Arquitecturas de nube con AWS

Ing. Fernando Lichtschein
Ing. Mora Villa Abrille

14. Arquitecturas serverless y microservicios

UNIVERSIDAD AUSTRAL

Objetivos

Definir arquitecturas *serverless*.

Identificar las características de los microservicios.

Diseñar una solución *serverless* con AWS Lambda.

Definir cómo se usan los contenedores en AWS.

Describir los tipos de flujo que soportan las AWS Step Functions.

Describir una arquitectura común para el Amazon API Gateway.

Aplicar los principios del AWS Well-Architected Framework para construir arquitecturas *serverless*.

Objetivos

De un arquitecto de nube

Reconocer cuándo usar arquitecturas *serverless* en AWS y qué servicios elegir, de acuerdo con el caso de uso.

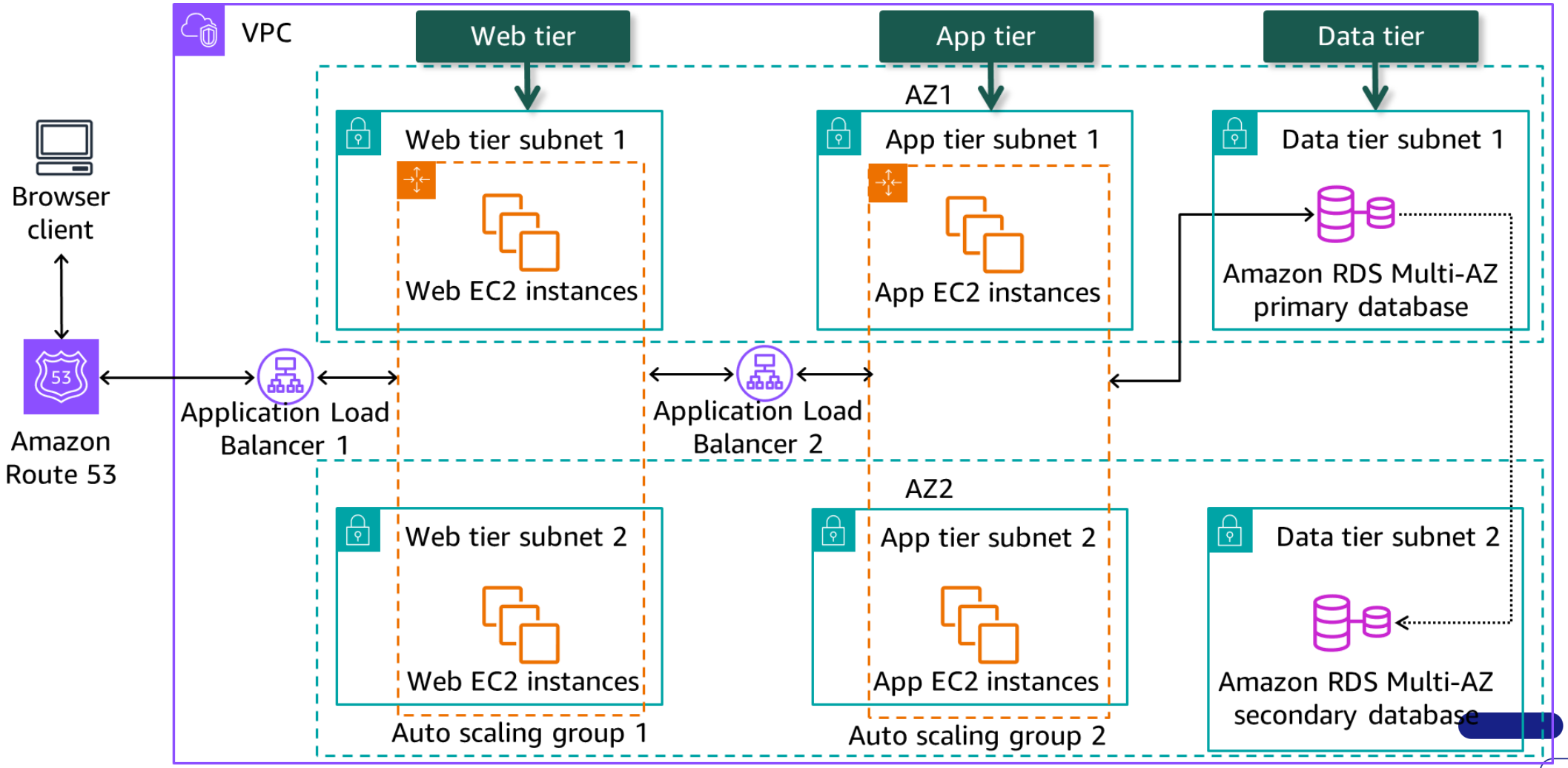
Implementar arquitecturas guiadas por eventos con microservicios para construir arquitecturas de microservicios escalables y resilientes.

Conocer cuándo usar herramientas de orquestación para que la arquitectura de microservicios trate las fallas con una intervención manual mínima.

Soluciones *serverless*

UNIVERSIDAD AUSTRAL

Diseño de una aplicación web en una VPC



Beneficios de AWS *serverless*

No requiere administrar servidores

Escalamiento continuo

Se paga por el valor del servicio

Alta disponibilidad desde el diseño

Adecuado para arquitecturas orientadas a eventos y microservicios

Servicios *serverless* de AWS

Compute

AWS Lambda and Lambda@Edge

AWS Fargate

Authentication

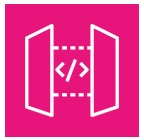
Amazon Cognito

Web hosting

AWS Amplify

Content delivery

Application integration



Amazon API Gateway

AWS AppSync

Amazon Simple Notification Service (Amazon SNS)

Amazon Simple Queue Service (Amazon SQS)

Amazon EventBridge

AWS Step Functions

Data stores

Amazon S3

Amazon EFS

Amazon DynamoDB

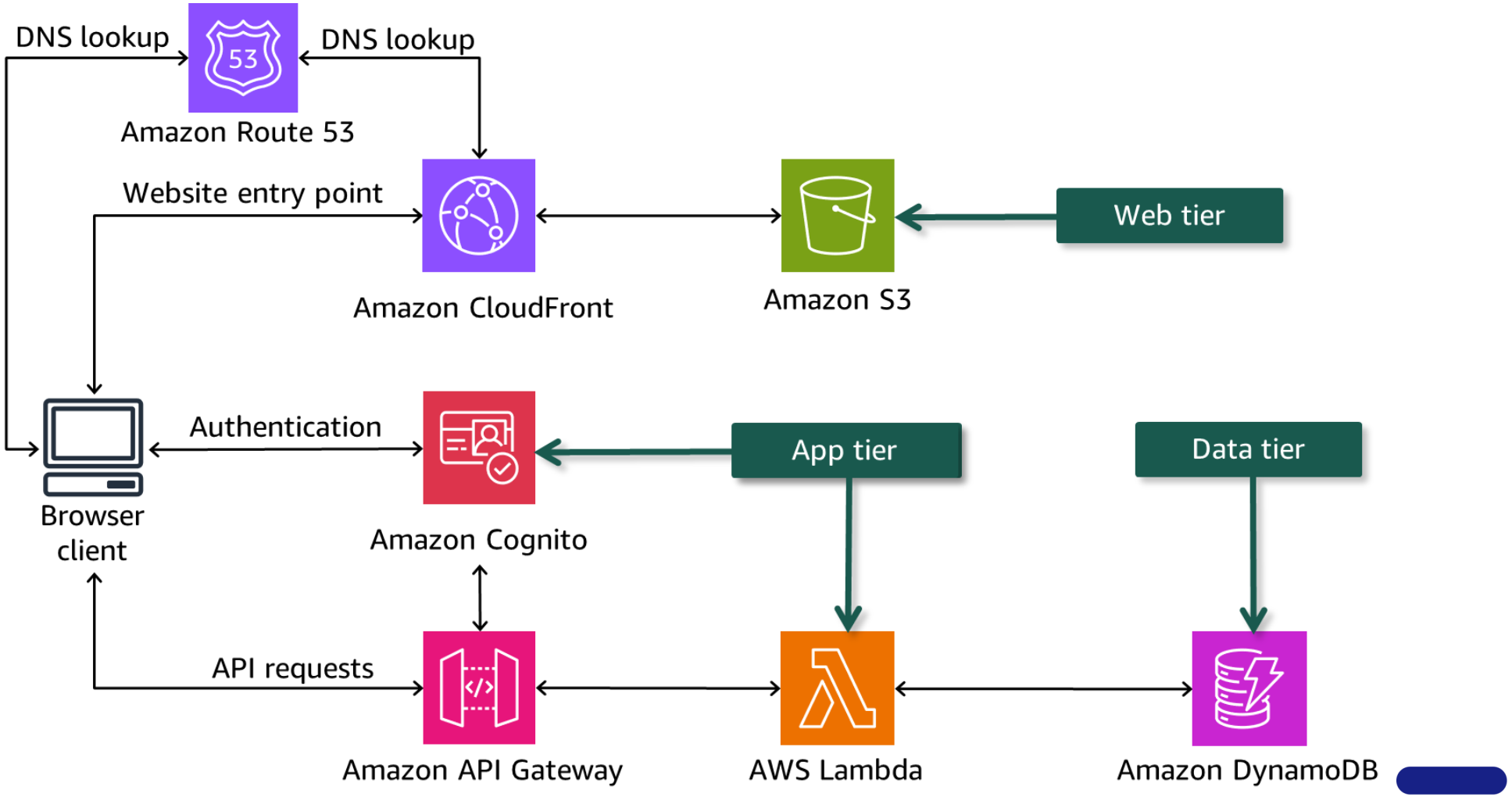
Amazon Neptune Serverless

Amazon Aurora Serverless

Amazon Redshift Serverless

Amazon OpenSearch Serverless

Aplicación web usando servicios *serverless*



Resumen

Los servicios *serverless* AWS tienen los siguientes beneficios:

- No es necesario administrar servidores
- Se paga por el valor del servicio
- Tienen escalamiento continuo
- Son tolerantes a fallas por diseño

Estos servicios son adecuados para arquitecturas orientadas a eventos y microservicios.

Arquitectura de microservicios *serverless*

UNIVERSIDAD AUSTRAL

Microservicios

Características

Autónomo

Se puede desarrollar e implementar sin afectar a otros microservicios

Escala de manera independiente

No comparte código con otros microservicios

Se comunica mediante APIs bien definidas

Especializado

Realiza una única función de negocio, resolviendo un problema específico.

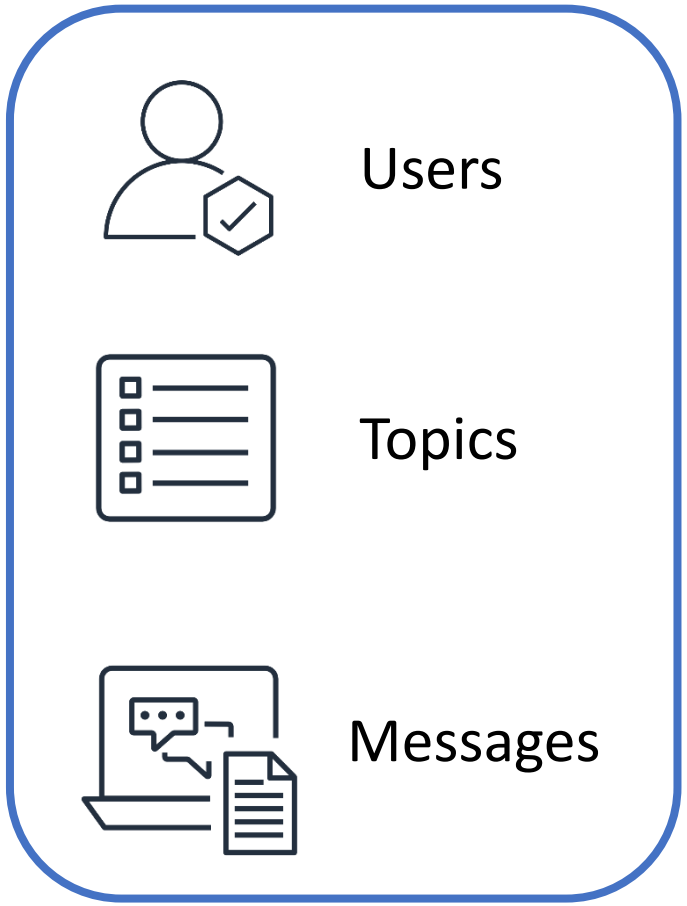
Pertenece a un equipo de desarrollo pequeño que selecciona herramientas de desarrollo

Es *stateless*

Tiene su propio almacenamiento de datos

Comparación

Monolithic application



Microservice application

Microservicios

Beneficios

Agilidad

Código reutilizable

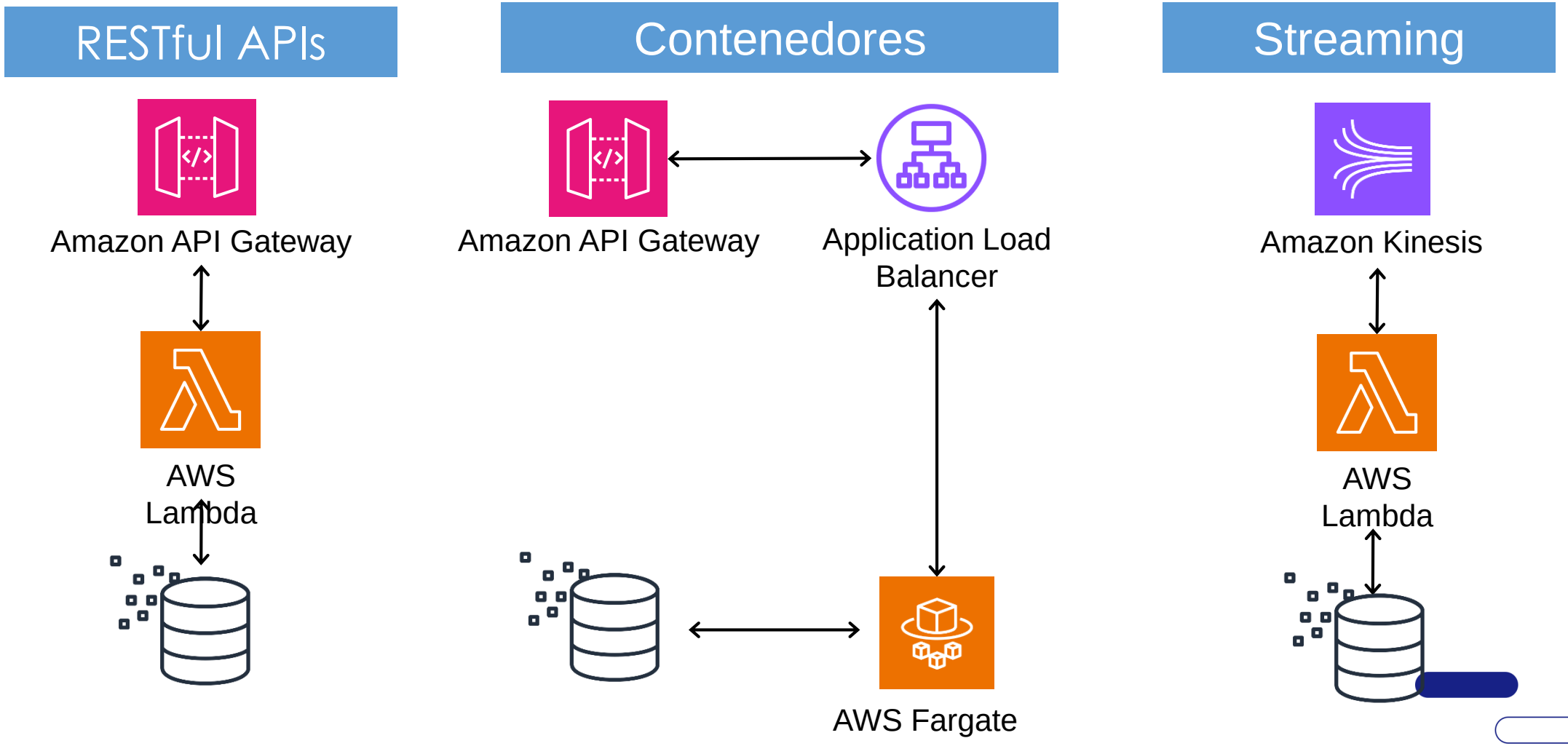
Escalamiento flexible

Libertad tecnológica

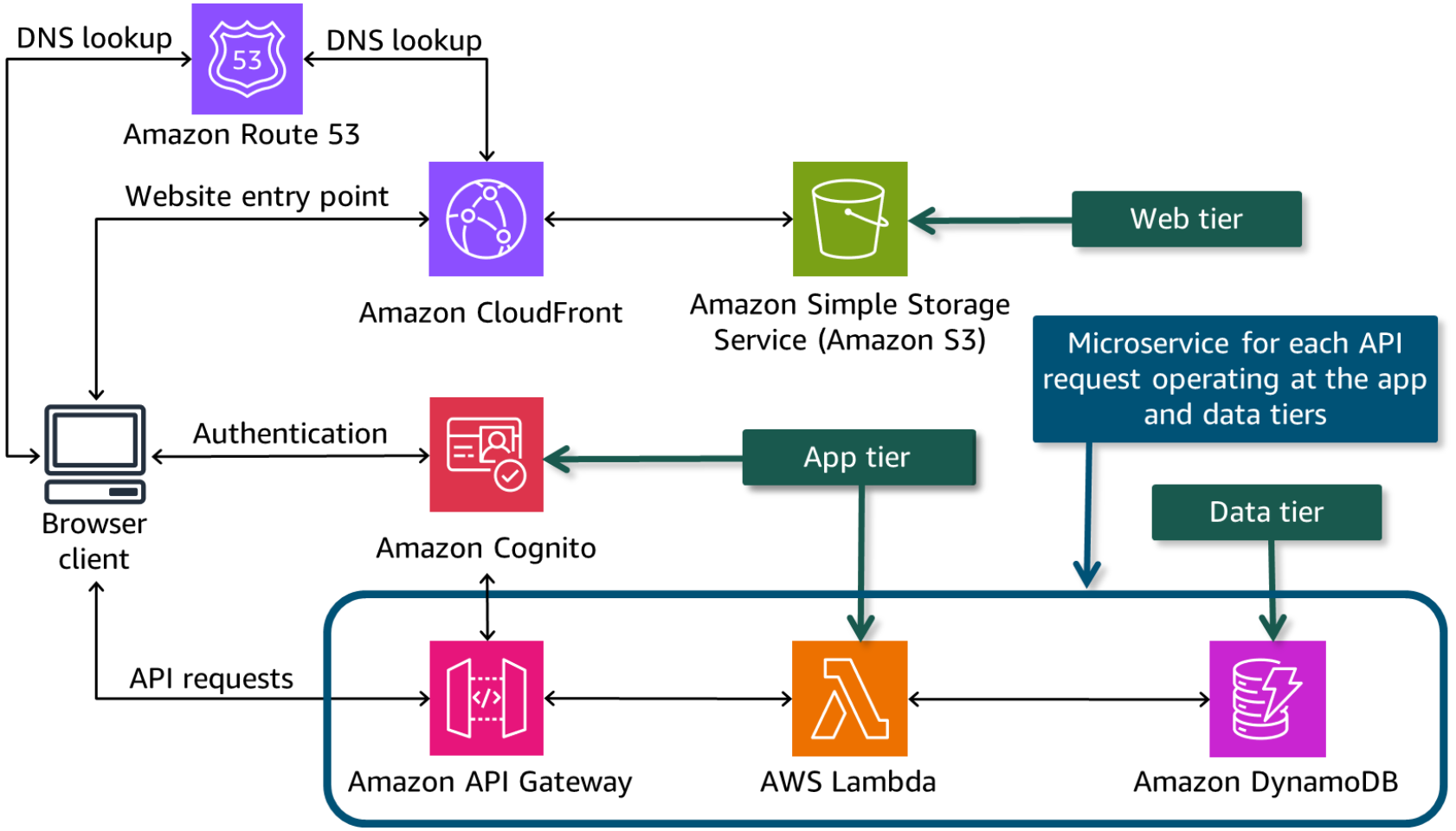
Resiliencia

Implementación simplificada

Patrones de microservicios serverless en AWS



Microservicios en una arquitectura web serverless



Resumen

Las aplicaciones basadas en microservicios están compuestas por servicios autónomos y especializados.

Los microservicios tienen ventajas como: agilidad, reutilización de código, escalamiento flexible, resiliencia, etc.

Los microservicios pueden formar parte de una arquitectura de tres capas, operando en las capas de datos y aplicación.

Arquitecturas *serverless* con AWS Lambda

UNIVERSIDAD AUSTRAL

Comparación de tareas operacionales

Tareas	Servidor en una VPC	Serverless
Configurar una instancia	Yes	No
Actualizar el sistema operativo	Yes	No
Instalar la plataforma de aplicación	Yes	No
Construir e implementar aplicaciones	Yes	Yes
Configurar el escalamiento y el balanceo de carga	Yes	No
Proteger y monitorear instancias continuamente	Yes	No
Monitorear y mantener las aplicaciones	Yes	Yes

AWS Lambda

AWS Lambda

AWS Lambda permite ejecutar funciones de código sin crear ni gestionar servidores.

Las funciones Lambda son configurables respecto de: lenguaje de ejecución, cantidad de memoria y duración.

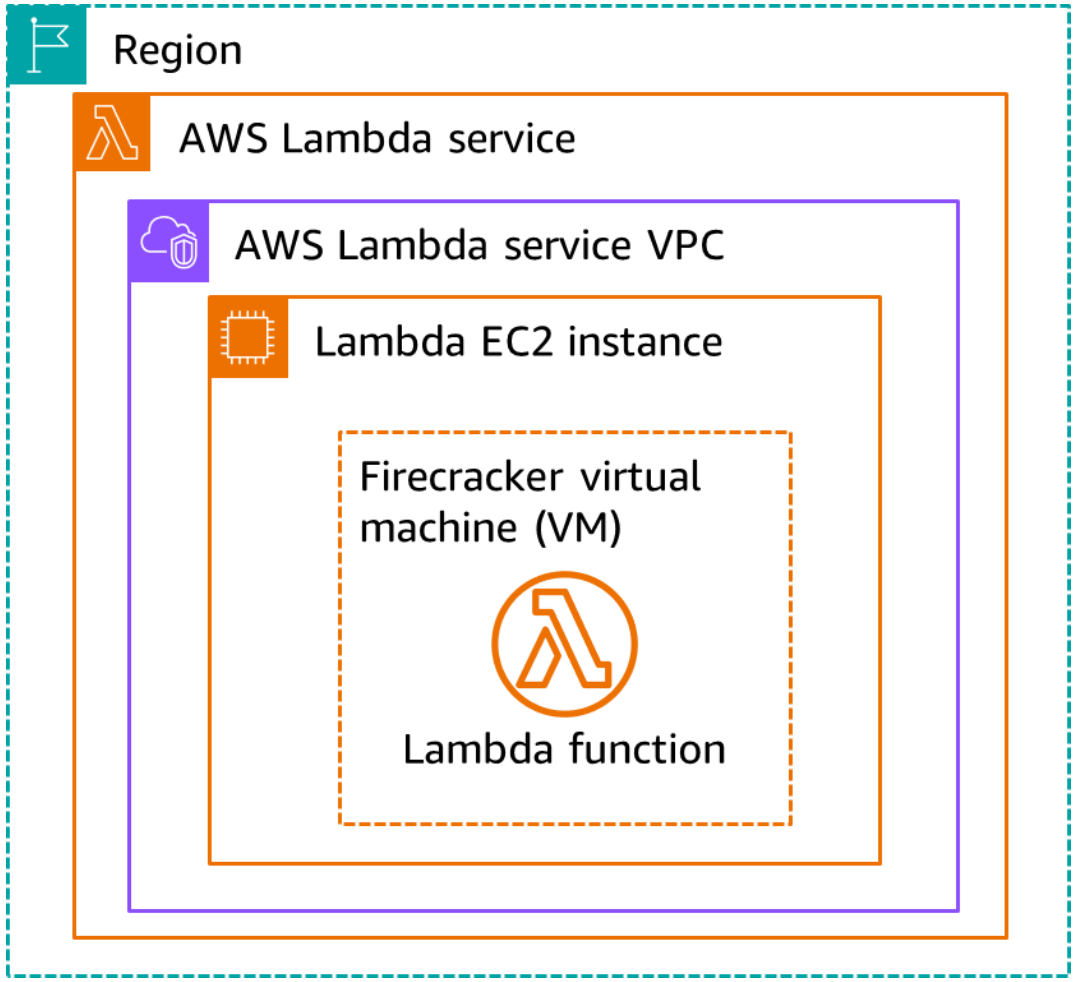
Una función puede durar15 minutos como máximo.

Las funciones se implementan como archivos .zip o imágenes de contenedores.

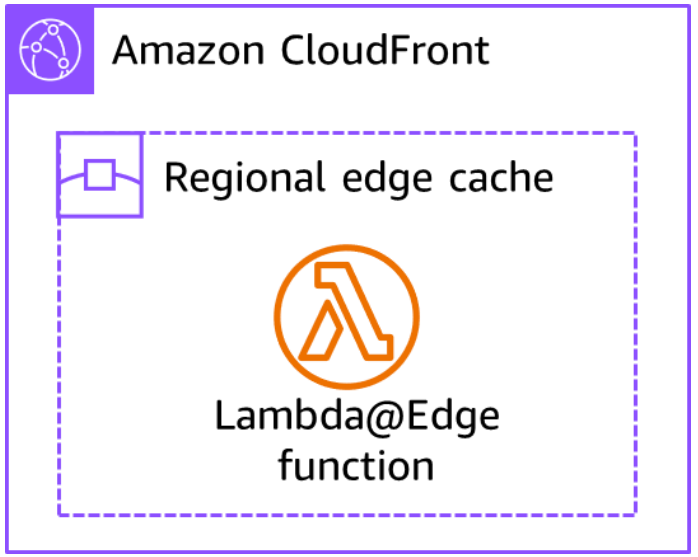
Funciones Lambda

Ubicaciones

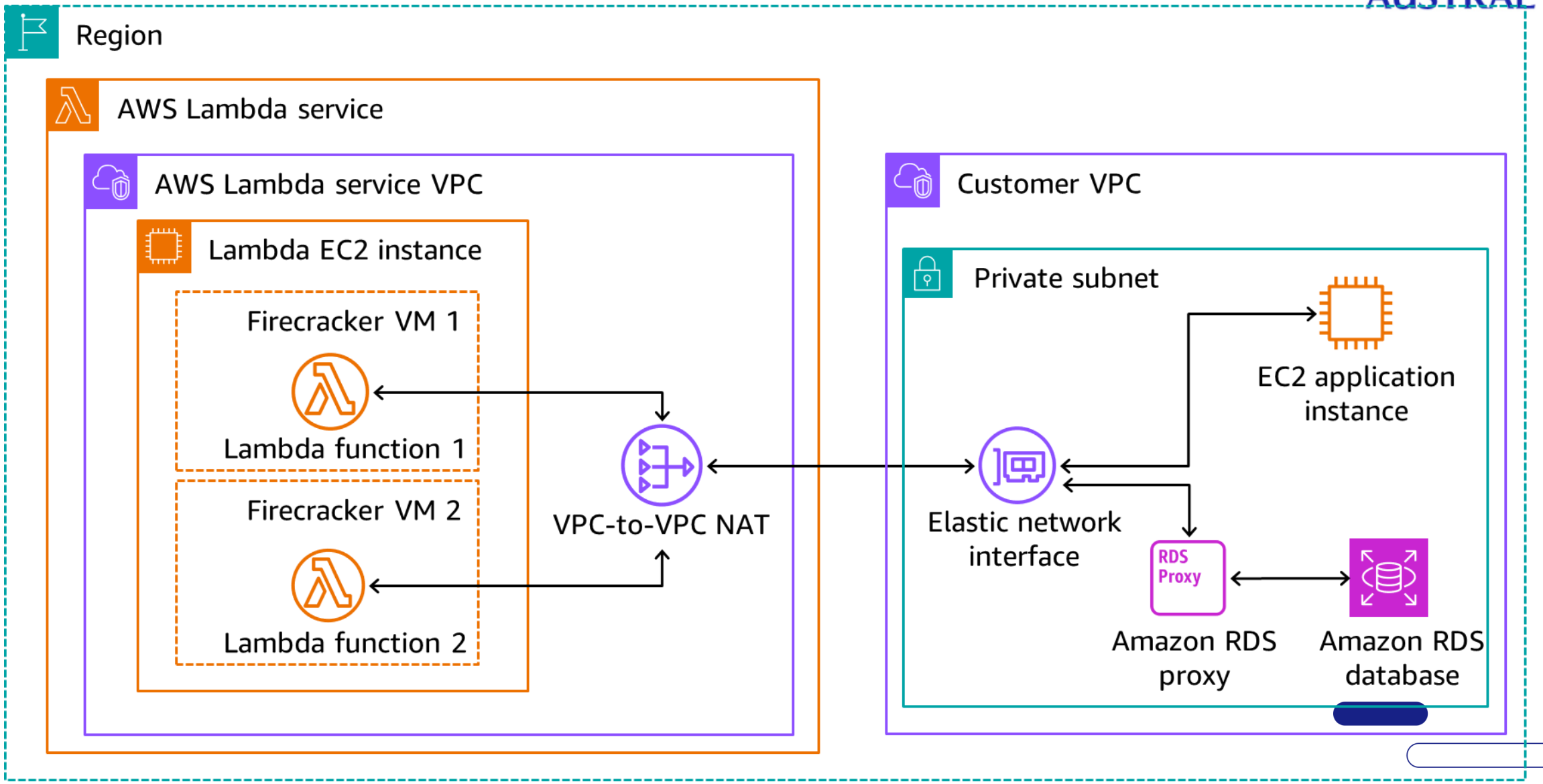
En el servicio AWS Lambda



En un cache regional de Amazon CloudFront



Conectar una función Lambda a una VPC



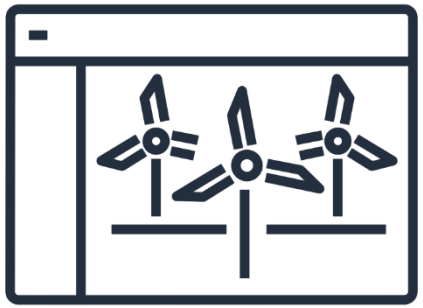
Escenarios para usar funciones Lambda

Procesamiento sincrónico

- Aplicaciones web
- Servicios web
- Microservicios
- Inferencias de machine learning

Procesamiento asincrónico

- Eventos programados
- Mensajes en colas
- Transformación de imagen o video
- Triggers de servicios de AWS



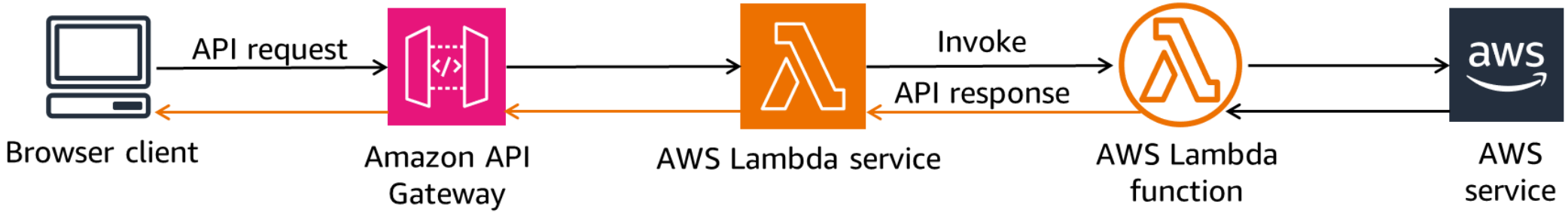
Procesamiento de streaming

Aplicaciones de streaming

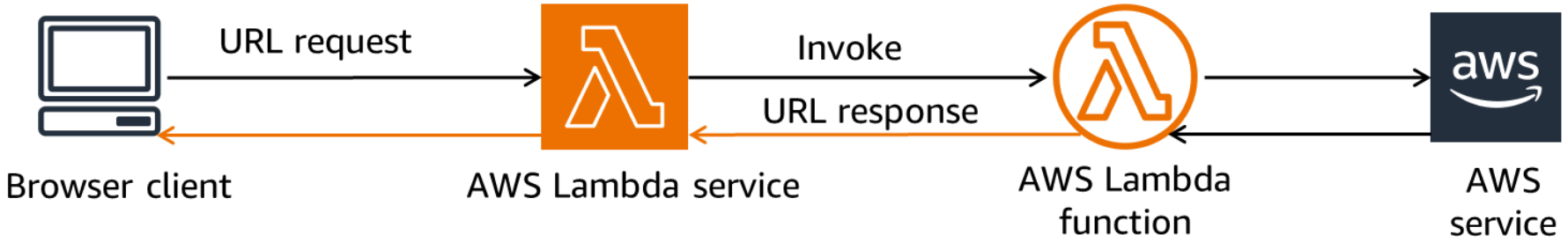
Funciones Lambda sincrónicas

Llamadas

Usando un API Gateway



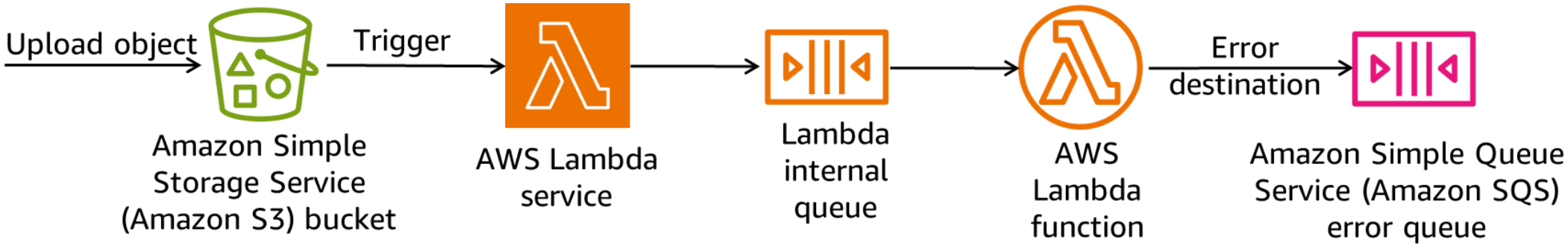
Usando la URL de la función Lambda desde una API externa



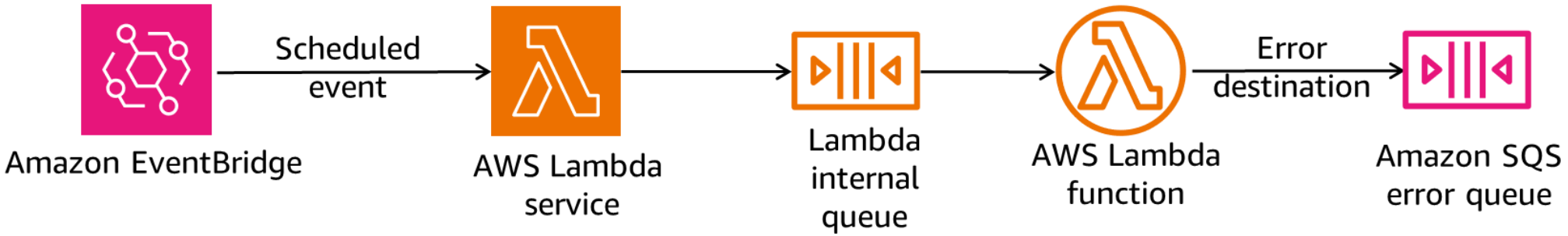
Funciones Lambda asincrónicas

Llamadas

Trigger de un servicio de AWS

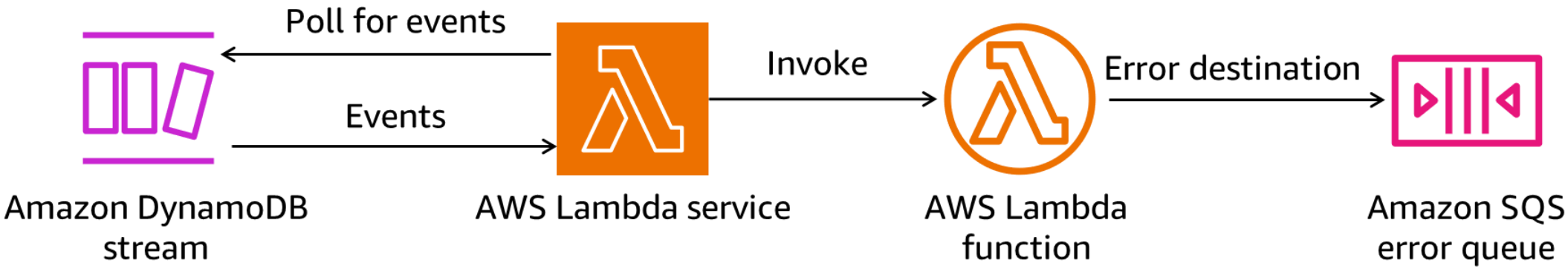


Evento programado



Streaming y colas

Source mapping



Handler de una función Lambda en Python

UNIVERSIDAD AUSTRAL

```
1    import json

2    def lambda_handler(event, context):

3     length=event['length']
4     width=event['width']

5     area = calculate_area(length, width)

6     data = {"area": area}
7     return json.dumps(data)


8    def calculate_area(length, width):
9     return length*width
```

Line 2: Event object is a JSON document containing input data and invoking service data

Line 2: Context object provides methods and properties about function runtime and invocation

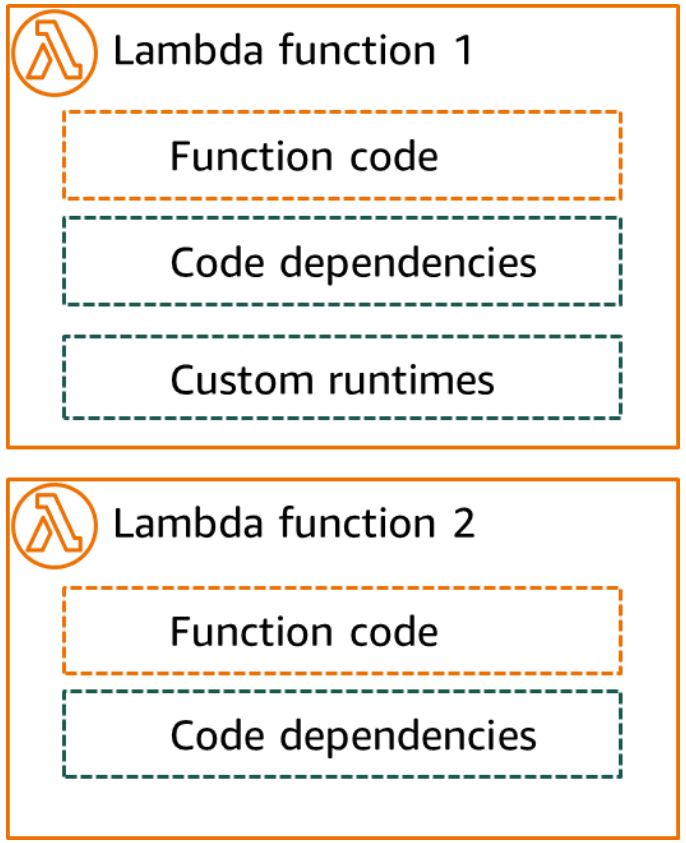
Line 7: json.dumps(data) returns result as a JSON document

Lines 8 and 9: Business logic method

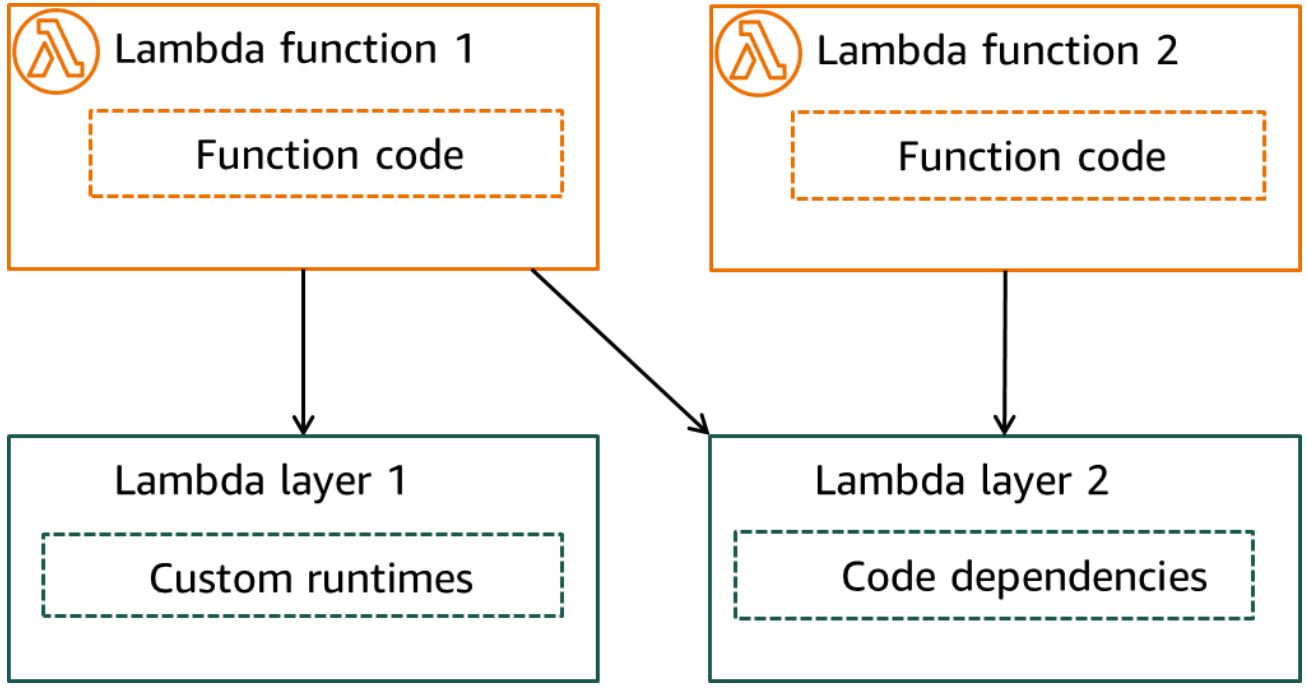
Capas de funciones lambda

UNIVERSIDAD AUSTRAL

Sin capas



Con capas



Resumen

AWS Lambda es un servicio que permite ejecutar funciones de código sin crear ni administrar servidores.

Una función Lambda puede ejecutarse dentro de una VPC que pertenece al servicio de AWS Lambda o como una Lambda@Edge en una cache regional de Amazon CloudFront.

Una función Lambda se puede configurar para que se conecte a una VPC y acceda a los servicios que se ejecutan en ella.

Se puede llamar a una función Lambda de manera sincrónica, asincrónica, y mapeando fuentes de eventos (para colas y *streams*).

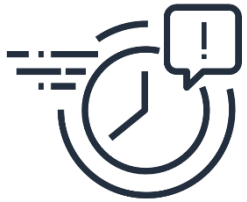
Las capas de Lambda permiten armar paquetes de dependencias o rutinas que pueden ser reutilizadas por todas las funciones Lambda de la región.

Contenedores en AWS

UNIVERSIDAD AUSTRAL

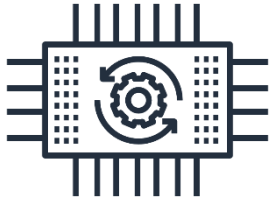
Contenedores

Casos de uso vs funciones lambda



Más de 15 minutos

AWS Lambda tiene un límite de 15 minutos de duración para la ejecución de funciones.



Aplicaciones que usan mucha memoria

Las cargas de trabajo que exceden los 10 GB de memoria no son apropiadas para funciones lambda.

Costo

Los contenedores pueden ejecutarse de manera continua con costo fijo.

El precio de una función lambda aumenta con la cantidad de ejecuciones, la duración y la memoria utilizada.

Migración de contenedores legacy

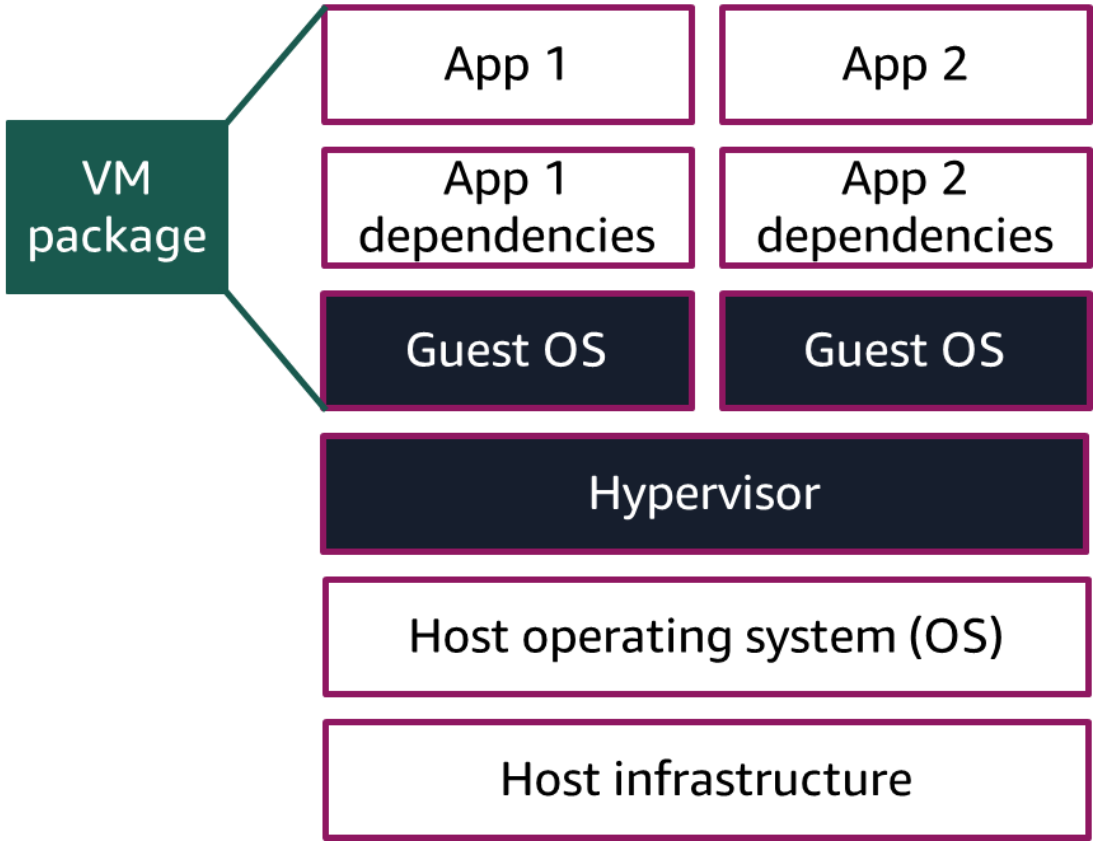
Los contenedores pueden ayudar a migrar aplicaciones legacy que se ejecutan on-premises o en instancias de EC2.

Contenedores

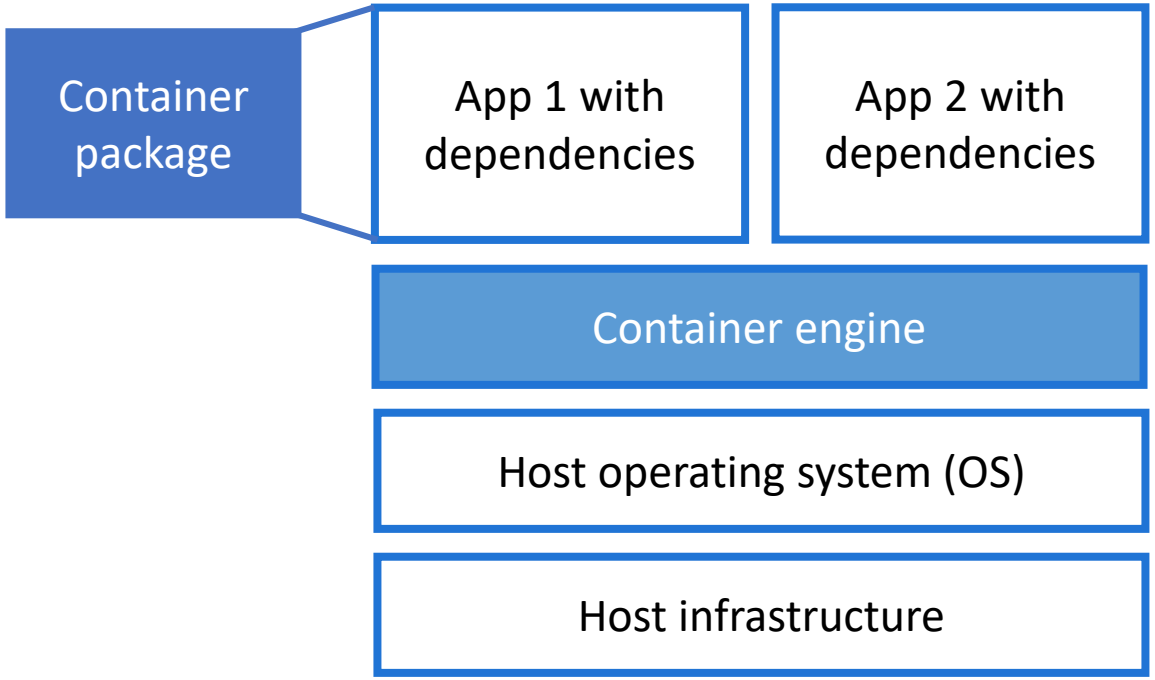
Beneficios

UNIVERSIDAD AUSTRAL

Máquinas virtuales



Contenedores



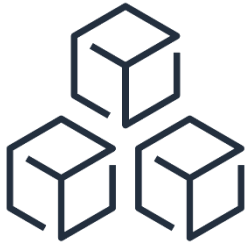
Contenedores

Casos de uso

Aplicaciones con microservicios

Procesamiento batch

Escalar modelos de machine learning (ML)

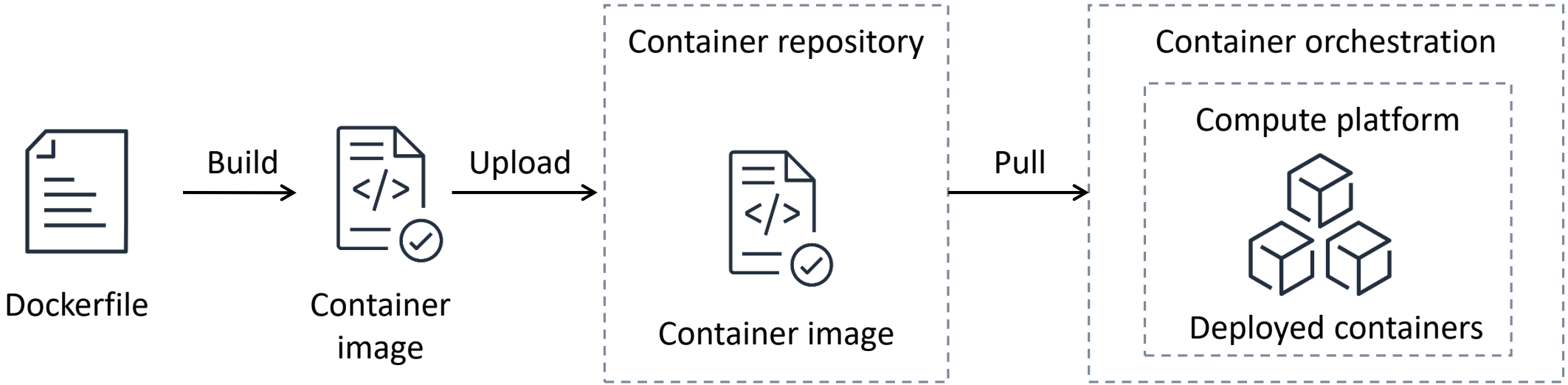


Estandarizar aplicaciones de arquitectura híbrida

Migrar aplicaciones a la nube

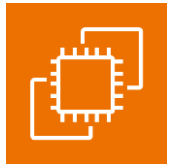
Contenedores

Docker containers

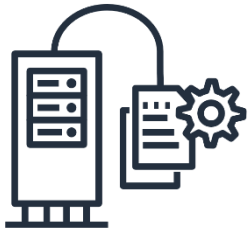


Servicios de contenedores en AWS

UNIVERSIDAD AUSTRAL

Registro	Orquestación	Cómputo
		
Amazon Elastic Container Registry (Amazon ECR)	Amazon Elastic Container Service (Amazon ECS)	Amazon Elastic Compute Cloud (Amazon EC2)
		
	Amazon Elastic Kubernetes Service (Amazon EKS)	AWS Fargate
		
		AWS Lambda

Beneficios de AWS Fargate



Sin gestión de clusters o servidores

No hay que crear ni mantener servidores.

No necesitamos optimizar clusters.

Facturación por segundo

Pago por uso

Escalamiento automático

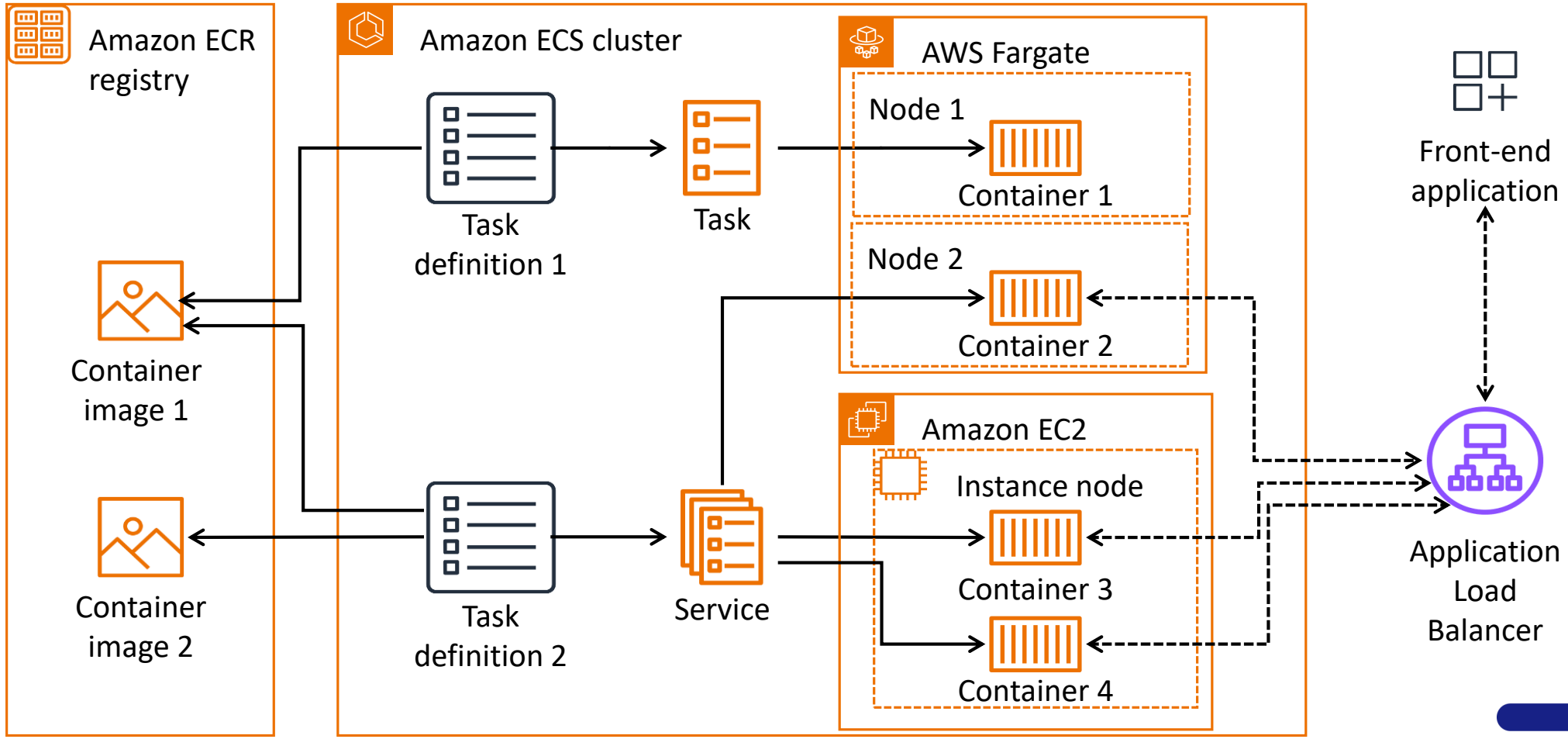
Escalamiento automático de tareas según el uso de CPU, memoria u otras métricas.

Adecuado para equipos nuevos

No requiere conocimiento profundo de la tecnología de contenedores.

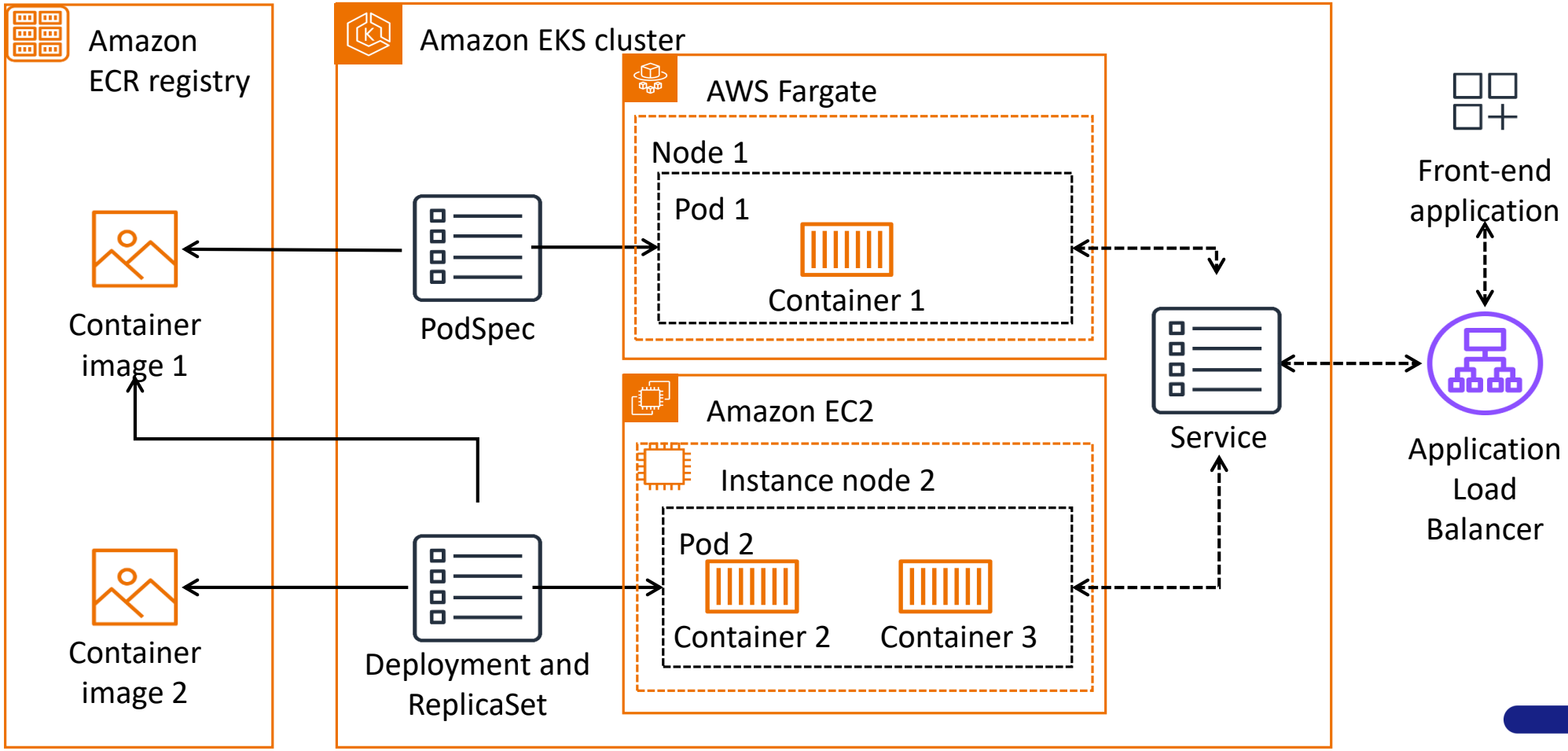
Amazon ECS

Implementación y ejecución de contenedores



Amazon EKS

Implementación y ejecución de contenedores



Amazon EKS y Amazon ECS

Diferencias

Tema	Amazon ECS	Amazon EKS
Complejidad	Simplifica la creación y el mantenimiento de clusters	Provee mayor control sobre los clusters, pero la interfaz es compleja
Escalamiento	Escalamiento automático a demanda	Configuración manual de los grupos de autoescalamiento
Herramientas	Amazon ECS	Kubernetes
Experiencia del equipo	Nuevo en la arquitectura de contenedores	Familiarizado con la arquitectura y los procesos de control de Kubernetes

Contenedores

Resumen

Los contenedores son una mejor solución que las funciones Lambda cuando la aplicación requiere recursos que exceden los límites de servicio de AWS Lambda

Amazon ECR proporciona un servicio de repositorio de imágenes de contenedores.

Amazon ECS es un servicio de orquestación de contenedores con herramientas propias de AWS.

Amazon EKS es un servicio de orquestación de contenedores con herramientas de Kubernetes.

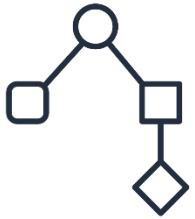
AWS Fargate administra nodos serverless y se puede implementar en clusters de Amazon ECS o Amazon EKS.

AWS Step Functions

UNIVERSIDAD AUSTRAL

Microservicios

Desafíos



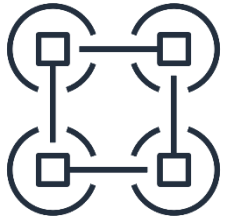
Dependencias

Administrar las dependencias entre microservicios

Encadenar los microservicios en secuencia o en paralelo

Escenarios de error

Ejecutar reintentos luego de errores o *timeouts*



Coordinación

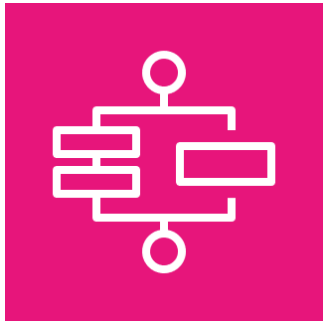
Escalar la aplicación

Mantener el estado de microservicios *stateless*

Pasar datos entre microservicios

Monitoreo y resolución de problemas

AWS Step Functions



Step Functions

- Servicio de orquestación *serverless* que maneja flujos entre distintos servicios de AWS
- Tiene máquinas de estado (workflows) que contienen una serie de estados dependientes de eventos (pasos)
- Gestiona el estado, los checkpoints y los reinicios de cada workflow
- Tiene funciones para el tratamiento de errores
- Puede transferir datos entre estados
- Los estados pueden filtrar y manipular datos

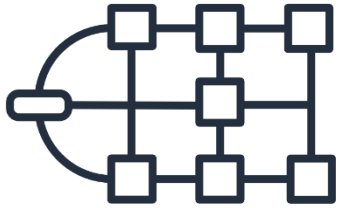
Flujos estándar o exprés

Criterio	Standard Workflows	Express Workflows
Duración	Larga	Breve, sin actividades
Métricas	Historia completa en la consola	Resultados en logs de CloudWatch
Procesamiento	Asincrónico	Sincrónico o asincrónico
Modelo de ejecución	Exactamente una vez	Sincrónico: al menos una vez Asincrónico: Como máximo, una vez
Progreso de la máquina de estado	Se persiste en cada transición de estado	Sin persistencia del estado en cada transición de estado
Precios	Por cantidad de transiciones de estado	Por cantidad y duración de llamadas por workflow

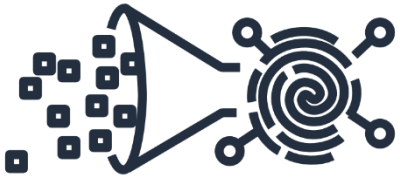
Step Functions

Casos de uso

UNIVERSIDAD AUSTRAL



Orquestar microservicios

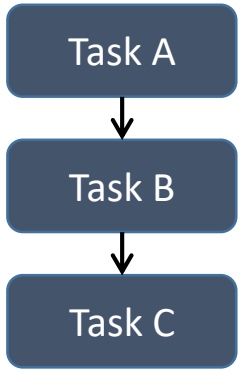


Procesamiento de datos

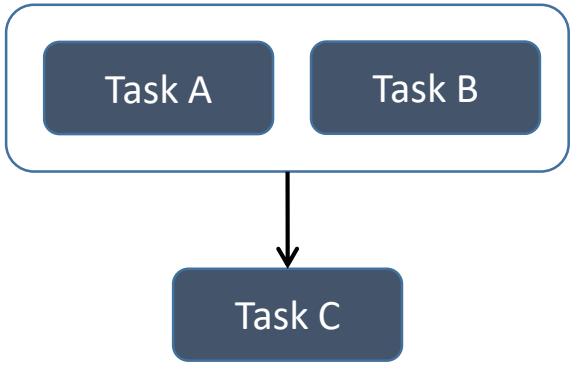
Machine learning (ML)

Automatización de seguridad

Coordinación de estados

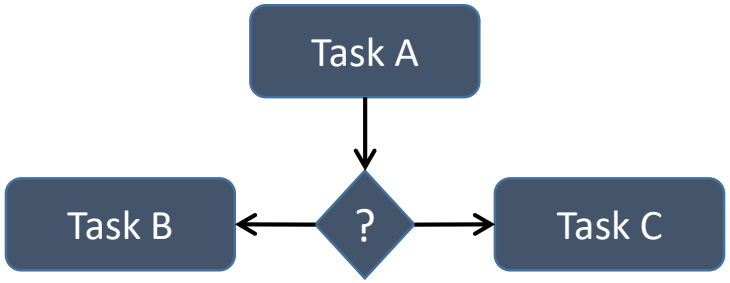


Ejecución secuencial

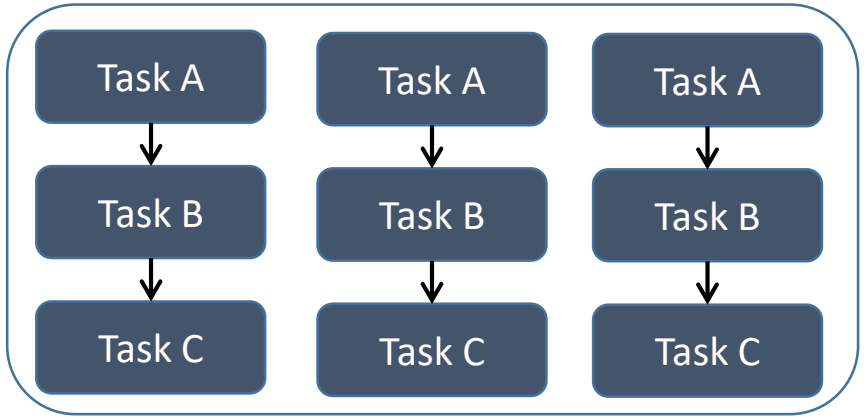


Ejecución en paralelo

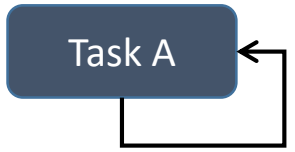
Gestión de errores (try-catch-finally)



Elegir tarea según los datos



Procesar tareas sobre registros de datos en paralelo



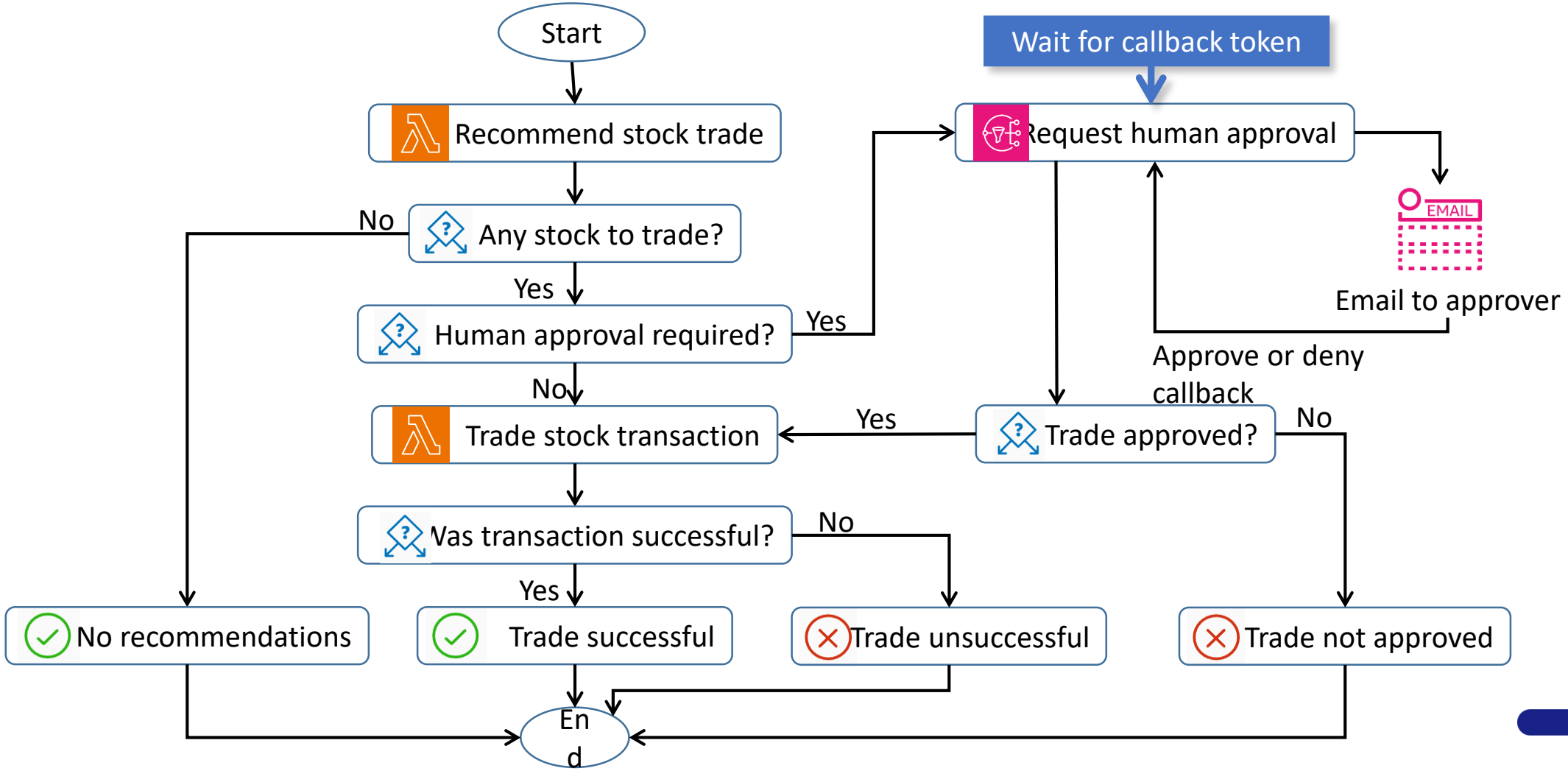
Reiteración de tareas fallidas

Tipos de estado

UNIVERSIDAD AUSTRAL

Estados de trabajo	Estados de transición	Estados de detención
Task: se integra con servicios de AWS	**Choice:** Agrega condiciones para controlar el flujo hacia el estado siguiente	**Success:** Detiene la ejecución y la marca como exitosa
Activity: Realiza una tarea en cualquier lugar	**Parallel:** Agrega ramas de máquinas de estado anidadas en una máquina de estado	**Fail:** Detiene la ejecución y la marca como fallida
Pass: Pasa o filtra datos de entrada al estado siguiente	**Map:** Separa el flujo de cada registro de datos en ejecuciones de data sets que corren en paralelo	**End parameter:** Detiene la ejecución
Wait: Demora el flujo por un período especificado		
State has wait for callback state option: Pausa el flujo y espera el retorno		

Ejemplo


Start
Recommend stock trade
No
Any stock to trade?
Yes
Human approval required?
Yes
No
Trade stock transaction
Was transaction successful?
No
Yes
No recommendations
Trade successful
Trade unsuccessful
End
Wait for callback token
Request human approval
EMAIL
Email to approver
Approve or deny callback
Yes
Trade approved?
No
Trade not approved

Resumen

AWS Step Functions es un servicio de orquestación serverless que maneja flujos entre múltiples servicios de AWS.
Una máquina de estados (workflow) es una serie de estados (pasos) manejados por eventos.
Una máquina de estado de Step Functions es una colección de estados definidos en el Amazon States Language.
Los estados se pueden agrupar en tres categorías: trabajo, transición y detención.
El estado de una tarea puede llamar a un servicio de AWS o generar una solicitud a una actividad en cualquier servicio que tenga una conexión HTTP.

Amazon API Gateway

UNIVERSIDAD AUSTRAL

Ventajas de las API

Estandarizar la comunicación entre apps

Estandarizar la conexión entre aplicaciones desarrolladas en distintos lenguajes

Ocultar la complejidad de la implementación

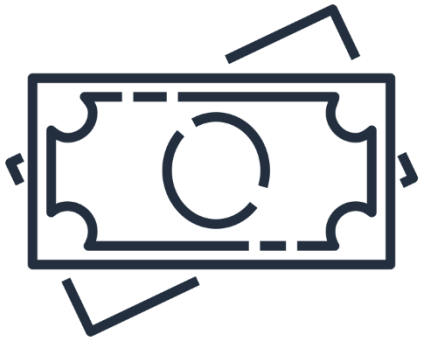
Proteger los microservicios

Decidir cuándo requerir autorización

Verificar los formatos de las solicitudes.

Manejar la cantidad de solicitudes.

Restringir el acceso a recursos

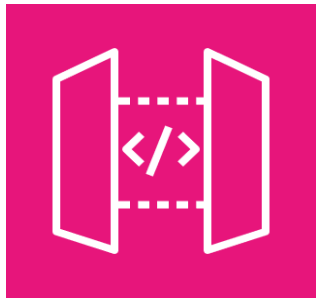


Monetizar las API y registrar estadísticas

Registrar el acceso de clientes para gestionar la facturación

Generar estadísticas de uso por cliente

API Gateway



API Gateway

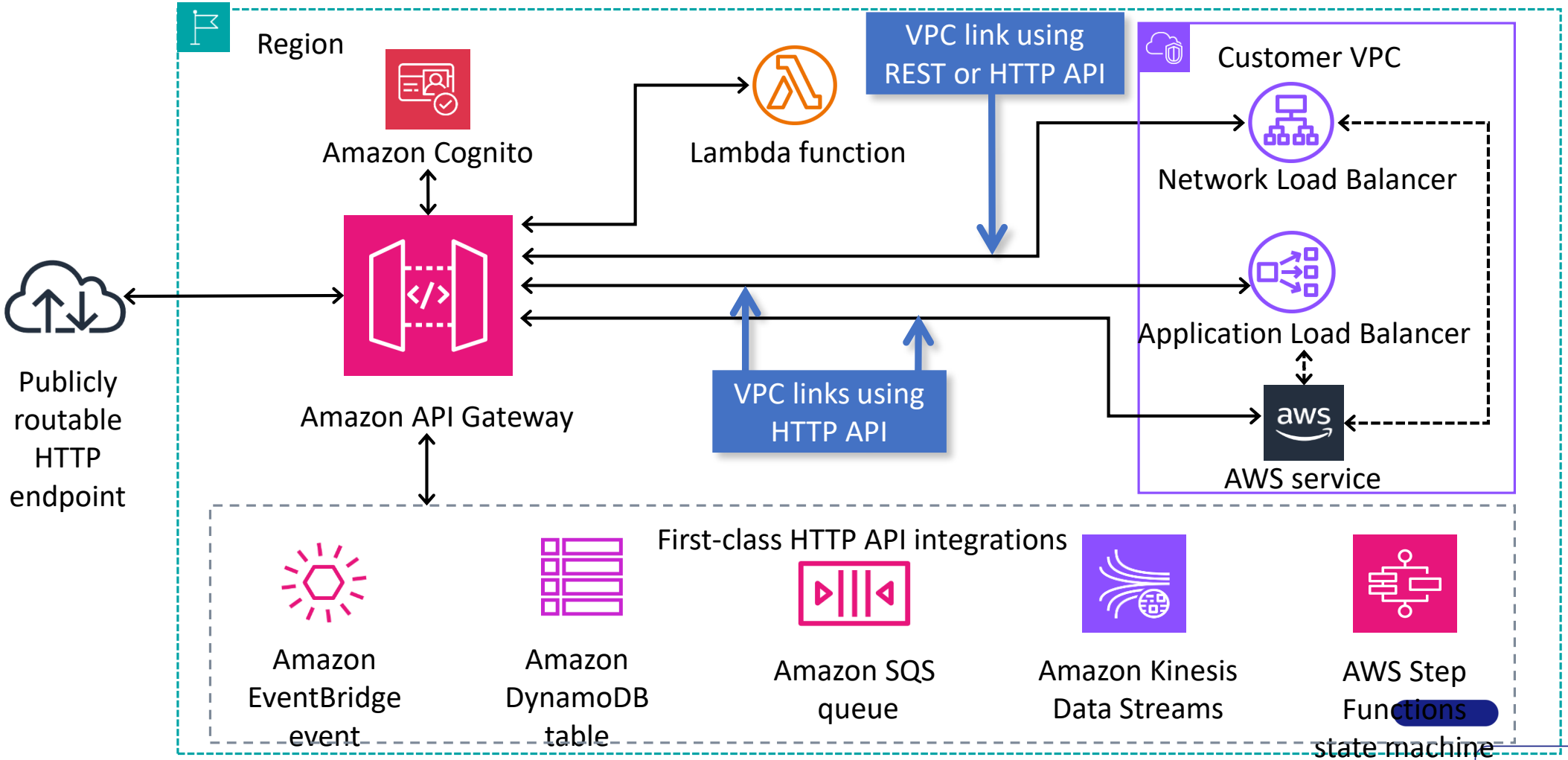
- Permite crear, publicar y mantener APIs de tipo REST, HTTP, y WebSocket
- Gestión de tráfico, autorización y control de acceso a recursos configurables
- Brinda acceso a servicios de AWS y a endpoints de acceso público
- Mantiene múltiples versiones de una API de aplicación
- Establece planes de uso por cliente para monetizar y controlar el uso de las APIs
- Puede mantener un cache de respuestas comunes

Tipos de API

REST APIs	HTTP APIs	WebSocket APIs
Colección de rutas y métodos	Colección de rutas y métodos	Colección de rutas de WebSocket
Para aplicaciones que requieren funciones de administración de API	Para microservicios	Para aplicaciones en tiempo real
Permite *cross-origin resource sharing* (CORS)	Menor latencia y menor costo que las API REST	Establece una sesión entre el cliente y los servicios de backend
Stateless	Soporta CORS	*Stateful*
	Stateless	

API Gateway

Integración con backend



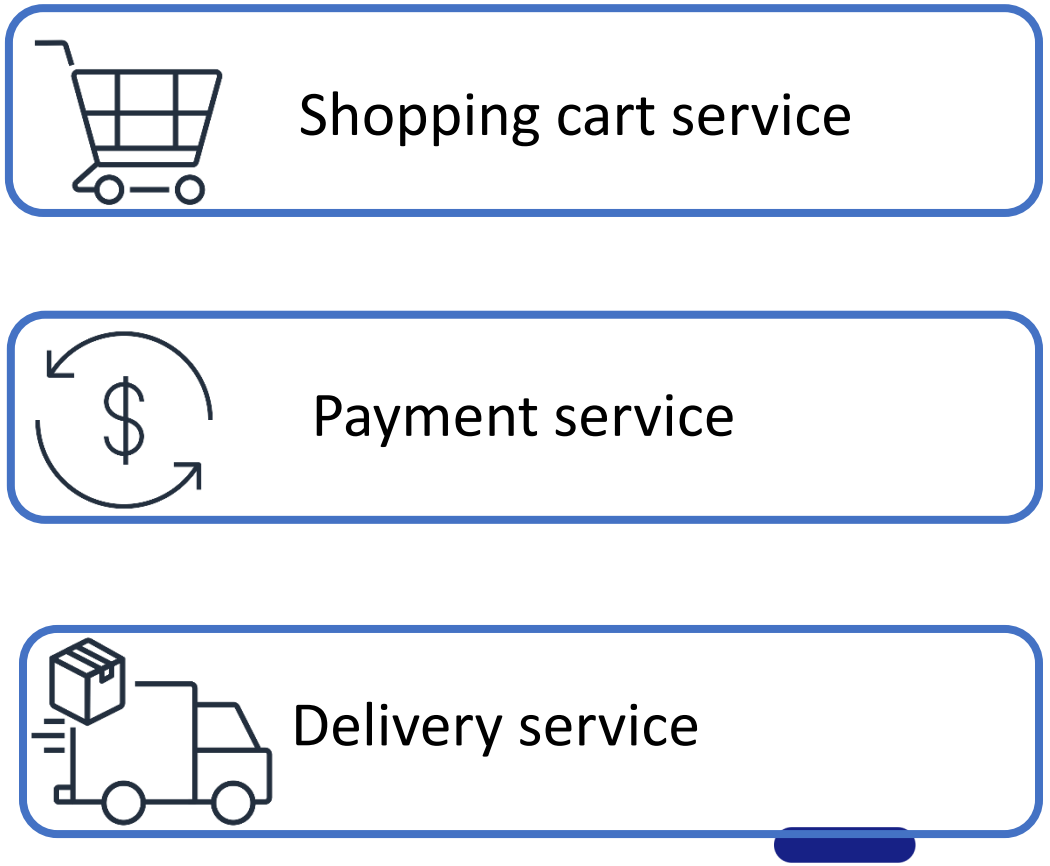
Actividad. API Gateway

Descomponer una aplicación monolítica en microservicios

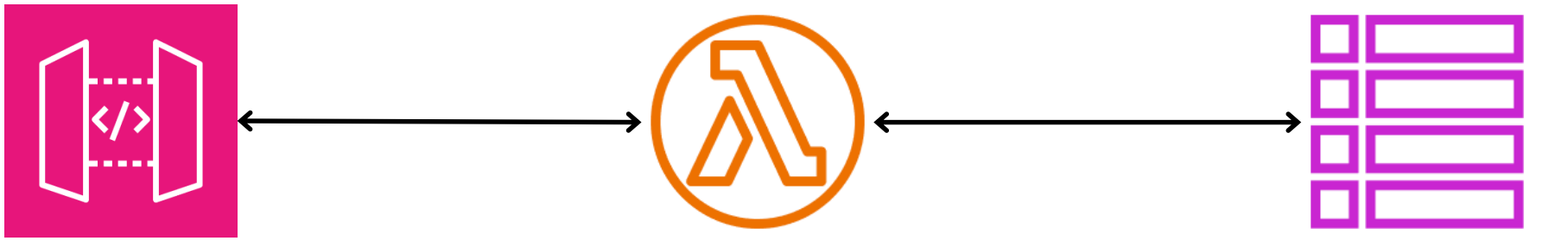
UNIVERSIDAD AUSTRAL

Aplicación monolítica

Aplicaciones basadas en servicios



Solución: microservicio carrito

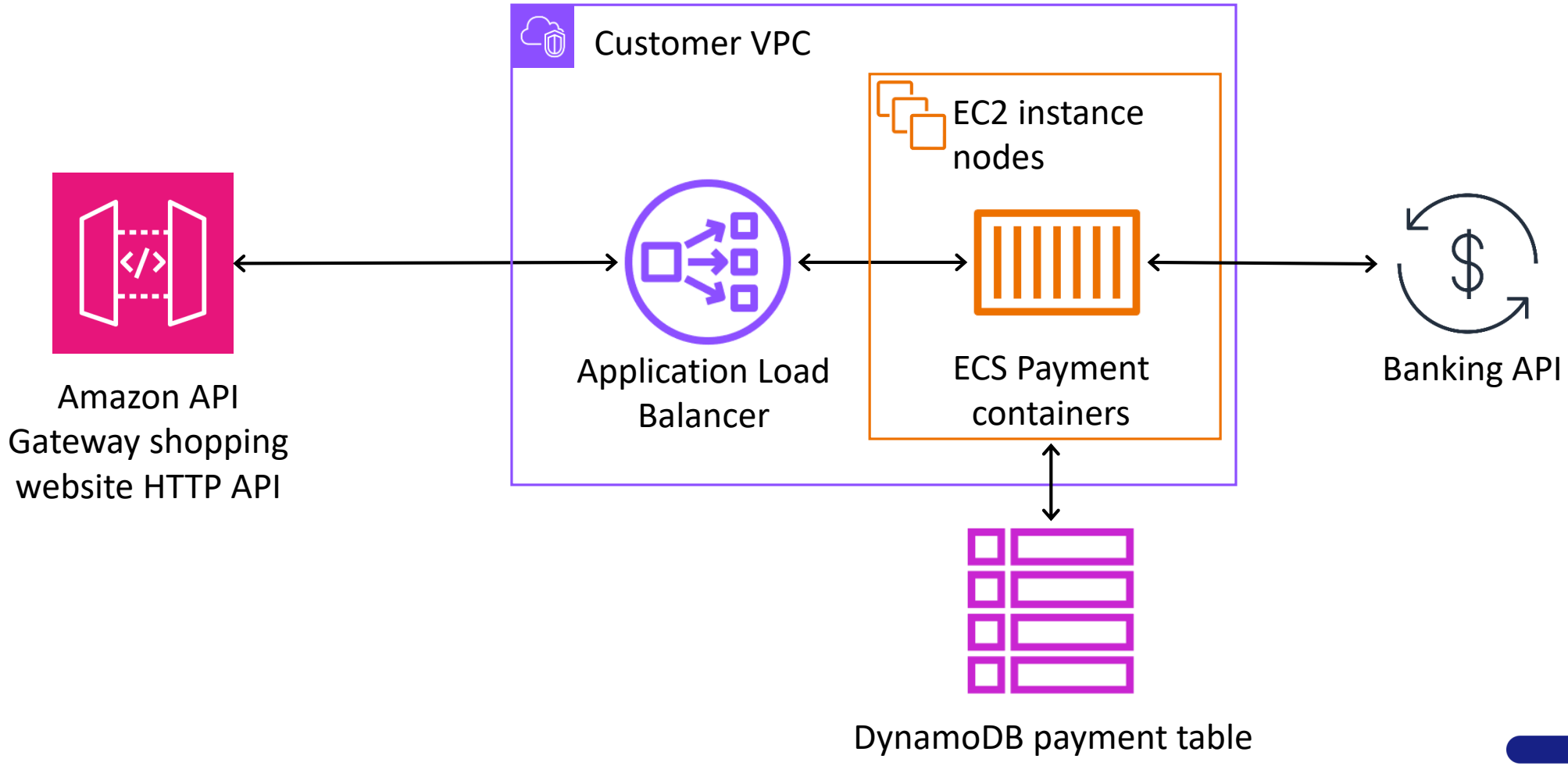


Amazon API Gateway shopping website HTTP API

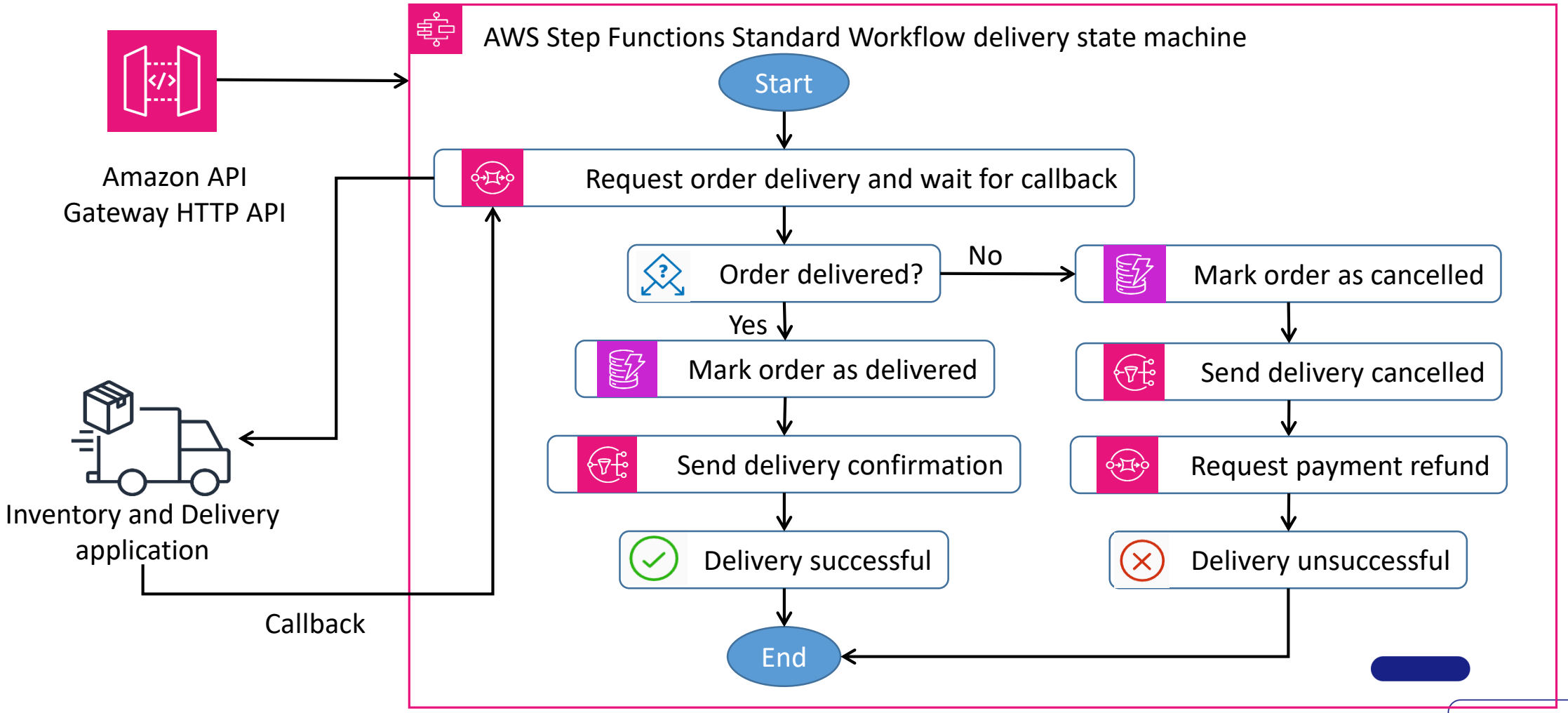
Lambda shopping cart function

Amazon DynamoDB shopping cart table

Solución: microservicio de pago



Solución: microservicio de circuito de distribución



Resumen

Amazon API Gateway permite crear, publicar y mantener APIs de aplicación

Proporciona acceso a servicios de AWS y a endpoints de acceso público.

Las API REST se usan cuando se requiere control y administración total de las APIs.

Las APIs HTTP se usa cuando la prioridad es lograr una menor latencia y menor costo.

Las APIs WebSocket se usan para aplicaciones en tiempo real que requieren una sesión activa.

AWS Well-Architected Framework

UNIVERSIDAD AUSTRAL

AWS Well-Architected Serverless Applications

Reliability

Security

Performance Efficiency

Cost Optimization

Buena práctica: gestión de fallas

Reliability

Buena práctica

Usar mecanismos DLQ para retener, investigar y volver a ejecutar transacciones fallidas.

Ejecutar un roll back de las transacciones fallidas.

Buena práctica: IAM

Buena práctica

Controlar el acceso a las API

Controlar el acceso a la aplicación *serverless*

Buena práctica: Protección de datos

Buena práctica

Cifrar los datos en tránsito y en reposo

Implementar seguridad de aplicaciones en las cargas de trabajo

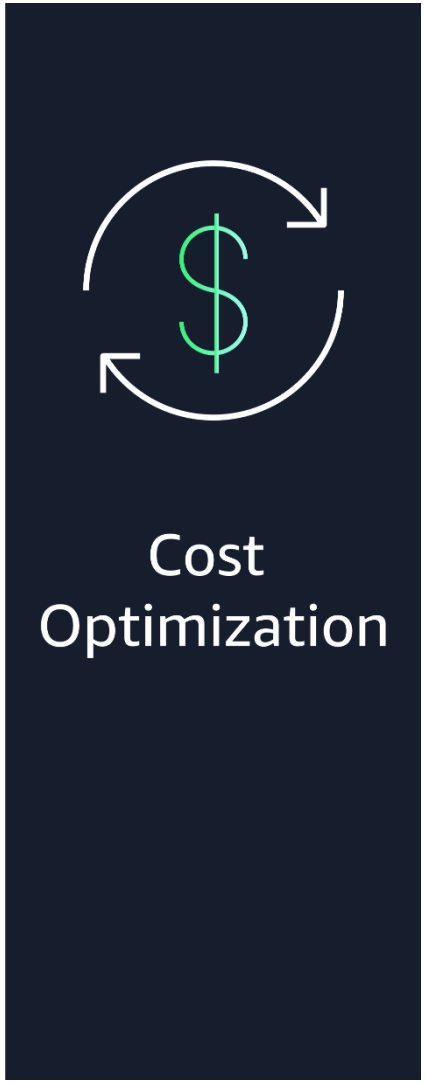
Buena práctica: Selección

Performance Efficiency

Buena práctica

Optimizar el rendimiento de la aplicación

Buena práctica: Costos de recursos



Buena práctica

Optimizar el costo de la aplicación

Usar integraciones de AWS

Resumen

Usar mecanismos DLQ para retener, investigar y volver a ejecutar transacciones fallidas.

Ejecutar *roll back* de transacciones fallidas.

Controlar el acceso a las API *serverless*.

Controlar el acceso a la aplicación *serverless*.

Cifrar los datos en tránsito y en reposo.

Implementar seguridad de aplicaciones en las cargas de trabajo.

Optimizar la performance de las aplicaciones *serverless*.

Optimizar el costo de aplicación.

Cuando estén disponibles, usar integraciones directas de AWS.

Módulo 14

Pregunta de práctica

What is the most effective use of Amazon Elastic Container Service (Amazon ECS) when refactoring a monolithic application to use a microservice architecture?

Identifiquemos las palabras o frases clave:

The following are the key words and phrases:

- Amazon ECS
- refactoring a monolith
- microservice architecture

Módulo 14

Pregunta de práctica

What is the most effective use of Amazon Elastic Container Service (Amazon ECS) when refactoring a monolithic application to use a microservice architecture?

Choice	Response
A	Create services that each provide a distinct function of the application, and run multiple services in a single container that Amazon ECS manages.
B	Port the application to a new image, and run it in a container that Amazon ECS manages.
C	Refactor the application and centralize common functions to create a smaller code footprint.
D	Create services that each provide a distinct function of the application, and run each service in a separate container that Amazon ECS manages.

Módulo 14

Pregunta de práctica

What is the most effective use of Amazon Elastic Container Service (Amazon ECS) when refactoring a monolithic application to use a microservice architecture?

Choice	Response
D	Create services that each provide a distinct function of the application, and run each service in a separate container that Amazon ECS manages.

Muchas gracias.

www.austral.edu.ar

UNIVERSIDAD AUSTRAL