Cesar Renato Cáceres Avendaño

Junio, 2024

Bootcamp DevOps - Codigo Facilito

Proyecto final[2] : tODO-lIST

Contenido

[To-Do-List Application 2](#_Toc168265023)

[Descripción General del Proyecto 2](#_Toc168265024)

[Arquitectura del Proyecto 2](#_Toc168265025)

[**1. Estructura de la Aplicación** 2](#_Toc168265026)

[**2. Contenedores y Orquestación** 2](#_Toc168265027)

[**3. Monitoreo y Gestión** 2](#_Toc168265028)

[Configuración de la Infraestructura 3](#_Toc168265029)

[**1. Docker** 3](#_Toc168265030)

[**2. Kubernetes con Minikube** 4](#_Toc168265031)

[**3. Instalación de Helm** 6](#_Toc168265032)

[Pasos de Implementación de DevOps 6](#_Toc168265033)

[**1. Configuración del Pipeline en Jenkins** 6](#_Toc168265034)

[**2. Integración Continua (CI)** 8](#_Toc168265035)

[**3. Entrega Continua (CD)** 8](#_Toc168265036)

[Prácticas de DevOps Seguidas 8](#_Toc168265037)

**To-Do-List Application**

**Descripción General del Proyecto**

La aplicación **To-Do-List** es una aplicación desarrollada en Node.js que permite a los usuarios gestionar sus tareas diarias. Este proyecto se ha configurado y desplegado utilizando diversas herramientas de DevOps para asegurar un ciclo de vida de desarrollo y despliegue eficiente.

**Arquitectura del Proyecto**

**1. Estructura de la Aplicación**

* **Frontend**: Implementado en Node.js para la gestión de las vistas y la interacción del usuario.
* **Backend**: API RESTful creada con Node.js para gestionar las operaciones CRUD (Crear, Leer, Actualizar, Borrar) de las tareas.
* **Base de Datos**: Puede ser MongoDB o cualquier base de datos NoSQL.

**2. Contenedores y Orquestación**

* **Docker**: Utilizado para contenerizar la aplicación y sus dependencias.
* **Docker Compose**: Utilizado para definir y ejecutar aplicaciones Docker multi-contenedor.
* **Kubernetes**: Utilizado para orquestar los contenedores Docker.
  + **Minikube**: Implementación local de Kubernetes para desarrollo y pruebas.
* **Helm**: Utilizado para gestionar la instalación de aplicaciones dentro de Kubernetes.

**3. Monitoreo y Gestión**

* **Prometheus**: Sistema de monitoreo y alerta.
* **Grafana**: Plataforma de análisis y visualización de métricas.
* **Jenkins**: Servidor de integración y entrega continua.

**Configuración de la Infraestructura**

**1. Docker**

**Dockerfile**

# Utiliza una imagen base oficial de Node.js

FROM node:14

# Establece el directorio de trabajo en /app

WORKDIR /app

# Copia package.json y package-lock.json al directorio de trabajo

COPY package\*.json ./

# Instala las dependencias de la aplicación

RUN npm install

# Copia el resto del código de la aplicación al directorio de trabajo

COPY . .

# Expone el puerto en el que tu aplicación va a correr (ajusta según sea necesario)

EXPOSE 4000

# Define el comando para correr tu aplicación

CMD ["npm", "start"]

**Docker Compose**

version: '3.8'

services:

backend:

build:

context: ./

dockerfile: Dockerfile

ports:

- "4000:4000"

environment:

- NODE\_ENV=production

**2. Kubernetes con Minikube**

**Deployment Manifest**

apiVersion: apps/v1

kind: Deployment

metadata:

name: devops-to-do-list

spec:

replicas: 1

selector:

matchLabels:

app: devops-to-do-list

template:

metadata:

labels:

app: devops-to-do-list

spec:

imagePullSecrets:

- name: myregistrykey

containers:

- name: devops-to-do-list

image: crcaceres05/devops\_to\_do\_list:latest

ports:

- containerPort: 4000

resources:

requests:

memory: "64Mi"

cpu: "250m"

limits:

memory: "128Mi"

cpu: "500m"

**Service Manifest**

apiVersion: v1

kind: Service

metadata:

name: devops-to-do-list-service

spec:

selector:

app: devops-to-do-list

ports:

- protocol: TCP

port: 80

targetPort: 4000

nodePort: 30001

type: NodePort

**3. Instalación de Helm**

**Prometheus**

helm install prometheus prometheus-community/prometheus

**Grafana**

helm install grafana grafana/grafana

**Jenkins**

helm install jenkins jenkinsci/jenkins

**Pasos de Implementación de DevOps**

**1. Configuración del Pipeline en Jenkins**

**Jenkinsfile**

pipeline {

agent any

stage('Checkout') {

steps {

git url: 'https://github.com/crcaceres05/DevOps---To-Do-List.git', branch: 'main'

}

stages {

stage('Build') {

steps {

script {

// Construir la imagen Docker

docker.image('crcaceres05/devops-to-do-list').pull()

docker.build('crcaceres05/devops-to-do-list')

}

}

}

stage('Test') {

steps {

// Aquí irían tus pasos de pruebas, por ejemplo, ejecutar pruebas unitarias

sh 'echo "Ejecutando pruebas..."'

}

}

stage('Deploy') {

steps {

// Desplegar la aplicación en Kubernetes

sh 'kubectl apply -f Kubernetes/deployment.yaml'

sh 'kubectl apply -f Kubernetes/Service.yaml'

}

}

stage('Prometheus') {

steps {

// Instalar Prometheus

sh 'helm install prometheus prometheus-community/prometheus'

}

}

stage('Grafana') {

steps {

// Instalar Grafana

sh 'helm install grafana grafana/grafana'

}

}

stage('Finalize') {

steps {

// Realizar cualquier tarea de limpieza o notificación necesaria

sh 'echo "Pipeline finalizado"'

}

}

}

}

**2. Integración Continua (CI)**

* **Construcción Automática**: Cada commit en el repositorio de GitHub dispara una construcción en Jenkins.
* **Pruebas Automáticas**: Las pruebas unitarias y de integración se ejecutan automáticamente.

**3. Entrega Continua (CD)**

* **Despliegue Automático**: Las nuevas versiones de la aplicación se despliegan automáticamente en el clúster de Kubernetes usando kubectl.

**Prácticas de DevOps Seguidas**

* **Infraestructura como Código (IaC)**: Definición de la infraestructura utilizando manifiestos de Kubernetes y archivos Docker.
* **Monitoreo y Observabilidad**: Implementación de Prometheus y Grafana para monitorear el estado y rendimiento de la aplicación.
* **Automatización**: Uso de Jenkins para automatizar el proceso de construcción, prueba y despliegue de la aplicación.
* **Gestión de Contenedores**: Uso de Docker para contenerizar la aplicación y sus dependencias, facilitando la portabilidad y consistencia del entorno.

**4. Puntos Clave:**

Fue un gran desafio siendo la primera vez trabajando en un proyecto DevOps he aprendido bastante resolviendo varios errores que se me presentaron con docker y kubernetes (minikube) seguire buscando mas proyectos para implementar y seguir mi camino como entusiasta en DevOps, debo de admitir que tuve problemas con los siguientes puntos y estan inconclusos en el proyecto:

* implementar el proyecto en un cloud provider
* implementar terraform
* funcionalidad del ci-cd en jenkins

**5. Evidencias del Proyecto**

* Ambiente de Trabajo - VSCode

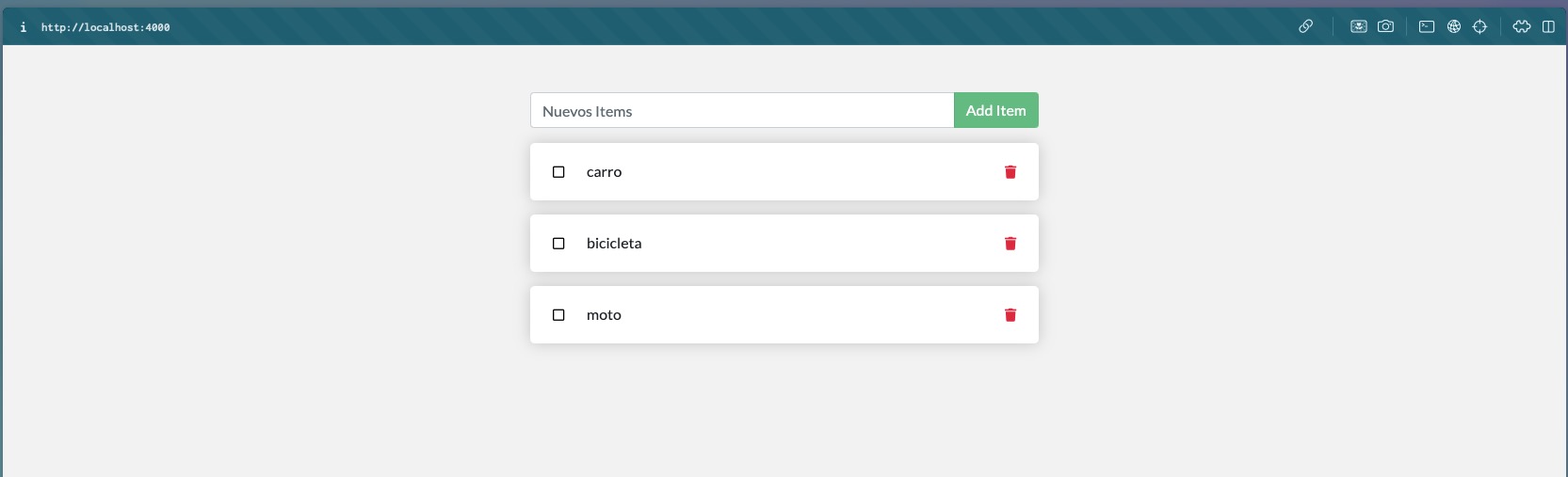
**A screenshot of a computer

Description automatically generated**

* Aplicacion corriendo con docker-compose

**A screen shot of a computer

Description automatically generated**

****

* + <http://localhost:4000>
* Docker Hub
  + <https://hub.docker.com/repository/docker/crcaceres05/devops_to_do_list/general>

A screenshot of a computer

Description automatically generated

* Minikube – Kubernetes

A screenshot of a computer

Description automatically generated

* + Kubectl get services

A screenshot of a computer screen

Description automatically generated

<http://127.0.0.1:57395>

A screenshot of a computer

Description automatically generated

* + Prometheus Metrics

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

* Grafana

A screen shot of a computer

Description automatically generated

* + <http://127.0.0.1:57998>

A screenshot of a computer

Description automatically generated

* Jenkins

A computer screen shot of a computer code

Description automatically generated

* + <http://127.0.0.1:58305>

A screenshot of a computer

Description automatically generated

Nota: Con Jenkins tuve problemas con el pipeline, comparto file de configuracion para un posible feedback:

pipeline {

agent any

stage('Checkout') {

steps {

git url: 'https://github.com/crcaceres05/DevOps---To-Do-List.git', branch: 'main'

}

stages {

stage('Build') {

steps {

script {

// Construir la imagen Docker

docker.image('crcaceres05/devops-to-do-list').pull()

docker.build('crcaceres05/devops-to-do-list')

}

}

}

stage('Test') {

steps {

// Aquí irían tus pasos de pruebas, por ejemplo, ejecutar pruebas unitarias

sh 'echo "Ejecutando pruebas..."'

}

}

stage('Deploy') {

steps {

// Desplegar la aplicación en Kubernetes

sh 'kubectl apply -f Kubernetes/deployment.yaml'

sh 'kubectl apply -f Kubernetes/Service.yaml'

}

}

stage('Prometheus') {

steps {

// Instalar Prometheus

sh 'helm install prometheus prometheus-community/prometheus'

}

}

stage('Grafana') {

steps {

// Instalar Grafana

sh 'helm install grafana grafana/grafana'

}

}

stage('Finalize') {

steps {

// Realizar cualquier tarea de limpieza o notificación necesaria

sh 'echo "Pipeline finalizado"'

}

}

}

}