UCI University of California, Irvine

Machine Learning

Leveraging cloud technologies to change the economics of making Smart Apps

Tashia Hughes, Caitlyn Ta, Charles Nguyen, Kamila Bajaria

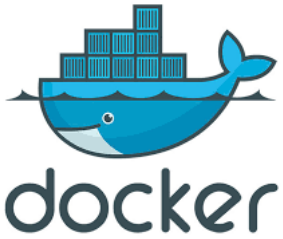
Saturday, April 21st 2018

The Challenge

- Demonstrate the application of machine learning to make a live app smarter
- Show how cloud technologies have profoundly affected the ability to apply ML in the real world
- Our app will start with a base of MNIST as a training set...
- But then we will *train our app* over time by providing it feedback about where it went wrong so it *gets smarter over time* and more *personalized* for a given user
- Demonstrate how cloud technologies are scaling up the value chain of ML by decoupling learning & modeling from developers calling prediction APIs “as a service”

Bill of Materials

Algorithms	K-Means, MXNet, TensorFlow, XGBoost
Coding	Python, Pandas, Flask, HTML/CSS/Bootstrap, JavaScript, Amazon Sagemaker SDK
Platforms	DynamoDB (instead of Mongo), Amazon AWS, Tableau, Docker Containers



Cloud Technology Stack

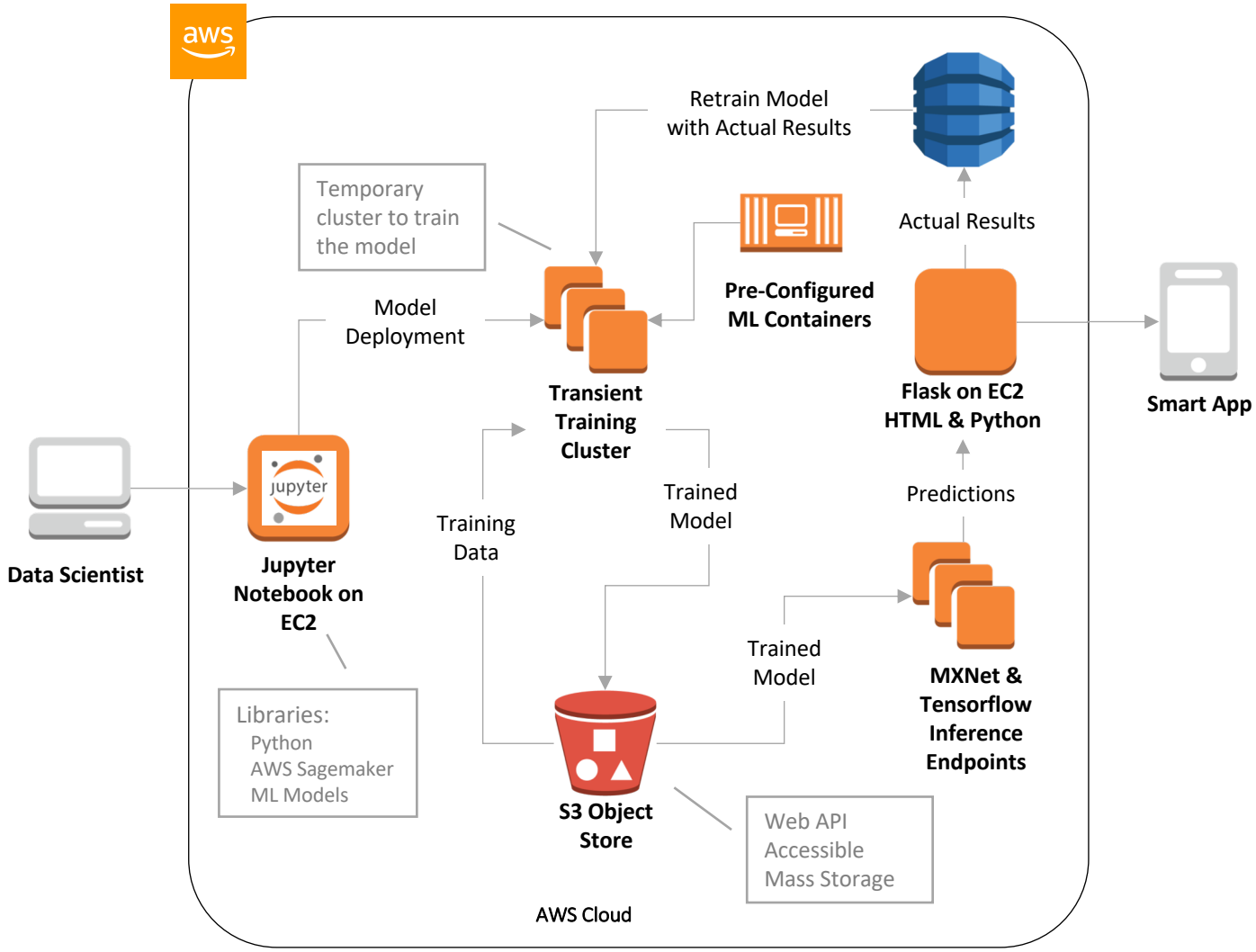
Our team chose Amazon's SageMaker service which provided the following benefits:

1. Python SDKs to cut down on lines of code for training and inferences
2. Ability to provision and terminate compute and storage resources on the fly, paying only for what we used
3. Hosted Jupyter notebooks with ML libraries pre-installed
4. Compute containers pre-loaded with ML algorithms ready to be trained

ML Enabled App Architecture in AWS

Features:

- Serverless app front end – no servers to manage, automatic scale up and down
- Large training clusters only used temporarily to train model – no paying for standing training clusters
- Pre-configured containers with ML libraries and algorithms pre-loaded
- Training data and trained model data stored in inexpensive, web-accessible object stores
- Hosted Jupyter notebooks with pre-installed libraries – no more 'pip install'
- CLI or GUI access for all steps
- Seamless integration between data scientists and smart app developers



Apps that get smarter over time

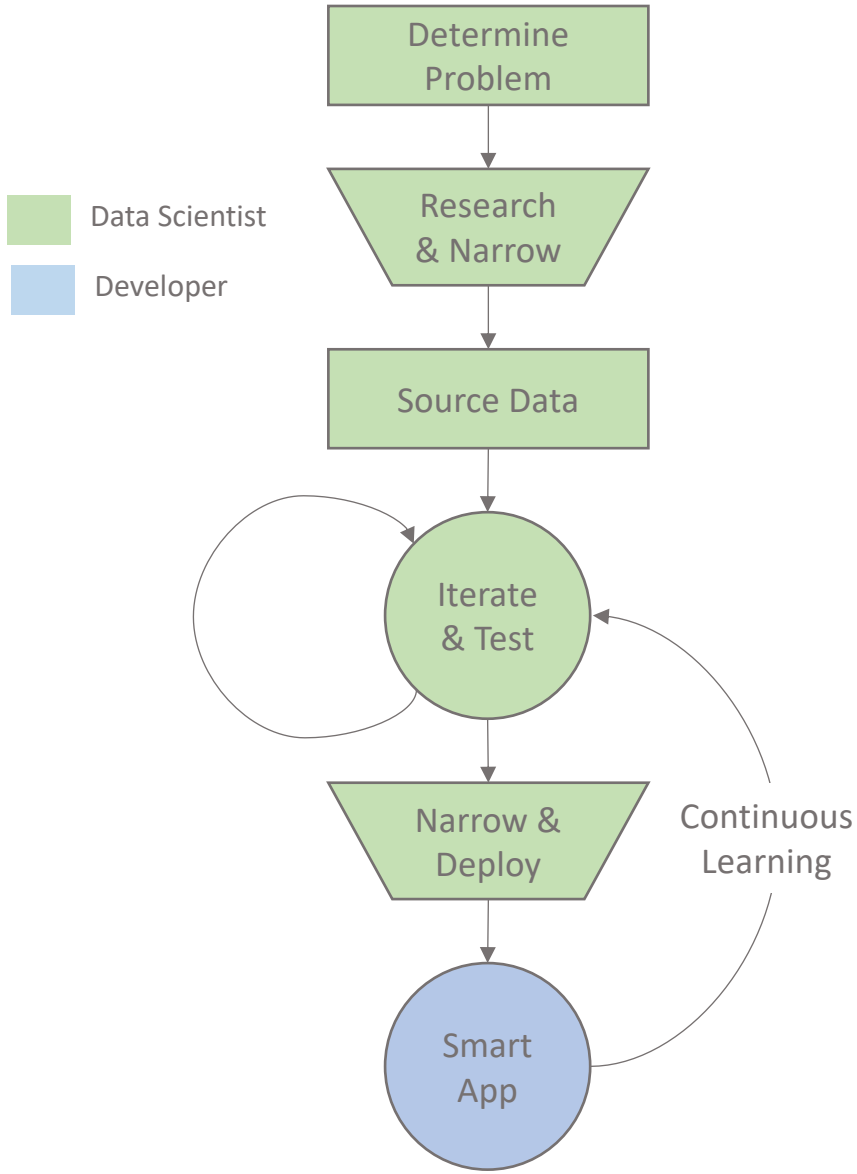
Smart application factories can rapidly scale up capabilities by decoupling the ML processes from app development

App developers can call endpoints provided by the data science function for predictions without knowing anything about machine learning or analytics

In this model, the intelligence of the app is provided “as a service” through an API available to developers for predictions

Data scientists, in turn, can use actual data from the live app to retrain and improve the inferences provided by their ML algorithms

The apps ”get smarter” over time

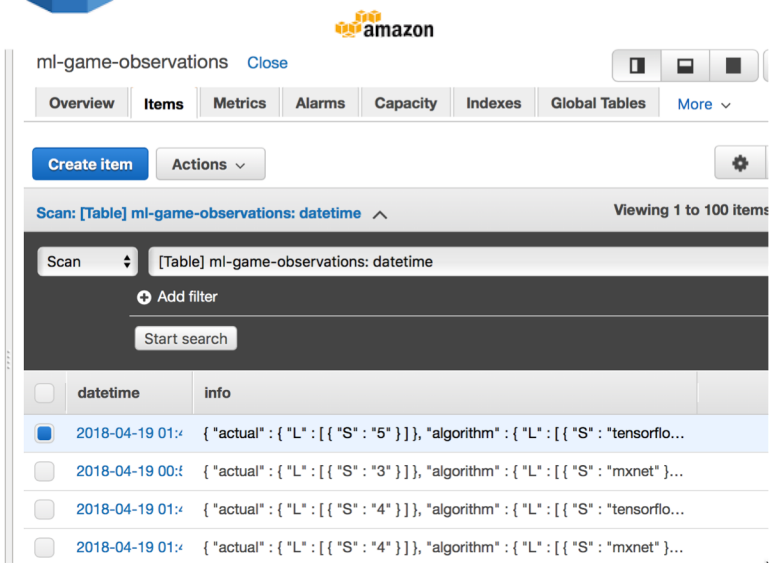


Getting Smarter Over Time

```
dynamodb = boto3.resource('dynamodb', region_name='us-west-2',
endpoint_url="https://dynamodb.us-west-2.amazonaws.com")
table = dynamodb.Table('ml-game-observations')

pe = "info"

response = table.scan(
    ProjectionExpression=pe
    )

import pandas as pd
dfTrain = pd.DataFrame(columns=['label', 'image'])

dfTrain.to_csv('train.csv', header=False, index=False)
```

Extract Data

DynamoDB



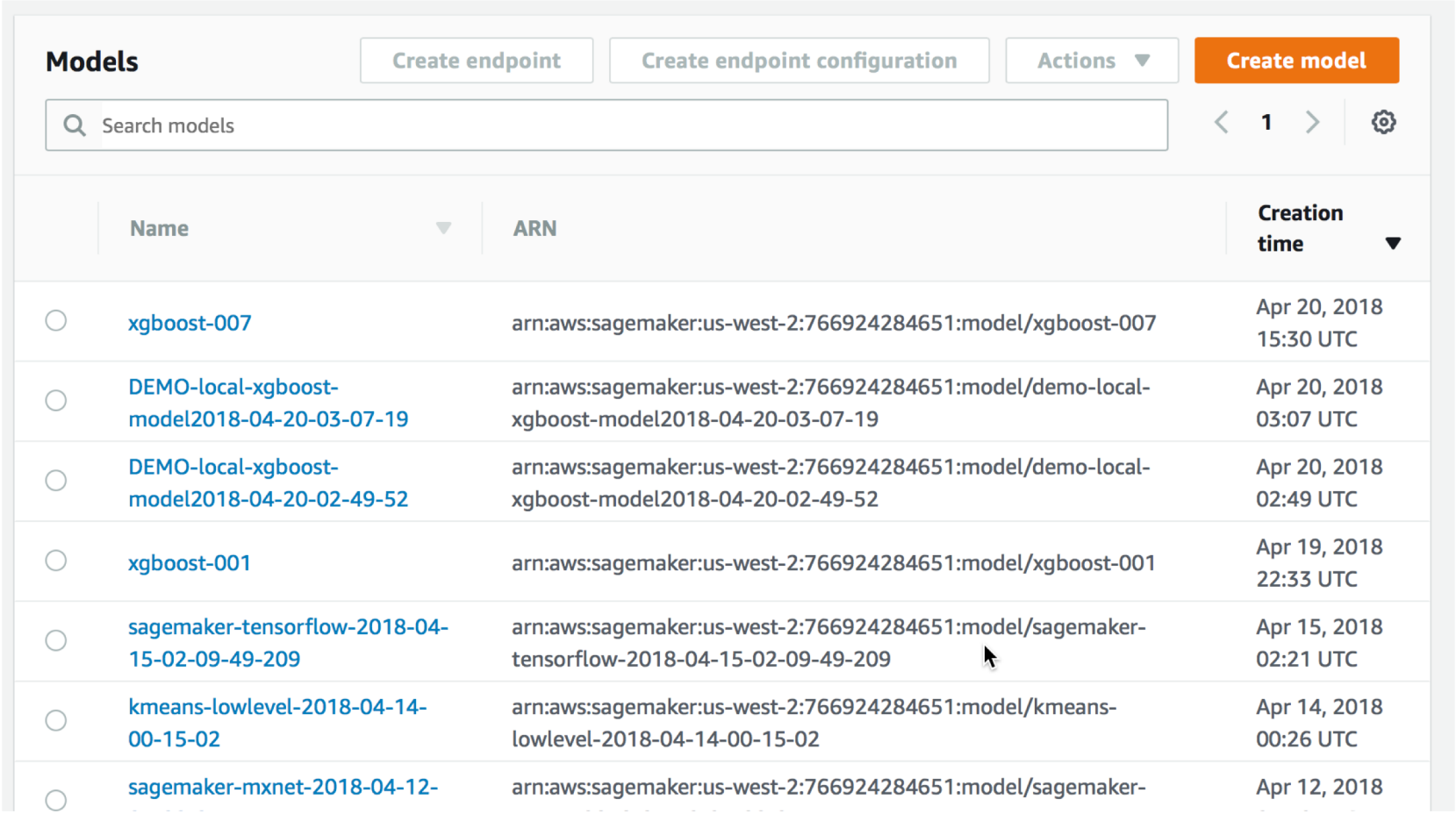
Application Persistence Layer

Retrain Model

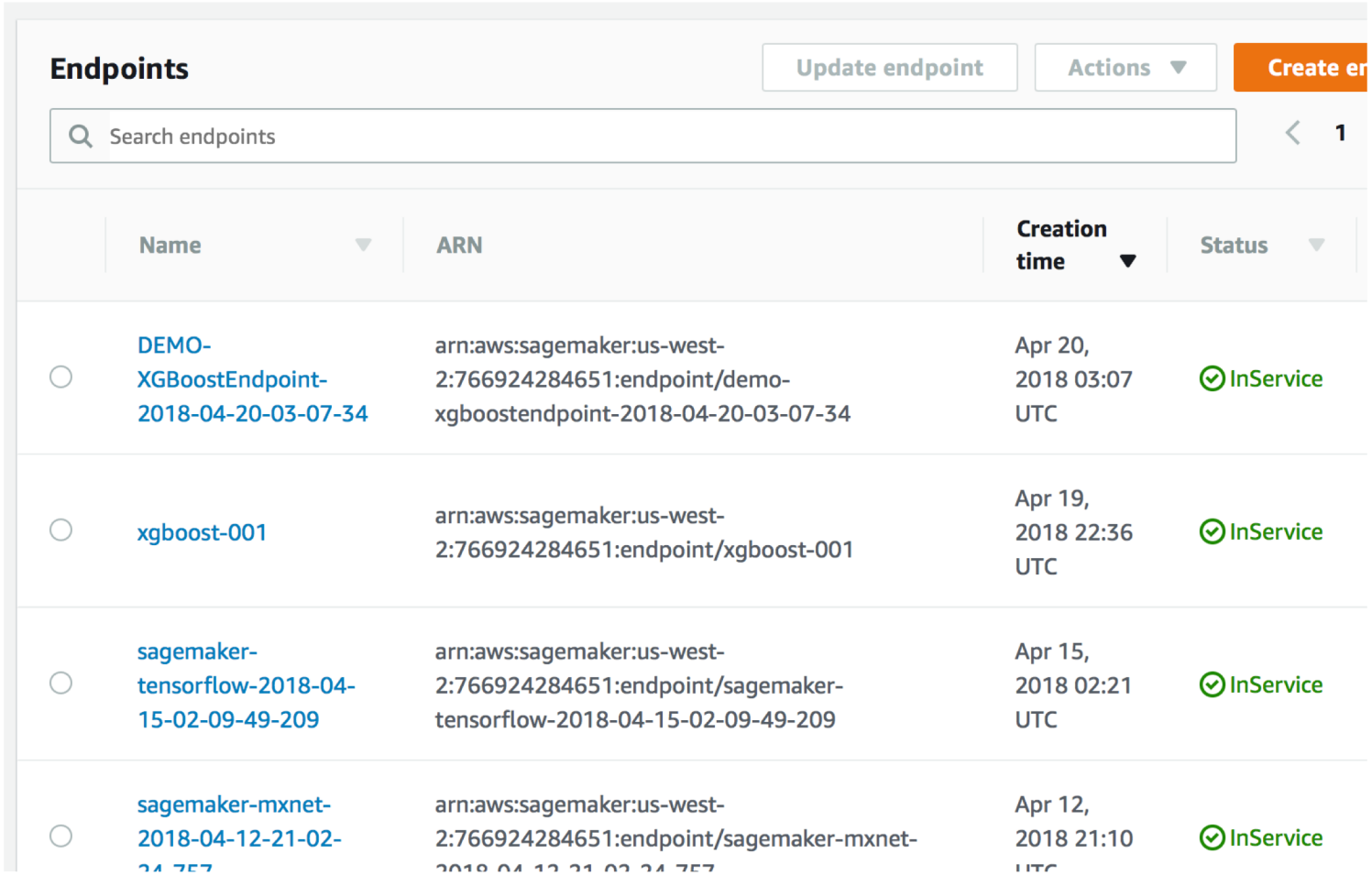
Amazon SageMaker

Deploy to Endpoints

SageMaker Manages Your Library of Trained Models



SageMaker Also Hosts Inference Endpoints for Developers to Call



Noteworthy Code Snippets

Cloud makes it easier…

Training the model with compute-optimized EC2 instances: **Data Scientist**

```
from sagemaker.tensorflow import TensorFlow

mnist_estimator =
TensorFlow(entry_point='mnist.py',
           role=role,
           training_steps=1000,
           evaluation_steps=100,
           train_instance_count=2,
           train_instance_type='ml.c4.xlarge')

mnist_estimator.fit(inputs)
```

Deploying the trained model on an inference endpoint

```
mnist_predictor =
mnist_estimator.deploy(initial_instance_count=1,

instance_type='ml.m4.xlarge')
```

Using the Sagemaker SDK to make calls to the inference endpoint easier: **Developer**

```
@app.route('/tensor')
def tensor():

    mynumber = request.args.getlist('image')

    predictor = TensorFlowPredictor('sagemaker-
          tensorflow-2018-04-15-02-09-49-209')

    mynumberarray = ast.literal_eval(mynumber[0])

    response = predictor.predict(mynumberarray)

    prediction =
          response['outputs']['classes']['int64Val'][0]

    answer= ("Most likely answer: {}".format(prediction))

    return(answer)
```

Key Lessons Learned

Economics	Cloud has revolutionized the economics of ML & AI. This is why so many things we buy these days are “smart”. The cost has come down to integrate ML into everyday things. Our team was able to train an algorithm with 70 thousand images in six minutes for pennies
Algorithms	Different algorithms produce varying results for a particular application. Experimentation and trial and error is required
Apps	ML in a vacuum is not nearly as useful as connecting ML to an app in a continuous feedback loop to make the model smarter over time

Conclusions

- Cloud economics have made ML functionality accessible to the masses
- Small companies with few capital resources can disrupt entire industries that make “dumb” things
- Combined with cloud-based IoT, Intelligence is being integrated into everyday things that get better over time
- ML is getting easier through publication of SDKs which abstract complexity
- Developers don’t need to know ML to make intelligent apps, Data Scientists can make their work available directly to apps through endpoints APIs

AWS Economics

- **Model training**:

  Two c4.xlarge instances @ $.199 per hour * 6 minutes = 4 cents to train model with 70,000 records

  8 CPU & 16 GB RAM

- **Inference Endpoint Hosting**:

  One m4.xlarge instance @ $.2 per running hour, or $1,013 for one year of sustained 24 x 7 usage

  4 CPU & 16 GB RAM

- **DynamoDB for app persistence**:

  Free tier: < 25 GB of storage and < 25 read/writes per second

  Infrastructure scales to meet demand

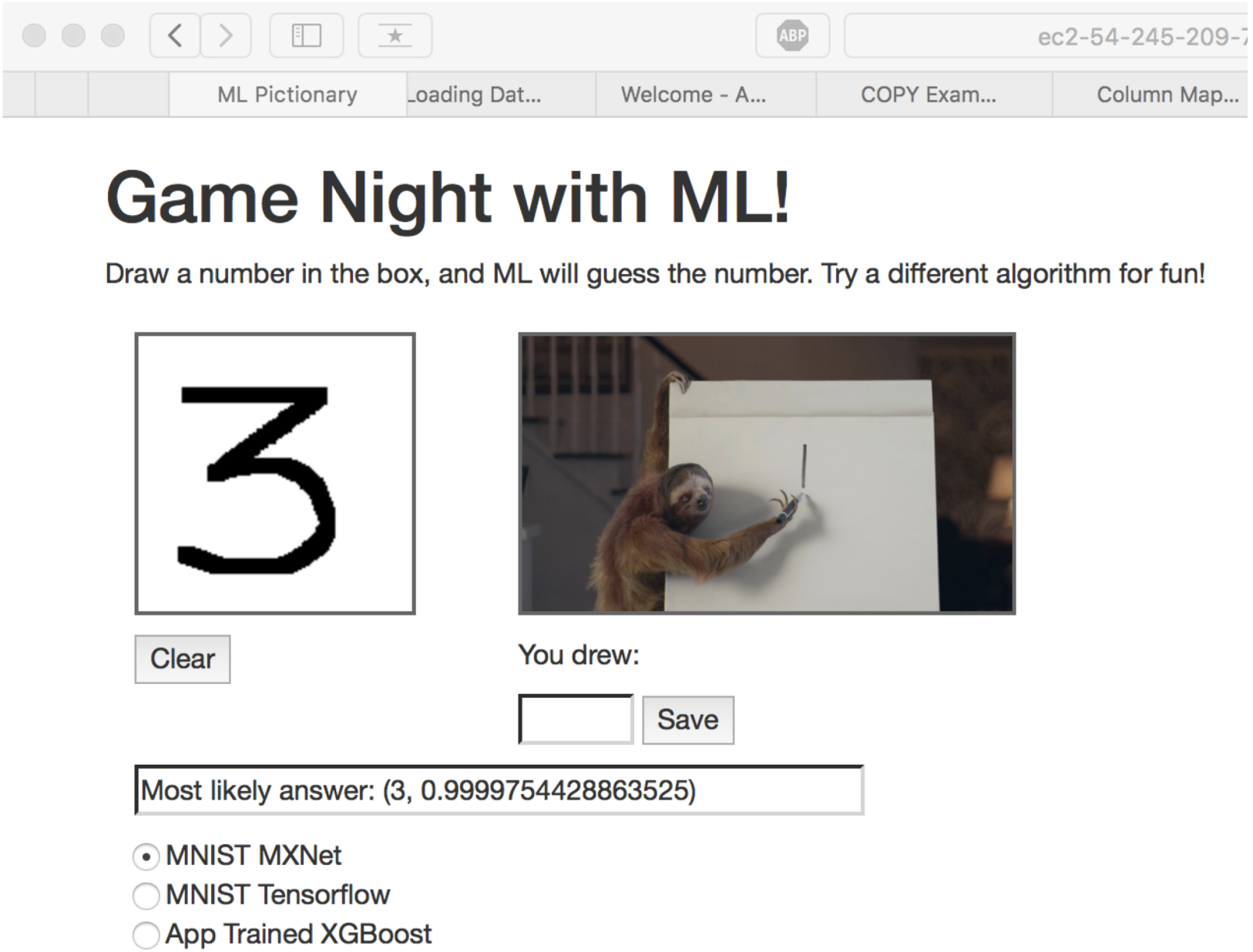
- **Jupyter Notebook Host**ing:

  One T2.medium @ $.0464 per running hour or $235 for one year of sustained 24 x 7 usage

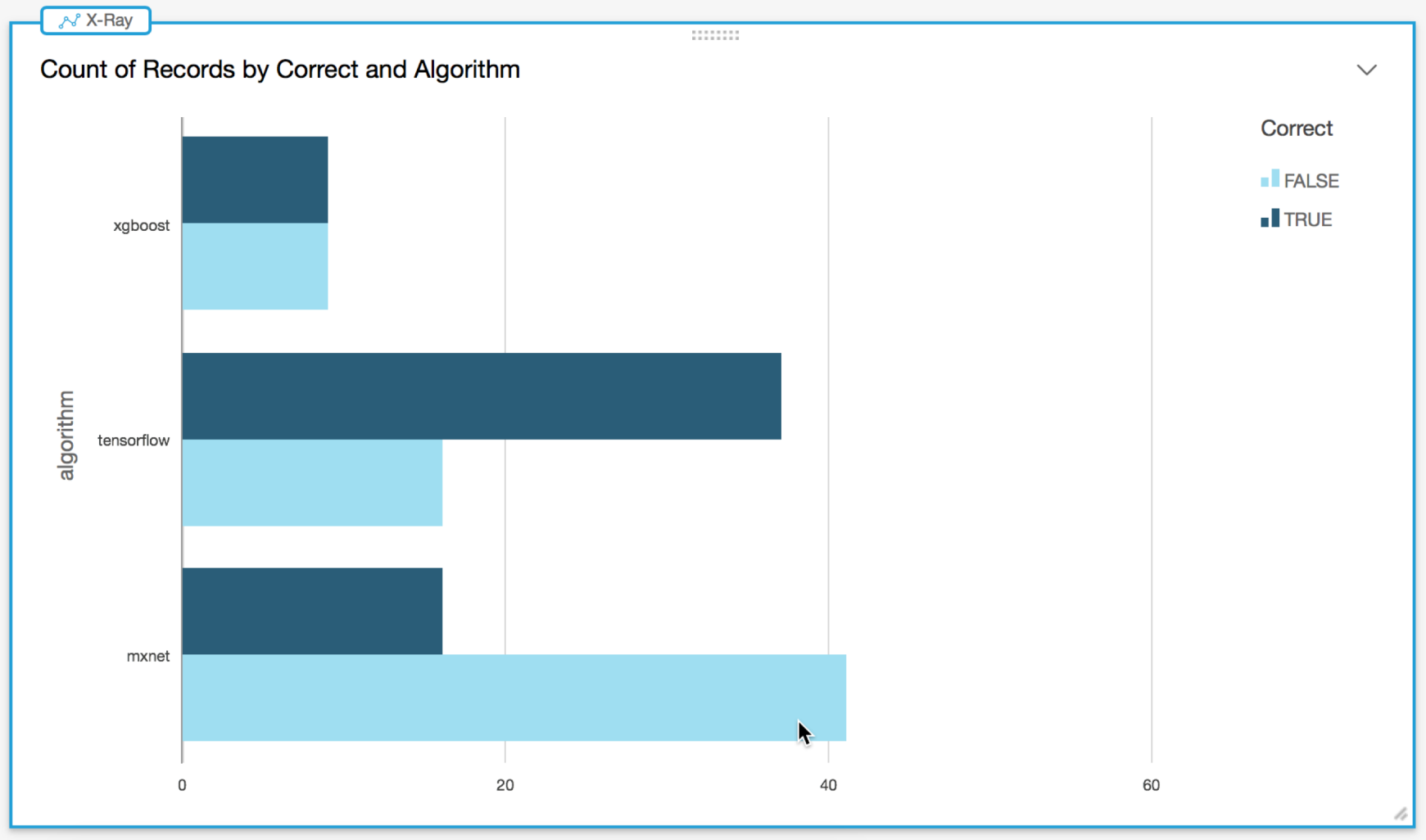
  2 CPU & 4 GB RAM

- **Sagemaker SDKs** = Free
- **Pre-configured algorithm containers** = Free
- **Software licenses** = Free

Smart App


ec2-54-245-209-7
ML Pictionary
Loading Dat...
Welcome - A...
COPY Exam...
Column Map...
Game Night with ML!
Draw a number in the box, and ML will guess the number. Try a different algorithm for fun!
Clear
You drew:
Save
Most likely answer: (3, 0.9999754428863525)
MNIST MXNet
MNIST Tensorflow
App Trained XGBoost

Tensorflow with MNIST was Most Accurate



‘2’ Was the most correctly guessed, while ‘0’ was least correctly guessed