# DESPLIEGUE DE ARQUITECTURA ESCALABLE EN KUBERNETES CON ENFOQUE GITOPS

PLATAFORMAS II - INGENIERÍA DE SOFTWARE V

PROFESORES: CHRISTIAN DAVID FLOR ASTUDILLO, JUAN CARLOS MUÑOZ FERNÁNDEZ

INTEGRANTES: SANTIAGO BARRAZA, LUISA CASTAÑO, JUAN YUSTES

# CONTENIDO


1 Introducción

2 Metodología

3 Estrategia de Branching

4 Diseño e Implementación

5 Seguridad impementada

6 Calidad de código

7 Observabilidad y monitoreo

8 Escalabilidad y resiliencia

9 Desafíos y conclusiones


# INTRODUCCIÓN

Este proyecto abordó la implementación de una arquitectura de microservicios en Kubernetes, basada en un sistema e-commerce, se aplicaron prácticas avanzadas de DevOps y GitOps para automatizar despliegues, gestionar configuraciones, asegurar comunicaciones y monitorear servicios. El objetivo fue lograr una solución escalable, segura y lista para producción en la nube.

# METODOLOGÍA

¿Por qué Kanban?

- Equipo de 3 personas
- Evita la rigidez de Scrum
- Flujo continuo y flexible
- Retroalimentación en cualquier momento

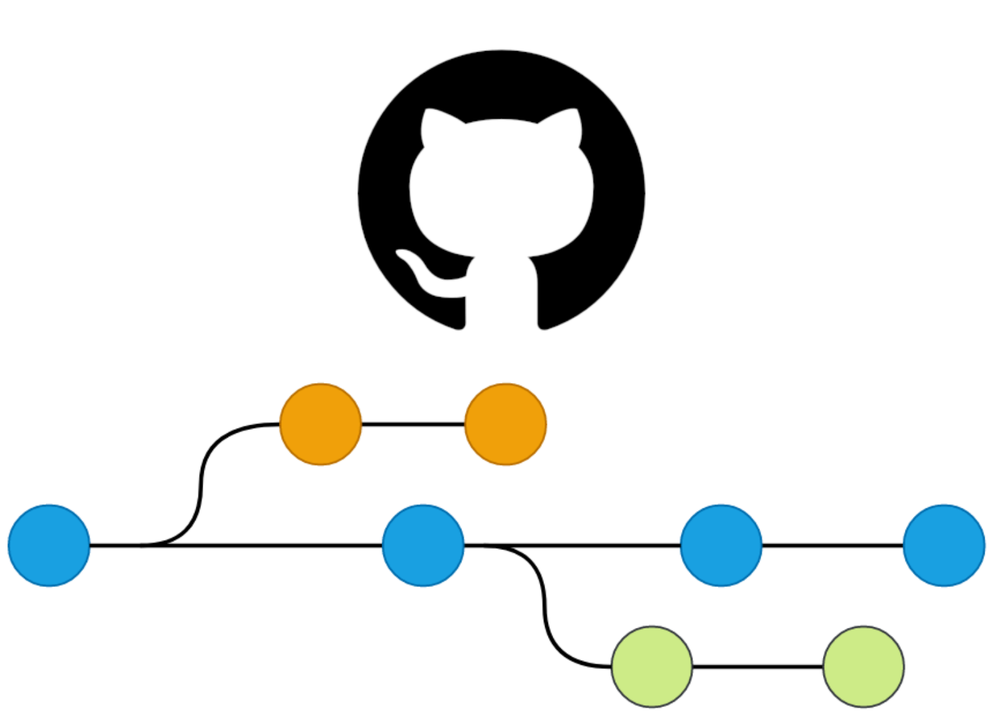
Tablero Kanban:

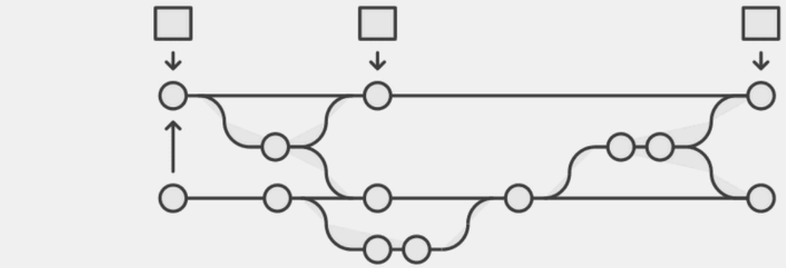
- Backlog: Tareas del proyecto
- Planned: Próximas tareas asignadas
- Doing: En curso
- Blocked: Problemas, se necesita ayuda
- Done: Finalizadas

# ESTRATEGIA DE BRANCHING

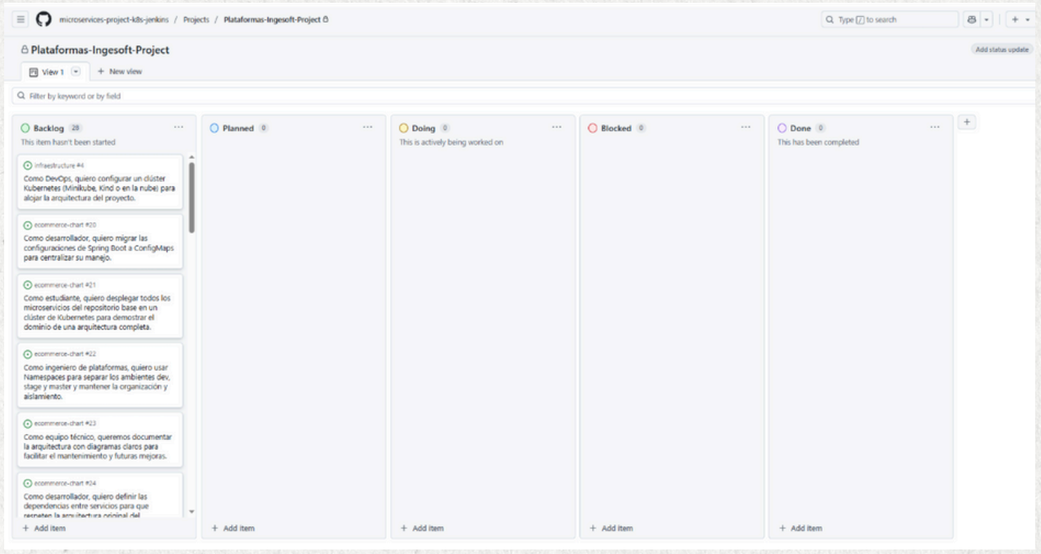
Durante el desarrollo del proyecto se utilizó GitHub Flow por su simplicidad, agilidad y enfoque en integración continua, lo cual facilitó el trabajo colaborativo y el despliegue rápido de cambios en un entorno de microservicios sobre Kubernetes en AWS, esta estrategia fue útil para avanzar rápidamente en etapas tempranas.

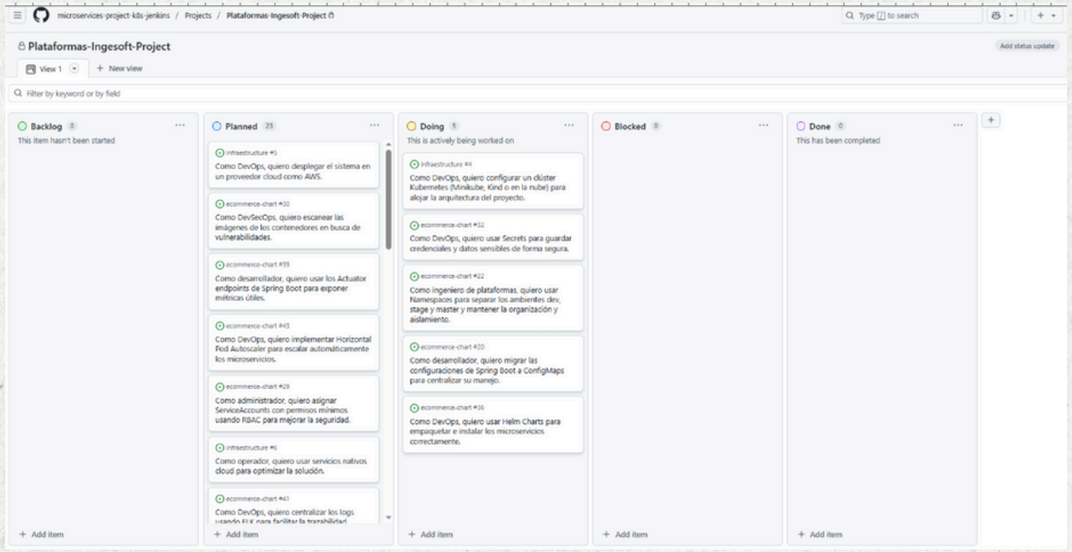
Sin embargo, el resultado final del proyecto está orientado a un modelo más controlado, por lo que se adoptará Git Flow, que permite una mejor gestión de ramas como develop, stage y main, alineándose con prácticas GitOps.

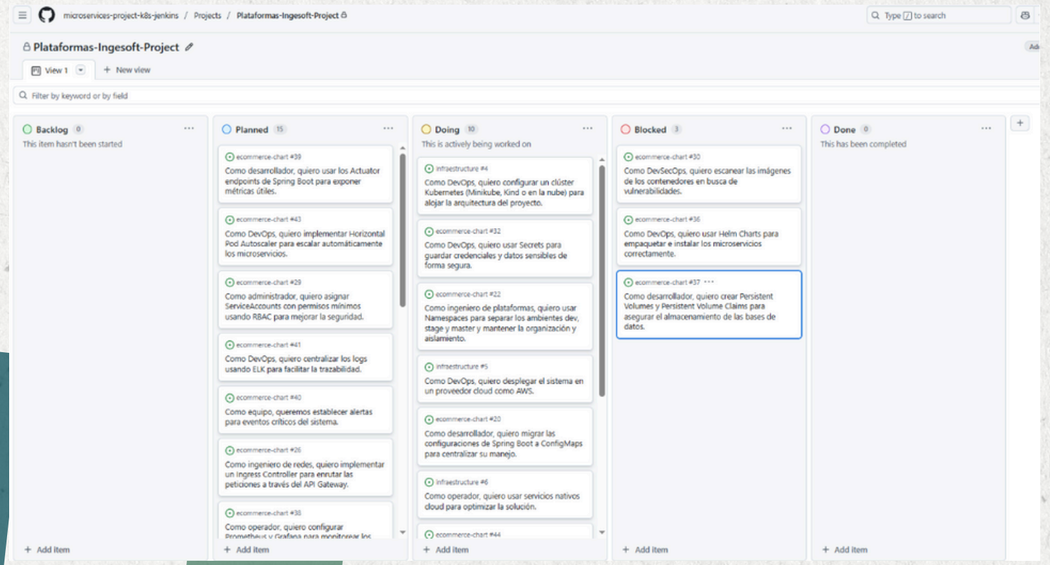


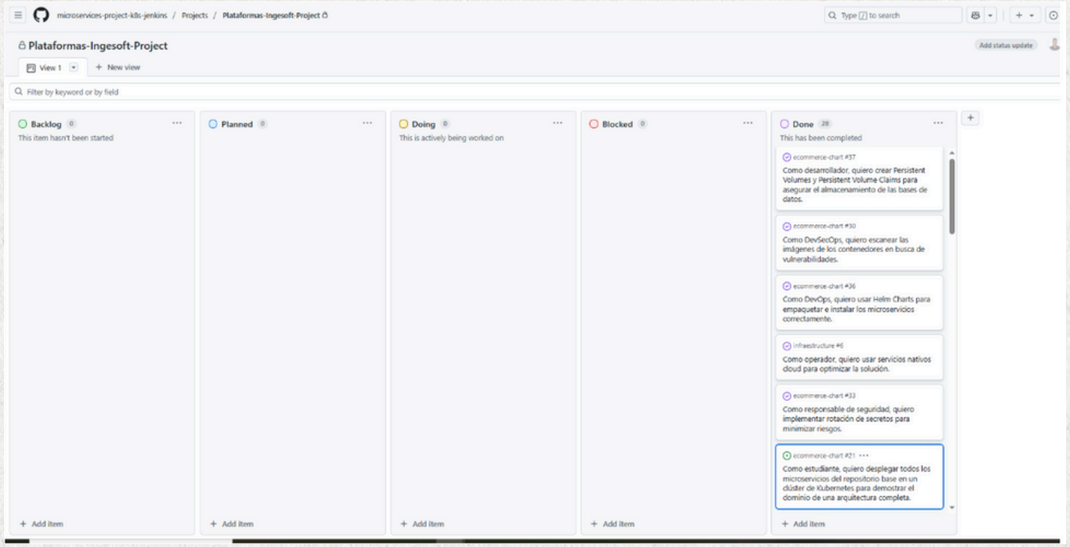


# EVIDENCIA


microservices-project-k8s-jenkins / Projects / Plataformas-Ingesoft-Project
Plataformas-Ingesoft-Project
View 1
New view
Filter by keyword or by field
Backlog 28
This item hasn't been started
Como DevOps, quiero configurar un clúster Kubernetes (Minikube, Kind o en la nube) para alojar la arquitectura del proyecto.
Como desarrollador, quiero migrar las configuraciones de Spring Boot a ConfigMaps para centralizar su manejo.
Como estudiante, quiero desplegar todos los microservicios del repositorio base en un clúster de Kubernetes para demostrar el dominio de una arquitectura completa.
Como ingeniero de plataformas, quiero usar Namespaces para separar los ambientes dev, stage y master y mantener la organización y aislamiento.
Como equipo técnico, queremos documentar la arquitectura con diagramas claros para facilitar el mantenimiento y futuras mejoras.
Como desarrollador, quiero definir las
Planned 0
This is actively being worked on
Doing 0
This is actively being worked on
Blocked 0
Done 0
This has been completed
Add item

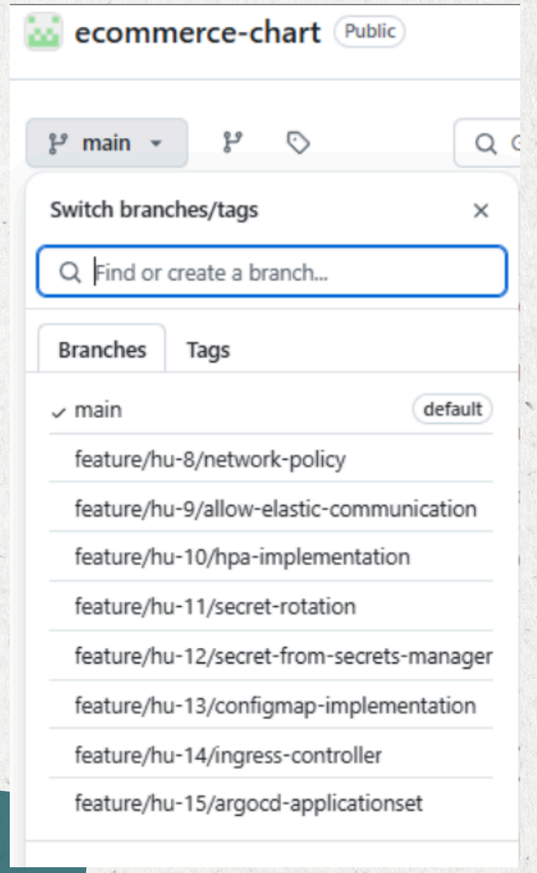

microservices-project-k8s-jenkins / Projects / Plataformas-Ingesoft-Project
Plataformas-Ingesoft-Project
View 1
New view
Filter by keyword or by field
Backlog 0
This item hasn't been started
Planned 23
Como DevOps, quiero desplegar el sistema en un proveedor cloud como AWS.
Como DevSecOps, quiero escanear las imágenes de los contenedores en busca de vulnerabilidades.
Como desarrollador, quiero usar los Actuator endpoints de Spring Boot para exponer métricas útiles.
Como DevOps, quiero implementar Horizontal Pod Autoscaler para escalar automáticamente los microservicios.
Como administrador, quiero asignar ServiceAccounts con permisos mínimos usando RBAC para mejorar la seguridad.
Como operador, quiero usar servicios nativos cloud para optimizar la solución.
Como DevOps, quiero centralizar los logs
Doing 5
This is actively being worked on
Como DevOps, quiero configurar un clúster Kubernetes (Minikube, Kind o en la nube) para alojar la arquitectura del proyecto.
Como DevOps, quiero usar Secrets para guardar credenciales y datos sensibles de forma segura.
Como ingeniero de plataformas, quiero usar Namespaces para separar los ambientes dev, stage y master y mantener la organización y aislamiento.
Como desarrollador, quiero migrar las configuraciones de Spring Boot a ConfigMaps para centralizar su manejo.
Como DevOps, quiero usar Helm Charts para empaquetar e instalar los microservicios correctamente.
Blocked 0
Done 0
This has been completed
Add item


microservices-project-k8s-jenkins / Projects / Plataformas-Ingesoft-Project
Plataformas-Ingesoft-Project
View 1
New view
Filter by keyword or by field
Backlog 0
This item hasn't been started
Planned 15
Como desarrollador, quiero usar los Actuator endpoints de Spring Boot para exponer métricas útiles.
Como DevOps, quiero implementar Horizontal Pod Autoscaler para escalar automáticamente los microservicios.
Como administrador, quiero asignar ServiceAccounts con permisos mínimos usando RBAC para mejorar la seguridad.
Como DevOps, quiero centralizar los logs usando ELK para facilitar la trazabilidad.
Como equipo, queremos establecer alertas para eventos críticos del sistema.
Como ingeniero de redes, quiero implementar un ingress Controller para enrutar las peticiones a través del API Gateway.
Como operador, quiero configurar
Doing 10
This is actively being worked on
Como DevOps, quiero configurar un clúster Kubernetes (Minikube, Kind o en la nube) para alojar la arquitectura del proyecto.
Como DevOps, quiero usar Secrets para guardar credenciales y datos sensibles de forma segura.
Como ingeniero de plataformas, quiero usar Namespaces para separar los ambientes dev, stage y master y mantener la organización y aislamiento.
Como DevOps, quiero desplegar el sistema en un proveedor cloud como AWS.
Como desarrollador, quiero migrar las configuraciones de Spring Boot a ConfigMaps para centralizar su manejo.
Como operador, quiero usar servicios nativos cloud para optimizar la solución.
Blocked 3
Como DevSecOps, quiero escanear las imágenes de los contenedores en busca de vulnerabilidades.
Como DevOps, quiero usar Helm Charts para empaquetar e instalar los microservicios correctamente.
Como desarrollador, quiero crear Persistent Volumes y Persistent Volume Claims para asegurar el almacenamiento de las bases de datos.
Done 0
This has been completed
Add item

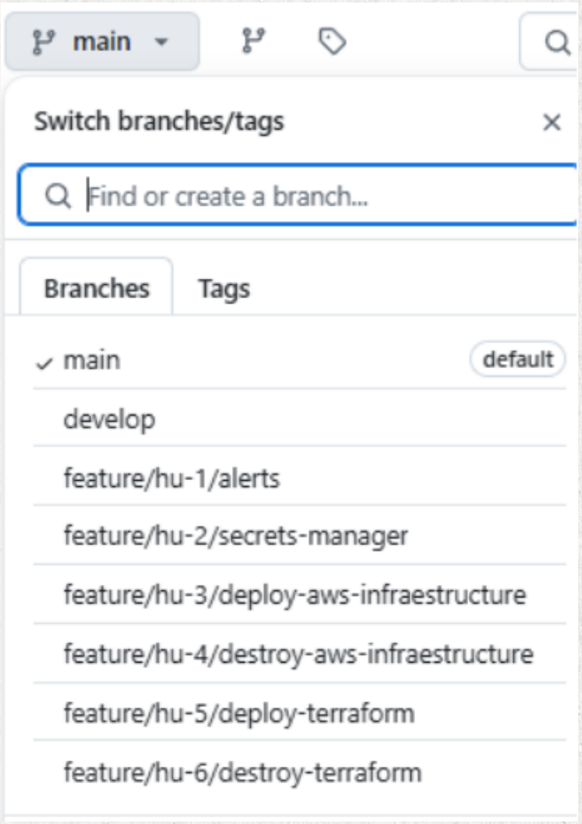

microservices-project-k8s-jenkins / Projects / Plataformas-Ingesoft-Project
Plataformas-Ingesoft-Project
View 1
New view
Filter by keyword or by field
Backlog 0
This item hasn't been started
Planned 0
Doing 0
This is actively being worked on
Blocked 0
Done 28
This has been completed
Como desarrollador, quiero crear Persistent Volumes y Persistent Volume Claims para asegurar el almacenamiento de las bases de datos.
Como DevSecOps, quiero escanear las imágenes de los contenedores en busca de vulnerabilidades.
Como DevOps, quiero usar Helm Charts para empaquetar e instalar los microservicios correctamente.
Como operador, quiero usar servicios nativos cloud para optimizar la solución.
Como responsable de seguridad, quiero implementar rotación de secretos para minimizar riesgos.
Como estudiante, quiero desplegar todos los microservicios del repositorio base en un clúster de Kubernetes para demostrar el dominio de una arquitectura completa.
Add item

# EVIDENCIA

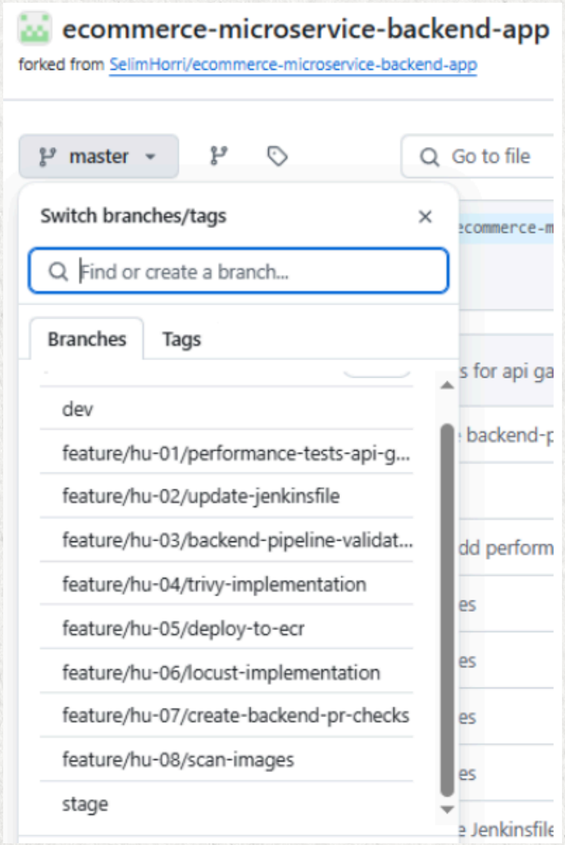
## ECOMMERCE-CHART:



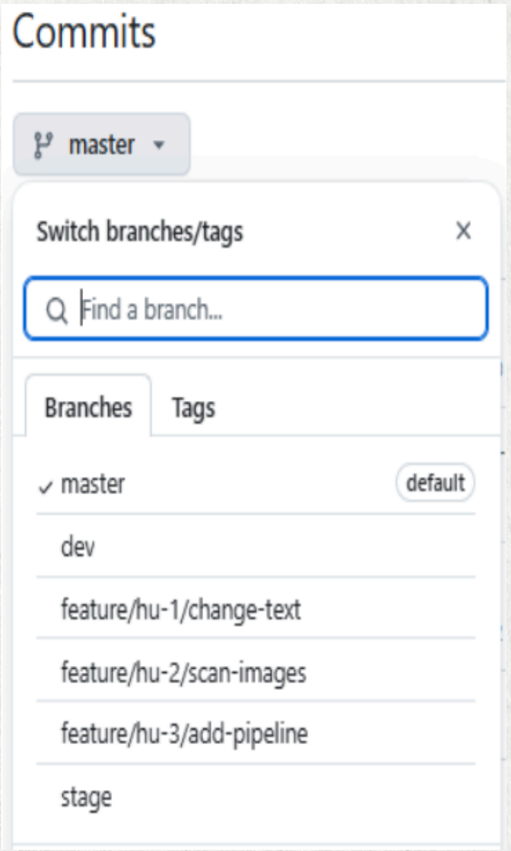

## INFRAESTRUCTURE:




## ECOMMERCE-MICROSERVICE-BACKEND-APP:

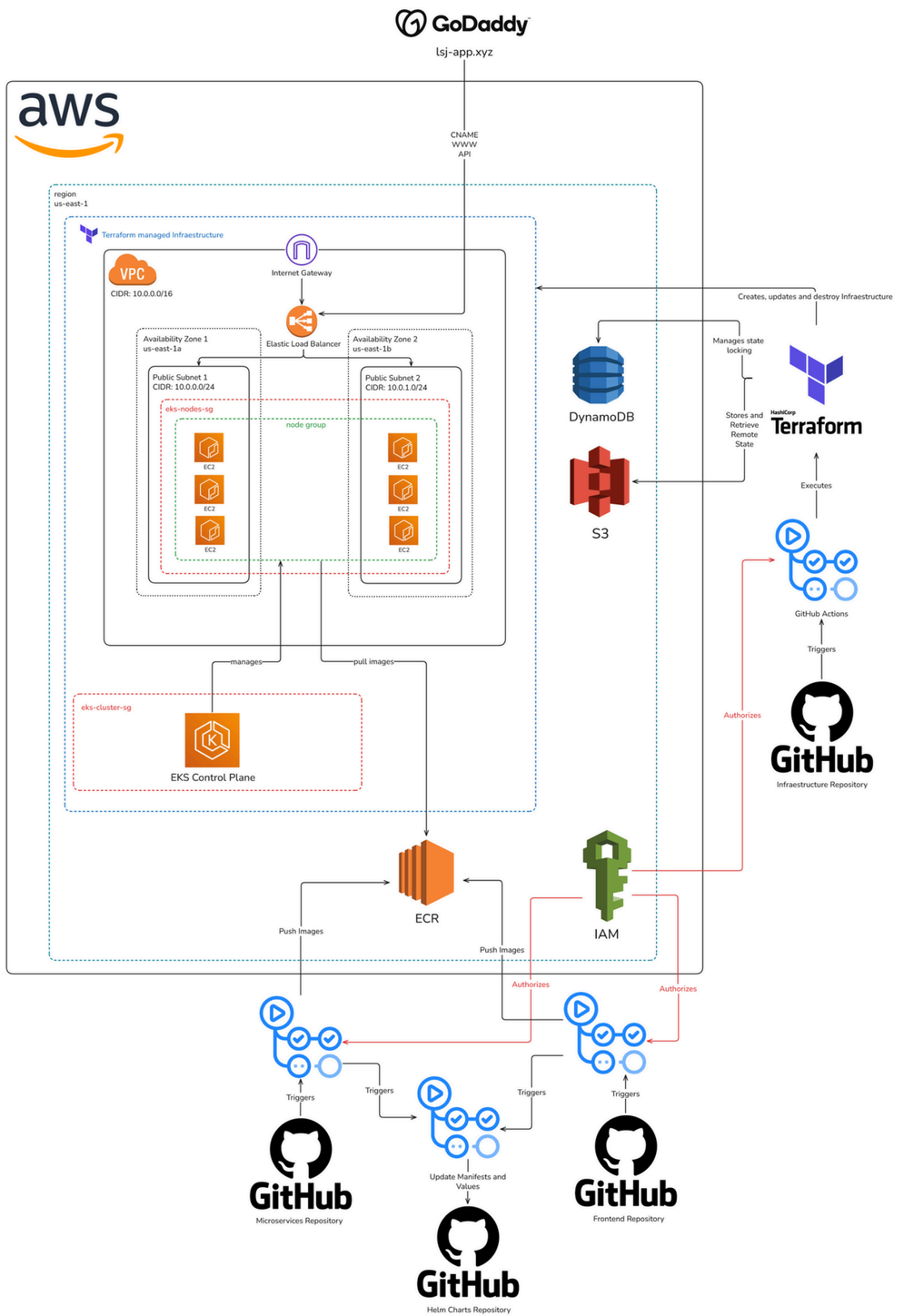



## ECOMMERCE-FRONTEND-WEB-APP:

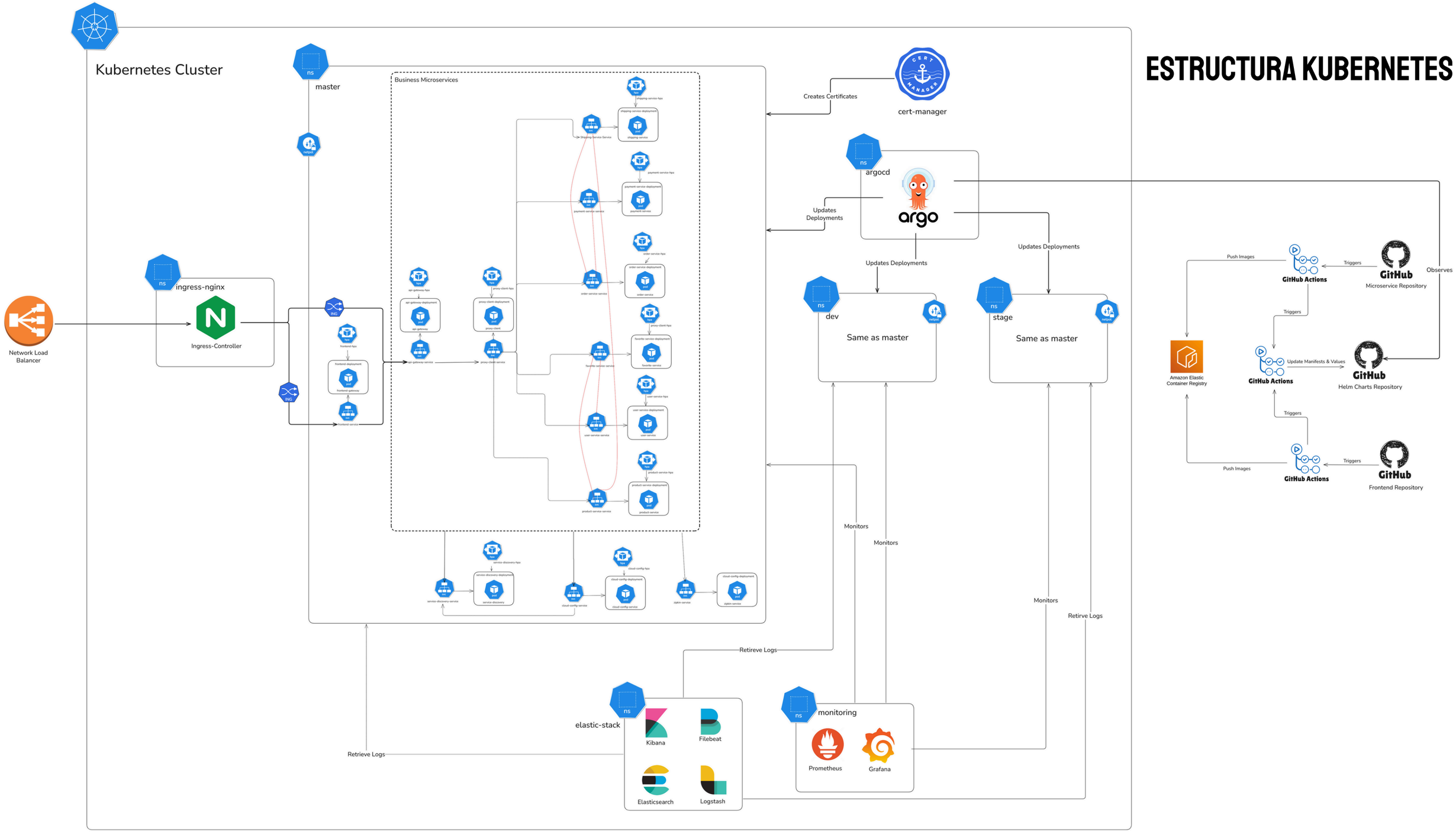


# DISEÑO E IMPLEMENTACIÓN

# ARQUITECTURA EN AWS



# ESTRUCTURA KUBERNETES



# PATRONES DE CLOUD

## HEALTH ENDPOINT MONITORING

Uso de actuator endpoints de Spring Boot.

```yaml
livenessProbe:
  httpGet:
    path: /user-service/actuator/health
    port: {{ index .Values "user-service" "port" }}
  initialDelaySeconds: 30
  periodSeconds: 10
  timeoutSeconds: 5
  failureThreshold: 3

readinessProbe:
  httpGet:
    path: /user-service/actuator/health
    port: {{ index .Values "user-service" "port" }}
  initialDelaySeconds: 10
  periodSeconds: 5
  timeoutSeconds: 5
  failureThreshold: 3
```

## EXTERNAL CONFIGURATION STORE


Uso de secrets y configmaps como configuración externa de spring boot

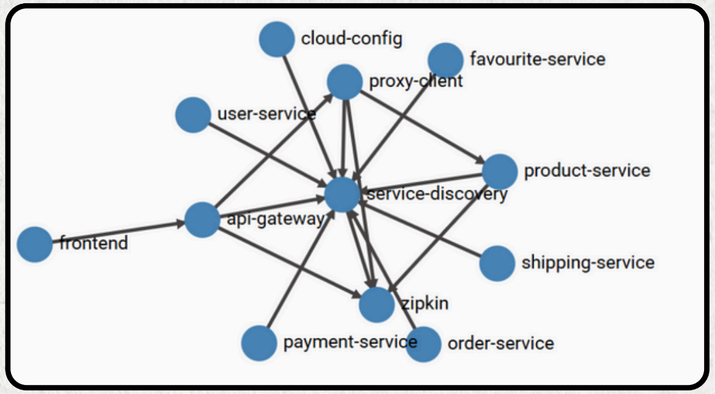
## RETRY

Reintento de solicitudes desde el frontend.

```
▸ GET http://localhost:4200/app/api/categories 404 (Not Found)
Error recibido: 404 Not Found
Reintentando petición (1/3) debido a error 404...
root
Content Script re-injected or page loaded
[webpack-dev-server] Live Reloading enabled.
▸ GET http://localhost:4200/app/api/categories 404 (Not Found)
Error recibido: 404 Not Found
Reintentando petición (2/3) debido a error 404...
▸ GET http://localhost:4200/app/api/categories 404 (Not Found)
Error recibido: 404 Not Found
Reintentando petición (3/3) debido a error 404...
```

## SIDECAR

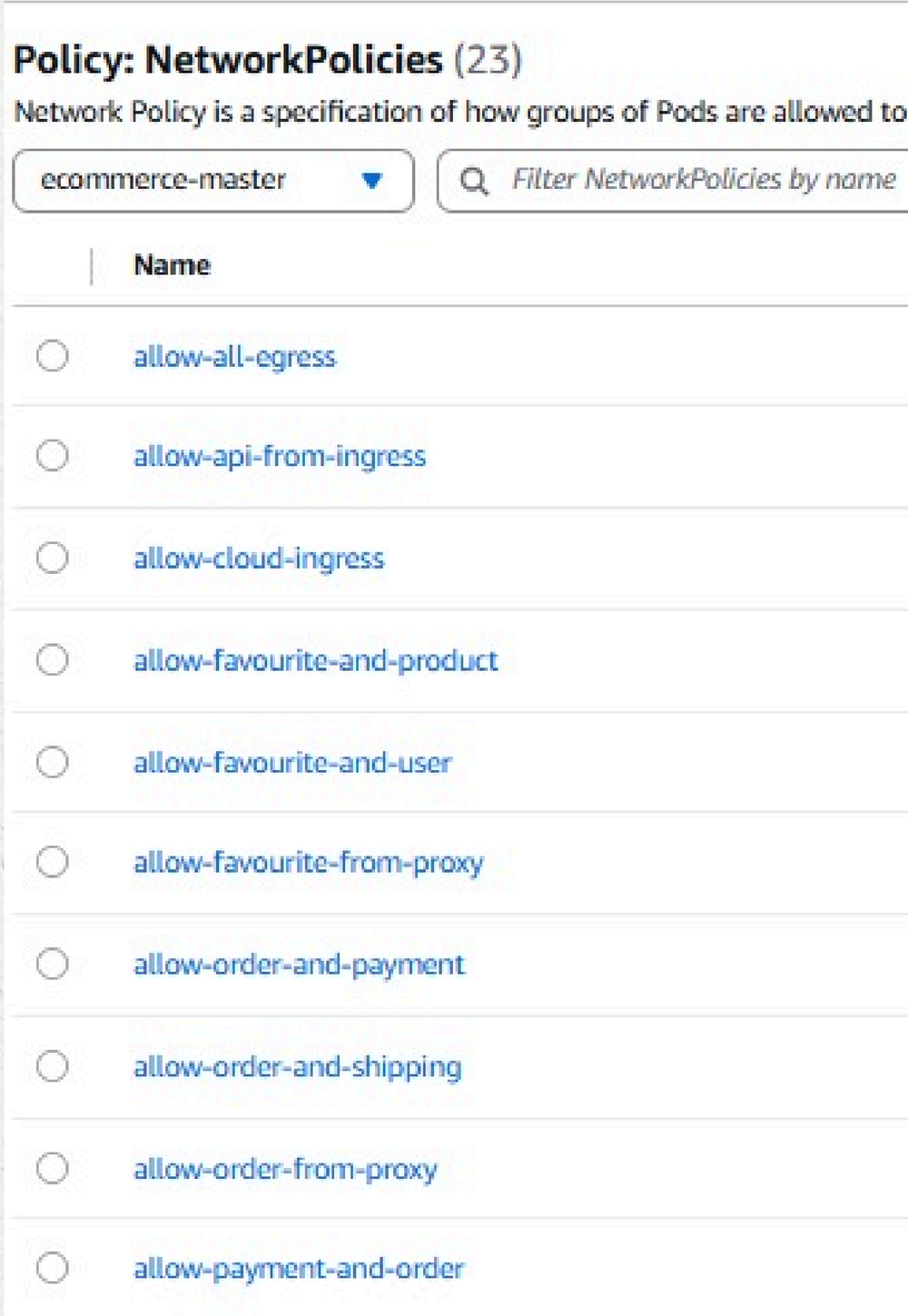
**Linkerd Service mesh**
Inyecta un contenedor a los pods para servir como proxy



# SEGURIDAD IMPLEMENTADA

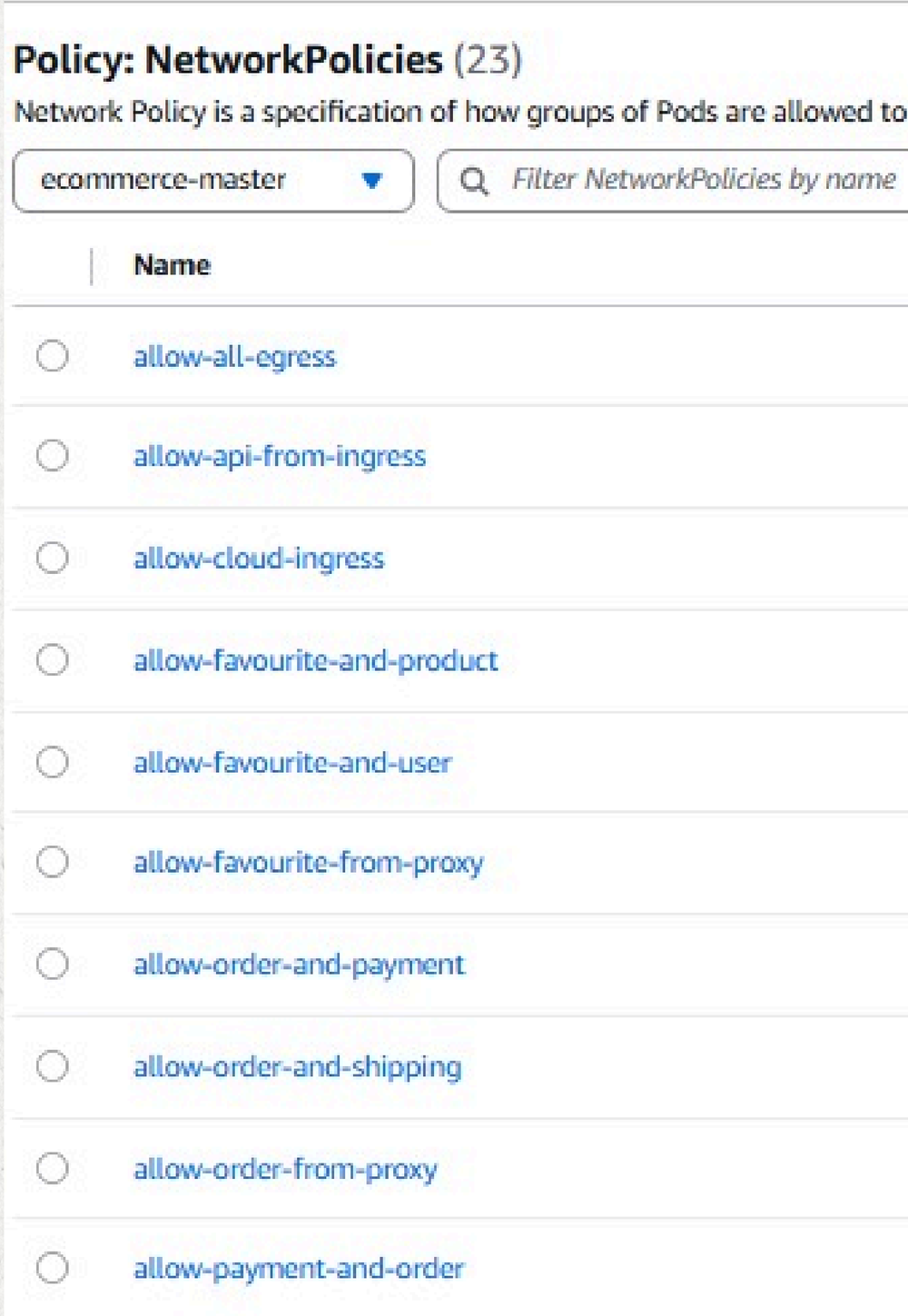
# NETWORKPOLICIES

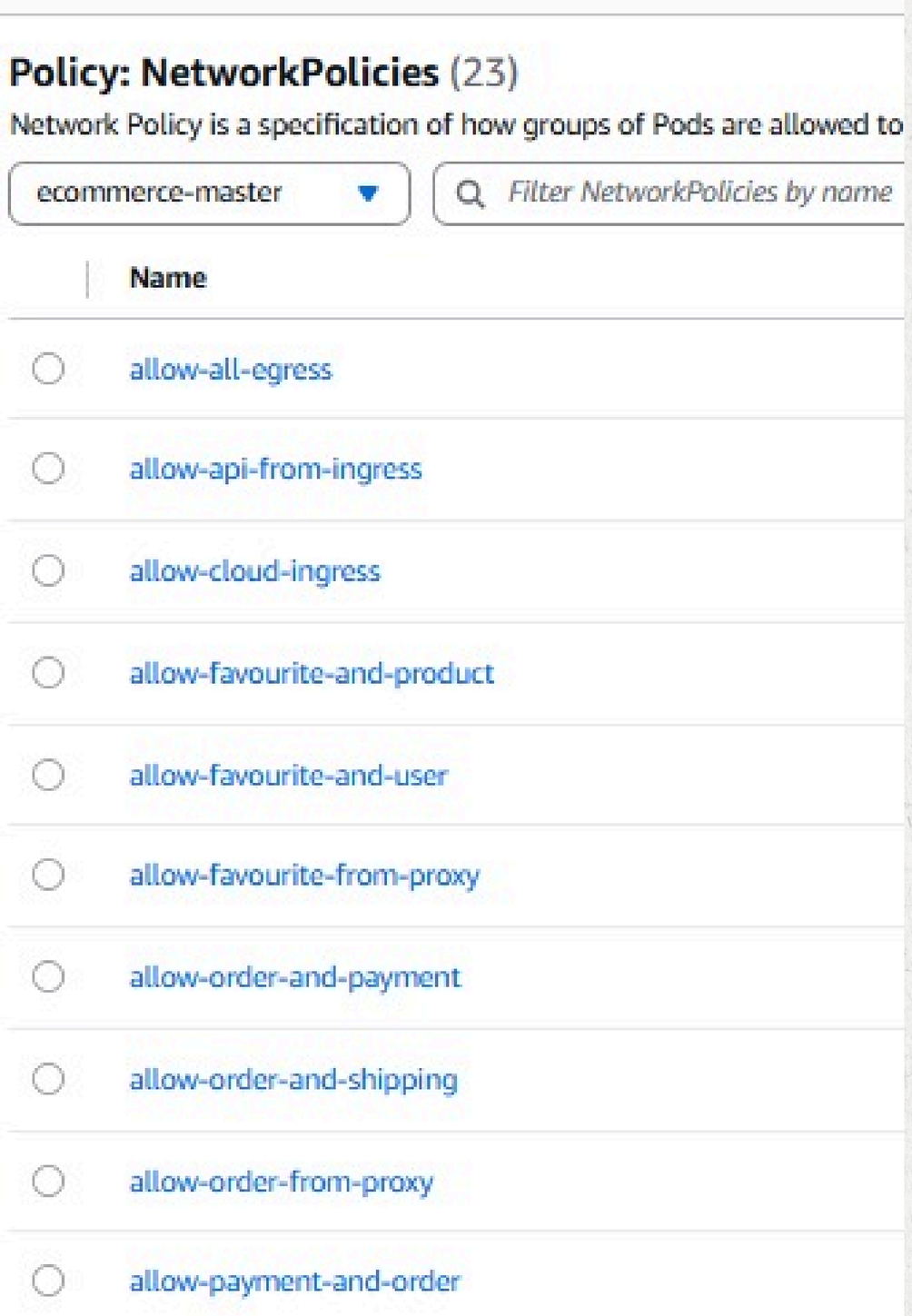
Debido a la necesidad de cumplir con el principio de mínimo privilegio se implementaron Network Policies usando Calico



# AISLAMIENTO CONTROLADO ENTRE PODS

Se implementó la política allow-all-egress, que permite a los pods enviar tráfico saliente sin restricciones, pero bloquea el tráfico entrante (Ingress) por defecto. Esto establece un aislamiento inicial y obliga a definir explícitamente qué conexiones entrantes son válidas entre servicios.






# PERMISOS ESPECÍFICOS ENTRE MICROSERVICIOS

Se definieron políticas Ingress específicas para controlar qué pods pueden comunicarse entre sí y en qué puertos.
Ejemplos clave:

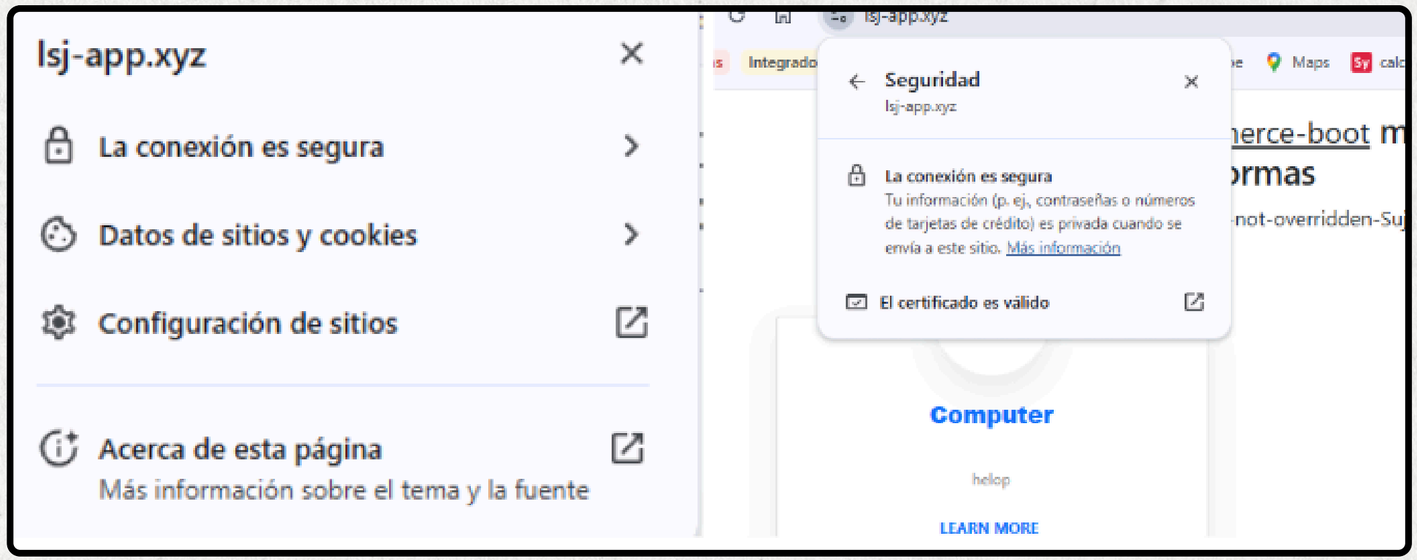
- api-gateway: solo accesible desde api-gateway-ingress (puerto 8080).
- proxy-client: accesible desde api-gateway (8900).
- favourite-service y product-service: comunicación mutua + acceso desde proxy-client.

# CERTIFICADOS TLS

Emitidos automáticamente por Cert-Manager usando Let's Encrypt. Cert-Manager se instaló con Helm y se definió un clusterIssuer para emitir los certificados.

```yaml
domains:
  frontend:
    host: www.lsj-app.xyz
    tlsSecret: frontend-tls
  api:
    host: api.lsj-app.xyz
    tlsSecret: api-gateway-tls
```

Se aplicaron en frontend y api-gateway, usando secretos TLS y dominios definidos en values-master.yaml, y se usó GoDaddy (www.lsj-app.xyz), configurando registros CNAME hacia la IP pública del Load Balancer

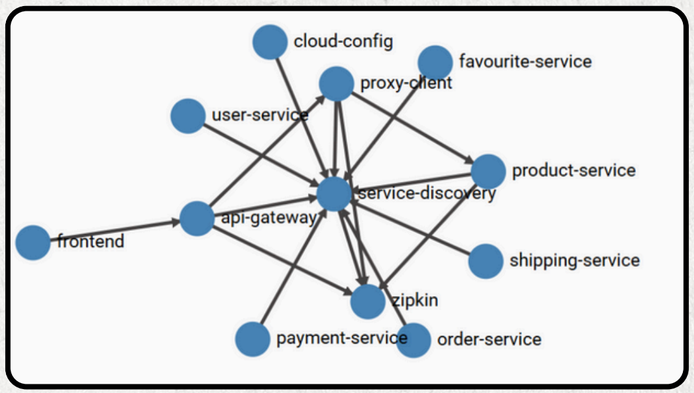


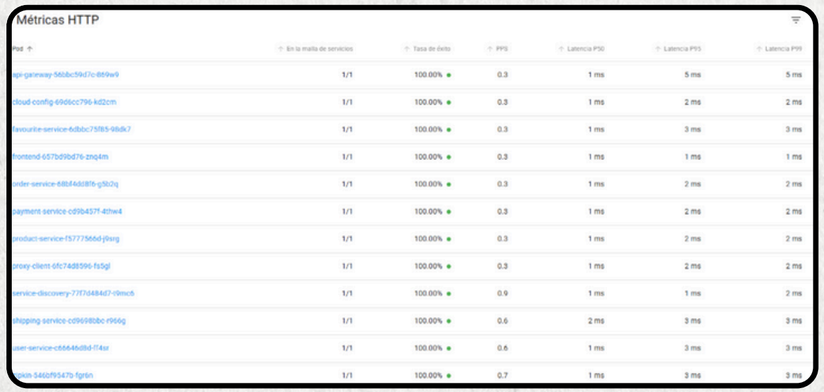
# ROTACIÓN DE SECRETOS:

Se implementó una rotación automatizada del secreto frontend-secret, visible en el frontend, se utilizó un CronJob que, en cada ejecución programada, crea un Job encargado de generar un nuevo valor para el secreto, actualizarlo en Kubernetes y reiniciar los pods del frontend para que tomen el nuevo valor. Este proceso cuenta con un ServiceAccount que gestiona los permisos necesarios.

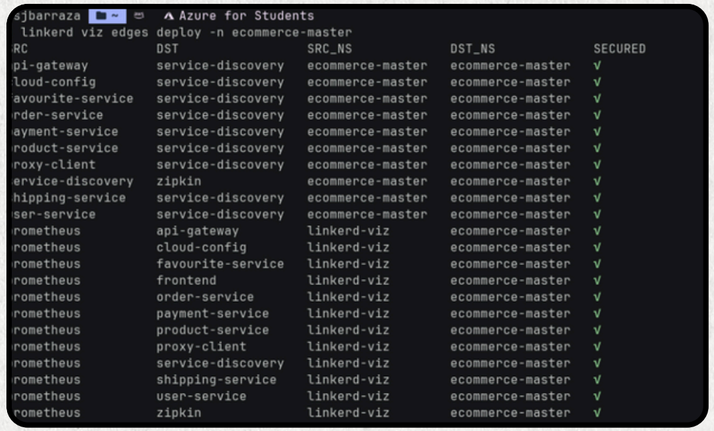
# SERVICE MESH (LINKERD)

Para garantizar la seguridad, observabilidad y control del tráfico entre microservicios, se implementó Linkerd como Service Mesh dentro del clúster Kubernetes. Esta herramienta permite inyectar proxies sidecar de forma automática en cada pod, lo que habilita características avanzadas como mTLS (Mutual TLS), balanceo de carga, métricas y trazabilidad, sin necesidad de modificar el código de los servicios.







# CALIDAD DE CODIGO

# TESTING DE LOS MICROSERVICIOS

```java
private UserDto buildValidUser(String email) {
    CredentialDto cred = CredentialDto.builder()
        .username("testuser")
        .password("1234")
        .isEnabled(true)
        .isAccountNonExpired(true)
        .isAccountNonLocked(true)
        .isCredentialsNonExpired(true)
        .build();
    return UserDto.builder()
        .firstName("Test")
        .lastName("User")
        .email(email)
        .credentialDto(cred)
        .build();
}
```

Se implementaron tests unitarios, de integración y E2E dentro de los microservicios user-service, order-service y product-service. Estos tests se ejecutan en la pipeline.

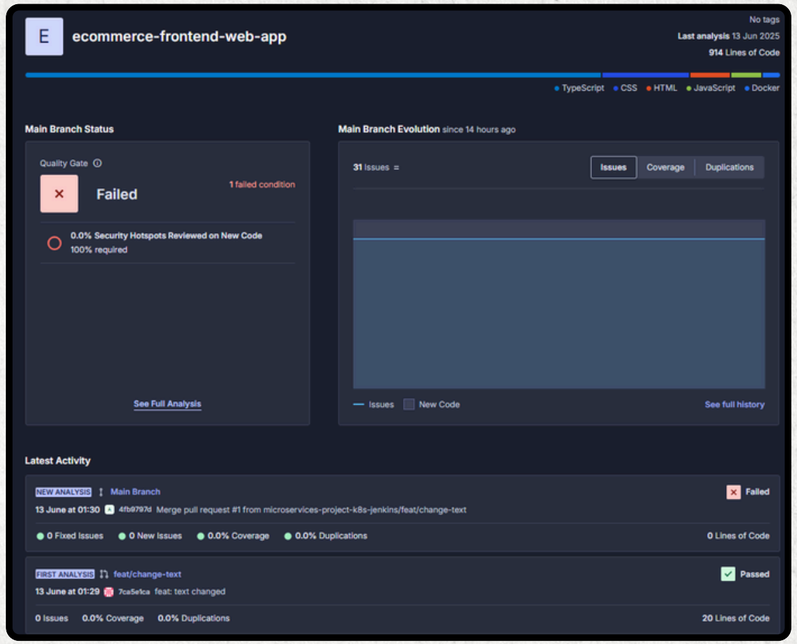
```java
@Test
void testCreateUser() throws Exception {
    UserDto userDto = buildValidUser("test@mail.com");
    mockMvc.perform(post("/api/users")
            .contentType(MediaType.APPLICATION_JSON)
            .content(objectMapper.writeValueAsString(userDto)))
            .andExpect(status().isOk())
            .andExpect(jsonPath("$.email").value("test@mail.com"));
}
```

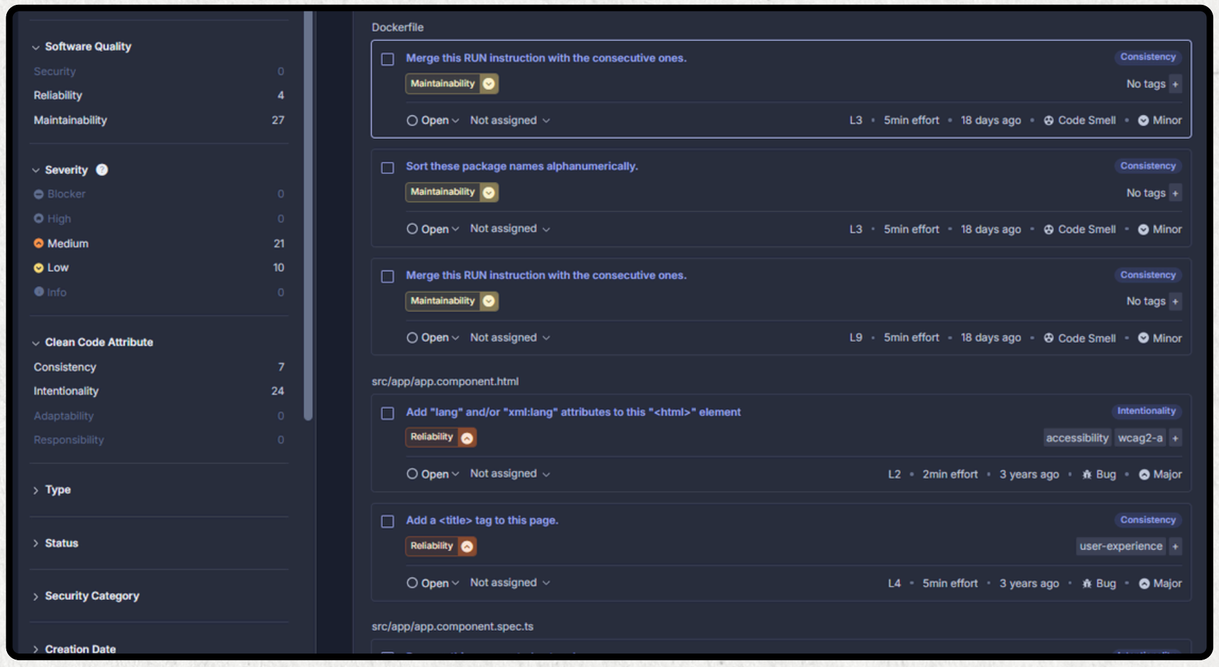
```java
@Test
void testSaveUser() {
    UserDto userDto = buildValidUser("test@mail.com");
    when(userRepository.save(any())).thenReturn(UserMappingHelper.map(userDto));
    UserDto result = userService.save(userDto);
    assertNotNull(result);
    assertEquals("test@mail.com", result.getEmail());
}
```

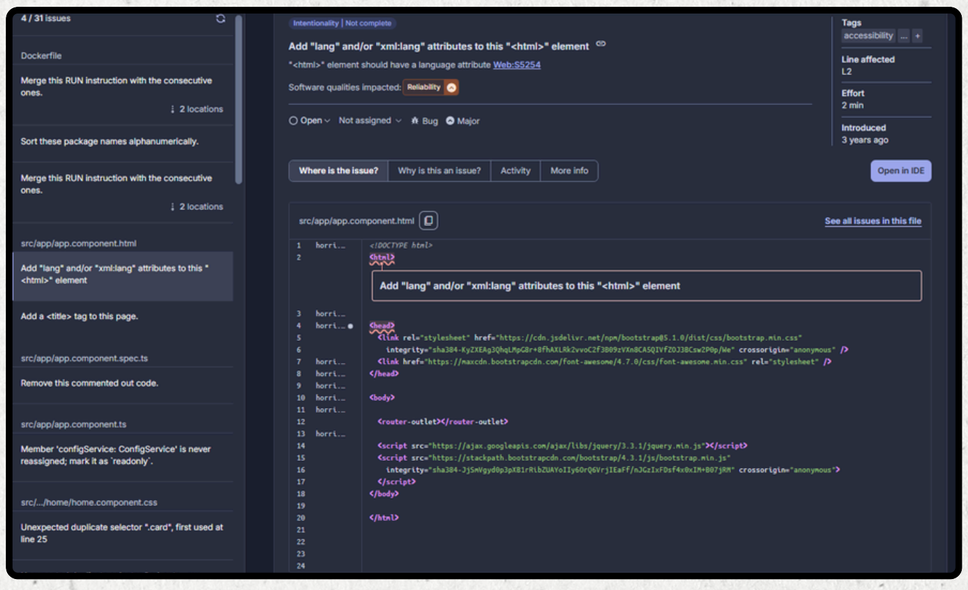
```java
@Test
void testUserRegisters() {
    UserDto user = buildValidUser("test@mail.com");
    ResponseEntity<UserDto> response = restTemplate.postForEntity("/api/users", user, UserDto.class);
    assertEquals(200, response.getStatusCodeValue());
    assertNotNull(response.getBody());
}
```

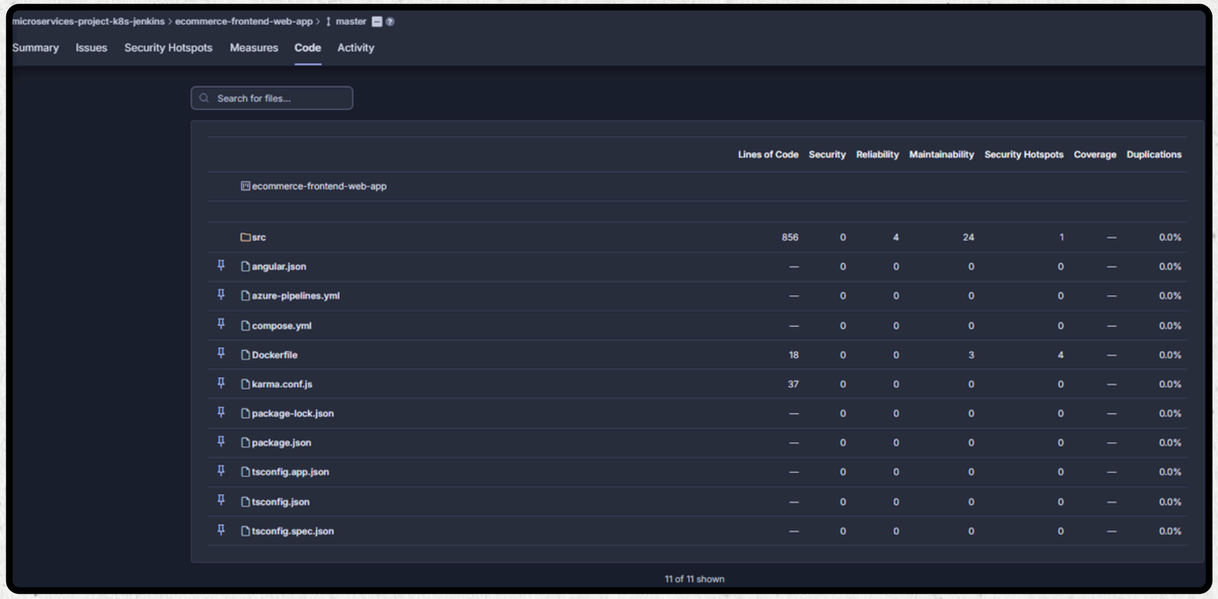
sonarqube

# CALIDAD DE CÓDIGO

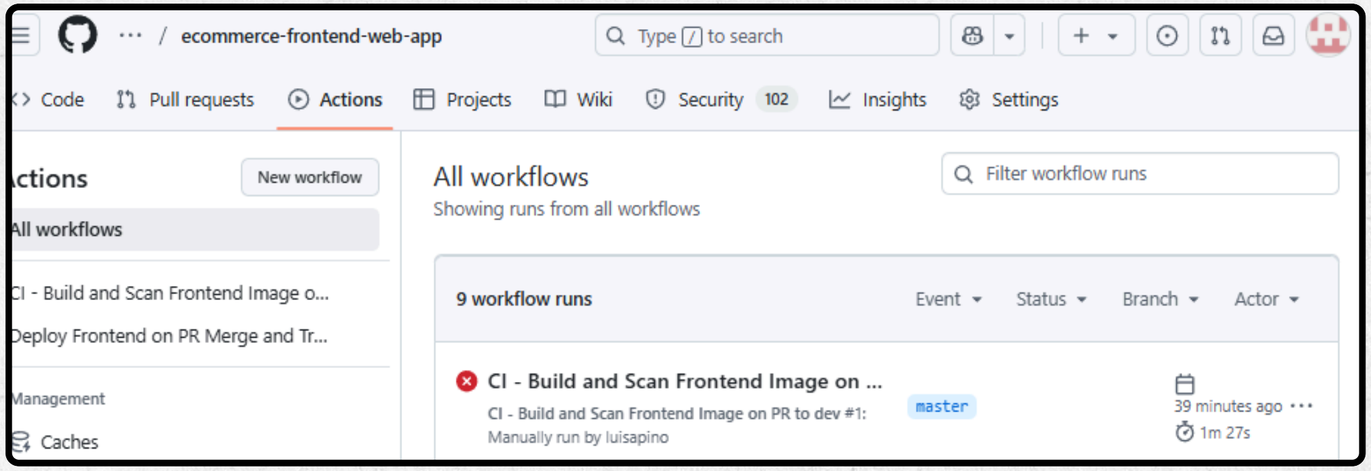


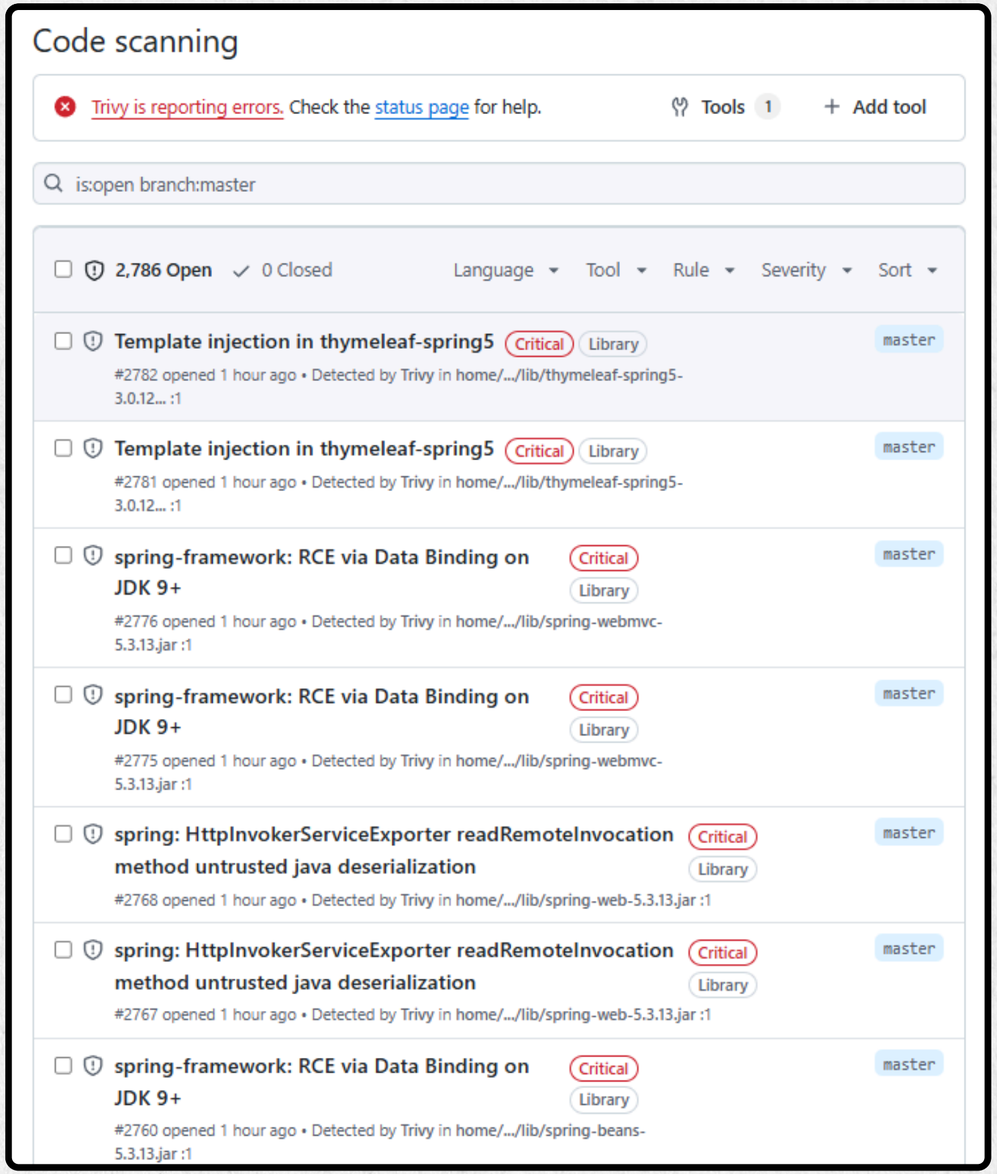
# ESCANEO DE VULNERABILIDADES




```
https://trivy.dev/v0.63/docs/scanner/vulnerability#severity-selection for details.
Error: Trivy found critical vulnerabilities in product-service. The build will be marked as
failed.
--------------------------------------------------
Processing service: shipping-service
```

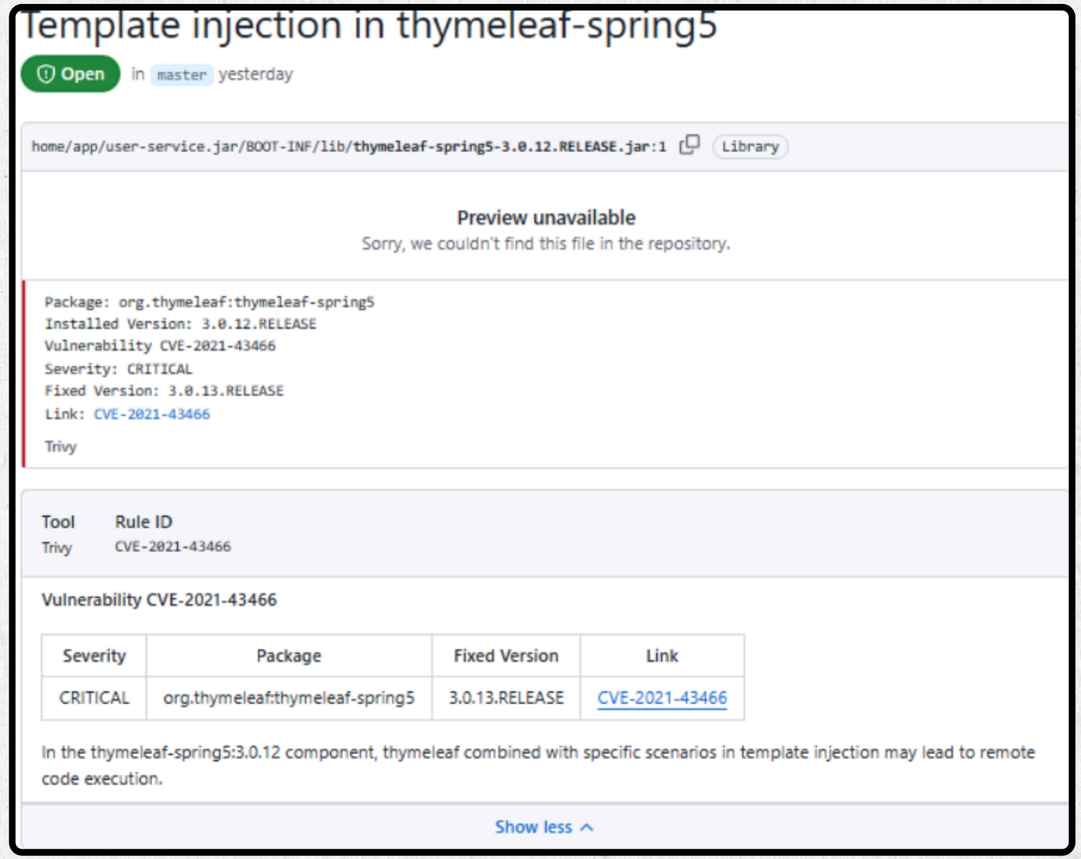
```
https://trivy.dev/v0.63/docs/scanner/vulnerability#severity-selection for details.
Error: Trivy found critical vulnerabilities in proxy-client. The build will be marked as failed.
--------------------------------------------------
Processing service: order-service
```

```
2025-06-11T20:50:04Z    WARN    Using severities from other vendors for some vulnerabilities. Read
https://trivy.dev/v0.63/docs/scanner/vulnerability#severity-selection for details.
Error: Trivy found critical vulnerabilities in api-gateway. The build will be marked as failed.
--------------------------------------------------
Processing service: proxy-client
--------------------------------------------------
```



# ESCANEO DE VULNERABILIDADES

aqua trivy



Template injection in thymeleaf-spring5

Open in master yesterday

home/app/user-service.jar/BOOT-INF/lib/**thymeleaf-spring5-3.0.12.RELEASE.jar**:1 Library

**Preview unavailable**

Sorry, we couldn't find this file in the repository.

```
Package: org.thymeleaf:thymeleaf-spring5
Installed Version: 3.0.12.RELEASE
Vulnerability CVE-2021-43466
Severity: CRITICAL
Fixed Version: 3.0.13.RELEASE
Link: CVE-2021-43466
```

Trivy

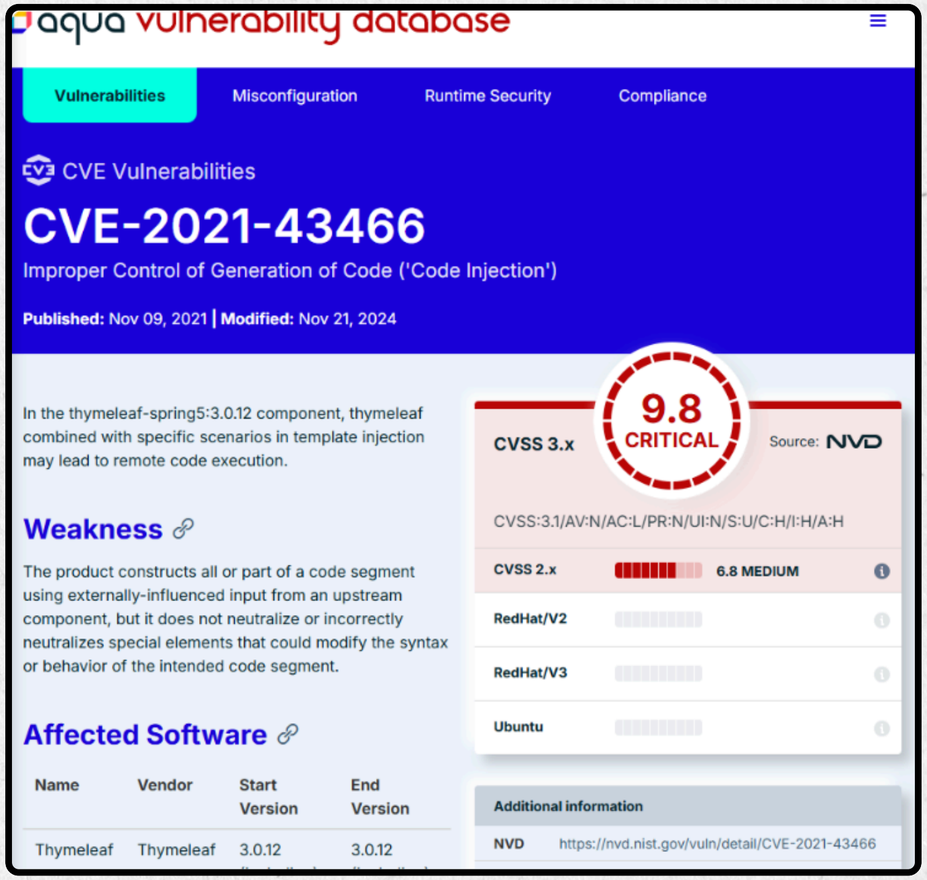
| Tool | Rule ID |
|---|---|
| Trivy | CVE-2021-43466 |

Vulnerability CVE-2021-43466

| Severity | Package | Fixed Version | Link |
|---|---|---|---|
| CRITICAL | org.thymeleaf:thymeleaf-spring5 | 3.0.13.RELEASE | CVE-2021-43466 |

In the thymeleaf-spring5:3.0.12 component, thymeleaf combined with specific scenarios in template injection may lead to remote code execution.

Show less



aqua vulnerability database

Vulnerabilities | Misconfiguration | Runtime Security | Compliance

CVE Vulnerabilities

**CVE-2021-43466**

Improper Control of Generation of Code ('Code Injection')

**Published:** Nov 09, 2021 | **Modified:** Nov 21, 2024

In the thymeleaf-spring5:3.0.12 component, thymeleaf combined with specific scenarios in template injection may lead to remote code execution.

**CVSS 3.x** 9.8 CRITICAL Source: NVD

CVSS:3.1/AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:H

**CVSS 2.x** 6.8 MEDIUM

RedHat/V2

RedHat/V3

Ubuntu

## Weakness

The product constructs all or part of a code segment using externally-influenced input from an upstream component, but it does not neutralize or incorrectly neutralizes special elements that could modify the syntax or behavior of the intended code segment.

## Affected Software

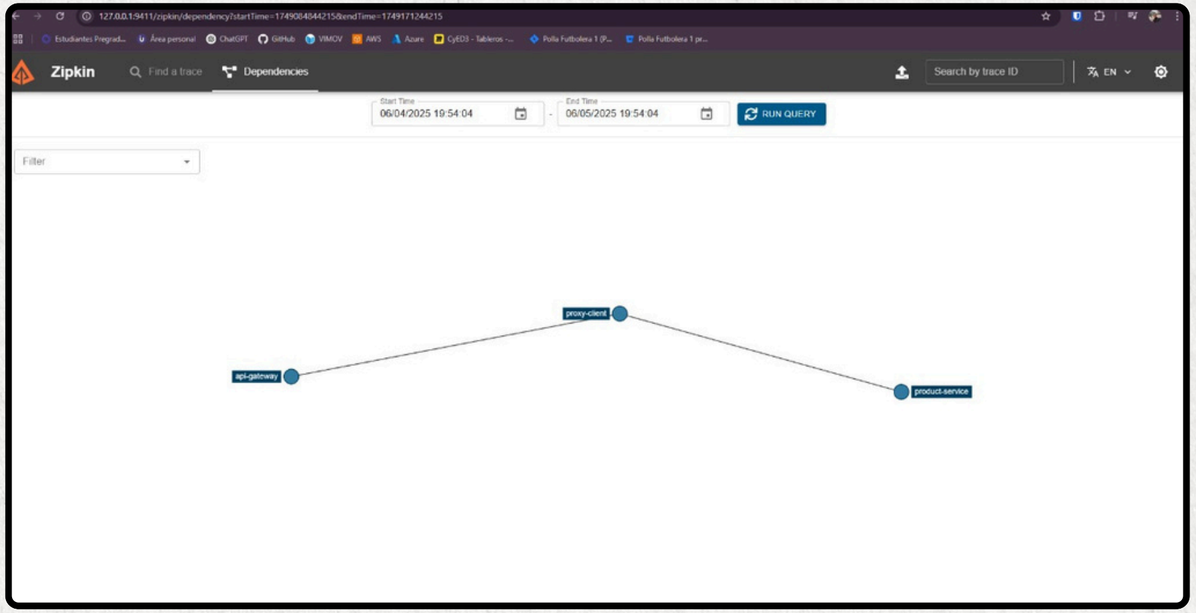
| Name | Vendor | Start Version | End Version |
|---|---|---|---|
| Thymeleaf | Thymeleaf | 3.0.12 | 3.0.12 |

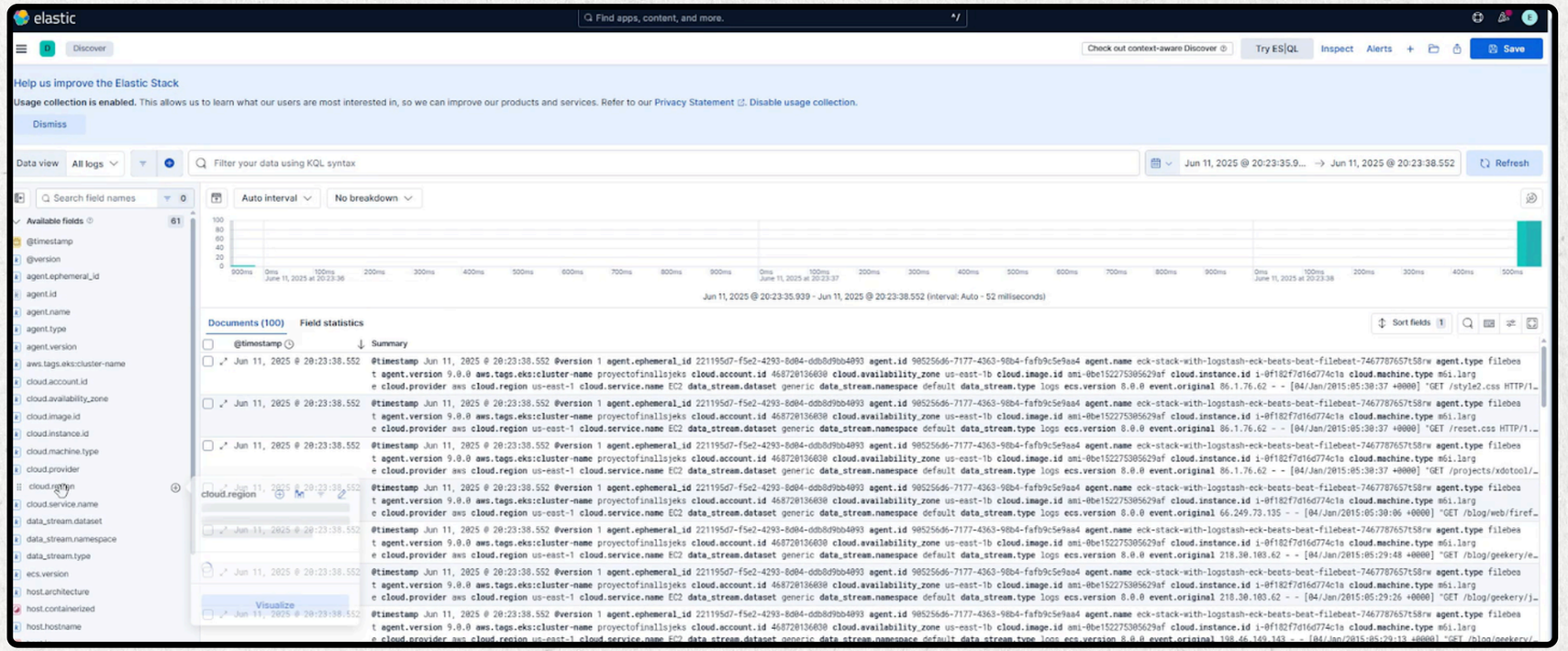
**Additional information**

NVD https://nvd.nist.gov/vuln/detail/CVE-2021-43466

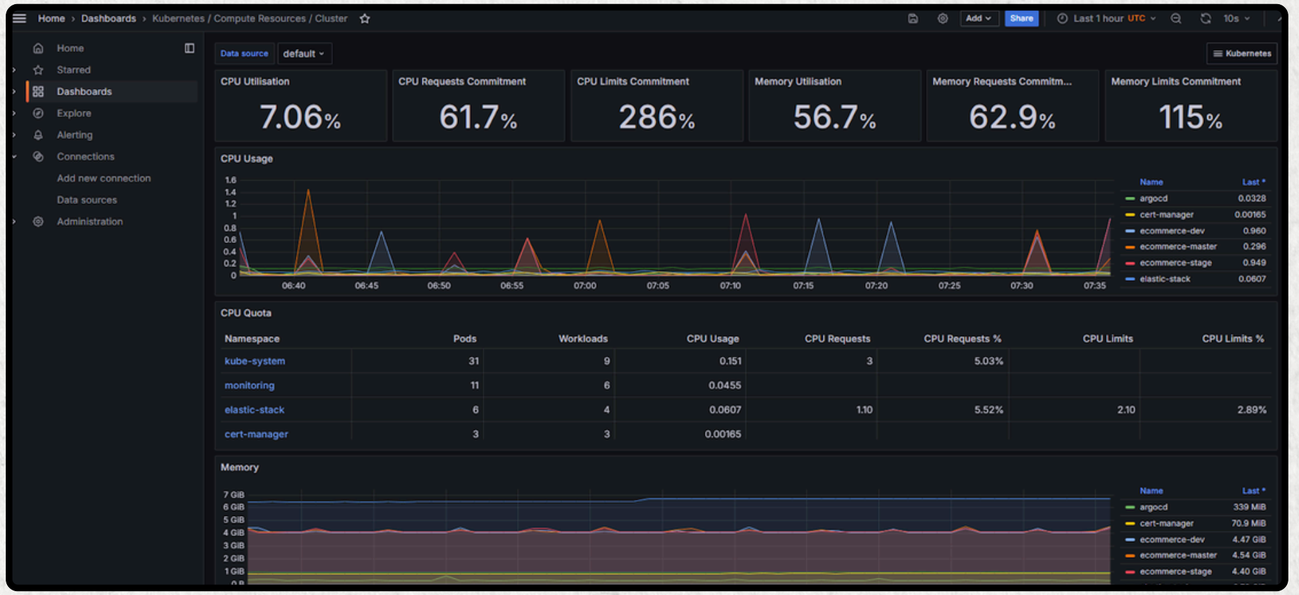
# OBSERVABILIDAD Y MONITOREO

# ELASTIC Y ZIPKIN



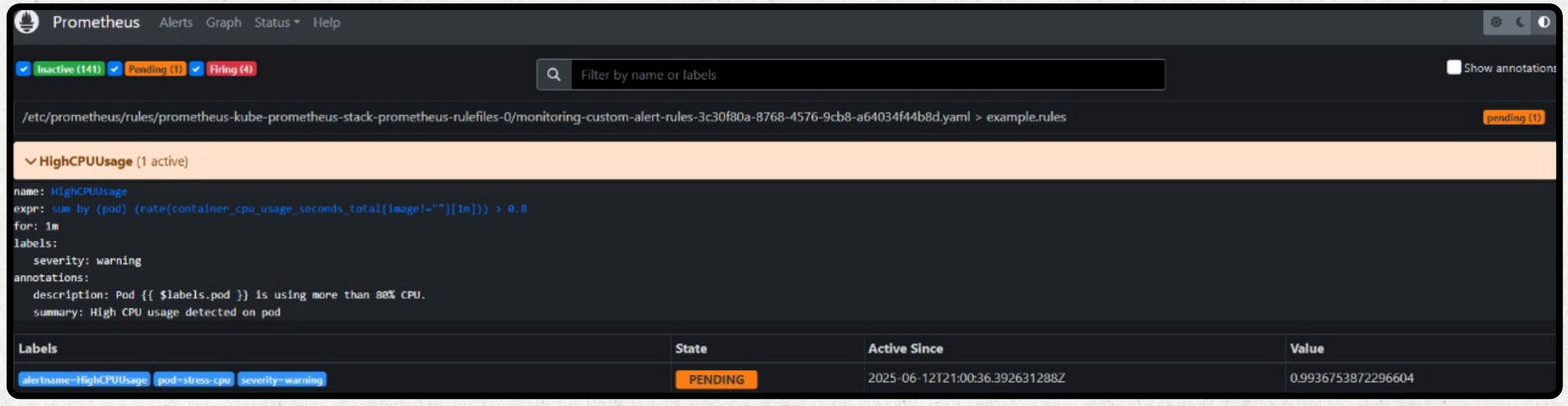



# PROMETHEUS Y GRAFANA

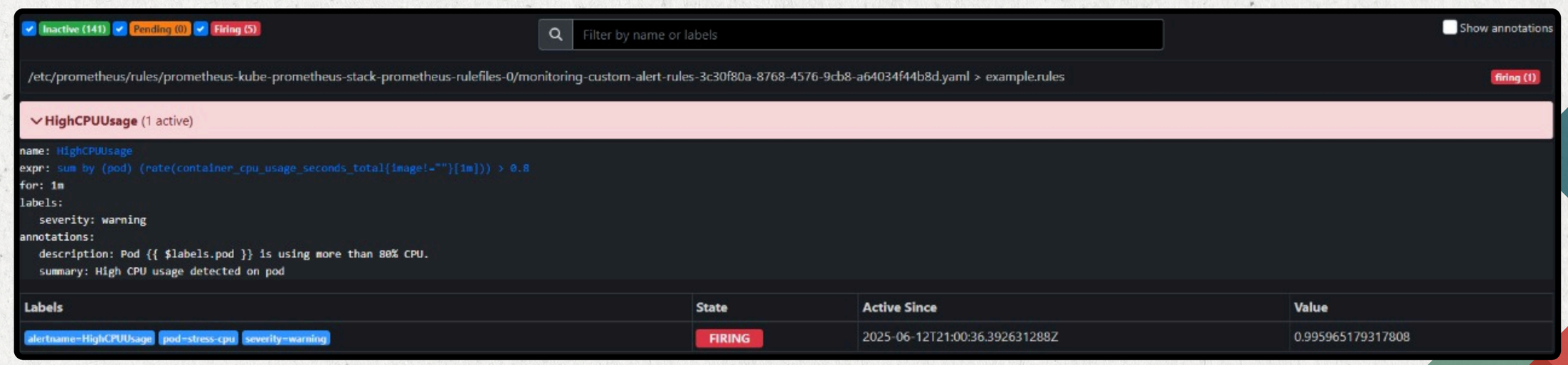




# ALERTAS




```yaml
apiVersion: v1
kind: Pod
metadata:
  name: stress-cpu
spec:
  containers:
  - name: stress
    image: progrium/stress
    args: ["--cpu", "1"]
    resources:
      limits:
        cpu: "1"
      requests:
        cpu: "0.5"
```



# ESCALABILIDAD Y RESILIENCIA

# IMPLEMENTACIÓN DE HORIZONTAL POD AUTOSCALER

```
kubectl get pods -n ecommerce-dev
NAME                                        READY   STATUS        RESTARTS   AGE
api-gateway-78899d5f7b-6rj47                1/1     Running       0          8m20s
cloud-config-868d8b7bf5-xshkb               1/1     Running       0          8m20s
favourite-service-65d7457cd6-hvtft          1/1     Running       0          8m20s
frontend-6f4cf9864c-mlrww                   1/1     Terminating   0          6m12s
frontend-756c794fbd-55z5z                   1/1     Running       0          106s
frontend-secret-rotator-cronjob-29162680-ckn98   0/1     Completed     0          11m
frontend-secret-rotator-cronjob-29162685-n4hlj   0/1     Completed     0          6m47s
frontend-secret-rotator-cronjob-29162690-kfx4c   0/1     Completed     0          107s
order-service-857987df75-pvgkj              1/1     Running       0          8m20s
payment-service-5b65fdfff6-wnjdf            1/1     Running       0          8m20s
product-service-d549b7975-79tsq             1/1     Running       0          8m20s
proxy-client-7f44fbfbfc-nfxgh               1/1     Running       0          8m20s
service-discovery-8888f8954-65b8n           1/1     Running       0          8m20s
shipping-service-7b57bddc68-98stp           1/1     Running       0          8m20s
```

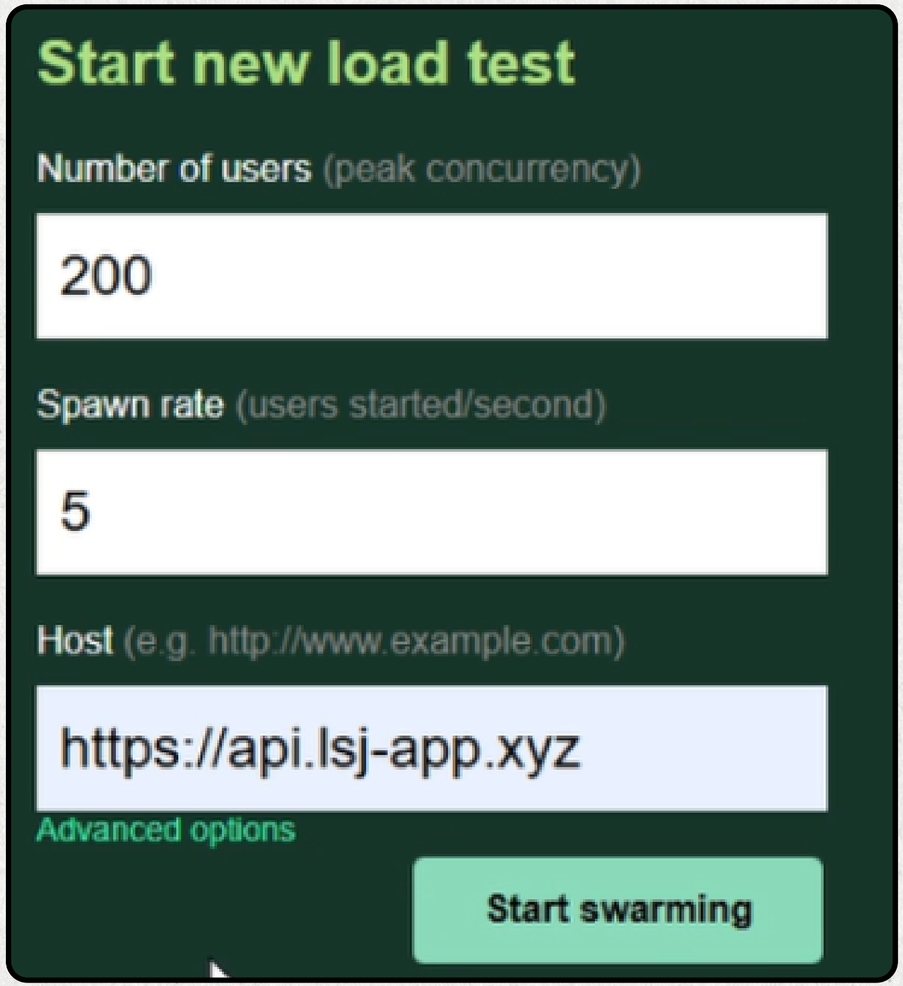
```
kubectl delete pod shipping-service-7b57bddc68-98stp -n ecommerce-dev
pod "shipping-service-7b57bddc68-98stp" deleted

sjbarraza ~  Azure for Students
kubectl get pods -n ecommerce-dev
NAME                                        READY   STATUS      RESTARTS   AGE
api-gateway-78899d5f7b-6rj47                1/1     Running     0          9m21s
cloud-config-868d8b7bf5-xshkb               1/1     Running     0          9m21s
favourite-service-65d7457cd6-hvtft          1/1     Running     0          9m21s
frontend-756c794fbd-55z5z                   1/1     Running     0          2m47s
frontend-secret-rotator-cronjob-29162680-ckn98   0/1     Completed   0          12m
frontend-secret-rotator-cronjob-29162685-n4hlj   0/1     Completed   0          7m48s
frontend-secret-rotator-cronjob-29162690-kfx4c   0/1     Completed   0          2m48s
order-service-857987df75-pvgkj              1/1     Running     0          9m21s
payment-service-5b65fdfff6-wnjdf            1/1     Running     0          9m21s
product-service-d549b7975-79tsq             1/1     Running     0          9m21s
proxy-client-7f44fbfbfc-nfxgh               1/1     Running     0          9m21s
service-discovery-8888f8954-65b8n           1/1     Running     0          9m21s
shipping-service-7b57bddc68-vij77           1/1     Running     0          8s
```

```yaml
apiVersion: autoscaling/v2
kind: HorizontalPodAutoscaler
metadata:
  name: user-service
spec:
  scaleTargetRef:
    apiVersion: apps/v1
    kind: Deployment
    name: user-service
  minReplicas: 1
  maxReplicas: 2
  metrics:
    - type: Resource
      resource:
        name: cpu
        target:
          type: Utilization
          averageUtilization: 50
```

# PRUEBAS DE RENDIMIENTO

200 usuarios como peak

Aumentan de 5 en 5




Solicitudes a la API cifrada con TLS

# PRUEBAS DE RENDIMIENTO

Reporte en Locust

## Locust Test Report

**During:** 11/6/2025, 5:22:50 p.m. - 11/6/2025, 5:26:24 p.m.

**Target Host:** https://api.lsj-app.xyz

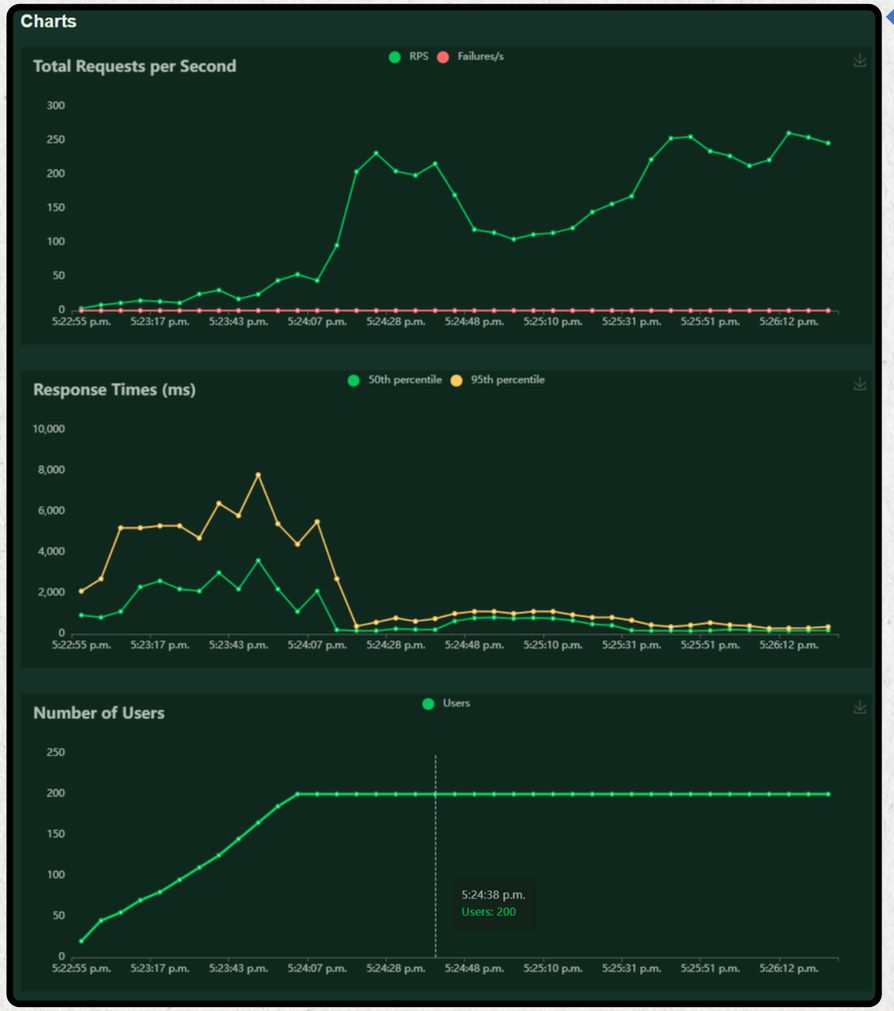
**Script:** locustfile.py

### Request Statistics

| Method | Name | # Requests | # Fails | Average (ms) | Min (ms) | Max (ms) | Average size (bytes) | RPS | Failures/s |
|---|---|---|---|---|---|---|---|---|---|
| GET | /favourite-service/api/favourites | 6706 | 0 | 467 | 95 | 7892 | 1589 | 31.4 | 0.0 |
| GET | /order-service/api/orders | 7014 | 0 | 469 | 88 | 9773 | 524 | 32.8 | 0.0 |
| GET | /order-service/api/orders/999999 | 391 | 0 | 378 | 93 | 4021 | 121 | 1.8 | 0.0 |
| GET | /product-service/api/products | 6954 | 0 | 460 | 89 | 9771 | 708 | 32.6 | 0.0 |
| GET | /user-service/api/users | 7082 | 0 | 479 | 90 | 7917 | 3999 | 33.2 | 0.0 |
| GET | /user-service/api/users/999999 | 391 | 0 | 635 | 94 | 7379 | 120 | 1.8 | 0.0 |
| | **Aggregated** | **28538** | **0** | **470** | **88** | **9773** | **1670** | **133.6** | **0.0** |

### Response Time Statistics

| Method | Name | 50%ile (ms) | 60%ile (ms) | 70%ile (ms) | 80%ile (ms) | 90%ile (ms) | 95%ile (ms) | 99%ile (ms) | 100%ile (ms) |
|---|---|---|---|---|---|---|---|---|---|
| GET | /favourite-service/api/favourites | 240 | 300 | 430 | 640 | 820 | 1100 | 4100 | 7900 |
| GET | /order-service/api/orders | 200 | 250 | 400 | 630 | 860 | 1200 | 4400 | 9800 |
| GET | /order-service/api/orders/999999 | 190 | 230 | 340 | 550 | 860 | 1000 | 3000 | 4000 |
| GET | /product-service/api/products | 200 | 250 | 390 | 640 | 830 | 1200 | 4300 | 9800 |
| GET | /user-service/api/users | 220 | 280 | 410 | 640 | 830 | 1200 | 4500 | 7900 |
| GET | /user-service/api/users/999999 | 250 | 360 | 570 | 750 | 1200 | 2700 | 5400 | 7400 |
| | **Aggregated** | **210** | **270** | **410** | **640** | **840** | **1200** | **4300** | **9800** |



# PRUEBAS DE RENDIMIENTO

Visualización en Grafana

# CHAOS MESH

## Implementación

```
sjbarraza ~  Azure for Students
 kubectl get schedule -n ecommerce-master
AME                              AGE
elay-product-service-schedule    3m54s

sjbarraza ~  Azure for Students
 kubectl get networkchaos -n ecommerce-master
AME                                      ACTION    DURATION
elay-product-service-schedule-6zxwh      delay     1m
```

## Antes

Métricas HTTP

| Deployment | En la malla de servicios | Tasa de éxito | PPS | Latencia P50 | Latencia P95 | Latencia P99 |
|---|---|---|---|---|---|---|
| api-gateway | 1/1 | 100.00% | 0.3 | 1 ms | 9 ms | 10 ms |
| cloud-config | 1/1 | 100.00% | 0.3 | 1 ms | 3 ms | 3 ms |
| favourite-service | 1/1 | 10.41% | 12.48 | 5 ms | 11 ms | 18 ms |
| frontend | 1/1 | 100.00% | 0.3 | 1 ms | 1 ms | 1 ms |
| order-service | 1/1 | 100.00% | 12.22 | 1 ms | 4 ms | 6 ms |
| payment-service | 1/1 | 100.00% | 0.3 | 1 ms | 2 ms | 2 ms |
| product-service | 1/1 | 100.00% | 5.43 | 1 ms | 3 ms | 16 ms |
| proxy-client | 1/1 | 100.00% | 0.3 | 1 ms | 3 ms | 3 ms |
| service-discovery | 1/1 | 100.00% | 0.93 | 1 ms | 1 ms | 2 ms |
| shipping-service | 1/1 | 100.00% | 0.6 | 2 ms | 8 ms | 10 ms |
| user-service | 1/1 | 100.00% | 29.03 | 1 ms | 2 ms | 5 ms |
| zipkin | 1/1 | 100.00% | 4.58 | 1 ms | 2 ms | 78 ms |

## Después

Métricas HTTP

| Deployment | En la malla de servicios | Tasa de éxito | PPS | Latencia P50 | Latencia P95 | Latencia P99 |
|---|---|---|---|---|---|---|
| api-gateway | 1/1 | 100.00% | 0.3 | 1 ms | 5 ms | 5 ms |
| cloud-config | 1/1 | 100.00% | 0.3 | 1 ms | 2 ms | 2 ms |
| favourite-service | 1/1 | 100.00% | 18.53 | 16 ms | 34 ms | 81 ms |
| frontend | 1/1 | 100.00% | 0.3 | 1 ms | 1 ms | 1 ms |
| order-service | 1/1 | 100.00% | 19.22 | 1 ms | 6 ms | 17 ms |
| payment-service | 1/1 | 100.00% | 0.3 | 1 ms | 2 ms | 2 ms |
| product-service | 1/1 | 100.00% | 73.42 | 1 ms | 2 ms | 9 ms |
| proxy-client | 1/1 | 100.00% | 0.3 | 1 ms | 2 ms | 2 ms |
| service-discovery | 1/1 | 100.00% | 0.98 | 1 ms | 2 ms | 3 ms |
| shipping-service | 1/1 | 100.00% | 0.6 | 2 ms | 3 ms | 3 ms |
| user-service | 1/1 | 100.00% | 74.5 | 1 ms | 2 ms | 10 ms |
| zipkin | 1/1 | 100.00% | 5.32 | 1 ms | 6 ms | 15 ms |

# DESAFÍOS

## DESAFÍOS CLAVE ENFRENTADOS

Inestabilidad de Nodos en AWS

Guerra de Recursos

Certificados TLS

# CONCLUSIONES

La importancia y simplicidad de Helm

Descubrimiento de ArgoCD

Vitalidad del tráfico con capa de seguridad

Importancia de herramientas de monitoreo

# ¡GRACIAS!