

ToDo App - Proyecto Final de Containerizacion y Orquestacion

Sistema de gestion de tareas (ToDo List) completamente containerizado y orquestado usando Docker y Kubernetes (K3D).

Autor: Wilver Vargas

Tecnologia: Docker Compose, Docker Swarm, Kubernetes (K3D)

Tabla de Contenidos

1. [Descripcion del Proyecto](#)
2. [Arquitectura del Sistema](#)
3. [Requisitos Previos](#)
4. [Instalacion](#)
5. [Despliegue](#)
6. [Verificacion y Pruebas](#)
7. [Acceso a la Aplicacion](#)
8. [Comandos Utiles](#)
9. [Troubleshooting](#)
10. [Limpieza](#)

Descripcion del Proyecto

Arquitectura de microservicios con 6 servicios independientes y completamente funcionales:

- **Frontend:** Interfaz web con Nginx y React
- **Backend API:** API REST con Node.js/Express
- **Worker:** Servicio de procesamiento background con Node.js
- **MongoDB:** Base de datos NoSQL
- **Redis:** Cache y cola de mensajes
- **Nginx:** Reverse proxy y load balancer

Caracteristicas Principales

- Containerizacion completa con Docker y Alpine Linux
- Orquestacion con Docker Compose y Docker Swarm
- Despliegue en Kubernetes (K3D)
- Balanceo de carga con Nginx
- Persistencia de datos con MongoDB
- Cache distribuido con Redis

- Procesamiento background con Worker

Servicios Implementados

Servicio	Tecnologia	Puerto	Descripcion
Frontend	React 18 + Nginx Alpine	3000	Interfaz de usuario web
Backend	Node.js 18 + Express	5000	API REST para gestion de tareas
MongoDB	MongoDB 7	27017	Base de datos NoSQL
Redis	Redis 7 Alpine	6379	Cache y almacenamiento temporal
Nginx	Nginx Alpine	80	Reverse proxy y load balancer
Worker	Node.js 18	N/A	Procesamiento en background

Arquitectura del Sistema

Componentes por Capa

Capa	Servicio	Tecnologia	Replicas
Load Balancer	Nginx	nginx:alpine	2
Frontend	Web UI	nginx:alpine	2
Backend	REST API	node:18-alpine	3
Worker	Background	node:18-alpine	2
Database	MongoDB	mongo:7-jammy	1
Cache	Redis	redis:7-alpine	1

Diagrama de Arquitectura

USUARIO (Navegador)

|

```

      v
NGINX (Port 80)
Reverse Proxy & Load Balancer

|

+-----+-----+

v     v     v
FRONTEND BACKEND WORKER
React:3000 Node:5000 Process

|       |       |
+-----+---+   |
      |           |
      v           v
MongoDB       Redis
:27017       :6379

```

Requisitos Previos

Software Necesario

- **Docker** (version 20.10+)
- **Docker Compose** (2.0+)
- **Git**
- **kubectl** (para Kubernetes)
- **K3D** (para Kubernetes)

Recursos del Sistema

- **RAM:** 4GB minimo (8GB recomendado para K3D)
- **CPU:** 2 cores minimo (4 recomendado)
- **Disco:** 10GB libres

- **Red:** Puertos 80, 3000, 5000, 6379, 27017 disponibles

Verificacion de Requisitos

```
# Verificar Docker docker --version # Debe mostrar: Docker version 20.10.0 o superior# Verificar Docker Compose docker compose version # Debe mostrar: Docker Compose version 2.0.0 o superior# Verificar Git git --version
```

Instalacion

Paso 1: Instalar Docker

```
# Actualizar sistema sudo apt-get update # Instalar dependencias sudo apt-get install -y \
ca-certificates \ curl \ gnupg \ lsb-release # Agregar clave GPG de Docker sudo
mkdir -p /etc/apt/keyrings curl -fsSL https://download.docker.com/linux/ubuntu/gpg | \
sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg # Agregar repositorioecho \ "deb
[arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu \ $(lsb_release -cs) stable" | \ sudo tee
/etc/apt/sources.list.d/docker.list > /dev/null # Instalar Docker sudo apt-get update sudo
apt-get install -y docker-ce docker-ce-cli containerd.io docker-compose-plugin # Agregar
usuario al grupo docker sudo usermod -aG docker $USER newgrp docker # Verificar
instalacion docker --version docker compose version
```

Paso 2: Instalar K3D

```
# Descargar e instalar K3D curl -s https://raw.githubusercontent.com/k3d-
io/k3d/main/install.sh | bash # Verificar instalacion k3d version
```

Paso 3: Instalar kubectl

```
# Descargar kubectl curl -LO "https://dl.k8s.io/release/$(curl -L -s
https://dl.k8s.io/release/stable.txt)/bin/linux/amd64/kubectl"# Instalar kubectl sudo install -o root -g
root -m 0755 kubectl /usr/local/bin/kubectl # Limpiar kubectl # Verificar instalacion
kubectl version --client
```

Paso 4: Clonar el Proyecto

```
# Clonar repositorio git clone https://github.com/W-Varg/ucb_containers_app_todo_list.git
cd ucb_containers_app_todo_list # O si ya tienes el proyecto cd /ruta/al/proyecto
```

Estructura del Proyecto

```
ucb-final/
├── backend/           # API Node.js
│   ├── server.js
│   ├── package.json
│   ├── Dockerfile
│   └── .dockerignore
│
├── frontend/         # Aplicacion React
│   ├── public/
│   │   └── index.html
│   ├── src/
│   │   ├── App.js
│   │   ├── App.css
│   │   ├── index.js
│   │   └── index.css
│   ├── package.json
│   ├── Dockerfile
│   └── .dockerignore
│
└── worker/           # Servicio de procesamiento
```

```
|   ├── worker.js
|   ├── package.json
|   ├── Dockerfile
|   └── .dockerignore
|
|── nginx/                # Reverse proxy
|   ├── nginx.conf
|   ├── Dockerfile
|   └── .dockerignore
|
|── mongodb-init/        # Scripts de inicializacion
|   └── init-mongo.js
|
|── kubernetes/          # Manifiestos Kubernetes
|   ├── 00-namespace.yaml
|   ├── 01-secrets-configmap.yaml
|   ├── 02-persistent-volumes.yaml
|   ├── 03-mongodb-deployment.yaml
|   ├── 04-redis-deployment.yaml
|   ├── 05-backend-deployment.yaml
|   ├── 06-worker-deployment.yaml
|   ├── 07-frontend-deployment.yaml
|   ├── 08-nginx-loadbalancer.yaml
|   └── 09-version-2-deployments.yaml
|
```

```
|— k3d/                # Manifiestos K3D
| |— cluster-config.yaml
| |— 00-namespace.yaml
| |— 01-config-secrets.yaml
| |— 02-persistent-volumes.yaml
| |— 03-mongodb.yaml
| |— 04-redis.yaml
| |— 05-backend.yaml
| |— 06-frontend.yaml
| |— 07-worker.yaml
| |— 08-nginx-ingress.yaml
| |— 09-nginx-config.yaml
| |— deploy-k3d.sh
| |— test-k3d.sh
| |— verify-k3d.sh
| |— cleanup-k3d.sh
| |— README-K3D.md
|
|— swarm/              # Configuración Docker Swarm
| |— stack-deploy.yml
| |— stack-simple.yml
| |— README-SWARM.md
|
|— docker-compose.yml   # Compose para desarrollo local
|— README.md            # Este archivo
```

```
|— CHANGELOG-v1.2.0.md    # Historial de cambios
|— .gitignore
```

Despliegue

Opcion 1: Docker Compose (5 minutos)

Ideal para desarrollo local y pruebas rapidas.

```
# 1. Construir imagenes docker compose build # 2. Iniciar servicios docker compose up -d #
3. Verificar estado docker compose ps # 4. Ver logs docker compose logs -f backend # 5.
Detener servicios docker compose down # 6. Limpiar volumen docker compose down -v
```

Opcion 2: Docker Swarm (10 minutos)

Despliegue en modo cluster con replicacion.

```
# 1. Inicializar Swarm docker swarm init # 2. Construir imagenes docker compose build # 3.
Desplegar stack simple docker stack deploy -c swarm/stack-simple.yml todoapp # 4.
desplegar stack completo con versionamiento docker stack deploy -c swarm/stack-deploy.yml
todoapp # 5. Verificar servicios docker service ls # 6. Ver logs de un servicio docker service
logs todoapp_backend # 7. Remover stack docker stack rm todoapp
```

Opcion 3: Kubernetes con K3D (15 minutos)

Despliegue automatico completo.

```
# 1. Instalar herramientas (solo primera vez) curl -s https://raw.githubusercontent.com/k3d-
io/k3d/main/install.sh | bash sudo snap install kubectl --classic # 2. Crear cluster k3d
cluster create --config k3d/cluster-config.yaml # 3. Cambiar contexto kubectl kubectl config
use-context k3d-todo-cluster # 4. Construir imagenes docker compose build # 5. Importar
imagenes al cluster k3d image import \    todo-backend:1.2.0 \    todo-frontend:1.2.0 \
todo-worker:1.2.0 \    todo-nginx:1.2.0 \    -c todo-cluster # 6. Desplegar aplicacion
chmod +x k3d/deploy-k3d.sh ./k3d/deploy-k3d.sh # 7. Verificar despliegue kubectl get all -n todo-
app kubectl get pods -n todo-app kubectl get services -n todo-app # 8. Ver logs kubectl
logs -f deployment/backend -n todo-app # 9. Eliminar cluster k3d cluster delete todo-
cluster
```


Opcion 4: Despliegue Kubernetes Manual

```
# 1. Crear namespace kubectl apply -f kubernetes/00-namespace.yaml # 2. Crear secrets y
configmaps kubectl apply -f kubernetes/01-secrets-configmap.yaml # 3. Crear volúmenes
persistentes kubectl apply -f kubernetes/02-persistent-volumes.yaml # 4. Desplegar bases
de datos kubectl apply -f kubernetes/03-mongodb-deployment.yaml kubectl apply -f
kubernetes/04-redis-deployment.yaml # 5. Desplegar aplicación kubectl apply -f
kubernetes/05-backend-deployment.yaml kubectl apply -f kubernetes/06-worker-
deployment.yaml kubectl apply -f kubernetes/07-frontend-deployment.yaml # 6.
Configurar load balancer kubectl apply -f kubernetes/08-nginx-loadbalancer.yaml # 7.
Verificar despliegue kubectl get all -n todoapp
```

Verificacion y Pruebas

Docker Compose

```
# Verificar estado de contenedores docker compose ps # Verificar logs docker compose logs
backend docker compose logs frontend # Prueba de API curl http://localhost:5000/health
curl http://localhost:5000/api/tasks # Acceso a bases de datos docker exec -it todo-
mongodb mongosh docker exec -it todo-redis redis-cli
```

Docker Swarm

```
# Listar servicios docker service ls # Ver estado del servicio docker service ps
todoapp_backend # Ver logs docker service logs todoapp_backend -f # Inspeccionar
servicio docker service inspect todoapp_backend
```

Kubernetes

```
# Ver todos los recursos kubectl get all -n todo-app # Ver pods específicos kubectl get pods -
n todo-app kubectl get pods -n todo-app -w # Ver descripción detallada kubectl describe
pod POD_NAME -n todo-app # Ver logs kubectl logs -f deployment/backend -n todo-app
kubectl logs POD_NAME -n todo-app # Acceder a un pod kubectl exec -it POD_NAME -n
todo-app -- /bin/bash # Ver eventos kubectl get events -n todo-app # Ver servicios y
exponer kubectl get services -n todo-app kubectl port-forward service/backend-service
5000:5000 -n todo-app
```

Acceso a la Aplicacion

URLs Disponibles

Componente	URL	Puerto
Frontend	http://localhost	80
Backend API	http://localhost:5000	5000
MongoDB	localhost	27017
Redis	localhost	6379

Puntos de Acceso por Entorno

Docker Compose

- Frontend: <http://localhost>
- Backend: <http://localhost:5000>
- API Health: <http://localhost:5000/health>

Docker Swarm

- Frontend: <http://localhost>
- Backend: <http://localhost:5000>
- API Health: <http://localhost:5000/health>

Kubernetes/K3D

- Frontend: <http://localhost:9080> (si usa port-forward)
- Backend: <http://localhost:9500> (si usa port-forward)

Test de API

```
# Health check backend curl http://localhost:5000/health # Obtener todas las tareas curl
http://localhost:5000/api/tasks # Crear nueva tarea curl -X POST
http://localhost:5000/api/tasks \ -H "Content-Type: application/json" \ -d '{"title":"Tarea de
prueba","description":"Descripcion"}# Obtener estadisticas curl http://localhost:5000/api/stats #
Actualizar tarea curl -X PUT http://localhost:5000/api/tasks/ID \ -H "Content-Type:
application/json" \ -d '{"completed":true}'# Eliminar tarea curl -X DELETE
http://localhost:5000/api/tasks/ID
```

Comandos Utiles

Docker

```
# Construir imagenes con tags especificos docker build -t kryshor/todo-backend:1.2.0 ./backend docker build -t kryshor/todo-frontend:1.2.0 ./frontend docker build -t kryshor/todo-worker:1.2.0 ./worker docker build -t kryshor/todo-nginx:1.2.0 ./nginx # Subir imagenes a Docker Hub docker push kryshor/todo-backend:1.2.0 docker push kryshor/todo-frontend:1.2.0 docker push kryshor/todo-worker:1.2.0 docker push kryshor/todo-nginx:1.2.0 # Listar imagenes docker images | grep todo # Eliminar imagen docker rmi kryshor/todo-backend:1.2.0 # Limpiar imagenes sin usar docker image prune -a # Ver volumen docker volume ls # Eliminar volumen docker volume prune
```

Docker Compose

```
# Construir imagenes docker compose build # Subir servicios en background docker compose up -d # Ver logs en tiempo real docker compose logs -f # Parar servicios docker compose stop # Reiniciar servicios docker compose restart # Ver estado docker compose ps # Ejecutar comando en contenedor docker compose exec backend sh # Remover todo docker compose down -v
```

Kubernetes/kubectl

```
# Cambiar contexto kubectl config use-context k3d-todo-cluster # Ver contexto actual kubectl config current-context # Listar todos los recursos kubectl get all -n todo-app # Listar pods kubectl get pods -n todo-app # Listar servicios kubectl get services -n todo-app # Listar deployments kubectl get deployments -n todo-app # Ver descripcion detallada kubectl describe pod POD_NAME -n todo-app # Ver logs kubectl logs -f POD_NAME -n todo-app # Acceder a pod kubectl exec -it POD_NAME -n todo-app -- /bin/bash # Port forward kubectl port-forward pod/POD_NAME 5000:5000 -n todo-app # Actualizar imagen kubectl set image deployment/backend backend=kryshor/todo-backend:1.2.0 -n todo-app # Verificar rollout kubectl rollout status deployment/backend -n todo-app # Rollback kubectl rollout undo deployment/backend -n todo-app # Eliminar recursos kubectl delete deployment/backend -n todo-app kubectl delete all -n todo-app
```

K3D

```
# Listar clusters k3d cluster list # Crear cluster k3d cluster create --config k3d/cluster-config.yaml # Eliminar cluster k3d cluster delete todo-cluster # Importar imagenes k3d image import IMAGE_NAME -c todo-cluster # Ver logs cluster k3d logs CLUSTER_NAME # Acceder a nodo master k3d node shell k3d-todo-cluster-server-0
```

Troubleshooting

Docker Compose

Los contenedores no inician

```
# Verificar logs detallados docker compose logs -f # Reconstruir desde cero docker compose down -v docker compose build --no-cache docker compose up -d
```

Puerto ya en uso

```
# Encontrar proceso usando el puerto sudo lsof -i :5000 # Matar proceso sudo kill -9 PID # O cambiar puerto en docker-compose.yml
```

Conexion a MongoDB fallida

```
# Verificar contenedor MongoDB docker compose ps mongodb # Ver logs de MongoDB docker compose logs mongodb # Reconectar docker compose restart mongodb backend
```

Kubernetes

Pod no inicia

```
# Ver descripcion detallada kubectl describe pod POD_NAME -n todo-app # Ver logs kubectl logs -f POD_NAME -n todo-app # Eliminar y recrear kubectl delete pod POD_NAME -n todo-app
```

Servicio no responde

```
# Verificar DNS kubectl run -it --rm debug --image=alpine --restart=Never -- sh nslookup backend-service.todo-app.svc.cluster.local # Verificar conectividad kubectl exec -it POD_NAME -n todo-app -- sh curl http://backend-service:5000/health
```

Problemas de persistencia

```
# Ver PVCs kubectl get pvc -n todo-app # Describir PVC kubectl describe pvc PVC_NAME  
-n todo-app # Ver PVs kubectl get pv
```

K3D

Cluster no crea

```
# Verificar Docker docker ps # Ver logs del cluster k3d logs todo-cluster # Recrear cluster k3d  
cluster delete todo-cluster k3d cluster create --config k3d/cluster-config.yaml
```

Imágenes no importan

```
# Verificar imágenes locales docker images # Construir si no existen docker compose build #  
Importar nuevamente k3d image import IMAGE_NAME -c todo-cluster
```

Limpieza

Limpiar Docker Compose

```
# Detener todo y eliminar volúmenes docker compose down -v # Eliminar imágenes docker  
rmi $(docker images | grep "todo" | awk '{print $3}')
```

Limpiar Docker Swarm

```
# Remover stack docker stack rm todoapp # Dejar el swarm docker swarm leave --force #  
Limpiar imágenes docker rmi $(docker images | grep "todo" | awk '{print $3}')
```

Limpiar Kubernetes/K3D

```
# Eliminar namespace kubectl delete namespace todo-app # Eliminar cluster completo k3d  
cluster delete todo-cluster # Limpiar imágenes docker rmi $(docker images | grep "todo" |  
awk '{print $3}') # Ejecutar script de limpieza chmod +x k3d/cleanup-k3d.sh ./k3d/cleanup-  
k3d.sh
```

Limpiar Todo

```
# Eliminar volúmenes de Docker docker volume prune -f # Eliminar redes de Docker docker network prune -f # Eliminar imágenes huérfanas docker image prune -a -f # Limpiar cache de construcción docker builder prune -a -f
```

Información de Versionamiento

Version Actual: 1.2.0

Última actualización: 29 de Octubre de 2025

Historial de Cambios

- **v1.2.0:** Optimizaciones y mejoras de estabilidad
- **v1.1.0:** Mejoras de rendimiento y nuevas funcionalidades
- **v1.0.0:** Versión inicial del proyecto

Ver CHANGELOG-v1.2.0.md para detalles completos.

Soporte y Documentación

- Problemas conocidos: Ver sección Troubleshooting
- Documentación de K3D: Ver k3d/README-K3D.md
- Documentación de Docker Swarm: Ver swarm/README-SWARM.md
- Comandos de versionamiento: Ver COMANDOS-VERSIONAMIENTO.txt

Licencia

MIT

Autor

Wilver Vargas

UCB - Proyecto Final de Containerización y Orquestación

Nota: Este README ha sido optimizado y reorganizado para mejor legibilidad y mantenimiento. Todos los emojis y caracteres especiales han sido removidos.

