
aws
SUMMIT
SAN FRANCISCO | APRIL 20 2022

AIM207

Train ML models at scale with Amazon SageMaker Training

Gal Oshri
Senior Product Manager
Amazon

Sam Palani
Principal Machine Learning Specialist
Amazon

Agenda

AI / ML Landscape on AWS

Training ML models on SageMaker

Large Scale Training

Hyperparameter Tuning

Key Takeaways

The AWS ML stack

AI SERVICES

SPECIALIZED

Business processes
Amazon Personalize
Amazon Forecast
Amazon Fraud Detector
Amazon Lookout for Metrics

Search
Amazon Kendra

Code + DevOps
Amazon CodeGuru
Amazon DevOps Guru

Industrial
Amazon Monitron
Amazon Lookout for Equipment
Amazon Lookout for Vision

Healthcare
Amazon HealthLake
Amazon Comprehend Medical
Amazon Transcribe Medical

CORE

Text/Documents
Amazon Textract
Amazon Comprehend
Amazon Translate

Chatbots
Amazon Lex

Speech
Amazon Polly
Amazon Transcribe
Amazon Transcribe Call Analytics

Vision
Amazon Rekognition
AWS Panorama

ML SERVICES

Label data

SAGEMAKER CANVAS
ML for business analysts

SAGEMAKER STUDIO LAB
Learn ML

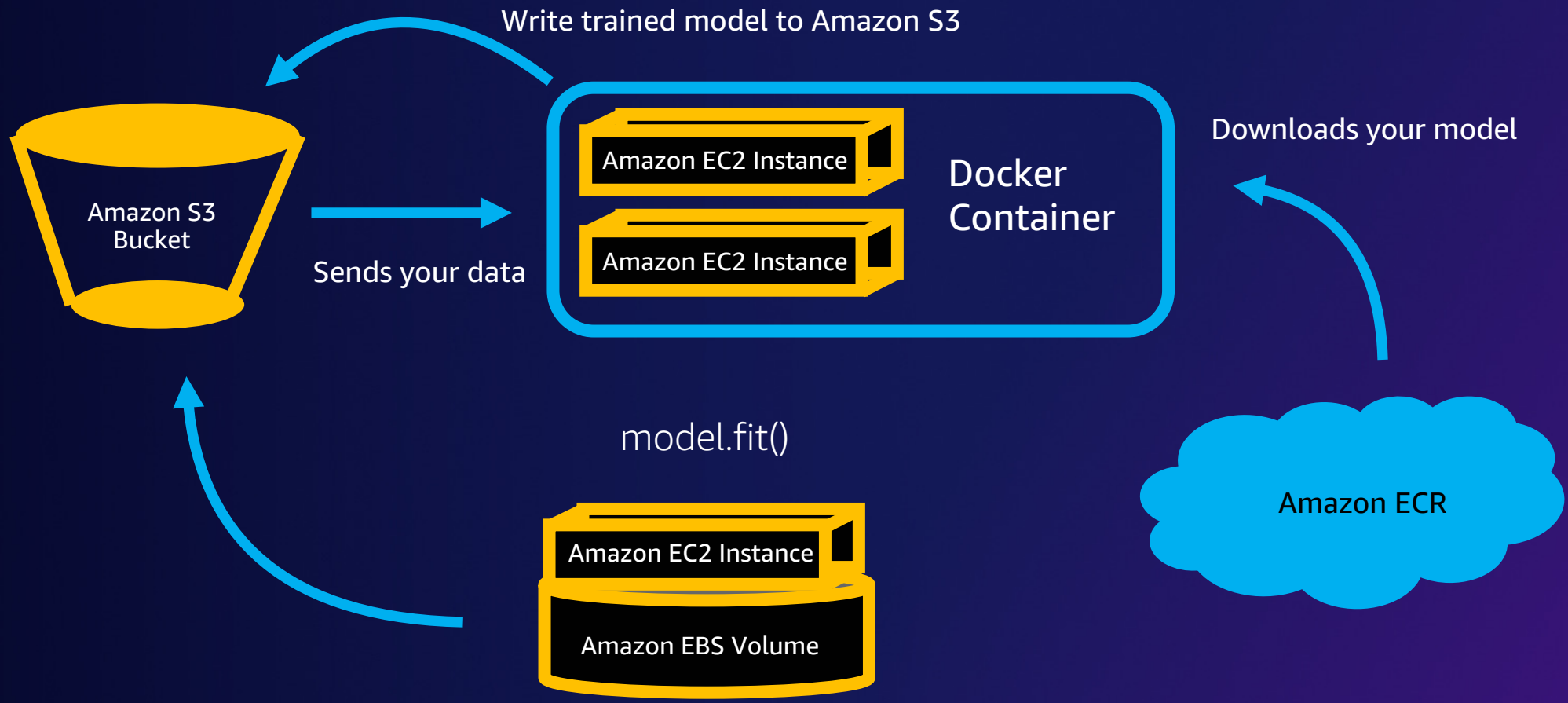
AMAZON SAGEMAKER STUDIO IDE
Prepare data | Store features | Detect bias | Build with notebooks | Train models | Tune parameters | Deploy in production | Explain predictions | Manage & monitor

Manage edge devices

ML FRAMEWORKS & INFRASTRUCTURE

TensorFlow, PyTorch, Apache MXNet, Hugging Face | Deep learning AMIs & containers | Amazon EC2 | CPUs | GPUs | AWS Inferentia | AWS Trainium | Habana Gaudi | FPGA | Elastic Inference

Amazon SageMaker training ephemeral clusters



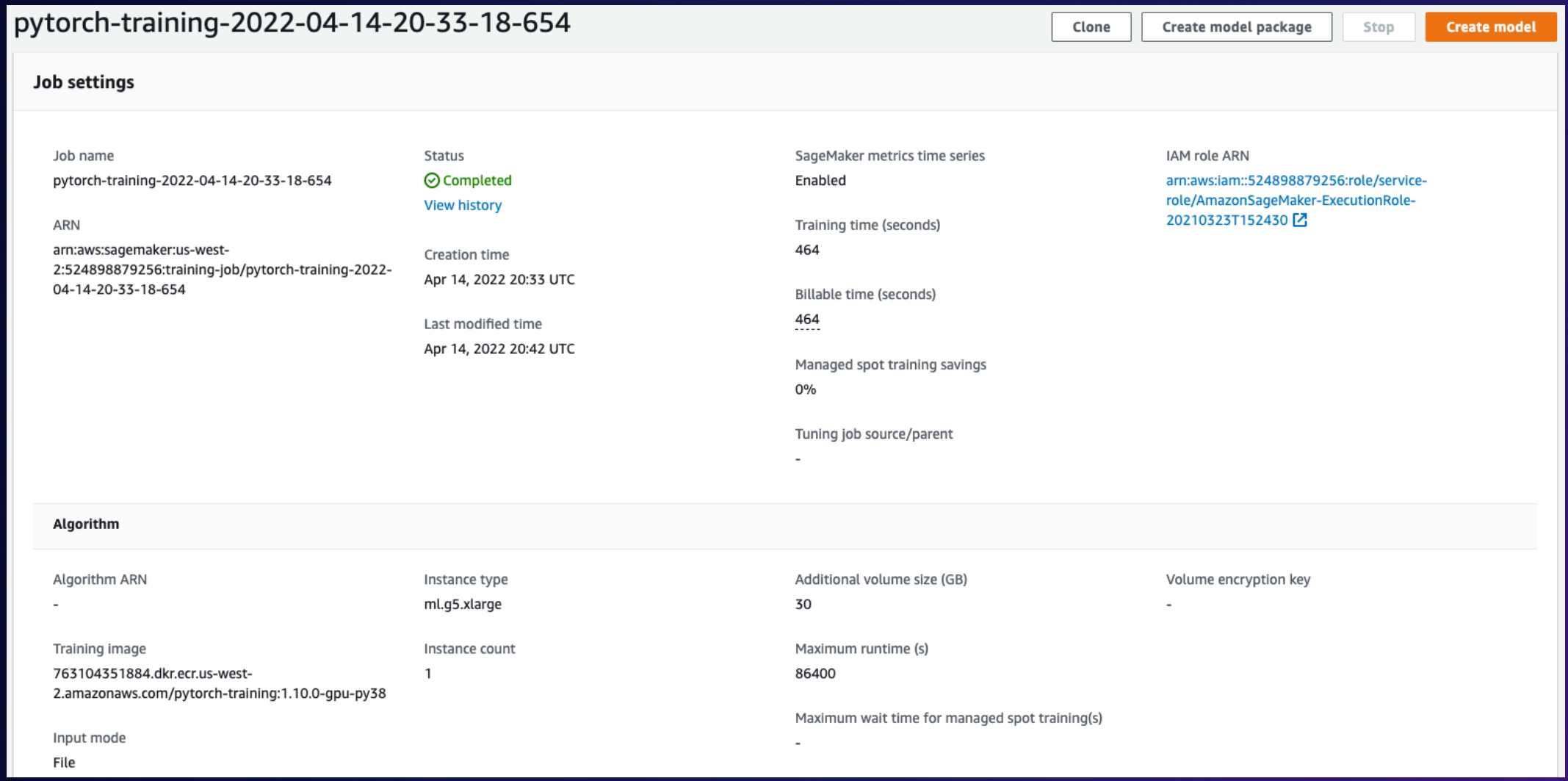
Many ways to train models on SageMaker

1. Built-in Algorithms
2. Script Mode
3. Docker container
4. Locally
5. In a pipeline

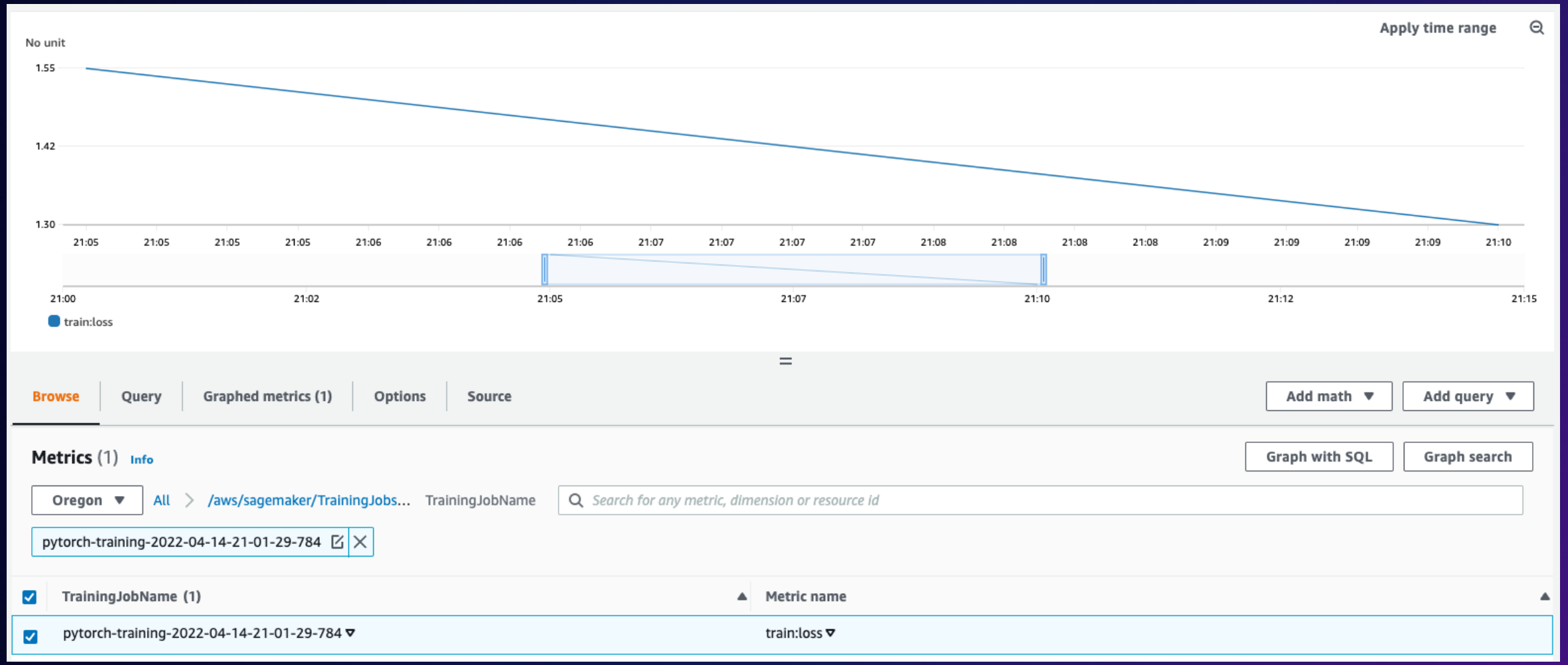
Train with your own deep learning model

```
from sagemaker.pytorch import PyTorch

estimator = PyTorch(entry_point='./cifar10.py',
                    role=role,
                    framework_version='1.10.0',
                    py_version='py38',
                    instance_count=1,
                    instance_type='ml.g5.xlarge',
                    metric_definitions=[{'Name': 'train:loss', 'Regex': 'loss: (.*)'}])

estimator.fit(inputs)
```

View training job in AWS Console



Monitor metrics in CloudWatch



Train and deploy 10K models with Hugging Face

```
from sagemaker.huggingface import HuggingFace

# hyperparameters, which are passed into the training job
hyperparameters={'epochs': 1,
                 'train_batch_size': 32,
                 'model_name':'distilbert-base-uncased'
                 }

huggingface_estimator = HuggingFace(entry_point='train.py',
                            source_dir='./scripts',
                            instance_type='ml.p3.2xlarge',
                            instance_count=1,
                            role=role,
                            transformers_version='4.12',
                            pytorch_version='1.9',
                            py_version='py38',
                            hyperparameters = hyperparameters)

huggingface_estimator.fit({'train': training_input_path, 'test': test_input_path})
```


https://docs.aws.amazon.com/sagemaker/latest/dg/hugging-face.html

SageMaker Training Compiler

Accelerate deep learning model training

Speed up training by as much as 50%

Minimal code changes required

Enable in minutes without any changes to workflow

Lower training costs

Free to use on SageMaker and additional savings from shortened training jobs

SageMaker Training Compiler

```
from sagemaker.huggingface import HuggingFace
from sagemaker.huggingface import TrainingCompilerConfig

# hyperparameters, which are passed into the training job
hyperparameters={'epochs': 1,
                 'train_batch_size': 32,
                 'model_name':'distilbert-base-uncased'
                 }

huggingface_estimator = HuggingFace(entry_point='train.py',
                            source_dir='./scripts',
                            instance_type='ml.p3.2xlarge',
                            instance_count=1,
                            role=role,
                            transformers_version='4.12',
                            pytorch_version='1.9',
                            py_version='py38',
                            hyperparameters = hyperparameters,
                            compiler_config=TrainingCompilerConfig())

huggingface_estimator.fit({'train': training_input_path, 'test': test_input_path})
```

Distributed Training

Distributed Training - Datasets

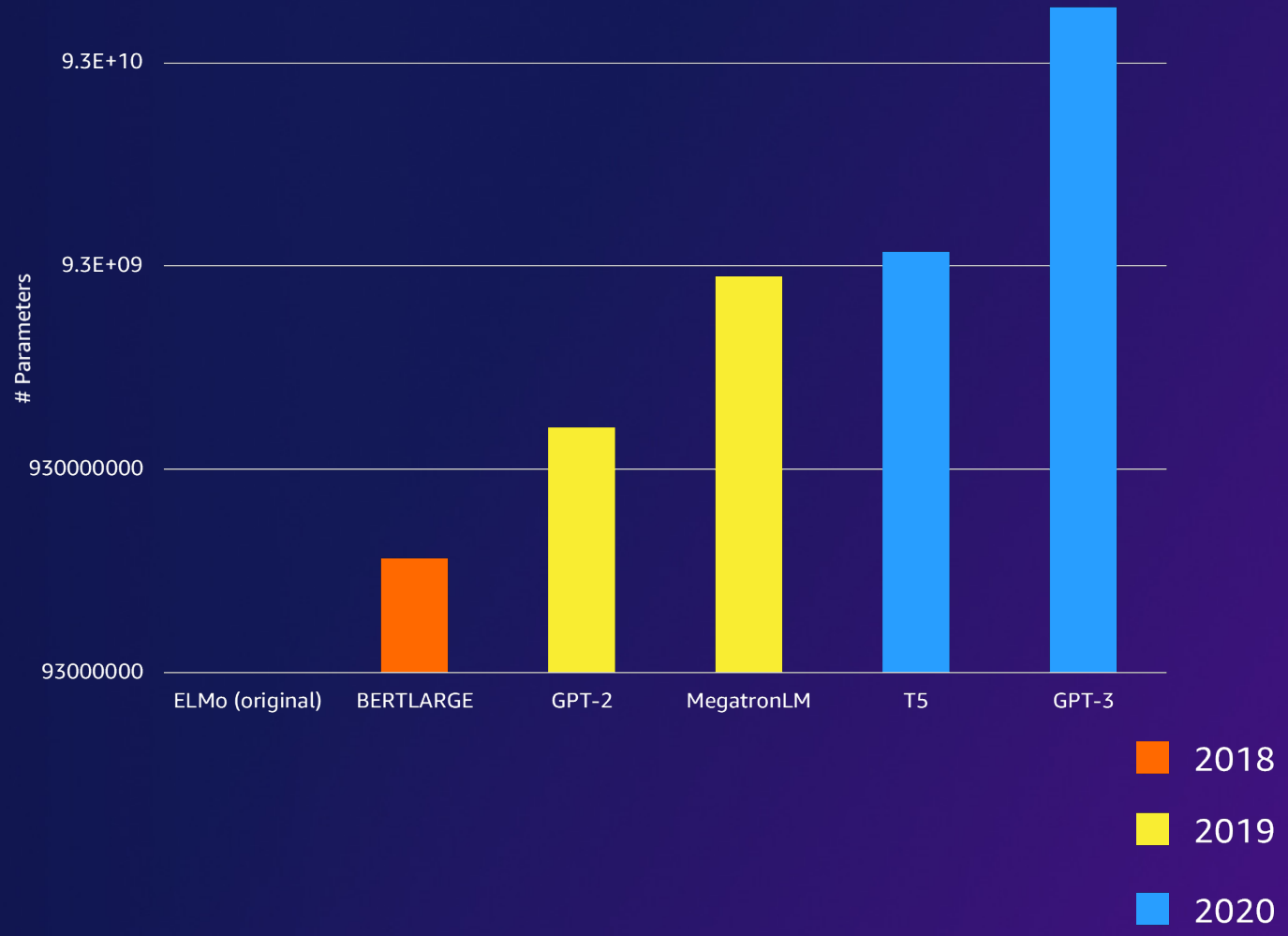
ImageNet

Data	2009	2020
Images	3.2 million	14.2 million
Categories	5,248	21,338

Version 1.1	100,000+ Q&A pairs
Version 2	+50,000 unanswerable Q's

Distributed Training - Models

From
93.6 million parameters in 2018 (ELMo) to
175 billion parameters in 2020 (GPT-3)



Distributed Training Solutions

Data parallelism

- Training dataset is split across multiple nodes.
- Each node has a replica of the model.
- Each node performs a forward and backward pass and synchronizes gradients.
- Still requires to fit the model and at least one record in memory (GPU/CPU).
- Additional complexity of computing gradients.

Model parallelism

- The model is partitioned across multiple nodes.
- Each node contains a subset of the model through which the data flows and the transformations are shared and complied.
- Efficiency driven by a pipeline execution schedule.

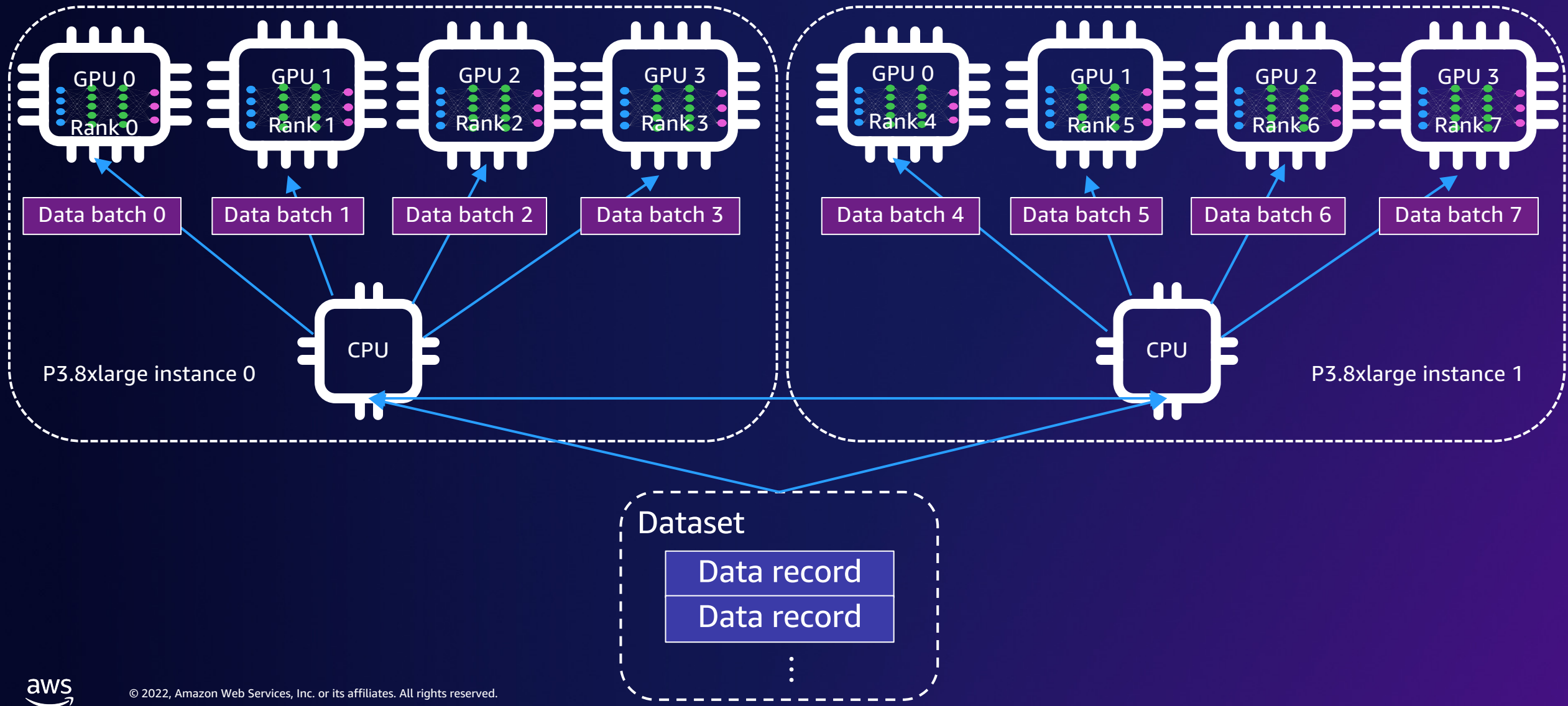
SageMaker Distributed Training Libraries

SageMaker Data Parallel (DDP)

SageMaker Model Parallel (SMP)

20%-40% faster and cheaper than NCCL and MPI based solutions

Distributed Training – Data Parallel (DDP)



SageMaker DDP – Training Script Updates

- Import the SageMaker Data Distributed library client & module

```
import smdistributed.dataparallel.torch.distributed as dist

from smdistributed.dataparallel.torch.parallel.distributed import DistributedDataParallel as DDP

dist.init_process_group()
```

- Resize the batch - by dividing by number of nodes

```
batch_size //= dist.get_world_size()
batch_size = max(batch_size, 1)
```

- Pin each GPU to a single SageMaker data parallel library rank

```
torch.cuda.set_device(dist.get_local_rank())
```

SageMaker DDP – Training Script Updates

- Wrap your model with the library's DDP

```
model = ...
# Wrap model with the library's DistributedDataParallel
model = DDP(model)
```

- Modify the torch.utils.data.distributed.DistributedSampler

```
train_sampler = DistributedSampler(train_dataset, num_replicas=dist.get_world_size(), rank=dist.get_rank())
```

- Finally save only checkpoints on the leader node

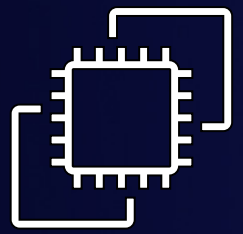
```
    # SageMaker data parallel: Save model on master node.
    if dist.get_rank() == 0:
        torch.save(...)
```

SageMaker DDP – Running The Training Job

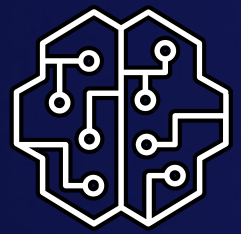
- ✓ Pre built TensorFlow or PyTorch containers or extend pre-built containers using the latest versions
- ✓ Create a estimator with distribution strategy as smdistributed dataparallel
- ✓ Optionally specify any custom Message Parsing Interface (mpi) options using the custom_mpi_options parameter in the Estimator

```
from sagemaker.pytorch import PyTorch
pt_estimator = PyTorch(
    base_job_name="training_job_name_prefix",
    entry_point="pt-train.py",
    role="SageMakerRole",
    # You must set py_version to py36
    py_version="py36",
    framework_version="1.8.1",
    # For training with multi node distributed training, set this count.
    # Example: 2,3,4,..8
    instance_count=2,
    # For training with p3dn instance use - ml.p3dn.24xlarge
    instance_type="ml.p3.16xlarge",
    # Training using smdistributed.dataparallel Distributed Training Framework
    distribution={"smdistributed": {"dataparallel": {"enabled": True}}}
)

pt_estimator.fit("s3://bucket/path/to/training/data")
```

Distributed Training – Model Parallel (SMP)



Automated model partitioning



Interleaved pipelined training

Managed SageMaker training

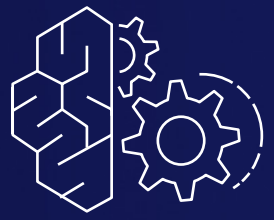
Clean framework integration

1. Use SMP to automate your model partitioning

ANALYZED MODEL

- Graph structure
- Sizes of trainable weights
- Sizes of exchanged tensors (using SageMaker Debugger)

RUN GRAPH PARTITIONING ALGORTIHM



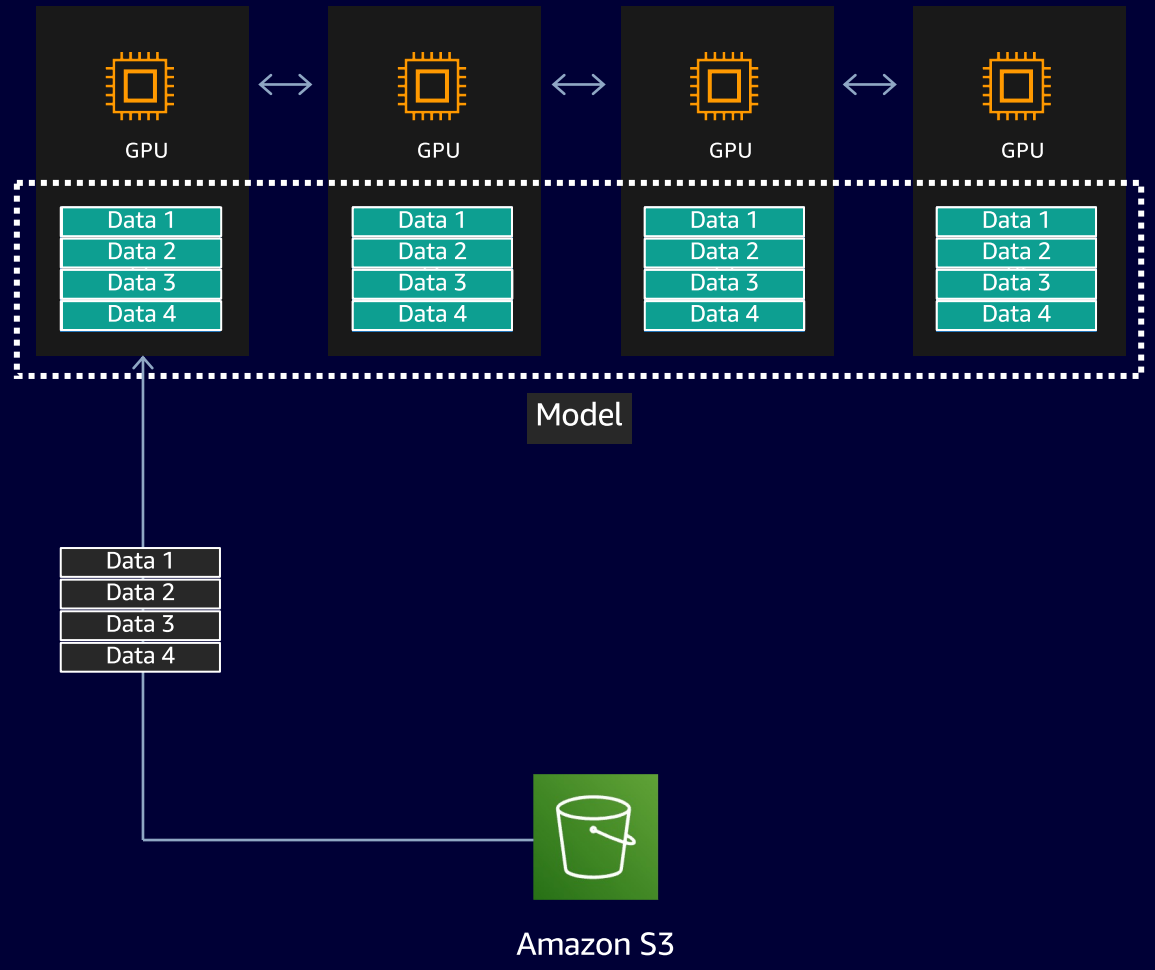
- Balance stored weights and activations
- Minimize communication

PLACE PARTITIONS ON DEVICES

- To be executed in a pipelined manner

2. Pipeline execution to stabilize GPU utilization

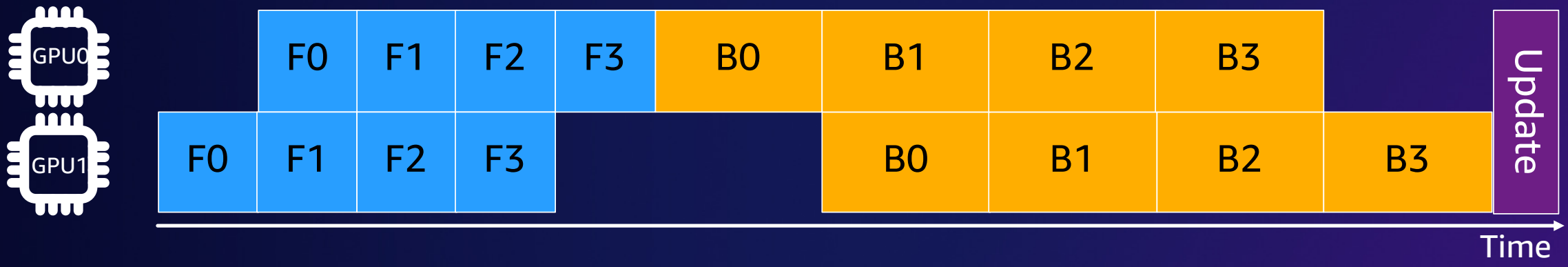
- Split minibatches into N “microbatches”
- Feed microbatches sequentially, but process them to keep GPU utilization more even
- Minimize “idle” time on GPUs



Pipeline execution schedule

Simple

F0 = forward pass, micro-batch 0
B1 = backward pass, micro-batch 1



Interleaved

Hyperparameter Tuning

The model features

Extra math

Try as much as you can

Hyperparameter Tuning Methods

- Grid Search
- Random Search
- Bayesian Search

What we have talked about so far

- SageMaker Training core capabilities
- Scale and distributed training
- Hyperparameter Tuning

Cost reduction with managed Spot Instances

```
from sagemaker.pytorch import PyTorch

estimator = PyTorch(entry_point='./cifar10.py',
                    role=role,
                    framework_version='1.10.0',
                    py_version='py38',
                    instance_count=1,
                    instance_type='ml.g5.xlarge',
                    use_spot_instances=True,
                    max_wait=86400,
                    metric_definitions=[{'Name': 'train:loss', 'Regex': 'loss: (.*)'}])

estimator.fit(inputs)
```

Up to 90% cost reduction

Getting Started

Product page: aws.amazon.com/sagemaker/train/

Documentation: docs.aws.amazon.com/sagemaker/latest/dg/how-it-works-training.html

SageMaker examples GitHub: github.com/aws/amazon-sagemaker-examples

AWS ML Blogs: aws.amazon.com/blogs/machine-learning/category/artificial-intelligence/sagemaker/

Thank you!

Gal Oshri
@galoshri

Sam Palani
@samx18

Please complete the session survey

“

”

Learn in-demand AWS Cloud skills

AWS Skill Builder

Access **500+ free** digital courses and Learning Plans

Explore resources with a variety of skill levels and **16+** languages to meet your learning needs

Deepen your skills with digital learning on demand

Train now

AWS Certifications

Earn an industry-recognized credential

Receive Foundational, Associate, Professional, and Specialty certifications

Join the **AWS Certified community** and get exclusive benefits

Access **new** exam guides

aws