Engineer

Kubernetes

⁂ Sistema que mantiene un conjunto de contenedores en marcha

Qué es

Kubernetes, a menudo escrito K8s, es software de código abierto que ejecuta aplicaciones en contenedores sobre un grupo de máquinas, llamado clúster, y las mantiene de acuerdo con una descripción que usted proporciona: qué imagen ejecuta cada aplicación, cuántas copias existen, cuánta memoria y CPU puede utilizar cada copia y cómo acceden a ella otros programas. La documentación del proyecto lo llama «una plataforma de código abierto, portátil y extensible, para gestionar cargas de trabajo y servicios en contenedores». El nombre significa timonel o piloto en griego; K8s conserva la K y la s y sustituye las ocho letras intermedias por su número; Google publicó el proyecto como código abierto en 2014.

Un clúster tiene dos mitades. El plano de control conserva la descripción: el servidor de API es la única interfaz HTTP por la que pasa todo cambio, etcd almacena lo que se le indica, el planificador asigna cada Pod nuevo a un nodo adecuado y el gestor de controladores ejecuta los bucles que actúan sobre la descripción. En cada nodo de trabajo, kubelet se asegura de que los Pods asignados estén en ejecución, kube-proxy mantiene las reglas de red de los Services y un entorno de ejecución de contenedores, como containerd o CRI-O, inicia los contenedores. Docker Engine dejó de ser un entorno de ejecución integrado en Kubernetes 1.24, y las imágenes construidas con Docker se ejecutan sin cambios porque son imágenes OCI estándar.

La descripción es un conjunto de objetos, normalmente escritos en YAML. Un Pod es la unidad más pequeña que planifica Kubernetes: uno o varios contenedores que comparten una dirección de red y almacenamiento en una máquina. Un Deployment indica cuántos Pods idénticos deben existir y los sustituye paso a paso cuando cambia su plantilla. Un Service da un nombre y una dirección estables a un conjunto cambiante de Pods, seleccionando sus miembros por etiqueta. Una aplicación web pequeña tiene este aspecto:

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: web
spec:
  replicas: 3
  selector:
    matchLabels:
      app: web
  template:
    metadata:
      labels:
        app: web
    spec:
      containers:
        - name: web
          image: registry.example.com/web:1.4.2
          resources:
            requests:
              cpu: 250m
              memory: 256Mi
            limits:
              memory: 256Mi
          readinessProbe:
            httpGet:
              path: /healthz
              port: 8080
```

Suele compararse con tres opciones más pequeñas. Docker Compose describe contenedores en una máquina y, cuando esta desaparece, también desaparecen ellos. Un gestor de servicios como systemd puede reiniciar un contenedor que falle y ponerlo en marcha al arrancar, lo que cubre muchas necesidades con mucha menos maquinaria, pero no sabe trasladar el trabajo a otra máquina. Las plataformas alojadas, como Google Cloud Run o Azure Container Apps, ejecutan contenedores sin entregarle un plano de control que deba operar, a cambio de ofrecer menos ajustes. La documentación también es clara sobre lo que Kubernetes no es: una plataforma como servicio tradicional que lo incluya todo. No construye su aplicación ni proporciona bases de datos o colas de mensajes propias.

Utilice el concepto en una decisión

Aplique «¿Qué es Kubernetes?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Software empaquetado para comportarse igual en un portátil, un servidor de pruebas y producción, y ampliar su capacidad cuando aumente la actividad.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. 20 aplicaciones Docker en Google Cloud y AWS — 20 aplicaciones en contenedores desplegadas y monitorizadas

20 aplicaciones Docker en Google Cloud y AWS Póngase en contacto →

Por qué importa

La idea central es que el clúster trabaja continuamente para ajustarse a una descripción escrita, en lugar de ejecutar una lista de órdenes una vez. Aplicar el archivo anterior no inicia nada directamente: registra un deseo. Los controladores son bucles de control que observan el clúster y actúan para acercar el estado actual al deseado. Si una máquina falla y tres copias pasan a ser dos, un controlador detecta la diferencia, se crea un Pod nuevo y el planificador lo sitúa en una máquina con capacidad. Como la descripción es un conjunto de archivos, puede residir en un repositorio y pasar por la misma revisión que el código que ejecuta.

La plataforma incorpora los despliegues de versiones. La estrategia predeterminada es una actualización gradual: se inician Pods nuevos y los antiguos se detienen a medida que los nuevos están listos, dentro de límites que, de forma predeterminada, permiten hasta un 25 % de Pods adicionales y hasta un 25 % no disponibles en cualquier momento. Cada cambio de la plantilla del Pod se conserva como revisión —diez de forma predeterminada—, y `kubectl rollout undo` vuelve a la anterior. El script que antes retiraba el tráfico de un servidor, sustituía un binario y lo reincorporaba se convierte en comportamiento integrado.

El planificador también cambia la economía del conjunto de máquinas. Sitúa cada Pod según la CPU y la memoria que solicita, por lo que muchos servicios pequeños comparten máquinas en lugar de recibir cada uno un servidor dimensionado para su peor día.

El coste es real. Un clúster es en sí mismo un sistema distribuido, con versiones que actualizar según un calendario, una capa de red, otra de almacenamiento, certificados y control de acceso, todo lo cual alguien debe entender cuando falla. Las ofertas gestionadas, como Google Kubernetes Engine, Amazon EKS y Azure Kubernetes Service, ejecutan el plano de control por usted, pero todos los manifiestos y, normalmente, los nodos de trabajo siguen siendo su responsabilidad. Para unos pocos servicios en una o dos máquinas, un gestor de servicios sencillo suele cumplir la función con menos piezas.

Cómo se aplica en la práctica

La página de tecnologías de la empresa expresa la regla claramente: Docker y Kubernetes para las cargas de trabajo que lo justifican, y unidades systemd sencillas para las demás. Kubernetes se elige para cada carga según el tamaño y la naturaleza del trabajo, no como opción predeterminada.

Un encargo muestra cargas que lo justificaban. Se desplegaron y mantuvieron veinte aplicaciones Docker en Google Cloud y AWS, entre ellas aplicaciones de analítica basadas en R, con configuración y monitorización coherentes para que ninguna se convirtiera en un caso especial. La resolución de problemas se hacía mediante Podman y la depuración de la orquestación mediante Kubernetes. La página informa de diagnósticos más rápidos y despliegues estables y reproducibles.

La otra mitad de la regla aparece en la plataforma propia de la empresa. Una capa de contenedores basada en Podman y Quadlet ejecuta cada contenedor como un servicio del supervisor del propio sistema operativo, descrito con la misma forma declarativa que todo lo demás en el host. Los contenedores utilizan la red del host y no publican puertos, de modo que las reglas del cortafuegos instaladas por el rol de refuerzo de seguridad siguen al mando. Donde Kubernetes distribuiría una imagen nueva mediante su API, un usuario de despliegue sin root ejecuta un único script fijo que construye la imagen y reinicia el servicio. En un host, eso da a cada contenedor una descripción escrita y el supervisor en el que el host ya confía, sin un segundo gestor de procesos. Lo que deja fuera es la distribución entre máquinas y la sustitución de un nodo que falla, que es donde Kubernetes empieza a compensar su peso.

Dónde surgen los problemas

La mayoría de los problemas de Kubernetes proceden de la distancia entre lo que pide un manifiesto y lo que informa el clúster.

Un despliegue queda a medias mientras la tarea de despliegue informa de éxito. Cuando los Pods nuevos nunca están listos, por una etiqueta de imagen incorrecta, un fallo al arrancar, una sonda de disponibilidad que falla o falta de cuota, el Deployment deja de avanzar. Tras el plazo de progreso, cuyo valor predeterminado es 600 segundos, registra el motivo `ProgressDeadlineExceeded`, y la documentación indica que Kubernetes no hace nada más: los Pods antiguos

que siguen en ejecución continúan atendiendo y nada revierte por sí solo. `kubectl rollout status` termina con un código distinto de cero cuando vence el plazo, por lo que un paso de despliegue puede esperarlo y revertir la versión:

```
kubectl apply -f web.yaml
kubectl rollout status deployment/web --timeout=10m \
  || kubectl rollout undo deployment/web
```

Un contenedor se reinicia con `OOMKilled` o se ralentiza en un nodo con CPU libre. El núcleo hace cumplir los límites y ambos recursos se comportan de forma distinta. Un contenedor que supera su límite de memoria puede terminarse y reiniciarse; al alcanzar su límite de CPU se restringe su uso, por lo que se ralentiza sin terminar, lo que parece un problema de la aplicación. El planificador sitúa los Pods según lo que solicitan, no lo que consumen: peticiones bajas saturan un nodo y las altas dejan máquinas ociosas. Un límite especificado sin petición se convierte en la petición. Ajuste la memoria al uso medido y trate los límites de CPU como una decisión deliberada.

Se fuerza la terminación de una aplicación de arranque lento antes de que complete el inicio. Una sonda de vitalidad que falla hace que kubelet termine y reinicie el contenedor, por lo que una aplicación que necesita un minuto para cargar y se comprueba a los diez segundos se reinicia indefinidamente. Una sonda de arranque aplaza las demás comprobaciones hasta que la aplicación funciona. Una sonda de disponibilidad retira un Pod de su Service sin reiniciarlo, lo que encaja con un Pod ocupado o que está calentando una caché. Una sonda de vitalidad que consulta la base de datos reinicia todas las réplicas a la vez cuando esta va lenta, convirtiendo una dependencia lenta en una interrupción del servicio.

Un Pod iniciado a mano desaparece y una corrección manual se deshace. Un Pod creado de forma independiente permanece en su nodo hasta que termina, se elimina, se expulsa o falla el nodo, y ningún controlador lo recrea; la documentación recomienda un Deployment o un Job incluso para una sola copia. Un cambio realizado con `kubectl edit` vive solo en el clúster, y la siguiente aplicación desde el repositorio restaura el valor anterior, por lo que una corrección de emergencia debe incorporarse a los archivos en cuanto pase la emergencia.

Lecturas adicionales

- Visión general de Kubernetes — la explicación del propio proyecto de qué es Kubernetes, qué ofrece y qué deja fuera.
- Pods — la unidad más pequeña que planifica Kubernetes, su ciclo de vida y por qué rara vez se crean Pods a mano.
- Deployments — actualizaciones graduales, reversiones, historial de revisiones y plazo de progreso.
- Service — nombres y direcciones estables ante un conjunto cambiante de Pods.
- Gestión de recursos para Pods y contenedores — peticiones, límites y cómo hace cumplir el núcleo cada uno.
- Configurar sondas de vitalidad, disponibilidad y arranque — las tres sondas y cómo proteger un contenedor de arranque lento.

Glosario

Elija según sus restricciones reales

Describa la ruta de despliegue, los riesgos operativos y quiénes se encargan de las guardias. Podemos definir el alcance de un cambio con comprobaciones, una vía de recuperación y responsabilidades claras.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es Kubernetes?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/kubernetes/