Adobe Experience Manager

Mike Heymans

2017

Contents

1 Voorwoord

In het kader van mijn opleiding Toegepaste Informatica aan de Hogeschool Gent heb ik onderzoek gedaan naar het Adobe Experience Manager platform, beter gekend als AEM. Het resultaat dat nu voor u ligt is er n van drie maanden hard werk waarin heel wat gevloek heeft plaatsgevonden. In nazicht van dit onderzoek ben ik tevreden over het resultaat maar vooral voldaan met de verworven kennis. Al heeft deze kennis gezorgd voor nieuwe vraagtekens die in de toekomst mijn aandacht zullen opslorpen.

Dit project is uitgevoerd in opdracht van AS Adventure, met mijn projectleider Steven Rymenans als begeleider.

Deze proef was niet mogelijk geweest zonder de inzet van AS Adventure, die zo vriendelijk waren om een licentie beschikbaar te stellen alsook mij te begeleiden in mijn ontdekkingsreis binnen de wereld van AEM. Ik wil ook mijn collegas bedanken voor hun uitmuntende begeleiding en eindeloos geduld.

Ik bedank ook mijn begeleiders voor hun samenwerking en ondersteuning tijdens dit traject.

Veel plezier met het lezen van deze scriptie.

2 Reden van onderzoek

Adobe Experience Manager is het gelicentieerde contentmanagementsysteem (CMS) van Adobe. Een CMS-applicatie maakt het mogelijk voor mensen zonder kennis van html of css om webpaginas aan te maken of te wijzigen. Het is ideaal voor websites die vaak de inhoud van hun site wijzigen of snel willen inspelen op actuele gebeurtenissen. Dit komt doordat designers zelf de paginas kunnen editeren in plaats van eerst content uit te denken en de realisatie over te laten aan een front-end developer.

Toen AS Adventure besloot om zijn huidige website, en die van zijn zusterbedrijven, niet langer uit te besteden aan een extern bedrijf maar deze intern te gaan beheren en ontwikkelen, is er besloten om deze via het AEM-platform op te zetten. Dit zou het bedrijf in staat stellen om zijn content dagelijks aan te passen zonder dat hiervoor ontwikkeling zou moeten gebeuren. Denk maar aan de homepage die van achtergrondafbeelding wijzigt of een gepersonaliseerde pagina voor een merk, allemaal mogelijk dankzij AEM.

Buiten de content die door het marketingteam wordt voorzien is er ook data die vanuit een databank moet komen. Hierbij hebben we het over data die niet manueel per record wordt aangepast maar met duizenden tegelijk of data die niet enkel betrekking heeft op de site maar ook tijdens andere processen nodig. Een voorbeeld hiervan zijn de prijzen, tijdens de solden is het niet reel om elke prijs handmatig aan te passen op de paginas. In het onwaarschijnlijke geval dat het handmatig aanpassen van een specifieke prijs nodig is, zou deze data enkel op de pagina gewijzigd zijn en niet in het ERP. Het is dus logischer om deze wijziging in het ERP te doen en deze te laten doorkomen op de pagina. Een tweede voorbeeld is een 1+1 gratis actie op bepaalde producten, deze willen we ook via het ERP kunnen instellen en laten doorkomen op de site alsook het weghalen hiervan.

Toen we aan dit project begonnen zijn, hebben we enkele manieren gevonden om deze wijzigingen weer te geven op de site en toen onze eerst twee sites, Juttu en Yaya, live stonden verliep alles vrij vlot. Maar deze shops zijn relatief nieuw en beperkt in aanbod, toen de nieuwe AS Adventure site live ging staken er enkele problemen de kop op. Aangezien deze problemen in productie voorkomen was wachten niet echt een optie en hebben we deze opgelost, niet zozeer met de beste maar wel de snelste oplossing.

Deze proef heeft als nut het herzien van wat we hebben opgebouwd, hoe we het AEM-platform aangepast hebben om aan onze behoeften te voldoen en of dit ook effectief de beste manier is om de vereiste functionaliteit te bekomen. Door het in kaart brengen van onze methodologien hopen we een dieper inzicht te verwerven in het platform op zich alsook een leidraad te voorzien voor ontwikkelaars die een eerste keer kennis maken met AEM of zij die reeds werken met AEM en geconfronteerd worden met diezelfde problemen waarvoor wij een

oplossing zoeken. Zoals het gezegde luidt: het is goed te leren van je fouten maar beter om te leren uit fouten van een ander..

3 De database

Om onze AEM-applicatie te bouwen hebben we nood aan een database waaruit we onze informatie kunnen halen. De manier waarop we deze informatie halen zal afhangen van model tot model daarom zullen we elke mogelijkheid proberen te voorzien tijdens de opzet.

3.1 Apache Cassandra

Apache Cassandra is een opensource, NoSql (Not only Sql) database die de dag van vandaag door vele internationale bedrijven in gebruik is genomen. Netflix, Instagram en het CERN hebben in Cassandra een performante, stabiele databank gevonden, die de beschikbaarheid van hun data garandeert.

Deze garantie wordt geboden door de structuur waarmee Cassandra data beheert. Een databank bestaat uit een collectie van noden die een cluster vormen. Het beheer gebeurt decentraal, elke node heeft dezelfde rol, dit maakt dat wanneer een node offline is, de andere ongehinderd blijven werken. Om het volle potentieel van Cassandra te benutten, is het aangeraden om de noden niet enkel over verschillende servers maar ook over verschillende datacenters te verspreiden. Bij deze setup mag er een datacenter offline gaan, de gebruikers zullen hier geen weet van hebben. En wanneer het datacenter terug online is, zal de data gedeeld worden met de herstelde noden. Dit is ook positief voor de schaalbaarheid. Wanneer een node wordt toegevoegd, haalt deze zijn configuratie op bij een bestaande node en sluit zich naadloos aan in de cluster.

Bij het aanmaken van een cluster kan men een replicatie factor meegeven. Deze geeft aan op hoeveel verschillende noden een record moet bewaard worden. Bij een factor 3 zal elk databaserij op 3 noden beschikbaar zijn. Des te groter de factor, des te groter de performantie en stabiliteit maar ook de duplicatie aan data vergroot mee. Bij het kiezen van de replicatiefactor moet er een afweging gemaakt worden tussen niveau van beschikbaarheid en het extra geheugen dat de replicatie vereist.

En van de grootste verschillen met een standaard relationele databank is dat het linken van verscheidene tabellen niet mogelijk is. In plaats daarvan werkt Cassandra met collecties die rechtstreeks in een kolom worden opgeslagen. Dit wordt mogelijk door het aanmaken van UDTs (User Defined Types). Wanneer een lijst van UDTs (of primitieve types) wordt opgeslagen, wordt deze omgevormd naar een json-achtige structuur. Doordat alle data in n rij zitten, hoeven er geen joins gedaan te worden over meerdere tabellen wat de performantie aanzienlijk verhoogt.

Er zijn ook enkele nadelen verbonden aan het kiezen voor Cassandra als database,

n van deze nadelen is dat het, out-of-the-box, onmogelijk is om tabellen te querien zoals je een SQL zou doen. Dit komt omdat Cassandra werkt met key-value paren om de data op te slagen en dus enkel aan de hand van een key (of deel van een key) naar een record gezocht kan worden. Dit heeft ook als gevolg dat aggregatie functies enkel op partitie niveau kunnen worden uitgevoerd.

Aangezien Cassandra een non-relationele databank is, is een ander nadeel de afwezigheid van transacties op databank niveau. Enkel op rijniveau kan men gebruik maken van lightweight transacties door een conditie toe te voegen voor een schrijf operatie. Hiervoor zou men bv. een version id kunnen toevoegen aan een tabel zodat wanneer men een schrijf operatie wil doen, er gekeken kan worden of de versies nog matchen. Indien dit niet het geval is wordt de operatie niet uitgevoerd en de gebruiker hiervan op de hoogte gebracht.

3.2 Een Cassandra cluster opzetten

Nu we onze databank gekozen hebben, is het tijd dat we deze opzetten. De stappen die nodig zijn, beginnend van niets, om een cluster op te zetten worden in dit segment besproken. We zullen eerst de machines aanmaken, vervolgens Cassandra installeren en als laatste stap onze nodes met elkaar in contact brengen. Voor dit project heb ik gekozen om met 3 nodes te werken, dit om toch enige performantie te hebben zonder het prijzig te maken.

3.2.1 De machines

Als eerste maken we de 3 machines aan waarop we Cassandra zullen draaien. Via de AWS-console navigeren we naar EC2 -¿ Instances waar we de Launch instance kiezen. Het eerste scherm laat ons kiezen welk besturingssysteem we willen gebruiken, hier kiezen we Amazon Linux AMI. In stap 2 bepalen we hoe krachtig onze machines zullen zijn. Het is perfect mogelijk om met de t2.micro (deze is gratis te gebruiken bij een Free Tier) maar met slecht 1 CPU en 1GB RAM heeft deze niet zo veel te bieden. Hier heb ik voor de t2.large geopteerd, deze zal beter aansluiten bij onze behoeften. Stap 3 hoeven we enkel het aantal instances dat we willen aanmaken wijzigen naar 3, de rest laten we ongewijzigd. Stap 4 en 5 slagen we over, deze zijn niet van toepassing. In Stap 6 maken we een nieuwe security group aan die de poorten nodig voor het gebruik van onze databank openzet. Voor onderstaande poorten maken we een Custom TCP Rule en maken we ze compleet publiek (in productie omgevingen is dit ten strengste af te raden.) door de toegestane IP-adressen op 0.0.0.0/0 te zetten. Figuur 1 toont ons hoe we dit bekomen.

Indien deze stap afgehandeld is, krijgen we een overzicht te zien. Als alles naar wens is klikken we op Launch waardoor er een prompt opengaat. Vanuit deze

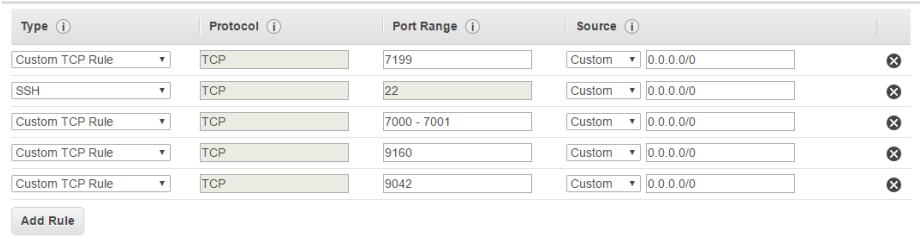

Figure 1: security settings.

prompt kunnen we een pem-file aanmaken en downloaden, het is zeer belangrijk om deze op een veilige plaats op te slaan, deze nodig is om via ssh een connectie naar onze machines te maken. Bij het aanmaken van nieuwe machines kunnen we dezelfde pem-file hergebruiken zodat we n key hebben voor al onze instances. We slagen het bestand op onder aem-ec2.pem.

3.2.2 Cassandra als service

Nu we onze machines hebben is het tijd om hiervan Cassandra nodes te maken. Via het Instance scherm kunnen we achterhalen wat de publieke IP-adressen zijn van deze machines (laten we ervan uitgaan dat deze 1.0.0.1,1.0.0.2 en 1.0.0.3 zijn.). Volgende stap moet op elke machine identiek herhaalt worden buiten de nodige aanpassingen aan de cassandra.yaml. Laten we eerst op onze machines inloggen via het commando:

```
$ ssh  i  aem-ec2.pem ec2-user@1.0.0.1
```

Nu zijn we ingelogd als ec2-user op onze machine. Om Cassandra 3.x te kunnen draaien hebben we Java 1.8 nodig, via het commando java -version kunnen we dit controleren. Indien de versie lager ligt, zijn we verplicht om Java 1.8 te installeren. Een mogelijke manier om dit te doen is als volgt:

We navigeren naar onze jvm folder

```
$ cd /usr/lib/jvm
```

Vervolgens downloaden we de java 1.8 tar file

```
$ sudo wget --no-cookies --no-check-certificate --
    header "Cookie: gpw_e24=http://www.oracle.com/;
    oraclelicense=accept-securebackup-cookie" http://
    download.oracle.com/otn-pub/java/jdk/8u121-b13/
    e9e7ea248e2c4826b92b3f075a80e441/jdk-8u121-linux-
    x64.tar.gz
```

En pakken we deze uit

```
$ sudo tar xzf jdk-8u121-linux-x64.tar.gz
```

Na het uitpakken kunnen we onze jdk registreren als java optie

```
$ sudo alternatives --install /usr/bin/java java /usr/
    lib/jvm/jdk1.8.0_121/bin/java 2
```

Als we nu het volgende commando runnen zien we twee java mogelijke java engines, diegene dat al genstalleerd was en onze nieuwe. Hier kiezen we om onze nieuwe als default te gebruiken.

```
$ sudo alternatives --config java
```

```
$ Enter to keep the current selection[+], or type
    selection number: 2
```

Als alles correct verlopen is zien we nu 1.8 staan wanneer we opnieuw het commando java -version uitvoeren.

Nu volgt de installatie van Cassandra zelf, hierbij is de eerste stap het toevoegen van de datastax.repo zodat we via het yum commando Cassandra kunnen installeren. Als dit gebeurt is kunnen we via yum zowel Cassandra als Nodetool installeren.

Maak het bestand datastax.repo aan.

```
$ sudo touch /etc/yum.repos.d/datastax.repo
```

Maak het bestand datastax.repo aan.

```
$ sudo touch /etc/yum.repos.d/datastax.repo
```

Vul deze met de nodige data

```
$ sudo touch /etc/yum.repos.d/datastax.repo
```

En zorg dat deze conform is aan Figuur 2

```
[datastax]
name = DataStax Repo for Apache Cassandra
baseurl = http://rpm.datastax.com/community
enabled = 1
gpgcheck = 0
```

Figure 2: datastax.repo

Nu kunnen we Cassandra en Nodetool installeren.

```
$ sudo yum install dsc30
```

```
$ sudo yum install cassandra30_tools
```

Als we nu Cassandra zouden opstarten op de 3 machines, hebben we 3 clusters met elks 1 node gemaakt. Vanzelfsprekend is dit niet het gezochte resultaat en willen we 1 cluster met 3 nodes, om dit mogelijk te maken moeten we de aanpassingen maken, getoond in Figuur 3 (locatie: /etc/cassandra/conf/cassandra.yaml).

```
cluster_name: 'Aem-cluster'

seed_provider:
    - class_name: org.apache.cassandra.locator.SimpleSeedProvider
      parameters:
          - seeds: "1.0.0.1,1.0.0.2"

listen_address:
# Het publieke IP-adres van de machine
broadcast_address: 1.0.0.1
# Dit stelt poort 9042 open aan alle thirth party applicaties waaronder
# DevCenter en Java-applicaties
rpc_address: 0.0.0.0
# Default gelijk aan het rpc_address waardoor
# we deze manueel op het publieke IP-adres
broadcast_rpc_address: 1.0.0.1
```

Figure 3: cassandra.yaml

Nu dat onze yamls correct staan kunnen we op onze machines cassandra starten met het commando sudo service cassandra start. Na enkele minuten zien we dan dat de 3 nodes elkaar gevonden hebben via het commando nodetool status.

Nu we onze databank hebben kunnen we onze nodige modellen toevoegen, het gebruikte script kan je terugvinden in de bijlagen. Voor het uitvoeren van deze scripts kan gebruikt maken van DevCenter, een open-source tool waarmee een connectie kan gemaakt worden met een cluster. Deze manier van werken is aangenamer dan telkens te moeten sshen naar een node om daar via het cqlsh commando onze databank te kunnen queryen. Onze drie tabellen die we gaan gebruiken zijn Product, Store en Price. Deze drie vertonen elk een andere graad van dynamiek, gepast voor het onderzoek dat we zullen verrichten. We voorzien telkens ook een log tabel voor deze drie zodat we aan de hand van een datum, die fungeert als ondergrends, de gewijzigde rijen kunnen opvragen. Eenmaal onze databank operationeel is kunnen we beginnen aan de service die deze zal aanspreken.