



CONF42: CLOUD NATIVE 2023

One Step Ahead

Modern Application Architectures with AWS Step Functions

Oskar Neumann

Solutions Architect

A person is standing on a stage in front of a large screen. The screen displays a blue digital background with a globe and data lines. The text on the screen reads: "SO WHAT DOES THE FUTURE LOOK LIKE?" followed by "ALL THE CODE YOU EVER WRITE IS BUSINESS LOGIC" in a larger font.

SO WHAT DOES THE FUTURE LOOK LIKE?
ALL THE CODE YOU EVER WRITE IS BUSINESS LOGIC

The AWS Serverless ecosystem is much more than compute

COMPUTE



AWS
Lambda



AWS
Fargate

DATA STORES



Amazon
S3



Amazon Aurora
Serverless

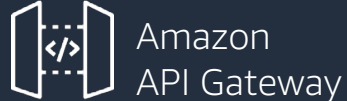


Amazon
DynamoDB

INTEGRATION



Amazon
EventBridge



Amazon
API Gateway



Amazon
SQS



Amazon
SNS



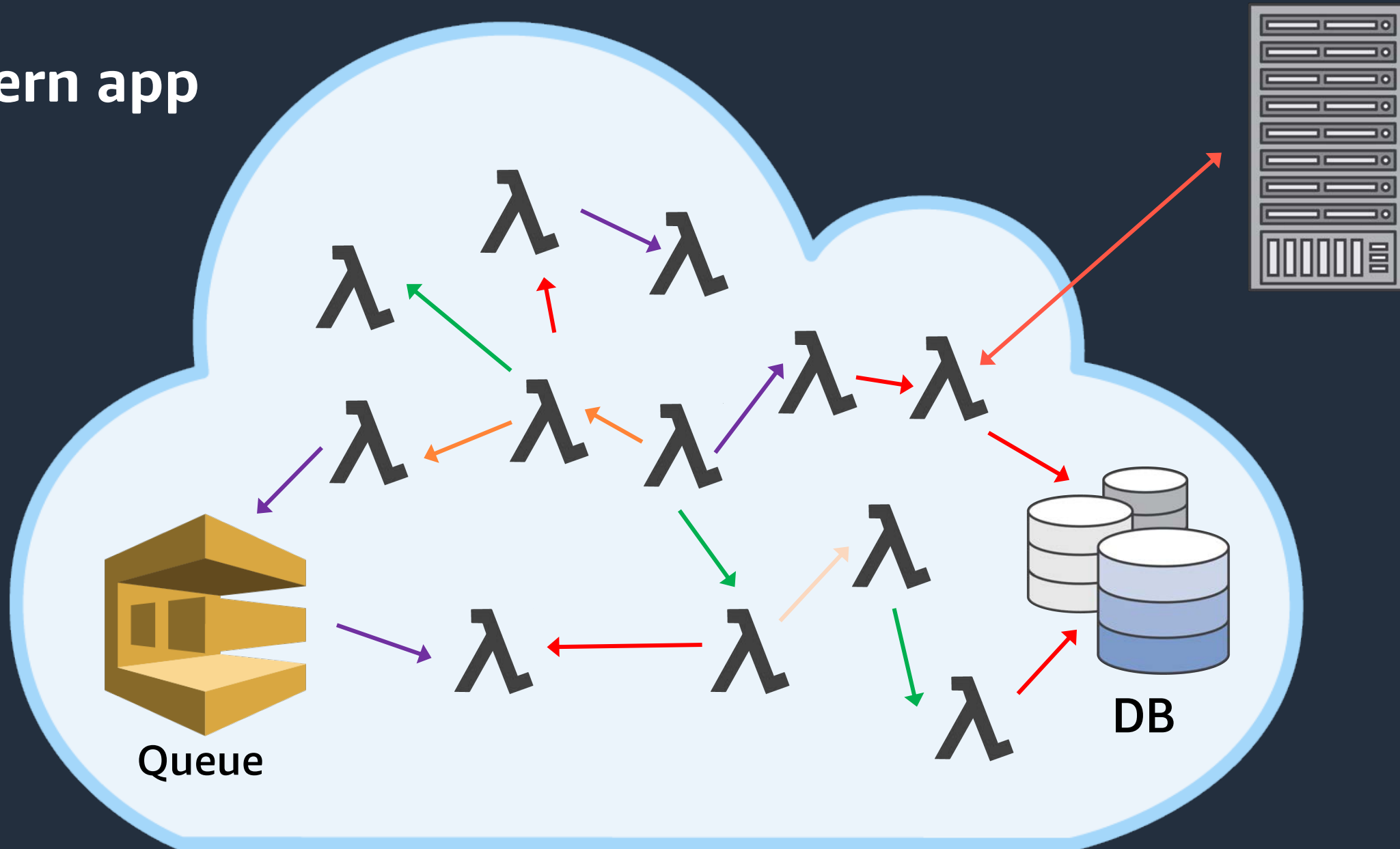
AWS
Step Functions



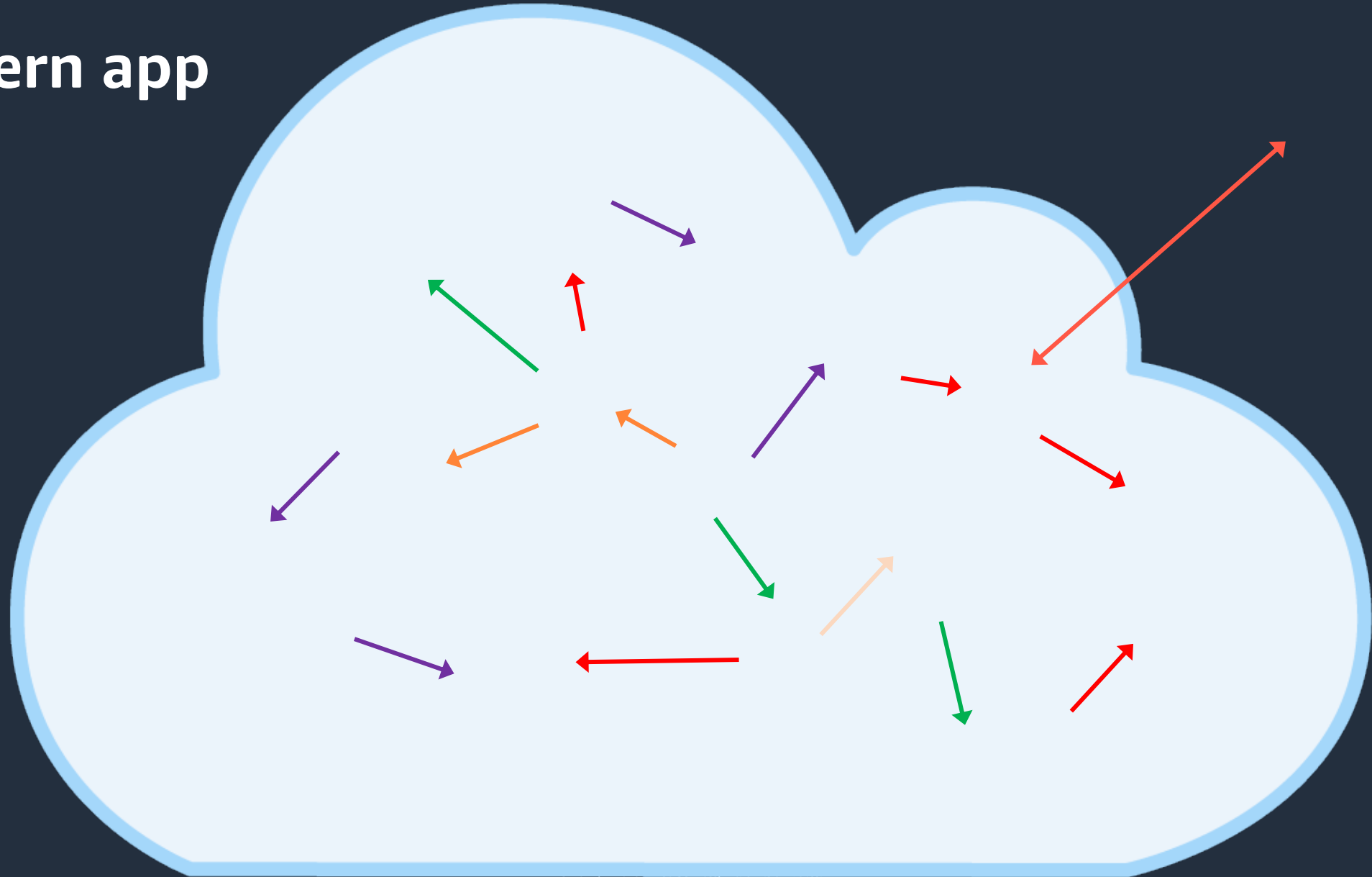
AWS
AppSync

Why orchestration?

Modern app



Modern app



Why orchestration?

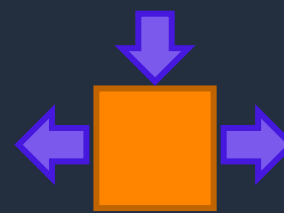
"I want to sequence tasks"



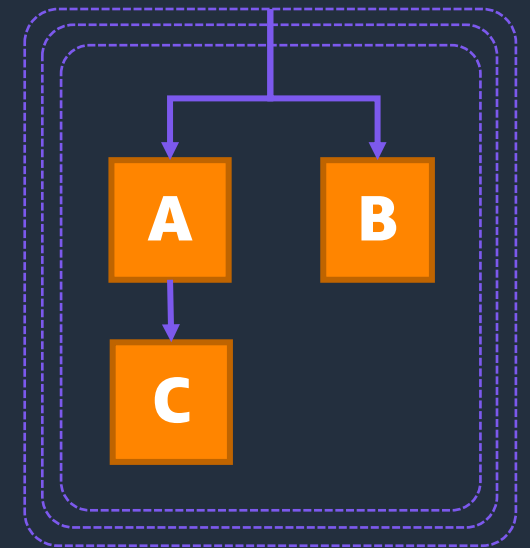
"I want to retry failed tasks"



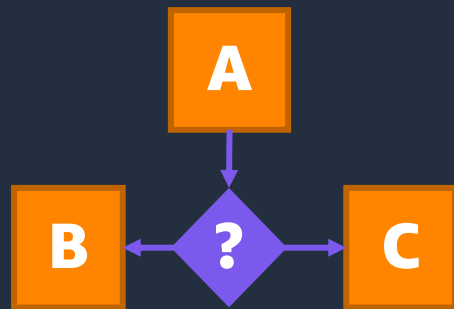
"I want try/catch/finally"



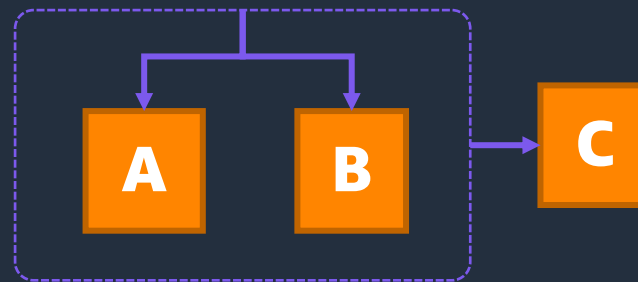
"I want concurrent and iterative tasks"



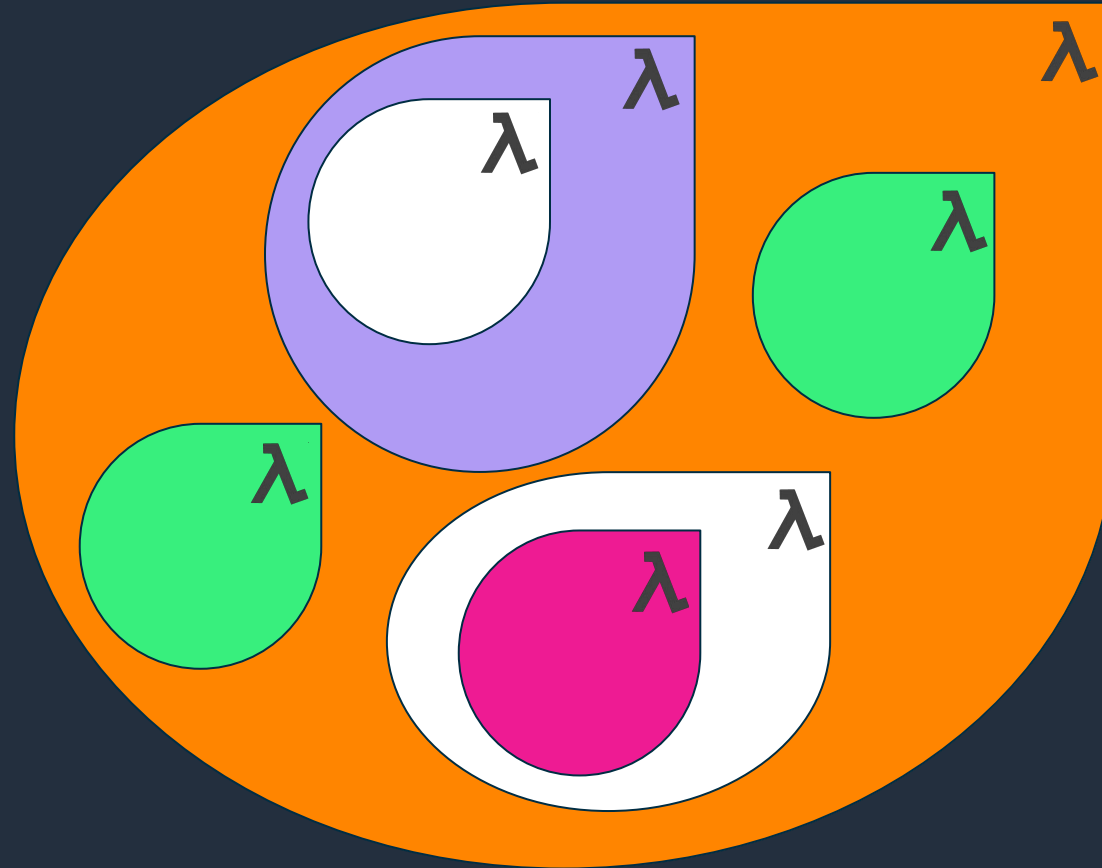
"I want to select tasks based on data"



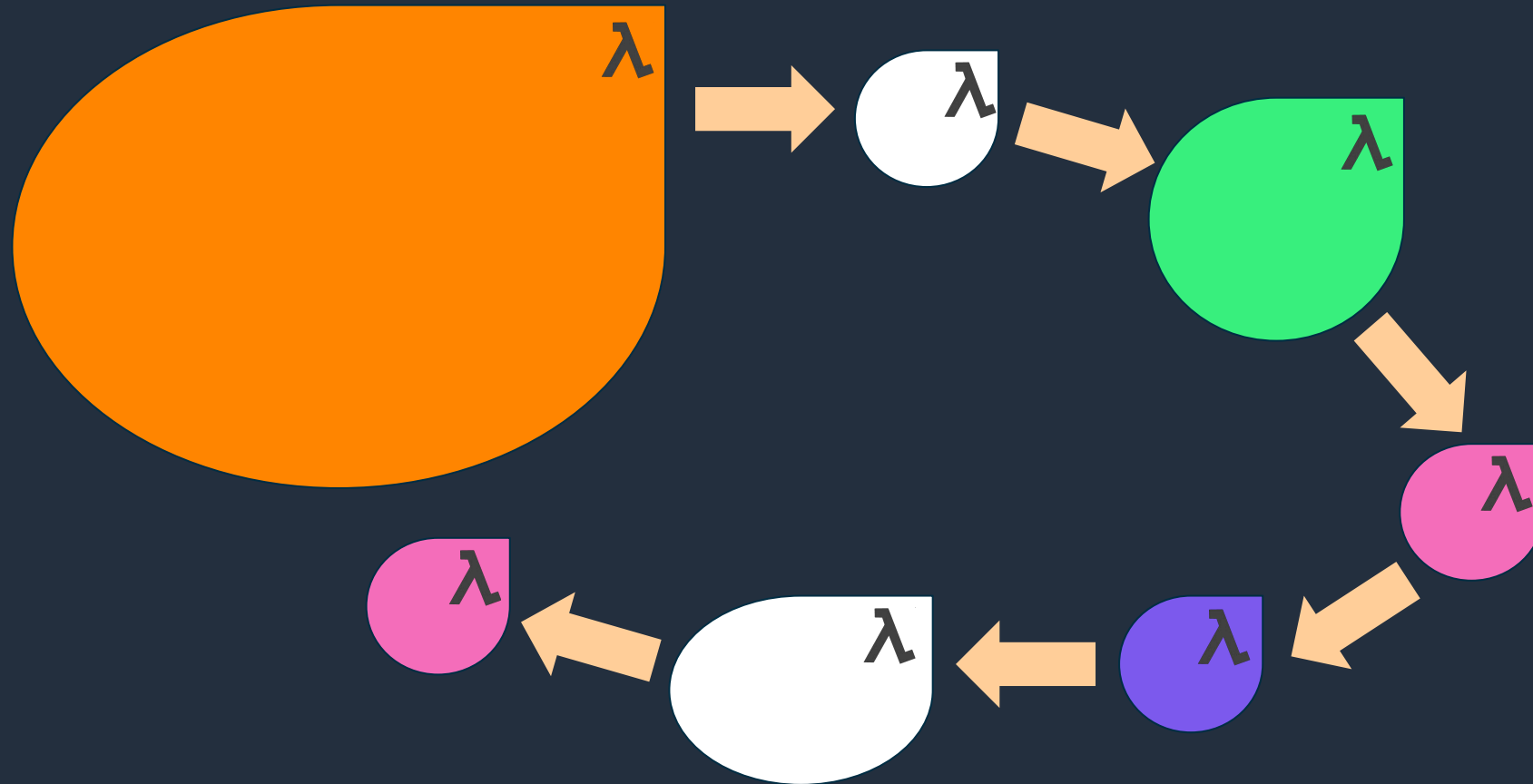
"I want to run tasks in parallel"



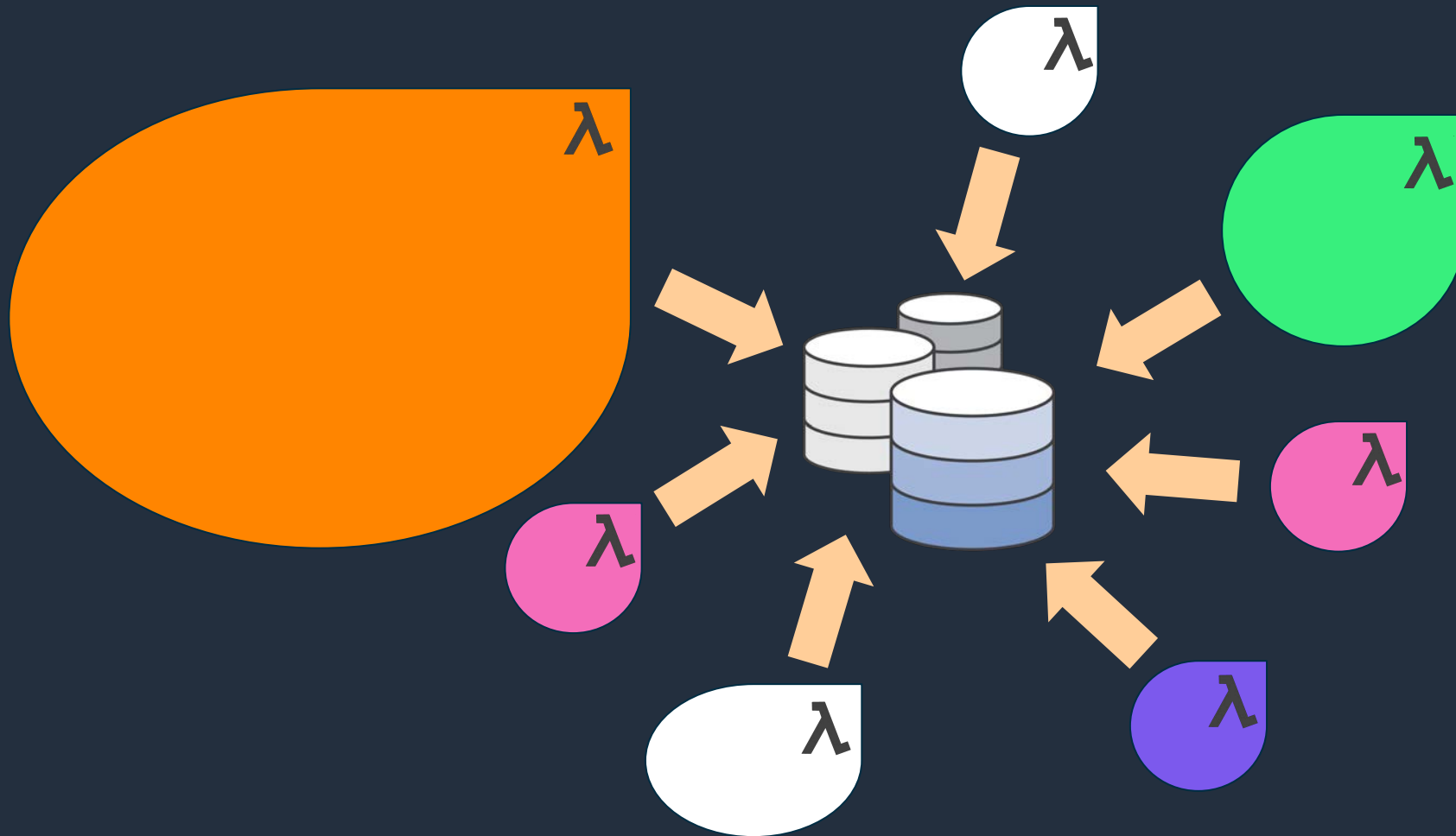
Coordination by method call



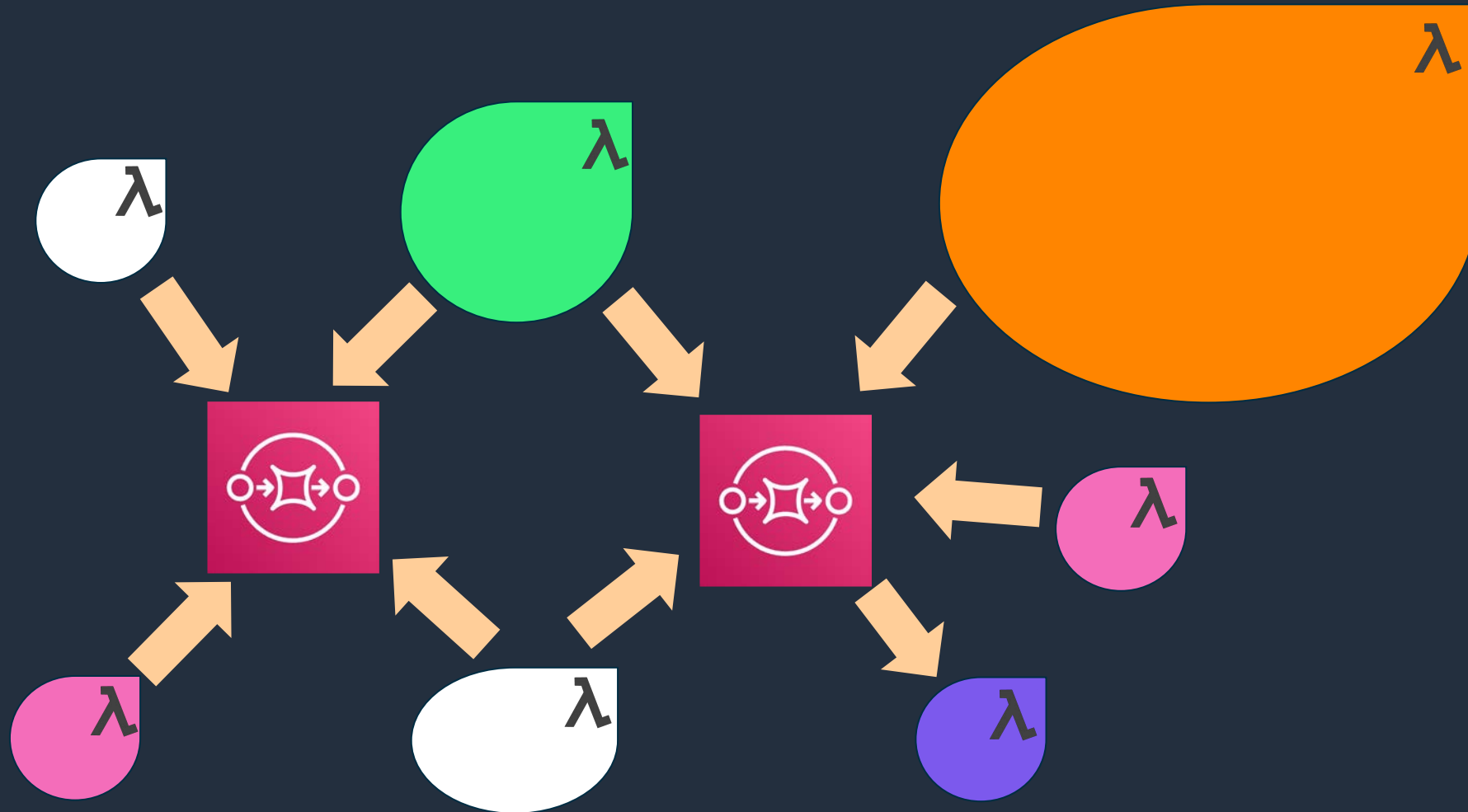
Coordination by function chaining



Coordination by database

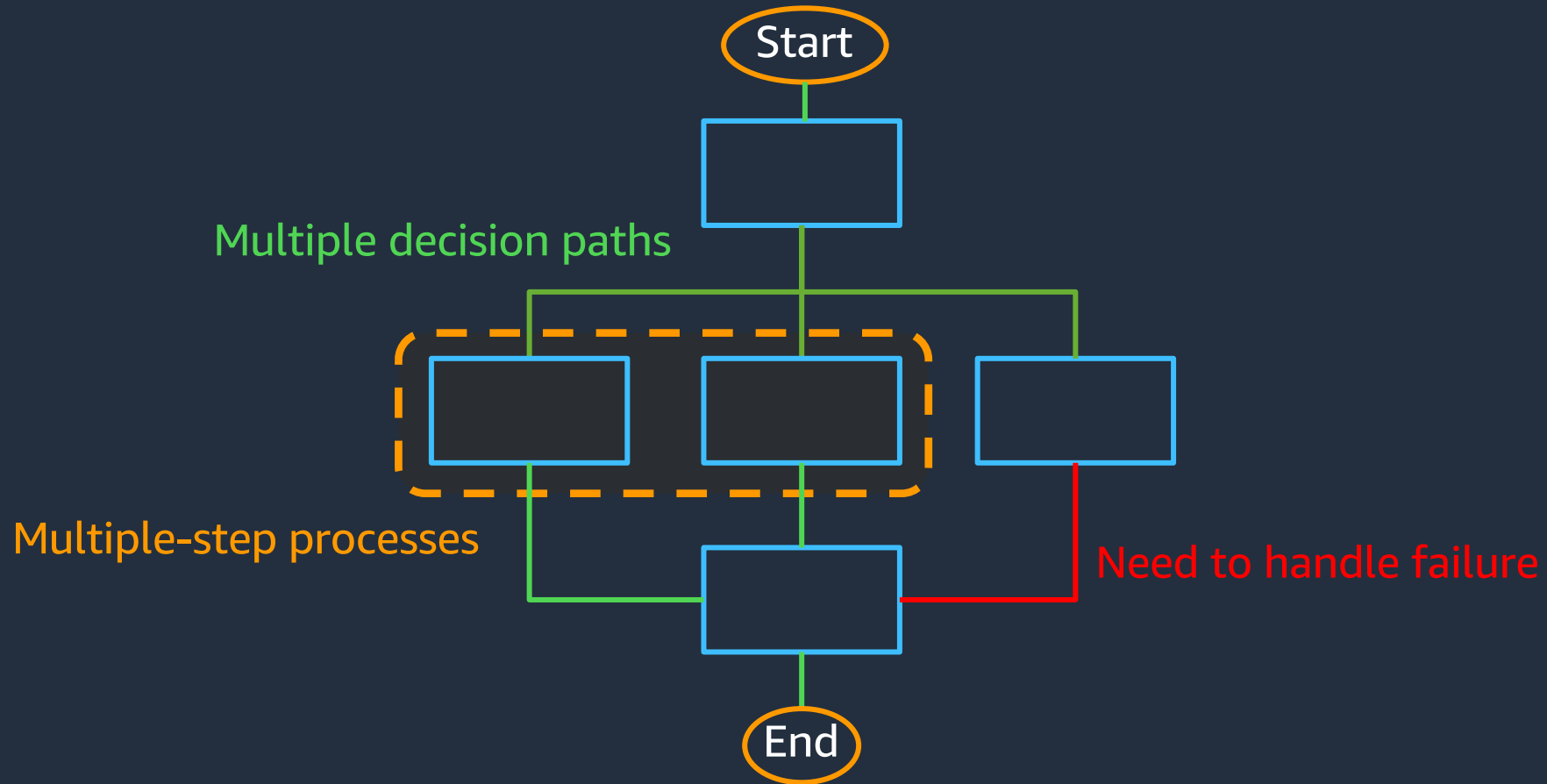


Coordination by queues



How to orchestrate?

Business workflow is rarely sequential start to finish



What is AWS Step Functions?

What is AWS Step Functions?

A **serverless** workflow orchestration service offered by AWS.

A workflow built on AWS Step Functions...

...is built using a **state machine**.

...is composed of **steps** called **states**.

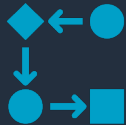
...moves from one state to another via a **state transition**.

...is written using **Amazon States Language** or ASL (think of it as the workflow assembly language).

...can be used to **orchestrate** multiple AWS services.



AWS Step Functions



The **workflows** you build with Step Functions are called **state machines**, and **each step** of your workflow is called a **state**.



When you execute your state machine, each **move** from one state to the next is called a **state transition**.



You can **reuse components**, easily edit the sequence of steps or swap out the code called by task states as your needs change.

The screenshot displays the AWS Step Functions Workflow Designer interface. At the top, there's a navigation bar with 'aws', 'Services', 'Resource Groups', and user account information. Below this is a blue banner with a message: 'Thanks for trying the preview! Please leave feedback so we can improve it.' and a 'Leave feedback' button. The main area is titled 'Step Functions workflow designer (Preview)'. On the left, there's a 'Service APIs' sidebar with a search bar and a list of services including AWS Lambda, Amazon SNS, Amazon ECS, AWS Glue, AWS Step Function, AWS Batch, AWS API Gateway, Amazon Athena, and AWS Codebuild. The central canvas shows a workflow diagram: a yellow 'Start' node connects to a 'Lambda: Invoke Invoke-Function-1' task, which connects to a 'Lambda: Invoke Invoke-Function-2' task, which finally connects to a yellow 'End' node. On the right, the configuration panel for 'Invoke-Function-2' is open, showing tabs for 'Configuration', 'Input', 'Output', and 'Error handling'. The 'Error handling' tab is active, showing options for 'Retry on errors' (with 'Retriever #1' and 'Retriever #2' sections), 'Catch errors', 'Timeout', and 'Heartbeat'. Each section has a '+ Add new' button. At the bottom of the interface, there are links for 'Feedback', 'English (US)', and copyright information: '© 2008 - 2019, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use'.

AWS STEP FUNCTIONS WORKFLOW STUDIO



Step Functions benefits

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

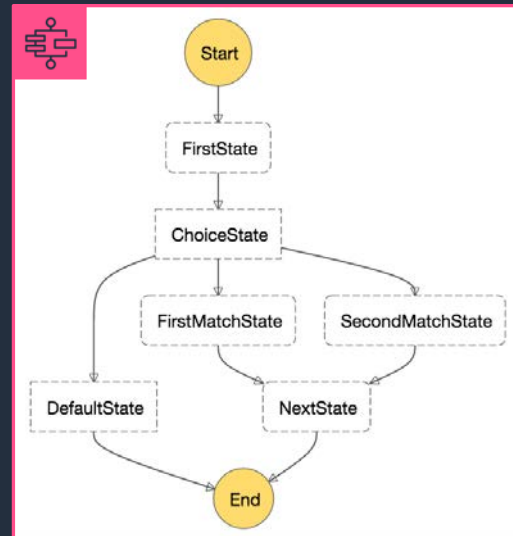
AWS Step Functions manages state, checkpoints and restarts for you to make sure that your application executes in order and as expected. Built-in try/catch, retry and rollback capabilities deal with errors and exceptions automatically.

Visual workflows

Define

- Drag-and-drop with Workflow Studio
- JSON (Amazon States Language)
- CDK (TypeScript, JavaScript, Python, Java, C#)
- Data Science SDK (Python)

Visualize



Execute and monitor

Execution Status: **Succeeded**

State Machine ARN: `arn:aws:states:::central-1:492419596455:stateMachine:Orderer`

Execution ID: `arn:aws:states:::central-1:492419596455:execution:Orderer-New_Order`

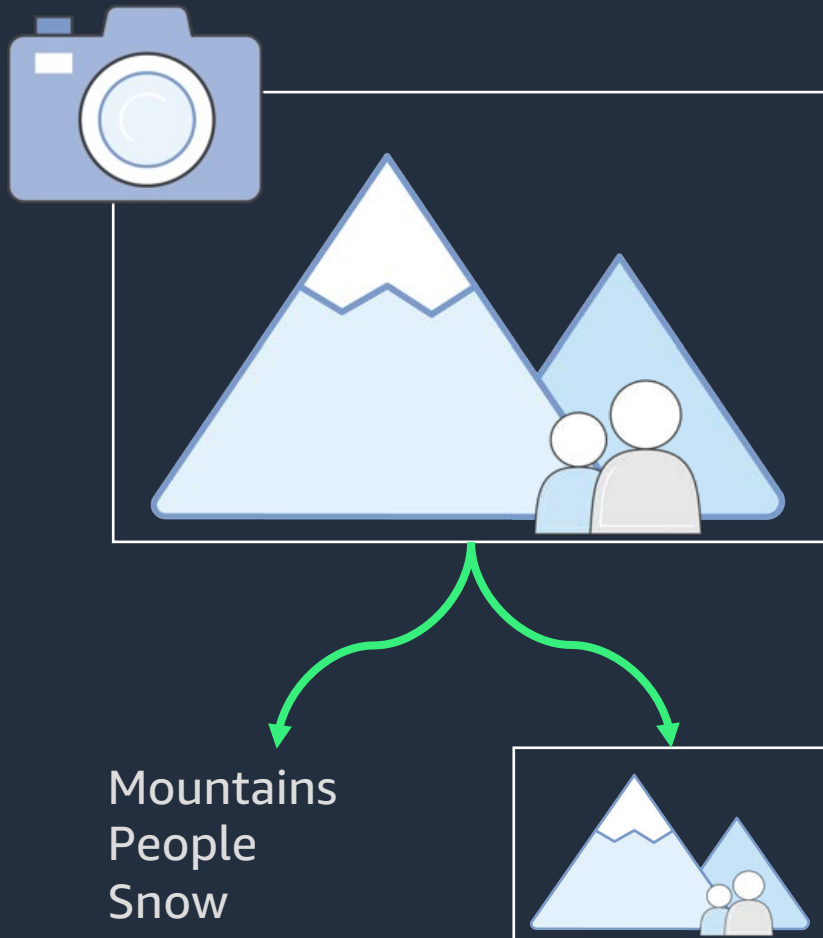
Started: Nov 20, 2016 9:58:28 AM

Closed: Nov 20, 2016 9:58:32 AM

ID	Type	Timestamp
1	ExecutionStarted	Nov 20, 2016 9:58:28 AM
2	TaskStateEntered	Nov 20, 2016 9:58:28 AM
3	LambdaFunctionScheduled	Nov 20, 2016 9:58:28 AM

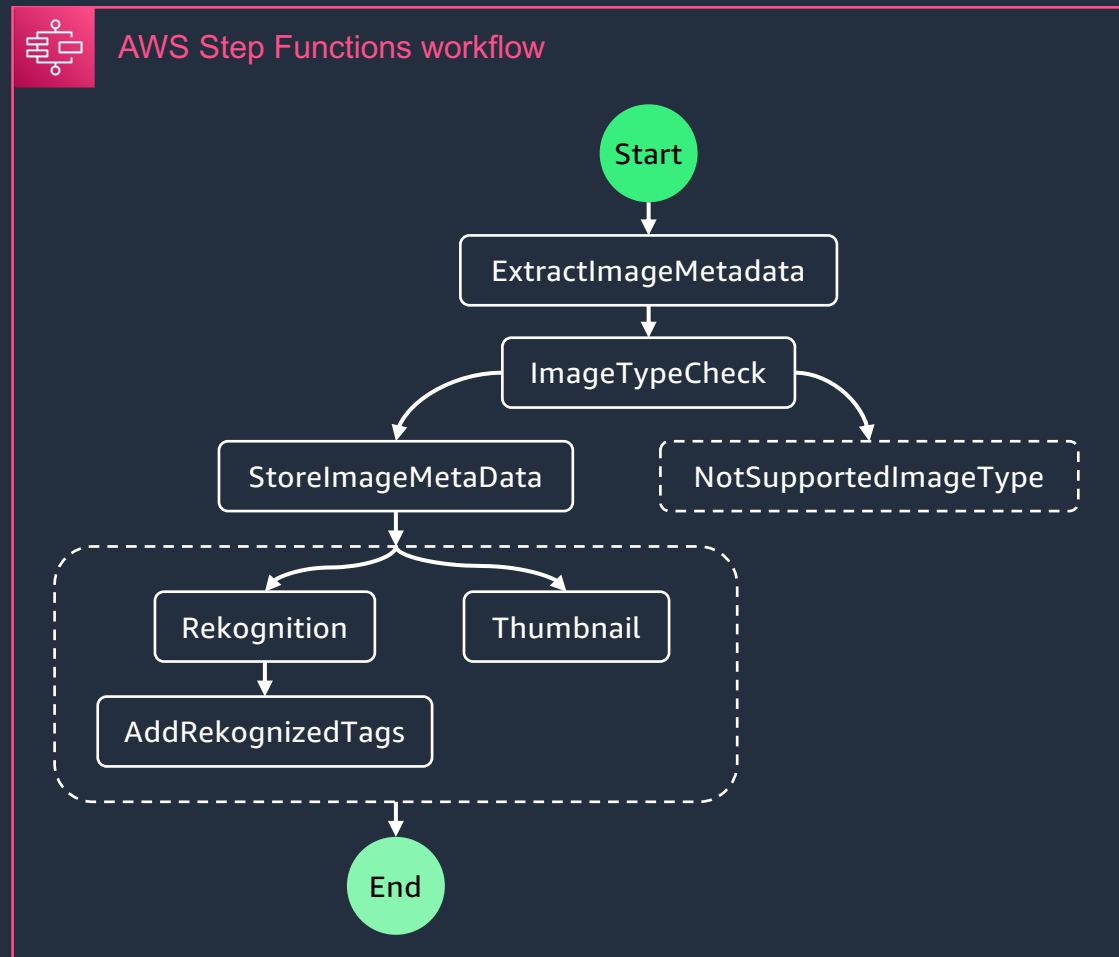
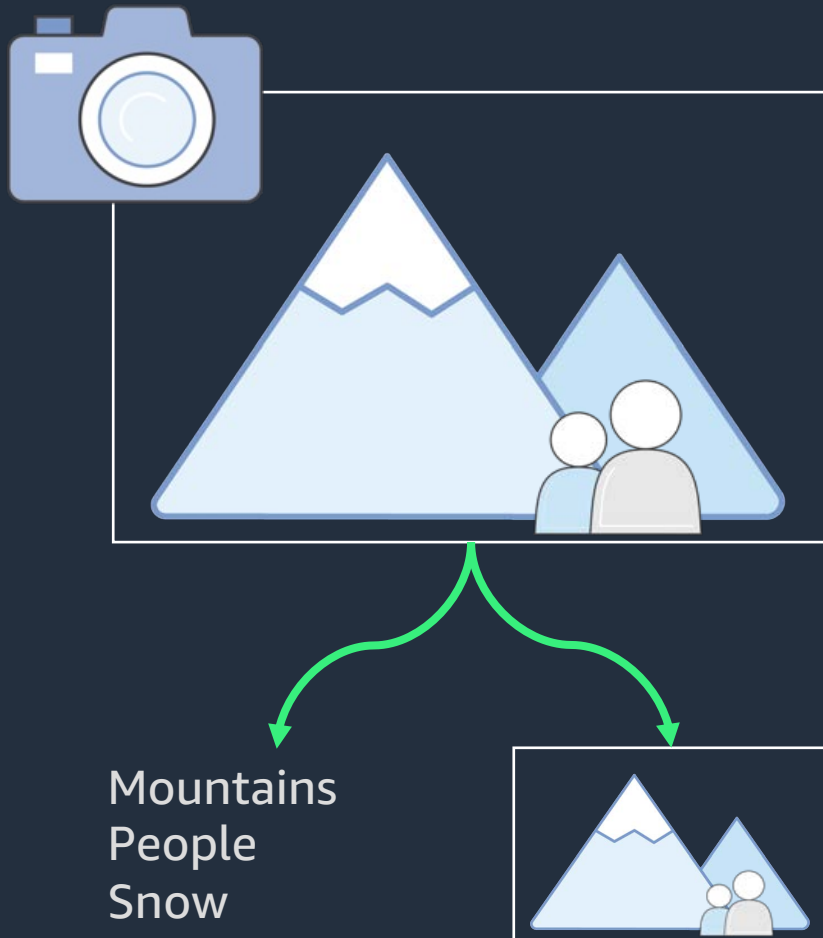
```
{
  "id": "1",
  "type": "ExecutionStarted",
  "details": {
    "roleArn": "arn:aws:iam::[redacted]:role/service-role/Temp211262019",
    "input": "(\\n \\tjobName: \\tjob-\\n \\tjobDefinition: \\tarn:aws:batch:ca-central-1:[redacted]:job-definition/sampleJobDefinition",
    "previous_event_id": "0",
    "event_timestamp": "1574789533445",
    "execution_arn": "arn:aws:states:ca-central-1:[redacted]:execution:Temp211262019:9f0fa2e2-ade7-74b9-40be-0a468b495709:c2073d15-496f-42d2-acd7"
```

An image processing workflow



1. Make a thumbnail
2. Identify features
3. Store image metadata

Build workflows using eight state types



Amazon States Language (ASL)

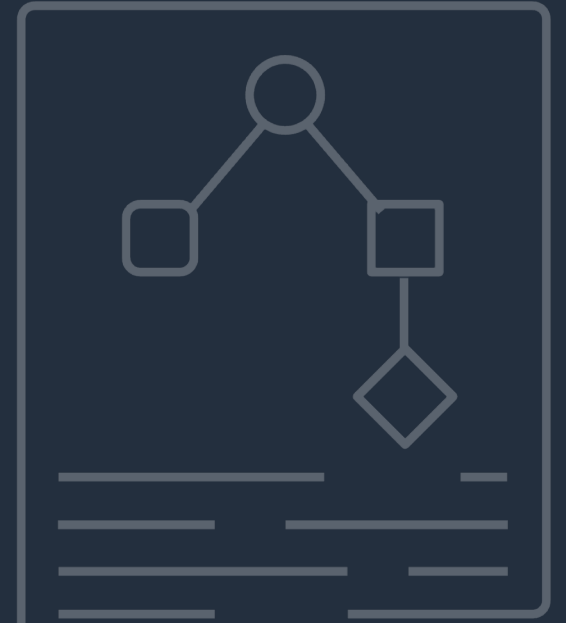
```
{
  "Comment": "An example of the Amazon States Language using a choice state.",
  "StartAt": "FirstState",
  "States": {
    "FirstState": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:FUNCTION_NAME",
      "Next": "ChoiceState"
    },
    "ChoiceState": {
      "Type": "Choice",
      "Choices": [ {
        "Variable": "$.foo",
        "NumericEquals": 1,
        "Next": "FirstMatchState"
      },
      {
        "Variable": "$.foo",
        "NumericEquals": 2,
        "Next": "SecondMatchState"
      } ],
      ...
    }
  }
}
```

<https://states-language.net/spec.html/>

States

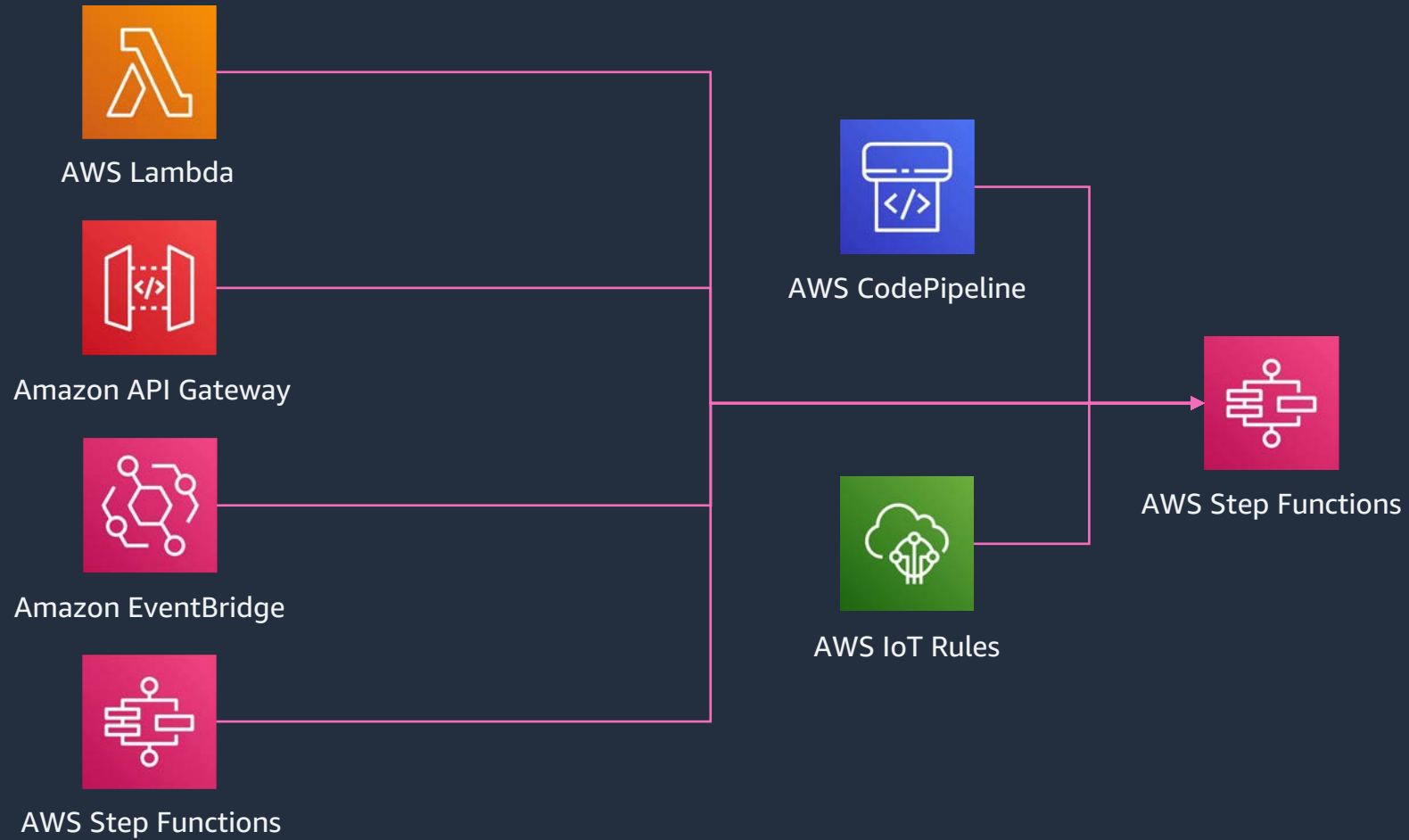
State types

- Task** Execute work
- Choice** Add branching logic
- Wait** Add a timed delay
- Parallel** Execute branches in parallel
- Map** Process each of an input array's items with a state machine
- Pass** Pass input to output
- Succeed** Signal a successful execution and stop
- Fail** Signal a failed execution and stop



Integrations

Triggers



Integration types

Optimized integrations

- Customized by Step Functions to provide additional functionality and UI elements in Workflow Studio.
- Support for 17 AWS services.
- No performance benefit over SDK integrations.

```
arn:aws:states:::serviceName:apiAction.[serviceIntegrationPattern]
```

SDK integrations

- Use standard AWS API calls directly in your workflows.
- Write less custom code: reduces need for Lambda functions.
- Support for over 200 AWS services and over 10,000 API actions.

```
arn:aws:states:::aws-sdk:serviceName:apiAction.[serviceIntegrationPattern]
```

Service integration patterns

Request Response

Step Functions will wait for an HTTP response and then progress to the next state. Step Functions will not wait for a job to complete.

Run a Job (.sync)

Call a service and have Step Functions wait for a job to complete.

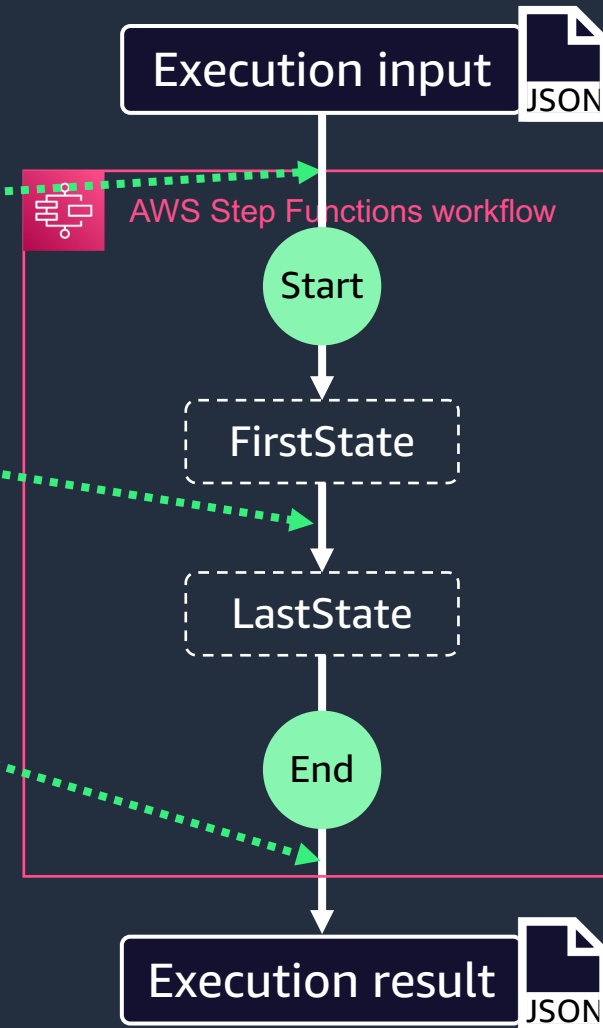
Wait for callback

Call a service with a task token and have Step Functions wait until that token is returned with a payload.

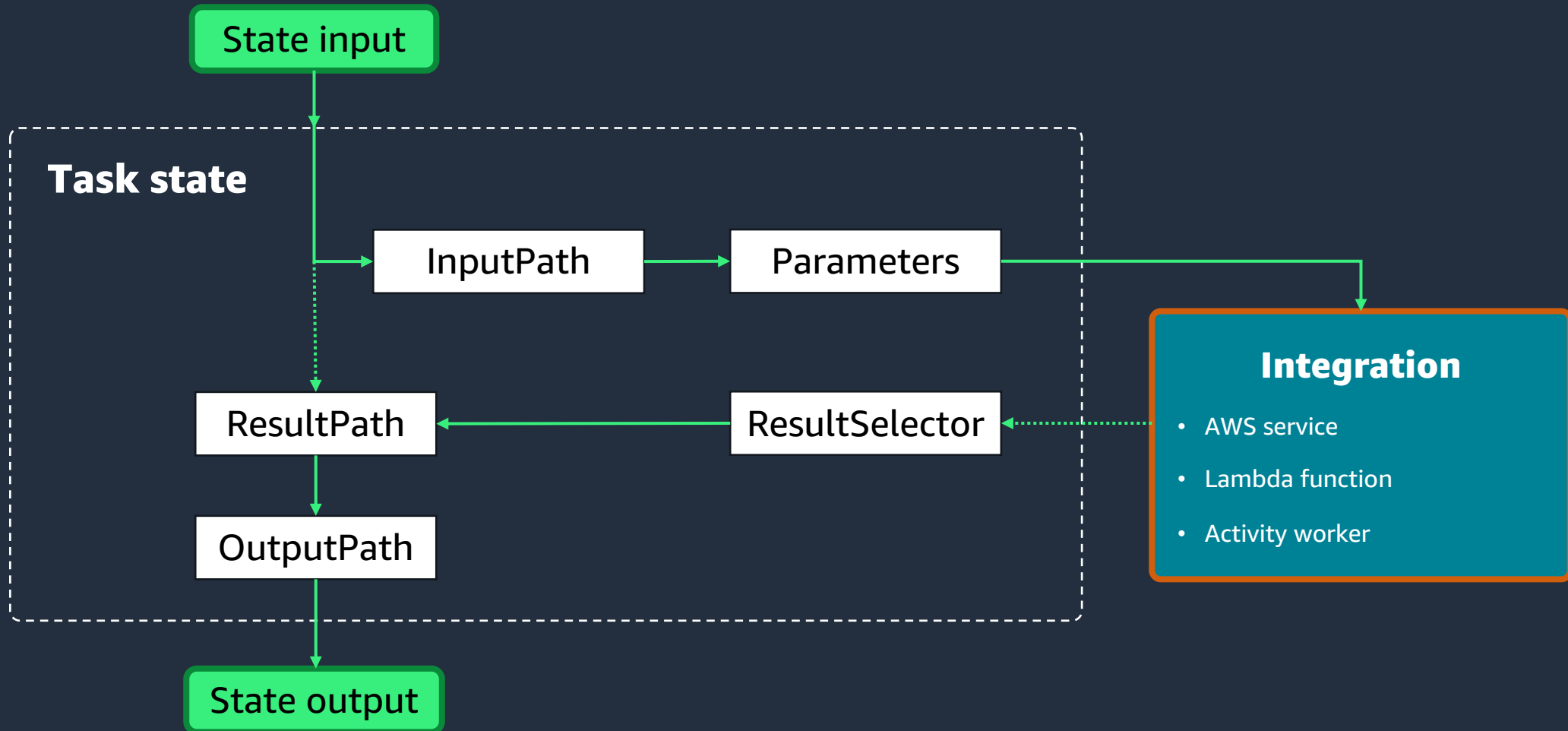
State machine data

Input and output processing

- A Step Functions execution receives a JSON text as input and **passes that input to the first state** in the workflow.
- Individual states **receive JSON as input** from the previous state and usually **pass JSON as output** to the next state.
- The output of the workflow's last state **becomes the result of the Step Functions execution.**
- ASL provides tools to **filter, manipulate and transform input and output** between states.



Anatomy of a state

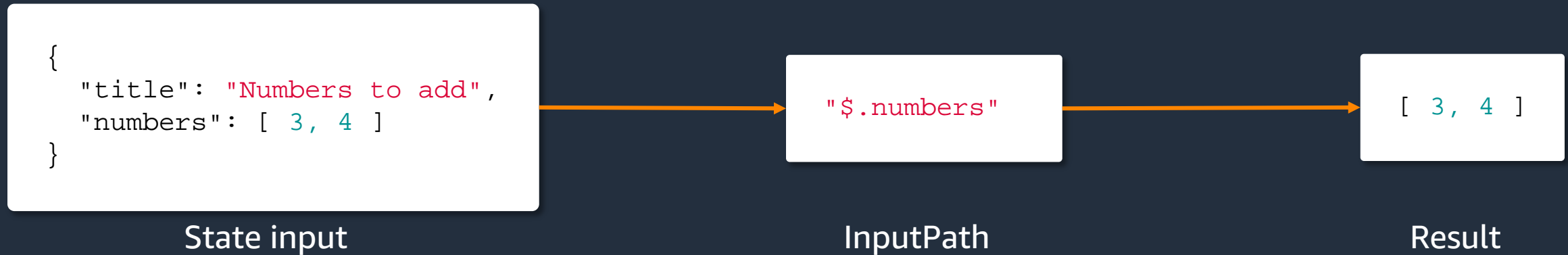


InputPath

Use the **InputPath** filter to select a portion of the JSON state input to use. Specify which part of the JSON to use with a JSONPath expression.

If not provided, state gets raw input, as-is.

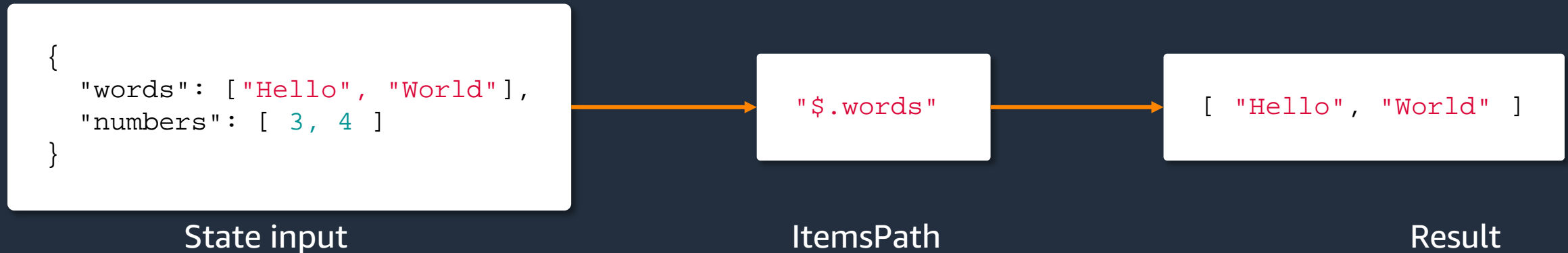
If null, state gets an empty JSON {}.



ItemsPath

Use **ItemsPath** field in a Map state to select an array in the input. Specify which part of the JSON to use with a JSONPath expression.

If not provided, state gets entire JSON.



Parameters

Use the **Parameters** fields to create a collection of key-value pairs that are passed as input to an AWS service integration.

The values can be static, or they can be selected from the filtered input using a JSONPath expression. When the value is selected with a JSONPath expression, the key name must end in `$. $`.

```
{  
  "title": "Numbers to add",  
  "numbers": [ 3, 4 ]  
}
```

State input

```
{  
  "staticValue": "Just a string",  
  "SecondNumber.$": "$.numbers[1]"  
}
```

Parameters

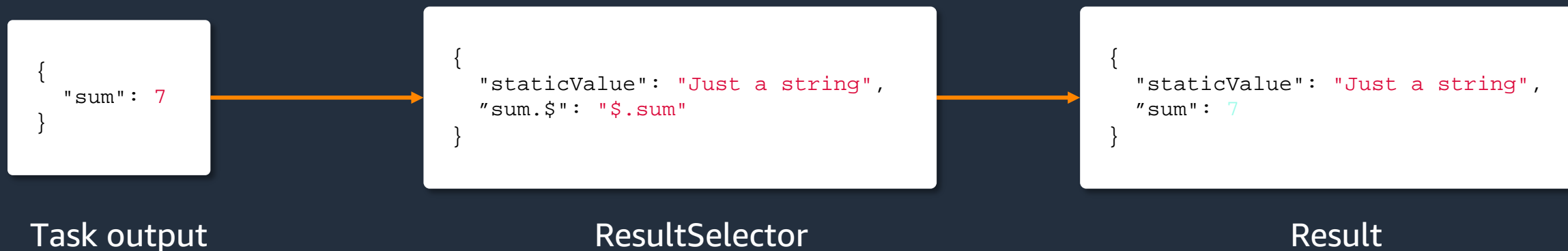
```
{  
  "staticValue": "Just a string",  
  "SecondNumber": 4  
}
```

Result

ResultSelector

Use the **ResultSelector** filter to construct a new JSON object using parts of the task result.

The values can be static, or they can be selected from the task result using a JSONPath expression. When the value is selected with a JSONPath expression, the key name must end in `$. $`.



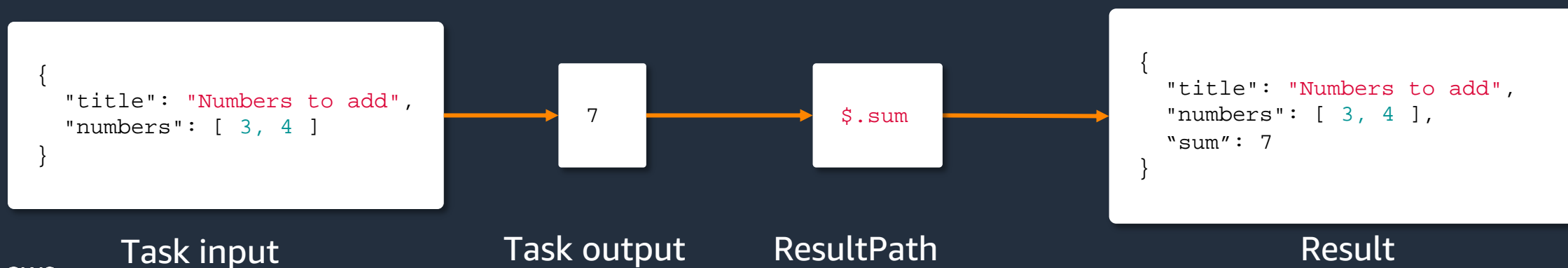
ResultPath

Use **ResultPath** to add the result into the original state input. The specified path indicates where to add the result.

If unspecified or \$, the result becomes the output and the input is discarded.

If null, state's input becomes state's output and task's output is discarded.

If a JSONPath expression, the result will be inserted into the state input.



OutputPath

Use **OutputPath** to filter the final result before it becomes the state's output



Context object

`$$.Execution .Id`

```
{
  "Execution": {
    "Id": "arn:aws:states:us-east-1:123456789012:execution:stateMachineName:executionName",
    "Input": {
      "key": "value"
    },
    "Name": "executionName",
    "RoleArn": "arn:aws:iam::123456789012:role...",
    "StartTime": "2019-03-26T20:14:13.192Z"
  },
  "State": {
    "EnteredTime": "2019-03-26T20:14:13.192Z",
    "Name": "Test",
    "RetryCount": 3
  },
  "StateMachine": {
    "Id": "arn:aws:states:us-east-1:123456789012:stateMachine:stateMachineName",
    "Name": "name"
  },
  "Task": {
    "Token": "h7XRiCdLtd/83p1E0dMccoxlzFhglSDKzPK9mBVKZsp7d9yrT1W"
  }
}
```

```
"Map": {
  "Item": {
    "Index": "Number",
    "Value": "String"
  }
}
```

Accessible from anywhere in the state machine via context object:

- Execution metadata
- State machine input

Intrinsic functions

```
{  
  "Name": "Alice",  
  "JSONString": "{\\"foo\\": \\"bar\\"}",  
  "JSON": {"Key": "Value"},  
  "Array": 12345  
}
```

States.Format('Hello, my name is {}. ', \$.Name) → Hello, my name is Alice.

States.StringToJson(\$.JSONString) → { "foo": "bar" }

States.JsonToString(\$.JSON) → {\\"Key\\": \\"Value\\"}

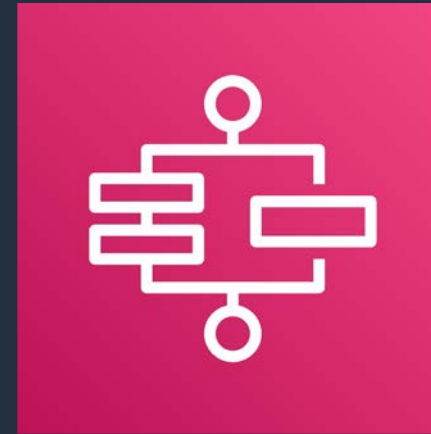
"BuildId.\$": "States.Array(\$.Id)" → "BuildId": [123456]

Workflow types

Workflow types



STANDARD



EXPRESS WORKFLOWS

Which one is right for my customer use case?

Standard workflows

- IT automation
- Report generation
- Order processing
- Payment and billing processing
- ML model training
- ETL and big data orchestration (AWS Glue, Amazon EMR)
- Media processing (video, image, audio)

Express workflows

- Event driven microservices orchestration
- High volume data processing
- IoT data ingestion
- Order / cart validation
- Payment reconciliation

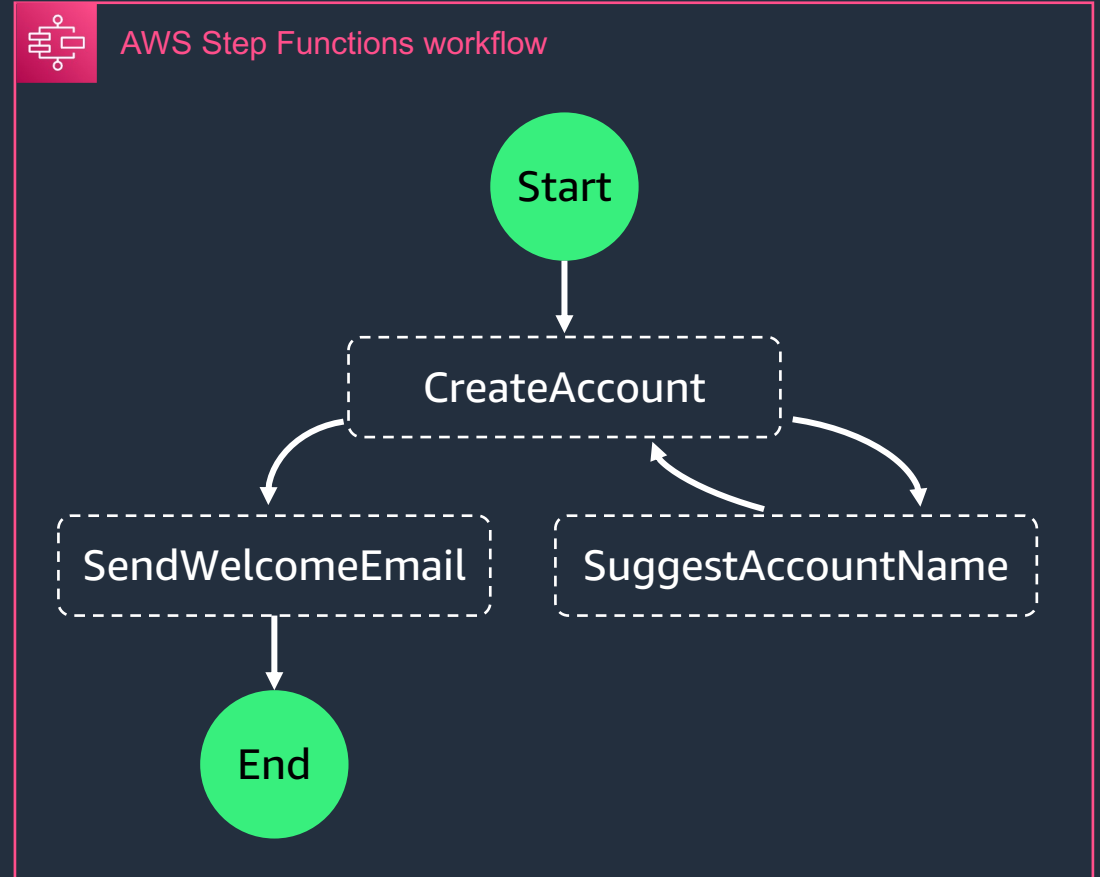
Standard vs. express workflows

	Standard	Express
Maximum duration	365 days	5 minutes
Execution start rate	Over 2,000 per second	Over 100,000 per second
State transition rate	Over 4,000 per second per account	Nearly unlimited
Executions	Executions persisted for 90 days in Step Functions, with tooling for visual debugging in the console, and have ARNs. Optionally sent to CloudWatch Logs.	Executions logs are only sent to CloudWatch Logs.

Workflow patterns

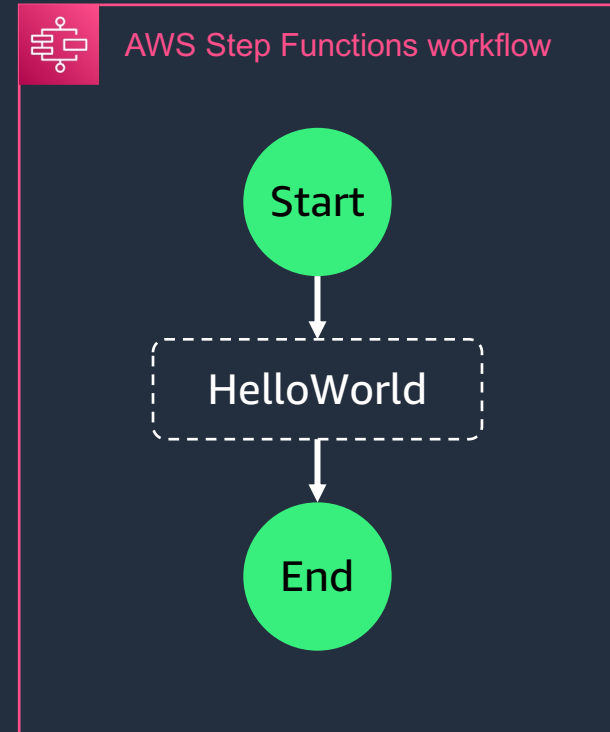
Error handling - Catch

```
"CreateAccount": {  
  "Type": "Task",  
  ...  
  "Catch": [  
    {  
      "ErrorEquals": ["AccountAlreadyExistsError"],  
      "Next": "SuggestAccountName"  
    }  
  ]  
}
```

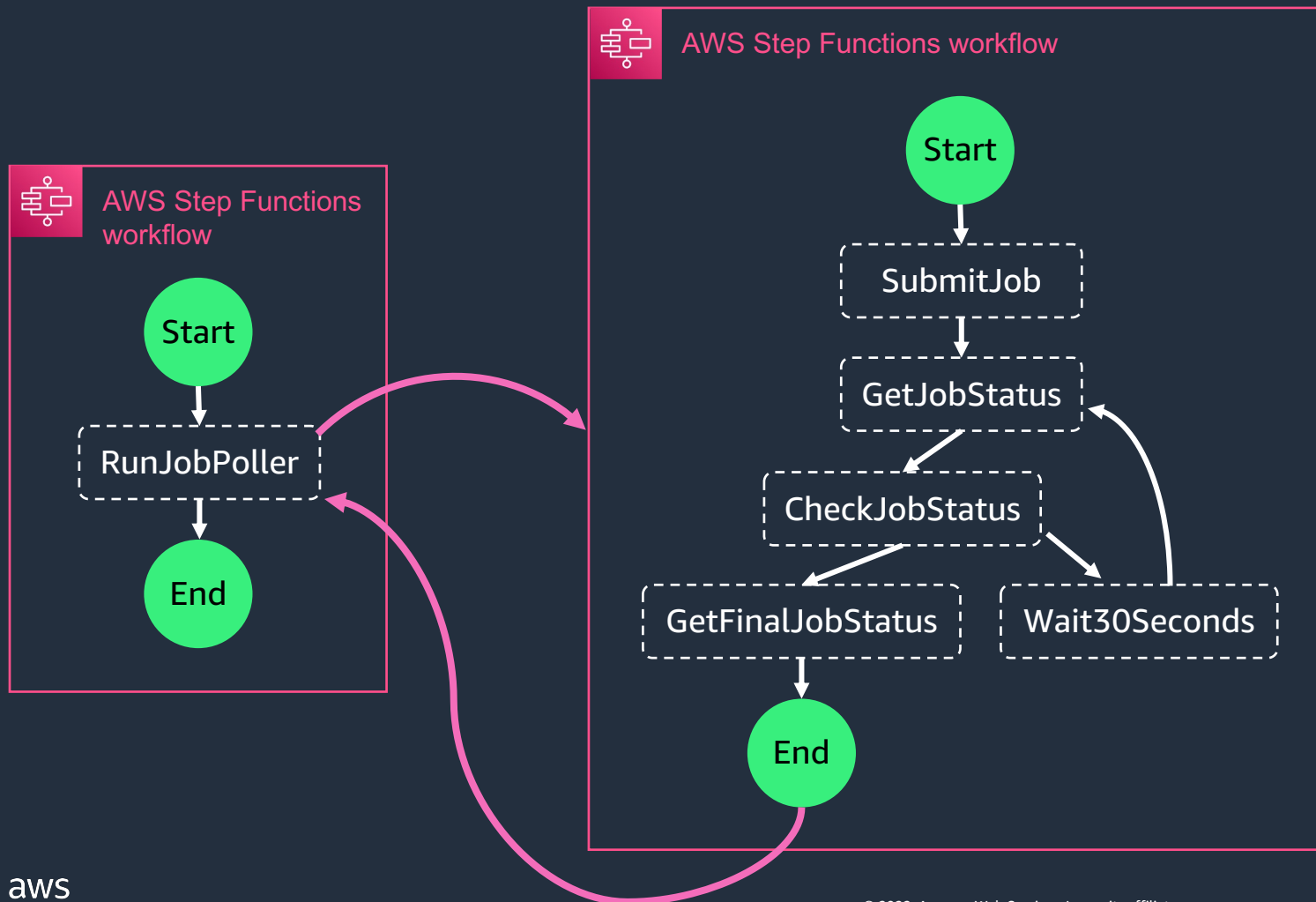


Error handling - Retry

```
"HelloWorld": {  
  "Type": "Task",  
  ...  
  "Retry": [  
    {  
      "ErrorEquals": ["States.TaskFailed"],  
      "IntervalSeconds": 1,  
      "MaxAttempts": 2,  
      "BackoffRate": 2.0  
    },  
    ...  
  ]  
}
```



Nested “child” workflows



- Easily build complex workflows with libraries of state machines.
- Swap and reorganize workflow modules without customizing code.
- Simplifies development and deployment.

Use cases

Use cases for AWS Step Functions

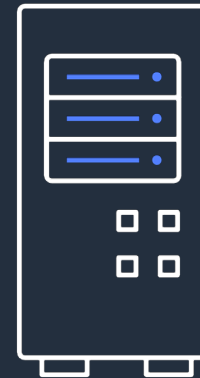
Data processing



IT and security automation



Machine learning



Microservice orchestration



Developer tooling

Workflow Studio

- Low-code visual workflow designer
- Rapid prototyping and experimentation
- Generates and validates ASL template
- Export ASL for local development and infrastructure as code integration

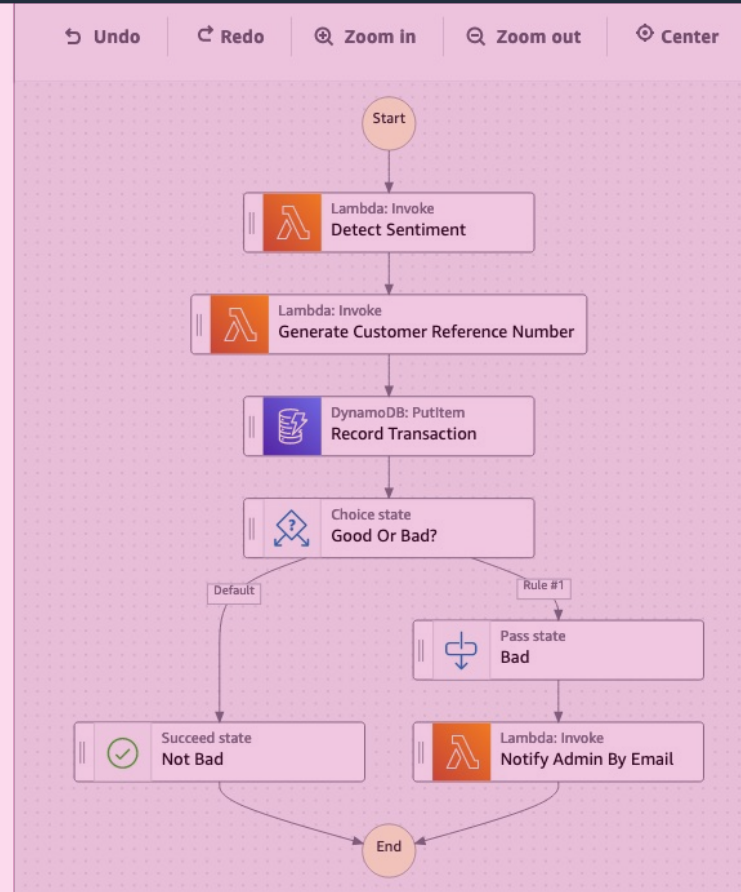
States Browser

Search

Actions Flow

- AWS Lambda Invoke
- Amazon SNS Publish
- Amazon ECS RunTask
- AWS Step Functions StartExecution
- AWS Glue StartJobRun
- AWS Glue DataBrew StartJobRun
- Amazon EventBridge PutEvents
- AWS Batch SubmitJob
- Amazon API Gateway Request
- Amazon Athena StartQueryExecution

Canvas



Inspector Panel

Form Definition

Workflow

Start at
Choose which state is the starting point of the workflow

Detect Sentiment

Comment - optional
A human-readable description of the state machine.

A serverless form processor with a number of useful reference components

TimeoutSeconds - optional
The maximum number of seconds an execution of the state machine can run. If it runs longer than the specified time, the execution fails with a States.Timeout.

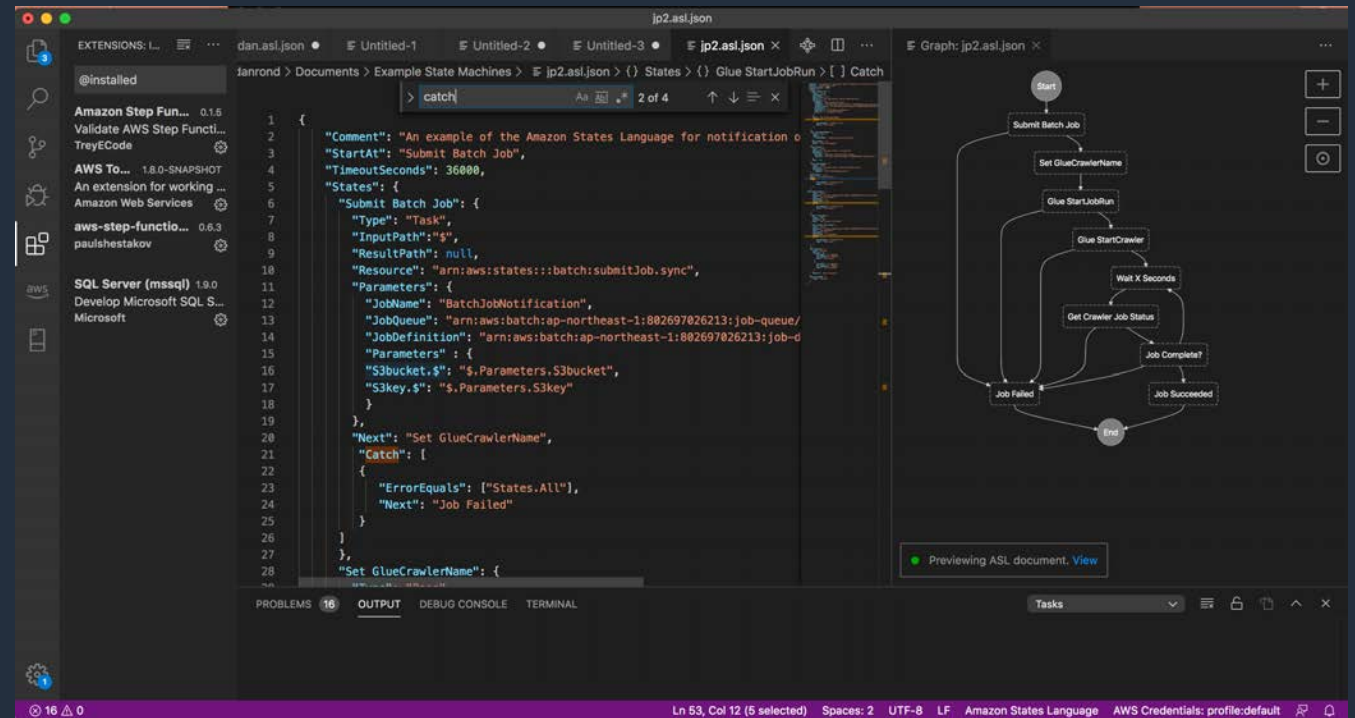
600

AWS Toolkit for Visual Studio Code

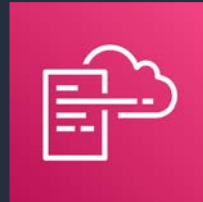


Local development platform

- Build state machines locally
- Workflow visualization
- State machine linting
- Code snippets and IntelliSense



Your Choice for Infrastructure as Code



AWS CloudFormation

DECLARATIVE

IMPERATIVE

AWS Serverless Application Model

- Declarative / template based
- JSON / YAML



AWS Serverless Application Model



AWS Cloud Development Kit

AWS Cloud Development Kit

- Imperative
- Author state machines in TypeScript, JavaScript, Python, Java, C#

Step Functions Local

- A downloadable version of Step Functions that lets you develop and test applications using a version of Step Functions **running in your own development environment**.
- **Local mocking** of all optimized and SDK service integrations.
- Available as a JAR package and as a Docker image.

```
java -jar StepFunctionsLocal.jar || docker run -p 8083:8083 amazon/aws-stepfunctions-local  
aws stepfunctions --endpoint-url http://localhost:8083 command
```

Best practices

Best practices

- Choose the right **workflow type**.
- Use **timeouts** to avoid stuck executions.
- Use **Amazon S3** to pass large payloads.
- Avoid reaching **history quota**.
- Handle Lambda **service exceptions**.

In closing



Useful resources

Serverless lens

<https://docs.aws.amazon.com/wellarchitected/latest/serverless-applications-lens/welcome.html>

Serverless Land

<https://serverlessland.com/>

Use cases

<https://aws.amazon.com/step-functions/use-cases/>

Customer testimonials

<https://aws.amazon.com/step-functions/customer-testimonials/>

re:Invent sessions

<https://www.youtube.com/watch?v=2zCvMcZTr1E>

<https://www.youtube.com/watch?v=z1rIfyP1LGg>



Thank you!

Oskar Neumann
Solutions Architect