UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
FACULTAD DE INGENIERÍA
ESCUELA DE CIENCIAS Y SISTEMAS
SISTEMAS OPERATIVOS 1
SEGUNDO SEMESTRE 2021
ING. SERGIO ARNALDO MENDEZ AGUILAR
AUX. CARLOS DAVID RAMIREZ ALTAN

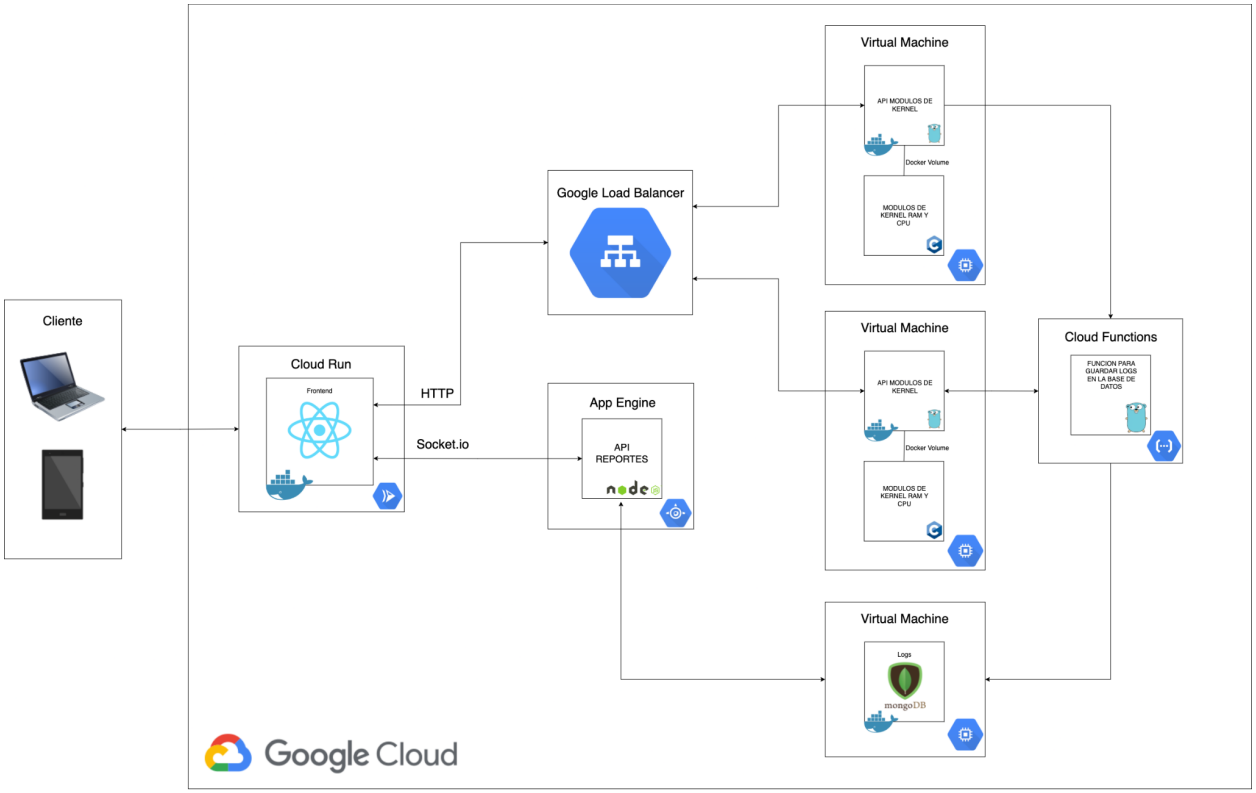
PROYECTO FASE 1

Realizar un sistema computacional distribuído, cloud native, utilizando diferentes servicios de Google Cloud Platform, virtualización a nivel de sistema operativo con Docker.

OBJETIVOS

- Aplicar conocimiento de contenedores, imágenes, archivos de composición y redes entre contenedores.
- Experimentar y utilizar tecnologías nativas de la nube que ayudan a desarrollar sistemas distribuidos modernos.
- Crear una app web utilizando React, uno de los frameworks con más demanda laboral en el mercado.
- Utilizar una amplia gama de servicios de la nube IaaS, PaaS y SaaS del proveedor GCP.
- Utilizar lenguajes modernos para la creación de una aplicación distribuída.
- Conocer el kernel de Linux y los módulos que actúan sobre el directorio /proc.
- Creación de módulos.
- Conocer la planificación de procesos de Linux.
- Conocer los estados de los procesos en los sistemas basados en Unix.

ARQUITECTURA



PRIMERA PARTE (MÓDULOS KERNEL, API MÓDULOS DE KERNEL CON GO)

Módulo RAM:

Se debe implementar un módulo de kernel que lea la RAM del sistema, dicho módulo deberá escribir un archivo en el directorio /proc el cual debe contener la siguiente información.

- Total de memoria RAM (en MB)
- Memoria RAM en uso (en MB)
- Porcentaje de memoria RAM en uso
- Memoria RAM libre (en MB)

Otras características sobre la implementación del módulo de RAM:

- Debe imprimir el mensaje “Módulo RAM del Grupo<numero_de_grupo> Cargado” al momento de cargar el módulo (insmod).

- Debe imprimir el mensaje “Módulo RAM del Grupo<numero_de_grupo> Desmontado” al momento de remover el módulo (rmmod).
- La información que se mostrará en el módulo debe ser obtenida por medio de los struct de información del sistema operativo y no de la lectura de otro archivo.

El nombre del módulo será: ram_grupo<numero_grupo> ejemplo: ram_grupo22

Módulo lista de procesos:

Se debe implementar un módulo de kernel que lea la lista de procesos del sistema, dicho módulo deberá escribir un archivo en el directorio /proc el cual debe contener la siguiente información.

- Nombre del proceso
- PID
- PID del padre
- Estado

Otras características sobre la implementación del módulo lista de procesos:

- Debe imprimir el mensaje “Módulo lista de procesos del Grupo<numero_de_grupo> Cargado” al momento de cargar el módulo (insmod).
- Debe imprimir el mensaje “Módulo lista de procesos del Grupo<numero_de_grupo> Desmontado” al momento de remover el módulo (rmmod).
- La información que se mostrará en el módulo debe ser obtenida por medio de los struct de información del sistema operativo y no de la lectura de otro archivo.

El nombre del módulo será: cpu_grupo<numero_grupo> ejemplo: cpu_grupo22

API:

Esta parte consiste en la creación de una api para que el cliente pueda consumir la información generada por los módulos del kernel, dicha api debe ser dockerizada de manera que la forma de ejecutar la api es mediante docker, para poder acceder a la información generada por los módulos del kernel debe de ser a través de un volume de docker.

LoadBalancer:

Dado que se requieren 2 instancias de máquinas virtuales que contengan tanto el de kernel de RAM Y el módulo de listas de procesos y la api que permite su consumo, se solicita que configure un balanceador de carga el cual recibirá todo el tráfico generado por el cliente.

SEGUNDA PARTE (LOGS)

Cada vez que se realice una solicitud a la api de módulos de kernel, se deberá generar un log de dicha solicitud y deberá ser almacenado en una base de datos.

CLOUD FUNCTIONS:

Para las inserciones de los logs a la base de datos deberá ser a través de una cloud function la cual será desarrollada utilizando go, esta función será la encargada de realizar la conexión a la base de datos e insertar el registro.

DATABASE:

La base de datos a utilizar para el almacenamiento de los logs será MongoDB la cual es una base de datos NoSQL documental que almacena la información utilizando el formato de datos JSON, dicha instancia de base de datos deberá estar ubicada en una máquina virtual de gcp y deberá utilizar una imagen de docker para la misma.

DATA:

La información que se solicita que cada log contenga es: la máquina virtual a la que pertenece dicho log, el endpoint al cual se realizó la solicitud, información que retorno el servidor al cliente y fecha y hora en que se hizo la solicitud. Un ejemplo de log seria:


```
{
    "nombreVM": "vm1",
    "endpoint": "/getDataRAM",
    "data": [{
        "total": 4000,
        "memoriaEnUso": 2000,
        "porcentaje": 50,
        "memoriaLibre": 2000
    }],
    "date": "07/02/2022 16:20:00"
}
```

API REPORTES:

Los logs almacenados deben de poderse ver desde la aplicación web por lo que se pide desarrollar una api que permita obtener dichos logs, esta api deberá ser desarrollada utilizando nodejs y deberá ser desplegada utilizando App Engine de GCP.

QUINTA PARTE (PÁGINA WEB)

Vista RAM: La vista de memoria ram consiste en un monitor que mostrará de forma gráfica en tiempo real el consumo de la memoria ram de la máquina virtual en la cual fue cargado el modulo de RAM por lo que para esta vista debe consumir la información que provee dicho módulo mediante la api desarrollada en go, esta vista deberá contener la siguiente información:

- Total de memoria RAM (en MB)
- Memoria RAM en uso (en MB)
- Porcentaje de memoria RAM en uso
- Memoria RAM libre (en MB)
- Gráfica de Memoria en uso.

Vista lista de procesos: En esta vista debe mostrar de manera tabulada todos los procesos que están siendo ejecutados en la máquina virtual en la cual fue cargado el módulo de lista de procesos, por lo que para esta vista debe consumir la información que provee dicho módulo mediante la api desarrollada en go, esta vista deberá contener la siguiente información:

- Nombre del proceso
- PID
- PID del padre
- Estado

Se deberá mostrar un árbol de procesos dinámico, es decir, se debe mostrar una lista de procesos que permita seleccionar uno y desplegar a todos sus hijos con PID y nombre.

Lista de procesos

VM	Nombre del proceso	PID	PID del padre	Estado	Hijos
vm1	kernel_task	0	0	running	Hijos
vm2	Proceso X	944	1	sleeping	
vm2	Proceso Y	2663	1	sleeping	

Deployment: Para el despliegue del frontend se debe utilizar el servicio cloud run de GCP.

Notas:

- **La aplicación web debe contar con vista para los datos de las 2 máquinas virtuales, lo que quiere decir que se debe tener una vista los datos del los módulos del kernel de la máquina A y otra para los datos de la máquina B pueden estar en la misma página o en otra eso queda a discreción del estudiante.**
- **El consumo de la api de información de los módulos del kernel se hará mediante el protocolo HTTP.**
- **El consumo de la api de logs se hará mediante sockets.**

OBSERVACIONES

- Todo debe implementarse utilizando el lenguaje o la herramienta especificada.
- La fase del proyecto debe realizarse en grupos con un máximo de 3 integrantes.
- El uso de GCP es obligatorio para desplegar todo lo solicitado.
- Deben escribir un manual de usuario y un manual técnico en el repositorio del proyecto utilizando Markdown.
- Las copias detectadas tendrán una nota de cero puntos y serán reportadas a la Escuela de Ciencias y Sistemas.
- No se aceptarán entregas después de la fecha y hora especificada.
- Se deberá agregar al usuario **racarlosdavid** al repositorio del proyecto.

ENTREGABLES

- Link del repositorio, debe incluir todo el código fuente con los archivos de manifiesto o cualquier archivo adicional de configuración.
- Manuales.

FECHA DE ENTREGA

11 de marzo antes de las 23:59 a través de UEDi. No hay prórrogas.