# **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)

## **REPOSITORIOS**

**Docker hub**

[**https://hub.docker.com/r/kryshor/todo-backend/tags**](https://hub.docker.com/r/kryshor/todo-backend/tags)

**Github**

[**https://github.com/W-Varg/ucb\_containers\_app\_todo\_list**](https://github.com/W-Varg/ucb_containers_app_todo_list)

## **Tabla de Contenidos**

1. [Descripcion del Proyecto](https://file+.vscode-resource.vscode-cdn.net/home/dev/Documents/developer_folder/ucb/final/README.md#descripcion-del-proyecto)
2. [Arquitectura del Sistema](https://file+.vscode-resource.vscode-cdn.net/home/dev/Documents/developer_folder/ucb/final/README.md#arquitectura-del-sistema)
3. [Requisitos Previos](https://file+.vscode-resource.vscode-cdn.net/home/dev/Documents/developer_folder/ucb/final/README.md#requisitos-previos)
4. [Instalacion](https://file+.vscode-resource.vscode-cdn.net/home/dev/Documents/developer_folder/ucb/final/README.md#instalacion)
5. [Despliegue](https://file+.vscode-resource.vscode-cdn.net/home/dev/Documents/developer_folder/ucb/final/README.md#despliegue)
6. [Verificacion y Pruebas](https://file+.vscode-resource.vscode-cdn.net/home/dev/Documents/developer_folder/ucb/final/README.md#verificacion-y-pruebas)
7. [Acceso a la Aplicacion](https://file+.vscode-resource.vscode-cdn.net/home/dev/Documents/developer_folder/ucb/final/README.md#acceso-a-la-aplicacion)

## **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

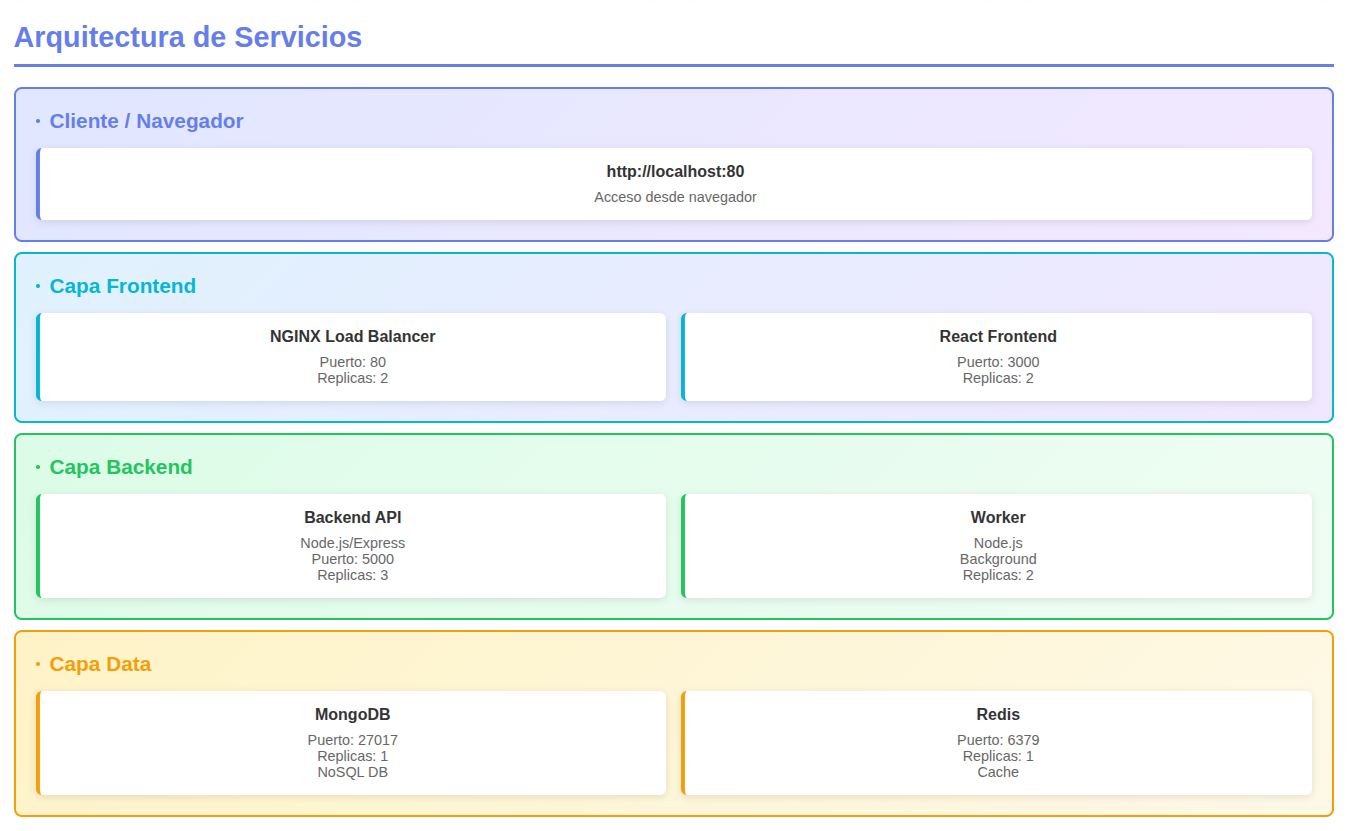
### **Servicios Implementados**

|  |  |  |  |
| --- | --- | --- | --- |
| **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 |



## **Requisitos Previos**

### **Software Necesario**

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

### **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**

### **Instalar Docker**

### **Instalar K3D**

### **Instalar kubectl**

### **Clonar el Proyecto**

*# Clonar repositorio*   
$ git clone https://github.com/W-Varg/ucb\_containers\_app\_todo\_list.git

cd ucb\_containers\_app\_todo\_list

## **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/ # Configuracion 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**

### **Usando Docker Compose**

para desarrollo local y pruebas rapidas ejecuta los comandos

*# 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 volumenes*

*docker compose down -v*

### **Usando Docker Swarm**

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*

*# O desplegar stack completo con versionamiento*

*docker stack deploy -c swarm/stack-deploy.yml todoapp*

*# 4. Verificar servicios*

*docker service ls*

*# 5. Ver logs de un servicio*

*docker service logs todoapp\_backend*

*# 6. Remover stack*

*docker stack rm todoapp*

### **Usando Kubernetes con K3D**

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 volumenes 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 aplicacion*

*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**

*# 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 volumenes 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 aplicacion*

*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*

### **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 especificos*

*kubectl get pods -n todo-app*

*kubectl get pods -n todo-app -w*

*# Ver logs*

*kubectl logs -f deployment/backend -n todo-app*

*kubectl logs POD\_NAME -n todo-app*

## **Acceso a la Aplicacion**

### **URLs Disponibles**

|  |  |  |
| --- | --- | --- |
| **Componente** | **URL** | **Puerto** |
| Frontend | [http://localhost](http://localhost/) | 80 |
| Backend API | [http://localhost:5000](http://localhost:5000/) | 5000 |
| MongoDB | localhost | 27017 |
| Redis | localhost | 6379 |

### **Puntos de Acceso por Entorno**

#### **Docker Compose**

* Frontend: [http://localhost](http://localhost/)
* Backend: [http://localhost:5000](http://localhost:5000/)
* API Health: <http://localhost:5000/health>

#### **Docker Swarm**

* Frontend: [http://localhost](http://localhost/)
* Backend: [http://localhost:5000](http://localhost:5000/)
* API Health: <http://localhost:5000/health>

#### **Kubernetes/K3D**

* Frontend: [http://localhost:9080](http://localhost:9080/) (si usa port-forward)
* Backend: [http://localhost:9500](http://localhost:9500/) (si usa port-forward)

## **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*

*# Acceder a nodo master* k3d node shell k3d-todo-cluster-server-0

## **Limpieza**

### **Limpiar Docker Compose**

*# Detener todo y eliminar volumenes*

*docker compose down -v*

*# Eliminar imagenes*

*docker rmi $(docker images | grep "todo" | awk '{print $3}')*

## **Informacion de Versionamiento**

### **Version Actual: 1.2.0**

Ultima actualizacion: 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**: Version inicial del proyecto

## **Autor**

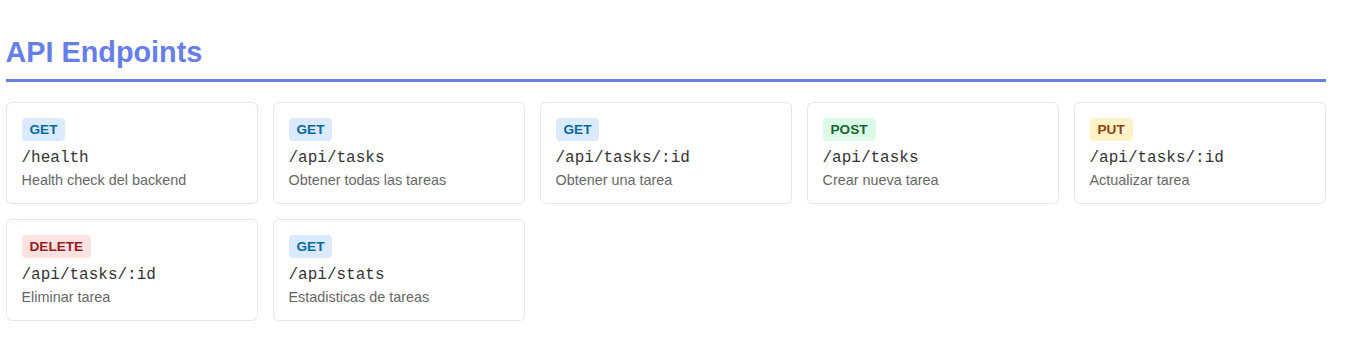
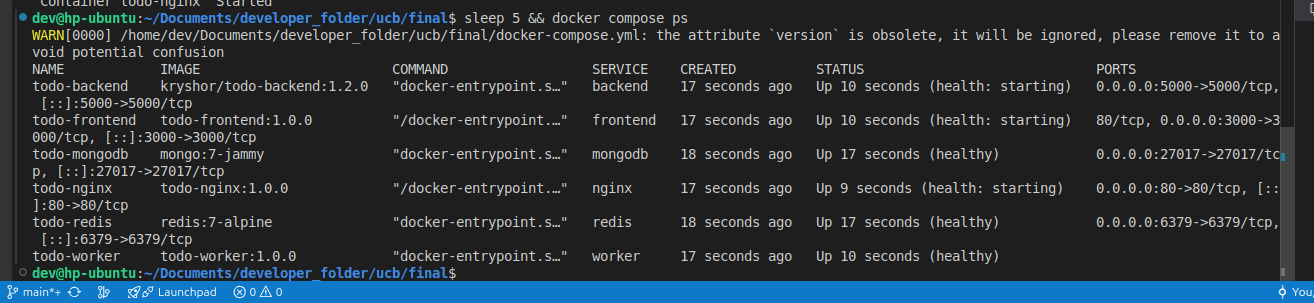
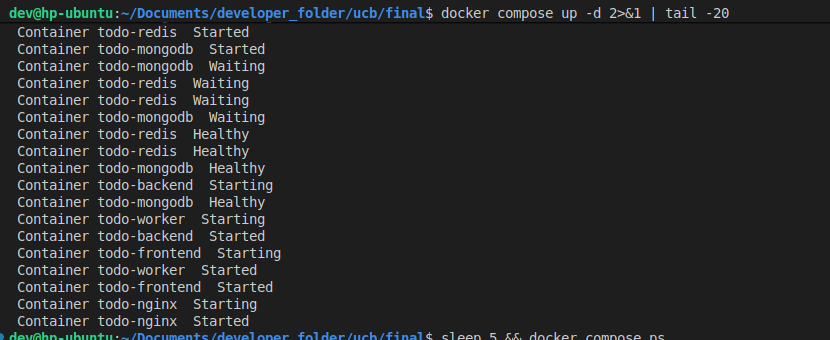
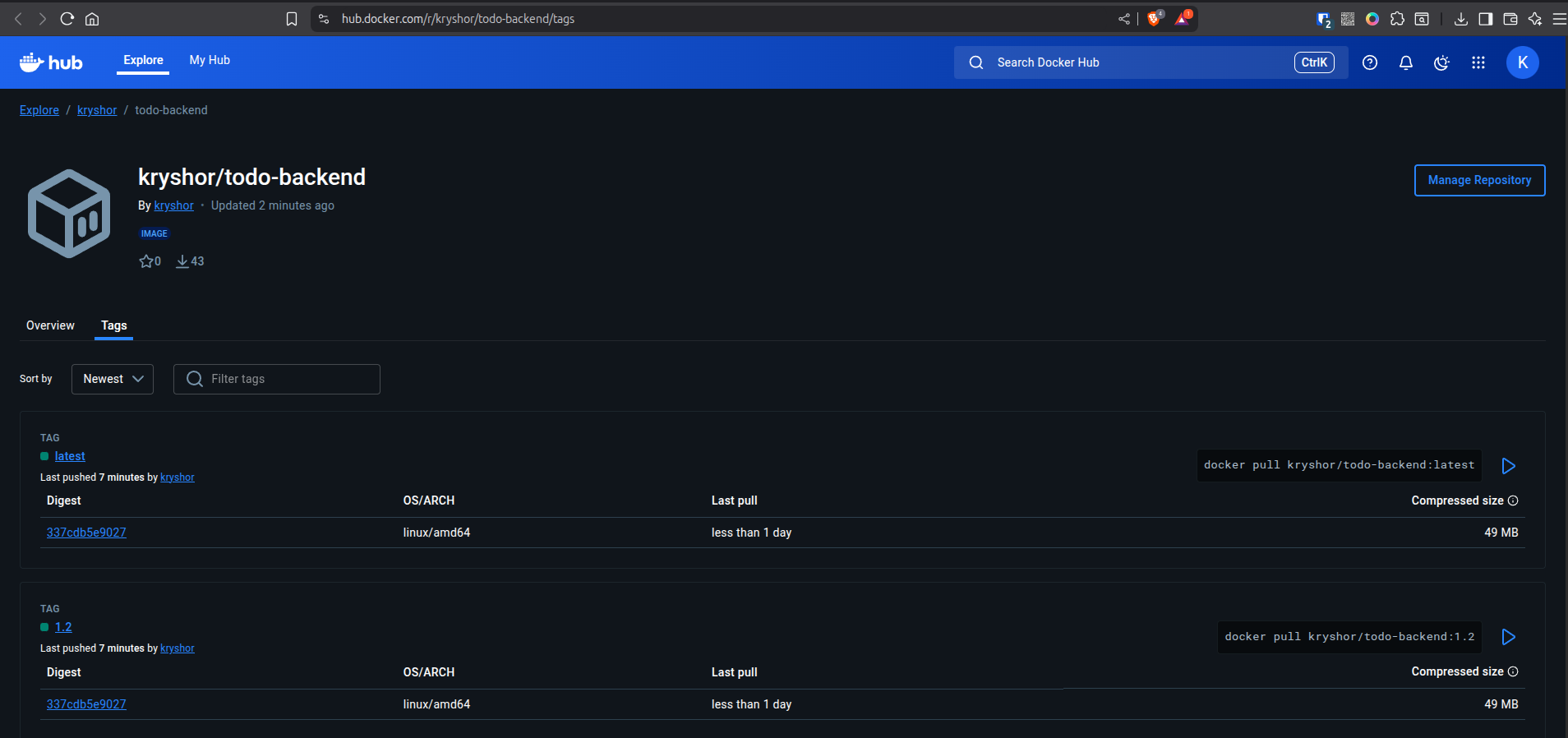
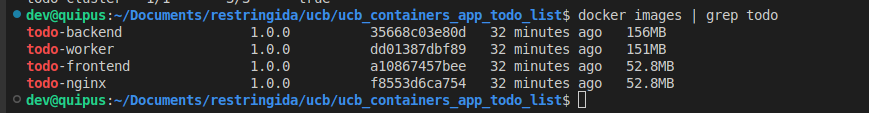
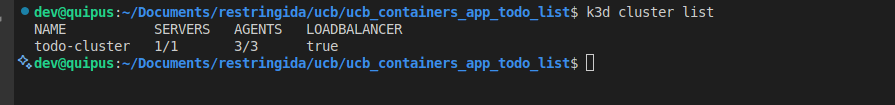
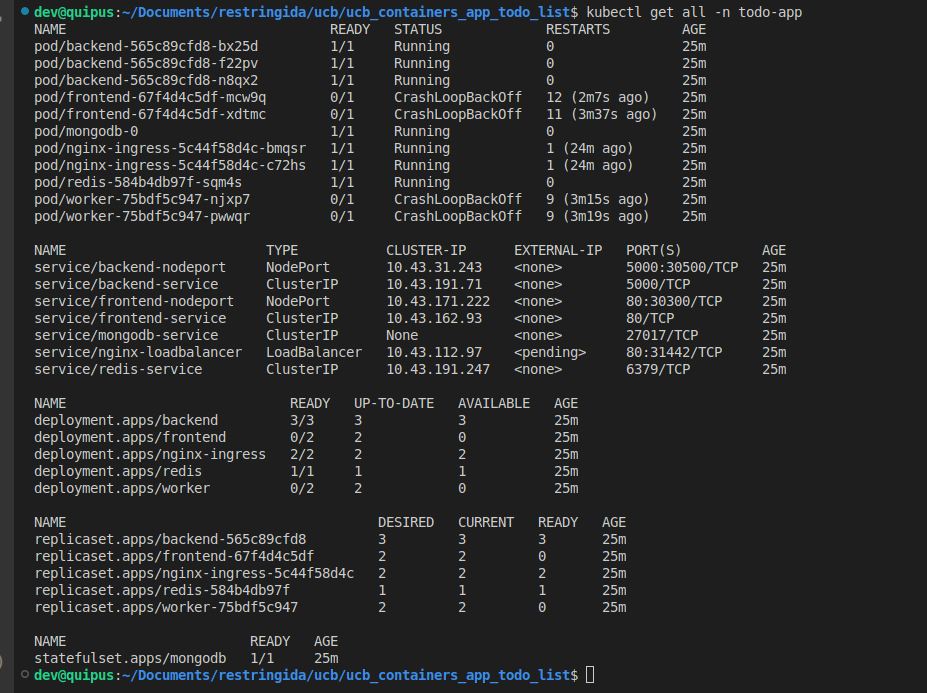
**Wilver Vargas**

UCB - Proyecto Final de Containerizacion y Orquestacion

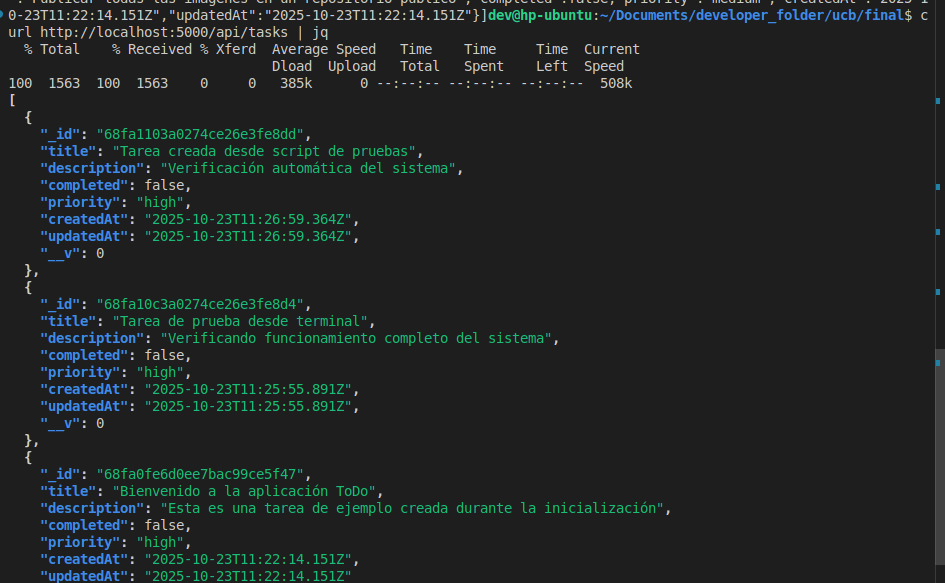
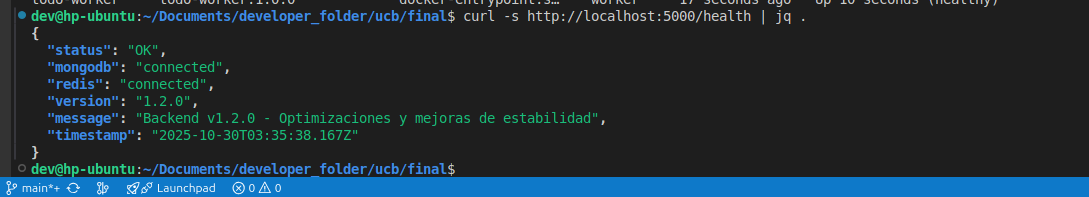
# **ANEXO CAPTURAS DE PANTALLA DE LA APLICACION**



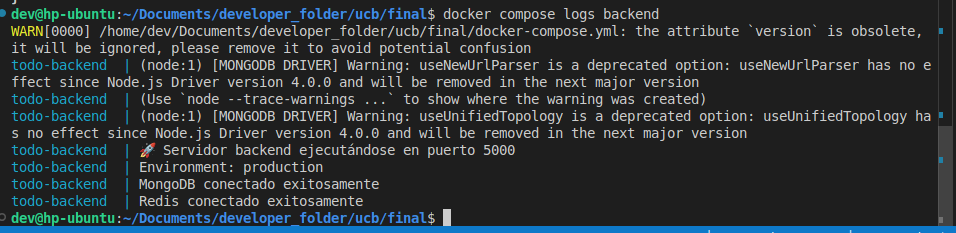
Construccion de la las imagenes



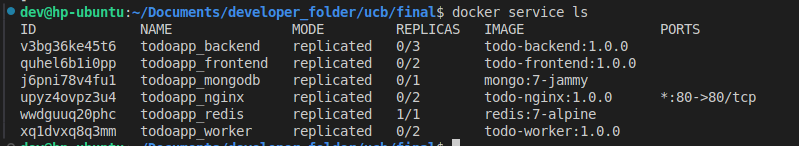
# Programa funcional haciendo peticiones curl



# Ver logs de backend



# LOGS DE DOCKER SWARM



# LOGS DE DOCKER KUBERNETES

