

# 이커머스 상품 데이터 파이프라인 구축 실습

# 1. 개요

- 지난 챕터들에서 배운 Spark, Cassandra 를 이용해,  
흩어져 있는 여러 상품 정보들을 한 곳에 모으는 간단한 ETL 파이프라인을 만드는 실습 진행.

# 1. 요구 조건 정의

# 1. Background

- 이커머스 사이트에서는 수많은 상품의 정보들을 저장하고 있음.
- ex) 네이버 쇼핑 - 아이폰 14 검색
  - [https://search.shopping.naver.com/search/all?query=%EC%95%84%EC%9D%B4%ED%8F%B0%2014&cat\\_id=&frm=NVSHATC](https://search.shopping.naver.com/search/all?query=%EC%95%84%EC%9D%B4%ED%8F%B0%2014&cat_id=&frm=NVSHATC)
- 상품의 이름, 카테고리, 가격, 판매자 정보, 리뷰 등등

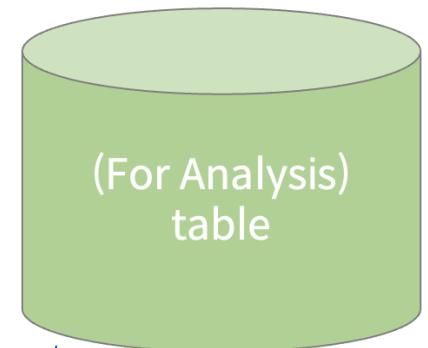
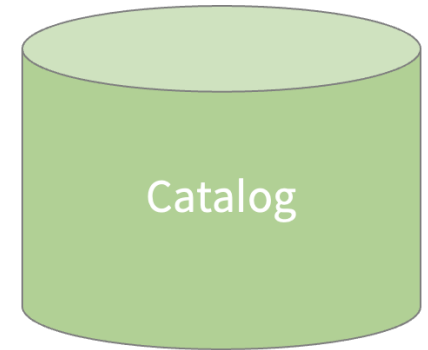
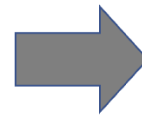
# 1. Background

- 회사 초기에는 모든 상품 관련 정보들을 한 시스템, 혹은 한 팀에서 모두 관리할 수 있을 것.
- 회사의 규모가 커지고 상품의 수가 많아지면서, 각각의 정보들은 여러 시스템에 나뉘어 저장되고 여러 팀이 관리하는 방향으로 바뀜. (MSA)
  - ex) 상품의 후기 정보 DB, 판매자 정보 DB 분리
  - MSA 관련 좋은 영상 자료 : [\[우아콘2020\] 배달의민족 마이크로서비스 여행기](#)
- **그러나 여러 곳에 나뉘어 있는 정보들을 한 곳에 모아야 할 유즈 케이스들이 존재!**
  1. 상품 정보를 필요로 하는 백엔드 서버에서 쿼리를 날릴 때,  
매번 관련 데이터들을 join하지 않고 미리 join 되어 있는 테이블로부터 데이터를 가져온다면,  
적은 부하로 빠르게 가져올 수 있음. (역정규화)
  2. 데이터 분석용 쿼리는 무겁고, 특정 컬럼만을 필요로 하는 경우가 많음.  
이 경우 column 기반의 포맷으로 데이터를 한 곳에 모아놓는 경우가 유리!

## 2. 실습 파이프라인 구조

- 로컬에 먼저 이 구조를 구현해본 후 AWS EMR에 배포!

### Input data



- 실습에서는 단순히 json파일을 읽어오지만, 다양한 소스들이 사용될 수 있음.  
(ex - MySQL, MongoDB, Cassandra, Kafka, json file, parquet file, etc...)

Apache iceberg

<https://iceberg.apache.org/>

### 3. Catalog? 용어 정리)

- 이커머스 도메인에서 일반적으로  
상품의 정보들을 저장하고 있는 Database (or Data warehouse) 를 지칭.
- 상품의 정보? -> 매우 다양한 속성들 포함.
  - 가격, 브랜드, 카테고리, 이름, 리뷰, 판매자, 할인 상품 여부, 등록일, 이미지 등
  - 카테고리 별로 다른 속성들을 가질 수 있음.
    - IT 기기 ? -> CPU, RAM, DISK size 등의 스펙
    - 의류 ? -> 허리 둘레, 가슴 사이즈 등
    - 도서? -> 인문/과학/IT, ISBN

## 코드 실습 (local)



# 1. 코드 실습 링크

- [https://github.com/startFromBottom/fc-spark-practices/tree/main/practical\\_ex](https://github.com/startFromBottom/fc-spark-practices/tree/main/practical_ex)
- input json data
  - [https://github.com/startFromBottom/fc-spark-practices/tree/main/practical\\_ex/input](https://github.com/startFromBottom/fc-spark-practices/tree/main/practical_ex/input)
- Spark ETL code
  - [https://github.com/startFromBottom/fc-spark-practices/blob/main/practical\\_ex/ecommerce\\_ex.py](https://github.com/startFromBottom/fc-spark-practices/blob/main/practical_ex/ecommerce_ex.py)
- CQL (create table) - Cassandra 실습에서 사용했던 docker container 그대로 사용
  - [https://github.com/startFromBottom/fc-spark-practices/blob/main/practical\\_ex/create\\_cassandra\\_table.sql](https://github.com/startFromBottom/fc-spark-practices/blob/main/practical_ex/create_cassandra_table.sql)
- Apache Iceberg
  - [https://github.com/startFromBottom/fc-spark-practices/blob/main/practical\\_ex/docker-compose.yml](https://github.com/startFromBottom/fc-spark-practices/blob/main/practical_ex/docker-compose.yml)
  - [https://github.com/startFromBottom/fc-spark-practices/blob/main/practical\\_ex/ecommerce\\_iceberg\\_ex.ipynb](https://github.com/startFromBottom/fc-spark-practices/blob/main/practical_ex/ecommerce_iceberg_ex.ipynb)

# AWS EMR 에 배포

## 2. AWS EMR 관련 참고 자료

AWS EMR 시작하기 (CLI, GUI 에서 EMR cluster를 생성하고, spark 잡 제출하고, 종료하는 법 포함)

- [https://docs.aws.amazon.com/ko\\_kr/emr/latest/ManagementGuide/emr-gs.html](https://docs.aws.amazon.com/ko_kr/emr/latest/ManagementGuide/emr-gs.html)

### EMR Best Practices Guides (정독 강력 추천)

- [https://aws.github.io/aws-emr-best-practices/features/spot\\_usage/best\\_practices/](https://aws.github.io/aws-emr-best-practices/features/spot_usage/best_practices/)

Monitor and Optimize Analytic Workloads on Amazon EMR with Prometheus and Grafana.

- <https://aws.amazon.com/ko/blogs/big-data/monitor-and-optimize-analytic-workloads-on-amazon-emr-with-prometheus-and-grafana/>

## 2. CLI에서 EMR 사용 가능하도록 세팅하는 방법)

### 1. aws cli 설치 (MAC)

- 문서 링크 : [https://docs.aws.amazon.com/ko\\_kr/cli/latest/userguide/getting-started-install.html](https://docs.aws.amazon.com/ko_kr/cli/latest/userguide/getting-started-install.html)
- `$ curl "https://awscli.amazonaws.com/AWSCLIV2.pkg" -o "AWSCLIV2.pkg" sudo installer -pkg AWSCLIV2.pkg -target`
- `$ which aws`
- `$ aws --version` (정상적으로 설치된 것을 확인)

### 2. IAM role 생성 (root user를 권장하진 않지만 root key 사용)

1. [https://us-east-1.console.aws.amazon.com/iamv2/home#/security\\_credentials](https://us-east-1.console.aws.amazon.com/iamv2/home#/security_credentials)
2. Access keys cmd+f로 찾고 Create Access key 누르기
3. Key 생성 후, Access key, Secret Access key 복사해두기

#### Access keys (1)

Use access keys to send programmatic calls to AWS from the AWS CLI, AWS Tools for PowerShell, AWS SDKs, or direct AWS API calls. You can have a maximum of two access keys (active or inactive) at a time. [Learn more](#)

Actions ▼

Create access key

Access key ID

Created on

Access key last used

Region last used

Service last used

Status

#### Retrieve access keys

##### Access key

If you lose or forget your secret access key, you cannot retrieve it. Instead, create a new access key and make the old key inactive.

Access key

Secret access key



[Redacted]



\*\*\*\*\* Show

## 2. CLI에서 EMR 사용 가능하도록 세팅하는 방법)

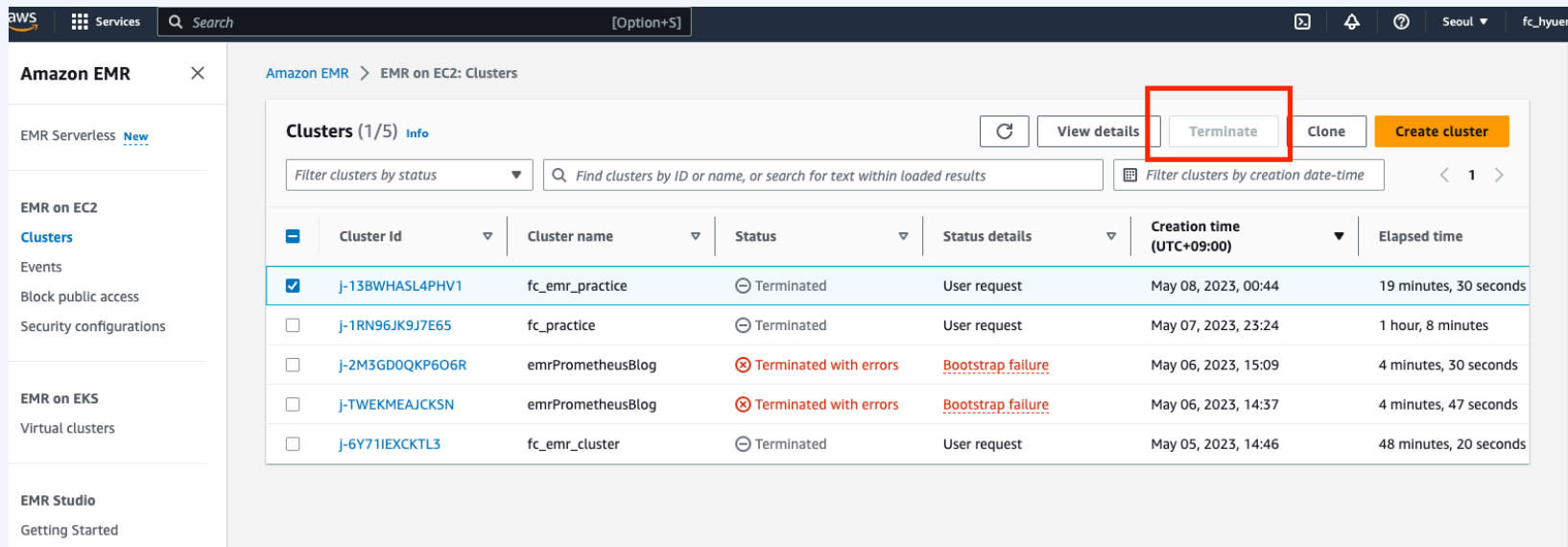
### 3. aws configure에서 access\_key, secret\_key, region 설정

```
→ aws configure
AWS Access Key ID [*****]:
AWS Secret Access Key [*****]:
Default region name [ap-northeast-2]:
```

### 4. 터미널에서 \$ aws emr create-default-roles 로 emr 관련 역할 생성

### 3. 실습 주의 사항

EMR Cluster 실습 완료 후에는 반드시!  
Cluster 종료 시키기



The screenshot shows the AWS Management Console interface for Amazon EMR. The left sidebar contains navigation links for Amazon EMR, EMR Serverless, EMR on EC2 (Clusters, Events, Block public access, Security configurations), EMR on EKS (Virtual clusters), and EMR Studio (Getting Started). The main content area displays the 'EMR on EC2: Clusters' page. At the top, there are buttons for 'View details', 'Terminate' (highlighted with a red box), 'Clone', and 'Create cluster'. Below these buttons is a table listing the clusters. The table has columns for Cluster Id, Cluster name, Status, Status details, Creation time (UTC+09:00), and Elapsed time. The first cluster, 'j-13BWHASL4PHV1' with name 'fc\_emr\_practice', is selected and its status is 'Terminated'. The other clusters are also listed with their respective statuses and details.

Cluster Id	Cluster name	Status	Status details	Creation time (UTC+09:00)	Elapsed time
<input checked="" type="checkbox"/> j-13BWHASL4PHV1	fc_emr_practice	Terminated	User request	May 08, 2023, 00:44	19 minutes, 30 seconds
<input type="checkbox"/> j-1RN96JK9J7E65	fc_practice	Terminated	User request	May 07, 2023, 23:24	1 hour, 8 minutes
<input type="checkbox"/> j-2M3GDOQKP6O6R	emrPrometheusBlog	Terminated with errors	Bootstrap failure	May 06, 2023, 15:09	4 minutes, 30 seconds
<input type="checkbox"/> j-TWEKMEAJCKSN	emrPrometheusBlog	Terminated with errors	Bootstrap failure	May 06, 2023, 14:37	4 minutes, 47 seconds
<input type="checkbox"/> j-6Y71IEXCKTL3	fc_emr_cluster	Terminated	User request	May 05, 2023, 14:46	48 minutes, 20 seconds