Amazon Elastic Search Use Cases

Logs analytics at Expedia Using Amazon Elastic Search

Speaker: Kuldeep, @this_is_kuldeep

In 2017 at Expedia we have

- + 150 Clusters (different sizes)
- + 450 EC2 (different sizes)
- + 30TB of data (not more than 3 days)
- + 30B documents

With a big infrastructure like that, it is difficult to monitor and quickly find a cause of failure. But since every service generate logs, how can we use these logs effeciently ?
For that, we decided to use Amazon Elastic Search.

Why did we choose Amazon ES ?

- It’s open source (Elastic Search)
- User-friendly console
- Easy to set up

Also AWS offers

- High availability
- Flexible storage options
- VPC
- High Security
- Monitoring with (CloudWatch)
- Backups

Before AWS our infrastrucutre wasn't fully automated

Different Log Analytics Architectures

1. Docker startup logs to Elasticsearch

> ***Problem**:*
>
> *We want to understand why a particular service is not starting.*

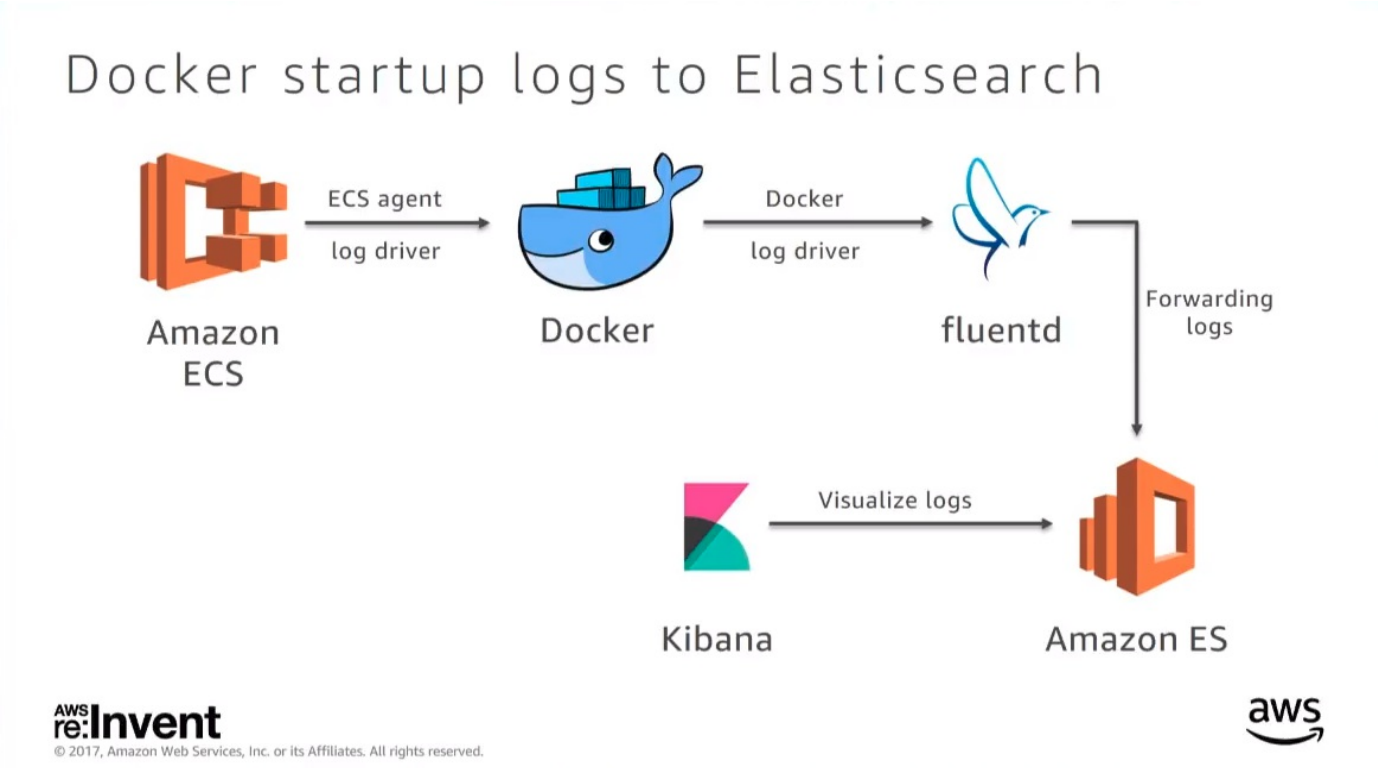
At one moment, we moved from EC2 to ECS, mainly because our microservices number started to explode.
Before Docker 1.7, The logs of Docker were printed in the console and not anywhere else but we needed these logs to be able to do some analytics.
Solution: write the logs in a file and put this file to the cluster (s3).

Docker 1.7 introduced *docker logger* which is a streaming option for

docker logs

From now then we send the logs directly to a streaming service - here we use **fluentd**.

Architecture:



This image explains how we get and process our logs.

When we start a cluster on ECS, in the back-end it start the docker containers and these containers through docker logger stream the logs to fluentd; fluentd forward these logs to Amazon ES for processing and we visualize the results with Kibana.

From these results, when a container failed to start, we automated an event by sending the Kibana url and all the description to the operator.

message example: fluentd log_driver configuration

```
{
```

```
    "log_driver": "fluentd",
    "options": {
        "fleuntd-adress": "<fluentd>:24224",
        "tag": "#{ImageNam}"
    }
}
```

The tag is the docker container tag.

Example of fluentd configuration to receive Docker logs

```
<source>
    @type forward
    port 24224
    bind 0.0.0.0
</source>
    <match *.**>
    @type copy
```

fluentd to Amazon ES

The logs are just forwarded to the elastic search domain

```
<match *.**>
    @type copy
    <store>
        @type elasticsearch
        host <elasticsearch domain>
```

```
        include_tag_key true
        tag_key @log_name
        flush_interval 1s
    </store>
```

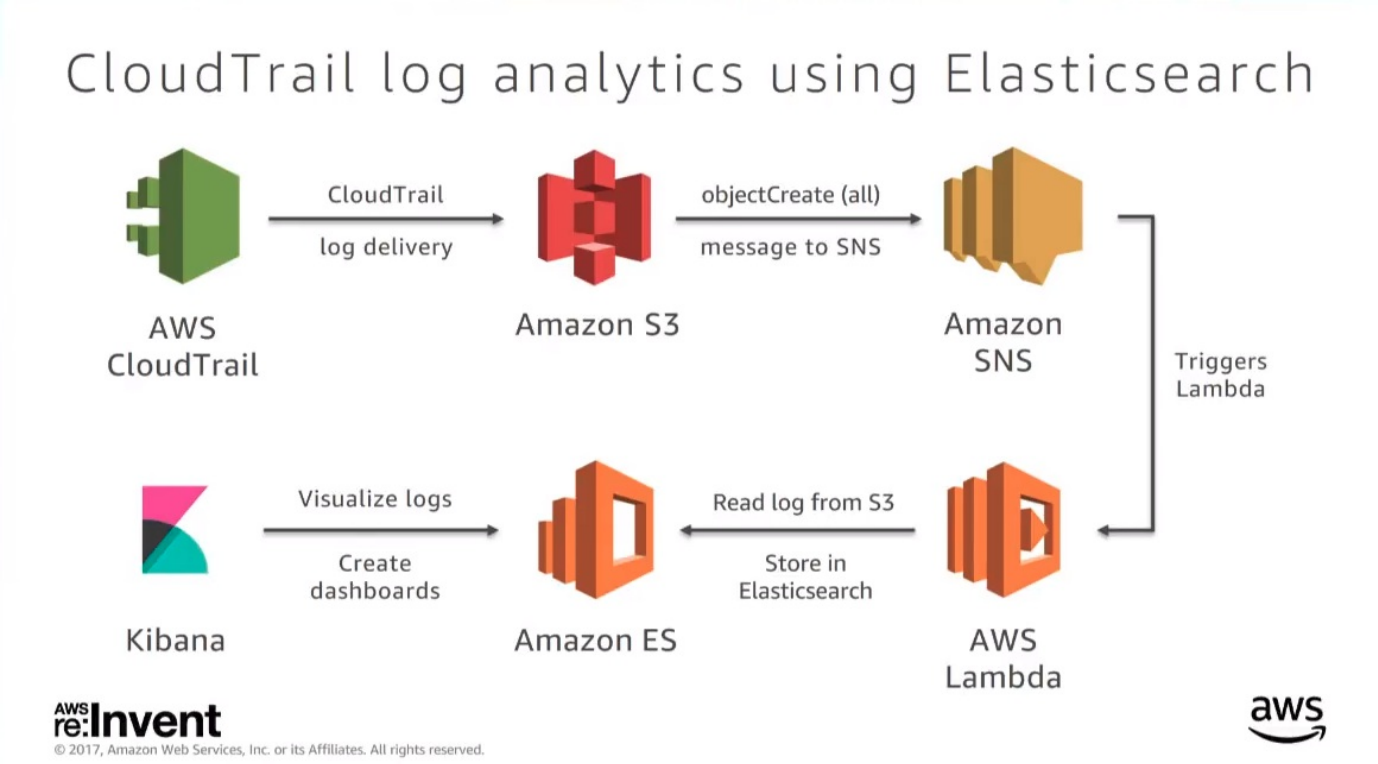
2. cloudtrail log analytics using Amazon ES

CloudTrail is an AWS service which records all the api calls made to AWS services and saves them in a s3 bucket.
From theses logs, you can do any processing you can think of. In fact the logs keep trace of *“who, where, when, what”* of every api call.

> ***Example of use case at Expedia**:*
> *Can be used for security purposes: For eg. if an unknown IP address makes a request to your AWS account, you can automatically find it out by analysin these cloudtrail logs.*

> ***Problem**:*
> *Expedia owns 3 AWS accounts and lot of services (ECS, EBS, EC2, etc) on each of them.*
> *Expedia would like to know who is making an api call and on which account and other statistics.*

Architecture:



Idea:

Cloud Trail generates the logs and saves them to a s3 bucket. From this s3 bucket we set up a SNS listener which triggers at every write. Then a Lambda (AWS Lambda) function is linked to this SNS listener. The lambda function does only one thing which is to forward the data to Amazon ES.

The Lamnda function's code

```
try:
    response = s3.get_object(Bucket=s3Bucket, key=s3Object
Key)
    content = gzip.GzipFile(fileobj=StringIO(response['bod
y'].read())).read()
for record in json.loads(content)['Records']:
    recordJson = json.dumps(record)
```

```
    logger.info(recordJson)
    indexName = 'ct-' + datetime.datetime.now().strftime("
%y-%m-%d")
    res = es.index(index=indexName, doc_type='record', id=
record['eventID'], body=recordJson)
    logger.info(res)
return True
```

Results

A dashboard with the top 10 api calls within every 10 mins.

Expedia made this solution available on their github repository.

3. CI/CD platform KPIs using Amazon ES

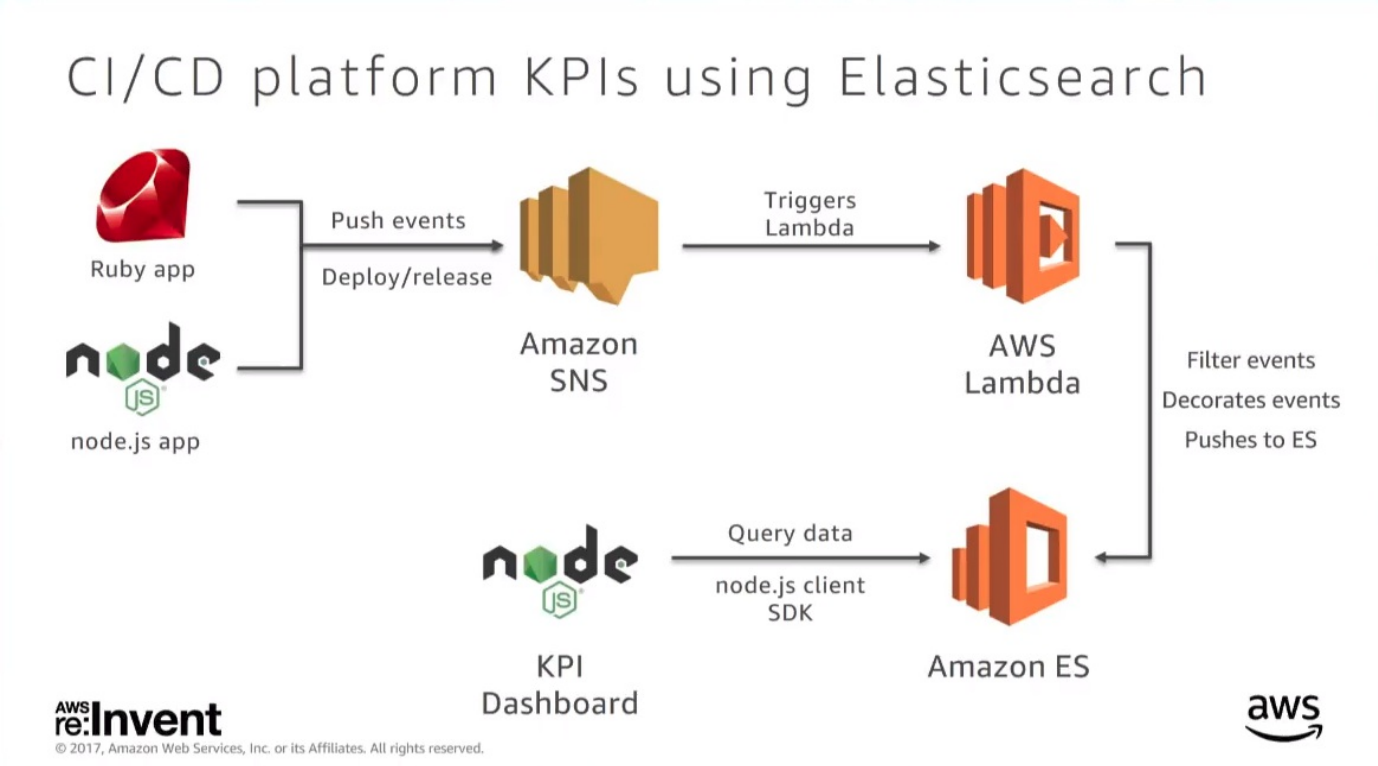
We use Primer for managing the microservices

- +1500 deployments/day all environments
- +300 on production

> ***Problem**:*
> *Why a particular service is failing?*
> *And for any failure, we would also like to figure out of its generated by a user or by the platform.*

The platform is built on top of *Ruby* and *Node.js* and both send the same event notifications (push event: deploy or realease) to SNS. Once the SNS catches an event, a lambda function is triggered, then fetch the data and performs some preprocessing.

Architecture:



4. Distributed tracing platform using Amazon ES

Several companies are moving from monolith architecture to microservices architecture. From that you need a way to trace how a particular request traverses your system.

> ***Problem**:*
> *How to trace a particular request traversing a distributed (microservices) system?*

Idea

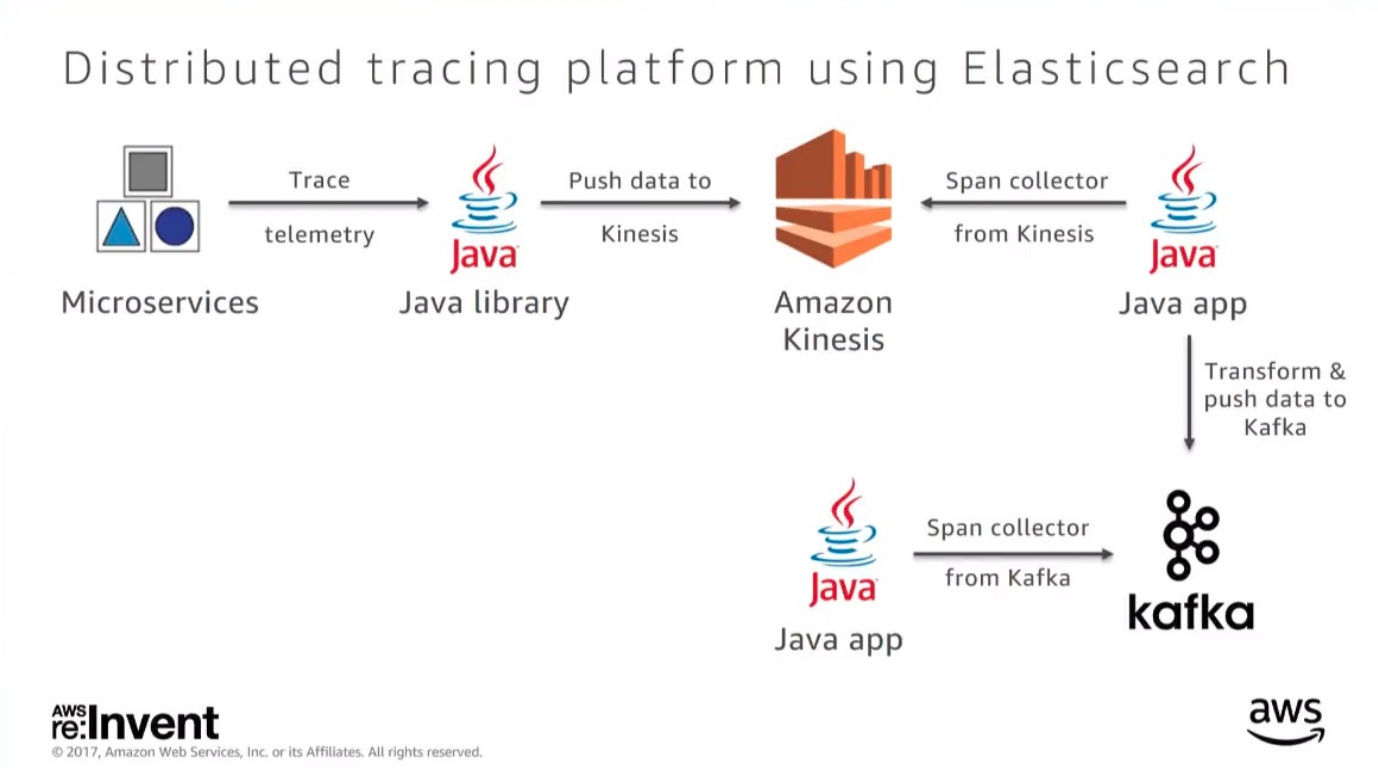
Use a distributed tracing system.

At Expedia we use apache zipkin.

Apache Zipkin is open source and build based on google apper.

For every web request, Zipkin can trace its span (how long it takes). The trace contains other information as well.

Architecture



In this architecture we attached only one consumer to Amazon Kinesis because with more consumers amazon kinesis became expensive. This single consumer transforms and pushes the data to an apache kafka server to which many consumers are attached.\

5. Hotel image metadata repository using Amazon ES

In this last case, Amazon ES is used as a ametada repository for hotels

images.

For an image we have some metadata including:

- URL
- Latitude, Longitude
- Width
- Height
- Labels
- Google Vision tags
- Etc.

Once the data is in Amazon Elastic Search, we create a dashboard where we can find for example the number of image with a given google vision tag (eg: room with bed).

Things to keep in mind

- Scaling of cluster results in a new cluster with the data beign synchronized
- Monitor and optimize the cluster yourself (Don’t just go by the default, optimize it for your need)

You can find the re:Invent video here.

If you want to know more about AWS, don’t hesitate to go on AWS Educate and Qwiklabs for great tutorials labs.

Blogger:

Chriss Santi

CS Master student at the RUG University

Email: osler.santi@gmail.com

Github: github.com/anostdev

PS: This article is a derived from the presentation made by Kuldeep during the Amazon re:Invent 2017

English is not my first language ;) sorry for the mistakes – improvements will follows as I write articles :p