Analytics

Kinesis, Redshift, ElasticSearch, Data Pipeline, EMR

AWS Certified Solutions Architect– Professional (SAP-C01) - Study notes - Sep'2019

Kinesis

Use cases for Kinesis

https://aws..com/kinesis/data-streams/faqs/

Typical scenarios:

Accelerated log and data feed intake: For example, system and application logs can be continuously added to a data stream and be available for processing within seconds.

Real-time metrics and reporting:

Real-time data analytics: example: clickstreams analytics

Complex stream processing: multiple streams

Q: How does Kinesis Data Streams differ from SQS?

Kinesis enables **real-time processing of streaming big data**. It provides **ordering of records**, as well as the **ability to read and/or replay records in the same order** to multiple Kinesis Applications. **easier to build multiple applications reading from the same Kinesis data stream** (for example, to perform counting, aggregation, and filtering).

SQS lets you easily move data between distributed application components and helps you build applications in which messages are processed independently (with message-level ack/fail semantics), such as automated workflows.

Q: When should I use Kinesis Data Streams, and when should I use SQS?

Use Kinesis for:

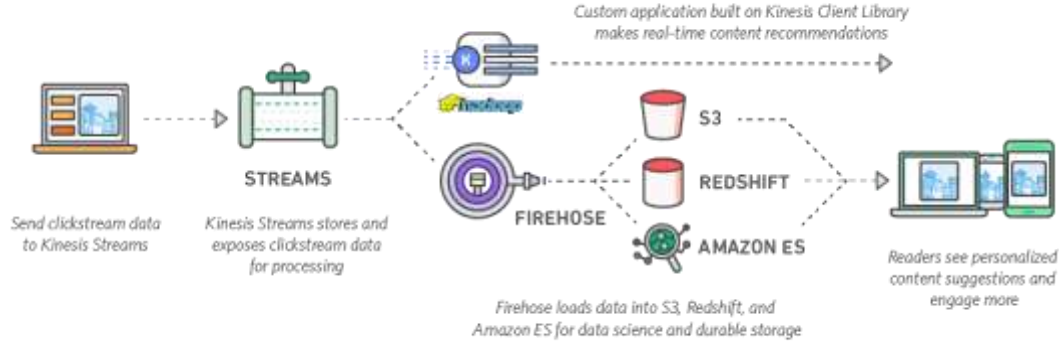
- Routing related records to the same record processor (as in streaming MapReduce).
- You want to maintain Ordering of records.
- Ability for multiple applications to consume the same stream concurrently.
- Ability to consume records in the same order a few hours later.

Use SQS for:

- Messaging semantics (such as message-level ack/fail) and visibility timeout.
- Individual message delay.
- Dynamically increasing concurrency/throughput at read time
- Leveraging SQS's ability to scale transparently.

Streaming Data Solutions on AWS with Kinesis - Kinesis Firehose

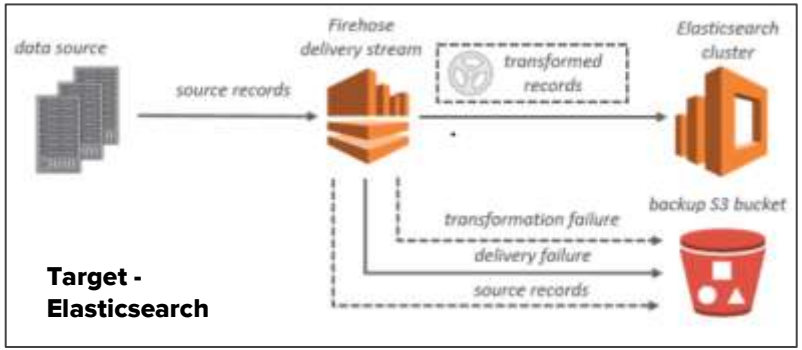
https://d0.awsstatic.com/whitepapers/whitepaper-streaming-data-solutions-on-aws-with--kinesis.pdf

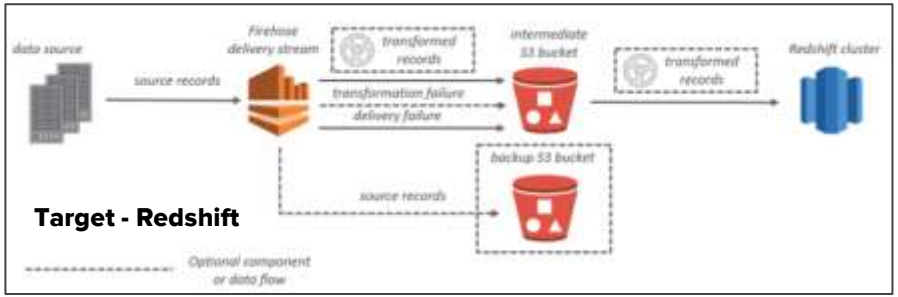


Firehose

Data injestion: Using API, Kinesis Java Agent

Lambda data transformation





Streaming Data Solutions on AWS with Kinesis - Kinesis Streams

https://d0.awsstatic.com/whitepapers/whitepaper-streaming-data-solutions-on-aws-with--kinesis.pdf

*A common use for **Kinesis Streams** is the real-time aggregation or analysis of data followed by loading the aggregate data into a data warehouse or map-reduce cluster. 3 ways to injest data:*

- *maintains MESSAGE ORDER*
- *IS DURABLE (7 days retention post which send to S3 via Firehose) & SCALABLE*
- *How many simultaneous consumers ? Kinesis read fan-out??*

Data Injestion:

1. *Kinesis Agent - the agent can send data to Firehose or to Streams*
2. *Kinesis Producer Library (KPL) - helps in aggregation, retries, integrates with KCL, publish CW metrics. Prefer async mode*

   If your application does not log records to a local file, and it creates a large number of small records per second, consider using the KPL.

1. *AWS SDK Apis - Use this if you cannot use Agent or KPL*

Processing data in Kinesis Streams:

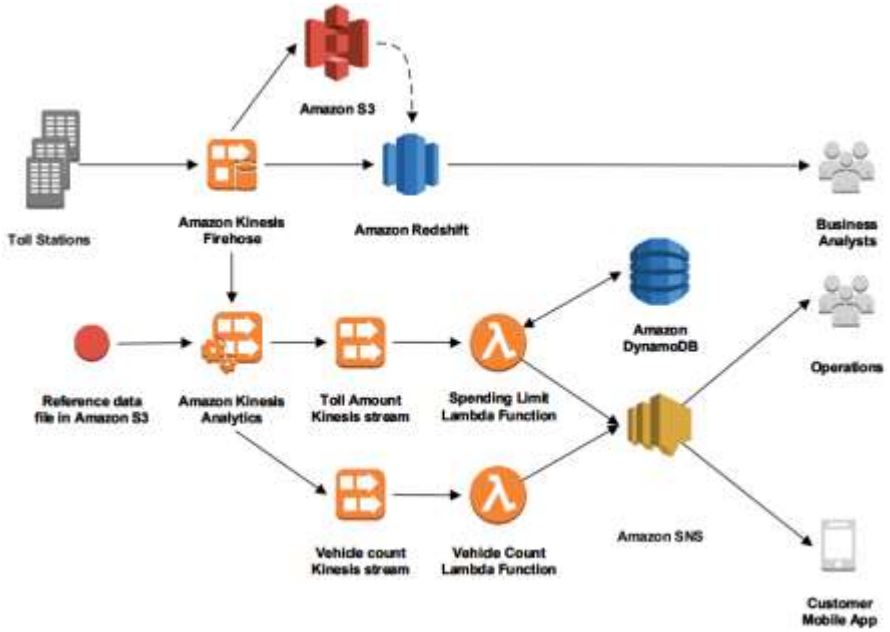
1. *Kinesis Analytics* - described in last slide
2. *Kinesis Client Library (KCL)* - helps in distributed computing issues such as load balancing across multiple instances, responding to instance failures, checkpointing processed records, and reacting to resharding. The KCL enables you to focus on writing record-processing logic.
3. *AWS lambda* - You can subscribe Lambda functions to automatically read batches of records off your Kinesis stream and process them if records are detected on the stream. AWS Lambda then polls the stream periodically (once per second) for new records. When it detects new records, it invokes your Lambda function by passing the new records as a parameter. If no new records are detected, your Lambda function is not invoked.
4. *API* - write your own consumer application from scratch, this is PULL model

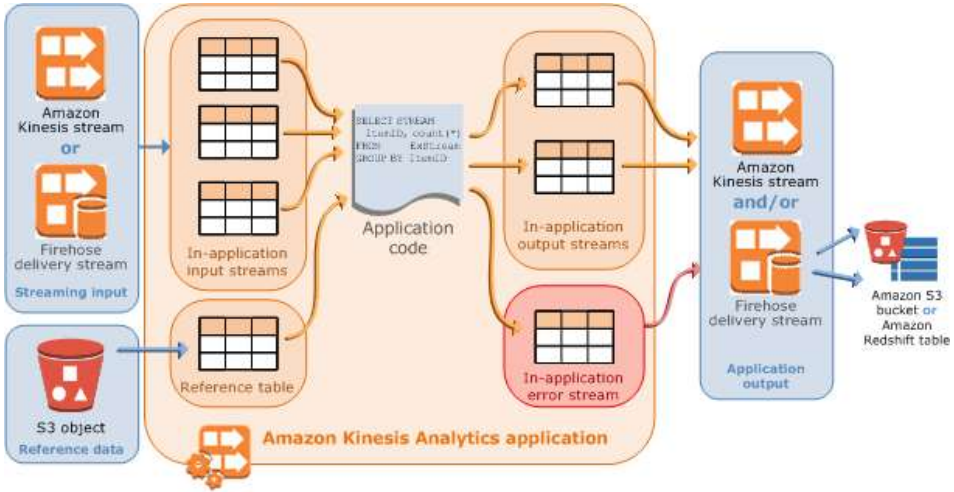
Start with lambda -> KCL -> APIs

Streaming Data Solutions on AWS with Kinesis - Kinesis Analytics

https://d0.awsstatic.com/whitepapers/whitepaper-streaming-data-solutions-on-aws-with--kinesis.pdf

In addition to S3, ES and RS, Firehose can stream data to ***Kinesis Analytics***. With Kinesis Analytics, you can process and analyze streaming data using SQL. The service enables you to quickly author and run powerful SQL code against streaming sources to perform time series analytics, feed real-time dashboards, and create real-time metrics.





Data Pipeline

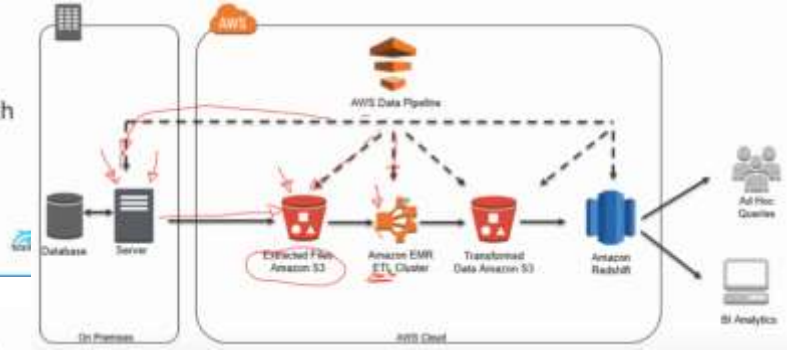
Data Pipeline use cases

- **Move to Cloud**
  - Easily copy data from your on-premises data store, like a MySQL database, and move it to an AWS data store, like S3 to make it available to a variety of AWS services such as Amazon EMR, Amazon Redshift, and Amazon RDS.

- **ETL Data to Amazon Redshift**
  - Copy RDS or DynamoDB tables to S3, transform data structure, run analytics using SQL queries and load it to Redshift.

- **ETL Unstructured Data**
  - Analyze unstructured data like clickstream logs using Hive or Pig on EMR, combine it with structured data from RDS and upload it to Redshift for easy querying.

- **Data Loads and Extracts**
  - Copy data from your RDS or Redshift table to S3 and vice-versa.



- **Load AWS Log Data to Amazon Redshift**
  - Load log files such as from the AWS billing logs, or AWS CloudTrail, Amazon CloudFront, and Amazon CloudWatch logs, from Amazon S3 to Redshift.

- **Amazon DynamoDB Backup and Recovery**
  - Periodically backup your Dynamo DB table to S3 for disaster recovery purposes.

- **Amazon DynamoDB data copy to another Region**
  - You can use Data Pipeline to schedule a cross region copy of your DynamoDB table, and you can use time stamp attributes so you can only copy the data that has changed since last copy process

Glue

Amazon Glue

- AWS Glue is a fully-managed, **serverless, pay-as-you-go**, extract, transform, and load (ETL) service that automates the time-consuming steps of data preparation for analytics.
  - It makes it simple and cost-effective to categorize the data, clean it, enrich it, and move it reliably between various data stores.
  - It protects the data in transit and at rest.
  - AWS Glue offers a persistent metadata store for your data in Amazon S3

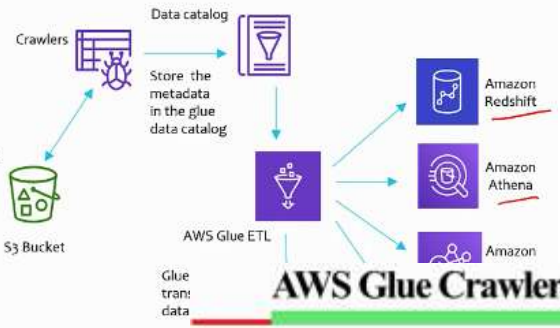
- AWS Glue consists of:
  - **Automatic crawling** to populate a central metadata repository known as the AWS Glue Data Catalog,
    - For a given data set, what can be stored includes, the data table definition (schema), physical location, add business relevant attributes, as well as track how this data has changed over time.
  - **Job Authoring,** an ETL engine that automatically recommends and generates Python or Scala code, to transform the source data into target schemas.
    - It runs the ETL jobs on a fully managed, scale-out Apache Spark environment to load the data into its destination.
    - It also allows for the setup, orchestration, and monitoring of complex data flows.
  - **Job execution,** a flexible scheduler that handles dependency resolution, job monitoring, and retries.

6 people bookmarked this moment.

- AWS Glue uses other AWS services to orchestrate your ETL (extract, transform, and load) jobs to build a data warehouse.

Glue Catalog

- The AWS Glue Data Catalog is a central repository to store structural and operational metadata for all the relevant data assets.
  - AWS Glue automatically discovers and profiles the data via the Glue Data Catalog,
- The AWS Glue Data Catalog is Apache Hive Metastore compatible
  - It can be used as a drop-in replacement for the Apache Hive Metastore for Big Data applications running on EMR.
- The AWS Glue Data Catalog also provides out-of-box integration with **Amazon Athena, EMR, and Redshift Spectrum**.
  - With AWS Glue (Data Catalog), access and analysis of the data happens through one unified interface without having to load it into multiple data silos.
  - Once the table definitions (schemas) are added to the Glue Data Catalog, they are available for ETL.
    - Also the schemas will be readily available for querying in Amazon Athena, EMR, and Redshift Spectrum.
      - ➢ This is how data catalog allows for a common view of the data between these services.



AWS Glue Crawlers

- An **AWS Glue crawler** is a data crawling/cataloging program that can automatically scan your data sources, identify data formats, and derive the **schema**.
  - An AWS Glue crawler connects to a data store, progresses through a prioritized list of classifiers to extract the schema of your data and other statistics, and then populates the Glue Data Catalog with this metadata.
- Crawlers can run periodically to detect the availability of new data as well as changes to existing data, including table definition changes.
- Crawlers automatically add new tables, new partitions to existing table, and new versions of table definitions.
  - Glue crawlers can be customized to classify your own file types.

Glue Catalog Use cases

- **Use AWS Glue to run serverless queries against an Amazon S3 data lake.**
  - AWS Glue can catalog an Amazon Simple Storage Service (Amazon S3) stored data, making it available for querying with **Amazon Athena** and **Amazon Redshift Spectrum.**
  - With crawlers, the metadata stays in sync with the underlying data.
    - Athena and Redshift Spectrum can directly query your Amazon S3 data lake using the AWS Glue Data Catalog.
  - With AWS Glue, you access and analyze data through one unified interface without loading it into multiple data silos.

- **Use AWS Glue to build a Data Warehouse to organize, cleanse, validate, and format data.**
  - You can transform and move AWS Cloud data into your data store.
  - You can also load data from disparate sources into your data warehouse for regular reporting and analysis.
    - By storing it in a data warehouse, you integrate information from different parts of your business and provide a common source of data for analytics and decision making.

- **Create event-driven ETL pipelines with AWS Glue.**
  - You can run your ETL jobs as soon as new data becomes available in Amazon S3 by invoking your AWS Glue ETL jobs from an AWS Lambda function.
  - You can also register this new dataset in the AWS Glue Data Catalog as part of your ETL jobs.

- **Use AWS Glue to understand your data assets.**
  - Use AWS Glue to discover properties of the data you own, transform it, and prepare it for analytics.
  - **Data Sources:** Glue can automatically discover both structured and semi-structured data stored in your data lake on Amazon S3, Data Warehouse in Amazon Redshift, and various DBs on AWS.
  - **Data Targets:** It provides a unified view of your data via the Glue Data Catalog that is available for ETL, querying and reporting using services like Amazon Athena, EMR, and Redshift Spectrum.
  - Glue automatically generates Scala or Python code for your ETL jobs that you can further customize using tools you are already familiar with.

Glue vs Data Pipeline

- AWS Glue provides a managed ETL service that runs on a serverless Apache Spark environment.
- It serverless hence, you do not have access to the underlying services or infrastructure used to do the Glue tasks.

- Use Data Pipeline (with EMR) when:
  - A managed orchestration service that gives you greater flexibility in terms of the execution environment, access and control over the compute resources that run your code, as well as the code itself that does data processing.
  - AWS Data Pipeline launches resources in your account, hence, you do have low level access to EC2 instances or EMR clusters.
  - If your use case requires you to use an engine other than Apache Spark (with Python or Scala)
  - If you want to run a heterogeneous set of jobs that run on a variety of engines like Hive, Pig, etc., then AWS Data Pipeline would be a better choice.

Redshift

AMZ Big data technologies

	Use cases	Anti-patterns
Kinesis (Streams)	Real-time data analytics, Log and data feed intake and processing, Real-time metrics and reporting	Small scale consistent throughput Long-term data storage and analytics
EMR (Hadoop)	Log processing and analytics • Large extract, transform, and load (ETL) data movement • Risk modeling and threat analytics • Ad targeting and click stream analytics • Genomics • Predictive analytics • Ad hoc data mining and analytics	Small data sets, ACID transaction requirements
AWS Glue (ETL)	Automatically crawl your data and generate code to execute or data transformations and loading processes. • Integration with services like Athena, EMR, and Redshift • Serverless, no infrastructure to provision or manage • AWS Glue uses familiar technology – Python and Spark.	Streaming data, Multiple ETL engines, NoSQL Databases
ML	Enable applications to flag suspicious transactions, Forecast product demand, Personalize application content, Predict user activity, Listen to social media	Very large data sets (limited to 100 GB). Alternative is Spark's Machine Learning Library (MLlib), Unsupported learning tasks
Redshift	Analyze global sales data for multiple products • Store historical stock trade data • Analyze ad impressions and clicks • Aggregate gaming data • Analyze social trends • Measure clinical quality, operation efficiency, and financial performance in health care	Small data sets, OLTP, Unstructured data, BLOB data
ElasticSearch	Analyze activity logs, e.g., logs for customer facing applications or websites • Analyze CloudWatch logs • Analyze product usage data • Analyze social media sentiments, CRM data and find trends for your brand and products • Analyze data stream updates from other AWS services, e.g., Kinesis and DynamoDB • Provide customers a rich search and navigation experience. • Usage monitoring for mobile applications	OLTP, Ad hoc data querying - alternative is using Athena
Athena	Interactive ad hoc querying for web logs, To query staging data before loading into Redshift, Send AWS Service logs to S3 for Analysis with Athena, Building Interactive Analytical Solutions with notebook-based solutions, e.g., RStudio, Jupyter, or Zeppelin	Enterprise Reporting and Business Intelligence Workloads (alternative Redshift), ETL Workloads, RDBMS

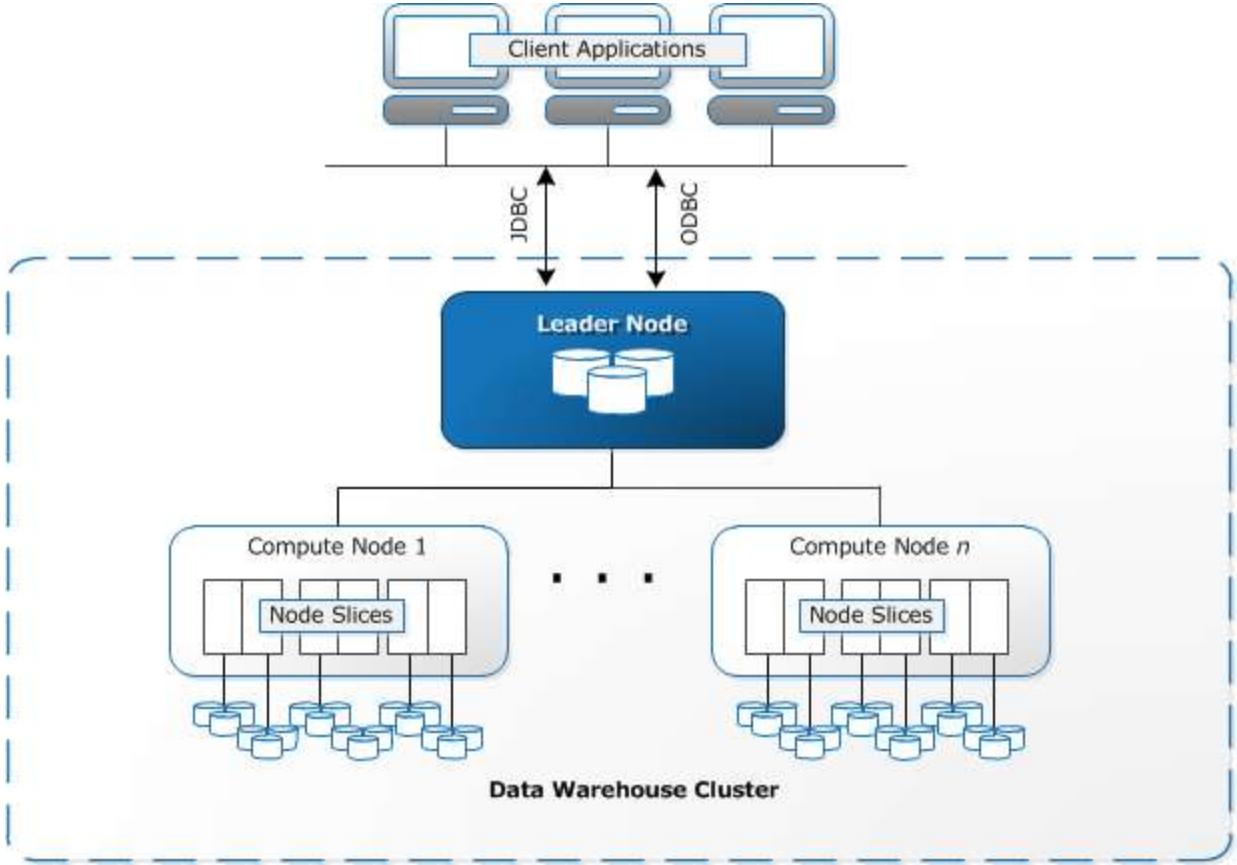
AMZ Big data technologies

Data Movement	Data Lake	Analytics	Machine Learning
On-premises data movement: **DirectConnect, Snowmobile, Snowball, Storage Gateway** Real-time data movement: **Amazon Kinesis Data Firehose**, **Amazon Kinesis Video Streams**, and **AWS IoT Core**.	Object Storage: **S3** Backup and Archive: **Glacier** Data Catalog: **Glue**	Interactive Analytics: **Athena** Big Data Processing **Amazon EMR** Data Warehousing **Amazon Redshift** Real-Time Analytics **Amazon Kinesis** Operational Analytics **Amazon Elasticsearch Service** Dashboards and Visualizations **Amazon QuickSight**	Frameworks and Interfaces **AWS Deep Learning AMIs** Platform Services **Amazon SageMaker** Application Services

Redshift - Introduction

An Redshift data warehouse is an enterprise-class relational database query and management system. When you execute analytic queries, you are retrieving, comparing, and evaluating large amounts of data in multiple-stage operations to produce a final result. Redshift achieves efficient storage and optimum query performance through a combination of massively parallel processing, columnar data storage, and very efficient, targeted data compression encoding schemes. It is based on PostgreSQL

- The core infrastructure component of an Redshift data warehouse is a **cluster**.
- A cluster is composed of one or more compute nodes. If a cluster is provisioned with two or more compute nodes, an additional leader node coordinates the compute nodes and handles external communication.
- The **leader node** manages communications with client programs and all communication with **compute nodes**.
- The **leader node compiles code for individual elements** of the execution plan and assigns the code to individual compute nodes. The **compute nodes execute the compiled code and send intermediate results back to the leader node** for final aggregation.
- **Node slices** - A compute node is partitioned into slices. Each slice is allocated a portion of the node's memory and disk space, where it processes a portion of the workload assigned to the node. The leader node manages distributing data to the slices and apportions the workload for any queries or other database operations to the slices. The slices then work in parallel to complete the operation.
- **Internal network -** Redshift takes advantage of high-bandwidth connections, close proximity, and custom communication protocols to provide private, very high-speed network communication between the leader node and compute nodes. The compute nodes run on a separate, isolated network that client applications never access directly.
- **Databases -** A cluster contains one or more databases. User data is stored on the compute nodes. As compared to OLTP, Redshift is optimized for high-performance analysis and reporting of very large datasets.


Client Applications
JDBC
ODBC
Leader Node
Compute Node 1
Node Slices
Compute Node n
Node Slices
Data Warehouse Cluster

Redshift - Performance characteristics

➔Massively Parallel Processing:

Multiple compute nodes handle all query processing leading up to final result aggregation, with each core of each node executing the same compiled query segments on portions of the entire data. By selecting an appropriate distribution key for each table, you can optimize the distribution of data to balance the workload and minimize movement of data from node to node.

➔Columnar Data Storage

Storing database table information in a columnar fashion reduces the number of disk I/O requests and reduces the amount of data you need to load from disk. Loading less data into memory enables Redshift to perform more in-memory processing when executing queries.
When columns are sorted appropriately, the query processor is able to rapidly filter out a large subset of data blocks.

Choose the Best Sort Key:
Redshift stores your data on disk in sorted order according to the sort key. The Redshift query optimizer uses sort order when it determines optimal query plans.

- If recent data is queried most frequently, specify the timestamp column as the leading column for the sort key.
- If you do frequent range filtering or equality filtering on one column, specify that column as the sort key.
- If you frequently join a table, specify the join column as both the sort key and the distribution key. Doing this enables the query optimizer to choose a sort merge join instead of a slower hash join. Because the data is already sorted on the join key, the query optimizer can bypass the sort phase of the sort merge join.

Redshift - Performance characteristics

➔Data Compression

Data compression reduces storage requirements, thereby reducing disk I/O, which improves query performance.

➔Query Optimizer

The Redshift query optimizer implements significant enhancements and extensions for processing complex analytic queries that often include multi-table joins, subqueries, and aggregation.

➔Result Caching

To reduce query execution time and improve system performance, Redshift caches the results of certain types of queries in memory on the leader node. If a match is found in the result cache, Redshift uses the cached results and doesn't execute the query. Result caching is transparent to the user and is is enabled by default. Redshift uses cached results for a new query when all of the following are true:

- The user submitting the query has access privilege to the objects used in the query.
- The table or views in the query haven't been modified.
- The query doesn't use a function that must be evaluated each time it's run, such as GETDATE.
- The query doesn't reference Redshift Spectrum external tables.
- Configuration parameters that might affect query results are unchanged.
- The query syntactically matches the cached query.

➔Compiled Code

The leader node distributes fully optimized compiled code across all of the nodes of a cluster. Compiling the query eliminates the overhead associated with an interpreter and therefore increases the execution speed, especially for complex queries. The compiled code is cached and shared across sessions on the same cluster, so subsequent executions of the same query will be faster, often even with different parameters.

Using Redshift with other AWS services

- **Moving Data Between Redshift and S3:**
  Simple Storage Service (S3) is a web service that stores data in the cloud. Redshift leverages parallel processing to read and load data from multiple data files stored in S3 buckets. You can also use parallel processing to export data from your Redshift data warehouse to multiple data files on S3.

- **Using Redshift with DynamoDB:**
  DynamoDB is a fully managed NoSQL database service. You can use the COPY command to load an Redshift table with data from a single DynamoDB table.

- **Importing Data from Remote Hosts over SSH:**
  You can use the COPY command in Redshift to load data from one or more remote hosts, such as EMR clusters, EC2 instances, or other computers. COPY connects to the remote hosts using SSH and executes commands on the remote hosts to generate data. Redshift supports multiple simultaneous connections. The COPY command reads and loads the output from multiple host sources in parallel.

- **Automating Data Loads Using AWS Data Pipeline:**
  You can use AWS Data Pipeline to automate data movement and transformation into and out of Redshift. By using the built-in scheduling capabilities of AWS Data Pipeline, you can schedule and execute recurring jobs without having to write your own complex data transfer or transformation logic. For example, you can set up a recurring job to automatically copy data from DynamoDB into Redshift.

- **Migrating Data Using AWS Database Migration Service (AWS DMS):**
  You can migrate data to Redshift using AWS Database Migration Service. AWS DMS can migrate your data to and from most widely used commercial and open-source databases such as Oracle, PostgreSQL, Microsoft SQL Server, Redshift, Aurora, DynamoDB, S3, MariaDB and MySQL

Redshift - Workload management (WLM)

You can use WLM to define multiple query queues and to route queries to the appropriate queues at run time. For example, suppose that one group of users submits occasional complex, long-running queries that select and sort rows from several large tables. Another group frequently submits short queries that select only a few rows from one or two tables and run in a few seconds. In this situation, the short-running queries might have to wait in a queue for a long-running query to complete.

With Concurrency Scaling, you can support virtually unlimited concurrent users and concurrent queries, with consistently fast query performance.

Automatic WLM -
You can enable Redshift to manage how resources are divided to run concurrent queries with automatic WLM. Automatic WLM manages the resources required to run queries. Redshift determines how many concurrent queries and how much memory is allocated to each dispatched query.

Manual WLM -
Alternatively, you can manage system performance and your users' experience by modifying your WLM configuration to create separate queues for the long-running queries and the short-running queries. At run time, you can route queries to these queues according to user groups or query groups.

Configuration: WLM is part of ***parameter group configuration***. A cluster uses the WLM configuration that is specified in its associated parameter group. When you create a parameter group, the default WLM configuration contains one queue that can run up to five queries concurrently. You can add additional queues and configure WLM properties in each of them if you want more control over query processing. Each queue that you add has the same default WLM configuration until you configure its properties.

Redshift - Best practices for High-Performance ETL Processing (1/2)

https://aws..com/blogs/big-data/top-8-best-practices-for-high-performance-etl-processing-using--redshift/

An ETL (Extract, Transform, Load) process enables you to load data from source systems into your data warehouse. This is typically executed as a batch or near-real-time ingest process to keep the data warehouse current and provide up-to-date analytical data to end users.

1.COPY data from multiple, evenly sized files.

When you load data into Redshift, you should aim to have each slice do an equal amount of work. When you load the data from a single large file or multiple uneven sizes, some slices do more work than others. As a result, the process runs only as fast as the slowest, or most heavily loaded, slice.

1.Use workload management to improve ETL runtimes.

Use Redshift's workload management (WLM) to define multiple queues dedicated to different workloads (for example, ETL versus reporting) and to manage the runtimes of queries.

1.Perform table maintenance regularly.

To get the best performance from your Redshift database, you must ensure that database tables regularly are VACUUMed and ANALYZEd.

1.Perform multiple steps in a single transaction.

ETL transformation logic often spans multiple steps. Because commits in Redshift are expensive, if each ETL step performs a commit, multiple concurrent ETL processes can take a long time to execute.

Use Time-Series Tables

If your data has a fixed retention period, you can organize your data as a sequence of time-series tables. In such a sequence, each table is identical but contains data for different time ranges.

Use a Staging Table to Perform a Merge (Upsert):

Redshift doesn't support a single merge statement (update or insert, also known as an upsert) to insert and update data from a single data source. To do a merge, load your data into a staging table and then join the staging table with your target table for an UPDATE statement and an INSERT statement.

Redshift - Best practices for High-Performance ETL Processing (2/2)

https://aws..com/blogs/big-data/top-8-best-practices-for-high-performance-etl-processing-using--redshift/

5. Loading data in bulk.

- Using S3 you can stage and accumulate data from multiple source systems before executing a bulk COPY operation.
- Use a manifest file to ingest large datasets that span multiple files.
- Use temporary staging tables to hold the data for transformation.
- User ALTER table APPEND to swap data from the staging tables to the target table.

6. Use UNLOAD to extract large result sets.

Fetching a large number of rows using SELECT is expensive and takes a long time. Use UNLOAD to extract large results sets directly to S3. After it's in S3, the data can be shared with multiple downstream systems. By default, UNLOAD writes data in parallel to multiple files according to the number of slices in the cluster. All the compute nodes participate to quickly offload the data into S3.

7. Use Redshift Spectrum for ad hoc ETL processing.

Events such as data backfill, promotional activity, and special calendar days can trigger additional data volumes that affect the data refresh times in your Redshift cluster. To help address these spikes in data volumes and throughput, I recommend staging data in S3. After data is organized in S3, Redshift Spectrum enables you to query it directly using standard SQL. In this way, you gain the benefits of additional capacity without having to resize your cluster.

8. Monitor daily ETL health using diagnostic queries.

9. Load Data in Sort Key Order:

Load your data in sort key order to avoid needing to vacuum. If each batch of new data follows the existing rows in your table, your data is properly stored in sort order, and you don't need to run a vacuum. You don't need to presort the rows in each load because COPY sorts each batch of incoming data as it loads.

Redshift Advisor Recommendations

To help you improve the performance and decrease the operating costs for your Amazon Redshift cluster, Amazon Redshift Advisor offers you specific recommendations about changes to make. Advisor develops its customized recommendations by analyzing performance and usage metrics for your cluster. These tailored recommendations relate to operations and cluster settings. To help you prioritize your optimizations, Advisor ranks recommendations by order of impact.

Recommendations:

1. Compress Table Data
2. Compress Amazon S3 File Objects Loaded by COPY
3. Isolate Multiple Active Databases
4. Reallocate Workload Management (WLM) Memory
5. Skip Compression Analysis During COPY
6. Split Amazon S3 Objects Loaded by COPY
7. Update Table Statistics
8. Enable Short Query Acceleration
9. Replace Single-Column Interleaved Sort Keys
10. Alter Distribution Keys on Tables

Redshift backup

Amazon Redshift takes automatic, incremental snapshots of your data periodically and saves them to Amazon S3. Additionally, you can take manual snapshots of your data whenever you want.

Configuring Cross-Region Snapshot Copy for a Non-Encrypted Cluster

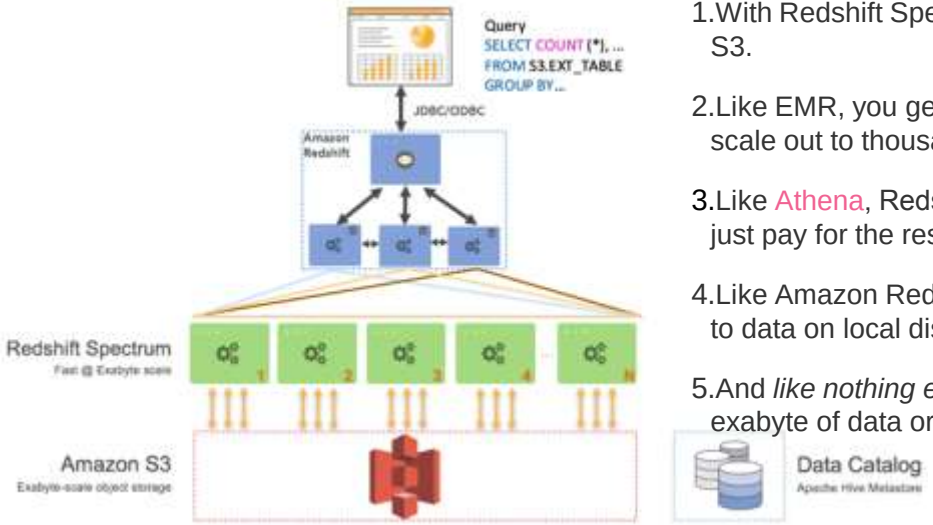
You can configure Amazon Redshift to copy snapshots for a cluster to another region. To configure cross-region snapshot copy, you need to enable this copy feature for each cluster and configure where to copy snapshots and how long to keep copied automated snapshots in the destination region. When cross-region copy is enabled for a cluster, all new manual and automatic snapshots are copied to the specified region. Copied snapshot names are prefixed with **copy**:.

Configure Cross-Region Snapshot Copy for an AWS KMS-Encrypted Cluster

When you launch an Amazon Redshift cluster, you can choose to encrypt it with a master key from the AWS Key Management Service (AWS KMS). AWS KMS keys are specific to a region. If you want to enable cross-region snapshot copy for an AWS KMS-encrypted cluster, you must configure a *snapshot copy grant* for a master key in the destination region so that Amazon Redshift can perform encryption operations in the destination region.

Redshift Spectrum

https://aws.amazon.com/blogs/big-data/amazon-redshift-spectrum-extends-data-warehousing-out-to-exabytes-no-loading-required/



1. With Redshift Spectrum, Amazon Redshift customers can easily query their data in Amazon S3.

2. Like EMR, you get the benefits of open data formats and inexpensive storage, and you can scale out to thousands of nodes to pull data, filter, project, aggregate, group, and sort.

3. Like Athena, Redshift Spectrum is serverless and there's nothing to provision or manage. You just pay for the resources you consume for the duration of your Redshift Spectrum query.

4. Like Amazon Redshift itself, you get the benefits of a sophisticated query optimizer, fast access to data on local disks, and standard SQL.

5. And *like nothing else*, Redshift Spectrum can execute highly sophisticated queries against an exabyte of data or more—in just minutes.

Redshift Spectrum's architecture offers several advantages.

1. It elastically scales compute resources separately from the storage layer in Amazon S3.
2. It offers significantly higher concurrency because you can run multiple Redshift clusters and query the same data in S3
3. It leverages the Redshift query optimizer to generate efficient query plans, even for complex queries with multi-table joins and window functions.
4. It operates directly on your source data in its native format (Parquet, RCFile, CSV, TSV, Sequence, Avro, RegexSerDe and more to come soon). This means that no data loading or transformation is needed. This also eliminates data duplication and associated costs.
5. Operating on open data formats gives you the flexibility to leverage other AWS services and execution engines across your various teams to collaborate on the same data in Amazon S3. You get all of this, and because Redshift Spectrum is a feature of Amazon Redshift, you get the same level of end-to-end security, compliance, and certifications as with Amazon Redshift

Redshift Spectrum - Best practices

https://aws.amazon.com/blogs/big-data/10-best-practices-for-amazon-redshift-spectrum/

Best practices for concurrency

1. Use Redshift Spectrum to improve concurrent workloads
2. Use multiple on-demand Amazon Redshift clusters to scale concurrency

Best practices for storage

3. Consider columnar format for performance and cost
4. Partition files on frequently filtered columns

Best practices for cluster configuration

5. Optimize performance with the right Amazon Redshift cluster configuration

Best practices for when to use Redshift Spectrum

6. Achieve faster scan- and aggregation-intensive queries with Redshift Spectrum
7. Simplify ETL pipelines

Best practices for query performance

8. Improve Amazon S3 query performance with predicate pushdown
9. Replace complex DISTINCT operations with GROUP BY in your queries

Best practices for table placement and statistics

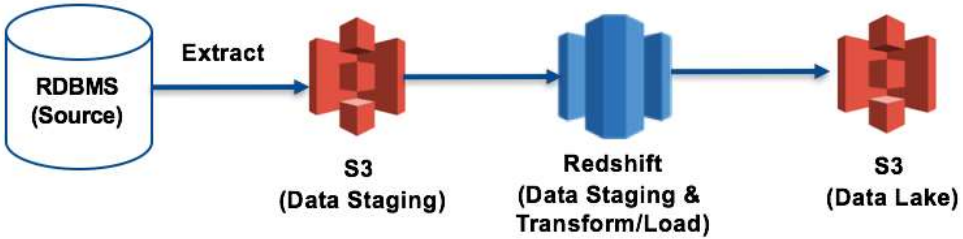
10. Determine the best place to store your tables
11. Set the table statistics (numRows) manually for S3 external tables

Best practices for query cost control

12. Pay attention to query cost and use query monitoring rules

Redshift - Example ETL process

https://aws..com/blogs/big-data/top-8-best-practices-for-high-performance-etl-processing-using--redshift/



Step 1: Extract from the RDBMS source to a S3 bucket

In this ETL process, the data extract job fetches change data every 1 hour and it is staged into multiple hourly files. Organizing the data into multiple, evenly sized files enables the COPY command to ingest this data using all available resources in the Redshift cluster. Further, the files are compressed (gzipped) to further reduce COPY times.

Step 2: Stage data to the Redshift table for cleansing

Ingesting the data can be accomplished using a JSON-based manifest file. Using the manifest file ensures that S3 eventual consistency issues can be eliminated and also provides an opportunity to dedupe any files if needed.

Step 3: Transform data to create daily, weekly, and monthly datasets and load into target tables

Data is staged in the "stage_tbl" from where it can be transformed into the daily, weekly, and monthly aggregates and loaded into target tables.

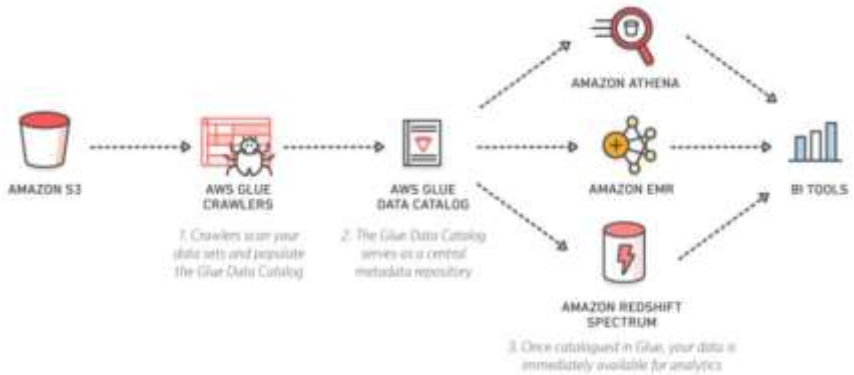
Step 4: Unload the daily dataset to populate the S3 data lake bucket

The transformed results are now unloaded into another S3 bucket, where they can be further processed and made available for end-user reporting using a number of different tools, including Redshift Spectrum and Athena.

Big data problem scenarios and AWS solution

https://d1.awsstatic.com/whitepapers/Big_Data_Analytics_Options_on_AWS.pdf?did=wp_card&trk=wp_card

Example 1: Queries against an S3 Data Lake



1. Crawlers can run periodically to detect the availability of new data as well as changes to existing data, including table definition changes. Crawlers automatically add new tables, new partitions to existing table, and new versions of table definitions.

2. For a given data set, you can store its table definition, physical location, add business relevant attributes, as well as track how this data has changed over time. The AWS Glue Data Catalog is Apache Hive Metastore compatible and is a drop-in replacement for the Apache Hive Metastore for Big Data applications running on EMR.

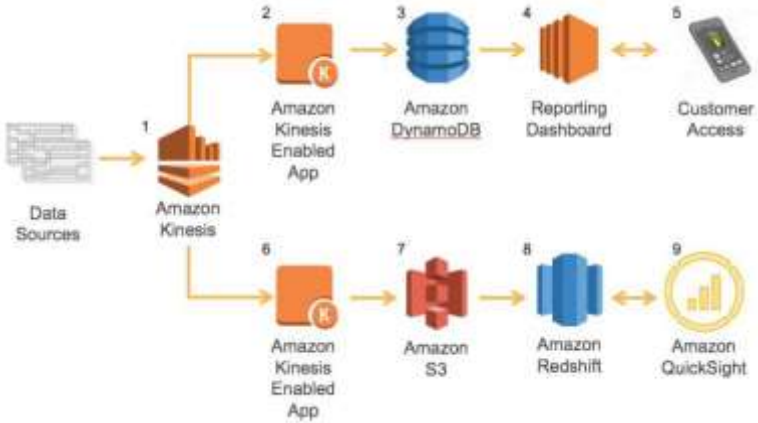
3. Once you add your table definitions to the AWS Glue Data Catalog, they are available for ETL and also readily available for querying in Athena, EMR, and Redshift Spectrum so that you can have a common view of your data between these services.

4. Using a BI tool like QuickSight enables you to easily build visualizations, perform ad-hoc analysis, and quickly get business insights from your data. QuickSight supports data sources like: Athena, Redshift Spectrum, S3 and many others

Big data problem scenarios and AWS solution

https://d1.awsstatic.com/whitepapers/Big_Data_Analytics_Options_on_AWS.pdf?did=wp_card&trk=wp_card

Example 2: Capturing and Analyzing Sensor Data



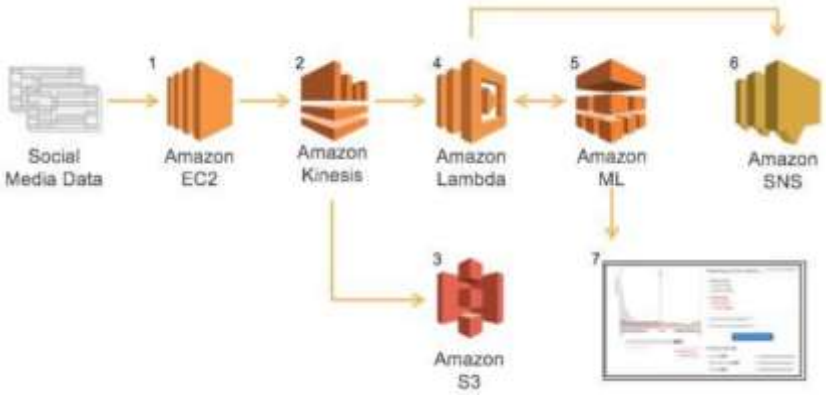
Break it up into two work streams, both originating from the same data:

- A/C unit's current information with near real-time requirements and a large number of customers consuming this information.
- All historical information on the A/C units to run trending and analytics for internal use

Big data problem scenarios and AWS solution

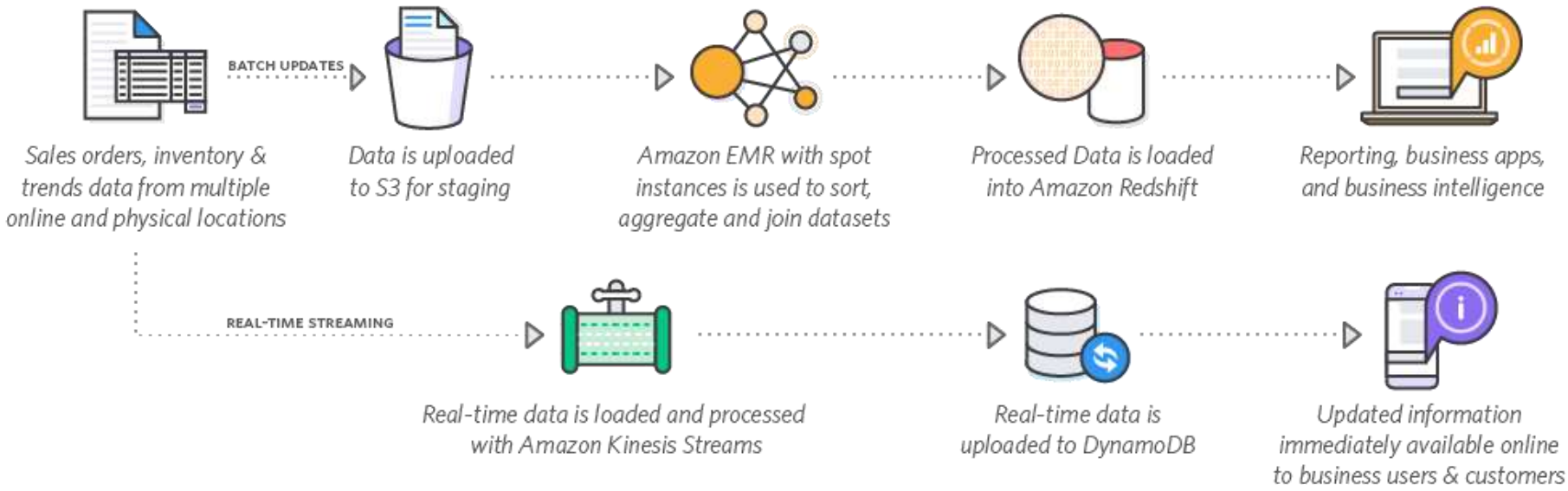
https://d1.awsstatic.com/whitepapers/Big_Data_Analytics_Options_on_AWS.pdf?did=wp_card&trk=wp_card

Example 3: Sentiment Analysis of Social Media



Big data problem scenarios and AWS solution

https://aws.amazon.com/big-data/use-cases/

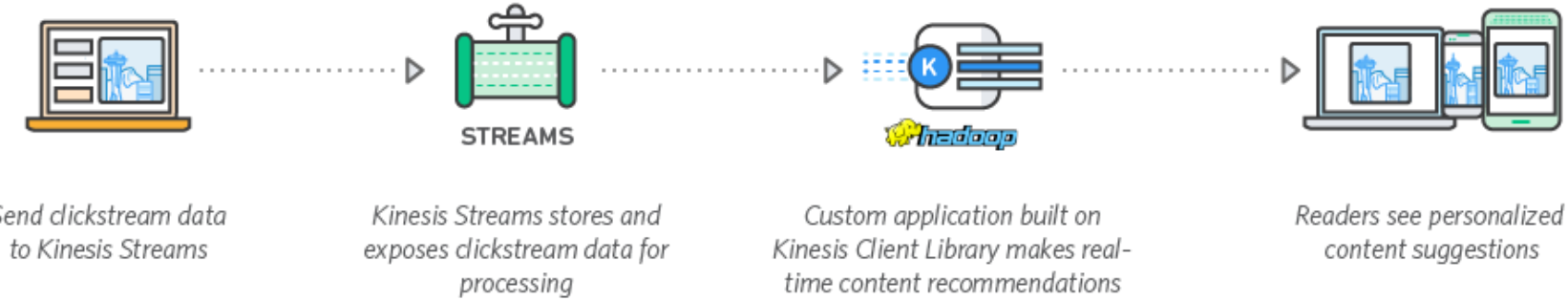


On-Demand Big Data Analytics

Redfin provides real estate listing & recommendations to millions of homebuyers. Every day, Redfin uses Amazon EMR with spot instances – dynamically spinning up & down Apache Hadoop clusters – to perform large data transformations and deliver data to internal and external customers.

Big data problem scenarios and AWS solution

https://aws.amazon.com/big-data/use-cases/

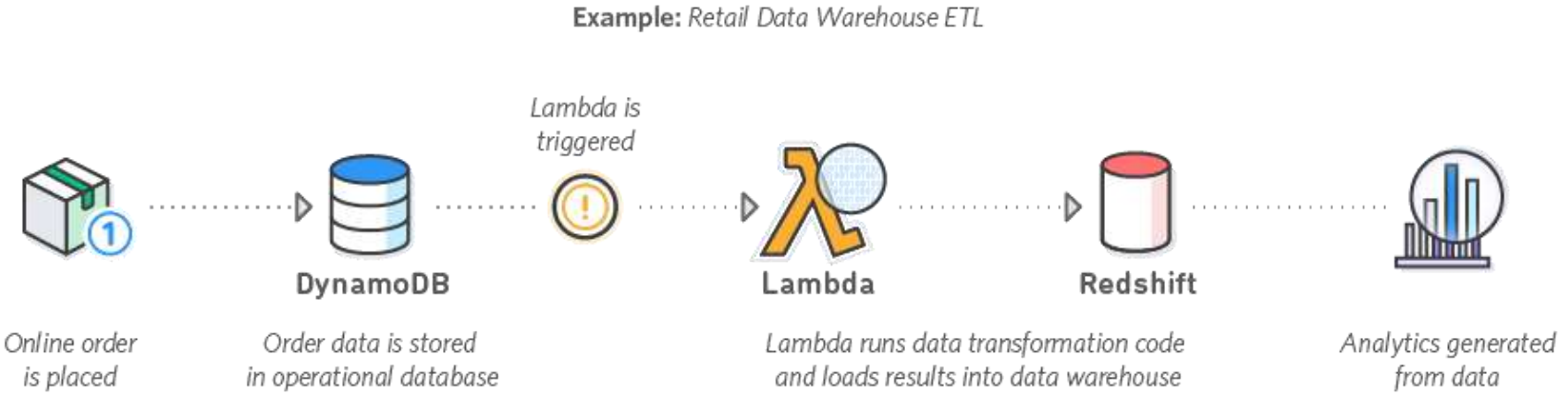


Clickstream Analysis

Improve your customer's digital experience and get a better understanding of your website. Collect, process, analyze, and visualize clickstream insights in real-time with AWS.
Hearst Corporation monitors trending content for over 250 digital properties worldwide and processes more than 30TB of data per day. Using an architecture that includes Amazon Kinesis and Spark running on Amazon EMR, Hearst corporation delivers real-time insights to data scientists and business stake-holders.

Big data problem scenarios and AWS solution

https://aws.amazon.com/big-data/use-cases/

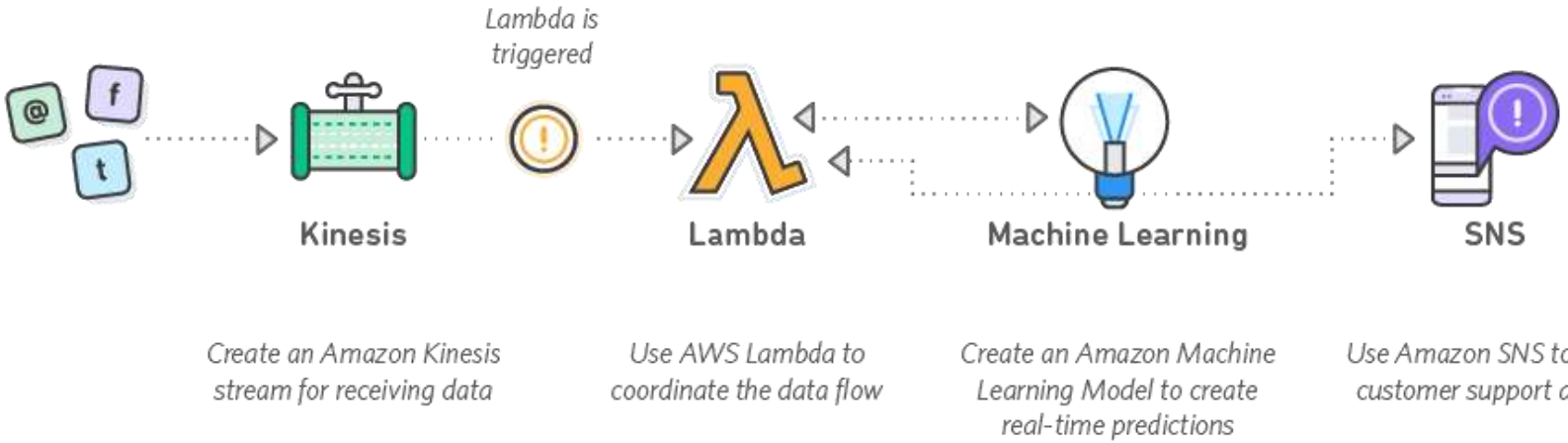


Event-driven Extract, Transform, Load (ETL)

Use AWS Lambda to perform data transformations - filter, sort, join, aggregate, and more - on new data, and load the transformed datasets into Amazon Redshift for interactive query and analysis. Zillow uses AWS Lambda and Amazon Kinesis to manage a global ingestion pipeline and produce quality analytics in real-time without building infrastructure.

Big data problem scenarios and AWS solution

https://aws.amazon.com/big-data/use-cases/

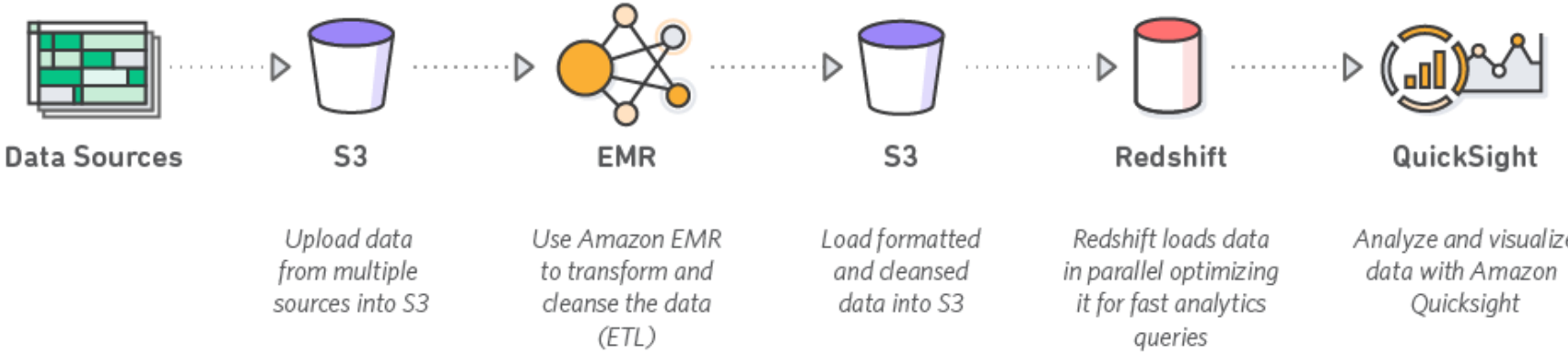


Smart Applications

Use Amazon Machine Learning to easily add predictive capabilities to your applications. Combine the power of Amazon Kinesis to ingest data from social media or other sources in real time and use Machine Learning to generate predictions on that data.

Big data problem scenarios and AWS solution

https://aws.amazon.com/big-data/use-cases/



Data Warehousing

Optimize query performance and reduce costs by deploying your data warehousing architecture in the AWS cloud. Amazon EMR allows you to leverage the power of the Apache Hadoop framework to perform data transformations (ETL) and load the processed data into Amazon Redshift for analytic and business intelligence applications.
Nasdaq achieved faster, richer analytics and data warehousing capabilities while reducing costs by 57% by shifting to Amazon Redshift and using Amazon EMR for ETL.

AWS Solutions

AI-Driven Social Media Dashboard

https://aws.amazon.com/solutions/ai-driven-social-media-dashboard/?did=sl_card&trk=sl_card