UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
SISTEMAS OPERATIVOS 1 SECCIÓN A
ING. EDGAR RENE ORNELIS HOIL
AUX. DIEGO ALEJANDRO JUAREZ BRAN
VACACIONES JUNIO 2025

PROYECTO FASE 2

Monitoreo Cloud de VMs

Objetivos

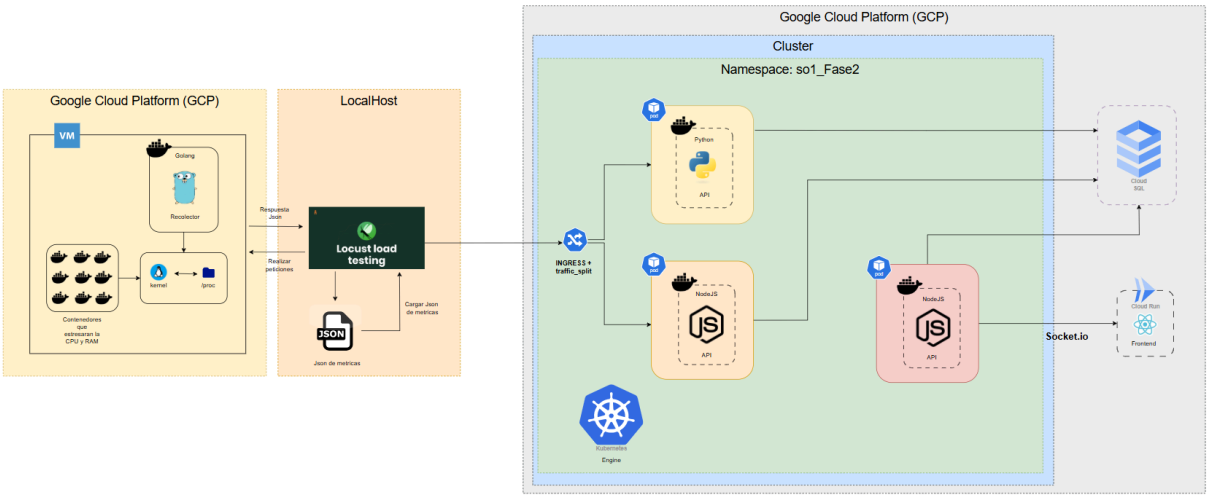
- **Desplegar un clúster de Kubernetes** que permita gestionar y escalar los servicios involucrados en la recolección y análisis de métricas del sistema.
- **Explorar el enfoque Serverless** utilizando **Cloud Run** para desplegar aplicaciones ligeras de monitoreo o visualización de datos.
- **Implementar una base de datos en la nube** mediante **Cloud SQL** para almacenar de forma segura y escalable las métricas recolectadas.
  Configurar balanceadores de carga que permitan distribuir eficientemente el tráfico generado por las pruebas de carga hacia la VM objetivo.
- **Diseñar una aplicación web** que presente de forma visual e interactiva las estadísticas del rendimiento, facilitando la toma de decisiones.

Introducción

El objetivo de este proyecto es evaluar la capacidad de una máquina virtual (VM) para soportar cargas de trabajo específicas, mediante la generación de tráfico controlado que permita medir su rendimiento en tiempo real. Para ello, se implementará un sistema de monitoreo que recolecta métricas clave como el uso de CPU, consumo de memoria RAM y procesos activos del kernel. Las pruebas de carga y rendimiento serán realizadas utilizando herramientas como **Locust**, y la infraestructura se desplegará sobre **Kubernetes**, haciendo uso de servicios en la nube como **Cloud SQL** y **Cloud Run** para asegurar escalabilidad y disponibilidad.

Como parte del proyecto, se desarrollará una aplicación web que visualice las métricas recolectadas mediante gráficos y reportes, permitiendo un análisis detallado del comportamiento de la VM bajo distintas condiciones. Esta plataforma facilitará la toma de decisiones sobre la viabilidad del uso de dicha VM en entornos productivos, garantizando que cumpla con los requerimientos de rendimiento establecidos.

Arquitectura



Módulos del Kernel

En la fase1 los estudiantes implementaron 2 modelos uno para lectura de cpu y otra para lectura de ram, para la fase2 los estudiantes deberán implementar un módulo que sea capaz de mostrar información sobre los procesos del sistema, debe tener la siguiente estructura:

```
{
        "procesos_corriendo": 123,
        "total_procesos": 233,
        "procesos_durmiendo": 65,
        "procesos_zombie": 65,
        "procesos_parados": 65,
}
```

El nombre del módulo será: procesos_<carnet>

Generador de Trafico

Esta parte consiste en la creación de un generador de tráfico de entrada y salida utilizando Locust y Python como lenguaje de programación. Al inicio Locust generará peticiones a una VM para generar la data que será guardada en formato JSON el cual será enviado a un Ingress y dividido con un traffic split.

Configuración de Locust

Se debe de generar un scripts en Locust para simular el comportamiento de usuarios realizando peticiones a una VM. El JSON de salida tendrá un formato similar al siguiente:

```
{
    "total_ram": 2072,
    "ram_libre": 1110552576,
    "uso_ram": 442,
    "porcentaje_ram": 22,
    "porcentaje_cpu_uso": 22,
    "porcentaje_cpu_libre": 88,
    "procesos_corriendo": 123,
    "total_procesos": 233,
    "procesos_durmiendo": 65,
```

```
    "procesos_zombie": 65,
    "procesos_parados": 65,
    "hora": "2025-06-17 02:21:54"
}
```


Después de haber generado tráfico por al menos 3 min a la VM, con un número de usuario de 300, realizando peticiones entre 1, 2 seg y agregando nuevos usuarios cada seg. de esta forma se genera un Json que debe tener aprox. 2000 registros.

Luego de generar el Json con los datos anteriores, el usuario enviará la data usando locust al traffic split, con un número de usuario de 150, realizando peticiones entre 1, 4 seg y agregando nuevos usuarios cada seg.

Kubernetes

Se encargará de la orquestación de los contenedores de las diferentes partes de la aplicación. Se debe de crear un Namespace llamado: **so1_fase2,** y trabajar dentro de él para crear los **PODS** y los Servicios.

Ingress

El Ingress se encargará de gestionar el acceso externo a los servicios dentro del clúster. Actúa como una puerta de entrada que posibilita la exposición de servicios HTTP y HTTPS fuera del clúster. Por otro lado, el Traffic Split se encargará de distribuir de manera equitativa el tráfico de datos. En otras palabras, la configuración del tráfico será del 50% para la ruta 1 y del 50% para la ruta 2.

Rutas

En esta sección se centra en desarrollar un método altamente eficiente para registrar datos en MySQL a través de comunicación RPC (Remote Procedure Call). El propósito es evaluar y contrastar el rendimiento con respecto a otro enfoque existente. La implementación de la primera estrategia implica recibir los datos de entrada mediante una API en Python y posteriormente, almacenar los datos en la base de datos de MYSQL que está en el servicio de CLOUD SQL. El Flujo de la segunda ruta será la misma que la primera exceptuando que la API será realizada en NodeJS y también almacena datos en CLOUD SQL.

Ruta 1

La Primera Ruta será desarrollada en Python, estructura:

1. Escribir en la base de datos de MySQL.

Ruta 2

La Segunda Ruta será desarrollada en NodeJS, estructura:

a. Escribir en la base de datos de MySQL.

SOCKET IO

Se debe implementar un canal de websockets para conexión en tiempo real entre el frontend y el backend, este backend será una 3ra API realizada en NodeJS , de esta manera el consumo de recursos estará optimizada para gran cantidad de información.

CLOUD SQL

Es un servicio de base de datos relacional totalmente administrado. Proporciona un almacenamiento eficiente y escalable para los datos del sistema de calificaciones. Para este Proyecto se debe de Crear una Base de Datos Mysql, el modelo de la base de datos queda a discreción del estudiante.

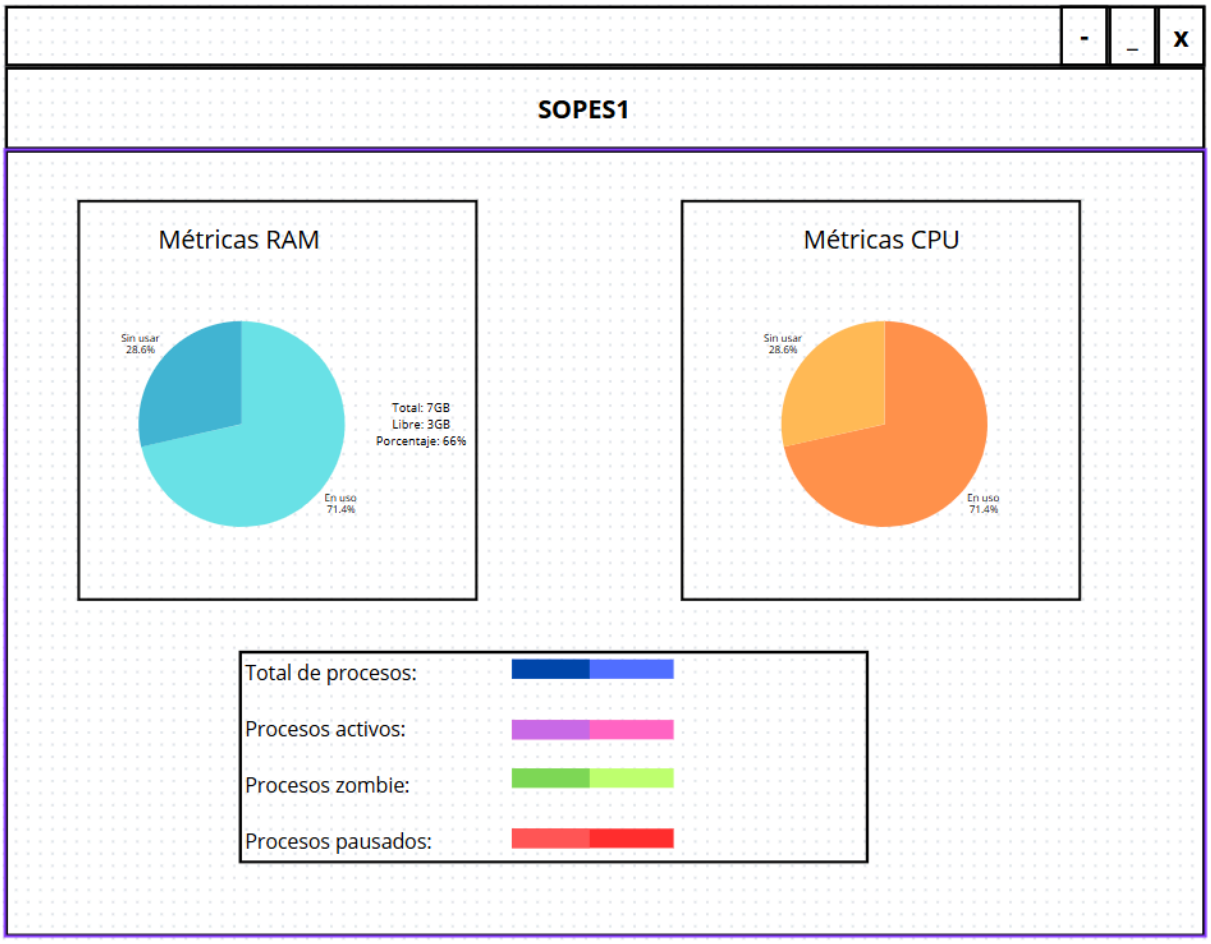
CLOUD RUN

Permite ejecutar aplicaciones en contenedores de manera completamente gestionada y sin preocuparse por la infraestructura subyacente. Se utilizará el Servicio de Cloud Run para desplegar y virtualizar nuestra aplicación frontend.

APLICACIÓN WEB

Se debe crear un sitio web para mostrar en tiempo real los datos cargados, React para su desarrollo, este debe ser un Dashboard y mostrar lo siguiente:

- Gráfica en Tiempo Real del porcentaje de utilización de la memoria RAM.
- Gráfica en Tiempo Real del porcentaje de utilización del CPU.
- Tabla con la información de los procesos.



Restricciones

• El proyecto se realizará de forma individual.
• La obtención de la información se hará a través de los módulos de Kernel en C.
• El Agente de Monitoreo debe ser realizado con Golang.
• Una API debe ser realizada con NodeJS, la otra debe ser realizada en Python y la última debe ser realizada con NodeJS utilizando websockerts.
• El Frontend debe ser realizado en React y utilizar sockets.io.
• Las máquina local debe tener instalada una distribución de Ubuntu en su versión 22.04 o 24.xx.
• Las imágenes de los contenedores deben de ser publicadas y consumidas por medio de DockerHub.
* Todo debe estar contenerizado a excepción de la base de datos y Locust.

Github

- El código fuente debe de ser gestionado por un repositorio privado de Github
- Crea una Carpeta en el repositorio llamado: Proyecto_Fase2
- Agregar al auxiliar al repositorio de GitHub: **dialjub19**

Calificación

- Al momento de la calificación se verificará la última versión publicada en el repositorio de GitHub
- El estudiante debe tener todo listo al momento de la calificación en caso contrario obtendrá una nota 0.
- No se permite la modificación o alteración de archivos o código de la aplicación a menos que el auxiliar lo permita o solicite.
- La calificación tendrá una duración de 15min.
- Se podrá solicitar explicaciones o alteraciones de código por medio del auxiliar para validar la autenticidad del proyecto presentado.
- Cualquier copia parcial o total tendrán nota de 0 puntos y serán reportadas al catedrático y a la Escuela de Ciencias y Sistemas.
- Si el proyecto se envía después de la fecha límite, se aplicará una penalización del 25% en la puntuación asignada cada 12 horas después de la fecha límite. Esto significa que, por cada período de 12 horas de retraso, se reducirá un 25% de los puntos totales posibles. La penalización continuará acumulándose hasta que el trabajo alcance un retraso de 48 horas (2 días). Después de este punto, si el trabajo se envía, la puntuación asignada será de 0 puntos.
- Repositorios vacíos o con la app incompleta tendrán nota 0.

Entregables

- La forma de entrega es mediante UEDI subiendo el enlace del repositorio.
- Manual técnico en formato Mark Down o PDF.

Fecha de Entrega

- Domingo 29 de junio de dos mil veinticinco (29/06/2025) antes de las 23:59 hrs.
- **La Calificación es en modalidad virtual.**