Big Data on AWS

Data Processing

Agenda

01 What is Amazon EMR?

02 Benefits of Amazon EMR

03 EMR Architecture

04 EMR Operations

05 EMR Applications

06 AWS Lambda

07 Glue

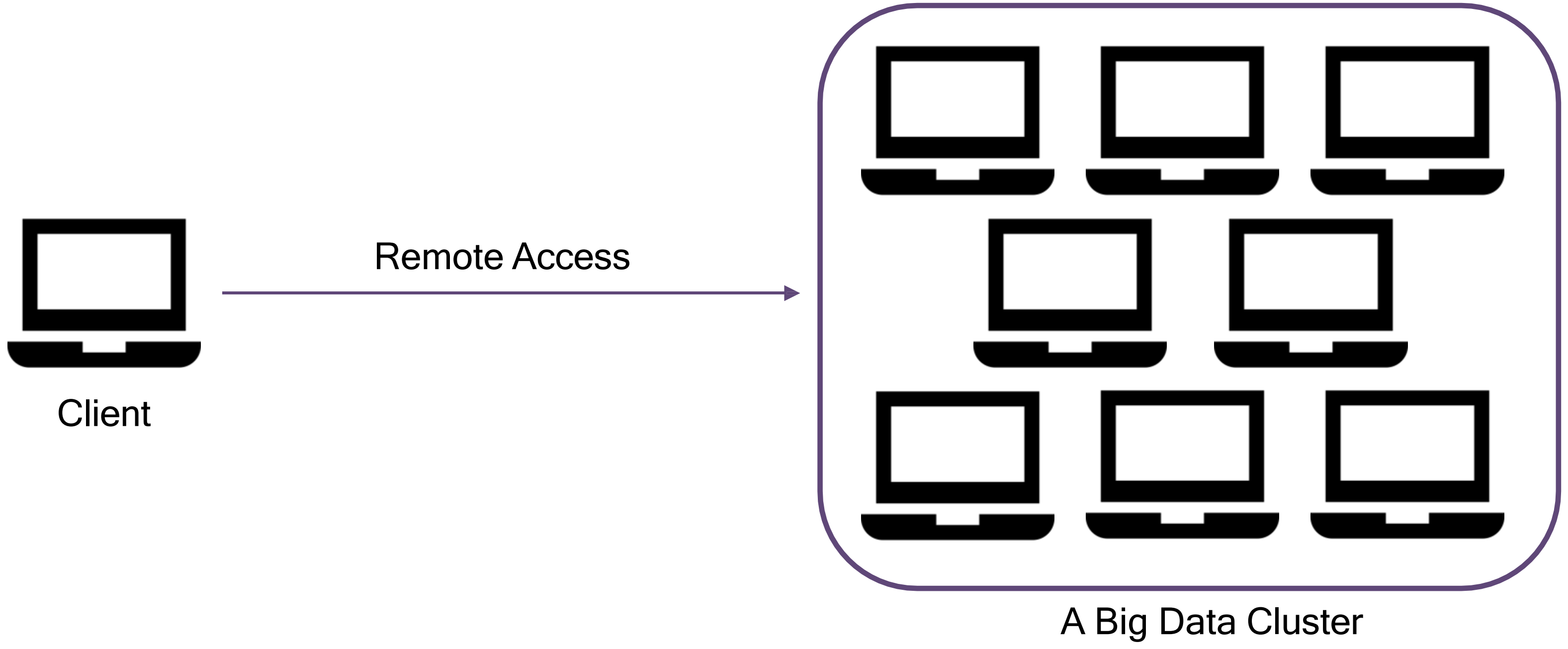
08 Benefits of Glue

09 Glue Workflow

What is Amazon EMR?

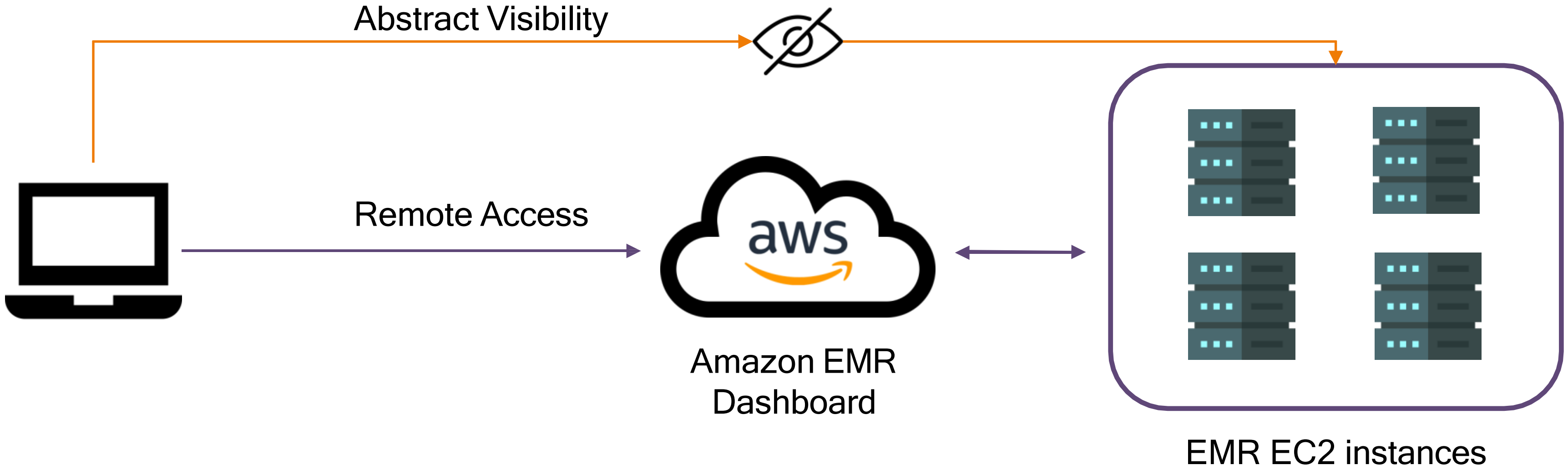
What is Amazon EMR?

The traditional way of implementing Big Data



What is Amazon EMR?

Using EMR to implement Big Data



What is Amazon EMR?

Amazon EMR is a **managed cluster platform** that **simplifies running big data frameworks**, such as Apache Hadoop and Apache Spark, on AWS to process and analyze vast amounts of data. By using these frameworks and related open-source projects, such as Apache Hive and Apache Pig, we can process data for analytics purposes and Business Intelligence workloads

Benefits of Amazon EMR

Benefits of Amazon EMR

1. Cost Efficient
2. Allows AWS Integration
3. Easy Deployment
4. Scalable and Flexible
5. Reliable
6. Secure
7. Easy to Monitor
8. Easy Management

amazon EMR

Benefits of Amazon EMR

1 Cost Efficient

- EMR pricing is done according to per-second usage
- We pay for the resources we use on a job duration basis
- No manual resources are required in the cluster or nodal setup
- Pricing depends on the instance type and region
- We can reduce the cost even further by purchasing Reserved Instances or Spot Instances

Benefits of Amazon EMR

2 Allows AWS Integration

- Amazon EC2 instances are used as various nodes for a cluster
- Amazon VPC is used for configuring the virtual network
- S3 can be used for storing the input and output data
- Amazon CloudWatch can be used to monitor cluster performance and configure alarms
- AWS Data Pipeline can be used to schedule and start our cluster

Benefits of Amazon EMR

3 Easy Deployment

- We can choose all frameworks we want to add before starting EMR (e.g., Apache Hadoop, Apache Spark, etc.)
- We can choose the instance/cluster size according to our need
- The instance type can also be chosen as per cluster purpose
- We can choose the applications that we want to add to our cluster (e.g., Apache Hive, Apache Pig, etc.)
- We can deploy the cluster with a simple click; no further configuration or installation steps are required

amazon EMR

Benefits of Amazon EMR

4 Scalable and Flexible

- It is easy to add new nodes to the cluster depending on requirements
- It is easy to remove nodes
- There is a simple option to increase and decrease instance volumes on the go
- We can choose between **EMR File System (EMRFS)** and **HDFS** for storage

EMRFS is used to separate the computational layer from the data layer. Amazon S3 is used as the data layer in this case. This way, our stored data persists after the life cycle of the cluster has ended

amazon EMR

Benefits of Amazon EMR

5 Reliable

- All nodes are constantly monitored
- Any failed instances will automatically be replaced with functional ones
- Clusters will self-terminate or remain dependent on the settings

6 Secure

- The usage of IAM
- Amazon VPC
- EC2 Key Pairs
- AWS CloudTrail
- Security Groups

Benefits of Amazon EMR

7 Easy to Monitor

- EMR provides the ability to store cluster log files in S3
- CloudWatch is used to obtain performance metrics

8 Easy Management

- We can access our cluster using AWS Console
- We can access our cluster using AWS CLI as well
- Amazon provides SDKs to create and manage clusters

EMR Architecture

EMR Architecture

EMR architecture **remains identical** to the normal Hadoop Cluster architecture

Local file systems of the nodes, **HDFS** or **EMRFS**, are used as storage modes for EMR

Using the EMR File System (EMRFS), Amazon EMR extends Hadoop to add the ability to directly access the data stored in Amazon S3 as if it were a file system like HDFS. We can use either HDFS or Amazon S3 as the file system in our cluster. Most often, Amazon S3 is used to store input and output data, and intermediate results are stored in HDFS

Storage
hadoop HDFS
Amazon S3
EBS Block Storage

EMR Architecture

YARN is used as the **resource management layer**, by default, which is a component used by Apache Hadoop 2.0

Amazon EMR release **version 5.19.0 and later uses the built-in YARN node labels feature to achieve this** (earlier versions used a code patch). Properties in the yarn-site and capacity-scheduler configuration classifications are configured by default so that the YARN capacity-scheduler and fair-scheduler take advantage of node labels. Amazon EMR automatically labels core nodes with the CORE label and sets properties so that application masters are scheduled only on nodes with the CORE label. Manually modifying related properties in the yarn-site and capacity-scheduler configuration classifications, or directly in associated XML files, could break this feature or modify this functionality

Resource Management

EMR Architecture

The main processing frameworks available for **Amazon EMR** are **Hadoop MapReduce** and **Spark**

EMR Operations

Hands-on: Launching a Default Cluster

Launching a Default Cluster

- Select **Create Cluster** after opening EMR service
- Set Configurations for the cluster using default options
- Select a key to be able to SSH into the master node
- Launch the cluster with the set settings

Hands-on: Launching an Advanced Cluster

Launching an Advanced Cluster

- Select **Create Cluster** after opening EMR service
- Set Configurations for the cluster using the **advanced options** and include all add-ons needed
- Select a key to be able to SSH into the master node
- Launch the cluster with the set settings

EMR Applications

EMR Applications

Amazon EMR supports many applications, such as **Hive, Pig,** and the **Spark Streaming library**, to provide capabilities such as using higher-level languages to create processing workloads, leveraging Machine Learning algorithms, making stream processing applications, and building data warehouses. In addition, Amazon EMR also supports open-source projects that have their own cluster management functionality instead of using YARN

HIVE

Apache Pig

Spark MLlib

HUE

Hands-on: Hue with EMR

Hue with EMR

- Create an EMR cluster and make sure it has a SSH key and Hue installed
- Download PuTTY.exe to the computer
- Start PuTTY
- In the Category list, click on Session
- In the Host Name field, type the SSH host name
- In the Category list, expand Connection > SSH > Auth
- For the Private key file for authentication, click on Browse and select the private key file used to launch the cluster
- In the Category list, expand Connection > SSH, and then click on Tunnels
- In the Source port field, type 8157 (a randomly chosen, unused local port)
- Select Dynamic and Auto options
- Leave the Destination field empty and click on Add
- Click on Open
- Click Yes to dismiss the security alert

Hue with EMR (Contd.)

- Download and install the standard version of FoxyProxy
- Restart Chrome after installing FoxyProxy
- Using a text editor, create a file named **foxyproxy-settings.xml**
- Click on the FoxyProxy icon in the toolbar and select Options
- Click on Import/Export
- Click on Choose File, select foxyproxy-settings.xml, and then click on Open
- In the Import FoxyProxy Settings dialog, click on Add
- At the top of the page, for Proxy mode, choose Use proxies based on their predefined patterns and priorities
- To open the web interface, in the browser's address bar, type master-public-dns followed by the port number or URL

Hands-on: Hive with EMR

Hive with EMR

- Create an **Advanced** or **Default Cluster** with **Hue and Hive** installed and a key assigned for SSH access
- Upload the input dataset on the S3 bucket
- Create a Hive import table using the dataset from S3
- Perform queries on the Hive table

Hands-on: HBase with EMR

HBase with EMR

- Create an **Advanced** or **Default Cluster** with **Hive and HBase** installed and a key assigned for SSH access
- SSH into the EMR Master Node using PuTTY
- Create a table using HBase Shell
- Create an external table in Hive using the above HBase table
- Query the Hive table using HiveQL

Hands-on: Spark with EMR

Spark with EMR

- Create an **Advanced** or **Default Cluster** with **Spark** installed and a key assigned for SSH access
- SSH into the EMR Master Node using PuTTY
- Start Spark Shell
- Create an RDD out of a text file in S3
- Perform desired Spark operations

AWS Lambda

AWS Lambda

AWS Lambda is a ‘serverless’ compute service where our code is triggered in response to an event occurrence, and it also manages the underlying compute resources for us!

AWS Lambda

AWS Lambda	Amazon EC2
AWS lambda is a Platform as a Service (PaaS) with a remote platform to run and execute out backend code	AWS EC2 is an Infrastructure as a Service (IaaS) provided with virtualized computing resources
Environment restrictions are there as it is restricted to a few languages only	No environment restrictions are here
We can choose our environment where we want to run our code and push the code into AWS Lambda	For the first time in EC2, we have to choose the OS and install all the required software and then push our code in EC2
No flexibility to log in to compute instances; we need to choose a customized OS or language runtime	It offers the flexibility to choose variety of instances, custom operating systems, network and security patches, etc.

AWS Lambda

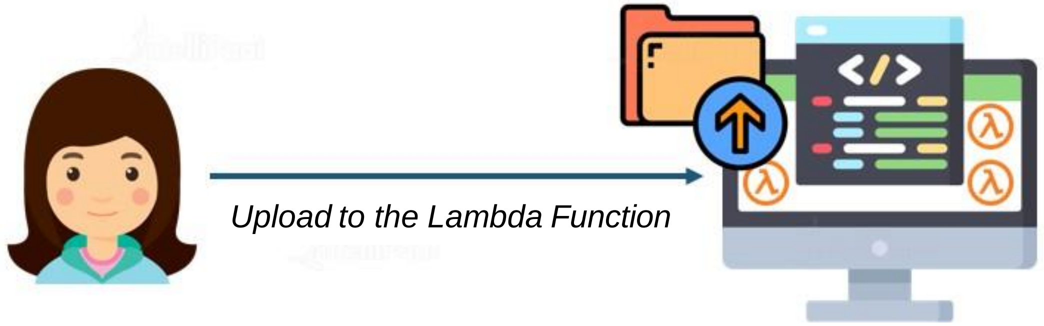
- ✓ Serverless architecture
- ✓ Code freely
- ✓ No need to create VMs
- ✓ Pay-as-you-go pricing
- ✓ Monitor performance

AWS Lambda

- ✖ The maximum disk space provided is 512 MB for the runtime environment
- ✖ Memory volume varies between 128 and 3008 MB to the function while execution
- ✖ Function timeout is set to only 900 seconds (15 minutes). The default is 3 seconds
- ✖ Only the available languages in the Lambda Editor can be used

AWS Lambda

- First, we upload our code to Lambda in one or more Lambda functions
- AWS Lambda will then execute the code on our behalf
- After the code is invoked, Lambda automatically takes care of provisioning and merging the required servers



AWS Lambda

- First, we upload our code to Lambda in one or more Lambda functions
- AWS Lambda will then execute the code on our behalf
- After the code is invoked, Lambda automatically takes care of provisioning and merging the required servers

Provisioning Servers

Monitoring and Managing Servers

Glue

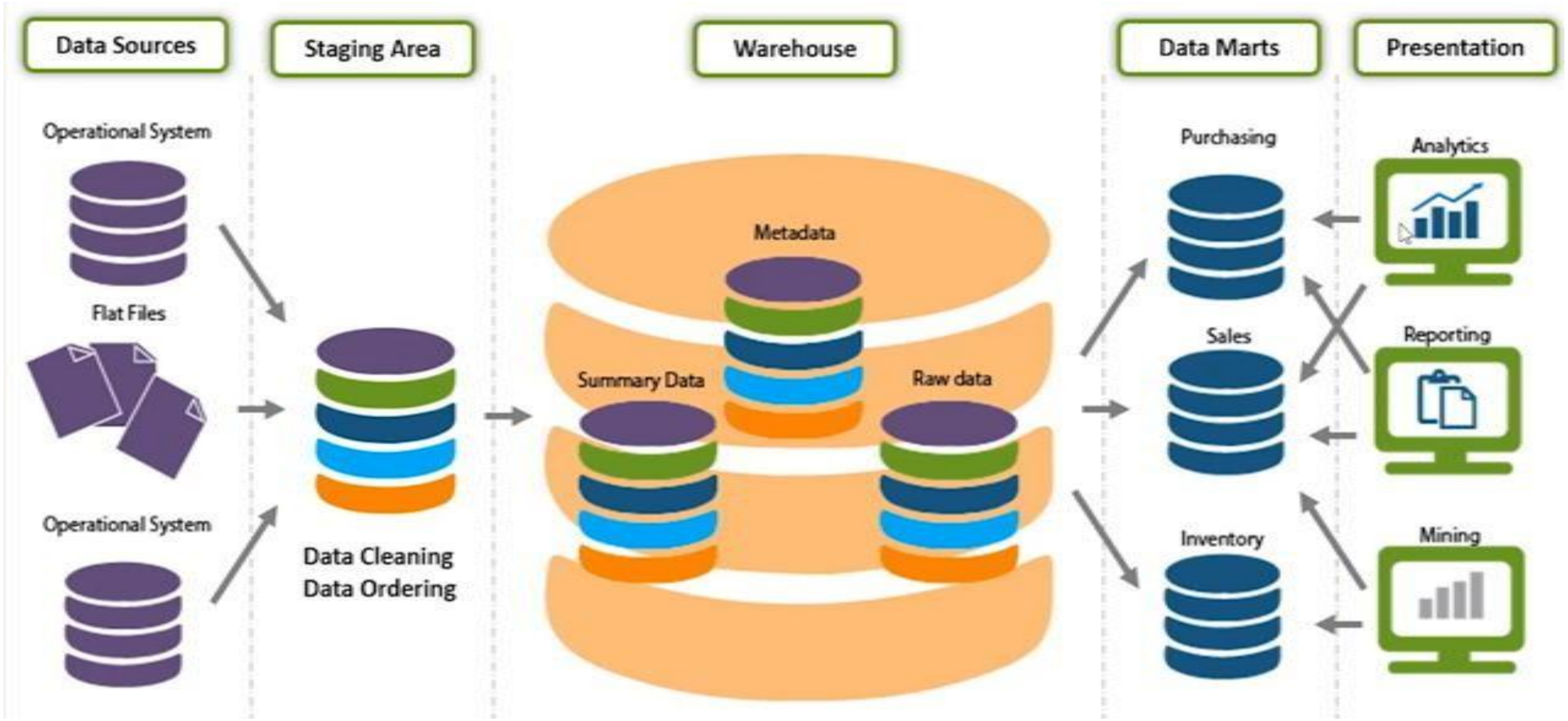
Glue

AWS Glue is a fully managed **extract, transform**, and **load** (ETL) service that makes it easy for customers to prepare and load their data for analytics

Amazon Glue

Extract
Load
Transform

Glue (Working of an ETL Tool)



Benefits of Glue

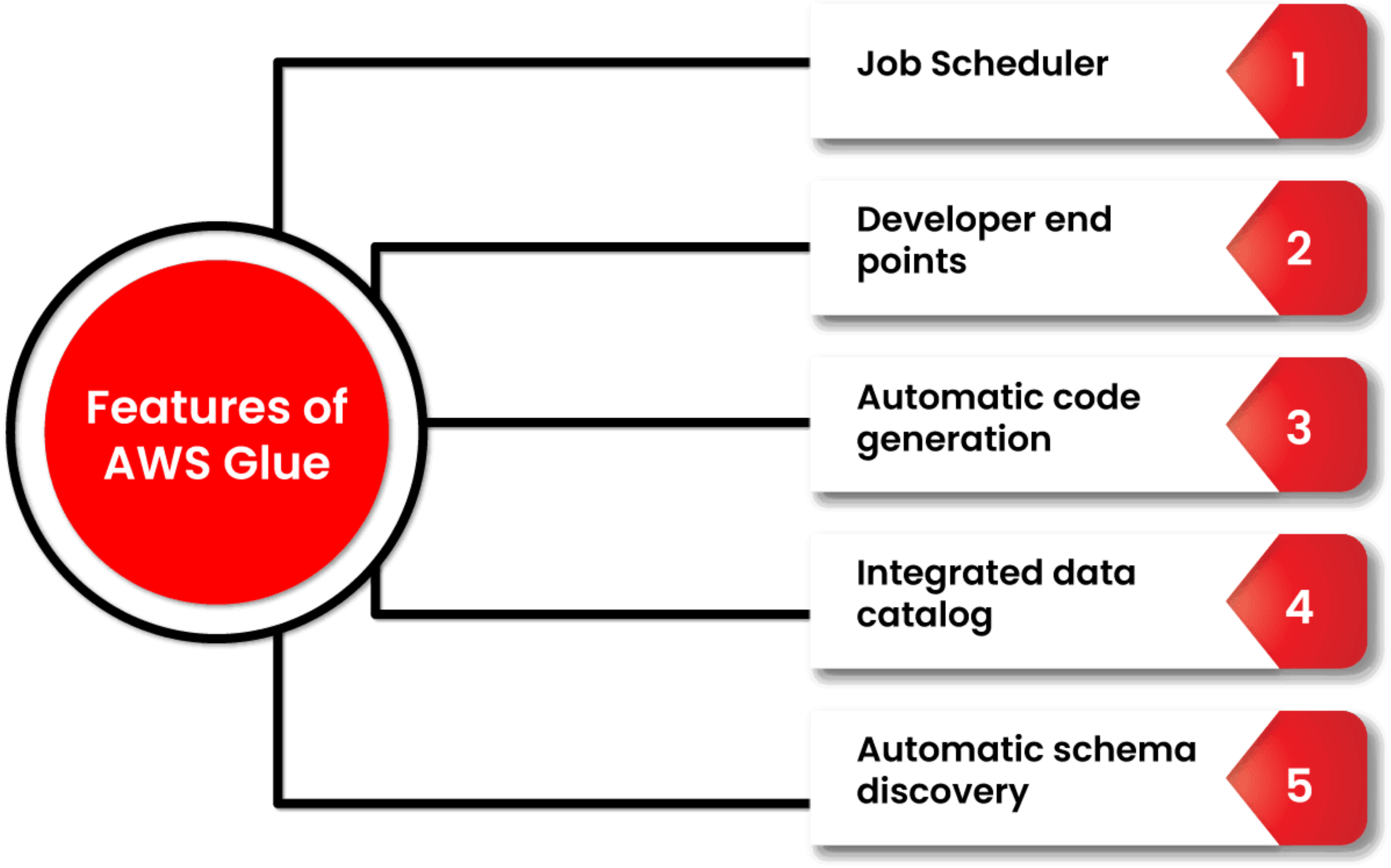
Benefits of Glue

Easy Management: AWS provides an integrated environment with a plethora of services. Since Glue and all of these services are managed, no installation and manual integration are involved

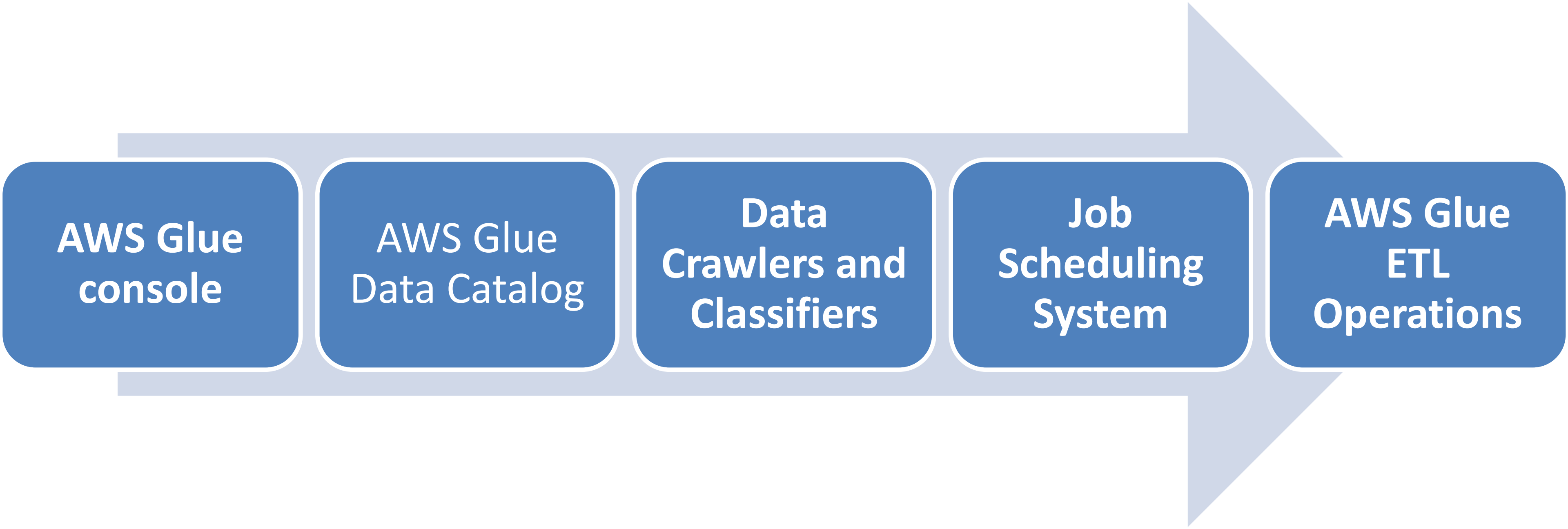
Cost Efficient: There is no infrastructure that requires management as AWS Glue is serverless. Payment is based solely on the duration of service usage

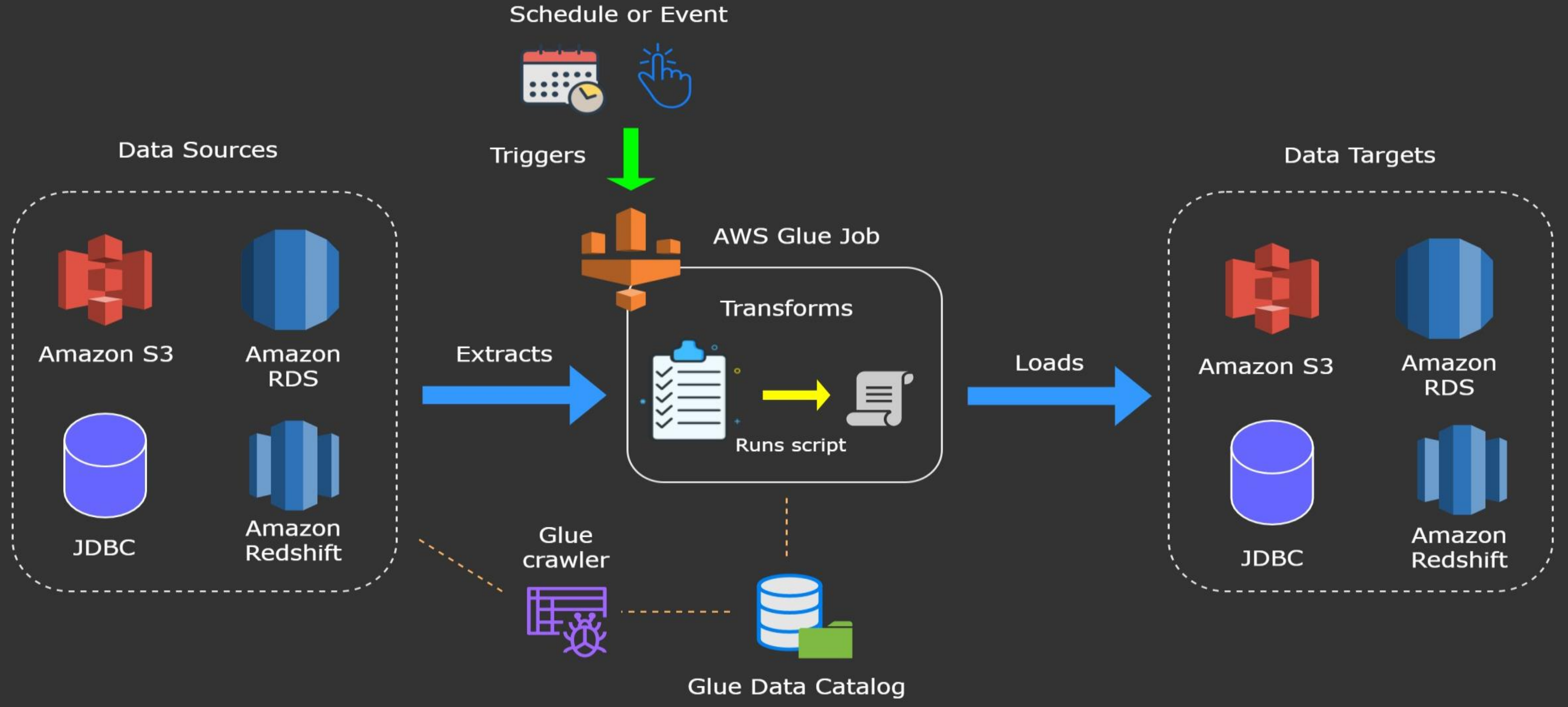
Superior Functionality: AWS Glue automates much of the building, maintaining, and running of ETL jobs. It crawls into our data sources, identifies data formats, and suggests schemas and transformations. It automatically generates the code to execute our data transformations and loading processes

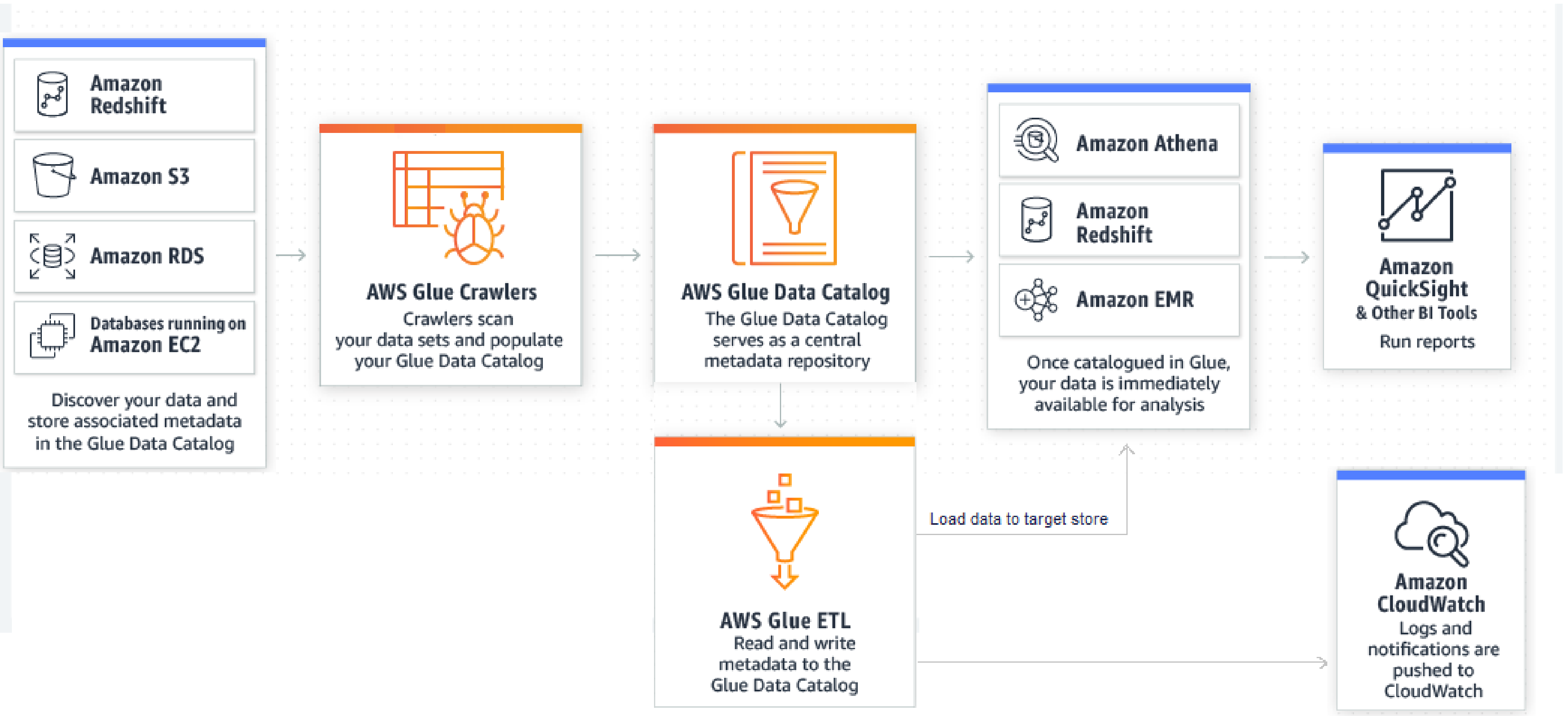
Glue Workflow

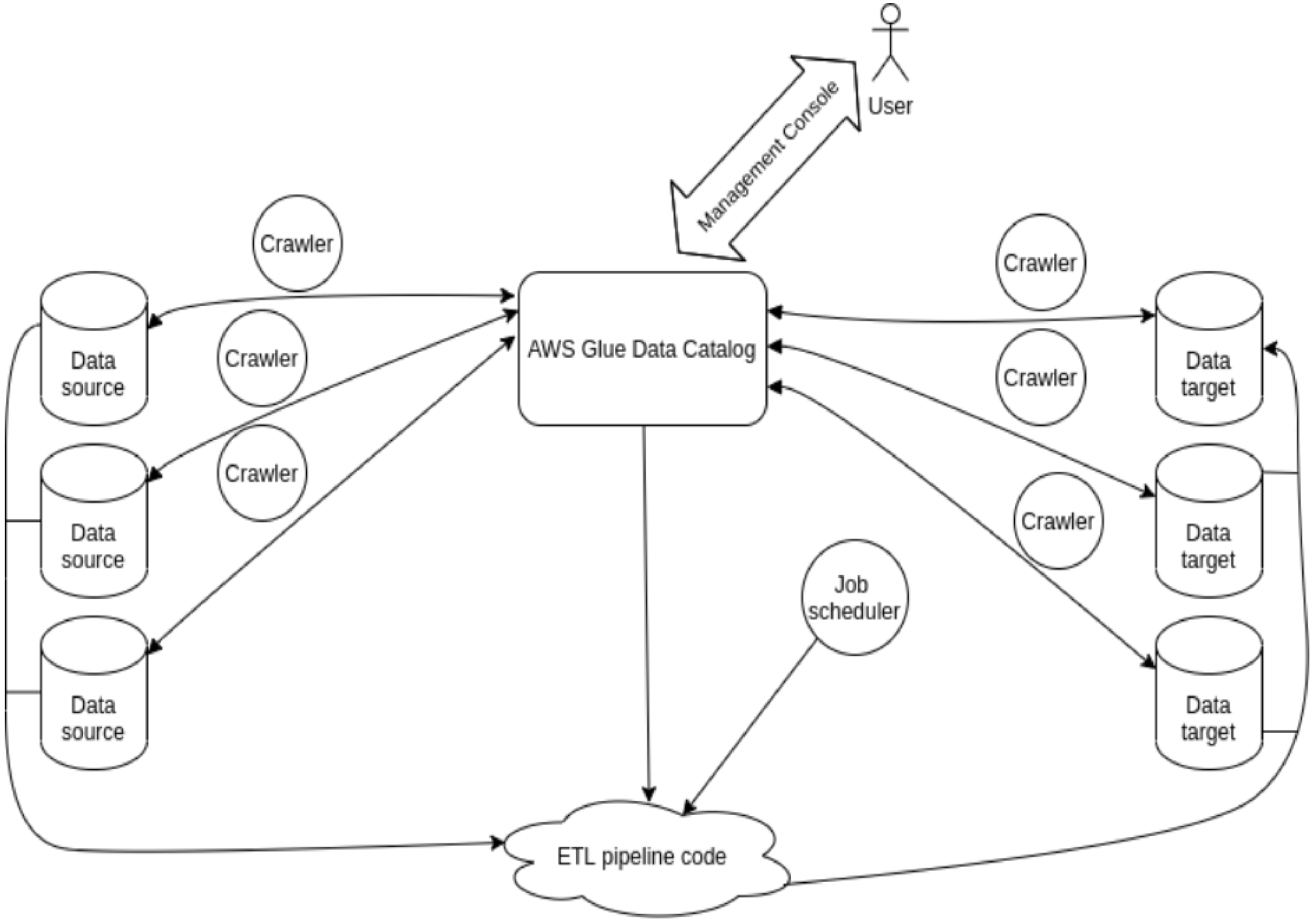

Features of AWS Glue
Job Scheduler
1
Developer end points
2
Automatic code generation
3
Integrated data catalog
4
Automatic schema discovery
5

AWS Glue Components




Schedule or Event
Triggers
Data Sources
Amazon S3
Amazon RDS
JDBC
Amazon Redshift
Extracts
AWS Glue Job
Transforms
Runs script
Loads
Data Targets
Amazon S3
Amazon RDS
JDBC
Amazon Redshift
Glue crawler
Glue Data Catalog


Amazon Redshift
Amazon S3
Amazon RDS
Databases running on Amazon EC2
Discover your data and store associated metadata in the Glue Data Catalog
AWS Glue Crawlers
Crawlers scan your data sets and populate your Glue Data Catalog
AWS Glue Data Catalog
The Glue Data Catalog serves as a central metadata repository
Amazon Athena
Amazon Redshift
Amazon EMR
Once catalogued in Glue, your data is immediately available for analysis
Amazon QuickSight & Other BI Tools
Run reports
Load data to target store
AWS Glue ETL
Read and write metadata to the Glue Data Catalog
Amazon CloudWatch
Logs and notifications are pushed to CloudWatch


User
Management Console
Crawler
Crawler
Crawler
Crawler
Crawler
Crawler
AWS Glue Data Catalog
Data source
Data source
Data source
Data target
Data target
Data target
Job scheduler
ETL pipeline code

Hands-on: Glue

Glue

- Download the dataset
- Upload the file to an S3 bucket
- Open the ‘Glue’ service on AWS
- Click on ‘Add Tables Using a Crawler’
- Enter any crawler name and click on ‘Next’
- Select ‘Data stores’ and click on ‘Next’
- Specify the path of the .csv file on the S3 bucket as the source and click on ‘Next’
- Mention any IAM role and click on ‘Next’
- Run the crawler

Glue (Contd.)

- Wait for the crawler to finish executing
- Upload the file to the S3 bucket
- Check the output
- Go to Jobs and click on ‘Add job’
- Use the following settings and the IAM role that was created earlier
- Select the data source created through the crawler and click on ‘Next’
- Select ‘Change schema’ and choose the dataset and a data target
- Save the job and edit the script
- Run the job

+919030485102

rganesh0203@gmail.com

https://topmate.io/rganesh_0203