Amazon SageMaker Train, Tune and Deploy

In this session: Train, Tune and Deploy models

Prepare

Web-based IDE for ML

Automatically build and train models

Fully managed data processing jobs and data labeling workflows

10101101 0
01010101 0
00001111 0

Collect and prepare training data

Build

One-click collaborative notebooks and built-in, high performance algorithms and models

Choose or build an ML algorithm

Train & Tune

One-click training

Set up and manage environments for training

Debugging and optimization

Train, debug, and tune models

Visually track and compare experiments

Manage training runs

Deploy & Manage

One-click deployment and auto-scaling

Deploy model in production

Automatically spot concept drift

Monitor models

Add human review of predictions

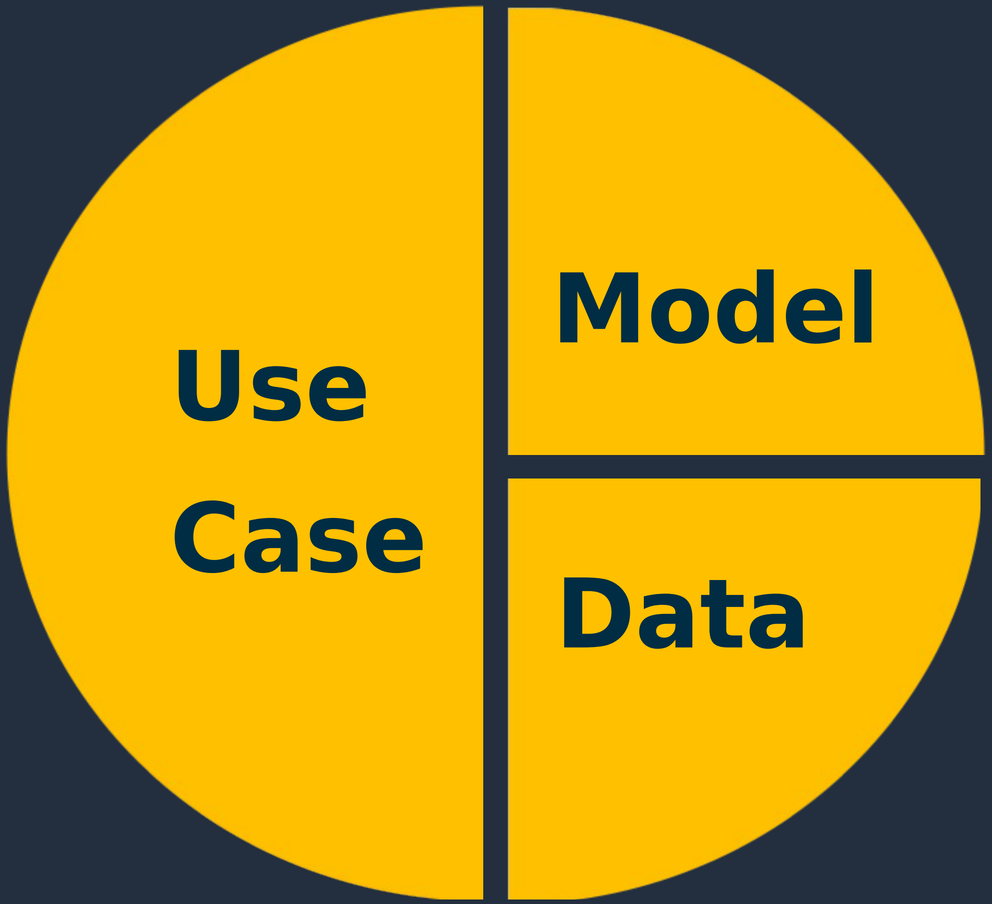
Validate predictions

Fully managed with auto-scaling for 75% less

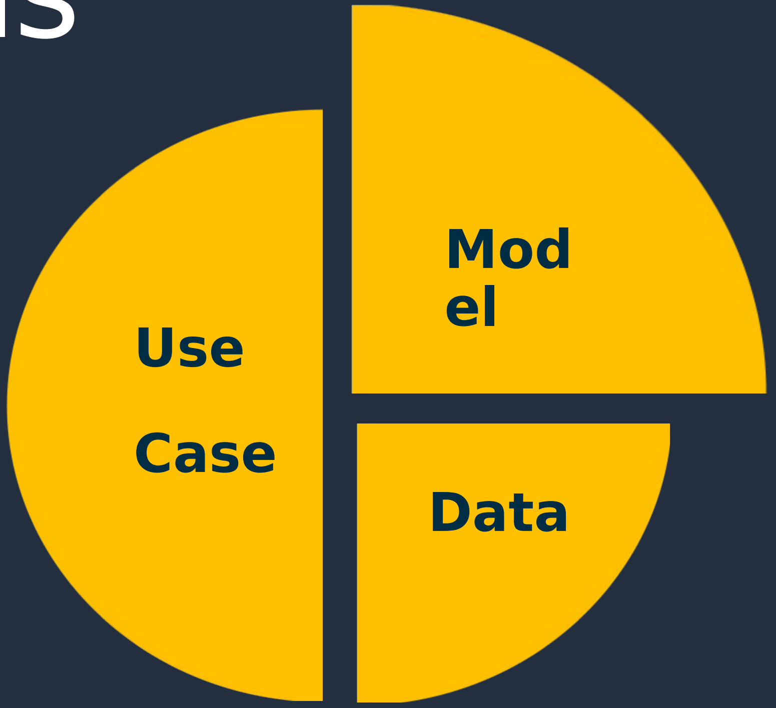
Scale & manage the production environment

Train ML Model in SageMaker

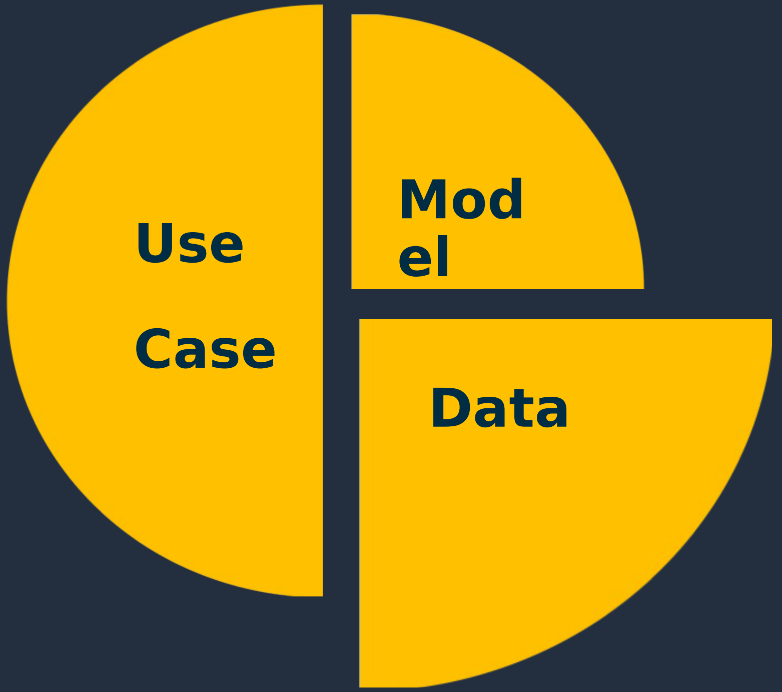
Learning Theory Fundamentals



Overfitting

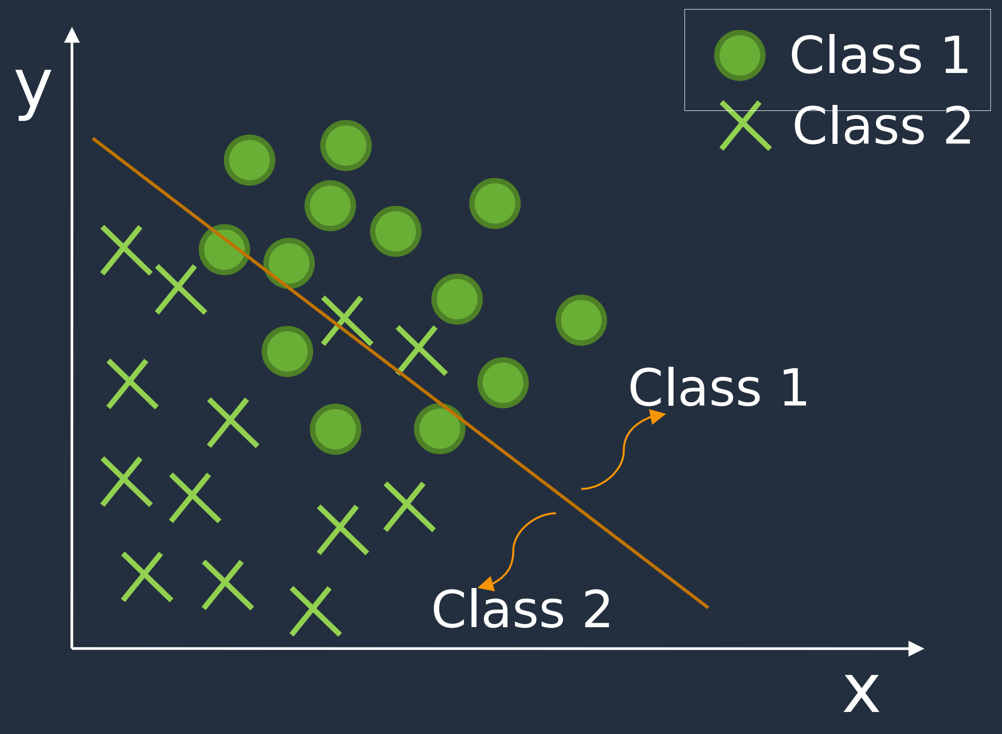


Underfitting



Underfitting

Underfitting: The model is not good enough to describe the relationship between the input data (x) and output (y).

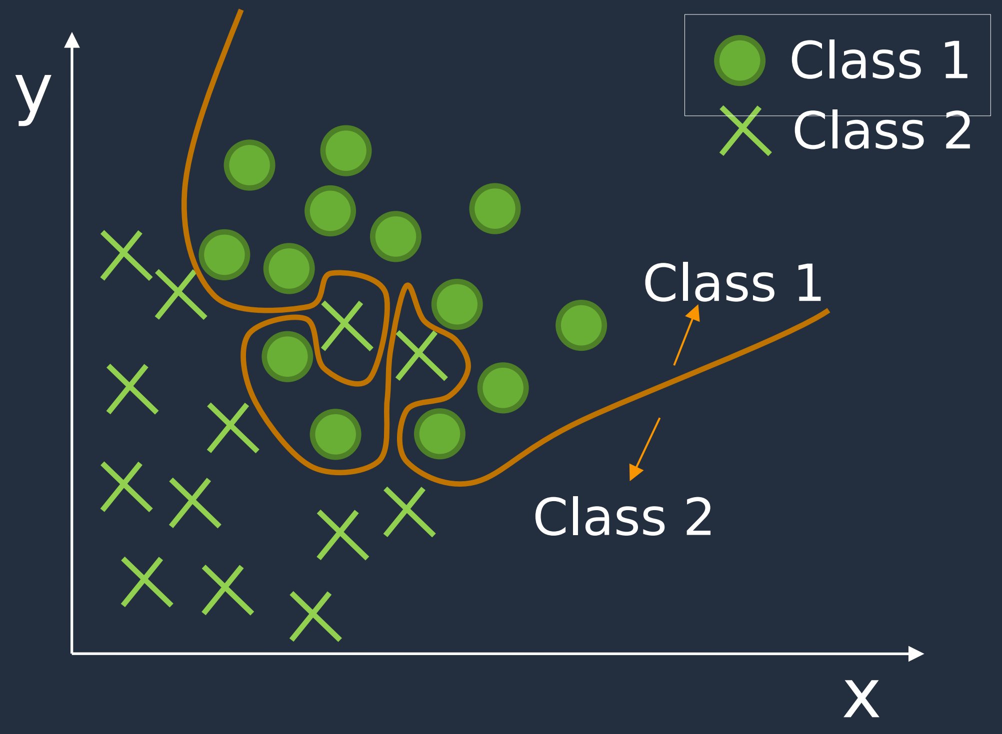


The model is too simple to capture the input/output relationship.

It will have poor training and test performance.

Overfitting

Overfitting: Model memorizes or imitates training data and doesn't generalize well with new "unseen" data (test data).



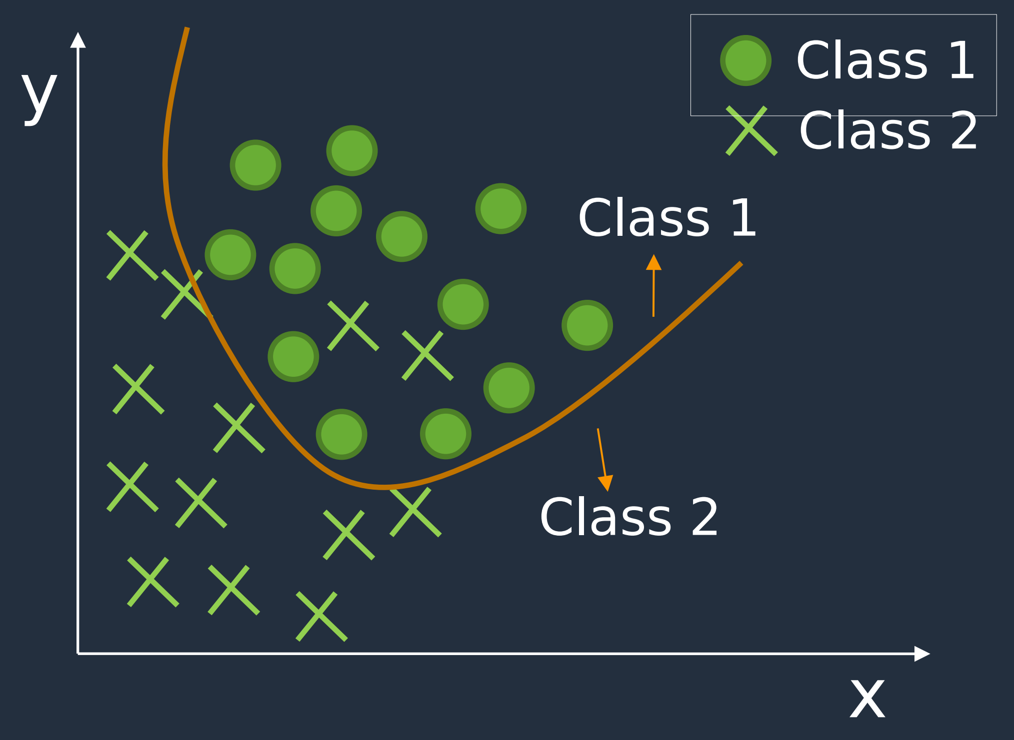
Having too complex models for simple problems can cause overfitting.

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

Appropriate fitting: It captures the general relationship between the input data (x) and output (y).



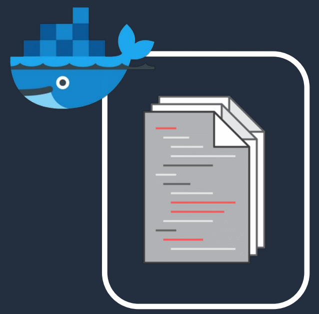
Amazon SageMaker Algorithms

- Matrix factorization
- Regression
- Principal component analysis
- K-means clustering
- Gradient boosted trees
- And more!

17 Built-in algorithms

mxnet
PYTORCH
TensorFlow
Chainer
scikit learn
APACHE Spark

Bring your own script
(Amazon SageMaker managed container)

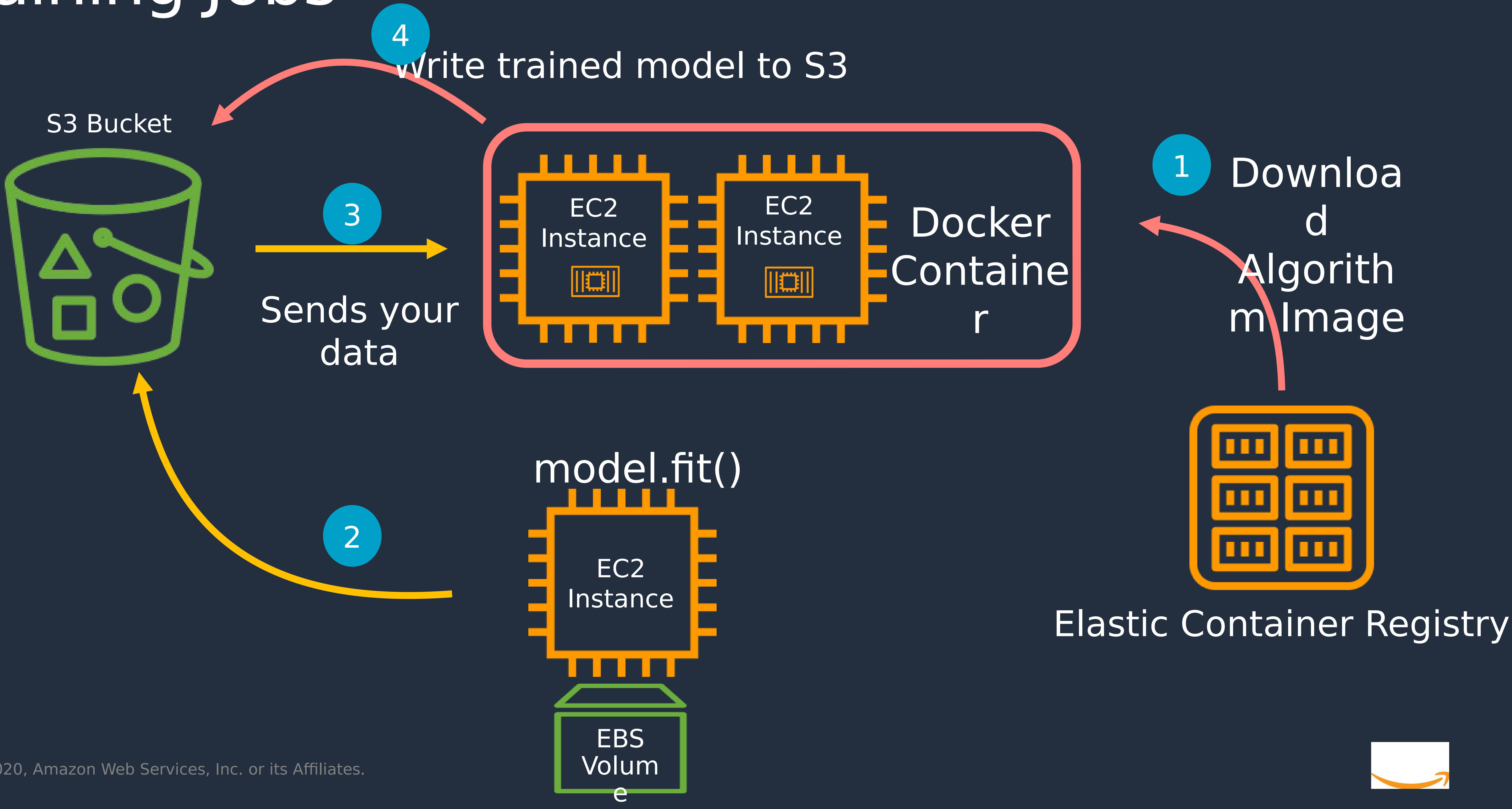


Bring your own algorithm
(you build the Docker container)

awsmarketplace

Subscribe to Algorithms and Model Packages on AWS Marketplace

Training Jobs

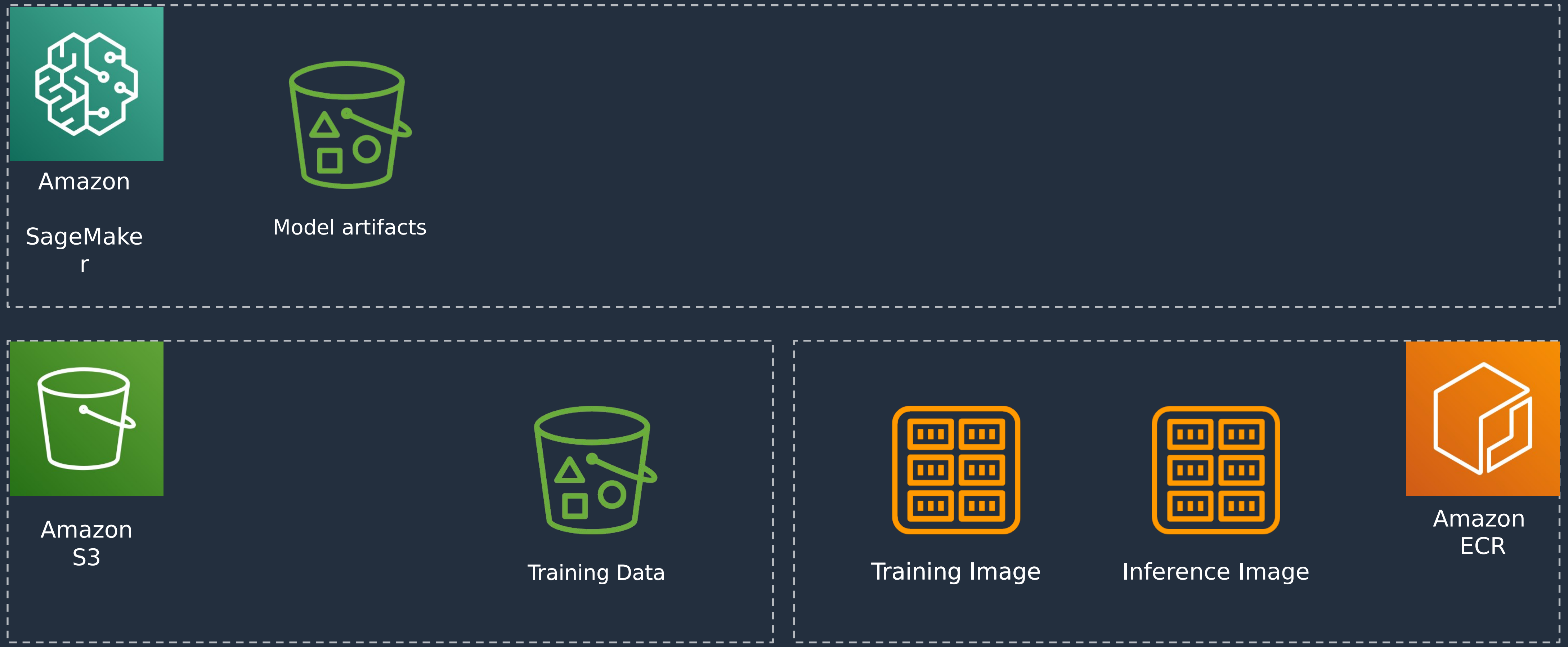


Every model run on a SageMaker training job has its own ephemeral cluster.

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

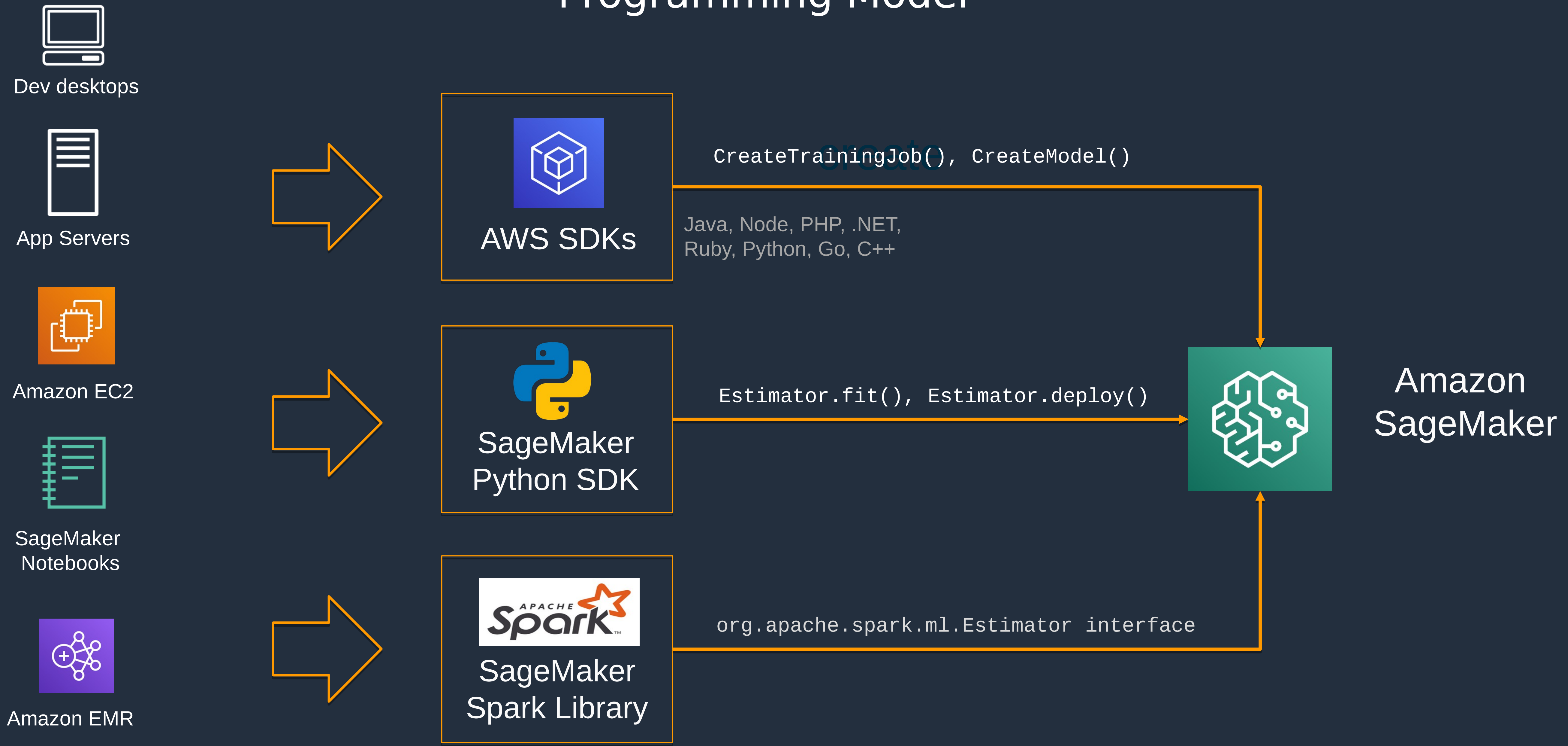
This cluster comes down immediately after the model finished training.

Amazon SageMaker training service

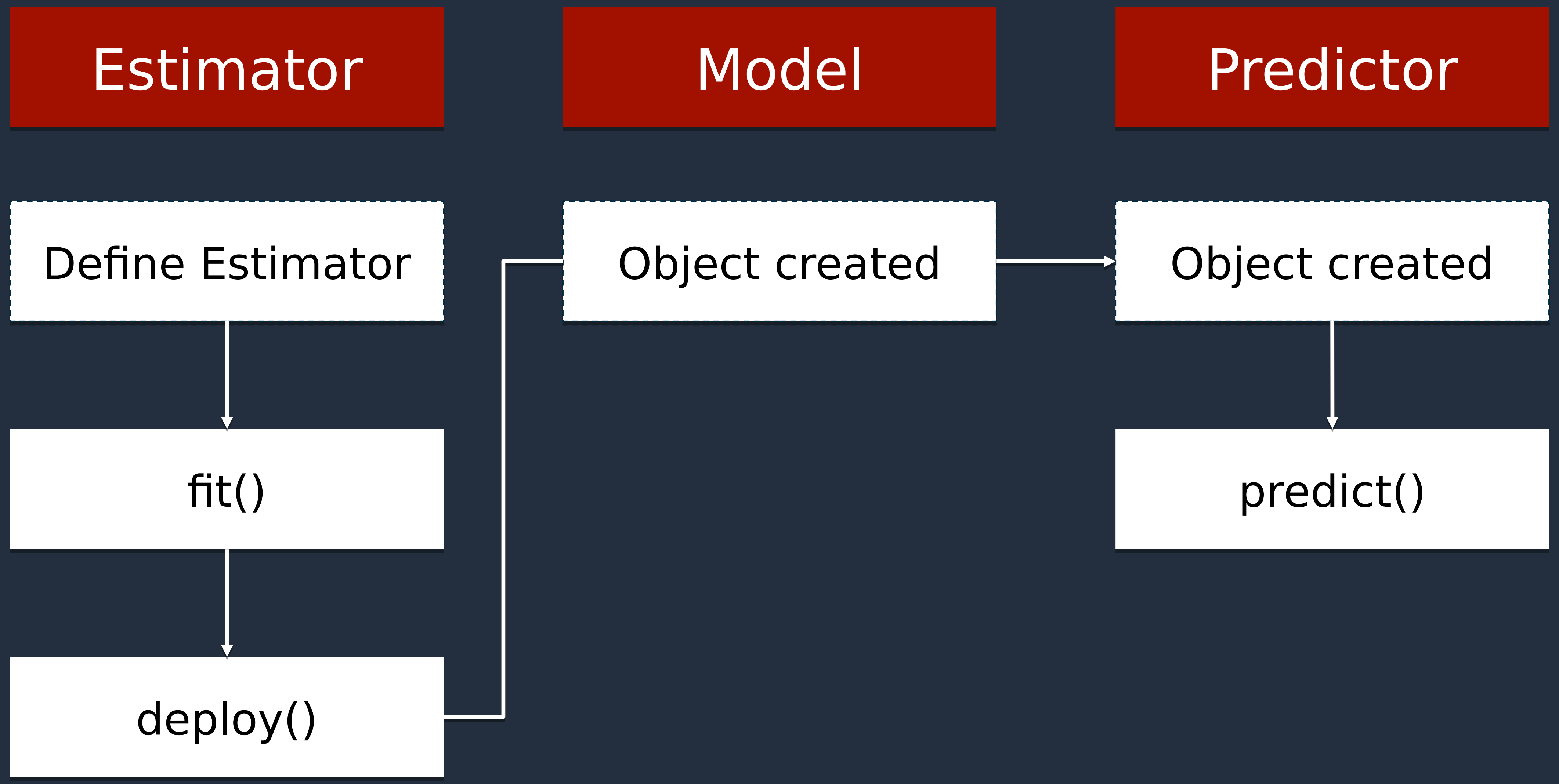


Amazon SageMaker | Training

Programming Model



End-to-End Flow



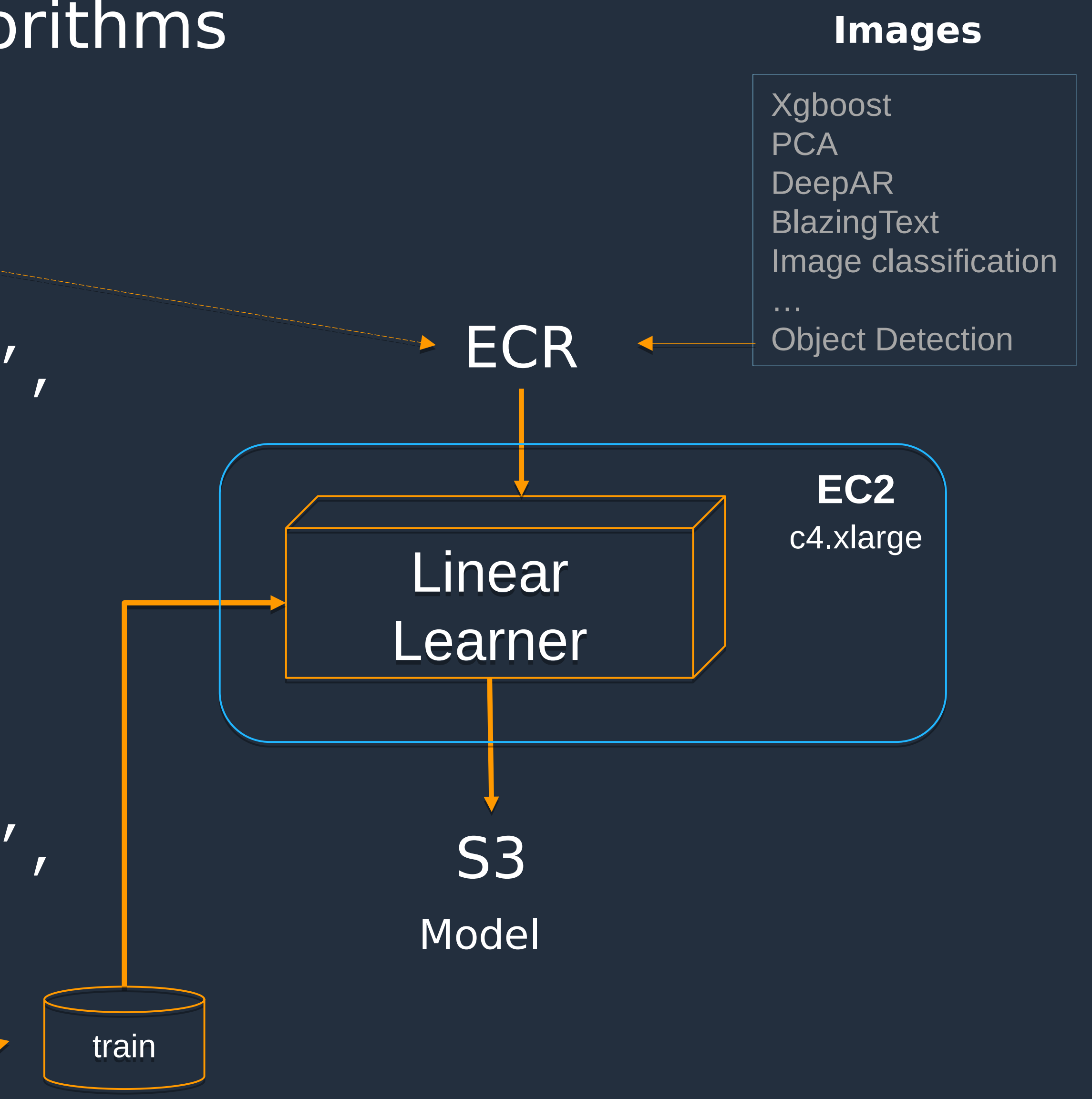
Amazon SageMaker | Training

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})
```



```
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

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

Confusion Matrix

	Your Model's Predictions	
Your Labeled Data	Positive	Negative
Positive	True Positive	False Negative
Negative	False Positive	True Negative

Recall

Precision

Evaluating classification models

		Predicted Response	
		$\hat{y} = 1$	$\hat{y} = 0$
True Response	y = 1	True Positive	False Negative
	y = 0	False Positive	True Negative

Recall (sensitivity):

Precision:

Precision: Accuracy of a predicted positive outcome.

Recall (sensitivity): Measures the strength of the model to predict a positive (1) outcome.

F1 – Score: Harmonic mean of precision and recall.

(good)

Tune ML Model in SageMaker

Amazon SageMaker Automatic Model Tuning

Hyperparameter Optimizer

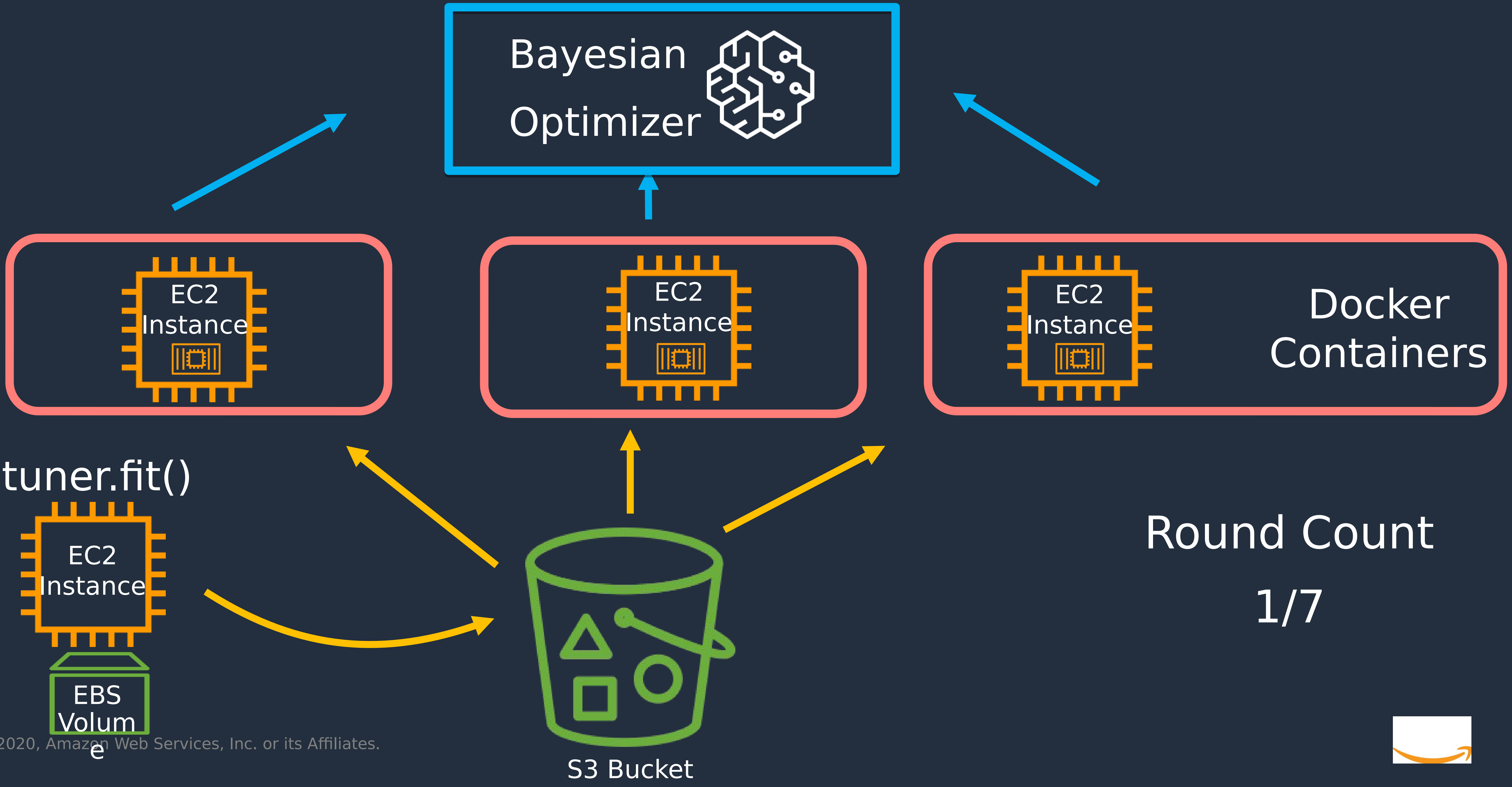
Decision Trees	Neural Networks
Tree depth	Number of layers
Max leaf nodes	Hidden layer width
Gamma	Learning rate
Eta	Embedding dimensions
Lambda	Dropout
Alpha	…
…	

…

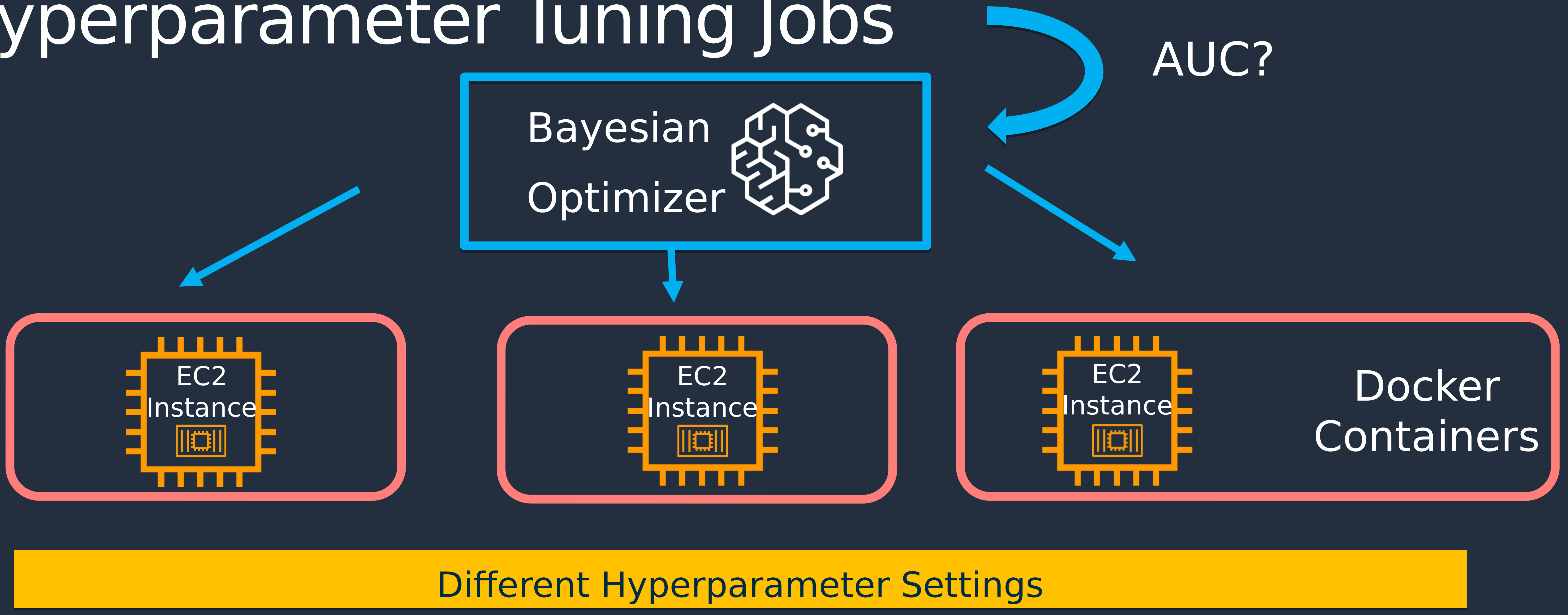
"Hyperparameters"

(algorithm parameters that significantly affect model quality)

Hyperparameter Tuning Jobs



Hyperparameter Tuning Jobs

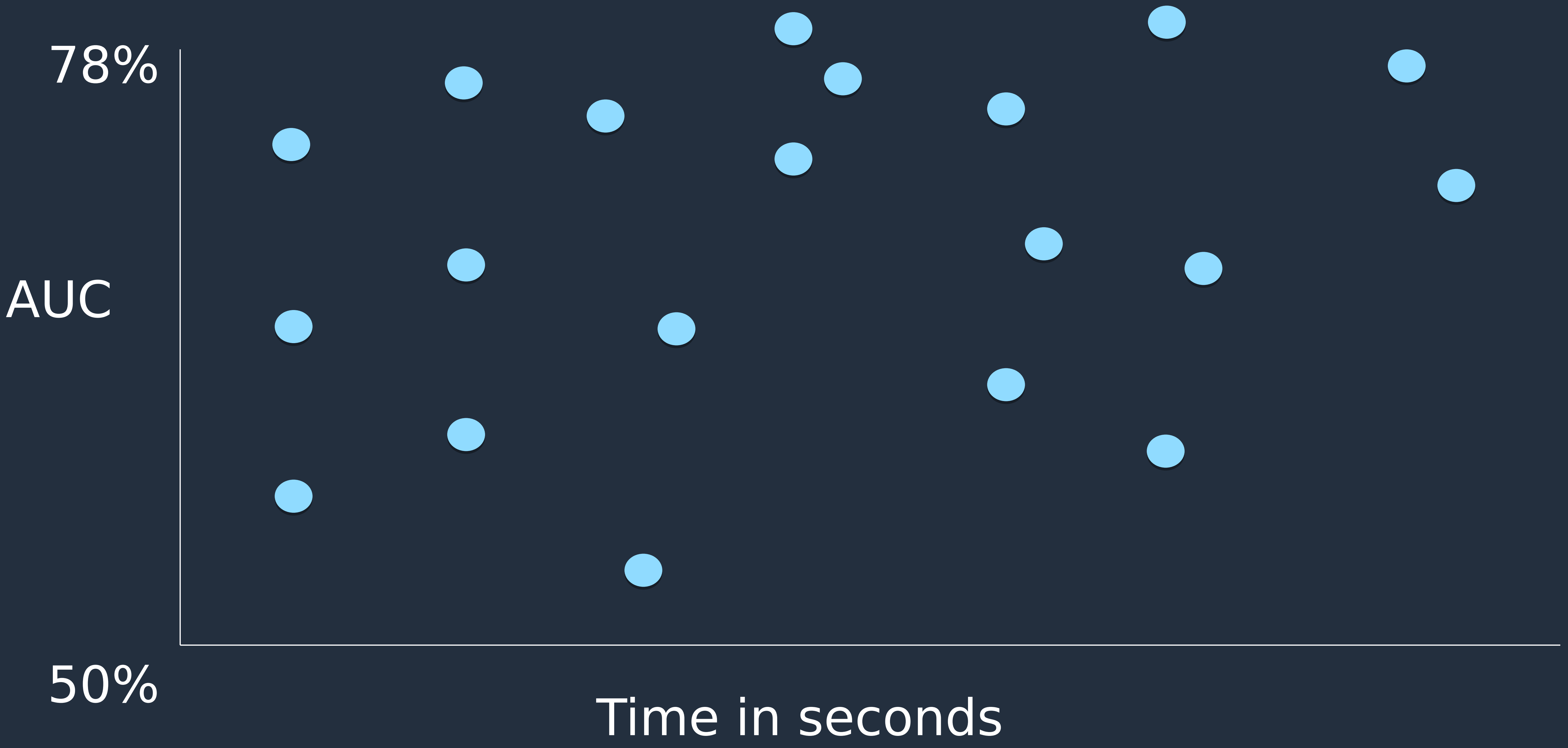


Round Count

2/7

Hyperparameter Tuning Jobs

Round 7/7



How do I set up a hyperparameter tuning job?

Objective Metric

δ

Hyperparameters & Ranges

Job Specs

Can I use hyperparameter tuning with my own model?

Yes!!!

1

Built-in Algorithms

2

Docker

3

Script Mode

Fully Customizable

Can I use hyperparameter tuning with my own model?

Setting the hyperparameters

In [5]:
```
hyperparameters = dict(batch_size=32, data_augmentation=True, learning_rate=.0001,
                       width_shift_range=.1, height_shift_range=.1, epochs=1)
hyperparameters
```

Out[5]:
```
{'batch_size': 32,
 'data_augmentation': True,
 'learning_rate': 0.0001,
 'width_shift_range': 0.1,
 'height_shift_range': 0.1,
 'epochs': 1}
```

Docker ✓

Script Mode ✓

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)
```

What if I need all my jobs tuned at the same time?

Random search

What if I need all my jobs tuned at the same time?

What if I need all my jobs tuned at the same time?

Random search

```
{
    "ParameterRanges": {...}
    "Strategy": "Random",
    "HyperParameterTuningJobObjective": {...}
}
```

```
tuner = HyperparameterTuner(
    sagemaker_estimator,
    objective_metric_name,
    hyperparameter_ranges,
    max_jobs=20,
    max_parallel_jobs=20,
    strategy="Random"
)
```

Amazon SageMaker Automatic Model Tuning

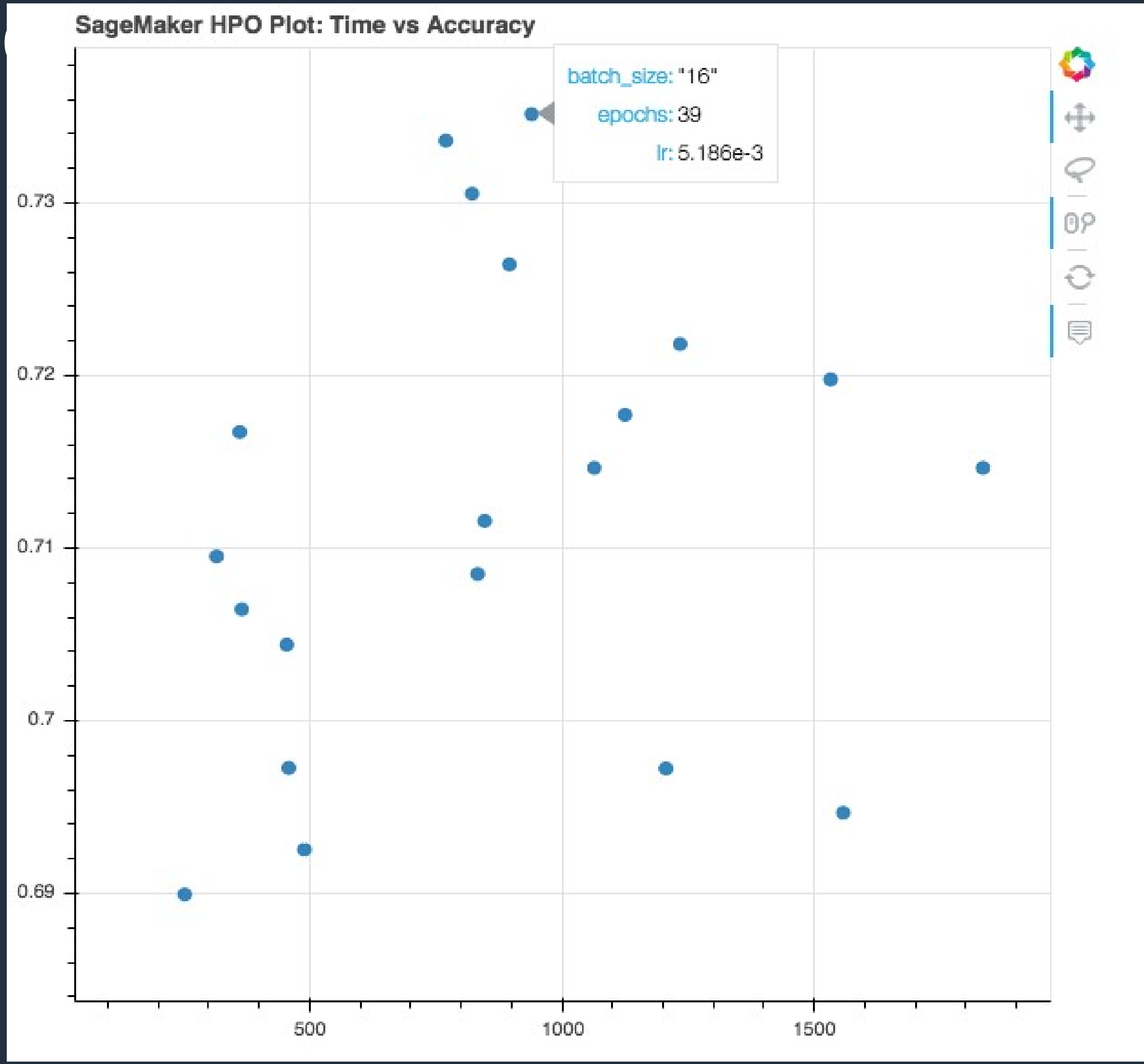
```
from sagemaker.tuner import IntegerParameter, CategoricalParameter, ContinuousParameter, Hyperparameter

hyperparameter_ranges = {
                         'batch_size': CategoricalParameter([16, 32, 64, 128]),
                         'learning_rate': ContinuousParameter(0.0009, 0.
                         'epochs': IntegerParameter(10, 100)}

objective_metric_name = 'Validation-accuracy'
metric_definitions = [{'Name': 'Validation-accuracy',
                       'Regex': 'Validation-accuracy=([0-9\\.]+)'}]

tuner = HyperparameterTuner(estimator,
                            objective_metric_name,
                            hyperparameter_ranges,
                            metric_definitions,
                            max_jobs=20,
                            max_parallel_jobs=4)

tuner.fit(dataset_location, job_name=job_name)
```

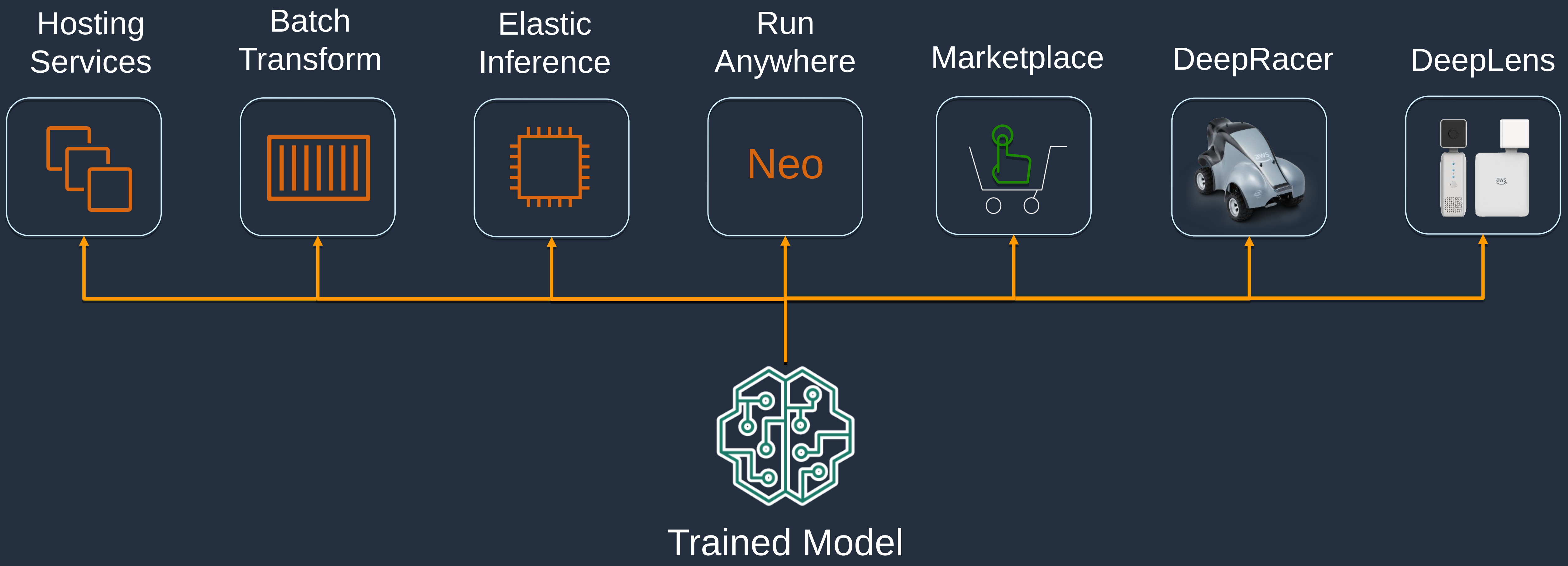


Pro Tips

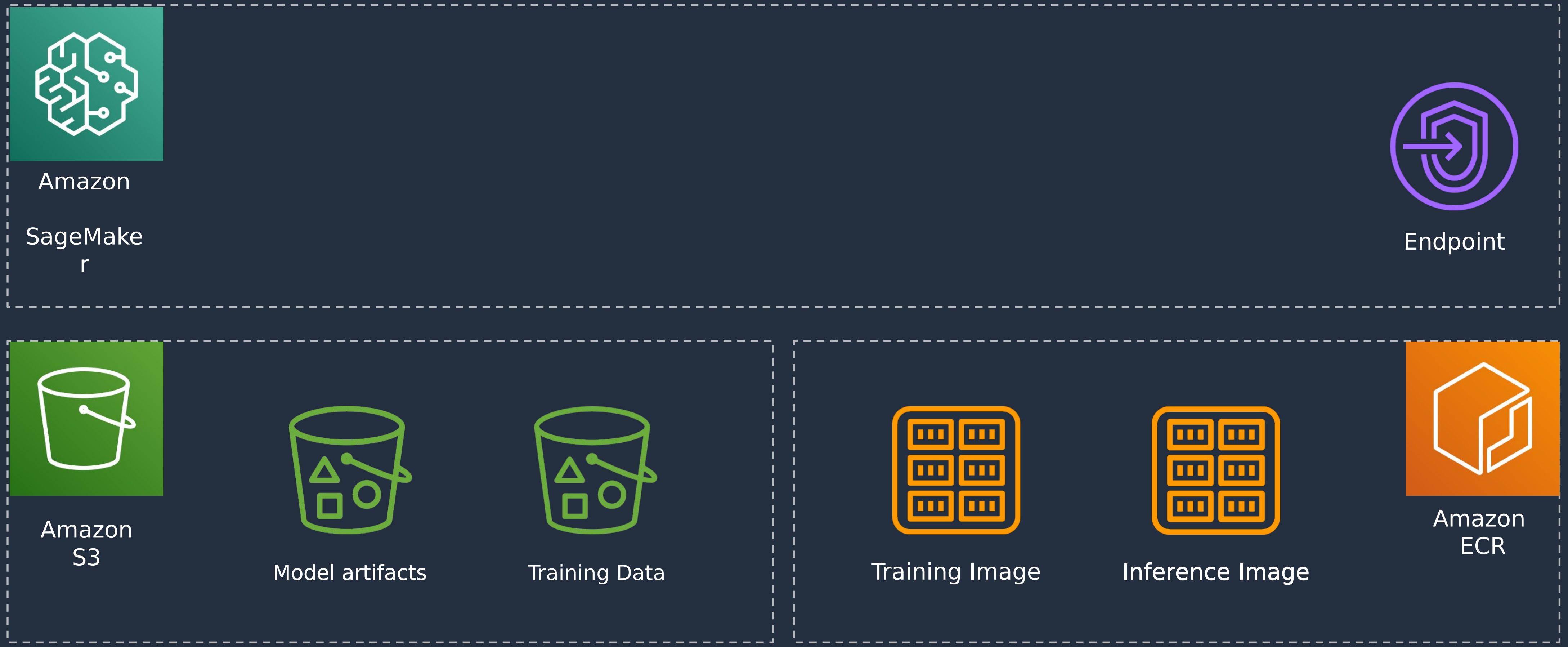
- S3 bucket must be in the same region as your training job
- Use SageMaker Search to compare results of previous jobs
- Run training jobs in parallel from your notebook
- Soft limit on number of instances used for training job
- Check docs on which hyperparameters are valid for tuning
- When bringing your own algorithm, set hyperparameters as JSON object

Model Deployment and Inferencing with SageMaker

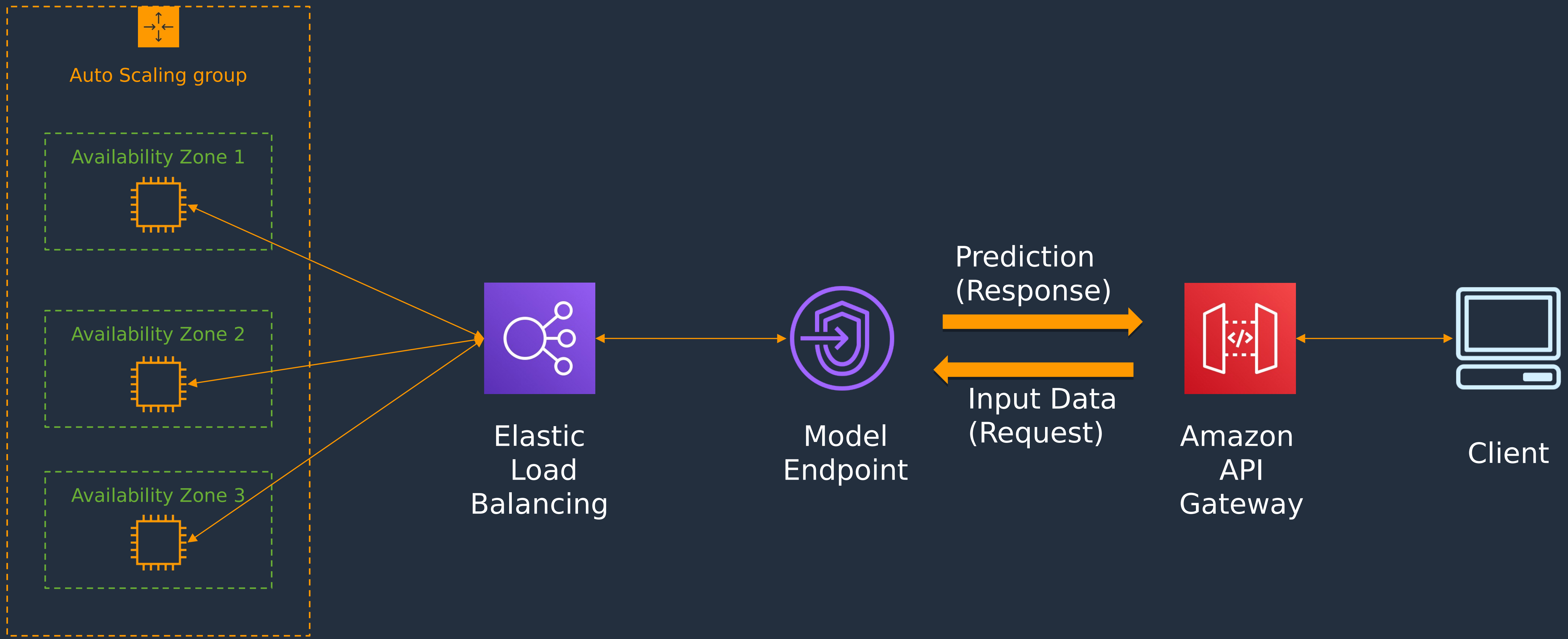
Amazon SageMaker Deployment Options



Amazon SageMaker hosting service



SageMaker Endpoints



Deployment / Hosting
Amazon SageMaker ML Compute Instances

Command Line - Creating and Scaling Model Endpoints

Easy deployment to production REST API

Scalable, high throughput, and high reliability

Creating endpoints

Model

```
aws sagemaker create-model
  --model-name model1
  --primary-container ‘{“Image”: “123.dkr.ecr.amazonaws.com/algo”,
                        “ModelDataUrl”: “s3://bkt/model1.tar.gz”}’
  --execution-role-arn arn:aws:iam::123:role/me
```

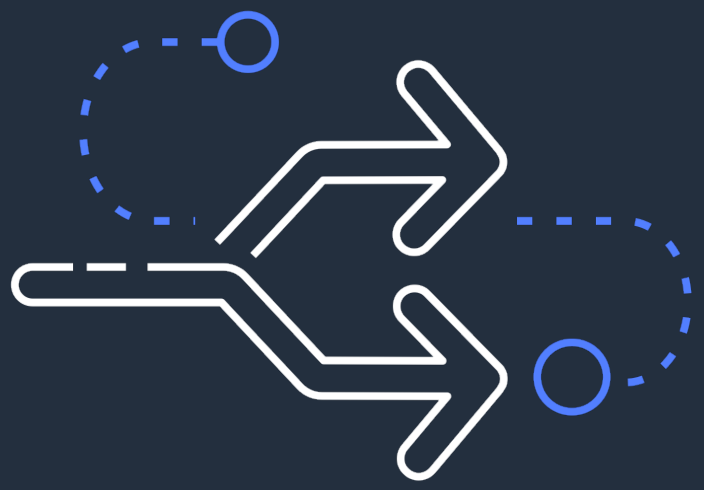
Endpoint configuration

```
aws sagemaker create-endpoint-config
  --endpoint-config-name model1-config
  --production-variants ‘{“InitialInstanceCount”: 2,
                          “InstanceType”: “ml.m4.xlarge”,
                          ”InitialVariantWeight”: 1,
                          ”ModelName”: “model1”,
                          ”VariantName”: “AllTraffic”}’
```

Endpoint

```
aws sagemaker create-endpoint
  --endpoint-name my-endpoint
  --endpoint-config-name model1-config
```

Updating endpoints



Blue-green deployments mean no scheduled downtime

Deploy one or more models behind the same endpoint

Updating endpoints

New model

```
aws sagemaker create-model
 --model-name model2
 --primary-container ‘{“Image”: “123.dkr.ecr.amazonaws.com/algo”,
                      “ModelDataUrl”: “s3://bkt/model2.tar.gz”}
 --execution-role-arn arn:aws:iam::123:role/me
```

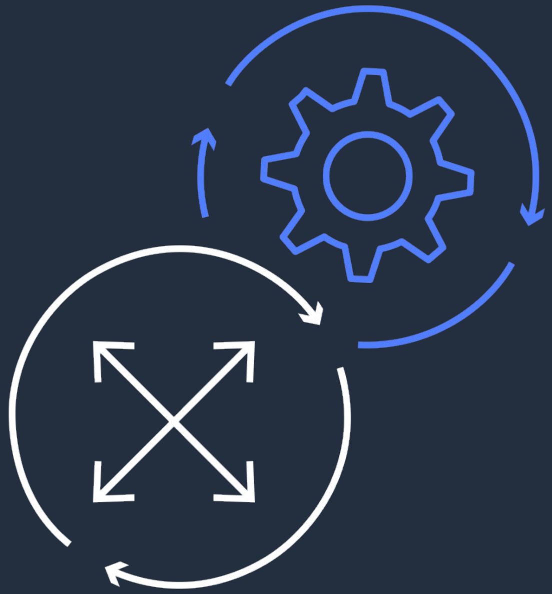
New endpoint configuration

```
aws sagemaker create-endpoint-config
  --endpoint-config-name model2-config
  --production-variants ‘{“InitialInstanceCount”: 2,
                          “InstanceType”: “ml.m4.xlarge”,
                          ”InitialVariantWeight”: 1,
                          ”ModelName”: “model2”,
                          ”VariantName”: “AllTraffic”}’
```

Same endpoint

```
aws sagemaker update-endpoint
  --endpoint-name my-endpoint
  --endpoint-config-name model2-config
```

Reduced risk deployments



Incrementally retrain models with new data

Try new models and improved algorithms

Reduced risk deployments

Two-model endpoint configuration

```
aws sagemaker create-endpoint-config
  --endpoint-config-name both-models-config
  --production-variants ‘[{“InitialInstanceCount”: 2,
                          “InstanceType”: “ml.m4.xlarge”,
                          ”InitialVariantWeight”: 95,
                          ”ModelName”: “model1”,
                          ”VariantName”: “model1-traffic”},
                         {“InitialInstanceCount”: 2,
                          “InstanceType”: “ml.m4.xlarge”,
                          ”InitialVariantWeight”: 5,
                          ”ModelName”: “model2”,
                          ”VariantName”: “model2-traffic”}]’
```

Same endpoint

```
aws sagemaker update-endpoint
  --endpoint-name my-endpoint
  --endpoint-config-name both-models-config
```

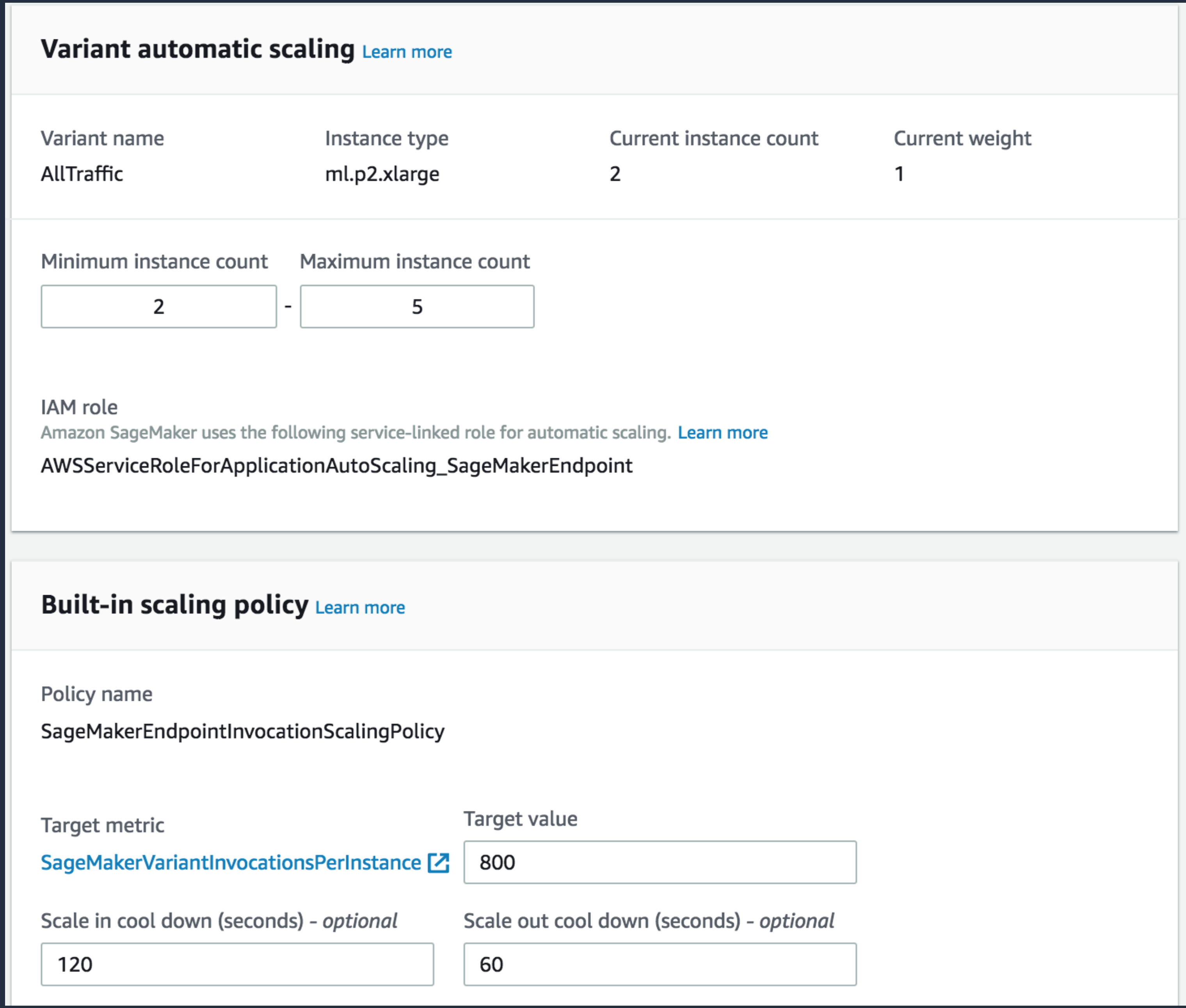
Swap

```
aws sagemaker update-endpoint-weights-and-capacities
  --endpoint-name my-endpoint
  --desired-weights-and-capacities ‘{“VariantName”: ”model1”,
                                    “DesiredWeight”: 5}’
```

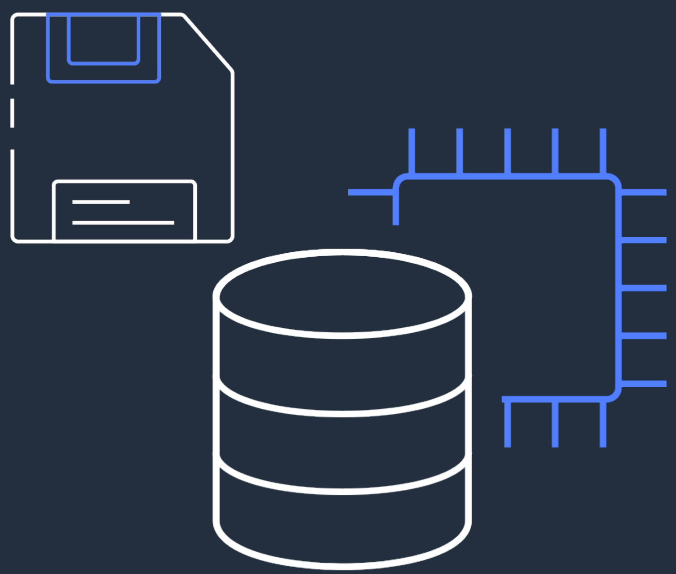
Automatic scaling endpoints

SageMaker console settings:

- Min and max instances
- Target invocations per instance
- Scaling cooldowns



Scaling criteria



Algorithms have different memory, CPU, or GPU requirements

Automatically scale based on endpoint instance's Amazon CloudWatch metrics

Creating an automatic scaling policy

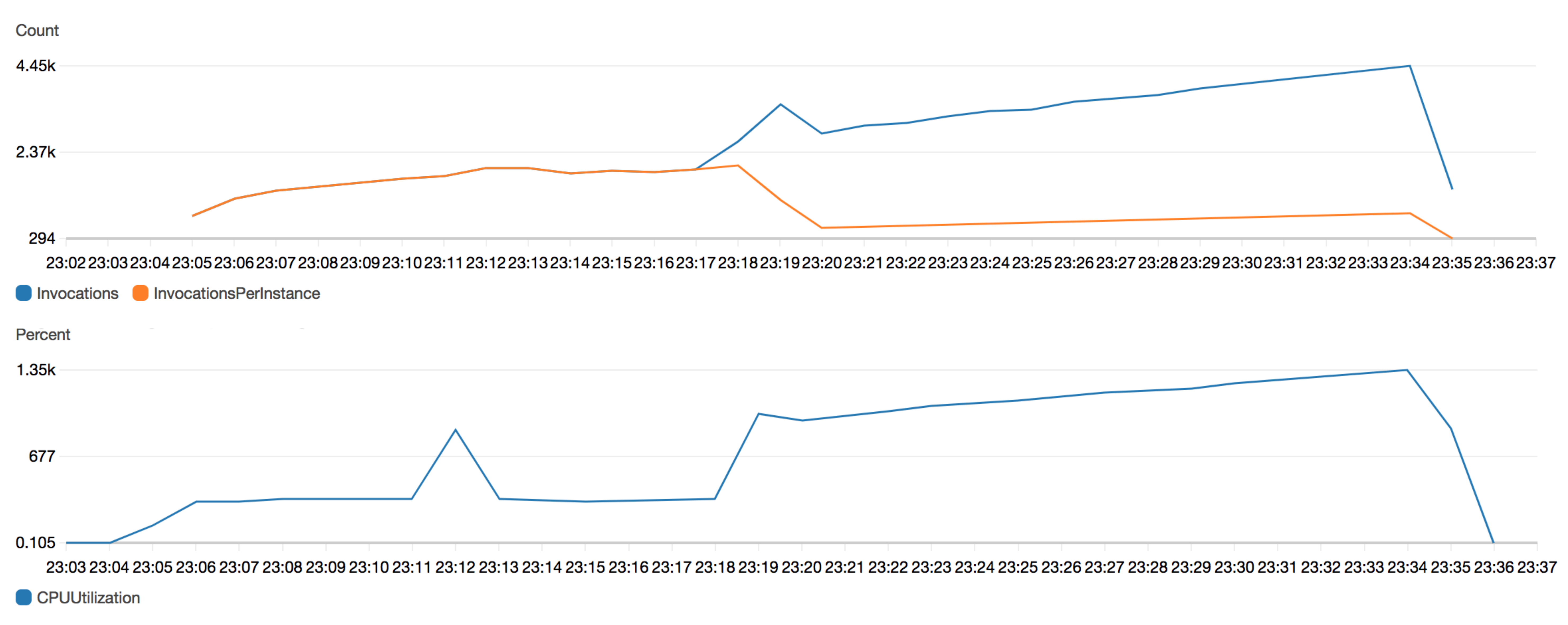
Variant

```
aws application-autoscaling register-scalable-target
  --service-namespace sagemaker
  --resource-id endpoint/my-endpoint/variant/model2
  --scalable-dimension sagemaker:variant:DesiredInstanceCount
  --min-capacity 2
  --max-capacity 5
```

Policy

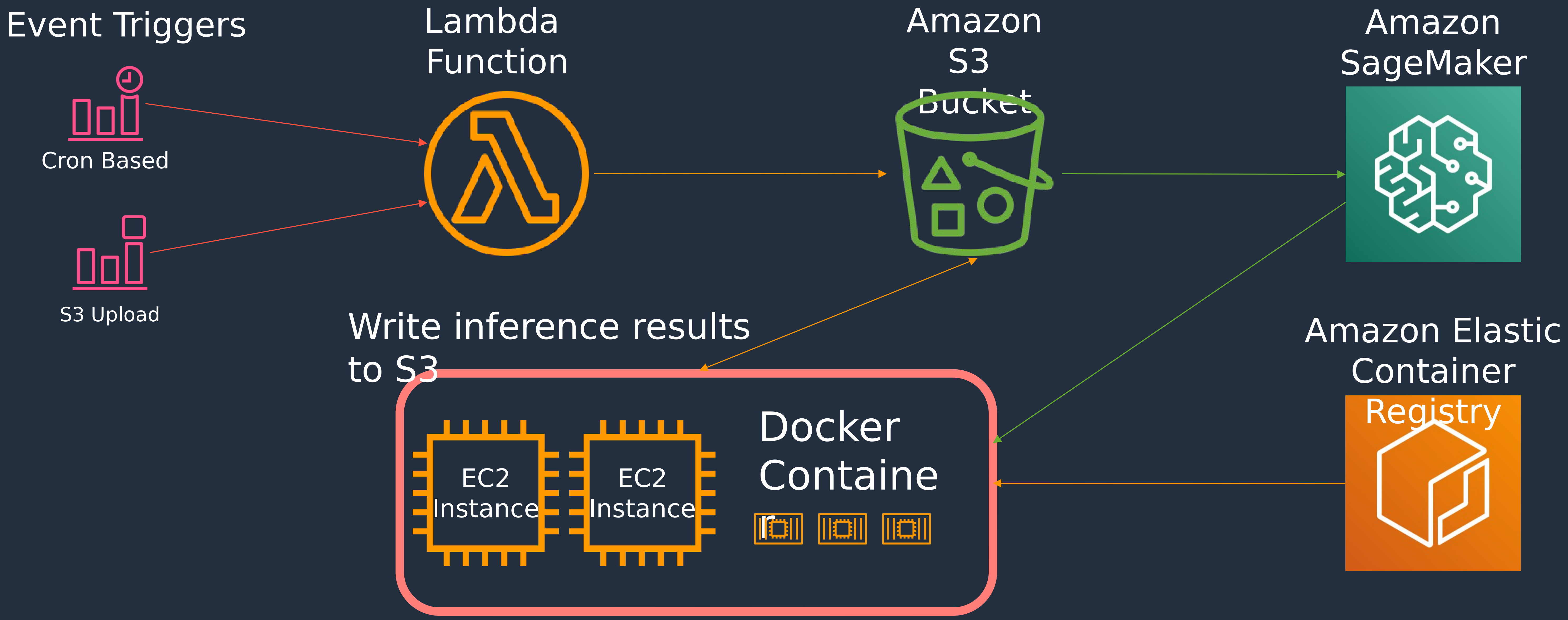
```
aws application-autoscaling put-scaling-policy
  --policy-name model2-scaling
  --service-namespace sagemaker
  --resource-id endpoint/my-endpoint/variant/model2
  --scalable-dimension sagemaker:variant:DesiredInstanceCount
  --policy-type TargetTrackingScaling
  --target-tracking-scaling-policy-configuration
    ‘{"TargetValue": 50,
      "CustomizedMetricSpecification":
        {"MetricName": "CPUUtilization",
         "Namespace": "/aws/sagemaker/Endpoints",
         "Dimensions":
           [{"Name": "EndpointName", "Value": "my-endpoint"},
            {"Name": "VariantName","Value": ”model2"}],
        "Statistic": "Average",
        "Unit": "Percent”}}’
```

Scale by utilization



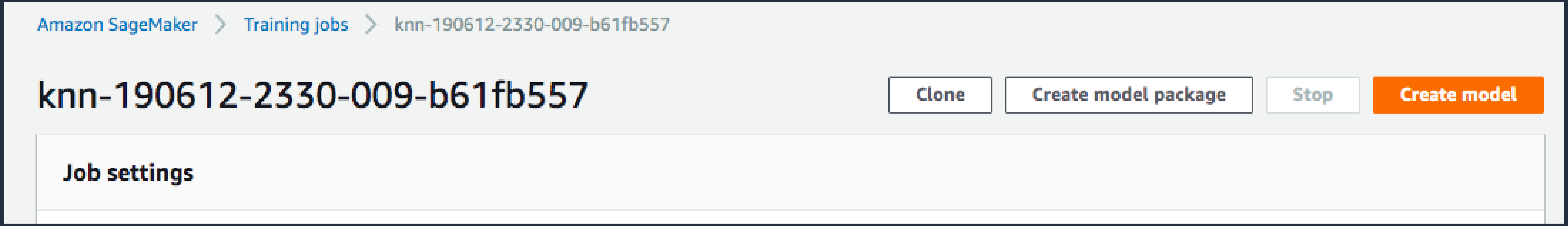

Endpoints are easy to spin up, but what if I need to serve predictions on a schedule?
Batch transform!
Customer
Cloud Architect

Batch Transform Common Design Pattern

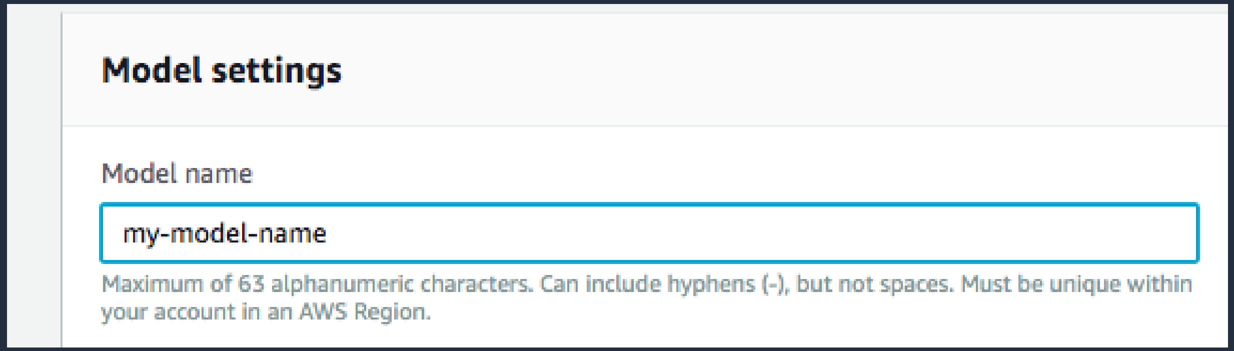


SageMaker Batch Transform

1.

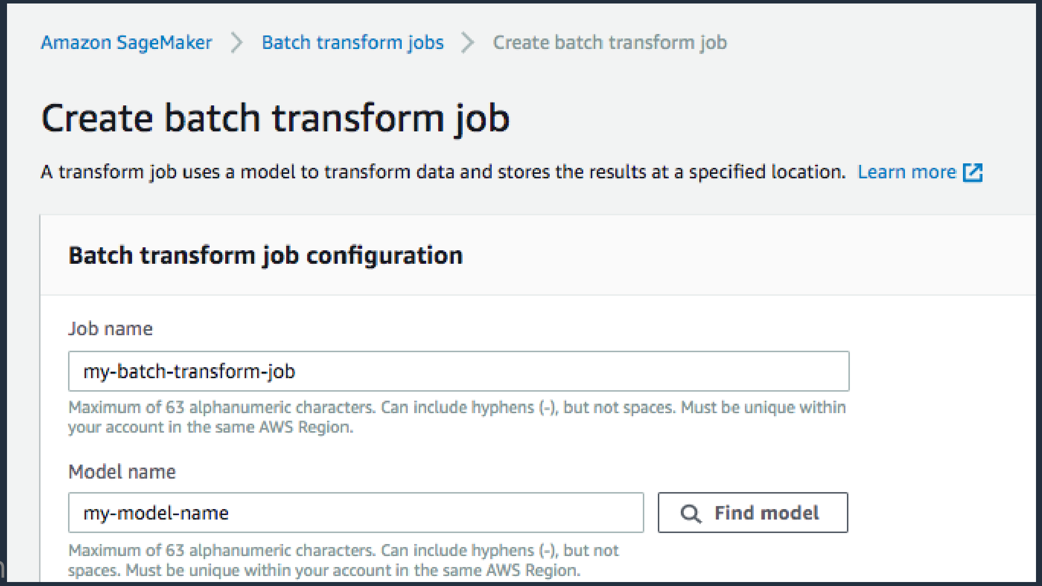


2.



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

3.



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

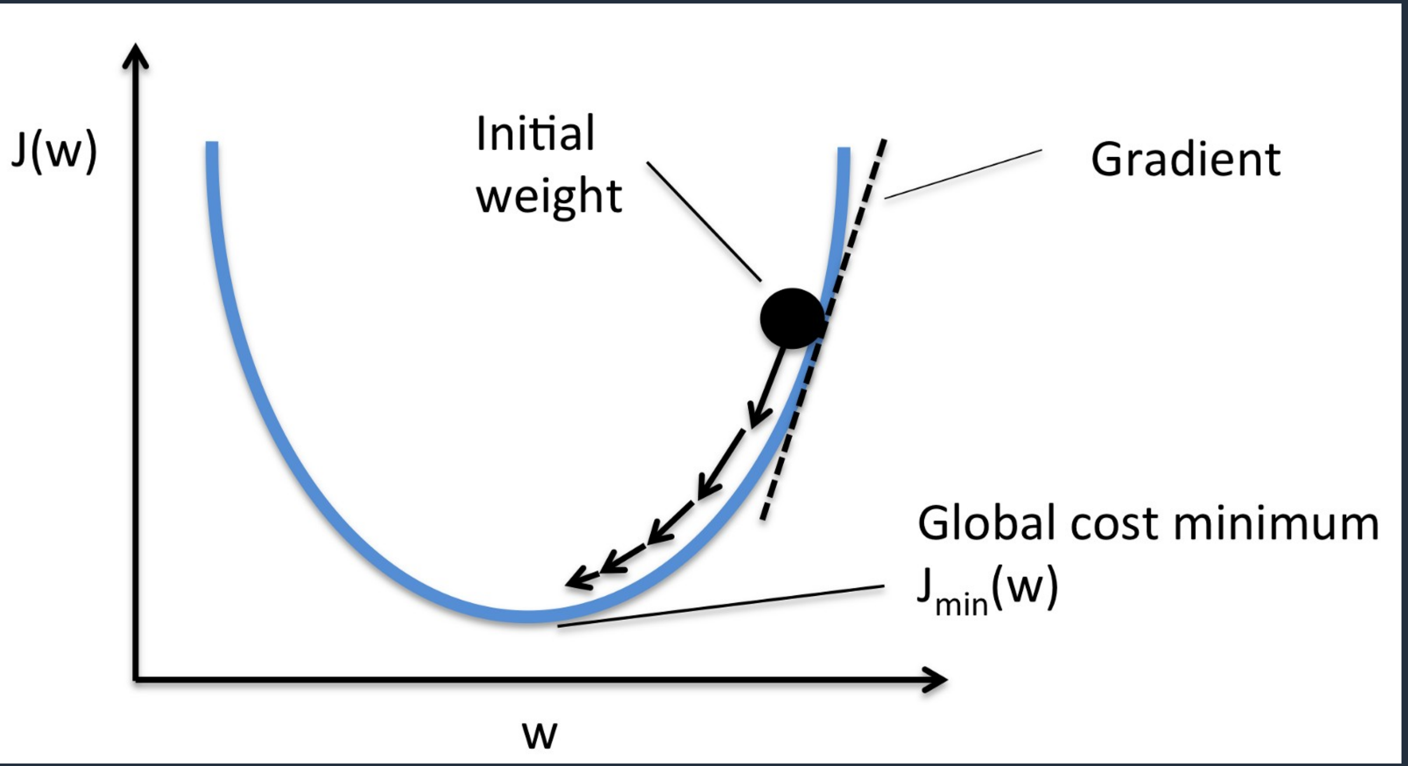
Pro Tips

- Turn your endpoints off when not in use with Lambda
- Use inference pipelines for pre and post processing
- Bring more models in your own Docker container
- Consider the size of data hitting a single endpoint
- You can train your model elsewhere and host in SageMaker

Thank you!!!

Appendix

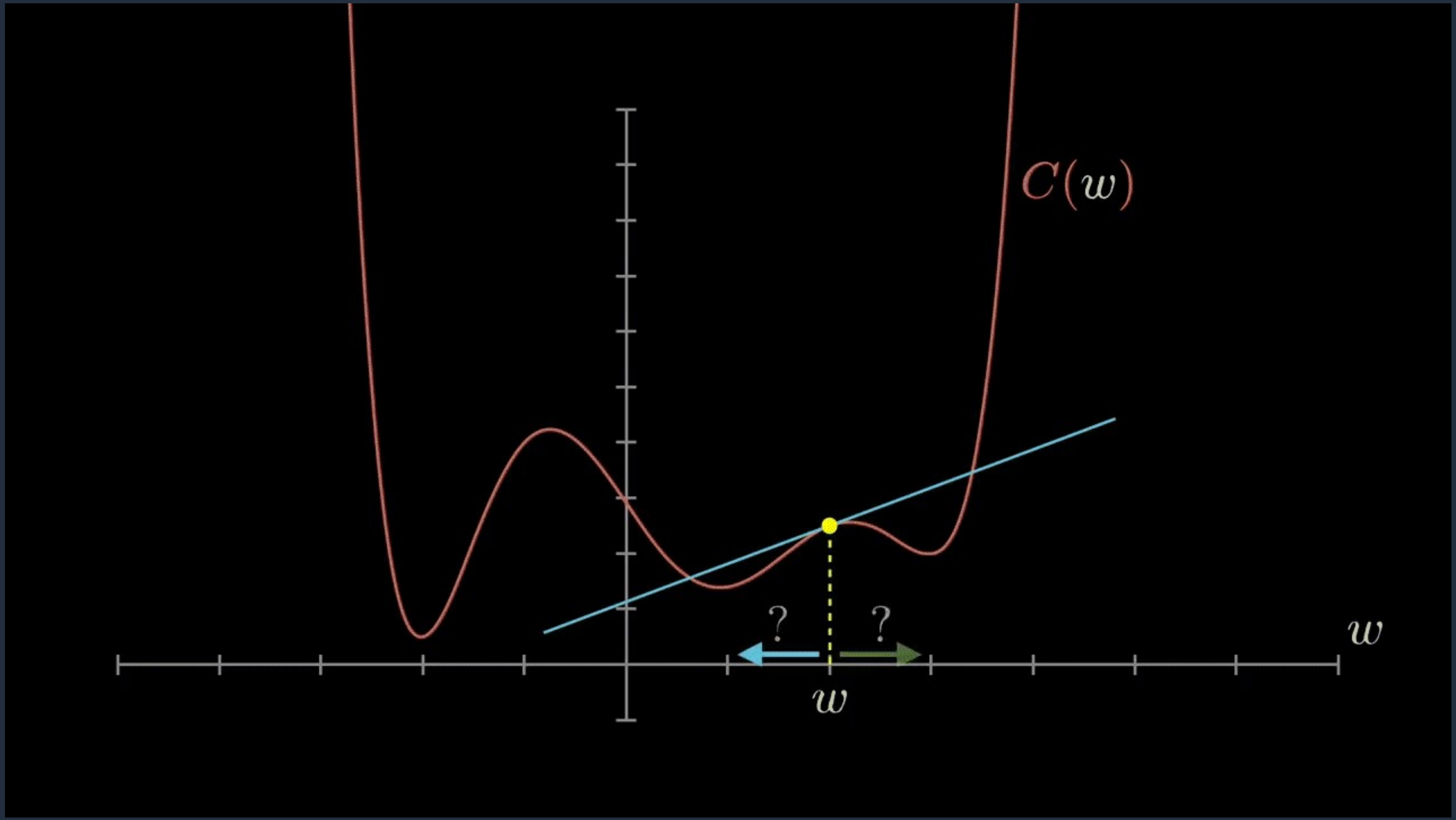
Gradient Descent



At each step, we take a step towards minimum by the value of gradient*learning rate.

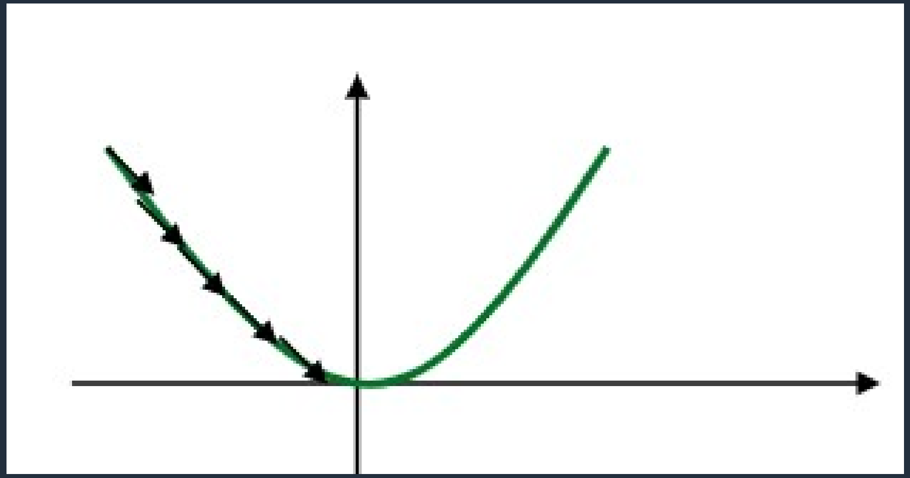
Towards mid sections, gradient (slope) becomes smaller, therefore, steps become smaller.

Gradient Descent

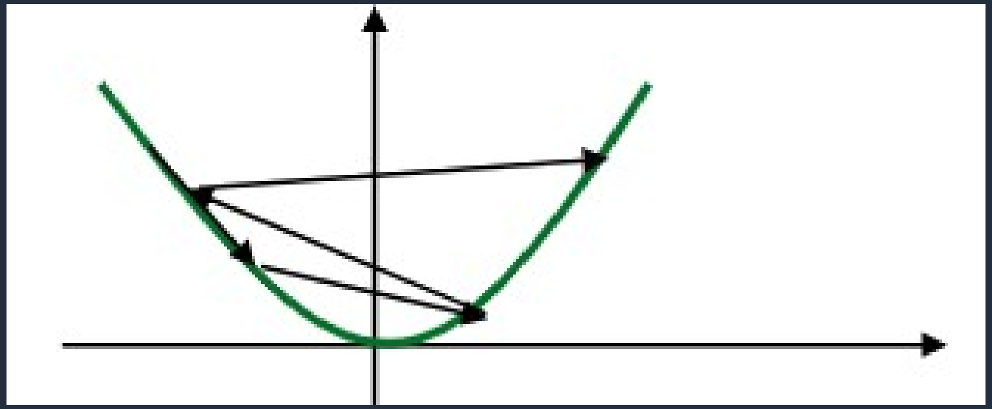


Learning rate

If learning rate is small, it can converge slowly.



If learning rate is too large, gradient descent can over-shoot the minimum.



Bagging

Bagging is an ensemble learning technique where we:

1. Draw bootstrap samples from the training set

2. Train a ML model on each sample