Copyright Notice

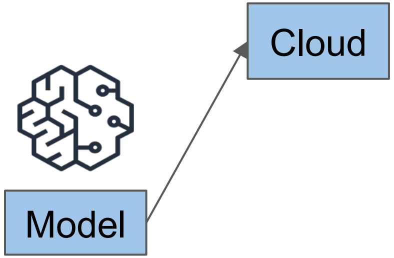
Model Deployment

Overview

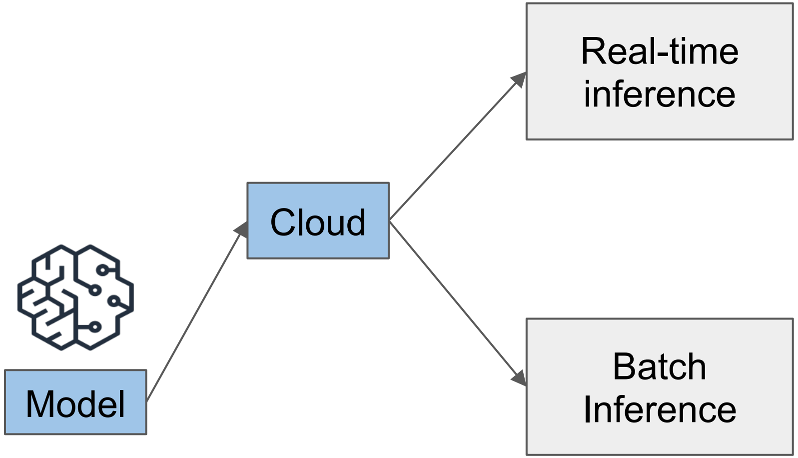
Machine Learning Workflow

Ingest & Analyze	Prepare & Transform	Train & Tune	Deploy & Manage
Data exploration	Feature engineering	Automated ML	Model deployment
Bias detection	Feature store	Model train and tune	Automated pipelines
Amazon S3 & Amazon Athena	Amazon SageMaker Data Wrangler	Amazon SageMaker Autopilot	Amazon SageMaker Endpoints
AWS Glue	Amazon SageMaker Processing Jobs	Amazon SageMaker Training & Debugger	Amazon SageMaker Batch Transform
Amazon SageMaker Data Wrangler & Clarify	Amazon SageMaker Feature Store	Amazon SageMaker Hyperparameter Tuning	Amazon SageMaker Pipelines

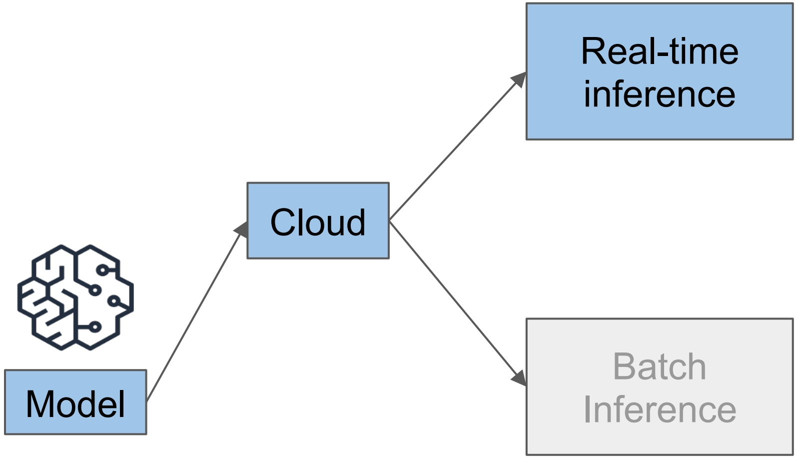
Deployment Options



Deployment Options

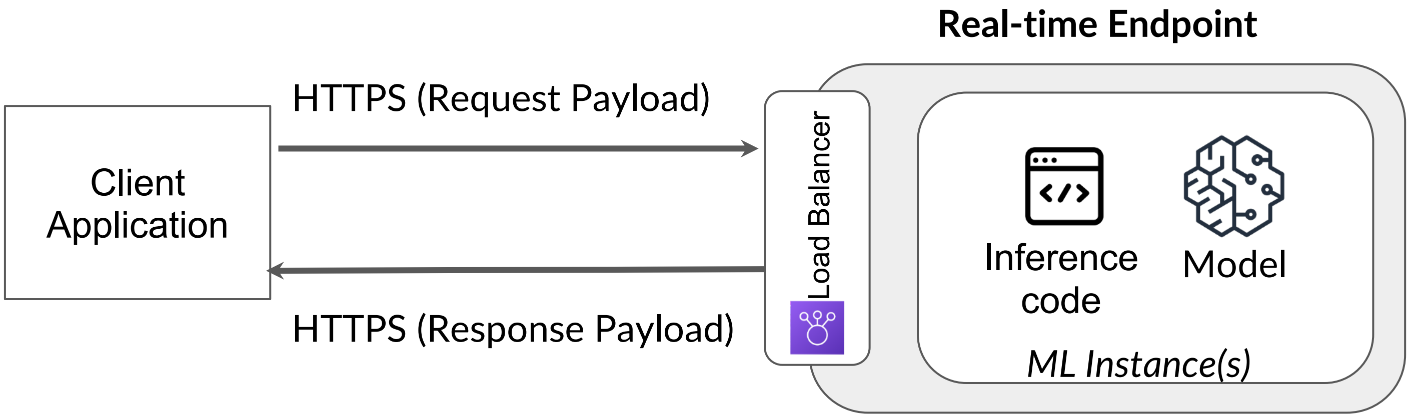


Deployment Options



Deployment Options

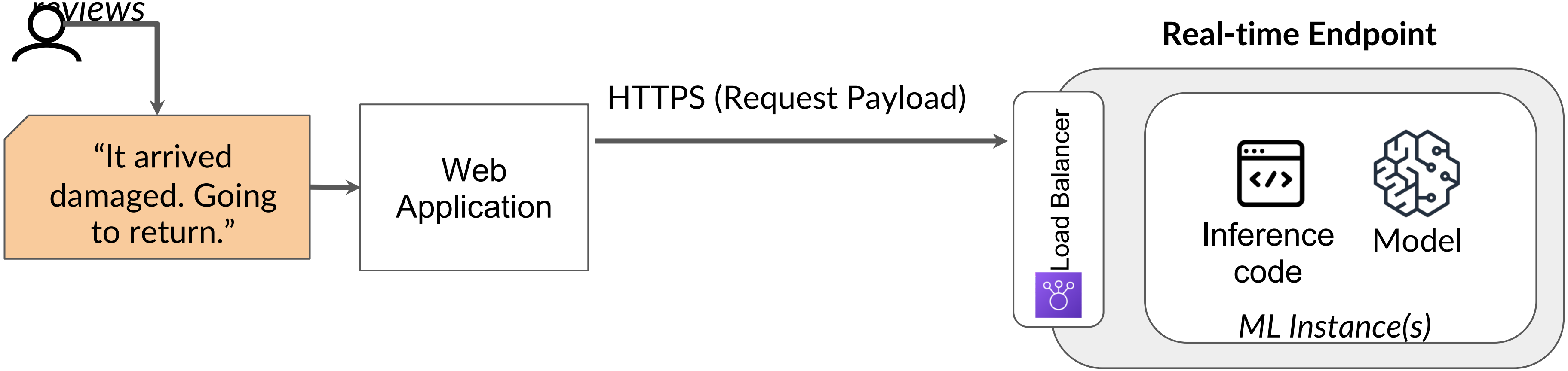
Real-Time Inference



Deployment Options

Real-Time Inference - Product Review Example

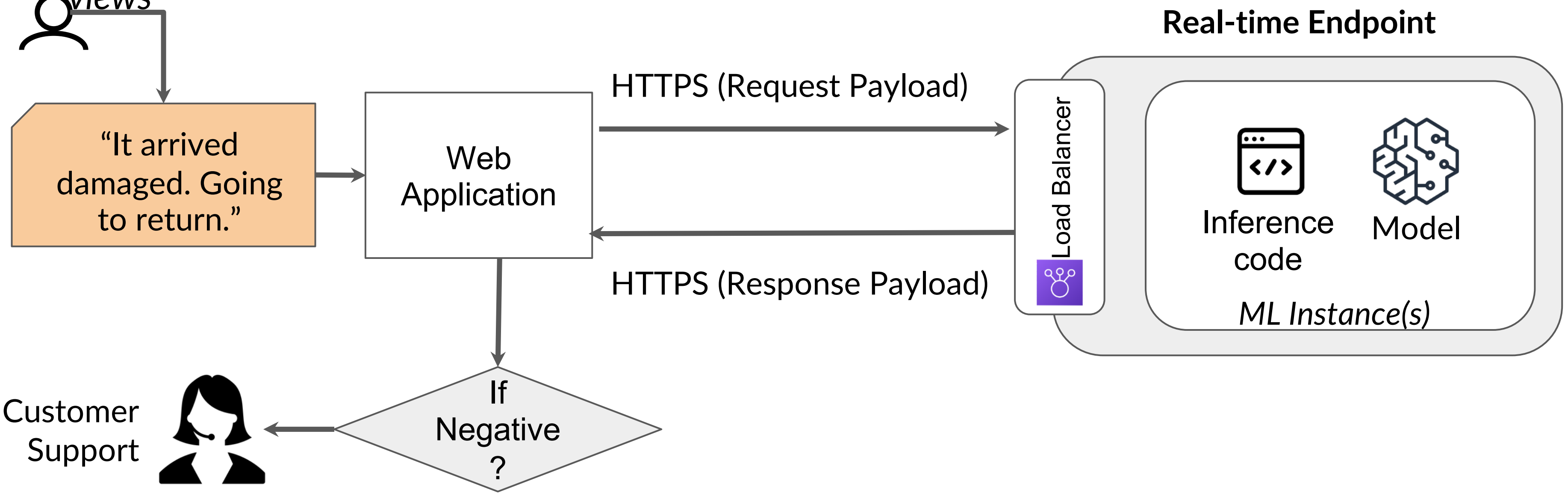
Goal: *Improve customer experience with quick responses to negative reviews*



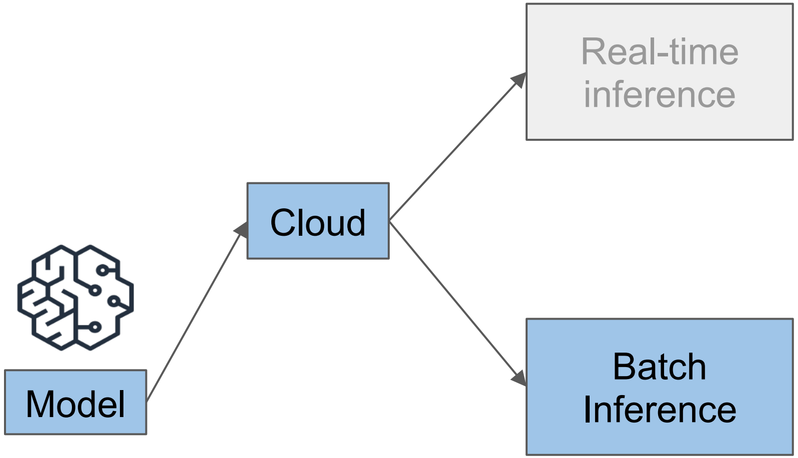
Deployment Options

Real-Time Inference - Product Review Example

Goal: *Improve customer experience with quick responses to negative reviews*

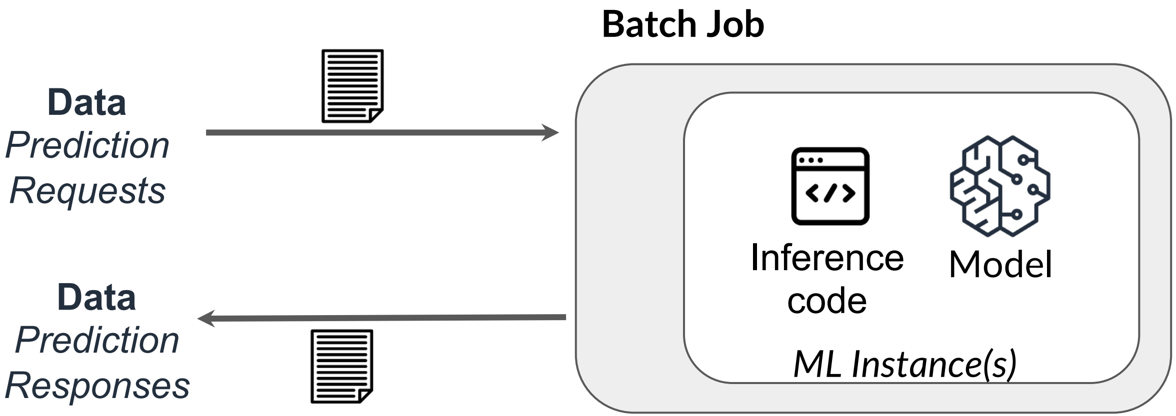


Deployment Options



Deployment Options

Batch Inference



Deployment Options

Batch Inference

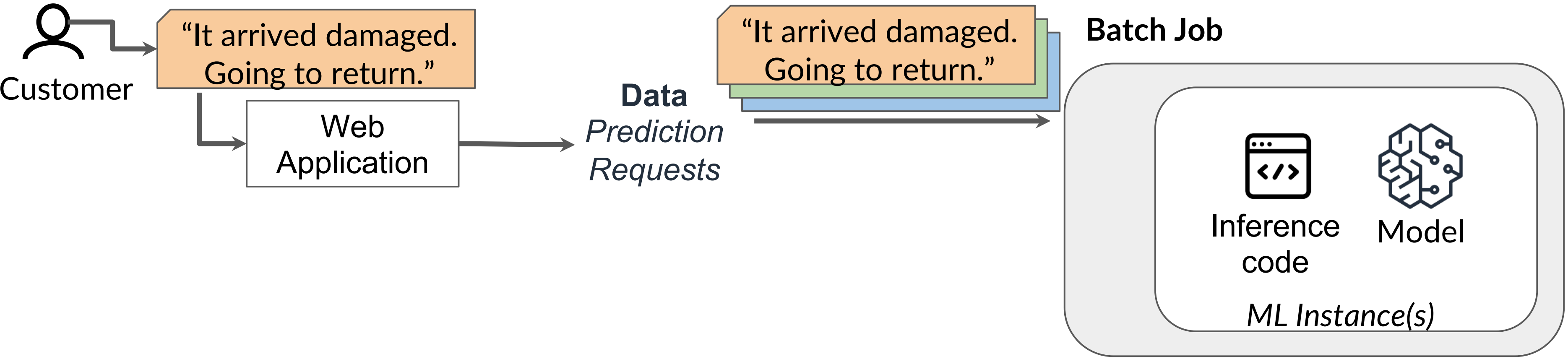
Deployment Options

Batch Inference

Deployment Options

Batch Inference - Product Review

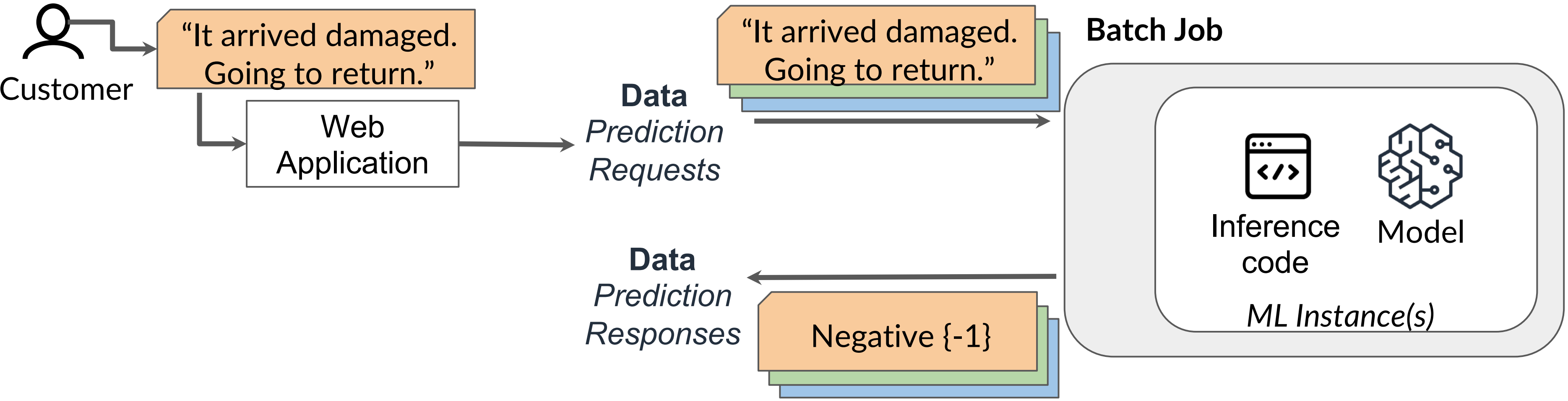
Goal: *Identify vendors with potential quality issues*



Deployment Options

Batch Inference - Product Review

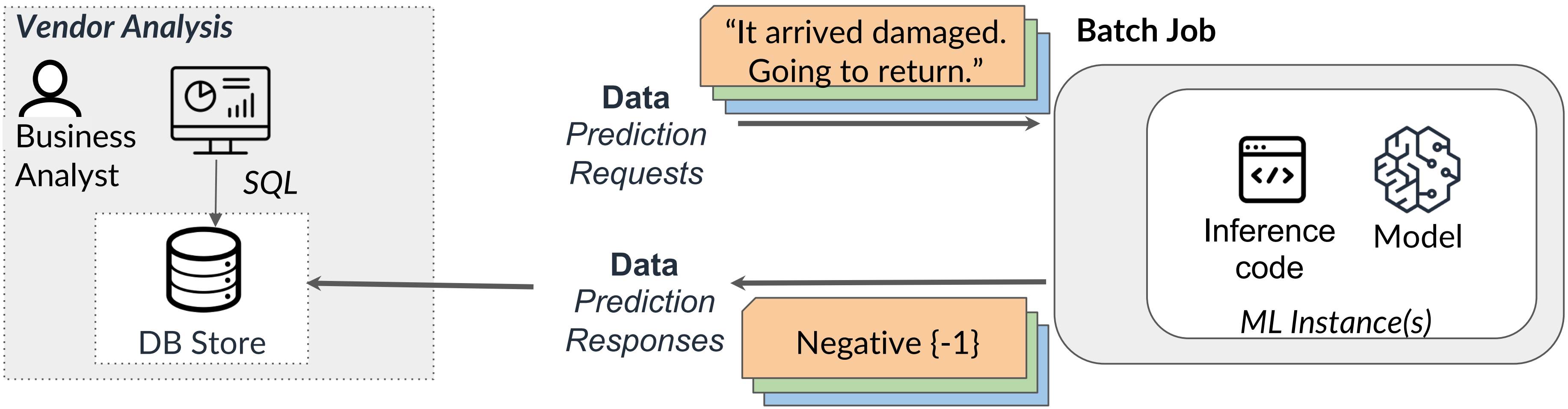
Goal: *Identify vendors with potential quality issues*



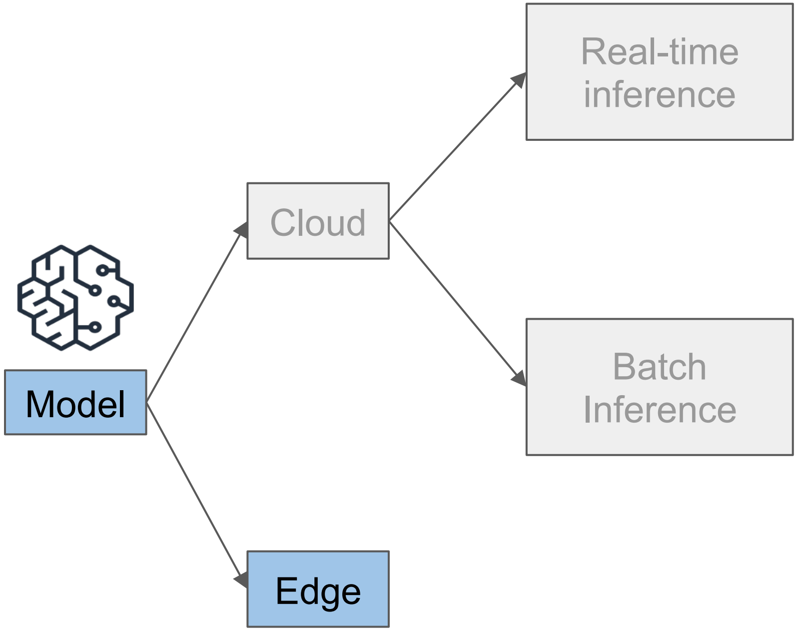
Deployment Options

Batch Inference - Product Review

Goal: *Identify vendors with potential quality issues*

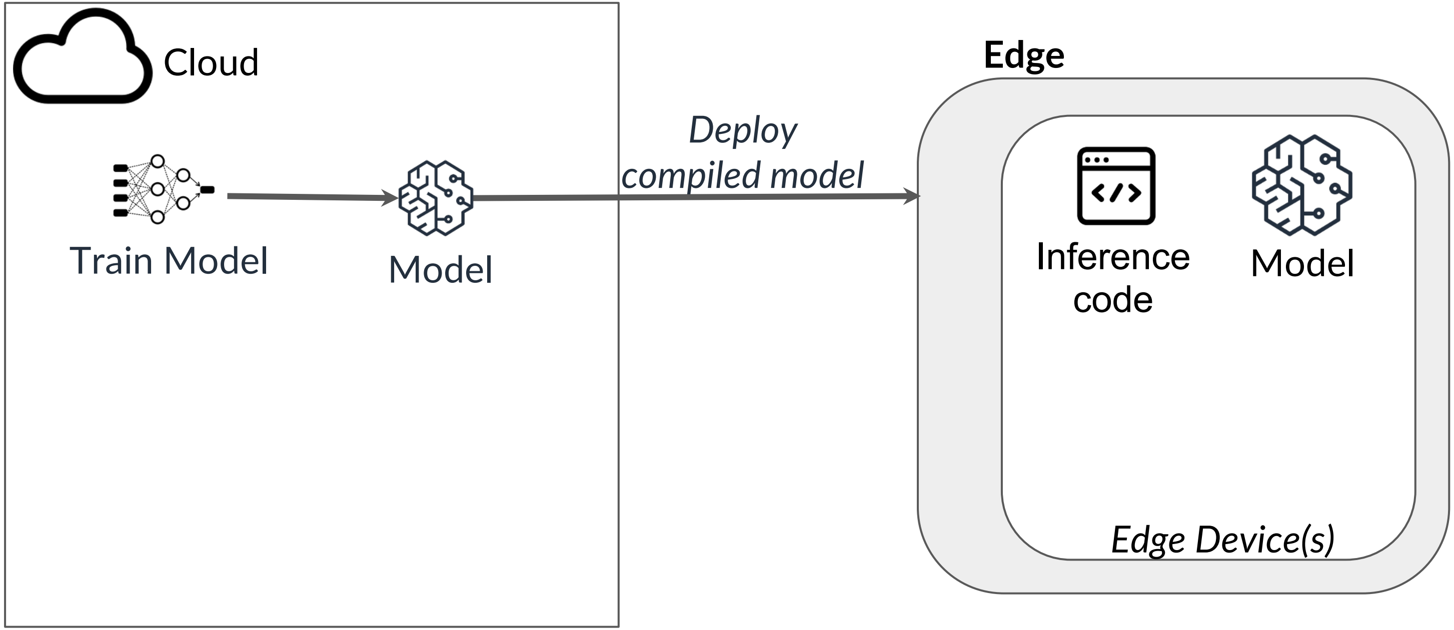


Deployment Options



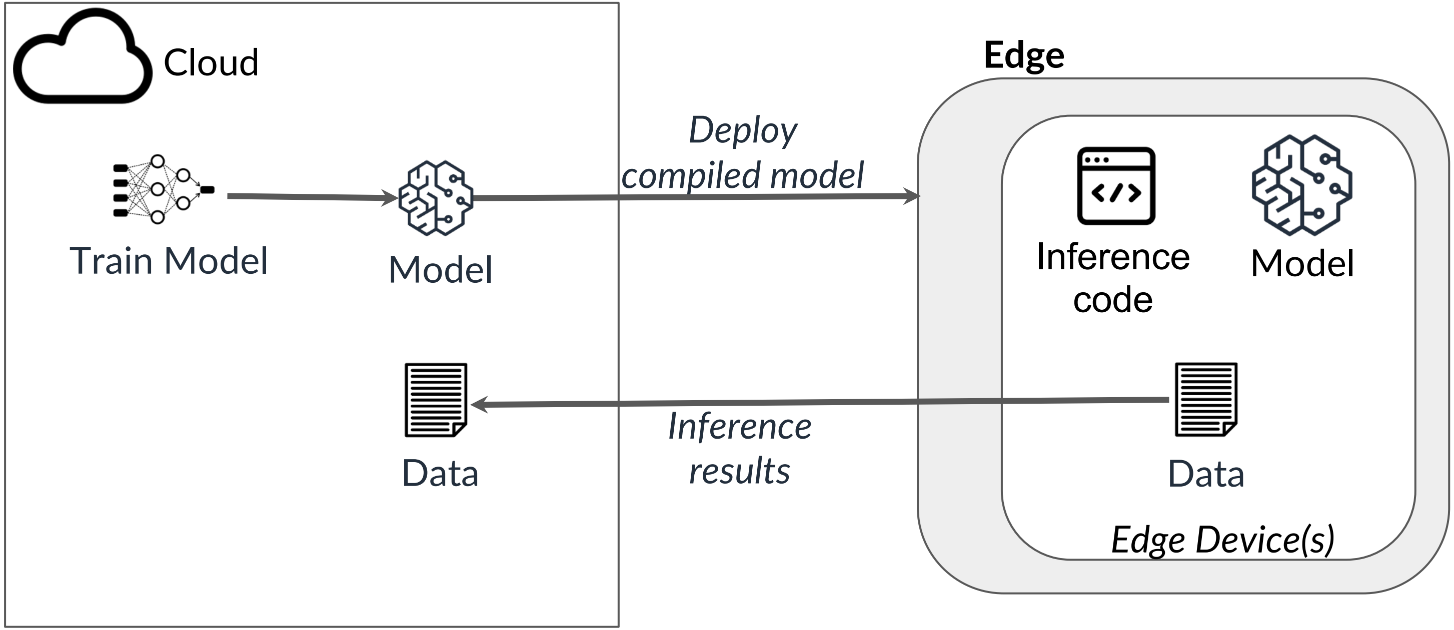
Deployment Options

Edge



Deployment Options

Edge



Deployment Options

Choose the deployment option that best fits the use case

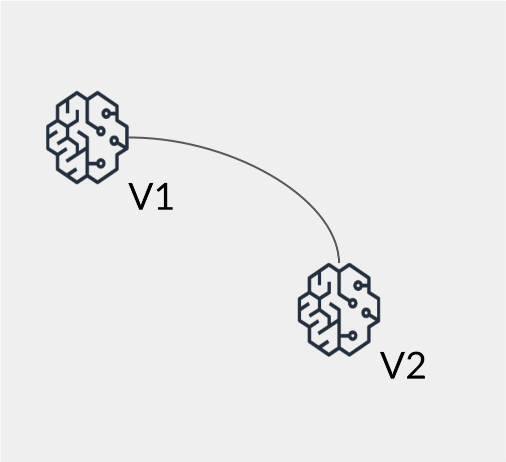
	Real-Time Inference	Batch Inference	Edge
When to use	Low latency real-time predictions (Ex. Interactive Recommenders)	Batch request & response prediction is acceptable for your use case (Ex. Forecasting)	Models need to deployed to edge devices (Ex. Limited connectivity, Internet of Things)
Cost	Persistent endpoint - pay for resources while endpoint is running	Transient environments - pay for resources for the duration of the batch job	Varies

Model Deployment

Deployment Strategies

Deployment Strategies

Strategies to deploy new and updated models



Goals:

- Minimize risk
- Minimize downtime
- Measure model performance

Deployment Strategies

Common Strategies to deploy new and updated models

Blue/Green	Shadow/ Challenger	Canary	A/B	Multi-Armed Bandits

Deployment Strategies

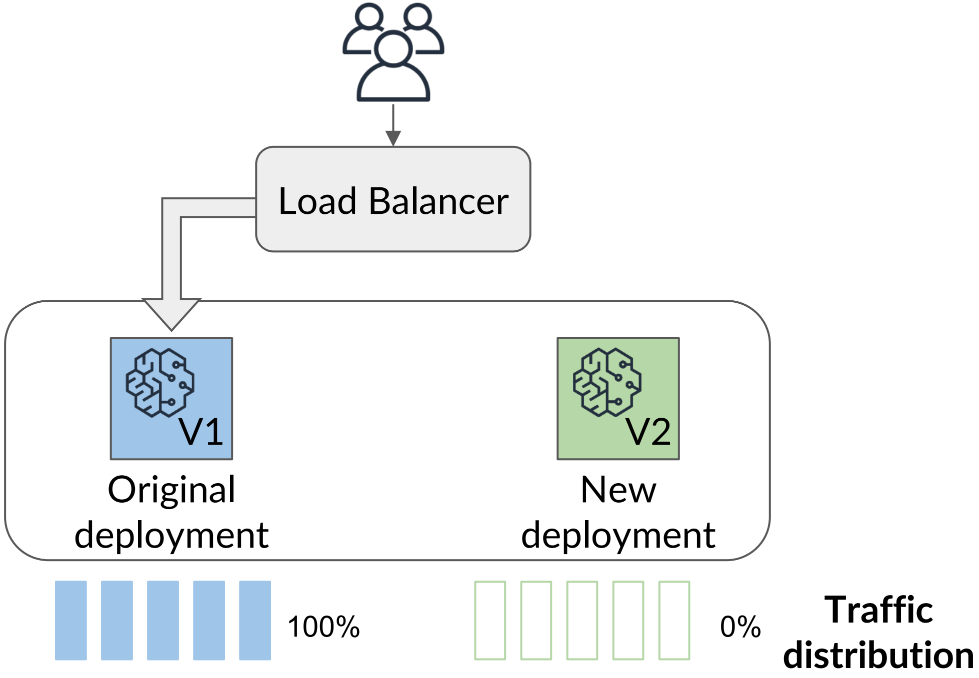
Common Strategies to deploy new and updated models

Blue/Green	Shadow/ Challenger	Canary	A/B	Multi-Armed Bandits

- Swap prediction request traffic
- Easy Rollback

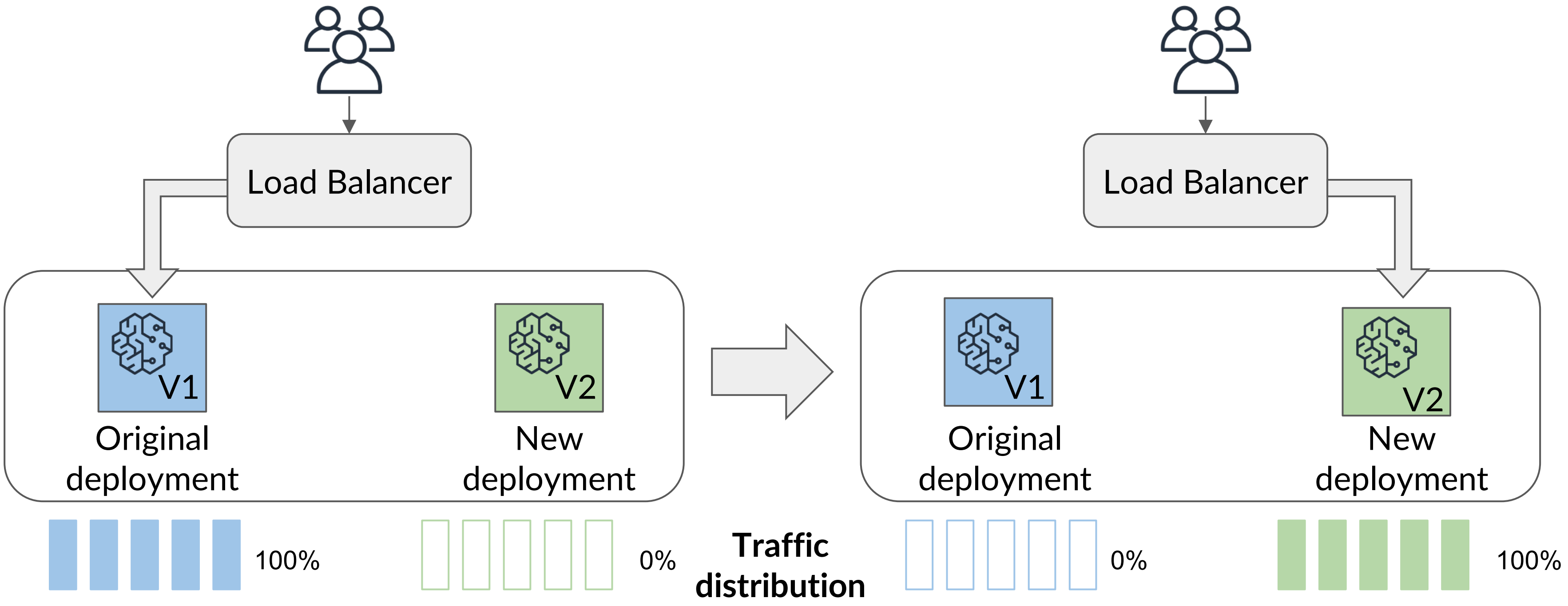
Deployment Strategies

Blue/Green: Shift all traffic to the new model



Deployment Strategies

Blue/Green: Shift all traffic to the new model



Deployment Strategies

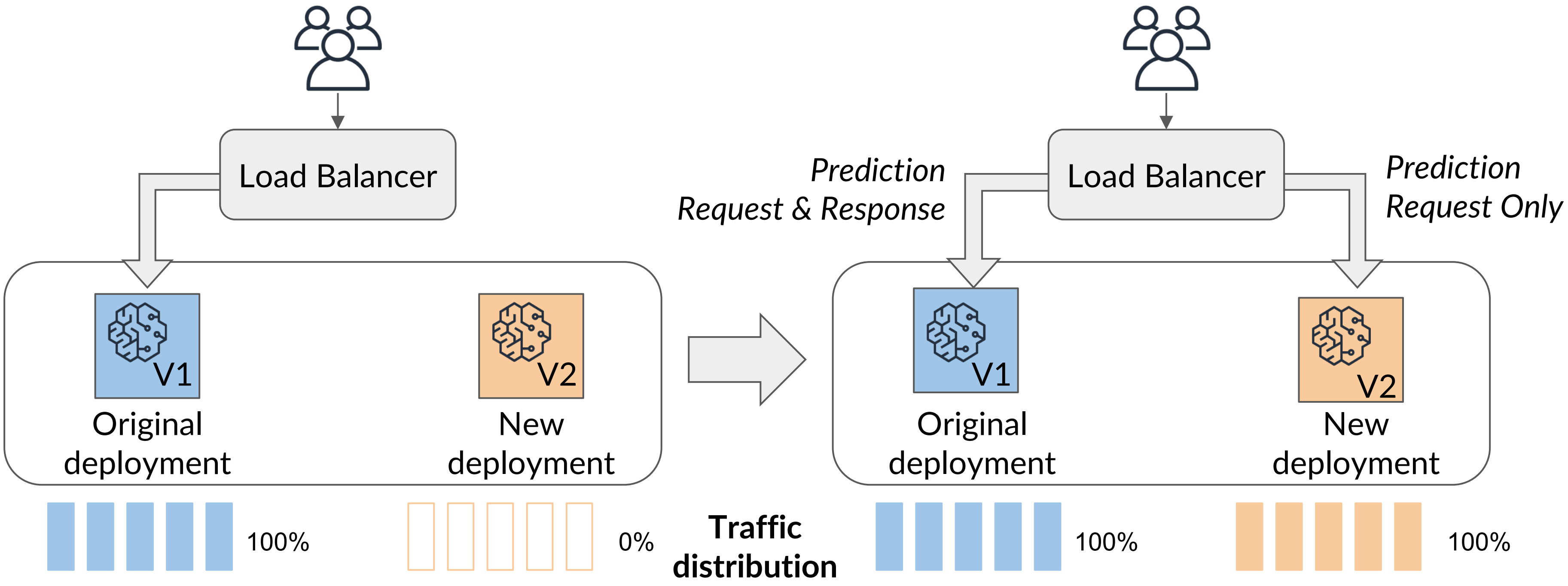
Common Strategies to deploy new and updated models

Blue/Green	Shadow/ Challenger	Canary	A/B	Multi-Armed Bandits

- Parallel prediction request traffic
- Validate new version without impact

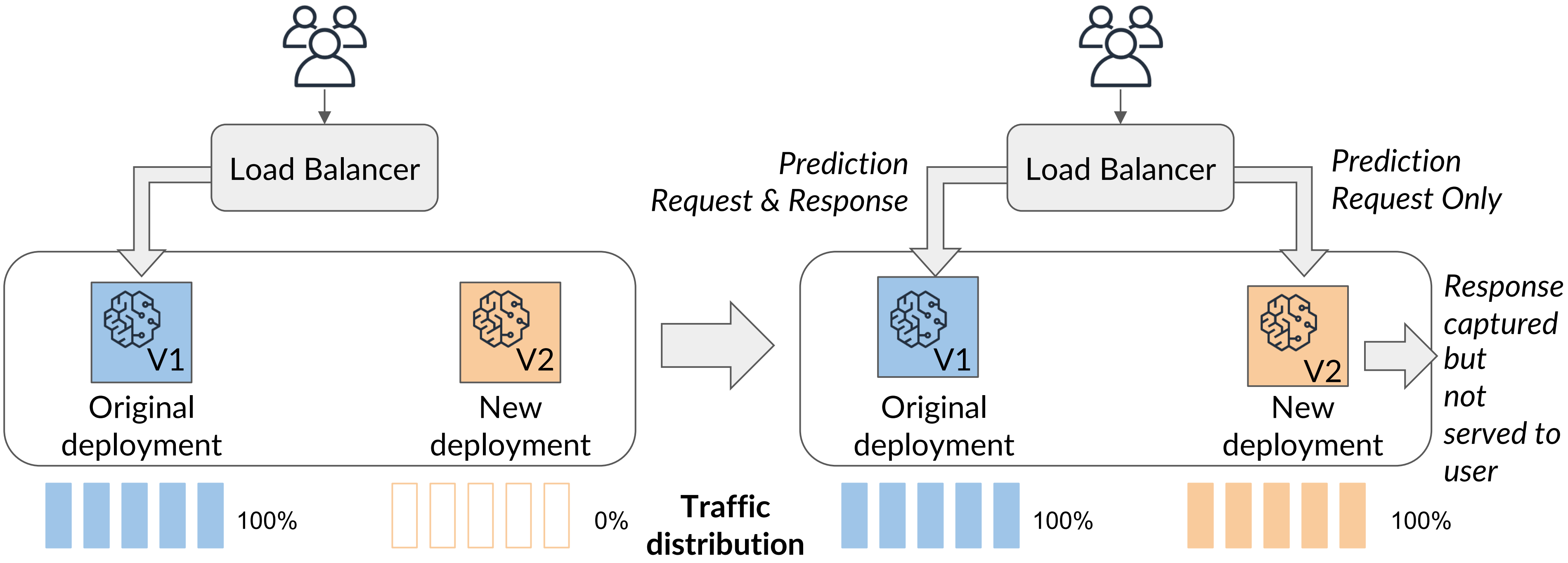
Deployment Strategies

Shadow/Challenger: Run multiple versions in parallel with one serving live traffic



Deployment Strategies

Shadow/Challenger: Run multiple versions in parallel with one serving live traffic



Deployment Strategies

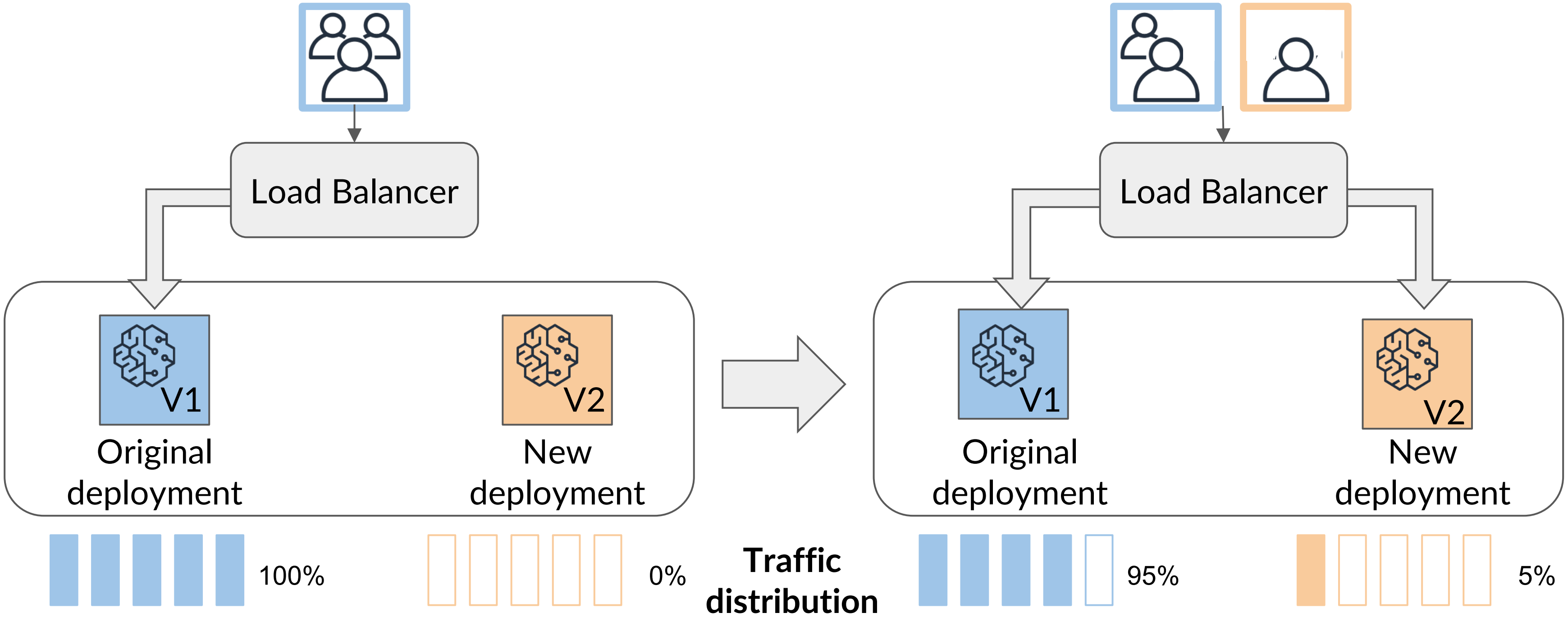
Common Strategies to deploy new and updated models

Blue/Green	Shadow/ Challenger	Canary	A/B	Multi-Armed Bandits

- Split traffic
- Target smaller specific users/groups
- Shorter validation cycles
- Minimize risk of low performing model

Deployment Strategies

Canary: Split traffic to compare model versions with target groups/users



Deployment Strategies

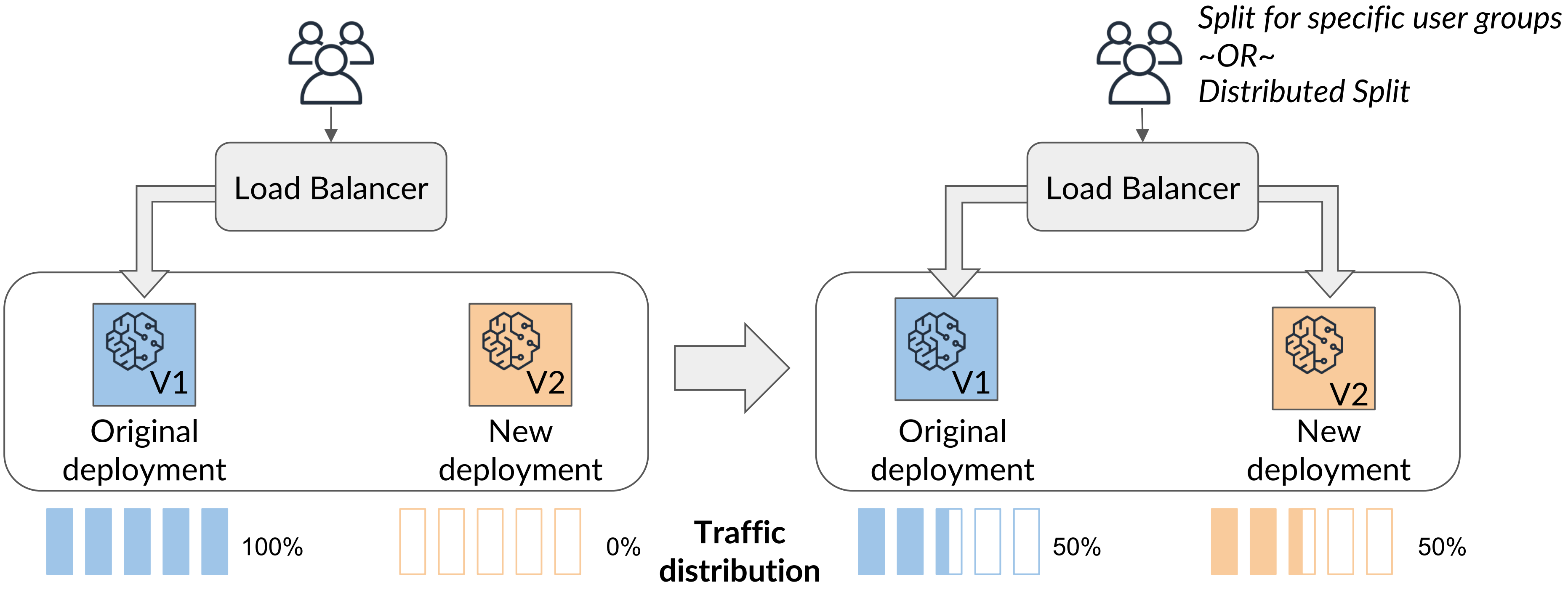
Common Strategies to deploy new and updated models

Blue/Green	Shadow/ Challenger	Canary	A/B	Multi-Armed Bandits

- Split traffic
- Target larger users/groups ~OR~ Distribute % of traffic
- Longer validation cycles
- Minimize risk of low performing model

Deployment Strategies

A/B: Split traffic to compare model versions



Deployment Strategies

Common Strategies to deploy new and updated models

- Blue/Green
- Shadow/ Challenger
- Canary
- A/B

Static Approaches

- Multi-Armed Bandits

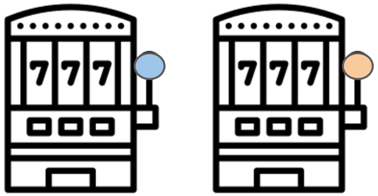
Deployment Strategies

Common Strategies to deploy new and updated models

Static Approaches				Dynamic Approach
Blue/Green	Shadow/ Challenger	Canary	A/B	Multi-Armed Bandits

Deployment Strategies

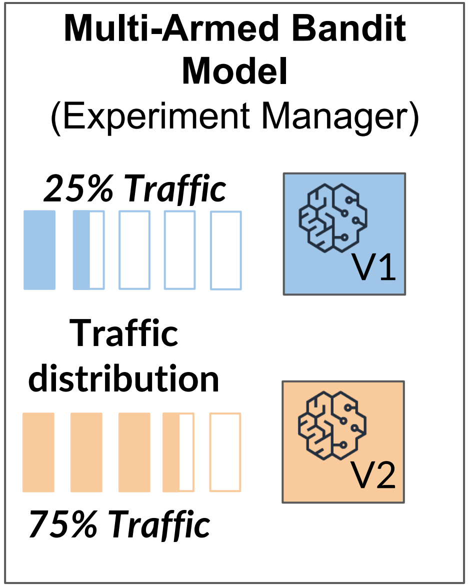
Multi-Armed Bandits (MABs)

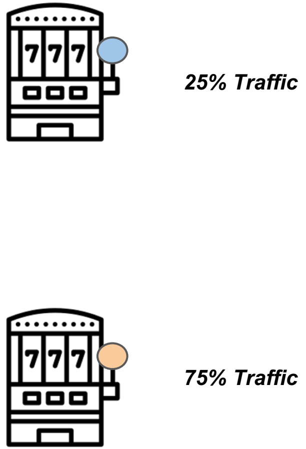


- **Dynamic testing method for model version using Reinforcement Learning**
- **Exploit & Explore**
    - **Exploit:** Reward the winning model with more traffic
    - **Explore:** Continue to send traffic to the non-winning model(s) in case behavior changes

Deployment Strategies

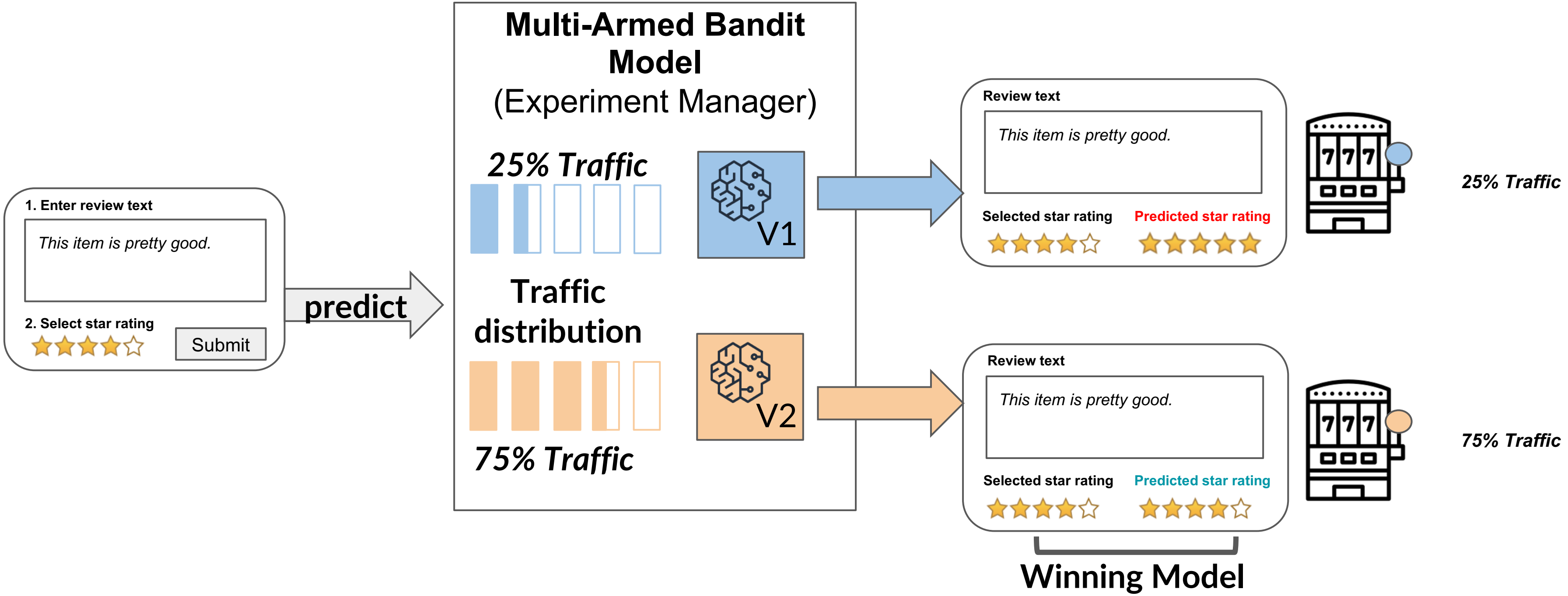
Multi-Armed Bandits: Dynamically shift traffic to the winning model





Deployment Strategies

Multi-Armed Bandits: Dynamically shift traffic to the winning model



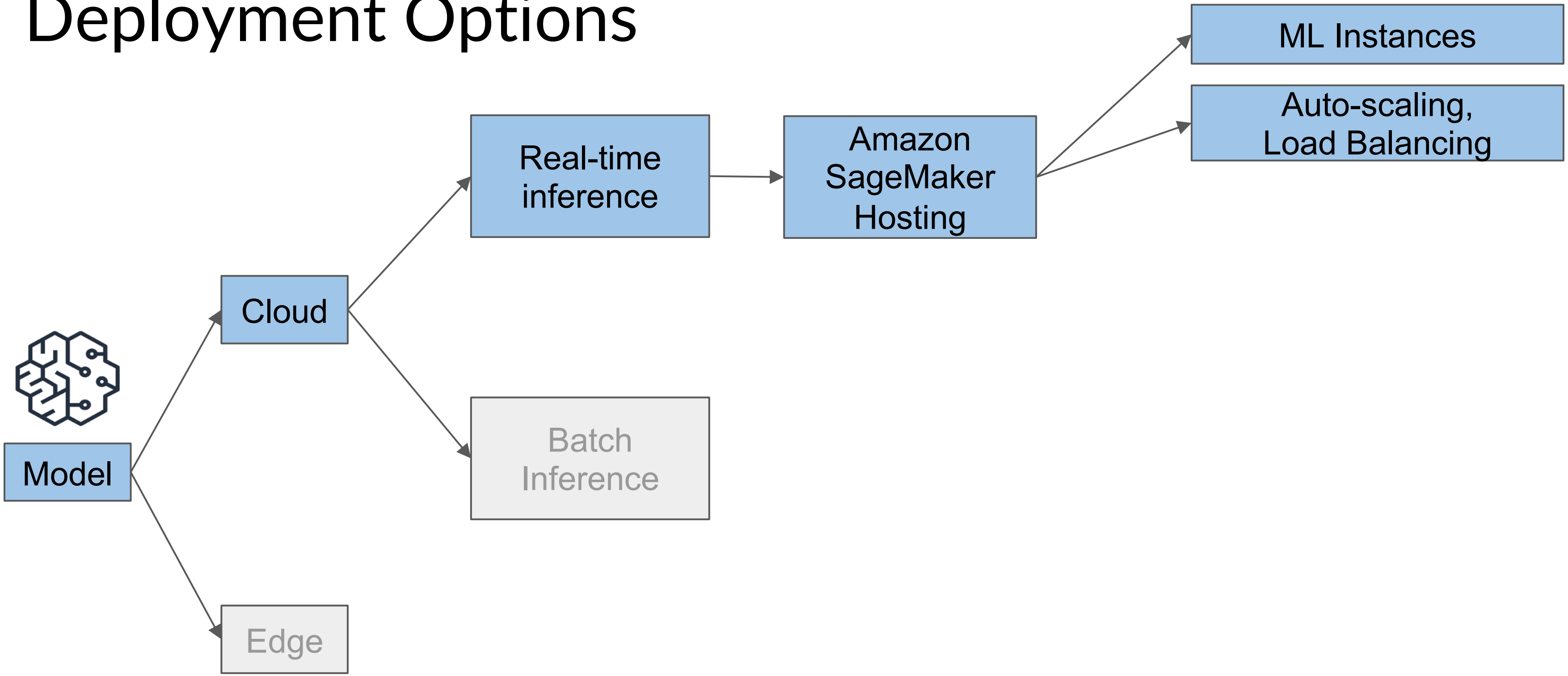
Amazon SageMaker Hosting

Real-Time Inference

Machine Learning Workflow

Ingest & Analyze	Prepare & Transform	Train & Tune	Deploy & Manage
Data exploration	Feature engineering	Automated ML	Model deployment
Bias detection	Feature store	Model train and tune	Automated pipelines
Amazon S3 & Amazon Athena	Amazon SageMaker Data Wrangler	Amazon SageMaker Autopilot	Amazon SageMaker Endpoints
AWS Glue	Amazon SageMaker Processing Jobs	Amazon SageMaker Training & Debugger	Amazon SageMaker Batch Transform
Amazon SageMaker Data Wrangler & Clarify	Amazon SageMaker Feature Store	Amazon SageMaker Hyperparameter Tuning	Amazon SageMaker Pipelines

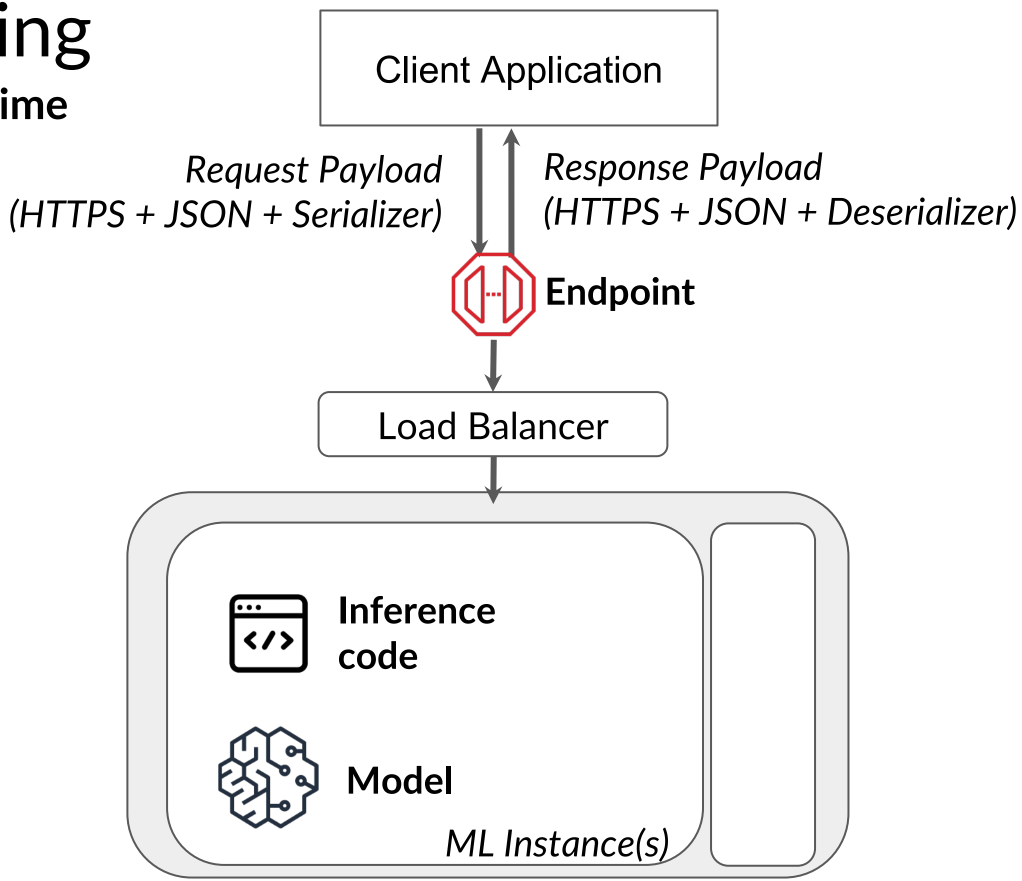
Deployment Options



Amazon SageMaker Hosting

Deploy models to serve predictions in real-time

- Optimized for **low latency** of model predictions
- **Example:** As product reviews are coming in through online channels, you want to predict the sentiment for immediate action

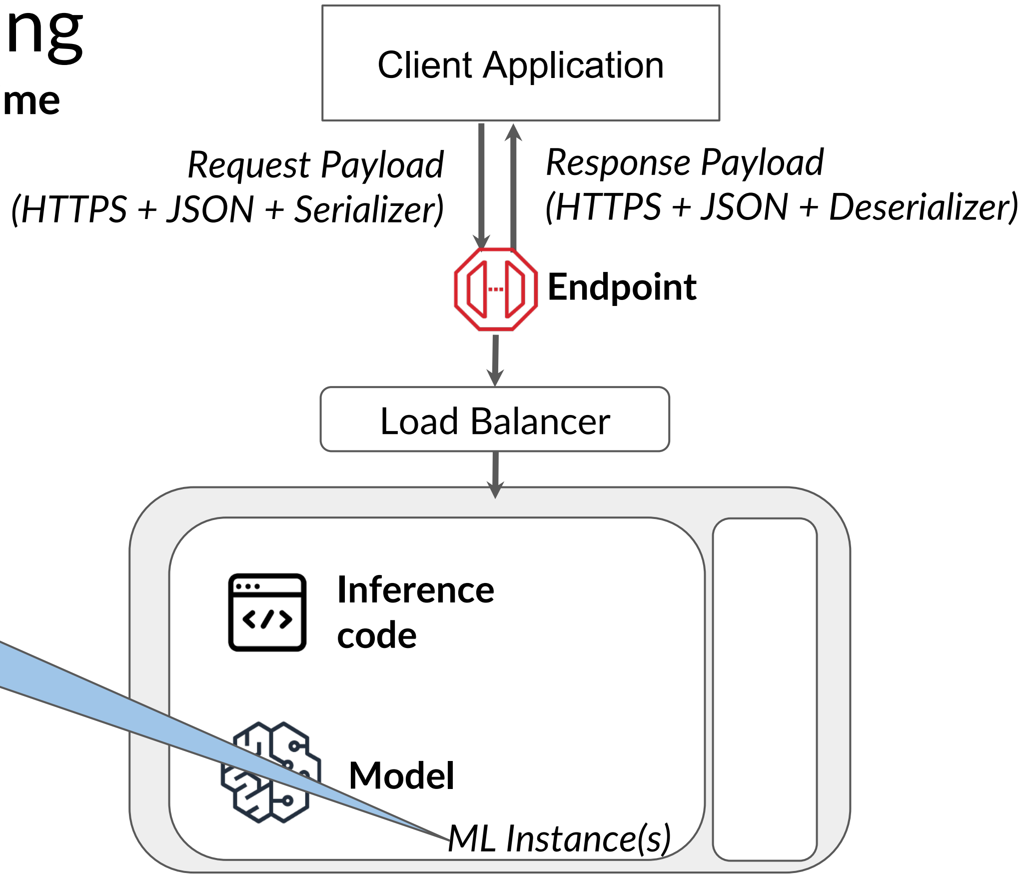


Amazon SageMaker Hosting

Deploy models to serve predictions in real-time

You choose:

- Instance Type
- Instance Size
- Number of Instances
- Autoscaling Options



Amazon SageMaker Hosting

Options to deploy models to serve predictions in real-time

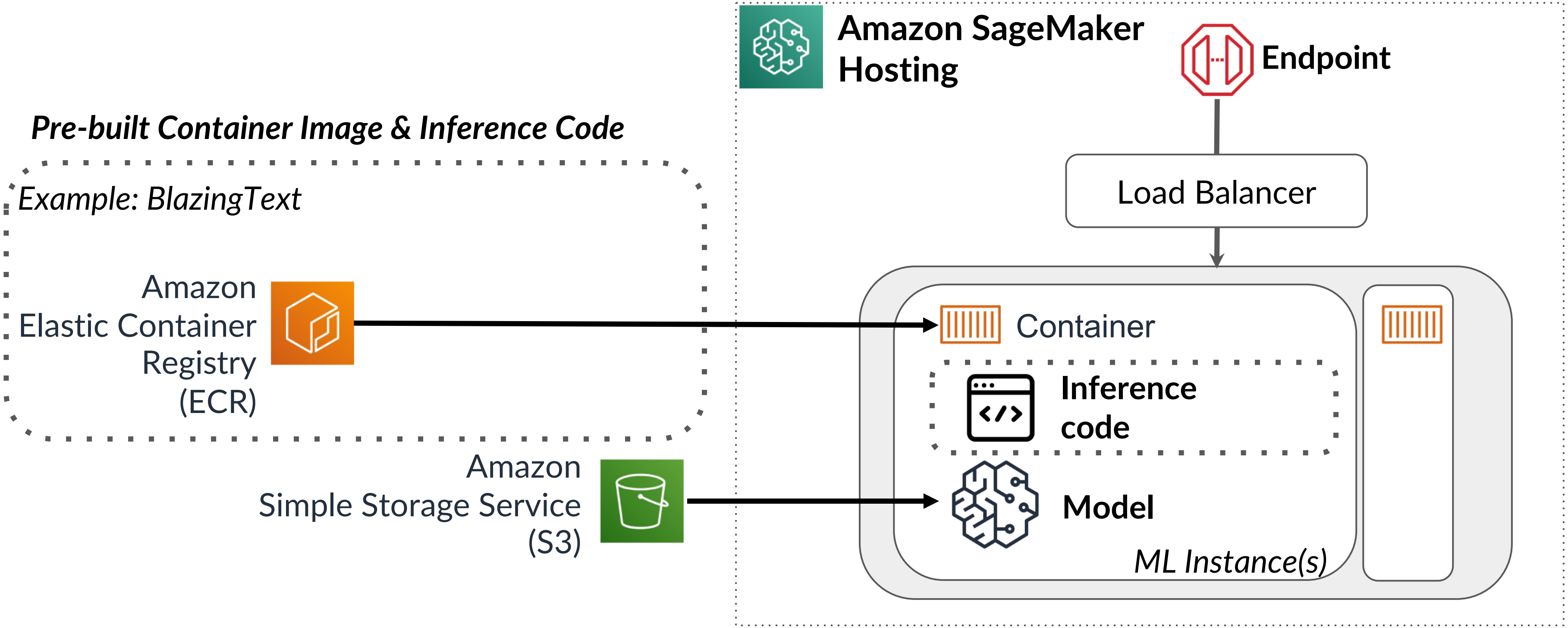
Amazon SageMaker	Amazon SageMaker	Amazon SageMaker
Built-in Algorithms	*Bring Your Own Script*	*Bring Your Own Container*

Less Code

More Customizable

Amazon SageMaker Hosting

Built-In Algorithm: Pre-built code & serving container



Amazon SageMaker Hosting

Options to deploy models to serve predictions in real-time

Amazon SageMaker

Built-in Algorithms

Amazon SageMaker

Bring Your Own Script

Amazon SageMaker

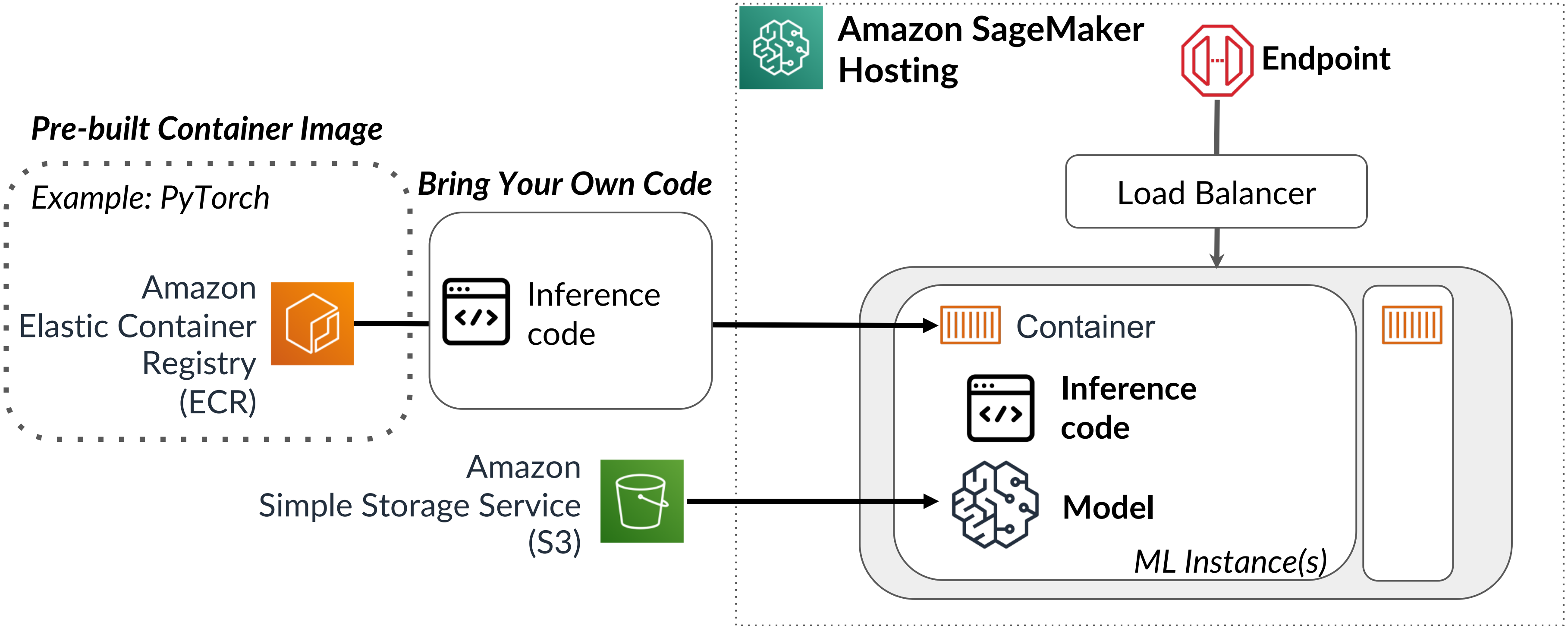
Bring Your Own Container

Less Code

More Customizable

Amazon SageMaker Hosting

Bring Your Own Script: Pre-built container & Bring your own code



Amazon SageMaker Hosting

Options to deploy models to serve predictions in real-time

Amazon SageMaker

Built-in Algorithms

Amazon SageMaker

Bring Your Own Script

Amazon SageMaker

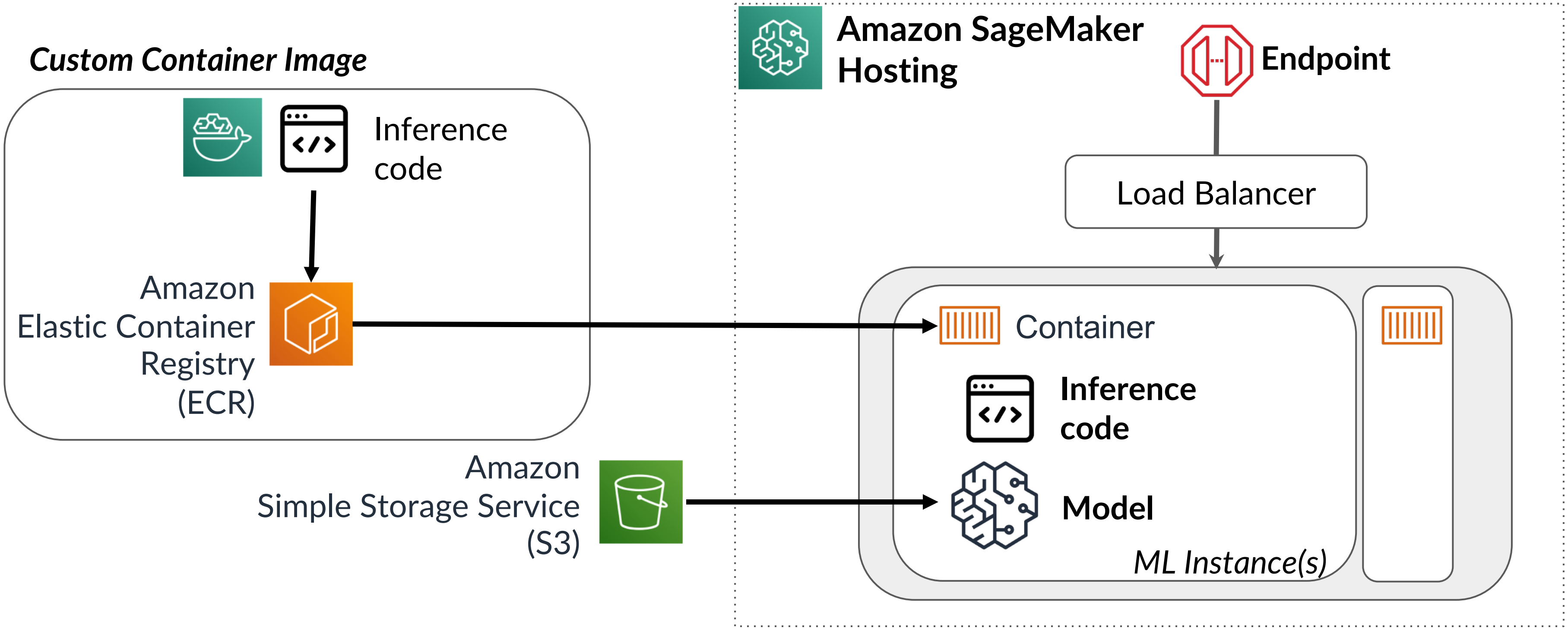
Bring Your Own Container

Less Code

More Customizable

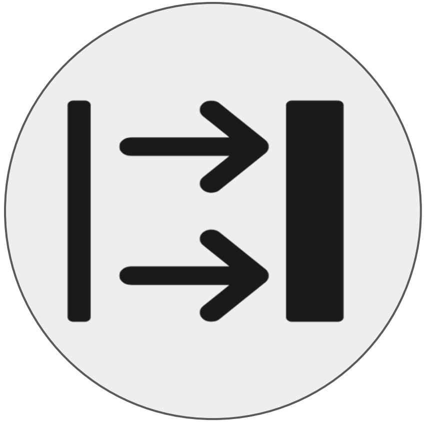
Amazon SageMaker Hosting

Bring Your Own Container: Bring your own code & custom container



Amazon SageMaker Hosting

Autoscaling Endpoints



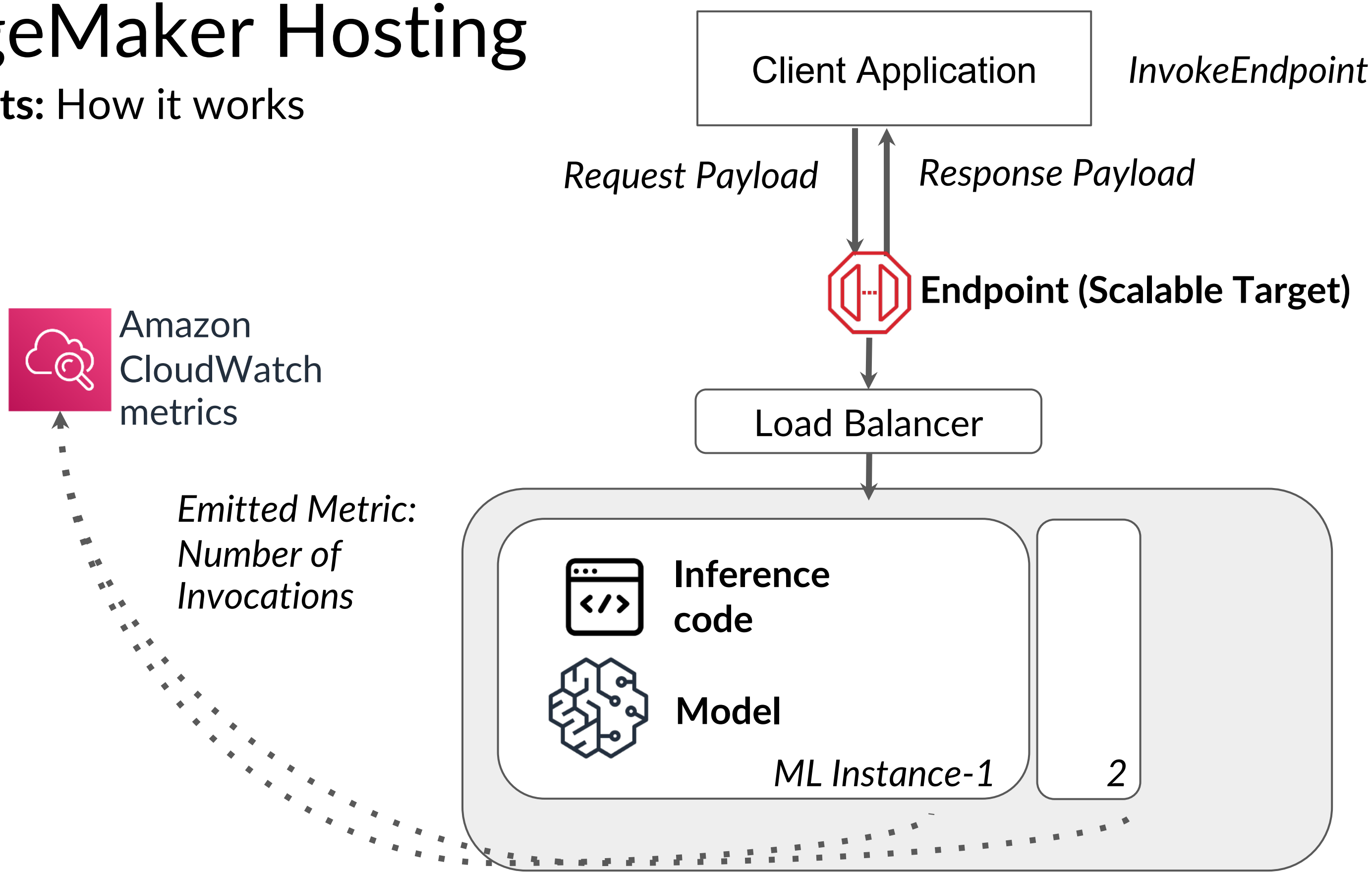
Why?

- Ensure you can meet the demands of your workload

- Cost optimization

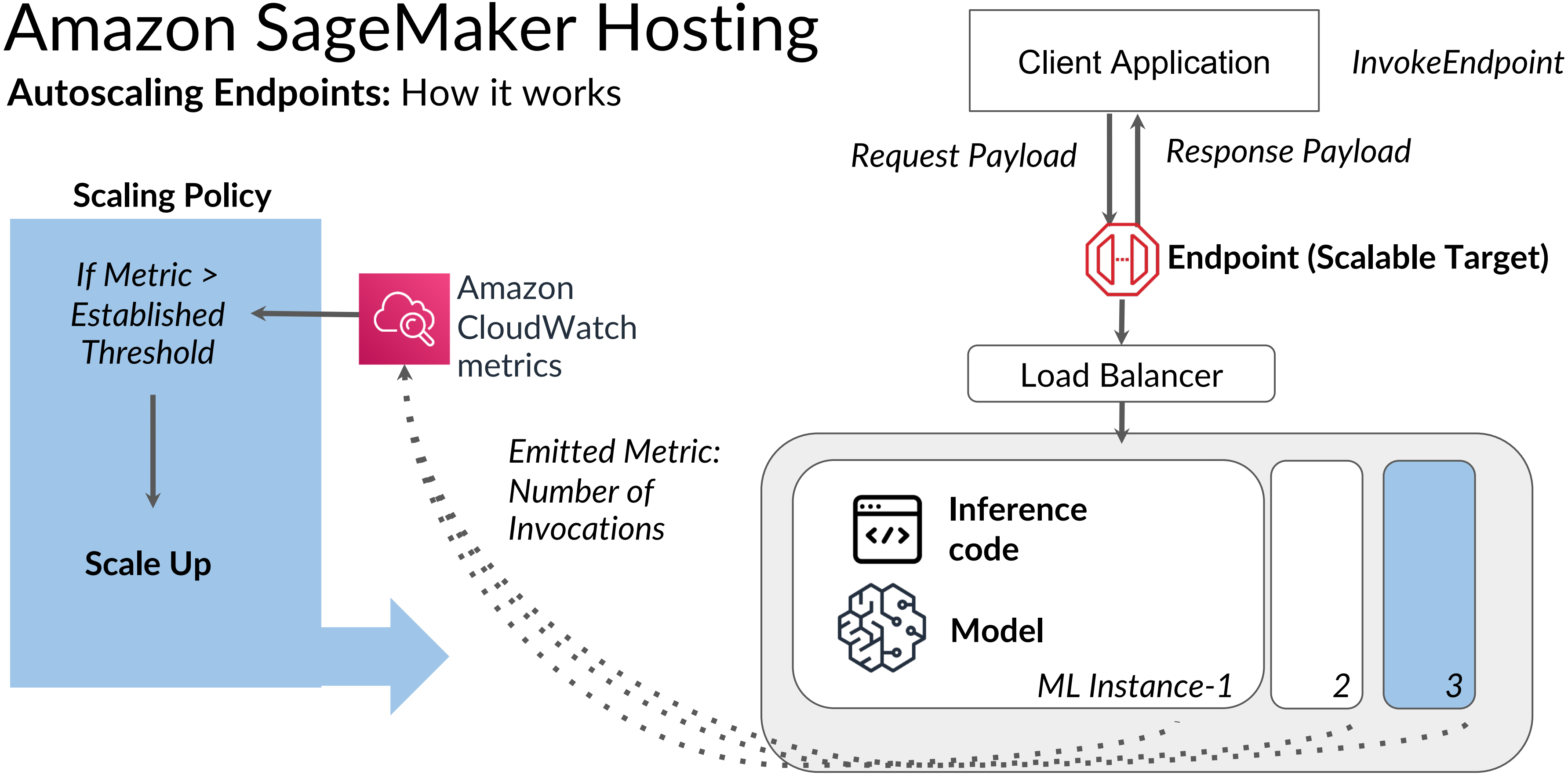
Amazon SageMaker Hosting

Autoscaling Endpoints: How it works



Amazon SageMaker Hosting

Autoscaling Endpoints: How it works



Autoscale Amazon SageMaker Endpoints

Register Scalable Target

```
autoscale.register_scalable_target(
         ServiceNamespace="sagemaker",
    ResourceId="endpoint/" + endpoint_name,
    ScalableDimension="sagemaker:variant:DesiredInstanceCount",
    MinCapacity=1,
    MaxCapacity=2,
    RoleARN=role,
    SuspendedState={
        "DynamicScalingInSuspended": False,
        "DynamicScalingOutSuspended": False,
        "ScheduledScalingSuspended": False,
    })
```

Autoscale Amazon SageMaker Endpoints

Register Scalable Target → **Define Scaling Policy**

```
scaling_policy = {
          "TargetValue": 2.0,
          "PredefinedMetricSpecification": {
          "PredefinedMetricType": "SageMakerVariantInvocationsPerInstance",
          },
          "ScaleOutCooldown": 60,
          "ScaleInCooldown": 300,
          },
}
```

Scaling Metric

Wait time, in seconds, before beginning another **scale out** activity after last one completes

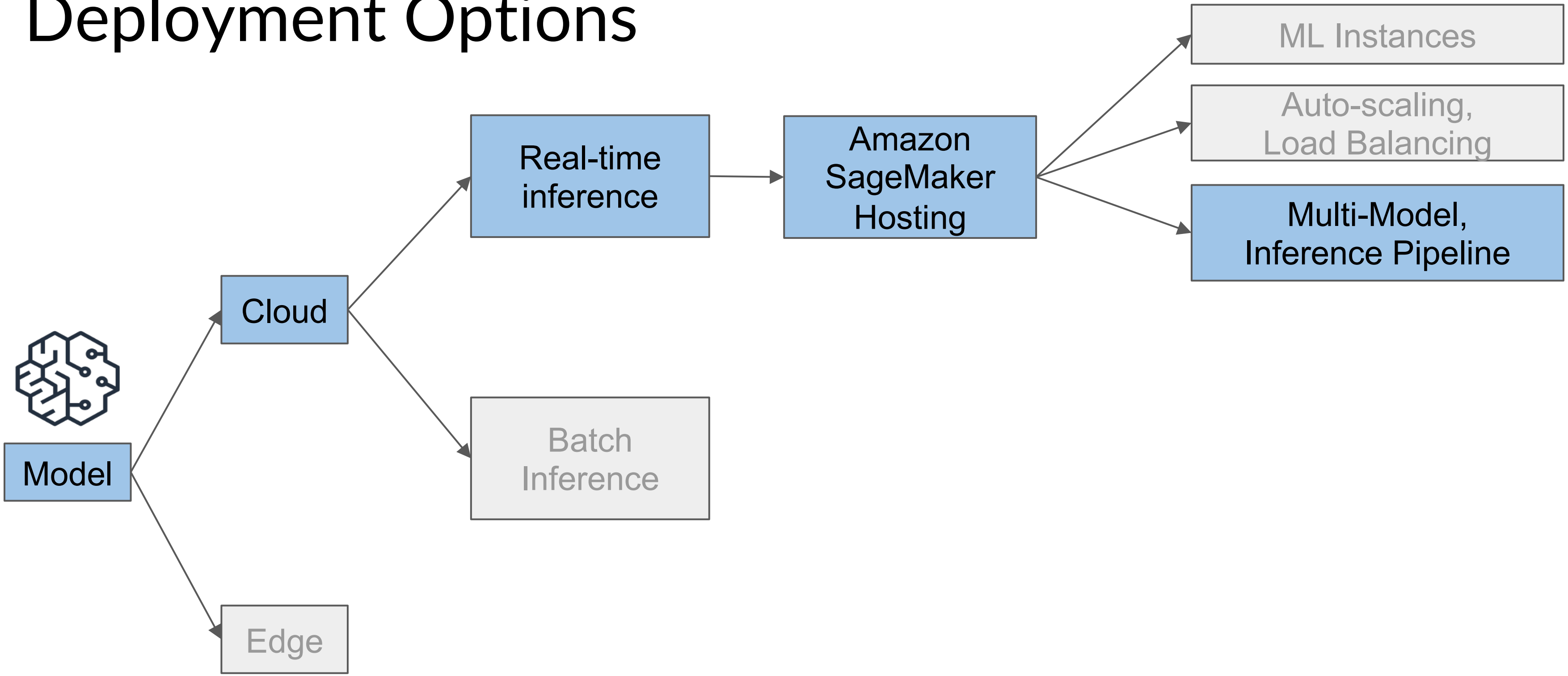
Wait time, in seconds, before beginning another **scale in** activity after last one completes

Autoscale Amazon SageMaker Endpoints

Register Scalable Target → Define Scaling Policy → **Apply Scaling Policy**

```
autoscale.put_scaling_policy(
        PolicyName=...,
        ServiceNamespace="sagemaker",
        ResourceId="endpoint/" + endpoint_name,
        ScalableDimension="sagemaker:variant:DesiredInstanceCount",
        PolicyType="TargetTrackingScaling",
        TargetTrackingScalingPolicyConfiguration=scaling_policy)
```

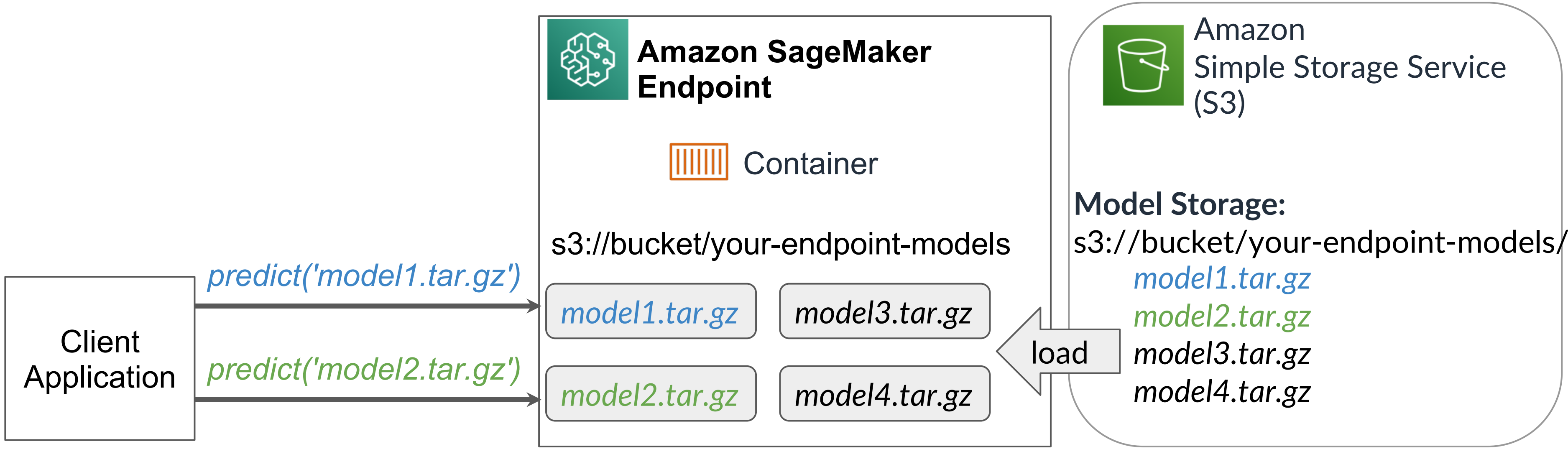
Deployment Options



Advanced Deployment Options

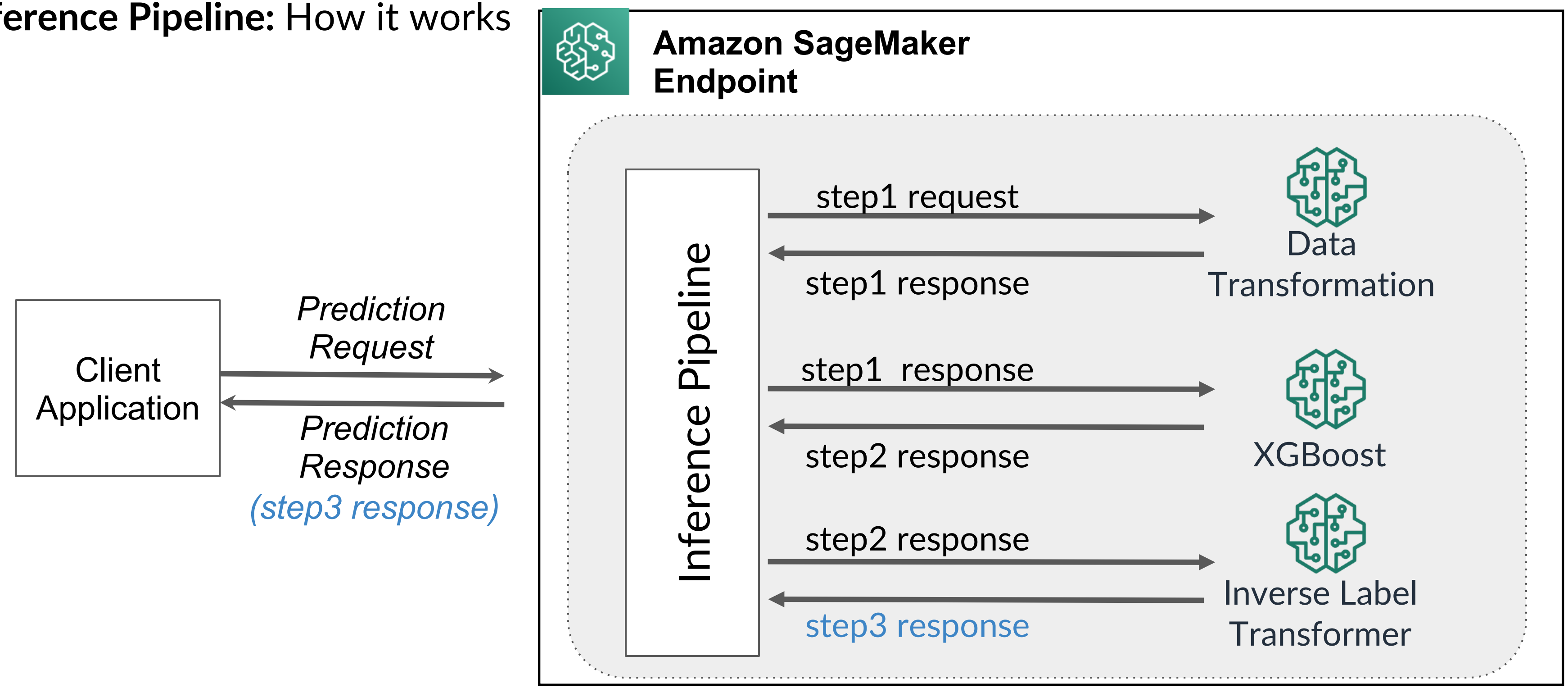
Multi-Model Endpoints: How it works

Deploy Multiple Models to a Single Endpoint



Advanced Deployment Options

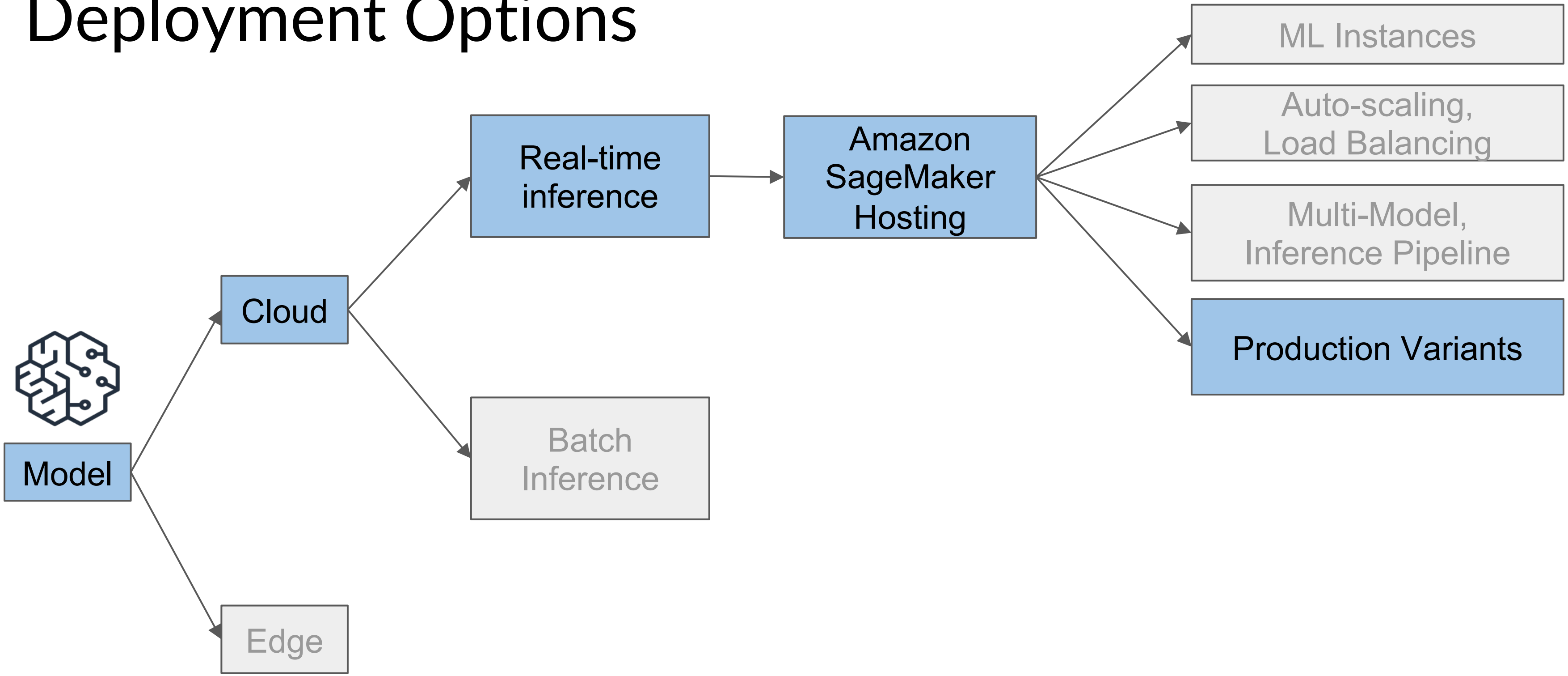
Inference Pipeline: How it works



Amazon SageMaker Hosting

Real-Time Inference
Production Variants

Deployment Options



Amazon SageMaker Production Variants

What is a Production Variant?

SageMaker
Model

Configuration Example(s):

- *Amazon S3 model artifact*
- *Inference image(s)*
- *Execution AWS IAM Role*
- ***Model name***

+

Hosting Resources
Configuration

Example:

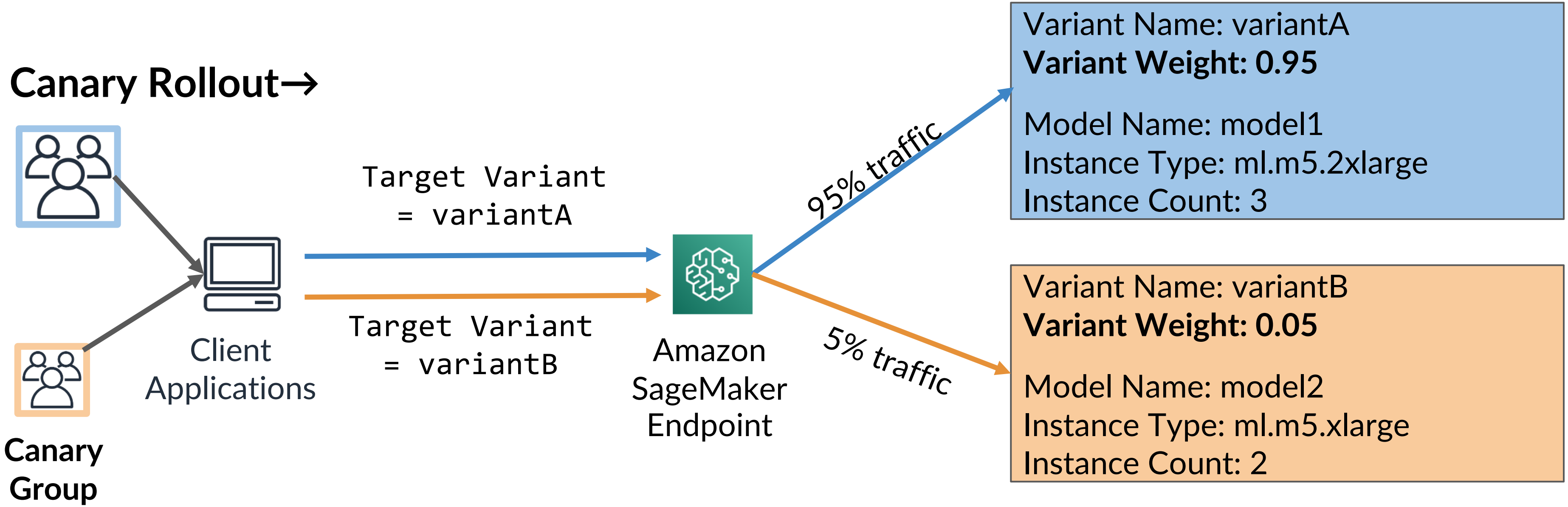
- *Number of instances*
- *Instance type*
- ***Model name***
- *Variant name*
- *Variant weight*

=

Production
Variant

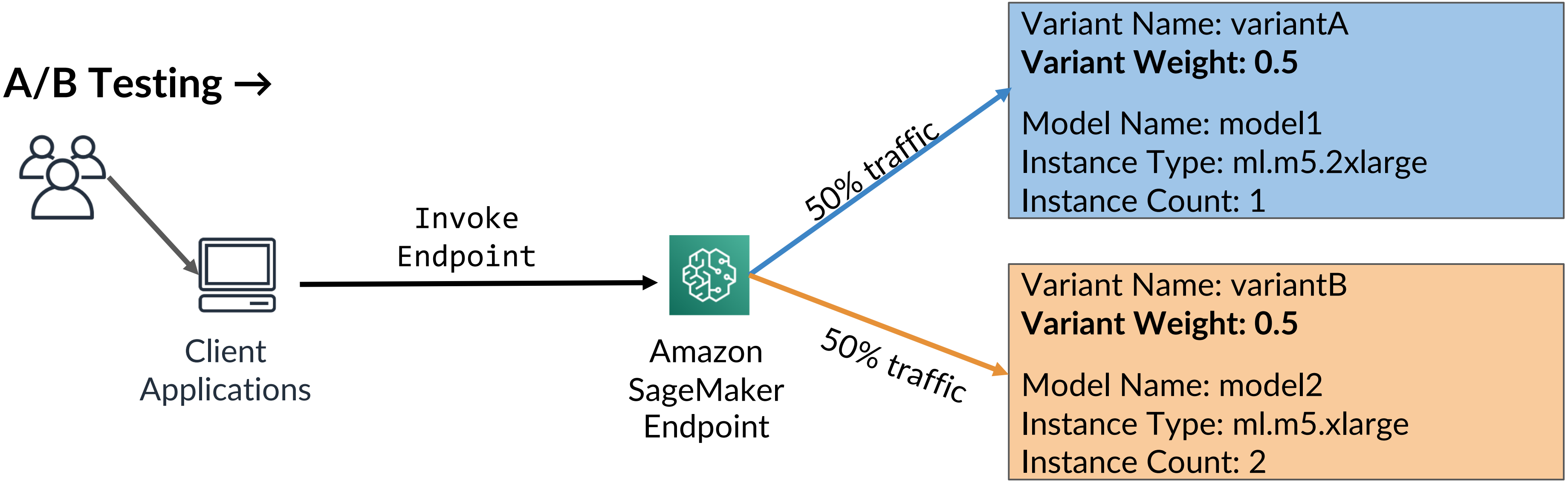
Amazon SageMaker Production Variants

Using Production Variants for a Canary Rollout



Amazon SageMaker Production Variants

Using Production Variants for A/B Testing



Amazon SageMaker Hosting

Using Production Variants for A/B Testing with Bring-Your-Own Script

Amazon SageMaker	Amazon SageMaker	Amazon SageMaker
Built-in Algorithms	*Bring Your Own Script*	*Bring Your Own Container*

Less Code

More Customizable

Using Production Variants for A/B Testing

A/B Testing with PyTorch Bring-Your-Own Script

Construct Docker Image URI

```
import sagemaker

inference_image_uri = sagemaker.image_uris.retrieve(
     framework=..., # PyTorch, TensorFlow, etc...
     version='1.6.0',
     instance_type='ml.m5.xlarge',
     py_version='py3',
     image_scope='inference'
)
```

Using Production Variants for A/B Testing

A/B Testing with PyTorch Bring-Your-Own Script

Construct Docker Image URI → Create SageMaker Models

```
sm.create_model(
        name=model_name_a,
        ...
)
```

```
sm.create_model(
        name=model_name_b,
        ...
)
```

SageMaker Model A

SageMaker Model B

Using Production Variants for A/B Testing

A/B Testing with PyTorch Bring-Your-Own Script

Construct Docker Image URI → Create SageMaker Models → Create Production Variants

```
from sagemaker.session \
    import production_variant

variantA = production_variant(
         model_name=...,
     instance_type=...,
         initial_instance_count=1,
         variant_name='VariantA',
         initial_weight=50,
)
```

```
from sagemaker.session \
    import production_variant

variantB = production_variant(
         model_name=...,
     instance_type=...,
         initial_instance_count=1,
         variant_name='VariantB',
         initial_weight=50,
)
```

Using Production Variants for A/B Testing

A/B Testing with PyTorch Bring-Your-Own Script

Construct Docker Image URI → Create SageMaker Models → Create Production Variants → **Create Endpoint Configuration**

```
endpoint_config = sm.create_endpoint_config(
          EndpointConfigName=...,
     ProductionVariants=[variantA, variantB]
)
```

Using Production Variants for A/B Testing

A/B Testing with PyTorch Bring-Your-Own Script

Construct Docker Image URI → Create SageMaker Models → Create Production Variants → Create Endpoint Configuration → Create Endpoint

```
endpoint_response = sm.create_endpoint(EndpointName=...,
                                       EndpointConfigName=...)
```

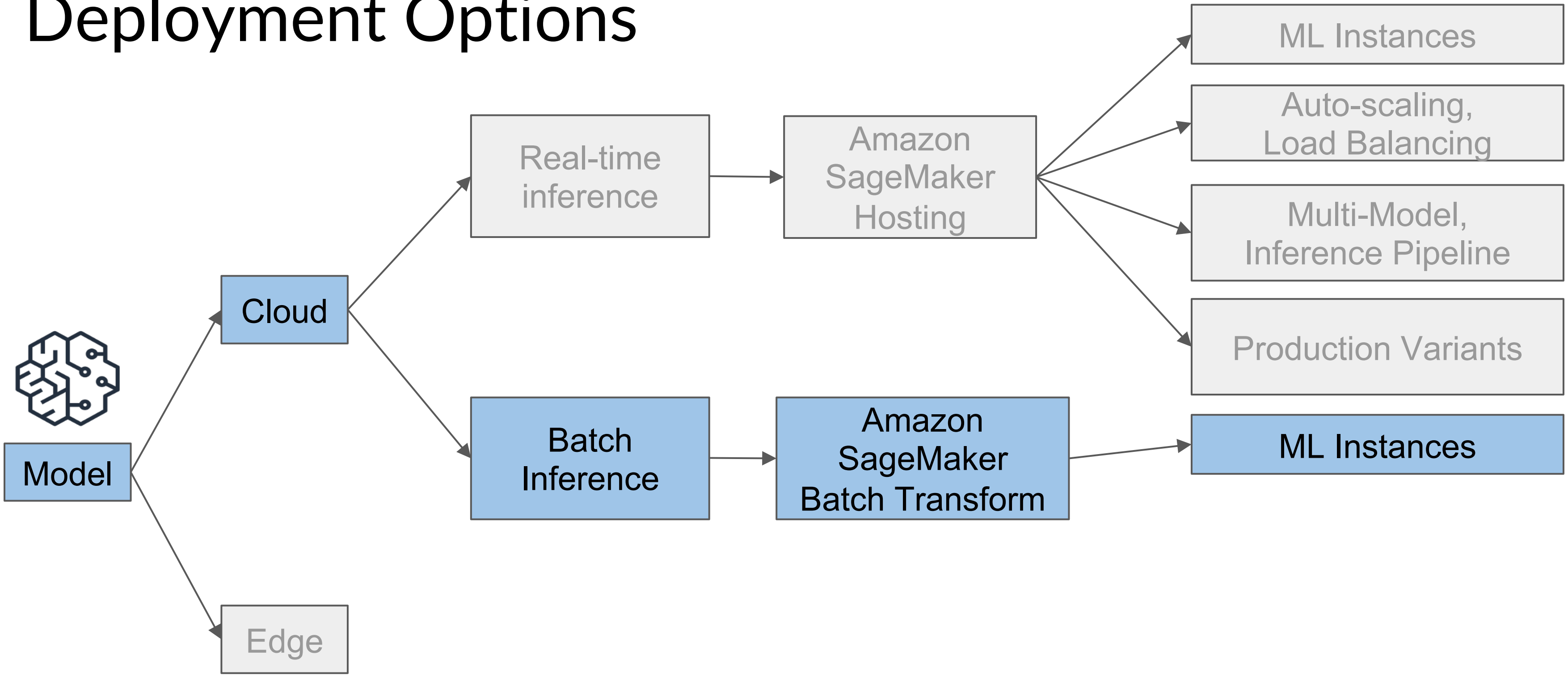
Amazon SageMaker Batch Transform

Batch Inference

Machine Learning Workflow

Ingest & Analyze	Prepare & Transform	Train & Tune	Deploy & Manage
Data exploration	Feature engineering	Automated ML	Model deployment
Bias detection	Feature store	Model train and tune	Automated pipelines
Amazon S3 & Amazon Athena	Amazon SageMaker Data Wrangler	Amazon SageMaker Autopilot	Amazon SageMaker Endpoints
AWS Glue	Amazon SageMaker Processing Jobs	Amazon SageMaker Training & Debugger	Amazon SageMaker Batch Transform
Amazon SageMaker Data Wrangler & Clarify	Amazon SageMaker Feature Store	Amazon SageMaker Hyperparameter Tuning	Amazon SageMaker Pipelines

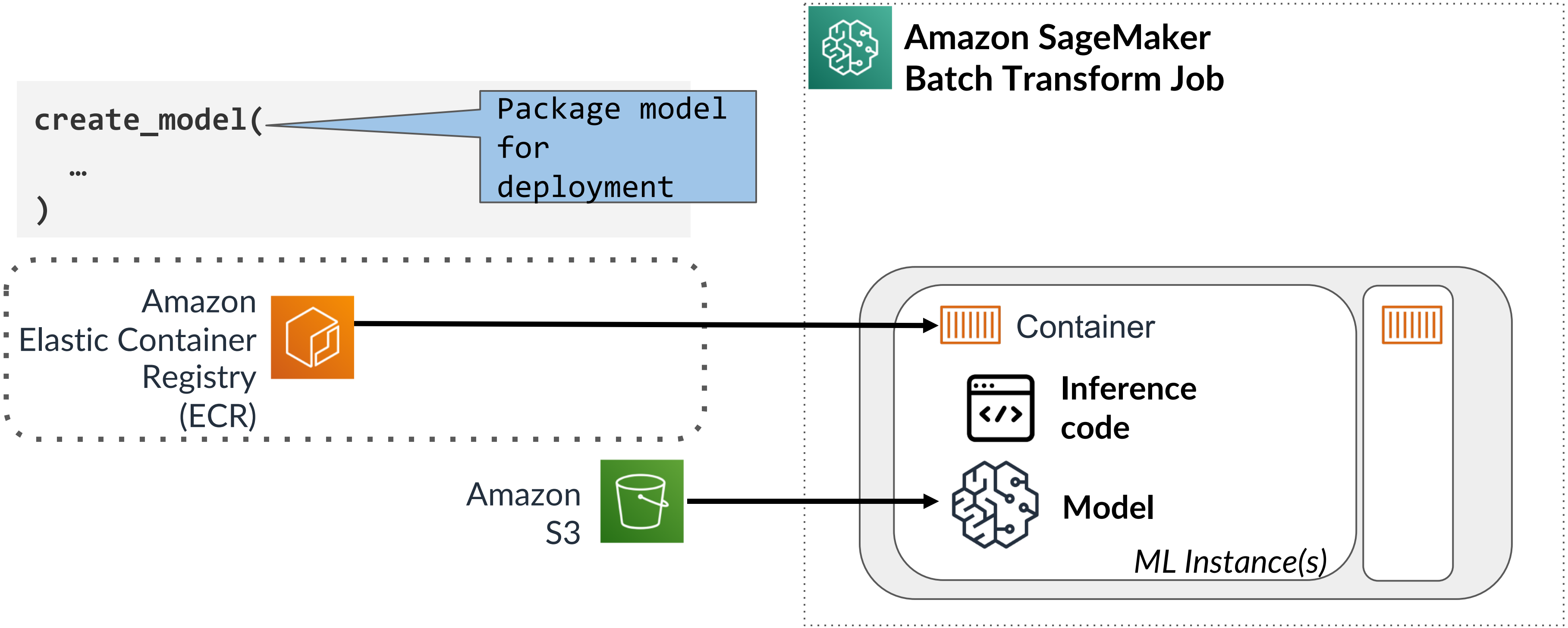
Deployment Options



Amazon SageMaker Batch Transform

Deploy Model For Batch Inference

```
create_model(
  …
)
```



Amazon SageMaker Batch Transform

Run Batch Transform Job For Batch Inference

```
sm_transformer = Transformer(
    ...
)
```

Configure Batch Transform Job

Amazon SageMaker Batch Transform Job

Identify Configuration →

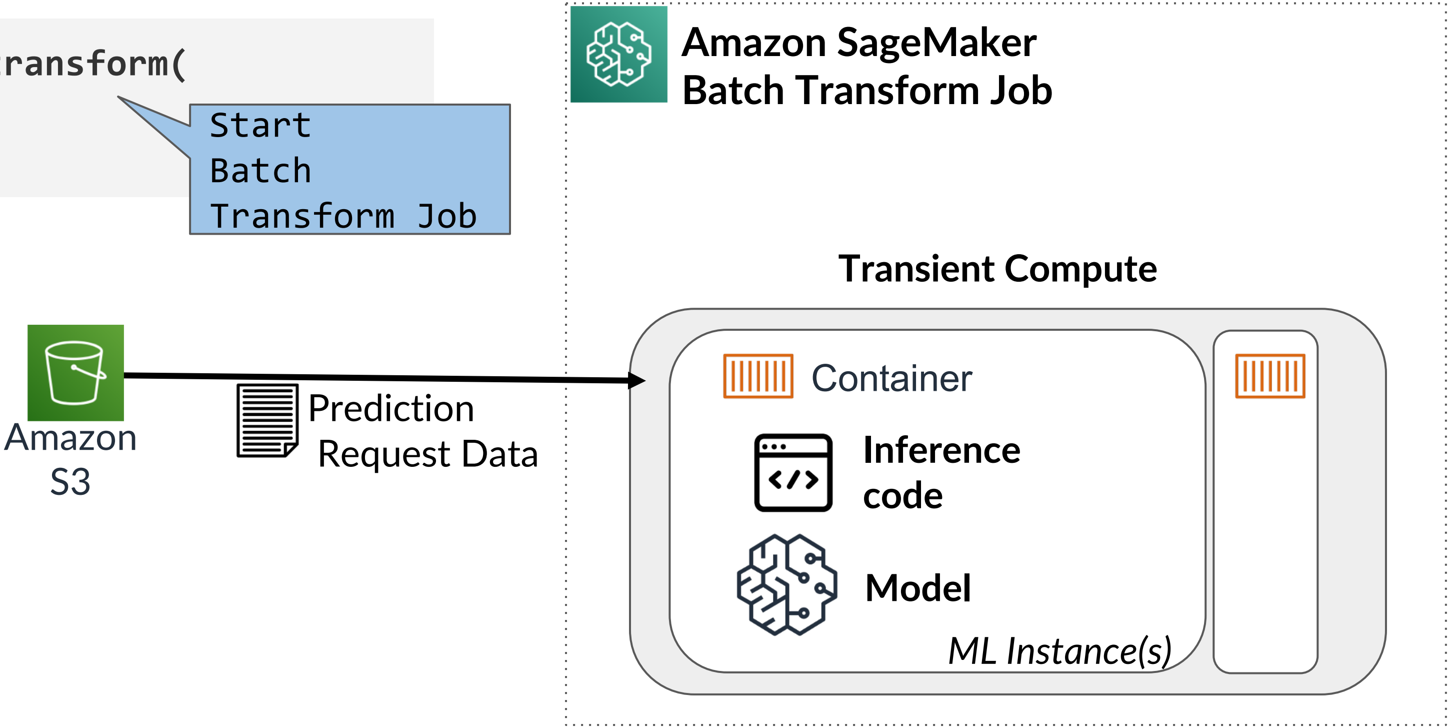
- Instance type
- Instance count
- Model name
- S3 output path
- ...

Amazon SageMaker Batch Transform

Run Batch Transform Job For Batch Inference

```
sm_transformer.transform(
    ...
)
```

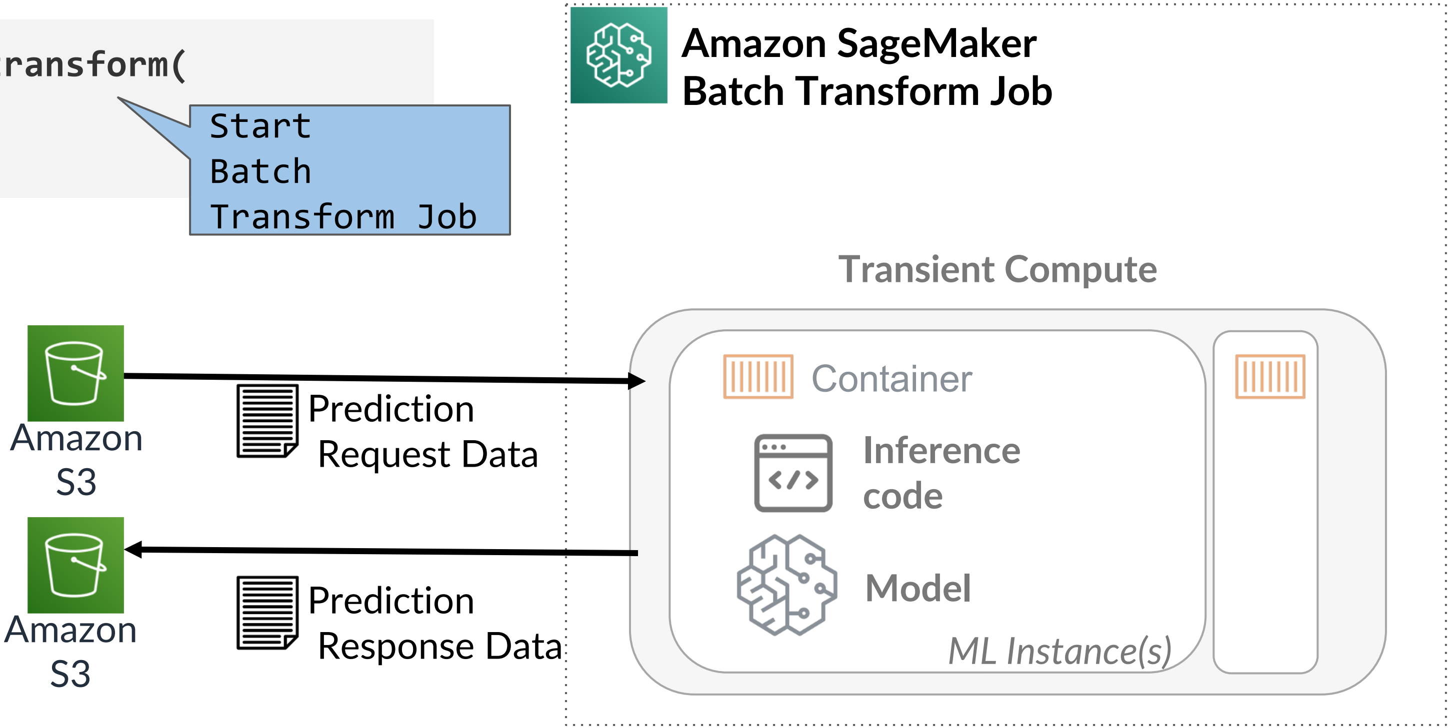
Start Batch Transform Job



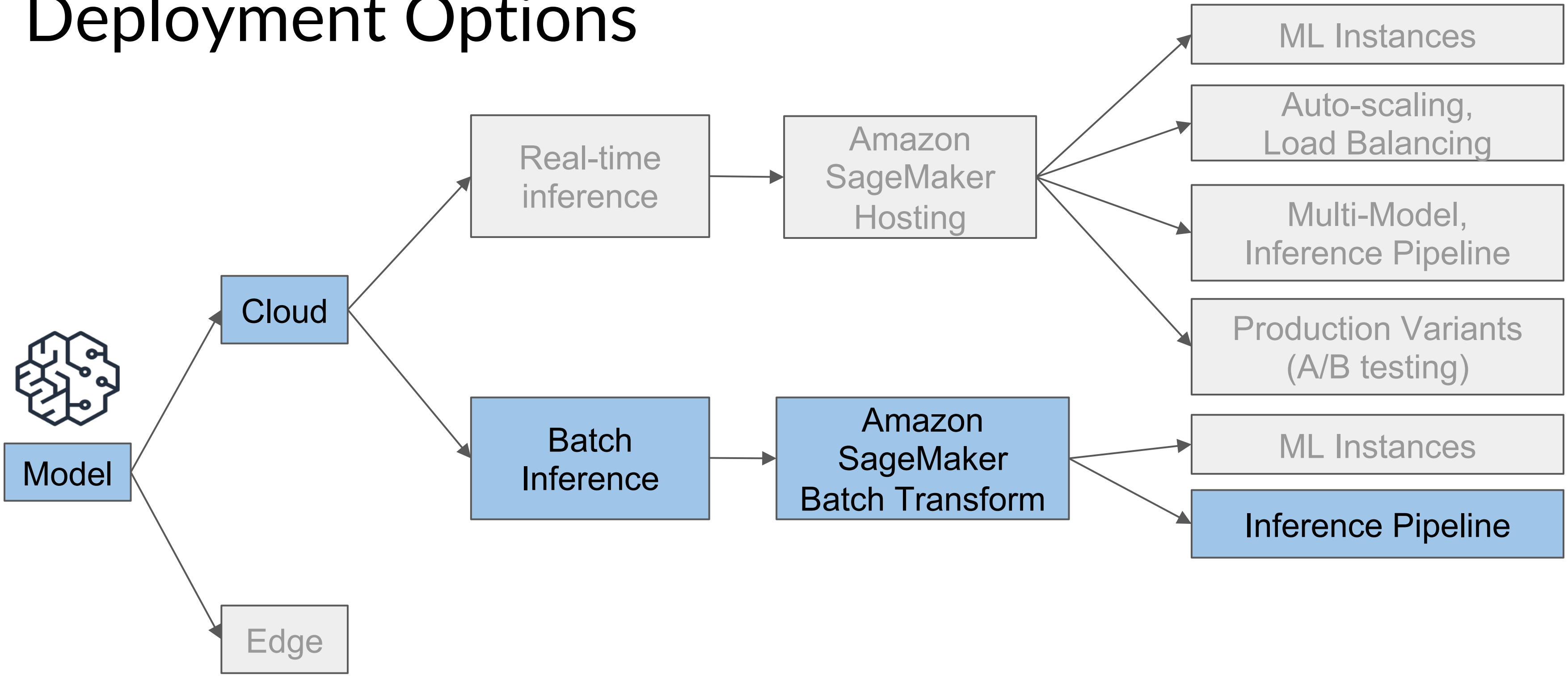
Amazon SageMaker Batch Transform

Run Batch Transform Job For Batch Inference

```
sm_transformer.transform(
    ...
)
```

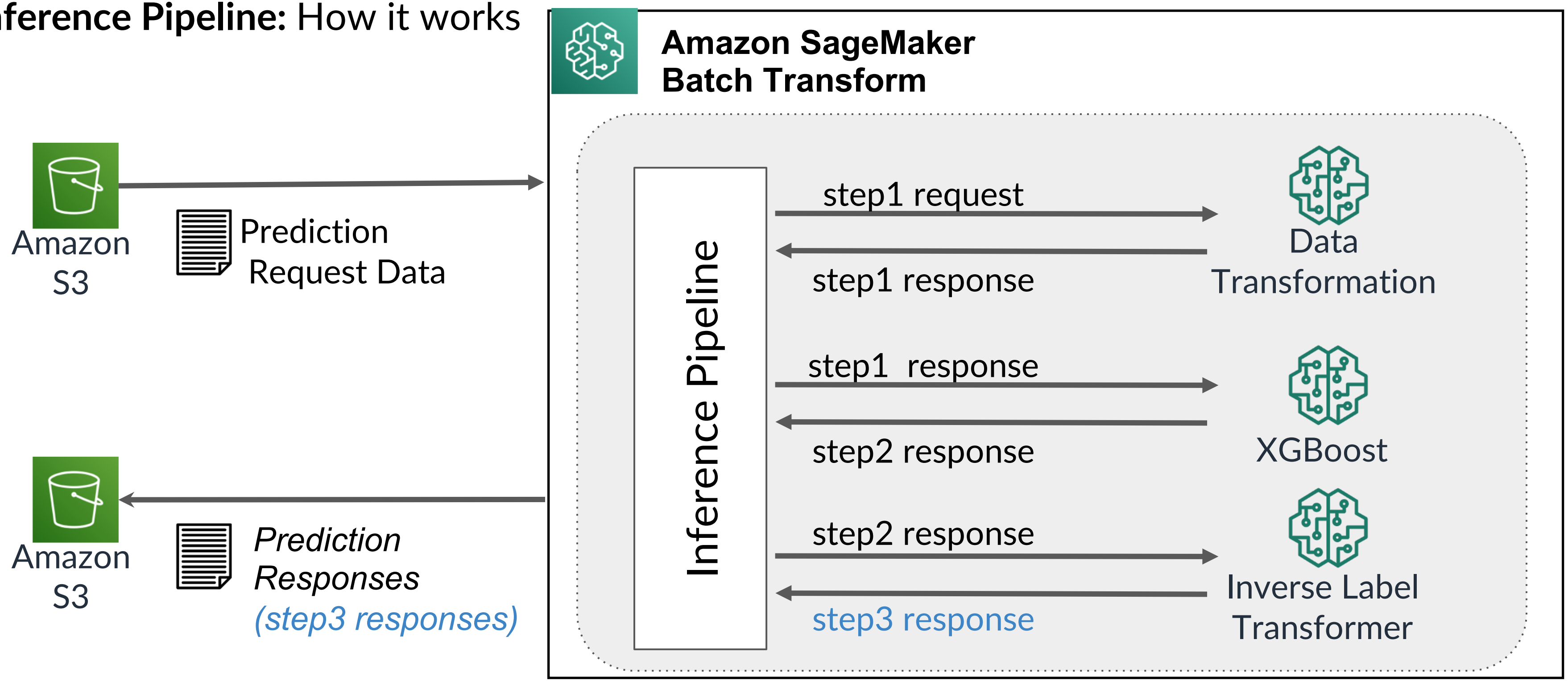


Deployment Options



Advanced Deployment Options

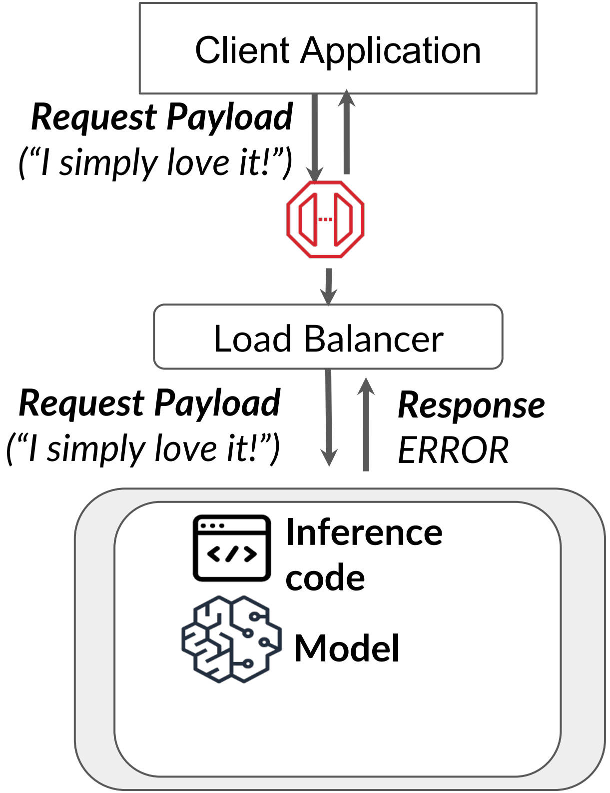
Inference Pipeline: How it works



Model Integration

Amazon SageMaker Model Integration

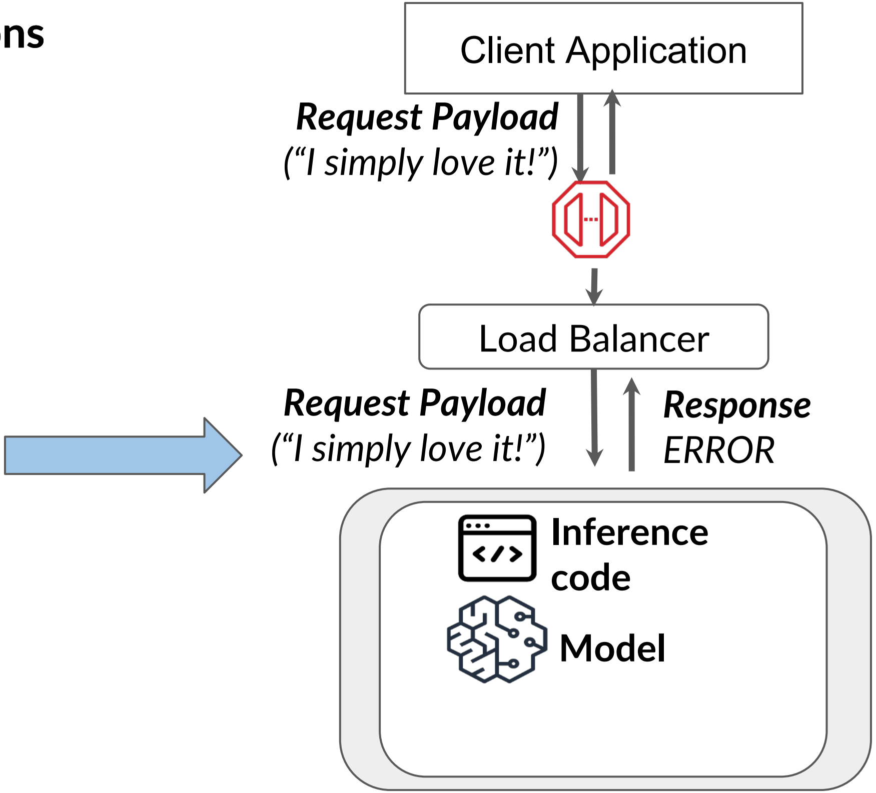
Integrating Models with ML Applications



Amazon SageMaker Model Integration

Integrating Models with ML Applications

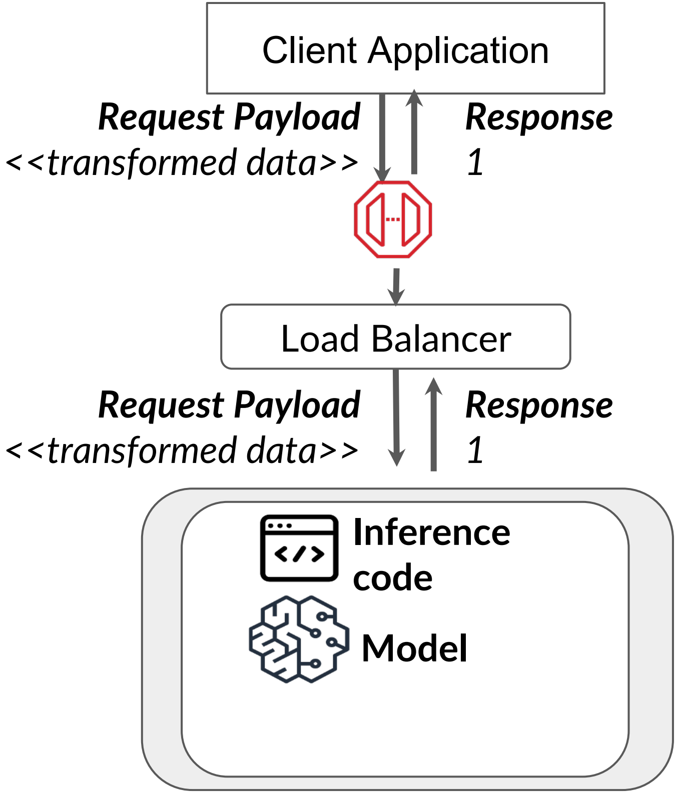
Need to apply the same data transformations used during training



Amazon SageMaker Model Integration

Prepare Data for Inference in Client Application

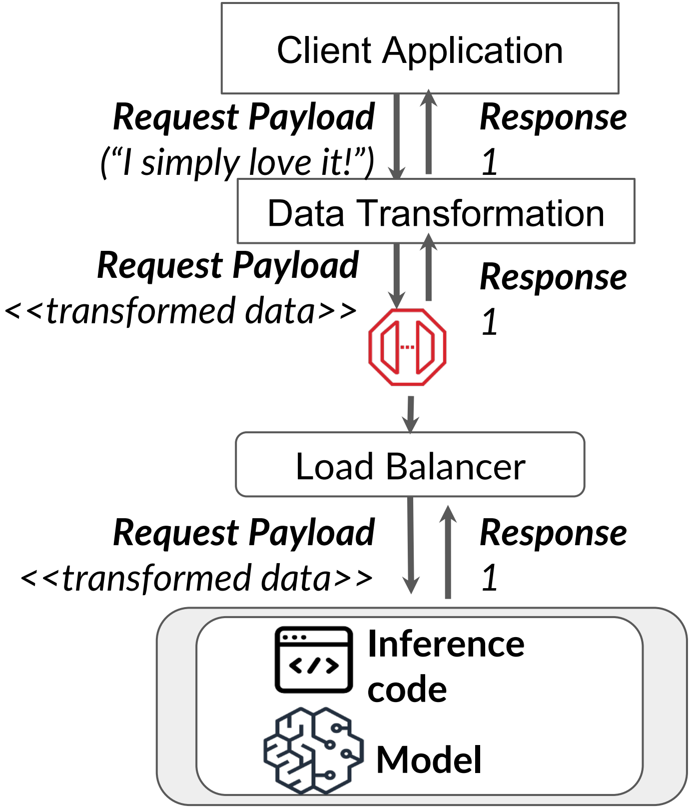
- **Implement data transformations in Client Application**
  - **Challenge:** Difficult to scale & manage
  - **Consideration:** Response may need to be transformed
    (1 = Positive)



Amazon SageMaker Model Integration

Prepare Data for Inference in Client Application

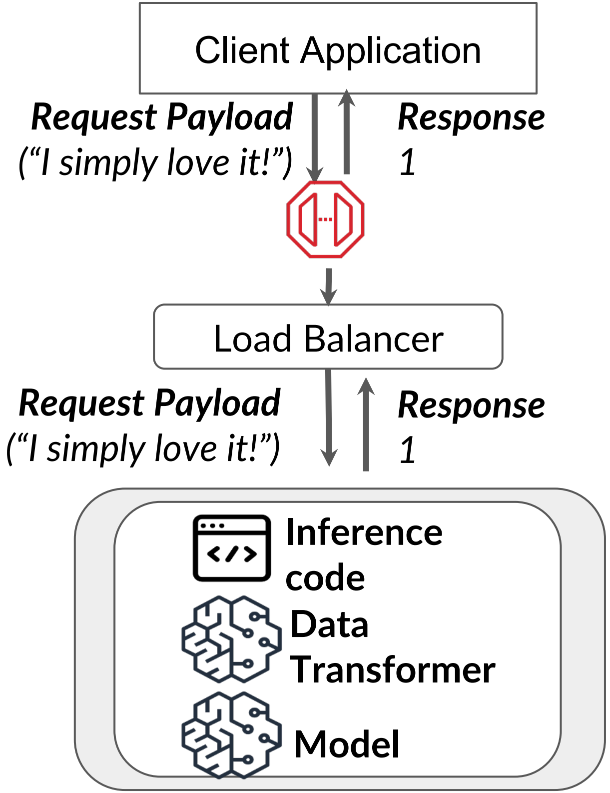
- **Implement transformation code before calling hosted model**
  - **Challenge:** Need to ensure transformation code stays in sync with training code



Amazon SageMaker Model Integration

Prepare Data as Part of an Inference Pipeline

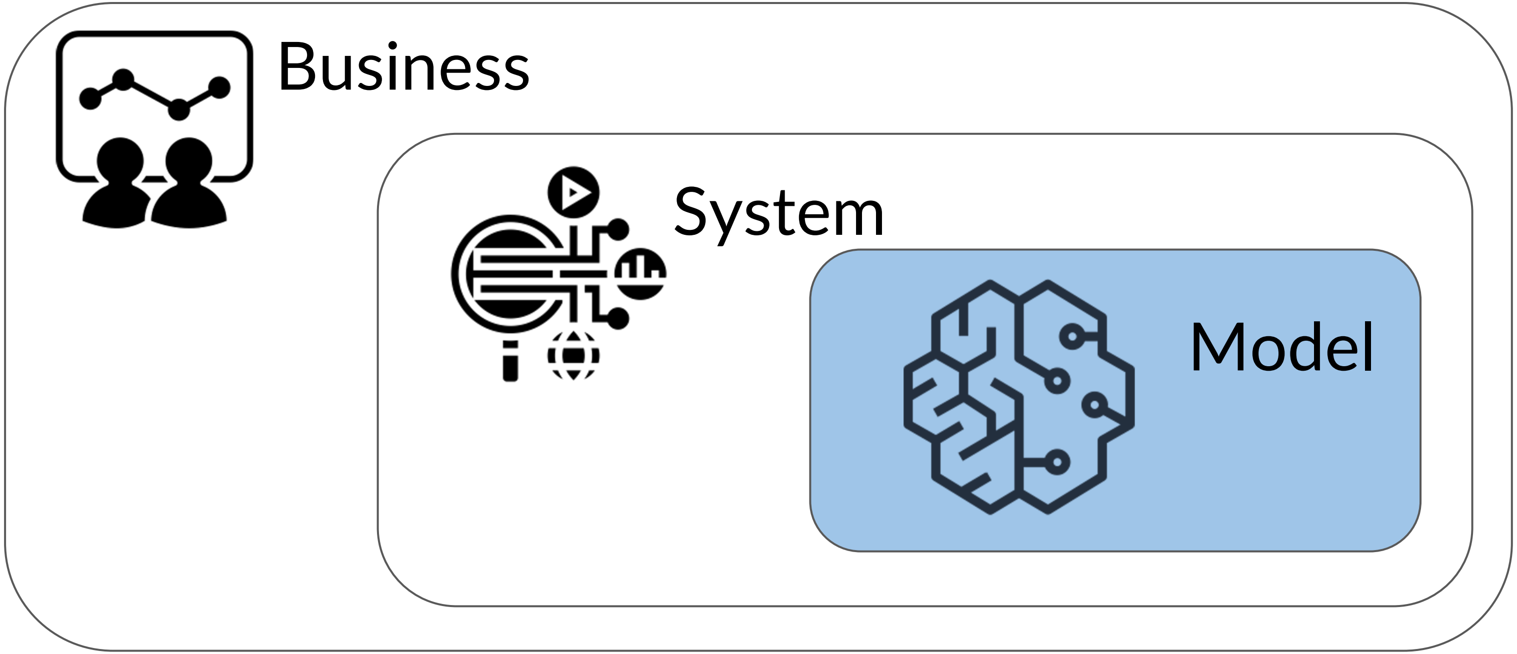
- **Implement data transformations in Inference Pipeline**
    - **Benefit:** Keep training & inference code in sync
    - **Consideration:** Additional Data Transformer for response may need to be transformed (1 = Positive)



Monitoring ML Workloads

Monitoring Machine Learning Workloads

Considerations



Monitoring Machine Learning Workloads

Model Monitoring

Why? Models degrade over time

Monitoring Machine Learning Workloads

Model Monitoring

Why? Models degrade over time

Examples

Customer behavior change
Ex. Product change, Demand change

Monitoring Machine Learning Workloads

Model Monitoring

Why? Models degrade over time

Examples

Customer behavior change
Ex. Product change, Demand change

Changing business environment
Ex. New products

Monitoring Machine Learning Workloads

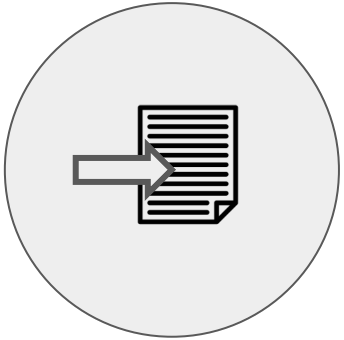
Model Monitoring

Why? Models degrade over time

Examples

Customer behavior change
Ex. Product change, Demand change

Changing business environment
Ex. New products



Changing data pipeline
Ex. Feature data suddenly missing

Monitoring Machine Learning Workloads

Model Monitoring

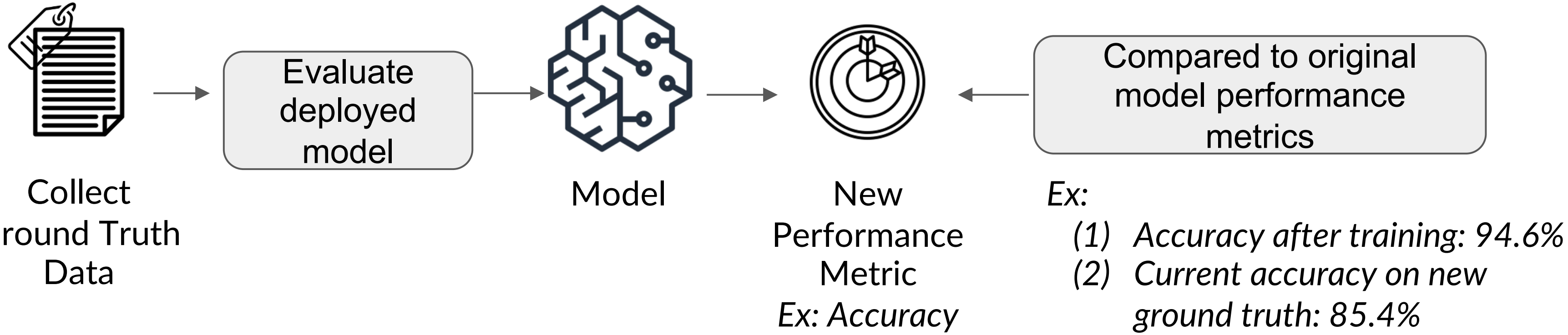
Monitoring Machine Learning Workloads

Concept Drift

What causes **concept drift?**

- Environment changes that impact the context of the predicted target

Methods to detect:



Monitoring Machine Learning Workloads

Data Drift

What causes **Data Drift?**

- Changes in the model input data

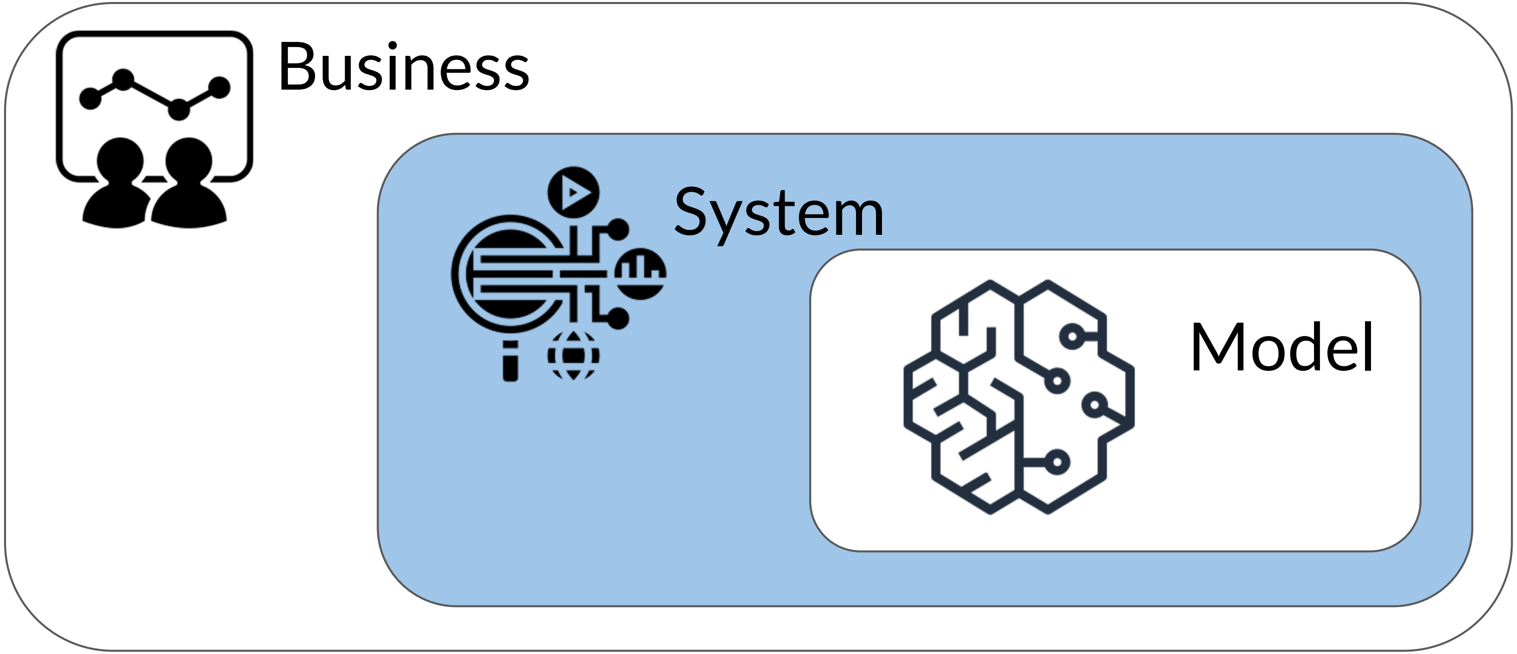
Methods to detect:

Example: Deequ - Open Source Library

- **Data Profiling:** Gather statistics about each feature used to train the model
- **Establish Constraints:** Boundaries on normal/expected data
- **Detect Data Anomalies:** Understand when prediction data violates constraints

Monitoring Machine Learning Workloads

Considerations

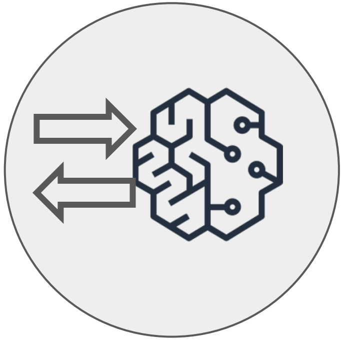


Monitoring Machine Learning Workloads

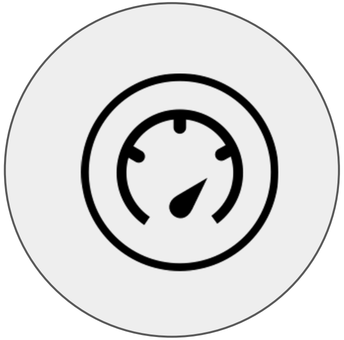
System Monitoring

Why? Ensure your model and supporting resources are functioning as expected

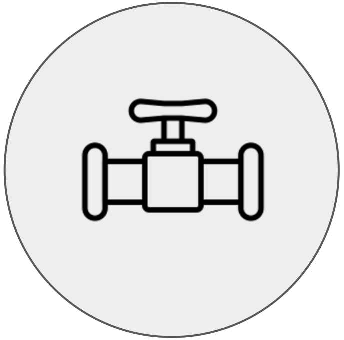
Examples



Model Latency



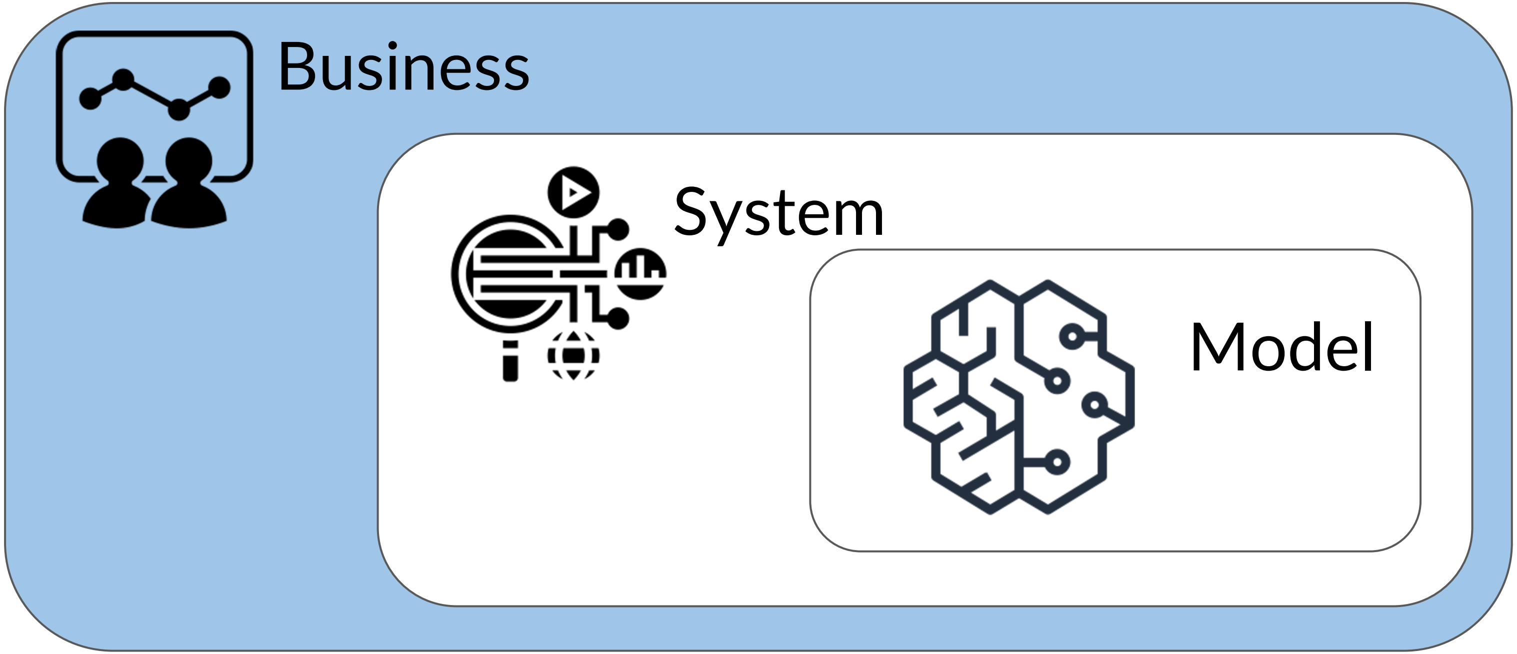
System Metrics



ML Pipeline

Monitoring Machine Learning Workloads

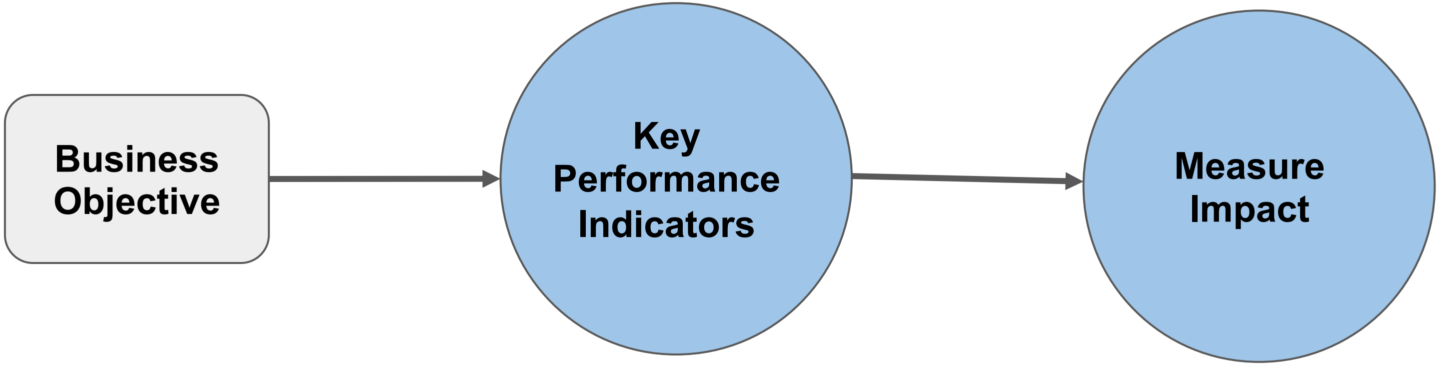
Considerations



Monitoring Machine Learning Workloads

Monitoring Impact on Business Objectives

Why? Ensure your model has impact on the business objective

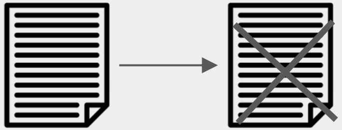


Model Monitoring

Using Amazon SageMaker Model Monitor

Amazon SageMaker Model Monitor

Monitor Types

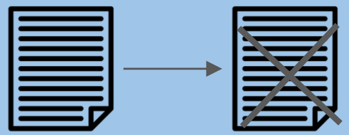
Data Drift	*Concept Drift*	*Concept Drift*	*Data Drift*
Data Quality	**Model Quality**	**Statistical Bias Drift**	**Feature Attribution Drift**
			
Monitor drift in data quality	Monitor drift in model quality metrics	Monitor statistical bias drift in model predictions	Monitor drift in feature attribution

Amazon SageMaker Model Monitor

Monitor Types

Data Drift

Data Quality



Monitor drift in data quality

Concept Drift

Model Quality

Monitor drift in model quality metrics

Concept Drift

Statistical Bias Drift

Monitor statistical bias drift in model predictions

Data Drift

Feature Attribution Drift

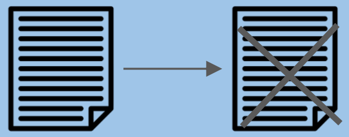
Monitor drift in feature attribution

Amazon SageMaker Model Monitor

Monitor Type: Data Quality Monitor

Data Drift

Data Quality



Monitor drift in data quality

- Monitor when inference data drifts away from baseline (training) data
- Model Monitor uses, **Deequ**, an open source library built on Apache Spark

Amazon SageMaker Model Monitor

Monitor Type: Data Quality Monitor

Deploy Endpoint → Enable Data Capture

% of data to capture

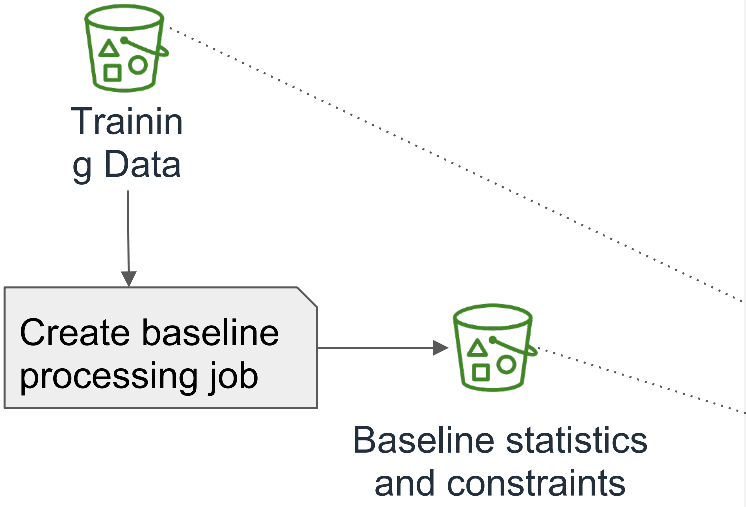
```
data_capture_config=DataCaptureConfig(
                        enable_capture = True,
                        sampling_percentage=100,

destination_s3_uri=s3_capture_upload_path)

predictor = model.deploy(initial_instance_count=1,
                instance_type='ml.m4.xlarge',
                endpoint_name=endpoint_name
                data_capture_config=data_capture_config)
```

Amazon SageMaker Model Monitor

Monitor Type: Data Quality Monitor




```
data_monitor = DefaultModelMonitor(
    #baseline processing job configuration - data
    ...
)

data_monitor.suggest_baseline(
    baseline_dataset=s3train_data_uri,
    dataset_format=DatasetFormat.csv(header=True),
    output_s3_uri=baseline_results_uri,
    wait=True
)
```

Amazon SageMaker Model Monitor

Monitor Type: Data Quality Monitor

Deploy Endpoint → Enable Data Capture → Create Baseline

Baseline statistics and constraints

statistics.json

- Columnar statistics for each feature
- Examples:
    - Numeric → missing values, mean, min, max, distribution
    - String → missing values, distinct values, categorical distribution

Amazon SageMaker Model Monitor

Monitor Type: Data Quality Monitor

Deploy Endpoint → Enable Data Capture → Create Baseline

Baseline statistics and constraints

statistics.json

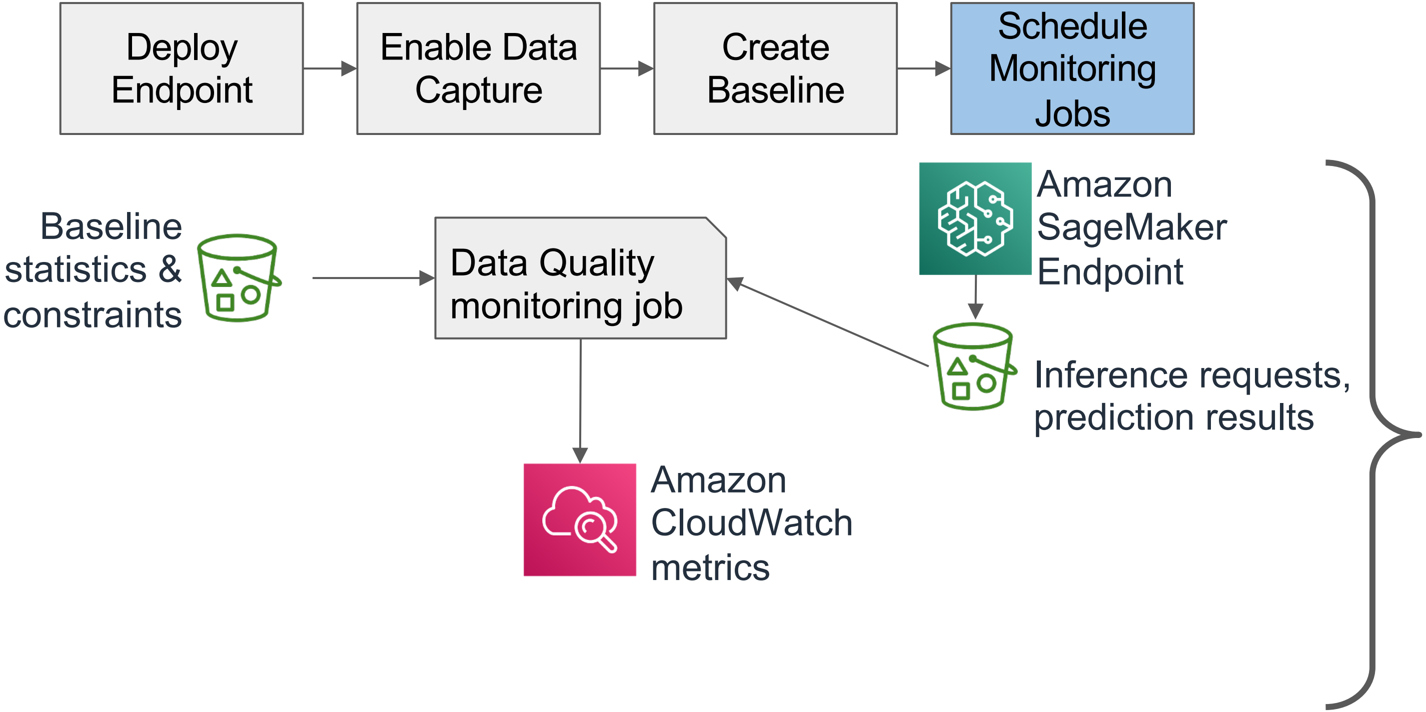
- Columnar statistics for each feature
- Examples:
  - Numeric → missing values, mean, min, max, distribution
  - String → missing values, distinct values, categorical distribution

constraints.json

- Constraints that are used to evaluate potential data drift
- Examples:
  - Numeric → non-negative
  - String → observed values

Amazon SageMaker Model Monitor

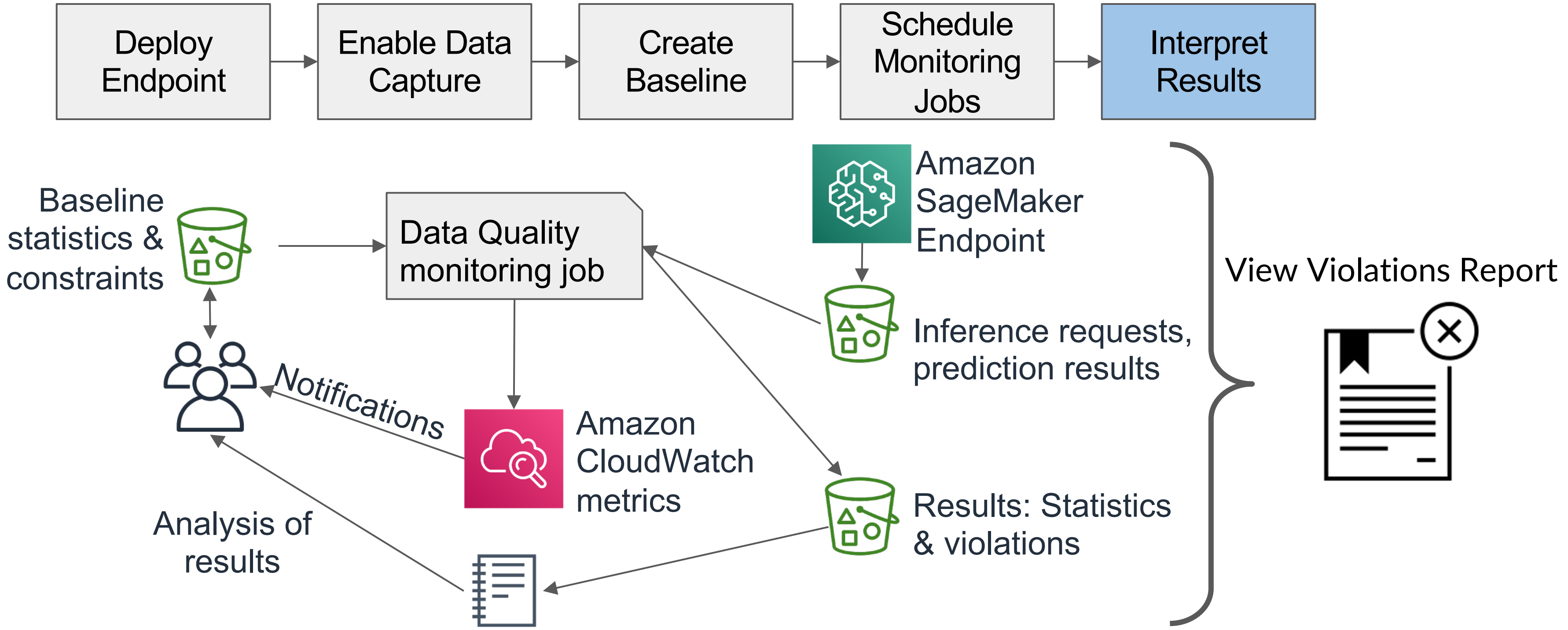
Monitor Type: Data Quality Monitor



Continuously monitor the data collected on a schedule

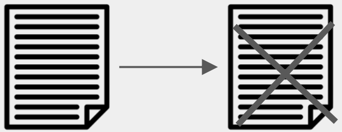
Amazon SageMaker Model Monitor

Monitor Type: Data Quality Monitor



Amazon SageMaker Model Monitor

Monitor Types

Data Drift	*Concept Drift*	*Concept Drift*	*Data Drift*
Data Quality	**Model Quality**	**Statistical Bias Drift**	**Feature Attribution Drift**
			
Monitor drift in data quality	Monitor drift in model quality metrics	Monitor statistical bias drift in model predictions	Monitor drift in feature attribution

Amazon SageMaker Model Monitor

Monitor Type: Model Quality Monitor

Concept Drift

Model Quality

Monitor drift in model quality metrics

- Monitor model quality by comparing model predictions with ground truth labels

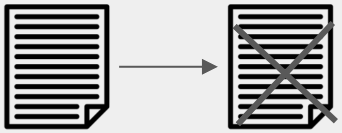
Training Data	Model Metric *Ex. Accuracy*	VS	Ground Truth Data	Model Metric *Ex. Accuracy*

Amazon SageMaker Model Monitor

Monitor Types

Data Drift

Data Quality



Monitor drift in data quality

Concept Drift

Model Quality

Monitor drift in model quality metrics

Concept Drift

Statistical Bias Drift

Monitor statistical bias drift in model predictions

Data Drift

Feature Attribution Drift

Monitor drift in feature attribution

Amazon SageMaker Model Monitor

Monitor Type: Statistical Bias Drift

Concept Drift

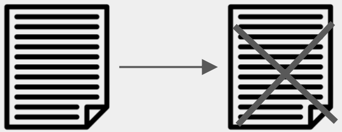
Statistical Bias Drift

Monitor statistical bias drift in model predictions

- Monitor predictions for statistical bias
- Amazon SageMaker Clarify integrates with Amazon SageMaker Model Monitor to detect statistical bias drift

Amazon SageMaker Model Monitor

Monitor Types

Data Drift	*Concept Drift*	*Concept Drift*	*Data Drift*
Data Quality	**Model Quality**	**Statistical Bias Drift**	**Feature Attribution Drift**
			
Monitor drift in data quality	Monitor drift in model quality metrics	Monitor statistical bias drift in model predictions	Monitor drift in feature attribution

Amazon SageMaker Model Monitor

Monitor Type: Feature Attribution Drift

Data Drift

Feature Attribution Drift

Monitor drift in feature attribution

- Monitor features contributing to predictions over time
- Amazon SageMaker Clarify integrates with Amazon SageMaker Model Monitor to detect feature attribution drift
- Utilizes SHAP for baselining