Amazon SageMaker

Build, Train and Deploy ML Model

Immersion Day

Michael Lin
Senior Solutions Architect
AWS

In this section – Train, Tune and Deploy

Amazon SageMaker

PREPARE

SageMaker Ground Truth
Label training data for machine learning

SageMaker Data Wrangler
Aggregate and prepare data for machine learning

SageMaker Processing
Built-in Python, BYO R/Spark

SageMaker Feature Store
Store, update, retrieve, and share features

SageMaker Clarify
Detect bias and understand model predictions

BUILD

SageMaker Studio Notebooks
Jupyter notebooks with elastic compute and sharing

Built-in and Bring your-own Algorithms
Dozens of optimized algorithms or bring your own

Local Mode
Test and prototype on your local machine

SageMaker Autopilot
Automatically create machine learning models with full visibility

SageMaker JumpStart
Pre-built solutions for common use cases

TRAIN & TUNE

Managed Training
Distributed infrastructure management

SageMaker Experiments
Capture, organize, and compare every step

Automatic Model Tuning
Hyperparameter optimization

Distributed Training
Training for large datasets and models

SageMaker Debugger
Debug and profile training runs

Managed Spot Training
Reduce training cost by 90%

DEPLOY & MANAGE

Managed Deployment
Fully managed, ultra low latency, high throughput

Kubernetes & Kubeflow Integration
Simplify Kubernetes-based machine learning

Multi-Model Endpoints
Reduce cost by hosting multiple models per instance

SageMaker Model Monitor
Maintain accuracy of deployed models

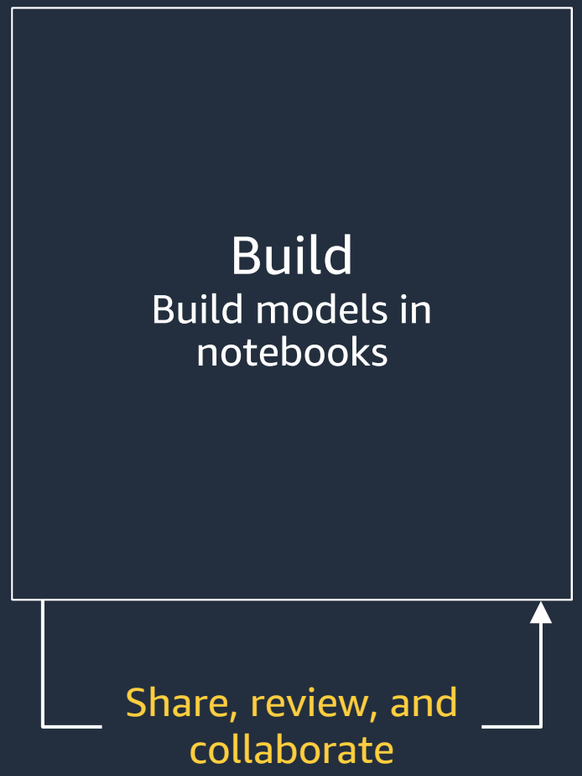
SageMaker Edge Manager
Manage and monitor models on edge devices

SageMaker Pipelines
Workflow orchestration and automation

SageMaker Studio
Integrated development environment (IDE) for ML

aws

Build fast and collaborate



aws

Model options

Training code

AWS Marketplace for Machine Learning

Amazon SageMaker AutoPilot

- XGBoost - Gradient Boosted Trees
- Matrix Factorization
- Regression
- Principal Component Analysis
- K-Means Clustering
- And More!

Built-in Algorithms (17)
No ML coding required

Bring Your Own Script
Amazon SageMaker builds the ccontainer
Open source containers

Bring Your Own Container
Full control
You build the ccontainer
R, C++, etc

Fully Managed, Distributed, Auto-Scaled, Secured

aws

Amazon SageMaker

has built-in algorithms
or bring your own

Computer vision

Image classification | Object detection | Semantic segmentation

Topic modeling

LDA | NTM

Classification

Linear Learner | XGBoost | KNN

Recommendation

Factorization machines

Forecasting

DeepAR

Working with text

BlazingText | Supervised | Unsupervised

Regression

Linear Learner | XGBoost | KNN

Clustering

KMeans

Sequence translation

Seq2Seq

Anomaly detection

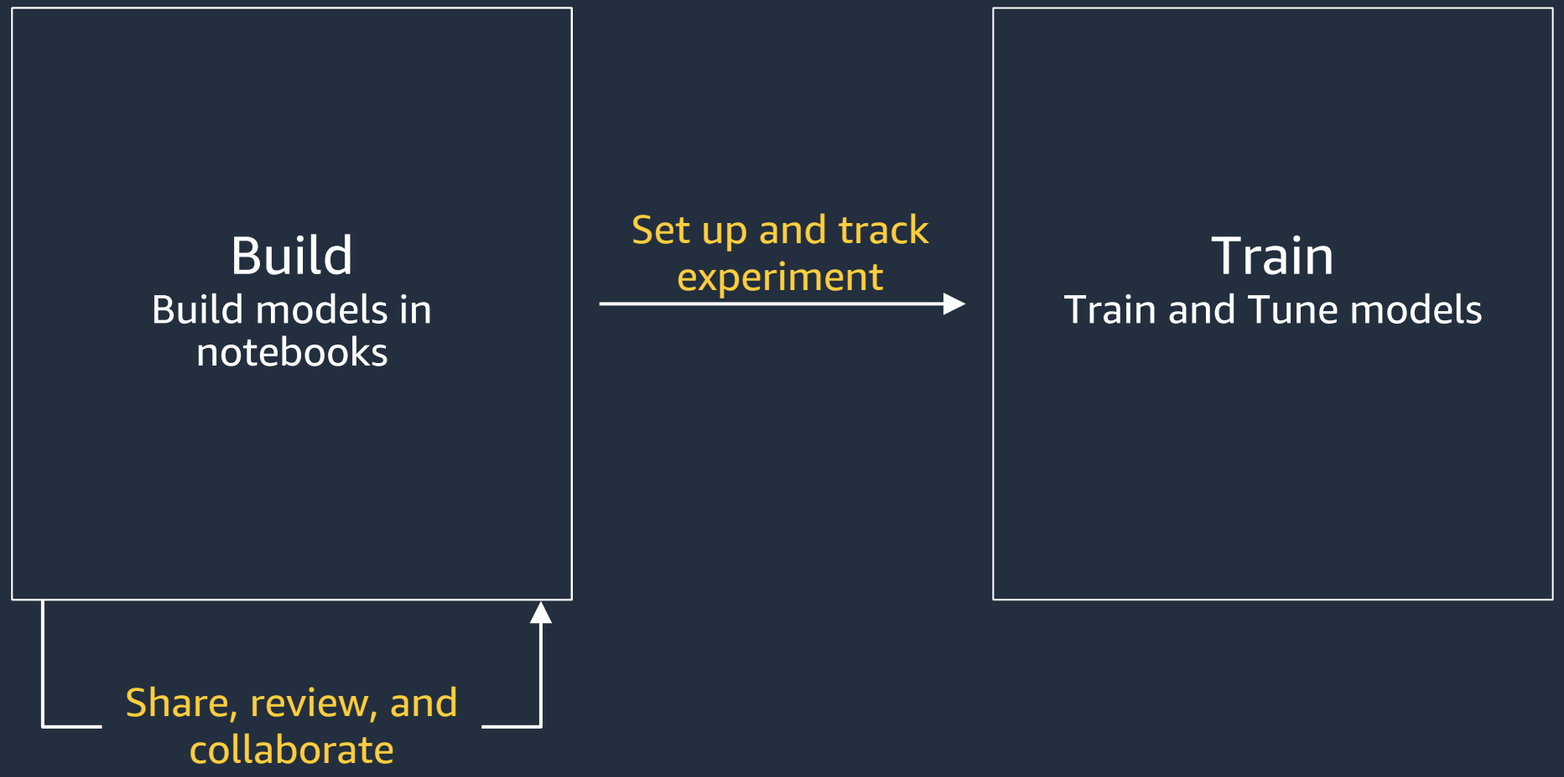
Random cut forests | IP Insights

Feature reduction

PCA

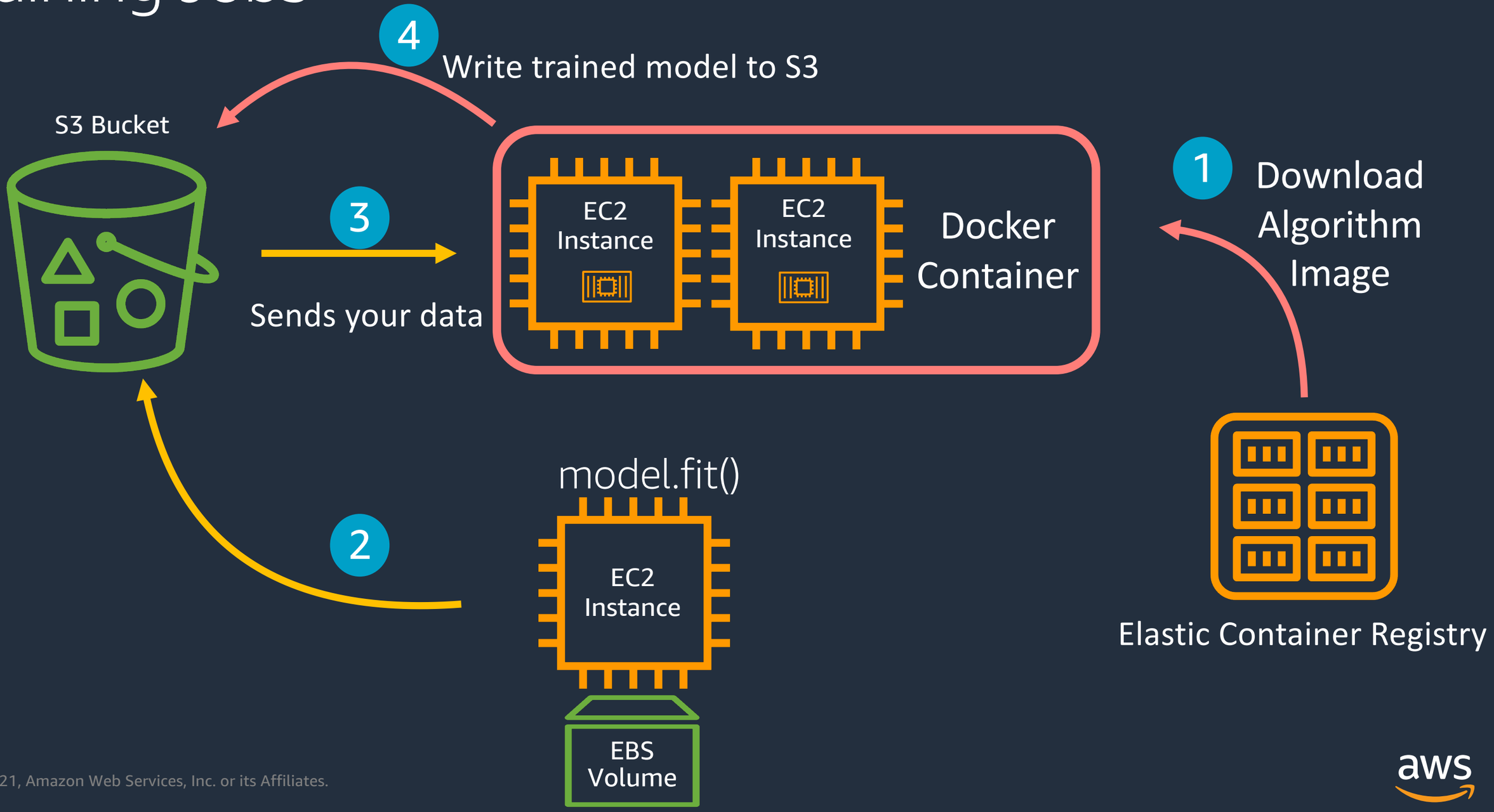
aws

Train



aws

Training Jobs



aws

Pre-built Docker Containers for Built-In Algorithms

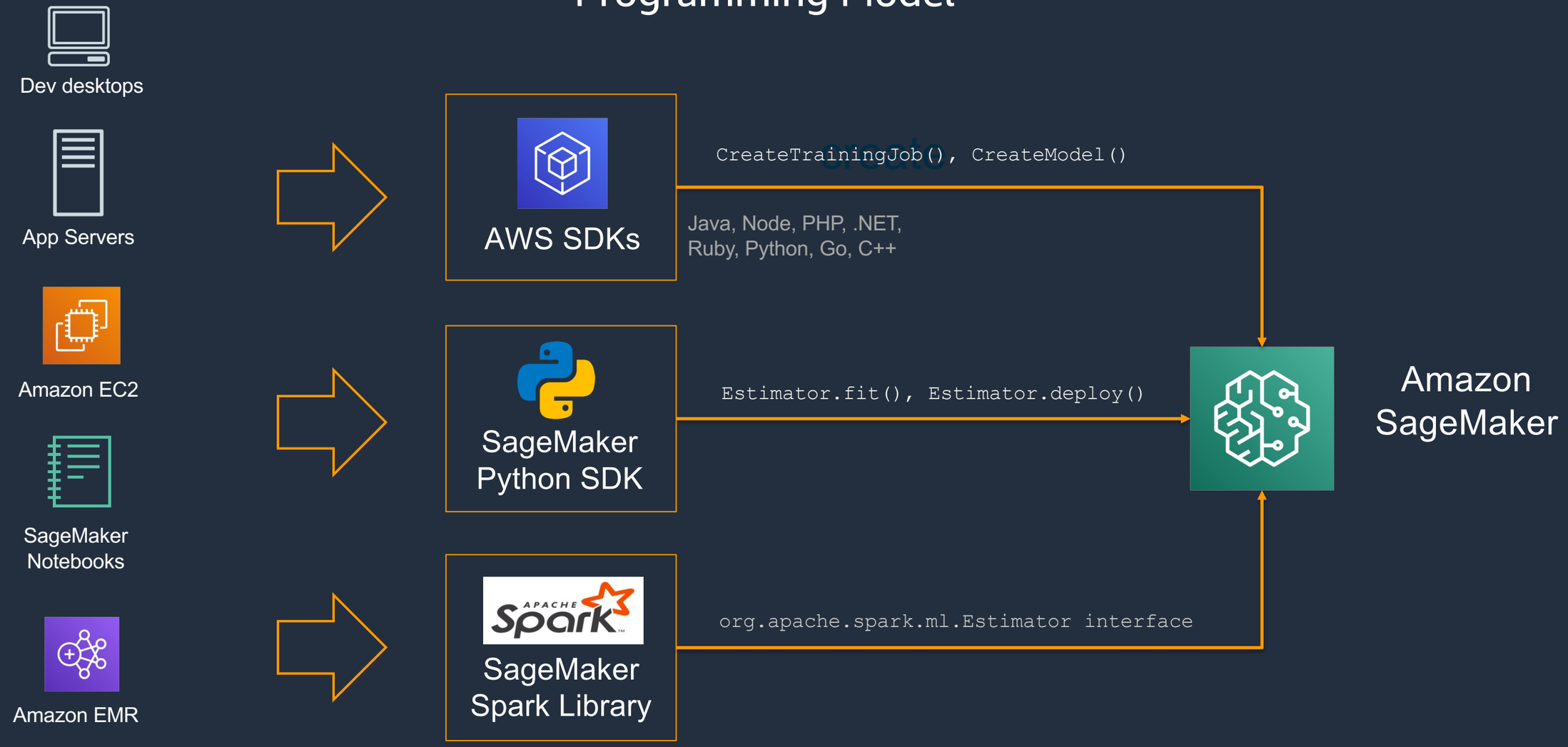

```
container = sagemaker.image_uris.retrieve(region=boto3.Session().region_name, framework='xgboost', version='latest')
print(container)
```

```
685385470294.dkr.ecr.eu-west-1.amazonaws.com/xgboost:latest
```

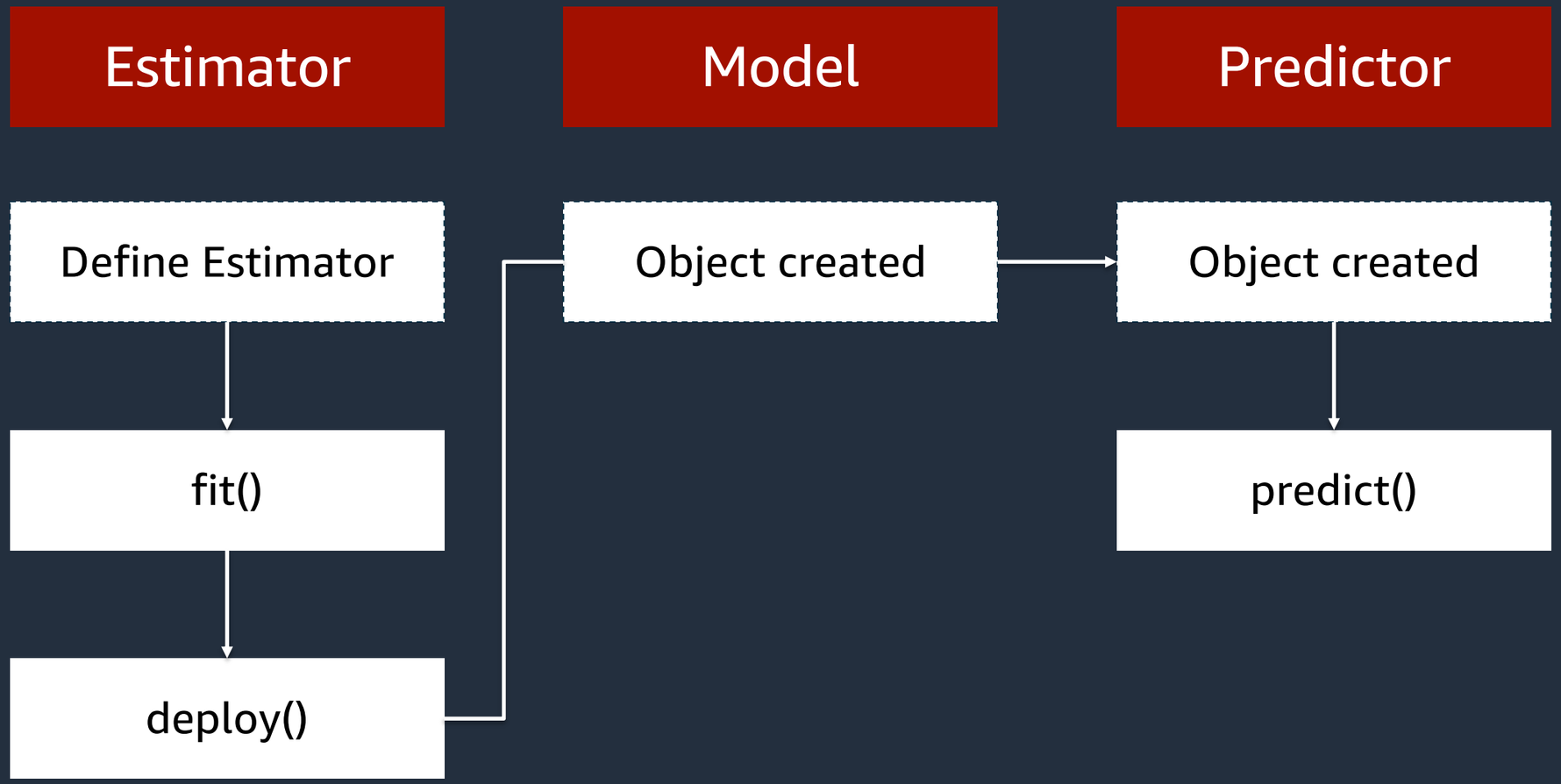

aws

Amazon SageMaker | Training

Programming Model



End-to-End Flow



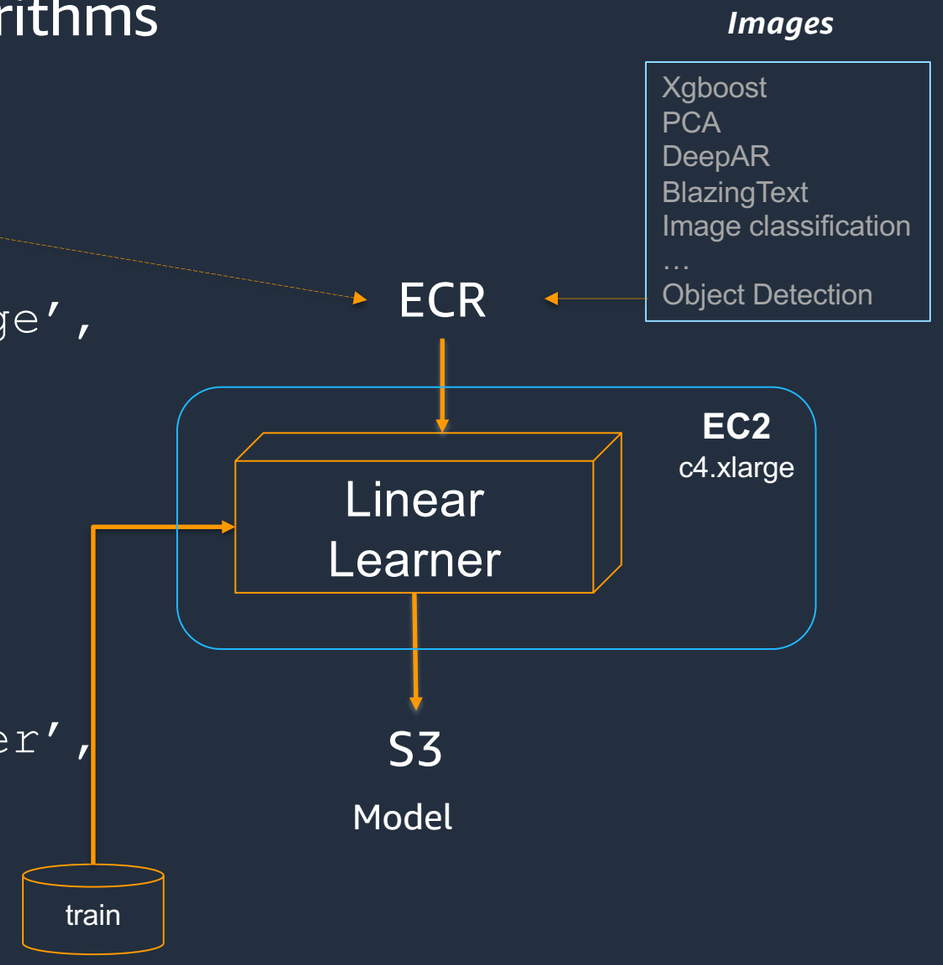
Amazon SageMaker | Training

Use built-in algorithms

```
linear = Estimator('linear-learner',
                   train_instance_count=1,
                   train_instance_type='ml.c4.xlarge',
                   output_path=output_location,
                   sagemaker_session=sess)
```

```
linear.set_hyperparameters(
                   feature_dim=784,
                   predictor_type='binary_classifier',
                   mini_batch_size=200)
```

```
linear.fit({'train': s3_train_data})
```



aws

```
In [ ]: bt_model = sagemaker.estimator.Estimator(container,
                                                 role,
                                                 train_instance_count=1,
                                                 train_instance_type='ml.c4.4xlarge',
                                                 train_volume_size = 30,
                                                 train_max_run = 360000,
                                                 input_mode= 'File',
                                                 output_path=s3_output_location,
                                                 sagemaker_session=sess)
```

Number of EC2 instances

Disk space

Type of EC2 instances

aws

Algorithm Container

```
In [ ]: bt_model = sagemaker.estimator.Estimator(container,
                                                 role,
                                                 train_instance_count=1,
                                                 train_instance_type='ml.c4.4xlarge',
                                                 train_volume_size = 30,
                                                 train_max_run = 360000,
                                                 input_mode= 'File',
                                                 output_path=s3_output_location,
                                                 sagemaker_session=sess)
```

SageMaker Estimator

Execution Role

```
In [ ]: bt_model.fit(inputs=data_channels, logs=True)
```

Cluster comes online

Logs to CloudWatch

Monitor via console or notification stream

aws

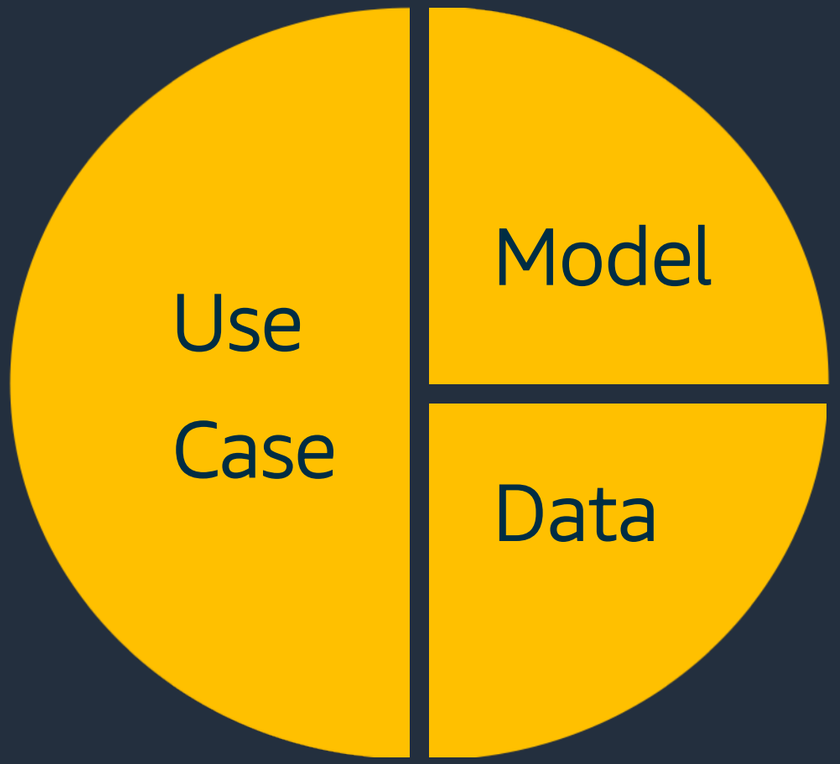
Every model run on a SageMaker training job has its own ephemeral cluster.

That means you have a dedicated EC2 instance alive for the number of seconds your model needs to train.

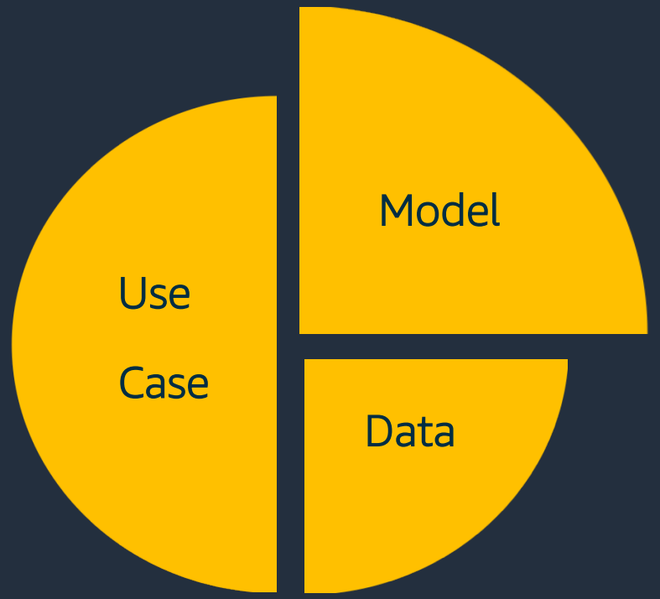
This cluster comes down immediately after the model finished training.

aws

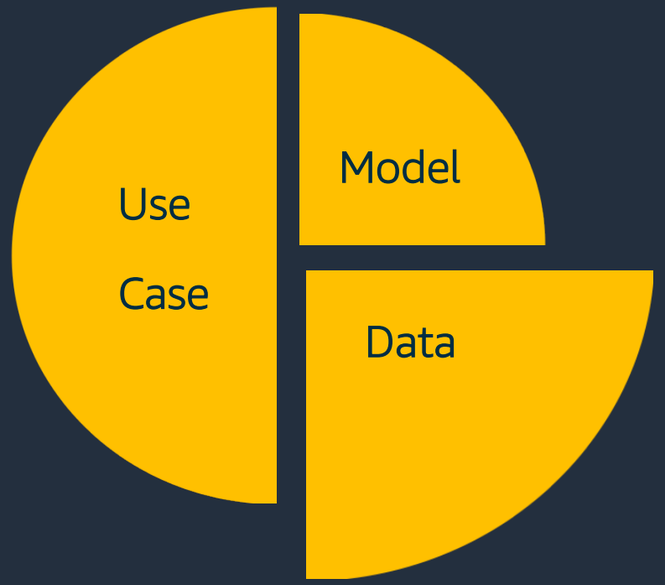
Learning Theory Fundamentals



Overfitting

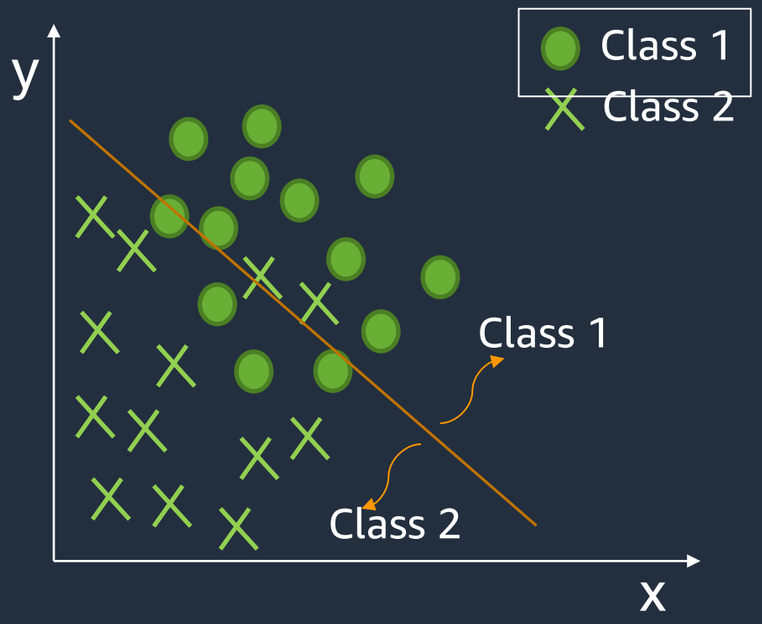


Underfitting



aws

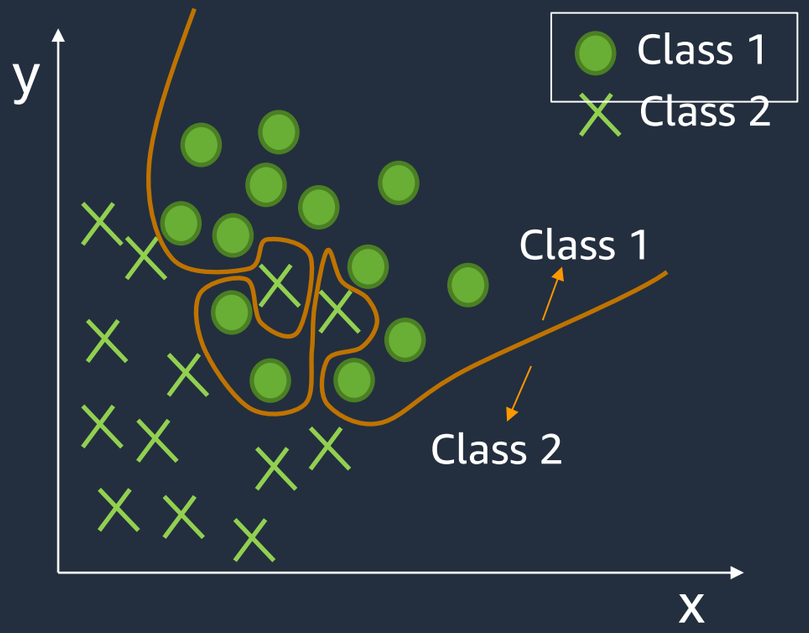
Underfitting



The model is too simple to capture the input/output relationship.

It will have poor training and test performance.

Overfitting

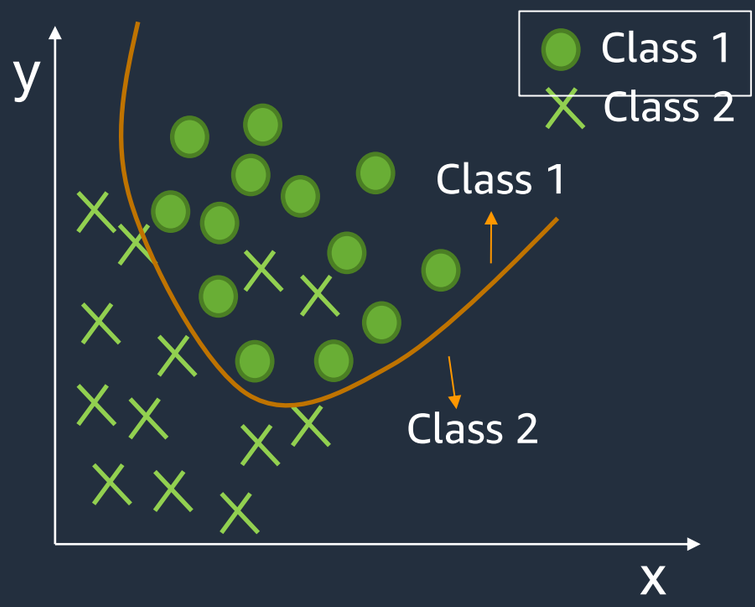


Having too complex models for simple problems can cause overfitting.

The model picks up the noise instead of the underlying relationship.

We will see good scores in training data but poor in test data.

Appropriate Fitting



No bias, near right predication

aws

Amazon SageMaker Automatic Model Tuning

Hyperparameter Optimizer

Decision Trees	Neural Networks
Tree depth	Number of layers
Max leaf nodes	Hidden layer width
Gamma	Learning rate
Eta	Embedding
Lambda	dimensions
Alpha	Dropout
…	…

…

"Hyperparameters"

(algorithm parameters that significantly affect model quality)

aws

Amazon SageMaker Automatic Model Tuning

Automatically tune hyperparameters in your algorithms

Tuning at scale
Adjust thousands of different combinations of algorithm parameters

Automated
Uses ML to find the best parameters

Faster
Eliminate days or weeks of tedious manual work

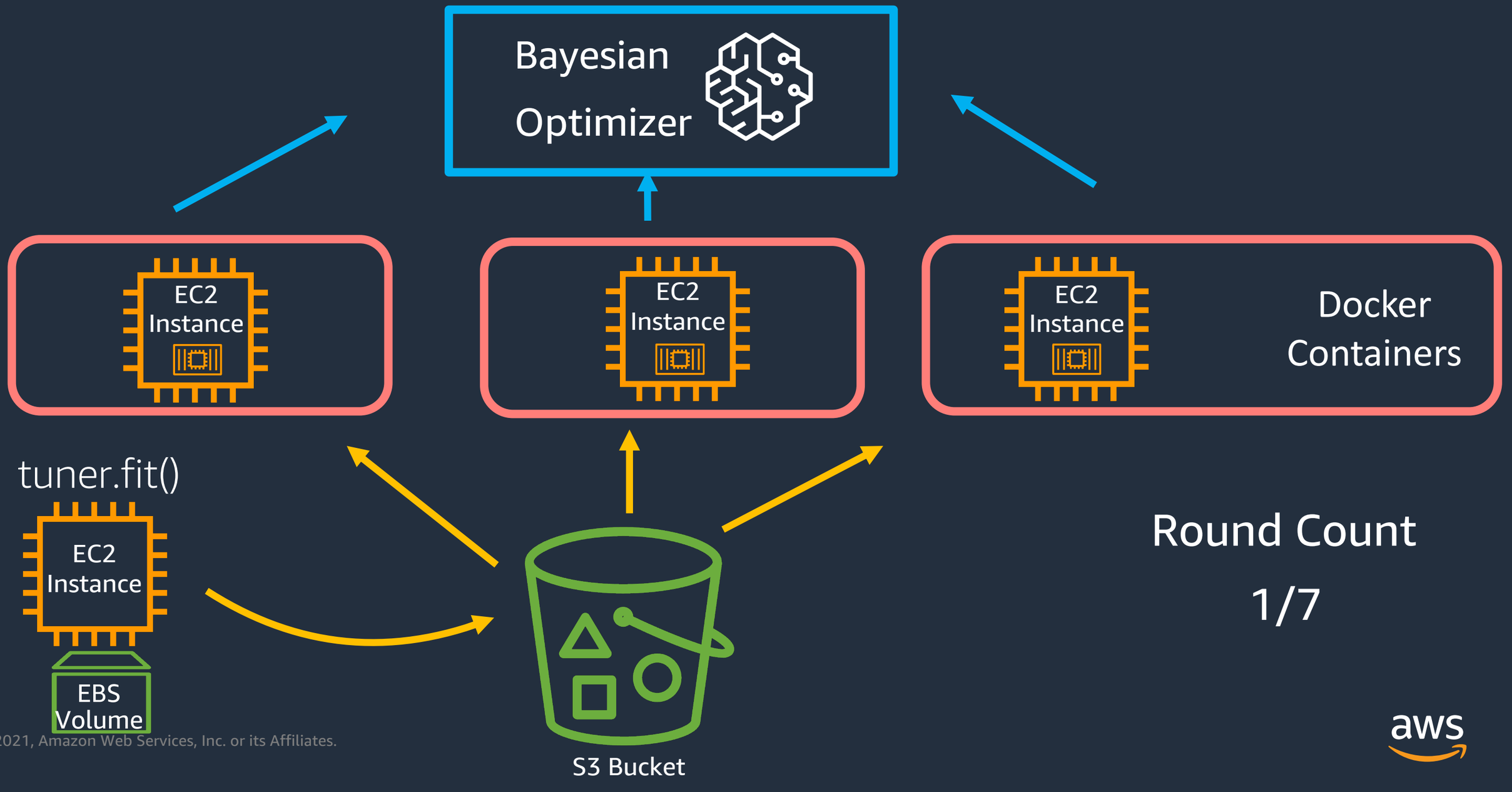
EXAMPLES

Decision trees
Tree depth | Max leaf nodes | Gamma | Eta | Lambda | Alpha

Neural networks
Number of layers | Hidden layer width | Learning rate | Embedding dimensions | Dropout

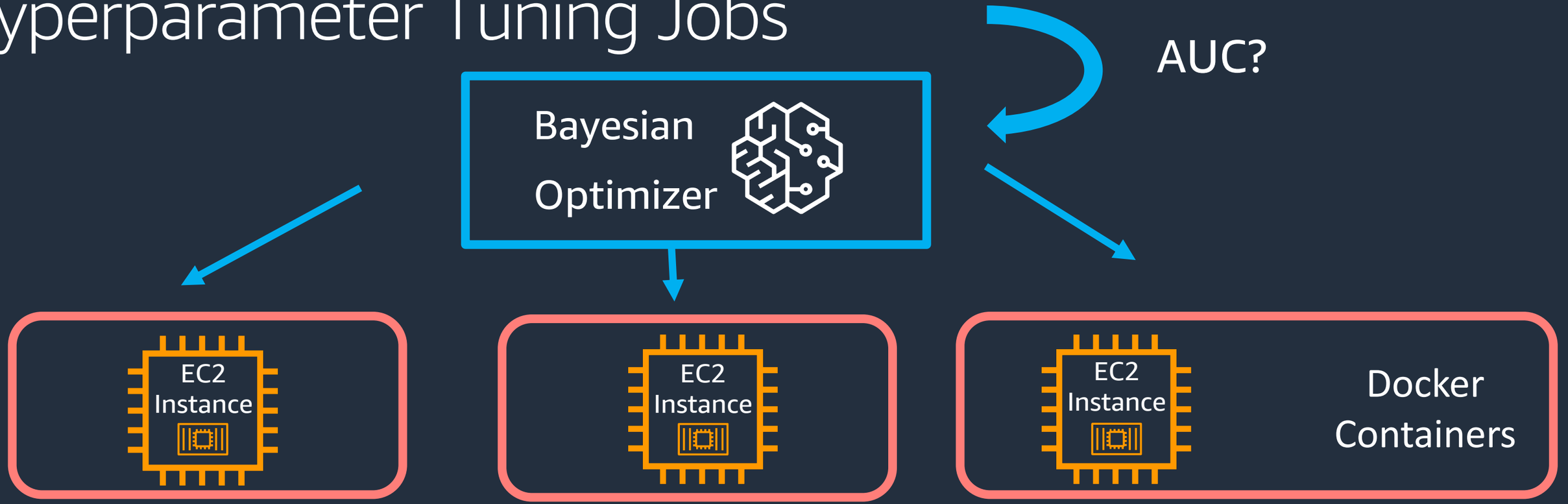
aws

Hyperparameter Tuning Jobs



aws

Hyperparameter Tuning Jobs



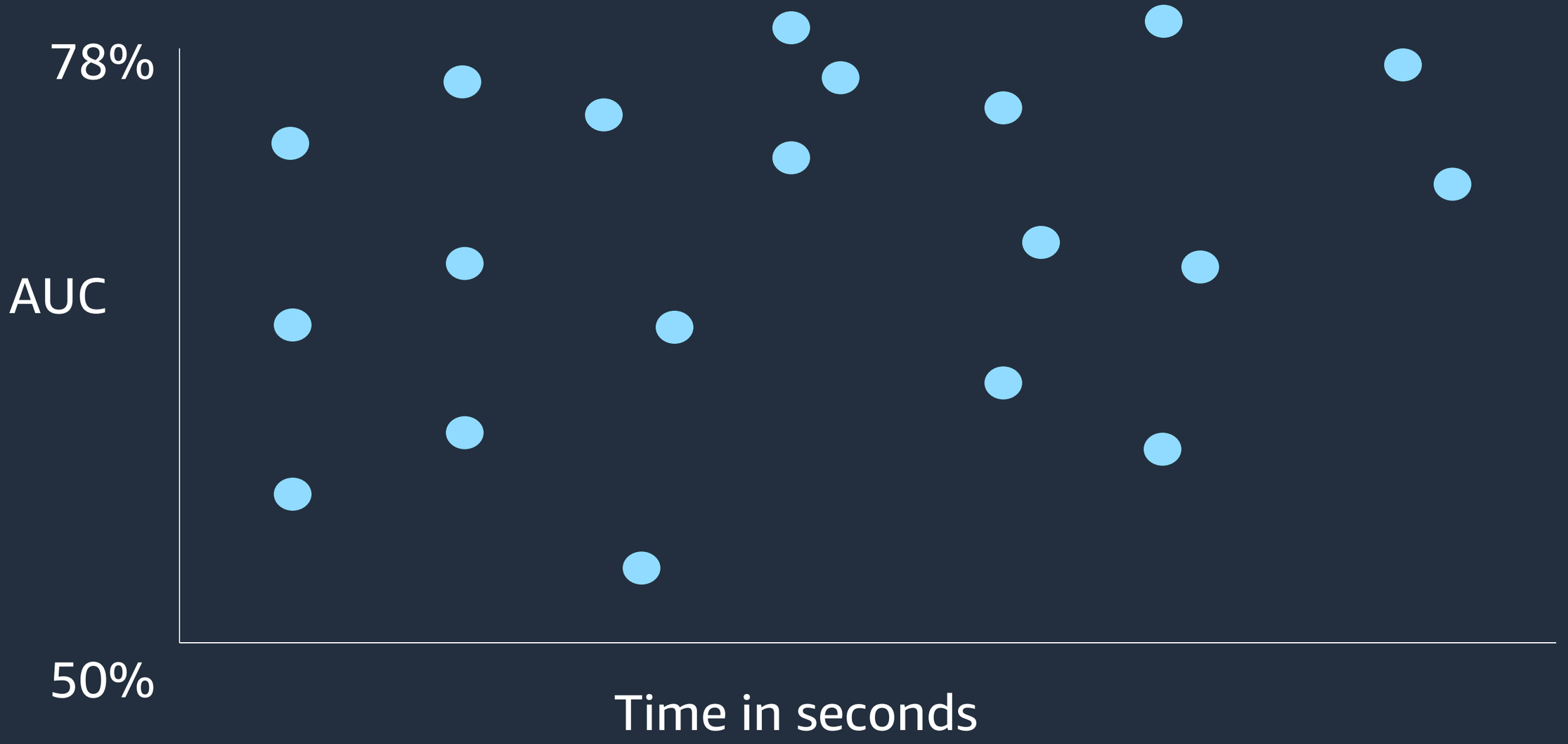
Different Hyperparameter Settings

Round Count

2/7

aws

Hyperparameter Tuning Jobs



aws

How do I set up a hyperparameter tuning job?

Objective Metric

Hyperparameters
& Ranges

Job Specs

aws

Can I use hyperparameter tuning with my own model?

Yes!!!

Built-in Algorithms

Docker

Script Mode

Fully Customizable

aws

1. Pick hyperparameters and ranges

```
hyperparameter_ranges = {'eta': ContinuousParameter(0, 1),
                         'min_child_weight': ContinuousParameter(1, 10),
                         'alpha': ContinuousParameter(0, 2),
                         'max_depth': IntegerParameter(1, 10)}
```

2. Pick objective metric

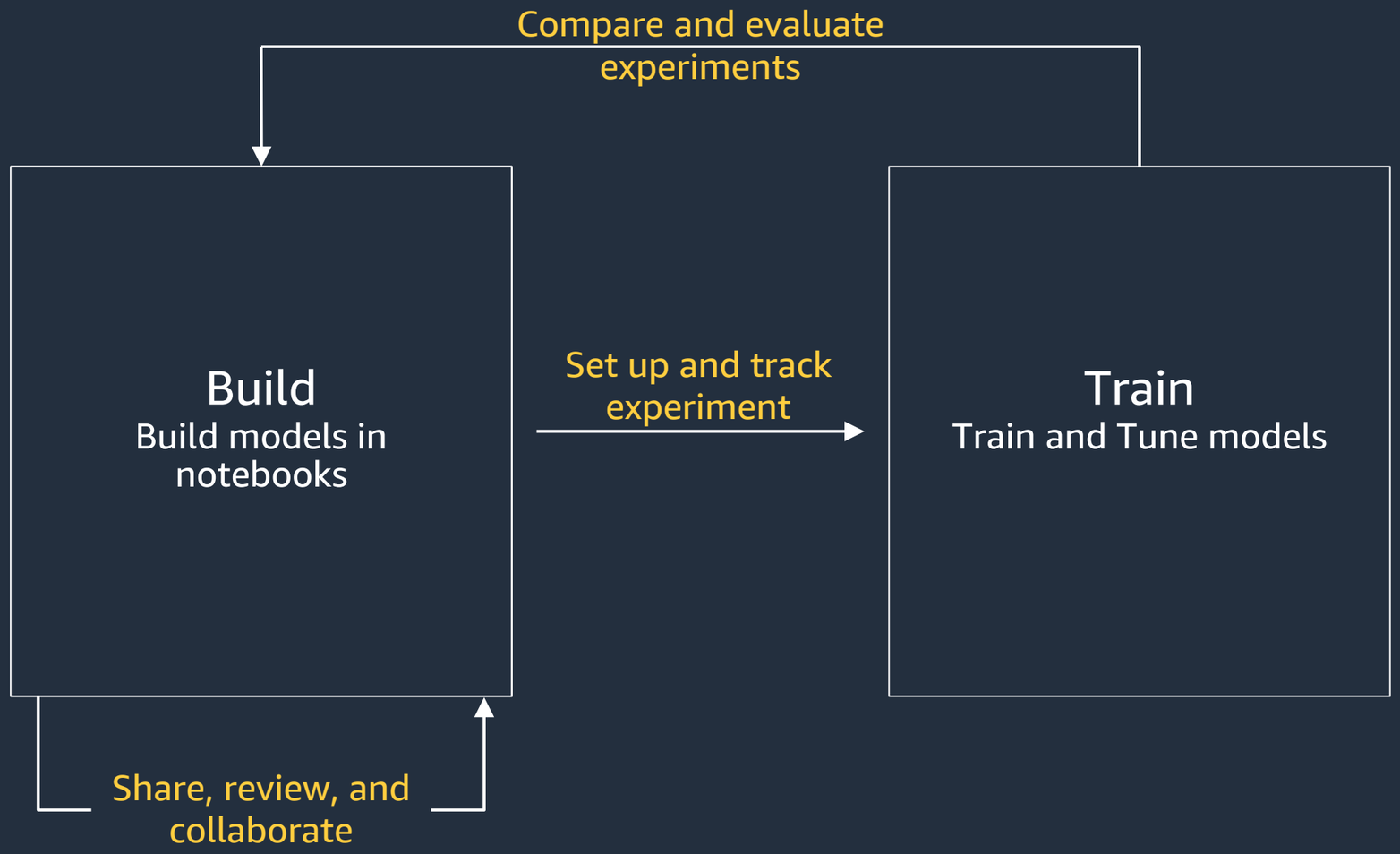
```
objective_metric_name = 'validation:auc'
```

3. Pick job parameters

```
tuner = HyperparameterTuner(xgb,
                            objective_metric_name,
                            hyperparameter_ranges,
                            max_jobs=20,
                            max_parallel_jobs=3)
```


aws

Manage Experiments



aws

Amazon SageMaker Experiments

Organize, track, and compare training experiments

Tracking at scale
Track parameters and metrics across experiments and users

Custom organization
Organize experiments by teams, goals, and hypotheses

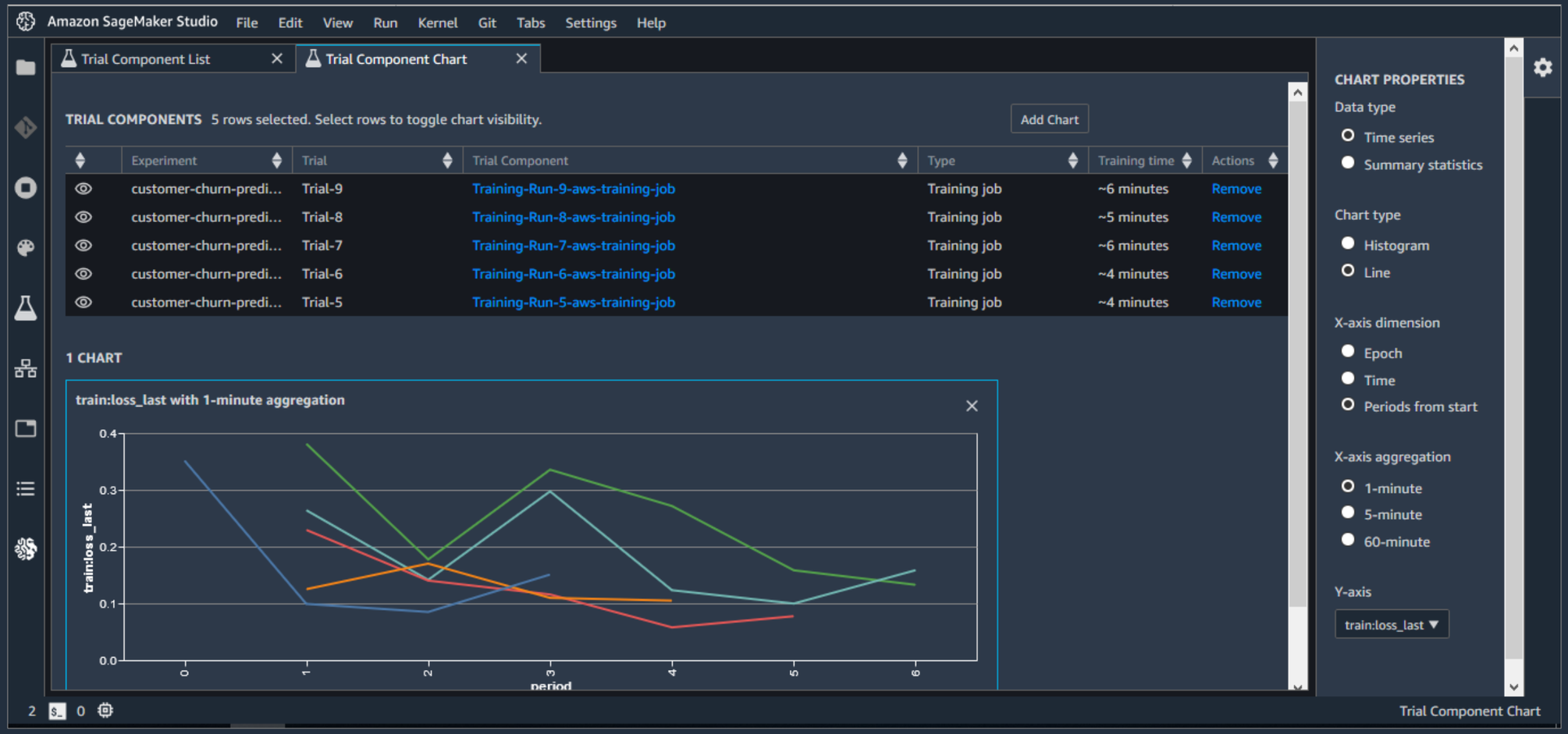
Visualization
Easily visualize experiments and compare

Metrics and logging
Log custom metrics using the Python SDK and APIs

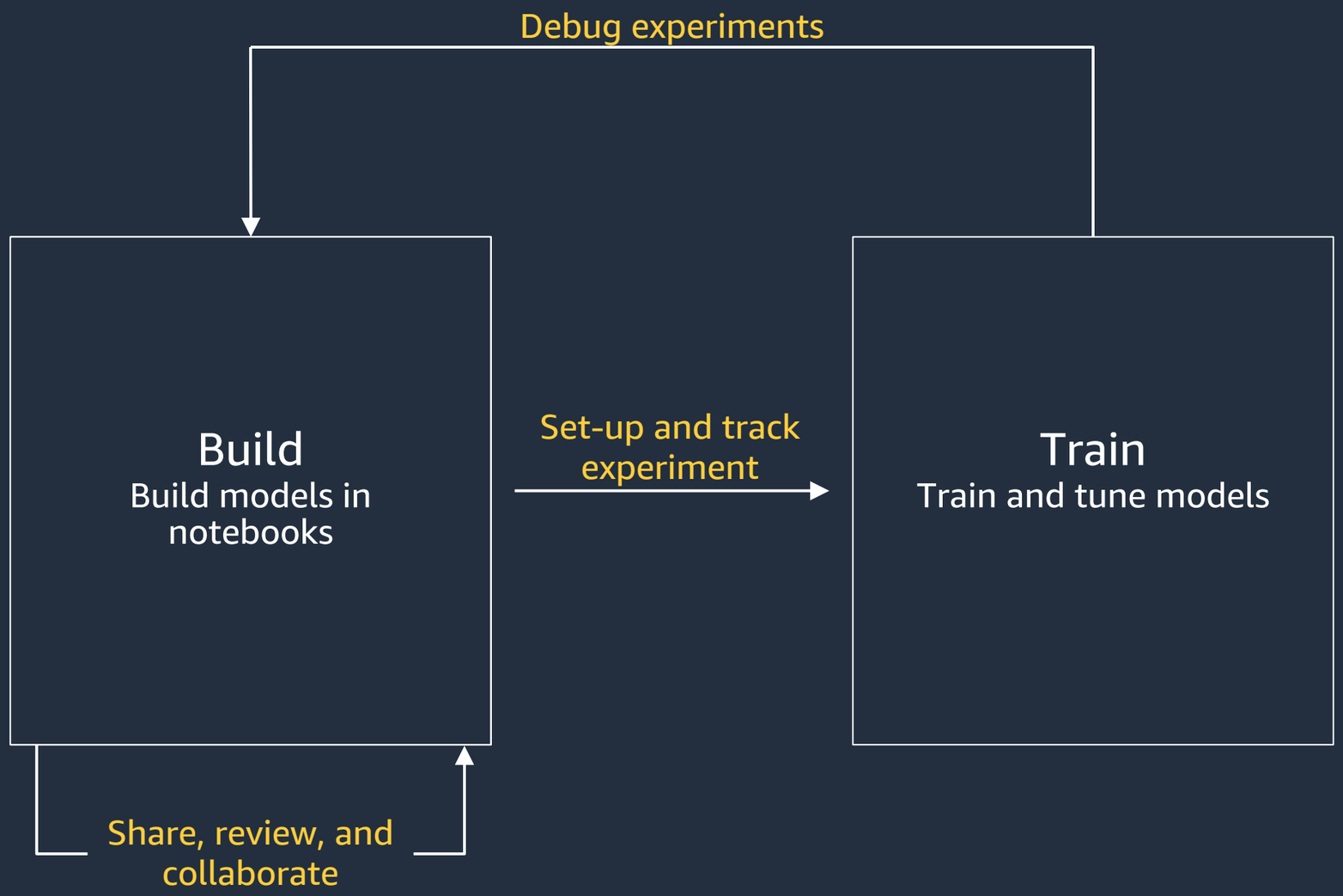
Fast iteration
Quickly go back and forth, and maintain high-quality

aws

Use Amazon SageMaker Experiments to track and manage thousands of experiments



Debug training runs



aws

Amazon SageMaker Debugger

Detect bottlenecks and training problems in real-time, and train models faster

Generate ML models faster

Detect bottlenecks and issues during training in real-time and correct problems to deploy models faster, with a single, unified tool

Optimize resources with no additional code

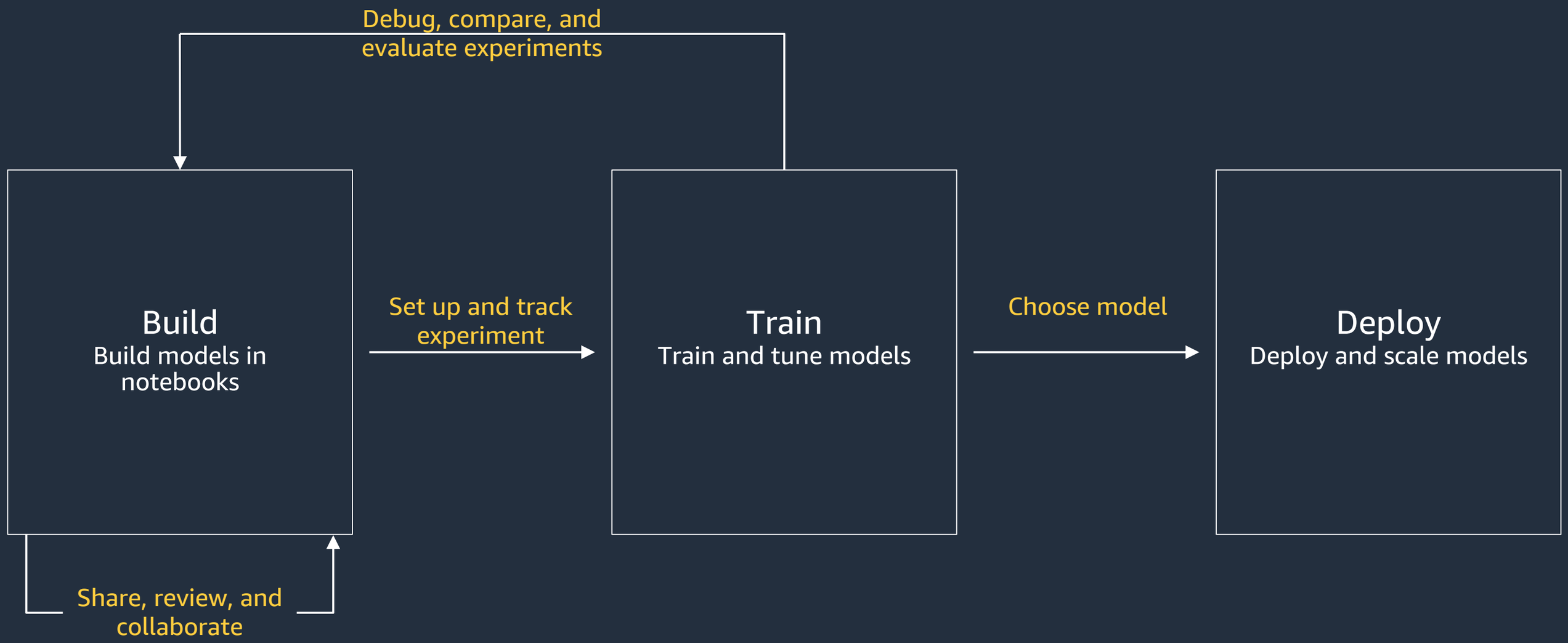
Monitor and profile system resources without code and get recommendations to optimize resources effectively

Make ML training transparent

Get complete insights into the ML training process in real-time and offline

aws

Deploy models



aws

Deployment options

Model in Amazon S3

Amazon SageMaker Real-time endpoint

1 line of code
Vanilla HTTPS
Post data, get a prediction
Any tool, any language
Auto Scaling available

Amazon SageMaker batch transform

1 line of code
Predict data stored in S3
Read results from S3

Amazon container services (ECS, EKS, Fargate)

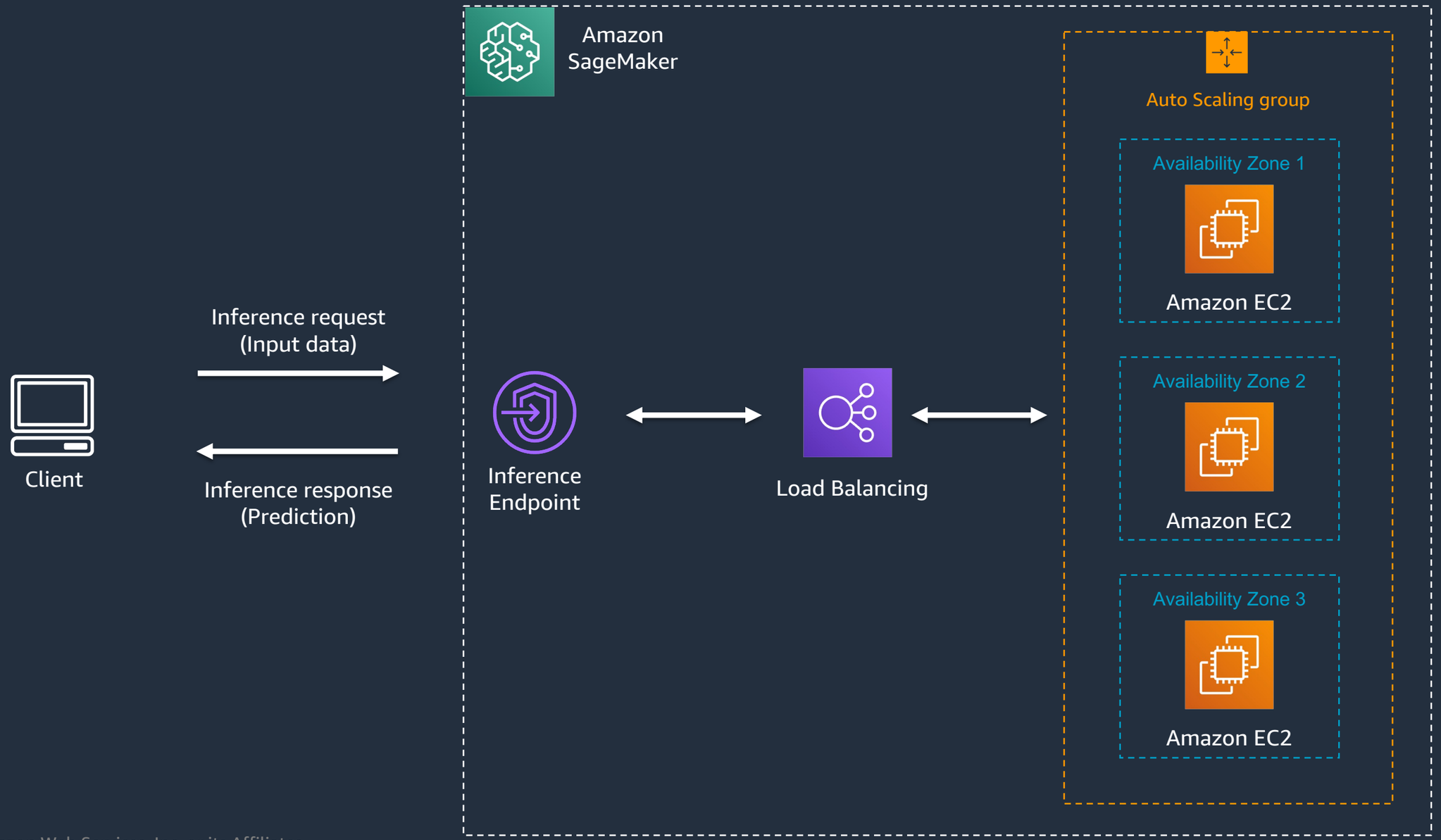
Use AWS Deep Learning containers
Use your own container

Anywhere you like

Grab the model in S3 and run

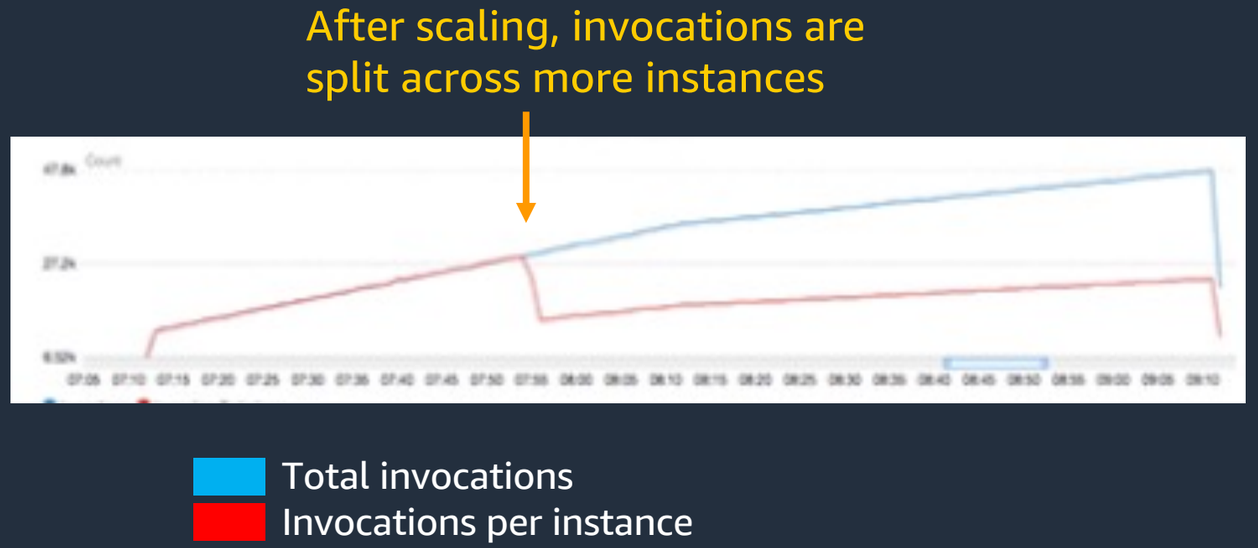
aws

Amazon SageMaker Real-time endpoint

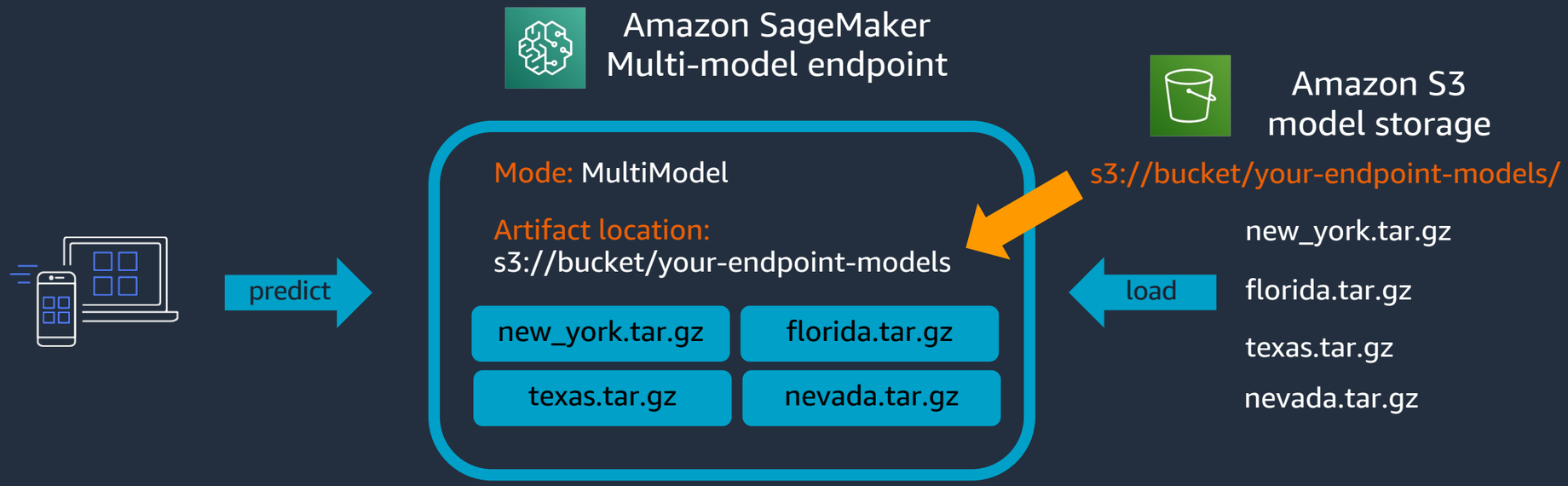


Autoscaling for real-time inference endpoints

- 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



Multi-Model Endpoints



Key Capabilities

- Deploy tens to tens of thousands of models; target model for each inference request
- Two modes for model caching: Caching Enabled, Caching Disabled
- Support for built in algorithms, SageMaker frameworks, and custom models
- Support for inference pipelines and condition keys to restrict access to models

aws

Multi-model endpoints

Significant savings for large-scale deployments

10 separate endpoints
$3,430/month

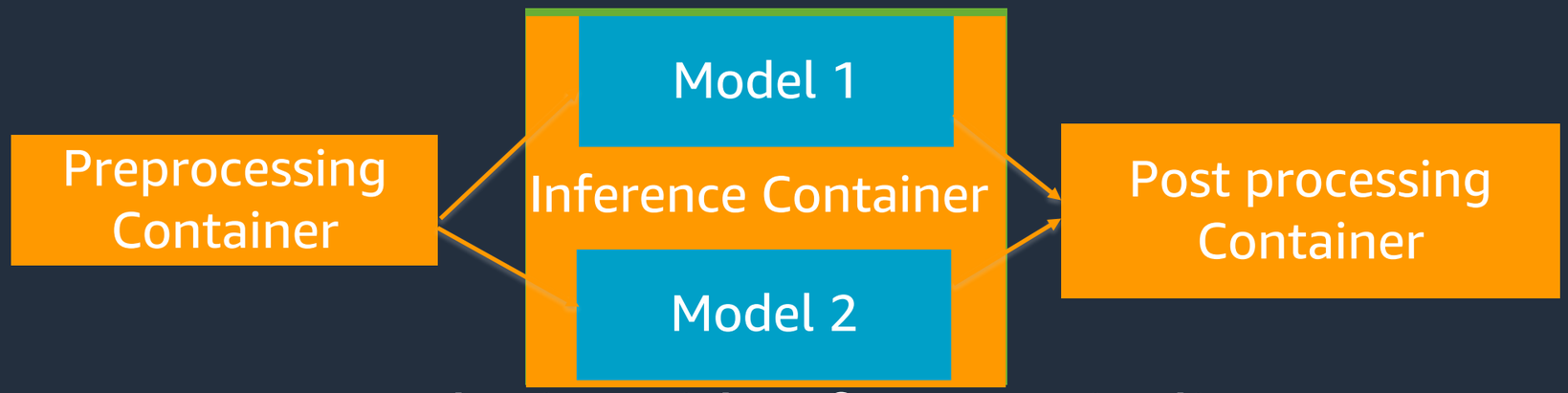
1 multi-model endpoint
$343/month

Sample scenario: ml.c5.xlarge, $0.238/hour, 2 instances running 24/7

aws

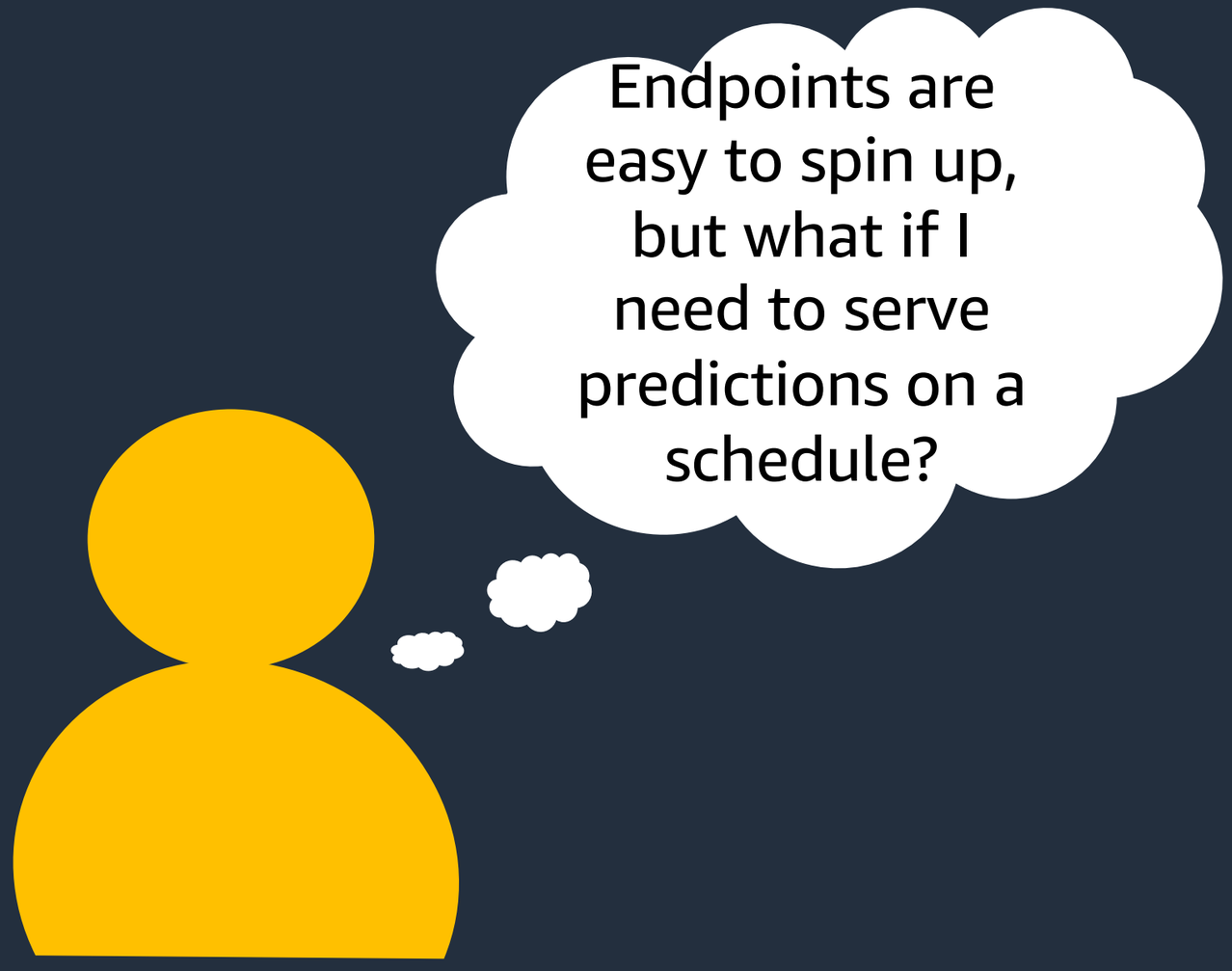
Inference Pipeline

- Amazon SageMaker model composed of a linear sequence of 2 to 5 containers
- Deploy any combination of pretrained SageMaker built-in algorithms and custom algorithms packaged in Docker containers
- Combine preprocessing, predictions, and post-processing data science tasks.
- Fully managed sequence of HTTP requests



Endpoint with Inference Pipeline

aws

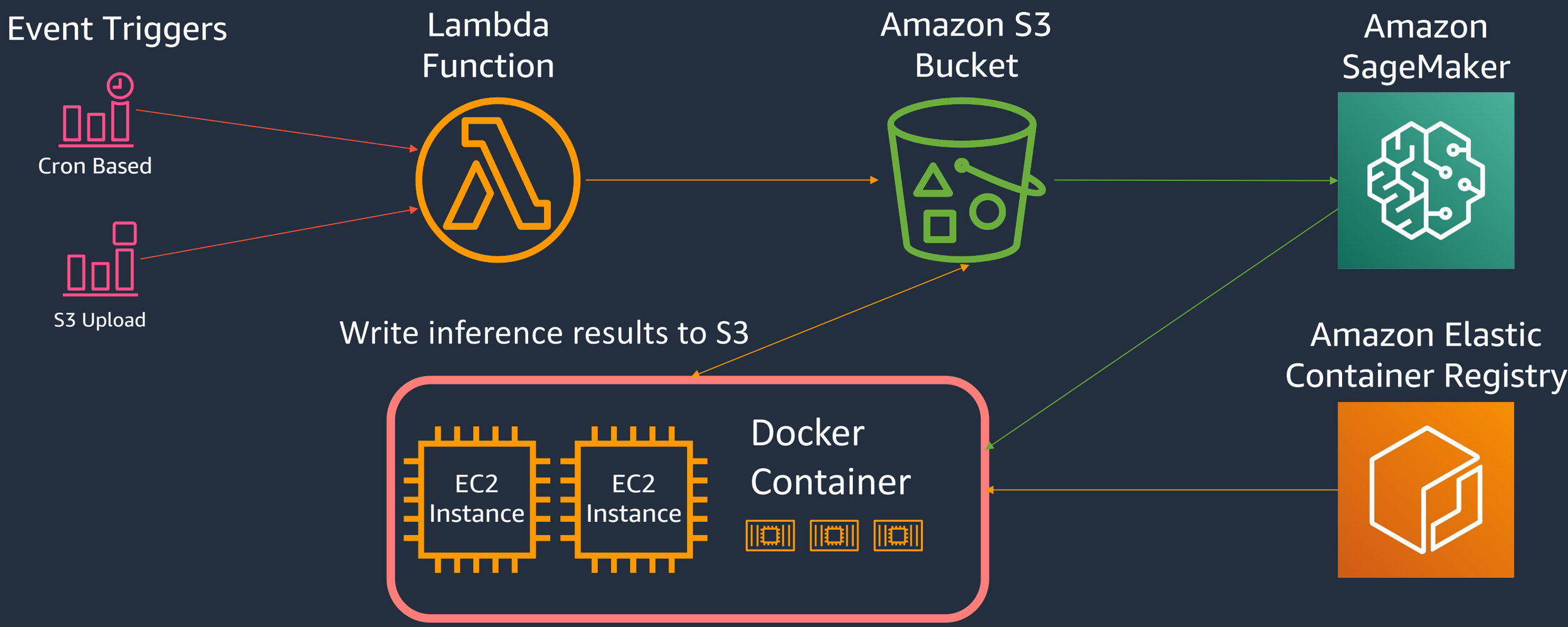


Customer

Cloud Architect

aws

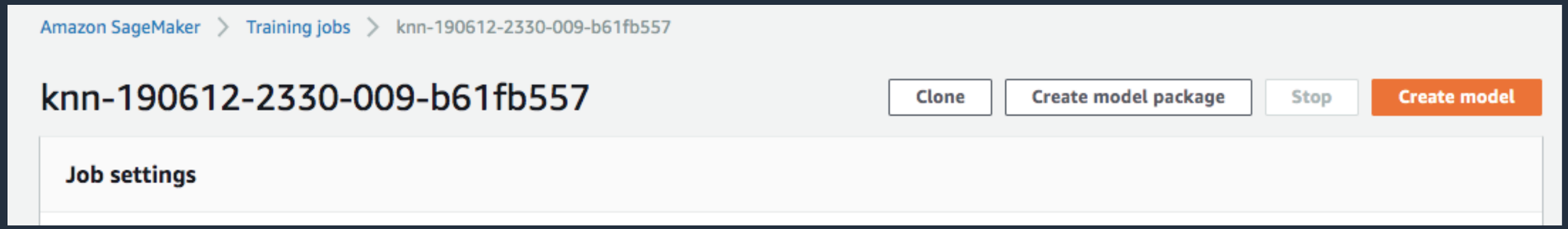
Batch Transform Common Design Pattern



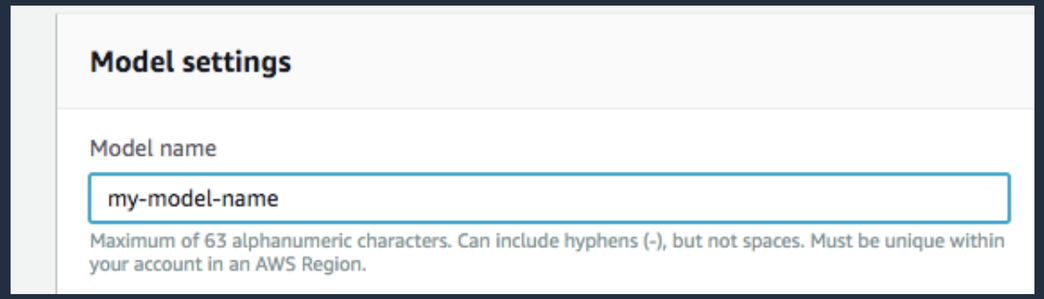
aws

SageMaker Batch Transform

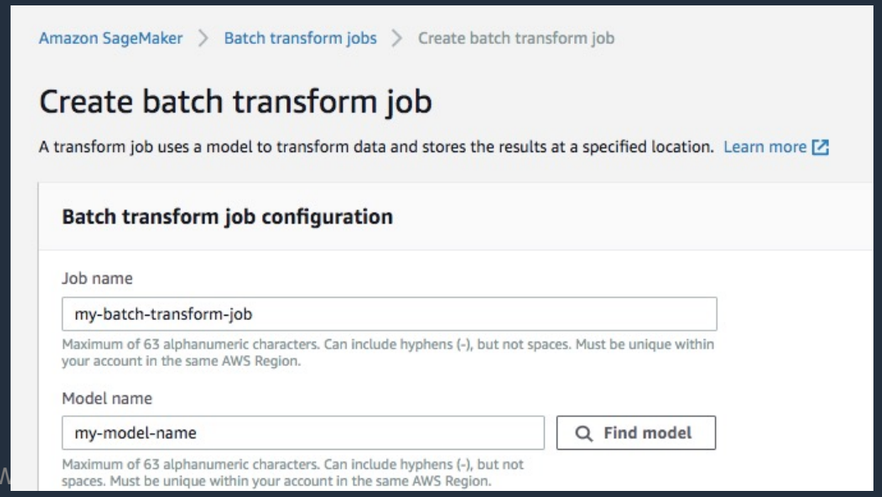
1.



2.



3.



The cluster will spin down immediately after the job is finished.

aws

Choosing Your Deployment Option

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

aws

SageMaker Serverless Inference (preview)

Deploying ML models using SageMaker Serverless Inference (Preview)

by Ram Vegiraju, Michael Pham, Rishabh Ray Chaudhury, and Shelbee Eigenbrode | on 05 JAN 2022 | in Amazon Machine Learning, Amazon SageMaker, Artificial Intelligence | Permalink | Comments | Share

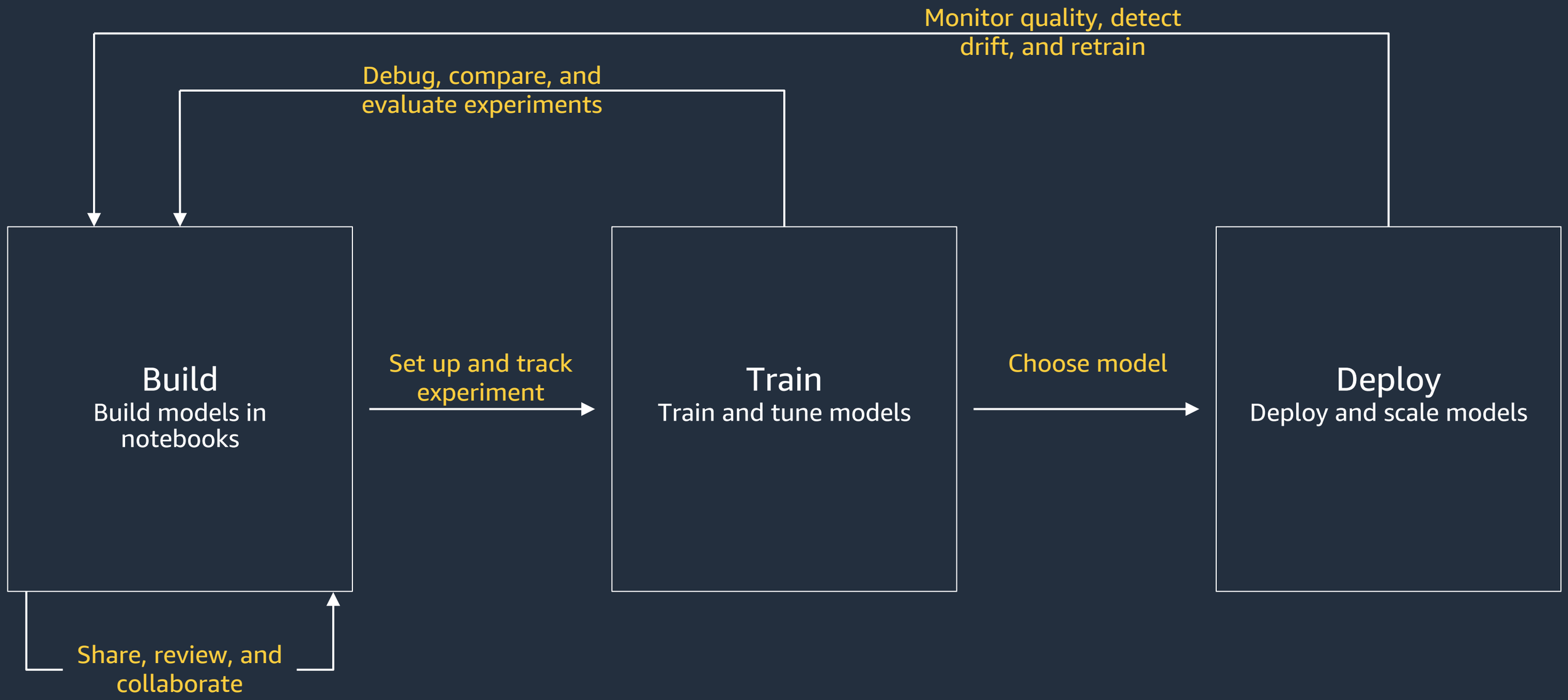
Amazon SageMaker Serverless Inference (Preview) was recently announced at re:Invent 2021 as a new model hosting feature that lets customers serve model predictions without having to explicitly provision compute instances or configure scaling policies to handle traffic variations. Serverless Inference is a new deployment capability that complements SageMaker's existing options for deployment that include: SageMaker Real-Time Inference for workloads with low latency requirements in the order of milliseconds, SageMaker Batch Transform to run predictions on batches of data, and SageMaker Asynchronous Inference for inferences with large payload sizes or requiring long processing times.

Serverless Inference means that you don't need to configure and manage the underlying infrastructure hosting your models. When you host your model on a Serverless Inference endpoint, simply select the memory and max concurrent invocations. Then, SageMaker will automatically provision, scale, and terminate compute capacity based on the inference request volume. SageMaker Serverless Inference also means that you only pay for the duration of running the inference code and the amount of data processed, not for idle time. Moreover, you can scale to zero to optimize your inference costs.

Source: https://aws.amazon.com/blogs/machine-learning/deploying-ml-models-using-sagemaker-serverless-inference-preview/

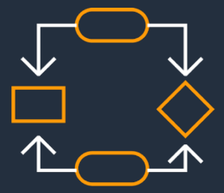
aws

Monitor models



aws

Amazon SageMaker Model Monitor



Collect data

Periodically collects data from endpoints into Amazon S3

Analyze

Computes feature statistics

Monitor

Monitors trends in data and detects drifts

Alerts

Alerts on Amazon CloudWatch and Amazon SageMaker Studio

aws

Thank you

Michael Lin
linmicht@amazon.com

aws