DIGITALHOUSE

La importancia de un orquestador de contenedores

Certified Tech Developer
The Ultimate Degree

Índice

01 Introducción

Introducción

Los contenedores han cambiado la forma en la que los programadores conciben el desarrollo, el despliegue y el mantenimiento de las aplicaciones. Los contenedores son tan ligeros y flexibles que han dado lugar a nuevas y complejas arquitecturas de aplicaciones.

Este nuevo enfoque consiste en empaquetar los servicios que conforman una aplicación en contenedores separados, y luego desplegarlos a través de varias máquinas físicas o virtuales. Todo esto da pie a la necesidad de la orquestación de contenedores, es decir, disponer de alguna herramienta o sistema que gestione el despliegue, el escalado, la interconexión y la disponibilidad de nuestras aplicaciones basadas en contenedores.

02

¿Qué problemas resuelven los orquestadores?

¿Qué es una orquestación de contenedores?

Los contenedores se pueden hacer altamente escalables, que se pueden crear a pedido. Si bien esto es bueno para algunos contenedores, imaginemos que tenemos cientos de ellos. Se volverá extremadamente difícil administrar el ciclo de vida del contenedor y su funcionamiento cuando los números aumentan dinámicamente con la demanda.

La orquestación de contenedores resuelve este problema automatizando la programación, implementación, escalabilidad, equilibrio de carga, disponibilidad y redes de contenedores. **La orquestación de contenedores es la automatización y gestión del ciclo de vida de contenedores y servicios.** Afortunadamente, hay muchas herramientas de orquestación de contenedores disponibles en el mercado.

Es un proceso de gestión y organización de múltiples contenedores y arquitectura de microservicios a escala.

Características de los orquestadores

Una herramienta de orquestación de contenedores tiene como propósito **el manejo del ciclo de vida de los contenedores**. Principalmente, son utilizados para el manejo de diferentes tareas, entre las cuales podemos citar:

- Despliegue y ejecución automática de servicios basados en contenedores.
- Autoescalado y autoreinicio de contenedores.
- Intercambio de datos y networking.
- Monitorear la salud de los contenedores.
- Gestión de redundancia y de alta disponibilidad.
- Balanceo de carga y descubrimiento de servicios.
- Repartición de recursos y otras tareas.
- Mantenimiento de parámetros "secretos" y configuraciones.

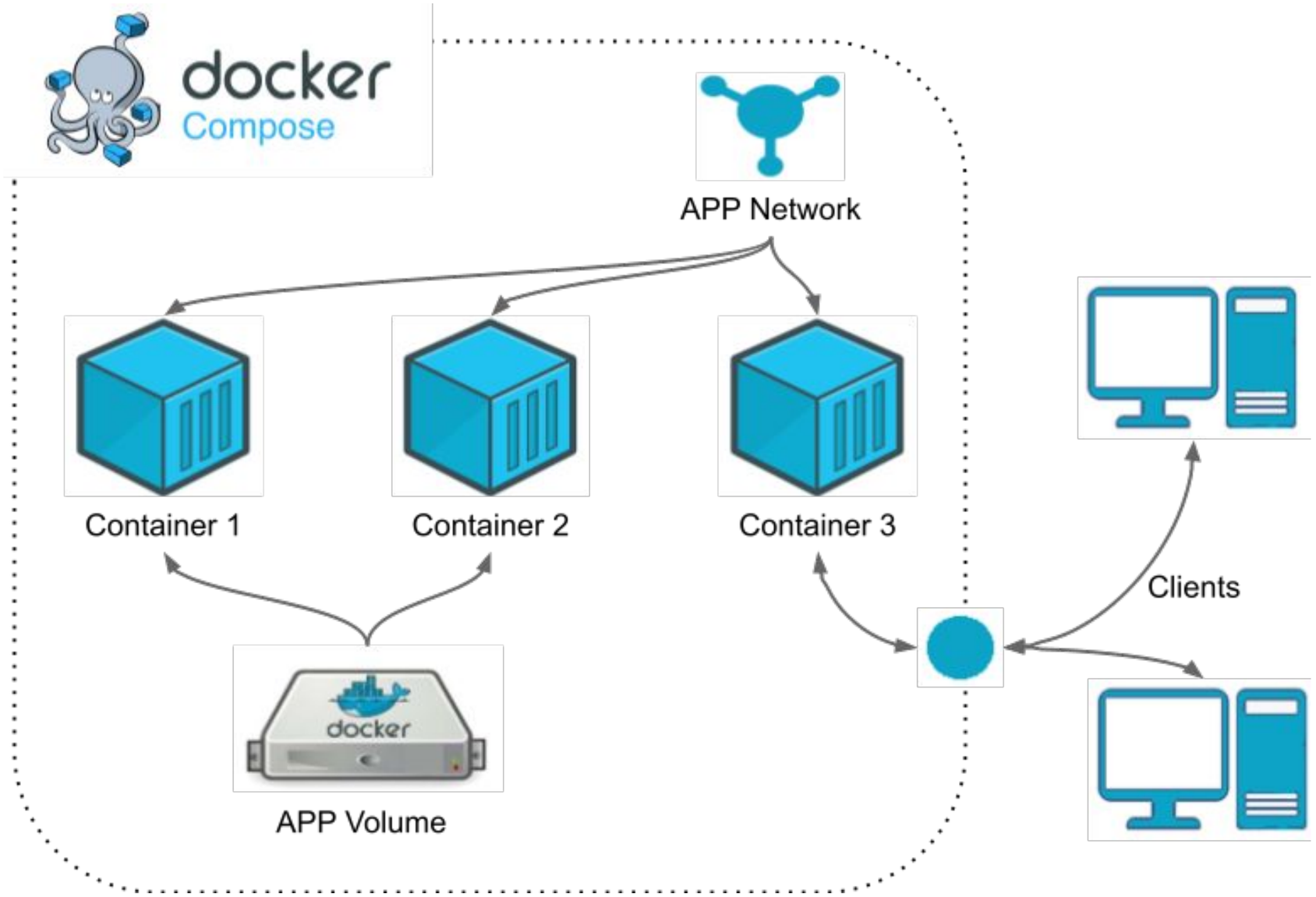
03 Herramientas

Docker Compose

En la clase anterior vimos a Docker Compose, una herramienta que **ayuda a establecer las reglas para aplicaciones de varios contenedores**.
Generalmente, se usa para describir grupos de servicios interconectados que comparten dependencias de software que están orquestados y deben ser escalados en conjunto.
Se utiliza un archivo YAML para configurar los contenedores que se van a utilizar para ejecutar los servicios de las diferentes aplicaciones.
Docker Compose se ha centrado tradicionalmente en el desarrollo, pero ahora se están enfocando en funciones más orientadas a la producción.

Veamos algunas de las ventajas que nos ofrece:

- Múltiples entornos aislados en un solo host.
- Conservación de los datos de los contenedores a través de las diferentes ejecuciones.
- Regeneración de contenedores que hayan sufrido modificaciones.
- Configuración de variables y composición de diferentes entornos de trabajo.
- Organización del trabajo para que varios contenedores funcionen juntos.

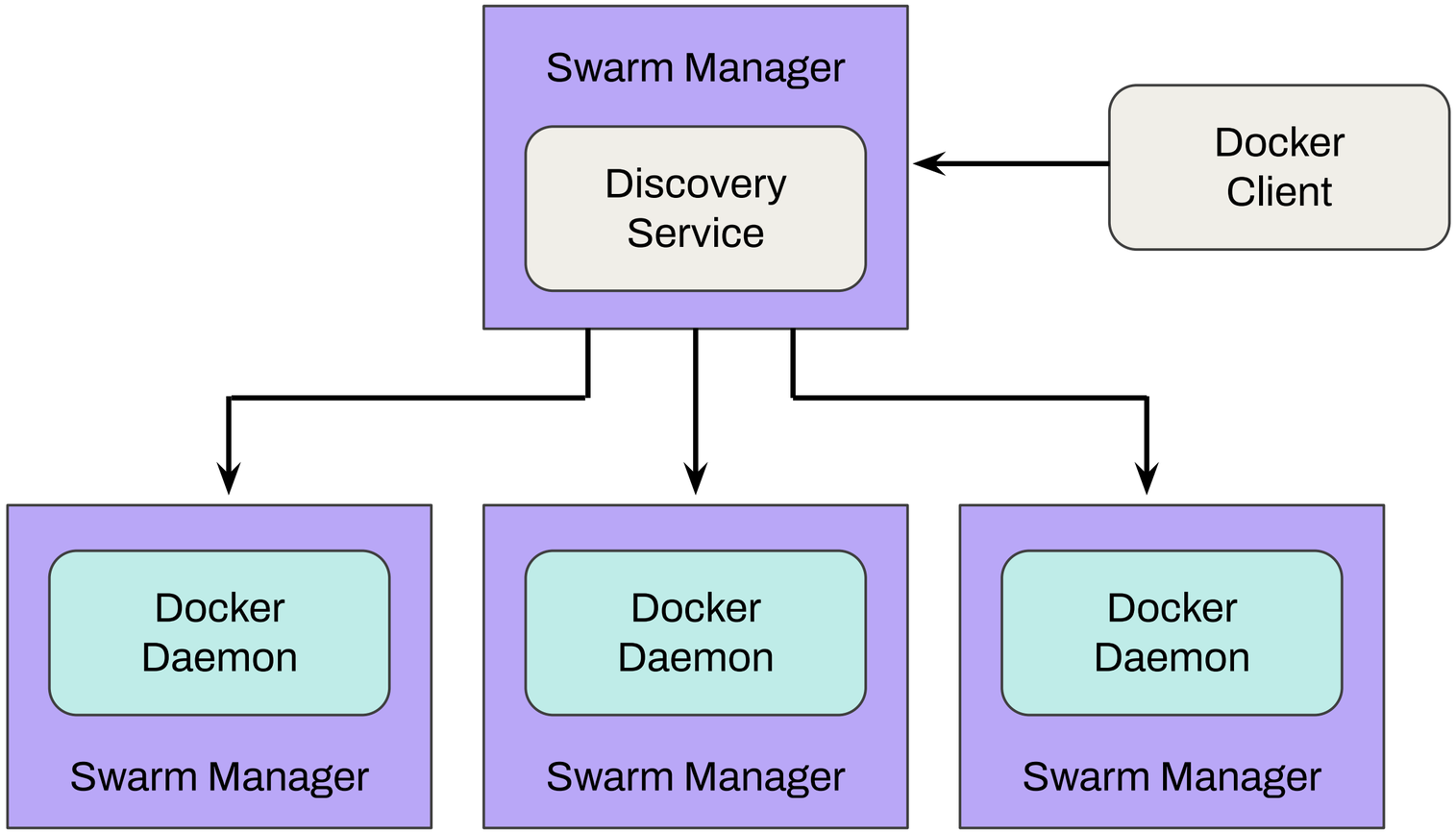

docker
Compose
APP Network
Container 1
Container 2
Container 3
Clients
docker
APP Volume

Docker Swarm

Swarm es otra **herramienta integrada en Docker** que permite agrupar una serie de hosts de Docker en un clúster y gestionarlos de forma centralizada, así como orquestar contenedores.
Es la solución que propone Docker ante los problemas de los desarrolladores a la hora de orquestar y planificar contenedores a través de muchos servidores.
Veamos las ventajas que nos ofrece:

- Escalar declarando tareas que pueden correr para cada servicio a desplegar.
- Manejar la red de los servicios que se despliegan.
- Descubrimiento de servicios.
- Múltiples entornos aislados en un solo host.
- Conservar los datos de volumen cuando se crean contenedores.
- Solo vuelve a crear contenedores que hayan cambiado.
- Integración con la API de Docker Engine.
- Redistribuye las cargas de trabajo si hay algún fallo en los nodos, así se asegura alta disponibilidad.
- Posible escalado manual y rolling updates.
- No es necesario ninguna configuración adicional para utilizarlo con Docker.
- Variables y movimiento de una composición entre entornos.

Architecture Diagram

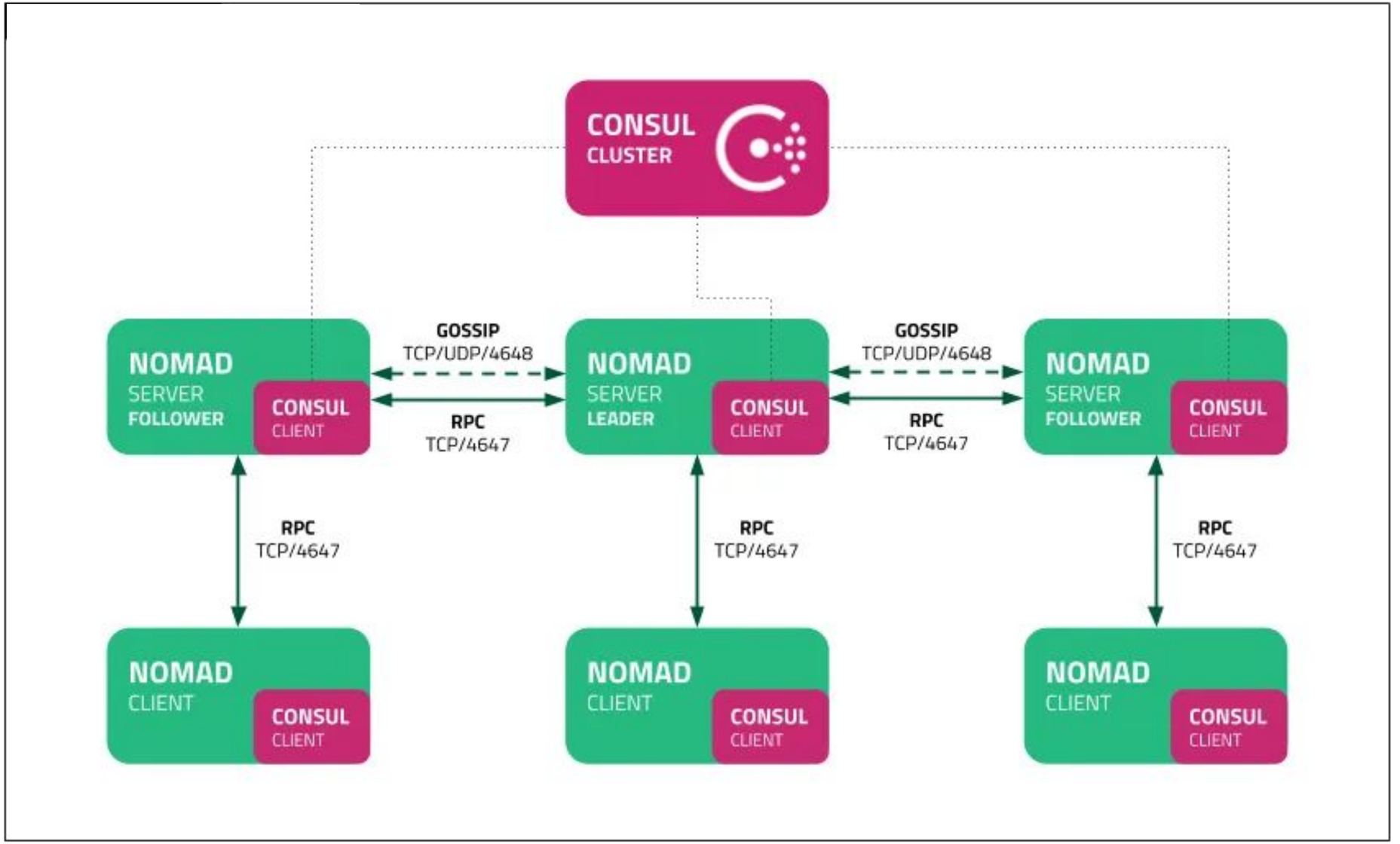


Nomad

Es una herramienta de orquestación de cargas simples de trabajo de código abierto diseñada para desplegar y gestionar contenedores y aplicaciones no contenerizadas a escala. Se ejecuta como un solo binario con muy poco uso de recursos y es compatible con macOS, Windows y Linux.
No es una solución completa de orquestación de contenedores. **Su enfoque es ser muy sencillo y eficaz con su cometido** y encajar con distintas piezas para cada funcionalidad extra que pueden ser utilizadas o no en función de las necesidades.

Características:

- Simple y confiable.
- Diseñado para ser escalado de forma eficiente.
- Cargas de trabajo flexibles.
- Multidatacenter y Multiregion.
- Ofrece balanceo de carga cuando se lo utiliza con Nginx o HAProxy.
- Se integra con Consul para ofrecer la funcionalidad de descubrimiento.
- No dispone de una gestión de secretos, pero para ello se integra perfectamente con Vault.

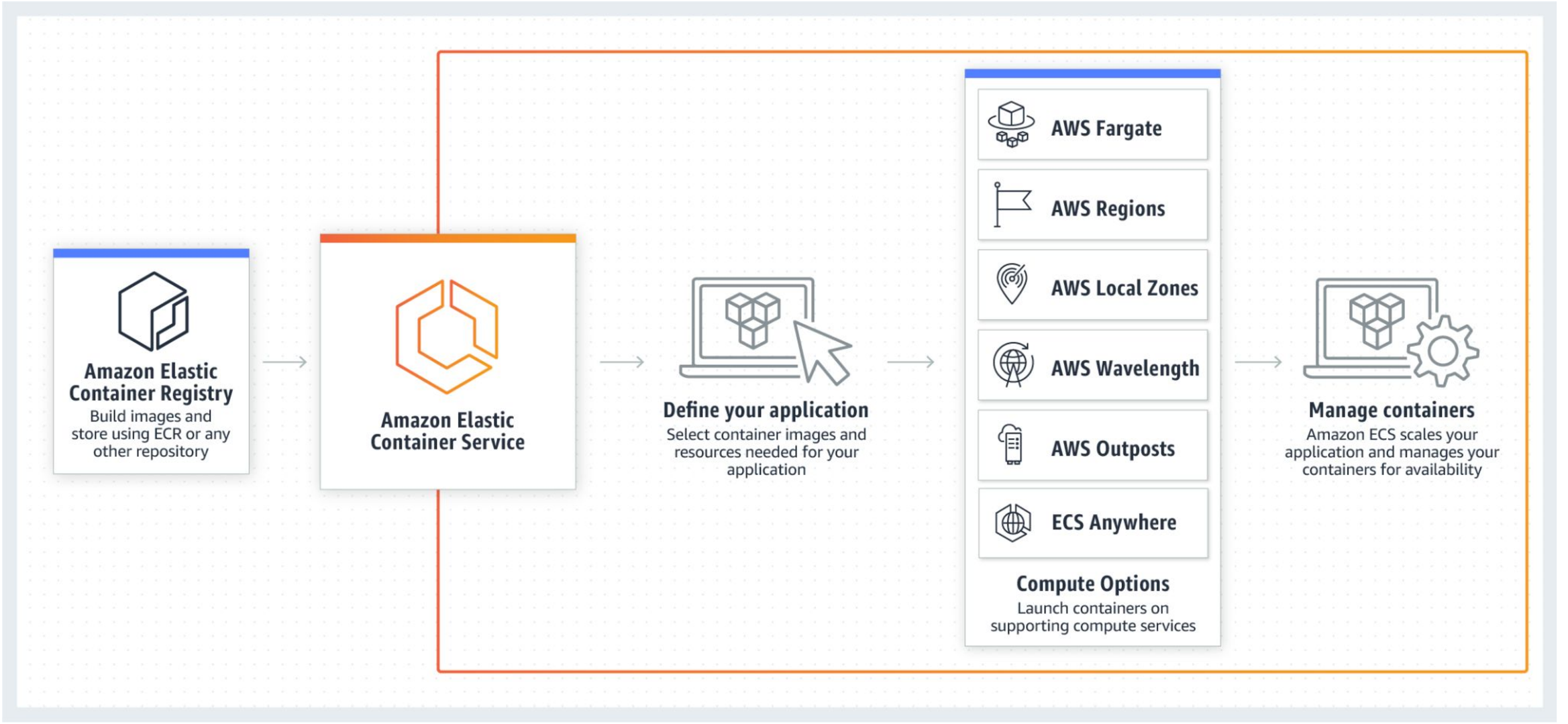

CONSUL
CLUSTER
NOMAD
SERVER
FOLLOWER
CONSUL
CLIENT
GOSSIP
TCP/UDP/4648
RPC
TCP/4647
NOMAD
SERVER
LEADER
CONSUL
CLIENT
GOSSIP
TCP/UDP/4648
RPC
TCP/4647
NOMAD
SERVER
FOLLOWER
CONSUL
CLIENT
RPC
TCP/4647
RPC
TCP/4647
RPC
TCP/4647
NOMAD
CLIENT
CONSUL
CLIENT
NOMAD
CLIENT
CONSUL
CLIENT
NOMAD
CLIENT
CONSUL
CLIENT

Amazon ECS

El servicio de AWS para orquestación de contenedores, Amazon ECS, es un sistema de gestión muy escalable que permite a los desarrolladores ejecutar aplicaciones en contenedores sobre instancias EC2.
La principal desventaja es que no existe soporte para ejecutar contenedores fuera de EC2, los aspectos a favor incluyen ventajas propias del servicio AWS, tales como CloudTrail, CloudWatch, Elastic Load Balancing, etc.

Características:

- Sin servidor de manera predeterminada con AWS Fargate.
- Seguridad y aislamiento por diseño.
- Compatibilidad con Docker.
- Compatibilidad con contenedores de Windows.
- Recuperación automática de los contenedores.
- Posibilidad de uso con otros servicios de AWS.
- Operaciones autónomas del plano de control.

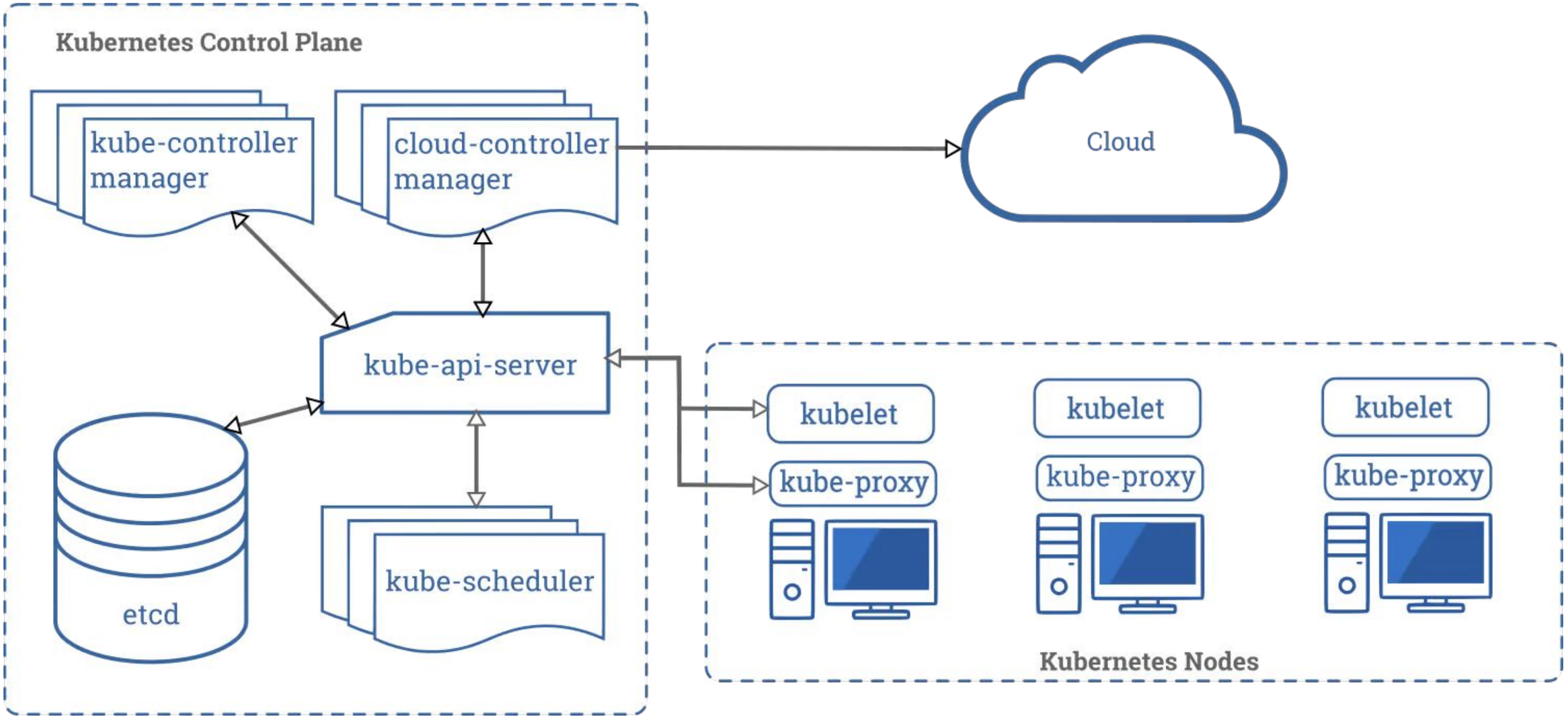

Amazon Elastic Container Registry
Build images and store using ECR or any other repository
Amazon Elastic Container Service
Define your application
Select container images and resources needed for your application
AWS Fargate
AWS Regions
AWS Local Zones
AWS Wavelength
AWS Outposts
ECS Anywhere
Compute Options
Launch containers on supporting compute services
Manage containers
Amazon ECS scales your application and manages your containers for availability

04 Kubernetes

Kubernetes

Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios sobre la que trabajaremos en **profundidad en las siguientes clases**. Ayuda a automatizar la implementación, el escalado y la administración de cargas de trabajo y servicios en contenedores. **Es la plataforma más utilizada para la orquestación de contenedores**.
Veamos algunas las funciones que nos ofrece:

- Orquestar contenedores en múltiples hosts.
- Hacer un mejor uso del hardware para maximizar los recursos necesarios para ejecutar sus aplicaciones empresariales.
- Controlar y automatizar las implementaciones y actualizaciones de las aplicaciones.
- Montar y añadir almacenamiento para ejecutar aplicaciones con estado.
- Escalar las aplicaciones en contenedores y sus recursos sobre la marcha.
- Servicio de descubrimiento y balanceo de carga.
- Orquestación de almacenamiento.
- Gestión de secretos y configuraciones.
- Administrar servicios de forma declarativa, que garanticen que las aplicaciones implementadas siempre se ejecuten del modo que se implementaron.


Kubernetes Control Plane
kube-controller manager
cloud-controller manager
Cloud
kube-api-server
kubelet
kubelet
kubelet
kube-proxy
kube-proxy
kube-proxy
etcd
kube-scheduler
Kubernetes Nodes

¿Qué no es Kubernetes?

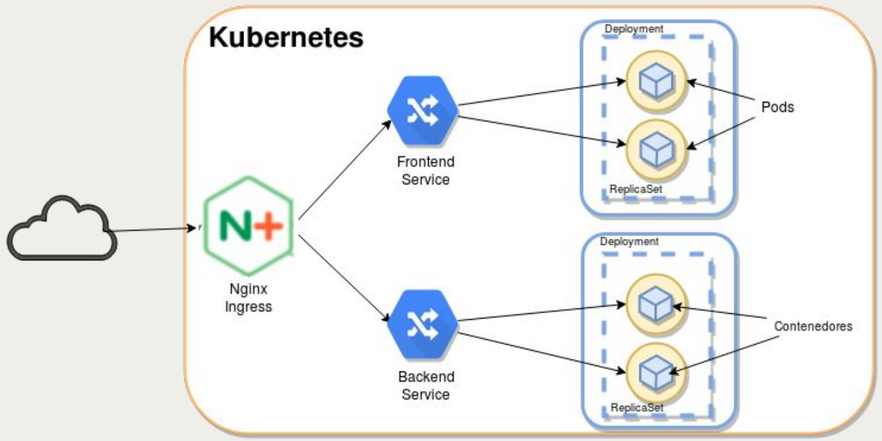
Para tener en claro cuál es el alcance de Kubernetes podemos decir para qué no está diseñado. De esta manera podremos hacer un uso correcto de la plataforma:

- **No es un compilador**.
- No es un proveedor de servicios, como buses de mensaje, BD or caché.
- No es una herramienta de monitoreo de contenedores.
- No es un framework que obliga a utilizar un determinado lenguaje o BD.

Ventajas

Ágil creación y despliegue de aplicaciones

- Es más eficaz para crear imágenes de contenedor en vez de máquinas virtuales.



Desarrollo, integración y despliegue continuo

- Facilita los **rollbacks**, ya que las imágenes construidas son inmutables.

Dev y Ops

- Facilita la separación de tareas entre Dev y Ops. Crea imágenes del contenedor al momento de compilar y no al momento de desplegar, **desacoplando la aplicación de la infraestructura.**

Observabilidad

- Obtiene información y métricas del sistema operativo y también de la salud de la aplicación.

Consistencia

- La aplicación funciona de la misma manera en nuestra computadora con el sistema operativo que contenga, o en la nube en la que decidamos realizar el despliegue.

Y más...

Portabilidad

Despliega la aplicación en Ubuntu, RHEL, CoreOS, un datacenter físico, Google Kubernetes Engine, EKS y todos los demás.

Microservicios

Microservicios distribuidos, elásticos, liberados y **débilmente acoplados**. Aplicaciones separadas en piezas pequeñas independientes que pueden ser desplegadas y administradas de forma dinámica.

Recursos

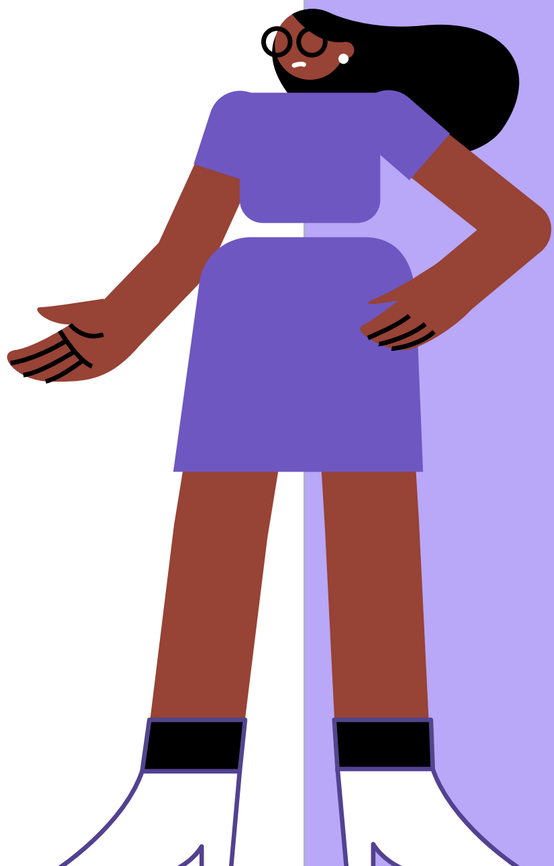
Predice el rendimiento de la aplicación y permite mayor eficacia de la utilización de recursos.

05

Microservicios, contenedores y orquestadores

Arquitectura de microservicios

Siguiendo la filosofía de “divide y vencerás”, muchos desarrollos de software serán más sencillos de implementar si los descomponemos y dividimos en diferentes partes que, finalmente, formen un todo compacto y robusto. Empleando un enfoque de **arquitectura de microservicios** para el desarrollo de aplicaciones de software, trabajamos en el desarrollo de pequeños componentes modulares que actúan de forma independiente y se actualizan por separado, pero que a su vez forman parte de un todo, se comunican entre sí y forman un engranaje de procesos y servicios integrados.



Relación entre orquestadores y los microservicios

Visto el rompecabezas de componentes independientes de la arquitectura de microservicios, nos encontramos con la orquestación. Los diferentes elementos que forman una aplicación software desarrollada en microservicios deben orquestarse, o lo que es lo mismo, relacionarse entre sí para actuar como un todo robusto que realice la funciones para las que ha sido creado.

La tecnología de contenedores responde a estas capacidades de orquestación. De hecho, los contenedores están diseñados para ejecutar cualquier funcionalidad sobre la base de pequeñas piezas. Con la “contenerización” unida a los microservicios, se hace más fácil administrar y actualizar cada uno de los servicios que se integran en una aplicación software.

Relación entre orquestadores y los microservicios

Docker ha sido la tecnología de contenedores más empleada durante los últimos tiempos, ya que permitía aislar los contenedores por procesos y servicios de forma efectiva. Sin embargo, cuando es necesario administrar escenarios más complejos que coordinan múltiples servicios y procesos, Docker no resulta tan efectivo. Por este motivo, durante los últimos años, han surgido nuevas tecnologías de código abierto, como **Kubernetes**, que son más eficientes en escenarios más complejos.

En definitiva, podemos determinar que la arquitectura de microservicios ha llegado para quedarse, especialmente ante desarrollos de aplicaciones de software complejos. La tecnología que respalda este tipo de arquitectura sería la de los contenedores que, como hemos visto, da un paso más allá, gracias a nuevas tecnologías como Kubernetes que superan las funcionalidades de Docker.

DIGITALHOUSE

¡Muchas gracias!