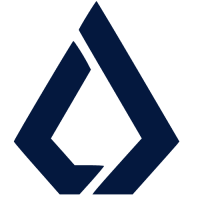
Case study: Serverless

Seven step approach



Project: Lisk Delegates

Project team: BBB-BC

Team: Joeri Berman (2834499)

Luc Urlings (3071081)

Stijn Baltessen (3064565)

Wouter Vandenboorn (3099571)

Floris Feddema (3188256)

Davy de Haas (3099695)

Opdrachtgever: Case study, Bartosz Paszkowski

Versie: 0.1

Versie datum: 21 april 2020

Status: Concept

# Document revisies

|  |  |  |  |
| --- | --- | --- | --- |
| **Versie** | **Wijzigen** | **Auteur** | **Datum** |
| 0.1 | Opzet Document | Stijn Baltessen | 21-04-2020 |

# Inhoudsopgave

[**Document revisies**](#_bu2l4jqkimrr) **2**

[**Inhoudsopgave**](#_fi4lxnndl53b) **3**

[**Introductie**](#_eko9gepalqi) **4**

[**Beschrijving case**](#_jyhc082m4ryc) **5**

[**Seven step approach**](#_djfvx4apkl6h) **6**

[**1. Bespreek de casus**](#_vpuo84y31t3k) **7**

[**2. Identificeer de vragen**](#_n9zlm068dlrb) **8**

[**3. Brainstorm**](#_80ry44jx0yvq) **9**

[**4. Analyseer**](#_u1vxzcd3o77b) **10**

[**5. Formuleer de leerdoelen**](#_agufbfdlk758) **11**

[**6. Individuele studie**](#_5zb5qcvdcaud) **12**

[**7. Discussieer de bevindingen**](#_9wt8awkaq71f) **13**

# Introductie

Om de case study serverless op te lossen gaan we de seven step approach toepassen, in dit document zal elke stap worden uitgelegd. Uiteindelijke conclusie wordt op het einde opgeschreven.

Eerst wordt de case study uitgelegd en hierna de zeven stappen die genomen worden.

# Beschrijving case

Ze weet niet welke keuze van cloud provider

# Seven step approach

Hier staan de zeven stappen van de [seven step approach](https://portal.fhict.nl/Studentenplein/LMC/1920vj/Software%20engineering/Afstudeerfase/EnterpriseSoftware/CaseStudies/SevenStepsApproach.docx) methode die bij probleem gebaseerd leren voorkomt.

1. Bespreek de casus

2. Identificeer de vragen

3. Brainstorm over potentiële bekende oplossingen

4. Analyseer resultaat van brainstorm sessie

5. Formuleer de leerdoelen en wat er nog geleerd moet worden

6. Individuele studie om de kennis van 5 te kunnen vergaren

7. Discussieer de bevindingen van 6 en hoe dit 5 en 2 beantwoord

# 1. Bespreek de casus

Ze weet niet welke keuze cloud provider ze willen om hun rest-service to validate IBANs te hosten. Motiveer waarom een service provider de beste optie is.

Let op de punten: lage kosten en able to handle expected load.

Geef een schatting van jaarlijkse kosten.

Probleemstelling: Welke cloud-based solution is juist voor het hosten van een rest-service waarmee je een valideer algoritme kan uitvoeren. En dit makkelijk kan op- en af schalen, zodat de kosten optimaal zijn. Deze oplossing moet voldoen aan:

* Low response times
* Cloud resource usage at a minimum
* All transaction have to be available at request
* Scale vertically and horizontally
* Schatting van kosten op jaarbasis gebaseerd op het verwachte gebruik.

# 2. Identificeer de vragen

* Wat houdt verticaal en horizontaal schalen in?
* Wat zijn de 12-factor app guidelines?
* Wat zijn guidelines en best practices waarmee rekening gehouden moet worden om de cloud services en applicaties onderhoudbaar te houden?
* Welke onderdelen van een applicatie kunnen worden verplaatst naar cloud services, is het wel handig/veilig om alle services op een cloud omgeving te hosten?
* Welke verschillende opties zijn er om een cloud services te implementeren?
* Wat zijn de kosten van de te implementeren cloud services, en zijn er manieren om hier kosten op te besparen door onderdelen juist wel of niet in de cloud te laten draaien?
* Hoe kan worden bepaald of opslag op een cloud service veilig is? Kunnen we bevestigen dat een cloud hosting bedrijf genoeg stappen heeft gezet om veiligheid van data te garanderen?

# 3 + 4. Brainstorm & Analyseer

Wat houdt verticaal en horizontaal schalen in?

* Verticaal is het upgraden van hardware op een bestaande instantie. Horizontaal is meerdere instanties van een applicatie draaien om de load te verdelen.

Wat zijn de 12-factor app guidelines?

* Geen voorkennis.

Wat zijn guidelines en best practices waarmee rekening gehouden moet worden om de cloud services en applicaties onderhoudbaar te houden?

* Geen voorkennis.

Welke onderdelen van een applicatie kunnen worden verplaatst naar cloud services, is het wel handig/veilig om alle services op een cloud omgeving te hosten?

* REST api’s en databases kunnen allemaal naar cloud services worden verplaatst, vrijwel alles zolang het juist is ingericht.
* Database in verband met veiligheid en privacy (gdpr) en dit soort zaken kunnen wat ingewikkelder liggen en heeft dus wat onderzoek nodig.

Welke verschillende opties zijn er om een REST api op een cloud service te deployen?

* Denk aan vm, serverless, containers, web-app, managed clusters (docker swarm, kubernetes)

Wat zijn de kosten van de te implementeren cloud services, en zijn er manieren om hier kosten op te besparen door onderdelen juist wel of niet in de cloud te laten draaien?

* Hoge kosten, verder geen voorkennis.
* Vergelijking maakt met machines kopen en hier geld mee terug te verdienen.
* Als je bestaande servers hebt kan je deze niet anders inrichten om deze scalable te maken en kosten te besparen.
* Er zijn veel wendingen die je kan nemen om kosten te besparen context van de casus is te weinig om hier advies op te geven direct maar we kunnen wel punten noemen waar je op kan letten.

Hoe kan worden bepaald of opslag op een cloud service veilig is? Kunnen we bevestigen dat een cloud hosting bedrijf genoeg stappen heeft gezet om veiligheid van data te garanderen?

* Database in verband met veiligheid en privacy (gdpr) en dit soort zaken kunnen wat ingewikkelder liggen en heeft dus wat onderzoek nodig.
* Cloud service transparantie kan je op letten van veiligheid.
* Verder geen voorkennis.

# 

# 5. Formuleer de leerdoelen

De student weet wat cloud services zijn.

De student weet welke grote cloud services er zijn.

De student weet welke onderdelen van een applicatie hij als cloud service kan hosten.

* De student weet wat de 12-factor guidelines zijn en waarom deze van belang is.

De student kan een redelijk schatting maken van hoe duur cloud services zijn.

De student weet de best-practices om een applicatie in een cloud-service onderhoudbaar op te zetten.

De student weet de voor en nadelen van de verschillende soorten cloud-services (vm, containers, web-app, clusters, **cloud functions**).

# 6. Individuele studie

## Wat houdt verticaal en horizontaal schalen in?

## scaling-image

Verticaal schalen is de hoeveelheid resources van de machine verhogen. Horizontaal schalen betekent dat er meer instanties van machines worden toegevoegd om op deze manier meer resources beschikbaar te krijgen.

Voordelen verticaal schalen:

* Het is heel eenvoudig
* Er is altijd meer performance
* De code hoeft er niet voor te veranderen

Nadelen verticaal schalen:

* Het is duur
* Kansen op downtime zijn hoog

Voordelen van horizontaal schalen

* Het is goedkoop
* Minder load per server
* Minder kans op downtime

Nadelen van horizontaal schalen:

* Eventual consistency is de manier om te zorgen dat alles up to date is
* Code moet soms worden aangepast

## Wat zijn guidelines en best practices waarmee rekening gehouden moet worden om de cloud services en applicaties onderhoudbaar te houden?

### MaaS

Binnen moderne Cloud services omgevingen hebben een heleboel sensoren en monitoring apparaten behorend bij het systeem. Deze sensoren verzamelen vervolgens data, deze data kan gebruikt worden voor verschillende doeleinden. Een hiervan is het automatisch schalen van systemen zodra dit nodig is, als er genoeg data verzameld is kan dit geautomatiseerd van te voren al worden gedaan, en er kan ook automatisch teruggeschaald worden zodra de piek weg is.

MaaS oftewel Maintenance as a Service is de benaming van de verschillende moderne sensoren en data die er voor zorgen dat het mogelijk is om de cloud omgevingen gedeeltelijk geautomatiseerd te onderhouden.

Onder MaaS vallen een enorm aantal aan verschillende services, welke services worden aangeboden verschilt per cloud hosting provider. Het kan dus zijn dat een provider een of meerdere van de onder genoemde services aanbiedt, maar het kan ook zijn dat ze niet beschreven services aanbieden aan de klant.

#### Fault detection as a Service

Deze service heeft als doel om de klant te informeren over mogelijk automatische gedetecteerde fouten. Vaak wordt deze detectie gebaseerd op de statistieken die gemeten worden door de End of life van producten te meten en door de Mean Time Between Failures bij te houden.

#### Recommendations as a Service

Deze service informeert de klant over mogelijk uit te voeren onderhoud voor specifieke onderdelen van een component. Deze recommandaties zijn gebaseerd op de Overall Equipment Efficiency in combinatie met kansberekening.

#### Maintenance operations management as a Service

Deze service is voornamelijk een vorm van visualisatie. Een klant krijgt een gevisualiseerd overzicht van de status van zijn services zoals deze zijn beschreven in het contract. Hierin kan ook gekeken worden naar meer specifieke onderdelen van een service. In grote lijnen is het dus een service die ervoor zorgt dat je een visualisatie krijgt van hoe goed er aan het contract gehouden wordt door de leverende partij.

#### Simulation as a Service

Deze service is bedoeld om toekomstige features en onderdelen te simuleren. De gegevens van voorgaande systemen wordt gebruikt om een inschatting te maken van deze simulaties. Met behulp van deze service kan er gebaseerd op bekende issues ervoor worden gezocht dat deze in toekomstige implementaties niet meer aanwezig zullen zijn. Mogelijke issues kunnen hierdoor ook eerder worden ontdekt en onderhouden worden.

#### Training as a Service

Training is een belangrijk onderdeel van onderhoudbaarheid. Werknemers met extensieve kennis over de gebruikte systemen kunnen het systeem beter onderhouden dan werknemers met minder kennis. In de artikelen wordt vaak gerefereerd aan VR en AR oplossingen geleverd door fabrikanten die ervoor kunnen zorgen dat de werknemers die het systeem moeten onderhouden beter getraind kunnen worden zonder dure kosten die het sturen van experts met zich mee zou brengen.

### Wat moet een klant zelf onderhouden van een cloud service

Een klant die services van een cloud hosting bedrijf afneemt hoeft natuurlijk geen van de bovenstaande onderhoudsmethodes te gebruiken aangezien de klant nooit toegang kan krijgen tot de fysieke machines. Natuurlijk zijn er wel een aantal onderdelen waar de provider geen onderhoud aan kan plegen en welke toch echt door de klant moeten worden onderhouden.

Een voorbeeld hiervan is het up to date houden van de geïnstalleerde en gebruikte software op de cloud omgeving. Een klant is natuurlijk zelf verantwoordelijk over wat er precies op de cloud machine gebeurt. Als hier software opgezet wordt door de klant is het dan ook aan hem om dit up to date te houden en te onderhouden.

OS software is een andere kwestie en de gebruikelijke werkwijze hiervan kan veranderen per provider. Als voorbeeld zullen we hier Microsoft's Azure platform gebruiken.

Azure rolt periodiek geautomatiseerde updates uit op verschillende onderdelen van het systeem. Dit wordt gedaan door Azure zelf en de klant hoeft hier verder geen rekening mee te houden. Ook heeft Azure nog meer services geïmplementeerd waardoor een klant zo min mogelijk impact ondervindt door het uitrollen van deze updates.

De meeste updates hebben al minimaal tot geen impact op vm’s en degene die dat wel hebben zijn vaak de updates waardoor een systeem voor zou moeten herstarten. Als een systeem moet herstarten wordt gekeken naar oplossingen met de minste impact. Bijvoorbeeld: Hetzelfde systeem draait in verschillende regio’s Deze worden dan een voor een herstart zodat de functionaliteiten altijd beschikbaar blijven, met mogelijk tijdelijk meer latency.

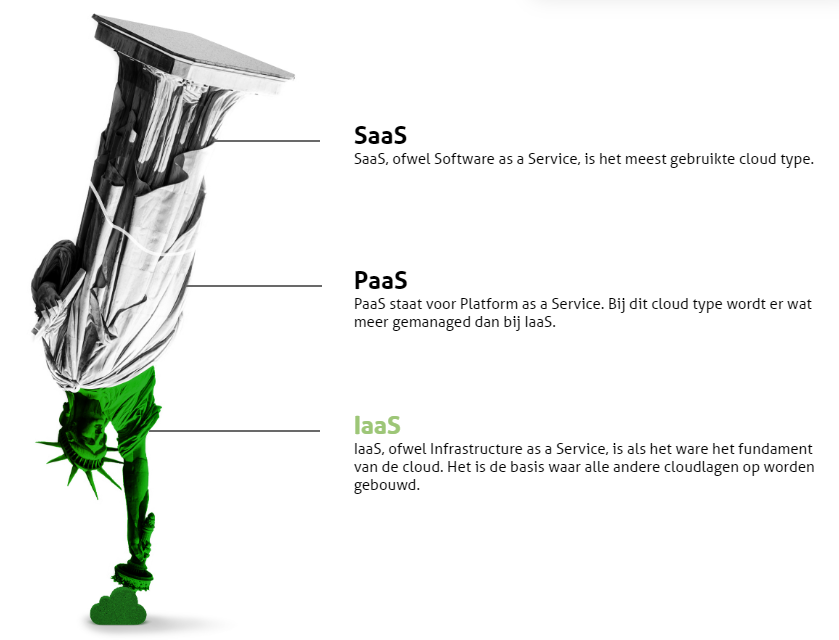
Als een systeem niet per se hoeft te herstarten voor een update kan het alsnog voorkomen dat deze lang kunnen duren of veel impact kunnen hebben op het systeem. Als een van deze updates kort duurt wordt de VM tijdelijk gepauzeerd totdat de update klaar is zodat er geen gegevensverlies kan voorkomen. Duurt een update langer? Dan kan er gekeken worden naar een live migratie. Hierbij wordt een kopie van de VM tijdelijk naar een andere locatie verplaatst en wordt de applicatie vanuit hier uitgevoerd. Dit geeft de originele VM de ruimte om de update uit te voeren zonder dat er impact is op de services en functionaliteiten die aangeboden worden.

## 

## Welke verschillende opties zijn er om een cloud service te implementeren?

Voordat er kan worden gekeken naar de implementatie van een cloud service moet er eerst worden bepaald welk cloud type kan worden gebruikt.

De cloud bestaat vaak uit meerdere lagen. Het begint bij de infrastructuur. Daarop wordt een platform gebouwd. Dit platform zorgt ervoor dat de applicaties gedraaid kunnen worden. Hierdoor ontstaan er drie verschillende lagen;



### 

### Infrastructure as a Service

Infrastructure as a Service (IaaS) is het fundament van de cloud. Het is de basis waar alle andere cloud lagen op worden gebouwd en is daarmee ook de minimale laag die te outsourcen valt. Het is vergelijkbaar met on-premises services die worden gebruikt, maar dan staat de gehele infrastructuur op de cloud.

Als gebruiker beheer je zelf het operating system, de databases en de applicaties. De leverancier is dan ook alleen verantwoordelijk voor het leveren van onder andere de hardware, databases en virtuele machines.

De voordelen hiervan zijn dat de klant zelf geen hardware hoeft aan te schaffen, te onderhouden en dat er gebruikgemaakt wordt van een abonnement. Hierdoor betaal je alleen waar je ook echt gebruik van maakt en zijn er geen grote investeringen nodig.

### Platform as a Service

Platform as a Service (PaaS) is een cloud type waarbij er meer wordt gemanaged dan bij IaaS en biedt een platform aan. Hier kan een gebruiker software installeren om SaaS diensten aan te bieden.

Bij PaaS hoef je niet zelf je virtuele machine in te richten, zoals bij IaaS wel het geval is. Er is dan ook meer verantwoordelijkheid bij de leverancier van dit cloud type en minder bij de klant zelf.

Deze leverancier moet dan ook niet alleen de hardware leveren, maar ook de virtualisatie laag en alle bijbehorende diensten (middleware als Java en .Net) managen. De inrichting van software wordt voor de klant geïnstalleerd en beheerd.

Binnen PaaS zijn er wel meer beheertaken voor de klant. De klant blijft zelf verantwoordelijk voor de data en het beheer en de ontwikkeling van de applicaties.

Het voordeel van PaaS is dat er snel een test, ontwikkel en deployment omgeving kan worden opgezet.

### Software as a Service

Software as a Service (SaaS) is de meest gebruikte cloud type. Dit komt omdat het veel gemak biedt voor de klant en deze alleen verantwoordelijk is voor het leveren van de applicatie of software die moet worden gedraaid.

De leverancier is verantwoordelijk voor alles eromheen, zoals het beheren van servers, netwerken, opslag en de applicatie. Dit betekent dat er geen gebruik gemaakt hoeft te worden van on premises resources en niets zichtbaar is van de achterliggende infrastructuur voor de klant.

### Implementatie

Een applicatie naar een cloud omgeving overzetten is niet makkelijk. Om een goede migratie, implementatie en onderhoud van de applicatie te hebben moet er dus naar een aantal punten worden gekeken. Deze punten zorgen ervoor dat er snel profijt uit het overzetten kan worden gehaald.

* Definieer het project - Sommige applicaties zouden nooit op de cloud mogen staan. Bedenk wat je wilt verplaatsen en of het wel of geen nut heeft.
* Selecteer het platform - Kies een platform dat snel, makkelijk en veilig is om te deployen. Zorg ervoor dat het ook flexibel is zodat het schalen hiervan gemakkelijk kan in de toekomst, als er meer vraag is.
* Begrijp het beleid over veiligheid - Veel service leveranciers geloven dat het veilig opslaan van data bij de klant zelf ligt. Maak daarom duidelijk dat er een goede verstandhouding is van wie verantwoordelijk is voor welke data en dat er de juiste middelen zijn om het te beveiligen.
* Selecteer de cloud service leverancier - Het is slim om met een leverancier in zee te gaan die succes heeft of heeft gehad met vergelijkbare bedrijven en hebben verstand van de technologieën die de klant gebruikt.
* Duidelijke service level agreement - Ben naast uptime duidelijk met de leverancier wanneer het komt op het service level agreement (SLA) waarin staat wat er wel en wat er niet aan service wordt geleverd, zoals data availability of data protection.
* Spreek af wie verantwoordelijk is voor recovery - Het is belangrijk om te weten of de leverancier assistentie biedt tijdens een uitval.
* Migratie in delen - Zorg ervoor dat de migratie geleidelijk aan wordt gedaan zodat er meer tijd is voor het aanpassen van problemen en er een minimaal risico is in een complete faal van de applicatie.
* Denk vooruit - De business eisen kunnen altijd veranderen, dus het kiezen van een cloud oplossing dat het mogelijk maakt om te kunnen switchen tussen een on-premise en cloud omgeving of tussen verschillende cloud service leveranciers. Het creëren en implementeren van een cloud strategie kost tijd, energie en moeite. Het is dus belangrijk om de juiste cloud strategie en bijbehorende cloud type te kiezen die je kunt helpen voor mogelijke nieuwe kansen binnen de markt, het groeien van de business en het verbeteren van klantbehoud.

#### Approach



## 

## Wat zijn de 12-factor app guidelines?

De 12-factor guidelines is een methodologie voor het maken van software-as-a-service. De 12-factor guidelines gaan voornamelijk over de best practices bij het bouwen van zo’n stuk software. Wanneer deze guidelines worden gevolgd zorgt het ervoor dat de software zonder grote veranderingen kan worden opgeschaald en dat er weinig grote verschillen zijn tussen development en productie, wat voor een soepelere development lifecycle zorgt. De twaalf factoren zijn:

### Codebase

Bouw bovenop één codebase, volledig gevolgd door een versiebeheersysteem (VCS). Implementaties moeten automatisch gebeuren, zodat alles zonder extra werk in verschillende omgevingen kan worden uitgevoerd. Er moet altijd één opslagplaats zijn voor een individuele toepassing om CI / CD-pijplijnen te vergemakkelijken.

### Dependencies

Kopieer geen dependencies naar de codebase, maar gebruik een package manager. Alleen juiste versies van dependencies moeten worden gebruikt, zodat alle omgevingen synchroon lopen en hetzelfde gedrag reproduceren.

### Config

Sla de configuratie op in omgevingsvariabele. Er moet een strikte scheiding zijn tussen configuratie en code. De code moet hetzelfde blijven, ongeacht waar de applicatie wordt geïmplementeerd, maar configuraties kunnen verschillen.

### Backing services

Behandel ondersteunende services als bijgevoegde bronnen, aangezien services gemakkelijk uitwisselbaar moeten zijn. De back-upservice moet gemakkelijk van de ene provider naar de andere kunnen gewisseld worden zonder grote code wijzigingen. Dit zorgt voor een goede draagbaarheid en helpt het ​​systeem beter te onderhouden.

### Build, release, run

Een toepassing met twaalf factoren vereist een strikte scheiding tussen de fasen Build, Release en Run. Elke release moet altijd een unieke release-ID hebben en releases moeten een rollback mogelijkheid hebben. Automatisering en onderhoud van het systeem moet zo eenvoudig mogelijk zijn. Vervolgens wordt alles samengevoegd in iets dat kan worden installeren en uitvoeren van de software mogelijk maakt.

### Stateless Processes

Applicaties moeten als één, stateless proces worden uitgevoerd. De Twaalf-factor processen zijn staatloos en delen niets. Deze factor vormt de kern van de microservices-architectuur.

### Port Binding

De service moet zichtbaar zijn voor anderen via port binding. Als een service is gebouwd, zorg er dan voor dat andere services dit als een bron kunnen behandelen. Een twaalf-factor-app staat volledig op zichzelf.

### Concurrency

Verdeel de app in microservices in plaats van te proberen de applicatie groter te maken (door een enkele instantie uit te voeren op de krachtigste beschikbare machine). Kleine, gedefinieerde apps maken het mogelijk om naar behoefte uit te schalen om zo de verschillende belastingen aan te kunnen. Elk proces moet individueel worden geschaald, met Factor 6 (Stateless) is het eenvoudig om de services te schalen.

### Disposability

Processen zouden minder tijdrovend moeten zijn. Zorg ervoor dat een applicatie snel kan rennen en stoppen. En dat het falen aankan. Zonder dit worden automatische schaalvergroting, implementatie en ontwikkeling vermoeilijkt. Dit kan met containers.

### Dev/prod parity

Houd ontwikkeling, staging en productie zo veel mogelijk op elkaar zodat iedereen het kan begrijpen en vrijgeven. Voor continue implementatie is continue integratie vereist op basis van overeenkomende omgevingen om afwijkingen en fouten te beperken. Dit stimuleert ook impliciet een DevOps-cultuur waarin softwareontwikkeling en -operaties verenigd zijn. Containerisatie is hier een enorme hulp.

### Logs

Behandel logboeken als gebeurtenis stromen. Logging is belangrijk voor het debuggen en controleren van de algemene status van de applicatie. Tegelijkertijd mag de applicatie zich niet bezighouden met de opslag van deze informatie. In plaats daarvan moeten deze logboeken worden behandeld als een continue stroom die wordt vastgelegd en opgeslagen door een afzonderlijke service.

### Admin processes

Beheer- / beheertaken uitvoeren als eenmalige processen - taken zoals database migratie of het uitvoeren van eenmalige scripts in de omgeving. Om te voorkomen dat er met de database geknoeit wordt, wordt er gebruik gemaakt van tooling die de database doet controleren.

Meer lezen op: [http://12factor.net/](https://12factor.net/)

## Wat zijn de kosten van de te implementeren cloud services, en zijn er manieren om hier kosten op te besparen door onderdelen juist wel of niet in de cloud te laten draaien?

In de casus staat dat er jaarlijks ongeveer 4 miljard pintransacties worden gedaan waar een algoritme wordt uitgevoerd.

### Aannames

Om hier een duidelijke inschatting te maken over hoeveel dit gaat kosten gaan we aannames opstellen van de services die de pintransacties verwerkt, deze aannames worden gemaakt omdat er te weinig context in de casus staat.

Hiermee kunnen we duidelijker het leerdoel “De student kan een redelijk schatting maken van hoe duur cloud services zijn.” aantonen.

* De applicatie die de verzoeken van transacties ontvangt is een Rest API.
* De database staat los en hoeft niet gehost te worden.
* De Rest API krijgt 4 miljard verzoeken in een jaar

### klant eisen

De klant wil dat de cloud-service aan de volgende eisen voldoet:

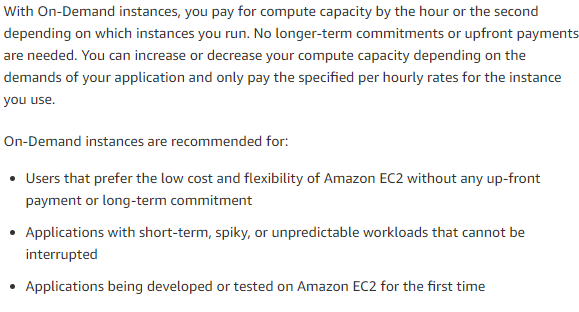
* High community acceptance
* low cost
* fast response
* minimal resource use/ auto scaling
* Usage spike / autoscaling

### De grootste spelers

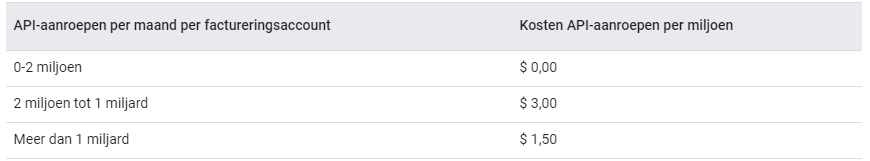
Op dit moment zijn er 3 cloud-providers die de grootste spelers zijn voor het hosten van cloud-services wereldwijd. Hier wordt informatie weergeven van de kosten om een api te hosten.

* Windows Azure  
  <https://azure.microsoft.com/en-us/pricing/details/api-management/>  
  keuze per verzoek te doen, of per uur, goedkoopste per uur kost ongeveer 700 per jaar.

SLA zegt uptime 99.95+%.

* Amazon EC2

SLA zegt 99.99%

* Google 

Meest opvallende van elke van deze 3 providers is dat ze support bieden voor het automatisch scalen van de service, dus low usage low resource, high usage meer resources. Prijzen hangen niet af van hoeveel gebruik op een moment, prijzen worden per uur bepaald, of per verzoek.

### Prijzen per jaar

Elke cloud-provider heeft zijn eigen manier om dit te berekenen, en er zijn ook nog meer manieren per service om het te berekenen. Bij amazon heb ik gekeken naar vm’s hosten en bij google en azure naar rest-api’s.

**De grote cloud-providers geven je een manier om de kosten te berekenen waardoor het gehele proces een stuk makkelijker is.**

Google: 4 miljard/12 = 333.333.333,333 verzoeken per maand / 1.000.000 \* 3 = 1000$

Azure: 613.2 euro

amazon: 10 vm’s met 4 cores 16gb ram **5,235 per**  3 jaar

### Conclusie

Elke van deze providers ondersteunt de eisen van de klant. Het voordeel nu met Azure en Google is dat je minder vast zit aan je service hosting, Amazon zit je meteen 3 jaar vast voor een vm, maar je kan hier ook een api op hosten wat dezelfde voordelen biedt.

Prijs verschil is vrij minimaal voor een grootschalig bedrijf dat de IBAN’s regelt. Dus wil je meer opletten aan wat je vast zit en welke verschillende voordelen elk biedt.

## Welke onderdelen van een applicatie kunnen worden verplaatst naar cloud services, is het wel handig/veilig om alle services op een cloud omgeving te hosten?

In principe kunnen alle onderdelen naar de cloud verplaatst worden. Met onderdelen worden de volgende dingen bedoeld:

* Frontend
* Backend
* Datastores
* Middleware

Voordat de beslissing gemaakt wordt of een applicatie naar de cloud overgezet wordt is het handig om een goed idee te krijgen van het applicatielandschap dat naar de cloud overgezet moet worden. Hierbij is het handig om een schatting te hebben van de prijsopgave voor het eerder genoemde applicatie landschap.

Ook veiligheid is een belangrijke factor in deze overweging. Een cloud service provider moet een bepaalde veiligheid garanderen, dit niveau van veiligheid is vaak beter dan wat een bedrijf kan betalen als ze eigen servers aanschaffen.

Een aantal andere voordelen van cloud services tegenover een lokale service zijn:

* Ook wordt er bij cloud services een uptime gegarandeerd van 99%>. Dit zit bij de prijs inbegrepen
* In principe is er onbeperkte schaalbaarheid. Op het moment dat de applicatie meer performance nodig heeft kan dit in een fractie van een seconde geleverd worden.

Dit maakt het ook erg makkelijk om bijvoorbeeld met kerst, wanneer de serverload van bijvoorbeeld bol.com hoger wordt dan normaal, automatisch mee te schalen. Op het moment dat dit bij een lokale machine gedaan moet worden is het een stuk moeilijker en ligt de prijs vaak een stuk hoger.

* Ook kunnen cloud services relatief makkelijk opgezet worden.

Dit zijn dingen die meegenomen moeten worden in de keuze voor een cloud service provider of een lokaal systeem.

## Hoe kan worden bepaald of opslag op een cloud service veilig is? Kunnen we bevestigen dat een cloud hosting bedrijf genoeg stappen heeft gezet om veiligheid van data te garanderen?

### Loss of data

De 3 manieren waarop meestal data verloren wordt zijn:

1. **Toevallige verwijdering / gebruikersfout**: misschien is de meest voorkomende bron van gegevensverlies bij het gebruik van cloud opslag onbedoelde verwijdering.
2. **Overschrijven van gegevens**: het is ook mogelijk dat informatie per abuis door gebruikers of door toepassingen wordt overschreven. Software-as-a-Service (SaaS) -toepassingen zijn een potentiële bron van enorm gegevensverlies. Deze apps bevatten en updaten grote datasets continu. Nieuwe informatie kan oude informatie overschrijven en kan ervoor zorgen dat datasets gedeeltelijk worden overschreven.
3. **Schadelijke acties**: de meeste providers van cloud opslag doen er alles aan om hun netwerken en uw gegevens te beveiligen, maar niet alle aanvallen kunnen worden voorkomen.

Microsoft Azure, Amazon AWS en Google Cloud hebben allemaal meerdere manieren om data loss tegen te gaan. Ze maken hiervoor gebruik van een DLP. De DLP wordt bij iedere cloud service op een eigen manier vorm gegeven. Dit zijn paginas die er dieper op ingaan per cloud service:

* [Amazon AWS](https://cloud.netapp.com/blog/aws-data-loss-prevention-tools-and-strategies)
* [Microsoft Azure](https://azure.microsoft.com/nl-nl/blog/data-integrity-in-azure-sql-database/)
* [Google Cloud](https://cloud.google.com/dlp?hl=nl) ([demo](https://cloud.google.com/dlp/demo/?hl=nl#!/))

### DDoS

Je service wordt overladen door verkeer van een netwerk met malicious intent, hierdoor kan jouw service niet meer normaal functioneren en gaat alles veel trager en wordt de response tijd langer.

bron (<https://azure.microsoft.com/nl-nl/services/ddos-protection/#features>)

Als voorbeeld hier pakken we azure, azure doet het volgende ter preventie van DDoS aanvallen:

* Grote capaciteit.
* Netwerk configuratie dat controleert op DDoS aanvallen waardoor deze vrijwil in realtime detecteren.
* Analysetool waarmee je jouw service kan afstellen om DDoS tegen te gaan.
* Meerlaagse bescherming.
* Azure monitor waardoor je overzicht creëert van je netwerk verkeer.
* Aanval analyse, wanneer een aanval gebeurt wordt deze geanalyseerd om de volgende beter preventie kan leveren.
* Rapid Response team
* Bescherming van kosten, wanneer iets als een DDoS gebeurt dan worden deze kosten niet gemaakt voor jou, dit is de verantwoordelijkheid van Azure zelf en dus wordt je vergoed.

### Malware

bron (<https://docs.microsoft.com/en-us/azure/security/fundamentals/antimalware>)

Veel grote microservice platformen zoals Azure hebben real-time malware protections tools geintegreert in het platform zitten. Deze tools helpen met het identificeren en verwijderen van virussen. Ook stuurt het notificaties wanneer een kwaadwillend programma zichzelf probeert te installeren op een Azure systeem.

Microsoft Antimalware for Azure wordt gebruikt vanwege de volgende features:

* Real-time protection, het monitoren van de activiteit op de cloud services om malware te detecteren en blokkeren
* Scheduled scanning, de systemen worden regelmatig gescand om malware te detecteren
* Malware remediation, er wordt automatisch actie genomen tegen gedetecteerde malware door ze in quarantaine te plaatsen of te verwijderen
* Signature updates, automatisch nieuwe updates uitvoeren op het gebied van veiligheid
* Antimalware Engine Updates, automatische updates voor het Microsoft Antimalware engine
* Antimalware Platform updates, automatische updates voor het Microsoft Antimalware platform
* Active protection, rapporteren van metadata om sneller te kunnen handelen
* Samples reporting, deelt data met over gevonden malware en verdachte resources met Azure om sneller te kunnen handelen in andere gevallen
* Exclusions, er is de mogelijkheid om uitzonderingen in te stellen wanneer dit nodig is
* Antimalware event collection, een collectie van service gezondheid, verdachte activiteiten en acties die er tegen genomen zijn

### Access rights en Access levels

Access levels zorgen ervoor dat een gebruiker beperkte toegang heeft tot een server. Dit kan heel handig zijn wanneer een persoon geen root level access nodig heeft om zijn taak uit te voeren. Dit zorgt er niet alleen voor dat die persoon niet zomaar bij verschillende delen binnen de server kan komen maar ook dat hij/zij niet zomaar bepaalde instellingen kan aanpassen bij de cloud provider. Dit zorgt ervoor dat er een stuk risico wordt ingeperkt wanneer meerdere mensen toegang moeten hebben tot de cloud service. Ook wanneer een wachtwoord wordt gekraakt heeft de hacker maar beperkte toegang.

### Service-Level Agreement

Een SLA is een overeenkomst tussen cloud provider en consument. In deze overeenkomst worden een aantal garanties uiteengezet. Deze garanties hebben betrekking tot de uptime, availability en security van de cloud platform. Hoewel dit geen garantie is voor de veiligheid van de cloud provider geeft dit wel aan wat de consequenties zijn voor het niet voldoen aan de garanties die zijn opgezet in de SLA. Wanneer een provider zich niet houdt aan de afspraken kan dit grote financiële gevolgen hebben voor het bedrijf. Hoewel dit geen garantie is voor de veiligheid van de software of data die men host bij die provider, is het wel een goede indicatie dat een bedrijf er alles aan zal doen om die veiligheid te waarborgen.

# 7. Discussieer de bevindingen

Leerdoelen van de student zijn behaald, na het bespreken van de individuele studie.

## Advies

Het is niet verstandig om voor serverless te gaan want dit biedt geen goede support voor een groot aantal verzoeken. Ook heeft het gebruik van serverless limitaties op het gebied van technieken die gebruikt kunnen worden.

Het advies is om voor een web API te gaan die wordt gehost op een van de grote cloud providers, het kostenverschil is minimaal voor de grote van schaal.  
  
Wat wel bedacht moet worden is welke voordelen haalt dit bedrijf eruit om de service op de cloud te draaien. Hierover kunnen we dus geen advies leveren over dit onderdeel aangezien we context missen.