# AI in Java and Scala on AWS

Julien Simon

AI/ML Evangelist, EMEA

@julsimon

aws

# Agenda

- Artificial Intelligence At Amazon
- Text-to-speech: Amazon Polly
- Object and face detection: Amazon Rekognition
- Machine Learning as a service: Amazon Machine Learning
- Spark MLlib on Amazon EMR
- Apache MXNet: Deep Learning
- Resources

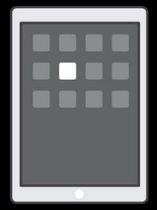
- Artificial Intelligence: design software applications which exhibit human-like behavior, e.g. speech, natural language processing, reasoning or intuition
- Machine Learning: teach machines to learn without being explicitly programmed
- Deep Learning: using neural networks, teach machines to learn from complex data where features cannot be explicitly expressed

# Artificial Intelligence At Amazon

## Thousands Of Employees Across The Company Focused on AI

Discovery & Search

Fulfilment & Logistics



Enhance Existing Products

Define New Categories Of Products

Bring Machine Learning To All

# Selected customers running AI on AWS





















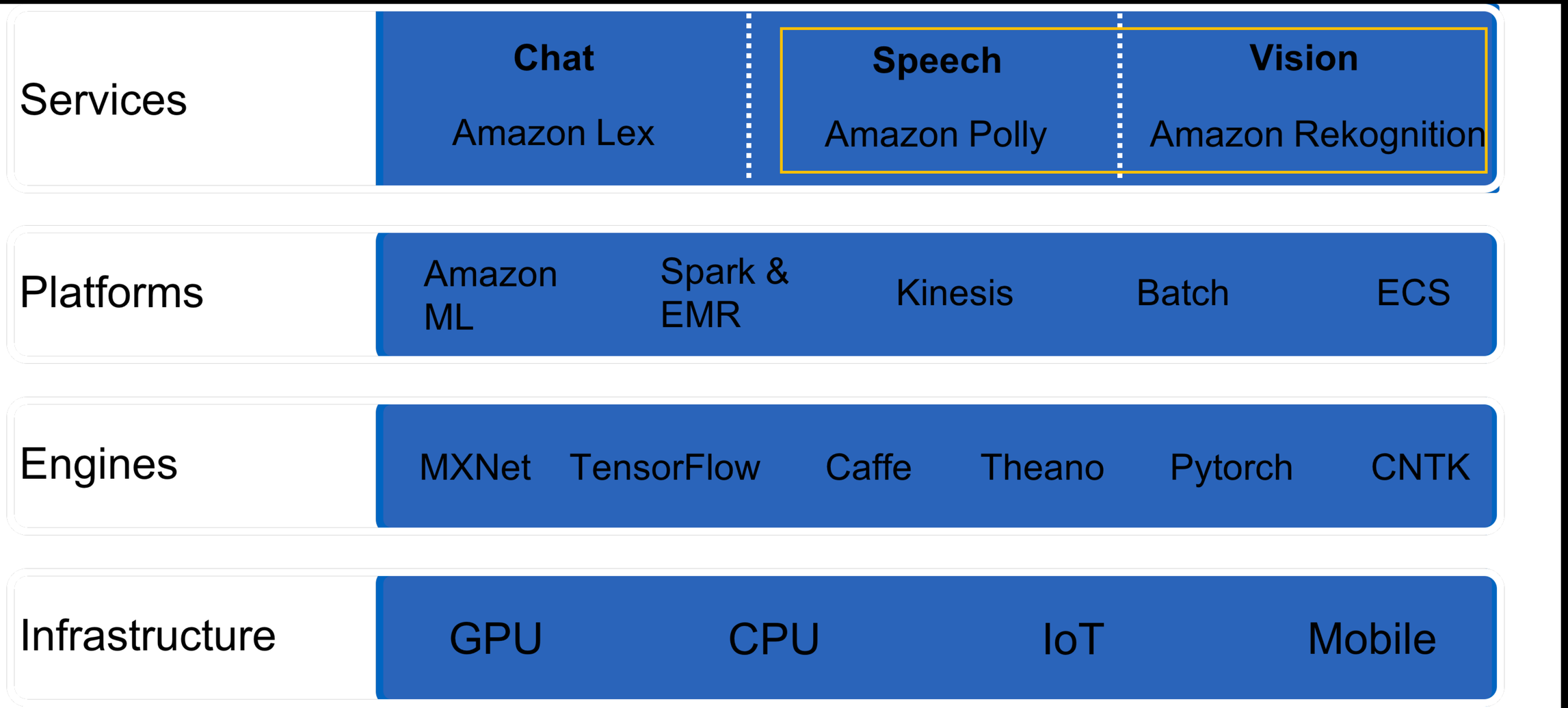









# Amazon AI for every developer



# **Polly:** Life-like Speech Service

Converts text
to life-like speech

Fully
managed

50
voices

24
languages

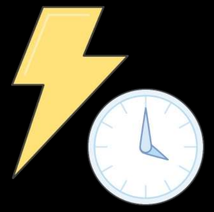
Low
latency,
real time


hue
PHILIPS
P
NEW
Wikipe
The Economist
wink
IF
we mo
amazon

# Polly: plain text & SSML text

```
AmazonPolly pollyClient = AmazonPollyClientBuilder.standard().withRegion(Regions.EU_WEST_1)
        .withCredentials(new AWSStaticCredentialsProvider(credentials)).build();

SynthesizeSpeechRequest req = new SynthesizeSpeechRequest().withVoiceId("Brian")
        .withText("Hello, My name is Brian. I'm in the kitchen.").withOutputFormat("mp3");

SynthesizeSpeechResult result = pollyClient.synthesizeSpeech(req);
play(result.getAudioStream());

String ssmlMessage = "<speak>Your reservation for <say-as interpret-as=\"cardinal\"> 2 </say-as> rooms on the "
        + "<say-as interpret-as=\"ordinal\">4th</say-as> floor of the hotel on"
        + "<say-as interpret-as=\"date\" format=\"mdy\">3/21/2012</say-as>, with early"
        + "arrival at <say-as interpret-as=\"time\" format=\"hms12\">12:35pm</say-as> has been confirmed. "
        + "Please call <say-as interpret-as=\"telephone\" format=\"1\">(888) 555-1212</say-as> with any questions.</speak>";

req = new SynthesizeSpeechRequest().withVoiceId("Amy").withTextType("ssml").withText(ssmlMessage)
        .withOutputFormat("mp3");

result = pollyClient.synthesizeSpeech(req);
play(result.getAudioStream());
```

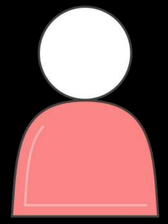
# **Rekognition:** Search & Understand Visual Content



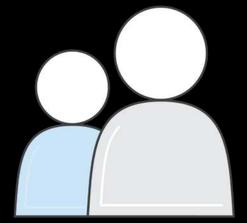
Real-time & batch image analysis

Object & Scene Detection

Facial Detection



Facial Analysis



Face Search

Visual Similarity Search

Find similar faces

Celebrity Detection

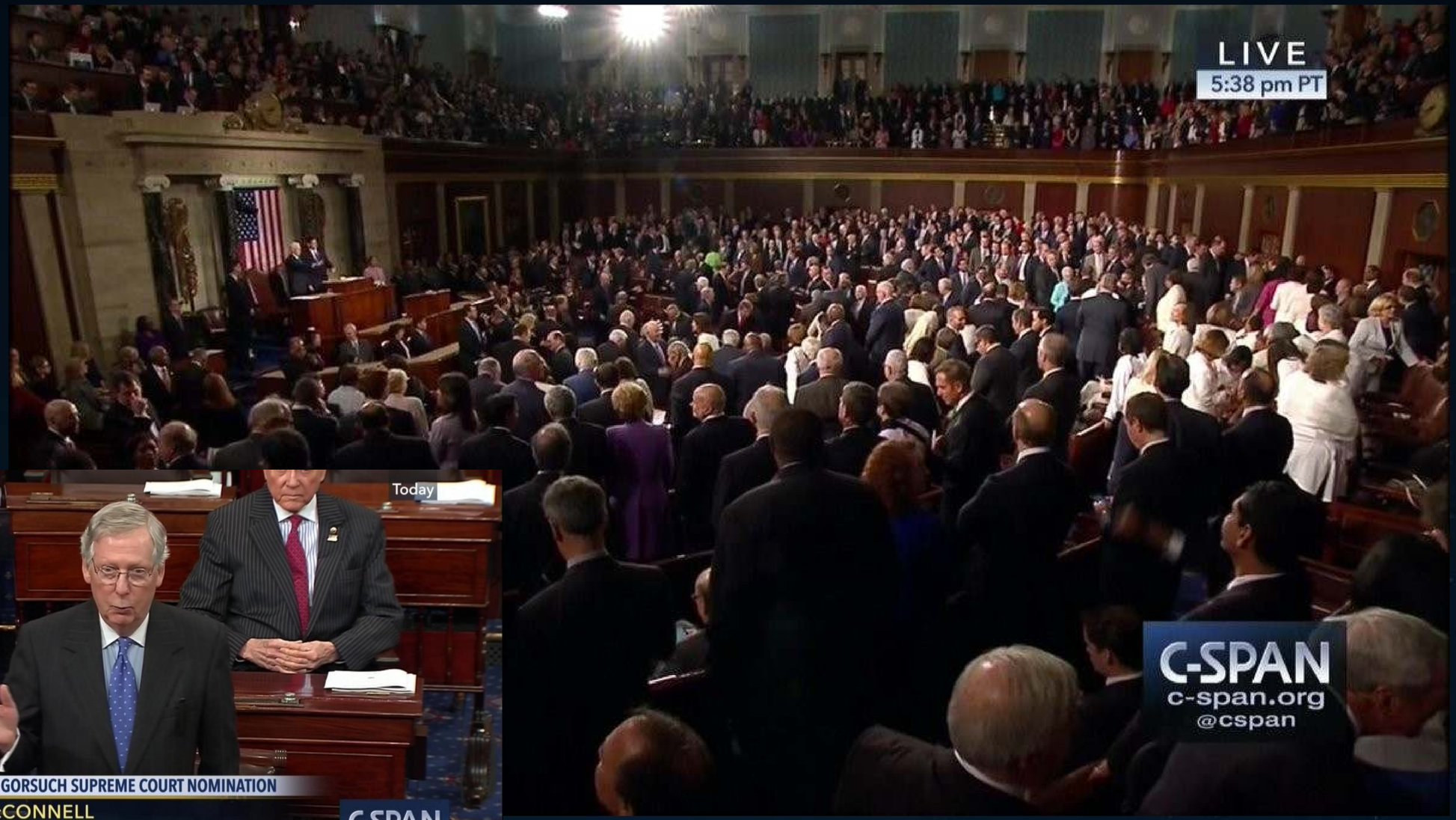
Sports, music, movies, etc.

Content Moderation

Explicit, suggestive, etc.


Text In Image

C-SPAN







https://aws.amazon.com/solutions/case-studies/cspan/

# Rekognition: object detection

```
DetectLabelsRequest request = new DetectLabelsRequest()
        .withImage(new Image()
                .withS3Object(new S3Object().withName(photo).withBucket(bucket)))
        .withMaxLabels(10).withMinConfidence(75F);

try {
    DetectLabelsResult result = rekognitionClient.detectLabels(request);
    List<Label> labels = result.getLabels();
```

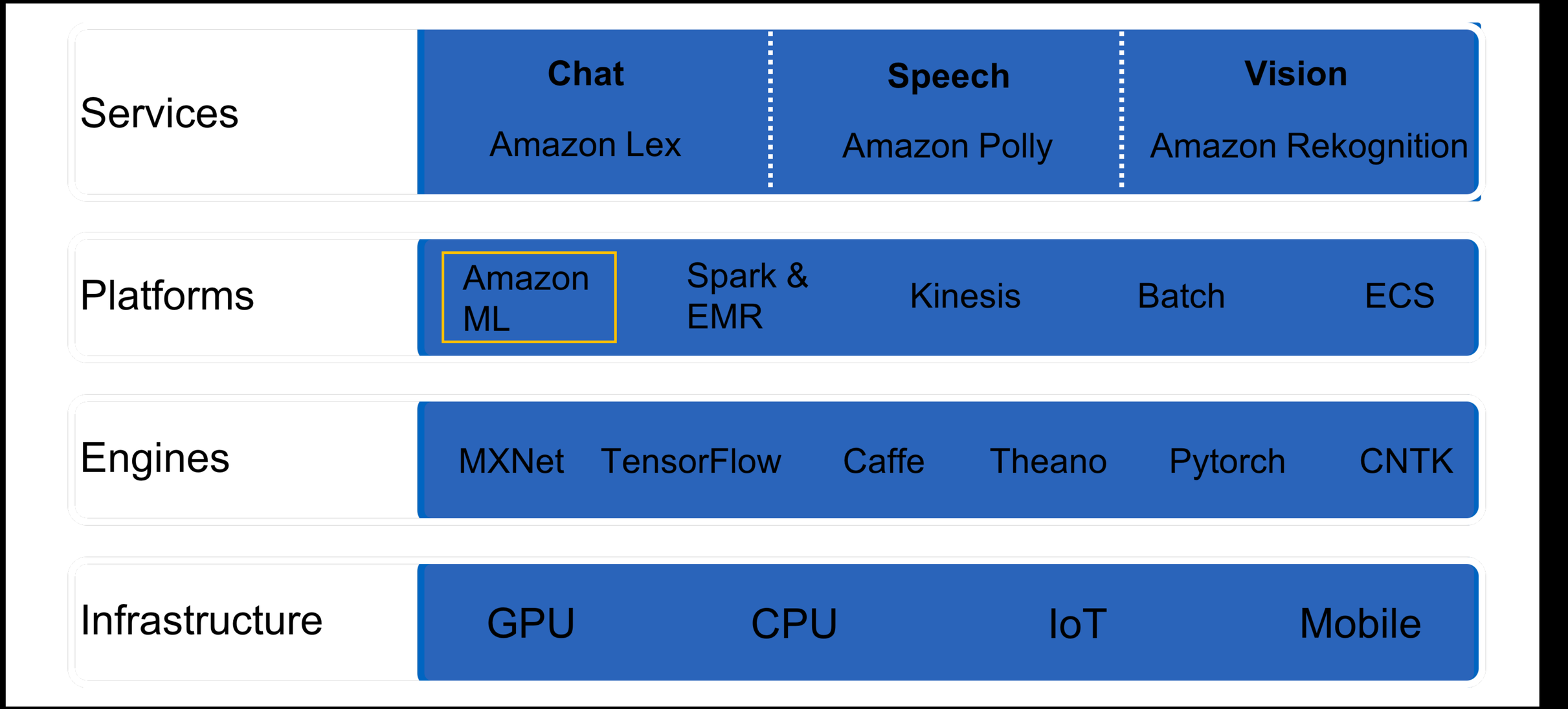
# Rekognition: face detection

```
DetectFacesRequest request = new DetectFacesRequest()
    .withImage(new Image()
        .withS3Object(new S3Object()
            .withName(photo)
            .withBucket(bucket)))
    .withAttributes(Attribute.ALL);

try {
    DetectFacesResult result = rekognitionClient.detectFaces(request);
    List < FaceDetail > faceDetails = result.getFaceDetails();
```

# Rekognition: face comparison

```
CompareFacesRequest request = new CompareFacesRequest()
        .withSourceImage(source)
        .withTargetImage(target)
        .withSimilarityThreshold(similarityThreshold);

CompareFacesResult compareFacesResult = rekognitionClient.compareFaces(request);
List<CompareFacesMatch> faceDetails = compareFacesResult.getFaceMatches();
```

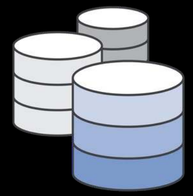
# Amazon AI for every developer



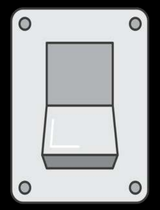
# Amazon Machine Learning

Easy-to-use, managed machine learning service built for developers

Robust, powerful technology based on Amazon's internal systems



Create regression and classification models using your data already
stored in the AWS Cloud

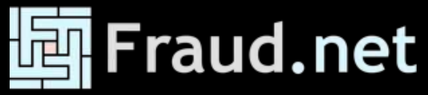


Deploy models to production in seconds

# Fraud.net Uses AWS to Quickly, Easily Detect Online Fraud

“Amazon Machine Learning helps us reduce complexity and make sense of emerging fraud patterns.”

**Oliver Clark**
CTO,
Fraud.net



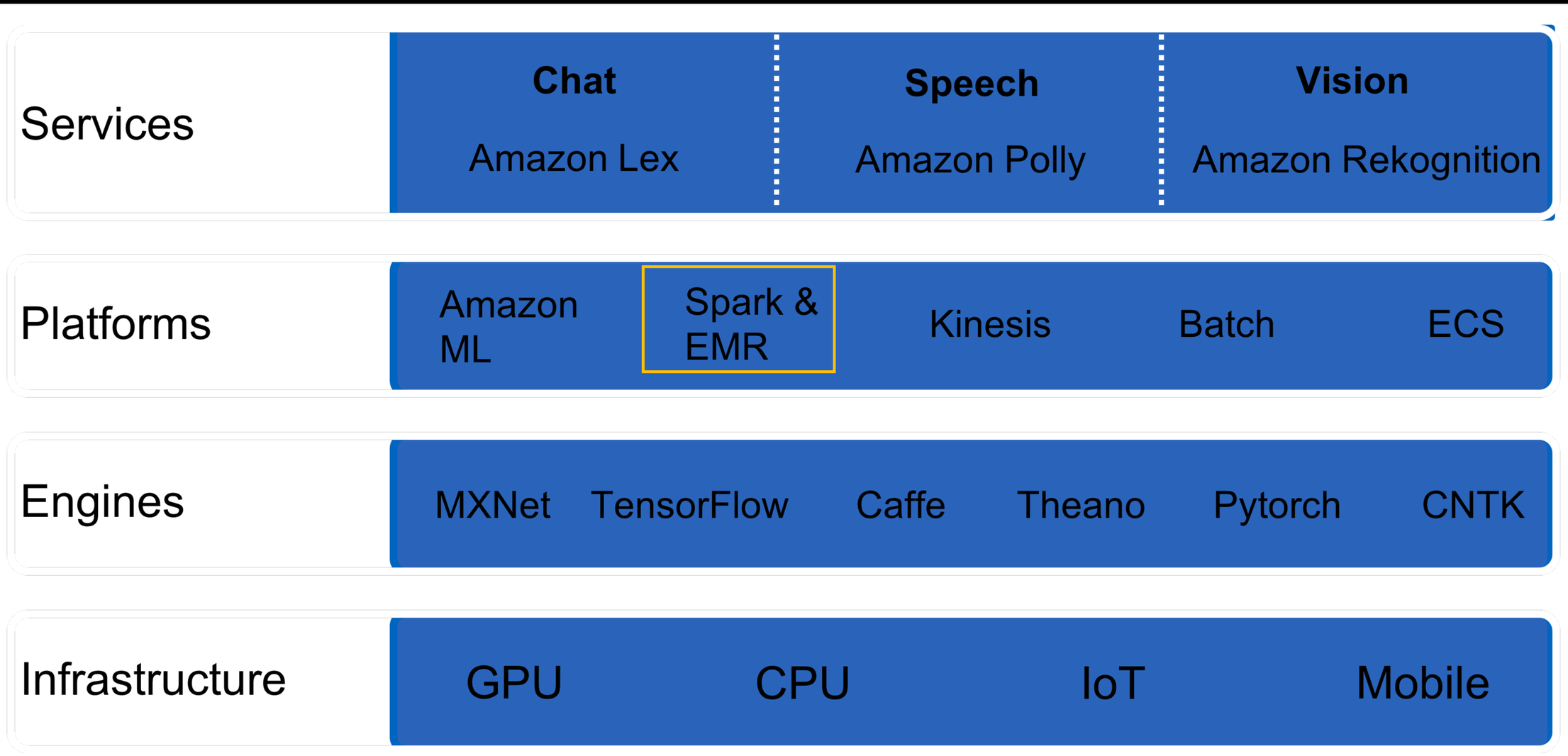

Fraud.net is the world’s leading crowdsourced fraud prevention platform.

- Needed to build and train a larger number of more targeted machine-learning models
- Uses Amazon Machine Learning to provide more than 20 models
- Easily builds and trains models to effectively detect online payment fraud
- Reduces complexity and makes sense of emerging fraud patterns
- Saves clients $1 million weekly by helping them detect and prevent fraud

# Amazon Machine Learning: real-time prediction

```
// Build a prediction request
PredictRequest request = new PredictRequest();
// Select prediction model
request.setMLModelId(model.getMLModelId());
// Select realtime endpoint
request.setPredictEndpoint(model.getEndpointInfo().getEndpointUrl());

// Build data to be predicted
request.addRecordEntry("age", "32").addRecordEntry("job", "services").addRecordEntry("marital", "divorced")
.addRecordEntry("education","basic.9y").addRecordEntry("default", "no").addRecordEntry("housing", "unknown")
.addRecordEntry("loan", "yes").addRecordEntry("contact", "cellular").addRecordEntry("month", "dec")
.addRecordEntry("day_of_week", "mon").addRecordEntry("duration", "110").addRecordEntry("campaign","1")
.addRecordEntry("pdays", "11").addRecordEntry("previous", "0").addRecordEntry("poutcome", "nonexistent")
.addRecordEntry("emp_var_rate", "-1.8").addRecordEntry("cons_price_idx", "94.465").addRecordEntry("cons_conf_idx", "-36.1")
.addRecordEntry("euribor3m", "0.883").addRecordEntry("nr_employed", "5228.1");

// Send prediction request
PredictResult result;
try {
    long start = System.currentTimeMillis();
    result = client.predict(request);
    long end = System.currentTimeMillis();
    System.out.println("Request time: " + (end - start) + " ms");
} catch (Exception e) {
    throw new AmazonClientException("Prediction failed", e);
}

// Display predicted value
System.out.println("Prediction: " + result.getPrediction());
```

# Amazon AI for every developer



# Amazon Elastic Map Reduce (EMR)

- Map Reduce, Apache Spark, Presto, etc.
- Launch a cluster in minutes
- Open source distribution or MapR distribution
- Elasticity of the cloud
- Built in security features
- Pay by the hour and save with Spot instances
- Flexibility to customize

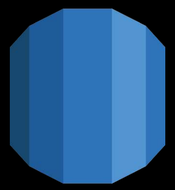
# Integration with AWS backends

Amazon DynamoDB

EMR-DynamoDB connector

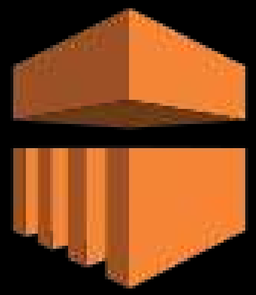
Spark-Redshift connector

Amazon Redshift



Amazon RDS

JDBC Data Source w/ Spark SQL



**Amazon EMR**

Streaming data connectors

Amazon Kinesis

Amazon ES

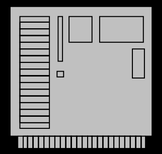
ElasticSearch connector

EMR File System (EMRFS)

Amazon S3

# Running Spark jobs on EMR

Submit a Spark application

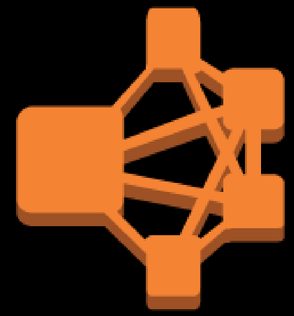


Amazon EMR Step API

Apache Zeppelin

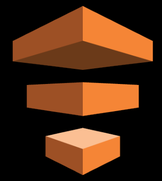
Use AWS Lambda to submit applications to EMR Step API or directly to Spark on your cluster

AWS Lambda



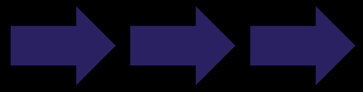
**Amazon EMR**

AWS Glue



AWS Data Pipeline

Create a pipeline to schedule job submission or create complex workflows



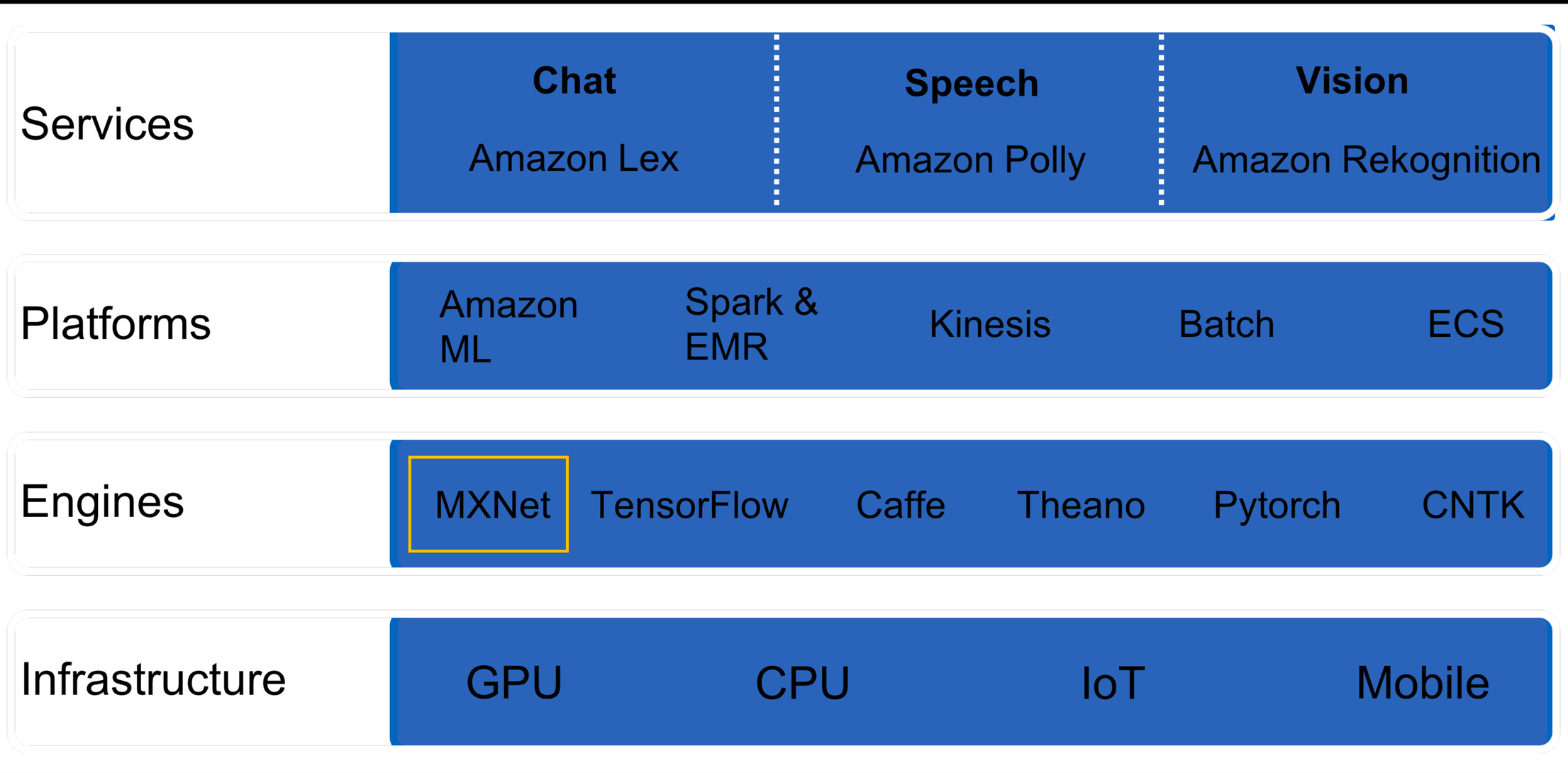
Airflow, Luigi or other schedulers on EC2

# Spark ML on Amazon EMR: spam detector

Adapted from https://github.com/databricks/learning-spark/blob/master/src/main/scala/com/oreilly/learningsparkexamples/scala/MLlib.scala

```
// Load 2 types of emails from text files: spam and ham (non-spam).
// Each line has text from one email.
val spam = sc.textFile("s3://jsimon-public/spam")
val ham = sc.textFile("s3://jsimon-public/ham")

// Create a HashingTF instance to map email text to vectors of 1000 features.
val tf = new HashingTF(numFeatures = 1000)
// Each email is split into words, and each word is mapped to one feature.
val spamFeatures = spam.map(email => tf.transform(email.split(" ")))
val hamFeatures = ham.map(email => tf.transform(email.split(" ")))

// Create LabeledPoint datasets for positive (spam) and negative (ham) examples.
val positiveExamples = spamFeatures.map(features => LabeledPoint(1, features))
val negativeExamples = hamFeatures.map(features => LabeledPoint(0, features))

val data = positiveExamples.union(negativeExamples)
data.cache()
val Array(trainingData, testData) = data.randomSplit(Array(0.8, 0.2))
trainingData.cache()

// Create a Naive Bayes trainer
val model = NaiveBayes.train(trainingData, 1.0)

val predictionLabel = testData.map(x=> (model.predict(x.features),x.label))
val accuracy = 1.0 * predictionLabel.filter(x => x._1 == x._2).count() / testData.count()
```

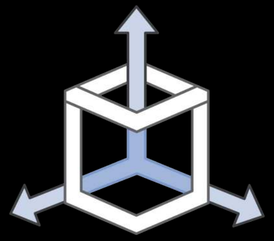
# Amazon AI for every developer



# Apache MXNet: Open Source library for Deep Learning

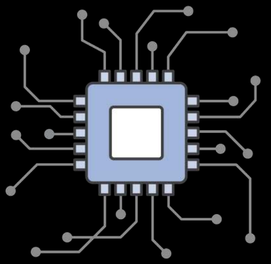
## Programmable

Simple syntax,
multiple languages



## Portable

Highly efficient
models for mobile
and IoT

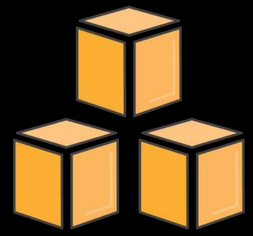


## High Performance

Near linear scaling
across hundreds of GPUs

## Most Open

Accepted into the
Apache Incubator

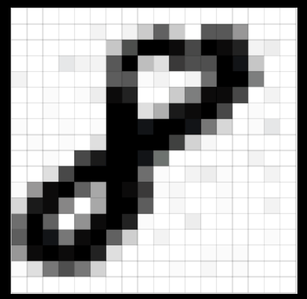


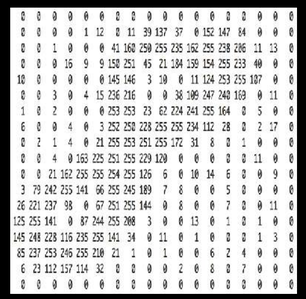
## Best On AWS

Optimized for
Deep Learning on AWS

https://
mxnet.incubator.apache.org/

# Apache MXNet demo: learning MNIST





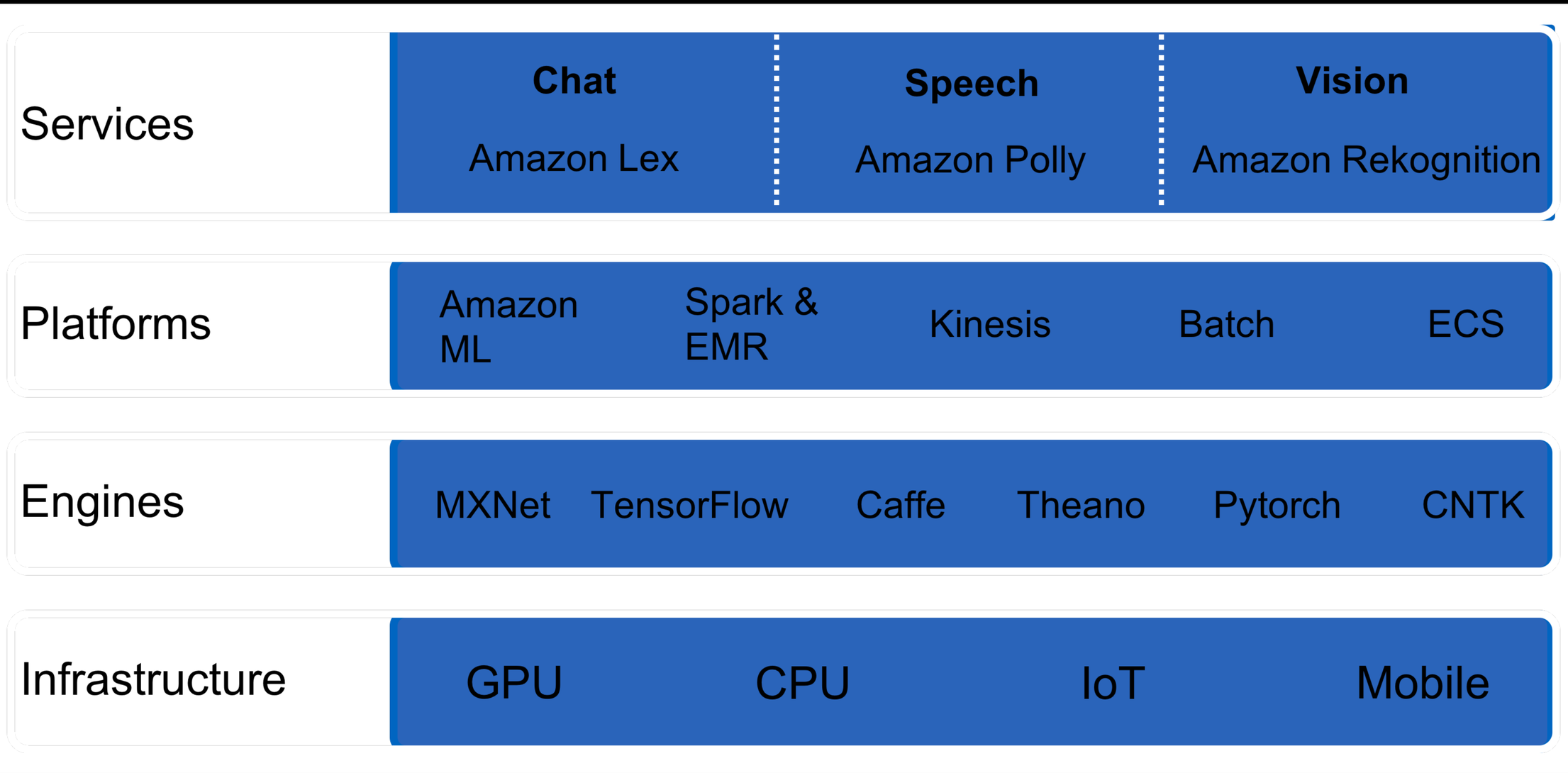
```
val model = new FeedForward(ctx = devs,
                            symbol = network,
                            numEpoch = numEpochs,
                            optimizer = optimizer,
                            initializer = new Xavier(factorType = "in", magnitude = 2.34f),
                            argParams = argParams,
                            auxParams = auxParams,
                            beginEpoch = beginEpoch,
                            epochSize = epochSize)
if (monitorSize > 0) {
  model.setMonitor(new Monitor(monitorSize))
}
model.fit(trainData = train,
          evalData = validation,
          evalMetric = new Accuracy(),
          kvStore = kv,
          batchEndCallback = new Speedometer(batchSize, 50),
          epochEndCallback = checkpoint)
```

https://mxnet.incubator.apache.org/tutorials/scala/
https://github.com/apache/incubator-mxnet/blob/master/scala-package

# Running MXNet in Spark

- Amazon EMR supports GPU instances (g2, p2, p3) and MXNet
- Run Data Processing and Deep Learning on the same cluster
- MXNet also has experimental support for Spark
  https://github.com/apache/incubator-mxnet/tree/master/scala-package/spark

# Amazon AI for every developer



# Resources

**https://aws.amazon.com/ai**

**https://aws.amazon.com/machine-learning**

**https://aws.amazon.com/emr**

**https://aws.amazon.com/blogs/big-data/**

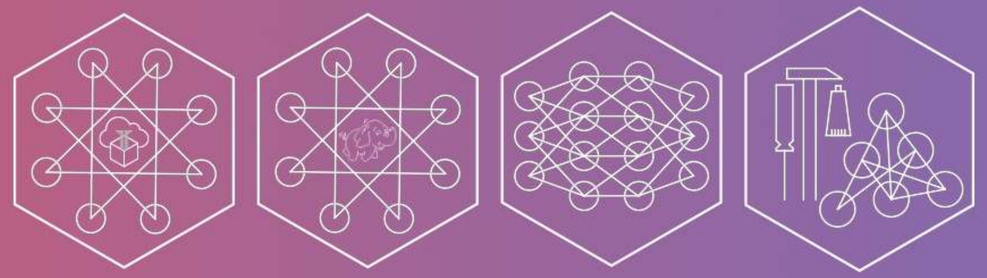
**https://aws.amazon.com/blogs/ai/**

**https://github.com/aws/{aws-sdk-java, aws-scala-sdk}**

**Code samples: https://github.com/juliensimon/aws/tree/master/ML**

# Semaine IA sur AWS

Du 11 au 15 novembre



https://aws.amazon.com/fr/events/


https://live.awsevents.com

# Thank you!

Julien Simon

AI/ML Evangelist, EMEA

@julsimon
https://aws.amazon.com/evangelists/julien-simon/