APPLICATION PLATFORM AS A SERVICE

KOEN VANDERKIMPEN

Smals SECTIE ONDERZOEK 03/2014 • RESEARCH NOTE • 31

Abstract. De cloud is onze wereld van computing drastisch aan het veranderen. *Infrastructure* en *Software as a Service* (IaaS en SaaS) raakten reeds sterk ingeburgerd, maar nu is ook het middelste deel van de cloud stack – *Platform as a Service* (PaaS) – aan een opmars bezig. Dit keer is het de softwareontwikkeling zélf die de impact van de cloud voelt. In deze studie werd de grootste groep van PaaS-producten, de Application PaaS (aPaaS) onder de loep genomen. aPaaS-platformen leggen een sterke focus op het standaardiseren en automatiseren van een groot aantal ontwikkelingsaspecten, en zorgen op die manier voor een kortere time-to-market, lagere kosten en een verhoogde dienstverlening.

Résumé. Le cloud est en train de révolutionner le monde de l'informatique. En effet, si l'*Infrastructure as a Service* et le *Software as a Service* (IaaS et SaaS) sont déjà profondément ancrés, la couche centrale de la pile du cloud - *Platform as a Service* (PaaS) - est à son tour en pleine progression. Cette fois, l'impact du cloud se fait sentir sur le développement de logiciels même. Dans cette étude, nous avons passé sous la loupe le plus grand groupe de produits PaaS, l'Application PaaS (aPaaS). Les plateformes aPaaS sont clairement axées sur la standardisation et l'automatisation d'une multitude d'aspects du développement, permettant une réduction du time-to-market, une baisse des coûts et une augmentation du service.

Inhoud

1. Wat is een aPaaS?

Evoluties op infrastructuurniveau hebben steevast een impact op softwareontwikkeling. Voor de cloud, de grootste evolutie die we de laatste jaren al meemaakten, komt deze impact er voornamelijk in de vorm van Platform as a Service.

Met 'Platform' wordt normaliter de middleware bedoeld. Dit zijn dus de applicatieservers, messaging, databases, security tools, integratie-hulpmiddelen, etc. Kortom, het is de infrastructuur die de meeste applicaties nodig hebben om goed te kunnen werken, zeker op enterprise-niveau.

Met 'as a Service' bedoelen we dan het aanbieden van die middleware als een gemakkelijk te consumeren dienst, gebruikmakende van cloud-technologie. Kenmerkend hierbij is dat onderliggende beschikbare resources gepoold kunnen worden en gebruikt worden om middelen met een hoger niveau van abstractie aan te bieden. Daardoor hoeft de gebruiker deze onderliggende resources niet meer te beheren en wordt hij ondersteund bij het schalen van zijn verbruik.

Er zijn verschillende types PaaS. Om er een paar te noemen: 'integration PaaS' (iPaaS), 'business process management PaaS' (bpmPaaS) en 'data store platforms as a service' (dbPaaS). De belangrijkste categorie is echter 'Application Platform as a service' (aPaaS). Deze komt ook het meeste voor in de industrie en wordt daardoor vaak kortweg PaaS genoemd.

aPaaS-platformen vangen een aantal taken op die bij een softwareontwikkelingsproject telkens opnieuw moeten gebeuren en soms vertraging opleveren. Dit zijn, onder meer, het opzetten van servers (inclusief OS), het installeren en configureren van de vereiste applicatie- en webservers, databases en andere middleware, en vaak ook het opzetten van de tooling rond de applicatielevenscyclus, zoals b.v. versiecontrole. Daarnaast bieden ze vaak mechanismen om het schalen van een toepassing te ondersteunen en om te meten hoeveel van de onderliggende middelen een applicatie verbruikt.

Een iets bredere inleiding op aPaaS-technologie gaven we ook reeds mee in een blogpost[1]: 'Productiviteitsverhoging met PaaS'.

2. Cloud Context

PaaS-platformen vormen de middelste laag in de zogenaamde *'cloud stack'*. De onderste laag in deze stack is *Infrastructure as a Service* (IaaS). Dit zijn de typische cloud-diensten, waarbij men snel en gemakkelijk virtuele machines kan laten werken via het netwerk. Deze zijn al sterk ingeburgerd. Ook bij Smals heeft deze technologie zich reeds in een initiatief vertaald om een *'private cloud'* op te zetten, wat dus eigenlijk een intra-muros IaaS-platform is.

[1] http://www.smalsresearch.be/archives/5995

Fig. 1: De Cloud Stack

Aan de andere kant van het spectrum zien we *Software as a Service* (SaaS). Bij deze cloud-technologie komt er bij de gebruiker slechts weinig of geen IT-kennis meer aan te pas: er wordt, op aanvraag, software aangeboden die eventueel gecustomiseerd kan worden en daarna als online dienst aan eindgebruikers kan worden aangeboden. Een typisch voorbeeld is Gmail for Business. De dienst die verleend wordt is e-mail, en de klant (een bedrijf) kan hiervan een aantal zaken configureren om de dienst op haar noden en die van haar eindgebruikers (werknemers van het bedrijf) af te stemmen. Het voordeel voor Google is dat eenzelfde platform dat zij aanbieden door tal van bedrijven tegelijk gebruikt kan worden. Het voordeel voor het bedrijf is dat het niet meer zelf moet instaan voor het opzetten en onderhouden van de e-mailinfrastructuur.

Tussen deze twee uitersten, namelijk het hebben van virtuele machines versus volledig afgewerkte software, situeert zich dus PaaS. Daarbij plaatst men een middleware-laag boven de machines om abstractie te maken van deze onderliggende infrastructuur en het ontwikkelen van software te ondersteunen.

Men kan de drie lagen van de stack ook naast elkaar plaatsen als in figuur 2. Hier zien we de toenemende automatisatie (een belangrijk kernwoord bij cloud-technologie) naarmate men hoger in de stack gaat: bij IaaS zien we dat o.a. het applicatieplatform nog door de gebruiker moet worden voorzien. Bij SaaS is alles geautomatiseerd. Voor wie software ontwikkelt, is PaaS de gulden middenweg: enkel de applicatie moet nog worden gebouwd, de onderliggende lagen zijn zoveel mogelijk geautomatiseerd en voorzien ‘als dienst’.

De andere kant van dit verhaal is uiteraard dat hoe meer controle men moet afstaan, des te hoger men in de stack gaat. In alle lagen van de stack leveren we de virtualisatie en controle van hardware en storage over aan degene die de dienst voorziet, bij PaaS komt hier ook het OS en het applicatieplatform bij, en bij gebruik van SaaS zijn we helemaal gebonden aan de applicatie zoals ze wordt voorzien.

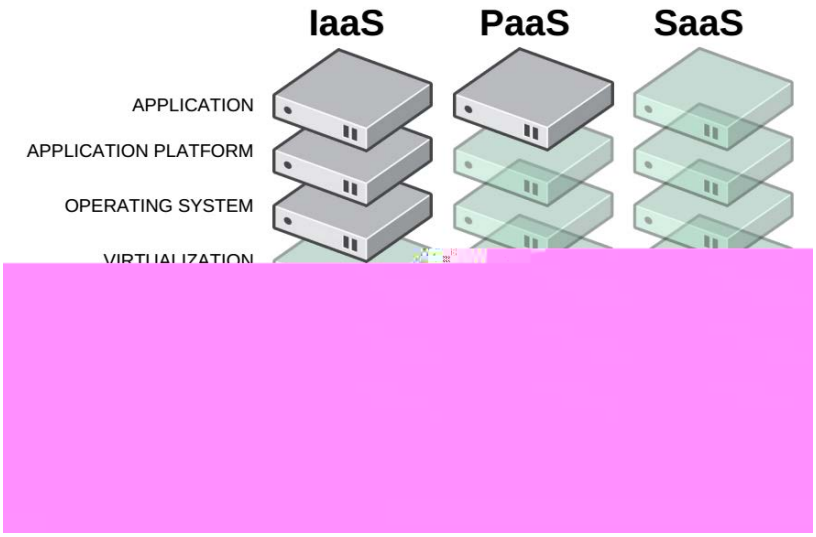

Fig. 2: Niveaus van Controle en Automatisatie

Uiteraard behouden we die controle wel indien we tegelijk gebruiker en aanbieder van de cloud-dienst zijn (via een private cloud). Hoe dit werkt is te zien in figuur 4 in de sectie 'Eigenschappen en Marktstudie', waar we zien wie welke laag controleert in een public cloud en in een private cloud.

3. Voordelen & impact

De belangrijkste doelstellingen van een aPaaS zijn het sneller opleveren van applicaties (time-to-market) en meer algemene kostenverlagingen. Deze voordelen volgen voor een groot stuk uit doorgedreven **automatisatie** en **standaardisatie**. Deze laatste twee eigenschappen hebben dan ook hun gevolgen voor de organisatie. In deze sectie zullen we van de belangrijkste eigenschappen van aPaaS bespreken hoe ze hun voordeel opleveren en welke impact ze verder hebben.

3.1. Gecontroleerde Self-Service

Veruit de meest kenmerkende eigenschap van cloud-technologieën, en dus ook van aPaaS, is zelfbediening. In dit geval zijn het software-ontwikkelaars (niet noodzakelijk enkel de developers, maar b.v. ook architecten) die aan self-service kunnen doen.

In het ideale geval zal de ontwikkelaar zich aanmelden op het platform en zijn applicatie initialiseren. Vervolgens kiest hij de grootte van de nodige (gestandaardiseerde) resources, de middleware, databronnen, biblio-theken etc. die zijn applicatie zal nodig hebben. Tenslotte krijgt hij de mogelijkheid om broncode naar het platform te sturen (of deze online in te voeren) en zijn applicatie automatisch te laten bouwen en uit te voeren. Al deze stappen, behalve het eigenlijke schrijven van de applicatie, hoeven slechts minuten in beslag te nemen. Dit is uiteraard een winst in termen

van time-to-market, indien men dit vergelijkt met de situatie waarin de ontwikkelaar zelf nog allerlei middleware moet installeren en configureren, of eventueel moet wachten tot andere teams dit voor hem doen. Een voorbeeld hiervan zien we in figuur 3: hier is het zo dat het aPaaS-platform een Jenkins-server gebruikt om de applicatie te bouwen en ook automatisch te ontplooien.

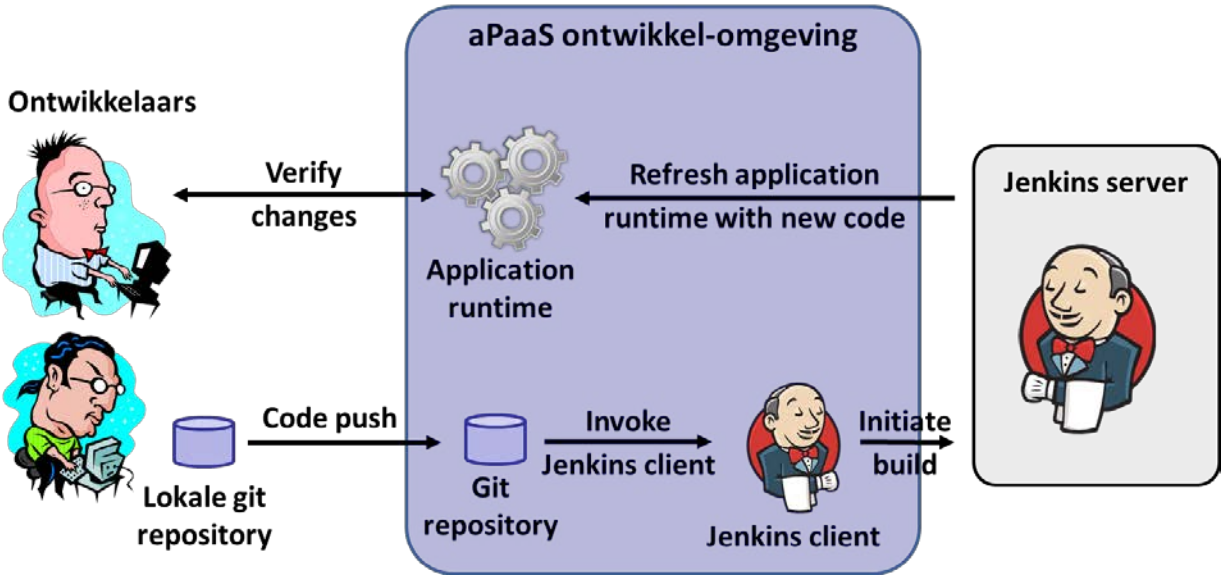

Fig. 3: Voorbeeld van een mogelijke workflow bij gebruik van aPaaS

Een sterke ondersteuning van de juiste tools is bij deze nieuwe manier van werken nog steeds van belang. In dat opzicht zijn er verschillende mogelijkheden. Er zijn b.v. aPaaS-producten die een sterke integratie hebben met traditionele ontwikkeltools (e.g. Eclipse) en waarmee ontwikkelaars dus goed vertrouwd zullen zijn. Anderzijds bestaan er aPaaS-platformen die sterke eigen tooling aanbieden (b.v. volledig online platformen).

Ook de traditionele workflow binnen een ontwikkelingstool kan door deze self-service worden veranderd. De ontwikkelaar kan hierdoor namelijk in principe meer eigen beslissingen nemen betreffende gebruikte technologie en tempo van inproductiestelling. Vaak krijgt men ook kortere release cycles. Zulke zaken moeten dan ook goed omkaderd worden: men kan er b.v. voor opteren om het toevoegen van nieuwe technologische keuzes enkel door bepaalde leden van een team te laten gebeuren. Daarnaast kan men ervoor zorgen dat de ontwikkelteams de toepassing enkel rechtstreeks in een test- of acceptatieomgeving kunnen ontplooien, en niet in een productie-omgeving.

De meeste aPaaS-platformen zijn flexibel genoeg in gebruik om verschillende ontwikkelingsmethodologieën te ondersteunen. Om bepaalde zaken ook effectief technisch af te dwingen, zullen soms kleine technische ingrepen nodig zijn, of support van een vendor.

3.2. Release Management

Bij het ontwikkelen van software maakt men doorgaans gebruik van een aantal verschillende stadia in de levenscyclus, b.v. Dev, Test, Acc en Prod. Deze stadia uiten zich vaak in een aantal verschillende 'omgevingen' waarin de toepassing zal werken. Release Management is nodig om een toepassing op een bepaald moment te migreren naar een volgende omgeving: dit is het zogenaamde 'promoveren'.

APaaS kan ook hier een voordeel opleveren: door het gemak en de snelheid waarmee men voor een applicatie de nodige onderliggende infrastructuur kan voorzien, zijn de promoties sneller te verwezenlijken. Bovendien krijgt men een grotere uniformiteit in de verschillende omgevingen dan voorheen, doordat deze bestaan uit gestandaardiseerde resources van het aPaaS-platform. Door dit laatste zullen er minder fouten zijn als gevolg van verschillen tussen de omgevingen. Op die manier krijgt men de kosten van het release management lager, bovenop wederom een kortere time-to-market.

3.3. Elastisch Horizontaal Schalen

Een aPaaS-platform omvat doorgaans een grote hoeveelheid onderliggende infrastructuur, van dewelke de organisatie wordt weggenomen van de ontwikkelaar en wordt overgenomen door het platform. Op deze manier werkt de ontwikkelaar met een systeem dat een hoger niveau van abstractie heeft. De infrastructuur kan op die manier aan applicaties worden aangeboden in een vorm waarin ze gemakkelijker op een schaalbare manier te gebruiken is. Dit effect wordt nog versterkt doordat het platform ook effectieve hulp aanbiedt bij het schalen van de applicatie.

Concreet zullen meer of minder van de door het platform gedefinieerde standaard resources (in termen van middleware) aan applicaties kunnen worden toegekend naargelang hun noden veranderen. Dit kan bijvoorbeeld gaan om het aantal web- of applicatieservers, waarbij load balancing door het platform wordt voorzien.

Deze elasticiteit op zich heeft uiteraard het voordeel van een verhoogde SLA (betere beschikbaarheid van de service, mindere gevoeligheid aan pieken in gebruik). Daar komt nog bij dat de kosten om dit te verwezenlijken lager zullen liggen.

Ten slotte heeft de schaalbaarheid als gevolg dat bij het beheer van een aPaaS-platform de nadruk meer zal liggen op Capacity Management dan op de directe dienstverlening aan ontwikkelteams. Er zal b.v. op moeten worden gelet wat het gemiddelde en maximale verbruik is van de applicaties en op basis hiervan zal de nodige onderliggende hardware worden voorzien om slechts een bepaald percentage van de tijd onvoldoende capaciteit te hebben (afhankelijk van SLA's). Het voordeel is dat men deze oefening enkel moet doen voor het platform, en niet voor elke applicatie apart.

3.4. Multi-tenancy & Isolatie

Zoals in alle 'as a Service'-modellen zit het aspect multi-tenancy ingebakken in aPaaS. In dit specifieke geval gaat het dan over de applicaties als zogenaamde 'tenants' van het systeem. Concreet betekent dit onder meer dat er tussen de applicaties een hoge graad van isolatie zit: ze kunnen elkaar niet rechtstreeks beïnvloeden, en ze kunnen geen gebruikmaken van elkaars resources (voor zover er genoeg resources beschikbaar zijn in het platform). Dit betekent dat b.v. een fout in de ene applicatie slechts tot het falen van die applicatie kan leiden, en niet van het volledige systeem of van andere applicaties.

Ook binnen een applicatie kan men spreken van een zekere isolatie tussen verschillende componenten, en dan vooral tussen identieke componenten die onderdeel uitmaken van een geschaalde toepassing. Indien er een falen zou optreden in één van deze als gevolg van een aan de applicatie exogene fout, dan blijft dit falen ook beperkt tot die ene component en kan de toepassing in haar geheel verder blijven werken.

Op deze manier zal het isolatie-aspect, samen met de schaalbaarheid, een verhoogde graad van redundantie tot gevolg hebben. Toepassingen zullen dus weerbaarder zijn voor fouten en een hogere beschikbaarheid hebben, wat een betere SLA mogelijk maakt.

Naast het isolatie-aspect, zijn aPaaS-platformen in staat om het verbruik van resources van elke applicatie apart te meten, en dit op een fijnschalige manier (b.v. in geheugenverbruik, CPU-verbruik, aantal benodigde applicatieservers of eventueel zelfs het aantal rijen in een database, dit alles i.p.v. b.v. het aantal cores of servers). Dit wordt mogelijk doordat men deze verbruiksmeting slechts eenmaal moet opzetten voor het platform in zijn geheel, in tegenstelling tot de traditionele manier waarbij men dit zou moeten doen voor elke applicatie apart. Het platform in zijn geheel is daarnaast ook gemakkelijker te onderhouden dan alle applicaties apart.

De fijnmazige manier van verbruiksmeting wordt dan ook vaak gebruikt als basis voor de facturatie, om zo het 'as a Service'-model volledig uit te spelen. Er ontstaat een win-win-situatie: de klant betaalt enkel voor zijn verbruik (een betere SLA), en de aanbieder van het platform kan zijn onderhoudskosten verminderen dankzij de consolidatie van resources.

3.5. Integratie

Nagenoeg alle tegenwoordig ontwikkelde software moet integreren met bestaande systemen. Dit is zelfs niet louter om functionele redenen: integratie kent ook heel wat non-functionele toepassingen, zoals het verbinden van een systeem met een monitoringsysteem.

Bij het gebruik van een aPaaS-platform kan men zich de vraag stellen of zo een systeem het niet moeilijker maakt om erop draaiende toepassingen te integreren met reeds bestaande diensten? Het is namelijk zo dat de toepassing als het ware 'in' het aPaaS-platform leeft, waardoor communicatie met zaken buiten het platform eventueel technisch

moeilijker wordt. Gelukkig is dit bij de meeste producten geen enkel probleem: de op een typische aPaaS ontplooide toepassingen hebben dezelfde verbindingsmogelijkheden als een typische applicatie op een Linux-server.

Op twee zaken dient echter te worden gelet: indien de cloud waarin de aPaaS functioneert in een zone ligt die niet te vertrouwen is (e.g. de public cloud) wordt integratie extra bemoeilijkt door de veiligheidsproblematiek. Ten tweede, zijn de integratiemogelijkheden van de zogenaamde 'zero coding'-platformen (zie verder) iets beperkter.

De beste manier om aan integratievereisten te voldoen is het volgen van één van de kernbegrippen van aPaaS: **standaardisatie**. Standaardmanieren om toepassingen of diensten met elkaar te integreren (e.g. web services) zullen namelijk quasi altijd mogelijk, zo niet ondersteund zijn.

3.6. Overzicht

De kerneigenschappen van aPaaS, automatisatie en standaardisatie, hebben verschillende gevolgen. Wij vatten deze als volgt samen:

- Gecontroleerde Self-Service
  - Verlaagt de Time-to-Market
  - 'Empowert' de ontwikkelaar
- Gemakkelijke Migratie / Promotie
  - Verlaagt de Time-to-Market
  - Verlaagt de kosten
- Elastisch Horizontaal Schalen
  - Maakt een betere SLA mogelijk
  - Verlaagt de kosten
  - Legt de nadruk op Capacity Management
- Multi-Tenancy & Isolatie
  - Maakt een betere SLA mogelijk door verhoogde redundantie
  - Verlaagt de kosten
- Integratie
  - Blijft een aandachtspunt
  - Nood aan standaardisatie

4. Types en Eigenschappen, Marktoverzicht

We treffen binnen de wereld van aPaaS erg uiteenlopende producten aan, een effect dat nog eens wordt versterkt door de populariteit van de term 'PaaS'. Op basis van een aantal eigenschappen kunnen we verschillende elkaar kruisende categorieën opstellen.

4.1. Deployment Model

Het duidelijkste onderscheid dat kan worden gemaakt, is dat op basis van waar de oplossing draait. Een aPaaS-platform kan zowel in een publieke cloud als in een private cloud aangeboden worden. Bovendien zijn er leveranciers die zowel een publiek cloud-platform aanbieden, als software om een aPaaS op te zetten op eigen infrastructuur. In figuur 4 zien we uitgebeeld wie in elk deploymentmodel de controle uitoefent over elk deel van het systeem. Hieruit blijkt dat de gebruikerservaring van de ontwikkelaar (de gebruiker van het platform) in principe onveranderd blijft. In het private model blijft echter alle onderliggende infrastructuur, en dus ook de data, onder controle van de eigen IT-dienst. Dit is b.v. nuttig indien er vertrouwelijke gegevens worden verwerkt op het platform.

Fig. 4: Beheer van aPaaS-lagen in public en private cloud

Wat we zien in de markt, is dat de meeste grote spelers zowel een 'public' als 'private cloud'-oplossing aanbieden (e.g. Microsoft met het Azure[2]-platform en een private cloud gebaseerd op Windows Server; Oracle met Oracle Java Cloud Service[3] en het private Oracle Cloud Platform). Kleinere spelers beperken zich vaker tot één model.

[2] https://www.windowsazure.com/en-us/

[3] https://cloud.oracle.com/java

Het voordeel aan vendors of producten die zowel publieke als private ondersteuning bieden, is de mogelijkheid om met een *'hybrid cloud'* te werken. Applicaties kunnen dan met een beperkte effort gemigreerd worden tussen 'public' en 'private cloud'-omgevingen. Men spreekt hier ook van *'cloud portability'*. Dit model kan voor een aantal zaken nuttig zijn:

- Cloud bursting: pieken opvangen in de cloud;
- Experimenteren in de public cloud, in productie zetten in de private cloud;
- Load testing.

4.2. Niveau van Abstractie

PaaS is dan wel het midden van de 'cloud stack', tussen IaaS en SaaS, maar kan eigenlijk ook gezien worden als een spectrum van producten die tussen deze twee lagen liggen.

Zo zijn er aPaaS-producten die niet veel meer doen dan een automatisatielaag toevoegen bovenop de IaaS-laag, die er b.v. voor zorgt dat de nodige applicatieservers en databases worden geïnstalleerd, maar die deze producten niet omvatten of beheren. Een voorbeeld is GigaSpaces Cloudify[4].

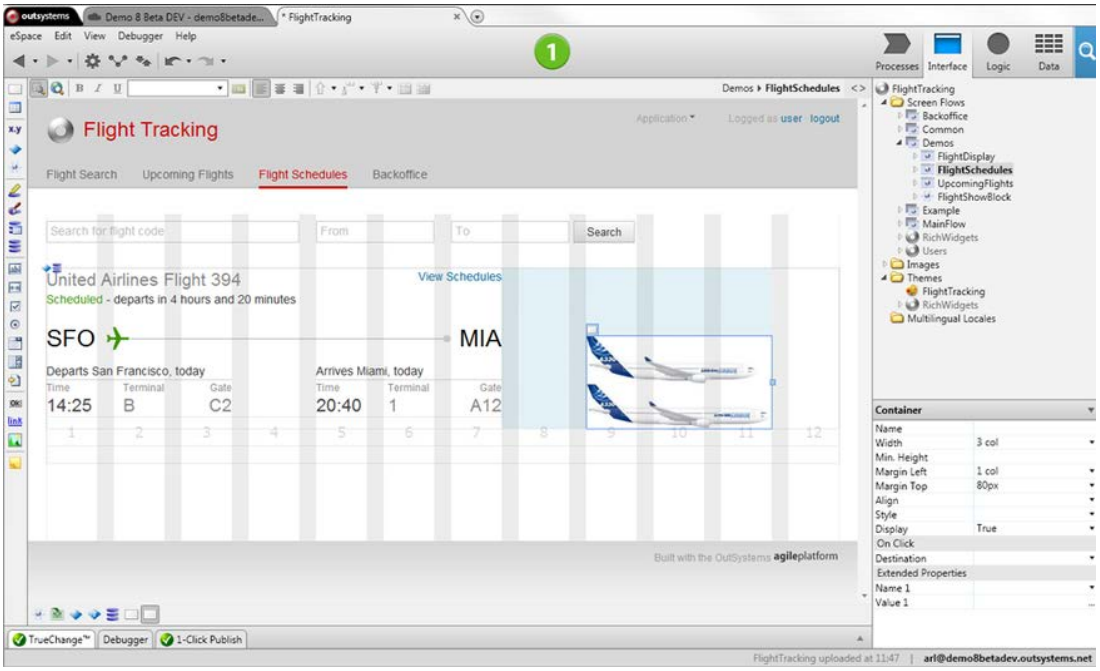

Fig. 5: Outsystems, een 'zero-coding'-oplossing

Aan de andere kant van het spectrum vinden we de zogenaamde *'zero coding'*-oplossingen. Deze producten leunen dicht bij SaaS aan, in die zin dat het grootste deel van de ontwikkeling van een applicatie online gebeurt, d.m.v. grafische hulpmiddelen in een webtoepassing, zoals dit

[4] http://www.cloudifysource.org/

ook gebeurt bij de configuratie en parametrisatie van echte SaaS-producten. Het verschil is dat het hier niet gaat om een bepaalde software die men aan zijn behoeften aanpast, maar om een platform waar men nog alle kanten mee uit kan en verschillende soorten toepassingen mee kan voorzien. Uiteraard zijn er, door de aard van dit soort producten, wel iets meer beperkingen op de mogelijkheden dan bij PaaS-producten met een lager niveau van abstractie – het typische verhaal van verhoogde automatisatie en verminderde controle. Deze verminderde controle vertoont dan ook het volgende nadeel: typisch laten 'zero coding'-toepassingen enkel integratie met externe systemen toe op geijkte manieren, b.v. via een welgedefinieerde API of webservice. Dit beperkt uiteraard de integratiemogelijkheden.

Voorbeelden van 'zero coding'-oplossingen zijn Outsystems[5] en Zoho Creator[6]. Ook Force.com van Salesforce leunt hier dicht bij aan. De meeste producten in deze categorie zijn enkel nog maar te verkrijgen in de publieke cloud. Toch zijn er ook al een aantal te verkijgen voor de private cloud-markt: b.v. Wavemaker[7], Longjump, Mendix, Outsystems (Fig. 5), en Rollbase[8].

Ergens in het midden van het spectrum, vinden we de producten terug die voor 'traditionele' software-ontwikkelaars de gulden middenweg vormen: ze abstraheren de infrastructuur in voldoende mate weg, zodanig dat er enkel nog sprake is van de middlewarecomponenten die een applicatie nodig heeft. Deze componenten worden dan door het platform voor een veelvoud aan applicaties tegelijk opgezet, beheerd en gemonitord.

In deze categorie vinden we de grote 'Open Source'-spelers terug, zoals Pivotal Cloud Foundry[9] en Red Hat Openshift[10]; beide bieden echter ook betalende versies van hun producten aan.

Het niveau van abstractie van aPaaS-producten komt vrij goed overeen met wat men het onderscheid tussen *'cloud native'*- en *'cloud based'*-producten noemt. 'Cloud native'-producten zijn die aPaaS-platformen die reeds van bij het begin werden geconcipiëerd voor gebruik in de cloud en dus ook een hogere mate van automatisatie en integratie bieden. De 'cloud based'-producten zijn de reactie van vendors met een reeds hoge bestaande installed base van niet-cloud-gebaseerde software stacks. Deze laatste focussen zich op het toevoegen van cloud-technologie aan meer traditionele middlewareproducten.

Elke aanpak heeft voor- en nadelen: cloud native biedt b.v. meer automatisatie, maar de 'cloud based'-producten bevatten meer bij developers reeds gekende technologie.

---

[5] http://www.outsystems.com/

[6] https://www.zoho.com/creator/

[7] http://www.wavemaker.com/

[8] http://www.progress.com/products/rollbase

[9] http://www.cloudfoundry.com/

[10] https://www.openshift.com/

De thematiek van het spectrum van aPaaS-producten werd ook reeds uitvoerig geïllustreerd in een blog[11] op de site van Onderzoek: 'as a Service: een Waaier aan Mogelijkheden'.

4.3. Buy vs Build

Bij het opzetten van een aPaaS-platform zal zich de keuze stellen of dit aangekocht dan wel zelf opgezet zal worden. Onder opzetten moet hier natuurlijk wel worden verstaan: het zelf installeren van een bestaand (Open Source) aPaaS-platform binnen het eigen datacenter, en niet het effectieve bouwen ervan. Dit soort producten zit namelijk complex in elkaar en is ook vrij omvangrijk. Er zijn ook genoeg kwalitatieve producten op de markt, dus het zal altijd goedkoper zijn om het wiel niet te willen heruitvinden.

Ondanks dat het beperkt blijft tot installatie, is er tussen buy en build toch nog wel enig verschil, zelfs wanneer men zich enkel richt op een eigen aPaaS-platform in de private cloud (bij gebruik van de public cloud zit men sowieso in buy-modus).

buy

Wanneer men een aPaaS koopt, krijgt men een 'aPaaS-in-a-box'. Dit wil zeggen: de verkoper komt dit in het datacenter installeren als een zwarte doos en doet ook het beheer. Op die manier heb je dus als bedrijf zelf controle over de fysieke locatie van je gegevens, maar niet over het volledige beheer van de aPaaS-software. Het voordeel is dat alle mogelijke problemen de verantwoordelijkheid worden van de leverancier.

Vaak is het zo dat de 'buy'-oplossingen op hun beurt gebaseerd zijn op een 'open source'-oplossing, maar dan herverpakt en verbeterd door een leverancier. Een goed voorbeeld hiervan is Stackato[12] van ActiveState. Zij baseerden hun product op het 'open source'-platform Cloud Foundry, beheerd door de Pivotal-groep. Stackato is een stuk makkelijker in gebruik.

download

Bij gebruik van een 'open source'-oplossing staat men zelf in voor de volledige installatie en het beheer van het platform. Hier komt nog bij dat 'open source'-oplossingen typisch iets minder gebruiksvriendelijk zijn wat betreft het opzetten. Dit komt natuurlijk doordat de grote 'open source'-spelers ook graag hun – betalende – 'public cloud'-oplossingen verkopen, ofwel support.

[11] http://www.smalsresearch.be/archives/6108

[12] http://www.activestate.com/stackato

support

Met support komen we dan tot de gulden middenweg: men kan altijd uitgebreide support aanschaffen bij gebruik van een 'open source'-oplossing, zodat men ondersteuning heeft bij de installatie en exploitatie ervan, zonder enige controle te verliezen over het product of over de data.

Een goed voorbeeld van dit model is Red Hat. Van hun aPaaS, Openshift, bestaan zowel een 'public cloud'-versie, een 'open source private cloud'-versie en een 'enterprise private cloud'-versie. Voor deze laatste kan men dan nog bijkomende support aanschaffen.

4.4. Evolutie van de markt

De aPaaS-markt is nog redelijk nieuw en er zijn tientallen spelers. Een vrij groot aantal recente overnames zijn dan ook geen verrassing. Zo werd b.v. Rollbase overgenomen door Progress Software en Longjump13 door Software AG.

Fig. 6: Een greep uit het aPaaS-aanbod

De typische grote spelers in de softwarewereld, zoals b.v. Oracle en IBM, hebben een lichte achterstand op de kleinere, 'puur PaaS'-bedrijven. Microsoft biedt in deze categorie momenteel de meest volledige oplossing, met het 'Azure'-platform, weliswaar beperkt tot de .NET-wereld. Oracle werkt aan een 'public cloud'-aanbod gebaseerd op zijn traditionele stack (e.g. Oracle DB as a Service, Weblogic as a Service), en zet tegelijk in op de 'private cloud'-markt met de 'Exalogic'-systemen. IBM trekt ten slotte

[13] http://www.softwareag.com/special/longjump/index.html

de kaart van een ‘open source’-product, Cloud Foundry, en brengt speciale versies uit van de Websphere-server voor deze aPaaS.

Een jonge markt, in volle evolutie, betekent natuurlijk ook een zekere dosis immaturiteit. Wie nu reeds een keuze maakt voor een product loopt daardoor het risico deze over een paar jaar te moeten herevalueren. Aan de andere kant is het een strategisch voordeel om bij de ‘early mainstream adopters’ te horen.

Desondanks de nog immature markt zagen we in de studie duidelijk een aantal leiders naar voren komen, die vrij volledige oplossingen aanbieden en niet meer zo lang nodig zullen hebben om op enterprise-niveau te werken. Hier gaat het dan om **Pivotal Cloud Foundry, RedHat OpenShift en WSO2 Stratos**[14].

[14] http://wso2.com/cloud/stratos/

5. Besluit & Aanbevelingen

Application Platform as a Service (aPaaS) is de **middleware stack van de cloud**. Deze platformen ondersteunen het ontwikkelen van applicaties in het algemeen, en ook specifiek gericht op uitrol in een cloud-gebaseerde infrastructuur (public of private). Ze leggen een sterke focus op het standaardiseren en automatiseren van een groot aantal ontwikkelingsaspecten, en bieden op die manier de volgende voordelen:

- Kortere time-to-market
- Lagere kosten
- Verhoogde dienstverlening (betere SLA's mogelijk)

De markt is nog niet geheel matuur en oplossingen zijn nog jong. Naast de nood aan integratie met bestaande systemen is dit een risico, desalniettemin zijn er grote voordelen verbonden aan het gebruik van aPaaS.

We raden dan ook aan om de tendens naar het omarmen van deze cloud-technologie te volgen. Er moet kennis worden opgedaan, en er kan reeds worden begonnen met het **ontwikkelen van niet-kritische toepassingen op een aPaaS-platform**. Dit alles uiteraard in een **on-premise** deployment, om de eventuele vertrouwelijkheid van de door het systeem verwerkte gegevens te vrijwaren.

Een laatste algemeen advies is het volgende: volg niet alleen de technologie, maar ook de achterliggende *filosofie* van aPaaS: **automatisatie gecombineerd met het gebruik van standaarden**. Vendor lock-in kan voor een stuk vermeden worden door customisaties tot een minimum te beperken.

6. Demo

Tijdens de infosessie verbonden aan deze studie gaven we verschillende demo's. Deze lieten telkens zien hoe we een applicatie op een aPaaS-platform ontwikkelden. Filmpjes van de demo's (zonder geluid) zijn terug te vinden op de website[15] van Smals Research. De slides van de infosessie (december 2013) zijn daar eveneens beschikbaar.

Sectie Onderzoek van Smals brengt met regelmaat verschillende publicaties uit over een hele waaier aan topics in de huidige IT-markt. U kan deze publicaties opvragen via :

http://www.smalsresearch.be

[15] http:/www.smalsresearch.be