Machine Learning Workflows with Amazon SageMaker and AWS Step Functions

Julien Simon, Global Evangelist, AI & Machine Learning
@julsimon

March 2019

Today’s agenda

Build, train, and deploy machine learning models with Amazon SageMaker

Build serverless workflows with less code to write and maintain using AWS Step Functions

Learn how Cox Automotive combined SageMaker and Step Functions to improve collaboration between data scientists and software engineers

New features to build and manage ML workflows even faster

A quick introduction to Amazon SageMaker and AWS Step Functions

Amazon SageMaker: Build, Train, and Deploy ML Models at Scale

Collect and prepare training data

Choose and optimize your ML algorithm

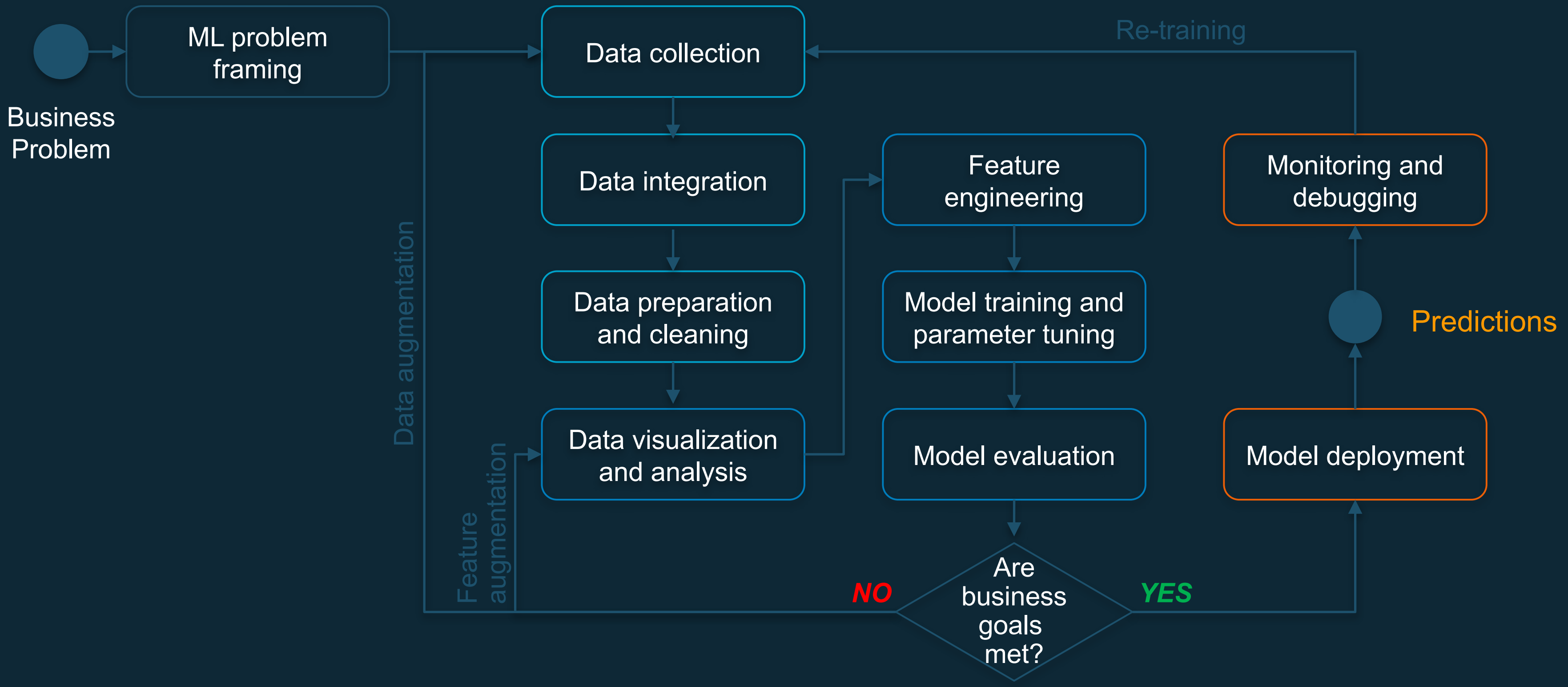
Set up and manage environments for training

Train and Tune ML Models

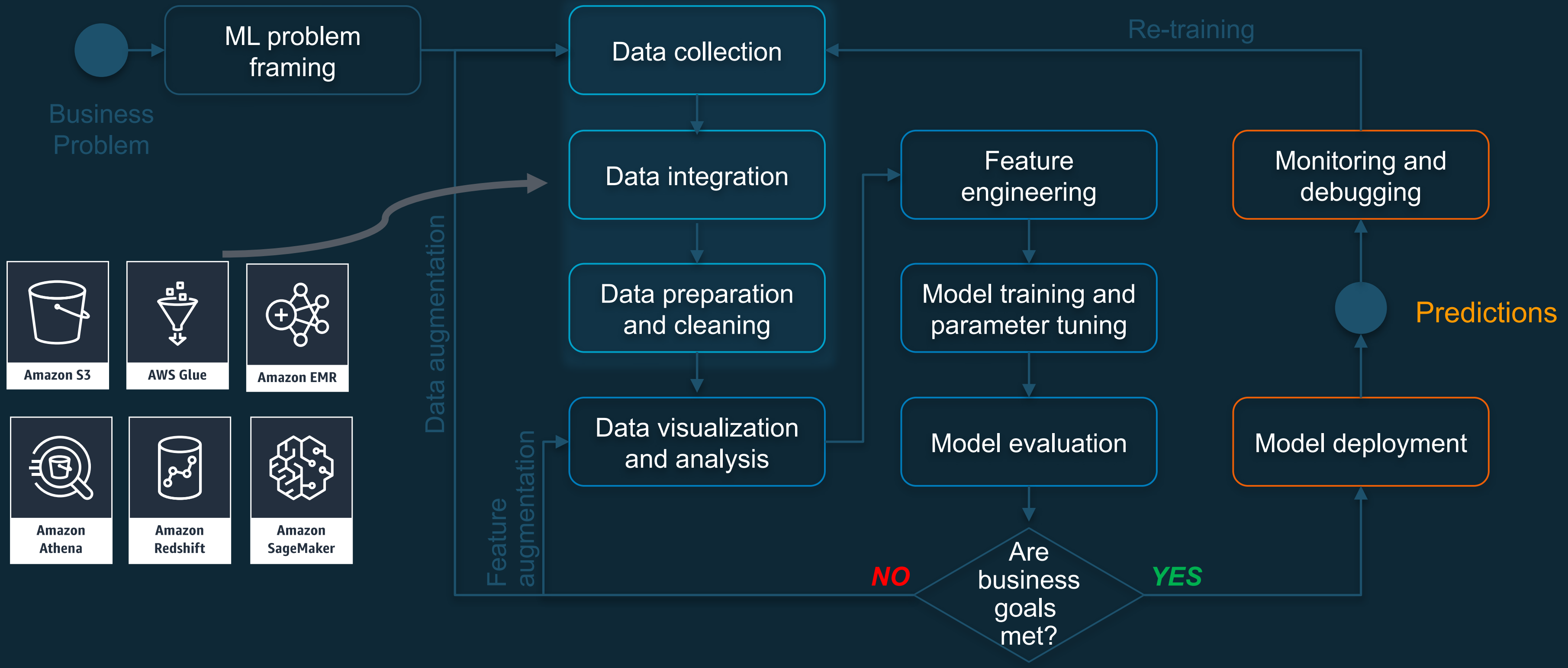
Deploy models in production

Scale and manage the production environment

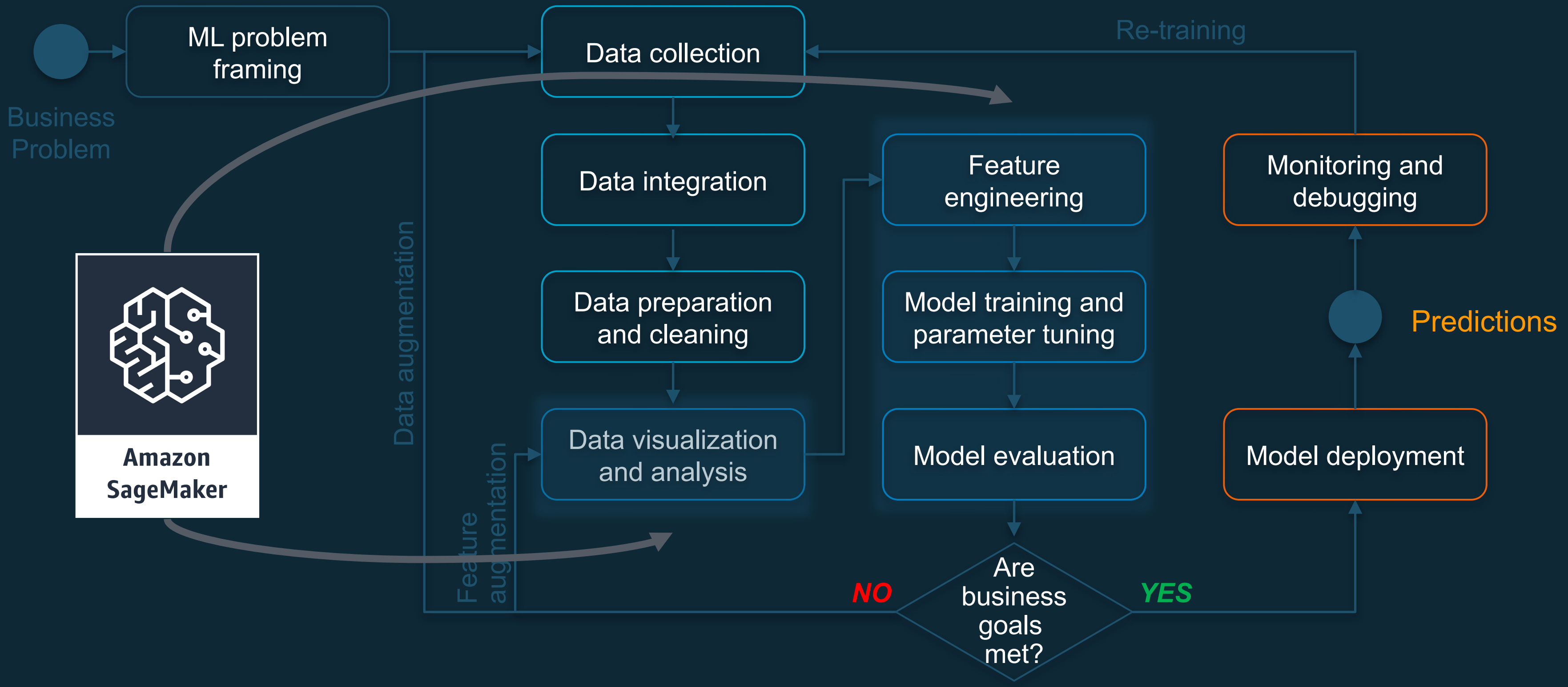
Machine learning cycle



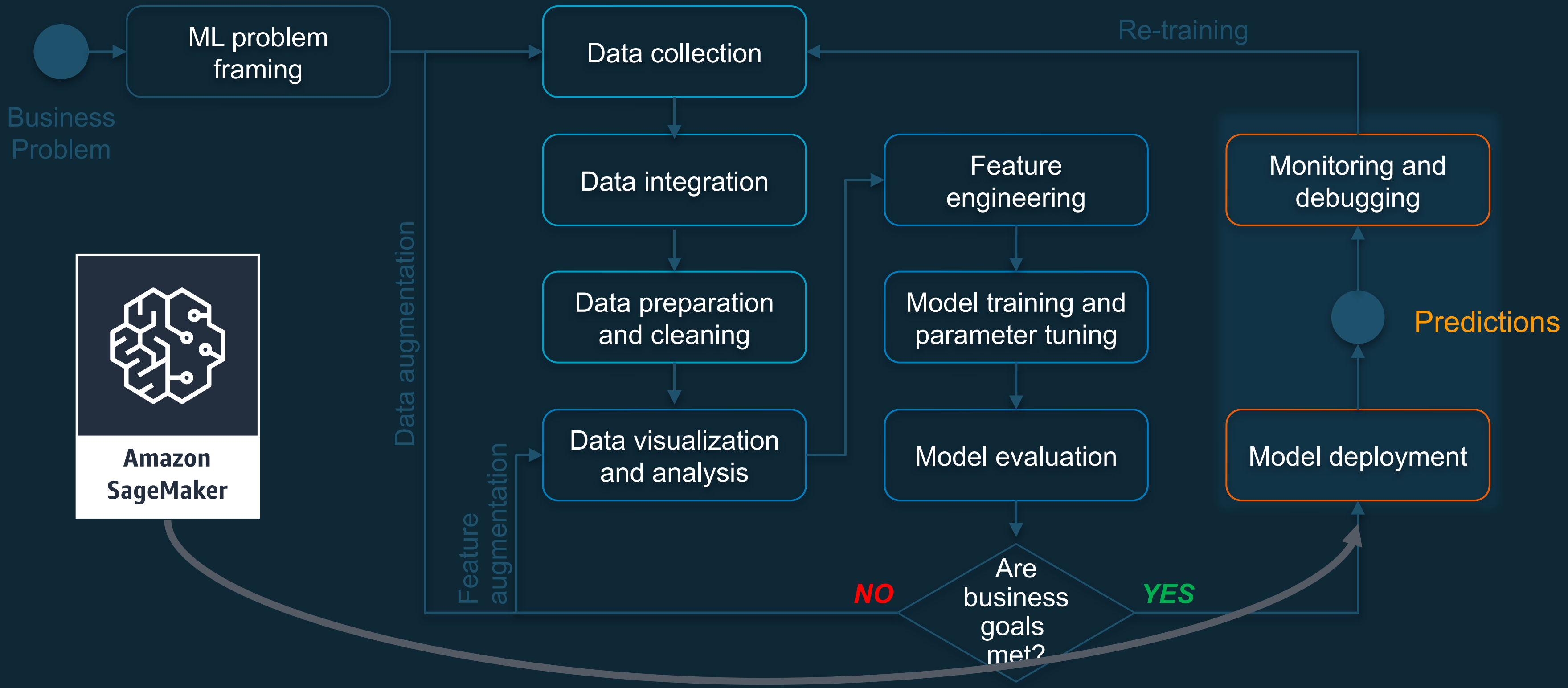
Manage data on AWS



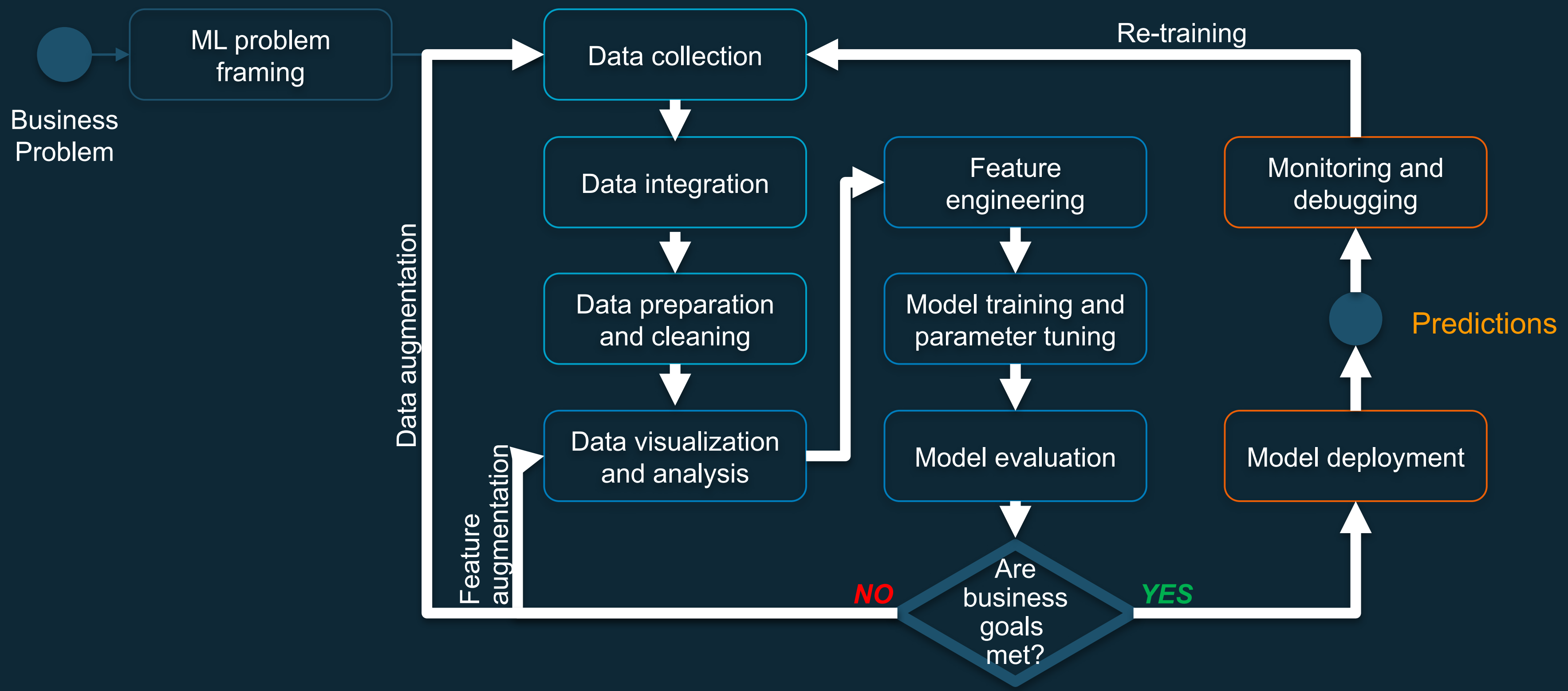
Build and train models using SageMaker



Deploy models using SageMaker

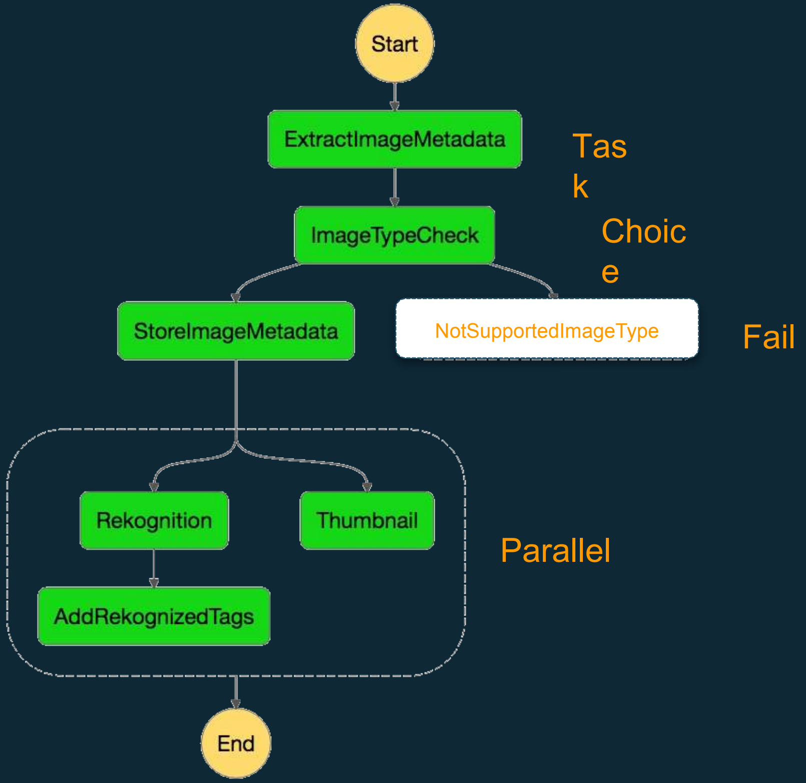


What about the lines between the steps?



What is AWS Step Functions?

Mountains
People
Snow

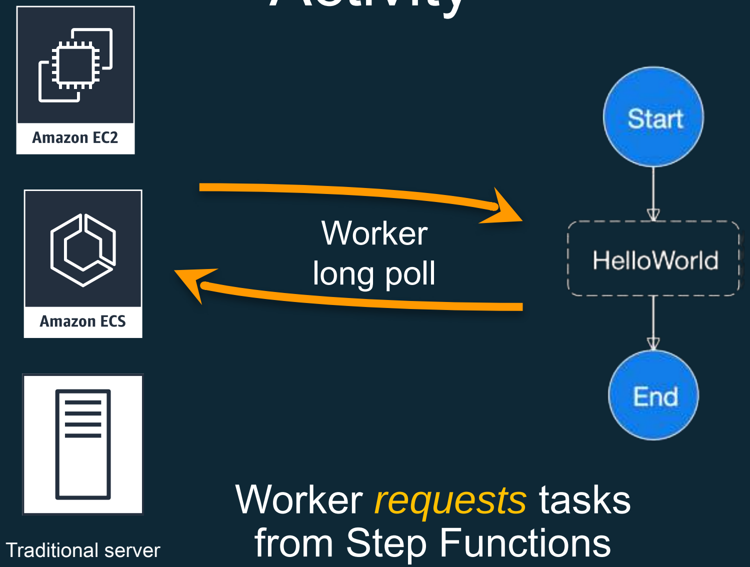


Step Functions uses Amazon States Language (JSON)

```
{
"Comment": "Image Processing workflow",
"StartAt": "ExtractImageMetadata",
"States": {
     "ExtractImageMetadata": {
     "Type": "Task",
     "Resource": "arn:aws:lambda:::function:photo-backendExtractImageMetadata-...",
     "InputPath": "$",
     "ResultPath": "$.extractedMetadata",
     "Next": "ImageTypeCheck",
     "Catch": [ {
          "ErrorEquals": [ "ImageIdentifyError"],
          "Next": "NotSupportedImageType"
          } ],
     "Retry": [ {
          "ErrorEquals": [ "States.ALL"],
          "IntervalSeconds": 1,
          "MaxAttempts": 2,
          "BackoffRate": 1.5 }, ...
```

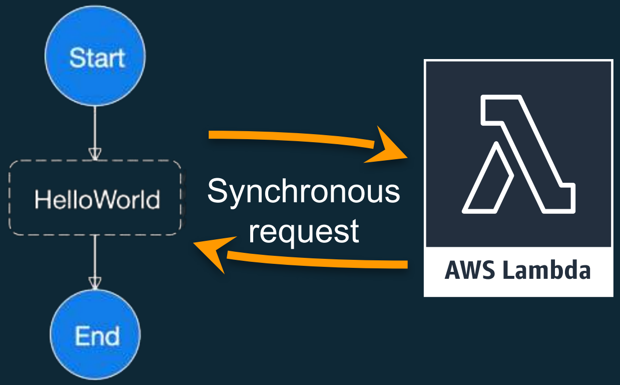
Run tasks with any compute resource

Activity



Worker *requests* tasks
from Step Functions

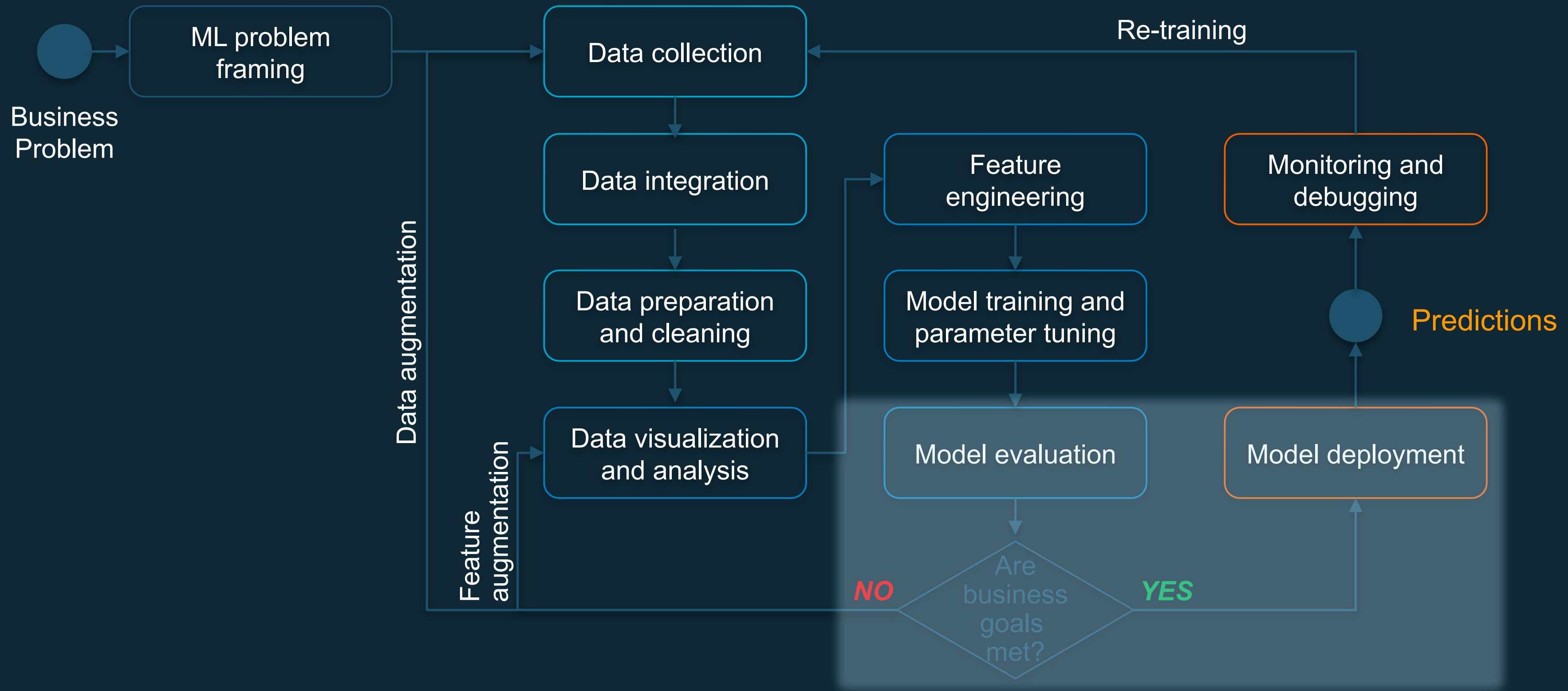
AWS Lambda function



Step Functions *invokes*
Lambda function

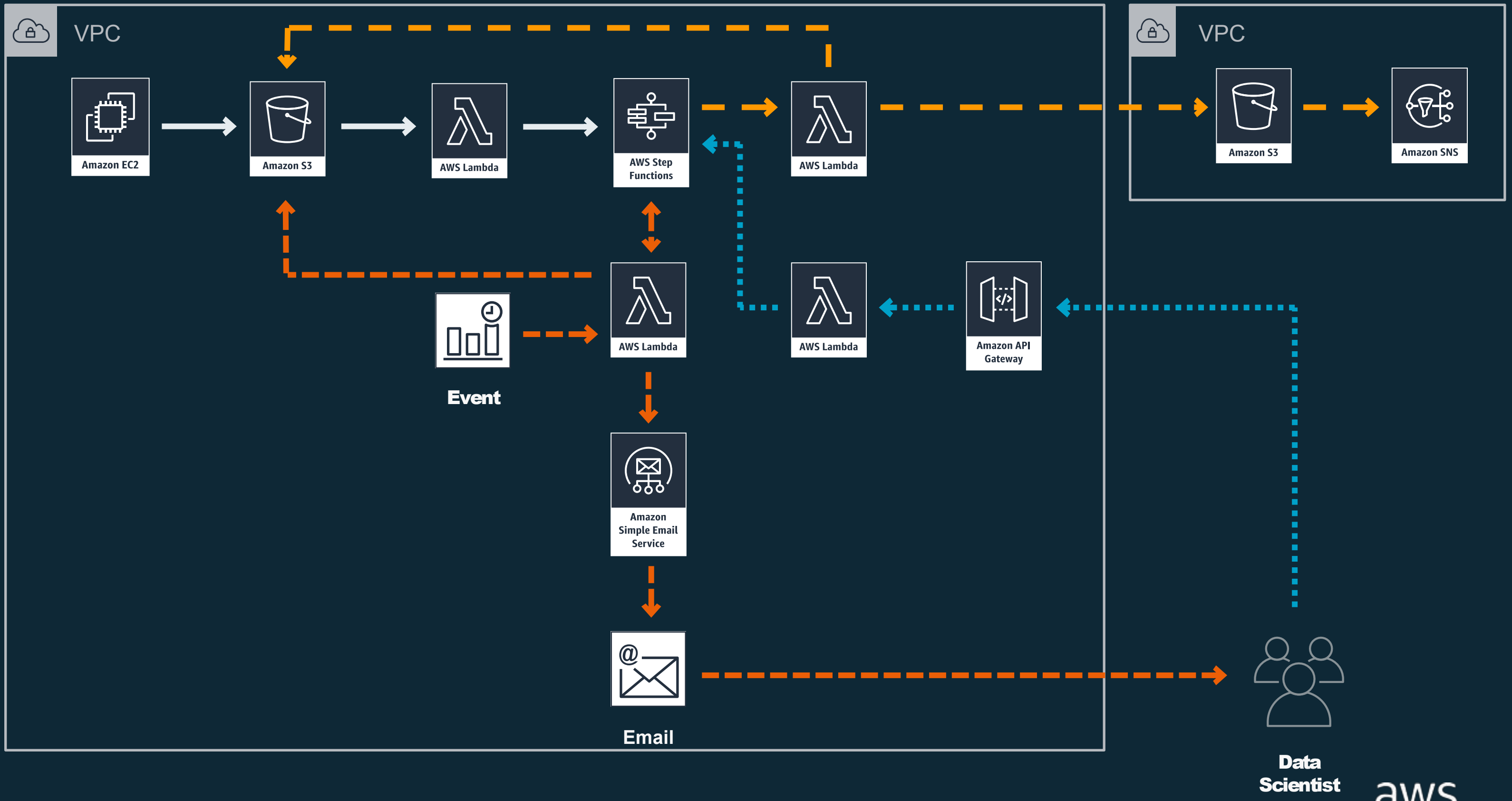
Case study: Cox Automotive

Machine learning cycle

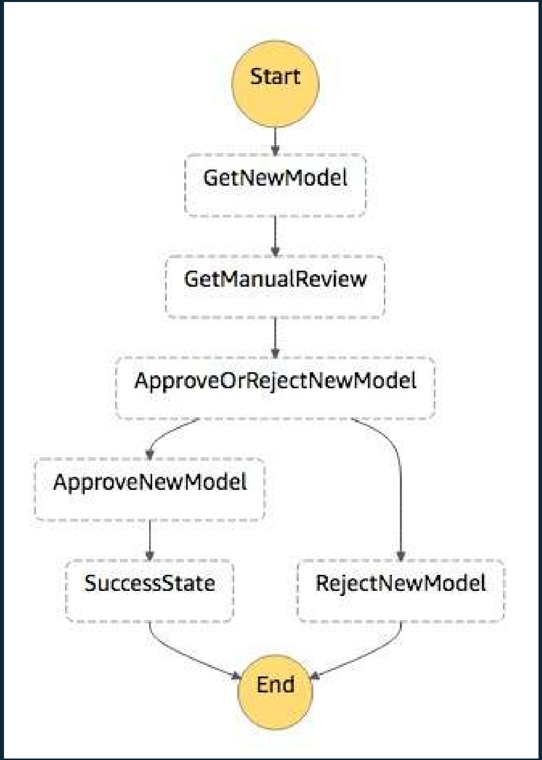


How can we enable our Decision Science team to deliver a model in a way that doesn’t require any work to ingest the model in our deployment pipeline?

Amazon SageMaker model deployment pipeline

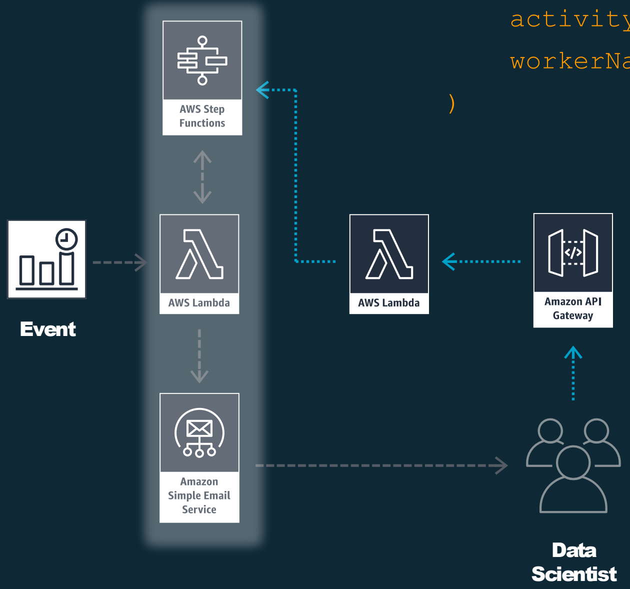


AWS Step Functions state machine definition



```
…
  "StartAt": "GetNewModel",
  "States": {
    "GetNewModel": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:${region}:${act}:
function:model-review-GetNewModelFunction",
      "ResultPath": "$",
      "Next": "GetManualReview"
    },
    "GetManualReview": {
      "Type": "Task",
      "Resource": "arn:aws:states:${region}:${act}:
activity:model-review-getModelReviewDecision",
      "ResultPath": "$.taskresult",
      "TimeoutSeconds": 604800,
      "Next": "ApproveOrRejectNewModel”
    },
   …
```

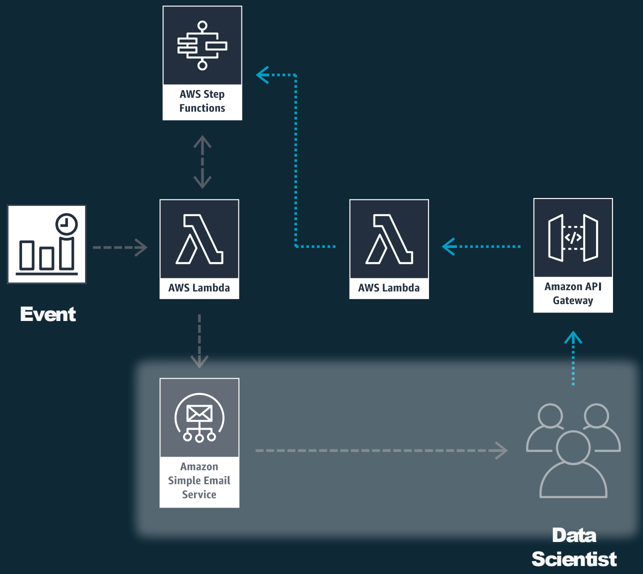
Activity token journey: Send models for review

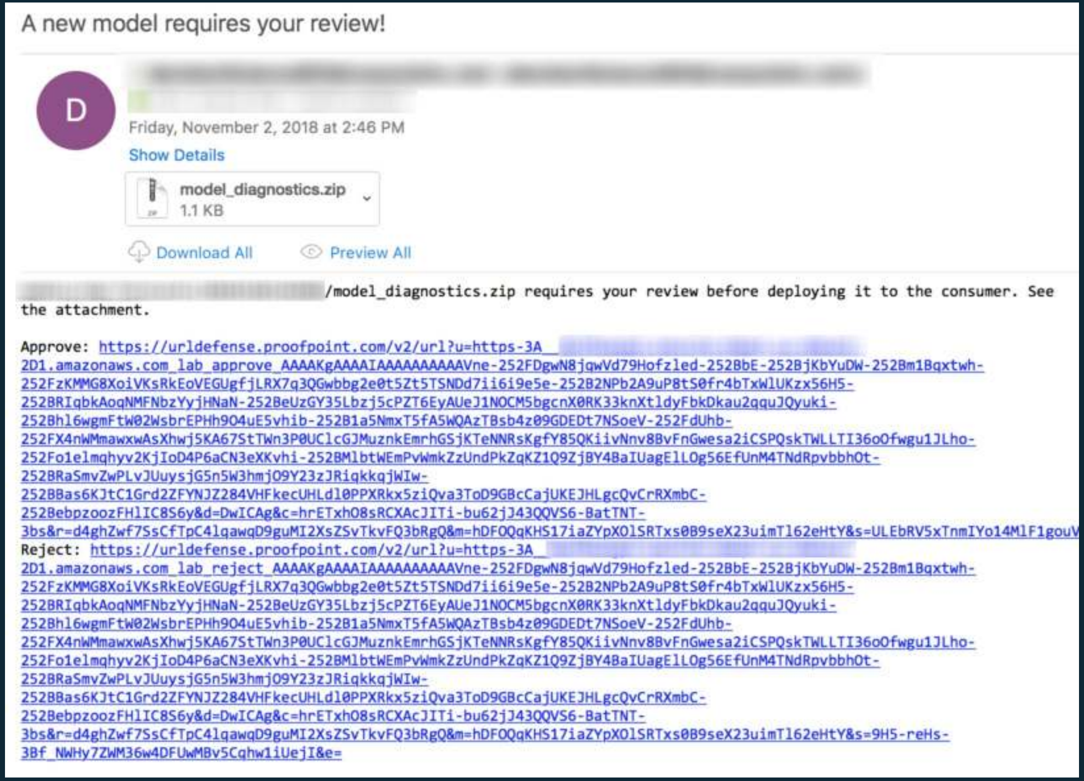


```
sfnClient = boto3.client('stepfunctions')
getActivityTaskResponse = sfnClient.get_activity_task(
    activityArn=activityArn,
    workerName='checkStateMachineActivityStatus’
)
```

```
taskToken = getActivityTaskResponse['taskToken’]
sendEmail(taskToken, diagnosticsFileName, diagnosticsFile,
diagnosticsFilePath, apiUrl)
…
def sendEmail(taskToken, diagnosticsFileName,
diagnosticsFile, diagnosticsFilePath, apiUrl):
    sesClient = boto3.client('ses')
    encodedtaskToken = quote(taskToken, safe='')
    approveLink = apiUrl + '/approve/' + encodedtaskToken
    rejectLink = apiUrl + '/reject/' + encodedtaskToken
```

Activity token journey: generate review request




A new model requires your review!

D [redacted]

Friday, November 2, 2018 at 2:46 PM

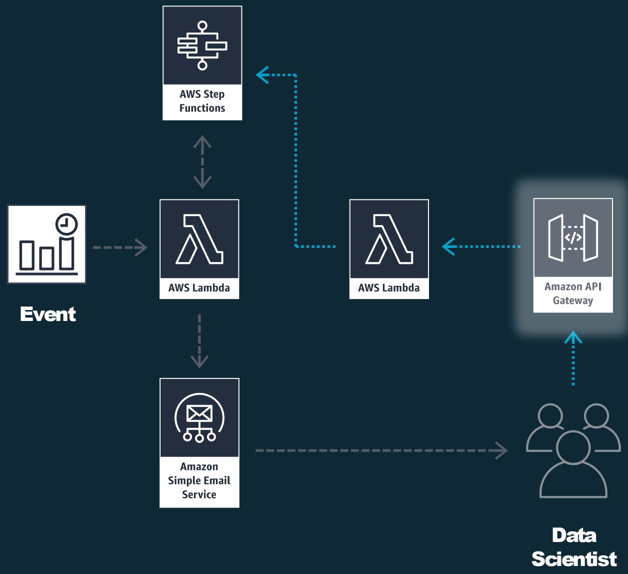
Show Details

model_diagnostics.zip
1.1 KB

Download All Preview All

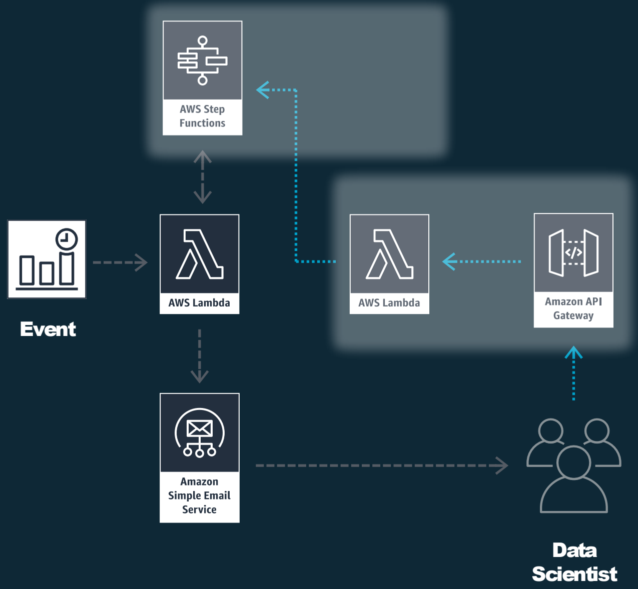
```
[redacted]/model_diagnostics.zip requires your review before deploying it to the consumer. See
the attachment.

Approve: https://urldefense.proofpoint.com/v2/url?u=https-3A__[redacted]
2D1.amazonaws.com_lab_approve_AAAAKgAAAAIAAAAAAAAAAVne-252FDgwN8jqwVd79Hofzled-252BbE-252BjKbYuDW-252Bm1Bqxtwh-
252FzKMMG8XoiVKsRkEoVEGUgfjLRX7q3QGwbbg2e0t5Zt5TSNDd7ii6i9e5e-252B2NPb2A9uP8tS0fr4bTxWlUKzx56H5-
252BRIqbkAoqNMFNbzYyjHNaN-252BeUzGY35Lbzj5cPZT6EyAUeJ1NOCM5bgcnX0RK33knXtldyFbkDkau2qquJQyuki-
252Bhl6wgmFtW02WsbrEPHh9O4uE5vhib-252B1a5NmxT5fA5WQAzTBsb4z09GDEDt7NSoeV-252FdUhb-
252FX4nWMmawxwAsXhwj5KA67StTWn3P0UClcGJMuznkEmrhGSjKTeNNRsKgfY85QKiivNnv8BvFnGwesa2iCSPQskTWLLTI36oOfwgu1JLho-
252Fo1elmqhyv2KjIoD4P6aCN3eXKvhi-252BMlbtWEmPvWmkZzUndPkZqKZ1Q9ZjBY4BaIUagElLOg56EfUnM4TNdRpvbbhOt-
252BRaSmvZwPLvJUuysjG5n5W3hmjO9Y23zJRiqkkqjWIw-
252BBas6KJtC1Grd2ZFYNJZ284VHFkecUHLdl0PPXRkx5ziQva3ToD9GBcCajUKEJHLgcQvCrRXmbC-
252BebpzoozFHlIC8S6y&d=DwICAg&c=hrETxhO8sRCXAcJITi-bu62jJ43QQVS6-BatTNT-
3bs&r=d4ghZwf7SsCfTpC4lqawqD9guMI2XsZSvTkvFQ3bRgQ&m=hDFOQqKHS17iaZYpXOlSRTxs0B9seX23uimTl62eHtY&s=ULEbRV5xTnmIYo14MlF1gouV
Reject: https://urldefense.proofpoint.com/v2/url?u=https-3A__[redacted]
2D1.amazonaws.com_lab_reject_AAAAKgAAAAIAAAAAAAAAAVne-252FDgwN8jqwVd79Hofzled-252BbE-252BjKbYuDW-252Bm1Bqxtwh-
252FzKMMG8XoiVKsRkEoVEGUgfjLRX7q3QGwbbg2e0t5Zt5TSNDd7ii6i9e5e-252B2NPb2A9uP8tS0fr4bTxWlUKzx56H5-
252BRIqbkAoqNMFNbzYyjHNaN-252BeUzGY35Lbzj5cPZT6EyAUeJ1NOCM5bgcnX0RK33knXtldyFbkDkau2qquJQyuki-
252Bhl6wgmFtW02WsbrEPHh9O4uE5vhib-252B1a5NmxT5fA5WQAzTBsb4z09GDEDt7NSoeV-252FdUhb-
252FX4nWMmawxwAsXhwj5KA67StTWn3P0UClcGJMuznkEmrhGSjKTeNNRsKgfY85QKiivNnv8BvFnGwesa2iCSPQskTWLLTI36oOfwgu1JLho-
252Fo1elmqhyv2KjIoD4P6aCN3eXKvhi-252BMlbtWEmPvWmkZzUndPkZqKZ1Q9ZjBY4BaIUagElLOg56EfUnM4TNdRpvbbhOt-
252BRaSmvZwPLvJUuysjG5n5W3hmjO9Y23zJRiqkkqjWIw-
252BBas6KJtC1Grd2ZFYNJZ284VHFkecUHLdl0PPXRkx5ziQva3ToD9GBcCajUKEJHLgcQvCrRXmbC-
252BebpzoozFHlIC8S6y&d=DwICAg&c=hrETxhO8sRCXAcJITi-bu62jJ43QQVS6-BatTNT-
3bs&r=d4ghZwf7SsCfTpC4lqawqD9guMI2XsZSvTkvFQ3bRgQ&m=hDFOQqKHS17iaZYpXOlSRTxs0B9seX23uimTl62eHtY&s=9H5-reHs-
3Bf_NWHy7ZWM36w4DFUwMBv5Cqhw1iUejI&e=
```

Amazon API Gateway configuration



```
GetReviewDecisionFunction:
    handler: handler.getReviewDecision
    role: "${self:custom.terraformed.service.role}"
    events:
      - http:
          path: approve/{taskToken}
          method: get
          request:
            parameters:
              paths:
                taskToken: true
      - http:
          path: reject/{taskToken}
          method: get
          request:
            parameters:
              paths:
                taskToken: true
```

Activity token journey: prepare arguments & output



```
path = event['path']
taskToken = unquote(event['pathParameters']['taskToken'])
taskSuccessOutput = '{"decision": "Approved"}'
taskFailureOutput = '{"decision": "Rejected"}'

if path.startswith('/reject'):
    message = "The model has been rejected and will not be promoted"
    status = 'rejected'
    kwargs = {
        'taskToken': taskToken,
        'output': taskFailureOutput
    }
else:
    if path.startswith('/approve'):
        message = "The model has been approved and will be promoted"
        status = 'approved'
        kwargs = {
            'taskToken': taskToken,
            'output': taskSuccessOutput
        }
    else:
        message = "The parameter does not match the expected parameter"
        print(message)
```

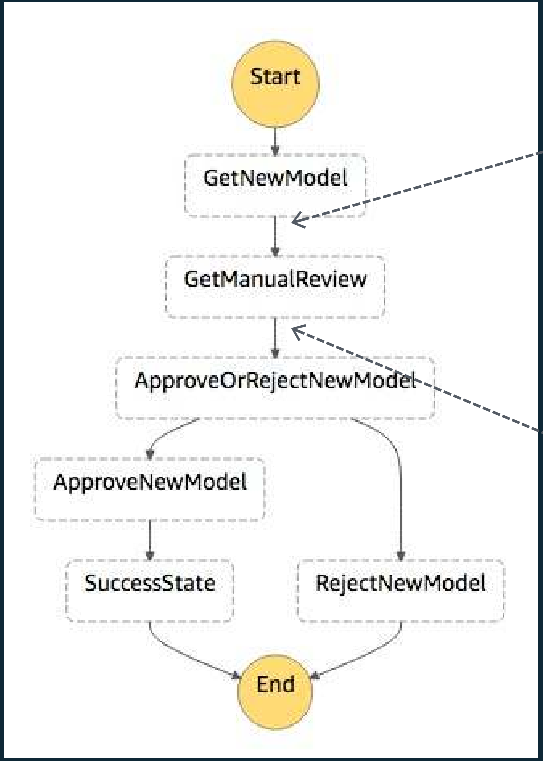
State input & output processing

Lambda state can be shared downstream via the state output, which is a mutable JSON object used to carry inputs & output data between states.

Benefits

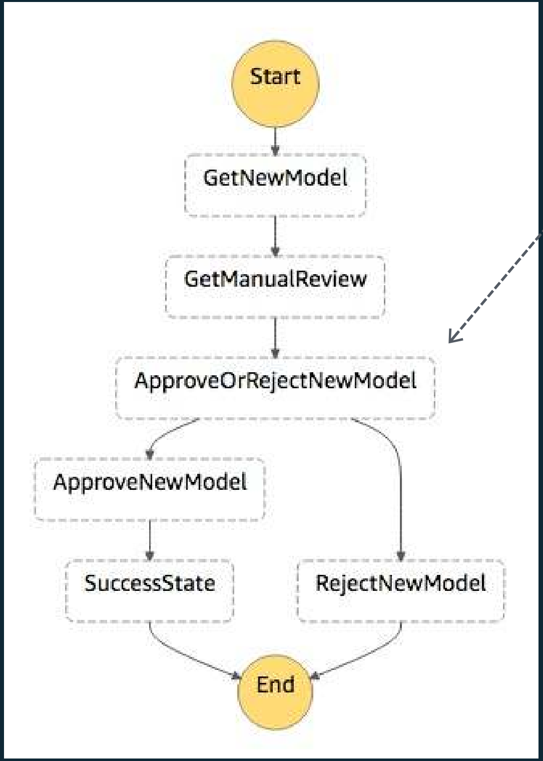
- Upstream worker output can be used as input for downstream workers (to reduce the number of repeat calls)
- Maintain state of upstream states

State input & output processing: append to output



```
{
  "name": "GetNewModel”,
  "output": {
    "diagnosticsFilePath": “20181102/model_diagnostics.zip",
    "diagnosticsFileName": "model_diagnostics.zip”
  }
}

# State is configured to append the decision to its input
{
  "name": "GetManualReview",
  "output": {
    "diagnosticsFilePath": "20181102/model_diagnostics.zip",
    "diagnosticsFileName": "model_diagnostics.zip",
    "taskresult": {
      "decision": "Approved"
    }
  }
}
```

State input & output processing: choice states



```
"ApproveOrRejectNewModel": {
  "Type": "Choice",
  "Choices": [
    {
      "Variable": "$.taskresult.decision",
      "StringEquals": "Approved",
      "Next": "ApproveNewModel"
    },
    {
      "Variable": "$.taskresult.decision",
      "StringEquals": "Rejected",
      "Next": "RejectNewModel"
    }
  ]
}
```

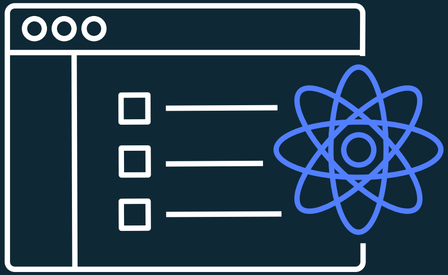
Back to our story…

Amazon SageMaker

AWS Step Functions

Example ML workflow

Amazon SageMaker Notebook



Retrieve data

```
def upload_to_s3(channel, file):
    s3 = boto3.resource('s3')
    data = open(file, "rb")
    key = channel + '/' + file
    s3.Bucket(bucket).put_object(Key=key, Body=data)
```

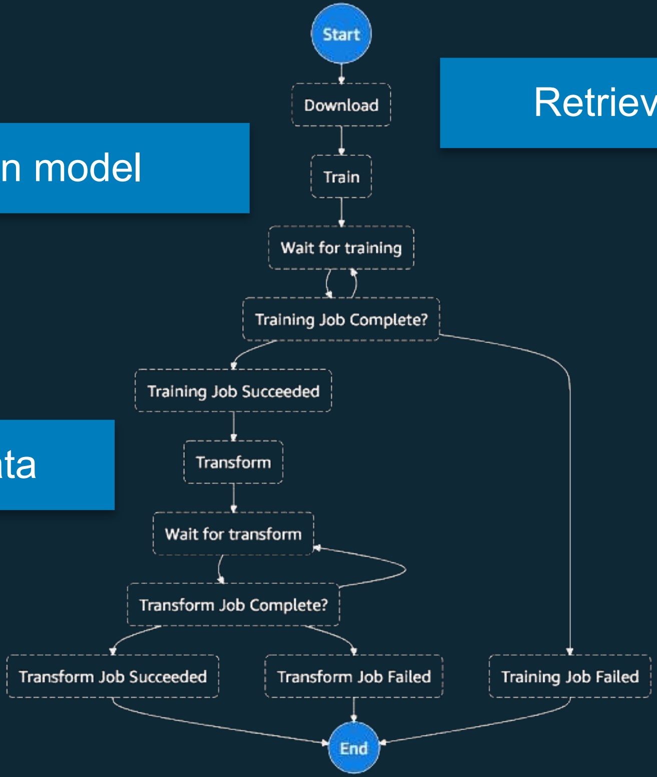
Train model

```
train = sagemaker.s3_input('s3://{}/train/'.format(bucket), content_type='application/x-recordio')
validation = sagemaker.s3_input('s3://{}/validation/'.format(bucket),
                content_type='application/x-recordio')
```

Transform data

```
input_data = 's3://batch-test-data/caltech256/'
output_data = 's3://batch-test-output/DEMO-image-classification'

transformer = training_job.transformer(2,  'ml.p3.2xlarge', output_path=output_data,
                                    assemble_with='Line', max_payload=8, max_concurrent_transforms=8)
transformer.transform(input_data, content_type='application/x-image')
```

ML workflow in Step Functions



Retrieve data

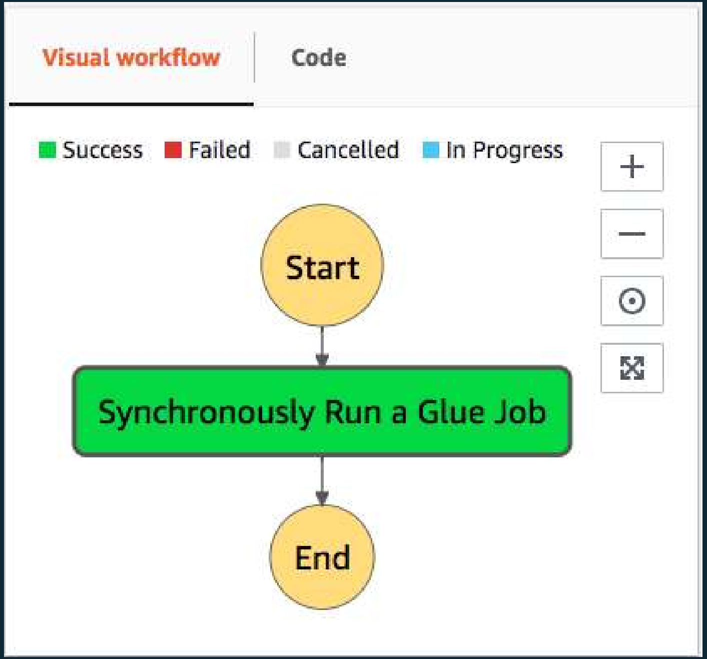
Train model

Transform data

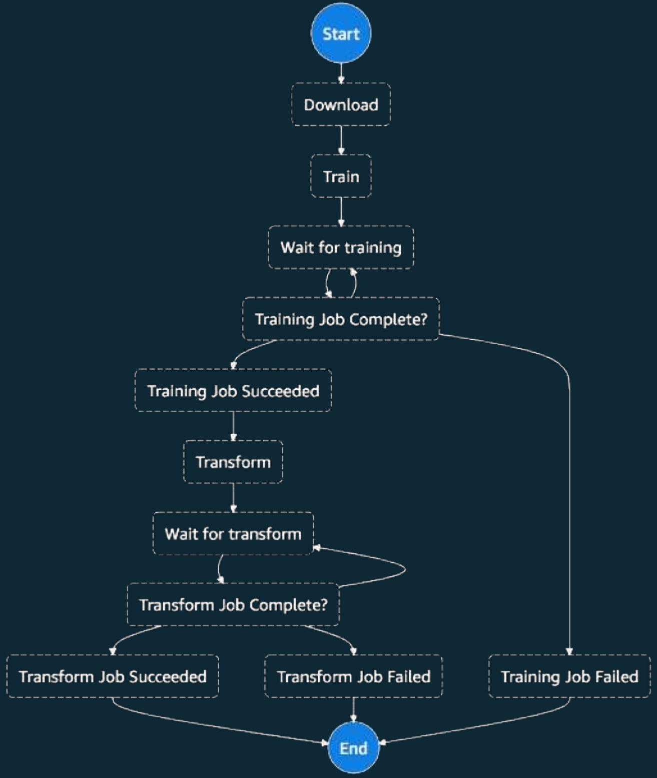
Manage asynchronous jobs without writing code!

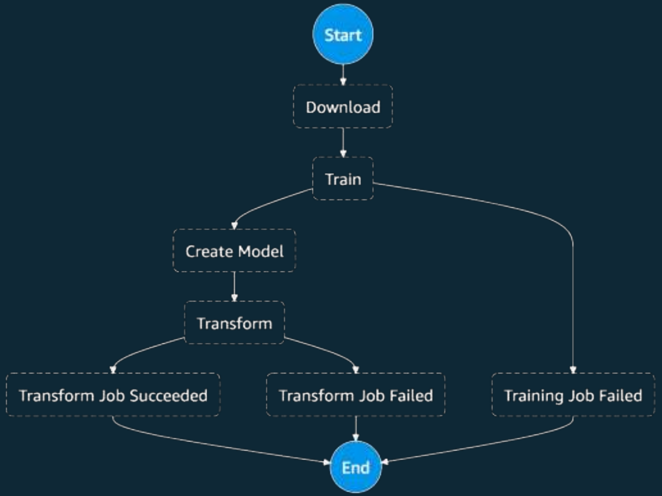
AWS
Glue

Amazon
SageMaker



Simplify machine learning workflows





Add AWS Glue ETL jobs in your workflows

```
"Synchronously Run a Glue Job": {
    "Type": "Task",
    "Resource": "arn:aws:states:::glue:startJobRun.sync",
    "Parameters":
          {
          "JobName.$": "$.myJobName”,
          “AllocatedCapacity”: 3
          },
    "Catch": [
      {"ErrorEquals": ["States.TaskFailed"],
       "ResultPath": "$.cause",
       "Next" : "Notify on Error"
       } ],
    "ResultPath": "$.jobInfo",
    "Next": "Report Success"
 }
```

Add Amazon SageMaker jobs in your workflows

```
"Synchronously Run a Training Job": {
    "Type": "Task",
    "Resource":
"arn:aws:states:::sagemaker.createTrainingJob.sync",
    "Parameters":
           {
      "AlgorithmSpecification": {...},
         "HyperParameters": {...},
      "InputDataConfig": [...],
      ...
           },
    "Catch": [
      {"ErrorEquals": ["States.TaskFailed"],
       "ResultPath": "$.cause",
       "Next" : ”Notify on Error"
       } ],
    "ResultPath": "$.jobInfo",
    "Next": "Report Success"
 }
```

```
"Synchronously Run a Transform Job": {
    "Type": "Task",
    "Resource":
"arn:aws:states:::sagemaker.createTransformJob.sync",
    "Parameters":
           {
            "TransformJobName.$": "$.transform",
            "ModelName.$": "$.model",
            "MaxConcurrentTransforms": 8,
            ...
           },
    "Catch": [
      {"ErrorEquals": ["States.TaskFailed"],
       "ResultPath": "$.cause",
       "Next" : ”Notify on Error"
       } ],
    "ResultPath": "$.jobInfo",
    "Next": "Report Success"
 }
```

Define workflows in JSON

```
{
"StartAt": "Download",
"States": {
     "Download": {
          "Type": "Task",
          "Resource": "arn:aws:lambda:REGION:ACCT:function:download_data”,
          "Next": "Train"
          },
     "Train": {
          "Type": "Task",
          "Resource": "arn:aws:states:::sagemaker:createTrainingJob.sync",
          "ResultPath": "$.training_job",
          "Parameters": {
               "AlgorithmSpecification": {
               "TrainingImage": "811284229777.dkr.ecr.us-east-1.amazonaws.com/
     image-classification:latest",
               "TrainingInputMode": "File"
          }…
```

Amazon SageMaker

AWS Step Functions

Spend more time on the code that differentiates your business and deliver faster.

Getting started

https://aws.amazon.com/free

https://ml.aws

https://aws.amazon.com/sagemaker

https://aws.amazon.com/step-functions

Thank you!

Julien Simon, Global Evangelist, AI & Machine Learning
@julsimon
https://medium.com/@julsimon