Track 4|Session 3

利用 AWS Step Functions 建構穩健的業務處理流程

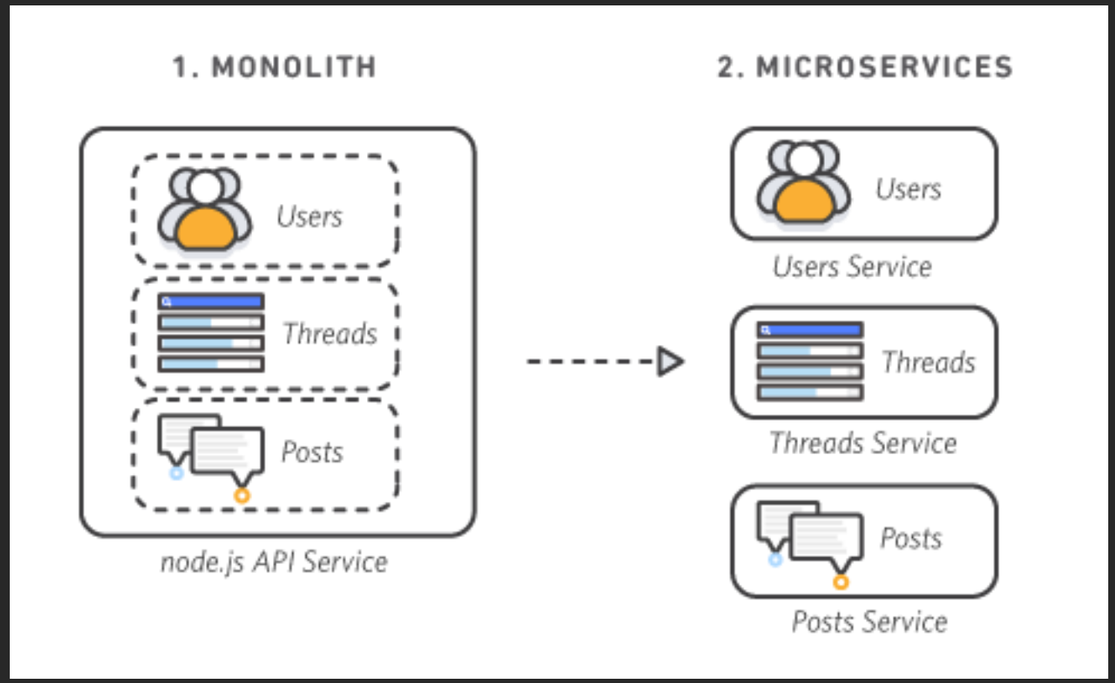
Scott Li
Cloud Support Engineer
Amazon Web Services

aws SUMMIT ONLINE

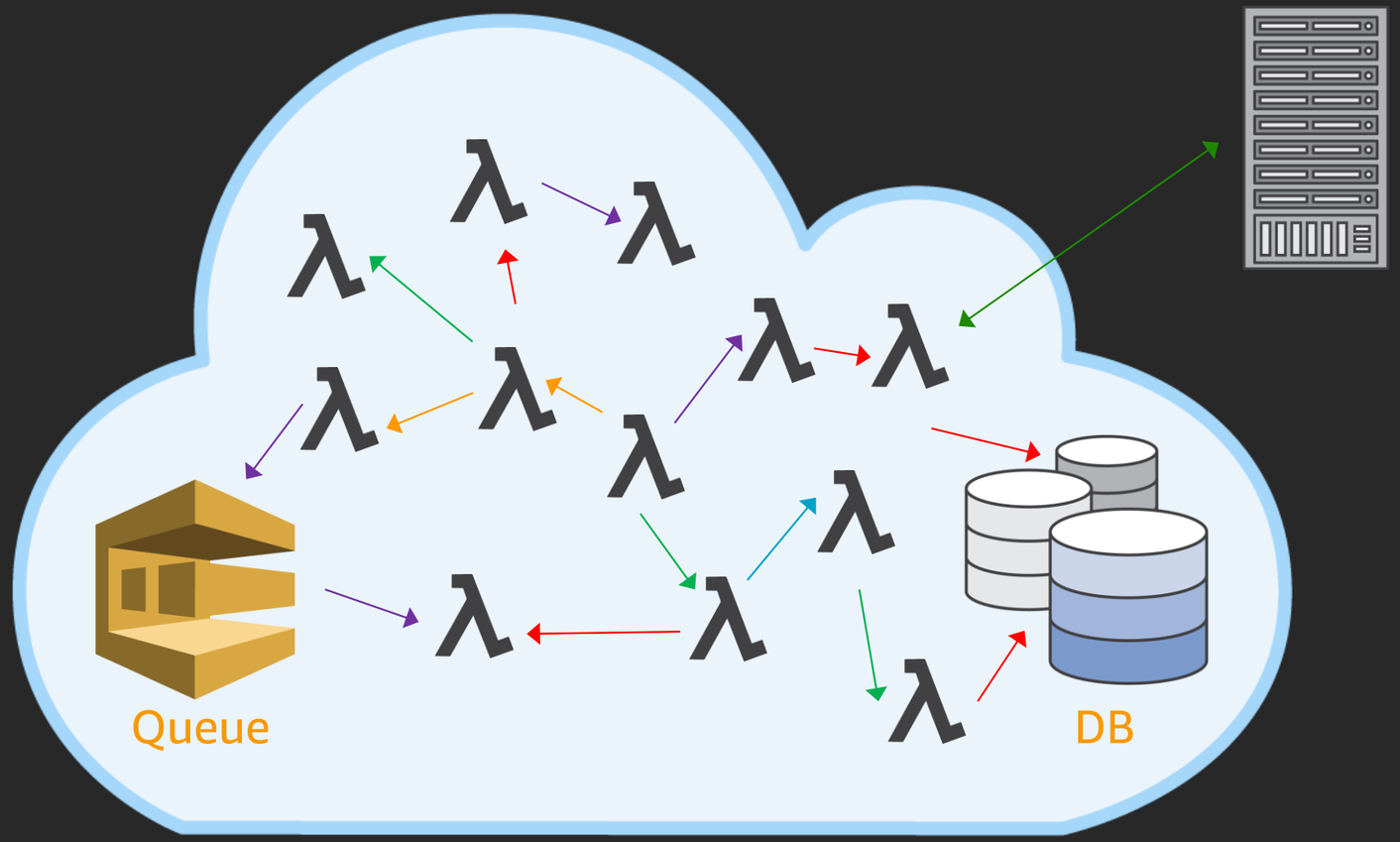
Agenda

- Modern Application
- AWS Step Functions Features
- Workflow Examples
- Customer Use Cases
- Summary & Best Practices

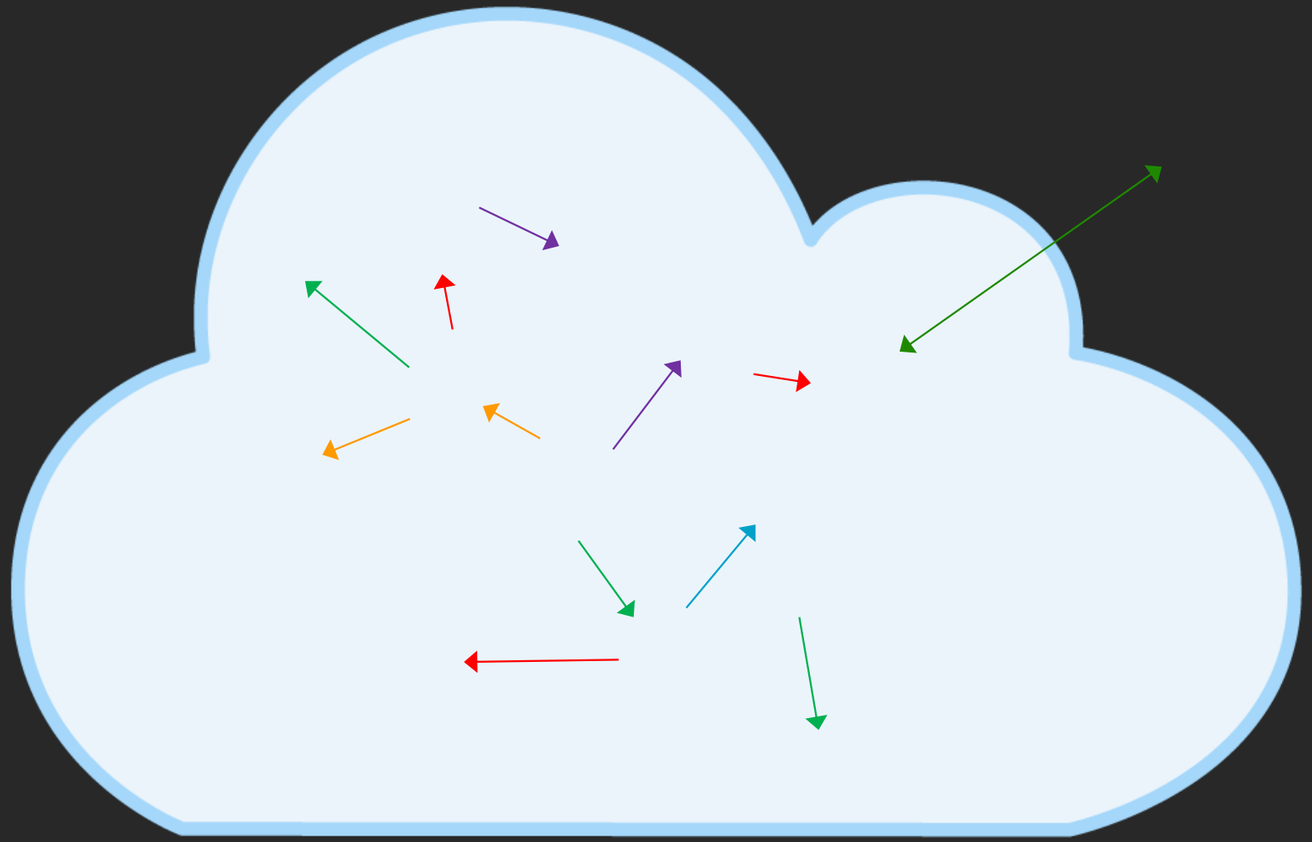
Modern Applications



Business Logic Orchestration

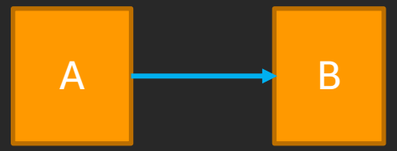


Workflow Coordination

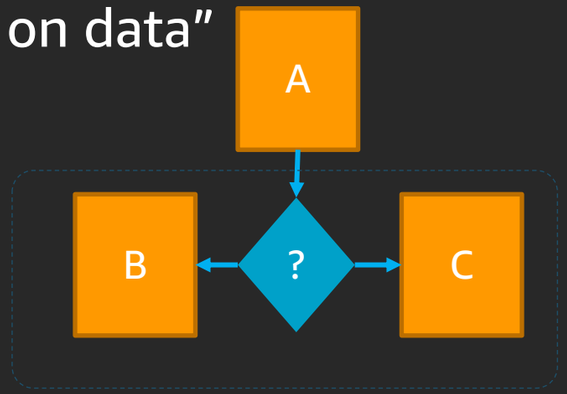


Is this you?

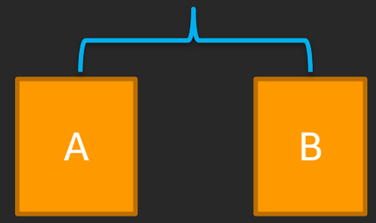
“I want to sequence tasks”



“I want to select tasks based on data”



“I want to run tasks in parallel”



“I want to retry failed tasks”

“I want try/catch/finally”

Coordination must-haves

- Easy to build & operate
- Scales out
- Doesn’t lose state
- Deals with errors/exceptions
- Auditable

AWS Step Functions Features

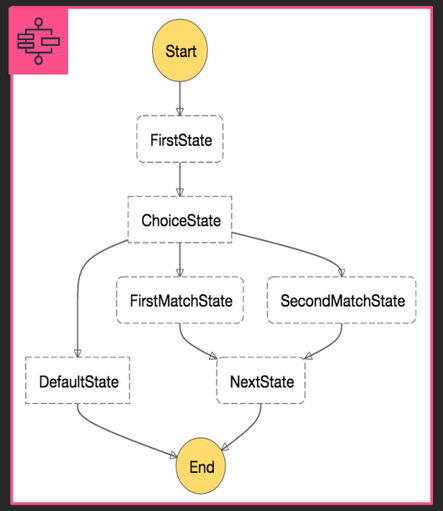
AWS Step Functions Features

Define in JSON

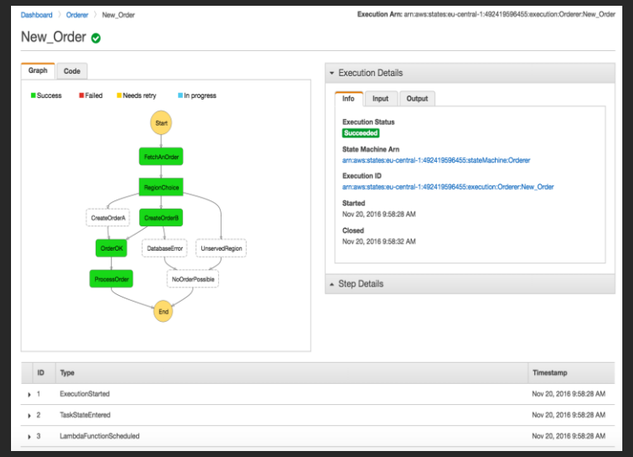
Code

```
{
  "Comment": "An AWL example using a choice state.",
  "StartAt": "FirstState",
  "States": {
    "FirstState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:FUNCTION_NAME",
      "Next": "ChoiceState"
    },
    "ChoiceState": {
      "Type" : "Choice",
      "Choices": [
        {
```

Visualize in the Console



Monitor Executions



Workflow Type

Flexibility to select the right workflow type for your needs, or integrate them together where needed

Standard Workflows

- Long-running, durable and auditable workflows
- 2000 per second (stared rate)
- 4000 per second (state transition)
- Log events on console & CloudWatch Log
- Max execution time of 1 year
- $0.025 per 1,000 state transitions

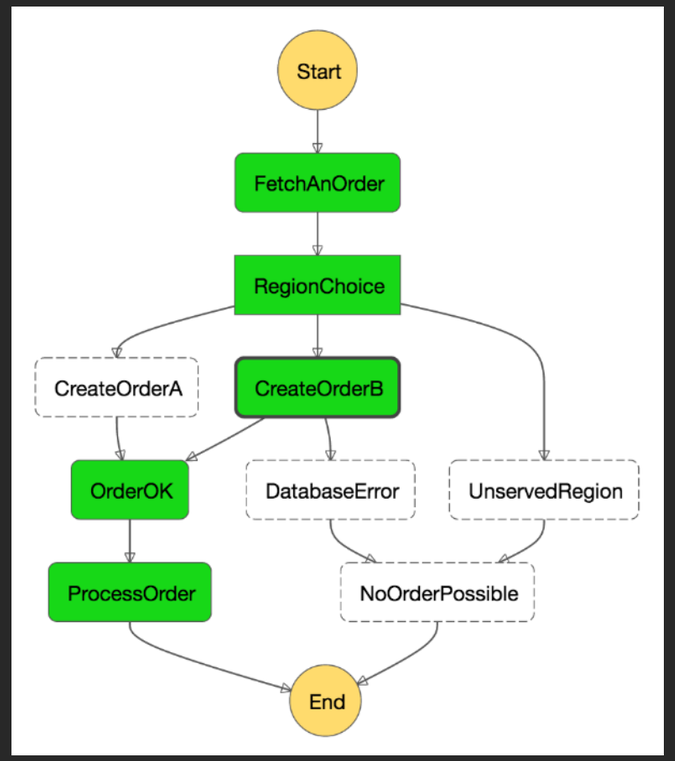
Express Workflows

- Simplified capabilities, geared for speed and scale
- Supporting event rates up to 100,000 per second
- No logging events on console
- Max execution time of 5 mins
- Pay per use at $1.00 per 1M requests and $0.000001 per request

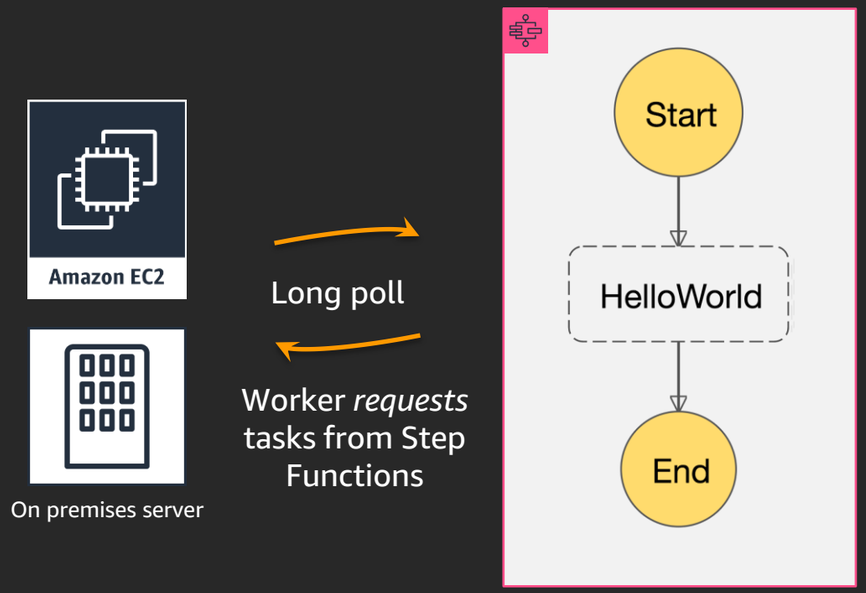
Orchestrating Your Business Logic with Eight States

The Amazon States Language is a JSON-based, structured language used to define your state machine

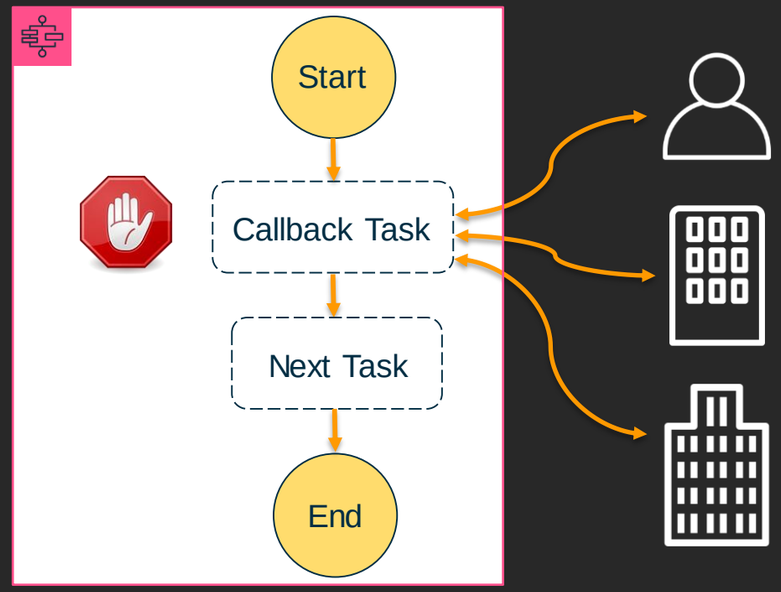
- Task
  - Lambda Function
  - Activity
  - AWS Services
- Choice
- Parallel
- Map
- Pass
- Wait
- Succeed
- Fail



Tasks: Activity
Hybrid workflows with servers and instances



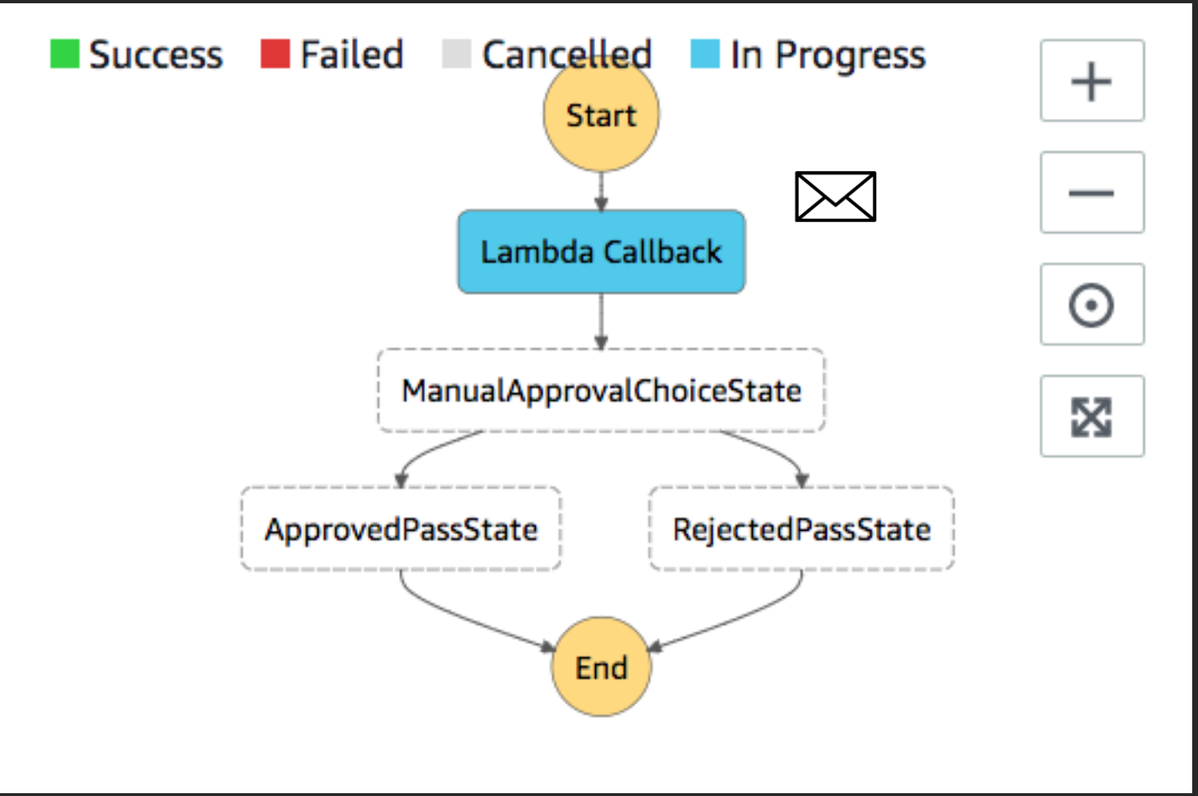
Tasks: Callback



Call an external resource. Pause workflow for a callback event from the resource. Wait for as long as you need – minutes, days, weeks, or months:

- Human activity
- Third party API
- Legacy application

Callback tasks : Human Activities



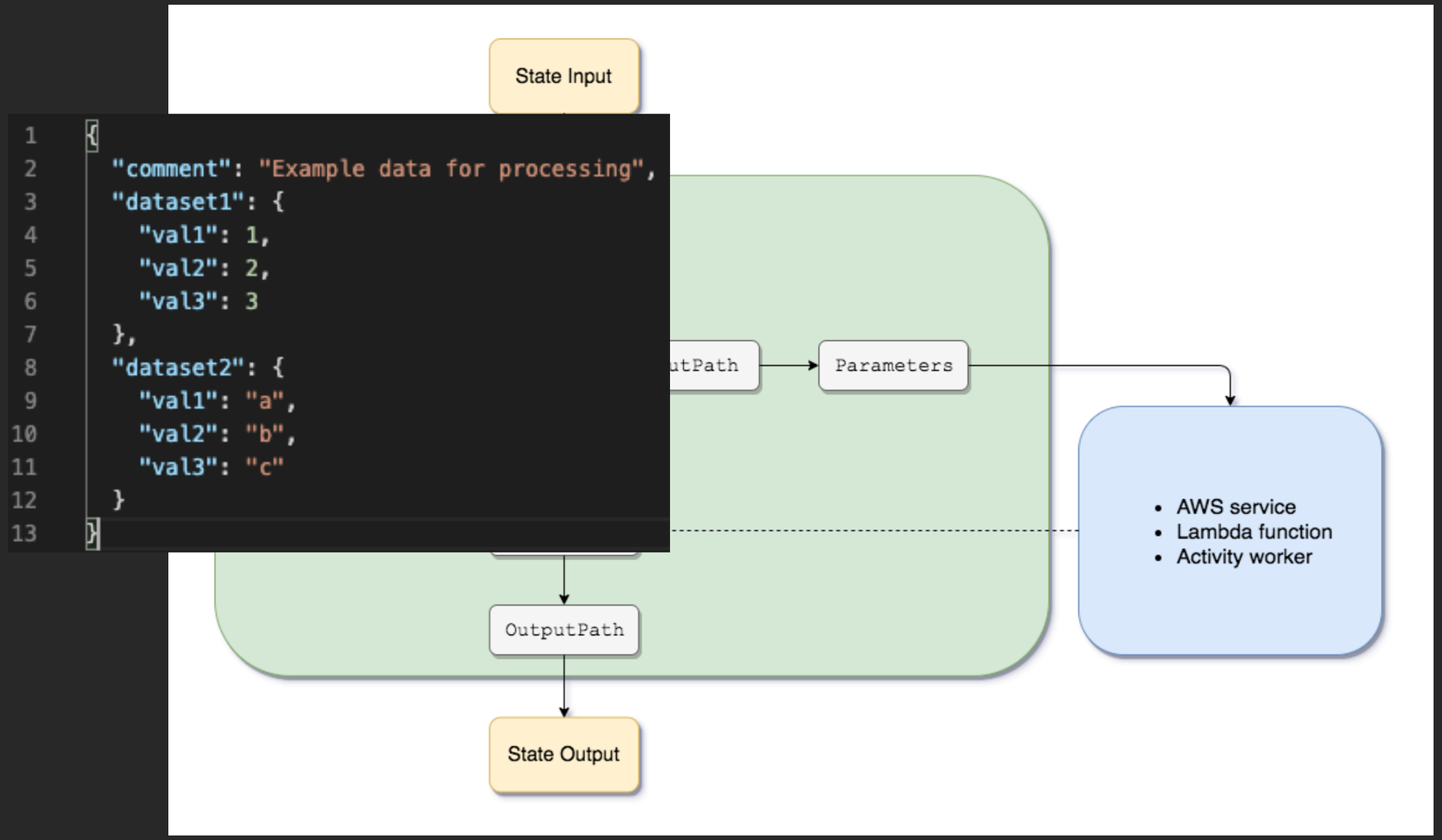
Call an external resource. Pause workflow for a callback event from the resource. Wait for as long as you need – minutes, days, weeks, or months:

- Human activity
- Third party API
- Legacy application

AWS Step Functions: Input and Output Processing

- InputPath
- Parameters
- ResultPath
- OutputPath

- ItemPath



AWS Step Functions: Input and Output Processing

InputPath

```
{
  "comment": "Example for InputPath.",
  "dataset1": {
    "val1": 1,
    "val2": 2,
    "val3": 3
  },
  "dataset2": {
    "val1": "a",
    "val2": "b",
    "val3": "c"
  }
}
```

Syntax

```
"InputPath" : "$.dataset2"
```

Output

```
{
    "val1": "a",
    "val2": "b",
    "val3": "c"
}
```

AWS Step Functions: Input and Output Processing

Parameters

```
{
    "Comments": "Example",
    "dataset1": {
        "val1": 1,
        "val2": 2,
        "val3": 3
    },
    "dataset2": {
        "val1": "a",
        "val2": "b",
        "val3": "c"
    }
}
```

Syntax

```
"Parameters": {
    "Comments":"selecting the value that I care about",
    "Entry01.$":"$.dataset1.val1",
    "Entry02.$":"$.dataset2.val3",
    "StaticValue": "Newly added"
}
```

Output

```
{
    "Comments":"selecting the value that I care about",
    "Entry01":1,
    "Entry02":"c",
    "StaticValue": "Newly added"
}
```

AWS Step Functions: Input and Output Processing

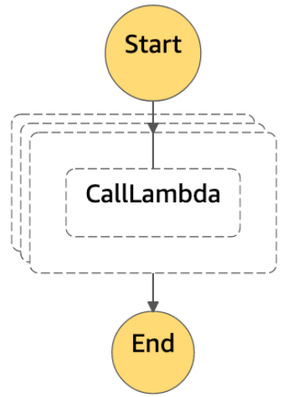
ItemPath

```
{
    "ThingsPiratesSay": [
      {
        "say": "Avast!"
      },
      {
        "say": "Yar!"
      },
      {
        "say": "Walk the Plank!"
      }
    ],
    "ThingsGiantsSay": [
      {
        "say": "Fee!"
      },
      {
        "say": "Fi!"
      },
      {
        "say": "Fo!"
      },
      {
        "say": "Fum!"
      }
    ]
  }
```

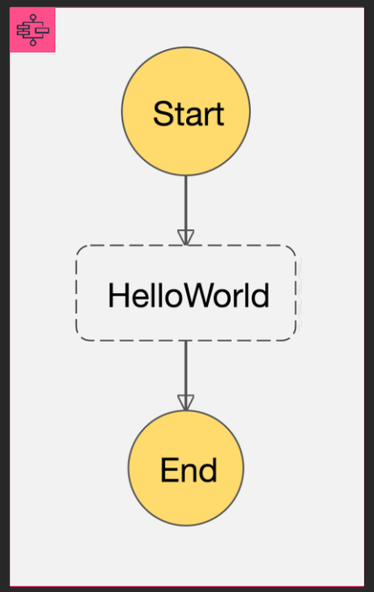
Definition

Export ▼ Layout ▼

```
{
  "StartAt": "ExampleMapState",
  "States": {
    "ExampleMapState": {
      "Type": "Map",
      "ItemsPath": "$.ThingsPiratesSay",
      "Iterator": {
         "StartAt": "CallLambda",
         "States": {
           "CallLambda": {
             "Type": "Task",
             "Resource": "arn:aws:lambda:ap-northeast-1:399415179832:function:lab-summit-2020",
             "End": true
           }
         }
      },
      "End": true
    }
  }
}
```



Service Integrations



Request

Step Functions *invokes* the task and pauses until the task completes

AWS Lambda

Amazon ECS

AWS Fargate

AWS Batch

Amazon SQS

Amazon SNS

Amazon DynamoDB

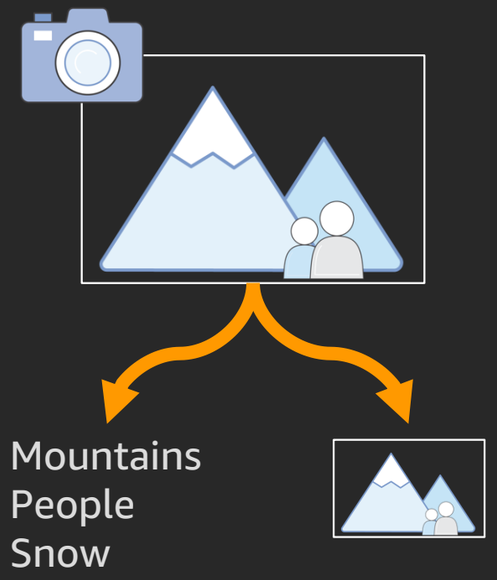
Amazon SageMaker

AWS Glue

Amazon EMR

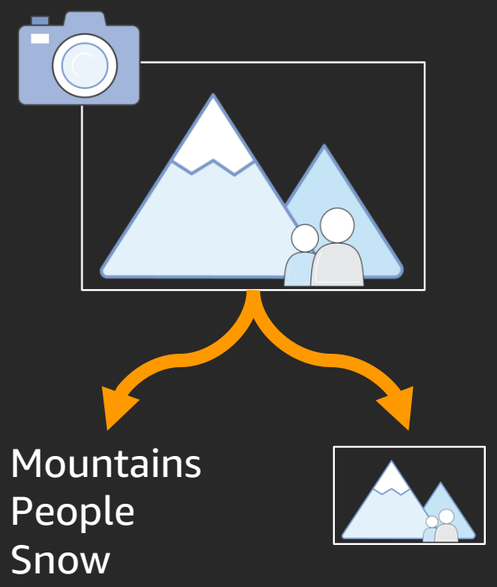
Workflow Examples

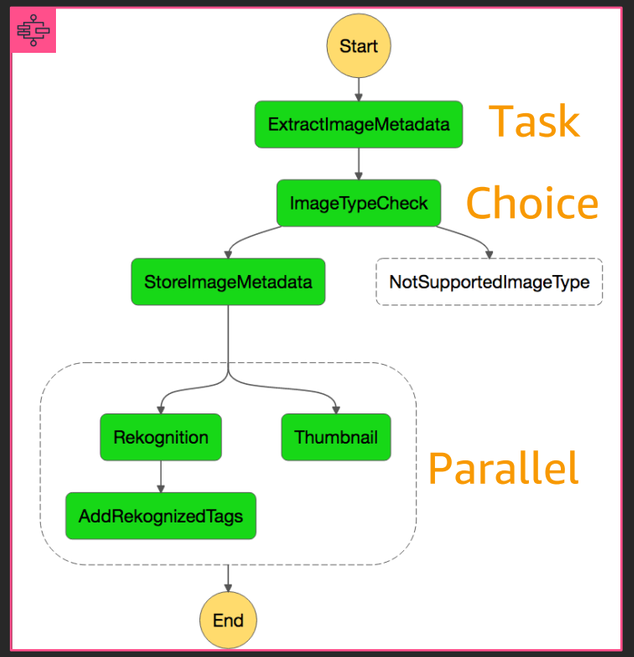
An Image Processing Workflow



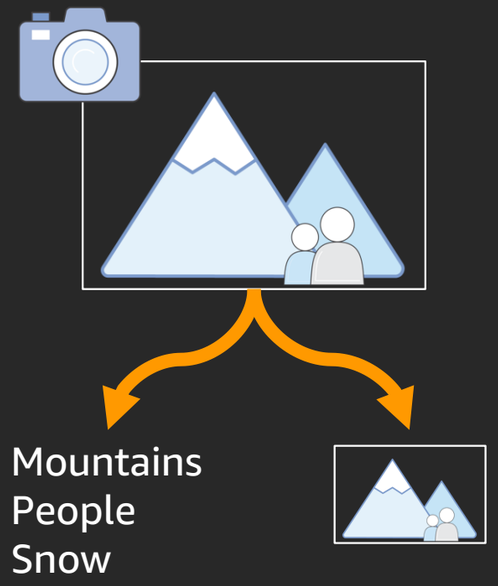
- Make a thumbnail
- Identify features
- Store image metadata

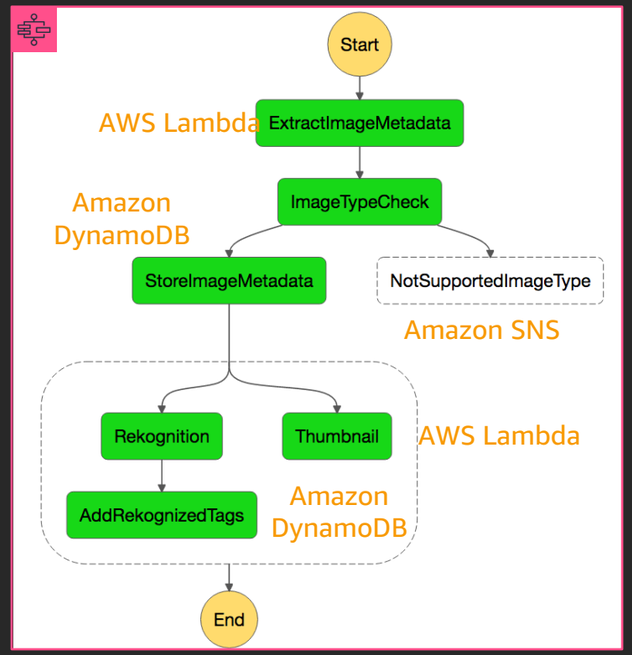
Build Workflows



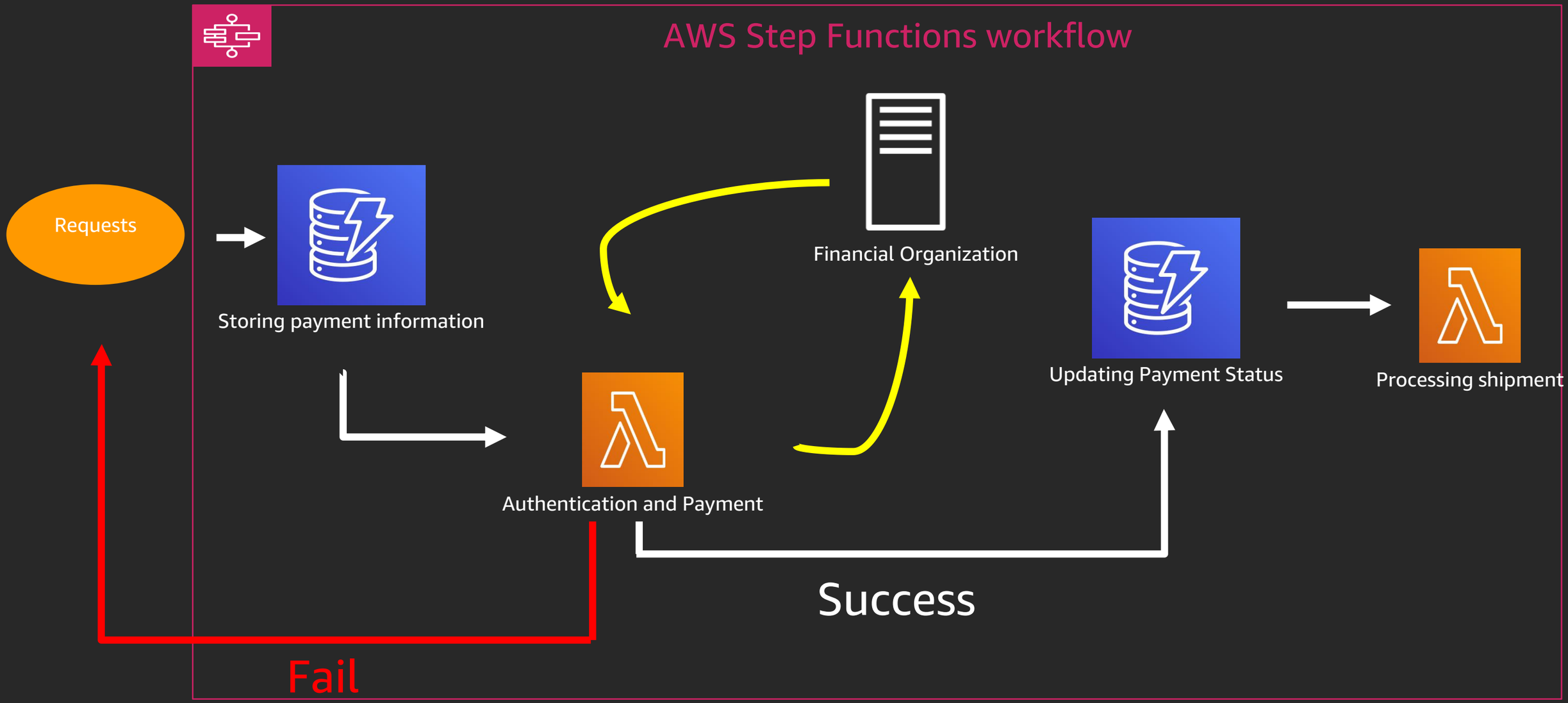


Build Workflows Using Service Integrations

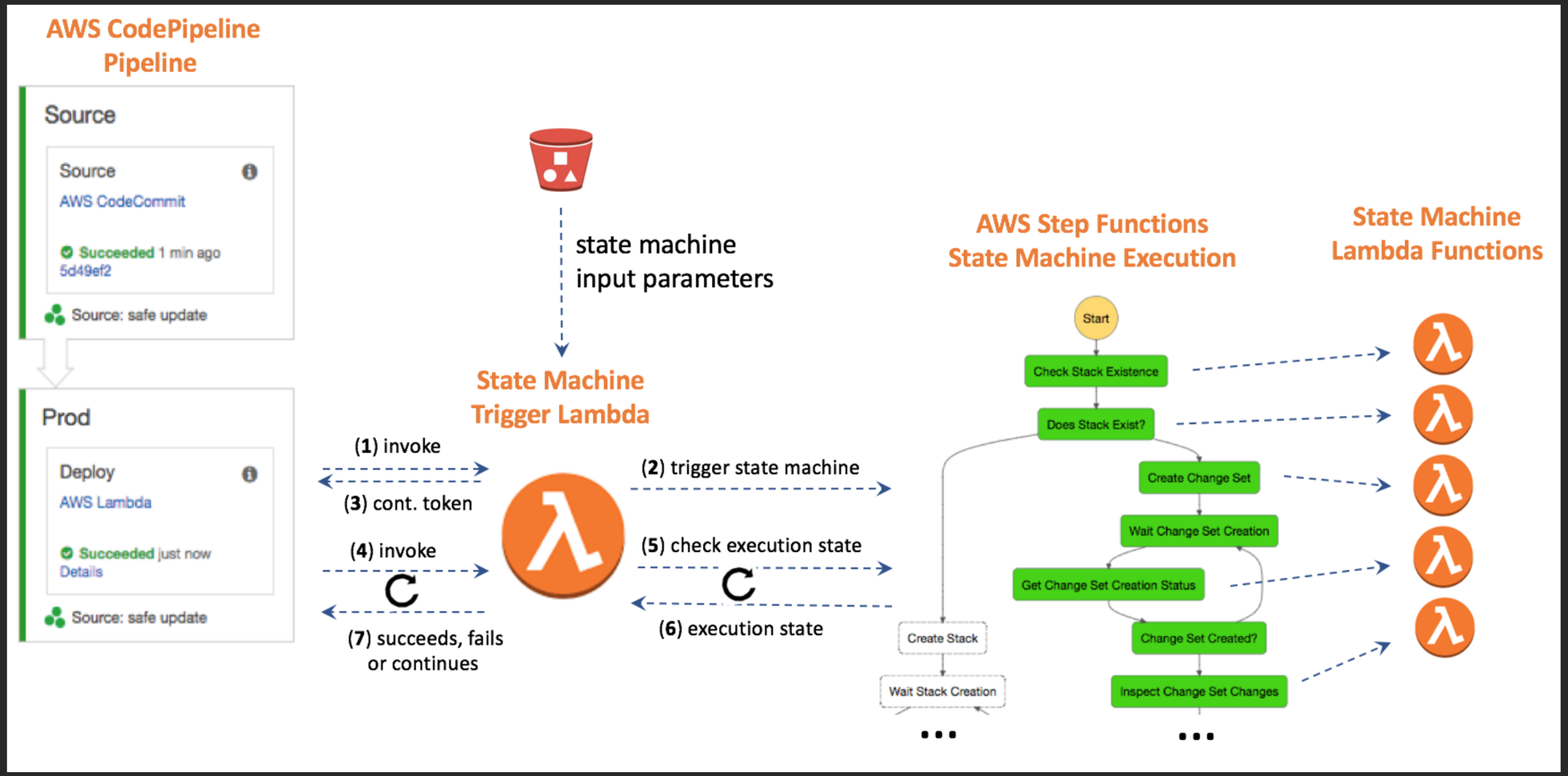




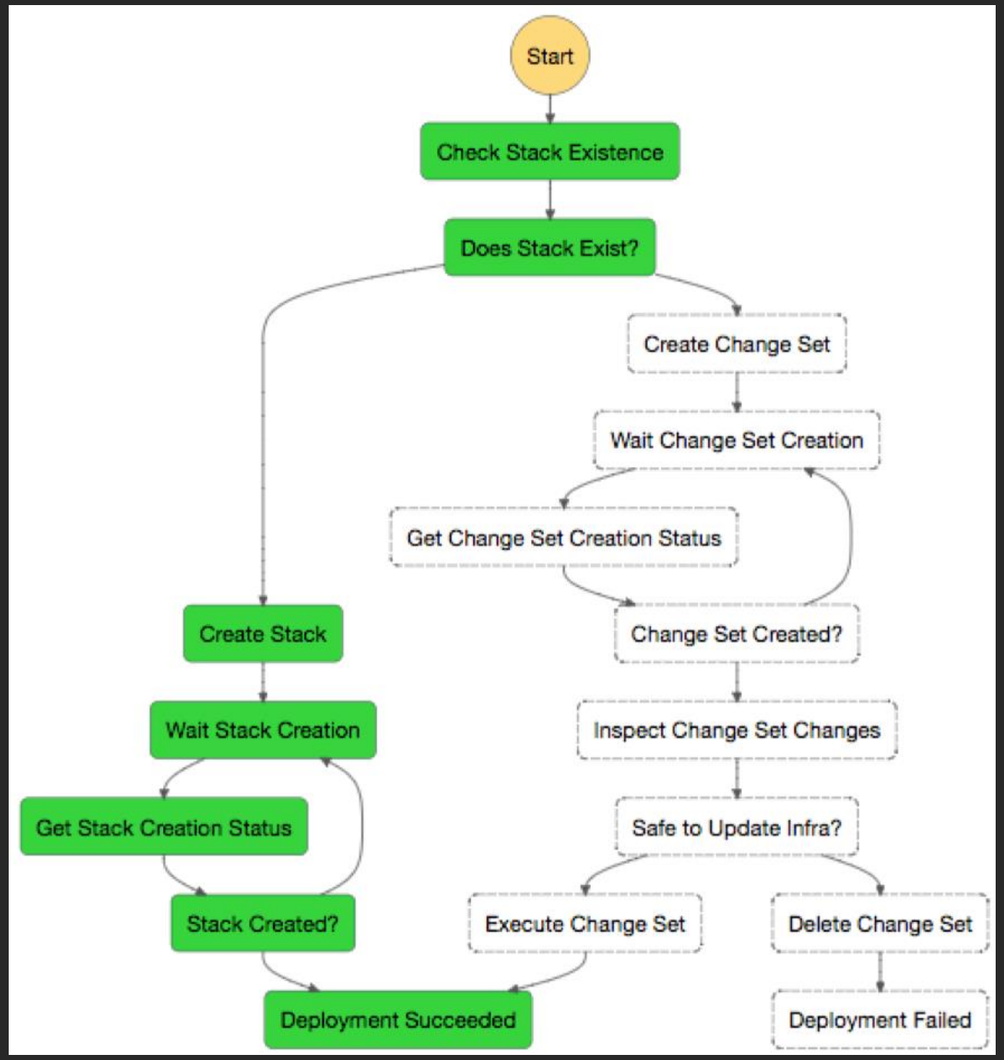
Payment and Order Processing Flow



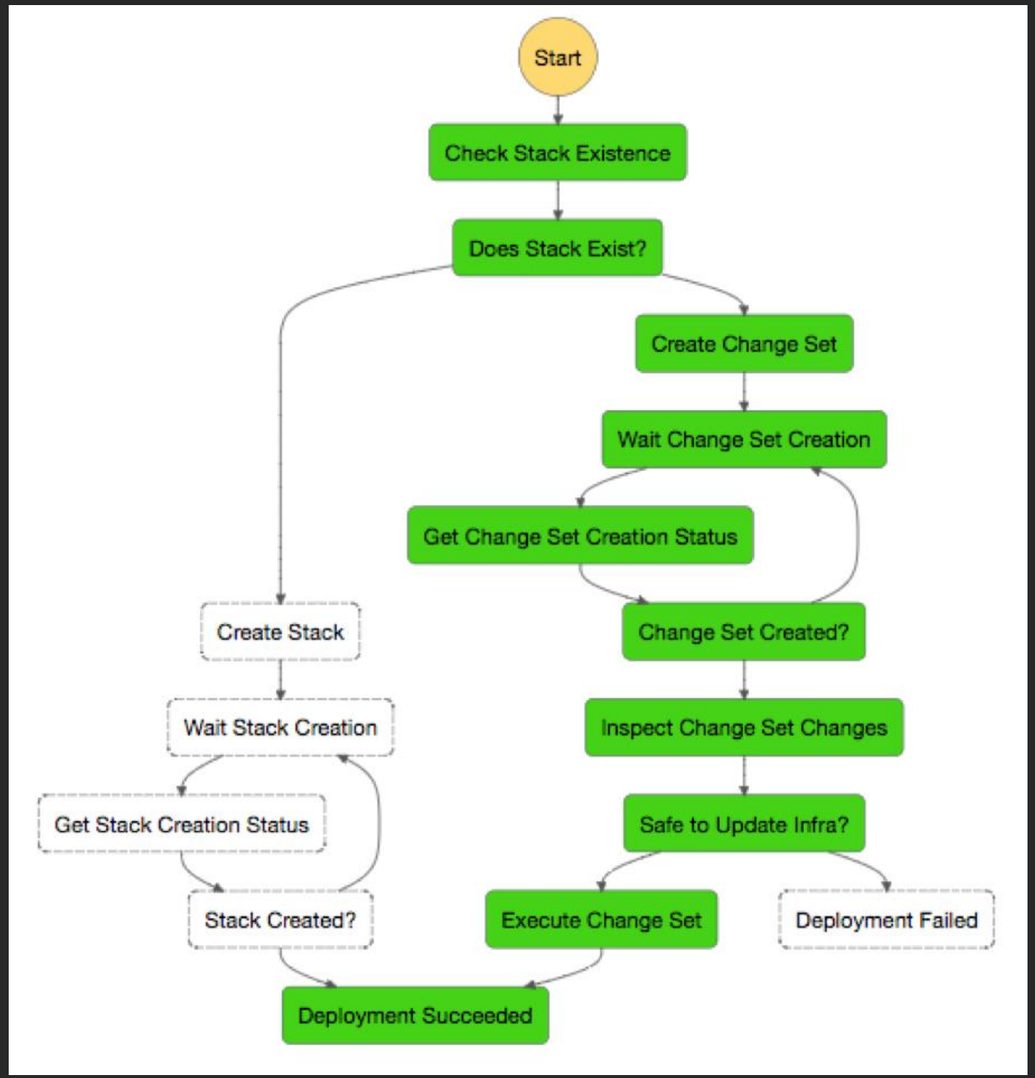
Workflow-Driven AWS CodePipeline Actions



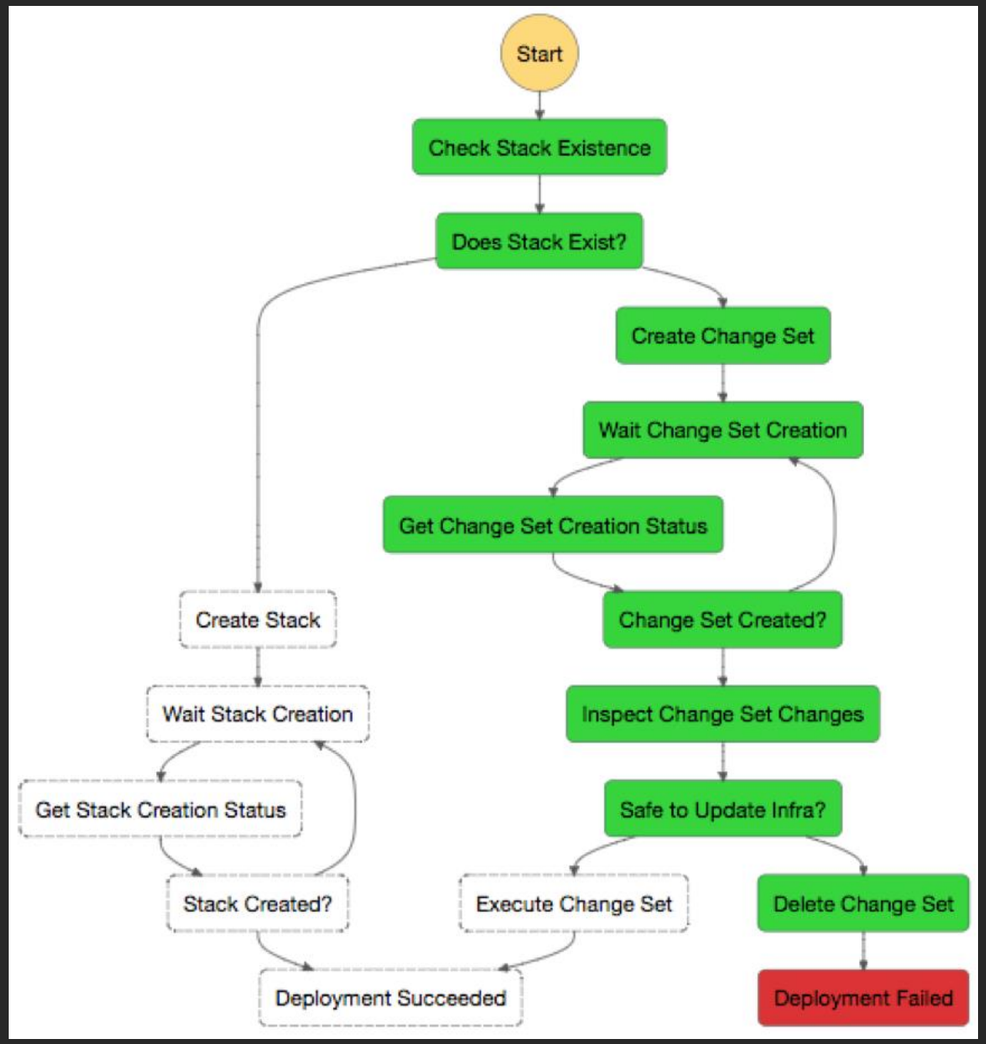
Workflow-Driven AWS CodePipeline Actions



Workflow-Driven AWS CodePipeline Actions



Workflow-Driven AWS CodePipeline Actions



Customer Use Cases

Use Cases for AWS Step Functions

Process data

THOMSON REUTERS

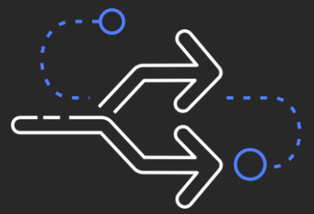
Automate tasks

coinbase

Modernize monoliths

yelp

Orchestrate applications



Frame.io

Thomson Reuters

Serverless split video transcoding

Industry: Media & Entertainment
Solution: Big Data / Machine Learning
Architecture: Serverless

Challenge

- Transcode ~350 news video clips per day into 14 formats each – as quickly as possible

Results

- Able to process video segments in parallel
- Reduced processing time from ~20 min to ~2 minutes
- The bigger the source video, the more segments, the bigger the savings

coinbase

Buy and sell digital currency

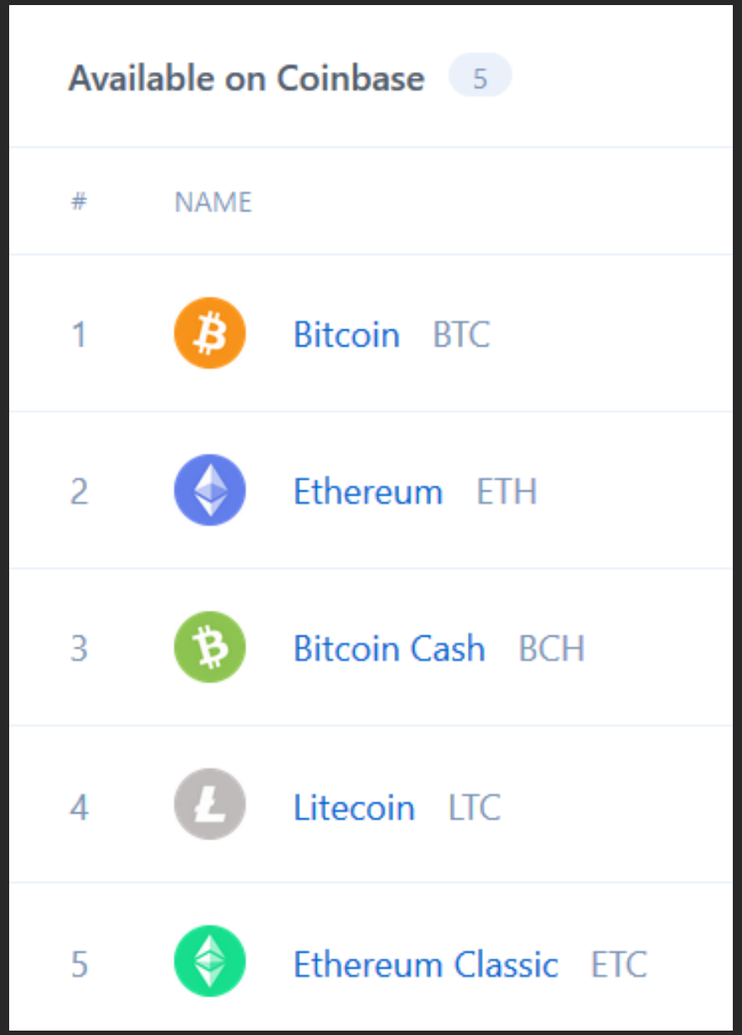
coinbase

Challenge

- Secure, reliable, immutable deployments with visibility by their engineers
- Scale to multiple AWS accounts easily and quickly

Results

- Reduced new account deployment from days to seconds
- Increased the time a deployment can reliably run from 2 hours to 48 hours
- Mission critical services deployment rate rose 7%



Yelp

Transformed a decade-old monolith to microservices

Industry: Internet
Solution: Application modernization
Architecture: Serverless

Challenge

- Make a very old, very critical monolithic billing application easier to modify & maintain
- Improve account billing process
- High refactoring risk

Results

- Able to refactor gradually and safely
- Able to process multiple accounts at once (concurrency)
- Stable and improved performance (parallelism)
- Enhanced observability
- Simple evolution to Lambda

Frame.io

Custom solution to optimize real-time transcoding

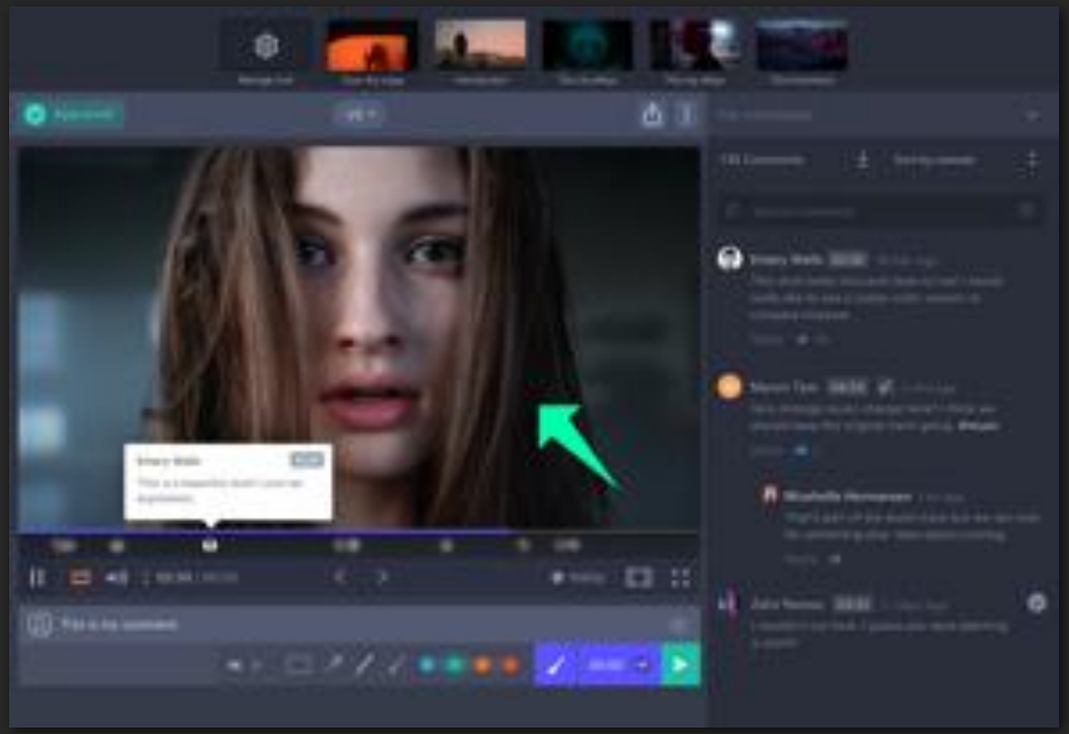
Industry: Media & Entertainment
Solution: Digital Supply Chain
Architecture: Microservices

Challenge

- Workflow management platform for video teams has to be able to process high volumes of media, transcode to different formats, cost-effectively

Results

- Improved performance and lower costs
- Code is easily managed and debugged
- Increased code releases 20x using CI/CD
- Better performance, expect to scale to running 1,000 state machines a minute



Summary & Best Practice

Business Processes on AWS Step Functions

Build and update apps quickly

AWS Step Functions lets you build visual workflows that enable fast translation of business requirements into technical requirements. You can build applications in a matter of minutes, and when needs change, you can swap or reorganize components without customizing any code.

Write less code

AWS Step Functions manages the logic of your application for you, and implements basic primitives such as branching, parallel execution, and timeouts. This removes extra code that may be repeated in your microservices and functions.

Improve resiliency

AWS Step Functions manages state, checkpoints and restarts for you to make sure that your application executes in order and as expected. Built-in try/catch, retry and rollback capabilities deal with errors and exceptions automatically.

Best Practice

- Use Timeouts to Avoid Stuck Executions
- Use ARNs Instead of Passing Large Payloads
- Avoid Reaching the History Quota
- Handle Lambda Service Exceptions
- Avoid Latency When Polling for Activity Tasks
- Choosing Standard or Express Workflows

Thank you!

aws SUMMIT ONLINE