Informe: Aplicación de DevOps en sitio web de clínica

Lorena soto San Martin

1. Introducción

Durante los últimos meses se ha trabajado en el desarrollo de una aplicación PWA con React, un sitio web titulado “Clínica Chillán”. Actualmente, el sitio ya cuenta con todas sus funcionalidades desarrolladas y se han completado pruebas básicas asociadas a los flujos de los usuarios contemplados para el sitio. Por lo cual, se desea que el sitio quede operativo, sin embargo, el desarrollo solo se encuentra a nivel local, y no cuenta con un repositorio central ni un contenedor que contenga sus configuraciones y dependencias.

Es por esto, que en este informe se implementara una estrategia de implementación de DevOps al sitio web de la clínica para que este quede en producción y pueda ser gestionado y monitoreado.

2. Aplicación de Integración y Entrega Continua

Para poder integrar DevOps a la aplicación, es necesario partir por configurar un pipeline CI/CD, ya que esto permitirá automatizar la construcción, pruebas y despliegue de aplicaciones, asegurando que el código se integre de manera continua y se entregue de forma ágil y sin problemas.

Para configurar un pipeline existen diversa herramienta como Jenkins, GitLab CI o CircleCI, sin embargo, se ha optado por la utilización de GitHub Action, ya que cuenta amplia documentación respecto a su funcionamiento, lo cual facilitara el proceso y cuenta con la ventaja que está completamente integrado en GitHub, lo que implica que todo se maneja dentro de la misma plataforma, lo que facilita la configuración y el mantenimiento.

Los pasos para configurar el pipeline se detallan a continuación:

- **Creación y configuración de archivo Workflow:** en el directorio de GitHub se debe crear un archivo YAML que defina el flujo del pipeline, en él se definen la siguiente información:
    - Se define cuando comenzara a ejecutarse el pipeline, por ejemplo, al ejecutar push en la rama main.
    - Se define con que sistema operativo se ejecutara el pipeline
    - Indicaciones para clonar el repositorio
    - Información asociada a la configuración del entorno (React)
    - Se define instrucciones para la instalación de dependencias
    - Instrucciones para la ejecución de pruebas unitarias y de integración
    - Instrucciones para la construcción del proyecto
    - Información e instrucciones asociadas al despliegue de la aplicación en un servidor

De acuerdo a las características del proyecto “clínica Chillán”, se presenta el archivo YAML configurado:

```yaml
name: React CI/CD Pipeline (Windows)

on:
  push:
    branches:
      - main  # O la rama en la que deseas ejecutar el pipeline
  pull_request:
    branches:
      - main  # Para PRs hacia la rama principal

jobs:
  build:
    runs-on: windows-latest  # Definir el sistema operativo Windows

    steps:
      - name: Check out code
        uses: actions/checkout@v2

      - name: Set up Node.js
        uses: actions/setup-node@v2
        with:
          node-version: '23'  # Especifica la versión de Node.js que usas en tu proyecto

      - name: Install dependencies
        run: npm install  # O 'yarn install' si usas Yarn

      - name: Run tests
        run: npm test -- --coverage  # Ejecuta las pruebas unitarias y genera un reporte de cobertura

      - name: Build the app
        run: npm run build  # Esto genera la versión optimizada de la aplicación en el directorio 'build'
```

Imagen 1. Archivo YAML - Workflow

- **Configuración de secreto en GitHub actions:** En la sección settings/secret se deben agregar las claves necesarias para la aplicación que se necesitan mantener oculta, como, por ejemplo, una clave SSH.

- **Verificación:** de acuerdo a lo configurado en el archivo YAML, en este caso, al ejecutar la sentencia push, se debe ejecutar de forma automática el pipeline definido. En la sección Actions en GitHub se puede revisar el estado de los pipelines y su historial.

Uno de los puntos importantes a integrar en este pipeline CI/CD, son las pruebas automatizadas, ya que permitirá garantizar que el código que se despliega esté libre de errores y cumpla con los requisitos de calidad.

Para proyectos React (JavaScript) algunas de las herramientas más utilizadas para la generación de pruebas son Jest, Mocha o Jasmine. Para implementar en este caso pruebas con Jest se deben realizar los siguientes pasos:

- **Instalar Jest**: Mediante npm, ejecutando el comando: *npm install --save-dev jest*

- **Escribir pruebas:** Se debe crear un archivo con la extensión test.js donde se describa una prueba a ejecutar. A modo de ejemplo, se muestra el archivo inicio.test.js con unas pruebas asociadas a la página de inicio de la aplicación.

```
import { render, screen } from '@testing-library/react';
import PaginaInicio from '../PaginaInicio';

describe('PaginaInicio', () => {
  test('muestra el título de bienvenida correctamente', () => {
    render(<PaginaInicio />);

    // Verificar que el título de la página está presente
    const titulo = screen.getByText('Bienvenidos a Clínica Chillán');
    expect(titulo).toBeInTheDocument();
  });

  test('muestra la lista de servicios', () => {
    render(<PaginaInicio />);

    // Verificar que los servicios están presentes
    const servicio1 = screen.getByText('Consulta General');
    const servicio2 = screen.getByText('Laboratorio');
    const servicio3 = screen.getByText('Urgencias');

    expect(servicio1).toBeInTheDocument();
    expect(servicio2).toBeInTheDocument();
    expect(servicio3).toBeInTheDocument();
  });
});
```

Imagen 2: inicio.test.js

- **Actualizar package.json**: Como se ve en el archivo (imagen 1) el paso (step) 4 permite ejecutar las pruebas definidas en los archivos test.js y generar un reporte de acuerdo a los resultados. Para que esto ocurra, se debe agregar en el archivo package.json un script de test, como el siguiente:

```
"scripts": {
  "dev": "vite",
  "build": "vite build",
  "lint": "eslint .",
  "preview": "vite preview",
  "test": "jest"
},
```

Imagen 3: configuración de test – package.json

3. Uso de Contenedores y Orquestación

El software necesita contar con un contenedor que permita empaquetar su código, dependencias y configuraciones, actualmente dentro de los contenedores mas utilizados se encuentra Docker, por lo cual, se decide implementar su uso en el proyecto.

De acuerdo a la documentación, los pasos para implementar Docker en el proyecto, son los siguientes:

- **Instalación:** De acuerdo al sistema operativo utilizado, se debe instalar Docker, las instrucciones se encuentran en su pagina oficial https://docs.docker.com/.

- **Crear Dockerfile:** Dockerfile es un archivo de texto plano con un conjunto de instrucciones que define como se debe construir el contenedor de la aplicación. Entre las instrucciones que puede contener se encuentran: establecer variables, sistema operativo, imagen base, dependencias necesarias, comando de inicio, entre otros.

- **Construir la imagen de dockers:** Con el archivo Dockerfile desarrollado, este se puede leer y construir una imagen siguiendo las instrucciones ahí descritas, para ello se debe ejecutar el comando: *docker build -t nombre_imagen.*

- **Ejecutar el contenedor:** Al ya tener el contenedor construido, este se puede ejecutar mediante el comando: *docker run -d -p puerto:puerto nombre__imagen.*

- **Visualizar aplicación:** Al abrir un navegador e ingresar la url ***http://localhost:puerto*** se podrá visualizar la aplicación, si su construcción fue correcta.

- **Docker compose:** Como la aplicación cuenta con mas de un servicio, se necesita utilizar Docker compose, para manejar los contenedores asociados a la aplicación. Para ello se debe crear el archivo *docker-compose.yml*, el cual especifica que se necesita para implementar la aplicación, sus configuraciones, redes, volúmenes entre otras opciones. Luego se pueden inicializar mediante el comando: *docker-compose up.*

Al trabajar con un contenedor también es necesario contar con una herramienta para su gestión, para ello se utilizará Kubernetes, esto permitirá administrar, coordinar y programar los contenedores necesarios para el funcionamiento del sistema. Además, utilizar ambas herramientas en conjunto proporciona múltiples ventajas a las aplicaciones como escalabilidad, agilidad y resistencia.

Para implementar Kubernetes, ya contando con una imagen Docker de la aplicación, se deben realizar los siguientes pasos:

- **Subir imagen a registro de contenedores**: Para que Kubernetes pueda acedar a imagen, esta debe estar disponible en un registro de contenedores, el utilizado, será Docker Hub. Para ello se deben ejecutar los siguientes comandos:
    - *docker login*
    - *docker tag nombre-imagen tu_usuario/nombre-imagen:tag*
    - *docker push tu_usuario/nombre-imagen:tag*

- **Crear archivos de configuración de Kubernetes:** Para que Kubernetes pueda usar la imagen se deben crear dos archivos:
    - **deployment.yaml:** Corresponde al archivo de despliegue, describe cómo Kubernetes debe ejecutar tu contenedor.
    - **service.yaml:** Corresponde al archivo de servicio, este archivo es necesario para proporcionar una dirección estable a la aplicación.

- **Despliegue de la aplicación:** Con los archivos del paso a anterior construidos, ya se puede desplegar la aplicación en Kubernetes, para ello se deben ejecutar los siguientes comandos:
    - kubectl apply -f deployment.yaml
    - kubectl apply -f service.yaml
    - kubectl get pods
    - kubectl get services

  Si todo se ejecuta correctamente, se deberían mostrar los pods y los servicios corriendo.

- **Acceso a la aplicación:** Para acceder a la aplicación se debe obtener si IP publica, para ello se puede ejecutar el comando: *kubectl get services* y luego acceder a la aplicación visitando la IP.

4. Monitoreo y Seguridad en DevOps

Para garantizar el rendimiento, la escalabilidad y la estabilidad de la aplicación y la infraestructura, es necesario contar con una estrategia para el monitoreo de log y métricas del sistema, esto permitirá que la aplicación se mantenga operativa y se pueda tomar decisiones y actuar oportunamente ante cualquier inconveniente. Para ello, existen múltiples herramientas que facilitan esta labor, permitiendo automatizar ciertas funciones y obtener métricas en tiempo real. Actualmente, la herramienta de monitoreo más utilizada es Prometheus, por lo cual, es la elegida para realizar el monitoreo de sistema y evaluar ciertos aspectos de la seguridad del sitio.

Para poder asegurar tanto la eficiencia como la seguridad de la aplicación, se implementará una estrategia basada en los siguientes puntos:

- **Monitoreo:** Mediante las funcionalidades ofrecidas por Prometheus se medirán diferentes métricas asociadas al rendimiento y eficiencia de la aplicación y su infraestructura. Además, como un complemento se integrará la herramienta Grafana, la cual, proporcionada un dashboard que facilitará el análisis de las métricas obtenidas lo que permitirá detectar rápidamente cualquier problema con la aplicación.
- **Uso de alertas:** Prometheus tiene un sistema de alertas integrado que permite definir alertas de acuerdo a ciertos comportamientos de las métricas analizadas. Por

lo cual, para sacar el mayor beneficio de la herramienta se definirá un conjunto de alertas, en el caso particular de la clínica, en una primera instancia, las alertas definidas tienen relación con la cantidad de errores en un periodo determinado, alerta por caída de sitio, tiempos de respuesta superiores al promedio identificado y alto uso de recursos.

- **Seguridad:** Mediante las funcionalidades proporcionadas por la herramienta de monitoreo, se analizarán métricas asociadas a el acceso no autorizado y la autenticación. Además, se realizarán configuraciones e la aplicación para monitorear errores y fallas en la aplicación que pueden ser potencialmente explotados.