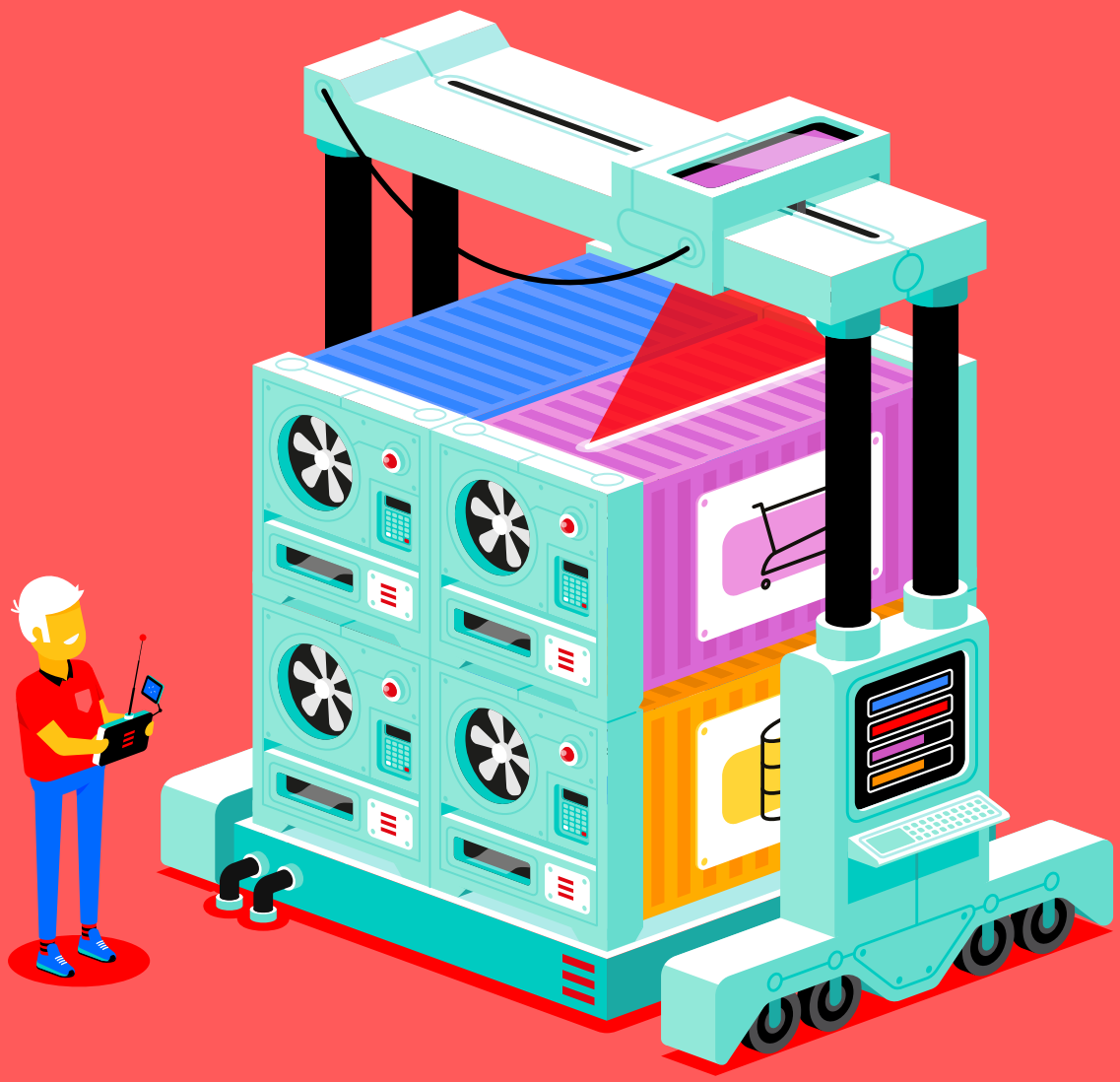
E-BOOK

Wat je kunt leren van de Kubernetes-pioniers die je voorgingen



DELL EMC | intel

Wil je meer informatie over de Kubernetes-oplossingen van True? 0800 BEL TRUE of ga naar true.nl/kubernetes

TRUE

E-BOOK

Inleiding

Kubernetes is voor veel bedrijven een reis naar het onbekende. Het is nieuwe technologie die om nieuwe werkwijzen en inzichten vraagt. Tools uit het ecosysteem helpen je op weg bij de adoptie van een containerized omgeving, maar een ongeluk zit in een klein hoekje en kan grote gevolgen hebben.

Een gevolg van Kubernetes is dat één probleem vaak uitmondt in meerdere problemen. Je moet als het ware een octopus zijn die alle onderdelen tegelijkertijd in de gaten houdt.

Het mooie van de Kubernetes-community is dat het een open community is. Open in de open-source tools die men omarmt en open in de openheid die men naar elkaar geeft over de reis die iedereen maakt. Meetups en conferences schieten als paddenstoelen uit de grond, waar developers hun geleerde lessen delen met collega-developers.

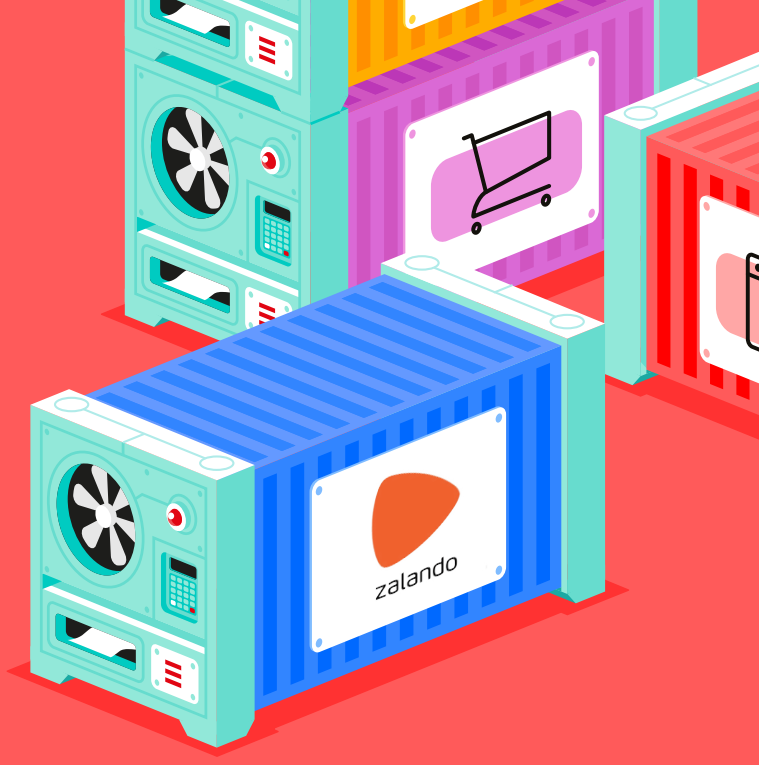
Vanuit True willen we hier ook een steentje aan bijdragen, daarom hebben we zes openbare Kubernetes-cases gebundeld. Je leest cases van grote bedrijven (Zalando, NU.nl), maar ook van kleinere bedrijven (Turnitin, NRE Labs). Allen te bestempelen als pioniers waar je van kunt leren.

Veel leesplezier!

Inhoudsopgave

1 Zalando

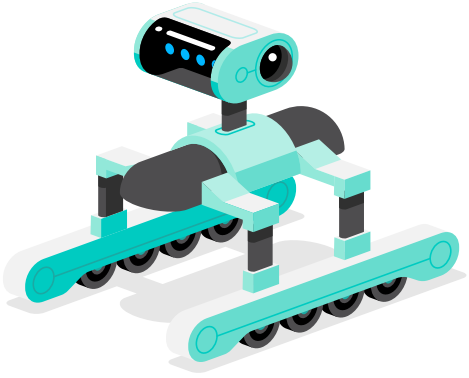
Zalando is met ruim 300.000 producten en 200 miljoen webbezoekers per maand een van de grootste webshops in Europa en is sinds 2008 exponentieel gegroeid.



De situatie

In 2015 ontstond het idee om de oorspronkelijke website te vernieuwen met nieuwe producten en diensten. Deze vernieuwing ging gepaard met een radicaal andere manier van werken, waarbij vooral gestuurd werd op zelforganiserende teams binnen de techafdeling van Zalando. Het betekende onder andere dat techneuten aan de slag moesten met een nieuwe infrastructuur die mee kon schalen met de snelle groei van de organisatie.

Een overstap naar Amazon Web Services (AWS) gaf Zalando het startschot om te experimenteren met containers en Kubernetes.



Wat ging er mis?

Op maandag 7 januari 2019 werden alle product- en outfitpagina's op het Zalando platform getroffen door een hele hoop errors. De oorsprong van het probleem bleek een DNS-storing die door een reeks andere problemen in Kubernetes werd veroorzaakt.

Het incident vond plaats toen een K8s-service in de aggregratie-laag een time-out kreeg. De aggregatie-laag stuurde 404-errors door en de klanten bleven pagina's verversen. Met als gevolg dat er dat er een verhoging kwam in het aantal verzoeken naar de aggregratielaag.

Tegelijkertijd was er een soortgelijke verhoging in het aantal DNS-queries dat werd uitgevoerd op de cluster DNS-infrastructuur (de zogenoemde CoreDNS). Deze verhoging zorgde ervoor dat er te veel geheugen binnen de CoreDNS-pods werd gebruikt, waardoor deze zonder geheugen kwamen te zitten en constant gekilld werden (OOM killing).

De CoreDNS kon niet herstellen van de initiële OOM-killing: de memory bleef als het ware in een cyclus opgaan en werd continu gekilld. Dit leidde uiteindelijk tot een totale DNS-outage voor het volledige cluster. Als bijwerking gingen alle interne monitoringtools inclusief alle alerts en push-metrics plat, waardoor de verantwoordelijke Kubernetes-engineers niet tijdig in konden grijpen. Hierdoor duurde het herstel langer.

Wat merkten klanten ervan?

Door de DNS-storing kwam de hele DNS-infrastructuur van Zalando plat te liggen. Klanten konden de betreffende pagina's ruim één uur lang niet bezoeken. Hierdoor kregen klanten te maken met onbereikbare pagina's (404's).

Wat kun je hiervan leren?

Zalando geeft in een GitHub-post aan dat er meerdere factoren meespeelden in de DNS-uitval. Ze delen daarnaast een aantal lessen:

1. Zorg ervoor dat caching aanstaat. Dat geldt met name voor processen die snel meerdere keren kunnen afvuren (zoals DNS-lookups). Zo verminder je de load op kwetsbaardere elementen in het platform.

2. Maak de monitoring-stack onafhankelijk van je OTAP-stack. Een failure in één stack betekent dan niet automatisch een failure in je andere stack.

3. Isoleer de 'blast radius' van je applicatie. In dit specifieke geval was het slechts één Node.js applicatie die een complete uitval veroorzaakte. De 'blast radius' van de applicatie zou je dus moeten isoleren naar een enkele node binnen het cluster, idealiter in een enkele pod.

4. Houd je monitoring-tools veerkrachtig. Dat was bij Zalando niet het geval, waardoor engineers niet tijdig konden ingrijpen. Om die reden is er een project in het leven geroepen met tools en chatbots om de responstijd voor incidenten te verkorten.

2 Target

Met ruim 1.800 winkels is Target een van de grotere retailers van Amerika. Ook digitaal maakt Target flinke stappen.

De situatie

Target runt een heterogene infrastructuur binnen een eigen datacenter en gebruikt diverse hosting-platformen om workloads neer te zetten. Om dit allemaal goed te kunnen managen is de Target Application Framework (TAP) ontwikkeld: een interface voor het runnen en managen van workloads tussen de verschillende platformen.

Elke applicatie die te maken heeft met de belangrijkste processen binnen Target wordt geïmplementeerd via TAP, waardoor developers geen kennis nodig hebben over de infrastructuur. Zij kunnen zich richten op het ontwikkelen en beheren van applicaties.

Daarnaast maakt Target gebruik van Kafka - een messaging service die informeert over gebeurtenissen tussen systemen. Voor infrastructuur checkt Kafka bijvoorbeeld logfiles en metrics over de back-end systemen. Het voordeel hiervan is dat alle applicaties direct geïnstalleerd worden met een aantal ingebouwde monitoring-agents.

Wat ging er mis?

Kafka runt in een OpenStack-infrastructuur en op een avond werd er een upgrade uitgevoerd op een onderliggend subsysteem van het netwerk. Deze upgrade zou betekenen dat er slechts tijdelijk een disruptie plaats zou vinden op het netwerk. Maar omdat er een issue plaatsvond ten tijde van de upgrade, was het netwerk uiteindelijk een paar uur offline.

Omdat er ook veel Kubernetes-clusters onder TAP in combinatie met Kafka worden gerund, ontstonden daar ook problemen. Een van de grotere clusters bestond uit ongeveer 2.000 ontwikkelomgevingen en die kwam in de knoei.

Onder normale omstandigheden gebruikte dit cluster minimale CPU, maar omdat er problemen waren met de connectiviteit van Kafka werden de 'logging sidecars' van de Kubernetes-clusters als het ware 'gewekt'. Hierdoor werd er een hogere load op de Docker-nodes gezet in het Kubernetes-cluster. De Kubernetes-scheduler probeerde

vervolgens de nodes te herstellen door workloads van de geaffecteerde nodes te verschuiven naar nodes die nog wel gezond waren. Hier ontstond echter een probleem: de ongezonde nodes verschoven naar gezonde nodes, maar de Kubernetes-scheduler hield geen rekening met het balanceren van workloads in het cluster, waardoor sommige nodes meer aanspraak maakten op de CPU dan andere nodes. Met als gevolg dat ongezonde naar gezonde nodes werden gebracht, maar dat ook enkele gezonde nodes ongezond werden omdat ze de load simpelweg niet aankonden. Dit werd op een gegeven moment een cyclus.

Tevens waren er gevolgen voor Consul, een deployment-agent die ervoor zorgt dat applicaties vanuit verschillende infrastructuren met elkaar corresponderen. De agent zorgt ervoor dat nieuwe workloads geregistreerd worden als service. Omdat er tijdens het reschedulen ruim 41.000 nodes werden opgespind en gekilld, begon Consul dit te zien als nieuwe workloads. Dit resulteerde in een verhoogd CPU-gebruik binnen Consul en een vertraging in laadsnelheid. Ook zorgde het voor een verhoging van het totale CPU-gebruik. Al met al werden de problemen opgestapeld door verschillende diensten die elkaar wisten te vergiftigen.

Na uitgebreid testen werden er diverse oplossingen gevonden en uitgevoerd. Zo vonden er onder andere upgrades plaats in de Consul-agent.

Wat merkten klanten ervan?

De klanten merkten hier in feite niets van, omdat de TAP productie-omgeving niet werd getroffen. In een andere productie-omgeving werd de mesh ook gepoisoned, maar de fouten vonden daar ook niet plaats omdat het een kleinere omgeving was.

Wat kun je hiervan leren?

1. “Smaller clusters, more of them” - De workloads in kleinere development Kubernetes-clusters werden niet zo hard getroffen als de grotere omgevingen.

2. Logfiles throttelen - Toen, vanwege netwerk-disruptie Kafka geen contact meer kon maken met andere systemen, vuurden duizenden containers allemaal logs af. Je zou deze logs kunnen throttelen, oftewel niet al te vaak achter elkaar afvuren.

3. Gedeelde Docker deamons is een kwetsbaar punt. Je moet harder werken om het getroffen gebied te herstellen ten opzichte van een individuele daemon. Ook dit gaat hand in hand met ‘Smaller clusters, more of them’.

4. Elke workload shippen met eigen logging en metrics sidecars kan slim zijn. Na het incident is Target ervan bewust dat het slim is om elke workload te shippen met eigen logging en metrics sidecars. Als dit een shared resource was met het cluster, was het incident waarschijnlijk moeilijker om te fixen.

3

NU.nl

NU.nl is een van de meest bezochte websites in Nederland. Naast de primaire nieuws-website ontwikkelt NU.nl een applicatie die zowel beschikbaar is voor iOS als Android.

De situatie

Sinds eind 2018 heeft het NU.nl team een aantal Kubernetes-clusters in productie draaien. Ze zagen de potentie van Kubernetes, met name in het verhogen van productiviteit van de developers en het verbeteren van bestaande CI/CD-pipelines. Het plan is om op termijn uiteindelijk grote workloads via Kubernetes te implementeren zodra ze genoeg kennis en vertrouwen in het platform hebben.

Wat ging er mis?

Het team had een paar problemen op een van de clusters. Het was geen megagroot probleem, het cluster bleef namelijk gewoon draaien, maar het had wel effect op de tools en dashboards die het NU.nl tech-team gebruikte om Kubernetes te monitoren. Door goed te kijken naar de oorzaak van de 'cascading failures' wisten ze het probleem op te lossen.

Het techteam van NU.nl ontving een rapportage van services met wat onbetrouwbare data zoals onverwachte error pages, hoge laadtijden en time-outs. Toen ze dit onderzochten in Grafana kwamen zij dit probleem ook tegen. Daarom werd er verder gezocht in de console, waar uit bleek dat enkele nodes de status 'NotReady' hadden gekregen.

Het zag ernaar uit dat het besturingssyteem van de node-processen aan het killen was voordat de 'kubelet' het geheugen kon terugeisen. Omdat de nodes in het cluster onderdeel zijn van een auto-scaling groep, werd ervan uitgegaan dat er perio-

diek een uitval plaatsvond, waardoor er gekozen werd om de NotReady-nodes één voor één te killen om te zien of de nieuwe nodes stabiel genoeg bleven. Dit was niet het geval, want bestaande nodes kregen hierna ook nog de status 'NotReady'.

De kern van het probleem zat in een Grafana set-up die zowel grafieken toonde voor pod-memory en CPU-gebruik. Het probleem zat in het component ElastAlert - een tool die alerts laat afgaan op basis van logfiles. Een van de alerts werd kort voor het incident afgevuurd, waarbij het ging om een alert die de 'cloudfront-*'indexes zou killen als ze niet op tijd nieuwe documenten genereerde. Omdat het ging om miljoenen verzoeken per uur, werd er heel veel geheugen verbruikt. Het gevolg: vertraging.

Na het vinden van de oorzaak zou het probleem snel gefixt zijn, zou je denken. Maar dat was niet het geval. Terwijl er namelijk werd gewerkt aan een fix, vond er een cascade of failures plaats. Oftwel: een opeenstapeling aan fouten.

Ze werkten aan diverse oplossingen. Zo werden alerts gedefinieerd in relatie tot resource-gebruik zoals CPU, geheugen en schijfruimte.

De oplossing was echter om Grafana buiten het cluster te plaatsen. Ook kwamen zij door het incident erachter dat nog niet iedereen binnen het tech-team van NU.nl dezelfde kennis had. Zo was niet iedereen op de hoogte van een wijziging binnen ElastAlert, waren er geen smoke tests gedaan na deploy en was de algemene kennis van het opereren van Kubernetes niet bij iedereen even groot.

Wat merkten klanten ervan?

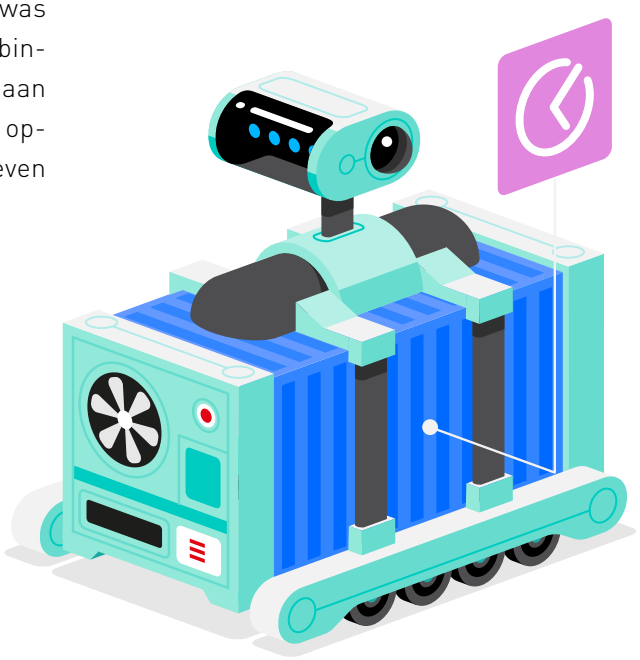
Omdat het om interne tools ging, merkten klanten hier niets van. Alleen het tech-team van NU.nl kreeg te maken met de disruptie.

Wat kun je hiervan leren?

1. Houdt iedereen's kennis up-to-date. Veel van de problemen hadden te maken met (een gebrek aan) kennis. NU.nl is van plan om deze fouten te door te vertalen naar een systeem voor SCRUM / Kanban zodat het meekan in hun bestaande werkwijze en ze ook de voortgang kunnen meten.

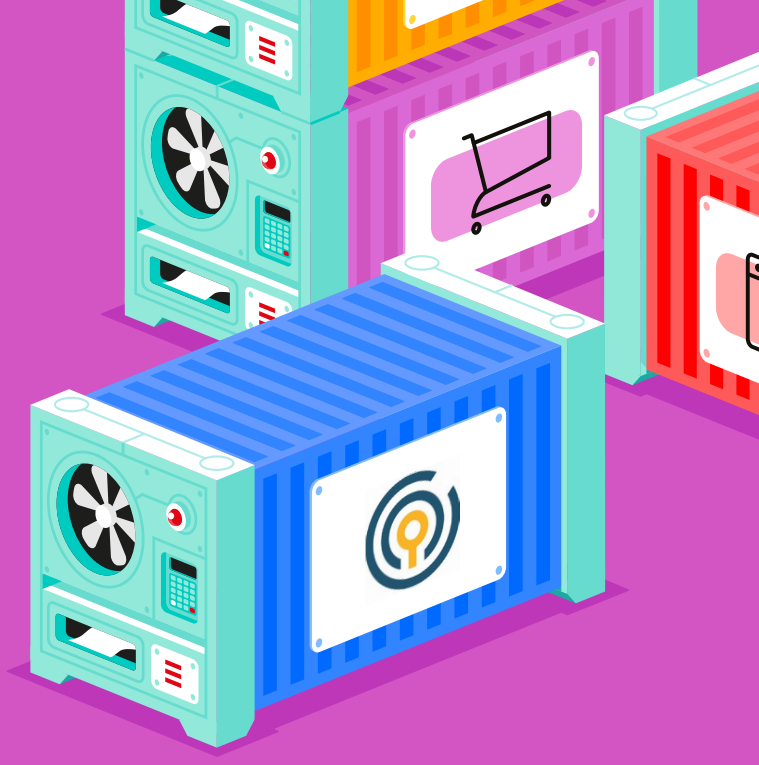
2. Zet monitoring en productie niet in hetzelfde cluster. Zodat de problemen bij de één niet zorgen voor problemen bij de ander. Dit komt de alerting en probleemoplossing achteraf niet ten goede.

3. Scheid ook development en productie-omgevingen. Zelfs bij zaken zoals logging, monitoring, visualisatie en messaging-workloads.



4 Civis

Civis is een platform voor dataweten-schappers waarmee zij een platform hebben voor het analyseren van data en het maken van datagedreven-besluiten.



De situatie

Een van de tools, Civis Analytics, draait al enige tijd containerized applicaties met behulp van Ku-bernetes. Deze analyse-tool heeft daarnaast ook diensten beschikbaar gemaakt aan hun klanten, waarmee zij zelf ook containerized applicaties kunnen deployen. Het cluster van Civis Analytics bestaat uit ongeveer 80 EC2 (Amazon Virtual Ma-chines) -instanties.

Wat ging er mis?

Toen Civis een nieuwe applicatie migreerde (Cro-quet) kwamen zij voor het eerst stabiliteitsproble-men tegen binnen het Kubernetes-cluster. Devel-opers rapporteerden een hoge latency voor kubectl-commando's en gebruikers rappor-teerden dat het moeizaam was om services te starten. Ook zagen de developers dat het cluster overladen werd met scripts. Door auto-scaling groeide het cluster uit naar 290 EC2 (AWS) -clus-ters, terwijl het verbruik nooit hoger was geweest dan 80 EC2-clusters. Men wist echter niet waar-om het cluster overladen werd met scripts.

De grote aantallen scripts hadden veel nieuwe nodes tot gevolg en deze hadden weer impact op de Master-node. De Master-node zorgt voor aans-turing van de onderliggende nodes, maar doet dit niet 'gratis'. Des te meer nodes, des te meer re-sources de Master-node aan moet spreken om goede prestaties te kunnen garanderen. Deze re-sources waren echter gelimiteerd én gedeeld met de 'generieke' nodes, waardoor er uiteindelijk minder resources beschikbaar waren voor de

generieke nodes omdat de Master-nodes alles claimden.

Door de Master-node meerdere resources te geven en er meerdere versies van in de lucht te houden, waren er weer genoeg resources om de andere nodes en pods succesvol aan te sturen. Ook ging de latency naar beneden. Dit is echter wel een 'brute force' manier en kan uiteindelijk alsnog voor issues zorgen. Verder zou Civis uiteindelijk weer tegen hetzelfde probleem aanlopen, zelfs met meer resources. Het probleem kon niet consistent gefixd worden door meer resources beschikbaar te stellen - men moest kijken naar de oorzaak van de exponentieel toenemende load. Civis besloot de kern van het probleem aan te pakken om de benodigde load op de Master-nodes te verlagen.

De oorzaak zat in de Datadog en fluentd 'DaemonSets'. Met weinig nodes zorgt dit niet voor issues, maar naarmate het aantal nodes stijgt, stijgt het aantal DaemonSets mee. Deze DaemonSets vragen de API-node regelmatig om updates en veroorzaken een zware load op de API-node. Met Datadog is er uiteindelijk een nieuwe 'Datadog Cluster Agent' service als buffer tussen alle Datadog DaemonSets en de API-node gezet. Nu werd de API-node nog maar door één machine aangesproken. Dit verhielp het probleem.

Wat merkten klanten ervan?

Klanten merkten vooral vertraging in de applicatie. Ook konden zij tijdelijk geen scripts uploaden.

Wat kun je hiervan leren?

1. Aansturen in Kubernetes is niet 'gratis'. Een Kubernetes-cluster met duizenden nodes en honderdduizenden pods kan dat niet zomaar aansturen zonder significante resources nodig te hebben voor die aansturing. Bij het toewijzen van resources moet er ook rekening gehouden worden met de resources die de operationele laag nodig heeft.

2. Scheid resources tussen Master-nodes en onderliggende nodes. Deze twee hebben een inverse relatie: des te meer onderliggende nodes die zelf resources claimen, des te meer resources de Master-node nodig heeft.

3. Monitor niet alleen op CPU en memory. Je moet ook monitoren op het aantal API-requests dat de API-server afvuurt, de latency van requests, en andere cruciale metrics. Prometheus kan hierbij helpen, want die monitort dit allemaal.

4. Doe tests voor het alloceren van resources. Vaar niet blind op de aanbevelingen van Kubernetes, maar doe ook je eigen tests om er zeker van te zijn dat de Master-nodes genoeg resources hebben om hun werk goed te kunnen doen.

5. Monitoring heeft impact op je resources. Elke request die elke agent maakt kan impact hebben op de gebruikte resources. En daar kunnen problemen door ontstaan. Wees daarvan bewust.

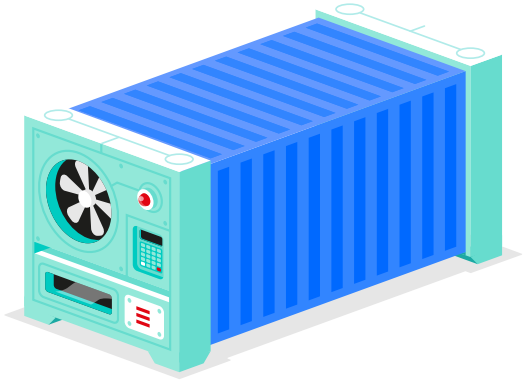
5 Turnitin

Turnitin is een tool voor het checken van plagiaat en het verbeteren van academische schrijfvaardigheden.

De situatie

Turnitin is begonnen met de adoptie van Kubernetes vanwege de groeiende behoefte om meer software en diensten binnen hun applicatie-omgeving te implementeren.

Turnitin heeft een serie van clusters in een AWS-omgeving, geprovisioned door kube-aws. Deze clusters worden gemonitord door een serie van health-checks die de gezondheid van de clusters in de gaten houden.



Wat ging er mis?

Om alle clusters in de gaten te houden, bevat Turnitin ook een grote hoeveelheid alarms om de gezondheid van de clusters te monitoren. Tijdens het opspinnen van elke cluster maken zij gebruik van een YAML-file die voor een private ELB voor de cluster-API beschikbaar stelt, maar ook andere componenten volledig kan draaien. Normaal gesproken ging dat goed, maar plotseling was er een uitval van ongeveer vijftien minuten. Van de 200 pods gingen er 100 offline.

Wat is er dan misgegaan? Ongeveer de helft van alles nodes/pods gaf een 'NotReady'-foutmelding af. De kubelet op deze nodes kon ook geen rapport afgeven. Voorafgaand aan de downtime zagen ze ontzettend veel netwerkverkeer op een van de nodes die op 'NotReady' stond.

Een van de dynamische IP's die aan de externe load balancer (ELB) gekoppeld was werd veranderd. Deze verandering in bereikbaarheid van de ELB werd niet doorgevoerd in de losse Kubelets. Hierdoor kon de load balancer geen

toegang verschaffen aan het cluster-API en rapporteerden de helft van de Kubelets dat ze 'NotReady' waren. De andere helft van deze kubelets kreeg hierdoor meer verkeer te verwerken, wat de spike in netwerkverkeer verklaarde.

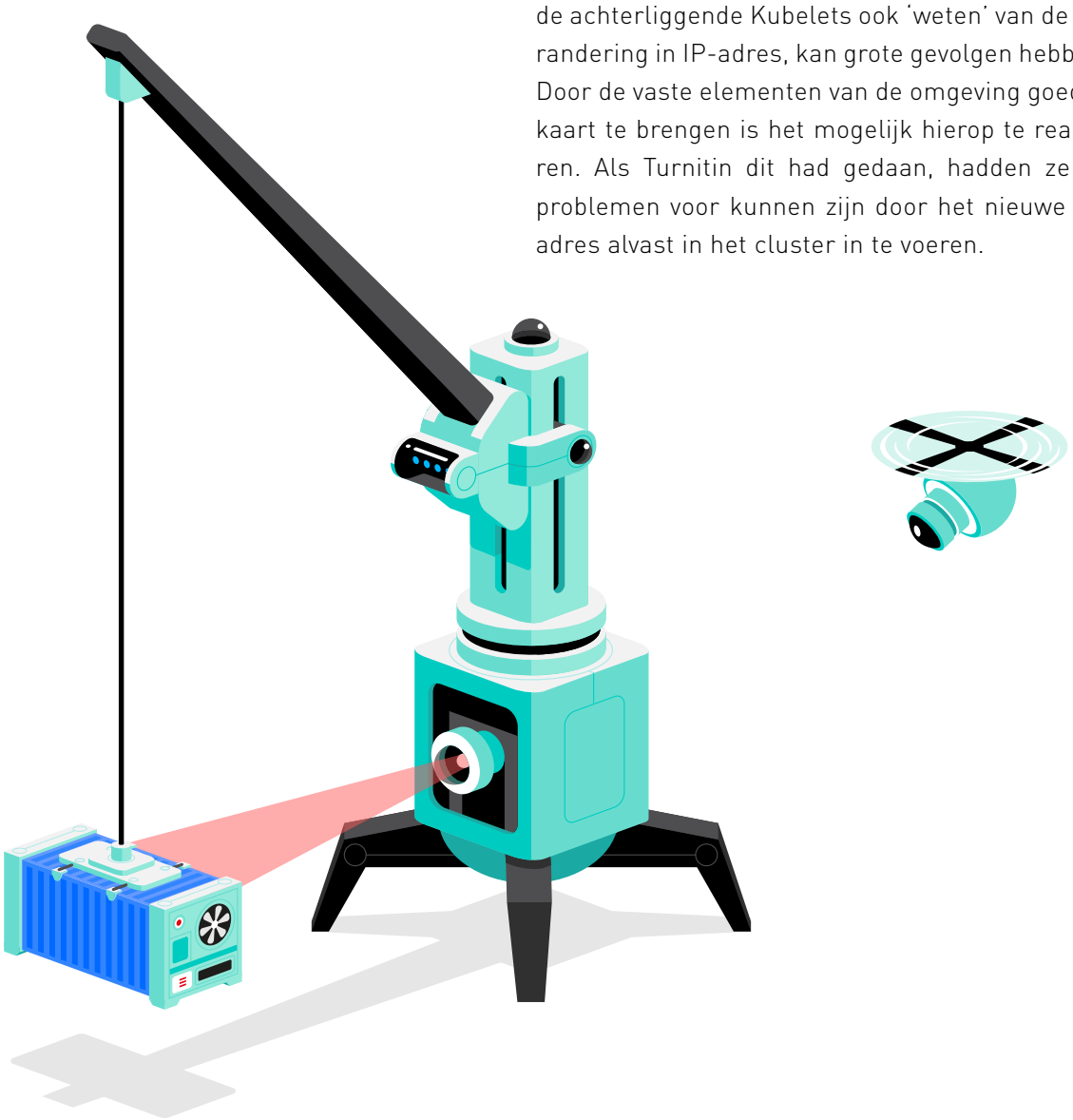
Wat merkten klanten ervan?

Onbekend, maar aangezien het een productieworkload betreft, kan je aannemen dat ongeveer de helft van de gebruikers er voor maximaal vijftien minuten last van heeft gehad.

Wat kun je hiervan leren?

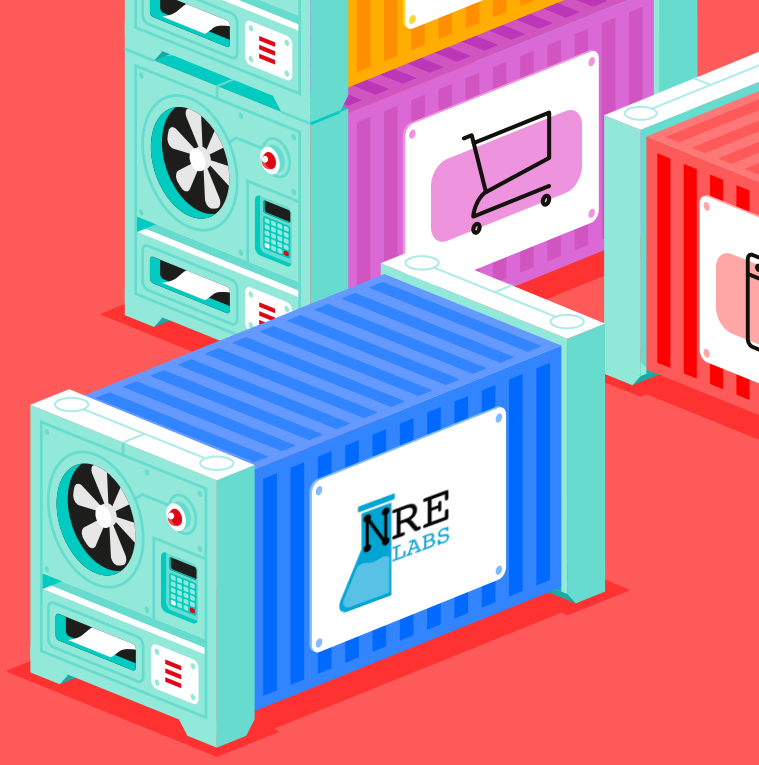
1. Werk met zo min mogelijk vaste variabelen in alle elementen van de omgeving. Een cluster die naar een externe load balancer zoekt op een vast IP-adres, kan in de problemen komen wanneer dit IP-adres niet meer bestaat. Wanneer je werkt met containers en de 'ephemeral state' die containers hebben, is het aan te raden om in alle onderdelen van de omgeving niet te rekenen op vaste waardes, maar op variabele waardes.

2. Pas niet zomaar je IP aan zonder rekening te houden met achterliggende Kubeletes. IP-veranderingen aan de externe load balancers kunnen een grote impact hebben op de gezondheid van het cluster. Hier aanpassingen in doen zonder dat de achterliggende Kubelets ook 'weten' van de verandering in IP-adres, kan grote gevolgen hebben. Door de vaste elementen van de omgeving goed in kaart te brengen is het mogelijk hierop te reageren. Als Turnitin dit had gedaan, hadden ze de problemen voor kunnen zijn door het nieuwe IP-adres alvast in het cluster in te voeren.



6 NRE Labs

NRE Lab is een community-centered initiatief om automatiseringsvaardigheden bij te brengen aan iedereen die daar maar interesse in heeft. Aan de hand van korte, simpele oefeningen vanuit de browser leer je over de tools, skills en processen die je nodig hebt om een Network Reliability Engineer te worden. De website is gemaakt op Antidote, een open-source project voor het vereenvoudigen van automatisering.



De situatie

NRE Labs bouwt hun Kubernetes-cluster met kubeadm, de CLI-tool die door velen als standaard wordt beschouwd. Met kubeadm kan NRE-labs een single-master cluster bouwen rechtstreeks vanuit de command line, inclusief alle master-services (ectd, API-server, scheduling, controller), en deze gelijk op een single node draaien. Deze master-services worden als statische pod geïmplementeerd in de 'single node'. Het voordeel is dat ze hiermee simpelweg een cluster kunnen opstarten, met het nadeel dat alles in één 'single node' draait. Dit is geen high-availability setup, maar is wel handig omdat NRE Labs nog in de startfase zit.

Wat ging er mis?

Op een ochtend kwam NRE Labs erachter dat er geen users bezig waren met een les en dat was gek; normaal gesproken waren er namelijk altijd wel wat mensen bezig met een online les. Het ging mis met de connectie van het cluster.

Om die reden besloten ze om de kubelet en pod opnieuw op te starten. Dat had even effect, maar niet voor lang, dus besloten ze het complete cluster weg te halen en opnieuw op te starten. Binnen Kubernetes is dit zo geregeld als je de juiste instellingen en tools voor automatisering hebt ingezet. Na het opstarten was het probleem nog niet opgelost.

Men kwam erachter dat het probleem zat in de 'ectd'-pod, een key/value database dat alle states van Kubernetes bevat. Deze pod was slechts twee minuten online voordat het crashte en opnieuw werd opgestart.

NRE Labs heeft als test gekozen om de health-checks uit te zetten. De pod werkte daarna als

vanouds en was niet kapot, dus het probleem zat uiteindelijk in de check die foutief was.

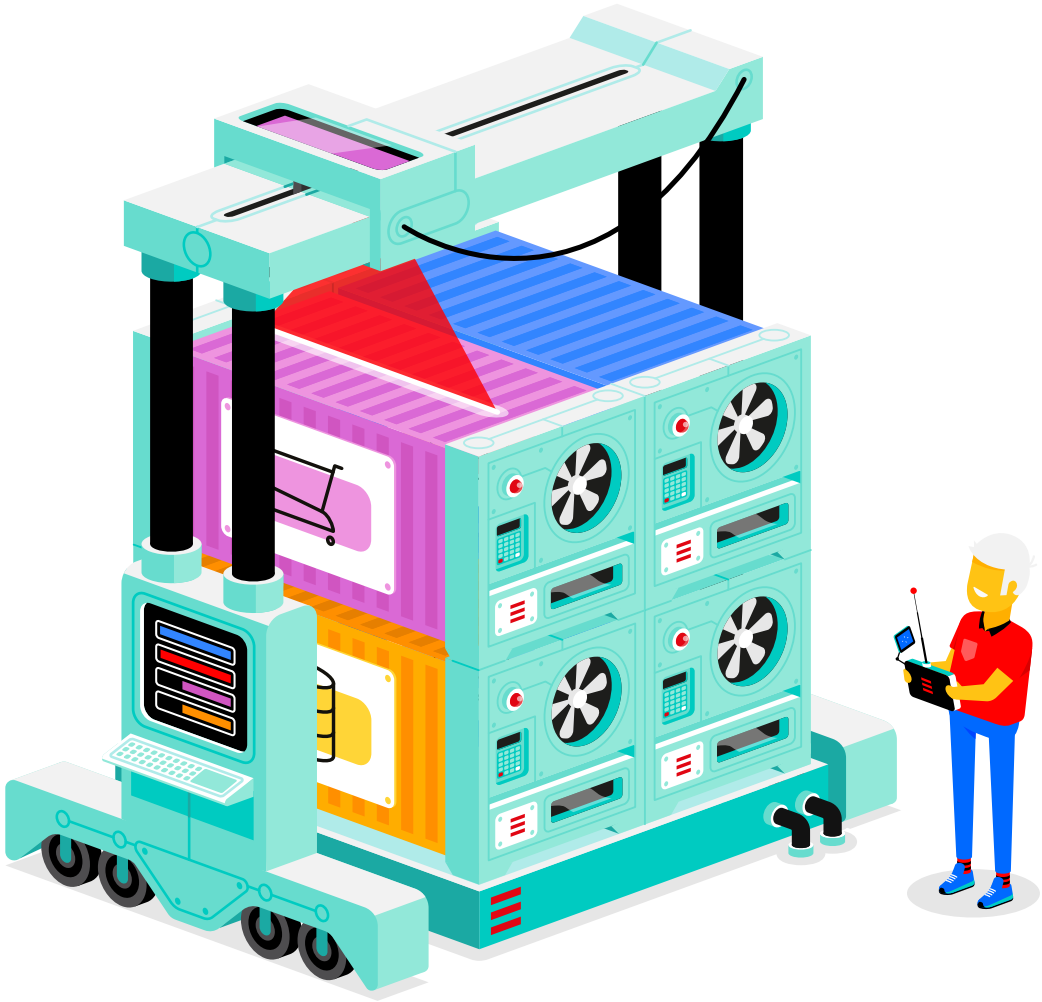
Wat merkten klanten ervan?

NRE Labs levert online lessen aan hun gebruikers om een network engineer te worden. Deze gebruikers konden een tijdje geen lessen starten, maar de exacte downtime is onbekend.

Wat kun je hiervan leren?

1. Problemen zijn snel op te lossen door infrastructuur te automatiseren. Ansible en Terraform zijn je vrienden binnen container-technologie. Maar let wel op, want door alles opnieuw op te starten, vernietig je ook logs en het 'bewijs' wat er eerst fout mee was. Kies er dus voor om logs altijd apart op te slaan, zodat deze niet verloren gaan wanneer een container besluit zichzelf te vernietigen en het systeem een nieuwe pod in het leven roept.

2. Monitoring redt levens, maar het probleem kan ook in de checks en rapportage zelf zitten. Vertrouw niet blind op de monitoring tool, maar doe ook altijd handmatige checks om de check te controleren.



7 Wat kun je leren?

- Voeg de productie-omgeving en monitoring-omgeving niet samen - want als de productie-omgeving de monitoring suite meetrekt in een failure, dan kan je niet meer rapporteren en troubleshooten.

- Monitoring-agents gebruiken ook resources, helemaal wanneer deze agents een fout spotten en elke minuut/seconde een waarschuwing gaan afvuren.

- Kubernetes-management is niet gratis - de Master-nodes hebben meer resources nodig naarmate het aantal onderliggende nodes stijgt. Houd hier met de toewijzing van infrastructuur rekening mee.

- Doe uitgebreide tests met verschillende schaalgroottes - enkele, tientallen en honderden containers.

- Documenteer je omgeving en de afhankelijkheden in je omgeving goed. Dit maakt het troubleshooten gemakkelijker en sneller.

- Maak duidelijke afspraken met andere IT-teams in je organisatie. Het offline brengen van applicaties of een onverhoopt langere downtime van het netwerk kan onbedoelde impact hebben op losstaande systemen.

- Werk niet met vaste waardes zoals fixed IP-adressen in een variabele, schaalbare omgeving.

- Goede orkestratie van containers vergt kennis over cloud-architectuur, monitoring, networking, databases, storage én applicaties. Dit geldt dubbel wanneer er een fout optreedt en men moet troubleshooten. Zorg er dus voor dat deze kennis in je team op meerdere plekken is geborgd en gedocumenteerd. Train developers in meerdere disciplines.

Geraadpleegde bronnen

→ Zalando: https://www.slideshare.net/try_except_/lets-talk-about-failures-with-kubernetes-hamburg-meetup

→ Target: https://medium.com/@daniel.p.woods/on-infrastructure-at-scale-a-cascading-failure-of-distributed-systems-7cff2a3cd2df

→ NU.nl: https://www.tibobeijen.nl/2019/02/01/learning-from-kubernetes-cluster-failure/

→ Civis: https://medium.com/civis-analytics/https-medium-com-civis-analytics-breaking-kubernetes-how-we-broke-and-fixed-our-k8s-cluster-adfa6fbade61

→ Turnitin: https://itnext.io/kubernetes-and-the-menace-elb-the-tale-of-an-outage-c00bef678fc0

→ NRE Labs: https://medium.com/civis-analytics/https-medium-com-civis-analytics-breaking-kubernetes-how-we-broke-and-fixed-our-k8s-cluster-adfa6fbade61

True bedankt de originele auteurs voor hun openheid in het publiekelijk delen van de problemen, oorzaken en oplossingen die ze hebben ervaren.

True & Kubernetes

Kubernetes biedt veel kansen voor developers. Je kunt een infrastructuur runnen zoals een van de grootste techbedrijven ter wereld (Google) en kunt je resources compacter en efficiënter inrichten. Dit levert niet alleen een kostenvoordeel op, maar vooral efficiëntie in ontwikkeltijd en diepgaande mogelijkheden voor CI/CD en een DevOps-werkwijze.

Veel cases uit deze longread illustreren hoe lastig het is om Kubernetes in de praktijk in te zetten. Heb je de tijd, expertise of ambitie niet om je eigen Kubernetes-cluster of -omgeving te beheren? True helpt je in de transitie naar Kubernetes.

Bij True heb je Kubernetes engineers achter de hand als je capaciteit tekortkomt. Onze Nederlandstalige experts implementeren en beheren containerized applicaties in een volledig geoptimaliseerde Managed Kubernetes-omgeving. Je hoeft zelf niets meer uit te zoeken: True doet dat volledig voor je. Heb je desondanks toch vragen, dan staat onze servicedesk 24/7 voor je klaar.

Ga voor meer informatie naar:
www.true.nl/kubernetes

Bezoekadres AMS
Keienbergweg 100
1101GH Amsterdam

Bezoekadres MST
Amerikalaan 1
6199AE Maastricht-Airport

0800 BEL TRUE
info@true.nl
www.true.nl