



# To-Do App – Conteinerización y Orquestación (Compose · Swarm · Kubernetes KIND)

**Alumno:** Elmer Alan Cornejo Quito

**Usuario Docker Hub:** [alancor](#)

**Repositorio:** [AlanCornejoQ/todo-app](#)

## 1 Descripción

Aplicación **To-Do** modular con seis servicios:

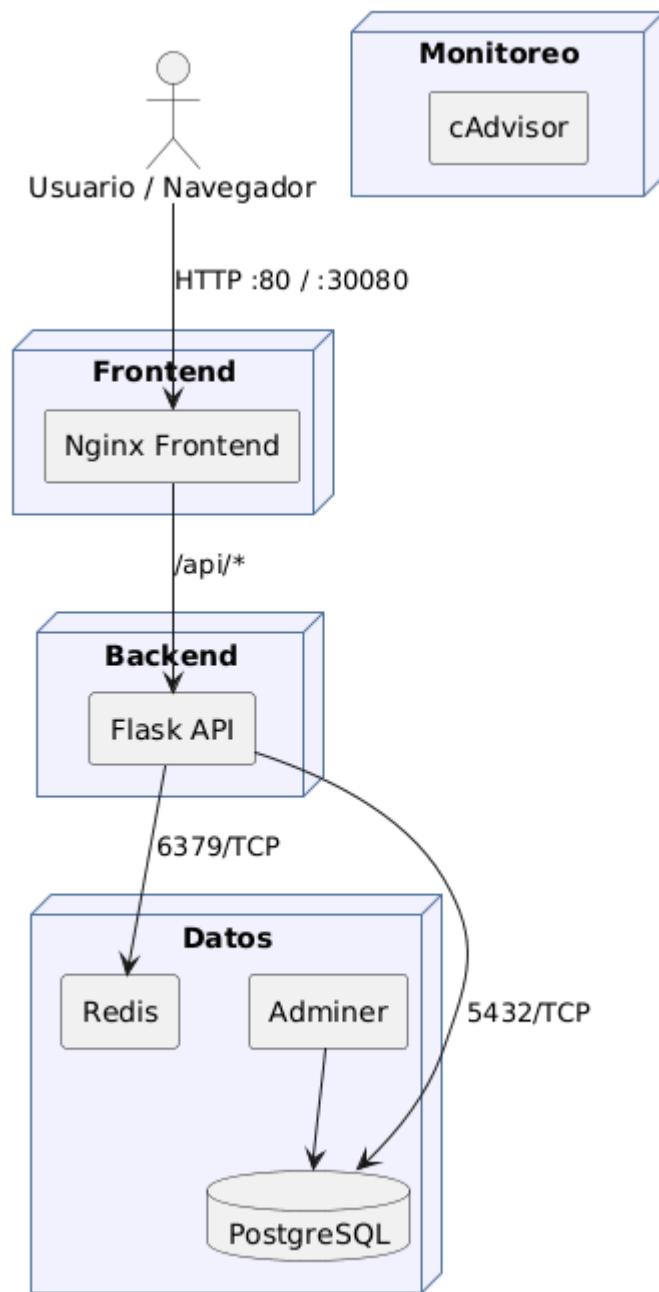
Servicio	Rol	Imagen
frontend	UI Nginx (HTML + JS)	<a href="#">alancor/todo-frontend:v1.1.0</a>
api	Backend Flask + Gunicorn	<a href="#">alancor/todo-api:v1.1.0</a>
db	Base de datos PostgreSQL 16-alpine	<a href="#">postgres:16-alpine</a>
redis	Cache/mensajería	<a href="#">redis:7-alpine</a>
adminer	GUI de BD	<a href="#">adminer:latest</a>
cadvisor	Monitoreo de contenedores (Compose)	<a href="#">gcr.io/cadvisor/cadvisor:v0.49.1</a>

Repositorio público de imágenes en Docker Hub (namespace **alancor**):

The screenshot shows the Docker Hub profile of user **Elmer Alan Cornejo Quito**. The profile page includes a blue circular icon with a white letter 'A', the user's name, their status as a 'Community User', and links for 'Repositories' and 'Starred'. Below this, there is a search bar and a message indicating 1 to 2 repositories. Two repositories are listed: **alancor/todo-frontend** and **alancor/todo-api**. Each repository card displays its image, name, owner, pull requests, stars, and last updated time.

Repository	Owner	Pulls	Stars	Last Updated
<a href="#">alancor/todo-frontend</a>	alancor	33	0	about 7 hours
<a href="#">alancor/todo-api</a>	alancor	43	0	about 7 hours

## 2 Arquitectura



- Red interna entre servicios (app-net)
- Variables de entorno para credenciales y conexiones
- Volumen persistente para la base de datos

## 3 Repositorio público de imágenes

Imagen	Versión	Descripción
alancor/todo-api:v1.0.0	Versión inicial	CRUD de tareas básicas
alancor/todo-api:v1.1.0	Actualización	Endpoint <b>/</b> con información de versión
alancor/todo-frontend:v1.0.0	Versión inicial	Interfaz web To-Do

Imagen	Versión	Descripción
alancor/todo-frontend:v1.1.0	Actualización	Encabezado "To-Do v1.1.0"

Imágenes personalizadas construidas con Dockerfiles propios (base **Alpine**), optimizadas con **.dockerignore** y publicadas en el repositorio público **Docker Hub**.  
Cumple los requerimientos de **conteinerización, versionamiento y repositorio público**.

## 4 Despliegue local – *Docker Compose*

```
docker compose build
docker compose up -d
docker compose ps
```

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ docker compose build
WARN[0000] /home/alan/Documents/maestria/todo-app/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please re
move it to avoid potential confusion
[+] Building 1.2s (24/24) FINISHED
=> [internal] load local bake definitions
=> => reading from stdin 0.01k
=> [api internal] load build definition from Dockerfile
=> => transferring dockerfile: 331B
=> [frontend internal] load build definition from Dockerfile
=> => transferring dockerfile: 1638
=> [frontend internal] load metadata for docker.io/library/nginx:alpine
=> [api internal] load metadata for docker.io/library/python:3.12-alpine
=> [auth] library/python:pull token for registry-1.docker.io
=> [auth] library/nginx:pull token for registry-1.docker.io
=> [api internal] load .dockerignore
=> => transferring context: 928
=> [frontend internal] load .dockerignore
=> => transferring context: 928
=> [api 1/6] FROM docker.io/library/python:3.12-alpine@sha256:d82291d418d5c47f267708393e40599ae836f2260b0519dd38670e9d281657f5
=> [api internal] load build context
=> => transferring context: 638
=> [frontend 1/3] FROM docker.io/library/nginx:alpine@sha256:61e01287e546aac28a3f56839c136b31f590273f3b41187a36f46f6a03bbfe22
=> [frontend internal] load build context
=> => transferring context: 648
=> CACHED [api 2/6] WORKDIR /app
=> CACHED [api 3/6] RUN apk add --no-cache build-base postgresql-dev
=> CACHED [api 4/6] COPY requirements.txt .
=> CACHED [api 5/6] RUN pip install --no-cache-dir -r requirements.txt
=> CACHED [api 6/6] COPY app.py .
=> [api] exporting to image
=> => exporting layers
=> => writing image sha256:0331ce16e135349e8313d0e92cdab2a836f15c5661050cc5a2d34b8dc1e3125b
=> => naming to docker.io/alan/todo-api:v1.0.0
=> CACHED [frontend 2/3] COPY index.html /usr/share/nginx/html/index.html
=> CACHED [frontend 3/3] COPY default.conf /etc/nginx/conf.d/default.conf
=> [frontend] exporting to image
=> => exporting layers
=> => writing image sha256:91408d2ce4a412403dd48dc8ff6ad716f02e43f52f1640f8283bba5956636932
=> => naming to docker.io/alan/todo-frontend:v1.0.0
=> [api] resolving provenance for metadata file
=> [frontend] resolving provenance for metadata file
[+] Building 2/2
✓ alan/todo-api:v1.0.0      Built
✓ alan/todo-frontend:v1.0.0   Built
(base) alan@alanC9:~/Documents/maestria/todo-app$ 
```

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ docker compose up -d
WARN[0000] /home/alan/Documents/maestria/todo-app/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please re
move it to avoid potential confusion
[+] Running 6/6
✓ Container todo-app-redis-1    Running
✓ Container todo-app-api-1     Running
✓ Container todo-app-db-1      Running
✓ Container todo-app-cadvisor-1 Running
✓ Container todo-app-frontend-1 Running
✓ Container todo-app-adminer-1  Started
(base) alan@alanC9:~/Documents/maestria/todo-app$ 
```

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ docker compose ps
WARN[0000] /home/alan/Documents/maestria/todo-app/docker-compose.yml: the attribute 'version' is obsolete, it will be ignored, please remove it to avoid potential confusion
NAME           IMAGE          COMMAND            SERVICE        CREATED         STATUS          PORTS
todo-app-adminer-1  adminer:latest  "/entrypoint.sh dock..."  adminer  2 minutes ago  Up About a minute  0.0.0.0:8081->8080/tcp, [::]:8081->8080/tcp
todo-app-api-1    alan/todo-api:v1.0.0  "gunicorn -w 2 -b 0..."  api    2 minutes ago  Up 2 minutes   8000/tcp
todo-app-cadvisor-1  gcr.io/cadvisor/cadvisor:v0.49.1  "/usr/bin/cadvisor ..."  cadvisor  2 minutes ago  Up 2 minutes (healthy)  0.0.0.0:8082->8080/tcp, [::]:8082->8080/tcp
todo-app-db-1      postgres:16-alpine  "/docker-entrypoint..."  db     2 minutes ago  Up 2 minutes   5432/tcp
todo-app-frontend-1  alan/todo-frontend:v1.0.0   "/docker-entrypoint..."  frontend  2 minutes ago  Up 2 minutes   0.0.0.0:8080->80/tcp, [::]:8080->80/tcp
todo-app-redis-1   redis:7-alpine    "/docker-entrypoint..."  redis   2 minutes ago  Up 2 minutes   6379/tcp
```

## Accesos

- Frontend → <http://localhost:8080>
- Adminer → <http://localhost:8081>
- cAdvisor → <http://localhost:8082>

Se levanta toda la solución en red personalizada `app-net`, con volumen persistente `dbdata` y variables desde `.env`.

## 5 Despliegue en *Docker Swarm*

```
docker swarm init
printf "todosecret" | docker secret create db_password -
printf "production" | docker config create app_env -
docker stack deploy -c stack-deploy.yml todoapp
docker stack services todoapp
docker service ps todoapp_api
```

### Salida relevante

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ docker stack services todoapp
ID          NAME      MODE      REPLICAS  IMAGE
znjrrri24bhtu  todoapp_adminer  replicated  1/1      adminer:latest
3ro4xkqlsn7w   todoapp_api    replicated  2/2      alancor/todo-api:v1.1.0
vyhcbm552rzd   todoapp_db     replicated  1/1      postgres:16-alpine
yn38nsf9zhmz   todoapp_frontend  replicated  2/2      alancor/todo-frontend:v1.1.0
te7t9fw28gra   todoapp_redis   replicated  1/1      redis:7-alpine
(base) alan@alanC9:~/Documents/maestria/todo-app$ 
(base) alan@alanC9:~/Documents/maestria/todo-app$ docker service ps todoapp_api
ID          NAME      IMAGE      NODE      DESIRED STATE  CURRENT STATE      ERROR      PORTS
ktersdxclcti  todoapp_api.1  alancor/todo-api:v1.1.0  alanC9  Running
vmf5akla4lu2  \_ todoapp_api.1  alan/todo-api:v1.0.0  alanC9  Shutdown
jt24ej8jpvj   todoapp_api.2  alancor/todo-api:v1.1.0  alanC9  Running
qml9nzorio4h  \_ todoapp_api.2  alan/todo-api:v1.0.0  alanC9  Shutdown
(base) alan@alanC9:~/Documents/maestria/todo-app$ 
```

Clúster Swarm inicializado correctamente, con **rélicas, overlay network, configs y secrets**.

Cumple el requisito de *Despliegue en Swarm con servicios replicados*.

## 6 Despliegue en *Kubernetes (KIND)*

Crear clúster con puerto NodePort mapeado

```
kind: Cluster
apiVersion: kind.x-k8s.io/v1alpha4
nodes:
  - role: control-plane
    extraPortMappings:
      - containerPort: 30080
        hostPort: 30080
        protocol: TCP
```

```
kind create cluster --name todo-cluster --config kind-config.yaml
```

## Aplicar manifiestos

```
kubectl apply -f k8s/namespace.yaml
kubectl -n todo apply -f k8s/secret.yaml -f k8s/configmap.yaml \
-f k8s/db.yaml -f k8s/redis.yaml -f k8s/api.yaml -f k8s/frontend.yaml
kubectl -n todo get pods,svc,deploy -o wide
```

## Resultado observado

### Pods y servicios activos en el clúster:

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ kubectl -n todo get pods,svc,deploy -o wide
NAME          READY   STATUS    RESTARTS   AGE      IP           NODE   NOMINATED-NODE   READINESS   GATES
pod/api-74bfbfb846-2x2c9   1/1     Running   0          6h14m   10.244.0.13   todo-cluster-control-plane   <none>        <none>
pod/api-74bfbfb846-7hq4s   1/1     Running   0          6h15m   10.244.0.12   todo-cluster-control-plane   <none>        <none>
pod/db-69dbbdd89f-nvv9r   1/1     Running   0          6h20m   10.244.0.11   todo-cluster-control-plane   <none>        <none>
pod/frontend-7fd9769f89-bdc89  1/1     Running   0          6h12m   10.244.0.15   todo-cluster-control-plane   <none>        <none>
pod/frontend-7fd9769f89-zgdgf  1/1     Running   0          6h12m   10.244.0.14   todo-cluster-control-plane   <none>        <none>
pod/redis-79d9b8c76d-rrms6   1/1     Running   0          6h20m   10.244.0.5    todo-cluster-control-plane   <none>        <none>

NAME            TYPE        CLUSTER-IP      EXTERNAL-IP    PORT(S)        AGE      SELECTOR
service/api     ClusterIP   10.96.90.214   <none>        8000/TCP     6h20m   app=api
service/db      ClusterIP   10.96.44.226   <none>        5432/TCP     6h20m   app=db
service/frontend NodePort    10.96.82.40    <none>        80:30080/TCP  6h20m   app=frontend
service/redis    ClusterIP   10.96.15.223   <none>        6379/TCP     6h20m   app=redis

NAME          READY   UP-TO-DATE   AVAILABLE   AGE      CONTAINERS   IMAGES          SELECTOR
deployment.apps/api  2/2      2           2           6h20m   api          alancor/todo-api:v1.1.0   app=api
deployment.apps/db   1/1      1           1           6h20m   postgres     postgres:16-alpine   app=db
deployment.apps/frontend  2/2      2           2           6h20m   frontend    alancor/todo-frontend:v1.1.0  app=frontend
deployment.apps/redis  1/1      1           1           6h20m   redis        redis:7-alpine   app=redis
(base) alan@alanC9:~/Documents/maestria/todo-app$
```

### Detalle del despliegue de la API con versión actualizada:

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ kubectl -n todo describe deploy api
Name:           api
Namespace:      todo
CreationTimestamp: Sun, 26 Oct 2025 10:58:04 -0400
Labels:          <none>
Annotations:    deployment.kubernetes.io/revision: 2
Selector:        app=api
Replicas:       2 desired | 2 updated | 2 total | 2 available | 0 unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=api
           version=v1.0.0
  Containers:
    api:
      Image:      alancor/todo-api:v1.1.0
      Port:       8000/TCP
      Host Port:  0/TCP
      Environment:
        POSTGRES_DB:      <set to the key 'POSTGRES_DB' of config map 'app-config'>  Optional: false
        POSTGRES_USER:    <set to the key 'POSTGRES_USER' of config map 'app-config'>  Optional: false
        POSTGRES_PASSWORD: <set to the key 'POSTGRES_PASSWORD' in secret 'db-secret'>  Optional: false
        POSTGRES_HOST:    <set to the key 'POSTGRES_HOST' of config map 'app-config'>  Optional: false
        POSTGRES_PORT:    <set to the key 'POSTGRES_PORT' of config map 'app-config'>  Optional: false
        REDIS_HOST:       <set to the key 'REDIS_HOST' of config map 'app-config'>  Optional: false
        REDIS_PORT:       <set to the key 'REDIS_PORT' of config map 'app-config'>  Optional: false
      Mounts:          <none>
      Volumes:         <none>
      Node-Selectors:  <none>
      Tolerations:    <none>
    Conditions:
      Type     Status  Reason
      ----
      Available  True    MinimumReplicasAvailable
      Progressing  True    NewReplicaSetAvailable
    OldReplicaSets:  api-946878d67 (0/0 replicas created)
    NewReplicaSet:   api-74bfefb846 (2/2 replicas created)
    Events:          <none>
(base) alan@alanC9:~/Documents/maestria/todo-app$ 
```

```
deployment.apps/api      alancor/todo-api:v1.1.0
deployment.apps/frontend  alancor/todo-frontend:v1.1.0
service/frontend  NodePort 80:30080/TCP
```

**Acceso:** <http://localhost:30080>

### Prueba funcional

```
curl http://localhost:30080/api/tasks
```

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ curl http://localhost:30080/api/tasks
[{"done":false,"id":1,"title":"tarea"}, {"done":false,"id":2,"title":"Guia latex UCB"}, {"done":false,"id":3,"title":"proyecto maestria"}]
(base) alan@alanC9:~/Documents/maestria/todo-app$ 
```

- Se verificó el correcto funcionamiento del clúster KIND y la exposición de servicios NodePort.
- Cumple Despliegue en Kubernetes (KIND).

## 7 Versionamiento y *Rolling Update* en K8s

### API (Flask – Backend)

```
# Construcción y push de nueva versión
docker build -t alancor/todo-api:v1.1.0 api
docker push alancor/todo-api:v1.1.0

# Actualización en el clúster (rolling update)
kubectl -n todo set image deployment/api api=alancor/todo-api:v1.1.0
kubectl -n todo rollout status deployment/api
kubectl -n todo describe deploy api | grep -i Image
```

## Evidencia

**General Tags Image Management BETA Collaborators Webhooks Settings**

Sort by Newest  Filter tags

TAG	Digest	OS/ARCH	Last pull	Compressed size
v1.1.0	<a href="#">1409bcd98a8</a>	linux/amd64	less than 1 day	258.2 MB

TAG	Digest	OS/ARCH	Last pull	Compressed size
v1.0.0	<a href="#">0555ecee8a30</a>	linux/amd64	less than 1 day	258.2 MB

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ kubectl -n todo describe deploy api | grep -i Image
  Image:    alancor/todo-api:v1.1.0
(base) alan@alanC9:~/Documents/maestria/todo-app$ kubectl -n todo rollout status deployment/api
deployment "api" successfully rolled out
```

Image: alancor/todo-api:v1.1.0

- La API se actualizó correctamente, manteniendo disponibilidad y reflejando la nueva versión

## Frontend (Nginx – Interfaz web)

```
# Construcción y push de nueva versión
docker build -t alancor/todo-frontend:v1.1.0 frontend
docker push alancor/todo-frontend:v1.1.0
```

```
# Actualización en el clúster (rolling update)
kubectl -n todo set image deployment/frontend frontend=alancor/todo-frontend:v1.1.0
kubectl -n todo rollout status deployment/frontend
kubectl -n todo describe deploy frontend | grep -i Image
```

## Evidencia

**Tags**

TAG	Digest	OS/ARCH	Last pull	Compressed size
v1.1.0	<a href="#">36099560cf61</a>	linux/amd64	less than 1 day	21.53 MB

TAG	Digest	OS/ARCH	Last pull	Compressed size
v1.0.0	<a href="#">fdb6d3c90288</a>	linux/amd64	less than 1 day	21.53 MB

```
(base) alan@alanC9:~/Documents/maestria/todo-app$ kubectl -n todo describe deploy frontend | grep -i Image
  Image: alancor/todo-frontend:v1.1.0
(base) alan@alanC9:~/Documents/maestria/todo-app$ kubectl -n todo rollout status deployment/frontend
deployment "frontend" successfully rolled out
```

Image: alancor/todo-frontend:v1.1.0

- El frontend se actualizó satisfactoriamente, mostrando el encabezado “To-Do v1.1.0” en la interfaz.
- Se realizó sin interrupción del servicio, verificando la correcta aplicación del rolling update en Kubernetes.
- Cumple y supera el requisito de versionamiento mínimo de 2 versiones, aplicando la actualización controlada en dos servicios (API y Frontend) dentro del clúster KIND

**Antes (v1.0.0)\***

**Después (v1.1.0)**

**Antes (v1.0.0)\*****Después (v1.1.0)****To-Do v1.1.0**

(Interfaz original sin indicador de versión)

Agregar

- #1 tarea
- #2 Guia latex UCB
- #3 proyecto maestria

**8 Verificación final**

Prueba	Comando / URL	Resultado
Listar pods y servicios	<code>kubectl -n todo get pods,svc,deploy -o wide</code>	Todos los pods <i>Running</i> en clúster KIND
API funcional	<code>curl http://localhost:30080/api/tasks</code>	Devuelve JSON con las tareas registradas
Frontend actualizado	<code>http://localhost:30080</code>	Interfaz visible con título “To-Do v1.1.0”
Verificar imágenes desplegadas	<code>kubectl -n todo describe deploy api \  grep -i Image y kubectl -n todo describe deploy frontend \  grep -i Image</code>	Muestran <code>alancor/todo-api:v1.1.0</code> y <code>alancor/todo-frontend:v1.1.0</code>
Swarm status	<code>docker stack services todoapp</code>	Servicios replicados y en estado <i>Running</i> ( <code>alancor/*:v1.1.0</code> )

Las pruebas confirman el funcionamiento completo de la aplicación en los tres entornos: **Compose, Swarm y Kubernetes (KIND)**.

Se validó la comunicación entre servicios, el correcto enrutamiento interno (frontend ↔ api ↔ db ↔ redis) y la persistencia de datos en PostgreSQL.

**9 Evaluación trazada**

Criterio oficial	Evidencia en este proyecto
Arquitectura	Seis servicios independientes: frontend, api, db, redis, adminer, cAdvisor; uso de red interna y variables de entorno
Contenedorización	Dockerfiles livianos (Alpine), <code>.dockerignore</code> , sin credenciales, etiquetas versionadas ( <code>v1.0.0</code> y <code>v1.1.0</code> )

Criterio oficial	Evidencia en este proyecto
<b>Compose / Swarm</b>	<code>docker-compose.yml</code> y <code>stack-deploy.yml</code> con réplicas, configs y secrets; validado con <code>docker stack services</code>
<b>Kubernetes</b>	Manifiestos YAML completos (Namespace, Secret, ConfigMap, PVC, Deployments, Services NodePort); clúster KIND funcional
<b>Documentación</b>	Este README con pasos reproducibles, evidencias de ejecución y capturas de verificación

Se cumplen **todos los criterios de evaluación** establecidos en la guía del examen final, incluyendo versionamiento, orquestación y documentación técnica reproducible.

## Conclusión

El proyecto demuestra la migración completa **del desarrollo local a la orquestación en clúster**, manteniendo la misma arquitectura modular y escalable.

Se lograron todos los objetivos específicos del examen:

- **Imágenes personalizadas y públicas** en Docker Hub (`alancor/*`)
- **Versionamiento mínimo y rolling updates** aplicados en **API** y **Frontend**
- **Despliegue funcional** en *Docker Compose, Docker Swarm y Kubernetes (KIND)*
- **Documentación integral y reproducible**, evidenciando el ciclo completo de containerización y orquestación
- **Código Fuente** publicados en el repositorio público de GitHub: [AlanCornejoQ/todo-app](https://github.com/AlanCornejoQ/todo-app)