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

amazon web services

What to Expect from the Session

What is AWS Step Functions?

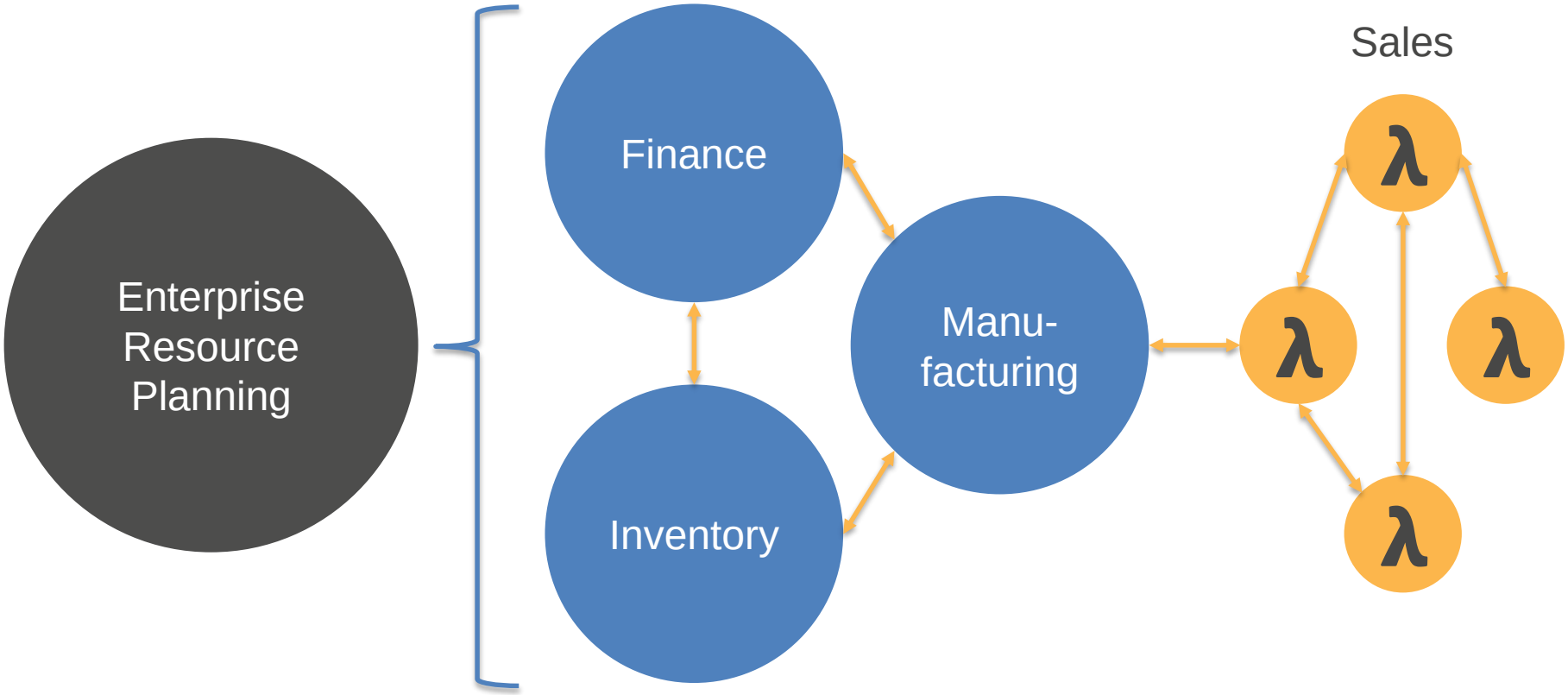
How does it work?

Why should I use it?

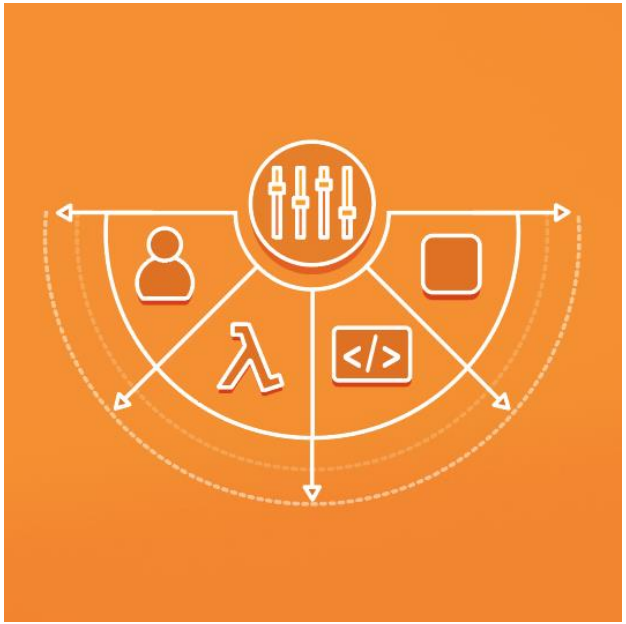
What can I do today?

What is AWS Step Functions?

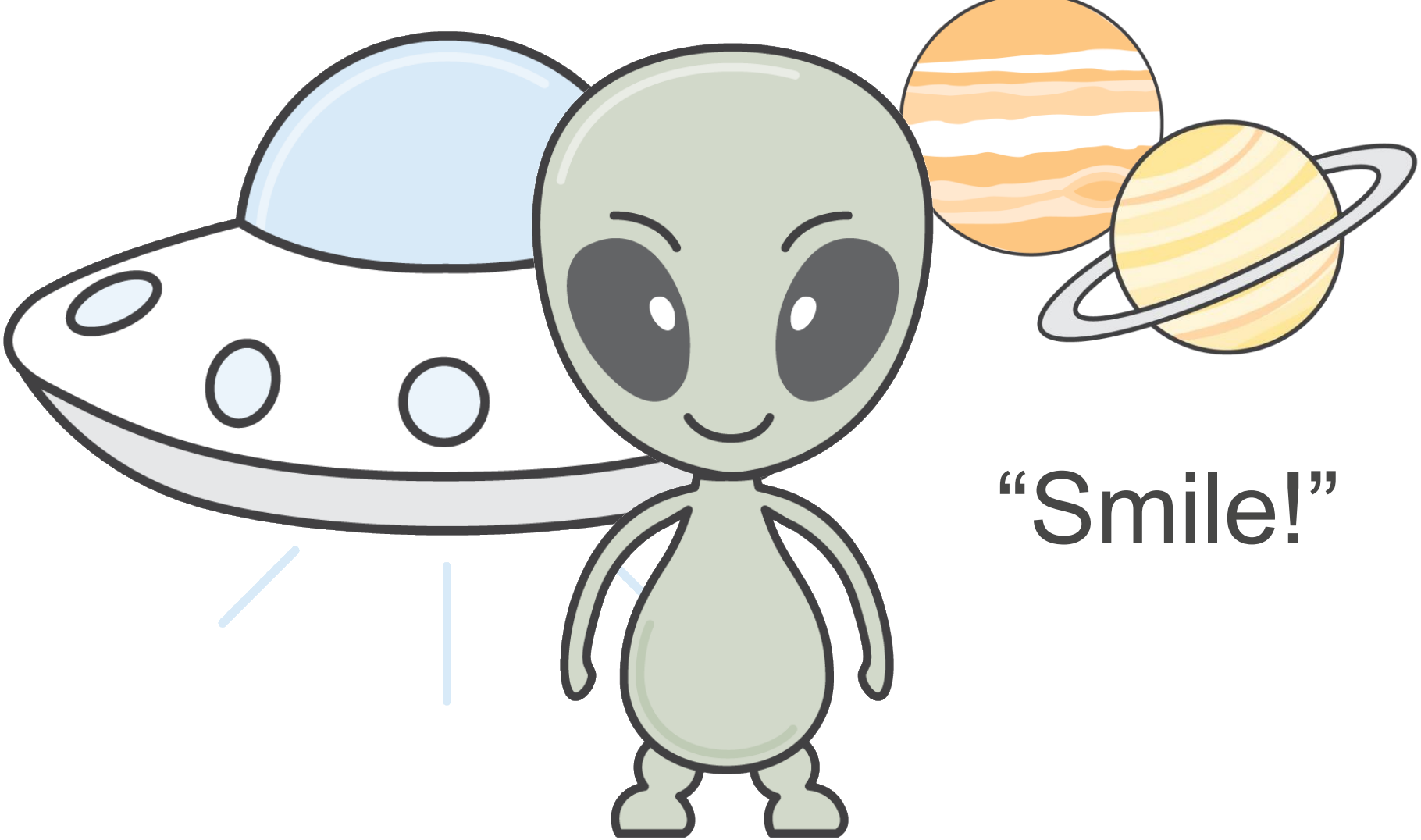
From Monoliths to Microservices to Functions



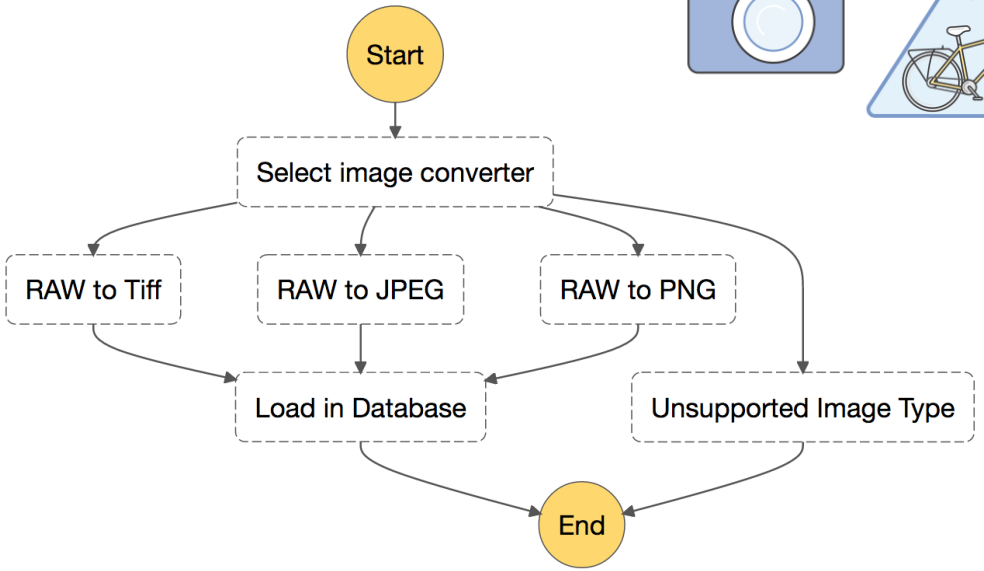
AWS Step Functions…

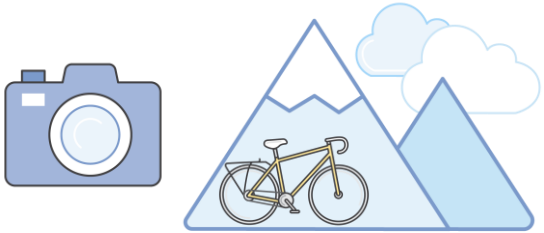


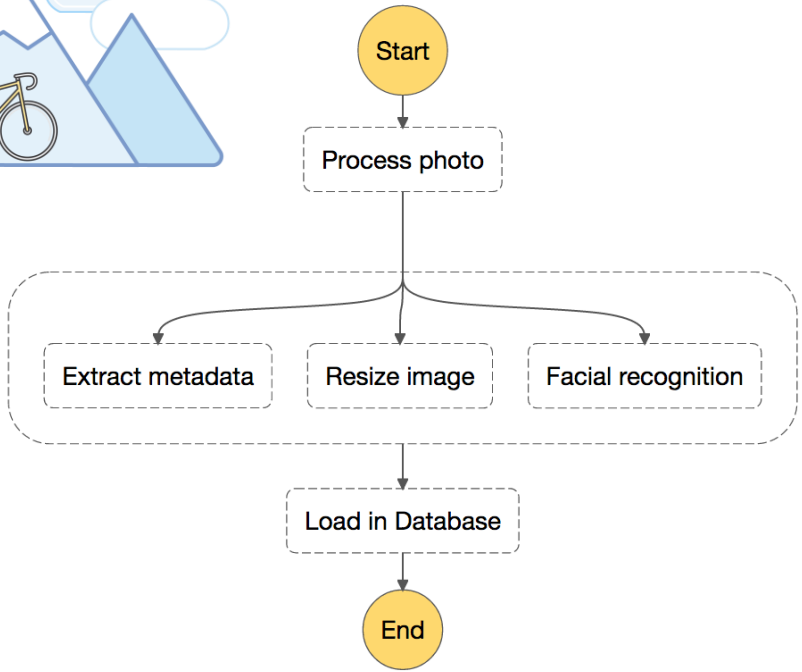
…makes it easy to coordinate the components of distributed applications using visual workflows.


“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”

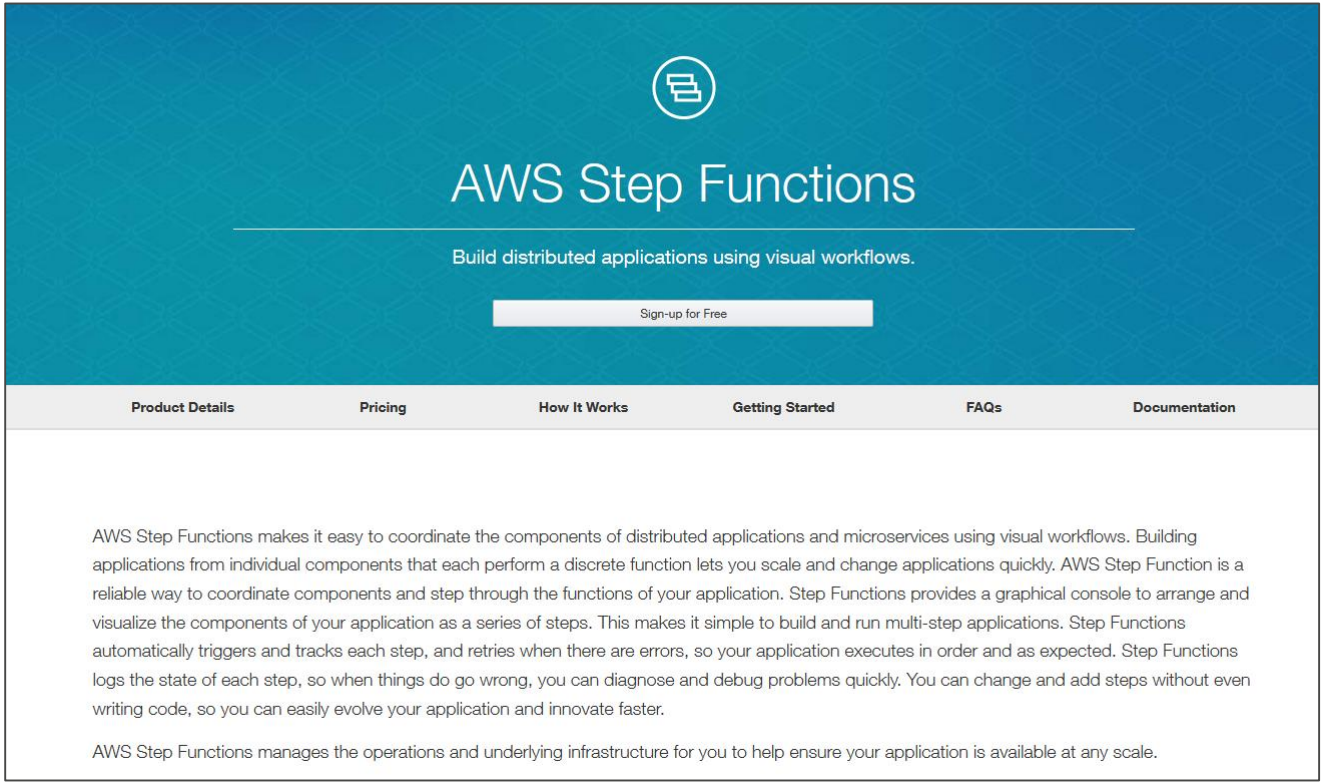
“I have code that runs for hours”

Frequently repeated processes

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

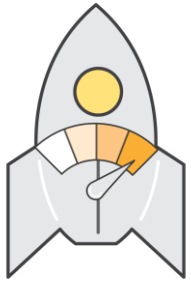
AWS Step Functions is Available Today

aws.amazon.com/step-functions



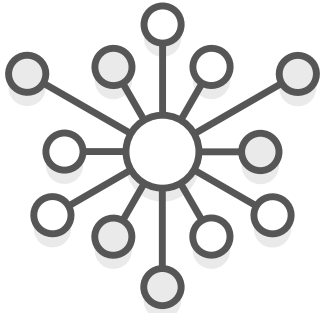
Benefits of AWS Step Functions

Productivity



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

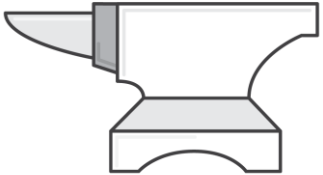
Agility



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?

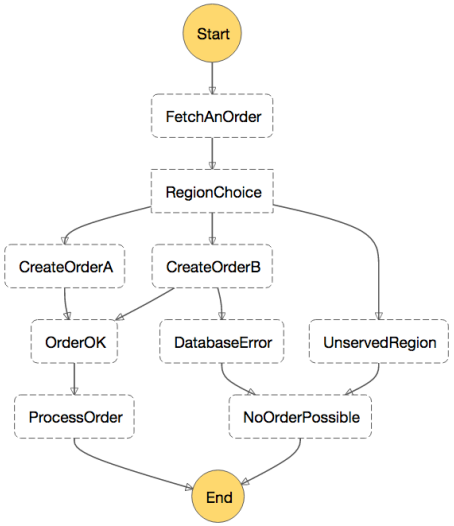
Application Lifecycle in AWS Step Functions

Define in JSON

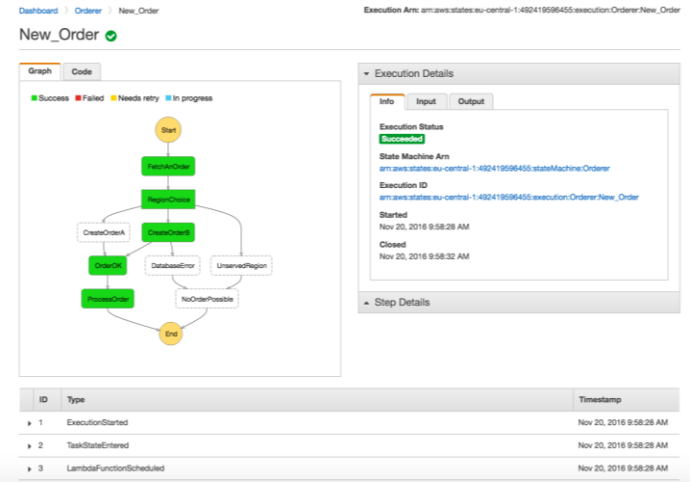
Code

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

Visualize in the Console

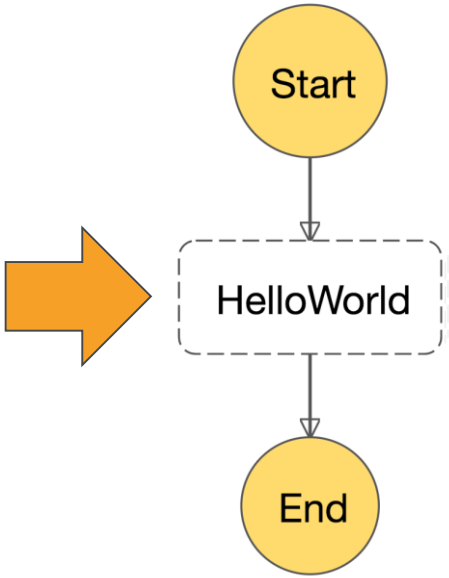


Monitor Executions

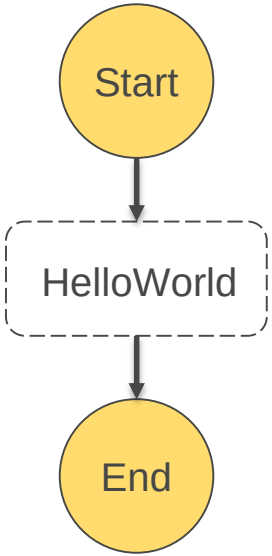


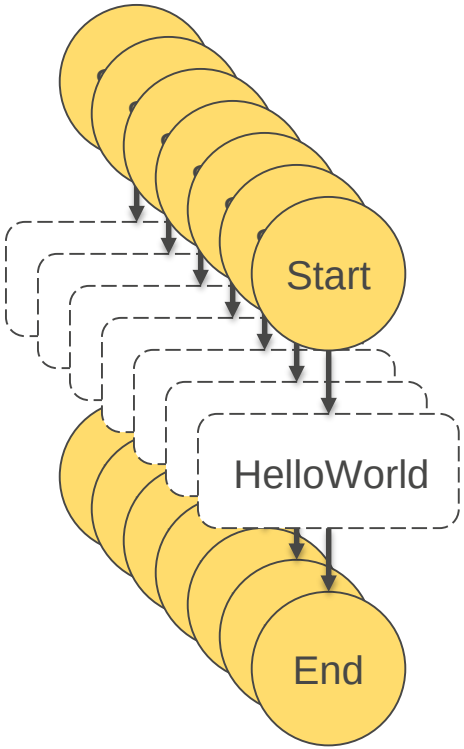
Define in JSON and Then Visualize in the Console

```
{
    ”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

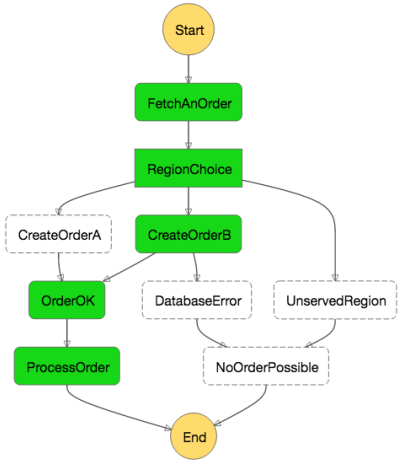
Dashboard 〉 Orderer 〉 New_Order

Execution Arn: arn:aws:states:eu-central-1:[redacted]5:execution:Orderer:New_Order

New_Order

Graph | Code

Success | Failed | Needs retry | In progress



Execution Details

Info | Input | Output

Execution Status
Succeeded

State Machine Arn
arn:aws:states:eu-central-1:[redacted]55:stateMachine:Orderer

Execution ID
arn:aws:states:eu-central-1:[redacted]55:execution:Orderer:New_Order

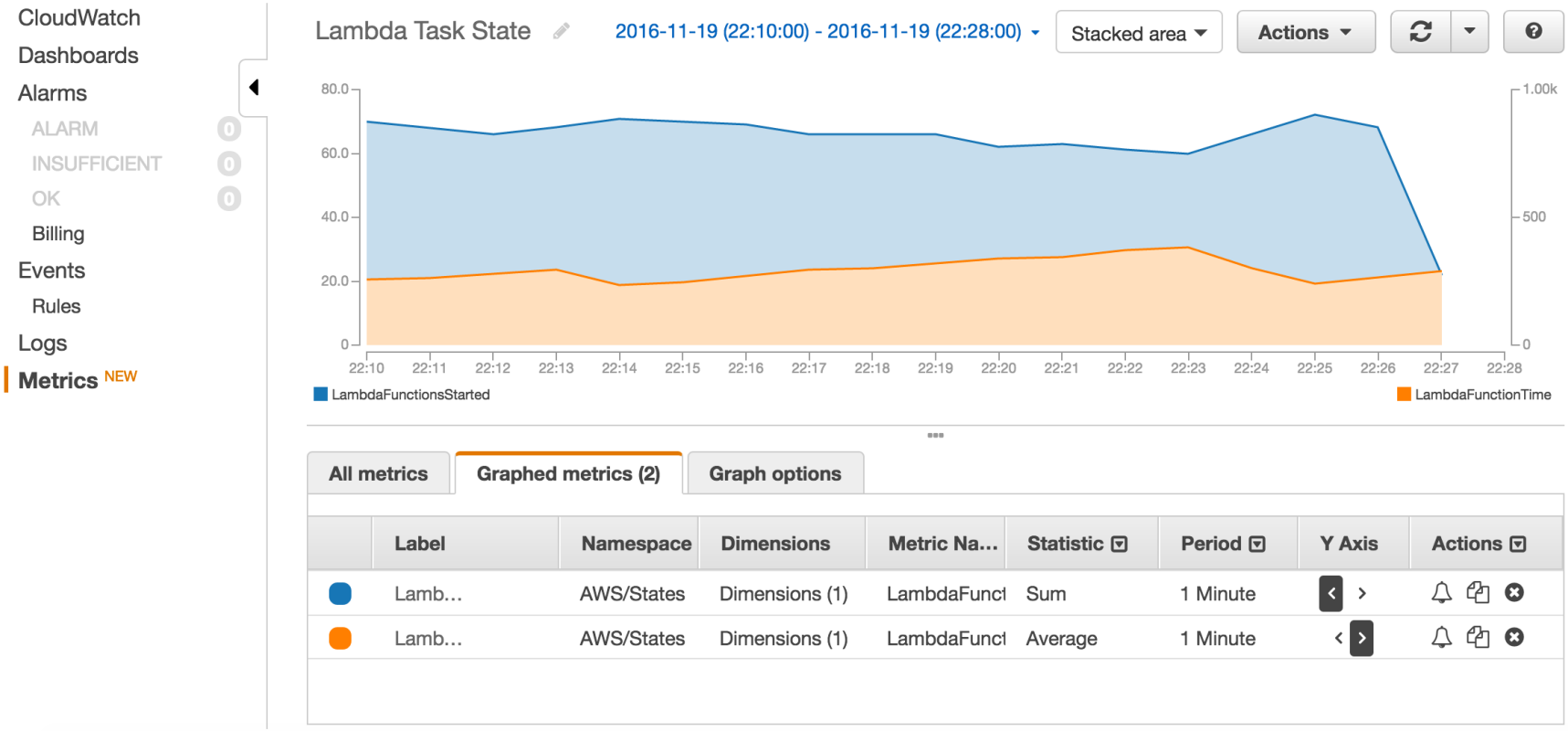
Started
Nov 20, 2016 9:58:28 AM

Closed
Nov 20, 2016 9:58:32 AM

Step Details

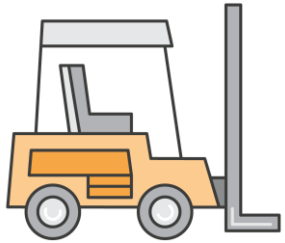
ID	Type	Timestamp
1	ExecutionStarted	Nov 20, 2016 9:58:28 AM
2	TaskStateEntered	Nov 20, 2016 9:58:28 AM
3	LambdaFunctionScheduled	Nov 20, 2016 9:58:28 AM

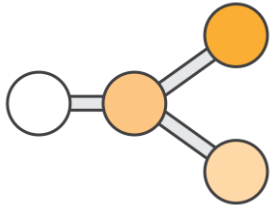
Monitor Executions from Amazon CloudWatch



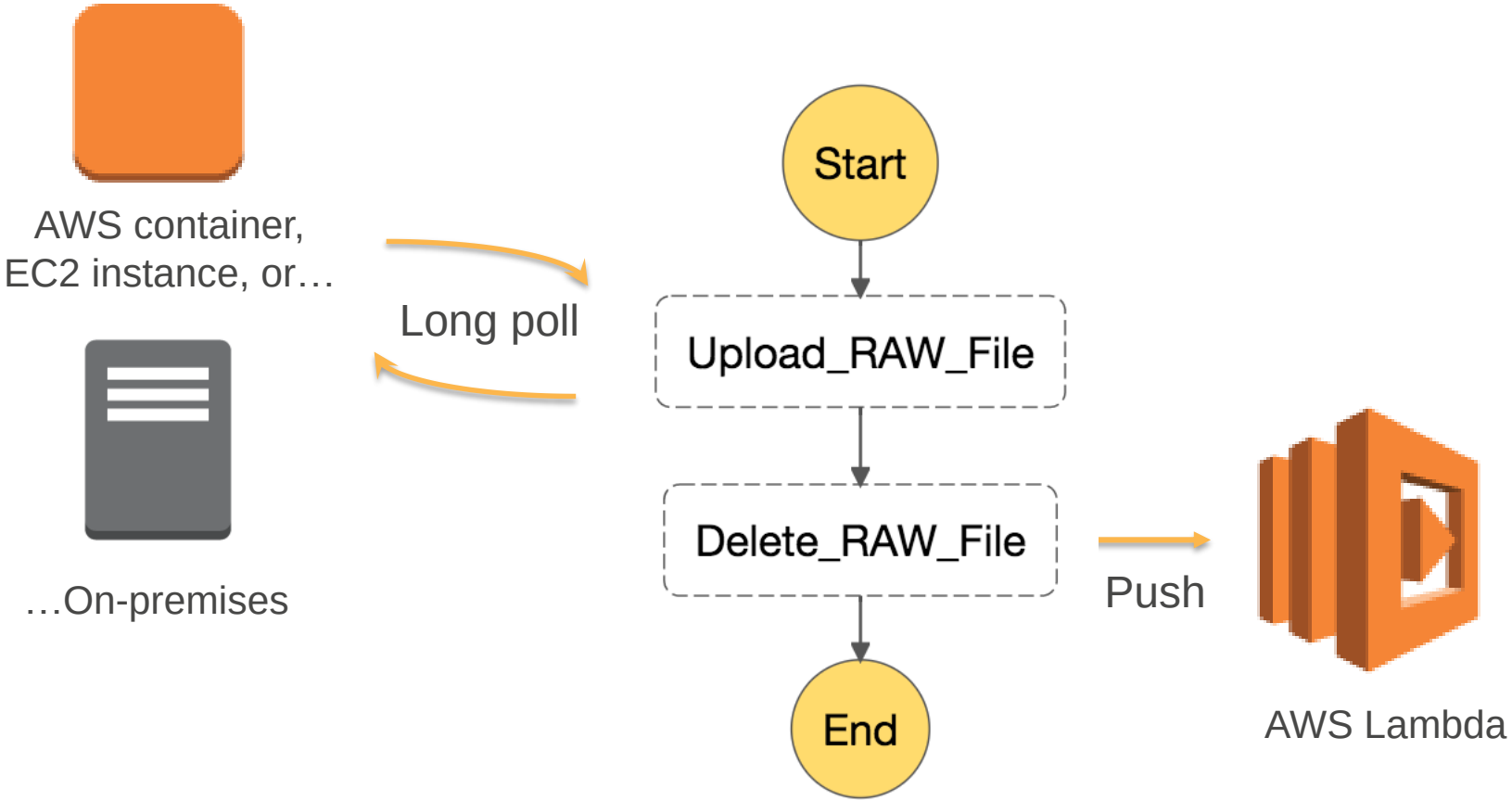
Seven State Types

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



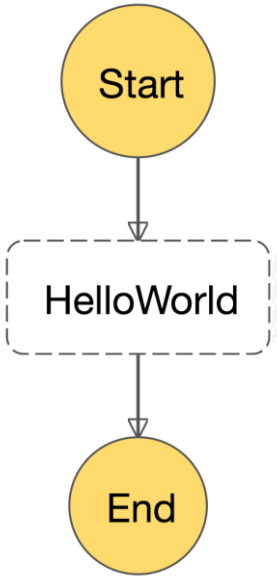


Task States Poll or Push



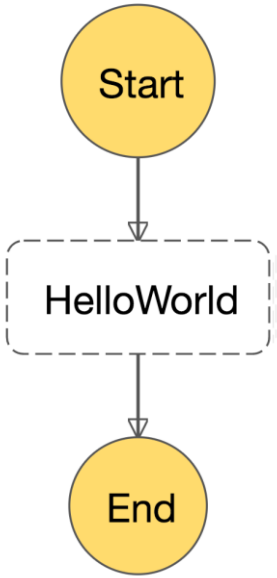
Task State to Dispatch Work

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



Retry Failures of Task States

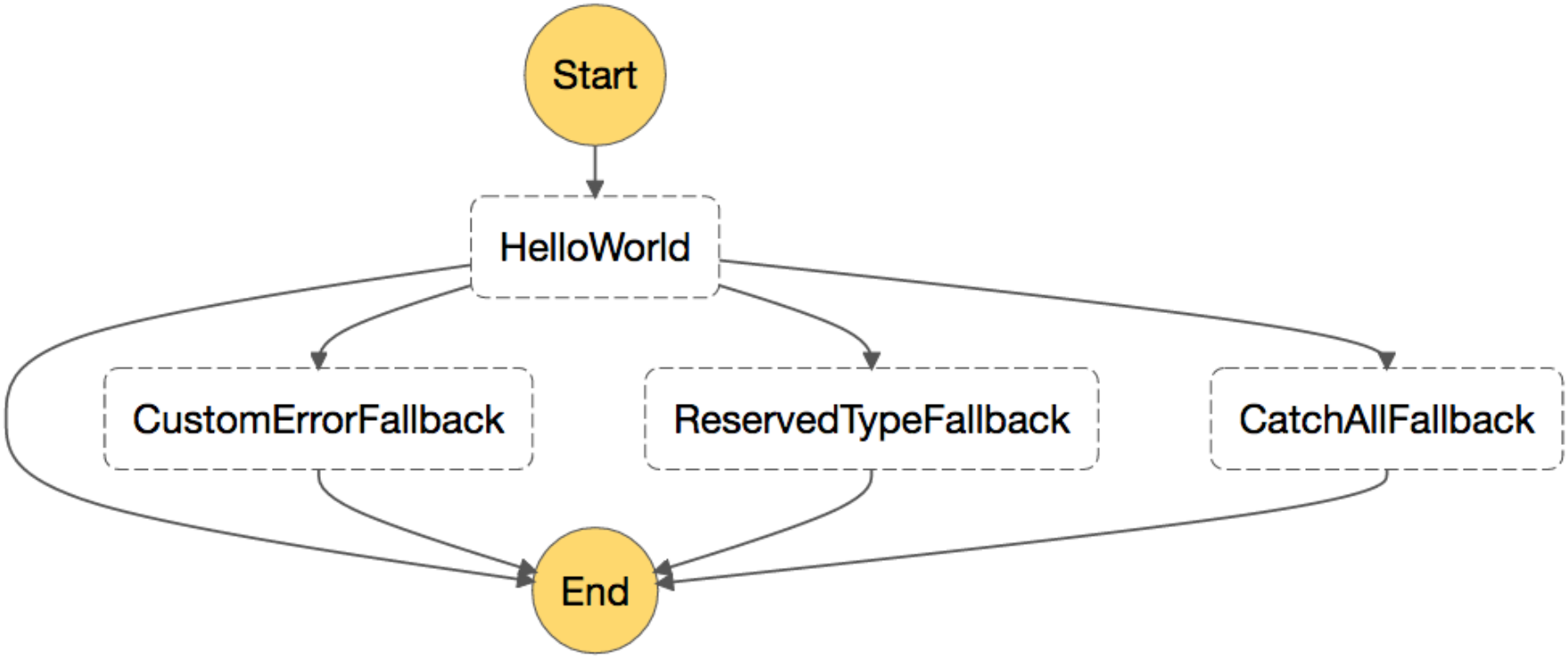
```
"HelloWorld": {
   "Type": "Task",
   …
   "Retry": [
   {
   "ErrorEquals": ["HandledError"],
   "IntervalSeconds": 1,
   "MaxAttempts": 2,
   "BackoffRate": 2.0
   },
   … ]
```



Catch Failure of Task States

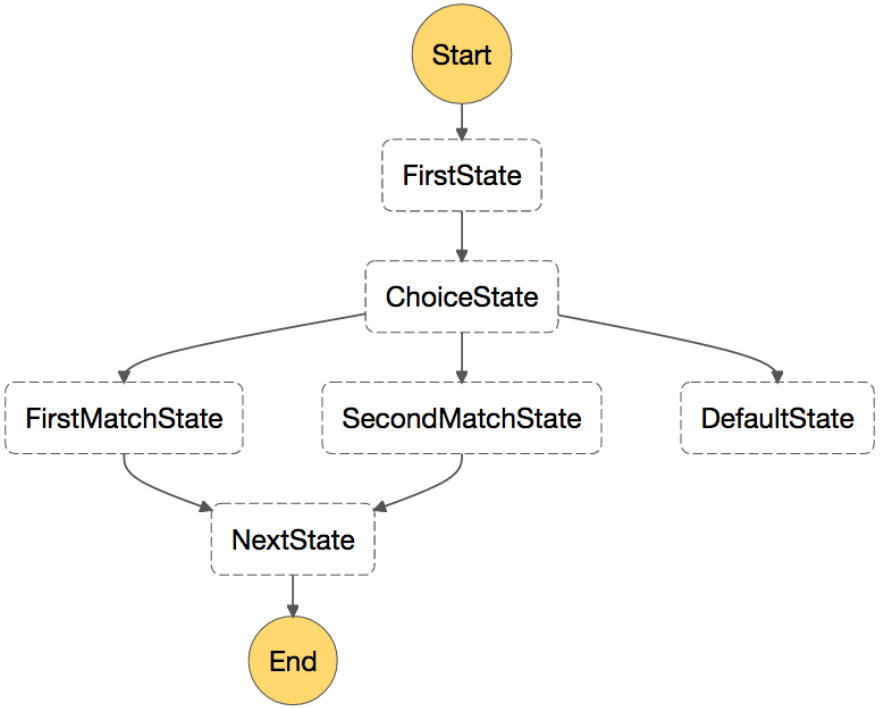
```
"HelloWorld": {
   "Type": "Task",
   …
   "Catch": [
   {
   "ErrorEquals": [”AllYourBasesAreBelongToUs"],
   "Next": "CustomErrorFallback"
   },
   {
   "ErrorEquals": ["States.TaskFailed"],
   "Next": "ReservedTypeFallback"
   … ]
```

Catch Failure of Task State Blueprint



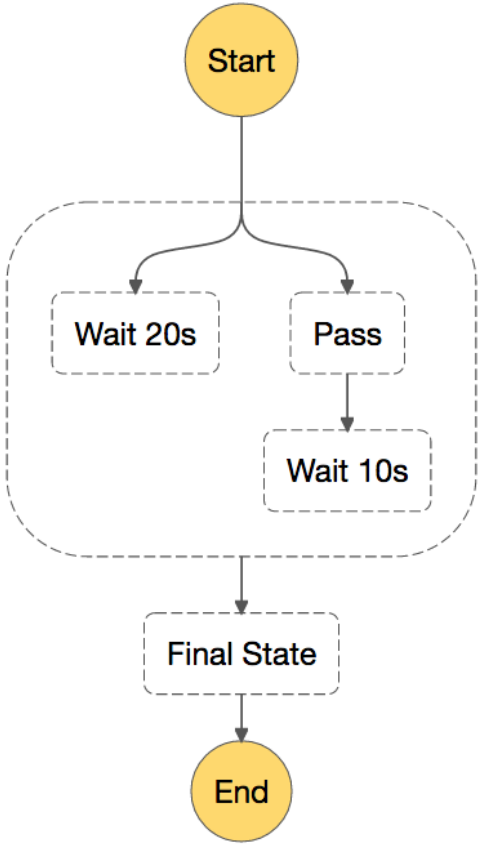
Choice State for Branching Logic

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



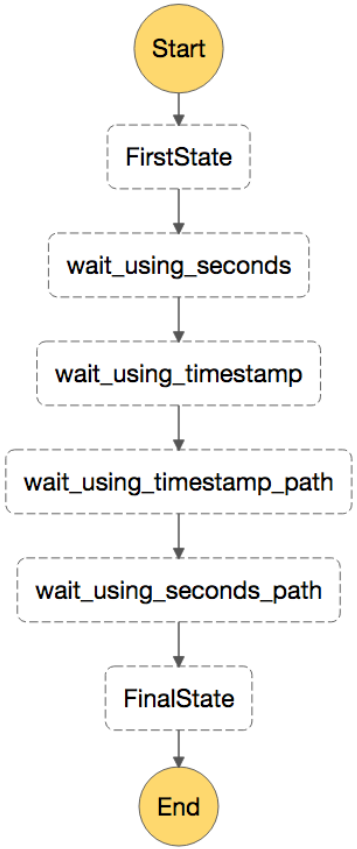
Parallel State to Fork and Join Processes

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

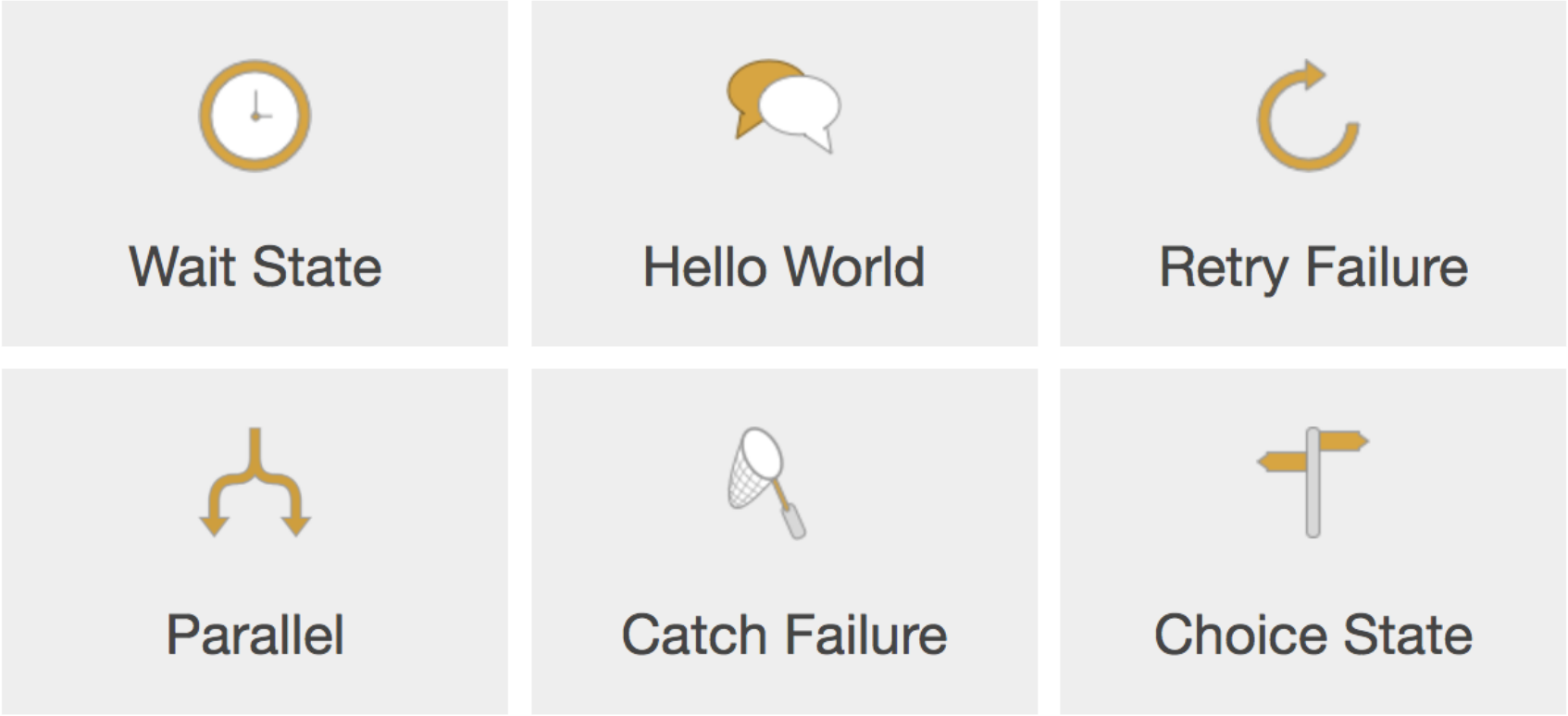


Wait State for Timed Delay

```
"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



Use Through the API

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

Why Should I Use AWS Step Functions?

Ensure Tasks Execute in Sequence

Reliably Process Orders

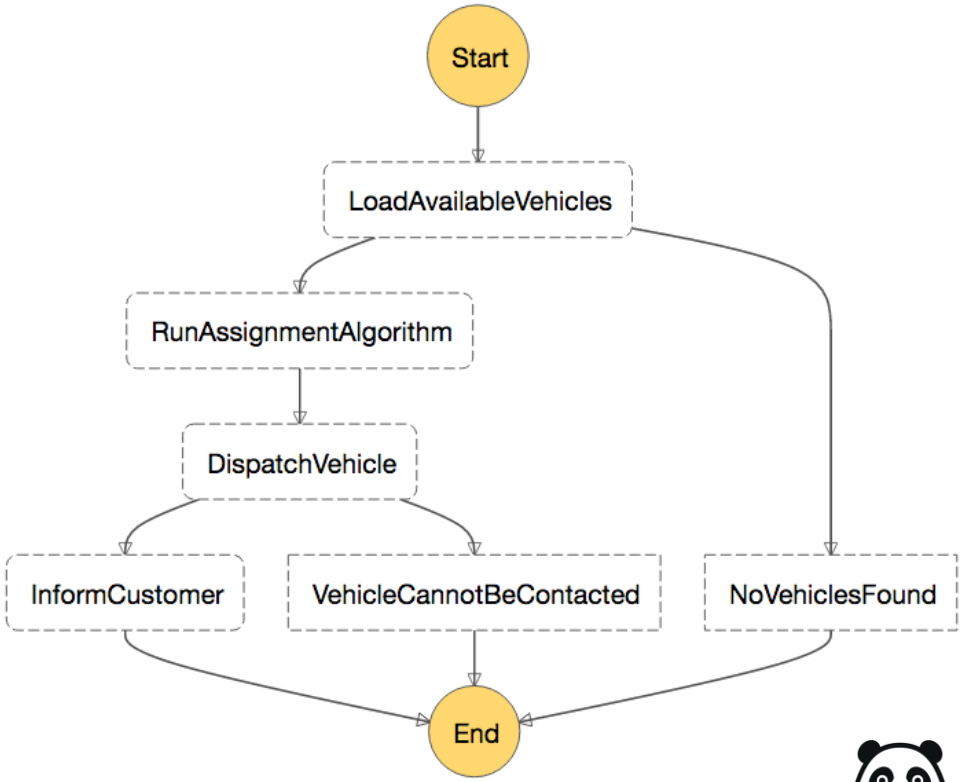
foodpanda

“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



foodpanda

Chose Logical Paths Based on Input Data

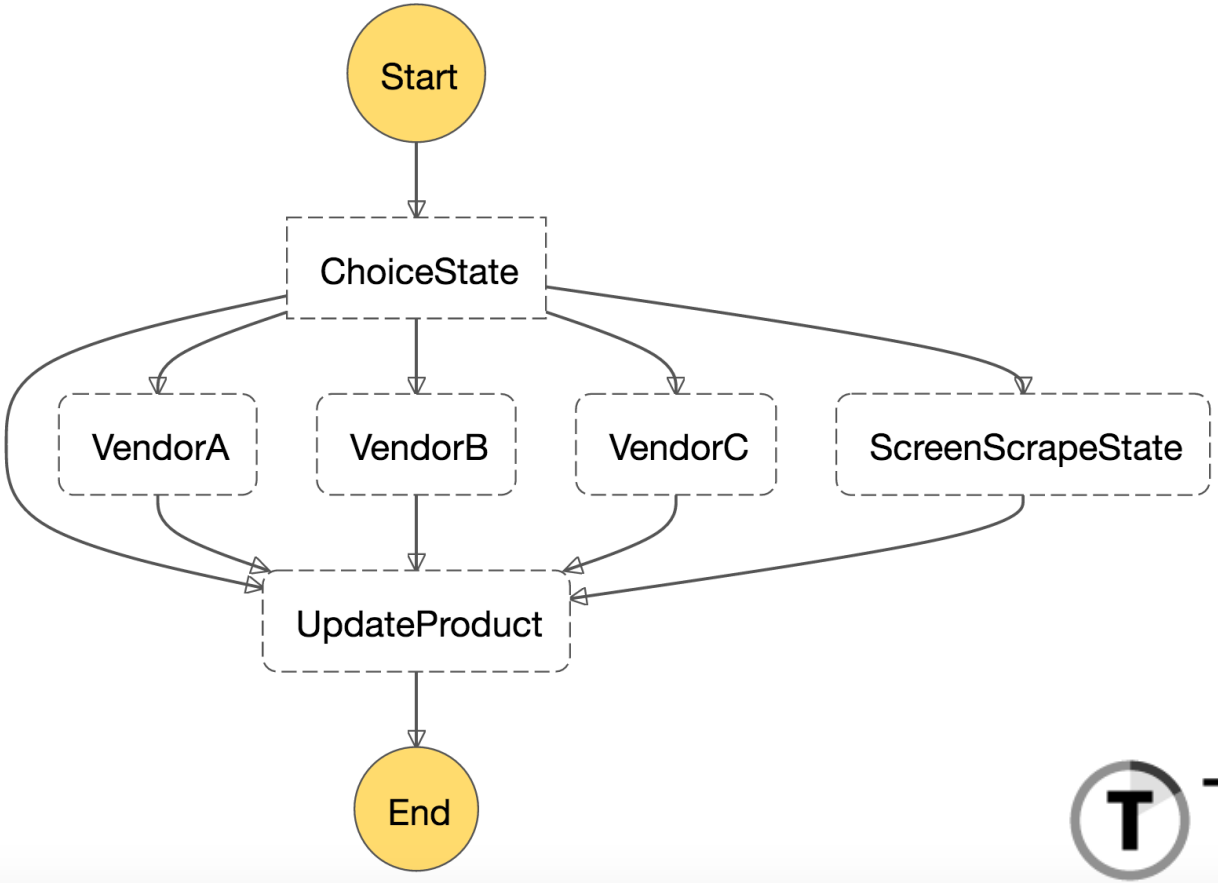
Reliably Curate Databases

THETAKE

"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



THETAKE

outsystems

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

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

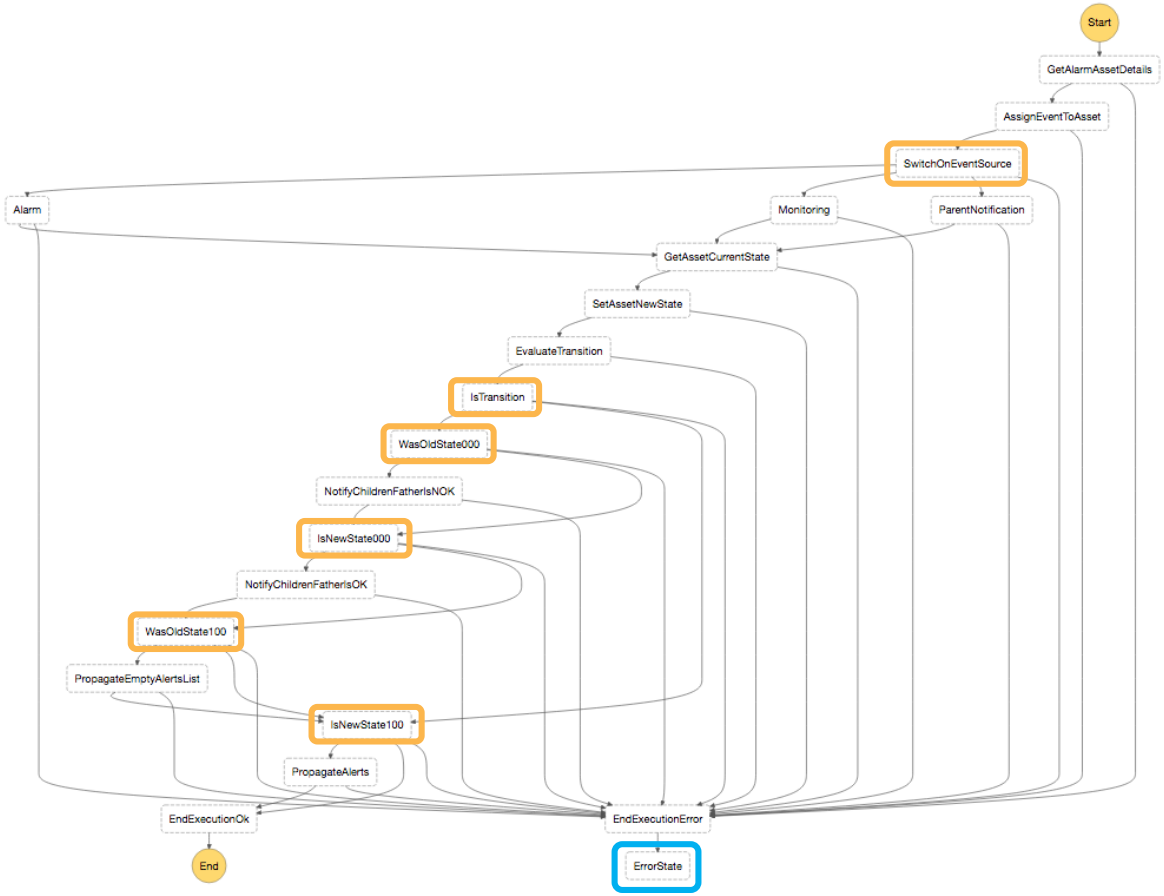
outsystems

Custom Monitoring Requirements

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

outsystems

Alarm State Machine



13 Lambda Task States

6 Choice States

1 Fail State

outsystems

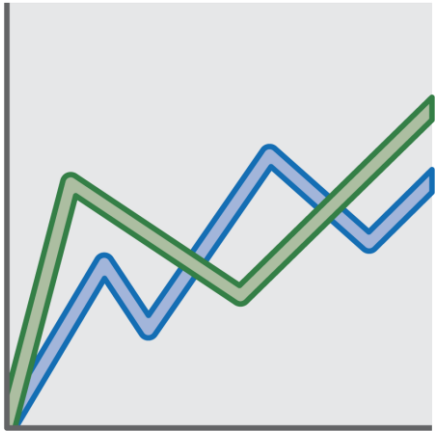
First Choice State of the Alarm State Machine

```
"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

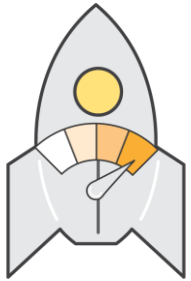


outsystems

What Can I Do Today?

Features of AWS Step Functions

Productivity



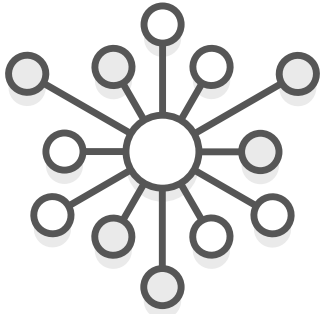
Declarative JSON

Works with AWS and on-premises compute

Branching logic

Fork and join tasks

Agility



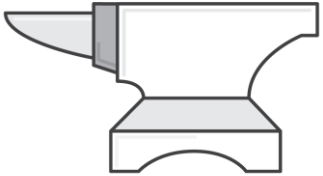
Visual console

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
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

Launch

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.

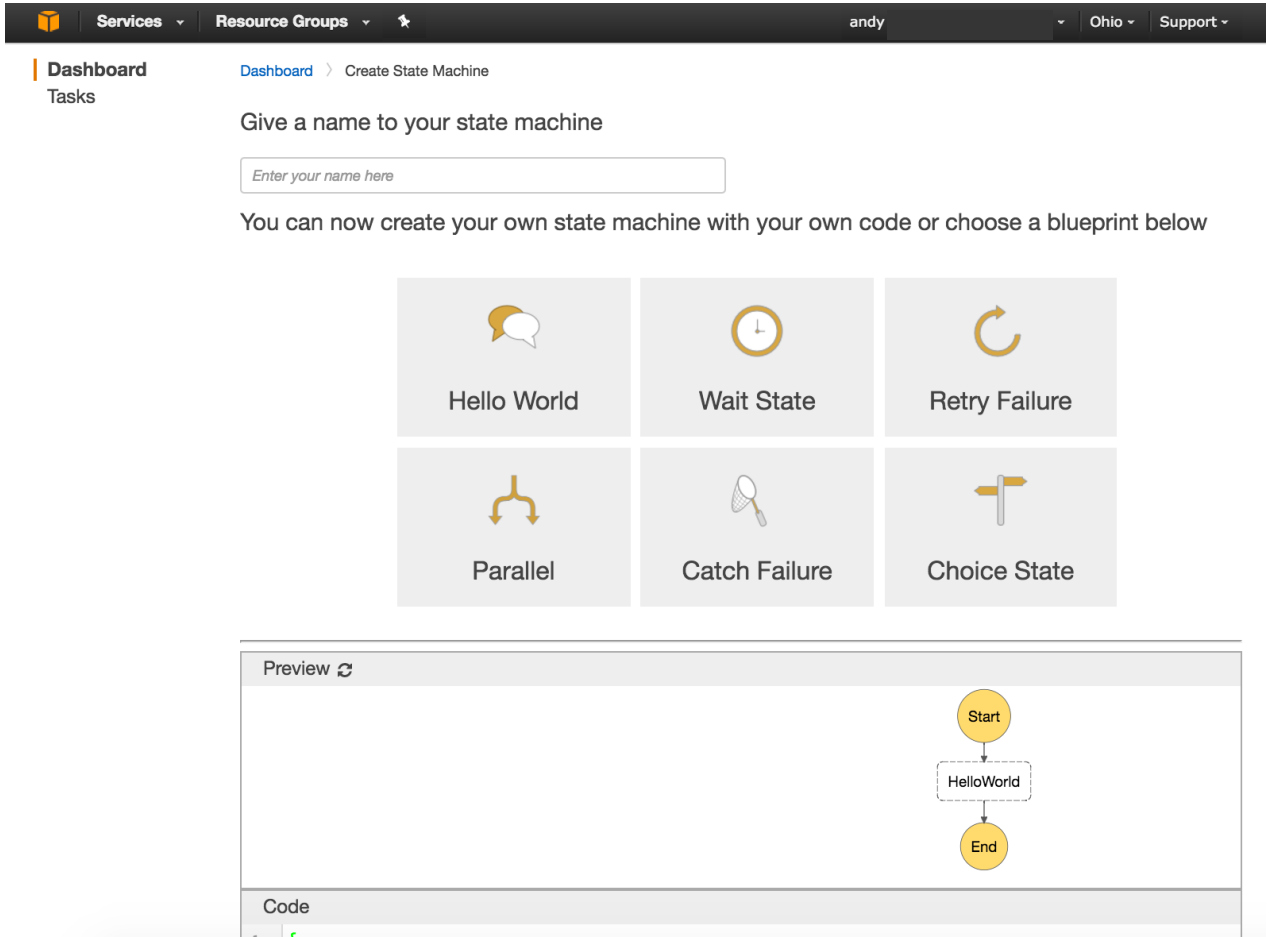
HTML | PDF

Statelint on Github

A tool to validate your Amazon States Language code.

Statelint

aws.amazon.com/step-functions



Thank you!

aws.amazon.com/step-functions

amazon web services

Remember to complete your evaluations!

Related Sessions

Serverless Apps with AWS Step Functions - SVR201