NLP inferences on Amazon SageMaker

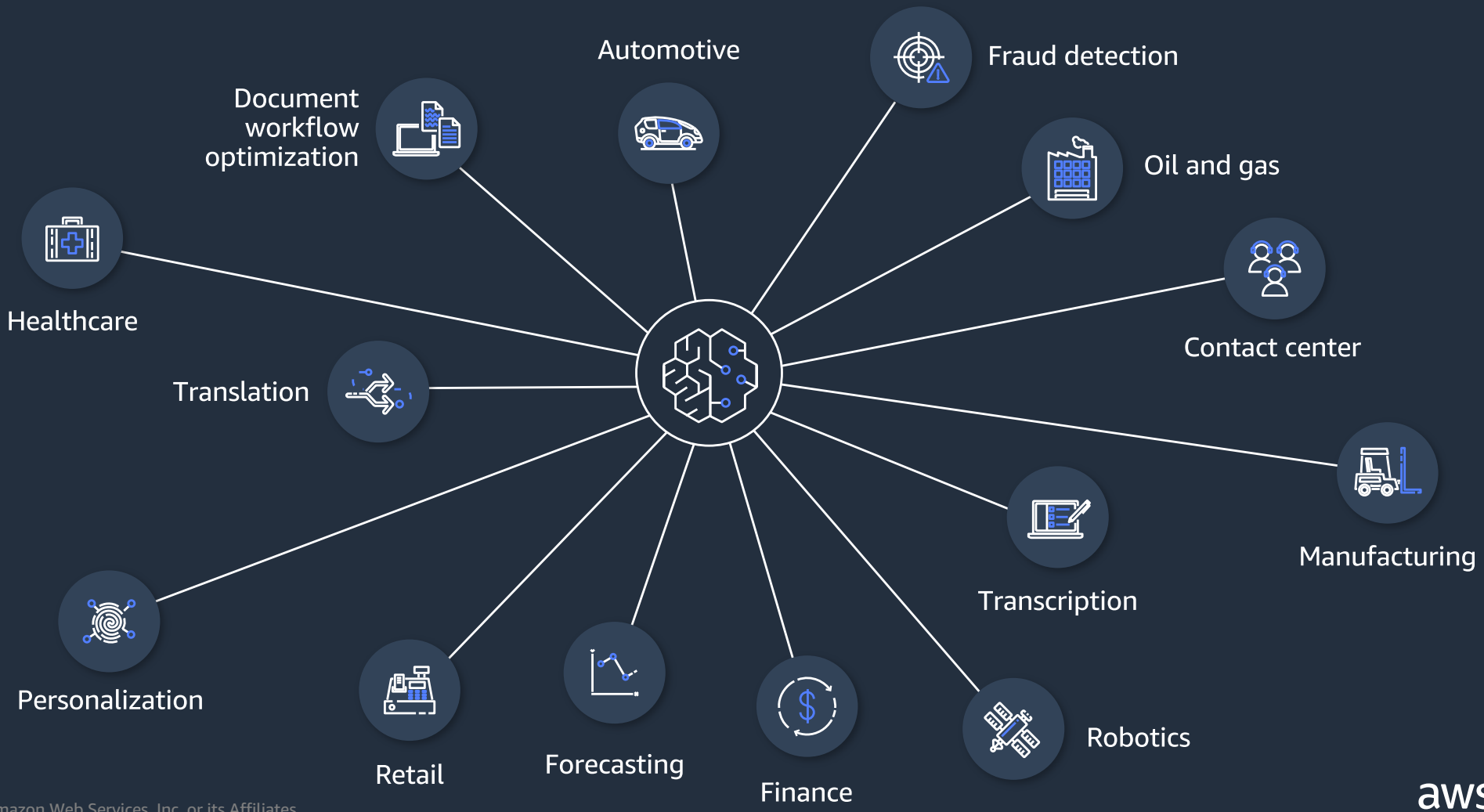
Mia Chang
ML Specialist Solutions Architect

Table of content

- Why talk about inferences
- Customer challenges
- Optimization
  - Hardware. Software. Network
- Security, Monitoring, and Auditability

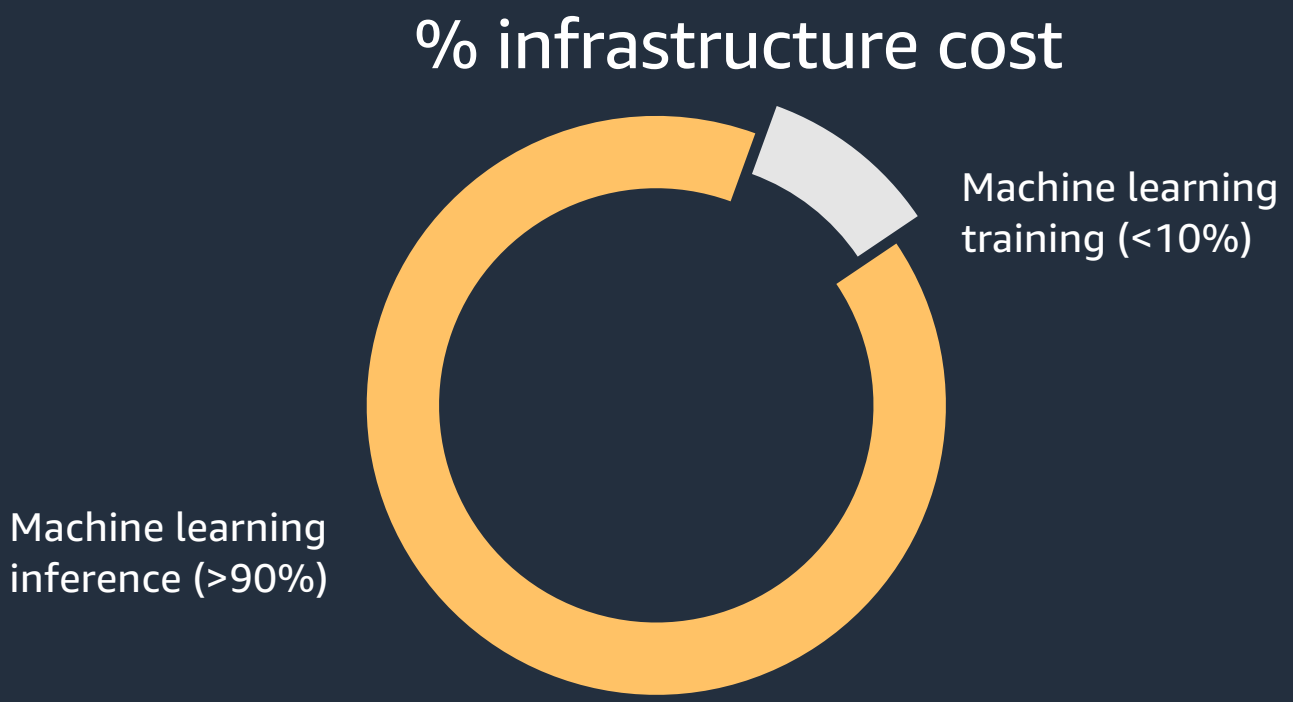
aws

The reach of ML is rapidly growing



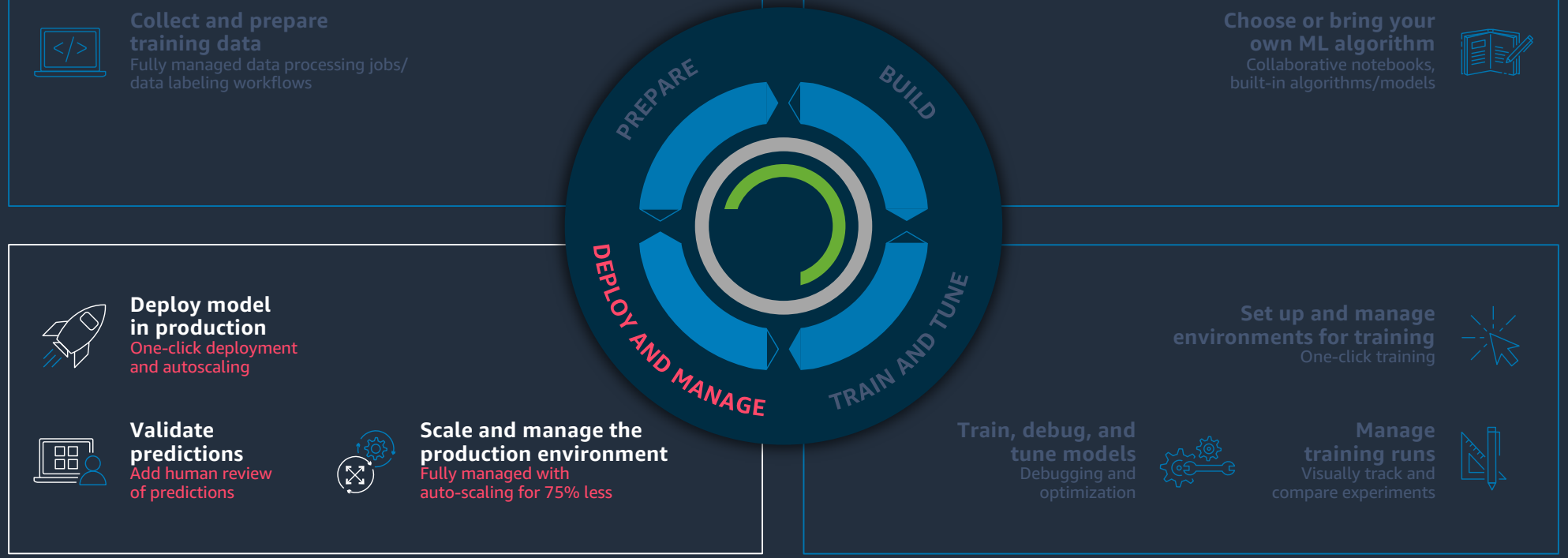
aws

Inference :
the majority of machine learning infrastructure costs



aws

Deploy and manage models



END-TO-END LIFECYCLE

AUTOMATICALLY BUILD AND TRAIN MODELS

aws

Customer Challenges (Model Hosting)

Moving from experiment to production

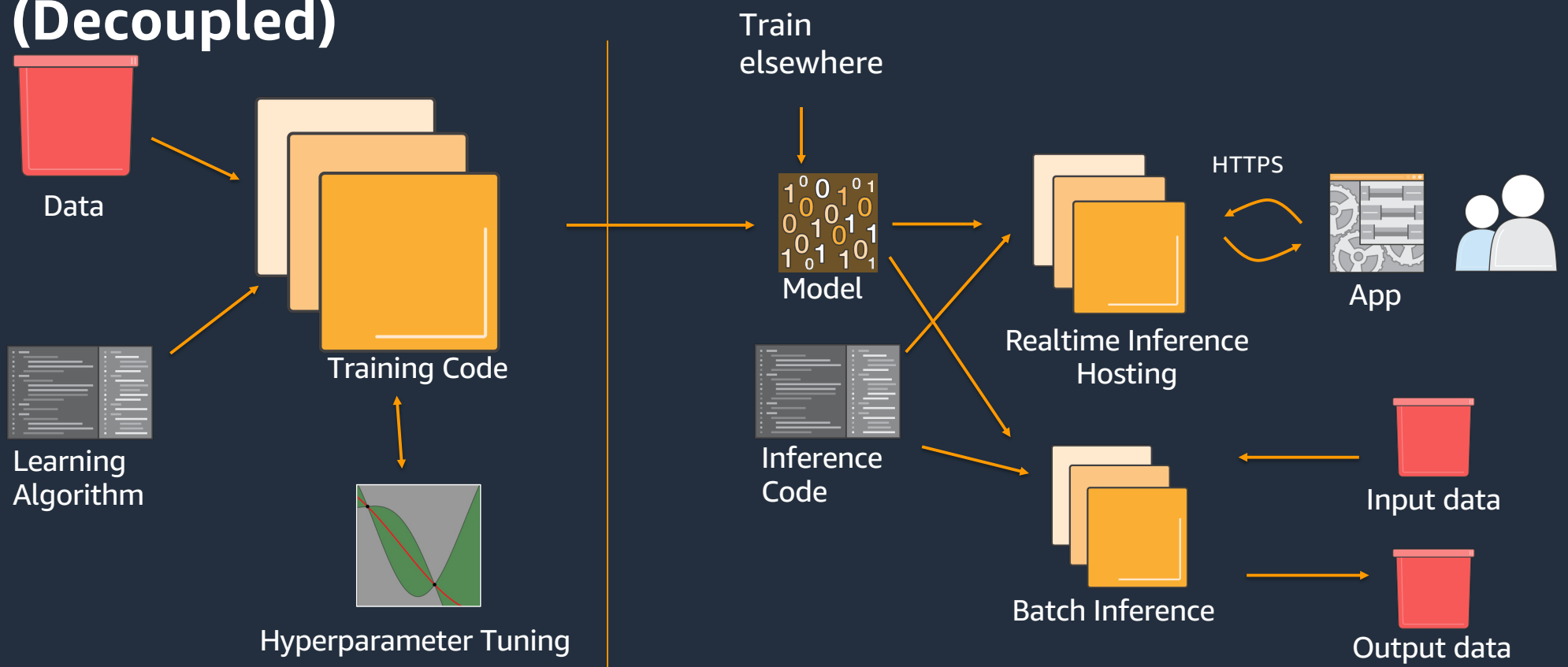
1	2	3	4	5
SLA	Models	Compute	Data	Cost
Latency	Multiple models	CPU, GPU, AI Accelerator	Preprocessing /Feature store	Budget
Throughput	Size, runtime, frameworks	Infrastructure management	Feature drift	Monitoring
Variable workload	Updates, AB testing	Efficient utilization	Large payloads	Control

6 Security and Compliance

© 2021, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Inference on Amazon SageMaker

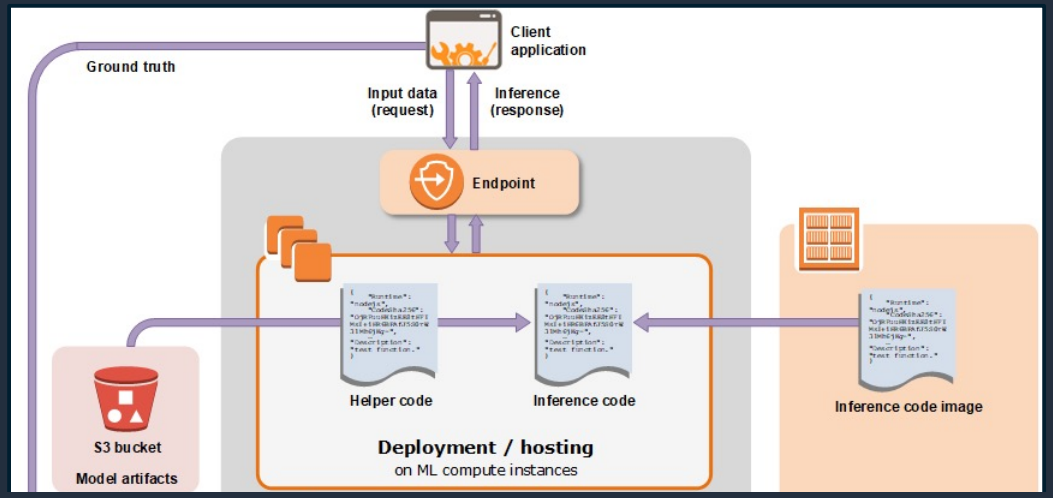
Amazon SageMaker Training and Inference Flow (Decoupled)



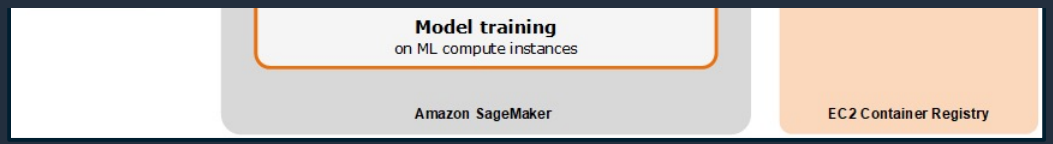
aws

Deploy a model (3 step process)

1. Create a model in SageMaker
   - Model artifacts S3 path
   - Inference container image
   - Inference code
2. Create an endpoint configuration
   - Model name
   - Production variant
   - ML compute instance/s
3. Create a HTTPS endpoint
   - Endpoint configuration
   - Endpoint name

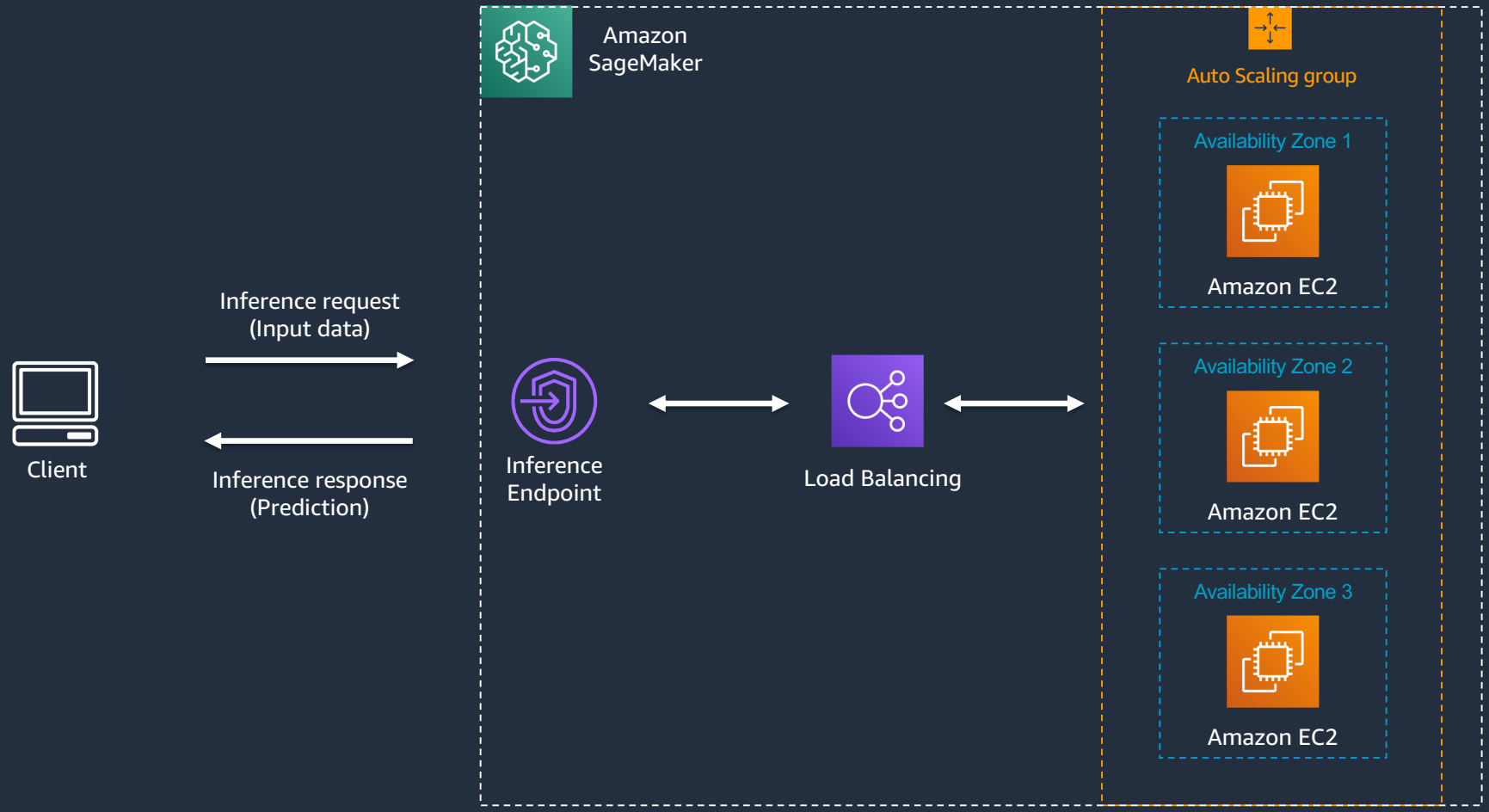


- **Model types**
  - Pre-trained from a model zoo
  - Trained externally
  - Trained with SM built-in algorithms
  - Trained with supported SM frameworks



aws

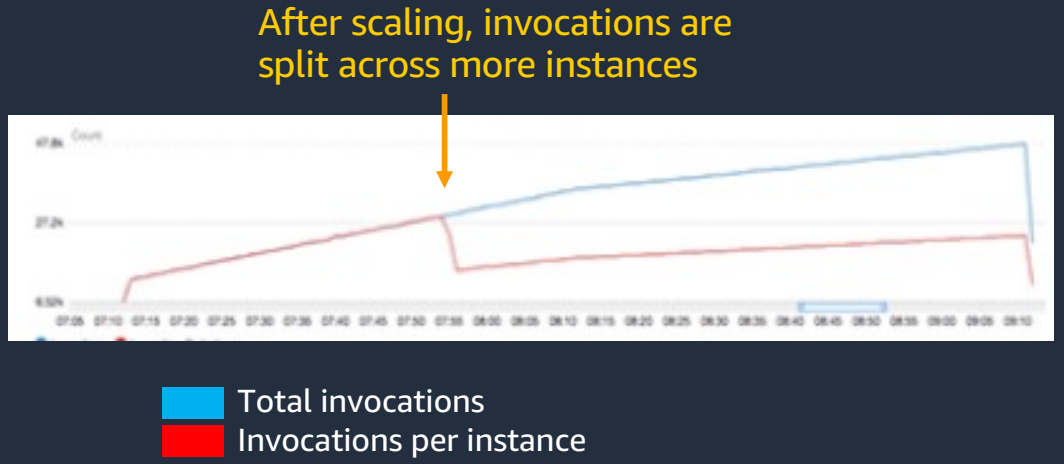
Expose a Real-time endpoint



aws

Autoscaling for real-time inference endpoints

- Provision steady state capacity
- Set minimum, maximum instance counts, scaling criteria
- SageMaker auto-scales up to meet demand, and scales back down
- Significant cost savings



https://amzn.to/3hIcVEy

aws

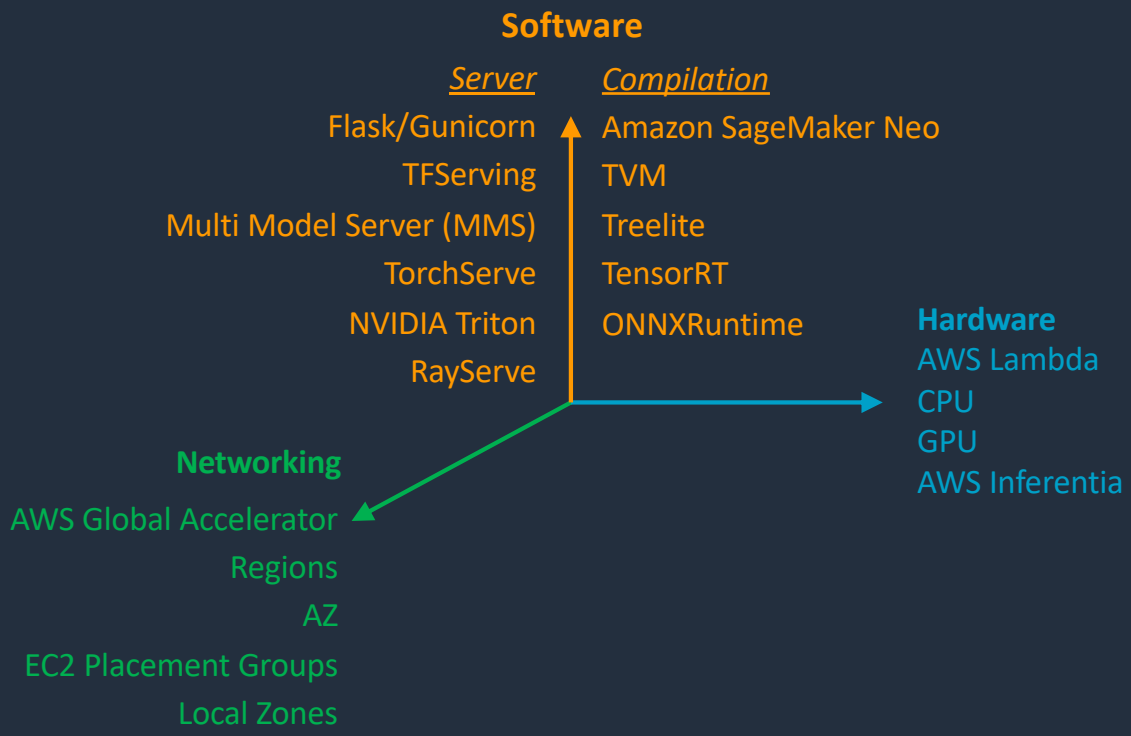
When to use RT Inference vs Batch Transform

	SageMaker Endpoint	Batch Transform
Scaling	Auto-scaling enabled	Configure size of cluster per run
Management	AWS-managed	AWS-managed
Flexibility	Depends on SM Docker	Depends on SM Docker
Cost	Higher: Cluster runs always and can scale based on demand	Lower: Cluster is decommissioned after use
Latency	Lowest latency	3-4 minute wait time

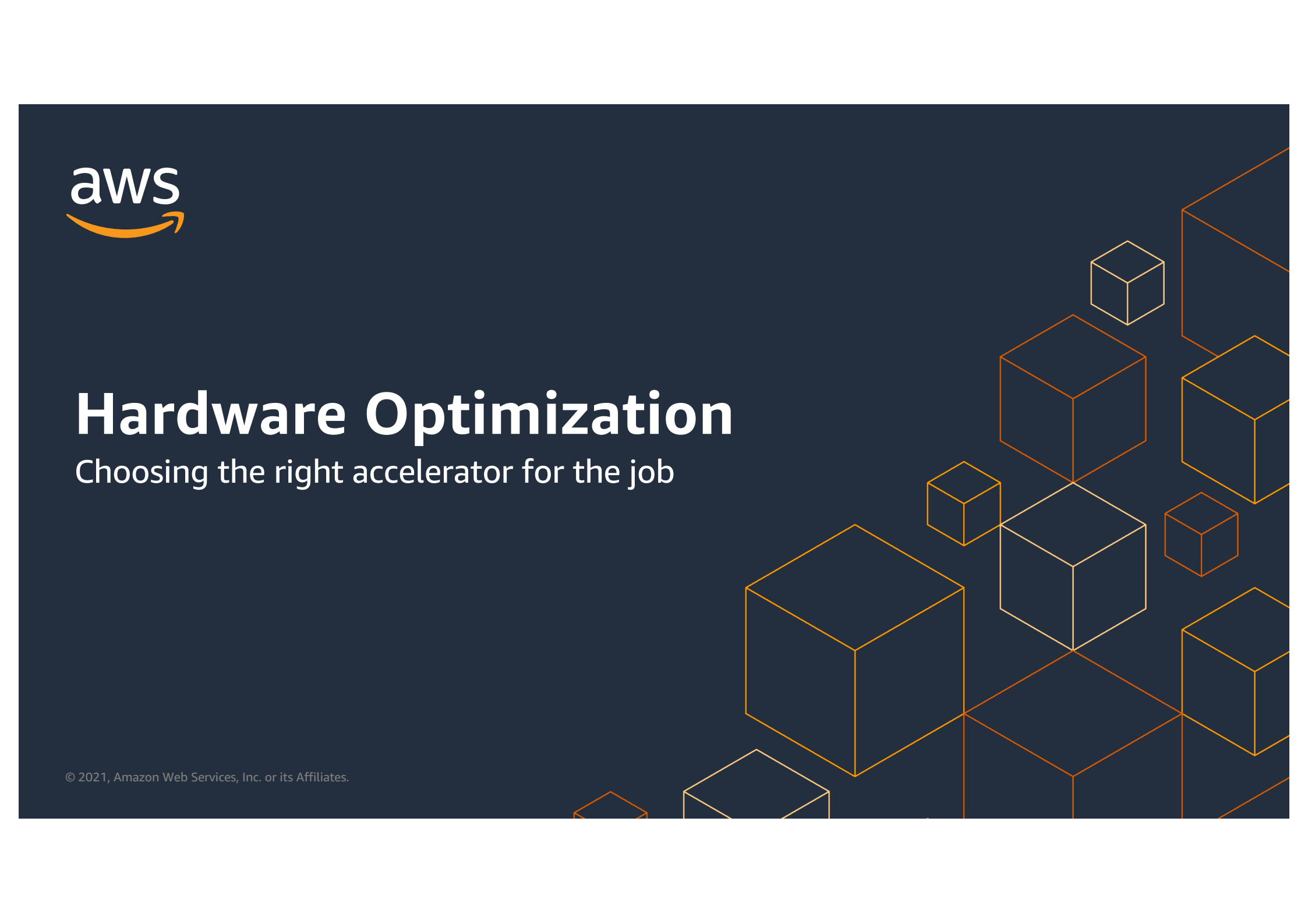
aws

Optimizing Inference

A 3-axis framework for ML inference latency reduction



aws

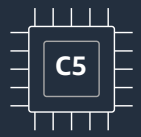


Hardware Optimization

Choosing the right accelerator for the job

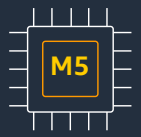
Wide selection of options for cost-effective inference

CPU INSTANCES



Small models,
low throughput

ELASTIC INFERENCE



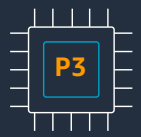
NETWORK ATTACHED

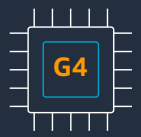
INFERENCE ACCELERATOR

eia1.medium

Mid-sized models, low-latency
budget with tolerance limits

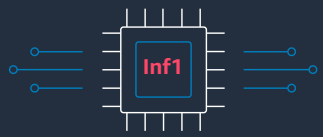
GPU INSTANCES





Large models, high throughput,
and low-latency access to CUDA

CUSTOM CHIP



Inf1: High throughput, high performance,
and lowest cost in the cloud

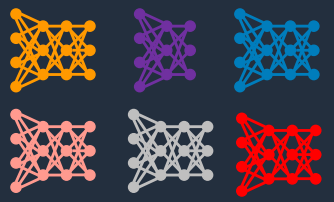
aws

Four key considerations

1. **Target performance**
   Target throughput under desired latency

2. **Cost efficiency**
   Maximizing instance utilization to reduce cost / inference

3. **Model and framework support**
   Support for custom and popular models (ResNet, BERT etc.)



4. **Ease of use and deployment**
   Easy to use compiler and runtime toolchain
   Support for popular deep learning frameworks

TensorFlow
mxnet
PyTorch
aws

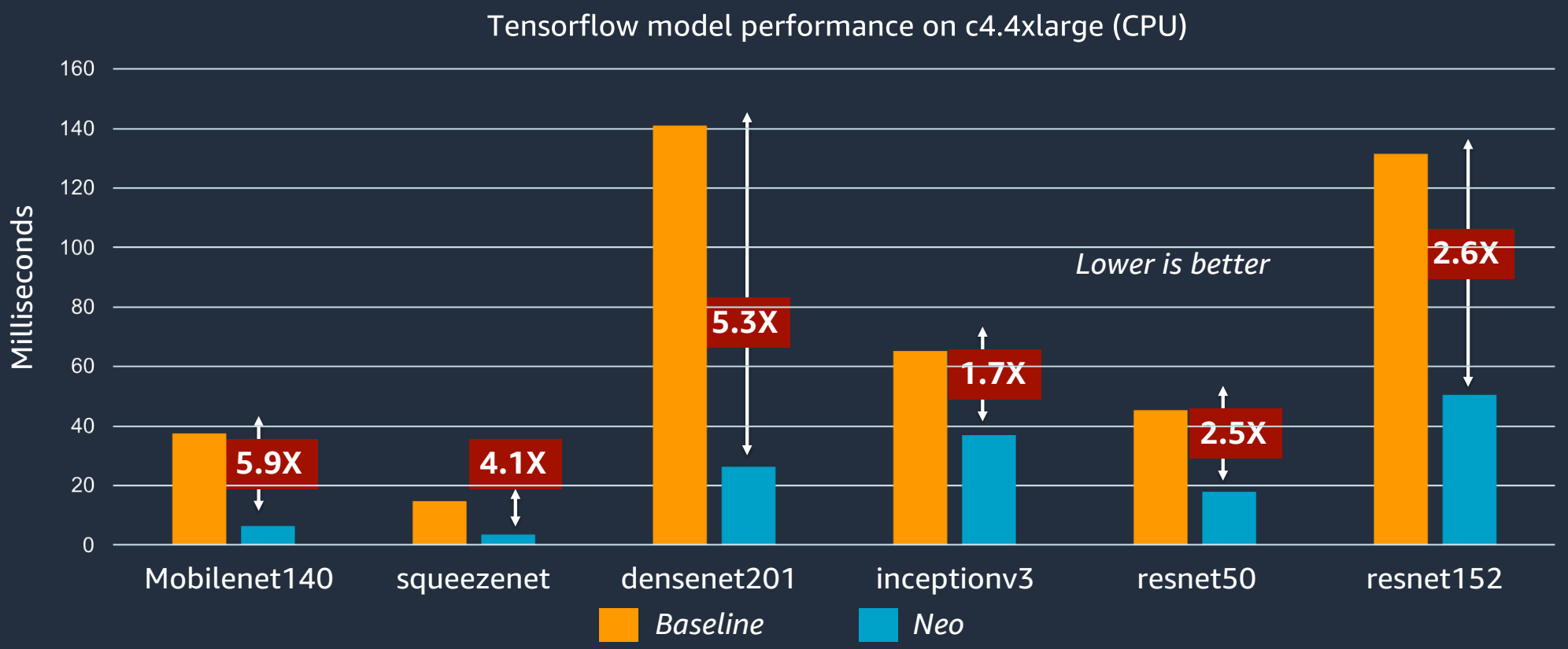
At a glance: instances and accelerators for ML inference

Inference accelerator instance type	Throughput	Latency	Cost efficiency	Model support and programmability	Ease of use	Framework support
CPU-only C4, C5 instance types	○	○	● Smaller models	●	●	●
GPU G4 instance type	●	●	● High-utilization	●	◑	●
Elastic Inference CPU instances + EI accelerator	◑	◑	●	◑	◑	◑
AWS Inferentia Inf1 instance type	●	●	●	◑	◑	◑

aws


aws
Model Optimization
Compile the model to optimize its performances
© 2021, Amazon Web Services, Inc. or its Affiliates.

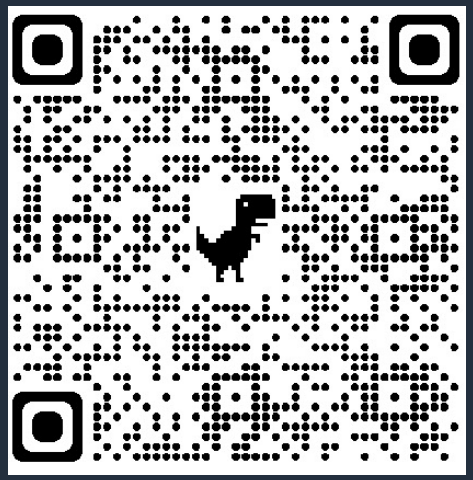
Improve inference performance with Neo



Baseline: TensorFlow optimized with nGraph and MKL-DNN

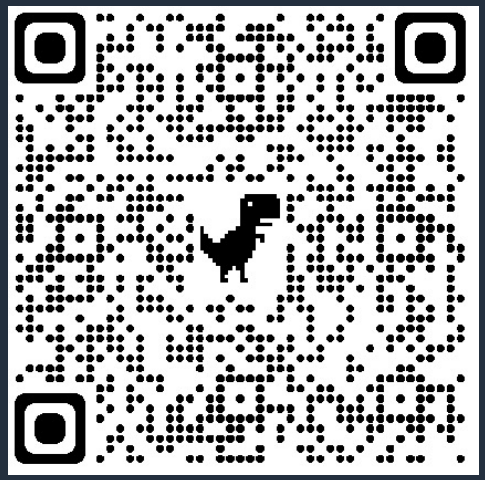
aws

How do I know if my model is supported by Neo?



Supported model/frameworks
for **CPU and GPU cloud instances**

Link



Supported model/frameworks
for **Inferentia instances**

Link

aws

How to compile a model with SageMaker SDK

```
pth_model = PyTorchModel(
    model_data=model_data,
    entry_point='compiled-inference.py',
    source_dir='code',
    role=role,
    framework_version='1.7',
    py_version='py3'
)
```

```
compiled_model = pth_model.compile(
    target_instance_family='ml_inf1',
    input_shape={"input0": [1, 3, 224, 224]},
    output_path=output_path,
    role=role,
    job_name=name_from_base(f'pytorch-resnet50-inf1'),
    compile_max_run=1000
)
```

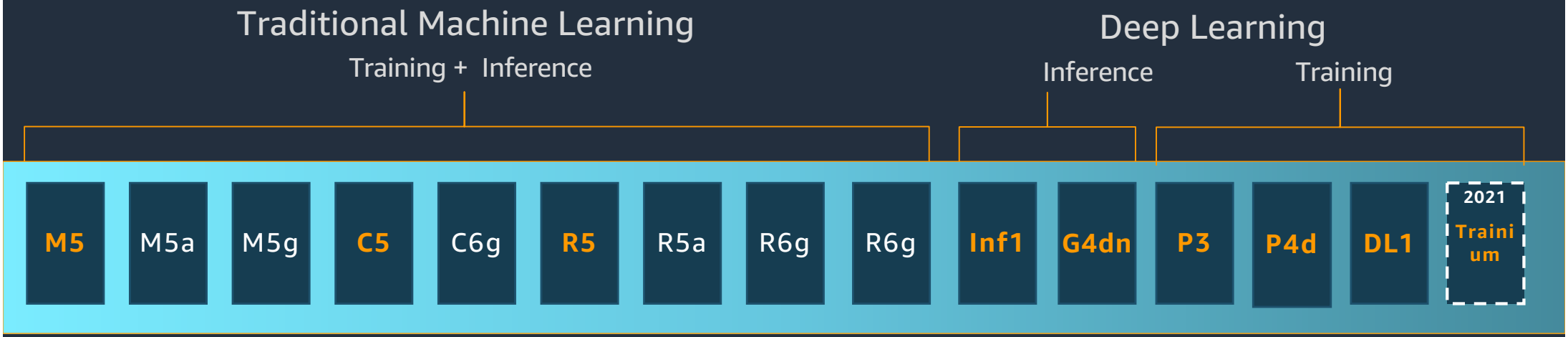

aws

Amazon EC2 Inf1 Instances

Featuring Amazon Inferentia Chips

Broadest and deepest compute infrastructure for AI/ML

Choice of CPUs, GPUs, and accelerators for your performance and budget needs



Cascade Lake CPU
Skylake CPU
Habana® Gaudi® Accelerators

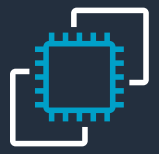
EPYC CPU

Graviton CPU
Inferentia Chip

A100, V100, T4 GPUs

Inf1 are built from the ground up by AWS

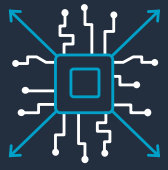
Purpose built for ML inference



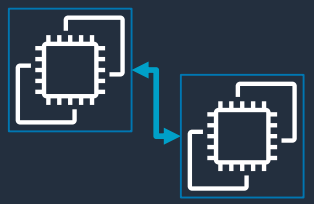
AWS Nitro

AWS Inferentia

High performance



AWS Custom 2nd Gen Intel Xeon Scalable Processors



100Gbps Networking

Low cost

aws

Inf1 Instance Choices

Instance size	vCPUs	Memory (GIB)	Inferentia Chips	Storage	Network B/W	EBS B/W
inf1.xlarge	4	8	1	EBS	Up to 25 Gbps	Up to 3.5 Gbps
inf1.2xlarge	8	16	1	EBS	Up to 25 Gbps	Up to 3.5 Gbps
inf1.6xlarge	24	48	4	EBS	25 Gbps	3.5 Gbps
inf1.24xlarge	96	192	16	EBS	100 Gbps	19 Gbps

- 1 to 16 Inferentia chips per instance
- Up to 192 GB of Memory

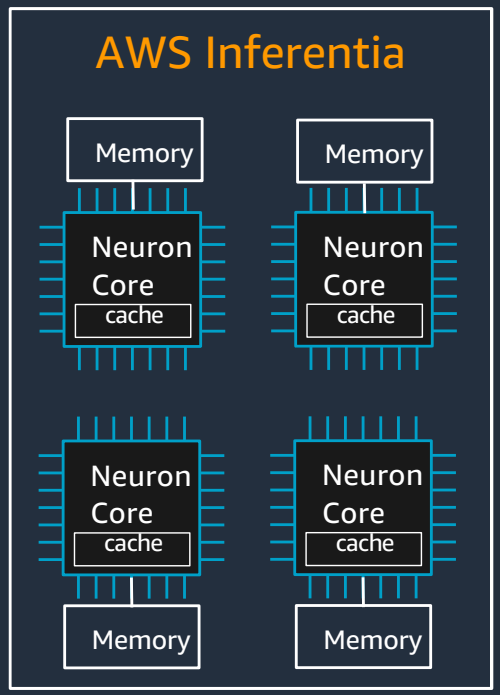
- AWS 2nd Gen Intel Xeon Host Processors
- Up to 100Gbps networking bandwidth

aws

AWS Inferentia quick tour

First ML chip designed by Amazon

- 4 NeuronCores with up to 128 TOPS
- 2-stage memory hierarchy: Large on-chip cache + 8 GB DRAM
- Supports FP16, BF16, INT8 data types with mixed precision
- 1 to 16 Inferentia cores per instance with high-speed interconnect
- Optimized for high throughput and real-time low latency



aws

AWS Neuron

High-performance Software Development Kit (SDK)

Neuron Compiler

Neuron Runtime

Profiling tools

Easy to get started

Integrated with major frameworks

PyTorch TensorFlow mxnet

Minimal Code Change

Deploy existing models with minimal code changes. Maintain hardware portability without dependency on AWS software

Documentation, Examples & Support

github.com/aws/aws-neuron-sdk

aws

Native integration with minimum code change

Run HuggingFace Transformers models on Inferentia with single line of code added

```
from transformers import BertTokenizer, BertModel
import torch
import torch_neuron

tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')
model = BertModel.from_pretrained('bert-base-uncased',return_dict=False)

inputs = tokenizer("Hello, my dog is cute", return_tensors="pt")

neuron_model = torch.neuron.trace(model,
                                  example_inputs = (inputs['input_ids'],inputs['attention_mask']),
                                  verbose=1)

outputs = neuron_model(*(inputs['input_ids'],inputs['attention_mask']))
```

Sample code taken from docstring for BERT Base model:
https://huggingface.co/transformers/model_doc/bert.html#bertmodel

aws

Best price/performance for DL inference in the cloud

Object Detection:

	Throughput (images/second)	Cost-per-1M inferences	Throughput: Inf1 vs. G4	Cost-per-inference: Inf1 vs. G4
inf1.2xlarge	118	$0.852	1.82x higher	62% lower
g4dn.xlarge	65	$2.248		

Results based on running YoloV4 model end-to-end with TensorFlow

Natural language:

PyTorch + 🤗	Throughput (sequences/second)	Cost-per-1M inferences	Throughput: Inf1 vs. G4	Cost-per-inference: Inf1 vs. G4
inf1.xlarge	985	$0.064	2.33x higher	81% lower
g4dn.xlarge	422	$0.346		

Results from Neuron SDK v 1.13 - based on Transformers bert-base-uncase model PyTorch, seqlen=128, batch sizes g4dn:64 inf1:6

Image classification:

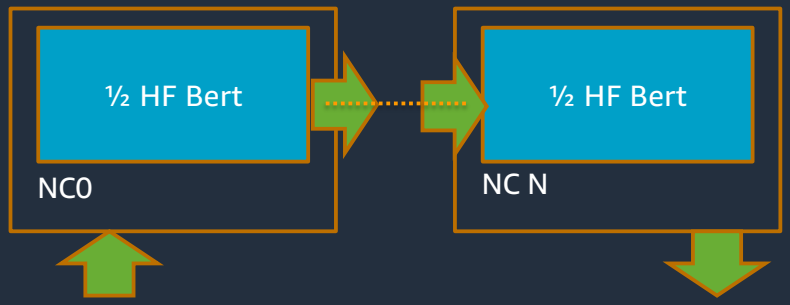
	Throughput (images/second)	Cost-per-1M inferences	Throughput: Inf1 vs. G4	Cost-per-inference: Inf1 vs. G4
inf1.xlarge	2,226	$0.026	24% higher	68% lower
g4dn.xlarge	1,792	$0.082		

Results based on running ResNet-50 model end-to-end with TensorFlow 1.15; G4dn results with TensorRT

aws

Blog: NeuronCore Pipeline for real-time inferencing

Achieve 12x higher throughput and 81% lower cost for PyTorch NLP applications out-of-the-box



BERT-Base, batch=1*	Throughput (sequences/second)	Latency p95 (seconds)	Cost-per-1M inferences	Throughput: Inf1 vs. G4	Cost-per-inference: Inf1 vs. G4
inf1.6xlarge	1793	0.0069	$0.183	12x higher	81% lower
g4dn.xlarge	149	0.0082	$0.981		

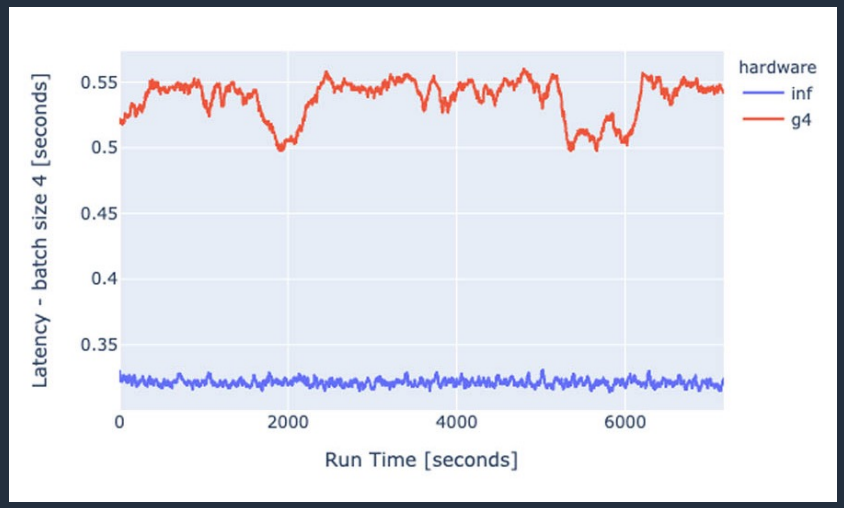
Results from Neuron SDK v 1.13 - based on Transformers bert-base-uncase model PyTorch, seqlen=128, batch sizes g4dn:1 inf1:1

https://aws.amazon.com/blogs/machine-learning/achieve-12x-higher-throughput-and-lowest-latency-for-pytorch-natural-language-processing-applications-out-of-the-box-on-aws-inferentia/

aws

Blog: YOLOv4 model on AWS Inferentia

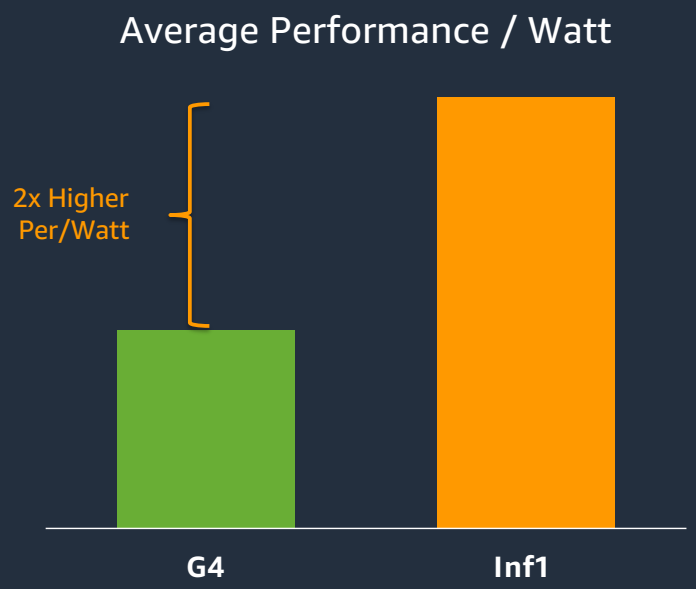
- Inf1 achieved up to **1.82 times higher throughput** and **62% lower cost** per image with significant lower variance when compared to G4dn.
- Inf1 enables customers to reduce latency by over 50% compared to GPUs
- Predictable latency for real-time applications



https://aws.amazon.com/jp/blogs/machine-learning/improving-performance-for-deep-learning-based-object-detection-with-an-aws-neuron-compiled-yolov4-model-on-aws-inferentia/

aws

High Efficiency Enables Sustainable ML Inference at Scale



- Inf1 reduces carbon footprint for ML Inference when compared to higher power GPUs.
- Over 2x higher average perf / watt over G4 instances

aws

Get Started Quickly with Inf1


Amazon Elastic Container Kubernetes


Amazon Elastic Container Service


Amazon SageMaker


AWS Deep Learning AMIs


AWS Deep Learning Containers

Amazon Sagemaker

- Easiest and quickest way to get started with Inf1 instances
- Amazon SageMaker is a fully managed service for building, training and deploying ML models quickly
- Inf1 instances and Neuron are integrated in SageMaker to provide one-click deployment of models onto Inf1

Amazon EKS & ECS

- Inf1 instances are also available for use via Amazon EKS or ECS
- Ideal for deploying models on Inf1 instances via managed containerized pipelines

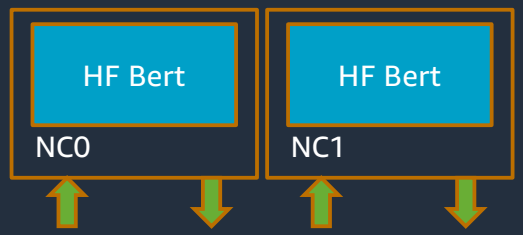
Amazon DLAMI & DL Containers

- Neuron comes pre-installed in AWS Deep Learning AMIs, as well as in AWS Deep Learning Containers

aws

Optimized for High Throughput and Real-Time Low Latency

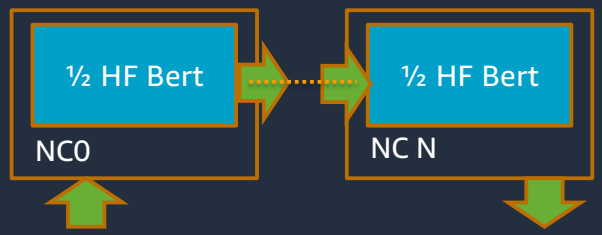
High Throughput



Data Parallel

- Cost optimal, scale out with single chip instances
- Example: Hugging Face BERT using 16 Core achieves 3880 seq/sec with a latency of 25.8ms

Real-time Low Latency



Model Parallel

- Minimum Latency: Scale up to 96 cores on single instance for very large models.
- Example: Hugging Face BERT using 16 Cores achieves 1793 seq/sec with a latency of 6.9ms

aws

Inferentia News and More Information

Customer Success Stories

November 12, 2020 AWS: Majority of Alexa Now Running on Faster, More Cost-Effective Amazon EC2 Inf1 Instances
April 7, 2021 Medium: How We Used AWS Inferentia to Boost PyTorch NLP Model Performance by 4.9x for the Autodesk Ava Chatbot

Technical References

October 6, 2020 AWS: Achieving 1.85x higher performance for deep learning based object detection with an AWS Neuron compiled YOLOv4 model on AWS Inferentia
May 4, 2021 AWS: Achieve 12x higher throughput and lowest latency for PyTorch Natural Language Processing applications out-of-the-box on AWS Inferentia (China)

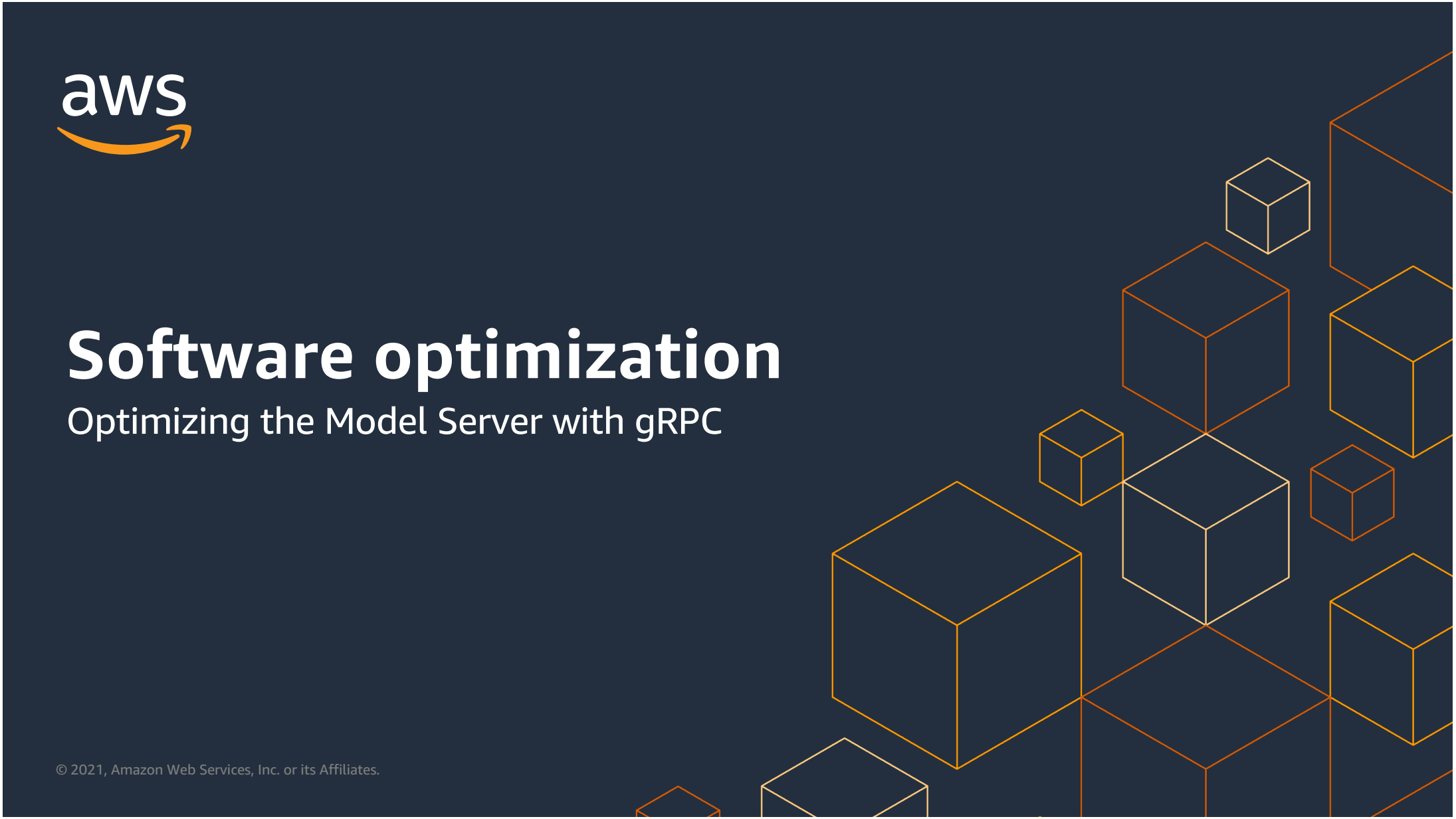
Updates

February 17, 2021 AWS: Amazon ml.Inf1 instances are now available on Amazon SageMaker in 14 additional AWS regions
March 23, 2021 AWS: Amazon EC2 Inf1 instances based on AWS Inferentia now available in 4 additional regions
June 23, 2021 AWS: Amazon EC2 Inf1 instances - New features, improved performance and lower prices

Learn More about Inferentia

February 5, 2021 YouTube: AWS re:Invent 2020: The journey to silicon innovation at AWS
March 10, 2021 Broadcast: GCR TrainingTALK | Accelerating ML Inference and Reducing Costs for Customers Including Alexa and Snap with AWS Inferentia
March 11, 2021 Broadcast: Accelerating ML Inference and Reducing Costs for Customers Including Alexa and Snap with AWS Inferentia
May 27, 2021 Broadcast: PoA Talk: Rethinking Machine-Learning acceleration with Inferentia & Trainium
May 27, 2021 YouTube: Learn How Machine Learning Developers are Saving up to 70% on Inference with AWS Inferentia
July 27, 2021 YouTube : Save up to 70% on Your Machine Learning Inference Costs Using Amazon EC2 Inf1 - AWS Online Tech Talk

aws

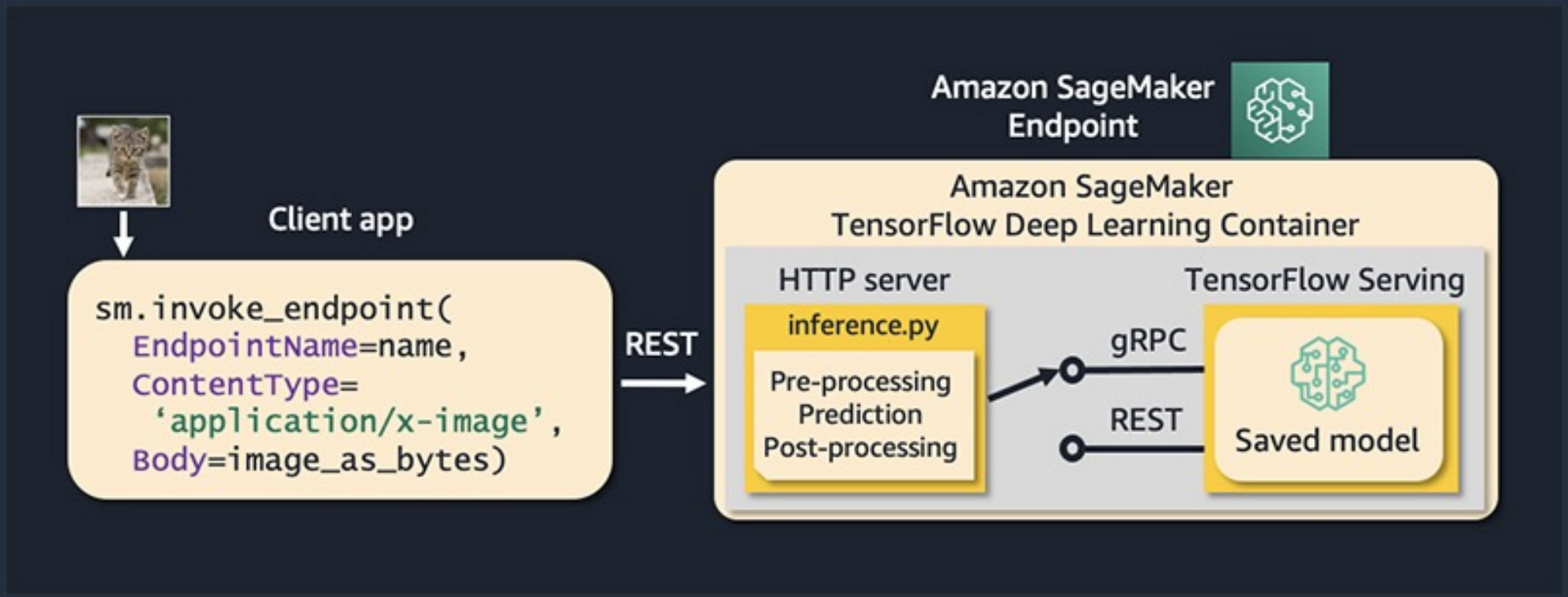


Software optimization

Optimizing the Model Server with gRPC

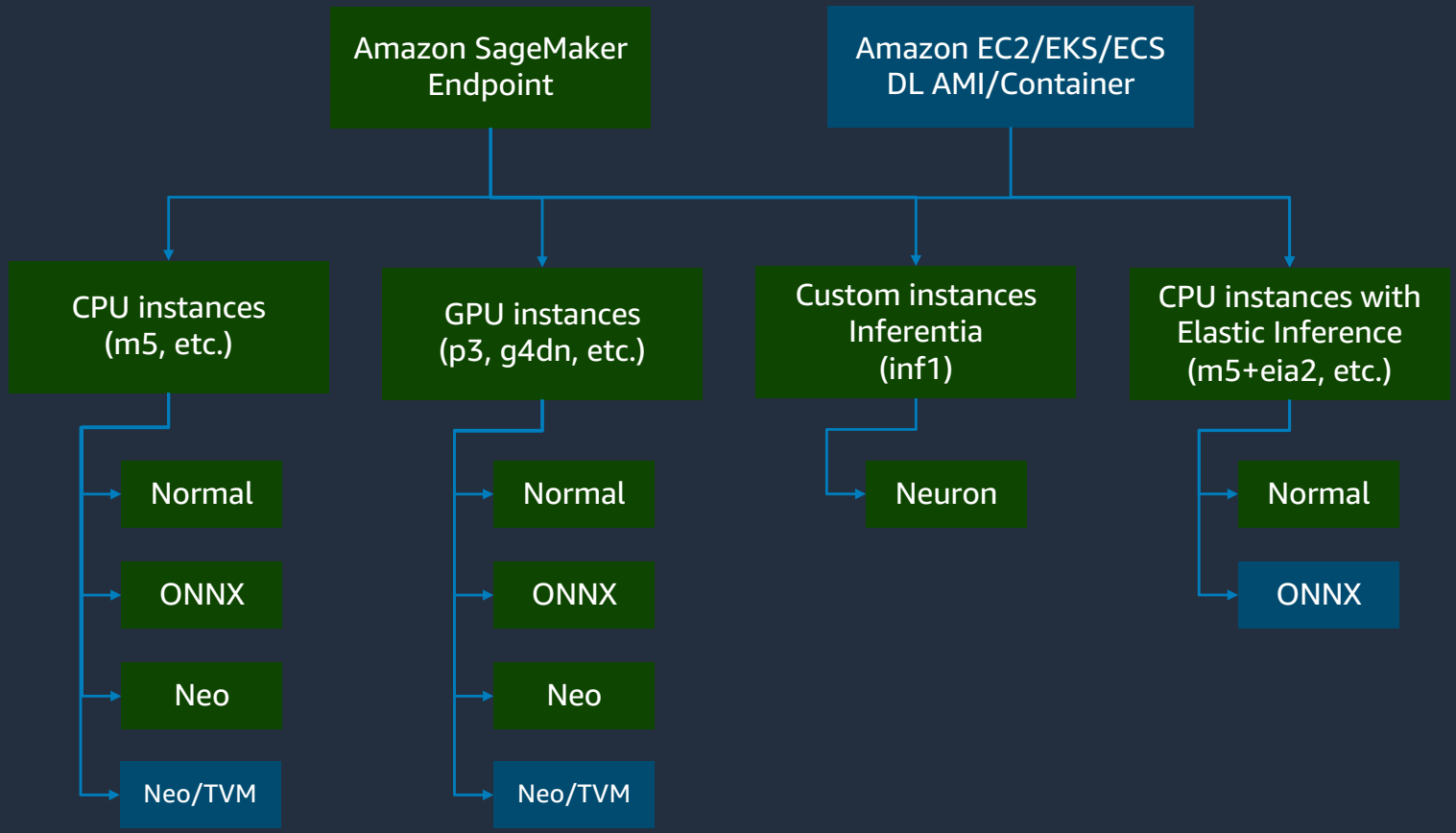
Reduce computer vision inference latency using gRPC with TensorFlow serving on Amazon SageMaker

Learn more at: https://amzn.to/3wommhM



aws

NLP benchmark tests – Q3 2021



aws

NLP benchmark tests – Q3 2021

With Majauts' BERT traced model:

Neo PyTorch

Test #	Optimized?	Instance	vCPU	RAM	GPU	GPU Mem	$/hour	Char/sec	MM-Chars/$	Notes
1	-	ml.m5.xlarge	4	16	-	-	0.257	1457.9	20.42	Prediction time: 1.4 secs, 2034 input-characters, 1457.9 char/secs
2	Neo	ml.m5.xlarge	4	16	-	-	0.257	¿?	¿?	*Inference request timed out in TVM*
3	-	ml.p3.2xlarge	8	61	1	16	4.131	2407.2	2.1	Prediction time: 0.8 secs, 2034 input-characters, 2407.2 char/secs
4	Neo	ml.p3.2xlarge	8	61	1	16	4.131	2700.4	2.35	Neo prediction time: 0.8 secs, 2034 input-characters, 2700.4 char/secs
5	-	ml.g4dn.xlarge	4	16	1	16	0.821	3007.9	13.19	Prediction time: 0.7 secs, 2034 input-characters, 3007.9 char/secs
6	Neo	ml.g4dn.xlarge	4	16	1	16	0.821	2703.6	11.86	Neo prediction time: 0.8 secs, 2034 input-characters, 2703.6 char/secs
7	Neo	ml.inf1.xlarge	4	8	1	-	0.33	¿?	¿?	*Unable to compile model for ml_inf1 with Neo*
8	Neo	ml.inf1.2xlarge	8	16	1	-	0.484	¿?	¿?	*Unable to compile model for ml_inf1 with Neo*
9	-	ml.m5.xlarge + eia2.medium	4	16	<1	2	0.377	1241.4	11.85	Prediction time: 1.6 secs, 2034 input-characters, 1241.4 char/secs

Neo PyTorch

https://docs.aws.amazon.com/sagemaker/latest/dg/neo.html

Neo TVM/DLR

- Compilation (TVM): https://github.com/neo-ai/tvm
- Serving (DLR): https://github.com/neo-ai/neo-ai-dlr

aws

NLP benchmark tests – Q3 2021

With standard bert-base-multilingual-uncased model:

Neo

Instance Type	Mode	Model Latency (ms)	Overhead Latency	Neo %-gain
m5	Normal	598.633	3.259	
m5	Neo compiled	31802.506	9.803	-52.12521
p3	Normal	17.46	1.963	
p3	Neo compiled	7.203	1.969	0.58746
g4dn	Normal	13.071	2.759	
g4dn	Neo compiled	15.834	1.867	-0.21138

ONNX

Notebook (ml.p3.2xlarge):

Non-compiled (model traced no grad) CPU (ml.m5.xlarge) - Avg inference time: 0.51 sec

Compiled ONNX runtime CPU - Avg inference time: 0.43 sec

ONNX gain vs non-compiled: 15.69%

SM Endpoint:

Non-compiled (model traced no grad) GPU (ml.g4dn.xlarge) - Avg inference time: 0.04 sec (0.02 + 0.012 overhead)

Compiled ONNX runtime CPU (ml.m5.xlarge) - Avg inference time: 2.01 sec

Compiled ONNX runtime GPU (ml.g4dn.xlarge) - Avg inference time: 0.05 sec (0.03 + 0.012 overhead)

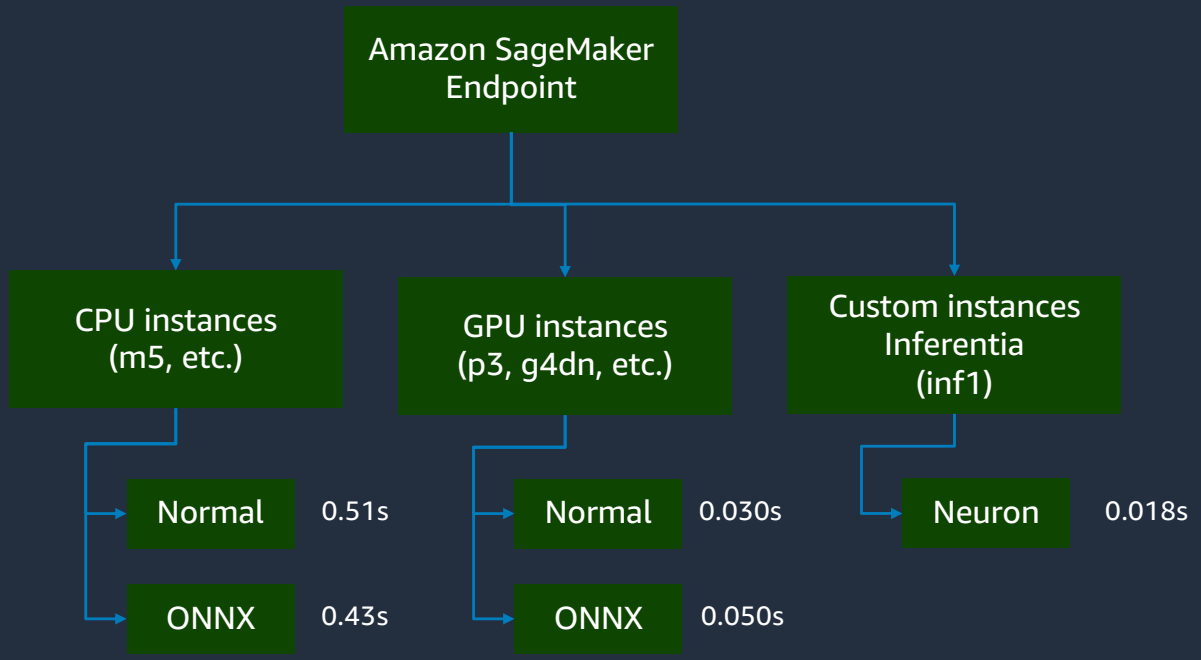
ONNX gain vs non-compiled (GPU): -20%

Inferentia

Experiment	TPS	p50 latency (ms)	cost/hour	TP / $ (divided by 100)	$ / 1M inferences	inferences / $	transaction / hour
m5.xlarge	3.6	545.8	0.23	0.156	17.75	56348	12960
g4dn.xlarge	53.9	34.3	0.736	0.732	3.79	263641	194040
p3.2xlarge	36.4	53.4	3.825	0.095	29.19	34259	131040
inf1.xlarge	100.3	18.5	0.297	3.38	0.82	1215758	361080
inf1.2xlarge	108.3	18	0.434	2.5	1.11	898341	389880
inf1.6xlarge	117.2	16.8	1.416	0.83	3.35	297966	421920

aws

NLP benchmark tests – Q3 2021



https://github.com/rodzanto/nlp-benchmark

aws

NLP benchmark tests – Q3 2021

Proposed next steps…

- Majauts to test with code examples in repo provided

- Explore Hugging Face @ AWS approach:
  - SageMaker & HF - https://docs.aws.amazon.com/sagemaker/latest/dg/hugging-face.html
  - Optimum - https://huggingface.co/blog/hardware-partners-program
  - Discuss with HF team directly

aws

Tips

Further latency reduction: tweaking the model

- **Model size and architecture influence latency**
- **Latency-optimized ML architectures exist in several ML paradigms**
- **Post-training, models can be compressed to reduce their computational budget**

Tweaking the model can reduce latency by an order of magnitude, but may come with concessions on accuracy and extra operational work

aws

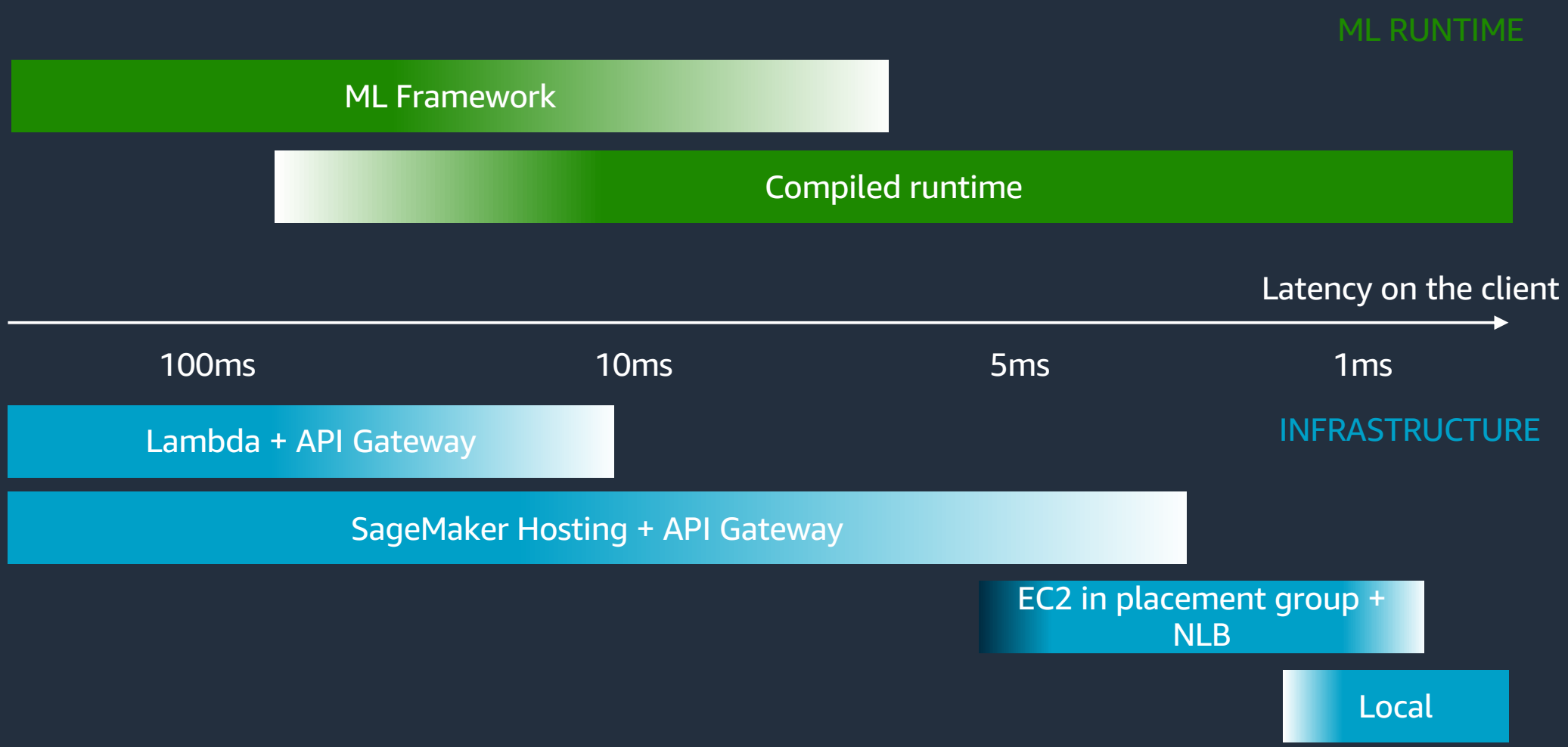
Further latency reduction: system architecture

- Can pre- and post-processing be collocated on same instance?
- Do you expect many model inference over same features?
- Is feature fetching a bottleneck?

Tweaking the model can reduce latency by an order of magnitude, but may come with concessions on accuracy and extra operational work

aws

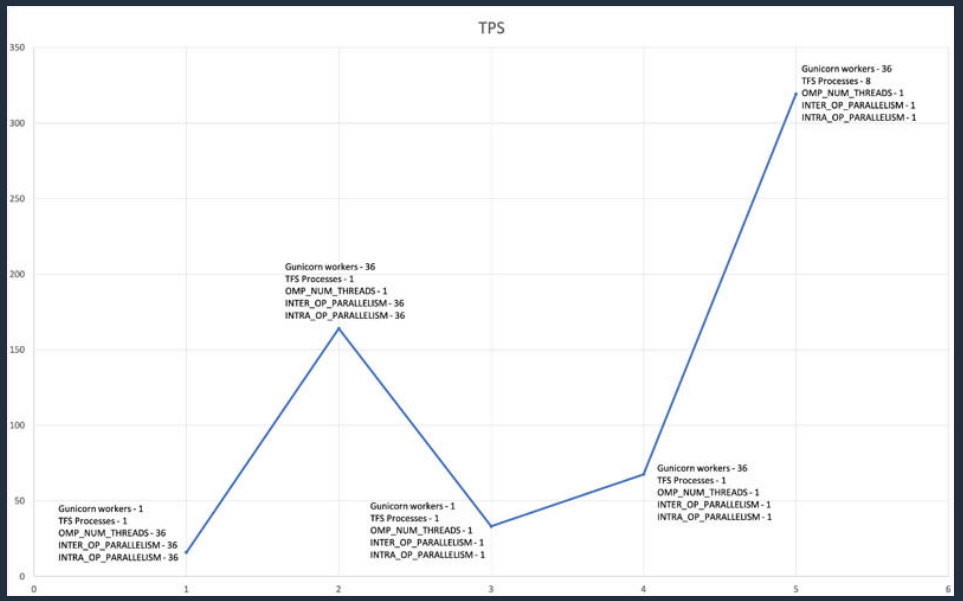
Summary: High-level Architecture given latency budget



aws

Further Reading

Maximize TensorFlow performance on Amazon SageMaker endpoints for real-time inference



aws

Monitoring & Auditability

aws

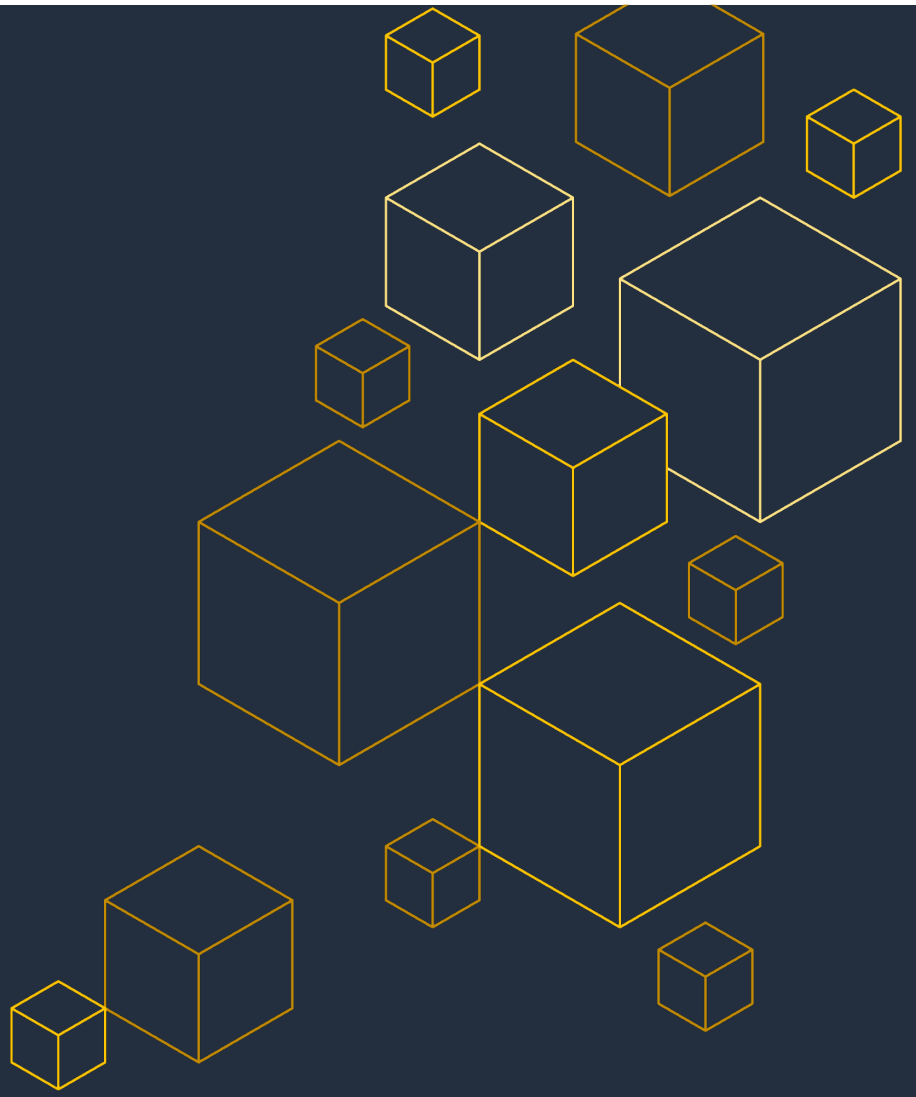
Monitoring & Auditability

Amazon SageMaker Considerations

- ✓ Logging & Monitoring
- ✓ Source & Artifact Management
- ✓ Security Vulnerability Scans
- ✓ Change Detection

aws

Monitoring & Auditability
Logging & Monitoring



aws

Logging & Monitoring

Amazon SageMaker Logging

Amazon SageMaker API Logging & Monitoring

- Traceability on SageMaker API requests

Amazon SageMaker Service Logging & Monitoring

- System Metrics
- Logs generated by provisioned resources (ex. Training Jobs, Processing Jobs, etc)

aws

Logging & Monitoring

Amazon SageMaker Logging

Amazon SageMaker API Logging & Monitoring

- Traceability on SageMaker API requests

Amazon SageMaker Service Logging & Monitoring

- System Metrics
- Logs generated by provisioned resources (ex. Training Jobs, Processing Jobs, etc)

aws

Logging & Monitoring

Amazon SageMaker API Logging

Log SageMaker API Calls

AWS CloudTrail

- ✓ Traceability
  - ❑ Log all SageMaker API Calls
  - ❑ Persist API trail logs in S3 / Cloudwatch Logs
- ✓ Integrity & Auditability
  - ❑ Centralize CloudTrail logs to Security Account
  - ❑ Log file integrity validation
- ✓ Support for Multiple Event Types
  - ❑ AwsApiCall
  - ❑ AwsServiceEvent

SageMaker API Calls: https://docs.aws.amazon.com/sagemaker/latest/APIReference/Welcome.html

aws

Logging & Monitoring

Amazon SageMaker API Logging

Log SageMaker API Calls

AWS CloudTrail

Example #1: IAM User – Bob , Opened a Jupyter Notebook Instance

Console

Event time	User name	Event name	Resource type	Resource name
2020-05-27, 12:26:02 PM	Bob	CreatePresignedNotebookInstanceUrl		

AWS access key	ASIAXCRRJEA2FZ4ICORB	**Event time**	2020-05-27, 12:26:02 PM
AWS region	us-east-1	**Read only**	false
Error code		**Request ID**	50b04c27-606e-411f-b6ab-6ef55d893407
Event ID	3eca4618-2548-4efb-a87f-09ab4b02b6a1	**Source IP address**	72.21.196.66
Event name	CreatePresignedNotebookInstanceUrl	**User name**	Bob
Event source	sagemaker.amazonaws.com		

Event Details (Limited view showing key fields)

View Event


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAXCRRJEA2GFNCLN6L5",
        "arn": "arn:aws:iam::[redacted]:user/Bob",
        "accountId": "[redacted]",
        "accessKeyId": "ASIAXCRRJEA2FZ4ICORB",
        "userName": "Bob",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-05-27T18:25:02Z"
            }
        }
    },
    "eventTime": "2020-05-27T18:26:02Z",
    "eventSource": "sagemaker.amazonaws.com",
    "eventName": "CreatePresignedNotebookInstanceUrl",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.196.66",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.776 Linux/4.9.184-0.1.ac.235.83.329.me
    "requestParameters": {
        "notebookInstanceName": "ProjectA",
        "sessionExpirationDurationInSeconds": 43200
    },
    "responseElements": null,
    "requestID": "50b04c27-606e-411f-b6ab-6ef55d893407",
    "eventID": "3eca4618-2548-4efb-a87f-09ab4b02b6a1",
    "eventType": "AwsApiCall",
    "recipientAccountId": "486508404788"
}
```


aws

Logging & Monitoring

Amazon SageMaker API Logging

Log SageMaker API Calls

AWS CloudTrail

Example #2: IAM User – TomAdmin , Created a Jupyter Notebook Instance

Console

Event time	User name	Event name	Resource type	Resource name
2020-05-27, 12:33:02 PM	TomAdmin	CreateNotebookInstance		

AWS access key	ASIAXCRRJEA2LD77B5F5	**Event time**	2020-05-27, 12:33:02 PM
AWS region	us-east-1	**Read only**	false
Error code		**Request ID**	f6d38862-89fb-4e96-bc92-b940f6c3e773
Event ID	b36ddf99-e65f-4d2b-9184-72b261fe6d32	**Source IP address**	72.21.196.66
Event name	CreateNotebookInstance	**User name**	TomAdmin
Event source	sagemaker.amazonaws.com		

Event Details (Limited view showing key fields)

View Event


```
{
    "eventVersion": "1.05",
    "userIdentity": {
        "type": "IAMUser",
        "principalId": "AIDAXCRRJEA2C4V6ZKAGH",
        "arn": "arn:aws:iam::[redacted]:user/TomAdmin",
        "accountId": "[redacted]",
        "accessKeyId": "ASIAXCRRJEA2LD77B5F5",
        "userName": "TomAdmin",
        "sessionContext": {
            "sessionIssuer": {},
            "webIdFederationData": {},
            "attributes": {
                "mfaAuthenticated": "false",
                "creationDate": "2020-05-27T18:32:26Z"
            }
        }
    },
    "eventTime": "2020-05-27T18:33:02Z",
    "eventSource": "sagemaker.amazonaws.com",
    "awsRegion": "us-east-1",
    "sourceIPAddress": "72.21.196.66",
    "userAgent": "aws-internal/3 aws-sdk-java/1.11.776 Linux/4.9.184-0.1.ac.235.83.329.me
    "requestParameters": {
        "notebookInstanceName": "ProjectA-BobNoteBookInstance",
        "instanceType": "ml.t2.medium",
        "roleArn": "arn:aws:iam::486508404788:role/Predictive-Maintenance-NotebookInstanc
        "directInternetAccess": "Enabled",
        "volumeSizeInGB": 5,
        "rootAccess": "Enabled"
    },
    "responseElements": {
        "notebookInstanceArn": "arn:aws:sagemaker:us-east-1:486508404788:notebook-instanc
    },
    "requestID": "f6d38862-89fb-4e96-bc92-b940f6c3e773",
    "eventID": "b36ddf99-e65f-4d2b-9184-72b261fe6d32",
    "eventType": "AwsApiCall",
    "recipientAccountId": "486508404788"
}
```


aws

Logging & Monitoring

API Log Monitoring & Analysis

Amazon SageMaker CloudTrail Logs
+ Amazon CloudWatch Logs

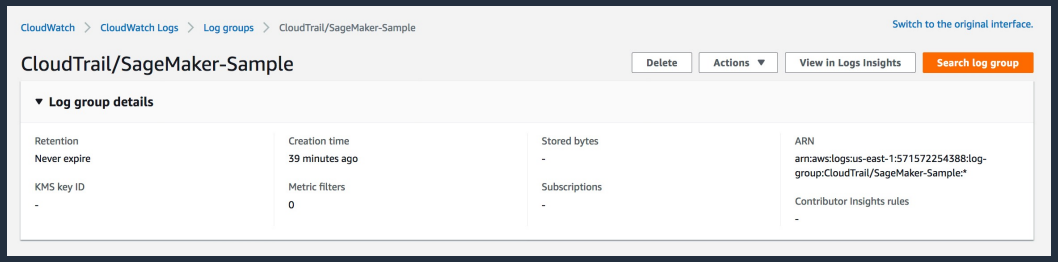
- ✓ Visibility
  - ❑ Gaining insights from logs
  - ❑ Setup responsive events
- ✓ Responsive Actions
  - ❑ Setup alerts indicating unusual behavior
- ✓ Integrating Services
  - ❑ Amazon Athena
  - ❑ Amazon CloudWatch Logs

aws

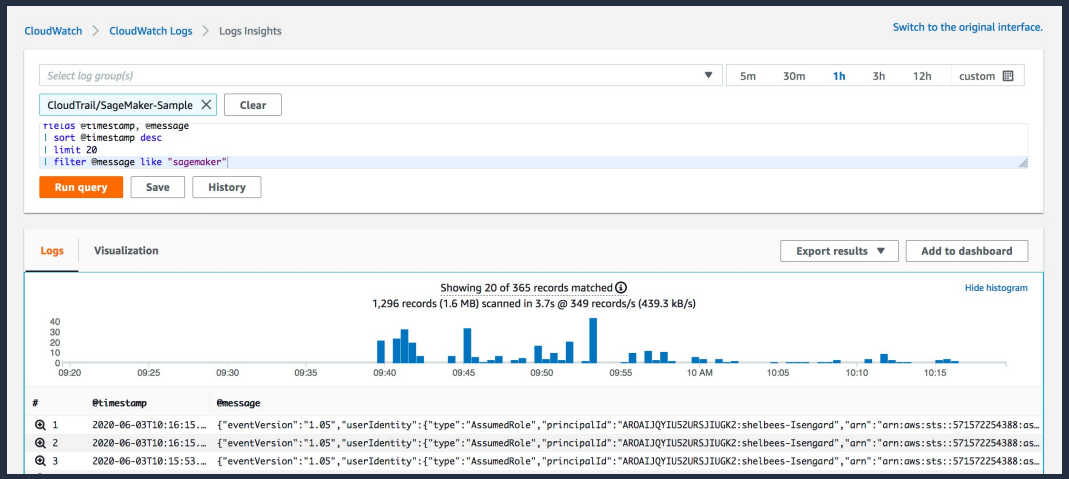
Logging & Monitoring

API Log Monitoring & Analysis

1) Create CloudTrail



2) Using Logs Insights, filter SageMaker



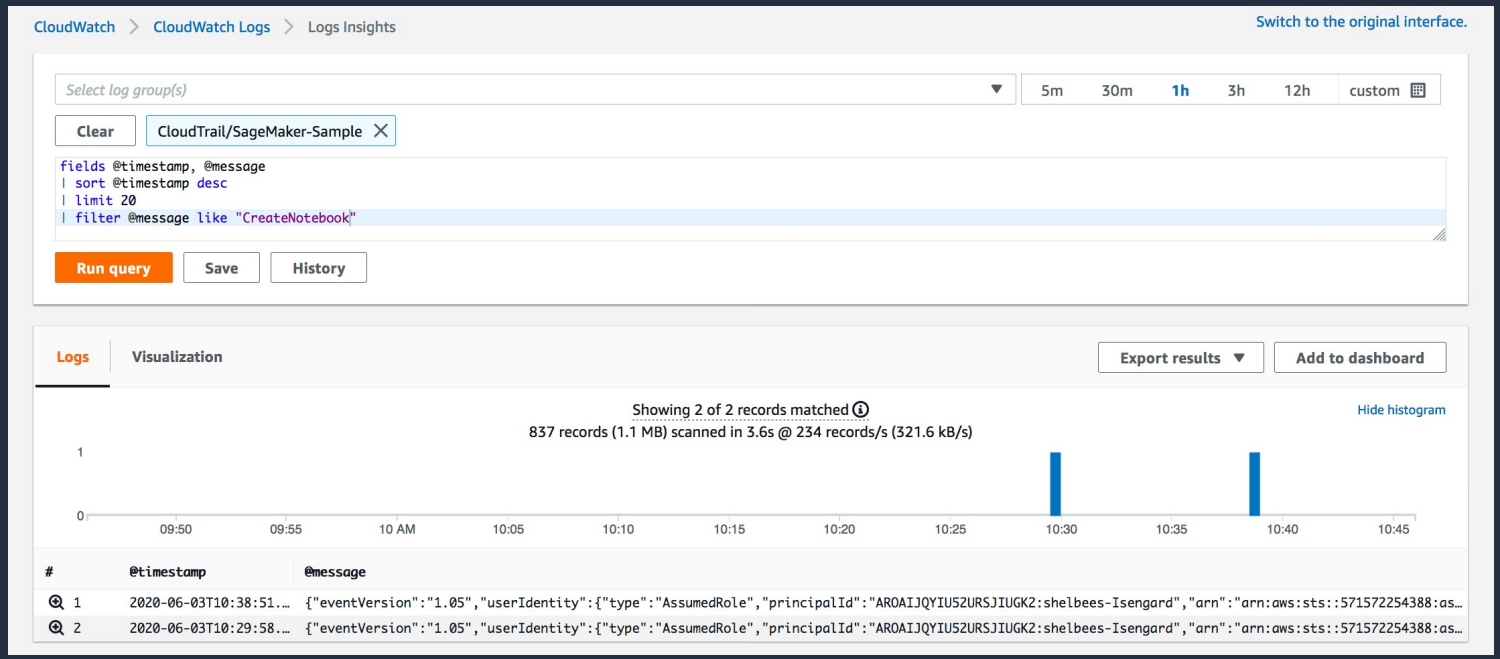
aws

Logging & Monitoring

API Log Monitoring & Analysis

3) Refine for specific activities

Example: View Notebook Creation Activity



Logging & Monitoring

Amazon SageMaker Logging

Amazon SageMaker API Logging & Monitoring

- Traceability on SageMaker API requests

Amazon SageMaker Service Logging & Monitoring

- System Metrics
- Logs generated by provisioned resources (ex. Training Jobs, Processing Jobs, etc)

aws

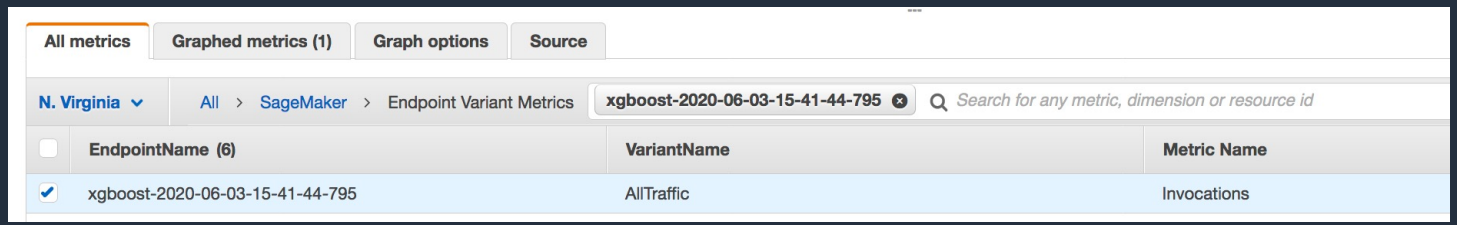
Logging & Monitoring
Amazon SageMaker Service

SageMaker Service Logging & Monitoring

Amazon CloudWatch

- ✓ Preventative Controls
  - ❑ AutoScaling: Consider Maximum capacity
- ✓ Baseline
  - ❑ Understand normal workload behavior

Example: SageMaker Endpoint - Innvocations



aws

Logging & Monitoring

Amazon SageMaker Service

Monitor Behavior

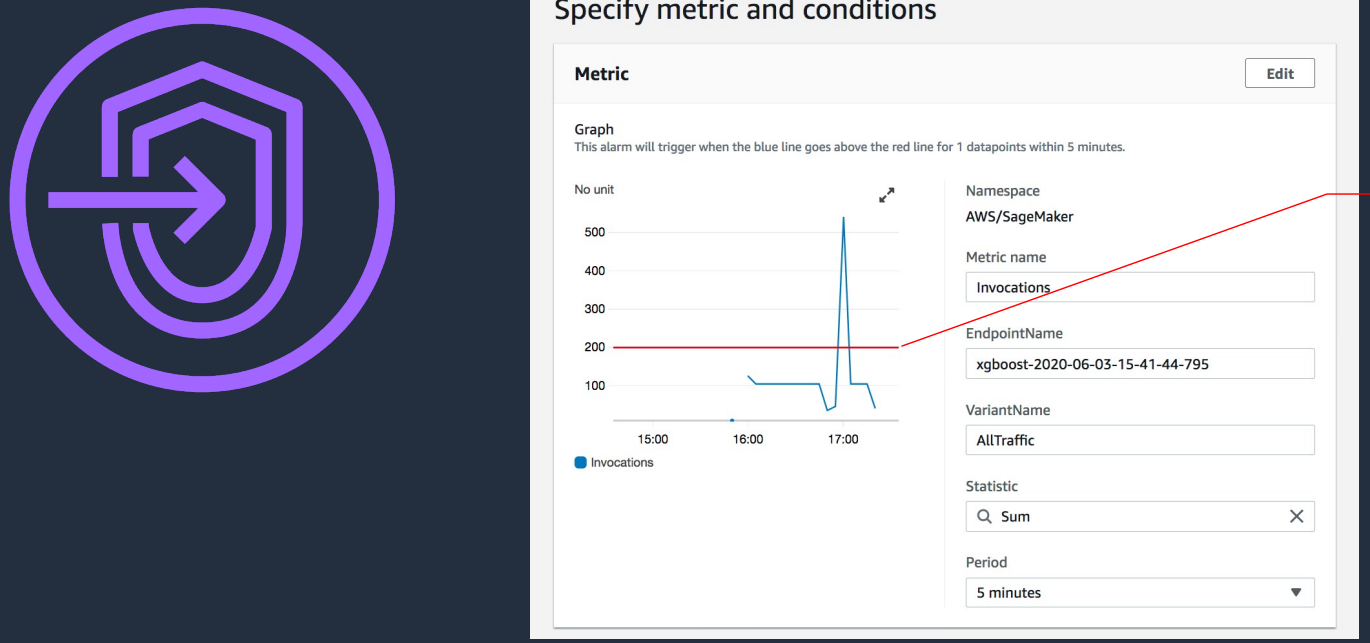
Amazon CloudWatch

aws

Logging & Monitoring
Amazon SageMaker Service

Configure Alarms & Setup Notifications

Amazon CloudWatch



Alarm Threshold (Ex: 200)

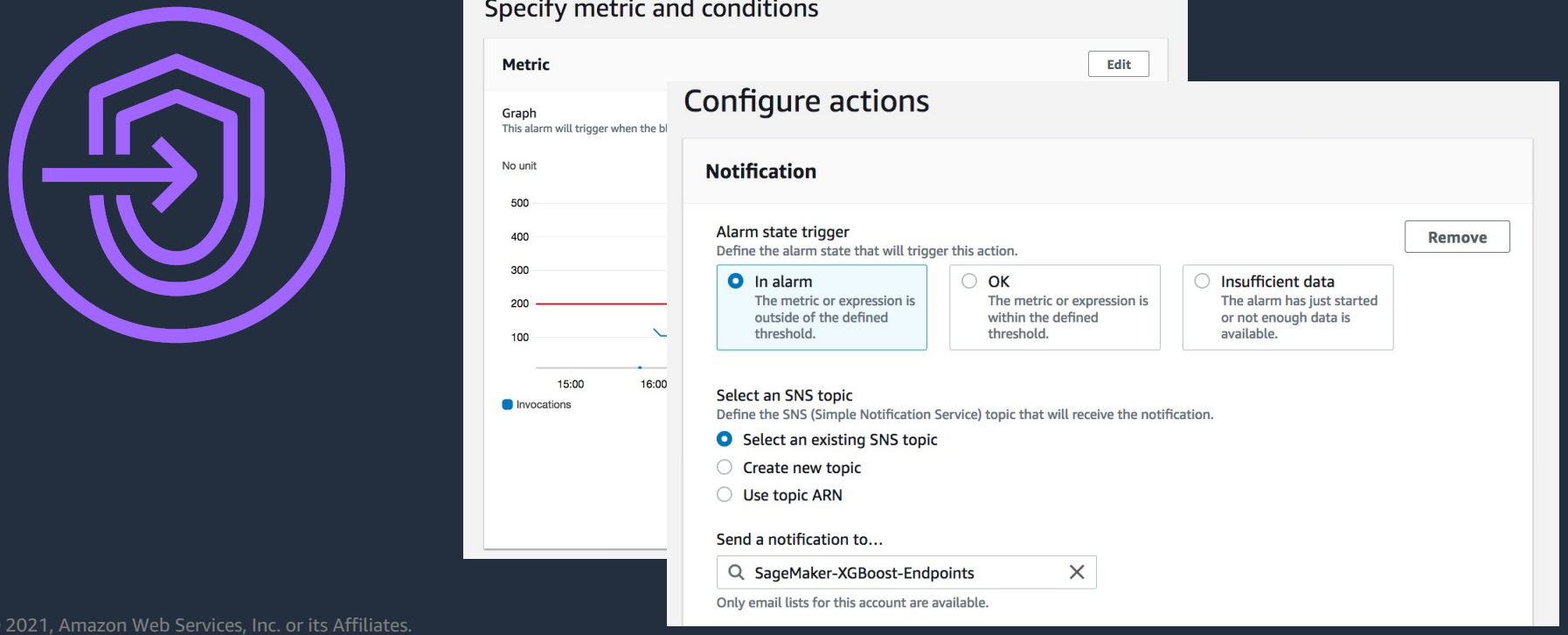
aws

Logging & Monitoring

Amazon SageMaker Service

Configure Alarms & Setup Notifications

Amazon CloudWatch

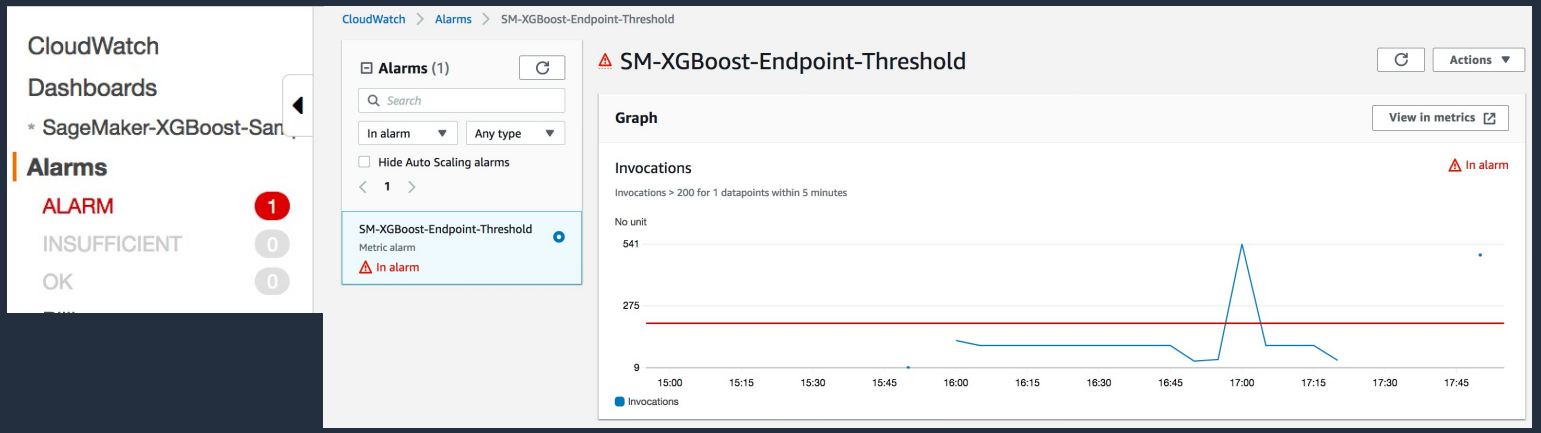


aws

Logging & Monitoring
Amazon SageMaker Service

Configure Alarms & Setup Notifications

Amazon CloudWatch



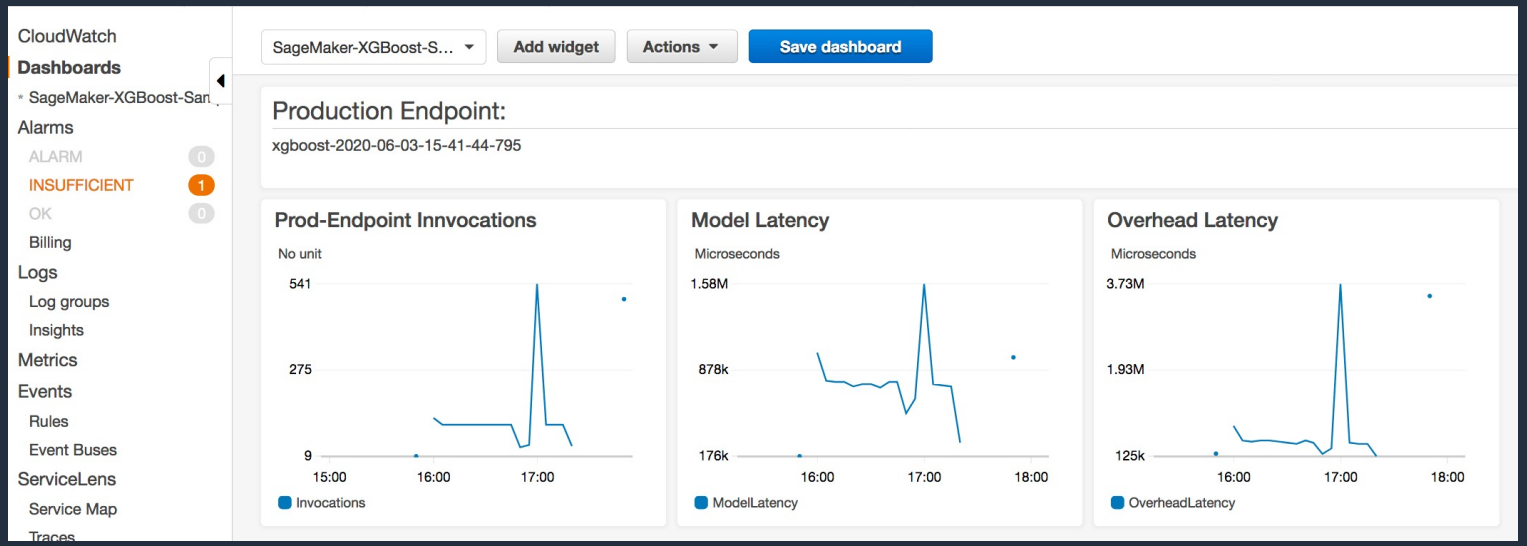
aws

Logging & Monitoring

Amazon SageMaker Service

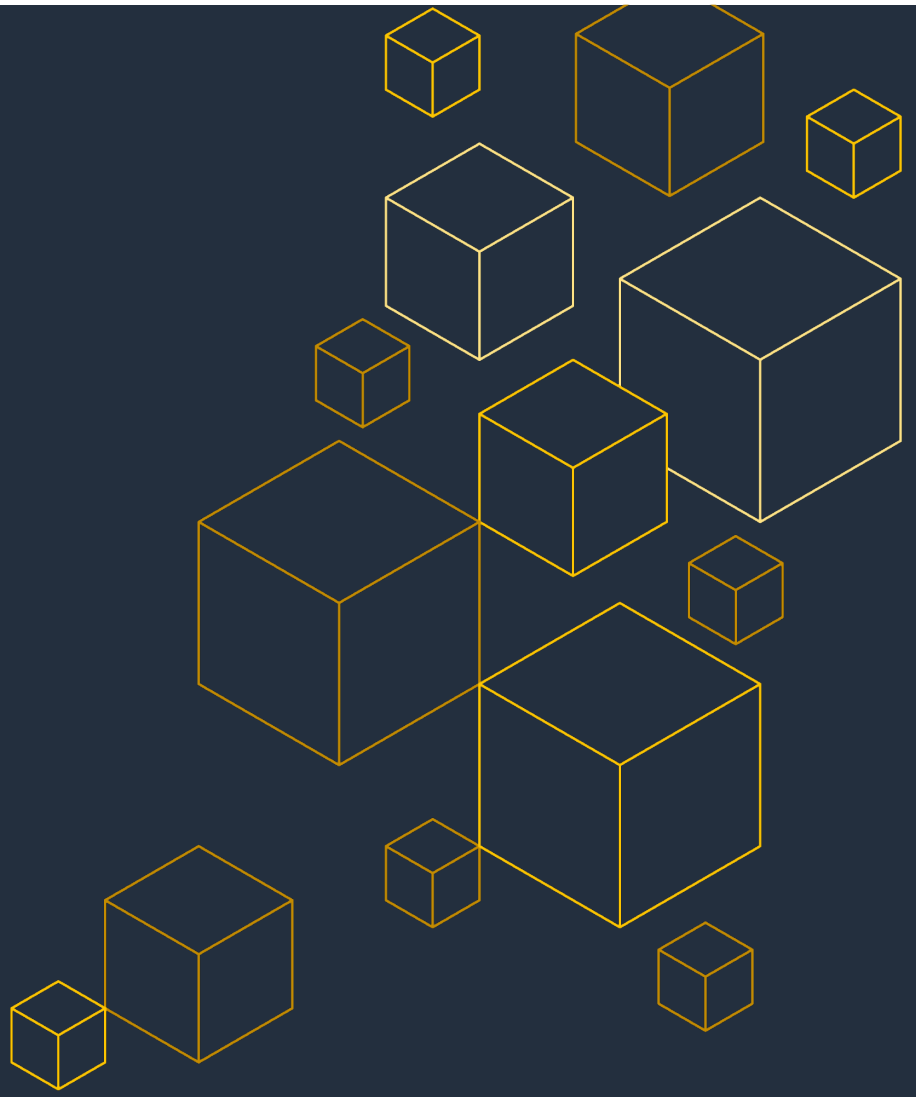
Monitor Behavior : Create Dashboards

Amazon CloudWatch



aws

Monitoring & Auditability
Artifact Management



aws

Monitoring & Auditability

Artifact Management - Traceability

Source Management

Version Control System
(e.g. AWS CodeCommit / GitHub)

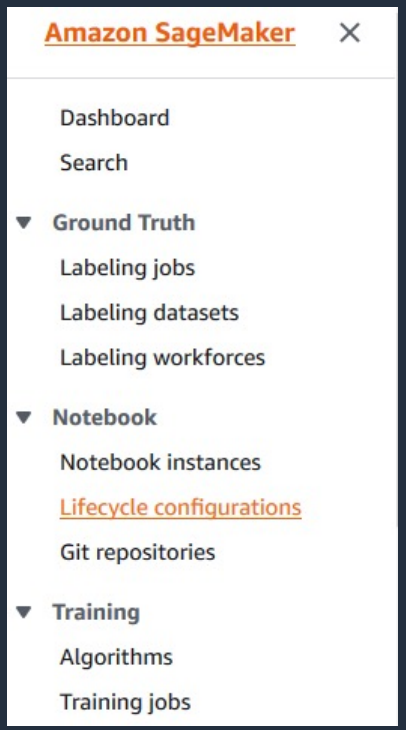
Environment / Solution	Processing	Algorithm Code	Inference/Helper Code
• Configuration Code • Supporting Lambda Code • ETL Scripts • Lifecycle Config Scripts	• Scripts • DockerFile	• Jupyter Notebook • Scripts • DockerFile	• Scripts • DockerFile

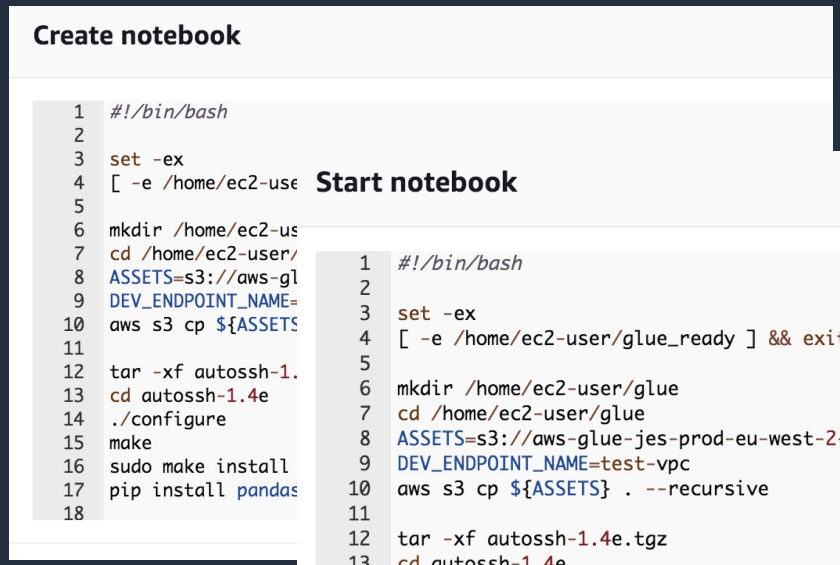
aws

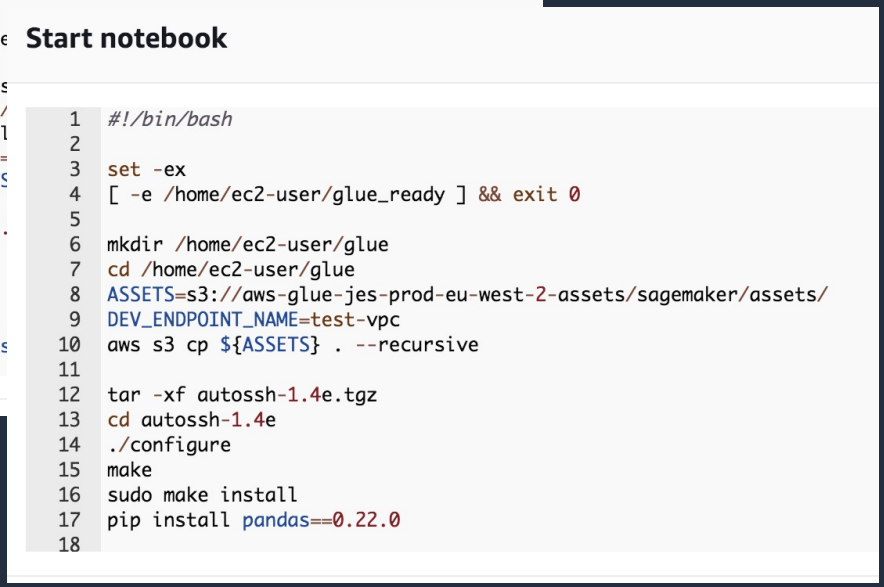
Monitoring & Auditability

Artifact Management - Traceability

Amazon SageMaker Notebook Instances: Lifecycle Configurations







[SageMaker Lifecycle Configuration Repo]

aws

Monitoring & Auditability

Artifact Management - Traceability

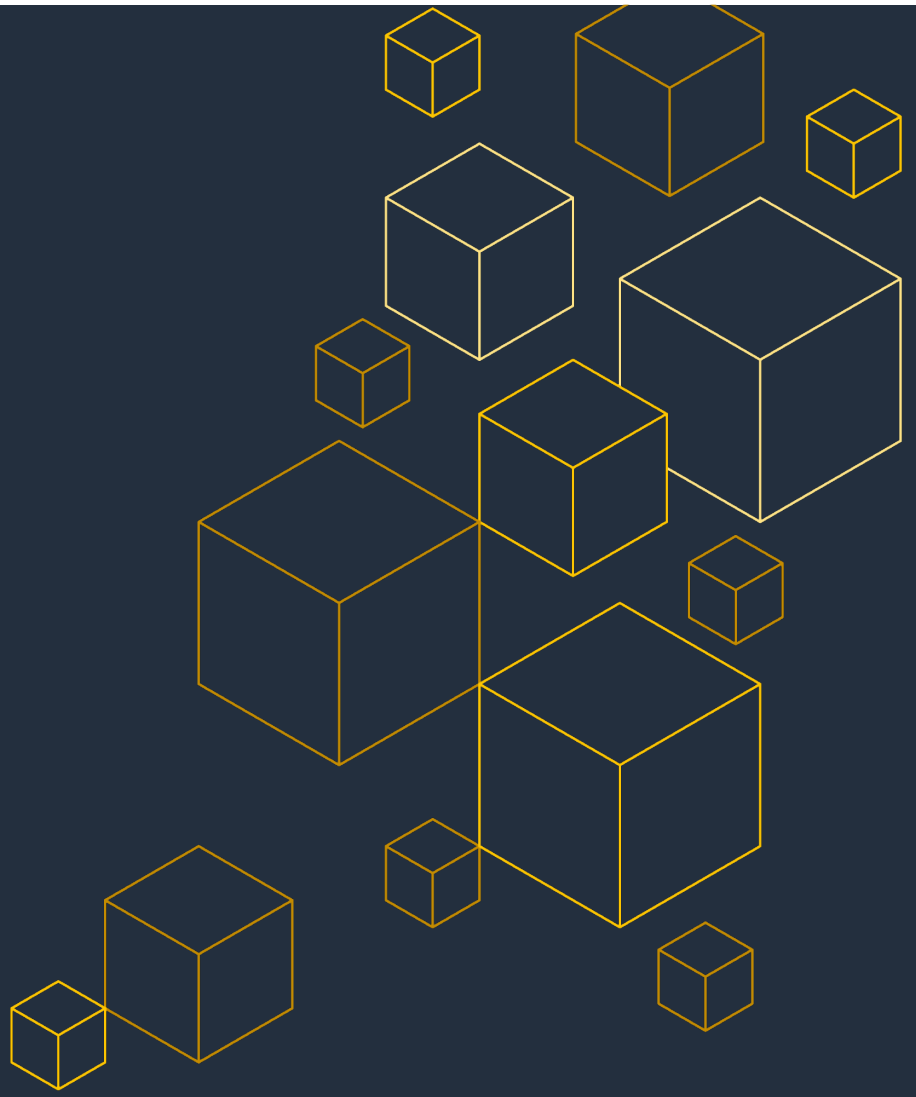
Library & Package Management

Repository Management
(e.g. Artifactory, S3)

Docker Images	**Model Artifacts**	**Libraries**
• Training image • Inference image • Processing image	• Model.tar.gz	• Python libraries • R libraries • RPM packages

aws

Monitoring & Auditability
Change Detection



aws

Monitoring & Auditability
Change Detection

CloudWatch Events

Leverage newly added CloudWatch Event Types to drive automated workflows and responses

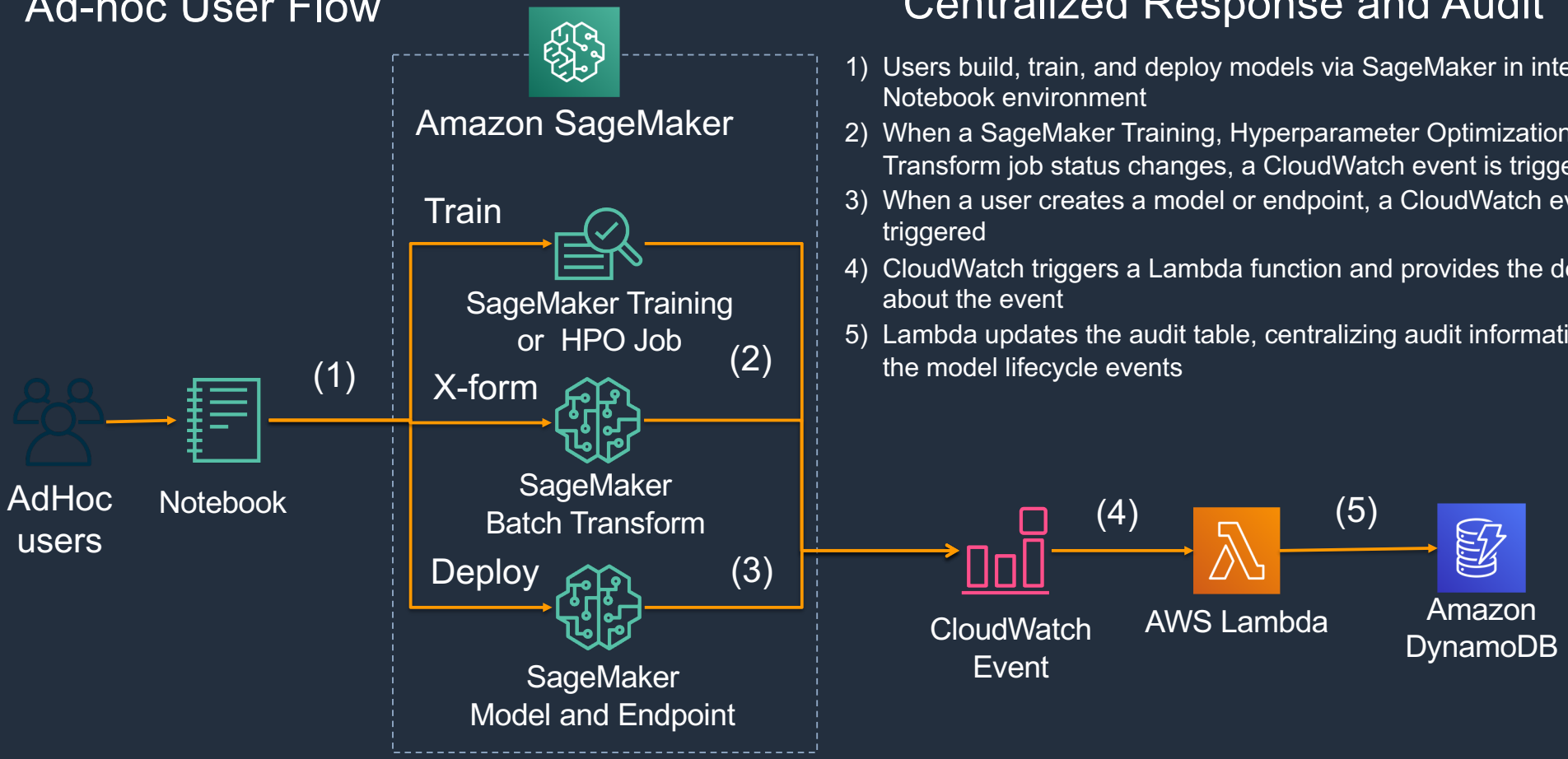
- Training Job State Change
- Tuning Job State Change
- Transform Job State Change
- Endpoint Config State Change
- Endpoint State Change
- Notebook Instance State Change
- Notebook Lifecycle State Change
- Algorithm State Change
- Model State Change
- Model Package State Change

aws

Monitoring & Auditability

Artifact Management - Auditability

Ad-hoc User Flow



Centralized Response and Audit

1) Users build, train, and deploy models via SageMaker in interactive Notebook environment
2) When a SageMaker Training, Hyperparameter Optimization, or Batch Transform job status changes, a CloudWatch event is triggered
3) When a user creates a model or endpoint, a CloudWatch event is triggered
4) CloudWatch triggers a Lambda function and provides the details about the event
5) Lambda updates the audit table, centralizing audit information from the model lifecycle events

aws

Summary

- Why talk about inferences
- Customer challenges
- Optimization
  - Hardware. Software. Network
- Security, Monitoring, and Auditability

aws