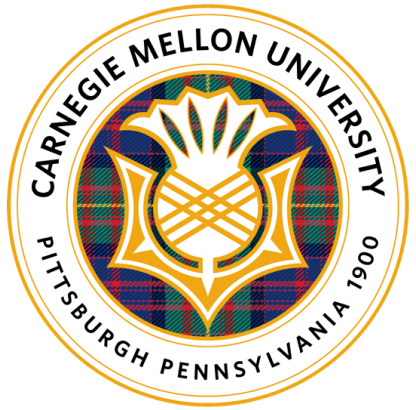
Project Alexandrite

Designing a Flexible Architecture for Real-Time Insights

Capstone Final Presentation | August 7, 2019



Agenda

- Team Introductions
- Sponsor Introductions
- Project Overview
- Research Phase
- Design Phase
- Prototype Phase
- Final Thoughts

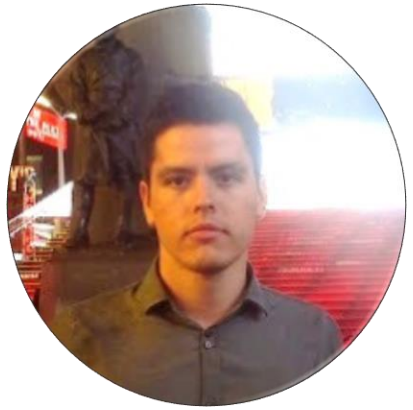
Team Introductions

Amanda Baker
United States

Harold Dansu
Ghana

Sayo Sanu
Nigeria

Sahib Singh
India



Carlos Velasquez
Colombia

Sponsor Introductions

Walgreens and EY have a 12+ year relationship based on trust, quality, and value.

Together they have collaborated on solutions spanning many of the Walgreens functional areas, including Information Technology.

Filled ~1.1 billion prescriptions on a 30-day adjusted basis in fiscal 2018

Operates ~9,560 drugstores

Interacts with ~8 million customers in its stores and online each day

People and operations in more than 150 countries.

Operates in four regions including the Americas, Asia-Pacific, EMEIA and Japan.

Project Overview

Problem

Walgreens executives needed a way to analyze and monitor the retail systems health performance of Walgreens stores

Solution

An IT strategy and architecture that will allow Walgreens to deliver more timely insights from its many applications into visualizations

Research

Research Goals

Divide the data flow process into logical steps

Identify potential solutions for each step

Narrow down recommendations

Design Methodology

Ingestion

Gather data from many different applications

Transformation

Cleaning and formatting data into usable formats

Storage

Holding data in a readily-accessible location

Visualization

Out of Scope

Research Highlights

Ingestion

- Messaging architecture enables asynchronous communication, which provides scalability and reliability
- There is a lower burden on source systems with Change Data Capture (CDC)
- Using an ESB middle layer provides system abstraction

Transformation

- The line between batch processing vs. micro-batch processing depends on data volume and timeliness
- There is a tradeoff between fast querying and flexibility with OLAP

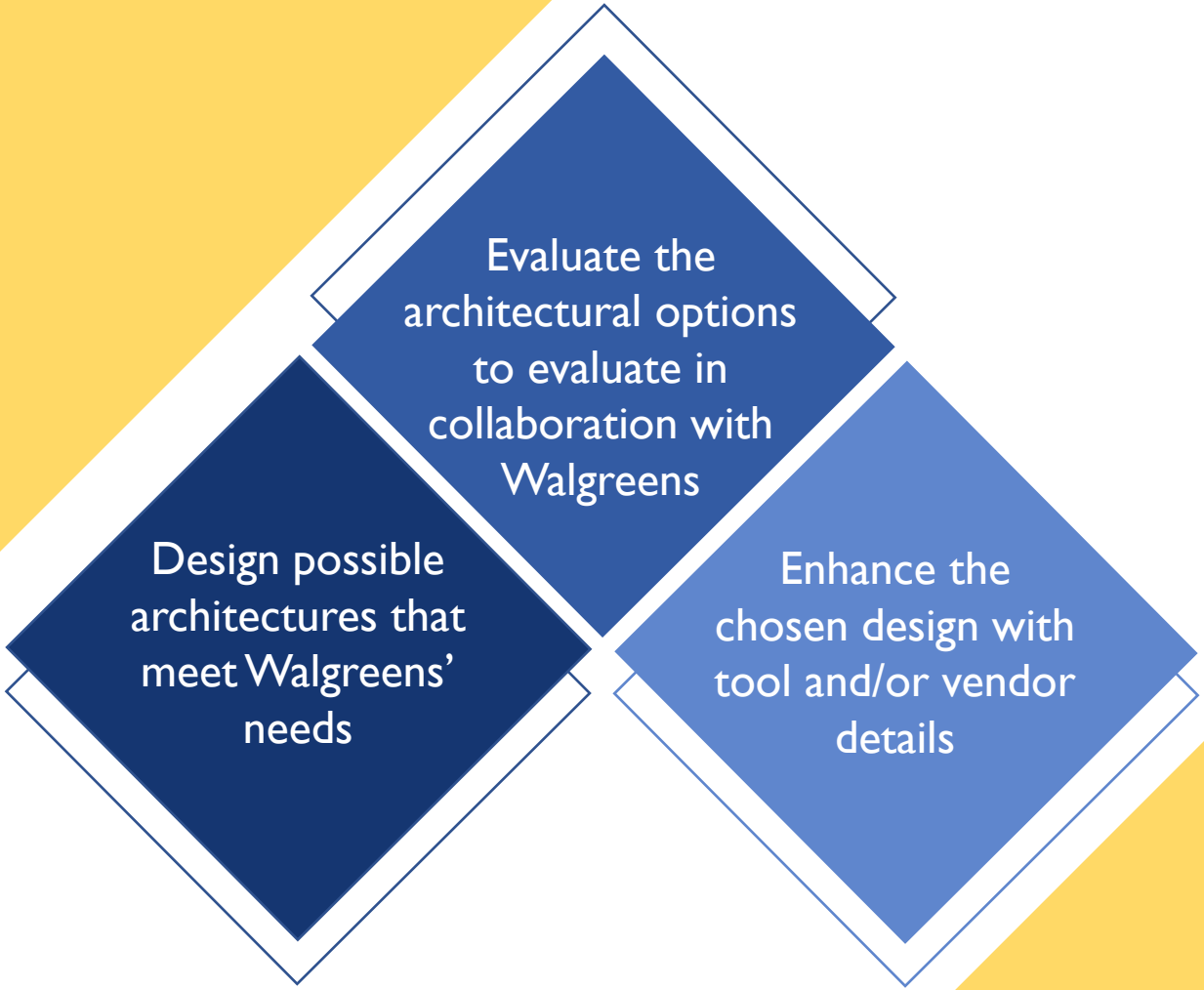
Storage

- Data warehouses and relational DBs are the best choice for complex joins; data warehouses are optimized for aggregations
- NoSQL databases provide more flexibility (no schema required) and horizontal scalability

Distribute processes. Combine solutions.

Design

Design Goals



Architecture Options

A. Peridot

- Focus on lighter technology stack
- Transformations computed between data lake and data warehouse

B. Amethyst

- Balances real-time and historical data
- Transformations computed between data warehouse and data marts

C. Andesine

- Optimized for real-time insights
- Transformations computed within the streaming application

Choices at the visualization stage will **impact upstream architectures**

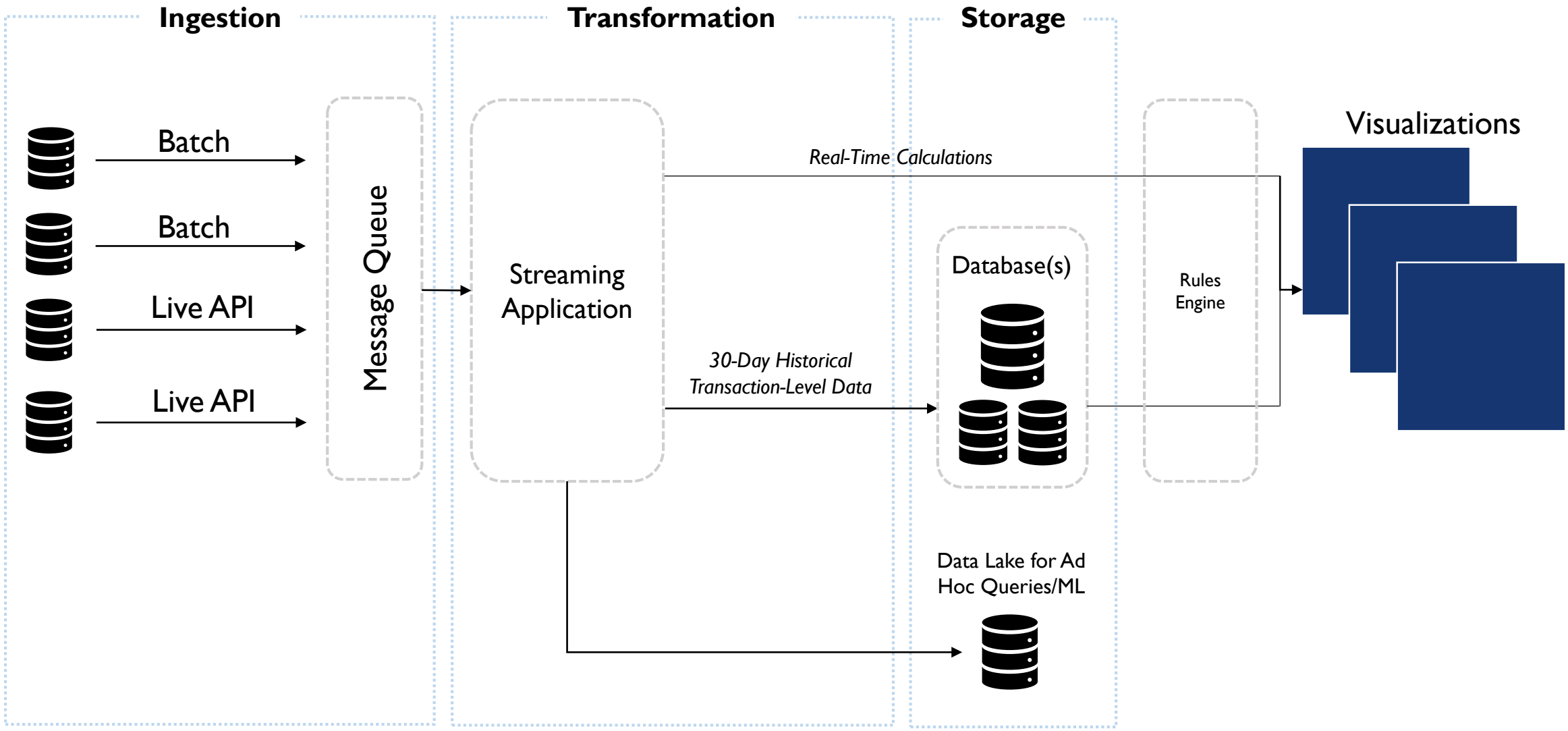
Design Outcomes

Andesine

The Andesine architecture is the most flexible, and will be optimal for accommodating as-yet-unseen needs without the need for an infrastructure overhaul

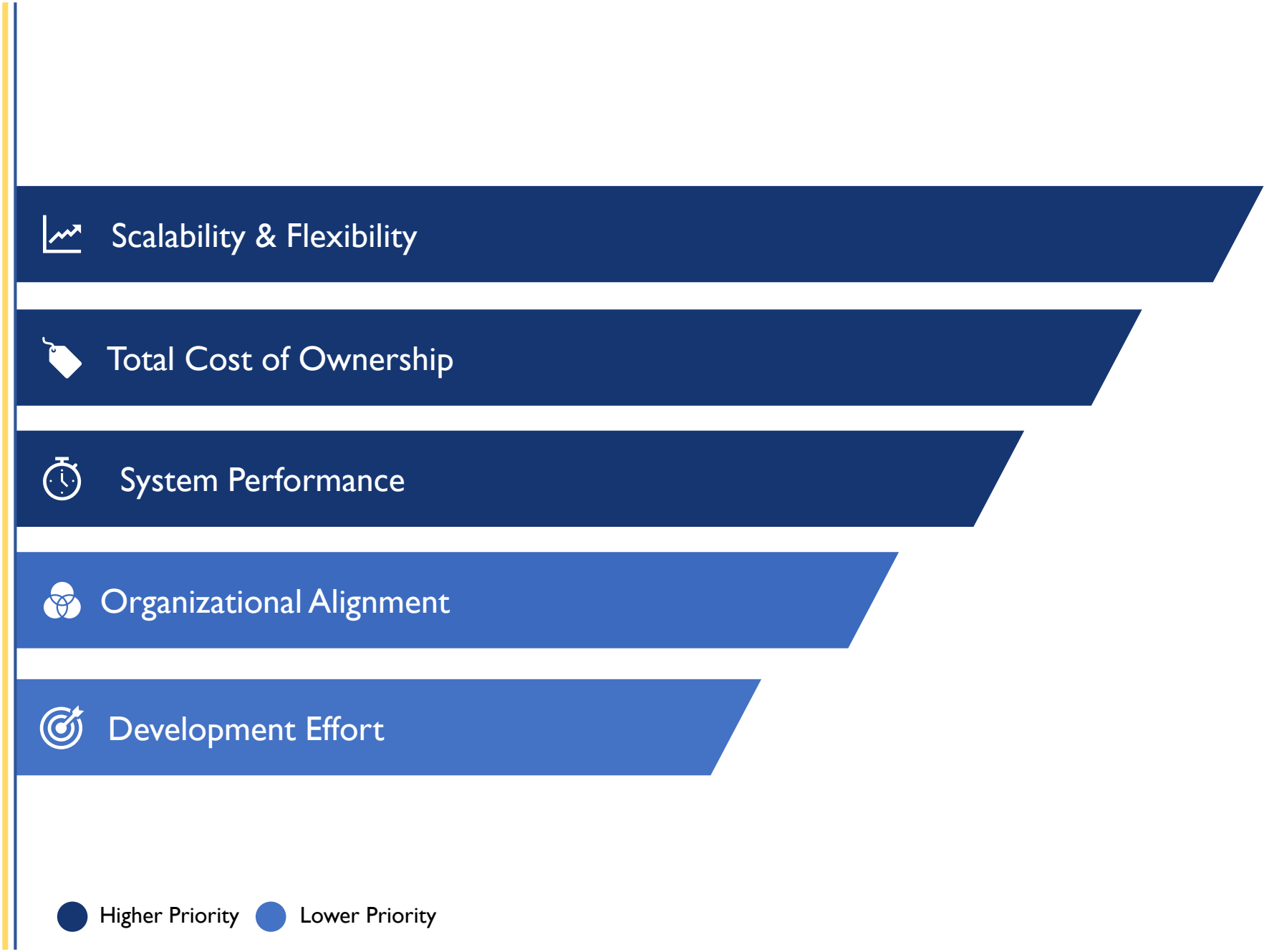
Andesine

Optimized for real-time insights; transformations computed within the streaming application



*Amethyst and Peridot designs in Appendix

Evaluation Scorecard

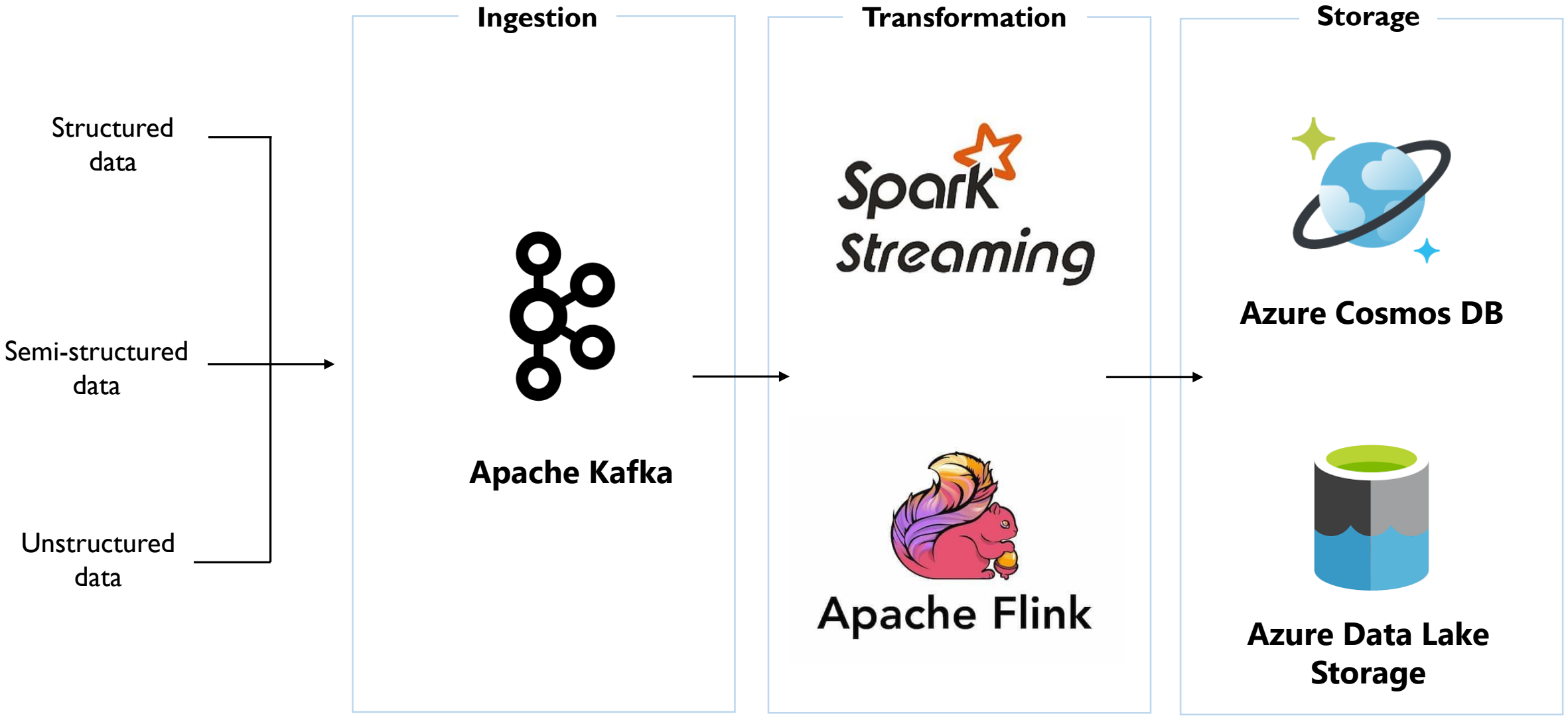


Evaluation Matrix

	Scalability & Flexibility	Total Cost of Ownership	System Performance	Org. Alignment	Development Effort	Weighted Score
Factor weight	5	5	5	3	3	
Andesine	5	3	5	5	3	89
Amethyst	5	3	3	5	3	79
Peridot	2	5	2	5	4	72

Andesine

Recommended tools and vendors



Evaluation Scorecard

Total Cost of Ownership

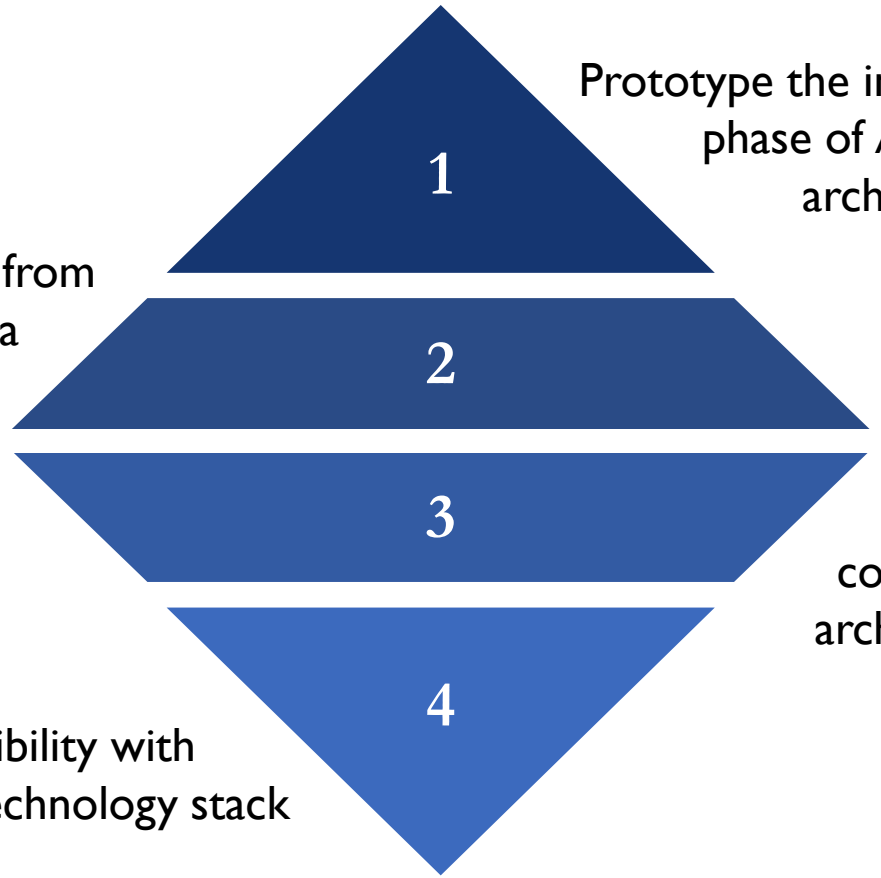
	Andesine	Amethyst	Peridot
Ingestion costs	$1,847	$1,847	$126
Transformation costs	$976	$752	$1,454
Storage costs	$1,818	$1,863	$1,863
Estimated Monthly Cost	**$4,541**	**$4,418**	**$3,329**

Assumptions

- Costs modeled using average data volumes from POS application only
- All pricing based on products available on Azure platform (East US region)
- Azure cloud product pricing structure:
    a. No Upfront costs
    b. No termination fees
    c. Pay-as-you-go pricing (discounts for monthly & annual tiers also available

Prototype

Prototype Goals



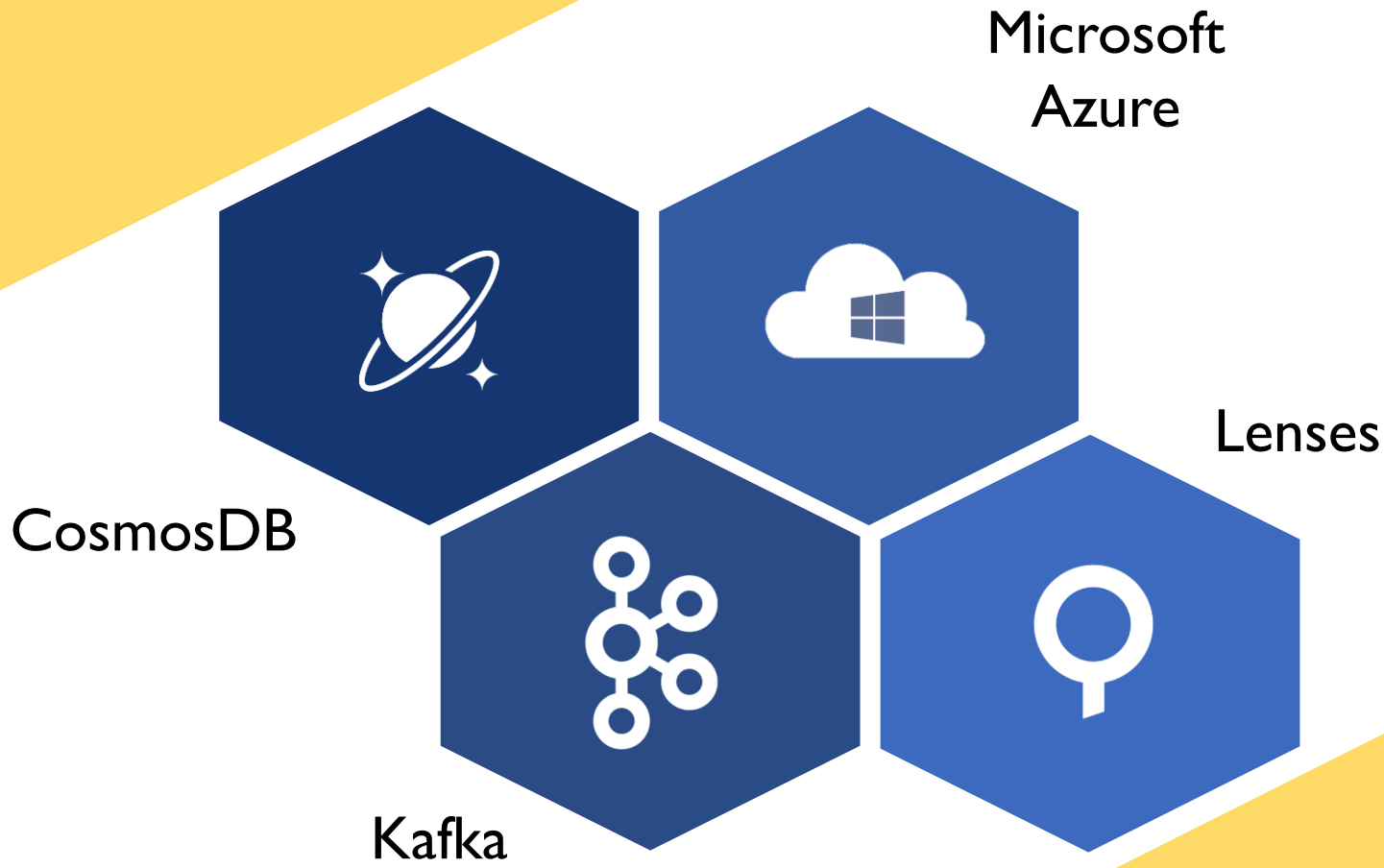
Prototype the ingestion phase of *Andesine* architecture

Simulate data flow from application to Kafka cluster

Build a proof of concept for specific architectural choices

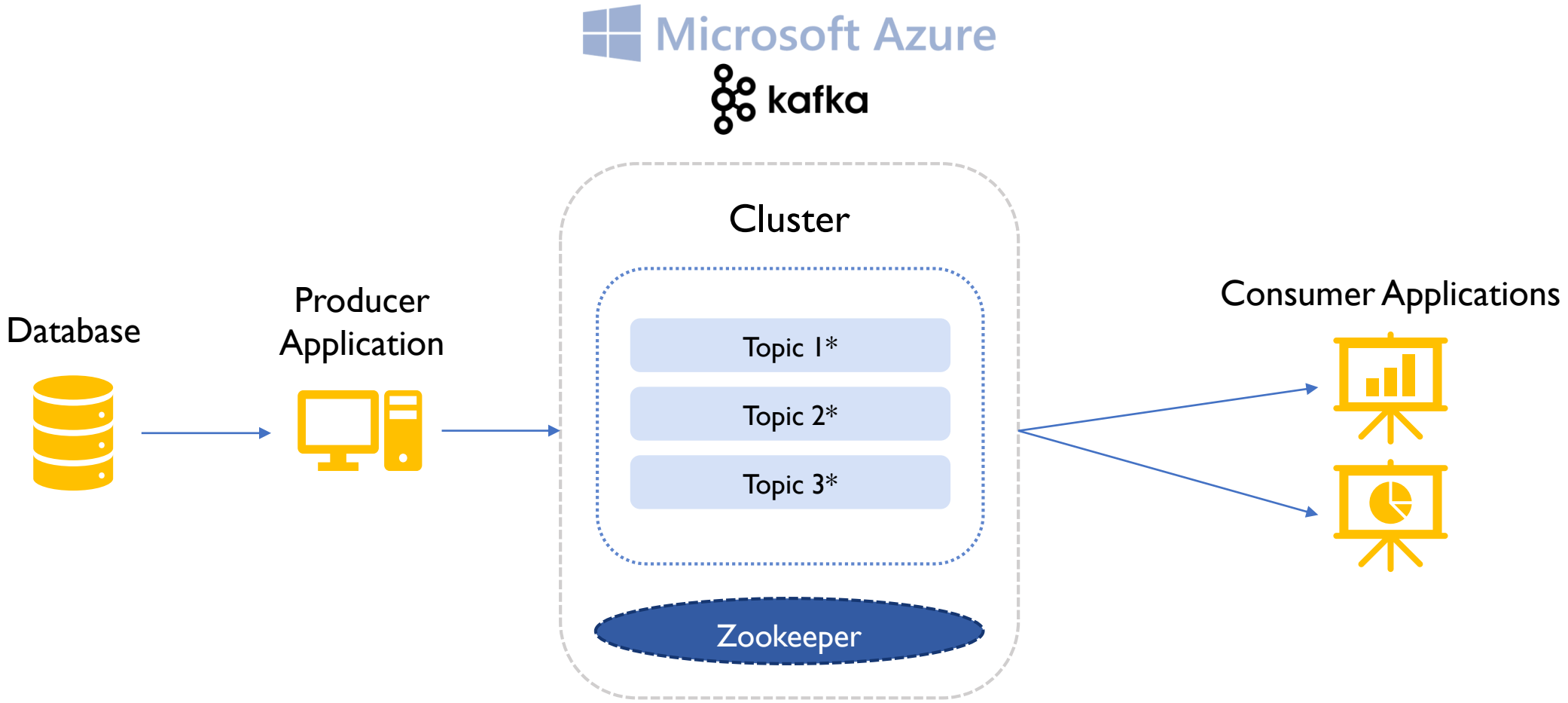
Demonstrate compatibility with Walgreens' current technology stack

Toolbox



Ingestion Framework

The Kafka system can deliver in-order, persistent, scalable messaging and enables massively parallel consumption, perfect for Walgreens' needs.



*A topic is Kafka's term for streaming data

Prototype Demo

New Item | Update | Discard | Delete | Upload Item

SQL API

- ToDoList
  - Items
- walgreensdb
  - Scale
  - container
    - Items
    - Settings
    - Stored Procedures
    - User Defined Functions
    - Triggers

Items

SELECT * FROM c

Edit Filter

id	/_id
dd9f60c0...	wit3ums...
fc14bc59...	ASt3ums...
fdf59314...	CCt3ums...
08cc2d5...	Fyt3ums...
75baacc9...	xhZ6ums...

Load more


```
{
    "_index": "pos-transaction-2019.07.04",
    "_type": "doc",
    "_id": "wit3umsBlMnyvNaGFmsI",
    "_score": 1,
    "_source": {
        "authRoundTripTimeAvg": 625,
        "store-lat": "",
        "indice": "pos-transaction",
        "tags": [
            "filebeat-register",
            "filebeat-transaction",
            "transaction",
            "beats_input_codec_plain_applied"
        ],
        "host": "S04054-RSS21",
        "promotionLookupStartTime": 0,
        "@timestamp": "2019-07-04T00:01:37.071Z",
        "paymentTimeAvg": 11078,
        "receiptEndTime": 1562198497071,
        "pos-store-rft-indicator": "RFT",
        "scanEndTime": 0,
        "promotionLookupDuration": 0,
        "prospector": {
            "type": "log"
        },
        "offset": 8331117,
```

kafkaclusterwalgreens
HDInsight cluster

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems
- Quick start
- Tools

Settings

- Cluster size
- Quota limits
- SSH + Cluster login
- Data Lake Storage Gen1
- Storage accounts
- Applications
- Script actions
- HDInsight partner
- Properties

Move Delete Refresh

Subscription ID : 609753b4-6149-4b8c-9e1c-18db2527fe7d

Tags (change) : Click here to add tags

Cluster dashboards
Cluster management interfaces

Ambari home

Cluster size

9 nodes

TYPE	SIZE	CORES	NODES
Head	D3 v2	8	2
Worker	D3 v2	12	3
Zookeeper	A2m v2	6	3
Edge nodes	DS3 v2	4	1

Cluster

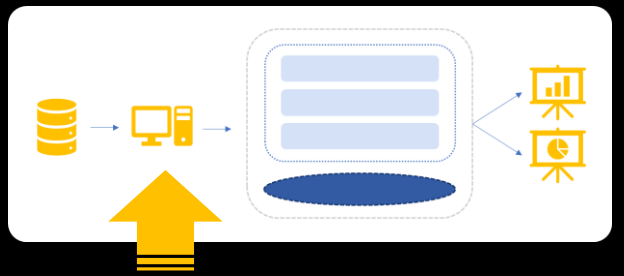
PaymentTime

ScanTime

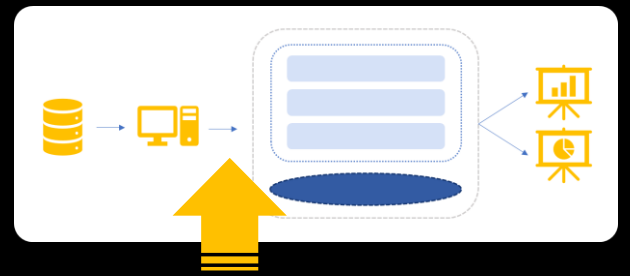
ReceiptPrintingDuration

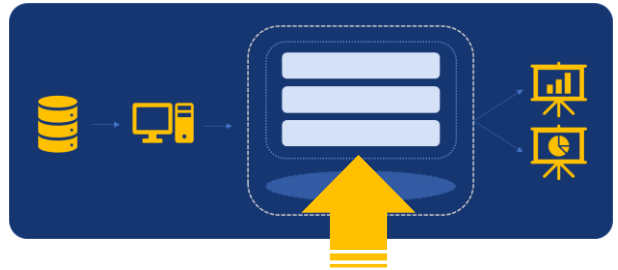
Zookeeper

```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

sahibsin@Azure:~$ ssh sshuser@kafkaclusterwalgreens-ssh.azurehdinsight.net
Authorized uses only. All activity may be monitored and reported.
sshuser@kafkaclusterwalgreens-ssh.azurehdinsight.net's password:
Welcome to Ubuntu 16.04.6 LTS (GNU/Linux 4.15.0-1049-azure x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

0 packages can be updated.
0 updates are security updates.

New release '18.04.2 LTS' available.
Run 'do-release-upgrade' to upgrade to it.



Welcome to Kafka on HDInsight.

Last login: Sat Aug  3 21:23:43 2019 from 104.211.48.205
sshuser@hn0-kafkac:~$ ls
HelloCosmosApplication.java  KafkaConsumerApp  KafkaConsumerApp.zip  KafkaProducerApp  KafkaProducerApp.zip  kafk
sshuser@hn0-kafkac:~$
```



{"scanTime":"672","id":"xSj00msBlMnyvNaG-gkn"}
{"scanTime":"766","id":"xSj00msBlMnyvNaG-gkn"}
{"scanTime":"812","id":"xSj00msBlMnyvNaG-gkn"}
{"id":"xSj00msBlMnyvNaG-gkn","paymentTime":"26531"}
{"receiptPrintingDuration":"6125","id":"xSj00msBlMnyvNaG-gkn"}
{"scanTime":"359","id":"xij00msBlMnyvNaG-gkn"}
{"scanTime":"484","id":"xij00msBlMnyvNaG-gkn"}
{"scanTime":"469","id":"xij00msBlMnyvNaG-gkn"}
{"id":"xij00msBlMnyvNaG-gkn","paymentTime":"13844"}
{"receiptPrintingDuration":"5266","id":"xij00msBlMnyvNaG-gkn"}
{"scanTime":"296","id":"xyj00msBlMnyvNaG-gkn"}
{"scanTime":"359","id":"xyj00msBlMnyvNaG-gkn"}
{"id":"xyj00msBlMnyvNaG-gkn","paymentTime":null}
{"receiptPrintingDuration":"4281","id":"xyj00msBlMnyvNaG-gkn"}
Number of messsages: 38207
[INFO] --
[INFO] BUILD SUCCESS
[INFO] --
[INFO] Total time: 10.507 s
[INFO] Finished at: 2019-08-06T21:52:41+00:00
[INFO] Final Memory: 26M/836M
[INFO] --
sshuser@hn0-kafkac:~/KafkaProducerApp$





LENSES

- Topics
- Schemas
- SQL Processors
- Connectors
- Consumers
- Services
- Audit Log
- ACLs
- Alerts Settings

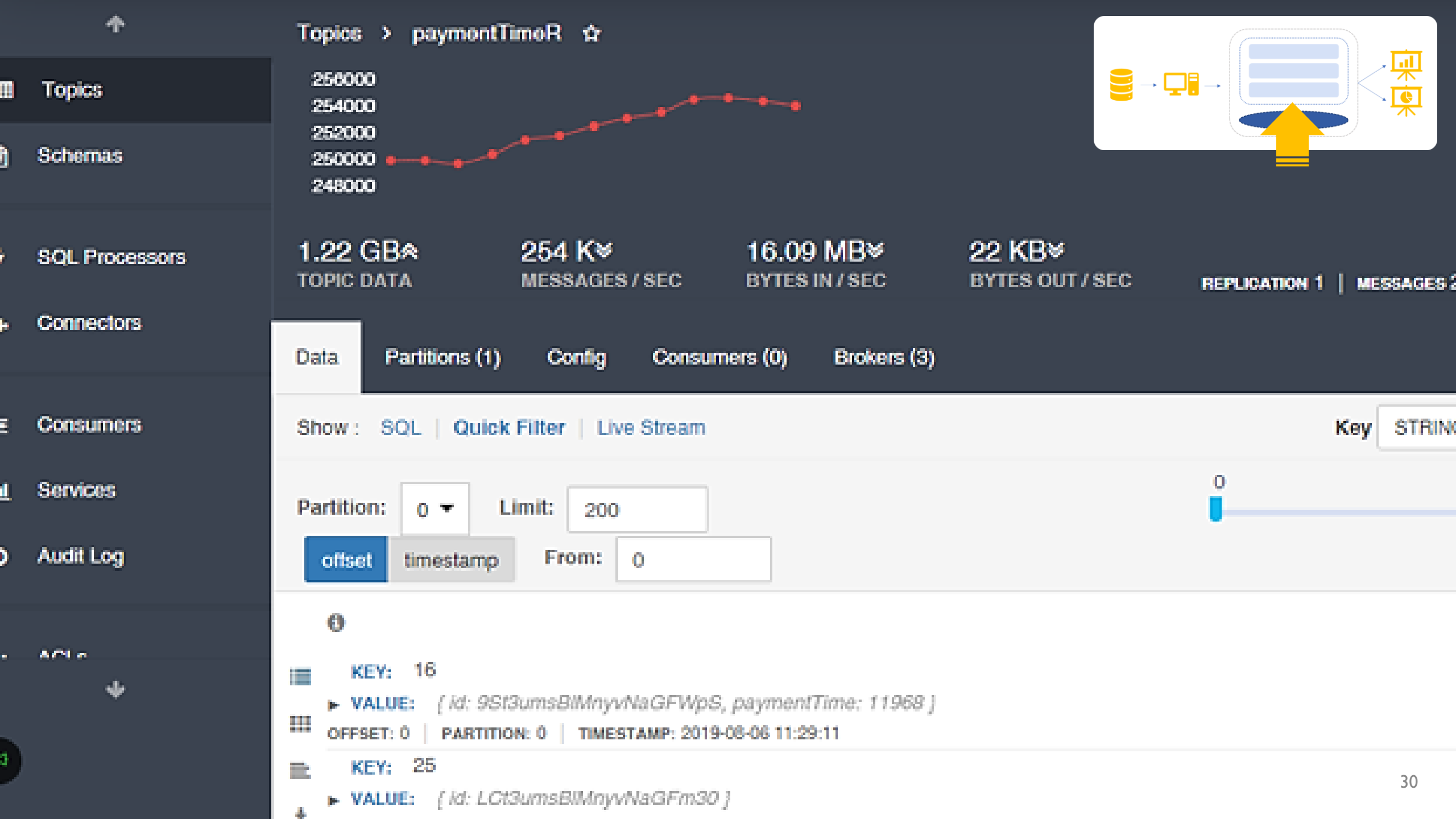
Topics (3)

18.54 MB	0	51B	0B
TOPIC DATA	MESSAGES / SEC	BYTES IN / SEC	BYTES OUT / SEC

search...

Show: **User Topics** System Topics

Topic	Total messages	Me
scanTime	119 K	
receiptsPrintingDuration	39 K	
paymentTime	39 K	


Topics
Schemas
SQL Processors
Connectors
Consumers
Services
Audit Log
Topics > paymentTimeR
256000
254000
252000
250000
248000
1.22 GB
TOPIC DATA
254 K
MESSAGES / SEC
16.09 MB
BYTES IN / SEC
22 KB
BYTES OUT / SEC
REPLICATION 1 | MESSAGES
Data
Partitions (1)
Config
Consumers (0)
Brokers (3)
Show : SQL | Quick Filter | Live Stream
Key STRIN
Partition: 0
Limit: 200
offset
timestamp
From: 0
0
KEY: 16
VALUE: { id: 9St3umsBlMnyvNaGFWpS, paymentTime: 11968 }
OFFSET: 0 | PARTITION: 0 | TIMESTAMP: 2019-08-06 11:29:11
KEY: 25
VALUE: { id: LCt3umsBlMnyvNaGFm30 }

RUNNING | Runners: 1 | Uptime a day | created by: admin

```
1  set autocreate = true;
2
3  insert into combinedM
4  with aux1 as (SELECT *
5  FROM scanTime), aux2  as (
6    select * from paymentTime)
7  select a.id from aux1 b
8  inner join  aux2 a on  a.id= b.id
```

Topology | Monitor | Runners

Stream Topology

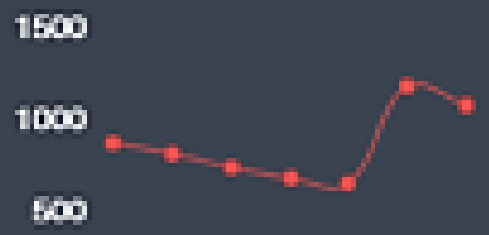
click on the nodes for details

Topic Data: combinedM

Items: 2

Listening for data...

Topics > combinedPS ☆



7.05 MB
TOPIC DATA

1093
MESSAGES / SEC

64 KB
BYTES IN / SEC

83 KB
BYTES OUT / SEC

REPLICATION 1 | MESSAGES 83 K

Data | Partitions (1) | Config | Consumers (2) | Brokers (3)

Show : SQL | Quick Filter | Live Stream

Key STRING Value

Partition: 0 Limit: 200

offset timestamp From: 0

0

KEY: XyxSumsBlMnyvNaG-jFn

VALUE: { id: XyxSumsBlMnyvNaG-jFn }

OFFSET: 0 | PARTITION: 0 | TIMESTAMP: 2019-08-06 11:36:44

KEY: JRl_umsB9LWQDNgqVivm

VALUE: { id: JRl_umsB9LWQDNgqVivm }

Prototype Lessons

- Spend time tuning the cluster for cost savings and performance improvements
- Test different configurations for different node types
- Invest in data ops to continually size requirements based on business needs

The overhead infrastructure represents significant investment of time and resources upfront, but is **easy to scale**

Next Steps

Continue evaluating the tradeoffs between cost and performance throughout all phases

Refine and deploy the Producer Application code base to meet data ingestion needs

Merge internal and external data sources to provide predictive analytics on when systems might go down

Thank You

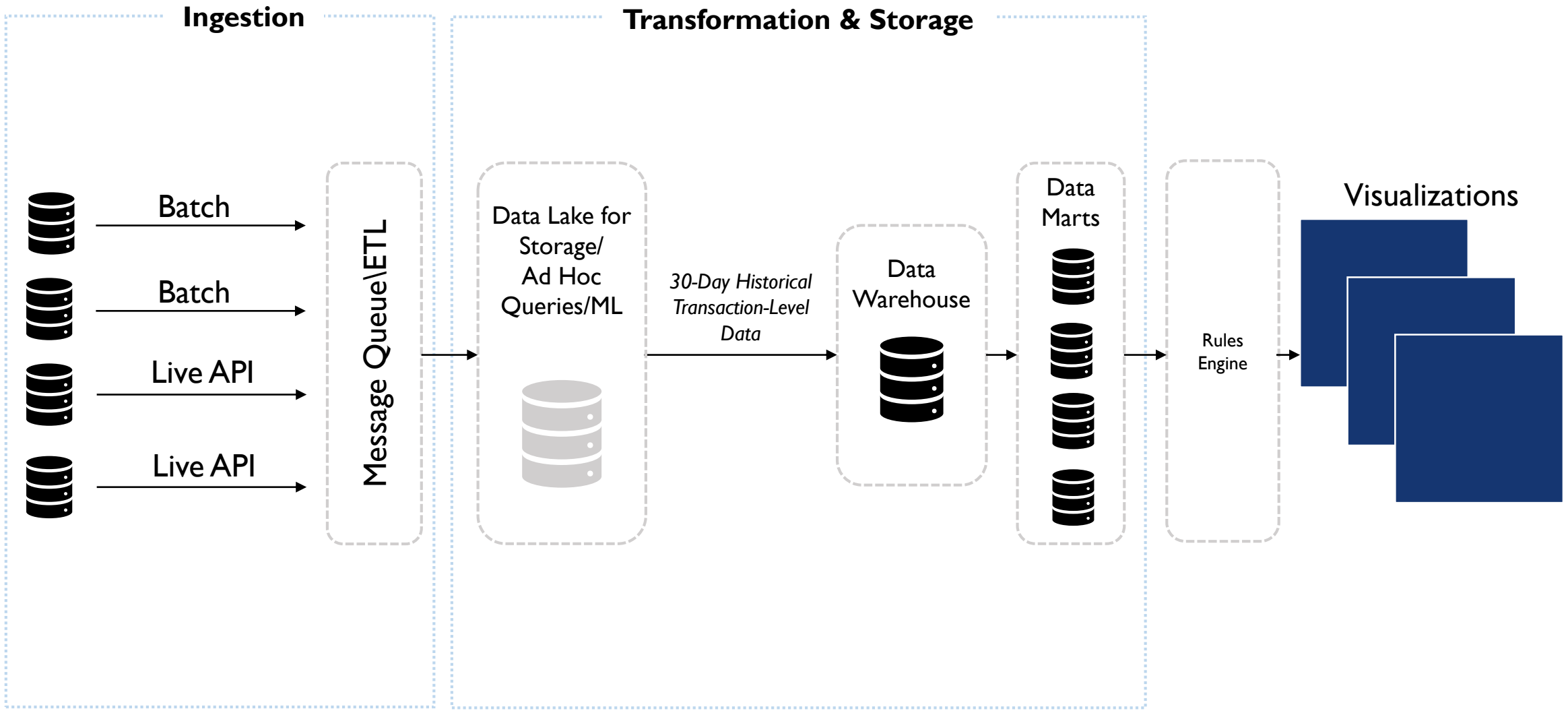
Questions?

Appendix

Additional Design Documentation

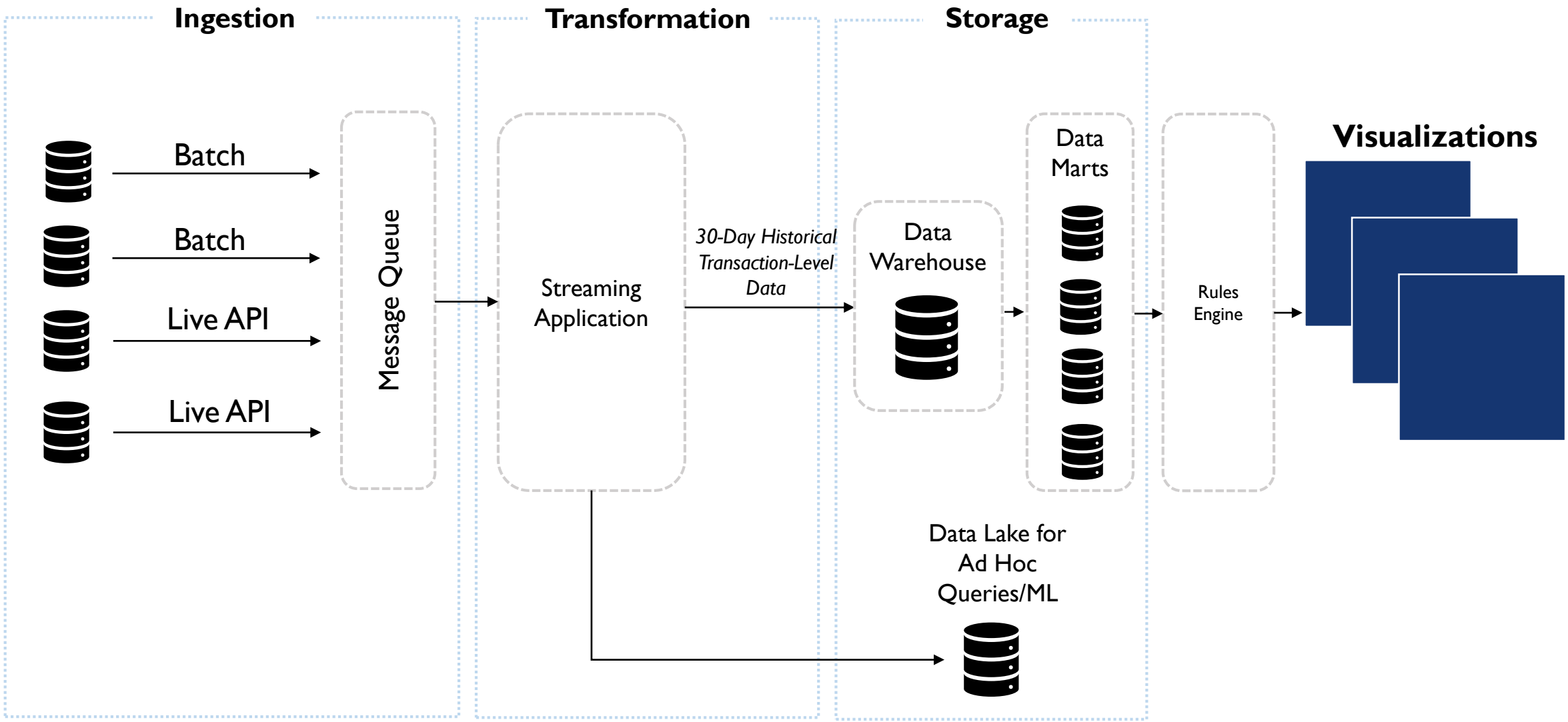
Peridot

Focus on lighter technology stack; transformations computed between data lake and data warehouse



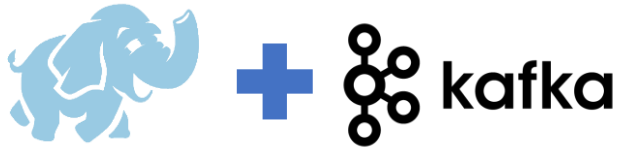
Amethyst

Balances real-time and historical data; transformations computed between data warehouse and data marts



Additional Cost Documentation

Ingestion Costs

		
	Kafka Connect	**Azure Bus**
Scalability & Flexibility	Can handle 10,000x current average data volume	Scalable but after 1000 brokered connections, pricing is opaque
System Performance	Transformations & Aggregations eliminates additional step	Messaging only—No transformations
Organizational Alignment	Available within Microsoft Azure cloud	Available within Microsoft Azure cloud
Development Effort	Equivalent development effort	Equivalent development effort
Estimated Cost of Ownership	$1,847	$426

Transformation Costs

	 	
	Apache Flink / Spark Streaming	**DW / Data Lake Transformations**
Scalability & Flexibility	• Can ingest and combine almost all static and streaming data sources • Multiple concurrent ingress and egress options	If architecture scale out, transformations would be slow or expensive for near-real time
System Performance	Lowest latency option	Slower than streaming application
Organizational Alignment	Available within Microsoft Azure cloud	Available within Microsoft Azure cloud
Development Effort	Significant development effort and performance tuning required	Minimal development effort
Estimated Cost of Ownership	$1,443	$1,806

Storage Costs

	 Cosmos DB	 Azure Data Lake
Scalability & Flexibility	Supports Cassandra, MongoDB SQL (and Javascript), and Apache Spark & Jupyter notebooks	Write transactions 10x more expensive which limits output options
System Performance	Comparable performance	Comparable performance
Organizational Alignment	Available within Microsoft Azure cloud	Available within Microsoft Azure cloud
Development Effort	Similar development effort	Similar development effort
Estimated Cost of Ownership	$511	$288

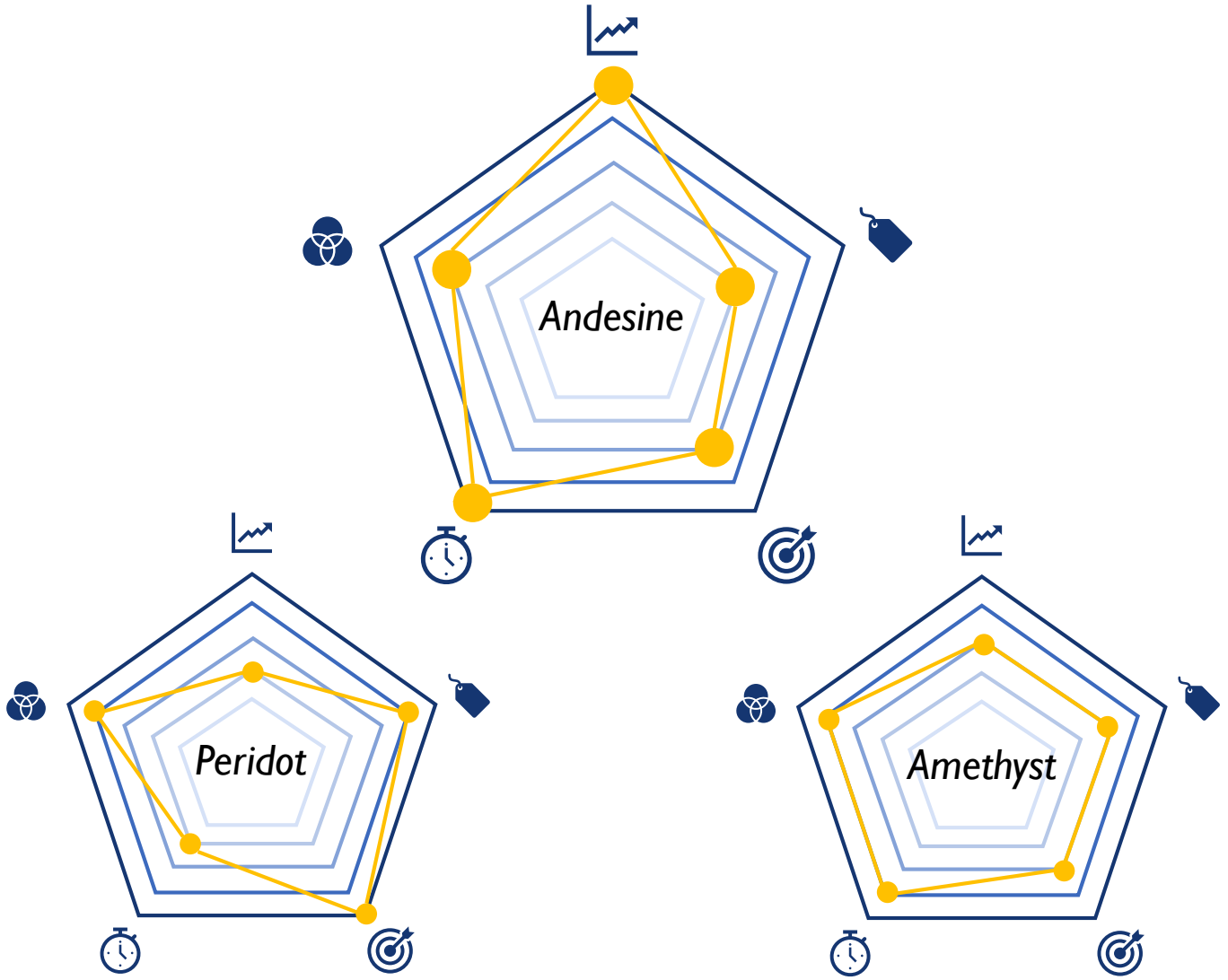
Risk Assessment

Project Alexandrite Risk Assessment

	Low Impact	Medium Impact	High Impact
High Probability	• Users revert to other means for monitoring store systems	• Project does not satisfy business needs • Poor documentation creates confusion and reduces user adoption • Over-prioritizing current needs vs future business needs	• Divergent goals & objectives among stakeholders • Inadequate communication with stakeholders • Dashboard does not meet user needs
Medium Probability		• KPIs not properly defined • SLAs not properly monitored • Deficient data dictionaries & models • Poor UI and/or training resulting in low usage	• Selected architecture does not scale • Lack of support from sponsors • Lack of centralized governance • Poor training result in misinformed users
Low Probability		• Azure availability not guaranteed.	• Incomplete masking or faulty encryption can lead to confidentiality breaches • Inconsistency and inaccuracies during ingestion or transformation • Information silos between units

Radar Charts for Evaluation

Overall Design Evaluations



Research Notes

Ingestion

Considerations & Options

Considerations

A. Supports foundational needs

- Near-real time capability
- Ingests both batch and streamed data

B. Scalable

- Scales out for additional & increased data volume
- Integrates with future input methods

C. Aligns with Walgreens' technology stack

- Cloud-enabled solution
- Azure compatibility

Options

1 Message Queues

- Form of asynchronous service-to-service communication
- Allow for temporal decoupling and load leveling

2 Change Data Capture

- Quickly identifies and processes only the data that has changed

3 Enterprise Service Bus (ESB)

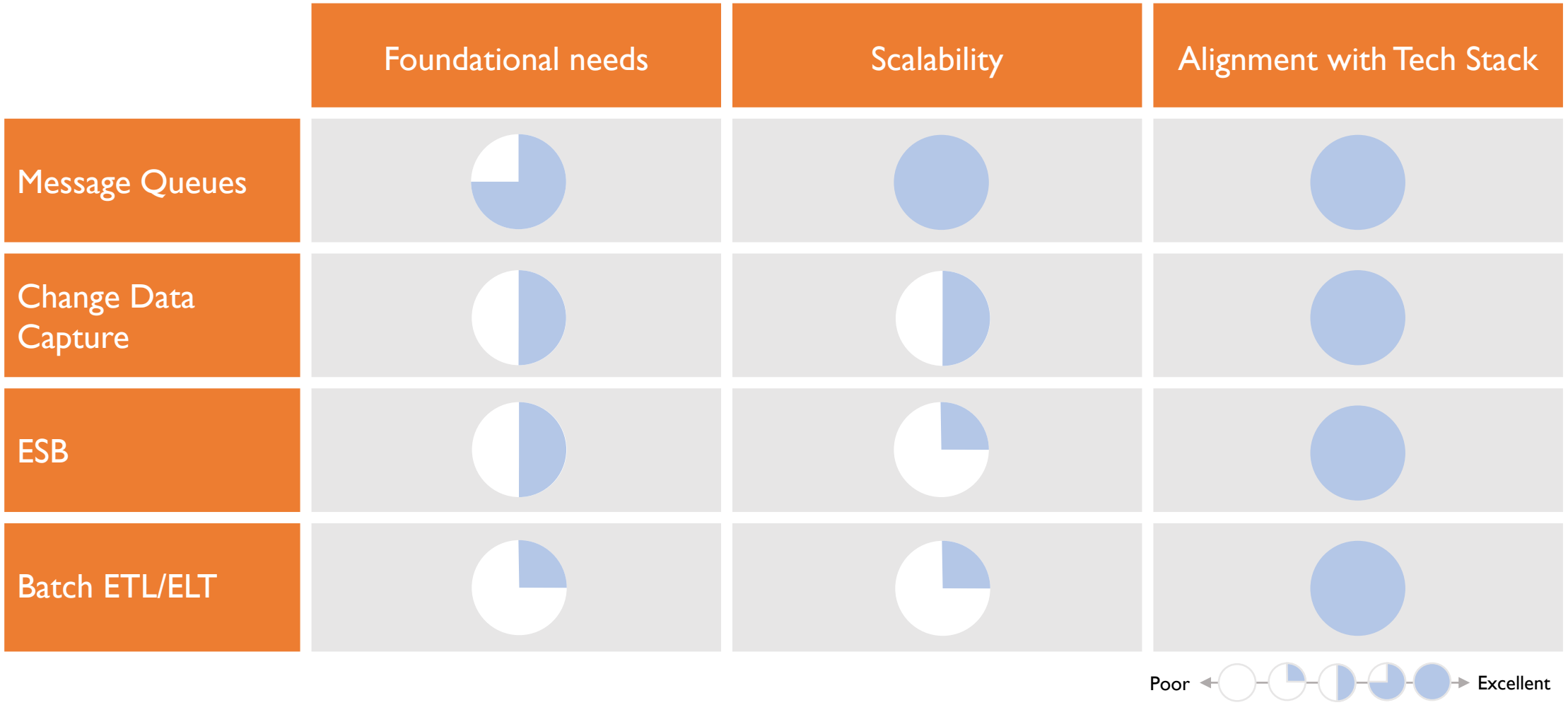
- Enables interoperability between different applications using service orientation

4 Batch E.T.L / E.L.T

- Extracts source data in batches
- Loads onto staging databases and transforms data (or vice versa)

Ingestion

Evaluation



Please refer to Slide 23 in the Appendix for a detailed evaluation of each option.

Ingestion: Additional Details

Methods	Pros	Cons
Batch (ETL/ELT)	• Can use multiple data sources (webservices, files, database) • Effective at getting data into structured format for storage • Can be micro-batched	• Not flexible for changes (e.g. new data sources (unstructured)), new metrics • Not ideal for real-time analytics • Operational and performance cost as data volume increases • Not ideal for time series data
Change Data Capture	• Near real-time • Scalable	• Not easy to implement if the destination is not a RDMS • It needs to have a robust error handling procedure • If you use triggers instead of logs, the performance of the application database can be compromised • When using logs, a full understanding of those is necessary for the cdc to be able to recognize the changing data
Message Queues	• The latency of stream processing will be in seconds or milliseconds • Good for time series data • Event-driven	• Data can only be sent sequentially (i.e. poor option in the case of historical data ingestion)
ESB (Enterprise Service Bus)	• Allows for the integration of systems using multiple technologies (e.g. JMS, Web Services, HTTP, JDBC, etc) • Ideal for integration of service-oriented systems	• May not be forward compatible with future architecture • Some implementations require commitment to particular set of technology or data format • Messaging only abilities. No storage or processing

Transformation

Considerations & Options

Considerations

A. Near-real time analytics

- High performance to process high volume data
- Timeseries data aggregated over windows

B. Scalable/Flexible

- Data transformation for structured & unstructured data
- Data formatting
- Aggregations (summary statistics & granularity)

Options

1 Stream Processing

- High processing power to provide continuous input, process, and output of data

2 Batch Processing

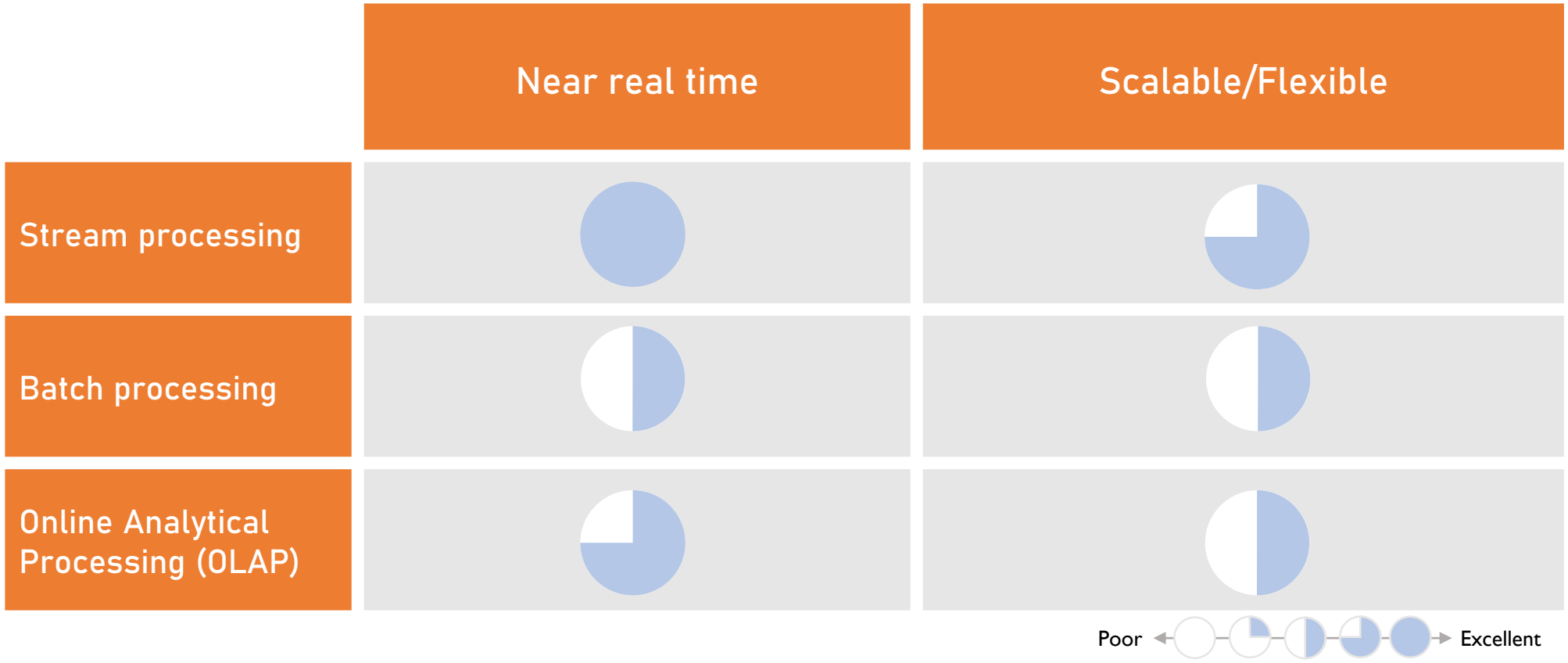
- Processes a group of transactions collected over a period of time
- Performance is impacted with higher volumes of data

3 Online Analytical Processing (OLAP)

- Provides fast access for defined reporting with multi-dimensional views of data stored in database
- Has issues with interference & inconsistencies

Transformation

Evaluation



Please refer to Slide 24 in the Appendix for a detailed evaluation of each option.

Transformation Options: Additional Evaluation

	Description	Merits	Limitations	
Stream processing	Provides continuous input, process, and output of data.	• Low latency • High processing power	• High cost • Analytical data needed	
Batch processing	Processes a group of transactions collected over a period of time	• High throughput	• Higher latency • Data out of date	
Online Analytical Processing (OLAP)	Provides multi-dimensional views of data stored in database	• Provides fast access for defined reporting	• Impact on processing speed with large volumes • Issues with interference & inconsistencies on distributed processing	

Storage

Considerations & Options

Considerations

A. Near real-time
- Fast read vs. fast write

B. Horizontal scalability
- Comparing high consistency, high availability, and partition tolerance (CAP)

C. Data analytics
- Fast aggregation vs. fast searching (document vs. column-oriented)

D. Time series capabilities

E. Joins

F. Raw Data

Options

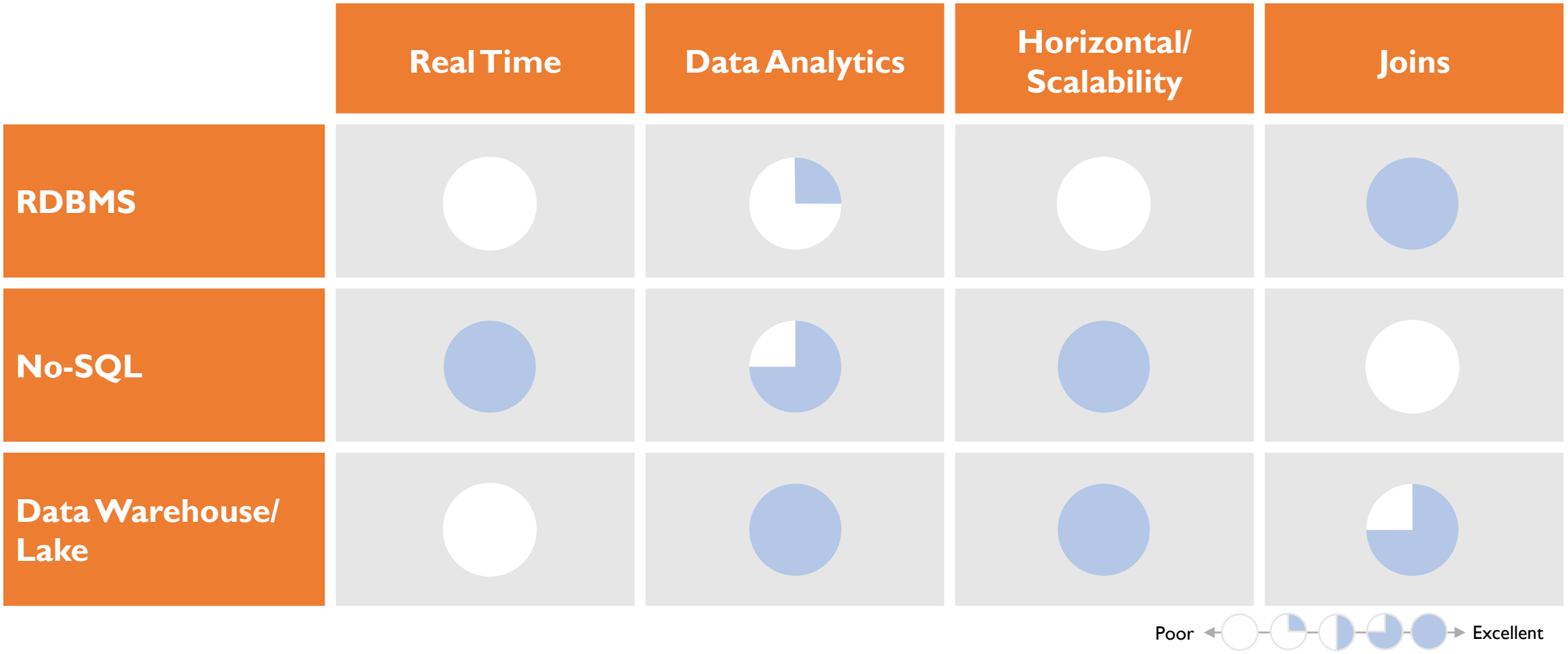
1 NoSQL
- Support for horizontal scaling
- Optimized for real time
- Maintainance of servers is less expensive

2 RDBMs
- Normalized data
- Highly-performant joins
- Support for complex queries

3 Data Warehouse
- Highly scalable
- Optimized for aggregations and big data
- Optimized for read operations

Storage

Evaluation



Storage: Additional Details

Methods	Pros	Cons
RDBMS	• Normalized data • High-performant joins • Use indexes to boost performance • Support for complex queries	• Not optimized for aggregation calculation • Slow reading • Only supports vertical scaling • Only supports tabular data • Fixed schema
No-SQL	• Support for horizontal scaling • Optimized for real time • Maintainance of servers is less expensive • Schemaless • Suited for hierarchical data storage	• Data replication • Not optimal for joins • Redundancy in data due to denormalization • Less support than RDBMS
Data Warehouse	• Highly scalable • Multiple sources • Historical data • Optimized for aggregations and big data • High-performant joins • Optimized for read operations	• Data replication • Not optimized for real time data

Storage: Initial Tool Identification

Methods	High Availability	High Consistency
Document-Oriented	• CouchDB • Riak	• Couchbase • MongoDB
Column-Oriented	• Cassandra	• Hbase
Key-value	• Dynamo	• Redis

Requirements (1/2)

Final set of requirements based on discussion with Walgreens

Requirement	Type	Understanding
Assess what would be the volume of data	Technical	CMU will assess the volume of data, and use this evaluation as part of the technical recommendations for the data lake
Design a roadmap for data the integration layer	Technical	CMU will build out the technical plan and provide documentation on how the data integration layer (lake, warehouse, and marts) can be executed
Design method to aggregate data from all the required business functions	Technical	See above
Design method to optimize the data ingestion process	Technical	CMU will use an understanding of each application and data to optimize the data intregration process • "Real time" is from 60 seconds to 5 minutes ("near-real time")

Requirements (2/2)

Final set of requirements based on discussion with Walgreens

Requirement	Type	Understanding
Define the SLA standards & engine rules	Functional	CMU will document the process for incorporating a rules engine into the architecture to the best of their ability given available information on SLAs and KPIs from Walgreens
Optimize and segment data interface views for user	Functional	CMU will document the architecture around required data marts with data interface requirements in mind
Define key information that would be most critical to an executive audience	Functional	CMU will translate Walgreen's input on KPIs and translate into data models/data marts. CMU will assist with their own knowledge and experience on best practices of KPI creation, but will rely on Walgreens for KPI definitions.
~~Define the KPI's to assess retail store health~~	~~Functional~~	~~See above~~ (Requirement removed, KPIs will be defined by Walgreens and key stakeholders)
~~Design GUI~~	~~Functional/ Technical~~	~~CMU will provide wireframes of custom dashboards and how they map to the data marts~~ (Requirement removed, GUI will be designed by Walgreens and a vendor)

Sources

- https://www.tutorialspoint.com/dwh/dwh_partitioning_strategy.htm
- https://www.digitalocean.com/community/tutorials/hadoop-storm-samza-spark-and-flink-big-data-frameworks-compared#batch-processing-systems
- https://www.slideshare.net/Hadoop_Summit/when-olap-meets-realtime-what-happens-in-ebay
- https://www.linkedin.com/pulse/spark-streaming-vs-flink-storm-kafka-streams-samza-choose-prakash/
- https://data-flair.training/blogs/batch-processing-vs-real-time-processing/
- https://docs.microsoft.com/en-us/azure/service-bus-messaging/service-bus-queues-topics-subscriptions
- https://www.oracle.com/technetwork/articles/soa/ind-soa-esb-1967705.html
- https://docs.oracle.com/cd/B19306_01/server.102/b14223/cdc.htm#i1025409
- https://docs.microsoft.com/en-us/sql/relational-databases/track-changes/about-change-data-capture-sql-server?view=sql-server-2017