



Data Engineering and Analytics on AWS

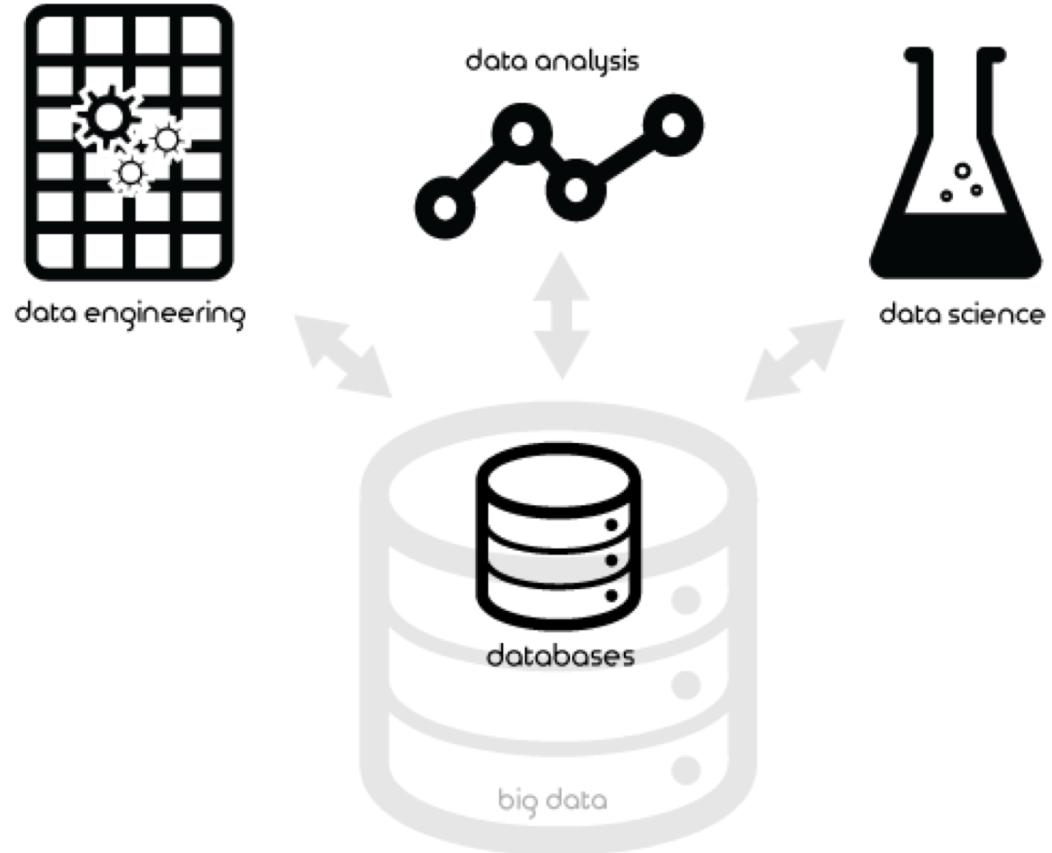
Collect data, visualize, and share insight

Jung, SeUng, Big Data & Analytics SA, AWS

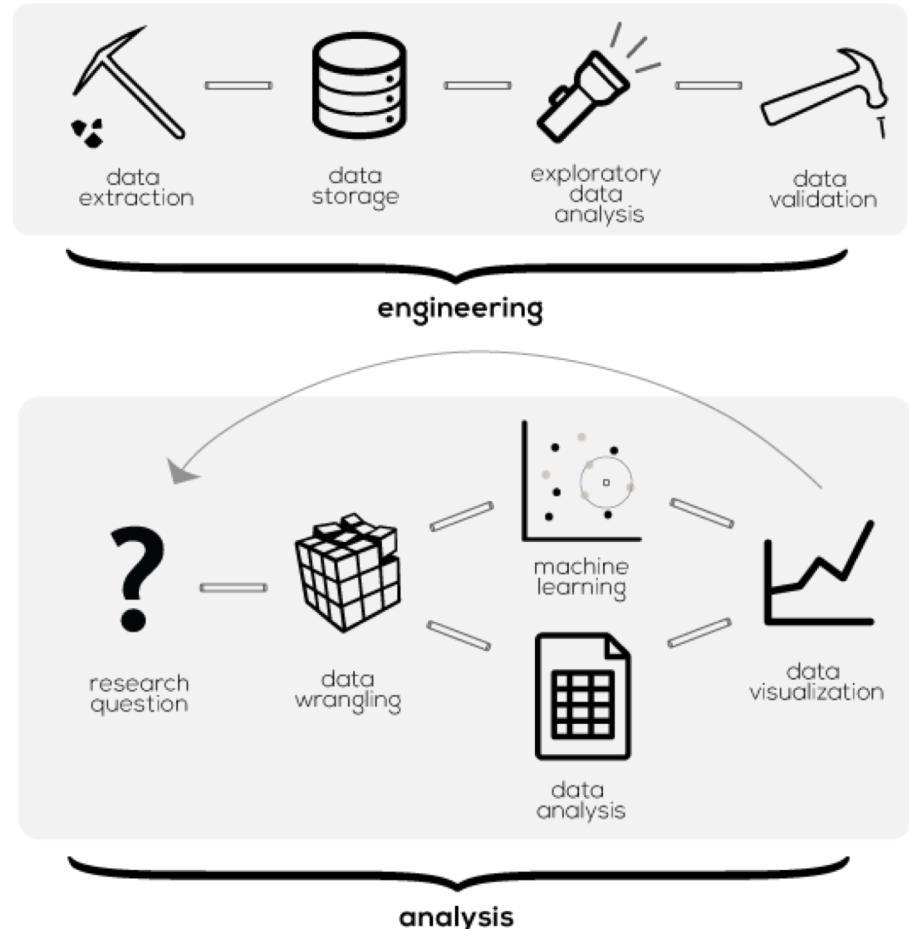
Data Engineering

Data Engineering

Prepare data pipeline, data catalog, and ETL for data analysis



the data pipeline



Catalog

상품 관련 추천

What's new about our new lower prices?

Absolutely nothing, except for the price.
Everything else is just as you remember it. Quality materials? Exactly the same. Unique design? Unchanged. Smart functionality? Smart as ever. Durability? Still solid. The only thing that has changed is the price tag. So, how do we do it? We just find simple ways to lower costs. Whether it's flat-packing our products to reduce shipping costs and fuel usage or keeping our designs simple yet practical to save money, we can slash prices without affecting style, quality or functionality.

It's like a sale that lasts all year long.
Hurry in to your local IKEA store and get the products you love, for less.



POÄNG armchair
NEW LOWER PRICE \$69
last year's price *\$89
Birch/lime natural. See p. 44.

상품 리스트 /
상세 정보



상품 카테고리

Living room 22
Living rooms, sofas, sofa-beds, armchairs, coffee tables, storage furniture, TV solutions

Dining 88
Dining spaces, tables, chairs

Kitchen 112
Kitchens, interior fittings, appliances, integrated lighting, kitchen services

Bedroom 144
Bedrooms, beds, mattresses, comforters and pillows, wardrobes, chests of drawers

Bathroom 200
Bathrooms, organizers and accessories

Children's IKEA 214
Children's rooms, storage, nursery, eating and toys

Workspaces 236
Workspaces, desks, drawer units, work chairs, wall shelves

Home organization 264
Secondary storage, waste sorting, laundry, clothes storage, mirrors, entrance, shoe storage

Decoration 284
Candle holders, vases, plant pots, wall decor

Eating 296
Tableware, glassware

Cooking 304
Kitchen tools, pots and pans, food storage

Textiles 314
Cushions and throws, rugs, bed textiles, curtains and bathroom textiles

Lighting 338
Table lamps, floor lamps, pendant lamps, wall lamps, track systems, shades and bases

Buying guides 350
Product parts and prices 350
Limited Warranties 358

Information
Special Offers 362
Services 364
Store Locations 368
Shopping at the store 369
Return policy 370
IKEA Gift Card 375
Index A-Z 372
Restaurant 373

All catalog prices are maximum prices valid until June 30, 2012

AWS Glue – 데이터 카탈로그

데이터를 쉽게 찾고 관리할 수 있게 해주는



Data Source : S3, JDBC 호환 Database, DynamoDB

크롤러는 자동적으로 데이터 스키마를 찾아서 저장

Apache Hive Metastore와 호환

데이터의 검색과 ETL 작업을 가능

데이터 스키마 정보와 컬럼 레벨 통계 정보를 포함

- S3 Data Source: 거의 모든 텍스트 유형의 파일 지원
(Parquet, ORC, Avro, JSON, CSV, TSV, Apache Log)

데이터 카탈로그 - 테이블 상세 정보 포함

The screenshot shows the AWS Glue Data Catalog interface for a table named 'simpletweets_json'. The left sidebar lists various AWS Glue services: Data catalog, Tables, Connections, Crawlers, Classifiers, ETL, Jobs, Triggers, Dev endpoints, Tutorials, Add crawler, Explore table, and Add job. The 'Tables' section is currently selected. The main content area displays the table details and its schema.

테이블 속성 (Table Properties):

- Name: simpletweets_json
- Description: analytics
- Database: analytics
- Classification: json
- Location: s3://gluesampleddata/simpletweets.json
- Connection: S3Crawler
- Deprecated: No
- Last updated: Thu Aug 10 16:25:24 GMT-700 2017

Properties:

- sizeKey: 456580
- objectCount: 1
- UPDATED_BY_CRAWLER
- S3Crawler
- CrawlerSchemaSerializerVersion: 1.0
- recordCount: 1001
- averageRecordSize: 456
- CrawlerSchemaDeserializerVersion: 1.0
- compressionType: none
- typeOfData: file

테이터 분포 통계 (Data Distribution Statistics):

Jobs, Triggers, Dev endpoints are highlighted with orange arrows.

테이블 스키마 (Table Schema):

	Column name	Data type
1	entities	struct
2	id	bigint
3	retweeted	boolean
4	text	string
5	user	struct

중첩 필드 구조 (Nested Field Structure):

The 'user' struct field is expanded in a modal window titled 'user schema details'.

```
user schema details
STRUCT
contributors_enabled:BOOLEAN
description:STRING
favourites_count:INT
followers_count:INT
friends_count:INT
id:INT
lang:STRING
location:STRING
name:STRING
profile_background_tile:BOOLEAN
```

Transforming Data

Over 90% of ETL jobs in the cloud are **hand-coded**

Which is good ...

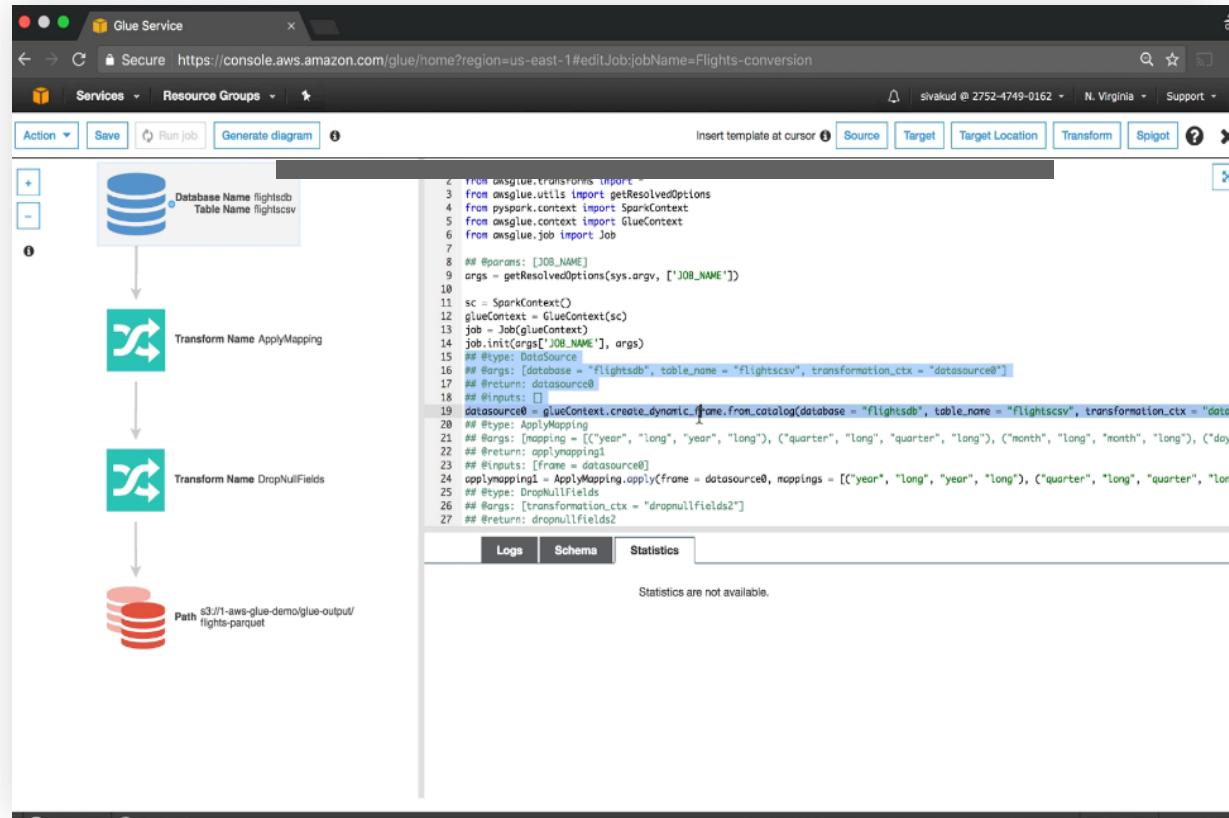
- Flexible
- Powerful
- Unit Tests
- CI/CD
- Developer Tools ...

... but also **bad!**

- Brittle
- Error-Prone
- Laborious
- Sources Change
- Schemas Change
- Volume Changes
- **EVERYTHING KEEPS CHANGING !!!**

AWS Glue - ETL 서비스

Job 스크립트 작성과 실행을 쉽게 도와주는



서비스 데이터 변환작업

Apache Spark 기반

클릭 몇번으로 생성되는 ETL code

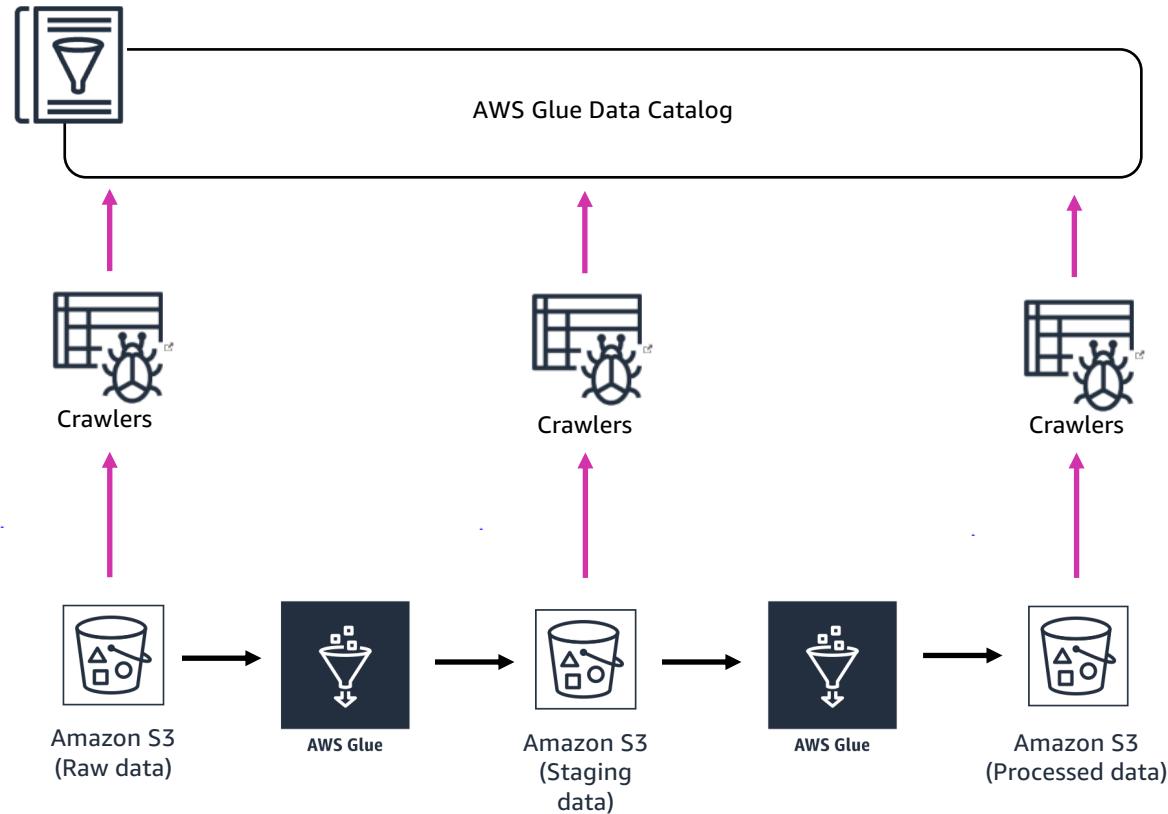
수정 / 추가가 가능한 PySpark과 Scala 코드

반복 일정과 이벤트에 따른 Job 스케줄링

Zeppelin, PyCharm 등 익숙한 환경에서 수정, 디버그, 테스트가 가능하도록 Dev Endpoint 제공 (Python 2.7 지원)

- Sample ETL Codes : <https://github.com/awslabs/aws-glue-samples>
- About Dev Endpoint : https://docs.aws.amazon.com/ko_kr/glue/latest/dg/dev-endpoint.html

AWS Glue - 데이터 클랜징, 준비, 카탈로그



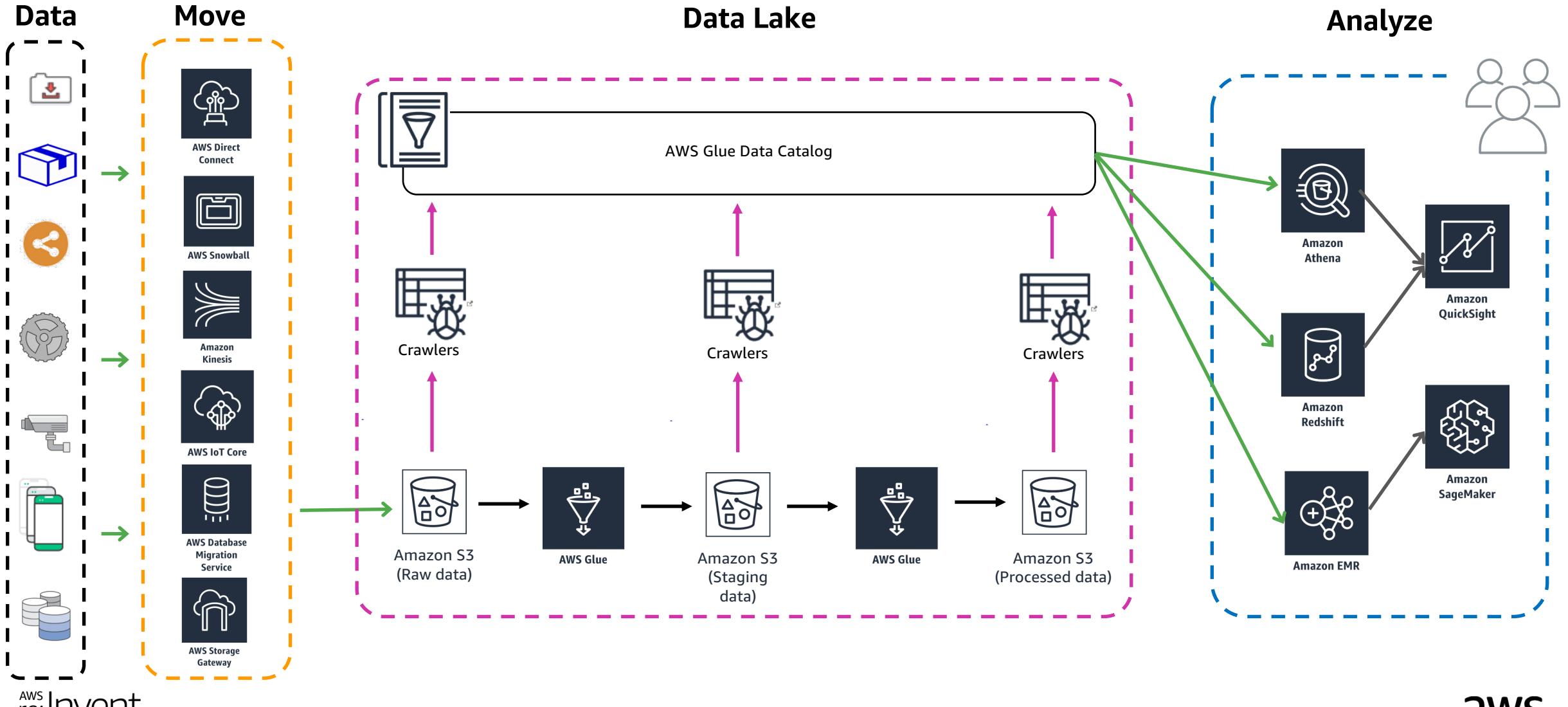
AWS Glue Data Catalog - 데이터 레이크에 대한 단일된 뷰

Automatically **discovers** data and stores schema
Makes data **searchable**, and available for ETL
Contains table definitions and custom metadata

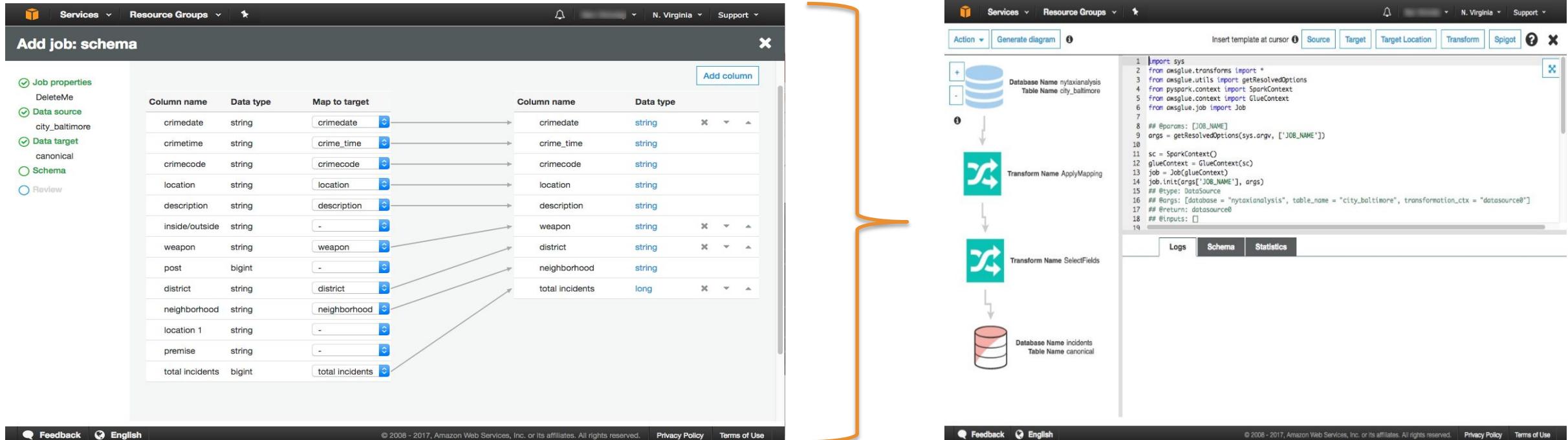
AWS Glue ETL job - 데이터 클랜징, 변환 등의 데이터 프로세싱

Serverless Apache Spark environment
Use Glue ETL libraries or bring your own code
Write code in **Python or Scala**
Call any AWS API using the AWS boto3 SDK

AWS Glue가 데이터 레이크의 중심 역할



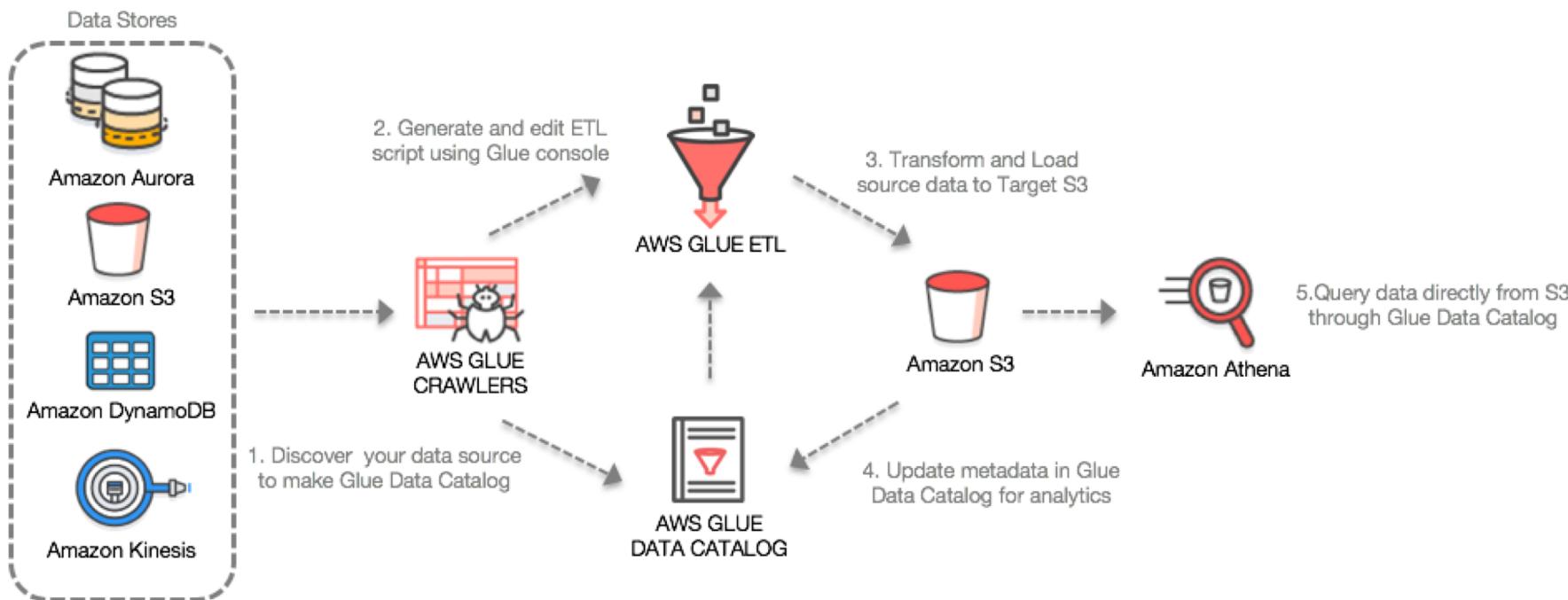
Job 생성 - 콘솔에서 코드 생성



1. 콘솔에서 컬럼 단위의 맵핑을 수정하면
2. Glue에서 자동적으로 데이터 변환 그래프와 **PySpark** (또는 **Scala**) 코드를 생성, 직접 수정 가능
3. 직접 사용하는 노트북 서비스로 **Dev Endpoint** 이용하여 코딩 가능

AWS Glue 를 활용한 ETL 주요 단계

1. 데이터 소스에 Crawler를 통해 데이터 카탈로그 생성
2. 컬럼 단위의 맵핑을 통해 자동 코드 생성
3. 편리한 환경에서 자유롭게 코드 수정 및 테스트(/w Dev-Endpoint)
4. 실제 운영 환경에서 Job 스케줄링 및 실행



Step 1 : Crawler 실행

AWS Glue Console - GitHub Events Data Crawler

Table Details:

- Name: githubevents_data
- Description: gitarchive
- Database: gitarchive
- Classification: json
- Location: s3://glue-sample-datasets/examples/data/
- Connection: N/A
- Deprecated: No
- Last updated: Wed Nov 22 07:52:09 GMT-800 2017
- Input format: org.apache.hadoop.mapred.TextInputFormat
- Output format: org.apache.hadoop.hive.ql.io.HiveIgnoreKeyTextOutputFormat
- Serde serialization lib: org.openx.data.jsonserde.JsonSerDe

Serde parameters: paths: actor,created_at,id,org,payload,public,repo,type

Table properties:

CrawlerSchemaSerializerVersion	1.0	recordCount	27833145	averageRecordSize	2423
CrawlerSchemaDeserializerVersion	1.0	compressionType	gzip	typeOfData	file

Schema:

Column name	Data type	Key
1	id	string
2	type	string
3	actor	struct
4	repo	struct
5	payload	struct
6	public	boolean
7	created_at	string
8	org	struct
9	year	string
10	month	string
11	day	string

Showing: 1 - 11 of 11

Feedback English (US) Privacy Policy Terms of Use

버킷 / 폴더 / 파일유형에 따라
테이블 메타정보 저장

Hive metastore 호환 파티션

AWS Glue Console - GitHub Events Data Crawler

Table Details:

- Last updated: 22 Nov 2017
- Table: Version (Current version)

Partitions:

year	month	day	View files	View properties
2017	02	15	View files	View properties
2017	03	12	View files	View properties
2017	05	17	View files	View properties
2017	10	12	View files	View properties
2017	12	18	View files	View properties
2017	01	09	View files	View properties
2017	03	07	View files	View properties
2017	06	28	View files	View properties

Showing: 1 - 100

Feedback English (US) Privacy Policy Terms of Use

Step 2 : 자동 ETL 코드 생성

AWS Glue Console

Secure | https://console.aws.amazon.com/glue/home?region=us-east-1#addJob:

Add job

Job properties
github_2_csv

Data source
2015

Data target
s3://glue-sample-ta...

Schema

Review

Map the source columns to target columns.

Verify the mappings created by AWS Glue. Change mappings by choosing other columns with Map to target. You can Clear all mappings and Reset to default AWS Glue mappings. AWS Glue generates your script with the defined mappings.

Source	Target			
Column name	Data type	Map to target	Column name	Data type
id	string	id	id	string
type	string	type	type	string
actor	struct	-	actor	string
id	int	-	repo	string
login	string	actor	payload	struct
gravatar_id	string	-		
url	string	-		
avatar_url	string	-		
repo	struct	-		
id	int	-		
name	string	repo		
url	string	-		
payload	struct	payload		
public	boolean	-		

Add column Clear Reset

Feedback English (US)

© 2008 - 2017, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

데이터 소스와 타겟 지정

컬럼 레벨의 스키마 변경 가능

Step 2 : 자동 ETL 코드 생성

Job: github_2_csv Action ▾ Save Run job Generate diagram ⓘ Insert template at cursor ⓘ Source Target Target Location Transform Spigot ⓘ X

Database Name gitarchive Table Name 2015

Transform Name ApplyMapping

Transform Name ResolveChoice

Transform Name DropNullFields

Path s3://glue-sample-target/output-dir

```
1 import sys
2 from awsglue.transforms import *
3 from awsglue.utils import getResolvedOptions
4 from pyspark.context import SparkContext
5 from awsglue.context import GlueContext
6 from awsglue.job import Job
7
8 ## @params: [JOB_NAME]
9 args = getResolvedOptions(sys.argv, ['JOB_NAME'])
10
11 sc = SparkContext()
12 glueContext = GlueContext(sc)
13 spark = glueContext.spark_session
14 job = Job(glueContext)
15 job.init(args['JOB_NAME'], args)
16 ## @type: DataSource
17 ## @args: [database = "gitarchive", table_name = "2015", transformation_ctx = "datasource0"]
18 ## @return: datasource0
19 ## @inputs: []
20 datasource0 = glueContext.create_dynamic_frame.from_catalog(database = "gitarchive", table_name = "2015", transformation_ctx = "datasource0")
21 ## @type: ApplyMapping
22 ## @args: [mapping = [{"id": "string", "id": "string"}, {"type": "string", "type": "string"}, {"actor.login": "string", "actor": "string"}, ("repo.name": "string", "repo.type": "string"}], transformation_ctx = "applymapping1"]
23 ## @return: applymapping1
24 ## @inputs: [frame = datasource0]
25 applymapping1 = ApplyMapping.apply(frame = datasource0, mappings = [{"id": "string", "id": "string"}, {"type": "string", "type": "string"}, {"actor.login": "string", "actor": "string"}, ("repo.name": "string", "repo.type": "string"}], transformation_ctx = "applymapping1")
26 ## @type: ResolveChoice
27 ## @args: [choice = "make_struct", transformation_ctx = "resolvechoice2"]
28 ## @return: resolvechoice2
29 ## @inputs: [frame = applymapping1]
30 resolvechoice2 = ResolveChoice.apply(frame = applymapping1, choice = "make_struct", transformation_ctx = "resolvechoice2")
31 ## @type: DropNullFields
32 ## @args: [transformation_ctx = "dropnullfields3"]
33 ## @return: dropnullfields3
34 ## @inputs: [frame = resolvechoice2]
35 dropnullfields3 = DropNullFields.apply(frame = resolvechoice2, transformation_ctx = "dropnullfields3")
36 ## @type: DataSink
37 ## @args: [connection_type = "s3", connection_options = {"path": "s3://glue-sample-target/output-dir"}, format = "parquet"]
38 ## @return: datasink4
39 ## @inputs: [frame = dropnullfields3]
40 datasink4 = glueContext.write_dynamic_frame.from_options(frame = dropnullfields3, connection_type = "s3", connection_options = {"path": "s3://glue-sample-target/output-dir"}, format = "parquet")
41 job.commit()
```

Initialize job bookmark

Annotations for graphical DAG

Read Dynamic Frame from source

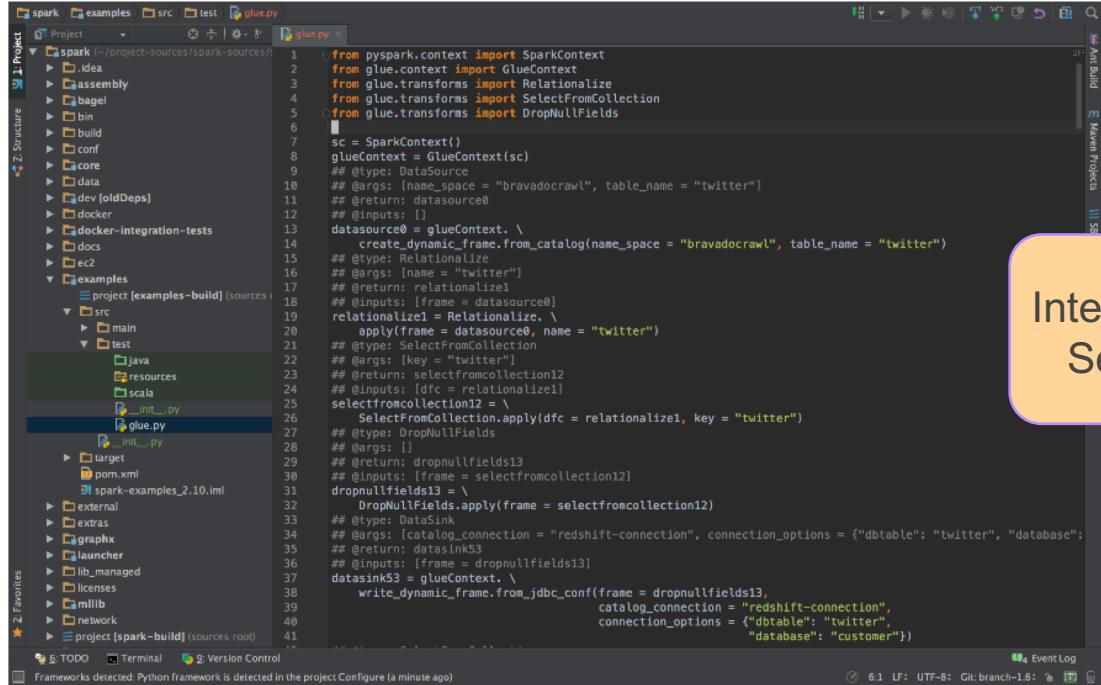
Data transformation + data cleaning functions

Write Dynamic Frame to sink

Commit job bookmark

Logs Schema

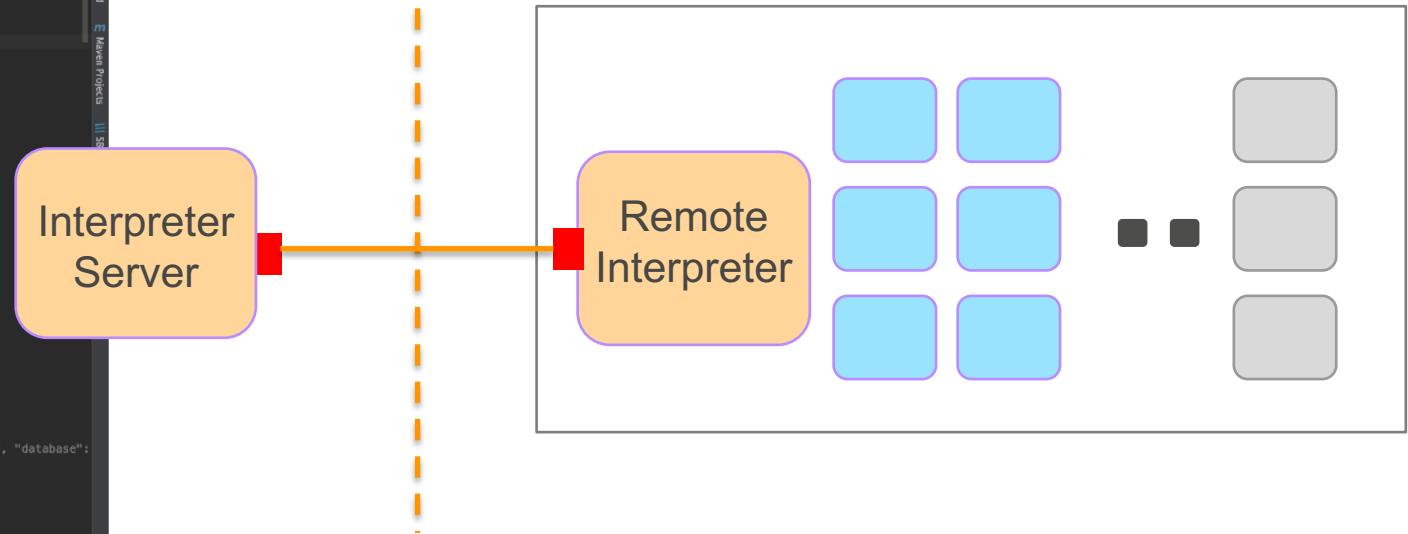
Step 3 : 코드 수정 및 데이터 탐색(/w Dev-Endpoint)



```
from pyspark.context import SparkContext
from glue.context import GlueContext
from glue.transforms import Relationalize
from glue.transforms import SelectFromCollection
from glue.transforms import DropNullFields

sc = SparkContext()
glueContext = GlueContext(sc)
## @type: DataSource
## @args: [name_space = "bravadocrawl", table_name = "twitter"]
## @return: datasource0
## @inputs: []
datasource0 = glueContext.create_dynamic_frame.from_catalog(name_space = "bravadocrawl", table_name = "twitter")
## @type: Relationalize
## @args: [name = "twitter"]
## @return: relationalize1
## @inputs: [frame = datasource0]
relationalize1 = Relationalize.apply(frame = datasource0, name = "twitter")
## @type: SelectFromCollection
## @args: [key = "twitter"]
## @return: selectfromcollection12
## @inputs: [dfc = relationalize1]
selectfromcollection12 = SelectFromCollection.apply(df = relationalize1, key = "twitter")
## @type: DropNullFields
## @args: []
## @return: dropnullfields13
## @inputs: [frame = selectfromcollection12]
dropnullfields13 = DropNullFields.apply(frame = selectfromcollection12)
## @type: DataSink
## @args: [catalog_connection = "redshift-connection", connection_options = {"dbtable": "twitter", "database": "customer"}]
## @return: datasink53
## @inputs: [frame = dropnullfields13]
datasink53 = glueContext.write_dynamic_frame.from_jdbc_conf(frame = dropnullfields13,
    catalog_connection = "redshift-connection",
    connection_options = {"dbtable": "twitter",
        "database": "customer"})
```

AWS Glue Spark environment



AWS Glue의 Dev-Endpoint를 이용해서 자신의 IDE 환경으로 접속

인터랙티브한 ETL code 개발, 디버그, 테스트 환경 구축

Step 3 : 코드 수정 및 데이터 탐색(/w Dev-Endpoint)

Zeppelin Notebook Job Search your Notes admin default

reInvent Head FINISHED

```
github_requests = glueContext.create_dynamic_frame_from_catalog(database = "gitarchive", table_name="2015")
github_requests.printSchema()
github_df = github_requests.toDF()
github_df.createOrReplaceTempView("github")
```

root

```
|-- id: string
|-- type: string
|-- actor: struct
|   |-- id: int
|   |-- login: string
|   |-- gravatar_id: string
|   |-- url: string
|   |-- avatar_url: string
|-- repo: struct
|   |-- id: int
|   |-- name: string
|   |-- url: string
```

Took 17 sec. Last updated by admin at November 20 2017, 12:05:56 PM. (outdated)

%sql

```
select type, repo.name, count(id) as c from github group by type, repo.name having c > 15 order by c desc
```

settings

Grouped Stacked

Event Type	Count	Repository
PushEvent	~140	KenanSulayman/hearth...
PullRequestReviewCommentEvent	~80	zendframework/zf2
DeleteEvent	~40	OpenLightingProject/...
WatchEvent	~20	candelorimarco/kerne...
IssuesEvent	~15	sakai-mirror/melete
		quhezheng/HomeIP
		mquinson/PLM-data
		floscher/linguist
		lcrespom/ycs
		oceanjack/oceanjack....
		taylorhxu/taylorxu....

Took 5 sec. Last updated by admin at November 20 2017, 12:24:18 PM.

Apache Zeppelin 노트북을 통한 AWS Glue Dev-Endpoint 접속

인터랙티브한 데이터 쿼리 및 탐색이 가능

https://docs.aws.amazon.com/ko_kr/glue/latest/dg/dev-endpoint-tutorial-EC2-notebook.html

Step 4 : Job 스케줄링 및 실행

Add trigger

Trigger properties
github_2_json_daily (Schedule)

Jobs to start

Review all steps

Set up your trigger's properties

Name: github_2_csv_daily

Trigger type:
 Schedule Jobs completed On-demand
Choose Schedule to fire the trigger on a timer, Jobs completed to fire the trigger when a job completes, and On-demand to fire the trigger immediately when started.

Frequency: Daily

Time: 00:00 UTC
00:00 UTC
00:30 UTC
01:00 UTC
01:30 UTC
02:00 UTC
02:30 UTC

이벤트 유형과 빈도 설정

Add trigger

Trigger properties
github_2_csv_daily (Schedule)

Jobs to start

Review all steps

Choose jobs to trigger

Choose jobs to start when this trigger fires.

All Jobs Jobs to start
Showing: 1 - 42 < > Showing: 1 - 1 < >

Job	Action
titanic_job	Add
2017_S3_to_S3	Add
github_2_csv	Add
TSVToRedshift	Add
m312	Add

Parameters passed to job github_2_csv when started
(Optional) Add parameters to override the default parameters passed to this job when started by this trigger.

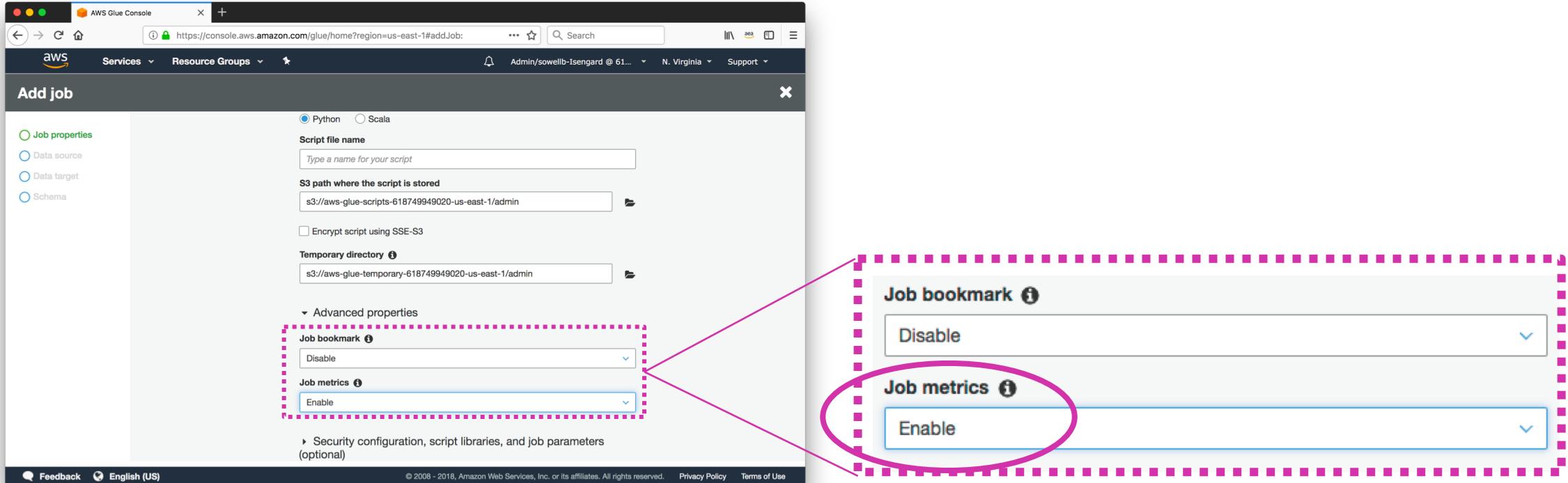
Job bookmark i

Enable:

Key	Value
<input type="text"/>	<input type="text"/>

필요한 파라미터 설정

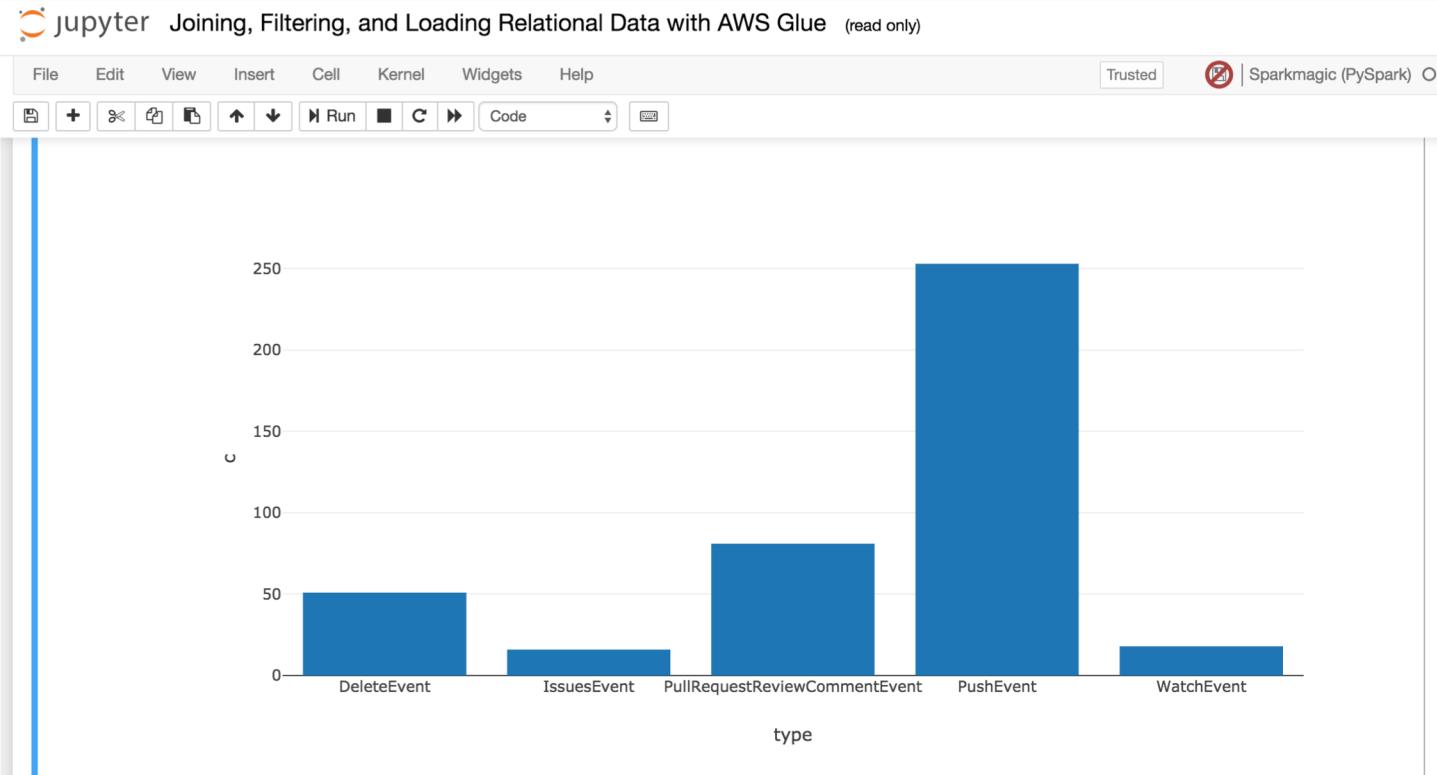
AWS Glue job metrics



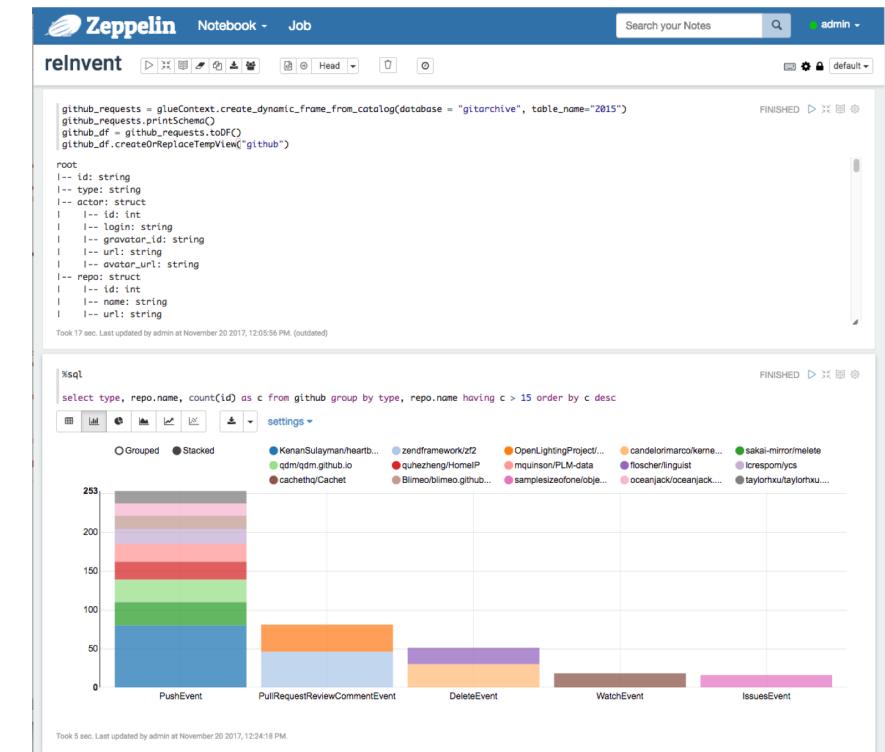
- Metrics can be enabled in the AWS Command Line Interface (AWS CLI) and AWS SDK by passing `--enable-metrics` as a job parameter key.

Beyond ETL: data science and exploration

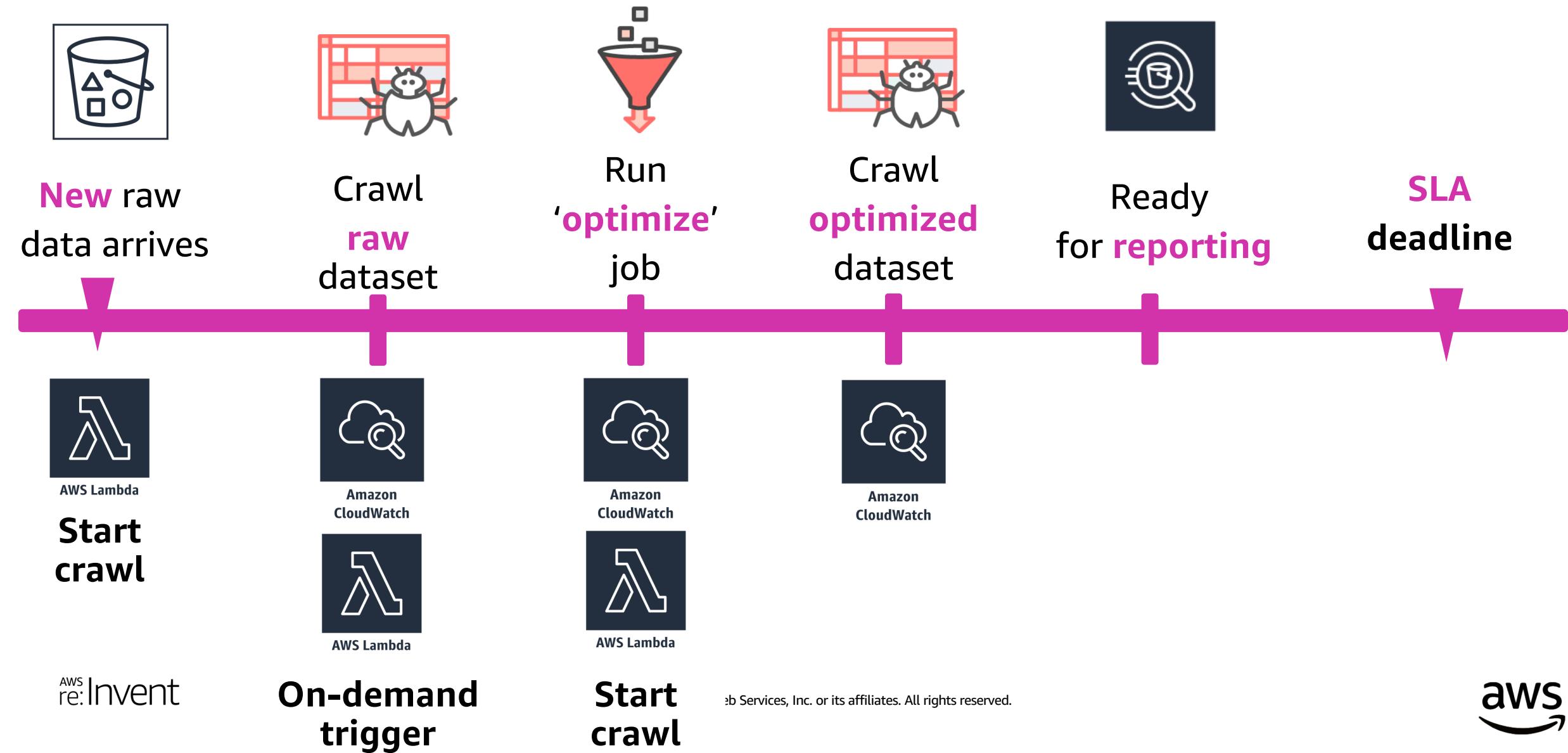
Amazon SageMaker integration



Apache Zeppelin notebook

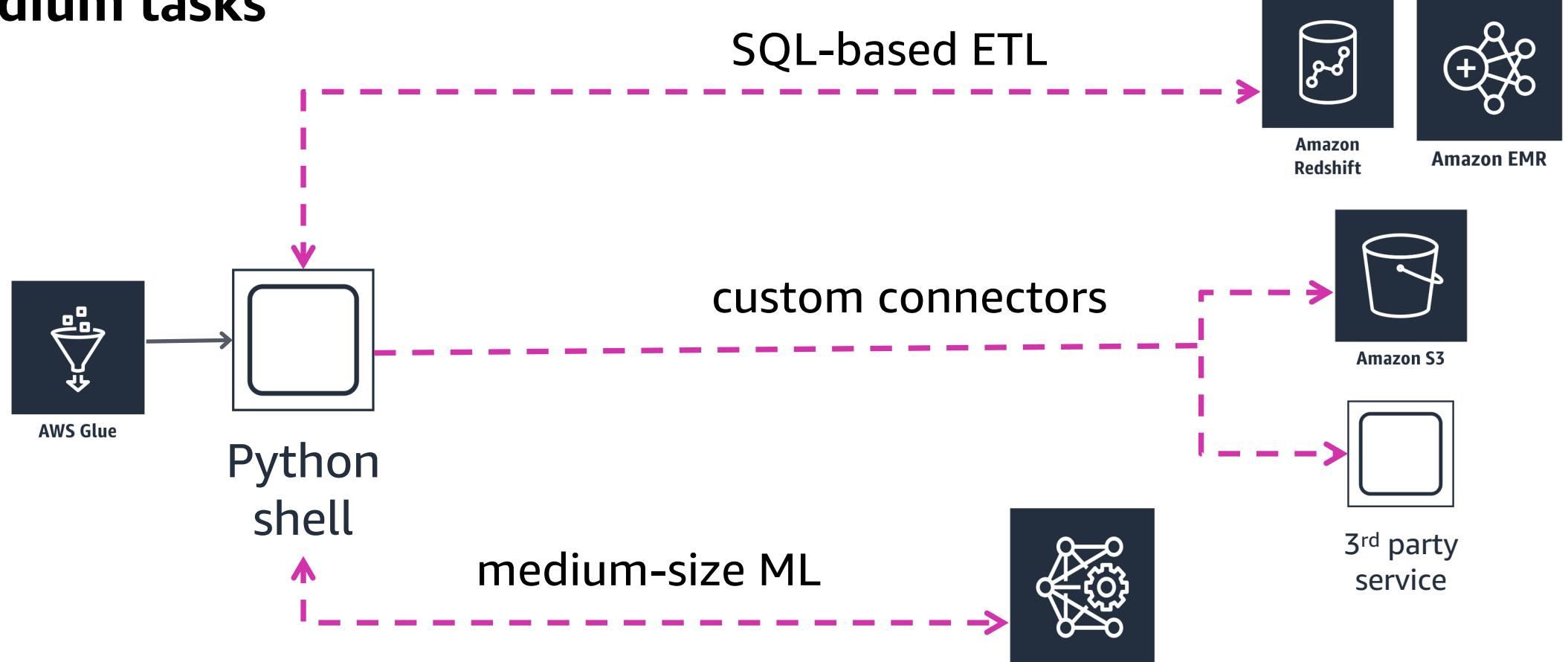


Example event-driven workflow



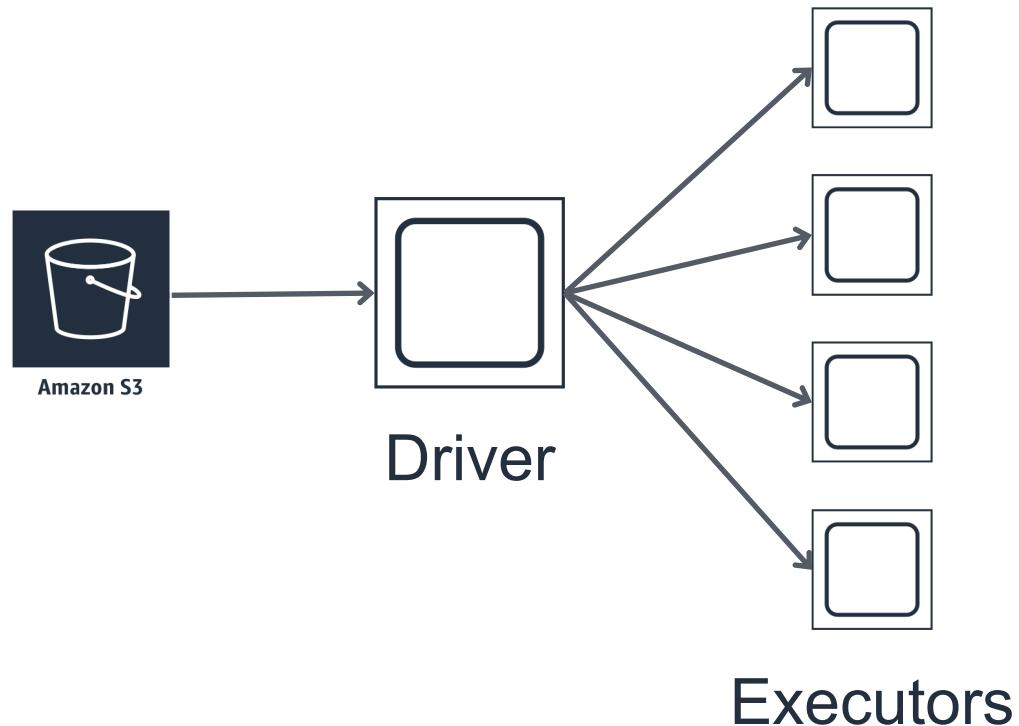
Announcing a new job type: Python shell

A new ***cost-effective*** ETL primitive for small to medium tasks



AWS Glue ETL

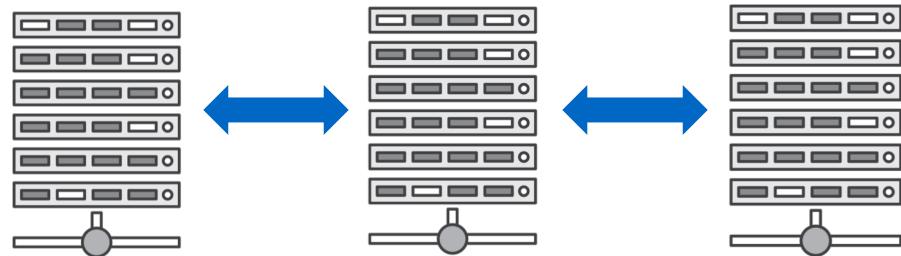
AWS Glue execution model: data partitions



- Apache Spark and AWS Glue are *data parallel*.
- Data is divided into *partitions* that are processed concurrently.
- 1 stage x 1 partition = 1 *task*

Overall throughput is limited by the number of partitions

Apache Spark and AWS Glue ETL



What is Apache Spark?

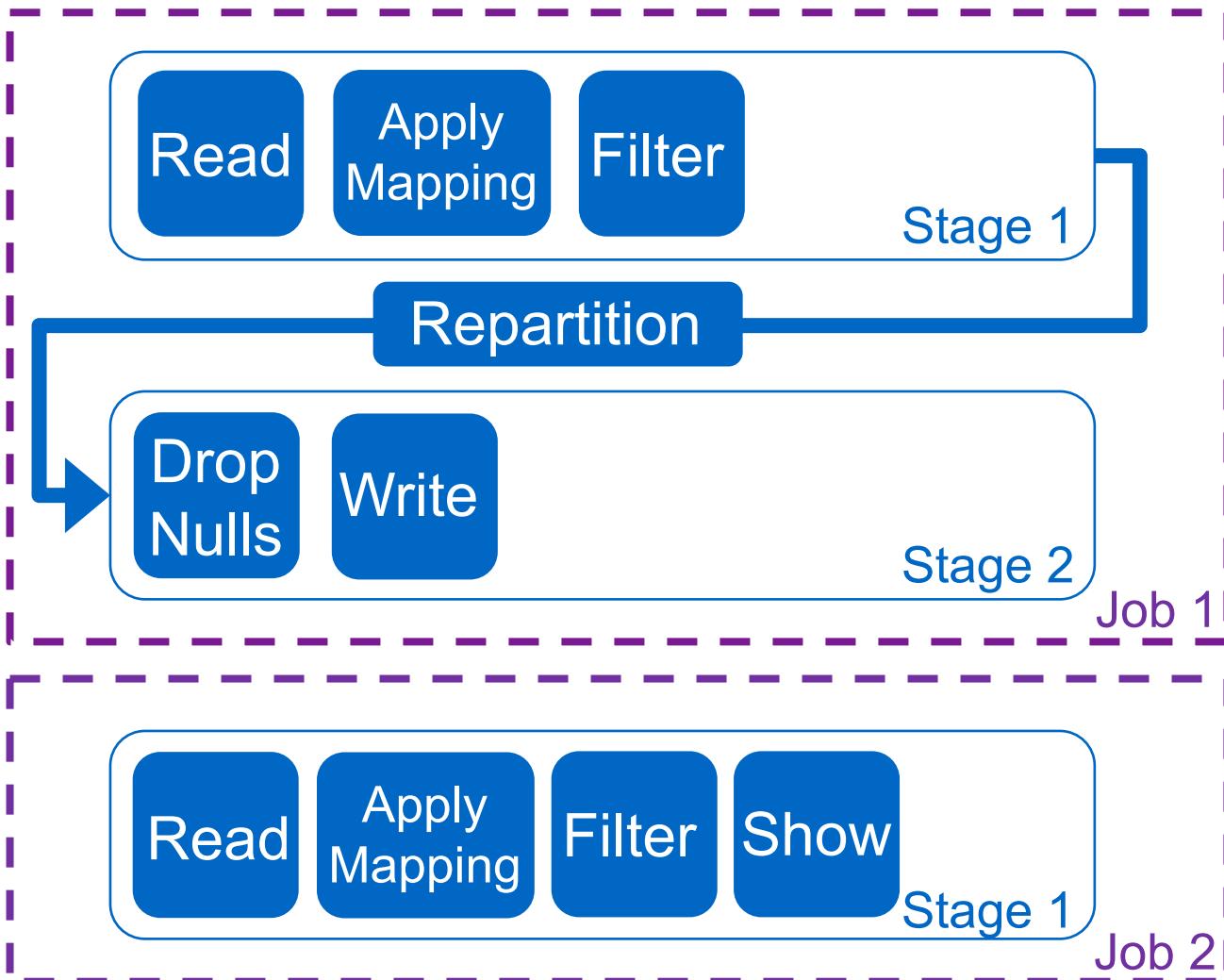
- Parallel, scale-out data processing engine
- Fault-tolerance built-in
- Flexible interface: Python scripting, SQL
- Rich eco-system: ML, Graph, analytics, ...

SparkSQL	AWS Glue ETL
Dataframes	DynamicFrames
Spark core: RDDs	

AWS Glue ETL libraries

- Integration: Data Catalog, job orchestration, code-generation, job bookmarks, S3, RDS
- ETL transforms, more connectors & formats
- New data structure: DynamicFrames

AWS Glue execution model: jobs and stages

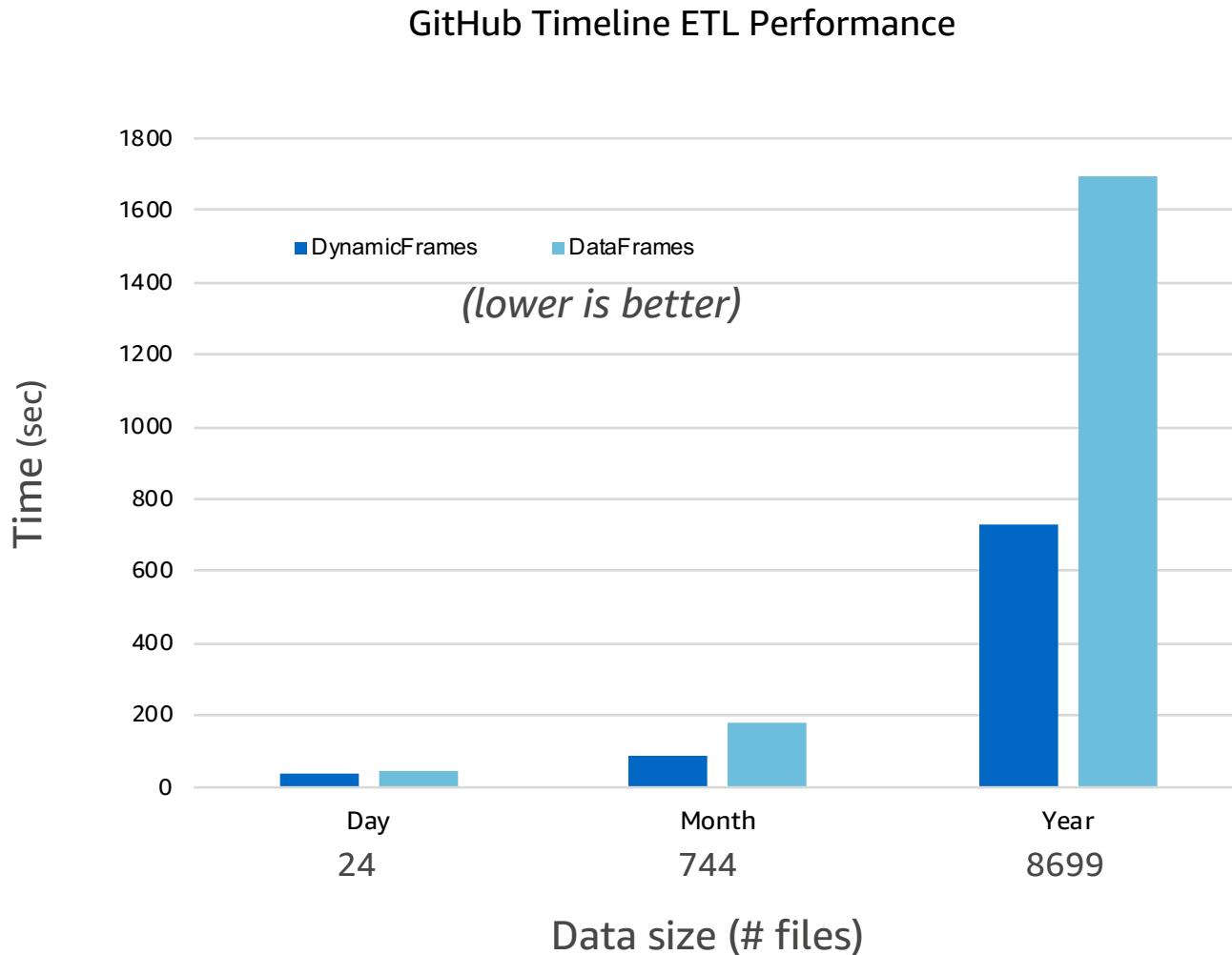


```
df = glueContext.getSource(...)  
applyMapping = df.applyMapping(...)  
filter = applyMapping.filter(...)  
repartition = filter.repartition(10)  
dropNulls = repartition.dropNulls()  
glueContext.getSink(...)\n    .writeDynamicFrame(dropNulls)
```

```
filter.show()
```

Jobs

Performance: AWS Glue ETL



Configuration

10 DPUs

Apache Spark 2.1.1

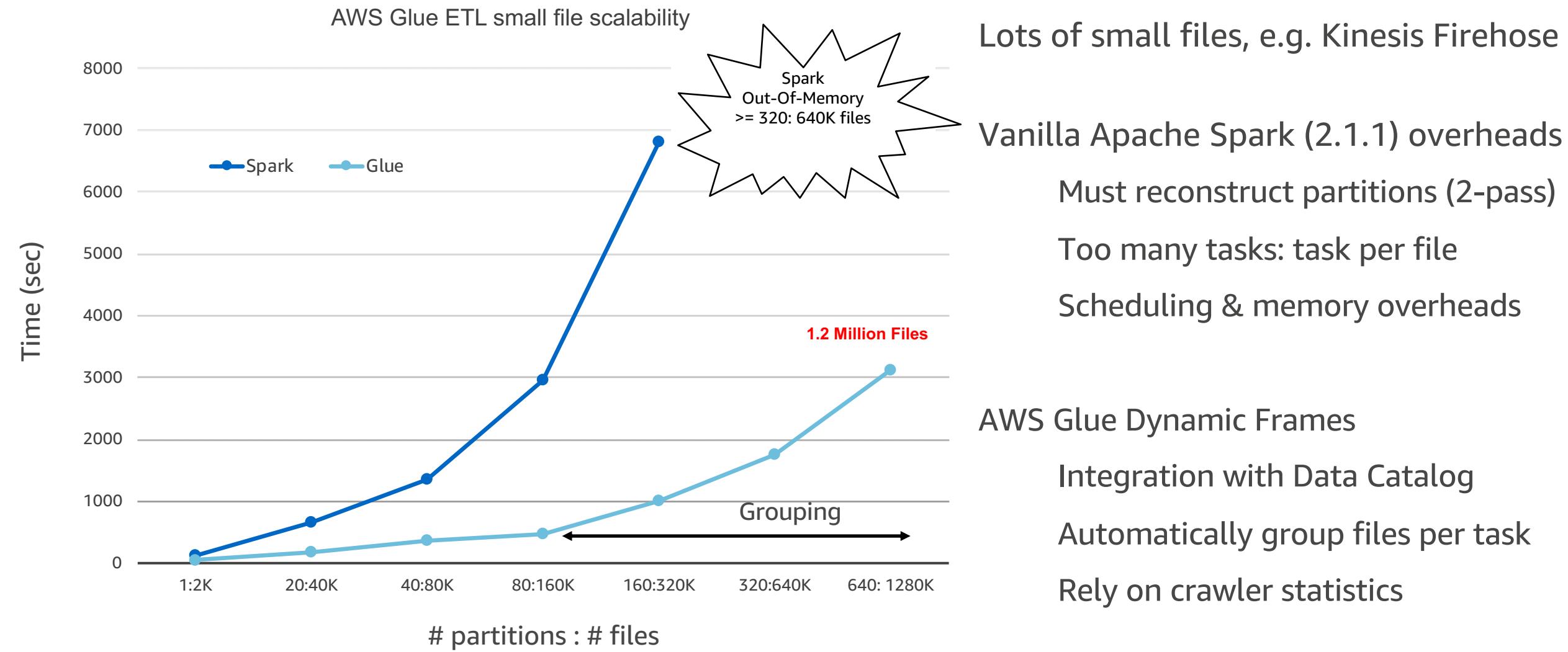
Workload

JSON to CSV

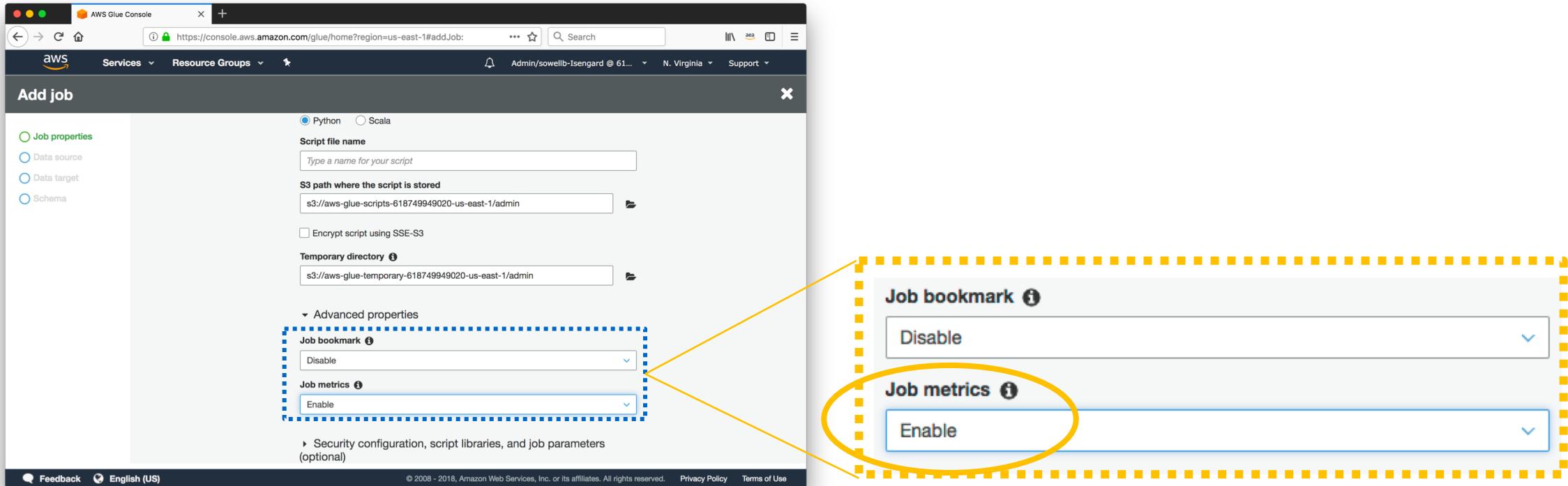
Filter for Pull events

On average: **2x performance improvement**

Performance: Lots of small files



Enabling job metrics

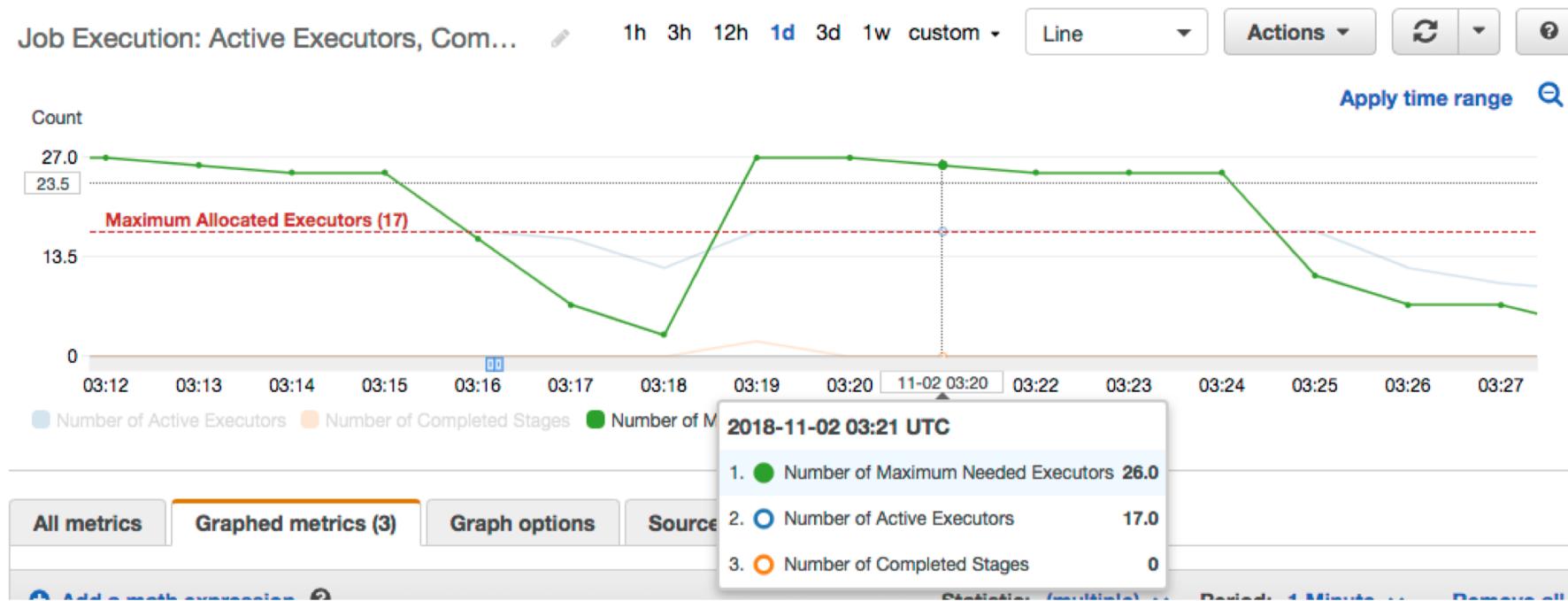


- Metrics can be enabled in the CLI/SDK by passing --enable-metrics as a job parameter key.

Example: optimizing parallelism

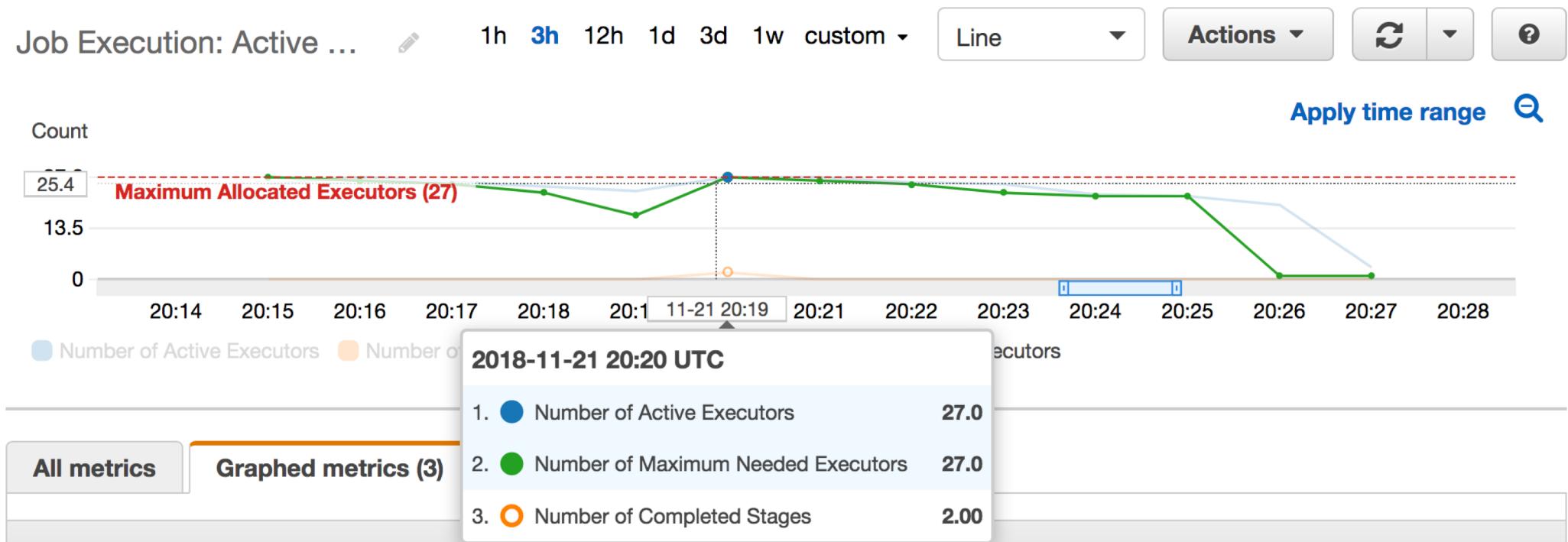
Processing large, split-able bzip2 files.

With 10 DPU, metric *maximum needed executors* shows room for scaling.



Example: optimizing parallelism

With 15 DPU, *active executors* closely tracks *maximum needed executors*.

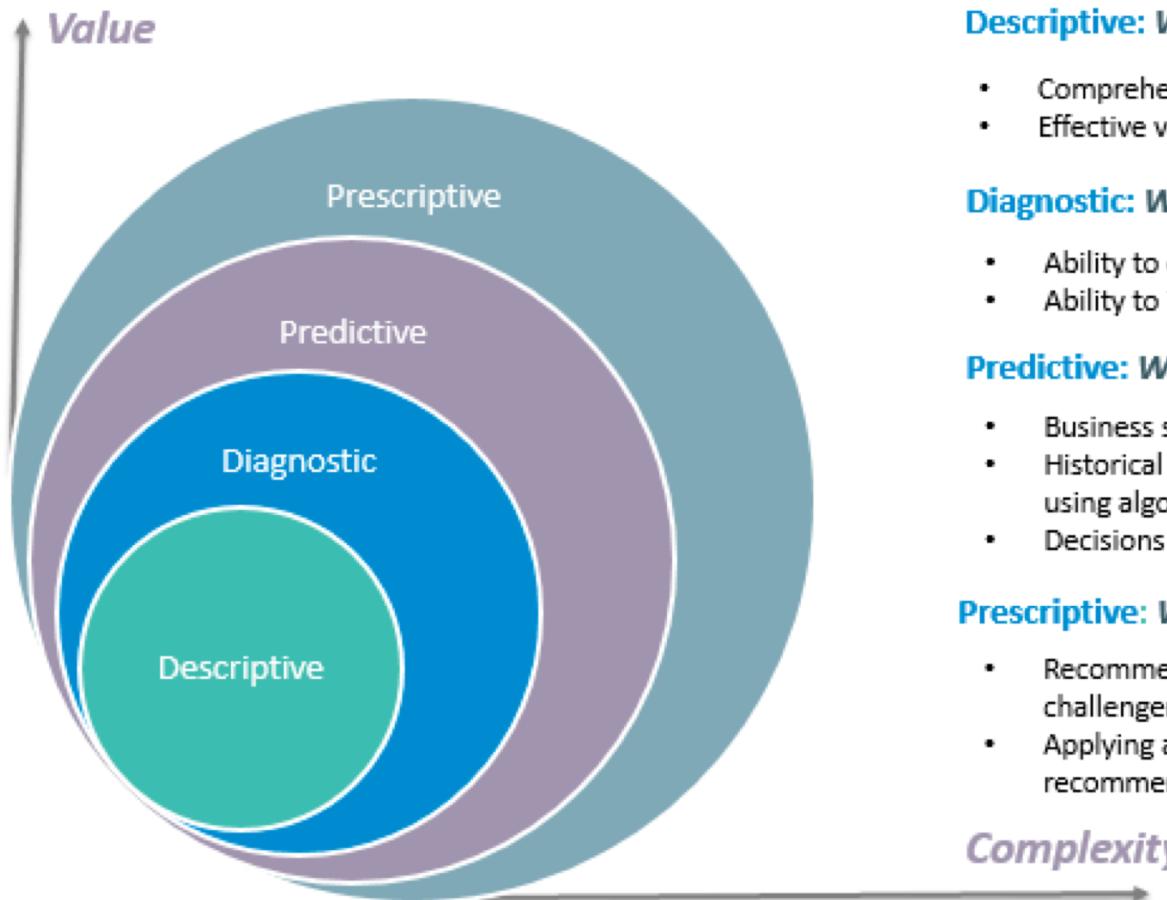


Data Analytics

데이터 분석이란?

데이터 분석은 의사결정에 필요한 의미있는 정보와 근거들을 얻어내기 위해 데이터를 정리, 가공, 변환, 조사, 모델링 하는 과정을 뜻한다.(Wikipedia)

4 types of Data Analytics



What is the data telling you?

Descriptive: *What's happening in my business?*

- Comprehensive, accurate and live data
- Effective visualisation

Diagnostic: *Why is it happening?*

- Ability to drill down to the root-cause
- Ability to isolate all confounding information

Predictive: *What's likely to happen?*

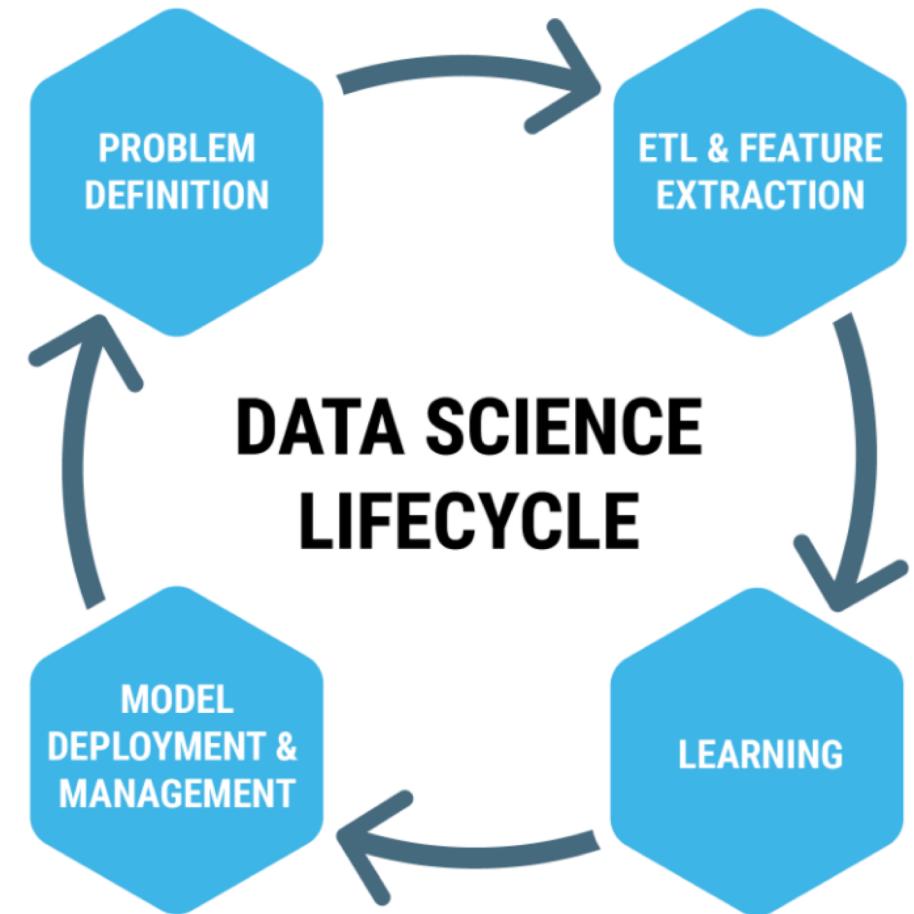
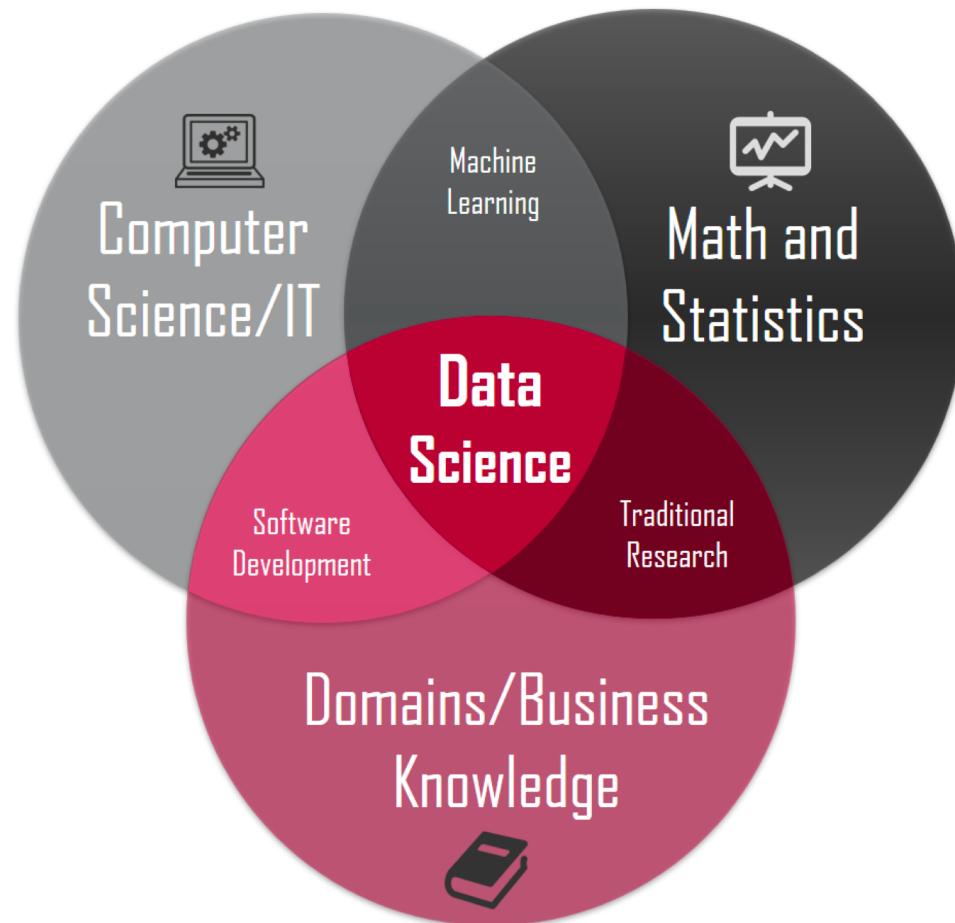
- Business strategies have remained fairly consistent over time
- Historical patterns being used to predict specific outcomes using algorithms
- Decisions are automated using algorithms and technology

Prescriptive: *What do I need to do?*

- Recommended actions and strategies based on champion / challenger testing strategy outcomes
- Applying advanced analytical techniques to make specific recommendations

데이터 과학이란?

데이터 과학은 다양한 정형, 비정형 데이터에서 지식과 통찰력을 뽑아내는 행위이고, 통계, 데이터 마이닝, 예측 분석등으로 이어지는 흐름이다.
(Wikipedia)

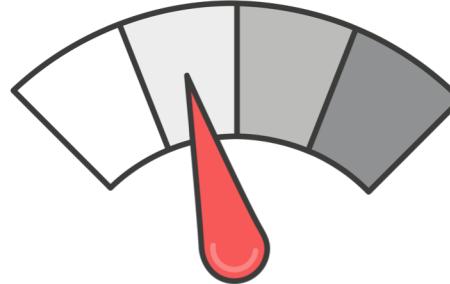


Enabling all Types of Data-Driven Analytics



Retrospective
analysis and
reporting

Past Data



Here-and-now
real-time processing
and dashboards

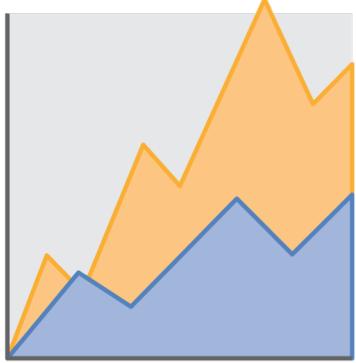
Present Data



Predictions
to enable smart
applications

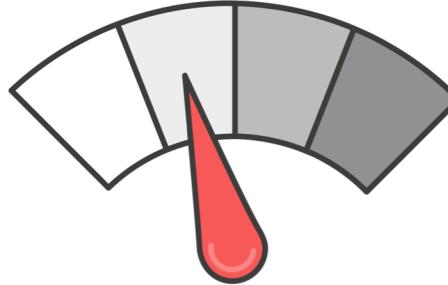
“Future Data”

Enabling all Types of Data-Driven Analytics



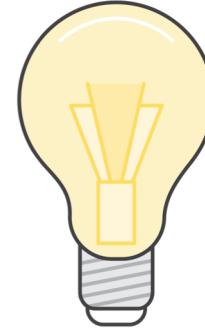
Retrospective
analysis and
reporting

Amazon Redshift
Amazon RDS
Amazon S3
Amazon EMR



Here-and-now
real-time processing
and dashboards

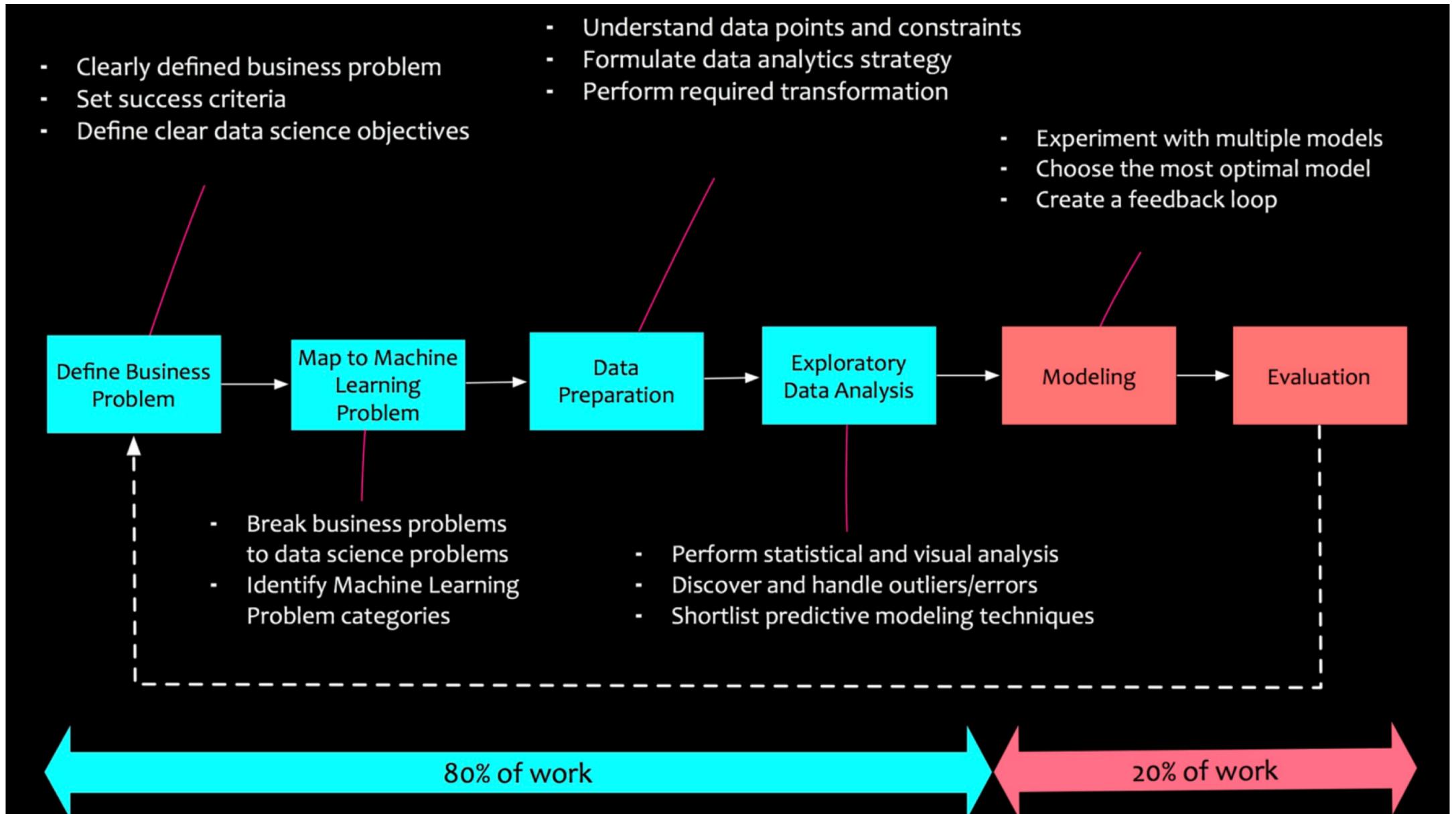
Amazon Kinesis
AWS Lambda
Amazon DynamoDB
Amazon EC2



Predictions
to enable smart
applications

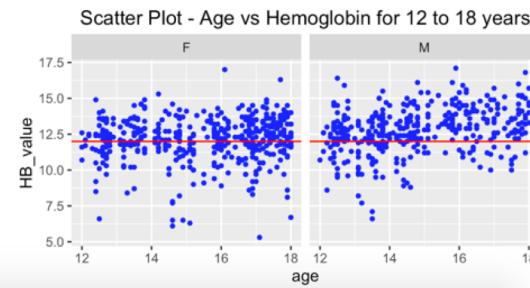
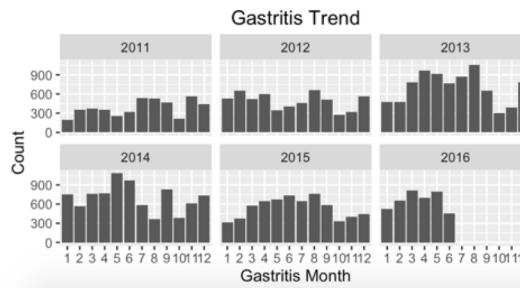
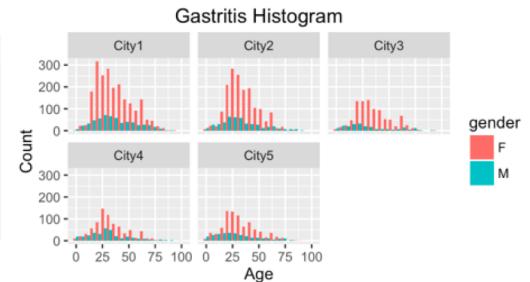
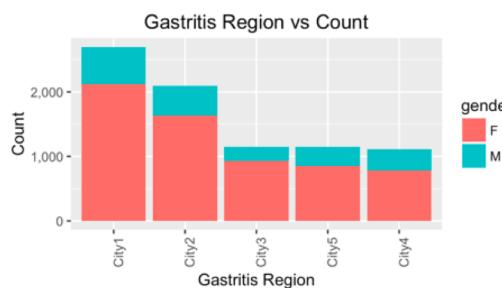
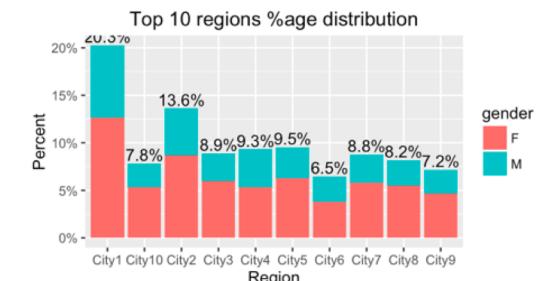
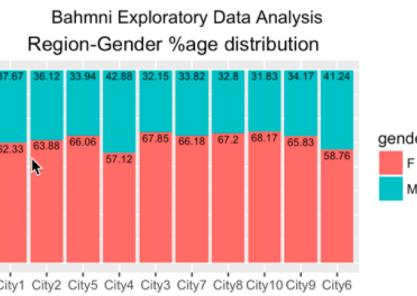
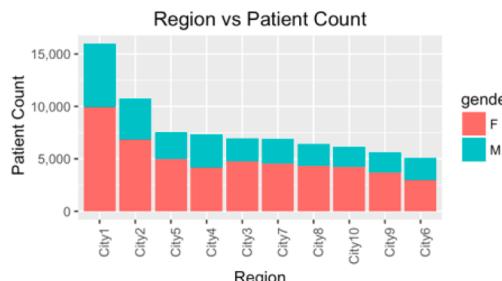
Amazon ML
Amazon Sagemaker

Data Science Process



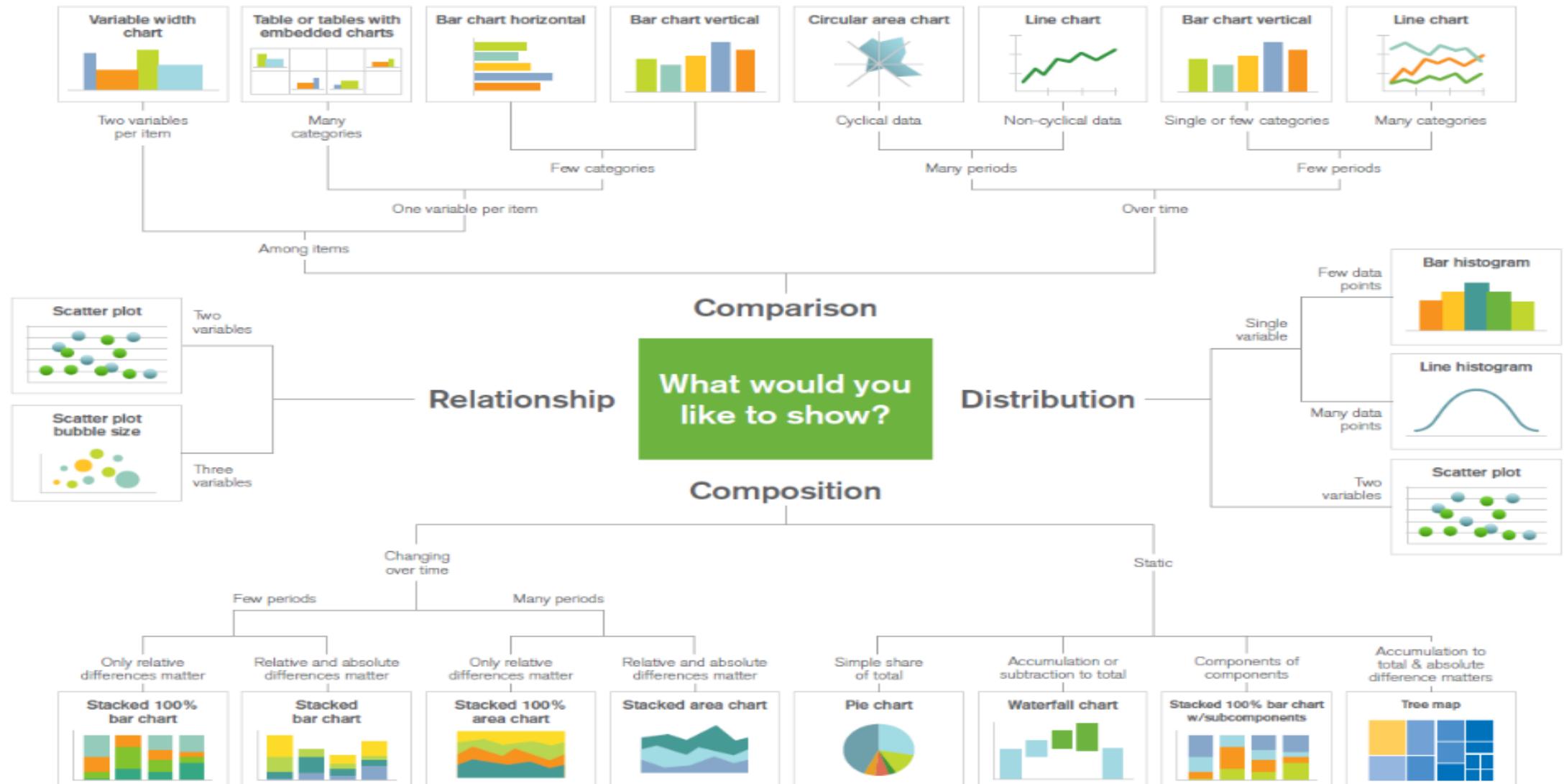
EDA(Exploratory data analysis)

EDA(탐험적 데이터 분석)은 데이터 분석의 초기 단계에 이해하기 위해 다양한 차트를 이용하여 데이터를 표현하고 이해하는 과정이다. 이를 통해 문제 해결에 필요한 데이터를 식별하고, 가설 수립, 알고리즘 선정 등을 할 수 있다.

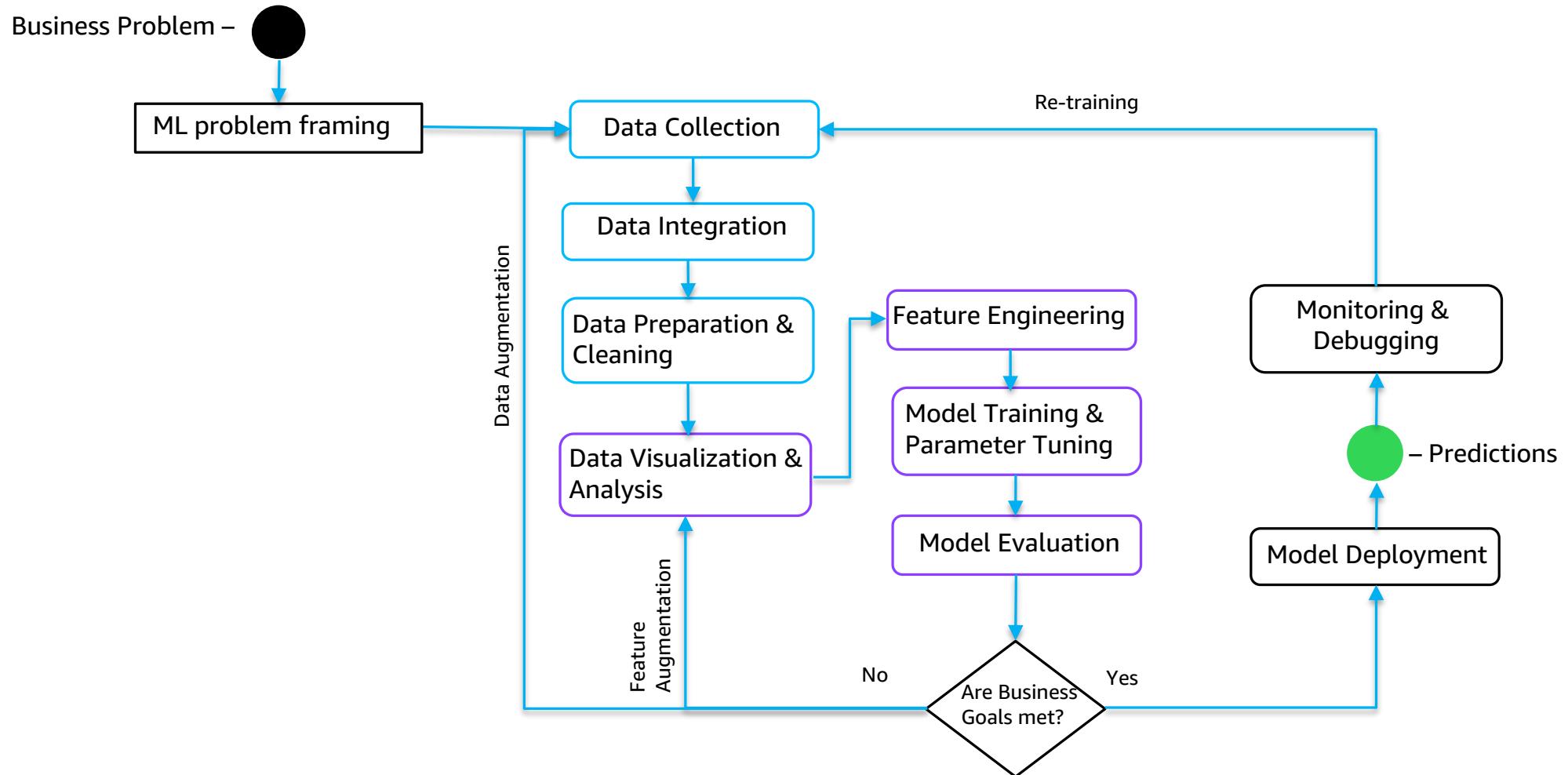


데이터 시각화 방법

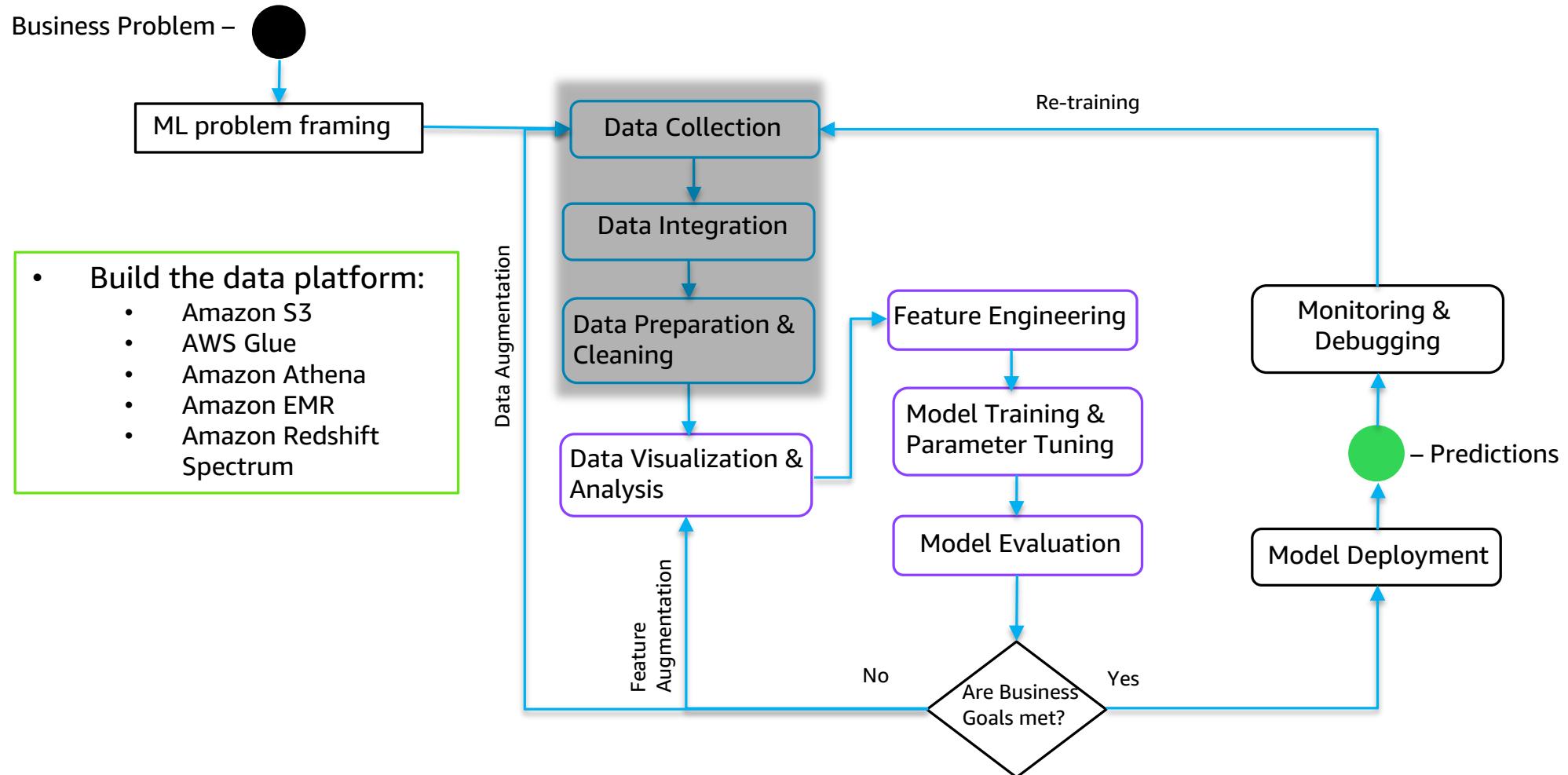
분포, 분류, 관계, 비교, 시간, 공간



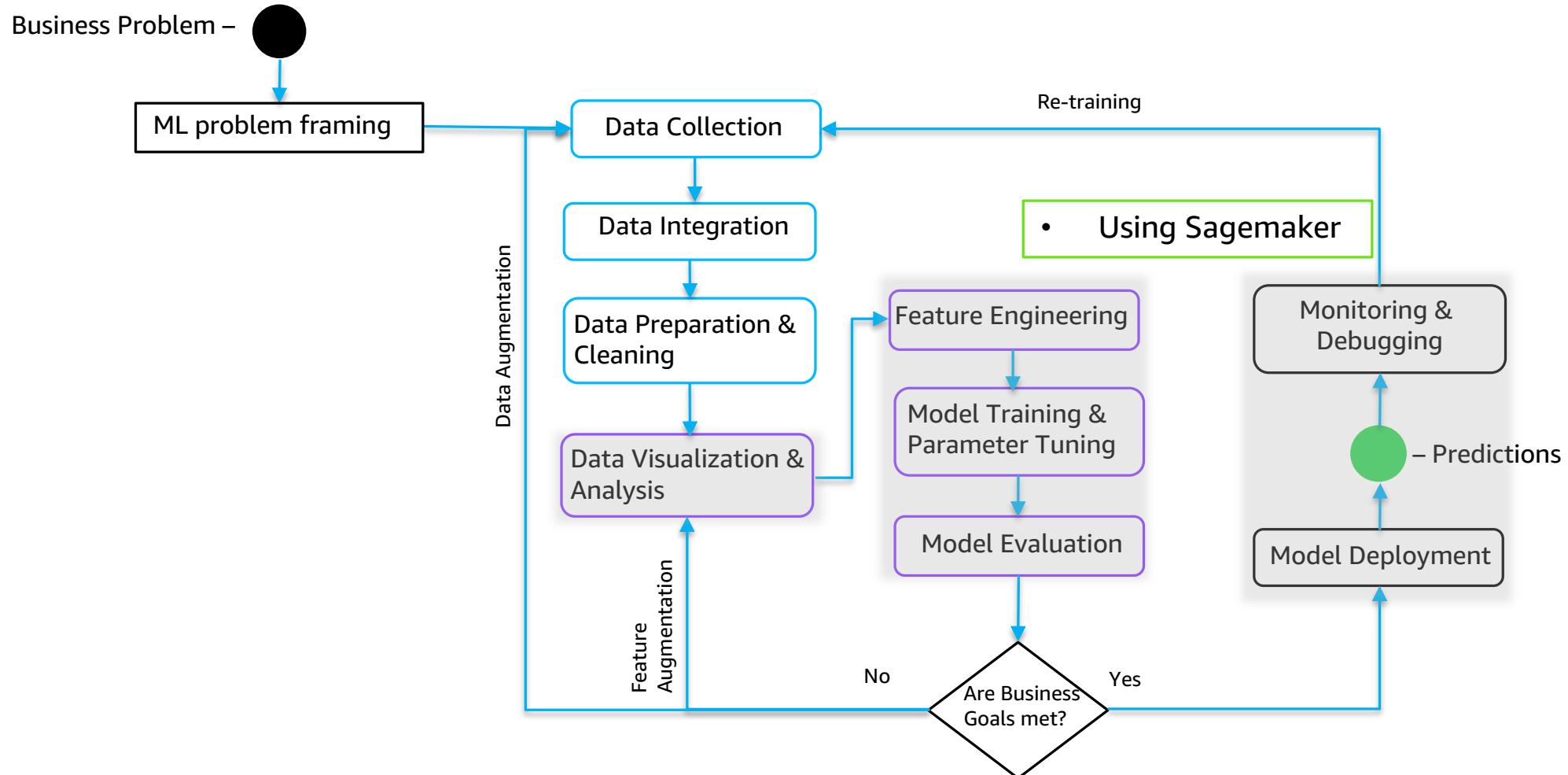
Machine Learning Process



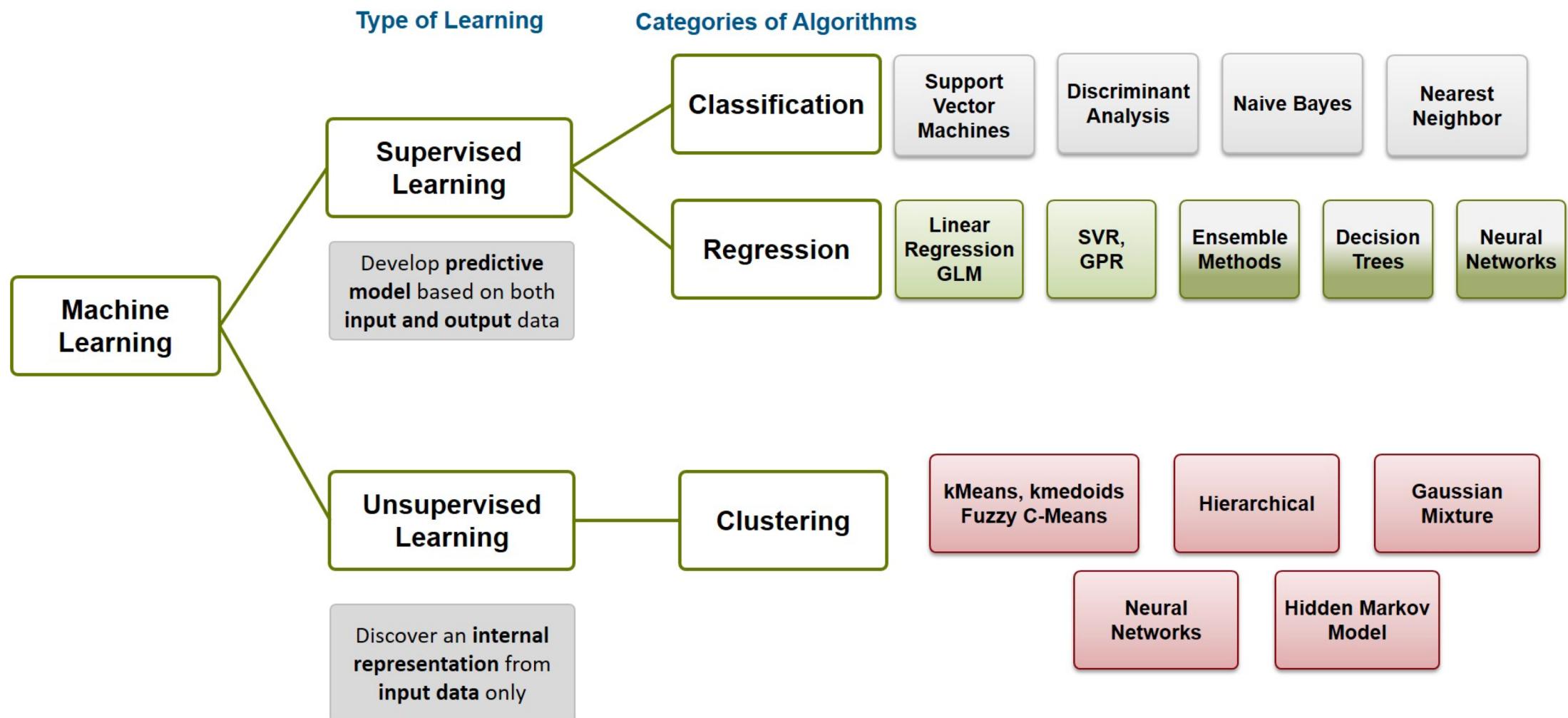
Machine Learning Process



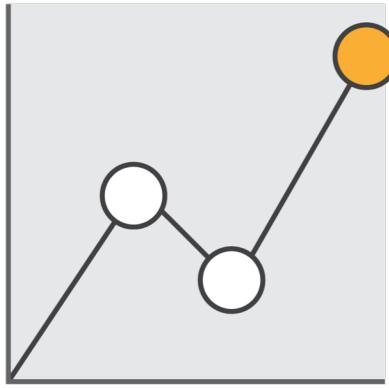
Machine Learning Process



Data Analysis Algorithms

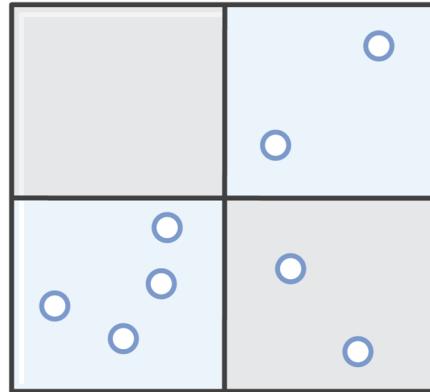


Amazon SageMaker Built-in ML Algorithm



Classification, Regression

- Linear Learner
- XGBoost
- Factorization Machines
- DeepAR



클러스터링, 차원축소

- K-Means
- PCA

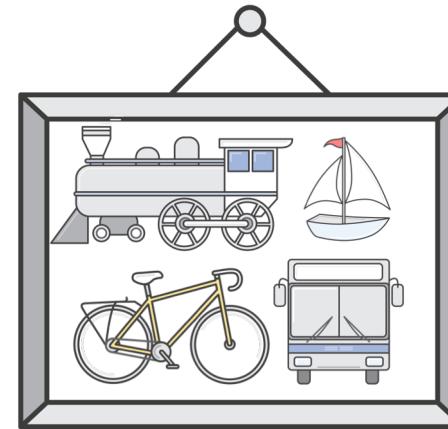
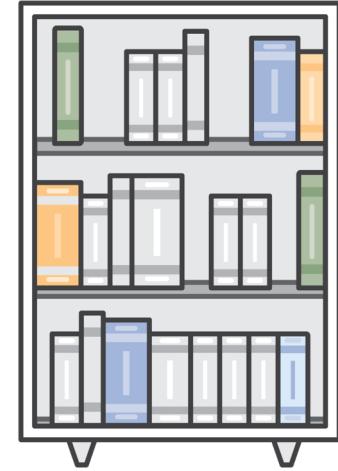


Image Classification

- Image Classifier (Resnet)



Natural language processing

- LDA, NTM (Topic modeling)
- Seq2Seq (Translation)
- BlazingText (Word2Vec)



NEW!

Amazon QuickSight—ML Insights (Preview)

Automated business insights powered by ML and natural language

The dashboard displays the following key metrics and insights:

- Daily Revenue:** Daily revenue decreased **-0.51%** (**-\$57,032.99**) on Nov 18, 2018, from \$11.9M to \$11.14M compared to the previous day and is **-1.78%** (**-\$202,111.70**) below goal of \$11.34M. We are **\$289,67K (0.34%) above** 30-day average revenue of \$8.35M. We're operating at a run rate of **\$4.06B**.
- YTD Revenue:** Year-to-date revenue increased by **61.95% (\$1.14B)** from \$1.65B to \$2.99B compared to the same period last year and is **-1.81% (\$55.03M)** below plan of \$3.05B. We are **98.19%** achievement of YTD goal and **84.61%** achievement for annual goal.
- Callouts By Product and Category anomalies:** Sorry, a problem occurred while detecting anomalies.
- Daily Revenue Forecast:** Daily revenue is predicted to reach **\$12.02M** by end of the year. We expect to exit the year with an annualized run rate of **\$4.39B**. Total revenue for 2018 is predicted to reach **\$3.47B, \$63.06M (-1.78%) below** annual target of \$3.54B.
- Callouts By Customers:** Daily revenue for MULTIDEL INC. on Nov 22, 2018 was **higher than expected** at \$60,433.95.
- Top / Bottom Movers by Product:** Top daily revenue increase by products are:
 - Electronics increased by **\$537.84 (7.61%)** from \$7,066.98 to \$7,604.82.
 - Clothing increased by **\$484.86 (0.06%)** from \$769,561.89 to \$770,046.75.
 - Industrial increased by **\$427.49 (0.04%)** from \$1,078,710.02 to \$1,079,137.51.
 - Home Services increased by **\$114.68 (0.21%)** from \$55,338.04 to \$55,452.72.
- Top / Bottom to Plan Variance by Product:** Top products **above** plan for today are:
 - Movies is **\$147,437.42 above goal.**
 - Financial Services is **\$98,111.46 above goal.**
 - Clothing is **\$42,220.37 above goal.**
 - Computers is **\$38,003.06 above goal.**
 - Outdoors is **\$27,404.67 above goal.**Top products **below** plan for today are:
 - Digital is **-\$236,951.70 below goal.**
- Revenue by Product Category:**

Product Ca...	Nov 18, 2018	Nov 17, 2018
Arts	\$4,988.40	\$4,988.18
Automotive	\$52,309.00	\$52,493.34
Baby Product	\$1,354,243.11	\$1,368,901.33
Beauty	\$8,114.71	\$8,116.06
Books	\$1,330,700.80	\$1,331,300.71
Business	\$38,736.02	\$41,916.59
Clothing	\$770,046.75	\$769,561.89

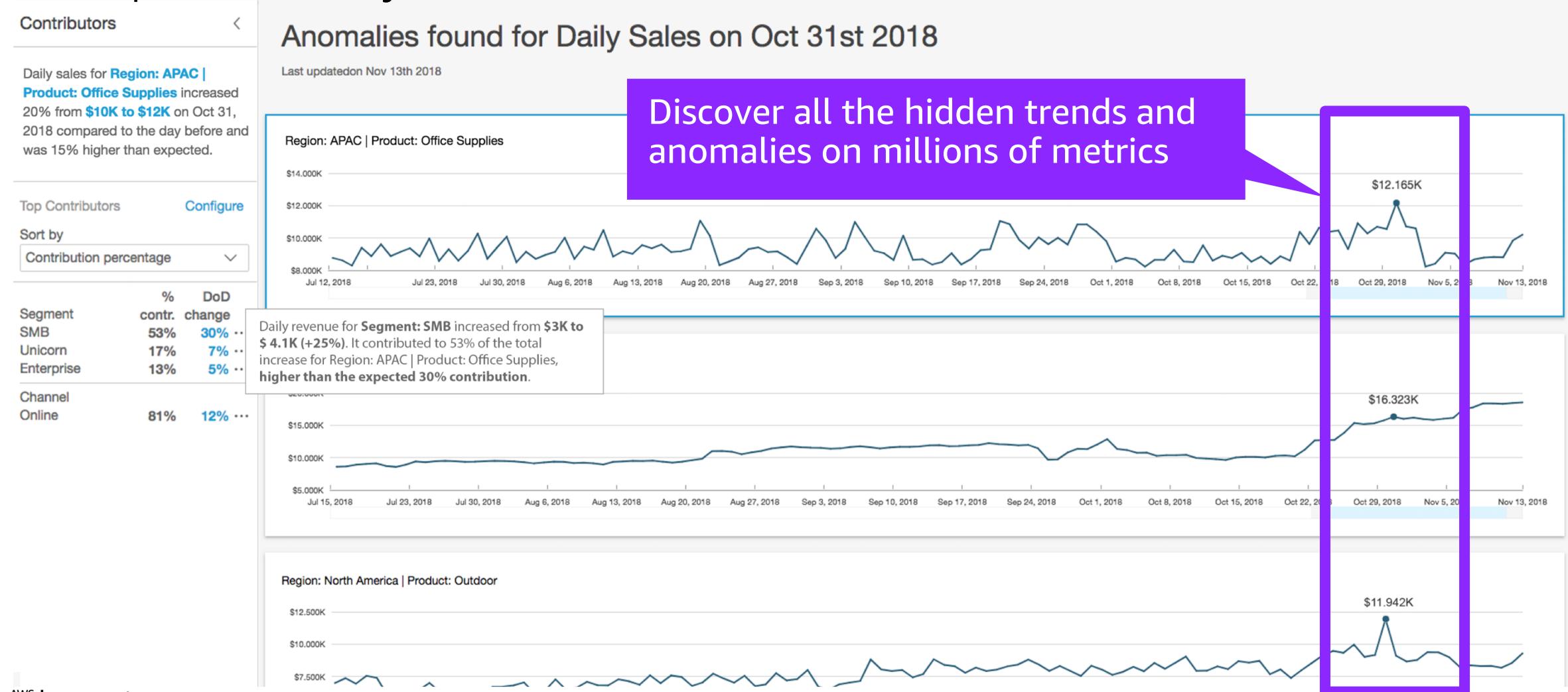
ML-powered anomaly detection

ML-powered forecasting

Auto-narratives

Amazon QuickSight—ML Insights

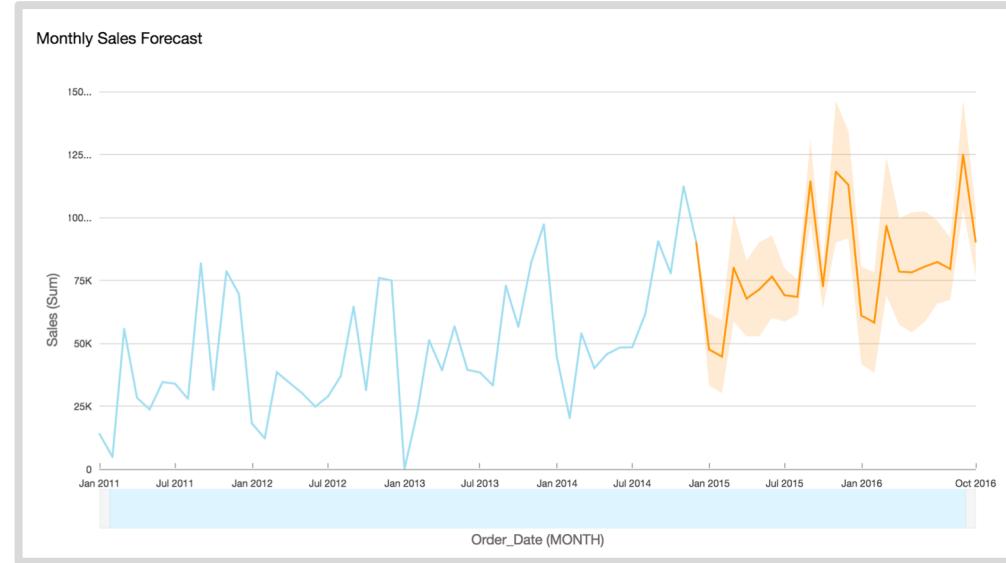
Example: anomaly detection



QuickSight ML Insights vs. traditional BI forecasting



QuickSight ML-powered forecasting



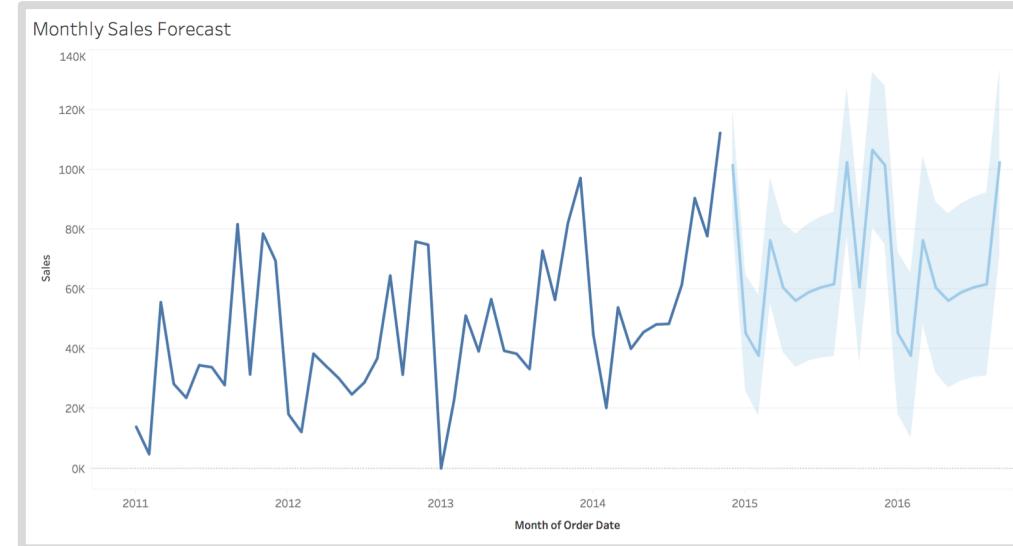
Captures seasonality and upward trends

Automatically excludes bad data

High confidence band

re:Invent

Traditional BI forecasting

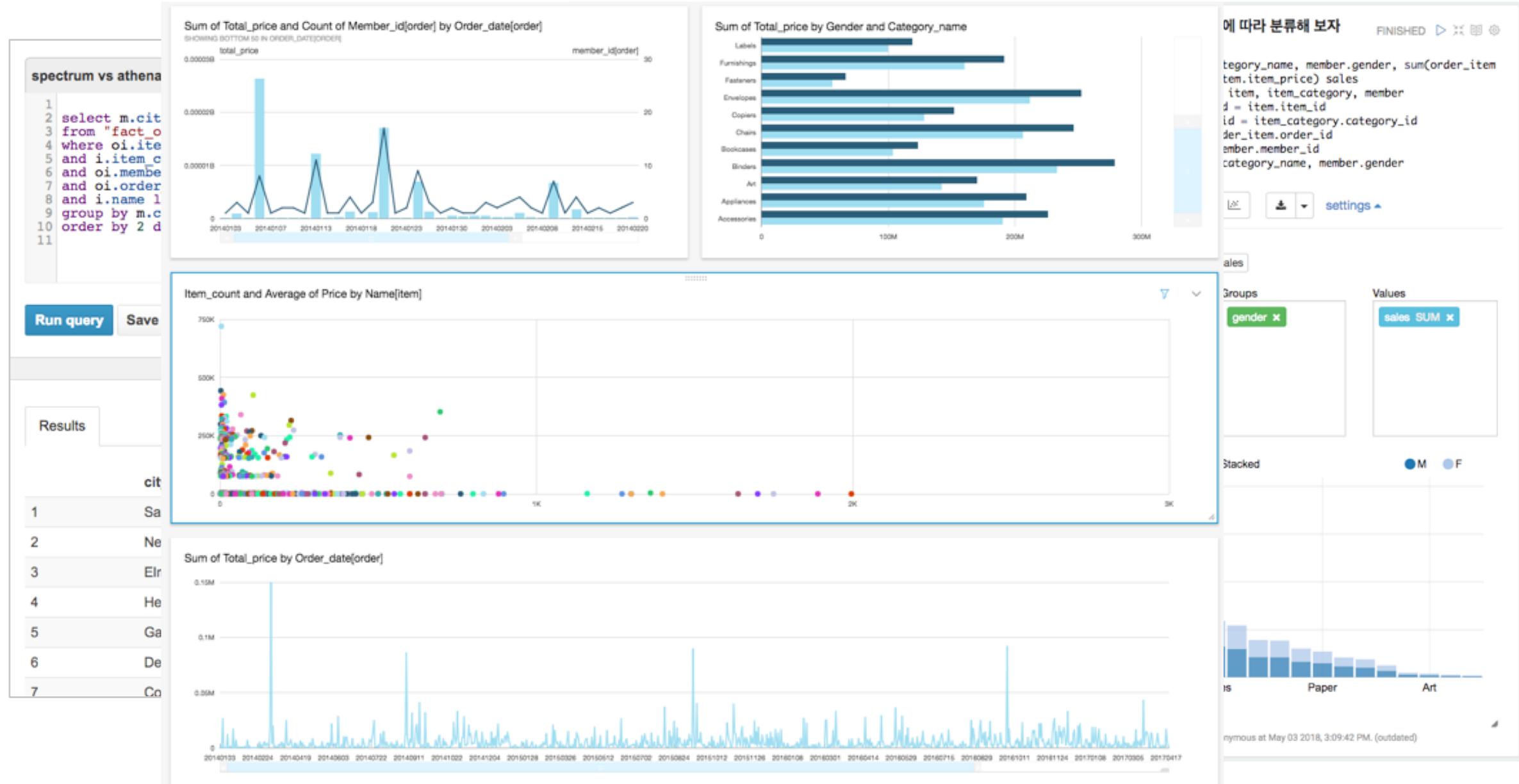


Captures only seasonality

Missing upward trend

Confidence band influenced by bad data

데이터 분석의 결과물 공유



데이터 분석의 결과물 공유

AWS Services Resource Groups Admin/animal-Isengard @ 874... Oregon Support

Amazon SageMaker

Dashboard Notebook instances Jobs

Resources Models Endpoint configuration Endpoints

Overview

The diagram shows a flow from data storage (represented by a database icon) through training (represented by neural network icons) to deployed models (represented by cubes) and finally to an endpoint (represented by a signal icon).

Notebook instance	Jobs	Models	Endpoint
Explore AWS data in your notebooks, and use algorithms to create models via training jobs.	Track training jobs at your desk or remotely. Leverage high-performance AWS algorithms.	View trained models or import external models into IM.	Deploy endpoints for developers to use in production. A/B Test model variants behind an endpoint.
Create notebook instance	View jobs	View models	View endpoints