Build a Location-Aware Recommendation Engine

Power your Apps with AWS Managed Services

Nate Slater, AWS Solution Architect

Wednesday January 20, 2016

amazon web services | Pop-up Loft

Agenda

- Overview of Key Concepts
- Review App Requirements
- Data Modeling
- Data Collection and Storage
- Machine Learning
- Demo

Overview of Key Concepts

Location Awareness

- A ”location aware” app tailors it’s content to be relevant for a given location.
- Location is derived from the geo-coordinates in the mobile device or web browser.
- Location data is used to filter search results to those that have proximity to the location:
    - Local shops, merchants, or attractions
    - News and events
- Location data also used to tag photos or social media posts so that you can search for content by location:
    - ”Show me the photos I took at dinner in San Francisco”

Recommendation Engine

- A recommendation engine supplies content to the app that it thinks the user will like.
- Recommendations can be generated algorithmically:

```
if (time == 12:00PM) {
    displayLocalRestaurants();
}
```

- Recommendations can be generated by a predictive model:

```
{gender: "m", age: "18-25", budget: "$10-20",
restaurant: "Taco Time"}
{predictedClass: "3_star", probability:
"0.478"}
```

App Requirements

App Requirements – Search and Location

- Users should be able to search for local businesses on their mobile device. Required search terms:
  - Name
  - Address
  - Description
  - Keyword (tacos, beer, flowers, etc.)

App Requirements – Search and Location

- Search results should be filtered by location and displayed on a map.
- Only show results on the rectangular map with geo-coordinates between:

  (x,y) top right

  (x,y) bottom left

App Requirements - Recommendations

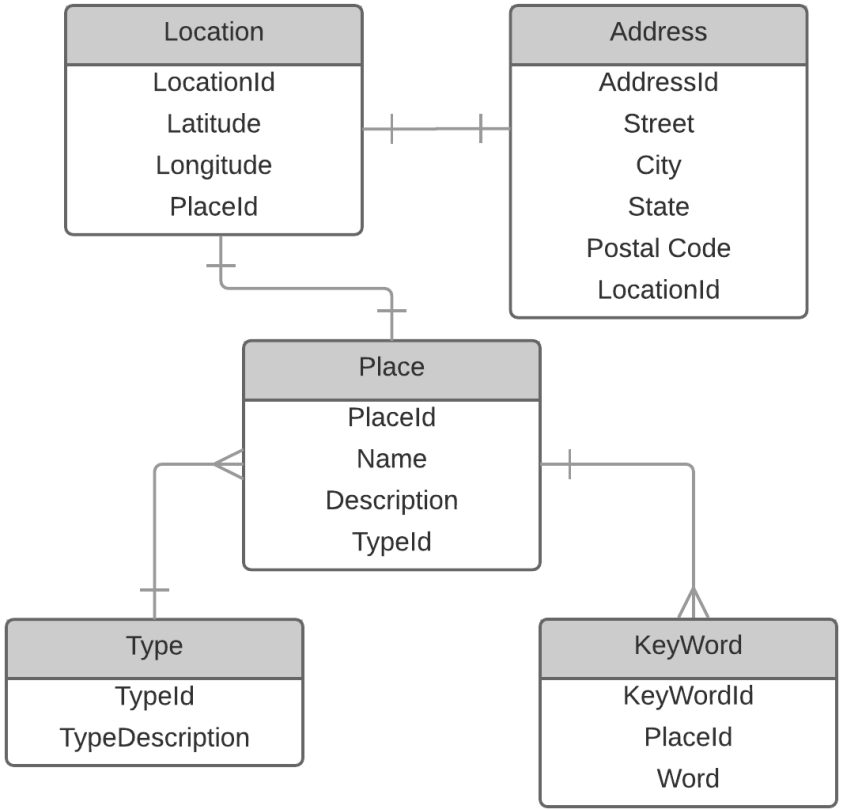
- Recommendations should be based on a predictive model.
- User registration form will collect minimal information:
  - Email Address
  - Gender
  - Age Group (under 18, 18-25, over 65, etc.)
  - Postal Code
- Recommendations will be offers from local merchants.
- Must be able to track user response to offers:
  - Click-through
  - Conversions

Data Modeling

Characterize the Data

- Location data:
  - Structured
  - Mutable
  - One-to-many relationships (e.g. keywords)
- Click-stream data:
  - Semi-structured (JSON)
  - Immutable

Location Data – ER Model



Location Data – Document Model

```
{    ”keywords": ["seafood"],
     "name": "Fog Harbor Fish House",
     "location": {
          "city": "San Francisco",
          "geocoordinate": {
               "lat": 37.8090391877147,
               "lon": -122.410291436563
          },
          "state": "CA",
          "postal_code": "94133",
          "address": ["Pier 39", "Ste A-202"]]
     },
     "type": ”restaurant",
     ”placeId": "a83d” }
```

ER vs Document Models

ER Model:

- Normalized
- Transactional
- Search requires indexing all attributes
- Adding new attributes requires schema change

Document Model:

- Non-Normalized
- Attributes can be added without schema change
- Entire document can be indexed for search

Data Collection and Storage

Types of Data

Transactional

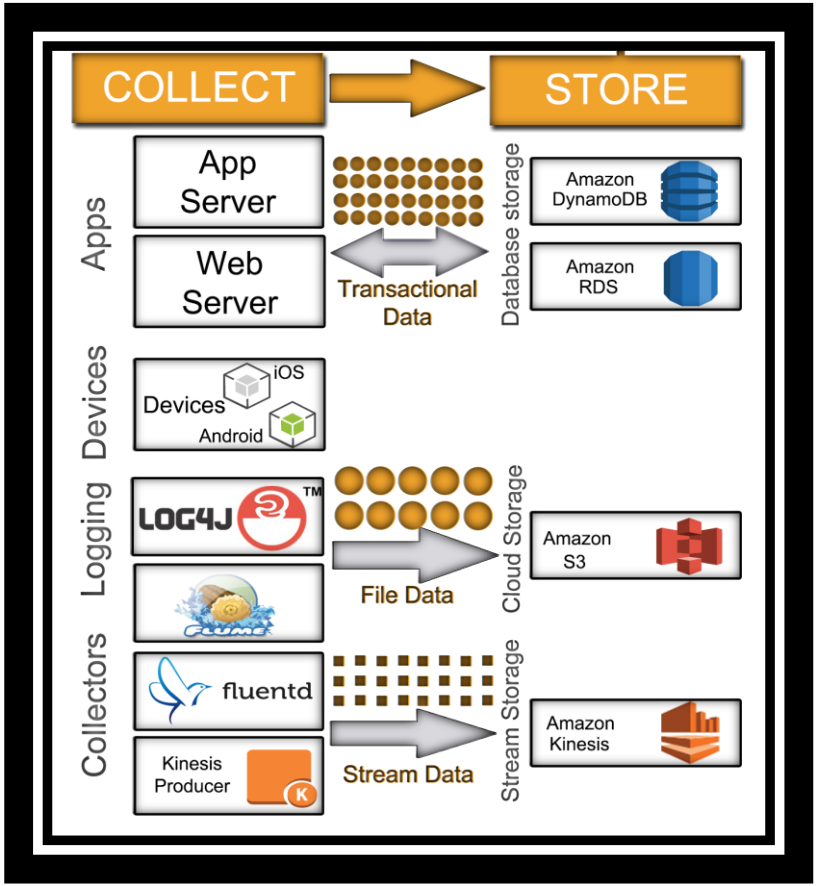
- OLTP
- Document

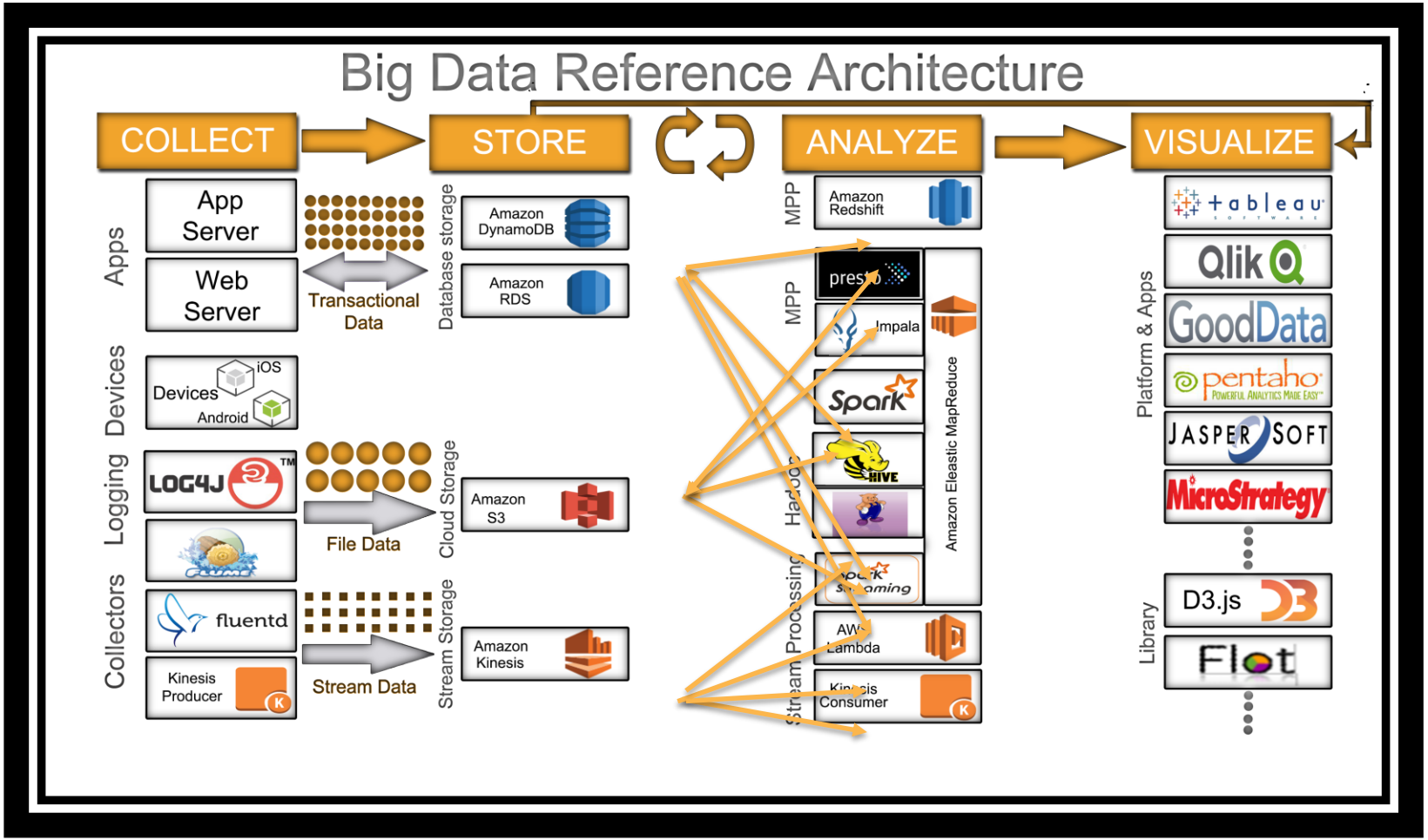
File

- Logs

Stream

- IoT
- Click-stream




Big Data Reference Architecture
COLLECT
STORE
ANALYZE
VISUALIZE
Apps
App Server
Web Server
Devices
Devices
iOS
Android
Logging
LOG4J
Collectors
fluentd
Kinesis Producer
Transactional Data
File Data
Stream Data
Database storage
Amazon DynamoDB
Amazon RDS
Cloud Storage
Amazon S3
Stream Storage
Amazon Kinesis
MPP
Amazon Redshift
MPP
presto
Impala
Spark
HIVE
Hadoop
Stream Processing
Amazon Kinesis Consumer
Amazon Elastic MapReduce
Platform & Apps
tableau
Qlik
GoodData
pentaho
JASPERSOFT
MicroStrategy
Library
D3.js
Flot

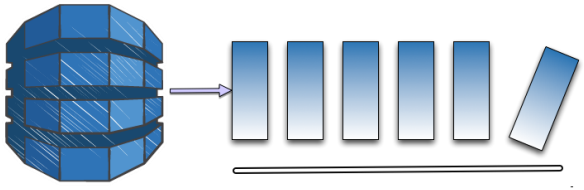
Why Transactional Data Storage?

- High throughput
- Read, Write, Update intensive
- Thousands or Millions of Concurrent interactions
- Availability, Speed, Recoverability

Amazon DynamoDB

- Managed NoSQL database service
- Supports both document and key-value data models
- Highly scalable – no table size or throughput limits
- Consistent, single-digit millisecond latency at any scale
- Highly available—3x replication
- Simple and powerful API

DynamoDB Streams



Stream of updates to a table

Asynchronous

Exactly once

Strictly ordered

- Per item

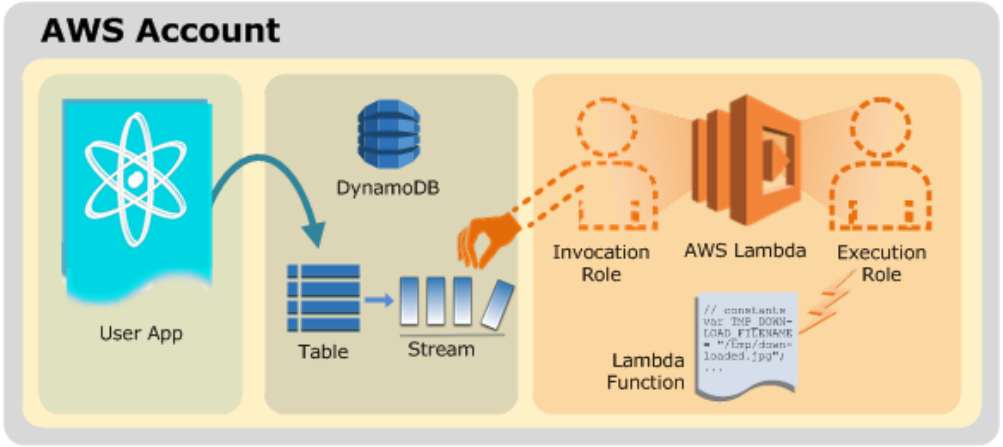
Highly durable

- Scale with table

24-hour lifetime

Sub-second latency

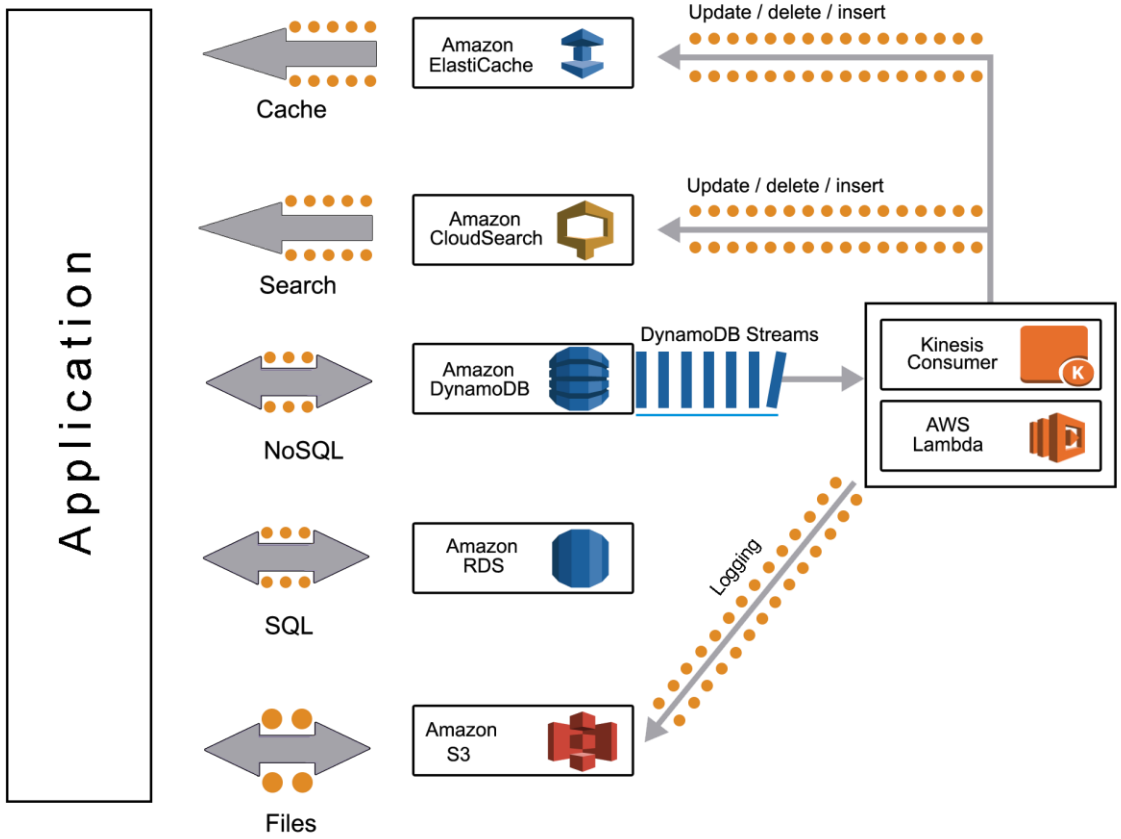
DynamoDB Streams and AWS Lambda




```
1  console.log('Loading event');
2  exports.handler = function(event, context) {
3    console.log("Event: %j", event);
4    for(i = 0; i < event.Records.length; ++i) {
5      record = event.Records[i];
6      console.log(record.EventID);
7      console.log(record.EventName);
8      console.log("DynamoDB Record: %j", record.Dynamodb);
9    }
10   context.done(null, "Hello World");  // SUCCESS with message
11 }
```

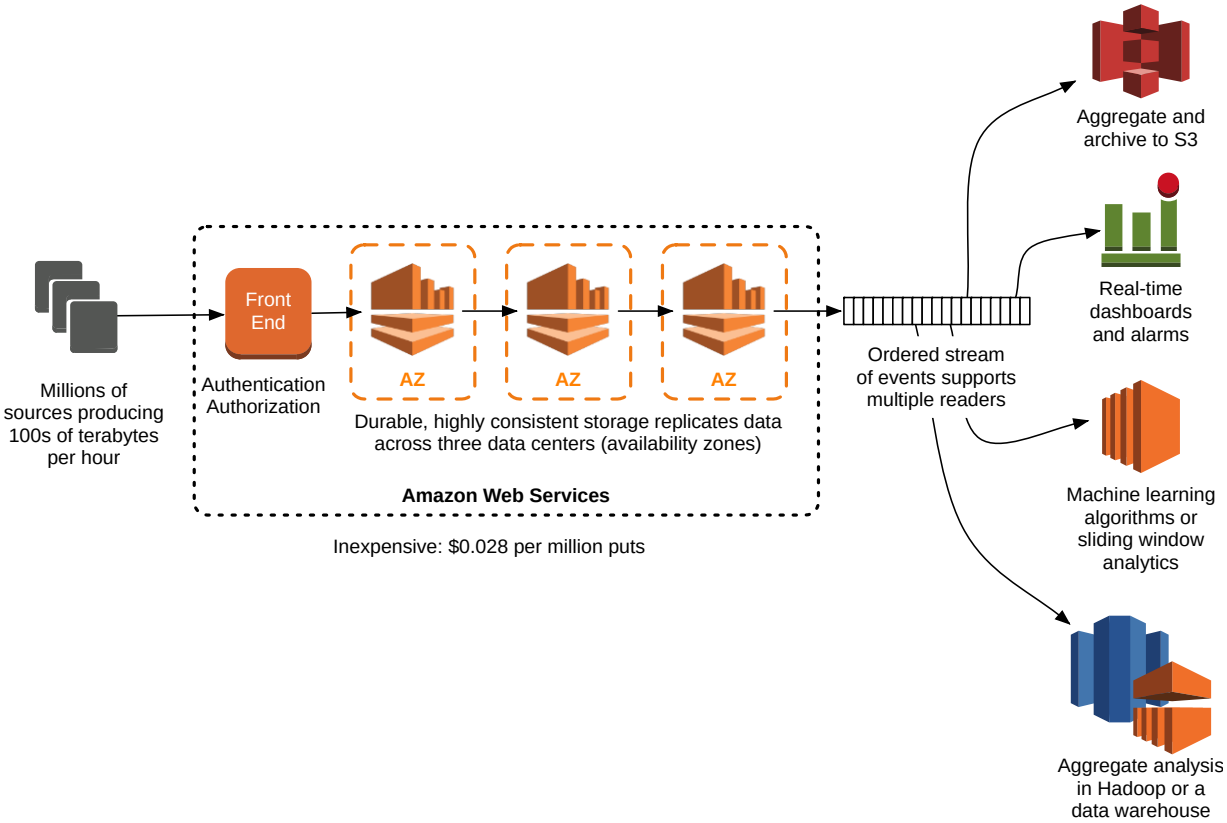
```
▸ 2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff INSERT
▸ 2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff DynamoDB Record:{ "NewImage": { "name": { "S": "sivar" }, "hk": { "S": "3" } }, "SizeBytes": 15, "StreamViewType": "NEW_AND_OLD_IMAG
▸ 2015-03-21T07:44:58.883Z 2ca3769a-cf9e-11e4-b270-ad4d24b312ff Message: "Hello World"
```

Architectural Pattern – Materialize Views on DynamoDB



Amazon Kinesis

Managed Service for streaming data ingestion, and processing



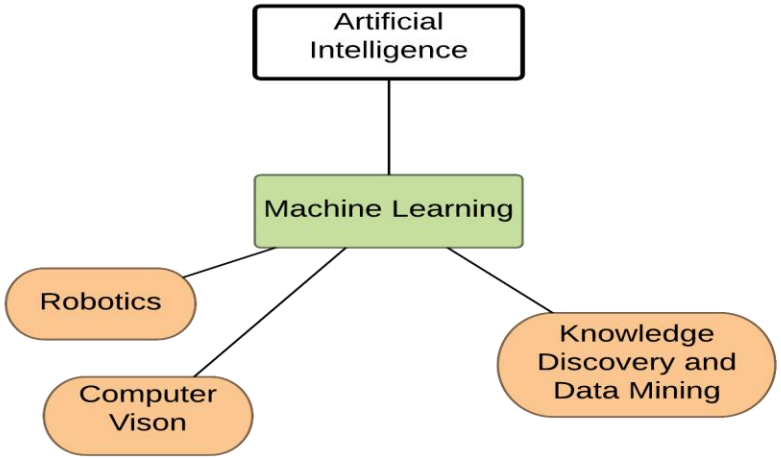
Kinesis Firehose

Makes stream processing even easier!

- Automatically delivers data to S3 and Redshift
  - No Kinesis-Client-Library or Lambda functions required
- Handles all scaling of the Kinesis stream's shards
- Near Real-time
  - Data loaded into S3 or Redshift within 60 seconds of hitting the stream
- Fully Managed
  - No operational overhead of managing streams, shards, or KCL applications

Machine Learning

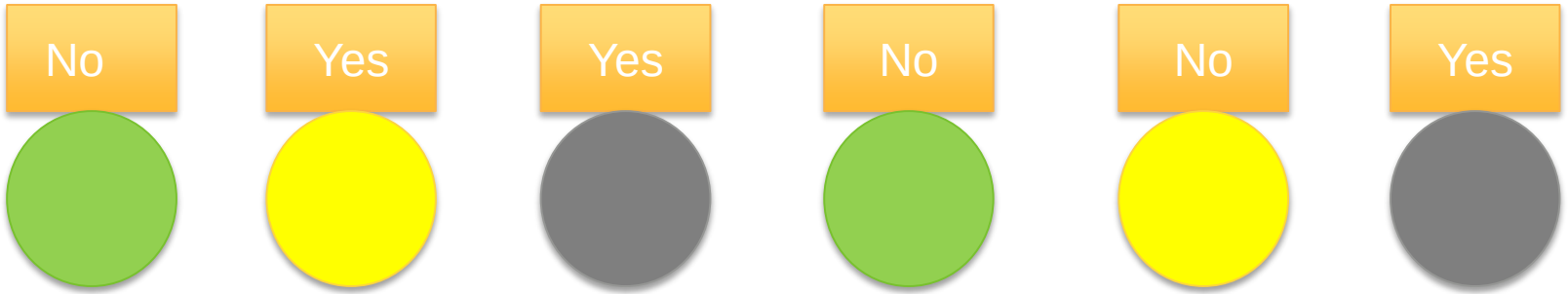
Origins of Machine Learning



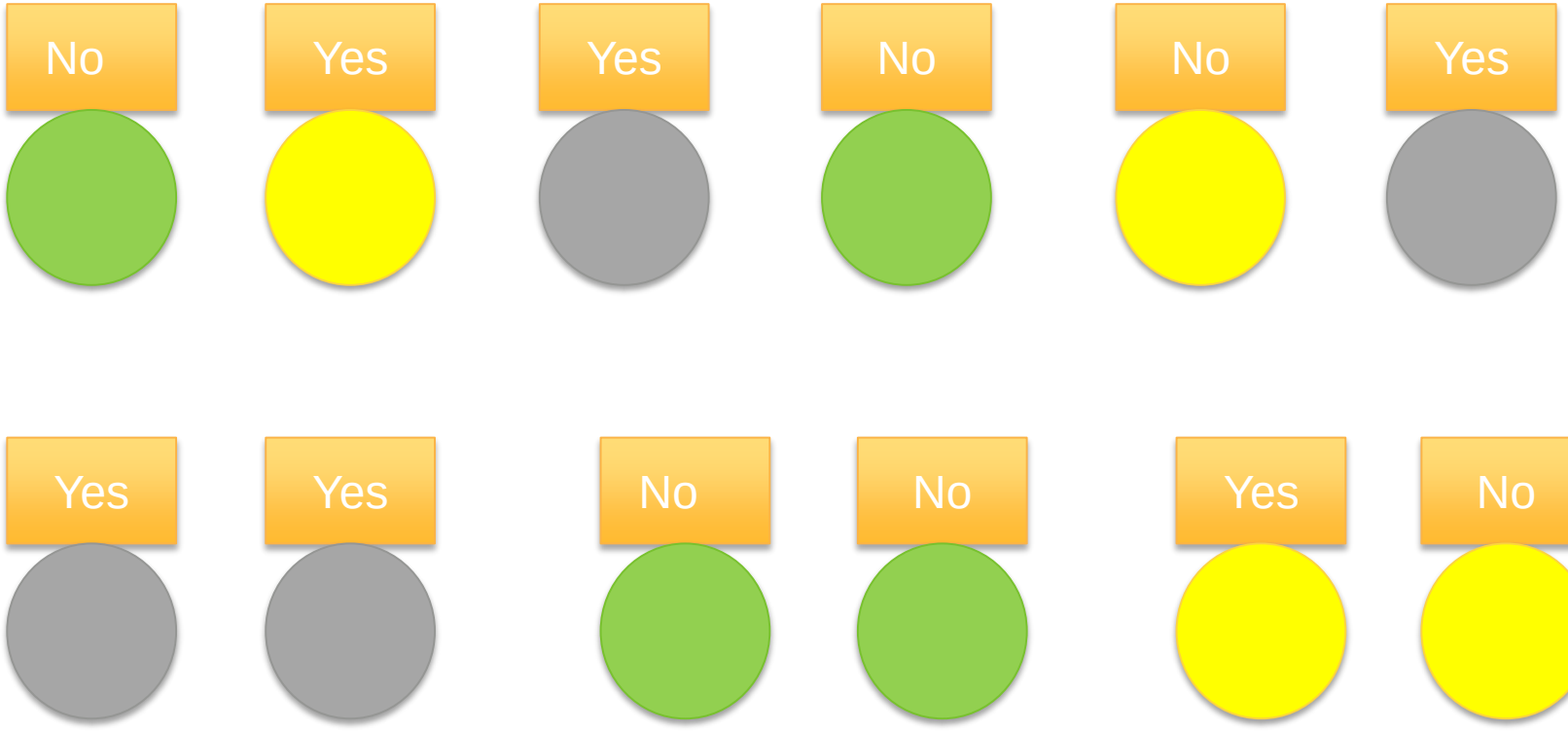
Machine Learning – Key Concepts

Segmentation:

- Divide the population into subgroups that have different values for the target variable



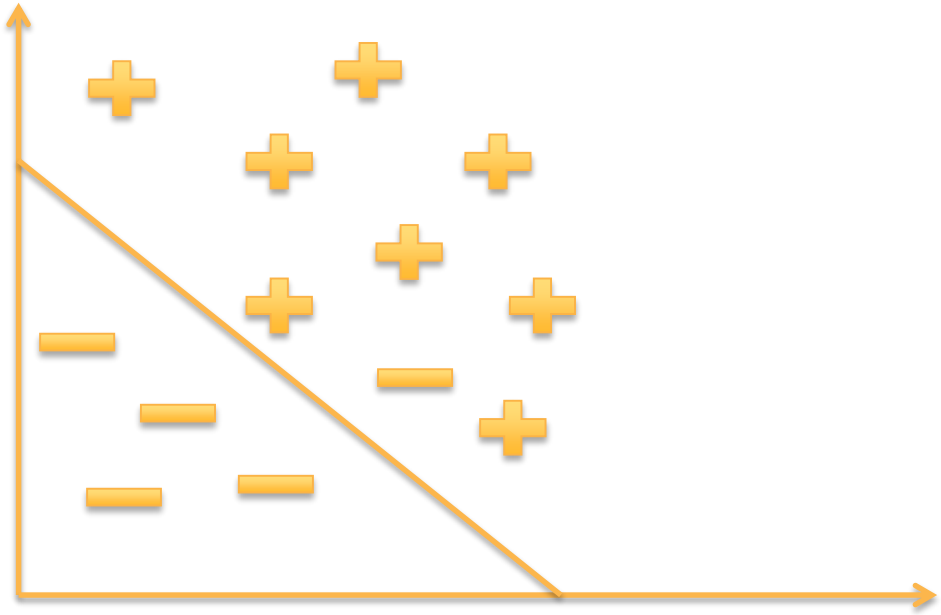
Machine Learning – Key Concepts



Machine Learning – Key Concepts

Linear Classification:

- Linear function is used to segregate the dataset



Amazon Machine Learning

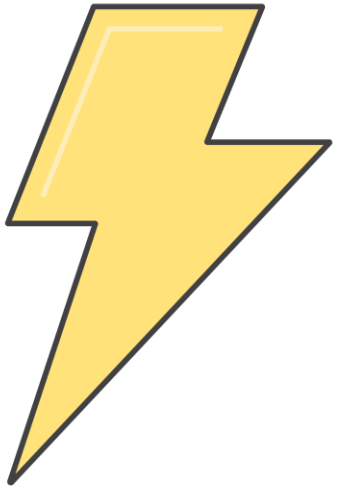
Easy to use, managed machine learning service built for developers

Robust, powerful machine learning technology based on Amazon's internal systems

Create models using your data already stored in the AWS cloud

Deploy models to production in seconds

Powerful Machine Learning Technology



Based on Amazon's battle-hardened internal systems

Not just the algorithms:

- Smart data transformations
- Input data and model quality alerts
- Built-in industry best practices

Grows with your needs

- Train on up to 100 GB of data
- Generate billions of predictions
- Obtain predictions in batches or real-time

Machine Learning Workflow

1

Build & Train model

2

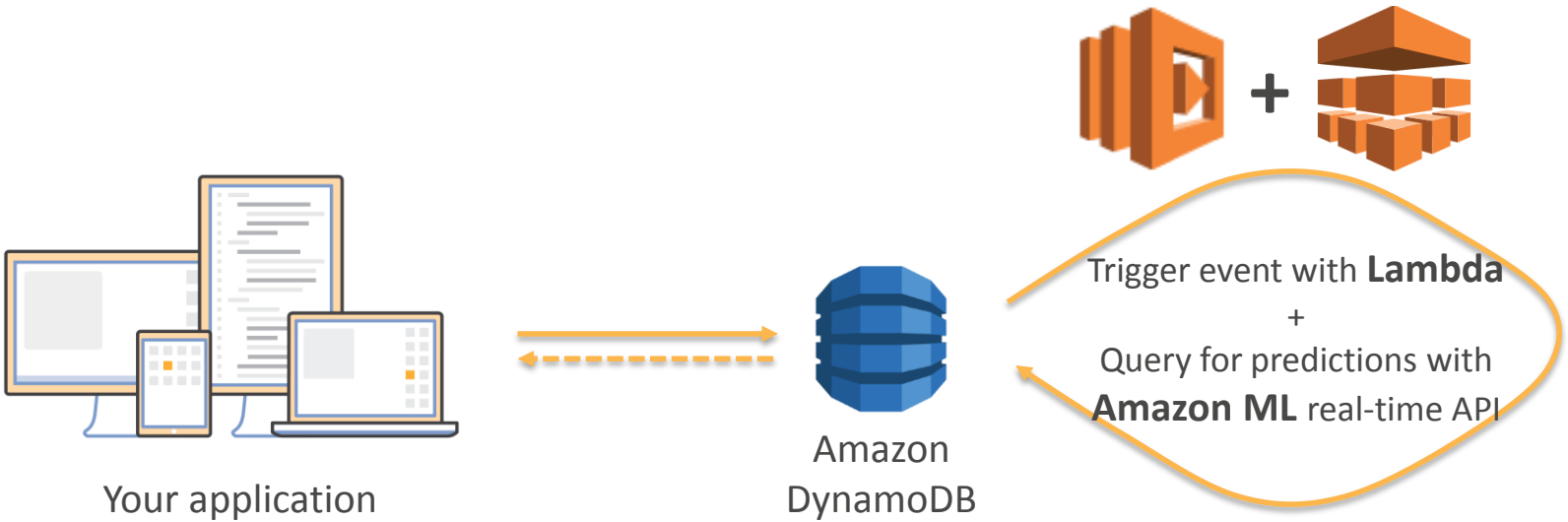
Evaluate and optimize

3

Retrieve predictions

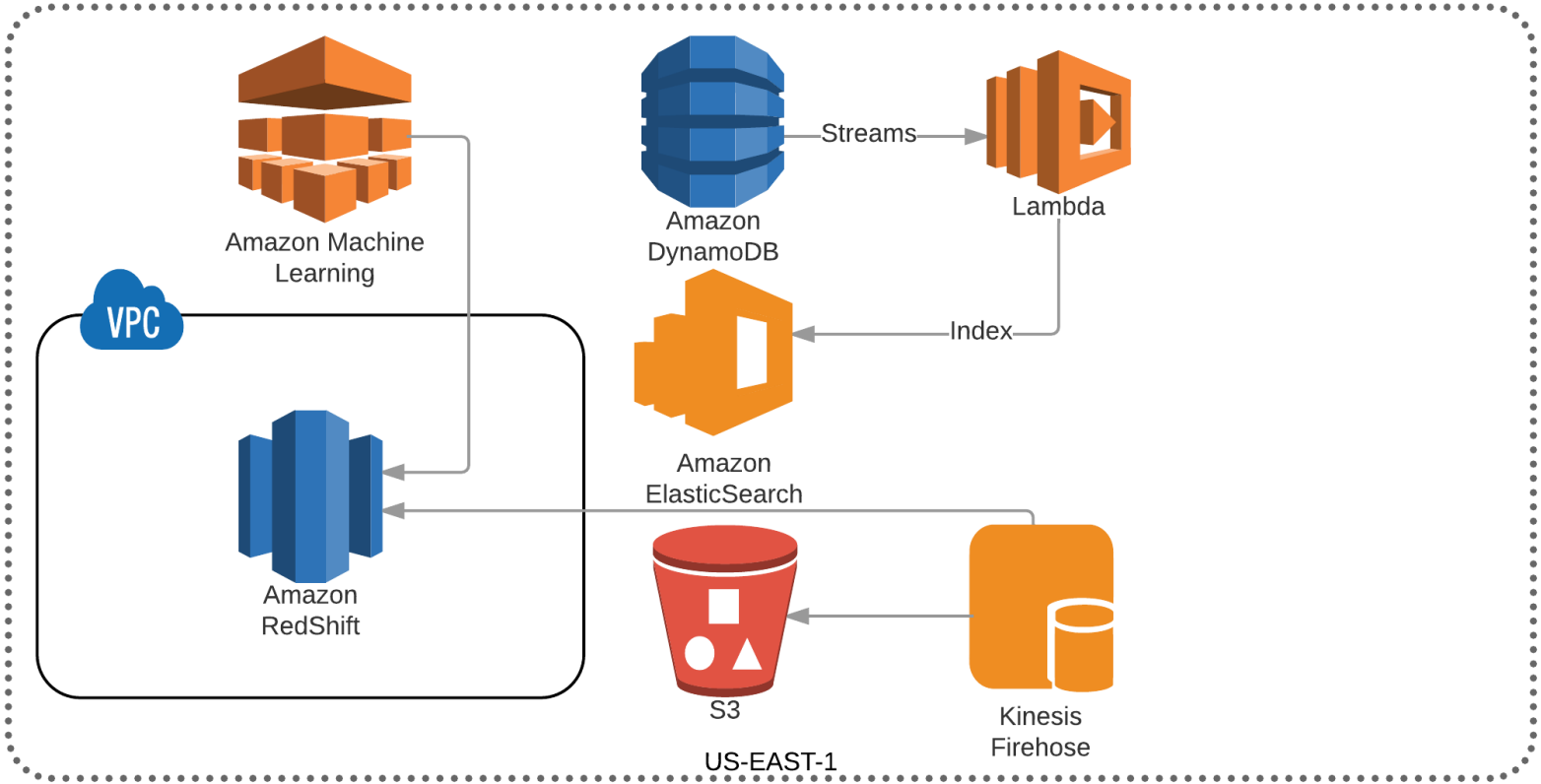
- Create a Datasource object pointing to your data
- Explore and understand your data
- Transform data and train your model

Add Predictions to Existing Data Flow



Demo

Data Architecture



Thank You!

Nate Slater, AWS Solution Architect