# Paso 1: Instalar Docker Desktop (para macOS con chip M4)

Docker Desktop instala y configura el motor de contenedores Docker que Colima utilizará como backend. Además, aunque Colima puede funcionar sin Docker Desktop, tenerlo instalado facilita la gestión y la compatibilidad.

## 🔧 Instrucciones para instalar:

1. Ve al sitio oficial:  
   👉 https://www.docker.com/products/docker-desktop/
2. Haz clic en el botón **"Download for Mac (Apple chip)"**.
3. Una vez descargado el .dmg, **abre el archivo** y **arrastra el icono de Docker a la carpeta Aplicaciones**.
4. Abre Docker desde "Aplicaciones" y dale permisos cuando lo solicite.
5. Espera que el icono de Docker aparezca en la barra superior (cerca del reloj) y muestre “Docker Desktop is running”.
6. Verifica en la terminal:

docker --version

Docker version 24.x.x, build xxxxxxx

# Paso 2: Instalar Kubectl (para macOS con chip M4)

## 🔹 ¿Por qué instalar kubectl antes que Colima?

Porque kubectl es la herramienta **oficial para interactuar con clústeres Kubernetes**, y Colima lo detecta automáticamente si ya está presente. Además, será **la principal herramienta** que usaremos para verificar, aplicar y administrar recursos en el clúster.

## 🔧 Instrucciones para instalar kubectl en macOS

### ✅ Opción recomendada: Homebrew

Si ya tienes Homebrew (el gestor de paquetes de macOS), simplemente ejecuta:

brew install kubectl

Y luego verifica:

kubectl version --client

Deberías ver algo como:

Client Version: v1.32.x

Kustomize Version: v5.5.0

### ❗ Si NO tienes Homebrew:

Instálalo con este comando:

/bin/bash -c "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/HEAD/install.sh)"

Luego instala kubectl:

brew install kubectl

# Paso 3: Instalar Colima

## 🔹 ¿Qué es Colima y por qué lo usamos?

**Colima** (Container On Localhost, Inspired by Lima) es una alternativa moderna a Docker Desktop que funciona muy bien con:

* Docker CLI (usando containerd o podman)
* Kubernetes (con soporte nativo a través de k3s)
* Virtualización eficiente para entornos Apple Silicon (como tu Mac M4)

Usarlo es ideal porque:

* Es **ligero y rápido**
* Es **open source**
* Funciona muy bien con herramientas modernas de desarrollo

## 🔧 ¿Cómo instalar Colima en macOS?

### ✅ Opción recomendada: Usar Homebrew

brew install colima

Una vez instalado, verifica que funciona:

colima version

Deberías ver algo como:

colima version 0.8.x

git commit: <algo>

### 🧪 Verifica que todo está bien instalado

colima --help

Esto debería mostrarte todas las opciones y subcomandos disponibles de Colima.

## Detener Colima

### 🛑 1. ****Detener Colima si está corriendo****

colima stop

Esto detiene la instancia actual de Colima.

### 🧹 2. ****Eliminar completamente la instancia****

Esto elimina la máquina virtual de Colima y todos sus recursos:

colima delete

✅ Si tienes múltiples perfiles de Colima, puedes especificar el nombre (por defecto es default):

colima delete --profile default

### 🧼 3. (Opcional) ****Eliminar configuración residual de Colima****

Si quieres ir más a fondo (como un "factory reset"), puedes borrar la carpeta de configuración manualmente:

rm -rf ~/.colima

Esto elimina **todos los perfiles, configuraciones, y datos persistentes** que Colima guarda localmente.

### 📦 4. (Opcional) ****Limpiar imágenes y volúmenes de Docker****

Ya que Colima usa Docker internamente (vía Lima), puedes limpiar:

docker system prune -a --volumes

**⚠️ Esto eliminará todas las imágenes, contenedores detenidos y volúmenes que no estén en uso.**

# Paso 4: Iniciar Colima con Kubernetes habilitado

Este paso crea una nueva máquina virtual con soporte para Kubernetes.

colima start --with-kubernetes --profile taskmanager --cpu 4 --memory 6 --disk 40

## 💬 Explicación de los parámetros:

| Opción | Significado |
| --- | --- |
| --with-kubernetes | Activa el clúster de Kubernetes dentro de Colima |
| --profile taskmanager | Crea un entorno aislado llamado taskmanager. Es un entorno de Colima que define un clúster Kubernetes aislado con su propia configuración de CPU, memoria, disco, puertos, etc. |
| --cpu 4 | Asigna 4 núcleos de CPU virtual |
| --memory 6 | Asigna 6 GB de RAM |
| --disk 40 | Asigna 40 GB de disco para almacenamiento persistente |

## 🚀 ¿Qué pasa después?

Se crea un perfil personalizado con los recursos definidos. Se levanta una máquina virtual Linux con Docker y Kubernetes embebido. Se configura automáticamente un contexto kubectl para ese perfil.

## ✅ Comprobaciones posteriores:

kubectl config get-contexts

Y veras algo como:

kubectl get nodes

# Paso 5: Verificación del clúster Kubernetes

## 1 Verificar el estado del nodo (ya hecho)

kubectl get nodes

Resultado esperado: debe mostrar el nodo colima-taskmanager en estado Ready.

## 2 Verificar los componentes del sistema (kube-system)

kubectl get pods -n kube-system

Esto lista todos los pods que forman parte de la infraestructura básica de Kubernetes. Ejemplo esperado:

NAME READY STATUS RESTARTS AGE

coredns-66bff467f8-lpqk4 1/1 Running 0 2m

metrics-server-6d56bdc9fc-5b8t6 1/1 Running 0 2m

local-path-provisioner-5cf85fd84d-8lctk 1/1 Running 0 2m

## 3 Verificar el contexto y namespace actual

kubectl config get-contexts

Debe mostrar \* colima-taskmanager como el contexto actual (con asterisco \*).

También puedes verificar el namespace actual en uso:

kubectl config view --minify | grep namespace

## 4 Probar el despliegue de prueba (opcional pero útil)

Este paso crea un **pod de prueba simple** para asegurarte de que puedes desplegar workloads:

kubectl run test-pod --image=nginx --restart=Never

### Y luego puedes verificar que esté corriendo:

kubectl get pods

### Paso siguiente: Diagnóstico en profundidad

Si el POD se está demorando mucho en estar en estado **RUNNING**, por favor ejecuta este comando para ver los detalles:

kubectl describe pod test-pod

Y revisa especialmente la sección Events: al final del output. Ahí puede aparecer algo como:

* Failed to pull image → no puede descargar la imagen
* ImagePullBackOff → falló varias veces al descargar
* ErrImagePull → problema con el nombre o acceso a la imagen
* Warning FailedMount → problema de volumen o almacenamiento

### Una vez en estado Running, acceder a él:

kubectl port-forward test-pod 8080:80

Y abrir en el navegador:  
👉 <http://localhost:8080/>

### Cuando termines, puedes borrarlo con:

kubectl delete pod test-pod

# ****Paso 6: Instalar el Dashboard oficial****

## 🔹 1. Instalar el Dashboard

Ejecuta en **una sola terminal**:

kubectl apply -f https://raw.githubusercontent.com/kubernetes/dashboard/v2.7.0/aio/deploy/recommended.yaml

🔍 Esto descargará e instalará el Dashboard en el namespace kubernetes-dashboard, creando todos los recursos necesarios (pods, servicios, roles, etc.).

## 🔹 2. Crear una cuenta de acceso (ServiceAccount + ClusterRoleBinding)

Ahora debes crear una cuenta de servicio con permisos de administrador y vincularla al dashboard.

**Crea un archivo llamado dashboard-admin.yaml con el siguiente contenido:**

# dashboard-admin.yaml

apiVersion: v1

kind: ServiceAccount

metadata:

name: dashboard-admin-sa

namespace: kubernetes-dashboard

---

apiVersion: rbac.authorization.k8s.io/v1

kind: ClusterRoleBinding

metadata:

name: dashboard-admin-sa

roleRef:

apiGroup: rbac.authorization.k8s.io

kind: ClusterRole

name: cluster-admin

subjects:

- kind: ServiceAccount

name: dashboard-admin-sa

namespace: kubernetes-dashboard

Y luego aplícalo con:

kubectl apply -f dashboard-admin.yaml

## 🔹 3. Obtener el token para ingresar al dashboard

Ejecuta esto para obtener el token (en una sola línea):

kubectl -n kubernetes-dashboard create token dashboard-admin-sa

📌 Copia el token completo. Lo usarás para ingresar al Dashboard en el navegador.

## 🔹 4. Exponer el Dashboard en el navegador

Ejecuta este comando para exponer el Dashboard localmente:

kubectl proxy

🔗 Luego accede en tu navegador a:

http://localhost:8001/api/v1/namespaces/kubernetes-dashboard/services/https:kubernetes-dashboard:/proxy/

🔐 Cuando te lo pida, pega el **token** generado en el paso anterior.

# Crear y gestionar múltiples clústeres con Colima

Puedes tener **varios clústeres en tu Mac** usando Colima, siempre que sepas que **solo uno puede estar activo al mismo tiempo**. Aquí tienes el **paso a paso para crear y gestionar múltiples clústeres con perfiles diferentes**:

## Crear un nuevo clúster con un perfil diferente

colima start --profile taskmanager-cluster –kubernetes

Este comando:

* Crea un perfil de Colima con ese nombre.
* Inicia un clúster Kubernetes aislado para ese perfil.

## Ver a qué clúster kubectl está conectado

kubectl config get-contexts

## Ver todos los perfiles de Colima

colima list

Esto muestra todos los perfiles (clústeres) que se han creado y cuál está corriendo.

## Cambiar de clúster (detener uno y arrancar otro)

colima stop

colima start --profile otro-cluster --kubernetes

## Ver a qué clúster kubectl está conectado

kubectl config get-contexts

# Crear un Namespace

## ¿Qué es un Namespace?

Es como una **carpeta lógica o espacio de trabajo** dentro del clúster donde puedes agrupar:

* Pods
* Services
* ConfigMaps
* Secrets
* Deployments
* etc.

### ¿Para qué sirve?

1. **Organización por proyectos o equipos:**
   * Puedes tener frontend, backend, dev, prod, taskmanager, etc.
2. **Evita conflictos de nombres:**
   * Puedes tener un service llamado api-gateway en dos namespaces distintos sin conflicto.
3. **Separación de entornos:**
   * dev, test, prod pueden vivir en el mismo clúster, pero con aislamiento.
4. **Control de acceso (RBAC):**
   * Puedes dar permisos solo sobre un namespace a ciertos usuarios o servicios.
5. **Facilita limpieza o despliegues:**
   * Puedes eliminar todo lo asociado a un proyecto eliminando su namespace.

## Diferencia entre Cluster y Namespace

| Concepto | Cluster | Namespace |
| --- | --- | --- |
| **Qué es** | Un clúster es un conjunto completo de nodos (máquinas) controlado por una única instancia de Kubernetes. | Un namespace es un espacio lógico dentro de un mismo clúster. |
| **Alcance** | Es el nivel más alto. Abarca todos los recursos. | Es un subconjunto de recursos dentro del clúster. |
| **Aislamiento** | Fuerte: entornos completamente separados (red, almacenamiento, configuración) | Lógico: útil para separar proyectos o equipos dentro del mismo entorno. |
| **Casos de uso** | Separar ambientes totalmente distintos: producción vs desarrollo. Multi-tenant fuerte. | Separar equipos, apps o microservicios dentro del mismo entorno técnico. |
| **Requiere más recursos** | Sí. Correr varios clústeres implica múltiples instancias de K8s y nodos. | No. Es liviano: todo corre en el mismo plano de control y nodos. |

## ¿Entonces para qué tener múltiples clústeres?

Hay varios motivos válidos:

### Seguridad y cumplimiento

* Empresas que requieren **aislamiento físico o de red completo** (por normas como PCI-DSS, HIPAA).
* Evitar que fallas o bugs de un entorno afecten a otro.

### Entornos verdaderamente independientes

* Uno puede usar un stack diferente (versiones, extensiones, plugins).
* Permite reiniciar todo un entorno sin afectar al resto.

### Escalabilidad

* A medida que crece la organización, se pueden distribuir cargas por clústeres.

### ¿Y cuándo usar solo Namespaces?

* Para separar microservicios dentro del mismo producto o app.
* Para dividir equipos que colaboran en un mismo entorno.
* Para entornos como dev/test/staging en una misma infraestructura.

### Conclusión

* **Namespaces = separación lógica ligera, flexible, ideal para desarrollo y pruebas.**
* **Clusters = separación física/logística total, ideal para producción, alta seguridad, multicliente, etc.**