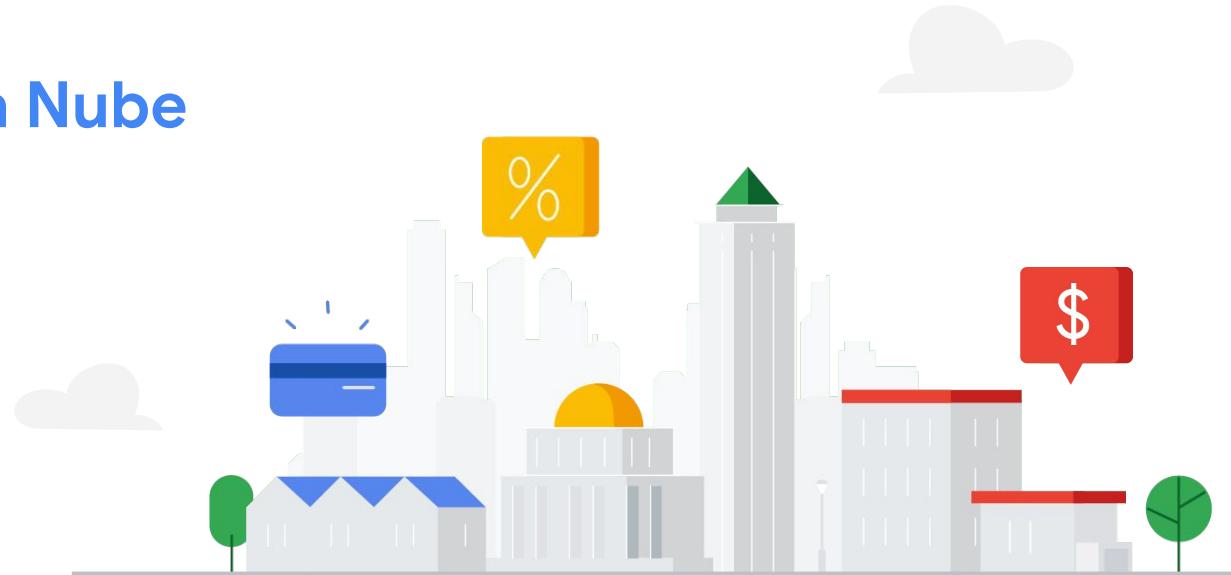


Contenedores en la Nube

Modulo 1



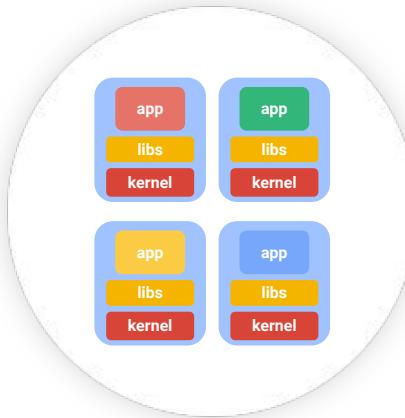
Introducción a los Contenedores



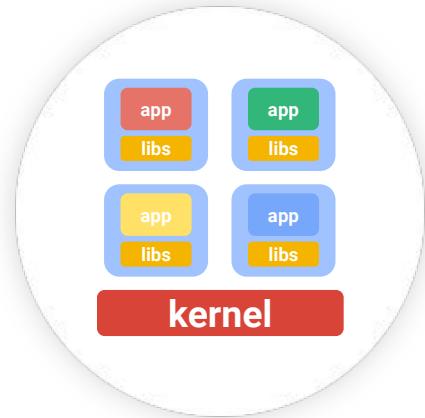
La evolución del cómputo



Máquinas compartidas



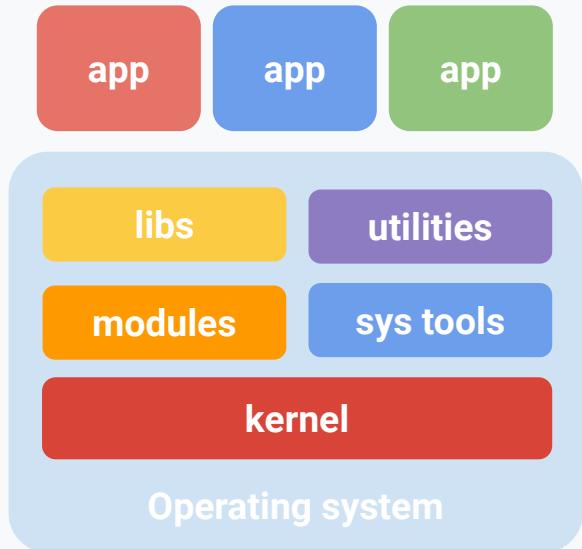
VMs / bare metal



Contenedores

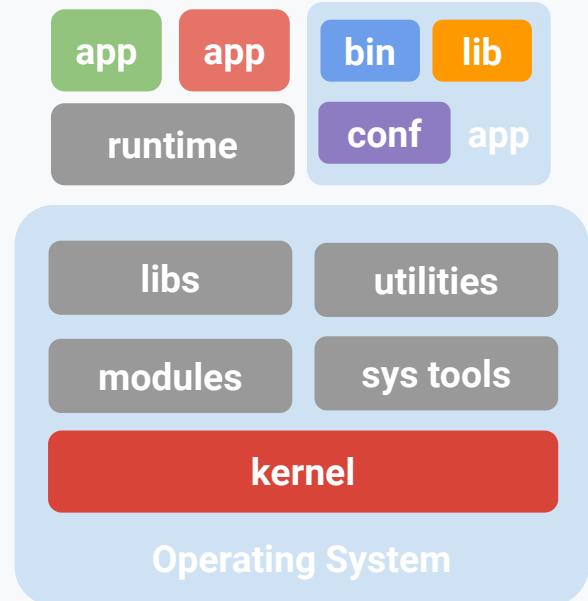
Máquinas compartidas

- Kernel — la parte principal del sistema operativo
- Módulos del Kernel (ip_tables, nfs)
- Controladores del dispositivo
- Herramientas del sistema (terminal, compilador, administración del disco)
- Librerías (socket, ssl, crypto)
- Utilerias (file manager)
- Otros



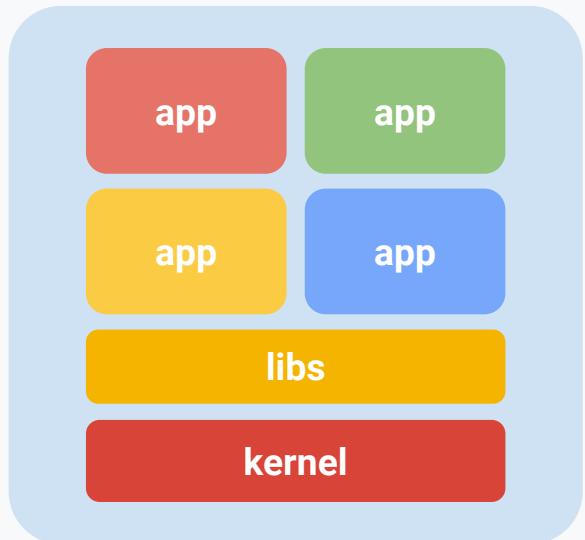
Dependencias aplicativas

- Kernel (requerido)
- Otras dependencias (opcionales)
 - Runtimes del Lenguaje (Java, Node.js, etc.)
 - Modulos del Kernel
 - Herramientas del sistema
 - Librerías de sistema
 - Configuración



La manera antigua: Instalar aplicaciones en el host

- Múltiples aplicaciones por máquina
- Dependencias compartidas
- Beneficios:
 - Ejecutables de tamaño pequeño
 - Menos uso de almacenamiento y memoria
 - Mayor utilización de recursos
- Problemas
 - Las dependencias se mezclaban unas con otras y con la máquina huésped
 - Una aplicación puede acaparar recursos, impactando a otras



La manera antigua: Instalar aplicaciones en el host

- Múltiples aplicaciones por máquina
- Dependencias compartidas
- Beneficios:
 - Ejecutables de tamaño pequeño
 - Menos uso de almacenamiento y memoria
 - Mayor utilización de recursos
- Problemas
 - Las dependencias se mezclaban unas con otras y con la máquina huésped
 - Una aplicación puede acaparar recursos, impactando a otras
 - **Difícil de replicar y comúnmente existían diferencias entre ambientes (¡corre bien en desarrollo!)**

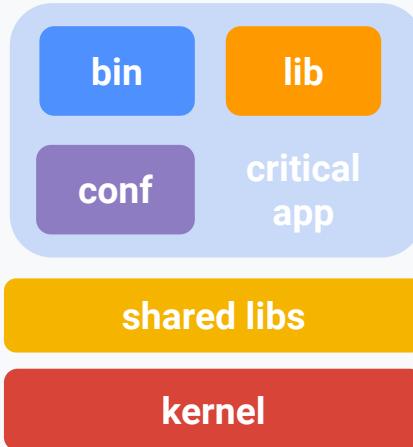


La solución (temporal):

- Dedicar máquinas huésped para las aplicaciones críticas
- Empaquetar todas las dependencias con la aplicación

Problemas

- Dedicar máquinas huésped desperdiciaba recursos
- Máquinas físicas toman tiempo en aprovisionar y configurar

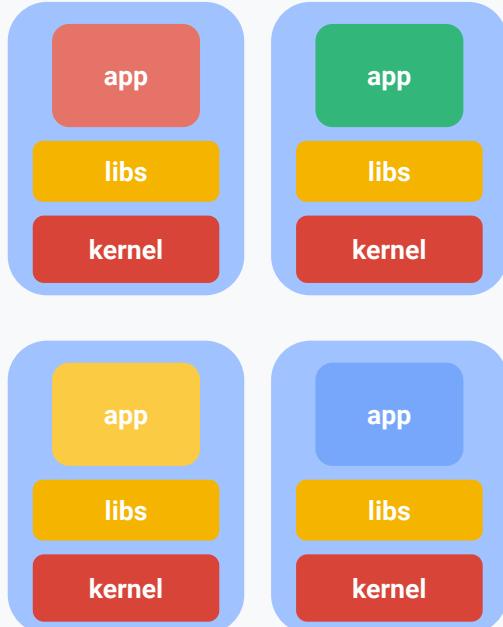


La era de la virtualización y de la nube

- Permitir que el hardware físico sea compartido entre aplicaciones como máquinas virtuales (VMs)
- Soluciones como Chef/Puppet/Ansible eran comúnmente utilizadas para administrar máquinas huésped y aplicativos
- VMs inmutables daban despliegues y rollbacks predecibles

Problemas

- Recursos desperdiciados (mucho overhead)
- Las VMs tardan mucho en iniciar

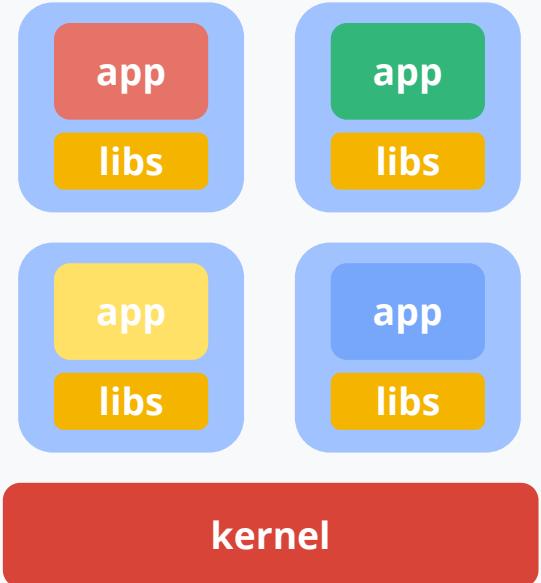


Mientras tanto en Google

- Los desarrolladores añadieron capacidades al kernel de Linux para soportar el aislamiento de procesos, poniendo las bases para la contenerización
 - Grupos de control (cgroups) --límites de recursos
 - Namespaces —aislamiento de procesos
 - Change Root (chroot) — aislamiento del sistema de archivos (ya existía)

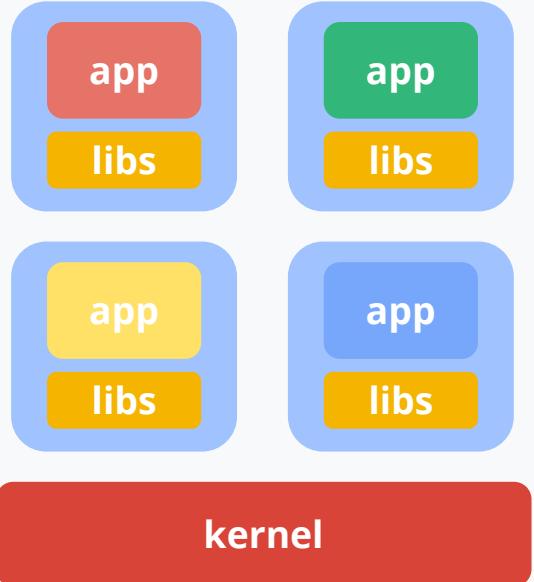
Esto permitió a las apps/procesos:

- Correr con cpu y memoria asignada (específica)
- Aislar de otros procesos
- Proveer acceso limitado al sistema de archivos



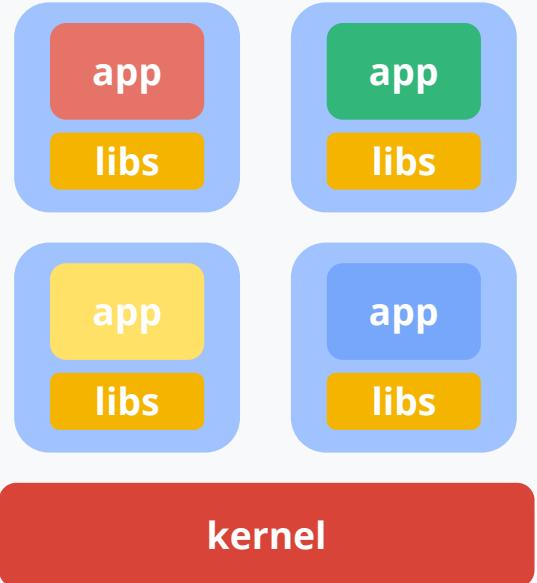
La nueva manera: Desplegar contenedores

- Virtualización a nivel OS (namespaces, cgroups, ...)
- Aislado, entre otros procesos y el host
 - Las imágenes de contenedor tienen su propio sistema de archivos, sus propios recursos y se ejecutan como su propio proceso
- Rápido



La nueva manera: Desplegar contenedores

- Portable entre sistemas operativos y entre nubes
 - Siempre y cuando el kernel objetivo fuera el mismo que el de la máquina huésped
- Ambientes consistentes entre desarrollo y producción
 - Creando imágenes inmutables durante el proceso de construcción, en lugar de durante el proceso de despliegue

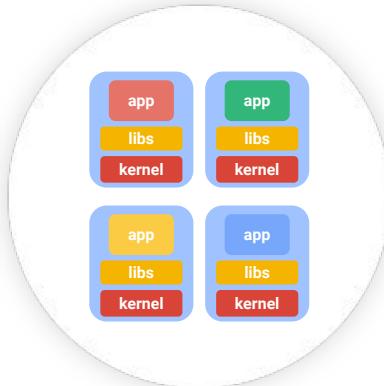


La evolución del cómputo



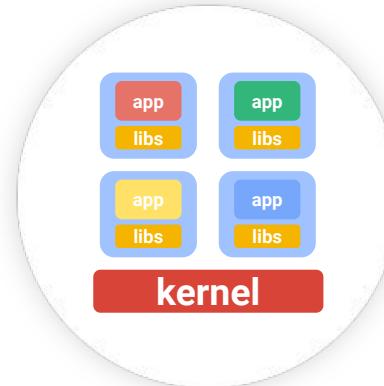
Máquinas compartidas

- ✖ Sin aislamiento
- ✖ Librerías en común
- ✖ Alto acoplamiento entre apps y SO



VMs / bare metal

- ✓ Aislamiento
- ✓ Sin librerías en común
- ✖ Caras e inefficientes
- ✖ Difíciles de mantener

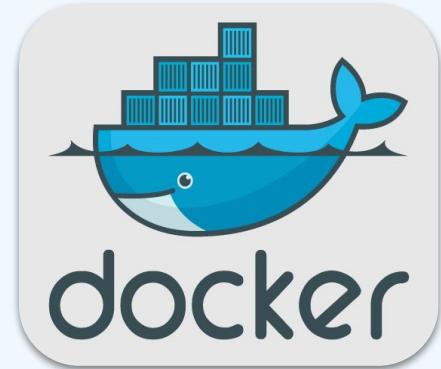


Contenedores

- ✓ Aislamiento
- ✓ Sin librerías en común
- ✓ Menos overhead
- ✓ Menos dependencia del sistema operativo huésped

Docker

- Herramienta de contenedores dominante
- Liberada como OSS en 2013 por DotCloud inc. (hosted PaaS)
- Logró crear y ejecutar imágenes de contenedores fácil y rápido**
- Adopción hyper-acelerada
- DotCloud Inc. → Docker Inc. (vendiendo el negocio PaaS)
- Empezó a cambiar su negocio (up the stack) en 2015 por presiones de generación de ingresos



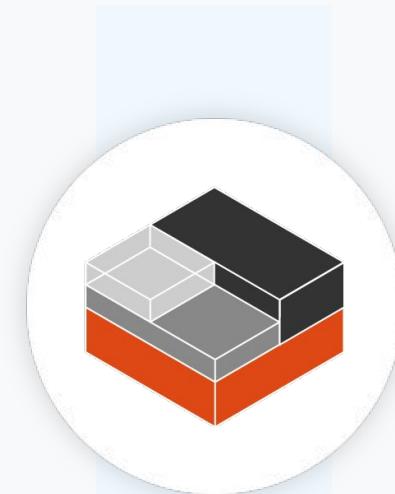
Docker (Hoy)

- Docker es un formato o especificación para construir imágenes de contenedores
- Imagen de contenedor Docker – binario empaquetado con un sistema operativo (sin el kernel y todas las dependencias)
- Docker container – proceso aislado (cgroup/chroot jail)
- Container registry (Docker Hub/GCR/Quay) – repositorio central de imágenes. Punto de acceso para la descarga de imágenes.

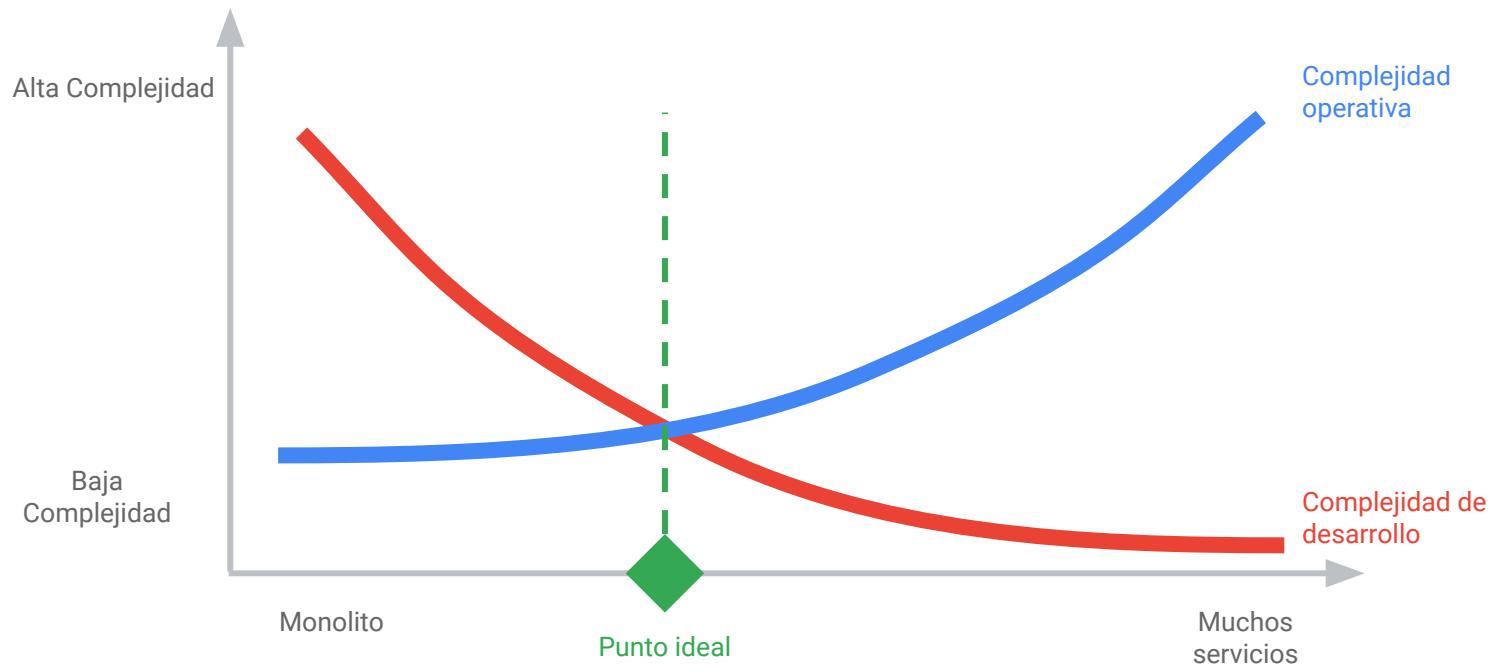


Estado actual

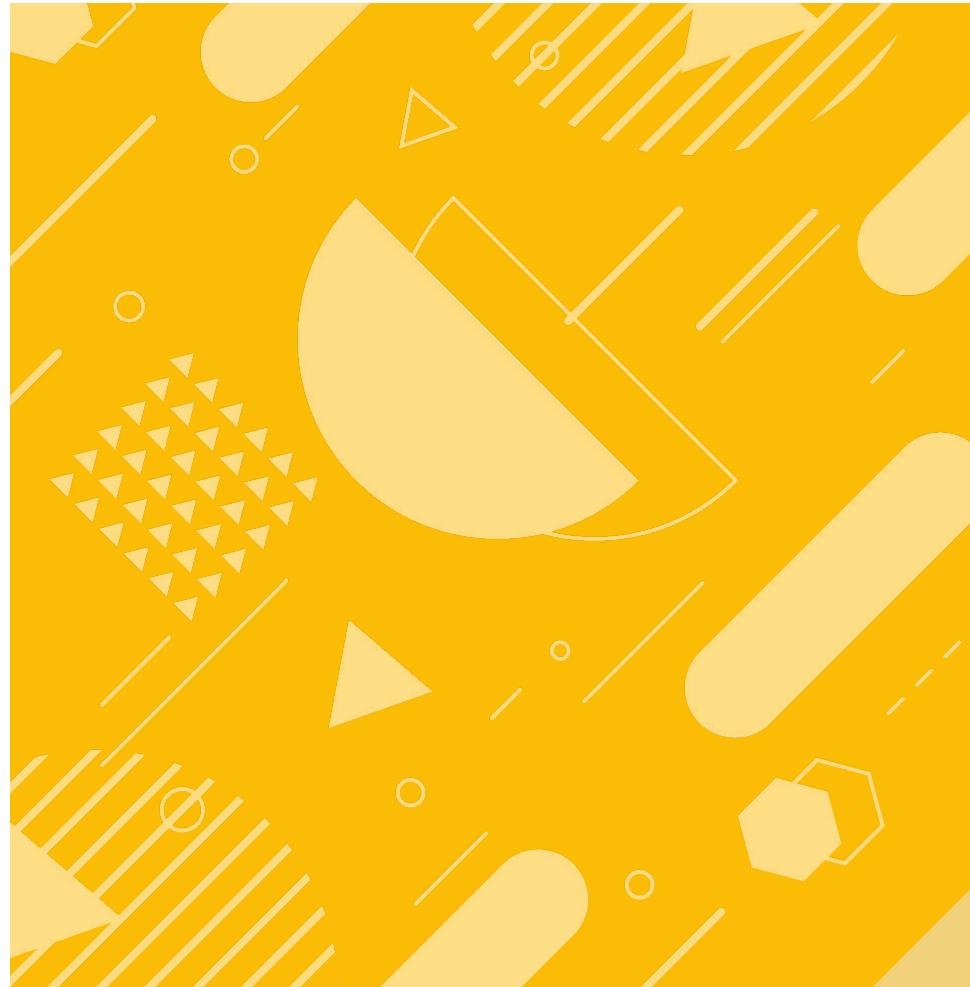
- Las funcionalidades del Kernel desarrolladas por Google han sido incorporadas con el kernel
- Nuevas funcionalidades han sido incorporadas con el paraguas de LinuxContainers.org
- La iniciativa de los contenedores abiertos ha sido formada para estandarizar los formatos de contenedores:
 - La especificación del runtime (runtime-spec)
 - La especificación de las imágenes (image-spec)
- Docker liberó como OSS el motor de ejecución (containerd)
- Otros runtimes fueron desarrollados (e.j. rkt)



Relación desarrollo/operaciones tradicional



Introducción a Kubernetes y GKE



Kubernetes (K8s)

Griego para “*Timonel*”; también la raíz para las palabras “gobernador” y “cibernético”

- Administra clusters de contenedores
- Inspirado e informado por la experiencia y los sistemas internos de Google
- Con soporte para despliegues multi-nube y sobre ambientes convencionales de virtualización y bare metal
- Soporta múltiples runtimes de contenedores
- **100% open source**, escrito en Go



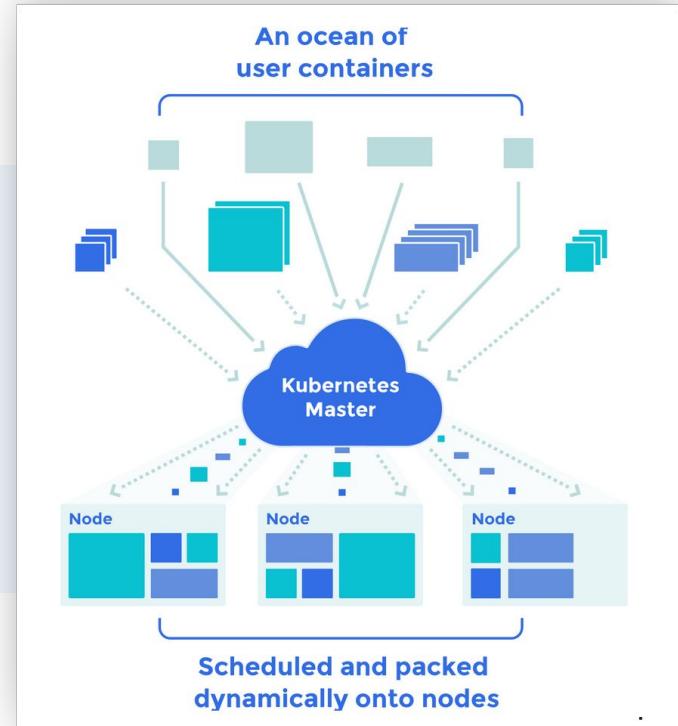
Administrar aplicaciones, no máquinas

Kubernetes (K8s)

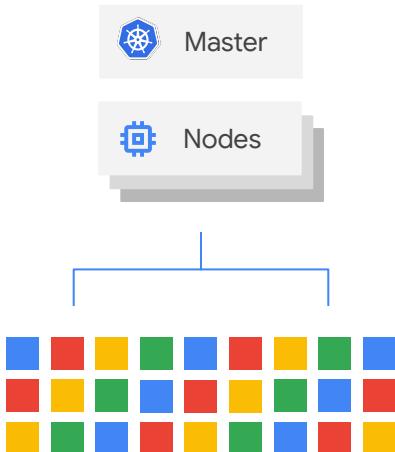
Kubernetes es el controlador para tu flotilla de cómputo

Brinda grandes funcionalidades para tus cargas de trabajo, como:

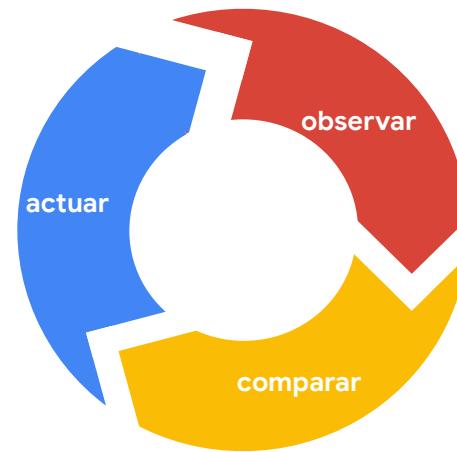
- Calendarización
- Encontrar el huésped adecuado
- Monitorear la salud
- Escalamiento elástico bajo demanda
- Movimiento bajo demanda



Piensa en Kubernetes como:

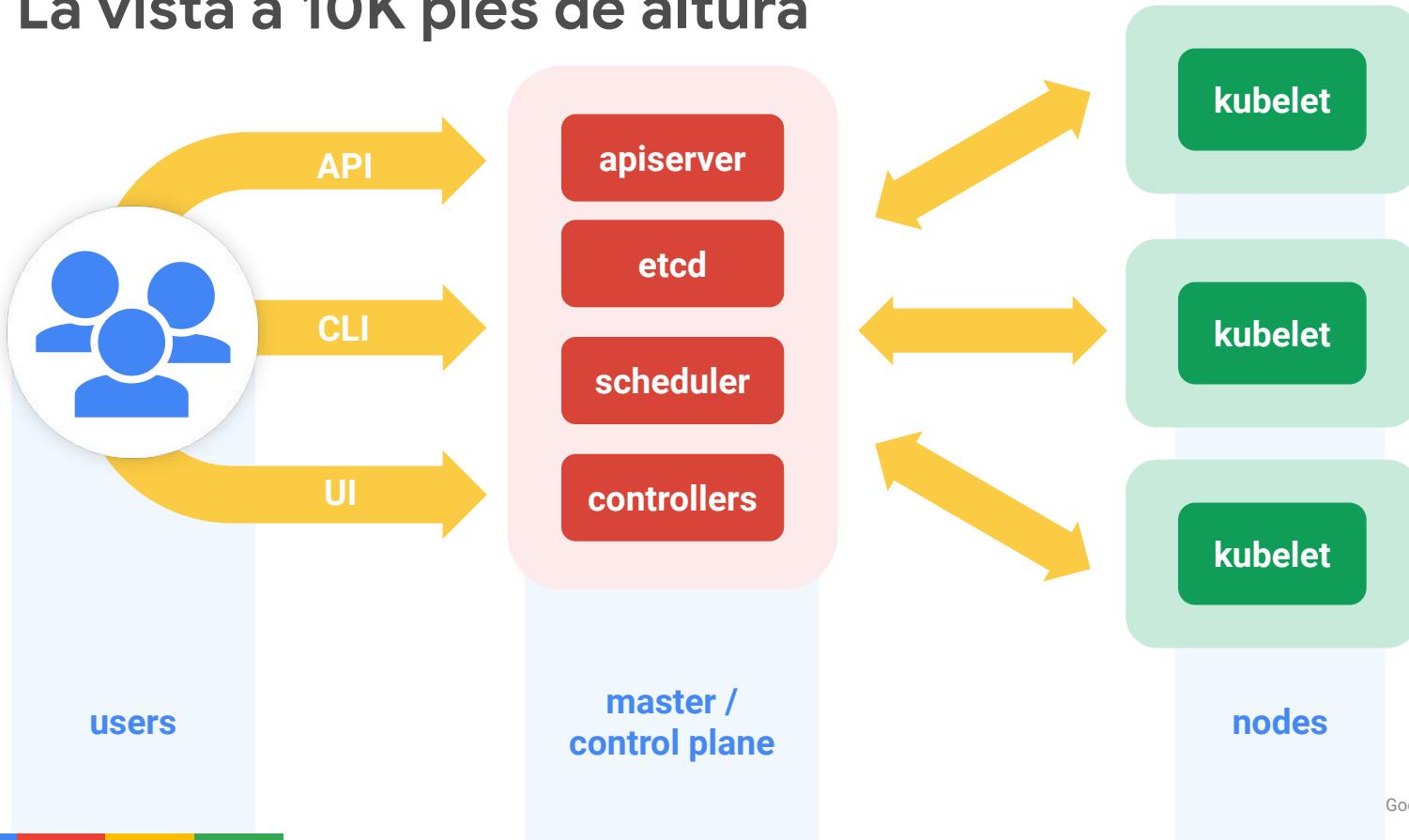


Abstracción sobre la infraestructura



Controladores declarativos

La vista a 10K pies de altura



Conceptos clave de Kubernetes



La vista a 10K pies de altura

Control Plane

- API Server: El frontend del control plane
- Fuente única de la verdad (etcd)
- Revisar por nuevos pods y asignarlos a un nodo (kube-scheduler)

Controladores

- Responden a cambios en los objetos
- Mover estado actual a estado deseado

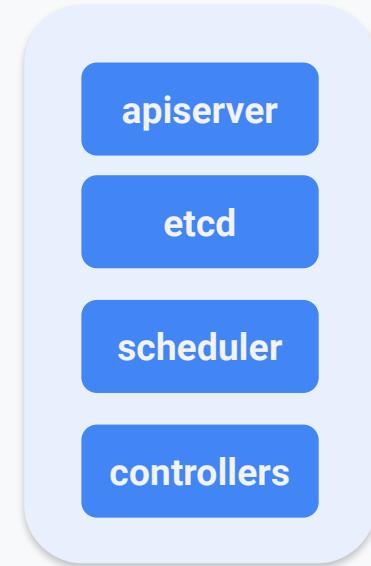
Todo es un recurso (objeto)

- kind (tipo)
- apiVersion
- metadata
- spec ← representación del estado deseado
- status ← representación del estado actual

```
kind: Deployment
apiVersion: extensions/v1beta1
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  replicas: 1
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
```

Componentes del Master

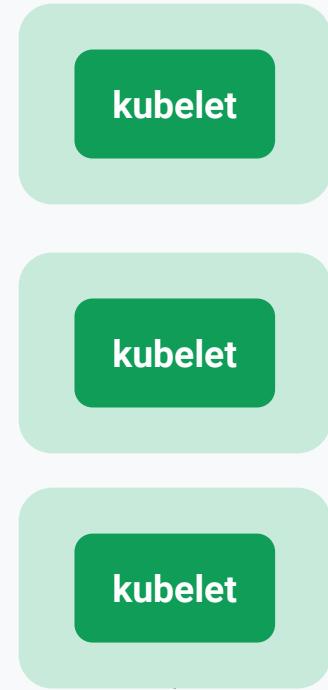
- ▶ API server (kube-apiserver)
- ▶ etcd — key-value store distribuida y confiable
- ▶ Calendarizador (kube-scheduler)
- ▶ Controladores
 - Kube controller (kube-controller-manager)
 - Controlador de Replicación
 - Controlador de endpoints
 - Controlador de tokens y cuentas de servicio
 - Cloud controller (cloud-controller-manager)
- ▶ Add-ons
 - Kube DNS (kube-dns)
 - Web UI (dashboard)
 - Monitoreo
 - Logueo en todo el cluster



master / control plane

Componentes de los Nodos

- ▶ Kubelet (kubelet)
- ▶ Kube proxy (kube-proxy)
- ▶ Container runtime
 - Docker (containerd)
 - Rkt
- ▶ Monitoring/Logging
 - Fluentd



Los clientes usan la línea de comandos (CLI) de Kube Control (`kubectl`) para interactuar con el clúster

<https://kubernetes.io/docs/concepts/overview/components/>

Concepto clave: El Pod

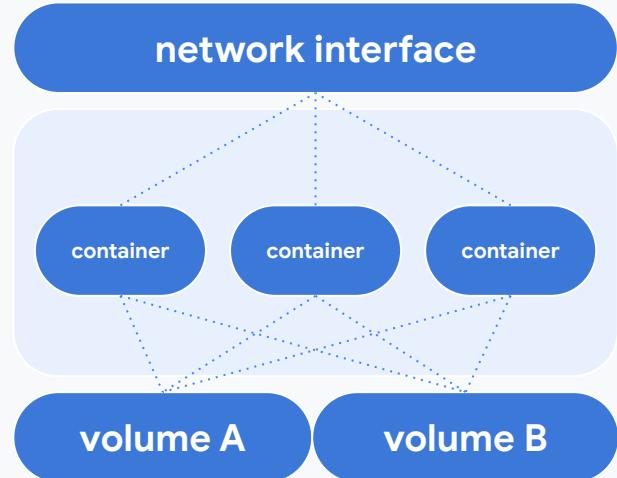
Unidad atómica de Kubernetes

Compuesto de uno o más contenedores con recursos de almacenamiento y red compartidos (e.j. servidor de apps y web server)

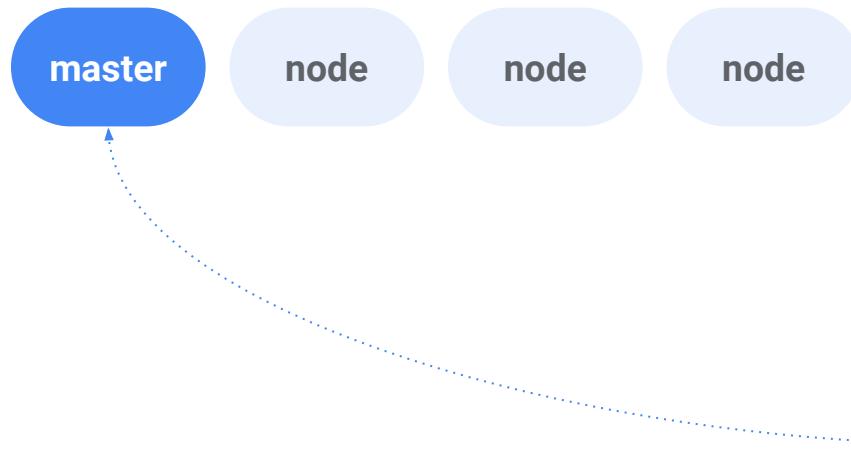
Los contenedores dentro de un pod comparten el mismo linux namespace, pero diferentes grupos de control

Kubernetes automatiza de buena manera la configuración de namespaces, cgroups, etc.

Unidad de despliegue (empaquetada)



El ciclo de vida de un Pod



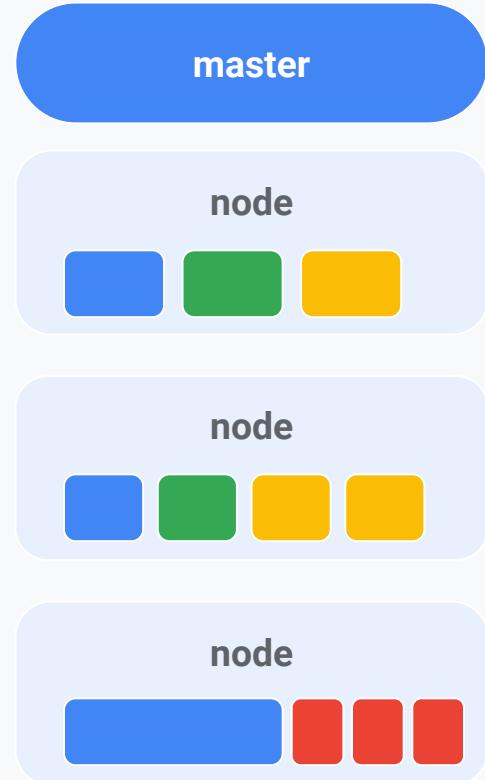
```
apiVersion: v1
kind: Pod
metadata:
  name: my-app
spec:
  containers:
    - name: my-app
      image: my-app
    - name: nginx-ssl
      image: nginx
  ports:
    - containerPort: 80
    - containerPort: 443
```

Concepto clave: el deployment

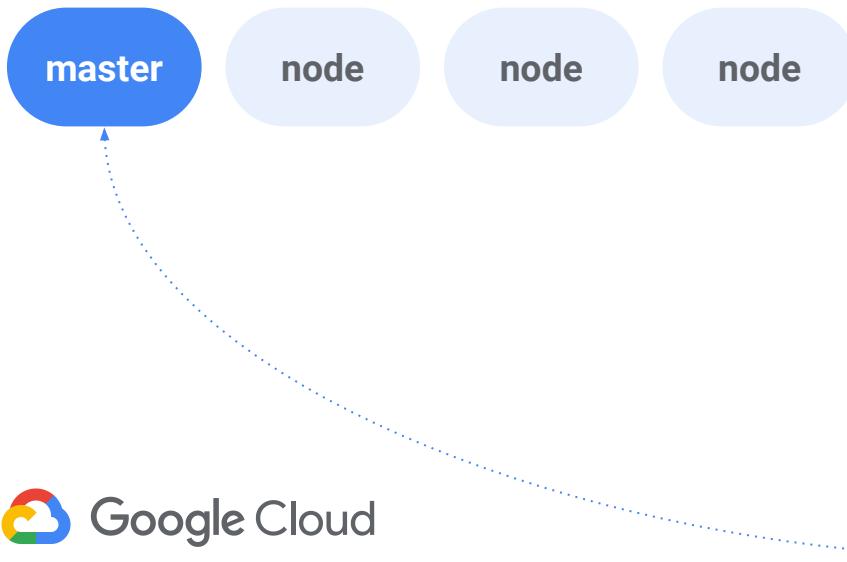
Es una abstracción que permite actualizar el número deseado de réplicas de un pod.

Los pods son mortales, pero las abstracciones como el deployment nos dan resiliencia.

Es una de muchas abstracciones que controlan cómo los pods son calendarizados y desplegados.

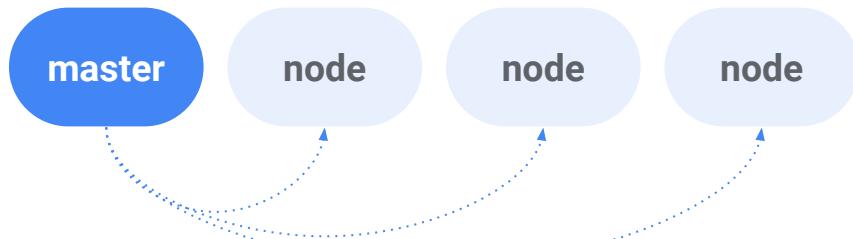


El ciclo de vida de un Pod



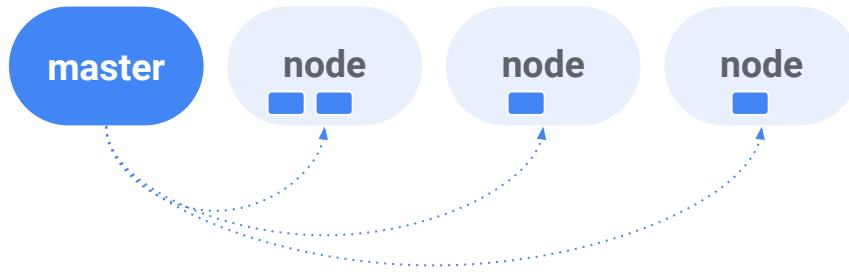
```
kind: Deployment
apiVersion: v1beta1
metadata:
  name: frontend
spec:
  replicas: 4
  selector:
    role: web
  template:
    metadata:
      name: web
      labels:
        role: web
    spec:
      containers:
      - name: my-app
        image: my-app
      - name: nginx-ssl
        image: nginx
        ports:
        - containerPort: 80
        - containerPort: 443
```

El ciclo de vida de un Pod



```
kind: Deployment
apiVersion: v1beta1
metadata:
  name: frontend
spec:
  replicas: 4
  selector:
    role: web
  template:
    metadata:
      name: web
      labels:
        role: web
    spec:
      containers:
      - name: my-app
        image: my-app
      - name: nginx-ssl
        image: nginx
        ports:
        - containerPort: 80
        - containerPort: 443
```

El ciclo de vida de un Pod



```
kind: Deployment
apiVersion: v1beta1
metadata:
  name: frontend
spec:
  replicas: 4
  selector:
    role: web
  template:
    metadata:
      name: web
      labels:
        role: web
    spec:
      containers:
      - name: my-app
        image: my-app
      - name: nginx-ssl
        image: nginx
        ports:
        - containerPort: 80
        - containerPort: 443
```

Concepto clave: Servicios

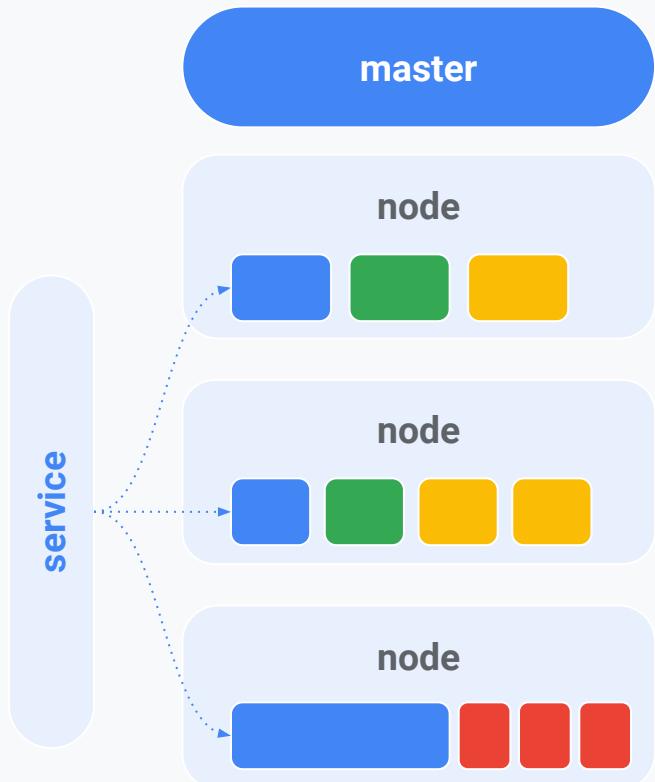
Endpoints estables para los pods

Si los pods son mortales, los servicios te dan una manera estable de accederlos

Proveen de capacidades de balanceo de carga entre múltiples pods

Con servicios, puedes comunicarte entre pods vía IPs externas, IPs internas al clúster o por DNS

Los servicios pueden apuntar a múltiples pods con el mismos metadatos (kv-pairs), conocidos como selectores de etiquetas



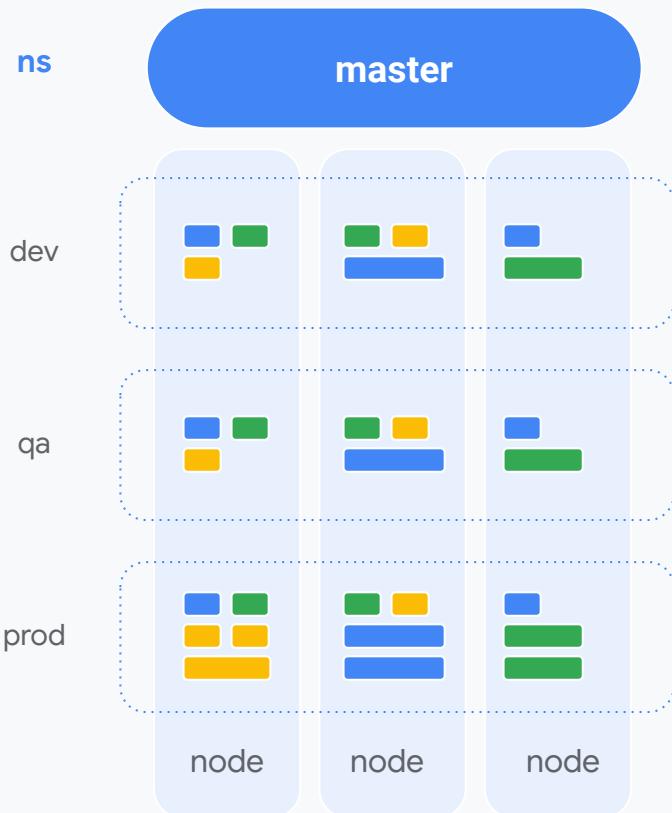
Concepto clave: Namespaces

Aislamiento lógico entre objetos de kubernetes
(incluyendo pods, deployments, etc.)

La mayoría de los recursos están acotados a un namespace, pero hay algunos componentes fuera del alcance de un namespace (e.j. Los nodos)

Pueden ser utilizados para acceso controlado por roles (RBAC)

Útiles para aislar ambientes con un único cluster entre múltiples equipos (multi-tenancy)



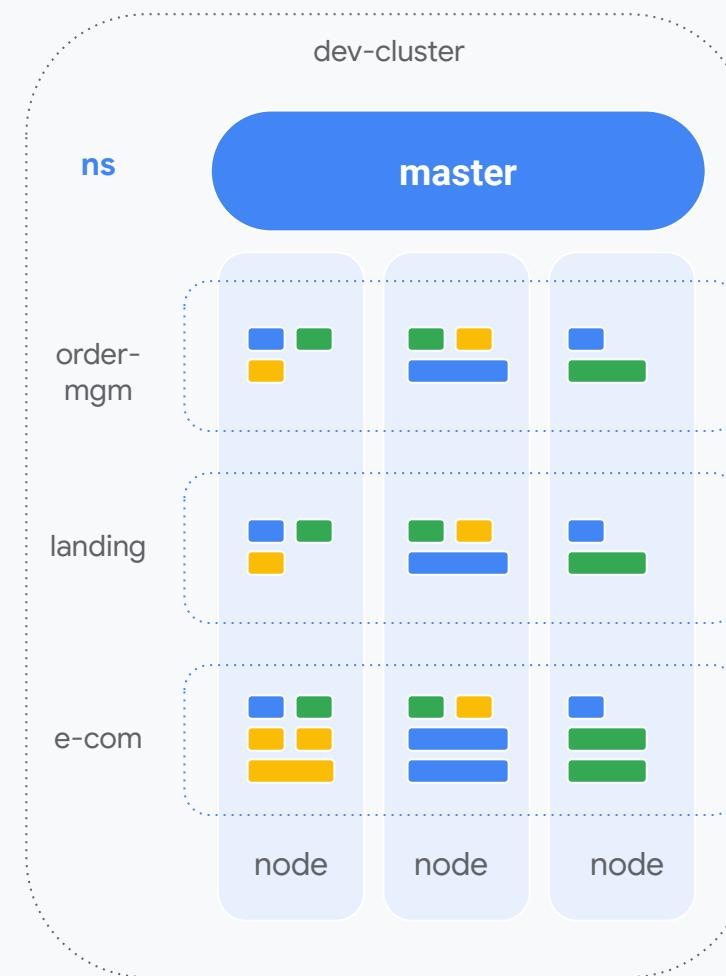
Concepto clave: Namespaces

Aislamiento lógico entre objetos de kubernetes
(incluyendo pods, deployments, etc.)

La mayoría de los recursos están acotados a un namespace, pero hay algunos componentes fuera del alcance de un namespace (e.j. Los nodos)

Pueden ser utilizados para acceso controlado por roles (RBAC)

Útiles para aislar ambientes con un único cluster entre múltiples equipos (multi-tenancy)

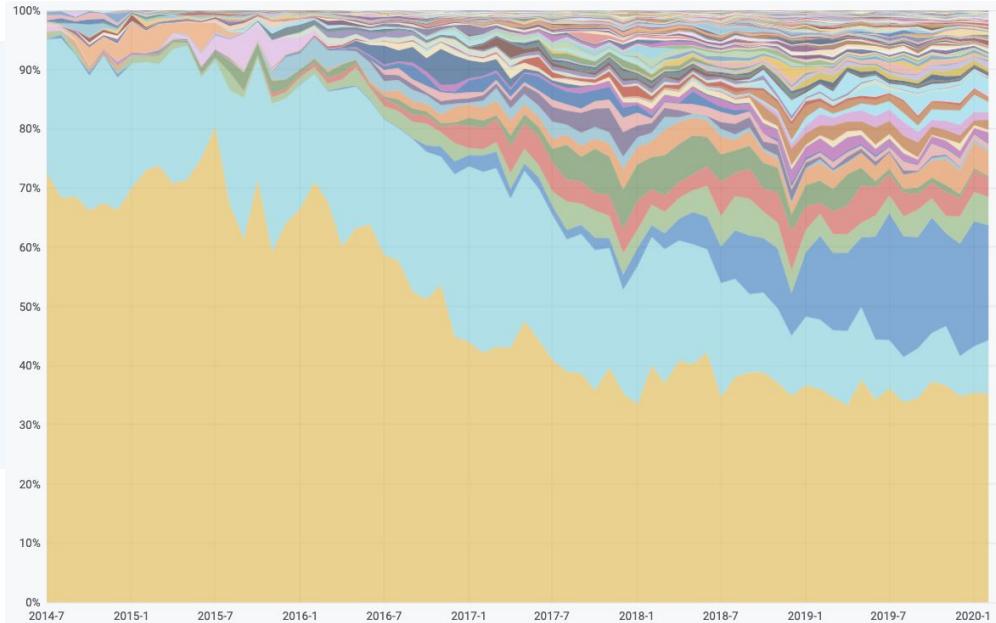


El compromiso de Google con la Comunidad

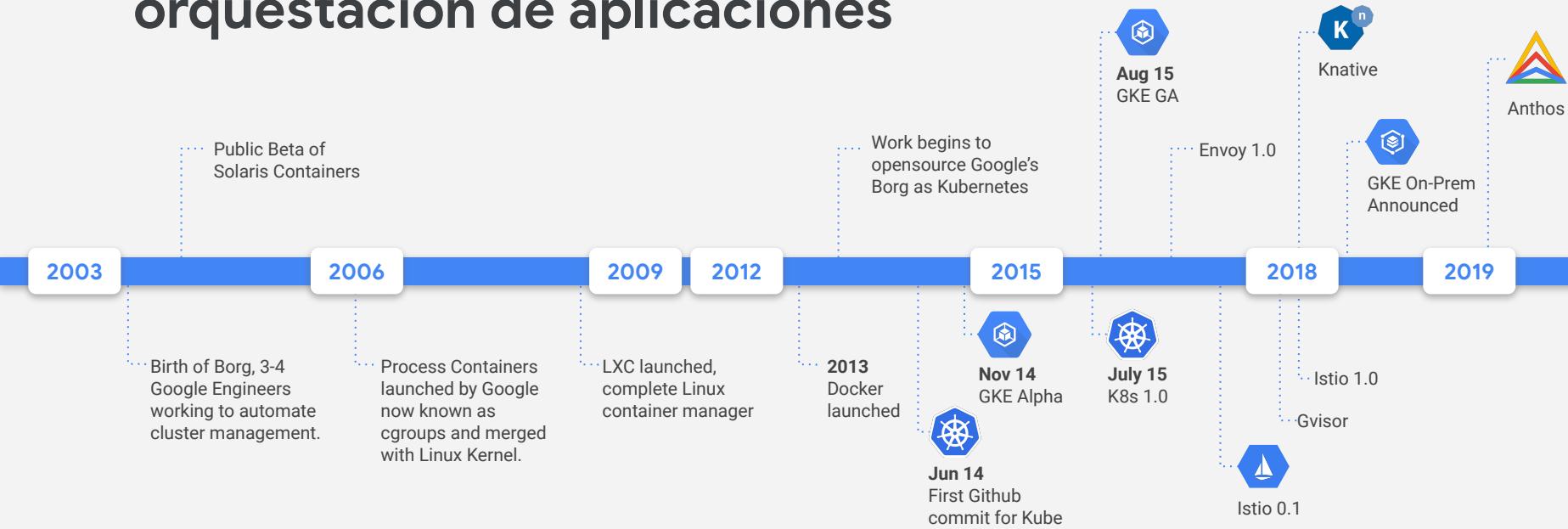
Desde el inicio, Google se ha mantenido como el mayor contribuidor a Kubernetes en cuanto a code commits, pull requests, issues filed, etc.

Actualmente seguido por VMware y Red Hat.

Kubernetes contributions per company (2014-7 – 2020-01)



15+ años de experiencia en la orquestación de aplicaciones



Introducción al cómputo híbrido y multi-nube



Visión de Google para Aplicaciones Modernas

El futuro de las aplicaciones y la infraestructura en la que se ejecutan, se crea con **microservicios** en **contenedores**, administrados a través de un **sistema declarativo** con una **experiencia de control única** que utiliza una **malla de servicios** para **abrir todas las ubicaciones de las aplicaciones**.

¿Por qué construimos Kubernetes?

Construido como el sistema interno
para **maximizar la productividad** de
nuestros desarrolladores de software.

Lo hicimos open source para **evitar
crear nuestra propia isla tecnológica**.



¿Cómo se ve el rendimiento máximo?



208 veces más

Despliegues de código frecuentes



106 veces más rápido

Tiempo de commit a deploy



2,604 veces más rápido

Tiempo de recuperación ante incidentes



7 veces menos

Tasa de cambios con fallos

Lección aprendida: Kubernetes estaba incompleto

Hace dos años, comenzamos a liberar como OSS una serie de frameworks encima de Kubernetes, que son equivalentes a sistemas internos de Google



Istio



Knative



OPA



Tekton



Kubeflow

...

“Out-of-the-Box”
FW, LB and App Security

“Out-of-the-Box”
Application Autoscaling

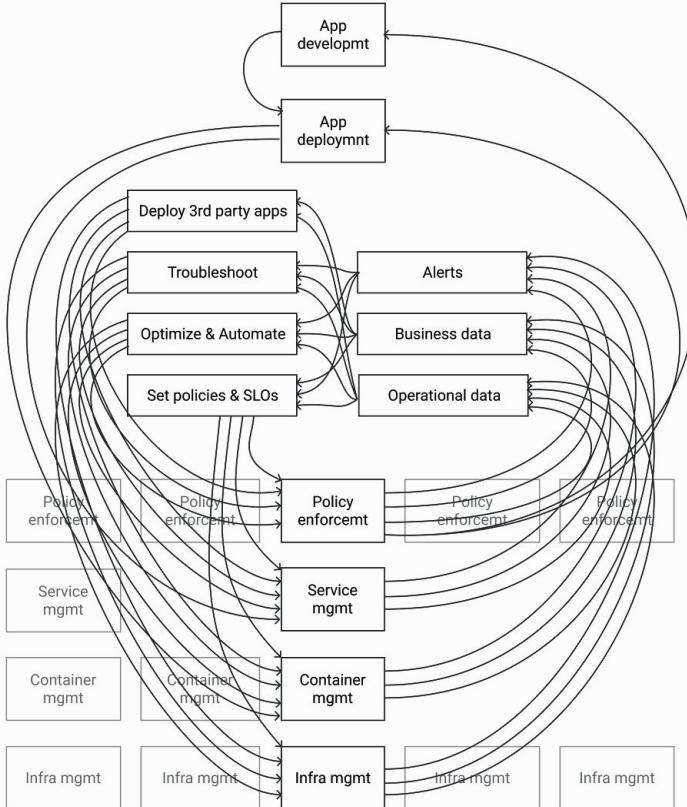
“Out-of-the-Box”
Policy Management

Run Anywhere
CI/CD

Run Anywhere
AI/ML

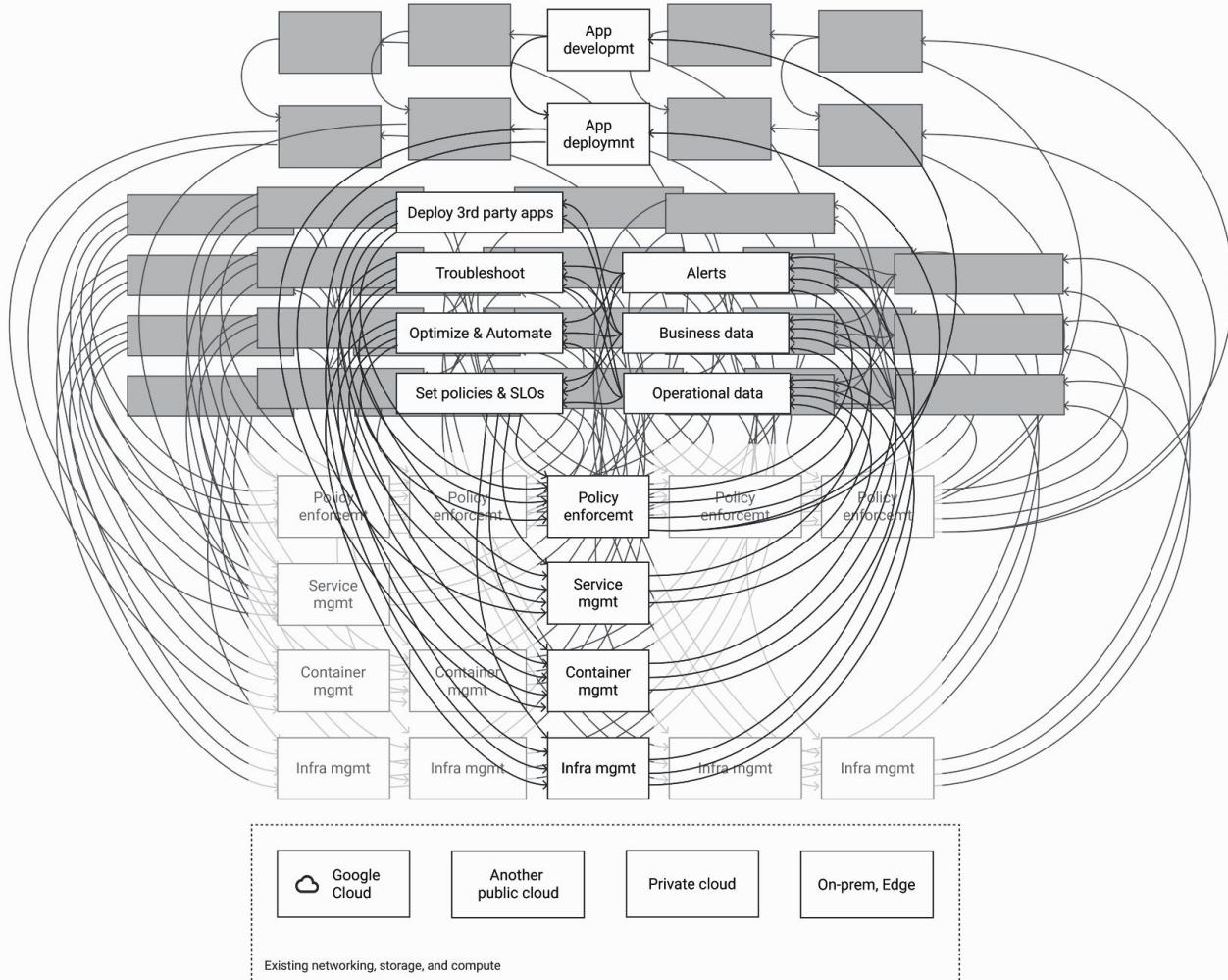
Un poco complicado...

Una plataforma de K8s del estado del arte el día de hoy tiene una distribución base con muchas de estas frameworks abiertas



Demasiado complicado...

Lograr que todos estos frameworks se ejecuten al mismo tiempo, con múltiples versiones, entre AWS, Azure, GCP, on-premise es demasiado complicado



¿Qué es Anthos?

Anthos es una **plataforma administrada** que extiende los **servicios y las prácticas de ingeniería** de Google Cloud en tus **ambientes** para que puedas **modernizar aplicaciones más rápido** y establecer **consistencia operacional** entre ellas.

¿Qué es Anthos?

Anthos es una **plataforma administrada** que extiende los **servicios y las prácticas de ingeniería** de Google Cloud en tus **ambientes** para que puedas **modernizar aplicaciones más rápido** y establecer **consistencia operacional** entre ellas.

Modelo SaaS
--updates a cargo de Google

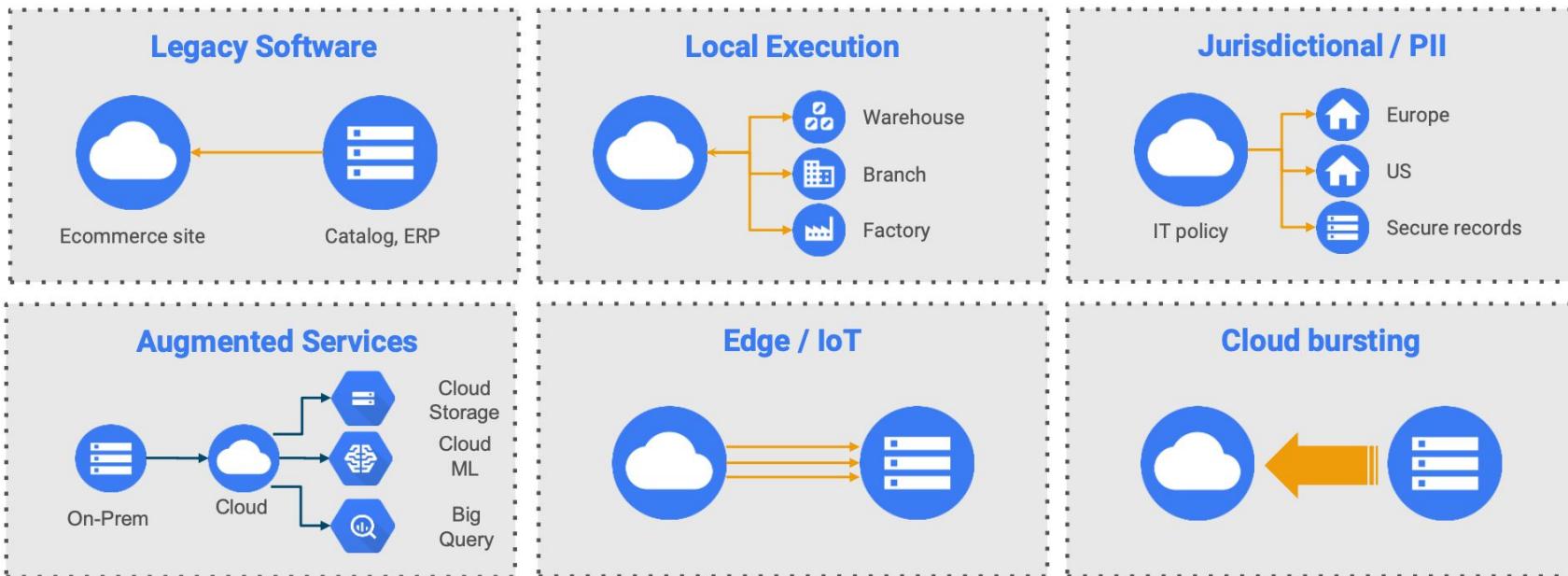
Tecnologías integradas (como Istio, KNative), servicios cloud y principios de SRE

Y políticas declarativas que aplican estándares de seguridad en todos los ambientes

Con herramientas para realizar migración de VMs a contenedores

Consistencia, seguridad, y cómputo eficiente para escenarios híbridos y multi-nube

¿Por qué las organizaciones eligen contextos híbridos?



Verdadera plataforma híbrida/multi-nube

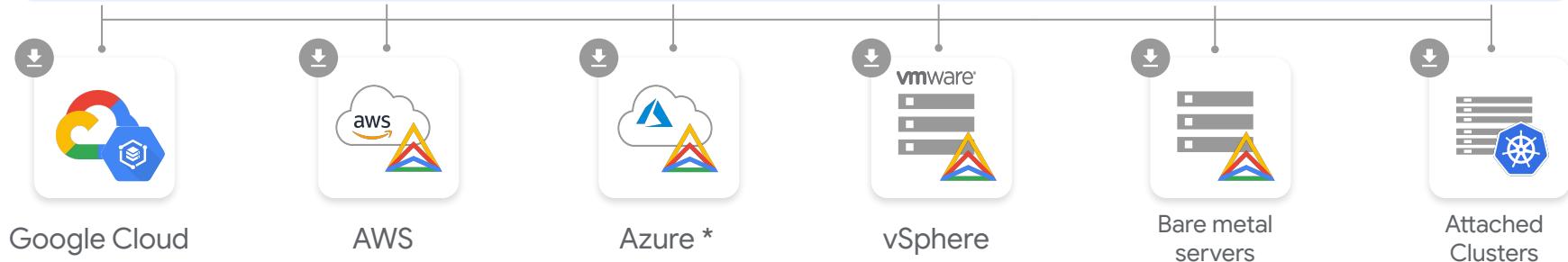
Gestión de políticas y configuración
Anthos Config Management

Administración de operaciones
Cloud Logging, Cloud Monitoring, Cloud Trace

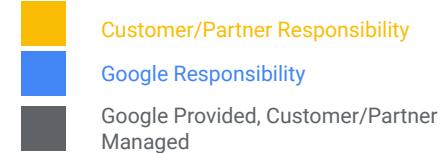
Administración de aplicaciones y servicios
Anthos Service Mesh, Cloud Run, Cloud Build, kf

Administración de infraestructura
Anthos Multi-cloud API, Anthos Hybrid Networking

Administración de Seguridad
Anthos Identity Services
Binary Authorization
GKE Sandbox



Anthos Shared Responsibility Model



	Anthos on GCP	Anthos On AWS, vSphere, Bare Metal	Anthos Attached Clusters	Anthos Private Mode
Cargas de trabajo del Cliente	Completamente bajo el control del cliente; sin soporte de GCP	Completamente bajo el control del cliente; sin soporte de GCP	Completamente bajo el control del cliente; sin soporte de GCP	Completamente bajo el control del cliente; sin soporte de GCP
Servicio de Administración: Única vista de visibilidad	Consola GCP	Consola GCP	Consola GCP	Sin conectividad a la consola de GCP; Manejado por el cliente vía el centro de administración de Anthos
Componentes y servicios de Anthos	Soporte (break/fix) para los componentes del cluster Servicios hospedados en GCP tienen SLAs	SW provisto por Google, Administrado por el Cliente. GCP provee soporte (break/fix)	SW provisto por Google, Administrado por el Cliente. GCP provee soporte (break/fix)	SW provisto por Google, Administrado por el Cliente. GCP provee soporte (break/fix)
Distribución de Kubernetes	Totalmente administrado con SLAs de Google	SW provisto por Google, Administrado por el Cliente. GCP provee soporte (break/fix)	Administrado por el cliente Sin soporte de GCP	SW provisto por Google, Administrado por el Cliente. GCP provee soporte (break/fix)
VMs y otra infraestructura	Totalmente administrado con SLAs de Google	Administrado por el cliente Sin soporte de GCP	Administrado por el cliente Sin soporte de GCP	Administrado por el cliente Sin soporte de GCP

Estamos innovando para darte simplicidad a escala

01

A escala on Git-Ops

Administra tu propia infraestructura usando funciones declarativas.

02

Consistencia con KRM

Apalanza nuestro soporte profundo del modelo de recursos de Kubernetes para construir y manejar **infraestructura, apps y servicios administrados.**

03

Cloud en todos lados

Servicios de Google de cómputo, desarrollo, datos, analítica, AI, y muchos más disponibles en múltiples ambientes.

04

Flexibilidad de consumo

En nube pública, privada, cloud, hypervisor o en bare metal, subscripción o pago-as-you go. La nube bajo tus términos.

Kubernetes, la forma fácil

Crea un cluster con un solo click.

Visualiza tus clusters y cargas de trabajo en un único plano de visión.

Google mantiene tu cluster listo y disponible.

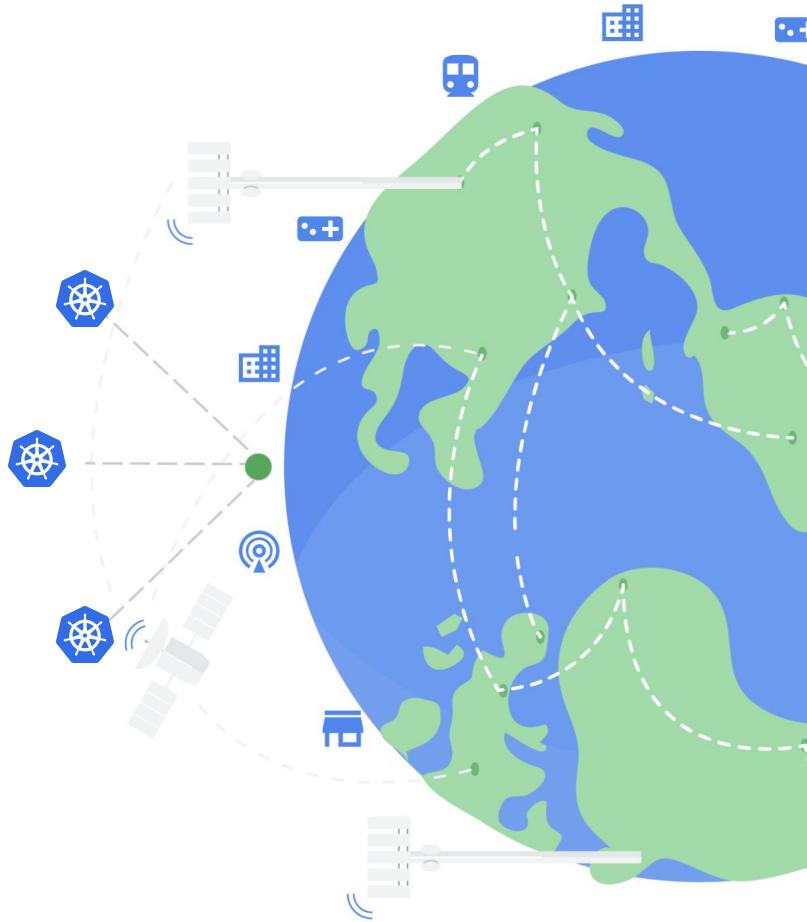


The screenshot shows the Google Cloud Platform K8S Garage interface. At the top, there's a navigation bar with the Google Cloud logo, the text "Google Cloud Platform", a "K8S Garage" dropdown, and a search icon. Below the navigation bar, there's a sidebar on the left with icons for "Kubernetes Engine", "Kubernetes clusters", "Workloads", "Discovery & load balancing", "Configuration", and "Storage". The main area is titled "Create a Kubernetes cluster". It includes fields for "Name" (set to "cluster-1"), "Description (Optional)", "Location" (set to "Zonal"), "Zone" (set to "us-central1-a"), "Cluster Version" (set to "1.8.7-gke.1 (default)"), and "Machine type" (set to "1 vCPU"). A "Cloud Launcher" button is visible at the bottom.

Google Kubernetes Engine

Google Kubernetes Engine (GKE) ofrece la mejor experiencia de Kubernetes del planeta

- Escalabilidad récord de nodos de 15K para un solo clúster
- El escalado automático, las reparaciones y las actualizaciones automáticas impulsados por IA mantienen los clústeres en buen estado y con una gran capacidad de respuesta a los picos masivos de recursos
- Integración de GPU / TPU de clase mundial
- Docenas de funciones de seguridad únicas (por ejemplo, eBPF)



Google Kubernetes Engine

Auto-repair

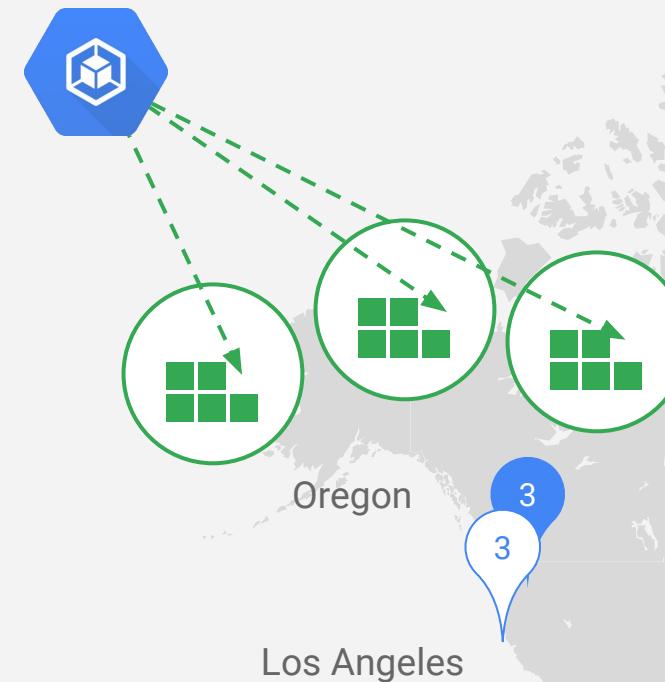
Iniciar automáticamente el proceso de reparación para los nodos que no superan una verificación de estado.

Auto-upgrade

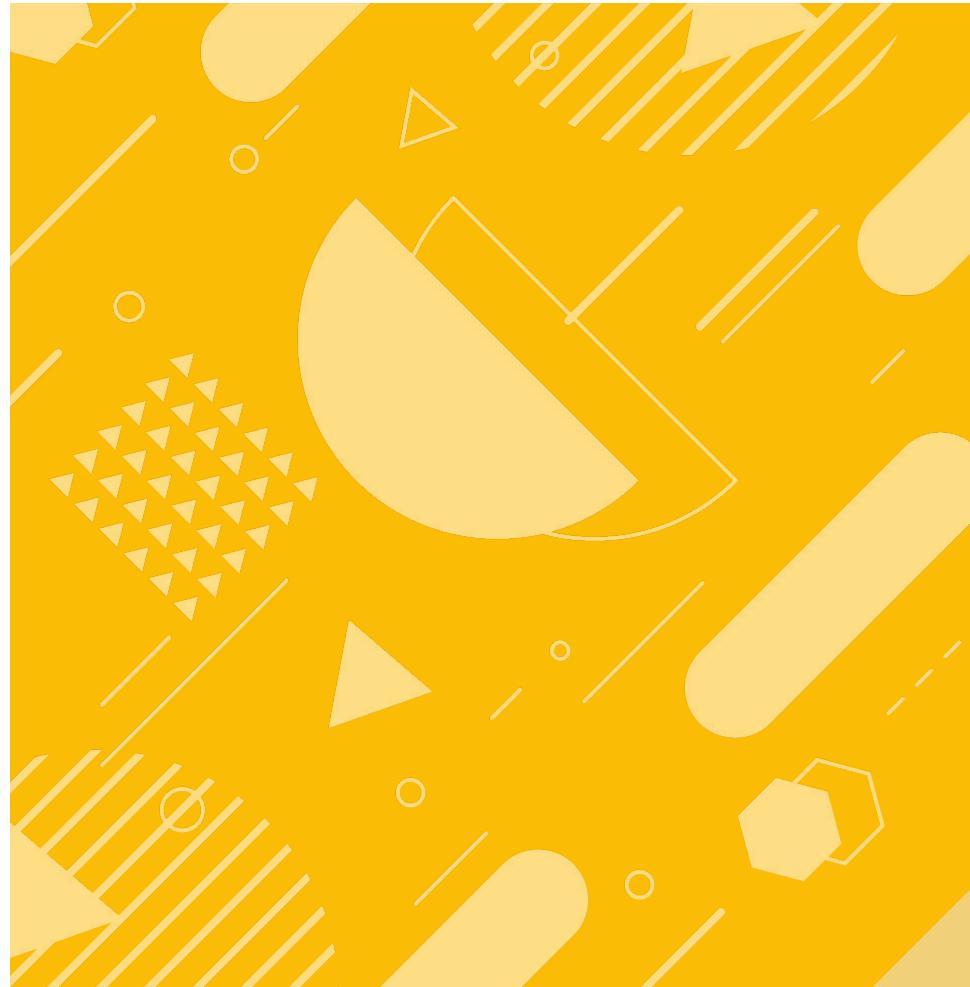
Mantenga el plano de control y los nodos del clúster actualizados con la última versión estable.

Auto-scale

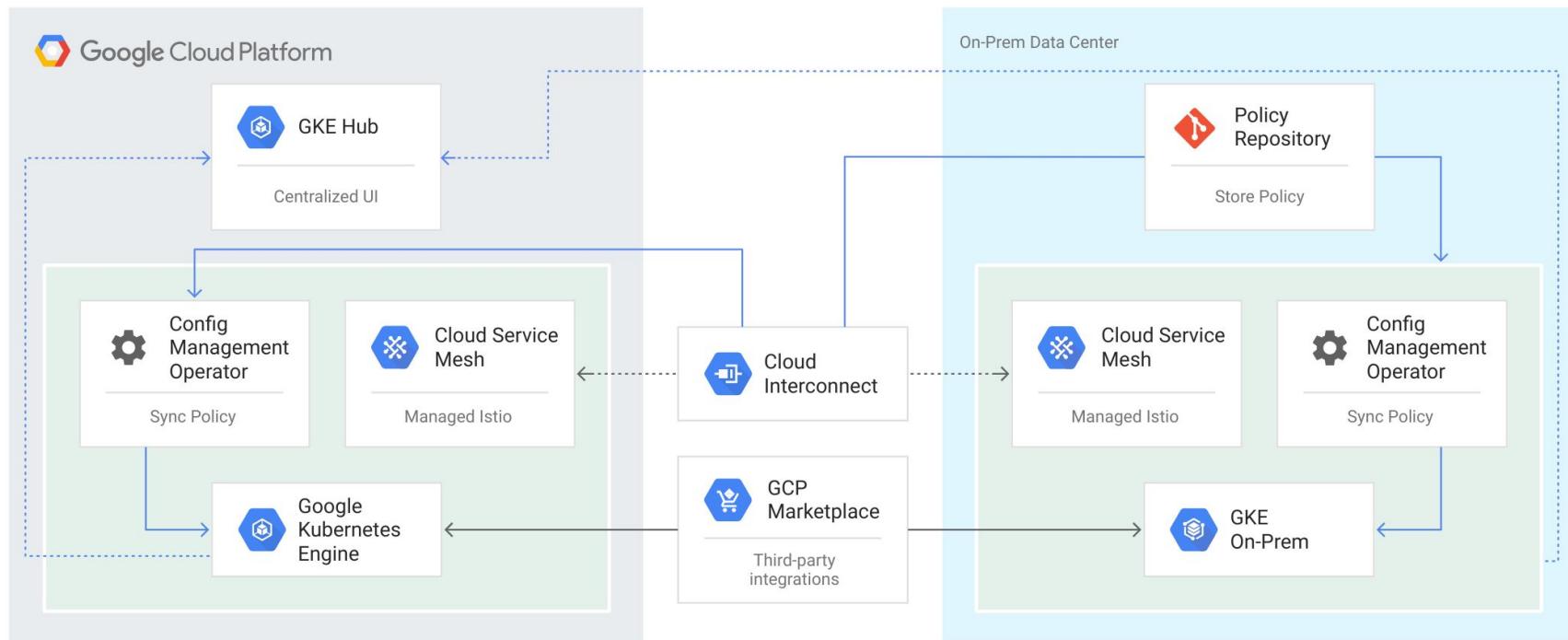
El ajuste de escala automático del clúster maneja la demanda creciente y se reduce según sea necesario.



Arquitectura de un despliegue híbrido y multi-nube



Anthos Typical Enterprise Interaction



- Policy Configuration Flow
- Cross Environment Service Traffic
- GKE Connect Agent Control Plane
- Third-p

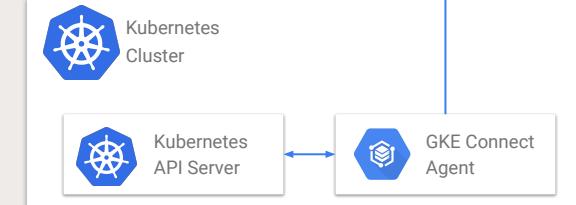
Conectando tu cluster a Google

- Registra un cluster con GCP e instala el GKE Connect Agent en tu cluster
- No se requiere de una IP pública
- Traversa VPNs, NATs, Firewalls y Proxies

Google Cloud Platform

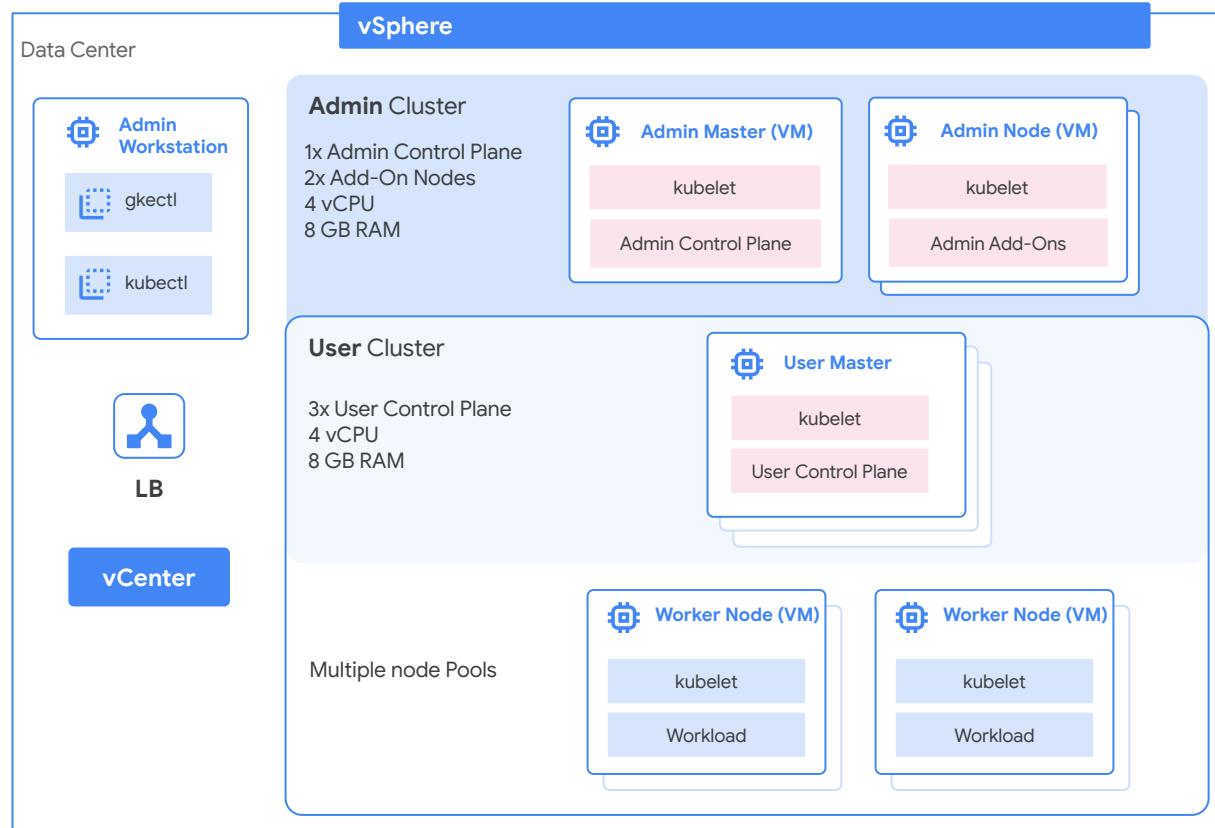


Infrastructure



Anthos on VMware High Level Architecture

Jump host
“gkeadm”



Admin Cluster

Maneja el ciclo de vida de las operaciones

Monitorea la salud del plano de control de los clusters de usuario

User Cluster

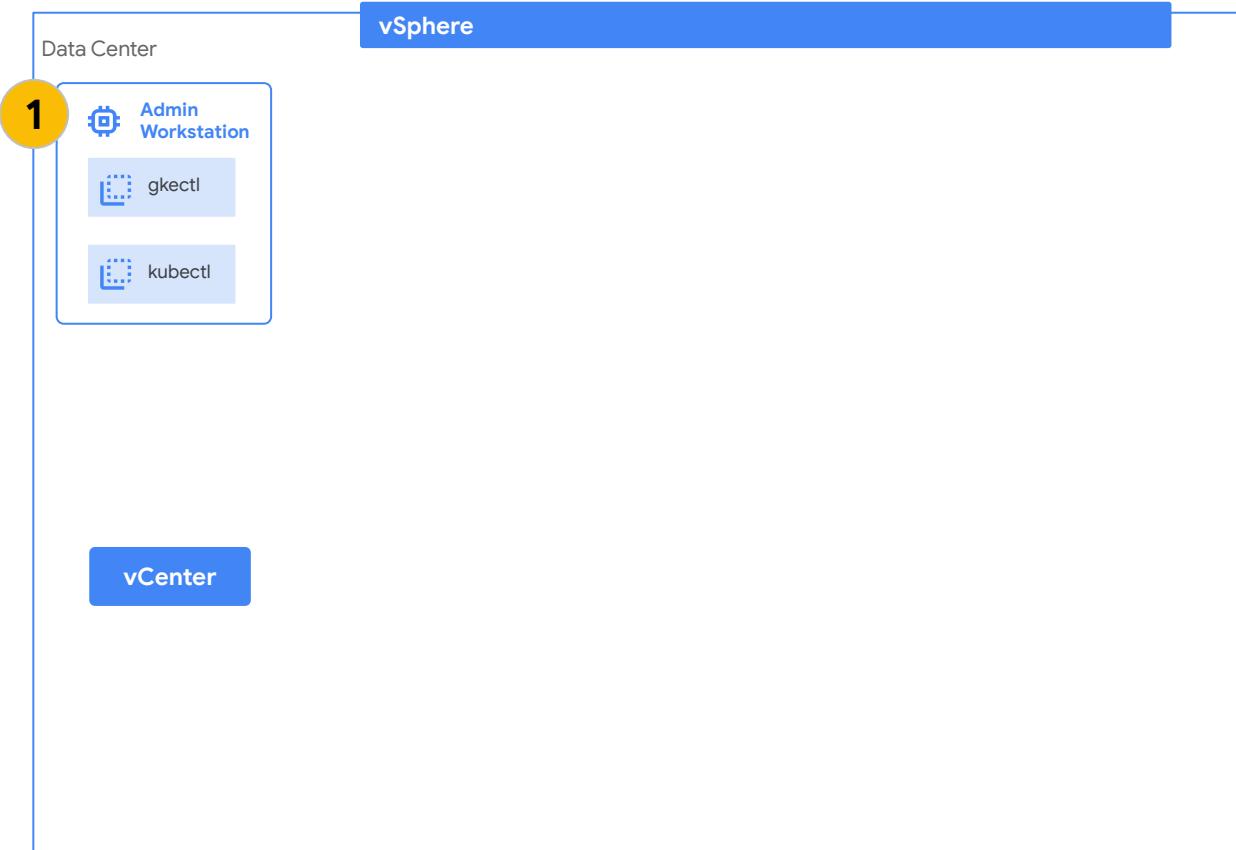
Ejecuta cargas de trabajo de usuario

Puede variar en tamaño y configuración

Anthos on VMware Arch Overview (1)

1 Admin Workstation

- Workstation de Linux
- Disponible como OVA (open virtualization appliance)
- Contiene herramientas para realizar la instalación, escalamiento y actualización

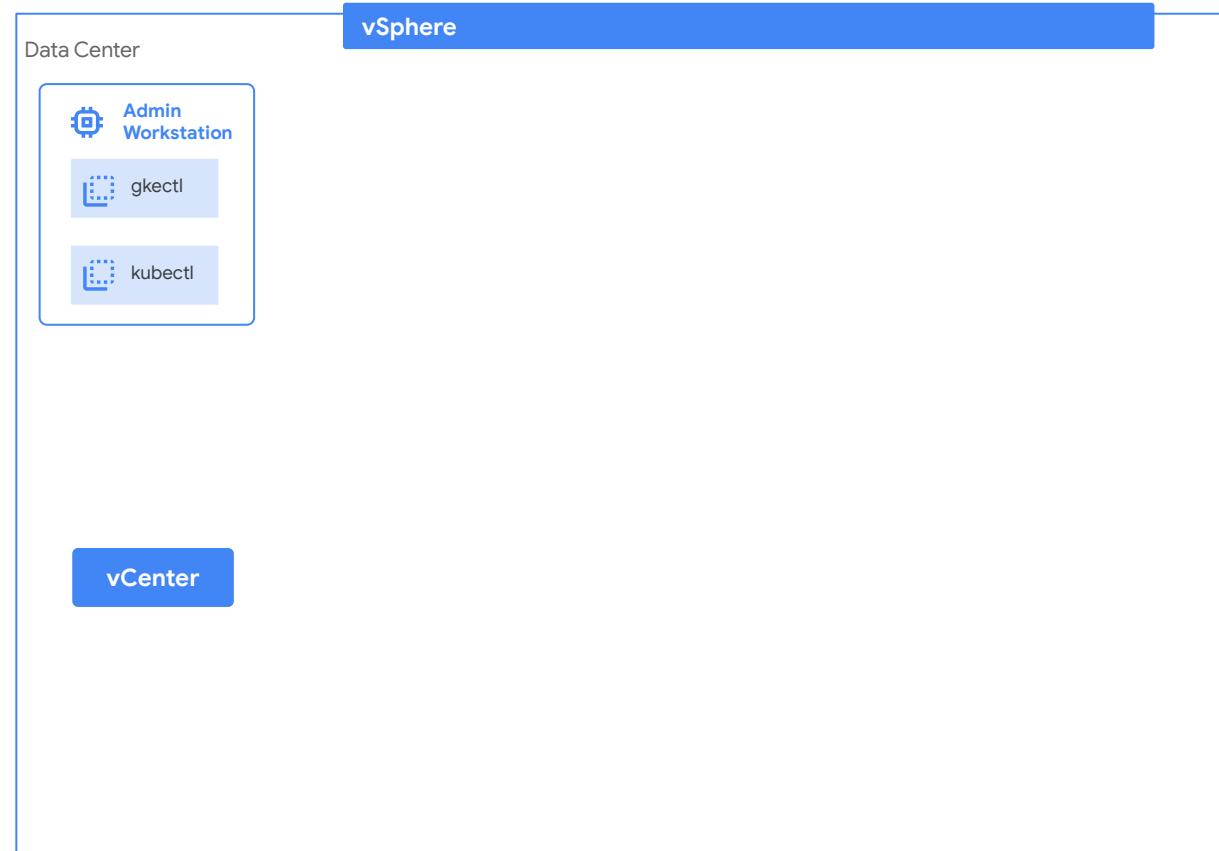


Anthos on VMware Arch Overview (2)

2

Jump host

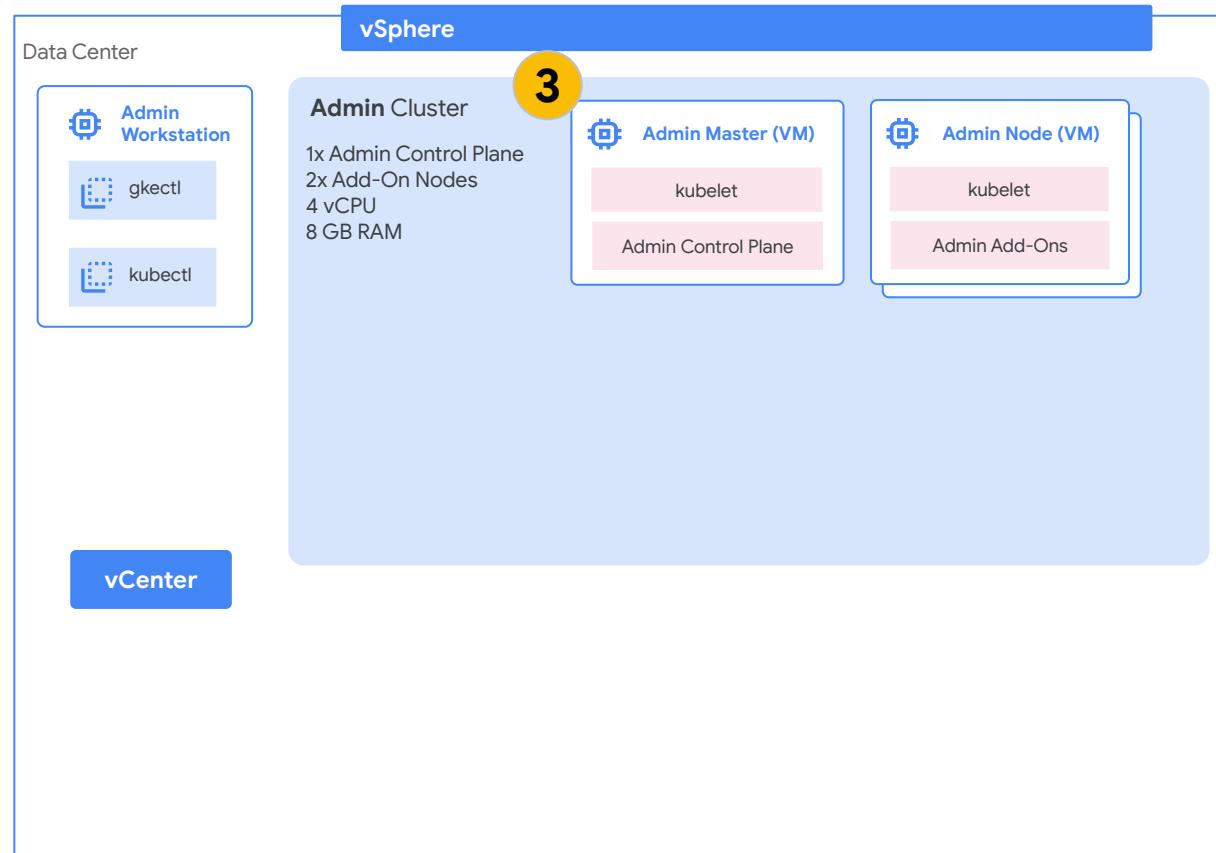
- Puede ser la “laptop” o el “desktop” de un usuario
- Administra el ciclo de vida del admin workstation utilizando la herramienta “gkeadm” (CLI)
- Soporta Linux, Windows y pronto Mac



Anthos on VMware Arch Overview (3)

3 Admin Cluster

- Administra el ciclo de vida de operaciones para los clusters de usuario: install, upgrade, etc.
- Monitorea la salud del plano de control de los clusters de usuario
- Ejecuta addons de sistema como seguridad, logging, monitoring
- Instancia única por vSphere Datacenter

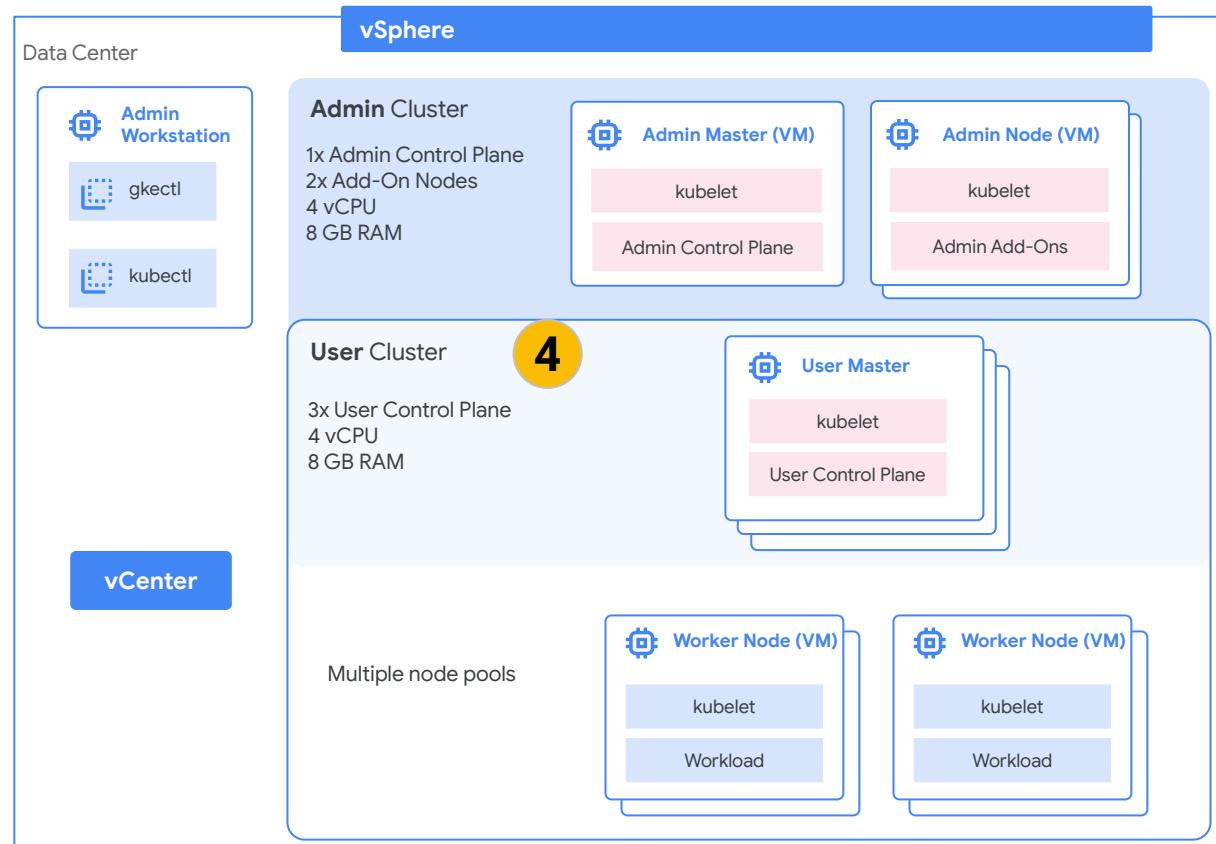


Anthos on VMware Arch Overview (4)

4

User Cluster(s)

- Ejecuta cargas de trabajo de usuario (aplicaciones)
- Puede variar en tamaño y configuración
- Puede ser desplegado en un vSphere datastore separado
- Cada admin cluster soporta 10 user clusters con 100 nodos (desde la versión 1.4)



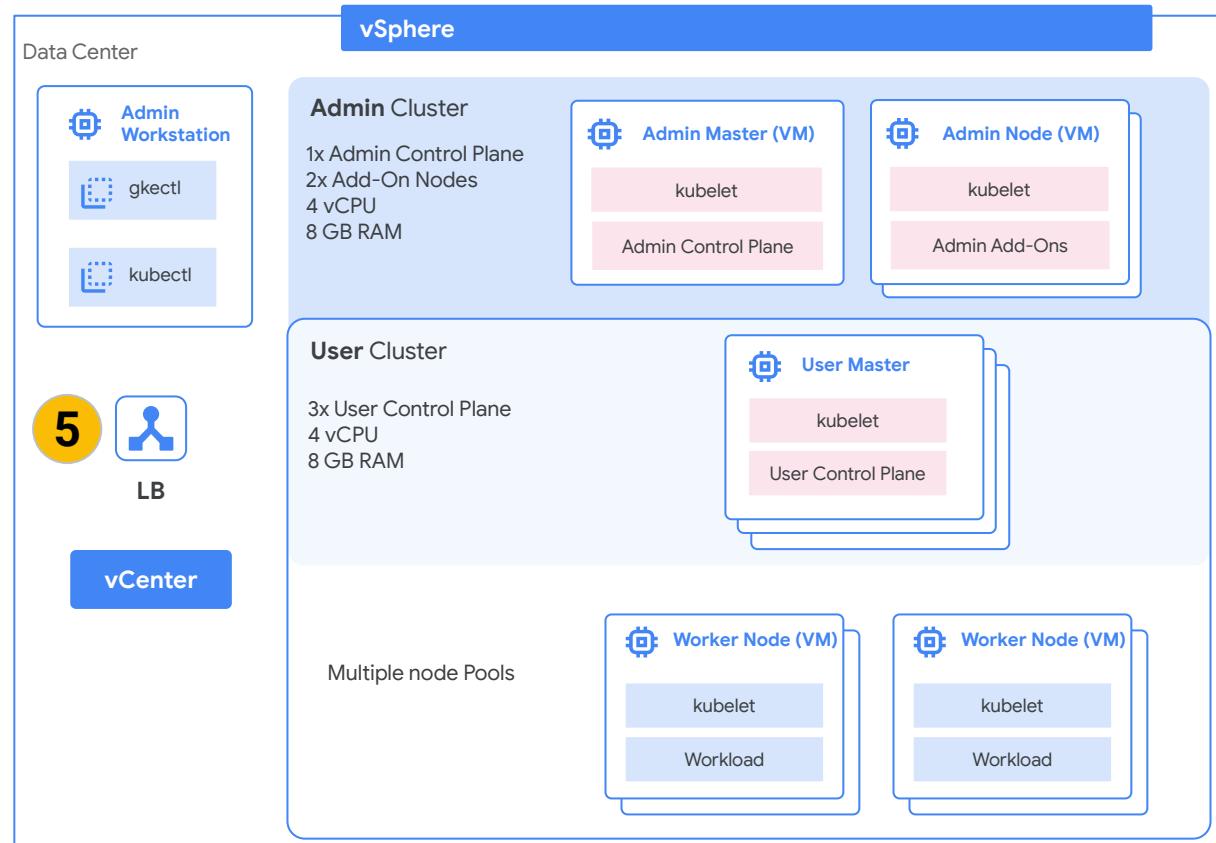
Anthos on VMware Arch Overview (5)

5 Load Balancer

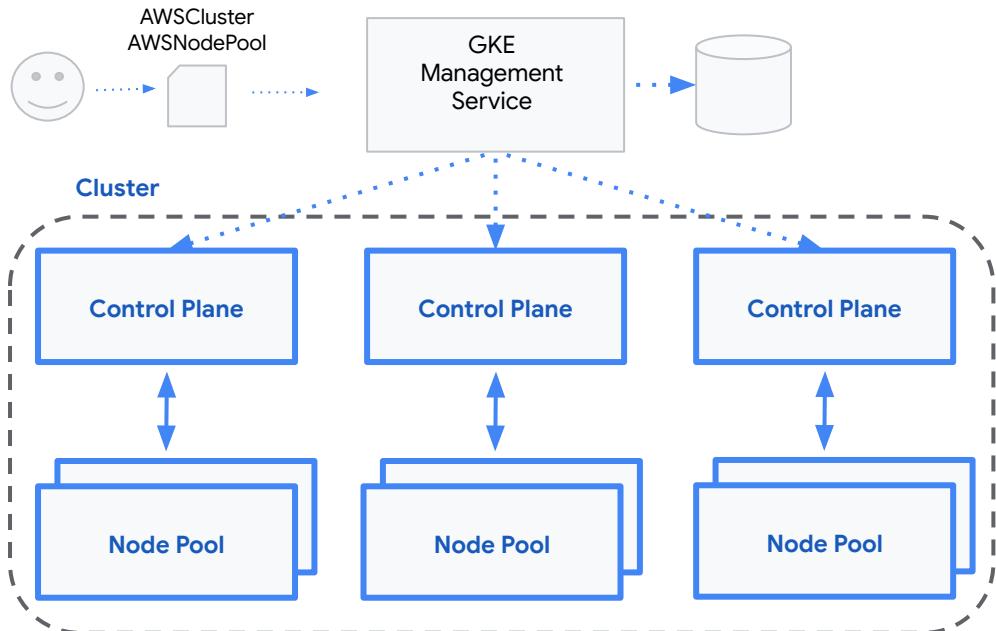
- Balancea cargas de trabajo y el master/api server de K8s
- Soporta:
 - Bundled-LB, Balanceador provisto y soportado por Google (Recomendado)
 - F5 Big IP (Físico & Virtual)
 - Bring your load balancer (BYOLB), i.e Citrix y pronto NSX



Jump host
“gkeadm”

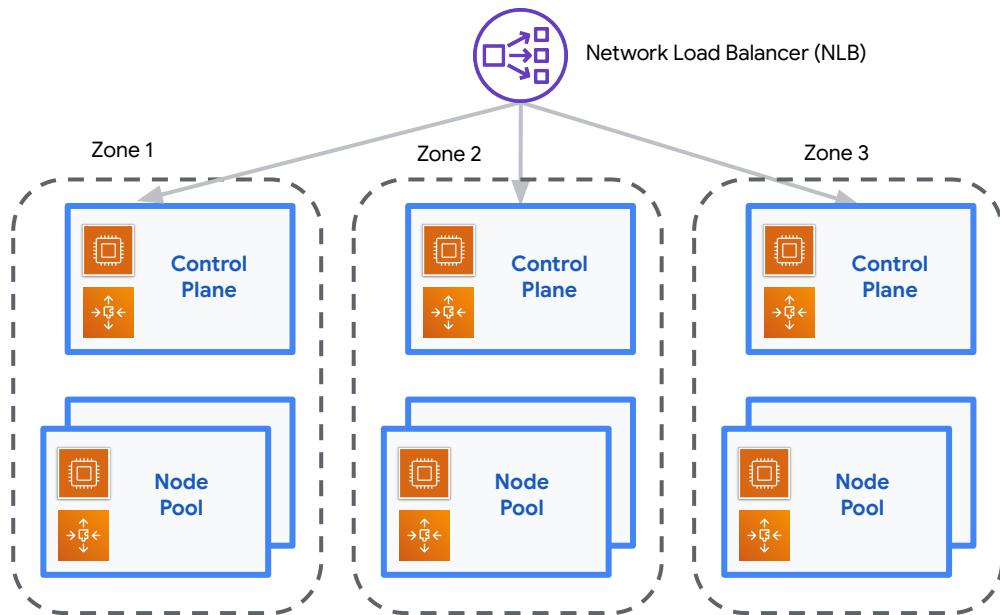


Arquitectura Multi-Nube



- **GKE Management Service** es desplegado en la misma VPC que los clusters que administra.
- GKE Management Service hace peticiones al API de AWS/Azure para aprovisionar recursos del cluster.
- El [Connect](#) agent es desplegado en el cluster para habilitar capacidades de administración y visibilidad multi-nube desde la consola de Google Cloud.

Topología en AWS para un cluster



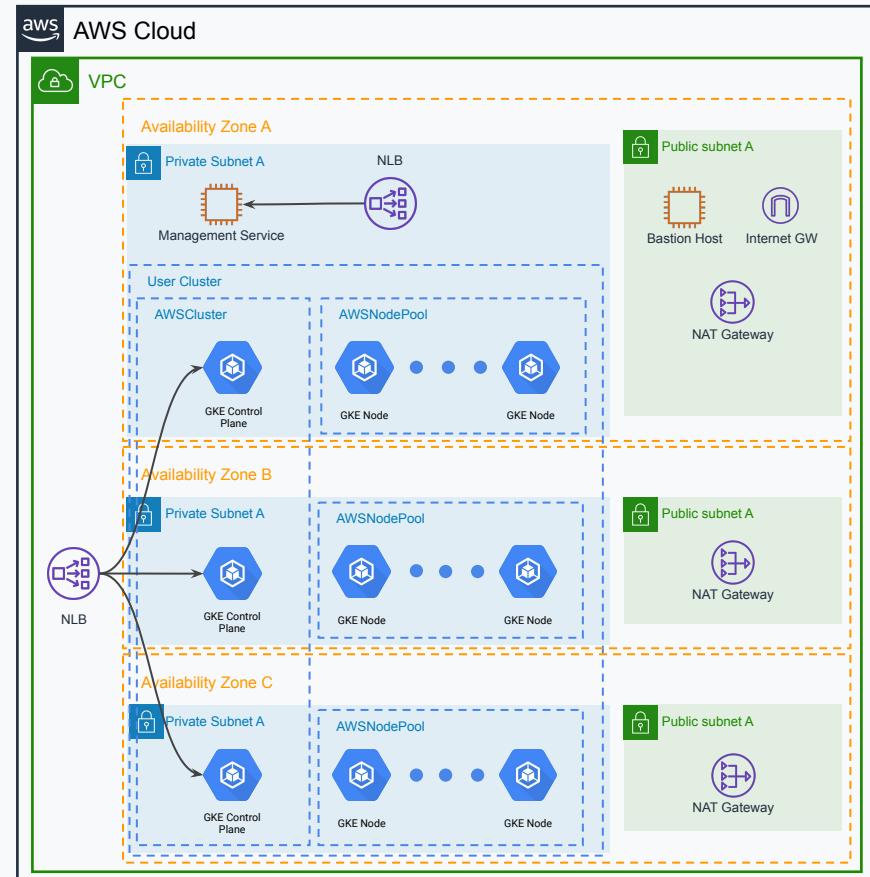
- NLB es utilizado por el endpoint del K8s API
- Las instancias del plano de control son desplegadas en 3 zonas de disponibilidad, y cada una cuenta con un grupo de autoescalamiento para resiliencia
- Múltiples node pools se pueden crear, y cada node pool tiene un grupo de autoescalamiento para resiliencia
- Cada node pool es desplegada en una única zona de disponibilidad (requerido para que el cluster autoscaling funcione de manera confiable)

Componentes de Arquitectura

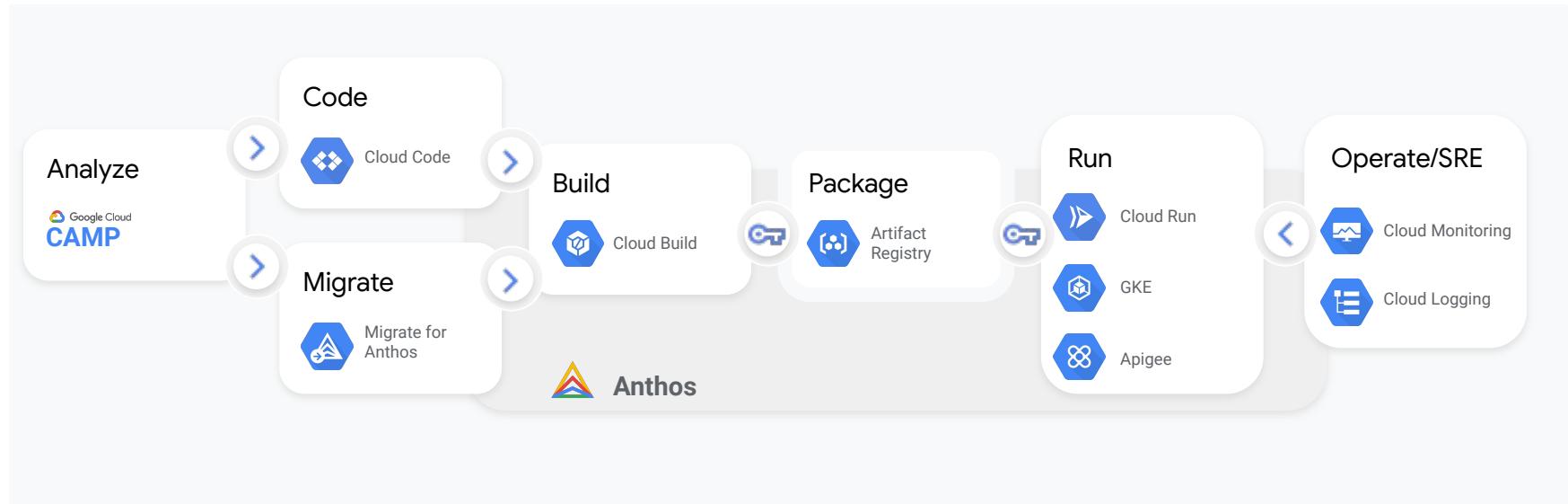
Cluster con Alta Disponibilidad

Anthos GKE on AWS soporta alta disponibilidad de un Cluster de Usuario dentro de una misma VPC.

- Los nodos de control de un **AWSCluster** son desplegados en **múltiples zonas de disponibilidad**.
- Las **AWSNodePools** son desplegadas en **zonas de disponibilidad individuales**. Múltiples Node Pools son necesarias para mayor disponibilidad y resiliencia.



Anthos es parte de una solución **end-to-end** para la entrega de aplicaciones modernas





Let's get solving

Google Cloud