Intel DevOps Assessment

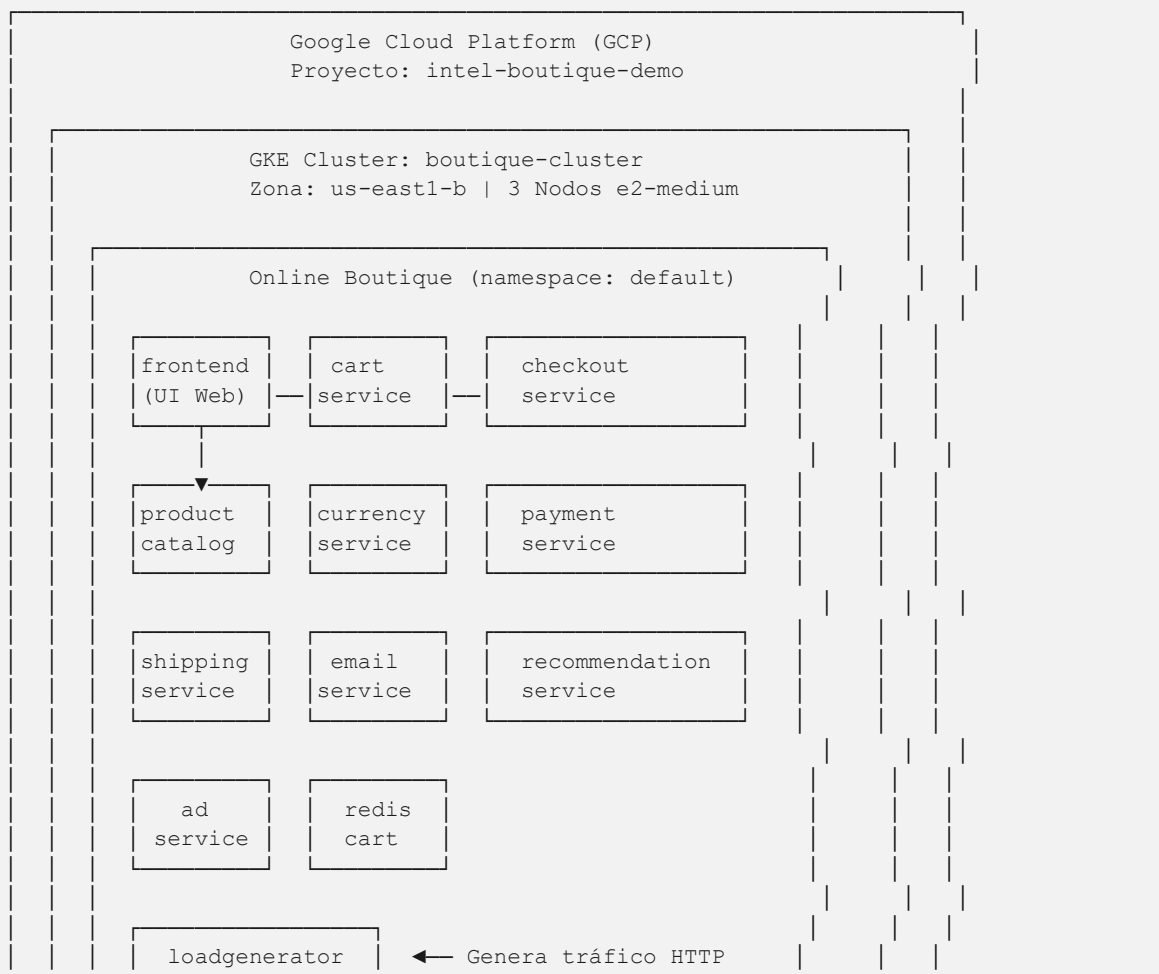
Documento de Arquitectura

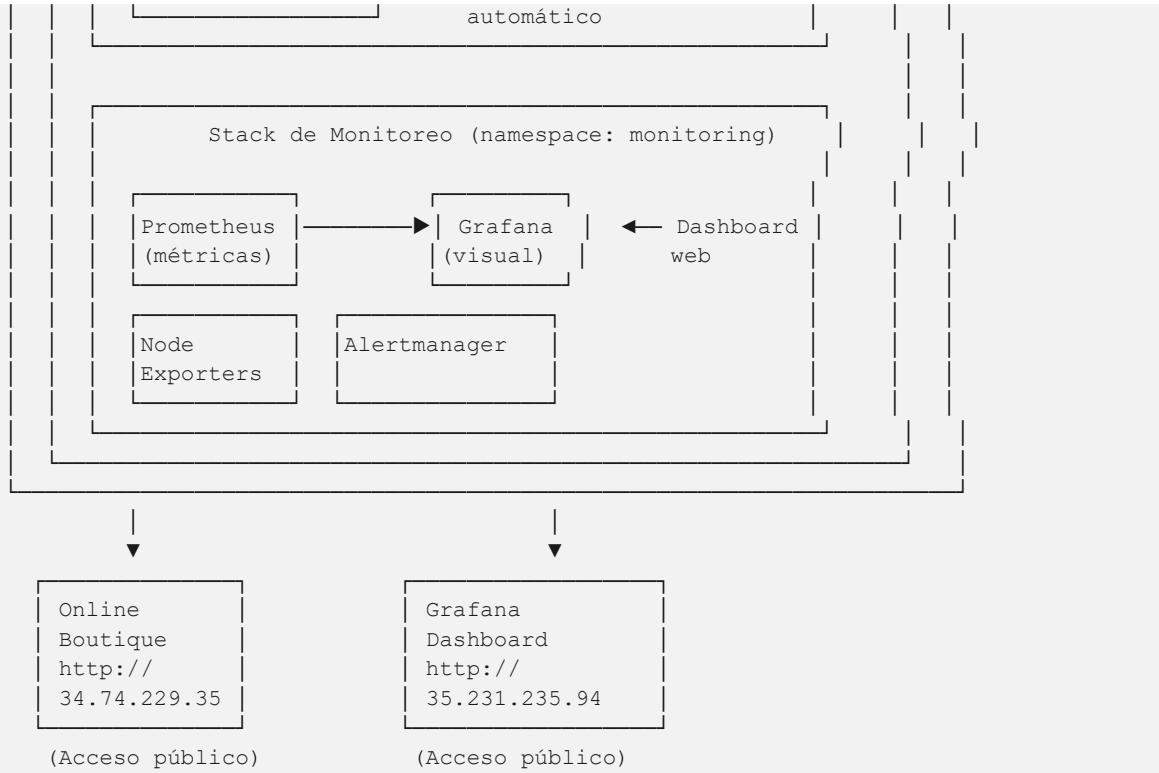
1. Resumen Ejecutivo

Este documento describe la arquitectura de la infraestructura desplegada para el assessment de Intel. El entorno consiste en un clúster de Google Kubernetes Engine (GKE) con la aplicación Online Boutique (12 microservicios) desplegada y un stack de monitoreo compuesto por Prometheus y Grafana. Todo el proceso de despliegue está automatizado mediante scripts que permiten reproducir el entorno completo desde cero con un solo comando.

2. Diagrama de Arquitectura

El diagrama a continuación muestra la estructura completa del entorno, desde la infraestructura de GCP hasta los microservicios y el stack de monitoreo.





3. Descripción de Componentes

3.1 Google Kubernetes Engine (GKE)

GKE es el servicio de Kubernetes gestionado de Google Cloud. Se eligió GKE porque proporciona un entorno de producción realista dentro del free tier de GCP ($300 de créditos). El clúster tiene 3 nodos de tipo e2-medium con 80GB de disco SSD cada uno, lo cual permite ejecutar los 12 microservicios con los exportadores de métricas de Prometheus sin exceder las cuotas del proyecto.

3.2 Online Boutique

Online Boutique es la aplicación de demostración de microservicios desarrollada por Google Cloud Platform. Consta de 12 servicios que simulan una tienda en línea completa. Cada servicio se despliega como un pod independiente en Kubernetes y se comunican entre sí usando gRPC y HTTP. El loadgenerator genera tráfico automático hacia el frontend, lo cual produce métricas constantes que Prometheus recopila.

3.3 Prometheus

Prometheus es el sistema de monitoreo que recopila métricas de todos los pods del clúster. Se instaló mediante el Helm chart kube-prometheus-stack, que instala automáticamente Prometheus, los node exporters (que recopilan métricas del sistema operativo de cada nodo) y Alertmanager. Prometheus hace scraping periódico de los

endpoints de métricas de cada pod y las almacena en su base de datos de series temporales.

3.4 Grafana

Grafana es la herramienta de visualización que se conecta a Prometheus como fuente de datos. Se creó un dashboard personalizado llamado "Online Boutique - Métricas" con dos paneles: uno para CPU por pod y otro para memoria por pod. Grafana está expuesta con una IP pública mediante un servicio de tipo LoadBalancer para permitir acceso externo.

3.5 Scripts de Automatización

El proceso de despliegue está automatizado mediante dos scripts: deploy.sh para entornos Linux/Mac y deploy.bat (junto con adjust-resources.ps1) para Windows. Ambos scripts realizan exactamente los mismos pasos: verifican prerrequisitos, crean el clúster GKE, despliegan la Online Boutique con ajustes de recursos, instalan el stack de monitoreo, y muestran la información de acceso al final. Los scripts son idempotentes, lo que significa que si un recurso ya existe, lo detectan y se saltan ese paso sin dar error.

4. Flujo de Datos

El flujo de datos en el entorno se desarrolla de la siguiente manera:

Paso	Descripción
1. Generación de tráfico	El loadgenerator envía peticiones HTTP continuamente al frontend de la Online Boutique. Estas peticiones recorren los microservicios internos (catálogo, carrito, checkout, etc.).
2. Procesamiento de peticiones	Cada microservicio procesa su parte de la petición y se comunica con los otros servicios que necesita. Por ejemplo, checkout llama a payment, shipping y email para completar una orden.
3. Exposición de métricas	Cada pod expone automáticamente métricas de rendimiento (CPU, memoria, latencia) en un endpoint HTTP. Estas métricas son generadas por las librerías de instrumentación de cada servicio.
4. Recopilación por Prometheus	Prometheus hace scraping periódico de los endpoints de métricas de todos los pods y los node exporters. Las métricas se almacenan en la base de datos de series temporales de Prometheus.
5. Visualización en Grafana	Grafana consulta a Prometheus usando PromQL (lenguaje de consulta de

	Prometheus) para obtener las métricas y las presenta visualmente en el dashboard. Los paneles se actualizan automáticamente.

5. Configuración del Entorno

Parámetro	Valor
Proyecto GCP	intel-boutique-demo
Zona	us-east1-b
Nombre del clúster	boutique-cluster
Cantidad de nodos	3
Tipo de máquina	e2-medium (2 vCPU, 4GB RAM)
Disco por nodo	80GB SSD
Microservicios desplegados	12
URL Online Boutique	http://34.74.229.35
URL Grafana	http://35.231.235.94
Namespace aplicación	default
Namespace monitoreo	monitoring

6. Métricas Configuradas en Grafana

El dashboard "Online Boutique - Métricas" contiene dos paneles que muestran el rendimiento de los microservicios en tiempo real:

Panel	Query PromQL	Descripción
CPU por pod	sum(rate(container_cpu_usage_seconds_total{namespace="default"}[5m])) by (pod)	Muestra el uso de CPU de cada pod en porcentaje. El rate calcula la tasa de cambio en los últimos 5 minutos y el sum agrupa por pod.
Memoria por pod	sum(container_memory_working_set_bytes{namespace="default"}) by (pod)	Muestra el uso de memoria de cada pod en bytes. El working_set representa la memoria

		que el contenedor está activamente usando.

7. Ajustes de Recursos

Para que los microservicios funcionen dentro de las cuotas del proyecto GCP en el free tier, se redujeron los recursos solicitados por cada servicio. En Kubernetes, los "requests" son el mínimo que garantiza el scheduler y los "limits" son el máximo permitido. Es obligatorio que requests sea menor o igual que limits.

Recurso	Valor original	Valor ajustado	Motivo
CPU requests (servicios)	200-300m	50m	Reducir la cantidad total de CPU reservada para que los 12 servicios caben en los 3 nodos.
CPU limits (servicios)	300-500m	100m	Debe ser mayor o igual que requests. Se mantuvo en 100m para permitir picos cortos.
CPU requests (loadgenerator)	300m	50m	El loadgenerator solo genera tráfico HTTP, no necesita mucho CPU.
Memoria (loadgenerator)	256Mi	64Mi	El loadgenerator no necesita mucha memoria para generar peticiones HTTP.

8. Consideraciones de Escalabilidad

El entorno actual está optimizado para el free tier de GCP, pero la arquitectura permite escalar fácilmente hacia un entorno de producción. Se puede aumentar el número de nodos del clúster, cambiar el tipo de máquina a uno más potente, y restaurar los recursos originales de los microservicios. Kubernetes maneja automáticamente la distribución de pods entre nodos, por lo que no se necesitan cambios en la configuración de la aplicación. Además, es posible agregar un Horizontal Pod Autoscaler (HPA) que escale automáticamente el número de réplicas de cada servicio según la demanda.