Build a Data Lake with Amazon Web Services (AWS)

Damon Cortesi

https://dacort.dev

Home About Useful Links

Hi 👋 I'm

Damon Cortesi

A Big Data Architect

I work at AWS on EMR, Athena, Glue, and Lake Formation.
On any given day, I can be found

- Crafting presentations for both technical and executive audiences
- Debugging open source Hive, HBase, and Spark code
- Building proof of concepts and example architectures

aws

Intro to Data Lakes

There is more data than people think

Data	Data platforms **need to**	
grows	live for	scale
>10x	15	1,000x
every 5 years	years	

* IDC, Data Age 20215: The Evolution of Data to Life-Critical Don't Focus on Big Data, Focus on the Data That's Big, April 2017.

Data Scientists

Business Users

Analysts

Applications

Secure **Real time**

Flexible **Scalable**

There are more people accessing data

And more requirements for making data available

AWS databases and analytics

Broad and deep portfolio, built for builders

Business Intelligence & Machine Learning

- Amazon QuickSight
- Amazon SageMaker
- Amazon Comprehend
- Amazon Rekognition
- Amazon Lex
- Amazon Transcribe
- AWS DeepLens

AWS Marketplace 250+ solutions

Databases

- QLDB (NEW) — Ledger Database
- ElastiCache — Redis, Memcached
- Aurora — MySQL, PostgreSQL
- RDS — MySQL, PostgreSQL, MariaDB, Oracle, SQL Server
- Neptune — Graph
- DynamoDB — Key value, Document
- Timestream (NEW) — Time Series
- RDS on VMWare (NEW)

Analytics

- Amazon Redshift — Data warehousing
- Athena — Interactive analytics
- Amazon EMR — Hadoop + Spark
- Kinesis Analytics — Real-time
- Amazon Elasticsearch service — Operational Analytics

Blockchain

- Managed Blockchain (NEW)
- Blockchain Templates

730+ Database solutions

600+ Analytics solutions

25+ Blockchain solutions

Data Lake

- S3/Amazon Glacier
- Lake Formation (NEW) — Data Lakes
- AWS Glue — ETL & Data Catalog

20+ Data lake solutions

Data Movement

Database Migration Service | Snowball | Snowmobile | Kinesis Data Firehose | Kinesis Data Streams | Data Pipeline | Direct Connect

30+ solutions

A data lake is a centralized repository that allows you to store all your structured and unstructured data at any scale

Damon's data lake is a centralized repository of various quantified self and other personal stats including open Chrome Tabs, unread email count, GitHub repository stats and more

aws

AWS databases and analytics

Broad and deep portfolio, built for builders

Business Intelligence & Machine Learning

- Amazon QuickSight
- Amazon SageMaker
- Amazon Comprehend
- Amazon Rekognition
- Amazon Lex
- Amazon Transcribe
- AWS DeepLens

AWS Marketplace 250+ solutions

Databases

- QLDB (NEW) — Ledger Database
- ElastiCache — Redis, Memcached
- Aurora — MySQL, PostgreSQL
- RDS — MySQL, PostgreSQL, MariaDB, Oracle, SQL Server
- Neptune — Graph
- DynamoDB — Key value, Document
- Timestream (NEW) — Time Series
- RDS on VMWare (NEW)

Analytics

- Amazon Redshift — Data warehousing
- Athena — Interactive analytics
- Amazon EMR — Hadoop + Spark
- Kinesis Analytics — Real-time
- Amazon Elasticsearch service — Operational Analytics

Blockchain

- Managed Blockchain (NEW)
- Blockchain Templates

730+ Database solutions

600+ Analytics solutions

25+ Blockchain solutions

Data Lake

- S3/Amazon Glacier
- Lake Formation (NEW) — Data Lakes
- AWS Glue — ETL & Data Catalog

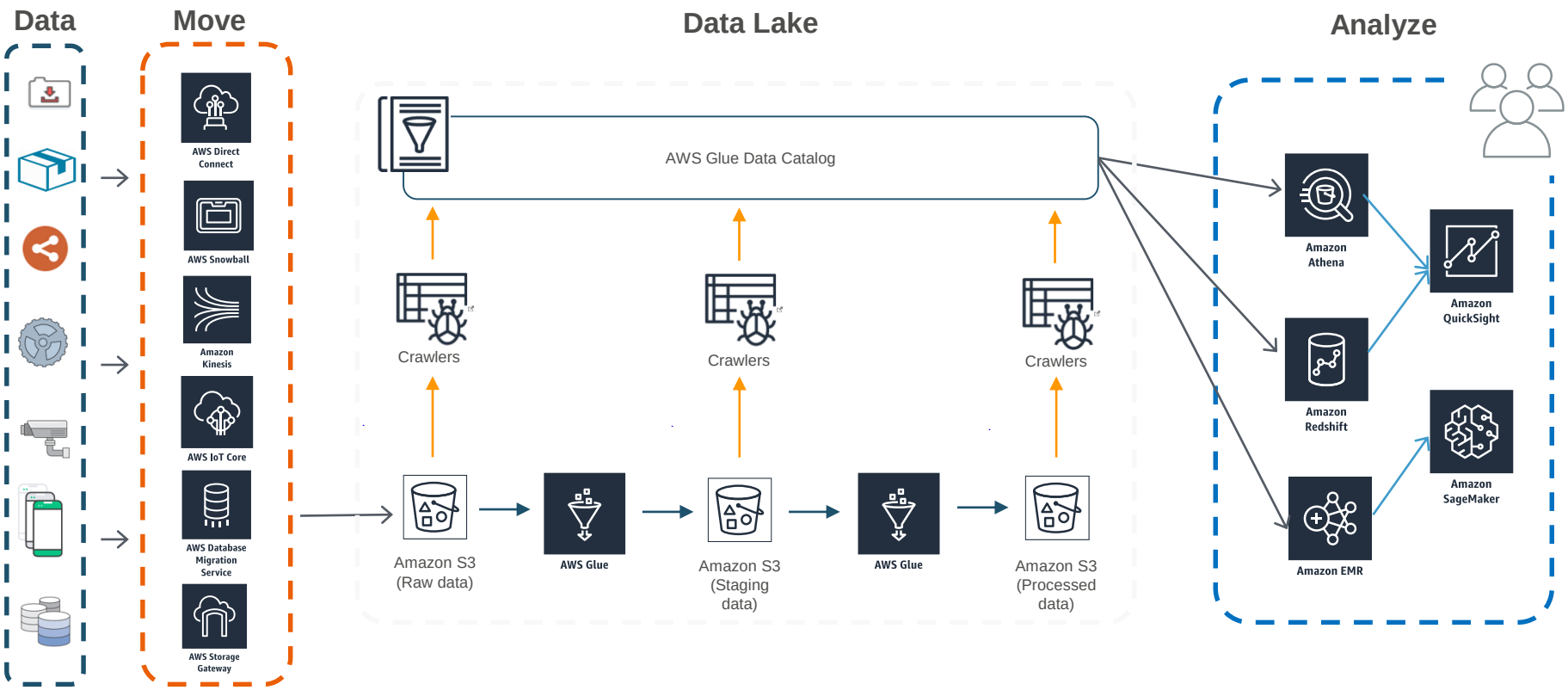
20+ Data lake solutions

Data Movement

Database Migration Service | Snowball | Snowmobile | Kinesis Data Firehose | Kinesis Data Streams | Data Pipeline | Direct Connect

30+ solutions

Data lake with AWS Glue



Data Ingestion

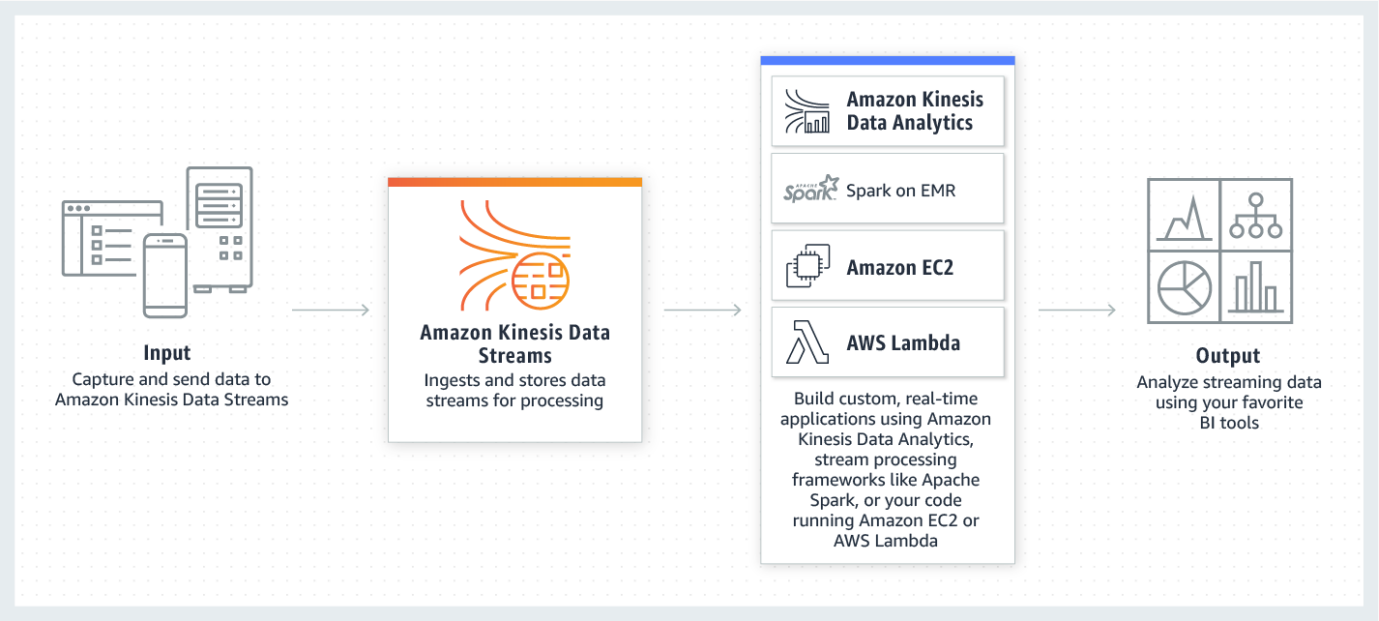
Amazon Kinesis Data Streams

Elastic

Real-time

Highly Scalable

Stream Analytics
with Kinesis Data Analytics



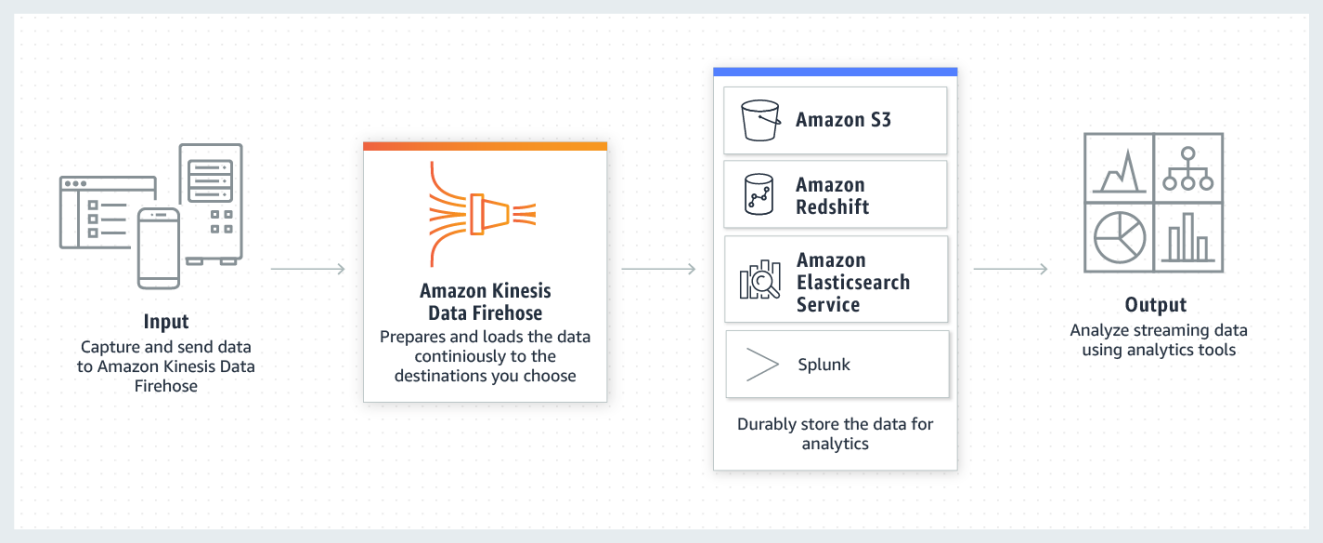
Amazon Kinesis Data Firehose

Serverless

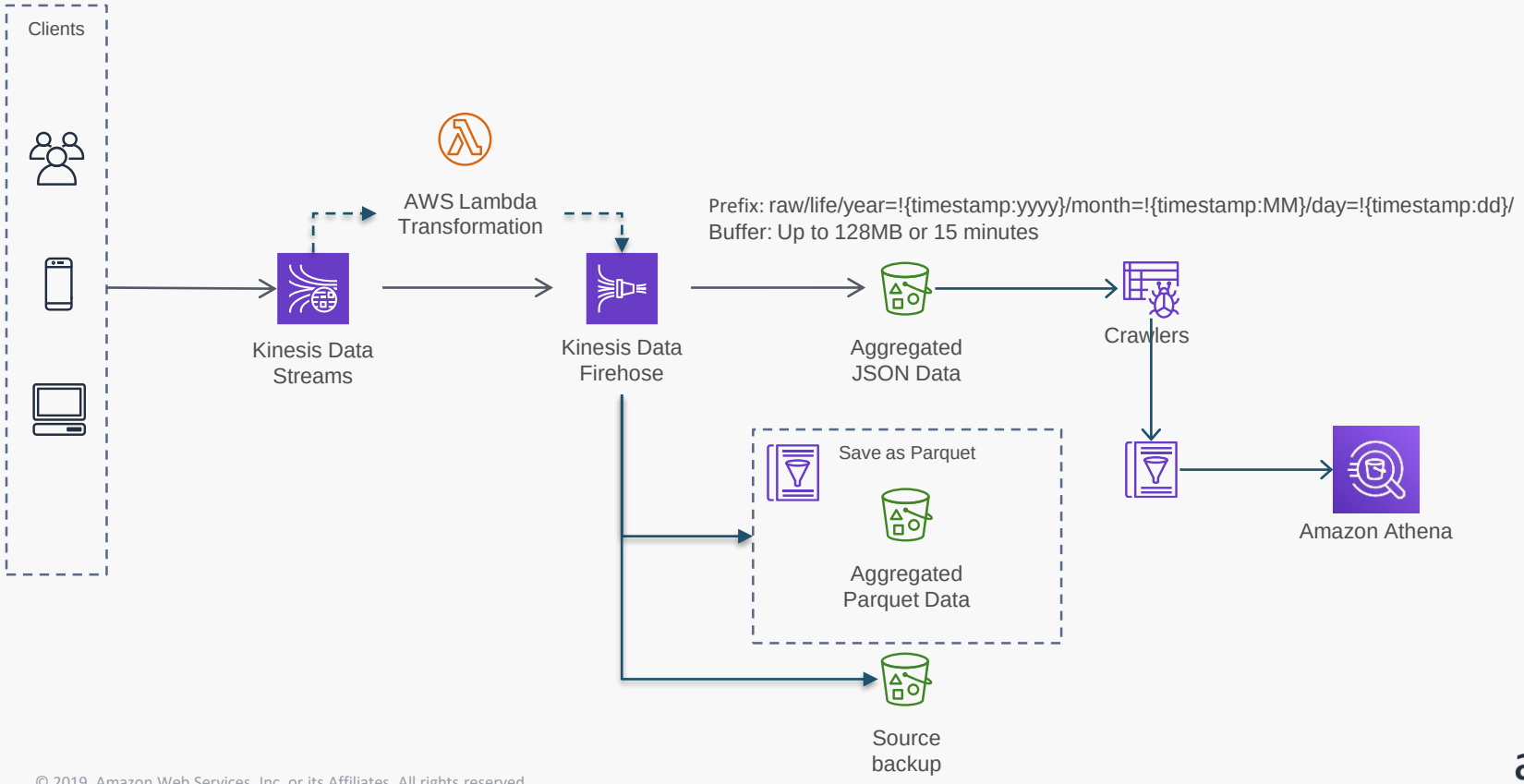
Data Transforms

Near real-time
loading to destination

Integrated



Kinesis Data Streams to Amazon S3



Productivity Publisher

```
unread_count() {
  osascript -e 'tell application "Microsoft Outlook"
                unread count of folder "Inbox" of default account
              end tell'
}

chrome_tabs() {
  osascript -e 'tell application "Google Chrome" to count every tab of every window'
}

iterm_tabs() {
  osascript -e 'tell application "iTerm" to count every tab of every window’
}

UNREAD_COUNT=$(unread_count)
CURRENT_TIME=$(date -u +"%Y-%m-%dT%H:%M:%SZ")
echo "Unread emails: ${UNREAD_COUNT}"
aws kinesis put-record --stream-name ${STREAM_NAME} --data '{
  "event":"outlook_unread", "type":"gauge",
  "value":"'${UNREAD_COUNT}'", "ts":"'${CURRENT_TIME}’”
}
' --partition-key 1
```

Kinesis Data Firehose to Amazon S3

Amazon S3 destination

S3 bucket	datalake-bucket
Prefix	raw/life/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/
Error prefix	kinesisErrors/life/year=!{timestamp:yyyy}/month=!{timestamp:MM}/day=!{timestamp:dd}/hour=!{timestamp:HH}/!{firehose:error-output-type}
Buffer conditions	128 MB or 900 seconds
Compression	GZIP
Encryption	Disabled

Raw Data in Amazon S3

```
❯ aws s3 ls s3://datalake-bucket/raw/life/year=2019/month=04/day=01/ --recursive | head

2019-04-01 06:28:56     369 hour=05/dcortesi-lifestream-1-2019-04-01-05-13-54-b03e43c6-a56d-4348-ad35-62d7b469f9e5.gz
2019-04-01 06:44:44     314 hour=05/dcortesi-lifestream-1-2019-04-01-05-29-42-acc96203-08ef-4a8d-8549-6ffaf9c8321a.gz
2019-04-01 07:00:33     311 hour=05/dcortesi-lifestream-1-2019-04-01-05-45-29-cb63ffad-cf1c-4157-bb41-006e9d31bb6c.gz
2019-04-01 07:16:14     328 hour=06/dcortesi-lifestream-1-2019-04-01-06-01-11-607e3fff-011a-45a5-bba0-97498e9383a7.gz
2019-04-01 07:31:53     349 hour=06/dcortesi-lifestream-1-2019-04-01-06-16-50-e71c0155-8c3d-4c5b-8724-e1d7eb82b93a.gz
2019-04-01 07:47:36     303 hour=06/dcortesi-lifestream-1-2019-04-01-06-32-33-3a28b061-26c0-42da-a789-463b33b7098a.gz
2019-04-01 08:03:22     307 hour=06/dcortesi-lifestream-1-2019-04-01-06-48-18-8f9c8f63-adf9-4ecb-b243-10e47c238c78.gz
2019-04-01 08:19:01     145 hour=07/dcortesi-lifestream-1-2019-04-01-07-03-59-c548c4c4-d233-4900-a7c6-236de3ba595d.gz
2019-04-01 16:07:15     345 hour=14/dcortesi-lifestream-1-2019-04-01-14-52-14-06a9b781-1ca4-448c-9ea8-f2f077a42265.gz
2019-04-01 16:23:11     344 hour=15/dcortesi-lifestream-1-2019-04-01-15-08-05-8be669f4-94ee-4633-b6bf-4239ec6ce6a4.gz
```

AWS Glue ETL

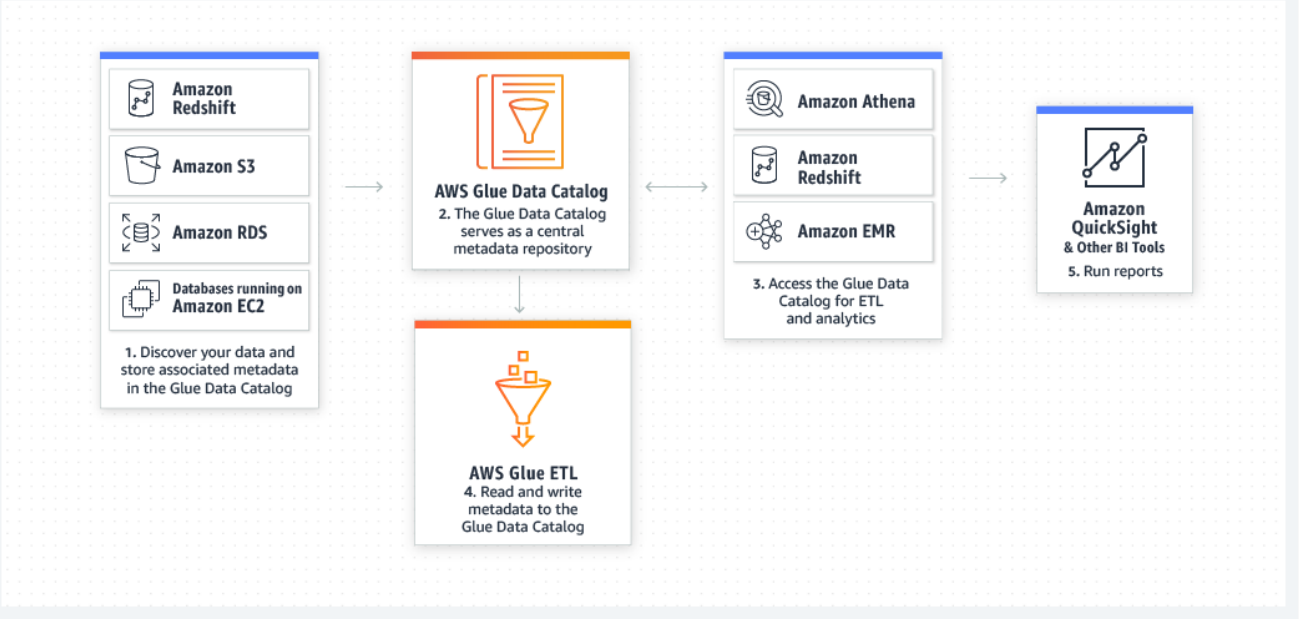
Serverless

Batch operations

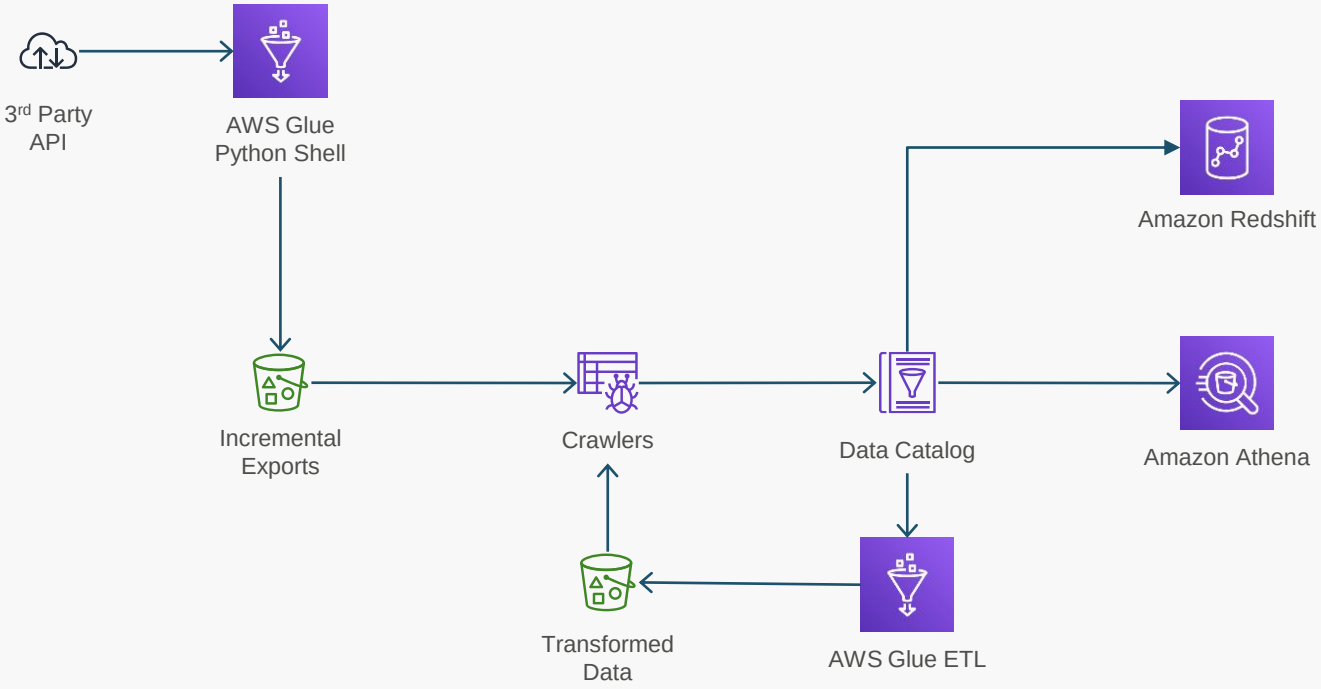
Many sources

Apache Spark
Python and Scala

Python Shell



Third-party API to Amazon S3

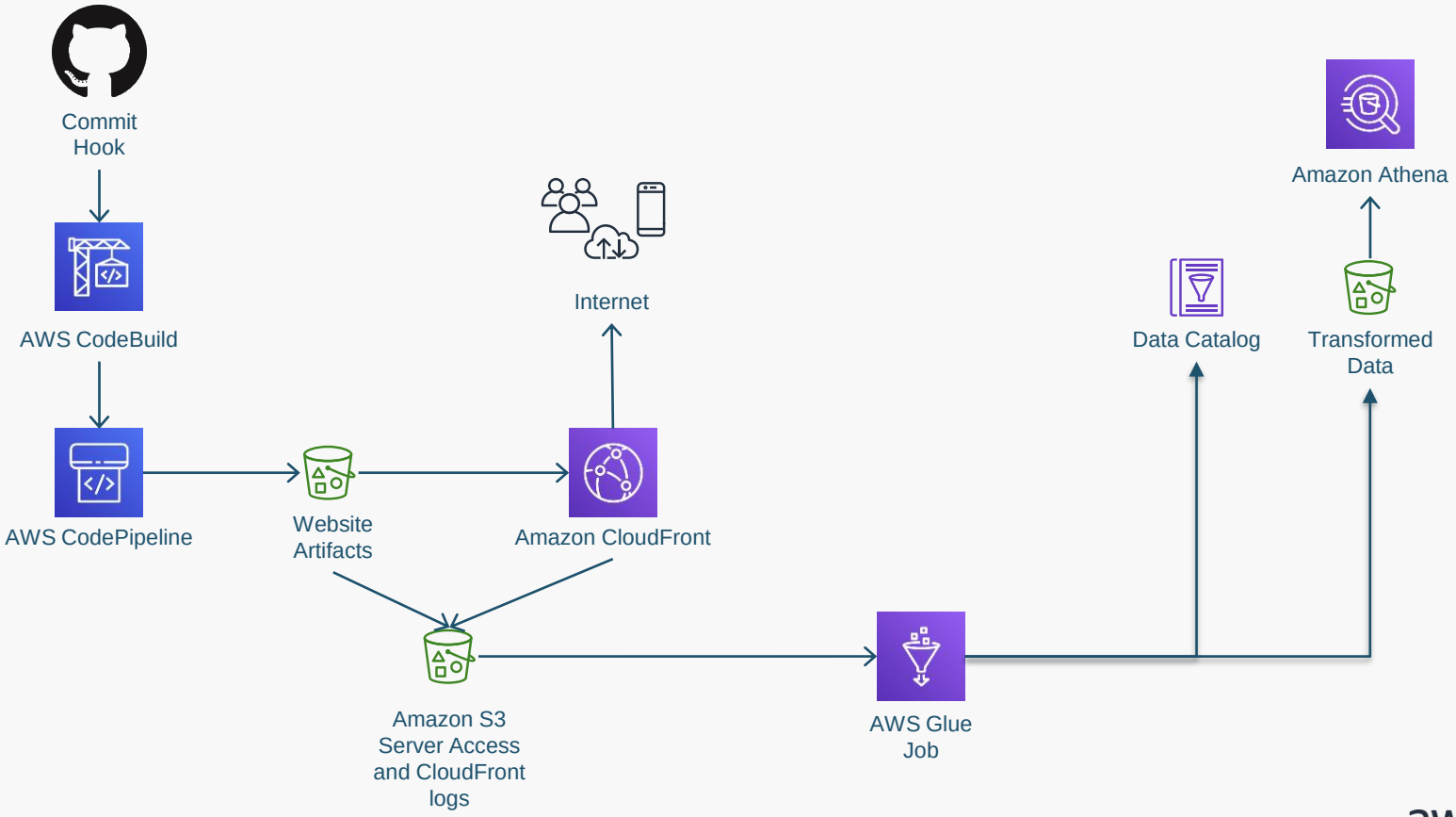


GitHub Traffic Stats

```
github_repos = ['dacort/athena-query-stats', 'dacort/damons-data-lake']
traffic_endpoints = ['popular/referrers', 'popular/paths', 'views', 'clones']

for repo in github_repos:
  for endpoint in traffic_endpoints:
    url = 'https://api.github.com/repos/' + repo + '/traffic/' + endpoint
    headers = {'Authorization': 'token ' + get_secret()}  # Use AWS Secrets Manager
    r = requests.get(url, headers=headers)
    if(r.ok):
      events = json.loads(r.text or r.content)
      # Only write out the file if we got any useful data
      if events:
        today = datetime.datetime.now().strftime('%Y-%m-%d')
        s3_key = '%s/%s/%s/%s.json' % (
          get_job_arg('prefix'),
          endpoint.replace('popular/', 'traffic/'),
          repo, today
        )
        save_results(get_job_arg('bucket'), s3_key, events)
```

Website Logs to Amazon S3



Amazon CloudFront Logs

```
#Version: 1.0
#Fields: date time x-edge-location sc-bytes c-ip cs-method cs(Host) cs-uri-stem sc-status cs(Referer) cs(User-Agent) cs-uri-query cs(Cookie) x-edge-result-type x-edge-request-id x-host-header cs-protocol cs-bytes time-taken x-forwarded-for ssl-protocol ssl-cipher x-edge-response-result-type cs-protocol-version fle-status fle-encrypted-fields

2019-04-18	09:13:43	LAX3-C3	3439	10.0.0.1	GET	d2tesbao5njk9n.cloudfront.net	/	200	-	Mozilla/5.0%2520(Macintosh;%2520Intel%2520Mac%2520OS%2520X%252010_12_6)%2520AppleWebKit/537.36%2520(KHTML,%2520like%2520Gecko)%2520Chrome/73.0.3683.103%2520Safari/537.36	-	-	Miss	U6zGSZXHb6peSeVwpRPJ8o0-TUGAy9bu6cA_c0obQt_pne7A4Q_25Q==	dacort.dev	https	255	0.169	-	TLSv1.2	ECDHE-RSA-AES128-GCM-SHA256	Miss	HTTP/2.0	-	-

2019-04-18	09:13:43	LAX3-C3	91094	10.0.0.1	GET	d2tesbao5njk9n.cloudfront.net	/assets/images/ab-img.png	200	https://dacort.dev/	Mozilla/5.0%2520(Macintosh;%2520Intel%2520Mac%2520OS%2520X%252010_12_6)%2520AppleWebKit/537.36%2520(KHTML,%2520like%2520Gecko)%2520Chrome/73.0.3683.103%2520Safari/537.36	-	-	Miss	TUmpSnnw4yCP77em-u6Hds9hHaYftx_JOSo8WkI59qwNY-NVHShXag==	dacort.dev	https	42	0.123	-	TLSv1.2	ECDHE-RSA-AES128-GCM-SHA256	Miss	HTTP/2.0	-	-
```

bucket-name.s3.amazonaws.com/*optional-prefix*/*distribution-ID.YYYY-MM-DD-HH.unique-ID*.gz

Amazon S3 Server Access Logs

79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be website-bucket [18/Apr/2019:09:13:43 +0000] 10.0.0.1 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F33A59F07 REST.GET.OBJECT assets/assets/css/responsive.css "GET /assets/assets/css/responsive.css HTTP/1.1" 200 - 16167 16167 35 35 "-" "Amazon CloudFront" - 3HL4kqtJvjVBH40Nrjfkd s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader website-bucket.s3.amazonaws.com TLSv1.2

79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be website-bucket [18/Apr/2019:09:13:44 +0000] 10.0.0.1 79a59df900b949e55d96a1e698fbacedfd6e09d98eacf8f8d5218e7cd47ef2be 3E57427F33A59F08 REST.GET.OBJECT assets/assets/icons/font-awesome-4.7.0/fonts/fontawesome-webfont.woff2 "GET /assets/assets/icons/font-awesome-4.7.0/fonts/fontawesome-webfont.woff2 HTTP/1.1" 200 - 77160 77160 71 71 "-" "Amazon CloudFront" - 3HL4kqtJvjVBH40Nrjfkd s9lzHYrFp76ZVxRcpX9+5cjAnEH2ROuNkd2BHfIa6UkFVdtjf5mKR3/eTPFvsiP/XV/VLi31234= SigV4 ECDHE-RSA-AES128-GCM-SHA256 AuthHeader website-bucket.s3.amazonaws.com TLSv1.2

bucket-name.s3.amazonaws.com/*optional-prefix*/*YYYY-mm-DD-HH-MM-SS-UniqueString*

Fairly common problem

- Unstructured
- Multiple log types
- Different format
- Different compression
- One solution – embed parsing/conversion code in common library
  github.com/awslabs/athena-glue-service-logs

```
from athena_glue_service_logs.job import JobRunner

job_run = JobRunner(service_name='s3_access’)
job_run.convert_and_partition()
```

Configure with Job Parameters

```
{
  "Job": {
    "Command": {
      "Name": "glueetl",
      "ScriptLocation": "s3://bucket/scripts/s3_access_job.py"
    },
    "DefaultArguments": {
      "--extra-py-files": "s3://bucket/athena_glue_converter_latest.zip",

      "--s3_source_location": "s3://bucket/service_logs/s3_access/",
      "--raw_database_name": "awslogs_raw",
      "--raw_table_name": "s3_access",

      "--s3_converted_target": "s3://bucket/converted/s3_access",
      "--converted_database_name": "awslogs",
      "--converted_table_name": "s3_access",

      "--job-language": "python",
      "--job-bookmark-option": "job-bookmark-enable",
      "--TempDir": "s3://bucket/tmp”
    }
  }
}
```

Source Data

Target Data

Run every hour

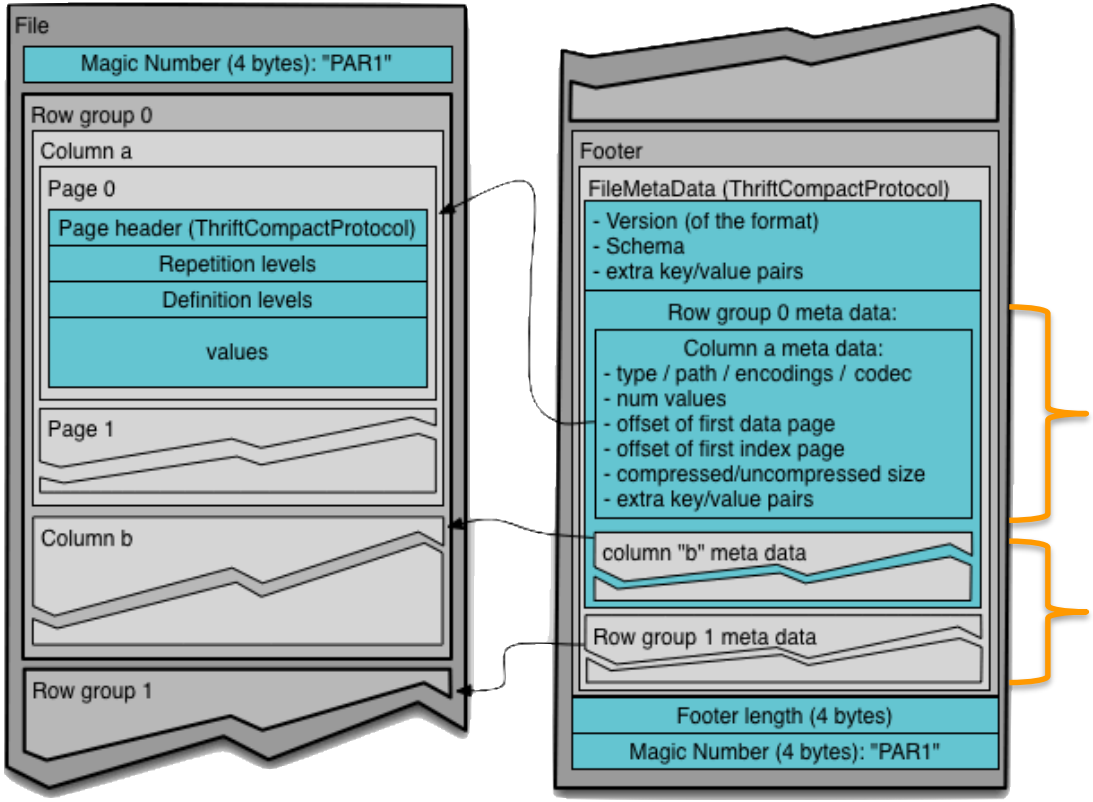
Trigger properties

Name	dcortesi-log-converter
Tags	-
Trigger type	Schedule
Schedule	At 05 minutes past the hour

Jobs to start

Jobs	dcortesi_dl_CloudFront_LogMaster_v5.3.3_dev, dcortesi_dl_S3Access_LogMaster_v5.3.4_dev

Parquet File Format



Columnar format is optimized for analytics.

Row group meta data allows Parquet reader to skip portions of, or all files.

Column meta-data allows for pre-aggregation

Results

Conversion	File format	Data Scanned	Run time
SELECT count(*) FROM s3_access_raw	text	7.1 GB	81 seconds
SELECT count(*) FROM s3_access_optimized	snappy parquet	0 KB	3.7 seconds
Speedup			22x faster

Conversion	File format	Data Scanned	Run time
SELECT count(DISTINCT key) FROM s3_access_raw	text	7.1 GB	78 seconds
SELECT count(DISTINCT key) FROM s3_access_optimized	snappy parquet	91 MB	4.9 seconds
Speedup			16x faster

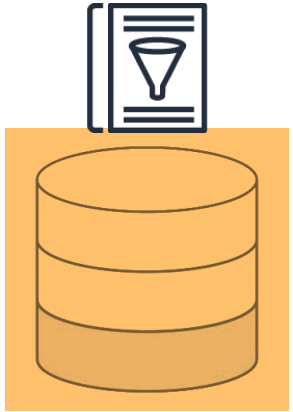
Data Discovery

So far we have…

- JSON traffic stats (uncompressed)
- JSON event streams (compressed)
- Tab-delimited service logs
- Regular Expression-based service logs
- Huge XML files
- Delivered via shell scripts, JavaScript, and email

- …not much different than the real world ☺

AWS Glue Data Catalog



Search and explore available data

Unified metadata repository across relational databases, Amazon RDS, Amazon Redshift, and Amazon S3.

Single searchable view into your data, no matter where it is stored

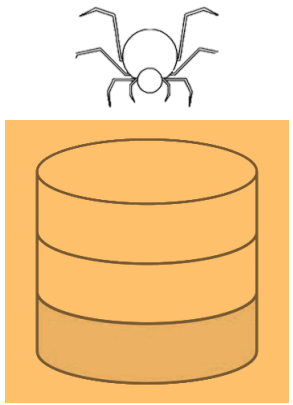
Ability to automatically crawl and **classify your data**

Augment technical metadata with **business metadata** for tables

Manage access to data using Fine Grain Access Controls. Even finer with AWS Lake Formation

Apache Hive metastore compatible and integrated with AWS Analytics services

AWS Glue Crawlers



Crawlers
Automatically catalog your data

Crawlers **automatically build** your Data Catalog and keep it in sync.

Automatically **discover** new data, extracts schema definitions

- Detect schema changes and version tables
- Detect Hive style partitions on Amazon S3

Built-in classifiers for popular types; custom classifiers using **Grok expression**

Run ad hoc or on a schedule; **serverless** – only pay when crawler runs

Crawler Definition

```
{
  "Crawler": {
    "Name": "damons_data_lake",

    "Targets": {
      "S3Targets": [
        { "Path": "s3://bucket/raw/life/", "Exclusions": [] },
        { "Path": "s3://bucket/raw/github/clones/", "Exclusions": [] },
        { "Path": "s3://bucket/raw/github/traffic/", "Exclusions": [] },
        { "Path": "s3://bucket/raw/github/views/", "Exclusions": [] }
      ]
    },
    "DatabaseName": "dcortesi",
    "TablePrefix": "dl_",
    "Schedule": {
      "ScheduleExpression": "cron(20 0/1 * * ? *)",
      "State": "SCHEDULED"
    },
    "Configuration": "{\"Grouping\":{\"TableGroupingPolicy\":\"CombineCompatibleSchemas\"}}"
  }
}
```

AWS Lake Formation

Build a secure data lake in days

Identify, ingest, clean, and transform data

Enforce security policies across multiple services

Gain and manage new insights

Analyze!

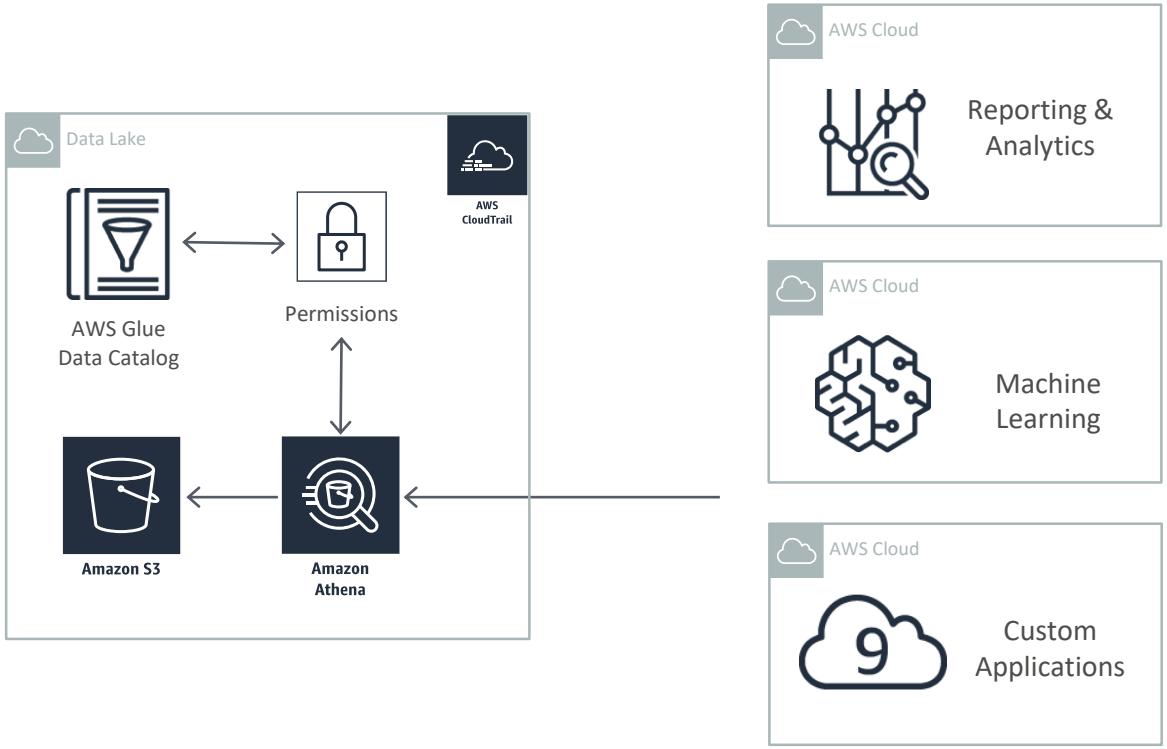
Amazon Athena

Serverless

Interactive Performance

Open.Powerful.Standard
Built on Apache Presto

Pay per query
$0.005 per GB scanned



Easily Query in Amazon Athena

New query 1

```
1 SELECT * FROM dl_life ORDER BY ts DESC LIMIT 10
```

Run query | Save as | Create | (Run time: 4.72 seconds, Data scanned: 174.4 KB)

Use Ctrl + Enter to run query, Ctrl + Space to autocomplete

Database

dcortesi

dl_life

Tables (1) Create table

dl_life (Partitioned)

- event (string)
- type (string)
- value (string)
- ts (string)
- metadata (struct<clipboard:struct<source:string,de

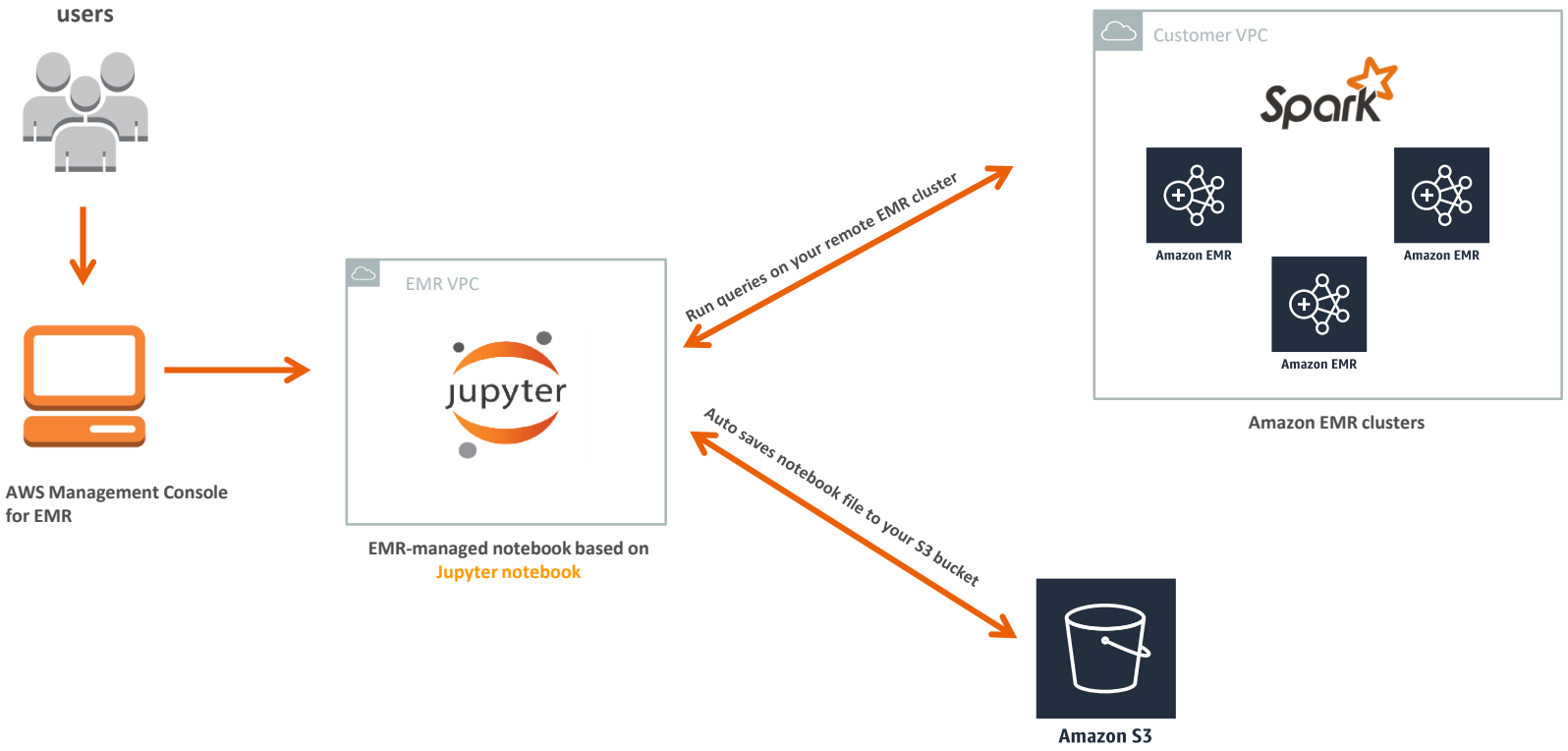
Results

	event	type	value	ts	metadata
1	iterm_tabs	gauge	6	2019-04-18T12:09:49Z	
2	outlook_unread	gauge	18	2019-04-18T12:09:49Z	
3	chrome_tabs	gauge	22	2019-04-18T12:09:49Z	
4	outlook_unread	gauge	18	2019-04-18T12:08:39Z	
5	chrome_tabs	gauge	22	2019-04-18T12:08:39Z	
6	iterm_tabs	gauge	6	2019-04-18T12:08:39Z	
7	chrome_tabs	gauge	22	2019-04-18T12:07:31Z	
8	iterm_tabs	gauge	6	2019-04-18T12:07:31Z	
9	outlook_unread	gauge	18	2019-04-18T12:07:31Z	
10	clipboard_copy	counter	1	2019-04-18T12:06:32Z	{clipboard={source=Google Chrome, destination=Google Chrome}}

aws

Amazon EMR Notebooks in the Console

A managed analytics environment based on Jupyter Notebooks



Apple Health Data

jupyter dcortesi-health Last Checkpoint: an hour ago (autosaved)

File Edit View Insert Cell Kernel Widgets Help

Data Exploration

In [3]:
```
from pyspark.sql import SparkSession
spark = SparkSession.builder.getOrCreate()

df = spark.read.format('xml').options(rowTag='Record').load('s3
```

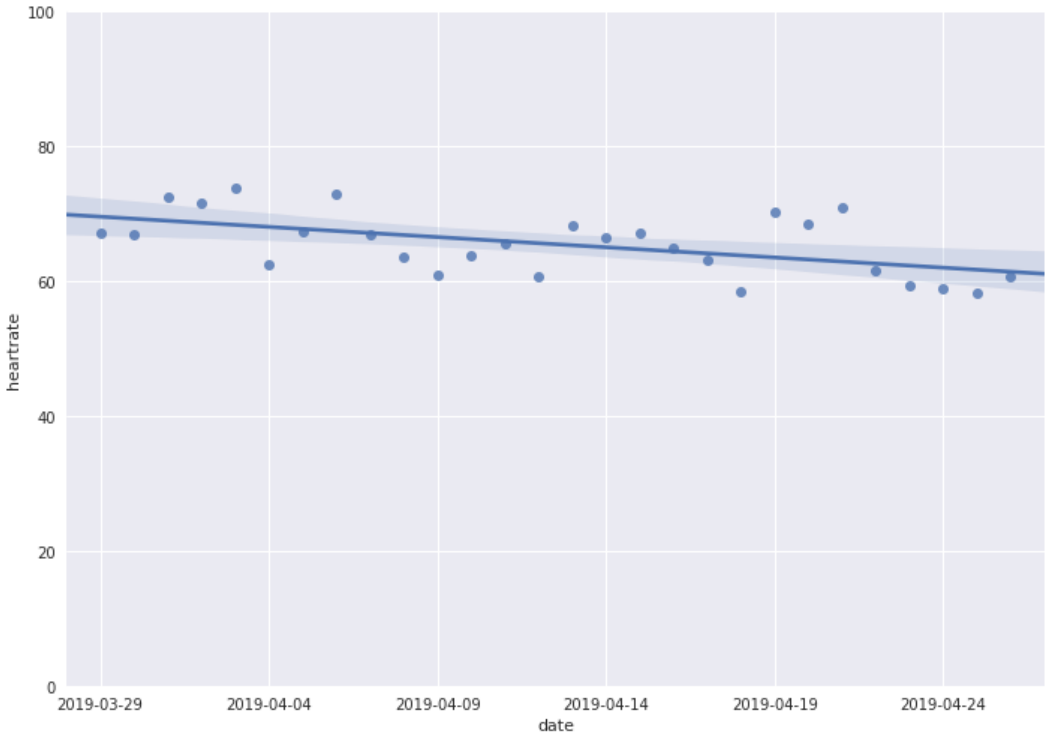
Data types for Motiv

In [4]:
```
df.where("_sourceName = 'Motiv'").groupBy("_type").count().show
```

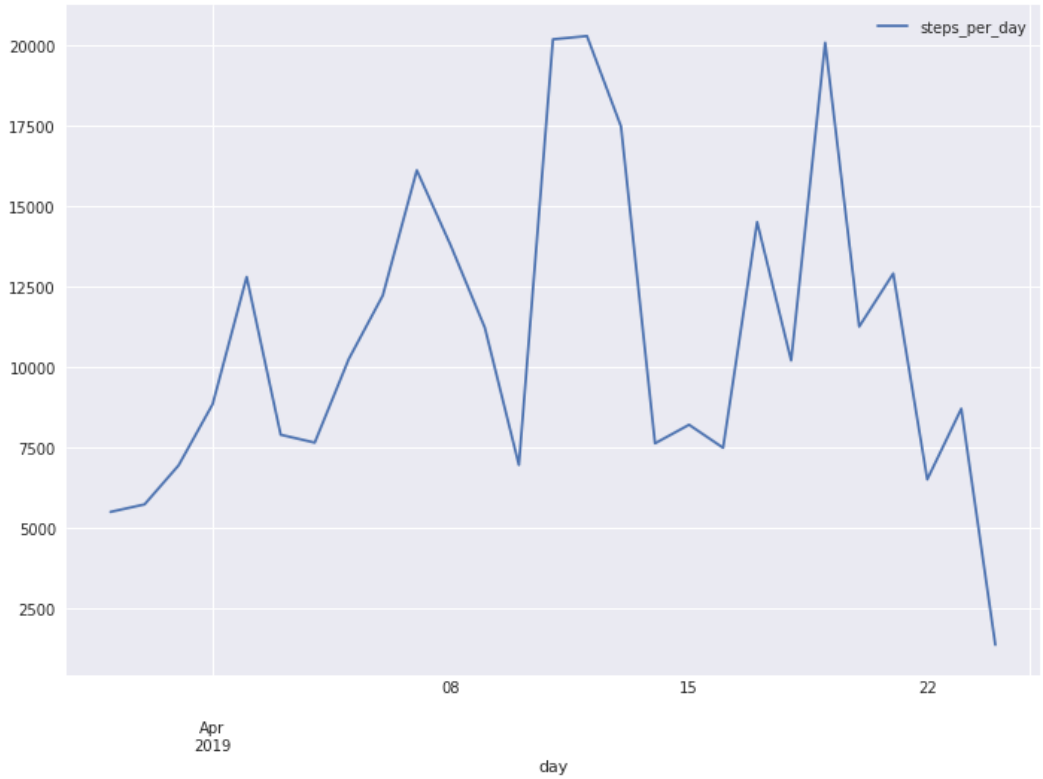
▸ Spark Job Progress

```
+-----------------------------------+-----+
|_type                              |count|
+-----------------------------------+-----+
|HKQuantityTypeIdentifierStepCount  |234  |
|HKQuantityTypeIdentifierHeartRate  |18304|
|HKCategoryTypeIdentifierSleepAnalysis|100  |
+-----------------------------------+-----+
```

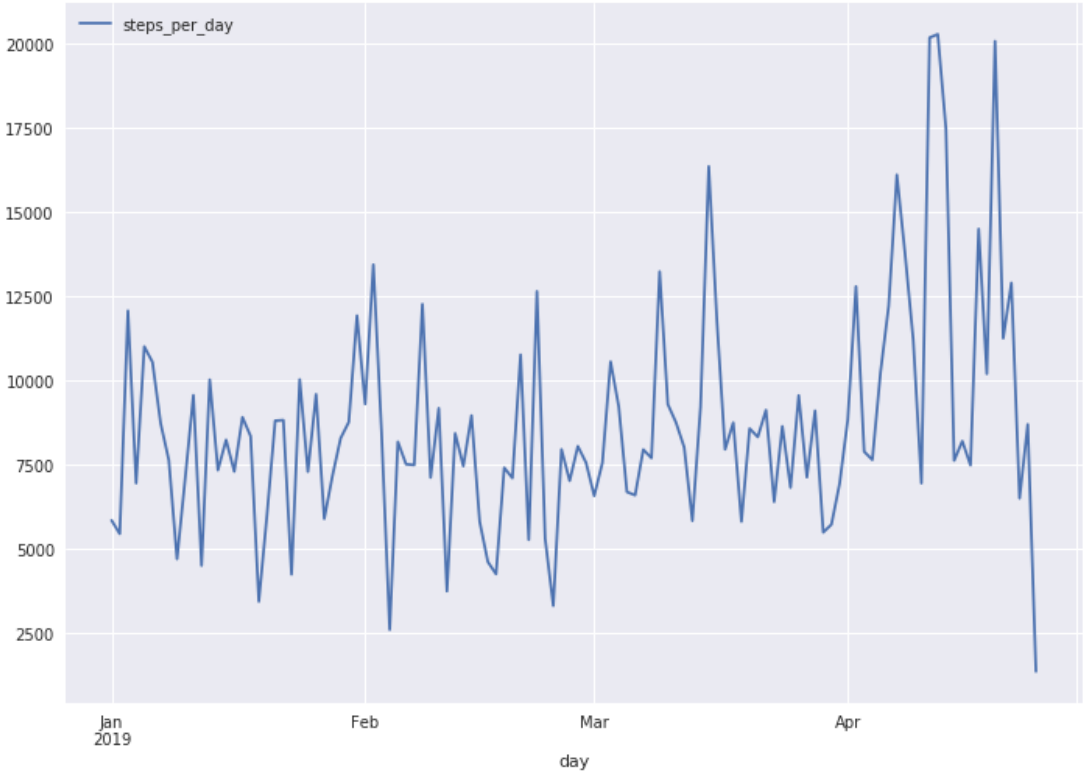
Am I getting healthier?



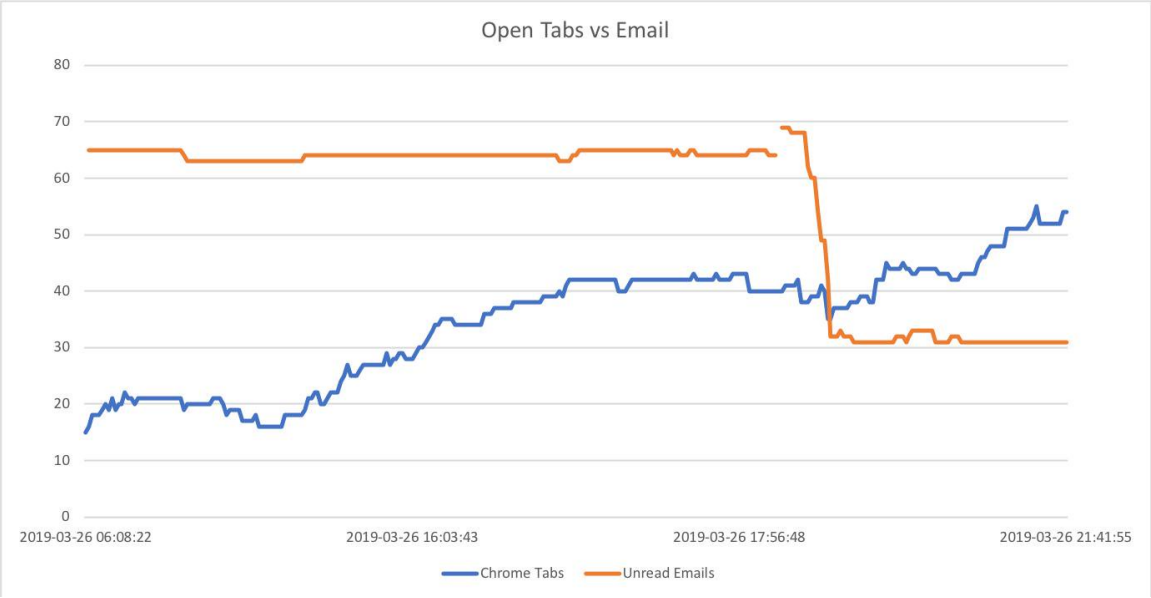
Step Count (Apr 2019)



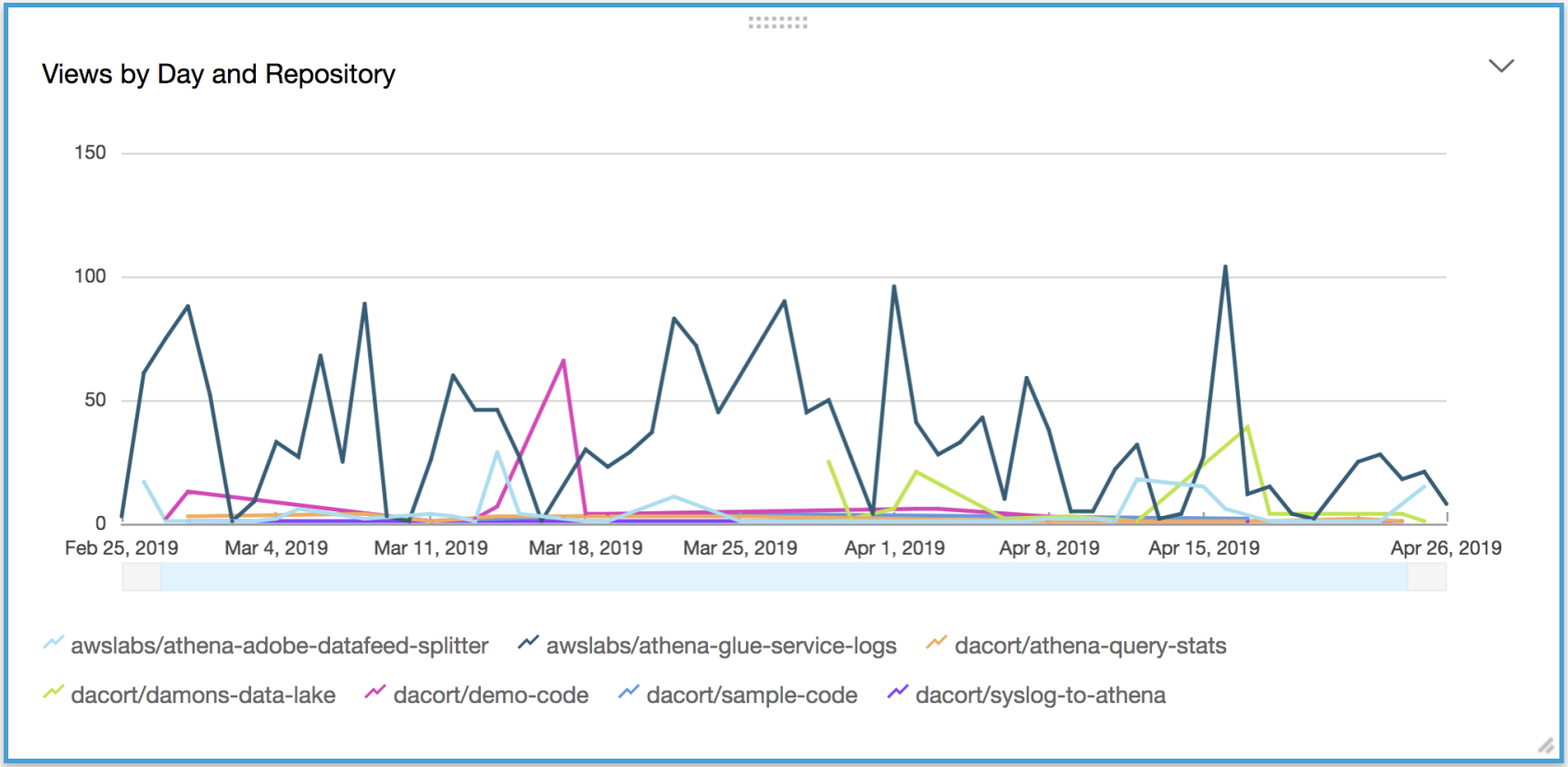
Step Count (Jan – Apr 2019)



Am I terrible at email?



Which projects should I focus on?



What apps are responsible for my RSI?

(Repetitive strain injury)

New query 1

```
SELECT
  metadata.clipboard.source,
  metadata.clipboard.destination,
  COUNT(*) AS copy_count
FROM dl_life
WHERE event = 'clipboard_copy'
GROUP BY 1,2
ORDER BY 3 DESC
LIMIT 10
```

Results

	source	destination	
1	Google Chrome	Google Chrome	562
2	iTerm2	iTerm2	148
3	Google Chrome	iTerm2	107
4	Code	Google Chrome	66
5	Code	Code	62
6	Google Chrome	Code	56
7	Code	iTerm2	41
8	Microsoft PowerPoint	Microsoft PowerPoint	37
9	iTerm2	Code	33
10	iTerm2	Google Chrome	32

What interesting user agents are visiting my site?

New query 1

```
SELECT count(*) as cnt, url_decode(url_decode(useragent))
FROM "dcortesi"."dl_cloudfront_raw"
WHERE uri = '/' AND useragent NOT LIKE '%Mozilla%'
group by 2 order by 1 desc
limit 10;
```

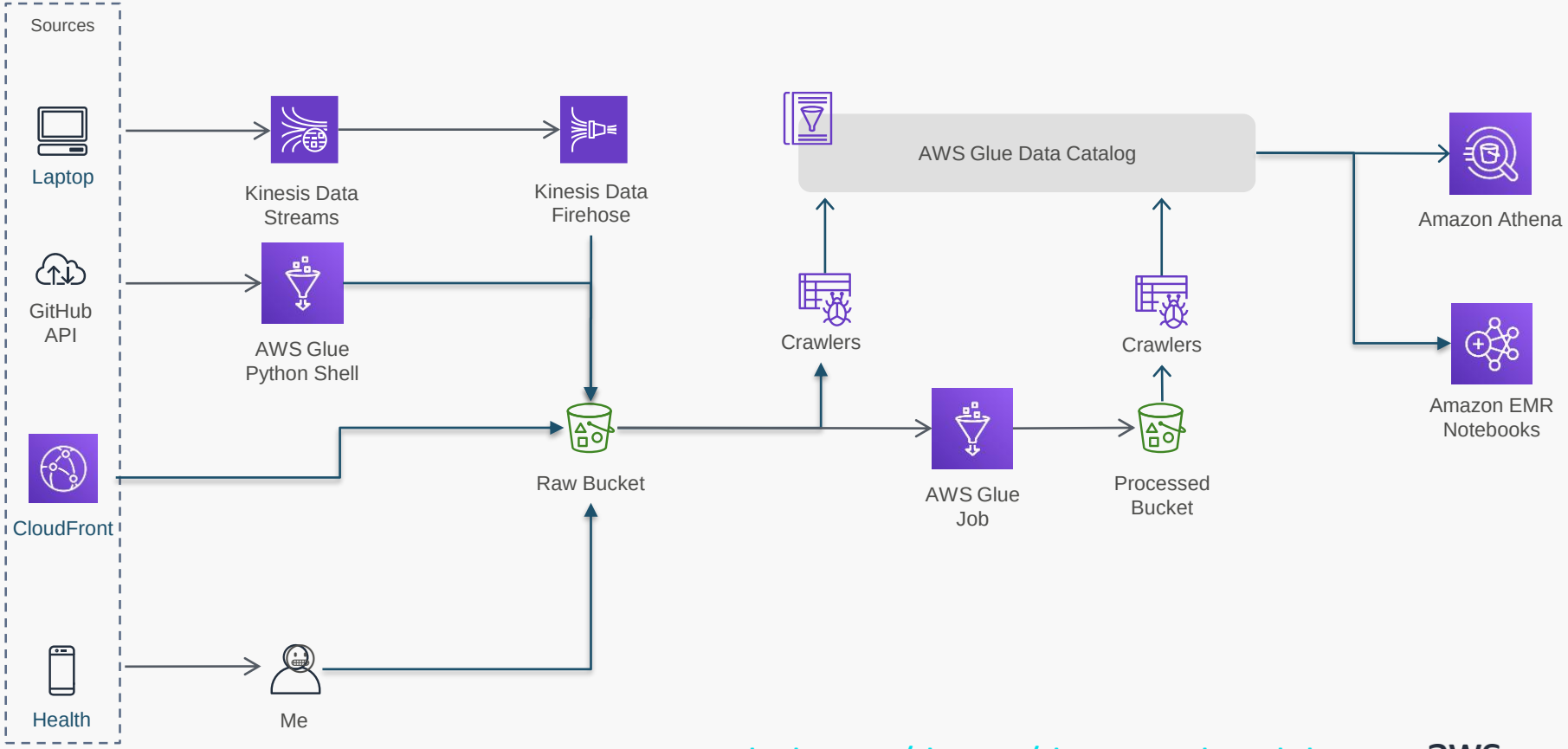
Run query | Save as | Create

(Run time: 2.18 seconds, Data scanned: 247.11 KB)

Results

	cnt	_col1
1	128	Go-http-client/1.1
2	68	Go-http-client/2.0
3	24	curl/7.58.0
4	21	Twitterbot/1.0
5	20	python-requests/2.21.0
6	19	Java/1.8.0_212
7	14	python-requests/2.22.0
8	11	RandomSurfer/1.0 (https://random.surf/bot)
9	10	Apache-HttpClient/4.5.2 (Java/1.8.0_171)
10	9	CheckMarkNetwork/1.0 (http://www.checkmarknetwork.com/spider.html)

What does this all look like?



github.com/dacort/damons-data-lake

Thank you!

Be sure to check out these re:Invent sessions!

- ANT202 - Turbocharge your Spark performance with Amazon EMR
- ANT205 - What's new with Amazon Athena
- ANT307 - Deep dive into Amazon Athena
- ANT308 - Deep dive into running Apache Spark on Amazon EMR