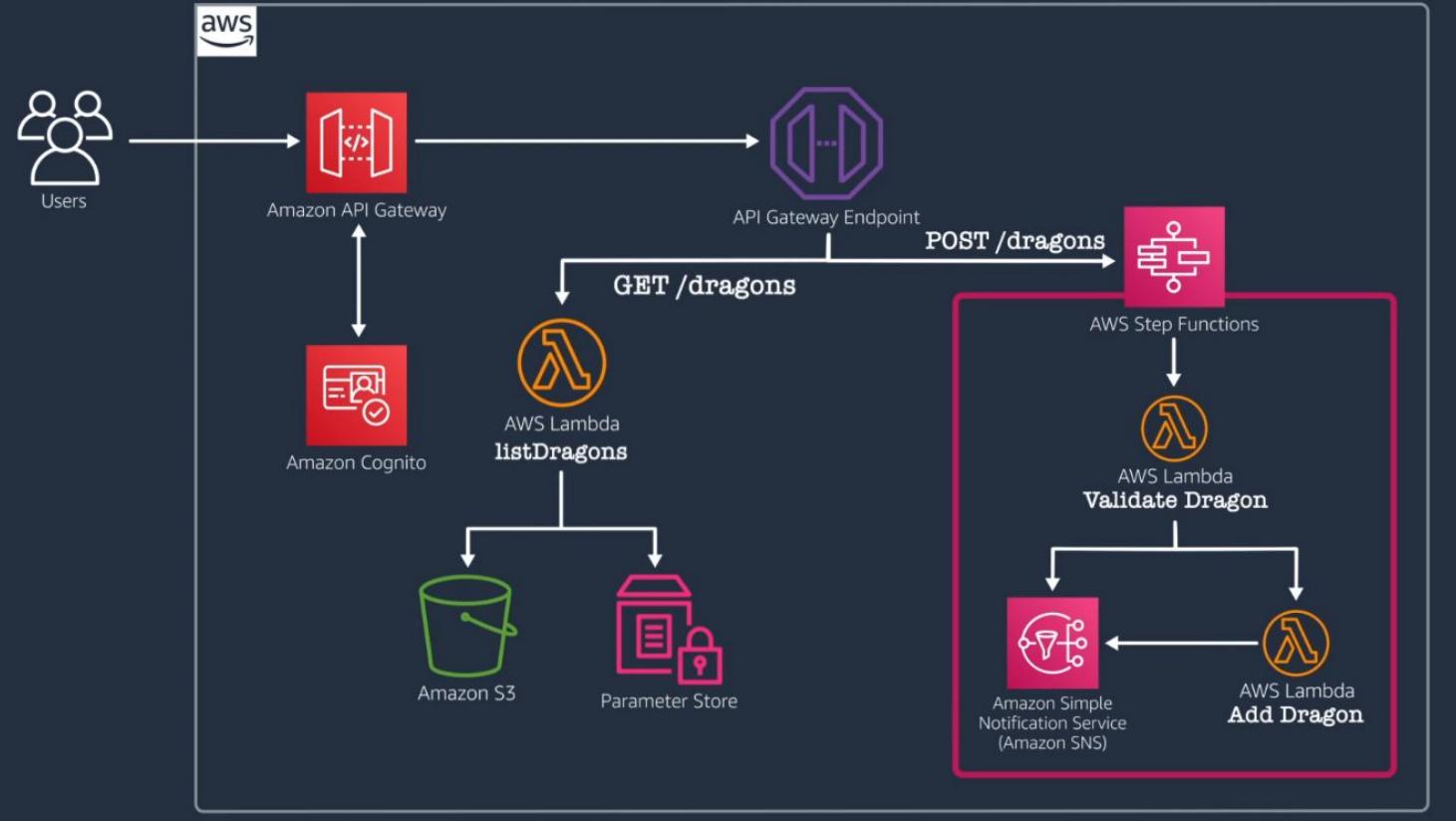


04 - AWS Step Function

recap

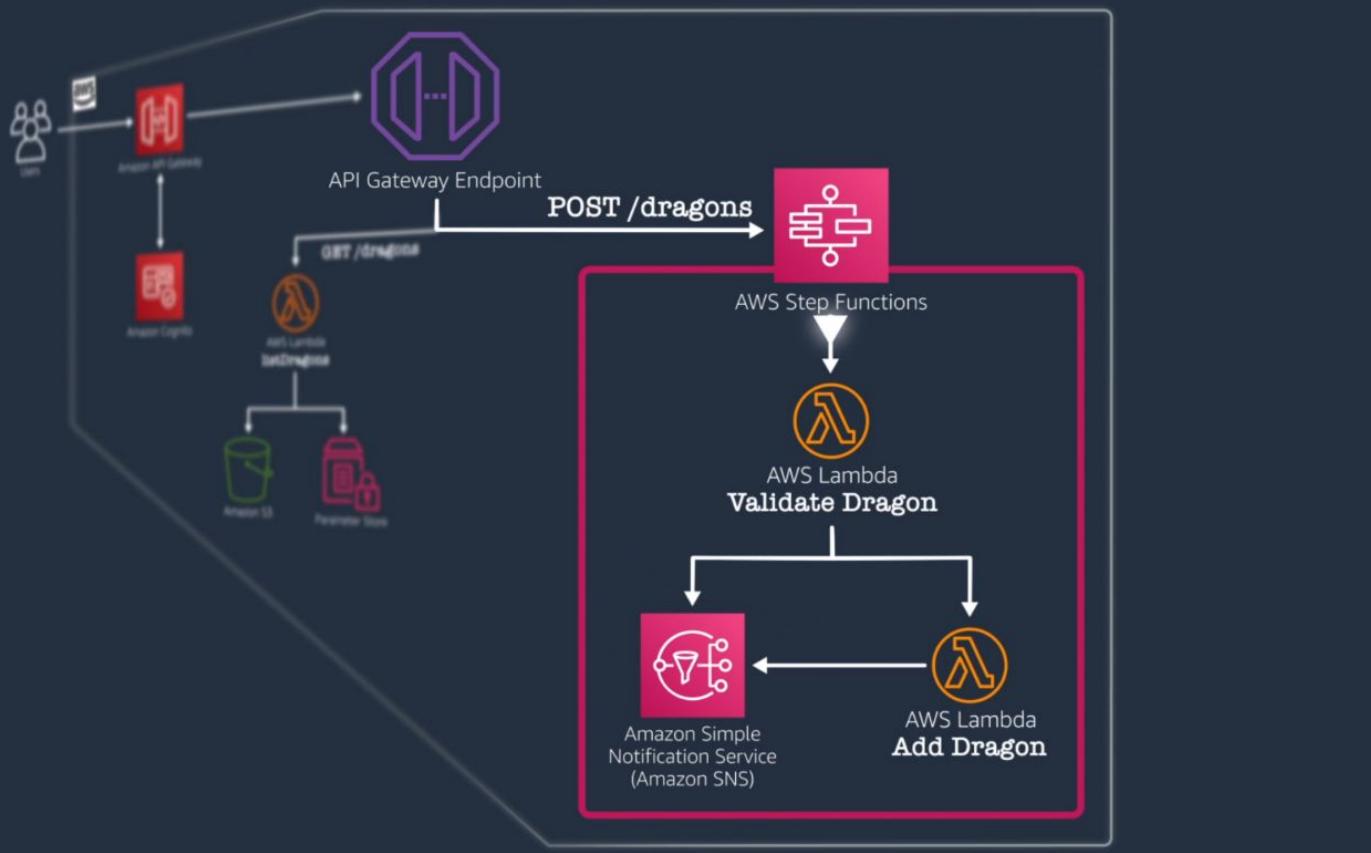
We have set up API Gateway with the POST method
that will accept the data about a newly cited dragon.

Currently, it is backed by a mock endpoint.



We have already created the Lambda functions
for validate dragon and add dragon.

but we haven't implemented anything that
makes these Lambda functions execute in the correct order.



We need to determine what order to run each task, whether to run tasks in parallel, or to run one task over another based on some sort of criteria.

We could have one Lambda call the next one depending on decisions made in the code.

But the problem with that is that those Lambda functions would be **tightly coupled**.

In general, it's **not a great idea** to **inject your workflow logic
into your code** when working with something like Lambda.

In order to change the way your process works,
you'll maybe need multiple code changes across multiple Lambda functions.

what tool can we use to create this workflow?

AWS Step Functions

AWS Step Functions is a serverless workflow tool.

The workflow in Step Functions is called a state machine.

Step Functions is based on the concept of **state machines**, states, and tasks.

A **State Machine** is a computational model used to represent the **behavior of a system**.

It defines a set of states and transitions between those states.

Real-World Example: Imagine a Traffic Light as a State Machine

Used in various domains like:

- Software Engineering (Workflow Management)
- Automation and Control Systems
- Game Development (Game States)
- Finite State Machines in Theory of Computation

State machines are built using
Amazon States Language to represent the required elements.

The Amazon States Language
is a JSON-based, structured language used
to define the state machine, the workers, the workflow and so on.

Step Functions allows you to create a state machine
that consists of one or many states.

It keeps track of the state of the state machine,
as well as keeping a history of executions of that state machine.

Step Functions logs the state of each step
so that you have the ability to diagnose and debug any problems encountered.

A **state** is essentially one step in your workflow.

A step might be executing a Lambda function,
running code inside of a container,
reaching out to another AWS service,
or many other options.

In our example, we will be creating different states for validating a dragon, sending an error text message, adding a dragon, and sending a success text message.

AWS Step Functions is **serverless**.

We don't need to manage a server to host the state machine.

The code running on Lambda also is not aware of the fact that it is part of a state machine.

Step Function State Types

In AWS Step Functions,
a "**state**" refers to a single step or task in your state machine workflow.

Each state defines what happens when it's executed
and what state should follow it.

There are several state types, each designed for different purposes.

Task State: This is where you specify exactly what to do when the state is executed.

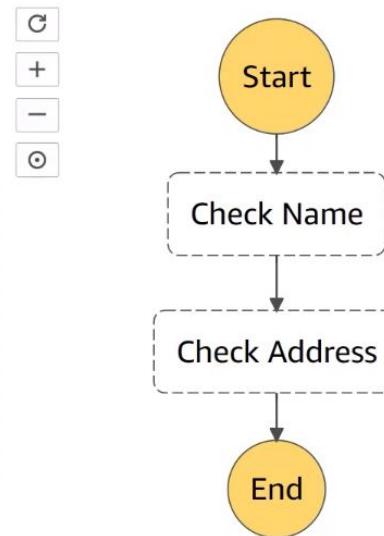
It can represent a single unit of work,
like invoking a Lambda function, making an API call, or running a container.

One of these integration is to invoke a lambda function,
wait for it's return and then use that to continue into the workflow.

Let's take an example here.

Definition

```
1 {
2     "Comment": "Task example",
3     "StartAt": "Check Name",
4     "States": {
5         "Check Name": {
6             "Type": "Task",
7             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
8             "Parameters": {
9                 "command": "CHECK_NAME",
10                "data": {
11                    "name.$": "$.application.name"
12                }
13            },
14            "Next": "Check Address"
15        },
16        "Check Address": {
17            "Type": "Task",
18            "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",
19            "Parameters": {
20                "command": "CHECK_ADDRESS",
21                "data": {
22                    "address.$": "$.application.address"
23                }
24            },
25            "End": true
26        }
27    }
28 }
```



validate name and address

In AWS Step Functions, are written in Amazon States Language using JSON.

```
{  
  "Comment": "A simple state machine example",  
  "StartAt": "HelloWorld",  
  "States": {  
    "HelloWorld": {  
      "Type": "Task",  
      "Resource": "arn:aws:lambda:us-east-1:123456789012:function:my-function",  
      "End": true  
    }  
  }  
}
```

Definition

```
1 * {
2     "Comment": "Task example",
3     "StartAt": "Check Name",
4     "States": {
5         "Check Name": {
6             "Type": "Task",
7             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
8             "Parameters": {
9                 "command": "CHECK_NAME",
10                "data": {
11                    "name.$": "$.application.name"
12                }
13            },
14            "Next": "Check Address"
15        },
16        "Check Address": {
17            "Type": "Task",
18            "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",
19            "Parameters": {
20                "command": "CHECK_ADDRESS",
21                "data": {
22                    "address.$": "$.application.address"
23                }
24            },
25            "End": true
26        }
27 }
```

Let's look at the code to validate name and address

Definition

```
1 * {
2     "Comment": "Task example",
3     "StartAt": "Check Name", ←
4     "States": { ←
5         "Check Name": {
6             "Type": "Task",
7             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
8             "Parameters": {
9                 "command": "CHECK_NAME",
10                "data": {
11                    "name.$": "$.application.name"
12                }
13            },
14            "Next": "Check Address"
15        },
16        "Check Address": {
17            "Type": "Task",
18            "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",
19            "Parameters": {
20                "command": "CHECK_ADDRESS",
21                "data": {
22                    "address.$": "$.application.address"
23                }
24            },
25            "End": true
26        }
27 }
```

a "StartAt" field to specify the initial state
"States" field where you define individual states.

Definition

```
1 * {
2     "Comment": "Task example",
3     "StartAt": "Check Name",
4     "States": {
5         "Check Name": {
6             "Type": "Task",
7             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
8             "Parameters": {
9                 "command": "CHECK_NAME",
10                "data": {
11                    "name.$": "$.application.name"
12                }
13            },
14            "Next": "Check Address"
15        },
16        "Check Address": {
17            "Type": "Task",
18            "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",
19            "Parameters": {
20                "command": "CHECK_ADDRESS",
21                "data": {
22                    "address.$": "$.application.address"
23                }
24            },
25            "End": true
26        }
27 }
```

Each state is defined as a JSON object with various properties like "Type", "Resource", and others depending on the type of state.

Definition

```
1 * {
2     "Comment": "Task example",
3     "StartAt": "Check Name",
4     "States": {
5         "Check Name": {
6             "Type": "Task",
7             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
8             "Parameters": {
9                 "command": "CHECK_NAME",
10                "data": {
11                    "name.$": "$.application.name"
12                }
13            },
14            "Next": "Check Address"
15        },
16        "Check Address": {
17            "Type": "Task",
18            "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",
19            "Parameters": {
20                "command": "CHECK_ADDRESS",
21                "data": {
22                    "address.$": "$.application.address"
23                }
24            },
25            "End": true
26        }
27 }
```

Next specifies the state that should be executed next after the current state completes successfully.

Definition

```
1 * {
2     "Comment": "Task example",
3     "StartAt": "Check Name",
4     "States": {
5         "Check Name": {
6             "Type": "Task",
7             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
8             "Parameters": {
9                 "command": "CHECK_NAME",
10                "data": {
11                    "name.$": "$.application.name"
12                }
13            },
14            "Next": "Check Address"
15        },
16        "Check Address": {
17            "Type": "Task",
18            "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",
19            "Parameters": {
20                "command": "CHECK_ADDRESS",
21                "data": {
22                    "address.$": "$.application.address"
23                }
24            },
25            "End": true
26        }
27 }
```

Definition

```
1 * {
2     "Comment": "Task example",
3     "StartAt": "Check Name",
4     "States": {
5         "Check Name": {
6             "Type": "Task",
7             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
8             "Parameters": {
9                 "command": "CHECK_NAME",
10                "data": {
11                    "name.$": "$.application.name"
12                }
13            },
14            "Next": "Check Address"
15        },
16        "Check Address": {
17            "Type": "Task",
18            "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",
19            "Parameters": {
20                "command": "CHECK_ADDRESS",
21                "data": {
22                    "address.$": "$.application.address"
23                }
24            },
25            "End": true
26        }
27 }
```

End is used to indicate that the current state is a terminal state, meaning it's the last state in the state machine. When a state is marked as an "End" state, it signifies that the state machine has completed its execution.

Definition

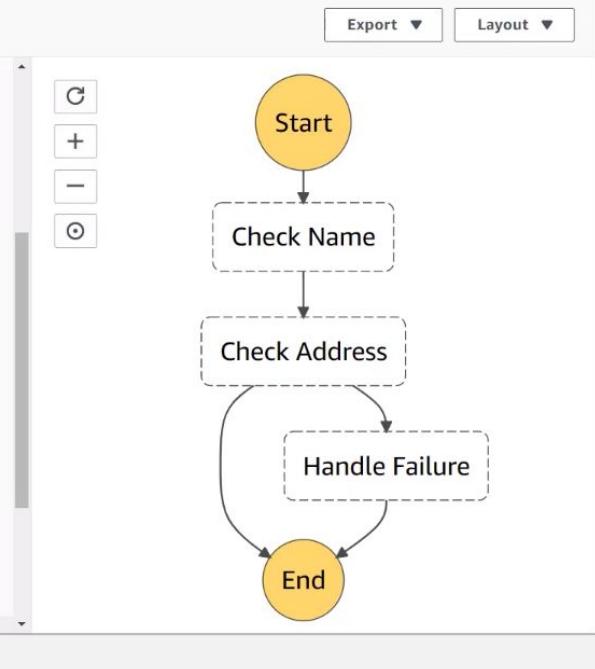
```
1 * {
2     "Comment": "Task example",
3     "StartAt": "Check Name",
4     "States": {
5         "Check Name": {
6             "Type": "Task",
7             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
8             "Parameters": {
9                 "command": "CHECK_NAME",
10                "data": {
11                    "name.$": "$.application.name"
12                }
13            },
14            "Next": "Check Address"
15        },
16        "Check Address": {
17            "Type": "Task",
18            "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",
19            "Parameters": {
20                "command": "CHECK_ADDRESS",
21                "data": {
22                    "address.$": "$.application.address"
23                }
24            },
25            "End": true
26        }
27    }
28 }
```

In the Parameters field, you map input parameters using the "inputParam.\$": "\$.inputParam" syntax.



Definition

```
16 "Check Address": {  
17     "Type": "Task",  
18     "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",  
19     "Parameters": {  
20         "command": "CHECK_ADDRESS",  
21         "data": {  
22             "address.$": "$.application.address"  
23         }  
24     },  
25     "Retry": [  
26         {  
27             "ErrorEquals": [  
28                 "KnownError"  
29             ],  
30             "IntervalSeconds": 1,  
31             "MaxAttempts": 2,  
32             "BackoffRate": 2  
33         }  
34     ],  
35     "Catch": [  
36         {  
37             "ErrorEquals": [  
38                 "KnownError"  
39             ],  
40             "Next": "Handle Failure"  
41         }  
42     ]  
43 }
```



Export

Layout

Let's add another state here to Handle Failure



buildingmodernapps @ 3022-1...

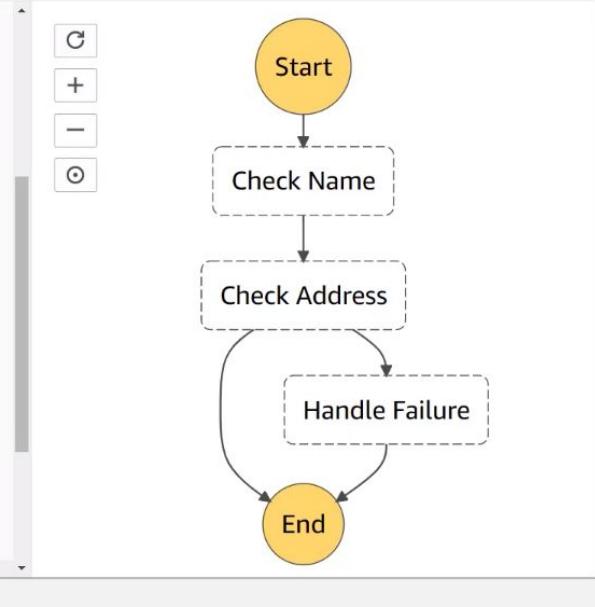
N. Virginia

Support



Definition

```
16 "Check Address": {  
17     "Type": "Task",  
18     "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkName",  
19     "Parameters": {  
20         "command": "CHECK_ADDRESS",  
21         "data": {  
22             "address.$": "$.application.address"  
23         }  
24     },  
25     "Retry": [  
26         {  
27             "ErrorEquals": [  
28                 "KnownError"  
29             ],  
30             "IntervalSeconds": 1,  
31             "MaxAttempts": 2,  
32             "BackoffRate": 2  
33         }  
34     ],  
35     "Catch": [  
36         {  
37             "ErrorEquals": [  
38                 "KnownError"  
39             ],  
40             "Next": "Handle Failure"  
41         }  
42     ]  
43 }
```

[Export](#)[Layout](#)

The **retry** parameter is used to define how many retry attempts, back off rate, and intervals between items when an error is caught.

Choice State: This state adds branching logic to your workflow.

It allows you to make decisions based on the input and transition to different states accordingly.



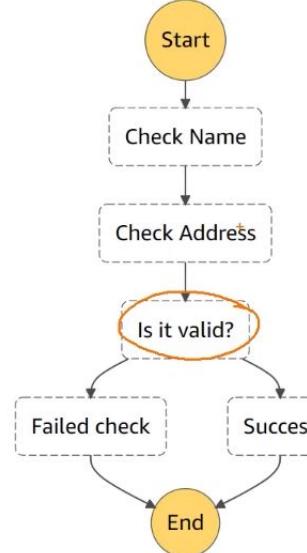
Definition

```
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
```

```
        },
        "Next": "Is it valid?"
    },
    "Is it valid?": {
        "Type": "Choice",
        "Choices": [
            {
                "Variable": "$.valid",
                "BooleanEquals": false,
                "Next": "Failed check"
            }
        ],
        "Default": "Success"
    },
    "Failed check": {
        "Type": "Pass",
        "End": true
    },
    "Success": {
        "Type": "Pass",
        "End": true
    }
}
```



Export ▾ Layout ▾



A sample of Choice State type

Feedback

English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

Pass State: This state simply passes its input to its output.

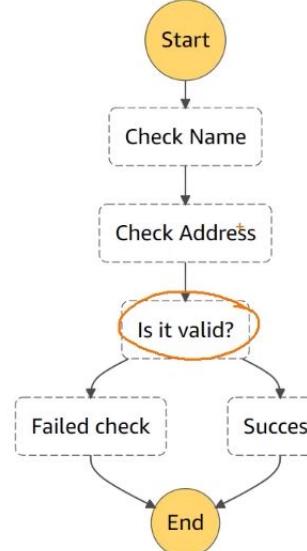
It's often used for debugging or to transform the input in some way.



Definition

```

23
24
25     },
26     "Next": "Is it valid?"
27 },
28 "Is it valid?": {
29     "Type": "Choice",
30     "Choices": [
31         {
32             "Variable": "$.valid",
33             "BooleanEquals": false,
34             "Next": "Failed check"
35         }
36     ],
37     "Default": "Success"
38 },
39 "Failed check": {
40     "Type": "Pass",
41     "End": true
42 },
43 "Success": {
44     "Type": "Pass",
45     "End": true
46 }
47 }
```



Export ▾

Layout ▾

In this simple examples, the Success and Failed Check states are set to Pass for simplicity.



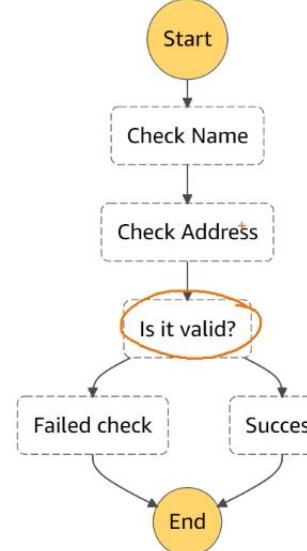
Definition

```

23
24
25     },
26     "Next": "Is it valid?"
27 },
28 "Is it valid?": {
29     "Type": "Choice",
30     "Choices": [
31         {
32             "Variable": "$.valid",
33             "BooleanEquals": false,
34             "Next": "Failed check"
35         }
36     ],
37     "Default": "Success"
38 },
39 "Failed check": {
40     "Type": "Pass",
41     "End": true
42 },
43 "Success": {
44     "Type": "Pass",
45     "End": true
46 }
47 }
```



Export ▾ Layout ▾



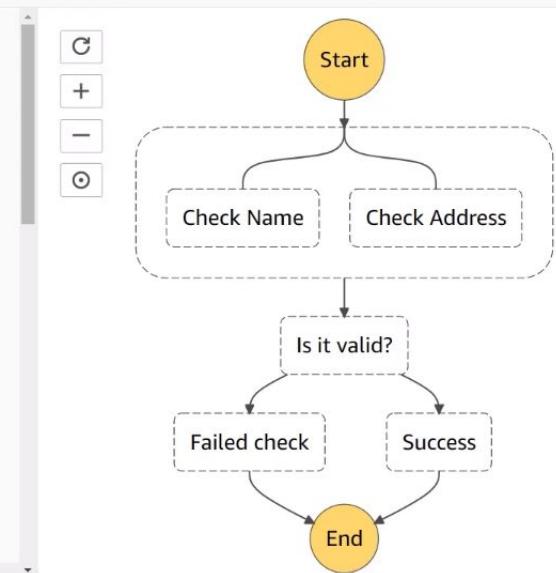
why check name and check address are done in a serial way? Check address doesn't need the output of check name to work. They could be done in parallel.

Parallel State: This state enables parallel execution of multiple branches of your workflow.
Each branch is a separate state.



Definition

```
1  {
2      "Comment": "Parallel example",
3      "StartAt": "Check Data",
4      "States": {
5          "Check Data": {
6              "Type": "Parallel",
7              "Branches": [
8                  {
9                      "StartAt": "Check Name",
10                     "States": {
11                         "Check Name": {
12                             "Type": "Task",
13                             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
14                             "Parameters": {
15                                 "command": "CHECK_NAME",
16                                 "data": {
17                                     "name.$": "$.application.name"
18                                 }
19                             },
20                             "End": true
21                         }
22                     }
23                 },
24                 {
25                     "StartAt": "Check Address",
26                     "States": {
27                         "Check Address": {
28                             "Type": "Task",
29                             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
30                             "Parameters": {
31                                 "command": "CHECK_ADDRESS",
32                                 "data": {
33                                     "address.$": "$.application.address"
34                                 }
35                             },
36                             "End": true
37                         }
38                     }
39                 }
40             }
41         }
42     }
43 }
```





Services

Resource Groups



buildingmodernapps @ 3022-1...

N. Virginia

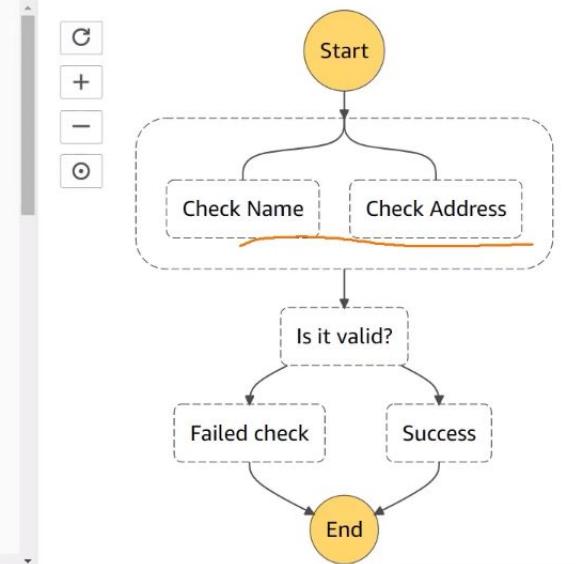
Support

Guest



Definition

```
1  {
2      "Comment": "Parallel example",
3      "StartAt": "Check Data",
4      "States": {
5          "Check Data": {
6              "Type": "Parallel",
7              "Branches": [
8                  {
9                      "StartAt": "Check Name",
10                     "States": {
11                         "Check Name": {
12                             "Type": "Task",
13                             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
14                             "Parameters": {
15                                 "command": "CHECK_NAME",
16                                 "data": {
17                                     "name.$": "$.application.name"
18                                 }
19                             },
20                             "End": true
21                         }
22                     }
23                 },
24                 {
25                     "StartAt": "Check Address",
26                     "States": {
27                         "Check Address": {
28                             "Type": "Task",
29                             "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
30                             "Parameters": {
31                                 "command": "CHECK_ADDRESS",
32                                 "data": {
33                                     "name.$": "$.application.name"
34                                 }
35                             },
36                             "End": true
37                         }
38                     }
39                 }
40             ]
41         }
42     }
43 }
```



StartAt state is set to “Check Data” which is of type parallel.

Feedback

English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

AWS Services Resource Groups

buildingmodernapps @ 3022-1... N. Virginia Support

Definition

```

1 "Comment": "Parallel example",
2 "StartAt": "Check Data",
3 "States": {
4     "Check Data": {
5         "Type": "Parallel",
6         "Branches": [
7             {
8                 "StartAt": "Check Name",
9                 "States": {
10                     "Check Name": {
11                         "Type": "Task",
12                         "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
13                         "Parameters": {
14                             "command": "CHECK_NAME",
15                             "data": {
16                                 "name.$": "$.application.name"
17                             }
18                         },
19                         "End": true
20                     }
21                 }
22             },
23             {
24                 "StartAt": "Check Address",
25             }
26         ]
27     }
28 }

```

Export Layout

```

graph TD
    Start((Start)) --> CheckName[Check Name]
    Start --> CheckAddress[Check Address]
    CheckName --> Decision{Is it valid?}
    CheckAddress --> Decision
    Decision -- Failed check --> End((End))
    Decision -- Success --> Start

```

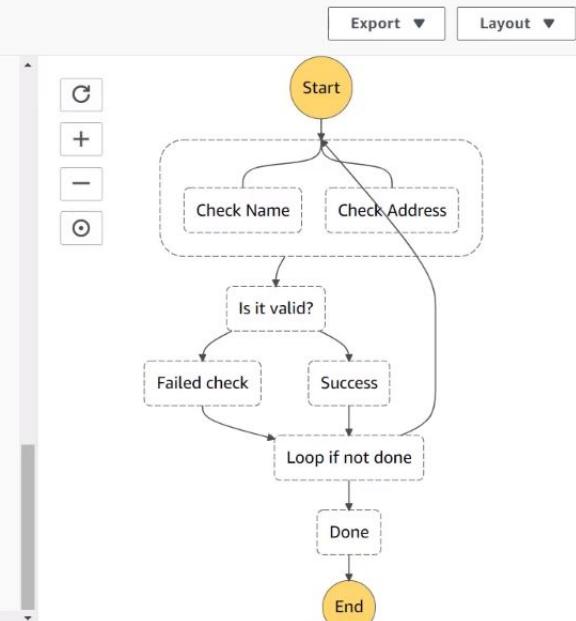
This has branches which takes an array of state machines.



Definition

```

60 "Failed check": {
61     "Type": "Pass",
62     "Next": "Loop if not done"
63 },
64 "Success": {
65     "Type": "Pass",
66     "Next": "Loop if not done"
67 },
68 "Loop if not done": {
69     "Type": "Choice",
70     "Choices": [
71         {
72             "Variable": "$.iterator.continue",
73             "BooleanEquals": true,
74             "Next": "Check Data"
75         }
76     ],
77     "Default": "Done"
78 },
79 "Done": {
80     "Type": "Pass",
81     "End": true
82 }
83 }
84 }
```

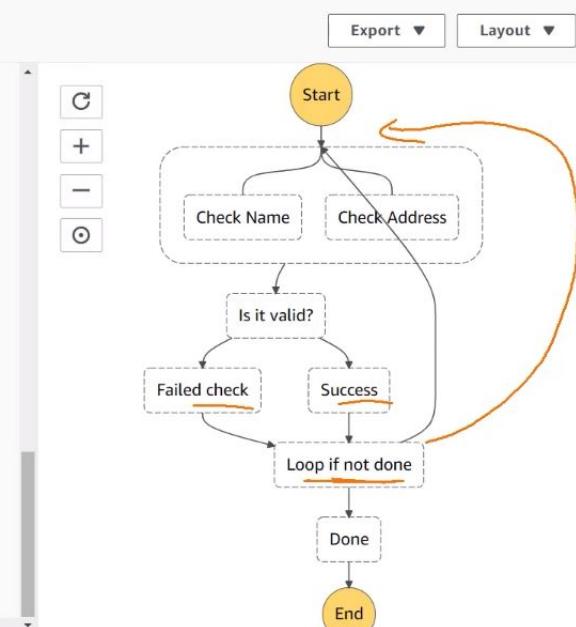


we can add a loop to the states



Definition

```
60 Failed check": {  
61     "Type": "Pass",  
62     "Next": "Loop if not done"  
63 },  
64 "Success": {  
65     "Type": "Pass",  
66     "Next": "Loop if not done"  
67 },  
68 "Loop if not done": {  
69     "Type": "Choice",  
70     "Choices": [  
71         {  
72             "Variable": "$.iterator.continue",  
73             "BooleanEquals": true,  
74             "Next": "Check Data"  
75         }  
76     ],  
77     "Default": "Done"  
78 },  
79 "Done": {  
80     "Type": "Pass",  
81     "End": true  
82 }  
83 }  
84 }
```



note that we created loop using a choice type. It's not a new step type.

Wait State: This state inserts a delay or pause into your workflow. It can wait for a specified duration or until a specified timestamp.



Services

Resource Groups



buildingmodernapps @ 3022-1...

N. Virginia

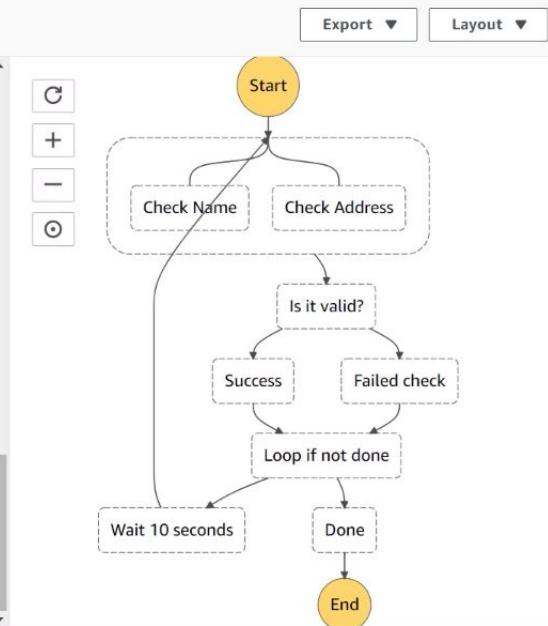
Support

Guest



Definition

```
65      "Type": "Pass",
66      "Next": "Loop if not done"
67 },
68 +
69 "Loop if not done": {
70     "Type": "Choice",
71     "Choices": [
72         {
73             "Variable": "$.iterator.continue",
74             "BooleanEquals": true,
75             "Next": "Wait 10 seconds"
76         }
77     ],
78     "Default": "Done"
79 },
80 "Wait 10 seconds": {
81     "Type": "Wait",
82     "Seconds": 10,
83     "Next": "Check Data"
84 },
85 "Done": {
86     "Type": "Pass",
87     "End": true
88 }
89 }
```



Feedback

English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Privacy Policy

Terms of Use

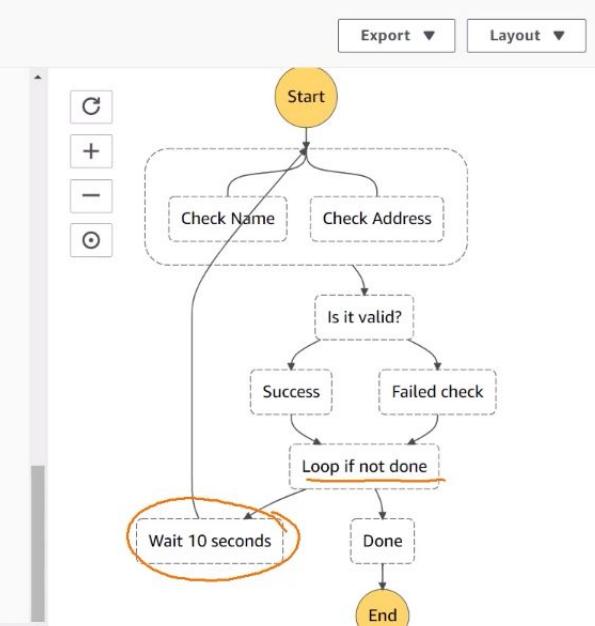
wait state

Definition

```

65      "Type": "Pass",
66      "Next": "Loop if not done"
67    },
68  },
69  "Loop if not done": {
70    "Type": "Choice",
71    "Choices": [
72      {
73        "Variable": "$.iterator.continue",
74        "BooleanEquals": true,
75        "Next": "Wait 10 seconds"
76      }
77    ],
78    "Default": "Done"
79  },
80  "Wait 10 seconds": {
81    "Type": "Wait",
82    "Seconds": 10,
83    "Next": "Check Data"
84  },
85  "Done": {
86    "Type": "Pass",
87    "End": true
88  }
89 }

```



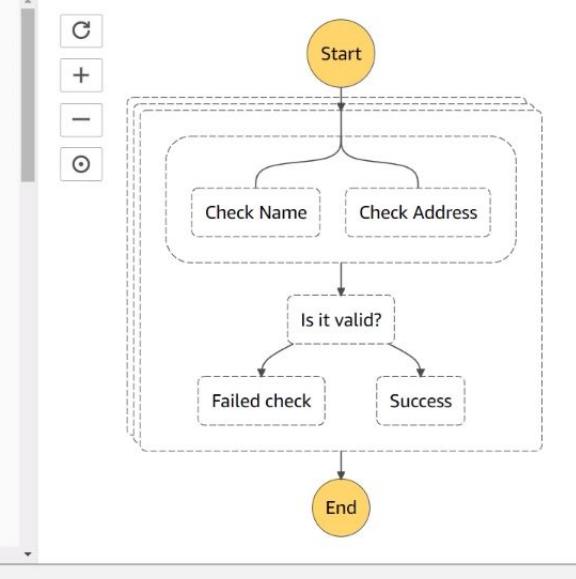
wait for 10 sec and the go to the “Check Data” state

Map State: This state enables you to iterate over a list of items and perform the same set of operations on each item.



Definition

```
1 < "Comment": "Map example",
2 "StartAt": "Map",
3 "States": {
4 < "Map": {
5   "Type": "Map",
6   "MaxConcurrency": 10,
7   "Iterator": {
8     "StartAt": "Check Data",
9     "States": {
10       "Check Data": {
11         "Type": "Parallel",
12         "Branches": [
13           {
14             "StartAt": "Check Name",
15             "States": {
16               "Check Name": {
17                 "Type": "Task",
18                 "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress",
19                 "Parameters": {
20                   "command": "CHECK_NAME",
21                   "data": {
22                     "name.$": "$.application.name"
23                   }
24                 },
25               }
26             }
27           }
28         ]
29       }
30     }
31   }
32 }
```

[Export](#)[Layout](#)



buildingmodernapps

@ 3022-1...

N. Virginia

Support

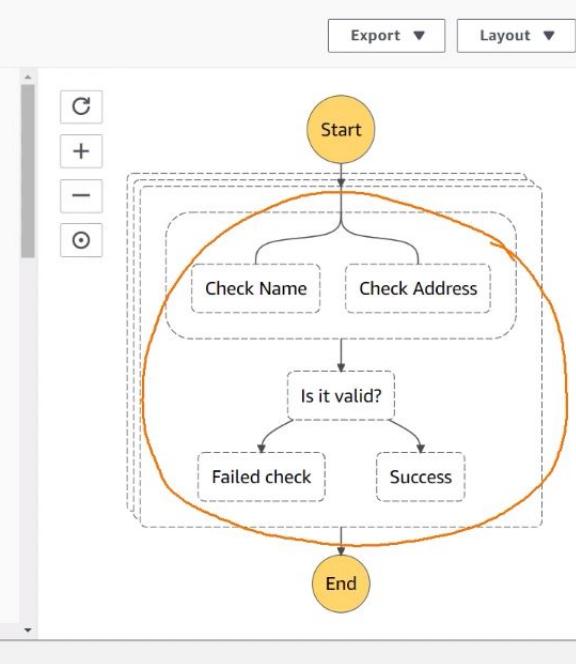


?

Definition

```

1 "Comment": "Map example",
2 "StartAt": "Map", ←
3 "States": { ←
4     "Map": { ←
5         "Type": "Map", ←
6         "MaxConcurrency": 10, ←
7         "Iterator": { ←
8             "StartAt": "Check Data", ←
9             "States": { ←
10                "Check Data": { ←
11                    "Type": "Parallel", ←
12                    "Branches": [ ←
13                        { ←
14                            "StartAt": "Check Name", ←
15                            "States": { ←
16                                "Check Name": { ←
17                                    "Type": "Task", ←
18                                    "Resource": "arn:aws:lambda:us-east-1:302211264422:function:checkAddress", ←
19                                    "parameters": { ←
20                                        "command": "CHECK_NAME", ←
21                                        "data": { ←
22                                            "name.$": "$.application.name" ←
23                                        } ←
24                                    }, ←
25                                } ←
26                            } ←
27                        } ←
28                    } ←
29                } ←
30            } ←
31        } ←
32    } ←
33 }
```



the same state as before into a iterator

[Feedback](#)

[English \(US\)](#)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved.

[Privacy Policy](#)

[Terms of Use](#)

The last two state types are for stopping the execution of a state machine.

Succeed and **Fail**

Succeed State: This state immediately stops the execution of the state machine and marks it as successful.

Fail State: This state immediately stops the execution of the state machine and marks it as failed.

Eight state types of state function:

- **Task**: Do some work in your state machine
- **Choice**: Make a choice between branches of execution
- **Parallel**: Begin parallel branches of execution
- **Map**: Dynamically iterate steps
- **Pass**: Simply pass its input to its output or inject some fixed data
- **Wait**: Provide a delay for a certain amount of time or until a specified time/date
- **success** and **failure**: Stop an execution with a failure or success

Step Functions for Dragon App

States are the individual elements of a state machine.

Each state can make decisions based on its input,
perform actions and pass output to other states.

Tasks (workers) represent one unit of work done through activities or AWS services.



Services

Resource Groups



buildingmodernapps @ 3022-1... N. Virginia

Step Functions

Step Functions > State machines

State machines

Activities

Feature spotlight

Join our feedback panel

State machines (8)



View details

Edit

Copy to new

Delete

Create state mac

Search for state machines

Any type

< 1 >

	Name	Type	Creation date	Status	Logs	Running	Succeeded	Failed	Timed out	A
	task-simple	Standard	Jul 3, 2020 12:47:04.967 AM	Active	-	0	0	0	0	0
	wait	Standard	Jul 2, 2020 10:53:44.247 PM	Active	-	0	0	0	0	0
	loop	Standard	Jul 2, 2020 10:39:34.561 PM	Active	-	0	0	0	0	0
	map	Standard	Jul 2, 2020 10:24:47.591 PM	Active	-	0	0	0	0	0
	choice	Standard	Jul 2, 2020 08:57:35.650 PM	Active	-	0	0	0	0	0
	parallel	Standard	Jul 2, 2020 08:47:54.341 PM	Active	-	0	0	0	0	0
	task	Standard	Jul 2, 2020 08:38:02.861 PM	Active	-	0	0	0	0	0
	MyStateMachine	Standard	Jun 30, 2020 06:38:24.778 PM	Active	/aws/states/My...	0	1	0	0	0

Step Function console with some state machines

Step 1
Define state machineStep 2
Specify details

Define state machine

Author with code snippets

Author your workflow using Amazon States Language. You can generate code snippets to easily build out your workflow steps.

Run a sample project

Deploy and run a fully functioning sample project in minutes using CloudFormation.

Start with a template

Get started quickly with common patterns for Amazon States Language.

Type

 Standard

Durable, checkpointed workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

▶ Help me decide

 Express New

Event-driven workflows for streaming data processing, microservices orchestration, IoT data ingestion, mobile backends, and other short duration, high-event-rate workloads.

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

Export ▾

Layout

Generate code snippet

Format JSON

```
1 v  {
2   "Comment": "A Hello World example of the Amazon States Language using Pass states",
3   "StartAt": "Hello",
4   "States": {
5     "Hello": {
6       "Type": "Pass",
7       "Result": "Hello",
8       "Next": "World"
9     }
10  }
```



Start

Create a new state machine

▶ Help me decide

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

Export ▾

Layout

Generate code snippet

Format JSON

```
1 v  {
2   "Comment": "A Hello World example of the Amazon States Language using Pass states",
3   "StartAt": "Hello",
4   "States": {
5     "Hello": {
6       "Type": "Pass",
7       "Result": "Hello",
8       "Next": "World"
9     },
10    "World": {
11      "Type": "Pass",
12      "Result": "World",
13      "End": true
14    }
15  }
16 }
```



Cancel

This is the default state machine using Amazon State Language in JSON

▶ Help me decide

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

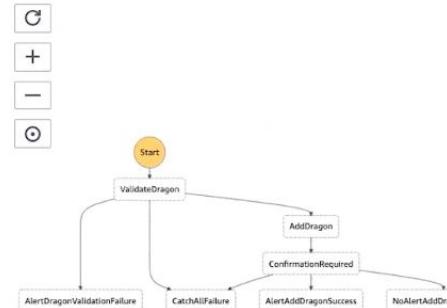
Export ▾

Layout

Generate code snippet

Format JSON

```
52 },
53 {
54     "Variable": "$.confirmationRequired",
55     "BooleanEquals": false,
56     "Next": "NoAlertAddDragonSuccess"
57 }
58 ],
59 "Default": "CatchAllFailure"
60 },
61 "AlertAddDragonSuccess": {
62     "Type": "Task",
63     "Resource": "arn:aws:states:::sns:publish",
64     "Parameters": {
65         "Message": "The dragon you reported has been added!",
66         "PhoneNumber.$": "$.reportingPhoneNumber"
67     },
68     "End": true
69 },
70 "NoAlertAddDragonSuccess": {
71     "Type": "Succeed"
72 }
73 }
74 }
```



Cancel

Type

Standard

Durable, checkpointed workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

Express New

Event-driven workflows for streaming data processing, microservices orchestration, IoT data ingestion, mobile backends, and other short duration, high-event-rate workloads.

▶ Help me decide

Definition

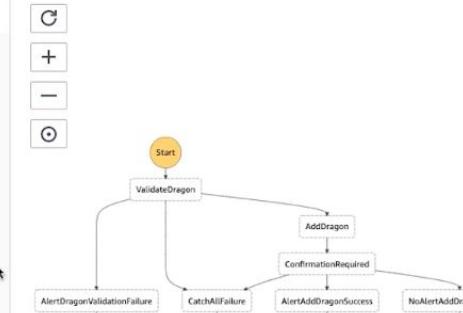
Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

Export ▾

Layout

Generate code snippet ▾ Format JSON

```
1 <?xml version="1.0" encoding="UTF-8"?>
2 <WorkflowDefinition>
3   <StartState>Start</StartState>
4   <States>
5     <State>
6       <Name>ValidateDragon</Name>
7       <Type>Task</Type>
8       <Resource>arn:aws:lambda:us-east-1:302211264422:function:validate-dragon</Resource>
9       <Catch>
10      <ErrorEquals>
11        <ErrorType>DragonValidationException</ErrorType>
12      </ErrorEquals>
13      <Next>AlertDragonValidationFailure</Next>
14      <ResultPath>null</ResultPath>
15    </Catch>
16    <Catch>
17      <ErrorEquals>
18        <ErrorType>States.ALL</ErrorType>
19      </ErrorEquals>
20      <Next>CatchAllFailure</Next>
21    </Catch>
22    <EndState>End</EndState>
23  </States>
24</WorkflowDefinition>
25
26 "AlertDragonValidationFailure": {
```



the first step is the ValidateDragon state

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

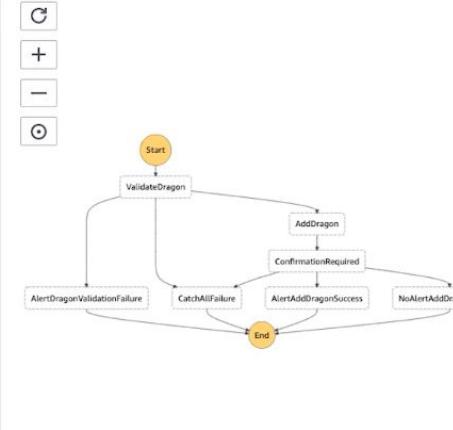
Export ▾

Layout

Generate code snippet

Format JSON

```
25
26 "AlertDragonValidationFailure": {
27     "Type": "Task",
28     "Resource": "arn:aws:states:::sns:publish",
29     "Parameters": {
30         "Message": "The dragon you reported failed validation and was not added",
31         "PhoneNumber.$": "$.reportingPhoneNumber"
32     },
33     "End": true
34 }
35 "CatchAllFailure": {
36     "Type": "Fail",
37     "Cause": "Something unknown went wrong"
38 },
39 "AddDragon": {
40     "Type": "Task",
41     "Resource": "arn:aws:lambda:us-east-1:302211264422:function:add-dragon",
42     "Next": "ConfirmationRequired",
43     "ResultPath": null
44 },
45 "ConfirmationRequired": {
46     "Type": "Choice",
47     "Choices": [
```



Feedback English (US)

© 2018 - 2020 Amazon Web Services Inc. or its affiliates. All rights reserved.

Privacy

Alert a user that their dragon reporting failed.

This is an example of integrating with **Amazon Simple Notification Service** or SNS.

By including the \$ after the parameter name and then using the \$. syntax, it is able to pull the phone number off of the payload being passed into the state machine execution.

This is an example of how to pull info from the path.

In Amazon State Language, a "Path" is used to specify a **JSONPath expression** that selects a specific piece of data from the input.

This selected data can then be used within the state.

```
"Parameters": {  
    "InputData.$": "$.input", // This is the Path expression  
    "OutputData.$": "$.input.property" // Another example of using Path}
```

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

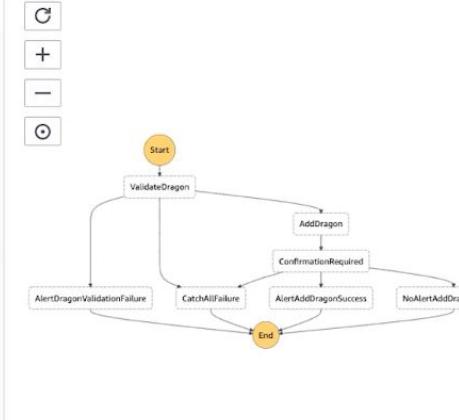
Export ▾

Layout

Generate code snippet ▾

Format JSON

```
25
26 "AlertDragonValidationFailure": {
27     "Type": "Task",
28     "Resource": "arn:aws:states:::sns:publish",
29     "Parameters": {
30         "Message": "The dragon you reported failed validation and was not added",
31         "PhoneNumber.$": "$.reportingPhoneNumber"
32     },
33     "End": true
34 },
35 "CatchAllFailure": {
36     "Type": "Fail",
37     "Cause": "Something unknown went wrong"
38 },
39 "AddDragon": {
40     "Type": "Task",
41     "Resource": "arn:aws:lambda:us-east-1:302211264422:function:add-dragon",
42     "Next": "ConfirmationRequired",
43     "ResultPath": null
44 },
45 "ConfirmationRequired": {
46     "Type": "Choice",
47     "Choices": [
```



if the alert dragon validation failure is executed, the end is set to be true so that nothing else will execute and the state machine execution ends.

Type

Standard

Durable, checkpointed workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

▶ Help me decide

Express New

Event-driven workflows for streaming data processing, microservices orchestration, IoT data ingestion, mobile backends, and other short duration, high-event-rate workloads.

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

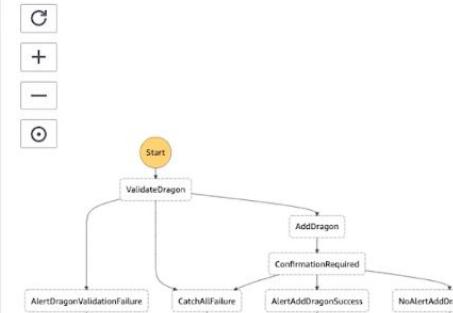
Export ▾

Layout

Generate code snippet

Format JSON

```
35 "CatchAllFailure": {  
36     "Type": "Fail",  
37     "Cause": "Something unknown went wrong"  
38 },  
39 "AddDragon": {  
40     "Type": "Task",  
41     "Resource": "arn:aws:lambda:us-east-1:302211264422:function:add-dragon",  
42     "Next": "ConfirmationRequired",  
43     "ResultPath": null  
44 },  
45 "ConfirmationRequired": {  
46     "Type": "Choice",  
47     "Choices": [  
48         {  
49             "Variable": "$.confirmationRequired",  
50             "BooleanEquals": true,  
51             "Next": "AlertAddDragonSuccess"  
52         },  
53         {  
54             "Variable": "$.confirmationRequired",  
55             "BooleanEquals": false,  
56             "Next": "NoAlertAddDragonSuccess"  
57         }  
58     ]  
59 }
```



CatchAllFailure is executed if the catch caught an unknown error.

Type

Standard

Durable, checkpointed workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

Express New

Event-driven workflows for streaming data processing, microservices orchestration, IoT data ingestion, mobile backends, and other short duration, high-event-rate workloads.

▶ Help me decide

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

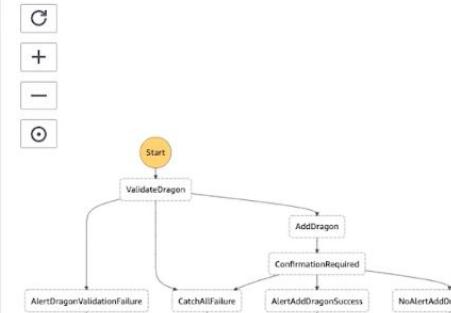
Export ▾

Layout

Generate code snippet

Format JSON

```
--  
39 "AddDragon": {  
40     "Type": "Task",  
41     "Resource": "arn:aws:lambda:us-east-1:302211264422:function:add-dragon",  
42     "Next": "ConfirmationRequired",  
43     "ResultPath": null  
44 },  
45 ConfirmationRequired : {  
46     "Type": "Choice",  
47     "Choices": [  
48         {  
49             "Variable": "$.confirmationRequired",  
50             "BooleanEquals": true,  
51             "Next": "AlertAddDragonSuccess"  
52         },  
53         {  
54             "Variable": "$.confirmationRequired",  
55             "BooleanEquals": false,  
56             "Next": "NoAlertAddDragonSuccess"  
57         }  
58     ],  
59     "Default": "CatchAllFailure"  
60 },  
61 "AlertAddDragonSuccess": {
```



AddDragon state will invoke the associated lambda function

Type

Standard

Durable, checkpointed workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

▶ Help me decide

Express New

Event-driven workflows for streaming data processing, microservices orchestration, IoT data ingestion, mobile backends, and other short duration, high-event-rate workloads.

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

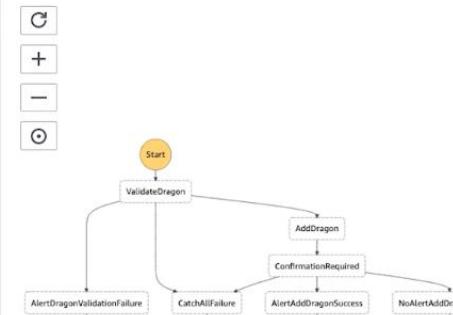
Export ▾

Layout

Generate code snippet

Format JSON

```
--  
39 "AddDragon": {  
40   "Type": "Task",  
41   "Resource": "arn:aws:lambda:us-east-1:302211264422:function:add-dragon",  
42   "Next": "ConfirmationRequired",  
43   "ResultPath": null  
44 },  
45 "ConfirmationRequired": {  
46   "Type": "Choice",  
47   "Choices": [  
48     {  
49       "Variable": "$.confirmationRequired",  
50       "BooleanEquals": true,  
51       "Next": "AlertAddDragonSuccess"  
52     },  
53     {  
54       "Variable": "$.confirmationRequired",  
55       "BooleanEquals": false,  
56       "Next": "NoAlertAddDragonSuccess"  
57     }  
58   ],  
59   "Default": "CatchAllFailure"  
60 },  
61 "AlertAddDragonSuccess": {
```



ConfirmationRequired state is a choice state.

Type

Standard

Durable, checkpointed workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

▶ Help me decide

Express New

Event-driven workflows for streaming data processing, microservices orchestration, IoT data ingestion, mobile backends, and other short duration, high-event-rate workloads.

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

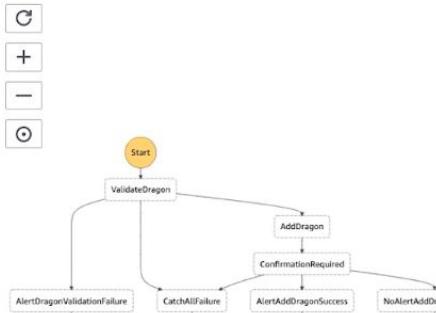
Export ▾

Layout

Generate code snippet

Format JSON

```
52     },
53     {
54         "Variable": "$.confirmationRequired",
55         "BooleanEquals": false,
56         "Next": "NoAlertAddDragonSuccess"
57     }
58 ],
59 "Default": "CatchAllFailure"
60 ,
61 "AlertAddDragonSuccess": {
62     "Type": "Task",
63     "Resource": "arn:aws:states:::sns:publish",
64     "Parameters": {
65         "Message": "The dragon you reported has been added!",
66         "PhoneNumber.$": "$.reportingPhoneNumber"
67     },
68     "End": true
69 },
70 "NoAlertAddDragonSuccess": {
71     "Type": "Succeed"
72 }
73 }
74 }
```



next is the alert add dragon success which is another integration with SNS.

Type

Standard

Durable, checkpointed workflows for machine learning, order fulfillment, IT/DevOps automation, ETL jobs, and other long-duration workloads.

▶ Help me decide

Express New

Event-driven workflows for streaming data processing, microservices orchestration, IoT data ingestion, mobile backends, and other short duration, high-event-rate workloads.

Definition

Define your workflow using [Amazon States Language](#). Refresh the graph to render the definition.

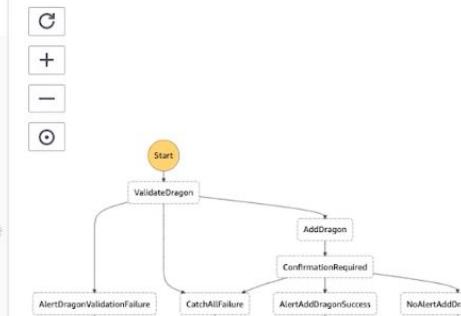
Export ▾

Layout

Generate code snippet

Format JSON

```
52     },
53     {
54         "Variable": "$.confirmationRequired",
55         "BooleanEquals": false,
56         "Next": "NoAlertAddDragonSuccess"
57     }
58 ],
59 "Default": "CatchAllFailure"
60 },
61 "AlertAddDragonSuccess": {
62     "Type": "Task",
63     "Resource": "arn:aws:states:::sns:publish",
64     "Parameters": {
65         "Message": "The dragon you reported has been added!",
66         "PhoneNumber.$": "$.reportingPhoneNumber"
67     },
68     "End": true
69 },
70 "NoAlertAddDragonSuccess": {
71     "Type": "Succeed"
72 }
73 }
74 }
```



Finally, there's a NoAlertAddDragonSuccess which is a succeed state type.
It marks the execution as a success.

Now lets create an execution

```
{  
    "dragon_name_str": "Peter",  
    "description_str": "New dragon",  
    "family_str": "green",  
    "location_city_str": "Houston",  
    "location_country_str": "USA",  
    "location_state_str": "TX",  
    "location_neighborhood_str": "Downtown",  
    "reportingPhoneNumber": "15555555555",  
    "confirmationRequired": false  
}
```

This is the payload that we will use to create executions.



Services

Resource Groups



buildingmodernapps @ 3022-1...

N. Virginia

Step Functions



State machine successfully created

State machines

Activities

Feature spotlight

Join our feedback panel

Step Functions > State machines > DragonsStateMachine

DragonsStateMachine

Edit

Start execution

Delete

Ac

Details

ARN

arn:aws:states:us-east-1:302211264422:stateMachine:DragonsStateMachine

Type

Standard

IAM role ARN

arn:aws:iam::302211264422:role/service-role/StepFunctions-DragonsStateMachine-role-c863aad7



Creation date

Jul 29, 2020 04:16:31.575 PM

Executions

Logging

Definition

Tags

Executions (0)



View details

Stop execution

Start exec

Search for executions

Filter by status

< 1 >

Name



Status

Started



End Time

No executions

Start execution

click on Start Execution

Step Functions

State machine successfully created

State machines

Step Functions ▾ State machines ▾ DragonsStatsMachine

Activities

New executionStart an execution using the latest definition of the state machine. [Learn more](#)

Feature sp...

Join our fe...

Enter an execution name - optional

Enter your execution id here

Peter

Input - optional

Enter input values for this execution in JSON format

```
1 + {  
2     "dragon_name_str": "Peter",  
3     "description_str": "New dragon",  
4     "family_str": "green",  
5     "location_city_str": "Houston",  
6     "location_country_str": "USA",  
7     "location_state_str": "TX",  
8     "location_neighborhood_str": "Downtown",  
9     "reportingPhoneNumber": "15555555555",  
10    "confirmationRequired": false  
11 }
```

 Open in a new browser tab

Cancel

Start execution

name it and paste the payload



Services

Resource Groups



buildingmodernapps @ 3022-1...

N. Virginia

Step Functions



State machines

Activities

Feature spotlight

Join our feedback panel

Execution details

Execution Status



Succeeded

Started

Jul 29, 2020 04:18:20.075 PM

Execution ARN

arn:aws:states:us-east-1:302211264422:execution:DragonsStateMachine:Peter

End Time

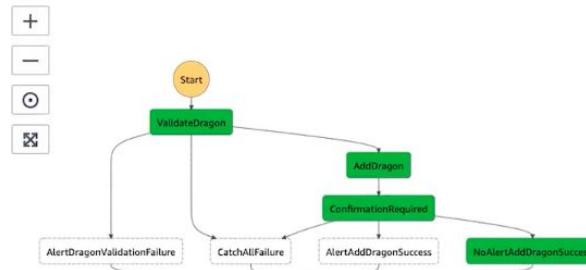
Jul 29, 2020 04:18:22.275 PM

▶ Input

▶ Output

Visual workflow

Export ▾



Code

Step details

Select a step to view its details.

This one was success and looking through, we can see the process path that was taken through the state machine.



Services

Resource Groups



buildingmodernapps @ 3022-1...

N. Virginia

Step Functions

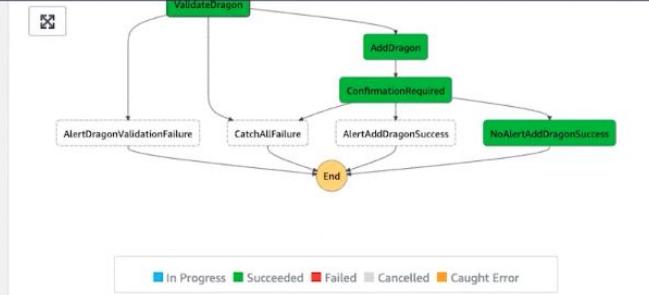


State machines

Activities

Feature spotlight

Join our feedback panel



Succeeded

Resource

arn:aws:lambda:us-east-1:302211264422:function:validate-dragon | CloudWatch logs |

Input

```
{  
    "dragon_name_str": "Peter",  
    "description_str": "New dragon",  
    "family_str": "green",  
    "location_city_str": "Houston",  
    "location_country_str": "USA",  
    "location_state_str": "TX",  
    "location_neighborhood_str": "Downtown",  
    "reportingPhoneNumber": "15555555555",  
    "confirmationRequired": false  
}
```

Output

```
{  
    "dragon_name_str": "Peter",  
    "description_str": "New dragon",  
    "family_str": "green",  
    "location_city_str": "Houston",  
    "location_country_str": "USA",  
    "location_state_str": "TX",  
    "location_neighborhood_str": "Downtown",  
    "reportingPhoneNumber": "15555555555",  
    "confirmationRequired": false  
}
```

Exception

we can see the process path as well as the input and output of each of the states.



Services

Resource Groups



buildingmoderna

Step Functions X

State machines

Activities

Feature spotlight

Join our feedback panel

Execution event history

ID	Type	Step	Resource	Elapsed Time (ms)
▶ 1	ExecutionStarted		-	0
▶ 2	TaskStateEntered	ValidateDragon	-	38
▶ 3	LambdaFunctionScheduled	ValidateDragon	Lambda CloudWatch logs	38
▶ 4	LambdaFunctionStarted	ValidateDragon	Lambda CloudWatch logs	70
▶ 5	LambdaFunctionSucceeded	ValidateDragon	Lambda CloudWatch logs	1043
▶ 6	TaskStateExited	ValidateDragon	-	1043
▶ 7	TaskStateEntered	AddDragon	-	1051
▶ 8	LambdaFunctionScheduled	AddDragon	Lambda CloudWatch logs	1051
▶ 9	LambdaFunctionStarted	AddDragon	Lambda CloudWatch logs	1060
▶ 10	LambdaFunctionSucceeded	AddDragon	Lambda CloudWatch logs	2093
▶ 11	TaskStateExited	AddDragon	-	2093
▶ 12	ChoiceStateEntered	ConfirmationRequired	-	2100
▶ 13	ChoiceStateExited	ConfirmationRequired	-	2100
▶ 14	SucceedStateEntered	NoAlertAddDragonSuccess	-	2200
▶ 15	SucceedStateExited	NoAlertAddDragonSuccess	-	2200
▶ 16	ExecutionSucceeded		-	2200

All events loaded

Scrolling down, you can also see the event history

Now let's a different path.

Step Functions



PeterAgain

[Edit state machine](#)[New ex](#)

State machines

Activities

New executionStart an execution using the latest definition of the state machine. [Learn more](#)

Feature sp

Join our fe

Enter an execution name - optional

Enter your execution id here

Atlas

Input - optional

Enter input values for this execution in JSON format

```
1 * {  
2     "dragon_name_str": "Atlas",  
3     "description_str": "New dragon",  
4     "family_str": "green",  
5     "location_city_str": "Houston",  
6     "location_country_str": "USA",  
7     "location_state_str": "TX",  
8     "location_neighborhood_str": "Downtown",  
9     "reportingPhoneNumber": "15555555555",  
10    "confirmationRequired": false  
11 }
```

 Open in a new browser tab

Cancel

Start execution

Add Atlas (already exist)



Services

Resource Groups



buildingmodernapps @ 3022-1... N. Virginia

Step Functions



State machines

Activities

Feature spotlight

Join our feedback panel

Execution details

Execution Status

Succeeded

Execution ARN

arn:aws:states:us-east-1:302211264422:execution:DragonsStateMachine:Atlas

▶ Input

Started

Jul 29, 2020 04:21:47.430 PM

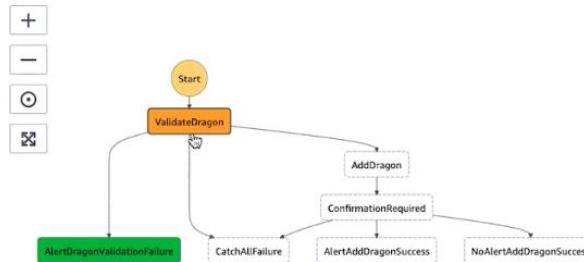
End Time

Jul 29, 2020 04:21:47.827 PM

▶ Output

Visual workflow

Export ▾

■ In Progress ■ Succeeded ■ Failed ■ Cancelled ■ Caught ErrorCode Step details

Name

Type

ValidateDragon

Task

Status

⚠ Caught Error

Resource

arn:aws:lambda:us-east-1:302211264422:function:validate-dragon | CloudWatch logs

▶ Input

▶ Output

▶ Exception

ValidateDragon step encountered an error, which changed the flow that was taken.

Good Resources:

[Getting Started with AWS Step Functions](#)

[InputPath, ResultPath, and OutputPath Examples - AWS Step Functions](#)

[Sample projects for Step Functions](#)

AWS Step Function Service Integrations

recap:

Let's say you have an input JSON data like this:

```
{ "input": { "name": "John Doe", "address": "123 Main Street" } }
```

And you want to pass the name and address as parameters to a Lambda function.

```
{
  "StartAt": "MyState",
  "States": [
    { "MyState": {
        "Type": "Task",
        "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:FUNCTION_NAME",
        "Parameters": {
          "Name.$": "$.input.name",
          "Address.$": "$.input.address"
        },
        "End": true
      }
    }
  ]
}
```

and you invoke an AWS Step Function with input data:

```
aws stepfunctions start-execution --state-machine-arn arn:aws:states:REGION:ACCOUNT_ID:stateMachine:STATE_MACHINE_NAME
--input '{"name": "John Doe", "address": "123 Main Street"}'
```

One feature of Step Functions is its ability to integrate with other AWS services without needing to write a Lambda function or a remote program to call AWS APIs.

In order to integrate with another AWS service from Step Functions,
all you need is a task state.

One common example is you can invoke a Lambda function from a state.

```
"Invoke Lambda function": {  
    "Type": "Task",  
    "Resource": "arn:aws:states:::lambda:invoke",  
    "Parameters": {  
        "FunctionName":  
            "arn:aws:lambda:<REGION>:<ACCTNUMBER>:function:<FUNCTIONNAME>",  
        "Payload": {  
            "Input.$": "$"  
        }  
    },  
    "Next": "NEXT_STATE"  
}
```

Here's a snippet showing a state that invokes a Lambda function.

```
"Invoke Lambda State": {  
    "Type": "Task",  
    "Resource": "arn:aws:lambda:<REGION>:<ACCTNUMBER>:function:<FUNCTIONNAME>"  
    "Next": "NEXT_STATE"  
}
```

This is a bit of a simpler way to invoke a Lambda function from a state machine.

```
"Put item into DynamoDB": {  
    "Type": "Task",  
    "Resource": "arn:aws:states:::dynamodb:putItem",  
    "Parameters": {  
        "TableName": "MyDynamoDBTable",  
        "Item": {  
            "DragonName": {  
                "S": "Atlas"  
            },  
            "Family": {  
                "S": "Red"  
            }  
        }  
    },  
    "Next": "NEXT_STATE"  
}
```

Integration with DynamoDB: a state that inserts an item into a DynamoDB table.
The parameters section is where you specify the parameters the AWS API needs.

```
"Amazon SNS: Publish a message": {  
    "Type": "Task",  
    "Resource": "arn:aws:states:::sns:publish",  
    "Parameters": {  
        "Message": "Hello from Step Functions!",  
        "PhoneNumber": "+15555555555"  
    },  
    "Next": "NEXT_STATE"  
}
```

integration with SNS

Step Functions integrates with a lot of other AWS services as well:

- Running an AWS batch job
- Running a task on Amazon elastic container service
- Sending a message to Amazon simple queue service
- Managing a job for AWS Glue or Amazon SageMaker
- Building workflows for executing an Amazon EMR job
- Running another step function workflow

API Gateway and Step Function Integration Demo

Now that the state machine for reporting a new Dragon is created,
we want to configure API Gateway to start a new execution
every time a POST request comes into the dragon API.

We use the **API Gateway Integration Request Mapping**
to change the format of the incoming data to match the format that Step Functions needs.

```
{  
    "dragon_name_str": "Peter",  
    "description_str": "New dragon",  
    "family_str": "green",  
    "location_city_str": "Houston",  
    "location_country_str": "USA",  
    "location_state_str": "TX",  
    "location_neighborhood_str": "Downtown",  
    "reportingPhoneNumber": "15555555555",  
    "confirmationRequired": false  
}
```

This is a valid new dragon data. This JSON payload should pass request validation because it matches the Dragon model.

Step Functions Management X API Gateway

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/dhajpep0vi

aws Services Resource Groups

Amazon API Gateway APIs > dragons (it9pqb5zm7) > Resources > / (dhajpep0vi)

Show all hints ?

APIs Resources Actions / Methods

No methods defined for the resource.

API: dragons

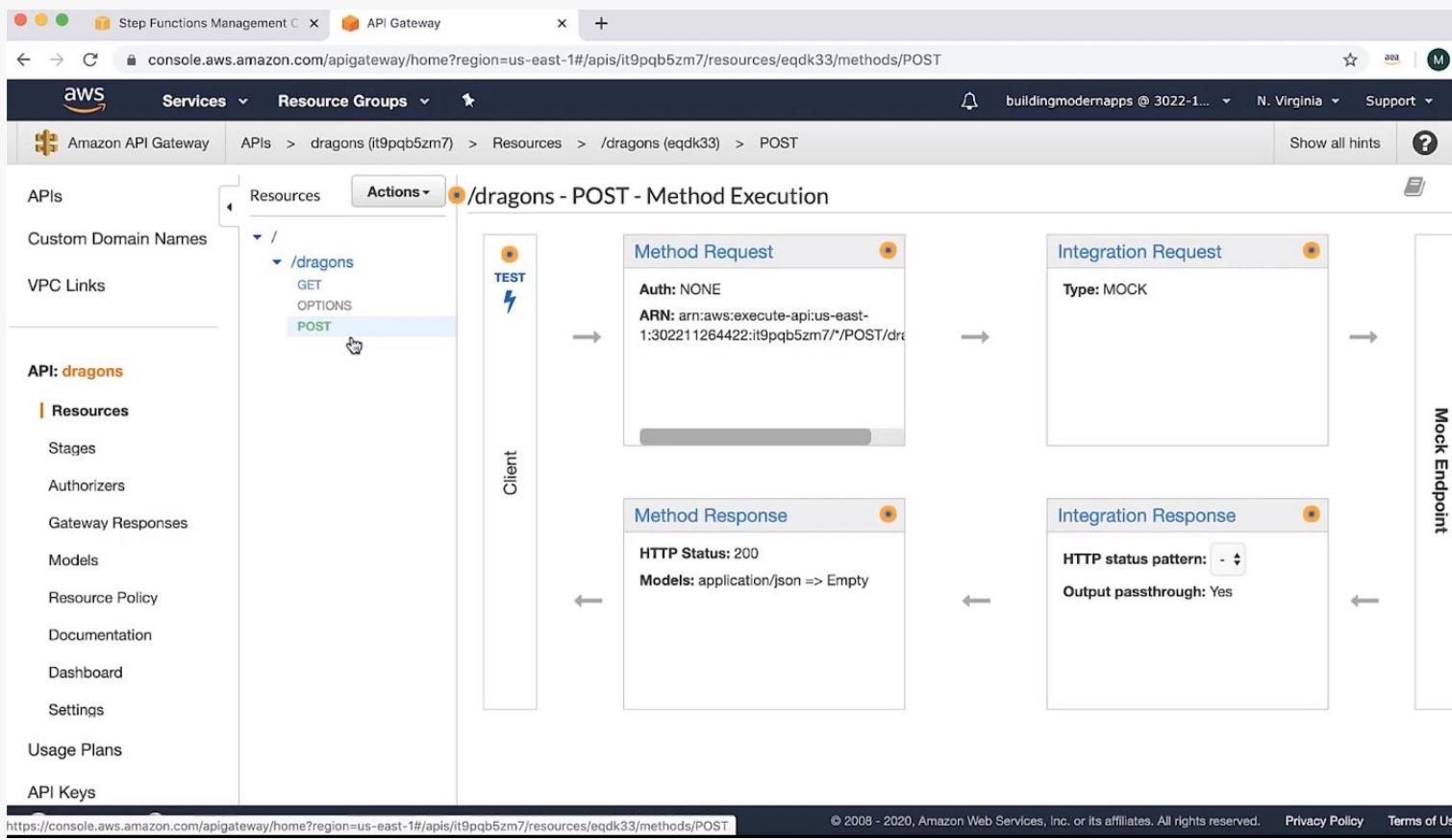
Resources Stages Authorizers Gateway Responses Models Resource Policy Documentation Dashboard Settings Usage Plans API Keys

Feedback English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

The screenshot shows the AWS API Gateway console interface. On the left, a sidebar lists various API management components like Stages, Authorizers, and Resource Policies. The main area displays an API named 'dragons'. Under the '/' resource, three methods are listed: GET, OPTIONS, and POST. A cursor is hovering over the POST method. The top navigation bar includes tabs for Step Functions Management and API Gateway, and shows the URL as console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/dhajpep0vi. The top right corner shows user information and navigation icons.

AWS API Console



Post method. Navigate to the integration request.

Step Functions Management C X API Gateway

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST

aws Services Resource Groups

Amazon API Gateway APIs > dragons (it9pqb5zm7) > Resources > /dragons (eqdk33) > POST Show all hints ?

APIs Resources Actions ▾ Method Execution /dragons - POST - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function 
 HTTP 
 Mock 
 AWS Service 
 VPC Link 

AWS Region

AWS Service

AWS Subdomain

HTTP method

Action Type Use action name
 Use path override

Path override (optional) rest/resource/path

Execution role arn:aws:iam::myAccount:role/myRole 

Feedback English (US)

2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Step Functions Management X API Gateway

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST

aws Services Resource Groups

Amazon API Gateway APIs > dragons (it9pqb5zm7) > Resources > /dragons (eqdk33) > POST Show all hints ?

APIs Resources Actions ▾ Method Execution /dragons - POST - Integration Request

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function i
 HTTP i
 Mock i
 AWS Service i
 VPC Link i

AWS Region us-east-1

AWS Service Step Functions

AWS Subdomain

HTTP method

Action Type Use action name
 Use path override

Path override (optional) rest/resource/path

Execution role arn:aws:iam::myAccount:role/myRole i

Feedback English (US)

2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

The screenshot shows the AWS API Gateway integration configuration for a POST method on the '/dragons' resource. The 'Integration type' is set to 'AWS Service' (Step Functions). The 'AWS Region' is 'us-east-1' and the 'AWS Service' is 'Step Functions'. The 'HTTP method' dropdown is open. The 'Action Type' is set to 'Use path override', and the 'Path override (optional)' field contains 'rest/resource/path'. The 'Execution role' is 'arn:aws:iam::myAccount:role/myRole'. The left sidebar shows the API structure: APIs > dragons (it9pqb5zm7) > Resources > /dragons (eqdk33) > POST.

Screenshot of the AWS API Gateway Step Functions Management interface showing the configuration for a POST method on the /dragons resource.

The left sidebar shows the navigation menu:

- VPC Links
- API: dragons**
- Resources
- Stages
- Authorizers
- Gateway Responses
- Models
- Resource Policy
- Documentation
- Dashboard
- Settings
- Usage Plans
- API Keys
- Client Certificates
- Settings

The main configuration area for the POST method on the /dragons resource:

- Integration type:** AWS Service (selected)
- AWS Region:** us-east-1
- AWS Service:** Step Functions
- AWS Subdomain:** (empty)
- HTTP method:** (dropdown menu)
- Action Type:** Use path override (selected)
- Path override (optional):** rest/resource/path
- Execution role:** arn:aws:iam::302211264422:role/APIGatewayStepFunctions
- Content Handling:** Passthrough
- Use Default Timeout:** checked

At the bottom right is a **Save** button.

This execution role is what API Gateway is going to assume in order to invoke the state machine. That means that we have to already have an IAM Role created in this account that has the necessary permissions that API Gateway will need to execute the state machine or start a new execution.

Step Functions Management X API Gateway

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST

VPC Links

API: dragons

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Dashboard

Settings

Usage Plans

API Keys

Client Certificates

Settings

/dragons

GET

OPTIONS

POST

IAM Role ARN
arn:aws:iam::302211264422:role/APIGatewayStepFunctions

Step Functions ARN
arn:aws:states:us-east-1:302211264422:stateMachine:MyStateMachine

VTL

```
#set($data = $input.path('$'))  
  
#set($input = { "dragon_name_str" : "$data.dragonName", "description_str" :  
    "$data.description", "family_str" : "$data.family", "location_city_str" : "$data.city",  
    "location_country_str" : "$data.country", "location_state_str" : "$data.state",  
    "location_neighborhood_str" : "$data.neighborhood", "reportingPhoneNumber" :  
    "$data.reportingPhoneNumber", "confirmationRequired" : $data.confirmationRequired})
```

Path override (optional) rest/resource/path

Execution role arn:aws:iam::myAccount:role/myRole

Content Handling Passthrough

Use Default Timeout

Save

Mapping Templates

Feedback English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Keep all this information in a file

Step Functions Management X API Gateway

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST

VPC Links

API: dragons

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Dashboard

Settings

Usage Plans

API Keys

Client Certificates

Settings

/dragons

GET
OPTIONS
POST

PROVIDE INFORMATION ABOUT THE TARGET BACKEND THAT THIS METHOD WILL CALL AND WHETHER THE INCOMING REQUEST DATA SHOULD BE MODIFIED.

Integration type Lambda Function i
 HTTP i
 Mock i
 AWS Service i
 VPC Link i

AWS Region us-east-1

AWS Service Step Functions

AWS Subdomain

HTTP method

Action Type Use action name
 Use path override

Action StartExecution

Execution role arn:aws:iam::302211264422:role/APIGatewayStepFunctions i

Content Handling Passthrough i

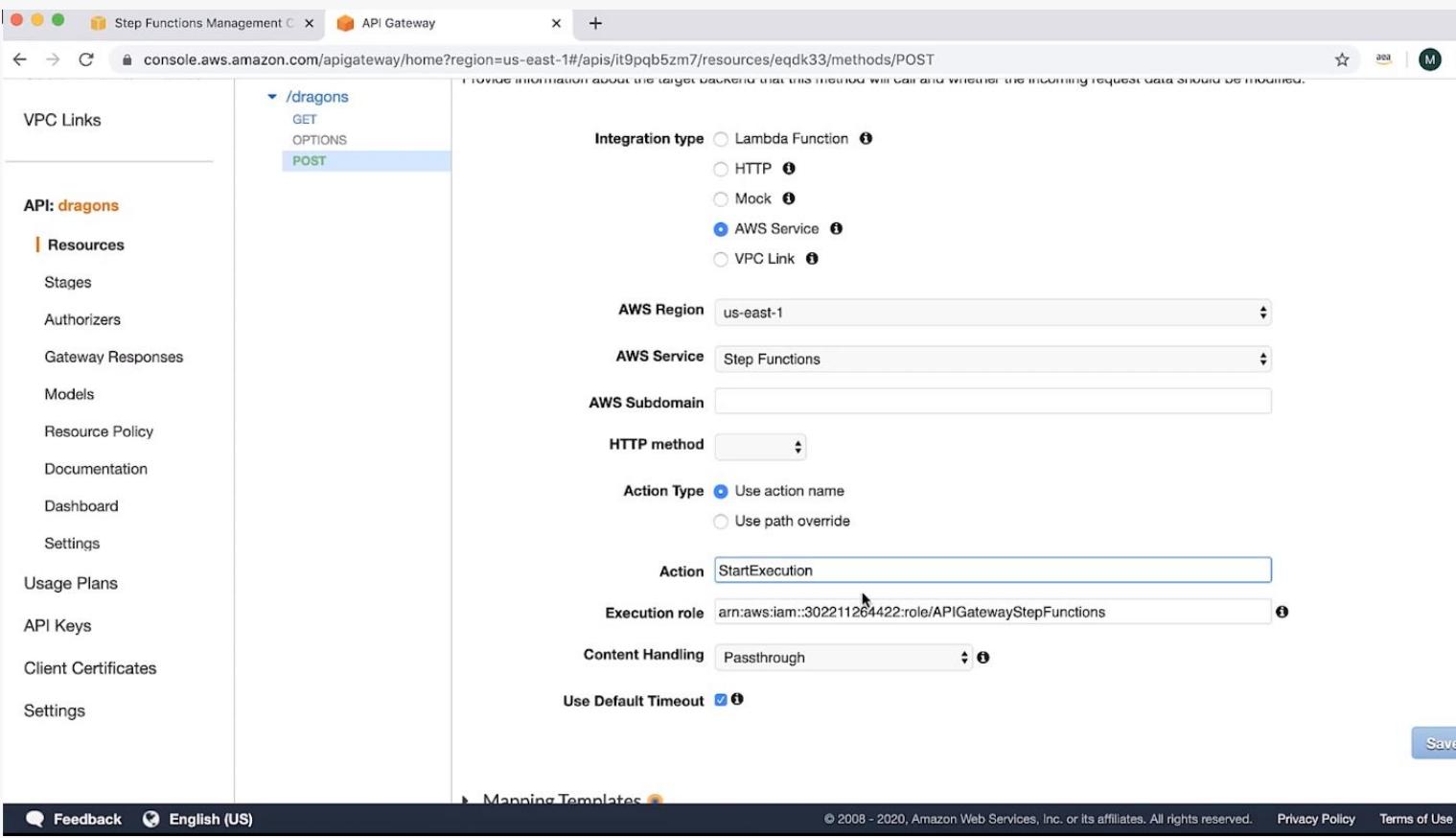
Use Default Timeout i

Save

Mapping Templates i

Feedback English (US)

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use



StartExecution is the AWS API for starting a new execution.

Notice that we didn't insert ARN for the state machine.

So looking at this, how a customer send in a POST request,
and then invoke that specific state machine?

One way is that Client have to include the ARN for that state machine on the request.

But, it's not a good idea to expose that ARN to clients.

Instead, we need to find a way to inject that ARN
into this payload without them having to provide it.

We can take advantage of the mapping templates.

Mapping templates are written in VTL and they allow you to change the payload.

Step Functions Management X API Gateway

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pb5zm7/resources/eqdk33/methods/POST

HTTP method

Action Type Use action name
 Use path override

Action StartExecution

Execution role arn:aws:iam::302211264422:role/APIGatewayStepFunctions 

Content Handling Passthrough 

Use Default Timeout 

Save

▼ Mapping Templates 

Request body passthrough When no template matches the request Content-Type header  
 When there are no templates defined (recommended) 
 Never 

Content-Type 

application/json 

+ Add mapping template

application/json

© 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Step Functions Management X API Gateway

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST

Request body passthrough When no template matches the request Content-Type header ⚠

When there are no templates defined (recommended) ⓘ

Never ⓘ

Content-Type
application/json

+ Add mapping template

application/json

Generate template: dropdown

```
1 #set($data = $input.path('$'))
2
3 #set($input = { "dragon_name_str" : "$data.dragonName",
4     "$data.description" : "$data.description",
5     "family_str" : "$data.family",
6     "location_city_str" : "$data.city",
7     "location_country_str" : "$data.country",
8     "location_state_str" : "$data.state",
9     "location_neighborhood_str" : "$data.neighborhood",
10    "reportingPhoneNumber" : "$data.reportingPhoneNumber",
11    "confirmationRequired" : $data.confirmationRequired})
12
13 {
14     "input": "$util.escapeJavaScript($input).replaceAll("\\\\\"", '\"')",
15     "stateMachineArn": "<Replace>"}
```

Cancel Save

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

This is a mapping VTL which also add state machine ARN to the end of request.

Step Functions Management X API Gateway

← → C 🔒 console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST

Generate template:

```
1 #set($data = $input.path('$'))  
2  
3 #set($input = { "$dragon_name_str" : "$data.dragonName", "description_str" : "$data.description", "family_str" : "$data.family", "location_city_str" : "$data.city"  
"location_country_str" : "$data.country", "location_state_str" : "$data.state", "location_neighborhood_str" : "$data.neighborhood", "reportingPhoneNumber" : "$di  
.reportingPhoneNumber", "confirmationRequired" : $data.confirmationRequired})  
4  
5  
6 {  
7     "input": "$util.escapeJavaScript($input).replaceAll("\\\"", '\"'),  
8     "stateMachineArn": "<Replace>"  
9 }  
10 }
```

DescribeStateMachineAlias
DescribeStateMachineForExecution
GetActivityTask
GetExecutionHistory
ListActivities
ListExecutions
ListMapRuns
ListStateMachineAliases
ListStateMachines
ListStateMachineVersions
ListTagsForResource
PublishStateMachineVersion
SendTaskFailure
SendTaskHeartbeat
SendTaskSuccess
StartExecution
StartSyncExecution
StopExecution
TagResource
UntagResource
UpdateMapRun

Note

`StartExecution` is idempotent for STANDARD workflows. For a STANDARD workflow, if you call `StartExecution` with the same name and input as a running execution, the call succeeds and return the same response as the original request. If the execution is closed or if the input is different, it returns a `400 ExecutionAlreadyExists` error. You can reuse names after 90 days.

`StartExecution` isn't idempotent for EXPRESS workflows.

Request Syntax

```
{  
    "input": "string",  
    "name": "string",  
    "stateMachineArn": "string",  
    "traceHeader": "string"  
}
```

Request Parameters

For information about the parameters that are common to all actions, see [Common Parameters](#).

The request accepts the following data in JSON format.

[See also](#)

this is the `StartExecution` request syntax. We only used `input` and `stateMachineArn` in this example

Client Format

```
{  
  "dragonName": "Henry",  
  "description": "New dragon",  
  "family": "green",  
  "city": "Seattle",  
  "country": "USA",  
  "state": "WA",  
  "neighborhood": "Downtown",  
  "reportingPhoneNumber": "5555555555",  
  "confirmationRequired": true  
}
```

Backend Format

```
{  
  "description_str": "New Dragon",  
  "dragon_name_str": "Henry",  
  "family_str": "green",  
  "location_city_str": "Seattle",  
  "location_country_str": "USA",  
  "location_neighborhood_str":  
    "Downtown",  
  "location_state_str": "WA"  
}
```

This is what client send, and what the API mapping does with input to send to the backend.

Step Functions Management X API Gateway X StartExecution - AWS Step Fun X +

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST?screen=integration-request

APIs Resources Actions ▾ Method Execution /dragons - POST - Integration Request

Custom Domain Names

VPC Links

API: dragons

Resources

Stages

Authorizers

Gateway Responses

Models

Resource Policy

Documentation

Dashboard

Settings

Usage Plans

API Keys

Client Certificates

Settings

/

/dragons

 GET

 OPTIONS

POST

Provide information about the target backend that this method will call and whether the incoming request data should be modified.

Integration type Lambda Function 

HTTP 

Mock 

AWS Service 

VPC Link 

AWS Region us-east-1 

AWS Service Step Functions 

AWS Subdomain 

HTTP method POST 

Action StartExecution 

Execution role arn:aws:iam::302211264422:role/APIGatewayStepFunctions 

Credentials cache Do not add caller credentials to cache key 

Content Handling Passthrough  

Use Default Timeout 

URL Path Parameters

Everything is ready to test

Step Functions Management X API Gateway StartExecution - AWS Step Fun X +

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST?screen=integration-request

Request body passthrough When no template matches the request Content-Type header [?](#)
 When there are no templates defined (recommended) [?](#)
 Never [?](#)

Content-Type
application/json [-](#)

+ Add mapping template

application/json

Generate template: [▼](#)

```
1 #set($data = $input.path('$'))  
2  
3 #set($input = { "dragon_name_str": "$data.dragonName",  
4 "description_str": "$data.description", "family_str":  
5 "": "$data.family", "location_city_str": "$data.city",  
6 "location_country_str": "$data.country",  
7 "location_state_str": "$data.state",  
8 "location_neighborhood_str": "$data.neighborhood",  
9 "reportingPhoneNumber": "$data.reportingPhoneNumber",  
10 "confirmationRequired": $data.confirmationRequired})  
11  
12 {  
13     "input": "$util.escapeJavaScript($input).replaceAll("\\\"", '\"')"  
14 },  
15     "stateMachineArn": "arn:aws:states:us-east-1:302211264422  
16         :stateMachine:MyStateMachine"
```

Cancel Save

Feedback English (US) © 2008 - 2020, Amazon Web Services, Inc. or its affiliates. All rights reserved. Privacy Policy Terms of Use

Everything is ready to test (cont.)

Step Functions Management API Gateway StartExecution - AWS Step Function

console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST?screen=integration-request

Dashboard Settings Usage Plans API Keys Client Certificates Settings

Stage Variables No stage variables exist for this method.

Client Certificate No client certificates have been generated.

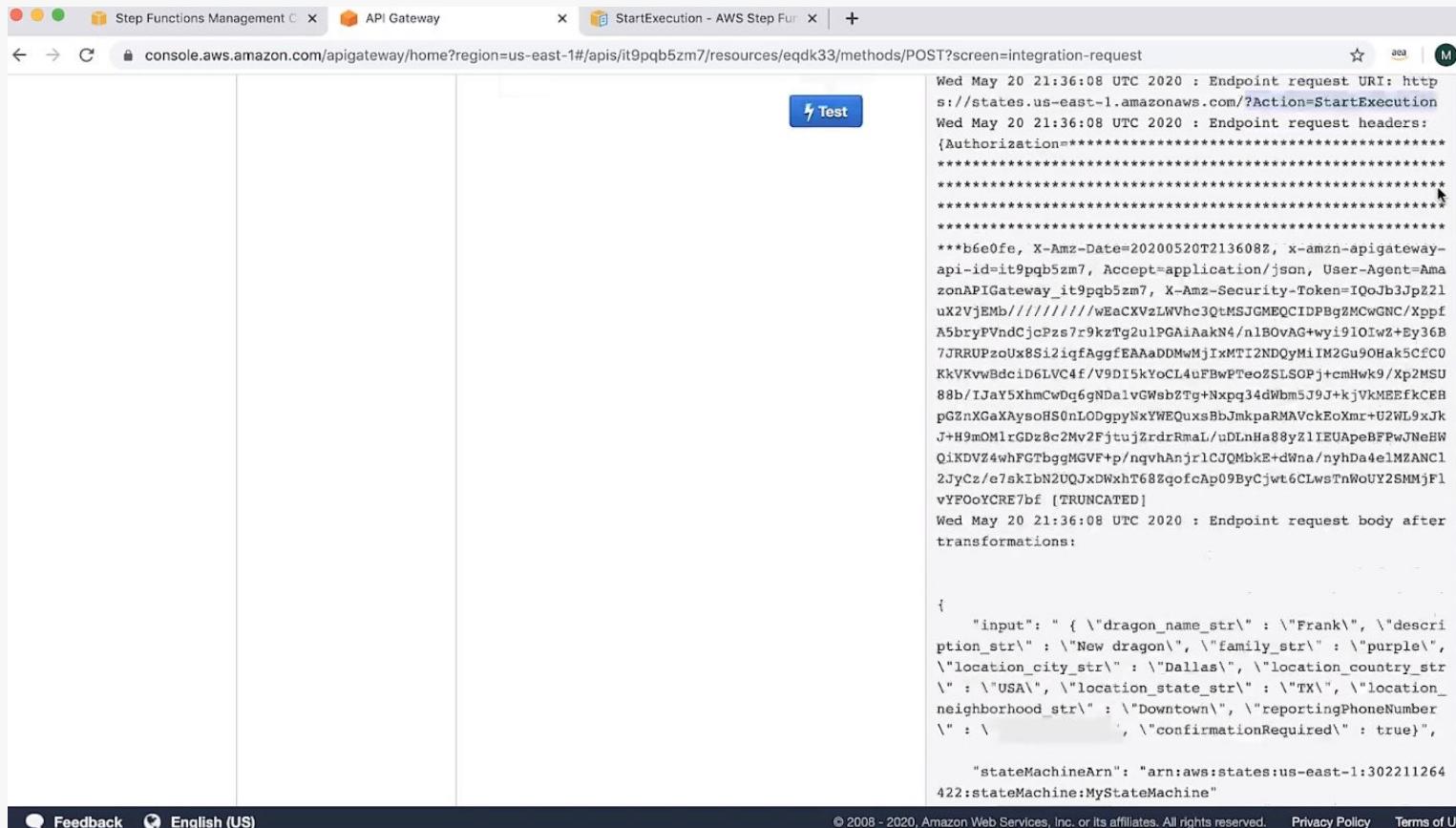
Request Body

```
1- {  
2 "dragonName": "Frank",  
3 "description": "New dragon",  
4 "family": "purple",  
5 "city": "Dallas",  
6 "country": "USA",  
7 "state": "TX",  
8 "neighborhood": "Downtown"  
9 "reportingPhoneNumber": "+  
10 "confirmationRequired": true  
11 }  
12
```

Test

Logs

```
Execution log for request clf5496c-bca2-4409-b7f1-962ef2cb4176  
Wed May 20 21:36:08 UTC 2020 : Starting execution for request: clf5496c-bca2-4409-b7f1-962ef2cb4176  
Wed May 20 21:36:08 UTC 2020 : HTTP Method: POST, Resource Path: /dragons  
Wed May 20 21:36:08 UTC 2020 : Method request path: {}  
Wed May 20 21:36:08 UTC 2020 : Method request query string: {}  
Wed May 20 21:36:08 UTC 2020 : Method request headers: {}  
Wed May 20 21:36:08 UTC 2020 : Method request body before transformations: {}  
"dragonName": "Frank",  
"description": "New dragon",  
"family": "purple",  
"city": "Dallas",  
"country": "USA",  
"state": "TX",  
"neighborhood": "Downtown",  
"reportingPhoneNumber": "+",  
"confirmationRequired": true  
}  
  
Wed May 20 21:36:08 UTC 2020 : Request validation succeeded for content type application/json  
Wed May 20 21:36:08 UTC 2020 : Endpoint request URI: http://states.us-east-1.amazonaws.com/?Action=StartExecution  
Wed May 20 21:36:08 UTC 2020 : Endpoint request headers:  
{Authorization=*****  
*****  
*****  
*****  
*****}
```



test result

Step Functions Management X API Gateway StartExecution - AWS Step Fun X + console.aws.amazon.com/apigateway/home?region=us-east-1#/apis/it9pqb5zm7/resources/eqdk33/methods/POST?screen=integration-request

Q1KDvZ4wnfGtDggMgv+p/nqvnAnjriCJQMDKs+dwna/nynda4eIMZANC1
2JyCz/e7skIBN2UQJxDWxh768zqofcAp09ByCjwt6CLwsTnWoUY2SMmjF1
vYFOoYCRE7bf [TRUNCATED]

Wed May 20 21:36:08 UTC 2020 : Endpoint request body after transformations:

```
{  
    "input": { \"dragon_name_str\" : \"Frank\", \"description_str\" : \"New dragon\", \"family_str\" : \"purple\",  
    \"location_city_str\" : \"Dallas\", \"location_country_str\" : \"USA\", \"location_state_str\" : \"TX\", \"location_neighborhood_str\" : \"Downtown\", \"reportingPhoneNumber\" : \", \"confirmationRequired\" : true},  
  
    "stateMachineArn": "arn:aws:states:us-east-1:302211264422:stateMachine:MyStateMachine"  
}
```

Wed May 20 21:36:08 UTC 2020 : Sending request to https://states.us-east-1.amazonaws.com/?Action=StartExecution

Wed May 20 21:36:09 UTC 2020 : Received response. Status: 200, Integration latency: 201 ms

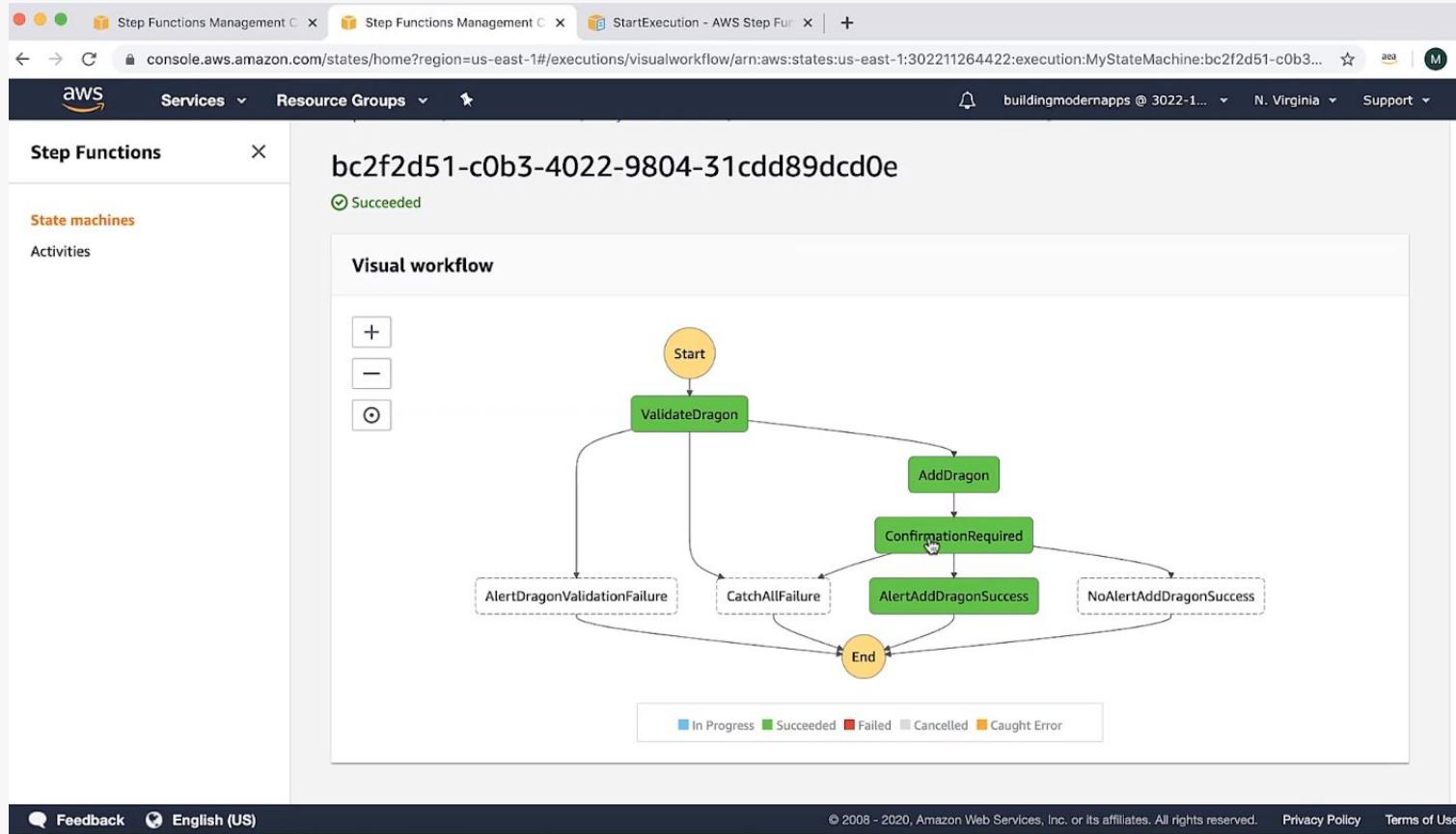
Wed May 20 21:36:09 UTC 2020 : Endpoint response headers: {x-amzn-RequestId=c90d9872-b1f0-4769-a574-f65c926e2c3b, Content-Type=application/x-amz-json-1.0, Content-Length=147}

Wed May 20 21:36:09 UTC 2020 : Endpoint response body before transformations: {"executionArn":"arn:aws:states:us-east-1:302211264422:execution:MyStateMachine:bc2f2d51-c0b3-4022-9804-31cdd89dc0e","startDate":1.590010568906E9}

Wed May 20 21:36:09 UTC 2020 : Method response body after transformations: {"executionArn":"arn:aws:states:us-east-1:302211264422:execution:MyStateMachine:bc2f2d51-c0b3-4022-9804-31cdd89dc0e","startDate":1.590010568906E9}

Wed May 20 21:36:09 UTC 2020 : Method response headers: {X-Amzn-Trace-Id=Root=1-sec5a2c8-d4012685ffdc054c0e7f14fe, C

We can see the mapping here



visual workflow of this execution

Good Resource for learning VTL

[Resolver mapping template programming guide - AWS AppSync](#)

[Apache Velocity Engine VTL Reference](#)

Run a Job and Wait for Callback Patterns

In the service integration of Step Functions,
we can call different AWS services directly.

There are three patterns for this integration:

Request Response, Run a Job, and Wait for Callback

AWS Step Functions

Developer Guide

What is AWS Step Functions?

Prerequisites

Getting started

Use cases

How Step Functions works

Workflow Studio

Tutorials

Developer tools

Testing state machines locally

Best practices

Working with other services

Sample projects for Step Functions

Quotas

Logging and monitoring

Security

Migrating workloads from AWS Data Pipeline

Troubleshooting

Related Information

Recent feature launches

Document history

AWS glossary

Supported service integrations

	Service	Request Response	Run a Job (.sync)	Wait for Callback (.waitForTaskToken)
Optimized integrations	Lambda	✓		✓
	AWS Batch	✓	✓	
	DynamoDB	✓		
	Amazon ECS/AWS Fargate	✓	✓	✓
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Glue	✓	✓	
	SageMaker	✓	✓	
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	CodeBuild	✓	✓	
	Athena	✓	✓	
	Amazon EKS	✓	✓	✓
	API Gateway	✓		✓
	AWS Glue DataBrew	✓	✓	
	Amazon EventBridge	✓		✓
AWS Step Functions	✓	✓	✓	
AWS SDK integrations	Over two hundred	✓		✓

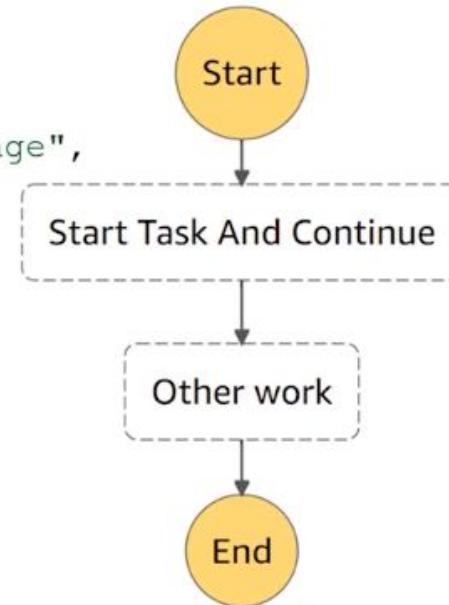
<https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html>

So far, we been looking at the first pattern.

In this section, we want to go in more details on the other two patterns.

Let's take an example, with the Amazon Simple Queue Service (SQS).
This is a queue service, which you can access with the HTTPS Protocol
to send messages and pull message out of that queue.

```
"Start Task And Continue": {  
    "Type": "Task",  
    "Resource": "arn:aws:states:::sns:publish",  
    "Parameters": {  
        "TopicArn": "<sns TOPIC ARN>",  
        "Message": "$.input.message"  
    }  
}
```

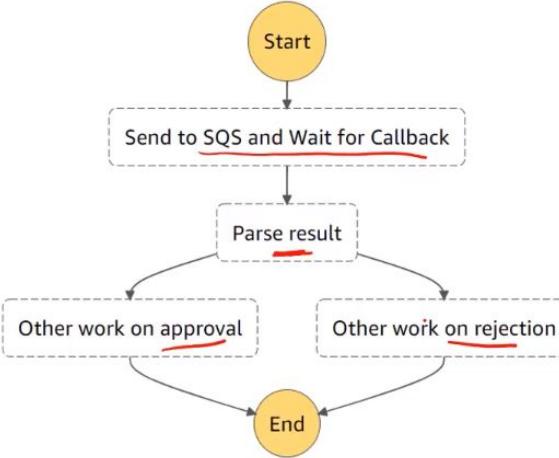


The way we are calling SQS here is via the request response pattern.

This means, that Step Functions will send a message in the queue, wait to make sure it was successfully accepted by the SQS Service, and if so, it will proceed with the task “Other work”.

This means that it doesn't matter what's happening with the message in the queue, the state machine we'll just keep going.

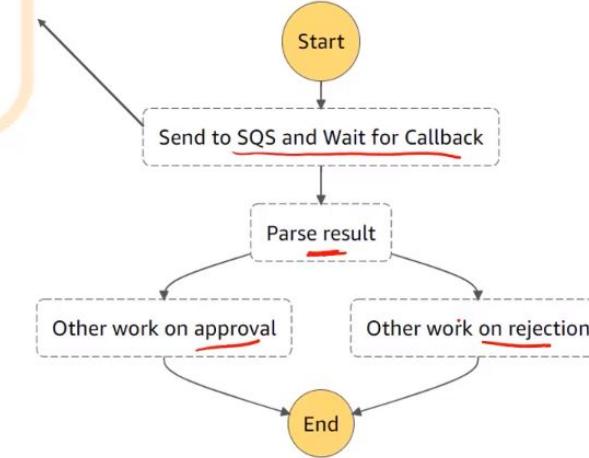
This is a valid pattern, but sometimes you may want to wait for that message to be pooled and be processed.



in this example, we choose a path based on the result from the SQS

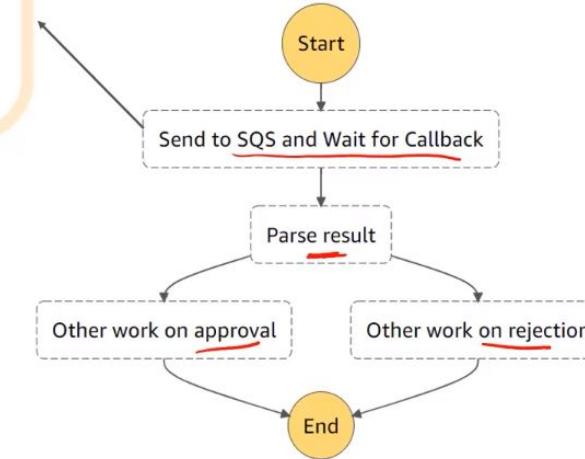
Task calls SQS with task token

```
"Start Task And Wait For Callback": {  
    "Type": "Task",  
    "Resource": "arn:aws:states::: sqs:sendMessage.waitForTaskToken",  
    "Parameters": {  
        "QueueUrl": "<SQS QUEUE URL>",  
        "MessageBody": {  
            "Input.$": "$",  
            "TaskToken.$": "$$.Task.Token"  
        }  
    }  
}
```



Task calls SQS with task token

```
"Start Task And Wait For Callback": {  
    "Type": "Task",  
    "Resource": "arn:aws:states::: sqs:sendMessage.waitForTaskToken",  
    "Parameters": {  
        "QueueUrl": "<SQS QUEUE URL>",  
        "MessageBody": {  
            "Input.$": "$",  
            "TaskToken.$": "$$.Task.Token"  
        }  
    }  
}
```



Task calls SQS with task token

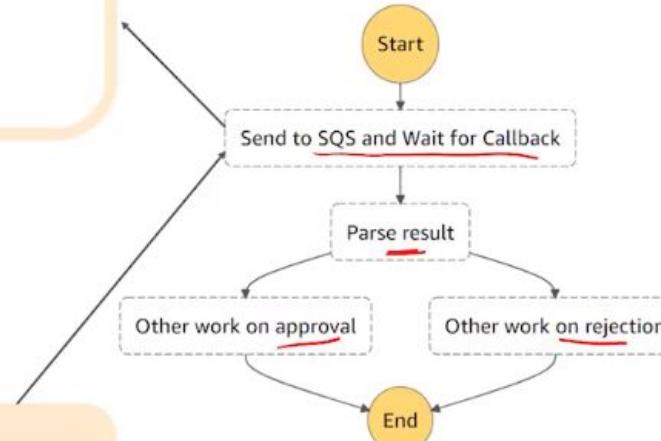
```
"Start Task And Wait For Callback": {  
    "Type": "Task",  
    "Resource": "arn:aws:states::: sqs:sendMessage.waitForTaskToken",  
    "Parameters": {  
        "QueueUrl": "<SQS QUEUE URL>",  
        "MessageBody": {  
            "Input.$": "$",  
            "TaskToken.$": "$$.Task.Token"  
        }  
    }  
}
```

External Process

1. SQS receives task token from Step Functions.
2. A backend process takes the message off the queue.
3. The backend process performs some work.
4. The process responds back to Step Functions.

Process returns task token with SendTaskSuccess

```
{"  
    "output": "approved",  
    "taskToken": "ASHDOIAASHDOIASHDLKASNLEKAHEOASIE"  
}
```



call, wait, and callback processes

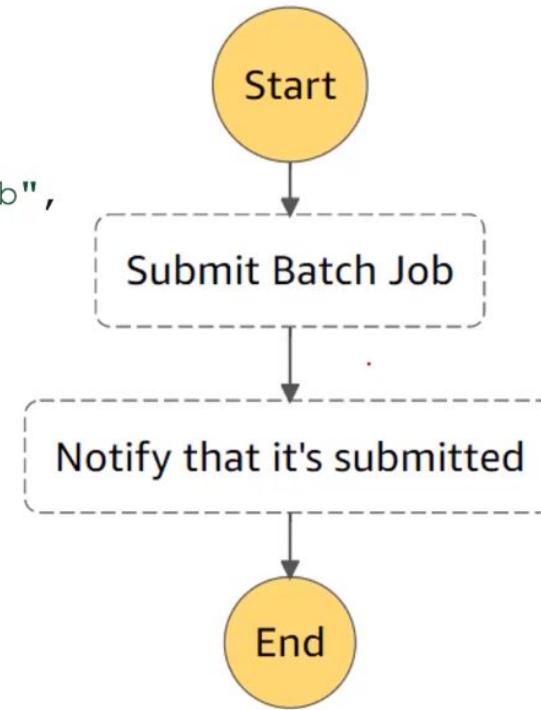
The third pattern is the `RunAJob` or `.SyncCommand`

This is for an integration with a service where you want Step Functions to wait for a request to complete before progressing to the next state.

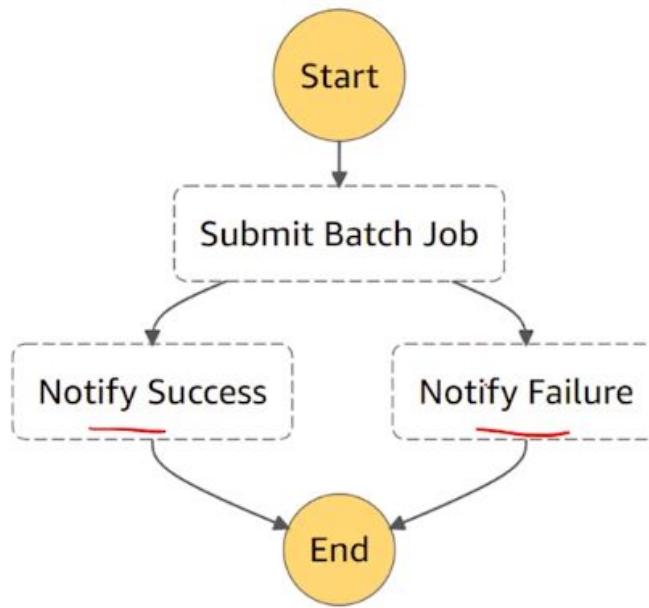
Let's take an example with the AWS Batch Service,
which allows to spin up some Servers,
execute scripts on those Servers to perform some work,
and tears down all of those Servers down afterward.

when executing this call, if all you cared about is to start AWS Batch (successful and not),
then run that service as a request response.

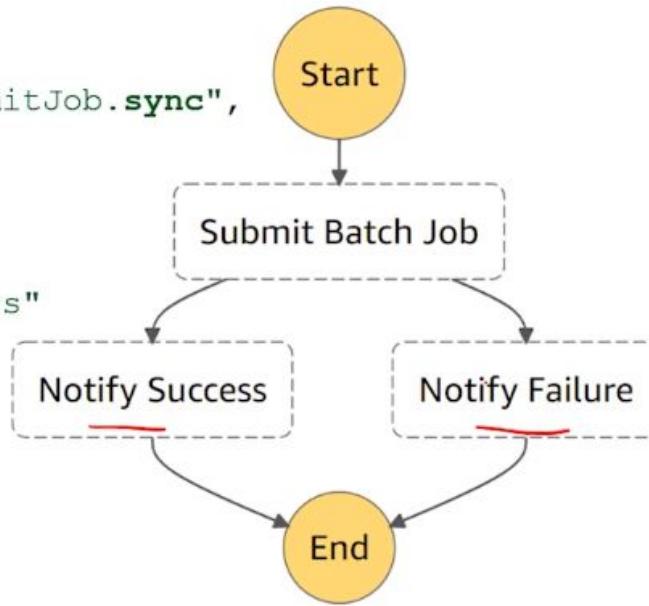
```
"Submit Batch Job": {  
    "Type": "Task",  
    "Resource": "arn:aws:states:::batch:submitJob",  
    "Parameters": {  
        "JobDefinition": "preprocessing",  
        "JobName": "preprocessingBatchJob",  
        "JobQueue": "primaryQueue",  
        "Parameters.$": "$.batchjob.parameters"  
    }  
}
```



However, if you wanted Step Functions to wait for the execution of that job to finish in AWS batch, to know if it ran successfully, or if it errored out, that workflow does not help.



```
"Submit Batch Job": {  
    "Type": "Task",  
    "Resource": "arn:aws:states:::batch:submitJob.sync",  
    "Parameters": {  
        "JobDefinition": "preprocessing",  
        "JobName": "preprocessingBatchJob",  
        "JobQueue": "primaryQueue",  
        "Parameters.$": "$.batchjob.parameters"  
    }  
}
```



we need to modify the Resource with adding .sync

AWS Step Functions

Developer Guide

What is AWS Step Functions?

Prerequisites

Getting started

Use cases

How Step Functions works

Workflow Studio

Tutorials

Developer tools

Testing state machines locally

Best practices

Working with other services

Sample projects for Step Functions

Quotas

Logging and monitoring

Security

Migrating workloads from AWS Data Pipeline

Troubleshooting

Related Information

Recent feature launches

Document history

AWS glossary

Supported service integrations

	Service	Request Response	Run a Job (.sync)	Wait for Callback (.waitForTaskToken)
Optimized integrations	Lambda	✓		✓
	AWS Batch	✓	✓	
	DynamoDB	✓		
	Amazon ECS/AWS Fargate	✓	✓	✓
	Amazon SNS	✓		✓
	Amazon SQS	✓		✓
	AWS Glue	✓	✓	
	SageMaker	✓	✓	
	Amazon EMR	✓	✓	
	Amazon EMR on EKS	✓	✓	
	Amazon EMR Serverless	✓	✓	
	CodeBuild	✓	✓	
	Athena	✓	✓	
	Amazon EKS	✓	✓	✓
	API Gateway	✓		✓
	AWS Glue DataBrew	✓	✓	
	Amazon EventBridge	✓		✓
AWS Step Functions	✓	✓	✓	
AWS SDK integrations	Over two hundred	✓		✓

<https://docs.aws.amazon.com/step-functions/latest/dg/welcome.html>

(recap)

Standard vs Express Step Functions

Up until now, we have been discussing standard workflows in Step Functions.

But there are actually two types of workflows: Standard and Express.

	Standard workflow	Express workflow
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

	Standard workflow	Express workflow
State transition persistence	Persisted to disk	In memory only
Execution semantics	Exactly-once workflow execution	At-least-once workflow execution

	Standard workflow	Express workflow
Pricing	Priced per state transition	Priced by the number of executions you run, their duration, and memory consumption
Maximum duration	365 days	5 minutes

	Standard workflow	Express workflow
Execution history	Stored in Step Functions and can be sent to Amazon CloudWatch Logs	Sent to Amazon CloudWatch Logs

	Standard workflow	Express workflow
Service integrations	Supports all service integrations and patterns	Supports all service integrations. Does not support Job-run (.sync) or Callback (.waitForTaskToken) patterns.

Good Resources:

[Choosing Standard or Express Workflows - AWS Step Functions](#)

Event Driven Architectures

A client sends an event through API gateway
and lets step function take responsibility for doing what it has to do.

This is an event-driven architecture.

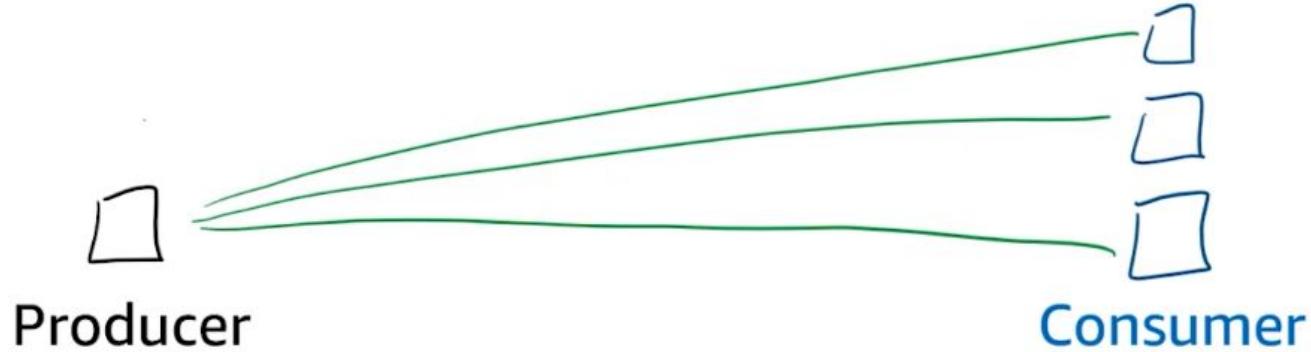
An **event-driven architecture** uses events to trigger and communicate between decoupled services and is common in modern applications, built with microservices.

In an event-driven system, the flow of the program is determined by **events** such as user actions (clicks, keystrokes), sensor outputs, or messages from other programs or services.

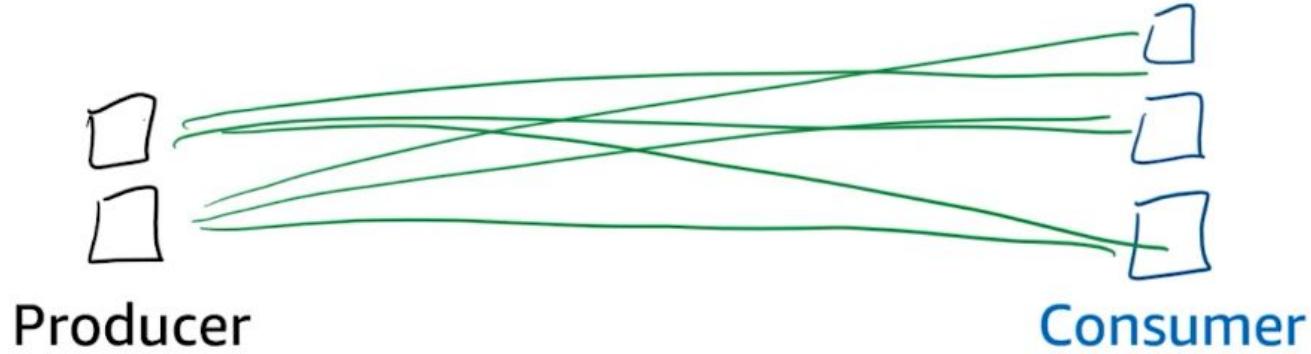
When we talk about events,
there is always a **producer** and a **consumer**.



When there are only one producer and one consumer, it's not very difficult.



with more consumer, the producer need to provide service to more consumers



finally, you have to add more producer which your architecture become a mess



Producer

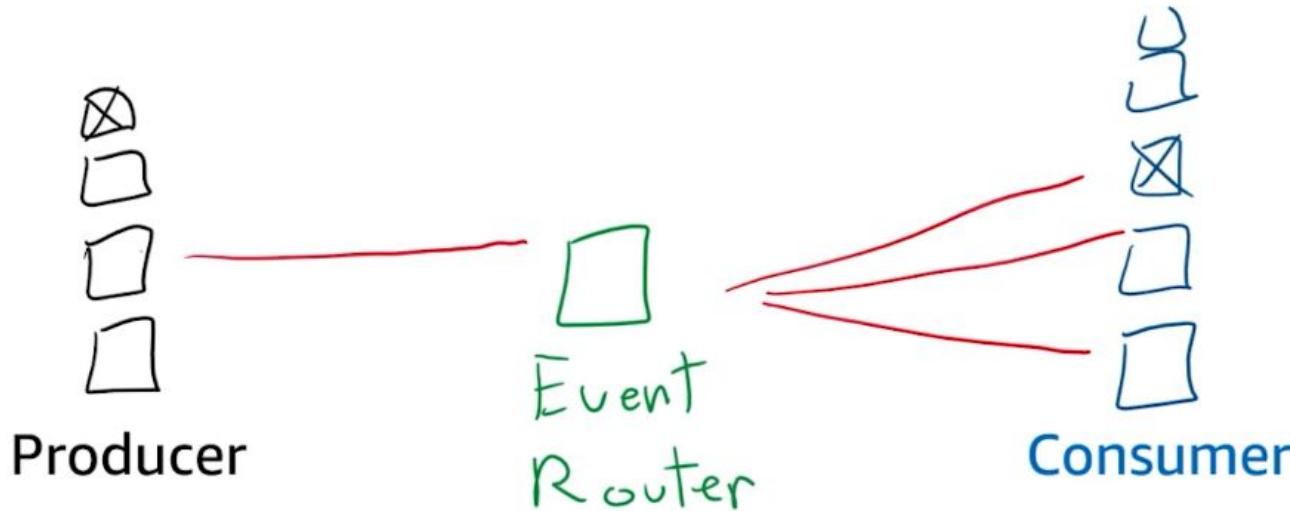


Event
Router



Consumer

Event Router decouple producers and consumers



Step Function — orchestr

Step Functions could act as a router, as it orchestrates events.

There are three more AWS services
that can act as routers.

Use them if you don't want as much orchestration as what Step Functions provide.

The first is a fully managed messaging queue service:
Amazon Simple Queue Service or SQS.

Using SQS, you can send, store and receive messages
between producers and consumers at any volume
without losing messages or requiring other services to be available.

Messages will stay in the queue for up to 14 days.

SQS offers two types of message queues: standard queues and FIFO

Standard queue offers a maximum throughput
but with a best effort ordering and in at least once delivery.

SQS First-In-First-Out are designed to guarantee that messages
are processed exactly once and in the exact order that they really have been sent.

To send messages, your producer will use the HTTPS protocol
and intend to receive messages,
your consumer will need to pull from the queue using the HTTPS protocol as well.

if there are no messages in the queue,
consumer is staying there, idling, and waiting for a message to finally arrive.

That's actually how queues work.

If you like your consumer to be notified that something was sent by a producer,
that's where you need a publish subscribe messaging service.

Amazon Simple Notification Service or SNS

SNS is a fully managed publish-subscribe messaging service
that's highly available, durable and secure.

It uses SNS topics for high-throughput, push-based, many-to-many messaging.

The third service is Amazon EventBridge service.

It's a service bus that makes it easy to connect producers and consumers together
using data your own application or
integrated software as a service application, or even AWS services.

Good Resources: SQS, SNS, EventBridge

[Amazon Simple Queue Service](#)

[Getting started with Amazon SNS - Amazon Simple Notification Service](#)

[Amazon EventBridge Documentation](#)

Exercise 5: AWS Step Functions