Introducción a la Arquitectura de Microservicios (MSA)

Administración de Recursos 2020

Temario

1. Introducción.
2. Enfoque monolítico vs enfoque microservicios.
3. Características de la arquitectura de microservicios.
4. Ventajas de la arquitectura de microservicios.
5. Desventajas de la arquitectura de microservicios.
6. ¿Cuándo utilizar una arquitectura de microservicios?
7. Patrones relacionados.

Microservicios

Estilo de arquitectura en el que una aplicación se desarrolla como un conjunto de servicios que:

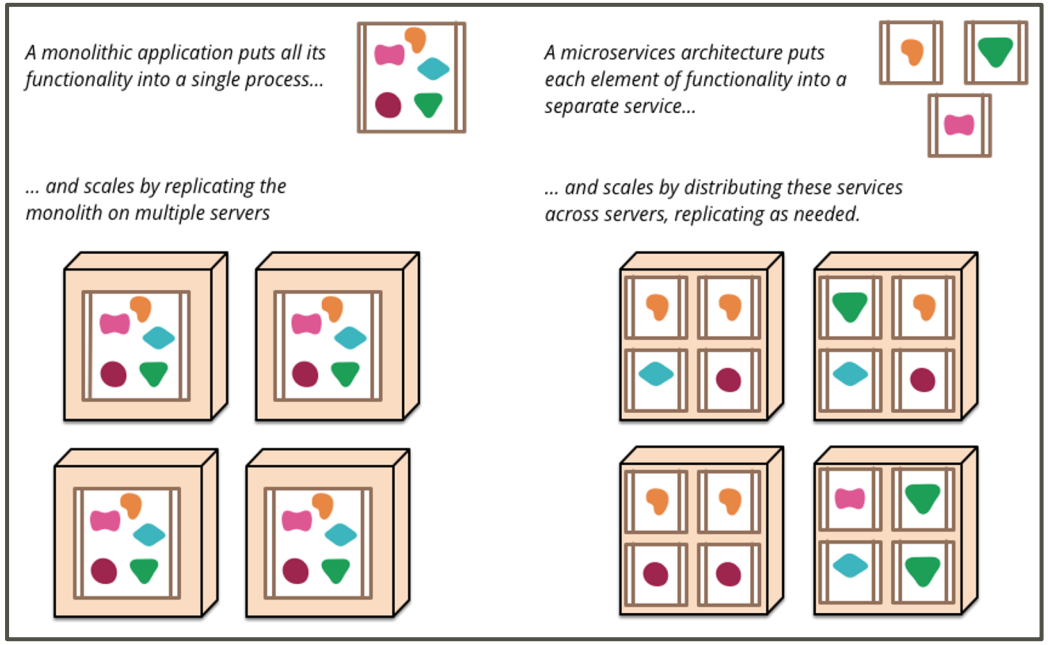
- Ejecutan en su propio proceso y se comunican con mecanismos ligeros.
- Se construyen en torno a capacidades de negocio.
- Pueden desplegarse de forma independiente mediante procesos automatizados.
- Poseen una mínima gestión centralizada.
- Pueden escribirse en diferentes lenguajes de programación y utilizar diferentes tecnologías de almacenamiento de datos.

Monolítico vs Microservicios

Estilo de arquitectura monolítico:

- La aplicación se construye como una unidad (*monolito*).
- Se utilizan las herramientas del lenguaje de implementación para modularizar la aplicación (clases, funciones, namespaces).
- Todos lo módulos ejecutan dentro de un mismo proceso y sobre un mismo hardware.
- Cualquier cambio implica construir y desplegar una nueva versión de toda la aplicación.
- El escalado requiere que escale toda la aplicación.

Monolítico vs Microservicios



Monolítico vs Microservicios

Ventajas del enfoque monolítico:

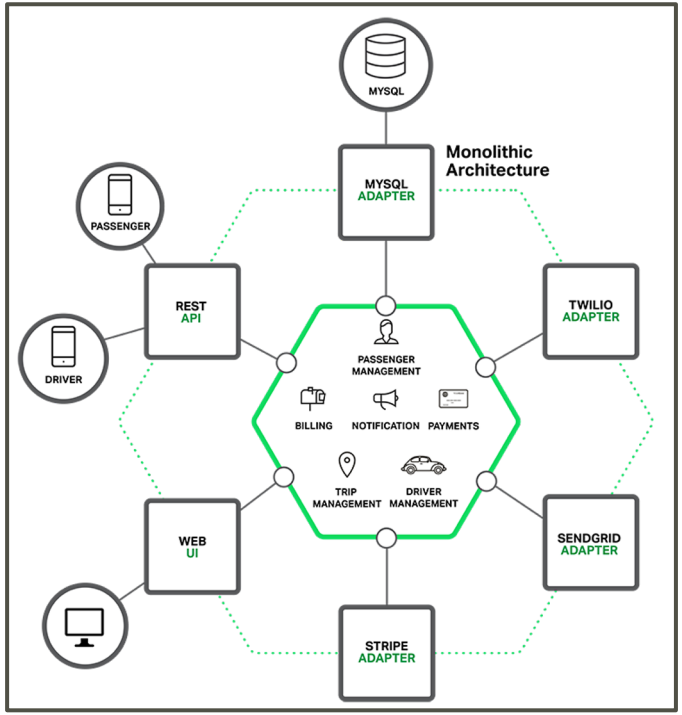
- Simplicidad.
- Facilidad de desarrollo.
- Facilidad de testeo.
- Facilidad de despliegue.
- Facilidad de operación.
- Facilidad de escalado.

Monolítico vs Microservicios

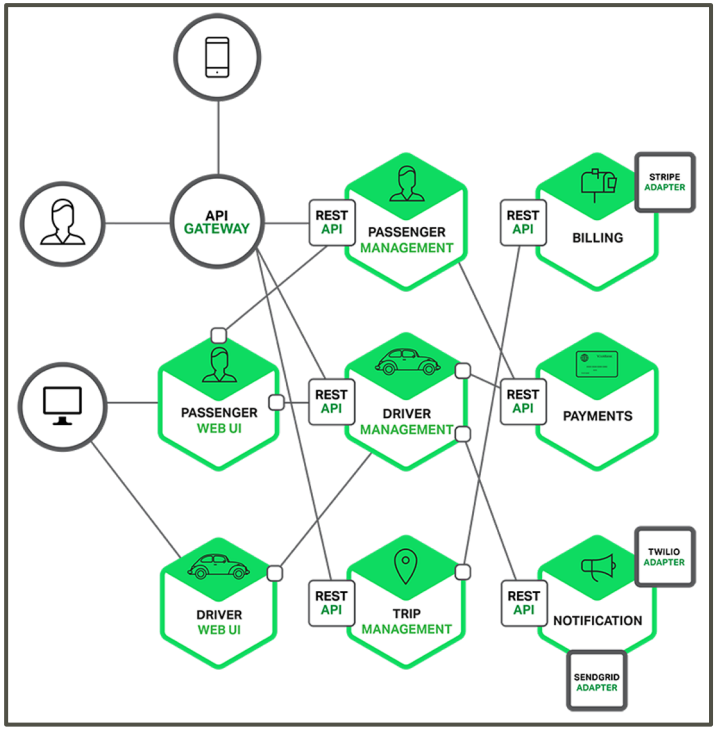
Inconvenientes del enfoque monolítico:

- Ciclos de cambios acoplados.
- A medida que la aplicación crece:
  - Aumenta la complejidad del monolito.
  - Aumenta la dificultad de mantener la modularidad.
  - Se ralentiza el tiempo de inicio de la aplicación.
  - Se dificulta la incorporación de nuevas tecnologías.
  - Disminuye la fiabilidad; un error en cualquier módulo puede potencialmente afectar la disponibilidad de toda la aplicación.
- Escalado ineficiente.
  - Escala toda la aplicación.
    - Todos lo módulos ejecutan sobre el mismo hardware.

Monolítico vs Microservicios



Monolítico vs Microservicios



MSA - Características

Principales características de una arquitectura de microservicios:

- Componentización a través de servicios.
- Organizada en torno a funcionalidades de negocio.
- Productos, no proyectos.
- Smart endpoints and dumb pipes.
- Gobierno descentralizado.
- Gestión de datos descentralizada.
- Automatización de infraestructura.
- Diseño tolerante a fallos.
- Diseño evolutivo.

MSA - Características

Componentización a través de servicios.

Las MSA utilizan bibliotecas pero su forma principal de crear componentes de su propio software es segmentarlo en servicios.

- Componente: unidad de software que se puede reemplazar y actualizar independientemente.
- Bibliotecas: componentes vinculados a un programa que se invocan mediante llamadas a funciones en memoria.
- Servicios: componentes fuera del proceso que se comunican mediante mecanismos como peticiones a web services o RPC.

MSA - Características

Componentización a través de servicios (cont).

Utilización de servicios como componentes.

- Se basa en que los servicios son independientemente desplegables.
- Resulta en interfaces de componentes más explícitas.
- Tiene algunas desventajas:
  - Las llamadas remotas son más costosas que las invocaciones en memoria.
  - Se dificulta el cambio de asignación de responsabilidades entre componentes cuando los movimientos de comportamientos cruzan los límites del proceso.

MSA - Características

Organizada en torno a funcionalidades de negocio.

Ley de Conway:

"Cualquier organización que diseñe un sistema (definido ampliamente) producirá un diseño cuya estructura es una copia de la estructura de comunicación de la organización."

- Melvyn Conway, 1967

MSA - Características

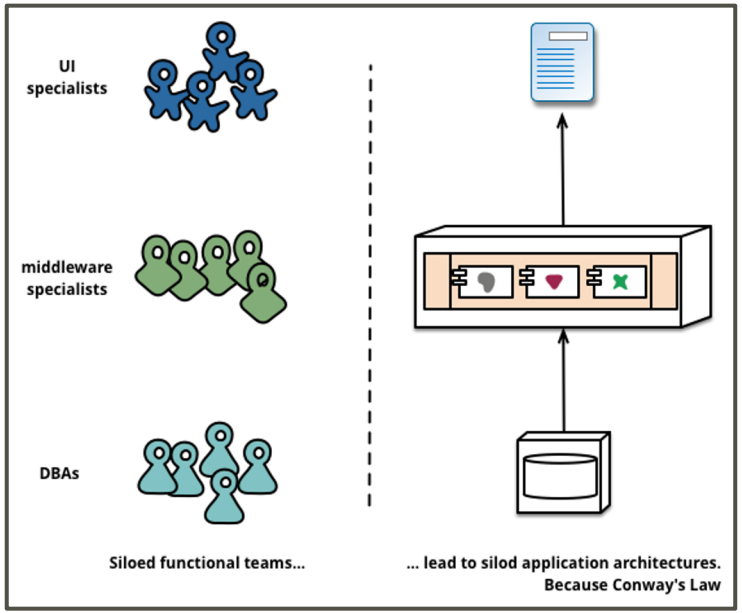
Organizada en torno a funcionalidades de negocio (cont).

Equipos funcionales aislados:

- Incluso los cambios simples pueden llevar a que un proyecto entre equipos tome tiempo y requiera aprobación presupuestaria.
- Los equipos suelen intentar optimizar su rendimiento introduciendo lógica en la parte sobre la que tienen control directo.

MSA - Características

Organizada en torno a funcionalidades de negocio (cont).



MSA - Características

Organizada en torno a funcionalidades de negocio (cont).

El enfoque en microservicios es diferente:

- Los servicios son definidos en torno a funcionalidades de negocio.
- Cada servicio requiere una implementación amplia de software para el área de negocio a la que pertenece por lo que los equipos son **multifuncionales** y poseen la gama completa de habilidades requeridas.

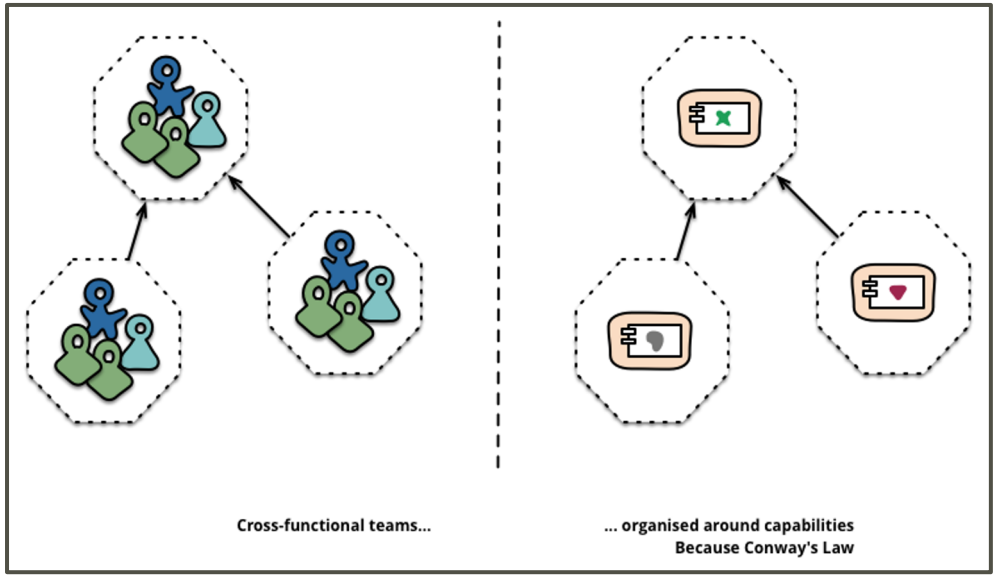
MSA - Características

Organizada en torno a funcionalidades de negocio (cont).

- Grandes aplicaciones monolíticas también se pueden modularizar en torno a funciones de negocio.
    - Las líneas modulares requieren una gran disciplina para cumplirse.
    - En arquitecturas de microservicios la separación necesariamente más explícita entre servicios componentes facilita la delimitación de los equipos.

MSA - Características

Organizada en torno a funcionalidades de negocio (cont).



MSA - Características

Productos, no proyectos.

Enfoque de proyecto:

- El objetivo es entregar una pieza de software que en algún momento se considerará terminada.

- Al finalizar, el software se asignará a una organización de mantenimiento y el equipo que lo creó se disuelve.

MSA - Características

Productos, no proyectos.

Mentalidad de producto:

- Un equipo debe poseer un producto durante toda su vida útil.
  - "*You build it, you run it*" (Amazon).
    - El equipo de desarrollo asume toda la responsabilidad del software en producción.
- Se establece una mejor relación entre el producto y la capacidad de negocio a la que pertenece.
- Se concibe al software en una relación contínua en la que se busca como éste puede ayudar a sus usuarios a mejorar sus capacidades en el negocio.

MSA - Características

Smart endpoints and dumb pipes.

- La lógica de la aplicación se encuentra en los servicios y no en los mecanismos de comunicación.
- Los mensajes son coreografiados utilizando protocolos simples en lugar de protocolos complejos como WS-Choreography o BPEL u alguna herramienta de orquestación central.
- Los protocolos más utilizados son:
    - HTTP request / response con resource APIs.
    - Lightweight messaging.

MSA - Características

Gobierno descentralizado.

- Un gobierno centralizado tiende a estandarizar una única plataforma tecnológica.
- La segmentación de una aplicación en servicios permite elegir para cada uno de ellos cuál es el stack tecnológico que mejor se adapta a sus necesidades.
- Se suelen implementar patrones de diseño específicos para gestionar los contratos de los servicios buscando reducir su acoplamiento, limitando la necesidad de un gobierno central de los contratos.
- El concepto "*You build it, you run it*" es quizás la mayor expresión de gobierno descentralizado.

MSA - Características

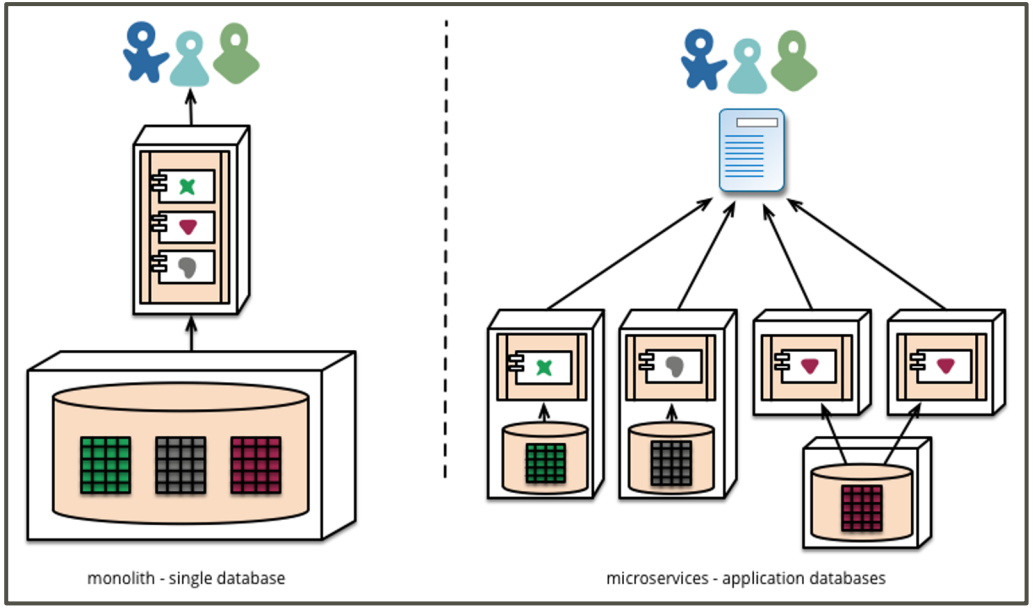
Gestión de datos descentralizada.

- Descentralización del modelo conceptual.
  - Bounded Context (DDD): Segmenta un dominio complejo en múltiples dominios delimitados y mapea las relaciones entre ellos.

- Descentralización de las decisiones de persistencia de datos.
  - Cada servicio administra su propia base de datos.
    - Diferentes instancias de una misma tecnología.
    - Persistencia políglota.

MSA - Características

Gestión de datos descentralizada (cont).



MSA - Características

Gestión de datos descentralizada (cont).

Teorema de CAP: es imposible para un **sistema distribuido de persistencia de datos** proveer simultáneamente:

- Consistencia: Cada lectura recibe la escritura más reciente o un error.
- Disponibilidad: Cada request recibe una respuesta no errónea, sin garantizar que contenga la escritura más reciente.
- Tolerancia al particionamiento: el sistema sigue funcionando incluso si un número arbitrario de mensajes son descartados (o retrasados) entre nodos de la red.

MSA - Características

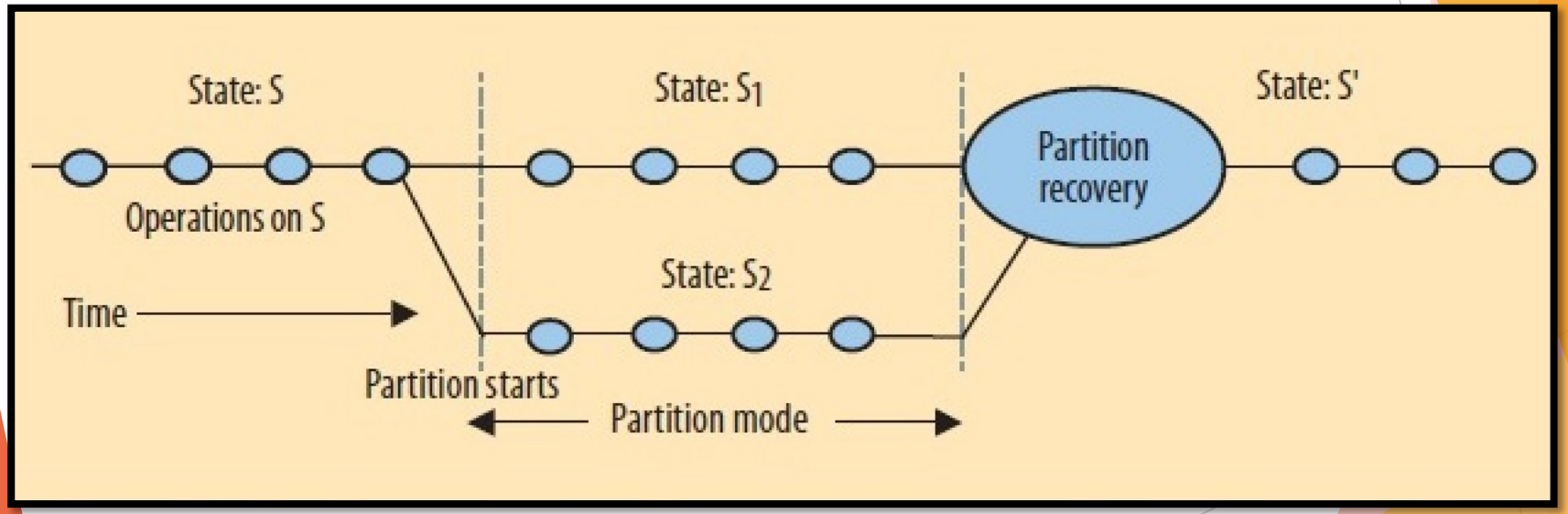
Gestión de datos descentralizada (cont).

Manejo de actualizaciones.

- Enfoque tradicional de transacciones.
  - Alto nivel de consistencia.
  - Impone un fuerte acoplamiento temporal.
- Transacciones distribuidas.
  - Alto costo de implementación.
- Coordinación de servicios sin transacciones.
  - Reconocimiento explícito de que la consistencia puede ser sólo **consistencia eventual.**
  - Los problemas se resuelven con operaciones de compensación.

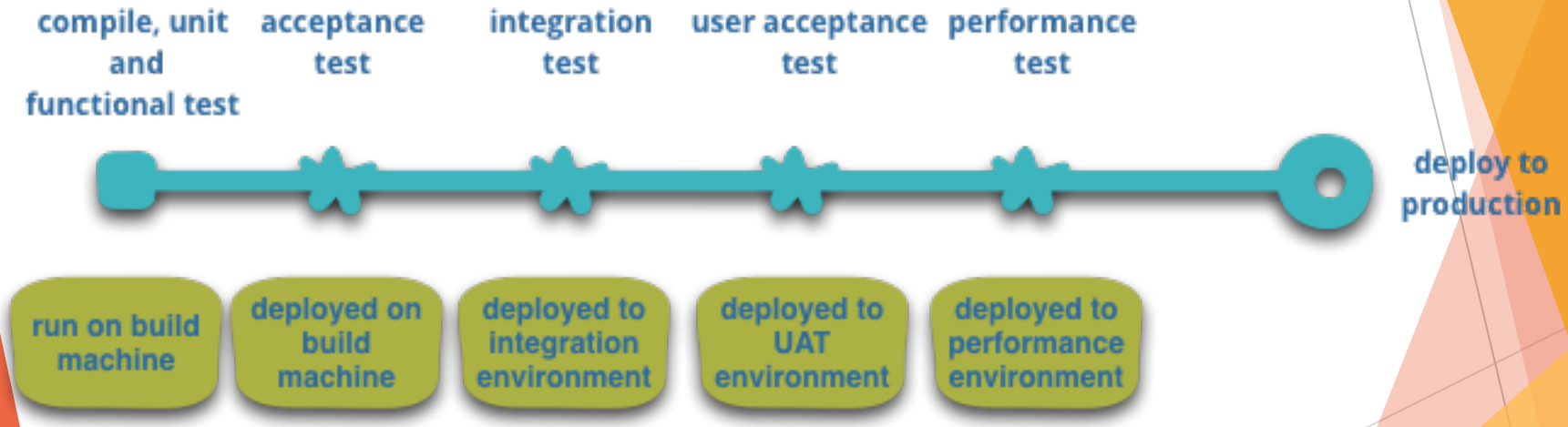
MSA - Características

Gestión de datos descentralizada (cont).



MSA - Características

Automatización de infraestructura



MSA - Características

Automatización de infraestructura (cont.)

- Pipeline de entrega contínua (Continuous Delivery)
  - Integración contínua del software desarrollado, construyendo ejecutables y ejecutando pruebas automatizadas sobre éstos.
  - Despliegue de componentes en entornos cada vez más similares a producción.
- Automatización en la gestión de microservicios en producción.
  - Orquestación de contenedores.

MSA - Características

Diseño tolerante a fallos

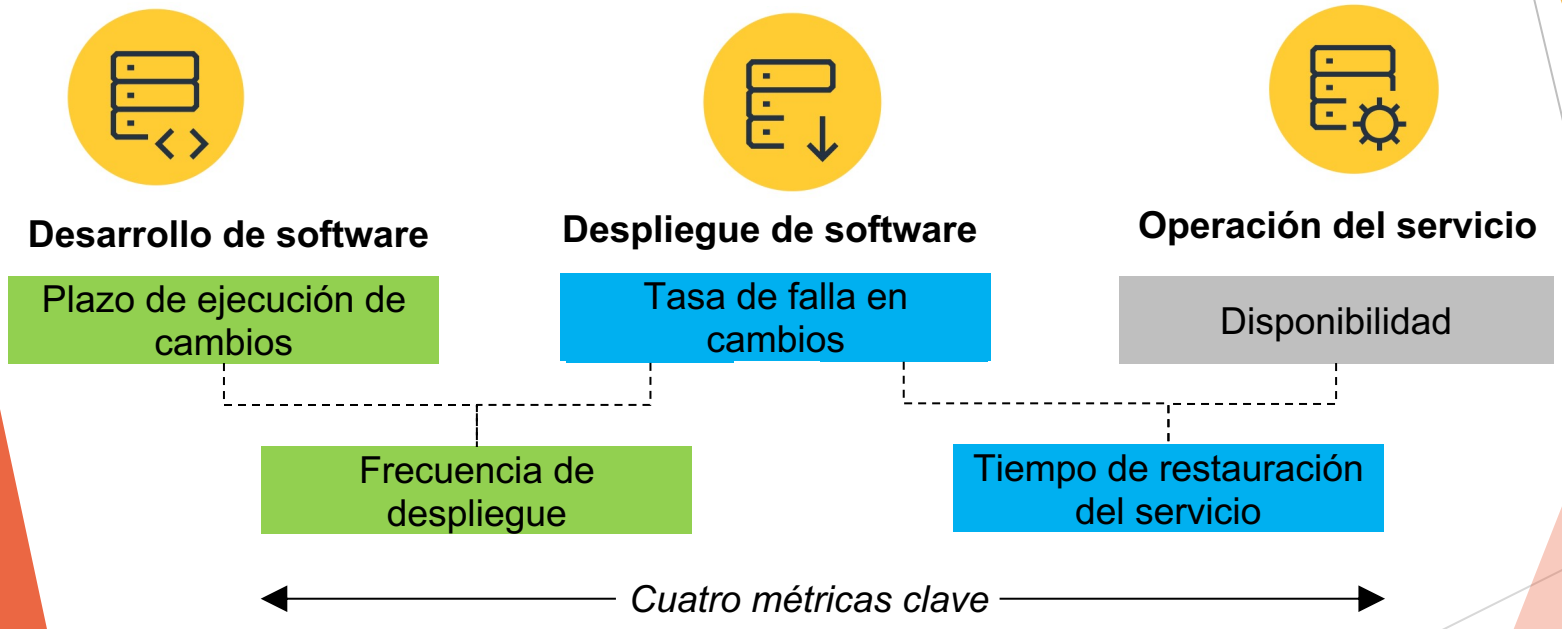
- Las comunicaciones sobre redes son inherentemente poco confiables.
- Las aplicaciones se diseñan para ser resilientes y manejar errores, no sólo para prevenirlos.
  - Se aplican patrones de diseño específicos.
- Fuerte énfasis en el monitoreo en tiempo real de la aplicación.
  - Monitoreo de elementos de arquitectura.
  - Monitoreo de métricas relevantes del negocio.

MSA - Características

Diseño evolutivo

- El diseño modular busca mantener las cosas que cambian al mismo tiempo dentro de un mismo módulo.
  - La propiedad clave de un componente es la noción de su independencia de cambio y actualización.
- Los servicios evolucionan buscando reducir al mínimo el impacto de los cambios en sus consumidores.
- Los servicios se diseñan buscando el mínimo acoplamiento posible al contrato de sus proveedores.
- La utilización de servicios como componentes posibilita planeamientos de despliegues más granulares.

Métricas


Desarrollo de software
Despliegue de software
Operación del servicio
Plazo de ejecución de cambios
Tasa de falla en cambios
Disponibilidad
Frecuencia de despliegue
Tiempo de restauración del servicio
Cuatro métricas clave

Métricas

Aspecto de la performance de la entrega de software	Élite	Alto	Medio	Bajo
Frecuencia de despliegue	Bajo demanda (varios despliegues por día)	Entre una vez por día y una vez por semana	Entre una vez por semana y una vez por mes	Entre una vez por mes y una vez por semestre
Plazo de ejecución de cambios	Menos de un día	Entre un día y una semana	Entre una semana y un mes	Entre uno y seis meses
Tiempo de restauración de servicio	Menos de una hora	Menos de un día	Menos de un día	Entre una semana y un mes
Tasa de falla en cambios	0-15%	0-15%	0-15%	46-60%

MSA - Ventajas

- Facilita el *continuous delivery* de aplicaciones grandes y complejas.
  - Mejora la mantenibilidad.
    - Descomposición modular de la complejidad.
    - Ciclos de evolución de componentes más desacoplados.
    - Límites y contratos de servicios más explícitos.
  - Facilita la incorporación de nuevas tecnologías.
  - Permite el despliegue y escalado independiente de servicios.
  - Permite organizar los esfuerzos de desarrollo en torno a múltiples equipos autónomos.

MSA - Ventajas

- Los servicios son relativamente pequeños:
  - Son más fáciles de entender para un desarrollador.
  - La aplicación inicia en menor tiempo.
    - Aumento de productividad de desarrollo.
    - Mejora los tiempos de implementación.
- Mejora el aislamiento de fallos.
- Elimina compromisos a largo plazo con un stack tecnológico.

MSA - Inconvenientes

- Aumento significativo de la complejidad.
  - Complejidad adicional propia de un sistema distribuido.
    - Requiere la implementación de mecanismos de comunicación entre servicios y el manejo de fallos parciales.
    - Se dificulta el testing de interacción entre servicios.
    - Aumenta la complejidad de implementación, gestión y monitoreo.
    - Se dificulta la detección de errores en tiempo de ejecución.
    - Las herramientas/IDEs están orientados al desarrollo de aplicaciones monolíticas.

MSA - Inconvenientes

- Complejidad de la arquitectura de persistencia de datos particionada.
  - Son muy comunes las transacciones de negocio que requieren actualizaciones en repositorios pertenecientes a múltiples servicios.
    - Las transacciones distribuidas no siempre son una opción.
    - El enfoque de persistencia eventual es más complejo que el enfoque tradicional transaccional.
- Las soluciones que abarcan múltiples servicios son más complejas y requieren una ajustada coordinación entre equipos.
- Aumenta el consumo de memoria de la aplicación.

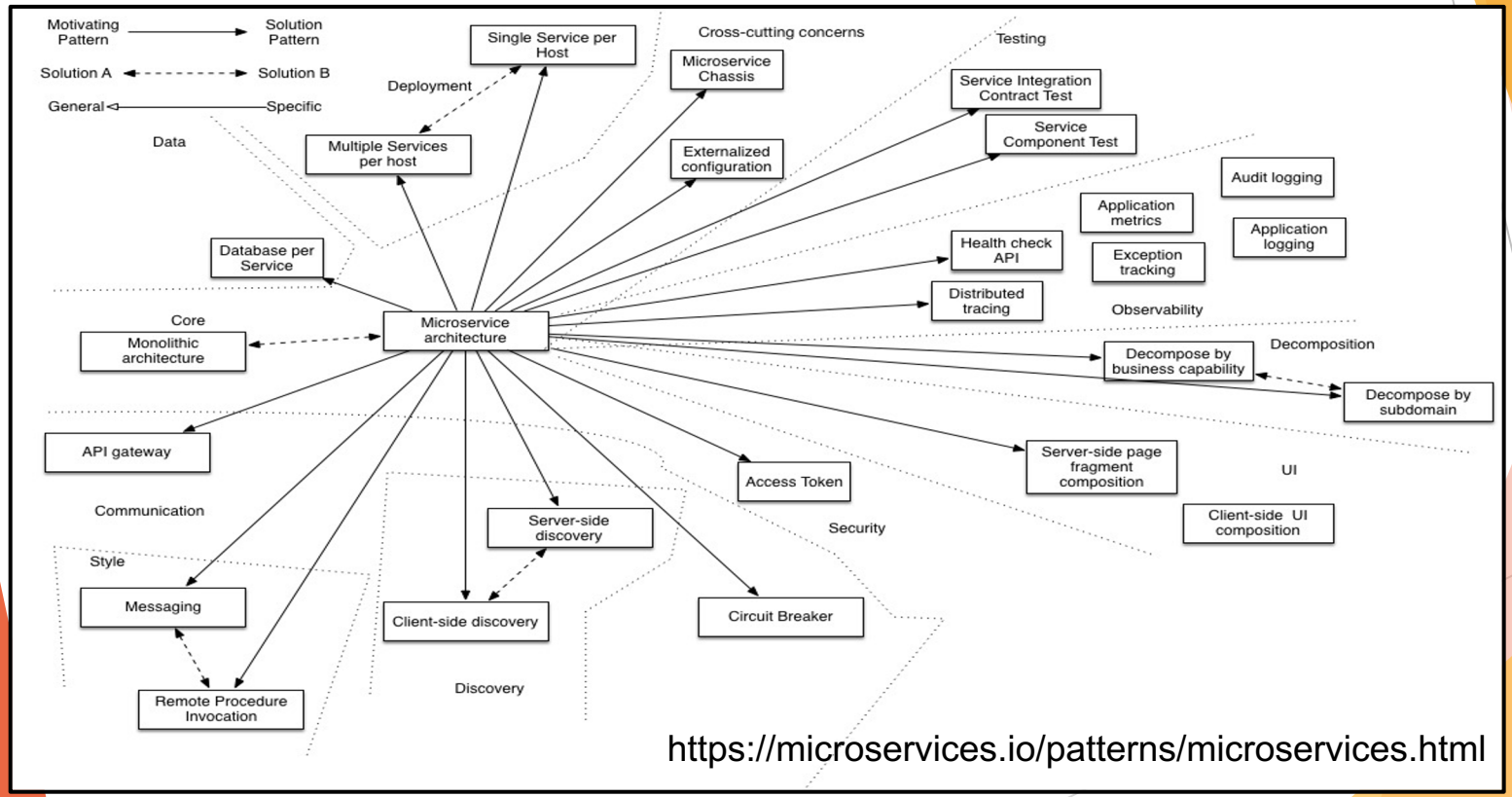
¿Cuando utilizar una MSA?

- Como toda arquitectura, tiene su compensación.
- **En diversas situaciones una arquitectura monolítica es la mejor opción.**
- Generalmente, las primeras versiones de una aplicación no tienen los problemas que resuelve este enfoque.
- Si se requiere de una aplicación en el menor tiempo posible se debe considerar que el uso de una arquitectura elaborada y distribuida ralentizará el desarrollo.
- A medida que una aplicación va creciendo y surgen los desafíos de escalado y segmentación funcional, se complejiza la descomposición de un monolito en un conjunto de servicios.

¿Cuando utilizar una MSA?

- Las MSA tienen serias consecuencias en la operación de la aplicación por lo que existen un conjunto de capacidades a considerar al evaluar su factibilidad de implementación:
  - Rápido aprovisionamiento.
  - Monitoreo básico.
    - Técnico (latencia, disponibilidad del servicio, etc).
    - De negocio (cantidades de pedidos)
  - Rápida implementación.
    - Generalmente, mediante un pipeline.

Patrones relacionados



https://microservices.io/patterns/microservices.html

Referencias

- https://martinfowler.com/articles/microservices.html
- https://microservices.io/patterns/microservices.html
- https://www.nginx.com/blog/introduction-to-microservices/
- https://developers.redhat.com/promotions/microservices-for-java-developers/
- https://martinfowler.com/bliki/ContinuousDelivery.html
- https://martinfowler.com/articles/microservice-trade-offs.html
- https://martinfowler.com/bliki/MicroservicePrerequisites.html
- https://www.infoq.com/articles/cap-twelve-years-later-how-the-rules-have-changed/
- https://services.google.com/fh/files/misc/state-of-devops-2019.pdf