



Arquitectura y Sistemas Operativos



Unidad 3: Virtualización y contenedores



¿Qué es virtualización? ¿Qué sucede cuando virtualizamos?

¿Qué son contenedores? ¿Docker?



Virtualización



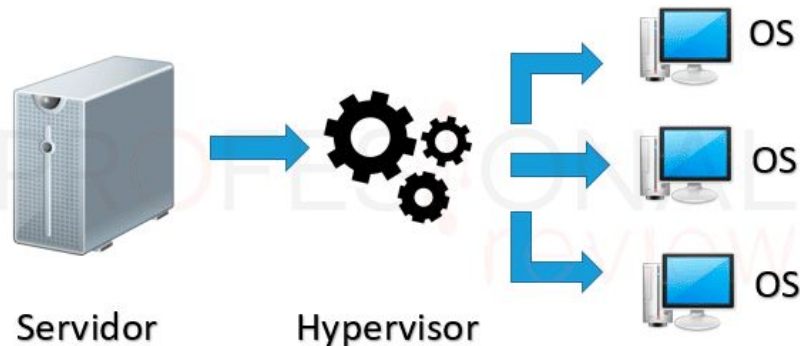
- La virtualización crea un entorno informático simulado, o virtual, en lugar de un entorno físico.
- A menudo, incluye versiones de hardware, sistemas operativos, dispositivos de almacenamiento, etc., generadas por un equipo.
- Permite a las organizaciones particionar un equipo o servidor físico en varias [máquinas virtuales](#).
- Cada máquina virtual puede interactuar de forma independiente y ejecutar sistemas operativos o aplicaciones diferentes mientras comparten los recursos de una sola máquina host.






Virtualización - ¿Por qué?

- Al crear varios recursos a partir de un único equipo o servidor, la virtualización mejora la **escalabilidad** y las **cargas de trabajo**, al tiempo que permite usar menos servidores y reducir el consumo de energía, los costos de infraestructura y el mantenimiento.



Virtualización - Categorías

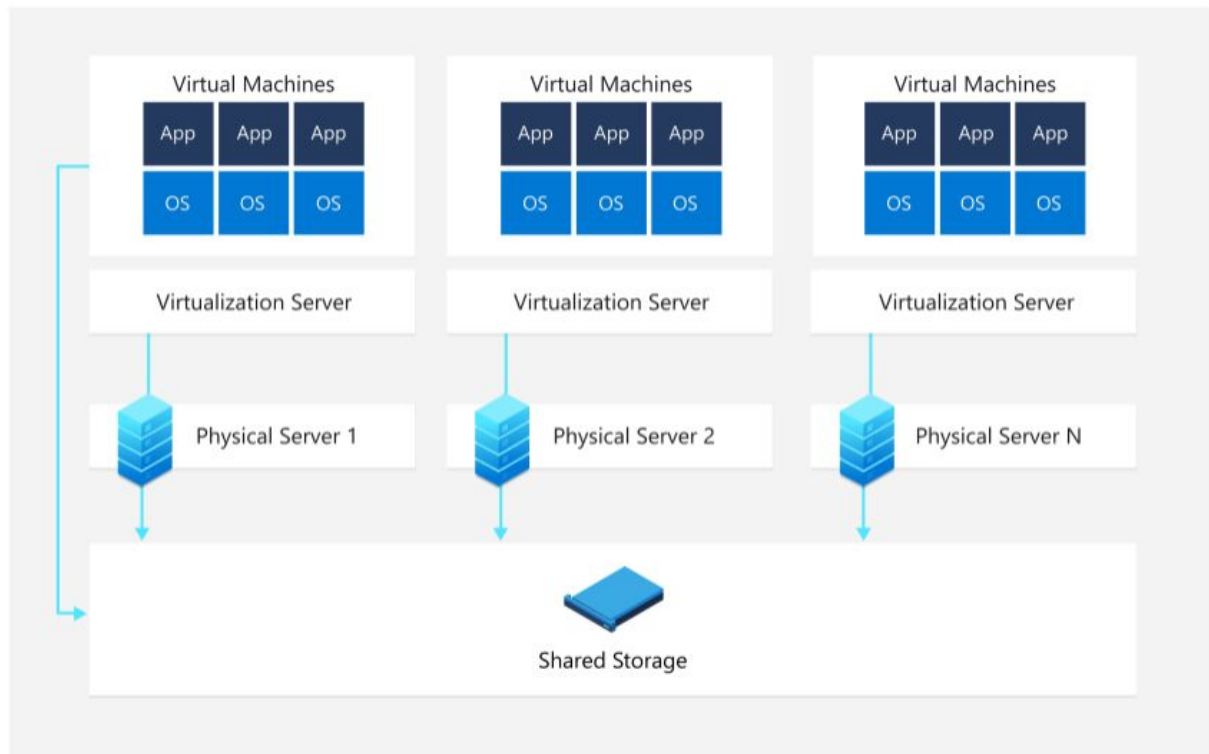
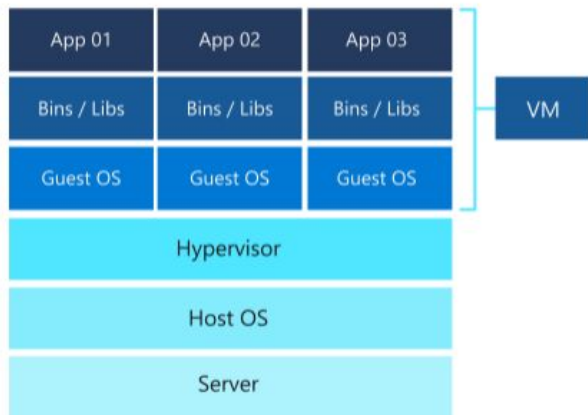
- 
- La primera es la **virtualización de escritorio**, que permite que un servidor centralizado ofrezca y administre escritorios individualizados.
 - La segunda es la **virtualización de red**, diseñada para dividir el ancho de banda de una red en canales independientes que se asignan a servidores o dispositivos específicos.
 - La tercera categoría es la **virtualización de software**, que separa las aplicaciones del hardware y el sistema operativo.
 - Y la cuarta es la **virtualización de almacenamiento**, que combina varios recursos de almacenamiento en red en un solo dispositivo de almacenamiento accesible por varios usuarios.

¿Qué es una máquina virtual?



- Una máquina virtual (término que a menudo se abrevia como VM) no es diferente a cualquier otro equipo físico, como un portátil, un smartphone o un servidor.
- Tiene una CPU, memoria, discos para almacenar los archivos y puede conectarse a Internet si es necesario.
- Mientras los componentes de su PC (denominados hardware) son físicos y tangibles, las máquinas virtuales suelen considerarse equipos virtuales o equipos definidos por software dentro de servidores físicos, donde solo existen como código

¿Qué es una máquina virtual?





¿Cómo funciona una máquina virtual?

- La virtualización es el proceso de crear una versión basada en software o "virtual" de un equipo, con cantidades dedicadas de CPU, memoria y almacenamiento que se "toman prestadas" de un equipo host físico, como su PC, o un servidor remoto, como un servidor en el centro de datos de un proveedor de nube.
- Una máquina virtual es un archivo de PC, que suele denominarse "imagen", que se comporta igual que un equipo real.
- Se puede ejecutar en una ventana como un entorno informático aparte, a menudo para ejecutar un sistema operativo diferente o, incluso, para la experiencia informática completa del usuario, como es habitual en los equipos de trabajo de muchas personas.
- La máquina virtual está en una partición separada del resto del sistema, lo que significa que el software que se encuentra dentro de una máquina virtual no puede interferir con el sistema operativo principal del equipo host.



¿Para qué se usan las máquinas virtuales?

- Compilar e implementar aplicaciones en la nube.
- Probar un nuevo sistema operativo (SO), incluidas las versiones beta.
- Poner en marcha un nuevo entorno para que les resulte más sencillo y rápido a los desarrolladores ejecutar escenarios de desarrollo y pruebas.
- Crear copias de seguridad del sistema operativo.
- Acceder a datos infectados por virus o ejecutar una versión anterior de una aplicación con la instalación de un sistema operativo anterior.
- Ejecutar software o aplicaciones en sistemas operativos para los que no se habían diseñado inicialmente.



Beneficios de Máquinas Virtuales (VMs)

- Al ejecutarse como equipos individuales con aplicaciones y sistemas operativos individuales, las máquinas virtuales tienen la ventaja de que permanecen completamente independientes entre sí y respecto al equipo host físico.
- Un software denominado hipervisor, o administrador de máquinas virtuales, permite ejecutar diferentes sistemas operativos en varias máquinas virtuales al mismo tiempo. Esto permite ejecutar máquinas virtuales Linux, por ejemplo, en un sistema operativo Windows o ejecutar una versión anterior de Windows en el sistema operativo Windows más actual.
- Además, dado que las máquinas virtuales son independientes entre sí, también son extremadamente portables. Puede mover una máquina virtual de un hipervisor a otro que esté en una máquina completamente diferente casi de forma inmediata.
- Veamos las ventajas que brindan su flexibilidad y portabilidad ->



Ventajas de Máquinas Virtuales (VMs)

- **Ahorro de costos:** la ejecución de varios entornos virtuales en una única infraestructura significa que puede reducir drásticamente la superficie física de la infraestructura. Esto aumenta los beneficios, ya que reduce la necesidad de mantener tantos servidores y los costos de mantenimiento y electricidad.
- **Agilidad y velocidad:** la puesta en marcha de una máquina virtual es relativamente fácil y rápida, y es mucho más sencilla para sus desarrolladores que el aprovisionamiento de un entorno nuevo completo. La virtualización hace que el proceso de ejecución de escenarios de desarrollo y pruebas sea mucho más rápido.
- **Tiempo de inactividad reducido:** las máquinas virtuales son muy portables y fáciles de migrar de un hipervisor a otro en un equipo diferente, por lo que son una solución excelente para copias de seguridad, en el caso de que el host deje de funcionar de forma inesperada.
- **Escalabilidad:** las máquinas virtuales permiten escalar más fácilmente las aplicaciones agregando más servidores virtuales o físicos para distribuir la carga de trabajo entre varias máquinas virtuales. Como resultado, puede aumentar la disponibilidad y el rendimiento de las aplicaciones.
- **Ventajas de seguridad:** dado que las máquinas virtuales se ejecutan en varios sistemas operativos, el uso de un sistema operativo invitado en una máquina virtual permite ejecutar aplicaciones de una seguridad dudosa y proteger el sistema operativo host. Las máquinas virtuales también permiten un mejor análisis forense de la seguridad y suelen usarse para estudiar virus informáticos de forma segura, aislándolos para evitar riesgos en el equipo host.



Desventajas de Máquinas Virtuales (VMs)

- **Performance:** Cuando varias máquinas virtuales se ejecutan simultáneamente en una computadora host, cada máquina virtual puede presentar un rendimiento inestable, que depende de la carga de trabajo en el sistema por parte de otras máquinas virtuales en ejecución
- **La máquina virtual no es tan eficiente como una verdadera cuando se accede al hardware.**
- **Problemas colaterales:**
 - Hay muchos casos en que el software de virtualización tiene dificultades para proporcionar un envío completo y confiable de dispositivos USB y seriales al entorno virtual (por ejemplo, VMWare, ESX, Microsoft Hyper-V). O se puede acceder al dispositivo desde la máquina virtual, pero es posible que no funcione correctamente.
- Otros inconvenientes en su implementación (próxima slide ->)

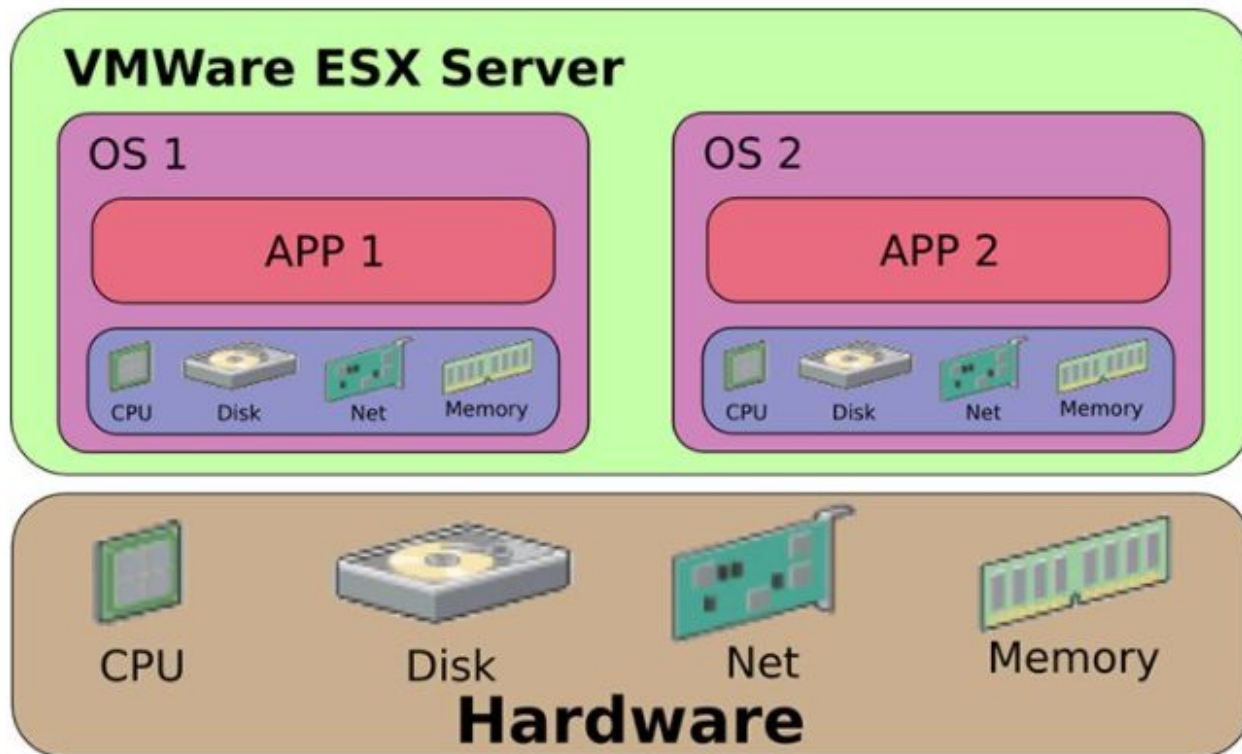


Problemas de Virtualización de Servidores

- **Daño colateral en su Facilidad y velocidad de implementación de servidores virtuales:** En entornos virtuales, todo se puede hacer o cambiar extremadamente más rápido. Existe la necesidad de mitigar el riesgo de errores y de la actividad maliciosa con la misma rapidez. Si las organizaciones carecen de buenas prácticas de planificación, el “factor velocidad” puede exacerbar las debilidades en los procesos de su empresa.
- **Seguridad:** A diferencia de un servidor físico, una máquina virtual es un conjunto de ficheros que residen físicamente en un almacenamiento compartido. Este tipo de encapsulación de la máquina virtual ofrece un nuevo tipo de “robo de datos”. Un servidor entero puede ser copiado a un dispositivo USB o copiado durante un proceso de backup no autorizado a un lugar no protegido por su personal de seguridad o administrador de su entorno virtual.



Estructura simplificada de Virtualización





Software que ofrecen virtualización



- **VirtualBox (licencia GNU - software libre)**
- **Citrix XenServer**
- **Sandboxie**
- **VMware Workstation Pro**
- **Cameyo**
- **Parallels (para Mac)**
- **Xen Hypervisor (similar a Citrix)**
- **QEMU (gratuito)**
- **Microsoft Hyper-V Server**
- **KVM**
- **Aviat Design (para virtualización de redes)**



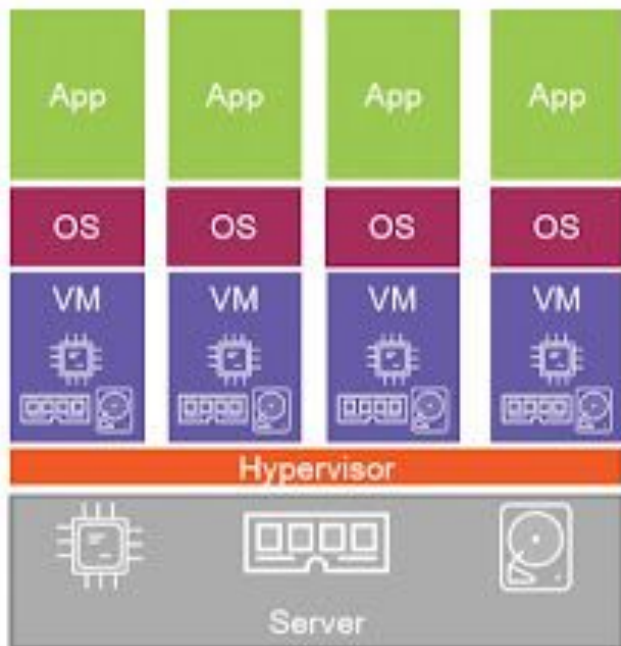
Contenerización de aplicaciones



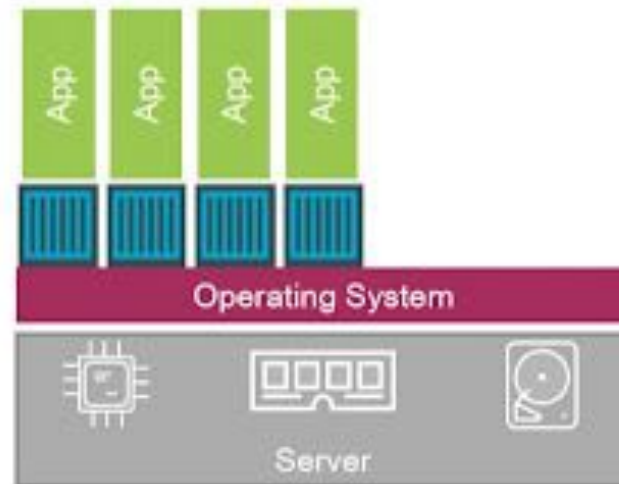
- La contenerización de aplicaciones es un método de virtualización de nivel de sistema operativo (nivel OS) para implementar y ejecutar aplicaciones distribuidas sin lanzar una máquina virtual completa (VM) para cada aplicación
- En su lugar, varios sistemas aislados se ejecutan en un único host de control y acceden a un único kernel.
- Los contenedores de aplicaciones contienen los componentes, como archivos, variables de entorno y bibliotecas, necesarios para ejecutar el software deseado.
- Debido a que los recursos se comparten de esta manera, se pueden crear contenedores de aplicaciones que ponen menos presión a los recursos globales disponibles.
- Por ejemplo, si se desea una variación de la imagen estándar, se puede crear un contenedor que contenga solo la nueva biblioteca.



Contenerización de aplicaciones



Hypervisor
Architecture



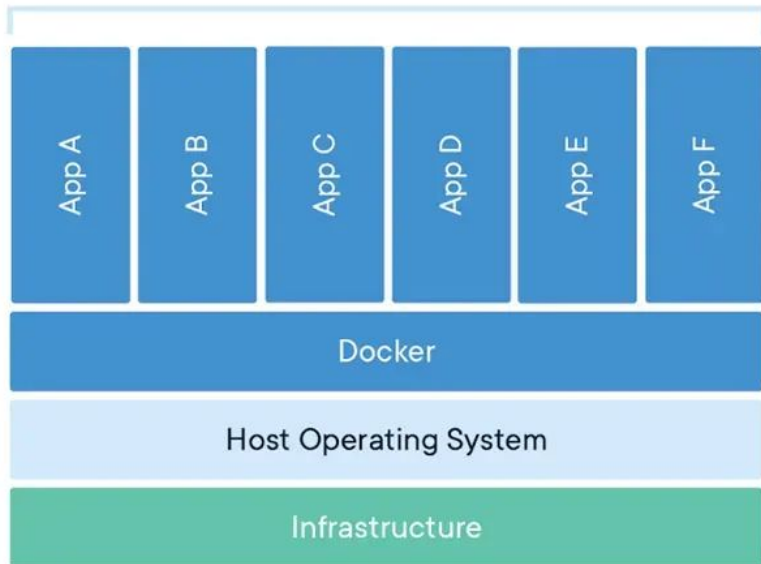
Container
Architecture



Contenerización de aplicaciones

 Publish Image

Containerized Applications



Virtual Machine

App A

Guest
Operating
System

Virtual Machine

App B

Guest
Operating
System

Virtual Machine

App C

Guest
Operating
System

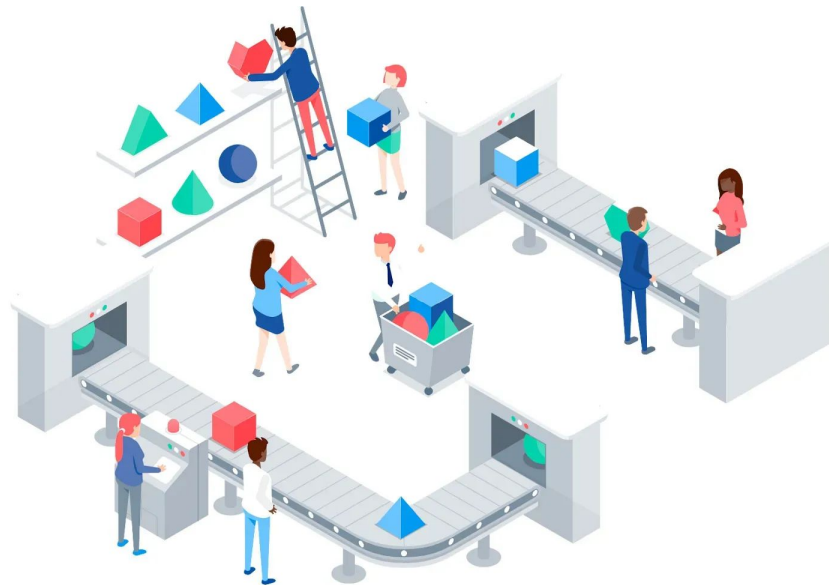
Hypervisor

Infrastructure



¿Qué es un *container*?

- Un *container* es una unidad de software estandarizada que “empaqueta el código y todas sus dependencias para que la aplicación se ejecute de forma rápida y de forma viable, de un entorno a otro”.





Contenerización de aplicaciones



- Se dice que existe un aumento de la eficiencia de la memoria, la CPU y el almacenamiento como ventajas clave de este enfoque, en comparación con la virtualización tradicional.
- Debido a que los contenedores de aplicación no tienen la sobrecarga requerida por las máquinas virtuales, es posible soportar muchos más contenedores en la misma infraestructura.
- La portabilidad es también un beneficio. Siempre y cuando la configuración del servidor sea idéntica entre los sistemas, un contenedor de aplicación puede ejecutarse en cualquier sistema y en cualquier nube sin requerir cambios de código.
- No hay variables de entorno de SO huésped o dependencias de biblioteca para administrar.
- Un inconveniente potencial de la contenerización es la falta de aislamiento del sistema operativo central. Dado que los contenedores de aplicación no se extraen del sistema operativo host en una máquina virtual, las amenazas de seguridad tienen un acceso más fácil a todo el sistema.



Docker

- La contenerización ganó prominencia con Docker (de código abierto).
- Los contenedores de Docker están diseñados para funcionar en todo, desde computadoras físicas hasta máquinas virtuales, servidores bare metal, clústeres de nube OpenStack e instancias públicas.
- Docker trabaja a nivel del núcleo del sistema operativo (tanto Linux como Windows). ¿Y esto en qué se traduce? En un importante ahorro tecnológico y económico.
- Para el departamento técnico, la utilización de Docker simplifica notablemente el paso de desarrollo a producción, siendo una herramienta muy utilizada en entornos DevOps.
- Docker maneja contenedores que se van desplegando progresivamente y que pueden lanzarse y cerrarse de forma muy rápida permitiendo gestionar eficientemente los recursos de la infraestructura donde se ejecuta.
- Estos contenedores funcionan a través de imágenes autocontenidas, es decir, que incluyen todos los ficheros necesarios para ejecutar la aplicación en el dispositivo y entorno que sea. Así, nunca más oirás entre tus técnicos eso de "en mi ordenador funcionaba" durante los ciclos de pruebas.



Docker

- Gracias a las bases sobre las que se asienta Docker, el *footprint* de la aplicación es mucho más ligero. Este sistema de contenedores sin duda ha revolucionado el proceso de despliegue y mantenimiento de aplicaciones.
- Es mucho más minimalista, más fácil de utilizar, fácil de mantener y tiene una curva de aprendizaje que llega a hacer que los técnicos perciban su interfaz como la propia de su empresa.
- ¿Y qué sucede con la seguridad? Docker permite desplegar las aplicaciones automáticamente y sin repeticiones. De esta manera, el procedimiento se vuelve mucho más ligero y seguro, ya que evita el error humano.
- Las imágenes firmadas garantizan la integridad de su contenido y al contener solo lo indispensable para la ejecución de la aplicación está menos expuesta a vulnerabilidades y ataques
- Docker es realmente interesante para desarrollar y mantener microservicios. El sistema de contenedores no monolítico permite que cada unidad que da servicio al usuario pueda reforzarse de forma única, con el consiguiente ahorro económico.
- De esta forma facilita el *continuous delivery* y reduce el *time to market* tanto en caso de error como para lanzar nuevas aplicaciones que hagan tu negocio más competitivo



Docker

- Docker realiza virtualizaciones a nivel de sistema operativo
- Con el lema de “Desarrollar una vez y ejecutar en cualquier sitio”, Docker pretende explotar el potencial de tu organización, dando a tus desarrolladores y equipos de IT, la libertad de desarrollar, gestionar y proporcionar un entorno seguro a las aplicaciones, y a los datos críticos para el negocio sin que estos se vean limitados por la tecnología, o por la [infraestructura](#).



Docker Ventajas y beneficios:

- **Ágil creación y despliegue de aplicaciones:** Mayor facilidad y eficiencia al crear imágenes de contenedor en vez de máquinas virtuales
- **Desarrollo, integración y despliegue continuo:** Permite que la imagen de contenedor se construya y despliegue de forma frecuente y confiable, facilitando los rollbacks pues la imagen es inmutable
- **Separación de tareas entre Dev y Ops:** Puedes crear imágenes de contenedor al momento de compilar y no al desplegar, desacoplando la aplicación de la infraestructura
- **Observabilidad** No solamente se presenta la información y métricas del sistema operativo, sino la salud de la aplicación y otras señales.



Docker Ventajas y beneficios:

- **Consistencia entre los entornos de desarrollo, pruebas y producción:** La aplicación funciona igual en un laptop y en la nube
- **Portabilidad entre nubes y distribuciones:** Funciona en Ubuntu, RHEL, CoreOS, tu datacenter físico, Google Kubernetes Engine y todo lo demás
- **Administración centrada en la aplicación:** Eleva el nivel de abstracción del sistema operativo y el hardware virtualizado a la aplicación que funciona en un sistema con recursos lógicos
- **Microservicios** distribuidos, elásticos, liberados y débilmente acoplados: Las aplicaciones se separan en piezas pequeñas e independientes que pueden ser desplegadas y administradas de forma dinámica, y no como una aplicación monolítica que opera en una sola máquina de gran capacidad
- **Aislamiento de recursos:** Hace el rendimiento de la aplicación más predecible
- **Utilización de recursos:** Permite mayor eficiencia y densidad



Contenerización vs Virtualización - Diferencias:

- Los *containers* y las **máquinas virtuales** tienen objetivos parecidos. Ambos quieren aislar una aplicación y sus dependencias en una unidad auto-contenida que se ejecute donde quieras. De la misma forma, los *containers* y las máquinas virtuales no requieren de recursos físicos.
- Los *containers* *virtualizan* el sistema operativo, y no el *hardware*, o la funcionalidad de un ordenador físico. Son, entonces, “más portables” y más eficientes. Usar los *containers* y las máquinas virtuales de forma conjunta puede proporcionar una mayor flexibilidad para desplegar y gestionar aplicaciones.
- Las máquinas virtuales son abstracciones del hardware físico que transforman un servidor en muchos servidores, mientras que los *contenedores* de Docker son una abstracción en la capa de la aplicación que empaquetan el código y sus dependencias.
- De esta forma, múltiples contenedores pueden ejecutarse en la misma máquina y compartir el núcleo del sistema operativo con otros contenedores que ejecutan sus procesos de forma aislada.



Docker (contenerización) vs Virtualización - continuación

- En el caso de los contenedores, el hecho de que no necesiten un sistema operativo completo sino que reutilicen el subyacente **reduce mucho la carga** que debe soportar la máquina física, **el espacio** de almacenamiento utilizado **y el tiempo** necesario para lanzar las aplicaciones. Un sistema operativo puede ocupar desde poco menos de 1GB para algunas distribuciones de Linux con lo mínimo necesario, hasta más de 10GB en el caso de un sistema Windows completo. Además, estos sistemas operativos, para funcionar requieren un mínimo de memoria RAM reservada, que puede ir desde 1 hasta varios GB, dependiendo de nuestras necesidades. Por lo tanto **los contenedores son mucho más ligeros que las máquinas virtuales**.



Docker (contenerización) vs Virtualización - continuación

- Cuando definimos **una máquina virtual debemos indicar de antemano cuántos recursos físicos le debemos dedicar**. Por ejemplo, podemos decir que nuestra VM va a necesitar 2 vCores (procesadores virtuales), 4GB de RAM y un espacio en disco de 100 GB. En el caso de los procesadores, es posible compartirlos entre varias máquinas virtuales (pero no conviene pasarse o irán fatal de rendimiento), y el espacio en disco se puede hacer que solo ocupe lo que de verdad se esté utilizando, de modo que crezca en función de las necesidades y no ocupe siempre tanto como habíamos reservado. Pero en el caso de la memoria y otros elementos (acceso a unidades externas o dispositivos USB) la reserva es total. Por eso, aunque nuestra aplicación no haga uso en realidad de los 4GB de RAM reservados da igual: no podrán ser utilizados por otras máquinas virtuales ni por nadie más. **En el caso de los contenedores** esto no es así. De hecho no indicamos qué recursos vamos a necesitar, sino que es Docker Engine, en función de las necesidades de cada momento, el encargado de **asignar lo que sea necesario para que los contenedores funcionen** adecuadamente.

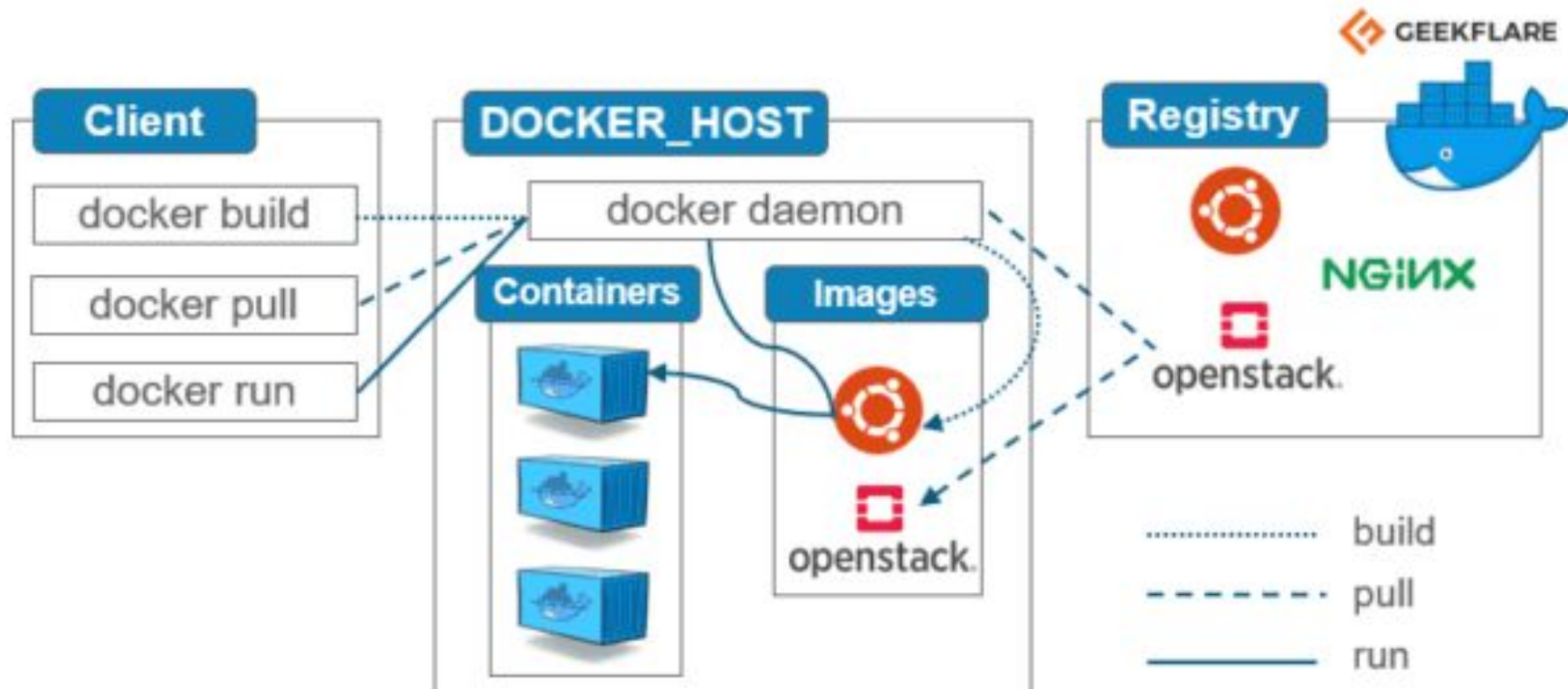


Docker (contenerización) vs Virtualización - continuación

- Esto hace que los entornos de ejecución de Docker sean mucho más ligeros, y que se aproveche mucho mejor el *hardware*, además de permitir levantar muchos más contenedores que VMs en la misma máquina física. Mientras que una VM puede tardar un minuto o más en arrancar y tener disponible nuestra aplicación, **un contenedor Docker se levanta y responde en unos pocos segundos** (o menos, según la imagen). **El espacio ocupado en disco es muy inferior con Docker** al no necesitar que instalemos el sistema operativo completo



Docker





¿Qué es Kubernetes?

- Kubernetes es una plataforma portable y extensible de código abierto para administrar cargas de trabajo y servicios. Kubernetes facilita la automatización y la configuración declarativa. Tiene un ecosistema grande y en rápido crecimiento. El soporte, las herramientas y los servicios para Kubernetes están ampliamente disponibles.



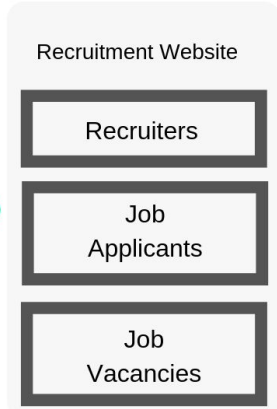
¿Qué es Kubernetes?



Monolithic Application

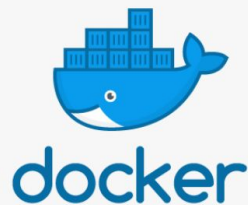


Transition to Microservices



Docker

Create containers for your application



Kubernetes

Launch your containerised application in K8s





Kubernetes

Kubernetes tiene varias características. Puedes pensar en Kubernetes como:

- una plataforma de contenedores
- una plataforma de microservicios
- una plataforma portable de nube

y mucho más.

Kubernetes ofrece un entorno de administración **centrado en contenedores**. Kubernetes orquesta la infraestructura de cómputo, redes y almacenamiento para que las cargas de trabajo de los usuarios no tengan que hacerlo. Esto ofrece la simplicidad de las Plataformas como Servicio (PaaS) con la flexibilidad de la Infraestructura como Servicio (IaaS) y permite la portabilidad entre proveedores de infraestructura.



Kubernetes

A pesar de que Kubernetes ya ofrece muchas funcionalidades, siempre hay nuevos escenarios que se benefician de nuevas características. Los flujos de trabajo de las aplicaciones pueden optimizarse para acelerar el tiempo de desarrollo. Una solución de orquestación propia puede ser suficiente al principio, pero suele requerir una automatización robusta cuando necesita escalar. Es por ello que Kubernetes fue diseñada como una plataforma: para poder construir un ecosistema de componentes y herramientas que hacen más fácil el desplegar, escalar y administrar aplicaciones.



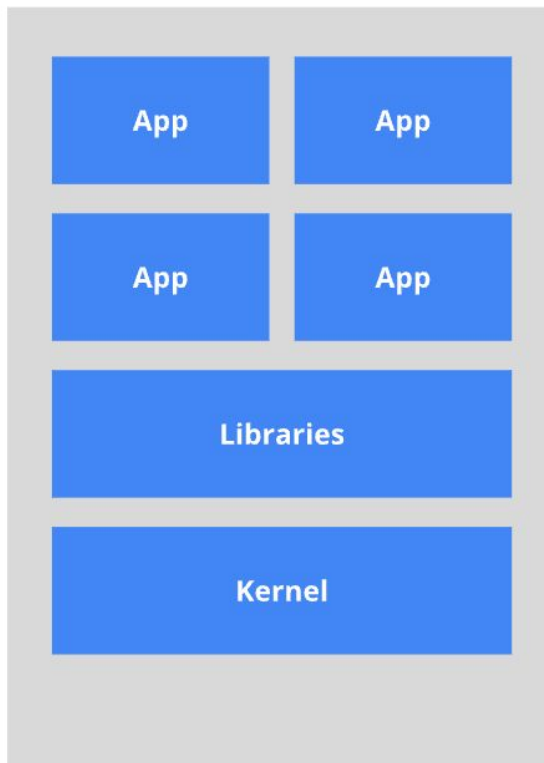
Kubernetes



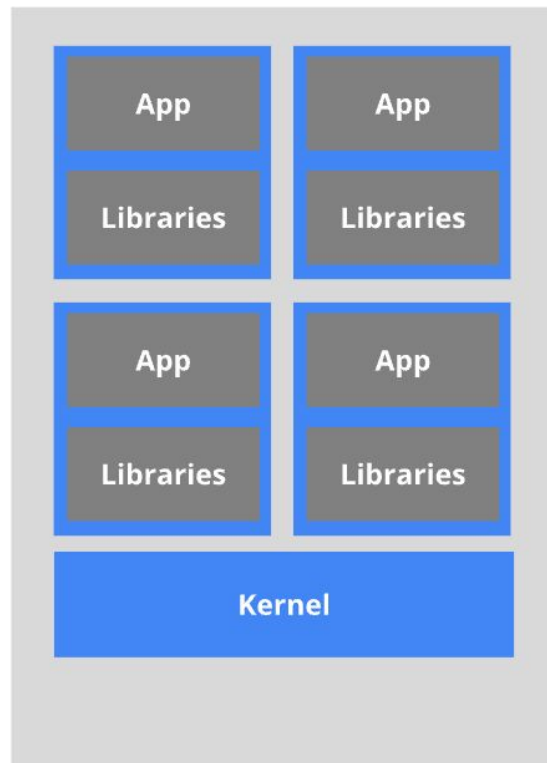
¿Por qué usar contenedores?

¿Te preguntas las razones para usar contenedores?

The old way: Applications on host



The new way: Deploy containers





Kubernetes - Componentes - Plano de Control



Los componentes que forman el plano de control toman decisiones globales sobre el clúster (por ejemplo, la planificación) y detectan y responden a eventos del clúster, como la creación de un nuevo pod cuando la propiedad replicas de un controlador de replicación no se cumple.

Estos componentes pueden ejecutarse en cualquier nodo del clúster. Sin embargo para simplificar, los scripts de instalación típicamente se inician en el mismo nodo de forma exclusiva, sin que se ejecuten contenedores de los usuarios en esos nodos. El plano de control se ejecuta en varios nodos para garantizar la alta disponibilidad.



Kubernetes - Componentes - Plano de Control



kube-apiserver

El servidor de la API es el componente del plano de control de Kubernetes que expone la API de Kubernetes. Se trata del frontend de Kubernetes, recibe las peticiones y actualiza acordeamente el estado en etcd.

La principal implementación de un servidor de la API de Kubernetes es kube-apiserver. Es una implementación preparada para ejecutarse en alta disponibilidad y que puede escalar horizontalmente para balancear la carga entre varias instancias.



Kubernetes - Componentes - Plano de Control



etcd

Almacén de datos persistente, consistente y distribuido de clave-valor utilizado para almacenar toda la información del clúster de Kubernetes.

Si tu clúster utiliza etcd como sistema de almacenamiento, échale un vistazo a la documentación sobre estrategias de backup.



Kubernetes - Componentes - Plano de Control



kube-scheduler

Componente del plano de control que está pendiente de los Pods que no tienen ningún nodo asignado y selecciona uno donde ejecutarlo.

Para decidir en qué nodo se ejecutará el pod, se tienen en cuenta diversos factores: requisitos de recursos, restricciones de hardware/software/políticas, afinidad y anti-afinidad, localización de datos dependientes, entre otros.



Kubernetes - Componentes - Plano de Control



kube-controller-manager

Componente del plano de control que ejecuta los controladores de Kubernetes.

Lógicamente cada controlador es un proceso Independiente, pero para reducir la complejidad, todos se compilan en un único binario y se ejecuta en un mismo proceso.

Estos controladores incluyen:

- Controlador de nodos: es el responsable de detectar y responder cuándo un nodo deja de funcionar
- Controlador de replicación: es el responsable de mantener el número correcto de pods para cada controlador de replicación del sistema
- Controlador de endpoints: construye el objeto Endpoints, es decir, hace una unión entre los Services y los Pods
- Controladores de tokens y cuentas de servicio: crean cuentas y tokens de acceso a la API por defecto para los nuevos Namespaces.



Kubernetes - Componentes - Plano de Control



cloud-controller-manager

cloud-controller-manager ejecuta controladores que interactúan con proveedores de la nube. El binario cloud-controller-manager es una característica alpha que se introdujo en la versión 1.6 de Kubernetes.

cloud-controller-manager sólo ejecuta ciclos de control específicos para cada proveedor de la nube. Es posible desactivar estos ciclos en kube-controller-manager pasando la opción `--cloud-provider= external` cuando se arranque el kube-controller-manager.



Kubernetes - Componentes - Plano de Control

cloud-controller-manager permite que el código de Kubernetes y el del proveedor de la nube evolucionen de manera independiente. Anteriormente, el código de Kubernetes dependía de la funcionalidad específica de cada proveedor de la nube. En el futuro, el código que sea específico a una plataforma debería ser mantenido por el proveedor de la nube y enlazado a cloud-controller-manager al correr Kubernetes.

Los siguientes controladores dependen de alguna forma de un proveedor de la nube:

- Controlador de nodos: es el responsable de detectar y actuar cuándo un nodo deja de responder
- Controlador de rutas: para configurar rutas en la infraestructura de nube subyacente
- Controlador de servicios: para crear, actualizar y eliminar balanceadores de carga en la nube
- Controlador de volúmenes: para crear, conectar y montar volúmenes e interactuar con el proveedor de la nube para orquestarlos



Kubernetes - Componentes - de Nodo



Los componentes de nodo corren en cada nodo, manteniendo a los pods en funcionamiento y proporcionando el entorno de ejecución de Kubernetes.

kubelet

Agente que se ejecuta en cada nodo de un clúster. Se asegura de que los contenedores estén corriendo en un pod.

El agente kubelet toma un conjunto de especificaciones de Pod, llamados PodSpecs, que han sido creados por Kubernetes y garantiza que los contenedores descritos en ellos estén funcionando y en buen estado.



Kubernetes - Componentes - de Nodo



kube-proxy

kube-proxy permite abstraer un servicio en Kubernetes manteniendo las reglas de red en el anfitrión y haciendo reenvío de conexiones.

Runtime de contenedores

El runtime de los contenedores es el software responsable de ejecutar los contenedores. Kubernetes soporta varios de ellos: Docker, containerd, cri-o, rktlet y cualquier implementación de la interfaz de runtime de contenedores de Kubernetes, o Kubernetes CRI.



Kubernetes - Addons



Los addons son pods y servicios que implementan funcionalidades del clúster. Estos pueden ser administrados por Deployments, ReplicationControllers y otros. Los addons asignados a un espacio de nombres se crean en el espacio kube-system.

Más abajo se describen algunos addons. Para una lista más completa de los addons disponibles, por favor visite [Addons](#).

DNS

Si bien los otros addons no son estrictamente necesarios, todos los clústers de Kubernetes deberían tener un DNS interno del clúster ya que la mayoría de los ejemplos lo requieren.

El DNS interno del clúster es un servidor DNS, adicional a los que ya podrías tener en tu red, que sirve registros DNS a los servicios de Kubernetes.

Los contenedores que son iniciados por Kubernetes incluyen automáticamente este servidor en sus búsquedas DNS.



Kubernetes - Addons

Interfaz Web (Dashboard)

El Dashboard es una interfaz Web de propósito general para clústeres de Kubernetes. Le permite a los usuarios administrar y resolver problemas que puedan presentar tanto las aplicaciones como el clúster.

Monitor de recursos de contenedores

El Monitor de recursos de contenedores almacena de forma centralizada series de tiempo con métricas sobre los contenedores, y provee una interfaz para navegar estos datos.

Registros del clúster

El mecanismo de registros del clúster está a cargo de almacenar los registros de los contenedores de forma centralizada, proporcionando una interfaz de búsqueda y navegación.



Kubernetes - Pods



Los Pods son las unidades de computación desplegables más pequeñas que se pueden crear y gestionar en Kubernetes.

Que es un Pod? Es un grupo de uno o más contenedores (como contenedores Docker), con almacenamiento/red compartidos, y unas especificaciones de cómo ejecutar los contenedores. Los contenidos de un Pod son siempre coubicados, coprogramados y ejecutados en un contexto compartido. Un Pod modela un "host lógico" específico de la aplicación: contiene uno o más contenedores de aplicaciones relativamente entrelazados. Antes de la llegada de los contenedores, ejecutarse en la misma máquina física o virtual significaba ser ejecutado en el mismo host lógico.

Mientras que Kubernetes soporta más runtimes de contenedores a parte de Docker, este último es el más conocido y ayuda a describir Pods en términos de Docker.



Kubernetes - Pods



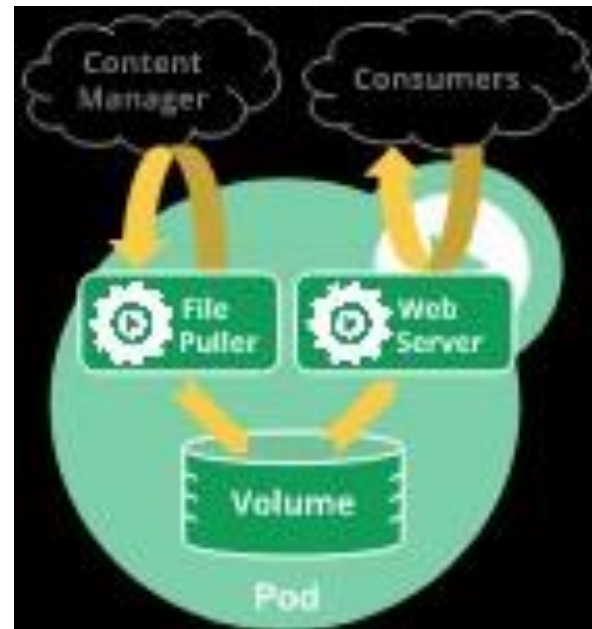
El contexto compartido de un Pod es un conjunto de namespaces de Linux, cgroups y, potencialmente, otras facetas de aislamiento, las mismas cosas que aíslan un contenedor Docker. Dentro del contexto de un Pod, las aplicaciones individuales pueden tener más subaislamientos aplicados.

Los contenedores dentro de un Pod comparten dirección IP y puerto, y pueden encontrarse a través de localhost. También pueden comunicarse entre sí mediante comunicaciones estándar entre procesos, como semáforos de SystemV o la memoria compartida POSIX. Los contenedores en diferentes Pods tienen direcciones IP distintas y no pueden comunicarse por IPC sin configuración especial. Estos contenedores normalmente se comunican entre sí a través de las direcciones IP del Pod



Kubernetes - Pods

Cuando se dice que algo tiene la misma vida útil que un Pod, como un volumen, significa que existe mientras exista ese Pod (con ese UID). Si ese Pod se elimina por cualquier motivo, incluso si se crea un reemplazo idéntico, el recurso relacionado (por ejemplo, el volumen) también se destruye y se crea de nuevo.





Kubernetes - Pods



Las aplicaciones dentro de un Pod también tienen acceso a volúmenes compartidos, que se definen como parte de un Pod y están disponibles para ser montados en el sistema de archivos de cada aplicación.

En términos de Docker, un Pod se modela como un grupo de contenedores de Docker con namespaces y volúmenes de sistemas de archivos compartidos.

Al igual que los contenedores de aplicaciones individuales, los Pods se consideran entidades relativamente efímeras (en lugar de duraderas).

Kubernetes - Pods - Usos

Los Pods pueden ser usados para alojar pilas de aplicaciones integradas (por ejemplo, LAMP), pero su objetivo principal es apoyar los programas de ayuda coubicados y coadministrados, como:

sistemas de gestión de contenido, loaders de datos y archivos, gestores de caché locales, etc.

copia de seguridad de registro y punto de control, compresión, rotación, captura de imágenes, etc.

observadores de cambio de datos, adaptadores de registro y monitoreo, publicadores de eventos, etc.

proxies, bridges y adaptadores.

controladores, configuradores y actualizadores.

Los Pods individuales no están diseñados para ejecutar varias instancias de la misma aplicación, en general.