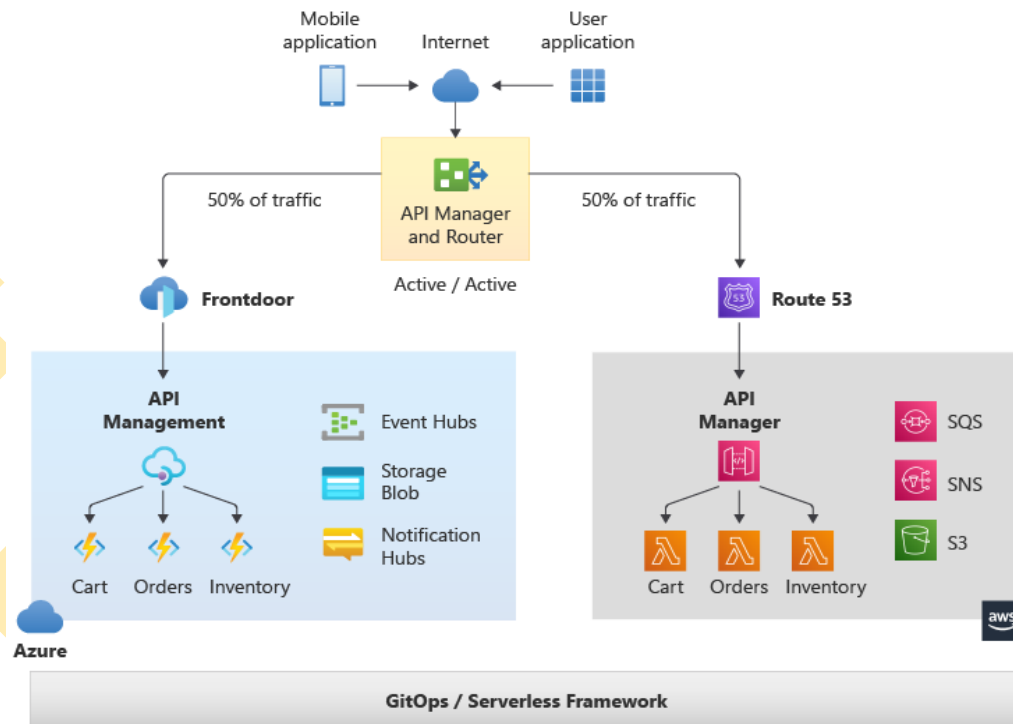


# Arquitectura Serverless, tradeoffs y functionless

Ventajas e inconvenientes y  
sus implicaciones en la  
arquitectura

# Introducción. Principios y decisiones



- La finalidad es delegar una parte del funcionamiento y el despliegue de la aplicación en un tercero para que nos ofrezca un servidor como un Servicio (BaaS, Backend as a Service)
- Reducir tiempo y esfuerzo en cosas inútiles que "ya están hechas"
- Desde el punto de vista del desarrollador no tiene que preocuparse por crear y mantener un servidor

ORACLE

 mongoDB

# SERVERLESS



# ¿Qué es serverless?



- Serverless es pagar por unos recursos computacionales sin tener que preocuparnos de la infraestructura.
- Faas (Function as a Service) nos permite crear aplicaciones con una alta escalabilidad, concurrencia y alta disponibilidad.
- Acarrea una serie de problemas que debemos afrontar.



# Beneficios

Decisiones de arquitectura a bajo coste \_\_\_\_\_

Peticiones aisladas \_\_\_\_\_

Pagas por lo que usas \_\_\_\_\_

Alta disponibilidad y tolerancia a fallos \_\_\_\_\_

Escalabilidad y elasticidad de la arquitectura \_\_\_\_\_

El aislamiento y la seguridad \_\_\_\_\_

Productividad \_\_\_\_\_

Variedad en los equipos de desarrollo \_\_\_\_\_

Facilidad de testing \_\_\_\_\_

Alto nivel de abstracción \_\_\_\_\_





# Inconvenientes

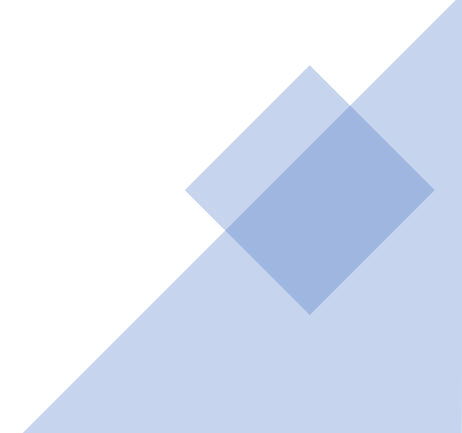
Complejidad añadida \_\_\_\_\_

Riesgo de elevados gastos económicos \_\_\_\_\_

Mayor complejidad de los tests de integración \_\_\_\_\_

Tendencia a comprimir código \_\_\_\_\_

Cold Start \_\_\_\_\_

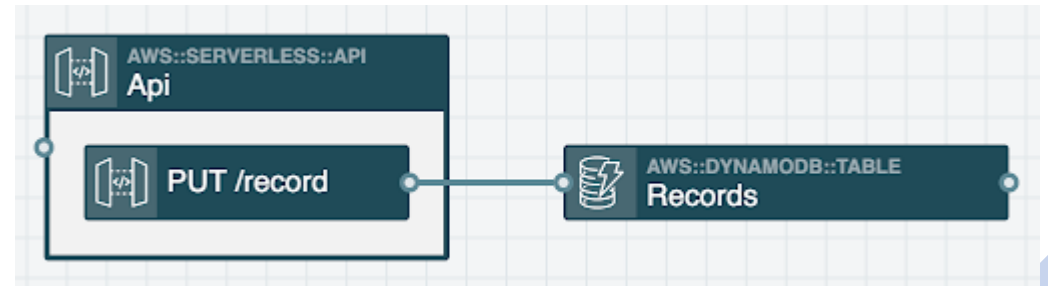
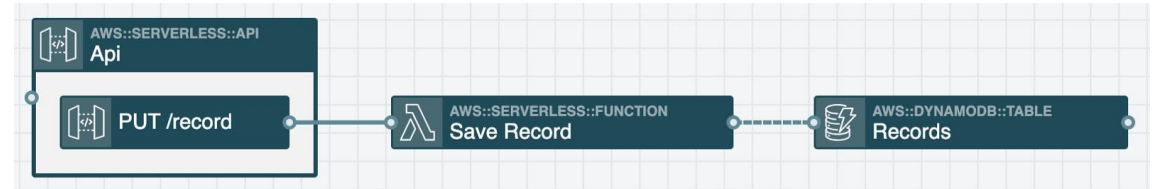


# FUNCTIONLESS



# ¿Qué es functionless?

- Functionless es un enfoque de serverless que trata de llevar al extremo sus conceptos









# Beneficios

Latencia baja \_\_\_\_\_

Mejor mantenibilidad \_\_\_\_\_

Gratis \_\_\_\_\_

Mayor escalabilidad \_\_\_\_\_

Mejor seguridad \_\_\_\_\_

Menos puntos de ruptura \_\_\_\_\_





# Inconvenientes

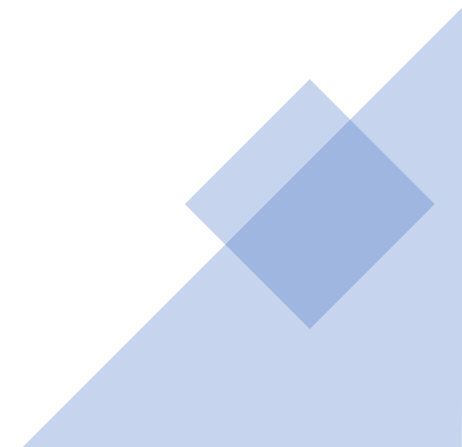
Curva de aprendizaje pronunciada \_\_\_\_\_


Malas herramientas \_\_\_\_\_

Sin componibilidad \_\_\_\_\_

Riesgo de integridad de datos \_\_\_\_\_

Difíciles de testar \_\_\_\_\_





# ¿Serverless o functionless?

Dos cosas a tener en cuenta

¿Tiene lógica de negocio?

¿En el futuro podríamos querer cambiar el enfoque?

