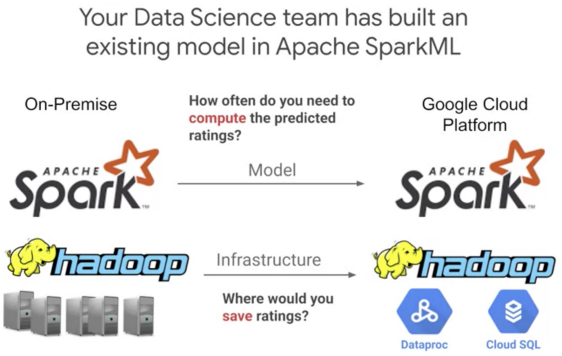
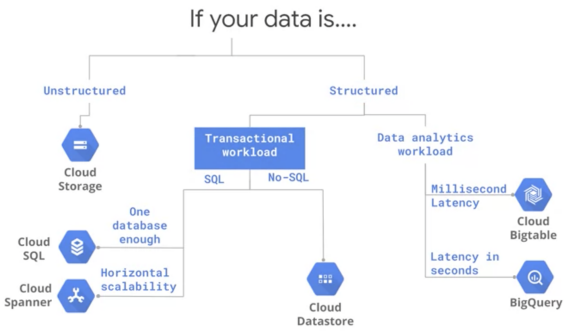
Dataproc - Spark ML



- Spark 컴퓨팅 작업은 Dataproc으로, 작업결과는 Cloud SQL RDBMS에 저장
- Dataproc은 일괄 처리, 쿼리, 스트리밍, 머신 러닝에 오픈소스 데이터 도구를 활용할 수 있는 관리형 Spark 및 Hadoop 서비스입니다. Dataproc 자동화를 통해 신속하게 클러스터를 만들고 손쉽게 관리하며 불필요한 클러스터를 사용 중지하여 비용을 절감할 수 있습니다. 관리 시간과 비용이 절감되므로 작업과 데이터에 집중할 수 있습니다.
  - Created a fully-managed Cloud SQL instance for rentals
  - Created tables and explored the schema with SQL
  - Ingested data from CSVs
  - Edited and ran a Spark ML job on Dataproc
  - Viewed prediction results

Choose your solutions based on access pattern

	Cloud Storage	Cloud SQL	Datastore	Bigtable	BigQuery
Capacity	Petabytes +	Gigabytes	Terabytes	Petabytes	Petabytes
Access metaphor	Like files in a file system	Relational database	Persistent Hashmap	Key-value(s), HBase API	Data warehouse
Read	Have to copy to local disk	SELECT rows	filter objects on property	scan rows	SELECT rows
Write	One file	INSERT row	put object	put row	Batch/stream
Update granularity	An object (a "file")	Field	Attribute	Row	Field
Usage	Store blobs	No-ops SQL database on the cloud	Structured data from AppEngine apps	No-ops, high throughput, scalable, flattened data	Interactive SQL* querying fully managed warehouse



GCP의 데이터 관리 솔루션들

Aa 항목	Cloud Storage	Cloud SQL	Datastore	Bigtable	BigQuery
capacity(용량)	Pb + (1000Tb +) 매우 큼	Gb (작음)	Tb (1000Gb)	Pb	Pb
access metaphor	형태 없는 모든 파일데이터들	transaction 관계형 테이블	transaction NoSQL(비관계형)	**Hbase API**	데이터 웨어하우스
읽기	local로 받아서 사용	쿼리를 날려서 사용	filtering?	scan?	쿼리를 날려서 사용
쓰기	파일 하나하나씩	쿼리를 날려서 사용	put?	put?	배치/스트리밍
사용법	**Blobs**	No-ops	structured data from appengine	실시간 대용량처리	데이터세트에 대한 분석

▼ **HBase란?**

아파치 HBase(Apache HBase)는 하둡 플랫폼을 위한 공개 비관계형 분산 데이터 베이스이다. 구글의 빅테이블(BigTable)을 본보기로 삼았으며 자바로 쓰여졌다. 아파치 소프트웨어 재단의 아파치 하둡 프로젝트 일부로서 개발되었으며 하둡의 분산 파일 시스템인 HDFS위에서 동작을 한다. 대량의 흩어져 있는 데이터 저장을 위한 무정지 방법을 제공하는 구글의 빅테이블과 비슷한 기능을 한다.

HBase는 압축, 인메모리 처리, 초기 빅테이블에 제시되어 있는 Bloom 필터 기능을 제공한다.[1] HBase에 있는 테이블들은 하둡에서 동작하는 맵리듀스 작업을 위한 입출력을 제공하며 자바 API나 REST, Avro 또는 Thrift 게이트웨이를 통하여 접근할 수 있다.

HBase는 기존의 SQL 데이터 베이스를 직접적으로 대체하지는 않지만 페이스북의 메시징 플랫폼[2]과 같은 데이터를 많이 사용하는 웹사이트에서 사용된다.

▼ **BLOB이란?**

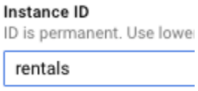
바이너리 라지 오브젝트(Binary large object, BLOB)는 데이터베이스 관리 시스템의 하나의 엔티티로서 저장되는 이진 데이터의 모임이다. BLOB은 일반적으로 그림, 오디오, 또는 기타 멀티미디어 오브젝트인 것이 보통이지만, 바이너리 실행 코드가 BLOB으로 저장되기도 한다. BLOB에 대한 데이터베이스 지원은 보편적인 것은 아니다.

자료형과 정의는 전통적인 컴퓨터 데이터베이스 시스템에 본래 정의되지 않은 데이터를 기술하기 위해 도입되었다. 당시 저장하려는 크기가 너무 컸기 때문에 1970년대와 1980년대에 데이터베이스 시스템의 필드에 처음 정의되었다. 디스크 공간의 값이 떨어졌을 때 이 자료형은 실용적으로 되

실습!

1. Create a Cloud SQL instance

1. In the Google Cloud Console, Select **Navigation menu > SQL** (in the Databases section).
2. Click **Create instance**.
3. Click **Choose MySQL**.
4. For **Instance ID**, type **rentals**.



2. Create tables

```
CREATE DATABASE IF NOT EXISTS recommendation_spark;

USE recommendation_spark;

DROP TABLE IF EXISTS Recommendation;
DROP TABLE IF EXISTS Rating;
DROP TABLE IF EXISTS Accommodation;

CREATE TABLE IF NOT EXISTS Accommodation
(
  id varchar(255),
  title varchar(255),
  location varchar(255),
  price int,
  rooms int,
  rating float,
  type varchar(255),
  PRIMARY KEY (ID)
);

CREATE TABLE  IF NOT EXISTS Rating
(
  userId varchar(255),
  accoId varchar(255),
  rating int,
  PRIMARY KEY(accoId, userId),
  FOREIGN KEY (accoId)
    REFERENCES Accommodation(id)
);

CREATE TABLE  IF NOT EXISTS Recommendation
(
  userId varchar(255),
  accoId varchar(255),
  prediction float,
  PRIMARY KEY(userId, accoId),
```

```
  FOREIGN KEY (accoId)
    REFERENCES Accommodation(id)
);

SHOW DATABASES;
```

3. Stage Data

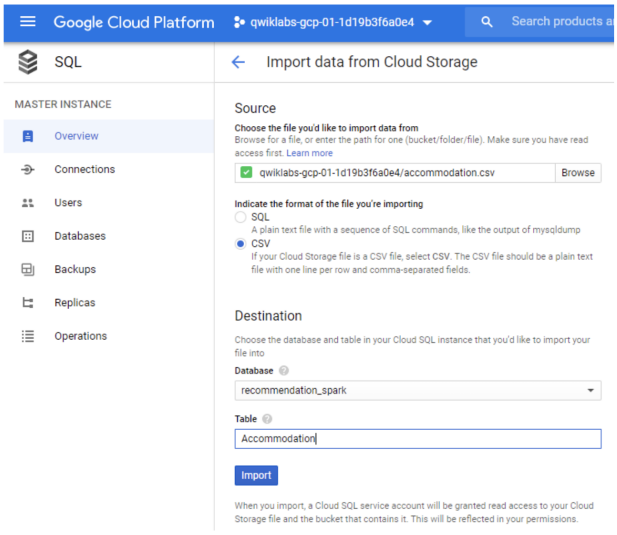
1. Option 1: Use the command line

```
echo "Creating bucket: gs://$DEVSHELL_PROJECT_ID"
gsutil mb gs://$DEVSHELL_PROJECT_ID

echo "Copying data to our storage from public dataset"
gsutil cp gs://cloud-training/bdml/v2.0/data/accommodation.csv gs://$DEVSHELL_PROJECT_ID
gsutil cp gs://cloud-training/bdml/v2.0/data/rating.csv gs://$DEVSHELL_PROJECT_ID

echo "Show the files in our bucket"
gsutil ls gs://$DEVSHELL_PROJECT_ID

echo "View some sample data"
gsutil cat gs://$DEVSHELL_PROJECT_ID/accommodation.csv
```

2. Option 2: Use the Cloud Console UI
    a. Navigate to **Storage** and select **Cloud Storage > Browser**.
    b. Click **Create Bucket** (if one does not already exist).
    c. Specify your project name as the bucket name.
    d. Click **Create**.
    e. Download the below files locally and then upload them inside of your new bucket

4. Load data from Cloud Storage into Cloud SQL tables

- 콘솔의 SQL로 이동해서 import를 통해서 accommodation, ratings load



5. Explore Cloud SQL data

6. Launch Dataproc

1. SQL에서 Dataproc API 허용
2. cluster 생성
3. Master/Workers 노드 생성

```
echo "Authorizing Cloud Dataproc to connect with Cloud SQL"
CLUSTER=rentals
CLOUDSQL=rentals
ZONE=us-central1-c
NWORKERS=2

machines="$CLUSTER-m"
for w in `seq 0 $(($NWORKERS - 1))`; do
   machines="$machines $CLUSTER-w-$w"
done

echo "Machines to authorize: $machines in $ZONE ... finding their IP addresses"
ips=""
for machine in $machines; do
    IP_ADDRESS=$(gcloud compute instances describe $machine --zone=$ZONE --format='value(networkInterfaces.accessConfigs[].natIP)' | sed "s
    echo "IP address of $machine is $IP_ADDRESS"
    if [ -z  $ips ]; then
       ips=$IP_ADDRESS
    else
       ips="$ips,$IP_ADDRESS"
    fi
done

echo "Authorizing [$ips] to access cloudsql=$CLOUDSQL"
gcloud sql instances patch $CLOUDSQL --authorized-networks $ips
```

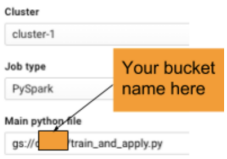
4. SQL Public IP copy

7. Run the ML model

```
gsutil cp gs://cloud-training/bdml/v2.0/model/train_and_apply.py train_and_apply.py
cloudshell edit train_and_apply.py
```

8. Run your ML job on Dataproc

1. Submit Job



9. Explore inserted rows with SQL