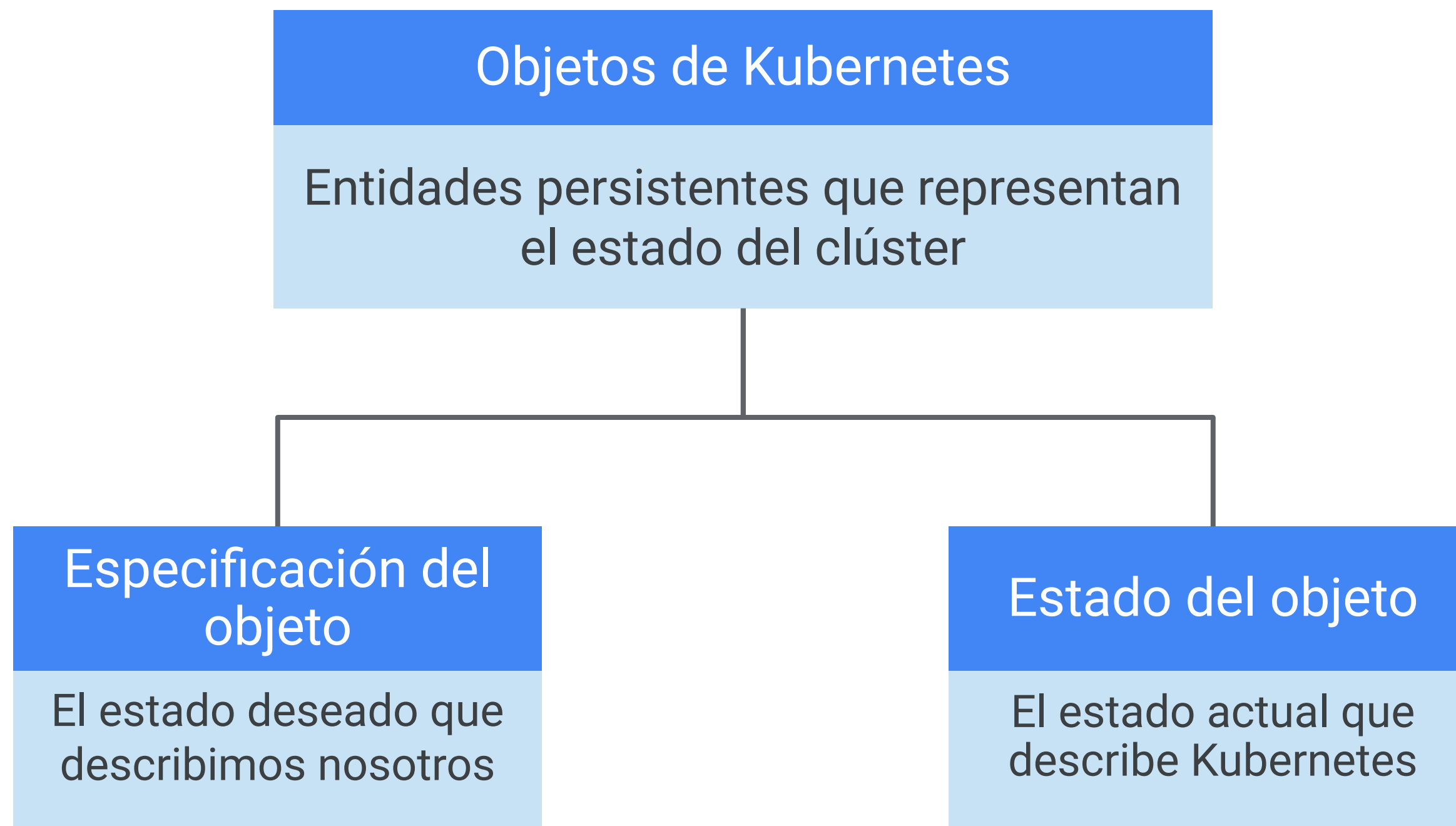




# Arquitectura de Kubernetes



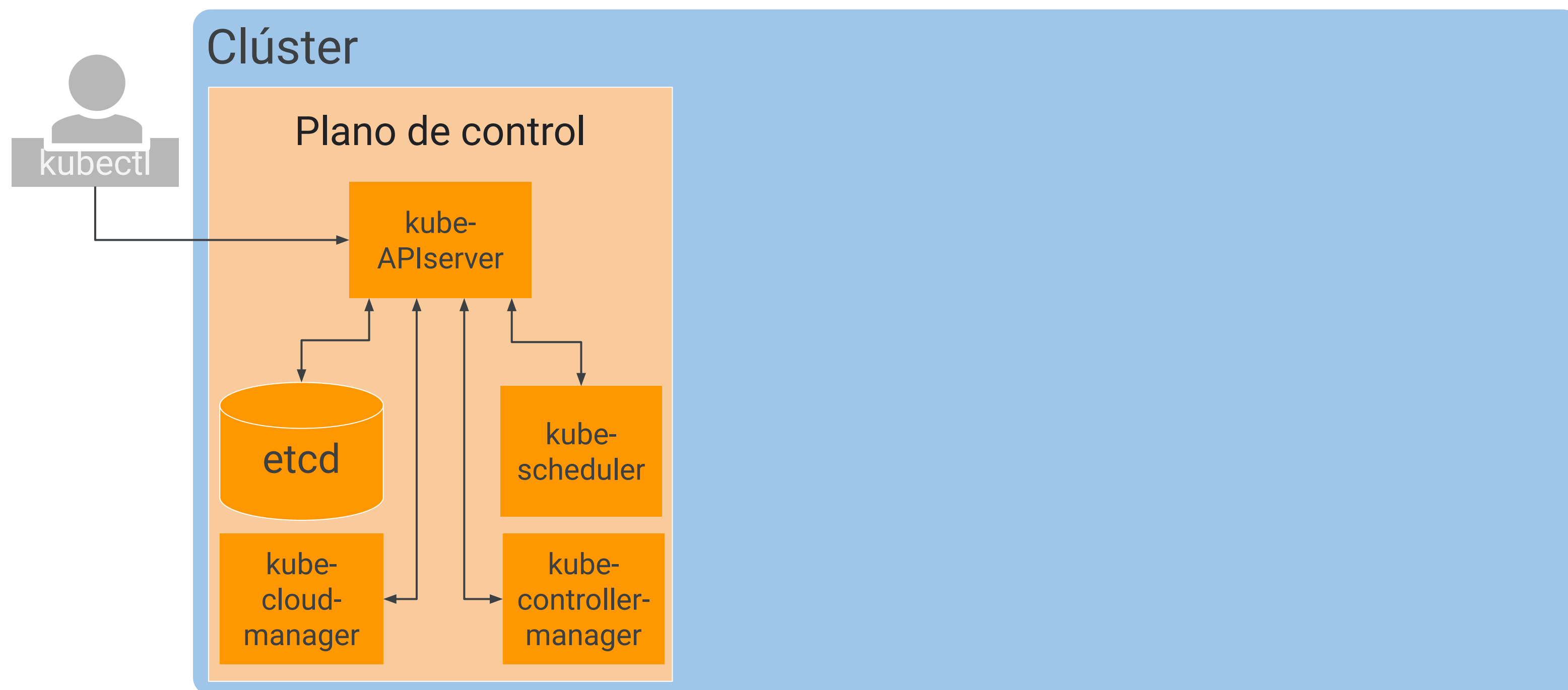
# Existen dos elementos en los objetos de Kubernetes



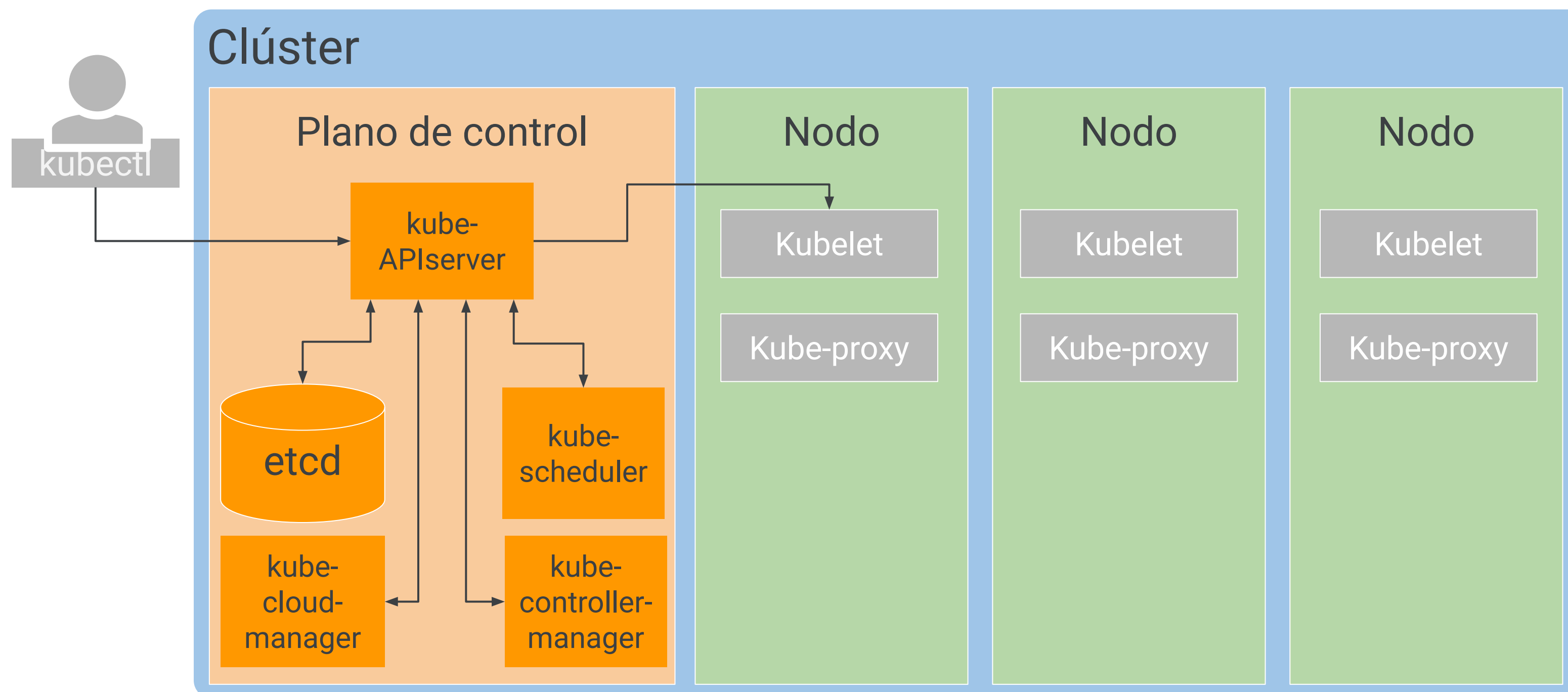
# Los contenedores de un Pod comparten recursos



# Los procesos cooperativos hacen funcionar los clústeres de Kubernetes



# Los procesos cooperativos hacen funcionar los clústeres de Kubernetes



---

# Temario

Conceptos de Kubernetes

Componentes de Kubernetes

Conceptos de  
Google Kubernetes Engine

Administración de objetos

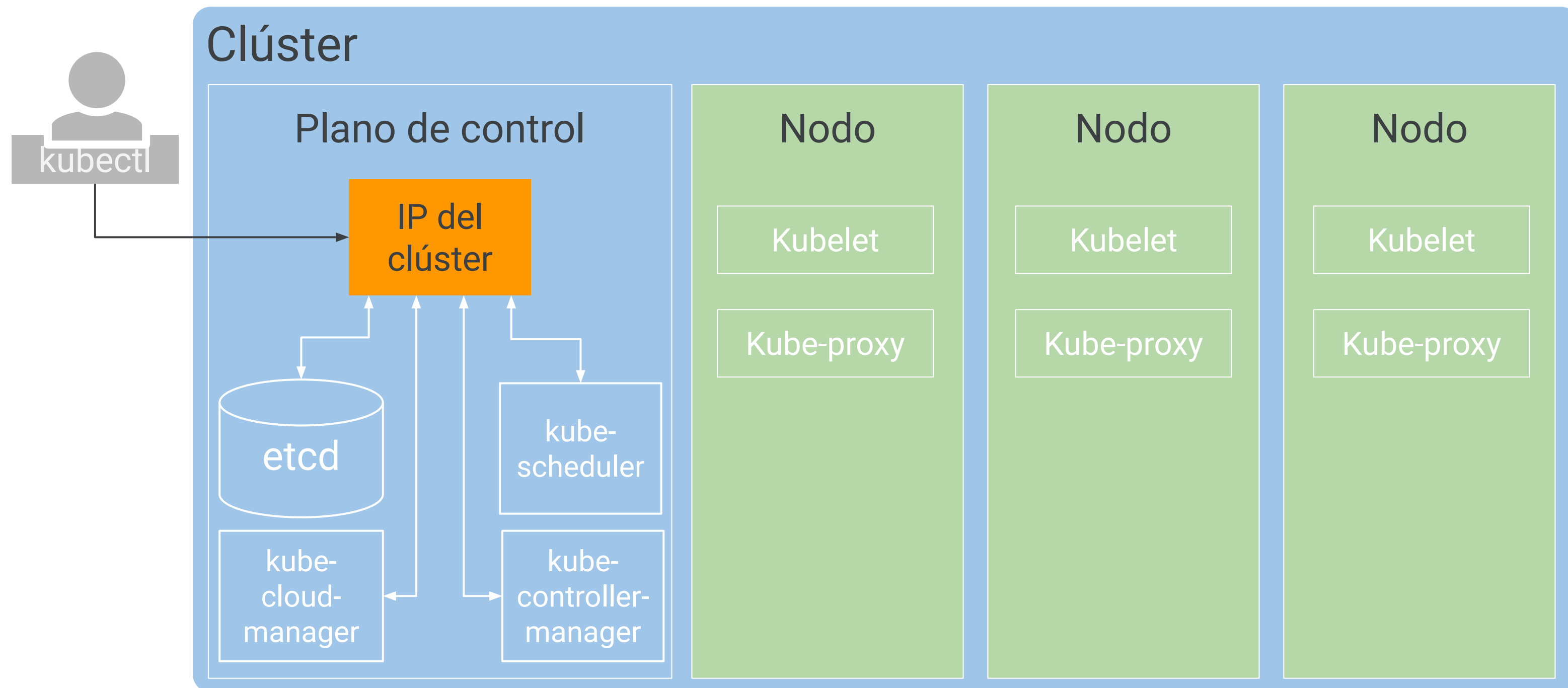
Lab: Cómo implementar  
Google Kubernetes Engine

Migrate for Anthos

Cuestionario

Resumen

# GKE administra todos los componentes del plano de control



---

# GKE: Más información sobre los nodos



Kubernetes no crea nodos.  
Los administradores de clústeres crean nodos y los agregan a Kubernetes.



GKE administra este aspecto implementando y registrando las instancias de Compute Engine como nodos.



# GKE: Más información sobre los nodos

## Machine configuration

**Machine family**

GENERAL-PURPOSE COMPUTE-OPTIMIZED MEMORY-OPTIMIZED GPU

Machine types for common workloads, optimized for cost and flexibility


**Series**

E2

CPU platform selection based on availability

**Machine type**

e2-medium (2 vCPU, 4 GB memory)

	vCPU	Memory
	1 shared core	4 GB

**CPU platform**

Automatic

## Machine configuration

**Machine family**

GENERAL-PURPOSE COMPUTE-OPTIMIZED MEMORY-OPTIMIZED GPU

High-performance machine types for compute-intensive workloads

**Series**

C2

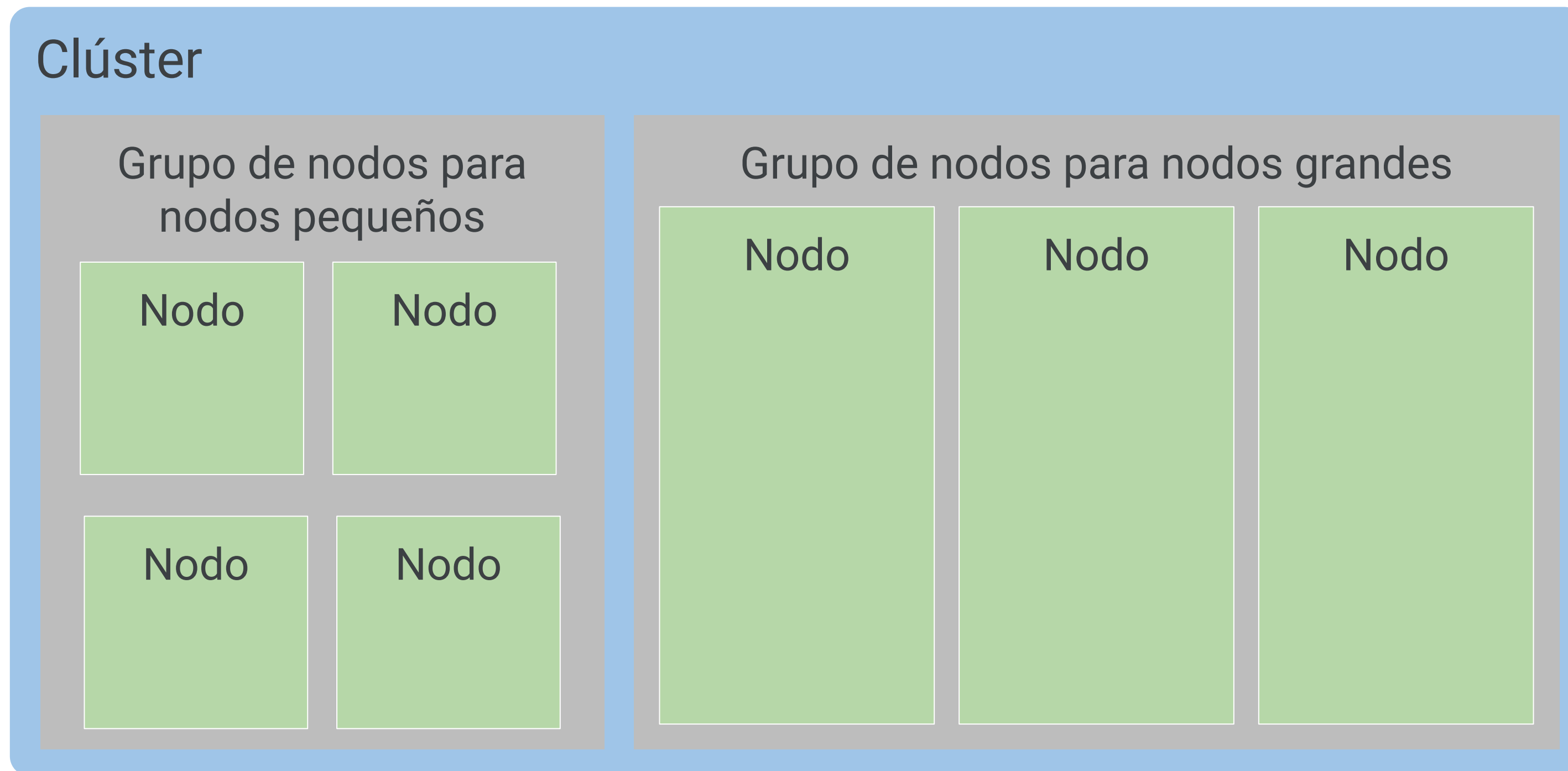
Powered by Intel Cascade Lake CPU platform

**Machine type**

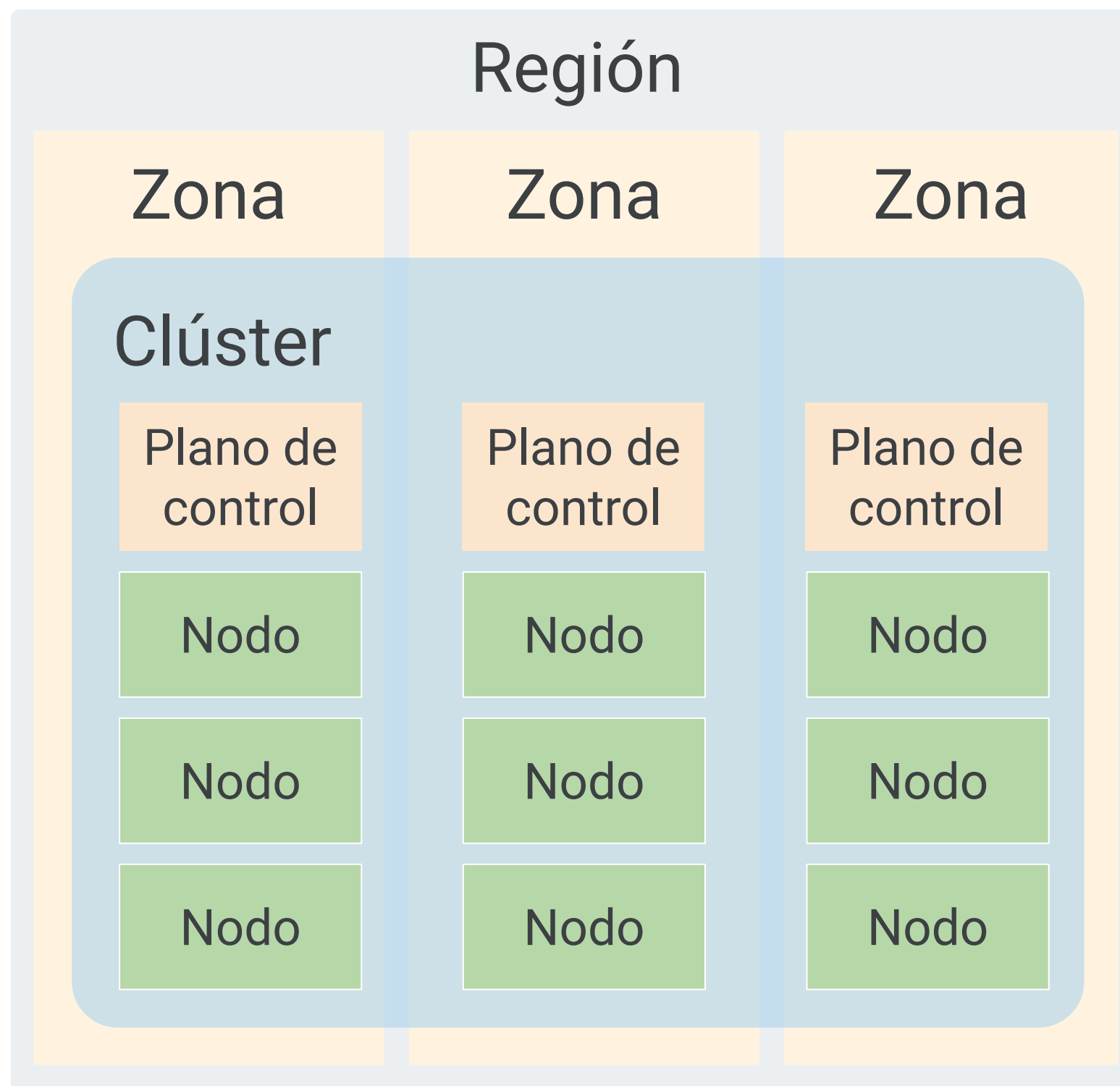
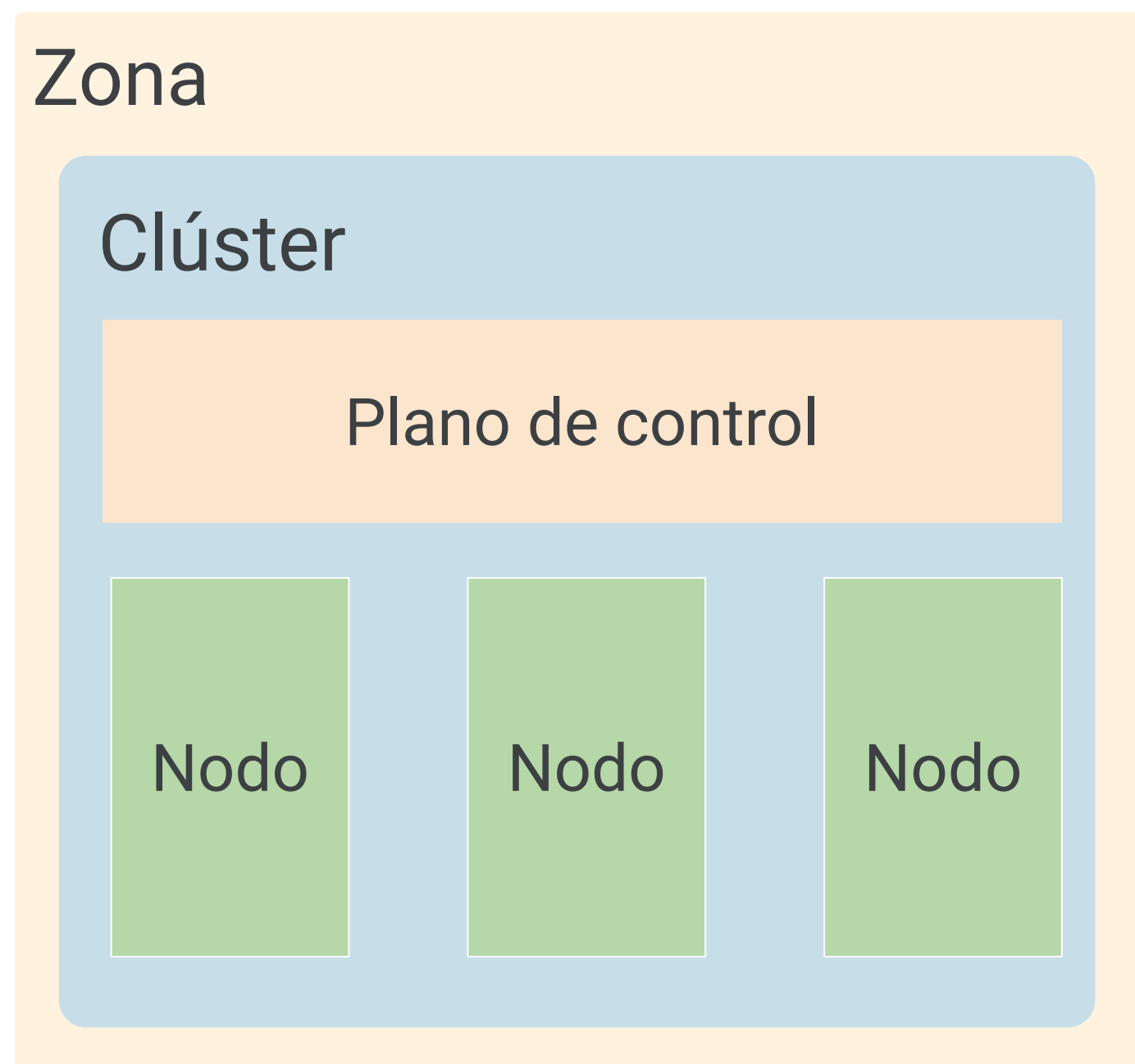
- c2-standard-4  
4 vCPU, 16 GB memory
- c2-standard-8  
8 vCPU, 32 GB memory
- c2-standard-16  
16 vCPU, 64 GB memory
- c2-standard-30  
30 vCPU, 120 GB memory
- c2-standard-60  
60 vCPU, 240 GB memory

+ ADD GPU

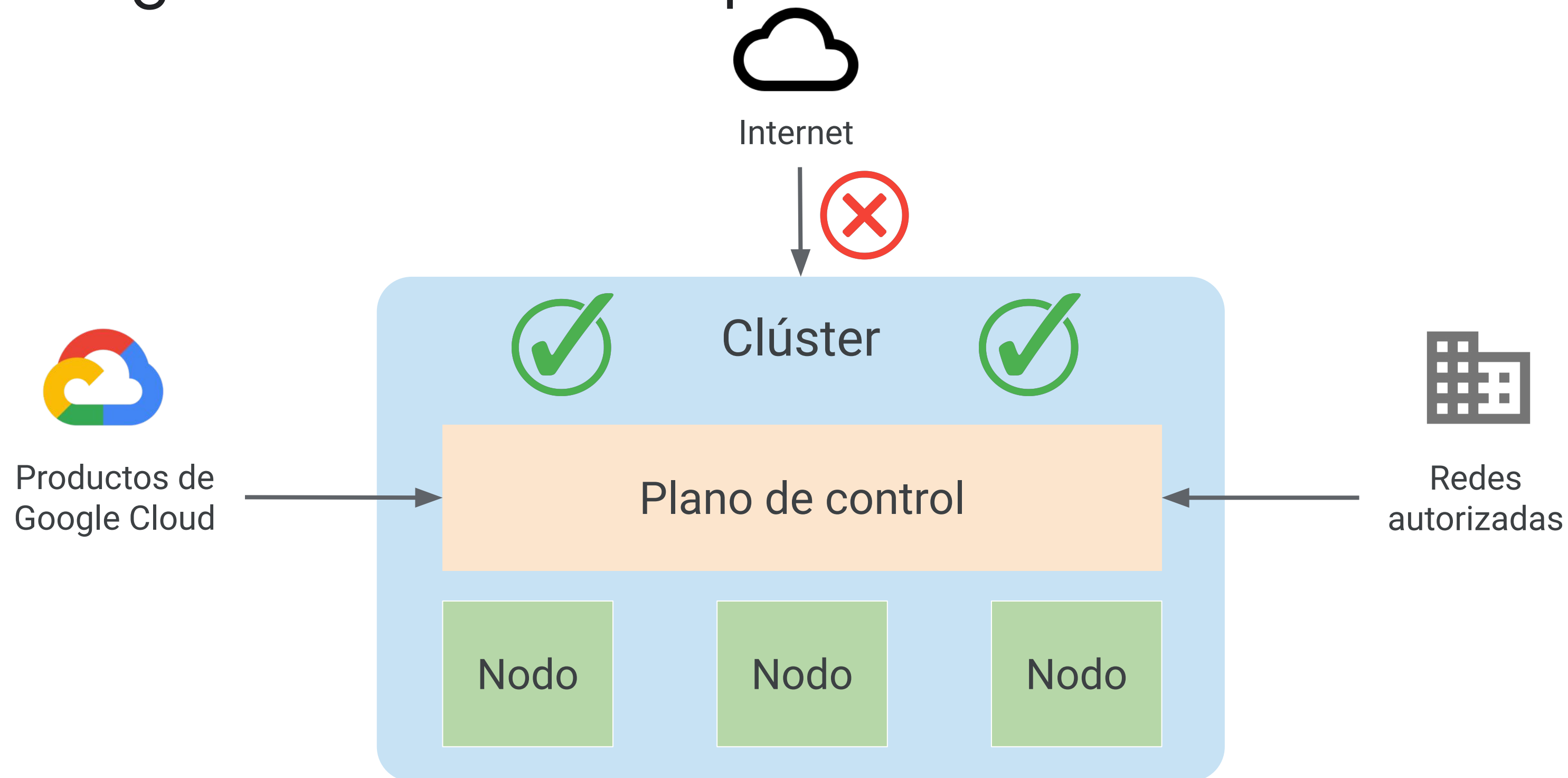
# Usa grupos de nodos para administrar diferentes tipos de nodos



# Clústeres zonales versus regionales



# Un clúster de GKE regional o zonal también se puede configurar como clúster privado



---

# Temario

Conceptos de Kubernetes

Componentes de Kubernetes

Conceptos de  
Google Kubernetes Engine

Administración de objetos

Lab: Cómo implementar  
Google Kubernetes Engine

Migrate for Anthos

Cuestionario

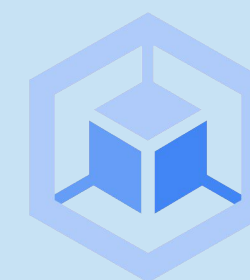
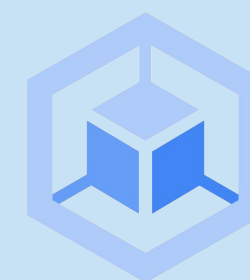
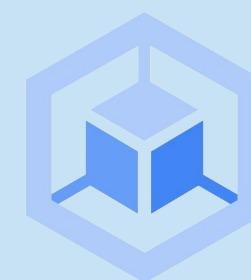
Resumen

# Cómo ejecutar tres contenedores de NGINX

Quieres tres contenedores de NGINX que se ejecuten todo el tiempo

¿Cómo creamos Pods para estos contenedores?

Especificaciones de objeto  
de los objetos de Kubernetes (estado  
deseado)

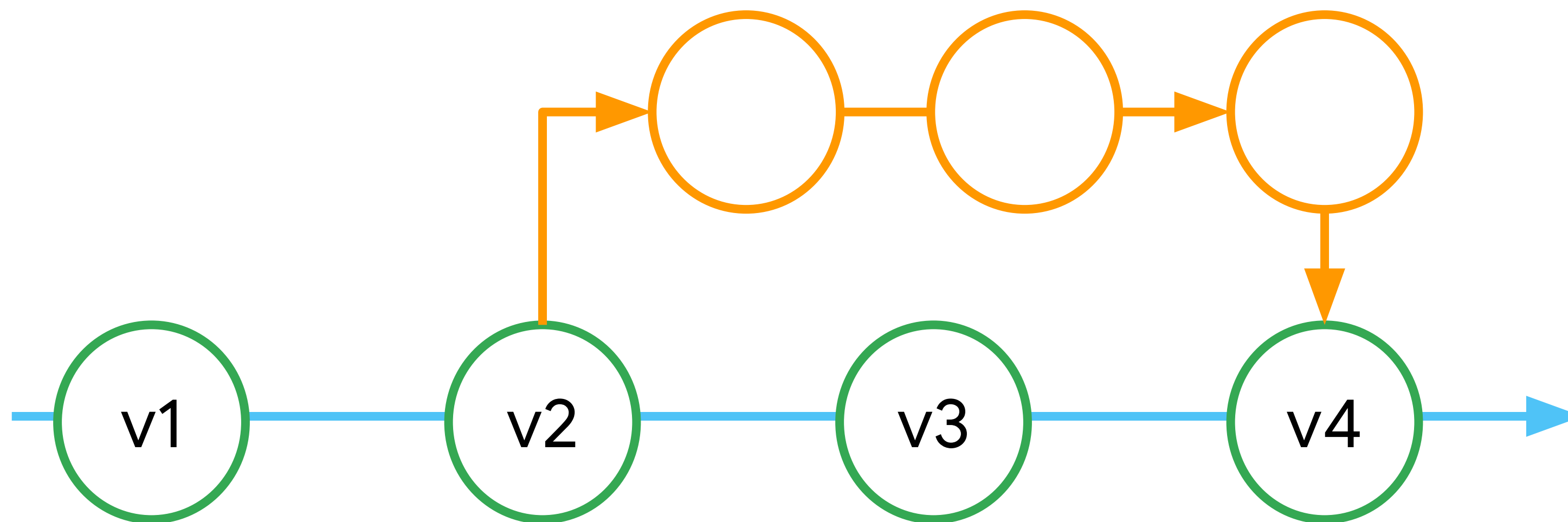


---

# Los objetos se definen en un archivo YAML

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
spec:
  containers:
  - name: nginx
    image: nginx:latest
```

# Práctica recomendada: Usa el control de versión en los archivos YAML





# Todos los objetos se identifican con un nombre

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
[...]
```

No puede haber dos del mismo tipo de objeto con el mismo nombre

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
[...]
```

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
[...]
```

Si se borra un objeto, el nombre se puede volver a usar.

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
[...]
```

---

# Kubernetes les asigna a todos los objetos un identificador único (UID)

```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
  uid: 4dd474fn-f389-11f8-b38c-42010a8009z7
[...]
```

# Las etiquetas se pueden hacer coincidir con selectores de etiquetas

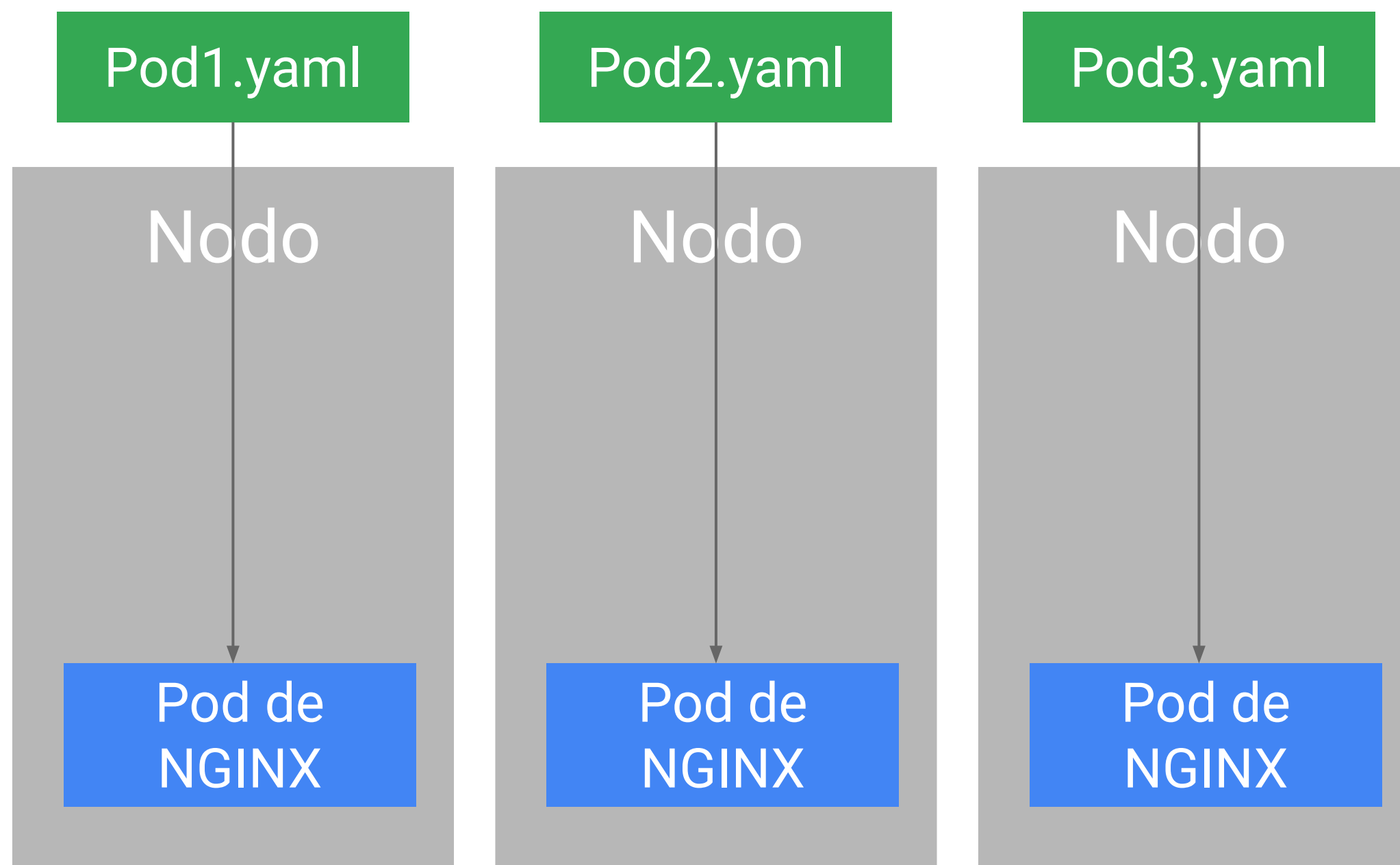
```
apiVersion: apps/v1
kind: Pod
metadata:
  name: nginx
  labels:
    app: nginx
    env: dev
    stack: frontend
spec:
  selector:
    matchLabels:
      app: nginx
```



El administrador emite un comando

```
kubectl get pods --selector=app=nginx
```

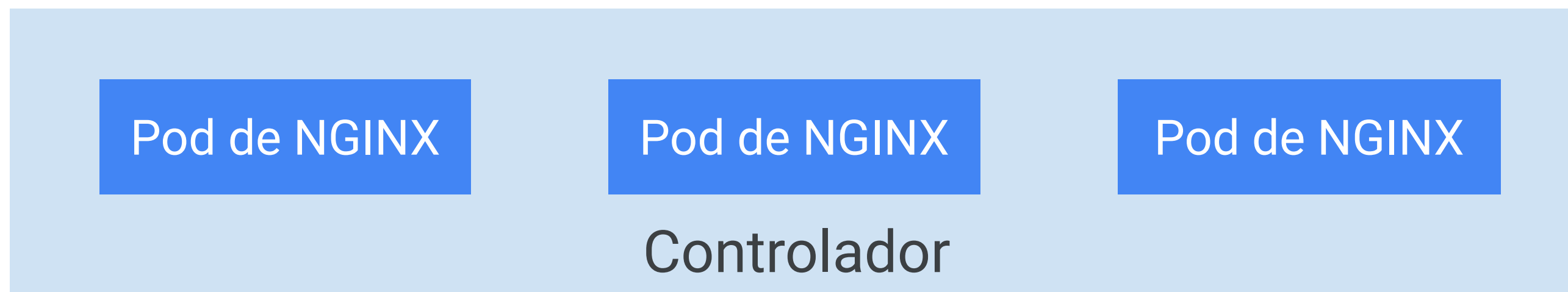
Una carga de trabajo se distribuye uniformemente entre los nodos disponibles de forma predeterminada



# Los Pods tienen un ciclo de vida



# Pods y objetos del controlador



Tipos de objetos del controlador

- Deployment
- StatefulSet
- DaemonSet
- Job

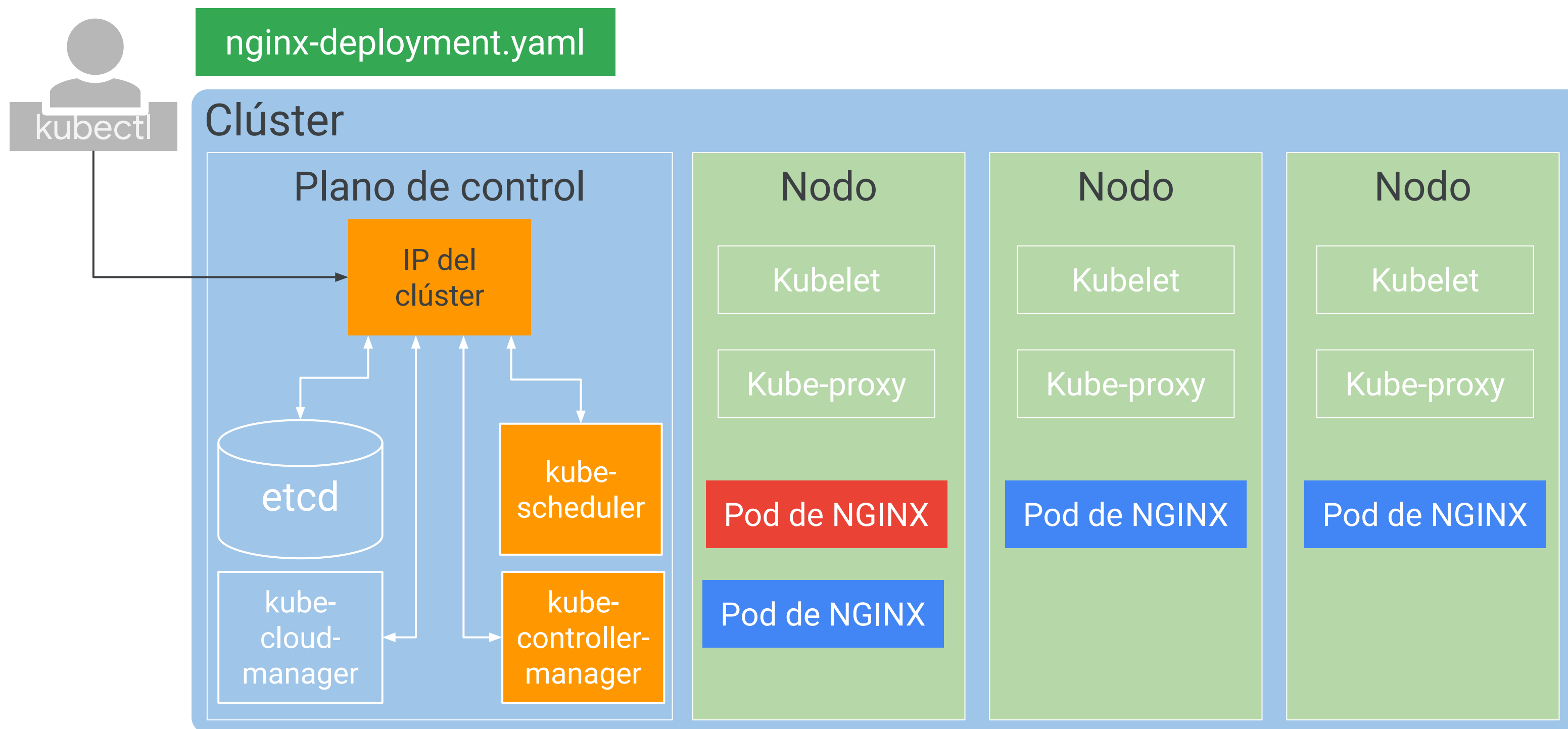
---

# Los objetos Deployment son una opción excelente para los componentes de software de larga duración

Quieres tres contenedores de NGINX que se ejecuten todo el tiempo

¿Cómo Kubernetes mantiene 3 contenedores de NGINX en cualquier momento?

# Un objeto Deployment mantiene el estado deseado





---

# Los objetos Deployments garantizan que los conjuntos de Pods se estén ejecutando

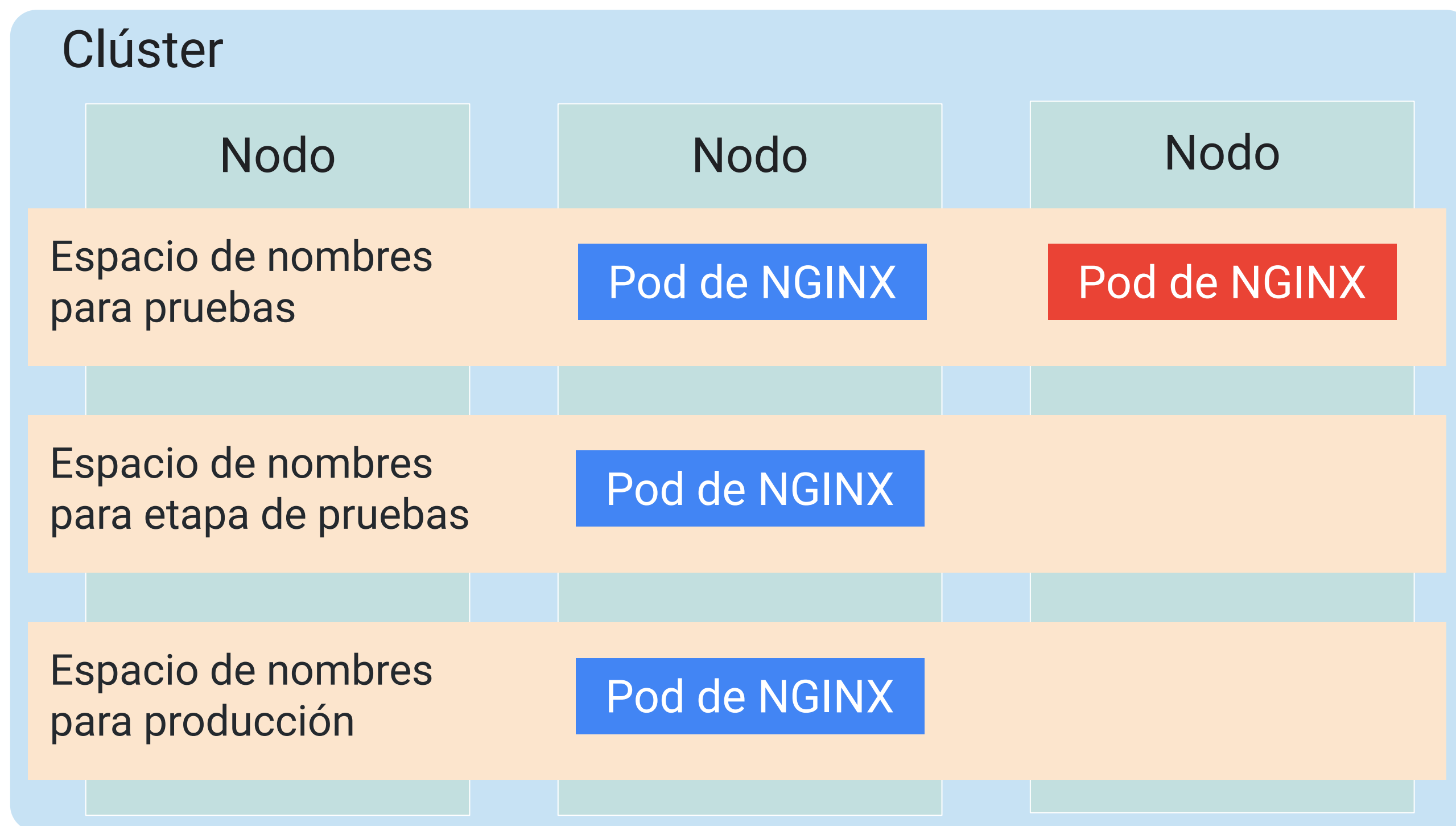
```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: nginx-deployment
  labels:
    app: nginx
spec:
  replicas: 3
  template:
    metadata:
      labels:
        app: nginx
    spec:
      containers:
        - name: nginx
          image: nginx:latest
```

---

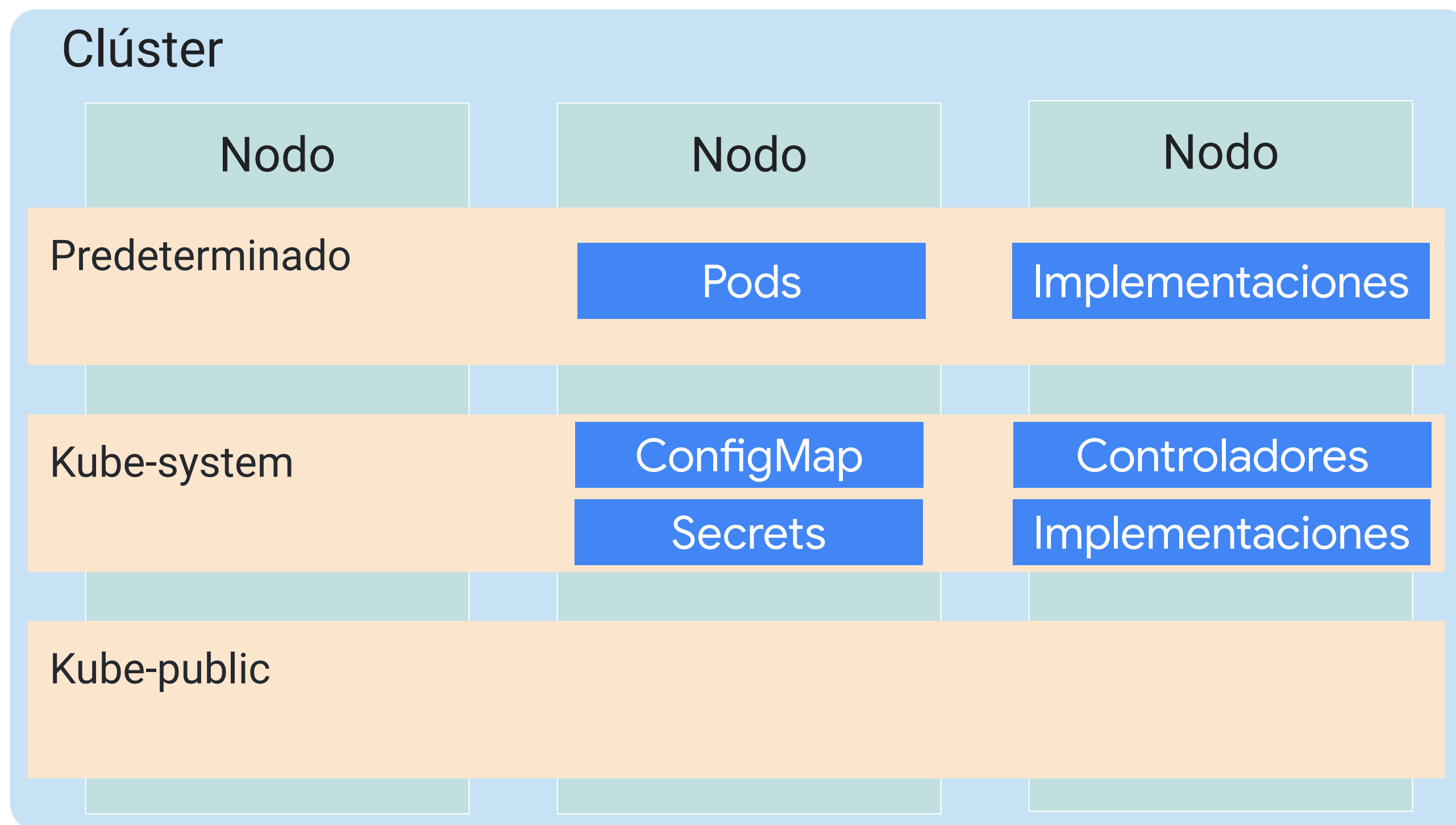
# Administración de recursos para los Pods y contenedores

- ✓ Es importante que los contenedores tengan suficientes recursos para ejecutarse.
- ✓ Las aplicaciones podrían usar más recursos de los que deberían.
- ✓ Los recursos de CPU y memoria (RAM) son los recursos especificados más comunes.

# Los espacios de nombres proporcionan permiso para nombrar recursos



# Existen tres espacios de nombres iniciales en un clúster



---

# Práctica recomendada: Sintaxis de YAML con neutralidad de espacios de nombres



Más flexible:

```
kubectl -n demo apply -f mypod.yaml
```



Admitida, pero  
menos flexible:

```
apiVersion: v1
kind: Pod
metadata:
  name: mypod
  namespaces: demo
```

---

# Introducción al lab

Cómo implementar  
Google Kubernetes Engine



---

# Migrate for Anthos traslada VMs a contenedores



Traslada y convierte cargas de trabajo en contenedores.



Las cargas de trabajo pueden comenzar como servidores físicos o VMs.



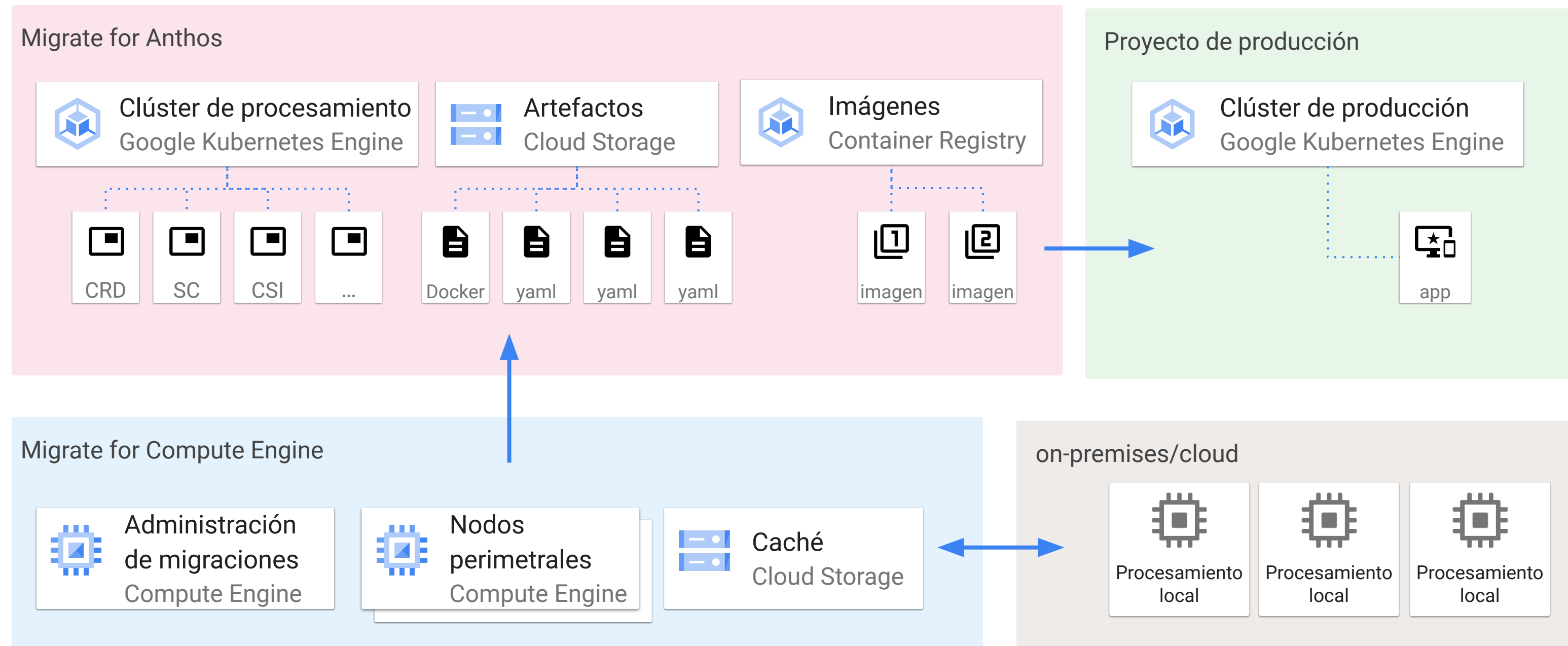
Traslada el procesamiento de cargas de trabajo a contenedores de inmediato (menos de 10 min).



Los datos se pueden migrar todos al mismo tiempo o “transmitir” a la nube hasta que la app esté activa en la nube.



# Para crear una migración, se necesita una arquitectura





# Una migración es un proceso de varios pasos



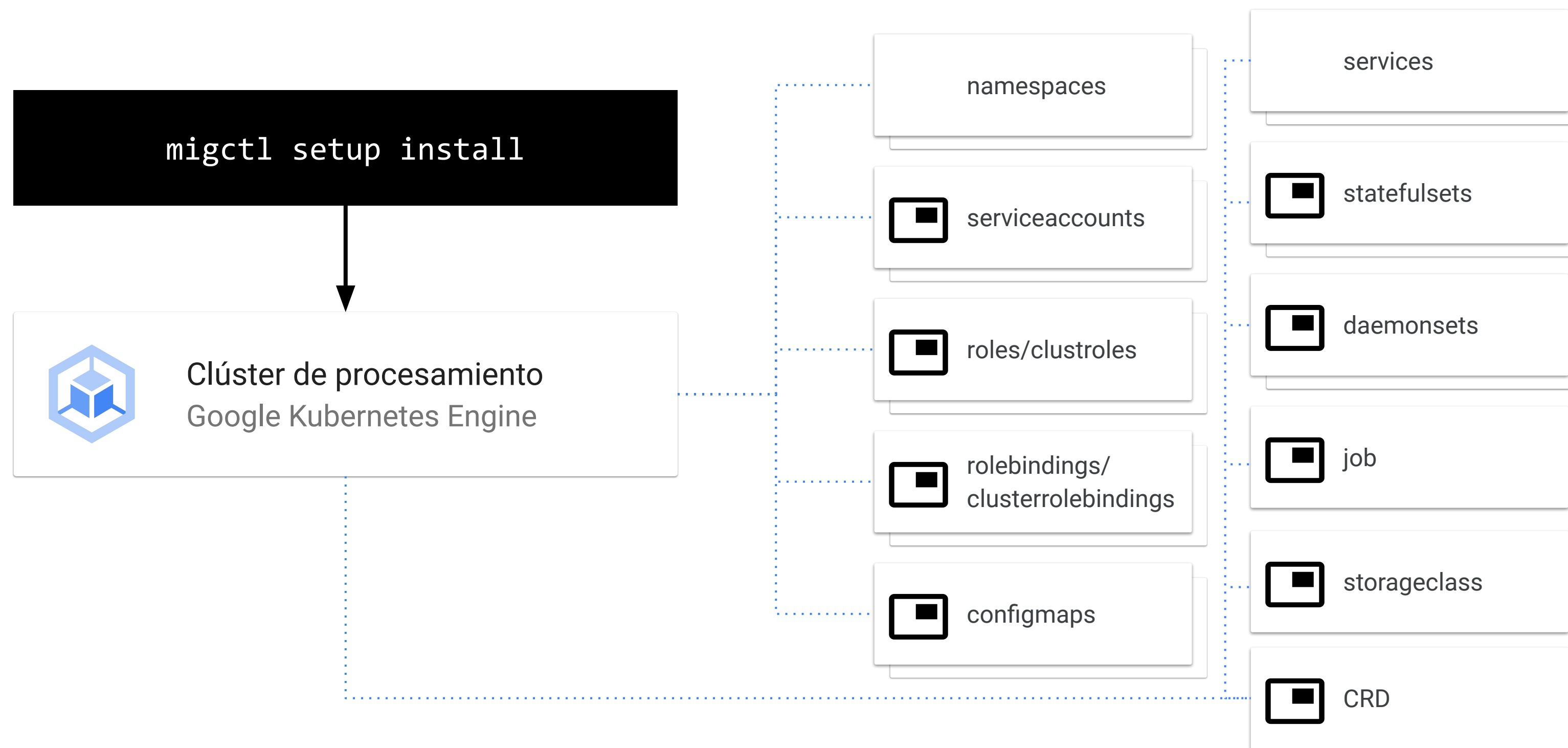
# Migrate for Anthos necesita un clúster de procesamiento

```
gcloud container --project $PROJECT_ID \
clusters create $CLUSTER_NAME \
--zone $CLUSTER_ZONE \
--username "admin" \
--cluster-version 1.14 \
--machine-type "n1-standard-4" \
--image-type "UBUNTU" \
--num-nodes 1 \
--enable-stackdriver-kubernetes \
--scopes "cloud-platform" \
--enable-ip-alias \
--tags="http-server"
```



Clúster de procesamiento  
Google Kubernetes Engine

# En la instalación de Migrate for Anthos, se usa `migctl`



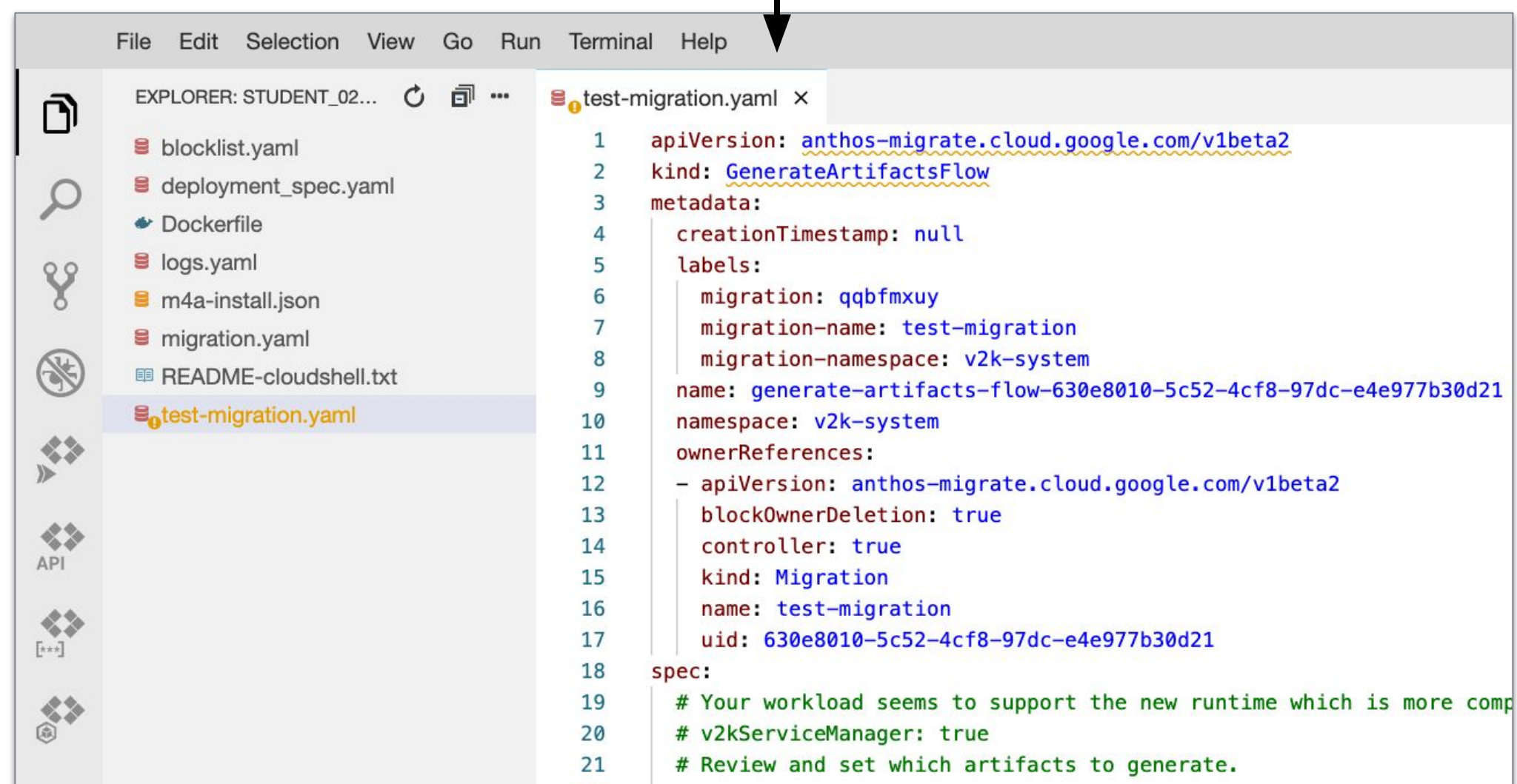
---

# Agregar una fuente permite las migraciones desde un entorno específico

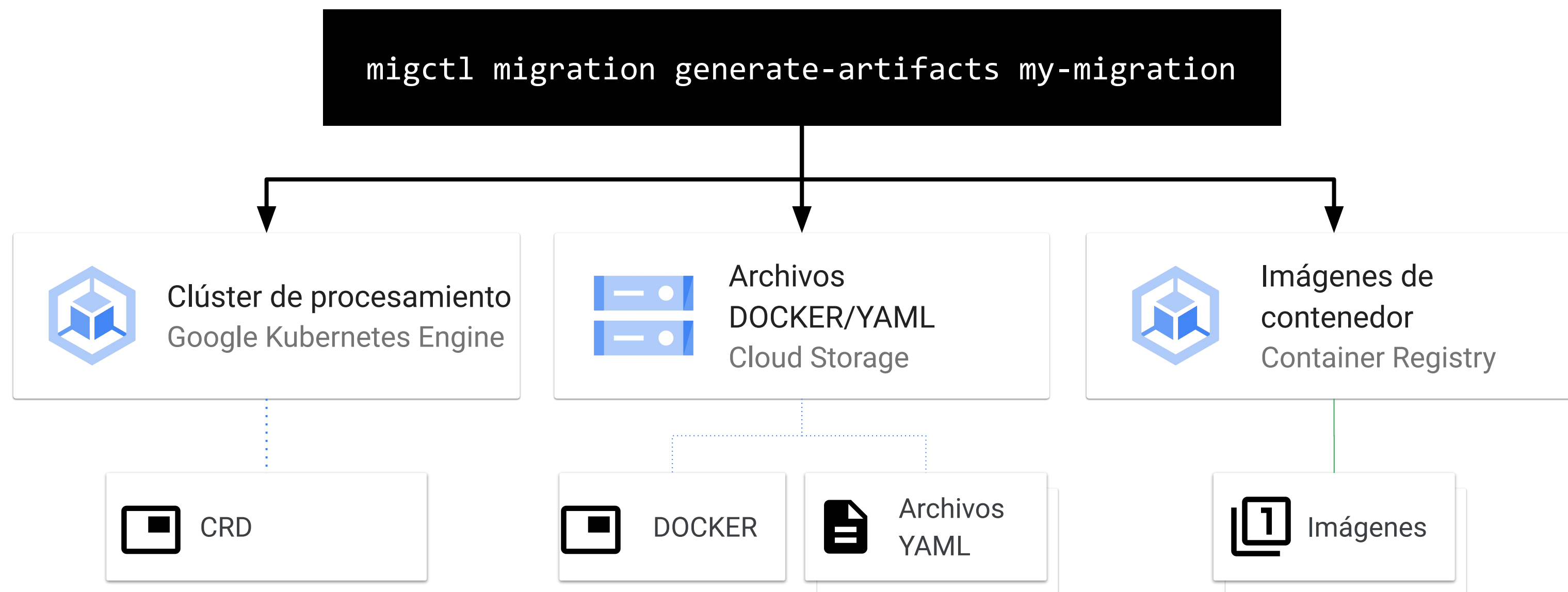
```
migctl source create ce my-ce-src --project my-project --zone zone
```

# Crear una migración genera un plan de migración

```
migctl migration create test-migration --source my-ce-src --vm-id my-id --intent Image
```

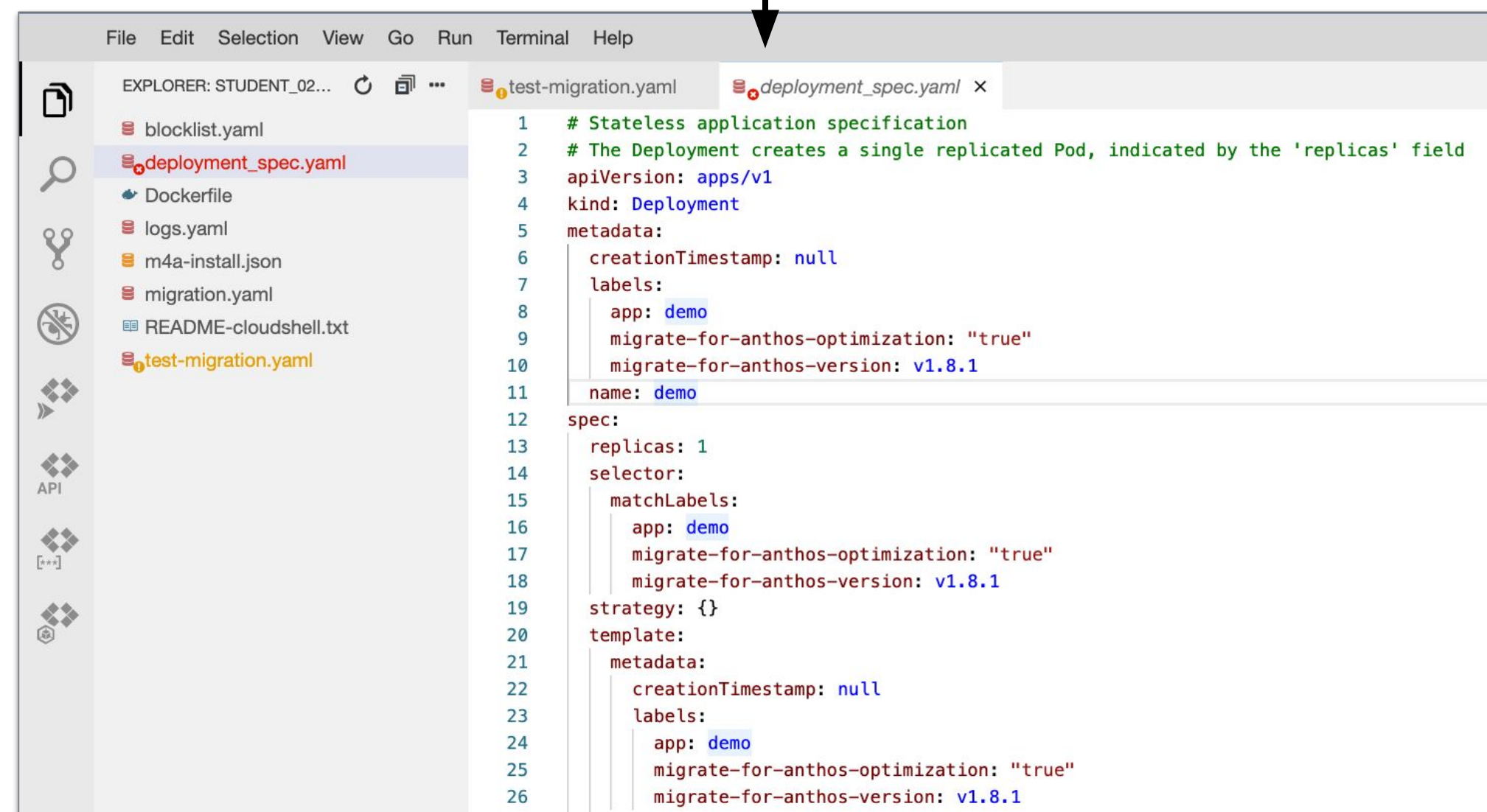


# Ejecutar una migración genera recursos y artefactos



# Los archivos de implementación normalmente necesitan modificaciones

```
migctl migration get-artifacts test-migration
```



```
1 # Stateless application specification
2 # The Deployment creates a single replicated Pod, indicated by the 'replicas' field
3 apiVersion: apps/v1
4 kind: Deployment
5 metadata:
6   creationTimestamp: null
7   labels:
8     app: demo
9     migrate-for-anthos-optimization: "true"
10    migrate-for-anthos-version: v1.8.1
11   name: demo
12 spec:
13   replicas: 1
14   selector:
15     matchLabels:
16       app: demo
17       migrate-for-anthos-optimization: "true"
18       migrate-for-anthos-version: v1.8.1
19   strategy: {}
20   template:
21     metadata:
22       creationTimestamp: null
23     labels:
24       app: demo
25       migrate-for-anthos-optimization: "true"
26       migrate-for-anthos-version: v1.8.1
```