

# **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 implementada

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:

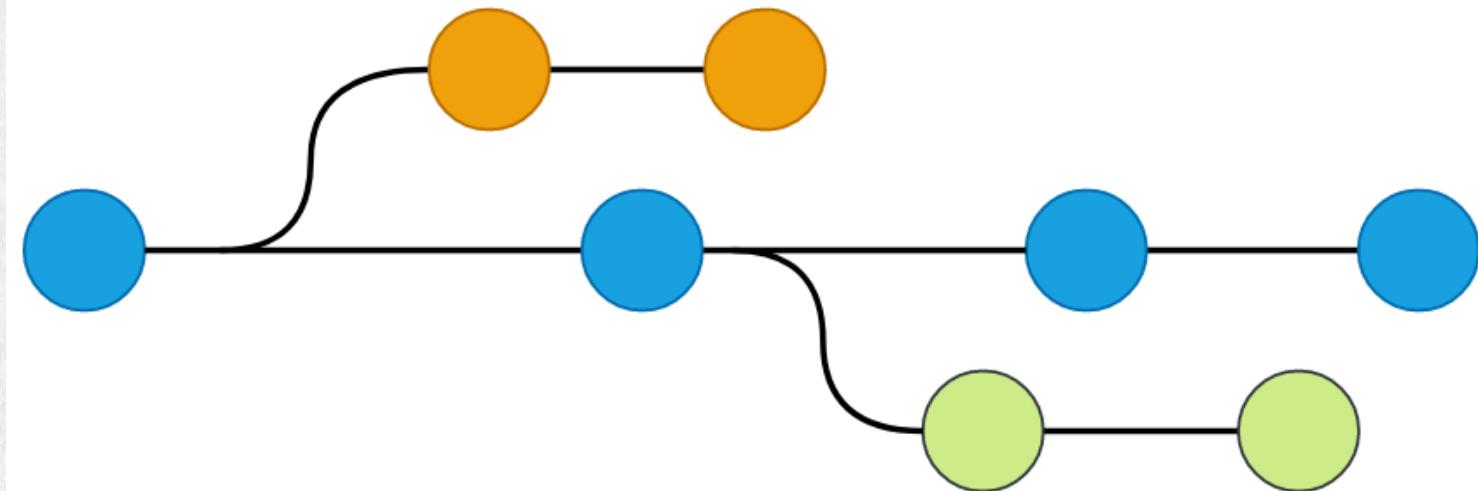
- Backlog: Tareas del proyecto
- Planned: Próximas tareas asignadas
- Doing: En curso
- Blocked: Problemas, se necesita ayuda
- Done: Finalizadas



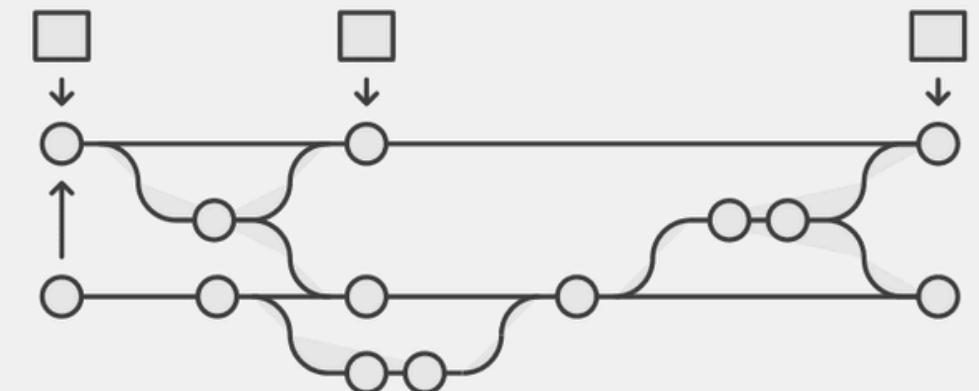
# ESTRATEGIA DE BRANCHING

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.



# Git Flow



# EVIDENCIA

Plataformas-Ingesoft-Project

View 1 | + New view

Q. Filter by keyword or by field

Backlog (8) This item hasn't been started

- infrastructure #4 Como DevOps, quiero configurar un clúster Kubernetes (Minikube, Kind o en la nube) para alojar la arquitectura del proyecto.
- ecommerce-chart #20 Como desarrollador, quiero migrar las configuraciones de Spring Boot a ConfigMaps para centralizar su manejo.
- ecommerce-chart #21 Como estudiante, quiero desplegar todos los microservicios del repositorio base en un clúster de Kubernetes para demostrar el dominio de una arquitectura completa.
- ecommerce-chart #22 Como ingeniero de plataformas, quiero usar Namespaces para separar los ambientes dev, stage y master y mantener la organización y aislamiento.
- ecommerce-chart #23 Como equipo técnico, queremos documentar la arquitectura con diagramas claros para facilitar el mantenimiento y futuras mejoras.
- ecommerce-chart #24 Como desarrollador, quiero definir las dependencias entre servicios para que renaten la arquitectura monolítica del sistema.

Planned (0)

Doing (0) This is actively being worked on

Blocked (0)

Done (0) This has been completed

+ Add item

Plataformas-Ingesoft-Project

View 1 | + New view

Q. Filter by keyword or by field

Backlog (0) This item hasn't been started

Planned (23)

- infrastructure #5 Como DevOps, quiero desplegar el sistema en un proveedor cloud como AWS.
- ecommerce-chart #20 Como DevSecOps, quiero escanear las imágenes de los contenedores en busca de vulnerabilidades.
- ecommerce-chart #22 Como desarrollador, quiero usar los Actuator endpoints de Spring Boot para exponer métricas útiles.
- ecommerce-chart #23 Como ingeniero de plataformas, quiero usar Namespaces para separar los ambientes dev, stage y master y mantener la organización y aislamiento.
- ecommerce-chart #20 Como desarrollador, quiero migrar las configuraciones de Spring Boot a ConfigMaps para centralizar su manejo.
- ecommerce-chart #26 Como DevOps, quiero usar Helm Charts para empaquetar e instalar los microservicios correctamente.

Doing (5) This is actively being worked on

Blocked (0)

Done (0) This has been completed

+ Add item

Plataformas-Ingesoft-Project

View 1 | + New view

Q. Filter by keyword or by field

Backlog (0) This item hasn't been started

Planned (15)

- ecommerce-chart #39 Como desarrollador, quiero usar los Actuator endpoints de Spring Boot para exponer métricas útiles.
- ecommerce-chart #43 Como DevOps, quiero implementar Horizontal Pod Autoscaler para escalar automáticamente los microservicios.
- ecommerce-chart #29 Como administrador, quiero asignar ServiceAccounts con permisos mínimos usando RBAC para mejorar la seguridad.
- ecommerce-chart #41 Como DevOps, quiero centralizar los logs usando ELK para facilitar la trazabilidad.
- ecommerce-chart #26 Como ingeniero de redes, quiero implementar un Ingress Controller para enrutar las peticiones a través del API Gateway.
- ecommerce-chart #38 Como operador, quiero configurar Prometheus y Grafana para monitorear los sistemas.

Doing (10) This is actively being worked on

Blocked (3)

- ecommerce-chart #30 Como DevSecOps, quiero escanear las imágenes de los contenedores en busca de vulnerabilidades.
- ecommerce-chart #36 Como DevOps, quiero usar Secrets para guardar credenciales y datos sensibles de forma segura.
- ecommerce-chart #37 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

Plataformas-Ingesoft-Project

View 1 | + New view

Q. 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

- ecommerce-chart #17 Como desarrollador, quiero crear Persistent Volumes y Persistent Volume Claims para asegurar el almacenamiento de las bases de datos.
- ecommerce-chart #30 Como DevSecOps, quiero escanear las imágenes de los contenedores en busca de vulnerabilidades.
- ecommerce-chart #36 Como DevOps, quiero usar Helm Charts para empaquetar e instalar los microservicios correctamente.
- infrastructure #6 Como operador, quiero usar servicios nativos cloud para optimizar la solución.
- ecommerce-chart #33 Como responsable de seguridad, quiero implementar rotación de secretos para minimizar riesgos.
- ecommerce-chart #21 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:

ecommerce-chart Public

Switch branches/tags

Find or create a branch...

Branches Tags

- ✓ main default
- feature/hu-8/network-policy
- feature/hu-9/allow-elastic-communication
- feature/hu-10/hpa-implementation
- feature/hu-11/secret-rotation
- feature/hu-12/secret-from-secrets-manager
- feature/hu-13/configmap-implementation
- feature/hu-14/ingress-controller
- feature/hu-15/argocd-applicationset

INFRASTRUCTURE:

main

Switch branches/tags

Find or create a branch...

Branches Tags

- ✓ main default
- develop
- feature/hu-1/alerts
- feature/hu-2/secrets-manager
- feature/hu-3/deploy-aws-infrastructure
- feature/hu-4/destroy-aws-infrastructure
- feature/hu-5/deploy-terraform
- feature/hu-6/destroy-terraform

ECOMMERCE-MICROSERVICE-BACKEND-APP:

ecommerce-microservice-backend-app forked from SelimHorri/ecommerce-microservice-backend-app

master

Switch branches/tags

Find or create a branch...

Branches Tags

- dev
- feature/hu-01/performance-tests-api-g...
- feature/hu-02/update-jenkinsfile
- feature/hu-03/backend-pipeline-validat...
- feature/hu-04/trivy-implementation
- feature/hu-05/deploy-to-ecr
- feature/hu-06/locust-implementation
- feature/hu-07/create-backend-pr-checks
- feature/hu-08/scan-images
- stage

ECOMMERCE-FRONTEND-WEB-APP:

Commits

master

Switch branches/tags

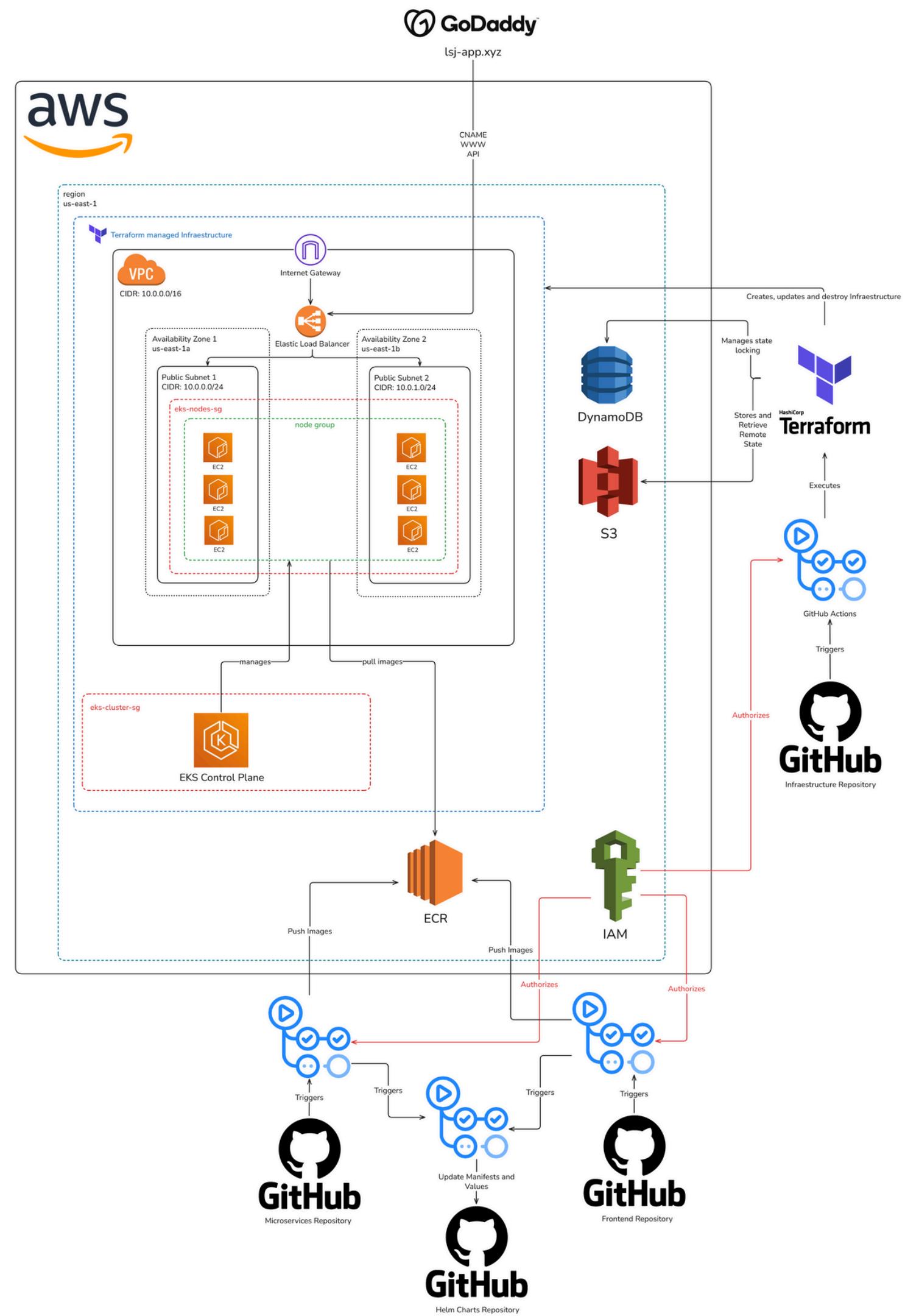
Find a branch...

Branches Tags

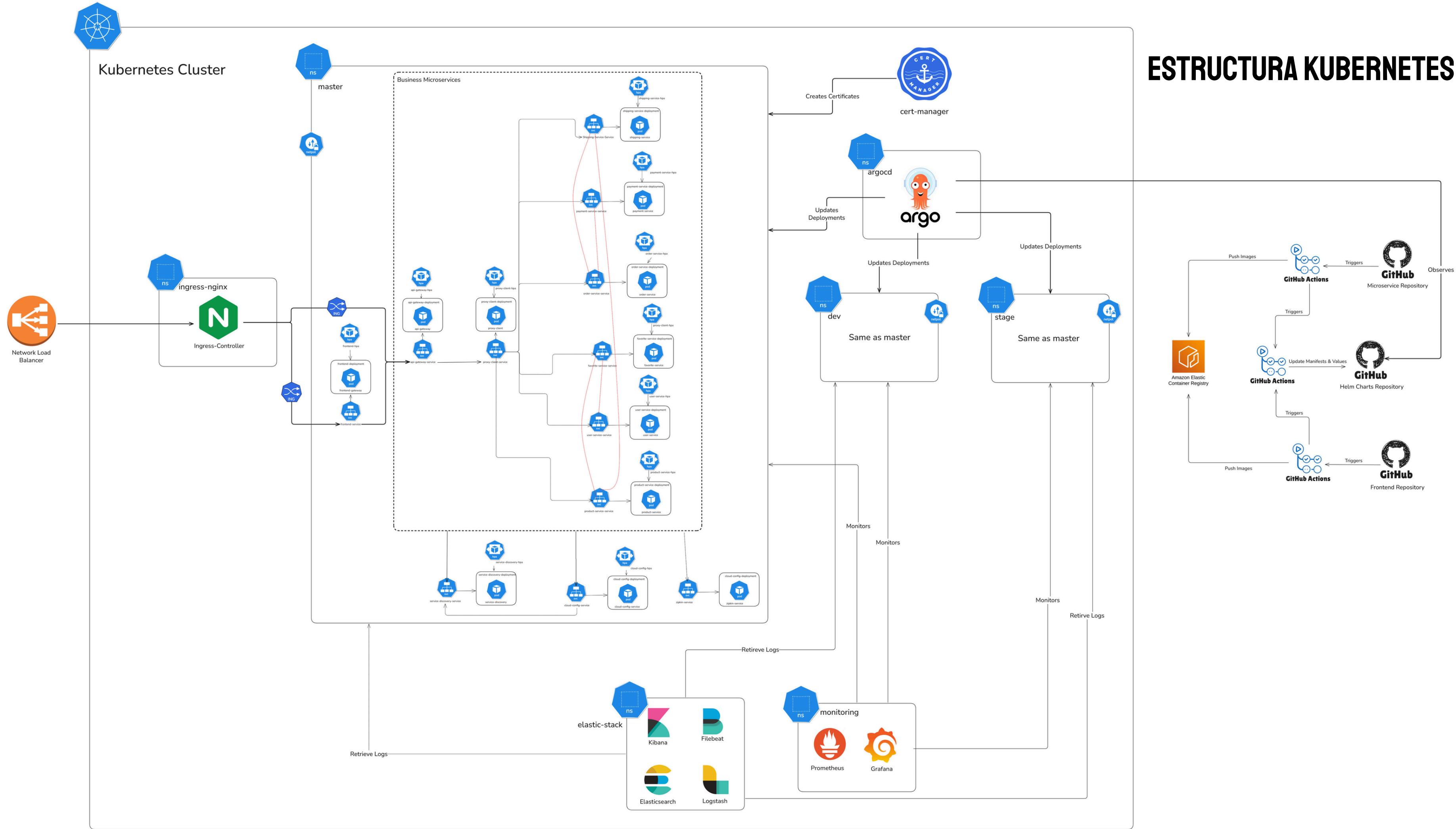
- ✓ master default
- dev
- feature/hu-1/change-text
- feature/hu-2/scan-images
- feature/hu-3/add-pipeline
- stage

# **DISEÑO E IMPLEMENTACIÓN**

# ARQUITECTURA EN AWS



# ESTRUCTURA KUBERNETES



# PATRONES DE CLOUD

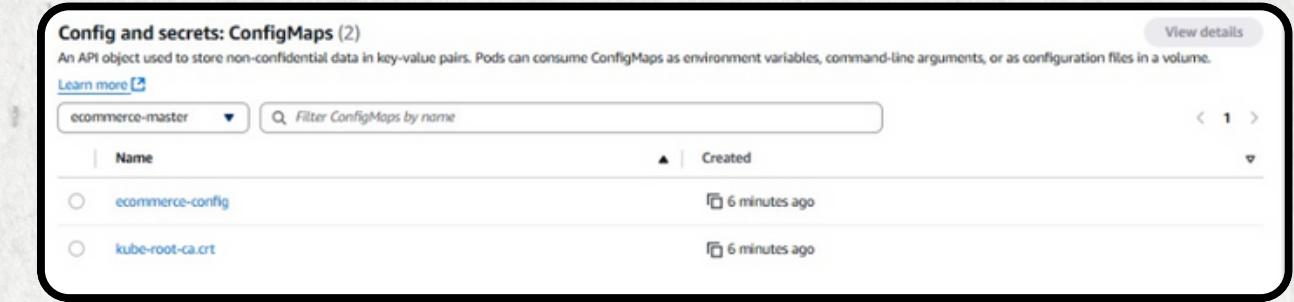
## HEALTH ENDPOINT MONITORING

Uso de actuator endpoints de Spring Boot.

```
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
```

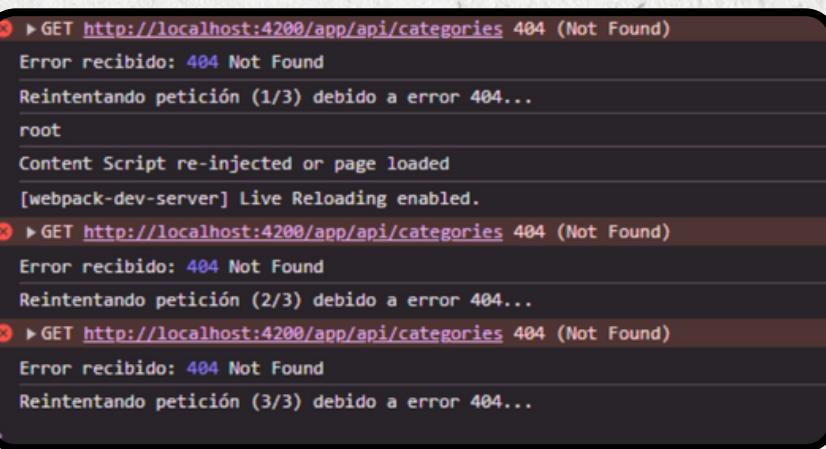
## EXTERNAL CONFIGURATION STORE

Uso de secrets y configmaps como configuración externa de spring boot



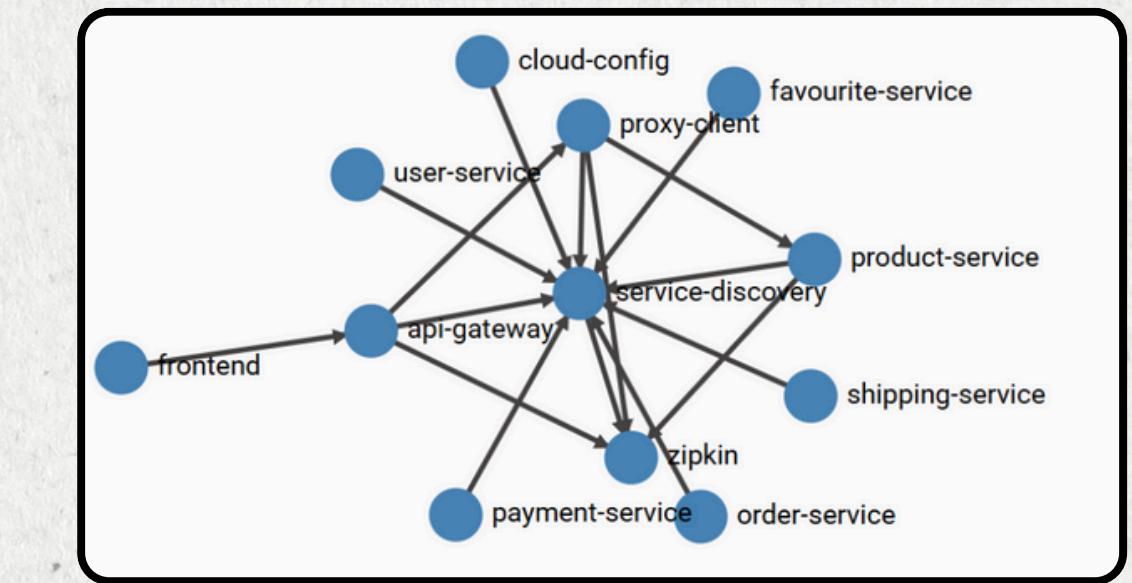
## RETRY

Reintento de solicitudes desde el frontend.



## SIDECAR

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



# **SEGURIDAD IMPLEMENTADA**

## Policy: NetworkPolicies (23)

Network Policy is a specification of how groups of Pods are allowed to

ecommerce-master



Filter NetworkPolicies by name



# NETWORKPOLICIES

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

Name

allow-all-egress

allow-api-from-ingress

allow-cloud-ingress

allow-favourite-and-product

allow-favourite-and-user

allow-favourite-from-proxy

allow-order-and-payment

allow-order-and-shipping

allow-order-from-proxy

allow-payment-and-order



# 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.

## Policy: NetworkPolicies (23)

Network Policy is a specification of how groups of Pods are allowed to

ecommerce-master



Filter NetworkPolicies by name

	Name
<input type="radio"/>	<a href="#">allow-all-egress</a>
<input type="radio"/>	<a href="#">allow-api-from-ingress</a>
<input type="radio"/>	<a href="#">allow-cloud-ingress</a>
<input type="radio"/>	<a href="#">allow-favourite-and-product</a>
<input type="radio"/>	<a href="#">allow-favourite-and-user</a>
<input type="radio"/>	<a href="#">allow-favourite-from-proxy</a>
<input type="radio"/>	<a href="#">allow-order-and-payment</a>
<input type="radio"/>	<a href="#">allow-order-and-shipping</a>
<input type="radio"/>	<a href="#">allow-order-from-proxy</a>
<input type="radio"/>	<a href="#">allow-payment-and-order</a>

## Policy: NetworkPolicies (23)

Network Policy is a specification of how groups of Pods are allowed to

ecommerce-master



Filter NetworkPolicies by name

Name

allow-all-egress

allow-api-from-ingress

allow-cloud-ingress

allow-favourite-and-product

allow-favourite-and-user

allow-favourite-from-proxy

allow-order-and-payment

allow-order-and-shipping

allow-order-from-proxy

allow-payment-and-order

# PERMISOS ESPECÍFICOS ENTRE MICROSERVICIOS

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

Ejemplos clave:

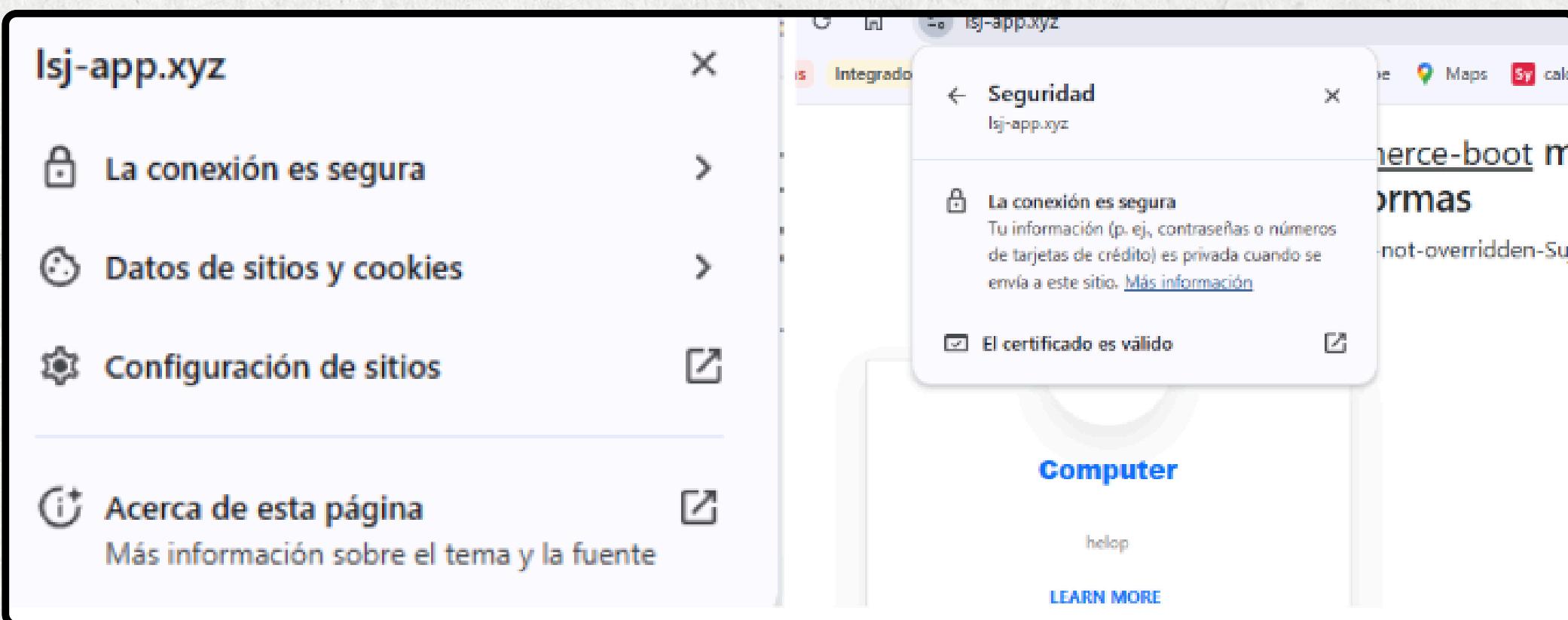
- api-gateway: solo accesible desde api-gateway-ingress (puerto 8080).
- proxy-client: accesible desde api-gateway (8900).
- favourite-service y product-service: comunicación mutua + acceso desde proxy-client.

# 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.

```
domains: [REDACTED]
  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.

**Workloads: CronJobs (3)**

Cron Job manages a Job that runs on a periodic schedule. [Learn more](#)

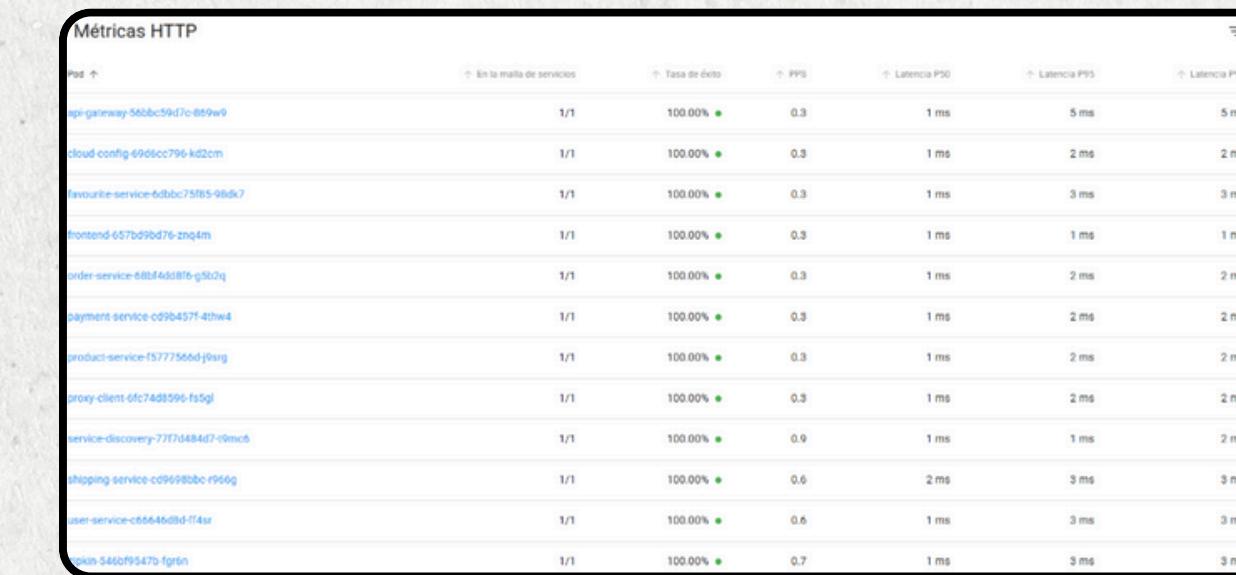
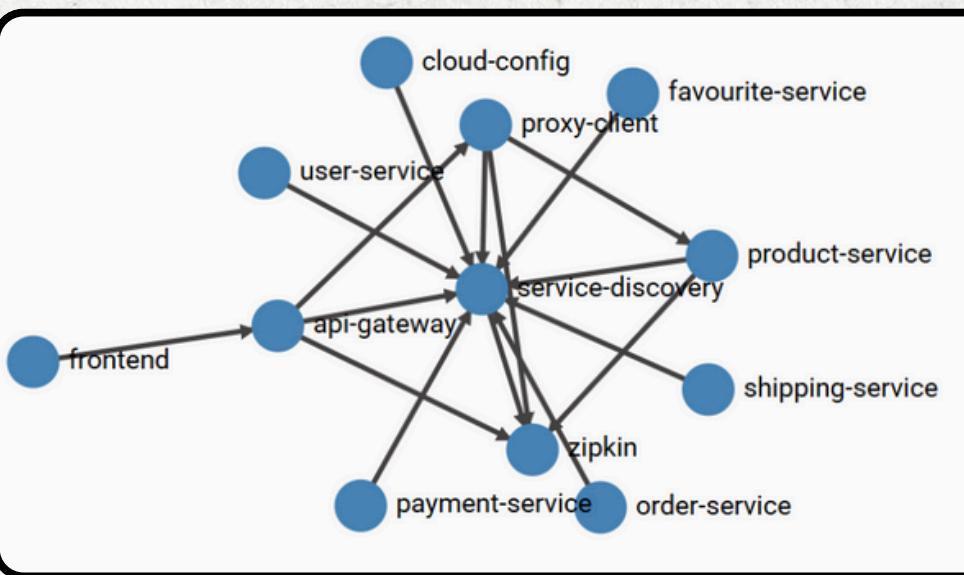
All Namespaces ▾  Filter CronJobs by name

Name	Created
frontend-secret-rotator-cronjob	4 minutes ago
frontend-secret-rotator-cronjob	4 minutes ago
frontend-secret-rotator-cronjob	4 minutes ago

# 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 injectar 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.



espacio

	Pods en la malla de servicios	Estado de la malla de servicios
argocd	0/7	
cert-manager	0/3	
default	0/0	
e-commerce-dev	0/12	
e-commerce-master	12/12	<div style="background-color: #28a745; color: white; padding: 2px;">Cambiado</div>
e-commerce-stage	0/12	
elastic-stack	0/6	
elastic-system	0/1	
ingress-engine	0/1	
kube-node-lease	0/0	
kube-public	0/0	
kube-system	0/31	
linkerd	3/3	<div style="background-color: #28a745; color: white; padding: 2px;">Cambiado</div>
linkerd-viz	5/5	<div style="background-color: #28a745; color: white; padding: 2px;">Cambiado</div>
monitoring	0/11	

sjbarraza

Azure for Students

	DST	SRC_NS	DST_NS	SECURED
pi-gateway	service-discovery	ecommerce-master	ecommerce-master	✓
loud-config	service-discovery	ecommerce-master	ecommerce-master	✓
favourite-service	service-discovery	ecommerce-master	ecommerce-master	✓
order-service	service-discovery	ecommerce-master	ecommerce-master	✓
payment-service	service-discovery	ecommerce-master	ecommerce-master	✓
product-service	service-discovery	ecommerce-master	ecommerce-master	✓
proxy-client	service-discovery	ecommerce-master	ecommerce-master	✓
service-discovery	zipkin	ecommerce-master	ecommerce-master	✓
shipping-service	service-discovery	ecommerce-master	ecommerce-master	✓
ser-service	service-discovery	ecommerce-master	ecommerce-master	✓
rometheus	api-gateway	linkerd-viz	ecommerce-master	✓
rometheus	cloud-config	linkerd-viz	ecommerce-master	✓
rometheus	favourite-service	linkerd-viz	ecommerce-master	✓
rometheus	frontend	linkerd-viz	ecommerce-master	✓
rometheus	order-service	linkerd-viz	ecommerce-master	✓
rometheus	payment-service	linkerd-viz	ecommerce-master	✓
rometheus	product-service	linkerd-viz	ecommerce-master	✓
rometheus	proxy-client	linkerd-viz	ecommerce-master	✓
rometheus	service-discovery	linkerd-viz	ecommerce-master	✓
rometheus	shipping-service	linkerd-viz	ecommerce-master	✓
rometheus	user-service	linkerd-viz	ecommerce-master	✓
rometheus	zipkin	linkerd-viz	ecommerce-master	✓

# **CALIDAD DE CODIGO**

# TESTING DE LOS MICROSERVICIOS

```
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.

```
@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"));  
}  
  
@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());  
}  
  
@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());  
}
```

**E ecommerce-frontend-web-app**

No tags      Last analysis 13 Jun 2025      914 Lines of Code

Main Branch Status  
Quality Gate Failed  
1 failed condition  
0.0% Security Hotspots Reviewed on New Code  
100% required

Main Branch Evolution since 14 hours ago  
31 Issues = 0 Issues    Coverage    Duplications

See Full Analysis

Latest Activity  
NEW ANALYSIS | Main Branch  
13 June at 01:30 Merge pull request #1 from microservices-project-k8s-jenkins/feat/change-text  
0 Fixed Issues    0 New Issues    0.0% Coverage    0.0% Duplications  
0 Lines of Code

FIRST ANALYSIS | feat/change-text  
13 June at 01:29 7caSelca feat: text changed  
0 Issues    0.0% Coverage    0.0% Duplications  
20 Lines of Code

Software Quality

Security: 0      Reliability: 4      Maintainability: 27

Severity: 0 Blocker    0 High    21 Medium    10 Low    0 Info

Clean Code Attribute: Consistency: 7    Intentionality: 24    Adaptability: 0    Responsibility: 0

Type:      Status:      Security Category:      Creation Date:

Dockerfile

- Merge this RUN instruction with the consecutive ones. (Consistency, Open, Not assigned, L3, 5min effort, 18 days ago, Code Smell, Minor)
- Sort these package names alphanumerically. (Consistency, Open, Not assigned, L3, 5min effort, 18 days ago, Code Smell, Minor)
- Merge this RUN instruction with the consecutive ones. (Consistency, Open, Not assigned, L9, 5min effort, 18 days ago, Code Smell, Minor)

src/app/app.component.html

- Add "lang" and/or "xml:lang" attributes to this "<html>" element (Intentionality, Reliability, Open, Not assigned, L2, 2 min effort, 3 years ago, Bug, Major)
- Add a <title> tag to this page. (Consistency, Reliability, Open, Not assigned, L4, 5min effort, 3 years ago, Bug, Major)

src/app/app.component.spec.ts

4 / 31 Issues

Dockerfile  
Merge this RUN instruction with the consecutive ones.  
2 locations

Sort these package names alphanumerically.  
Merge this RUN instruction with the consecutive ones.  
2 locations

src/app/app.component.html  
Add "lang" and/or "xml:lang" attributes to this "<html>" element  
<html> element should have a language attribute Web:55254  
Software qualities impacted: Reliability  
Open, Not assigned, Bug, Major

Where is the issue? Why is this an issue? Activity More info      Open in IDE

src/app/app.component.html

```

1 horri... <!DOCTYPE html>
2 horri... <html>
3 horri... <head>
4 horri...   <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.1.0/dist/css/bootstrap.min.css"
5 horri...   integrity="sha384-KyZEA9j0hqMpG8+I8fI8F4d6Z9LzO3wD1QFqYrxEZ8vPZl9jy6OrQVrjIEaf/nJGzIxFds4x0xIM+B07jR8" crossorigin="anonymous" />
6 horri...
7 horri...
8 horri...
9 horri...
10 horri...
11 horri...
12 horri...
13 horri...
14 horri...
15 horri...
16 horri...
17 horri...
18 horri...
19 horri...
20 horri...
21 horri...
22 horri...
23 horri...
24 horri...

```

See all issues in this file

microservices-project-k8s-jenkins > ecommerce-frontend-web-app > master

Summary    Issues    Security Hotspots    Measures    Code    Activity

Search for files...

	Lines of Code	Security	Reliability	Maintainability	Security Hotspots	Coverage	Duplications
ecommerce-frontend-web-app	856	0	4	24	1	—	0.0%
src	—	0	0	0	0	—	0.0%
angular.json	—	0	0	0	0	—	0.0%
azure-pipelines.yml	—	0	0	0	0	—	0.0%
compose.yml	—	0	0	0	0	—	0.0%
Dockerfile	18	0	0	3	4	—	0.0%
karma.conf.js	37	0	0	0	0	—	0.0%
package-lock.json	—	0	0	0	0	—	0.0%
package.json	—	0	0	0	0	—	0.0%
tsconfig.app.json	—	0	0	0	0	—	0.0%
tsconfig.json	—	0	0	0	0	—	0.0%
tsconfig.spec.json	—	0	0	0	0	—	0.0%

11 of 11 shown

# ESCANEO DE VULNERABILIDADES



Screenshot of a GitHub Actions workflow page showing a recent run for 'CI - Build and Scan Frontend Image on PR Merge and Tr...'. The run was manually triggered by 'luisapino' and completed 39 minutes ago. It shows 9 workflow runs.

```
2025-06-11T20:50:54Z  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 product-service. The build will be marked as failed.  
  
-----  
Processing service: shipping-service  
  
1  2025-06-11T20:50:12Z  WARN  Using severities from other vendors for some vulnerabilities. Read https://trivy.dev/v0.63/docs/scanner/vulnerability#severity-selection for details.  
22  Error: Trivy found critical vulnerabilities in proxy-client. The build will be marked as failed.  
23  -----  
24  Processing service: order-service  
  
267  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.  
268  Error: Trivy found critical vulnerabilities in api-gateway. The build will be marked as failed.  
269  -----  
270  Processing service: proxy-client  
271  -----
```

Code scanning

Trivy is reporting errors. Check the [status page](#) for help.

Tools 1 + Add tool

is:open branch:master

2,786 Open 0 Closed Language Tool Rule Severity Sort

Template injection in thymeleaf-spring5 Critical Library master  
#2782 opened 1 hour ago • Detected by Trivy in home/.../lib/thymeleaf-spring5-3.0.12...:1

Template injection in thymeleaf-spring5 Critical Library master  
#2781 opened 1 hour ago • Detected by Trivy in home/.../lib/thymeleaf-spring5-3.0.12...:1

spring-framework: RCE via Data Binding on JDK 9+ Critical Library master  
#2776 opened 1 hour ago • Detected by Trivy in home/.../lib/spring-webmvc-5.3.13.jar :1

spring-framework: RCE via Data Binding on JDK 9+ Critical Library master  
#2775 opened 1 hour ago • Detected by Trivy in home/.../lib/spring-webmvc-5.3.13.jar :1

spring: HttpInvokerServiceExporter readRemoteInvocation method untrusted java deserialization Critical Library master  
#2768 opened 1 hour ago • Detected by Trivy in home/.../lib/spring-web-5.3.13.jar :1

spring: HttpInvokerServiceExporter readRemoteInvocation method untrusted java deserialization Critical Library master  
#2767 opened 1 hour ago • Detected by Trivy in home/.../lib/spring-web-5.3.13.jar :1

spring-framework: RCE via Data Binding on JDK 9+ Critical Library master  
#2760 opened 1 hour ago • Detected by Trivy in home/.../lib/spring-beans-5.3.13.jar :1

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	<a href="#">CVE-2021-43466</a>

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.

**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

**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

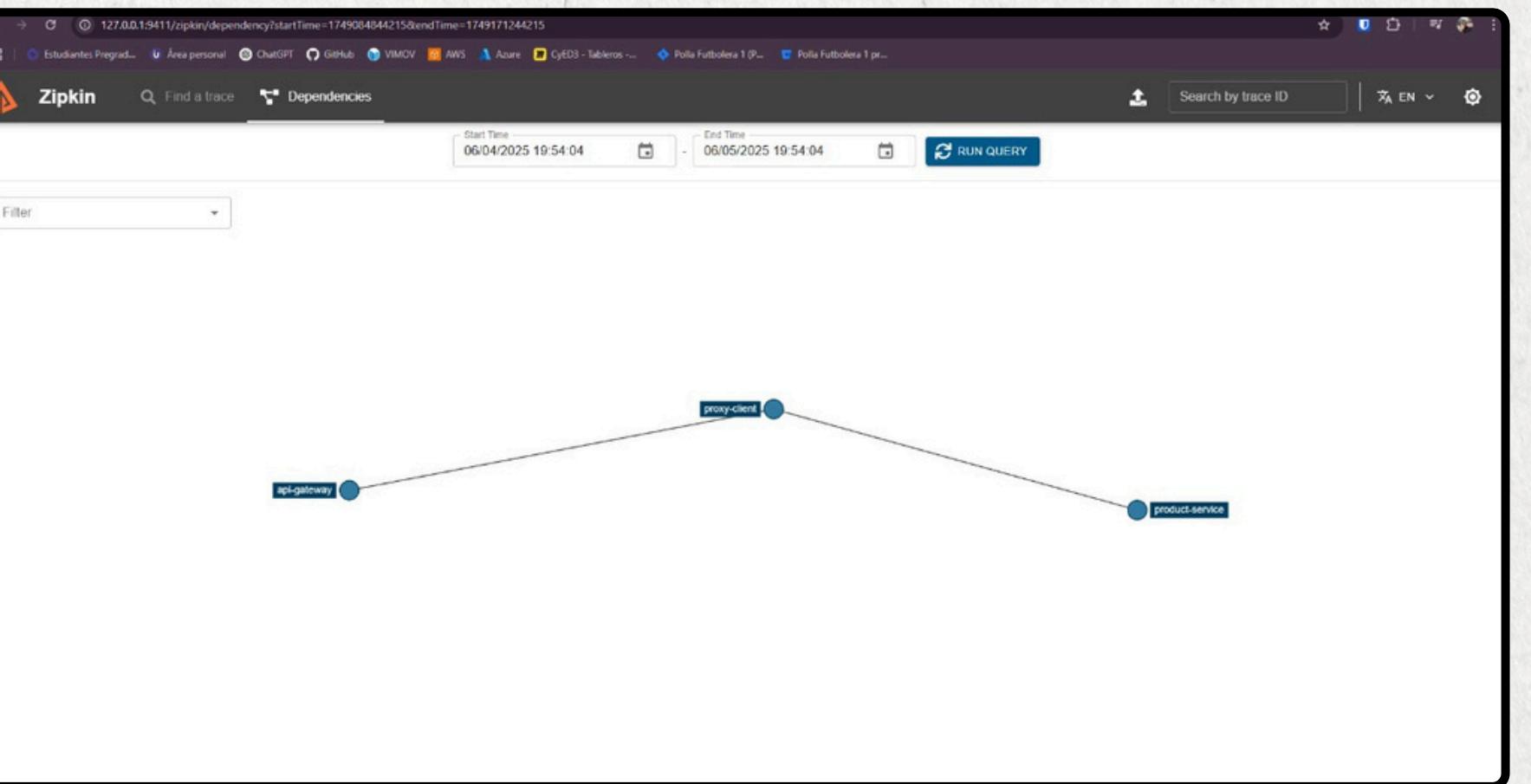
Ubuntu

Additional information

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

# **OBSERVABILIDAD Y MONITOREO**

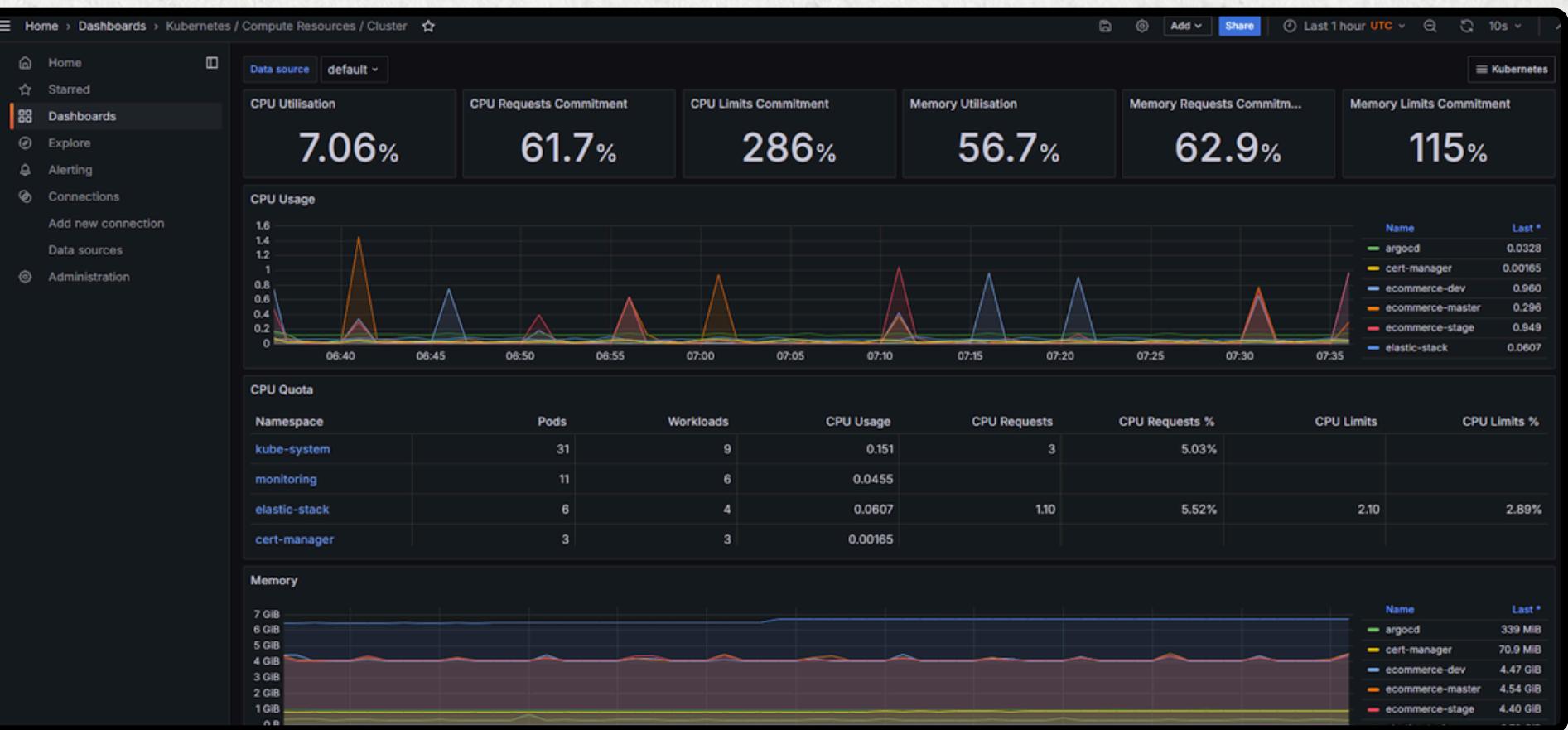
# ELASTIC Y ZIPKIN



The screenshot shows the Elastic Stack Discover interface. At the top, there is a search bar labeled "Find apps, content, and more." and a "Discover" tab. Below this is a "Help us improve the Elastic Stack" message and a "Usage collection is enabled" note. The main area has tabs for "Data view" and "All logs". A search bar with "Filter your data using KQL syntax" is present. The left sidebar lists "Available fields" such as @timestamp, @version, agent.ephemeral\_id, agent.id, agent.name, agent.type, agent.version, aws.tags.eks:cluster-name, cloud.account.id, cloud.availability\_zone, cloud.image.id, cloud.instance.id, cloud.machine.type, cloud.provider, cloud.region, cloud.service.name, data\_stream.dataset, data\_stream.namespace, data\_stream.type, ecs.version, host.architecture, host.containerized, and host.hostname. The right side shows a histogram for @timestamp from June 11, 2025 at 20:23:36 to Jun 11, 2025 at 20:23:38.552, with a count of 61. Below the histogram is a table titled "Documents (100)" showing log entries. The first entry is:

```
@timestamp Jun 11, 2025 @ 20:23:38.552 #version 1 agent.ephemeral_id 221195d7-f5e2-4293-8d04-ddb8d9bb4093 agent.id 905256d6-7177-4363-98b4-faf89c5e9aa4 agent.name eck-stack-with-logstash-eck-beats-beat-filebeat-7467787657t58rw agent.type filebeat agent.version 9.0.0 aws.tags.eks:cluster-name projectoffinalsjeks cloud.account.id 468720136698 cloud.availability_zone us-east-1b cloud.image.id ami-0be152275305629af cloud.instance.id i-0f182f7d16d774c1a cloud.machine.type m6i.larg e.cloud.provider aws cloud.region us-east-1 cloud.service.name EC2 data_stream.dataset generic_data_stream.namespace default data_stream.type logs ecs.version 8.0.0 event.original 86.1.76.62 - - [04/Jan/2015:05:38:37 +0000] "GET /style2.css HTTP/1.1"
```

# PROMETHEUS Y GRAFANA



# ALERTAS

Prometheus Alerts Graph Status Help

Inactive (141)  Pending (0)  Firing (4)

Filter by name or labels  Show annotations

/etc/prometheus/rules/prometheus-kube-prometheus-stack-prometheus-rulefiles-0/monitoring-custom-alert-rules-3c30f80a-8768-4576-9cb8-a64034f44b8d.yaml > example.rules pending (1)

HighCPUUsage (1 active)

```
name: HighCPUUsage
expr: sum by (pod) (rate(container_cpu_usage_seconds_total{image!=""}[1m])) > 0.8
for: 1m
labels:
  severity: warning
annotations:
  description: Pod {{ $labels.pod }} is using more than 80% CPU.
  summary: High CPU usage detected on pod
```

Labels	State	Active Since	Value
alertname=HighCPUUsage pod=stress-cpu severity=warning	PENDING	2025-06-12T21:00:36.392631288Z	0.9936753872296604

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

Inactive (141)  Pending (0)  Firing (5)

Filter by name or labels  Show annotations

/etc/prometheus/rules/prometheus-kube-prometheus-stack-prometheus-rulefiles-0/monitoring-custom-alert-rules-3c30f80a-8768-4576-9cb8-a64034f44b8d.yaml > example.rules firing (1)

HighCPUUsage (1 active)

```
name: HighCPUUsage
expr: sum by (pod) (rate(container_cpu_usage_seconds_total{image!=""}[1m])) > 0.8
for: 1m
labels:
  severity: warning
annotations:
  description: Pod {{ $labels.pod }} is using more than 80% CPU.
  summary: High CPU usage detected on pod
```

Labels	State	Active Since	Value
alertname=HighCPUUsage pod=stress-cpu severity=warning	FIRING	2025-06-12T21:00:36.392631288Z	0.995965179317808

# **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-mlrrww       1/1     Terminating   0          6m12s
Frontend-756c794fb-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 📂 ~ 2s ▲ 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-756c794fb-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-vii77  1/1     Running   0          8s
```

```
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
```



200 usuarios como  
peak

Aumentan de 5 en 5

# PRUEBAS DE RENDIMIENTO

**Start new load test**

**Number of users (peak concurrency)**

**Spawn rate (users started/second)**

**Host (e.g. http://www.example.com)**

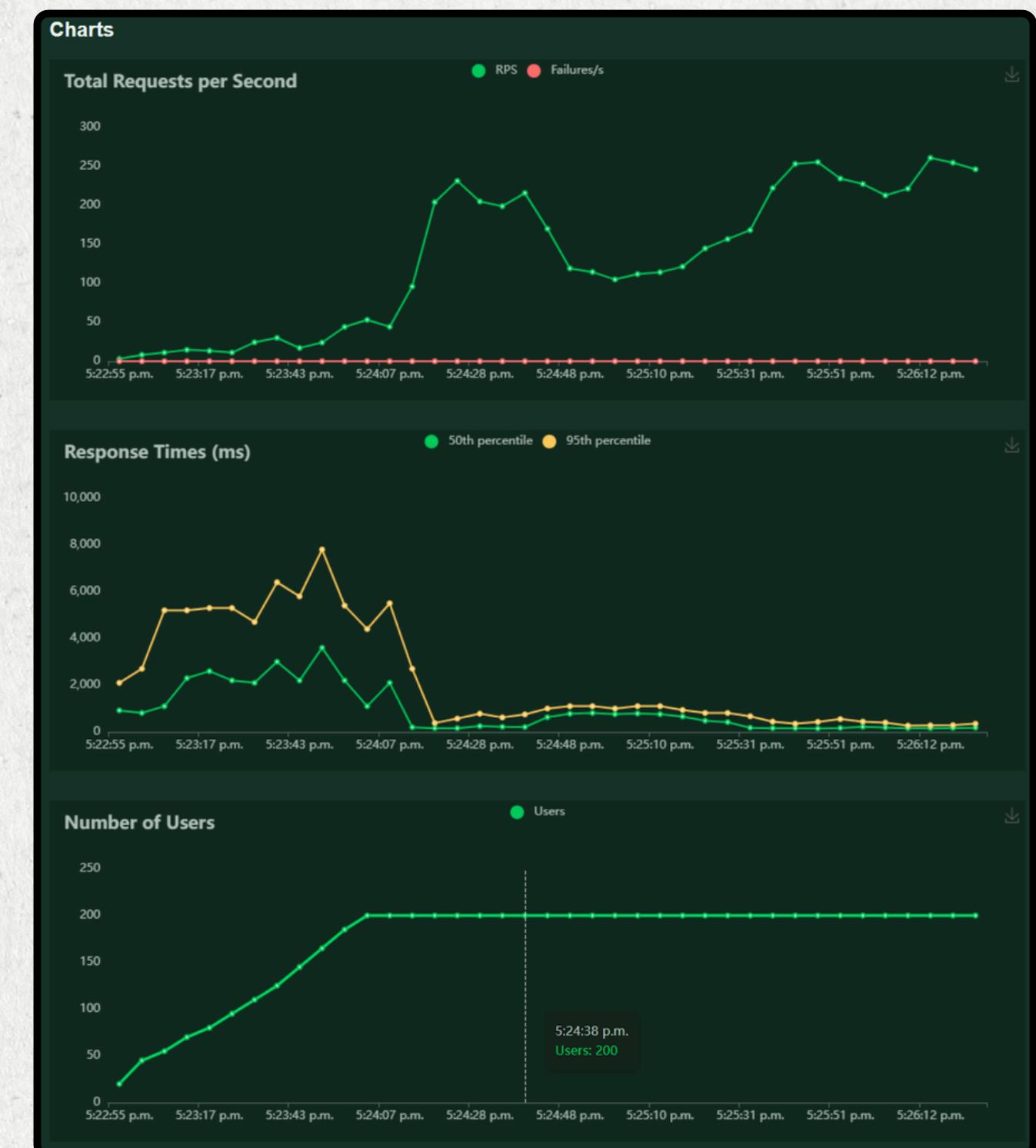
**Advanced options**

**Start swarming**

Solicitudes a  
la API cifrada con  
TLS

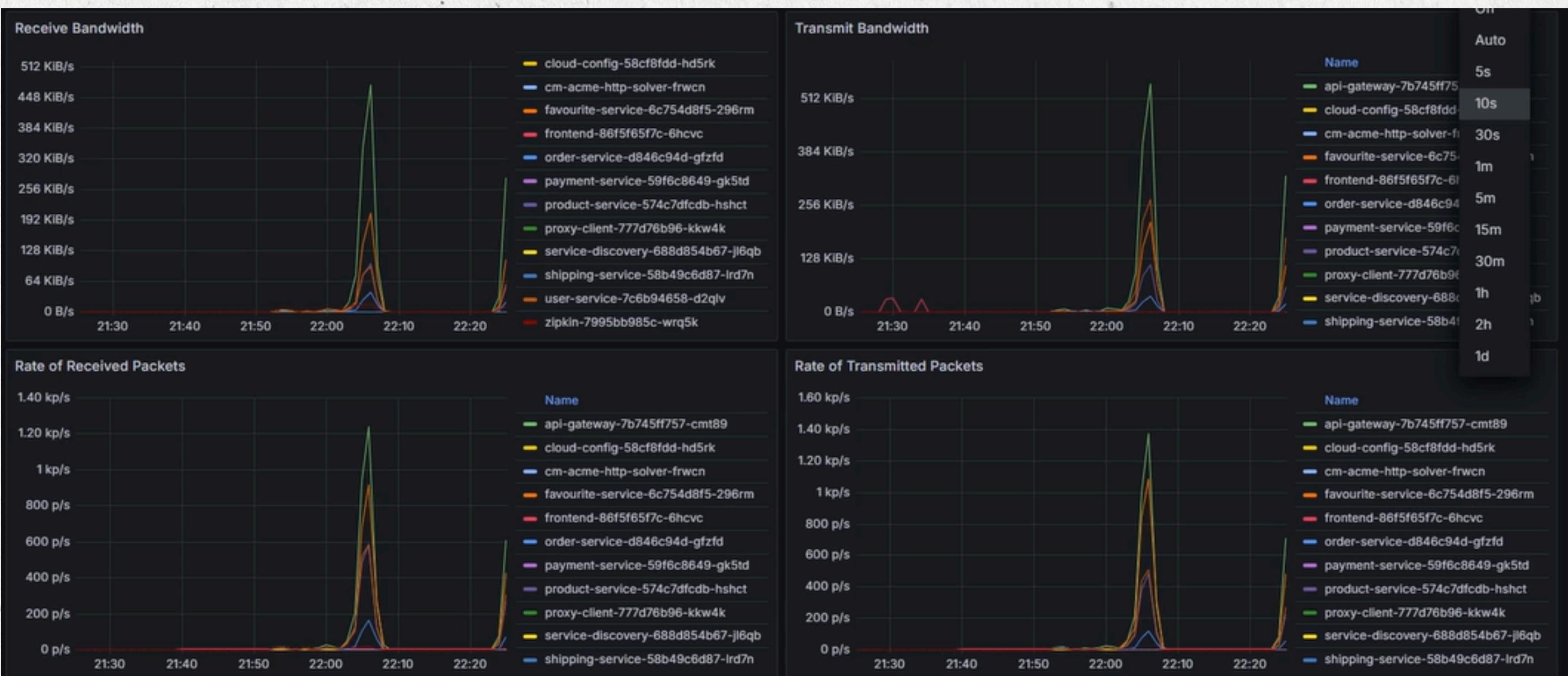
# PRUEBAS DE RENDIMIENTO

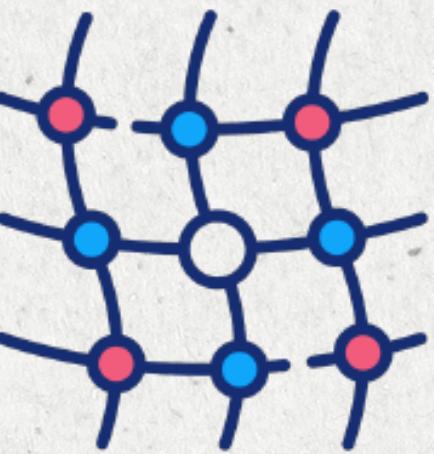
## Reporte en Locust



# PRUEBAS DE RENDIMIENTO

## Visualización en Grafana





# Implementación

```

sjbarraza 🖥 ~ AWS Azure for Students
  kubectl get schedule -n ecommerce-master
  AGE
  delay-product-service-schedule 3m54s

sjbarraza 🖥 ~ AWS Azure for Students
  kubectl get networkchaos -n ecommerce-master
  AGE
  delay-product-service-schedule-6zxwh ACTION DURATION
  delay 1m

```

# CHAOS MESH

## Antes

Métricas HTTP							
Deployment ↗	En la malla de servicios	Tasa de éxito	PPS	Latencia P50	Latencia P95	Latencia P99	Latencia P999
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	
rickin	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	Latencia P999
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	
rickin	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

**iGRACIAS!**