BigData for DevOps

Meer inzicht met Elasticsearch, Logstash en Kibana (ELK)

Bij het onderwerp 'Big Data' denken veel mensen al snel aan bedrijven als Google, Facebook en Twitter. En inderdaad: met 500 miljoen tweets per dag, 1,3 miljard actieve Facebook-gebruikers of 30 miljard webpagina's om te doorzoeken, gaat het echt om 'Big Data'. Veel kleinere bedrijven hebben zonder het te weten zelf ook 'Big Data' – de logfiles van hun applicaties. In dit artikel lees je hoe ING met Elasticsearch, Logstash en Kibana inzicht in deze gegevens krijgt, en hoe je dit zelf ook kunt doen.

Big Data?!

Bijna elke applicatie genereert logging. Dat kan om heel verschillende soorten gegevens gaan. Uiteraard is er de technische logging: database queries die lang duren, exceptions die optreden, fouten en waarschuwingen. Maar veel applicaties kennen ook andere logging, zoals bijvoorbeeld audit logging. Welke gebruiker heeft welke actie uitgevoerd op welk record? Wanneer en hoe zijn gebruikers ingelogd? Wie heeft de configuratie aangepast? Daarnaast is er operationele monitoring: zijn services beschikbaar, hoe zijn de responstijden en hoeveel geheugen is er gebruikt?

In de praktijk komt al die informatie op een schijf terecht in meerdere logfiles. Maar wie kijkt er dan nog naar? Misschien operations, als ze eraan denken en er rechten voor hebben. Maar hoe dan ook, om logfiles goed te interpreteren, is kennis vereist om ze goed te begrijpen. Als logfiles al regelmatig geanalyseerd worden, dan is dat maar zelden door de programmeurs die de applicatie hebben gebouwd. En juist voor hen is het ook nuttig en belangrijk om deze gegevens ter beschikking te hebben. Een webbased dashboard zou hiervoor een goede oplossing zijn: alle relevante informatie beschikbaar op elke plek, zonder omslachtige inlogprocedures, zodat iedereen real-time inzicht heeft in de actuele status van de productieomgeving.

ELK?!

Zo'n dashboard kun je heel eenvoudig inrichten met Kibana[1]. Kibana is een webbased tool die zeer eenvoudig data uit Elasticsearch[2] kan ontsluiten. Elasticsearch is een gedistribueerde opslag en zoekoplossing, gebaseerd op Lucene. Maar hoe komt de loginformatie in Elasticsearch? Daarvoor is Logstash[3] dan weer een zeer geschikte tool. Laten we ze eens wat nader bekijken.

Elasticsearch

Veel Java-developers kennen Lucene wel – een library voor het opslaan en ophalen van informatie. Lucene wordt met name veel gebruikt voor *full text search*. Informatie staat in Lucene opgeslagen in een document. Lucene is ook de basis voor Elasticsearch, één van de onderdelen van de ELK-stack. Informatie in Elasticsearch is ook in documenten opgeslagen. Elasticsearch is zeer goed schaalbaar, vooral horizontaal (meerdere servers naast elkaar). In tegenstelling tot verticaal schalen (zwaardere servers) heeft dit als voordeel dat meerdere servers op verschillende locaties kunnen draaien, waardoor je beter om kunt gaan met zaken als stroomuitval. Elasticsearch zal zelf de data over verschillende servers verdelen. Wanneer de nodes in verschillende datacenters (of Amazon regions) staan, kan duplicatie eenvoudig worden ingericht zodat alle gegevens op tenminste twee fysieke locaties staan. De back-up van een partitie staat dan altijd op een andere plek dan het origineel.

Een belangrijk verschil met Lucene is dat Elasticsearch een apart proces is dat een REST API aanbiedt om je data in te voeren of op te vragen. Zowel het uitvoeren van een query als het opslaan van nieuwe informatie gebeurt met JSON. Documenten staan in een *index* en alle documenten in een index hebben hetzelfde *type of schema*. Dat schema wijkt

Maarten Mulders
is Software Engineer
bij Info Support

af van traditionele databases: het beschrijft welke velden er tenminste in een document aanwezig moeten zijn en wat de eigenschappen zijn van die velden. Documenten kunnen ook andere velden hebben.

Een voorbeeld van een Elasticsearch document staat in **Listing 1**. We zien dat Elasticsearch het document in een *_index* heeft staan, dat het document een *_id* heeft, en verder zijn er velden die we zelf verzonnen hebben, zoals *application*, *host*, *level* en *message*. De velden *_index*, *_type* en *_id* samen zorgen voor een unieke identificatie van elk document. Het opvragen van documenten kan vrij eenvoudig met HTTP GET, als je *index*, *type* en *id* kent:

```
curl –XGET 'http://localhost:9200/
logs/log/XGZAwM0OSFhDQWDf0–drSA'
```

Gelukkig is er ook een API om mee te querien. Een eenvoudige query kun je uitvoeren met een browser of met curl. Met:

```
curl –XGET 'http://localhost:9200/
logs/log/_search?q=host:vps1459'
```

krijg je bijvoorbeeld alle logregels op waarbij *host* de waarde *vps1459* heeft. Dezelfde query kun je ook als document meegeven in een http POST: bijvoorbeeld

```
curl –XGET 'http://localhost:9200/logs/
log/_search' –d '{ "query" : { "term" :
{ "host" : "vps1459" } } }'
```

Dat is een stuk lastiger om vanuit een web browser te proberen; het voordeel is dat je er veel ingewikkeldere queries mee kan uitdrukken. Gelukkig is er goede tooling om deze query-documenten op te stellen; daarover later meer.

Zowel bij het aanmaken van een index als bij het aanmaken van een document kun je de werking van Elasticsearch behoorlijk beïnvloeden. Eén optie in het bijzonder is handig als het gaat om inzichtelijk maken van operationele logging, en dat is de mogelijkheid om een *_ttl* veld toe te voegen aan je document. Met dit veld geef je aan hoelang een document in de index mag blijven staan. Als de *_ttl* verstreken is, wordt het document vanzelf verwijderd.

Logstash

Allemaal leuk en aardig, maar het analyseren van data is pas mogelijk als we eerst data verzameld hebben. Hoe krijgen we logfiles van een beveiligde productieomgeving naar

```
{
  "_index":"_logs",
  "_id":"XGZAwM0OSFhDQWDf0–drSA",
  "_type":"log",
  "_version":"1",
  "application":"my–app",
  "host":"vps1459",
  "level":"error",
  "message":"hey, an exception occured!"
}
```

Listing 1: Voorbeeld van een JSON document in Elasticsearch

Elasticsearch? De oplossing voor dit probleem heet Logstash. Met Logstash kun je logfiles verzamelen, parsen en bewaren. Dit klinkt misschien vrij eenvoudig, het is in de praktijk heel krachtig.

Het 'verzamelen' betekent bijvoorbeeld dat je meerdere soorten logfiles parallel kunt verwerken. Genereert je applicatie verschillende vormen van logging? Geen probleem, je kunt ze alle drie verwerken en zelfs op verschillende manieren. De meest generieke vorm van input is een reguliere logfile. Maar het is ook mogelijk om gegevens in te lezen via een socket, van *standard-in*, uit de Windows event log, uit syslog of Twitter. Het is zelfs mogelijk een mailbox of IRC-kanaal te monitoren. Bij het verzamelen, kun je ook een codec opgeven. Een codec beschrijft de structuur waarin je data binnenkrijgt – bijvoorbeeld JSON of multi-line berichten, zoals stacktraces.

Het 'parsen' is zo mogelijk een nog krachtigere en flexibelere stap. Parsen wordt gedaan met filters, bewerkingen die je kunt doen op je log events. Het grok filter bijvoorbeeld probeert aan de hand van patterns je logevents op te splitsen in velden. Stel dat je HTTP-server logregels genereert die er uitzien als:

```
55.3.244.1 GET /index.html 15824 0.043,
```

dan kun je met het patroon:

```
%{IP:client} %{WORD:method}
%{URIPATHPARAM:request}
%{NUMBER:bytes}
%{NUMBER:duration}
```

de semantiek toevoegen. Na het uitvoeren van het grok filter met dit patroon heeft je log event extra velden client, method, request, bytes en duration gekregen met corresponderende waardes. Logstash komt standaard al met een uitgebreide set aan grok patterns. Er zijn zo'n 120 patronen meegeleverd, zoals voor standaard logregels van Apache, haproxy, syslog of Nagios.

KORTOM, MET DEZE STACK IS HET VOOR HET DEVOPS TEAM MOGELIJK OM REAL-TIME INZAGE TE KRIJGEN IN DE PRODUCTIE-OMGEVING

Daarnaast kun je eenvoudig zelf patronen toevoegen.

Andere nuttige filters zijn *mutate* (veranderen van tekst), *kv* (tekst opsplitsen in key/value paren), en *prune*. Prune is bijzonder krachtig, omdat het allerlei velden kan verwijderen uit het event. Als je bijvoorbeeld een audit log verwerkt, kan daar gevoelige informatie instaan die niet zomaar op een operationeel dashboard te zien mag zijn. Die kun je in deze stap eenvoudig verwijderen uit een audit log event. Als de 50 ingebouwde filters niet genoeg functionaliteit bieden: één van de filters heet *ruby* en kan willekeurige Ruby code uitvoeren.

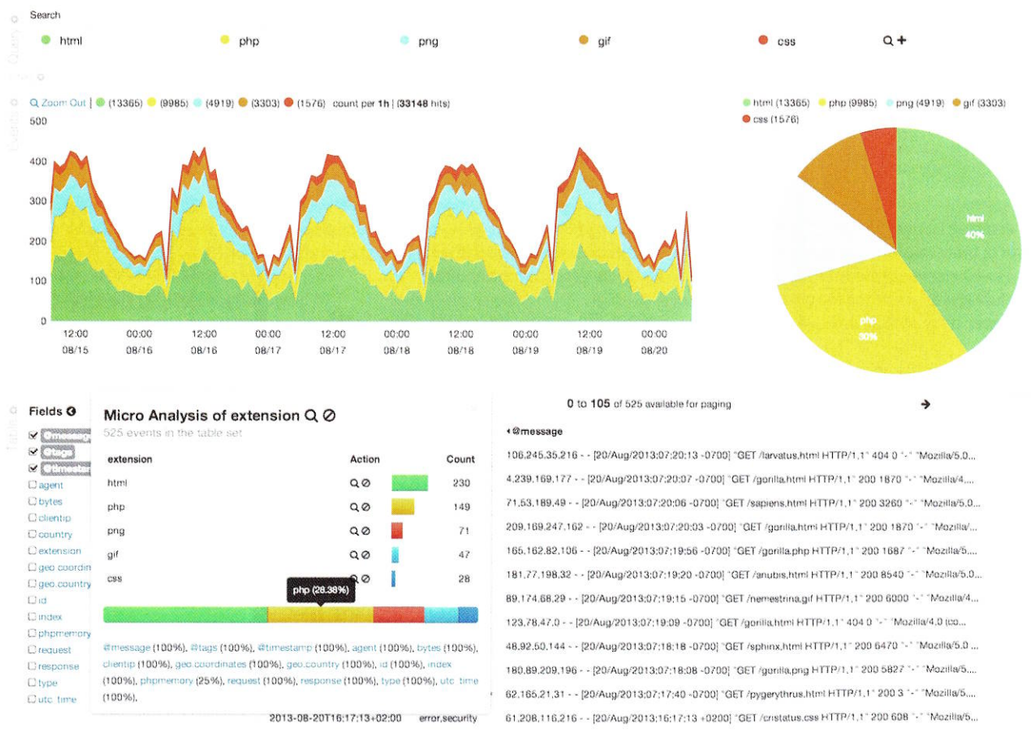
De laatste stap die logstash uitvoert is het 'bewaren' van de verwerkte log events. Logstash ondersteunt 55 manieren om je log events op te slaan of te verzenden. Dat varieert van eenvoudige methodes, zoals wegschrijven naar standard out of file, maar het is bijvoorbeeld ook mogelijk om events in een Mongo database zetten, via Hipchat te versturen, direct in JIRA in te schieten of te e-mailen.

Kibana

Nu hebben we een gevulde Elasticsearch database. De volgende stap is om deze informatie ook nog inzichtelijk maken – en het liefst in een flexibel dashboard. Hier is Kibana een zeer geschikte tool voor, die ook nog eens standaard met Elasticsearch wordt meegeleverd. Kibana sluit goed aan op Elasticsearch en biedt de mogelijkheid om één of meer indices te filteren en te querien. Dat klinken als twee synoniemen, maar in Kibana is dat niet zo.

Met een filter kun je in Kibana aangeven welke gegevens gebruikt moeten worden om je dashboard op te bouwen. Als bijvoorbeeld meerdere applicaties naar dezelfde Elasticsearch instantie schrijven, maar je een operationeel dashboard bouwt voor één applicatie, dan kun je filteren op de naam van die applicatie. Een ander voorbeeld is een filter op het `veld @timestamp`, bijvoorbeeld from: `now – 24h`, to: `now`. Hiermee beperk je jouw dashboard tot het afgelopen etmaal.

Een query in Kibana is in feite een vorm van labelling. Hiermee kun je bepaalde velden of



waardes definiëren die je kunt gebruiken om de verschillende onderdelen van je dashboard mee te vullen. Hierbij kun je denken aan labels als 'error' of 'warning', maar ook labelling naar functionele indeling van je applicatie of naar server in de productie-omgeving is mogelijk. Als je bijvoorbeeld twee servers in productie hebt, kun je eenvoudig de events van beide servers uit elkaar houden door ze een label (en daarmee een kleurtje) te geven.

Naast histogrammen kun je panels met allerlei andere visualisaties kiezen, zoals taartdiagrammen, staafdiagrammen, plots op een kaart, trendanalyses en tabellen. Om een panel van data te voorzien, kies je queries die gebruikt moeten worden als invoer en je kiest hoe de matchende events getoond moeten worden.

Kibana is met name zo'n krachtige tool, omdat het je in staat stelt de complexe queries die nodig zijn voor je visualisaties met een grafische interface samen te stellen. Het eerdere voorbeeld van een JSON-query op Elasticsearch was relatief eenvoudig vergeleken met het soort queries dat Kibana voor je opstelt. Als je dat met de hand zou moeten doen, zou je snel een fout kunnen maken, waardoor je onverwachte of zelfs onjuiste gegevens terugkrijgt. Kibana neemt je deze zorg uit handen.

ING

Tijd voor een praktijkvoorbeeld! ING gebruikt Elasticsearch om een hele keten aan systemen te monitoren. Naast het feit dat elke individuele applicatie Kibana kan gebruiken om de actuele status inzichtelijk te maken, geeft dit ook een goede mogelijkheid om log events uit meerdere applicaties, die betrekking hebben op dezelfde transactie, aan elkaar te relateren. Op die manier ontstaat een soort track-and-trace voor transacties door de hele keten. Om dat mogelijk te maken, schrijven alle applicaties in de keten hun log events naar dezelfde index. Elasticsearch kan dan alle documenten opvragen die daarmee te maken hebben. Omdat er veel systemen binnen de keten zijn die allemaal hun logginformatie naar Elasticsearch sturen – enkele honderden events per seconde – is er een cluster van Elasticsearch servers gemaakt.

Eén van de applicaties in die keten is een component die autorisaties verzamelt waarmee klanten hun transacties bevestigen, bijvoorbeeld een TAN-code. Deze applicatie heeft onder andere een audit log. In deze file zijn details van transacties terug te vinden. Deze informatie is belangrijk voor een bank om te bewaren, maar is niet relevant voor monitoring. Een prima kandidaat dus voor het prune filter in Logstash! Doordat Logstash op dezelfde server draait als de eigenlijke software garandeert dit dat we op een integere manier met de data omgaan.

Met de informatie uit dit systeem is een operationeel dashboard gemaakt in Kibana. In dat dashboard is een verdeling te zien van het soort transacties, de status waarin ze zich bevinden en hoe de load verdeeld wordt over de verschillende servers in productie. Daarnaast worden waarschuwingen en fouten gevisualiseerd. Dankzij dit dashboard is in één oogopslag te zien dat het systeem stabiel draait.

Kortom, met deze stack is het voor het DevOps team mogelijk om real-time inzage te krijgen in de productieomgeving. Geen handmatige acties meer en niet meer door te lange log files bladeren. In één oogopslag weten dat alles goed draait. Ook dat is Big Data! ■

REFERENTIES

1 http://bit.ly/1dtvp53
2 http://bit.ly/1oQJO3T
3 http://bit.ly/1wCBbJ9