



**AWS  
re:Invent**

**CMP319**

**NEW LAUNCH!**

# **Building Distributed Applications with AWS Step Functions**

Andy Katz, Senior Product Manager, AWS  
Manuel Pata, Cloud Automation Team Leader, OutSystems

December 1, 2016

# What to Expect from the Session



What is AWS Step Functions?

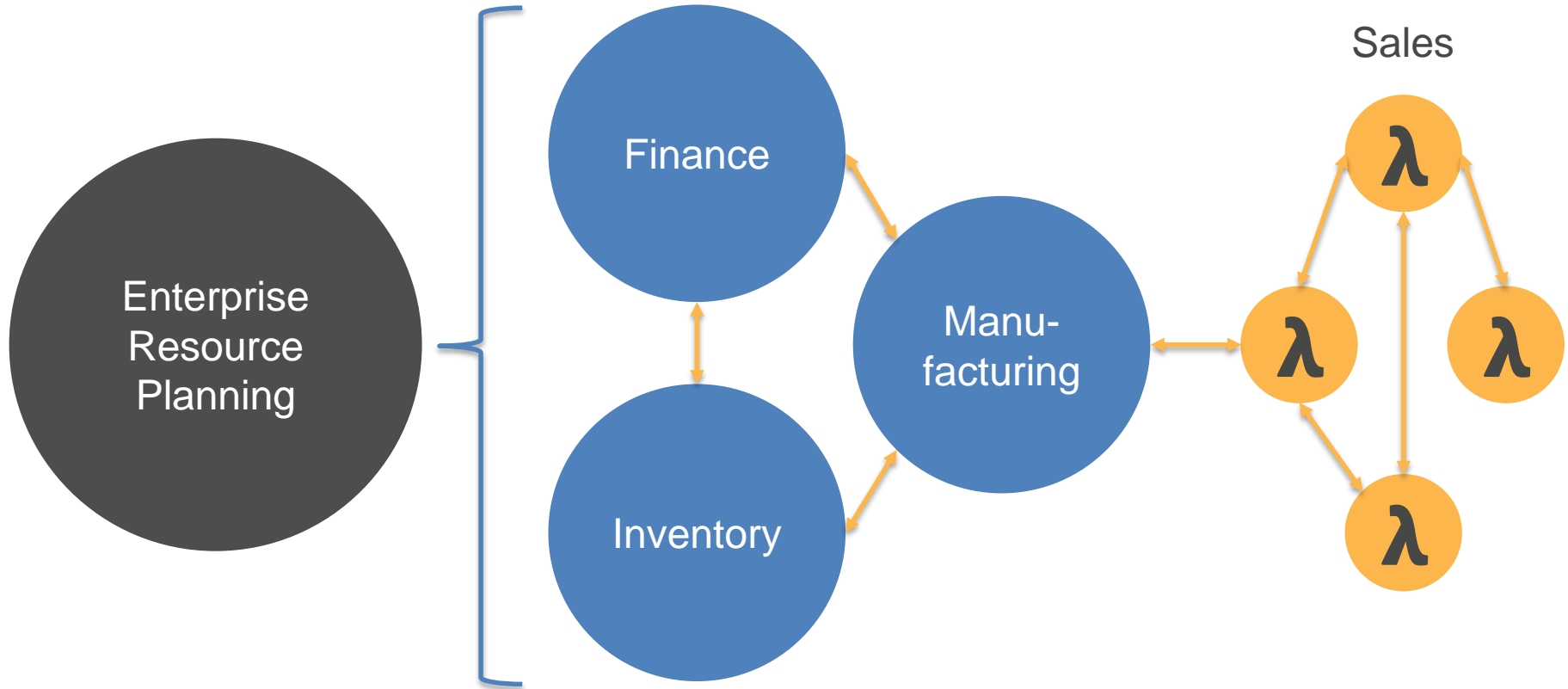
How does it work?

Why should I use it?

What can I do today?

**What is AWS Step Functions?**

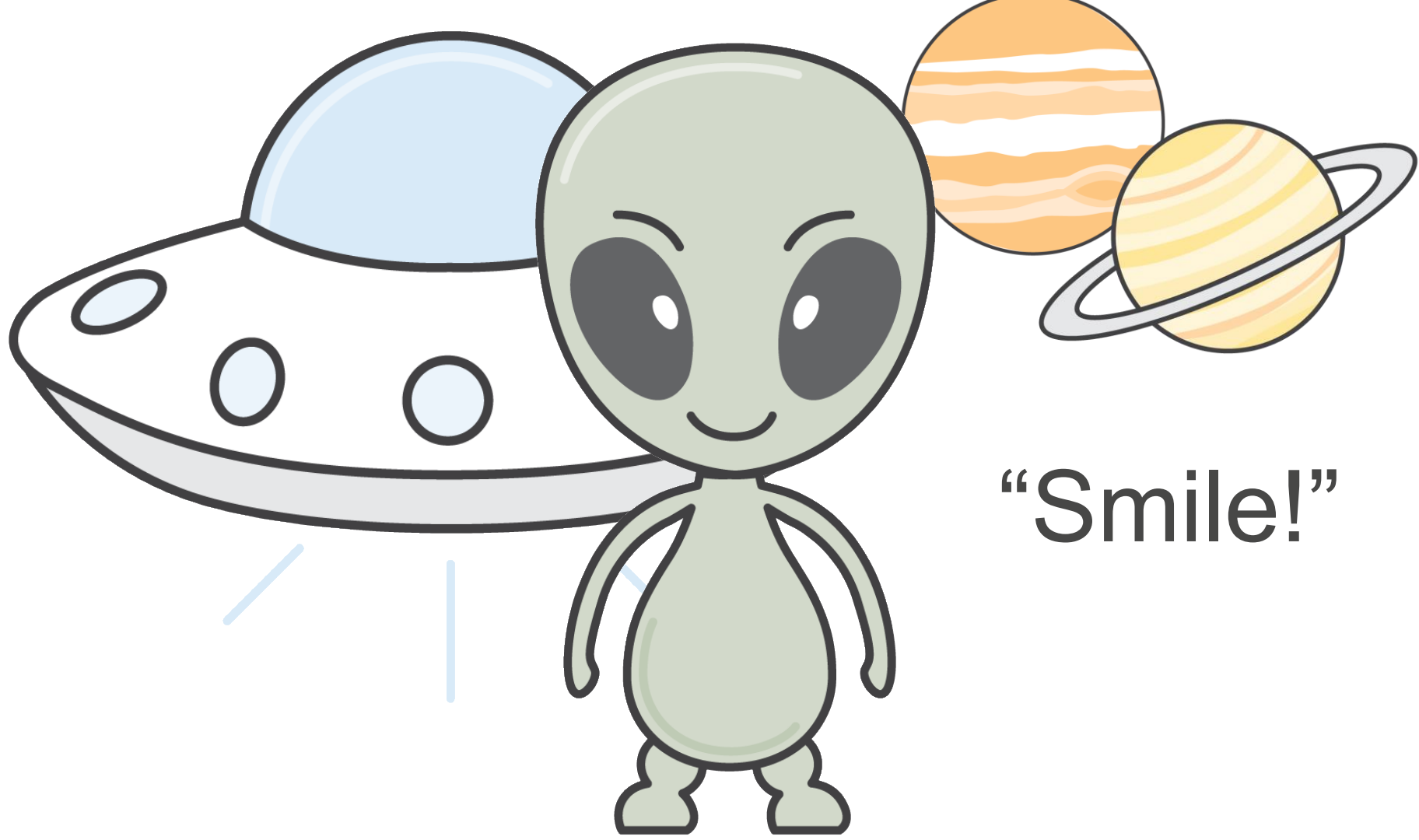
# From Monoliths to Microservices to Functions



# AWS Step Functions...

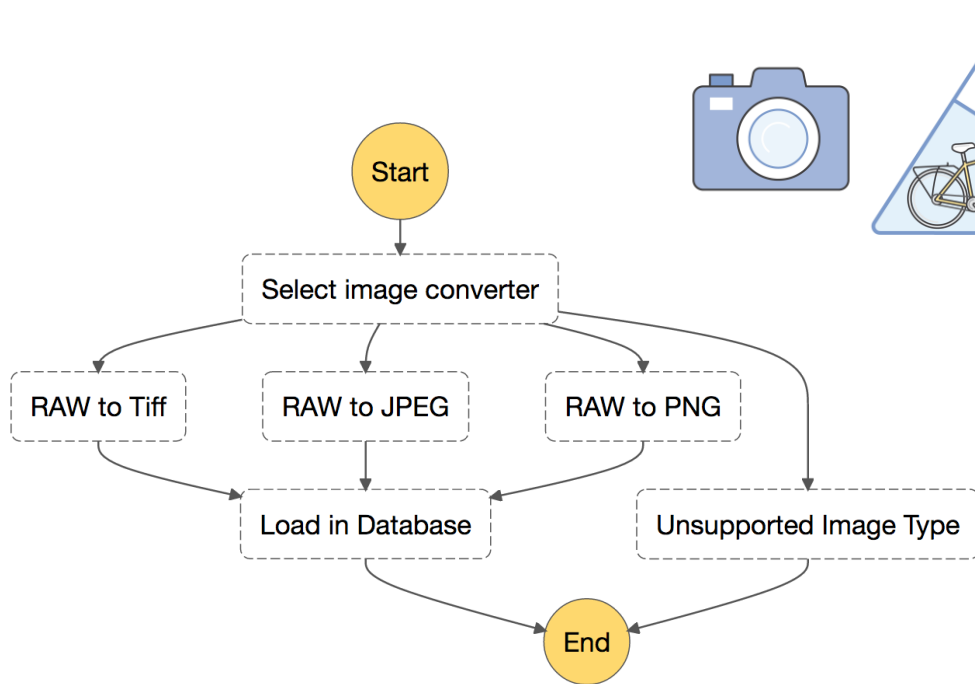


...makes it easy to coordinate the components of distributed applications using visual workflows.

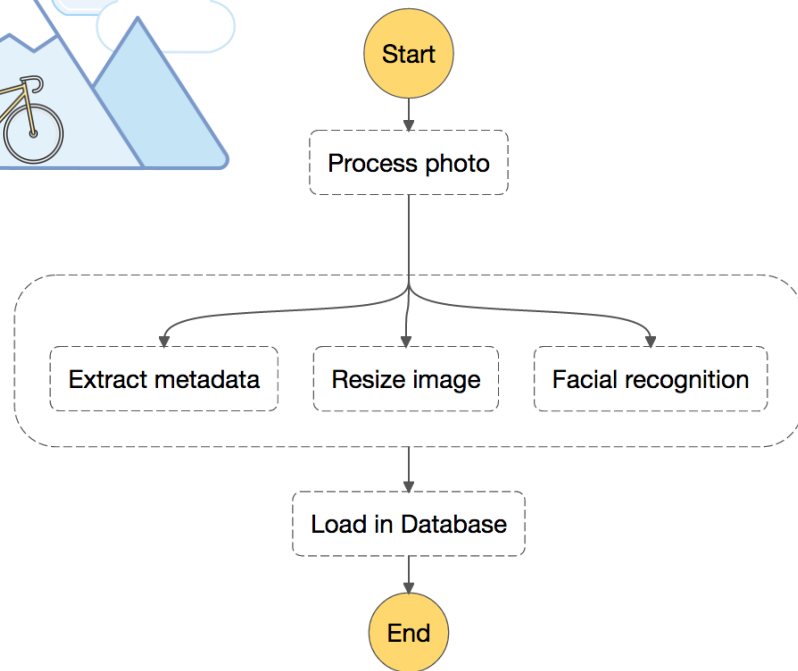
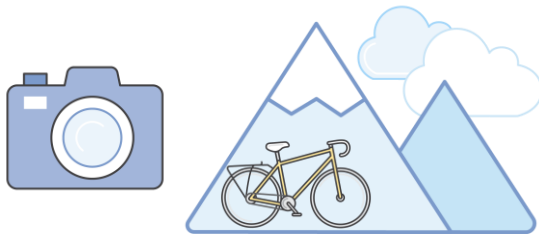


“Smile!”

# Step through Microservices and Functions



**File Type Conversion**



**Parallel Image Processing**

# Is this you?

**“I want to sequence my services”**

**“I want to run tasks in parallel”**

**“I want to select paths based on previous results”**

**“I want to retry automatically when a service is unavailable”**

**“I have code that runs for hours”**



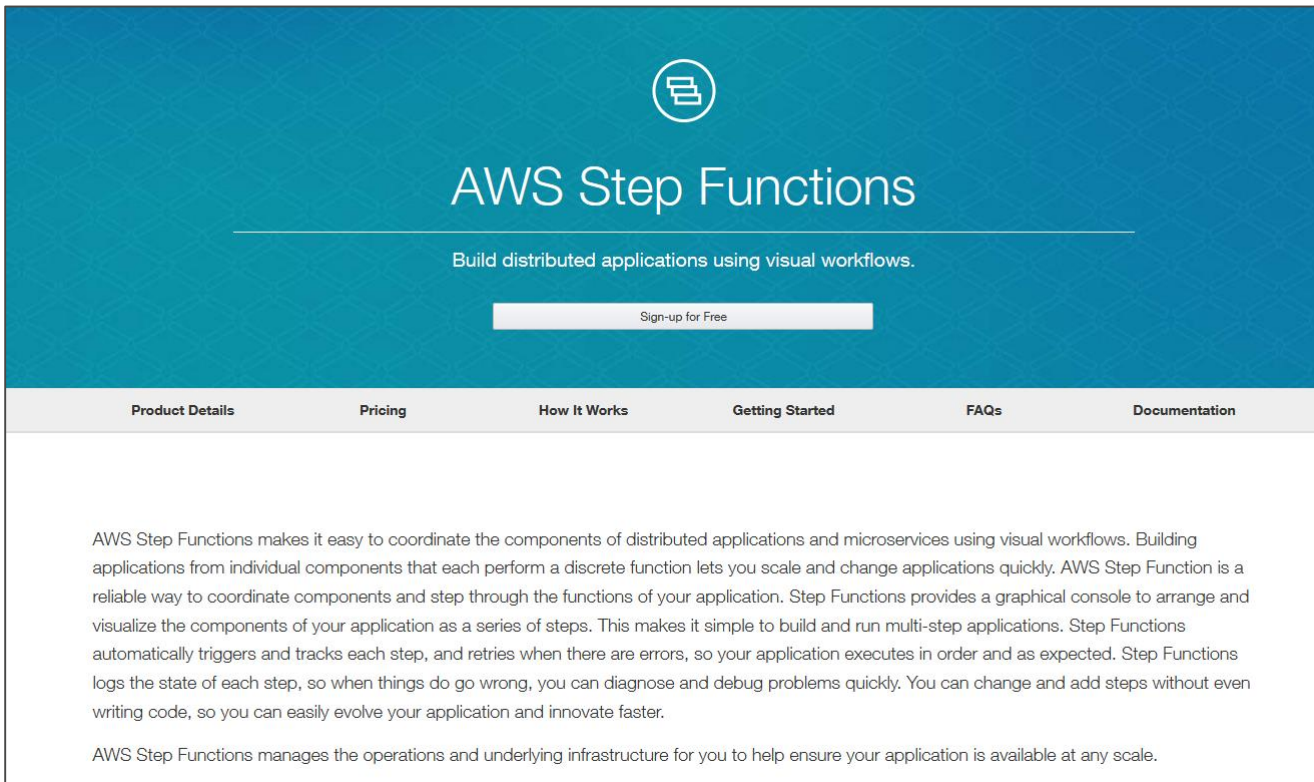
## **Frequently repeated processes**

- Report generation
- Order fulfillment
- Data processing
- Infrastructure automation



# AWS Step Functions is Available Today

[aws.amazon.com/step-functions](https://aws.amazon.com/step-functions)



The image shows a screenshot of the AWS Step Functions landing page. The page has a blue header with the AWS Step Functions logo (a circle containing a stack of three rectangles) and the text "AWS Step Functions". Below the header, there is a white box with the text "Build distributed applications using visual workflows." and a "Sign-up for Free" button. Below this is a navigation bar with links: "Product Details", "Pricing", "How It Works", "Getting Started", "FAQs", and "Documentation". The main content area has a white background and contains a paragraph of text about AWS Step Functions, followed by a sub-header "Getting started with AWS Step Functions" and a list of links: "Tutorial", "Getting started with AWS Step Functions", "AWS Step Functions console", "AWS Step Functions API", "AWS Step Functions SDKs", "AWS Step Functions CLI", "AWS Step Functions CloudFormation templates", "AWS Step Functions IAM permissions", "AWS Step Functions VPC subnets", "AWS Step Functions security groups", "AWS Step Functions logging", "AWS Step Functions monitoring", "AWS Step Functions troubleshooting", "AWS Step Functions FAQs", "AWS Step Functions documentation", "AWS Step Functions support", "AWS Step Functions contact us", "AWS Step Functions feedback", "AWS Step Functions suggestions", "AWS Step Functions ideas", "AWS Step Functions roadmap", "AWS Step Functions roadmap 2019", "AWS Step Functions roadmap 2020", "AWS Step Functions roadmap 2021", "AWS Step Functions roadmap 2022", "AWS Step Functions roadmap 2023", "AWS Step Functions roadmap 2024", "AWS Step Functions roadmap 2025", "AWS Step Functions roadmap 2026", "AWS Step Functions roadmap 2027", "AWS Step Functions roadmap 2028", "AWS Step Functions roadmap 2029", "AWS Step Functions roadmap 2030".

**AWS Step Functions**

Build distributed applications using visual workflows.

[Sign-up for Free](#)

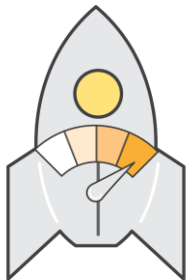
[Product Details](#) [Pricing](#) [How It Works](#) [Getting Started](#) [FAQs](#) [Documentation](#)

AWS Step Functions makes it easy to coordinate the components of distributed applications and microservices using visual workflows. Building applications from individual components that each perform a discrete function lets you scale and change applications quickly. AWS Step Function is a reliable way to coordinate components and step through the functions of your application. Step Functions provides a graphical console to arrange and visualize the components of your application as a series of steps. This makes it simple to build and run multi-step applications. Step Functions automatically triggers and tracks each step, and retries when there are errors, so your application executes in order and as expected. Step Functions logs the state of each step, so when things do go wrong, you can diagnose and debug problems quickly. You can change and add steps without even writing code, so you can easily evolve your application and innovate faster.

AWS Step Functions manages the operations and underlying infrastructure for you to help ensure your application is available at any scale.

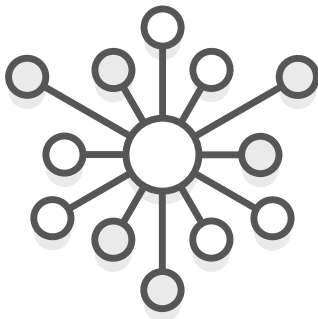
# Benefits of AWS Step Functions

## Productivity



Easy to connect and coordinate distributed components and microservices to quickly create apps

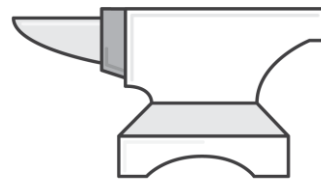
## Agility



Diagnose and debug problems faster

Adapt to change

## Resilience



Manages the operations and infrastructure of service coordination to ensure availability at scale, and under failure

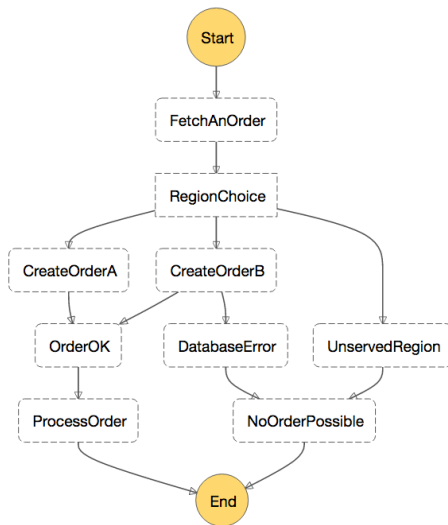
# How Does AWS Step Functions Work?

# Application Lifecycle in AWS Step Functions

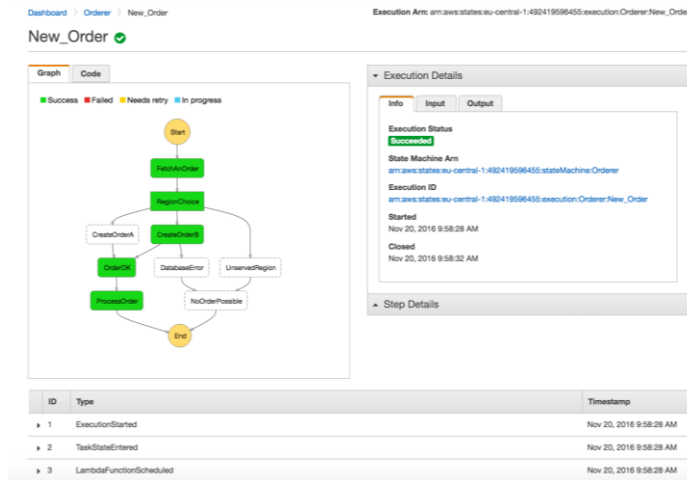
## Define in JSON

```
Code
1 {
2   "Comment": "An AWL example using a choice state.",
3   "StartAt": "FirstState",
4   "States": {
5     "FirstState": {
6       "Type": "Task",
7       "Resource": "arn:aws:lambda:REGION:ACCOUNT_ID:function:FUNCTION_NAME",
8       "Next": "ChoiceState"
9     },
10    "ChoiceState": {
11      "Type": "Choice",
12      "Choices": [
13        {
14          "Variable": "$?.OrderType",
15          "Succeed": "CreateOrderA",
16          "Fail": "CreateOrderB"
17        }
18      ]
19    }
20  }
21 }
```

## Visualize in the Console

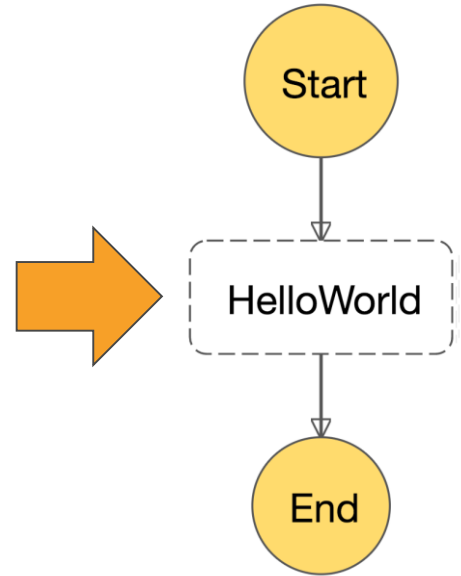


## Monitor Executions

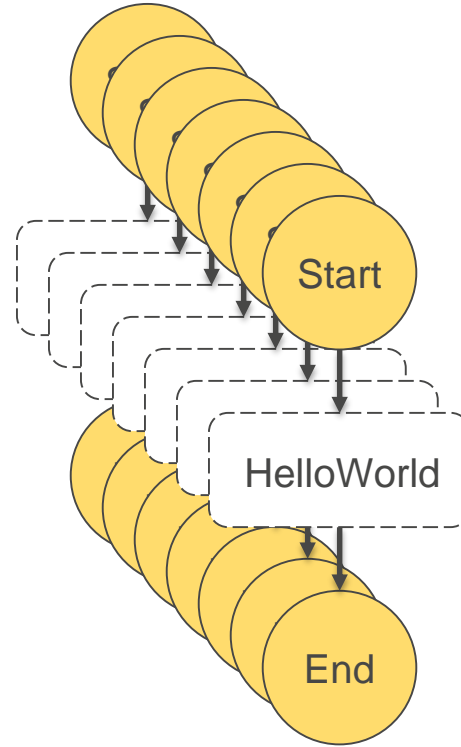
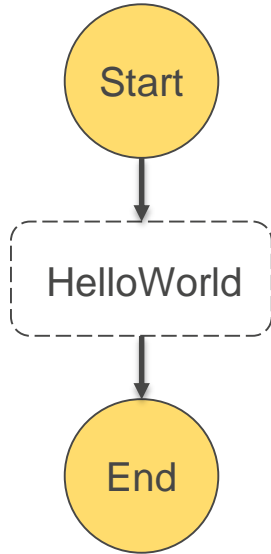


# Define in JSON and Then Visualize in the Console

```
{  
  "Comment": "Hello world Example",  
  "StartAt" : "HelloWorld",  
  "States" : {  
    "HelloWorld" : {  
      "Type" : "Task",  
      "Resource" :  
"arn:aws:lambda:REGION:ACCOUNT_ID:function:  
:FUNCTION_NAME",  
      "End" : true  
    }  
  }  
}
```



# Execute One or One Million

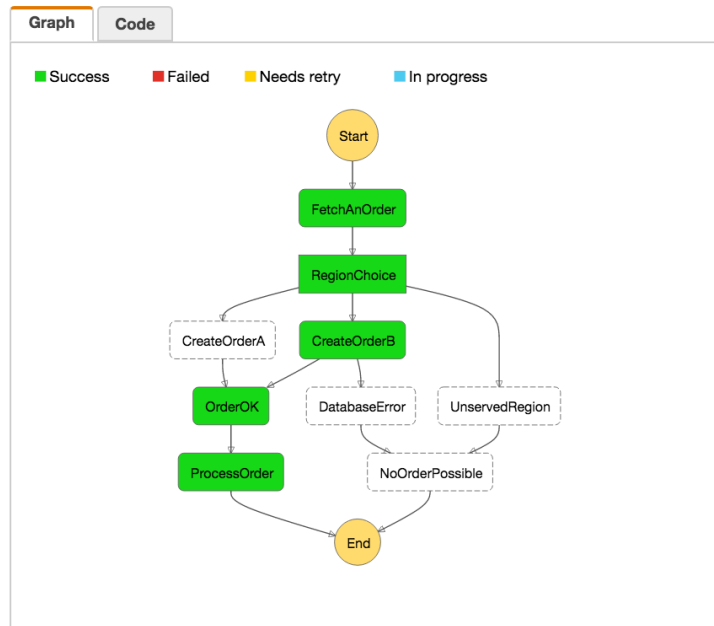


# Monitor Executions from the Console

[Dashboard](#) > [Orderer](#) > New\_Order

Execution Arn: [arn:aws:states:eu-central-1:55:execution:Orderer:New\\_Order](#)

New\_Order ✓



## Execution Details

Info

Input

Output

### Execution Status

**Succeeded**

### State Machine Arn

[arn:aws:states:eu-central-1:55:stateMachine:Orderer](#)

### Execution ID

[arn:aws:states:eu-central-1:55:execution:Orderer:New\\_Order](#)

### Started

Nov 20, 2016 9:58:28 AM

### Closed

Nov 20, 2016 9:58:32 AM

## Step Details

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

# Monitor Executions from Amazon CloudWatch

CloudWatch  
Dashboards

Alarms

ALARM

INSUFFICIENT

OK

Billing

Events

Rules

Logs

Metrics NEW

0

0

0

Lambda Task State

2016-11-19 (22:10:00) - 2016-11-19 (22:28:00)

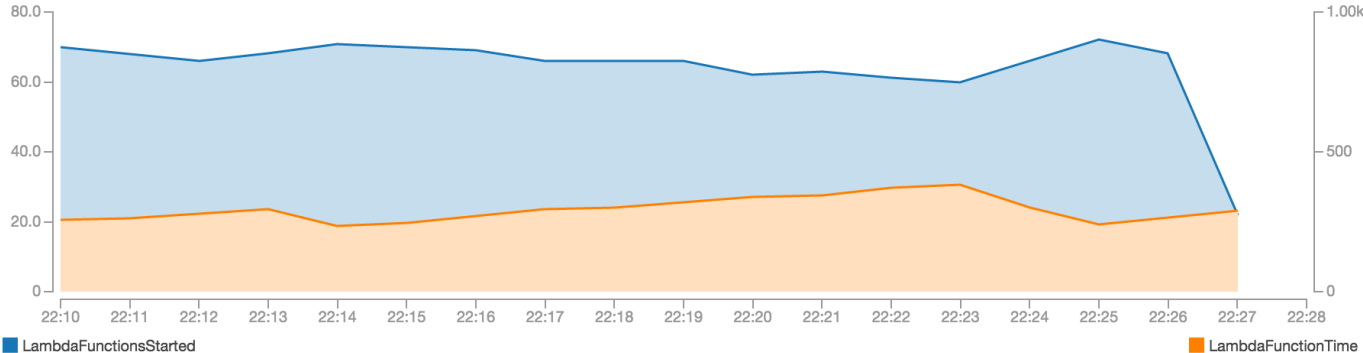
Stacked area

Actions

Refresh

Dropdown

Help

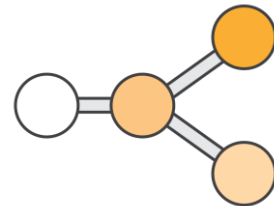


All metrics								
Graphed metrics (2)								
Graph options								
	Label	Namespace	Dimensions	Metric Na...	Statistic	Period	Y Axis	Actions
●	Lamb...	AWS/States	Dimensions (1)	LambdaFunc	Sum	1 Minute	< >	🔔 📄 ✕
●	Lamb...	AWS/States	Dimensions (1)	LambdaFunc	Average	1 Minute	< >	🔔 📄 ✕

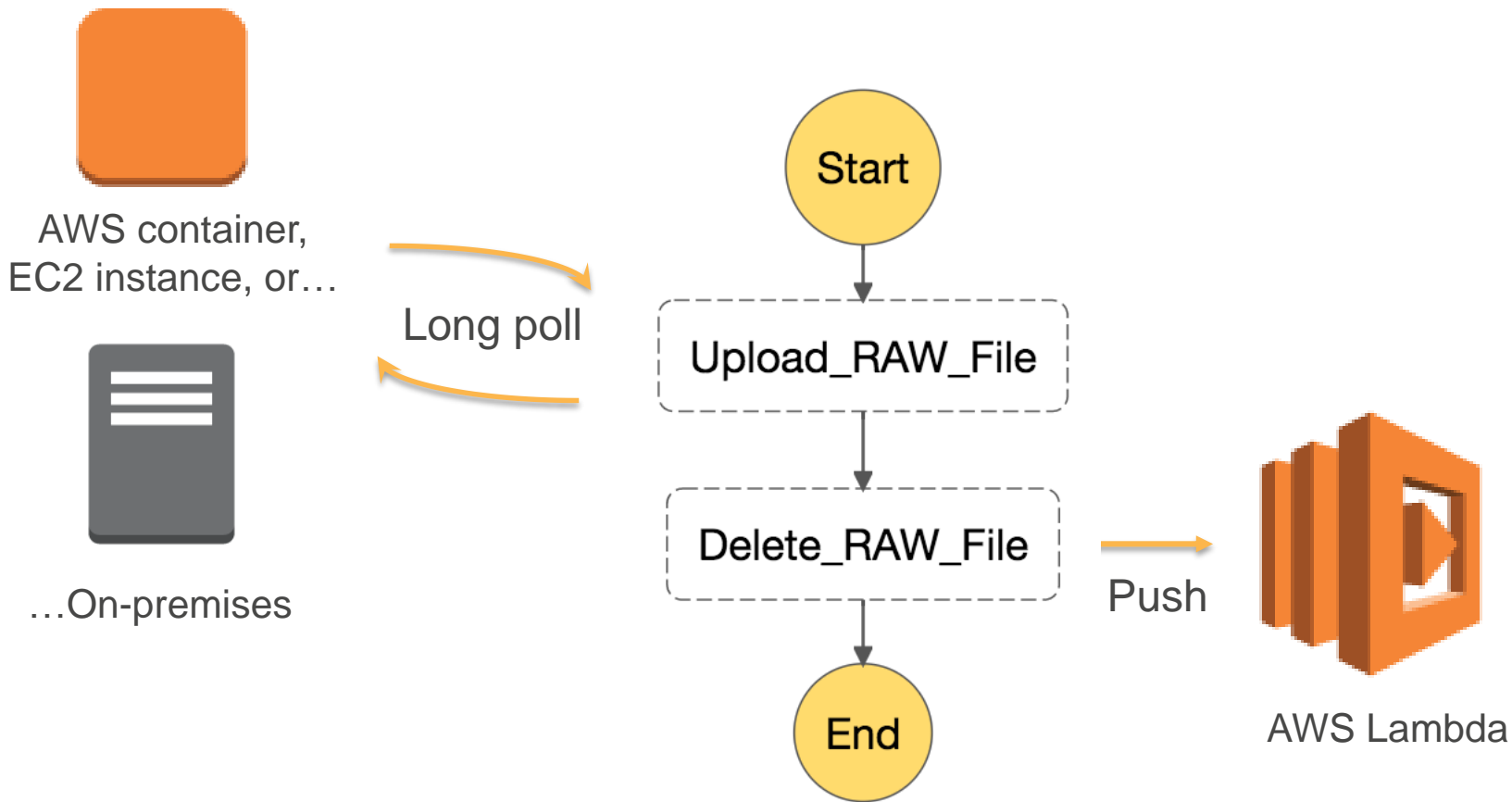


# Seven State Types

<b>Task</b>	A single unit of work
<b>Choice</b>	Adds branching logic
<b>Parallel</b>	Fork and join the data across tasks
<b>Wait</b>	Delay for a specified time
<b>Fail</b>	Stops an execution and marks it as a failure
<b>Succeed</b>	Stops an execution successfully
<b>Pass</b>	Passes its input to its output

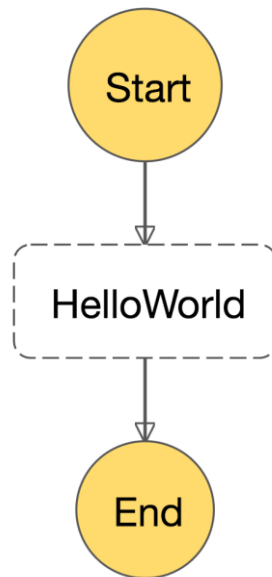


# Task States Poll or Push



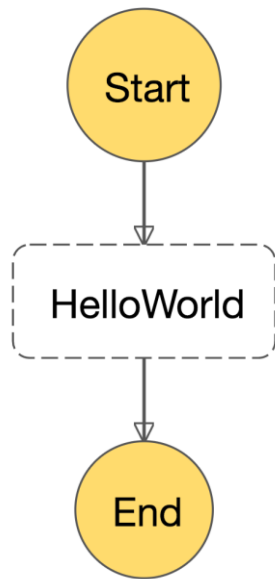
# Task State to Dispatch Work

```
{
  "startAt": "HelloWorld",
  "states": {
    "HelloWorld": {
      "Type": "Task",
      "Resource": "arn:aws:lambda:REGION:
        ACCOUNT_ID:function:FUNCTION_NAME",
      "End": true
    }
  }
}
```



# Retry Failures of Task States

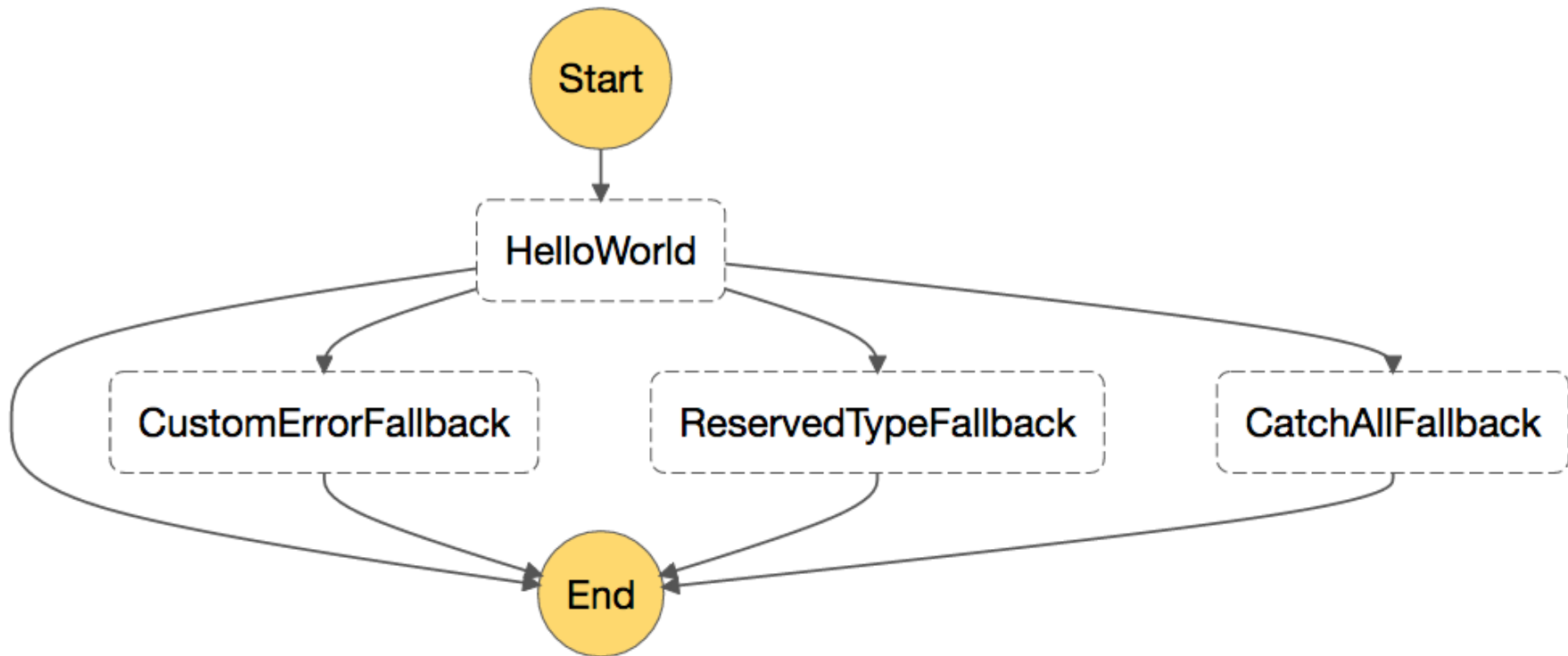
```
"HelloWorld": {  
  "Type": "Task",  
  ...  
  "Retry": [  
    {  
      "ErrorEquals": ["HandledError"],  
      "IntervalSeconds": 1,  
      "MaxAttempts": 2,  
      "BackoffRate": 2.0  
    },  
    ... ]  
}
```



# Catch Failure of Task States

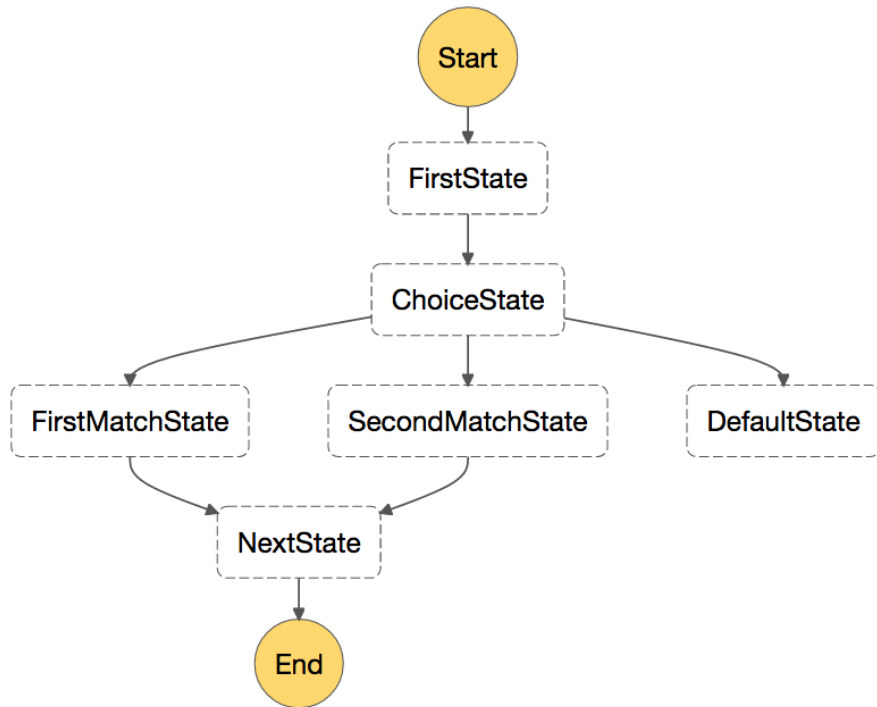
```
"HelloWorld": {  
  "Type": "Task",  
  ...  
  "Catch": [  
    {  
      "ErrorEquals": ["AllYourBasesAreBelongToUs"],  
      "Next": "CustomErrorFallback"  
    },  
    {  
      "ErrorEquals": ["States.TaskFailed"],  
      "Next": "ReservedTypeFallback"  
    }  
  ]  
}
```

# Catch Failure of Task State Blueprint



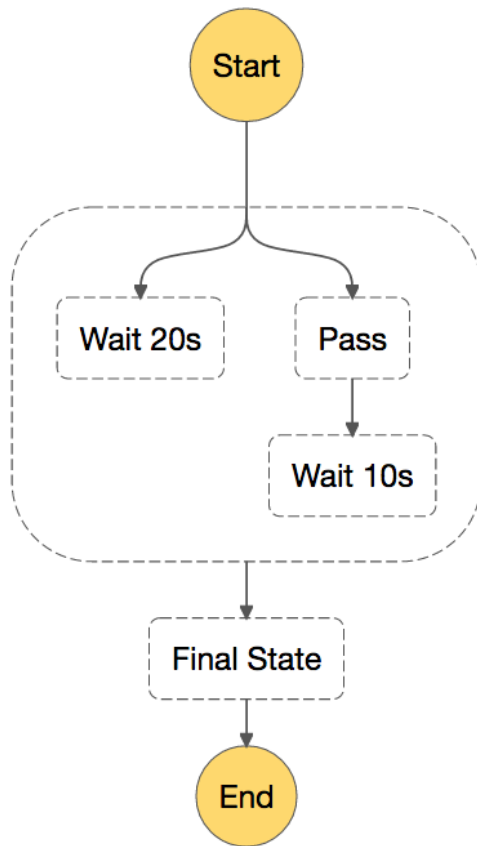
# Choice State for Branching Logic

```
"ChoiceState": {  
  "Type" : "Choice",  
  "Choices": [  
    {  
      "Variable": "$.foo",  
      "NumericEquals": 1,  
      "Next": "FirstMatchState"  
    },  
    {  
      "Variable": "$.foo",  
      "NumericEquals": 2,  
      ... ]  
}
```



# Parallel State to Fork and Join Processes

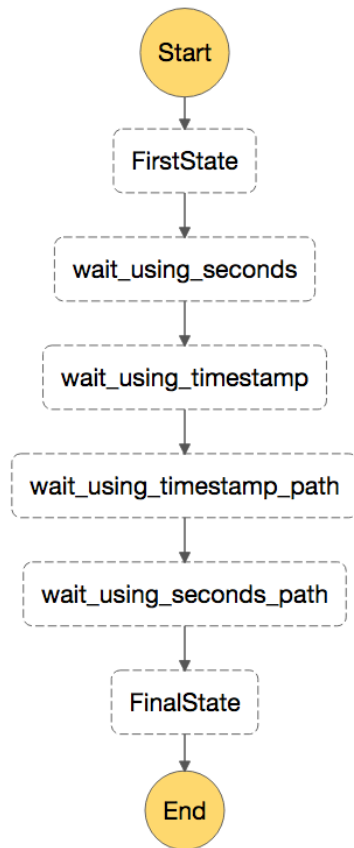
```
"Parallel": {  
  "Type": "Parallel",  
  "Next": "Final State",  
  "Branches": [  
    {  
      "StartAt": "wait 20s",  
      "States": {  
        "wait 20s": {  
          "Type": "wait",  
          "Seconds": 20,  
          "End": true  
        }  
      }  
    },  
    ... ]  
}
```





# Wait State for Timed Delay

```
"wait_using_seconds": {  
  "Type": "wait",  
  "Seconds": 10,  
  "Next": "wait_using_timestamp"  
},  
"wait_using_timestamp": {  
  "Type": "wait",  
  "Timestamp": "2015-09-04T01:59:00Z",  
  "Next": "wait_using_timestamp_path"  
},  
...
```



# Six Blueprints in the Console



Wait State



Hello World



Retry Failure



Parallel



Catch Failure



Choice State

# Use Through the API

<b>Create</b>	Upload state machines defined in JSON Register activity workers
<b>StartExecution</b>	Returns Execution ID
<b>StopExecution</b>	Stops a running state machine with Execution ID
<b>List</b>	All state machines, executions, and activities
<b>Describe</b>	Individual state machines, executions, and activities

# **Why Should I Use AWS Step Functions?**

# Ensure Tasks Execute in Sequence

## Reliably Process Orders



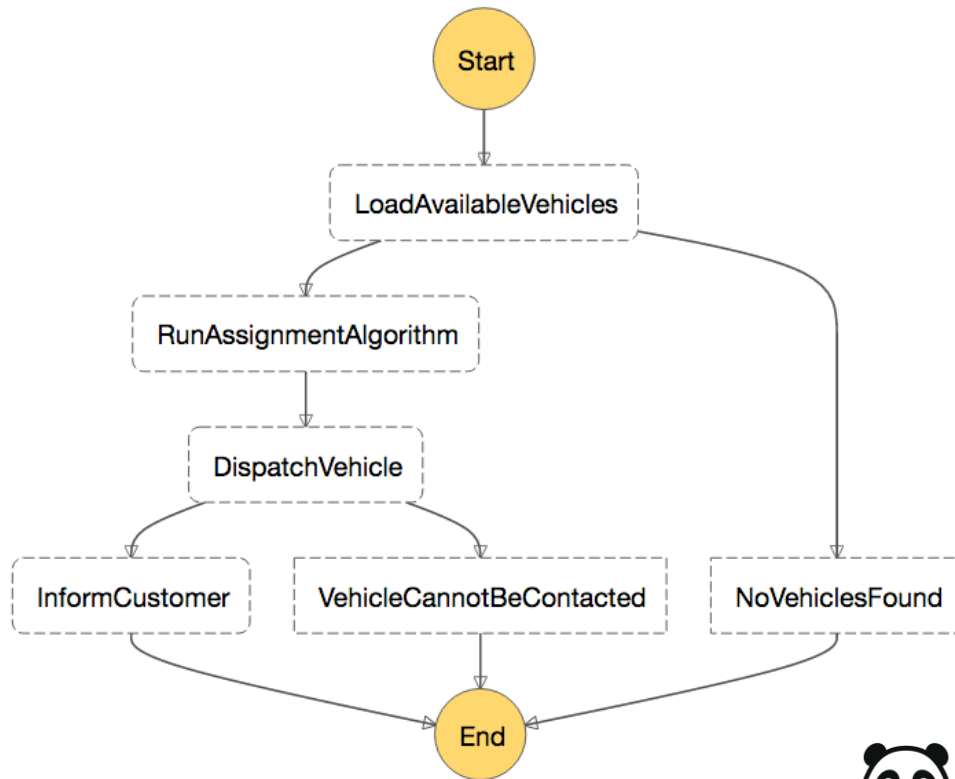
**foodpanda**

*“With AWS Step Functions, we can easily change and iterate on the application workflow of our food delivery service in order to optimize operations and continually improve delivery times.*

*AWS Step Functions lets us dynamically scale the steps in our food delivery algorithm so we can manage spikes in customer orders and meet demand.”*

Mathias Nitzsche, CTO, foodpanda

# Try-Catch-Finally in foodpanda's State Machine



# Chose Logical Paths Based on Input Data

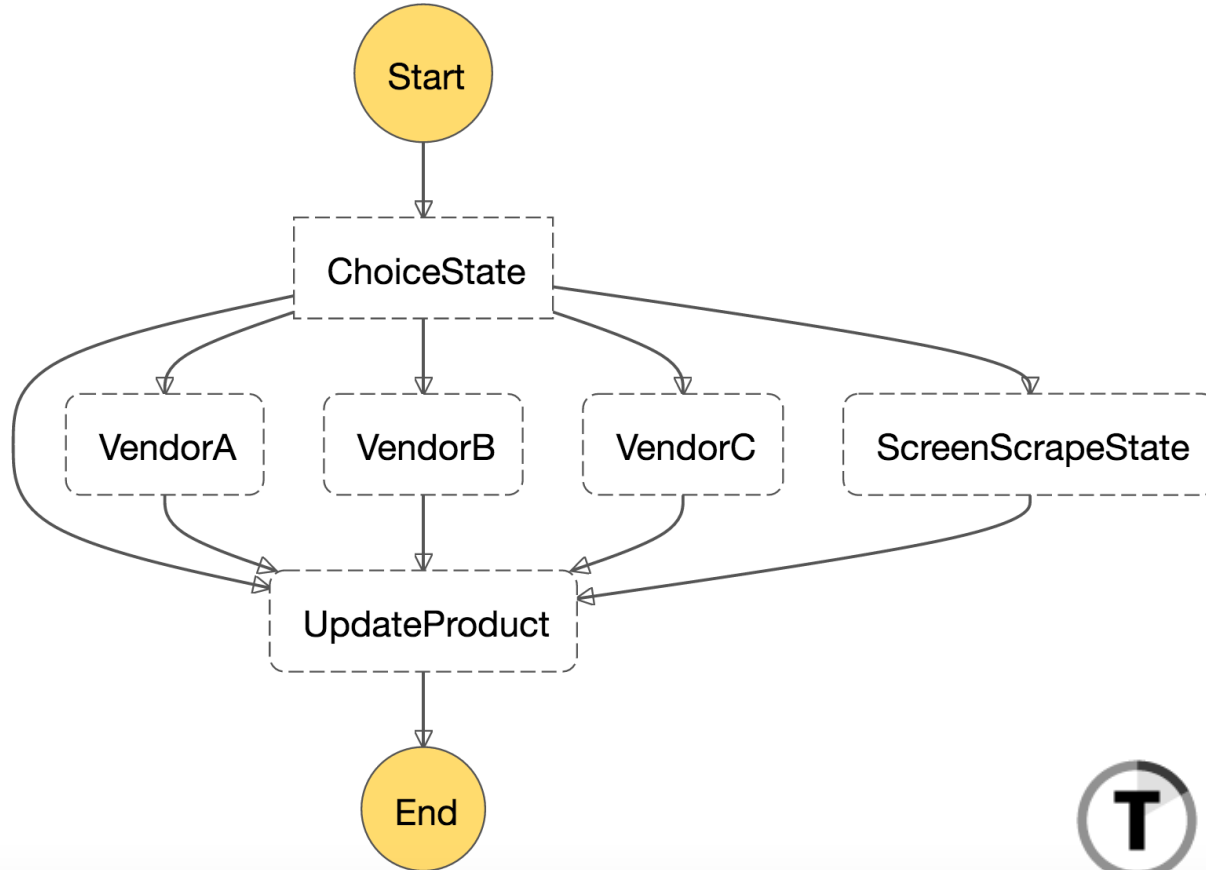
## Reliably Curate Databases



*“AWS Step Functions let us replace a manual product updating process with an automated series of steps, including built-in retry conditions and error handling. We now rely on it to ensure our database and website have the latest price and availability information before the release of a big show, and keep pace with rapidly changing fashions.”*

Jared Browarnik, CTO, TheTake

# Choice State in TheTake's State Machine







OutSystems is the #1 low-code platform  
for building enterprise-grade apps incredibly fast



**Visual Full-Stack  
Development**



**Full Life-Cycle  
Management**



**Deploy to  
Any Device**

# OutSystems' Challenge

A desire to have **intelligence spanning across services**

Goal of consolidating information from different sources

- Custom OutSystems applications
- Third-party monitoring
- Amazon CloudWatch

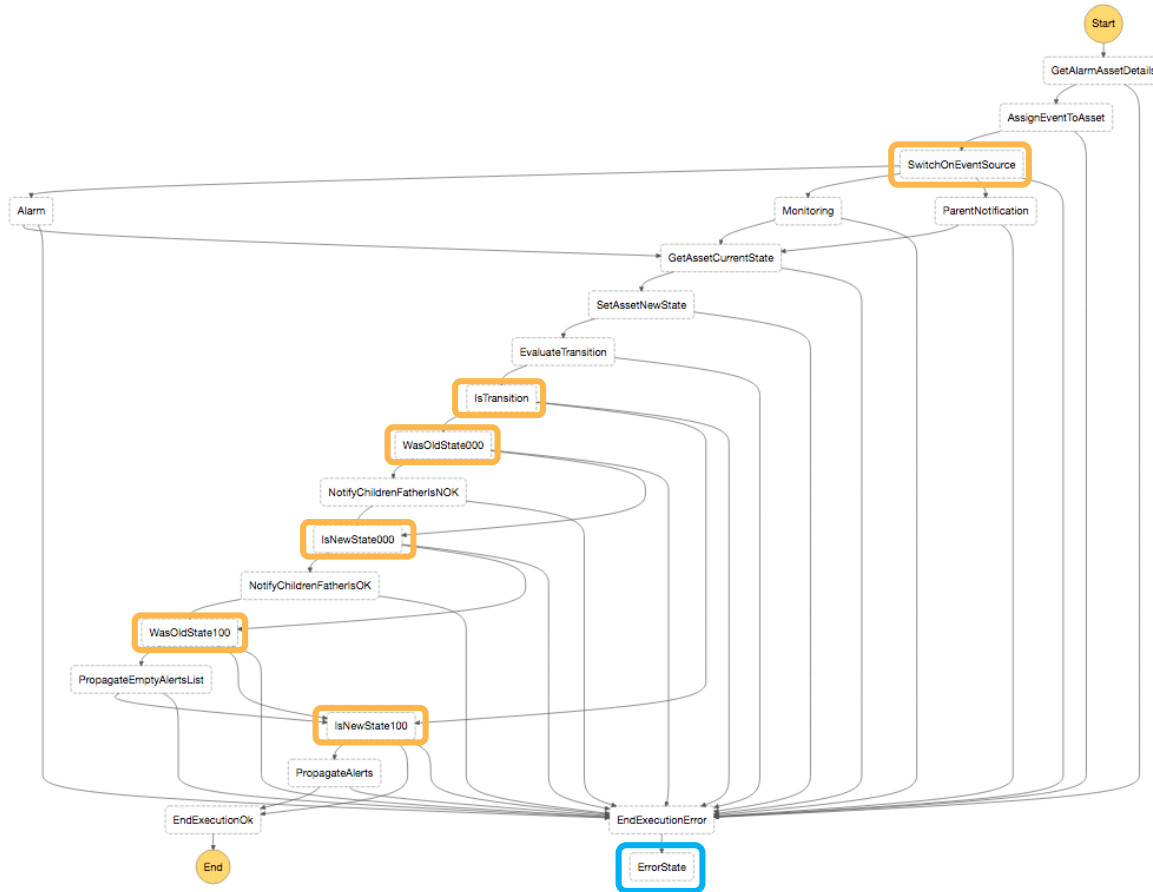
Each focused on business logic at the instance level



# Custom Monitoring Requirements

<b>Reliable</b>	No more "boy who cried wolf"
<b>Scalable</b>	Grows with us
<b>Highly-available</b>	No down time due to OS updates, or other maintenance
<b>Easily extended</b>	Our offer/infrastructure is always evolving

# Alarm State Machine



13 Lambda Task States

6 Choice States

1 Fail State

# First Choice State of the Alarm State Machine

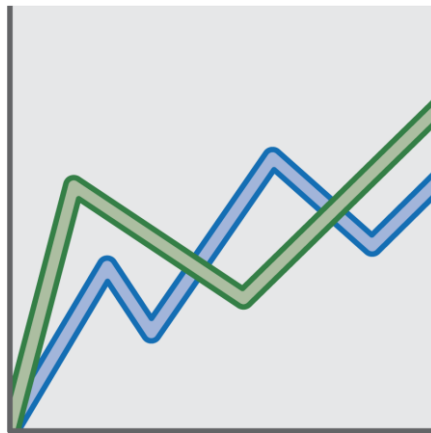
```
"SwitchOnEventSource": {  
  "Type": "Choice",  
  "Choices": [  
    {  
      "Variable": "$.Event.Source",  
      "StringEquals": "Zabbix",  
      "Next": "Alarm"  
    },  
    {  
      "Variable": "$.Event.Source",  
      "StringEquals": "CloudFramework",
```

# Our Experience with AWS Step Functions

**We Earn Trust**



**Supports Our Growth**

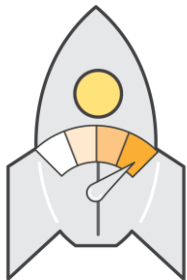


**What Can I Do Today?**



# Features of AWS Step Functions

## Productivity



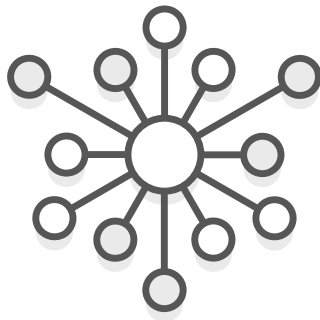
Declarative JSON

Works with AWS and  
on-premises compute

Branching logic

Fork and join tasks

## Agility



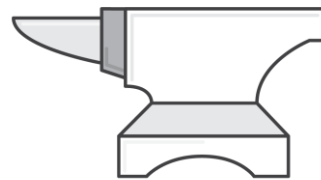
Visual console

Per execution history

Amazon CloudWatch

AWS CloudTrail

## Resilience



Scale automatically

Try-Catch-Finally

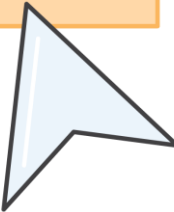

Timeouts and  
heartbeats

# How Much for AWS Step Functions?

**2.5¢ per thousand  
state transitions\***

**\*Free tier of 4,000 state transitions per month**

# Where is AWS Step Functions Available?

Region	Region Code	Launch
US East (N. Virginia)	us-east-1	 
US East (Ohio)	us-east-2	
US West (Oregon)	us-west-2	
EU (Dublin)	eu-west-1	
Asia Pacific (Tokyo)	ap-northeast-1	

# Getting Started

## AWS Step Functions Documentation

AWS Step Functions makes it easy to coordinate the components of distributed applications as a series of steps in a visual workflow. You can quickly build and run state machines to execute the steps of your application in a reliable and scalable fashion.

### [Developer Guide](#)

Describes key concepts of AWS Step Functions and provides instructions for using the features of AWS Step Functions.

[HTML](#) | [PDF](#)

### [Amazon States Language specification](#)

Describes the language that is used to define state machines for AWS Step Functions.

[HTML](#)

### [API Reference](#)

Documents the AWS Step Functions API.

[HTML](#) | [PDF](#)

### [Statelint on Github](#)

A tool to validate your Amazon States Language code.

[Statelint](#)

# aws.amazon.com/step-functions

Dashboard

Tasks

[Dashboard](#) > Create State Machine

Give a name to your state machine

You can now create your own state machine with your own code or choose a blueprint below



Hello World



Wait State



Retry Failure



Parallel

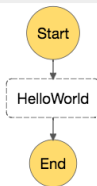


Catch Failure



Choice State

Preview ↺



Code



**AWS  
re:Invent**

**Thank you!**

**[aws.amazon.com/step-functions](https://aws.amazon.com/step-functions)**



**Remember to complete  
your evaluations!**

## Related Sessions

Serverless Apps with AWS Step Functions - SVR201