HO
GENT

Faculteit Bedrijf en Organisatie

Monitoring-tools voor Docker en Kubernetes

Olivier Troch

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

Promotor:
Bert Van Vreckem

Instelling: HoGent

Academiejaar: 2020-2021

Tweede examenperiode

Faculteit Bedrijf en Organisatie

Monitoring-tools voor Docker en Kubernetes

Olivier Troch

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

Promotor:
Bert Van Vreckem

Instelling: HoGent

Academiejaar: 2020-2021

Tweede examenperiode

Woord vooraf

Samenvatting

Inhoudsopgave

Lijst van figuren

Lijst van tabellen

1. Inleiding

1.1 Probleemstelling

Studenten binnen de opleiding Toegepaste Informatica in HoGent zijn niet bekend met technologieën en tools die gebruikt worden binnen het Kubernetes en Docker domein. Het monitoren van deze technologieën is daar een onderdeel van. Ten eerste zijn Kubernetes en Docker geen eenvoudige technologie om te implementeren en daarvoor zal er beroep gedaan worden op de bachelorproef van Detemmerman Cedric. Ten tweede is er een groot aanbod van mogelijke tools die hiervoor gebruikt kan worden en weten we niet welke voor deze doelgroep geschikt is.

1.2 Onderzoeksvraag

Dit onderzoek zal nagaan welke tool

Zoals eerder vermeld is er een groot aanbod aan monitoringtools voor een Kubernetes/-Docker environment. Om die reden zal dit onderzoek nagaan welke tools de studenten geschikt zijn. Zo bekomen we volgende hoofdonderzoeksvraag:

- Welke monitoringtool voor een Kubernetes/Docker environment is het meest geschikt voor de studenten Toegepaste Informatica om zo een goede basiskennis te bekomen.

Eveneens zijn er nog een aantal deelvragen die ook belangrijk zijn in dit onderzoek:

- Wat zijn de belangrijkste verschillen tussen de tools?
- Waar moeten de tools inzicht op geven?
- Welke alerts zijn belangrijk?
- Welk proof-of-concept is voldoende om de tool aan te leren?

1.3 Onderzoeksdoelstelling

Dit onderzoek heeft tot doel om een geschikte monitoringtool te vinden die voor de studenten Toegepaste Informatica een meerwaarde kan betekenen. Door dit onderzoek zal zowel een vergelijking van tools beschikbaar zijn, alsook een reproduceerbare proof-of-concept die studenten aan de hand van een handleiding kunnen bekomen. Aan de hand van dit proof-of-concept kunnen de educatieve doeleinden bereikt worden.

1.4 Opzet van deze bachelorproef

De rest van deze bachelorproef is als volgt opgebouwd:

In Hoofdstuk 2 wordt een overzicht gegeven van de stand van zaken binnen het onderzoeksdomein, op basis van een literatuurstudie.

In Hoofdstuk 3 wordt de methodologie toegelicht en worden de gebruikte onderzoekstechnieken besproken om een antwoord te kunnen formuleren op de onderzoeksvragen.

In Hoofdstuk 4, tenslotte, wordt de conclusie gegeven en een antwoord geformuleerd op de onderzoeksvragen. Daarbij wordt ook een aanzet gegeven voor toekomstig onderzoek binnen dit domein.

2. Stand van zaken

2.1 Virtualisatie

2.1.1 De geschiedenis van virtualisatie

Abstract beschreven is virtualisatie een term die verwijst naar de abstractie van computerbronnen. Het doel van een virtuele computer omgeving is het gebruik van hulpbronnen/resources te verbeteren door middel van een geïntegreerd besturingsplatform voor gebruikers en applicaties gebaseerd op aggregatie van heterogene en autonome middelen. Tegenwoordig is virtualisatie op meerdere niveaus belangrijk om systeembeveiliging, betrouwbaarheid en beschikbaarheid te verbeteren en garanderen alsook het verminderen van kosten en zorgen voor meer flexibiliteit. Virtualisatie is snel uitgegroeid tot een go-to-technologie voor het verhogen van de efficiëntie. Virtualisatie biedt een enorme flexibiliteit, zo kunnen ongelijksoortige architecturen (Linux, MacOS, Windows,...) op één enkele machine tegelijkertijd worden ingezet zonder interferentie (Radhwan, 2013).

Virtualisatie is een techniek waarin alle fysieke kenmerken van computerbronnen verborgen worden waarvan applicaties, andere systemen en toepassingen gebruik kunnen maken. Het is een softwareabstractielaag tussen de hardware en het besturingssysteem en de toepassingen die erbovenop draaien. Dit wordt een hypervisor of virtual machine monitor(VMM) genoemd. Doordat de hardwarebronnen door de VMM/hypervisor worden gebruikt en niet door de host, is het mogelijk meerdere besturingssystemen te draaien op dezelfde hardware. Al deze verschillende besturingssystemen worden dan virtuele machines (VM's) genoemd. Virtualiteit verschilt alleen van werkelijkheid in de formele wereld, terwijl zij eigenlijk een soortgelijke essentie of werking bezit. In de computerwereld wordt een virtuele omgeving waargenomen als die van een echte omgeving door

zowel toepassingsprogramma's als de rest van de wereld, hoewel de onderliggende mechanismen formeel wel verschillend zijn (Radhwan, 2013).

Virtuele machines zijn al meer dan 50 jaar in de computergemeenschap, ontwikkeld in 1960 door IBM Corporation, oorspronkelijk was dit om een grote mainframe computer te verdelen in verschillende instanties. Deze functie werd uitgevonden omdat het onderhoud van de grotere mainframe systemen omslachtig werd. Door het systeem te partitioneren, maakt het het mogelijk meerdere processen en toepassingen gelijktijdig te kunnen draaien, waardoor de efficiëntie en onderhoudskosten van de omgevingen verminderen. Onder toezicht en leiding van Professor Fernando J. Corbató beter bekend als 'Corby', zijn de ingenieurs en programmeurs begonnen met CTSS, de Compatible Time-Sharing System (Radhwan, 2013).

De CTSS-supervisor leverde een aantal virtuele machines, allemaal een IBM 7094. Een van deze virtuele machines was de achtergrondmachine en had toegang tot tapedrives(Een tapedrive is ene opslag apparaat). De andere virtuele machines waren voorgrondmachines. Deze machines konden normale 7094-machinetaal draaien met een snelheid van 7094 en konden ook een extra instructie uitvoeren, die een groot aantal 'supervisorservices' opriep. Deze 'services' omvatten de terminal-I/O en systeem-I/O. Programma's kunnen worden geschreven voor de voorgrondmachine in elke taal die beschikbaar is voor de 7094. Er werden bibliotheken beschikbaar gesteld zodat compilertalen zoals MAD gebruik konden maken van die 'supervisorservices'. CTSS leverde een bestandssysteem dat elke geregistreerde gebruiker een aparte map met bestanden gaf (Van Vleck, 2015).

De belangrijkste kenmerken van CTSS waren virtuele machines, hardware isolatie van gebruikers van de supervisor en van elkaar, en een bestandssysteem per gebruiker. Omdat de virtuele gebruikersmachine dezelfde architectuur en instructieset ondersteunde als de 7094, kon CTSS een groot aantal applicaties ondersteunen die oorspronkelijk waren ontwikkeld voor de FMS-batchomgeving bij MIT en elders (Van Vleck, 2015).

CTSS-software omvatte niet alleen de supervisor, maar ook een reeks voorgrond-opdrachten en subroutine bibliotheken. Veel van deze opdrachten lezen en schrijven bestanden van het bestandssysteem van de gebruiker. Zo las bijvoorbeeld de MAD-compiler een bestand met de broncode van een programma en schreef hetzelfde bestand met instructies in BSS(block starting symbol) formaat. Het grootste deel van de MAD-opdrachten op de voorgrond waren gelijk aan de MAD-compiler op de achtergrond op de FMS systeemtape, met de toevoeging van een wrapper voor het afhandelen van opdrachtprompt opties en vervangingen voor de invoer- en uitvoerroutines om bestanden te lezen en te schrijven in plaats van tape, en om foutmeldingen naar de terminal van de gebruiker te schrijven (Van Vleck, 2015).

In 1973 demonstreren Srodowa en Bates hoe je een virtuele machine kan maken op de IBM OS/360s. Ze beschrijven het gebruik van de IBM's virtuele machine monitor, de hypervisor om virtuele machines te maken en het efficiënt toewijzen van geheugen, opslag en I/O. Beiden hebben betrekking op onderwerpen die vandaag nog steeds worden bediscussieerd: prestatievermindering, capaciteit, CPU-toewijzing en opslag beveiliging. In 1974 concludeert Goldberg dat het merendeel van de toenmalige computersystemen geen

virtuele machines ondersteunen. De weinige virtuele machines die op dat moment operationeel zijn, bijvoorbeeld CP-67, gebruiken lastige en inadequate technieken vanwege ongeschikte architecturen. Goldberg stelt de Hardware Virtualizer voor, waarin een virtuele machine rechtstreeks zou communiceren met hardware in plaats van door de hostsoftware te gaan. Bijna 30 jaar later zijn brancheanalisten enthousiast over de aankondiging van hardware-architecturen die virtuele machines efficiënt kunnen ondersteunen. Zowel AMD als Intel onthulden specificaties voor Pacifica en Vanderpool chiptechnologieën met speciale virtualisatie-ondersteuningsfuncties (Radhwan, 2013).

De jaren tachtig en begin jaren negentig brachten het 'Distributed computing', wat een techniek is waarbij rekentaken niet door één enkele computer worden uitgevoerd, met zich mee naar de datacenters. Gecentraliseerd computergebruik en virtuele machines werden vervangen door stand-alone servers met specifieke taken: databank, web, applicaties, e-mail. Na aanzienlijke investeringen in gedistribueerde architecturen, is een hernieuwde focus op virtuele machines als aanvullende oplossing voor serverconsolidatieprojecten en datacenter beheerinitiatieven weer opgedoken. De virtuele machine is dus gemaakt op de mainframe. Het is pas onlangs geïntroduceerd op de middenklasse. Technologische vooruitgang in hardware en software maakt virtuele machines stabiel, betaalbaar en biedt enorme waarde, mits een juiste implementatie (Jeff, 2009).

Sinds begin 2000 is de opkomst van VMware hun VMware Workstation sterk gestegen. VMware werkt met een Hosted Architecture, dit wil zeggen dat de host een operating system nodig heeft zoals Windows of Linux. Om de complexe mix van prestaties, draagbaarheid, implementatiegemak, enz. te optimaliseren, fungeert het product zowel als een virtuele Machine Monitor(VMM) (rechtstreeks in gesprek met de hardware) als een applicatie die bovenop het hostbesturingssysteem draait. Dit laatste zorgt er voor dat de VMM niet moet omgaan met een groot aantal apparaten beschikbaar op de computers (Singh, 2004).

Het meest essentiële onderdeel van die virtualisatie is dus de Hypervisor, en wordt ook wel de Virtual Machine Monitor genoemd. De Hypervisor creëert een virtueel platform op de host om daarop meerdere gast besturingssystemen te draaien. Op deze manier kunnen meerdere dezelfde of verschillende exemplaren van een besturingssysteem draaien, en de hardwarebronnen delen die door de host worden aangeboden. Datacenters gebruiken tegenwoordig virtualisatie om abstractie te maken van de fysieke hardware, en deze bronnen aan gebruikers of klanten aan te bieden in de vorm van agile, schaalbare, geconsolideerde virtuele machines. Hoewel de technologie en de gebruiksredenen geëvolueerd zijn, blijft de kernbetekenis van virtualisatie hetzelfde: één host besturingssysteem in staat stellen om meerdere onafhankelijke besturingssystemen tegelijkertijd te laten draaien (Oracle, 2012).

2.1.2 Verschillende typen virtualisatie

Virtualisatie maakt een scheiding van onderliggende hardware mogelijk, wat ervoor zorgt dat je op een hoger niveau bewerkingen kan uitvoeren. Virtualisatietechnologie kan worden onderverdeeld in de volgende:

Full virtualisatie

In full virtualisatie is de hardware-interface die wordt aangeboden aan de hypervisor bijna dezelfde als die wordt geboden door het fysieke toestel. Dit betekent dat er voor virtualisatie weinig tot geen wijziging nodig is aan de besturingssystemen en applicaties als deze compatibel zijn met de hardware van het fysieke toestel. Deze virtualisatie kan worden onderverdeeld in zowel Bare-metal virtualisatie als Hosted virtualisatie. Bij bare-metal virtualisatie is er geen hostbesturingssysteem, dit wil zeggen dat de hypervisor rechtstreeks draait op de onderliggende hardware. Hosted virtualisatie wil zeggen dat het hostbesturingssysteem elk algemeen besturingssysteem kan zijn, zoals Windows, zoals Linux, etc. waarop de hypervisor draait. Deze virtualisatie heeft een extra softwarelaag die in het gast besturingssysteem draait en aan de hand van hulpprogramma's het mogelijk maakt om tijdens het gebruik van het gast besturingssysteem, toch bestanden te delen met de host (Kedia, 2013).

Para virtualisatie

Paravirtualisatie of para-virtualisatie is een virtualisatietechniek die een software-interface presenteert aan de virtuele machines die vergelijkbaar is, maar niet identiek, aan de onderliggende hardware/software interface. Het voordeel van deze techniek is dat het de overheadkosten voor virtualisatie vermindert en betere prestaties levert (Kedia, 2013).

Operating system virtualisatie

De kernel staat meerdere geïsoleerde besturingssystemen toe. Deze instanties draaien op een bestaand besturingssysteem en geven een reeks bibliotheken weer waarmee applicaties communiceren, waardoor ze de illusie krijgen dat ze worden uitgevoerd op een machine die speciaal voor het gebruik ervan bedoeld is. Deze virtualisatie staat beter bekend als container virtualisatie, wat later nog aan bod zal komen (Kedia, 2013).

Desktop virtualisatie

Desktop virtualisatie is een mechanisme om een PC omgeving met centrale applicaties vanaf een centrale server weer te geven. Het stelt gebruikers in staat applicaties voor verschillende besturingssystemen op een enkele host uit te voeren. Het biedt flexibiliteit om de applicaties en clients te verplaatsen wanneer nodig (Kedia, 2013).

Server virtualisatie

Bij server virtualisatie wordt basishardware gevirtualiseerd, waardoor gast besturingssystemen die er op draaien zonder de nood van de gehele operating system kunnen opereren. Servervirtualisatie is het proces waarbij een fysieke server wordt opgedeeld in meerdere unieke en geïsoleerde virtuele servers (Kedia, 2013).

Applicatie virtualisatie

Bij deze virtualisatie kan een gebruiker een servertoepassing lokaal draaien met behulp van lokale bronnen zonder de applicatie volledig lokaal te installeren. Het biedt elke gebruiker een geïsoleerde virtuele applicatieomgeving die fungeert als een laag tussen de host en het besturingssysteem. Een voorbeeld van deze soort virtualisatie is een Java Virtual Machine(JVM), aangezien het staat tussen het besturingssysteem en de java-applicatiecode (Kedia, 2013).

Storage virtualisatie

Opslag virtualisatie is een techniek waarbij de opslag van gegevens uit een fysieke opslag wordt opgehaald en over het netwerk wordt verspreid. Dit is een vorm van resource-virtualisatie. Dit wordt vaak gebruikt in een Storage Area Network(SAN) (Kedia, 2013).

Network virtualisatie

Netwerk virtualisatie is het proces waarbij hard- en softwarebronnen gecombineerd worden tot een virtueel netwerk als één enkele verzameling bronnen. Het zorgt voor een beter infrastructuur gebruik, zoals het hergebruik van poorten en fysieke bronnen, voor meerdere andere netwerkbronnen zoals hosts, virtuele machines, routers, switches etc. Zo helpt het ook bij verlagen van de kosten door netwerkbronnen te delen (Kedia, 2013).

Resource virtualisatie

Resource virtualisatie wordt beschouwd als opslagvolumes, namespaces en de netwerkbronnen in een gevirtualiseerd systeem. Alle onderdelen kunnen samen een grotere pool van bronnen worden en zo kan één enkele bron, zoals schrijfruimte, ook worden opgedeeld in een aantal kleinere en gemakkelijk toegankelijke bronnen (Kedia, 2013).

2.1.3 Container virtualisatie

Wat is container virtualisatie

Containergebaseerde virtualisatie is een lichtere benadering van virtualisatie met behulp van de host-kernel om meerdere virtuele omgevingen te draaien. Deze virtuele omgevingen worden meestal benoemd als containers. Containergebaseerde virtualisatie virtualiseert op het besturingssysteem niveau, en laat dus toe om meerdere applicaties te draaien zonder redundante host-kernels van andere machines. De containers zien eruit als normale processen van de buitenkant, die bovenop de kernel draaien die gedeeld wordt met de host. Ze bieden een geïsoleerde omgeving met de nodige middelen om applicaties uit te voeren. Deze middelen worden gedeeld met de host of afzonderlijk per container ter beschikking gesteld (Thanh, 2015).

Verschil tussen containers en virtuele machines

Containergebaseerde virtualisatie zorgt voor sommige voordelen tegenover een hypervisor-gebaseerde virtualisatie. Ten eerst zorgt een containergebaseerde virtualisatie voor een grotere dichtheid van virtuele omgevingen. Dit komt omdat een container geen volledig besturingssysteem heeft, de grootte en nodige middelen zijn ook minder om een applicatie in een container te draaien. Met als resultaat dat meerdere containergebaseerde virtuele machines opgezet kunnen worden op dezelfde machines t.o.v. traditionele virtuele machines. Ten tweede zorgen containers voor een betere performance (Cristian Ruiz, 2015).

Deze studies tonen aan dat de performance van containergebaseerde virtualisatie beter is dan met een hypervisor-gebaseerde virtualisatie in de meeste gevallen. Echter, desondanks het genoemde voordeel is containergebaseerde virtualisatie niet in staat om een verschil in virtuele omgevingen te ondersteunen, zoals een hypervisor dat wel kan. De oorzaak hiervan is, dat containers hetzelfde type moeten zijn als de host. Zo kan bijvoorbeeld een Windows host niet draaien op een Linux host. Aan de hand van een hypervisor-gebaseerde virtuele machine kan dat wel, waardoor je zowel Windows als Linux containers op een Linux host kan draaien (Cristian Ruiz, 2015).

Kubernetes

Kubernetes is een draagbaar, uitbreidbaar, opensource platform voor het beheren van gecontaineriseerde workloads en services, dat zowel manuele configuratie als automatisering mogelijk maakt (Kubernetes, 2021).

De kracht van Kubernetes is het uitvoeren van workloads op meerdere servers. Het is één van de beste tools als het gaat om containerclusterbeheer. Het voert de juiste containers op het juiste moment uit, schaalt ze op en neer op basis van belasting, lost hardware- of containerstoringen op en beheert netwerken en opslag (Rancher, 2020).

Origineel ontwikkeld door Google, geïnspireerd door een decennia van kennis door het uitrollen van schaalbare en betrouwbare systemen in containers door applicatie georiënteerde APIs. Sinds de introductie in 2014 is Kubernetes één van de grootste en meest populaire open source projecten in de wereld. Het is de standaard API geworden voor het bouwen van cloud-native applicaties, present in bijna elke publieke cloud. Kubernetes is een bewezen infrastructuur voor gedistribueerde systemen dat geschikt is voor cloud-native ontwikkelaars van alle schalen, zo kan het gaan over een cluster van een aantal Raspberry Pi's tot een warenhuis met de nieuwste computers. Het zorgt voor de nodige software om succesvol betrouwbare en schaalbare systemen op te zetten (Brendan Burns, 2019).

Alternatief voor Kubernetes

Hoewel er veel voordelen zijn voor het gebruik van Kubernetes, denk maar aan documentatie en support van de community, is het niet mis om de alternatieven te bekijken. Amazon Elastic Container Service, Mirantis Kubernetes Engine (Docker Enterprise) zijn

hier goede voorbeelden van. Daar dit onderzoek gaat over monitoring zal hier verder geen vergelijking plaatsvinden.

Docker

Docker is een source-platform waarop applicaties worden uitgevoerd en het proces gemakkelijker te ontwikkelen en distribueren is. De applicaties die in Docker gebouwd zijn, worden met alle ondersteunende afhankelijkheden verpakt in een zogenaamde container. De containers blijven op een geïsoleerde manier draaien bovenop de kernel van het besturingssysteem. De extra abstractielaag kan invloed hebben op de prestaties. Hoewel containervirtualisatie al meer dan 10 jaar bestaat, is Docker, een relatief nieuwe technologie, zonder twijfel één van de beste omdat het capaciteiten heeft dat voorgaande technologieën niet hadden (Babak Bashari Rad, 2017).

In eerste instantie biedt Docker de mogelijkheid om containers te maken en te controleren. Deze applicaties kunnen dan makkelijk in een containers gestoken worden. Daarnaast kunnen deze gevirtualiseerde applicaties makkelijk overal worden gebruikt zonder enige wijziging. Tenslotte kan Docker gemakkelijk samenwerken met externe programma's die helpen bij het opstellen en beheren van Docker containers (Babak Bashari Rad, 2017).

Alternatief voor Docker

Hoewel Docker één van de populairste containertechnologieën is, zijn er andere technologieën die Docker voorafgegaan zijn, die ontstaan zijn op hetzelfde moment of meer recentelijk zijn geïntroduceerd. Het basisconcept van de programma's zijn in grote lijnen gelijk en dan komt het vaak aan op technische verschillen. Voorbeelden van andere goede containertechnologieën zijn RKT(Rocket, CoreOS), LXD(Lexdi, Canoncical Ltd.), Linux VServer en Windows Containers. Maar zoals vermeld in het Alternatief voor Kubernetes gaat dit onderzoek niet verder gaan de vergelijken van deze applicaties.

Samenwerking tussen Docker en Kubernetes

In een Docker-cluster omgeving moeten enkele taken worden uitgevoerd, zoals plannen van deployments, communicatie en schaalbaarheid. Deze taken worden daarom uitgevoerd door orkestratietools, met als groot voorbeeld, Kubernetes. Kubernetes is één van de meest populaire projecten in de open source-geschiedenis (Journal, 2020).

Docker helpt bij het maken van containers, en Kubernetes laat toe dat de gebruiker ze tijdens runtime kan beheren. Docker wordt gebruikt voor het verpakken en verzenden van de applicatie en Kubernetes om de applicatie te implementeren en te schalen. Wanneer er gebruik gemaakt wordt van een klein aantal containers is het gebruik van Kubernetes nog niet nodig, maar vanaf het aantal containers begint op te schalen is het wel aan te raden. Dit is ook waar Kubernetes in uitblinkt (Journal, 2020).

Wanneer ze samen worden gebruikt, dienen Docker en Kubernetes voor het inschakelen van digitale transformatie en als tool voor moderne cloudarchitectuur. Het gebruik van

beide is een nieuwe norm in de branche en zorgt voor een snellere implementatie en release van applicaties (Journal, 2020).

2.2 Monitoring

2.2.1 Wat is monitoring

Monitoring is het proces om metrische gegevens te verzamelen over de werking van de hardware en software van een IT-infrastructuur om ervoor te zorgen dat alles functioneert zoals verwacht om applicaties en services te ondersteunen. Basis monitoring wordt uitgevoerd door middel van controle uit te voeren over de werking van een apparaat, terwijl meer geavanceerde monitoring gedetailleerde overzichten geeft van de operationele statussen, inclusief gemiddelde reactie tijden, aantal applicatie-instanties, foutboodschappen, CPU-gebruik en applicatie beschikbaarheid (TechTarget, 2020).

IT monitoring verspreidt zich over 3 secties:
Basis. De infrastructuur is de onderste laag van een softwarestack en omvat zowel fysieke als virtuele apparaten, zoals servers, CPU's en VM's.
Software. Dit deel wordt ook wel de bewakingssectie genoemd en het analyseert wat er op de apparaten in de basis gebeurd, inclusief CPU-gebruik, geheugen gebruik, load op de servers enzovoort.
Interpretatie. Verzameld statistieken die worden gepresenteerd via grafieken of diagrammen, vaak op een GUI-dashboard. Dit wordt vaak bereikt door integratie met tools die specifiek gericht zijn op datavisualisatie.

Monitoring kan met of zonder 'agents'. Een agent is een onafhankelijk programma die op een apparaat wordt geïnstalleerd om gegevens over hardware of software, prestatiegegevens verzameld en rapporteert aan een monitoringserver. Wanneer er geen agent is, zal de monitoringserver gebruik maken van communicatieprotocollen om een agent te emuleren met ongeveer dezelfde functionaliteiten om zo gegevens te verzamelen (TechTarget, 2020).

Er zijn ook twee soorten monitoring wanneer het over de frequentie gaat. Enerzijds is er 'Real Time Monitoring' wat een techniek is waarbij systemen continu gegevens verzamelen en toegankelijk zijn door IT-teams om de actieve en lopende status te bepalen. Metingen van real-time monitoringssoftware geven gegevens weer uit de huidige IT-omgeving, evenals het recente verleden, waardoor IT-personeel snel kunnen reageren op actuele gebeurtenissen. Door data op deze manier te monitoren, kan het IT-personeel de infrastructuur verbeteren of mogelijke complicaties identificeren voordat ze zich voordoen, omdat ze een patroon of trend in de gegevens herkennen. Anderzijds is er ook trend monitoring, een analyse van één specifieke gebeurtenis op een bepaald moment. Dit kan gebruikt worden om een probleem te identificeren dat onmiddellijk moet worden verholpen, zoals een 100% volle schijf of overmatig gebruik van de CPU (TechTarget, 2020).

2.2.2 Wat is monitoring binnen een container environment

De verandering van monitoring binnen een container omgeving ten opzichte van een hypervisor omgeving kan uitgelegd worden in 5 verschillende punten:

- De korte duur van containers
- De snelle groei van objecten, services en statistieken om te loggen
- Services zijn het nieuwe focuspunt van monitoring
- Een meer diverse groep van monitoring-eindgebruikers
- Nieuwe gedachten resulteren in nieuwe methoden

Korte duur en schaal van containers

De tijdelijke aard van containers en instanties van virtuele machines zorgt voor uitdagingen op het gebied van tracking. Omdat een container zeer dynamisch is, vereist dit nieuwe monitoringmethoden om observaties te doen over de status van die container. Door de korte duur van containers en de snelgroeiende schaal heeft het weinig zin om container per container te bekijken, in plaats daarvan wordt aangeraden om clusters (meerdere containers) te volgen/monitoren. Traditionele benaderingen van monitoring zijn gebaseerd op het introduceren van datacollectors, genaamd 'agents' of 'remote access hooks' in de systemen zelf. Deze kunnen de snelheid van de provisioning en het dynamisch opschalen echter niet aan. Je kan op twee manieren data opvangen van een server, enerzijds door te 'pollen' en anderzijds door te 'pushen'. De tweede aanpak heeft een voorkeur voor een container omgeving, daar deze containers van korte duur zijn en omdat het te lang kan duren om deze containers te vinden en te pollen. Om te kunnen pushen wordt een agent geïnstalleerd, wat het 'pol'-proces elimineert (Williams, 2016).

Snelle groei van objecten, services en statistieken

De explosie van gegenereerde gegevens is een bekend fenomeen. Vroeger vroegen mensen zich af hoe ze al die gegevens moeten opslaan, tegenwoordig ligt de nadruk meer op hoe die gegevens gebruikt kunnen worden zonder ze op te slaan. Door de opkomst van Internet of Things(IoT)-sensoren en de introductie van containers, zijn er nu meer objecten dan ooit om te monitoren. De overvloed aan datapunten, metrieken en objecten die moeten worden gevolgd, is een ernstig probleem. Het streamen van gegevens biedt veel mogelijkheden voor realtime analyses, maar het moet wel nog worden verwerkt en opgeslagen. Er zijn technische oplossingen die de schaal aankunnen, maar tegen aanzienlijke kosten voor zowel financiën als prestaties. Hoewel er wel databases van de volgende generatie zijn(NoSql), zijn ze niet geoptimaliseerd voor deze use case. Loggegevens kunnen echter niet voor onbepaalde tijd bewaard worden. Sommige oudere bestanden worden nooit gebruikt, wat gebruikers motiveert om eerder te focussen op metrische gegevens in plaats van logs. Metrische gegevens zijn gegevens die in geaggregeerde vorm of op regelmatige tijdstippen worden verzameld (Williams, 2016).

Services zijn het nieuwe aandachtspunt

Met een hernieuwde focus op wat er daadwerkelijk gemonitord moet worden, zijn er drie aandachtsgebieden: de gezondheid van containerclusters, microservices en toepassingen. Het beoordelen van clusters van containers - in plaats van afzonderlijke containers - is een betere manier voor infrastructuurbeheerders om de impact van services te begrijpen. Hoewel het waar is dat applicatiebeheerders individuele containers kunnen uitschakelen en herstarten, zijn ze meer geïnteresseerd in het begrijpen welke clusters gezond zijn. Met deze informatie kunnen ze het cluster inzetten op een andere infrastructuur of extra bronnen toevoegen om de optimale werking ervan te ondersteunen. Containerorkestratie-oplossingen helpen door een efficiënte planning van containers op clusters van hosts mogelijk te maken. Volgens Dynatrace Cloud Technology Lead Alois Mayr, in een interview met The New Stack: ”als je kijkt naar applicatiebewaking, ben je vooral geïnteresseerd in de services die in containers worden uitgevoerd, in plaats van in de containers zelf” (Williams, 2016).

Een meer diverse groep van monitoring-eindgebruikers

De focus op het monitoren van applicaties in plaats van alleen op infrastructuur gebeurt om twee redenen. Allereerst is er een nieuwe groep mensen bij de monitoring betrokken. Ten tweede zijn applicaties relevanter voor de algehele bedrijfsprestaties.

Ondanks de vooruitgang van de afgelopen jaren is de monitoring over het algemeen nog steeds reactief. Het is gericht op de doelstellingen van het IT-team dat de feitelijke infrastructuur beheert. Deze mentaliteit bewijst ontwikkelaars een slechte dienst, omdat ze over het algemeen gegevens uit de tweede hand ontvangen. Ontwikkelaars worden steeds vaker aangesproken op applicaties nadat ze in productie zijn genomen. Zoals de ”Effective Performance Engineering” van Todd DeCapua en Shane Evans opmerkt, wordt ontwikkelaars gevraagd om ”producten van de hoogste kwaliteit te leveren en continue feedback en optimalisatieaanbevelingen te geven, zodat andere teams snel en volledig geautomatiseerd kunnen opleveren.” De DevOps-beweging is gestegen, althans gedeeltelijk, als reactie op het verlangen van ontwikkelaars naar meer zichtbaarheid gedurende de volledige levenscyclus van een applicatie. Nu zijn DevOps-rollen vaak de volledige stackbeheerders en operators van applicaties.

Samen met DevOps zal de praktijk van ”site reliability engineering (SRE)” van invloed zijn op het gebruik van monitoringtools. Vanuit dit perspectief zal de monitoring nog steeds grotendeels worden beheerd door een operationeel team, maar de verantwoordelijkheid om ervoor te zorgen dat nieuwe applicaties en services worden gecontroleerd, kan worden gedelegeerd aan applicatieontwikkelaars. Shariq Rizvi, mede-oprichter van Netsil, zei in een interview met The New Stack dat SRE’s en DevOps-ingenieurs anders zijn dan software-ingenieurs. Hij vindt dat SRE-teams het beheer van diensten moeten opsplitsen, waardoor er meer specialisatie ontstaat. Dan Turchin, mede-oprichter en ”chief product officer” van Neva, zei in een interview met The New Stack dat hij gelooft dat DevOps-functies de technici van het ”Network Operations Center (NOC)” vervangen, die traditioneel vanuit het perspectief van een datacenter naar de dingen keken. Als de ouder-

wetse netwerkstatistieken worden verdrongen door statistieken van de cloudinfrastructuur, kan dit waar zijn (Williams, 2016).

Nieuwe gedachten resulteren in nieuwe methoden

Hoewel monitoring verandert om aan de behoeften van verschillende functies te voldoen, gaat het ook over een meer alomvattende benadering. Majors Charity, blogger bij "Honeycomb.io"(Majors, 2016), vertelt dat in plaats van te vertrouwen op een vaste reeks vragen en controles, mensen zouden moeten evolueren naar de observeerbaarheid van systemen. Dit moet gebeuren omdat die vaste datapunten, alleen, niet de benodigde inzichten opleveren. Er zijn nieuwe tools nodig om gelijke tred te houden en de mogelijkheid te bieden om te voorspellen wat er gaat breken. Veel van deze tools maken gebruik van artificiële intelligentie-technieken.

Observeerbaarheid wil niet zeggen dat testen het probleem altijd identificeert. Majors is dan ook van mening dat instrumentatie net zo belangrijk is als testen. Als je complexe systemen gebruikt, kun je niet alles in je hoofd modelleren. Naast het veranderen van de instrumentatie, stelt ze ook voor om te focussen op het begrijpbaar maken van de monitoringssystemen. Dit betekent feitelijk definiëren wat de gegevens vertegenwoordigen en dezelfde definities gebruiken als uw collega's, zowel binnen als buiten de organisatie. Verder is er ook frustratie over de noodzaak om door meerdere, statische dashboards te scrollen. Als reactie hierop maken ontwikkelaars meer intuïtieve, interactieve dashboards en maken zelf gebruik van artificiële intelligentie om voor elke server te bepalen welke informatie wanneer wordt weergegeven (Williams, 2016).

2.2.3 Hoe werkt monitoring binnen een container omgeving

Om ongelijksoortige metrische bronnen samen te brengen, is een robuust platform voor infrastructuurmonitoring vereist dat een enkel dashboard kan bieden om gegevens uit verschillende bronnen te bekijken. Het vereist ook veel aandacht en planning om ervoor te zorgen dat gegevens kunnen correleren om eenvoudige end-to-end foutopsporing mogelijk te maken.

Conceptueel worden gecontaineriseerde applicaties op dezelfde manier gemonitord als traditionele applicaties. Het vereist gegevens op verschillende lagen in de stacks voor verschillende doeleinden. Het meet en verzamelt metrische gegevens zowel op de container zelf als op de infrastructuurlaag voor een beter resourcebeheer. Er zijn ook toepassingsspecifieke gegevens, prestatiebeheer van toepassingen en traceringsinformatie nodig voor het oplossen van problemen met toepassingen.

Robuuste observatie en eenvoudige foutopsporing zijn de sleutel voor het bouwen van een microservices-architectuur. Naarmate uw systeem groeit, neemt ook de complexiteit van het bewaken van de gehele applicatie toe. Traditioneel wordt een onafhankelijke dienst gebouwd die maar één doel dient, en die communiceert met andere diensten om een grotere taak uit te voeren en dit allemaal naast bedrijfslogica. Deze taken worden echter erg complex als je honderden containers hebt die verschillende programmeertalen

in verschillende teams gebruiken. Dit is waar service-mesh in beeld komt.

Een servicemesh beheert de communicatielaag van uw microservicesysteem door een softwarecomponent als een sidecar proxy (een ontwerp dat bepaalde functies zoals communicatie tussen services, monitoring en beveiliging abstraheert om het volgen en onderhouden van de applicatie als geheel vergemakkelijkt (Gilles, 2019)) te implementeren. Dit softwarecomponent voert netwerkfuncties, verkeerscontrole, tracering en logboekregistratie uit. Elke service heeft zijn eigen proxyservice en alle proxyservices samen vormen de mesh. Elke service communiceert niet rechtstreeks met andere services, maar alleen met zijn eigen proxy. De service-to-service-communicatie vindt plaats tussen proxy's.

Met een servicemesh kan een netwerk van microservices gebouwd worden en het grootste deel van de netwerklaag gescheiden worden om het beheer en de waarneembaarheid te stroomlijnen. Als er zich problemen voordoen, kan de oorzaak van het probleem snel achterhaald worden (AWS, 2021).

2.3 Wat is het nut van monitoring in het lessenpakket

Dit onderdeel van de literatuurstudie moet duidelijkheid creëren waarom het lessenpakket monitoring moet bevatten voor de studenten van Toegepaste Informatica. Hoewel dit onderzoek vooral focus legt op monitoring in een containeromgeving, zal hier toch de nadruk gelegd worden op monitoring in het geheel. Daar dit eigenlijk zo goed als overeen komt.

2.3.1 Waarom zou je monitoring toepassen

Quality of service (QoS) is zo goed als de infrastructuur waaronder service wordt geleverd. Helaas besteden veel IT-afdelingen meer tijd aan het reageren op onverwachte problemen dan aan het voorkomen ervan. Het bewaken van de gehele IT-omgeving is een bedrijfskritische taak die zorgt voor optimale prestaties en bedrijfscontinuïteit (Matrix, 2016).

Monitoring is cruciaal omdat problemen niet altijd direct duidelijk zijn, maar de neiging hebben om zich in de loop van de tijd op te bouwen. De rol van IT als een belangrijke motor van operationele prestaties en innovatie maakt het een verplichting van netwerkbeheerders om een zeer responsief controlesysteem te implementeren om alle netwerkactiviteiten bij te houden (Matrix, 2016).

Er zijn een aantal belangrijke redenen waarom het hebben van een volledig zicht op de IT-infrastructuur belangrijk is voor bedrijven:

Volledige controle over IT-middelen

Het IT-netwerk is het hart van moderne bedrijven. Om de prestaties van hardware- en softwarebronnen bij te houden, moet de IT uit de eerste hand kennis hebben van de gezondheid van het netwerk en volledige controle hebben over trends in het gebruik van bronnen, servercapaciteiten en kwaadwillende activiteiten om problemen te voorkomen voordat ze zich voordoen (Matrix, 2016).

Geïnformeerde besluiten

IT-monitoring is meer dan alleen het bijhouden van infrastructuuractiviteiten. Het is een essentieel onderdeel van het creëren van een algemene IT-strategie voor bedrijven. Van CIO's (chief information officer) wordt verwacht dat ze aantonen dat ze de controle hebben over de IT-omgeving en dat ze toegang hebben tot kritieke gegevens voor belangrijke en onbelangrijke besluiten. Efficiënte monitoring wordt geleverd met effectieve rapportage om ervoor te zorgen dat alle belanghebbenden worden geïnformeerd en dat aan de belangrijkste prestatie-indicatoren wordt voldaan (Matrix, 2016).

Verhoogde productiviteit en verbeterde servicebeschikbaarheid

Wanneer de prestaties worden verbeterd, wordt de productiviteit ook verhoogd. Het is dus belangrijk voor IT-teams om de interacties van de verschillende systemen binnen het netwerk nauwlettend in de gaten te houden om gebieden te identificeren die verbetering behoeven. Kwetsbare gebieden hebben speciale aandacht nodig om te anticiperen op mogelijke problemen en om preventieve oplossingen te bieden.

Serviceonderbreking als gevolg van downtime is een vloek voor eindgebruikers. Een netwerk vertraging kan bijvoorbeeld veroorzaakt worden door eindgebruikers die te veel bronnen tegelijk willen gebruiken. Regelmatig en willekeurig controleren op de gebruikspatronen kan ervoor zorgen dat een evenwichtige werklastverdeling kan worden gecreëerd om een continue beschikbaarheid van de service te garanderen (Matrix, 2016).

Probleemdetectie voordat zich een ramp voordoet

Onvoorspelbare catastrofes worden veroorzaakt door de natuur en zijn vaak niet te voorkomen, maar door de mens veroorzaakte rampen, of ze per ongeluk of opzettelijk zijn is bijzaak, kunnen worden voorkomen door een robuust monitoringsysteem. Storingspatronen, frequentie van downtime, ontbrekende, verwijderde of toegevoegde bestanden, evenals bestanden en programma's die niet openen met legitieme wachtwoorden, zijn waarschuwingssignalen die moeten worden gecontroleerd en opgelost om een vermijdbare ramp te voorkomen(Matrix, 2016).

Besparingen en hogere winst

Kosten besparen en winst verhogen zijn legitieme zakelijke doelstellingen. Preventie door monitoring betekent dat er minder geld wordt uitgegeven in vergelijking met de kosten van oplossingen voor problemen die zich plotseling voordoen. Proactieve monitoring van infrastructuur helpt financiële verliezen te voorkomen als gevolg van niet-gedetecteerde uitval van systemen als gevolg van slechte (of een gebrek aan) monitoring (Matrix, 2016).

2.3.2 Waarom zou je een container monitoren

Hoewel containers vluchtig zijn is inzicht in metrische gegevens, logboeken en traceringen alleen maar voordelig voor operatoren van deze gecontaineriseerde platformen. Door te begrijpen wat er gebeurt, en niet alleen op cluster- of hostniveau, maar ook binnen de containerruntime en -toepassing, kunnen organisaties beter geïnformeerde beslissingen nemen, zoals wanneer instanties, taken of pods moeten worden geschaald in grootte. DevOps of systeemingenieurs kunnen ook de efficiëntie en snelheidsresolutie verbeteren door automatisering toe te voegen, zoals het gebruik van automatische waarschuwingen om het schalen van containers dynamisch te activeren. Door bijvoorbeeld het geheugengebruik actief te bewaken, kan er een drempelwaarde geïdentificeerd worden en operators op de hoogte gesteld worden wanneer het resourceverbruik de resourcelimiet bereikt, of door automatisch extra nodes toe te voegen vóór de beschikbare CPU of RAM uitgeput is (AWS, 2021).

2.3.3 Conclusie

Zoals beschreven in de bovenstaande alinea's is het monitoren van een containeromgeving en een gehele it omgeving alleen maar voordelig. Door middel van monitoring kunnen veel zaken voorkomen worden waardoor kosten en moeite gespaard blijven. Maar vaak wordt monitoring over het hoofd gezien omdat het voor bedrijven een kost is zonder opbrengst, wat ongetwijfeld niet waar is. Daarom is het belangrijk om het concept van monitoring vroeg aan te leren om monitoring in een IT-omgeving te standaardiseren. Met de opkomst van containers is dit onderzoek dus zeker op zijn plaats.

2.4 Welke tools zijn er om te monitoren in een containeromgeving

Hoewel deze studie focus legt op de metrische gegevens van containers, zullen er in deze opsomming ook tools staan, die ook dienen voor logs te monitoren. In het volgende hoofdstuk zal dan de vergelijking van de tools plaatsvinden om zo de, in dit onderzoek, minder interessante tools te filteren.

2.4.1 Docker Stats

Op een basis level kan je aan de hand van een ingebouwd commando Docker containers monitoren. Er kan gedetailleerde, realtime informatie verkregen worden over het gebruik van de containers. Deze informatie bestaat uit, CPU gebruik, geheugen gebruik, schijf -en netwerk-I/O en het aantal actieve processen. ”Docker Stats” bewaart geen geschiedenis en kan maar 1 host per keer monitoren. Aan de hand van scripts kunnen wel gegevens verzameld worden van meerdere host en zo bewaard worden. Op zich is de informatie die verzameld wordt minimaal maar door een combinatie met ”Docker logs” en ”Docker events” kan de monitoring naar een hoger niveau getild worden (Sissons, 2021).

2.4.2 cAdvisor

cAdvisor is een open-source project, oorspronkelijk ontwikkeld door Google, dat net als ”Docker Stats” de gebruikers informatie biedt over gebruik van bronnen tijdens het uitvoeren van containers. Net zoals ”Docker Stats” is het ook geïmplementeerd als een daemon-proces. Het verzamelt, aggregeert, verwerkt en exporteert informatie over de actieve containers.

Deze tool heeft een webinterface en kan meerdere grafieken genereren. Maar net zoals ”Docker Stats” kan cAdvisor maar één host monitoren. Indien de verzamelde informatie moet worden opgeslagen, moet ook een externe database geconfigureerd worden zoals Prometheus en InfluxDB. cAdvisor is op zich geen complete monitoringoplossing en dus vaak een onderdeel van andere monitoringtools (Sissons, 2021).

2.4.3 Scout

Scout is een monitoring applicatie die een monitoringservice biedt op cloudgebasseerde applicaties en cloudgebasseerde databanken. Scout biedt een uitgebreide functionaliteit voor het verzamelen, filteren en monitoren van gegevens met flexibele alerts en integraties met alerting-services van derden. Het team van Scout geeft documentatie voor het schrijven van scripts om gebruik te maken van de Docker Stats API (zoals hierboven vermeld) en geeft metrische gegevens door aan Scout voor monitoring. Er is een Docker-Scout-container beschikbaar op Docker Hub, die het installeren en configureren een stuk eenvoudiger maakt (Sissons, 2021).

2.4.4 Pingdom

Net zoals Scout, is Pingdom ook een cloudgebasseerde applicatie, beheerd door SolarWinds, een bedrijf dat zich richt op het monitoren van IT-infrastructuur. Bij Pingdom is belangrijkste domein het monitoren van websites, maar deze tool kan ook gebruikt worden voor servermonitoring met maar liefst 90 plugins. Pingdom onderhoudt docker-scout, dit is dezelfde StatsD-agent die door Scout gebruikt wordt (Sissons, 2021).

2.4.5 Datadog

Datadog is net zoals de twee vorige tools een cloudmonitoring service. Datadog biedt ook een container-agent voor installatie op een Docker-host. In plaats van generieke StatsD te gebruiken zoals de eerder genoemde cloudbewakingsoplossingen, heeft Datadog een verbeterde StatsD ontwikkeld, DogStatsD genaamd.

De Datadog-agent verzamelt de volledige set metrische gegevens die beschikbaar zijn via de Docker-API en geeft deze door, waardoor een zeer gedetailleerde monitoring wordt geboden. Datadog biedt een goede toegang tot metrische gegevens en meer mogelijkheden bij het definiëren van waarschuwingsvoorwaarden dan de eerder genoemde cloudservices. Net als de andere services kan Datadog ook worden gebruikt om andere services en applicaties te monitoren, en beschikt het over een bibliotheek met meer dan 200 plugins. Datadog bewaart ook gegevens gedurende 18 maanden, wat langer is dan de bovenstaande cloudservices (Sissons, 2021).

2.4.6 Sysdig

Sysdig is voornamelijk een cloudgebasseerde monitoringoplossing, gericht op containers. Een deel van zijn functionaliteiten zijn beschikbaar in open-source projecten, en Sysdig biedt ook om de applicatie lokaal te installeren (Sissons, 2021).

2.4.7 Prometheus

Prometheus is een populaire, open-source tool voor monitoring en alerting, oorspronkelijk gebouwd door de developers bij SoundCloud. Waar Prometheus verschilt tegenover de cloudgebasseerde tools, is dat de tool modulair is en zelf wordt gehost, wat betekent dat gebruikers de tool op hun clusters inzetten. Prometheus installeert op elke Docker-host een agent en aan de hand van een 'pull' protocol haalt het zo alle gegevens binnen via HTTP bij een reeks exporteurs. Sommige van deze exporteurs worden officieel onderhouden door Prometheus zelf, terwijl de overschot externe bijdragen zijn. Prometheus biedt functies zoals service-detectie, een aparte push-gateway voor specifieke typen metrieken, en heeft een ingebouwde querytaal (PromQL) die dient voor het opvragen van multidimensionale gegevens. Het heeft ook een uitgebreide ingebouwde webinterface en API. Deze GUI is wel sterk afhankelijk van de gebruikers die wat kennis bezitten van PromQL. Daarom ondersteunt Prometheus ook tools van derden om als dashboard te gebruiken, zoals Grafana. (Sissons, 2021).

2.4.8 ELK/EFK Stack

De ELK/EFK-stack is een open-source software stack die het monitoren van containers op zich neemt. Deze stack bestaat uit drie open-source projecten die zijn samengevoegd om de ELK/EFK stack te bekomen. De stack is veelzijdig en wordt veel gebruikt voor een verscheidenheid aan analytische toepassingen, waarbij het monitoren van logbestanden

een belangrijke rol speelt. ELK/EFK is genoemd naar drie componenten.

- Elasticsearch: wordt gebruikt om de logs op te slaan
- Logstash/Fluentd: wordt gebruikt voor zowel het transporteren als het verwerken en opslaan van logs, die dan doorgestuurd worden naar Elasticsearch
- Kibana: visualisatietool gehost door Nginx of Apache die de data van Elasticsearch gebruikt

Alhoewel de gelijkenissen tussen Logstash en Fluentd groter is dan de verschillen, gebruiken de meeste developers Fluentd. Maar zelfs gebruikers van Logstash staan voor op de curve als het aankomt op log management (Sissons, 2021).

2.4.9 Sensu

Sensu is een self-hosted monitoringtool voor algemene doeleinde die een verscheidenheid aan monitoringstoepassingen ondersteunt. Sensu maakt gebruikt van Agents die de nodige containers monitoren. Het aantal Sensu-plug-ins blijft groeien en er zijn tientallen door Sensu en community ondersteunde plug-ins waarmee metrische gegevens uit verschillende bronnen kunnen worden opgehaald. Sensu heeft vele features, maar een nadeel voor de gebruikers is dat het framework moeilijk te installeren, configureren en onderhoud is omdat de componenten zelf niet ”Dockerized” zijn (Sissons, 2021).

3. Methodologie

In dit hoofdstuk zullen alle tools uit hoofdstuk 2.4 vergeleken worden met elkaar om zo een best passende tool voor het onderzoek te vinden. Het onderzoek zal aan de hand van de ”MoSCoW’-techniek bepalen wat de vereisten zijn.

- Must have: alle noodzakelijke eigenschappen
- Should have: eigenschappen die niet verplicht zijn, maar wel een meerwaarde zijn
- Could have: eigenschappen die een kleine meerwaarde hebben
- Would have: alles wat een meerwaarde kan hebben maar niet nodig is voor dit onderzoek

Tenslotte wordt de opstelling van het onderzoek uitgelegd. Aan de hand van dit onderzoek zal een antwoord geformuleerd worden op de onderzoeksvraag.

3.1 MoSCoW strategie

3.1.1 Must have

- De tool moet volledig gratis zijn.
- De volledige PoC, inclusief de tool moet lokaal geïnstalleerd zijn.
- De PoC moet mogelijk zijn op de laptop van de studenten met beperkte specificaties.
- De PoC moet op niveau zijn van de studenten uit 2de of 3de graad. Hiervoor is een basiskennis vereist van:
  - Virtualbox
  - Linux
- Via de Proof-Of-Concept moeten de functies van de monitoringtool duidelijk zijn om de I/O metrics te analyseren.

3.1.2 Should have

- Aan de hand van één enkel commando kan de Proof-of-Concept opgezet worden zodat de studenten direct aan de slag kunnen met de monitoringtool.

3.1.3 Could have

- Alerting.
- Een basiskennis kan worden verwacht van de studenten voor:
  - Docker
  - Kubernetes

3.1.4 Would have

- Het verzamelen en analyseren van logs.

3.2 Onderzoeksmethodologie

Eerder vermeld in dit onderzoek, luidt de volgende onderzoeksvraag:

- Welke monitoringtool voor een Kubernetes/Docker environment is het meest geschikt voor de studenten Toegepaste Informatica om zo een goede basiskennis te bekomen.

Eveneens zijn er nog een aantal deelvragen die ook belangrijk zijn:

- Wat zijn de belangrijkste verschillen tussen de tools?
- Waar moeten de tools inzicht op geven?
- Welke alerts zijn belangrijk?
- Welk proof-of-concept is voldoende om de tool aan te leren?

Om een antwoord te formuleren op de vragen wordt een onderzoek gevoerd in verschillende stappen.

- Stap 1: Vergelijken van de tools
- Stap 2: Opstellen van een Proof-of-Concept
- Stap 3: Checklist overlopen met de vereisten

4. Conclusie

A. Onderzoeksvoorstel

Het onderwerp van deze bachelorproef is gebaseerd op een onderzoeksvoorstel dat vooraf werd beoordeeld door de promotor. Dat voorstel is opgenomen in deze bijlage.

A.1 Introductie

Containertechnologie vindt in recordtempo zijn weg naar de data-omgeving van de ondernemingen (Cole, 2016). Het gemak waarmee containerplatformen zoals Docker kunnen worden ingezet, suggereert dat ze de dominantste architectuur zijn en zullen blijven voor deze en de volgende generatie services en microservices. De uitdaging om een goede monitoring van containers op te stellen is belangrijk. Zoals te verwachten, zullen traditionele monitoringsplatforms die vooral gebaseerd zijn op virtualisatie, niet voldoende zijn. Containers zijn zeer vluchtig in ontstaan en net zo snel in het verdwijnen. Die snelheid is vaak te danken aan een geautomatiseerd proces waarbij een minimale input vereist is. Deze containers bevinden zich tussen host -en applicatielaag, wat het moeilijk maakt om in detail te zien hoe ze zich gedragen en of ze efficiënt gebruik maken van hulpbronnen (e.g. CPU, RAM, etc.) en goede systeem prestaties leveren. De valkuil voor vele bedrijven is het overschatten van een eenvoudige hostmonitoring voor een container. Dit idee wordt echter al snel uit de weg geruimd, omdat het aantal containers snel begint te vergoten wat een traditioneel hostmonitoring niet kan bijhouden. Binnen HoGent is er voorlopig geen opleidingsonderdeel omtrent monitoring, specifiek op een Docker omgeving in een Kubernetes orkestratie. Daar dit een zeer interessant en relevant onderwerp is, is het een goede zaak om deze leerstof bij te leren aan de toekomstige studenten van HoGent. De onderzoeksvraag komt in dit geval uit van een docent tewerkgesteld in HoGent. Omdat er nog geen intern onderzoek heeft plaatsgevonden, is deze onderzoeksvraag dus ontstaan.

De doelstelling van dit onderzoek is bepalen welke monitoringtool hiervoor geschikt is en welke haalbaar is voor de studenten.
Zo bekomen we volgende onderzoeksvragen:

- Welke monitoringtools zijn geschikt?
- Wat zijn de belangrijkste verschillen tussen de gekozen tools?
- Waar moet de gekozen tool inzicht op geven?
- Welke alerts zijn belangrijk en welke niet?
- Welk proof-of-concept is genoeg om de leerstof te verstaan?

A.2 State-of-the-art

Hoewel er vele onderzoeken zijn naar goede monitoringtools voor Docker, zijn het vooral onderzoeken waarbij ook de betalende tools vergeleken worden (Ribenzaft, 2020) en (Cirelly, 2020). Deze onderzoeken specificeren zich ook niet op één specifieke 'proof-of-concept' maar leggen vooral de voor- en nadelen van elke tool uit. Hoewel het boek van Alex Williams (Cole, 2016) ook niet echt een 'proof-of-concept' heeft, komt het onderzoek toch dichter in de buurt van wat het verwachte resultaat is van deze bachelorproef. Hierin wordt ook over een aantal andere onderwerpen gesproken omtrent monitoring die zeker interessant zijn. Daar een deel van dit boek ook een aantal tools beschrijft en vergelijkt is dit zeker relevant aan de onderzoeksvraag. De bedoeling van deze bachelorproef is het onderzoeken van zowel zelf-gehoste open source-oplossingen als commerciële cloud-gebaseerde services te bekijken, dit boek kaart ook deze opties aan. Volgens het boek van Alex Williams (Cole, 2016) is de conclusie dat de keuze sterk afhangt van de resultaten die je wenst te bereiken die bij uw werklast passen, eventueel met een combinatie van extra tools. Door opzoekingswerk en verzameling van interne informatie worden de vereisten bepaald,zo kan er gekozen worden voor de juiste monitoringtools. Tot dusver vind is er geen exacte kopie van de onderzoeksvraag, wat deze bachelorproef uniek maakt en interessanter.

A.3 Methodologie

Allereerst zal een MoSCoW requirements-analyse opgesteld worden aan de hand van een interview met mogelijke belanghebbende. Voor dit onderzoek zal vooral de mening van docenten die dit onderwerp zullen voordragen aan de studenten belangrijk zijn. Hierna zal een literatuurstudie opgesteld worden waarin volgende onderwerpen zullen afgehandeld worden:

- Concept monitoring, soorten monitoring, taken van monitoring
- Architectuur monitoringsystemen in de context van containers/Kubernetes
- Overzicht aanbod producten/tools in dit marktsegment

Vervolgens zullen de gevonden producten/tools afgetoetst worden aan de requirements die voordien zijn opgesteld door de belanghebbende. Nadien wordt een bestaande 'proof-of-concept' gebruikt (Cedric, 2019) waarop de meest belovende producten toegepast worden. Indien er in deze 'proof-of-concept' tekortkomingen zijn voor het bekomen van de vooraf bepaalde requirements, zal deze verwerkt worden tot het punt waar er deze tekortkomingen miniem zijn.

A.4 Verwachte resultaten

Uit deze studie, naar een gepaste monitoringtool, wordt verwacht dat aan de hand van de experimenten, het duidelijk wordt welke tool een basis kan zijn voor een deel van het opleidingsonderdeel waarop beslist wordt wat toekomstige studenten kunnen gebruiken. Het opstellen van een goede proof-of-concept is uiterst belangrijk voor het bereiken van goede resultaten.

A.5 Verwachte conclusies

Het experiment zou een goede basis moeten hebben om zo de essentiële onderdelen van monitoring aan te leren aan studenten toegepaste informatica. Met een basis wordt vooral bedoeld dat de 'proof-of-concept' makkelijk reproduceerbaar is en dat de 'beste' monitoringtool en zijn functionaliteiten allemaal uit te voeren zijn. Bovendien is het de bedoeling dat de 'proof-of-concept' alle deelaspecten van monitoring bevat met inzicht op, in geval van dit onderzoek, zicht op het voortdurend meten van allerlei performantiemetrieken. Indien het experiment aan deze eisen voldoet, mogen we concluderen dat dit een geslaagd onderzoek is.

Bibliografie

AWS. (2021, januari 1). *Container Monitoring - Why, how, and what to look out for* (AWS, Red.). Verkregen 7 mei 2021, van https://aws.amazon.com/cloudwatch/container-monitoring/

Babak Bashari Rad, M. A., Harrison John Bhatti. (2017, maart 1). *An Introduction to Docker and Analysis of its Performance* (tech. rap.). Asia Pacific University of Technology en Innovation.

Brendan Burns, K. H., Joe Beda. (2019, april 10). *Kubernetes Up Running - Dive into the Future of Infrastructure* (O'Reilly, Red.). https://books.google.be/books?hl=en&lr=&id=-5izDwAAQBAJ&oi=fnd&pg=PP1&dq=kubernetes&ots=IsB4lxkeeI&sig=RBWC7ndt9aEs0bUHZ1usfNSGgQw&redir_esc=y#v=onepage&q&f=false

Cedric, D. (2019). *Kubernetes Educatieve Mogelijkheden En Opstellingen*. HoGent.

Cirelly, J. (2020, september 20). *7 Best Docker Monitoring Tools*. https://www.itprc.com/docker-monitoring-tools/

Cole, A. (2016, september 29). *The Importance of Container Monitoring*. https://www.stratoscale.com/blog/importance-container-monitoring/

Cristian Ruiz, L. N., Emmanuel Jeanvoine. (2015, augustus 24). *Performance Evaluation of Containers for HPC* (tech. rap.).

Gilles, A. (2019, januari 1). *sidecar proxy* (TechTarget, Red.). Verkregen 7 mei 2021, van https://searchitoperations.techtarget.com/definition/sidecar-proxy

Jeff, D. (2009, november 1). *Server Virtualization Architectureand Implementation* (tech. rap.).

Journal, C. (2020, december 29). *How Docker and Kubernetes Work Together* (P. Belagatti, Red.). Verkregen 27 april 2021, van https://containerjournal.com/topics/container-ecosystems/how-docker-and-kubernetes-work-together/

Kedia, P. (2013, mei 24). *A survey on Virtualization Service Providers, Security Issues, Tools and Future Trends* (tech. rap.). Amity University.

Kubernetes. (2021, februari 1). *What is Kubernetes?* (K. Bradshaw, Red.). https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/

Majors, C. (2016, april 10). *Why Honeycomb? Black Swans, Unknown-Unknowns, and the Glorious Future of Doom* (C. Majors, Red.). Verkregen 1 mei 2021, van https://www.honeycomb.io/blog/why-honeycomb-black-swans-unknown-unknowns-and-the-glorious-future-of-doom/

Matrix. (2016, september 28). *The Importance of Infrastructure Monitoring* (Matrix, Red.). Verkregen 7 mei 2021, van https://matrix-ibs.com/2016/09/28/importance-infrastructure-monitoring/

Oracle. (2012, januari 1). *Introduction to Virtualization* (Oracle, Red.). https://docs.oracle.com/cd/E26996_01/E18549/html/VMUSG1011.html

Radhwan, A. Y. (2013, maart 1). *Survey of Server Virtualization* (tech. rap.). Mosul University.

Rancher. (2020, januari 1). *A guide to Kubernetes with Rancher* (tech. rap.). Rancher.

Ribenzaft, R. (2020, maart 9). *Top Six Open Source Tools for Monitoring Kubernetes and Docker*. https://devops.com/top-six-open-source-tools-for-monitoring-kubernetes-and-docker/

Singh, A. (2004, januari 1). *An Introduction to Virtualization* (A. Singh, Red.). https://webx.ubi.pt/~hgil/utils/Virtualization_Introduction.html

Sissons, G. (2021, januari 26). *Comparing 10 Docker Container Monitoring Solutions for Rancher* (G. Sissons, Red.). Verkregen 8 mei 2021, van https://rancher.com/comparing-10-container-monitoring-solutions-rancher/

TechTarget. (2020, mei 1). *IT monitoring* (A. Gilles, Red.). Verkregen 29 april 2021, van https://searchitoperations.techtarget.com/definition/IT-monitoring

Thanh, B. (2015, januari 13). *Analysis of Docker Security* (tech. rap.). Aalto University School of Science.

Van Vleck, T. (2015, maart 16). *The IMB 7094 and CTSS* (T. Van Vleck, Red.). https://multicians.org/thvv/7094.html

Williams, A. (2016, september 19). *Monitoring Management With Docker Containers*.