AWS Big Data

V2020.11.11

(Dr Yan Xu)

Kinesis Data Stream

1: Key Concepts

- Data Streams
- Producers
- Consumers

2: Producers

- Kinesis Producer Library (KPL) - Java/C++ only
  - Automatic retry, Asynchronous API, Batching
  - Must read by KCL
- Kinesis SDK
- Kinesis Agent (monitor log files)
  - sudo yum install –y aws-kinesis-agent
  - vim /etc/aws-kinesis/agent.json
  - sudo service aws-kinesis-agent start
- Per Shard: 1MB/s or 1K message/s

3: Consumers

- Kinesis Consumer Library (KCL) - Java/C++ only
- Kinesis SDK
- Kinesis Agent (monitor log files)
- Consumer Delay
  - Standard Consumer Delay ~ 200ms [Pull]
  - Enhanced Fan Out Delay ~ 70ms [Push]
- Per Shard: 2MB/s or 5 API/s

4: Maintenance

- Increase/decrease Shards
- Data Retention: < 7 days & Immutable
- Records are sorted in individual Shard

Kinesis Firehose

➔ A fully managed service for delivering real-time streaming data to destinations with small delays (e.g. 300 sec)

1: Producers

- Kinesis Data Stream
- AWS SDK
- Kinesis Agent

2: Configuration

- Buffer size (e.g. 100MB) & buffer interval (e.g. 300 sec)
- A record < 1,000 KB
- Error Log to S3 (optional)

DynamoDB

➔ High-scalable Low-latency Key-Value Database

➔ Concepts:
- Table
- Item/Row (<400KB)
- Primary Key
  - Partition Key ONLY
  - Partition Key + Sort Key/Range Key

➔ Capacity Provision
- WCU - Write Capacity Unit (KB/s)
- RCU - Read Capacity Unit
  - Strong Consistent Read (4KB/s)
  - Eventual Consistent Read (8KB/s)
- WCU/RCU evenly distributed -> partitions

➔ Index
- Local Secondary Index (same Partition Key)
- Global Secondary Index (new Partition Key)

➔ Cache - DAX

➔ KeyConditionExpression

➔ FilterExpression

➔ DynamoDB TTL (expiration time column)

➔ Point-in-Time Recovery (<35 days)
- Earliest restore date & Latest restore date

➔ On-Demand Backup and Restore (long retention)

➔ Large objects on S3 & Reference on DynamoDB

AWS Lambda

➔ Serverless Processing

➔ Key Concepts
- Function
- Qualifier/Versions
- Trigger
- Execution environment & Runtime
- Event {JSON-formatted inputs}
- Concurrency (reserved concurrency)

➔ Triggers (almost any event in your account)

➔ Runtimes
- Node.js, Java, Python 2&3, Ruby, .Net, Go & Custom.

➔ function handler
- def handler_name(event, context):

➔ Cold Starts
- an issue for first request comes in after deployment
- typically within 1 second (python)
- after API, an instance stays alive to be reused for minutes

➔ Timeout (max 15 minutes)

➔ CI/CD is not well supported, so do it on console.

AWS Glue

➔ Data Catalog (e.g. Database, Table)

➔ Crawler (to extract data catalog)

➔ Target database

➔ Glue ETL Job
- Apache Spark (e.g. Python or Scala)
- Glue Dynamic Frame
- Spark DataFrame

➔ Development Endpoints

EMR

➔ Hadoop Stack on AWS (~ Cloudera)

➔ Cluster
- Master node
- Core node (CPU + Storage)
- Task node (CPU)

➔ S3 (default) v.s. HDFS v.s. local FS

➔ Installed OS & Lib
- HDFS, YARN, MapReduce
- Spark, Hive, HBase, Presto, Hue
- Zeppelin, EMR Notebook (or Jupyter) # hosted elsewhere

➔ EMR notebook could integrate with Git

➔ Security
- VPC & Security Group
- IAM policies and roles
- SSH

➔ Common Instances: m4.large & m4.xlarge

Data Pipeline

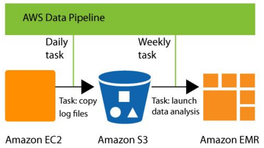
➔ Scheduling Pipelines (e.g. cron)

➔ Data Nodes (inputs and outputs e.g. S3)

➔ Activities (EMR, Hive, SQL, Shell, etc)

➔ Preconditions (e.g. S3KeyExists)

➔ Failure & Retry (3 times)



Step Functions

➔ Serverless state machines and tasks

➔ States (e.g. Wait, Choice, Task)

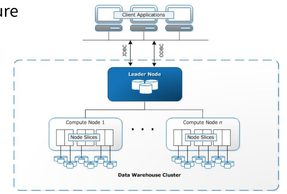
➔ Transitions (Next)

➔ Cloudformation

Redshift

➔ A fully managed, petabyte-scale data warehouse service in the cloud

➔ Architecture



➔ Performance
- Massively Parallel Processing
- Columnar Data Storage (OLAP)
- Column Compression

➔ Performance Tuning
- Distribution Key & Sort Key

➔ Data Distribution: Even, All, Key

➔ Durability
- Continuous and incremental backups to S3
- Automatically recover from node failures

➔ Workload management (WLM)
- query queues (<=8) & concurrency level (<=50)
- separate out time-consuming queries
- short query acceleration (dedicated space)

➔ Stored Procedures

➔ S3 Extension: Redshift Spectrum
- CSV, Parquet, ORC, JSON, GZIP, etc

➔ Security
- SSL in transit
- VPC isolation
- Encryption at rest
- Audit logging

➔ Some special commands:
- VACUUM
- ANALYZE
- EXPLAIN
- COPY -- parallel load & support decryption
- UNLOAD

➔ Scaling
- Vertical & Horizontal Scaling
- Read-only for mins/hours, so in maintenance window
- RA3: scale compute and storage independently

Amazon Elasticsearch

➔ A managed search and analytics engine for use cases such as log analytics, text search, app monitoring.

➔ Concepts:
- Document (e.g. JSON)
- Indices
- Shards

➔ Redundary
- 2 primary shards & 2 read replicas
- Snapshots to S3

➔ Data Ingestion
- REST API
- Kinesis Stream
- Logstash

➔ Full-text queries

➔ Kibana Access is not easy
- Cognito
- Reverse Proxy Server

Athena

➔ Analyze petabytes of data in S3 using SQL

➔ Supported data format on S3:
- CSV, JSON, ORC, Parquet

➔ Supported schema source:
- Glue, Hive
- SQL: CREATE EXTERNAL TABLE

Security

➔ Encryption in flight: SSL
- Kinesis, SQS, S3, DynamoDB, RDS, Redshift, etc

➔ Server side encryption on rest: KMS
- Kinesis, DynamoDB, RDS, Redshift, etc

➔ Client side encryption:
- SQS, S3, DynamoDB, Redshift, etc

➔ STS: temporary token access
- Cross Account Access & Federation