



# Orquestación de Servicios Cloud - Parte I

Por: Cristian David Dallos Bustos

# Agenda



Generalidades



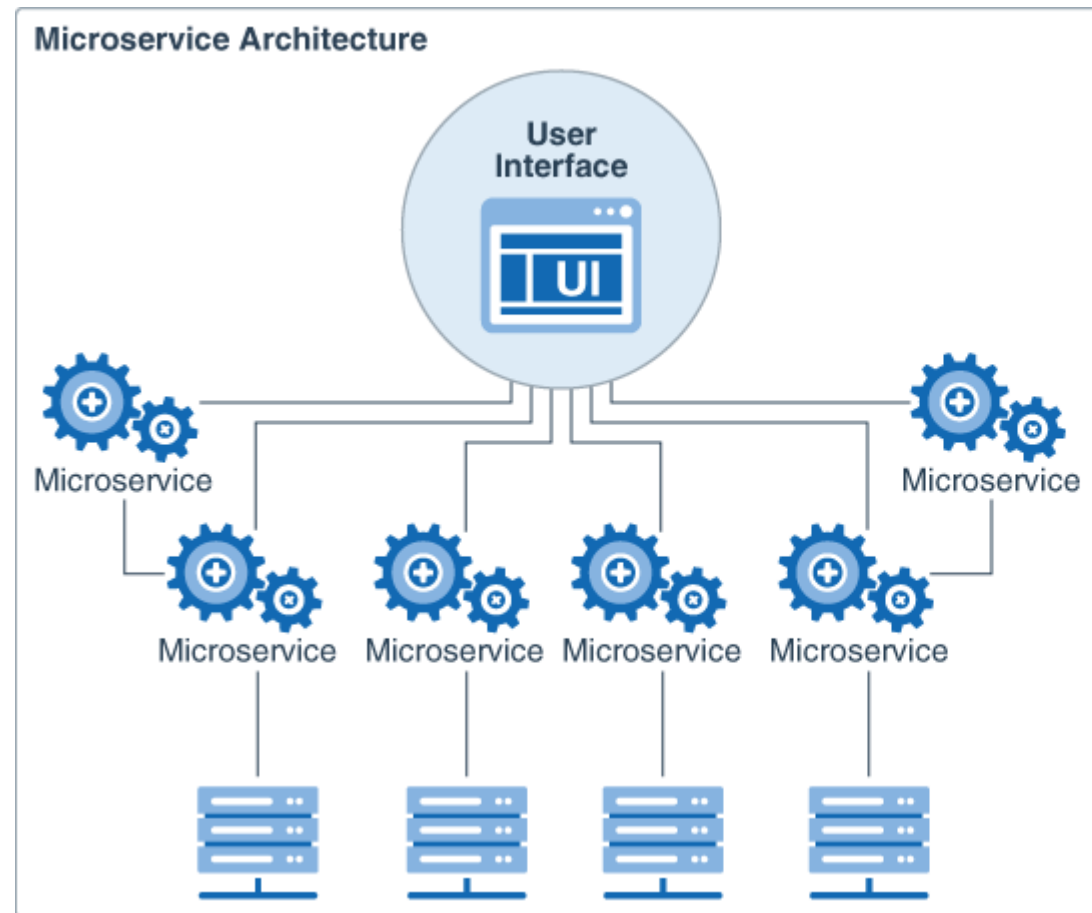
Step Functions



Demo

# Generalidades

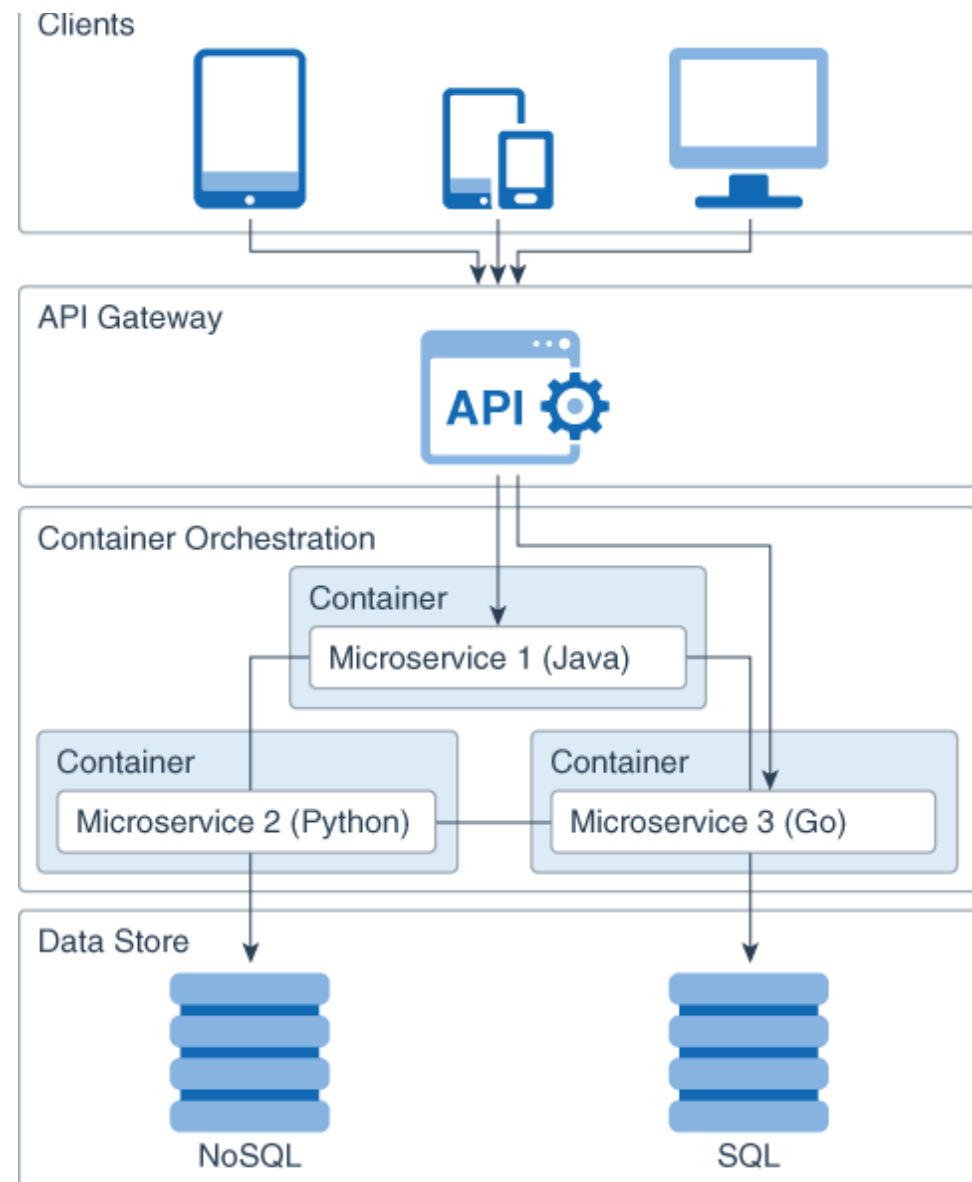
El estilo de arquitectura basado en microservicios no es más que la construcción de aplicaciones de software a partir de una serie de pequeños servicios independientes que se comunican a través de una API bien definida. Cada microservicio se crea para cumplimiento de una única función de negocio, lo cual lo define como un elemento completamente autónomo, entregando libertad para su continua evolución.



# Generalidades

Sin embargo, el cumplir con el principio de única responsabilidad en la definición y construcción del micorsevicio, causa que el cumplir una función de negocio, se vean involucrados dos o mas elementos dispuestos en la solución.

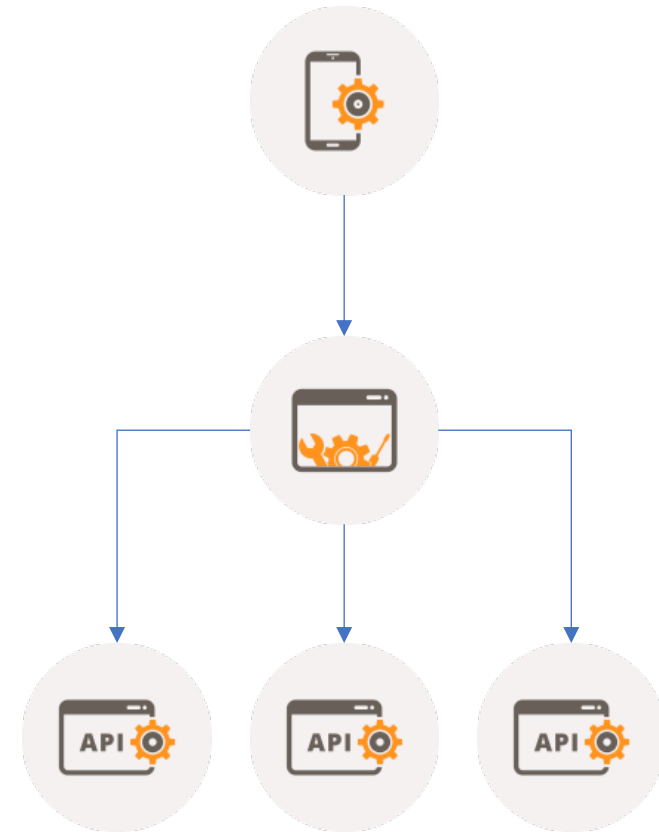
Con base en lo anteriormente descrito, las soluciones de software cuyo estilo arquitectónico se basa en microservicios, se apoya en el los conceptos definidos por la arquitectura SOA conocido como orquestación de servicios o coreografía de servicios.



# Generalidades

## Orquestacion de servicios

La orquestación de servicios implica controlar todos los elementos e interacciones desde un punto central de manera equivalente a lo que realiza un director de orquesta, es decir, cada elemento deberá esperar la intrucción del director para ejecutar sus acciones. En una orquestación de servicios, un controlador manejará todas la comunicaciones y acciones entre cada microservicio.

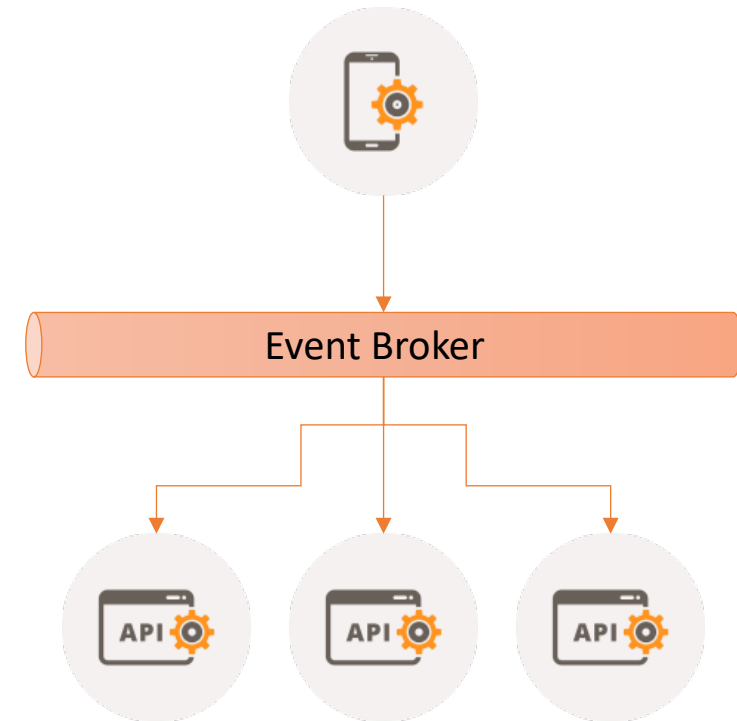


# Generalidades

## Coreografía de servicios

Así como tenemos el director de orquesta como similar en la orquestación de servicios, en la coreografía podemos encontrar su equivalente en un grupo de danza, en donde cada integrante, conoce lo que debe hacer y en que momento ejecutarlo.

Para que exista una coreografía de servicios, la solución deberá contar con un mecanismo que permita el intercambio de mensajes. En el momento en que un microservicio envía un mensaje, la tarea estará finalizada, es decir, todo lo que venga después, se dará de forma asíncrona, sin esperar respuesta ni preocupándose por lo que venga a continuación.





# Generalidades

Desafíos de las orquestaciones o coreografía

- Recuperación ante fallos
  - Disponibilidad
  - Escalabilidad
  - Observabilidad
-



## Orquestando servicios en AWS Usando el servicio Step Functions

Step Functions es un servicio de orquestación tipo serverless (es decir, las tareas de gestión de infraestructura tales como configuración de capacidad o parches de software, serán gestionados por AWS así como la configuración de autoescalado y alta disponibilidad) que permite combinar diferentes servicios de AWS para con ello crear aplicaciones críticas para el negocio.

- Lambda
  - DynamoDB
  - Amazon ECS / AWS Fargate
  - Amazon SNS
  - Amazon SQS
  - AWS Glue
  - SageMaker
  - Amazon EMR
  - CodeBuild
  - AWS Step Functions
-



# Orquestando servicios en AWS

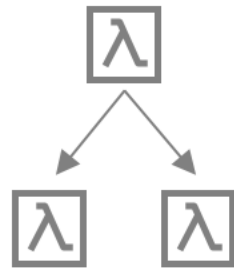
## Usando el servicio Step Functions

### Casos de Uso

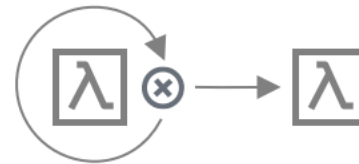
Caso de uso n.o 1: Orquestación de funciones



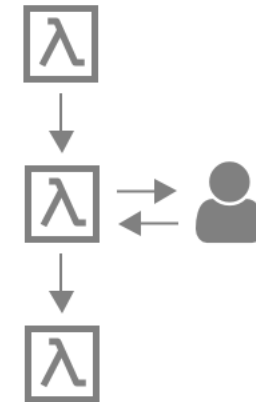
Caso de uso n.o 2: Derivación



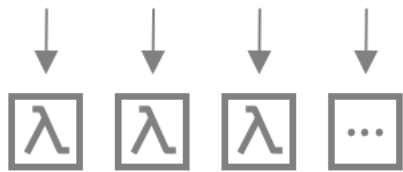
útil n.o 3: Control de errores



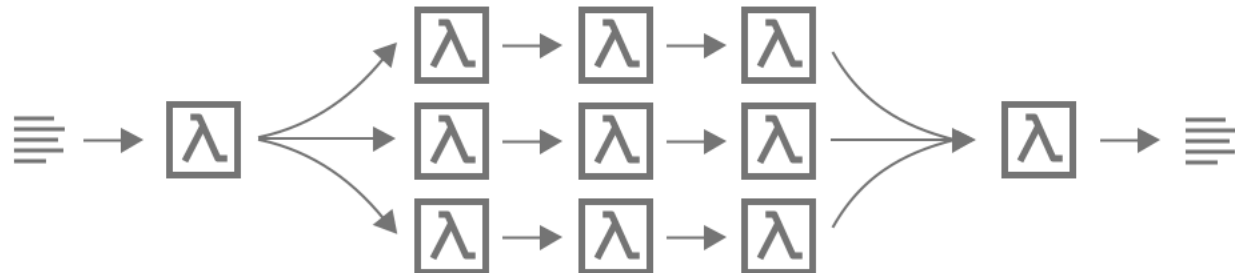
Caso de uso n.o 4: El ser humano en el bucle



útil n.o 5: Procesamiento paralelo



útil n.o 6: Paralelismo dinámico



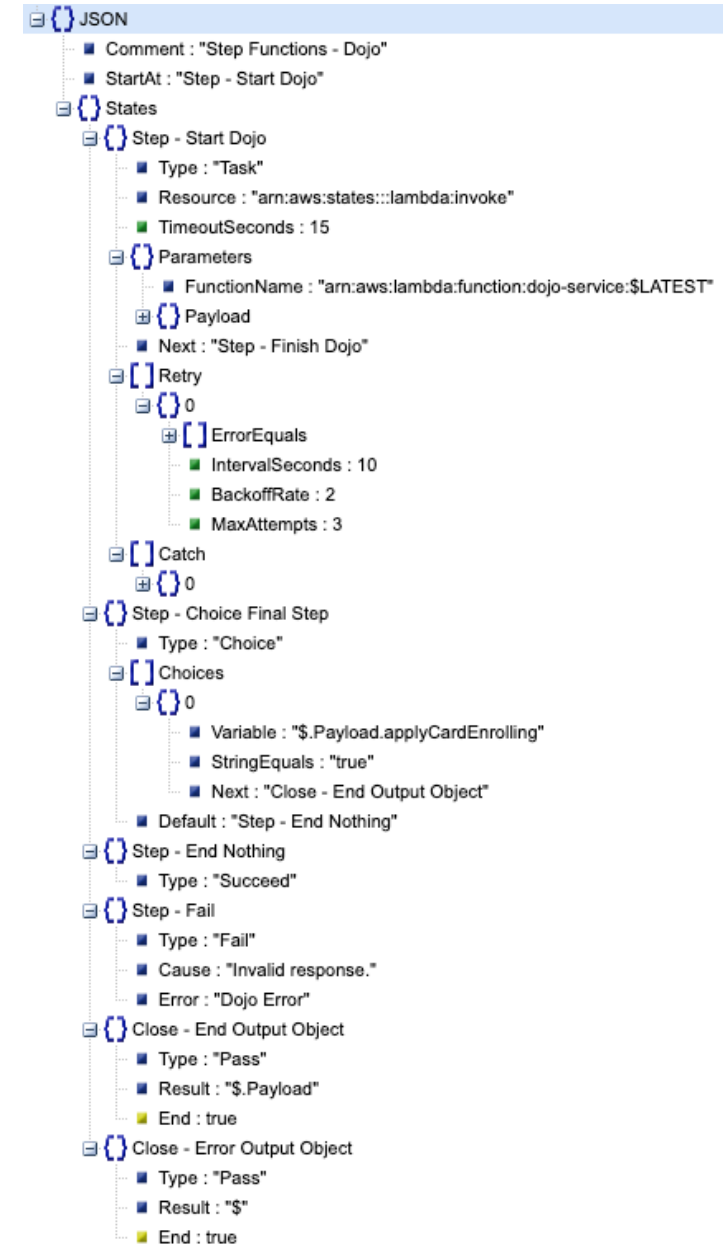
# Orquestando servicios en AWS

## Usando el servicio Step Functions

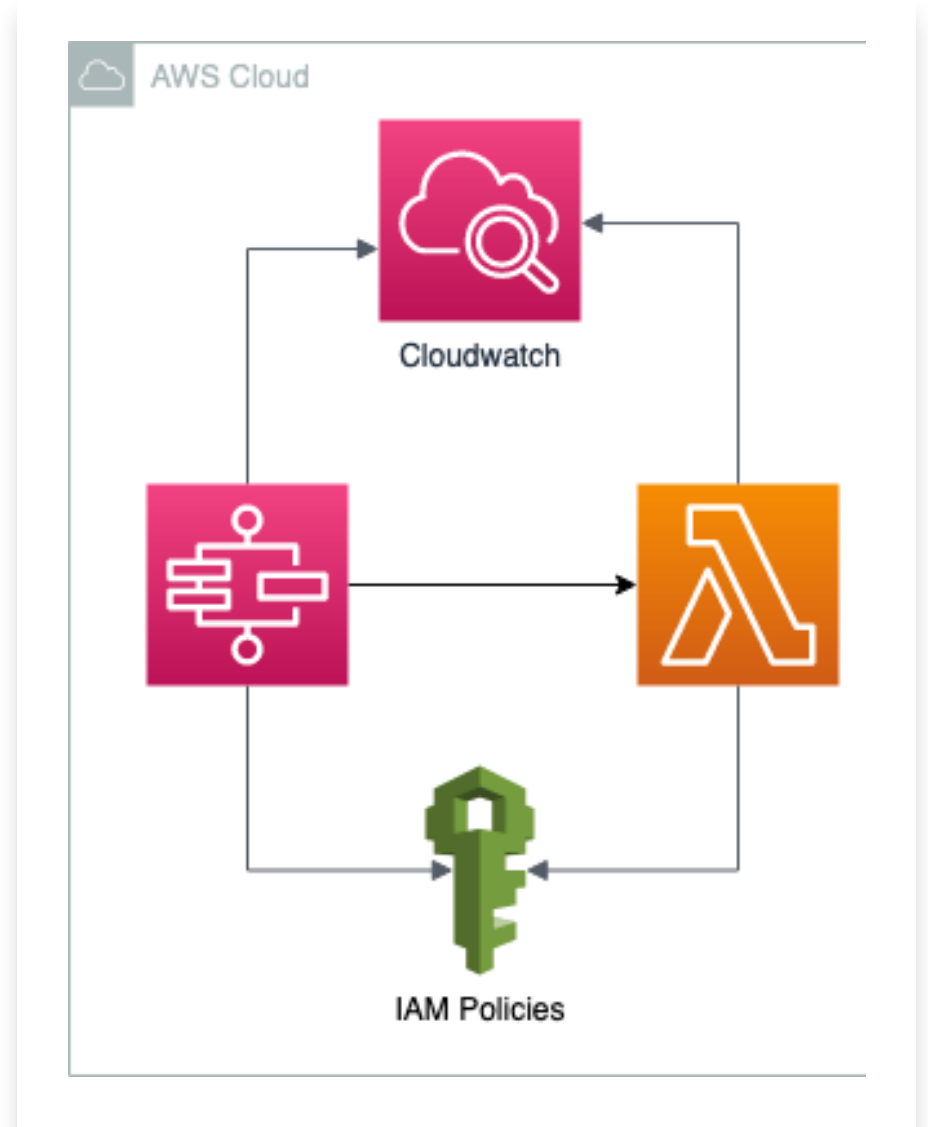
### Secciones

- Estados: Lenguaje estructurado basado en JSON. Se utiliza para definir máquinas de estado, que corresponden a conjuntos de estado (tareas), decisiones (choice), estados de error (fail), etc.

- Task
  - Resource
  - Parameters
  - Payload
  - Retry
  - Catch
- Choice
- Wait
- Succeed
- Fail
- Parallel
- Map



# Demo Arquitectura



# Demo

## 1. Crear Rol Para La Lambda

- Abrir el servicio IAM en la consola AWS.
- Seleccionar en el menú lateral izquierdo la opción roles.
- Clic en crear role.
- En las opciones desplegadas, seleccionamos AWS Service y a continuación, Lambda.
- Filtramos la opción “AWSLambdaBasicExecutionRole”.
- Ingresamos algunos tags.
- Ingresamos el nombre que consideres adecuado para el rol en creación.

### Create role

1 2 3 4

#### Review

Provide the required information below and review this role before you create it.

**Role name\*** role-lambda-basic-execution-dojo

Use alphanumeric and '+=,@-\_' characters. Maximum 64 characters.

**Role description** Allows Lambda functions to call AWS services on your behalf.

Maximum 1000 characters. Use alphanumeric and '+=,@-\_' characters.

**Trusted entities** AWS service: lambda.amazonaws.com

**Policies**  AWSLambdaBasicExecutionRole [↗](#)

**Permissions boundary** Permissions boundary is not set

The new role will receive the following tag

Key	Value
role	lambda invoke

# Demo

## 2. Crear Lambda Function

- Abrir el servicio Lambda en la consola AWS y luego hacemos clic en la opción “Crear Función”.
- Seleccionamos la opción “Author from scratch (crear desde cero)”.
- En la sección de información básica asignamos:
  - Function Name: StepFunctionsDojoLambda
  - Runtime: Node.js 12.x
  - Rol: Seleccionamos el rol previamente creado (role-lambda-basic-execution-dojo).

aws Services ▾

Lambda > Functions > Create function

### Create function [Info](#)

Choose one of the following options to create your function.

**Author from scratch** ☒

Start with a simple Hello World example.

**Use a blueprint**

Build a Lambda application from sample code and config common use cases.

#### Basic information

**Function name**  
Enter a name that describes the purpose of your function.

StepFunctionsDojoLambda

Use only letters, numbers, hyphens, or underscores with no spaces.

**Runtime** [Info](#)  
Choose the language to use to write your function.

Node.js 12.x

**Permissions** [Info](#)  
By default, Lambda will create an execution role with permissions to upload logs to Amazon CloudWatch Logs. You can customize this default role later when adding triggers.

▼ Change default execution role

**Execution role**  
Choose a role that defines the permissions of your function. To create a custom role, go to the [IAM console](#).

☐ Create a new role with basic Lambda permissions

☒ Use an existing role

☐ Create a new role from AWS policy templates

**Existing role**  
Choose an existing role that you've created to be used with this Lambda function. The role must have permission to upload logs to Amazon CloudWatch Logs.

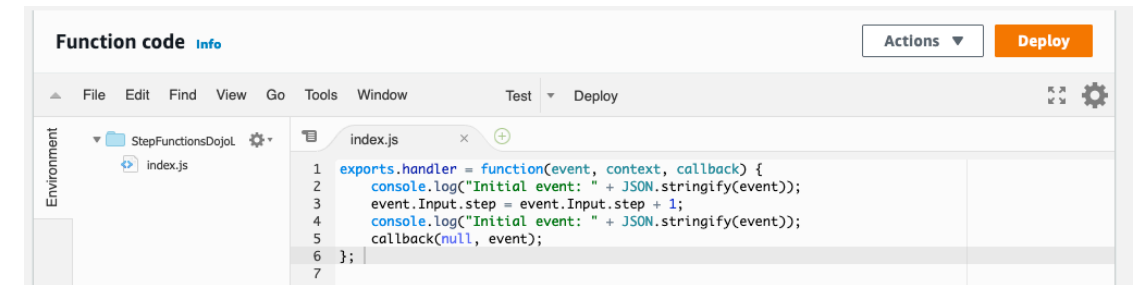
role-lambda-basic-execution-dojo

[View the role-lambda-basic-execution-dojo role](#) on the IAM console.

# Demo

## 3. Actualizar Lambda Function

- Ubicamos la sección “Function Code” y procedemos a pegar el fragmento de código ubicado en el archivo StepFunctionsDojoLambda.js disponible en: <https://github.com/Tutorial-Labs/step-functions-demo-01/blob/main/StepFunctionsDojoLambda.js>.
- Una vez se ha actualizado el código, procedemos a dar clic en la opción “Deploy”.

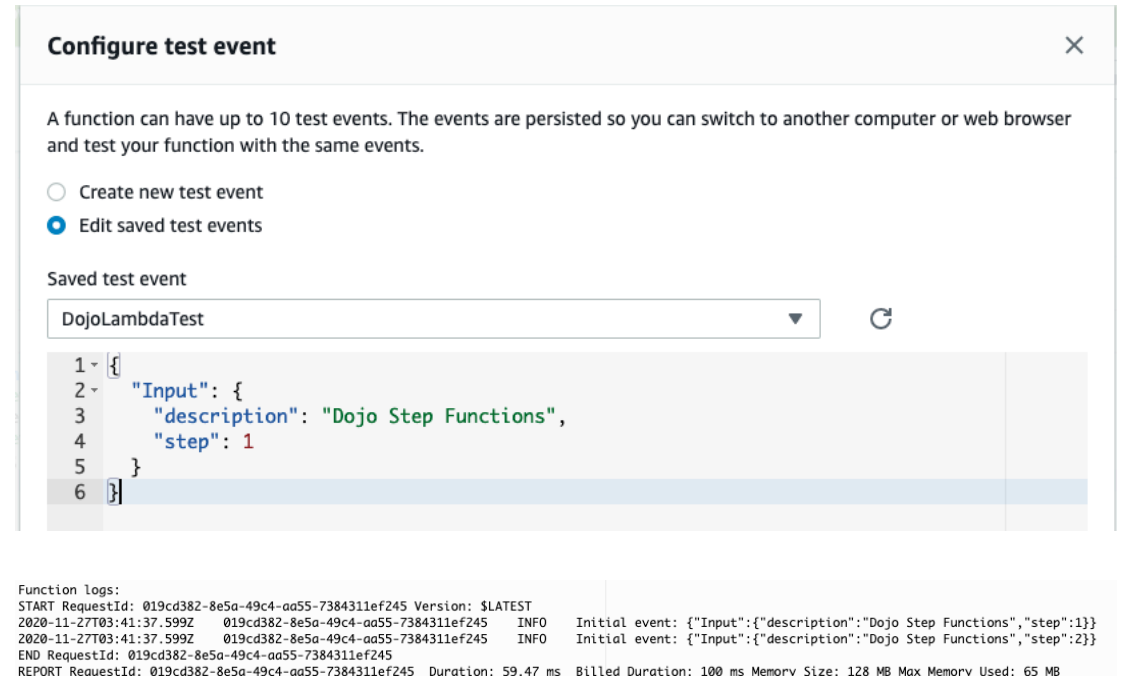


# Demo

## 4. Test Lambda Function

- Procedemos a dar clic en la opción “test event”, posteriormente, “configure test”.
- En la propiedad “EventName”, escribimos el valor “DojoLambdaTest”.
- Ingresamos el texto:

```
{
  "Input": {
    "description": "Dojo Step Functions",
    "step": 1
  }
}
```
- Para finalizar, procedemos a dar clic en la opción “Test”.



**Configure test event**

A function can have up to 10 test events. The events are persisted so you can switch to another computer or web browser and test your function with the same events.

☐ Create new test event

☒ Edit saved test events

Saved test event

DojoLambdaTest

```
1 {
2   "Input": {
3     "description": "Dojo Step Functions",
4     "step": 1
5   }
6 }
```

Function logs:

```
START RequestId: 019cd382-8e5a-49c4-aa55-7384311ef245 Version: $LATEST
2020-11-27T03:41:37.599Z    019cd382-8e5a-49c4-aa55-7384311ef245    INFO    Initial event: {"Input":{"description":"Dojo Step Functions","step":1}}
2020-11-27T03:41:37.599Z    019cd382-8e5a-49c4-aa55-7384311ef245    INFO    Initial event: {"Input":{"description":"Dojo Step Functions","step":2}}
END RequestId: 019cd382-8e5a-49c4-aa55-7384311ef245
REPORT RequestId: 019cd382-8e5a-49c4-aa55-7384311ef245  Duration: 59.47 ms  Billed Duration: 100 ms Memory Size: 128 MB Max Memory Used: 65 MB
```

# Demo

## 5. Crear Step Functions

- Abrir el servicio Step Functions en la consola AWS.
- Registramos el nombre “StepFunctionsDojo”.
- En la opción “Permissions”, seleccionamos “Create new role”.
- En la opción “Logging”, activamos la opción ALL en el atributo “Log Level”

### Specify details

#### Name

State machine name

StepFunctionsDojo

Must be 1-80 characters. Can use alphanumeric characters, dashes, or underscores.

#### Permissions

Execution role

The IAM role that defines which resources your state machine has permission to access during execution. To create a custom role, go to the [IAM console](#)

☒ Create new role

Let Step Functions create a new role for you based on your state machine's definition and configuration details.

☐ Choose an existing role

☐ Enter a role ARN

#### Logging

You can log your state machine's execution history to CloudWatch Logs. For Express state machines, you must enable logging to inspect and debug

Log level

Indicates which execution history events to log

ALL

☒ Include execution data

Log execution input, data passed between states, and execution output

CloudWatch log group

Create new log group

/aws/vendedlogs/states/StepFunctionsDojo-Logs

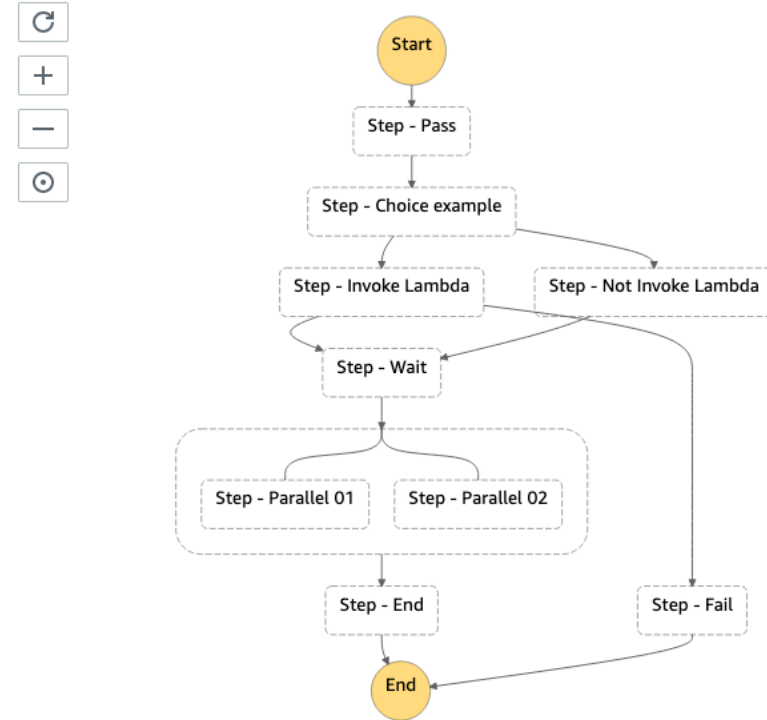
Maximum 512 alphanumeric characters. Can include hyphens, underscores, periods, and forward slashes



# Demo

## 5. Crear Step Functions

- En la vista recién desplegada, buscamos la opción “Edit state machine” y procedemos a pegar el código adjunto disponible en el repositorio: <https://github.com/Tutorial-Labs/step-functions-demo-01.git>, archivo state-machine.json en la ventana desplegada por AWS .
- Una vez el código se guarda, se deberá desplegar una imagen similar a la que se muestra a continuación:



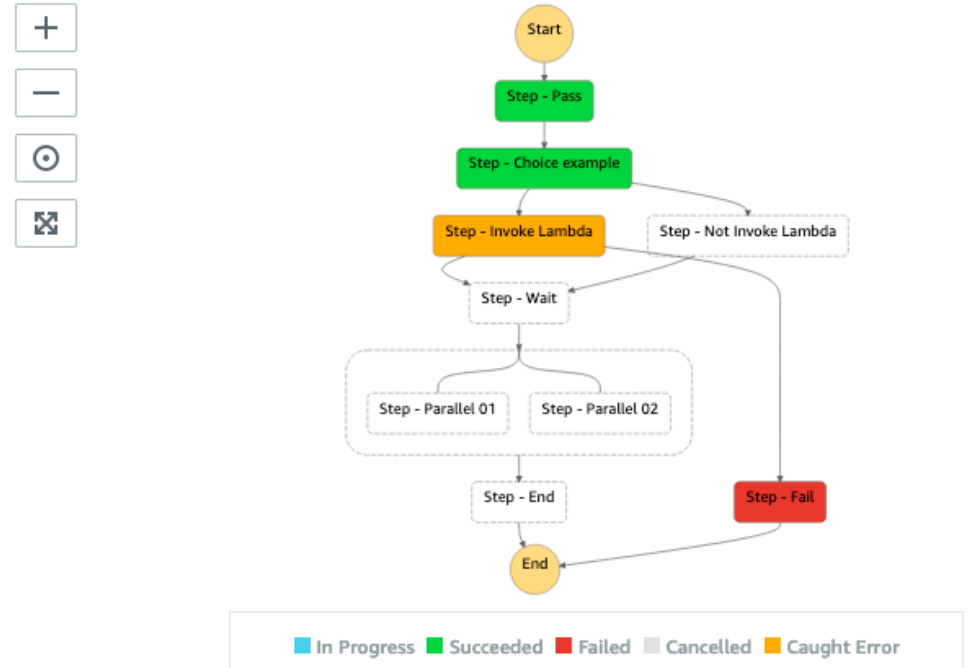
# Demo

## 6. Test Step Functions

- Ubicamos en la vista desplegada por pantalla, la opción “New execution” y en la ventana desplegada, pegamos el fragmento de código:

```
{  
  "description": "Dojo Step Functions",  
  "step":1,  
  "IsInvokeLambda": "true"  
}
```

- Al finalizar la ejecución, la step functions deberá finalizar en el step – fail, tal y como se muestra en la imagen a continuación:



# Demo

## 7. Ajustar Role Step Functions

- Abrir el servicio IAM en la consola AWS.
- Buscamos el role creado y asociado previamente a la step functions “StepFunctionsDojo”.
- Damos clic en la opción “Add inline policy”, y en la ventana desplegada, en la opción JSON, agregamos el siguiente fragmento de código:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "",
      "Effect": "Allow",
      "Action": "lambda:InvokeFunction",
      "Resource": "{arn aws lambda function}"
    }
  ]
}
```

- Posteriormente, le asignamos el nombre: InvokeStepFunctionsDojoLambda-Policy.

### Create policy

1 2

A policy defines the AWS permissions that you can assign to a user, group, or role. You can create and edit a policy in the visual editor and using JSON. [Learn more](#)

Visual editor

JSON

[Import managed policy](#)

```
1 {
2   "Version": "2012-10-17",
3   "Statement": [
4     {
5       "Sid": "",
6       "Effect": "Allow",
7       "Action": "lambda:InvokeFunction",
8       "Resource": "arn:aws:lambda:us-east-1:393478962179:function:StepFunctionsDojoLambda"
9     }
10  ]
11 }
```

### Create policy

1 2

#### Review policy

Before you create this policy, provide the required information and review this policy.

Name\* InvokeStepFunctionsDojoLambda-Policy

Maximum 128 characters. Use alphanumeric and '+', '=', '@', '-' characters.

#### Summary

Filter			
Service	Access level	Resource	Request condition
Allow (1 of 247 services) <a href="#">Show remaining 246</a>			
Lambda	Limited: Write	FunctionName   string like   StepFunctionsDojoLambda:\$LATEST	None

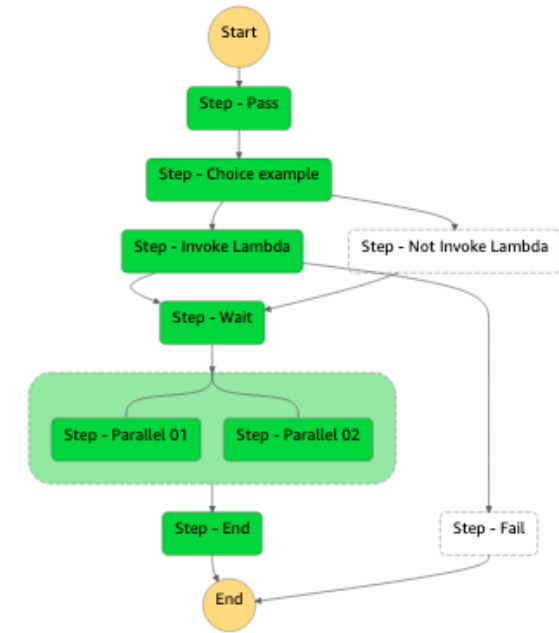
# Demo

## 8. Test Step Functions

- Ubicamos en la vista desplegada por pantalla, la opción “New execution” y en la ventana desplegada, pegamos el fragmento de código:

```
{  
  "description": "Dojo Step Functions",  
  "step":1,  
  "IsInvokeLambda": "true"  
}
```

- Al finalizar la ejecución, la step functions deberá finalizar en el step – End, tal y como se muestra en la imagen a continuación:



■ In Progress ■ Succeeded ■ Failed ■ Cancelled ■ Caught Error



## Referencias

- [https://docs.aws.amazon.com/es\\_es/step-functions/latest/dg/step-functions-dg.pdf](https://docs.aws.amazon.com/es_es/step-functions/latest/dg/step-functions-dg.pdf)
  - [https://docs.aws.amazon.com/es\\_es/step-functions/latest/dg/concepts-amazon-states-language.html](https://docs.aws.amazon.com/es_es/step-functions/latest/dg/concepts-amazon-states-language.html)
  - [https://docs.aws.amazon.com/es\\_es/step-functions/latest/dg/concepts-service-integrations.html](https://docs.aws.amazon.com/es_es/step-functions/latest/dg/concepts-service-integrations.html)
  - [https://docs.aws.amazon.com/es\\_es/step-functions/latest/dg/tutorial-creating-lambda-state-machine.html](https://docs.aws.amazon.com/es_es/step-functions/latest/dg/tutorial-creating-lambda-state-machine.html)
  - <https://solace.com/blog/microservices-choreography-vs-orchestration/>
  - [https://docs.aws.amazon.com/es\\_es/step-functions/latest/dg/amazon-states-language-task-state.html](https://docs.aws.amazon.com/es_es/step-functions/latest/dg/amazon-states-language-task-state.html)
-