📊 SageMaker Migration PoC Demo Deck

Slide 1: Title & Agenda

Element	Content
Title	**Accelerating ML with Amazon SageMaker**
Subtitle	Proof-of-Concept Demo: Unlocking Scale, Speed, and Efficiency
Goal	Validate technical assumptions and showcase tangible value by addressing key migration objectives.
Agenda	1. Why We're Demoing (Pain Points) 2. Demo 1: Data Preparation **(SageMaker Data Wrangler)** 3. Demo 2: Scalability & Concurrency **(SageMaker Training)** 4. Demo 3: Optimization & Efficiency **(SageMaker Automatic Model Tuning)** 5. Next Steps

Slide 2: Why We're Demoing: Addressing Your Pain Points

Reason for Demos (Value-Focused Messaging)

Your Current Challenge	Our SageMaker Solution	Demo Objective
🐢 **Slow, Manual Data Prep** (Time-consuming feature engineering)	**Visual, low-code data transformation** and pipeline export.	Validate complex transformation logic.
🛑 **Lack of Scalability/GPU Use** (Training takes days; no easy GPU access)	**Managed, on-demand GPU instances** with single-line code changes.	Demonstrate training speedup (CPU vs. GPU).
⚙️ **Resource Contention** (Data scientists waiting for resources)	**Isolated, concurrent Training Jobs** with automatic resource teardown.	Show multiple jobs running without conflict.
📉 **Manual Optimization** (Trial-and-error hyperparameter tuning)	**Automated Model Tuning** to efficiently find the best performing model.	Confirm objective metric and parameter ranges.

Slide 3: Demo 1: Data Preparation & Feature Engineering

Service: 🛠️ Amazon SageMaker Data Wrangler

Section	Explanation	Key Information Gathered
The Problem	Your team spends more time **cleaning and preparing data** than building models. It's error-prone and hard to reproduce.	**Source System Connectivity** (e.g., Redshift, Snowflake, S3)
The Solution	Data Wrangler provides a **visual interface** to inspect data, apply 300+ transformations, and automatically generate the necessary code for pipelines.	**Top 3 Transformation Steps** your team currently performs manually.
Key Features Shown	1. **Data Quality & Insights** (Quick analysis of feature distributions). 2. **No-Code Transformations** (Applying a custom logic, like missing value imputation). 3. **Export to Code** (Generating a SageMaker Processing Job script).	**Desired final Feature Set/Schema**.

Slide 4: Demo 2: Scalability & Concurrency

Service: 🚀 Amazon SageMaker Training Jobs

Section	Explanation	Key Information Gathered
The Problem	Your models run on limited CPU capacity. Using GPUs is complex, and **concurrent experiments block** other data scientists.	**Current Model Framework** (e.g., PyTorch, TensorFlow) and **dependencies**.
The Solution	SageMaker manages the compute cluster (**EC2 instances**) for you. You only specify the **instance type** (CPU or GPU) and the **training script**.	**Current Average Training Time** for a specific model.
Key Features Shown	1. **Lift-and-Shift Code:** Using a simple Python script as the `entry_point`. 2. **GPU Acceleration:** Launching the job on an `ml.m5.xlarge` (CPU) vs. `ml.g4dn.xlarge` (GPU). 3. **Concurrency:** Launching **multiple, simultaneous training jobs** using a single notebook command.	**Target Training Time** and **Maximum Concurrent Users/Jobs** needed.

Slide 5: Demo 3: Optimization & Efficiency

Service: 🎯 SageMaker Automatic Model Tuning (AMT)

Section	Explanation	Key Information Gathered
The Problem	Manually searching for the best hyperparameters is a **labor-intensive, non-linear process** that wastes compute resources and human time.	**Key Hyperparameters** they manually tune (e.g., learning rate, epochs).
The Solution	AMT uses **Bayesian Optimization** to intelligently search the parameter space, minimizing cost and time to find the best-performing model configuration.	**Objective Metric** for model success (e.g., AUC, F1-Score).
Key Features Shown	1. **Defining the Search Space** (`IntegerParameter`, `ContinuousParameter`). 2. **Setting the Objective** (Maximize/Minimize the target metric). 3. **Parallel Execution** (AMT automatically runs concurrent jobs to speed up the tuning process).	**Target Performance Threshold** (e.g., "We need an AUC of 0.85").

Slide 6: Summary & Next Steps

Element	Content
Summary: Value Delivered	📈 **Speed:** Move models from days to hours using managed GPU training. 🤝 **Collaboration:** Concurrent jobs enable the whole team to work in parallel. 🧠 **Intelligence:** Automated tuning finds better models, faster, freeing up data scientist time.
Key Client Input Needed	1. Sanitized sample dataset and data access points. 2. A simple version of a current, representative training script (`train_script.py`). 3. Confirmation on target instance types and concurrency needs.
Next Steps	✅ Finalize technical requirements based on demo feedback. 📅 Develop a detailed Migration Plan (WBS & Timeline). 🤝 Move to Phase 1: Pilot Migration.