



# aws INNOVATE

AI/ML EDITION

24 February 2022

# **Train ML models quickly and cost-effectively with Amazon SageMaker**

Alex Thewsey

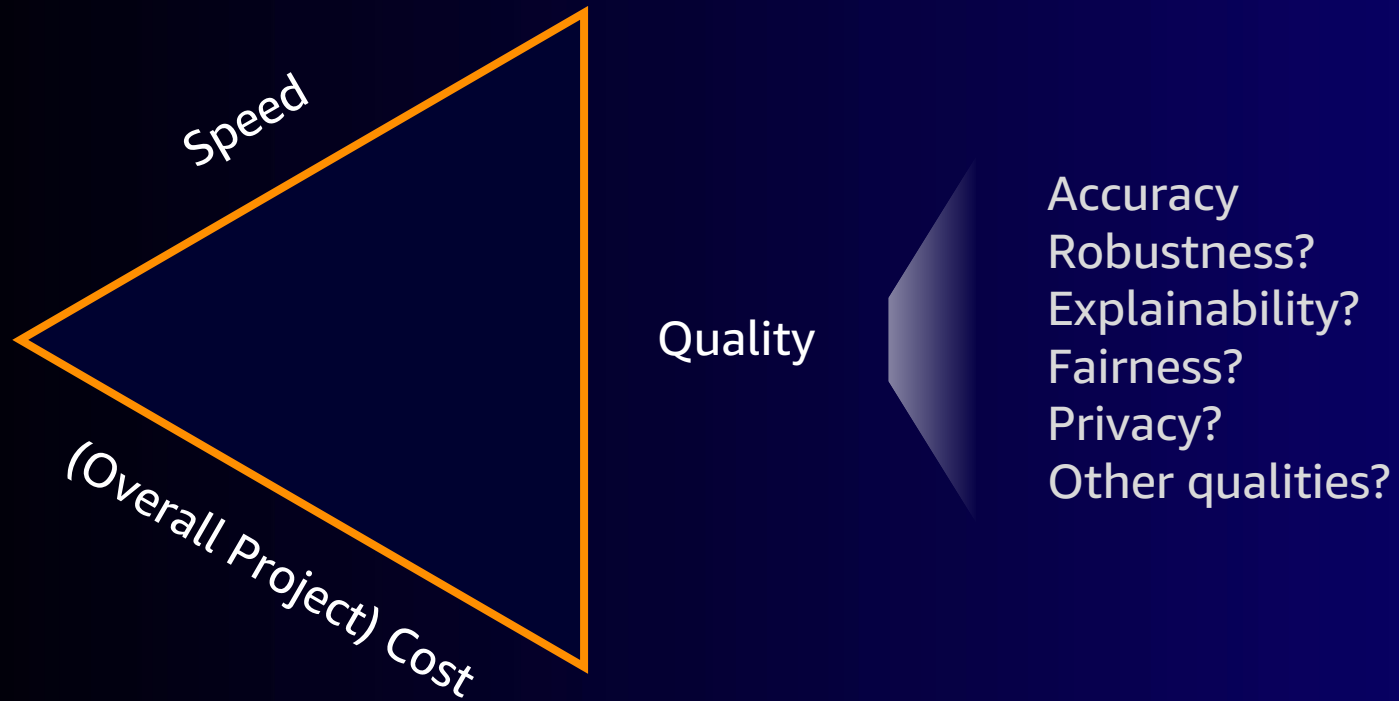
AI/ML Specialist Solutions Architect, AWS



# Agenda

1. **Why** optimize model training, and what to optimize for?
2. **How** model training works on Amazon SageMaker
3. **Tools and tips** for efficient training
4. **Demo**
5. **Recap** and resources

# Priorities first: What are we optimizing for?



# Priorities first: Training is only part of the story

Inference may drive  
up to

90%

of ML project  
infrastructure costs

Focus on training can be a

**symptom**

of organizational blockers to production deployment

# Priorities first: What are we optimizing for?

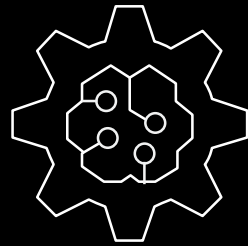


# Make use of pre-built algorithms and services



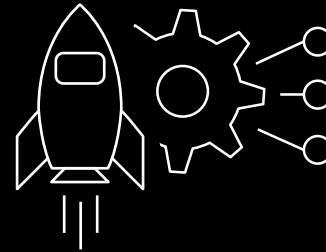
## AWS AI Services

Fully-managed services for both pre-trained and bring-your-own-data AI applications



## SageMaker Built-In Algos & Marketplace

17+ families of SageMaker-native algorithms, plus more via AWS Marketplace



## SageMaker Autopilot

Automated but transparent, end-to-end learning for tabular data



## SageMaker JumpStart

Deployable ML solution templates, additional algorithms and pre-trained public models

# Understanding Amazon SageMaker infrastructure

“Work *from* the notebook, not *on* the notebook”

The screenshot displays the Amazon SageMaker Studio interface, which is a cloud-based environment for developing machine learning models. The interface is divided into several panels:

- Code Editor:** The central panel shows a Jupyter notebook titled "Computing Anomaly Scores". The code defines a function to compute anomaly scores from a taxi dataset using a random forest model. It includes comments and a final section that prints and plots data points with scores greater than 3 standard deviations from the mean score.
- Output Console:** The bottom-left panel shows the output of the code execution, including the first few values of the taxi data and the resulting anomaly scores.
- Trial Component Chart:** The top-right panel displays a chart showing the trial components. The chart is a scatter plot with the x-axis representing the timestamp and the y-axis representing the test metric. The chart shows several data points, with one point highlighted as an anomaly.
- Trial Component List:** The bottom-right panel shows a table of trial components. The table lists the status, experiment, type, trial, trial component, and monitor for each component.

The interface also includes a sidebar on the left with navigation icons and a top bar with the SageMaker Studio logo and navigation links.



# Understanding Amazon SageMaker infrastructure

“Work *from* the notebook, not *on* the notebook”

The screenshot displays the Amazon SageMaker Studio interface. The central pane shows a Jupyter notebook titled "random\_cut\_forest.ipynb" with the following code:

```
[ ]: results = rcf_inference.predict(taxi_data_numpy)
scores = [datum['score'] for datum in results['scores']]

# add scores to taxi data frame and print first few values
taxi_data['score'] = pd.Series(scores, index=taxi_data.index)
taxi_data.head()
```

```
[ ]: fig, ax1 = plt.subplots()
ax2 = ax1.twinx()

# *Try this out* - change 'start' and 'end' to zoom in on the
# anomaly found earlier in this notebook
#
start, end = 0, len(taxi_data)
start, end = 5000, 6500
taxi_data_subset = taxi_data[start:end]

ax1.plot(taxi_data_subset['value'], color='C0', alpha=0.8)
ax2.plot(taxi_data_subset['score'], color='C1')

ax1.grid(which='major', axis='both')

ax1.set_ylabel('Taxi Ridership', color='C0')
ax2.set_ylabel('Anomaly Score', color='C1')

ax1.tick_params(y, colors='C0')
ax2.tick_params(y, colors='C1')

ax1.set_ylim(0, 40000)
ax2.set_ylim(min(scores), 1.4*max(scores))
fig.set_figwidth(10)
```

Below the code, there is a text block explaining the anomaly score spikes and a list of known anomalous events.

On the right side, there is a "Trial Component Chart" showing a scatter plot of "test-metric" vs "timestamp". The chart displays data points for different trials, with a legend indicating the trial names.

Below the chart, there is a "Trial Component List" table showing the status of various trial components.

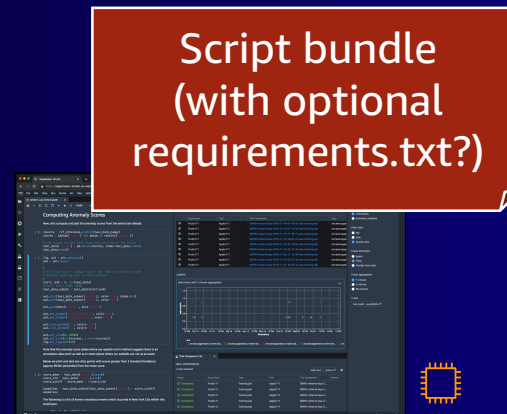
Status	Experiment	Type	Trial	Trial component	Monitor
Completed	Fruits111	Training job	Apple111	DEMO-minerva-byo-2...	
Completed	Fruits111	Training job	Apple111	DEMO-minerva-byo-2...	
Completed	Fruits111	Training job	Apple111	DEMO-minerva-byo-2...	
Completed	Fruits111	Training job	Apple111	DEMO-minerva-byo-2...	
Completed	Fruits111	Training job	Apple111	DEMO-minerva-byo-2...	

```
import boto3 # General-purpose AWS SDK for Python
import sagemaker # High-level Python SDK for SageMaker
# (Both open-source & available on PyPI)
```

# Understanding Amazon SageMaker infrastructure

“Work *from* the notebook, not *on* the notebook”

Development environment:  
e.g. Amazon SageMaker Studio  
Amazon SageMaker Notebook  
Instances



# Understanding Amazon SageMaker infrastructure

“Work **from** the notebook, not **on** the notebook”



Amazon Elastic Container Registry (Amazon ECR)

AWS Deep Learning Container images  
(for TensorFlow, PyTorch, MXNet, etc)



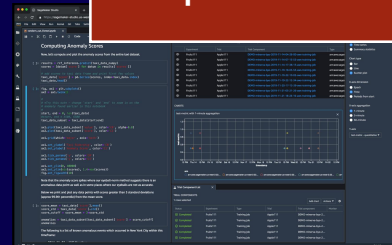
Amazon Simple Storage Service (Amazon S3)

Input A

Input B

Development environment:  
e.g. Amazon SageMaker Studio  
Amazon SageMaker Notebook Instances

Script bundle  
(with optional  
requirements.txt?)



# Understanding Amazon SageMaker infrastructure

“Work **from** the notebook, not **on** the notebook”



Amazon Elastic Container Registry (Amazon ECR)

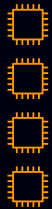
AWS Deep Learning Container images  
(for TensorFlow, PyTorch, MXNet, etc)



Amazon Simple Storage Service (Amazon S3)

Input A

Input B



Amazon SageMaker  
container runtime

Base Libraries

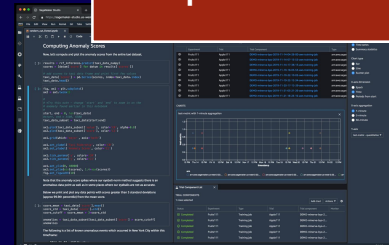
Boilerplate Code

Hyperparameters

Input Data

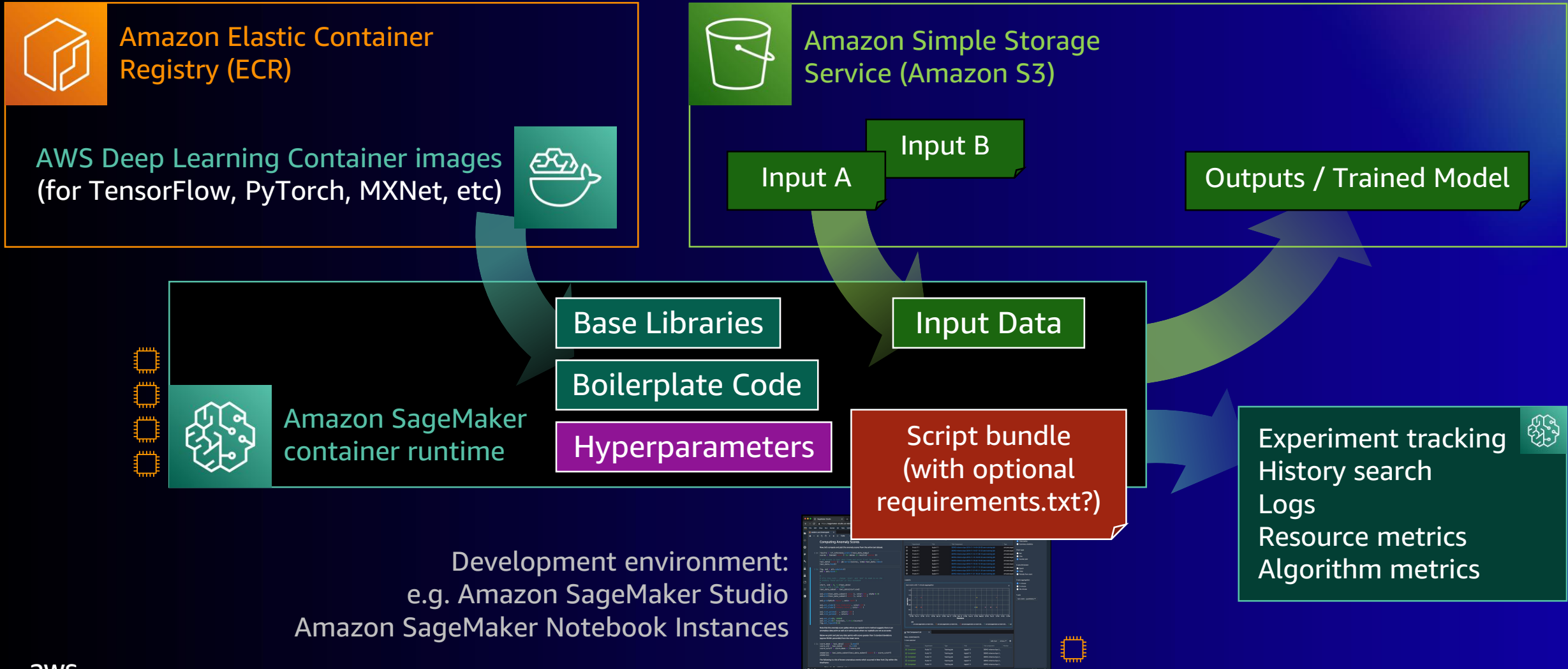
Script bundle  
(with optional  
requirements.txt?)

Development environment:  
e.g. Amazon SageMaker Studio  
Amazon SageMaker Notebook Instances



# Understanding Amazon SageMaker infrastructure

“Work **from** the notebook, not **on** the notebook”



# Training compute setup and managed spot

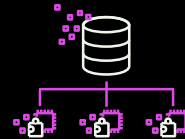
(In your notebook)

```
estimator = Estimator(  
    ...,  
    use_spot_instances=True,  
    max_run=60*60*4, # 4hrs training  
    max_wait=60*60*8, # 8hrs total  
    checkpoint_s3_uri="s3://...",  
  
    instance_count=4,  
    instance_type="ml.p3.8xlarge",  
    ...,  
)
```



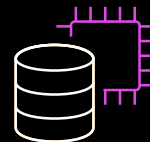
## Enabling Amazon SageMaker Managed Spot is simple

- Save up to 90% on compute costs!
- But as good practice, you should implement checkpointing



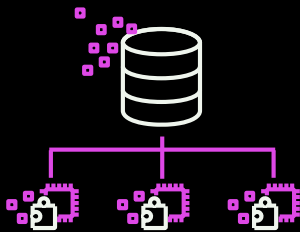
## Before scaling out, check:

- Your framework and script are set up for multi-node training (not just duplicating!)
- Input channel distributions (shard or replicate?)



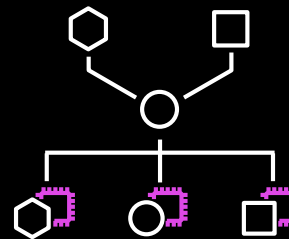
## Select appropriate instance type

# Amazon SageMaker distributed training libraries



## Data Parallelism

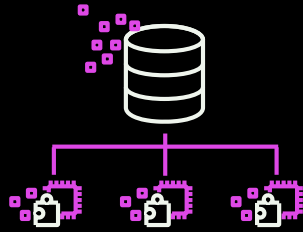
Scale out training clusters with near-linear efficiency, optimized for AWS networking and instance topology



## Model Parallelism

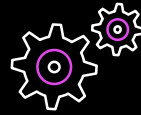
Train models too large to fit within GPU memory, with automated model splitting and sophisticated pipeline scheduling

# Optimize training with Amazon SageMaker Data Parallel



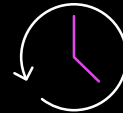
## Data Parallelism

Scale out training clusters with near-linear efficiency, optimized for AWS networking and instance topology



### Support for popular ML framework APIs

Re-use existing APIs such as Horovod and PyTorch DistributedDataParallel



### Reduced training time

~25% faster with synchronization across GPUs (as tested with BERT)



### Minimal code change

See SM Developer Guide for PyTorch & TensorFlow instructions – or use Hugging Face Trainer API scripts with **no code changes at all!**



# Tune (hyper)-parameters automatically

(In your training script)

Read in parameters via  
`/opt/ml/input/config/hyperparameters.json`  
or CLI arguments

`print()` / log metrics to the console

(In your notebook)

Define how Amazon SageMaker should **scrape metrics** from  
your training job logs **via RegEx**:

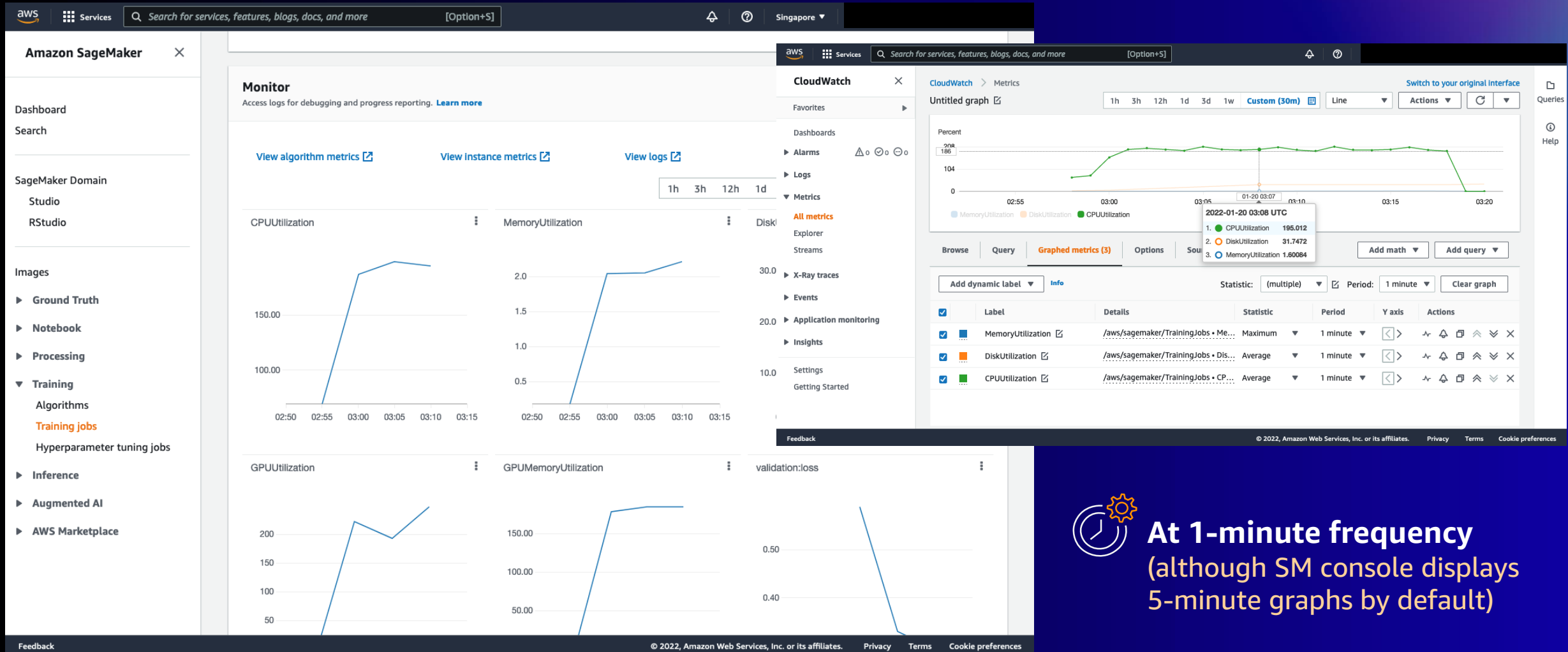
```
metric_definitions = [  
    {"Name": "loss", "Regex": r"'loss': (-?[0-9\\.\\-e]+[,}]")},  
]
```

The screenshot displays the Amazon SageMaker Studio interface. The top panel shows the 'Hyperparameters' section with a table of key-value pairs:

Key	Value
batch-size	128
epochs	12
model_dir	"s3://sagemaker-ap-southeast-1-123456789012/mnist-keras-2021-12-06-04-13-13-564/model"
sagemaker_container_log_level	20
sagemaker_job_name	"mnist-keras-2021-12-06-04-13-13-564"
sagemaker_parameter_server_enabled	true
sagemaker_program	"main.py"

The bottom panel shows the 'Experiments and trials' section with a list of trial components. The right sidebar displays a 'CHART PROPERTIES' panel with a line chart titled 'validationloss\_avg\_last with 1-minute aggregation'. The chart shows a decreasing trend in validation loss over time, with the Y-axis labeled 'validationloss\_avg\_last' and the X-axis labeled 'time'.

# Default Amazon SageMaker metrics via Amazon CloudWatch



**At 1-minute frequency**  
(although SM console displays  
5-minute graphs by default)

# Tune (hyper)-parameters automatically

(In your training script)

Read in parameters via  
/opt/ml/input/config/hyperparameters.json  
or CLI arguments

print() / log metrics to the console

(In your notebook)

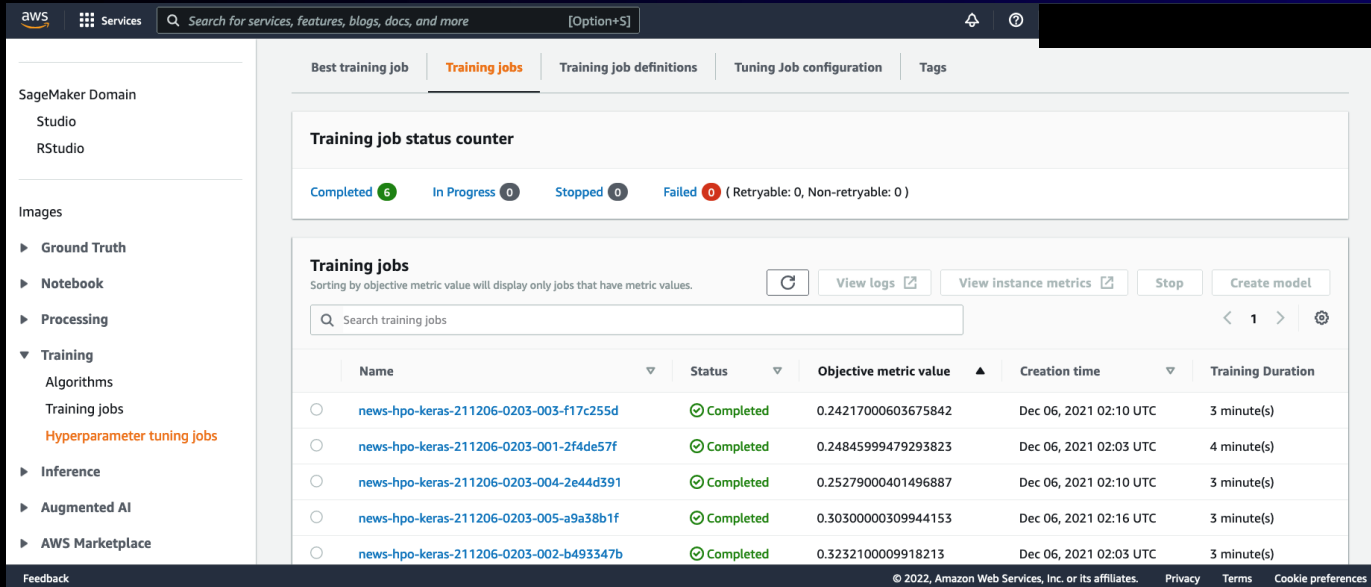
Define how Amazon SageMaker should **scrape metrics** from  
your training job logs **via RegEx**:

```
metric_definitions = [  
    {"Name": "loss", "Regex": r"'loss': (-?[0-9\\.\\-e]+[,}]")},  
]
```

Wrap your training job definition (or 'estimator')  
with automatic hyperparameter tuning!

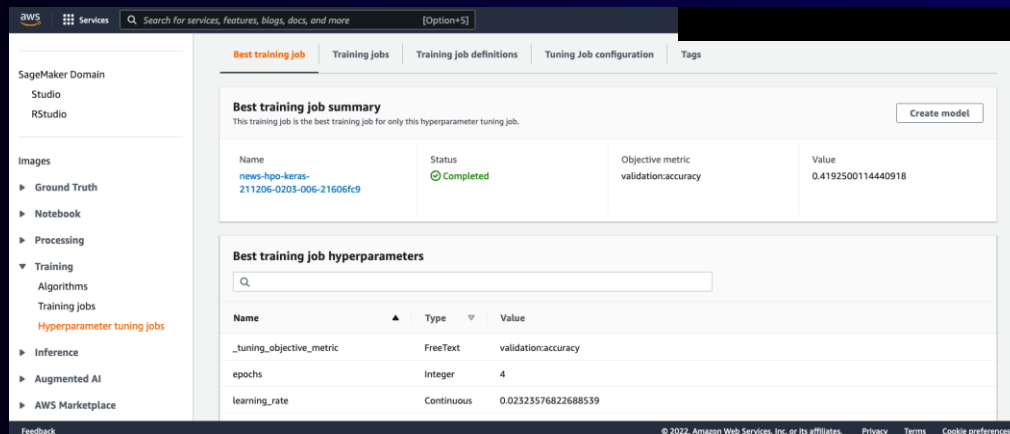
```
sagemaker.tuner.HyperparameterTuner(  
    estimator=...,  
    metric_definitions=[...],  
    objective_metric_name="loss",  
    objective_type="Minimize",  
    hyperparameter_ranges={  
        "learning_rate":  
sagemaker.parameter.ContinuousParameter(  
    min_value=1e-8,  
    max_value=1e-3,  
    scaling_type="Logarithmic",  
    ),  
    ...  
},  
    strategy="Bayesian",  
    max_jobs=50,  
    max_parallel_jobs=5,  
    ...  
)
```

# Tune (hyper)-parameters automatically



The screenshot shows the AWS SageMaker console interface. The left sidebar contains navigation links for SageMaker Domain, Studio, RStudio, Images, and various training and inference tasks. The main panel is titled 'Training jobs' and shows a status counter with 6 Completed, 0 In Progress, 0 Stopped, and 0 Failed jobs. Below the counter is a table of training jobs, all of which are 'Completed'.

Name	Status	Objective metric value	Creation time	Training Duration
news-hpo-keras-211206-0203-003-f17c255d	Completed	0.24217000603675842	Dec 06, 2021 02:10 UTC	3 minute(s)
news-hpo-keras-211206-0203-001-2f4de57f	Completed	0.24845999479293823	Dec 06, 2021 02:03 UTC	4 minute(s)
news-hpo-keras-211206-0203-004-2e44d391	Completed	0.25279000401496887	Dec 06, 2021 02:10 UTC	3 minute(s)
news-hpo-keras-211206-0203-005-a9a38b1f	Completed	0.30300000309944153	Dec 06, 2021 02:16 UTC	3 minute(s)
news-hpo-keras-211206-0203-002-b493347b	Completed	0.3232100009918213	Dec 06, 2021 02:03 UTC	3 minute(s)



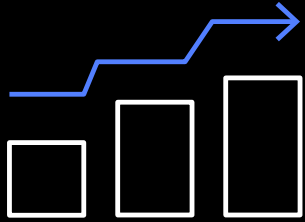
The screenshot shows the 'Best training job summary' and 'Best training job hyperparameters' in the AWS SageMaker console. The summary shows the job is 'Completed' with a validation accuracy of 0.4192500114440918. The hyperparameters table lists the tuning objective metric, epochs, and learning rate.

Name	Type	Value
_tuning_objective_metric	FreeText	validation:accuracy
epochs	Integer	4
learning_rate	Continuous	0.02323576822688539

Wrap your training job definition (or 'estimator') with automatic hyperparameter tuning!

```
sagemaker.tuner.HyperparameterTuner(  
    estimator=...,  
    metric_definitions=[...],  
    objective_metric_name="loss",  
    objective_type="Minimize",  
    hyperparameter_ranges={  
        "learning_rate":  
sagemaker.parameter.ContinuousParameter(  
    min_value=1e-8,  
    max_value=1e-3,  
    scaling_type="Logarithmic",  
    ),  
    ...  
},  
    strategy="Bayesian",  
    max_jobs=50,  
    max_parallel_jobs=5,  
    ...  
)
```

# Advanced HPO tips



## Continue a previous search

Warm-start the Bayesian optimizer to intelligently continue from previous HPO jobs

`warm_start_config` (sagemaker SDK)  
or `WarmStartConfig` (in API  
`CreateHyperParameterTuningJob`)

The screenshot shows the Amazon SageMaker console for a specific hyperparameter tuning job. The job is titled 'tuning-job-1-106c1aad6d57474cb6' and has a status of 'Completed'. It shows the job's ARN, creation time, and last modified time. Below this, there is a 'Training job definition' table with two entries.

Hyperparameter tuning job summary				
Name	Status	Approx. total training duration		
tuning-job-1-106c1aad6d57474cb6	Completed	8 hour(s), 15 minute(s)		
ARN	Creation time	Last modified time		
arn:aws:sagemaker:ap-southeast-1:123456789012:hyper-parameter-tuning-job/tuning-job-1-106c1aad6d57474cb6	Apr 15, 2021 15:13 UTC	Apr 15, 2021 16:54 UTC		

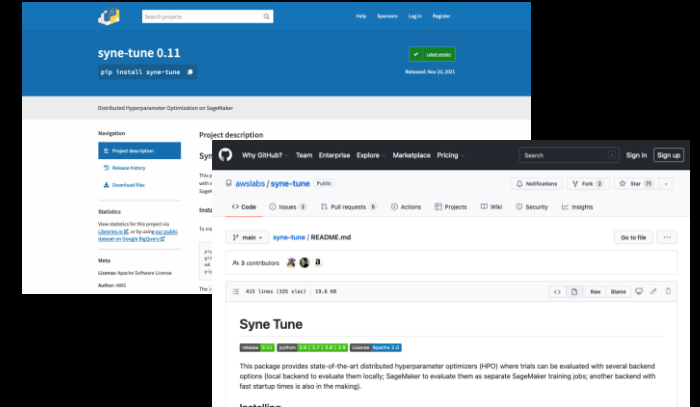
  

Training job definition				
Name	Objective metric	Type	Instance type	Instance count
cc-cls-spu-dgp0-vgb	validationf1	Maximize	ml.m5.4xlarge	1
cc-cls-spu-dgp1-l	validationbinary_f_beta	Maximize	ml.m5.4xlarge	1

## Multiple algorithms per HPO run

As used by Amazon SageMaker Autopilot! Searches may span multiple algorithms and container images

See `TrainingJobDefinitions` instead of `TrainingJobDefinition` in API!

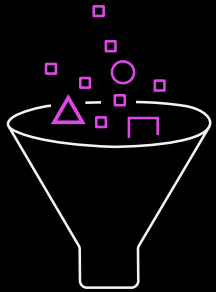


## Customize further: Syne Tune

For advanced practitioners wanting more control than Amazon SageMaker's built-in HPO – see AWS Labs' open source Syne Tune library!

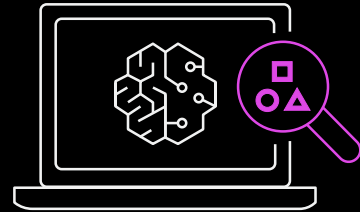
<https://github.com/aws-labs/syne-tune>

# Monitor and profile with Amazon SageMaker Debugger



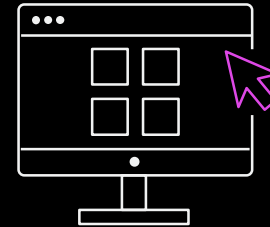
## Capture data from training jobs

Apache MXNet  
PyTorch  
TensorFlow  
XGBoost



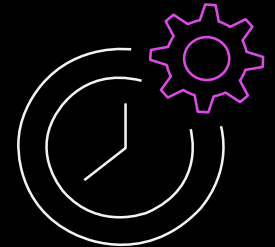
## Real-time monitoring

Get deeper visibility  
into the training  
process as it runs



## Automatic issue detection

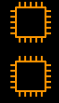
Receive alerts to find  
and fix issues early, and  
accelerate prototyping



## Resource profiling recommendations

Find bottlenecks and  
optimize compute  
resources

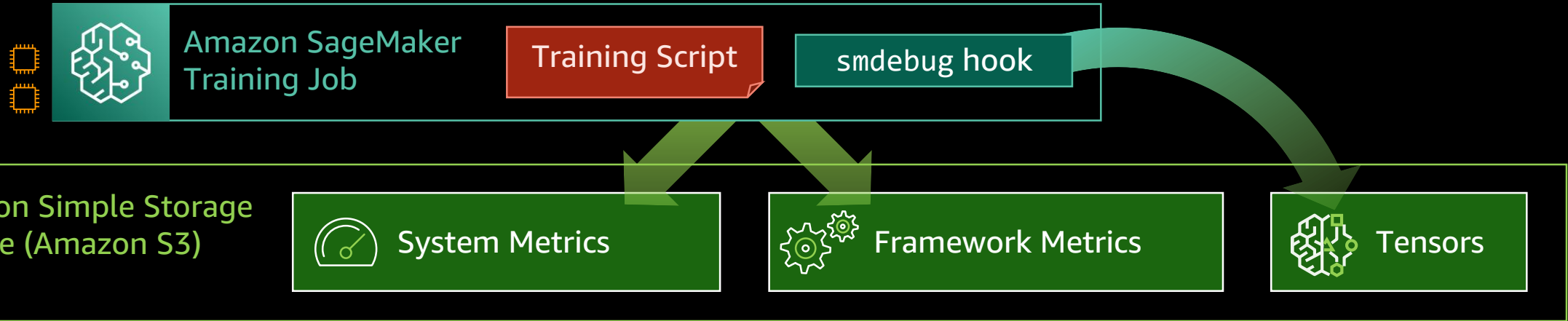
# Amazon SageMaker Debugger real-time monitoring



Amazon SageMaker  
Training Job

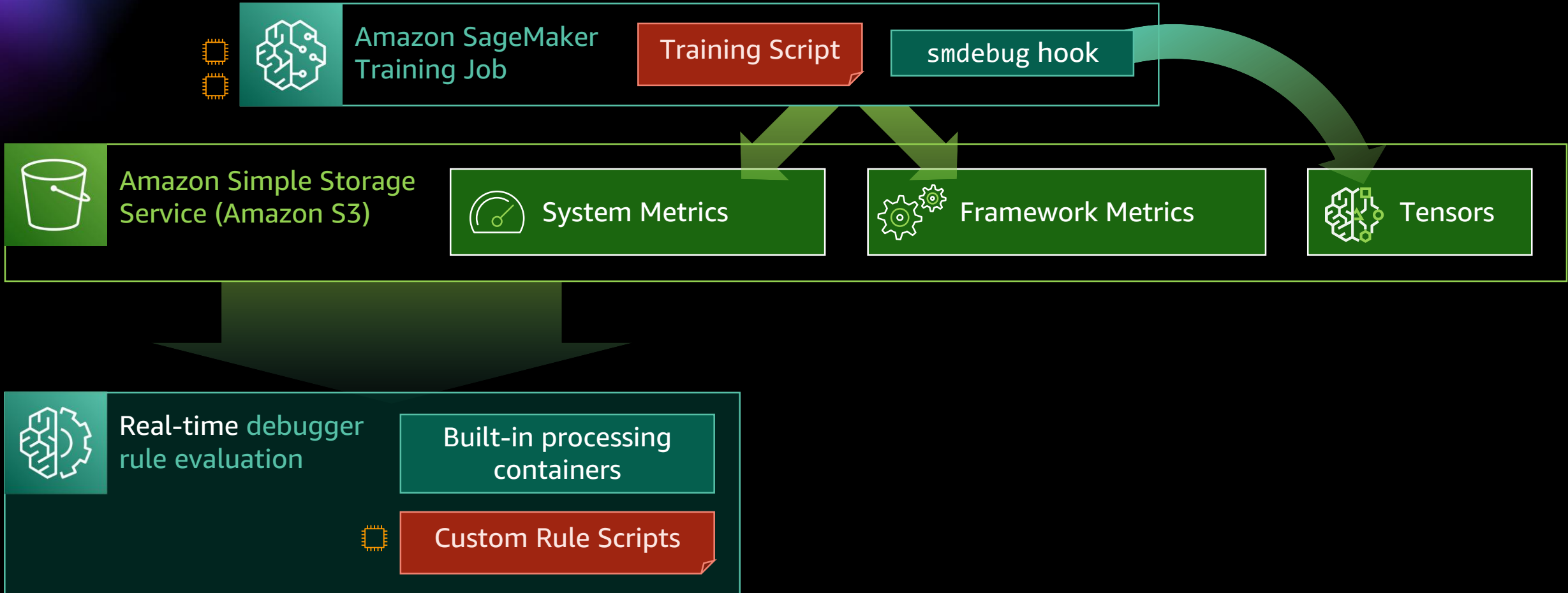
Training Script

# Amazon SageMaker Debugger real-time monitoring

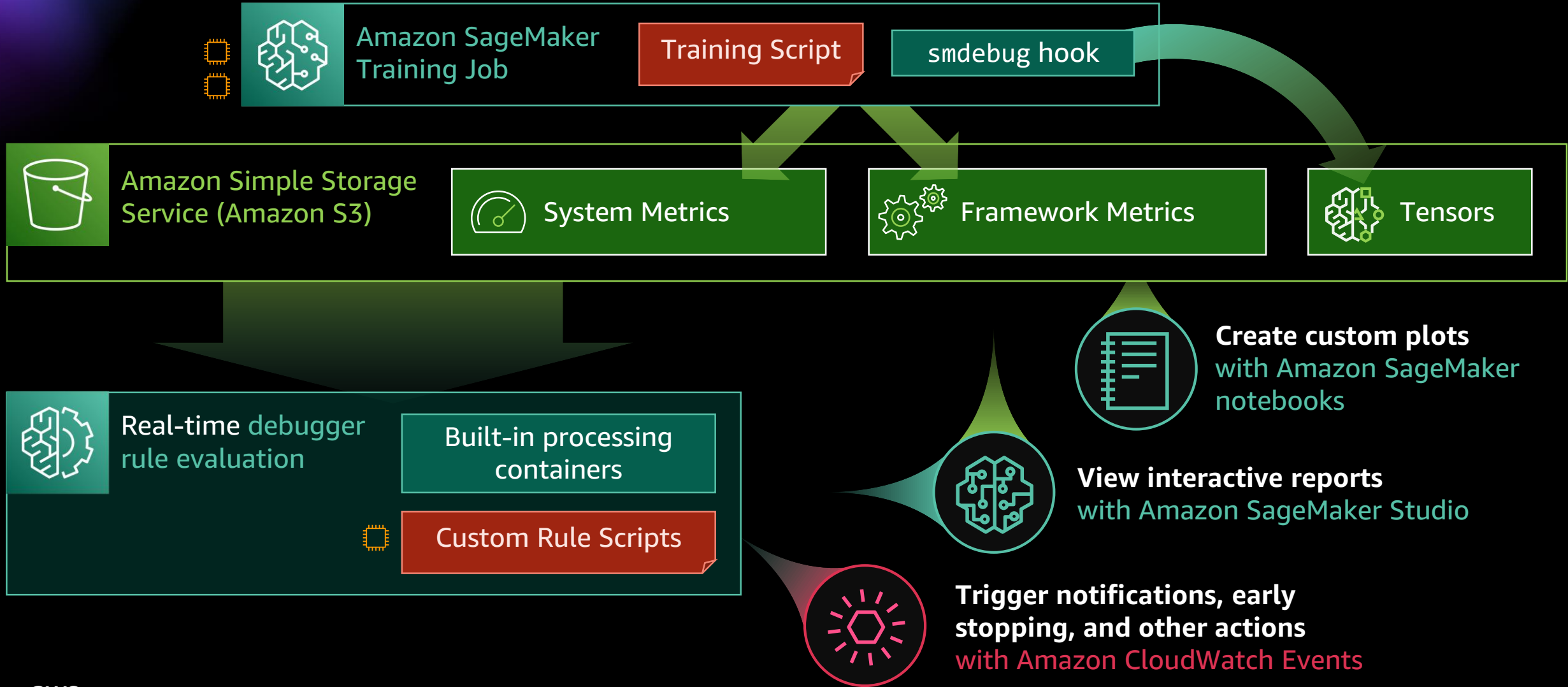




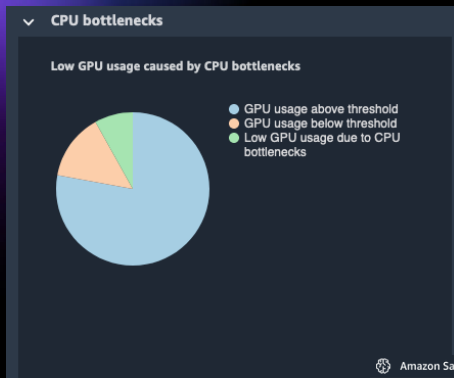
# Amazon SageMaker Debugger real-time monitoring



# Amazon SageMaker Debugger real-time monitoring

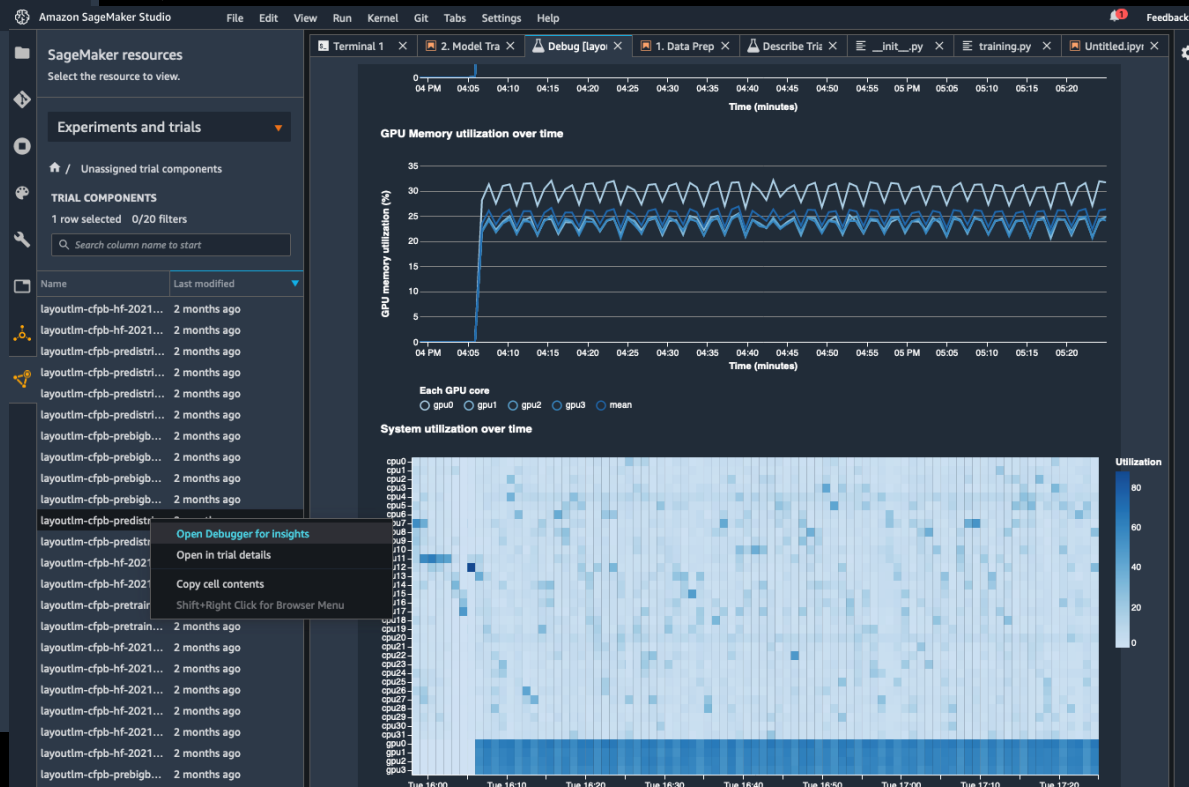


# Performance: Amazon SageMaker Debugger Profiler



```
estimator = PyTorch( # Or TensorFlow, MXNet, etc
```

```
... ,
profiler_config=sagemaker.debugger.ProfilerConfig(
    system_monitor_interval_millis=100,
    framework_profile_params=sagemaker.debugger.FrameworkProfile(),
),
```



## Granular system metrics down to 100ms interval



**Interactive reports**  
with downloadable options



## Automatic recommendations to resolve bottlenecks



### Extra options

for operator, data-loader, and  
Python function profiling



# Optimize training data input



## Amazon Simple Storage Service (Amazon S3)

Most common, feature-rich option, with advanced data lake capabilities



## Amazon FSx for Lustre

High-performance file system optimized for ML and HPC workloads



## Amazon Elastic File System (Amazon EFS)

Consider mainly just if your source data is already on Amazon EFS today

# Optimize training data input



## Amazon Simple Storage Service (Amazon S3)

Most common, feature-rich option, with advanced data lake capabilities



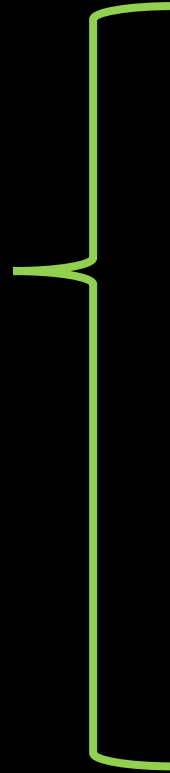
## Amazon FSx for Lustre

High-performance file system optimized for ML and HPC workloads



## Amazon Elastic File System (Amazon EFS)

Consider mainly just if your source data is already on Amazon EFS today



## File Mode (Default)

Up-front download to local filesystem before your job starts



## Pipe Mode

Stream data for serial access (usually needing code changes)



## Fast File Mode (NEW 2021)

File-like access backed by on-demand streaming!

# Optimize training data input



## Amazon Simple Storage Service (Amazon S3)

Most common, feature-rich option, with advanced data lake capabilities



## Amazon FSx for Lustre

High-performance file system optimized for ML and HPC workloads



## Amazon Elastic File System (Amazon EFS)

Consider mainly just if your source data is already on Amazon EFS today



## File Mode (Default)

Up-front download to local filesystem before your job starts



## Pipe Mode

Stream data for serial access (usually needing code changes)

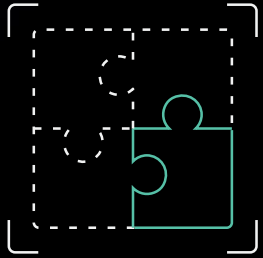


## Fast File Mode (NEW 2021)

File-like access backed by on-demand streaming!

```
estimator.fit({  
    "train": sagemaker.inputs.TrainingInput(  
        "s3://doc-example-bucket/a-folder/",  
        input_mode="FastFile",  
    ),  
    ...,  
})
```

# Other start-up optimization tips



## Don't use the whole dataset until needed

Initial, basic, functional tests should use **small subsets** of the data

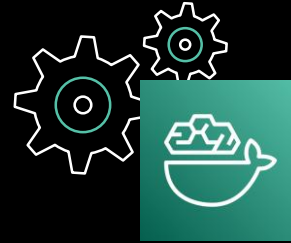


## Preprocess your data for performance

Avoid huge numbers of tiny files

Explore optimized formats like **RecordIO** & **TFRecord**

**Amazon SageMaker Processing** can help you scale out!

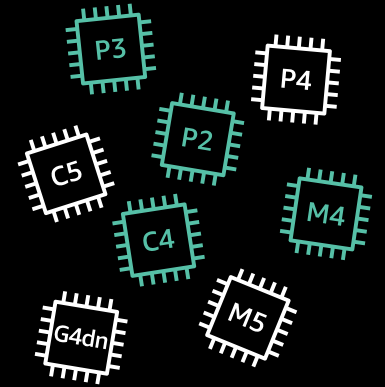


## Build-in common dependencies

`requirements.txt` is convenient but repetitive

Deriving images 'FROM' **AWS DLCs** may be easier!

Smaller containers can start jobs faster



## Instance type can affect start-up time

If working in an environment that supports **docker**, consider **Amazon SageMaker Local Mode** for initial functional tests

# Demo



# Recap and resources

## Recap

1. Amazon SageMaker Managed Spot Instances
2. Amazon SageMaker Distributed Libraries
3. Amazon Metrics and Automatic Hyperparameter Tuning
4. Amazon SageMaker Debugger and Profiler
5. Amazon Input Optimization including Fast File Mode and Amazon FSx for Lustre

## Resources

Amazon SageMaker developer guide: <https://docs.aws.amazon.com/sagemaker/latest/dg/>

Amazon SageMaker Python SDK documentation: <https://sagemaker.readthedocs.io/en/stable/>

Amazon SageMaker repository: <https://github.com/aws/amazon-sagemaker-examples>

AWS ML blog: <https://aws.amazon.com/blogs/machine-learning/>



# Visit the AI & Machine Learning resource hub for more resources

Dive deeper into these resources, get inspired and learn how you can use AI and machine learning to accelerate your business outcomes.

- The machine learning journey e-book
- 7 leading machine learning use cases e-book
- A strategic playbook for data, analytics, and machine learning e-book
- Accelerate machine learning innovation with the right cloud services & infrastructure e-book
- Choosing the right compute infrastructure for machine learning e-book
- Improving service and reducing costs in contact centers e-book
- Why ML is essential in your fight against online fraud e-book
- ... and more!



<https://bit.ly/3mwi59V>

Visit resource hub

# AWS Machine Learning (ML) Training and Certification



## AWS is how you build machine learning skills

Courses built on the curriculum leveraged by Amazon's own teams. Learn from the experts at AWS.

[aws.training/machinelearning](https://aws.training/machinelearning)



## Flexibility to learn your way

Learn online with on-demand digital courses or live with virtual instructor-led training, plus hands-on labs and opportunities for practical application.

[explore.skillbuilder.aws/learn](https://explore.skillbuilder.aws/learn)



## Validate your expertise

Demonstrate expertise in building, training, tuning, and deploying machine learning models with an industry-recognized credential.

[aws.amazon.com/certification](https://aws.amazon.com/certification)

# Thank you for attending AWS Innovate – AI/ML Edition

We hope you found it interesting! A kind reminder to **complete the survey**.  
Let us know what you thought of today's event and how we can improve the event experience for you in the future.



[aws-apj-marketing@amazon.com](mailto:aws-apj-marketing@amazon.com)



[twitter.com/AWSCloud](https://twitter.com/AWSCloud)



[facebook.com/AmazonWebServices](https://facebook.com/AmazonWebServices)



[youtube.com/user/AmazonWebServices](https://youtube.com/user/AmazonWebServices)



[slideshare.net/AmazonWebServices](https://slideshare.net/AmazonWebServices)



[twitch.tv/aws](https://twitch.tv/aws)

# Thank you!

Alex Thewsey

