

AWS CLOUD

SEOUL - KOREA



게임 서비스 품질 향상을 위한 데이터 분석 활용

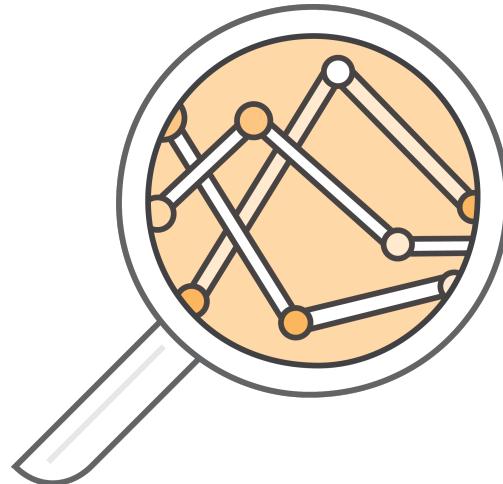
김필중 | Solutions Architect, AWS Korea
이의석 | Data Engineer, Bluehole

목차

- AWS와 데이터 분석
- 인게임 이벤트 시스템 설계
- 고객사례: 테라 렉 로그 분석 이야기 (블루홀)

더 나은 사용자 경험을 위한 데이터 분석

- 게임 서비스는 항상 데이터와 로그를 생성하고 저장함
- 데이터 포인트와 로그들은 사용자 경험을 이해하고 향상시켜 줌
- 로그 분석은 게임 서비스에 대해 통찰력을 가져다 줌



데이터 분석의 실시간성

데이터 분석

- 어제 아이템을 구매한 유저의 수는?
- 어느 맵에서 플레이시간이 제일 길었는지/짧았는지?
- 전체 유저중 PVP 모드를 즐기는 유저의 %?
- 어느 캐릭터가 제일 인기가 있는지?

실시간 데이터 분석

- 현재 비정상적인 전투 양상을 보이는 맵은?
- 현재 비정상적인 사용 패턴을 보이는 유저가 있는지?
- 마케팅 이벤트가 기대한 만큼의 유저를 끌어모으고 있는지?
- 유저들의 지금 현재 사용 패턴은?



COPACABANA

POWER

4



A TOUGH NUT TO CRACK

USE RATCHET'S DETONATORS TO MINE FOR SILICATE

• 0/10 SILICATE FRAGMENTS COLLECTED

- FIND THE SILICATE DEPOSIT FIELD



1200



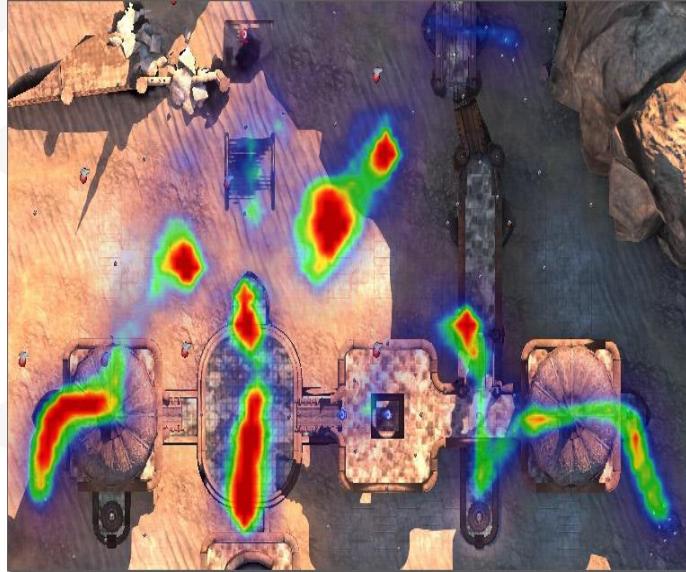
100 %

09:46PM

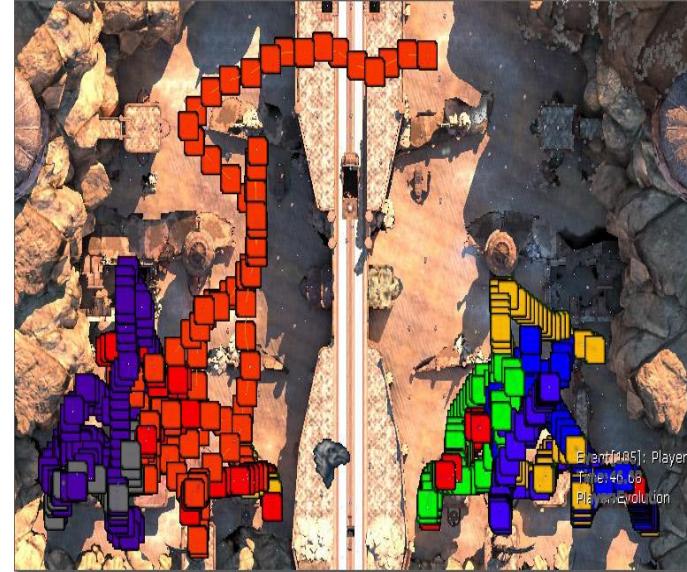


실시간 데이터 분석

예전 방식 게임 후 격전 지도

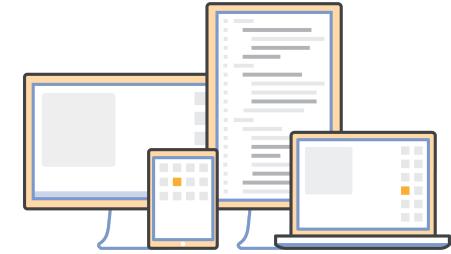
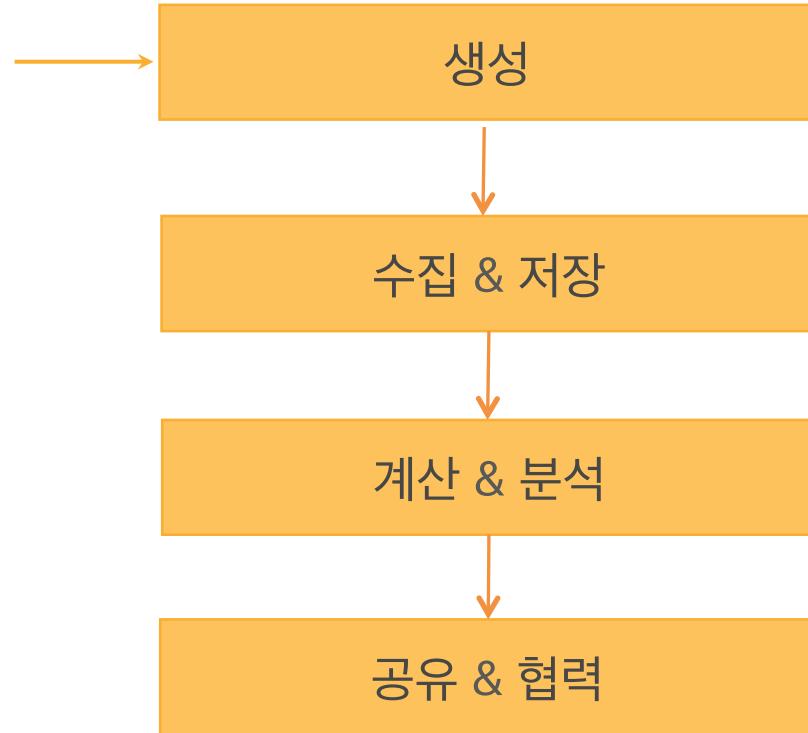


새로운 방식 실시간 격전 지도

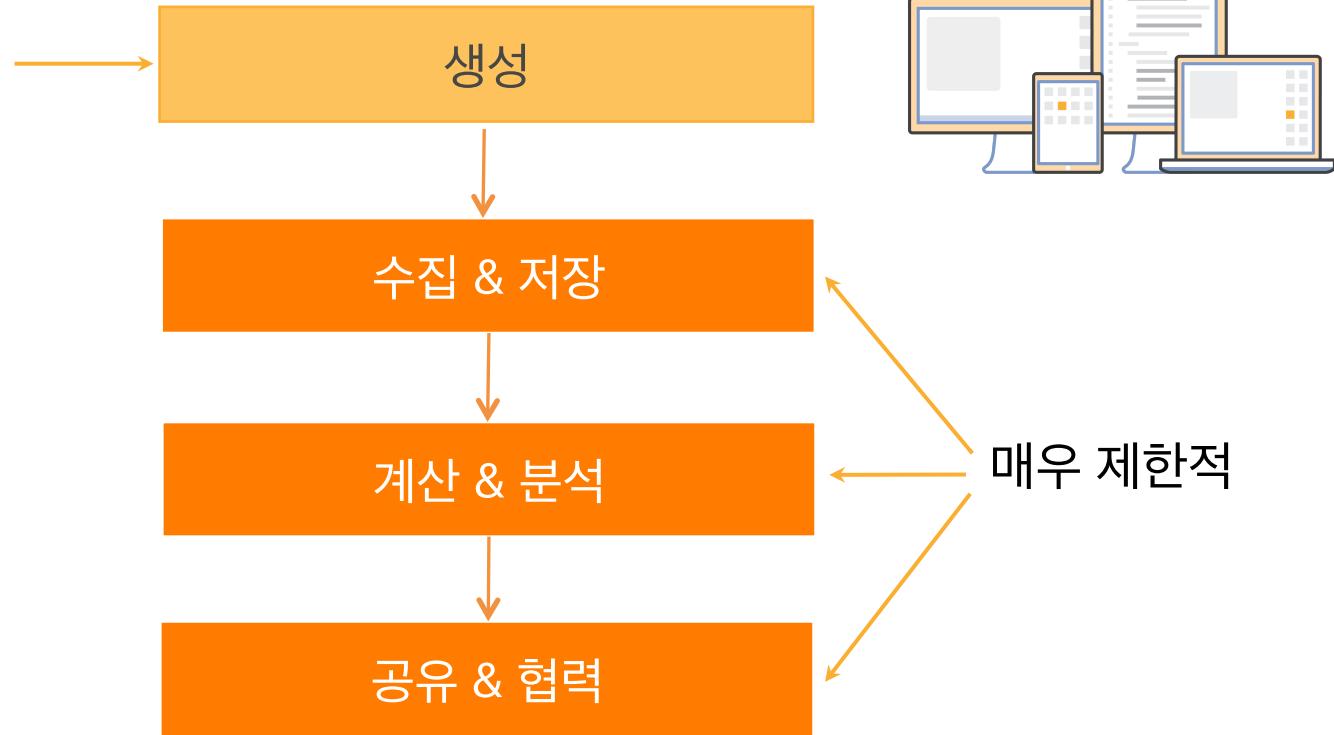




많은 기기들
낮은 비용
높은 처리량



많은 기기들
낮은 비용
높은 처리량



Amazon Web Services 는 제한을 제거하는데에 도움을 줍니다

데이터 분석과 AWS 클라우드

데이터 분석

- 방대한 데이터 집합
- 데이터 분석 인프라 사이징의 어려움
- 일정한 수준의 부하가 지속되지 않고, 등락이 큼
- 다양한 형태의 정형 및 비정형 데이터의 조합

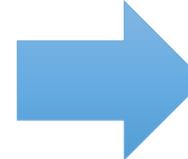
AWS 클라우드

- 사실상 거의 **무제한의 용량**
- **온 디멘드 인프라**를 통해 비용 효율적
- **매우 다양한 부하**를 위한 매우 우연한 인프라
- **정형 및 비정형, 스트림** 데이터를 관리하기 위한 도구 및 서비스들

고객 피드백 기반 클라우드 서비스 혁신

고객의 목소리

- 데이터베이스 관리의 부담이 많습니다.
- 관계형 DB는 확장성이 쉽지 않아요.
- Hadoop 배포 및 관리하기가 힘듭니다.
- 기존 DW는 복잡하고 비싸고 느립니다.
- 상용 DB는 고비용에 관리, 확장이 어려워요.
- 실시간 데이터는 수집하고 분석하기 힘듭니다.



AWS는 만들었습니다

- ✓ **Amazon RDS**
- ✓ **Amazon DynamoDB**
- ✓ **Amazon EMR**
- ✓ **Amazon Redshift**
- ✓ **Amazon Aurora**
- ✓ **Amazon Kinesis**

데이터 분석 파이프라인 간소화



수집



Amazon Mobile Analytics



AWS Import/Export
Snowball



Amazon Kinesis
Firehose



저장



Amazon RDS



Amazon
EFS



Amazon
DynamoDB



Amazon
Elasticsearch
Service



Amazon S3



Amazon
Glacier

처리



Amazon EMR



Amazon Machine
Learning



Amazon Redshift



Amazon
Lambda



Amazon
Kinesis
Analytics



Amazon
EC2

시각화



New

Amazon QuickSight



AWS + 데이터 분석 = 게임 서비스 품질 향상

쉽게 적용 가능한 곳: 메타 게임 기능!

- 일일 로그인 보상
- 자막 뉴스
- 리더보드
- 이벤트 시스템

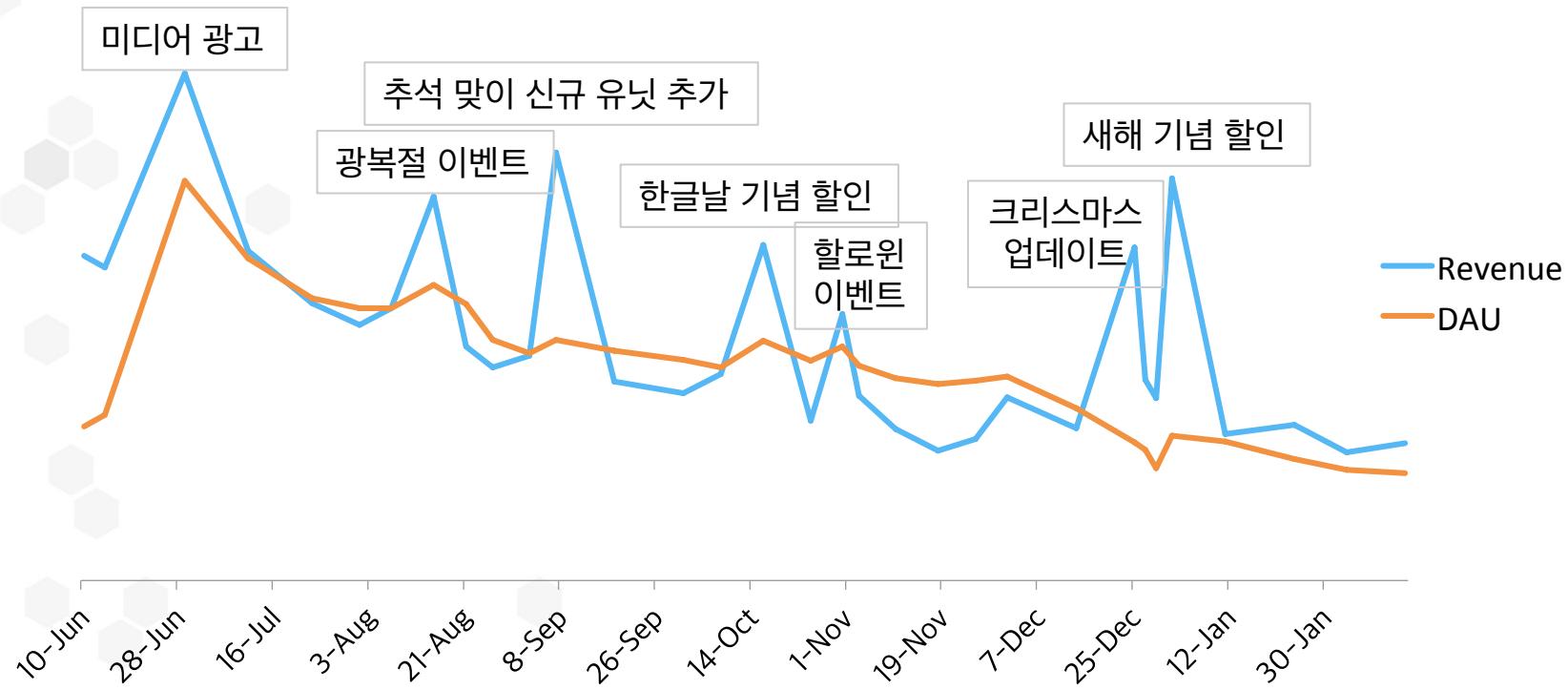
AWS CLOUD

2016년 1월 7일 | SEOUL - KOREA

인게임 이벤트 시스템 설계



왜 인게임 이벤트가 중요한가?



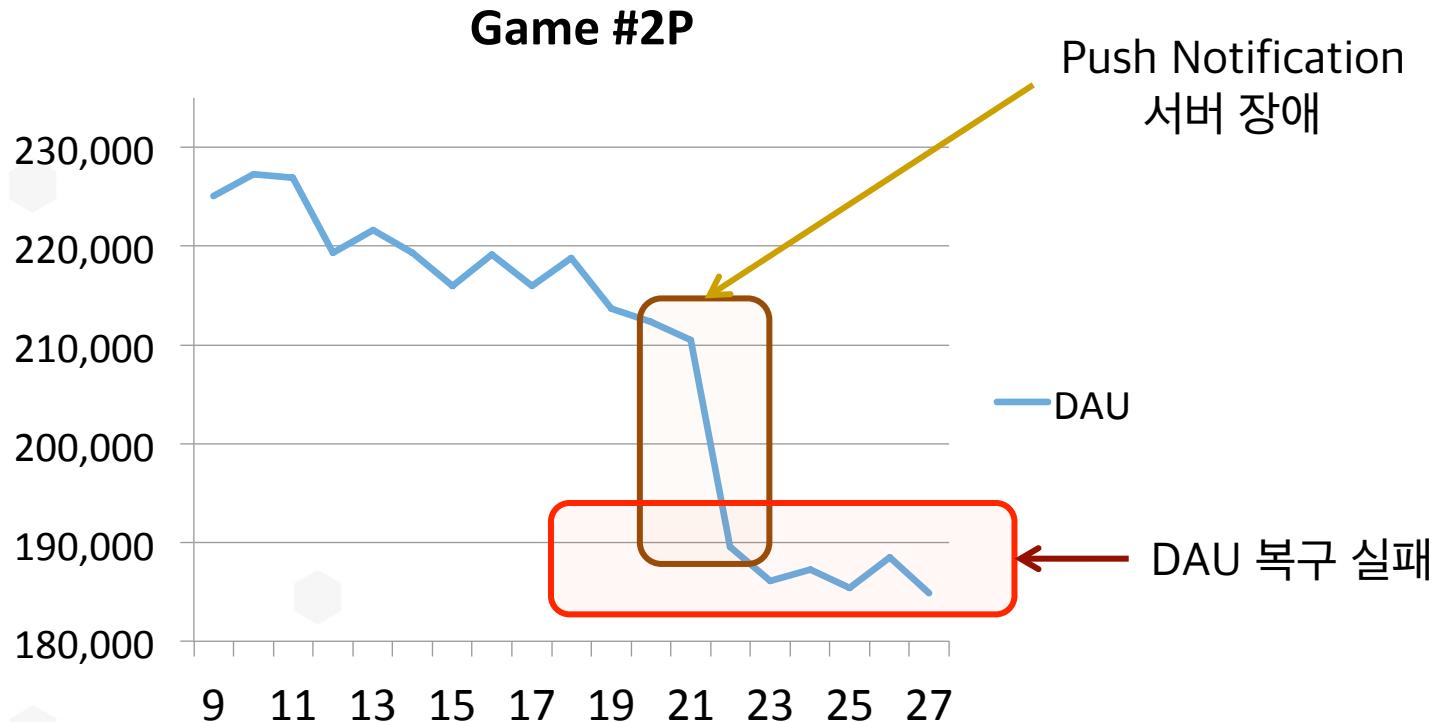
장기적인 리텐션의 중요성

Top 10 Multiplayer Games	% Buyers	ARPPU	Avg Tx Size	Avg # Tx	ARPU	% of Players	% of Revenue
1 play	0.03%	\$6.98	\$5.02	1.39	\$-	45.5%	0.1%
2 to 10	0.40%	\$11.01	\$6.63	1.66	\$0.04	40.3%	0.9%
11 to 50	4.93%	\$19.82	\$7.92	2.50	\$0.98	7.7%	3.9%
51 to 100	11.14%	\$33.14	\$8.97	3.70	\$3.69	2.3%	4.3%
101 to 250	17.11%	\$63.12	\$11.11	5.68	\$10.80	2.2%	12.0%
251 to 500	26.94%	\$123.92	\$14.09	8.80	\$33.38	1.0%	16.9%
500+	39.39%	\$270.58	\$19.03	14.22	\$106.58	1.1%	62.0%
Total	1.89%	\$102.74	\$15.03	6.84	\$1.95	100.0%	100.0%

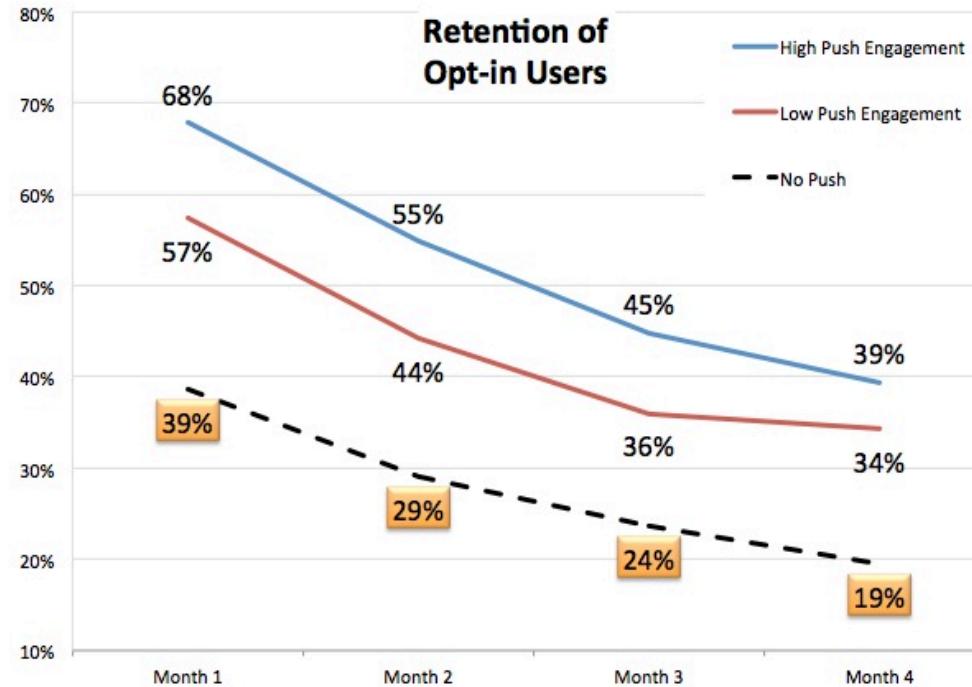
- 거의 대부분의 수입은 적어도 100번 이상 플레이를 한 유저로부터 발생
- 게임 매출 향상을 위해 할 수 있는 최선의 방법은 장기적인 리텐션에 집중하는 것일 것

Source: kongregate

고객 사례: 리텐션 하락



푸시 메시지 효과



Source: Urban Airship

시스템 특징 및 요구사항

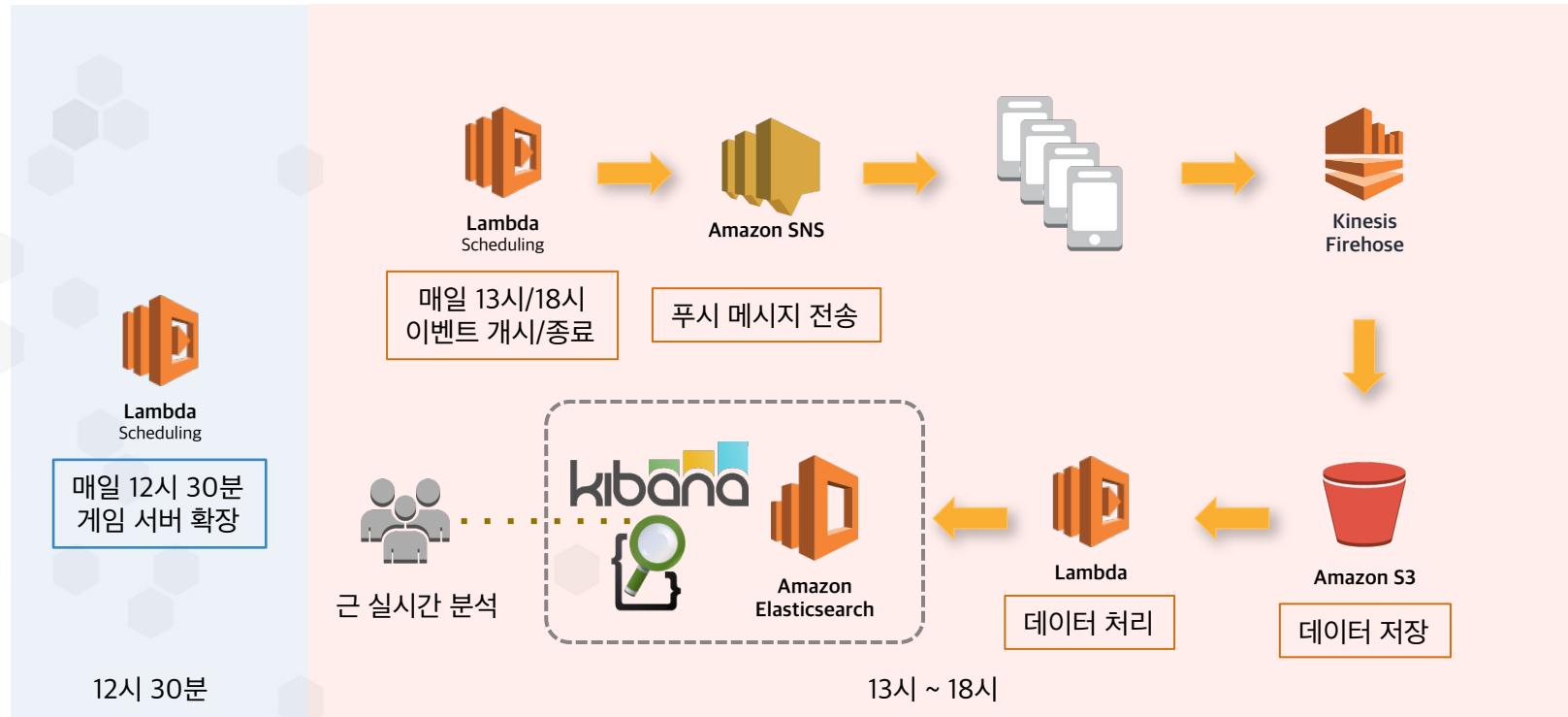
스테이지 방식의 캐주얼 게임이라 가정.

이벤트는 특정 스테이지에서 아이템 획득률 상승 및 스테이지 아이템 할인, 난이도 상승

- 매일 혹은 일정 주기로 반복 또는 특정일에 이벤트 진행
- 사용자에게 시작시간과 얻을 수 있는 혜택 등의 정보를 알려주어야 함
- 사용자의 순간적인 접속 증가로 인해 인프라에 높은 부하를 야기할 수 있음
- 주기적으로 반복적인 개발 혹은 운영 업무를 진행하여야 함
- 게임 진행 상황 및 사용자의 행동을 분석하여 다음 이벤트에 반영하여야 함

인게임 이벤트 시스템 아키텍처

13시/18시 이벤트 개시/종료 -> 푸시 메시지 전송 -> 데이터 수집 -> 저장 -> 처리 -> 근 실시간 분석



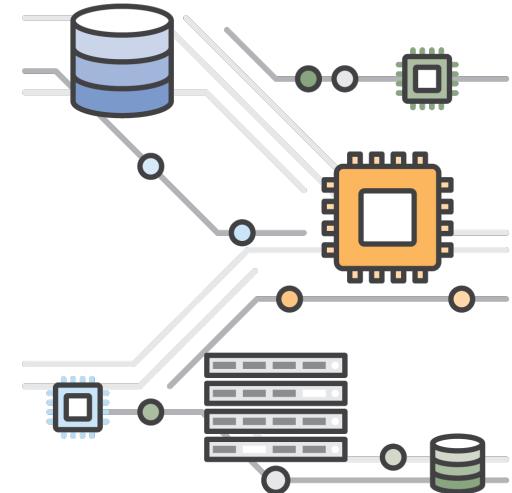
인게임 이벤트 시스템 아키텍처

13시/18시 이벤트 개시/종료 -> 푸시 메시지 전송 -> 데이터 수집 -> 저장 -> 처리 -> 근 실시간 분석



AWS Lambda

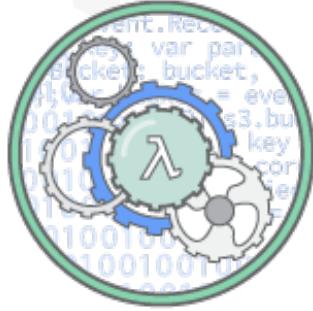
- 이벤트에 대한 응답으로 작성한 코드를 실행하는 컴퓨팅 서비스
- 이벤트 트리거들:
 - 비동기적 혹은 동기적인 직접 호출을 통해
 - Amazon S3 버킷으로의 오브젝트 업로드를 통해
 - API Gateway 엔드포인트 호출을 통해
 - Amazon SNS 메시지를 통해
 - 그 이외에도 다양한 방법으로
- 사용처:
 - 데이터 기반의 감사, 분석 및 알림 등을 수행할 때
 - 자동으로 확장하는 백엔드 서비스를 만들 때



AWS Lambda 의 이점

1

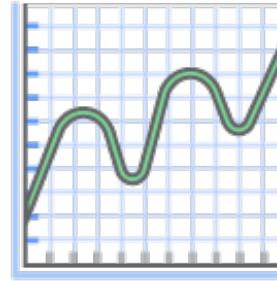
관리할 서버 없이



서버를 준비하거나 관리할 필요
없이 코드를 자동으로 실행.
코드를 작성하여 업로드만
하면 작동함

2

지속적으로 확장



코드가 각 트리거의 응답에
따라 실행되어 자동으로 확장
코드는 병렬로 실행하고
개별적으로 트리거를 처리,
워크로드의 볼륨에 정확하게
맞춰 확장

3

1초 이하의 미터링



매 100ms의 코드 실행
시간과 코드가 실행된
횟수에 따라 비용이 발생
코드가 실행되지 않을 땐
그 어떤 과금도 발생 안함



AWS

Services

Edit

Piljoong Kim | Tokyo | Support

Lambda > New function using blueprint hello-world-python

Step 1: Select blueprint

Step 2: Configure function

Step 3: Review

Configure function

A Lambda function consists of the custom code you want to execute. [Learn more](#) about Lambda functions.

Name* Description Runtime*

Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need custom libraries, you can upload your code and libraries as a .ZIP file.

Code entry type Edit code inline Upload a .ZIP file Upload a .ZIP from Amazon S3

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13    return event['key1'] # Echo back the first key value
14    #raise Exception('Something went wrong')
```



Lambda function handler and role

Handler* 

언어

Description A starter AWS Lambda function.

Runtime* Python 2.7

Lambda function code

Provide the code for your function. Use the editor if your code does not require custom libraries (other than boto3). If you need custom libraries, you can upload your code and libraries as a .ZIP file.

Code entry type Edit code inline Upload a .ZIP file Upload a .ZIP from Amazon S3

```
1 from __future__ import print_function
2
3 import json
4
5 print('Loading function')
6
7
8 def lambda_handler(event, context):
9     #print("Received event: " + json.dumps(event, indent=2))
10    print("value1 = " + event['key1'])
11    print("value2 = " + event['key2'])
12    print("value3 = " + event['key3'])
13    return event['key1'] # Echo back the first key value
14    #raise Exception('Something went wrong')
```

코드



```
11     print("value2 = " + event['key2']))
12     print("value3 = " + event['key3']))
13     return event['key1'] # Echo back the first key value
14     #raise Exception('Something went wrong')
```



Lambda function handler and role

Handler*

lambda_function.lambda_handler



Role*



Suggested role: Basic execution role

Ensure that popups are enabled to create a new role. [Learn more](#) about Lambda execution roles.

Advanced settings

These settings allow you to control the code execution performance and costs for your Lambda function. Changing your resource settings (by selecting memory) or changing the timeout may impact your function cost. [Learn more](#) about how Lambda pricing works.

Memory (MB)*

128



Timeout*

0

min

3

sec

Cancel

Previous

Next

* These fields are required.

메모리 /
실행시간

게임 서버 확장 Lambda 함수

```
1 var aws = require('aws-sdk');
2 var ec2 = new aws.EC2({
3     region: 'ap-northeast-1'
4 });
5 var runInstanceParams = {
6     ImageId: 'ami-383c1956', // Amazon Linux AMI HVM EBS
7     InstanceType: 'm3.xlarge',
8     MinCount: 3,
9     MaxCount: 5,
10    Monitoring: {
11        Enabled: true
12    }
13 };
14
15 exports.handler = function(event, context) {
16    ec2.runInstances(runInstanceParams, function(err, data) {
17        if (err) {
18            context.fail(err);
19        } else {
20            context.succeed(data);
21        }
22    });
23};
```

Lambda 함수 스케줄링

Add event source

Configure your Lambda function to respond to events from the event sources listed below. You may also call your Lambda function directly using the AWS mobile SDK for [Android](#) and [iOS](#).

Event source type

Scheduled Event



Name

RunServersForEvent



Description

Run servers for an event



Schedule expression

cron(0 12 ? * MON-FRI *)



Can either be a valid cron expression, such as "cron(0 17 ? * MON-FRI *)", or a fixed rate expression, such as "rate(5 minutes)". Cron expressions are in UTC.

Lambda will add the necessary permissions to invoke your Lambda function on behalf of the event source. Lambda uses the AWS Identity and Access Management (IAM) permissions model.

Enable event source

Enable now

Enable later

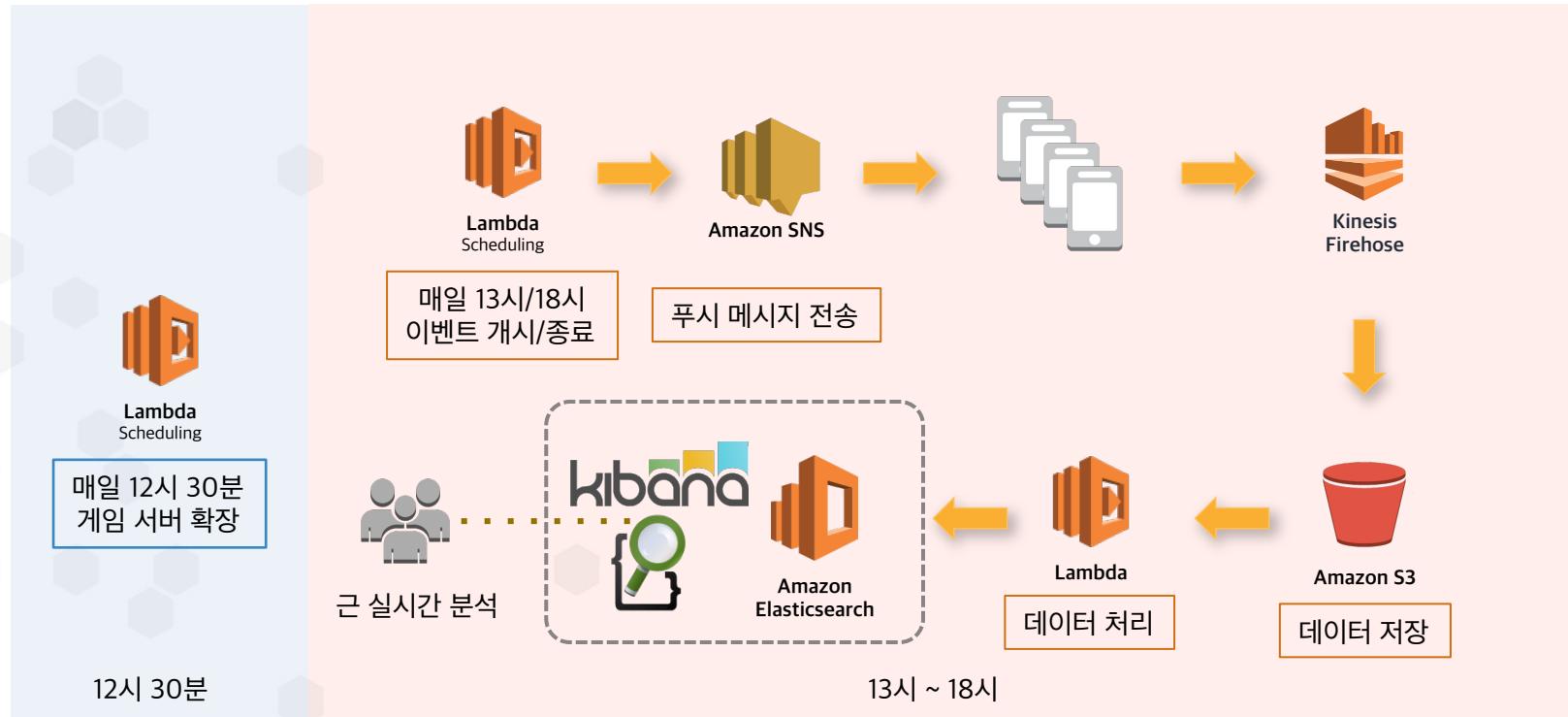


Cancel

Submit

인게임 이벤트 시스템 아키텍처

13시/18시 이벤트 개시/종료 -> 푸시 메시지 전송 -> 데이터 수집 -> 저장 -> 처리 -> 근 실시간 분석



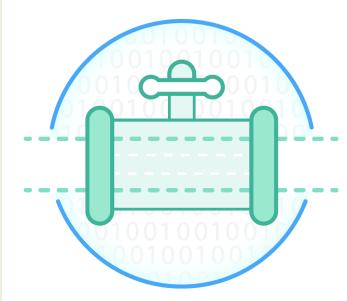
인게임 이벤트 시스템 아키텍처

13시/18시 이벤트 개시/종료 -> 푸시 메시지 전송 -> 데이터 수집 -> 저장 -> 처리 -> 근 실시간 분석



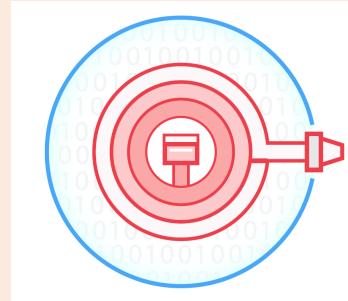
Amazon Kinesis: 스트리밍 데이터를 쉽게 이용

AWS에서 실시간 데이터 스트림 관련 작업을 쉽게하는 서비스



Amazon Kinesis Streams

스트리밍 데이터를 처리하거나 분석하는 커스텀 애플리케이션을 개발



Amazon Kinesis Firehose

방대한 볼륨의 스트리밍 데이터를 Amazon S3나 Redshift로 쉽게 로드



Amazon Kinesis Analytics

표준 SQL쿼리를 이용하여 데이터 스트림을 쉽게 분석

(출시예정)

Amazon Kinesis Firehose

방대한 양의 스트리밍 데이터를 Amazon S3 나 Amazon Redshift 로 쉽게 로드

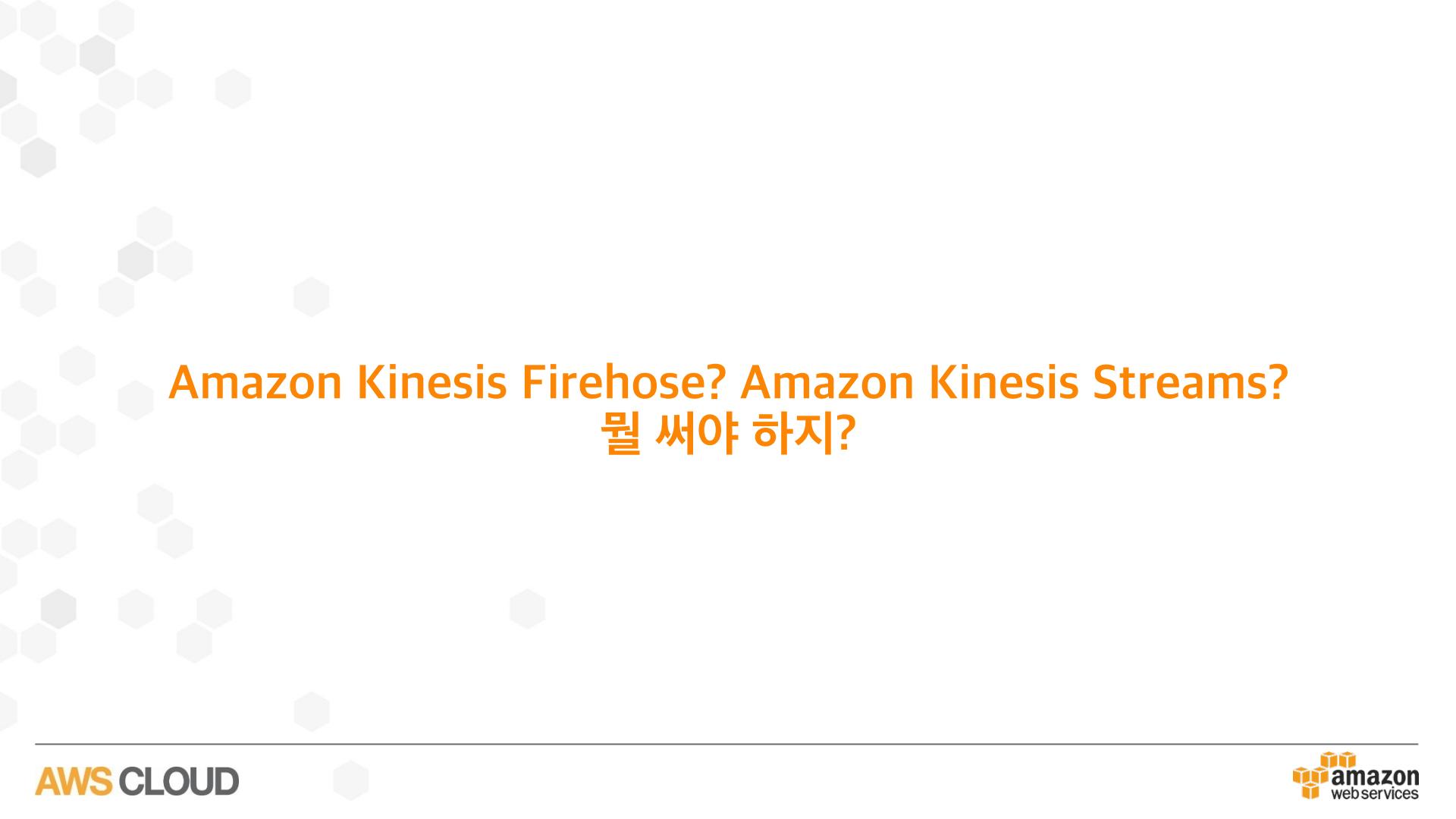


Firehose로 스트리밍 데이터
를 캡처하고 전송

Firehose는 지속적으로 S3 또는 Redshift로
스트리밍 데이터를 로드

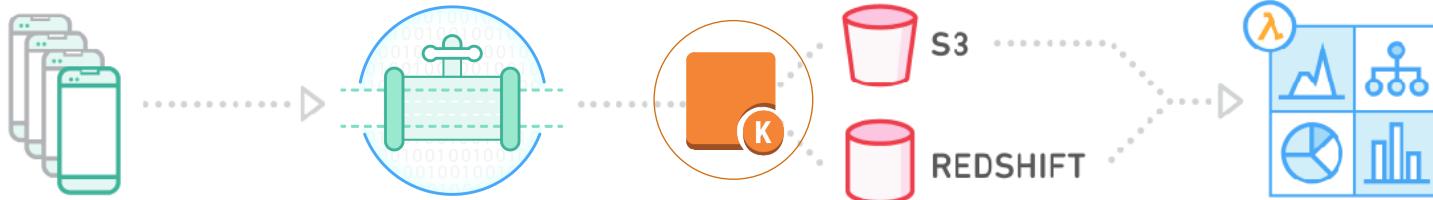
선호하는 BI 툴들을 사용하여 스트리밍 데이터를 분석

- **Zero administration:** 애플리케이션 개발 및 관리 인프라 없이 S3 또는 Redshift로 스트리밍 데이터를 캡처하여 전송
- **Direct-to-data store integration:** 간단한 설정만으로 스트리밍 데이터를 일괄처리, 압축, 암호화하여 거의 60초 이내에 목적지에 전송
- **Seamless elasticity:** 특별한 개입없이 데이터 처리량에 맞는 원활한 확장



Amazon Kinesis Firehose? Amazon Kinesis Streams? 뭘 써야 하지?

Amazon Kinesis Streams 와 Firehose



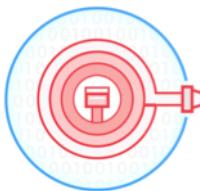
Amazon Kinesis Streams 는 1초 이하의 처리 지연으로 스트리밍되는 각 레코드마다
커스텀 처리를 수행하고 선택적으로 스트림 처리 프레임워크들을 사용할 수 있는
워크로드에 맞는 서비스

Amazon Kinesis Firehose 는 관리 운영 없이 **S3 또는 Redshift** 를 기반으로하는
기존 분석 툴들을 그대로 사용할 수 있고 60초 또는 그 이상의 데이터 지연을 요구하는
워크로드에 맞는 서비스

Amazon Kinesis Streams 와 Firehose



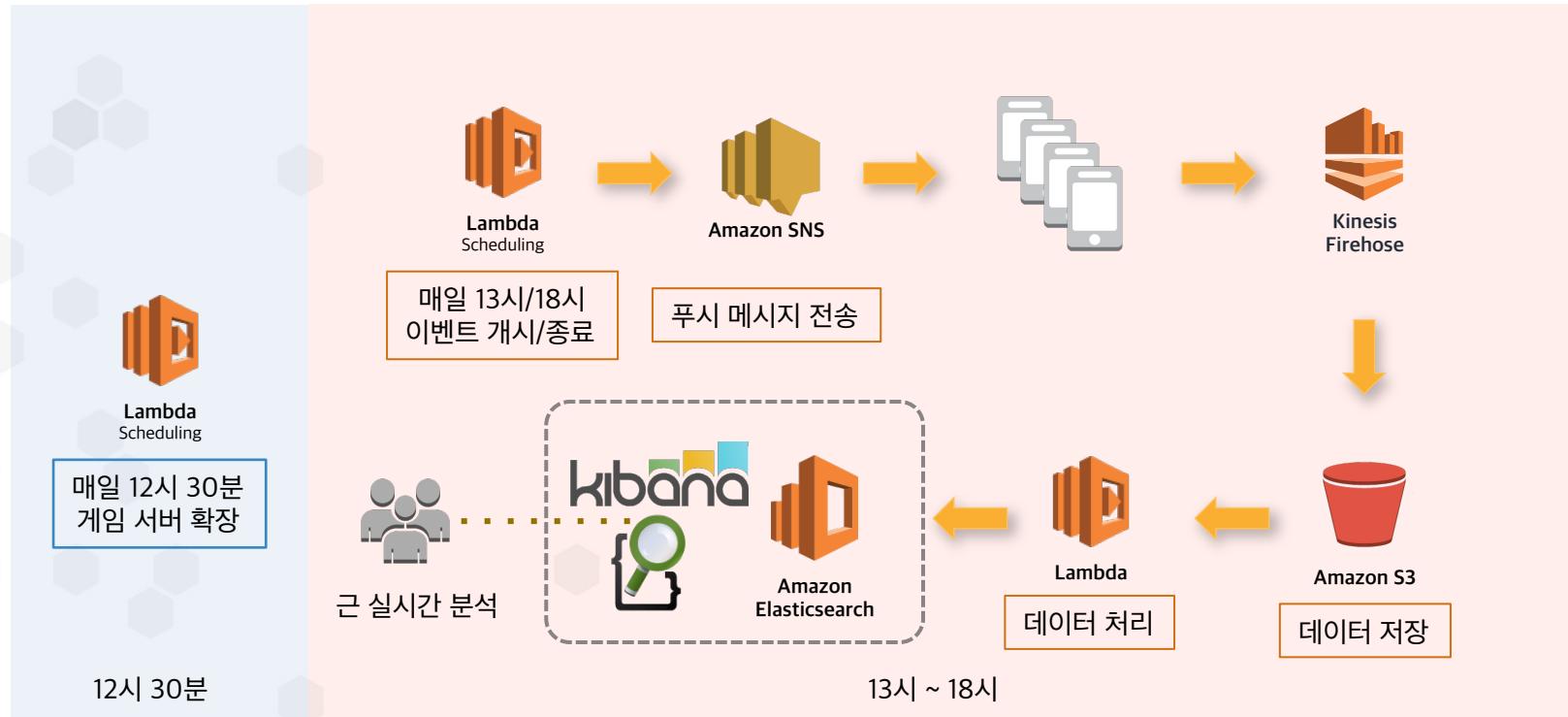
Amazon Kinesis Streams는 1초 이하의 처리 지연으로 스트리밍되는 각 레코드마다
커스텀 처리를 수행하고 선택적으로 스트림 처리 프레임워크들을 사용할 수 있는
워크로드에 맞는 서비스



Amazon Kinesis Firehose는 관리 운영 없이 **S3 또는 Redshift를 기반으로하는
기존 분석 툴들을 그대로 사용**할 수 있고 60초 또는 그 이상의 데이터 지연을 요구하는
워크로드에 맞는 서비스

인게임 이벤트 시스템 아키텍처

13시/18시 이벤트 개시/종료 -> 푸시 메시지 전송 -> 데이터 수집 -> 저장 -> 처리 -> 근 실시간 분석

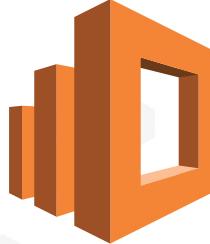


인게임 이벤트 시스템 아키텍처

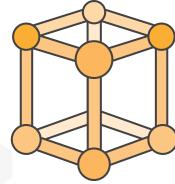
13시/18시 이벤트 개시/종료 -> 푸시 메시지 전송 -> 데이터 수집 -> 저장 -> 처리 -> 근 실시간 분석



Amazon Elasticsearch Service



Amazon Elasticsearch Service 는 클라우드에서 Elasticsearch 클러스터를 쉽게 구성하고, 운영하며 확장할 수 있는 관리형 서비스



쉽게 클러스터를
구성하고 설정을 관리

elasticsearch.



kibana

ELK 를 지원



IAM 를 통한 보안
Amazon CloudWatch를
통한 모니터링
CloudTrail 을 통한 감사



AWS 서비스들과 통합 가능
CloudWatch Logs,
Amazon DynamoDB,
Amazon S3, Amazon Kinesis

인게임 이벤트 시스템 데이터 분석

Top 10 Paying Users

Top 10 user	Sum of amount
69	1189609
84	1022713
67	983190
57	813079
71	776266
15	746156
33	743096
7	740120
51	717904
94	716882

이벤트 기간내 Top 10 결제 유저

Purchase On Event Stage

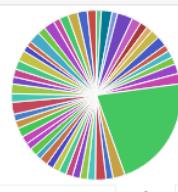
이벤트 스테이지에서 결제 발생 횟수 및 금액

243 12423839

Count

Sum of amount

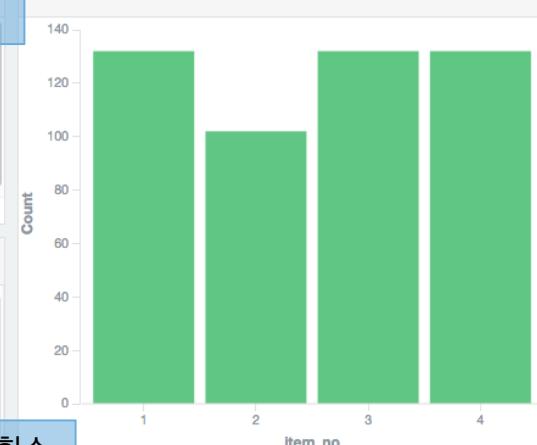
Purchase Count Per Stage



Event Item Obtained

596

Item Obtained Count



Legend
Count

Stage Clear Count

스테이지별 결제 발생 횟수 (녹색: 이벤트 스테이지)



Legend
Count

Stage Over Count

스테이지별 클리어 성공 및 실패 횟수



Legend
Count

분석 결과 활용

- 게임 내 비정상적 사용성 확인 -> 로직 에러 확인
- 이벤트 아이템 획득률 확인 -> 향후 이벤트에 반영
- 이벤트 기간내 Top 10 결제 유저 확인 -> VIP 프로모션 진행
- 최대의 효과를 얻기 위해 실시간 분석 -> 실시간 수치 변경

AWS CLOUD

2016년 1월 7일 | SEOUL - KOREA

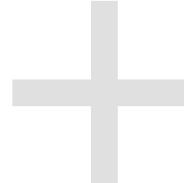
블루홀 테라 렉 로그 분석 이야기

북미 테라 유저 포럼에서
게임이 느리다는 말이 나오고 있습니다…

서버에는 별다른 이상은 없습니다.

클라이언트 로그를 수집하기로 하였습니다.

FPS, 대륙아이디, 채널,
위치정보, 이용자 상태 등
초마다 수집되는 데이터



직업, 종족, 성별, Full screen 여부,
하드웨어 정보, IO 정보 등
이용자 정보 데이터

로그는 1초마다 수집하여 2분동안의 **json format**의 데이터 셋을 만들기로 하였습니다.

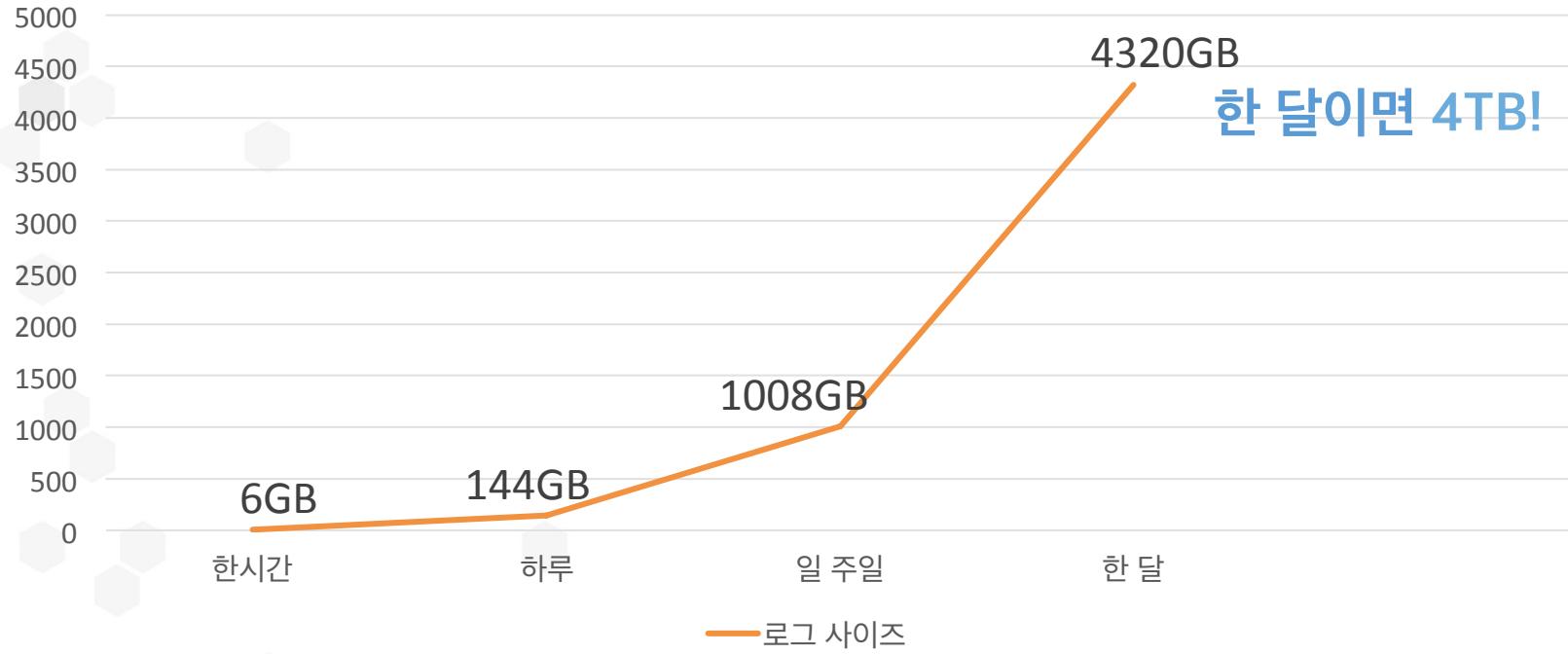
예상 로그 사이즈

로그의 사이즈와 유저 1명이 한시간에 보내는 로그의 사이즈를
파악하여 계산해 보았을 때
로그의 크기는 시간당 약6GB를 예상할 수 있습니다.

(대략적인 동접으로 계산했기 때문에 시간대별 편차는 있습니다.)



예상 로그 사이즈



어떻게 수집하지?

서버 새로 사야 하나? 몇대 사지?

어느 정도 스팩의 서버를 넣어야 할까?

어떻게 수집하지?

어떻게 분석하지?

그전에 유저는 이탈해서 영영 안 돌아 올지도…

빠르게 구축해야 합니다.

비용도 사용한만큼만 내면 되고,
필요할 때 바로 사용할 수 있으니 AWS를 이용해 보기로 하였습니다.



Kinesis로 정했습니다.



Kinesis

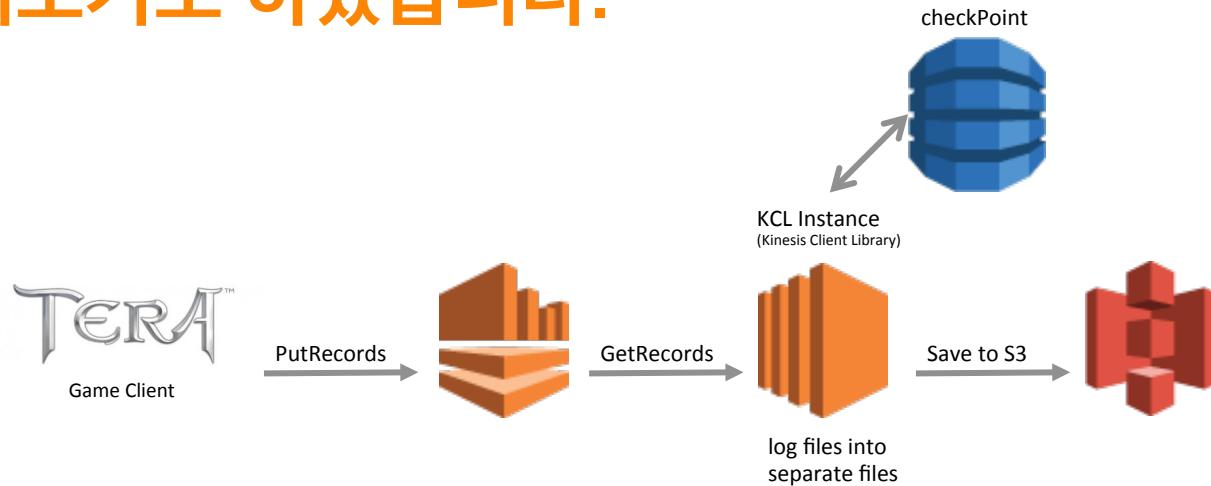
스트림 기반으로 데이터를 수집하고 KCL(Kinesis client library)를 이용하여 Kinesis에 저장된 데이터를 KCL Daemon을 이용하여 S3에 저장합니다. 예상가격도 한 달에 \$40정도 입니다.

Signing and Authenticating REST Requests

AWS SDK나 API Gateway를 사용하지 않고 REST Request를 통하여 데이터를 전송하였습니다.

<http://docs.aws.amazon.com/AmazonS3/latest/dev/RESTAuthentication.html>

이렇게 수집해보기로 하였습니다.



- Tera Client에서 매초마다의 데이터를 수집하여 2분단위로 json format의 데이터를 만듭니다.
- 10분에 한번 2분단위로 수집한 데이터들을 AWS Kinesis로 전송합니다. (PutRecords)
- KCL이 올라와 있는 EC2인스턴스에서는 Kinesis에 저장된 Record를 주기적으로 가져옵니다.(GetRecords)
2분단위로 저장된 데이터는 1초 단위의 데이터로 분리합니다. (2분 단위 1건 데이터 -> 1초 단위 120건 데이터)
- 분리된 데이터는 s3에 저장합니다.

하루 쌓이는 데이터 건수

506,425,200

(10월 1일 기준)

500 ~ 600GB/day

수집한 데이터를 어떻게 분석해야 할까요?



분석에 필요한 소프트웨어는 충분히 있습니다.

그럼 어떤걸 사용해야 할까요?

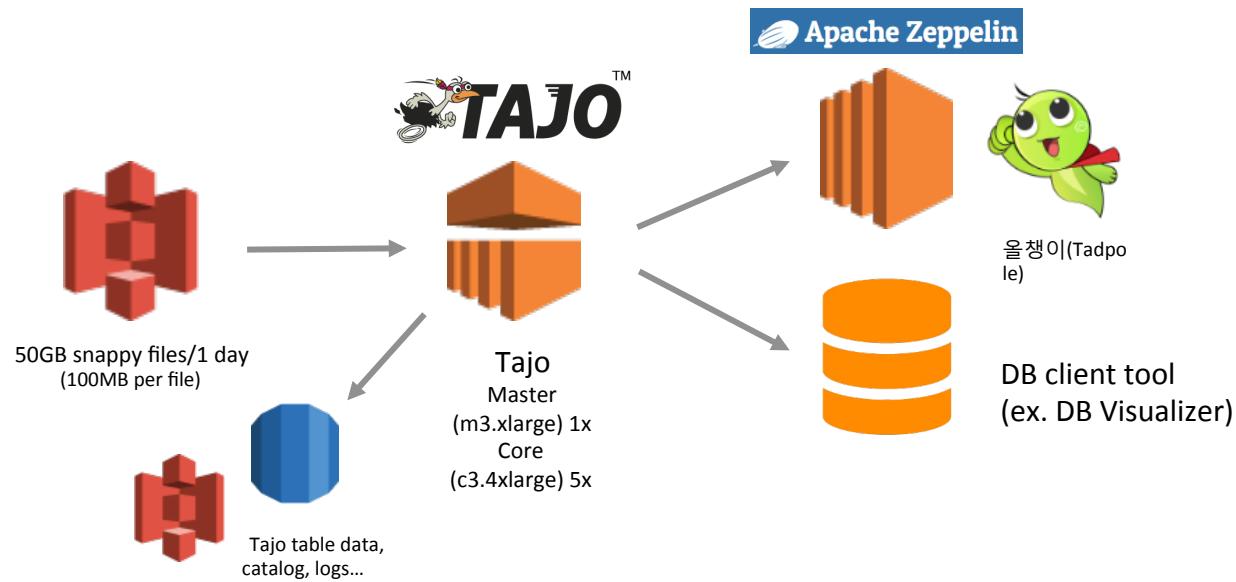
타조(Tajo)를 이용해 보기로 하였습니다.



<http://tajo.apache.org/>

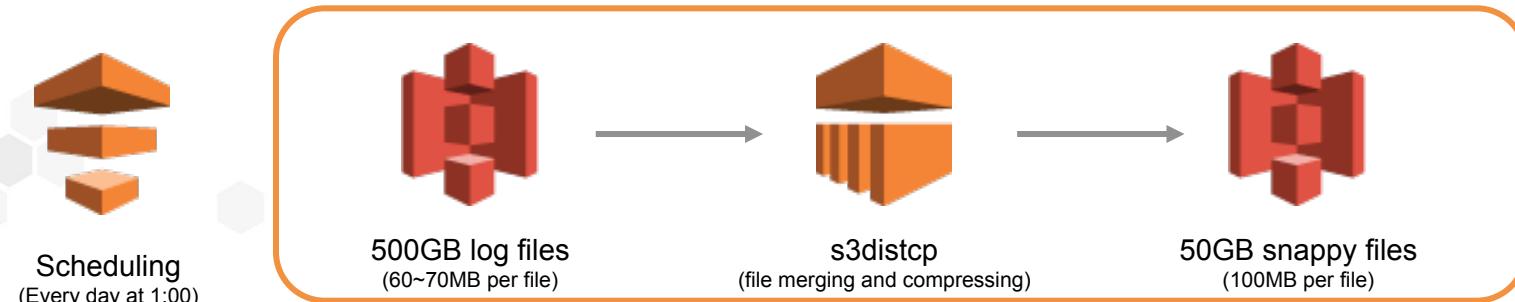
- 분석가에게 가장 익숙한 언어(SQL)로 분석할 수 있어야 합니다.
- DB Client software를 이용한 분석이 가능해야 합니다. (Jdbc, Odbc)
- 빅 데이터(?)를 잘 알지 못하더라도 이용 및 관리를 할 수 있어야 합니다.

데이터 분석



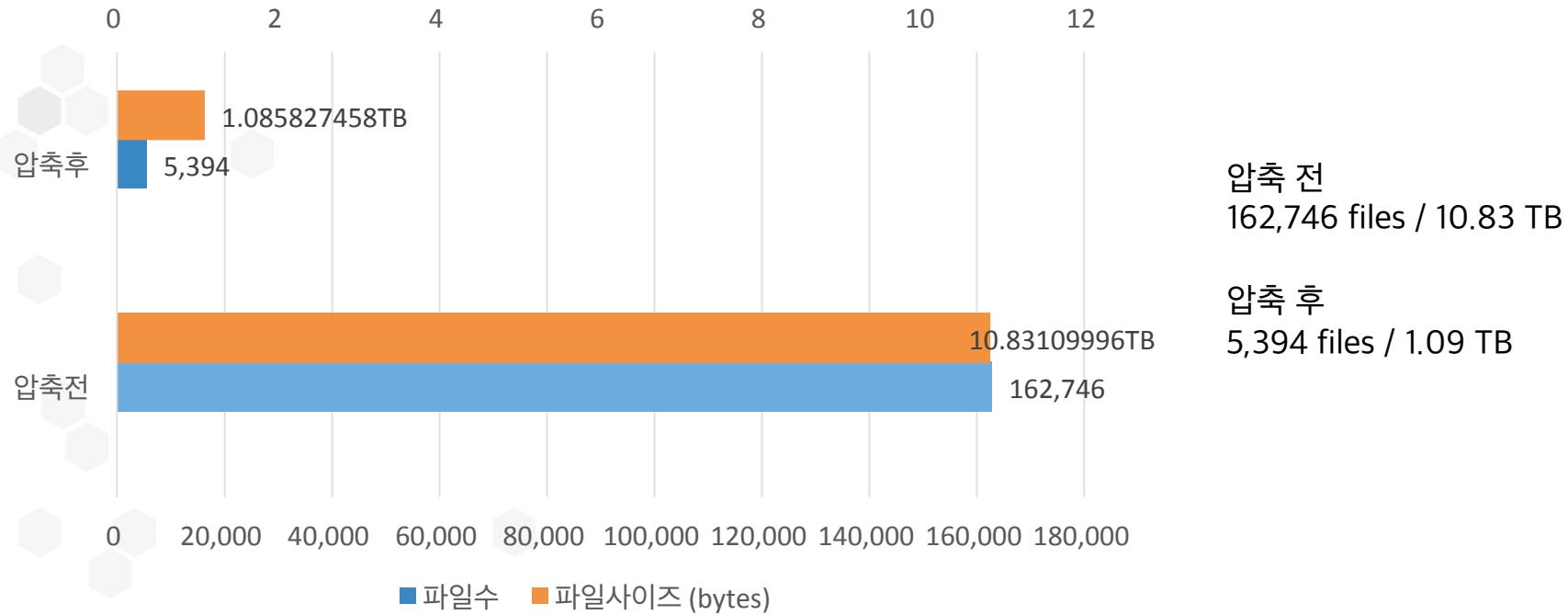
- s3에 압축되어 저장된 로그는 EMR에 세팅된 태조를 이용하여 분석을 합니다.
태조에서는 JDBC 커넥터를 제공하기 때문에 jdbc를 지원하는 다양한 Application에서 이용이 가능 할 수 있습니다.
- 매일 저장되는 약50GB의 데이터는 5억건정도로 이를 분석하기 위해서는 Tajo에서
column based partitioning table을 이용하여 일자별 파티셔닝을 하여 좀더 빠른 검색이 가능하도록 하였습니다.

데이터 압축(snappy)



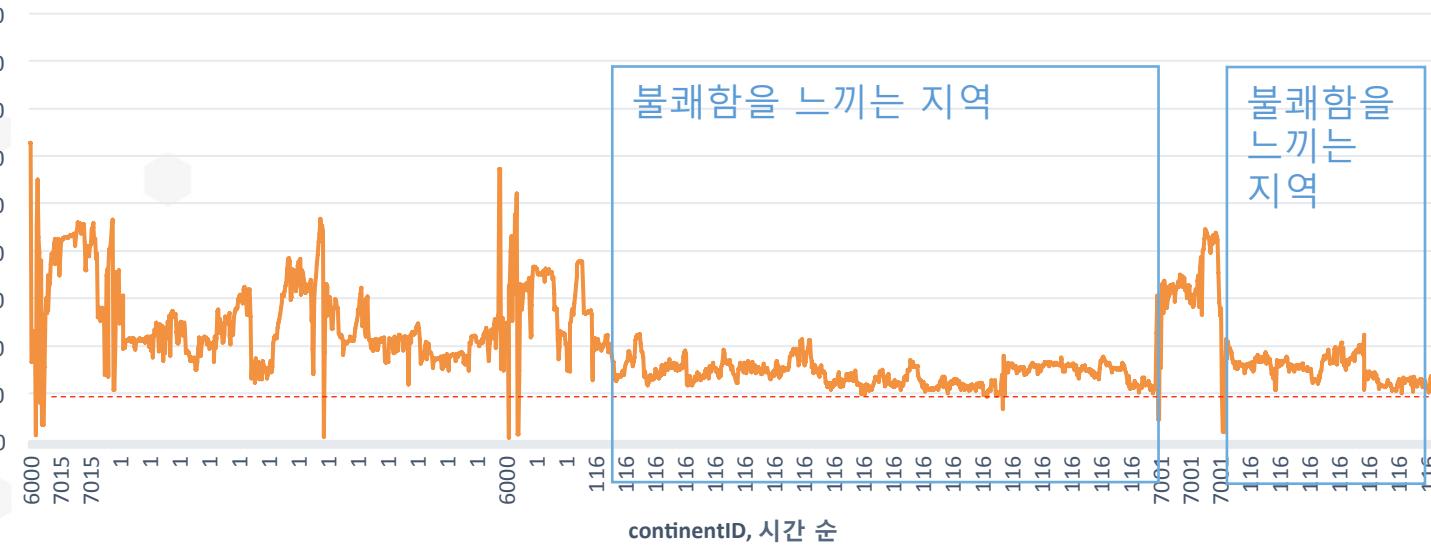
- Kinesis를 통해서 s3에 저장되는 파일은 버퍼의 사이즈에 따라 다르지만 60~70MB 단위의 텍스트 파일로 이루어져 있습니다. 이렇게 매일 쌓이는 데이터는 약 500GB정도로 분석을 하기에는 파일의 개수나 사이즈가 큰 편입니다.
- AWS Data pipeline으로 스케줄링 작업을 수행하여 매일 1시(UTC기준)에 s3에 저장된 데이터를 s3distcp를 이용하여 snappy로 압축을 하고 100~200MB 단위로 압축을 수행합니다.
- 압축된 파일은 s3의 다른 bucket에 저장됩니다.
- 매일 500GB 데이터는 snappy로 압축되어 50GB로 줄어듭니다. 파일은 100~120MB단위로 쪼개져 있습니다.

수치로 보는 데이터크기 (9월 22일 ~ 10월 14일 기준)



결과는 만족스러웠습니다.

특정 지역에서 유저들이 불편함을 느끼고 있었습니다.



프레임 수를 기준으로 쾌적부터 불쾌, 매우 불쾌로 구분하고

일정기준 이하의 불쾌함을 느끼는 유저들의 표본을 뽑아서 불쾌, 매우 불쾌한 지역을 찾아보았습니다.

불만이 접수 된 특정 이용자를 분석해 보았습니다.

20151007	20151008	20151009	20151010	20151011	20151012	20151013	20151014
				0.3%	0.1%	0.3%	0.1%
0.4%	4.9%	3.5%		2.4%	1.8%	0.5%	
0.4%	28.6%	1.5%	1%	23.8%		0.7%	1.5%
1%	3.3%	3%	1.9%	1%	1.7%	1.4%	1.5%
3.8%	1.8%	1%	1.6%	1.8%	2.8%	4.1%	1.9%
							30.9%
6.7%	4.5%	8.8%	6.9%	19%	9.9%	11.1%	16.8%
			1.3%				13.3%

고 사양 하드웨어를 갖추고 있었음에도 불만을 접수한 이용자를 분석해 본 결과,
실제로 느린 것이 아닌 체감에 의한 불만이라고 판단을 할 수도 있었습니다.

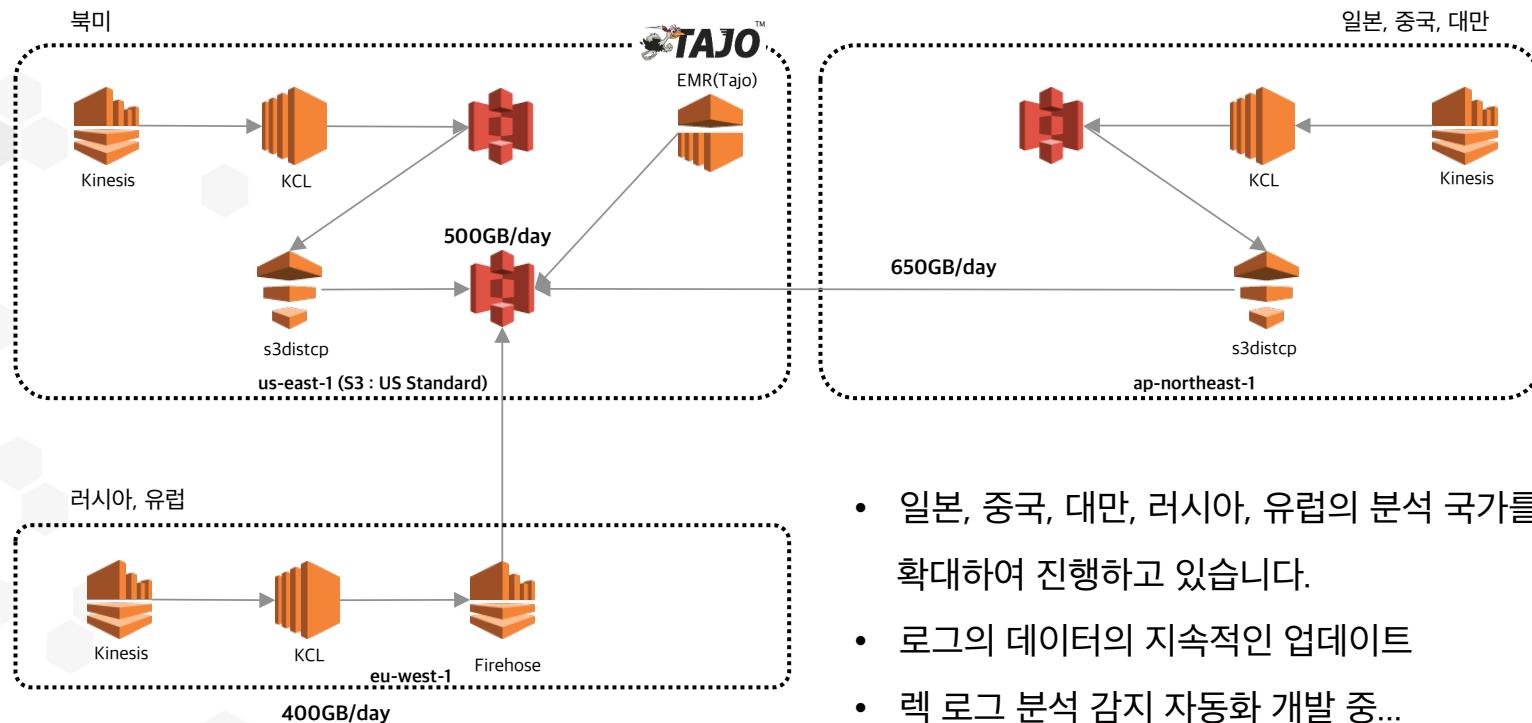
패치 이후 비교 분석

contId	16미만	0~10	11~15	16~20	21~30	31~
	매우 불쾌~불쾌	매우 불쾌	불쾌	보통	쾌적	매우 쾌적
110	23.9%	1.1%	22.8%	34.0%	39.3%	2.8%
8999	6.4%	0.7%	5.7%	7.9%	22.9%	62.8%
9063	3.0%	0.0%	3.0%	6.0%	16.4%	74.6%
9014	2.7%	0.0%	2.7%	13.8%	22.8%	60.7%
116	2.6%	0.1%	2.6%	14.3%	36.0%	47.0%
9021	2.2%	0.0%	2.2%	4.9%	13.9%	79.1%
9828	2.2%	0.0%	2.1%	10.3%	24.2%	63.4%
9757	2.1%	0.0%	2.1%	11.0%	30.2%	56.8%
9094	1.4%	0.0%	1.4%	9.9%	19.4%	69.2%
1	1.3%	0.0%	1.2%	13.0%	35.0%	50.8%
9062	1.2%	0.0%	1.2%	3.1%	18.2%	77.4%
9057	1.1%	0.0%	1.1%	8.2%	24.4%	66.4%
9716	1.0%	0.0%	1.0%	2.7%	15.6%	80.6%
9019	0.8%	0.0%	0.8%	9.1%	19.1%	71.0%
9727	0.7%	0.0%	0.7%	2.9%	21.4%	75.0%
9073	0.7%	0.0%	0.7%	6.1%	23.9%	69.3%
9754	0.6%	0.0%	0.6%	7.5%	26.9%	65.1%
7002	0.5%	0.0%	0.5%	10.3%	29.3%	59.9%
2	0.5%	0.1%	0.4%	6.1%	22.7%	70.7%

클라이언트 패치 이후

지역별 불쾌 또는 매우 불쾌감을
느끼는 유저의 비중이 줄었다!

북미 분석 이후...

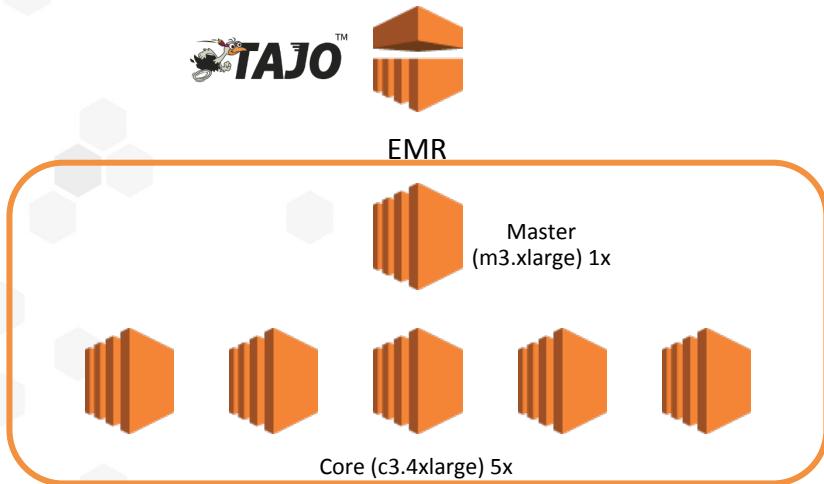


- 일본, 중국, 대만, 러시아, 유럽의 분석 국가를 확대하여 진행하고 있습니다.
- 로그의 데이터의 지속적인 업데이트
- 렉 로그 분석 감지 자동화 개발 중...

예상치 못한 문제가 생겼습니다.

생각 보다 많은 비용이 청구되고 있습니다...

AWS 비용 절약사례 (as-is)



AWS의 비용정책은 사용한 시간만큼의 비용을 지불합니다.

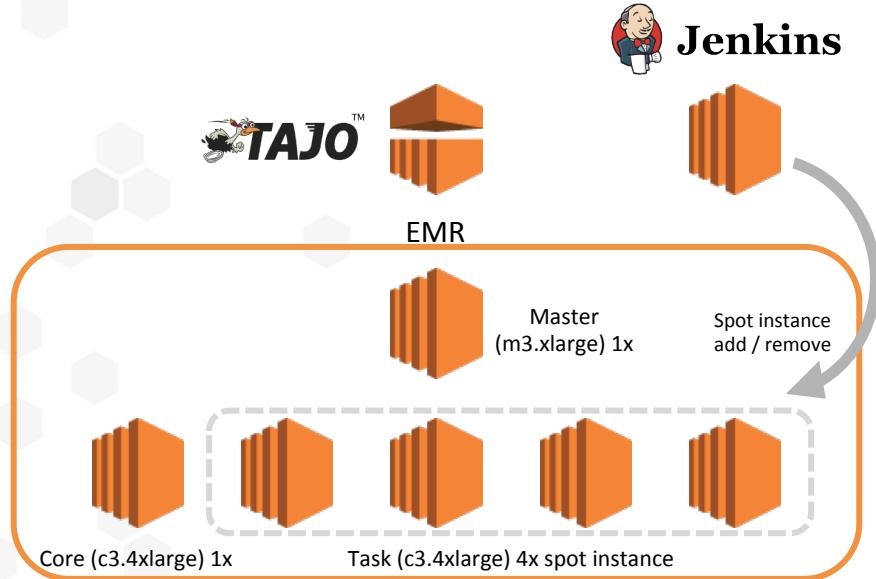
데이터를 분석하기 위한 Tajo환경을 구성하는데 있어서 EMR + EC2인스턴스 비용을 지불 (s3나 기타 RDS는 미비 하며 고정비용이므로 여기선 생략)

실제 사용시간 = 업무시간

그 외의 시간에도 가동됨에 따라 비용이 계속 발생. (15시간 낭비)



AWS 비용 절약사례 (to-be)



Instance type	EC2 pricing	EC2 Bidding price
c3.4xlarge	\$0.84 / hour	\$0.18 / hour

매일 출근 시간 30분전에 c3.4xlarge 4대를 Spot instance로 추가 합니다.
그 경우 희망하는 가격으로 입찰할 수 있습니다.
c3.4xlarge의 입찰가 범위는 \$0.16~0.19사이
(on-demand 가격은 \$0.84)

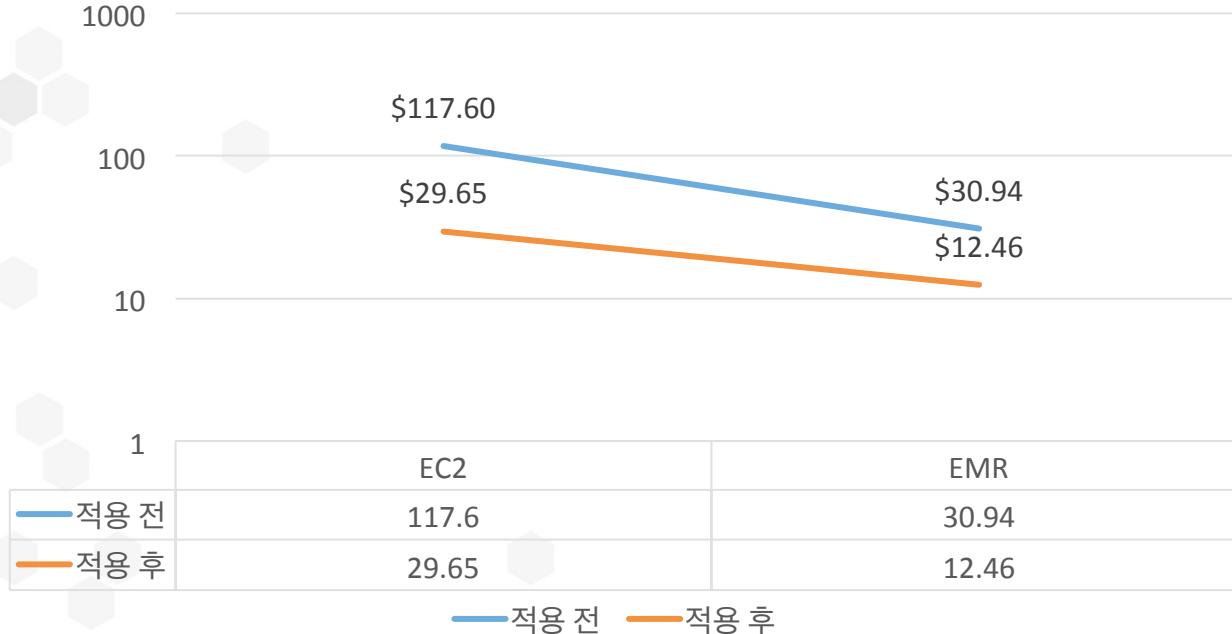
그리고 매일 오후 8시에 낙찰된 인스턴스를 반납합니다.

이렇게 매일 업무시간 이외에는 최소한 유지해야 하는 자원만 남길 수 있습니다.
(Master 1대, Core 1대)



AWS 비용 절약사례 (to-be)

절약 비용(일 기준)



매일 \$100정도 절약할 수 있게 되었습니다!!

물론 토요일, 일요일에는 2대만 가동되기 때문에 실제 한달 이용 요금은 더 저렴하다.

AWS도 저와 같은 고민을 하고 있었습니다.

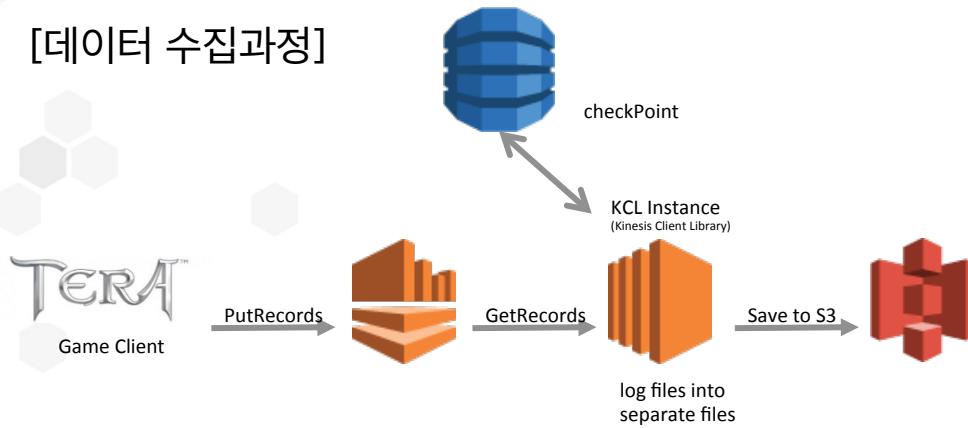


더욱 저렴하게 더 간편하게 구현이 가능해졌습니다.

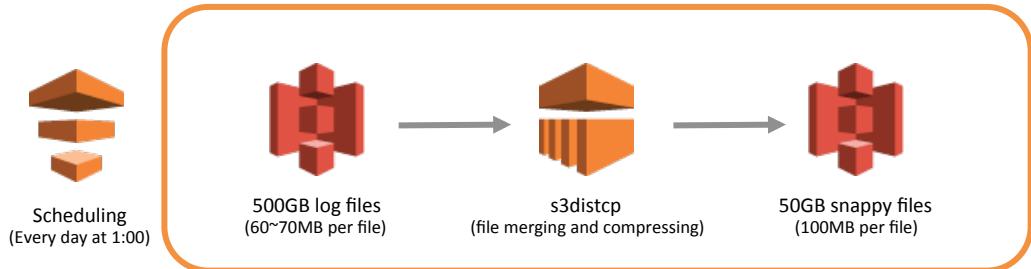
AWS re:Invent 2015

AWS도 저와 같은 고민을 하고 있었습니다.

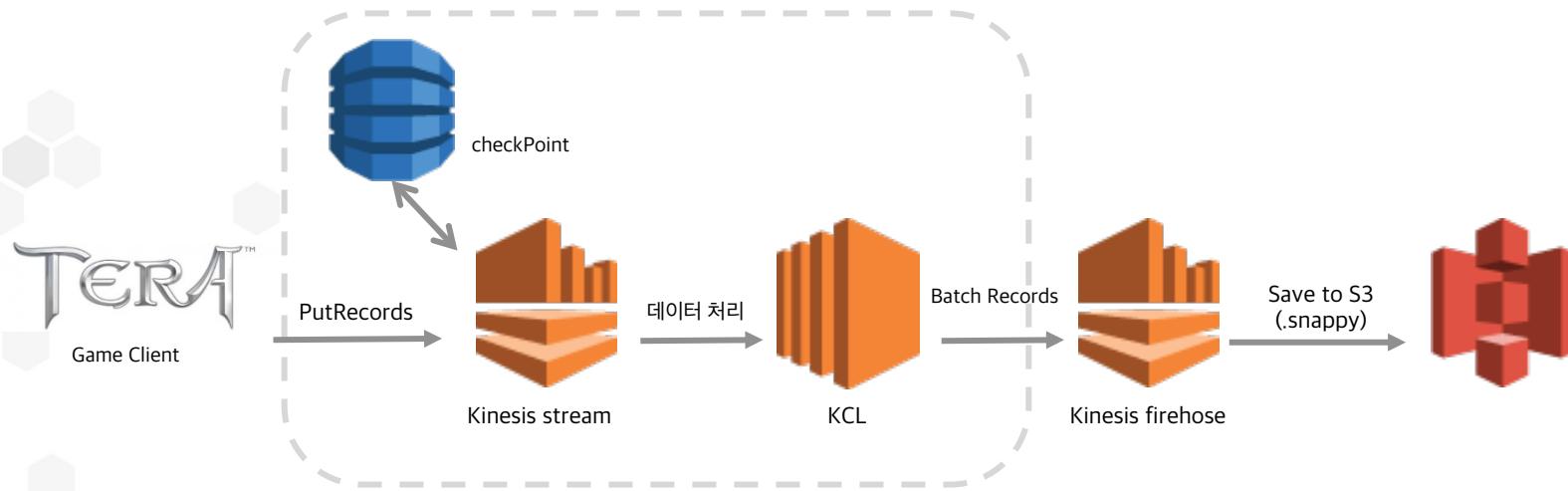
[데이터 수집과정]



[데이터 압축과정]



AWS Kinesis Firehose면 됩니다.



Kinesis Firehose만으로도 가능하겠지만, 2분동안 쌓인 로그를 초단위형식의
데이터를 변환하는 과정이 필요하였기 때문에 Kinesis Streams & KCL이 필요했습니다.

요약

- AWS를 이용해서 3주만에 데이터 분석 플랫폼을 구축할 수 있었습니다.
- 처음 사용시 불편한점이 없지 않아 있었지만 AWS의 지속적인 업데이트로 인해 많이 해소 되었습니다.
- 스팟 인스턴스를 통해 가격을 한차례 더 절감할 수 있었습니다.



플랫폼 서버 개발자 분들의 많은 관심 부탁드립니다!

맺으면서

- 사용 패턴에 맞는 AWS 서비스를 선택하세요!
 - 다양한 문제를 다양한 방법으로 해결할 수 있어요
- 데이터를 기반으로 한 의사결정은 큰 도움이 되요!
 - 데이터는 올바른 의사결정의 필요충분조건!
- AWS 매니지드 서비스의 이점을 누리세요!
 - 개발에만 집중하실 수 있어요
 - 게임 서비스 품질을 향상시킬 수 있어요

어떻게 시작할까요?

- AWS 빅데이터 추천 콘텐츠
 - 빅데이터 블로그 <https://blogs.aws.amazon.com/bigdata/blog/>
 - 주요 서비스 <http://aws.amazon.com/ko/big-data/>
 - 고객 사례 <https://aws.amazon.com/ko/solutions/case-studies/big-data/>
- 경험해보기
 - Qwiklabs 실습 - <https://qwiklabs.com>
 - 각종 샘플 코드 - <https://github.com/aws-labs/aws-big-data-blog>

AWS CLOUD

2016년 1월 7일 | SEOUL - KOREA

여러분의 피드백을 기다립니다!

- AWS 공식 블로그: <http://aws.amazon.com/ko/blogs/korea>
- AWS 공식 소셜 미디어



@AWSKorea



AWSKorea



AmazonWebServices



AWSKorea

