Postbus E0445
Keurenplein 41
1069CD Amsterdam
www.headease.nl
info@headease.nl

Paper: Basisprincipes applicatiebeheer

Datum: 19 mei 2021

Auteur(s): Remy Lamers - Head of product bij Headease, Joris Scharp - Team Lead Development bij Headease

Onderwerp: Basisprincipes applicatiebeheer

Introductie

Het doel van dit paper is om de door Headease gehanteerde beheersmogelijkheden te introduceren en toe te lichten, zodat organisaties deze principes kunnen meenemen in de overwegingen voor het inrichten van hun beheersstructuur en het zo eenvoudiger en goedkoper wordt om nieuwe toepassingen te beheren en ontwikkelen. Voorts kunnen deze principes functioneel beheer vereenvoudigen door geautomatiseerde rapportages en meldingsprocessen. De genoemde beheerprincipes zijn zowel toepasbaar op on-premise hosting als op cloud-hosting. Daarbij dient wel benoemd te worden dat hedendaagse cloud-hosting oplossingen in enkele gevallen tools aanbieden die het invullen van de beheersprincipes eenvoudiger maken. Voor on-premise hosting geldt dat deze principes over het algemeen ingevuld kunnen worden door (open-source) tools.

Organisaties zijn zeer afhankelijk van goed IT beheer, IT beheer is de ruggengraat van veel organisaties. Immers, zijn alle organisaties tegenwoordig afhankelijk van software en hardware. Door IT beheer goed in te richten, kunnen applicaties optimaal functioneren, is het mogelijk om vroegtijdig mogelijke problemen te identificeren en te verhelpen, en kan er efficiënt nieuwe software ontwikkeld en beschikbaar gesteld worden. Binnen IT beheer wordt er onderscheid gemaakt tussen applicatie-, technisch- en functioneel beheer. In dit paper richten wij ons vooral op moderne beheerprincipes die generiek toepasbaar zijn.
Onderdelen als ticketingsystemen - gebruikersbeheer- en autorisatiebeheer laten we in dit document bewust buiten beschouwing omdat deze onderdelen afhankelijk zijn van specifieke organisatie-voorkeuren.

Hedendaagse criteria aan applicaties zijn:

- Time-to-market: Het snel beschikbaar kunnen stellen van bedrijfskritische software, zodat de voordelen van de software direct ervaren kunnen worden.
- Goede klantervaring: Applicaties dienen doorlopend beschikbaar te zijn en uitstekende prestaties te bieden.
- Snelle en continue verbetering: Doorlopend toetsen en verbeteren van software matige processen, zodat de eindgebruikers een optimale gebruikservaring hebben.
- Schaalbare oplossingen: Systemen moeten snel op of af kunnen schalen op basis van het gebruik, zonder dat klanten hier problemen door ondervinden.
- Snelle oplossing van problemen: Het is uitermate belangrijk om snel te kunnen reageren wanneer er problemen optreden, of beter nog, voordat deze überhaupt optreden.
- Kostenefficiënt: Applicaties moeten kostenefficient ontwikkelt en beheerd kunnen worden.
- Security: Applicaties dienen te voldoen aan constant strenger wordende security eisen (AVG, NEN etc), uiteraard dienen security risico’s zo snel mogelijk geïdentificeerd en verholpen te worden.

Principes

Microservices

Microservices of een microservice-architectuur is een zeer populaire architectuurstijl voor het bouwen van onderhoudbare, robuuste, schaalbare, doelgerichte en onafhankelijk implementeerbare services. Een microservices architectuur bestaat uit een kleine verzameling van autonome en doelgerichte services, met elk hun eigen taak die gezamenlijk een reeks van taken uitvoeren. Het zorgt voor services die snel met veranderende wensen mee kunnen groeien en daardoor zeer goed onderhoudbaar zijn. Ook kan er op gedetailleerd niveau geschaald worden met services.

Containerization

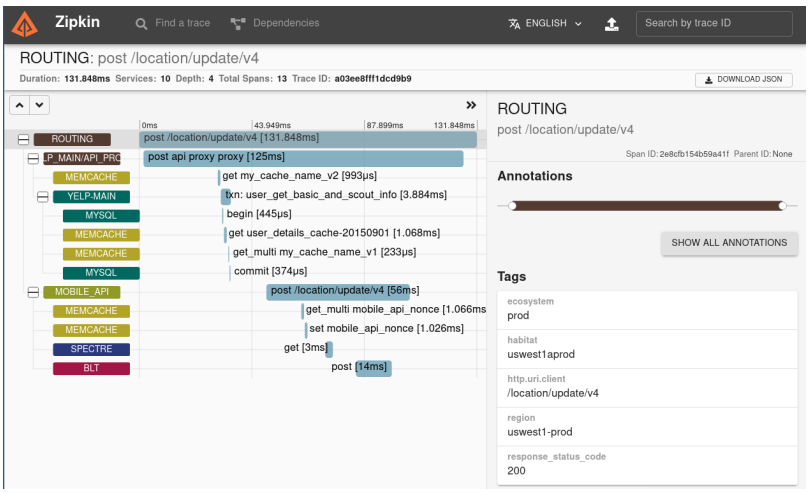
Containers bevatten een beknopte OS waar de te leveren software op werkt. Hierdoor werkt de te leveren software gegarandeerd op elk OS dat het type container ondersteunt. Headease werkt bijvoorbeeld met Docker containers.

Distributed logging

Een microservice architectuur kent veel applicaties die samenwerken. Het zou onhandig zijn om de logs van alle samenwerkende applicaties te moeten openen, de timestamps bij elkaar te zoeken om hier logica uit te halen. Distributed logging zorgt ervoor dat logs van alle applicaties binnen de architectuur op een centrale plek opgeslagen worden. Zo kunnen de logs van alle applicaties binnen een microservice architectuur makkelijk geaggregeerd worden en de complete flow van applicatielogs helder worden. Zo slaan wij alle logs op in ElasticSearch en kunnen we deze visueel maken middels Kibana.

Distributed tracing

Gedistribueerde tracering is een vereiste binnen een micro-service architectuur. Het helpt om vast te stellen waar in de micro-service architectuur storingen optreden en waar slechte prestaties worden veroorzaakt. Zo worden alle log statements die logischerwijs bij elkaar horen, vanuit alle verschillende services uit de distributed logging gegroepeerd. Ook kan deze informatie wordt gevisualiseerd in een overzicht, zodat over de hele chain van services traceerbaar is hoe de tijd over services verdeeld is.



Afbeelding 1: Zipkin - Distributed tracing

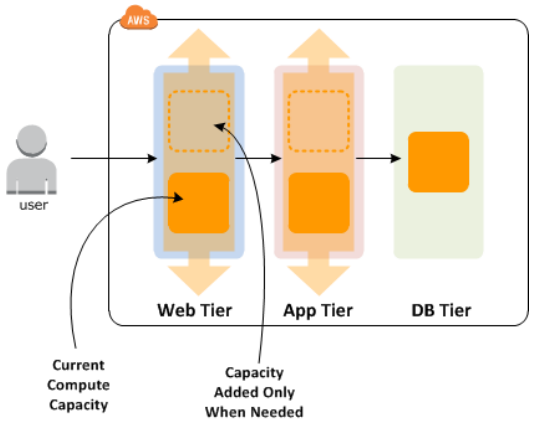
Auto scaling

Bij auto scaling of automatisch schalen worden de resources dynamisch toegewezen zodat deze overeenkomen met de prestatievereisten zoals beschreven in de SLA. Dit heeft grote voordelen omdat er zeer snel geacteerd kan worden naar gelang de behoefte, daarmee zorgt dit direct voor kostenefficiëntie.

- Als het gebruik toeneemt worden automatisch meerdere resources toegewezen
- Neemt het gebruik af en zijn de resources niet meer nodig? Dan kunnen de overbodige resources uitgeschakeld worden zodat de kosten passen bij het gebruik.
- Is er een instantie onbereikbaar? Dankzij auto-scaling wordt er automatisch een nieuwe instantie toegevoegd

Om auto scaling in te richten dient een proces vormgegeven te worden bestaande uit:

- Monitoring van de applicatie, service en infrastructuur. Vastleggen van CPU- en geheugengebruik, responstijden en queues.
- Logica waarmee besluitvorming gevoerd kan worden op basis van vooraf gedefinieerde thresholds, zodat er besloten kan worden om wel of niet- bij te schalen.
- Componenten die zorgdragen voor de schaling (in ons geval: Kubernetes)
- Teststrategie, controle en afstemming om te controleren of autoscale actie verloopt zoals verwacht.



Afbeelding 2: Auto-scaling

Security automation

Het automatiseren van security controles verkleint de kans op inbreuken en bijkomende problemen & kosten wanneer een security breuk zich voordoet. Handmatige processen zijn foutgevoelig en tijdsintensief, waardoor ze de identificatie van bedreigingen en verhelpen van bedreigingen kunnen vertragen.

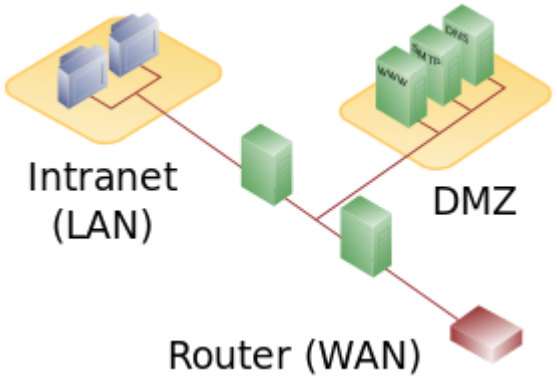
Automatisering van deze processen zorgt ervoor dat bedreigingen sneller geïdentificeerd, gevalideerd en geëscaleerd kunnen worden zonder handmatige interventie.

Denk hierbij aan:

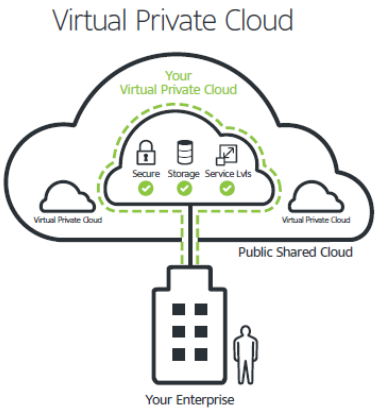
1. het automatisch scannen van de dependencies op security vulnerabilities;
2. het automatisch scannen van de OS libraries
3. het uitvoeren van een Dynamic Application Security Test (DAST) op elke code change. Een DAST voert geautomatiseerd een aanval uit op de applicatie.

Demilitarized zone (DMZ) / Virtual Private Cloud (VPC)

Een DMZ (on-premise) en VPC (cloud) hebben als doel om diverse componenten af te schermen van het internet. Zo is het bijvoorbeeld nooit de bedoeling om een database rechtstreeks beschikbaar te stellen naar iedereen op het internet. Een DMZ en VPC zorgen er voor dat deze enkel te bereiken is vanaf een “intern” netwerk.



Afbeelding 3: DMZ



Afbeelding 4: VPC

Automated monitoring

De inrichting van automatische monitoring is cruciaal voor de naleving van SLA’s. Het is van uiterst belang om te weten of een applicatie beschikbaar is en voldoet aan de eisen gesteld in de SLA.
Monitoring kan op verschillende niveaus ingericht worden, hierbij dienen geen compromissen gemaakt te worden. De voordelen van automatisch monitoring: Betrouwbare dienstverlening met betere gebruikerservaring, het sneller achterhalen of identificeren van problemen zodat deze op voorhand verholpen kunnen worden.

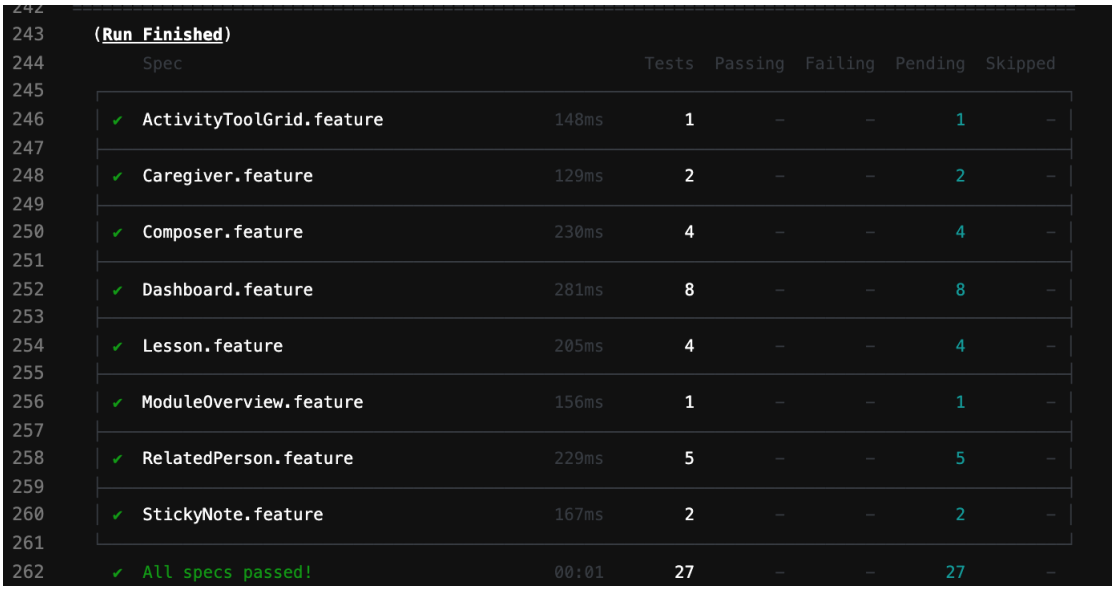
Bij het hanteren van een microservices architectuur, zullen zowel de microservices als het geheel aan services gemonitord moeten worden. Hierbij is een groot voordeel van de microservices architectuur, en monitoring per service, dat op individueel serviceniveau problemen geïdentificeerd kunnen worden. Door automatisch rapportages te genereren en informatie naar monitoring dashboards te publiceren, kan een organisatie real-time geïnformeerd worden. Daarnaast is het mogelijk om berichtgeving naar verschillende kanalen (email/SMS) in te richten, zodat belangrijke stakeholders snel en automatisch geïnformeerd kunnen worden indien bepaalde thresholds bereikt worden. Zo wordt de organisatie in staat gesteld om tot een snelle gerichte oplossing te komen.

Continuous integration (CI) & Continuous delivery (CD)

CI/CD wordt in de software-ontwikkeling als een manier gehanteerd om de afhandeling van code wijzigingen middels automatische processen vorm te geven. Denk hierbij aan het automatisch bouwen van wijzigingen, het automatisch testen en doorzetten naar omgevingen. Dankzij CI/CD kan veel tijd bespaard worden aan handmatige en complexe processen. CI/CD zorgt ervoor dat ontwikkelaars sneller en gemakkelijker wijzigingen in code kunnen doorvoeren, en dankzij de automatische testen is dit ook betrouwbaarder.

Automatische tests:

- Unit tests
- Functionele tests (F-test)
- Performance tests
- Security tests
- Integratie tests



Afbeelding 5: Overzicht succesvolle automatische functionele test

Continuous deployment

Dankzij continuous deployment wordt het build pakket en configuratie (de uitkomst van continuous delivery) automatische naar de verschillende omgeving doorgezet. Ook hierbij geldt weer dat handmatige (en daarmee foutgevoelige en tijd kostbare processen) niet langer noodzakelijk zijn. Continuous deployment zien wij als een streven, het is vereist dat alle overige principes zeer zorgvuldig zijn ingericht.

Samenvattend

Om beheer efficiënt in te richten is het voorwaardelijk dat de huidige knelpunten en behoeftes in beheer geïdentificeerd worden. Door de beheerorganisatie al te betrekken in de ontwikkelingsfase, kunnen knelpunten en behoeftes al vroeg in samenwerking met ontwikkelaars ingevuld worden, zo ontstaat er een optimale inrichting van IT beheer. Beheer is namelijk een cruciaal onderdeel in de dagelijkse processen van organisaties en het belang van goed beheer mag absoluut niet onderschat worden.

Moderne technieken en principes maken het mogelijk om beheer en software ontwikkeling efficiënt en kostenbesparend in te vullen.

Om software adequaat te kunnen beheren en ontwikkelen te kunnen plegen is het cruciaal om over real-time Het automatiseren van beheerscomponenten voorkomt foutgevoelige, handmatige en tijdrovende processen, zorgt voor kostenbesparing en uiteindelijke een betere gebruikerservaring. Overwegingen die organisaties zullen moeten maken zijn of zij gebruik maken van (open-source) tools en dedicated servers, of mogelijkheden die hedendaagse cloud-hosting mogelijk maken.