AWS re:Invent

NOV. 28 – DEC. 2, 2022 | LAS VEGAS, NV

AIM406

Tune performance & optimize ML inference with Amazon SageMaker, feat. Arlo

Giuseppe A. Porcelli (he/him)
Principal, ML Specialist SA
Amazon SageMaker

Ram Vegiraju (he/him)
ML Specialist SA
Amazon SageMaker

Dillibabu Rathinam (he/him)
Sr. Director, Cloud Engineering
Arlo Technologies Inc.

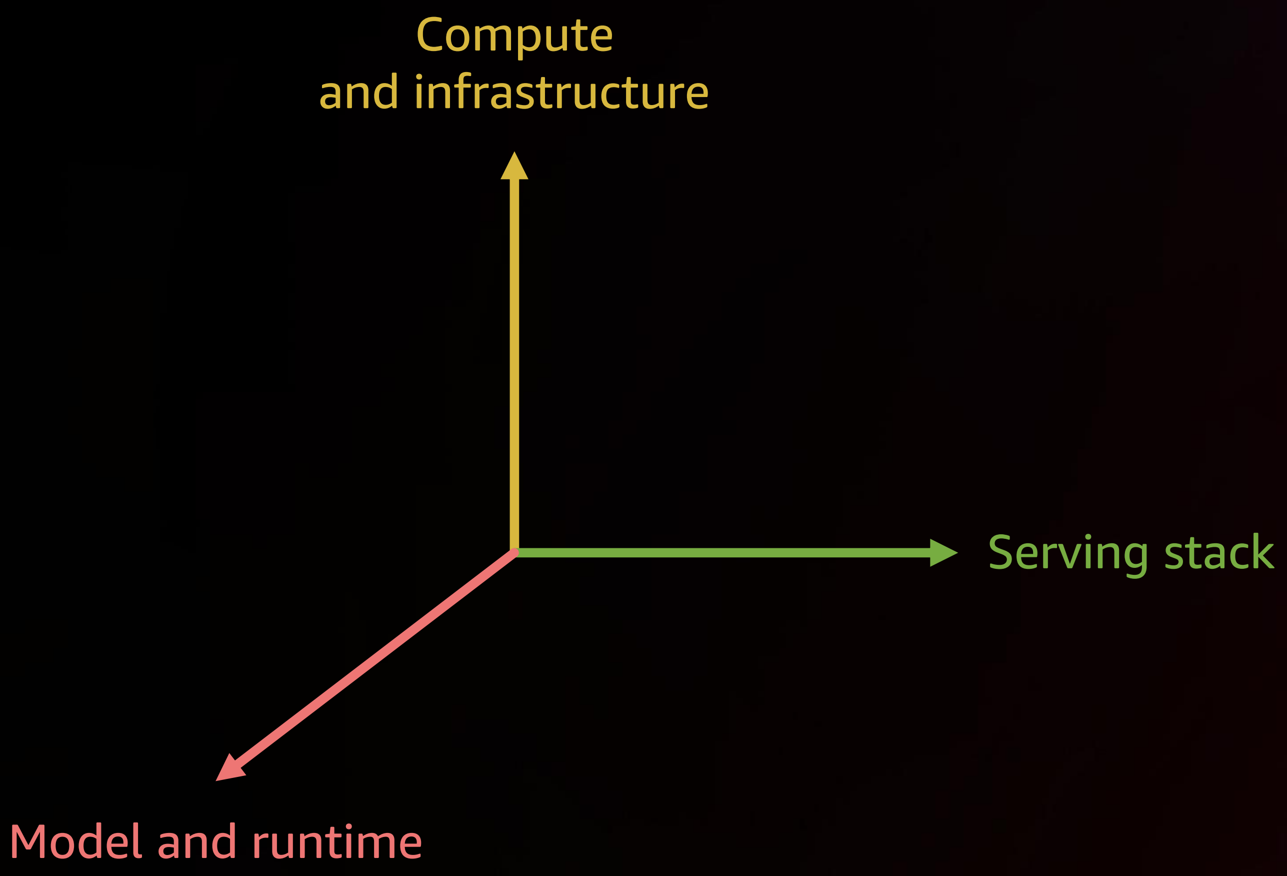
Sample scenario

Lucinda, ML Engineer at ACME Medical Clinic, has been asked to deploy an UNet image segmentation model on Amazon SageMaker for real-time inference.

The inference endpoint should be able to serve 700 TPS, with <150ms latency at P95, with the lowest cost possible.

The solution must be highly available, and able to scale automatically based on the traffic patterns.

Optimization dimensions



CPU/GPU instances, custom chips (AWS Inferentia)
Networking configuration
SageMaker fully-managed deployment options (multi-model, multi-container, and more)

Custom stack (such as Nginx > Gunicorn > Flask)
TorchServe, TFS, MMS, Nvidia Triton
Configure dynamic batching, # of workers, and more

Model compression (pruning, quantization, and more)
Model compilation (TVM, TreeLite, TensorRT, AWS Neuron, Amazon SageMaker Neo)

Amazon SageMaker inference stack

Amazon SageMaker

Amazon SageMaker

SAGEMAKER STUDIO IDE

- Real-time inference
- Async inference
- Serverless inference
- Batch inference
- Multi-model endpoints
- Multi-container endpoints
- Inference DAG and pipelines
- Manage and version models
- MLOps
- Model monitoring
- Metrics and logging in CloudWatch

SageMaker JumpStart

FRAMEWORKS

TensorFlow · PyTorch · mxnet · Keras · scikit learn · GLUON · ONNX

MODEL SERVERS

AWS Deep Learning Containers	TensorFlow Serving	TorchServe	NVIDIA Triton Inference Server	AWS Multi Model Server (MMS)

ML COMPUTE INSTANCES & ACCELERATORS

CPUs	GPUs	AWS Inferentia	AWS Graviton (ARM)

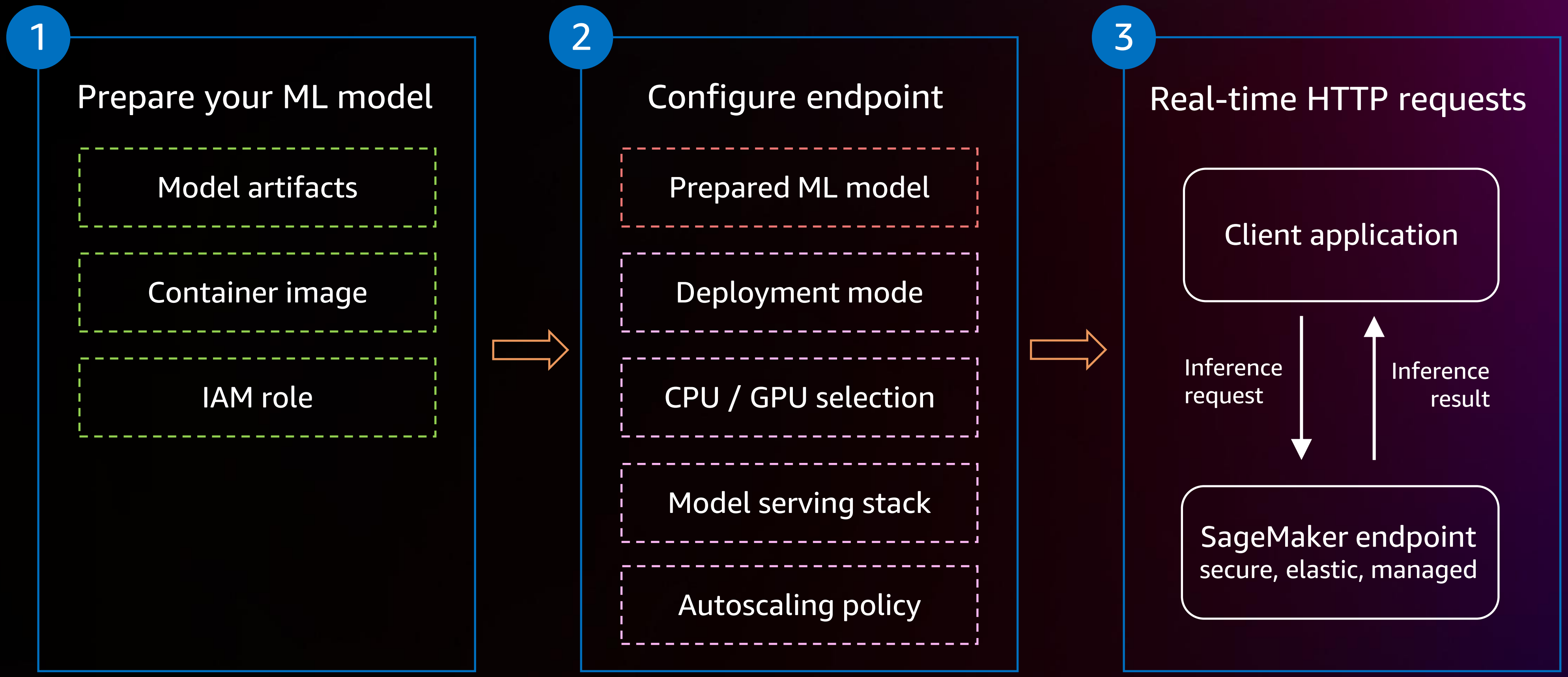
DEEP LEARNING COMPILERS AND RUNTIMES

SageMaker Neo	NVIDIA TensorRT/cuDNN	Intel oneDNN	ARM Compute Library

Compute and infrastructure

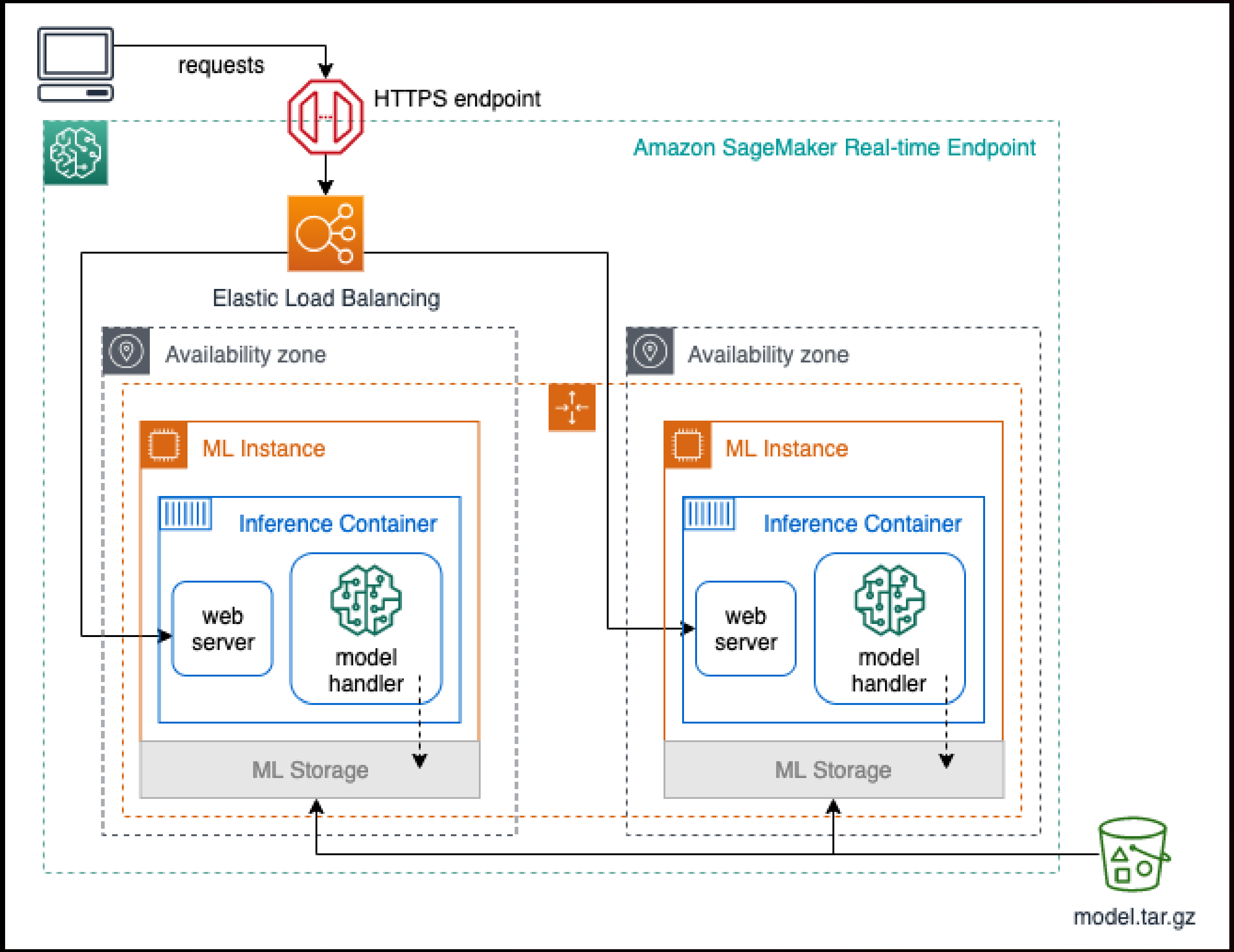
Amazon SageMaker Real-Time Inference

HOW IT WORKS



Amazon SageMaker Real-Time Inference

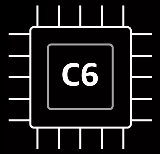
SINGLE MODEL DEPLOYMENT



- Choose the right instance type
- Choose storage size based on model size
- Configure autoscaling
- Deploy endpoints in a VPC
- Create AWS PrivateLink interface endpoint to connect to SageMaker endpoints, with private DNS enabled
- Configure model download and container startup health check timeouts to support large models

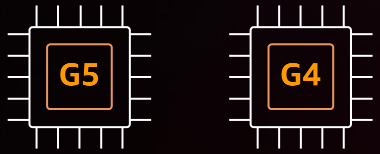
Instance selection

CPU INSTANCES



3rd generation Intel Xeon scalable
Support Intel AVX-512 VNNI

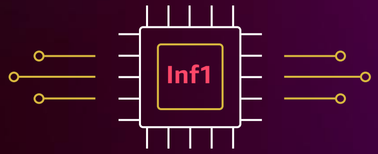
GPU INSTANCES



G5 NVIDIA A10G (24GB) – Up to 8

G4 NVIDIA T4 (16GB) – Up to 4

CUSTOM CHIP



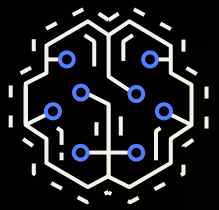
AWS Inferentia chip – Up to 16
Excellent price/performance for ML inference

Inference accelerator Instance type	Throughput	Latency	Cost efficiency	Model support, Programmability	Ease of use	Framework support
CPU-only C6 instance type	○	○	● Smaller models	●	●	●
GPU G5, G4 instance type	●	●	● High utilization	●	◐	●
AWS Inferentia Inf1 instance type	●	●	●	◐	◐	◐

Amazon SageMaker Real-Time Inference

MULTI-MODEL ENDPOINTS

SageMaker Multi-Model Endpoint

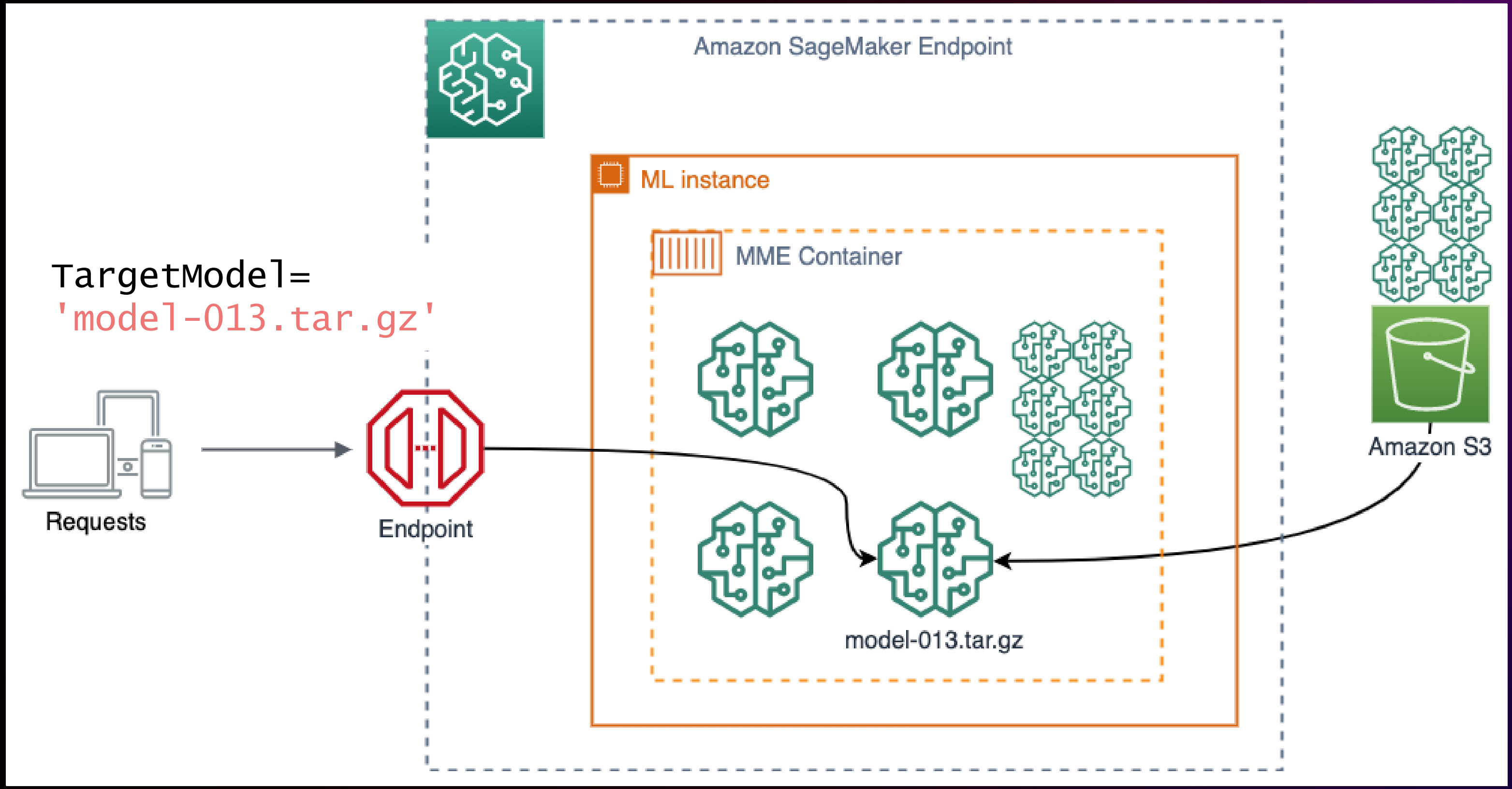


Host multiple models in one container

Direct invocation to target model

Improves resource utilization

Dynamic loading model from Amazon S3



Amazon SageMaker Real-Time Inference

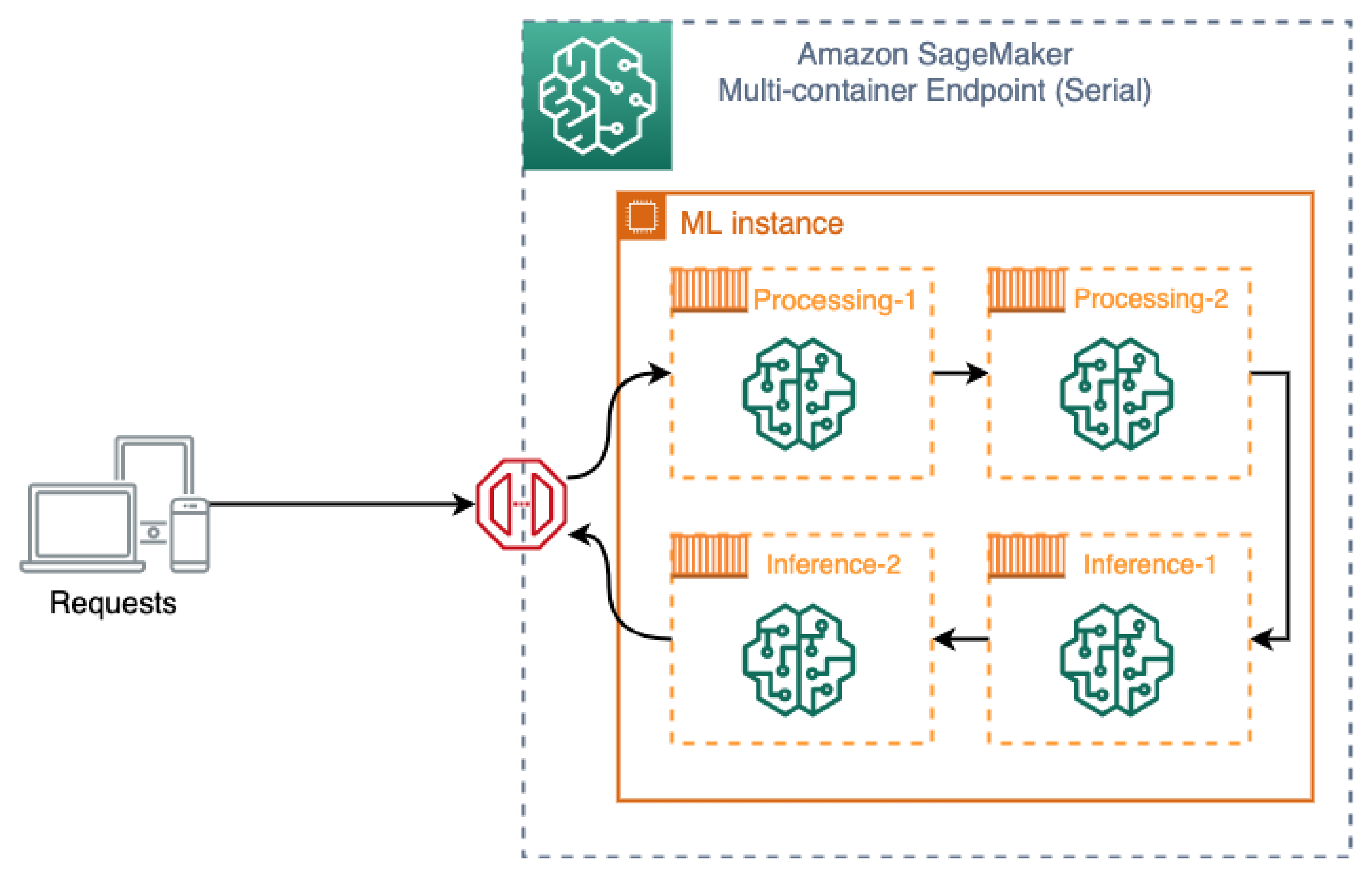
MULTI-CONTAINER ENDPOINTS

SageMaker multi-container endpoint

Host up to 15 distinct containers

Direct or serial invocation

No cold start vs. Multi-Model Endpoint



Autoscaling SageMaker Inference endpoints

Distributes your instances across Availability Zones

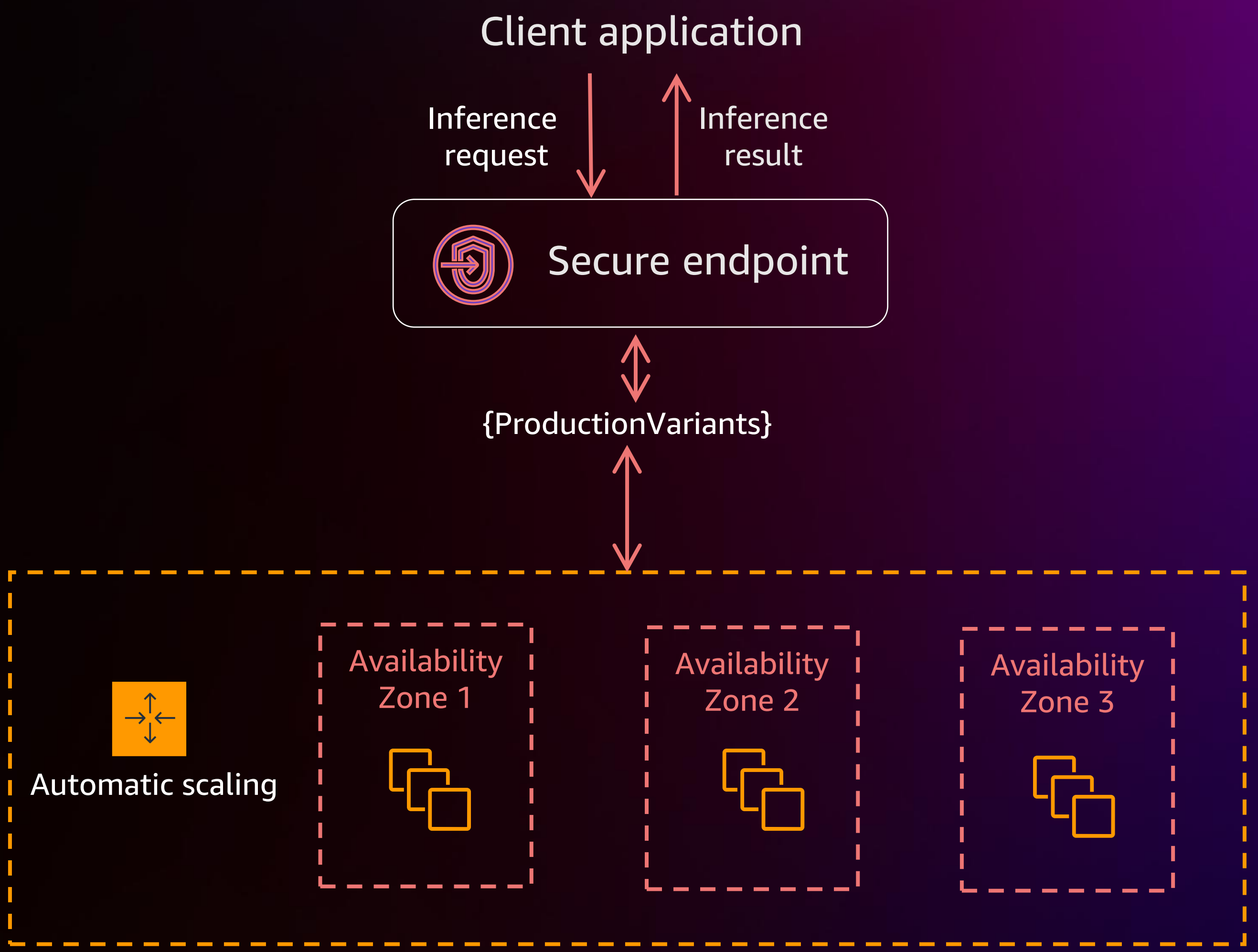
Dynamically adjusts the number of instances

No traffic interruption while instances are being added to or removed

Scale-in and scale-out options suitable for different traffic patterns

Support for predefined and custom metrics for auto scaling policy

Support for cooldown period for scaling in and scaling out



Serving stack

SageMaker Inference containers

FRAMEWORK CONTAINERS AND SERVING STACKS

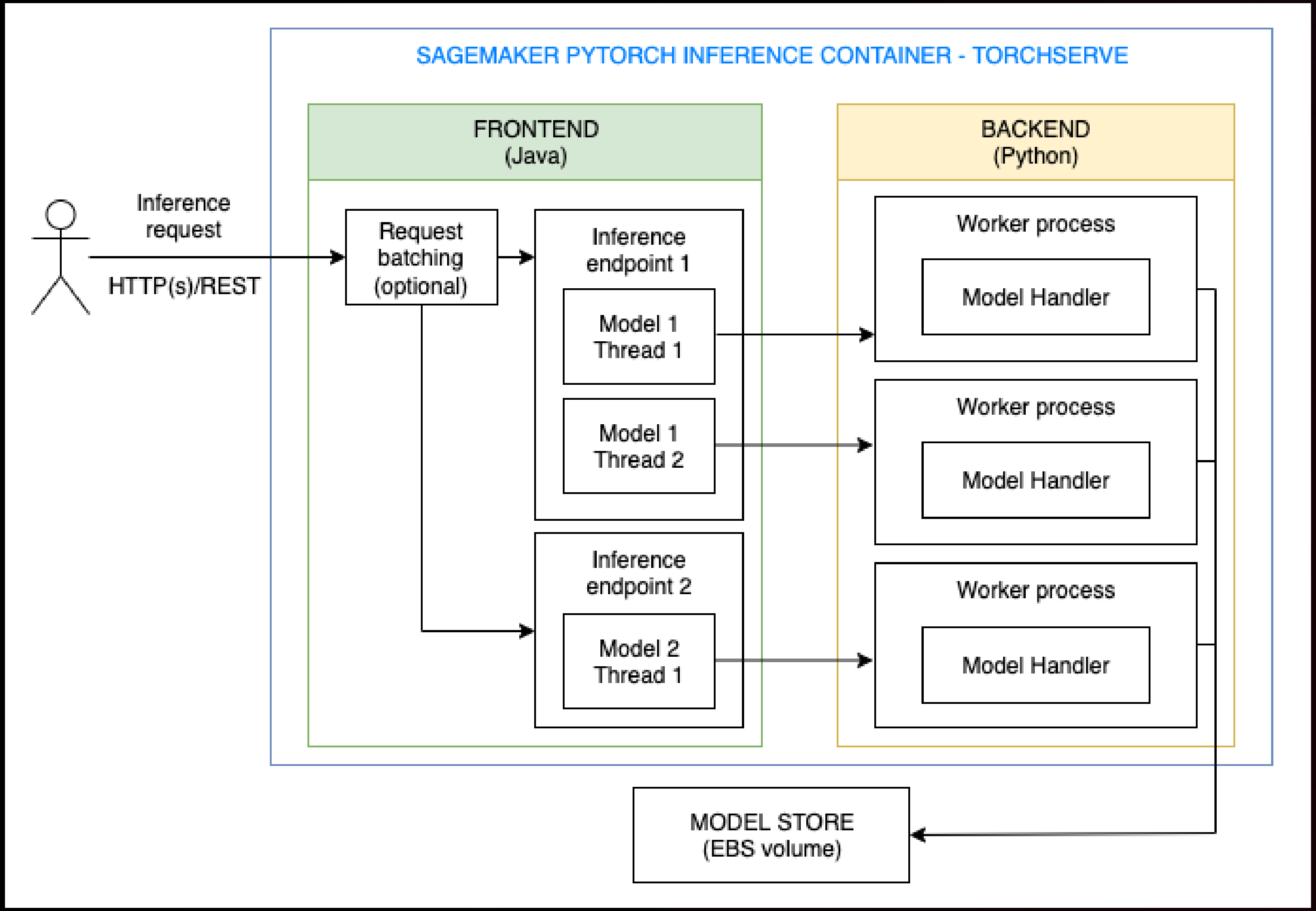
scikit learn	dmlc XGBoost	PyTorch	TensorFlow	NVIDIA TRITON INFERENCE SERVER
sagemaker-sklearn-container	sagemaker-xgboost-container	pytorch-inference (DLC)	tensorflow-inference (DLC)	sagemaker-tritonserver (DLC)
nginx-gunicorn-flask MMS	nginx-gunicorn-flask MMS	TorchServe	nginx-[gunicorn]-TFS	NVIDIA Triton

Hugging Face	Hugging Face	mxnet	AutoGluon	DeepJavaLibrary
huggingface-tensorflow-inference (DLC)	huggingface-pytorch-inference (DLC)	mxnet-inference (DLC)	autogluon-inference (DLC)	djl-serving
MMS	MMS	MMS	MMS	DJL

Additional containers are available to support AWS Neuron and SageMaker Neo runtimes
https://github.com/aws/deep-learning-containers/blob/master/available_images.md

SageMaker PyTorch Inference Container

MODEL SERVING WITH TORCHSERVE



Tune the stack at build time using configuration file

```
enable_envvars_config=true
decode_input_request=false
load_models=ALL
...
```

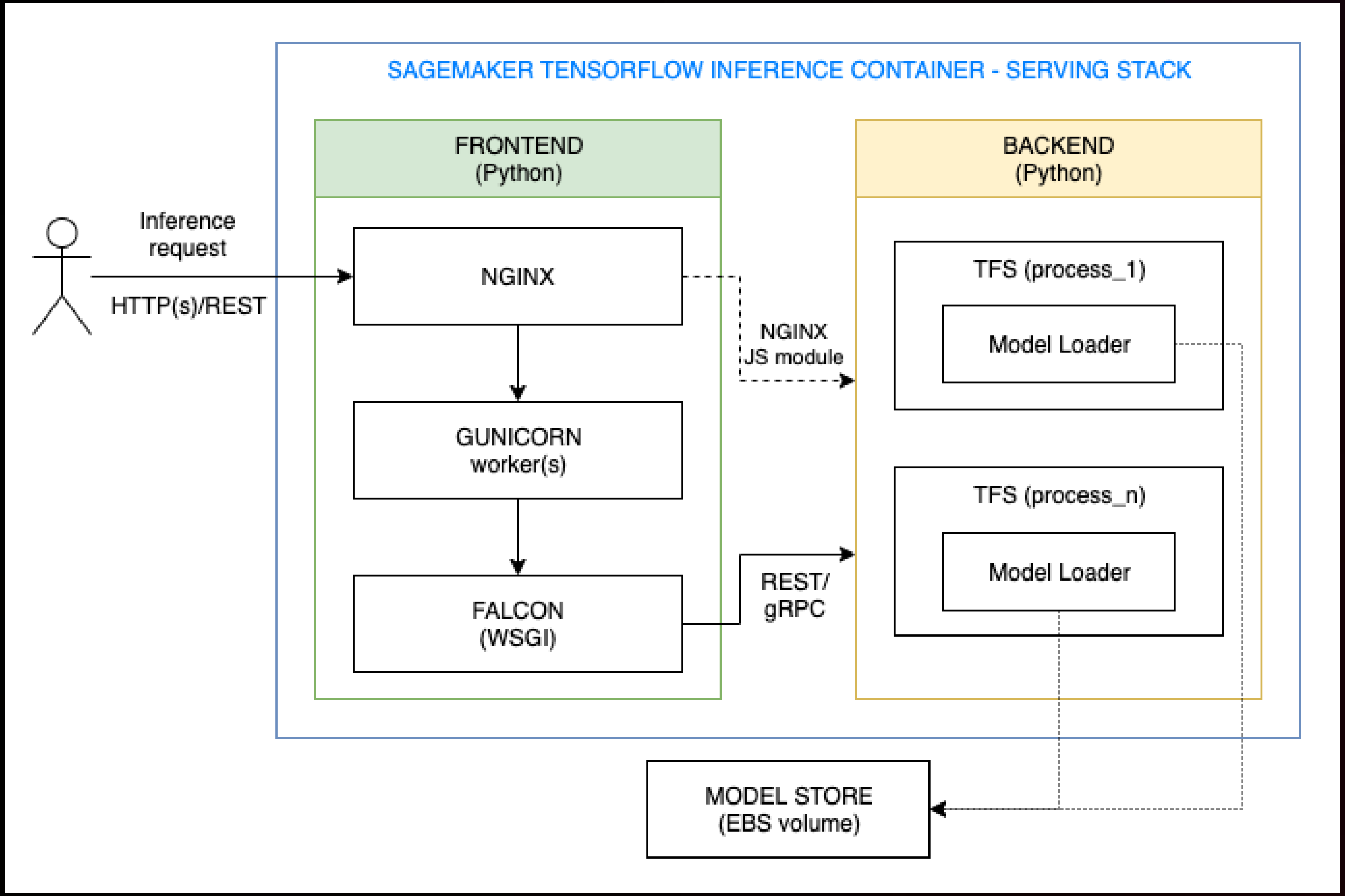
Tune the stack at runtime using environment variables

```
SAGEMAKER_TS_BATCH_SIZE
SAGEMAKER_TS_MAX_BATCH_DELAY
SAGEMAKER_TS_MIN_WORKERS
SAGEMAKER_TS_MAX_WORKERS
SAGEMAKER_TS_RESPONSE_TIMEOUT
...
```

https://aws.amazon.com/blogs/machine-learning/optimize-your-inference-jobs-using-dynamic-batch-inference-with-torchserve-on-amazon-sagemaker/

SageMaker TensorFlow Inference Container

MODEL SERVING WITH TENSORFLOW SERVING (TFS)



Gunicorn application server is enabled only when a custom inference script is provided, or when using SM MME

Tune the stack at runtime using environment variables

```
SAGEMAKER_NGINX_PROXY_READ_TIMEOUT_SECONDS
SAGEMAKER_GUNICORN_TIMEOUT_SECONDS
SAGEMAKER_GUNICORN_LOGLEVEL
SAGEMAKER_GUNICORN_WORKERS
SAGEMAKER_GUNICORN_THREADS
SAGEMAKER_TFS_INSTANCE_COUNT
SAGEMAKER_TFS_ENABLE_BATCHING
OMP_NUM_THREADS
...
```

https://aws.amazon.com/blogs/machine-learning/maximize-tensorflow-performance-on-amazon-sagemaker-endpoints-for-real-time-inference/

Bring your own inference container

MODEL SERVING WITH MULTI MODEL SERVER (MMS)

SageMaker Inference container execution

```
docker run [image] serve
```

Container reserved paths

```
/opt/ml
    |--/model   ← Model artifact copied from S3
```

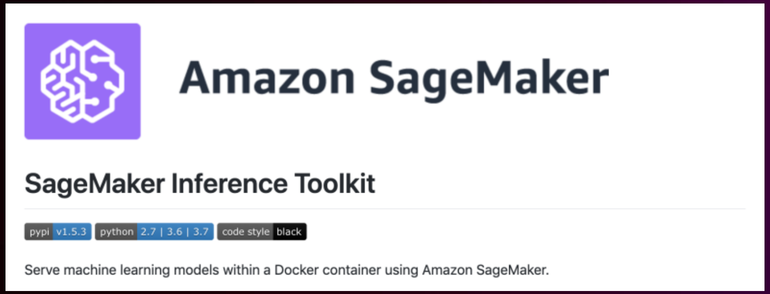
Endpoints to implement in the container

```
/ping:8080                    2s
/invocations:8080             60s
```

For the Docker ENTRYPOINT, use exec form. Example:

```
ENTRYPOINT ["python", "inference_entrypoint.py"]
```

Install Inference Toolkit and use MMS (Multi Model Server)



Starts MMS with the appropriate configuration

Enables support for custom inference scripts packaged within model archive

Allows using requirements file packaged within model archive

Customize with environment variables

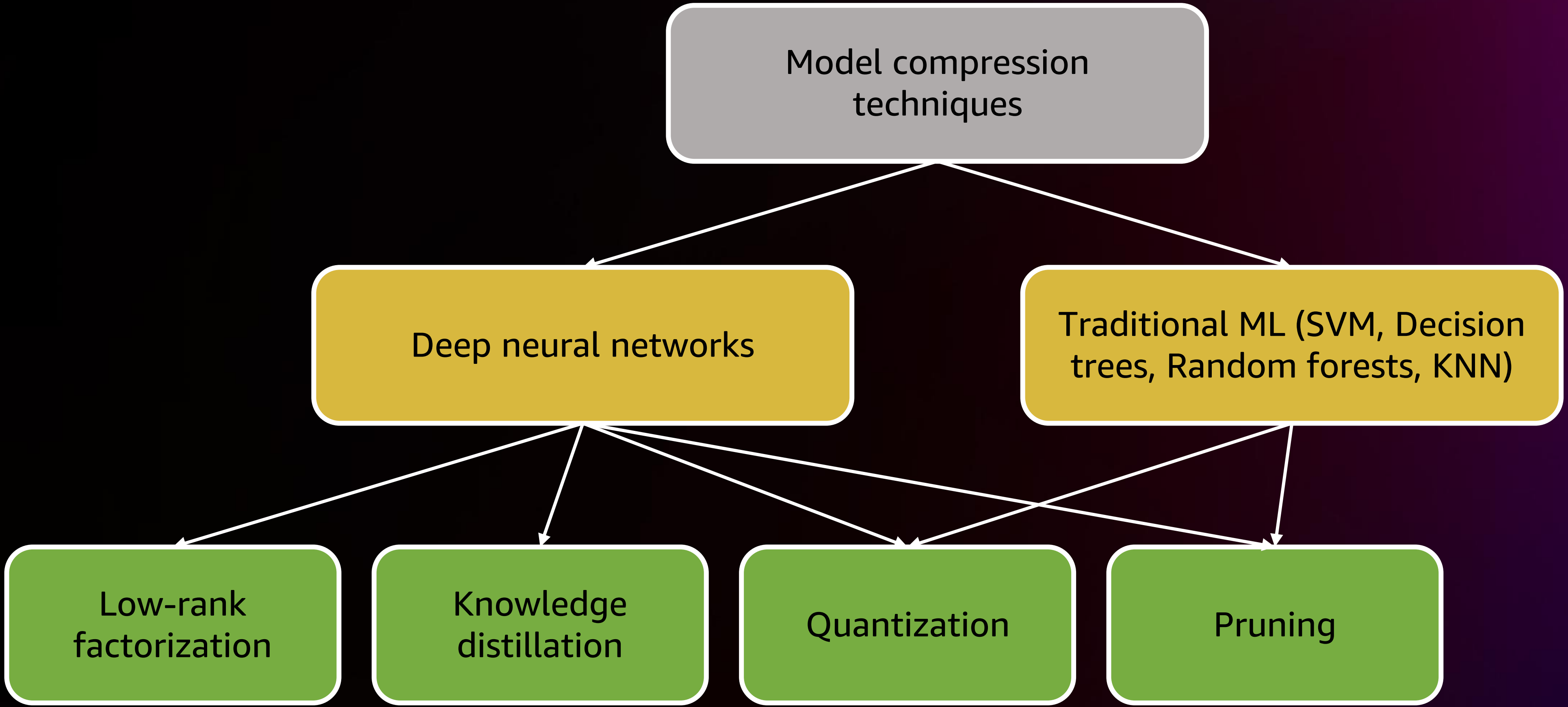
Handles payload encoding/decoding

Configures logging and error handling

Model and runtime

Model compression

COMMON TECHNIQUES



Model compilation

INFERENCE COMPILERS AND RUNTIMES

NVIDIA TensorRT

C++ DL inference optimizer and runtime for faster inference on NVIDIA GPUs
Built on NVIDIA CUDA
Run seamlessly with Triton
Precision calibration, tensor fusion, kernel auto-tuning, and more

tvm

Open-source compiler for CPU, GPU and other accelerators
Support deep learning models in Keras, MXNet, PyTorch, Tensorflow, CoreML, DarkNet, and more

dmlc/treelite

Open-source model compiler for decision tree ensembles
Compatible with XGBoost, LightGBM, SKLearn

AWS Neuron

SDK enabling high-performance deep learning acceleration using AWS Inferentia and Trainium
Support models in PyTorch and Tensorflow

Amazon SageMaker Neo

MODEL COMPILATION AS A SERVICE

Parses model

Convert a model into a common format

Optimizes graph

Detect patterns in the ML model structure to reduce the execution time

Optimizes tensors

Detect patterns in the shape of input data to allocate memory efficiently

Generates code

Use a low-level compiler to generate machine code for each target

Load testing

Load testing

DO IT YOURSELF

LOCUST

Open source, easy to use, scriptable and scalable performance testing tool

Support hundred of thousands of concurrent users

Describe your test as Python code

Web-based UI

APACHE JMeter™

Open source Java application designed to load test functional behavior and measure performance

Load test many different applications/server/protocol types

Full featured test IDE

Generate dynamic HTML reports

Amazon SageMaker Inference Recommender

Run extensive load tests

Get instance type recommendations (based on throughput, latency, and cost)

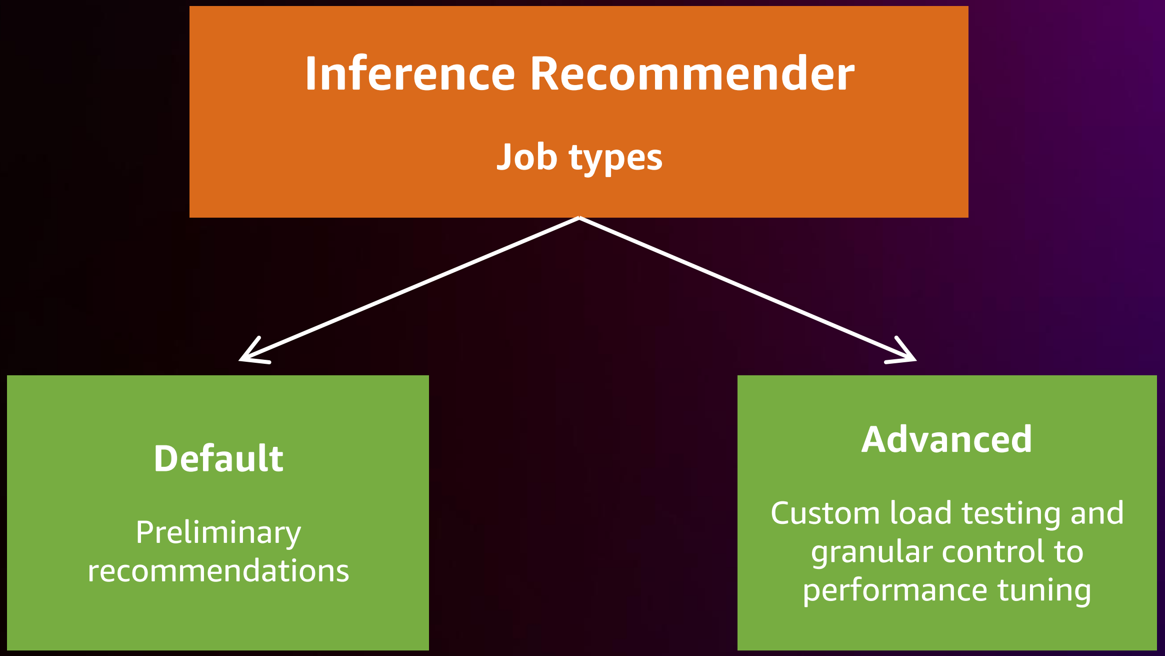
Integrate with model registry

Review performance metrics from SageMaker Studio

Customize your load tests

Fine-tune your model, model server, and containers

Get detailed metrics from Amazon CloudWatch



Real-Time inferencing in a managed SageMaker environment

ARLO TECHNOLOGIES

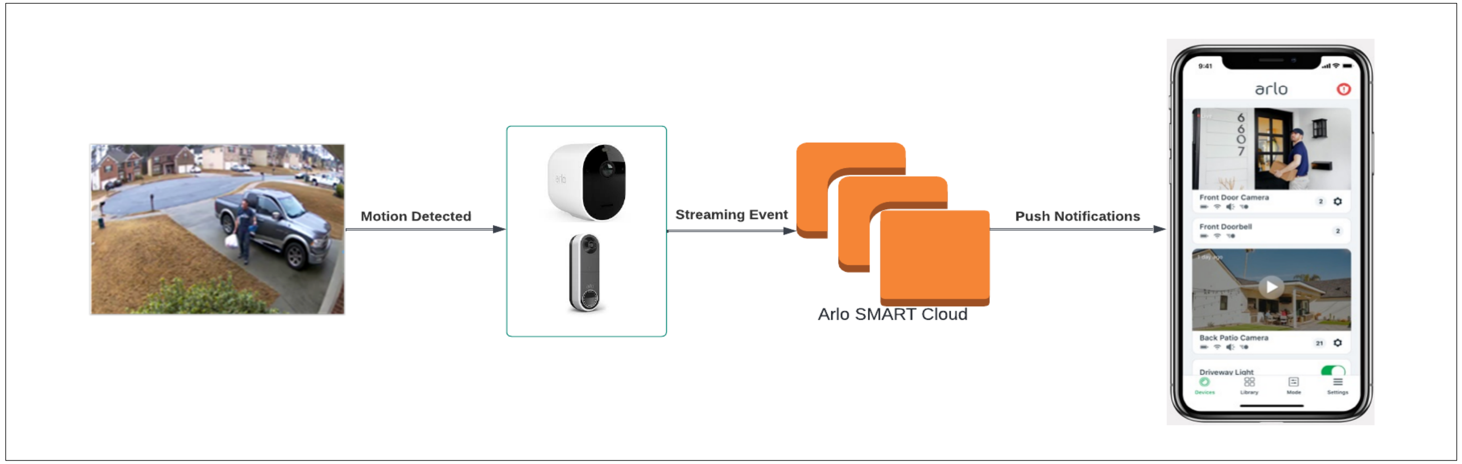
- Arlo is an industry leader in connected cameras and smart home security solutions

- Arlo uses machine learning and computer vision (Arlo SMART) to process millions of video streams each day

- Arlo SMART platform intelligently identifies objects and notifies customers of these AI-detected objects in real time
  - People
  - Animals
  - Packages
  - Vehicles

BUSINESS USE CASE

- Real-Time inferencing in a managed SageMaker environment
- Complete the inferencing in under 150 milliseconds to notify our customers
- Faster readiness checks to reduce the container start time
- Reduce the cost of running inference workload without compromising the response time
- Faster releases to production



SCALE

219M+
Videos uploaded per day

1300+ hours
Video uploaded every minute

75M+
Rich smart notifications per day

3.9B+
API calls per day

1400+
SageMaker instances at peak per day

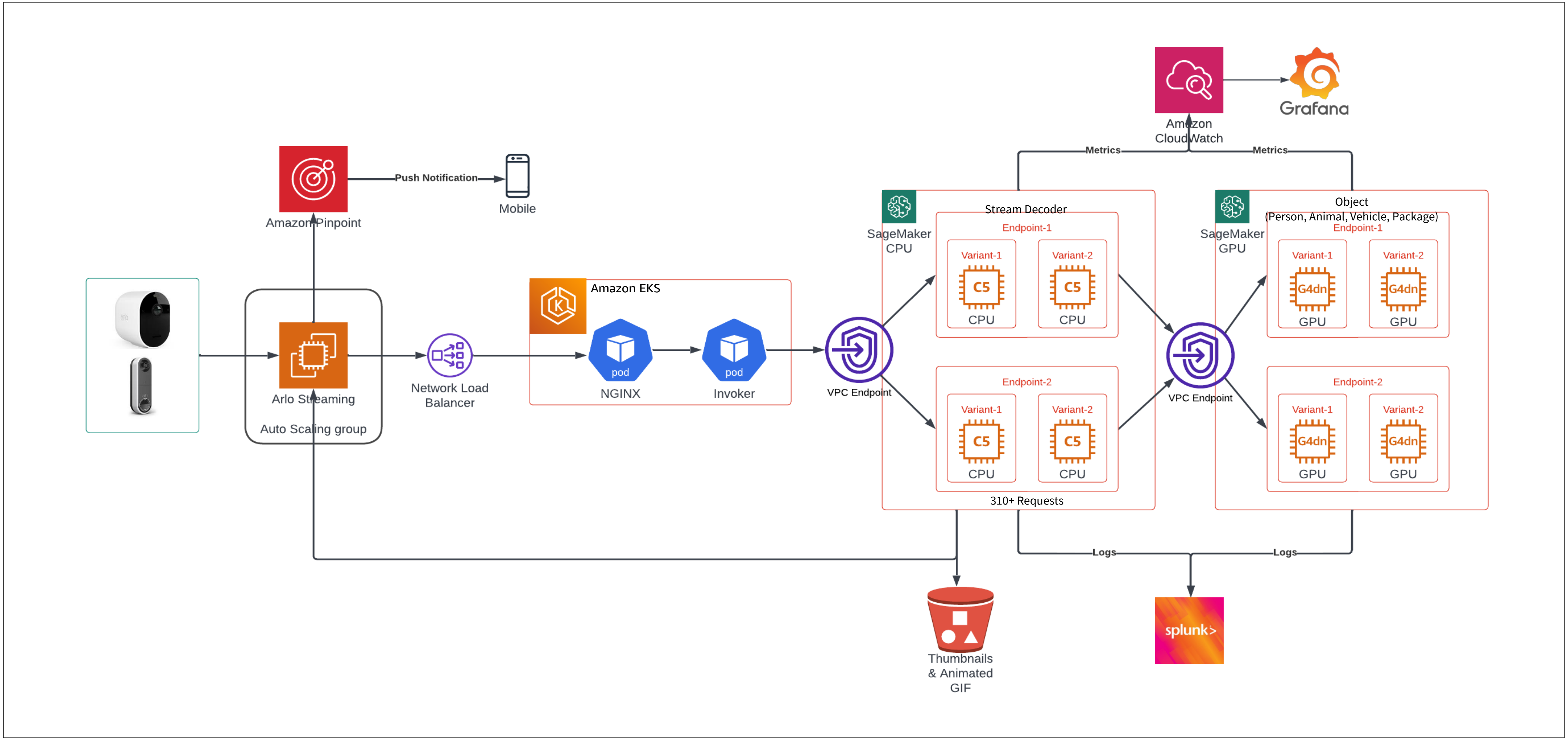
15,000+
CPU at peak per day

600+
GPU at peak per day

310+
Concurrent requests per instance

arlo

APPLICATION ARCHITECTURE

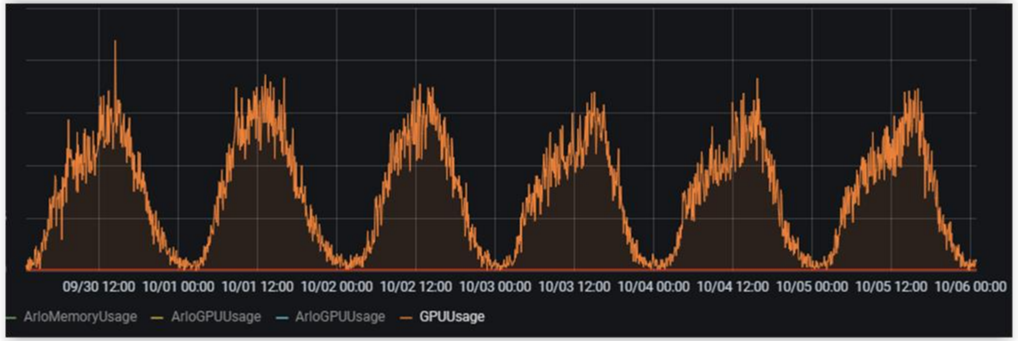


AUTO-SCALING

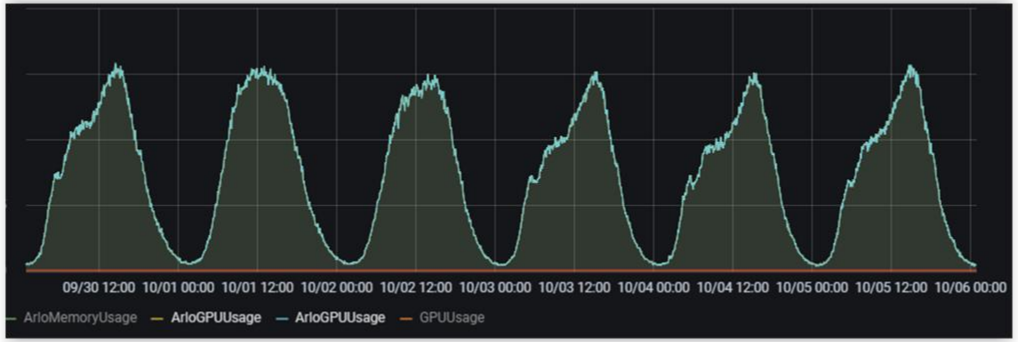
- SageMaker out-of-the-box GPU metrics
  - Publishes 2 data points per second
  - Irregular GPU usage
  - Flapping instances

- Custom code to capture GPU metrics
  - Captured ~15 data points per second
  - Smooth GPU usage
  - Used custom metric to auto-scale GPU instances

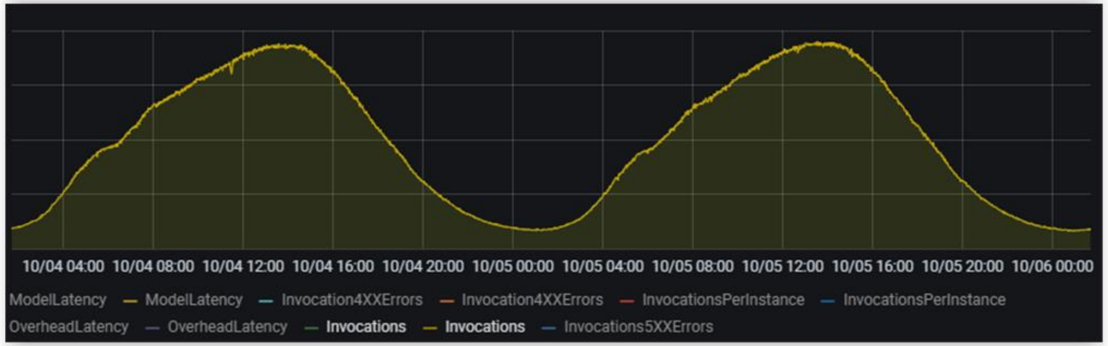
SageMaker – Default GPU usage monitoring



SageMaker – Arlo GPU usage monitoring

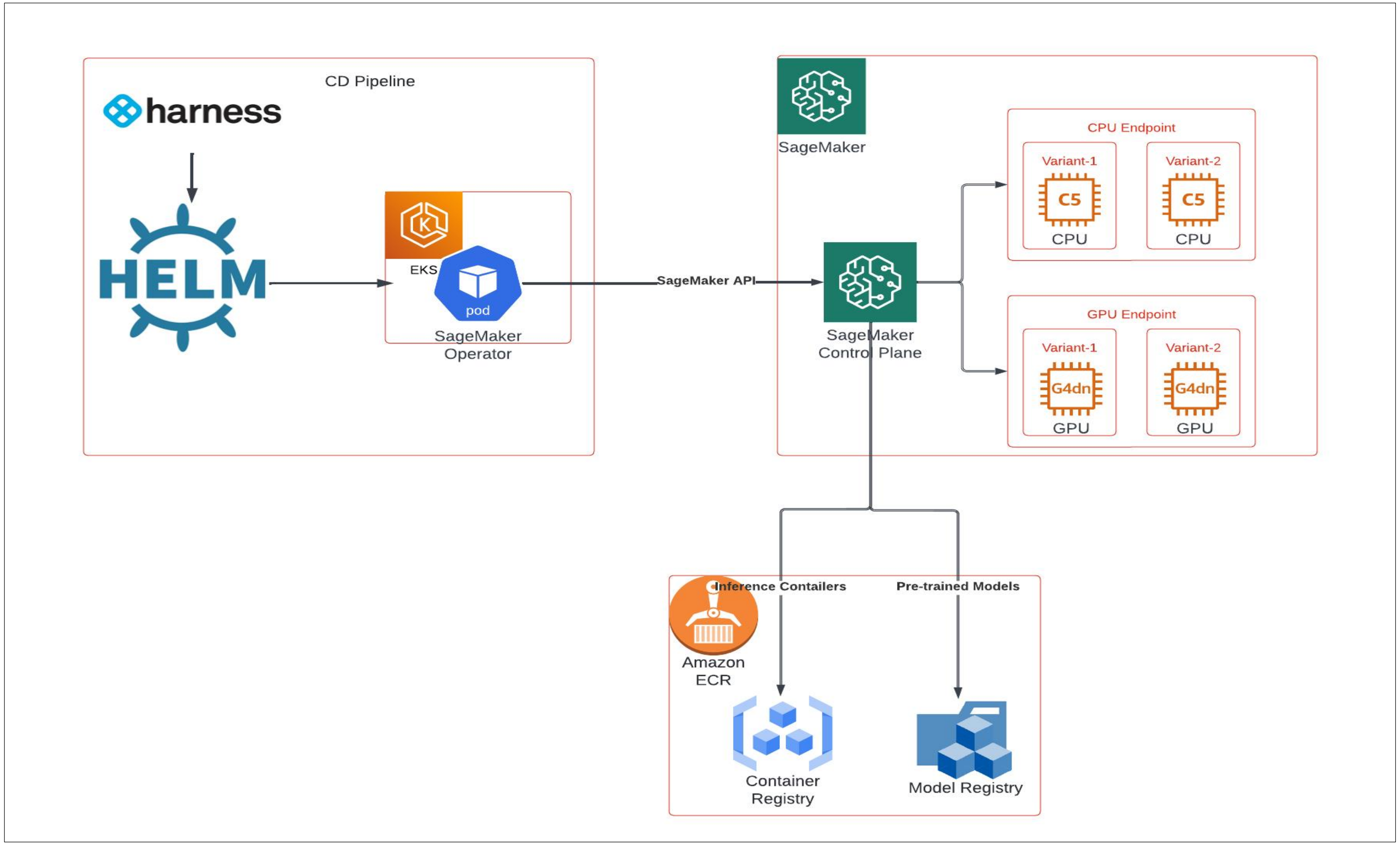


SageMaker – Invocations trend



SageMaker – No of instances trend using Arlo GPU usage

DEPLOYMENT USING SAGEMAKER OPERATOR



- A minimum of 2 endpoints configured for HA
- Each endpoint is configured with 2 variants
- Variant-1 acts as canary in each endpoint

OUTCOMES

- GPU server efficiency improved by 40%
- CPU server efficiency improved by 20%
- Improved container startup time by 50% to auto-scale quickly
- Resource overhead reduced by 10–15% by removing the other agents/plugins
- Intra-regional data transfer reduced by 50%
- Removed the LBs and improved the latency with reduction in cost

WHAT IS NEXT

- Use shared storage to reduce Data IN/OUT cost
- CELL-based architecture
- Run heterogenous instance types to reduce the inferencing cost
- Multi-Metrics based auto-scaling

arlo

SageMaker Inference optimization checklist

- ❑ Select the right instance type – CPU/GPU/AWS Inferentia
- ❑ Maximize instance utilization (MME, multi-container endpoints, Inference pipelines)
- ❑ Use the right automatic scaling policy based on your traffic patterns
- ❑ Performance tune container, model server and application code
- ❑ Compress and compile models
- ❑ Load-test your endpoints
- ❑ Use Amazon SageMaker Inference Recommender to help selecting the right configuration
- ❑ Delete endpoints not in use
- ❑ Use Amazon SageMaker Savings Plans

Thank you!

Please complete the session survey in the **mobile app**