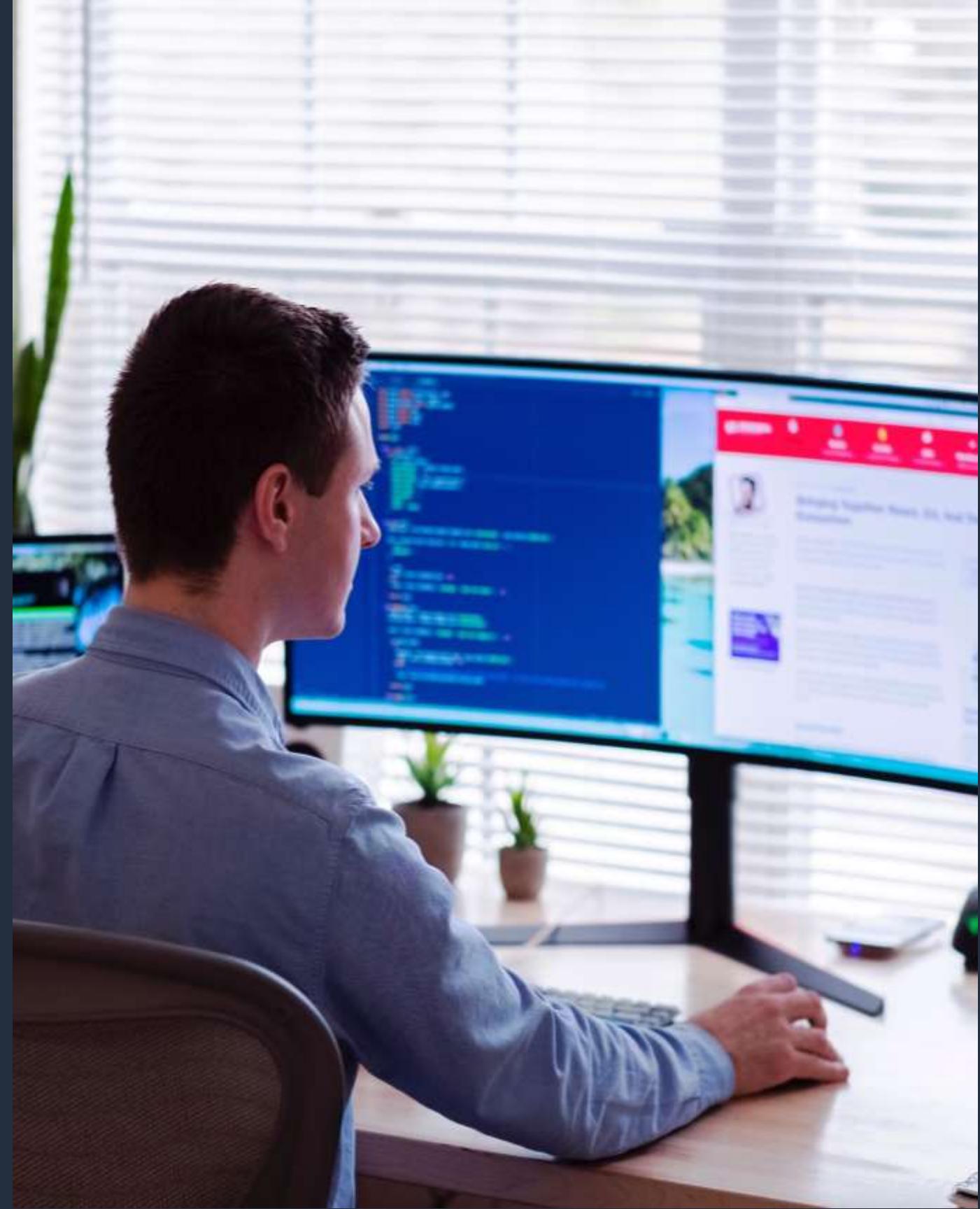


# ¡Bienvenidos!

## Arquitecto en Microservicios Cloud Native

Rubén Landa  
Arquitecto Cloud y Microservicios  
[landaitz104@gmail.com](mailto:landaitz104@gmail.com)  
<https://www.linkedin.com/rubenlanda/>





# Microservicios Cloud Native



# Objetivos del Curso



A dark, semi-transparent background image showing a group of people in what appears to be a classroom or lecture hall. A person in the foreground is pointing towards a screen, which is partially visible and shows some text. The overall atmosphere is focused and educational.

# Sección I

## Previamente -

## Monolitos

# Monolitos

Se refiere a que todo el código de un sistema se encuentra en un único código base o código fuente, que es compilado y produce un único artefacto o pieza de software.

- El código fuente puede estar bien estructurado, mediante clases y paquetes, y puede estar escrito implementando las mejores prácticas de desarrollo sin embargo, **no se encuentra dividido en distintos módulos para su mantenimiento, compilación, despliegue y ejecución.**



# Monolitos

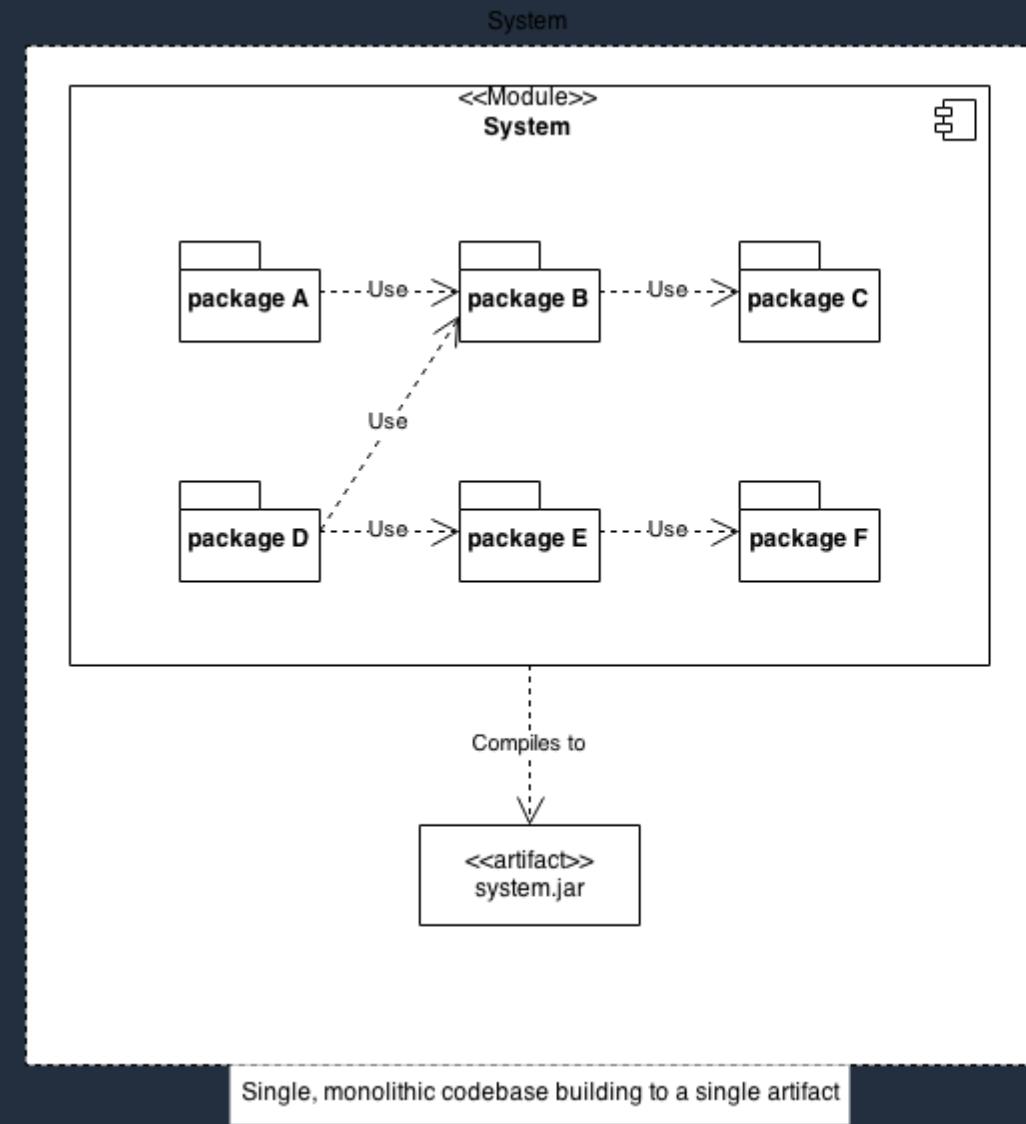
El diseño de un **módulo no-monolítico** mantiene el código fuente del sistema dividido o separado en múltiples módulos o librerías los cuales pueden ser compilados, mantenidos, desplegados y ejecutados de manera separada.

El diseño de un **módulo no-monolítico** puede estar almacenado en diferentes códigos base y diferentes repositorios, lo que permite que puedan ser referenciados o modificados de forma independiente cuando sea necesario.

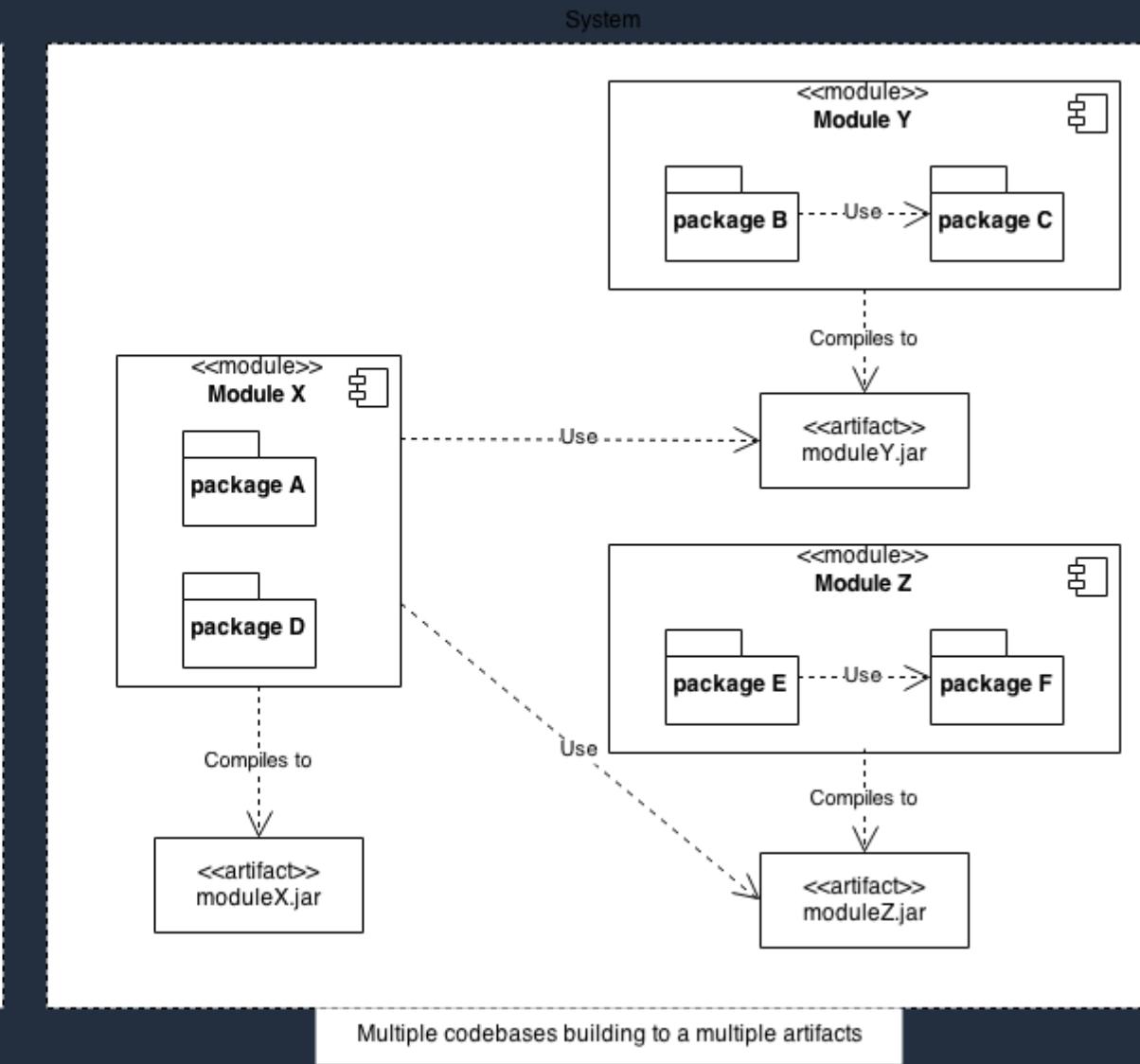


# Monolitos

## Monolito



## No Monolito



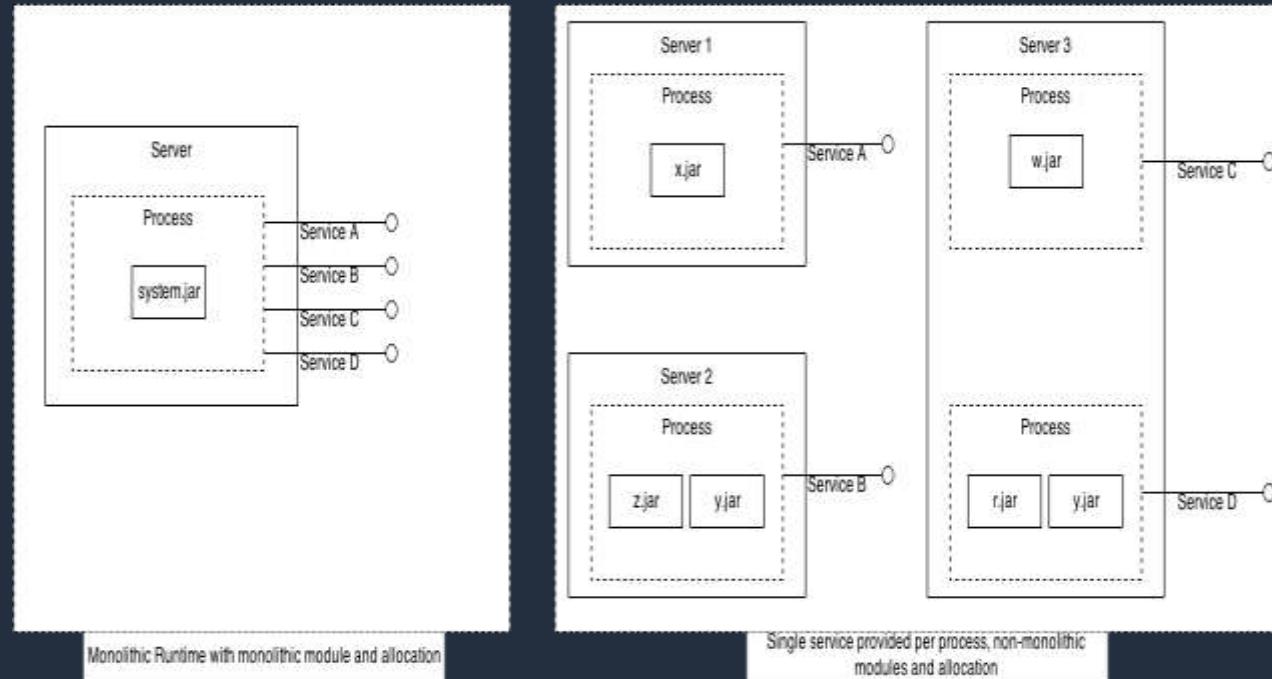
# Monolitos en Ejecución

Ejecución (runtime) monolítica.

**Un monolito en tiempo de ejecución tiene un solo proceso que ejecuta el trabajo del sistema (task).**

Muchos sistemas “tradicionales” se han escrito de forma monolítica y se ejecutan en un único proceso.

La ejecución monolítica es independiente de si la estructura del código fuente del módulo (o sistema) es monolítica o no-monolítica.



Un monolito de tiempo de ejecución a menudo implica un monolito de asignación (asignación monolítica) si solo se despliega en un único nodo o equipo como componente principal.



# Monolitos

## Ventajas:

- Facilidad para desarrollar en entornos no colaborativos.
- Facilidad de “debug & test”.
- Performance, acceder a datos compartidos en memoria es más rápido que comunicación entre procesos y aún más rápido que el “overhead” que ocasiona comunicación entre procesos en diferentes nodos mediante protocolos de comunicación como HTTP.
- Sencillos de construir (compilar).
- Simples de desplegar.
- Facilidad de escalar horizontalmente.
- Más fáciles de implementar seguridad. (**pero!!!! Y APIGEE???**)
- Facilidad de monitoreo, operaciones.
- Facilidad de planificación agile.
- Fáciles de diseñar tradicionalmente (basado en capas).
- Facilidad para la implementación de AOP.



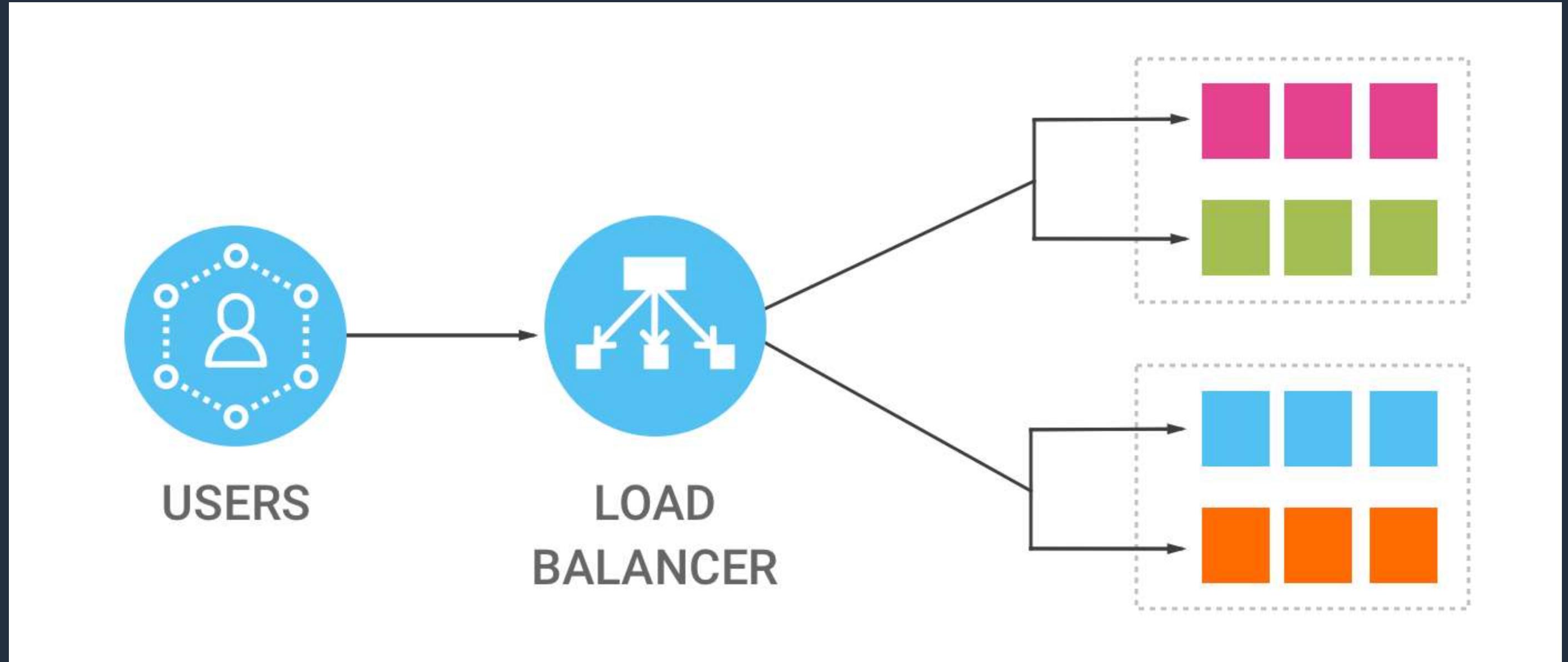
# Monolitos

## Desventajas

- Difíciles de mantener, difíciles de entender.
- Difíciles de evolucionar, difíciles de entender.
- Difíciles de desarrollar en entornos colaborativos.
- Detienen la operativa total del sistema dado un error en el mismo.
- Mantienen un alto acoplamiento entre sus componentes.
- Componentes difíciles de reutilizar.
- Costoso de escalar horizontal y verticalmente.
- Existe un punto en el que no podrá escalar horizontal ni verticalmente debido a limitaciones técnicas.
- Al crecer ampliamente el código base de un aplicativo (**WARS de 1gb? quién**)
- Mayor sobrecarga para IDEs de desarrollo.
- Mayor sobre carga de memoria RAM para el entorno de desarrollo.
- Mayor sobrecarga de contenedores de aplicaciones (contenedores web).
- Difíciles de desplegar.
- Dificultad para implementar CI / CD.
- Requiere un fuerte compromiso con el stack tecnológico utilizado para su desarrollo.



# Qué pasa con la alta demanda de tráfico o uso



# Qué pasa con la alta demanda de tráfico o uso

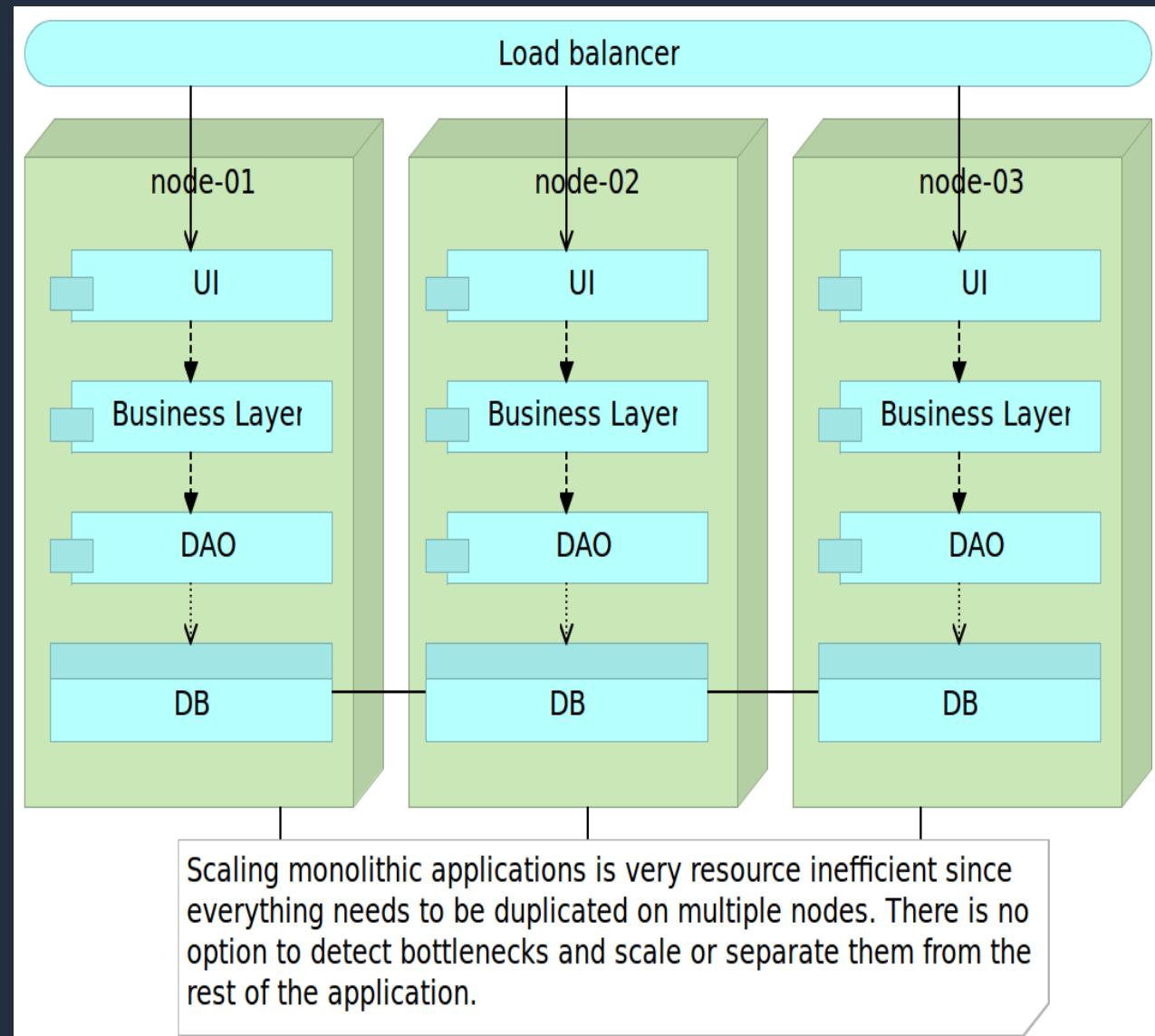
Los sistemas monolíticos pueden escalar en dos dimensiones:

- Escalamiento horizontal.
- Escalamiento vertical.

Un sistema monolítico, escala por completo, es decir, todos sus componentes escalan en la misma proporción y no sólo los componentes o servicios que se requiera

## Ventajas escalamiento horizontal.

- Amplio espectro de escalabilidad, debido a que se podrían agregar tantos servidores (nodos) como sean necesarios.
- Se puede combinar con el escalamiento vertical.
- **Permite la alta disponibilidad del servicio.**
- **Soporta balanceo de carga.**
- Facilidad de escalamiento si se cuenta con el conocimiento requerido.



# Qué pasa con la alta demanda de tráfico o uso

Desventajas escalamiento horizontal.

- Costos medianamente aceptables aunque puede crecer exponencialmente. **(la nube es una alternativa)**
- Requiere demasiado mantenimiento.
- El exceso en el mantenimiento aumenta los costos operativos.
- Requiere una infraestructura más compleja.
- La aplicación requiere estar diseñada para trabajar en cluster.
- Dificultad de implementar en onpremise (en sitio)



# Qué pasa con la alta demanda de tráfico o uso

## Ventajas escalamiento vertical.

- No implica cambios en el sistema a nivel código dado que el escalamiento es a nivel de infraestructura (hardware).
- Fácilidad de implementación.  
Rapidez, **(en la nube)**

## Desventajas escalamiento vertical.

- El crecimiento está limitado por la infraestructura.
- Fallas a nivel hardware debido a su actualizaciones pueden resultar catastróficas.  
**(anecdota de trabajo, siempre haz un plan de rollback)**
- El escalamiento vertical no proporciona alta disponibilidad del servicio.
- Elevados costos para la compra de infraestructura (disco, RAM y procesadores) más reciente y potente. **(en cloud la vida es más feliz)**
- No todos los equipos pueden escalar verticalmente, existe un límite.



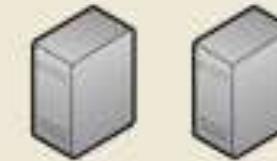
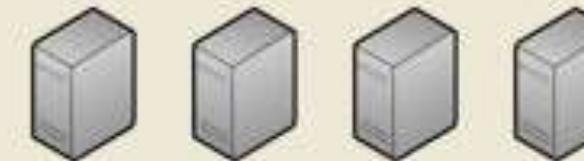
# Qué pasa con la alta demanda de tráfico o uso

Vertical



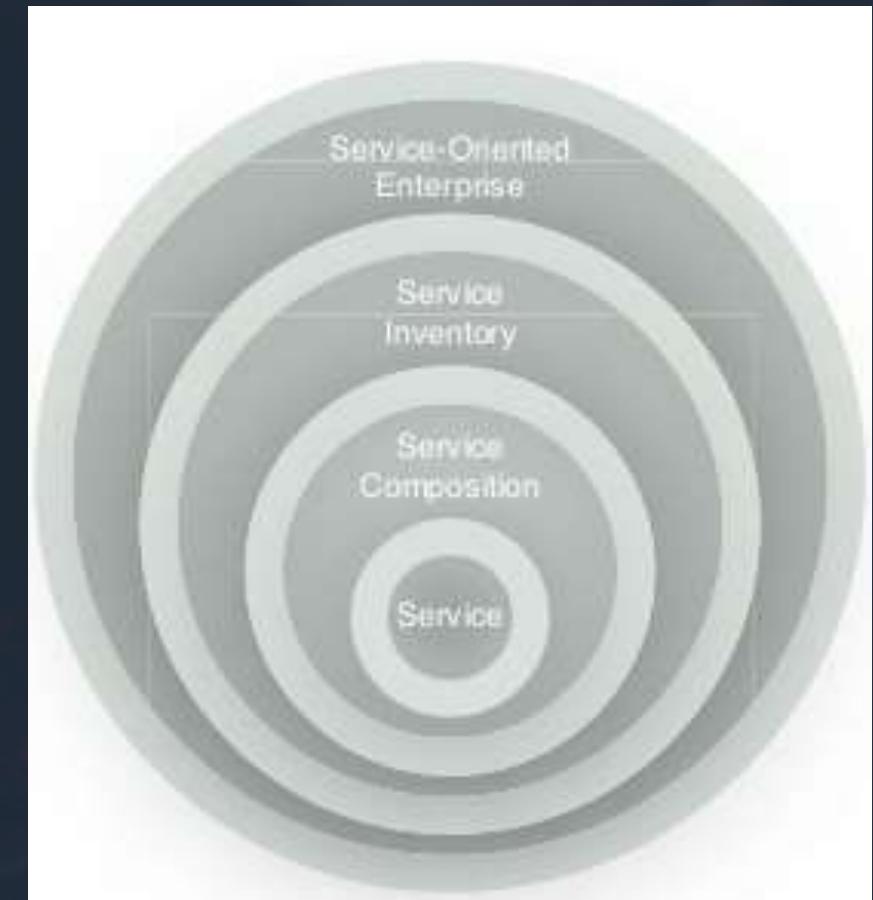
vs.

Horizontal



# Sección I

## SOA – Arquitectura orientada a servicios



# SOA

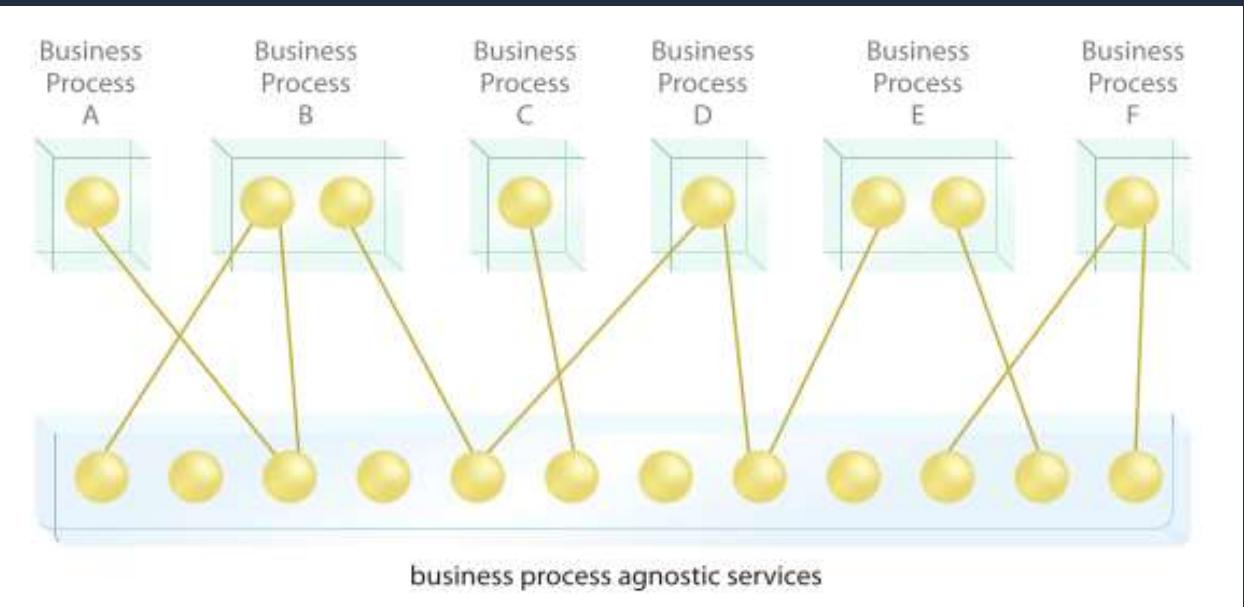
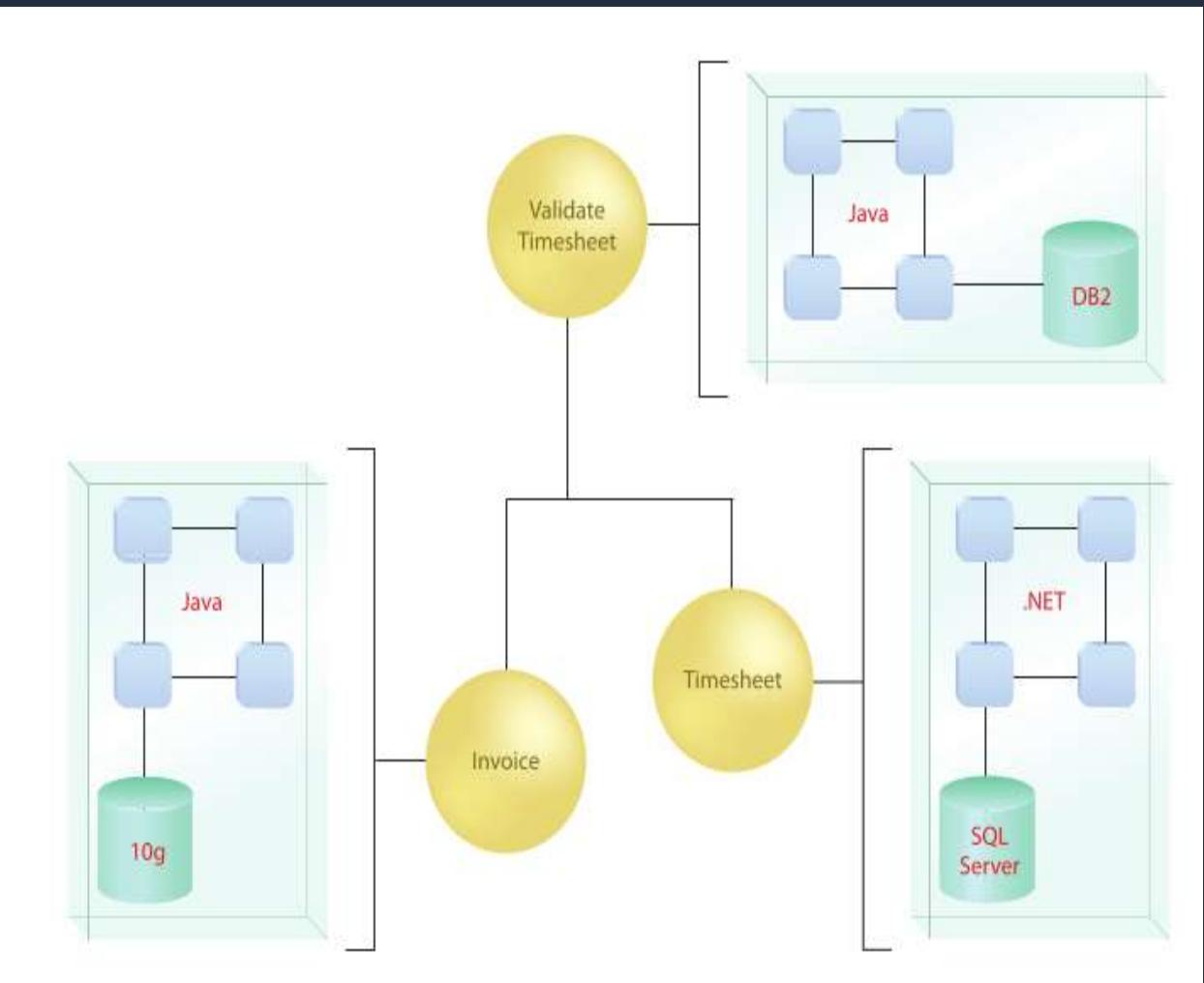
La Arquitectura Orientada a Servicios (SOA) es un framework conceptual que permite a las organizaciones **unir los objetivos de negocio con la infraestructura de TI integrando los datos y la lógica de negocio de sus sistemas separados.**

- SOA es un enfoque de desarrollo de aplicaciones de software empresarial, en el cual **los procesos de negocio se descomponen en servicios**, que después se hacen disponibles y visibles en una red.

SOA es una representación de una arquitectura abierta, extensible y federada basada en composición, que promueve la orientación a los servicios **interoperables e independientes de los proveedores**, los cuales pueden ser identificados en catálogos con gran potencial de **reutilización e implementados como servicios Web**.

- **Cada servicio expuesto provee funcionalidades para poder ser adecuado a las necesidades de la empresa, mientras esconde los detalles inherentes de implementación.**

SOA provee la infraestructura de tecnologías de la información que permite a diferentes aplicaciones intercambiar datos y participar en los procesos de negocio, independientemente del sistema operativo o de los lenguajes de programación con los cuales los servicios y los sistemas interconectados han sido desarrollados.



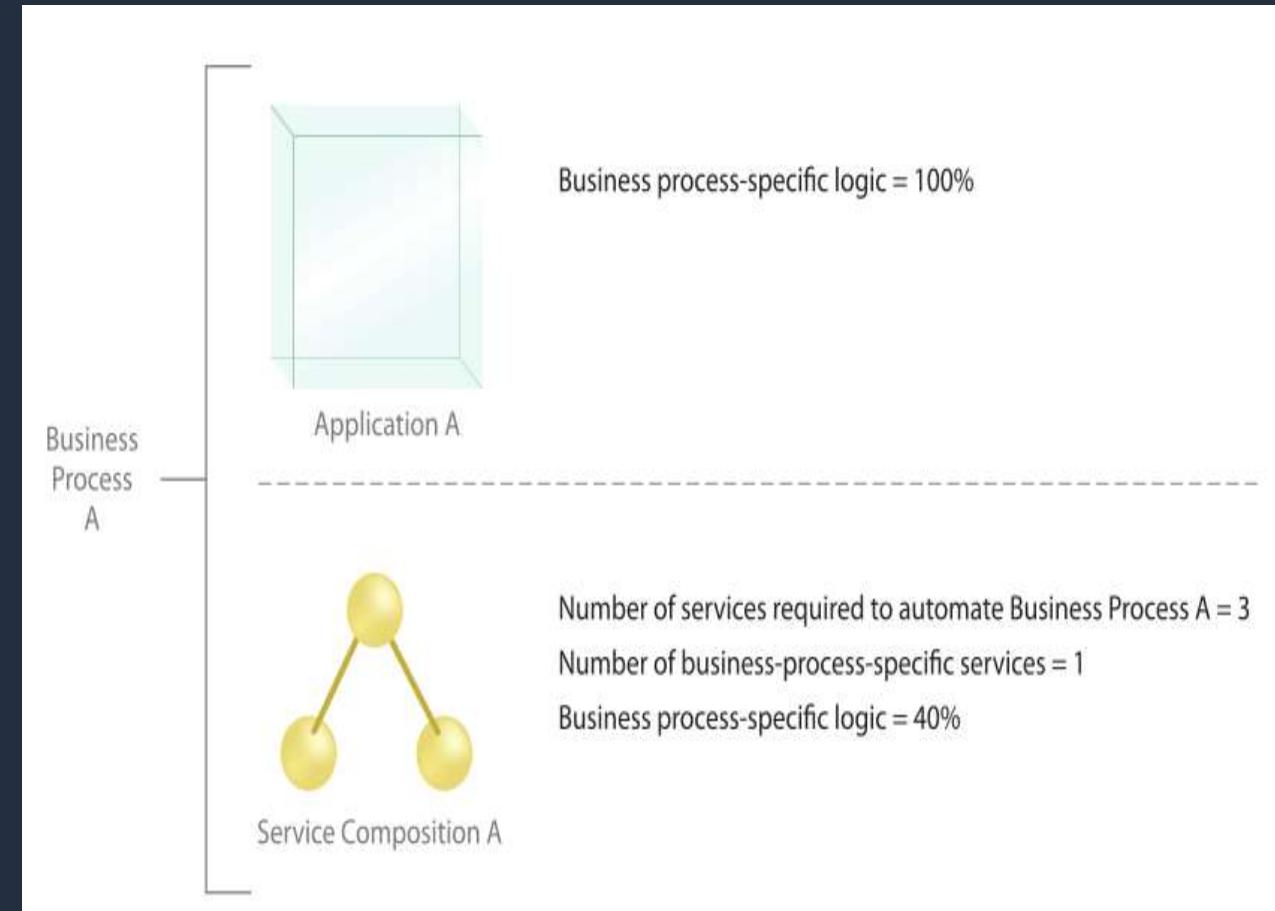
# SOA

Existen diversas definiciones de SOA que incluyen el término “servicios web” o “web-services” sin embargo, es necesario hacer la distinción de estos conceptos y aclarar que SOA no es lo mismo que Servicios Web.

- La Arquitectura Orientadas a Servicios (SOA), a diferencia de los Servicios Web, define y trata un paradigma de orientación a servicios, en tanto que **los Servicios Web son sólo una forma posible de implementar la infraestructura SOA** utilizando una estrategia de implementación tecnológica específica.

**Podemos resumir que SOA es un paradigma arquitectónico que permite el tratamiento de procesos de negocio distribuidos de sistemas heterogéneos, que se encuentran bajo el control o responsabilidad de diferentes propietarios donde sus conceptos clave son:**

- **Disponibilizar un activo de negocio como un servicio.**
- **La interoperabilidad entre diversos lenguajes y aplicaciones,**
- **El bajo acoplamiento entre los componentes.**
- **Los principales actores en SOA son la infraestructura, la arquitectura y los procesos**



El proceso empresarial A puede automatizarse mediante la aplicación A o la composición de servicio A.

la entrega de la Aplicación A puede dar como resultado un cuerpo de lógica de solución que es específica y adaptada para el proceso de negocio.

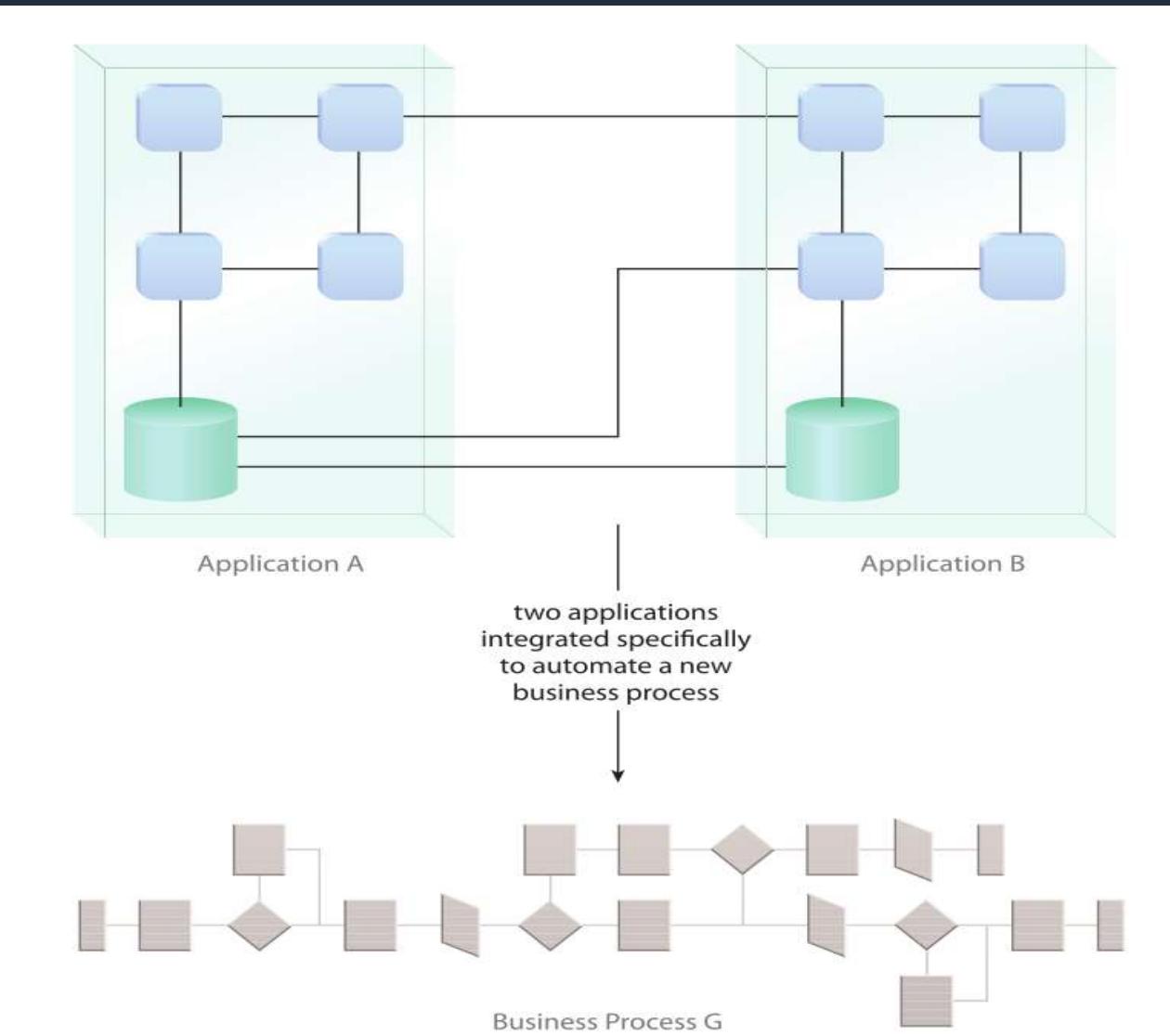
La composición de servicio A estaría diseñada para automatizar el proceso A con una combinación de servicios agnósticos y un 40% de lógica adicional específica para el negocio proceso.



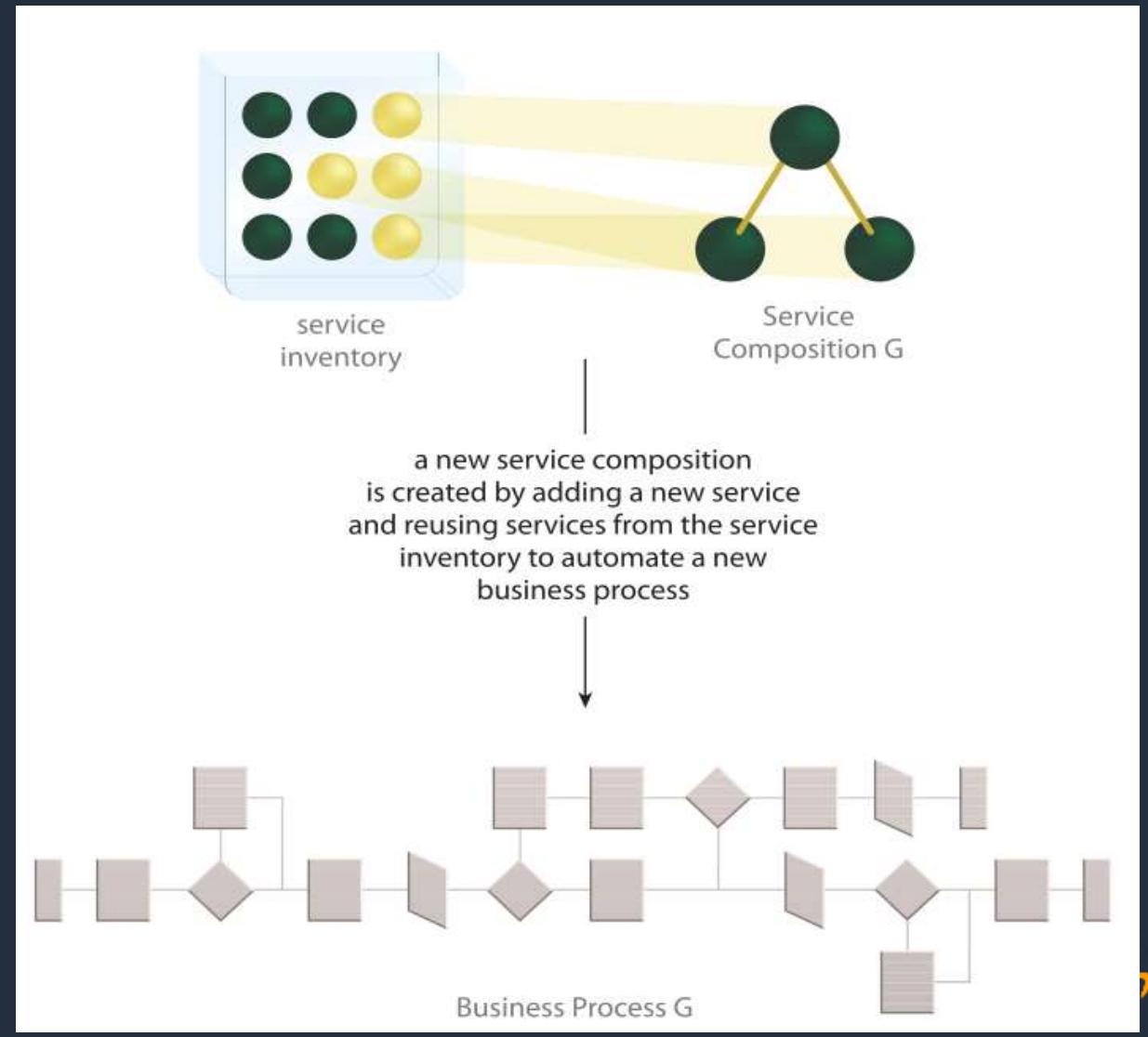
# SOA

## Modelando Proceso de negocio con Servicios

La arquitectura de integración tradicional, compuesta por dos o más aplicaciones conectadas de diferentes maneras para cumplir un nuevo conjunto de requisitos de automatización por el nuevo Proceso de Negocio G).



Reutilización de servicios del inventario, creando una nueva composición para poder crear la funcionalidad de un nuevo proceso de negocio



# Principios de diseño para SOA

## Principios de diseño de orientación a servicios:

- Contratos de servicios estandarizados.
- Bajo acoplamiento.
- Abstracción.
- Reusabilidad.
- Autonomía.
- Sin estado.
- Descubrimiento.
- Composición.

# Principios de Diseño Contrato de servicios estandarizados

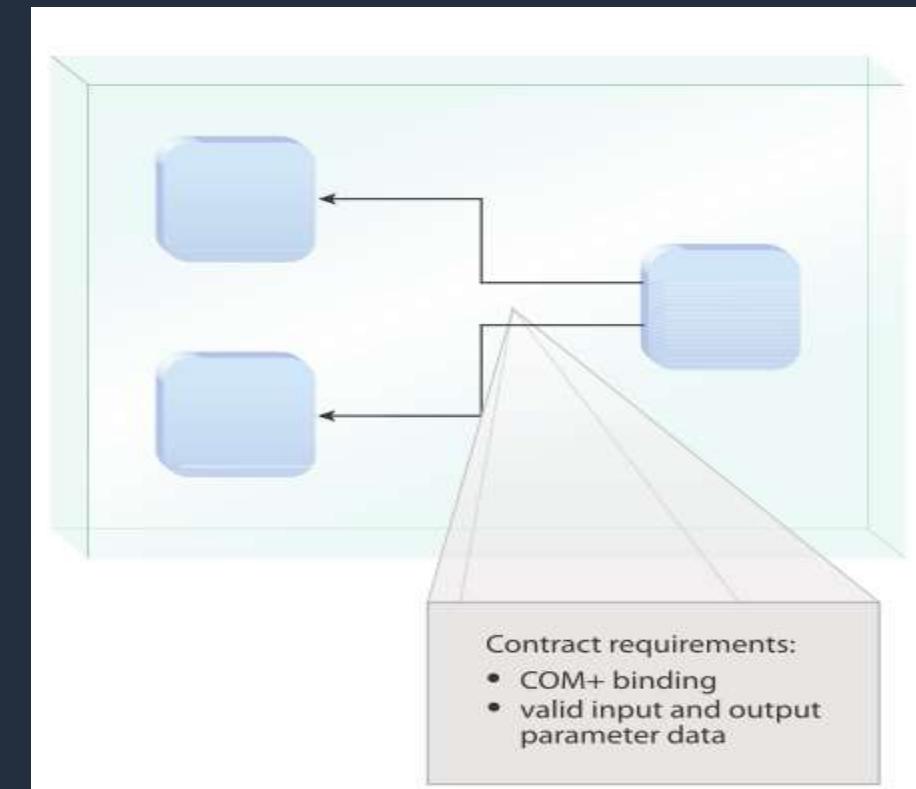
## Contrato de servicios estandarizados

- Cuando un servicio se implementa como un Servicio Web, se debe adherir un contrato o interfaz de comunicaciones explícitamente declarado y definido, colectivamente, por uno o más descriptores del servicio, en el cual debe figurar especificaciones como:

- Nombre del Servicio.
- Forma de acceso.
- Funcionalidades que ofrece.
- Descripción de los datos de entrada de cada funcionalidad que ofrece.
- Descripción de los datos de salida de cada funcionalidad que ofrece.

- Mediante contratos de servicios estandarizados, todo consumidor de servicios accederá a éste mediante su contrato definido, logrando la independencia entre el consumidor y la implementación del servicio.

- De esta forma, se evita el manejo incorrecto de los datos, se evita trabajo innecesario al momento de la invocación del servicio y también se pone de manifiesto de la existencia de un modelo de datos a nivel de servicio, siendo esta una de las primeras necesidades que se debe tener en cuenta al momento de definir servicios.

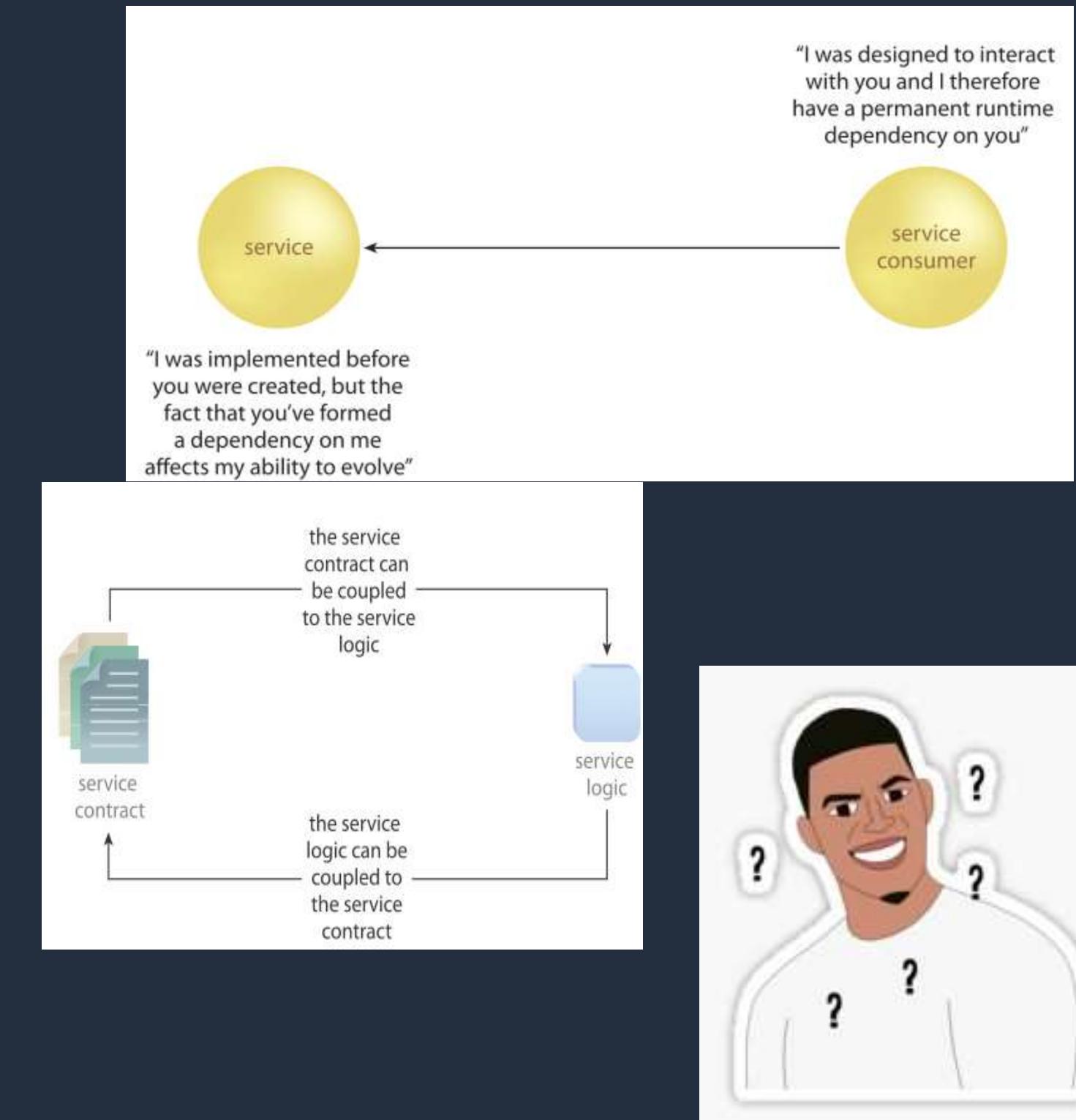


# Principios de Diseño Acoplamiento

Este principio hace referencia a que los servicios tienen que ser independientes los unos de los otros.

Para lograr el bajo acoplamiento entre los servicios, se define que el único medio de acceso a los servicios es el contrato o interfaz, logrando así la independencia entre el servicio que se va a ejecutar y el que lo llama.

El bajo acoplamiento permite que los servicios sean totalmente reutilizables.



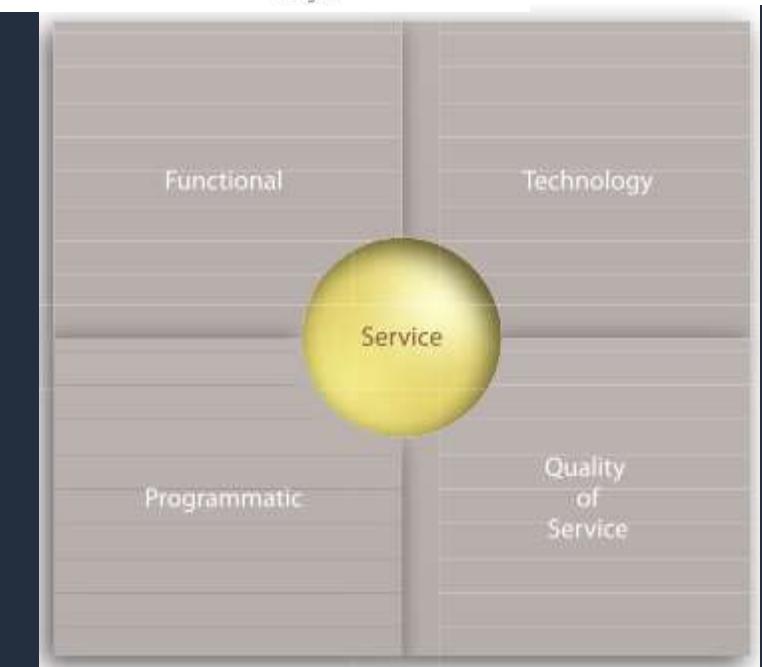
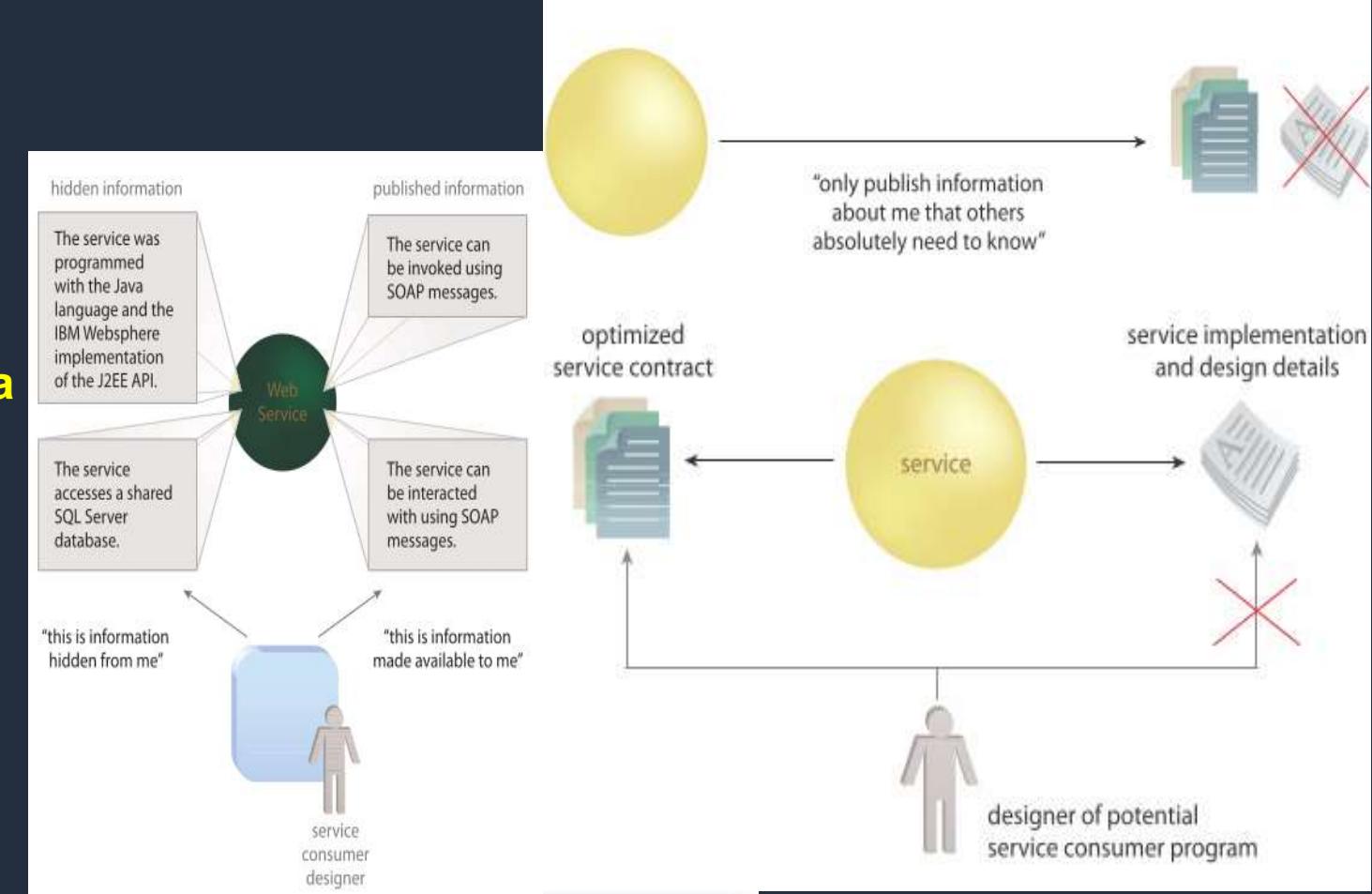
# Principios de Diseño Abstracción

A nivel fundamental, **la abstracción permite ocultar los detalles de implementación de un servicio, tanto como sea posible.**

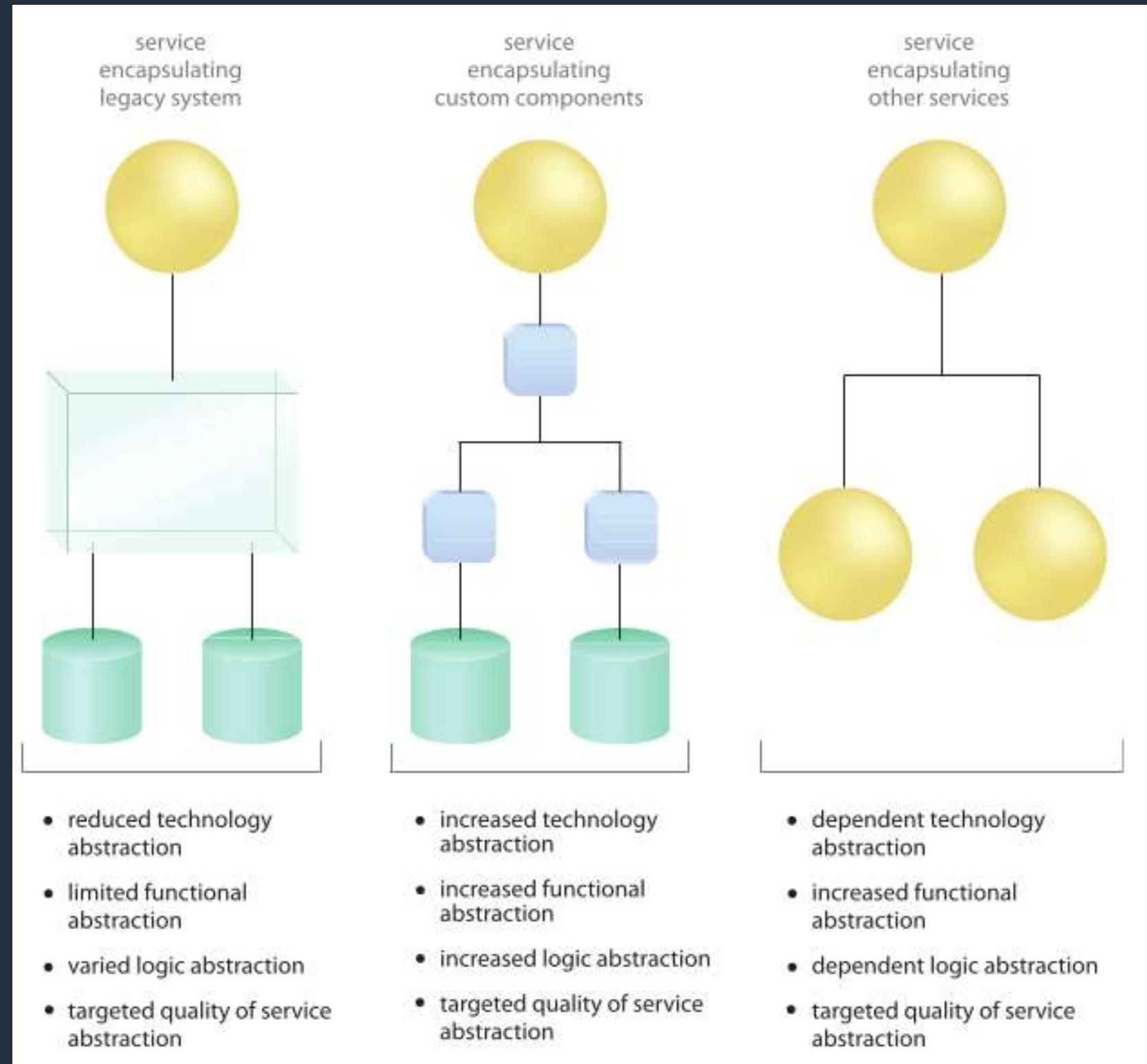
- **El principio de abstracción permite encapsular el servicio como una “caja negra”, del cuál no se saben los detalles y únicamente esta definido por su contrato, habilitando así el bajo acoplamiento**, que a su vez, la definición y especificación del contrato es el mínimo acoplamiento posible entre el servicio y un consumidor.

**El uso de estándares permite definir interfaces uniformes que esconden la lógica del servicio respecto del entorno que le rodea (sistemas proveedores y consumidores).**

**Este nivel de abstracción permite centrarse exclusivamente en la especificación del servicio, sin incluir información tecnológica ni de ninguna otra naturaleza, más allá de la propia especificación estándar del servicio, desde el punto de vista del negocio al que sirve, la cual queda definida en su contrato.**



# Principios de Diseño Abstracción



# Principios de Diseño Reusabilidad

La reusabilidad (reutilización) es el principio de la arquitectura SOA, cada servicio debe ser analizado, diseñado y construido de manera que su uso pueda ser explotado al máximo por otros sistemas consumidores del servicio.

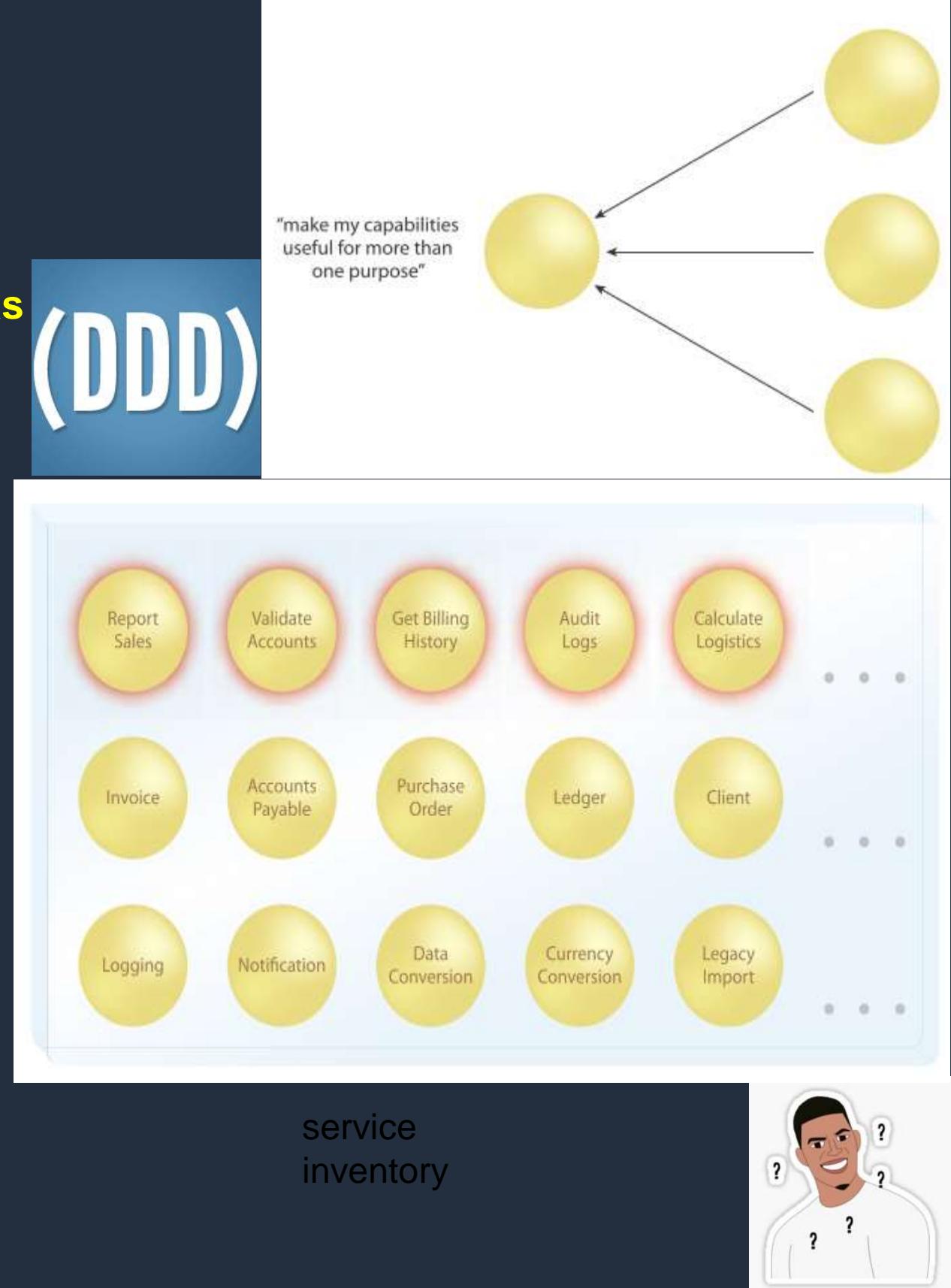
Los servicios deben de ser definidos de tal forma que puedan utilizarse

en diferentes contextos y satisfacer distintas objetivos de negocio, únicamente centrandose en las capacidades del negocio y no en alguna tecnología específica

De esta manera, los servicios pueden ser reusables dentro de la misma aplicación, dentro del dominio de aplicaciones de la empresa o incluso dentro del dominio público.

Por otro lado, estos servicios reutilizables deben estar diseñados de manera tal que su solución lógica sea independiente de cualquier proceso de negocio o tecnología en particular.

Con este principio se busca reducir las posibilidades de duplicación de lógica.



# Principios de Diseño Autonomía

Este principio refiere a que todo servicio debe tener su propio entorno de ejecución, logrando que dicho servicio sea totalmente Independiente.

- De esta forma sea asegura la reusabilidad del servicio desde el punto de vista de la plataforma de ejecución.

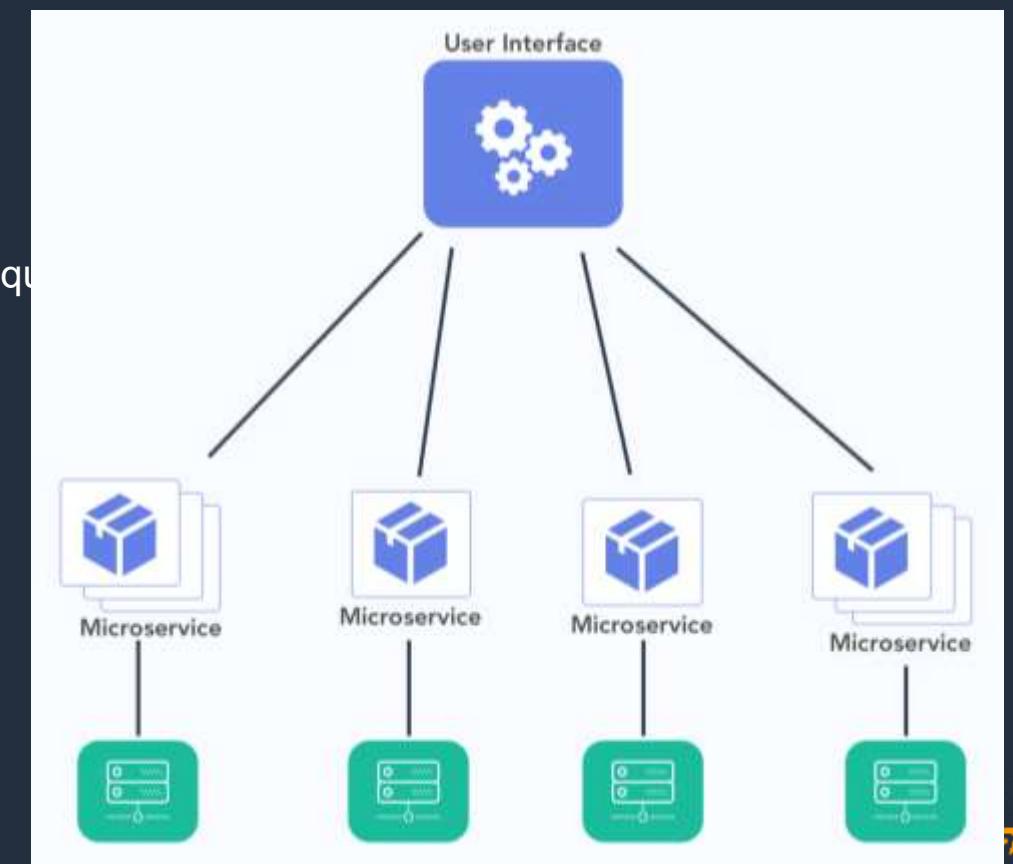
## Tipos de Autonomía

### Runtime Autonomy

- Rendimiento de ejecución en tiempo de ejecución constantemente aceptable
- Un mayor grado de fiabilidad de rendimiento
- La opción para que se aísle en respuesta a seguridad, confiabilidad o seguridad específicas requeridas
- Un mayor nivel de previsibilidad conductual (especialmente cuando se accede Simultáneamente)

### Design-Time Autonomy (governance)

- La capacidad de escalar un servicio en respuesta a mayores demandas de uso
- La opción de modificar o mejorar aún más el entorno de alojamiento de un servicio
- La libertad de aumentar, actualizar o reemplazar la tecnología de un servicio en respuesta a nuevos requisitos o un deseo de aprovechar las nuevas innovaciones



# Principios de Diseño Sin estado

**Los servicios no deben de almacenar algún tipo de información con referencia a los consumidores del mismo.**

La gestión de excesiva carga de información minimiza y deteriora la escalabilidad del servicio lo cual afecta gravemente la disponibilidad del servicio.

De forma ideal, todos **los datos que necesita un servicio para su ejecución deben provenir directamente de los parámetros de entrada que provee el consumidor del servicio.**



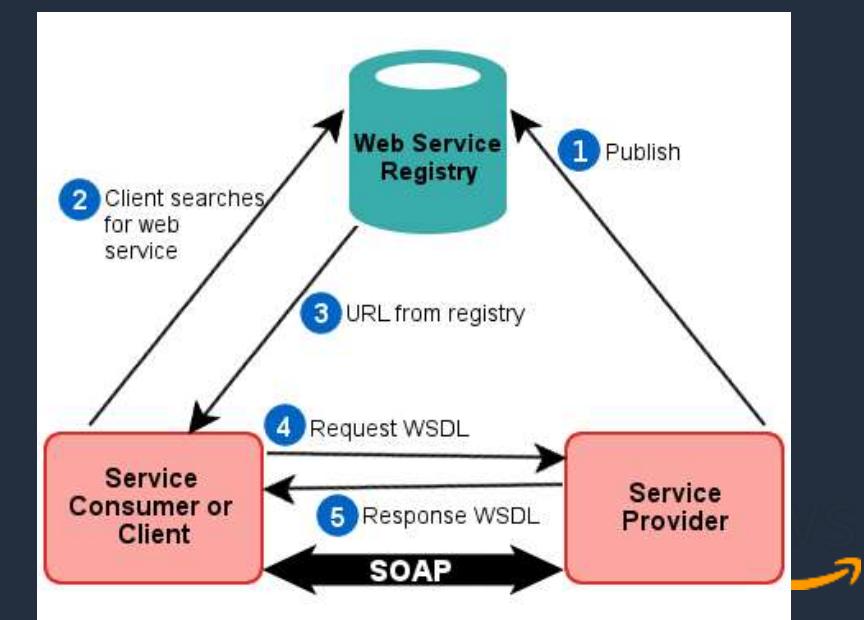
# Principios de Diseño Descubrimiento

Todo **servicio expuesto** debe poder ser descubierto de alguna forma para que pueda ser utilizado.

El descubrimiento de servicios evita la duplicidad accidental de servicios que proporcionen una misma funcionalidad.

Para el caso de aplicar descubrimiento de servicios en una arquitectura SOA basada en Servicios Web, se deberá contar con la publicación de los servicio a traves de un registro UDDI (Universal Description, Discovery and Integration).

## Service Registry



# Principios de Diseño Composición

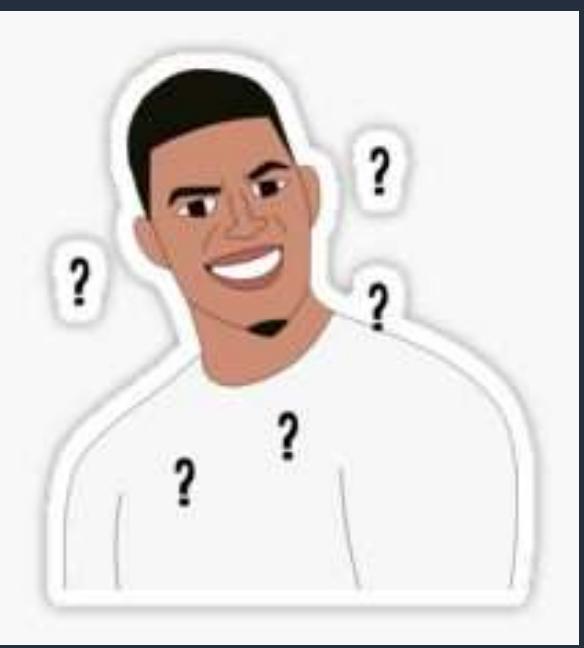
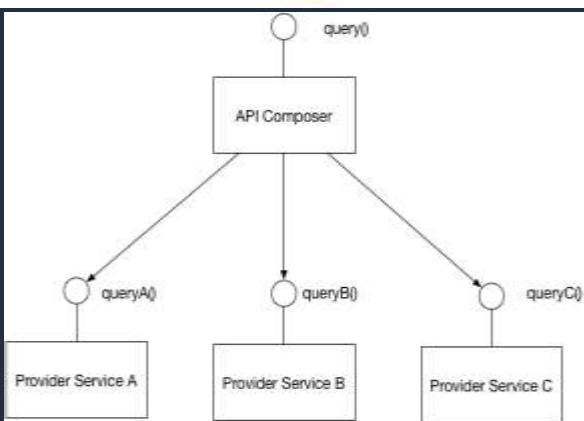
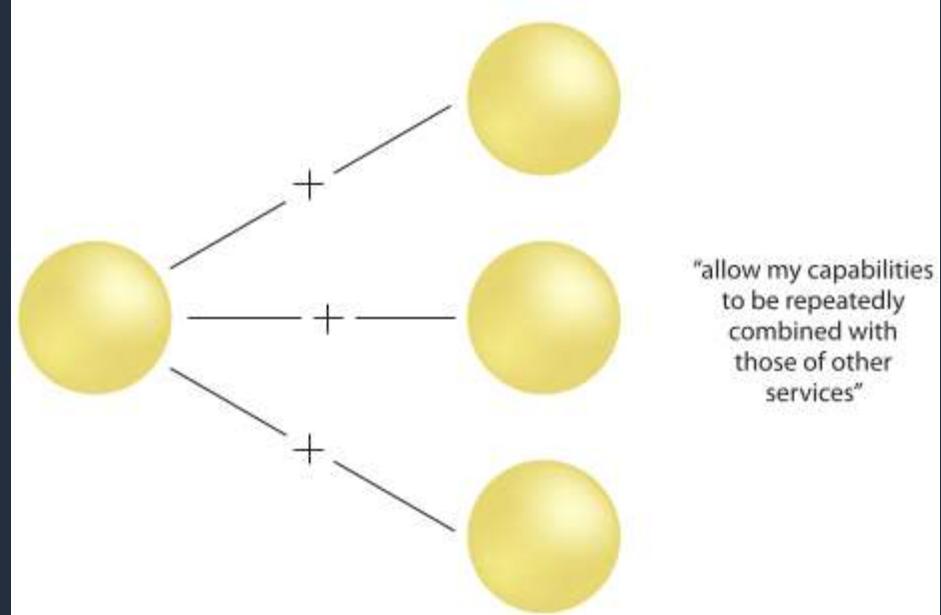
El principio de composición asegura la habilidad efectiva de un servicio para ser utilizado para componer otros servicios más complejos.

La composición es uno de los requisitos más críticos para lograr una arquitectura orientada a servicios.

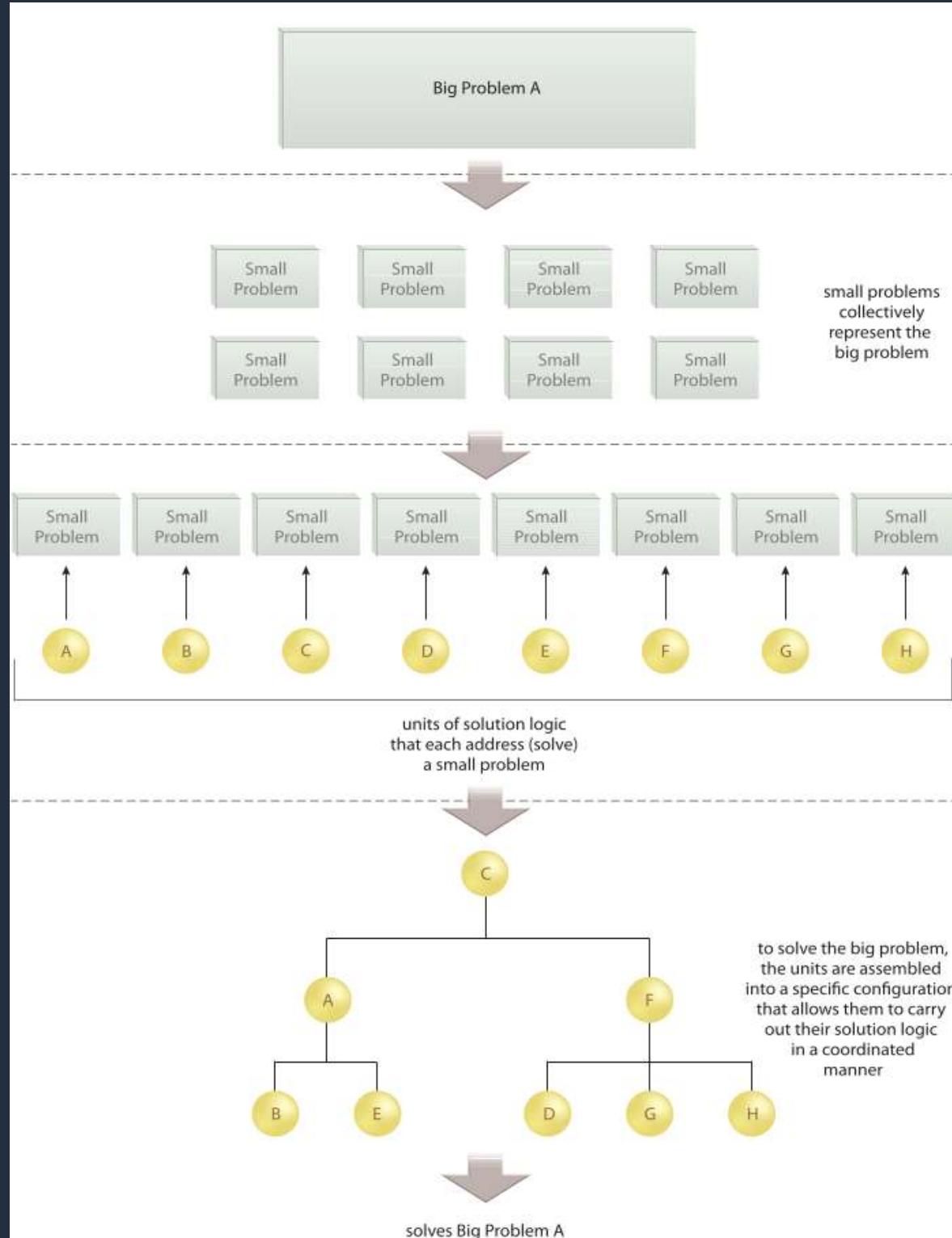
- El diseño de composición de servicios más complejos, basados en servicios de menor nivel (más atómicos) deben de ser visualizados con anticipación para evitar un doble esfuerzo.

El concepto de desarrollo de software a partir de componentes existentes de forma independiente, fomenta el concepto de Composición.

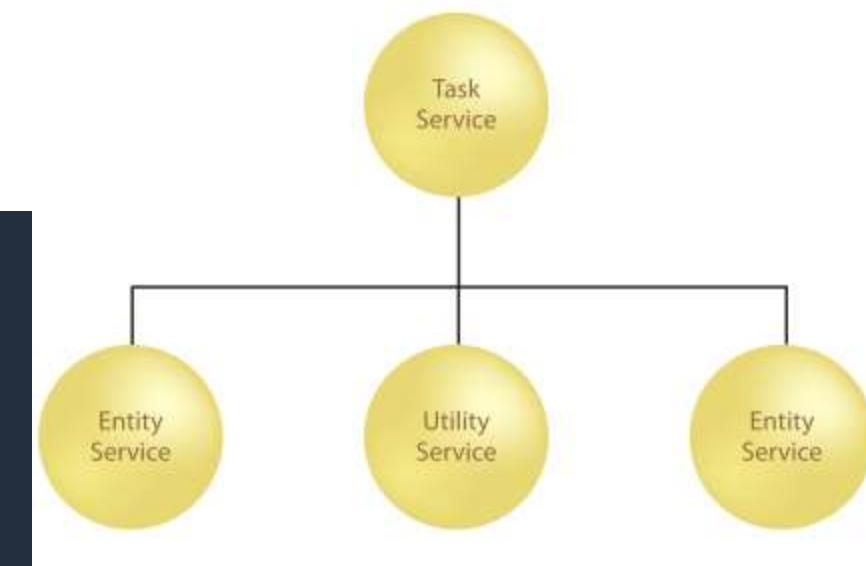
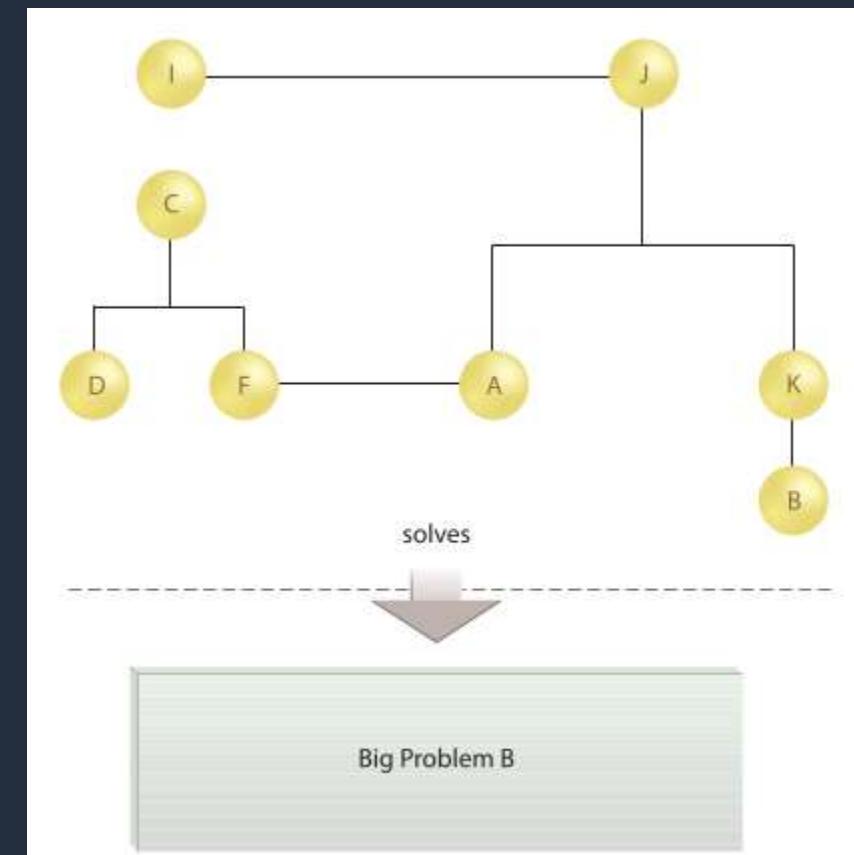
Dada la composición de servicios es que es posible automatizar un proceso de negocio debido a que, dicho proceso, se ejecuta mediante la combinación y composición de múltiples servicios.



# Principios de Diseño Composición



Las mismas unidades creadas originalmente para resolver un problema grande A se recompone para resolver colectivamente un problema diferente.



# Principios de Diseño para Microservicios

Principios de diseño de orientación para microservicios:

Contratos de servicios estandarizados.

Bajo acoplamiento.

Abstracción..

Autonomía.

Sin estado.



# Principios de Diseño para Microservicios

Los microservicios no proporcionan un mayor retorno de la inversión por medio de la alta reutilización, **lo que proveen es un potencial incremento de ingresos \$\$\$, al soportar el incremento de concurrencia por medio de estrategias de escalamiento en los periodos de alta demanda.**

- Entre más microservicios puedan soportar el incremento de la demanda, mayor es el incremento del ingreso.
- Generalmente la estrategia de escalamiento de los microservicios es por medio de contenedores.



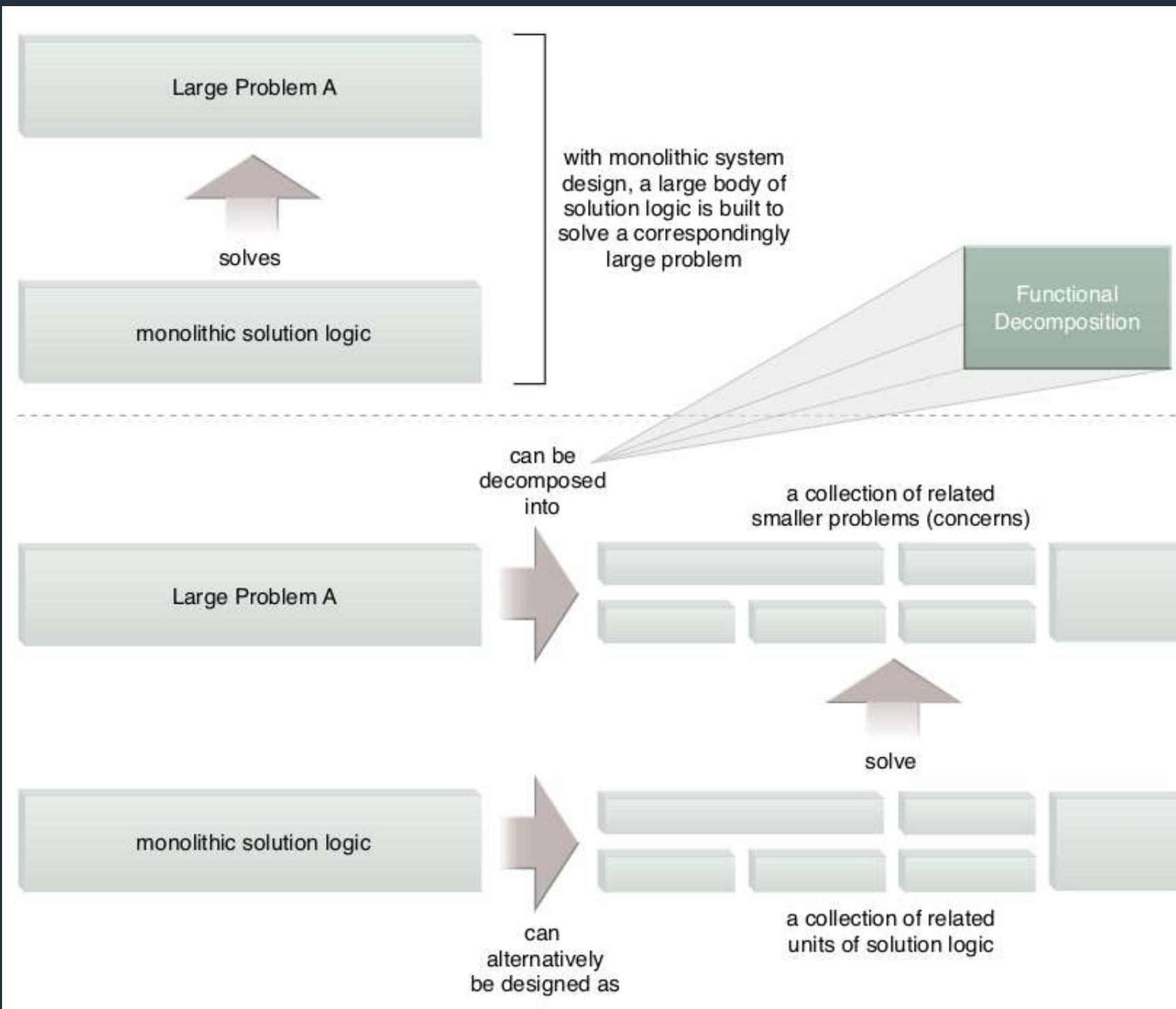
# Principios de Diseño para Microservicios

Los tipos de lógica permiten clasificar y representar los modelos del servicio

| Tipo de Servicio | Lógica de Negocio | Lógica de Utilidad Reusable | Lógica Agnóstica General | Lógica no-agnóstica Específica |
|------------------|-------------------|-----------------------------|--------------------------|--------------------------------|
| Task Service     | X                 |                             |                          | X                              |
| Microservice     | X                 |                             |                          | X                              |
| Entity Service   | X                 |                             | X                        |                                |
| Utility Service  |                   | X                           | X                        |                                |
|                  |                   |                             |                          |                                |
|                  |                   |                             |                          |                                |

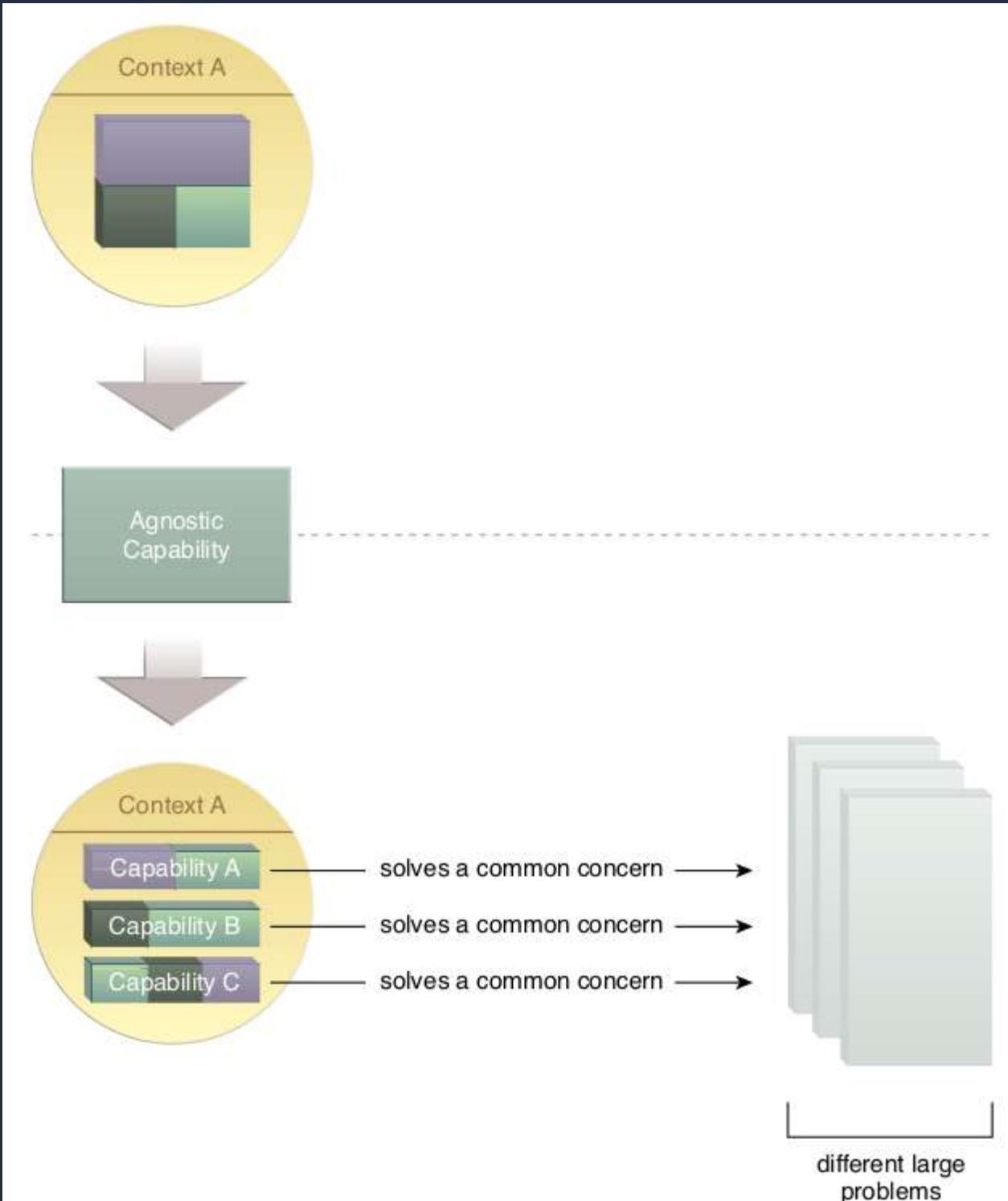


# Encapsulación de Servicios (Service encapsulation)



Al aplicar la separación de microservicios, el problema mayor se descompone en un conjunto de responsabilidades y la lógica de solución correspondiente se descompone en unidades más pequeñas

# Capa Agnóstica (Agnostic Capability) Orquestación



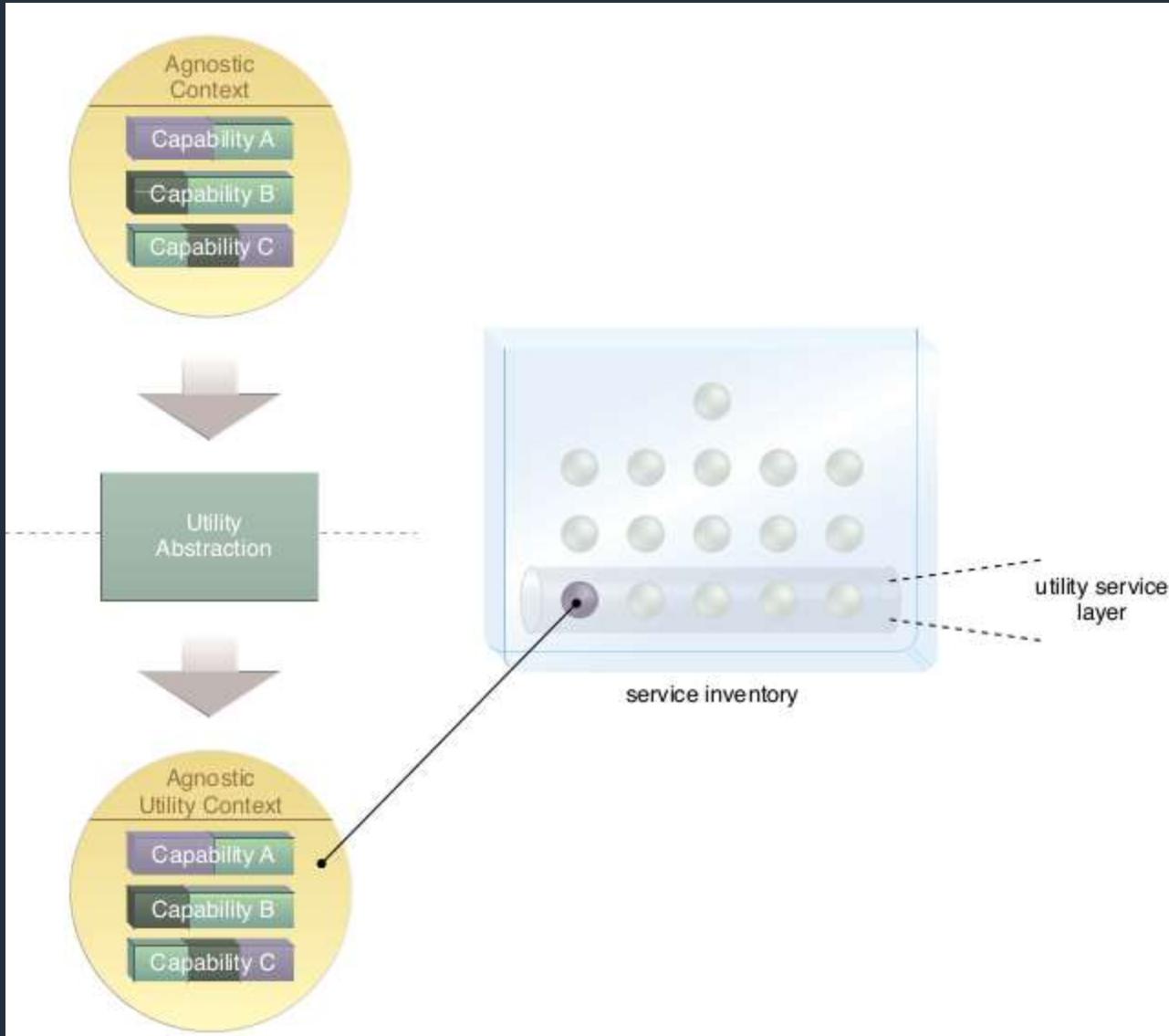
Se define un conjunto de capacidades de servicio agnóstico, cada una capaz de resolver una problema común.

## Procesos de negocio genericos

**Autorización de operaciones bancarias**  
**Autorizaciones de créditos bancarios**  
**Solicitud de créditos bancarios**



# Servicios de utilidad



## Abstracción de utilidad

El siguiente paso es separar la funcionalidad transversal común que no es específica a un proceso comercial ni a una entidad comercial. Esto establece una función agnóstica especializada. Contexto nacional limitado a la lógica que corresponde al modelo de servicio de utilidad.

## Repetiendo

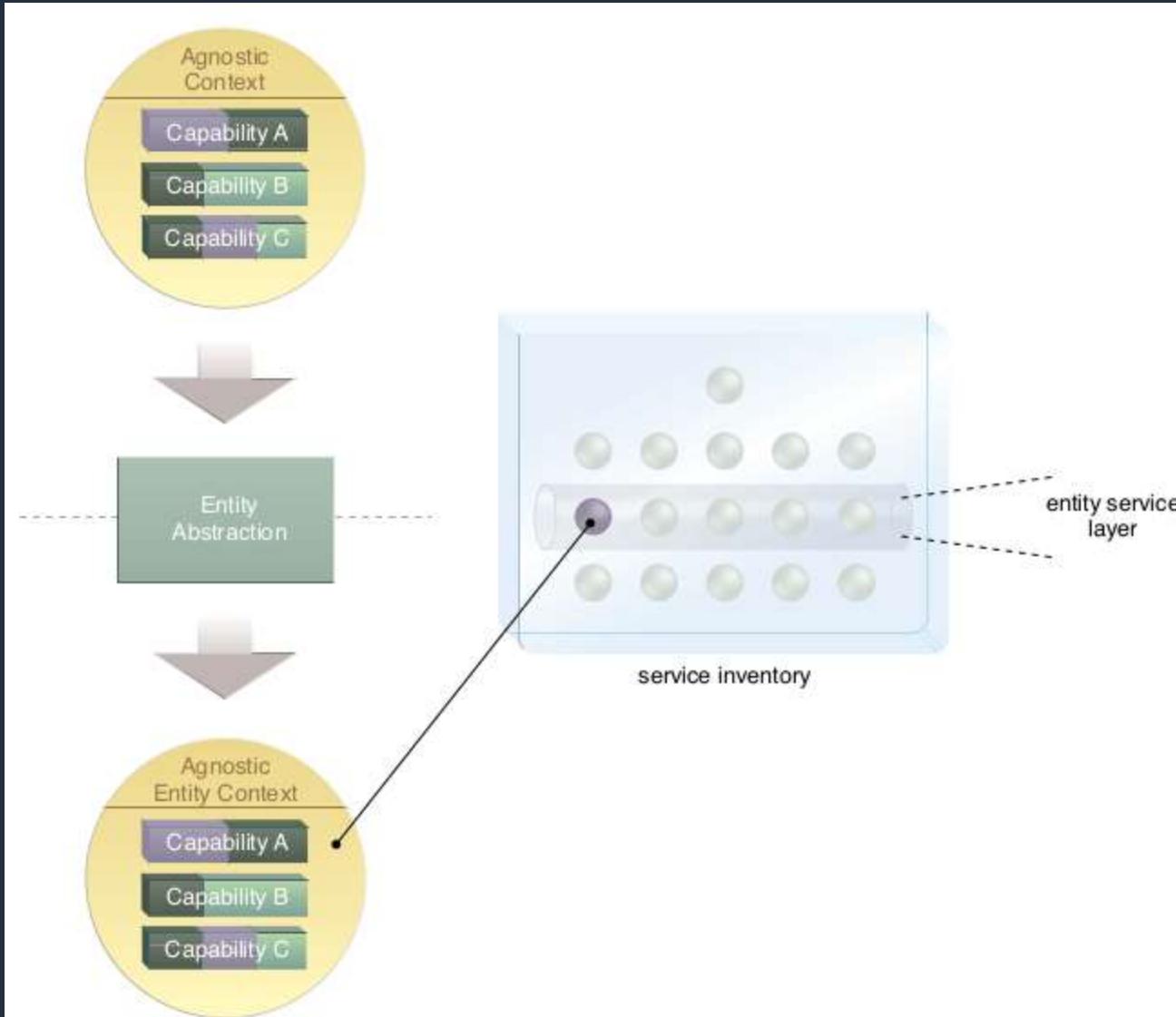
Este paso dentro de un inventario de servicios puede resultar en la creación de múltiples servicios públicos. candidatos y, en consecuencia, una capa de servicio de utilidad lógica.

**Mandar correo**  
**Generar pdf**  
**Notificaciones**

La lógica de servicio agnóstico centrada en la utilidad se organiza en una capa de servicio de la utilidad.



# Servicios de Entidad (API CRUD Específica)



Abstracción de entidad

Cada organización tiene **entidades comerciales** que representan artefactos clave relevantes sobre actividades operativas.

Este paso se centra en dar forma a lo funcional de un servicio para que se limite a la lógica que pertenece a uno o más negocios relacionados a las entidades.

Al igual que con la abstracción de la utilidad, repetir este paso tiende a establecer su propio capa de servicio lógica

**Crear cuenta bancaria**

**Consultar cuenta bancaria**

**Eliminar Cuenta bancaria**

**Obtener clientes**

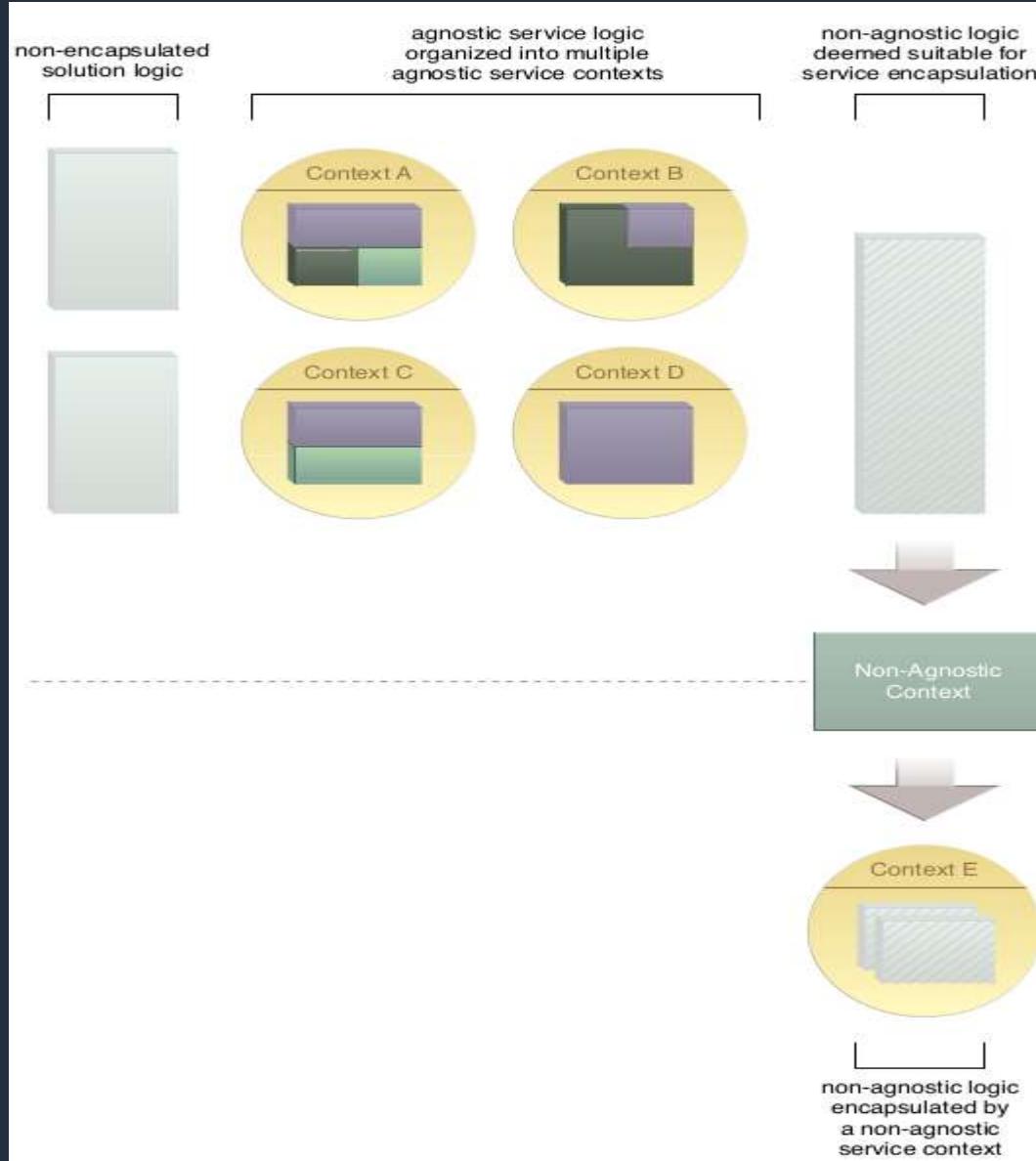
**Crear clientes**

**Obtener catalogos de tienda**

**Consulta inventario de productos**



# Servicios de Contexto No-Agnostico (Tareas Específicas)



El esfuerzo fundamental de identificación y definición del servicio detallado hasta ahora se ha centrado en la separación de la lógica de servicio multipropósito o agnóstico.

Lo que queda después de la lógica multipropósito ha sido separada es **lógica que es específica del proceso de negocio**.

Debido a que esta lógica se considera de naturaleza de un solo propósito, se clasifica como no-agnóstico

**Validación de RENAPO**

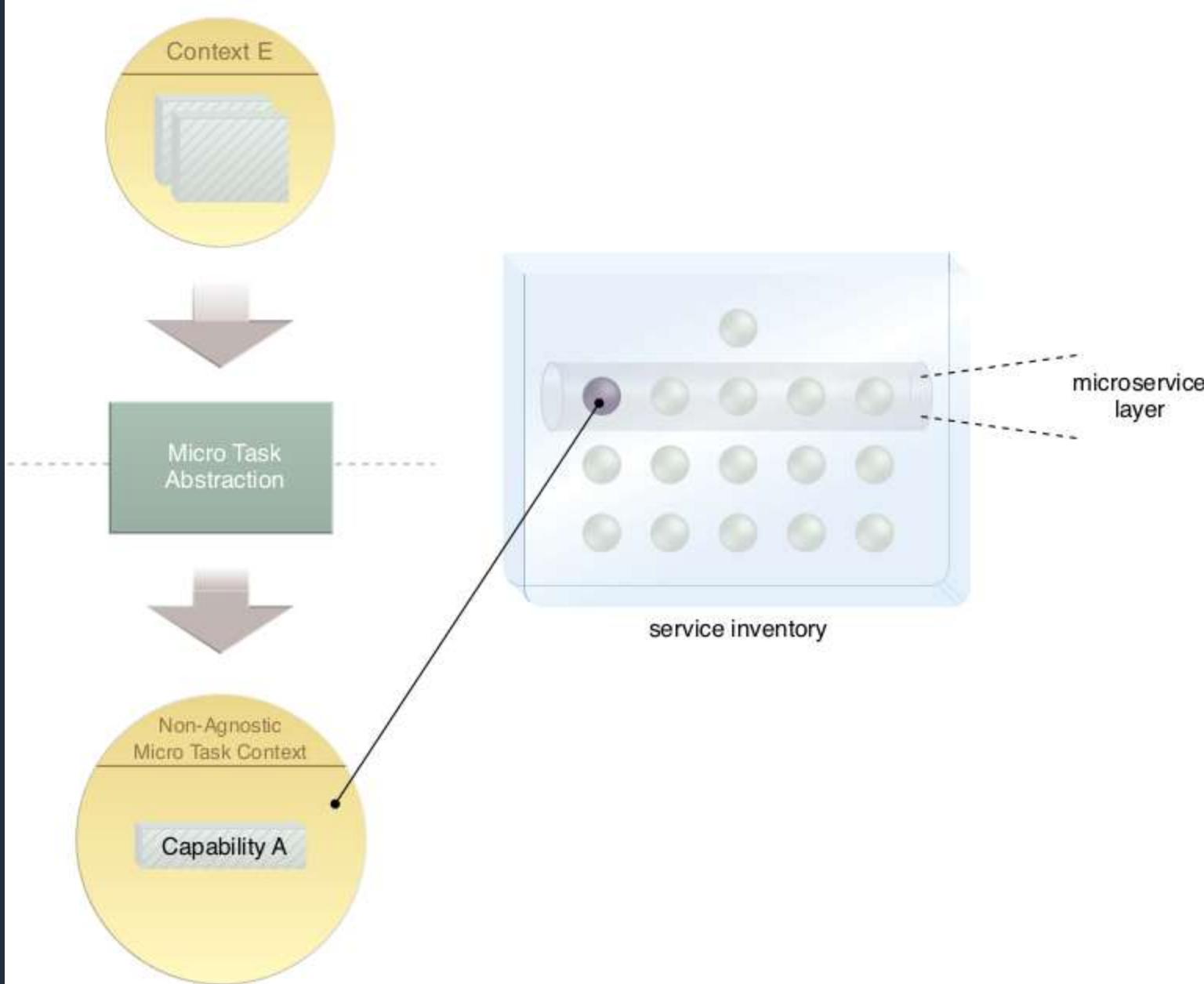
**Validación de INE**

**Validación de Buro de Crédito**

Al volver a el descomposición proceso, el restante lógica de servicio puede ahora ser categorizado como no-agnóstico



# Servicios Micro Task



Al revisar la lógica no-agnóstica disponible, puede ser evidente que los subconjuntos de este la lógica (o “**micro tareas**”) puede tener requisitos específicos de desempeño o confiabilidad.

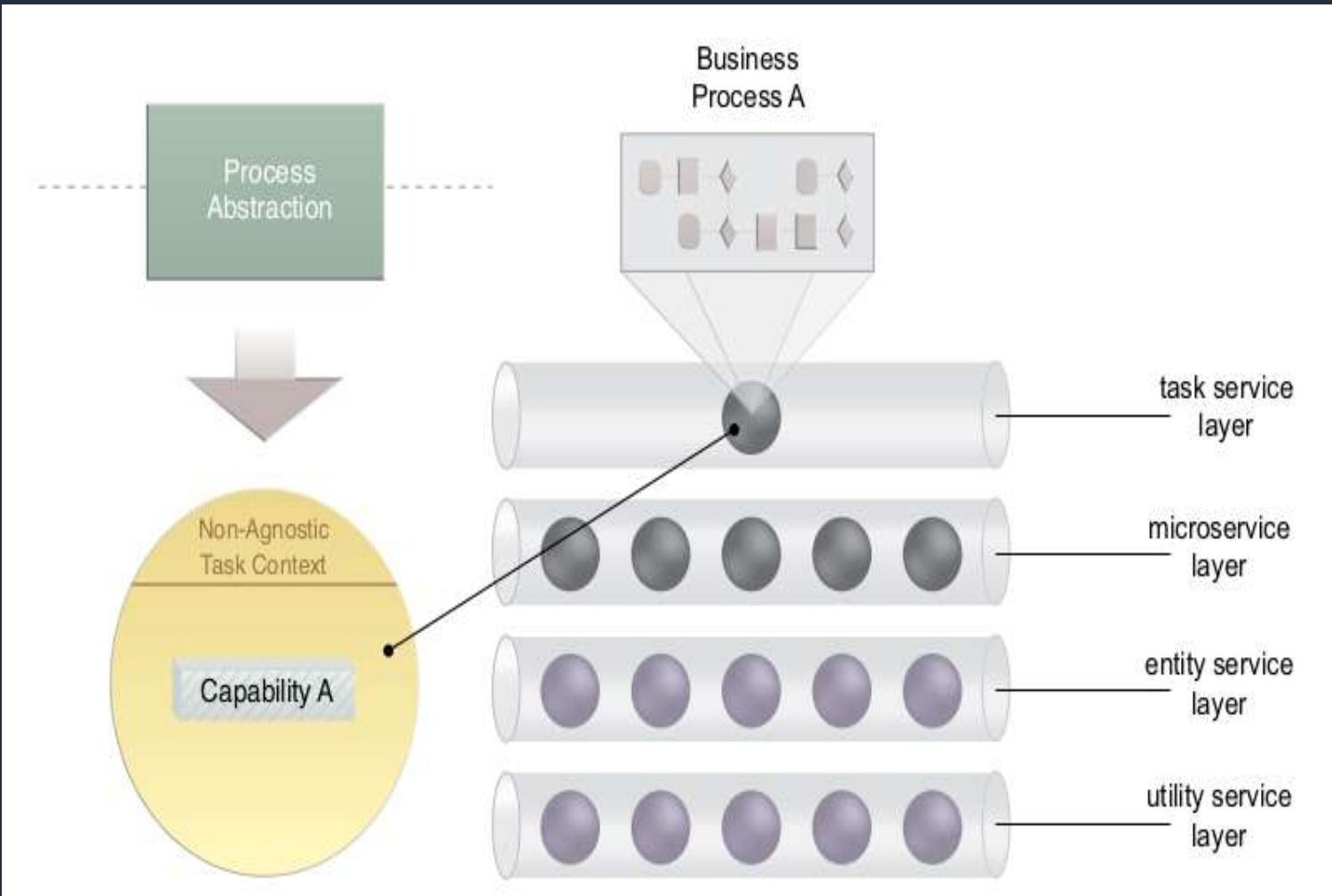
Este tipo de lógica de procesamiento puede **resumirse en una capa de servicio separada que puede beneficiarse de las distintas características de implementación de microservicios**

Seleccione la lógica **no agnóstica en candidatos de microservicio**.



# Servicios Task Services (Orquestación y Composición)

Abstracción de procesos y servicios de tareas



Resumir la lógica específica del proceso empresarial restante en su propia capa de servicio típicamente **resultan en la creación de un servicio de tareas, cuyo alcance es generalmente limitado vinculado al proceso de negocio principal**

Los tipos de lógica que generalmente son encapsulados por un servicio de tareas son la lógica de decisión, la lógica de composición y otras formas de lógica que es exclusiva del proceso de negocio que son responsables de automatizar.

Esta responsabilidad generalmente pone el servicio de tareas en control de la ejecución de un todo **“composición de servicio”, un rol conocido como controlador de composición.**

El servicio de tareas (Task Service) representa una parte de una capa de servicio principal y es responsable de encapsular el resto lógica específica del proceso de negocio principal.

**Producto - solicita tu prestamo BBVA**  
**Producto – Guardadito Banamex**  
**Producto – Inversión Joven HSBC**  
**Producto – Casa de Bolsa GBM**



A dark, slightly blurred photograph of a classroom or lecture hall. In the foreground, a person's hands are visible, holding a smartphone and pointing it towards the front of the room. Several other students are seated in rows, facing a front where a whiteboard or screen would typically be. The lighting is dim, creating a focused atmosphere on the central action.

# Sección I

# Estilo de Arquitectura

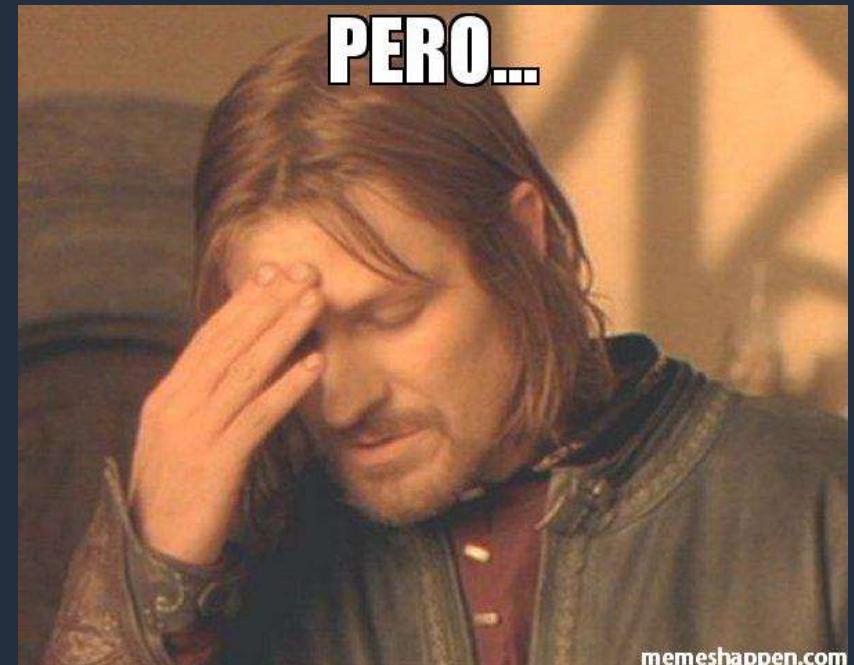
# Pero, Qué son los microservicios?

El estilo **arquitectónico de microservicios**, es un **enfoque** para **desarrollar una aplicación única**, como un conjunto de **pequeños servicios**, cada uno ejecutándose en su propio proceso y comunicándose con mecanismos ligeros, a menudo APIs de recursos HTTP



# Cuando implementar microservicios

- Aplicaciones grandes que requieran una alta velocidad de publicación de nuevos "releases" o funcionalidades.
- Aplicaciones complejas que requieren de gran escalabilidad y elasticidad.
- Aplicaciones que den soporte a negocios complejos donde se definan múltiples dominios o sub-dominios de negocio.
- Organizaciones que dispongan de pequeños equipos de trabajo.
- Organizaciones que dispongan de equipos de trabajo distribuidos.



# Cuando usarlos – Ventajas y Desventajas

- Desarrollo independiente.
- Equipos pequeños y centrados, metodologías ágiles.
- Despliegue independiente
- Escalabilidad independiente.
- Reusabilidad.
- Aislamiento de errores
- Stack tecnológico mixto(políglotas).
- Facilidad de mantenimiento.
- Facilidad de ejecución de pruebas unitarias.
- Tolerancia a fallos, alta disponibilidad y replicación.
- Distribución de carga, concurrencia y tiempos de respuesta.



# Cuando usarlos – Desventajas

**Complejidad:** Una aplicación de microservicios tiene más partes en movimiento que la aplicación monolítica equivalente. Cada servicio es más sencillo, pero el sistema como un todo es más complejo.

- **Dependencias para desarrollo y pruebas:** El rápido desarrollo de aplicaciones orientadas a microservicios suponen un mayor esfuerzo en el desarrollo (y pruebas) de microservicios dependientes de otros. La refactorización en las interfaces y en los límites del servicio puede resultar catastróficos. Debido a lo anterior, el versionado de los componentes es muy importante.

**Falta de Gobierno:** Debido a la falta de gobernabilidad, una arquitectura basada en microservicios puede acabar con tantos lenguajes y frameworks diferentes causando que la aplicación sea difícil de mantener. [Las APIs con Apigee](#)

- **Congestión y latencia de red:** Uno de los mayores problemas de las arquitecturas basadas en microservicios son las falacias de la computación distribuida. La comunicación entre muchos microservicios pequeños y detallados dar lugar a una mayor congestión y latencia en la Red.

**Integridad de los datos:** Cada microservicio es responsable de la conservación de sus propios datos, como consecuencia, la coherencia de los datos puede suponer un problema (eventual consistencia).

- **Administración:** Para tener éxito con los microservicios se necesita una cultura de DevOps consolidada debido a que el registro correlacionado entre microservicios puede resultar un desafío.



# Cuando usarlos – Desventajas

**Control de versiones:** Las actualizaciones de un servicio no deben interrumpir servicios que dependen del mismo. Es posible que varios microservicios se actualicen en cualquier momento, por lo tanto, sin un cuidadoso diseño entre sus interfaces y sin un adecuado versionamiento, podrían surgir problemas con la compatibilidad con versiones anteriores y/o posteriores del software.

- **Conjunto de habilidades:** Los microservicios son sistemas muy distribuidos. Es necesario evaluar si el equipo de desarrollo tiene los conocimientos y la experiencia para desenvolverse correctamente en el desarrollo de sistemas basados en arquitectura de microservicios.

**Mayor complejidad para los operadores:** Para los operadores, o equipos de monitoreo, se origina una explosión de mayores procesos a administrar y monitorear debido a que pueden desplegarse decenas, cientos o miles de microservicios en ejecución donde cada uno de ellos, se ejecuta en un proceso independiente.

- **Conjunto de habilidades:** Los microservicios son sistemas muy distribuidos. Es necesario evaluar si el equipo de desarrollo tiene los conocimientos y la experiencia para desenvolverse correctamente en el desarrollo de sistemas basados en arquitectura de microservicios.

**Mayor complejidad en la delimitación de los microservicios:** La complejidad del negocio puede aparentar que, sobre el papel, los microservicios estén bien delimitados, sin embargo, mientras se va desarrollando e implementando posibles caminos alternos, se descubre que los microservicios no son tan independientes entre. Ejemplo, compartición de los mismos datos.

- **Obviar la complejidad del estado entre microservicios:** El manejo de microservicios sin estado es efectivo, es decir que los servicios son “stateless”. El manejo de estado dificulta la escalabilidad. Los microservicios deberían de recibir como entrada todos los datos requeridos para operar.



# Cuando usarlos – Desventajas

Transaccionabilidad: Amplia dificultad para implementar operaciones transaccionales entre llamadas a microservicios.

Supone un gran esfuerzo y ello conyeva a aumentar los tiempos de respuesta de los microservicios, habilitando “cuellos de botella”.

**No se recomienda implementar transaccionabilidad entre microservicios, para ello se recomienda implementar servicios idempotentes o habilitar “eventual consistencia”.**

- **Monolítos disfrazados, microservicios o nanoservicios: Delimitar los microservicios es crucial. ¿Qué tan micro es un microservicio?**

Dificultad para el despliegue: Dado el alto número de microservicios que puede suponer un sistema en su totalidad, la administración para el despliegue de los microservicios supone un reto. Requiere automatización y una cultura DevOps madura.

Falacias de la computación distribuida, entre otras ...



# Cuando usarlos – Falacias de la computación distribuida

La red es confiable.

La latencia es cero.

El ancho de banda es infinito.

La red es segura.

La topología no cambia.

Hay uno y sólo un administrador.

El costo de transporte es cero.

La red es homogénea

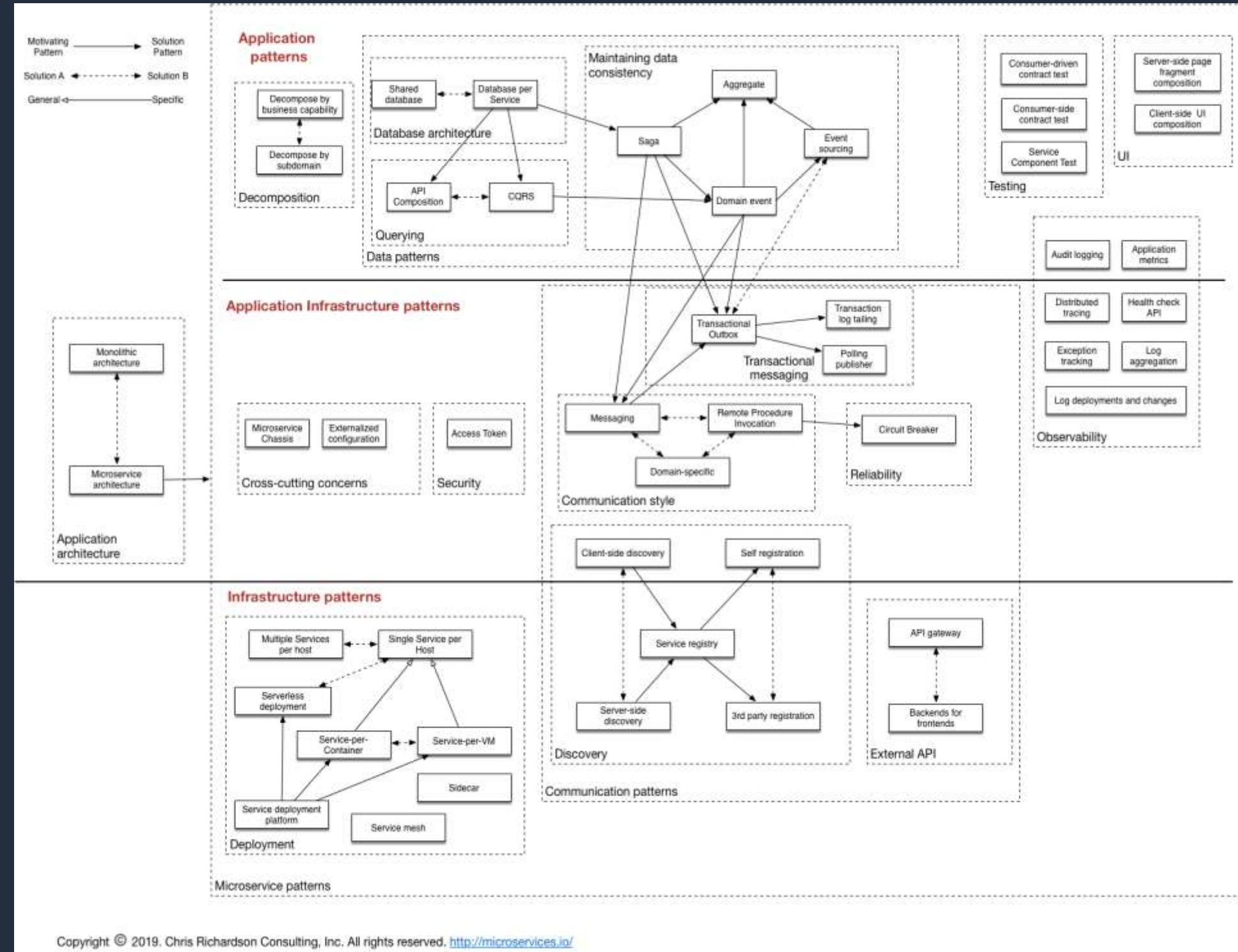


A dark, slightly blurred photograph of a classroom or lecture hall. In the foreground, a person's hands are visible, holding a smartphone and pointing it towards the front of the room. Several other people are seated in rows, facing a front where a presentation is being shown on a screen. The overall atmosphere is focused and educational.

# Sección I

# Patrones de Diseño

# Patrones a usar al crear microservicios



# Patrones para el estilo de arquitectura de microservicios

**La arquitectura del microservicio no es una bala de plata.** Tiene varios inconvenientes.

El lenguaje de patrones de arquitectura de microservicio es una colección de patrones para aplicar la arquitectura de microservicio.

Tiene dos objetivos:

- El lenguaje de patrones le permite decidir si los microservicios son adecuados para su aplicación.
- El lenguaje de patrones le permite utilizar la arquitectura de microservicio correctamente.

**Un buen punto de partida es el patrón de arquitectura monolítica, que es el estilo arquitectónico tradicional,** que sigue siendo una buena opción para muchas aplicaciones.

Sin embargo, tiene numerosas limitaciones y problemas, por lo que una mejor opción para **aplicaciones grandes / complejas** es el patrón de arquitectura **Microservice**.



# Patrones Varios

## Service Discovery

- Requerido para localizar los diferentes servicios disponibles.
- Implica usar un Servicio de Registro dinámico, eficiente y distribuido.

## Communication Patterns

- API gateway

## Data management

- Database per service
- Shared database
- Event-driven architecture
- Event sourcing
- CQRS
- Database triggers
- Application events

## Patrones de Despliegue

- Multiple instances per host
- Single Service instance per host
- Service instance per VM
- Service instance per container
- Serverless



# ANTIPATRONES

- Empezar un sistema con microservicios totalmente, Se recomienda empezar monolitos y extraer a microservicios cuando sea necesario.
- No Automatización de tareas de operaciones (CI/CD)
- Servicios diseñados en capas
- Datos (Cada microservicio debería tener las capas necesarias)
- No versionar los servicios
- No usar procesamiento basado en encolamiento (Resiliencia)



A blurred background image showing several people in an office environment. Some are sitting at desks with laptops, while others are standing or talking. The scene conveys a sense of teamwork and technology.

# Sección I

## 12 Factores SaaS (Software como servicio)

# The Twelve Factor App

“The twelve-factor app” es una metodología para construir aplicaciones SaaS

## I. Código base (Codebase)

Un código base sobre el que hacer el control de versiones y multiples despliegues

## II. Dependencias

Declarar y aislar explícitamente las dependencias

## III. Configuraciones

Guardar la configuración en el entorno

## IV. Backing services

Tratar a los “Backing services” como recursos conectables

## V. Construir, desplegar, ejecutar

Separar completamente la etapa de construcción de la etapa de ejecución

## VI. Procesos

Ejecutar la aplicación como uno o más procesos sin estado

## VII. Asignación de puertos

Publicar servicios mediante asignación de puertos

## VIII. Conurrencia

Escalar mediante el modelo de procesos

## IX. Desechabilidad

Hacer el sistema más robusto intentando conseguir inicios rápidos y finalizaciones seguras

## X. Paridad en desarrollo y producción

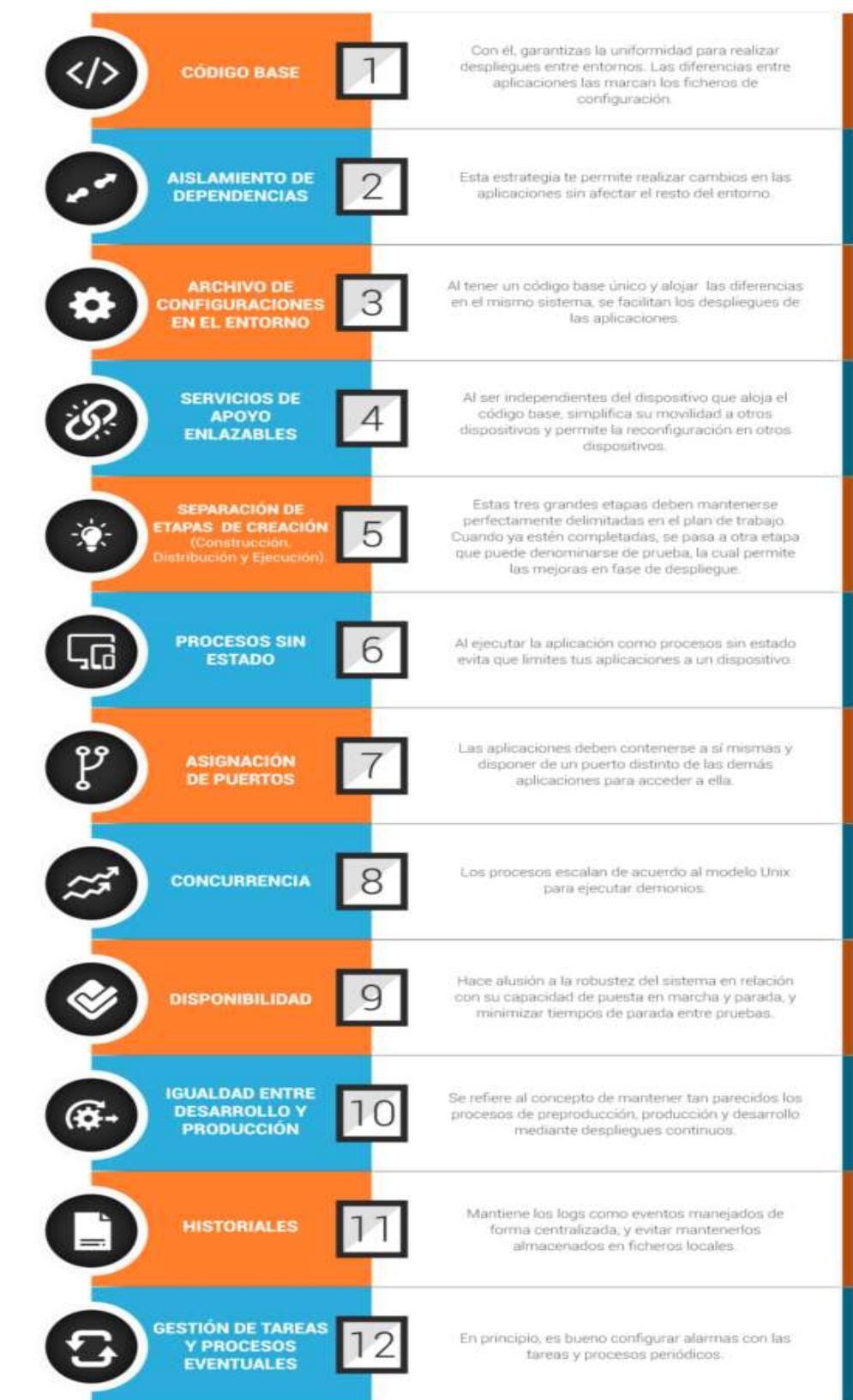
Mantener desarrollo, preproducción y producción tan parecidos como sea posible

## XI. Historiales

Tratar los historiales como una transmisión de eventos

## XII. Administración de procesos

Ejecutar las tareas de gestión/administración como procesos que solo se ejecutan una vez



A dark, slightly blurred background image showing a group of people in what appears to be a workshop or laboratory environment. In the foreground, a person's hands are visible on a laptop keyboard. The overall atmosphere is professional and focused.

# Sección I

# Domain Driven Desing

# Patrones DDD

**¿Qué es el dominio?** Área temática o campo a la que un usuario aplica un software.

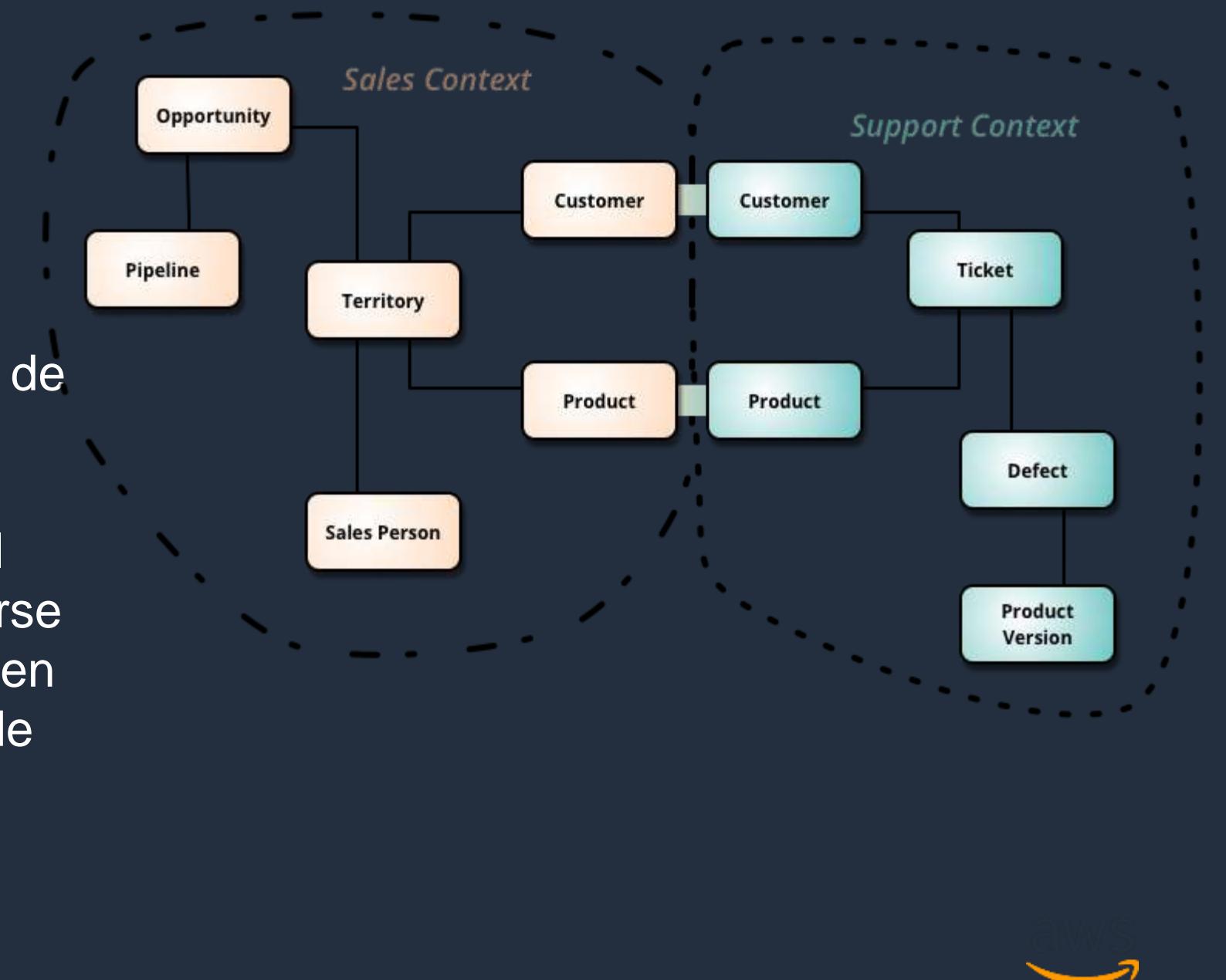
**¿Qué es el modelo de dominio?** Representa la terminología y los conceptos clave del dominio del problema. Identifica las relaciones entre las entidades incluidas dentro del ámbito del dominio del problema, identifica sus atributos y proporciona una visión estructural del dominio.



# Patrones DDD, Event Driven Architecture

Diseñe servicios descomponiendo por contexto “Bounded-context” y por requerimientos de escalabilidad.

- Consentir que las aplicaciones están compuestas por múltiples requerimientos funcionales, que tienen diferentes requerimientos de escalado.
- Posiblemente un mismo requerimiento funcional definido en un microservicio, pueda descomponerse en dos microservicios si algún servicio contenido en el microservicio, tiene diferentes requerimientos de escalabilidad.



A blurred background image showing several people in an office environment. Some are sitting at desks with laptops, while others are standing or talking. The scene conveys a sense of teamwork and technology.

# Sección I

## Protocolos de comunicación para Microservicios

# Comunicación en los Microservicios

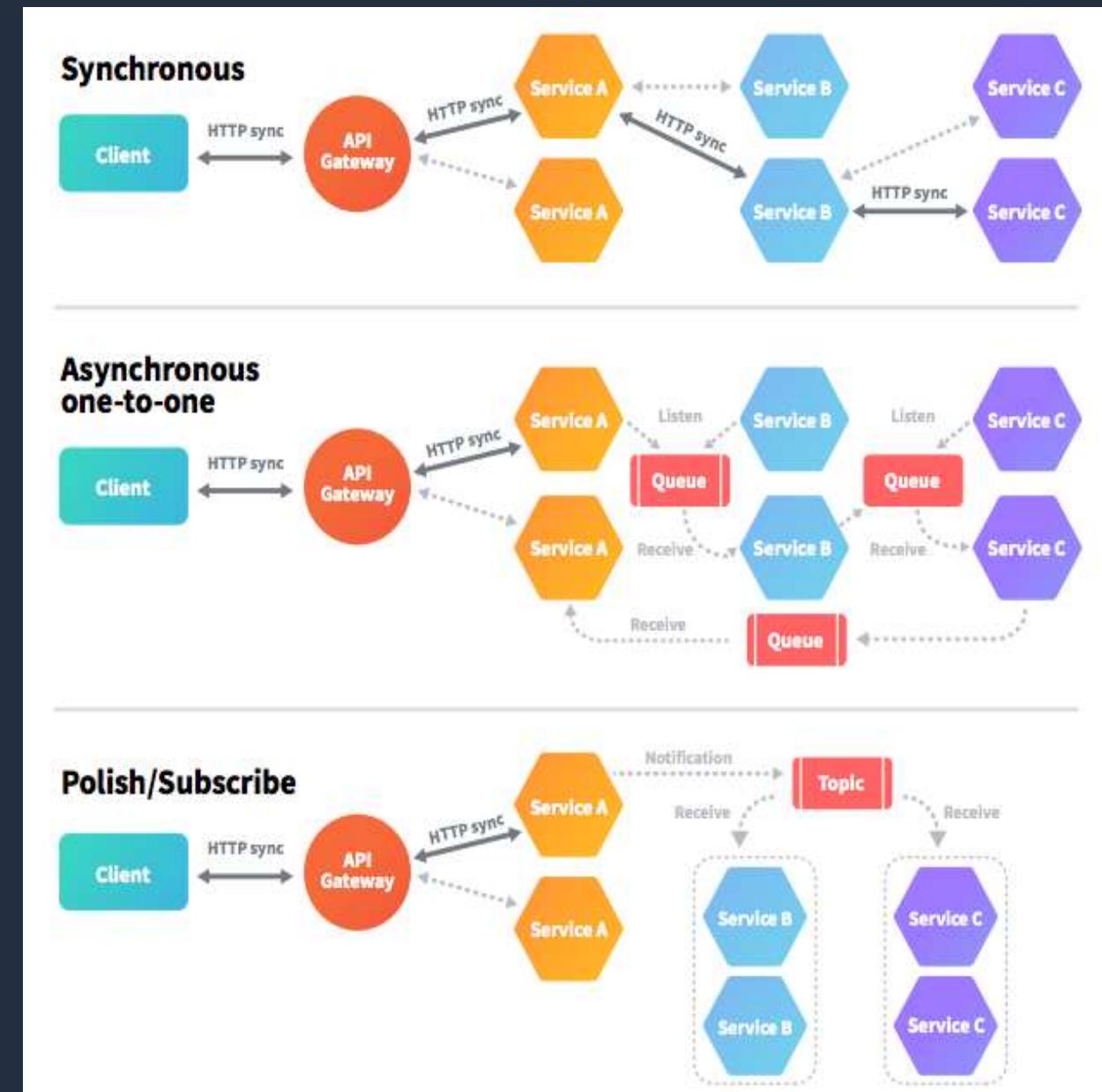
En una aplicación monolítica, las comunicaciones entre componentes se ejecutan en un único proceso, no necesariamente en un mismo hilo de ejecución, donde los componentes se invocan entre sí mediante llamadas de funciones (o llamadas a métodos) a nivel de lenguaje de forma acoplada o desacoplada.

**Lo más complicado al refactorizar una aplicación monolítica a microservicios es cambiar el mecanismo de comunicación entre los componentes mediante comunicación entre procesos (Inter Process Communication, IPC).**

No existe una única solución, para evitar las comunicaciones, **una posible solución puede ser delimitar los contextos de negocio lo más aislados posibles, de tal grado que no exista comunicación entre microservicios.**

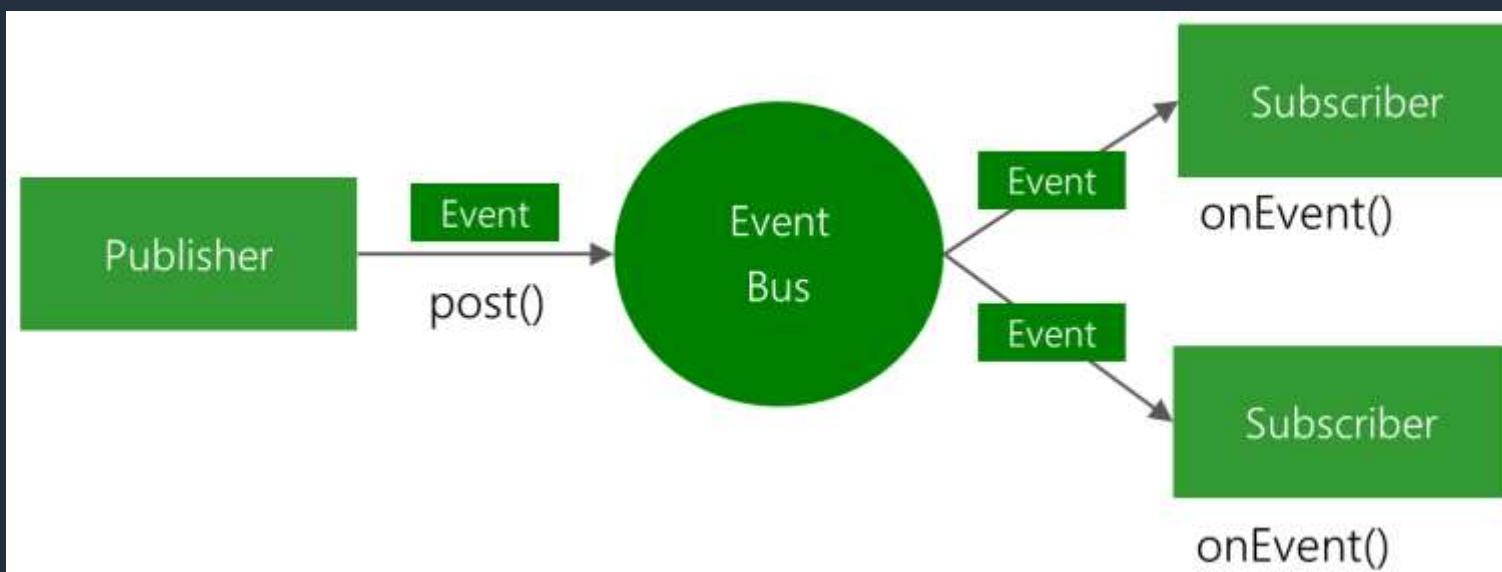
Dado que cada instancia de un microservicio es un proceso independiente, debe implementar un protocolo de comunicación entre procesos como son HTTP, AMQP, gRPC o TCP/IP, dependiendo de la función de cada microservicio.

**Una sana implementación de microservicios promueve que la intercomunicación entre microservicios sea mediante “smart endpoints and dumb pipes” (Puntos de conexión inteligentes y tuberías tontas) fomentando así un diseño desacoplado entre microservicios.**



# Comunicación en los Microservicios

Habitualmente se sugieren el protocolo HTTP para situaciones “request-response” mediante APIs REST y mensajería asíncrona ligera para comunicación de tipo “fire-and-forget”.

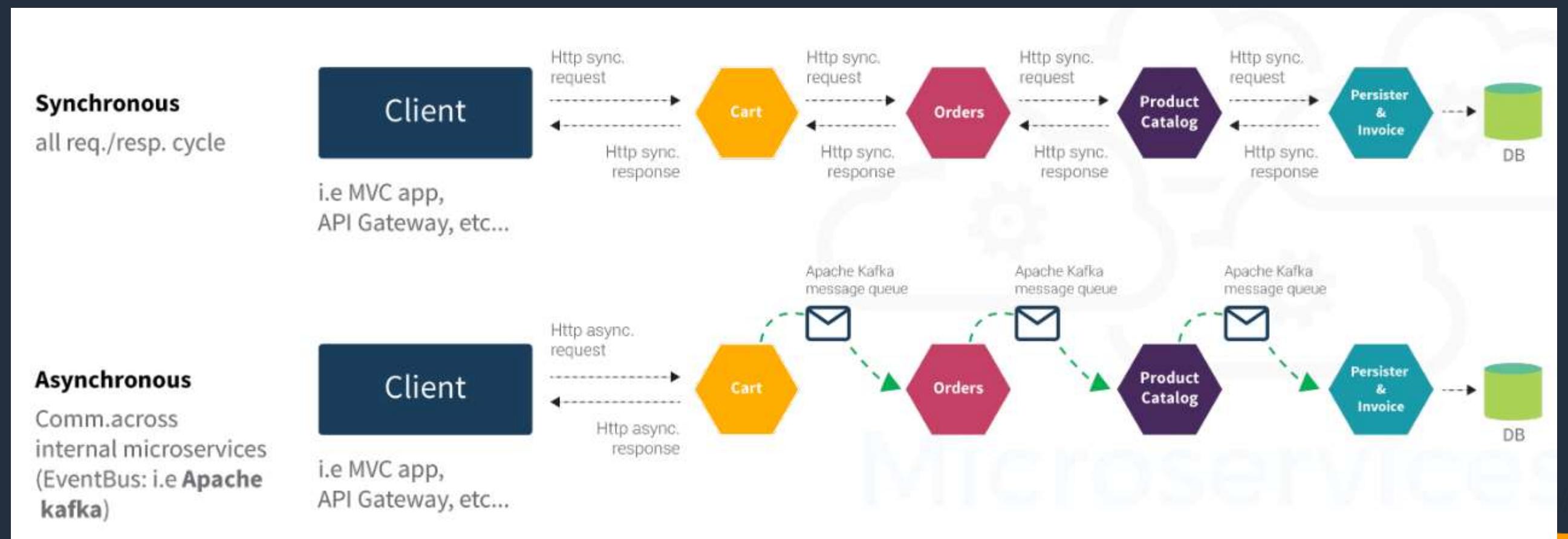


# Sección I

## Protocolos sincronos (HTTP / HTTPS)

# Comunicación sincrona

Lo más común en la comunicación desde la aplicación cliente, consumidora de un microservicio es mediante protocolo síncrono como HTTP/HTTPS, mientras que, internamente, los microservicios se comunican entre sí de forma asíncrona, lo cual habilita su independencia, obligando su autonomía, facilidad de escalabilidad, reusabilidad, tolerancia a fallos y disponibilidad.



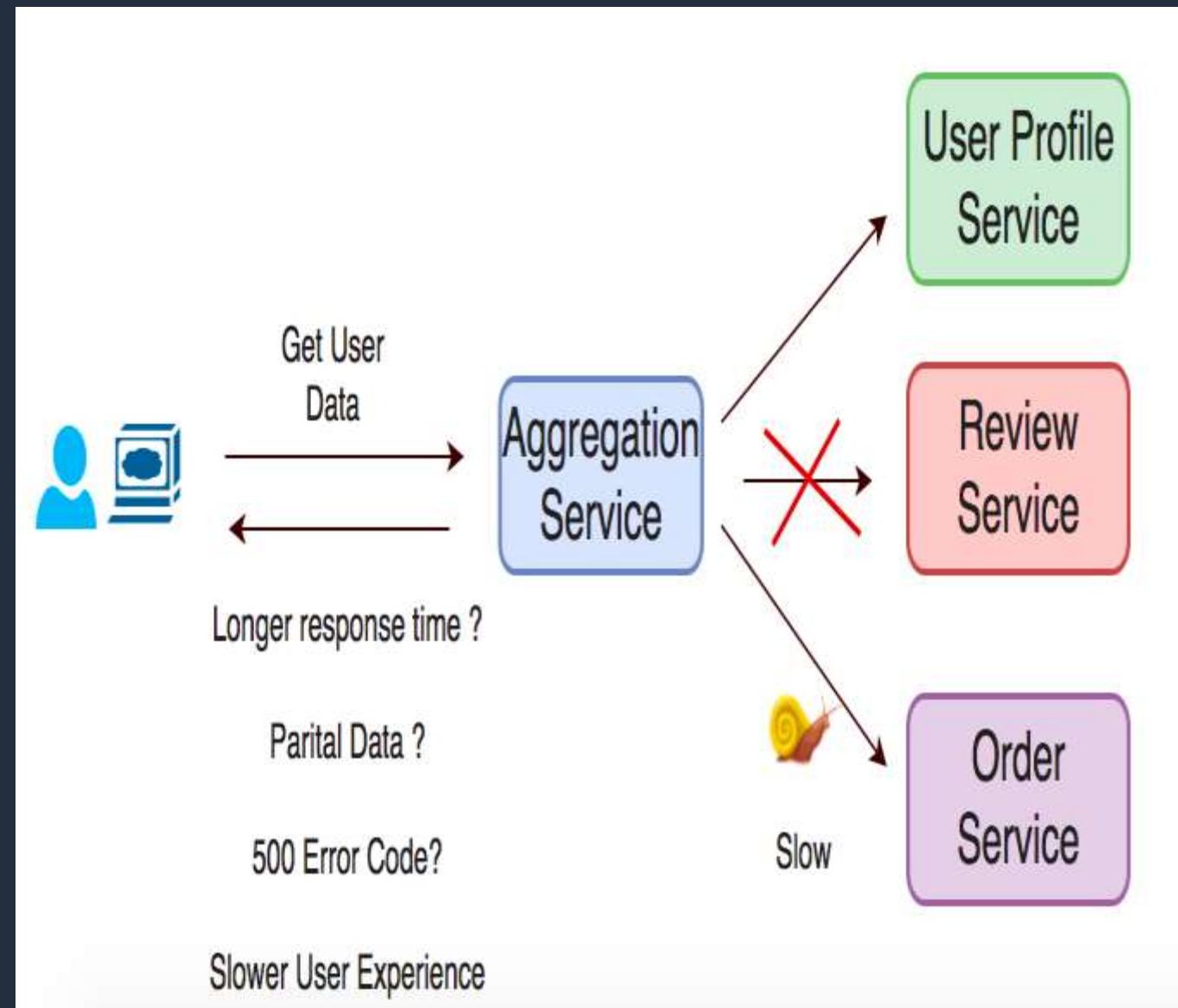
# Comunicación sincrona

Desventajas de comunicación síncrona:

- Baja complejidad en el diseño.
- Facilidad para el manejo de errores.
- Recepción de respuestas en tiempo real (on-the-go).

**El servicio debe estar disponible todo el tiempo, si el servicio no esta disponible, el hilo “caller” puede bloquearse por un tiempo, hasta que ocurra un error por “time-out”, causando problemas de performance.**

- \* Posibilidad de propagación no controlada de errores de comunicación.
- \* Respuestas lentas debido a que los servicios deben esperar a que termine de recibir la respuesta de los demás servicios involucrados.
- \* Necesidad de un protocolo orientado a conexión (TCP).



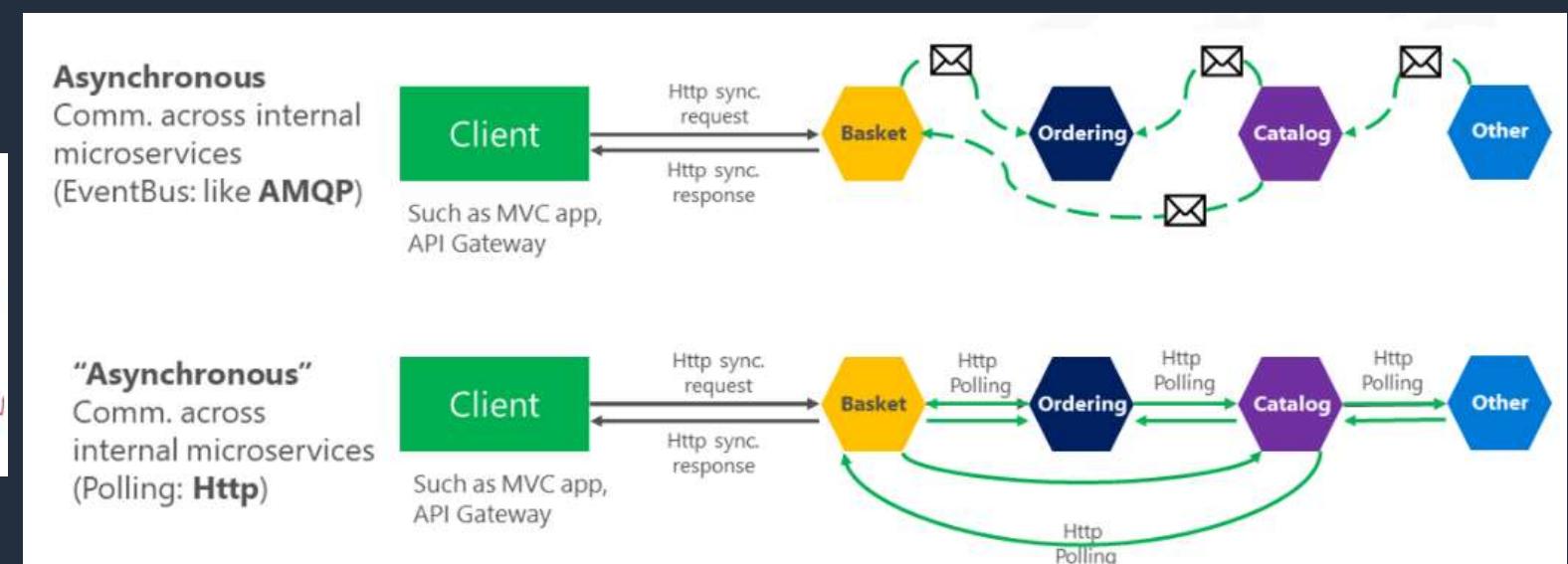
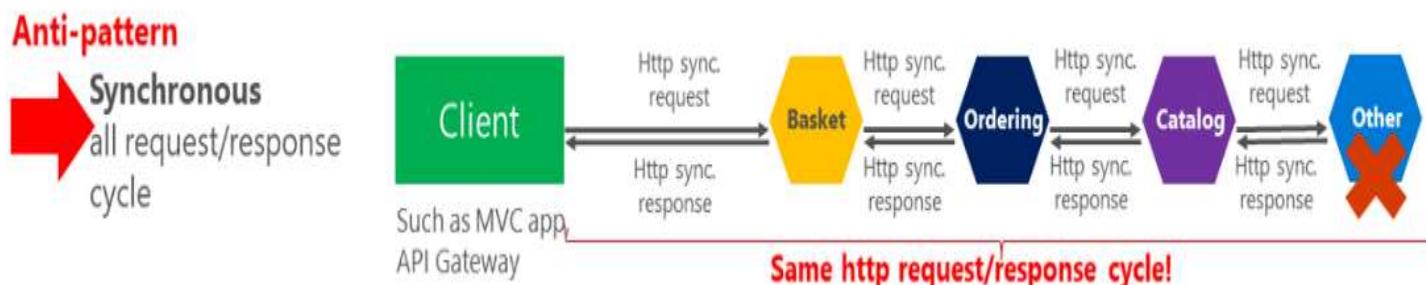
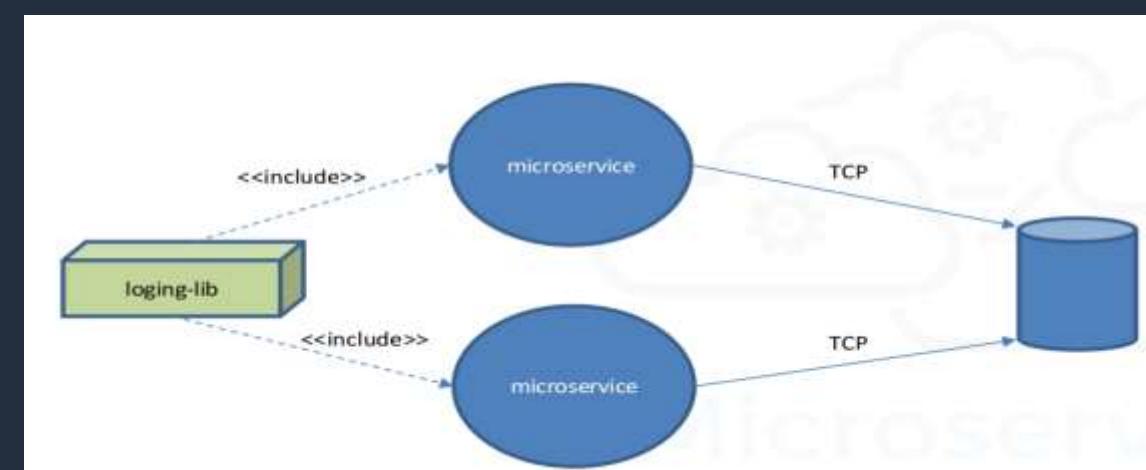
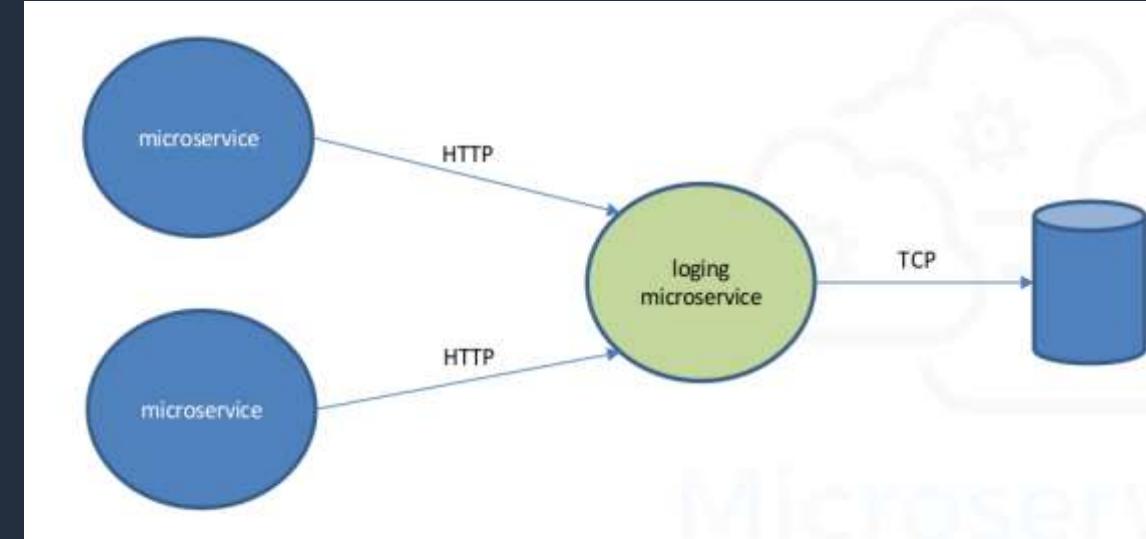
# Comunicación sincrona

Autonomía de microservicios

Idealmente es preferible **evitar las comunicaciones entre si, utilizar código reusable (librerias) en lugar de depender de llamadas a microservicios de uso general.**

Cuando sea necesaria la comunicación entre microservicios, dicha comunicación, **como regla fundamental debe ser asíncrona. Siendo posible jamás depender de una comunicación síncrona.**

**La comunicación síncrona entre microservicios promueve una alta dependencia entre microservicios**, lo cual evita su independencia y autonomía, indistintamente de que el despliegue de los microservicios sea autónomo e independiente.



# Sección I

## Protocolos asincronos (JMS / (TCP/IP), AMQP)

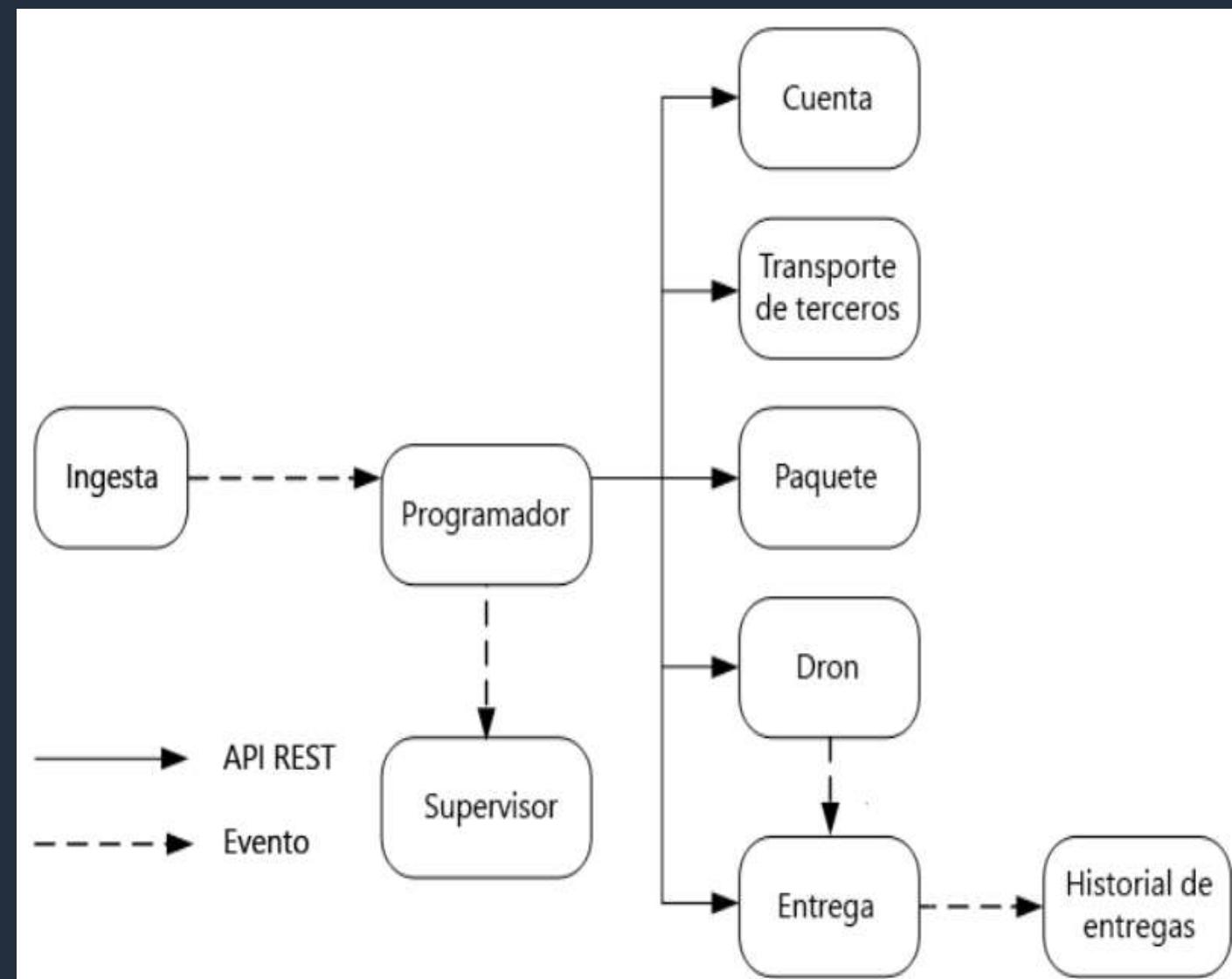
# Comunicación Asincrona

En este patrón, un servicio envía el mensaje sin esperar por la respuesta, y uno o más servicios procesan el mensaje de forma asincrónica.

**Acoplamiento reducido.** El remitente del mensaje no tiene que conocer al consumidor.

**Varios suscriptores.** Al utilizar un modelo de pub/sub, se pueden suscribir varios consumidores para recibir eventos.

**Aislamiento de errores.** Si se produce un error en el consumidor, el remitente puede seguir enviando mensajes. Los mensajes se recogerán cuando el consumidor se recupere. Esta capacidad es especialmente útil en una arquitectura de microservicios, dado que cada servicio tiene su propio ciclo de vida. Un servicio puede dejar de estar disponible o se puede sustituir por una versión más reciente en un momento dado.



# Comunicación Asincrona

## Ventajas de comunicación asíncrona:

- Sin necesidad de protocolos orientados a conexión ya que la información viaja mediante “brokers” de mensajes.
- Sin necesidad de esperar una respuesta del lado del consumidor, no hay problemas de performance.
- El productor (quien envía el mensaje) no necesita saber quien o cuantos son los consumidores (quien recibe el mensaje) del mensaje. Existe un bajo acoplamiento entre servicios.
- No es necesaria una alta disponibilidad del servicio consumidor.
- La comunicación asíncrona mejora la tolerancia a fallos, aísla los fallos.

## Desventajas de comunicación asíncrona:

- Mayor complejidad en el diseño de sistemas distribuidos mediante comunicación asíncrona.
- Dificultad de manejar errores en componentes con comunicación asíncrona.
- Alto acoplamiento con el “broker” de mensajes.
- Alto costo de latencia si la bandeja (cola) de mensajes alcanza su límite.
- Alto costo para el monitoreo y “debug” de la infraestructura de mensajes.

**Cuando un microservicio requiera datos para operar, cuyo propietario de los datos es otro microservicio, no dependa de la solicitud de dichos datos a ese otro microservicio.**

**Replique los datos en ambos microservicios  
(Database per Service Pattern).**

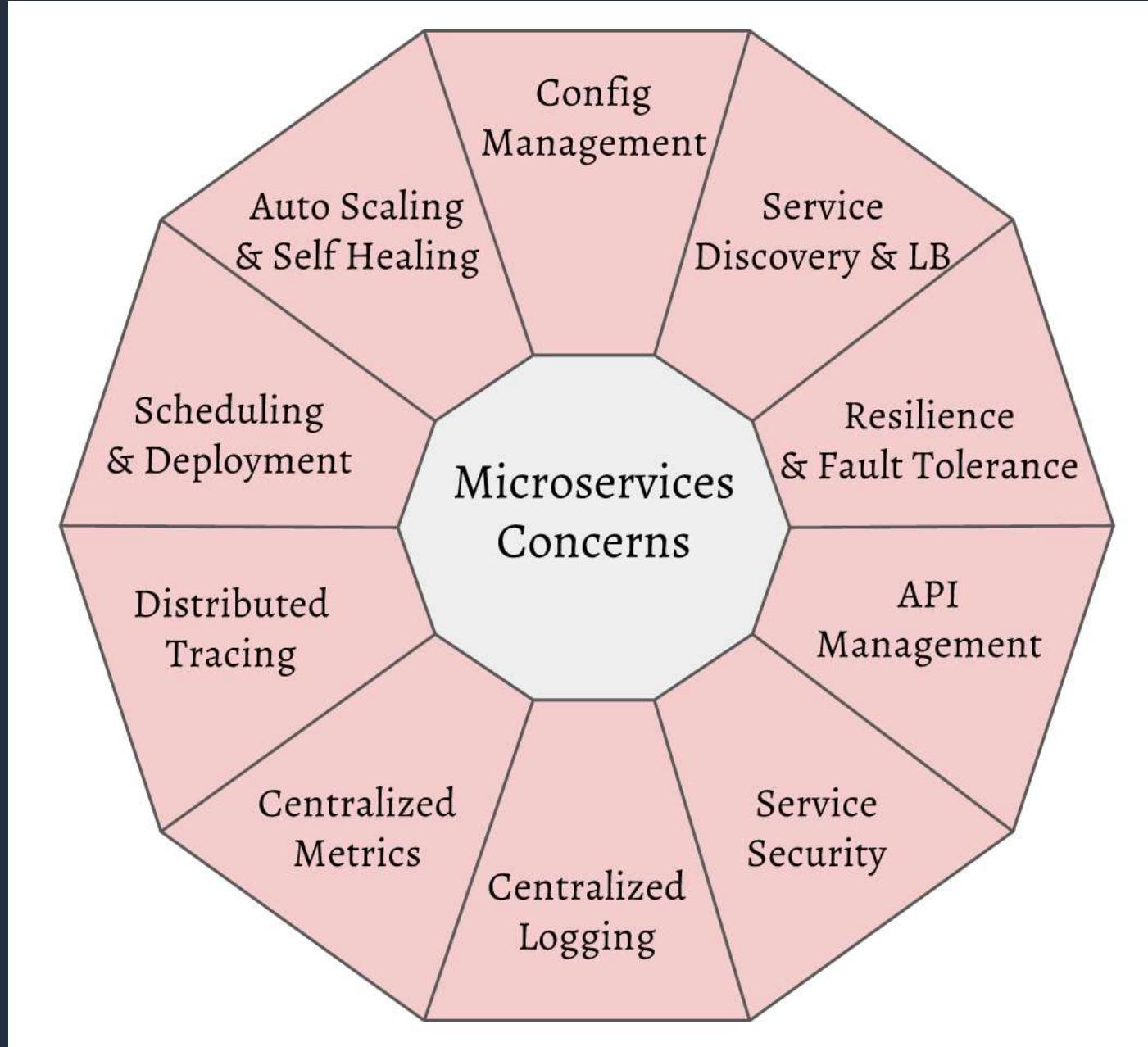


A blurred background image showing several people in an office environment. Some are sitting at desks with laptops, while others are standing or walking. The scene is lit with warm, ambient light.

# Sección I

## Consideraciones alrededor de los microservicios

# No solo es hacer microservicios...



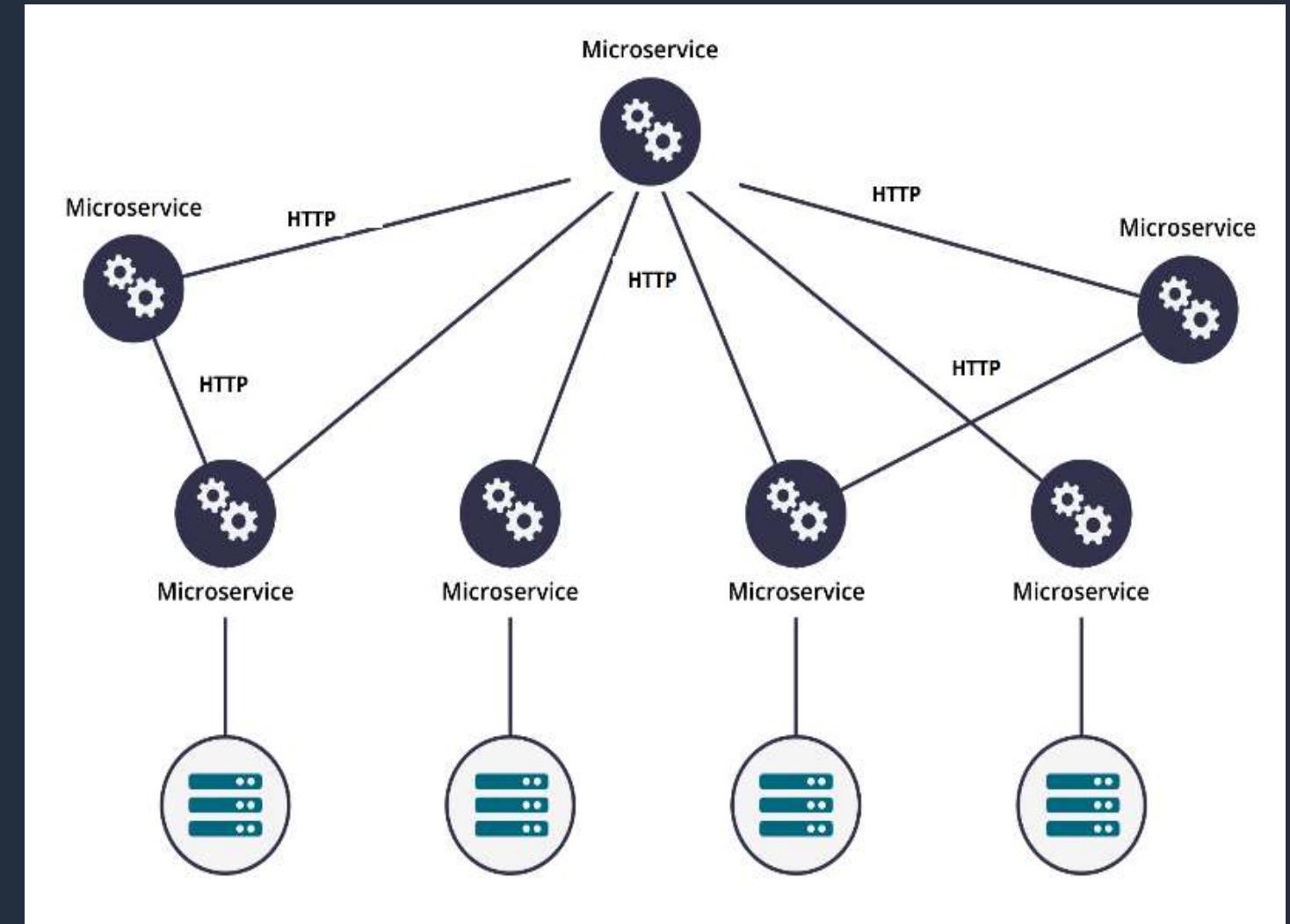
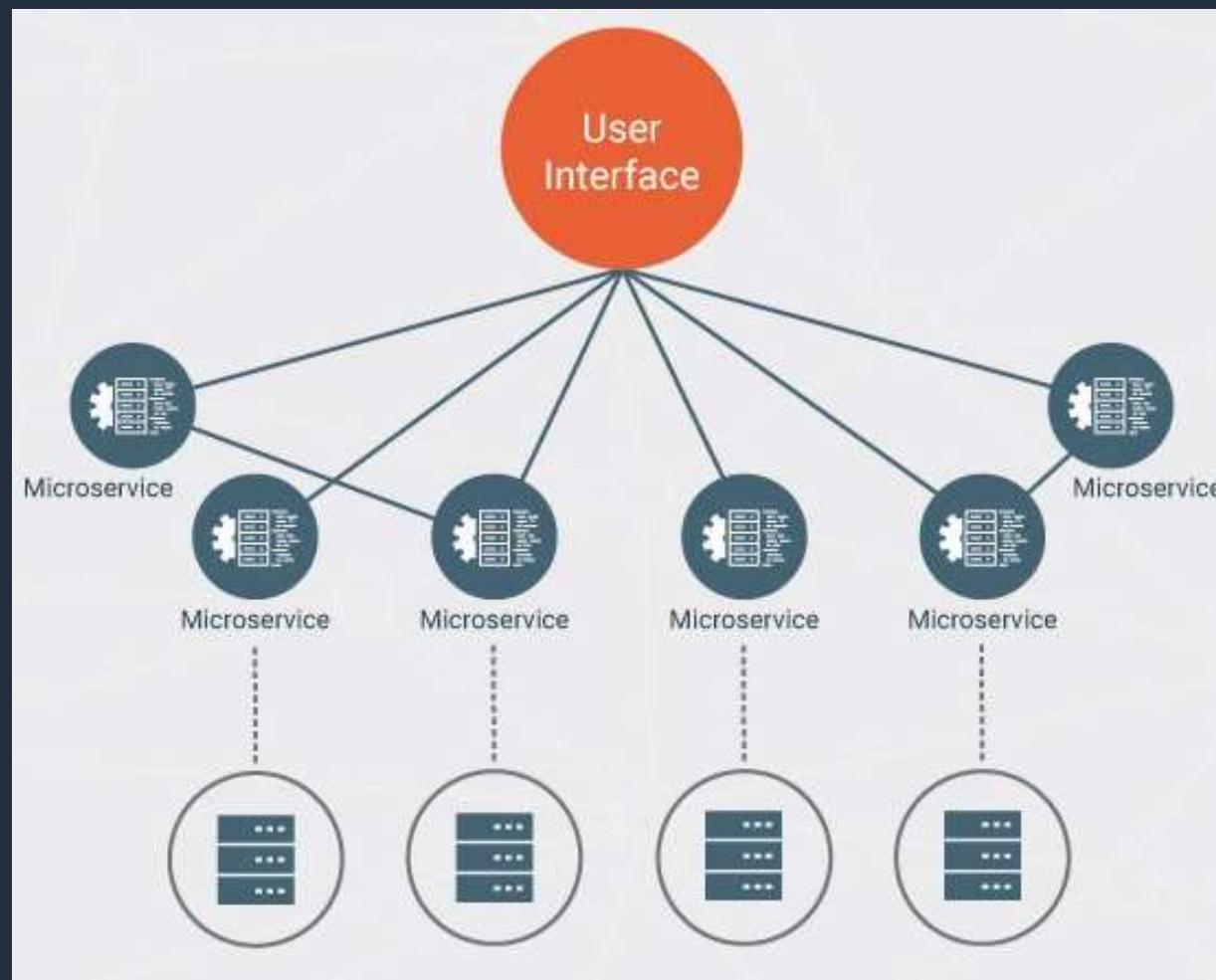
Los microservicios son más pequeños, desacoplados para realizar pequeñas funciones de manera eficiente.

Necesitaremos componentes que permiten una correcta implementación de la arquitectura. **Estos componentes deben de cubrirse por la plataforma y no propiamente en los microservicios**



# Persistencia

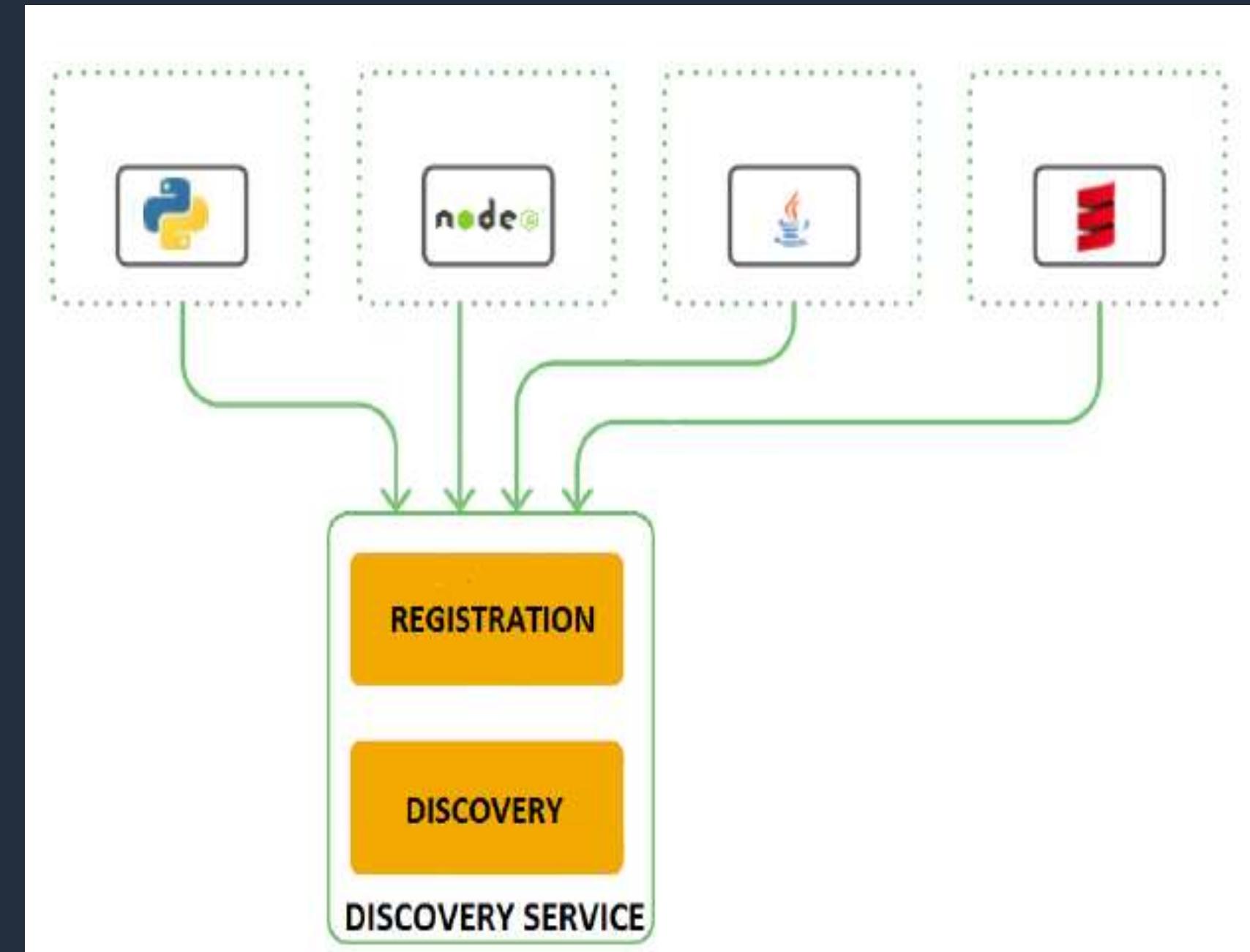
Igual que las aplicación monolíticas los microservicios pueden acceder a base de datos través de capas de persistencia.



# Discovery

En la medida que tenemos más demanda se hace necesario que despleguemos mayor número de microservicios o instancias de manera automática.

Se hace necesario que descubramos y registremos los nuevos servicios creados automáticamente.



# Discovery

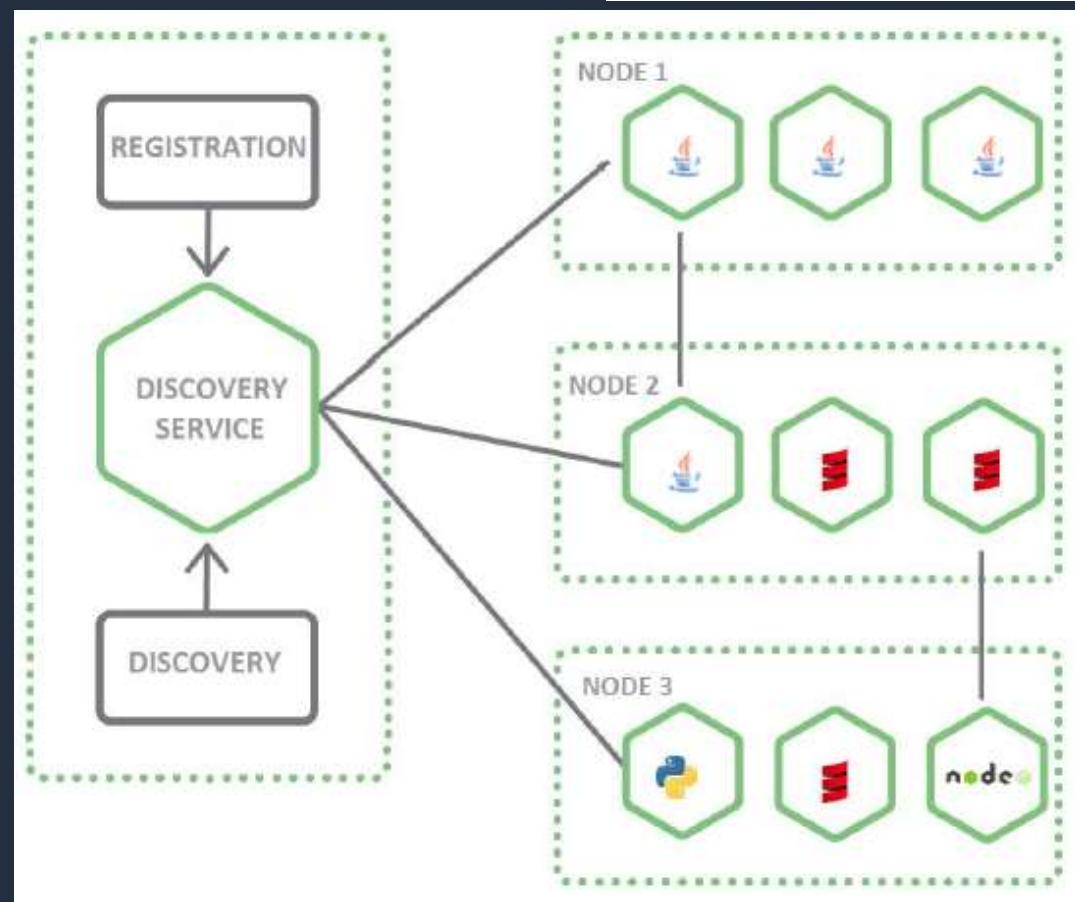
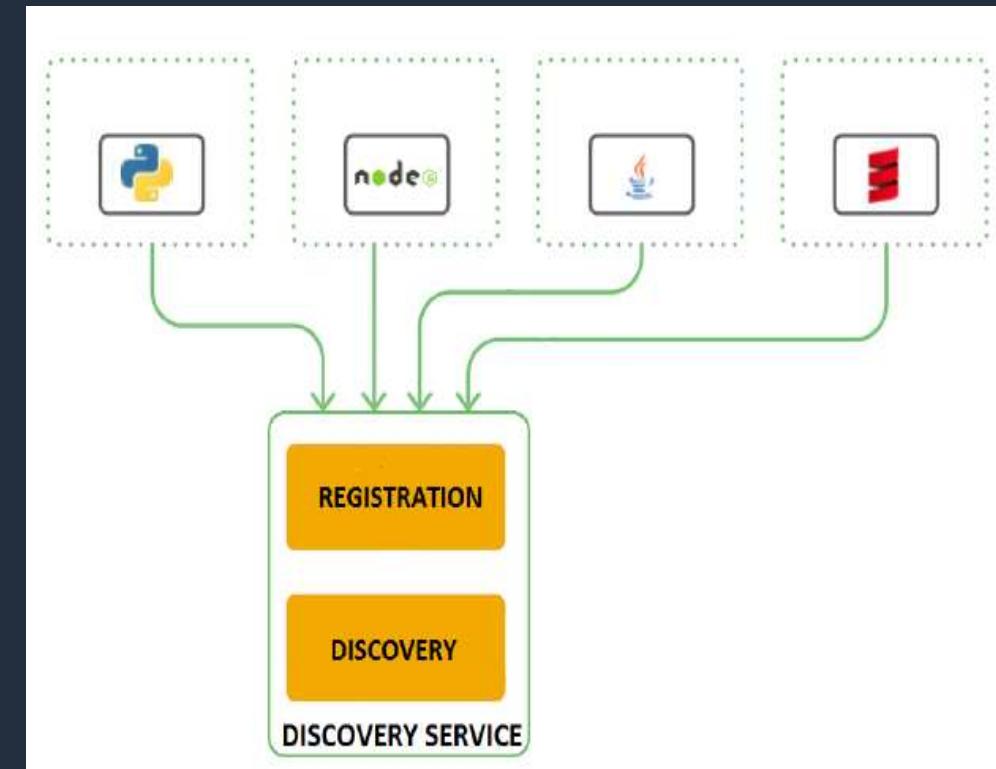
En la medida que tenemos más demanda se hace necesario que despleguemos mayor número de microservicios o instancias de manera automática.

Se hace necesario que descubramos y registremos los nuevos servicios creados automáticamente.

Componentes:

- **Registro:** Consiste en el registro de cada nueva instancia, dejando consignado sus datos de ubicación, como nodo, puerto e ip.

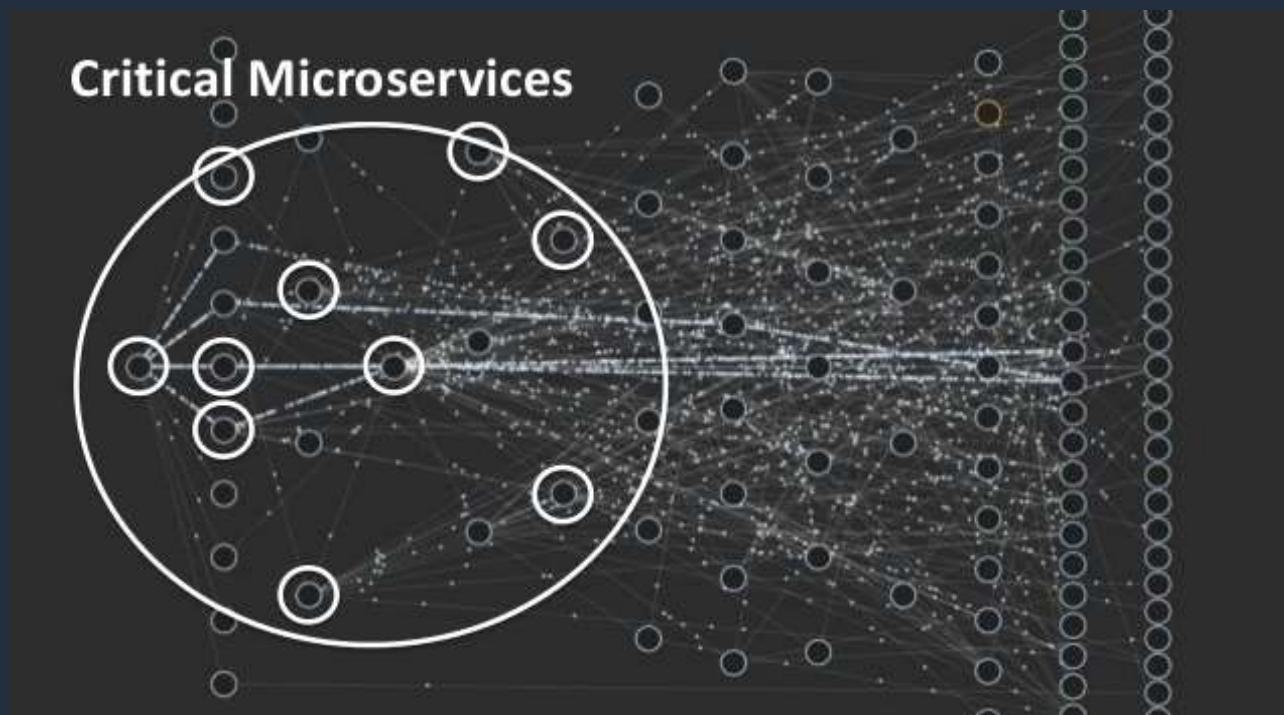
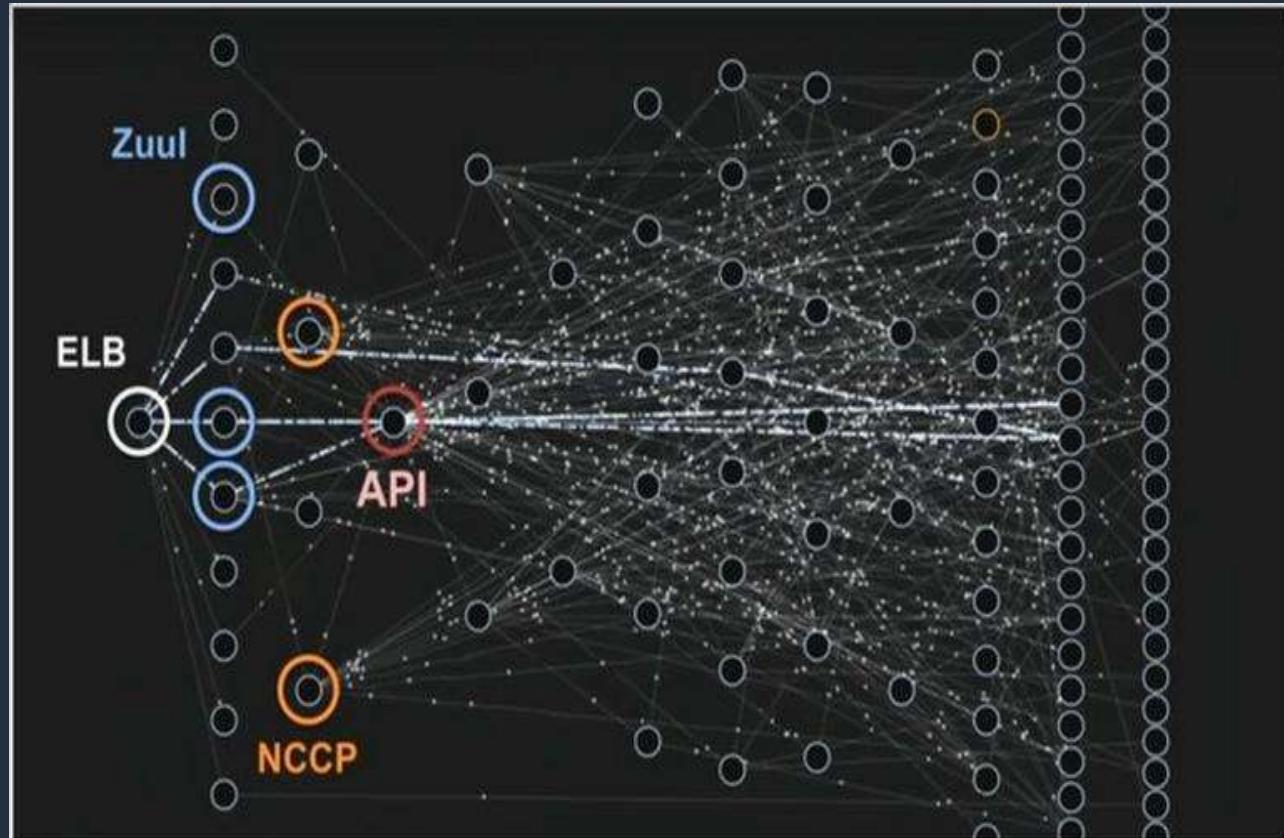
- **Descubrimiento:** Es el proceso que permite tener acceso a la información del registro.



# Scaling

Escalar (Scaling) en función de la demanda puede ser configurado con base al uso de cpu, memoria, request o métricas que se pueden customizar. Sus características son:

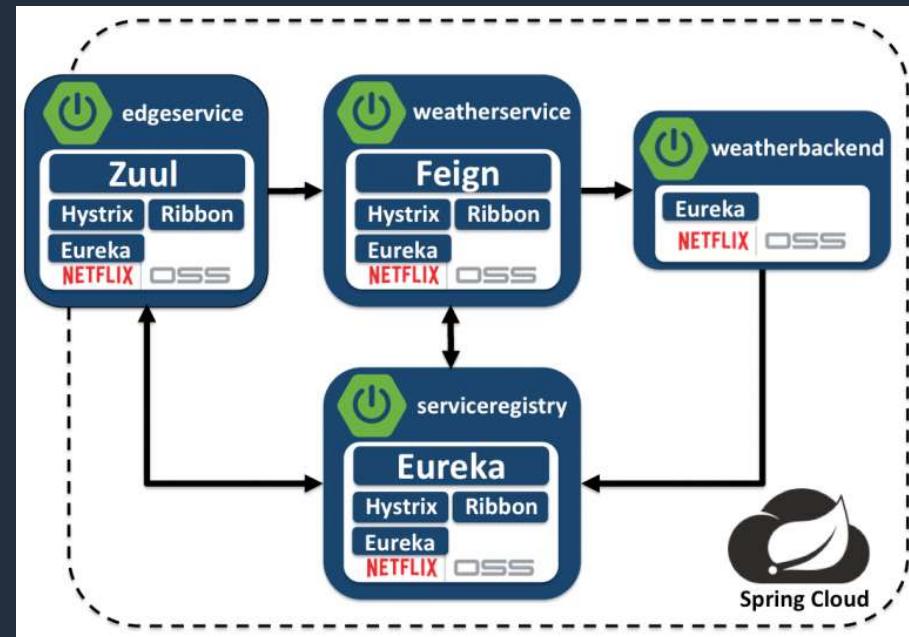
- Escalado horizontal.
- Alta tolerancia a fallos: Recuperación automática por configuración
- Despliegues basados en estrategias CI / CD.
- Abstracción de la capa de microservicios (Multi Runtime).



# Scaling

Herramientas disponibles para el servicio de Descubrimiento:

- Eureka (Spring Cloud Netflix)
  - Zookeper
  - Etcd
  - Consul
- 
- Herramientas para la orquestación y Scaling:
    - Kubernetes
    - Docker swarm
    - RancherOS
    - Openshift
    - Docker EE
    - EKS



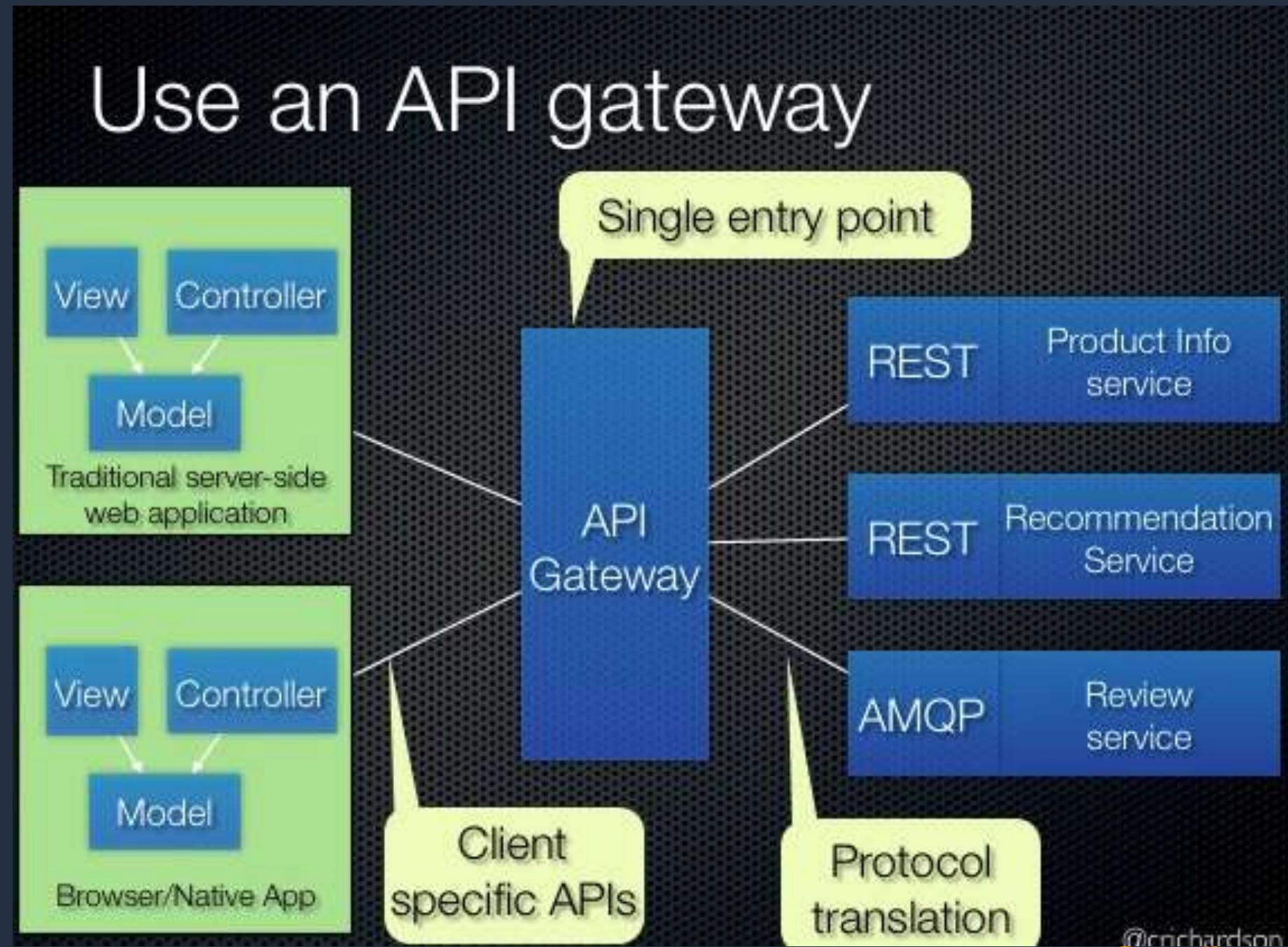
Pivotal CF



# API Gateway

Es importante resaltar componentes de un API Gateway:

- Solicitudes a través de un único punto de entrada (API Gateway).
- Mapeo global de request (Url's).
- Abstracción de los microservicios (MultiRuntime)
- Filtros dinámicos, redirecciónamiento con base en localización de las solicitudes.
- Zuul (Spring Cloud Netflix)
- TYK
- KONG
- Haproxy
- API Gateway (AWS)
- CA API Gateway
- Apigee



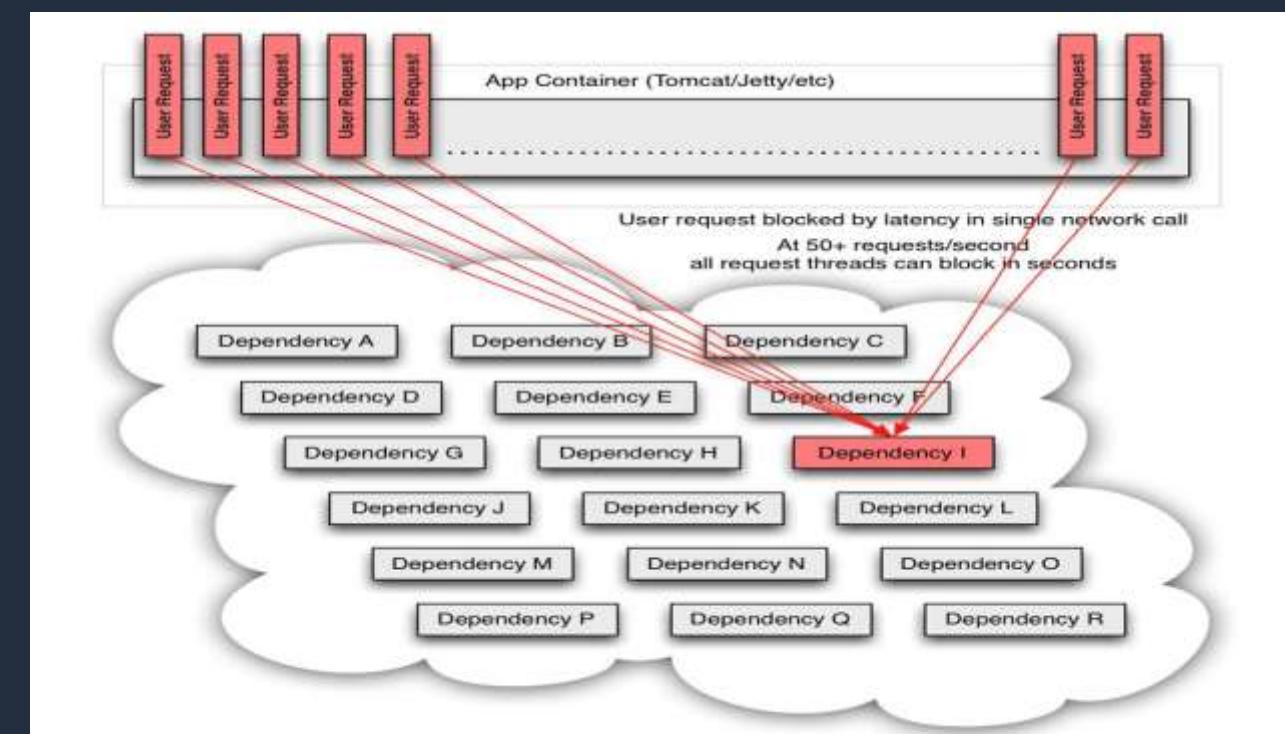
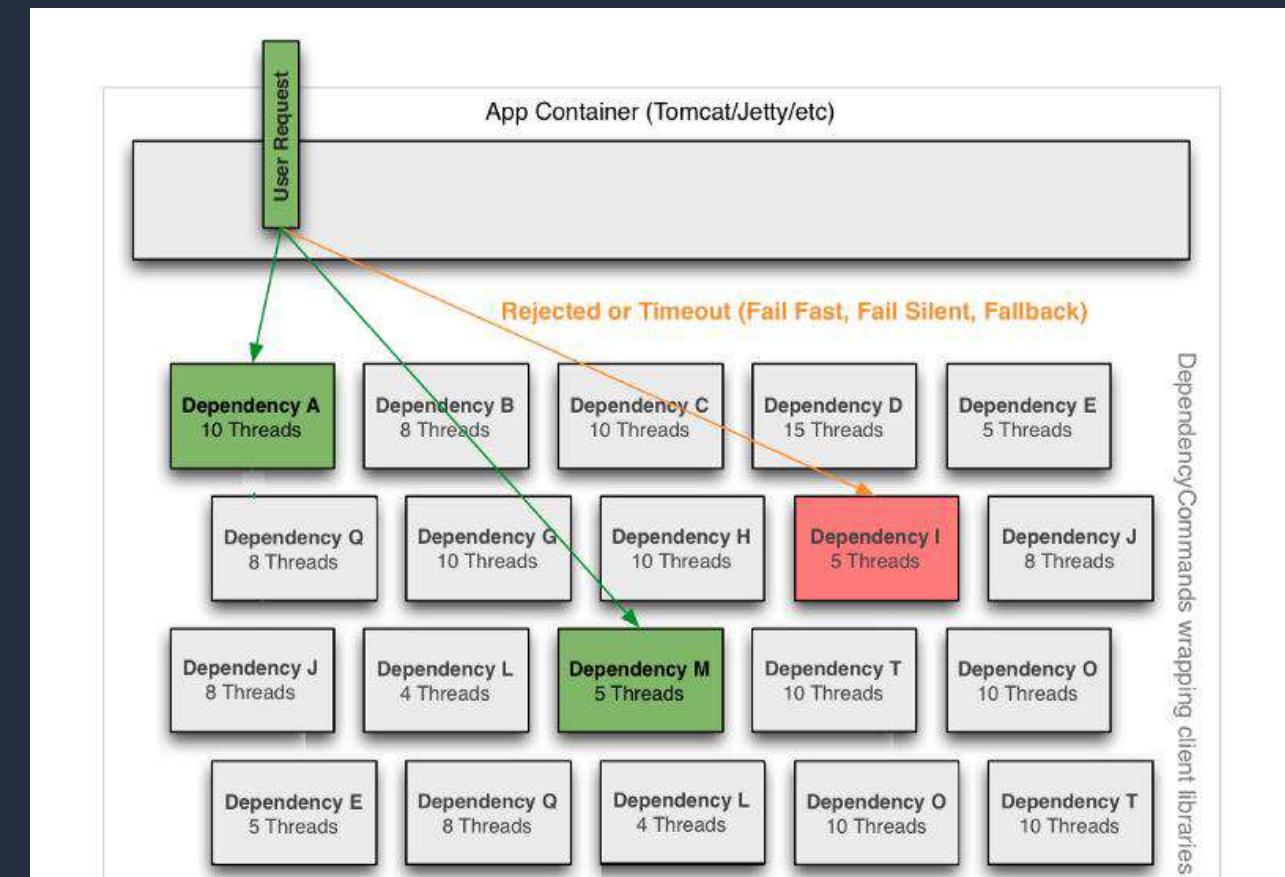
# Resiliencia y Balanceo

En las arquitecturas distribuidas una falla o demora en la respuesta puede causar una caída total de la aplicación.

Resiliencia

Es recomendado contar con un balanceador de carga externo que nos permita mejorar la fiabilidad de nuestra aplicación, incrementando además la disponibilidad y tolerancia a fallos.

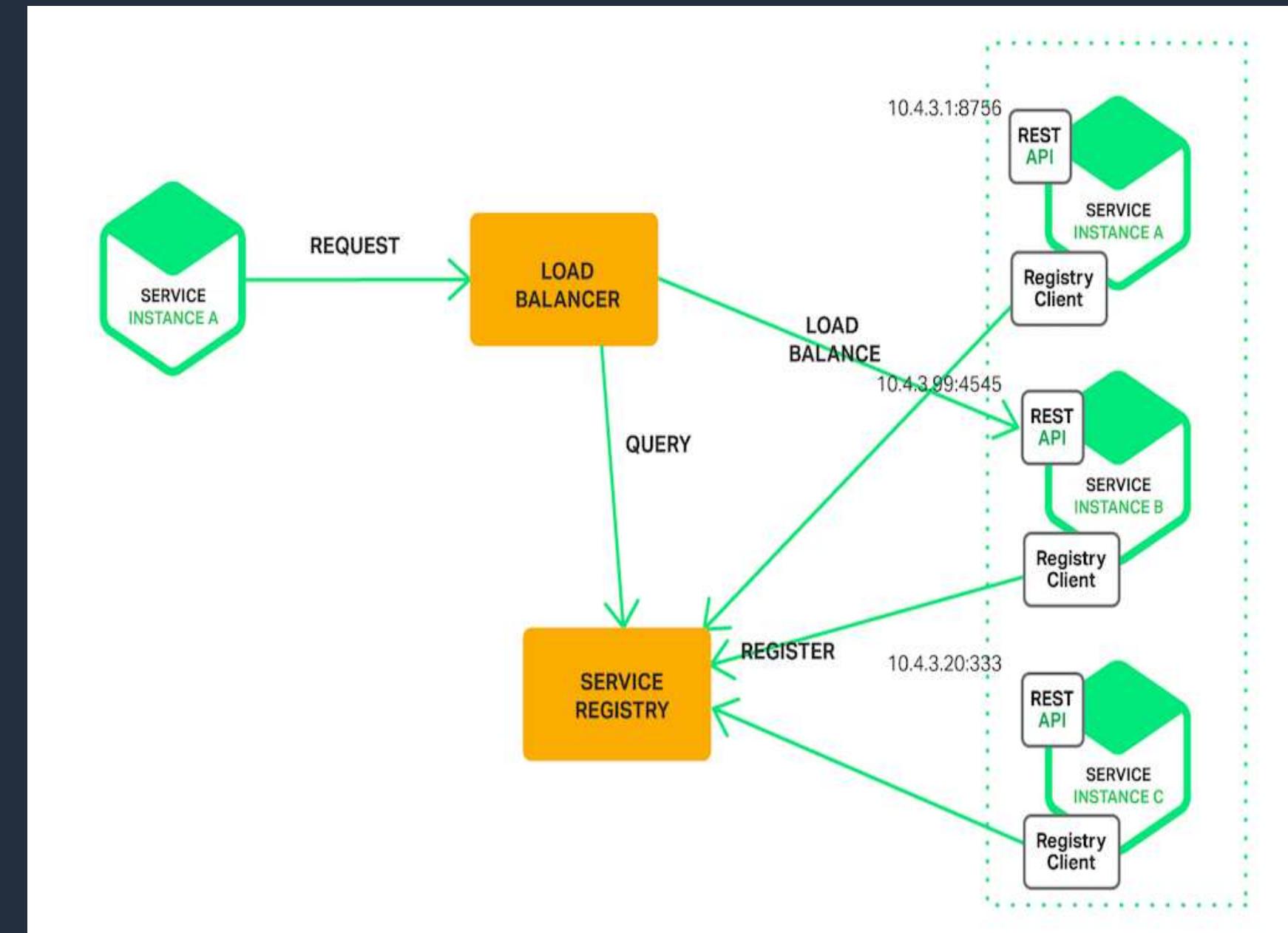
- AWS
- ELB
- F5



# Balanceo por recurso

Balancear la carga entre las distintas instancias de un microservicio, haciéndolo más efectivo a la hora de manejar las peticiones.

- Configuración de políticas de balanceo.
- Integración con el servicio de descubrimiento.
- Ejemplos:
  - Kubernetes Balanceo por POD
  - Ribbon (Spring Cloud Netflix).



# Monitoreo, Gestión de Logs

Un aspecto importante en la arquitectura a microservicios es la monitorización, contar con un panel unificado con información en tiempo real de cada microservicio o multiples servicios.

Ejemplos:

- Hystrix + turbine + Spring boot admin
- InfluxDB + Grafana
- Service Mesh (Istio, Linkerd)



La centralización y explotación de logs juega un papel importante de lo contrario seria **inmanejable la administración de los mismos.**

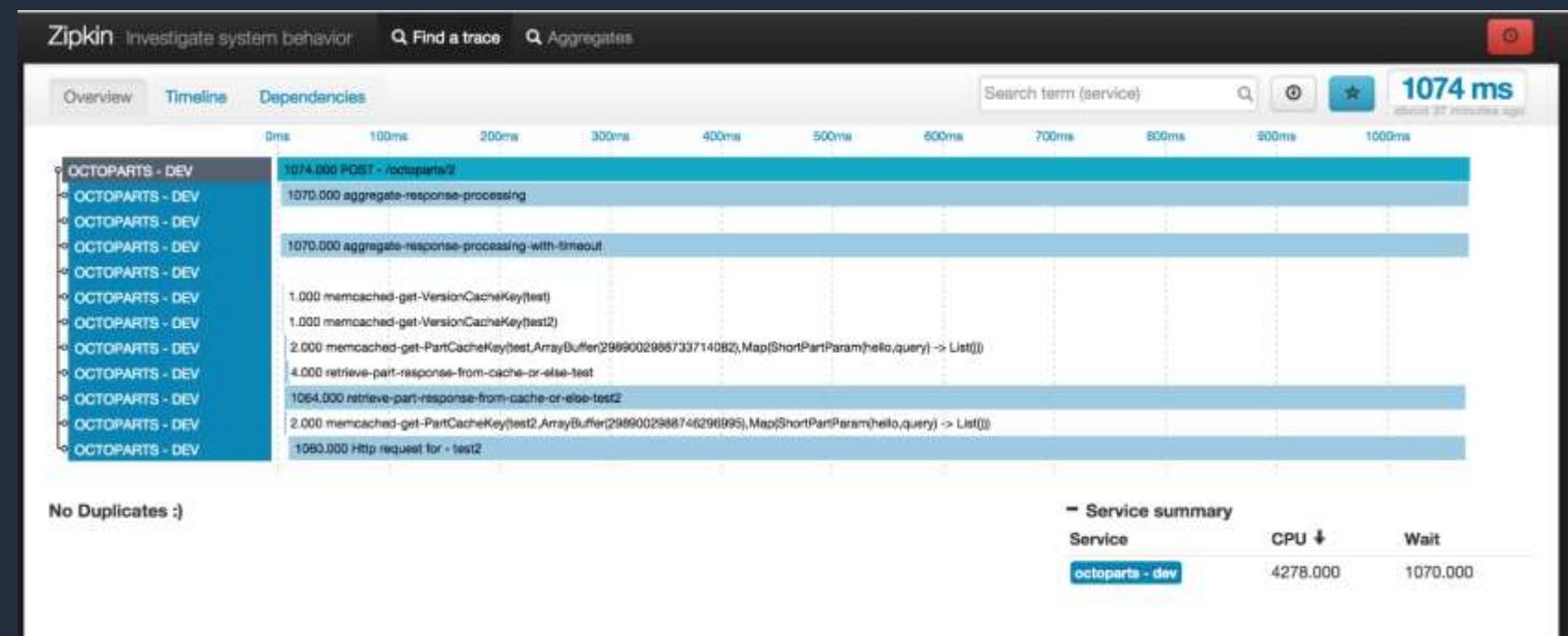
Ejemplos:

- EFK
- ELK
- Cloud Watch + S3 AWS



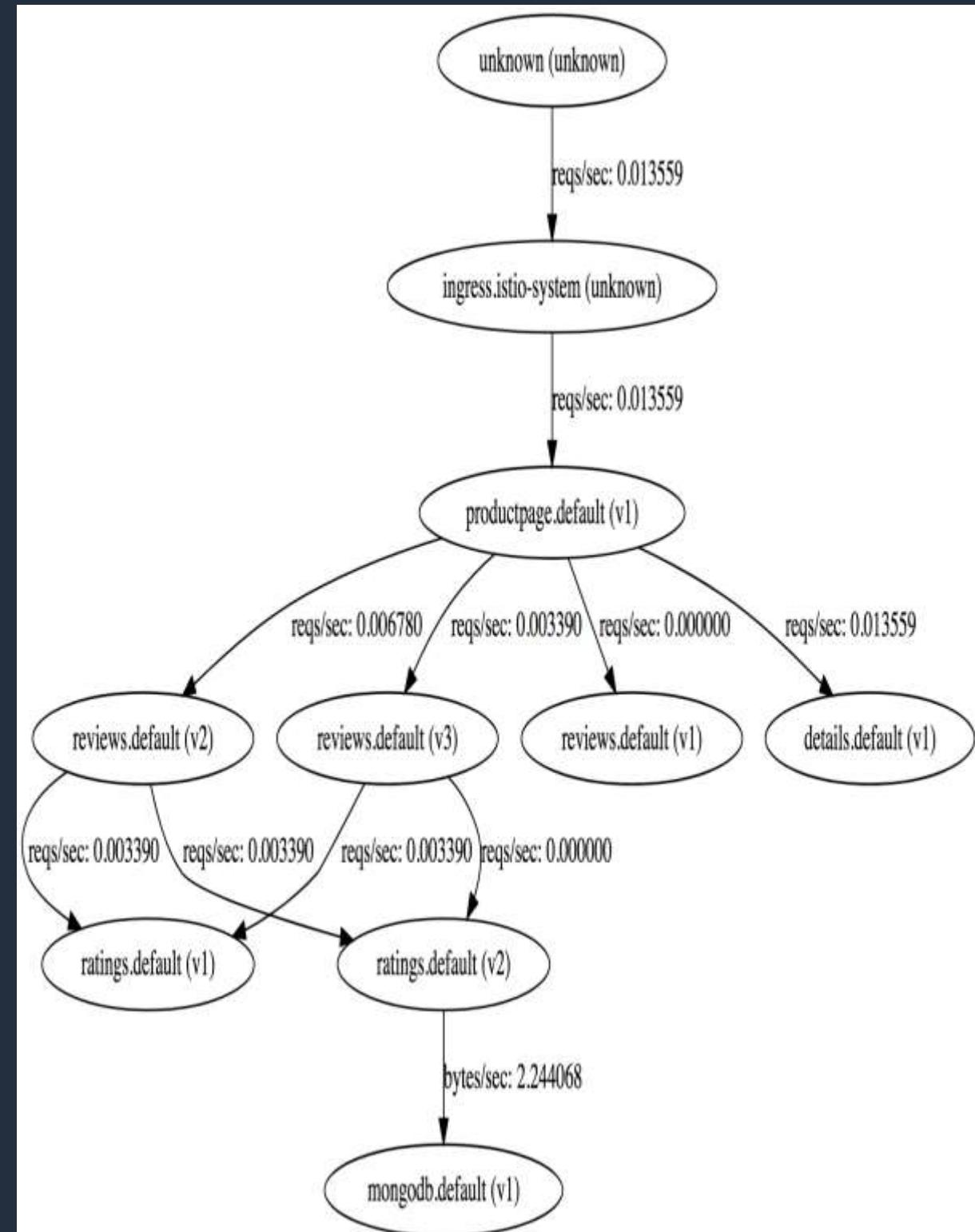
# Trazabilidad Distribuida

En un sistema distribuido debemos contar con la capacidad de registrar en nuestros logs las trazas a nivel de request.



Ejemplos:

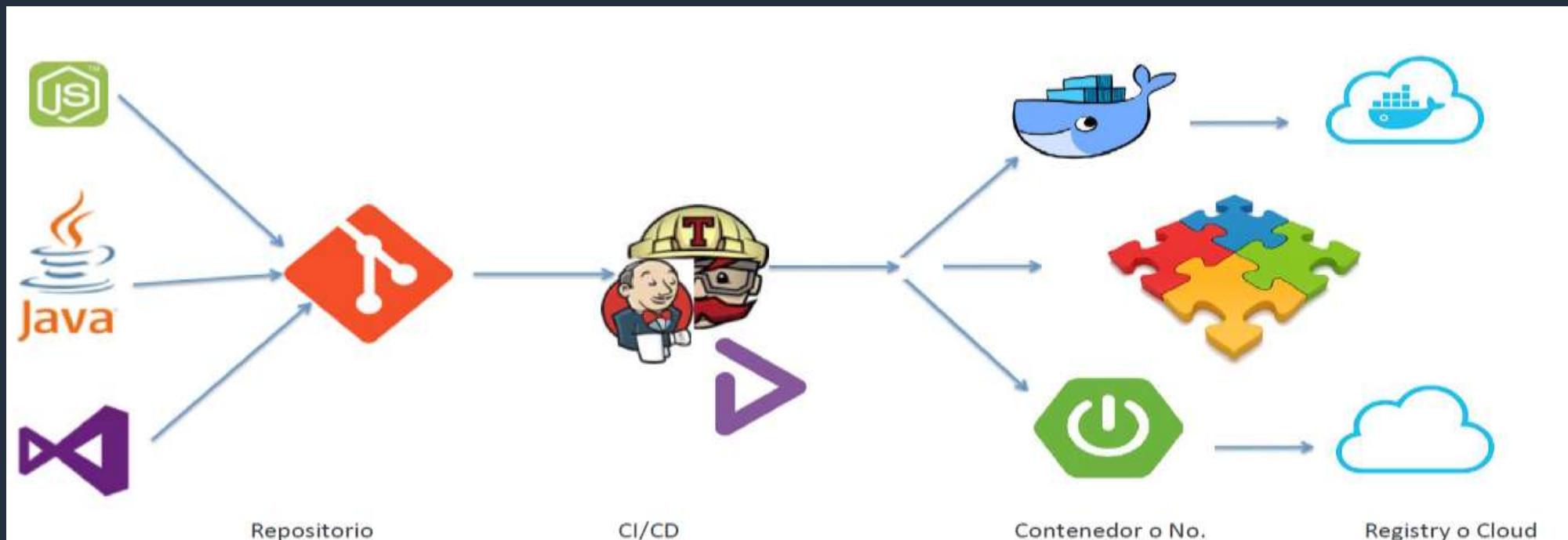
- Spring Sleuth + Zipkin
- Istio – Service Mesh
- Zipkin / Jaeger



# CI / CD Continuous Integration & Continuous Deployment

La creación de Microservicios es apoyada por herramientas IaaS y PaaS que complementan el proceso de automatización.

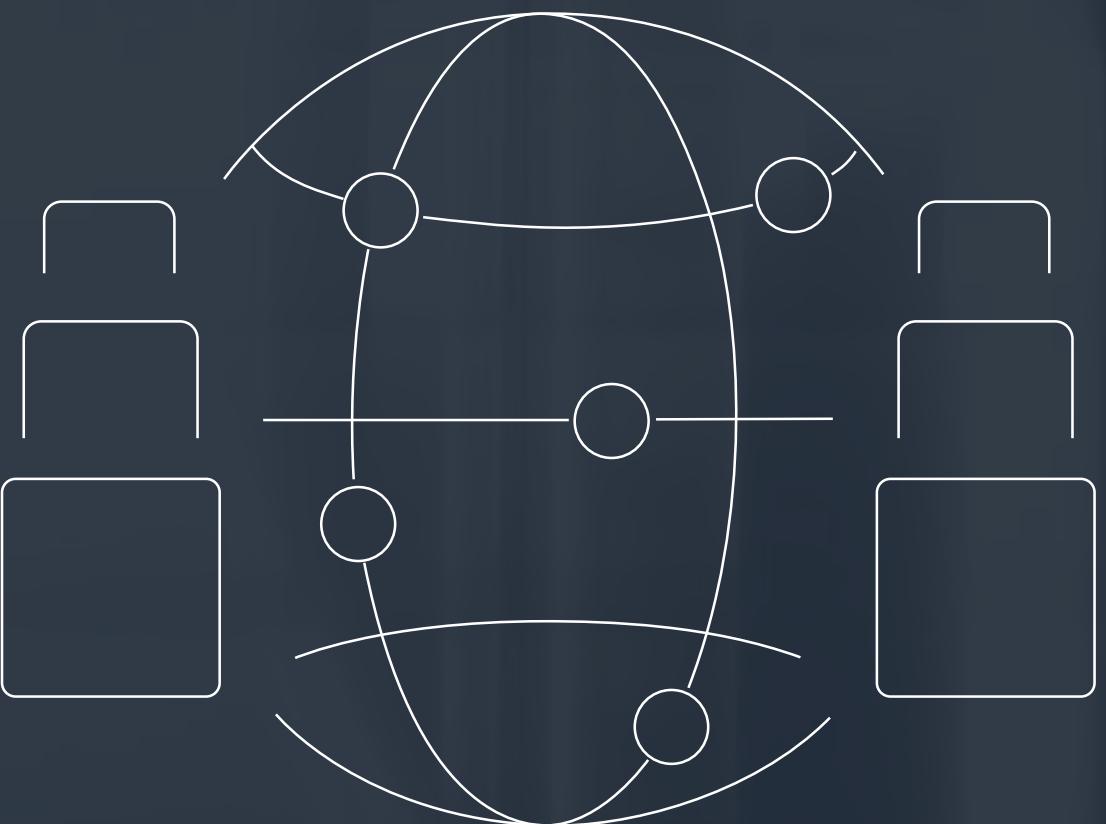
- Dependiendo de la infraestructura y plataforma podrá variar el ciclo de CI/CD.
- Es aquí donde Docker juega un papel importante ya que nos garantiza el despliegue de nuestros microservicios en cualquier plataforma.



# Sección 2

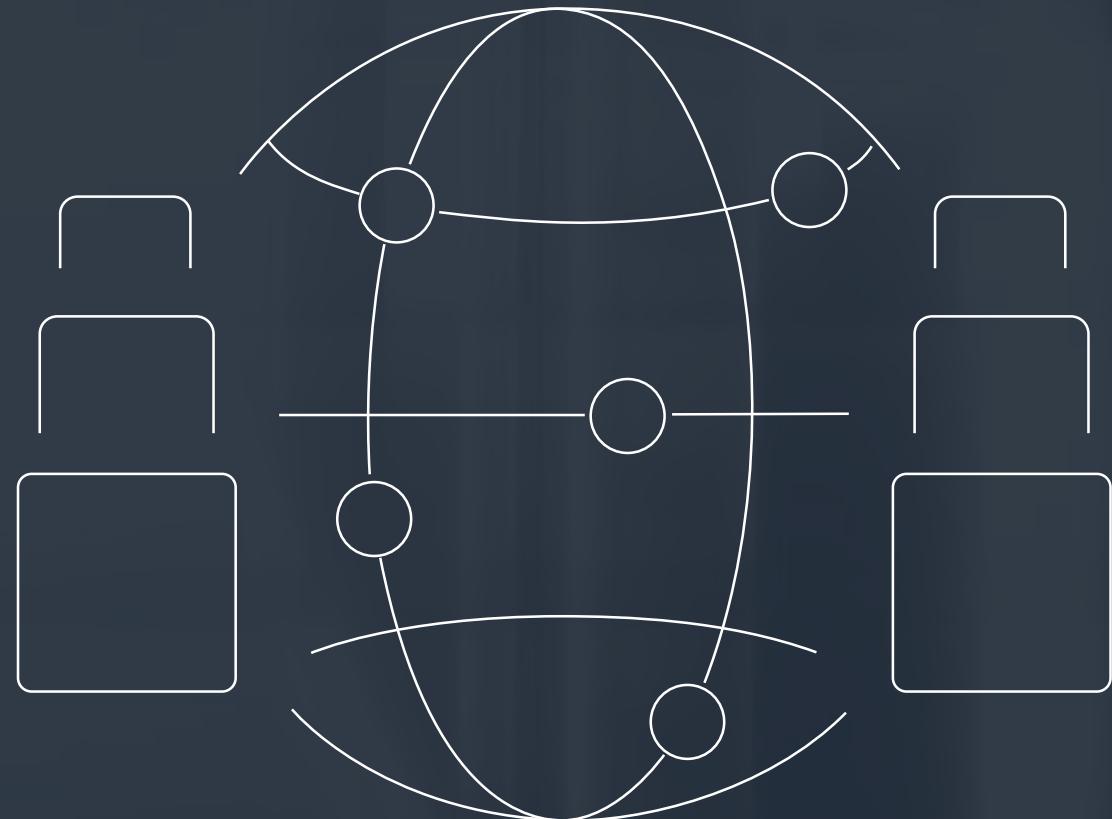
## Contenedores

- \* Docker
- \* Docker Compose



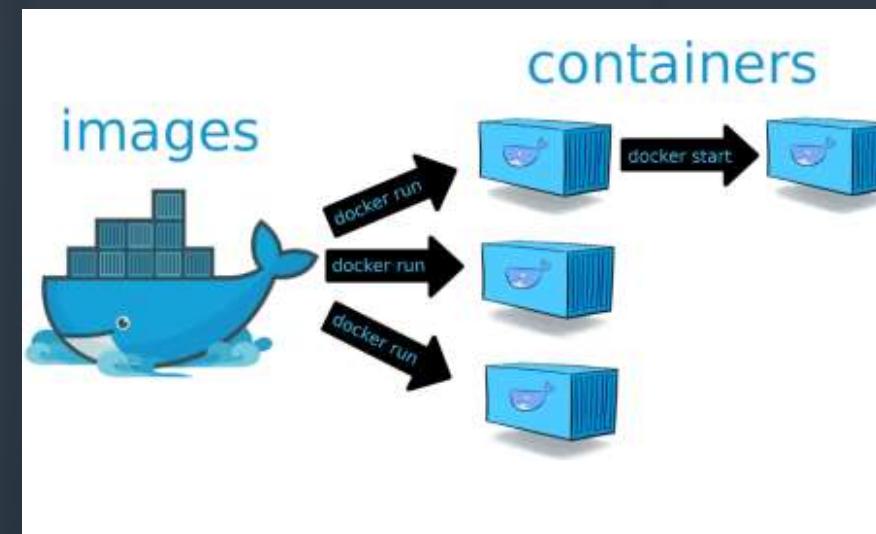
# ¿Qué es un contenedor y para qué sirve?

Un contenedor es un recipiente de carga para el transporte marítimo o fluvial, transporte terrestre y transporte multimodal.



# ¿Qué son los contenedores en TI?

Los contenedores son unidades ejecutables de software donde se empaqueta el código de una aplicación, junto con sus bibliotecas y dependencias, de forma común para que se pueda ejecutar en cualquier lugar, ya sea en el escritorio, en la TI tradicional o en la nube.



# Contenedores

## Virtualización VM

Las máquinas virtuales son la abstracción de hardware físico por medio de un hypervisor.

Cada máquina virtual ve el hardware emulado por el hypervisor.

Cada máquina virtual requiere de un sistema operativo.

## Contenedores

**Es una forma de virtualización muy ligera**

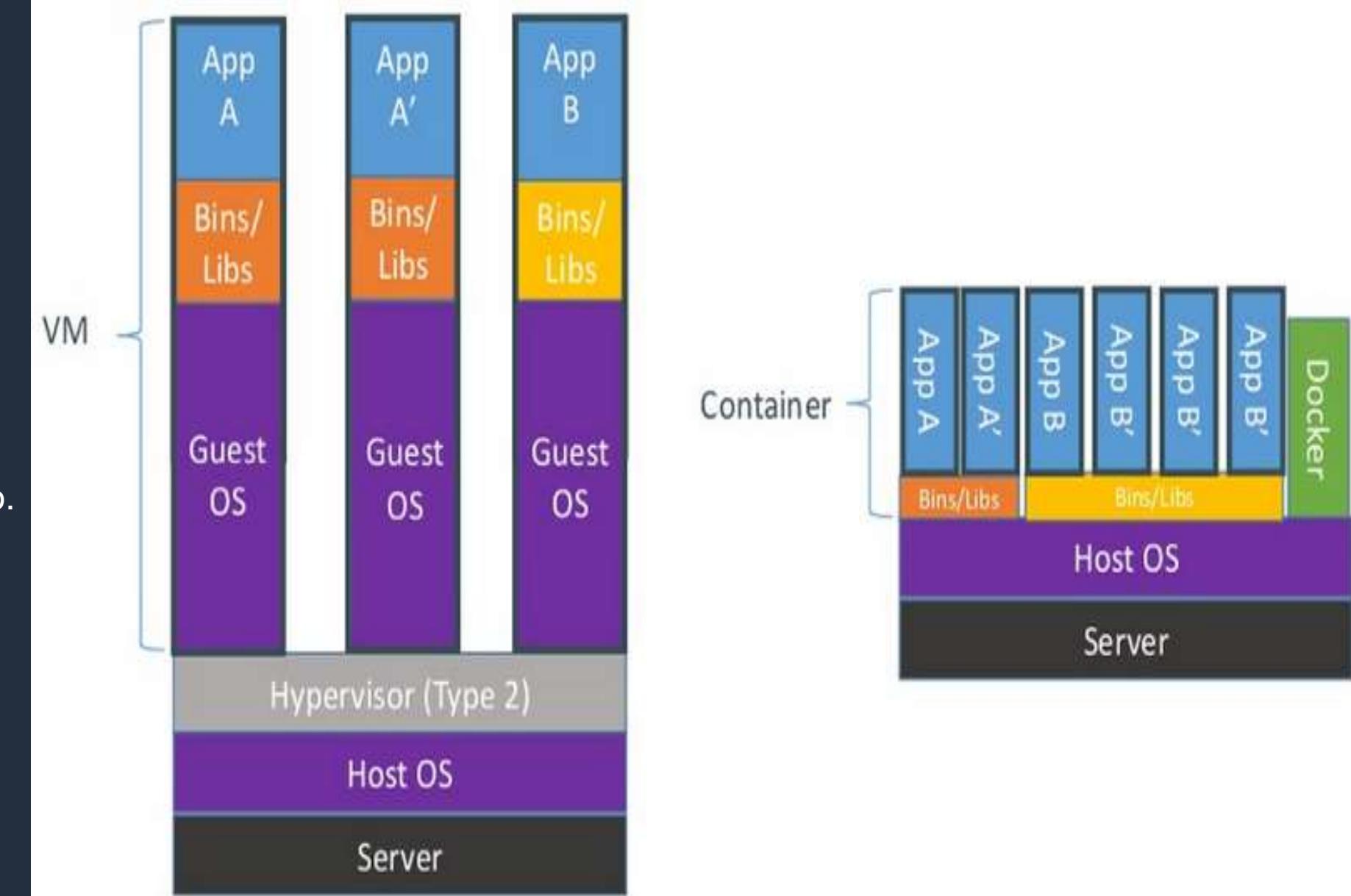
**Son la abstracción de una aplicación o servicio que empaqueta el código y sus dependencias juntas.**

**Multiples contenedores pueden ser desplegados en la misma máquina física o virtual y comparten el mismo sistema operativo base.**

**Cada contenedor se ejecuta de manera aislada.**

**Tiene sus propios recursos aislados de CPU, memoria, I/O y recursos de red (dirección IP)**

**Deben correr sobre un ambiente huésped (host)**



**Un contenedor comparte recursos de hardware con otros contenedores sin necesidad de apartarlos**

**Comparte el kernel del SO huésped**



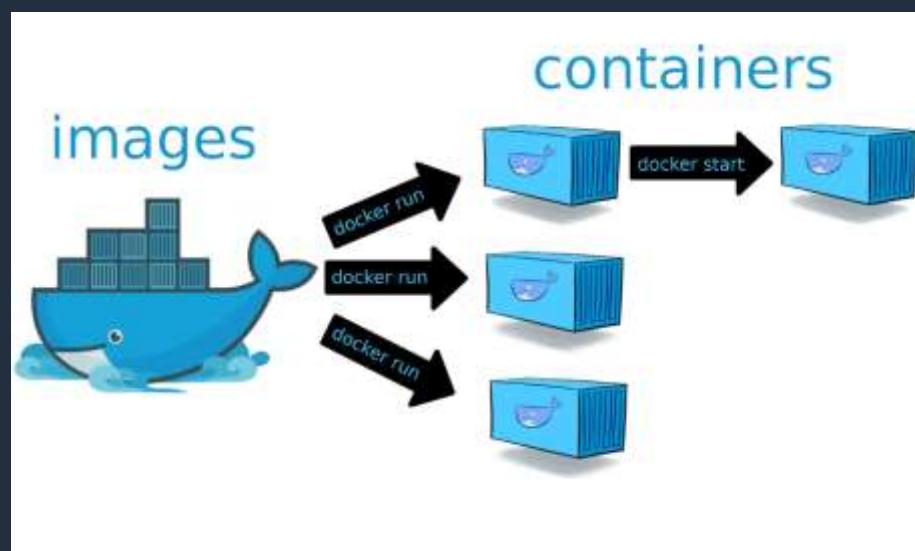
# Contenedores Docker

## Beneficios de Contenedores

- Portabilidad.
- Uso más eficiente de recursos.
- Agilidad en su aprovisionamiento y despliegue, ideales para la integración y entrega continua de los microservicios.
- En sistemas grandes, el uso de VMs implica duplicar instancias del mismo SO y muchos volúmenes de arranque redundantes.
- Debido a que los Linux Containers (LXC) son más eficientes y ligeros comparados a VMs, se pueden ejecutar más contenedores que VMs en el mismo hardware.
- Los contenedores desacoplan las aplicaciones del sistema operativo; esto implica que los usuarios pueden tener un Linux muy limpio y correr cualquier otra aplicación en contenedores aislados.
- Además de que el sistema operativo está también abstraído de los contenedores, se pueden mover los contenedores en cualquier Linux que soporte el entorno ejecución del contenedor.

## Inconvenientes de Contenedores

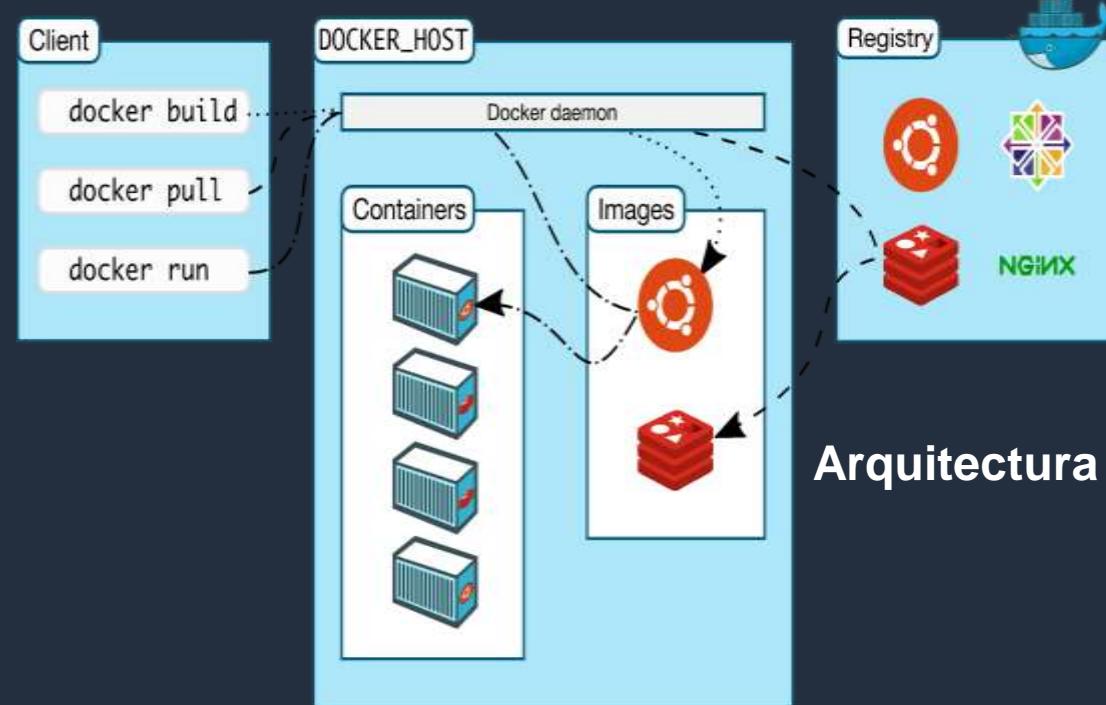
- Falta de aislamiento de su sistema operativo base incrementando el riesgo de amenazas de seguridad.
- Se puede reducir desplegando los contenedores en máquinas virtuales o desplegando cada contenedor en cada máquina física lo cual puede ser muy costoso.



# Docker Containers

Los contenedores de Docker te permiten desplegar, replicar, mover y respaldar una carga de trabajo de forma más fácil que usando Mvs.

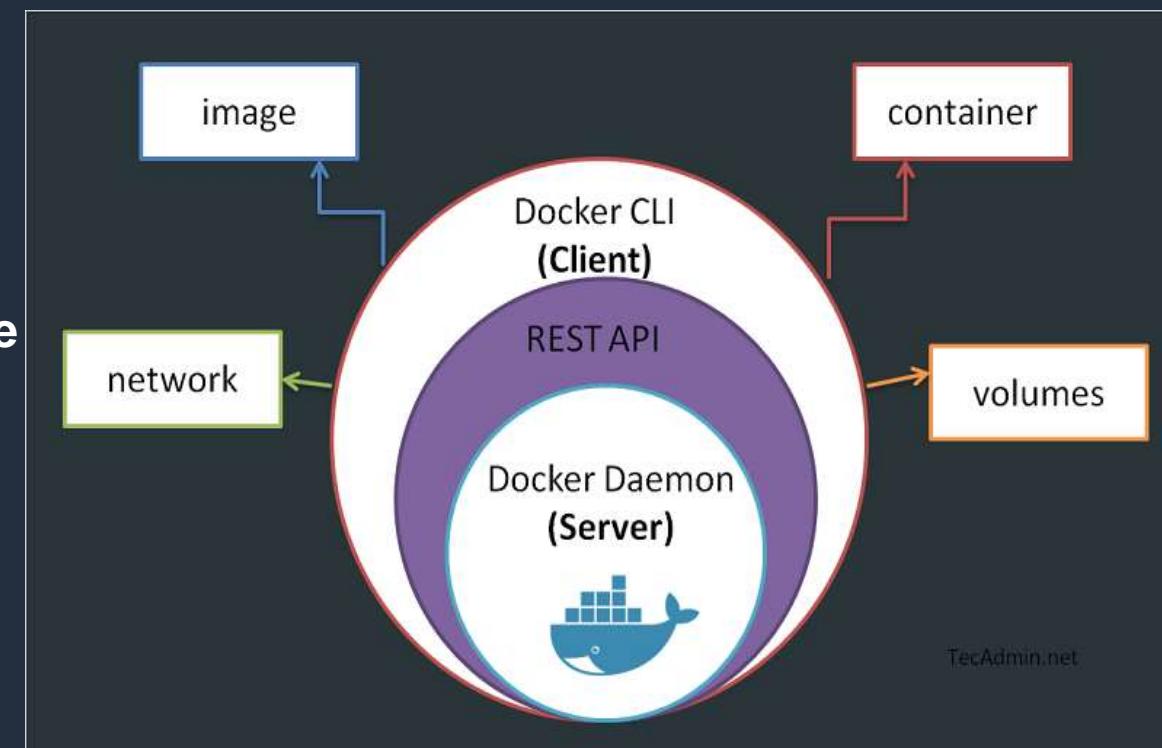
- Docker no soporta almacenamiento persistente.
- Para almacenamiento se usa el huésped a través de volúmenes de Docker.
- Docker proporciona “layers” de solo lectura.
- Cada cambio genera un nuevo layer que puede ser guardado, lo que genera una nueva versión de la imagen del contenedor.
- Los contenedores Docker son Stateless no guardan estado durante su ejecución, para esto tenemos los volúmenes o storage persistente



## Arquitectura Docker

- Docker daemon  
Corre en la máquina huésped. No se interactúa directamente, sino a través del Docker client.
- Docker client  
Interface primaria de Docker, acepta comandos de usuario y se comunica de ida y vuelta con el Docker daemon.
- Docker images  
Plantilla de solo lectura, se usan para crear contenedores de Docker. Una imagen puede tener una versión específica de Linux con tu aplicación instalada. Se pueden usar imágenes ya creadas o crear una personalizada.
- Docker registries  
Lugar donde se almacenan las imágenes. Pueden ser públicos o privados. DockerHub (<https://hub.docker.com>).
- Docker containers  
Son similares a un directorio. Almacena todo lo que una aplicación requiere para ejecutarse. Se crean a partir de una imagen. Pueden ser arrancados, iniciados, detenidos, movidos y borrados. Cada contenedor es una plataforma de aplicaciones aislada y segura.

## Docker Engine



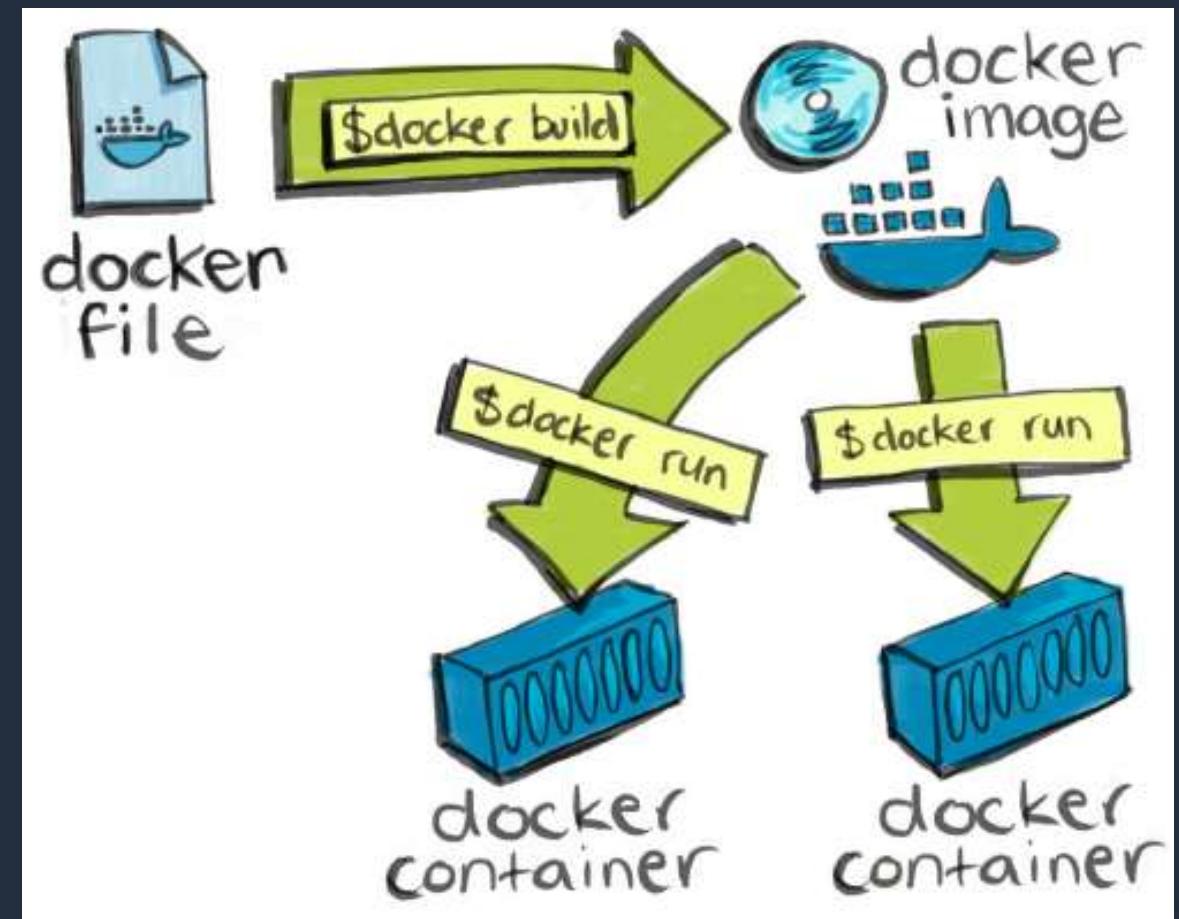
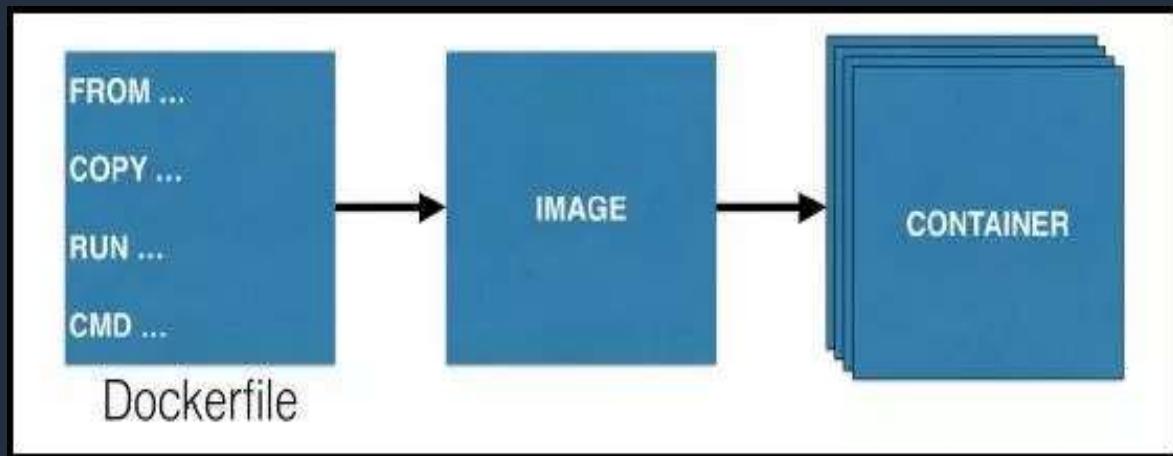
# Docker Image

- Cada imagen consiste en un conjunto de layers.
- Docker usa union file systems para combinar las capas en una sola imagen.
- Union file system permiten que archivos y directorios de Sistemas de archivos separados (branches), transparentemente sobreponen, formen un único y coherente file system.
- Cuando se cambia algo en la imagen, se construye un nuevo layer, por lo tanto solo se debe actualizar los nuevos layers en lugar de toda la imagen.

Cada imagen inicia de una imagen base, por ejemplo ubuntu o fedora.

- Las imágenes se crean a partir de estas imágenes base, usando pasos de un conjunto simple y sencillo de instructions.
- Cada instruction crea un nuevo layer en la imagen. Las instructions son acciones, ejemplo:
  - Ejecutar un comando
  - Agregar un archivo o directorio
  - Crear una variable de ambiente
  - Qué proceso se va a ejecutar cuando se lance un contenedor de la imagen.
- Las instructions se almacenan en un archivo llamado Dockerfile.
- Cuando se solicita construir la imagen, Docker lee el Dockerfile y ejecuta las instructions y regresa la imagen final.

Un Dockerfile es un script en archivo de texto que contiene las instrucciones y comandos para construir la imagen a partir de la imagen base.



# Docker Image

- Consiste en un sistema operativo, archivos agregados por el usuario y meta-data. Se construyen a partir de una imagen.
- La imagen indica lo que el contenedor almacena, el proceso que va a ejecutar. Es de solo lectura

## Tag en Docker

registryHost:puerto/{usuario}/nombreDelRepo:tag

- El registryHost por default es: hub.docker.com
- Puerto default es 443
- El tag por default es latest
- {usuario} es opcional

Ejemplo:

landa2910/miimagen:latest

landa2910/miimagen:0.1



# Hands-ON Docker Init

docker version  
docker -v

```
145      </p>
146    </li>
147  </ul>
148 </div>
149  );
150 }
151
152  renderWhatsNewLinks() {
153  return (
154    <div className={style.container}>
155      <h4 className={style.title}>
156        <ul className={style.list}>
157          {this.renderLinks(
158            this.renderWhatsNewLinks,
159            this.renderFooterMain,
160            this.renderFooterSub,
161            this.renderFooterSlogan,
162            this.renderFooterGlobal
163          )}
164        </ul>
165      </div>
166    );
167  }
168
169  renderWhatsNewItem(title, url) {
170  return (
171    <li className={style.footerItem}>
172      <a href={trackUrl(url)} target="_blank" rel="noopener noreferrer">
173        <span>{title}</span>
174      </a>
175    </li>
176  );
177}
178
179  renderFooterSub() {
180  return (
181    <div className={style.footerSub}>
182      <Link to="/" title="Home - Display" type="logo" className={style.footerSubLogo}>
183        <Icon type="logo" />
184      </Link>
185      <span className={style.footerSlogan}>
186        {this.renderFooterSlogan}
187      </span>
188    </div>
189  );
190}
191
192  renderFooterSlogan() {
193    return (
194      <span>{"Display"}</span>
195    );
196  }
197
198  render() {
199  return (
200    <footer className={style.footerGlobal}>
201      <div className="container">
202        {this.renderFooterMain()}
203        {this.renderFooterSub()}
204      </div>
205    </footer>
206  );
207}
```

# Hola Mundo Docker

docker run hello-world

docker container run hello-world

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic$ docker run hello-world
Unable to find image 'hello-world:latest' locally
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:0e11c388b664df8a27a901dce21eb89f11d8292f7fca1b3e3c4321bf7897bffe
Status: Downloaded newer image for hello-world:latest
```

Hello from Docker!

This message shows that your installation appears to be working correctly.

To generate this message, Docker took the following steps:

1. The Docker client contacted the Docker daemon.
2. The Docker daemon pulled the "hello-world" image from the Docker Hub.  
(amd64)
3. The Docker daemon created a new container from that image which runs the executable that produces the output you are currently reading.
4. The Docker daemon streamed that output to the Docker client, which sent it to your terminal.

To try something more ambitious, you can run an Ubuntu container with:

```
$ docker run -it ubuntu bash
```

Share images, automate workflows, and more with a free Docker ID:

```
https://hub.docker.com/
```

For more examples and ideas, visit:

```
https://docs.docker.com/get-started/
```

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic$ █
```



# Hola Mundo Docker – Bajando imágenes

```
jovani@jovani-dev: ~/Documents/apixcloudservice/arquitecto_apis_microservicios/03_microservices/lab1_contene... ◻
File Edit View Search Terminal Help
jovani@jovani-dev:~/Documents/apixcloudservice/arquitecto_apis_microservicios/03
microservices/lab1_contenedores$ docker pull hello-world
Using default tag: latest
latest: Pulling from library/hello-world
1b930d010525: Pull complete
Digest: sha256:9572f7cdcee8591948c2963463447a53466950b3fc15a247fcad1917ca215a2f
Status: Downloaded newer image for hello-world:latest
jovani@jovani-dev:~/Documents/apixcloudservice/arquitecto_apis_microservicios/03
microservices/lab1_contenedores$ docker images
REPOSITORY          TAG           IMAGE ID            CREATED             SIZE
hello-world         latest        fce289e99eb9    13 months ago      1.84kB
jovani@jovani-dev:~/Documents/apixcloudservice/arquitecto_apis_microservicios/03
microservices/lab1_contenedores$ █
```

docker pull nombre-imagen:tag  
docker pull hello-world  
docker images



# Eliminar contenedores

## docker images

```
jovani-dev@jovanidev:~$ docker images
REPOSITORY          TAG      IMAGE ID      CREATED        SIZE
hello-world         latest   bf756fb1ae65  6 months ago  13.3kB
jovani-dev@jovanidev:~$
```

Purgar todas las imágenes y contenedores  
docker system prune -a

```
jovani-dev@jovanidev:~$ docker system prune -a
WARNING! This will remove:
- all stopped containers
- all networks not used by at least one container
- all images without at least one container associated to them
- all build cache

Are you sure you want to continue? [y/N] y
Deleted Containers:
6da4be7d1638042d65ff8adbc92140cc8508b0b0335f23596d05709d1a7c731a

Deleted Images:
untagged: hello-world:latest
untagged: hello-world@sha256:49a1c8800c94df04e9658809b006fd8a686cab8028d33cfba2cc049724254202
deleted: sha256:bf756fb1ae65adf866bd8c456593cd24beb6a0a061dedf42b26a993176745f6b
deleted: sha256:9c27e219663c25e0f28493790cc0b88bc973ba3b1686355f221c38a36978ac63

Total reclaimed space: 13.34kB
jovani-dev@jovanidev:~$
```

# Sección I

## Imagenes Docker

### Tags - latest

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/01\\_primeros\\_pasos/01\\_primer\\_imagen](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/01_primeros_pasos/01_primer_imagen)

```
docker build -t miprimerImagen:0.0.1 .
docker build -t miprimerImagen:0.0.2 .
```

The image shows two terminal windows side-by-side. Both windows have a blue header bar with the text 'jovani-dev@jovanidev:~/Documents/certificatic/05\_microservices/lab1\_contenedores/01\_init/primer\_imagen\$' followed by a blue command line area.

**Left Terminal (Version 0.0.1):**

```
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_init/primer_imagen$ docker build -t jovaniac/miprimerImagen:0.1 .
Sending build context to Docker daemon 2.048kB
Step 1/2 : FROM ubuntu
--> 1e4467b07108
Step 2/2 : CMD echo "Mi primer imagen docker"
--> Using cache
--> 995b858bb0c5
Successfully built 995b858bb0c5
```

**Right Terminal (Version 0.0.2):**

```
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_init/primer_imagen$ docker build -t jovaniac/miprimerImagen:0.2 .
Sending build context to Docker daemon 2.048kB
Step 1/2 : FROM ubuntu
--> 1e4467b07108
Step 2/2 : CMD echo "Mi primer imagen docker 2021"
--> Running in 18312b4a5ed9
Removing intermediate container 18312b4a5ed9
--> 15fec613f250
Successfully built 15fec613f250
Successfully tagged jovaniac/miprimerImagen:0.2
```

## Ejecutar imagenes

The image shows a single terminal window with a blue header bar containing the text 'jovani-dev@jovanidev:~/Documents/certificatic/05\_microservices/lab1\_contenedores/01\_init/primer\_imagen\$'. Below the header is a blue command line area.

```
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_init/primer_imagen$ docker run -it jovaniac/miprimerImagen:0.1
Mi primer imagen docker 2021
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_init/primer_imagen$ docker run -it jovaniac/miprimerImagen:0.2
Mi primer imagen docker otra version
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_init/primer_imagen$
```

# Comandos y latest

**docker build -t landa/comandos-java:latest .**

docker run -it landa/comandos-java:latest

docker run -it landa/comandos-java

docker images

```
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_init/02_comandos_contenedor$ docker
  build -t jovaniac/comandos-java:latest .
  Sending build context to Docker daemon 2.048kB
  Step 1/2 : FROM openjdk:jdk-alpine
  jdk-alpine: Pulling from library/openjdk
  8e3ballec2a2: Pull complete
  311ad0da4533: Pull complete
  df312c74ce16: Pull complete
  Digest: sha256:1fd5a77d82536c88486e526da26ae79b6cd8a14006eb3da3a25eb8d2d682ccd6
  Status: Downloaded newer image for openjdk:jdk-alpine
  ---> 5801f7d008e5
  Step 2/2 : CMD java -version
  ---> Running in e5fc81fac40c
  Removing intermediate container e5fc81fac40c
  ---> f8320c1812dd
  Successfully built f8320c1812dd
  Successfully tagged jovaniac/comandos-java:latest
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_init/02_comandos_contenedor$ docker
  run -it jovaniac/comandos-java:latest
  openjdk version "1.8.0_171"
  OpenJDK Runtime Environment (IcedTea 3.8.0) (Alpine 8.171.11-r0)
  OpenJDK 64-Bit Server VM (build 25.171-b11, mixed mode)
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_init/02_comandos_contenedor$
```

## Tag : latest

| docker images           |            |              |                |        |
|-------------------------|------------|--------------|----------------|--------|
| REPOSITORY              | TAG        | IMAGE ID     | CREATED        | SIZE   |
| jovaniac/comandos-java  | latest     | f8320c1812dd | 6 minutes ago  | 103MB  |
| jovaniac/miprimerImagen | 0.2        | 44e34ac73aa1 | 11 minutes ago | 73.9MB |
| jovaniac/miprimerImagen | 0.1        | 56a37473a6be | 11 minutes ago | 73.9MB |
| ubuntu                  | latest     | 1e4467b07108 | 4 days ago     | 73.9MB |
| openjdk                 | jdk-alpine | 5801f7d008e5 | 2 years ago    | 103MB  |

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/01\\_primeros\\_pasos/02\\_comandos\\_contenedor](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/01_primeros_pasos/02_comandos_contenedor)

# Hands-ON

# Dockerizando Node Server

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/01\\_primeros\\_pasos/03\\_serverjs](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/01_primeros_pasos/03_serverjs)

```
145      </p>
146    </li>
147  </ul>
148 </div>
149  );
150 }
151
152  renderWhatsNewLinks() {
153    return (
154      <div className={styles.container}>
155        <h4>What's new</h4>
156        <ul className={classNames(styles.list, styles.whatsNewList)}>
157          {this.renderWhatsNewItem('New features in React', '/new-features')}
158          {this.renderWhatsNewItem('Improved performance', '/improved-performance')}
159          {this.renderWhatsNewItem('Bug fixes', '/bug-fixes')}
160          {this.renderWhatsNewItem('API updates', '/api-updates')}
161          {this.renderWhatsNewItem('New components', '/new-components')}
162          {this.renderWhatsNewItem('Performance improvements', '/performance-improvements')}
163          {this.renderWhatsNewItem('Bug fixes and security patches', '/bug-fixes-and-security-patches')}
164        </ul>
165      </div>
166    );
167  }
168
169  renderWhatsNewItem(title, url) {
170    return (
171      <a href={url} title={title}>
172        {title}
173      </a>
174    );
175  }
176
177  renderFooterSub() {
178    return (
179      <div className={styles.footerSub}>
180        <Link to="/" title="Home - Display">
181          <Icon type="Logo" className={styles.footerSubLogo}></Icon>
182        </Link>
183        <span className={styles.footerSlogan}>
184          A modern, efficient, and flexible front-end framework for building large-scale web applications.
185        </span>
186      </div>
187    );
188  }
189
190  render() {
191    return (
192      <footer className={styles.footerGlobal}>
193        <div className="container">
194          {this.renderFooterMain()}
195          {this.renderFooterSub()}
196        </div>
197      </footer>
198    );
199  }
200}
```

```
docker build -t server-app:0.1 .
```

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_
File Edit View Search Terminal Help

jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_mi
cios/lab1_contenedores/01_init/serverjs$ docker build -t jovaniac/server-app:0.1 .
Sending build context to Docker daemon 3.072kB
Step 1/5 : FROM registry.access.redhat.com/ubi8/nodejs-10
--> f297fd21f1a3
Step 2/5 : WORKDIR /opt/testjs
--> Using cache
--> ca4864cc2700
Step 3/5 : COPY app.js .
--> Using cache
--> 1f5c14acb19e
Step 4/5 : EXPOSE 5000
--> Using cache
--> 0ad9b475fd7b
Step 5/5 : CMD ["node", "app.js"]
--> Using cache
--> c4ca51dbeb8f
Successfully built c4ca51dbeb8f
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_mi
cios/lab1_contenedores/01_init/serverjs$ █
```

# docker images

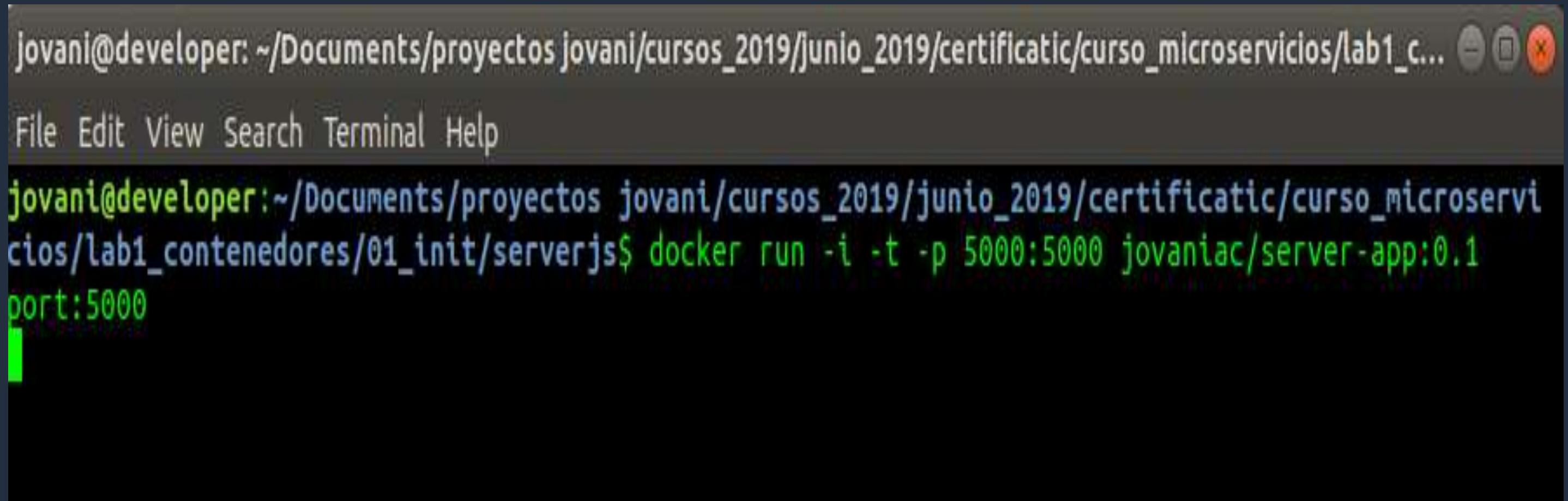
```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_cont  
js$ docker images  
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE  
jovaniac/server-app    0.1      c4ca51dbeb8f  16 minutes ago  720MB  
jovaniac/api-creditos   0.1      4e1e65ee1df5  13 days ago   910MB  
jovaniac/api-empleados    0.1      49345838b0cd  13 days ago   910MB  
jovaniac/api-clientes     0.1      ad0765ef69ca  13 days ago   910MB  
node                  8        a5c31320f223  2 weeks ago   895MB  
registry.access.redhat.com/ubi8/nodejs-10  latest    f297fd21f1a3  4 weeks ago   720MB  
hello-world           latest    fce289e99eb9  4 months ago  1.84kB  
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_cont  
js$ █
```

# docker image ls

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_cont  
js$ docker image ls  
REPOSITORY          TAG      IMAGE ID      CREATED       SIZE  
jovaniac/server-app    0.1      c4ca51dbeb8f  17 minutes ago  720MB  
jovaniac/api-creditos   0.1      4e1e65ee1df5  13 days ago   910MB  
jovaniac/api-empleados    0.1      49345838b0cd  13 days ago   910MB  
jovaniac/api-clientes     0.1      ad0765ef69ca  13 days ago   910MB  
node                  8        a5c31320f223  2 weeks ago   895MB  
registry.access.redhat.com/ubi8/nodejs-10  latest    f297fd21f1a3  4 weeks ago   720MB  
hello-world           latest    fce289e99eb9  4 months ago  1.84kB  
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_cont  
js$ █
```

# Exponiendo contenedor - Puertos

docker run -it -p 5000:5000 server-app:0.1



A screenshot of a terminal window titled "jovani@developer: ~/Documents/proyectos\_jovani/cursos\_2019/junio\_2019/certificatic/curso\_microservicios/lab1\_c...". The window has a standard OS X-style title bar with close, minimize, and maximize buttons. Below the title bar is a menu bar with "File", "Edit", "View", "Search", "Terminal", and "Help". The main terminal area shows the command "jovani@developer:~/Documents/proyectos\_jovani/cursos\_2019/junio\_2019/certificatic/curso\_microservicios/lab1\_contenedores/01\_init/serverjs\$ docker run -i -t -p 5000:5000 jovaniac/server-app:0.1 port:5000" being typed in. The command uses color-coded syntax highlighting for the host port (5000), container port (5000), and image name (server-app).

- El puerto se abre en el contenedor y no esta disponible al host.  
Aislamiento de recursos.
- Es posible acceder a esos puertos mediante el flag `-p`
- Se puede usar múltiples veces



docker ps

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
11e3e4136548        jovaniac/server-app:0.1   "container-entrypoint..."   24 seconds ago    Up 20
seconds           0.0.0.0:5000->5000/tcp, 8080/tcp   eager_lamarr
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$
```

docker stop <id contenedor>

docker stop 11e3e4136548

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_c... ─ ─ ─
File Edit View Search Terminal Help
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$ docker run -i -t -p 5000:5000 jovaniac/server-app:0.1
port:5000
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$ ┌─
```

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_c... ─ ─ ─
File Edit View Search Terminal Help
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$ docker ps
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS
11e3e4136548        jovaniac/server-app:0.1   "container-entrypoint..."   2 minutes ago      Up 2 minutes
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$ docker stop 11e3e4136548
11e3e4136548
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$ ┌─
```

```
docker run -d -p 5000:5000 serverapp:0.1
```

```
docker ps
```

```
jovani@developer: ~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$ docker run -d -p 5000:5000 jovaniac/server-app:0.1  
7ba22ae01199e8f20b6069a5abb2ec8a0374beee48eaf74979d1d7b87f953b1d  
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$ docker ps  
CONTAINER ID        IMAGE               COMMAND             CREATED            STATUS              PORTS  
7ba22ae01199        jovaniac/server-app:0.1   "container-entrypoin..."  
nutes              0.0.0.0:5000->5000/tcp, 8080/tcp   hungry_hodgkin  
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/01_init/serverjs$
```



```
docker run it -d -p 5000:5000 landa/serverapp:0.1
```

Modo detach

```
Docker run -it -d -p 6000:5000 landa/serverapp:0.1
```



# Hands-ON API Node Docker

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/01\\_primeros\\_pasos/03\\_serverjs/mock](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/01_primeros_pasos/03_serverjs/mock)

```
145      </p>
146    </li>
147  </ul>
148 </div>
149  );
150 }
151
152 ▼ renderWhatsNewLinks() {
153  return (
154    <div className={styles.container}>
155      <h4 className={style}>
156        <ul className={cls}>
157          {this.renderHeader()}
158          {this.renderHeader()}
159          {this.renderHeader()}
160          {this.renderHeader()}
161          {this.renderHeader()}
162          {this.renderHeader()}
163          {this.renderHeader()}
164          {this.renderHeader()}
165        </ul>
166      </div>
167    );
168  }
169
170 ▼ renderWhatsNewItem(title, url) {
171  return (
172    <li className={styles.footer}>
173      <a href={trackUrl(url)}>
174        target="_blank"
175        rel="noopener noreferrer"
176        >
177          {title}
178        </a>
179      </li>
180    );
181  }
182 }
183
184 ▼ renderFooterSub() {
185  return (
186    <div className={styles.footerSub}>
187      <Link to="/" title="Home - Unsplash">
188        <Icon type="Logo" className={styles.footerSubLogo}>
189      </Link>
190      <span className={styles.footerSlogan}>
191        </span>
192    </div>
193  );
194 }
195
196 }
197
198 ▼ render() {
199  return (
200    <footer className={styles.footerGlobal}>
201      <div className="container">
202        {this.renderFooterMain()}
203        {this.renderFooterSub()}
204      </div>
205    </footer>
206  );
207}
```

```
docker build -t landa/apimock:0.1 .
```

```
docker images
```

```
docker run -p 8000:8000 landa/apimock:0.1
```

```
s_pasos/03_serverjs/mock$ docker run -p 8000:8000 jovaniac/apimock:0.1
Fri, 11 Dec 2020 07:28:22 GMT express deprecated app.del: Use app.delete instead at mock.js:21:7
¡Hola Mundo!
¡A beber cerveza!
¡Método post!
¡Método delete!
```



# Hands-ON Scripts en contenedores

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/01\\_primeros\\_pasos/04\\_add-script](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/01_primeros_pasos/04_add-script)

```
145      </p>
146    </li>
147  </ul>
148 </div>
149  );
150 }
151
152 ▼ renderWhatsNewLinks() {
153   return (
154     <div className={styles.container}>
155       <h4>What's new</h4>
156       <ul className={classNames(styles.list, styles.whatsNewList)}>
157         {this.renderWhatsNewItem('New features', '/#/new')}
158         {this.renderWhatsNewItem('Bug fixes', '/#/bugfixes')}
159         {this.renderWhatsNewItem('Performance improvements', '/#/perfimprovements')}
160         {this.renderWhatsNewItem('Documentation updates', '/#/docs')}
161         {this.renderWhatsNewItem('API changes', '/#/api')}
162         {this.renderWhatsNewItem('Deployment notes', '/#/deploynotes')}
163         {this.renderWhatsNewItem('Known issues', '/#/knownissues')}
164       </ul>
165     </div>
166   );
167 }
168
169
170 ▼ renderWhatsNewItem(title, url) {
171   return (
172     <a href={url} title={title}>
173       <img alt="Icon" />
174       {title}
175     </a>
176   );
177 }
178
179
180 ▼ renderFooterSub() {
181   return (
182     <div className={styles.footerSub}>
183       <Link to="/" title="Home - Display">
184         <Icon type="Logo" />
185         <span>Display</span>
186       </Link>
187       <Link to="/about" title="About Us">
188         <Icon type="Info" />
189         <span>About Us</span>
190       </Link>
191     </div>
192   );
193 }
194
195
196 ▼ render() {
197   return (
198     <div>
199       <header>
200         <div>
201           <img alt="Logo" />
202           <h1>Display</h1>
203         </div>
204         <nav>
205           <ul>
206             {this.renderHeaderLinks()}
207           </ul>
208         </nav>
209       </header>
210       <main>
211         {children}
212       </main>
213       <Footer>
214         <div>
215           <h4>What's new</h4>
216           {this.renderWhatsNewLinks()}
217         </div>
218         <div>
219           <h4>Footer Sub</h4>
220           {this.renderFooterSub()}
221         </div>
222       </Footer>
223     </div>
224   );
225 }
```

# Script en Containers

```
docker build -t landa/script-app:0.1 .
```

```
Sending build context to Docker daemon  
3.072kB  
Step 1/4 : FROM alpine:3.6  
---> 43773d1dba76  
Step 2/4 : ADD scrip.sh /scrip.sh  
---> d1492b74d387  
Step 3/4 : RUN sh -c 'chmod +x /scrip.sh'  
---> Running in 9d0e9d488857  
Removing intermediate container 9d0e9d488857  
---> c7ab43f9fa2f  
Step 4/4 : ENTRYPOINT ["sh","/scrip.sh"]  
---> Running in 6fbf745c4105  
Removing intermediate container 6fbf745c4105  
---> 640449588b65  
Successfully built 640449588b65  
Successfully tagged landa/script-app:0.1
```

```
docker build -t landa/script-app:0.1  
docker run landa/script-app:0.1
```

-it is short for --interactive + --tty

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificati  
c/curso_microservicios/lab1_contenedores/02_parametros/scripting$ docker build -  
t jovaniac/script-app:0.1 .  
Sending build context to Docker daemon 3.072kB  
Step 1/4 : FROM alpine:3.6  
---> 43773d1dba76  
Step 2/4 : ADD scrip.sh /scrip.sh  
---> d1492b74d387  
Step 3/4 : RUN sh -c 'chmod +x /scrip.sh'  
---> Running in 9d0e9d488857  
Removing intermediate container 9d0e9d488857  
---> c7ab43f9fa2f  
Step 4/4 : ENTRYPOINT ["sh","/scrip.sh"]  
---> Running in 6fbf745c4105  
Removing intermediate container 6fbf745c4105  
---> 640449588b65  
Successfully built 640449588b65  
Successfully tagged jovaniac/script-app:0.1  
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificati  
c/curso_microservicios/lab1_contenedores/02_parametros/scripting$
```

# Hands-ON

## Eliminar imágenes y contenedores / Push image – DockerHub

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/01\\_primeros\\_pasos/06\\_push\\_registry](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/01_primeros_pasos/06_push_registry)

# Eliminar todo

## Lab 03\_serverjs

```
docker build -t landa/serverapp:0.1 .
```

### Eliminar contenedores

```
docker container rm -f $(docker container ls -aq)
```

### Eliminar imágenes

```
docker image rm -f $(docker image ls -qa)
```

# -f = forzar

### ./eliminar.sh



# Subir imágenes a registry

## Lab 03\_serverjs construir imagen

- . Subir la imagen sin espacio de nombres y muestra el error:

Iniciar sesión en Docker:

```
docker login
```

```
docker push landa/serverjs:0.1
```

- . Ejecutar registro:

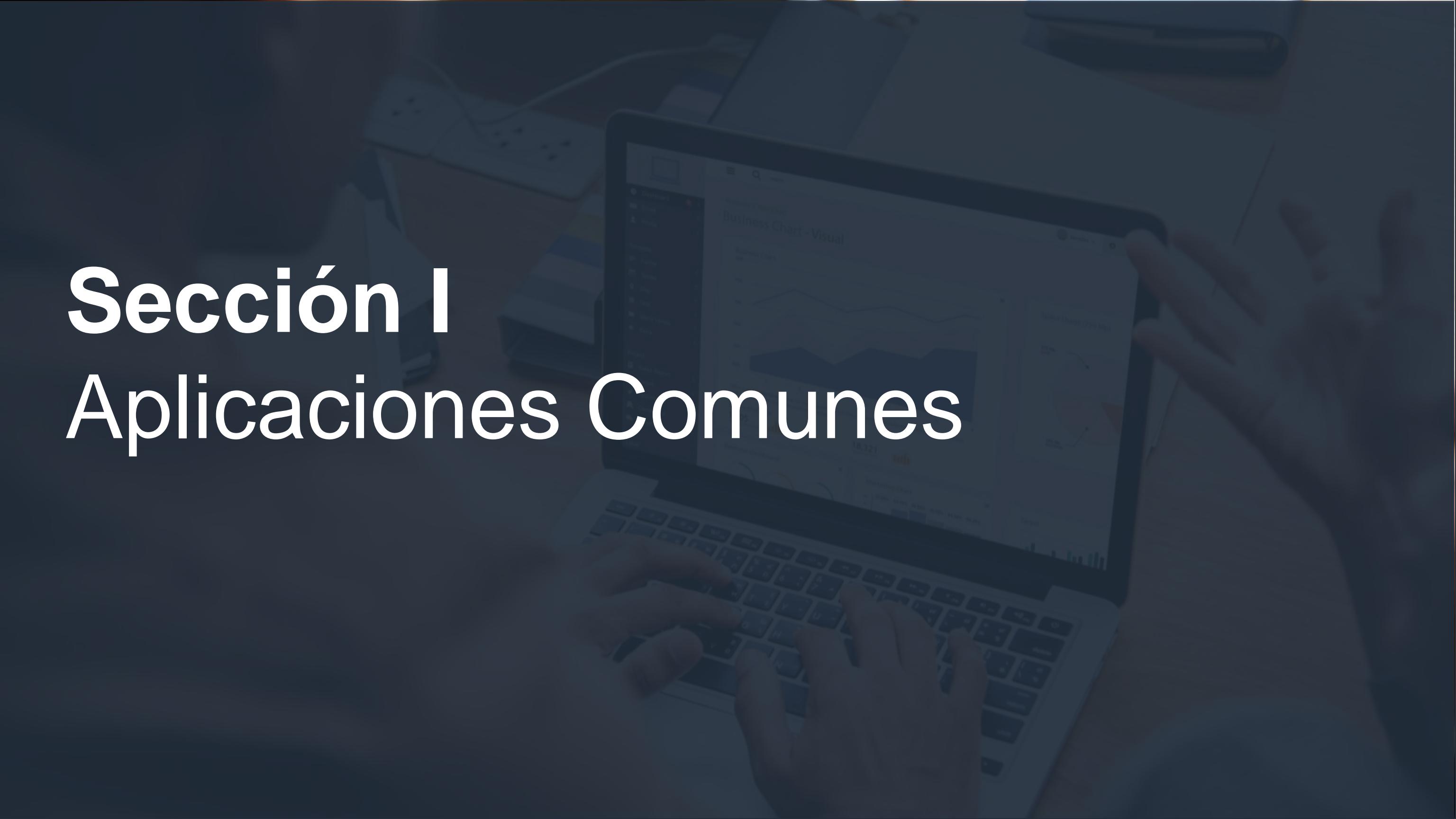
```
docker run -d -p 5000: 5000 --restart always landa/serverjs:0.1
```

```
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_primeros_pasos/03_serverjs$ docker build -t jovaniac/serverjs:0.1 .
Sending build context to Docker daemon 3.072kB
Step 1/6 : FROM registry.access.redhat.com/ubi8/nodejs-10
--> 0c2506a371ca
Step 2/6 : WORKDIR /opt/testjs
--> Using cache
--> 01c0dab131ec
Step 3/6 : COPY app.js .
--> Using cache
--> 821c5a91dba5
Step 4/6 : EXPOSE 5000
--> Using cache
--> 5a259efaaeb1
Step 5/6 : ENV MESSAGE Docker Containers "ARQUITECTO DE APIs Y MICROSERVICIOS 4 GENERACIÓN 2020"
--> Using cache
--> 82c1359b536a
Step 6/6 : CMD ["node", "app.js"]
--> Using cache
--> 06122d4096ee
Successfully built 06122d4096ee
Successfully tagged jovaniac/serverjs:0.1
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_primeros_pasos/03_serverjs$
```

```
jovani-dev@jovanidev:~/Documents/certificatic/05_microservices/lab1_contenedores/01_primeros_pasos/03_serverjs$ docker
push jovaniac/serverjs:0.1
The push refers to repository [docker.io/jovaniac/serverjs]
8907fe89dac9: Pushed
00156cf65b06: Pushed
dda614054ef4: Pushing [>] 2.202MB/142.4MB
7fbe9fcfa2da: Pushing [>] 3.29MB/363.4MB
0bfe5b62a1ad: Pushing [=====>] 5.427MB/54.44MB
70056249a0e2: Pushed
226bfaae015f: Pushing [>] 3.269MB/203.4MB
```

# Sección I

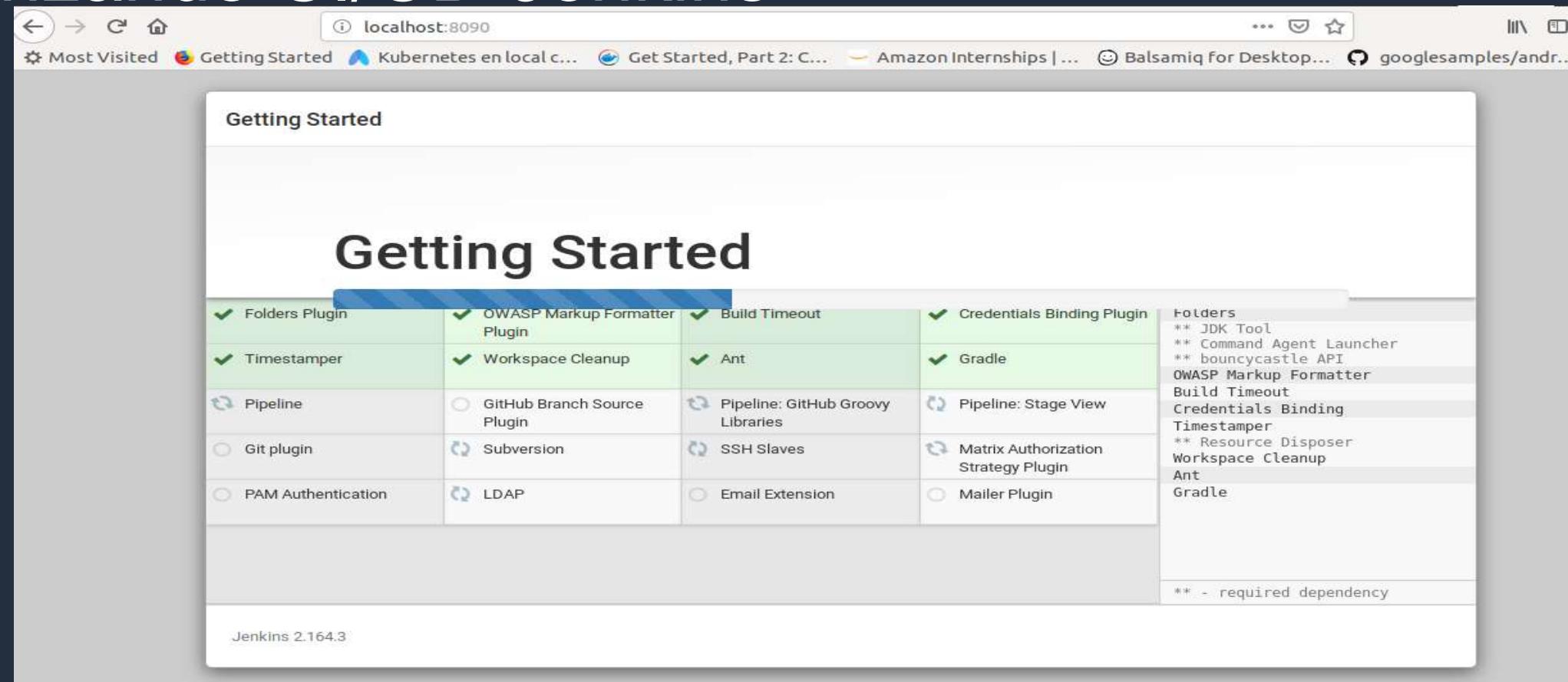
# Aplicaciones Comunes



# Hands-ON Deploy de aplicaciones comunes con Docker Jenkins

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/03\\_aplicaciones/ci-cd/jenkins/jenkins.md](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/03_aplicaciones/ci-cd/jenkins/jenkins.md)

# Contenerizando CI/CD Jenkins



```
docker run --rm -u root -p 8090:8080 -v "$PWD/data-jenkins/":/var/jenkins_home -v /var/run/docker.sock:/var/run/docker.sock jenkinsci/blueocean
```

contenedor con un volúmen, asociándolo a la carpeta  
“nombreVolumen:/var/jenkins\_home”

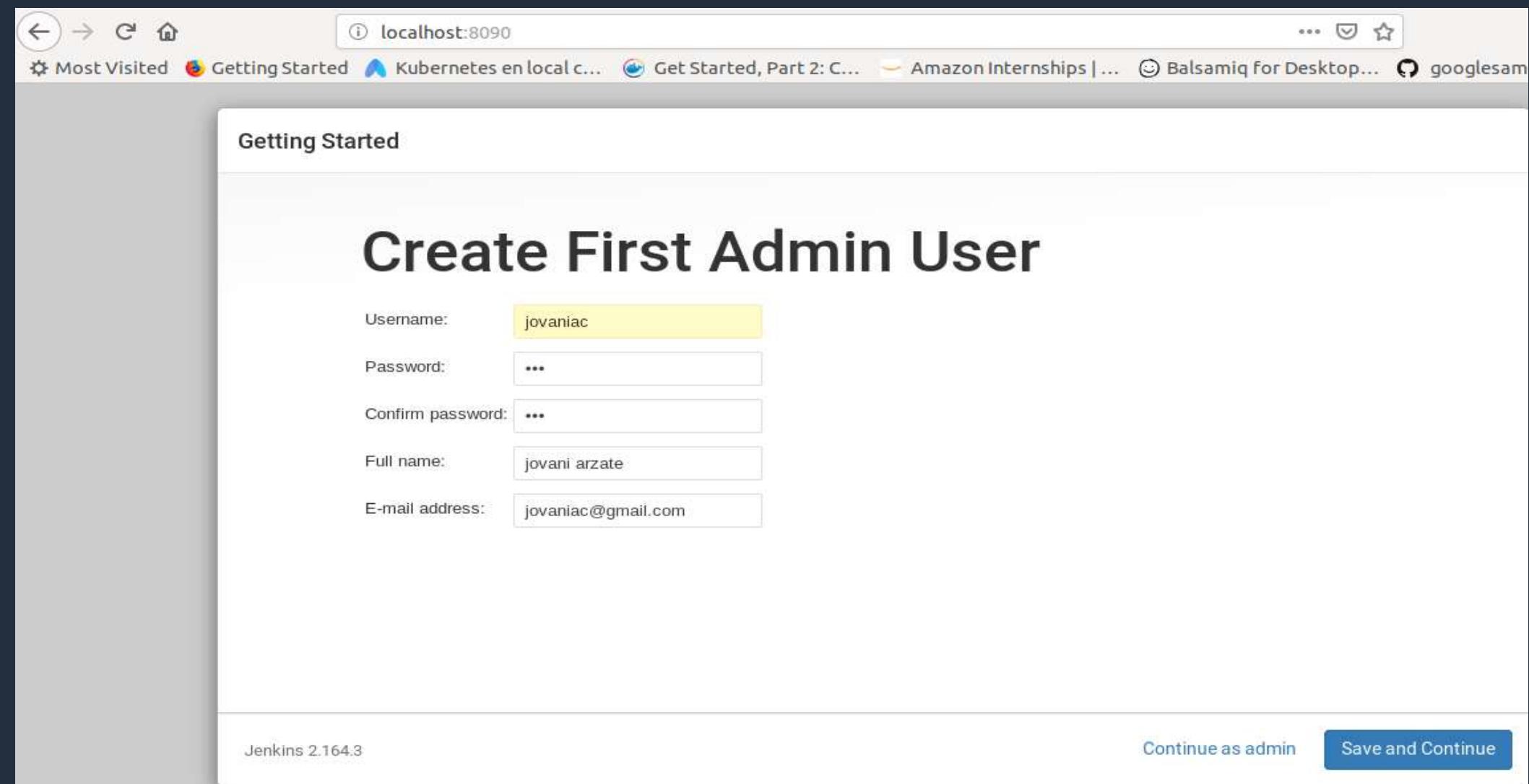
```
docker exec -it <id contenedor> /bin/bash
```



# CI/CD Jenkins

**/var/run/docker.sock:/var/run/docker.sock**

Es el socket Unix que el demonio Docker escucha de manera predeterminada, y se puede usar para comunicarse con el demonio desde un contenedor.



# CI/CD Jenkins

The screenshot shows a web browser window with the address bar displaying 'localhost:8090'. The main content is titled 'Getting Started' and 'Instance Configuration'. It features a 'Jenkins URL:' input field containing 'http://localhost:8090/'. Below the input field is explanatory text about the Jenkins URL's purpose and best practices for setting it. At the bottom, there are navigation buttons for 'Not now' and 'Save and Finish'.

localhost:8090

Getting Started

# Instance Configuration

Jenkins URL:

The Jenkins URL is used to provide the root URL for absolute links to various Jenkins resources. That means this value is required for proper operation of many Jenkins features including email notifications, PR status updates, and the BUILD\_URL environment variable provided to build steps.

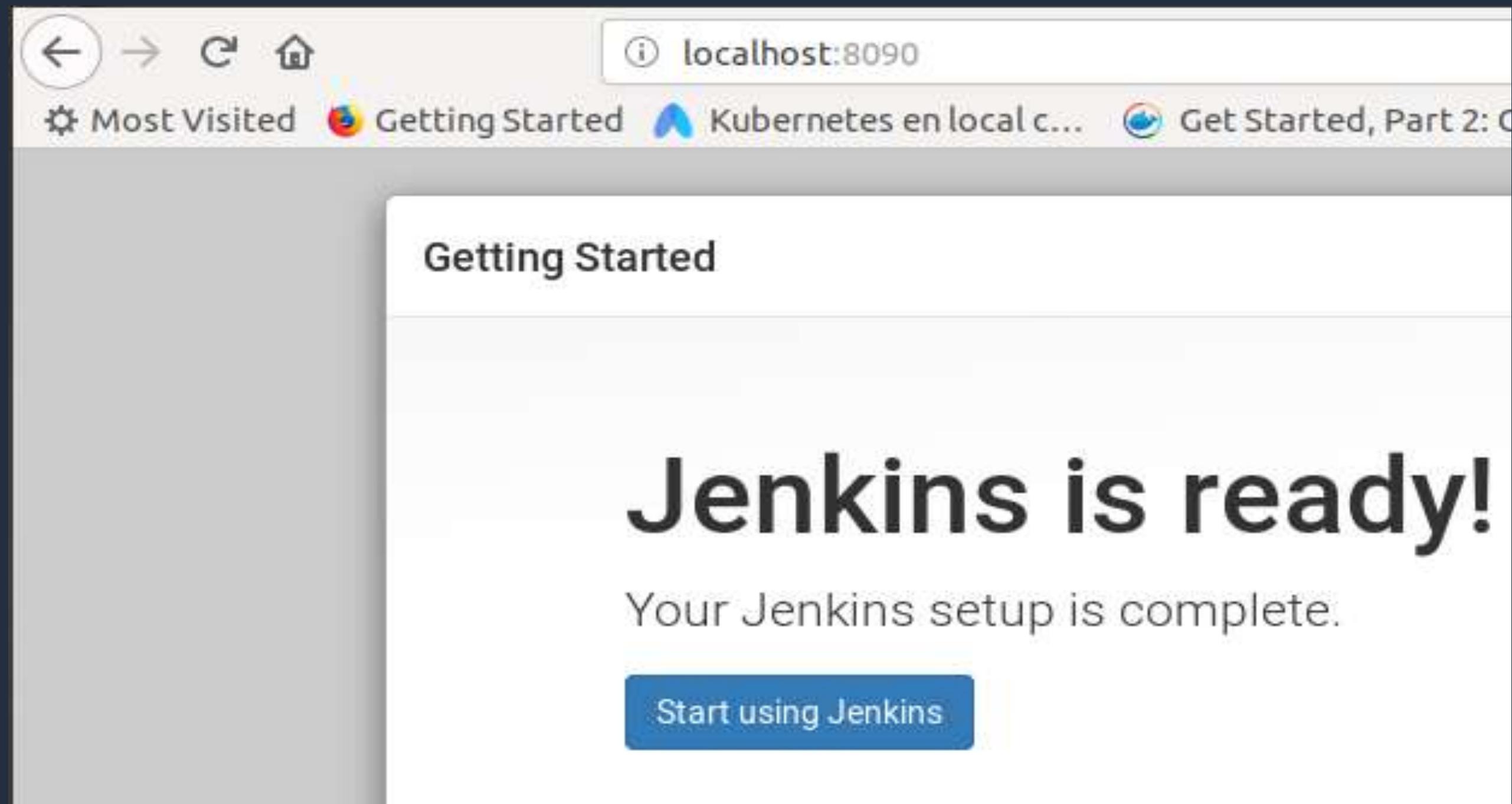
The proposed default value shown is **not saved yet** and is generated from the current request, if possible. The best practice is to set this value to the URL that users are expected to use. This will avoid confusion when sharing or viewing links.

Jenkins 2.164.3

Not now

Save and Finish

# CI/CD Jenkins



# CI/CD Jenkins

localhost:8090/view/all/newJob

Most Visited Getting Started Kubernetes en local c... Get Started, Part 2: C... Amazon Internships | ... Balsamiq fo...

## Jenkins

Jenkins All

### Enter an item name

microservicio-api-clientes » Required field

**Freestyle project**  
This is the central feature of Jenkins. Jenkins will build your project, combining any SCM with any...

**Pipeline**  
Orchestrates long-running activities that can span multiple build agents. Suitable for building pipeline...

**Multi-configuration project**  
Suitable for projects that need a large number of different configurations, such as testing on multip...

**Bitbucket Team/Project**  
Scans a Bitbucket Cloud Team (or Bitbucket Server Project) for all repositories matching some defin...

**Folder**  
Creates a container that stores nested items in it. Useful for grouping things together. Unlike view, name as long as they are in different folders.

**GitHub Organization**  
Scans a GitHub organization (or user account) for all repositories matching some defined markers.

**Multibranch Pipeline**  
Creates a set of Pipeline projects according to detected branches in one SCM repository.

If you want to create a new item from other existing, you can use this option:

 Copy from

**OK**

# CI/CD Jenkins

The screenshot shows the Jenkins Pipeline configuration page for a job named "microservicio-api-clientes". The "Advanced Project Options" tab is selected. In the "Pipeline" section, under the "Definition" tab, the "Pipeline script from SCM" option is chosen. The "Git" tab is selected under "SCM". The "Repositories" section contains one repository configuration:

- Repository URL:** `https://github.com/jovaniac/curso_microservicios_ci-cd.git`
- Credentials:** A dropdown menu set to "- none".
- Branch Specifier (blank for 'any'):** `*/master`
- Repository browser:** `(Auto)`
- Additional Behaviours:** A dropdown menu set to "Add".
- Script Path:** `Jenkinsfile`
- Lightweight checkout:**

At the bottom left, there are "Save" and "Apply" buttons.

The screenshot shows the Jenkins Credentials Provider window titled "Jenkins Credentials Provider: Jenkins". A new credential is being added:

- Domain:** Global credentials (unrestricted)
- Kind:** Username with password
- Scope:** Global (Jenkins, nodes, items, all child items, etc)
- Username:** `curso_arquitecto_febrero-1873039502730`
- Password:** `XXXXXXXXXXXXXX`
- ID:**
- Description:**

At the bottom are "Add" and "Cancel" buttons.

Configuración de credenciales de repositorio codecommit de AWS

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/arquitecto\\_apis\\_microservicios\\_ci-cd](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/arquitecto_apis_microservicios_ci-cd)

# CI/CD Jenkins

Build now



Jenkins

ms-clientes

## Pipeline ms-clientes



Recent Changes

### Stage View

No data available. This Pipeline has not yet run.

### Permalinks

Build History

trend =

find

#1 Feb 18, 2020 7:39 AM

Atom feed for all Atom feed for failures

localhost:8090/job/microservicio-api-clientes/1/console

```
> git config --add remote.origin.fetch+refs/heads/*:refs/remotes/origin/* # timeout=10
> git config remote.origin.url https://github.com/jovaniac/curso_microservicios_ci_cd.git # timeout=10
Fetching upstream changes from https://github.com/jovaniac/curso_microservicios_ci_cd.git
> git fetch --tags --progress https://github.com/jovaniac/curso_microservicios_ci_cd.git +refs/heads/*:refs/remotes/origin/*
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
> git rev-parse refs/remotes/origin/master^{commit} # timeout=10
Checking out Revision cae55983269daec4cb95e69fce2f91a4fde59894 (refs/remotes/origin/master)
> git config core.sparsecheckout # timeout=10
> git checkout -f cae55983269daec4cb95e69fce2f91a4fde59894
Commit message: "removiendo readme"
First time build. Skipping changelog.
[Pipeline]
[Pipeline] // stage
[Pipeline] withEnv
[Pipeline] {
[Pipeline] stage
[Pipeline] { (Build)
[Pipeline] echo
Building..
[Pipeline] script
[Pipeline] {
[Pipeline] sh
+ ./gradlew clean buildImage --no-daemon
To honour the JVM settings for this build a new JVM will be forked. Please consider using the daemon: https://docs.gradle.org/5.0/userguide/gradle\_daemon.html.
Daemon will be stopped at the end of the build stopping after processing
> Task :clean UP-TO-DATE
> Task :compileJava
> Task :processResources
> Task :classes
> Task :findMainClass
> Task :jar
> Task :bootRepackage
> Task :createDockerfile
> Task :buildImage
Building image using context '/var/jenkins_home/workspace/microservicio-api-clientes/build/libs'.
Using tag 'jovaniac/servicio-cliente:0.0.1-snapshot' for image.
Step 1/5 : FROM openjdk:8u151-jre-slim
--> 1c01b1ac4ced
Step 2/5 : MAINTAINER jovaniac@gmail.com"
--> Using cache
--> 8ac9bb41da60
Step 3/5 : COPY servicio-cliente.jar /opt/servicio-cliente.jar
```



# CI/CD Jenkins

Ejecutar para validar la construcción de imagen  
docker images  
watch docker images

REPOSITORY

landa/servicio-cliente

TAG

0.0.1-snapshot

IMAGE ID

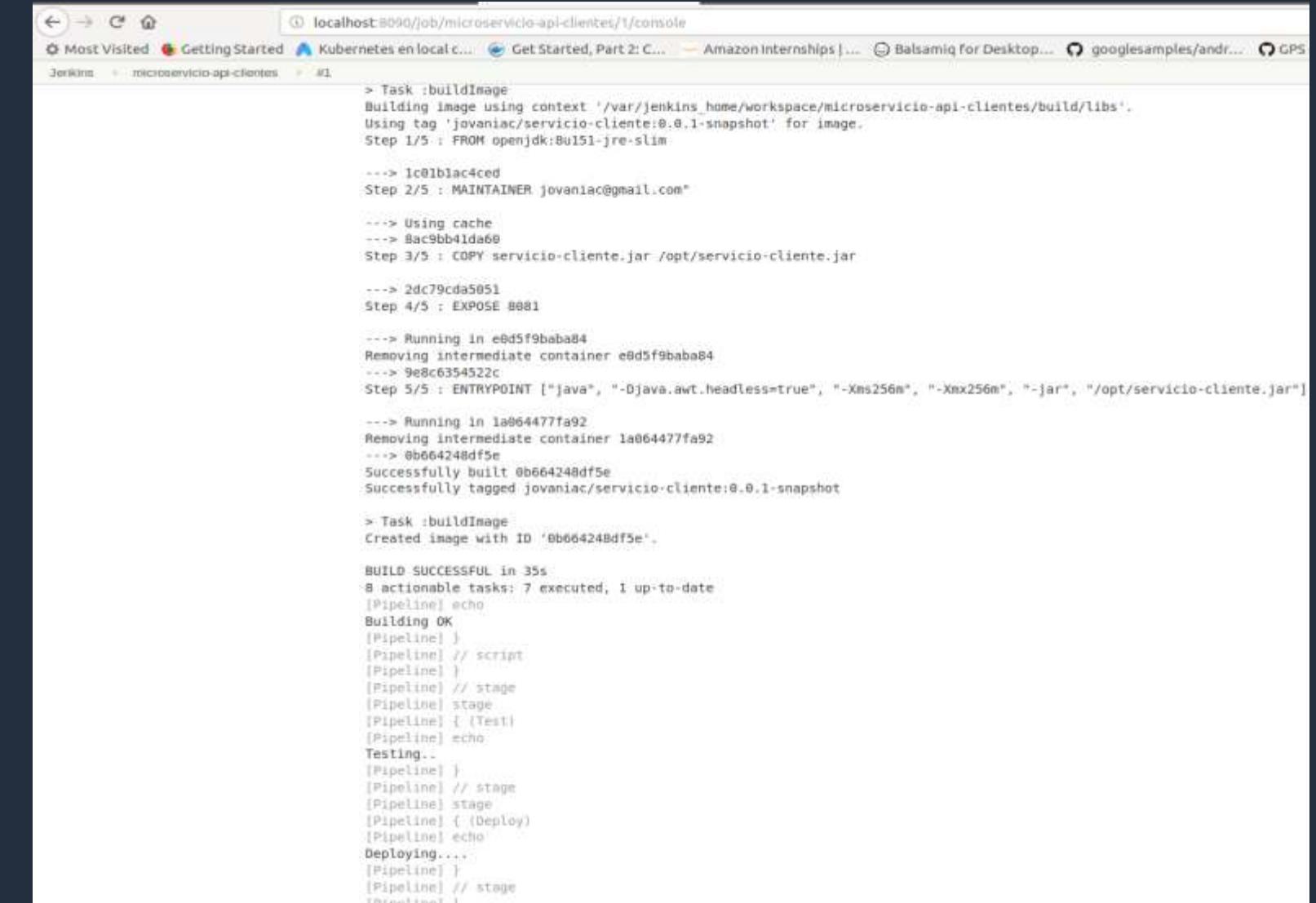
0b664248df5e

CREATED

About a minute ago

SIZE

246MB



The screenshot shows a Jenkins job console for a job named 'microservicio-api-clientes'. The log output is as follows:

```
> Task :buildImage
Building image using context '/var/jenkins_home/workspace/microservicio-api-clientes/build/libs'.
Using tag 'jovaniac/servicio-cliente:0.0.1-snapshot' for image.
Step 1/5 : FROM openjdk:8u151-jre-slim
--> 1c01bbac4ced
Step 2/5 : MAINTAINER jovaniac@gmail.com
--> Using cache
--> 8ac9bb41da68
Step 3/5 : COPY servicio-cliente.jar /opt/servicio-cliente.jar
--> 2dc79cda5051
Step 4/5 : EXPOSE 8081
--> Running in e0d5f9babaa84
Removing intermediate container e0d5f9babaa84
--> 9e8c6354522c
Step 5/5 : ENTRYPOINT ["java", "-Djava.awt.headless=true", "-Xms256m", "-Xmx256m", "-jar", "/opt/servicio-cliente.jar"]
--> Running in 1a064477fa92
Removing intermediate container 1a064477fa92
--> 0b664248df5e
Successfully built 0b664248df5e
Successfully tagged jovaniac/servicio-cliente:0.0.1-snapshot

> Task :buildImage
Created image with ID '0b664248df5e'.

BUILD SUCCESSFUL in 35s
8 actionable tasks: 7 executed, 1 up-to-date
[Pipeline] echo
Building OK
[Pipeline] }
[Pipeline] // script
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Test)
[Pipeline] echo
Testing...
[Pipeline] }
[Pipeline] // stage
[Pipeline] stage
[Pipeline] { (Deploy)
[Pipeline] echo
Deploying...
[Pipeline] }
[Pipeline] // stage
[Pipeline] }
```



# CI/CD Jenkins

File Edit View Search Terminal Help

Every 2.0s: docker images

| REPOSITORY                                | TAG            | IMAGE ID     | CREATED           | SIZE   |
|---|----------------|--------------|-------------------|--------|
| jovaniac/servicio-cliente                 | 0.0.1-snapshot | 0b664248df5e | 2 minutes ago     | 246MB  |
| <none>                                    | <none>         | fce869887d5e | 16 minutes ago    | 246MB  |
| jovaniac/script-app                       | 0.1            | 640449588b65 | About an hour ago | 4.03MB |
| <none>                                    | <none>         | 00a32ee17b19 | 2 hours ago       | 4.03MB |
| <none>                                    | <none>         | 19fb1170d17a | 2 hours ago       | 4.03MB |
| <none>                                    | <none>         | f1040d89e2c4 | 2 hours ago       | 4.03MB |
| jenkinsci/blueocean                       | latest         | 226e73d02a72 | 21 hours ago      | 550MB  |
| jovaniac/server-app                       | 0.1            | c4ca51dbeb8f | 25 hours ago      | 720MB  |
| jovaniac/api-creditos                     | 0.1            | 4e1e65ee1df5 | 2 weeks ago       | 910MB  |
| jovaniac/api-empleados                    | 0.1            | 49345838b0cd | 2 weeks ago       | 910MB  |
| jovaniac/api-clientes                     | 0.1            | ad0765ef69ca | 2 weeks ago       | 910MB  |
| node                                      | 10-alpine      | 56bc3a1ed035 | 2 weeks ago       | 71MB   |
| mysql                                     | latest         | 990386cbd5c0 | 2 weeks ago       | 443MB  |
| node                                      | 8              | a5c31320f223 | 3 weeks ago       | 895MB  |
| registry.access.redhat.com/ubi8/nodejs-10 | latest         | f297fd21f1a3 | 4 weeks ago       | 720MB  |
| alpine                                    | 3.6            | 43773d1dba76 | 2 months ago      | 4.03MB |
| hello-world                               | latest         | fce289e99eb9 | 4 months ago      | 1.84kB |
| openjdk                                   | 8u151-jre-slim | 1c01b1ac4ced | 14 months ago     | 205MB  |

| REPOSITORY             | TAG            | IMAGE ID     | CREATED            | SIZE  |
|------------------------|----------------|--------------|--------------------|-------|
| landa/servicio-cliente | 0.0.1-snapshot | 0b664248df5e | About a minute ago | 246MB |

# CI/CD Jenkins

Pipeline exitoso

localhost:8090/job/microservicio-api-clientes/

Most Visited Getting Started Kubernetes en local c... Get Started, Part 2: C... Amazon Internships | ... Balsamiq for Desktop...

## Jenkins

microservicio-api-clientes

- Back to Dashboard
- Status
- Changes
- Build Now
- Delete Pipeline
- Configure
- Full Stage View
- Open Blue Ocean
- Rename
- Pipeline Syntax

### Pipeline microservicio-api-clientes

Recent Changes

### Stage View

| Declarative: Checkout SCM | Build | Test  | Deploy |
|---------------------------|-------|-------|--------|
| 1s                        | 36s   | 366ms | 367ms  |
| 1s                        | 36s   | 366ms | 367ms  |

Average stage times:  
(Average full run time: ~42s)

#1 May 29, 2019 7:18 AM No Changes

### Build History

trend =

| find                    |
|-------------------------|
| #1 May 29, 2019 7:18 AM |

RSS for all RSS for failures

### Permalinks

- Last build (#1), 2 min 6 sec ago
- Last stable build (#1), 2 min 6 sec ago
- Last successful build (#1), 2 min 6 sec ago
- Last completed build (#1), 2 min 6 sec ago

# CI/CD Jenkins

- 1.- java -jar nombre-microservicio.jar
- 2.- docker run -it -p 8081:8081 nombre-imagen:tag
- 3.- Mostrar actuator localhost:8081/health
- 4.- Pruebas con postman

Postman:  
Contenedores/01\_API Rest Clientes post 201  
Contenedores/02\_API Rest Clientes get 200  
Contenedores/03\_API Rest Clientes delete 200

The screenshot shows a terminal window at the top with log output from a Spring Boot application. Below it is the Postman interface. In the left sidebar, there's a collection named 'contenedores' containing three items: '01\_API Rest Clientes post 201', '02\_API Rest Clientes get 200', and '03\_API Rest Clientes delete 200'. The main workspace shows a POST request to 'localhost:8081/api/v1/clients'. The 'Body' tab displays a JSON payload:

```
1  {
2     "folioCliente": "342177092",
3     "nombre": "jovani",
4     "apellidoPaterno": "arzate",
5     "apellidoMaterno": "cabrera",
6     "email": "jovaniac@gmail.com",
7     "direccion": "test",
8     "genero": "h",
9     "edad": 29
10 }
```

# Hands-ON Docker-compose - Gitlab

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/03\\_aplicaciones/ci-cd/gitlab](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/03_aplicaciones/ci-cd/gitlab)

# Docker Compose - Contenedores

Nos permite poder correr diferentes contenedores que se pueden comunicar entre si y a su vez nos ofrece una configuración simple para poder tener un ambiente de una forma rápida.

Con compose, utiliza un archivo YAML para configurar los servicios de su aplicación. Luego, con un solo comando, crea e inicia todos los servicios desde su configuración



# GitLab Docker - Compose

docker-compose up --build

- 1.- Crear grupos
- 2.- Crear repositorios
- 3.- Clonar repositorio
- 4.- agregar archivos
- 5.- push a repositorio

Borrar imágenes forzado  
docker images –f <id\_container>

localhost:10080

curso-apigateway-apigee

Auto DevOps (Beta)  
It will automatically build, test, and deploy your application based on a predefined CI/CD configuration.  
Learn more in the Auto DevOps documentation.

| Name        | Last commit           | Last update            |
|-------------|-----------------------|------------------------|
| jenkinsFile | mi propio versionador | less than a minute ago |

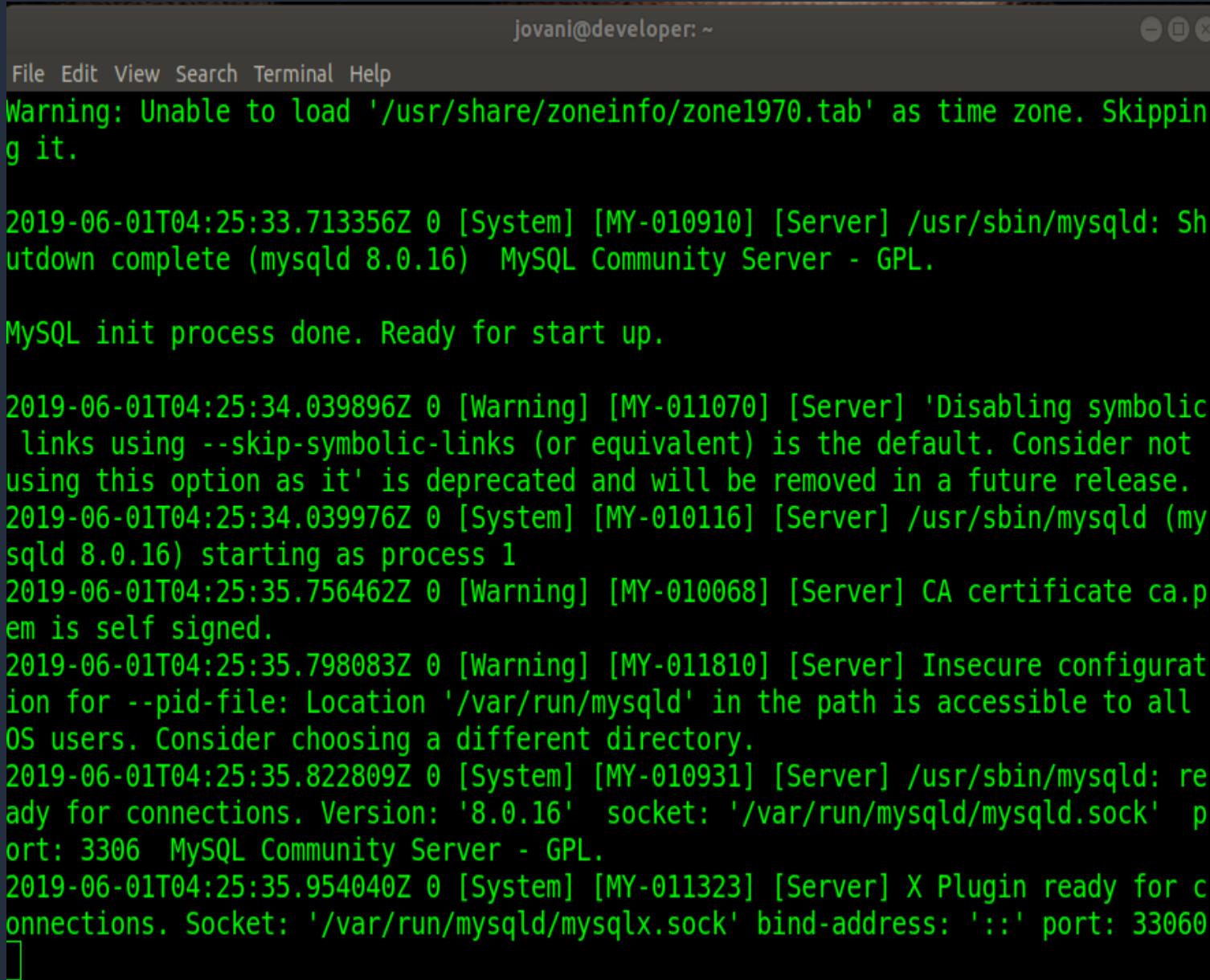
# Hands-ON Docker Mysql

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab1\\_contenedores/03\\_aplicaciones/databases/mysql.md](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab1_contenedores/03_aplicaciones/databases/mysql.md)

## Application

### Mysql

```
docker run -d -p 33060:3306 --name mysql-db -e MYSQL_ROOT_PASSWORD=secret -e MYSQL_USER=root mysql
```



jovani@developer: ~

File Edit View Search Terminal Help

Warning: Unable to load '/usr/share/zoneinfo/zone1970.tab' as time zone. Skipping it.

2019-06-01T04:25:33.713356Z 0 [System] [MY-010910] [Server] /usr/sbin/mysqld: Shutdown complete (mysqld 8.0.16) MySQL Community Server - GPL.

MySQL init process done. Ready for start up.

2019-06-01T04:25:34.039896Z 0 [Warning] [MY-011070] [Server] 'Disabling symbolic links using --skip-symbolic-links (or equivalent) is the default. Consider not using this option as it' is deprecated and will be removed in a future release.

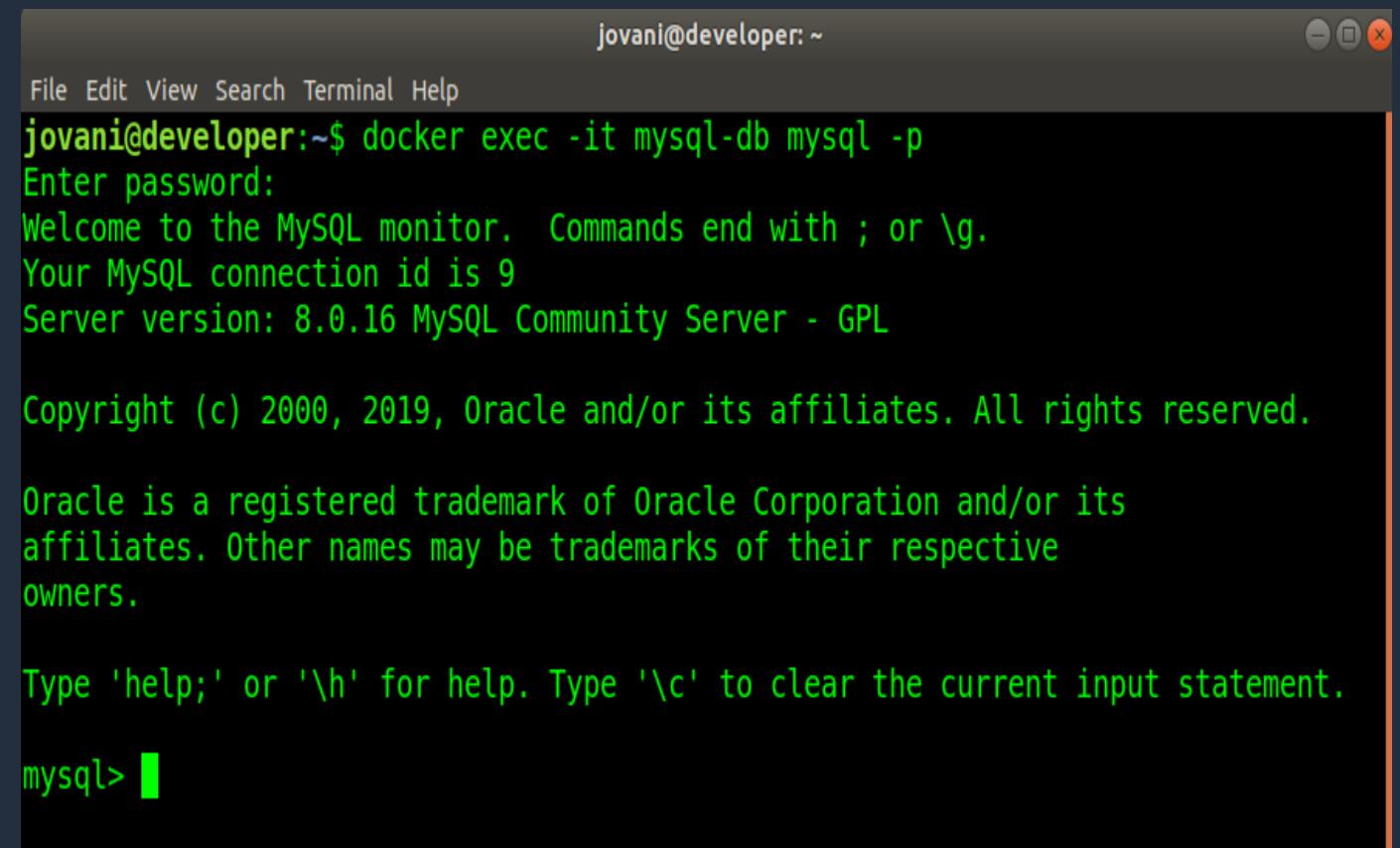
2019-06-01T04:25:34.039976Z 0 [System] [MY-010116] [Server] /usr/sbin/mysqld (mysqld 8.0.16) starting as process 1

2019-06-01T04:25:35.756462Z 0 [Warning] [MY-010068] [Server] CA certificate ca.pem is self signed.

2019-06-01T04:25:35.798083Z 0 [Warning] [MY-011810] [Server] Insecure configuration for --pid-file: Location '/var/run/mysqld' in the path is accessible to all OS users. Consider choosing a different directory.

2019-06-01T04:25:35.822809Z 0 [System] [MY-010931] [Server] /usr/sbin/mysqld: ready for connections. Version: '8.0.16' socket: '/var/run/mysqld/mysqld.sock' port: 3306 MySQL Community Server - GPL.

2019-06-01T04:25:35.954040Z 0 [System] [MY-011323] [Server] X Plugin ready for connections. Socket: '/var/run/mysqlx/mysqlx.sock' bind-address: '::' port: 33060



jovani@developer: ~

File Edit View Search Terminal Help

jovani@developer:~\$ docker exec -it mysql-db mysql -p

Enter password:

Welcome to the MySQL monitor. Commands end with ; or \g.

Your MySQL connection id is 9

Server version: 8.0.16 MySQL Community Server - GPL

Copyright (c) 2000, 2019, Oracle and/or its affiliates. All rights reserved.

Oracle is a registered trademark of Oracle Corporation and/or its affiliates. Other names may be trademarks of their respective owners.

Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

mysql> |

Entramos al contenedor para ir al cliente  
docker exec -it mysql-db mysql -p



Application  
Mysql

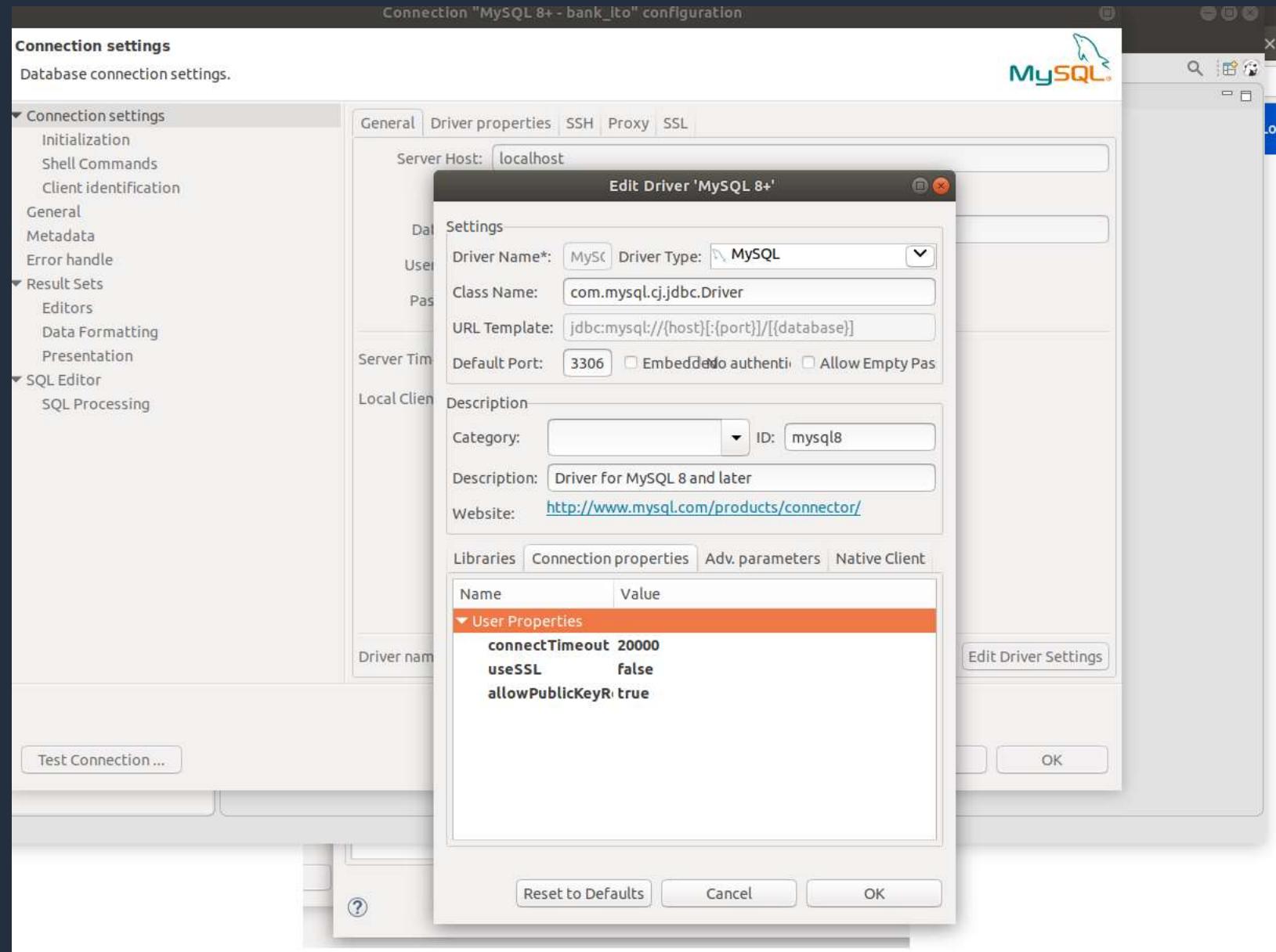
**show databases;**

**create database bank\_ito;**

**use bank-ito;**

**show tables;**

**Usardbeaver para conectarse**



Entramos al contenedor para ir al cliente  
docker exec –it mysql-db mysql -p

# Application Mysql

## Test de conexion Insertar datos

DBeaver 6.3.3 - clientes

File Edit Navigate Search SQL Editor Database Window Help

Database Navigator Projects

MySQL 8+ - bank\_it0 MySQL 8+ - bank\_it0 bank\_it0

Enter a part of table name here

MySQL 8+ - bank\_it0

Databases

bank\_it0

Tables

clientes

Views

Indexes

Procedures

Triggers

Events

performance\_schema

sys

Users

Administer

System info

Project-General

Name: DataSource

Bookmarks

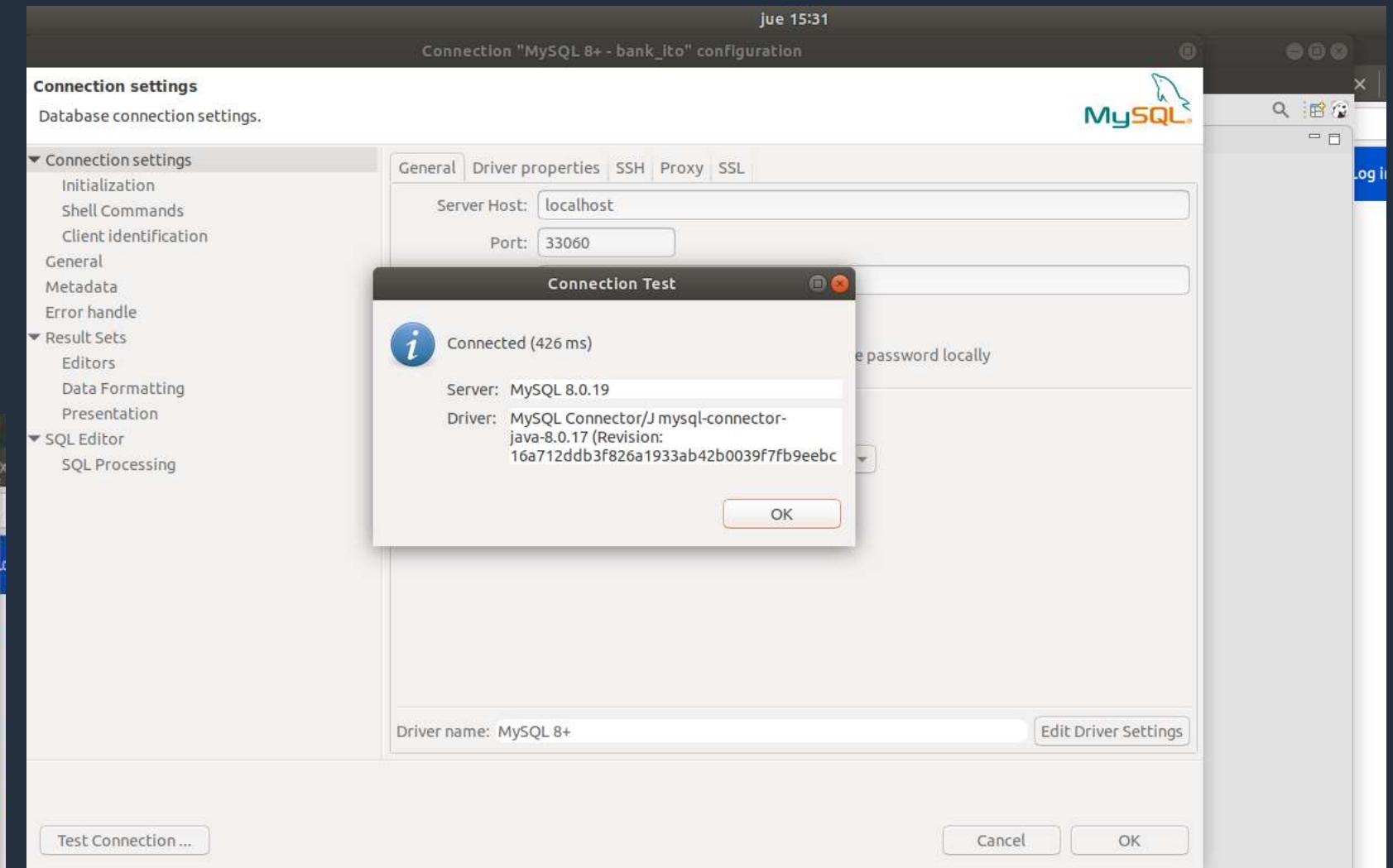
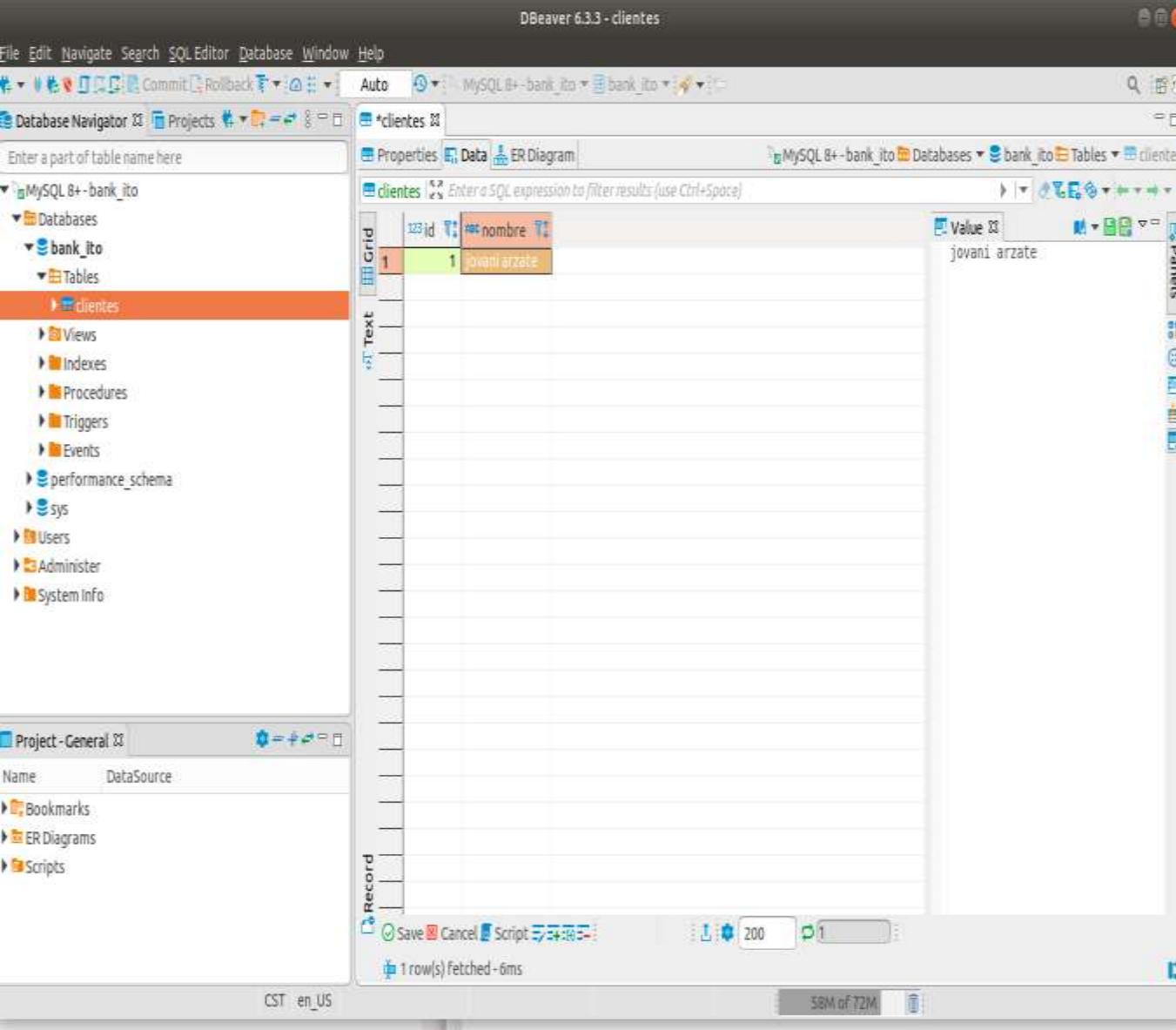
ER Diagrams

Scripts

Record

Save Cancel Script 200 1 1 row(s) fetched +6ms

CST en\_US



# Application Mysql

## Consulta a bd

The screenshot shows the MySQL Workbench interface. On the left, the Database Navigator pane displays the schema 'bank\_ito' with its tables, including 'clientes'. The main workspace shows the 'clientes' table in grid and text modes, with a single record (id: 1, nombre: 'jovani arzate') highlighted. To the right, a terminal window shows the MySQL command-line interface with the following session:

```
jovani@jovani-dev: ~
File Edit View Search Terminal Help
mysql> create table clientes(int id, nombre varchar(250));
ERROR 1064 (42000): You have an error in your SQL syntax; check the manual that
corresponds to your MySQL server version for the right syntax to use near 'int i
d, nombre varchar(250))' at line 1
mysql> create table clientes(id int, nombre varchar(250));
Query OK, 0 rows affected (0.03 sec)

mysql> show tables;
+-----+
| Tables_in_bank_ito |
+-----+
| clientes           |
+-----+
1 row in set (0.00 sec)

mysql> select * from clientes;
+----+-----+
| id | nombre |
+----+-----+
| 1  | jovani arzate |
+----+-----+
1 row in set (0.00 sec)
```

# Hands-ON Portal Web en Docker

**https://git-codecommit.us-east-1.amazonaws.com/v1/repos/rlicaciones/portal-static**

# Portal Web

`docker build -t portal-estatico:0.0.1 .`

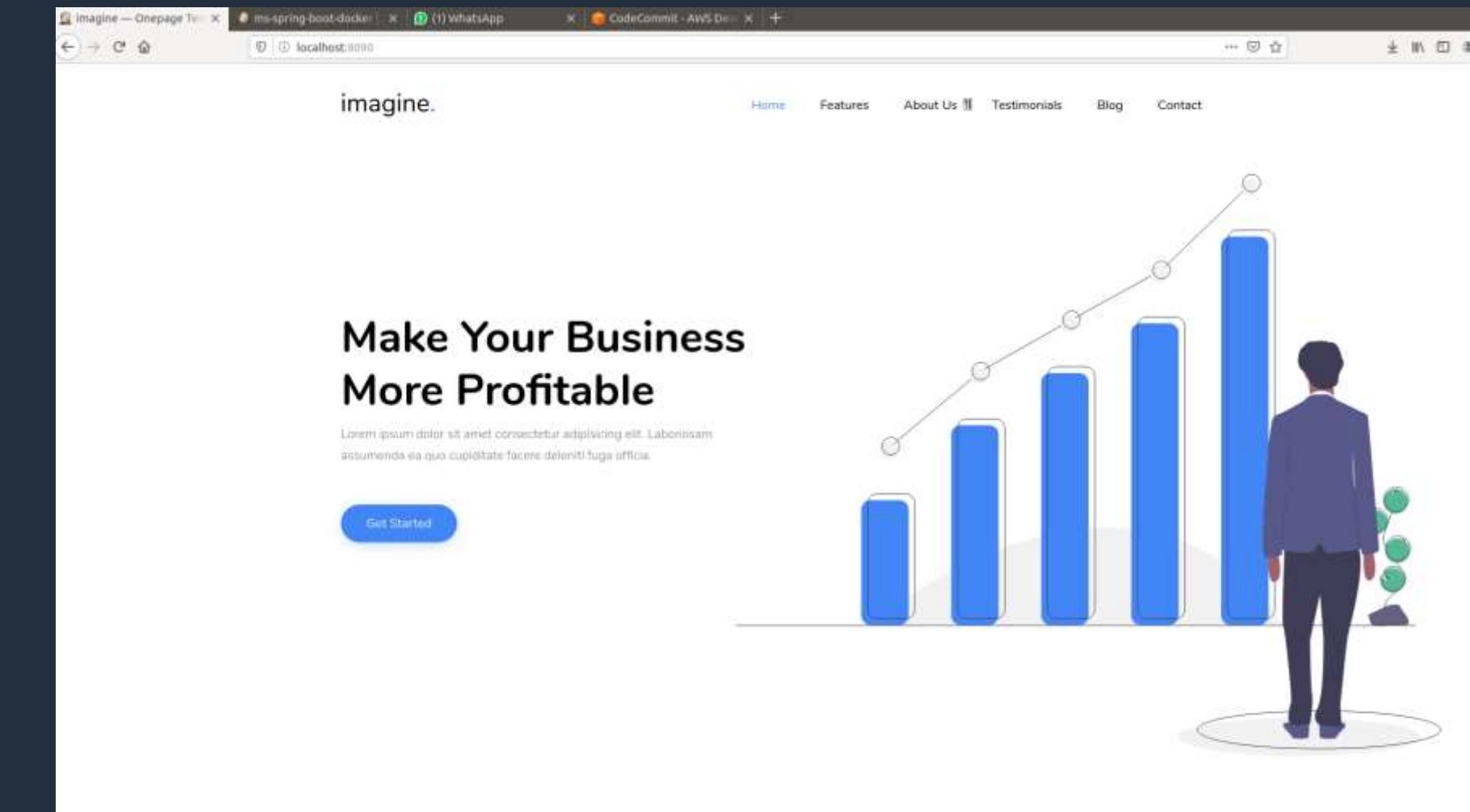
`docker run -it -p 8090:80 portal-estatico:0.0.1`

8090 puerto huesped host : 80 puerto contenedor

**RETO POSTERIOR  
PUBLICARLO EN  
KUBERNETES!!**

Cloud Shell GCP

`docker push portal-estatico:0.1`



<https://8000-dot-7325436-dot-devshell.appspot.com/?authuser=1>





# Sección I

# Cloud Native Patterns

# Cloud Native

“Cloud-native” es un nuevo enfoque de desarrollar y ejecutar aplicaciones las cuales explotan las ventajas del modelo de entrega de la computación en la nube (cloud-computing).

**Las aplicaciones “cloud-native” tratan de cómo se crean y despliegan las aplicaciones, más no en dónde.**

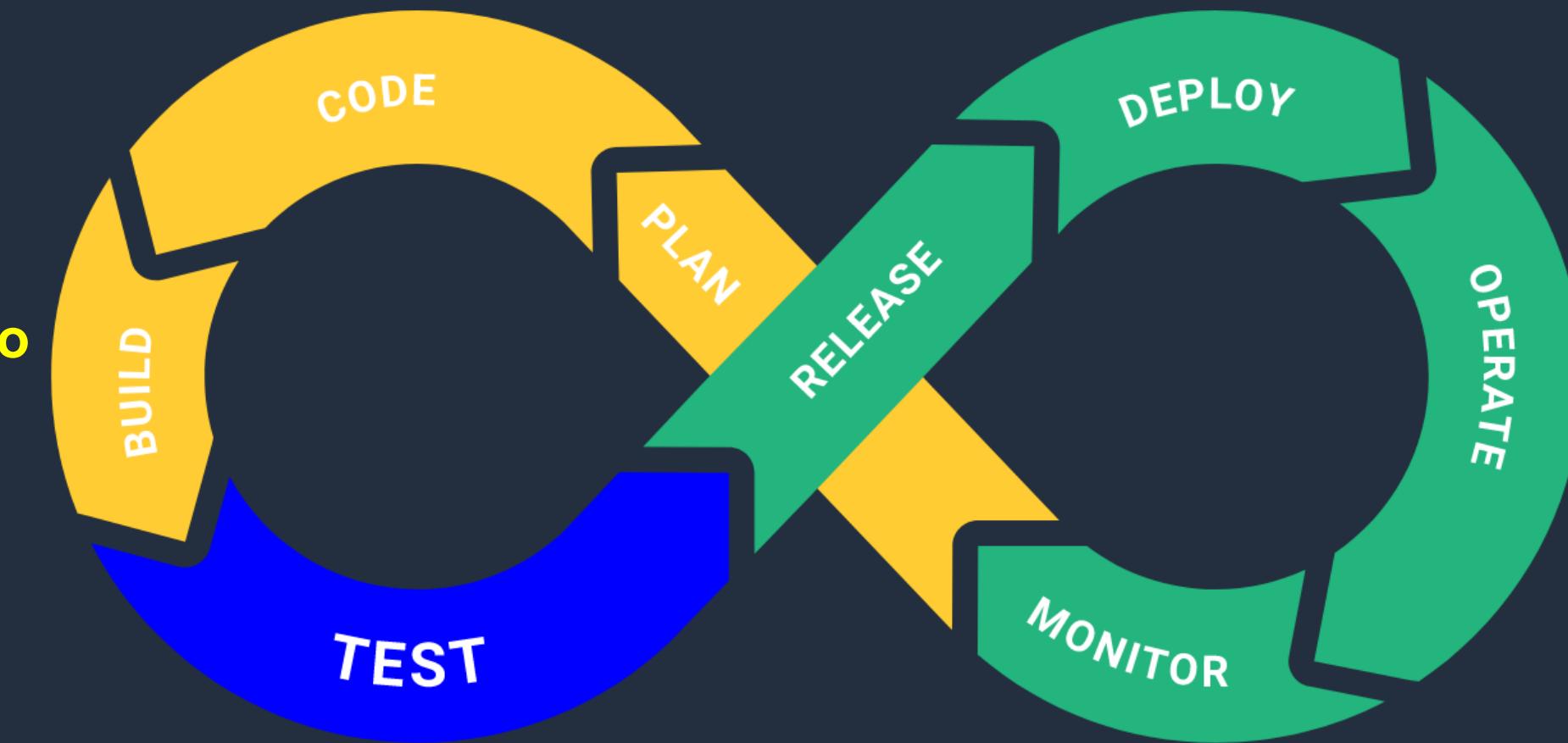
Las organizaciones requieren una plataforma para crear y operar aplicaciones y servicios “cloud-native” que automaticen e integren los conceptos de **DevOps, entrega continua (Continuous Delivery), microservicios y contenedores**.



# Cloud Native - DevOps

- Es una metodología que permite la colaboración entre desarrolladores y personal de operaciones (administradores de sistemas), sin embargo, **requiere un fuerte cambio cultural y de organización, para su correcta implementación, permitiendo la colaboración y la comunicación que permita integrar las áreas de desarrollo y de sistemas.**

- Una buena práctica de DevOps liberá a los desarrolladores para centrarse en hacer lo que mejor saben hacer: escribir software, debido a que DevOps elimina el trabajo y las preocupaciones de la puesta en producción del software una vez que está escrito.

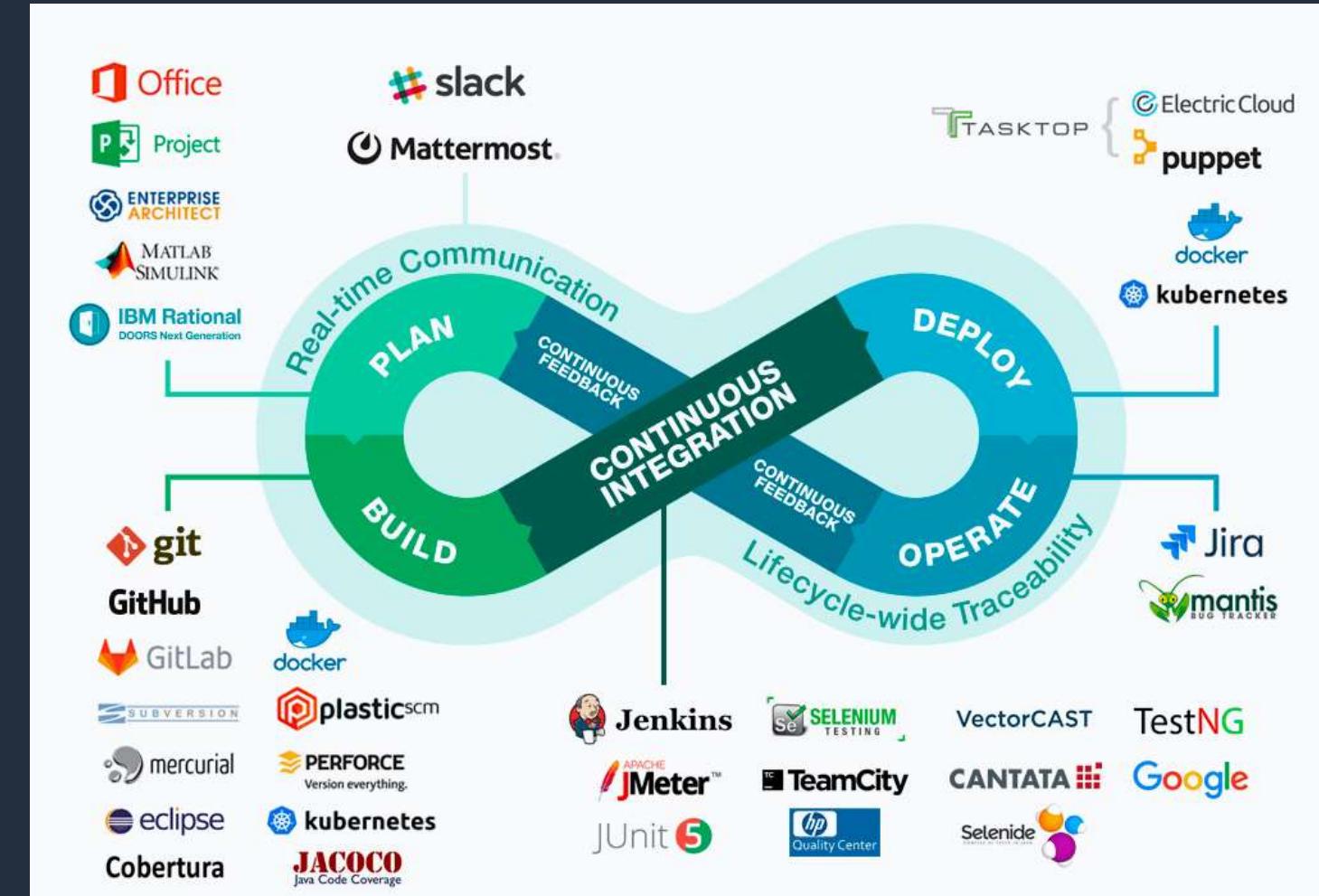


# Cloud Native

Entrega Continua (Continuous Delivery).

- Otro de los pilares de las aplicaciones “cloud-native” es que, mediante DevOps, **todos los pasos en la construcción y despliegue de las aplicaciones estén automatizados, para así evitar el error humano y, agilizar y mejorar el proceso de construcción y despliegue aumentando la calidad del software.**

- La entrega continua facilita que el producto de software **se despliegue y ejecute en entornos productivos en el menor tiempo posible y con mayor continuidad, con el menor costo y la máxima garantía de calidad.**



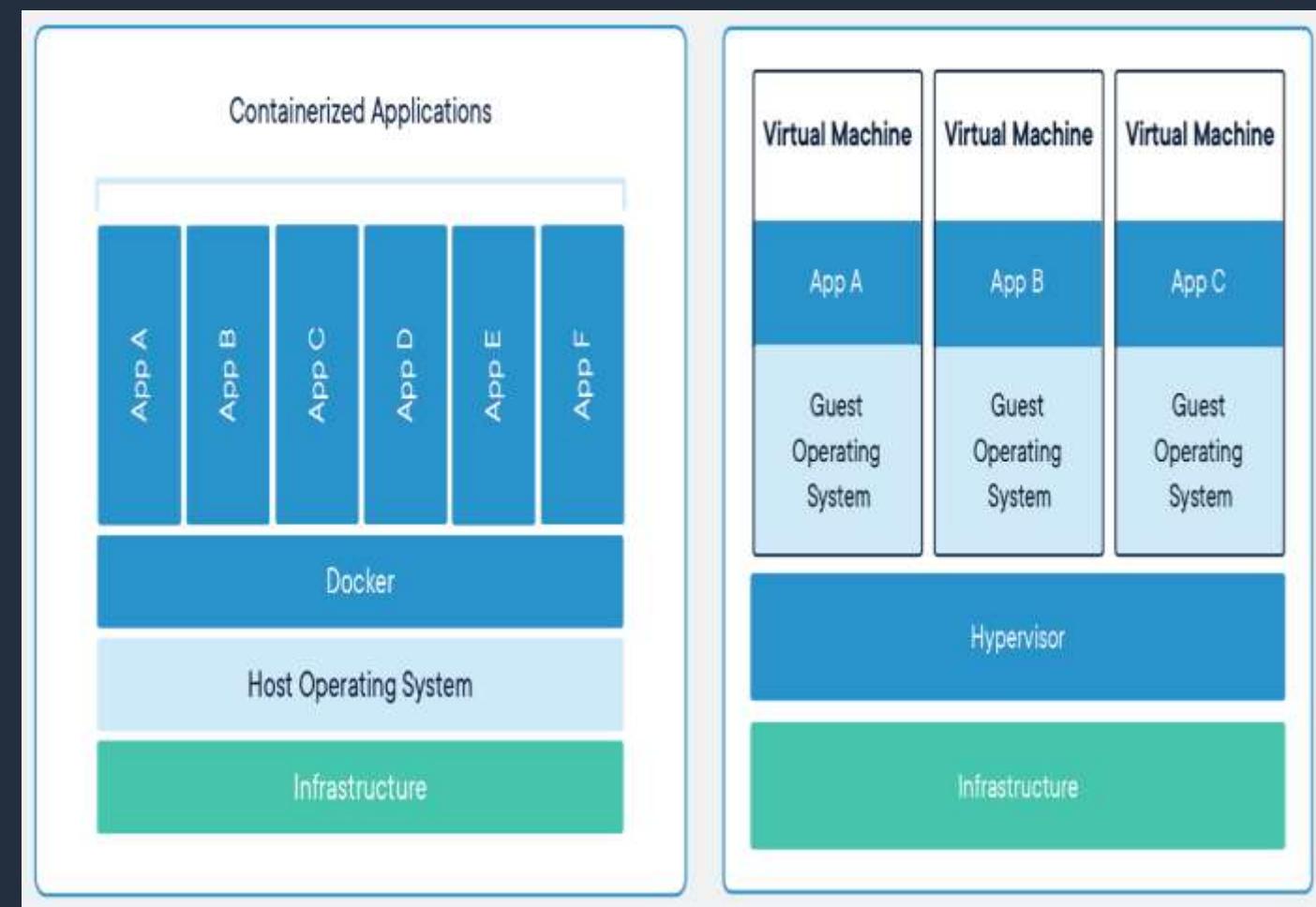
# Cloud Native

Contenedores (Containers).

Los contenedores habilitan garantizar que un producto de **software se ejecuta de la misma manera en cualquier entorno en el que éste se Despliegue.**

**Se evita el famoso dicho “en mi máquina si funciona”.**

Se simplifica ampliamente la virtualización de maquinas debido a que se omite el sistema operativo Host de las VMs, ofreciendo mayor eficiencia y velocidad de procesamiento.



# Cloud Native – Conceptos Claves

**1. Empaquetados como contenedores ligeros:** Las aplicaciones “cloud-native” son una colección de servicios independientes y autónomos que se empaquetan como contenedores livianos los cuales, pueden escalarse Rápidamente.

**2. Desarrollado con los mejores lenguajes y frameworks de trabajo:** Cada servicio de una aplicación “cloud-native” se desarrolla utilizando el lenguaje y/o framework más adecuado para la funcionalidad específica. Las aplicaciones “cloud-native” son políglotas.

**3. Diseñados como microservicios de bajo acoplamiento:** Los servicios que pertenecen a la misma aplicación se descubren en tiempo de ejecución. Los microservicios existen de forma independiente a otros servicios. Una correcta arquitectura de microservicios y la elasticidad de los servicios permite que los mismos se integren correctamente, puedan ser escalados con eficiencia y alto rendimiento.

Los servicios débilmente acoplados permiten a los desarrolladores tratar cada servicio de manera independiente.

**4. Centrado en las API para la interacción y la colaboración:** Los servicios “cloud-native” utilizan APIs ligeras que se basan en protocolos ligeros y abiertos de comunicación como HTTP/REST, gRPC (Google Remote Procedure Call), AMQP, TCP o NATS (open-source, cloud-native messaging system).

REST sobre HTTP, se utiliza ampliamente para exponer las APIs hacia el exterior de la aplicaciones y, para mejorar el rendimiento, se sugiere la implementación de gRPC, AMQP o NATS para la comunicación interna entre los microservicios.



# Cloud Native – Conceptos Claves

## 5. Arquitectura diseñada con una clara separación entre servicios con y sin estado:

Los servicios que son persistentes y duraderos (con estado o stateful) siguen un patrón diferente que asegura una mayor disponibilidad y resistencia. Los servicios sin estado (o stateless) existen independientemente de los servicios con estado.

## 6. Microservicios aislado de dependencias del servidor y del sistema operativo:

Las aplicaciones “cloud-native” no tienen afinidad con ningún sistema operativo en particular o máquina concreta.

La única excepción es cuando un microservicio necesita ciertas capacidades de unidades de estado sólido (SSD) o unidades de procesamiento de gráficos (GPU), que pueden ser ofrecidas exclusivamente por un conjunto de máquinas especializadas.

## 7. Los 10 principales atributos de las aplicaciones “cloud-native”.

Desplegados a través de autoservicio (self-service), mediante infraestructura elástica y en la nube: Las aplicaciones “cloud-native” se despliegan en infraestructura virtual, compartida y elástica, la cual es provista por el mismo equipo de desarrollo en un entorno de auto-Servicio

## 8. Gestionado a través de procesos ágiles de DevOps:

Cada servicio de una aplicación “cloud-native” pasa por un ciclo de vida (pipe-line) independiente, que se gestiona a través de un proceso ágil de DevOps. Múltiples “pipe-lines” de integración continua/entrega continua (CI / CD) trabajan en conjunto para desplegar y administrar una aplicación “cloud-native”.

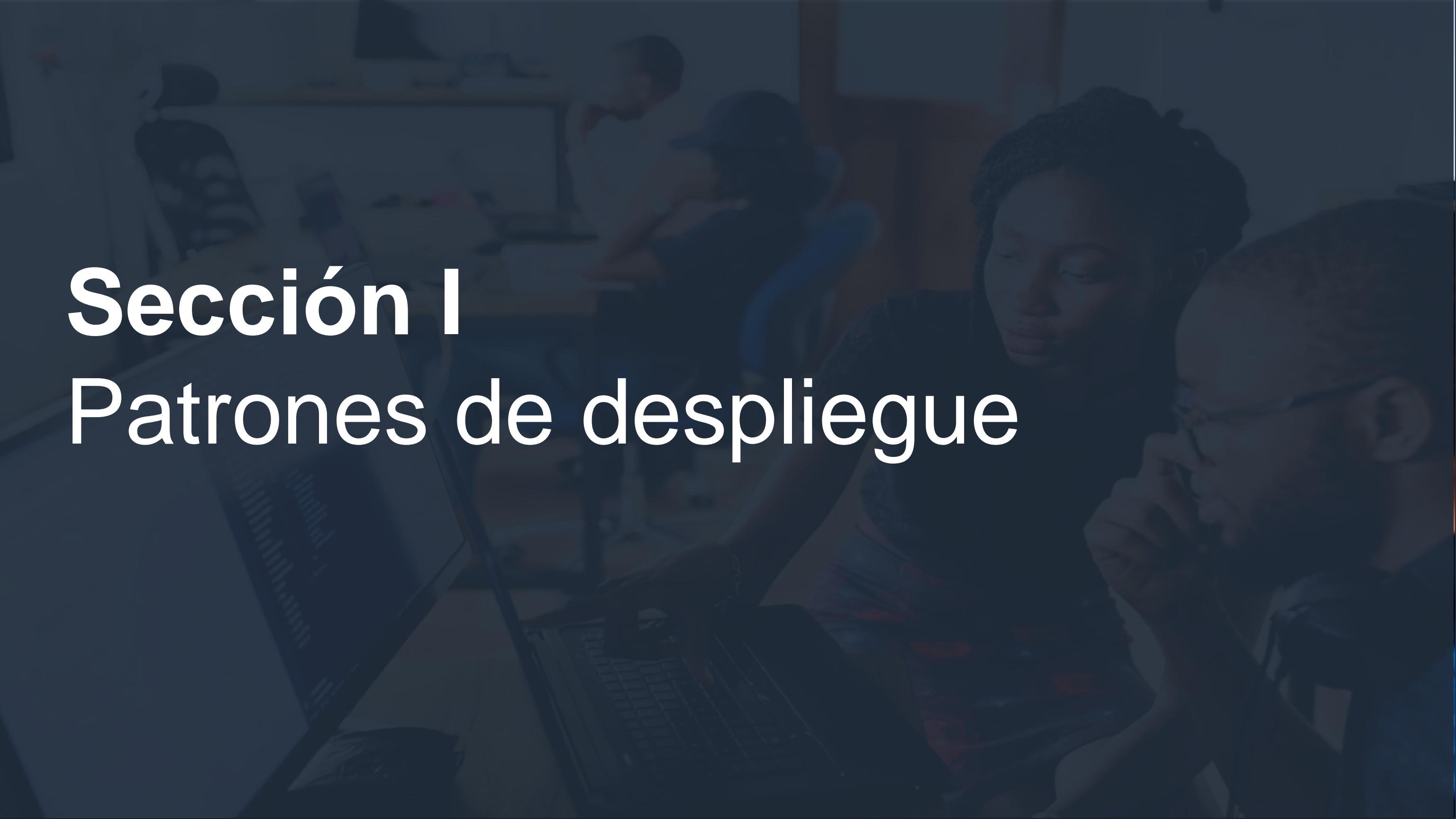


# Cloud Native – Conceptos Claves

9. Capacidades automatizadas: Las aplicaciones “cloud-native” deben ser altamente automatizadas; se implementan correctamente con el concepto de infraestructura como código mediante herramientas como Ansible o Terraform.

10. Asignación de recursos definida y basada en políticas de consumo:  
Las aplicaciones “cloud-native” se alinean con el modelo de gobierno definido a través de un conjunto de políticas de consumo. Se adhieren a políticas de consumo de CPU, cuotas de almacenamiento y, políticas de red que asignan recursos a los servicios.



A blurred background image showing several people in an office environment. Some are sitting at desks with laptops, while others are standing or walking. The scene conveys a sense of teamwork and productivity.

# Sección I

# Patrones de despliegue

# Hands-ON Patrones de Despliegue de microservicios: Dedicate Microservice Database

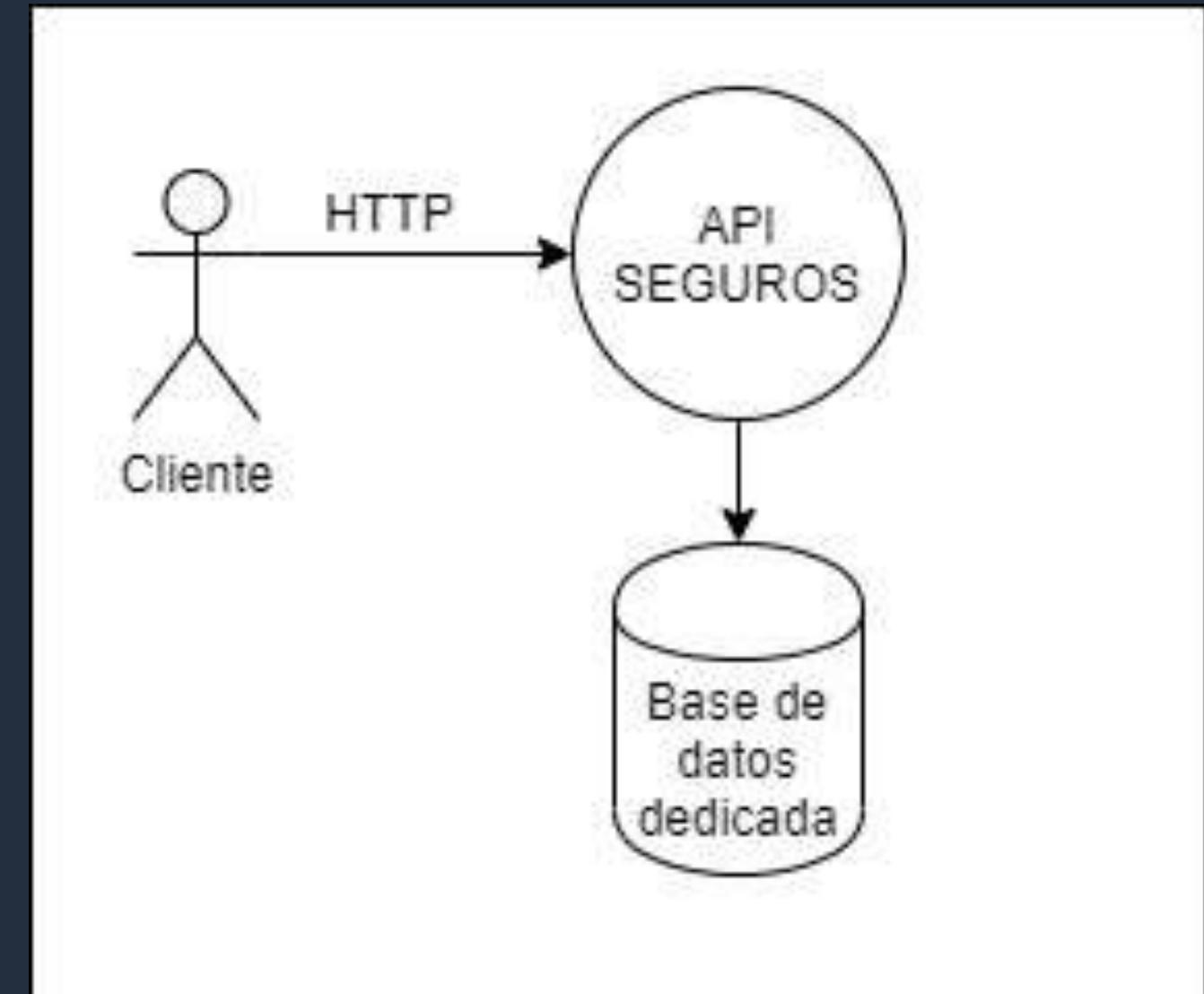
[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab2\\_patrones\\_despliegue](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab2_patrones_despliegue)

ABRIR STS E IMPORTAR PROYECTOS

# Patrones de despliegue y Autonomía en bases de datos

**Base de datos dedicada  
en un servidor dedicado**

**Cada implementación de  
microservicio tiene una  
base de datos dedicada  
en un servidor de bases  
de datos dedicado**

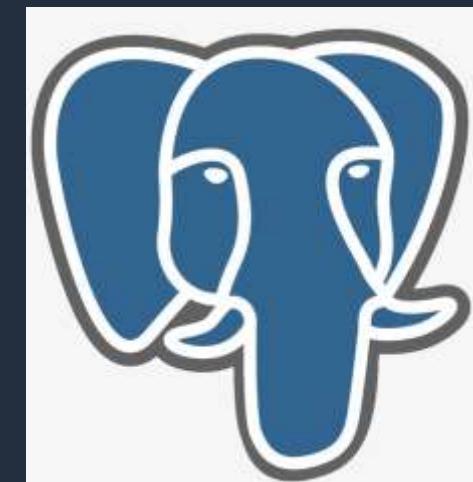
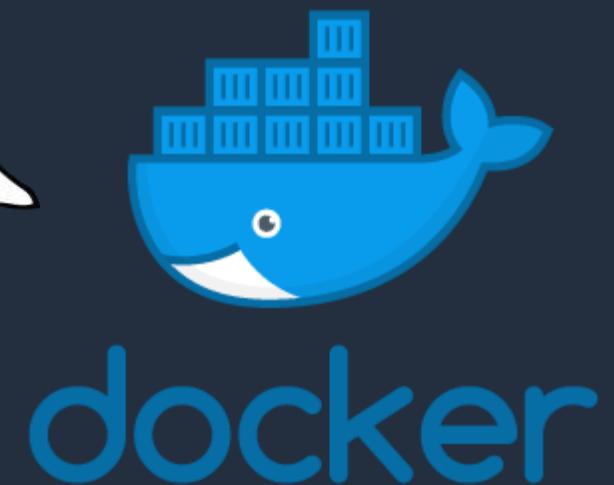


# Dedicate Microservice DataBase - Docker Compose

Tecnología:  
Spring Boot + JPA + Postgres + Docker + Migration + OpenFeign Netflix Oss + Docker-compose

Descripción de microservicios:

- 1.- API CLIENTES JAVA
- 2.- 1 BASE DE DATOS
- 3.- 1 ESQUEMA DE BASE DE DATOS CLIENTES



Podemos realizarlo por pasos:

Ejecutar: ./run-local.sh

```
#!/usr/bin/env sh
```

```
cd api-clientes-microservicio  
./gradlew clean buildImage
```

```
docker-compose up --build
```

```
docker-compose stop
```

```
docker-compose kill
```

```
docker-compose rm -f
```

Docker compose:

Nos permite poder correr diferentes contenedores que se pueden comunicar entre si y a su vez nos ofrece una configuracion simple para poder tener un ambiente de una forma rápida

Eliminar Imagen forzada  
docker rmi -f landa/cliente-ms-compose:0.0.1-snapshot



Podemos realizarlo por pasos:

Ejecutar: ./run-local.sh

```
#!/usr/bin/env sh
```

Validar comandos:!!!!!!!!!!!!!!

```
docker-compose up -d  
docker-compose logs -f  
docker-compose down
```



Dockerfile generado automáticamente:

```
FROM openjdk:8u151-jre-slim
MAINTAINER landa
COPY cliente-ms.jar /opt/cliente-ms.jar
EXPOSE 808
ENTRYPOINT ["java", "-Djava.security.egd=file:/dev/.urandom", "-jar", "/opt/cliente-ms.jar"]
```



## Explicación - Dedicate Microservice DataBase - Docker Compose

```
version: '3.3'
services:
  postgres:
    build: servicesData/postgresdb/ --> indica donde construir la imagen postgres
    environment:
      - POSTGRES_PASSWORD=secret
    ports:
      - 5433:5432 --> puerto host : puerto contenedor de base de datos
    volumes:
      - ./blockstorage/postgres:/var/lib/postgresql/dataz --> volumen base de datos
  clientes-microservicio:
    image: landa/cliente-ms-compose:0.0.1-SNAPSHOT --> imagen microservicio por gradle
    environment:
      - SPRING_DATASOURCE_URL=jdbc:postgresql://postgres:5432/bank_ito --> datasource injectado a el spring boot
    ports:
      - "8081:8081 -->puerto host : puerto contenedor del microservicio
    depends_on:
      - postgres
```



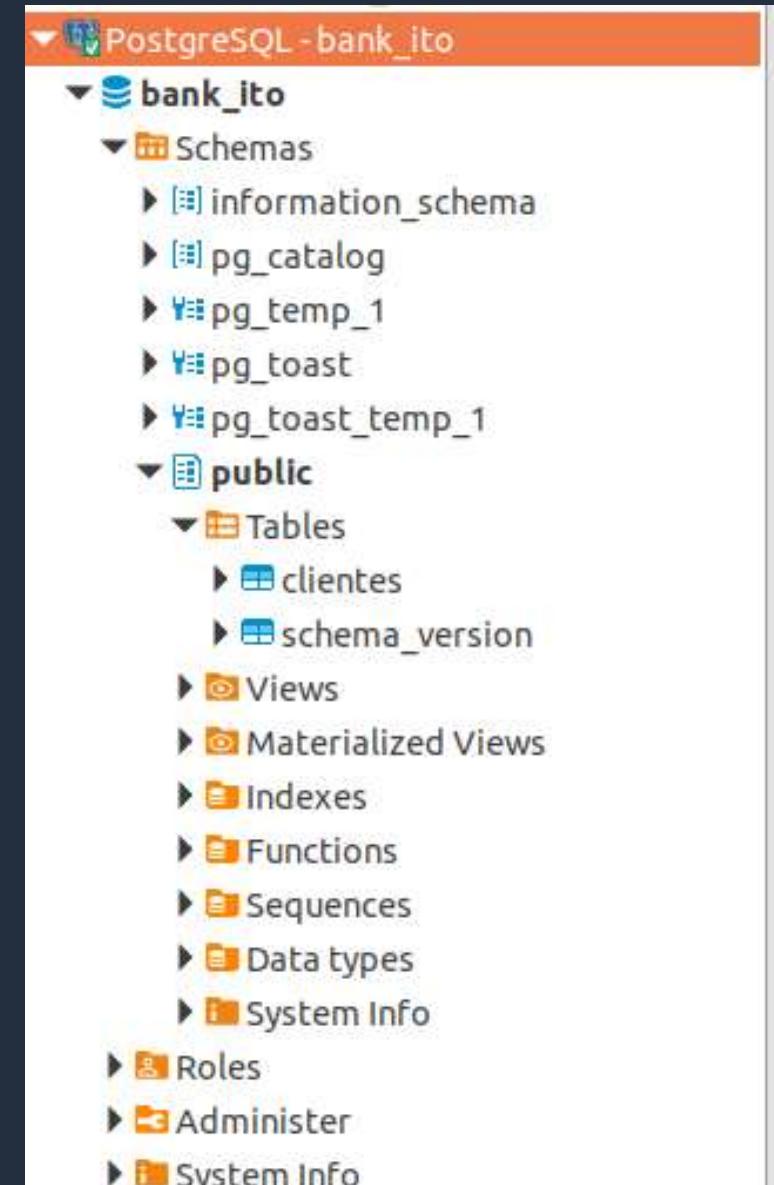
## Migración de base de datos automatizada - Flyway

Dependencias gradle para spring boot:

```
compile('org.springframework.boot:spring-boot-starter-data-jpa')
runtime('org.postgresql:postgresql')
compile('org.flywaydb:flyway-core')
```

Tabla de base de datos

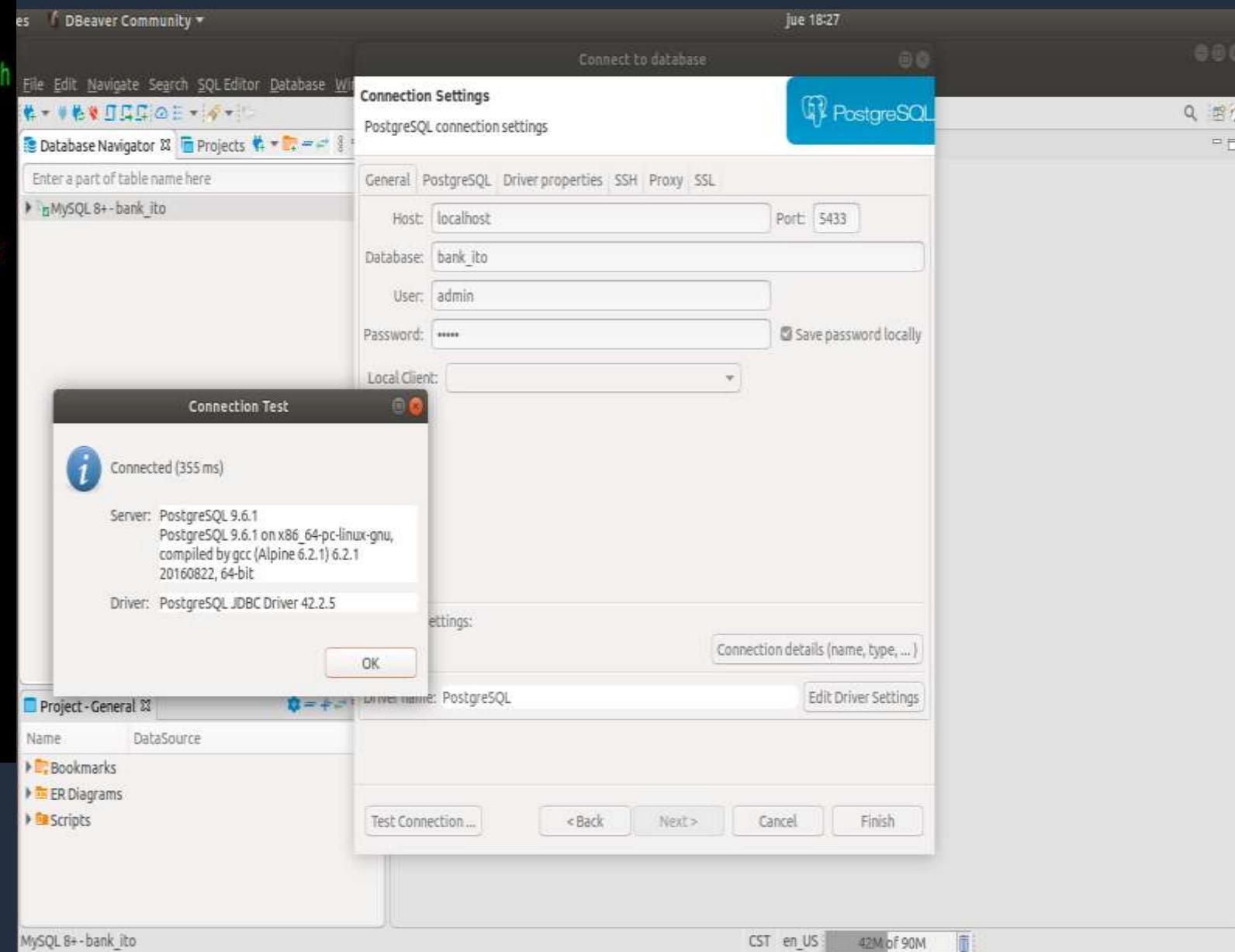
```
CREATE TABLE clientes (
    id      VARCHAR(255) PRIMARY KEY,
    nombre  VARCHAR(255) NOT NULL,
    apellido_paterno VARCHAR(255) NOT NULL,
    apellido_materno VARCHAR(255) NOT NULL,
    email   VARCHAR(255) NOT NULL,
    direccion VARCHAR(255) NOT NULL,
    genero  VARCHAR(255) NOT NULL,
    edad    Integer NOT NULL
);
```



# Explicación - Dedicate Microservice DataBase

```
jovani@developer:~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab1_contenedores/05_docke
File Edit View Search Terminal Help
postgres_1 a17470dcffca server started
postgres_1 a17470dcffca ALTER ROLE
postgres_1 a17470dcffca
postgres_1 a17470dcffca
postgres_1 a17470dcffca
postgres_1 a17470dcffca
postgres_1 a17470dcffca /docker-entrypoint.sh: running /docker-entrypoint-initdb.d/base-datos.sh
postgres_1 a17470dcffca CREATE ROLE
postgres_1 a17470dcffca CREATE DATABASE
postgres_1 a17470dcffca GRANT
postgres_1 a17470dcffca
postgres_1 a17470dcffca waiting for server to shut down....LOG: received fast shutdown request
postgres_1 a17470dcffca LOG: aborting any active transactions
postgres_1 a17470dcffca LOG: autovacuum launcher shutting down
postgres_1 a17470dcffca LOG: shutting down
postgres_1 a17470dcffca LOG: database system is shut down
postgres_1 a17470dcffca done
postgres_1 a17470dcffca server stopped
postgres_1 a17470dcffca
postgres_1 a17470dcffca PostgreSQL init process complete; ready for start up.
postgres_1 a17470dcffca
postgres_1 a17470dcffca LOG: database system was shut down at 2019-06-01 06:12:12 UTC
postgres_1 a17470dcffca LOG: MultiXact member wraparound protections are now enabled
postgres_1 a17470dcffca LOG: database system is ready to accept connections
postgres_1 a17470dcffca LOG: autovacuum launcher started
```

Conectarse a la base de datos



# Explicación - Dedicate Microservice DataBase



Se crea la tabla en la base de datos

Eliminar volumen:  
sudo rm -rf blockstorage/

Postman:

patrones\_despliege/01\_API Rest Clientes post 201  
dedicated\_microservice\_database

patrones\_despliege/02\_API Rest Clientes get 200  
dedicated\_microservice\_database

patrones\_despliege/03\_API Rest Clientes delete 200  
dedicated\_microservice\_database



# Explicación - Dedicate Microservice DataBase

## Datos persistidos en la BD

The screenshot shows the DBeaver 6.3.3 interface for a PostgreSQL database named 'bank\_ito'. The 'clientes' table is selected in the Database Navigator. The Data tab displays the following data:

| id                                   | nombre | apellido_paterno | apellido_materno | email              | direccion |
|--------------------------------------|--------|------------------|------------------|--------------------|-----------|
| f0ddb6d3-87a6-4eb1-b664-148267d4d0fa | jovani | arzate           | cabrera          | jovaniac@gmail.com | test      |

The 'clientes' table has the following columns:

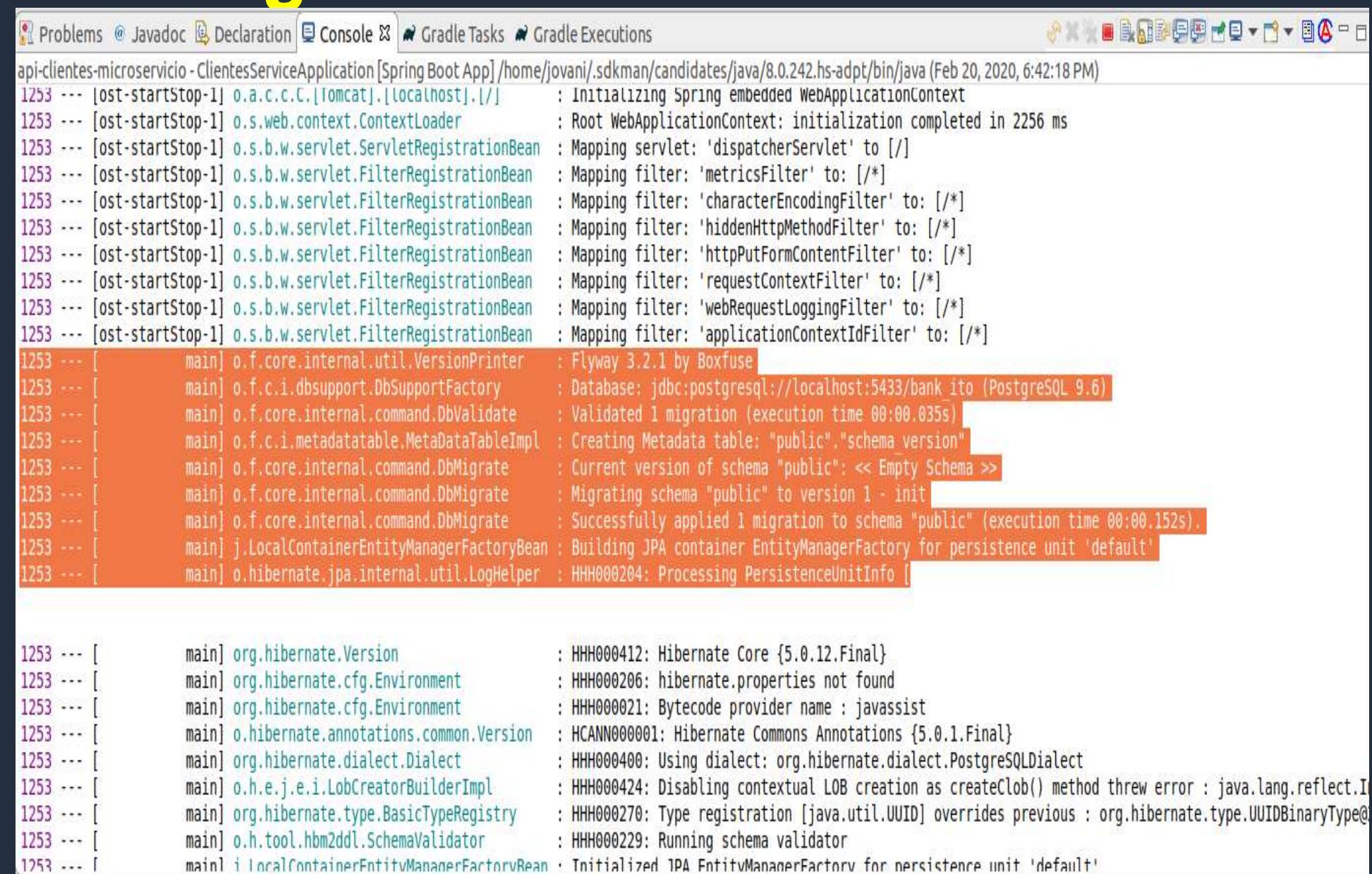
- id (varchar(255))
- nombre (varchar(255))
- apellido\_paterno (varchar(255))
- apellido\_materno (varchar(255))
- email (varchar(255))
- direccion (varchar(255))
- genero (varchar(255))
- edad (int4)

# Explicación - Dedicate Microservice DataBase

**Se pierde la información si se mueren los contenedores? - blockstorage**

**Migración inicial por flyway  
Borrar blockstorage para  
tener desde 0 la persistencia  
de postgres**

**El microservicio crea  
la BD y Tabla con Flyway**

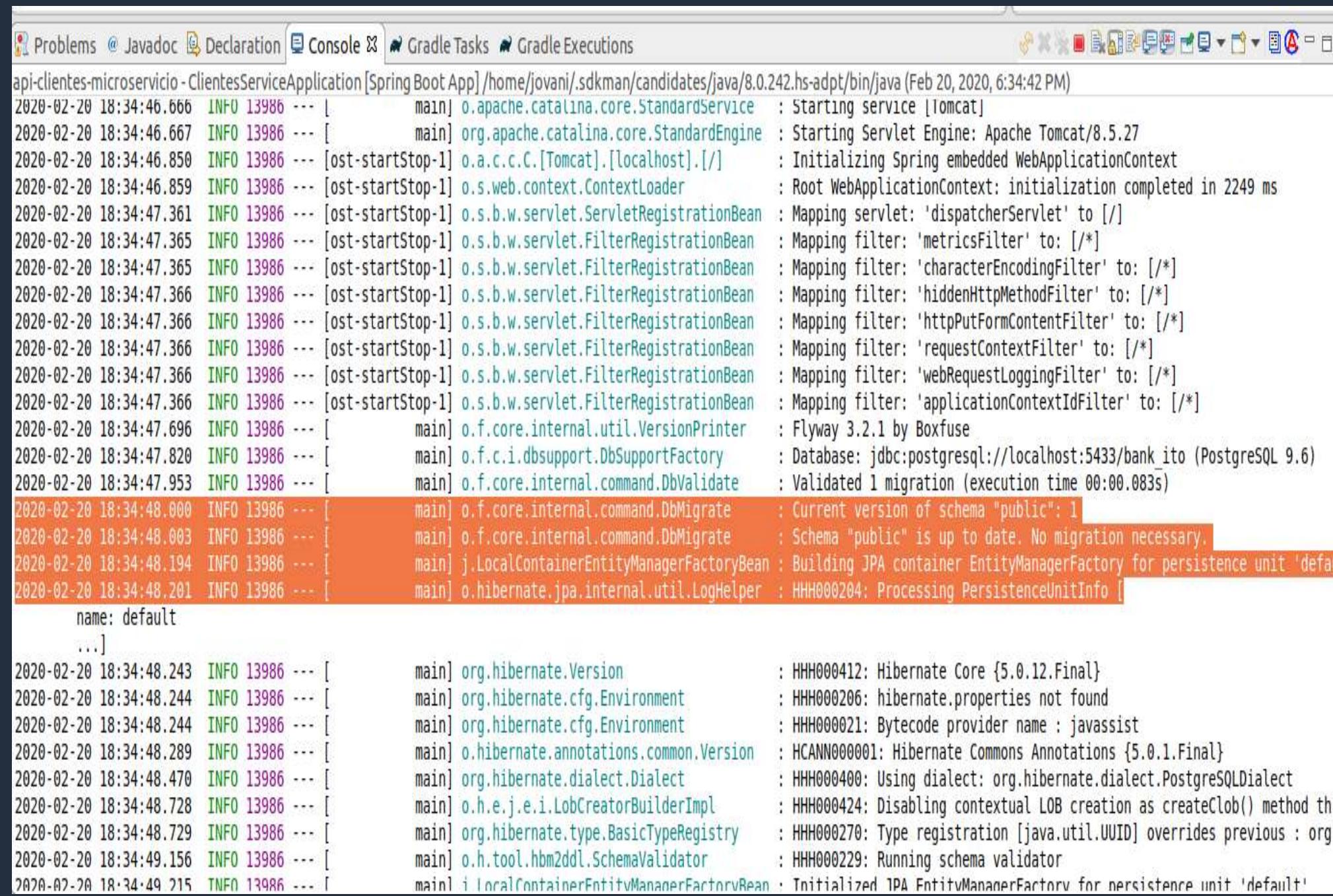


The screenshot shows a Java IDE's integrated terminal (Console tab) displaying the output of a Spring Boot application startup and a Flyway migration process. The logs are color-coded by source: blue for Spring framework logs, red for Flyway logs, and black for other logs.

```
api-clientes-microservicio - ClientesServiceApplication [Spring Boot App] /home/jovani/.sdkman/candidates/java/8.0.242.hs-adpt/bin/java (Feb 20, 2020, 6:42:18 PM)
1253 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].// : Initializing Spring embedded WebApplicationContext
1253 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 2256 ms
1253 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]
1253 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'metricsFilter' to: [*]
1253 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [*]
1253 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [*]
1253 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormContentFilter' to: [*]
1253 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'requestContextFilter' to: [*]
1253 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'webRequestLoggingFilter' to: [*]
1253 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'applicationContextIdFilter' to: [*]
1253 --- [main] o.f.core.internal.util.VersionPrinter : Flyway 3.2.1 by Boxfuse
1253 --- [main] o.f.c.i.dbsupport.DbSupportFactory : Database: jdbc:postgresql://localhost:5433/bank_it0 (PostgreSQL 9.6)
1253 --- [main] o.f.core.internal.command.DbValidate : Validated 1 migration (execution time 00:00.035s)
1253 --- [main] o.f.c.i.metadatatable.MetaDataTableImpl : Creating Metadata table: "public"."schema_version"
1253 --- [main] o.f.core.internal.command.DbMigrate : Current version of schema "public": <> Empty Schema >>
1253 --- [main] o.f.core.internal.command.DbMigrate : Migrating schema "public" to version 1 - init
1253 --- [main] o.f.core.internal.command.DbMigrate : Successfully applied 1 migration to schema "public" (execution time 00:00.152s).
1253 --- [main] j.LocalContainerEntityManagerFactoryBean : Building JPA container EntityManagerFactory for persistence unit 'default'
1253 --- [main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [
1253 --- [main] org.hibernate.Version : HHH000412: Hibernate Core {5.0.12.Final}
1253 --- [main] org.hibernate.cfg.Environment : HHH000206: hibernate.properties not found
1253 --- [main] org.hibernate.cfg.Environment : HHH000021: Bytecode provider name : javassist
1253 --- [main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.0.1.Final}
1253 --- [main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
1253 --- [main] o.h.e.j.e.i.LobCreatorBuilderImpl : HHH000424: Disabling contextual LOB creation as createClob() method threw error : java.lang.reflect.I
1253 --- [main] org.hibernate.type.BasicTypeRegistry : HHH000270: Type registration [java.util.UUID] overrides previous : org.hibernate.type.UUIDBinaryType@
1253 --- [main] o.h.tool.hbm2ddl.SchemaValidator : HHH000229: Running schema validator
1253 --- [main] i.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
```

# Explicación - Dedicate Microservice DataBase

## La segunda vez ya no se migra



The screenshot shows the Eclipse IDE interface with the 'Console' tab selected. The log output is from a Spring Boot application named 'api-clientes-microservicio - ClientesServiceApplication'. The log details the startup process, including the deployment of a Tomcat web application, the initialization of a Spring embedded WebApplicationContext, and the registration of various servlets, filters, and listeners. A significant portion of the log is dedicated to Flyway migrations, which successfully validate one migration and report that the 'public' schema is up-to-date. The log also includes Hibernate Core initialization details, such as provider name and dialect selection.

```
api-clientes-microservicio - ClientesServiceApplication [Spring Boot App] /home/jovani/.sdkman/candidates/java/8.0.242.hs-adpt/bin/java (Feb 20, 2020, 6:34:42 PM)
2020-02-20 18:34:46.666 INFO 13986 --- [           main] o.apache.catalina.core.StandardService : Starting service [tomcat]
2020-02-20 18:34:46.667 INFO 13986 --- [           main] o.apache.catalina.core.StandardEngine : Starting Servlet Engine: Apache Tomcat/8.5.27
2020-02-20 18:34:46.850 INFO 13986 --- [ost-startStop-1] o.a.c.c.C.[Tomcat].[localhost].[] : Initializing Spring embedded WebApplicationContext
2020-02-20 18:34:46.859 INFO 13986 --- [ost-startStop-1] o.s.web.context.ContextLoader : Root WebApplicationContext: initialization completed in 2249 ms
2020-02-20 18:34:47.361 INFO 13986 --- [ost-startStop-1] o.s.b.w.servlet.ServletRegistrationBean : Mapping servlet: 'dispatcherServlet' to [/]
2020-02-20 18:34:47.365 INFO 13986 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'metricsFilter' to: [/*]
2020-02-20 18:34:47.365 INFO 13986 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'characterEncodingFilter' to: [/*]
2020-02-20 18:34:47.366 INFO 13986 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'hiddenHttpMethodFilter' to: [/*]
2020-02-20 18:34:47.366 INFO 13986 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'httpPutFormContentFilter' to: [/*]
2020-02-20 18:34:47.366 INFO 13986 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'requestContextFilter' to: [/*]
2020-02-20 18:34:47.366 INFO 13986 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'webRequestLoggingFilter' to: [/*]
2020-02-20 18:34:47.366 INFO 13986 --- [ost-startStop-1] o.s.b.w.servlet.FilterRegistrationBean : Mapping filter: 'applicationContextIdFilter' to: [/*]
2020-02-20 18:34:47.696 INFO 13986 --- [           main] o.f.core.internal.util.VersionPrinter : Flyway 3.2.1 by Boxfuse
2020-02-20 18:34:47.820 INFO 13986 --- [           main] o.f.c.i.dbsupport.DbSupportFactory : Database: jdbc:postgresql://localhost:5433/bank_it (PostgreSQL 9.6)
2020-02-20 18:34:47.953 INFO 13986 --- [           main] o.f.core.internal.command.DbValidate : Validated 1 migration (execution time 00:00.083s)
2020-02-20 18:34:48.000 INFO 13986 --- [           main] o.f.core.internal.command.DbMigrate : Current version of schema "public": 1
2020-02-20 18:34:48.003 INFO 13986 --- [           main] o.f.core.internal.command.DbMigrate : Schema "public" is up to date. No migration necessary.
2020-02-20 18:34:48.194 INFO 13986 --- [           main] j.LocalContainerEntityManagerFactoryBean : Building JPA container EntityManagerFactory for persistence unit 'default'
2020-02-20 18:34:48.201 INFO 13986 --- [           main] o.hibernate.jpa.internal.util.LogHelper : HHH000204: Processing PersistenceUnitInfo [
    name: default
    ...
]
2020-02-20 18:34:48.243 INFO 13986 --- [           main] org.hibernate.Version : HHH000412: Hibernate Core {5.0.12.Final}
2020-02-20 18:34:48.244 INFO 13986 --- [           main] org.hibernate.cfg.Environment : HHH000206: hibernate.properties not found
2020-02-20 18:34:48.244 INFO 13986 --- [           main] org.hibernate.cfg.Environment : HHH00021: Bytecode provider name : javassist
2020-02-20 18:34:48.289 INFO 13986 --- [           main] o.hibernate.annotations.common.Version : HCANN000001: Hibernate Commons Annotations {5.0.1.Final}
2020-02-20 18:34:48.470 INFO 13986 --- [           main] org.hibernate.dialect.Dialect : HHH000400: Using dialect: org.hibernate.dialect.PostgreSQLDialect
2020-02-20 18:34:48.728 INFO 13986 --- [           main] o.h.e.j.e.i.LobCreatorBuilderImpl : HHH000424: Disabling contextual LOB creation as createClob() method throws java.sql.SQLException
2020-02-20 18:34:48.729 INFO 13986 --- [           main] org.hibernate.type.BasicTypeRegistry : HHH000270: Type registration [java.util.UUID] overrides previous : org.
2020-02-20 18:34:49.156 INFO 13986 --- [           main] o.h.tool.hbm2ddl.SchemaValidator : HHH000229: Running schema validator
2020-02-20 18:34:49.215 TNFO 13986 --- [           main] i.LocalContainerEntityManagerFactoryBean : Initialized JPA EntityManagerFactory for persistence unit 'default'
```



# Sección I

# Spring Cloud Netflix

# OSS

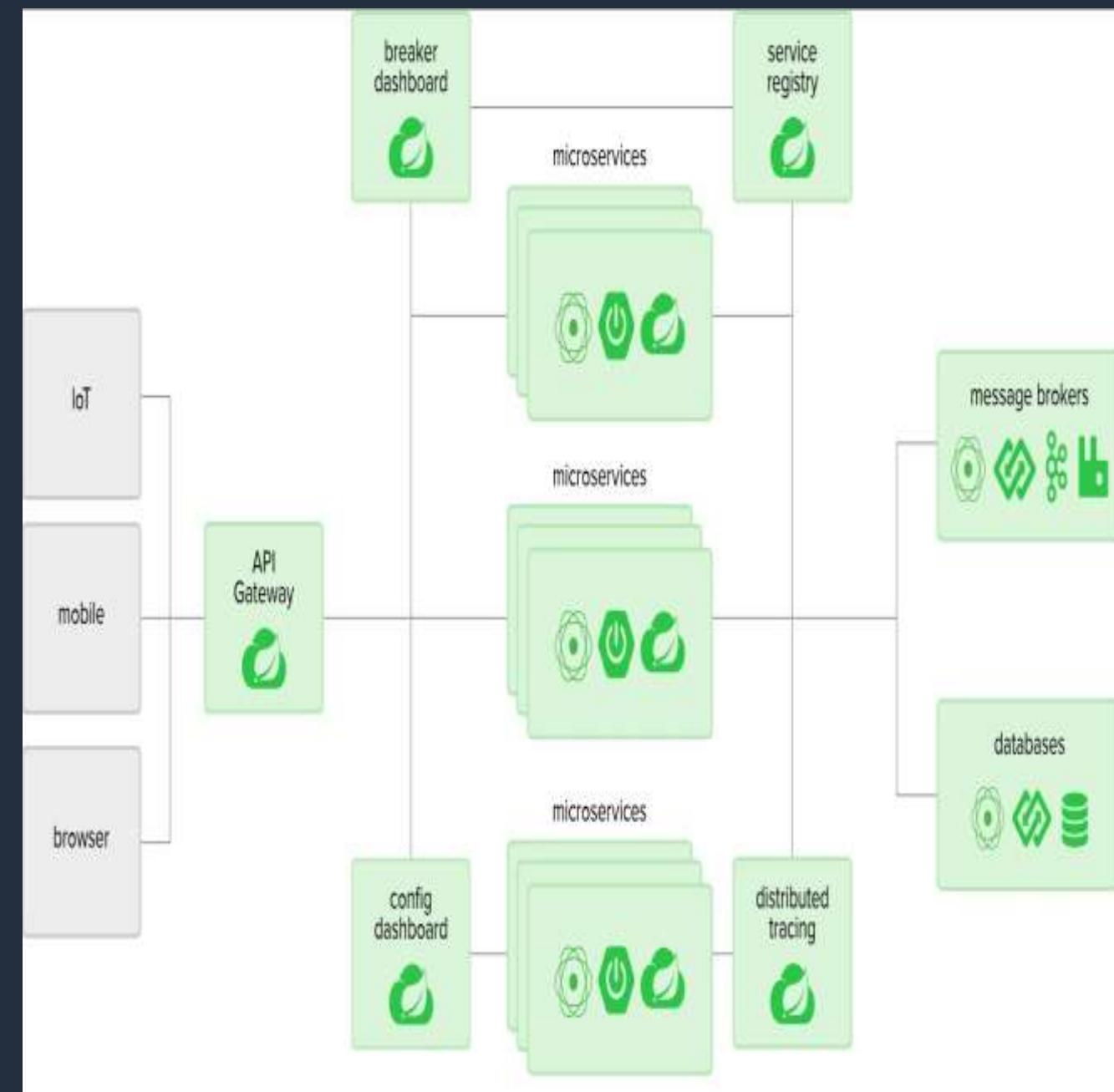


# Spring Cloud

# Indicar Spring Cloud Netflix OSS

Spring Cloud Netflix OSS (Open-Source Software) provee integraciones para Spring Cloud mediante implementaciones de patrones ampliamente probados por Netflix (“well-battle-tested-components”) para aplicaciones distribuidas tales como:

- Eureka: Servicio de descubrimiento y registro de servicios.
- Hystrix: Implementación de “Circuit Breaker Pattern”.
- Zuul: Servidor de borde o “Edge Server” implementando funcionalidades de API Gateway y enrutamiento inteligente.
- Ribbon: Balanceo de carga del lado del cliente.
- Feign: Implementación de clientes REST declarativos.
- Archaius: Servicio de configuración externa.
- entre otros: <https://netflix.github.io/>



# Sección I

# Comunicación HTTP



# Hands-ON Comunicación HTTP en Microservicios y Patron Shared Database

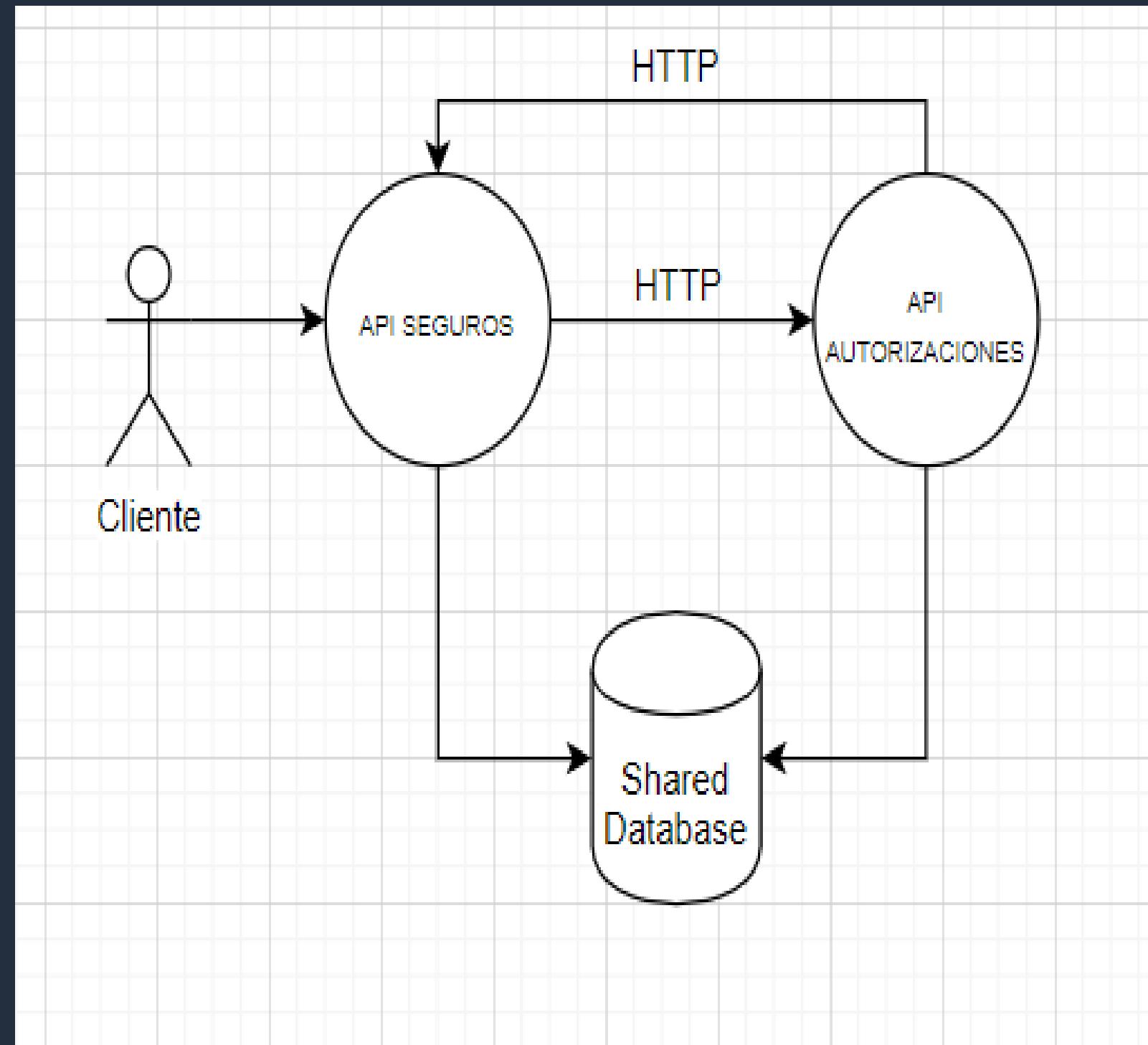
[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab3\\_comunicacion\\_http](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab3_comunicacion_http)

ABRIR STS E IMPORTAR PROYECTOS

# Patrones de despliegue y Autonomía en bases de datos

**Esquemas de base de datos dedicados en un servidor compartido.**

**Cada implementación de microservicio tiene acceso a un conjunto dedicado de esquemas alojados por una base de datos compartida en un servidor de base de datos**



# Caso de uso: **Seguros Guadalupe**

**Empresa dedicada, a ofrecer servicios de polizas de seguros para el público en general.**

**Tiene un departamento, encargado de validar todas las polizas y hacer estudio de mercado para garantizar a quién se le otorgarán los servicios de los seguros.**



# Clientes REST Feign Declarativos

## ¿Qué es Feign?

- Feign es una librería, escrita originalmente por Netflix, que se integra con Ribbon para crear clientes de servicios web REST sobre HTTP de forma declarativa.
- Permite realizar llamadas a "web-services" REST a partir de una interface y sin implementar el código de la llamada HTTP (al estilo de Spring Data para crear implementaciones DAO al vuelo u "on-the-fly").
- Feign es una alternativa a utilizar el API nativa de Ribbon o el RestTemplate balanceado con Ribbon.

OpenFeign es el nuevo nombre que se le dio al proyecto Feign.

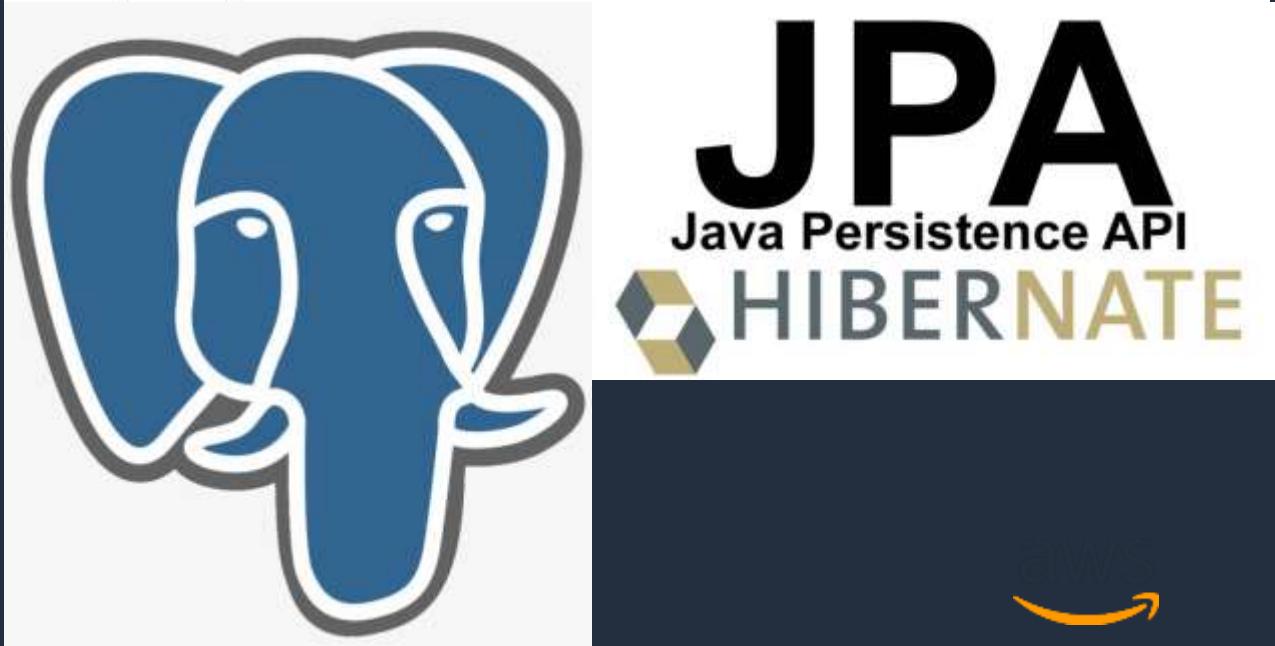
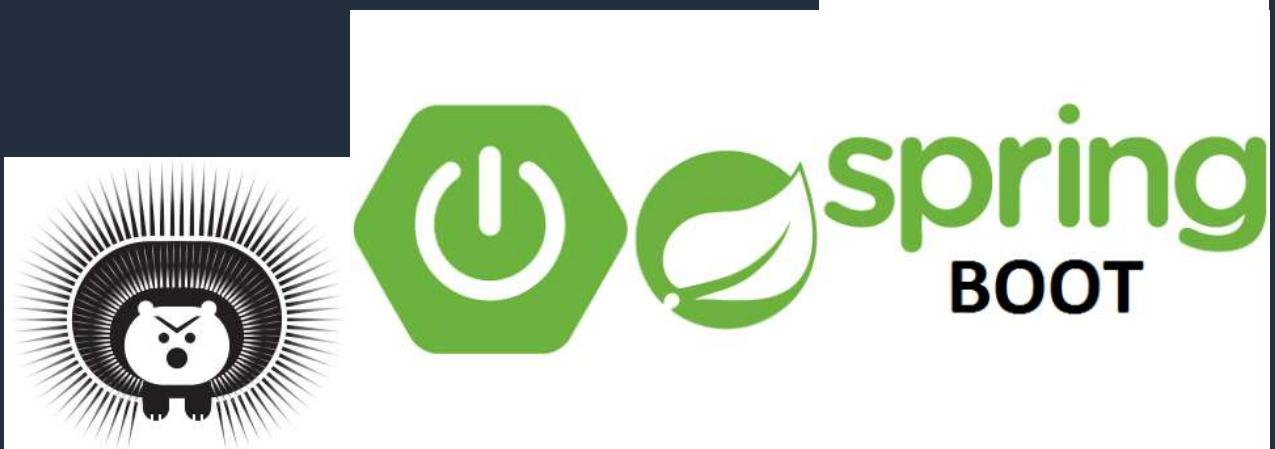
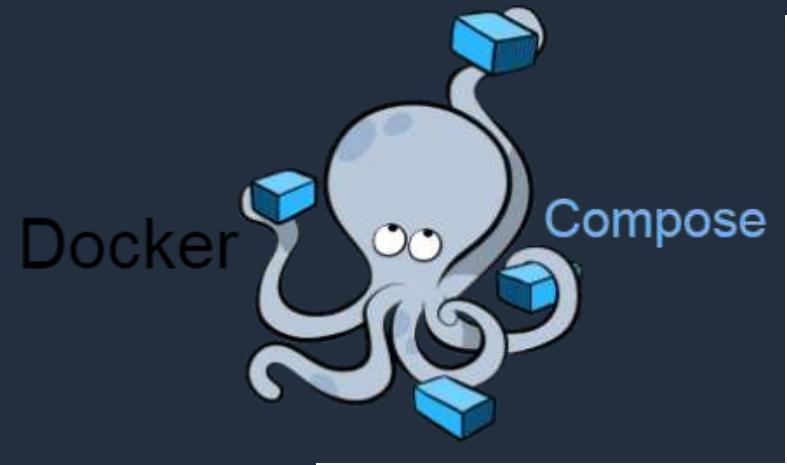
Spring Cloud Feign permite integrar OpenFeign, Ribbon y/o Eureka, en aplicaciones Spring Cloud, para crear clientes REST declarativos, mediante anotaciones (sobre interfaces) en tiempo de ejecución.

- No es requerido el uso de Eureka.
- Requiere Ribbon para su implementación.
- Soporta anotaciones de Spring MVC para definir el comportamiento del cliente REST.
- Las interfaces Feign pueden ser compartidas entre el cliente y el servidor de un "web-service" REST particular y, utilizado como "contrato" establecido, bien definido y versionado.



# Comunicación HTTP y Shared Database

Tecnologías:  
Spring Boot + JPA + PostgreSQL + Docker + Migration + OpenShift + Netflix OSS + Docker-compose + Hypack  
Descripción de microservicios:  
1.- API SEGURO  
2.- AUTORIZACION DE AUTORIZACIONES  
3.- BASE DE DATOS  
4.- MICROSERVICIOS EN LA EDI  
5.- ECUATORIA - SEGURO GUARDIA LIFE  
6.- ECUATORIA - SEGURO GUARDIA ADMINISTRACION



```
./run-local.sh  
sudo rm -rf blockstorage/  
docker stop <id container>
```



# Ejecución - Configuracion de arranque

Script de automatización

Ejecutar: ./run-local.sh

```
#!/usr/bin/env bash
```

```
cd api-seguros
```

```
./generalImagen.sh
```

```
cd ..
```

```
cd api-adm-autorizaciones
```

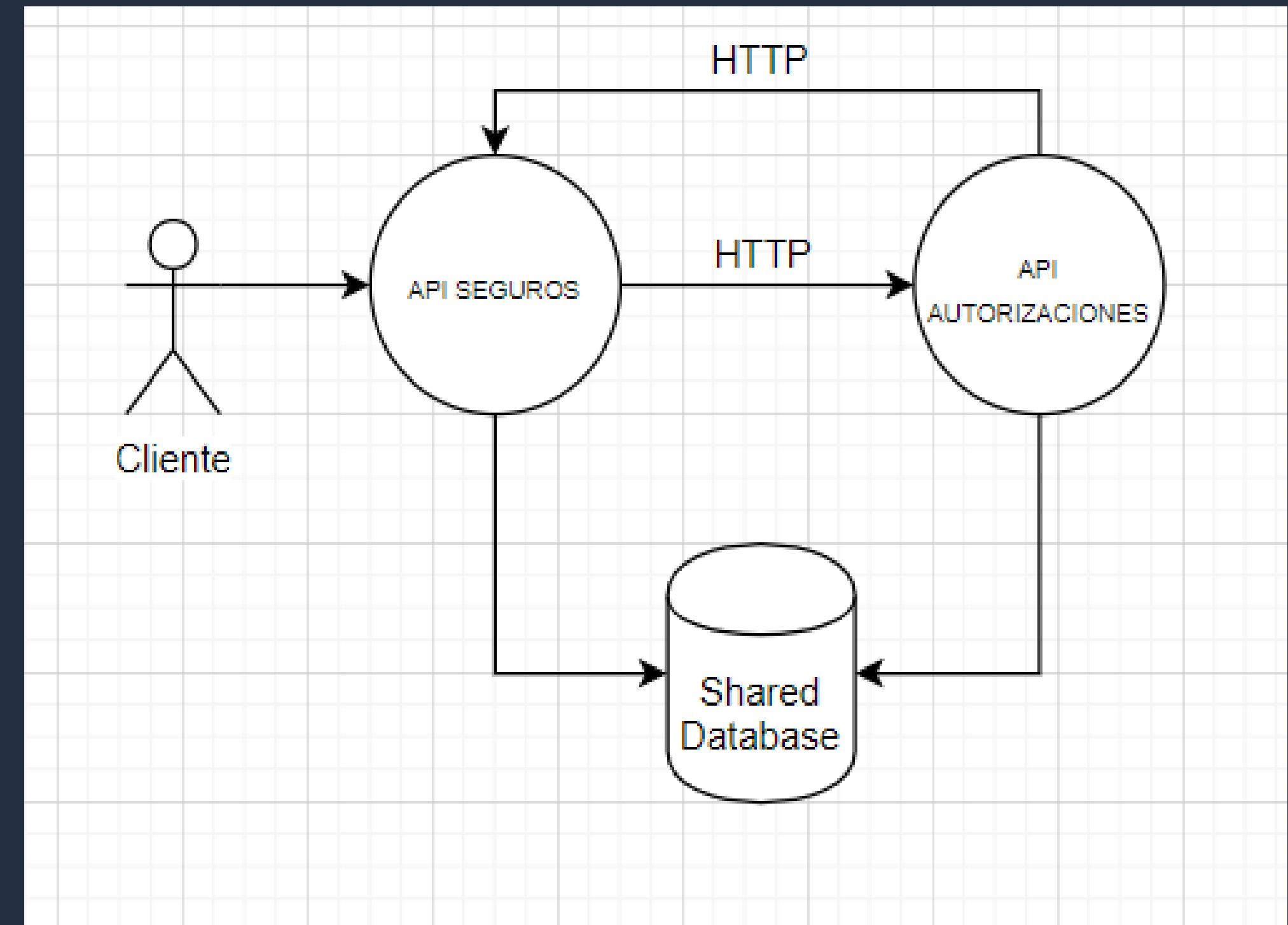
```
./generalImagen.sh
```

```
docker-compose up --build
```

```
docker-compose stop
```

```
docker-compose kill
```

```
docker-compose rm -f
```



# Explicación - Configuracion de arranque

## Orquestacion de contenedores docker-compose

```
version: '3.3'
services:
  postgres:
    build: servicesData/postgresdb/
    environment:
      - POSTGRES_PASSWORD=secret
    ports:
      - 5432:5432
    volumes:
      - ./blockstorage/postgres:/var/lib/postgresql/dataz
  api-seguros-guadalupe:
    image: landa/api-seguros-guadalupe-http:0.0.1-snapshot
    environment:
      - SPRING_DATASOURCE_URL=jdbc:postgresql://postgres:5432/seguros_guadalupe
    ports:
      - "8081:8081"
    depends_on:
      - postgres
  api-seguros-guadalupe-administracion:
    image: landa/api-seguros-guadalupe-administracion-http:0.0.1-snapshot
    environment:
      -
      SPRING_DATASOURCE_URL=jdbc:postgresql://postgres:5432/seguros_guadalupe_administracion
    ports:
      - "8082:8082"
    depends_on:
      - postgres
```



# Explicación - Configuracion de arranque

## Creacion de bases de datos

```
#!/bin/bash
```

```
set -e
```

```
psql -v ON_ERROR_STOP=1 --username "$POSTGRES_USER" <<-EOSQL
```

```
CREATE USER admin PASSWORD 'admin';
```

```
CREATE DATABASE seguros_guadalupe OWNER admin;
```

```
GRANT ALL PRIVILEGES ON DATABASE seguros_guadalupe TO admin;
```

```
CREATE DATABASE seguros_guadalupe_administracion OWNER admin;
```

```
GRANT ALL PRIVILEGES ON DATABASE seguros_guadalupe_administracion TO admin;
```

```
EOSQL
```



# Explicación - Configuracion de arranque

## Migracion de bases de datos api-seguros

```
CREATE TABLE seguros (
    id      VARCHAR(255) PRIMARY KEY,
    plazo double precision DEFAULT NULL,
    precio_poliza double precision DEFAULT NULL,
    tipo_cobertura Integer,
    vencimiento VARCHAR(1000) NOT NULL,
    suma_asegurada double precision DEFAULT NULL,
    nombre VARCHAR(255) NOT NULL,
    apellido_paterno VARCHAR(255) NOT NULL,
    apellido_materno VARCHAR(255) NOT NULL,
    fecha_nacimiento VARCHAR(255) NOT NULL,
    sexo VARCHAR(255) NOT NULL,
    ingresos_anuales double precision DEFAULT NULL,
    direccion VARCHAR(500) NOT NULL,
    autorizacion BOOLEAN DEFAULT FALSE
);
```

```
CREATE TABLE catalogo_seguros (
    id      VARCHAR(255) PRIMARY KEY,
    nombre VARCHAR(500) NOT NULL,
    descripcion VARCHAR(500) NOT NULL,
    estatus BOOLEAN NOT NULL DEFAULT FALSE
);
```

```
insert into catalogo_seguros(id, nombre, descripcion, estatus) VALUES(1,'seguro-medico-juventud','seguro para personas entre 12 - 20',true);
insert into catalogo_seguros(id, nombre, descripcion, estatus) VALUES(2,'seguro-medico-mayor','seguro para personas entre 23 - 35',true);
insert into catalogo_seguros(id, nombre, descripcion, estatus) VALUES(3,'seguro-medico-infantil','seguro para personas entre 35 - 55',true);
```



# Explicación - Configuracion de arranque

## Migracion de bases de datos api-administracion-autorizaciones

```
CREATE TABLE autorizaciones (
    id_autorizacion    VARCHAR(255) PRIMARY KEY,
    datos_poliza    VARCHAR NOT NULL,
    estatus BOOLEAN NOT NULL DEFAULT FALSE,
    fecha_apertura VARCHAR,
    fecha_confirmacion VARCHAR
);
```

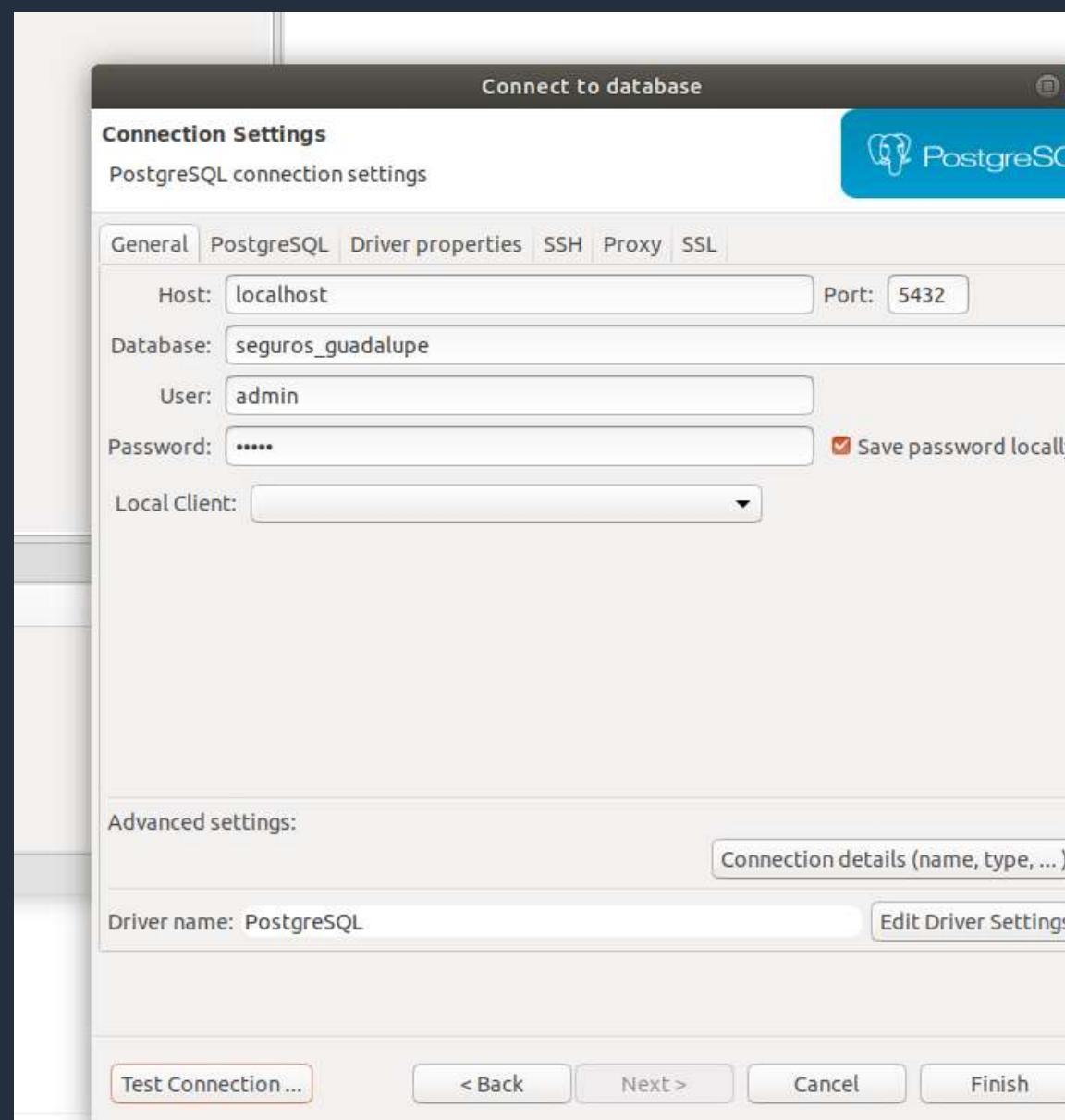


# Explicación - Comunicación HTTP y Shared Database

## 1.- Contenedores en linea

## 2.- abrir cliente de base de datosdbeaver

## 3.- hacer peticiones via postman



The terminal window displays the following logs from a Java application running in a Docker container:

```
jovani@jovani-dev:~/Documents/apixcloudservice/arquitecto_apis_microservicios/us_micservicios/labz_comunicacion_nccp$ java -jar target/api-seguros-guadalupe-0.0.1-snapshot.jar
2020-02-21 05:45:22.068 INFO 1 --- [           main] org.springframework.web.servlet.ModelAndView org.springframework.cloud.netflix.endpoint.HttpServletWrappingEndpoint.handle(javax.servlet.http.HttpServletRequest, javax.servlet.http.HttpServletResponse) throws java.lang.Exception
api-seguros-guadalupe_1 | 2020-02-21 05:45:22.068 INFO 1 --- [           main]
o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/health || /health.json],methods=[GET],produces=[application/vnd.spring-boot.actuator.v1+json || application/json]}" onto public java.lang.Object org.springframework.boot.actuate.endpoint.mvc.HealthMvcEndpoint.invoke(javax.servlet.http.HttpServletRequest,java.security.Principal)
api-seguros-guadalupe_1 | 2020-02-21 05:45:22.080 INFO 1 --- [           main]
o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/env],methods=[POST]}" onto public java.lang.Object org.springframework.cloud.context.environment.EnvironmentManagerMvcEndpoint.value(java.util.Map<java.lang.String, java.lang.String>)
api-seguros-guadalupe_1 | 2020-02-21 05:45:22.082 INFO 1 --- [           main]
o.s.b.a.e.mvc.EndpointHandlerMapping : Mapped "{[/env/reset],methods=[POST]}" onto public java.util.Map<java.lang.String, java.lang.Object> org.springframework.cloud.context.environment.EnvironmentManagerMvcEndpoint.reset()
api-seguros-guadalupe-administracion_1 | 2020-02-21 05:45:22.342 INFO 1 --- [           main]
o.s.j.e.a.AnnotationMBeanExporter : Registering beans for JMX exposure on startup
api-seguros-guadalupe-administracion_1 | 2020-02-21 05:45:22.414 INFO 1 --- [           main]
o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'configurationPropertiesRebinder' has been autodetected for JMX exposure
api-seguros-guadalupe-administracion_1 | 2020-02-21 05:45:22.425 INFO 1 --- [           main]
o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'refreshEndpoint' has been autodetected for JMX exposure
api-seguros-guadalupe-administracion_1 | 2020-02-21 05:45:22.430 INFO 1 --- [           main]
o.s.j.e.a.AnnotationMBeanExporter : Bean with name 'environmentManager' has been autodetected for JMX exposure
api-seguros-guadalupe-administracion_1 | 2020-02-21 05:45:22.433 INFO 1 --- [           main]
o.s.j.e.a.AnnotationMBeanExporter : Every 2.0s: docker images
REPOSITORY TAG
jovaniac/api-seguros-guadalupe-administracion-http 0.0.1-snapshot
jovaniac/api-seguros-guadalupe-http 0.0.1-snapshot
<none>
{} api clientes_mock.postman_collection <none>
{} api creditos_mock.postman_collection <none>
{} api empleados_mock.postman_collection <none>
{} api_gateway_apigee.postman_collection <none>
lab2_comunicacion_http_postgres latest
jovaniac/cliente-ms-compose 0.0.1-snapshot
```

The terminal also lists several Docker containers:

- REPOSITORY TAG
- jovaniac/api-seguros-guadalupe-administracion-http 0.0.1-snapshot
- jovaniac/api-seguros-guadalupe-http 0.0.1-snapshot
- <none>
- {} api clientes\_mock.postman\_collection <none>
- {} api creditos\_mock.postman\_collection <none>
- {} api empleados\_mock.postman\_collection <none>
- {} api\_gateway\_apigee.postman\_collection <none>
- lab2\_comunicacion\_http\_postgres latest
- jovaniac/cliente-ms-compose 0.0.1-snapshot

# Explicación - Comunicación HTTP y Shared Database

- 1.- Migracion realizada para api seguros**
- 2.- Tabla de seguros sin datos**
- 3.- Tabla de catalogos de seguros poblada**

- 1.- Migracion realizada para api-administracion seguros**
- 2.- Tabla de autorizaciones sin datos**

- 1.- Consulta de catalogos**
- 2.- Crear seguro**
- 3.- Solicitud de autorizacion**

Postman:  
`/comunicacion_http/01_API Seguros Catalogos GET 200`  
`/comunicacion_http/01_API Seguros POST 201`



# Explicación - Comunicación HTTP y Shared Database

- 1.- Se dio alta el seguro en estatus sin autorizacion
- 2.- El microservicio de seguros, llamo al de autorizaciones
- 3.- Se dio de alta la autorizacion pero no se ha procesado su estatus

MySQL Workbench Screenshot:

Database Navigator Projects

catalogo\_seguros seguros autorizaciones

Properties Data ER Diagram PostgreSQL-seguros\_guadalupe\_administracion seguros\_guadalupe\_administracion Schemas public

autorizaciones Enter a SQL expression to filter results (use Ctrl+Space)

Grid

|   | ingresosAnuales                               | direccion | autorizacion                        |
|---|---|-----------|-------------------------------------|
| 1 | "100.0,"10 de abril, #20, cuernavaca morelos" | false     | Fri Feb 21 06:04:18 UTC 2020 [NULL] |

Text

```
Pretty Raw Preview Visualize JSON
1 {
  "folio": "d9439a7e-ed2b-4acf-a2d9-69b95e13751",
  "mensaje": "Datos de la poliza",
  "seguros": [
    {
      "id": "d9439a7e-ed2b-4acf-a2d9-69b95e13751",
      "plazo": 5.5,
      "precioPoliza": 5.1,
      "tipoCobertura": 1,
      "vencimiento": "2030-01-01",
      "sumaAsegurada": 300.0,
      "nombre": "Ricardo",
      "apellidoPaterno": "Alvar",
      "apellidoMaterno": "Torre",
      "fechaNacimiento": "01-01-1990",
      "sexo": "M",
      "ingresosAnuales": 400.0,
      "direccion": "Argentina",
      "autorizacion": null
    }
  ],
  "autorizaciones": [
    {
      "folio": "b0b1773f-2f5c-42a7-8891-e202721ac010",
      "mensaje": "La autorizacion se genero correctamente"
    }
  ]
}
```

| direccion | autorizacion |
|-----------|--------------|
| [NULL]    | [NULL]       |

Postman:

/comunicacion\_http/05\_API Adm Polizas Autorizaciones

Postman

History Collections APIs

02\_API Seguros GET 200 By Id

GET localhost:8081/seguros/v1/34e6288b-4554-4eb2-88d2-cc0ed118546

Params Authorization Headers Body Pre-request Script Tools Settings Cookies Code

Query Params:

Body Cookies Headers Test Results Status: 200 OK Time: 167ms Size: 5173 Save Response

```
Pretty Raw Preview Visualize JSON
1 {
  "id": "34e6288b-4554-4eb2-88d2-cc0ed118546",
  "plaza": 1.5,
  "precioPoliza": 6.4,
  "tipoCobertura": 1,
  "vencimiento": "2021-01-01",
  "sumaAsegurada": 250.0,
  "nombre": "Selene",
  "apellidoPaterno": "May",
  "apellidoMaterno": "Can",
  "fechaNacimiento": "01-08-1990",
  "sexo": "M",
  "ingresosAnuales": 100.0,
  "direccion": "10 de abril, #20, cuernavaca morelos",
  "autorizacion": null
}
```

# Explicación - Comunicación HTTP y Shared Database

- 1.- Se realiza la autorización en el api de administración de autorizaciones, se cambian los estatus y fechas de autorización
- 2.- El microservicios de administracion de autorizaciones invoca al microservicio de api seguros y actualiza su estatus indicando que el seguro fue autorizado

The image shows two screenshots of a PostgreSQL database interface. The top screenshot displays the 'autorizaciones' table with two rows. The first row has 'estatus' set to 'false' and 'fecha\_apertura' set to 'Fri Feb 21 06:04:18 UTC 2020'. The second row has 'estatus' set to 'true' and 'fecha\_apertura' set to 'Fri Feb 21 06:22:33 UTC 2020'. Red arrows point from the 'estatus' column of these two rows to the corresponding columns in the bottom screenshot. The bottom screenshot displays the 'seguros' table with two rows. Both rows have 'autorizacion' set to 'true'. Red arrows point from the 'autorizacion' column of both rows to the 'autorizacion' column in the top screenshot.

| id | apellido_paterno | apellido_materno | fecha_nacimiento | sexo | ingresos_anuales | direccion                            | autorizacion |
|----|------------------|------------------|------------------|------|------------------|--------------------------------------|--------------|
| 1  | Cua              | Cuan             | 01-08-1990       | M    | 100              | 10 de abril, #20, cuernavaca morelos | false        |
| 2  | Cua              | Cuan             | 01-08-1990       | M    | 100              | 10 de abril, #20, cuernavaca morelos | true         |

The image shows a Postman interface with several API requests listed in the sidebar:

- 05\_API Adm Polizas get 200 Autorizaciones
- 02\_API Seguros GET 200 By Id
- 04\_API Adm Polizas PUT 200 Confirmacion
- 01\_API Seguros POST 201
- 02\_API Seguros GET 200 By Id
- 03\_API Seguros Catalogos GET 200
- PUT: 04\_API Adm Polizas PUT 200 Confirmacion
- GET: 05\_API Adm Polizas get 200 Autorizaciones
- GET: 06\_API Adm Polizas get 200 Autorizaciones ById
- comunicacion\_gateway
- monitoreo

The 'PUT: 04\_API Adm Polizas PUT 200 Confirmacion' request is selected. Its body is shown in JSON format:

```
[{"idAutorizacion": "bb3a33c2-9395-4239-a99e-acd10e97442", "confirmacion": "true"}]
```

The response tab shows the following JSON data:

```
{ "folio": "55e4c92b-5dc8-4f1d-8b2d-84b849296877", "mensaje": "Confirmacion de seguro, favor de validar estatus" }
```

# Explicación - Spring Boot + OpenFeign

## Configuración del microservicio de seguros

### Anotaciones necesarias:

```
@SpringBootApplication  
@EnableFeignClients  
@EnableCircuitBreaker  
@EnableHystrixDashboard  
public class SegurosApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(SegurosApplication.class, args);  
    }  
}
```

**./run-local.sh**

**sudo rm -rf blockstorage/  
docker stop <id container>**

```
@FeignClient(name = "api-seguros-guadalupe-administracion", url = "api-seguros-guadalupe-administracion:8082", fallback=AutorizacionesServiceClientFallback.class)  
public interface AdministracionSegurosServiceRemote {  
  
    @RequestMapping(method = POST, value = "administracion/v1/autorizaciones", produces = "application/json")  
    AutorizacionesResumenDto solicitaAutorizacion(@RequestBody SegurosDto datosSeguro);  
}
```

## Configuración del microservicio de administracion autorizaciones

```
@FeignClient(name = "api-seguros-guadalupe", url = "api-seguros-guadalupe:8081", fallback=SegurosServiceClientFallback.class)  
public interface SegurosServiceRemote {  
  
    @RequestMapping(method = PUT, value = "seguros/v1/confirmaciones", produces = "application/json")  
    ConfirmacionResumenDto confirmacionesPoliza(@RequestBody ConfirmacionDto confirmaciones);  
}
```

**Servicios de docker-compose, para comunicarse**  
url = "api-seguros-guadalupe-administracion:8082  
url = "api-seguros-guadalupe:8081"



# Explicación - Spring Boot + OpenFeign + Hystrix

Qué pasa si alguno de los microservicios falla?

**docker ps**

**docker stop**

**Validar bases de datos de ambos microservicios**

**Levantar autorizaciones en STS**

**fallback=AutorizacionesServiceClientFallback.class**

```
public class AutorizacionesServiceClientFallback implements Adminis-
tracionSegurosServiceRemote {
```

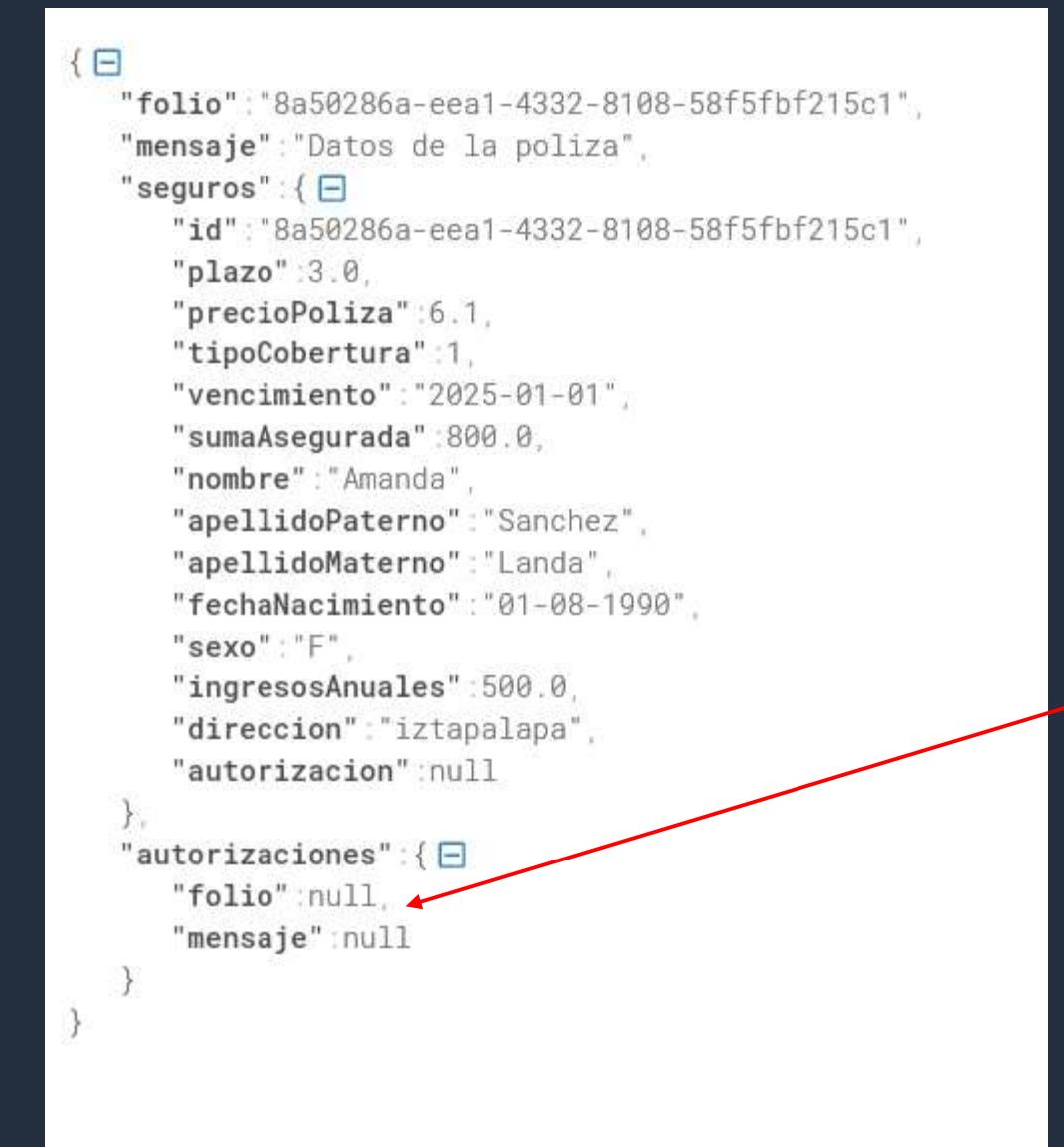
```
    @Override
```

```
    public AutorizacionesResumenDto solicitaAutorizacion(SegurosDto
datosSeguro) {
```

```
        // TODO Auto-generated method stub
```

```
        return new AutorizacionesResumenDto();
```

```
}
```



# Spring Boot + OpenFeign + Hystrix

## Spring Cloud Feign

- La anotación `@FeignClient` soporta::
- Atributo “url” donde se defina la URL “hard-codeada” del servicio.
- `@FeignClient(url = "localhost:9091/my-context")`
- Atributo “name” o “value” donde se defina el “clientId” o “spring.application.name” del proveedor del servicio, proporcionado por Eureka.
  - `@FeignClient(name = "my-service")`
  - `@FeignClient("my-service")` o `@FeignClient(value = "my-service")`

## La anotación `@FeignClient` soporta :

- Atributo “path” donde se define el “context-path” del proveedor del Servicio.

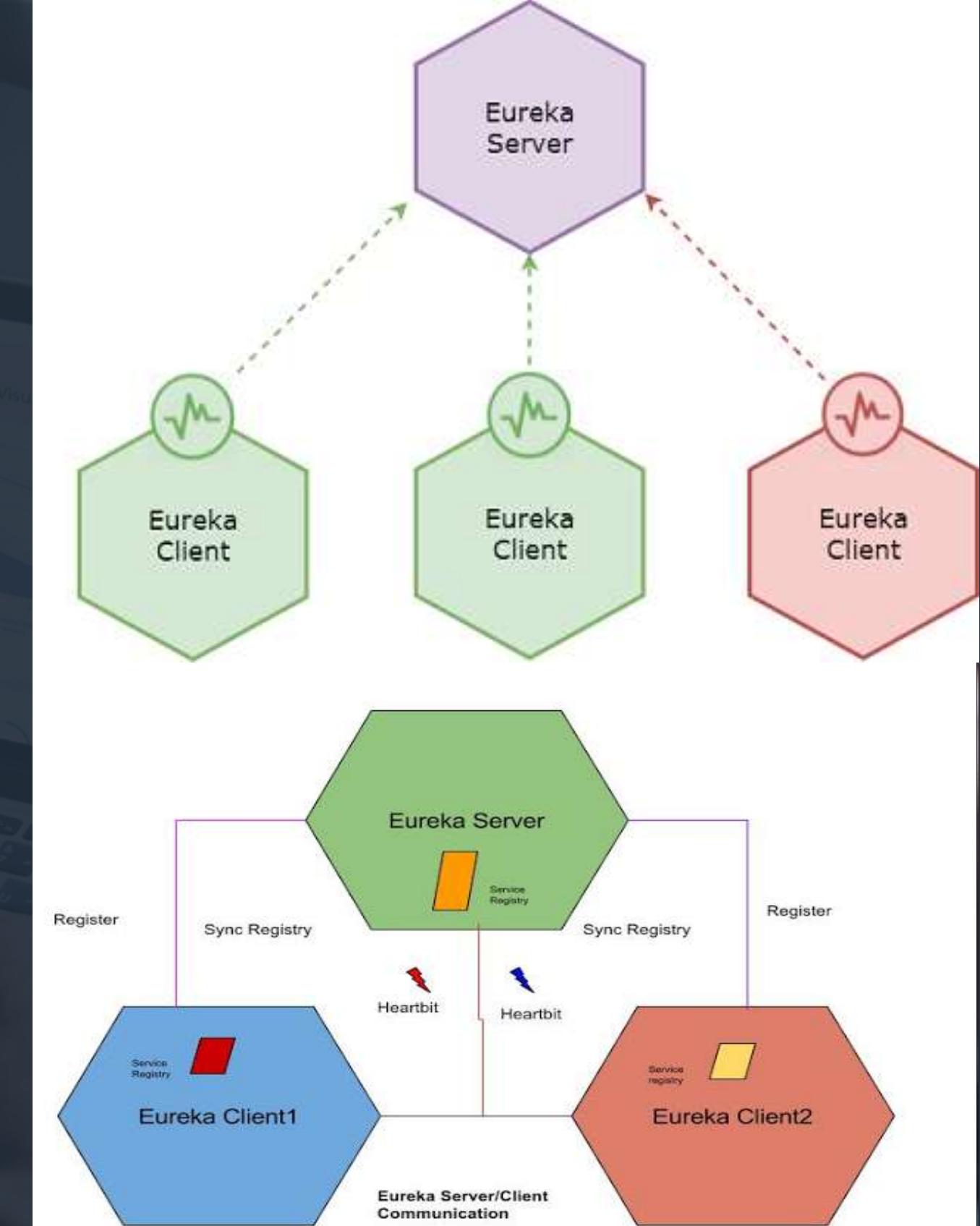
- `@FeignClient(name = "my-service", path = "my-context")`
- `@FeignClient(name = "my-service/my-context")` (no recomendado)
- `@FeignClient` soporta “place-holders”:
- `@FeignClient("${service-name:some-service-id}")`

Si utiliza Eureka como “service registry” y “service discovery” Feign (a través de Ribbon) utiliza el “servicId” de los microservicios auto-registrados en Eureka.



# Sección I

## Spring Cloud Netflix OSS - Discovery



# Hands-ON Comunicación HTTP en Microservicios por Service Discovery - Eureka Server y Patrón Shared Database

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab4\\_service\\_discovery](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab4_service_discovery)

ABRIR STS E IMPORTAR PROYECTOS

# Pattern Service Discovery

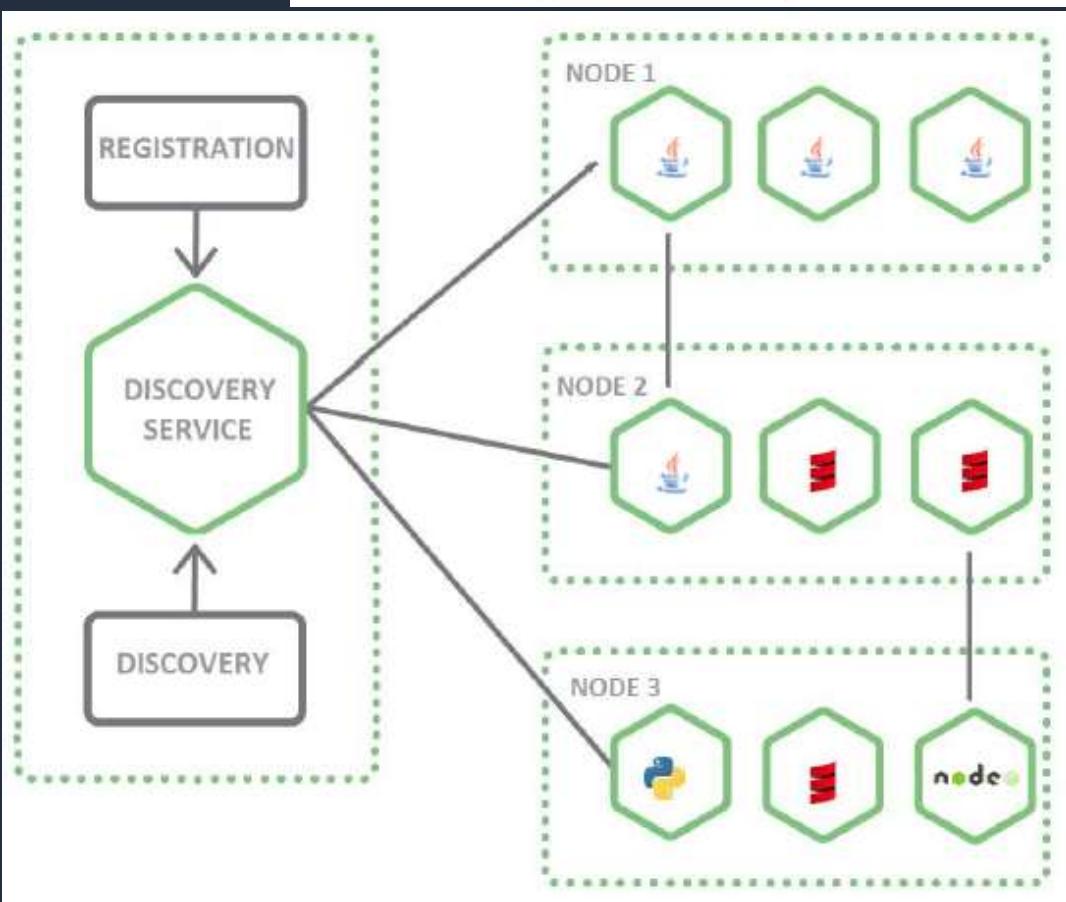
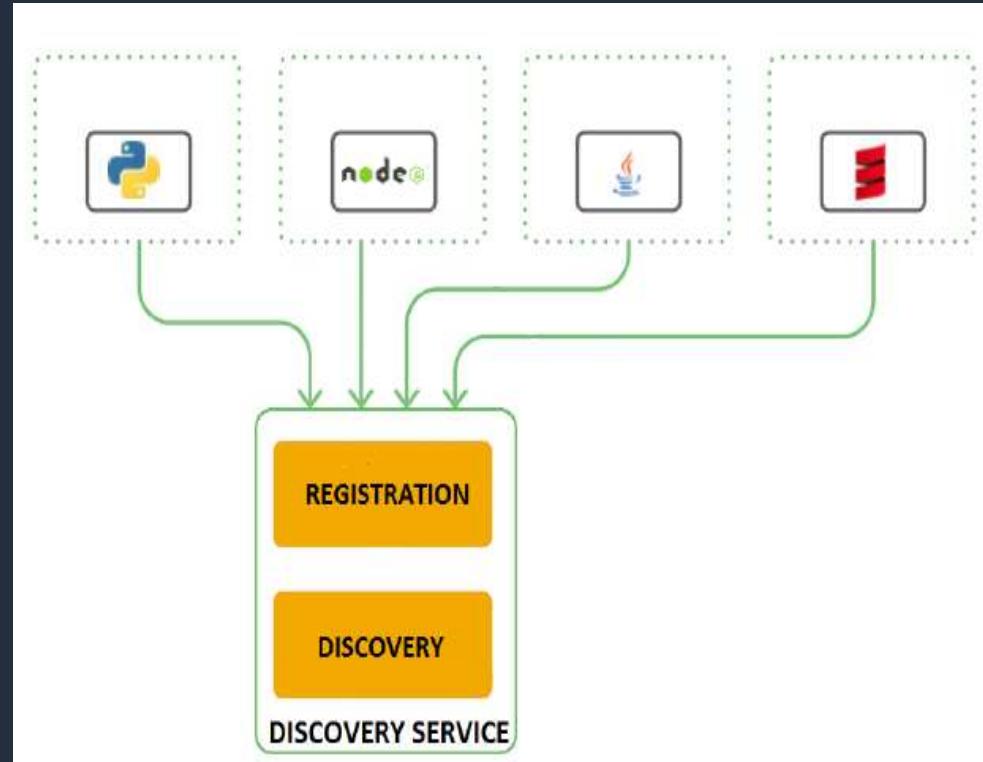
- En arquitecturas orientadas a microservicios típicamente las aplicaciones resultan en un gran número de microservicios operativos.

Éstos microservicios deben ser interconectados entre sí para realizar llamadas entre servicios.

En una arquitectura de microservicios, los servicios son “servidores” o “proveedores” del servicio que implementan sin embargo, a su vez pueden ser “clientes” o “consumidores” de servicios que otros microservicios exponen.

Un servicio de “descubrimiento de servicios” o “service discovery” es un componente fundamental en arquitecturas distribuidas que permite “encontrar” o “descubrir” la ubicación en la red de los servicios desplegados.

Los servicios o microservicios pueden desplegarse en múltiples IPs, hostnames y puertos, lo que ocasiona que realizar el seguimiento de conocer cuál es la ubicación en la red de cada servicio sea **difícil**.



# Pattern Service Discovery – Eureka Server

Spring Cloud Eureka.

Componente de registro y descubrimiento de servicios parte del stack tecnológico de Spring Cloud Netflix OSS.

Eureka provee el servicio de registro y descubrimiento o “lookup” de Servicios.

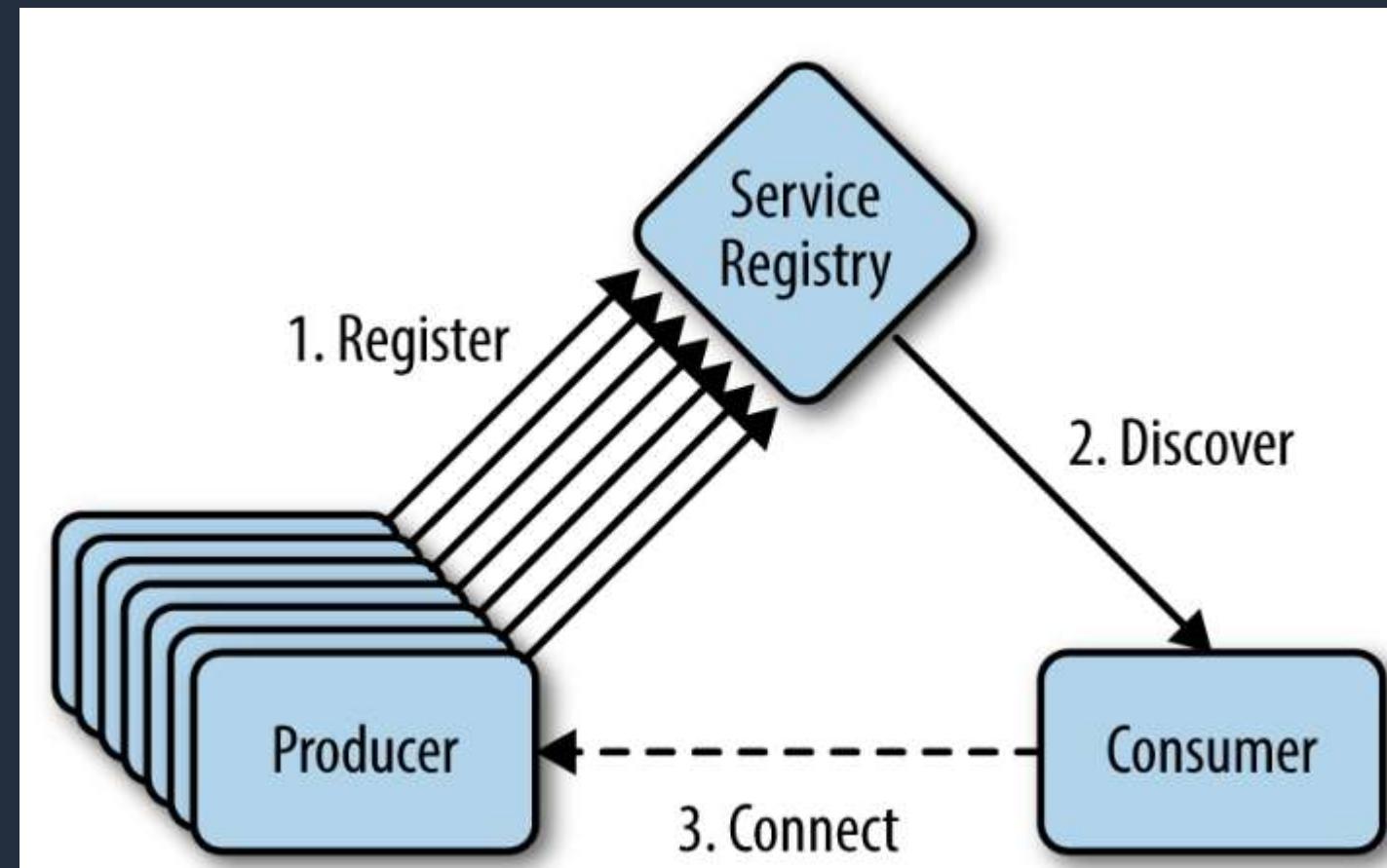
Permite mantener alta disponibilidad del servicio mediante ejecutar múltiples instancias de Eureka.

Permite replicar el estado de los servicios registrados entre las múltiples instancias de Eureka (cluster).

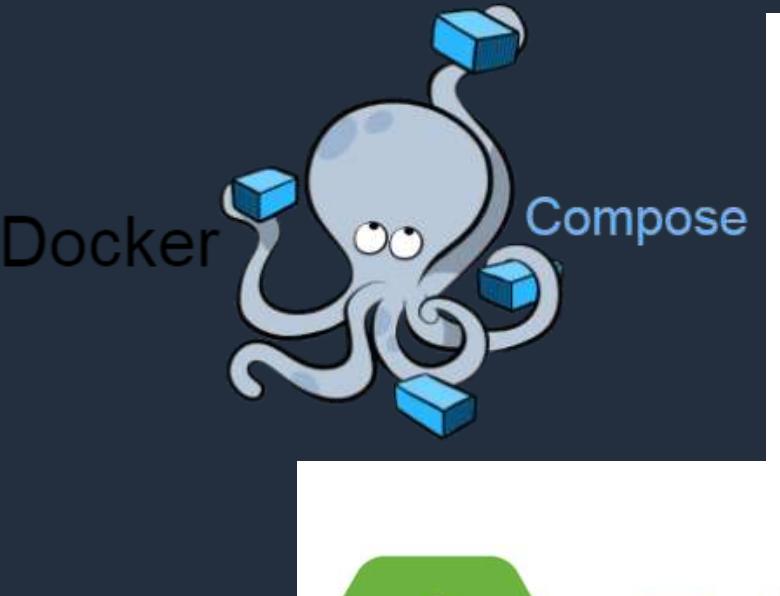
Los servicios/microservicios “servidores” se registran en Eureka para poder ser “descubiertos” por otros servicios “consumidores” o “clientes” de los servicios que exponen.

Envian metadatos como hostname, puerto, URI para verificación de “health check”, entre otros parámetros útiles para su Monitoreo, **utilizan las métricas y actuadores de Spring Boot Actuator**.

Los servicios envian “hearthbeats” al servidor Eureka para verificar su “vitalidad” o “liveness”.



Tecnologías:  
Spring Boot + JPA + PostgreSQL + Docker + Migration + OpenShift / Netflix OSS + Docker-compose + Flyway  
Descripción de microservicios:  
1.- API SEGURO  
2.- AUTORIZACION / AUTORIZACION DE AUTORIZACIONES  
3.- BASE DE DATOS  
4.- MICROSERVICIOS EN LA EDAD I  
5.- EUREKA, SEGURO, GUARDIA, API  
6.- EUREKA, GUARDIA, API, ADMINISTRACION  
7.- EUREKA SERVER



## Ejecución - Discovery – Eureka Server

### Ejecución: ./run-local.sh

```
#!/usr/bin/env bash
```

```
cd discovery-server  
./generalImagen.sh
```

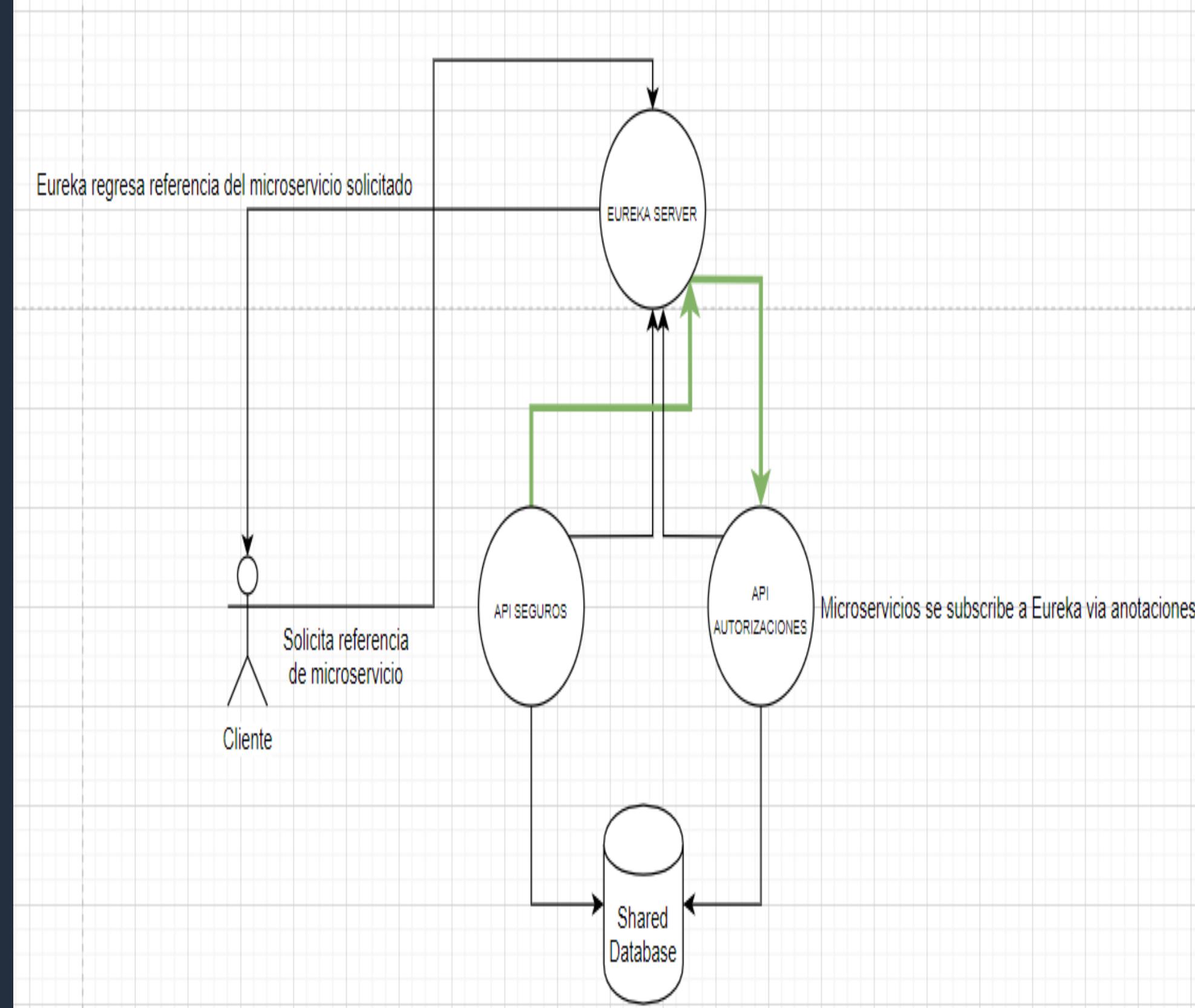
```
cd ../  
cd api-seguros  
./generalImagen.sh
```

```
cd ../  
cd api-adm-autorizaciones  
./generalImagen.sh
```

```
cd ..
```

```
docker-compose up --build
```

```
docker-compose stop  
docker-compose kill  
docker-compose rm -f
```



# Explicación - Service Discovery+ OpenFeign

Configuracion de Feign para usar Service Discovery Eureka por id de microservicio

```
@FeignClient(name = "api-seguros-guadalupe-administracion", fallback =  
AutorizacionesServiceClientFallback.class, configuration =  
ClientRemoteConfig.class)
```

```
public interface AdministracionSegurosServiceRemote {
```

```
    @RequestMapping(method = POST, value = "administracion/v1/autorizaciones",  
produces = "application/json") AutorizacionesResumenDto  
solicitaAutorizacion(@RequestBody SegurosDto datosSeguro);  
}
```

**Apuntamos al service Discovery**

```
name = "api-seguros-guadalupe"  
name = "api-seguros-guadalupe-administracion"
```

**Configuracion de cada microservicio para subscribirse a Eureka Server**

```
eureka:  
client:  
serviceUrl:  
    defaultZone: http://127.0.0.1:8761/eureka/  
instance:  
    hostname: localhost  
    prefer-ip-address: true #registra ip no hostname  
    leaseRenewalIntervalInSeconds: 1  
    leaseExpirationDurationInSeconds: 2  
    metadata-map:  
    instanceId: ${instance.id}
```

**Configuracion main para spring boot**

```
@EnableDiscoveryClient  
@EnableFeignClients
```

```
@SpringBootApplication  
@EnableFeignClients  
@EnableCircuitBreaker  
@EnableHystrixDashboard  
@EnableDiscoveryClient  
public class SegurosApplication {
```

```
    public static void main(String[] args) {  
        SpringApplication.run(SegurosApplication.class, args);  
    }  
}
```

**Bootstrap.yaml por cada microservicio para indicar nombre Para Eureka Server**

```
spring:  
application:  
    name: api-seguros-guadalupe
```

```
spring:  
application:  
    name: api-seguros-guadalupe-administracion
```



**http://localhost:8761/**

## Dependencias

**compile('org.springframework.cloud:spring-cloud-starter-eureka')**

## IMPORTARLO DESDE STS AL DETENER EL CONTENEDOR

The screenshot shows the Spring Eureka dashboard. At the top, it displays 'System Status' with environment 'test', data center 'default', current time '2019-05-02T22:31:26+0000', uptime '00:25', lease expiration enabled 'true', renew threshold '5', and renew last min '91'. Below this is the 'DS Replicas' section, which lists two instances registered with Eureka: 'API-SEGUROS-GUADALUPE' and 'API-SEGUROS-GUADALUPE-ADMINISTRACION'. Both instances are in the 'UP [1] - developer/api-seguros-guadalupe:8081' state.

Postman:

/comunicacion\_http\_service\_discovery/01\_API Seguros Catalogos GET 200  
/comunicacion\_http\_service\_discovery/01\_API Seguros POST 201  
/comunicacion\_http\_service\_discovery/04\_API Adm Polizas PUT 200  
Confirmaciones

**./run-local.sh**  
**sudo rm -rf blockstorage/**  
**docker stop <id container>**

**sudo lsof -i -P -n | grep LISTEN**



## El servicio de “service registry” de Eureka no persiste el estado o registro de los servicios auto-registrados.

- Todos los servicios auto-registrados deben enviar “**heartbeats**” para mantener vivo y al día su registro; por tanto dicho registro puede mantenerse en memoria.
- Los clientes que “descubren” los servicios auto-registrados sobre el servicio de “service discovery” de Eureka mantienen el registro en memoria para evitar obtener el registro en cada petición que deban ejecutar hacia algún servicio.

Por default, cada instancia de Eureka es un servidor y también un cliente que requiere al menos una URL, de un servidor Eureka, para localizar a un “peer” (par o nodo “igual”).

- Es posible no definir una URL de un servidor Eureka par (“peer”), sin embargo, la consola enviará muchos errores debido a que Eureka está diseñado para desplegarse en alta disponibilidad, auto-registrándose y sincronizando el estado de los servicios auto-registrados con los demás “peers” en el cluster.

Cada instancia, aplicativo o microservicio que se auto-registra en el servidor Eureka debe enviar “heartbeats” para mantener vivo y al día su registro.

```
# Eureka properties
# Eureka default port 8761
server.port=9099
eureka.client.register-with-eureka=false
eureka.client.fetch-registry=false
eureka.client.service-url.defaultZone=http://localhost:9099/eureka
```

```
# Eureka YAML properties
# Eureka default port 8761
server:
port: 9099
eureka:
client:
register-with-eureka: false
fetch-registry: false
service-url:
defaultZone: http://localhost:9099/eureka
```

```
# Spring Cloud Eureka Client
management:
endpoint:
health:
enabled: true
shutdown:
enabled: true
endpoints:
web:
exposure:
Include: '*'
```

Spring Cloud Eureka Client.  
- Por default Spring Cloud Eureka utiliza Spring Boot Actuator para exponer el actuador “health”.  
- Es necesario habilitar el actuador “health” y exponer los “actuators” vía web.



# Sección I

## Spring Cloud NetflixOSS – Metricas Centralizadas

# Hands-ON Metricas Centralizadas + Service Discovery + Microservicios

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab5\\_centralized\\_metrics](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab5_centralized_metrics)

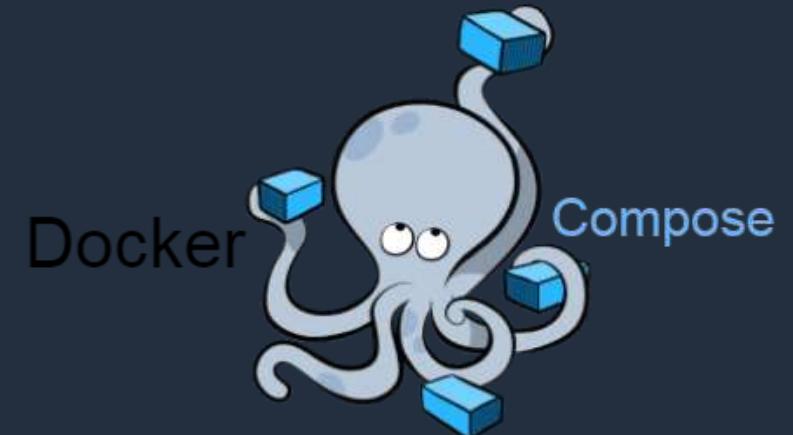
ABRIR STS E IMPORTAR PROYECTOS

# Microservicios HTTP + Eureka Server + Monitoreo

Tecnología:  
Spring Boot + Docker + OpenFeign Netflix Oss + Docker-compose + InfluxDB + Grafana

Descripción de microservicios:

- 1.- API CLIENTES
- 2.- API CREDITOS
- 3.- INFLUX DB - BASE DE DATOS DE SERIES DE TIEMPO
- 4.- EUREKA SERVER
- 5.- GRAFANA SERVER



Ejecución - Discovery –  
Eureka Server

## Ejecución: ./run-local.sh

```
#!/usr/bin/env bash
```

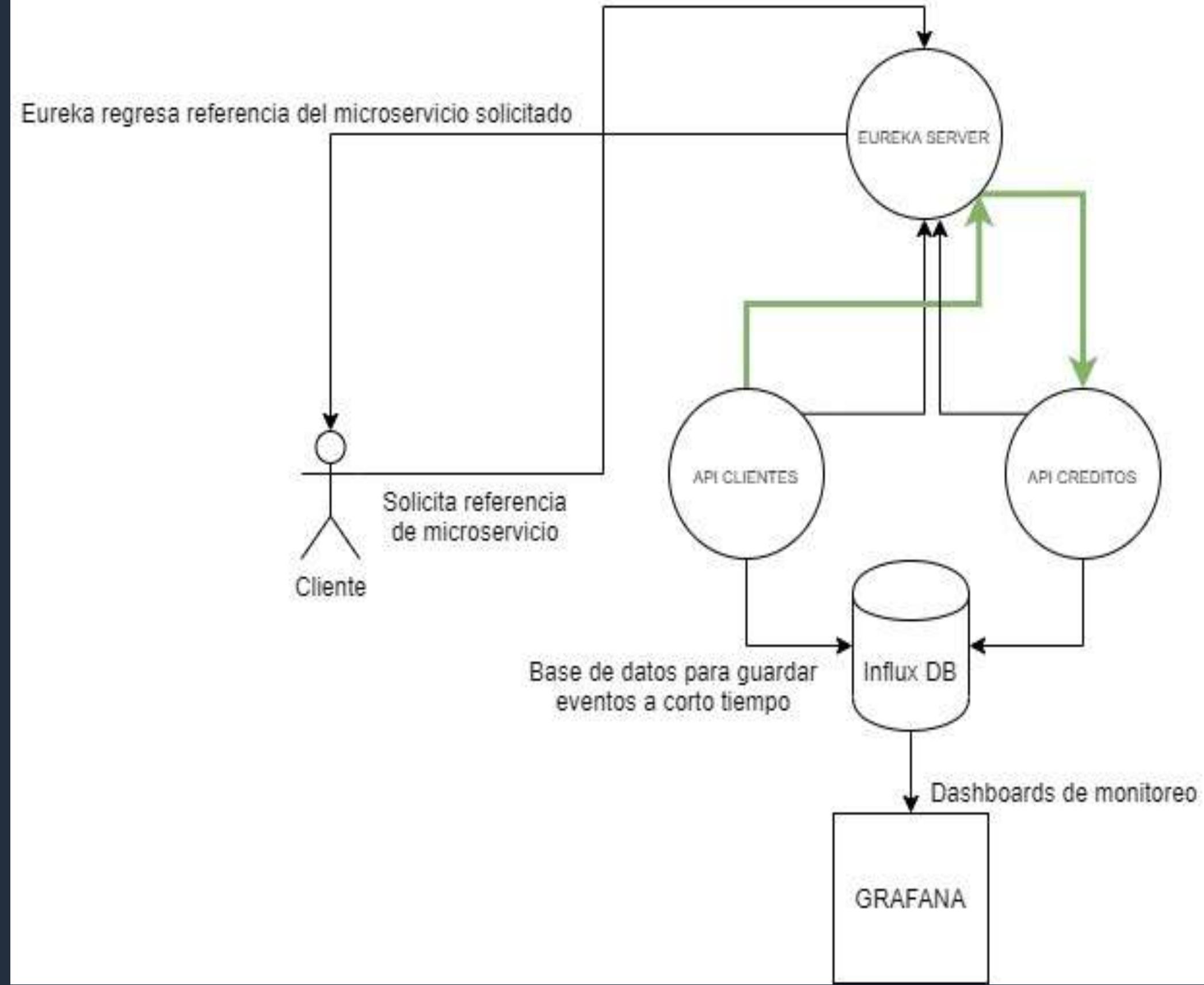
```
cd discovery-server  
./generalImagen.sh
```

```
cd ../  
cd api-seguros  
./generalImagen.sh
```

```
cd ../  
cd api-adm-autorizaciones  
./generalImagen.sh
```

```
cd ..  
  
docker-compose up --build
```

```
docker-compose stop  
docker-compose kill  
docker-compose rm -f
```



# Explicación - Service Discovery+ OpenFeign

## 1.- Configuracion main para spring boot

```
@SpringBootApplication  
@EnableFeignClients  
@EnableDiscoveryClient  
public class ClientesServiceApplication {  
  
    public static void main(String[] args) {  
        SpringApplication.run(ClientesServiceApplication.class, args);  
    }  
}
```

## 3.- Configuracion de cada microservicio para suscribirse a Eureka Server

```
@FeignClient(name = "ms-creditos",  
configuration = ClientRemoteConfig.class)  
public interface CreditosServiceRemoteClient {  
    @RequestMapping(method = POST, value = "api/v1/creditos")  
    ResumenCredito guardarCredito(@RequestBody Credito credito);  
}
```

## 2.- Configuración de InfluxDB para persistencia

```
@Configuration  
public class InfluxDBConfig {  
  
    private static final Logger log = Logger.getLogger(InfluxDBConfig.class);  
    @Value("${influx.host}")  
    String host;  
    @Value("${influx.port}")  
    Integer port;  
    @Value("${influx.user}")  
    String user;  
    @Value("${influx.password}")  
    String password;  
    @Value("${influx.database}")  
    String dbName;
```

## 4.- Configuración de Eureka Server

```
eureka:  
    client:  
        serviceUrl:  
            defaultZone: http://127.0.0.1:8761/eureka/  
    instance:  
        hostname: localhost  
        prefer-ip-address: true #registra ip no hostname  
        leaseRenewalIntervalInSeconds: 1  
        leaseExpirationDurationInSeconds: 2  
    metadata-map:  
        instanceId: ${instance.id}
```



# Explicación - Service Discovery+ OpenFeign

## 5.- Bootstrap.yaml por cada microservicio para indicar nombre para Eureka Server

```
spring:  
  application:  
    name: ms-clientes
```

```
spring:  
  application:  
    name: ms-creditos
```

## 6.- Escribir datos en los actuators de spring

```
@Autowired  
private CounterService counterService;  
  
counterService.increment("banco.poder.clientes.nuevos");
```

## 7.- Dependencias del proyecto

```
dependencies {  
  compile('org.springframework.boot:spring-boot-starter-actuator')  
  compile('org.springframework.boot:spring-boot-starter-web')  
  compile('org.springframework.cloud:spring-cloud-starter-eureka')  
  compile('org.springframework.cloud:spring-cloud-starter-hystrix')  
  compile('org.springframework.cloud:spring-cloud-starter-openfeign')  
  runtime('org.jolokia:jolokia-core') //expone como endpoint rest  
  compile('org.influxdb:influxdb-java:2.7') //persistencia influxdb  
}
```

## 8.- Configuración de actuator de spring - metrics

```
spring:  
  metrics:  
    export:  
      delay-millis: 1000  
      timeUnit: seconds  
      #interval: 1  
      #timeUnit: seconds  
      prefix: cliente  
      logger: com.microservicios.monitoreo  
      #includes: heap.used,heap.committed,mem,mem.free,threads,datasource.primary.activ  
e,datasource.primary.usage,gauge.response.persons,gauge.response.persons.id,gauge.re  
sponse.persons.remove,gauge.firstservice
```

Conexión a influxDB, indicamos BD  
influx:

```
  host: localhost  
  port: 8086  
  user: admin  
  password: admin  
  database: grafana-clientes
```

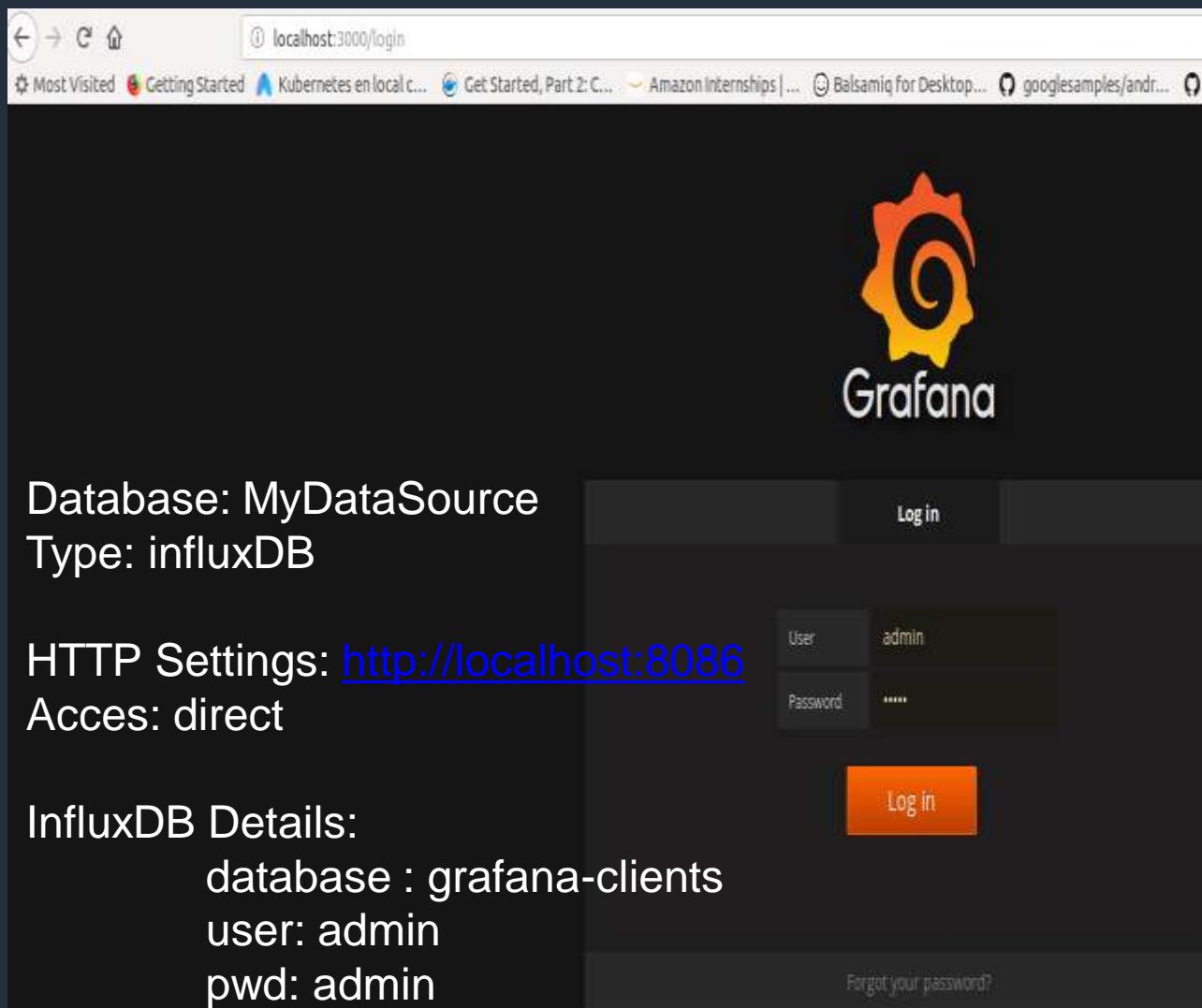


# Explicación - Grafana

localhost: 3000

Usr: admin

Pwd: admin

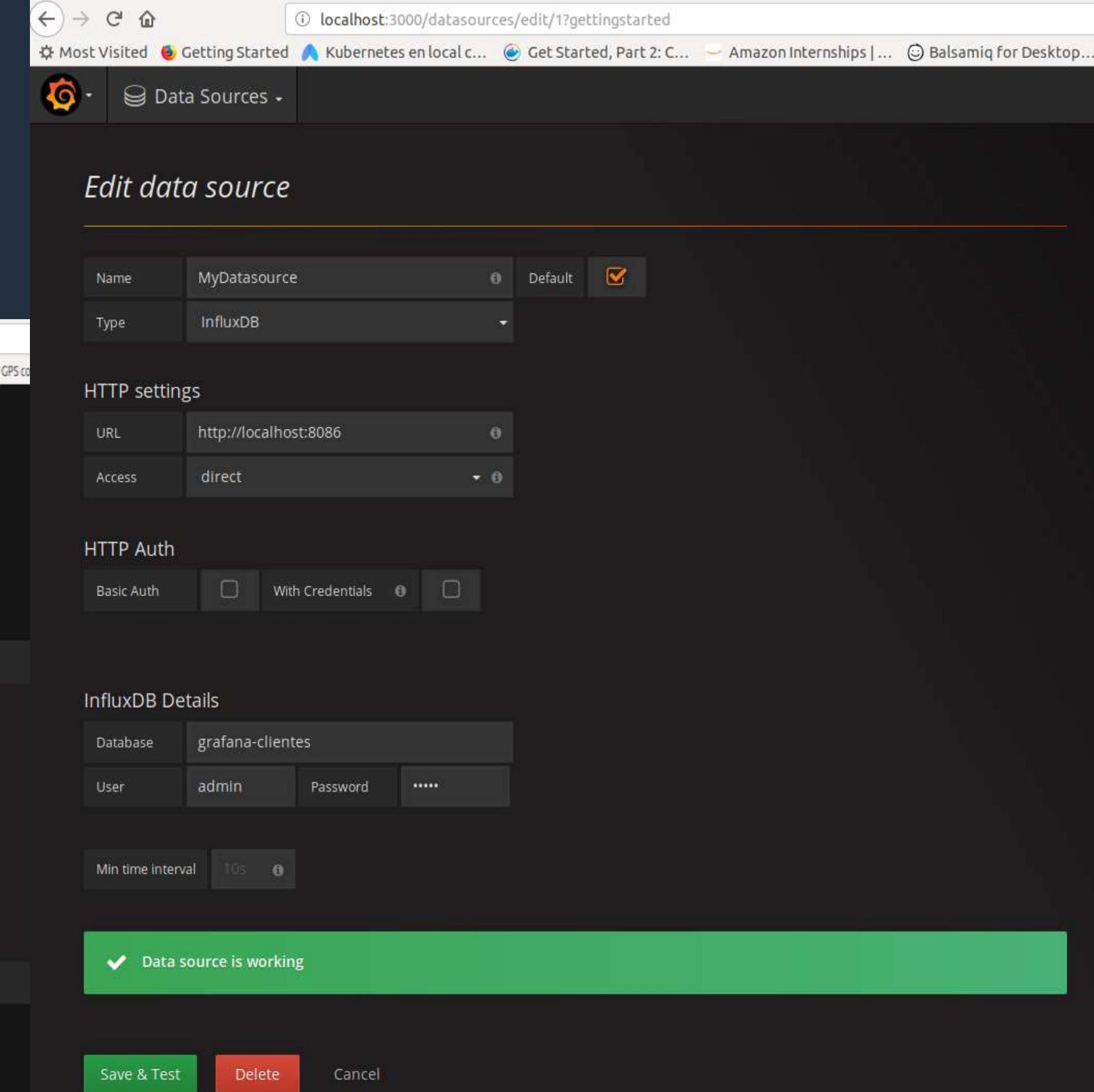


The screenshot shows the Grafana login interface. At the top, there's a navigation bar with links like 'Most Visited', 'Getting Started', 'Kubernetes en local c...', 'Get Started, Part 2: C...', 'Amazon Internships | ...', 'Balsamiq for Desktop...', 'googlesamples/andr...', and 'GPS co...'. Below the navigation is the Grafana logo and the word 'Grafana'. The main area has a 'Log in' button at the top right. Below it are fields for 'User' (set to 'admin') and 'Password' (set to '.....'). A large orange 'Log in' button is centered below these fields. At the bottom left is a 'Forgot your password?' link.

Database: MyDataSource  
Type: influxDB

HTTP Settings: <http://localhost:8086>  
Acces: direct

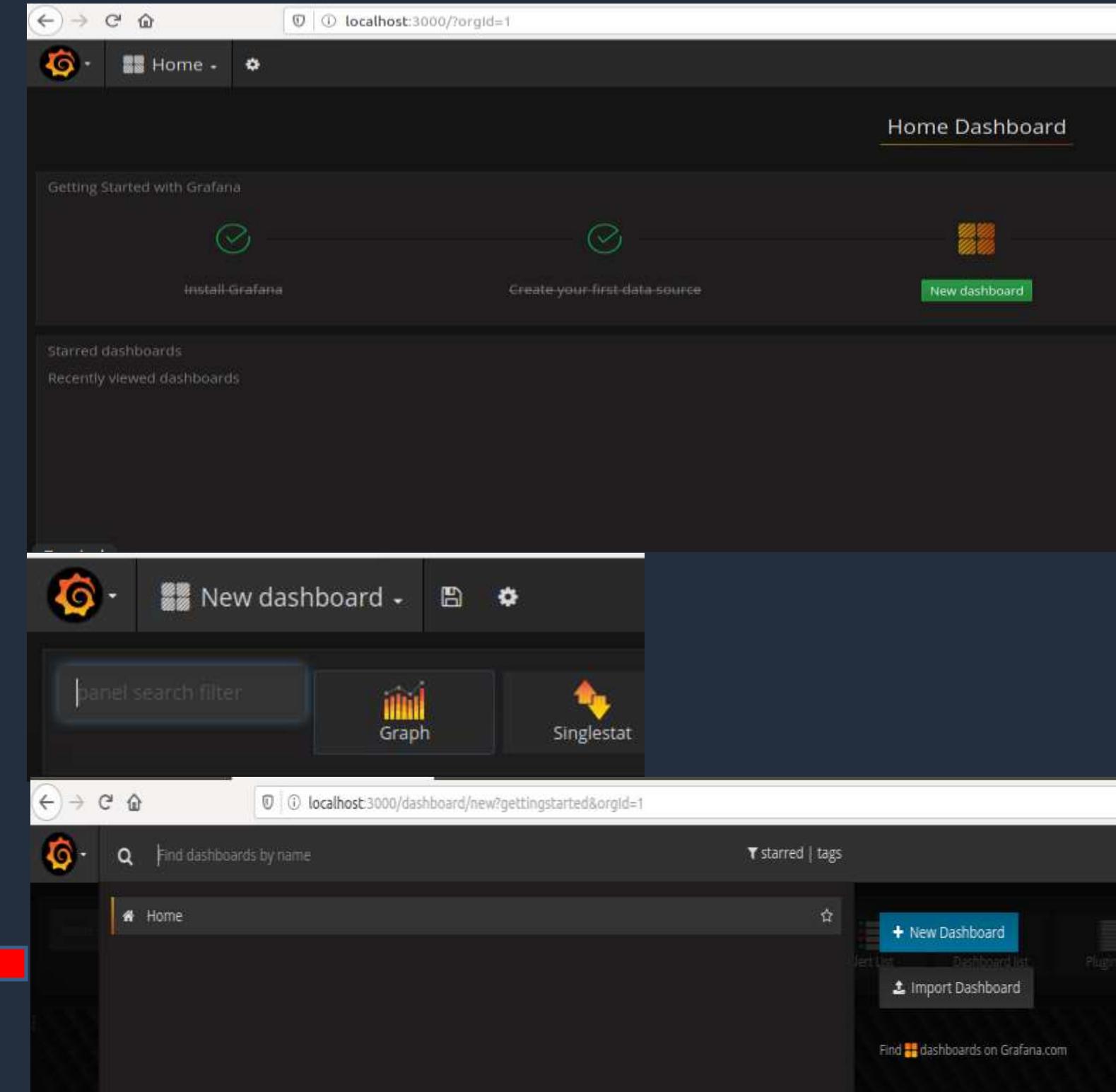
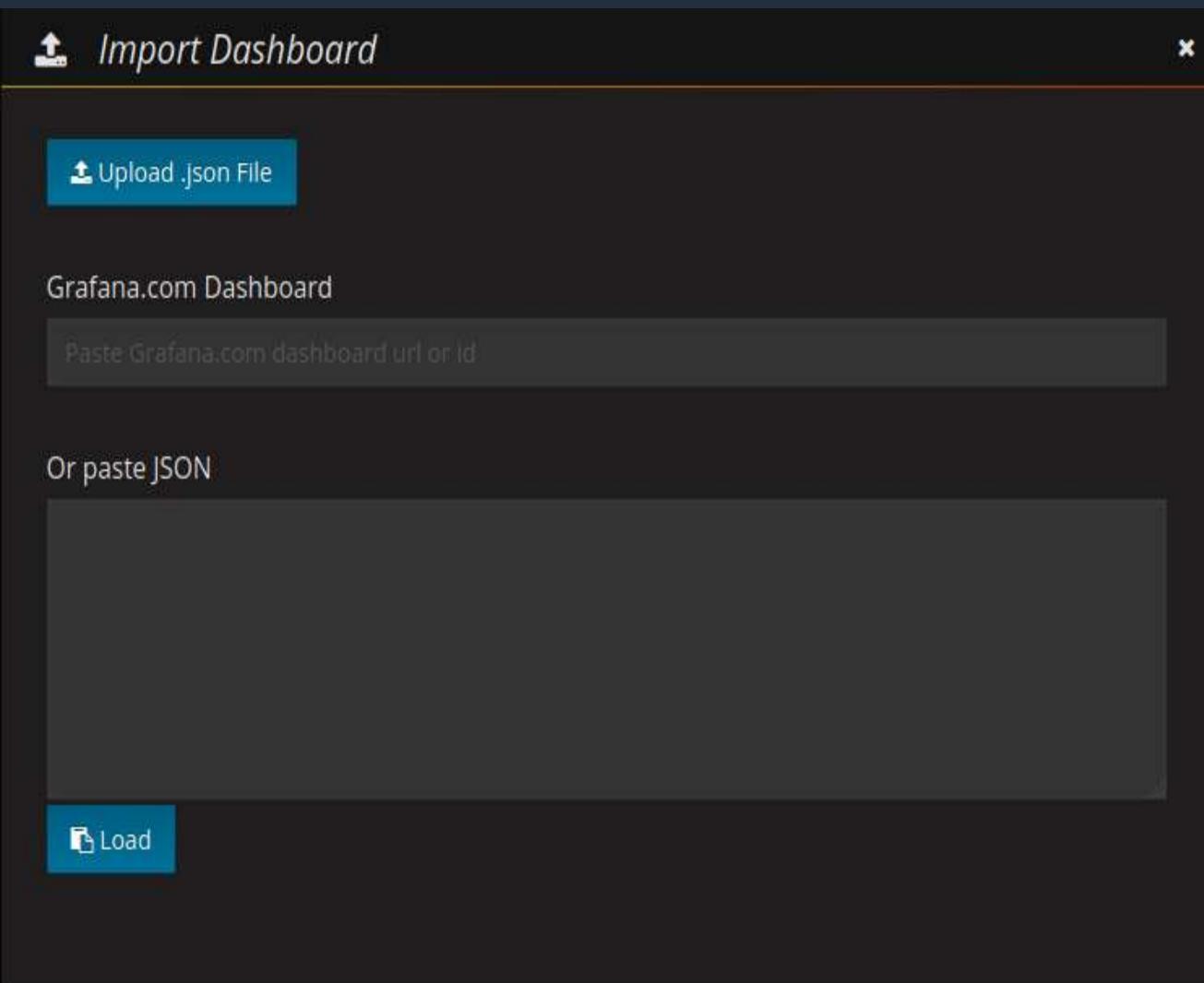
InfluxDB Details:  
database : grafana-clients  
user: admin  
pwd: admin



The screenshot shows the 'Edit data source' configuration page. At the top, the URL is 'localhost:3000/datasources/edit/1?gettingstarted'. The title is 'Edit data source'. There's a table with two rows: 'Name' (MyDataSource) and 'Type' (InfluxDB). A checked checkbox labeled 'Default' is next to the type row. Below the table are sections for 'HTTP settings' and 'HTTP Auth'. Under 'HTTP settings', the 'URL' is set to 'http://localhost:8086' and 'Access' is set to 'direct'. Under 'HTTP Auth', there are two options: 'Basic Auth' and 'With Credentials', with 'With Credentials' being selected. In the center, there's a section titled 'InfluxDB Details' with fields for 'Database' (grafana-clientes), 'User' (admin), 'Password' (.....), and 'Min time interval' (10s). A green success message at the bottom says 'Data source is working'. At the bottom right are 'Save & Test', 'Delete', and 'Cancel' buttons.

# Explicación – Grafana Dashboards

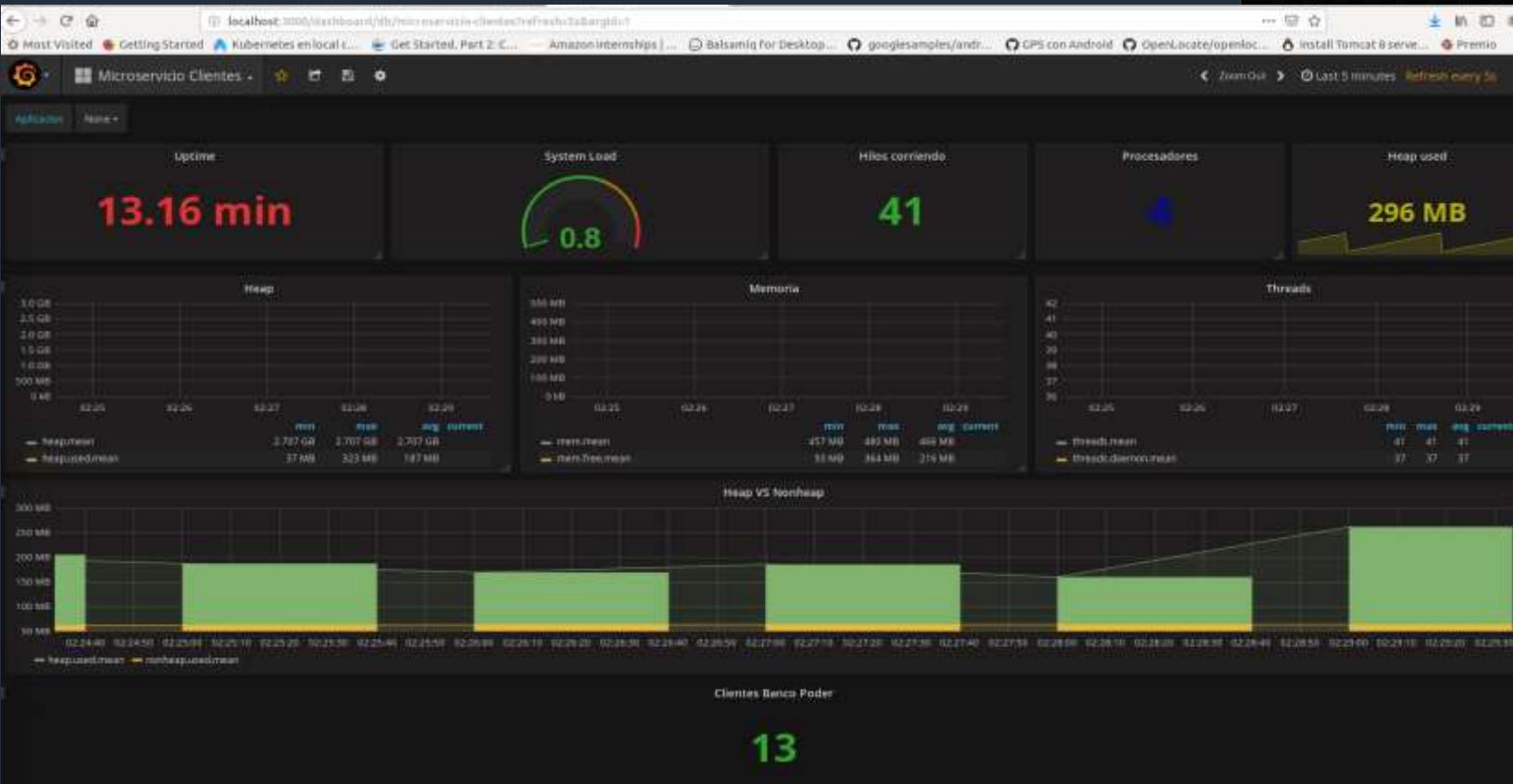
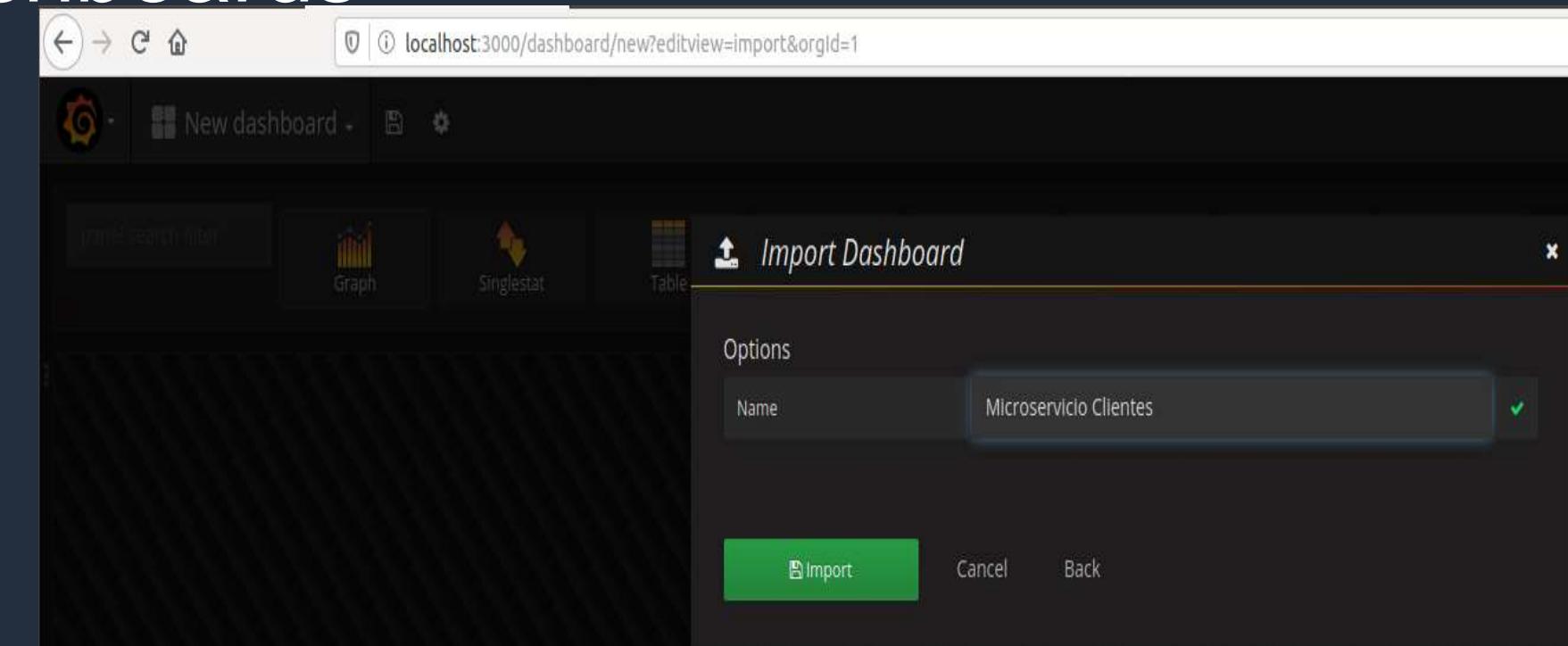
- 1.- Añadir Dashboards de microservicios
- 2.- New DashBoard
- 3.- Import Dashboard
- 4.- Upload.json file
- 5.- Load



# Explicación – Grafana Dashboards

1.- Indicar nombre del dashboard

2.- IMPORT



# Explicación – Grafana Dashboards

- 1.- Editar DashBoards, variable **counter.banco.poder.clientes.nuevos**
- 2.- Pruebas con Runner de Postman
- 3.- Pruebas con wrk

Postman:

/monitoreo/API Clientes post 201

/monitoreo/API Clientes Creditos post 201

The screenshot shows a Grafana dashboard with a single stat panel. The title of the panel is 'CLIENTES Banco Poder'. The value displayed is '1.004 K'. Below the value, there are tabs for 'General', 'Metrics', 'Options', 'Value Mappings', and 'Time range'. Under the 'Metrics' tab, there is a query editor with the following SQL-like query:

```
FROM default WHERE counter.banco.poder.clientes.nuevos  
SELECT field_value COUNT() AS count  
GROUP BY time(5m)  
FORMAT AS Time series  
ALIAS BY count  
Add Query
```

The screenshot shows the Postman Collection Runner interface. On the left, there is a sidebar with a list of collections, including 'curso\_microservicios' which is currently selected. In the main area, there is a search bar labeled 'Choose a collection or folder' with the text 'monitoreo' typed in. Below the search bar, there is a list of API requests: 'API Clientes post 201' and 'API Clientes Creditos post 201', both of which have a checkmark next to them. To the right, there are settings for 'Environment' (set to 'No Environment'), 'Iterations' (set to 1000), 'Delay' (set to 0 ms), and several checkboxes for 'Save responses', 'Keep variable values', 'Run collection without using stored cookies', and 'Save cookies after collection run'. At the bottom, there is a large blue button labeled 'Run curso\_micro...'. The top right corner of the interface shows the date 'lun 00:46' and the word 'Postman'.

# Explicación – Eureka Server

1.- Eureka Registra los microservicios clientes y creditos  
Permite la comunicación entre ellos con openFeing

localhost:8081/metrics  
localhost:8081/health

The image shows two browser tabs side-by-side. The left tab is titled 'localhost:8761' and displays the Spring Eureka dashboard. It includes sections for 'System Status' (Environment: test, Data center: default), 'DS Replicas' (Instances currently registered with Eureka: MS-CLIENTES, MS-CREDITOS), and a table of registered instances. The right tab is titled 'localhost:8081/metrics' and displays a JSON response from the Metrics endpoint. The response shows various metrics with their values, such as gauge, response, and counter metrics for clients and servo discovery client operations.

| Application | AMIs   | Availability Zones | Status                                 |
|-------------|--------|--------------------|--|
| MS-CLIENTES | n/a(1) | (1)                | UP (1) - b326fa5cdbd4:ms-clientes:8081 |
| MS-CREDITOS | n/a(1) | (1)                | UP (1) - d425d3290d6d:ms-creditos:8082 |

| Metric  | Value               |
|---|---------------------|
| gauge.response.api.v1.clientes.id.creditos:                                   | 902                 |
| counter.banco.poder.clientes.nuevos:  | 4                   |
| gauge.response.api.v1.clientes:   | 9                   |
| counter.status.500.api.v1.clientes.id.creditos:                               | 1                   |
| counter.status.201.api.v1.clientes:   | 4                   |
| gauge.servo.eurekaclient.registration.lastheartbeatsec_00480:                 | 0                   |
| counter.servo.eurekaclient.transport.request:                                 | 0                   |
| counter.servo.discoveryclient-httpclient_reuse:                               | 236                 |
| counter.servo.discoveryclient-httpclient_createnew:                           | 8                   |
| counter.servo.discoveryclient-httpclient_request:                             | 244                 |
| counter.servo.discoveryclient-httpclient_release:                             | 244                 |
| counter.servo.discoveryclient-httpclient_delete:                              | 7                   |
| normalized.servo.discoveryclient-httpclient_requestconnectiontimer.totaltime: | 0.02387108333333333 |
| normalized.servo.discoveryclient-httpclient_requestconnectiontimer.count:     | 1.0333333333333334  |
| gauge.servo.discoveryclient-httpclient_requestconnectiontimer.min:            | 0.014258            |
| gauge.servo.discoveryclient-httpclient_requestconnectiontimer.max:            | 0.13394999999999999 |

# Explicación – Eureka Server

## 1.- Invocar el API de créditos

En el endpoint indicar el id del cliente previamente creado

The screenshot shows the Postman interface. On the left, the 'Collections' tab is selected, displaying various API collections like 'api\_gateway\_apigee', 'curso\_microservicios', and 'kubernetes'. On the right, a specific POST request is being configured:

- Method:** POST
- URL:** `http://localhost:8081/api/v1/clientes/-233904255/creditos`
- Body (JSON):**

```
1 {  
2   "MontoCredito": "30000",  
3   "folioCliente": "-2120306162"  
4 }
```
- Headers (4):** (This section is visible but empty in the screenshot)
- Body (Pretty):**

```
1 {  
2   "folioCredito": "2052202058",  
3   "montoDeuda": 30000.0,  
4   "folioCliente": "-233904255"
```

```
wrk -t1 -c1 -d100s http://localhost:8081/api/v1/clientes
```



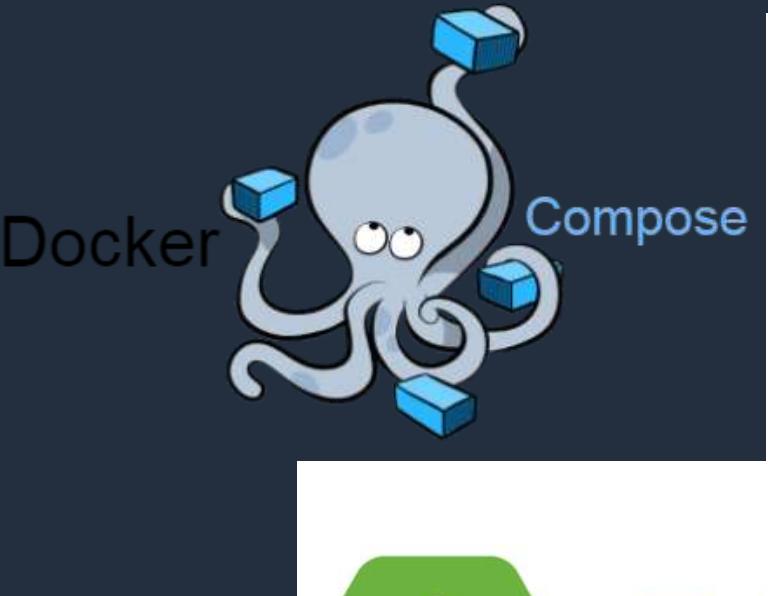
# Hands-ON Event Sourcing

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_202403\\_microservices/lab7\\_microservicios\\_event\\_sourcing\\_monitoreo](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_202403_microservices/lab7_microservicios_event_sourcing_monitoreo)

## ABRIR STS E IMPORTAR PROYECTO

# Microservicios Async + Monitoreo

Tecnología  
Spring Boot + API + Ingresos + Docker + Migración + Docker compose + PostgreSQL  
Descripción de microservicio:  
1. API SEGURO  
2. API ADMINISTRACIÓN DE AUTORIZACIONES  
3. MÚLTIPLES SERVICIOS EN LA BD  
4. ESTRUCTURA MONOLITICA  
5. ESTRUCTURA SEGURO, GUARDIA Y USO DE API  
6. SERVICIO BOOT ADMIN  
7. DESARROLLO GROWTH



# Ejecución - Microservicios Async + Monitoreo

## Ejecución: ./run-local.sh

```
#!/usr/bin/env bash
```

```
cd api-adm-autorizaciones  
./generalImagen.sh
```

```
cd ..
```

```
cd discovery-server  
./generalImagen.sh
```

```
cd ..
```

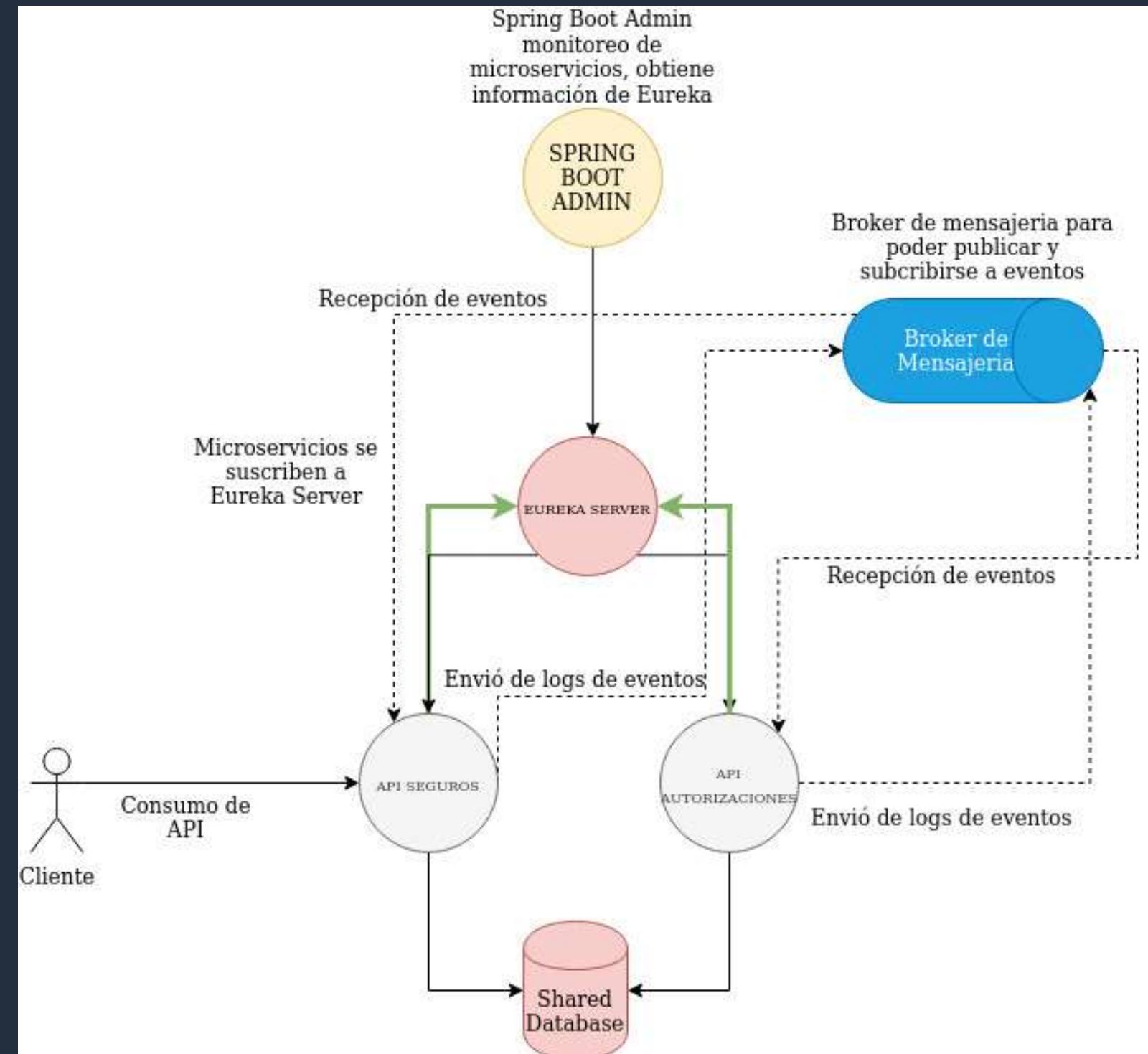
```
cd monitor-server  
./generalImagen.sh
```

```
cd ..
```

```
cd api-seguros  
./generalImagen.sh
```

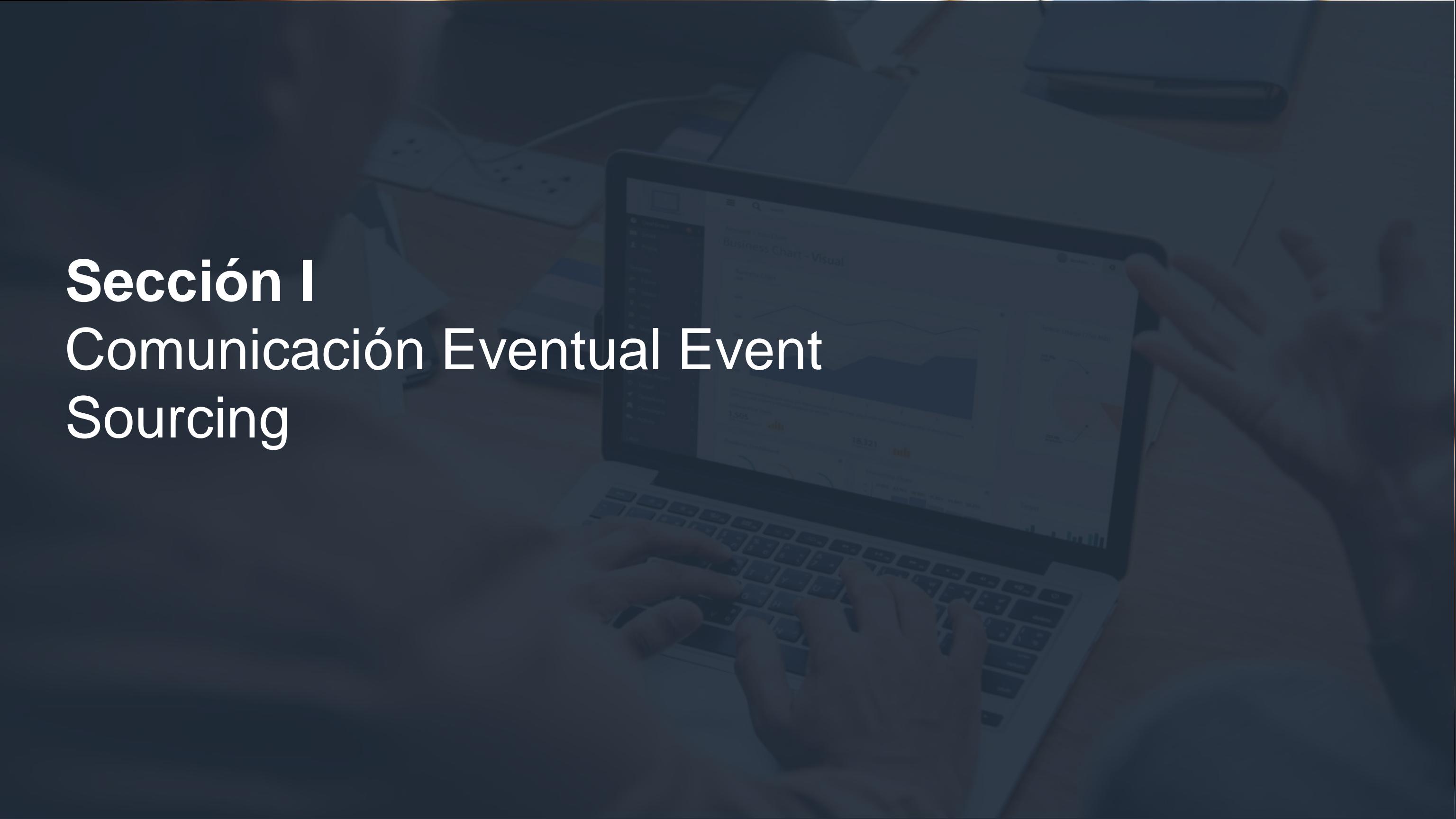
```
cd ..  
docker-compose up --build
```

```
docker-compose stop  
docker-compose kill  
docker-compose rm -f
```



# Sección I

## Comunicación Eventual Event Sourcing



# Hands-ON Patron Event Sourcing – Microservicios

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab7\\_microservicios\\_event\\_sourcing\\_monitoreo](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab7_microservicios_event_sourcing_monitoreo)

ABRIR STS E IMPORTAR PROYECTOS

# Event Source Pattern

**Event Sourcing Pattern asegura que los cambios aplicados a un objeto son almacenados como una secuencia de eventos, los cuales pueden ser consultados y reconstruidos, como parte de un log, en caso de alguna falla o en caso de requerir comprobar el estado final de un objeto a partir de replicar, los eventos aplicados al mismo.**

Intención.

- En lugar de almacenar sólo el estado actual de un objeto de dominio, implemente un repositorio con tipo de almacenamiento “append- only”, para registrar la serie completa de eventos ejecutados sobre los objetos.

El repositorio actúa como el sistema de registro o “log” el cuál, puede ser utilizado para materializar los objetos del dominio en el futuro.

**El repositorio o “log” de eventos, puede simplificar tareas en dominios de negocio complejos, evitando la necesidad de sincronizar el modelo de datos y/o el estado actual de un objeto; al tiempo que mejora el rendimiento, la escalabilidad y la capacidad de respuesta del sistema.**

**También brinda eventual consistencia para transacciones distribuidas, y permite mantener registros de auditoría completos y su historial que pueda habilitar acciones compensatorias.**



# Event Source Pattern

## Aplicabilidad.

- Cuando es necesario capturar los cambios de estado de un objeto de dominio como una serie de eventos.

**Cuando sea vital minimizar o eliminar conflictos de concurrencia sobre la actualización de los datos persistidos**

**Cuando sea requerido almacenar los eventos ocurridos sobre el estado de un objeto o sistema y sea necesario reproducirlos, en el futuro, para restaurar el estado del sistema o, para aplicar cambios de jpo “roll-back” o, simplemente para mantener un historial o pistas de auditoria confiables.**

Cuando el manejo de eventos es una funcionalidad natural de operación en el sistema; la implementación de Event Sourcing Pattern no requiere mayor esfuerzo.

**Cuando sea requerido desacoplar los procesos de entrada de datos, con los de actualización de datos y desea mantener un estado consistente de los mismos.**

**Cuando requiera implementar mecanismos de eventual consistencia es fundamental la aplicabilidad de Event Sourcing Pattern.**

## Ventajas:

**Implementar eventual consistencia entre servicios distribuidos**

Mantener el historial de cambios de estados sobre objetos de dominio los cuales pueden ser replicables en el futuro.

**Fácil integración en sistemas con arquitecturas orientada a eventos.**

**Facilita la compensación de transacciones.**

**Incrementa la resistencia del sistema a fallas de software y/o hardware habilitando la recuperación del sistema mediante ejecutar la serie de eventos previos.**

## Desventajas:

Latencia de la red puede ocasionar que un evento llegue al repositorio de eventos de forma tal que sea necesaria una reconciliación de dato

Dificultad para implementar consumidores de eventos de tipo “at-least-once” debido a manejar un mal diseño no Idempotente de Servicios.



# Rabbit MQ

<http://localhost:8090/>

<http://localhost:15672>

usr: guest

Pwd: guest

The screenshot shows the RabbitMQ Management Interface's Overview page. At the top, there are navigation links for Overview, Connections, Channels, Exchanges, Queues, and Admin. The Overview tab is selected. Below the tabs, the page title is "Overview". A section titled "Totals" displays metrics: Queued messages (chart: last minute), Currently idle, Message rates (chart: last minute), Currently idle, and Global counts. Below these are buttons for Connections: 0, Channels: 0, Exchanges: 8, Queues: 0, and Consumers: 0. A "Node" section shows a single node named "rabit@6c0997bdda29" with details like File descriptors, Socket descriptors, Erlang processes, Memory, Disk space, Rates mode, Info, and a Reset stats DB button. A "Paths" section lists Config file, Database directory, Log file, and SASL log file paths. At the bottom, there are sections for Ports and contexts, Export definitions, and Import definitions, along with links for HTTP API and Command Line.

- 1.- Queue para generar un evento de autorización
- 2.- Queue para obtener la confirmación de la autorización desde otro contexto de microservicio

## Microservicio api-seguros



# Explicación - Comunicación Eventual

- 1.- Consulta de catalogos funcional HTTP
  - 2.- Crear seguro y solicitar autorización
  - 3.- Solicitud de autorizacion de manera asincrona generando mayor tolerancia a fallos y generación de logs de dominios, asi como la eventual consistencia

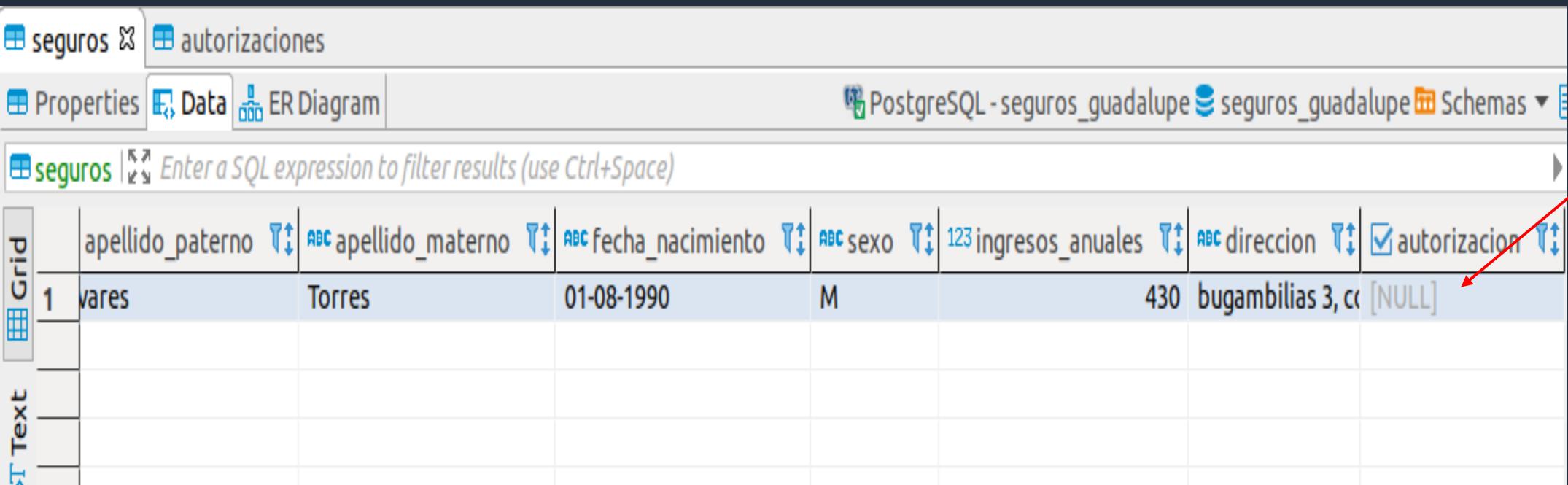
```
nnning the evict task with compensationTime 0ms  
discovery-server_1 | 2020-02-27 05:11:02.441 INFO 1 --- [thresholdUpdater] c.n.e.r.PeerAwareInstanceRegistryImpl : Cu  
rrent renewal threshold is : 5  
discovery-server_1 | 2020-02-27 05:11:05.776 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Ru  
nning the evict task with compensationTime 0ms  
api-seguros-guadalupe-administracion_1 | 2020-02-27 05:11:19.362 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Re  
solving eureka endpoints via configuration  
api-seguros-guadalupe_1 | 2020-02-27 05:11:21.917 INFO 1 --- [trap-executor-0] c.n.d.s.r.aws.ConfigClusterResolver : Re  
solving eureka endpoints via configuration  
discovery-server_1 | 2020-02-27 05:12:05.776 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Ru  
nning the evict task with compensationTime 0ms  
api-seguros-guadalupe_1 | 2020-02-27 05:12:35.299 INFO 1 --- [nio-8081-exec-4] c.s.api.SegurosController : >>  
> seguros/v1/catalogos leerCatalogos  
api-seguros-guadalupe_1 | Hibernate: select catalogos0_.id as id1_0_, catalogos0_.descripcion as descripc2_0_, catalogos0_.es  
tatus as estatus3_0_, catalogos0_.nombre as nombre4_0_ from catalogo_seguros catalogos0_ where catalogos0_.estatus=?  
api-seguros-guadalupe_1 | 2020-02-27 05:12:48.432 INFO 1 --- [nio-8081-exec-6] c.s.api.SegurosController : >>  
> seguros/v1 creando seguro  
api-seguros-guadalupe_1 | Hibernate: insert into seguros (apellido_materno, apellido_paterno, autorizacion, direccion, fecha  
nacimiento, ingresos_anuales, nombre, plazo, precio_poliza, sexo, suma_asegurada, tipo_cobertura, vencimiento, id) values (?, ?, ?, ?, ?, ?, ?,  
?, ?, ?, ?, ?, ?, ?, ?)  
api-seguros-guadalupe_1 | 2020-02-27 05:12:48.617 INFO 1 --- [nio-8081-exec-6] c.s.s.r.a.AdministracionSegurosProducer : >>  
>Envio de autorizacion correctamente...  
api-seguros-guadalupe_1 | 2020-02-27 05:12:48.617 INFO 1 --- [nio-8081-exec-6] c.s.service.SegurosServiceImpl : >>  
>Envia solicitud de autorizacion para el area de verificacion de seguros  
api-seguros-guadalupe-administracion_1 | 2020-02-27 05:12:48.700 INFO 1 --- [cTaskExecutor-1] c.s.a.s.remote.async.SegurosConsumer : Ev  
ent autorizaciones de seguros{"id":"c135cae2-f495-4ffa-9e0b-6137ca23956e","plazo":3.5,"precioPoliza":10.5,"tipoCobertura":2,"vencimiento":"20  
25-01-01","sumaAsegurada":800.0,"nombre":"Julior","apellidoPaterno":"Alvares","apellidoMaterno":"Torres","fechaNacimiento":"01-08-1990","sexo  
":"M","ingresosAnuales":430.0,"direccion":"bugambilias 3, colonia la nueva generacion"}  
api-seguros-guadalupe-administracion_1 | Hibernate: insert into autorizaciones (datos_poliza, estatus, fecha_apertura, fecha_confirmacion, i  
d_autorizacion) values (?, ?, ?, ?, ?)  
discovery-server_1 | 2020-02-27 05:13:05.776 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Ru  
nning the evict task with compensationTime 0ms
```

Postman:  
/comunicacion\_eventual/01\_API\_Seguros Catalogos GET 200  
/comunicacion\_eventual/01\_API\_Seguros POST 201



# Explicación - Comunicación Eventual

- 1.- Consulta de catalogos funcional HTTP
- 2.- Crear seguro y solicitar autorización
- 3.- Solicitud de autorizacion de manera asincrona generando mayor tolerancia a fallos y generación de logs de dominios, asi como la eventual consistencia



The screenshot shows a PostgreSQL database interface with the following details:

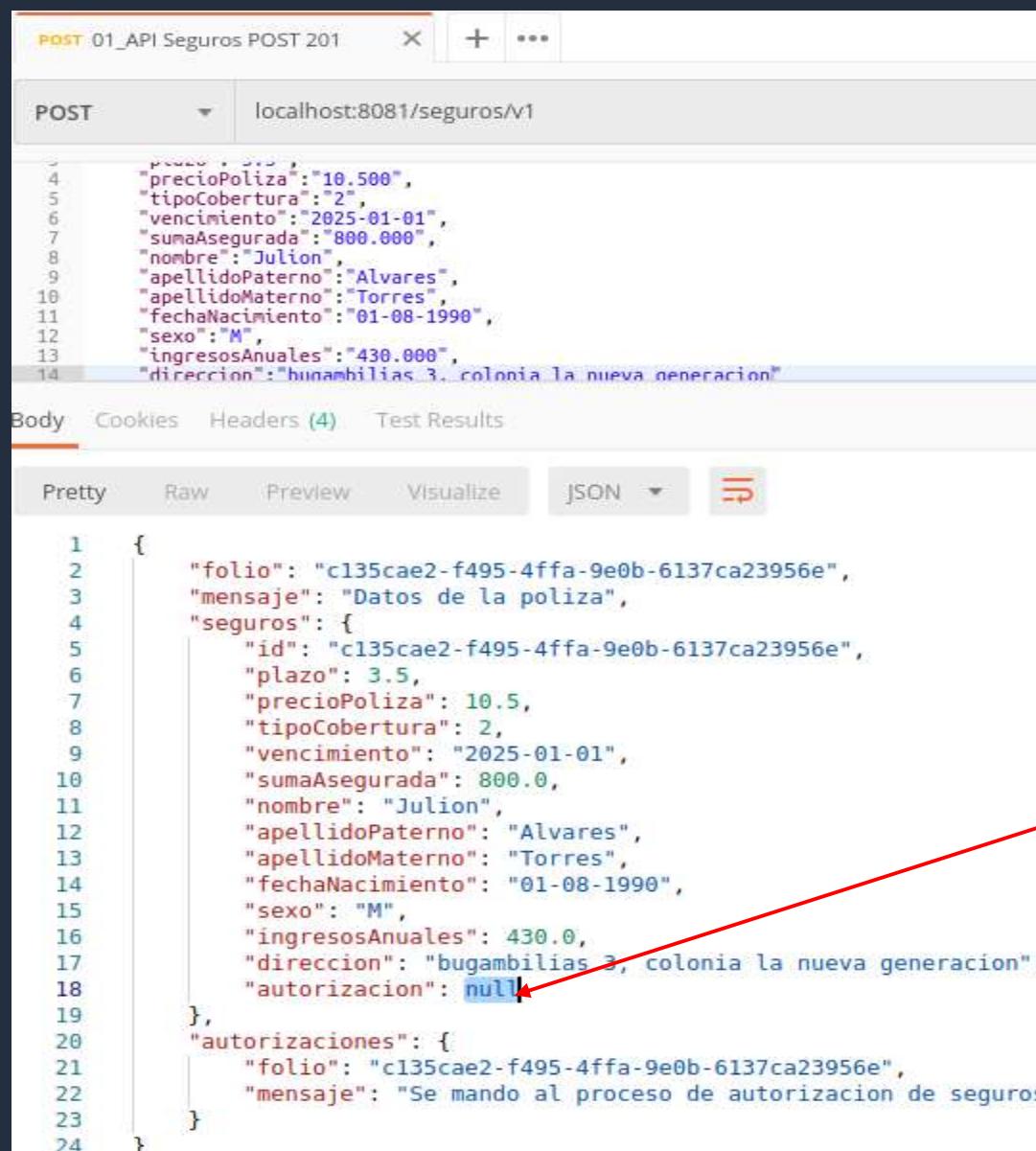
- Database:** PostgreSQL - seguros\_guadalupe
- Schemas:** seguros\_guadalupe
- Table:** seguros
- Columns:** apellido\_paterno, apellido\_materno, fecha\_nacimiento, sexo, ingresos\_anuales, direccion, autorizacion
- Data:** The first row shows values: Varela, Torres, 01-08-1990, M, 430, bugambilias 3, cc [NULL]

A red arrow points to the "autorizacion" column, highlighting the NULL value.



# Explicación - Comunicación Eventual

- 1.- Se dio alta el seguro en estatus sin autorización
- 2.- El microservicio de seguros, generó un evento para generar la autorización
- 3.- Un listener bajo el mensaje de autorización y lo dio de alta pero no se ha procesado su estatus



POST 01\_API Seguros POST 201 X + ...

POST localhost:8081/seguros/v1

```
4 "plazo": 3.5,
5 "precioPoliza": "10.500",
6 "tipoCobertura": 2,
7 "vencimiento": "2025-01-01",
8 "sumaAsegurada": "800.000",
9 "nombre": "Julion",
10 "apellidoPaterno": "Alvares",
11 "apellidoMaterno": "Torres",
12 "fechaNacimiento": "01-08-1990",
13 "sexo": "M",
14 "ingresosAnuales": "430.000",
15 "direccion": "bugambilias 3, colonia la nueva generacion"
```

Body Cookies Headers (4) Test Results

Pretty Raw Preview Visualize JSON

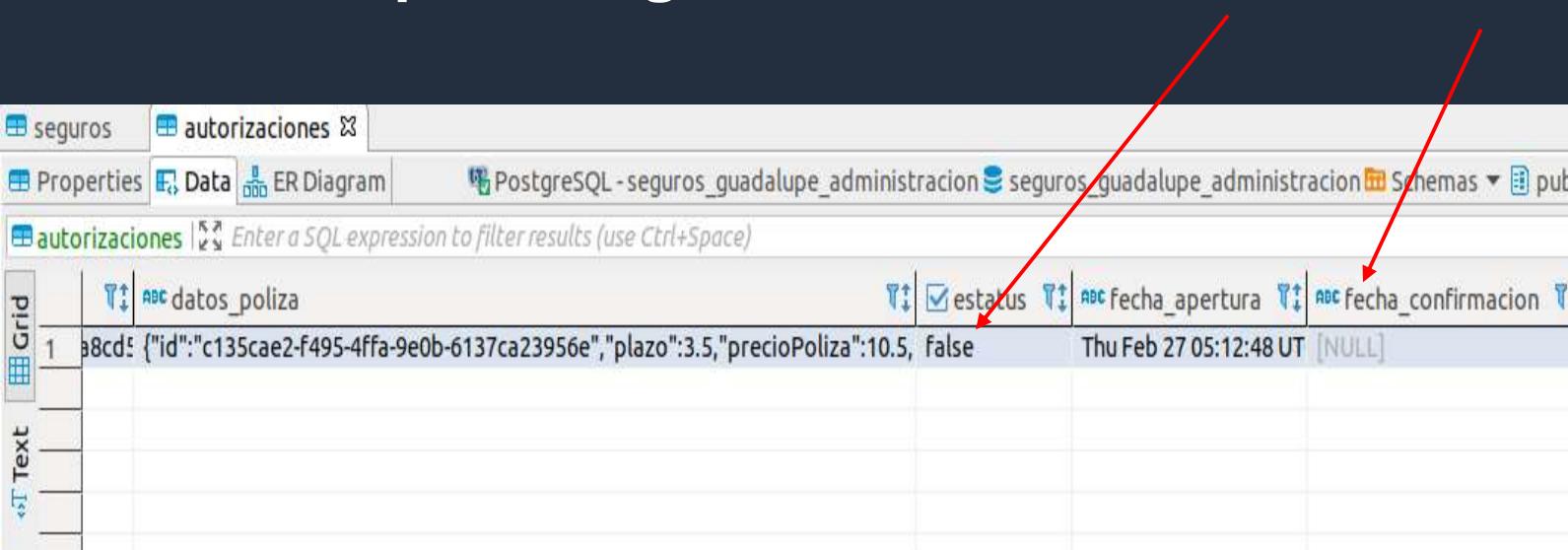
```
1 {
2     "folio": "c135cae2-f495-4ffa-9e0b-6137ca23956e",
3     "mensaje": "Datos de la poliza",
4     "seguros": {
5         "id": "c135cae2-f495-4ffa-9e0b-6137ca23956e",
6         "plazo": 3.5,
7         "precioPoliza": 10.5,
8         "tipoCobertura": 2,
9         "vencimiento": "2025-01-01",
10        "sumaAsegurada": 800.0,
11        "nombre": "Julion",
12        "apellidoPaterno": "Alvares",
13        "apellidoMaterno": "Torres",
14        "fechaNacimiento": "01-08-1990",
15        "sexo": "M",
16        "ingresosAnuales": 430.0,
17        "direccion": "bugambilias 3, colonia la nueva generacion",
18        "autorizacion": null
19    },
20    "autorizaciones": {
21        "folio": "c135cae2-f495-4ffa-9e0b-6137ca23956e",
22        "mensaje": "Se mando al proceso de autorizacion de seguros"
23    }
}
```

Postman:  
/comunicacion\_eventual/01\_API Seguros Catalogos GET 200  
/comunicacion\_eventual/01\_API Seguros POST 201

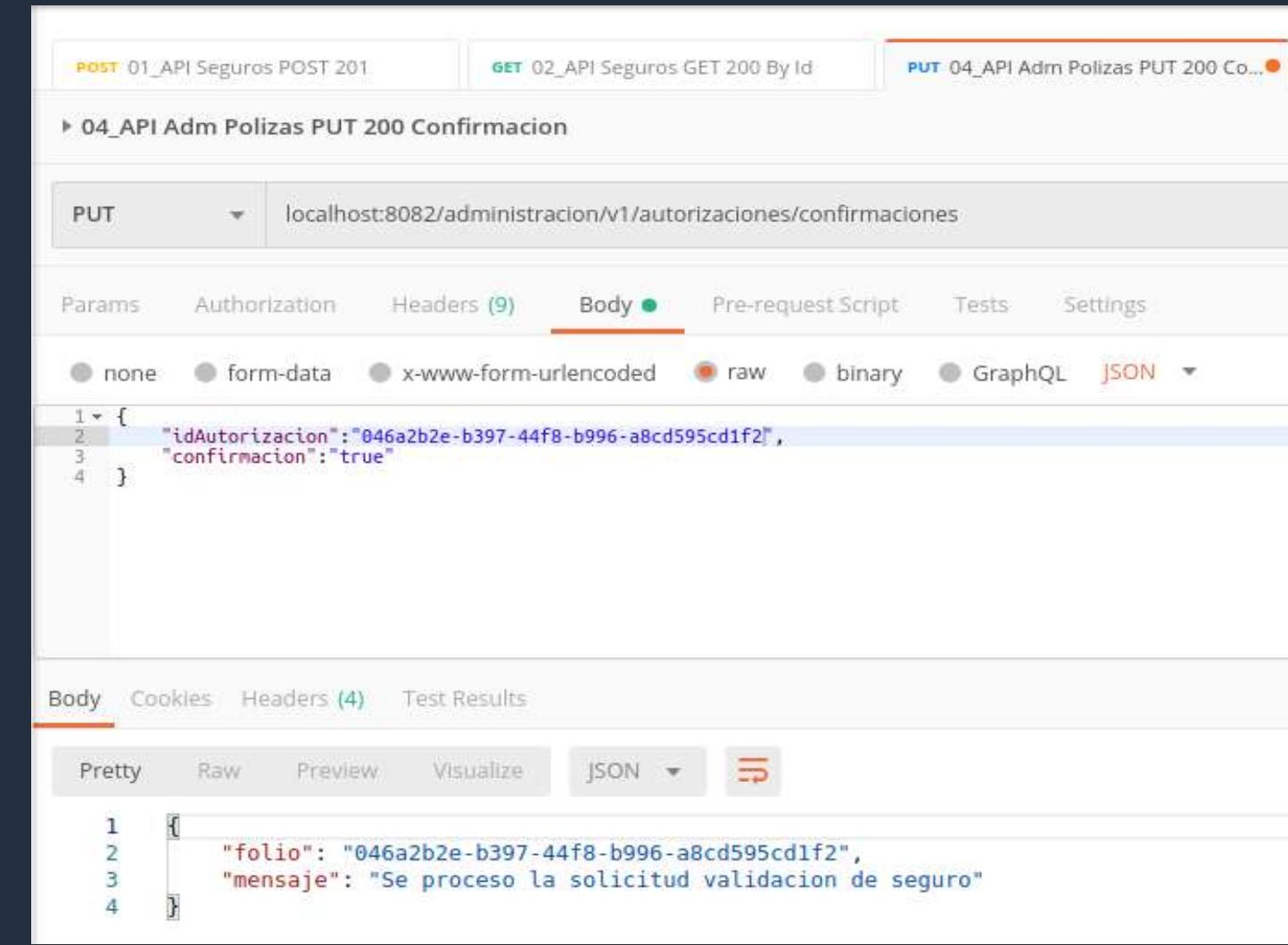


# Explicación - Comunicación Eventual

- 1.- Eventualmente se da de alta la autorización del seguro, para posteriormente realizar la confirmación de autorización en el api de administracion de autorizaciones, se cambian los estatus y fechas de autorizacion.
- 2.- El microservicios de administracion de autorizaciones genera un evento una vez realizada la autorización y el api de seguros via un listerner actualiza su estatus indicando que el seguro fue autorizado.



|   | datos_poliza  | estatus                             | fecha_apertura                | fecha_confirmacion |
|---|---|-------------------------------------|-------------------------------|--------------------|
| 1 | {"id":"c135cae2-f495-4ffa-9e0b-6137ca23956e","plazo":3.5,"precioPoliza":10.5, "false" | <input checked="" type="checkbox"/> | Thu Feb 27 05:12:48 UT [NULL] | [NULL]             |



```
PUT /localhost:8082/administracion/v1/autorizaciones/confirmaciones
```

Body (JSON)

```
{  
  "idAutorizacion": "046a2b2e-b397-44f8-b996-a8cd595cd1f2",  
  "confirmacion": "true"}  
Pretty Raw Preview Visualize JSON
```

Body  
Cookies  
Headers (4)  
Test Results

```
1  {  
2    "folio": "046a2b2e-b397-44f8-b996-a8cd595cd1f2",  
3    "mensaje": "Se proceso la solicitud validacion de seguro"  
4  }
```

Postman:  
/comunicacion\_eventual/04\_API Adm Polizas PUT 200  
Confirmación

# Explicación - Comunicación Eventual

```
api-seguros-guadalupe-administracion_1 | Hibernate: select autorizaci0_.id_autorizacion as id_autorl_0_0_, autorizaci0_.datos_poliza as datos_po2_0_0_, autorizaci0_.estatus as estatus3_0_0_, autorizaci0_.fecha_apertura as fecha_ap4_0_0_, autorizaci0_.fecha_confirmacion as fecha_co5_0_0_ from autorizaciones autorizaci0_ where autorizaci0_.id_autorizacion=?  
api-seguros-guadalupe-administracion_1 | Hibernate: update autorizaciones set datos_poliza=?, estatus=?, fecha_apertura=?, fecha_confirmacion=? where id_autorizacion=?  
api-seguros-guadalupe-administracion_1 | 2020-02-27 05:48:50.671 INFO 1 --- [nio-8082-exec-3] c.s.a.service.AdministracionServiceImpl : >>>Actuliza autorizacion...  
api-seguros-guadalupe-administracion_1 | 2020-02-27 05:48:50.708 INFO 1 --- [nio-8082-exec-3] c.s.a.s.remote.async.SegurosProducer : >>>Envio de confirmacion a seguros...  
api-seguros-guadalupe-administracion_1 | 2020-02-27 05:48:50.709 INFO 1 --- [nio-8082-exec-3] c.s.a.service.AdministracionServiceImpl : >>>Se notifico el estatus de la poliza remotamente...  
api-seguros-guadalupe_1 | 2020-02-27 05:48:50.773 INFO 1 --- [cTaskExecutor-1] dministracionSegurosConfirmacionConsumer : Event confirmacion del adm de autorizaciones{"autorizacion":true,"idSeguro":"c135cae2-f495-4ffa-9e0b-6137ca23956e"}  
api-seguros-guadalupe_1 | Hibernate: select seguros0_.id as id1_1_, seguros0_.apellido_materno as apellido2_1_, seguros0_.apellido_paterno as apellido3_1_, seguros0_.autorizacion as autoriza4_1_, seguros0_.direccion as direccio5_1_, seguros0_.fecha_nacimiento as fecha_na6_1_, seguros0_.ingresos_anuales as ingresos7_1_, seguros0_.nombre as nombre8_1_, seguros0_.plazo as plazo9_1_, seguros0_.precio_poliza as precio_10_1_, seguros0_.sexo as sexoll_1_, seguros0_.suma_asegurada as suma_as12_1_, seguros0_.tipo_cobertura as tipo_col3_1_, seguros0_.vencimiento as vencimi14_1_ from seguros seguros0_ where seguros0_.id=?  
api-seguros-guadalupe_1 | Hibernate: select seguros0_.id as id1_1_0_, seguros0_.apellido_materno as apellido2_1_0_, seguros0_.apellido_paterno as apellido3_1_0_, seguros0_.autorizacion as autoriza4_1_0_, seguros0_.direccion as direccio5_1_0_, seguros0_.fecha_nacimiento as fecha_na6_1_0_, seguros0_.ingresos_anuales as ingresos7_1_0_, seguros0_.nombre as nombre8_1_0_, seguros0_.plazo as plazo9_1_0_, seguros0_.precio_poliza as precio_10_1_0_, seguros0_.sexo as sexoll_1_0_, seguros0_.suma_asegurada as suma_as12_1_0_, seguros0_.tipo_cobertura as tipo_col3_1_0_, seguros0_.vencimiento as vencimi14_1_0_ from seguros seguros0_ where seguros0_.id=?  
api-seguros-guadalupe_1 | Hibernate: update seguros set apellido_materno=?, apellido_paterno=?, autorizacion=?, direccion=?, fecha_nacimiento=?, ingresos_anuales=?, nombre=?, plazo=?, precio_poliza=?, sexo=?, suma_asegurada=?, tipo_cobertura=?, vencimiento=? where id=?  
discovery-server_1 | 2020-02-27 05:49:05.789 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms  
discovery-server_1 | 2020-02-27 05:50:05.790 INFO 1 --- [a-EvictionTimer] c.n.e.registry.AbstractInstanceRegistry : Running the evict task with compensationTime 0ms
```

# Explicación - Comunicación Eventual

|   | datos_poliza  | estatus                | fecha_apertura               | fecha_confirmacion |
|---|---|------------------------|------------------------------|--------------------|
| 1 | {"id": "c135cae2-f495-4ffa-9e0b-6137ca23956e", "plazo": 3.5, "precioPoliza": 10.5, "estatus": true} | Thu Feb 27 05:12:48 UT | Thu Feb 27 05:48:50 UTC 2024 |                    |

|   | apellido_paterno | apellido_materno | fecha_nacimiento | sexo | ingresos_anuales | direccion           | autorizacion |
|---|------------------|------------------|------------------|------|------------------|---------------------|--------------|
| 1 | Varela           | Torres           | 01-08-1990       | M    | 430              | Bugambilias 3, cctv | true         |



# Explicación - Comunicación Eventual

**localhost:8082/administracion/v1/autorizaciones/confirmaciones**

PUT

```
{  
    "idAutorizacion": "d9e3b34b-d593-408a-a0ae-  
35bd783a0699",  
    "confirmacion": "true"  
}
```

RESPONSE

```
{  
    "folio": "d9e3b34b-d593-408a-a0ae-  
35bd783a0699",  
    "mensaje": "Se proceso la solicitud validacion de  
seguro"  
}
```



# Explicación - Comunicación Eventual

**GET**

**localhost:8081/seguros/v1/6a859c3c-03f1-4b37-953e-9285997e350f**

```
{  
    "id": "6a859c3c-03f1-4b37-953e-9285997e350f",  
    "plazo": 1.5,  
    "precioPoliza": 6.9,  
    "tipoCobertura": 1,  
    "vencimiento": "2021-01-01",  
    "sumaAsegurada": 250,  
    "nombre": "Erika",  
    "apellidoPaterno": "Ramirez",  
    "apellidoMaterno": "Acosta",  
    "fechaNacimiento": "01-08-1990",  
    "sexo": "M",  
    "ingresosAnuales": 100,  
    "direccion": "10 de abril, #20, cuernavaca  
morelos",  
    "autorizacion": true  
}
```



# Simular que se caen los microservicios

docker ps

docker stop <id\_contenedor>

```
jovani@developer:~$ docker ps
CONTAINER ID        IMAGE               COMMAND
CREATED             STATUS              PORTS
NAMES
f0846c3aa513        jovaniac/api-seguros-guadalupe-administracion:0.0.1-snapshot   "java -Djava.securit...
12 minutes ago      Up 12 minutes       0.0.0.0:8082->8082/tcp
                    lab4_microservicios_event_monitoreo_api-seguros-guadalupe-administracion_1_b6a3a
b074863
5bb28ea6a769        jovaniac/api-seguros-guadalupe:0.0.1-snapshot   "java -Djava.securit...
12 minutes ago      Up 12 minutes       0.0.0.0:8081->8081/tcp
                    lab4_microservicios_event_monitoreo_api-seguros-guadalupe_1_9bb8865b38be
7b5f9703750f        jovaniac/monitor-server:0.0.1-snapshot   "java -Djava.awt.he...
13 minutes ago      Up 12 minutes       0.0.0.0:8090->8090/tcp
                    lab4_microservicios_event_monitoreo_monitor_1_536b59a012c9
672713d23bfc        lab4_microservicios_event_monitoreo_postgres   "/docker-entrypoint...
13 minutes ago      Up 12 minutes       0.0.0.0:5432->5432/tcp
                    lab4_microservicios_event_monitoreo_postgres_1_d5f0426f1203
9ab756e9d3c9         rabbitmq:3.6.12-management-alpine   "docker-entrypoint.s...
13 minutes ago      Up 12 minutes       4369/tcp, 5671/tcp, 0.0.0.0:5672->5672/tcp, 15671/tcp, 25672/tcp,
0.0.0.0:15672->15672/tcp   lab4_microservicios_event_monitoreo_rabbitmq_1_b3ba0fc085d7
56f63815c707        jovaniac/eureka-discovery:0.0.1-snapshot   "java -Djava.awt.he...
13 minutes ago      Up 12 minutes       0.0.0.0:8761->8761/tcp
                    lab4_microservicios_event_monitoreo_discovery-server_1_b87d7fccf930
jovani@developer:~$ docker stop f0846c3aa513
```

# Simular que se caen los microservicios

## Creación de muchas polizas

The image shows two PostgreSQL database interfaces side-by-side, illustrating the creation of many policies.

**Left Database (seguros):**

- Table:** seguros
- Columns:** id, plazo, precio\_poliza, tipo\_cobertura, vencimiento, suma\_asegurada, estatus
- Data:** 13 rows of policy records.

| id                                    | plazo | precio_poliza | tipo_cobertura | vencimiento | suma_asegurada | estatus |
|---------------------------------------|-------|---------------|----------------|-------------|----------------|---------|
| fc53ce0d-f5e4-4f8c-99a6-75a4324d24bb  | 3.5   | 10.5          |                | 2025-01-01  | 800            | True    |
| 3fff3eae-84bc-4a74-b959-84f7eda23758  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| 7a142c59-acc2-4160-9825-51863ef7cbe7  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| 087d5940-79c8-4a7b-8edf-ce8e20817970  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| 239c3d5f-752f-4dbd-a161-6adeb51a8e5a  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| e26a307a-1a7c-4ade-a3c0-12957bda0b27  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| e869311d-e4fd-4fbcc-8861-0b2348377ad3 | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| 2f6051b1-f5e6-4ec8-bbbd-39bddbaaa7f6  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| b96bc7dc-076b-45f6-b1f8-bb32c13fb4c4  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| 9d1da340-8aaf-499f-a80e-f2cdf8b87016  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| c777a105-893b-47d7-b5c9-0749d9ff80a9  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| 1aad421b-bb6f-47af-8e7a-2f28d50a2ce9  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |
| 723d6f90-7fd8-46ba-b6c5-6859cde1f861  | 5.5   | 50.5          |                | 2030-01-01  | 800            | True    |

**Right Database (autorizaciones):**

- Table:** autorizaciones
- Columns:** id\_autorizacion, datos\_poliza, estatus
- Data:** 2 rows of authorization records.

| id_autorizacion                      | datos_poliza   | estatus |
|--------------------------------------|--|---------|
| 6b723c50-a26a-4d04-b0b0-070960bde4fd | {"id": "fc53ce0d-f5e4-4f8c-99a6-75a4324d24bb", "plazo": 3.5, "precioPoliza": 10.5, "vencimiento": "2025-01-01", "summaAsegurada": 800} | True    |
| 9d48cfb6-ac80-4e76-9239-979f53e9d00f | {"id": "3fff3eae-84bc-4a74-b959-84f7eda23758", "plazo": 5.5, "precioPoliza": 50.5, "vencimiento": "2030-01-01", "summaAsegurada": 800} | False   |

# Simular que se caen los microservicios

# Levantar nuevamente el microservicio gradle bootRun

```
jovani@developer: ~/Documents/proyectos_jovani/cursos_2019/junio_2019/certificatic/curso_microservicios/lab4_microservicios_event_monitoreo/api-adm-a...   
File Edit View Search Terminal Help  
rkServlet 'dispatcherServlet': initialization started  
2019-06-03 00:59:07.251 INFO 4473 --- [nio-8082-exec-1] o.s.web.servlet.DispatcherServlet : FrameworkServlet 'dispatcherServlet': initialization completed in 24 ms  
2019-06-03 00:59:07.605 INFO 4473 --- [cTaskExecutor-1] c.s.a.s.remote.async.SegurosConsumer : Event autorizaciones de seguros{"id":"94b2b8d3-a23f-4dc8-ae82-7a1c7ba076a3","plazo":1.5,"precioPoliza":6.9,"tipoCo  
bertura":1,"vencimiento":"2021-01-01","sumaAsegurada":250.0,"nombre":"Erika","apellidoPaterno":"Ramirez","a  
pellidoMaterno":"Acosta","fechaNacimiento":"01-08-1990","sexo":"M","ingresosAnuales":100.0,"direccion":"10  
de abril, #20, cuernavaca morelos"}  
Hibernate: insert into autorizaciones (datos_poliza, estatus, fecha_apertura, fecha_confirmacion, id_autori  
zacion) values (?, ?, ?, ?, ?)  
2019-06-03 00:59:08.362 INFO 4473 --- [cTaskExecutor-1] c.s.a.s.remote.async.SegurosConsumer : Event autorizaciones de seguros{"id":"bfd9eb28-ecc6-4c1c-8822-6e820ac9507c","plazo":1.5,"precioPoliza":6.9,"tipoCo  
bertura":1,"vencimiento":"2021-01-01","sumaAsegurada":250.0,"nombre":"Erika","apellidoPaterno":"Ramirez","a  
pellidoMaterno":"Acosta","fechaNacimiento":"01-08-1990","sexo":"M","ingresosAnuales":100.0,"direccion":"10  
de abril, #20, cuernavaca morelos"}  
Hibernate: insert into autorizaciones (datos_poliza, estatus, fecha_apertura, fecha_confirmacion, id_autori  
zacion) values (?, ?, ?, ?, ?)  
2019-06-03 00:59:08.413 INFO 4473 --- [cTaskExecutor-1] c.s.a.s.remote.async.SegurosConsumer : Event autorizaciones de seguros{"id":"a250321a-ac80-41d6-bc4f-78a5e4a5c922","plazo":1.5,"precioPoliza":6.9,"tipoCo  
bertura":1,"vencimiento":"2021-01-01","sumaAsegurada":250.0,"nombre":"Erika","apellidoPaterno":"Ramirez","a  
pellidoMaterno":"Acosta","fechaNacimiento":"01-08-1990","sexo":"M","ingresosAnuales":100.0,"direccion":"10  
de abril, #20, cuernavaca morelos"}  
Hibernate: insert into autorizaciones (datos_poliza, estatus, fecha_apertura, fecha_confirmacion, id_autori  
zacion) values (?, ?, ?, ?, ?)  
2019-06-03 00:59:08.598 INFO 4473 --- [cTaskExecutor-1] c.s.a.s.remote.async.SegurosConsumer : Event autorizaciones de seguros{"id":"35fbf4eb-488c-459a-adfe-1a0e3177144d","plazo":1.5,"precioPoliza":6.9,"tipoCo  
bertura":1,"vencimiento":"2021-01-01","sumaAsegurada":250.0,"nombre":"Erika","apellidoPaterno":"Ramirez","a  
pellidoMaterno":"Acosta","fechaNacimiento":"01-08-1990","sexo":"M","ingresosAnuales":100.0,"direccion":"10
```

# Simular que se caen los microservicios

## Encolamiento en el cluster de mensajes

The screenshot shows the RabbitMQ Management Console interface at the URL `localhost:15672/#/queues`. The top navigation bar includes links for Overview, Connections, Channels, Exchanges, Queues (which is highlighted in orange), and Admin. The main content area is titled "Queues" and displays two queues: "queue-administracion-autorizacion" and "queue-administracion-confirmacion".

**Queues**

All queues (2)

Pagination

Page 1 of 1 - Filter:   Regex (?)(?)

| Overview                          |          |       | Messages |         |       | Message rates |               |        |
|-----------------------------------|----------|-------|----------|---------|-------|---------------|---------------|--------|
| Name                              | Features | State | Ready    | Unacked | Total | incoming      | deliver / get | ack    |
| queue-administracion-autorizacion | D        | idle  | 0        | 0       | 0     | 0.00/s        | 0.00/s        | 0.00/s |
| queue-administracion-confirmacion | D        | idle  | 5        | 0       | 5     | 0.00/s        | 0.00/s        | 0.00/s |

Add a new queue

HTTP API | Command Line

## **Sección I**

Patrones de diseño para microservicios:

API Gateway

Service Discovery

Shared Database

Microservice per container

Monitoreo Centralizado

Event Sourcing



# Hands-ON Banco Poder -

**API Gateway**  
**Shared database**  
**Microservice per container**  
**Event Sourcing**  
**Service Discovery**  
**Monitoring**

[https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1\\_microservicios\\_2024/03\\_microservices/lab8\\_microservicios\\_banco\\_poder\\_gateway](https://git-codecommit.us-east-1.amazonaws.com/v1/repos/r1_microservicios_2024/03_microservices/lab8_microservicios_banco_poder_gateway)

ABRIR STS E IMPORTAR PROYECTOS

# Configuración – Microservicios Banco Poder

Eureka Server

<http://localhost:8761>

Spring Boot Admin

<http://localhost:8090>

Zuul Gateway

<http://localhost:8766>

RabbitMQ

<http://localhost:15672>

usr: guest

Pwd: guest

The image displays two side-by-side screenshots of the Spring Boot Admin interface.

**Left Screenshot (Eureka Dashboard):**

- System Status:**
  - Environment: test
  - Data center: default
  - Current time: Uptime
  - Lease expiration enabled
  - Renew threshold
  - Renews (last min)
- DS Replicas:**
  - Instances currently registered with Eureka:

| Application           | AMIs    | Availability Zones | Status   |
|-----------------------|---------|--------------------|--|
| API-ADMINISTRACION    | n/a (1) | [1]                | UP (1) - e5738aa422ad/api-administracion:8082    |
| API-CLIENTES          | n/a (1) | [1]                | UP (1) - bfaf37f5eb7f/api-clientes:8084          |
| API-CREDITOS          | n/a (1) | [1]                | UP (1) - c0541f6deb83/api-creditos:8085          |
| API-EMPLEADOS         | n/a (1) | [1]                | UP (1) - b10e910f8655/api-empleados:8083         |
| API-SEGUROS-GUADALUPE | n/a (1) | [1]                | UP (1) - 16c84e94c7ad/api-seguros-guadalupe:8081 |
  - General Info:**

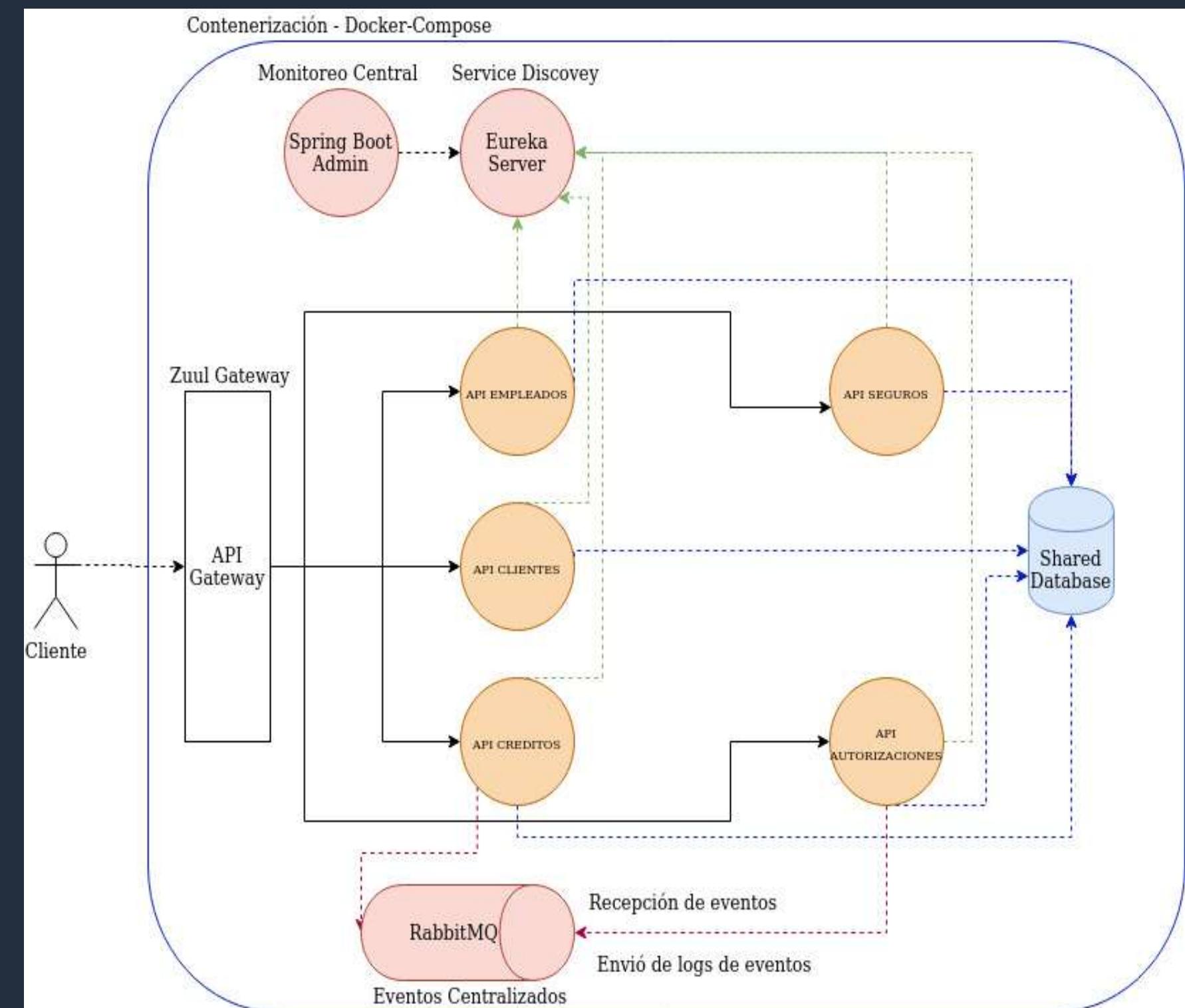
**Right Screenshot (Main Spring Boot Applications Dashboard):**

- Spring Boot® applications:**
  - Filter: (empty)
  - Table:

| Application ▲ / URL   | Version | Info |
|---|---------|------|
| api-administracion (3fc80c43)<br><a href="http://e5738aa422ad:8082/">http://e5738aa422ad:8082/</a>    |         |      |
| api-clientes (64c3c3f0)<br><a href="http://bfaf37f5eb7f:8084/">http://bfaf37f5eb7f:8084/</a>          |         |      |
| api-creditos (dcfaa1f7)<br><a href="http://c0541f6deb83:8085/">http://c0541f6deb83:8085/</a>          |         |      |
| api-empleados (4c7436ba)<br><a href="http://b10e910f8655:8083/">http://b10e910f8655:8083/</a>         |         |      |
| api-seguros-guadalupe (0411adfd)<br><a href="http://16c84e94c7ad:8081/">http://16c84e94c7ad:8081/</a> |         |      |

# Arquitectura de Microservicios

**API Gateway**  
**Shared database**  
**Microservice per container**  
**Event Sourcing**  
**Service Discovery**  
**Monitoring**

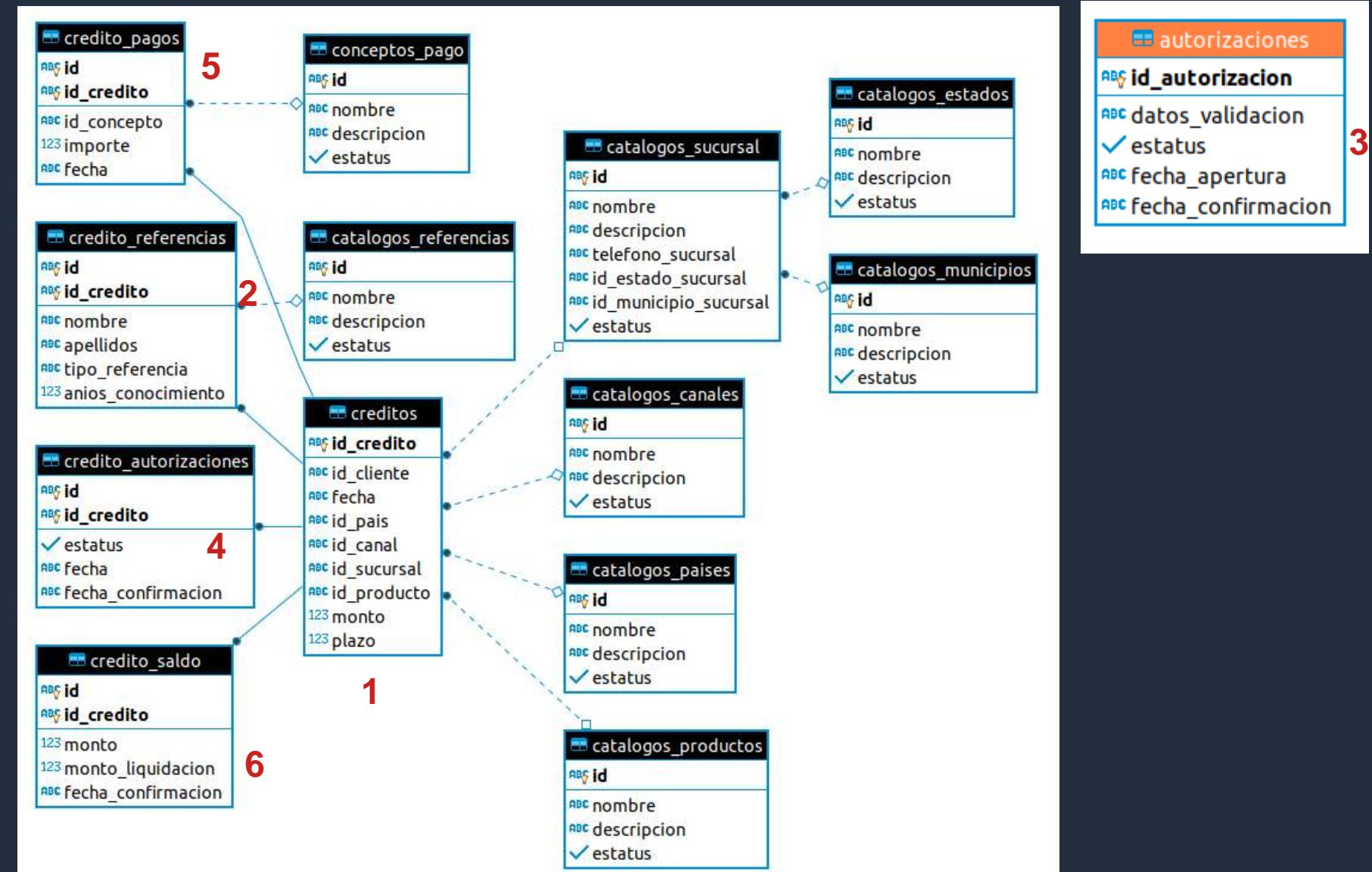


# Arquitectura de Microservicios

## Modelo de datos de microservicios

Postman:

API Empleados POST 201  
API Clientes POST 201  
API Creditos post 201  
API Creditos GET by ID 200  
API AUTORIZACION get 200  
API Adm PUT 200 Confirmacion  
API Creditos aprobaciones 200  
API Creditos aprobaciones 200 ByID  
API Creditos pagos post 201  
API Creditos pagos todos get 200  
API Creditos saldos 200  
API Creditos PUT 200  
API Creditos delete 200



# Configuración – Microservicios Banco Poder

Eureka Server  
<http://localhost:8761>

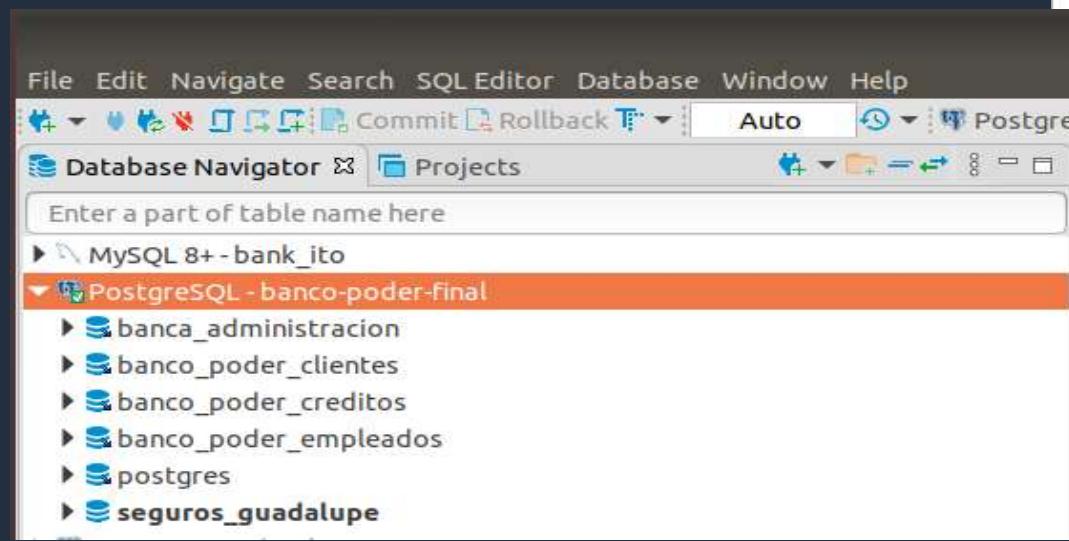
Spring Boot Admin  
<http://localhost:8090>

Zuul Gateway  
<http://localhost:8766>

RabbitMQ  
<http://localhost:15672>

usr: guest  
Pwd: guest

Postman:  
[/comunicacion\\_gateway](/comunicacion_gateway)



```
docker-compose.yaml      application.yml      run-local.sh
03_microservices > lab8_microservicios_banco_poder_gateway > gateway > src > main

46 zuul:
47   ignoredServices: '*'
48   host:
49     connect-timeout-millis: 20000
50     socket-timeout-millis: 20000
51
52 routes:
53   api-seguros-guadalupe:
54     path: /seguros/v1/**
55     serviceId: api-seguros-guadalupe
56     stripPrefix: false
57     sensitiveHeaders:
58
59   api-administracion:
60     path: /administracion/v1/**
61     serviceId: api-administracion
62     stripPrefix: false
63     sensitiveHeaders:
64
65   api-empleados:
66     path: /empleados/v1/**
67     serviceId: api-empleados
68     stripPrefix: false
```