ensitech®

Arquitecturas de Microservicios en Azure:

Contenedores y Serverless

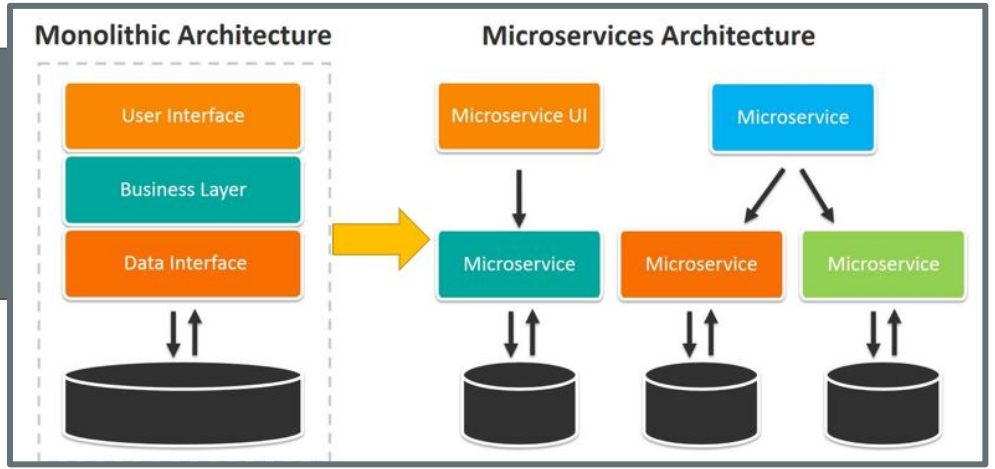
Agenda

- Microservicios
- Proyecto Real
- Diseño basado en Dominios (DDD)
- Arquitectura general
- Diseño de Microservicio
- DevOps (CI/CD)
- Arquitectura de Despliegue en Contenedores
- Arquitectura de Despliegue en Serverless
- Conclusiones y Lecciones Aprendidas

Microservicios

Básicamente, consiste en diseñar componentes con alta granularidad, responsabilidad mínima, bajo acoplamiento y segmentación del dominio de datos.

- Interfaces bien definidas
- Unidades independientes de codificación
  - Equipo de desarrollo pequeño (1-3 personas)
- Unidades independientes de despliegue
  - CI/CD
- Unidades independientes de ejecución
  - Infraestructura
  - Escalabilidad
  - Redundancia
- Deben ser ligeros (smaller codebase)
- Deben hacer “una sola cosa bien” (focus on their own area)



"Doing microservices just because everyone else is doing it is a terrible idea". Sam Newman, 2019; Monolith to Microservices.

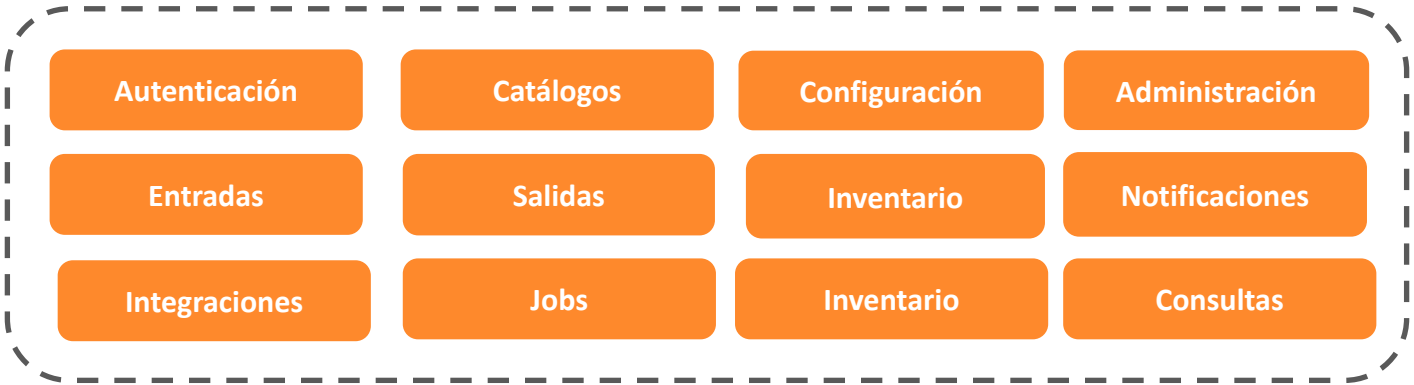
Proyecto

Sistema de inventario, con autenticación, administración (usuarios, perfiles), manejo de almacenes (entradas, salidas), notificaciones, catálogos, documentos, configuración y reportes.

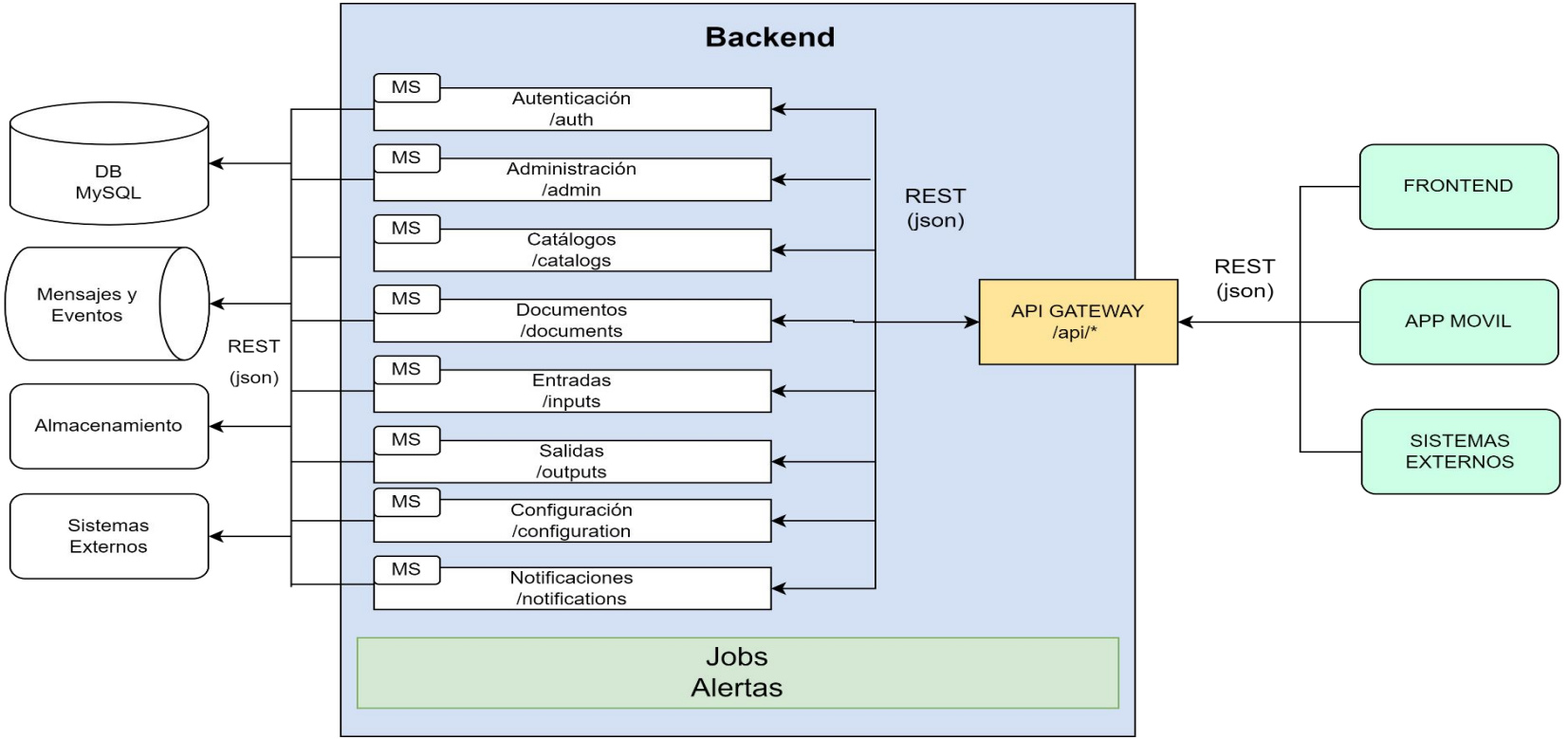
Diseño basado en dominios

El diseño basado por el dominio (DDD) propone un modelado basado en la realidad de negocio con relación a sus casos de uso. Describe áreas independientes como contextos delimitados, cada contexto delimitado está correlacionado con un microservicio.

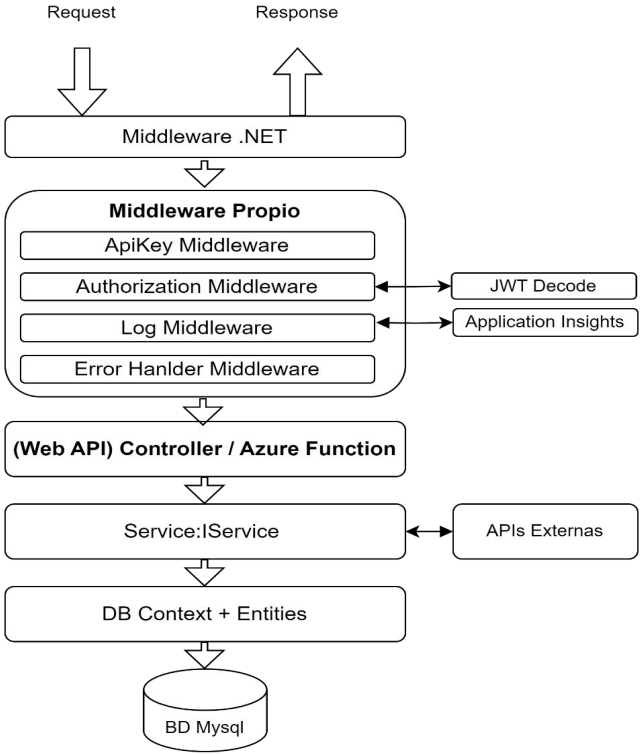
Uno de los patrones más usados para modelar la arquitectura de microservicios es **Bounded Context**, del DDD, y consiste en dividir lógicamente nuestro modelo de datos en contextos del dominio tanto de negocio como de los datos, con el fin de ir segmentando nuestro diseño global.



Arquitectura General

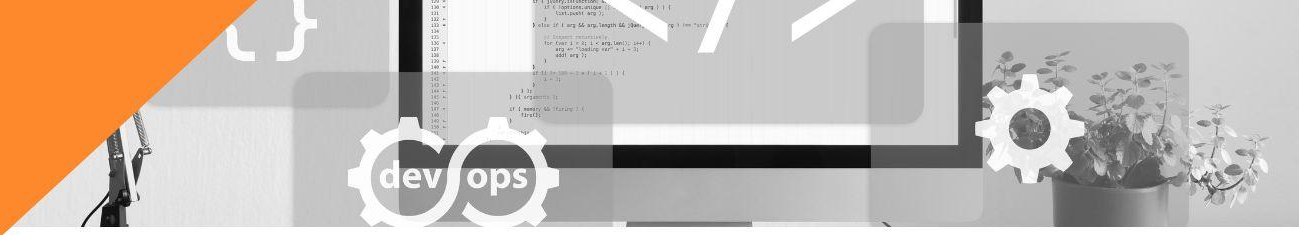


Diseño de Microservicio

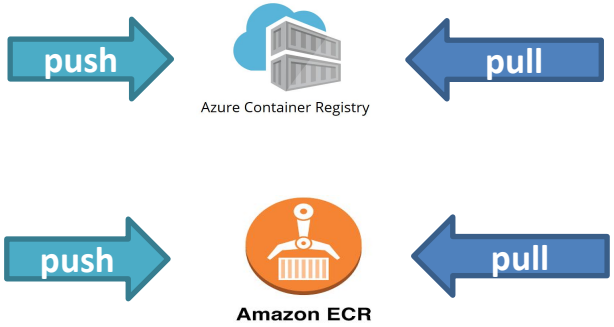


- .NET 7
- Contenedores: Web API
- Serverless: Azure Functions
- Seguridad y Autenticación
  - Api Key (x-api-key)
  - JWT (Authorization)
- Entity Framework / Dapper
- AutoMapper
- Event Grid
- Applicatoins Insights

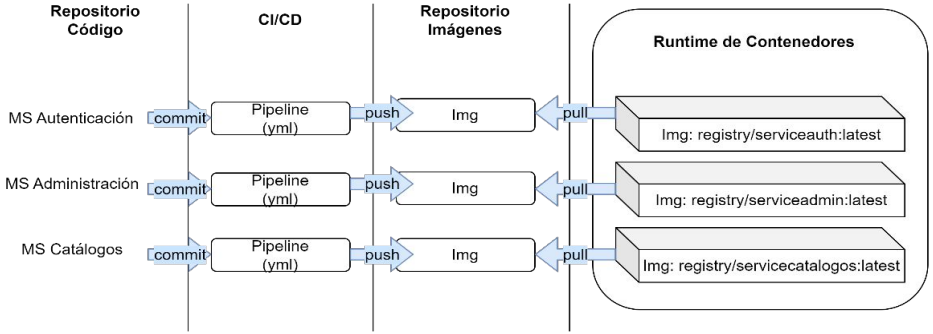
DevOps (CI/CD) para Contenedores

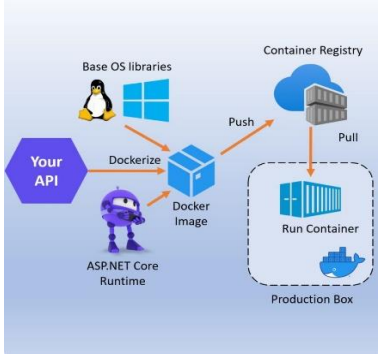


Proceso Automatizado (DevOps / Pipeline)

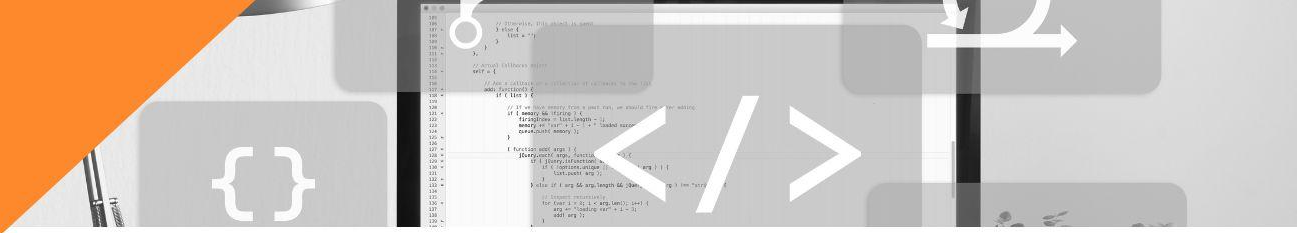


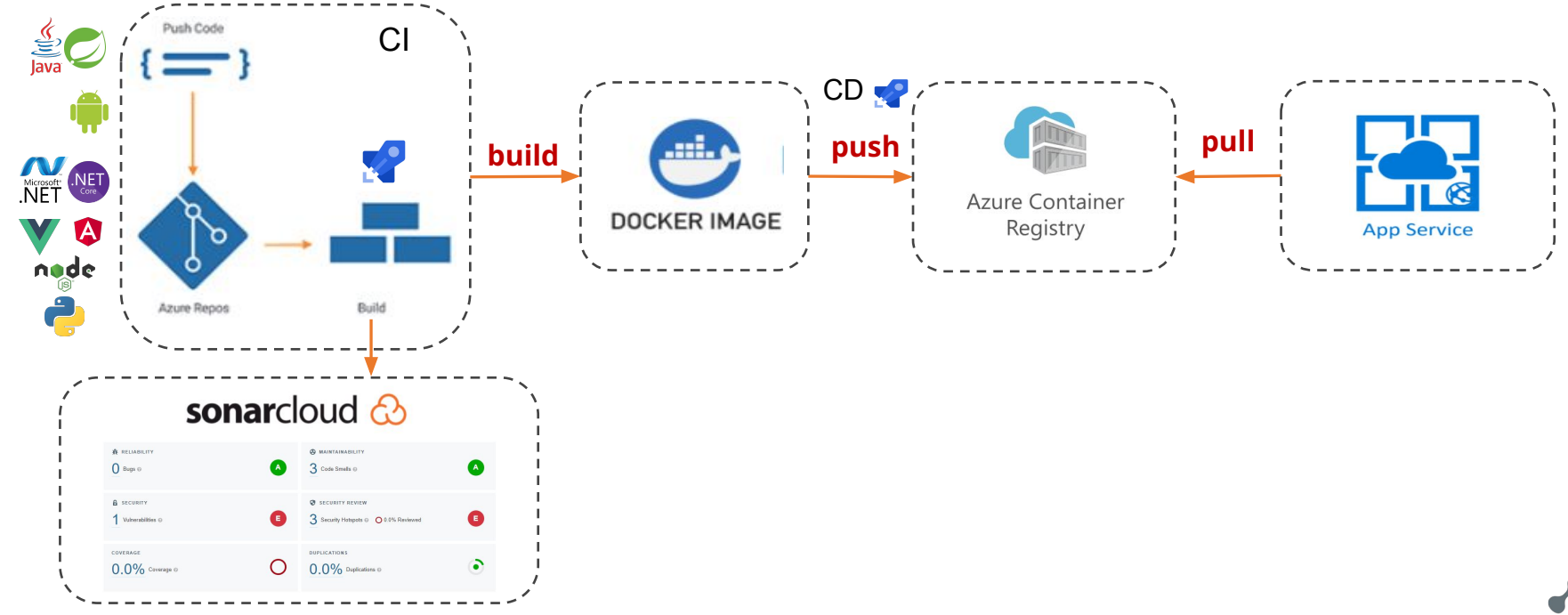
Cualquier tecnología que corra contenedores



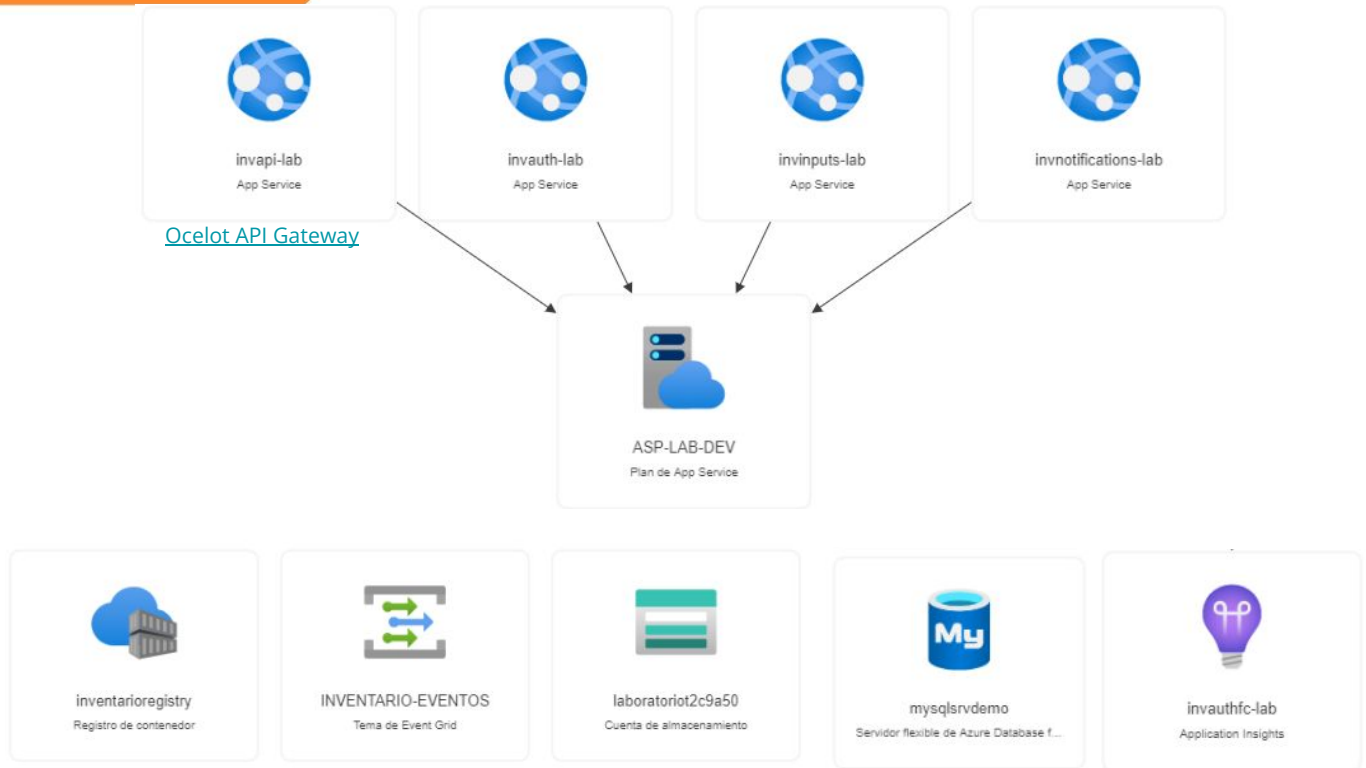


DevOps (CI/CD) para Contenedores

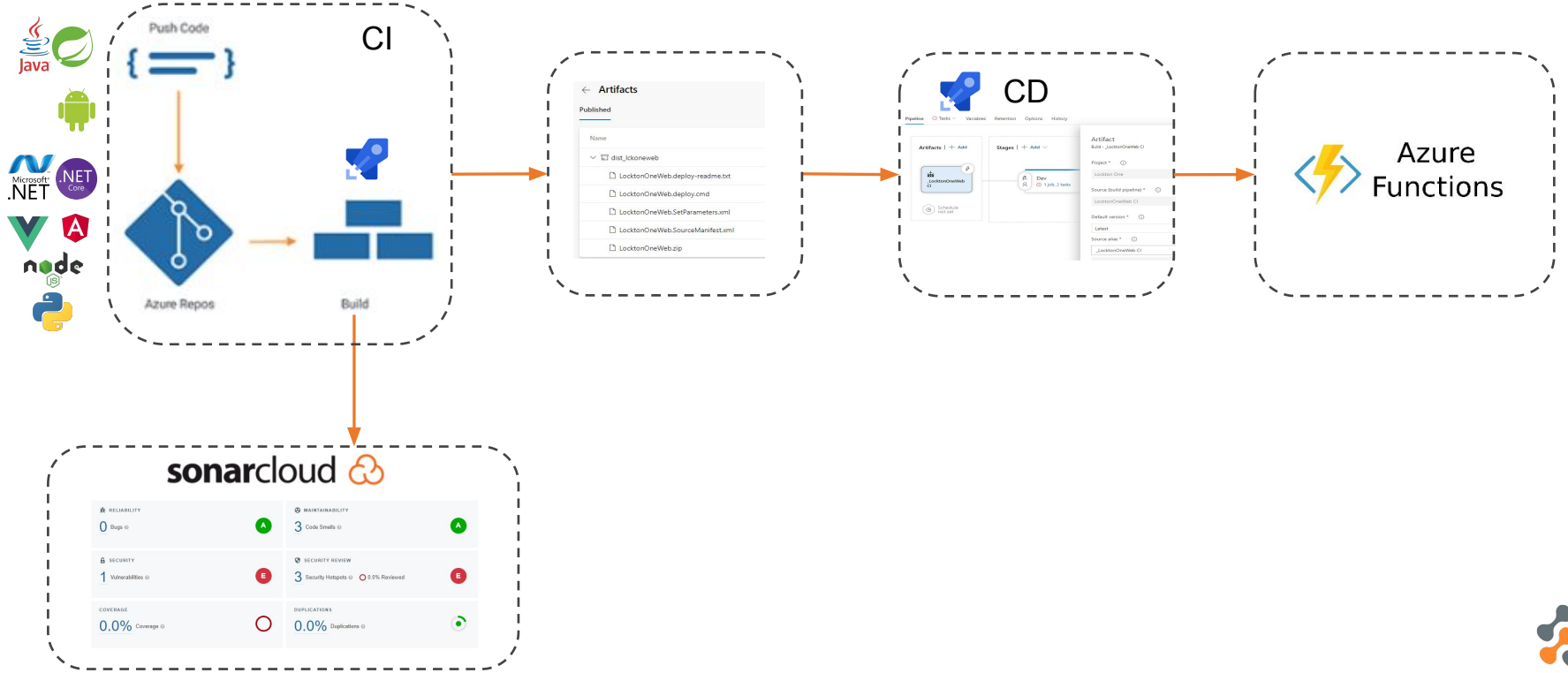




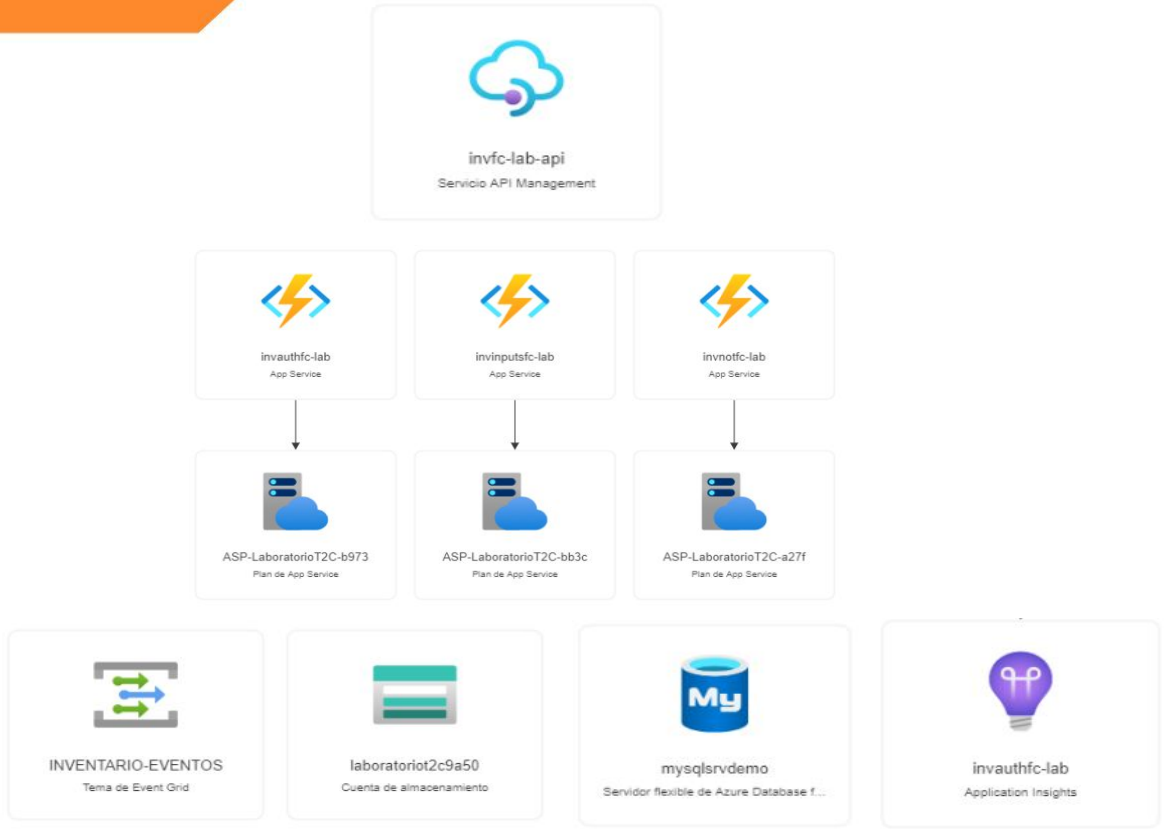
Arquitectura de Despliegue en Contenedores



DevOps (CI/CD) para Azure Functions



Arquitectura de Despliegue en Serverless



Conclusiones y Lecciones

- Modelado por dominio
- Separación de responsabilidades
- Arquitectura ligera favorece a la agilidad y reduce la curva de aprendizaje
- Posibilidad de varios lenguajes
- Template o repositorio base para crear servicios rápido
- Documentar (rápido) los estándares y contratos (estructura, códigos, errores, paginación)
- Comunicación asíncrona (Event Grid, Queue, gRPC) para notificaciones o eventos.
- Uso intensivo de DevOps (CI/CD), subir contínuamente a desarrollo (Lead time / Deployment frecuency)
- Proceso de releases, brancheo y revisiones establecido.
- Revisión contínua de la Deuda Técnica.

Vacantes TI

- Business Analyst
- SQL Reporting Developer
- GCP Data Engineer
- Azure DevOps Engineer
- AWS DevOps Engineer
- Sr Software Engineer .NET

Enviar CV a: rh@ensitech.com

Q&A
¡Gracias!

ensitech®