Modernización de aplicaciones en organizaciones complejas

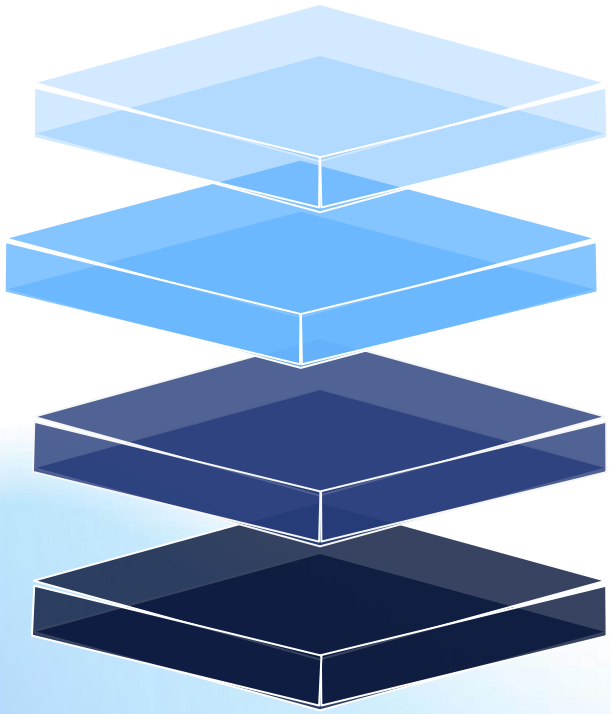
Cómo modernizamos la arquitectura para modernizar el negocio

Salva Bosque
Senior Business Development Manager
SEIDOR Opentrends

Modernizar la arquitectura para modernizar el negocio

¿Cómo lo logramos en organizaciones complejas?

¡No sólo con tecnología!



ORQUESTANDO
Gestionando Oficinas de Arquitectura y CCoEs en complejos ecosistemas **multiproveedor (+100 personas)**

OPTIMIZANDO
Optimizando la **rentabilidad y el ahorro de costes** en transformaciones cloud de gran volumen **(+1000 servidores)**

ESCALANDO
Desplegando grandes proyectos Journey to Cloud, Modernización y Transformación Digital en rollouts **multinacionales (+20 países)**

EFICIENTANDO
Potenciando la **automatización** con Dev*Ops, IAOps, Platform Engineering y GenAI **(+10% mejora productividad)**

EJEMPLOS DE CLIENTES

MAPFRE
CaixaBank
Ajuntament de Barcelona
Generalitat de Catalunya
zoetis
GB FOODS

Caso de éxito: Ayuntamiento de Barcelona

Cómo modernizamos la arquitectura con Openshift

Ayuntamiento de Barcelona

Modernización Plataforma con Openshift

RETO

Modernizar aplicaciones legacy utilizando una arquitectura SDA y pasar a **microservicios, contenedores, Kubernetes y Openshift** para fortalecer una arquitectura autoescalable para una SmartCity

SOLUCIÓN

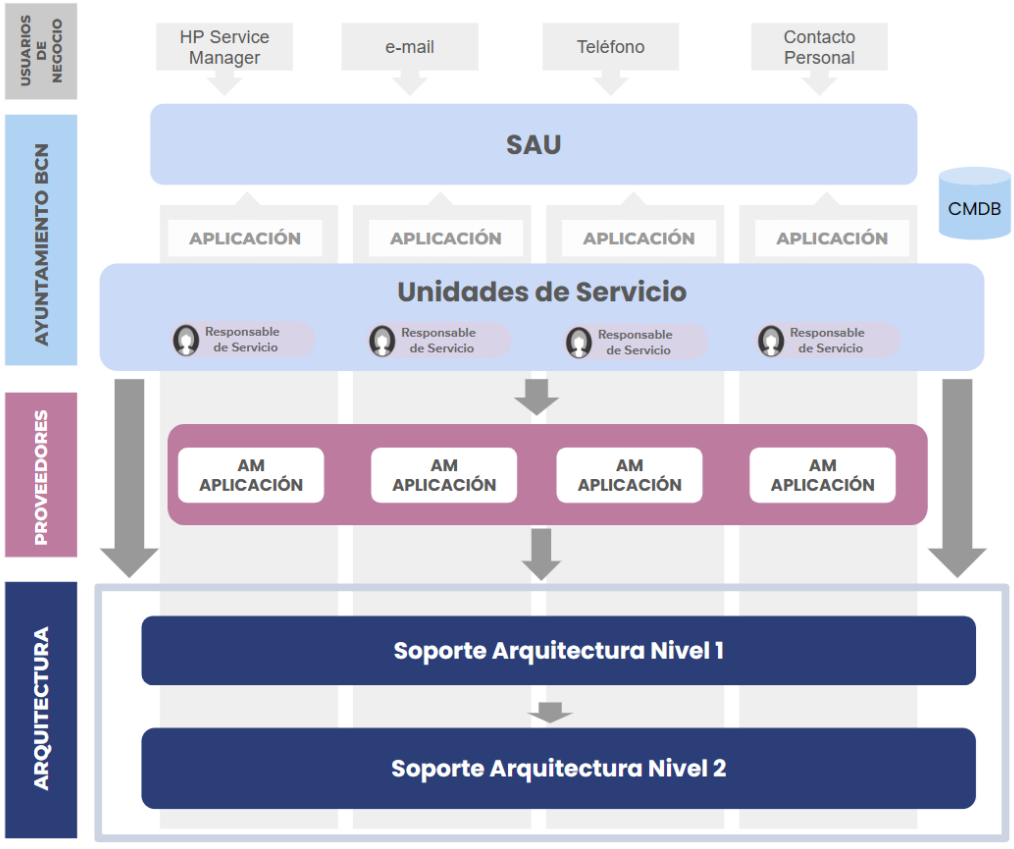
Desde 2012, nuestra **Oficina de Arquitectura** definió e implementó la hoja de ruta de arquitectura de modernización hasta el modelo actual con Kubernetes y Openshift.

RESULTADOS

- **ARQUITECTURA**: **Evolución Frameworks** (openFrameIMI) y **arquitecturas SPA Openshift, kubernetes**,...
- **REUSABILIDAD**: Creación de **Módulos Arquitectónicos Comunes:** accesibles desde todas las plataformas IMI:
- **SEGURIDAD**: Gestión de identidad, autorizaciones y permisos. **Firma electrónica y autenticación digital** de la firma
- **CONSOLIDACIÓN DE PLATAFORMA**: **Devops y ALM** para la gestión del ciclo de vida de aplicaciones: Gitlab, Jenkins, Nexus, Sonar...
- **ESCALABILIDAD**: Platform **Orchestrator** para coordinar todas las herramientas DevOps, ALM, Containers y Openshift

Ayuntamiento de Barcelona

Modernización basada en 2 pilares: modelo organizativo y arquitectura tecnológica



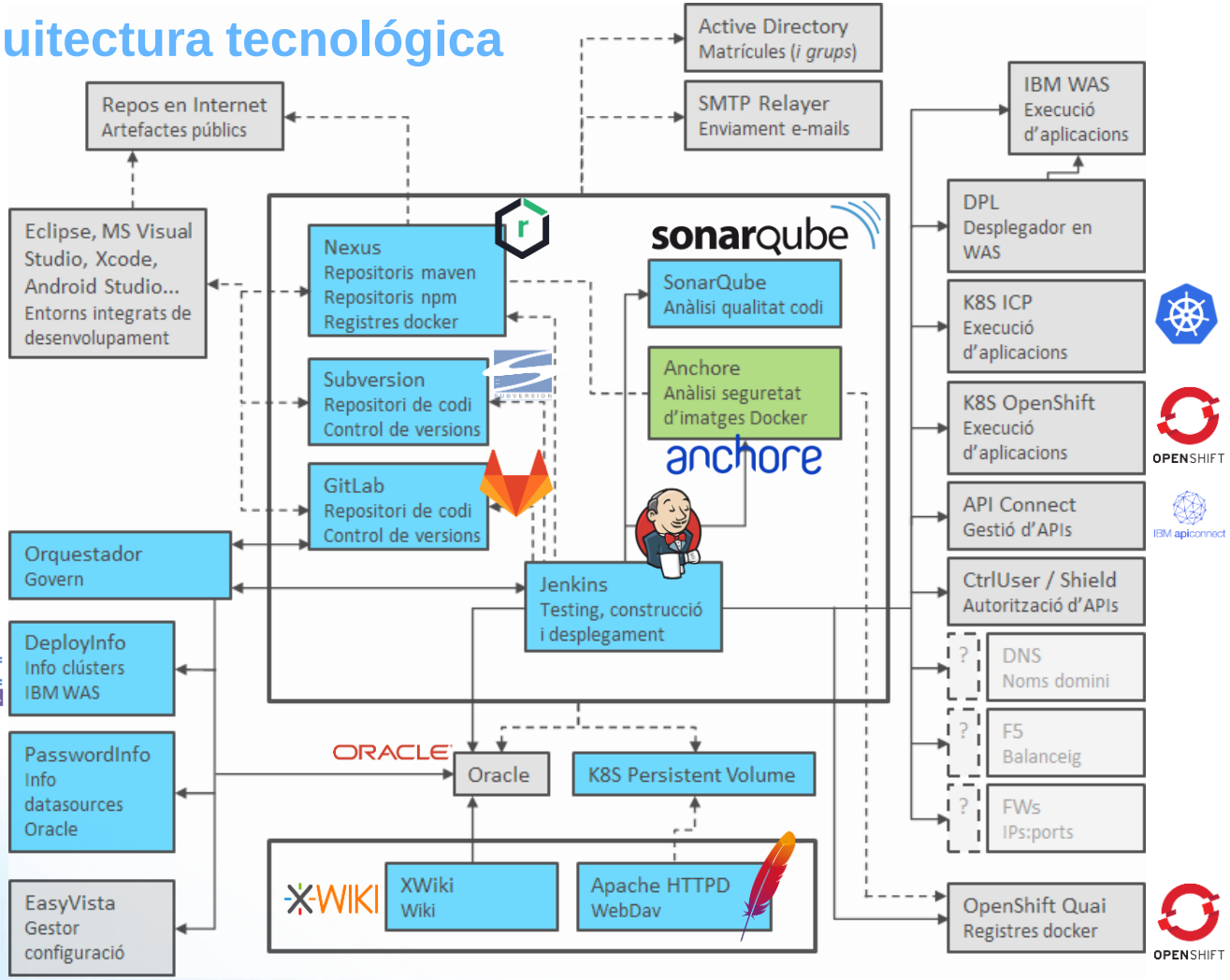
Red Hat OpenShift
kubernetes
IBM WebSphere
ORACLE
Java J2EE
Microsoft .NET
opentext™
documentum
android
iOS
IBM API Connect
SAP

Ayuntamiento de Barcelona

Modernización Arquitectura tecnológica

Nexus publica resultados y administra imágenes de Docker en OpenShift para **mejorar la tolerancia a fallos.**

Orchestrator es la pieza central que **gestiona todo el ciclo de vida de las aplicaciones conteneirizadas en Openshift**. Define cómo construir, implementar en Kubernetes, publicar APIs y ejecutar scripts.



OpenShift estandariza el **autoscalado** y despliegue mediante plantillas y gestión de configuración con ConfigMaps y Secrets. despliegues continuos y la gestión completa del ciclo de vida de las aplicaciones.

Anchore se centra en el **análisis de seguridad de imágenes de Docker**, garantizando que sean seguras y libres de vulnerabilidades.

Jenkins, con su shared lib para DevOps centraliza la **automatización de testing, builds y deploys**, junto con SonarQube para análisis de la calidad del código

Modernización con Openshift en el Ayuntamiento

Ventajas de la arquitectura tecnológica

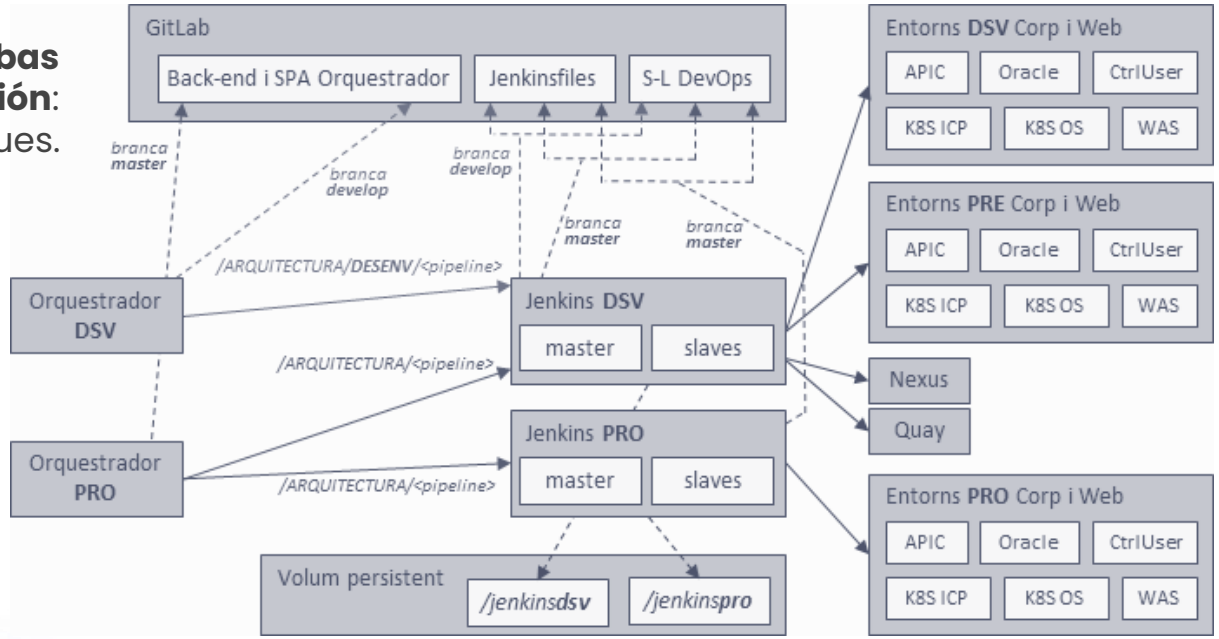
Flexibilidad para reconfigurar en caliente: Ajustar rápidamente el escenario necesario.

Entornos de desarrollo diferenciados: Evitar afectar producción durante el desarrollo y pruebas.

Desarrollo y pruebas sin afectar la producción: Cambios sin impactar los despliegues.

Plataformas Jenkins en Openshift independientes: Segregación y control.

Dos orquestadores independientes: Parametrizar en caliente las URLs de Jenkins en cada pipeline para flexibilidad y adaptabilidad.



Despliegues en entornos no productivos sin impacto ni afectación desarrollos vs producción.

Escalabilidad por separado: Mejor eficiencia.

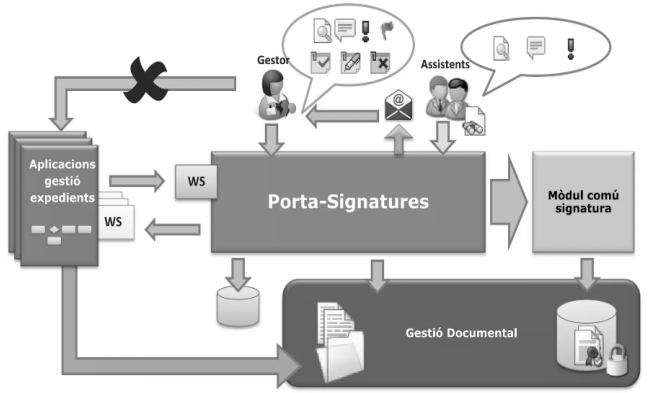
Mejoras en la tolerancia a fallos: Volumen persistente único para distribuir controlador y agentes en nodos diferentes

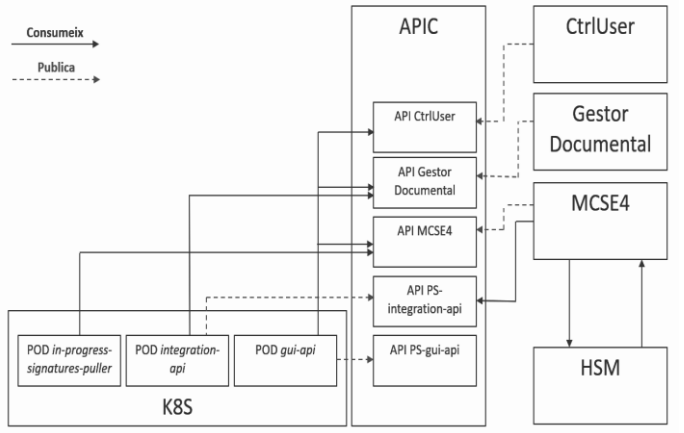
Construcción de versiones sin afectar despliegues: Evitar ralentizar los despliegues.

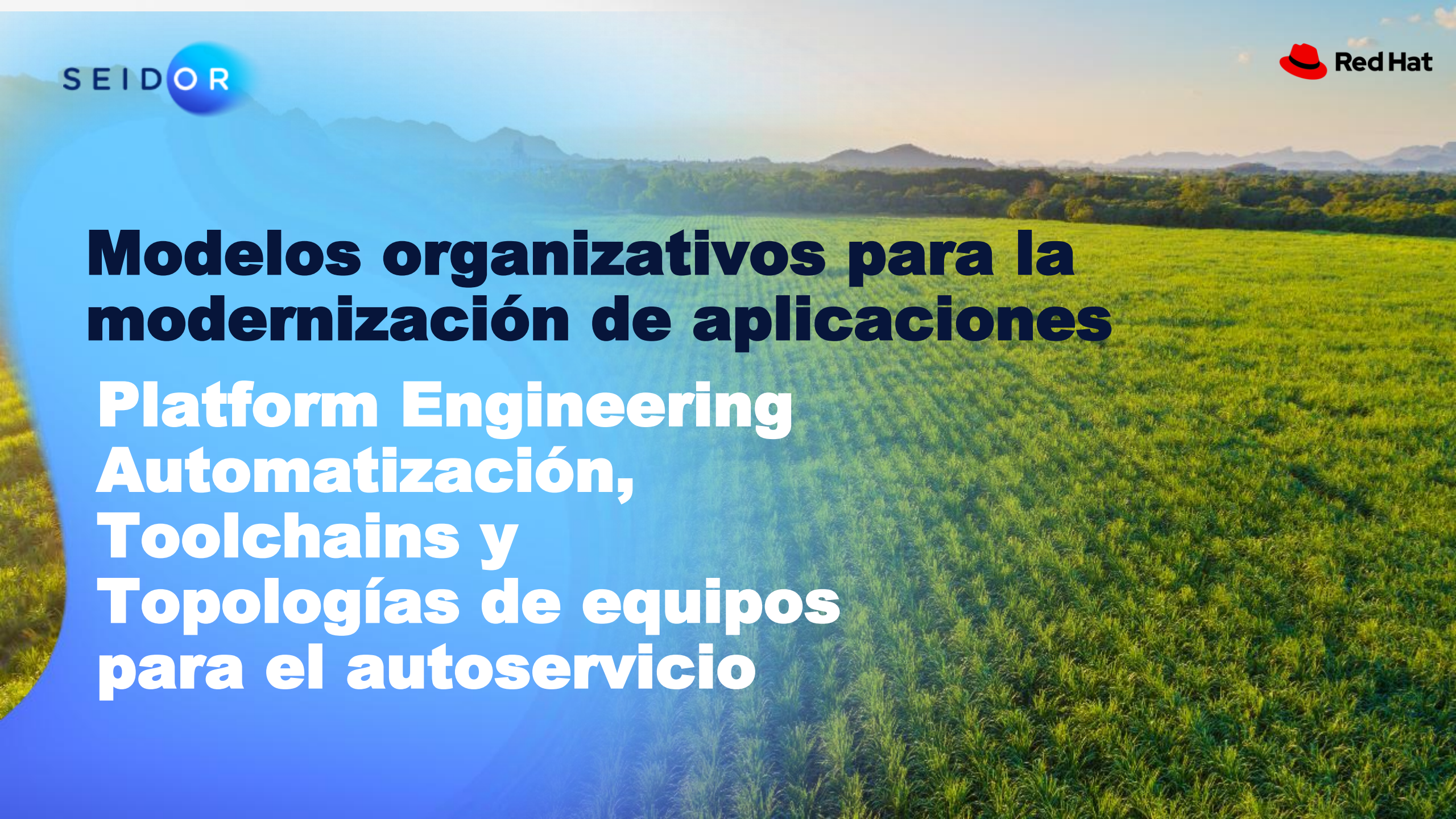
Caso práctico Ayuntamiento de Barcelona

Aspectos clave de la Modernización a Openshift

- **Assessment aplicaciones** existentes para la adopción de microservicios
- **Contenerización**: Realiza la contenerización de las aplicaciones, lo que significa empaquetarlas con sus dependencias para asegurar la consistencia en todos los entornos de ejecución, desde el desarrollo hasta la producción.
- Implantar estrategias para la **gestión de datos** en un entorno de microservicios.
- **Automatización** de despliegues: CI/CD para automatizar las pruebas y los despliegues,
- Estrategias de **escalado**: Aprovechar las funcionalidades de autoscaling de OpenShift para gestionar dinámicamente los recursos según la demanda
- **Seguridad**: Implementar políticas de seguridad robustas utilizando las funcionalidades de OpenShift, como el control de acceso basado en roles (RBAC), Secrets
- **Monitoreo** y registro
- **Formación y cultura organizacional:**
- Procesos diferidos de feedback a las aplicaciones cliente
- Estrategia de **despliegue** y puesta en marcha






SEIDOR
Red Hat
Modelos organizativos para la modernización de aplicaciones
Platform Engineering
Automatización,
Toolchains y
Topologías de equipos
para el autoservicio

Platform Engineering

¡No es hype! Llevamos haciéndolo hace años en IMI, Caixabank, Generalitat…

Dev*Ops

Chef que prepara el menú en el desarrollo y la operación del día a día, **automatizando pruebas de calidad, seguridad**

Utiliza los ingredientes disponibles en la cocina

Platform Engineering

Diseña, implanta y **mejora la cocina** con autoservicio y automatización

Ofrece servicios, **diseña flujos de trabajo y habilita toolchains para los equipos** de desarrollo, operaciones y Dev*Ops con pipelines e infrastructuras que facilitan el autoservicio

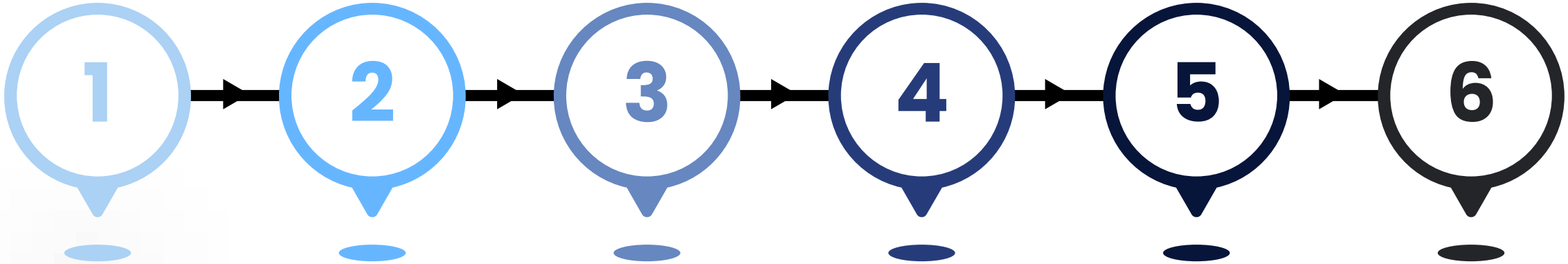
SRE

Foco en que la **cocina y restaurante estén siempre en marcha**

Monitorización, confiabilidad y escalabilidad, habilitando herramientas de uso interno para mejorar la disponibilidad y la colaboración

Platform Engineering

Pilares



Seguridad

Es clave facilitar el **autoservicio** para la creación de cuentas, la configuración de credenciales y las claves de API.

Pipeline

Facilitar curvas de aprendizaje, potenciar paradigma **as-a-Code y automatización**

Aprovisionamiento

Políticas de **aprovisionamiento as-a-Code**

Conectividad

Service Discovery, Service Mesh

Orquestación

Determinar la forma más **eficiente de asignar cargas de trabajo**

Observabilidad

Anticiparse de manera proactiva y preventiva a las incidencias

Platform Engineering

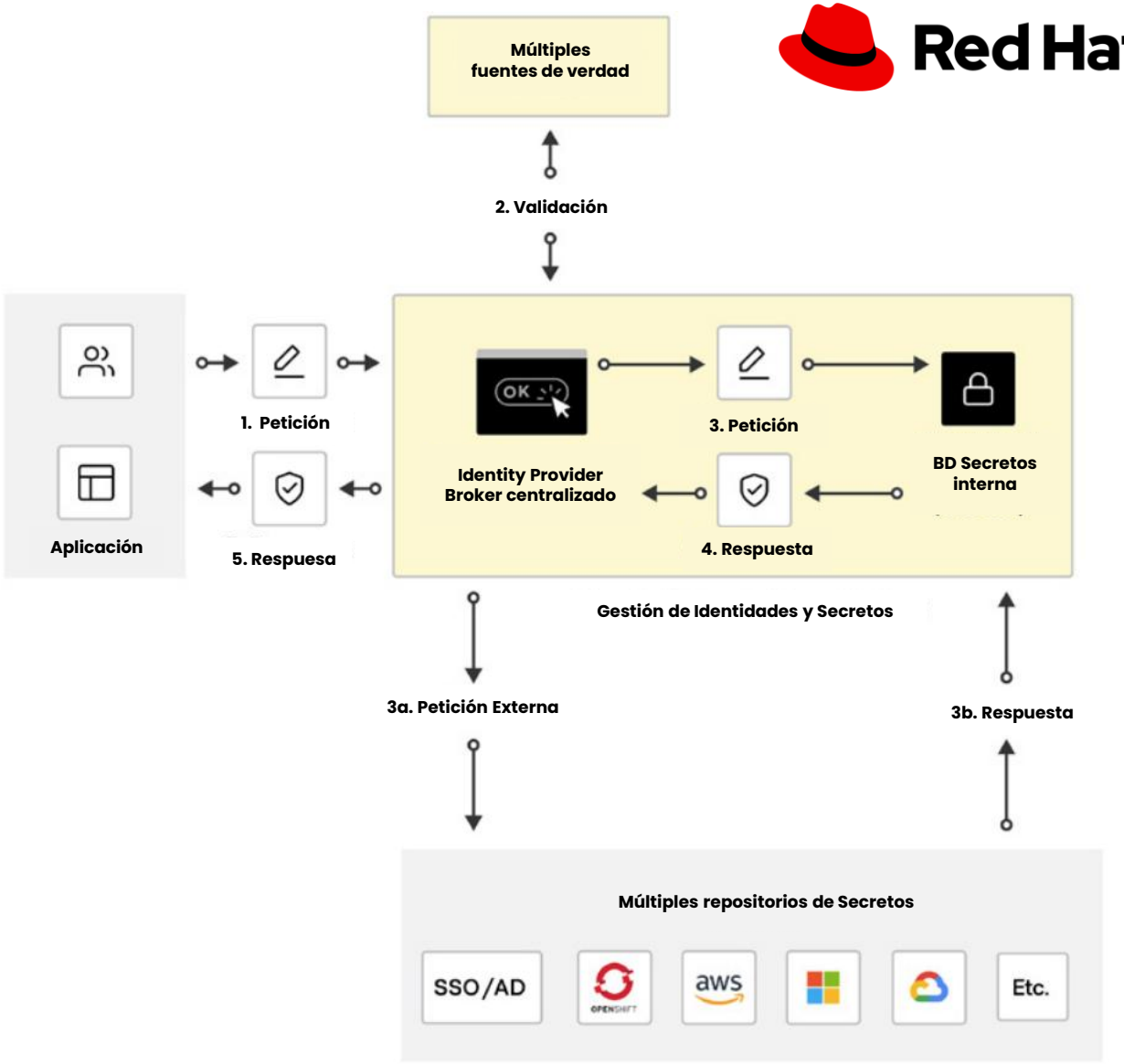
Seguridad: en todo el ciclo de vida

Facilitar el **autoservicio** para la creación de cuentas, la configuración de credenciales y las claves API.

El control de versiones, la integración continua y la provisión de infraestructura requieren prácticas seguras desde el principio.

Algunas organizaciones se centran en seguridad basada en el perímetro de red (el enfoque “castillo y foso”), pero las infraestructuras dinámicas requieren **seguridad más avanzada basada en la identidad, gestión centralizada de credenciales** y flujos de trabajo de cifrado.

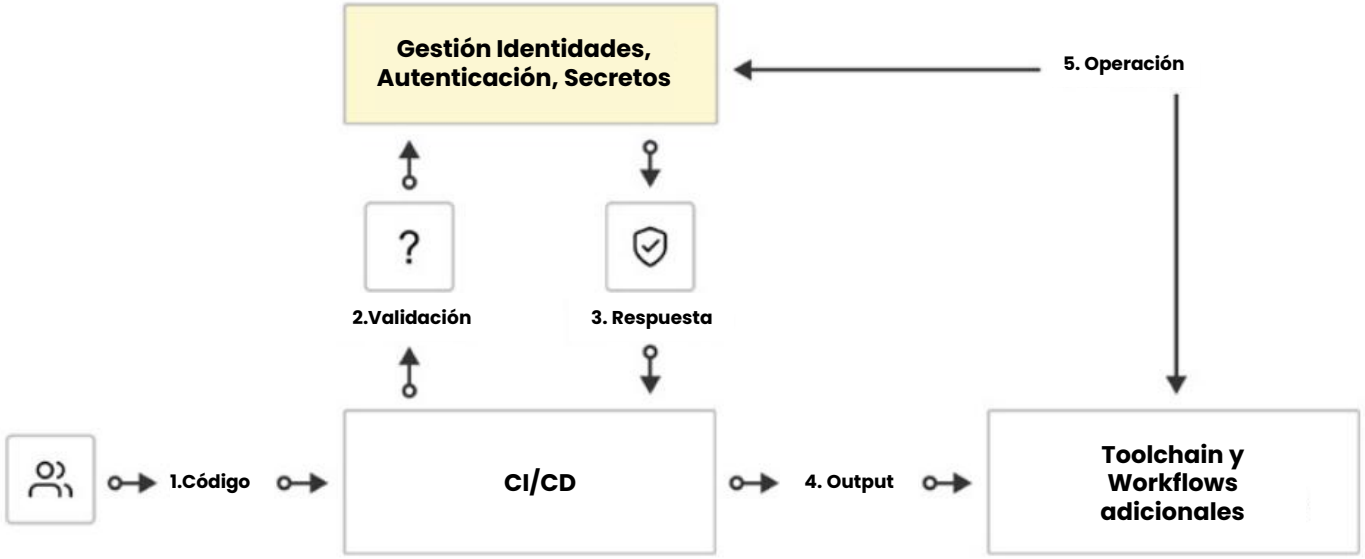
Priorizar la seguridad integrando **controles en todo el ciclo de vida del desarrollo de software.**



Platform Engineering

Pipeline: factores clave

- **Integración sencilla: Facilita la incorporación de nuevos miembros al equipo**
- **Curva de aprendizaje suave**: poco entrenamiento adicional con herramientas estándar
- Soporte para **pipelines como código**: Automatiza procesos.
- **Agnosticismo de plataforma** (basado en API): Funciona en diferentes plataformas.
- **Controles de seguridad integrados** (RBAC, auditoría, etc.)
- **Configuración automatizada** (infraestructura como código, runbooks): Simplifica la administración.
- **Gestión de secretos, identidad e integración de autorización:** Asegurar el acceso adecuado de los equipos



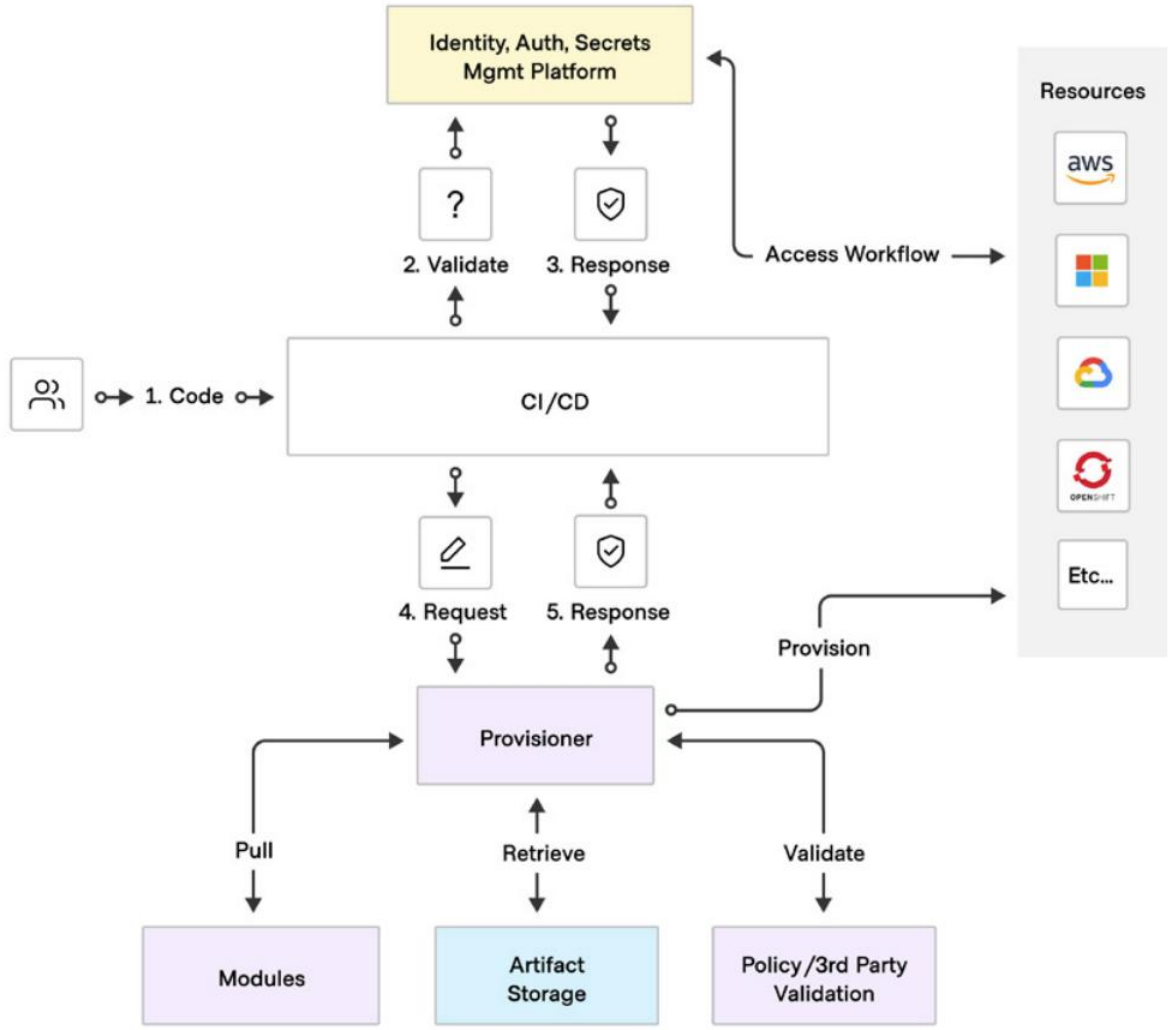
Platform Engineering

Aprovisionamiento “Policy as a code”

La provisión de infraestructura ha experimentado un cambio significativo debido a las prácticas ágiles de desarrollo. Antes era considerada un problema operativo, ahora se espera que la **provisión de infraestructura sea parte integral de la entrega de aplicaciones.**

Históricamente, el personal de operaciones aplicaba flujos de trabajo y procesos mediante ***tickets***, que incluían validación de acceso, aprobaciones, seguridad y costos. Todo el proceso se auditaba para cumplir con las prácticas de cumplimiento y control.

Ahora es crucial permitir que los desarrolladores y otros usuarios finales de la plataforma realicen la **provisión a través de flujos de trabajo de autoservicio**. Esto implica implementar un conjunto de controles de seguridad codificados **“policy as a code”**



1 2 3 4 5 6

Platform Engineering

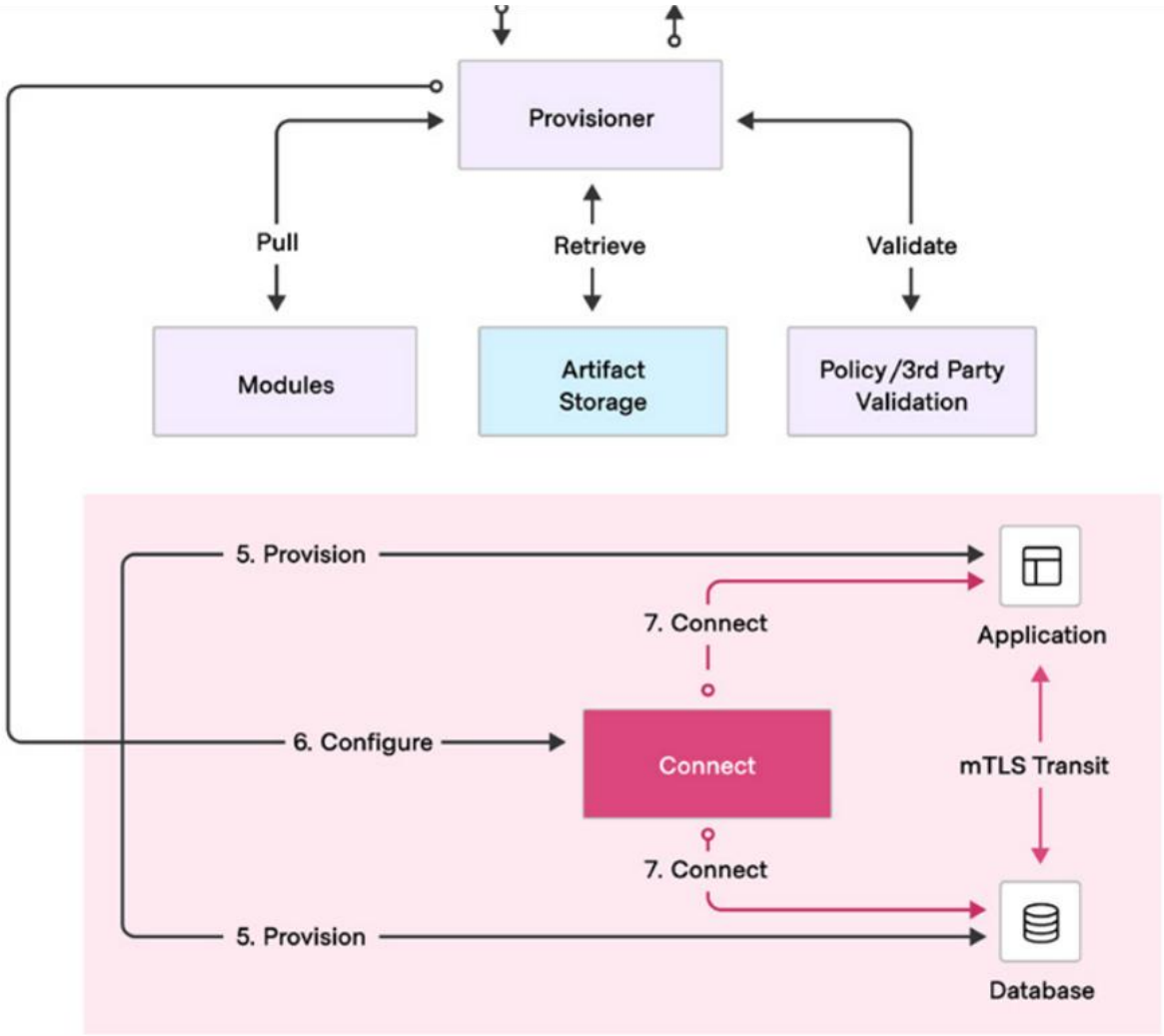
Conectividad: mejorar el workflow

A menudo pasada por alto, la red es un aspecto crucial de Platform Engineering. Los procesos tradicionales basados en tickets para tareas como la creación de entradas DNS y las actualizaciones de firewall causan demoras, errores manuales e ineficiencias y para ello se requiere:

Automatización: incorporar funciones de red en las configuraciones de infraestructura como código, beneficiándose de la automatización, la confiabilidad y el control de versiones.

Microservicios, Service Discovery y Service Mesh: arquitecturas de microservicios con descubrimiento de servicios permite conexiones automáticas basadas en políticas centralizadas en una red de confianza.

Registro Compartido Central: multi-nube, multi-región y multi-runtime para una conectividad óptima.

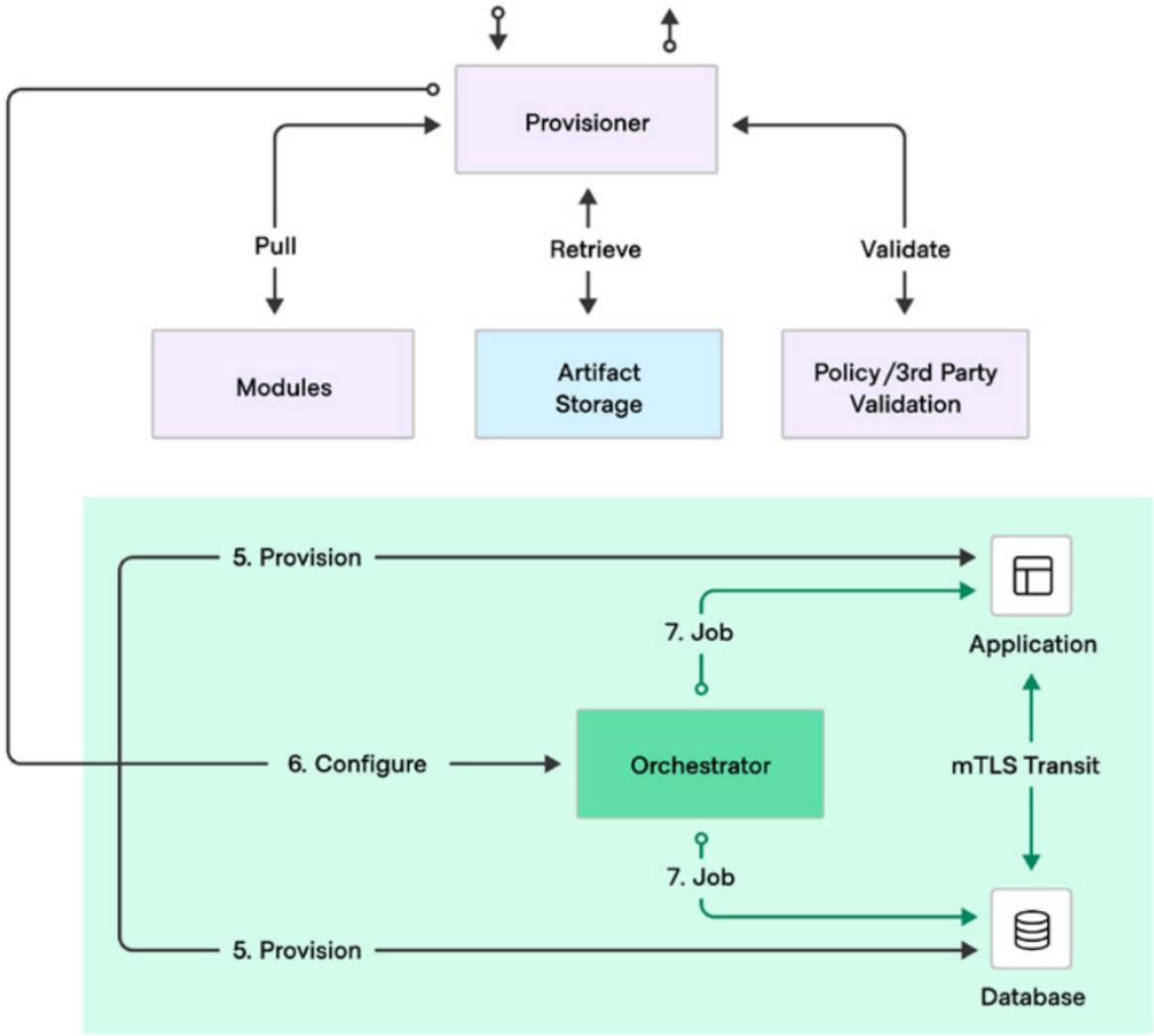


Platform Engineering

Orquestación

Los orquestadores proporcionan **algoritmos de optimización para determinar la forma más eficiente de asignar cargas de trabajo** a la infraestructura (por ejemplo, escalado automático, dimensionamiento de aplicaciones dinámico, etc.), lo que puede reducir los costos y mejorando la utilización de recursos

- **Escalabilidad y elasticidad**
- Soporte **multi-nube y nube híbrida**
- **Autoservicio** para desarrolladores
- **Descubrimiento de servicios y redes** (integrados o personalizables)
- Alta disponibilidad y tolerancia a fallos
- Control avanzado de programación y ubicación
- **Aislamiento y seguridad** de recursos



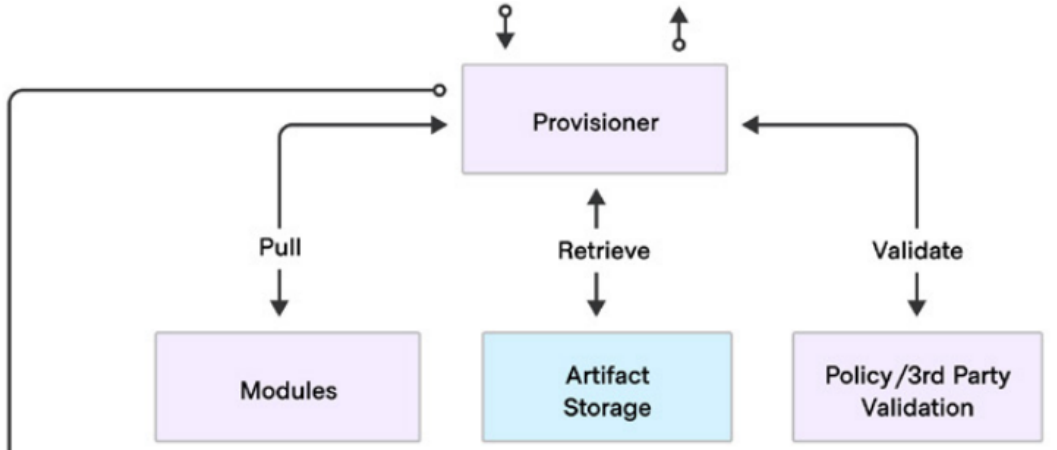
1 2 3 4 5 6

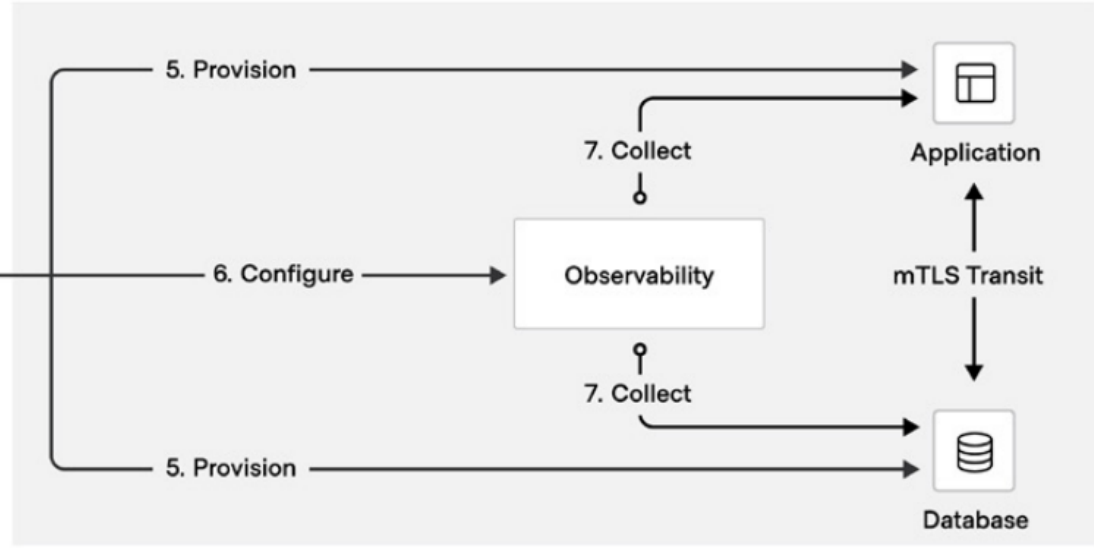
Platform Engineering

Observabilidad

La observabilidad es crucial para comprender y obtener información sobre los estados internos y comportamientos de las aplicaciones y poder anticiparse de manera proactiva y preventiva a las incidencias:

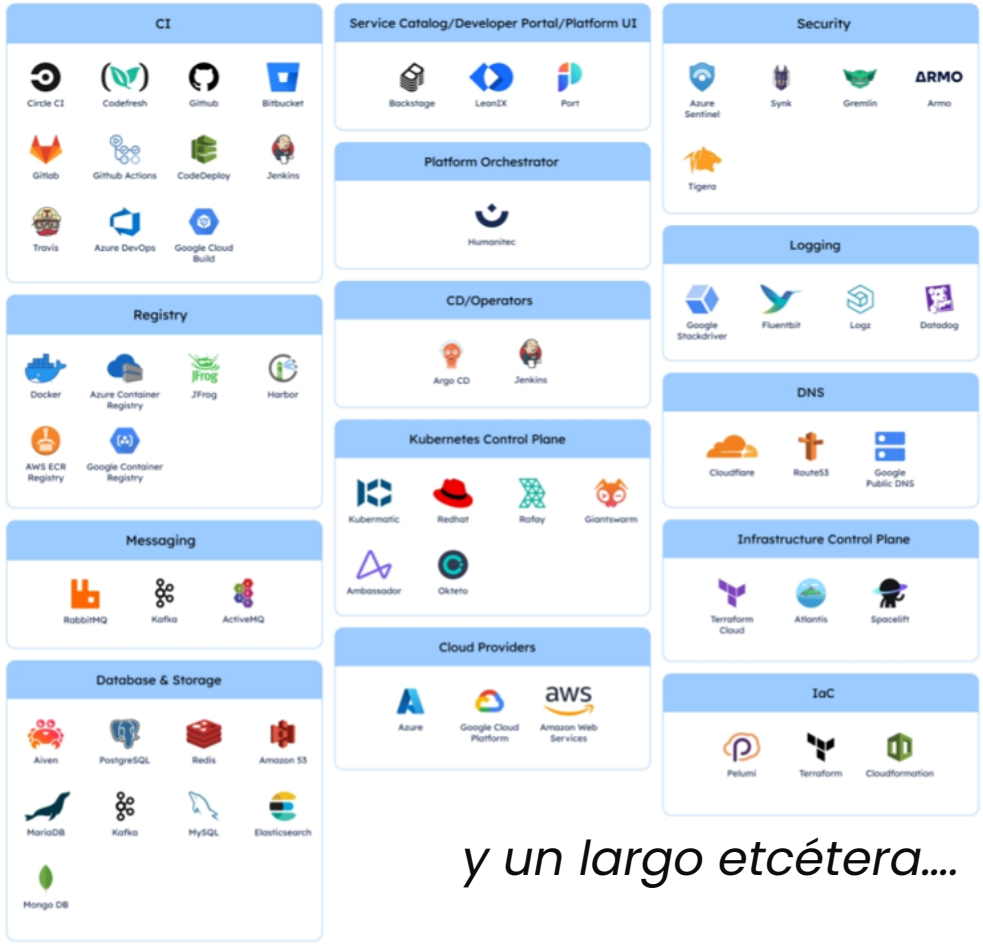
- **Mejora de la supervisión:** Detecta problemas antes de que afecten al sistema.
- **Resolución de problemas eficiente:** Facilita la identificación de errores y su corrección.
- **Experiencia del usuario optimizada:** Garantiza un rendimiento óptimo.
- **Mayor productividad:** Ayuda a los equipos a colaborar y comprender las interdependencias de los servicios en la nube





Platform Engineering

Toolchains



y un largo etcétera....

Pero hay algo...
que las herramientas no automatizan...

¡las personas!

SEIDOR

Platform Engineering

Cultura organizativa

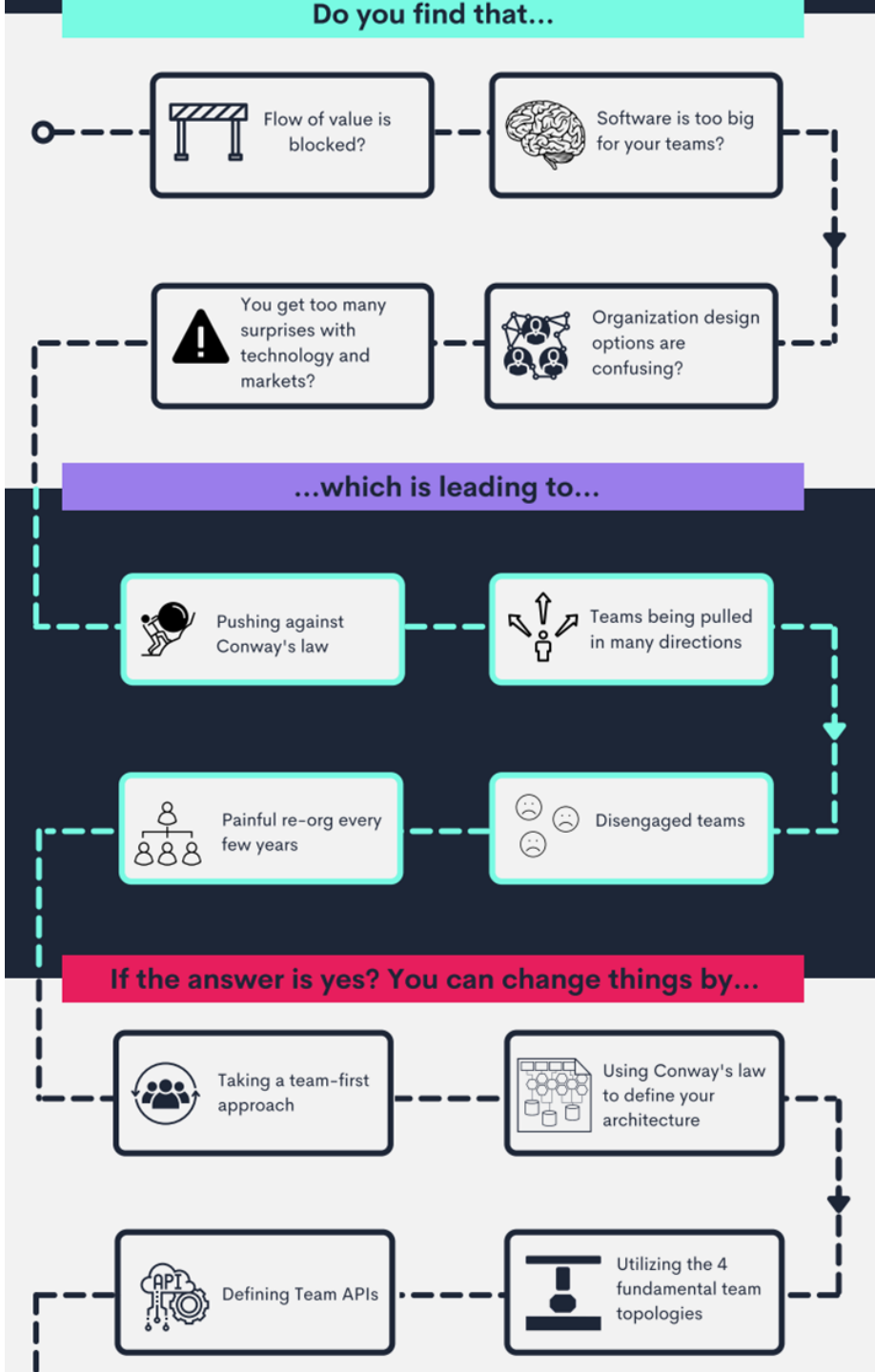
Team Topologies es uno de los métodos para el diseño organizacional que **complementa Platform Engineering**, optimizando las interacciones entre equipos.

Este enfoque ayuda a los CIOs a crear estructuras que permitan a los equipos de tecnología trabajar de manera más efectiva y responder rápidamente a los cambios del mercado y las demandas de los clientes.

Al proporcionar servicios internos con una **mentalidad de producto**, pueden mejorar enormemente la eficiencia operativa y la entrega de software. Esto implica tratar la **plataforma como un producto** para satisfacer las necesidades de sus clientes internos (los equipos de desarrollo) redudando en la **mejora de la experiencia de los clientes finales**

Team Topologies

Organizing business and technology teams for fast flow



El rol de AI en la modernización

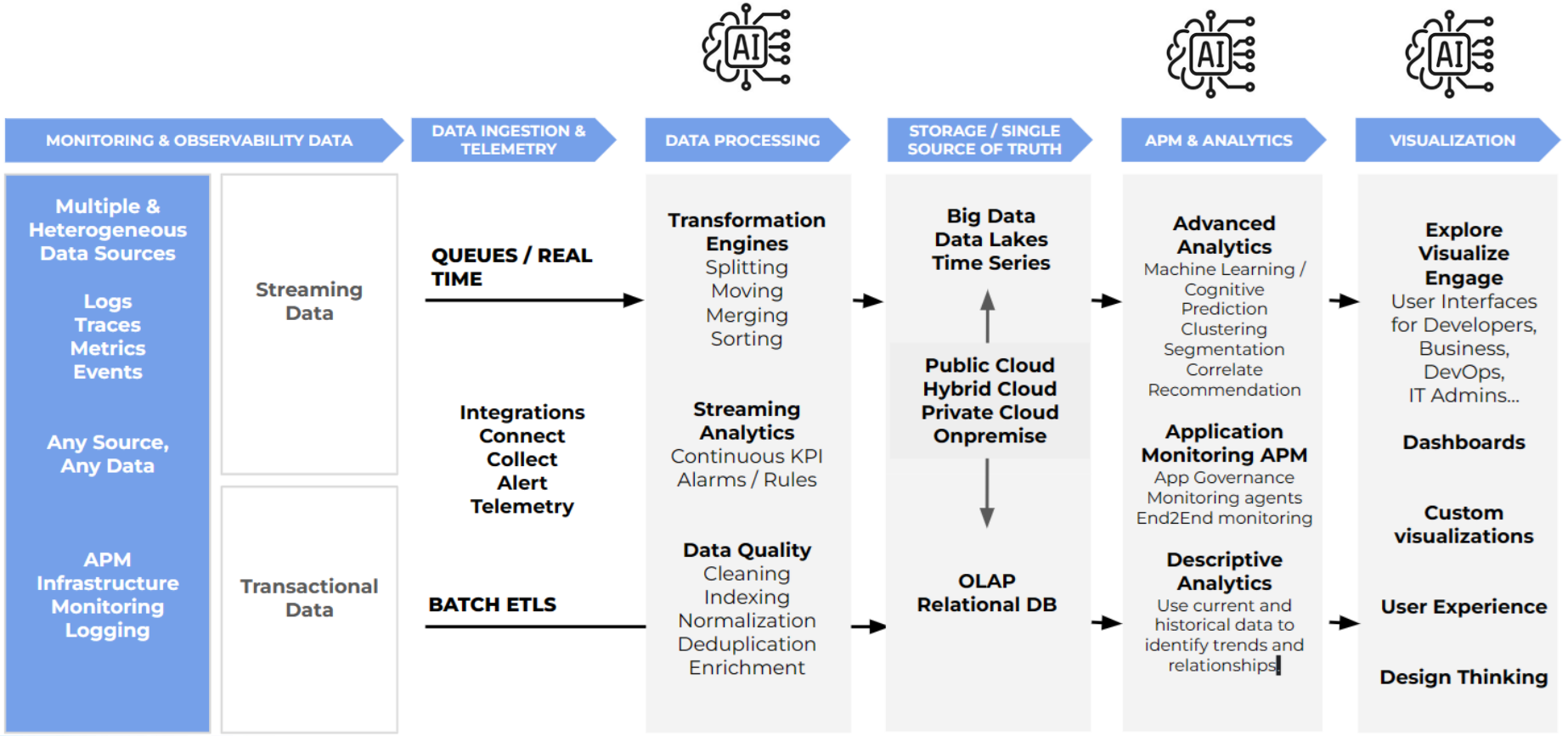
Casos de uso para Platform Engineering & SRE

Detección preventiva de incidencias en el proceso de modernización

Detectar **anomalías en base a los datos recogidos en el ciclo de vida de las aplicaciones**, y reaccionar de manera anticipada

Automatización operaciones y analítica del ciclo vida desarrollos

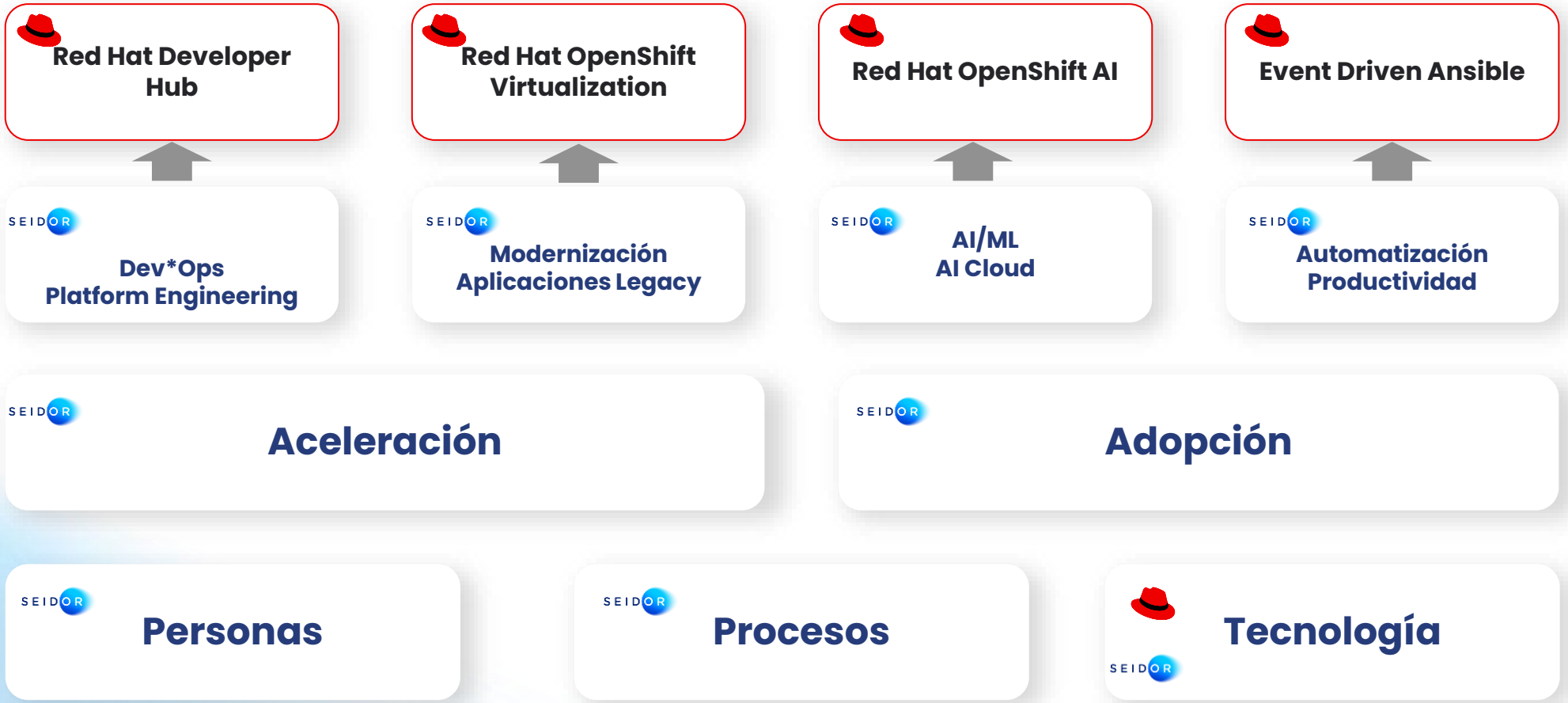
APM (**Aplication Performance Monitoring) con mayor proactividad** y anticipación




SEIDOR
Red Hat
Mapa
Modernización
Complementando el
Ecosistema
Red hat & SEIDOR

Innovación y complementareidad

Todos lo necesario para la agilidad, escalabilidad y modernización



SEIDOR

Red Hat

Gracias!

93 304 32 22
info@seidor.com
Company/seidor
@seidor