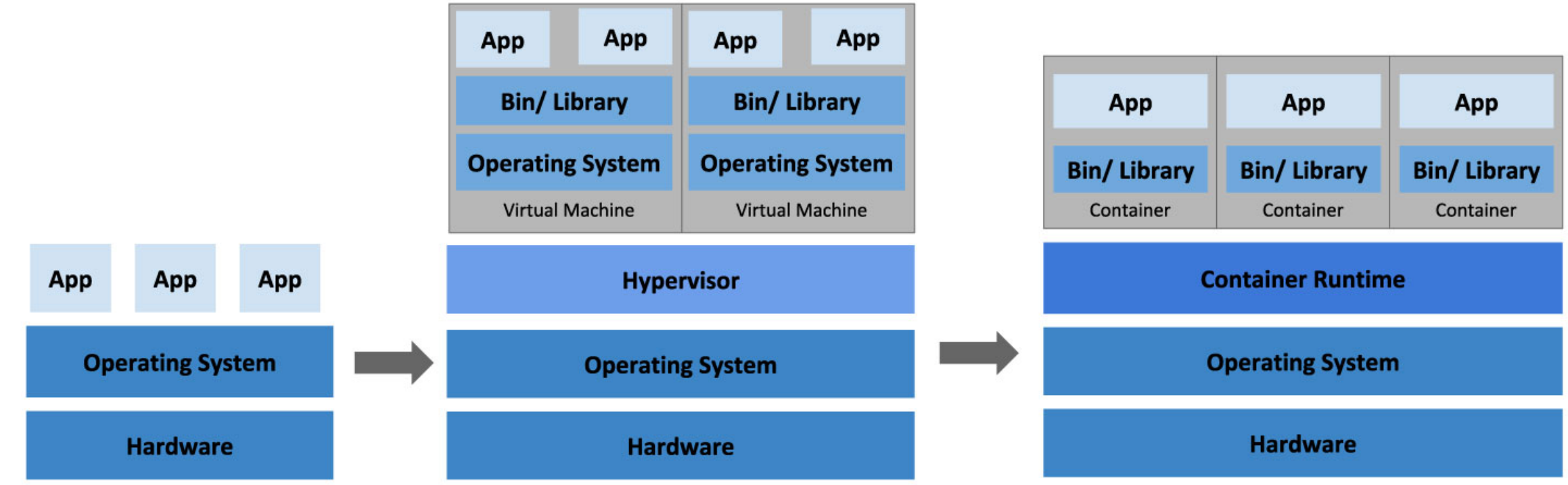
ORIGEN DE LOS CONTENEDORES

El origen de la contenedorización



VIRTUALIZACIÓN VS CONTENEDORIZACIÓN

1979: UNIX V7

Durante el desarrollo de Unix V7 en 1979 se introdujo el sistema chroot.

Chroot permitía cambiar el directorio raíz de un proceso y sus hijos a una nueva ubicación en el sistema de archivos.

Este avance fue el inicio del aislamiento de procesos: segregar el acceso a archivos para cada proceso.

Chroot se añadió a BSD en 1982.

2000: FREEBSD JAILS

Un proveedor de hosting de entorno compartido presentó FreeBSD jails.

Lograba una clara separación entre sus servicios y los de sus clientes, por seguridad y facilidad de administración.

FreeBSD Jails permite a los administradores separar un sistema informático FreeBSD en varios sistemas independientes, más pequeños, denominados «cárceles».

Cada uno podía tener su propia dirección IP.

2001: LINUX VSERVER

Linux VServer es otro mecanismo de «cárcel» símilar a FreeBDS Jails.

Permitía separar los recursos (sistemas de archivos, direcciones de red, memoria) de un sistema informático.

2004: SOLARIS CONTAINERS

Sun microsystems lanzó las zonas de solaris.

Permitían virtualizar los recursos del sistema y asignarlos a zonas, sobre las que se ejcutaban procesos.

Las zonas de solaris admitían clonación e instantáneas.

2005: OPEN VZ (OPEN VIRTUZZO)

Se trata de una tecnología de virtualización a nivel de sistema operativo que utiliza un kernel Linux parcheado para virtualización, aislamiento y checkpointing.

El código no fue lanzado como parte del kernel oficial de Linux.

2006: PROCESS CONTAINERS

Llanzado por Google en 2006.

Fue diseñado para limitar, contabilizar y aislar el uso de recursos (CPU, memoria, I/O de disco, red) de una colección de procesos.

Fue renombrado como «Grupos de control (cgroups)» un año después y finalmente se fusionó con el kernel 2.6.24 de Linux.

2008: LXC

LXC (LinuX Containers) fue la primera y más completa implementación del gestor de contenedores Linux.

Se dió en el año 2008 utilizando cgroups y namespaces de Linux, y funcionaba en un solo kernel de Linux sin necesidad de parches.

2011: WARDEN

CloudFoundry comenzó Warden en 2011, utilizando LXC en las etapas iniciales y posteriormente lo reemplazó con su propia implementación.

2013: LMCTFY

Let Me Contain That For You (LMCTFY) se lanzó en 2013 como una versión de código abierto de los contenedores de Google, proporcionando contenedores de aplicaciones Linux.

Las aplicaciones podían crear y administrar sus propios subcontenedores.

DOCKER

La explosión de la contenedorización

DOCKER

Cuando Docker emerge en 2013, los contenedores explotan en popularidad.

El crecimiento de Docker y el uso de contenedores van de la mano.

Con Docker, los desarrolladores pueden crear y ejecutar rápidamente contenedores, así como descargarlos y distribuirlos.

HISTORIA DE DOCKER

Docker Inc. se creó en un grupo de incubadoras entre 2010 y 2011.

Su producto Docker fue lanzado en marzo de 2013, bajo modalidad opensource.

- En 2013: Red Hat y Docker anunciaron una colaboración entre Fedora, Red Hat Enterprise Linux (RHEL) y OpenShift.
- En 2014: Microsoft anunció la integración del motor Docker en Windows Server.
- En 2014: Se anunciaron los servicios de contenedores Docker para Amazon Elastic Compute Cloud (EC2) y para IBM Cloud.
- En 2016: Microsoft da soporte nativo a Docker en Windows 10.

ECOSISTEMA DOCKER

docker-hub, docker engine, docker-compose...

ECOSISTEMA DOCKER

Docker es un conjunto de piezas de software para trabajar con contenedores:

- Docker Engine: Es un motor de contenedores. Permite la descarga y creación de imágenes, y la ejecución de contenedores. Existen dos distribuciones:
  - Docker CE: Opensource y gratuita. Su sitio oficial es https://www.docker.com
  - Docker EE: Código privativo y de pago. Su sitio oficial es https://www.mirantis.com
- Docker desktop: Para Windows y Mac. Permite realizar las operaciones de docker engine a través de una interfaz gráfica.

ECOSISTEMA DOCKER

- Docker repositories: Permiten almacenar, gestionar y descargar imágenes de contenedores.
  - Docker Hub: Es el repositorio oficial y más utilizado de imágenes de contenedores. Se puede utilizar en dos modalidades:
    - Modo gratuito: Donde las imágenes que se crean son de acceso público
    - Modo por subscripción: Donde las imágenes que se crean y comparten son privadas
- Docker compose: Permite trabajar con aplicaciones distribuidas en varios contenedores.

DOCKER ENGINE

Docker Engine es una tecnología de contenedorización open source para construir y ejecutar aplicaciones.

Docker Engine actúa como una aplicación cliente-servidor con:

- Un servidor con un proceso de demonio de larga ejecución llamado dockerd. El daemon crea y administra objetos Docker, como imágenes, contenedores, redes y volúmenes.
- Un API (REST) que especifica interfaces que los programas pueden usar para comunicarse con el demonio Docker.
- Un CLI (interfaz de línea de comando) llamado docker, que utiliza las API de Docker para controlar o interactuar con el demonio Docker a través de scripts o comandos de CLI directos.

DOCKER ENGINE

Utiliza la virtualización a nivel del sistema operativo para ejecutar los contenedores.

Los contenedores están aislados unos de otros, pero pueden comunicarse entre sí.

Todos los contenedores son ejecutados por un solo núcleo del sistema operativo y, por lo tanto, usan menos recursos que las máquinas virtuales.

QUE PERMITE DOCKER ENGINE

- Docker tiene la capacidad de reducir el tamaño del desarrollo al proporcionar una huella más pequeña del sistema operativo a través de contenedores.
- Con los contenedores, es más fácil para los equipos de diferentes unidades, como desarrollo, control de calidad y operaciones, trabajar sin problemas en todas las aplicaciones.
- Puede ejecutar contenedores en cualquier lugar, en cualquier máquina física y virtual e incluso en la nube.
- Como los contenedores Docker son bastante livianos, son fácilmente escalables.

DOCKER COMPOSE

Docker Compose es una herramienta para definir y ejecutar aplicaciones Docker de contenedores múltiples.

Utiliza archivos YAML (docker-compose.yml) para configurar los servicios (contenedores) de la aplicación y realiza el proceso de creación y puesta en marcha de todos los contenedores con un solo comando.

DOCKER COMPOSE

Dispone de un CLI que permite a los usuarios ejecutar comandos en varios contenedores a la vez, por ejemplo, crear imágenes, ejecutar contenedores o pararlos.

La primera versión beta pública de Docker Compose (versión 0.0.1) se lanzó el 21 de diciembre de 2013.

La primera versión lista para producción (1.0) se puso a disposición el 16 de octubre de 2014.

DOCKER ENGINE

Containerd, runc, contenedores, imágenes...

ARQUITECTURA DE DOCKER ENGINE

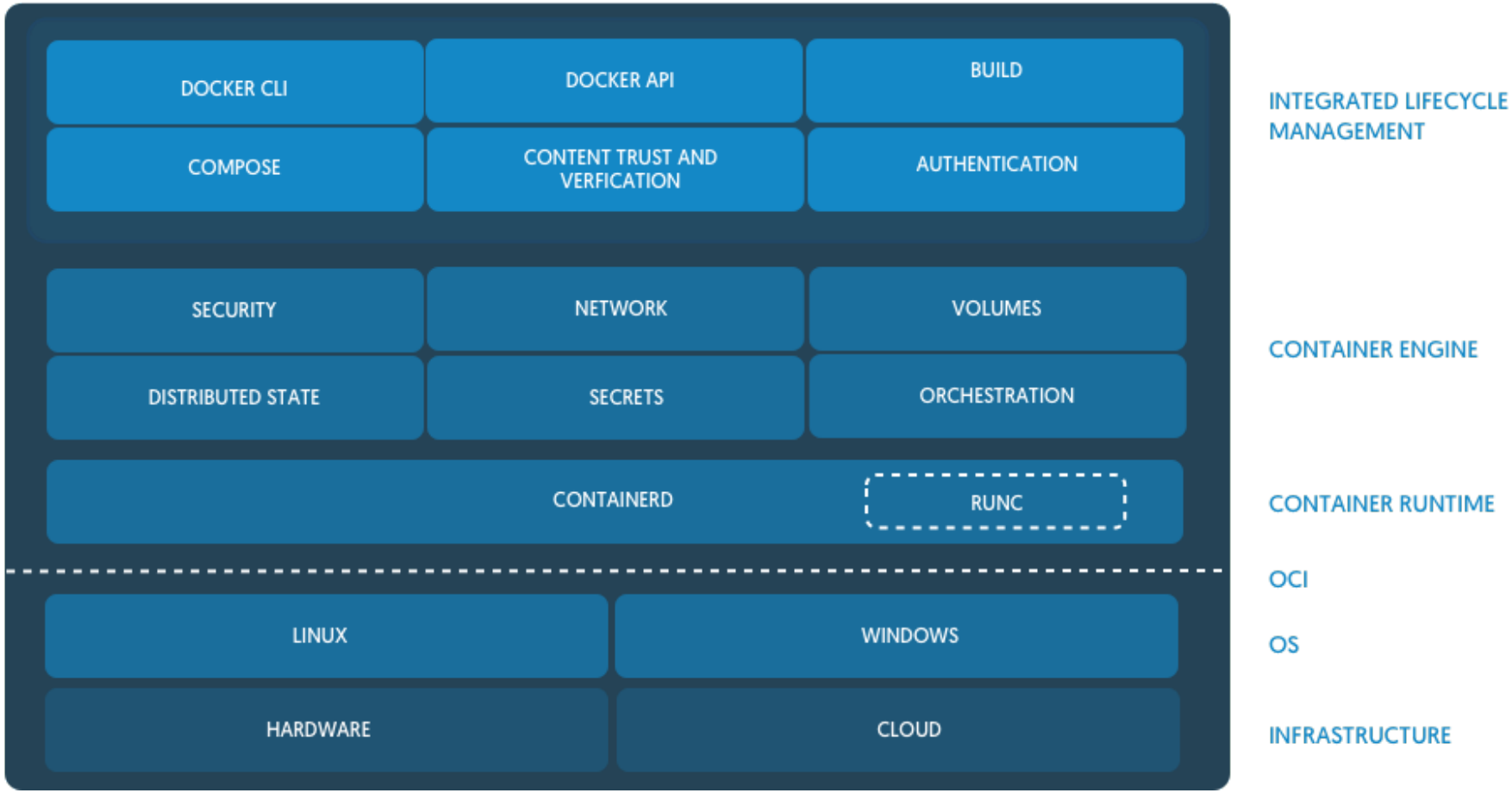
En su origen, Docker se basaba en la tecnología de virtualización a nivel de sistema operativo (SO) para Linux LXC.

Al año fue sustituida por una tecnología propia de virtualización escrita en el lenguaje de programación Go, llamada libcontainer.

Libcontainer fue donada a la Open Containers Initiative, y hoy en día se denomina runc.

Posteriormente la capa de gestión de imágenes y contenedores fue también desligada y donada a la Cloud Native Computing Foundation, que la distribuye bajo el nombre containerd.

CONTAINERD


DOCKER CLI
DOCKER API
BUILD
COMPOSE
CONTENT TRUST AND VERFICATION
AUTHENTICATION
SECURITY
NETWORK
VOLUMES
DISTRIBUTED STATE
SECRETS
ORCHESTRATION
CONTAINERD
RUNC
LINUX
WINDOWS
HARDWARE
CLOUD
INTEGRATED LIFECYCLE MANAGEMENT
CONTAINER ENGINE
CONTAINER RUNTIME
OCI
OS
INFRASTRUCTURE

IMÁGENES Y CONTENEDORES DOCKER

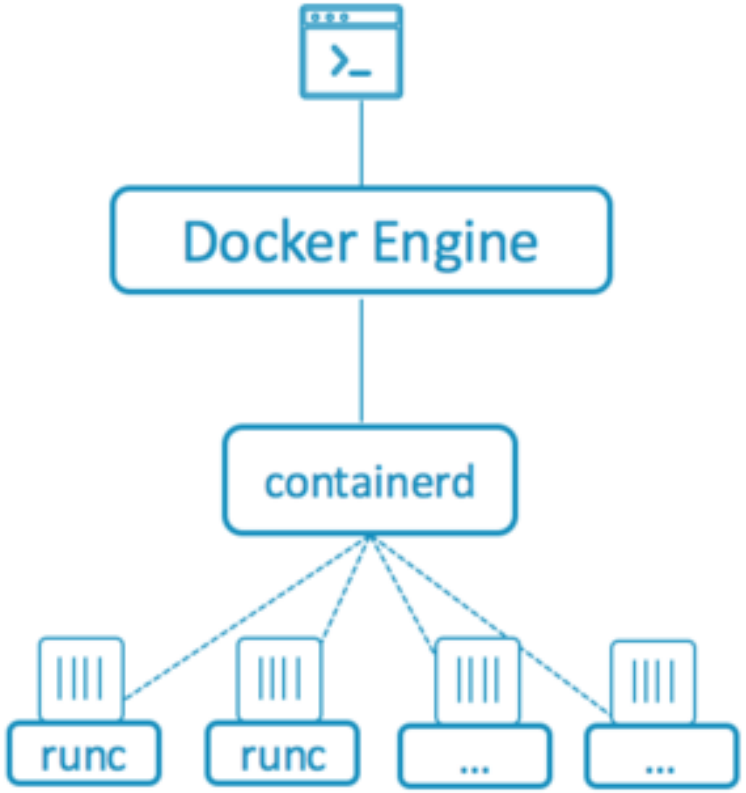
Una imagen es un paquete de software ligero, independiente y ejecutable que incluye todo lo necesario para ejecutar una aplicación:

- Aplicación
- Herramientas del sistema
- Bibliotecas
- Configuraciones del sistema

Las imágenes se convierten en contenedores en tiempo de ejecución.

Los contenedores aíslan el software de su entorno y aseguran que el software que contienen siempre se ejecutará igual, independientemente de la infraestructura.

ARQUITECTURA DOCKER



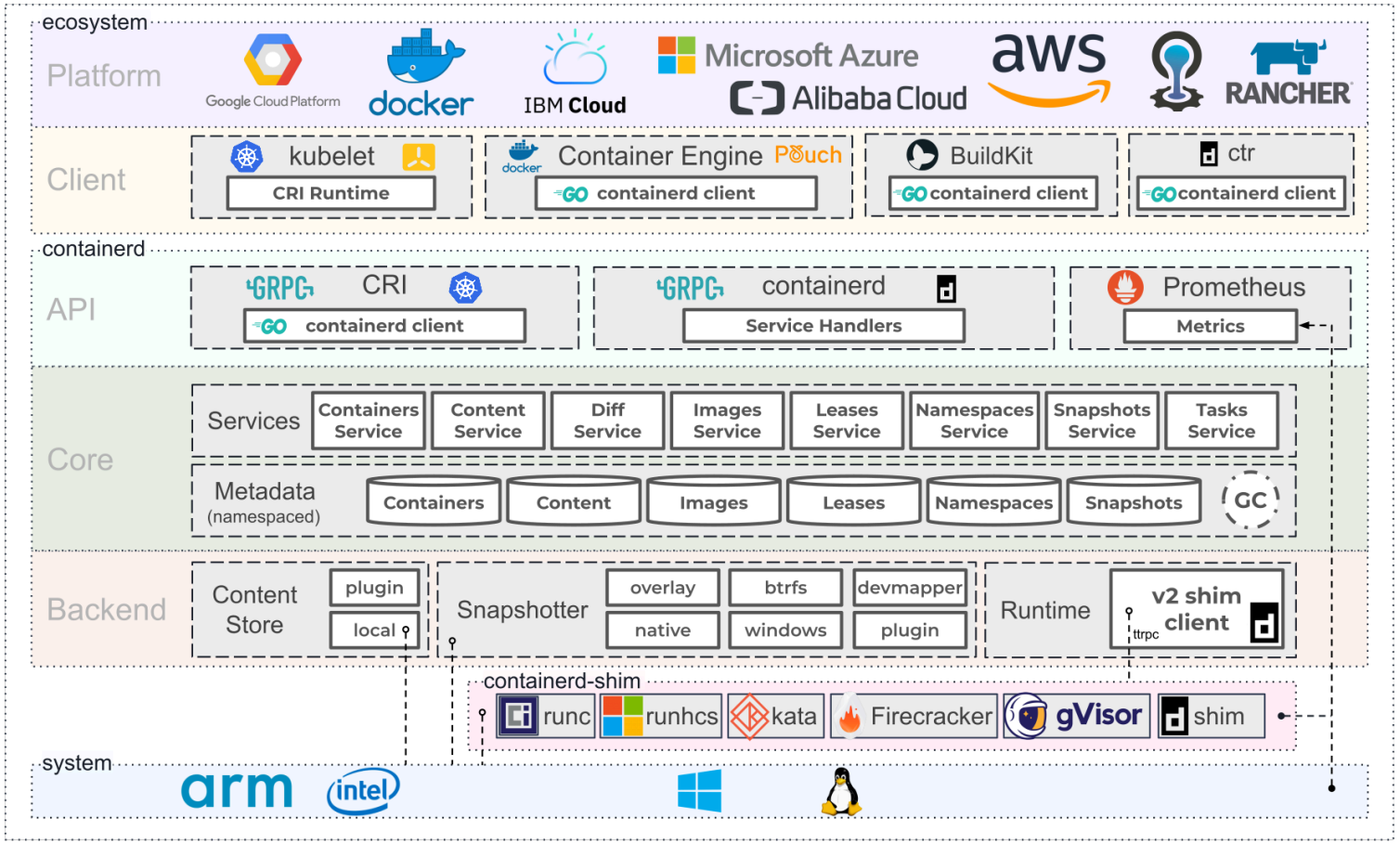
Same Docker UI and commands

User interacts with the Docker Engine

Engine communicates with containerd

containerd spins up runc or other OCI compliant runtime to run containers

IMÁGENES CONTAINERD



CARACTERÍSTICAS DE LOS CONTENEDORES

- Estándar: Docker creó el estándar de la industria para contenedores, lo que permite su portabilidad
- Ligeros: los contenedores comparten el núcleo del sistema operativo de la máquina y, por lo tanto, no requieren un sistema operativo por aplicación, lo que aumenta la eficiencia del servidor y reduce los costos de servidor y licencia
- Más seguros: las aplicaciones son más seguras en contenedores por las capacidades de aislamiento de los contenedores

KUBERNETES

Aplicaciones multicontenedor

¿QUÉ ES KUBERNETES?

Kubernetes es una plataforma portátil, extensible y de código abierto para gestionar cargas de trabajo y servicios en contenedores.

El nombre Kubernetes se origina del griego, que significa timonel o piloto.

Google abrió el proyecto Kubernetes en 2014.

¿POR QUÉ KUBERNETES?

Los contenedores son una buena forma de agrupar y ejecutar aplicaciones.

En un entorno de producción, es necesario administrar los contenedores que ejecutan las aplicaciones y asegurarse de que no haya tiempo de inactividad.

Por ejemplo, si un contenedor se cae, otro contenedor debe comenzar.

Kubernetes proporciona un marco para ejecutar sistemas distribuidos de manera resistente.

Se encarga del escalado y la conmutación por error de las aplicaciones.

FUNCIONALIDADES DE KUBERNETES

Descubrimiento de servicios y equilibrio de carga

- Kubernetes puede exponer un contenedor utilizando el nombre DNS o su propia dirección IP.
- Si el tráfico a un contenedor es alto, Kubernetes puede equilibrar la carga y distribuir el tráfico de la red para que la implementación sea estable.

Orquestación de almacenamiento

- Kubernetes permite montar automáticamente un sistema de almacenamiento de su elección, como almacenes locales, proveedores de nube pública y más.

FUNCIONALIDADES DE KUBERNETES

Despliegues y retrocesos automatizados

- Permite describir el estado deseado de los contenedores desplegados y cambiar el estado real al estado deseado a una velocidad controlada.
- Por ejemplo, puede automatizarse Kubernetes para crear nuevos contenedores para una implementación, eliminar los contenedores existentes y adoptar todos sus recursos para el nuevo contenedor.

FUNCIONALIDADES DE KUBERNETES

Empaquetado automático de contenedores

- Kubernetes gestiona un grupo de nodos que se puede usar para ejecutar tareas en contenedores.
- Kubernetes gestiona cuánta CPU y memoria (RAM) necesita cada contenedor.
- Kubernetes gestiona los contenedores en los nodos para aprovechar al máximo los recursos.

Autocuración

- Kubernetes reinicia contenedores que fallan, sustituye contenedores, mata los que no responden al chequeo de salud definido, y los anuncia a los clientes cuando estén listos para servir.

FUNCIONALIDADES DE KUBERNETES

Administración de secrets y de configuración

- Kubernetes permite almacenar y administrar información confidencial, como contraseñas, tokens OAuth y claves SSH.
- Pueden implementarse y actualizarse los secrets y la configuración de la aplicación sin reconstruir las imágenes del contenedor y sin exponer los secrets.

OBJETOS DE KUBERNETES

Pods, Services, ReplicaSets, Volumes...

PODS

Un pod es un nivel más alto de abstracción para agrupar componentes contenedorizados.

Un pod consta de uno o más contenedores que se garantiza que se ubican en la máquina host y pueden compartir recursos.

La unidad de programación básica en Kubernetes es un pod.

A cada pod en Kubernetes se le asigna una dirección IP exclusiva dentro del clúster, lo que permite que las aplicaciones usen puertos sin riesgo de conflicto.

PODS

Dentro del pod, todos los contenedores pueden hacer referencia entre sí mediante localhost, pero un contenedor dentro de un pod no tiene forma de dirigirse directamente a otro contenedor dentro de otro pod. Para eso, tiene que usar la Dirección IP del Pod.

Sin embargo, un desarrollador de aplicaciones nunca debe usar la Dirección IP del Pod para hacer referencia / invocar una capacidad en otro pod, ya que las direcciones IP del Pod son efímeras; el pod específico al que hacen referencia puede asignarse a otra dirección IP al reiniciar.

En su lugar, deberían usar una referencia a un Servicio, que contiene una referencia al pod de destino.

PODS

Un pod puede definir un volumen, como un directorio de disco local o un disco de red, y exponerlo a los contenedores en el pod.

Los volúmenes son también la base de las características de Kubernetes para ConfigMaps (proporcionar acceso a la configuración a través del sistema de archivos visible para un contenedor) y Secrets (para proporcionar acceso a las credenciales necesarias para acceder a recursos remotos de forma segura, proporcionando esas credenciales en el sistema de archivos visible solo para contenedores autorizados).

CONJUNTOS DE RÉPLICA

El propósito de un ReplicaSet es mantener un conjunto estable de Pods replicados en ejecución en un momento dado.

Se usa para garantizar la disponibilidad de un número específico de Pods idénticos.

La definición de un conjunto de réplicas se hace a través de un selector, cuya evaluación dará como resultado la identificación de todos los pods asociados.

SERVICIOS

Un servicio de Kubernetes es un conjunto de pods que trabajan juntos.

Kubernetes proporciona dos modos de descubrimiento de servicios, usando variables ambientales o usando Kubernetes DNS.

El descubrimiento del servicio asigna una dirección IP estable y un nombre DNS al servicio, y equilibra la carga de trabajo de forma circular (round-robin) de esa dirección IP entre los pods que coinciden con el selector (incluso cuando las fallas provocan que los pods se muevan desde máquina a máquina).

Por defecto, un servicio se expone dentro de un clúster, pero también puede exponerse fuera del mismo.

VOLÚMENES

Por defecto, los sistemas de archivos en los contenedores de Kubernetes proporcionan almacenamiento efímero.

Esto significa que un reinicio del pod elimina todos los datos en los contenedores.

Un volumen de Kubernetes proporciona almacenamiento persistente que existe durante toda la vida útil del pod.

Este almacenamiento también se puede utilizar como espacio de disco compartido para contenedores dentro del pod.

Los volúmenes se montan en puntos de montaje específicos dentro del contenedor, que están definidos por la configuración del pod.

ESPACIOS DE NOMBRES

Kubernetes proporciona una partición de los recursos que administra en conjuntos no superpuestos llamados Namespaces.

Están diseñados para su uso en entornos con muchos usuarios repartidos en múltiples equipos o proyectos, o incluso separando entornos como el desarrollo, las pruebas y la producción.

CONFIGMAPS Y SECRETS

Un desafío común de la aplicación es decidir dónde almacenar y administrar la información de configuración, algunos de los cuales pueden contener datos confidenciales.

Los datos de configuración pueden ser tan específicos como propiedades individuales o información detallada en archivos de configuración JSON o XML.

Kubernetes proporciona dos mecanismos estrechamente relacionados para hacer frente a esta necesidad: "mapas de configuración" y "secretos".

Permiten realizar cambios de configuración sin requerir una compilación de la aplicación.

Los datos de los mapas de configuración y los secretos estarán disponibles para cada instancia de la aplicación a la que se han vinculado estos objetos a través de la implementación.

CONFIGMAPS Y SECRETS

Un secreto y / o un mapa de configuración solo se envía a un nodo si un pod en ese nodo lo requiere.

Kubernetes lo mantendrá en la memoria de ese nodo.

Una vez que se elimina el pod que depende del secreto o del mapa de configuración, también se elimina la copia en memoria de todos los secretos y mapas de configuración vinculados.

CONFIGMAPS Y SECRETS

El pod puede acceder a los datos a través de una de dos maneras:

- Como variables de entorno (que Kubernetes creará cuando se inicie el pod)
- Desde el sistema de archivos del contenedor que solo es visible desde dentro del pod.

Los datos en sí se almacenan en el maestro, que es una máquina altamente segura a la que nadie debería tener acceso de inicio de sesión.

STATEFULSETS

Es muy fácil abordar el escalado de aplicaciones sin estado: uno simplemente agrega más pods en ejecución, que es algo que Kubernetes hace muy bien.

Las cargas de trabajo con estado son mucho más difíciles, ya que el estado necesita ser preservado si se reinicia un pod, y si la aplicación se escala hacia arriba o hacia abajo, entonces el estado puede necesitar ser redistribuido.

STATEFULSETS

Las bases de datos son un ejemplo de cargas de trabajo con estado.

Cuando se ejecutan en modo de alta disponibilidad, muchas bases de datos vienen con la noción de una instancia primaria y una instancia secundaria.

En este caso, la noción de ordenamiento de instancias es importante.

Otras aplicaciones como Kafka distribuyen los datos entre sus brokers, por lo que un brokers no es lo mismo que otro.

En este caso, la noción de unicidad de instancia es importante.

STATEFULSETS

Los StatefulSets son controladores que son proporcionados por Kubernetes para hacer cumplir las propiedades de unicidad y orden entre las instancias de un pod y que se pueden usar para ejecutar aplicaciones con estado.

ARQUITECTURA DE KUBERNETES

Aplicaciones multicontenedor

ARQUITECTURA DE KUBERNETES

Cuando se instala, configura y ejecuta Kubernetes, se obtiene un clúster.

Un clúster de Kubernetes consta de un conjunto de máquinas de trabajo, llamadas nodos, que ejecutan aplicaciones en contenedores.

Cada clúster tiene al menos un nodo de trabajo.

Los nodos de trabajo alojan los Pods, que son los componentes que ejecutan la carga de trabajo de la aplicación.

El plano de control, gestiona los nodos de trabajo y los pods en el clúster.

En entornos de producción, el plano de control generalmente se ejecuta en varias computadoras y un clúster generalmente ejecuta múltiples nodos, lo que proporciona tolerancia a fallas y alta disponibilidad.

COMPONENTES DEL PLANO DE CONTROL

Los componentes del plano de control toman decisiones globales sobre el clúster.

También detectan y responden a eventos de cluster.

Los componentes del plano de control se pueden ejecutar en cualquier máquina del clúster.

Por simplicidad, los scripts de configuración suelen iniciar todos los componentes del plano de control en la misma máquina y no ejecutan contenedores de usuario en esta máquina.

KUBE-APISERVER

El API server es un componente del plano de control de Kubernetes que expone la API de Kubernetes.

El API server es el front-end para el plano de control de Kubernetes.

La implementación más común de API server de Kubernetes es kube-apiserver.

Permite escalado horizontal: se escala mediante la implementación de más instancias.

Pueden ejecutarse varias instancias de kube-apiserver y equilibrar el tráfico entre esas instancias.

ETCD

Es un almacén de pares clave-valor consistente y de alta disponibilidad.

Es utilizado por kubermetes como almacén de respaldo de todos los datos del clúster.

KUBE-SCHEDULER

Vigila los Pods creados sin nodo asignado y selecciona un nodo para que se ejecuten.

Los factores que se tienen en cuenta para las decisiones incluyen:

- requisitos de recursos individuales y colectivos
- restricciones de hardware / software
- especificaciones de afinidad y antiafinidad
- ubicación de datos
- interferencia entre cargas de trabajo
- plazos

KUBE-CONTROLLER-MANAGER

Ejecuta los procesos del controlador, que incluyen:

- Node controller (Controlador de nodos): responsable de controlar y responder cuando algún nodo se cae
- Replication controller (Controlador de replicación): responsable de mantener el número correcto de pods en el sistema
- Endpoints controller (Controlador de puntos finales): vincula los Servicios y los Pods
- Service Account & Token controllers (Cuenta de servicio y controladores de tokens): crea cuentas predeterminadas y tokens de acceso API para los nuevos espacios de nombres.

CLOUD-CONTROLLER-MANAGER

Incorpora lógica de control específica de la nube.

El administrador del controlador de la nube permite vincular su clúster a la API de un proveedor de la nube y separa los componentes que interactúan con esa plataforma de la nube de los componentes que solo interactúan con su clúster.

Si está ejecutando Kubernetes on-premises, el clúster no tiene un administrador de controlador de nube.

Por ejemplo, los siguientes controladores pueden tener dependenciasde proveedores en la nube:

- Controlador de nodos: para determinar si un nodo se ha eliminado en la nube después de que deja de responder
- Controlador de ruta: para configurar rutas en la infraestructura de la nube
- Controlador de servicio: para crear, actualizar y eliminar equilibradores de carga del proveedor de la nube

COMPONENTES DE NODO

Los componentes de nodo se ejecutan en cada nodo.

Mantienen los pods en ejecución.

Proporcionan el entorno de tiempo de ejecución de Kubernetes.

KUBELET

Se asegura de que los contenedores se ejecuten en un Pod.

Kubelet toma un conjunto de especificaciones de Pods que se proporcionan y garantiza que los contenedores descritos estén funcionando en buen estado.

Kubelet no gestiona contenedores que no fueron creados por Kubernetes.

KUBE-PROXY

kube-proxy es un proxy de red que se ejecuta en cada nodo.

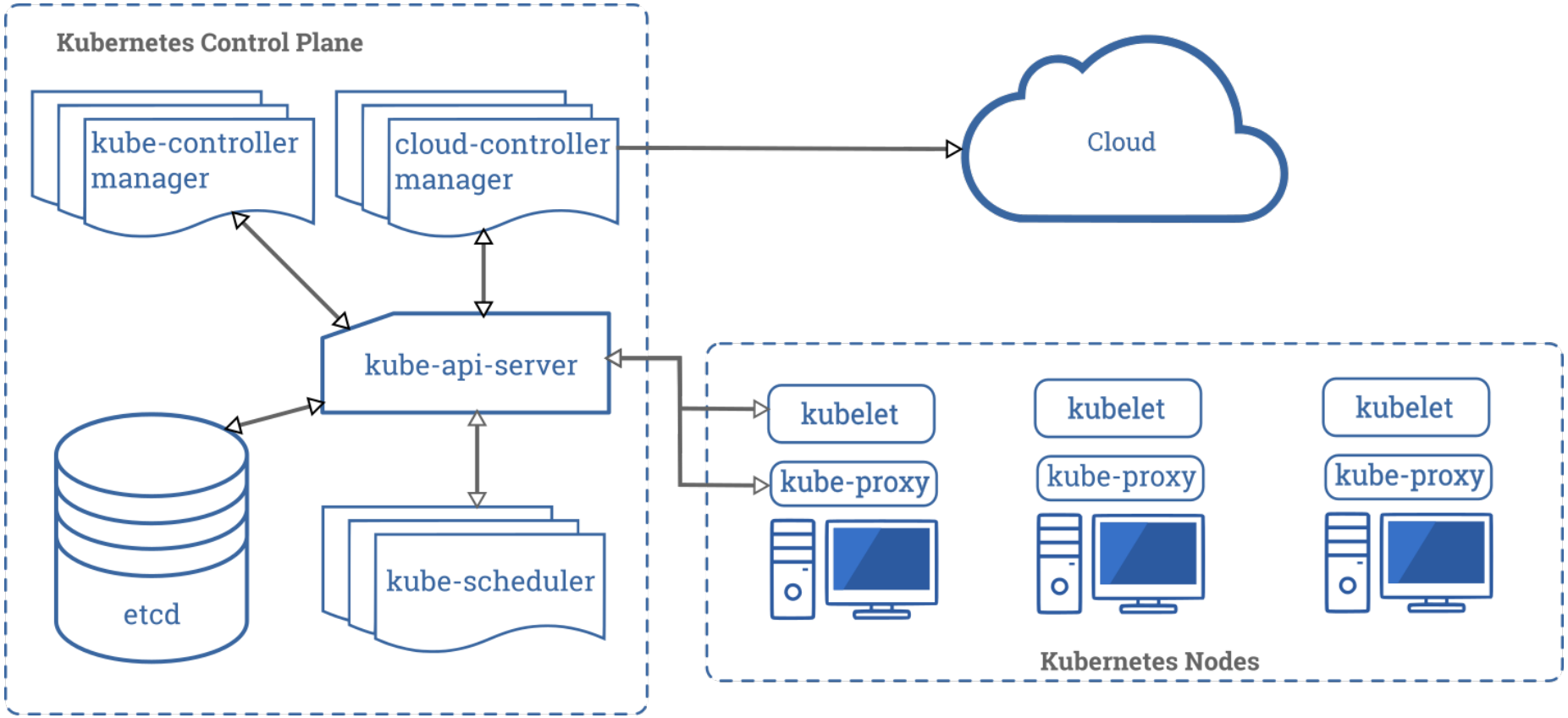
kube-proxy mantiene las reglas de red en los nodos. Estas reglas de red permiten la comunicación con los Pods desde dentro del cluster o desde fuera del mismo.

kube-proxy usa la capa de filtrado de paquetes del sistema operativo si hay una y está disponible. De lo contrario, kube-proxy se encarga del trabajo.

CONTAINER RUNTIME

Es el software responsable de ejecutar los contenedores.

Kubernetes admite varios tiempos de ejecución de contenedores: Docker, containerd , CRI-O , y cualquier implementación de Kubernetes CRI (Interfaz de tiempo de ejecución de contenedores de Kubernetes)



ARQUITECTURA DE KUBERNETES

COMPLEMENTOS

Los complementos usan los recursos de Kubernetes para implementar otras funcionalidades.

Debido a que proporcionan características a nivel de clúster, los recursos para complementos pertenecen al espacio de nombres kube-system.

DNS

Si bien los otros complementos no son estrictamente necesarios, todos los clústeres de Kubernetes deben tener un DNS de clúster.

Es un servidor DNS, independiente de los otros servidores DNS que se utilicen, que sirve registros DNS para los servicios de Kubernetes.

Los contenedores iniciados por Kubernetes incluyen automáticamente este servidor DNS entre sus DNS.

WEBUI (DASHBOARD)

Dashboard es una interfaz de usuario basada en web de uso general para clústeres de Kubernetes.

Permite a los usuarios administrar el cluster y solucionar problemas de aplicaciones que se ejecutan en él.

CONTAINER RESOURCE MONITORING

La supervisión de recursos de contenedores registra métricas genéricas sobre contenedores en una base de datos central y proporciona una interfaz de usuario para examinar esos datos.

CLUSTER-LEVEL LOGGING

Un mecanismo de log a nivel de clúster es responsable de guardar los logs de los contenedores en un almacén de log central con interfaz de búsqueda / exploración.