UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

ESCUELA DE CIENCIAS Y SISTEMAS

SISTEMAS OPERATIVOS 1 SECCIÓN N

ING. JESUS GUZMAN POLANCO

AUX. JOSÉ DANIEL VELÁSQUEZ OROZCO

AUX. JHONATHAN DANIEL TOCAY

SEGUNDO SEMESTRE 2023

PROYECTO 1

Plataforma de Monitoreo en GCP

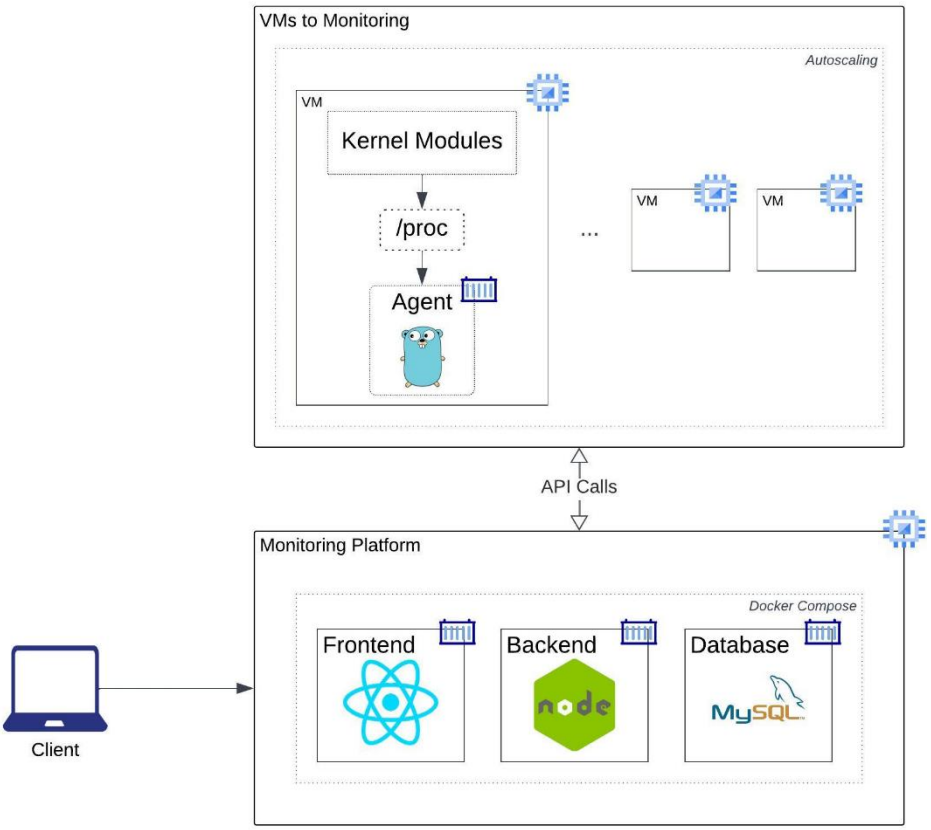
Objetivos

- Conocer el Kernel de Linux mediante módulos de C.
- Hacer uso de programación asíncrona con rutinas de Golang.
- Comprender el funcionamiento de los contenedores usando Docker.
- Utilizar GCP Compute Engine y autoscaling.

Introducción

En este proyecto, se tiene como objetivo principal implementar un sistema de monitoreo de recursos del sistema y gestión de procesos, empleando varias tecnologías y lenguajes de programación. El sistema resultante permitirá obtener información clave sobre el rendimiento del computador, procesos en ejecución y su administración a través de una interfaz amigable. El para el despliegue del proyecto se utilizará GCP Compute Engine y autoscaling para simular múltiples máquinas virtuales a monitorear.

Arquitectura



VMs a Monitorear

Kernel Modules

Módulo de RAM

Este módulo deberá de estar alojado en un archivo ubicado en el directorio **/proc**.

Características:

- Importar la librería **< sys/sysinfo.h>**, **<linux/mm.h>**
- La información que se mostrará en el módulo debe ser obtenida por medio de los struct de información del sistema operativo y no por otro medio.
- El nombre del módulo será: **ram_<carnet>**

Módulo de CPU

Este módulo deberá de estar alojado en un archivo ubicado en el directorio **/proc**.

Características:

- Importar la librería **<linux/sched.h>**, **<linux/sched/signal.h>**
- La información que se mostrará en el módulo debe ser obtenida por medio de los struct de información del sistema operativo y no por otro medio.

- El nombre del módulo será: **cpu_<carnet>**

Agente de Monitoreo

Este es un programa escrito en Golang y contenerizado, que es instalado en cada una de las VMs a monitorear. Este permite la comunicación este la VM y la Plataforma de Monitoreo. Este cuenta con los siguientes componentes:

- Recolector

Se encarga de realizar llamadas los módulos de Kernel por medio de rutinas para obtener la información del estatus de CPU y RAM. Así mismo este enviará la información al API de NodeJS de la Plataforma de Monitoreo para que sean almacenadas en la base de datos MySQL.

- Service Killer

Expone un API para matar algún proceso por medio de su PID mediante llamadas a señales “KILL (-9)”. El API será llamada por el Frontend de la Plataforma de Monitoreo.

Plataforma de Monitoreo

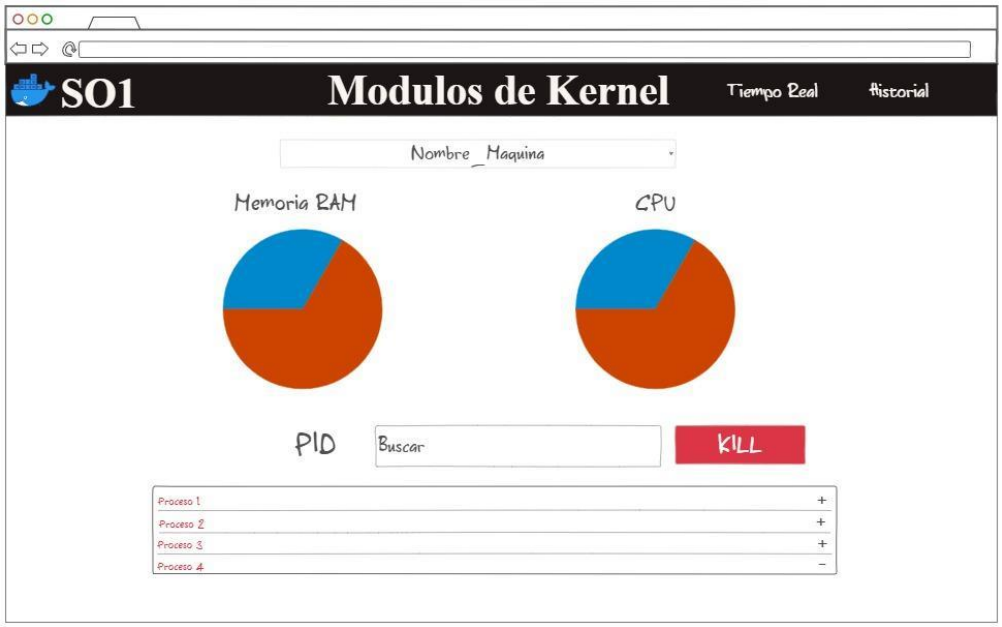
Frontend

Se debe de crear una Página Web que se divide en:

Monitoreo en Tiempo Real

Se debe mostrar

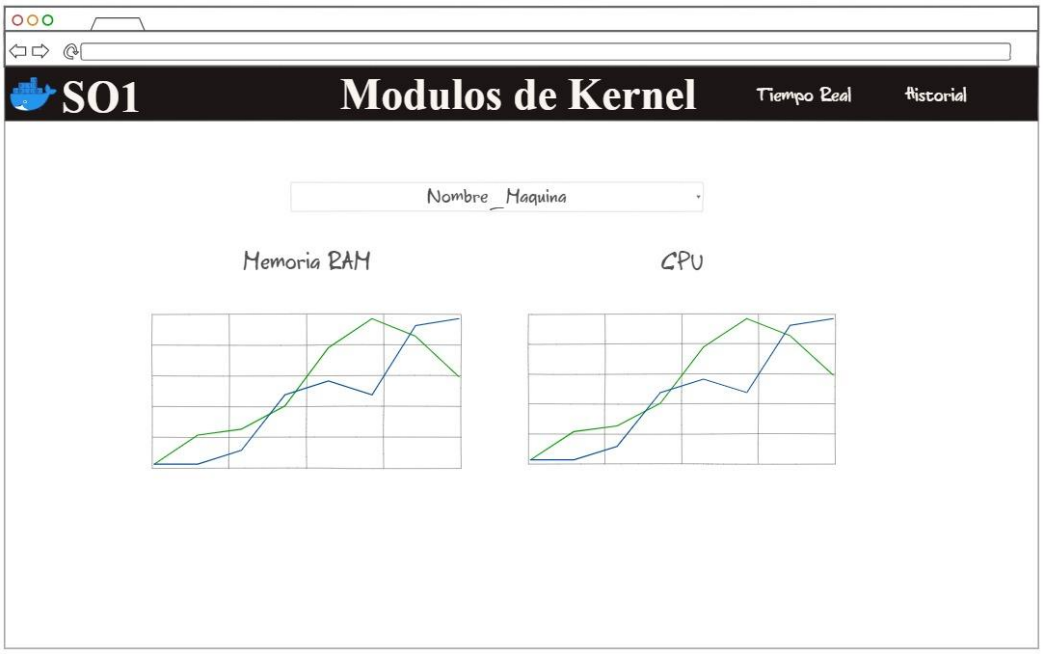
- Un **Componente tipo Select** en el cual se puede elegir la máquina de la cual se visualizará los datos.
- Grafica en Tiempo Real del porcentaje de utilización de la memoria RAM.
- Grafica en Tiempo Real del porcentaje de utilización del CPU.
- Una tabla donde se muestren los Procesos, en esta se debe de mostrar la siguiente información para cada Proceso
    - **PID**, Identificador del Proceso
    - **Nombre** del Proceso
    - **Usuario** que ejecuto el Proceso
    - **Estado** en el que se encuentra el Proceso
    - **%RAM** que utiliza el proceso
- Un campo de texto para buscar y filtrar por PID o nombre de un proceso, a su vez un botón para enviar la señal que pueda eliminar dicho proceso.



Monitoreo a lo largo del Tiempo

En esta página de debe de Agregar:

- Un Select para elegir la máquina de la cual se visualizará la información.
- Una Grafica del Rendimiento a lo largo del tiempo de la RAM.
- Una Grafica de Rendimiento a lo largo del tiempo del CPU.



API NODEJS

Este servicio expone diversas APIs que posibilitan la comunicación con la base de datos. Dichas APIs serán empleadas tanto para la lectura de información desde el Frontend como para la escritura de datos desde los agentes instalados en las máquinas sujetas a monitoreo.

Base de datos

Se debe implementar una base de datos MySQL por medio de un contenedor de Docker, esto para guardar los datos de los procesos, hijos, información de memoria ram y cpu. La base de datos debe de tener persistencia por lo que se requiere un Volumen de Docker para evitar que los datos se pierdan cada vez que el contenedor se reinicia.

Docker Compose

Para facilitar el despliegue de los contendores de la Plataforma de Monitoreo se utilizará Docker Compose el cual permite gestionar múltiples contendores en un solo bloque lo cual facilita la administración.

DockerHub

Se debe de utilizar DockerHub para alojar las imágenes de los contenedores utilizados.

Google Cloud Platform

Plataforma de Monitoreo

Se deberá de crear una máquina virtual en GCP (Compute Engine) en la cual se levantará con ayuda de Docker-Compose el Frontend, API de NodeJS y el contenedor de MYSQL.

VMs a monitorear

Se deberá de crear un Grupo de Instancias y se deberá de configurar el Autoscaling indicando la métrica de utilización a un 60%, como mínimo se tendrá 1 instancia y máximo 4. Y dentro de cada instancia debe de estar alojado los módulos de Kernel y el Agente para permitir la comunicación con la Plataforma de Monitoreo.

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.
- La API debe ser realizada con NodeJS.
- El Frontend debe ser realizada con React.
- Las máquinas virtuales deberán ser de Ubuntu 20.04 o 22.04.
- Las imágenes de los contenedores deben de ser publicadas de DockerHub
- El proyecto debe de ser publicado en GCP usando Compute Engine y autoscaling

Github

- El código fuente debe de ser gestionado por un repositorio privado de Github

- Nombre del repositorio: **so1_proyecto1_2S2023_<*carnet*>**
- Agregar a los auxiliares al repositorio de GitHub: **DannyLaine158** y **JhonathanTocay2020**

Calificación

- Al momento de la calificación se verificará la última versión publicada en el repositorio de GitHub
- 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 deducirá 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.

Entregables

- **La entrega se debe realizar antes de las 23:59 del 21 de septiembre de 2023.**
- **La forma de entrega es mediante UEDI subiendo el enlace del repositorio.**
- Manual técnico con explicación de todos los componentes utilizados en el proyecto por ejemplo Web UI, Database, Service Killer, Módulos, GPC, etc. Este manual debe ser un archivo .md (Markdown), por lo que deberá estar en el README del repositorio.

Referencias

- Docker
- Docker Compose
- GCP Compute Engine
- Autoscaling