DATA+AI SUMMIT 2022

Pipeline Migration

A Case Study in Rearchitecting an On-Prem Pipeline in the Cloud

Duke

Mary Clair Thompson

Data Engineering Team Lead, Duke University

ORGANIZED BY databricks

Background

- Several data pipelines handled by Spark (on-prem)
  - Spark cluster going away
    - Physically leaving data center!
- Need to migrate workflows to Azure
- For this talk:
  - STINGAR logs pipeline

Background

- STINGAR
  - Shared Threat Intelligence for Network Gatekeeping and Automated Response
    - Partnership among universities for sharing information on network attacks
    - Use data for analytics, reporting and machine learning
- Data aren't huge
  - ~162MB/day shared partner attack data
- Analysts need:
  - Timely access to (lightly processed) raw data (within a few hours)
  - Aggregations (day-level)

Background

Sample log


```
{
        "ids_type": "network",
        "dest_ip": "172.19.0.2",
        "type": "cowrie.sessions",
        "loggedin": ["root", "E5efEHW65"],
        "src_port": 49686,
        "@timestamp": "2022-05-19T17:08:21.199Z",
        "command": "lscpu | grep Model",
        "tags": ["beats_input_raw_event"],
        "src_ip": "43.154.138.122",
        "vendor_product": "Cowrie",
        "severity": "high",
        "dest_port": 2222,
        "protocol": "ssh",
        "app": "cowrie",
        "transport": "tcp",
        "signature": "command attempted on cowrie honeypot",
        "sensor": "nrao-forwarder"
}
```

Background

- What we needed to replicate:
  - Log Ingestion
  - Processing
    - Field extractions
    - Formatting
    - Aggregations
  - Storage
    - Cleaned raw data
    - Aggregated views
  - Analytics platform with access to underlying data

Requests for Updated Pipeline

- Easy for analysts to search raw data by timestamp
- Spark cluster for analysis
- Ease transition for analysts
  - new environment (cloud vs on-prem)
  - experience similar to Jupyter notebooks
  - allow them to recycle Spark code

Original Pipeline

STINGAR

Capture logs

Filebeats

Monitor for new logs/forward to Logstash

Logstash

Write logs to HDFS

HDFS

Store raw, cleaned and aggregated logs

Spark

Perform field extractions and aggregations

Jupyter

Analytics platform

Original Pipeline

- Logstash -> HDFS
  - json
  - 1 file/minute (regardless of logged timestamp)
- Rewrites:
  - Spark job
    - ran ever 5 minutes
    - performed field extractions/cleaning
    - final files written to parquet
    - filenames based on logged timestamp
      - down to 5-minute level
      - 2020-06-12_13:15:00/part-<UID>.parquet

Original Pipeline

- Some problems:
  - decision to store in parquet made before fleshing out full pipeline
    - parquet not necessary for STINGAR data
  - none of the tools in the rest of the pipeline could read/write parquet
  - rewrite process ran into problems with field inconsistencies
  - hard to get throttling correct
    - 5-minute process might still be running when next one starts
    - complicated renaming scheme with potential for files left in zombie state
    - had to hard-code throttling constants on a per-pipeline basis

Original Pipeline

- More problems:
  - Processed logs written out based on detailed timestamp
    - Files dated down to 5 minute intervals
      - Intended to make searching files by timestamp easy for analysts
      - Is this necessary? Isn't this what Spark is for?
  - All processing/aggregation handled by Spark
    - Not a huge amount of data
    - Do we really need Spark for this part of the pipeline?

Architectural Decisions

Needs

- Log Ingestion
- Processing
- Storage
- Analytics

Choices

Architectural Decisions

Needs

- **Log Ingestion**
- Processing
- Storage
- Analytics

Choices

- filebeats/logstash
  - forward on-prem logs -> cloud
- Azure Eventhubs
  - pub/sub

Architectural Decisions

Needs

- Log Ingestion
- **Processing**
- Storage
- Analytics

Choices

- Azure functions
  - serverless compute
  - field extractions/cleaning
  - write processed logs to blob storage
  - chose Premium plan
    - pre-warmed instances
    - avoid cold start problem

Architectural Decisions

Needs

- Log Ingestion
- Processing
- **Storage**
- Analytics

Choices

- Azure Blob (data lake)
  - inexpensive
  - no current use case for structured data/sql querying
- Store in json format
  - relatively small amount of data
  - prefer human-readable files
  - easier to deal with field inconsistencies

Architectural Decisions

Needs

- Log Ingestion
- Processing
- Storage
- **Analytics**

Choices

- Azure Databricks
  - Spark access for analysts

Initial Cloud Architecture

Logstash

Forward logs

Eventhub

Message queue

Azure Function

Processes logs and writes to storage account

Storage Account

Archives logs

Databricks

Hooked up to storage account for analytics on top of data

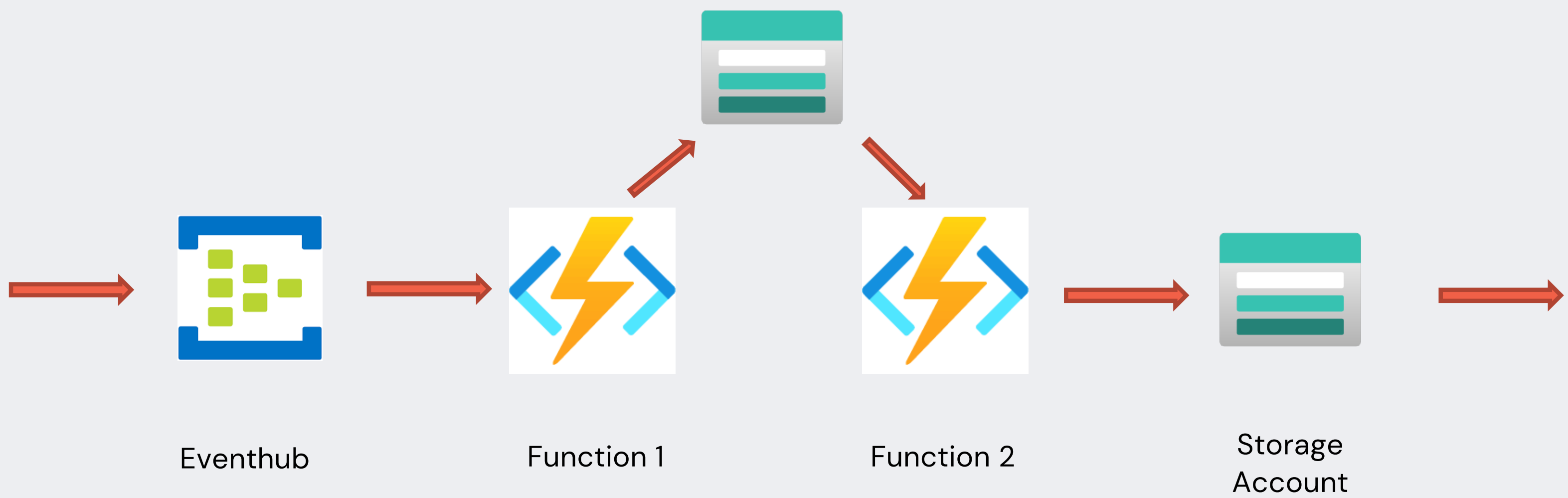
PREFECT

Aggregates data in batch and stores

Complications

- Can’t tune batch size between pub/sub and function layer so...
  - lots of tiny files in blob storage
  - slow to read/process
- Try append blobs
  - problem: Spark can’t read these!

Updated Architecture



Processes logs and appends to a file named by *time of ingest*

Reads append blobs, breaks up by *logged timestamp,* writes out for analysis.
Runs every 10 minutes.

DATA+AI
SUMMIT 2022

Function Details

- Function 1
  - field extractions
  - minor cleanup
  - add timestamped filename based on log time

Core Function Code

Function 1

```
def process_json(event: func.EventHubEvent) -> dict:
        data = event.get_body().decode('utf-8’)
        event_data_json = json.loads(data)
        timestamp = event_data_json.get('@timestamp', dt.now()))
        logging.info(f'Python EventHub trigger processed an event for {timestamp}’)
        ts = dt.strptime(timestamp, '%Y-%m-%dT%H:%M:%S.%fZ’)
        filename_timestamp = dt.strftime(ts, "%Y-%m-%d/%H")
        updated_event = dict(event_data_json, **{"filename": filename_timestamp})
        return updated_event

```

Function Details

- Function 2
  - all the processing is already done so….
    - break up data by timestamped filename field
    - write final files
      - timestamp filenames to hour level

Advantages of Updated Architecture

- New pipeline significantly more stable
  - pipeline hiccups are rare….
    - data scientists trust that the data is up-to-date and reliable
  - although they do happen
    - straightforward recovery
  - no collisions
    - 10-minute processes could handle many orders of magnitude more data
- More flexible architecture -> many more use case possibilities
  - data augmentation
  - aggregations
  - external jobs relying on data

Pipeline Enhancements

- Databricks layer
  - significantly more flexible than in-house Jupyter notebooks solution
    - ad-hoc augmentation of base data
    - integration with external packages and tooling
- GitLab integration
  - CI/CD definition and function template allow for seamless development/testing
    - dev branch deploys code to test function
    - main deploys to production function
    - extremely reusable!
- Networking/access control
  - All components live in dedicated cloud virtual network
    - peered to Duke's network
    - multiple layers of access control
      - easy to provide access to specific datasets

Advantages of Move to the Cloud

- Fit solution to the problem—not vice versa
- Flexibility
  - add/swap out pipeline components
  - scale compute on the fly
  - fine-grained authentication controls
  - on-demand Spark cluster
    - for analysis (where we need it)
    - not for processing (where we don't)
- Appropriate tooling
  - pub/sub
  - lightweight serverless compute
  - Databricks

"This Azure Databricks setup definitely restored my interest in STINGAR"

--Gagan Kaur

Data Scientist

Lessons Learned

- Don’t blindly replicate existing infrastructure
  - Could handle cleaning/processing without Spark
  - Filenames dated to 5-minute intervals unnecessary/added overhead
  - Parquet not necessary here
    - Use json to deal with changing schema
- Don’t discount existing ideas
  - Eventually used the original 2-pronged approach for processing/storage
    - albeit with updates!

Outcome and Related Work

- Data from new pipeline used for:
  - machine learning on attack trends
  - long-term analysis and reporting
  - information-sharing among partner universities
- Applied our learning to harder problem
  - near-real time DNS monitoring
    - higher throughput
      - ~ 16.5GB *compressed* data/day
    - lower latency
      - need logs within 5-10 minutes of ingest

DATA+AI SUMMIT 2022

Thank you

Duke

Mary Clair Thompson

Data Engineering Team Lead