behulp van Jira niet super. Na de eerste 3 weken hebben we een reflectie gedaan en hebben we met de feedback van de leerkrachten onze werkmethodes aangepast. Zeker het gebruik van taken in Jira en deze toekennen aan iemand heeft geholpen om de volgende sprints vlotter te laten verlopen. Reviews uitvoeren volgens de *Definition of Done* was iets dat doorheen het hele researchproject is verbeterd. Ook omdat we na elke sprint een *retrospective* gedaan hebben, hieruit hebben we onze werkpunten opgesomd die we als leerproces doorheen volgende sprints zijn gaan verbeteren.

Iedereen heef zelf een stukje bijgedragen aan het project. Elke sprint werd er gekeken wie welke taken wou uitwerken. Iedereen heeft zelf zijn oplossing mogen uitwerken. Zo zijn we erachter gekomen dat er nooit 1 juiste uitvoering is en daardoor met diverse technologieën hebben kunnen experimenteren. Het meest interessante aan dit project was dat je zowel de leerstof van vorig jaar als dit jaar kon verwerken in de taken die je uitvoerde. Bijvoorbeeld het automatiseren van het *deployen* van de Sock Shop via Ansible. Hierdoor ging zowel het project vooruit alsook had je een herhaling van de leerstof van bepaalde vakken. Daarnaast leerde je ook met nieuwe technologieën zoals Datadog en kubernetes werken.

De research paper hebben we gebaseerd op de laatste 3 à 2 sprints, waarbij we een onderzoek hebben gedaan wat de verschillen zijn op bijvoorbeeld gebiedt van snelheid en responstijd van de Sock Shop wanneer we deze manueel opzetten of via kubernetes. Door middel van een monitoringsysteem van Datadog hebben we er veel hulp aan gehad, omdat we hierdoor de beste KPI-waardes hebben kunnen bepalen. Om de Sock Shop zo goed mogelijk te kunnen testen hebben we via locust real world testingscenario's opgebouwd die we meerder keren hebben uitgevoerd om zo tot de belangrijkste KPI-waardes te komen.

Wat voor mijzelf het meest leerrijke was, is het leren werken in sprints. Uitwerking en onderverdeling van taken in Jira, kijken wat de beste methodes zijn. Een *retrospective* uitvoeren en hieruit leren wat we in de volgende sprints kunnen verbeteren. Hieruit is de belangrijkste les dat je als team samenwerkt en reviews doet voor een taak om zo tot een beste oplossing tot komen. Ook het leren werken met nieuwe technologieën was zeker een pluspunt zoals Datadog, Ansible en zeker tot stand brengen van real world loadtesting scrips met behulp van locust.

Door: Victor Beckx

Als tweedejaarsstudenten van de hogeschool PXL in de afstudeerrichting toegepaste informatica met als optie systeem en netwerken maken de leerling kennis met het vak Research Project. Dit vak heeft als doel samenwerking en processen aan te leren aan hun studenten door middel van een project en paper uit te werken in groepsvorm. Het onderwerp van het Research Project dit jaar was “The Sock Shop”. Dit is een webshop die wij de leerlingen moesten *deployen* met alle microservices op verschillende server, geautomatiseerd, monitoring en high availability.

Tijdens dit project werd er de scrum methode toegepast. Voor mij was dit niets nieuwes aangezien dit bij eerdere projecten heb toegepast. Het toepassen van scrum binnen onze groep was in het begin wel moeilijk. De eerste sprint die uitgevoerd werd was zeer slecht gepland en dit gaf als resultaat dat de eerste sprint een kleine flop was. Deze fouten hebben we verbeterd naar sprint 2 toe. Deze sprint werd er gefocust op het opsplitsen van de microservices en dan uiteindelijk de Sock Shop online te krijgen met verschillende server waar deze microservices op draaiden. De eerste sprint vond ik persoonlijk één van de leuker omdat je dan je groep leert kennen en er een groepsdynamiek ontstaat.

Bij de tweede sprint lag de focus op het automatiseren van wat wij bij de eerste sprint hadden gedaan. Dit om het ons makkelijker te maken naar de toekomst toe en het sneller kunnen op zetten van de Sock Shop. Dit hadden we dan gedaan via Ansible. Dit was voor mij de eerste keer werken met Ansible zelf. Dit was een zeer leerrijke en oog openende ervaring. Dit omdat ik zelf nog niet wist wat Ansible was. Het werken met Ansible ging zeer vlot door de goeie documentatie van Ansible zelf. Wat ik zeer leuk vond was dat Ansible via ssh werkt en je geen agents of dergelijke moest installeren op de server. Het leuke aan Ansible is dat ja alle server kunt aan spreken met 1 playbook. Deze sprint verliep zeer goed omdat we een goede planning hadden voorzien en iedereen mekaar hielp als men vastzat.

In de derde sprint hadden wij ons gefocust op loadtesten en monitoren. Voor het monitor gedeelte moesten we Datadog gebruiken dit is een monitoring tool die wij ook moesten onderzoeken voor onze paper. Deze sprint had ik mij toe gefocust op het monitoring gedeelte. Dit was mijn eerste ervaring met monitoring zelf. Ik was verbaasd hoe uitgebreid Datadog is, voor elke service of installatie had Datadog wel een integratie. Deze waren ook zeer makkelijk te integreren wat DataDog zeer gebruiksvriendelijk maakt. Het paper gedeelte over Datadog was veel opzoeken omdat er niet veel over te vinden is buiten de artikels van Datadog zelf. Dit maakte het moeilijk om te kunnen fact-checken met meerdere bronnen. Deze sprint verliep hetzelfde als sprint twee, een goede planning en goed teamwerk.

De vierde en laatste sprint werd er aan high availability gedaan. Voor was dit een mindere sprint omdat ik hier minder snel mee weg was dan de andere van mijn team. Daarom dat ik bij deze sprint meer ondersteunende gewerkt heb dan zeer diep te gaan in bepaalde zaken. Dit vond ik daarom wel jammer omdat je het team een beetje in de steek laat en je zelf niet veel bijleert voor later.

Uiteindelijk was ik heel blij met het eindresultaat. De groepssfeer en samenwerking was zeer leuk, iedereen heeft zijn steentje bijgedragen en hebben we zo het project tot een goed en mooi einde kunnen brengen.