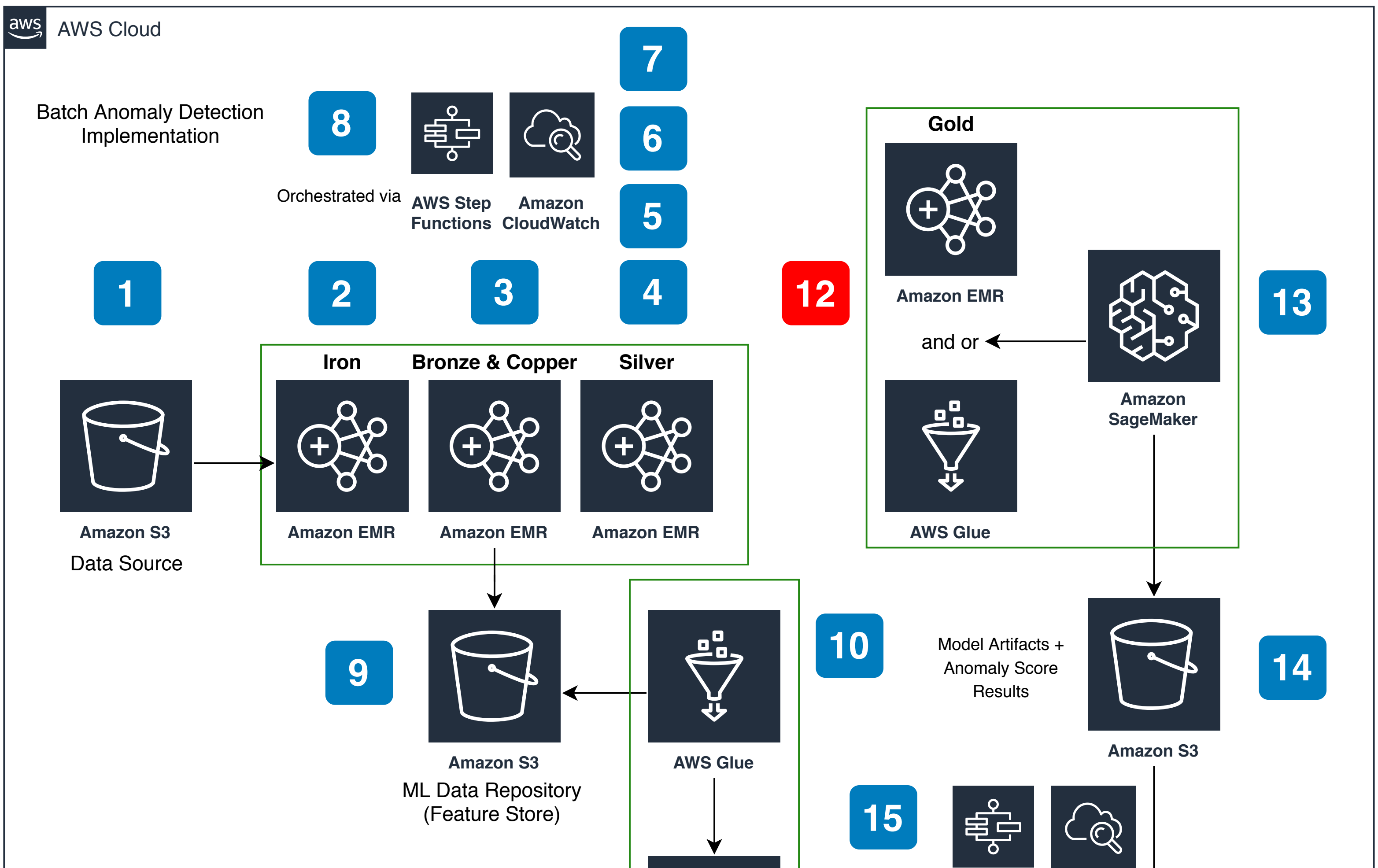
IIoT Machine Learning Pipeline (Detailed Architecture Design)

Developer: Garrett R Peternel - ML Specialist



Index Key:

bucket = {aero-iiot}

raw-json-data-prefix = {datasets/greengrass}

ml-repo-prefix = {datasets/e363549/iiot-analytics-pipeline}

Mapping Step Paths:

1.) s3://{bucket}/{raw-json-data-prefix}/raw/

*2.) s3://{bucket}/{ml-repo-prefix}/**iron**/*

*3.) s3://{bucket}/{ml-repo-prefix}/**bronze**/*

*3.) s3://{bucket}/{ml-repo-prefix}/**copper**/*

*4.) s3://{bucket}/{ml-repo-prefix}/**silver**/autoclaves/*

*5.) s3://{bucket}/{ml-repo-prefix}/**silver**/mills/*

*6.) s3://{bucket}/{ml-repo-prefix}/**silver**/drills/*

*7.) s3://{bucket}/{ml-repo-prefix}/**silver**/tubebenders/*

*12.) s3://{bucket}/{ml-repo-prefix}/**gold**/*

13.) s3://{bucket}/{ml-repo-prefix}/artifacts/

14.) s3://{bucket}/{ml-repo-prefix}/results/

This Big Data Analytics & Machine Learning Pipeline has been designed to implement scalable & elastic cloud computing infrastructure, loosely coupled fault tolerance & high availability, cost effective transient batch analytics & ML automation, and fast performing data access, retrieval, and curated reusability ... to support a plethora of IoT use cases (Supervised Predictive Maintenance, Unsupervised Anomaly Detection, Rule Based Heuristics) ... mainly centered around Machine Learning and Deep Learning to develop offline pre-processing and model training ... however the data is efficiently available to perform Descriptive BI Analytics if desired.

The goal of this advanced architecture is to help the corporation augment, innovate, and improve its cloud computing competencies through Digital Transformation via cutting edge technology being utilized by competitors and industry leaders globally to best solve their business problems and respond quickly to agile change.

Main Functional Capabilities & Technical Components

Technical Pipeline Capability	Workflow Step(s)	Problem Solver(s)	Benefit(s)
incremental streaming conversion of incoming semi-structured data to structured columnar/compressed format schema	2	"baseline data in structured parquet schema" & "compressed/columnar format available"	incrementally processed as 1 step (JSON schema normalization & file format conversion) instead of 2 persisted steps which saves storage cost, extra metadata/schema layer, and additional ETL management
all data available (last 2 years ~12 billion rows) in analytics format (parquet)	3	"organized data lake structure" & "historical data available"	typically more data available results in better BI aggregations, statistical data distribution understanding, and DL/ML model results
elastic scalability of increasing data volume	2, 3, 4, 5, 6, 7, 12, 13	""not limited to single machine data processing bottleneck" & "distributed computing scale-out capability"	transient jobs tuned accordingly to scale computing infrastructure as needed to process large data volumes (i.e. terabytes) for ML/DL development
optimized columnar format with even file size partitions	3, 4, 5, 6, 7	"small file issue" & "predicate pushdown capability"	significant read/write performance (SQL queries for BI, distributed computing jobs & tuning for ML + ETL/pre-processing)
automation/orchestration of transient batch analytics & ML workflows via cluster/instance creation	8, 15	"scheduling transient workflows" & "cost effectiveness"	avoids long running instance $ costs, orchestrates $ integrates chained pipelines, as well as, de-couples the architecture (separate ETL jobs from ML training jobs)
feature store repository of extracted curated datasets for DL/ML training, model artifacts, and inference results	9, 14	"feature extraction layer" & "model management"	training data input reusability & sharing, organized train/validation/test sets, and loosely coupled architecture (permanent storage always available) highly available and fault tolerant; in case a component of architecture stops working it ensures other steps are not impacted
AWS fully managed (serverless) service selection and native API integration	10, 11, 12, 13	"less infrastructure management" & "IT setup"	added AWS service integration & self-service utilization: querying, ETL/pre-processing, workflow orchestration, and algorithm development & deployment
separate ETL and ML	ETL: 2-8; ML: 12-15	"less CPU/GPU cost" & "faster RAM & IO read/write"	ETL and ML processes have different compute and memory requirements hence de-couple both processes to avoid mixing GPU for ETL and CPU for ML bottlenecks

1 Persisted semi-structured nested JSON data following MTConnect ontology containing IIoT device sensor output. Data is delivered from Kinesis Firehose and partitioned by Date in S3.

2 Spark Scala Streaming job continuously reads raw data and flattens/normalizes semi-structured JSON data from #1 to convert schema to Parquet format for columnar storage and cost savings. The job runs in transient mode on a specified timer command line argument (ex: 20 min) then self terminates the cluster. Parquet format results are stored in S3.

3 Spark Scala job reads data from #2 and re-partitions + optimizes file size so all partitions (split files) are the same size to improve I/O performance. The job runs in transient mode with command line arguments for desired file size and number of partitions estimate. Parquet format results are stored in S3. ***Data process (~2TB uncompressed to ~750GB compressed) filtered to last ~2 years***.

4 Spark Scala job reads from #3 and transforms schema so the data-item-id column values (features to be used for ML models) are now individual columns via pivot mapping files for SAMPLE, EVENT, and CONDITION category column. The job runs in transient mode and does this per device for AUTOCLAVES (currently AC1, AC2, AC3, AC4, AC5). Parquet and CSV format results are stored in S3.

5 Spark Scala job reads from #3 and transforms schema so the data-item-id column values (features to be used for ML models) are now individual columns via pivot mapping files for SAMPLE, EVENT, and CONDITION category column. The job runs in transient mode and does this per device for MILLS (currently FOG1, FOG2, DIAMOND-NORTH, DIAMOND-SOUTH). Parquet and CSV format results are stored in S3.

6 Spark Scala job reads from #3 and transforms schema so the data-item-id column values (features to be used for ML models) are now individual columns via pivot mapping files for SAMPLE, EVENT, and CONDITION category column. The job runs in transient mode and does this per device for DRILLS (currently FAD1, FAD2). Parquet and CSV format results are stored in S3.

7 Spark Scala job reads from #3 and transforms schema so the data-item-id column values (features to be used for ML models) are now individual columns via pivot mapping files for SAMPLE, EVENT, and CONDITION category column. The job runs in transient mode and does this per device for TUBE BENDERS (currently CRIPPA_NE, CRIPPA_NW, CRIPPA_SE, CRIPPA_SW). Parquet and CSV format results are stored in S3.

8 EMR Jobs #2, #3, #4, #5, #6, and #7 are orchestrated via AWS Step Functions using State Machine's Amazon State Language (ASL) which can be scheduled via CloudWatch Events.

9 ML data repository established in S3 for storing transformed and curated datasets for ML (i.e. train, validation, test sets)

10 Glue crawlers are scheduled to scan persisted Parquet files generated from #3, #4, #5, #6, and #7 to build database tables for server-less schema inference and metadata management.

11 Athena tables are available to query the transformed IoT data for server-less EDA, reporting, and aggregation purposes.

12 Python jobs (per AUTOCLAVES, MILLS, DRILLS, TUBE BENDERS) read from #9 and perform pre-processing (time series re-sampling, cleaning, imputing, feature selection, and feature engineering) steps to prepare data for ML algorithms. ***Due to VPC networking and security connectivity challenges the goal is to integrate EMR and or Glue with SageMaker via SageMaker Spark SDK***. Currently jobs are run on standalone EC2 instance. CSV format results are stored in S3.

Update: Task of connecting EMR and SageMaker has been accomplished via VPC Subnet and Security Group configurations and will be leveraged accordingly

13 SageMaker training jobs (per device per dataset read from #12) are executed via SageMaker Python SDK using the Random Cut Forest Anomaly Detection algorithm container. Model artifacts are stored in S3.

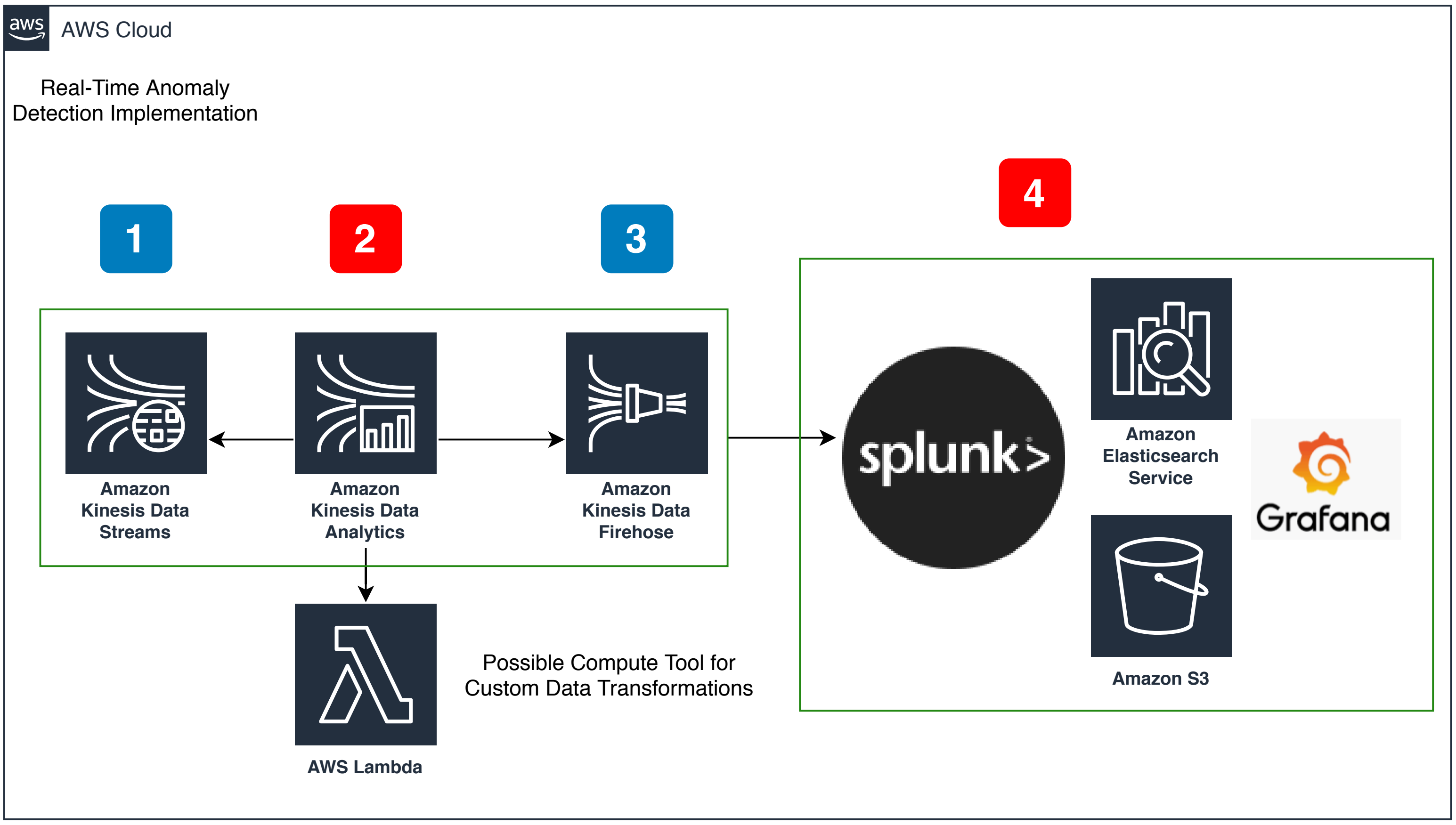
14 SageMaker batch transform jobs (per model from #13) are executed via SageMaker Python SDK. Anomaly score + timestamp results are stored in S3 as CSV format for each device. Score results are at 1 min intervals ranging between DEC 2018 to JUNE 2020. Post-processing Python script unions all results together.

Disclaimer: data discrepancies exist (time series gaps) and are more prevalent for some devices

15 SageMaker Jobs #13 and #14 are orchestrated via AWS Step Functions using State Machine's Amazon State Language (ASL) which can be scheduled via CloudWatch Events.

16 Discussions on:
- who is the customer?
- what SMEs can validate anomaly scores to help generate labeled "ground truth" datasets for predictive maintenance?
- what front-end will display ML results?
- what use case/who needs ML to solve problems?
- deploy as batch transform or self-service endpoint?
- customer wants results in real-time or batch?

Update: results have been uploaded to Grafana for SME exploration and validation via dashboard visuals



aws
AWS Cloud
Real-Time Anomaly Detection Implementation
1
2
3
4
Amazon Kinesis Data Streams
Amazon Kinesis Data Analytics
Amazon Kinesis Data Firehose
splunk>
Amazon Elasticsearch Service
Grafana
Amazon S3
AWS Lambda
Possible Compute Tool for Custom Data Transformations

1 Kinesis Data Streams Producer puts IoT device data into stream via some data retention period (24 hours to 7 days).

2 Kinesis Data Analytics (***pending AWS GovCloud availability***) consumes IoT device data from Kinesis Data Stream. Heavy data transformations, cleaning, and schema mappings will take place to prepare data for Random Cut Forest Anomaly Detection algorithm. ***Unsure if Kinesis Data Analytics is capable of performing these data engineering tasks + handling data discrepancies (time series gaps, missing data-item-id values, missing values/columns, pivoting, one-hot-encoding, etc.)***

3 Pump Anomaly Score results from #2 to Kinesis Data Firehose to bundle results and output to target destination (ex: Splunk, S3, Elastic Search)

4 Discussions on:
- who is the customer?
- what SMEs can validate anomaly scores to help generate labeled "ground truth" datasets for predictive maintenance?
- what front-end will display ML results?
- what use case/who needs ML to solve problems?
- deploy as batch transform or self-service endpoint?
- customer wants results in real-time or batch?