Automated Content Recommendation and Personalization

Gopi Krishnamurthy

Data Engineering

This version was last updated 23.06.2019 (v1.0)

Executive Summary - Strategy

Automated Content Recommendation & Personalization

Statement of Content Discovery Management Strategy: Focus on building modern data engineering platform with the goal of providing customers, the best in class personalized user experience and intelligent content discovery using recommendations powered by Machine Learning / Artificial Intelligence.

State of Content Discovery in 2019

- Customer Retention < 15%
- Premium User Conversion Rate < 20%
- App Rating = 2-3 (5 being highest)
- Cost Revenue Ratio >= 3%
- YoY Revenue increase 6.8%
- Percentage of IT budget spent on data curation and engineering > 10%.
- No of non-standard data management / analytic tool set: 15-20

Sample

Top Initiatives

1. Launch data engineering circle to optimize data flows and improve efficiency of data management by adopting niche technologies and industry standards.
2. Redesign/Refactor personalization by adopting machine learning/ artificial intelligence that can provide real-time inferences.
3. Optimize data engineering platform/processes with necessary tools/techniques to decrease time to market the recommendations algorithms.
4. Benchmark recommendations and personalization by continuously monitoring KPIs (e.g. click through rate) and continuously improve algorithms based on A/B testing.
5. Harmonize and integrate data engineering landscape with standards.

Architecture Principles and Rationalized Assumptions

1. AWS Public Cloud Platform and managed services provided by AWS or its partners via Market Place are preferred over other platform/ vendor options.
2. Data/Application Architecture must be based on: 1. Microservice, 2. Event based 3. Serverless, 4. Adoption of Managed Services (also known as PaaS) 5. Self Service 6. Continuous delivery and Integration 7. Service Delivery via API First Approach 8.Separate Storage from compute for analytics 9.Federated Identities 10. Audit-Save.
3. Architecture must be Secure, Scalable and Cost Effective (in order of preference)
4. Reusable and standard building blocks are preferred (Compose over Buy over Build).

State of Content Discovery* in 2020

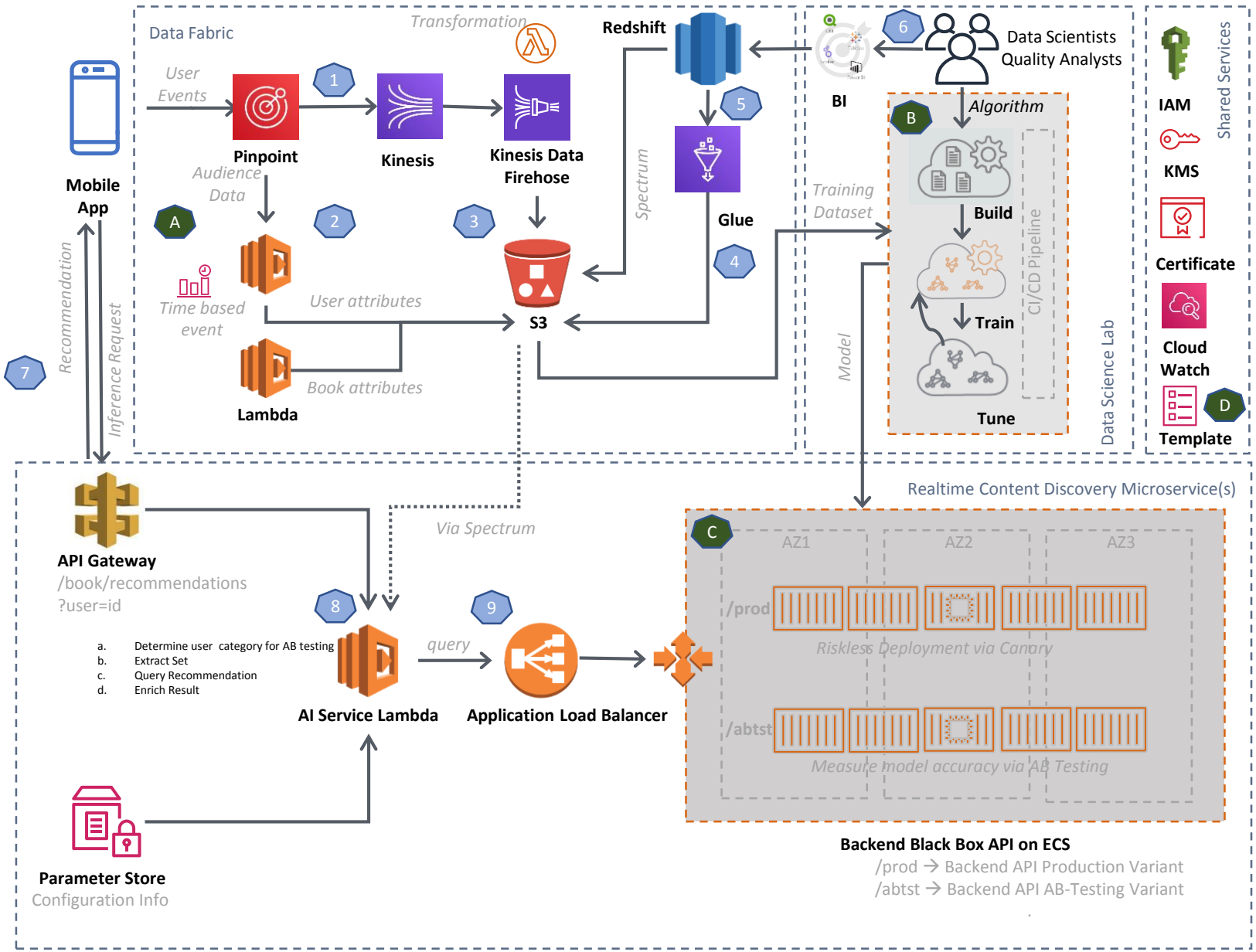
- Customer Retention >= 40%
- Premium User Conversion Rate >= 50%
- Increase App Rating to average 4-5 (5 being the highest)
- Target number of analytic tools = 8-12 with 90% standards across IT
- Percentage of IT budget spent on data management and engineering <= 5%

Sample

* Target state KPI are based on industry benchmark and market research.

Proposed Solution Architecture (Scoped within design constraints)

Automated Content Recommendation & Personalization



Assumptions:

- **A** Events created by apps are provided by a pipeline powered by Kinesis. User and Book attributes are available as JSON Rest Endpoints.
- **B** API exists to build/train/tune models. Furthermore it is also assumed that there exists toolsets to continuously integrate and deploy them as API via CI/CD pipelines.
- **C** Trained model (including all variants) are deployed as backend API on a container (ECS).
- **D** The solution adheres to all required governance and management related requirements like federated auth, data encryption both at transit/rest, endpoint verification, logging and infrastructure as code

Architecture:

1. User Engagement Events data (including custom events) are collected via existing data pipeline which streams through Kinesis.
2. User and Book attributes are collected by invoking a time based event triggered lambda which retrieves information from JSON endpoint.
3. User events, User attributes and Book attributes are stored in S3 in its native format to serve data science use cases. S3 serves as “Data Hub”
4. Glue data crawlers (supports CSV, JSON, Parquet, etc) discover S3 data schema and records into “Data Catalogue”. Glue ETL provides a serverless spark environment to curate data for training the models.
5. Glue ETL can be used to load into/unload data from Redshift. Glue Data Catalogue supports ad-hoc querying via Redshift Spectrum. Together they server as the “Ware house” for business intelligence.
6. Data Scientists can use the data in Redshift Spectrum/S3 to perform feature set engineering. Quality Engineers can benchmark A/B Testing metrics on performance of recommendation/personalization models.
7. The recommendation/personalization models are exposed to mobile app server via API GW to query real-time events and receive inferences.
8. Lambda acts as an integration layer to connect front end API with back end hosted endpoints of models. It also stitches/translates input data with mapping data (features) to query backend service and enriches the result back to mobile devices.
9. Backend hosted services routed via load balancer to production variant or A/B test variant of the model based on the path. Lambda utilized query parameters to derive the path. The backend hosted endpoints are deployed on multiple AZs with auto scaling enabled.

References:

1) Stream based mobile and web event tracking backed by aws kinesis by Sebastian Schleicher, VP Engineering & Architecture at Blinkist
2) Icon Reference: https://aws.amazon.com/architecture/icons/

Architecture Tradeoff Analysis Method

Automated Content Recommendation & Personalization

Based in attributes/dimensions in slide 7

1. **Solution Composition:** The solution is composed using managed services from AWS hence avoiding any heavy lifting, operations, maintenance tasks.
2. **Support for Standardization:** All building blocks of the solution are considered industry standards in the respective capability. For example, S3 for cheap, highly scalable, available, secure object storage. Same can be said for Kinesis, Glue, Redshift, APIGW, Lambda etc. Most importantly, no customer specific adaptations in these components.
3. **Cost**: All technology components are pay per use, hence no upfront investments required. The technology components also provide various variants that can provide cost effective solution depending on the use case. For example S3 storage classes, Kinesis Firehose vs Data Streams
3. **Availability/Flexibility/Security:** Kinesis, S3, Glue, Redshift, APIGW, Lambda, ECS are inherently built as highly available, automatically scaled and extremely secure (IAM, Data Encryption).
4. **Operational Efficiency:** The solution uses AWS management, governance technologies such Config Rules, Cloud watch which eases operations and low cost high efficient system. The system is designed as infrastructure as code that natively supports devOps.
5. **Performance:** Recommendation/Personalization models can be benchmarked with variants and continuously improved with risk-free canary deployment model.

1. **Cost:** Technologies like Kinesis, Redshift if not properly designed and used can result in high costs. This is not the problem of the technology but architecture & governance and end user training. For example Calculations on no of shards could get tricky with higher volume of data. Another example, running some non-analytical queries on Redshift.
2. **Operational Efficiency:** Recommendation/Personalization model has become a commodity in the market: there are possibility to adopt industry standard and reduce operational tasks (like Amazon Personalize, Amazon Sagemaker) rather than deploying your model on a container service
3. **Performance:** Spectrum performance needs to be benchmarked for real time query. Reorganizing data from JSON to PARQUET will help performance but it adds another pipeline in the overall architecture.
4. **Service Limits:** Most AWS services have service limits to bear in mind. For example, Kinesis has some limits in data size and data retention but can live with it.

Limitations of the Proposed Architecture and Future Enhancements

Automated Content Recommendation & Personalization

Problems / Limitations:

1. Over Engineering? (Hope not ☺)

2. Recommendation/Personalization engine **is not fully managed**. Bringing your own model and deploying on our own runtime image means we do the heavy lifting in terms of deployment.

3. Using **multiple tools to collect events** from different channels (Pinpoint, PIWIK, etc) means extra efforts to standardize and be cohesive between Channels (browser vs app). Same for distributions of data.

4. **Redshift** Spectrum separates storage from compute, the core redshift engine is not designed to be cloud native or handle JSON type of non-columnar data types. Plus it brings additonal overhead in terms of maintenance.

5. Data wrangling by data scientists often require libraries and capacities not met by **Glue**.

Improvements:

1. Revisit the architecture after reinvent: 2019

2. I would try Amazon Personalize if there was not already models developed by our scientists. It integrates well with Pinpoint as well. Amazon **Sagemaker** would perfectly fit (either bring algorithm or trained model) to host the models on AWS.

3. **Segment** IO* seems to be widely accepted across many ecommerce vendors to collect events across multiple channels and distribute to multiple destinations in standardized manner.

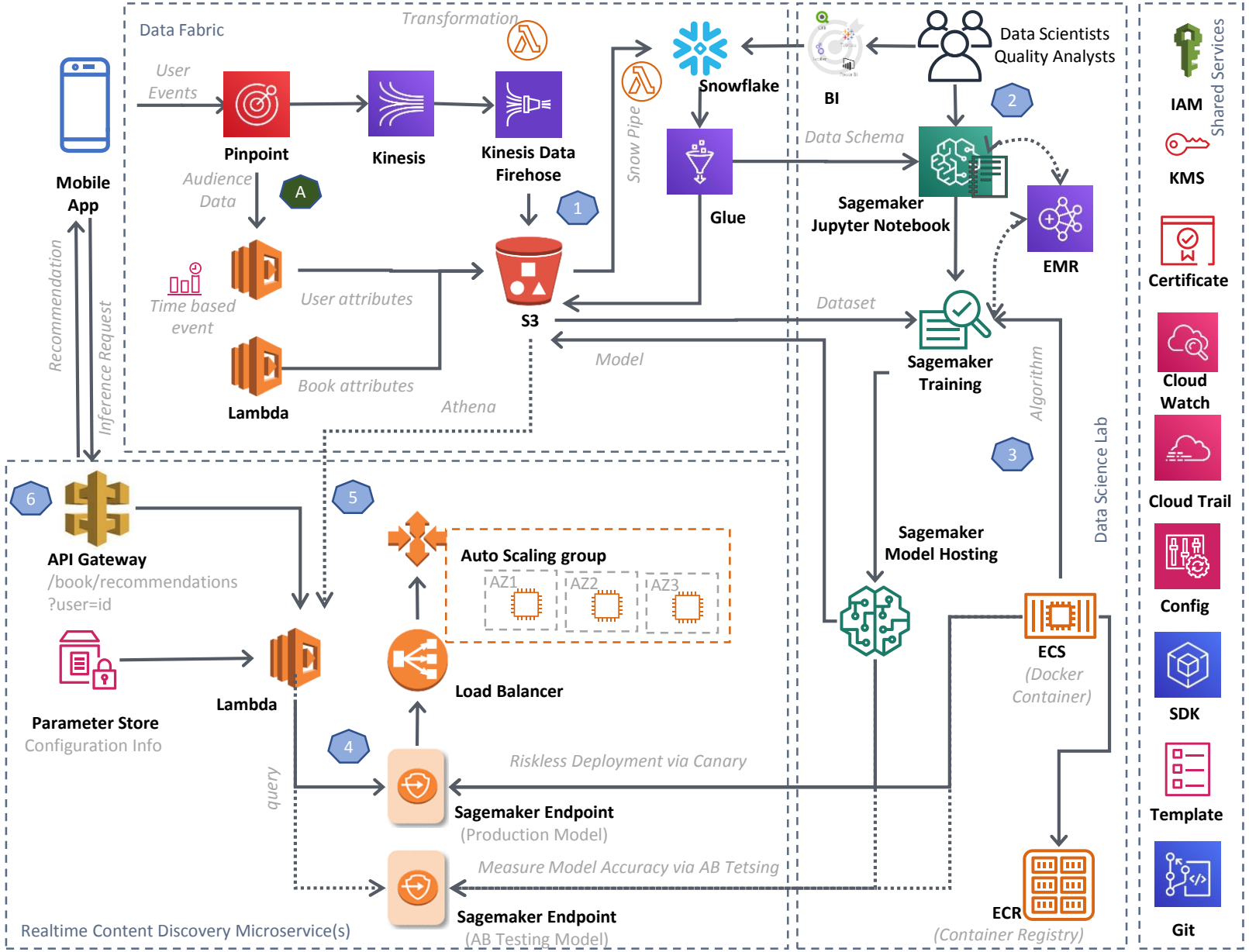
4. **Snowflake** is cloud native data warehouse that separates storage from compute. Can be scaled up and down near real-time. No overhead tasks in terms of design, maintenance, performance management etc.

5. Integrating an **elastic spark environment** enhances performance model training as well data preparation.

** Have not used in production but received great feedback from industry partners/peers*

Target State Solution Architecture

Automated Content Recommendation & Personalization



Assumptions:

A — Event collection via Pinpoint is assumed to be working well and there is no need for standardization across channels. But in case interested in Segment IO, target architecture for data fabric would look like in Appendix B.

Architecture:

1 — Considering limitations/pain points of Redshift, real-time event data reporting would delivered via Snowflake. Data would be loaded into Snowflake near real-time via Snow Pipe (if no other ETL tool)

2 — Data Science Team would use Sagemaker as their platform to build their models and deploy them intelligently. Juoyter notebooks would serve as the IDE that integrates well with S3/Glue to do data preparation. If needed, EMR cluster can be created with spot instances for more specific data wrangling and training cases.

3 — Sagemaker provides multiple options to deliver machine learning use cases into production ready solutions:

1. You can use built in algorithm, train, tune and host endpoints.
2. You can bring in your own algorithm and train on sagemaker.
3. You can bring your own trained model, store in S3 and host on sagemaker.

4 — Sagemaker provides endpoints that are highly available, auto scalable, secure. Deployments can be made real-time continuously without downtime. You can choose to go with Canary deployment or AB testing or both completely risk-free.

5 — Lambda integration layer will enrich feature for real-time query as well enhance output result using Athena. It can query endpoints for different cases based on parameter store configurations and stitch results together.

6 — API GW offers offers caching, throttling (free vs premium users), authorization and easy integration with 3rd party via developer portals or SDK.

References:

1) AWS re:Invent 2018: Build & Deploy ML Models Quickly & Easily with Amazon SageMaker (AIM404-R)
2) Building, Training and Deploying Custom Algorithms Such as Fast.ai with Amazon Sagemaker
2) Call an Amazon SageMaker model endpoint using Amazon API Gateway and AWS Lambda By Rumi Olsen
4) Disney Case Study in re:Invent 2018
5) Icon Reference: https://aws.amazon.com/architecture/icons/

Appendix A

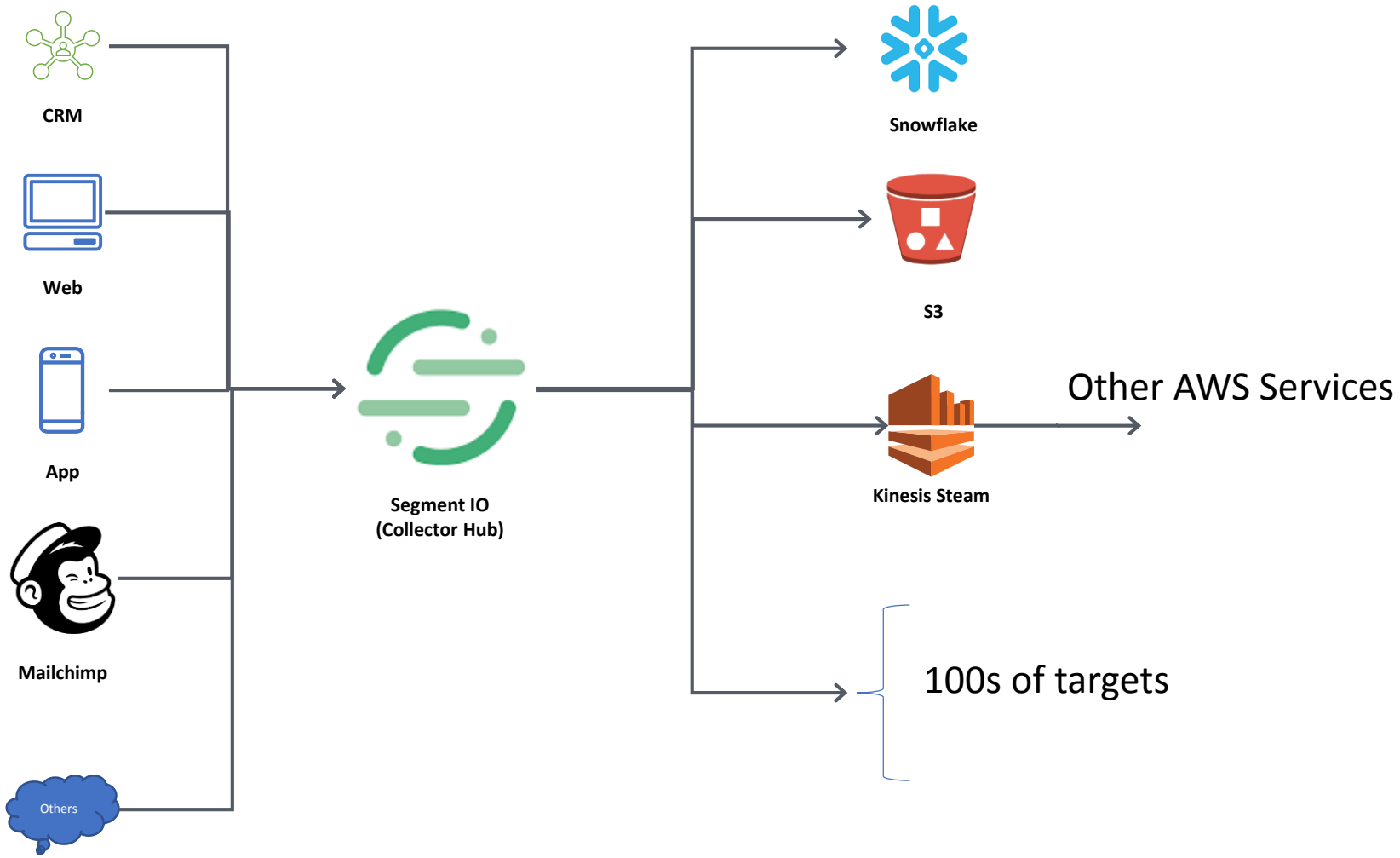
Credits:

Methodology adapted from Carnegie Mellon University Software Architecture.

https://resources.sei.cmu.edu/library/asset-view.cfm?assetid=5177

Valuation Attributes	Dimension	Description
A. Solution Composition	Landscape Impact	**COMPOSE:** use existing building block that exists in application landscape **BUY:** Commodity of the Shelf. **BUILD:** Home grown solution.
B. Impact on Landscape	Landscape Impact	**Complexity**: measured as the number of additional components introduced into the landscape **Integration**: the amount of changes to interfaces required by the solution **Decommissioning**: the number of existing components that can be decommissioned using the solution **Rationalization**: the potential for streamlining the landscape in alignment with business goals and capabilities **Redundancy**: indicates how many components support similar functions like the proposed solution **Risk**: measure on technology and business risk imposed by the solution
C. Support of Standardization	Landscape Impact	**Standardization**: measure to what extent the solution is composed of standard building blocks **Customization**: degree of customer-specific adaptations needed to achieve required business value
D. Implementation Cost Efficiency	Cost	Measured as the **estimated price of implementing a target architecture or solution**. Includes all non-labor costs (i.e., capital expenditures), and all labor costs (i.e. costs associated with staffing for all activities related to the planning, actual implementation, and rollout of the technology).
E. Operational Cost Efficiency	Cost	Measured as the **estimated price of running the IT systems based on a target architecture or solution on an ongoing basis**. Includes all non-labor costs (i.e., upgrades to hardware assets), all labor costs, (i.e., people to support those assets), and all other costs associated with applying operations (i.e., cost of network operations due to heavy payload).
F. Savings on Future Solutions	Cost	Estimated as the **reduction in cost of implementation and deployment of future solutions** based on the proposed target architecture compared to the current state. In theory, once the technology is implemented, future applications using it could require more or less effort to develop and would be delivered at a higher or lower cost.
G. Performance	Architecture Quality	**Managability**: how easy it is for administrators to manage the system **Throughput/Latency**: number of events within a given time **Bandwidth:** estimated network performance or special QoS required by the solution (i.e., for SaaS components).
H. Availability	Architecture Quality	**Downtime**: time system not available (scheduled + unscheduled maintenance, network or load issues) **Capacity**: peak capacity of the technology with appropriate variable (users, transactions, amount of data) **Utilization**: "normal" load divided by peak capacity **Recoverability**: ability to respond to data or system failure
I. Flexibility	Architecture Quality	**Maintainability**: ability to change, modify components of the system or add functionality **Scalability**: amount of time needed to increase or decrease capacity **Reusability**: number of system components which are available for reuse in other systems **Testability**: how easy is it to create and execute test criteria for the system
K. Security	Architecture Quality	Measured as the ability of a target architecture or individual solution to prevent malicious or accidental actions outside of the designed usage, and to prevent disclosure or loss of information.

Appendix B


CRM
Web
App
Mailchimp
Others
Segment IO
(Collector Hub)
Snowflake
S3
Kinesis Steam
Other AWS Services
100s of targets