Venta de Helados

Aplicación de microservicios.

Integrantes:

- José Maximiliano Díaz Méndez
- Barajas Zavala Ulises

Descripción

Este proyecto consta de un punto de venta de una heladería en el que se divide en 2 servicios la funcionalidad del servidor para ser adaptada a una arquitectura de microservicios, a través del uso de un cluster de kubernetes.

Servicios

- **CRUD:** Esta parte se ocupa de hacer las llamadas a la Base de Datos, todas la llamadas relacionadas al CRUD, ya sea la de crear registros, leer, modificar y eliminar.

- **Auth:** Esta parte se ocupa de autenticar las llamadas que se hacen desde la web app, para agregar seguridad y privacidad, de modo que nadie más que la web-app puedan hacer solicitudes al CRUD.

  Esto a través de un token que se entrega a la web-app, este token tiene una duración de 24 hrs y el CRUD confirma este token con auth cada que CRUD interactúa con la base de datos.

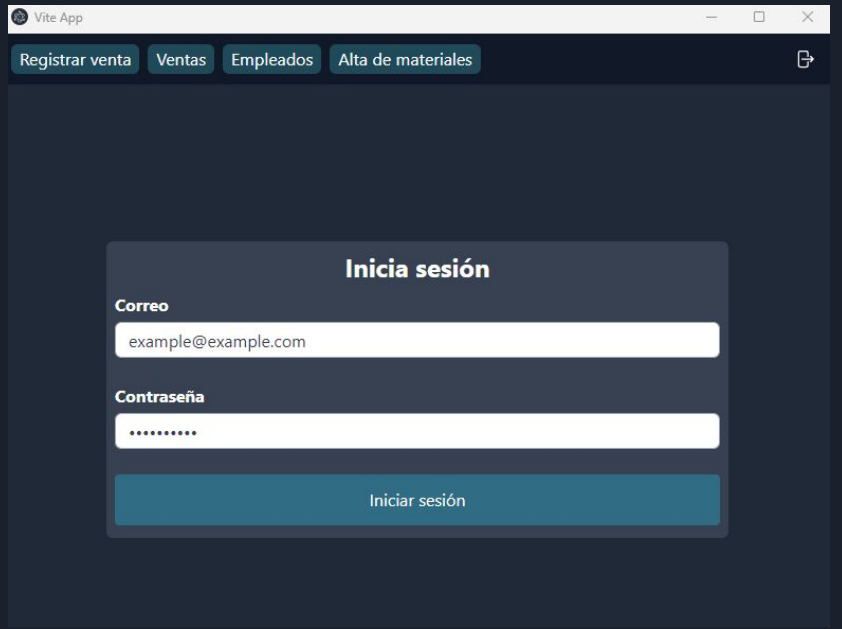
*Estos servicios utilizan una **réplica** para garantizar la consistencia de los servicios en la app*

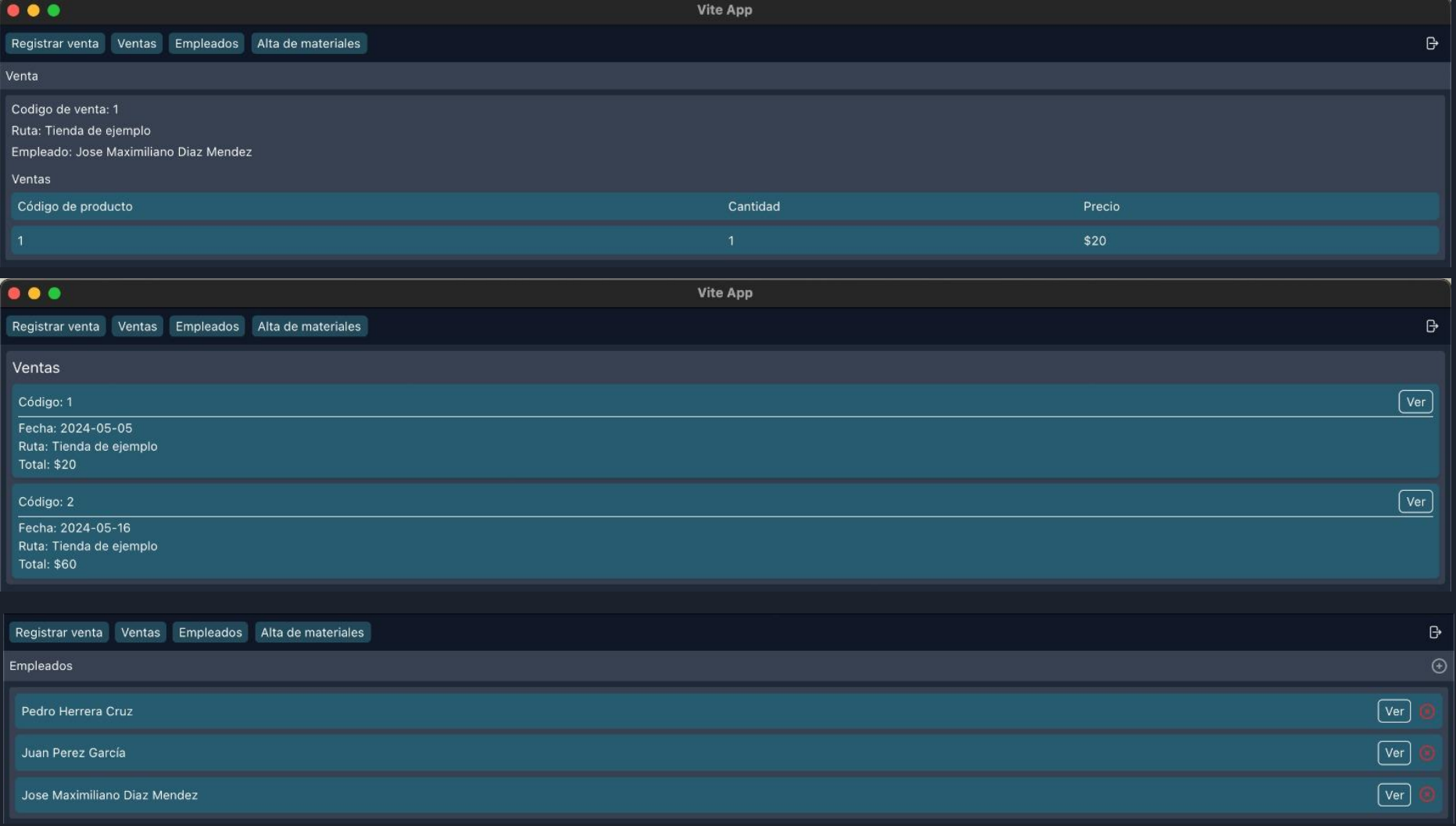
WEB - APP

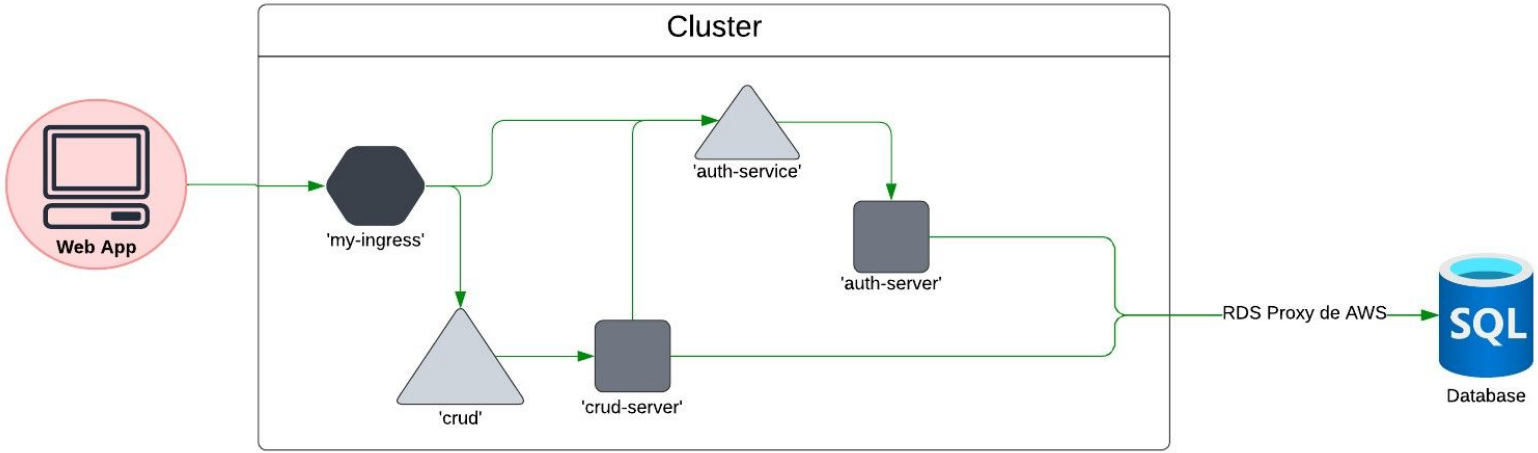
La web - app creada usando Vue.js, un framework de JavaScript utilizado para construir **interfaces de usuario y aplicaciones de una sola página** (SPA).

También fue envuelto en **electron.js,** un framework de código abierto que permite desarrollar aplicaciones de escritorio multiplataforma utilizando tecnologías web como HTML, CSS y JavaScript.

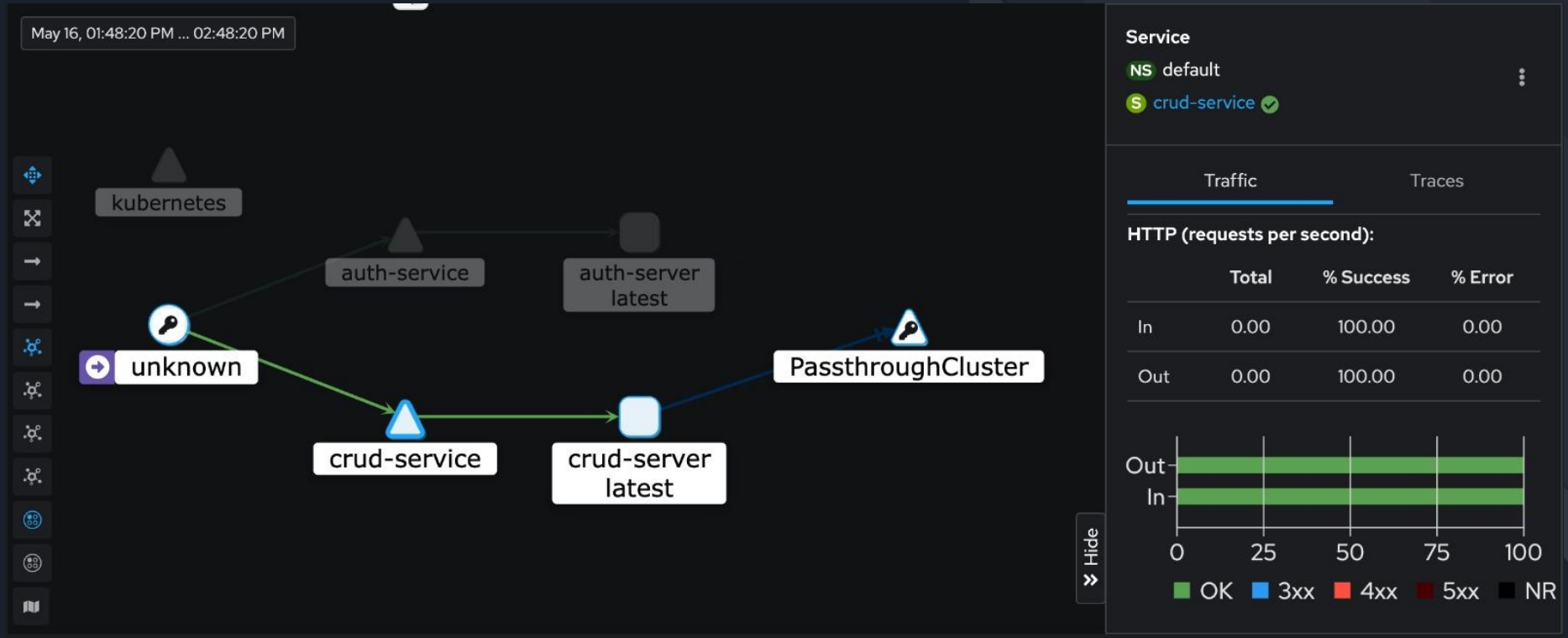
Con esto en mente la parte de la web app no cuenta como un servicio ya que es una app en sí mismo que se comunica con los servicios antes descritos.




Vite App
Registrar venta
Ventas
Empleados
Alta de materiales
Venta
Codigo de venta: 1
Ruta: Tienda de ejemplo
Empleado: Jose Maximiliano Diaz Mendez
Ventas
Código de producto
Cantidad
Precio
1
1
$20
Vite App
Registrar venta
Ventas
Empleados
Alta de materiales
Ventas
Código: 1
Ver
Fecha: 2024-05-05
Ruta: Tienda de ejemplo
Total: $20
Código: 2
Ver
Fecha: 2024-05-16
Ruta: Tienda de ejemplo
Total: $60
Registrar venta
Ventas
Empleados
Alta de materiales
Empleados
Pedro Herrera Cruz
Ver
Juan Perez García
Ver
Jose Maximiliano Diaz Mendez
Ver


Cluster
Web App
'my-ingress'
'auth-service'
'auth-server'
'crud'
'crud-server'
RDS Proxy de AWS
SQL
Database

Istio


May 16, 01:48:20 PM ... 02:48:20 PM
kubernetes
auth-service
auth-server
latest
unknown
PassthroughCluster
crud-service
crud-server
latest
Service
NS default
S crud-service
Traffic
Traces
HTTP (requests per second):
Total
% Success
% Error
In
0.00
100.00
0.00
Out
0.00
100.00
0.00
Out
In
0
25
50
75
100
OK
3xx
4xx
5xx
NR
Hide

Arquitectura

- **Un secreto:**

  ‘*db-secrets’*

  Almacena de forma segura los datos de inicio de sesión de la base de datos, incluyendo el nombre de host, usuario, contraseña y nombre de la base de datos.

- **Pod**:

  ‘*crud-server’*

  Ejecuta el servidor CRUD, se replica 2 veces para proporcionar redundancia y escalabilidad.

  El servidor CRUD se comunica con la base de datos utilizando los datos de inicio de sesión almacenados en el secreto db-secrets.

Arquitectura

- **Pod:**

  ‘auth-server’

  Ejecuta el servidor de autenticación, que al igual que el servidor CRUD, se replica 2 veces para garantizar la disponibilidad. Este servidor utiliza los mismos datos de inicio de sesión de la base de datos del secreto db-secrets.

- **2 servicios internos:**

  ‘crud-service’ ‘auth-service’

  Exponen los pods *crud-server* y *auth-server*, respectivamente y facilitan la comunicación entre los diferentes componentes de la aplicación dentro del clúster de Kubernetes.

Arquitectura

- **Base de datos:**
  Para la base de datos se usa **2** instancias de **AWS RDS (Una de escritura y otra de lectura)** que se acceden mediante un **proxy de RDS** el cual enruta el tráfico de lectura y escritura a la instancia correspondiente.

  Además de permitir que en caso de error en la instancia de escritura cambiar la de lectura a escritura durante el tiempo que tome estabilizar la instancia para después sincronizarlas y restablecer sus roles.

Arquitectura

- **Ingress:**

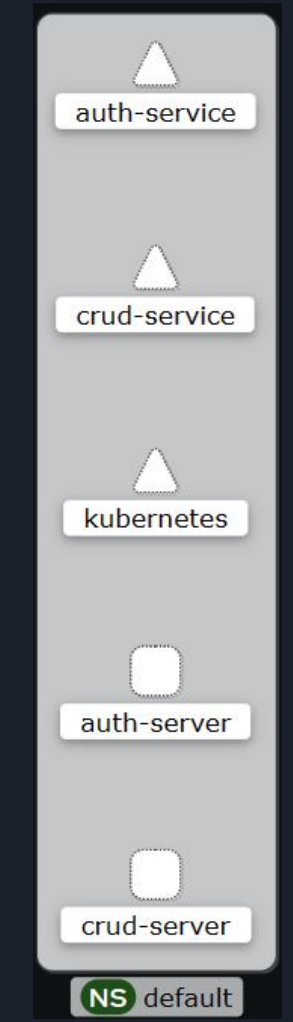
  ‘my-ingress’

  Actúa como punto de entrada a la aplicación desde el exterior del clúster, enruta el tráfico HTTP entrante a los servicios internos *crud-service* y *auth-service* según las rutas especificadas.

  De esta manera, permite que las solicitudes externas lleguen a los componentes correspondientes de la aplicación.

- La arquitectura también incluye **Istio** como una capa de servicio de malla, que proporciona funcionalidades como el enrutamiento, el control de tráfico, la seguridad y la observabilidad para los servicios desplegados en el clúster de Kubernetes.

  A través de la configuración de reglas de tráfico y políticas de seguridad, Istio ayuda a gestionar y asegurar las comunicaciones entre los diferentes componentes de la aplicación.



Ingeniería del Caos

La ingeniería del caos se refiere al estudio y la aplicación de principios de sistemas caóticos y dinámicos no lineales en el diseño, análisis y control de sistemas informáticos y redes.

Este enfoque busca comprender y aprovechar el comportamiento impredecible pero estructurado de los sistemas caóticos para mejorar la robustez, la seguridad y la eficiencia de los sistemas informáticos.

Chaos Toolkit

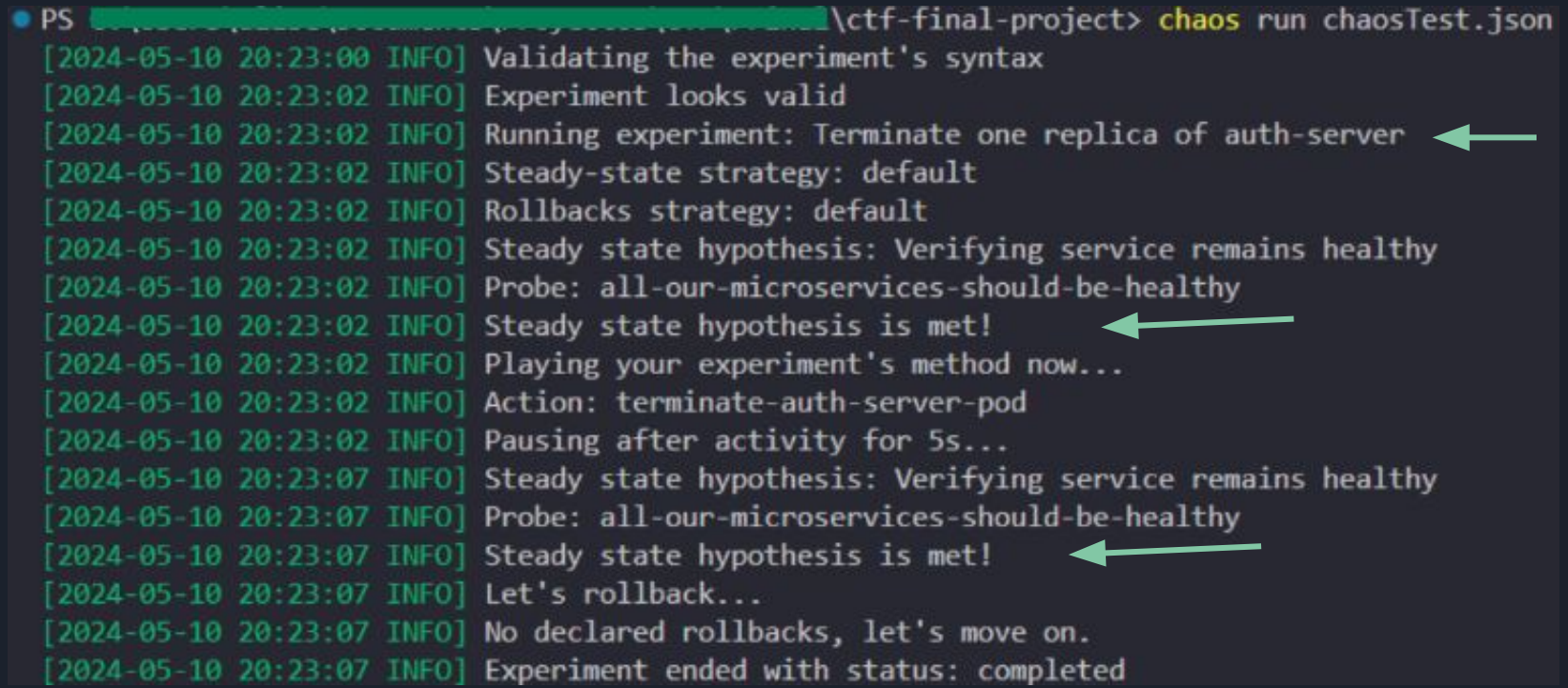
Es una herramienta de código abierto utilizada para realizar experimentos de ingeniería del caos en sistemas distribuidos. Proporciona un conjunto de comandos y bibliotecas que permiten a los equipos de ingeniería introducir y controlar el caos en sus sistemas para probar su resiliencia y capacidad de recuperación frente a situaciones inesperadas.

También ofrece una integración sólida con **Kubernetes**, permitiendo a los ingenieros realizar experimentos de caos dentro de los clústeres de Kubernetes.

Con configuración sencilla mediante archivos **YAML o JSON**, puede simular eventos caóticos como la eliminación de nodos o la restricción de recursos. Esto nos proporcionó una manera eficiente de evaluar la resiliencia de la aplicación de microservicios.

Chaos Test

En este caso escribimos una prueba en un JSON para apagar uno de nuestros pods de manera aleatoria, chaos toolkit maneja teorías en sus pruebas, que le permiten a la herramienta evaluar por sí misma el estado de la arquitectura, y decidir si se cumplio el objetivo de la prueba o si la estructura sufrió graves problemas con el inconveniente. Al correr la prueba podemos ver este output:



Video explicativo:

https://youtu.be/6V7sdCSvdeI