



SUMMIT
ONLINE

INT 06

DevOps for data science: Operationalising machine learning

Julian Bright

AI Specialist Solutions Architect
Amazon Web Services

Agenda

What is MLOps?

What's new in Amazon SageMaker for MLOps

Orchestration frameworks and tools

MLOps Demo

Sidebar: Data integration options

Wrap up

What is MLOps?

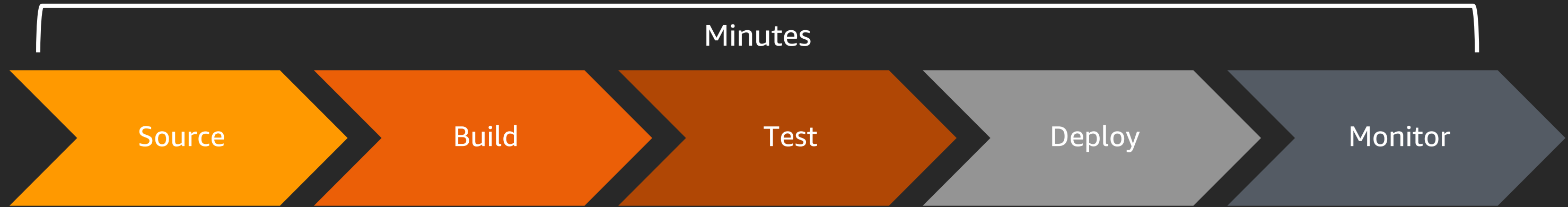
"Our highest priority is to satisfy the customer through early and continuous delivery of valuable software."

agilemanifesto.org/principles

"Our highest priority is to satisfy the customer through early and continuous delivery of valuable ~~software~~ insights from data."

agilemanifesto.org/principles

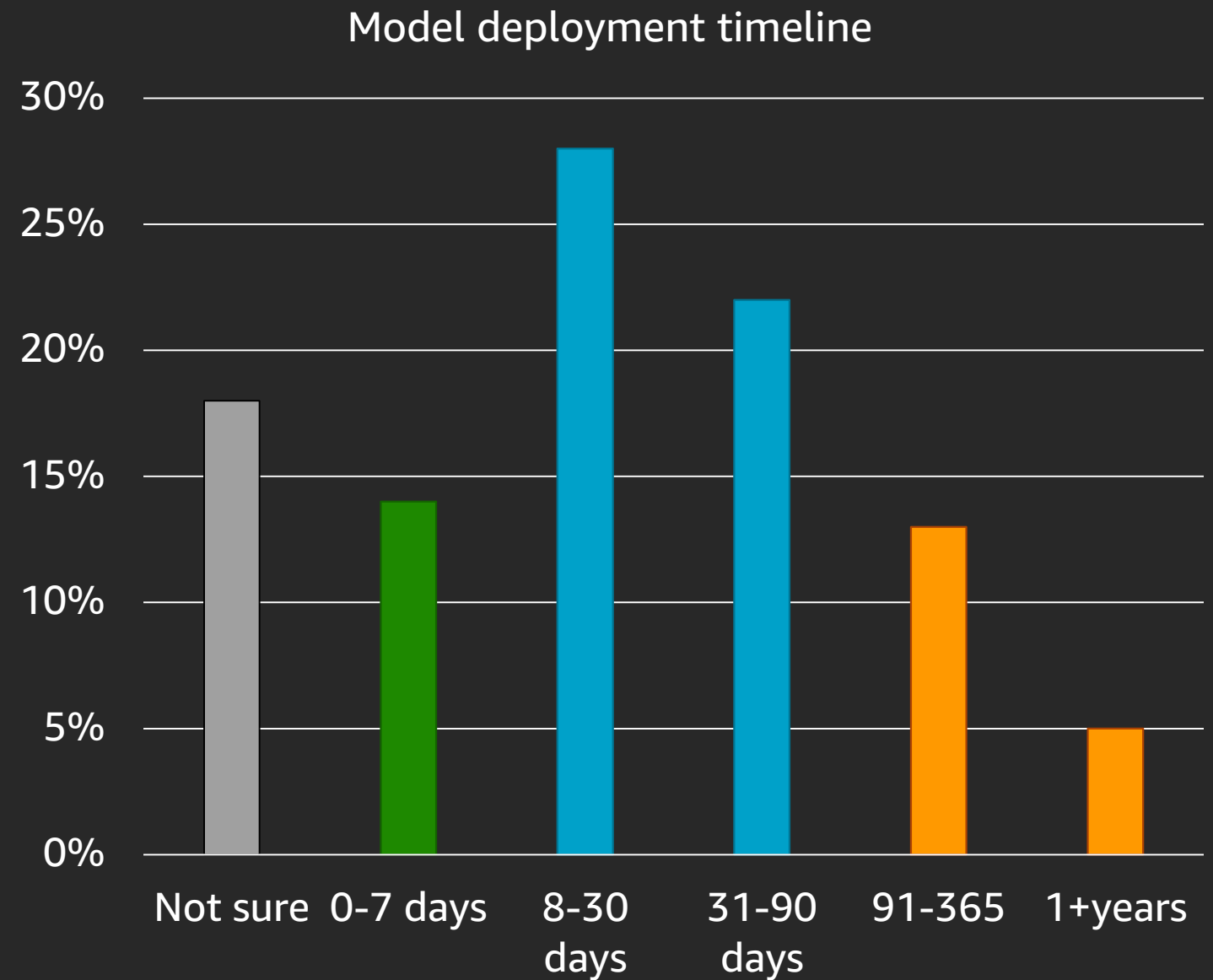
Dev Ops for traditional software development



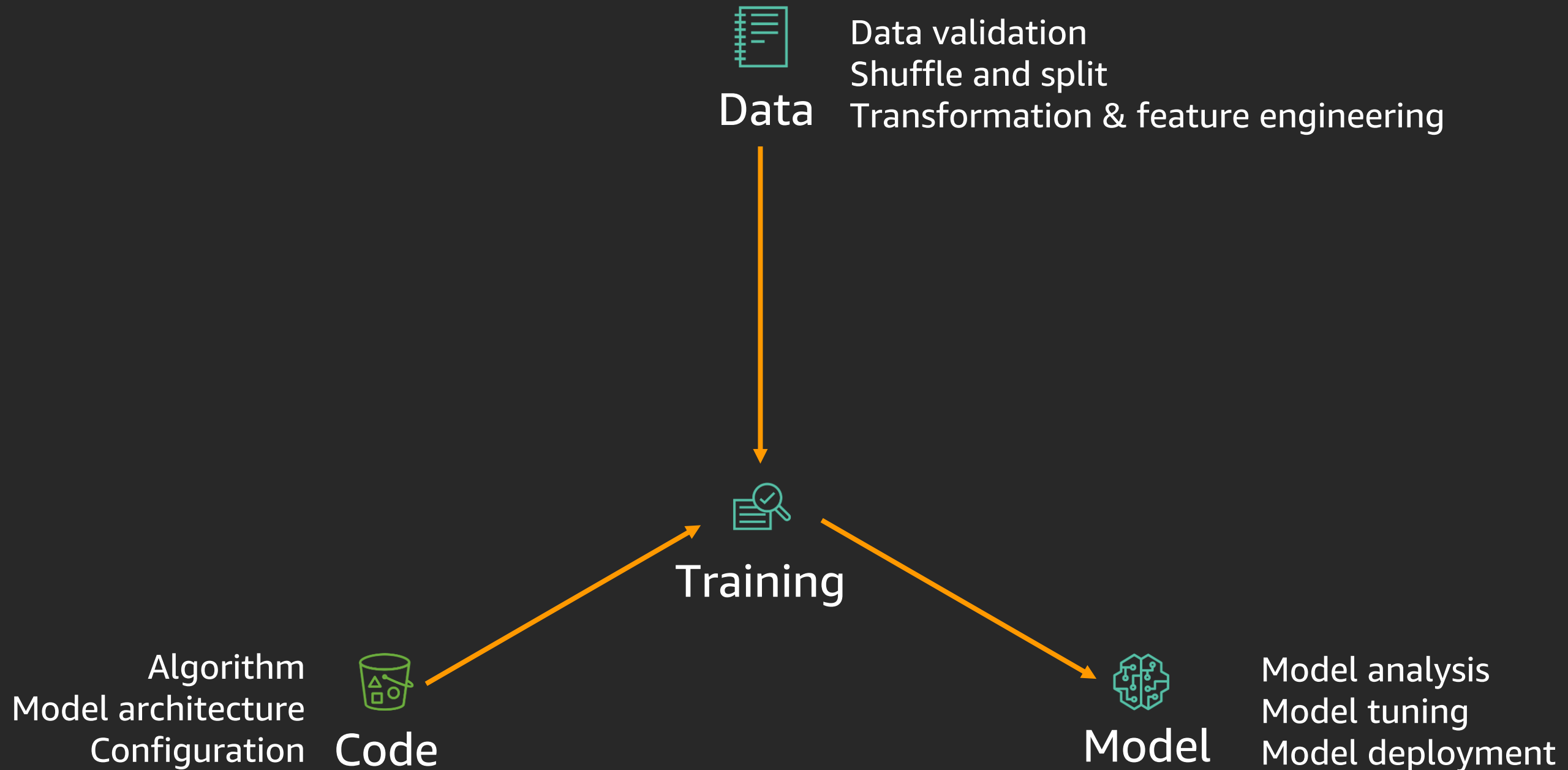
Machine Learning Deployment

Algorithma survey found 55% of companies have not deployed a machine learning model.

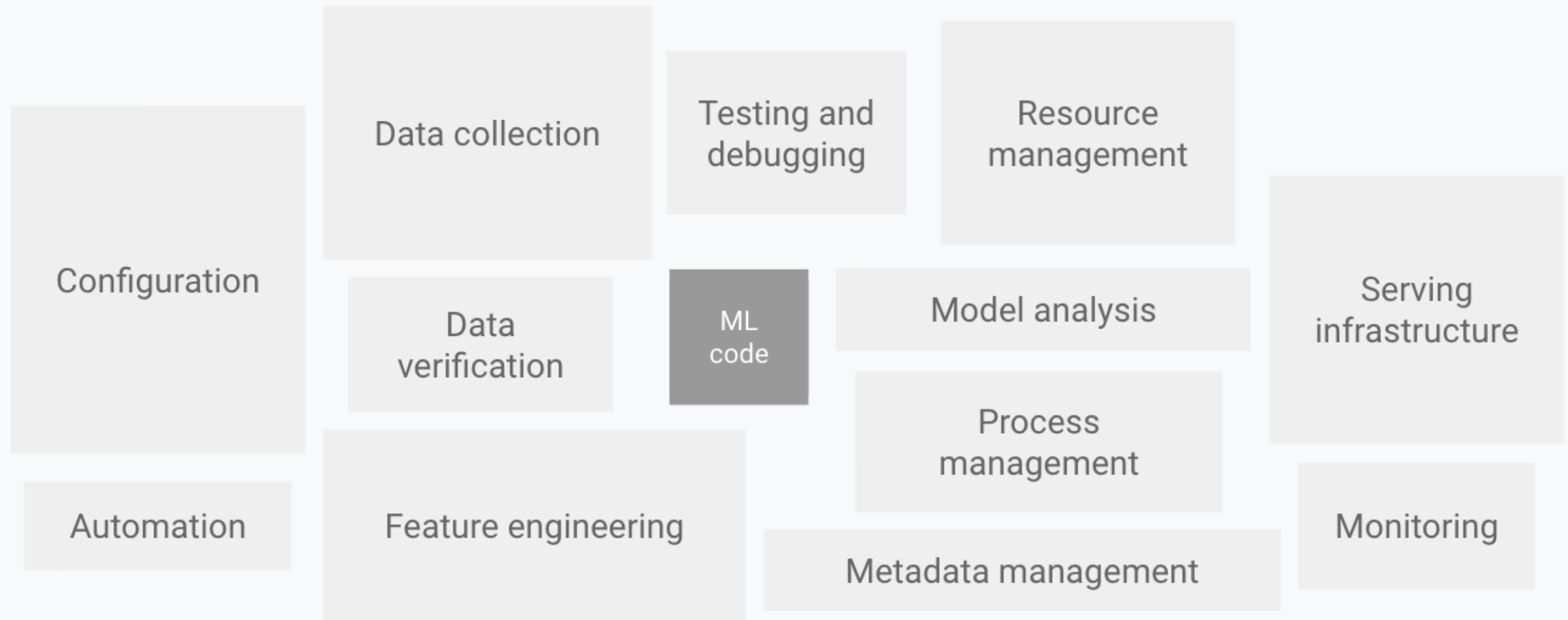
- 15% < 1 week
- 50% 1 week < 3 months
- 18% > 3 months



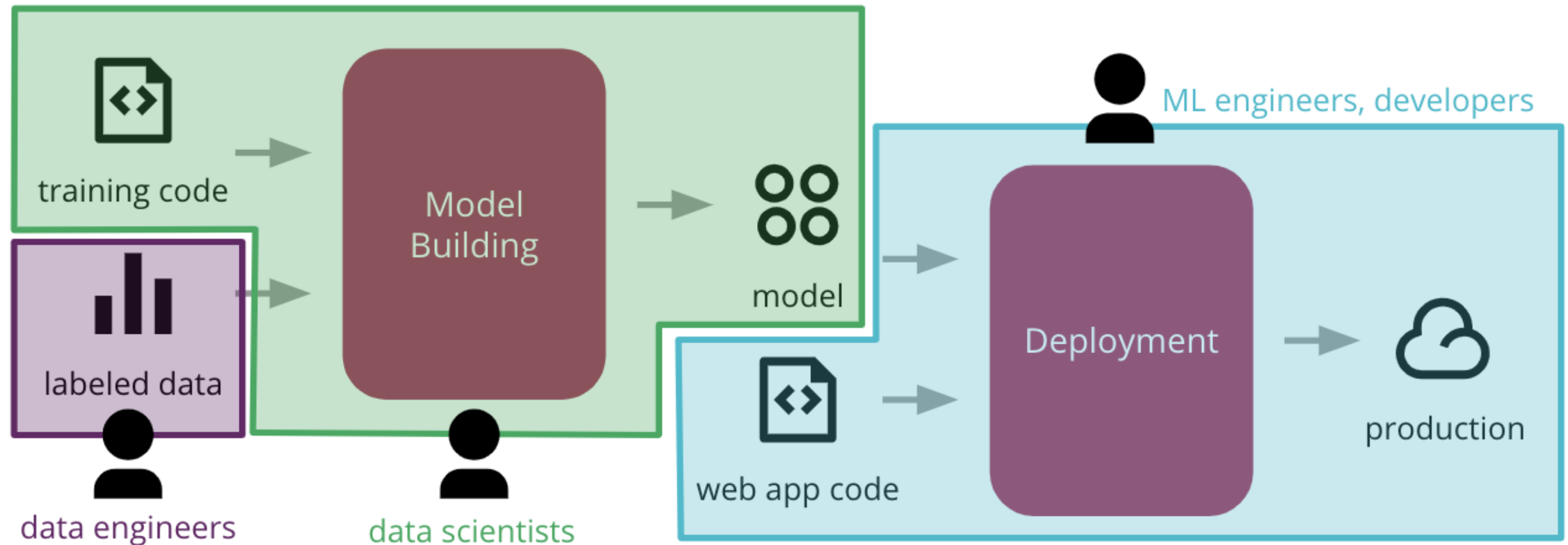
Machine Learning code and data are independent



Challenge: ML code is only small part of the solution



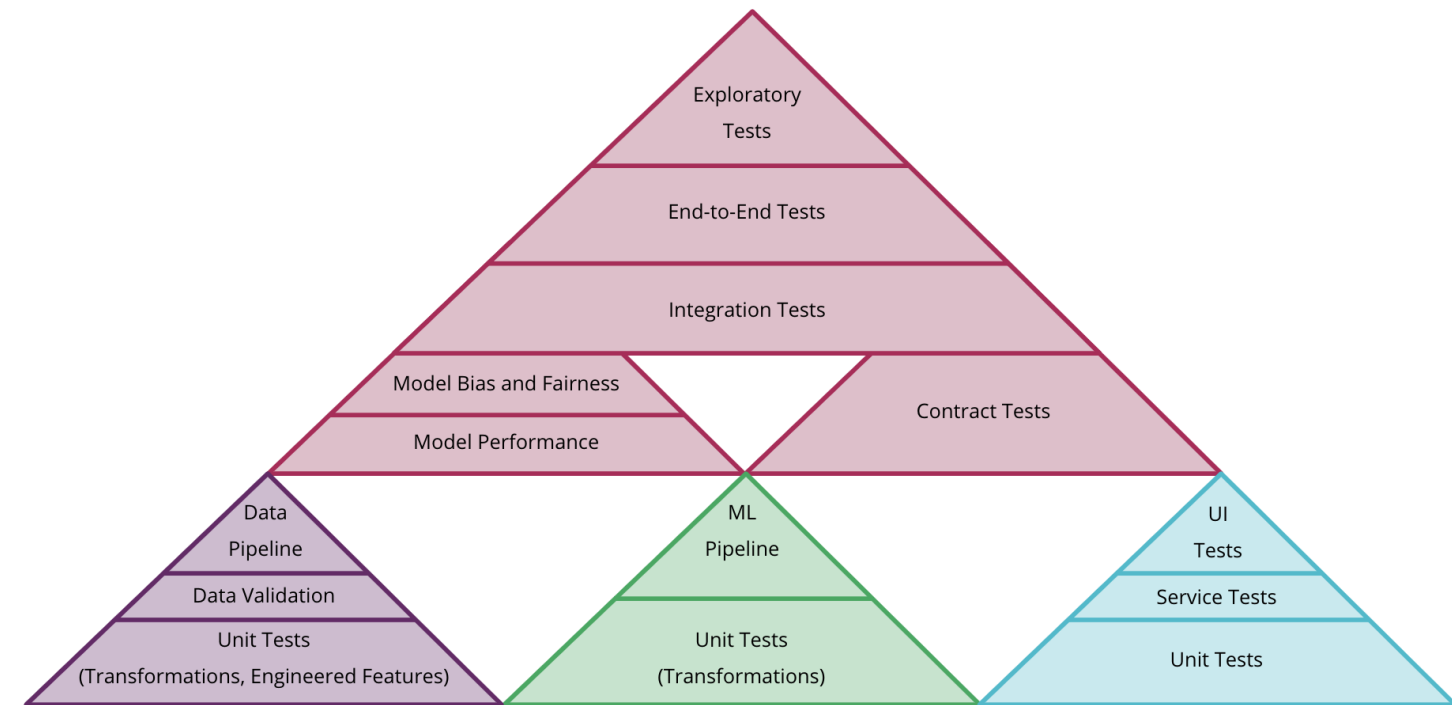
Challenge: Different teams might own part of process



Challenge: Ensuring test quality

Automated tests can add value and improve overall quality of ML

- Validating data distribution
- Validating model quality metrics
- Validating model bias & fairness



<https://martinfowler.com/bliki/TestPyramid.html>

Challenge: Detecting model drift

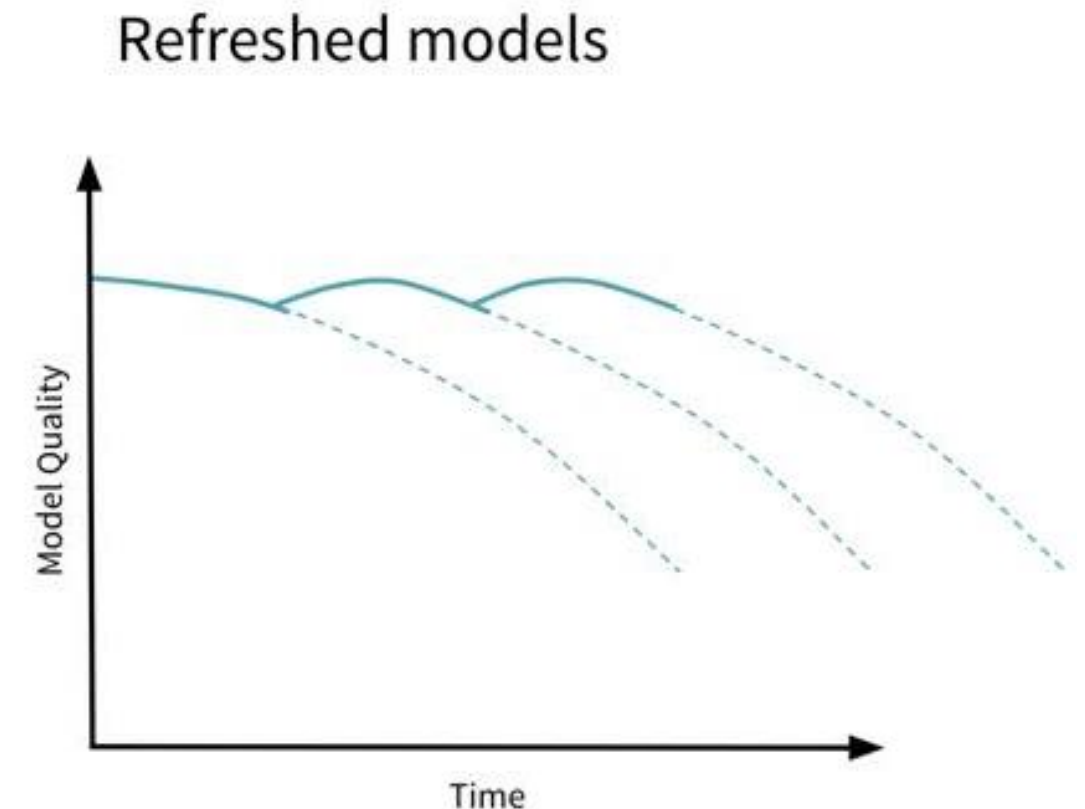
Drift could happen from various sources and hence you should monitor all these sources to ensure full coverage.

Training data

- Schema & distribution of incoming data
- Distribution of labels

Prediction responses

- Distribution of predictions
- Quality of predictions via feedback



ML Ops

Key challenges we are looking to solve across MLOps culture and practice

Role	Model reproducibility	Model audibility and explainability	Model deployment and monitoring
Data Scientist	Sharing and collaboration	Model analysis and evaluation	Concept drift detection

ML Ops

Key challenges we are looking to solve across MLOps culture and practice

Role	Model reproducibility	Model audibility and explainability	Model deployment and monitoring
Data Scientist	Sharing and collaboration	Model analysis and evaluation	Concept drift detection
Data Engineer	Machine learning pipelines	Code and data artifact integrity	Scalability and performance

ML Ops

Key challenges we are looking to solve across MLOps culture and practice














Role	Model reproducibility	Model audibility and explainability	Model deployment and monitoring
Data Scientist	Sharing and collaboration	Model analysis and evaluation	Concept drift detection
Data Engineer	Machine learning pipelines	Code and data artifact integrity	Scalability and performance
Dev Ops	Visualise experiment management	Capture training logs and prediction results	Continuous deployment and rollback

What's new in Amazon SageMaker


The AWS machine learning stack

Broadest and most complete set of machine learning capabilities

AI services

Vision	Speech		Text			Search	Chatbots	Personalization	Forecasting	Fraud	Development	Contact centers
												
Amazon Rekognition	Amazon Polly	Amazon Transcribe <i>+Medical</i>	Amazon Comprehend <i>+Medical</i>	Amazon Translate	Amazon Textract	Amazon Kendra	Amazon Lex	Amazon Personalize	Amazon Forecast	Amazon Fraud Detector	Amazon CodeGuru	Contact Lens <i>for Amazon Connect</i>

ML services

 Amazon SageMaker	Ground Truth	ML Marketplace	Amazon SageMaker Studio IDE								Amazon SageMaker Neo	Augmented AI
			Built-in algorithms	Notebooks	Experiments	Model training & tuning	Debugger	Autopilot	Model hosting	Model Monitor		

ML frameworks & infrastructure

 TensorFlow				AWS Deep Learning AMIs & Containers	GPUs & CPUs	Amazon Elastic Inference	AWS Inferentia	FPGA
								
			DeepGraphLibrary					

Amazon SageMaker Experiments

Organise, track, and compare training experiments



Tracking at scale

Track parameters and metrics across experiments and users



Custom organisation

Organise experiments by teams, goals and hypotheses



Visualisation

Easily visualise experiments and compare



Metrics and logging

Log custom metrics using the Python SDK and APIs



Fast Iteration

Quickly go back & forth and maintain high quality

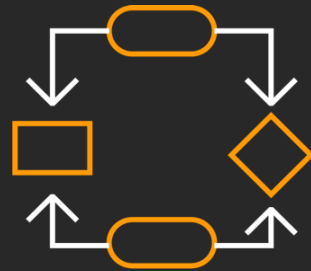
Amazon SageMaker Debugger

Analysis and debugging, explainability, and alert generation



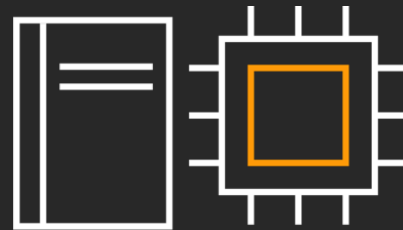
Relevant data
capture

Data is automatically
captured for analysis



Data analysis &
debugging

Analyse and debug data
with no code changes



Automatic error
detection

Errors are automatically
detected based on rules



Improved productivity
with alerts

Take corrective action
based on alerts



Visual analysis
and debugging

Visually analyse and
debug from Amazon
SageMaker Studio

Amazon SageMaker model monitor

Continuous monitoring of models in production



Automatic data
collection

Data is automatically
collected from
your endpoints



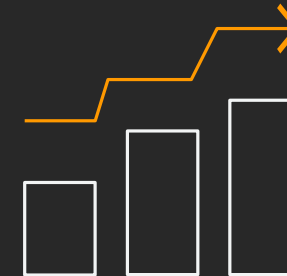
Continuous
monitoring

Define a monitoring
schedule, and detect
changes in quality against
a pre-defined baseline



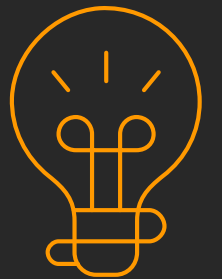
Flexibility
with rules

Use built-in rules to
detect data drift, or
write your own rules
for custom analysis



Visual
data analysis

See monitoring results,
data statistics, and
violation reports in Amazon
SageMaker Studio



Integration
with Amazon
CloudWatch

Automate corrective
actions based on
CloudWatch alerts

Orchestration frameworks and tools

MLOps orchestration options

Amazon SageMaker provides native integration for a number of orchestration frameworks

1. Amazon SageMaker Operators

- Apache Airflow
- Kubernetes

2. AWS Developer Tools

- AWS CodePipeline, AWS CodeBuild, AWS CodeDeploy and AWS CloudFormation

3. AWS Step Functions

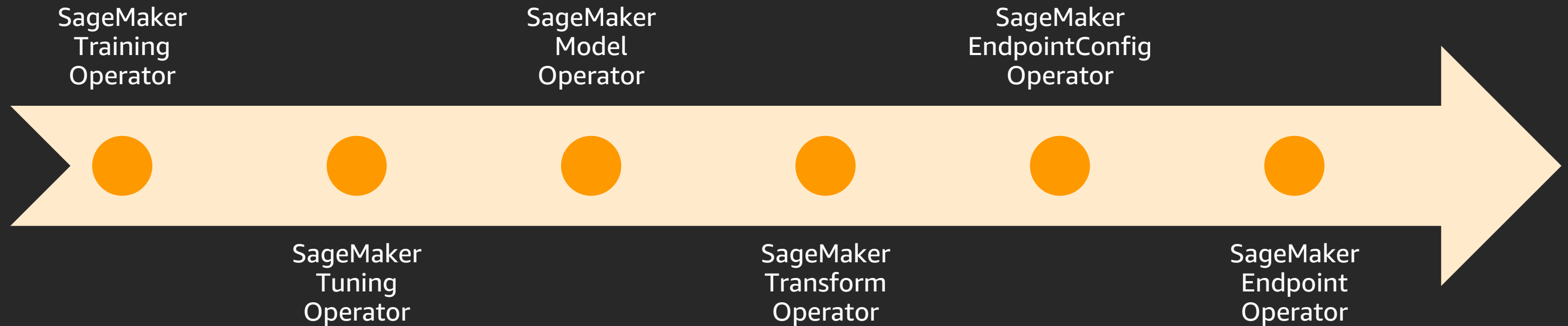
- AWS Step Functions Data Science SDK for Amazon SageMaker

4. Third Party open source

- ML Flow, Netflix Metaflow

Amazon SageMaker Operators for Apache Airflow

Orchestrate and automate sequences of ML tasks



Amazon SageMaker Operators for Kubernetes

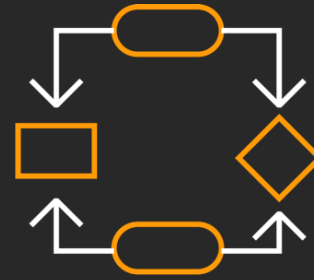
Kubernetes users can train, tune, and deploy models in Amazon SageMaker



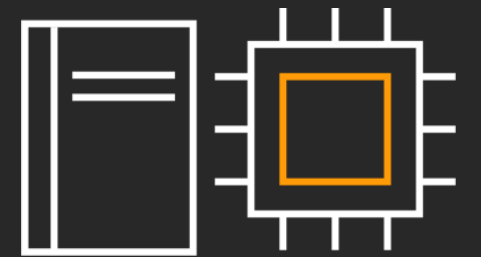
Train, tune, and
deploy models in
Amazon SageMaker



Orchestrate ML
workloads from your
Kubernetes
environments



Create pipelines and
workflows in
Kubernetes



Fully managed
infrastructure in
Amazon SageMaker

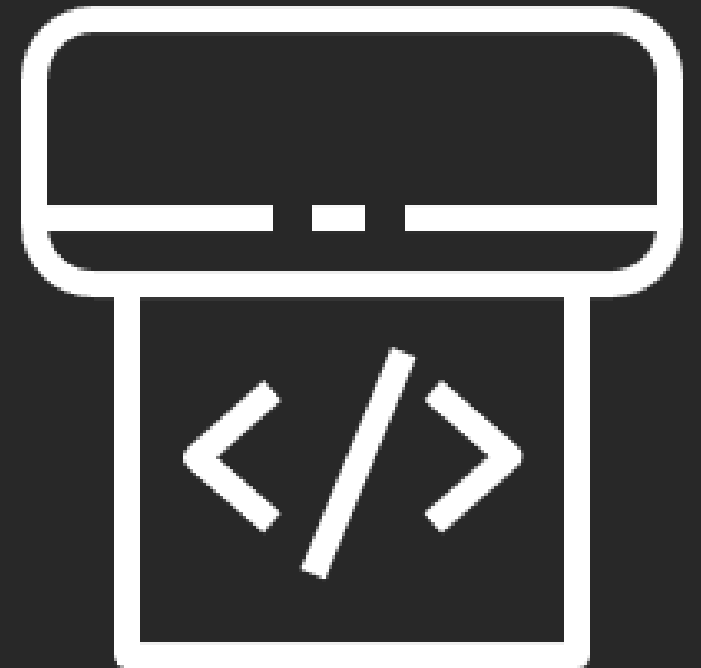
AWS Developer Tools



1. Automatically kick off a new build when new code is checked in
2. Build and test code in a consistent, repeatable environment
3. Continually have an artifact ready for deployment
4. Continually close feedback loop when build fails

AWS CodePipeline

- Continuous delivery service for fast and reliable application updates
- Model and visualise your software release process
- Builds, tests, and deploys your code every time there is a code change
- Integrates with third-party tools and AWS
- Pipeline execution variables



Continuous deployment goals



1. Automatically deploy new changes to staging environments for testing
2. Deploy to production safely without impacting customers
3. Deliver to customers faster: Increase deployment frequency and reduce change lead time and change failure rate

AWS CodeDeploy

- Automates code deployments to any instance and AWS Lambda
- Handles the complexity of updating your applications
- Avoid downtime during application deployment
- Roll back automatically if failure is detected
- Deploy to Amazon EC2, Amazon ECS, Lambda, or on-premises servers



AWS CloudFormation with CodeDeploy

Best practices for AWS CodeDeploy provisioning in CloudFormation

- Layer your application to reduce blast radius when updating resources
- Use multiple, isolated environments for testing, production, development, staging, etc.
- Smaller files are easier to write, test, and troubleshoot



Front-end
API

Amazon API Gateway

Lambda
functions

AWS CodeDeploy with Pre/Post Hooks

ML
resources

Amazon SageMaker Automatic Scaling

Monitoring
resources

Amazon Cloudwatch Alarms, dashboards

Base
network

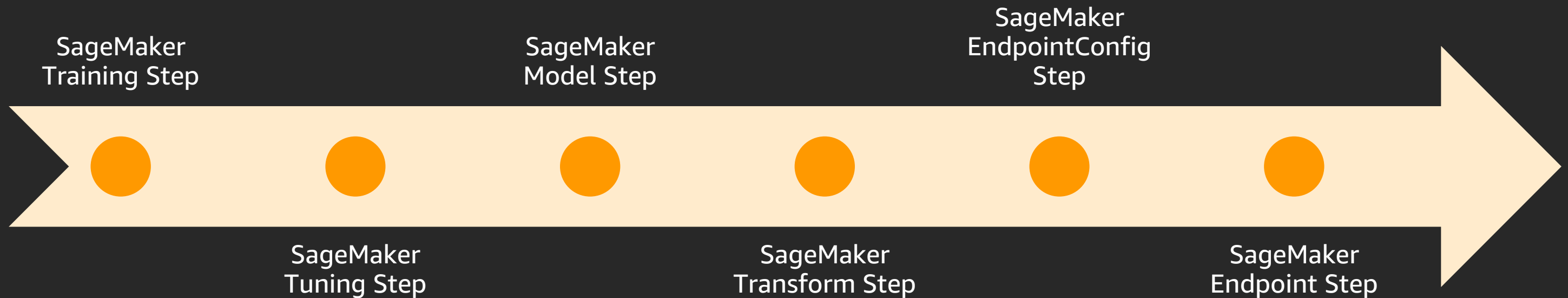
VPCs, NAT gateways, VPNs, subnets

Identity &
security

IAM users, groups, roles, policies

AWS Step Functions Data Science SDK

Visualise end-to-end data science workflows



AWS Step Functions



Simplify building workloads, such as order processing, report generation, and data analysis

Write and maintain less code; add services in minutes

Direct service integrations:



Amazon SNS



Amazon SQS



Amazon
SageMaker



AWS Glue



AWS Batch



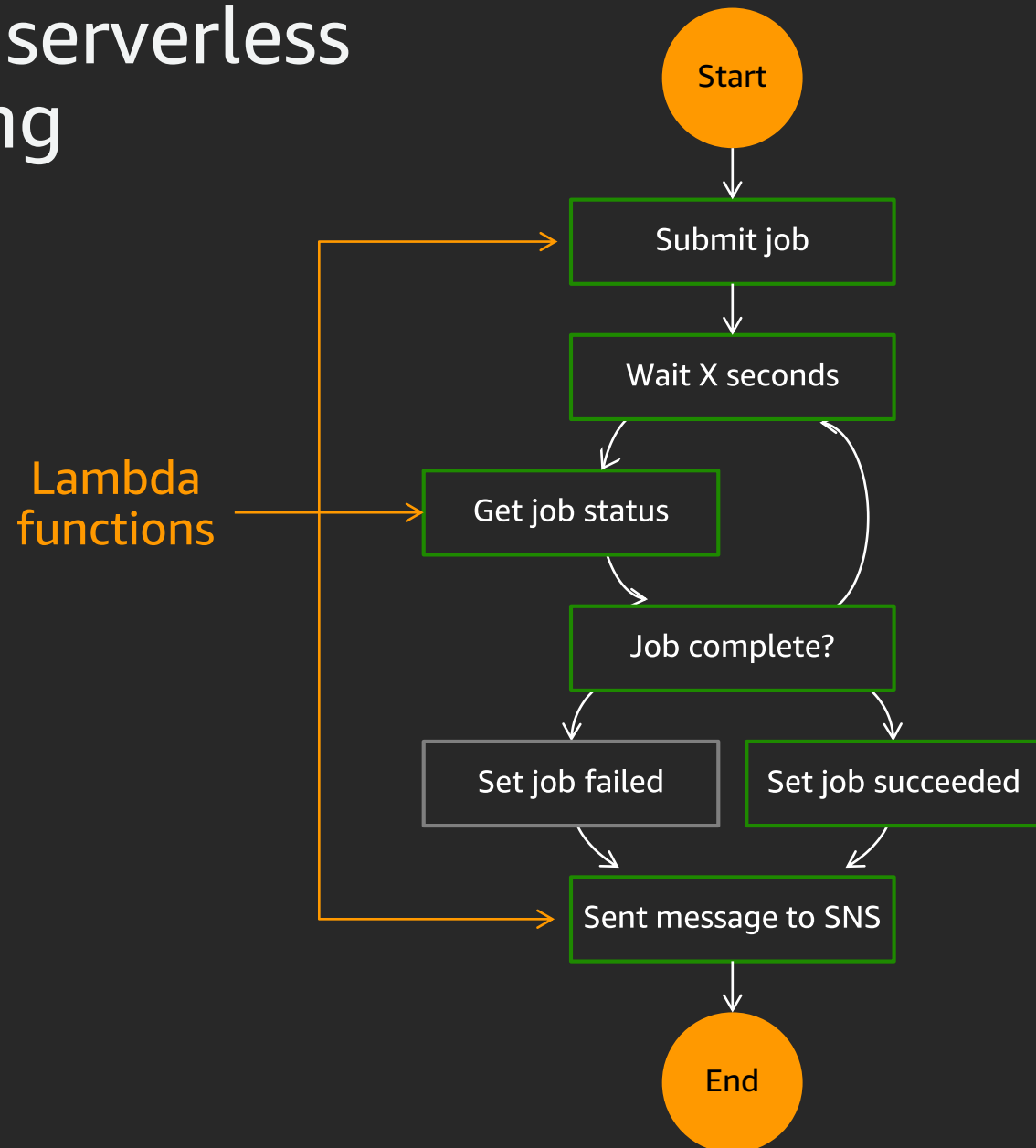
Amazon ECS



AWS Fargate

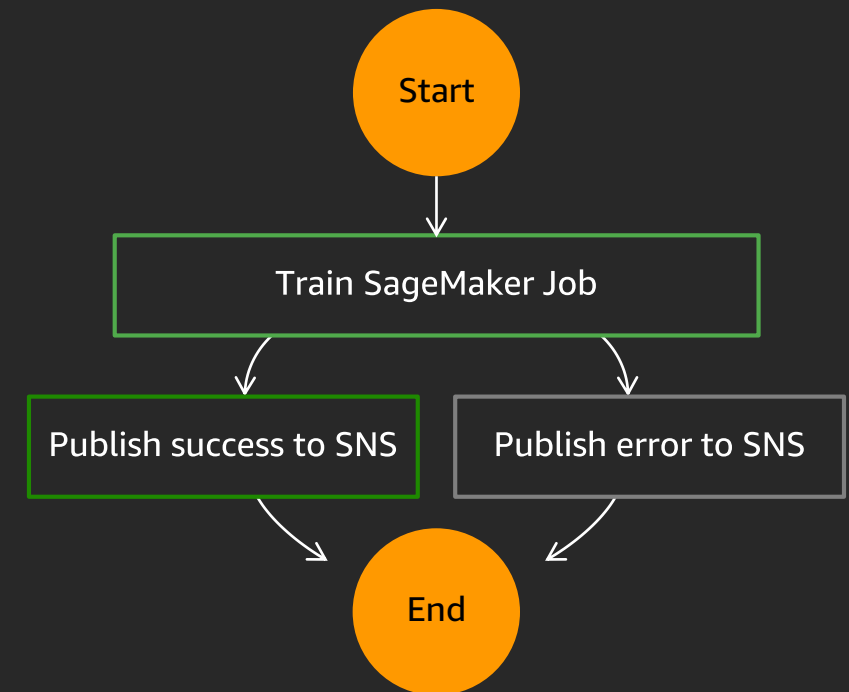
Simpler integration, less code

With serverless polling



With direct service integration

No
Lambda
functions



Third party open source integrations

MLflow:

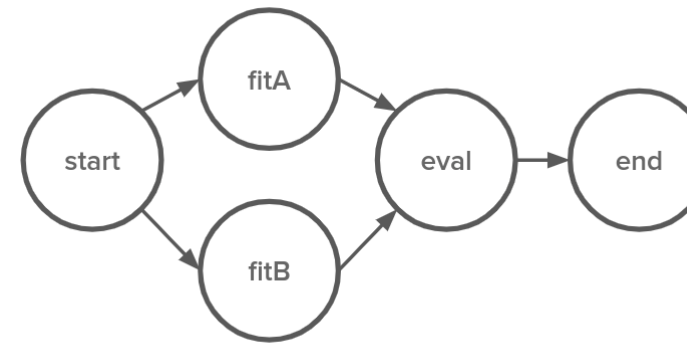
An open source platform for the machine learning lifecycle



Third party open source integrations

Netflix Metaflow:

Build and manage real-life data science projects with ease



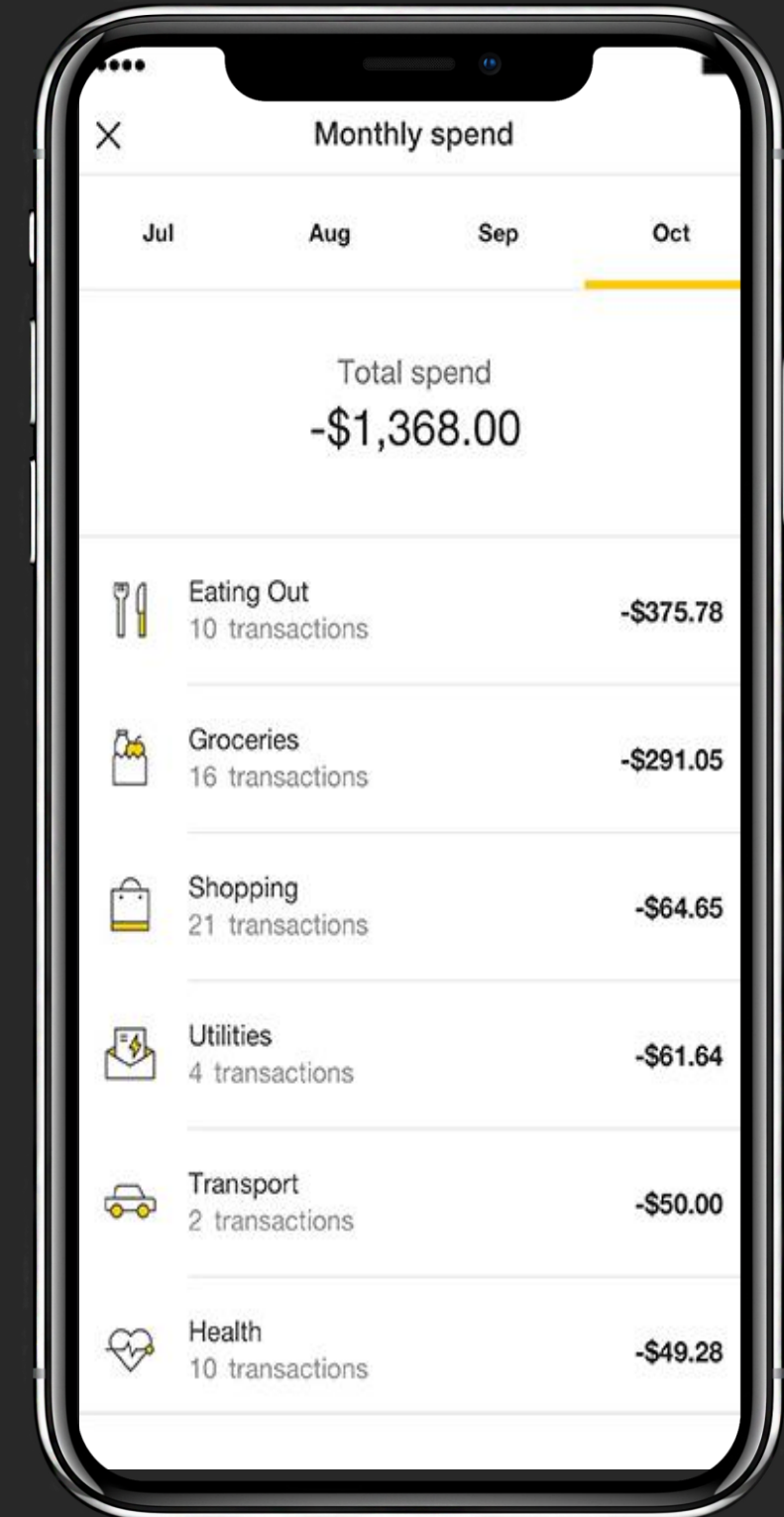
```
class MyFlow(FlowSpec):  
  
    @step  
    def start(self):  
        self.data = load_data()  
        self.next(self.fitA, self.fitB)  
  
    @step  
    def fitA(self):  
        self.model = fit(self.data, model='A')  
        self.next(self.eval)  
  
    @step  
    def fitB(self):  
        self.model = fit(self.data, model='B')  
        self.next(self.eval)  
  
    @step  
    def eval(self, inputs):  
        self.best = max((i.model.score, i.model)  
                        for i in inputs)[1]  
        self.next(self.end)  
  
    @step  
    def end(self):  
        print('done!')
```

MLOps demo

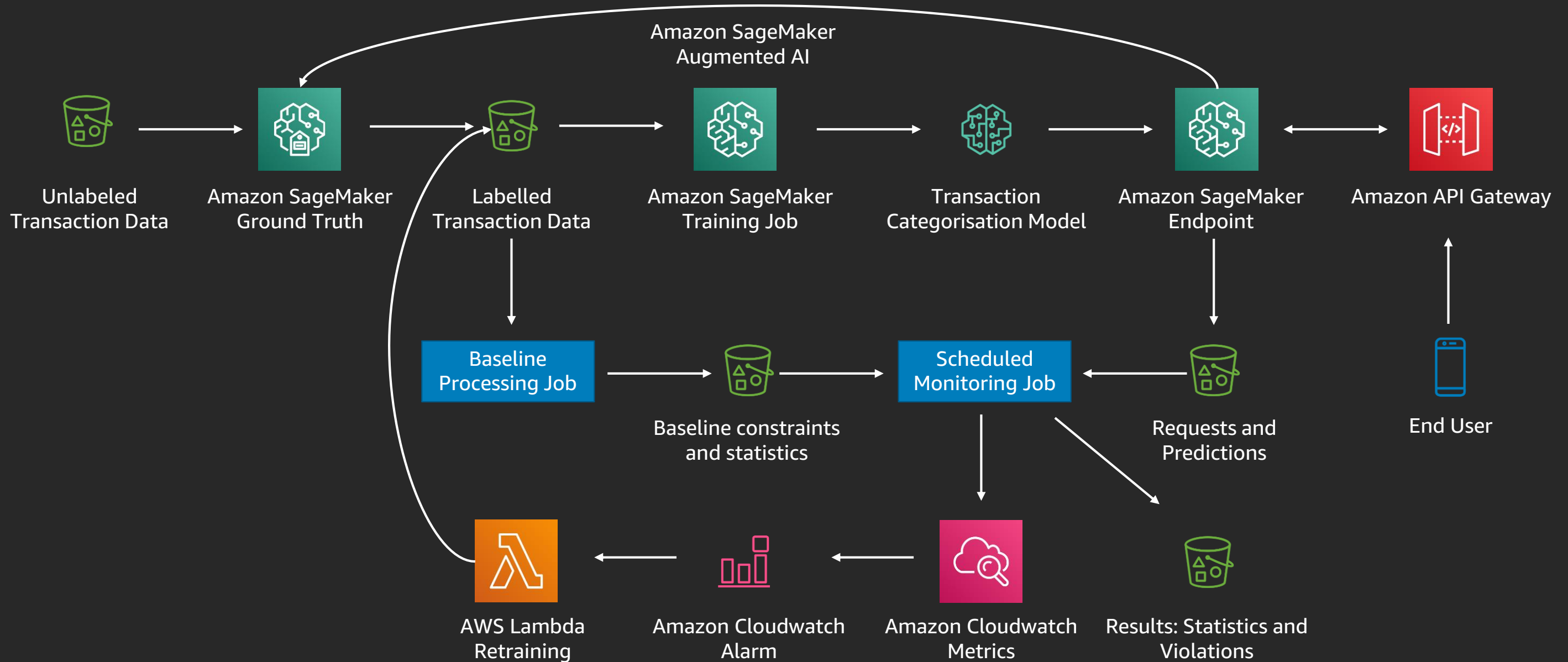
Use case: Transaction categorisation

Machine learning classifier predicts bank transaction category to provide insights on spending.

Automatic model retraining with new labelled data.



Transaction categorisation architecture



AWS CodePipeline

Following are the high level steps:

1. Source
2. Build Artifacts
3. Train
4. Deploy Dev
5. Manual Approval
6. Deploy Prod
7. Monitor

The screenshot displays the AWS CodePipeline console interface. The top section shows the **Source** stage, which has completed successfully. Below this, three provider actions are listed: **GitSource** (GitHub), **EcrSource** (Amazon ECR), and **DataSource** (Amazon S3). Each action shows a success status and a commit ID. Below the actions, the build artifacts are listed: **EcrSource** with artifact `sha256:8f65708592c0`, **GitSource** with artifact `a0630051`, and **DataSource** with artifact `R52G.D02Z22ZfdV0GbSe_5UTVcd5NHfz`. A **Disable transition** button is visible. The bottom section shows the **Build** stage, which is currently **In progress**. The **BuildTemplates** action (AWS CodeBuild) is listed with an in-progress status.

Source Succeeded
Execution ID: [f4afefba-2b0a-432a-9546-5919a50260f8](#)

GitSource	EcrSource	DataSource
GitHub	Amazon ECR	Amazon S3
Succeeded - Just now a0630051	Succeeded - Just now <code>sha256:8f65708592c0</code>	Succeeded - Just now

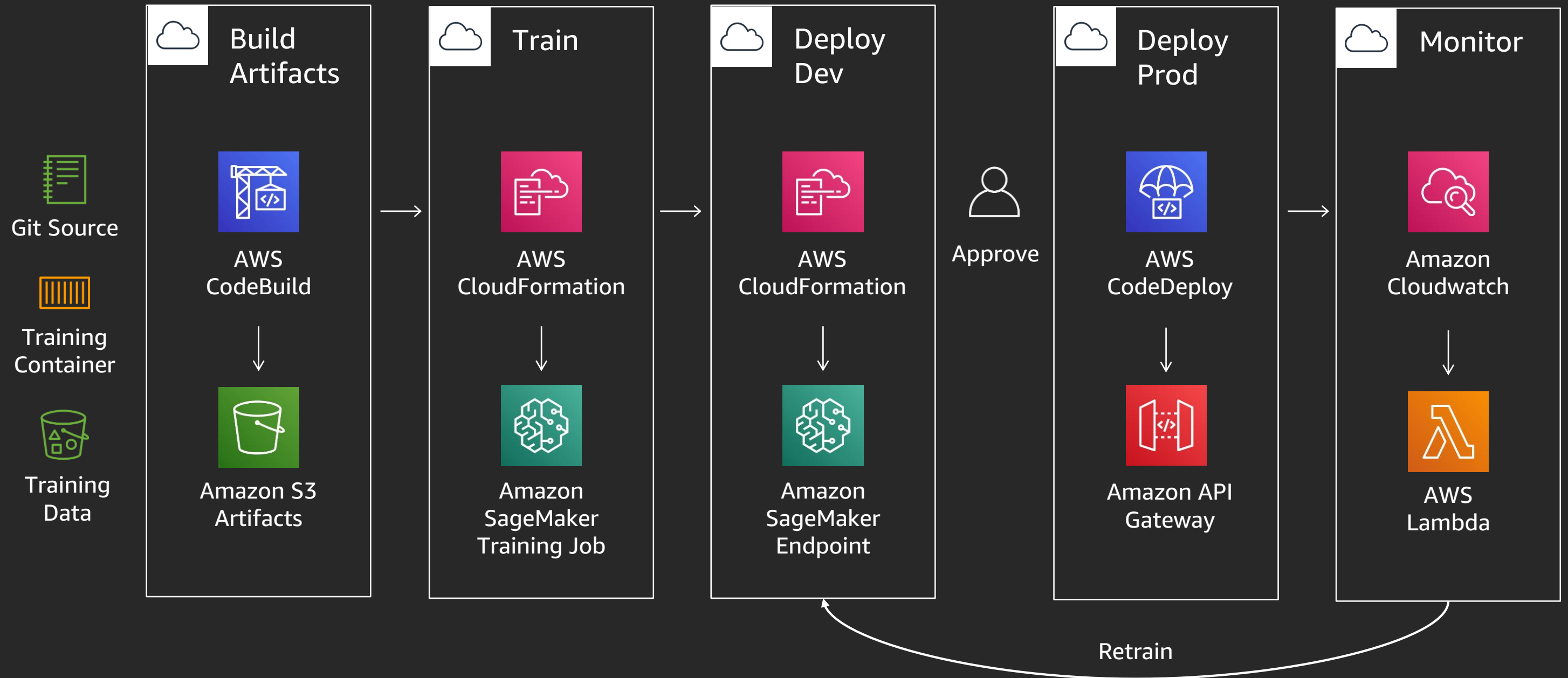
EcrSource: `sha256:8f65708592c0`
[a0630051](#) GitSource: Update to include EcrSource and DataSource inputs for BuildTemplates
DataSource: Amazon S3 version id: `R52G.D02Z22ZfdV0GbSe_5UTVcd5NHfz`

Disable transition

Build In progress
Execution ID: [f4afefba-2b0a-432a-9546-5919a50260f8](#)

BuildTemplates
AWS CodeBuild
In progress - Just now Details

Serverless Continuous Integration and Deployment



AWS CodeBuild build specification

version: 0.2

phases:

install:

runtime-versions:

python: 3.7

commands:

- pip install boto3

build:

commands:

- python run.py --pipeline-name=\${CODEBUILD_INITIATOR#codepipeline/}

post_build:

commands:

- aws cloudformation package --template-file assets/deploy-prd.yml
--output-template-file assets/template-prd.yml --s3-bucket \$ARTIFACT_BUCKET

artifacts:

files:

- assets/*

discard-paths: yes

Code Build Environment

Install python libraries

Run python script

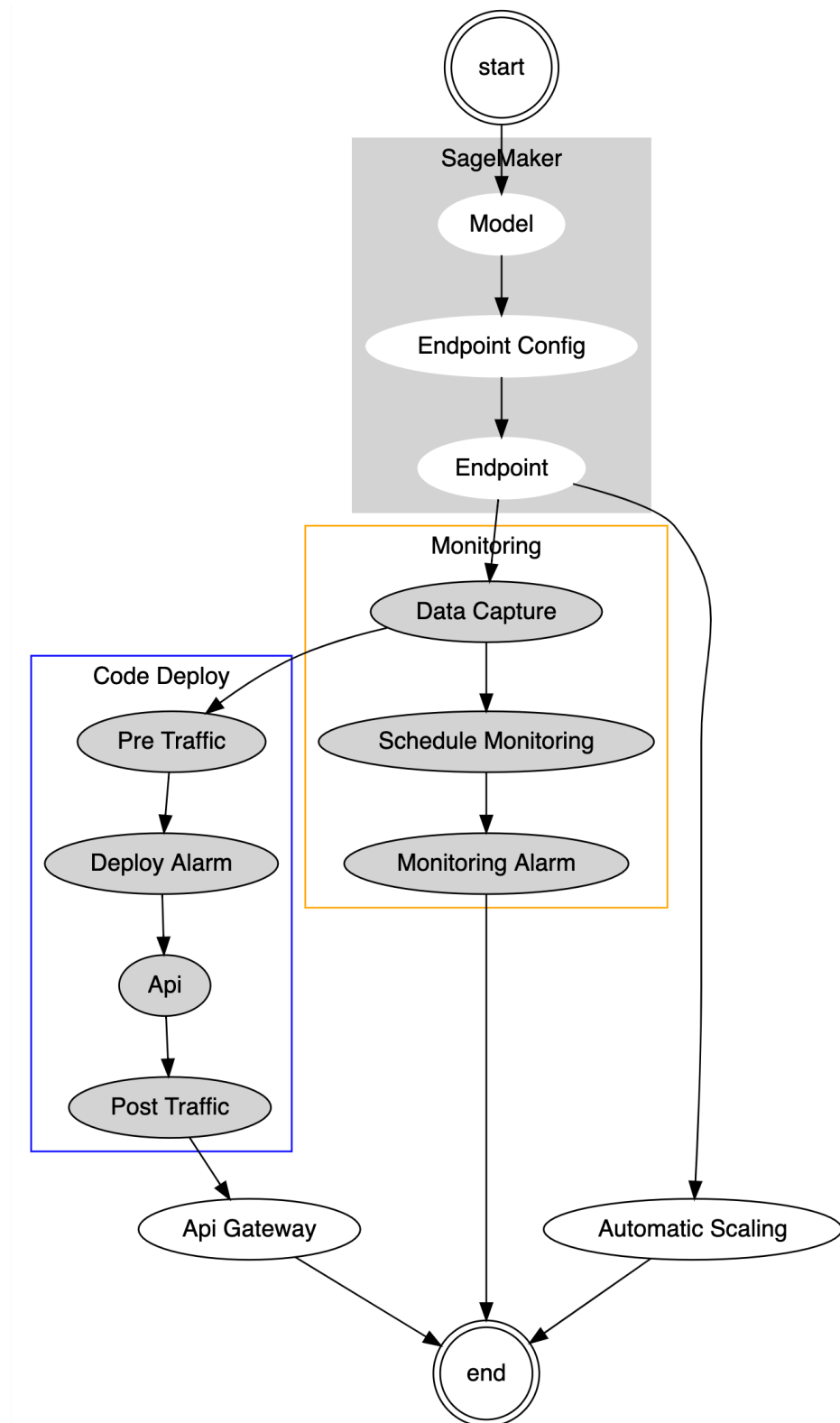
Package Cloud Formation Template

Output artifacts

AWS CloudFormation Prod Deployment

Extend Dev CloudFormation to include
Blue/Green Deployment and Monitoring

1. Create Amazon SageMaker Endpoint
2. Enable SageMaker Data Capture, Schedule Monitoring and Alarms
3. Execute AWS CodeDeploy Blue/Green Lambda deployment
4. Update Amazon API Gateway and SageMaker Automatic Scaling



AWS CodeDeploy Blue/Green Lambda deployment

Deployment status

Step 1

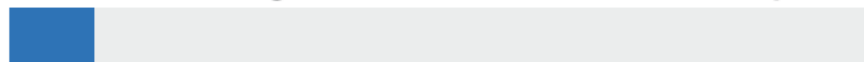
Pre-deployment validation Completed



✓ Succeeded

Step 2

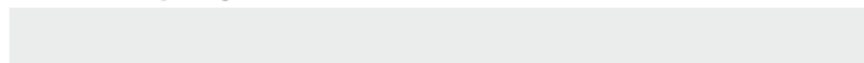
Traffic shifting 10% complete



• In progress

Step 3

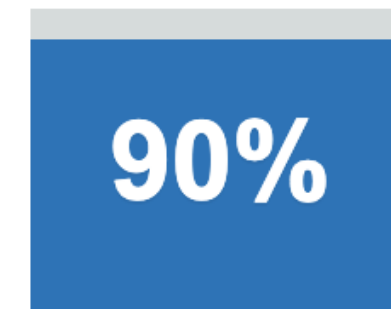
Post-deployment validation Not started



Traffic shifting progress

Next: The deployment will shift 90% of traffic from the current version to the replacement version at approximately 5 minute(s) after the deployment started.

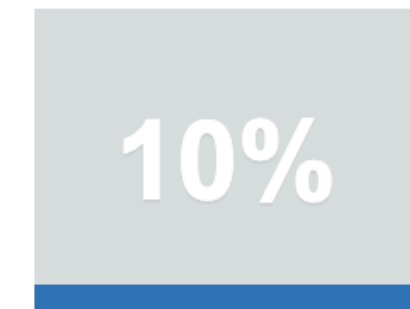
Original



Deployment results Info

90% of traffic

Replacement



10% of traffic

AWS CloudFormation custom resource Lambda

@helper.create

@helper.update

```
def create_handler(event, context):
```

```
    # Call boto3 to enable data capture
```

```
    return update_endpoint(event)
```

@helper.delete

```
def delete_handler(event, context):
```

```
    delete_endpoint_config(event)
```

@helper.poll_create

@helper.poll_update

```
def poll_create(event, context):
```

```
    endpoint_name = get_endpoint_name(event)
```

```
    return is_endpoint_ready(endpoint_name)
```

<https://github.com/aws-cloudformation/custom-resource-helper>

Amazon SageMaker Endpoint Data Capture sample

```
{
  "captureData": {
    "endpointInput": {
      "observedContentType": "text/csv",
      "mode": "INPUT",
      "data": "text,price\nspotify p0d3d89b19 sydney au,11.99",
      "encoding": "CSV"
    },
    "endpointOutput": {
      "observedContentType": "text/csv",
      "mode": "OUTPUT",
      "data": "__label__entertainment,0.0000068826,0.5860334635,0.0012867898...\n",
      "encoding": "CSV"
    }
  }
}
```

Amazon SageMaker Model Monitoring

Concept drift detection is a two stage process orchestrated by the production deployment.

1. Create a baseline from the dataset you used to train model
2. Schedule monitoring to measure model quality with open source Deequ library

<https://github.com/awslabs/deequ>

	data_type	completeness	baseline_drift	categorical_values
class_predictions	String	96.07%	45.92%	N/A
class_probabilities_	Fractional	100.00%	N/A	N/A
class_probabilities__label_eating_out	Fractional	100.00%	10.98%	N/A
class_probabilities__label_education	Fractional	24.53%	N/A	N/A
class_probabilities__label_entertainment	Fractional	100.00%	N/A	N/A
class_probabilities__label_fees_and_interest	Fractional	0.56%	N/A	N/A
class_probabilities__label_groceries	Fractional	100.00%	N/A	N/A
class_probabilities__label_health	Fractional	100.00%	N/A	N/A
class_probabilities__label_home	Fractional	100.00%	N/A	N/A
class_probabilities__label_shopping	Fractional	100.00%	N/A	N/A
class_probabilities__label_transport	Fractional	100.00%	N/A	N/A
class_probabilities__label_travel	Fractional	100.00%	N/A	N/A
class_probabilities__label_utilities	Fractional	0.56%	N/A	N/A
class_probability	Fractional	0.56%	N/A	N/A

Amazon Cloudwatch Metrics & Alarms

Our solution monitors

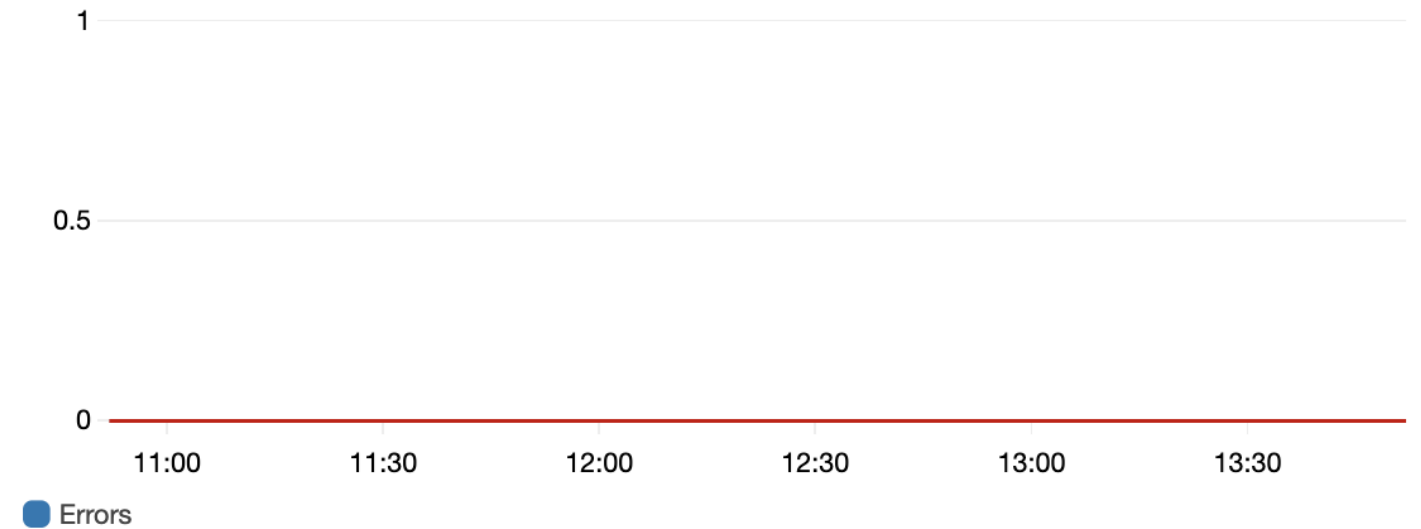
1. AWS CodeDeploy Blue/Green Deployment Alarms
2. Amazon SageMaker Endpoint Latency and Response Codes
3. Amazon SageMaker Model Monitoring Drift Alarm

Model re-training can be initiated on drift detection

Errors

Insufficient data

Errors > 0 for 2 datapoints within 2 minutes



Details

Name

mlops3-deploy-
AliasErrorMetricGreaterThanZeroAlarm-
54R12MNKQ80

Description

Lambda Function Error > 0

State

Insufficient data

Threshold

Errors > 0 for 2 datapoints within 2 minutes

Last change

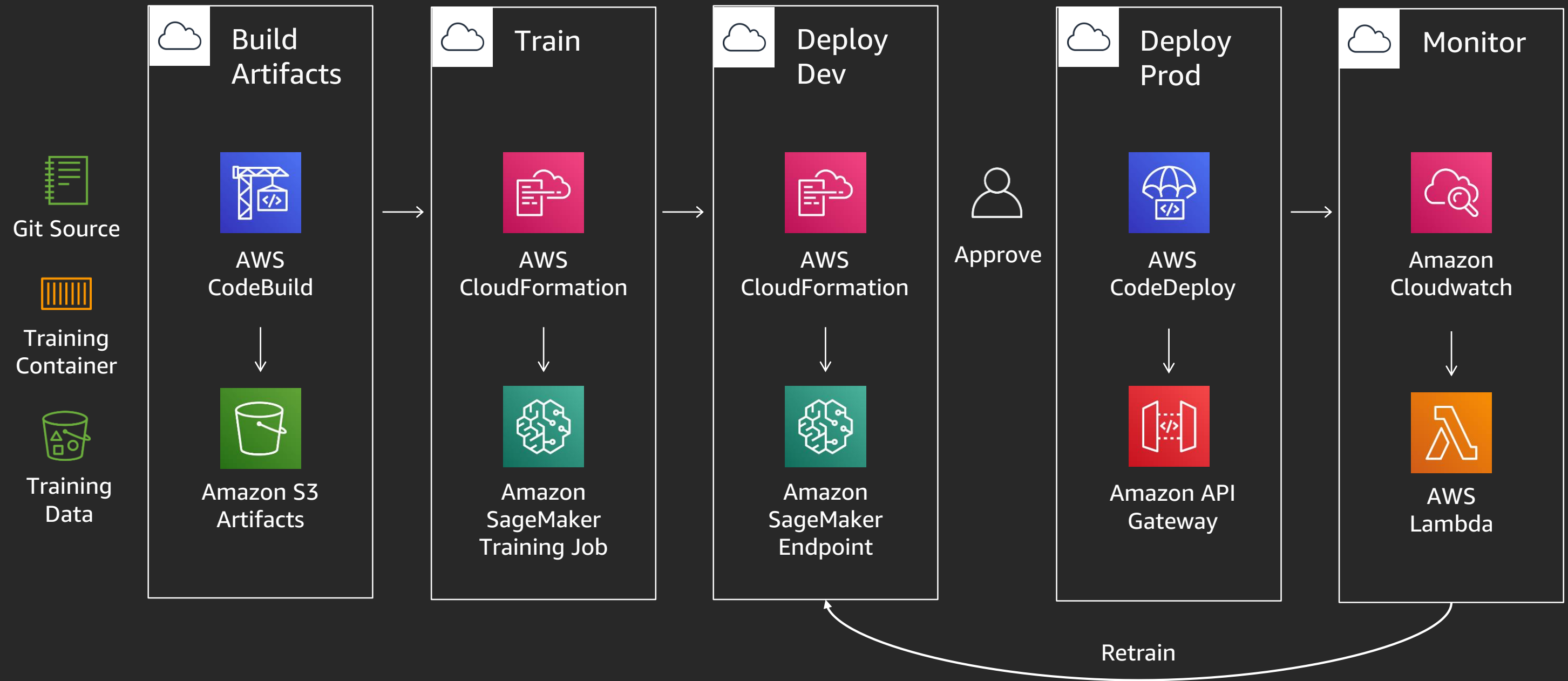
2020-02-14 15:32:28

ARN

arn:aws:cloudwatch:ap-southeast-
2:691313291965:alarm:mlops3-deploy-
AliasErrorMetricGreaterThanZeroAlarm-
54R12MNKQ80

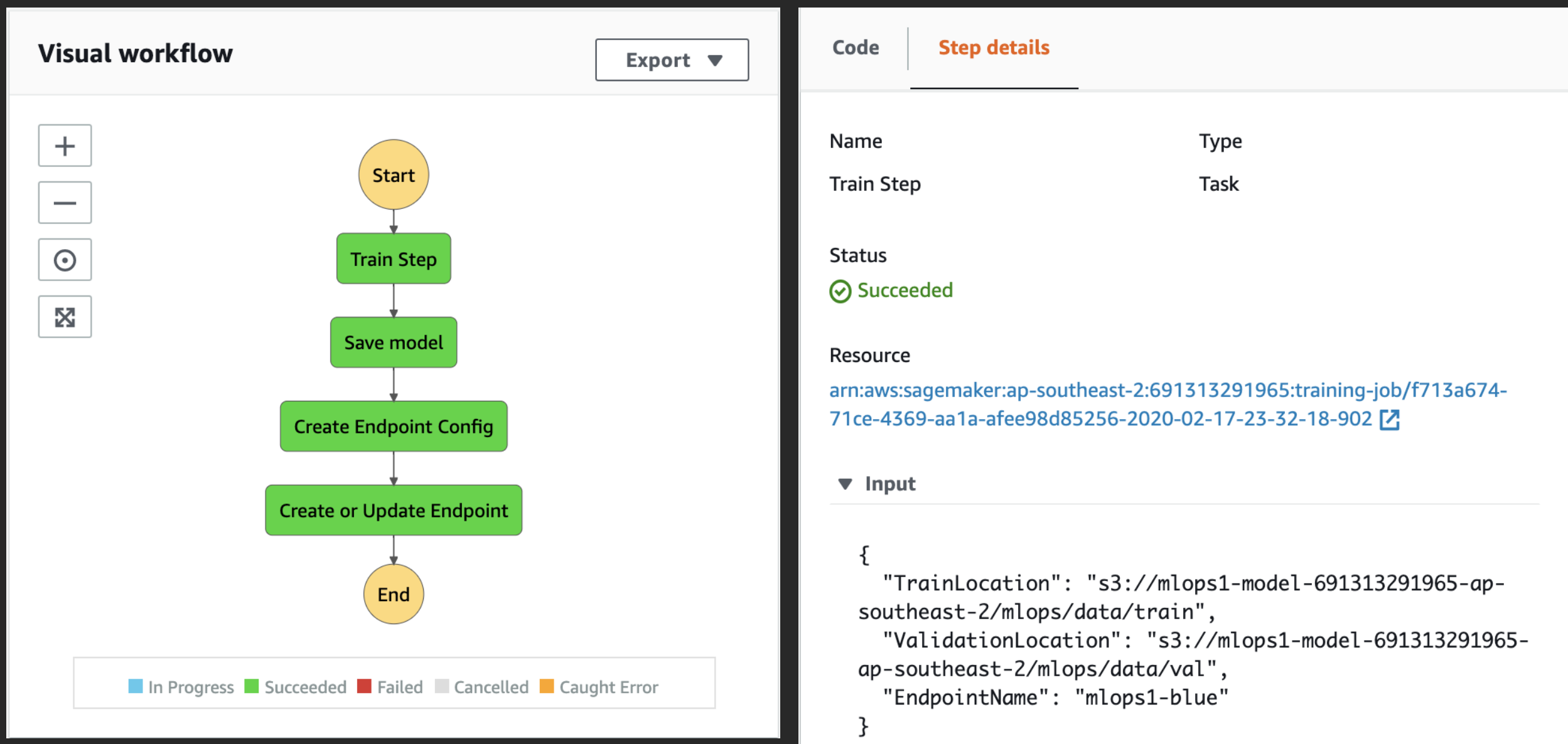
Demo

Recap: Serverless CI/CD



What about AWS Step Functions?

Example AWS Step Functions Workflow



AWS Step Functions Data Science SDK

Create workflow steps

```
training_step = steps.TrainingStep(  
    'Train Step',  
    estimator=xgb,  
    data={  
        'train': s3_input(inputs['TrainLoc']),  
        'validation': s3_input(inputs['ValLoc'])  
    },  
)
```

Combine into a workflow definition

```
workflow_def = steps.Chain([  
    training_step, model_step,  
    endpoint_config_step, endpoint_step  
])
```

Update workflow

```
workflow = Workflow.attach(workflow_arn)  
workflow.update(definition=workflow_def)
```

Execute workflow

```
execution = wf.execute(inputs={  
    'TrainLoc': s3_train_path,  
    'ValLoc': s3_val_path,  
    'EndpointName': 'mlops-blue'  
})
```

Orchestration approach summary

Considerations when comparing the following managed approaches

Role	CI/CD + Cloud Formation	DAG / Step functions
Continuous Integration	Source actions detect changes	AWS Step Functions service integrations poll for changes

Orchestration approach summary

Considerations when comparing the following managed approaches

Role	CI/CD + Cloud Formation	DAG / Step functions
Continuous Integration	Source actions detect changes	AWS Step Functions service integrations poll for changes
Staging actions	Native support for blocking at stages, and superseding source	Executions run independently, state management required

Orchestration approach summary

Considerations when comparing the following managed approaches

Role	CI/CD + Cloud Formation	DAG / Step functions
Continuous Integration	Source actions detect changes	AWS Step Functions service integrations poll for changes
Staging actions	Native support for blocking at stages, and superseding source	Executions run independently, state management required
Flow	Single pass with support for parallel actions	Full flexibility in designing loops, conditional logic, retries

Orchestration approach summary

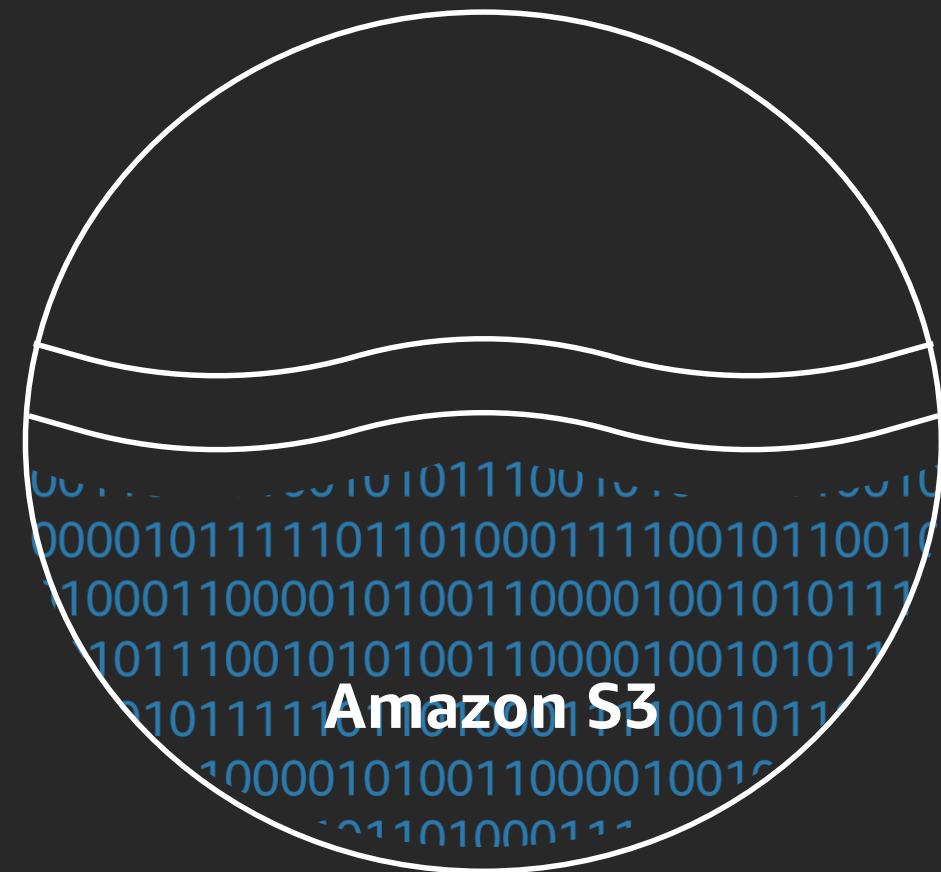
Considerations when comparing the following managed approaches

Role	CI/CD + Cloud Formation	DAG / Step functions
Continuous Integration	Source actions detect changes	AWS Step Functions service integrations poll for changes
Staging actions	Native support for blocking at stages, and superseding source	Executions run independently, state management required
Flow	Single pass with support for parallel actions	Full flexibility in designing loops, conditional logic, retries
Change control	Automatic rollbacks. Support for manual approval	Custom compensating actions required or error

Sidebar: Data integration options

Data Lineage and provenance is key for model reproducibility

- Versioned S3 Data Lake
- Apache Spark on AWS Glue or Amazon EMR
- Third Party Open Source
 - Hudi
 - Data Version Control (DVC)
 - Pachyderm



Call to action

MLOps is culture and technology working together

Automation increases your deployment velocity, and reduces costs

Leverage Managed Orchestration

Monitor and Alert on deployment lifecycle

Retrain on drift detection

Thank you!

Julian Bright

julbrigh@amazon.com