Amazon SageMaker AI

Build, train, deploy, and manage GenAI & ML models at scale

Giuseppe Zappia

Principal AI/ML Specialist Solutions Architect,
Amazon Web Services (AWS)

Agenda

SageMaker AI and Bedrock

Model Training & Fine-Tuning

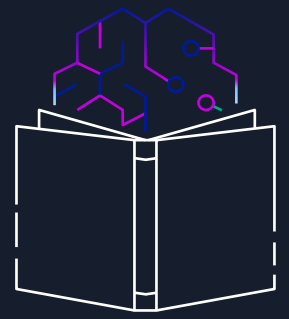
Model Deployment Options

Why should I fine-tune?

Improved Relevance

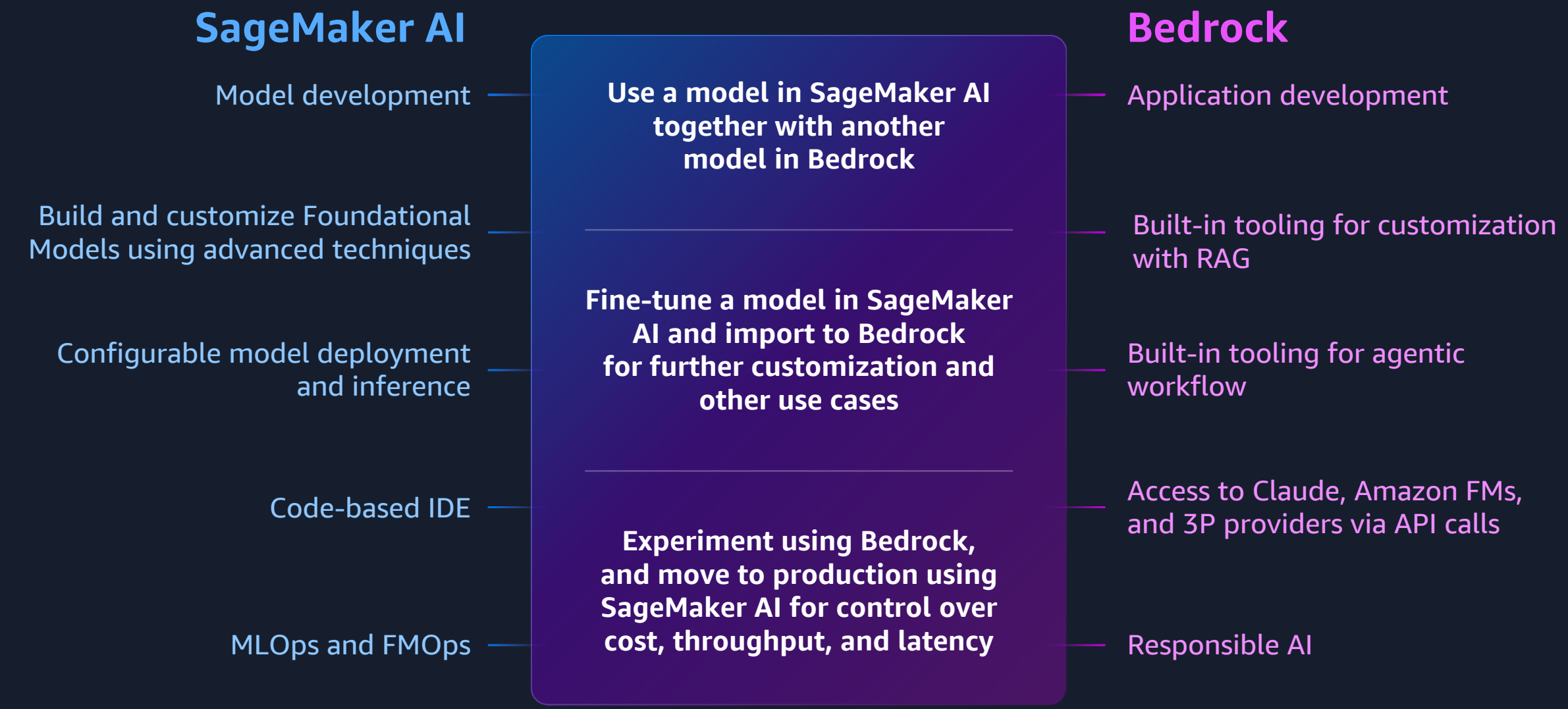
Enhanced Generation Quality

Increase Accuracy



Domain Customization

Amazon SageMaker AI and Amazon Bedrock



Amazon SageMaker AI

Build, train, and deploy ML models at scale, including FMs

Build FM's from scratch

Create your own ML models, including FMs, with integrated purpose-built tools and high-performance, cost-effective infrastructure

Customize foundation models

Access and evaluate 250+ FMs that can be customized easily for your use case

Implement MLOps & governance

Create reliable and repeatable workflows incorporating MLOps practices with purpose-built tooling

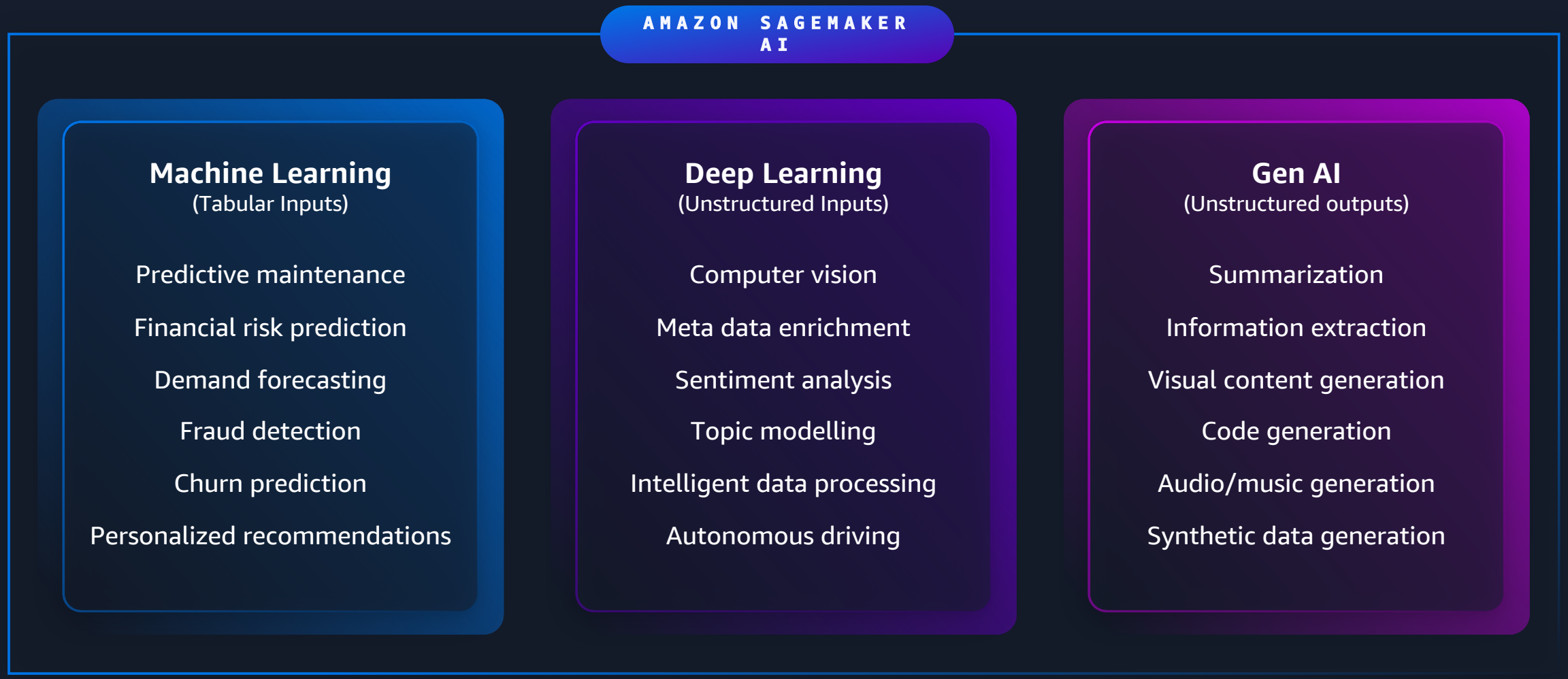
Improve ML governance

Enhance model governance and compliance with built-in governance tools

Manage and deploy models for inference

Easiest way to deploy AI & ML models including foundation models (FMs) to make inference requests at the best price performance for any use case

SageMaker AI supports ML, Deep Learning and GenAI



Model Building

Single, fully managed IDE for notebooks, code, and data

Accelerate and scale data prep for AI and ML

Use the most comprehensive set of tools for both structured and unstructured datasets

Access data

Easily access and query data from a wide variety of data sources

Cleanse, label, enrich

Create high quality labeled datasets for training models using your tool of choice or through human feedback

Analyze and visualize

Explore data through purpose-built analysis and visualization tools, or visualize geospatial data on interactive 3D accelerated maps

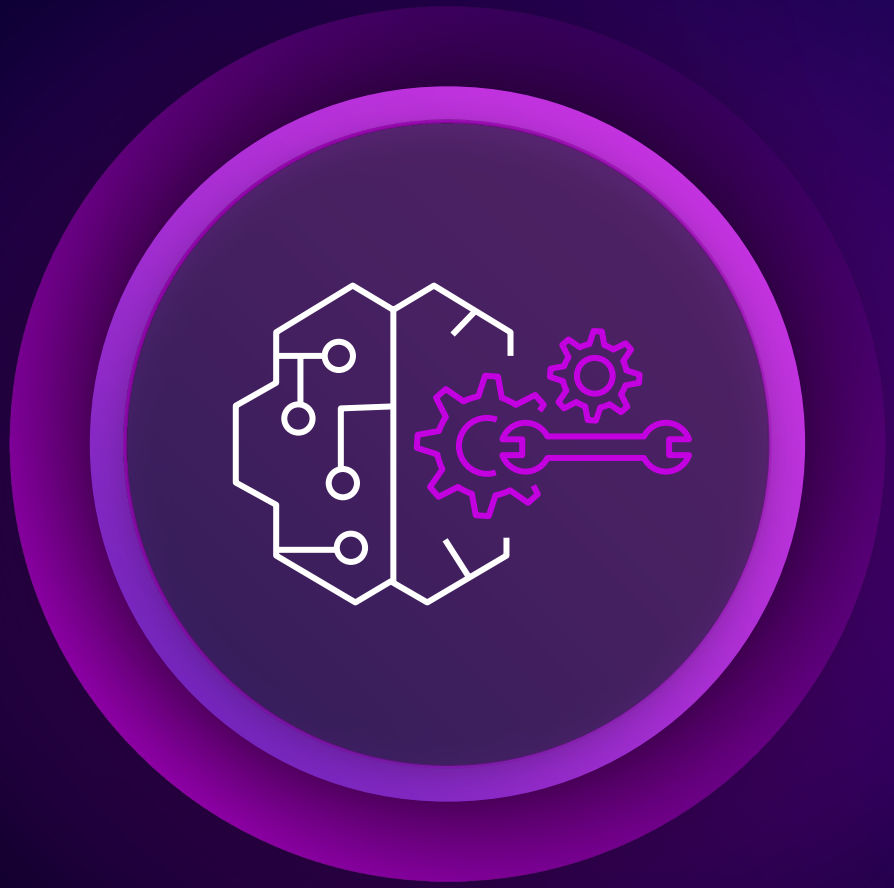
Scale

Efficiently process large amounts of data while reducing cost

Store and share

Securely store, manage, and share features to be used the ML lifecycle

aws



Model Training & Fine-Tuning

Fast and cost-efficient ML and Gen AI model training

SageMaker AI offers two training options

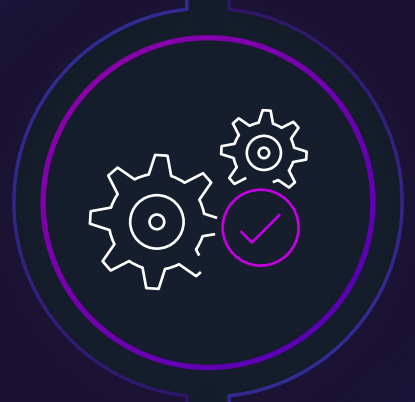
PURPOSE-BUILT INFRASTRUCTURE FOR FM TRAINING

Fully managed Training jobs

Fully managed resilient infrastructure for large-scale and cost-effective training

Focus on model building rather than IT

Provide access to flexible on-demand GPU cluster with a pay as you go option



Amazon SageMaker HyperPod

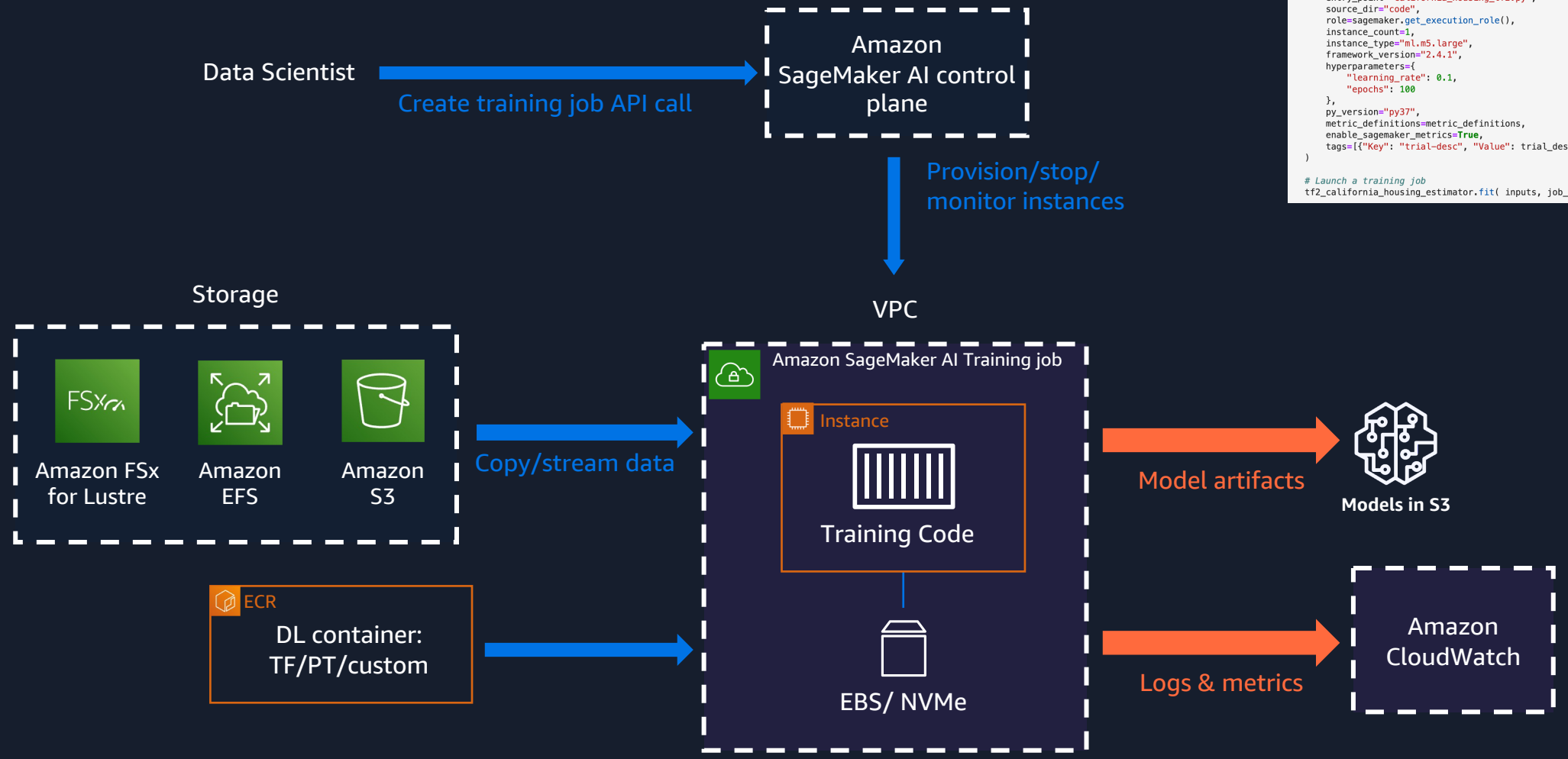
Resilient and **self orchestration** infrastructure for maximum resource control

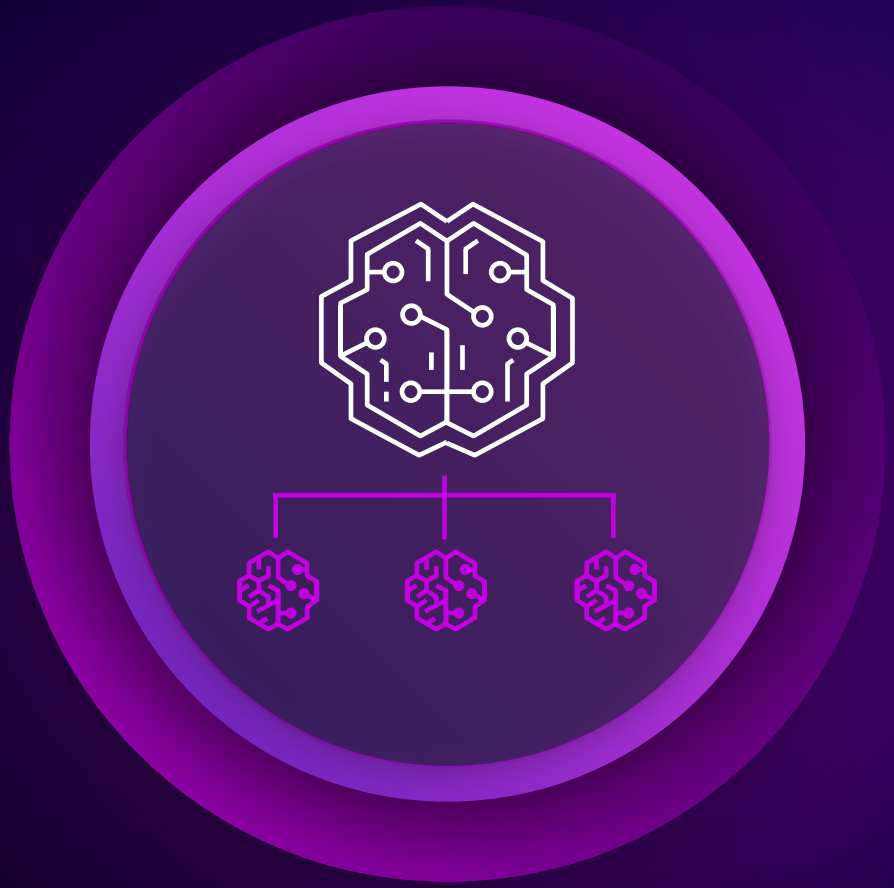
Customize and manage cluster orchestration (Slurm or EKS)

Schedule workloads to maximize cluster utilization across teams

Training on Amazon SageMaker AI

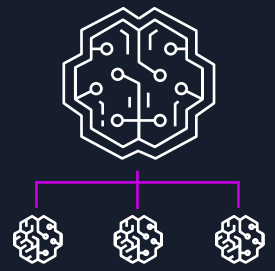
```python
# Input data from s3
inputs = {"train": s3_inputs_train, "test": s3_inputs_test}

metric_definitions = [
    {"Name": "loss", "Regex": "loss: ([0-9\\.]+)"},
    {"Name": "accuracy", "Regex": "accuracy: ([0-9\\.]+)"},
    {"Name": "val_loss", "Regex": "val_loss: ([0-9\\.]+)"},
    {"Name": "val_accuracy", "Regex": "val_accuracy: ([0-9\\.]+)"},
]

# Create a TensorFlow Estimator
tf2_california_housing_estimator = TensorFlow(
    entry_point="california_housing_tf2.py",
    source_dir="code",
    role=sagemaker.get_execution_role(),
    instance_count=1,
    instance_type="ml.m5.large",
    framework_version="2.4.1",
    hyperparameters={
        "learning_rate": 0.1,
        "epochs": 100
    },
    py_version="py37",
    metric_definitions=metric_definitions,
    enable_sagemaker_metrics=True,
    tags=[{"Key": "trial-desc", "Value": trial_desc}],
)

# Launch a training job
tf2_california_housing_estimator.fit( inputs, job_name=training_job_name)
```





Model Deployment

Easily deploy AI and ML models - From low latency and high throughput to long-running inference



Deploy AI and ML models

Fully managed deployment for inference at scale

Wide selection of infrastructure

70+ instance types with varying levels of compute and memory to meet the needs of every use case. Deliver up to 40% better inference price performance with Inf2 instances

Deploy models in production for inference for any use case

From low latency and high throughput to long-running inference

Cost-effective deployment

Reduce inference cost by at least 50% with multi-model/multi-container endpoints, serverless inference, and elastic scaling

Shadow testing & automatic deployment guardrails

Validate the performance of new ML models against production models. Minimize risk when deploying new model versions on SageMaker AI using linear, canary, or blue green traffic switching

Built-in integration for MLOps

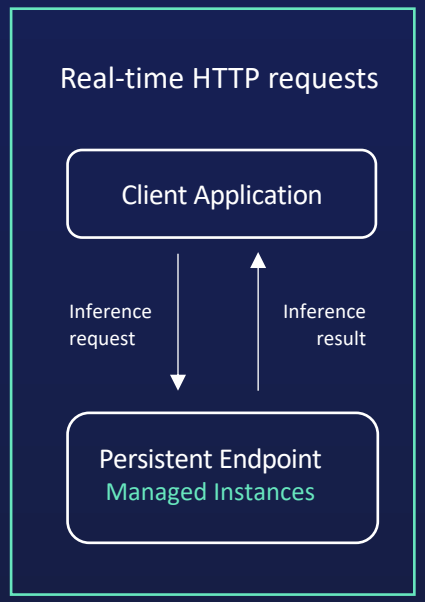
ML workflows, CI/CD, feature management, lineage tracking, and model management

Large model inference container

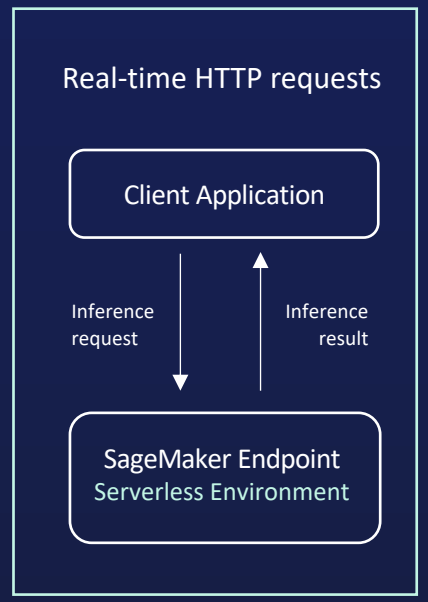
Achieve best price performance with the latest inference optimizations tools, model servers, and libraries packaged into a single container

Amazon SageMaker Deployment Modes

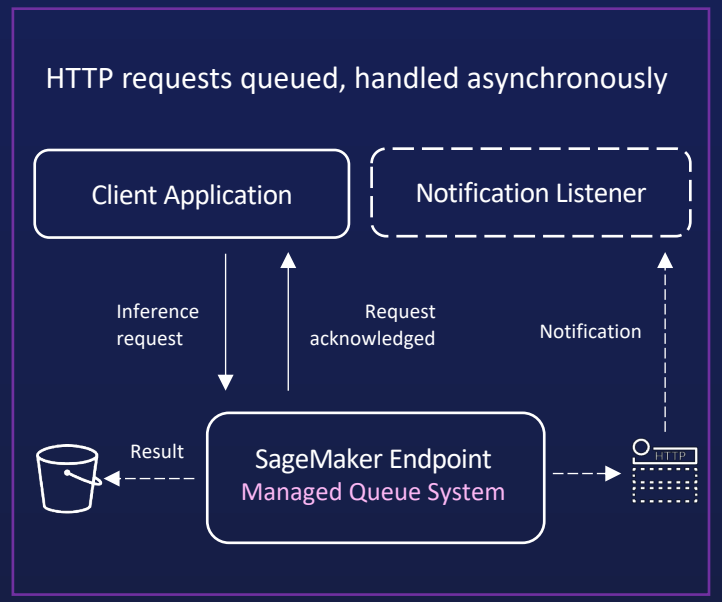
Real-time Inference



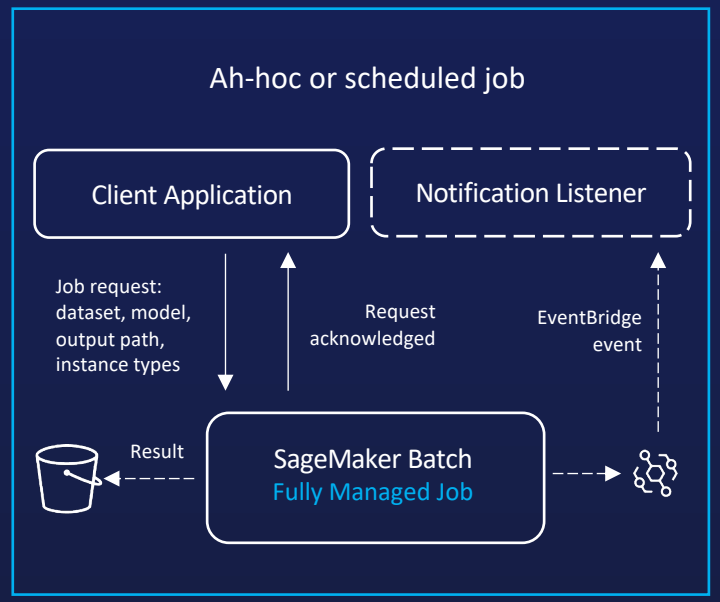
Serverless Inference



Asynchronous Inference



Batch Transform



Example use cases and technical considerations

Ad serving, search, personalized recommendations, Generative AI	Extract data from documents, form processing, chatbots, model dev/test	Video processing, large image processing, decoupled applications and systems	Business forecasting, propensity modeling, churn prediction, predictive maintenance
Low latency, high throughput Supports multi-model endpoints	Automatically scale to accommodate unpredictable traffic (scales to zero)	Ability to scale resources to zero Responses can be near real-time	Suitable for periodic arrival of large datasets
Responses within milliseconds	Response times vary (warm/cold start)	Processing times vary: queue size, worker status	Jobs can be long running
Max request payload: 6 MB Timeout: 60 sec	Max request payload: 4 MB Timeout: 60 sec	Inference input: Pointers to S3 objects (1 GB max) Ideal for models with long processing times (15min max)	Ideal for large datasets (Batch Transform allows for splitting of datasets across multiple instances)

Custom Model Import

Import your models customized on AWS, on premises, or elsewhere into Amazon Bedrock

Leverage your existing model customization investments

A unified developer experience across base, customized, and imported models

Fully managed and serverless

Use the security and IP protection of Amazon Bedrock

Design patterns

Supported architecture

- Model architecture
  - **Llama 2 and 3**
  - **Mistral**
  - **Flan-T5**
- Precision supported
  - **FP16**
  - **FP32**
  - **BF16**

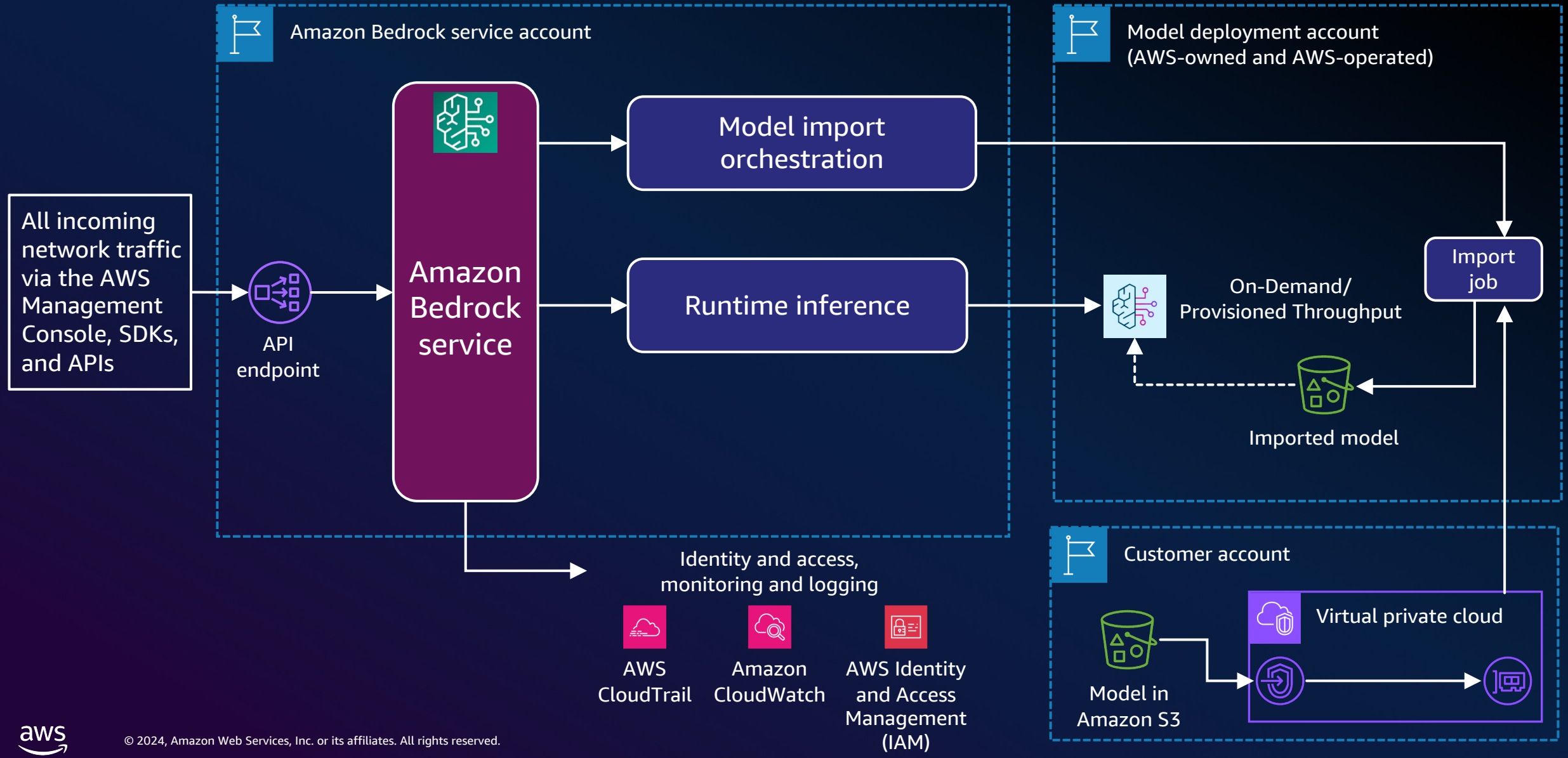
Supported patterns

- Safetensors format
- Open source from Hugging Face
- Fine-tuned on SageMaker
- Fine-tuned on premises/ Amazon EC2
- LoRA-PEFT merged adapters

Supported inference mode

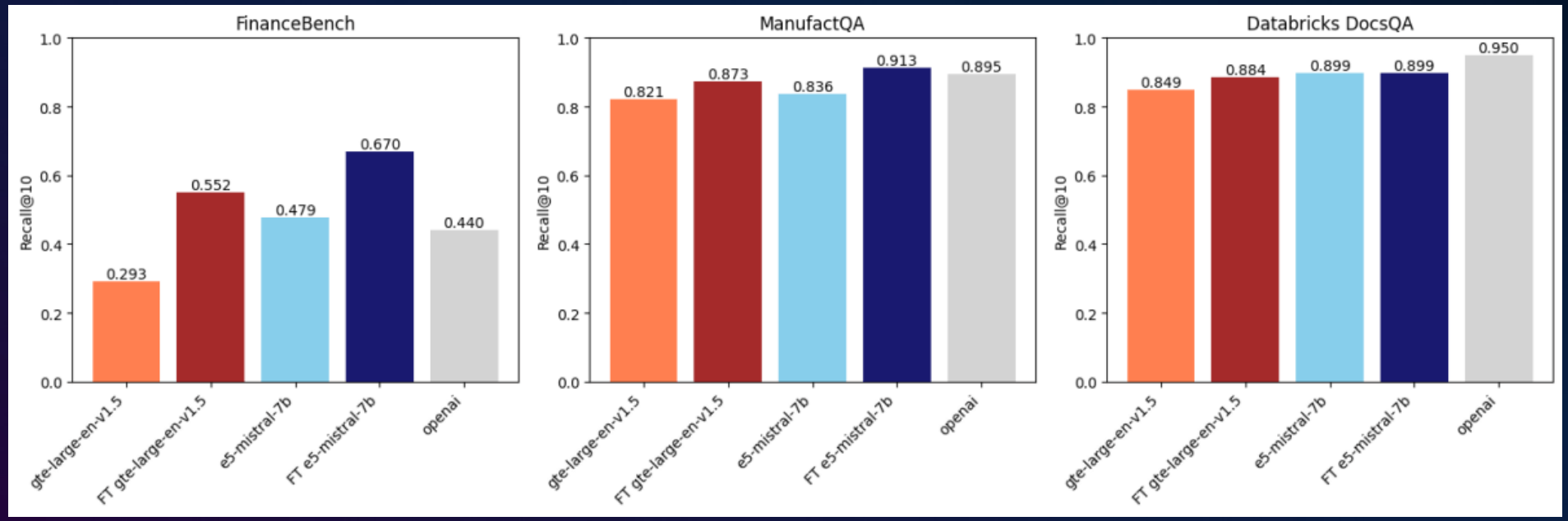
- On-Demand
- Provisioned Throughput

Custom Model Import architecture overview



Fine-tuning Embeddings Benefits

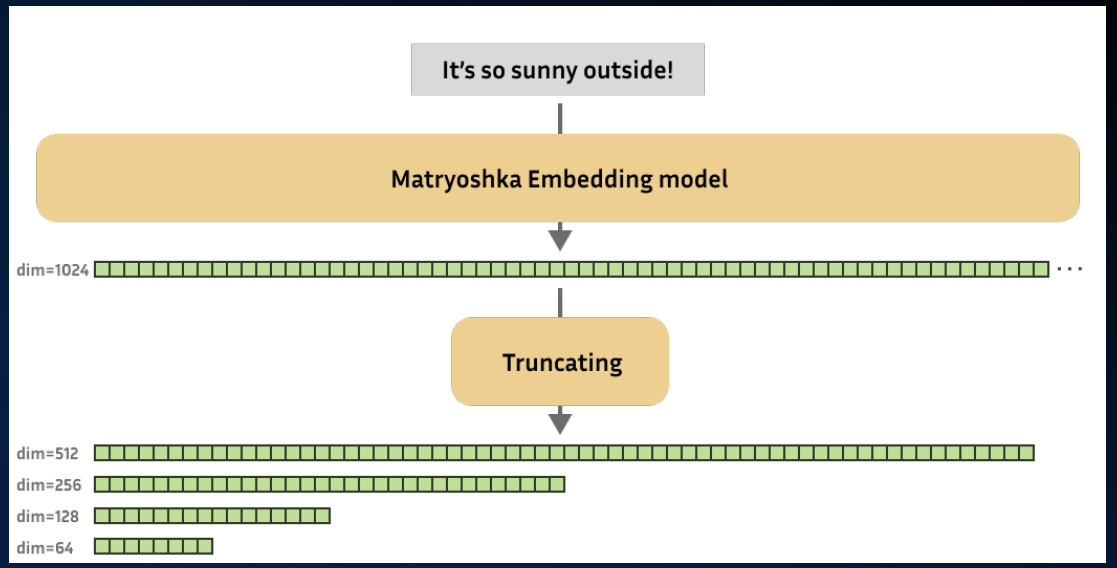
- Allows you to potentially use smaller models, increasing performance
- Improve vector search relevancy
- Because of improved retrieval, it can lead to less hallucinations and more grounded responses
- Can even outperform as well or better than reranking



Source: https://www.databricks.com/blog/improving-retrieval-and-rag-embedding-model-finetuning

Matryoshka Embeddings

- Typically dense embeddings produce outputs of a fixed size (e.g. 768, 1024)
- Matryoshka Representation Learning (arxiv:205.13147) illustrates how embedding models can be shrunk without a significant performance penalty
- This is achieved by storing coarse grained information in the lower dimensions of the vector, and fine-grained information at the higher dimensions
- Combining this with fine tuning allows for potentially using much smaller vector dimensions and achieving similar performance profiles to non-tuned full embeddings

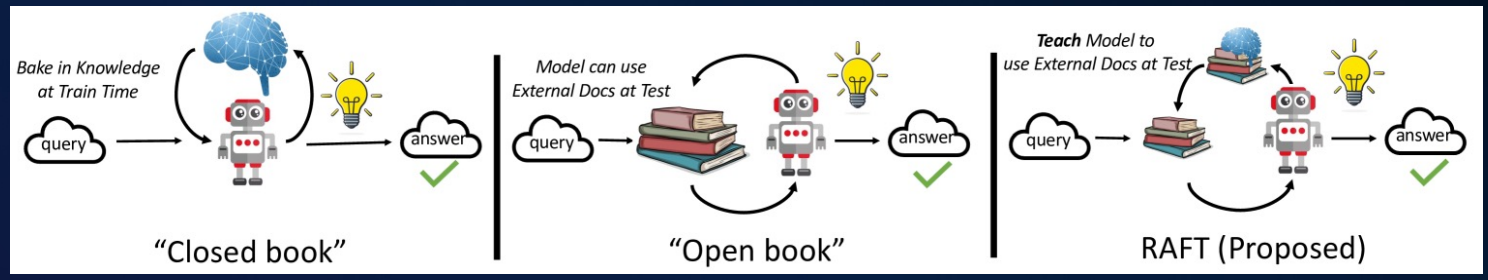


Source: https://huggingface.co/blog/matryoshka

Source : https://aws.amazon.com/blogs/machine-learning/nemo-retriever-llama-3-2-text-embedding-and-reranking-nvidia-nim-microservices-now-available-in-amazon-sagemaker-jumpstart/

Fine-tuning RAG Generation

- RAFT: Adapting Language Model to Domain Specific RAG (arxiv: 2403.10131) showcases a technique for fine-tuning RAG generation models to improve performance.



- By training the model with a corpus of information consist of "oracle" documents (the ones that lead to the correct output) and "distractor" documents (irrelevant ones), we can end up with a model that consistently outperforms untrained models.

- The output can be further improved by incorporating Chain-of-thought reasoning.

Code Samples

workshops/building-rag-workflows-with-sagemaker-and-bedrock

GenAI on SageMaker

Thank you!