Building a Turbo-fast Data Warehousing Platform with Databricks

Parviz Deyhim

databricks

Agenda

- Introduction to Databricks
- Building a end-to-end Data warehouse platform
  - Infrastructure
  - Data ingest
  - ETL
  - Performance optimizations
  - Process & Visualize
- Securing your platform
- Conclusion

About the Speakers

Parviz Deyhim (Speaker)

Parviz works with variety of different customers and helps them with adopting Apache Spark and architecting scalable data processing platform with Databricks Cloud. Previous to joining Databricks, Parviz worked at AWS as a big-data solutions architect.

Denny Lee (Moderator)

Denny is a Technology Evangelist with Databricks. Previous to joining Databricks, Denny worked as a Senior Director of Data Sciences Engineering at Concur and was part of the incubation team that built Hadoop on Windows and Azure (currently known as HDInsight).

Introduction to Databricks

We are Databricks, the company behind Spark

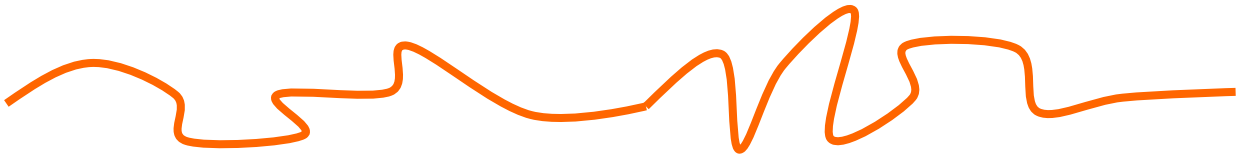
- Founded by the creators of Apache Spark
- Contributed ~75% of the Spark code in 2014
- Created Databricks cloud, a cloud-based big data platform on top of Spark to make big data simple

Typical big data project is far from ideal

Get a cluster up and running

Import and explore data

Build and deploy data applications



Months to build, weeks to provision in existing

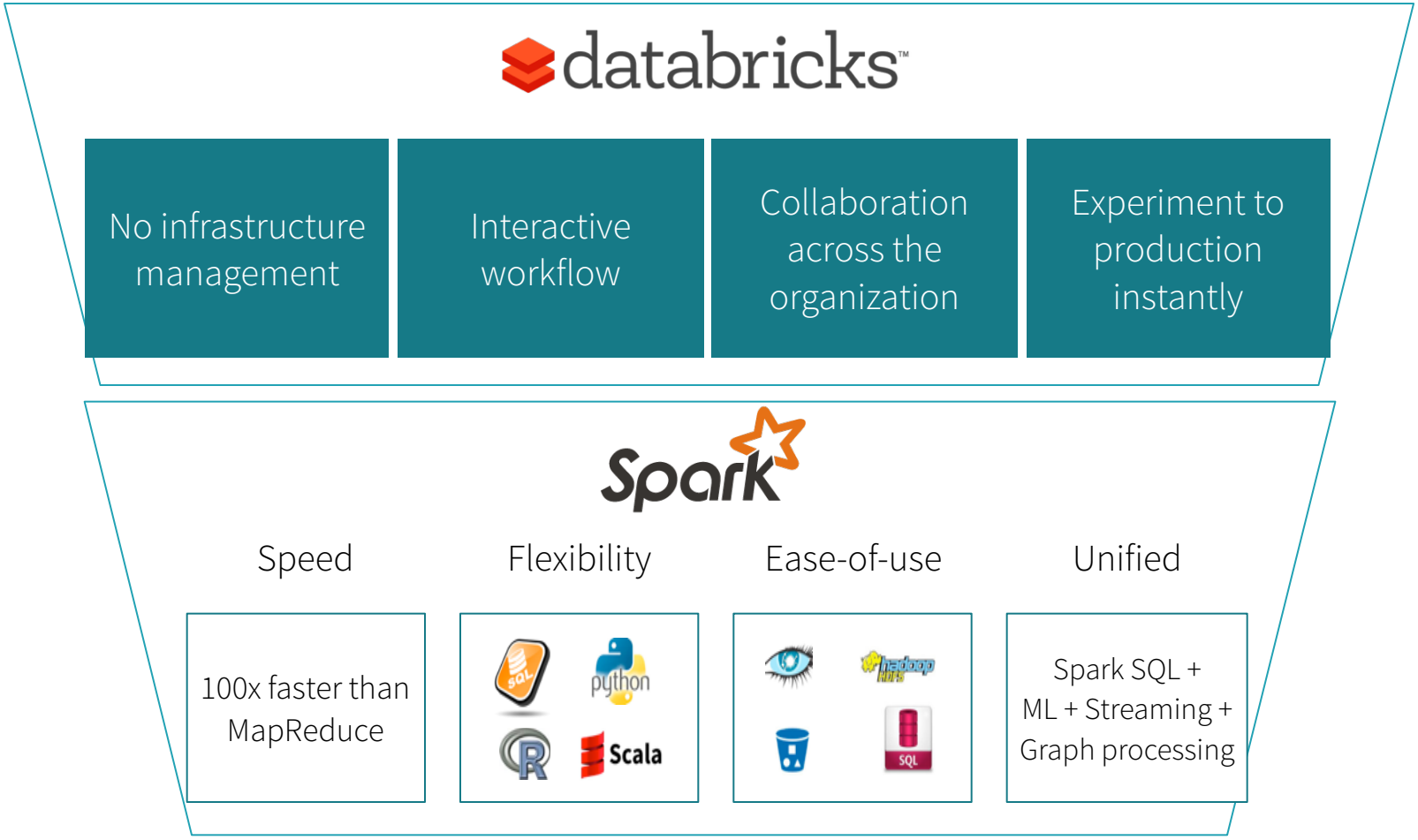
Weeks to prepare then explore data, and find insights

Months of re-engineering to deploy as an application

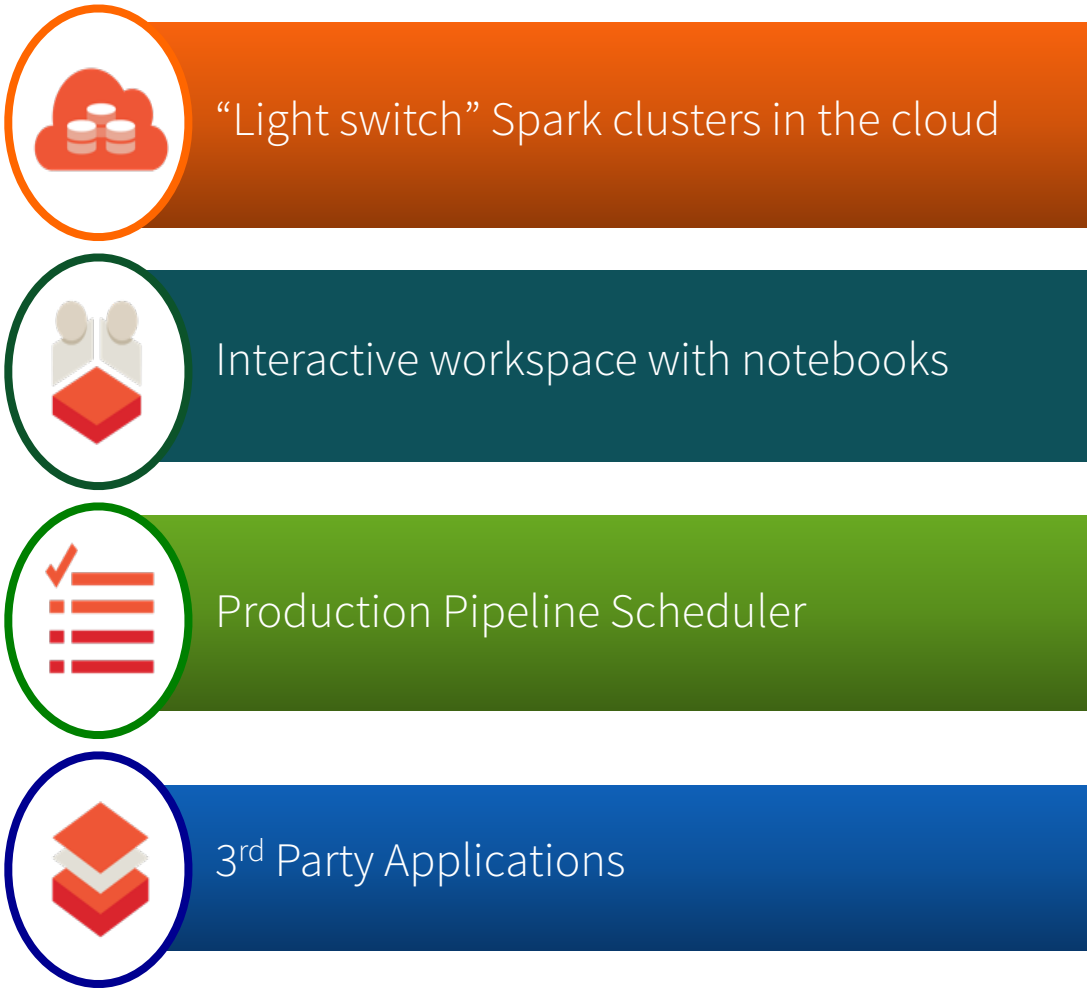
DevOps

For each new project, it takes months until results

How Databricks powered by Spark helps our customers



Databricks helps you to harness the power of Spark



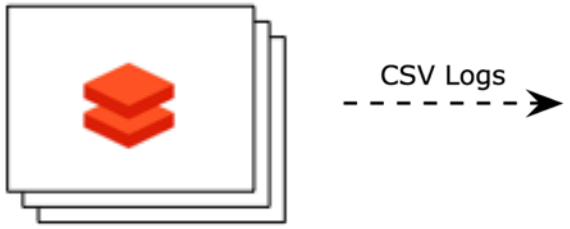
Databricks Internal Data Warehouse Use Case

Databricks Internal DWH Use Case

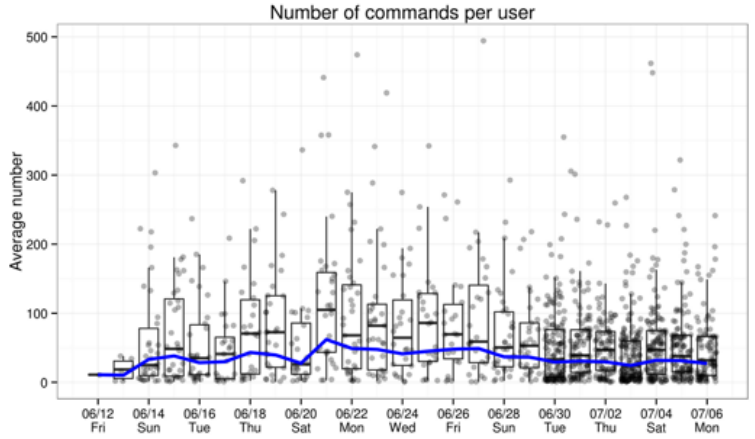
Today: Collect logs from deployed customer clusters

Our Goal:

- Understand customers behavior
- Create reports for various teams (e.g. customer success & support)



CSV Logs

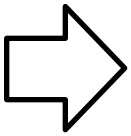


Stages

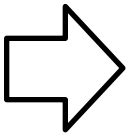
Build & Maintain Infrastructure  Data Ingest  Transform & Store  Process & Visualize

Stages

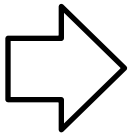
Build & Maintain Infrastructure



Data Ingest



Transform & Store



Process & Visualize

Challenges of Building a Data Warehouse

Datacenter or Cloud?

- Build/rent data center or use a public cloud offering?

Picking the right resources

- If datacenter: what server sizes and types? Storage?
- In cloud: what instance size, how large of a disk/SSD to use?

Deployment and Automation

- How to automate the deployment process:
  - Chef, Puppet, Cloudformation and etc

Challenges of Building a Data Warehouse

Maintenance

- How to perform seamless upgrades?

Securing the platform

- How to encrypt datasets?
- Controls, Policies, Audits

Databricks Hosted Platform

Managed and automated hosted platform

- Fully deployed on AWS
- Create resources with a single click
- Zero touch maintenance

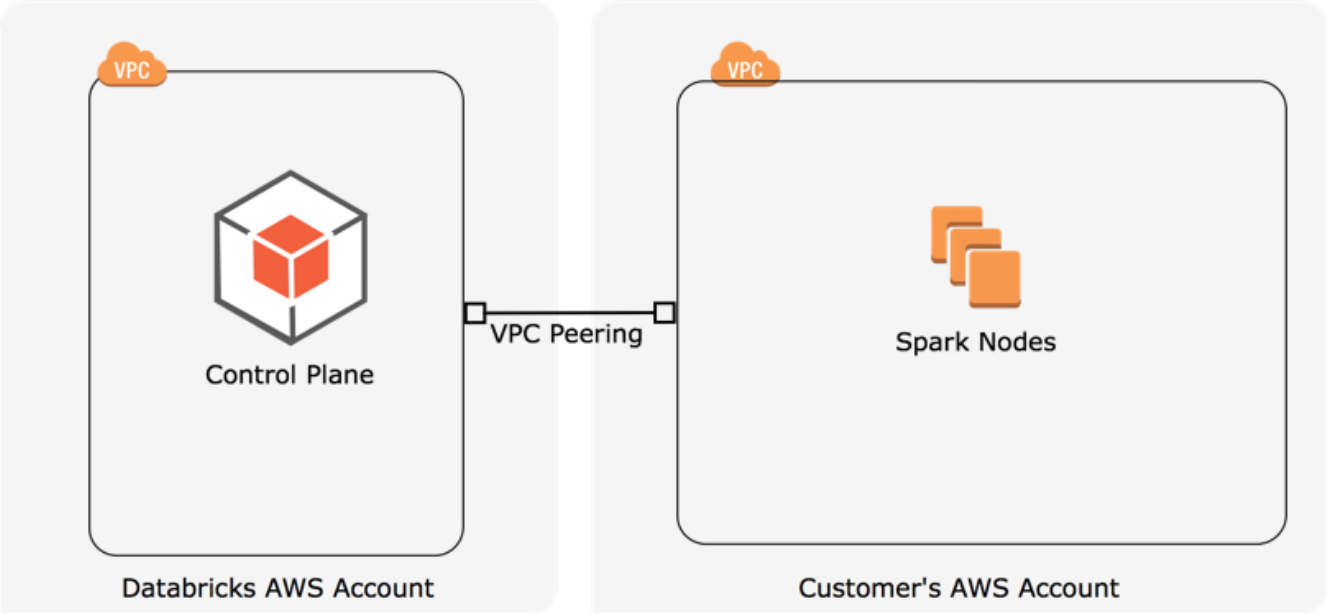
Compute Resources

Automatic Instance Provisioning

- R3.2xlarge instances
- Use SSD for caching
- No EBS
- Deployed in major regions and more coming

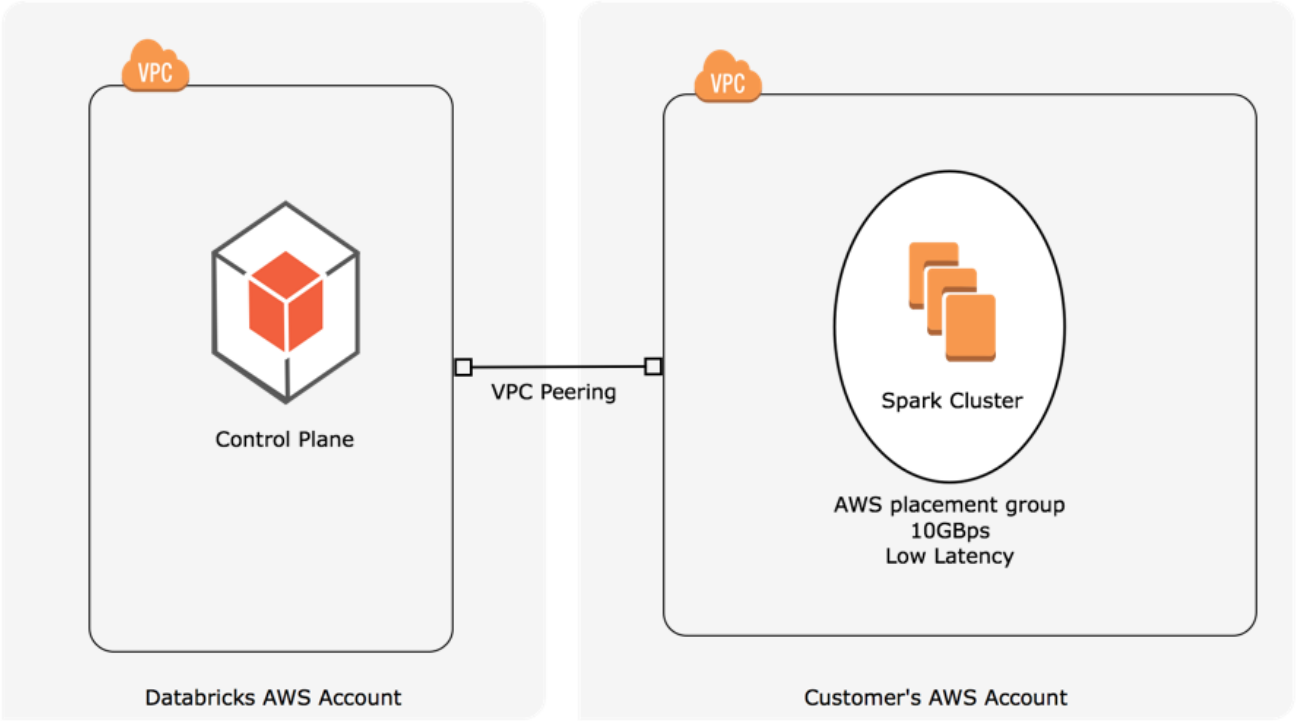
Networking: VPC

Security & Isolation with AWS VPC



Networking: Enhanced Networking

High performance node to node connectivity with placement groups



Integration with AWS services

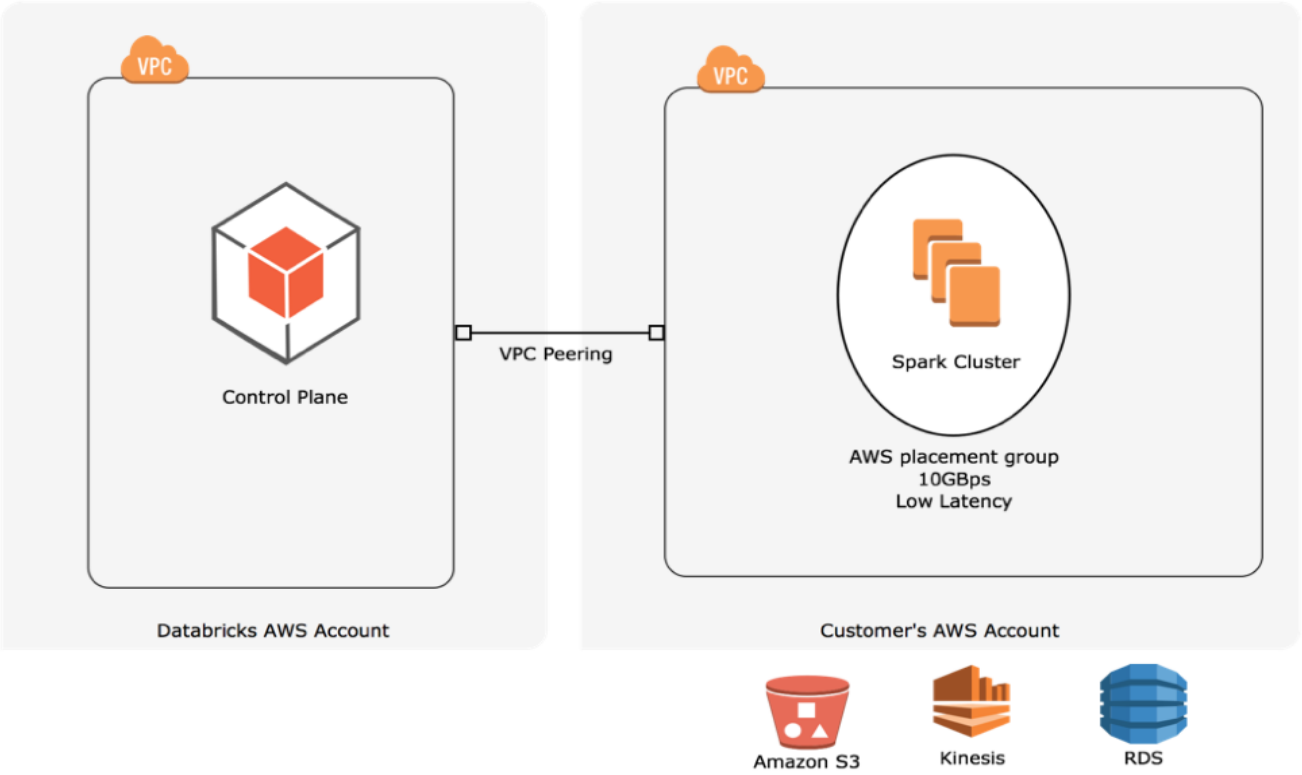
S3

Kinesis

RDS

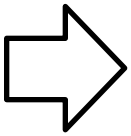
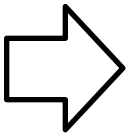
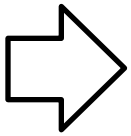
Redshift

...



Databricks Demo

Stages

Build & Maintain Infrastructure  Data Ingest  Transform & Store  Process & Visualize

Customer Data Sources

Customer have variety of different data sources

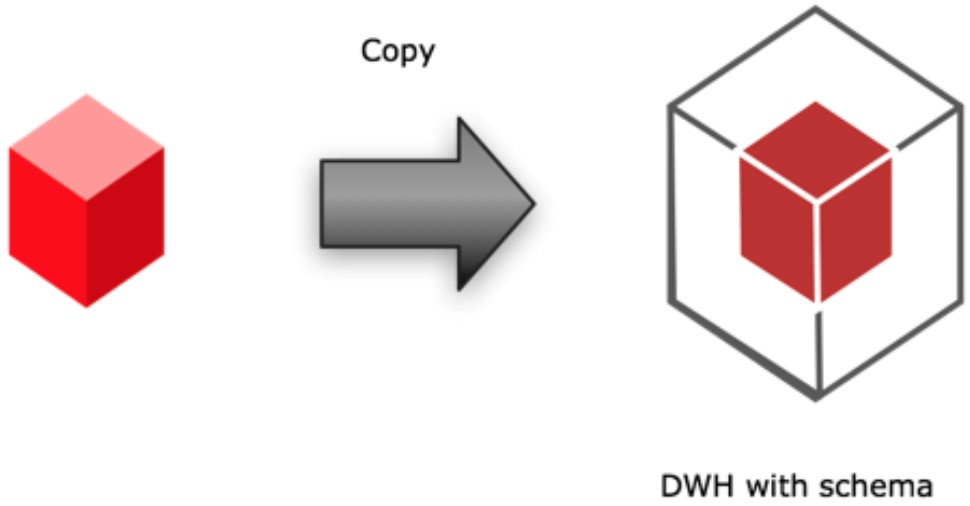
- Cloud storage: S3
- Databases: MySQL, NoSQL
- APIs: Facebook, SalesForce and etc

Often required to join datasets

Traditional Approach

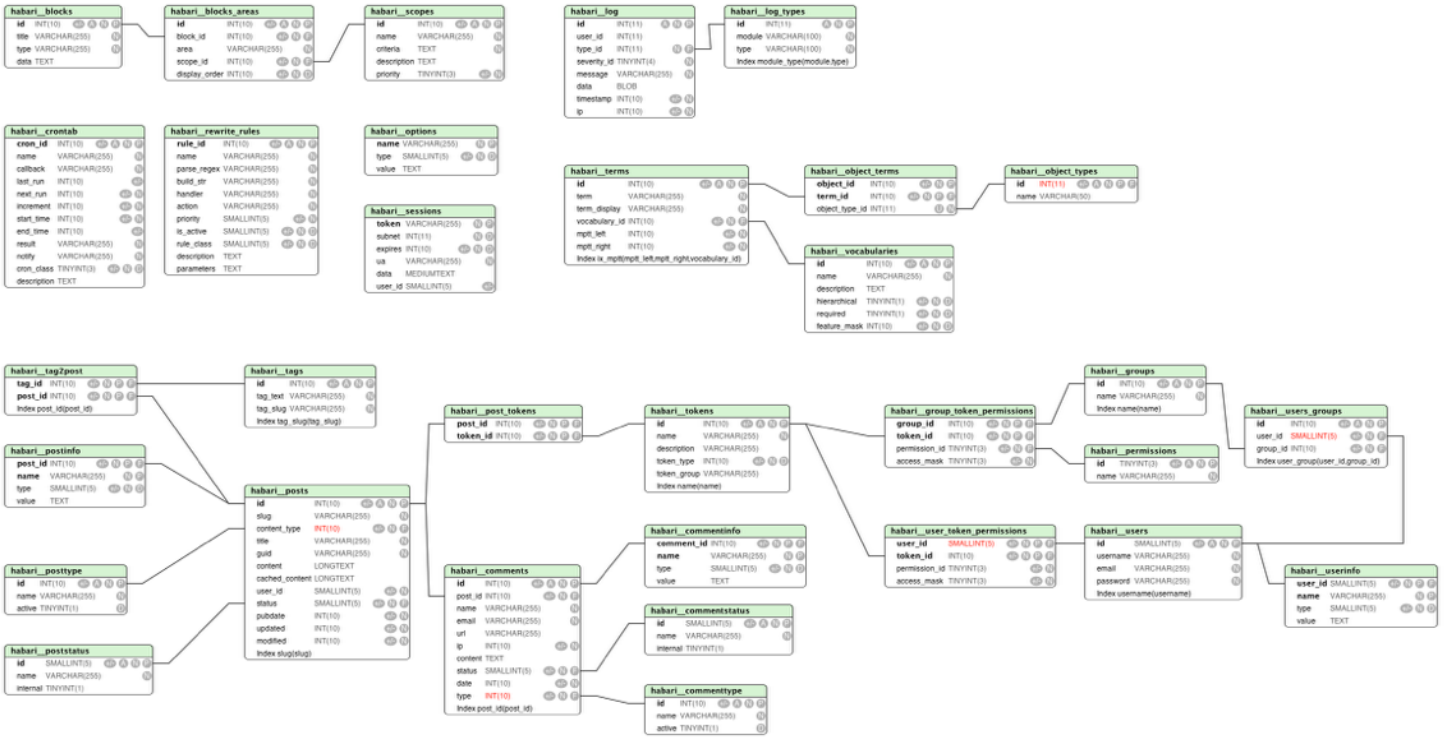
Traditionally data warehouses require data to be copied

Common Question: How do I move my datasets to Databricks?



Traditional Approach

Required to create a schema before data is copied



databricks

Traditional Approach: Challenges

Moving Data:

- Very expensive and time consuming
- Creates inconsistency as data gets updated

Predefined Schema:

- Challenging to change schema for different use-case

Databricks Approach: Data Sources

De-coupling compute from storage

- Leverage S3. No HDFS

Read directly from data sources

- Eliminate the need to copy data

Schema definition on read

- SparkSQL

Spark Data Sources Support

Formats and Sources supported by DataFrames

Databricks Use Case

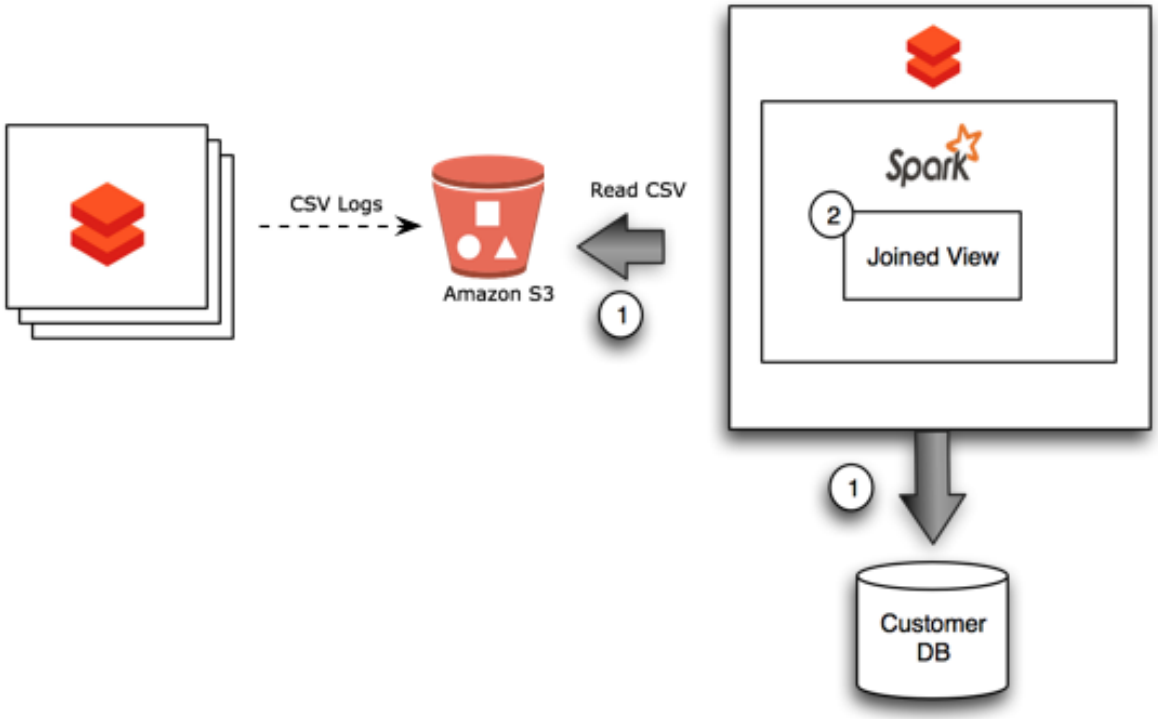
Different data sources

- Customer metrics on S3
- Internal CRM

Need a single view of our customers

Databricks Use Case

We use Spark to join datasets



Databricks Demo

1. Reading data from external API
2. Reading usage logs data from S3
3. Joining usage and external datasets

Link

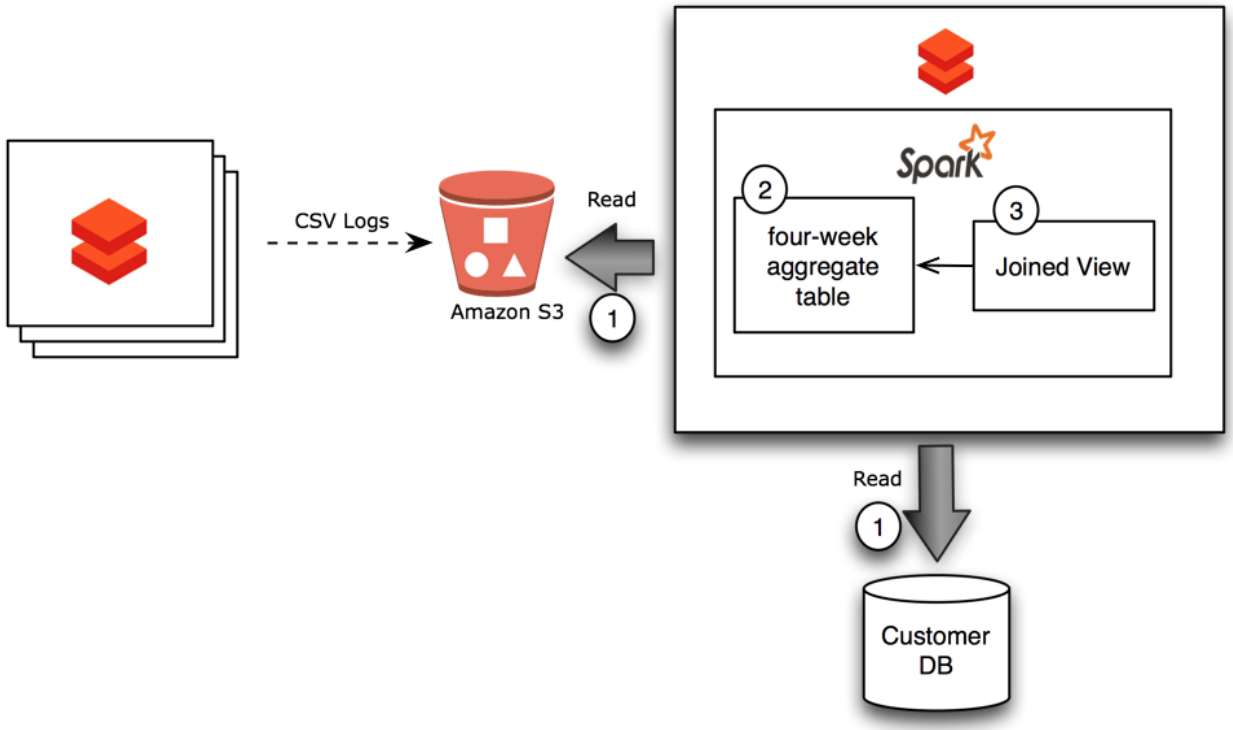
Stages


Build & Maintain Infrastructure
Data Ingest
Transform & Store
Process & Visualize

Data Transformation

Need to transform data before the join operation

- Aggregation
- Consolidation
- Data cleansing



Databricks Demo

Link

ETL: Common Approaches

Two common approaches

- Offline
- Streaming/real-time

Extract & Transformation

Offline ET

- Data gets stored in raw format (as is)
- Some recurring job perform ET on the dataset
- New transformed dataset gets stored for later processing

Advantage

- Easy and quick to setup

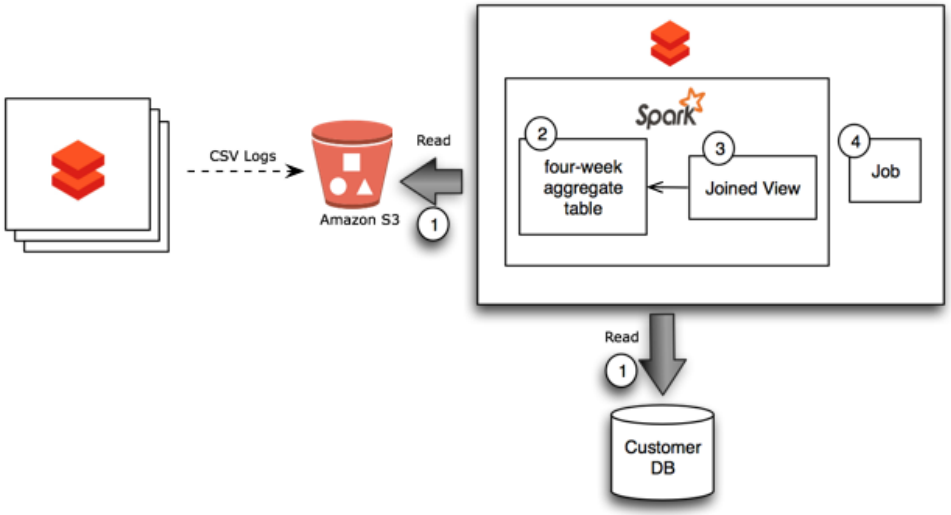
Disadvantages

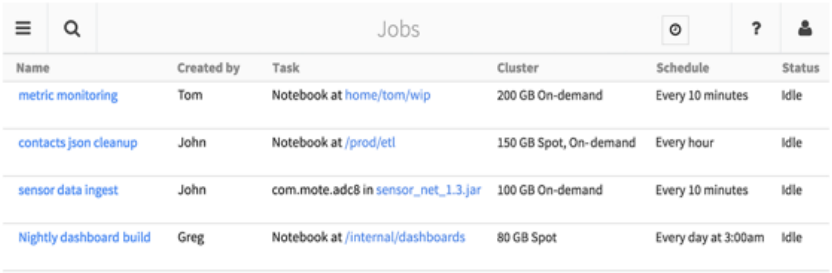
- Traditionally slow process

Databricks Jobs

Databricks Jobs

- Schedule Production workflows using Notebooks or Jars
- Create pipelines
- Monitor results





Databricks Demo

Jobs

Performance Optimizations

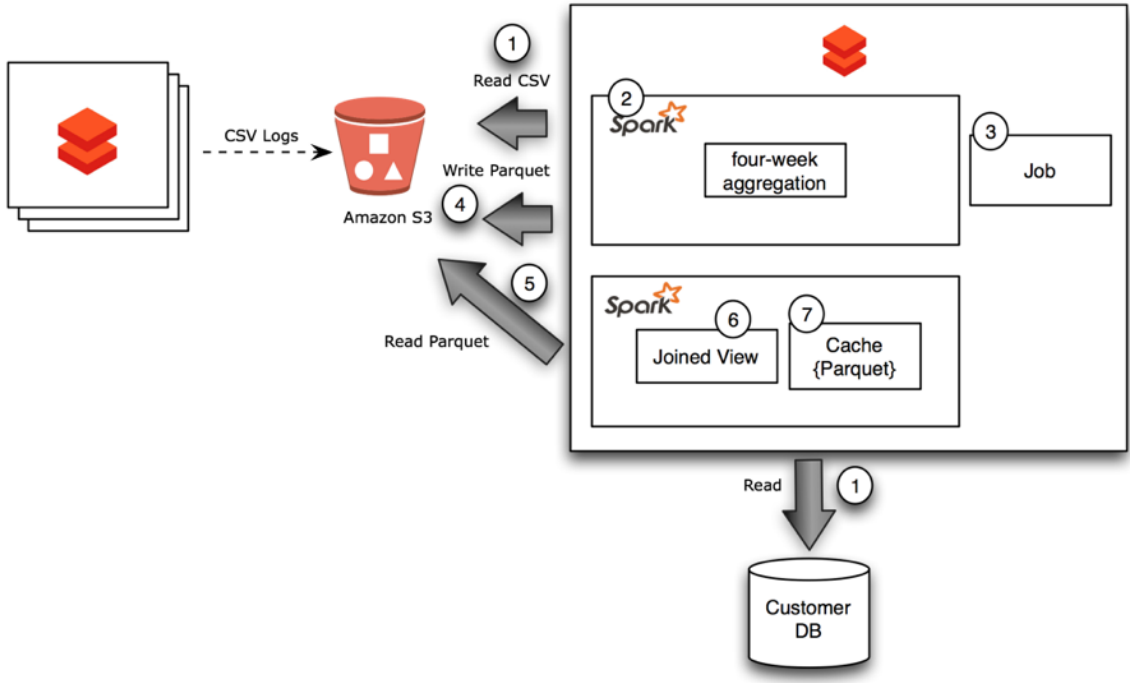
Performance Optimizations

Storing data in parquet

Partitioning dataset

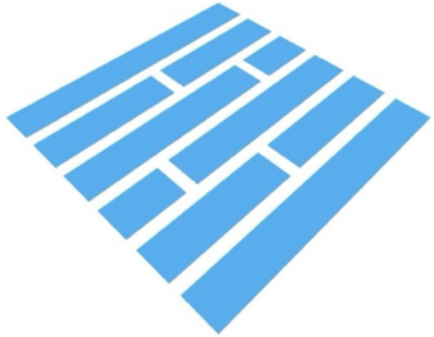
Spark caching

- JVM
- SSD



Spark allows data to be stored in different data sources

Parquet: Efficient columnar storage format for data warehousing use-cases



Optimization: Parquet

Columnar

- Faster Scans

Better compression

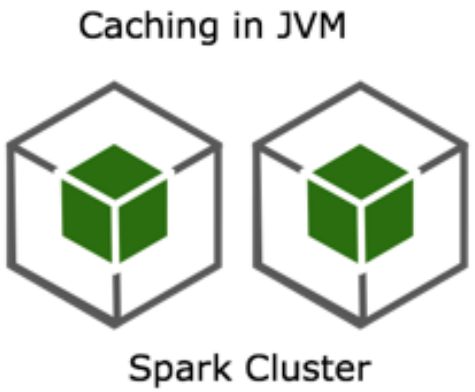
Optimized for storage

- Memory
- Disk

Optimization: Caching (JVM)

Spark caching: JVM

```
> %sql cache table usagelogs
```



Advantages

- Fast memory access

Disadvantages

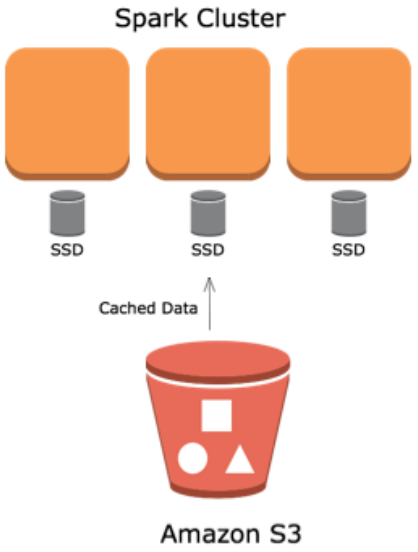
- GC pressure
- No durability after JVM crash

Optimization: Caching (SSD)

Spark caching: SSD

```
> dbutils.fs.cacheTable("usagelogs")
```

```
> dbutils.fs.cacheFiles("s3n://somebucket/usagelogstable/")
```



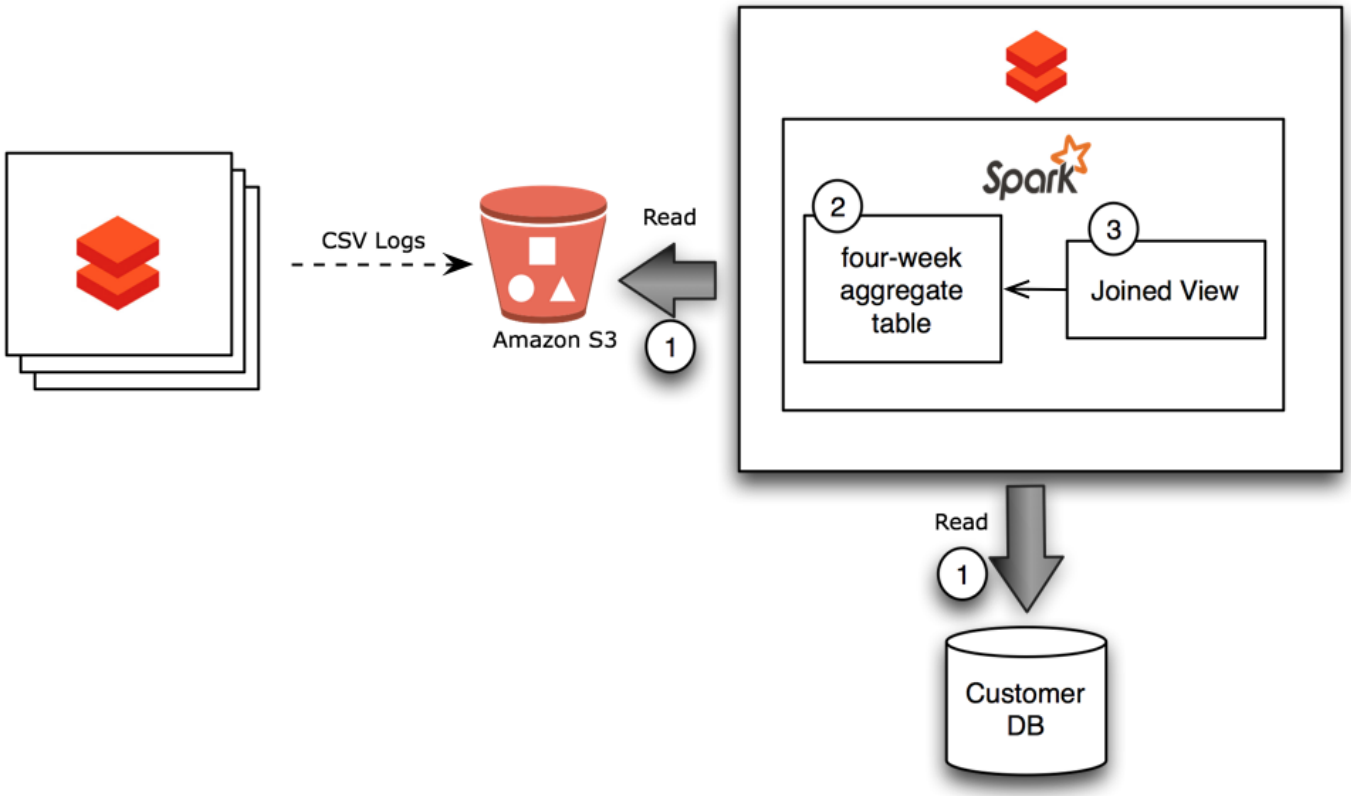
Advantages

- Survives JVM and instance crash

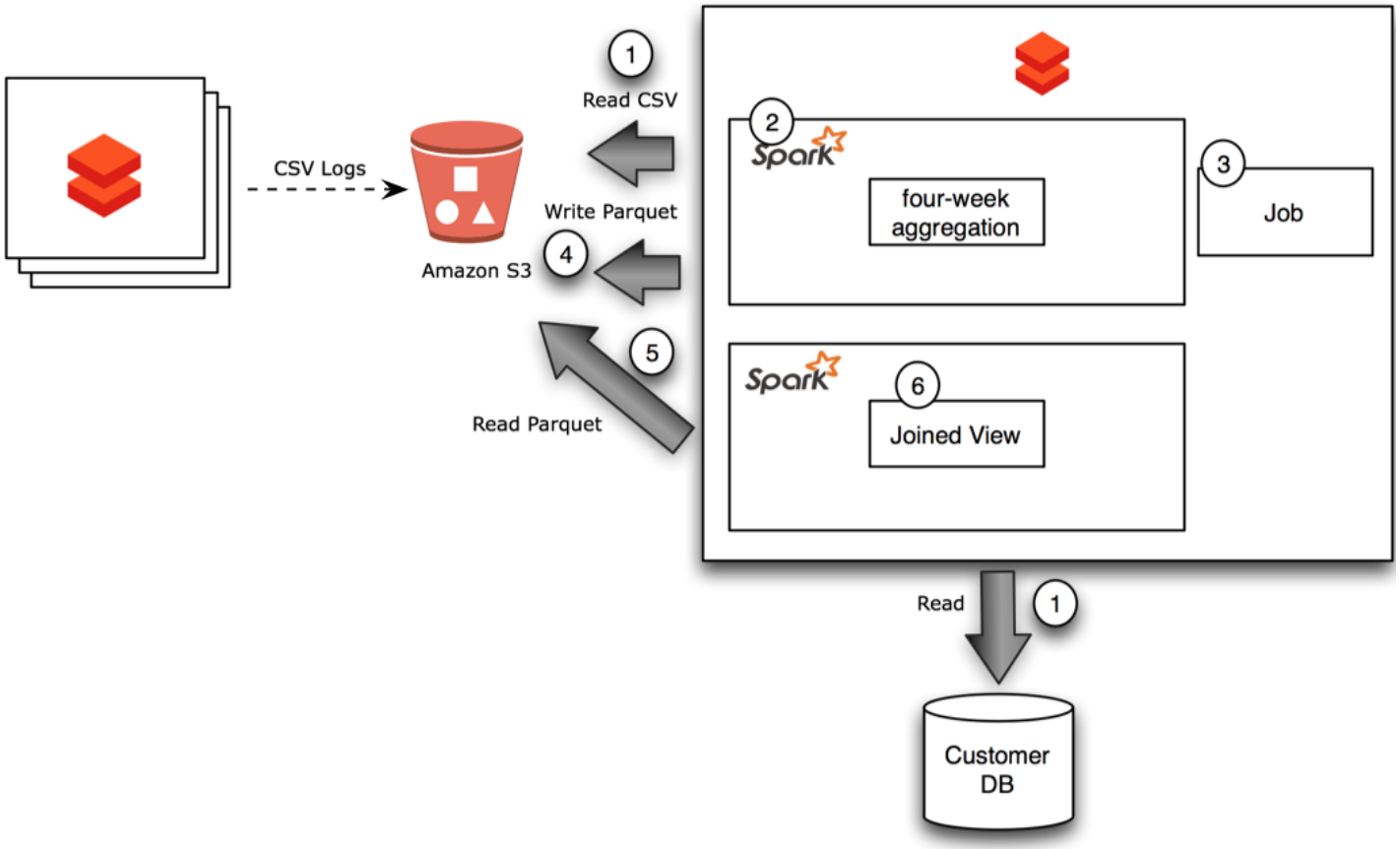
Disadvantages

- Much slower than JVM caching

Databricks Use Case: Storing aggregate data in Parquet



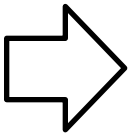
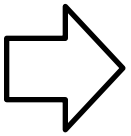
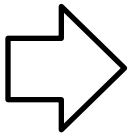
Databricks Use Case: Storing aggregate data in Parquet



Databricks Demo

Link

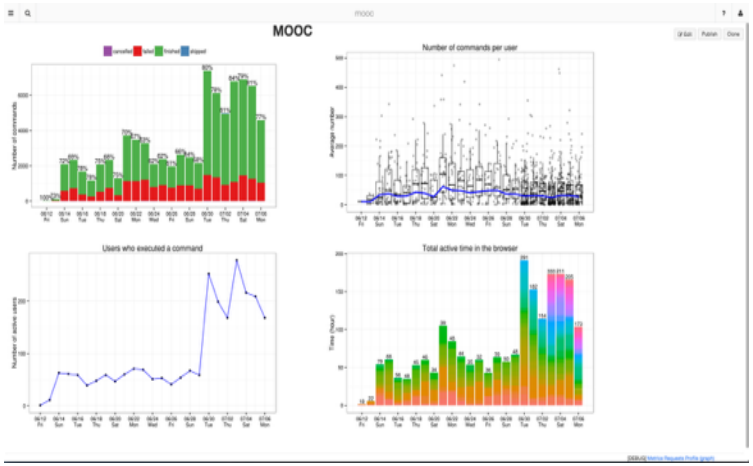
Stages

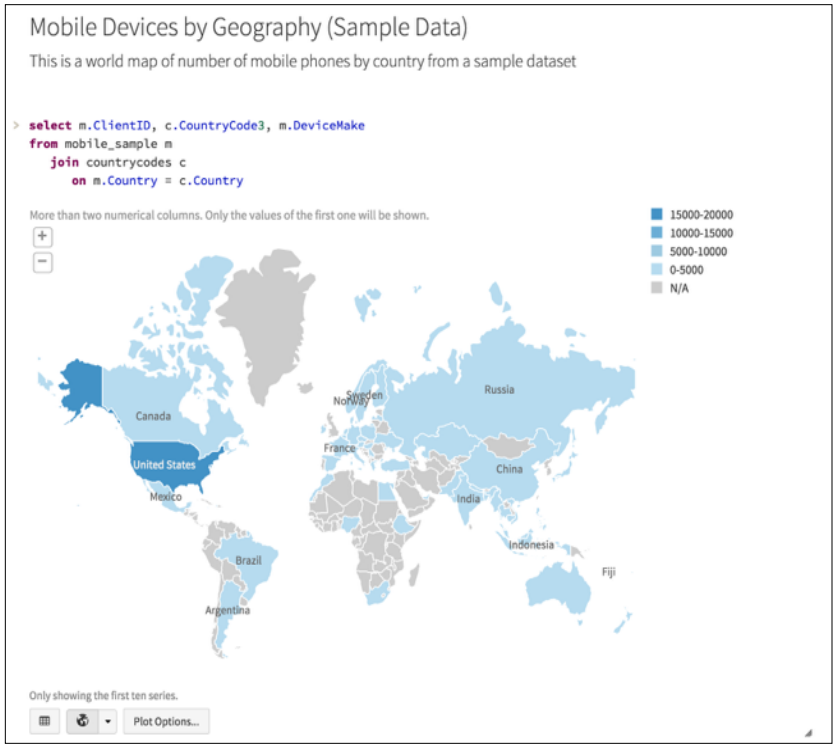
Build & Maintain Infrastructure  Data Ingest  Transform & Store  Process & Visualize

Databricks Visualizations

Notebook Visualizations

1. Built-in graphing capabilities
2. ggplot and matplotlib
3. D3 visualizations





databricks

Databricks Visualizations

Notebook Visualizations (DEMO)

D3/SVG

3rd Party Visualizations

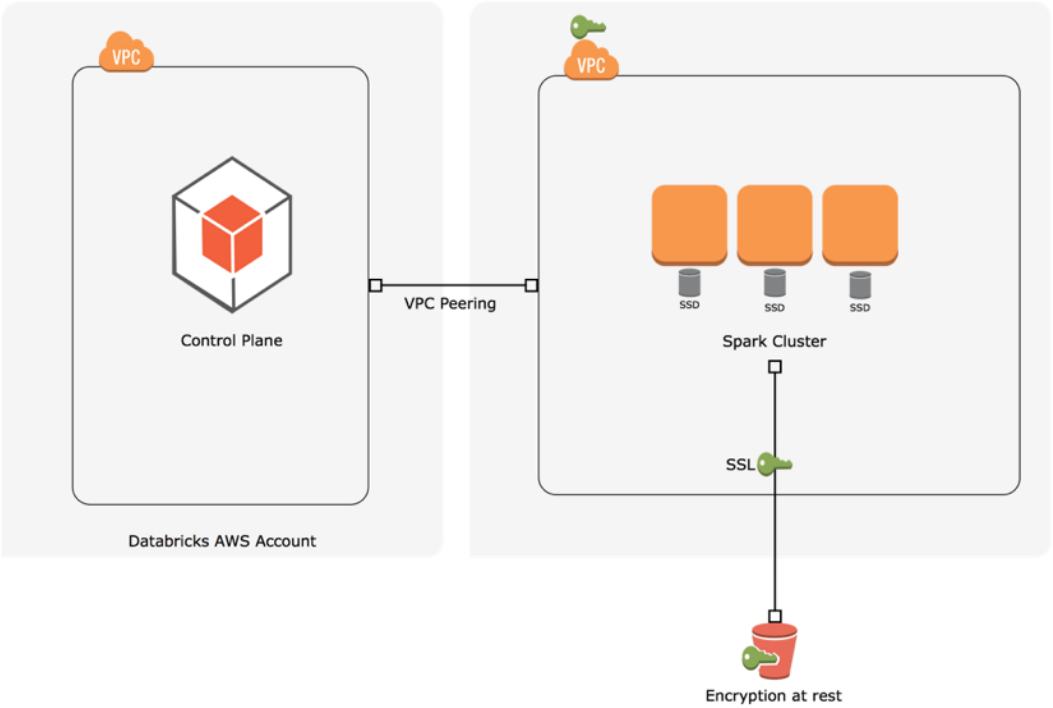
Zoomdata

Securing Your Platform

Secure Platform

Encryption

1. In flight: SSL
2. At rest: S3 Encryption

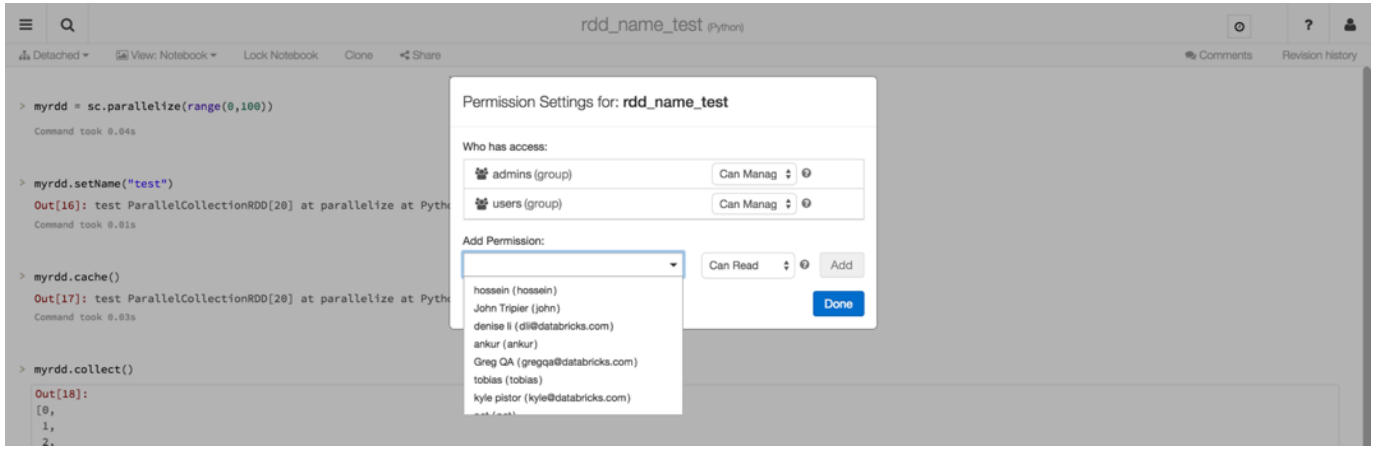


Secure Platform

User Management: ACLs

Notebooks read-write-execute

Admin users



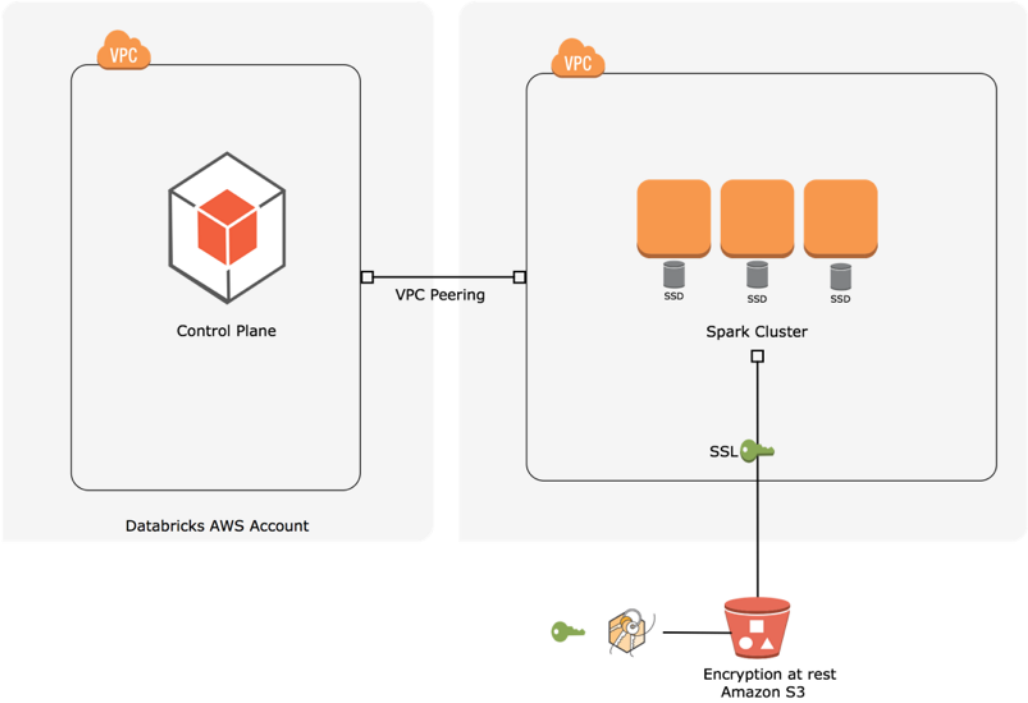
Secure Platform

On Our Roadmap

S3 KMS encryption

Single Sign On (SSO)

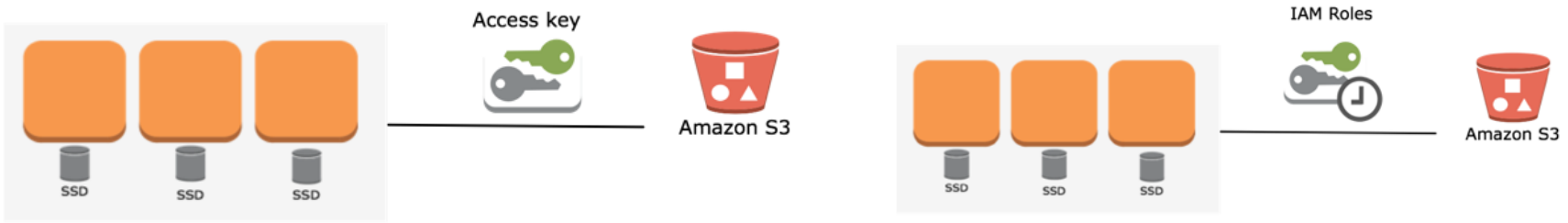
AD/LDAP support



Secure Platform

On Our Roadmap

IAM Roles for Spark nodes



Thank you