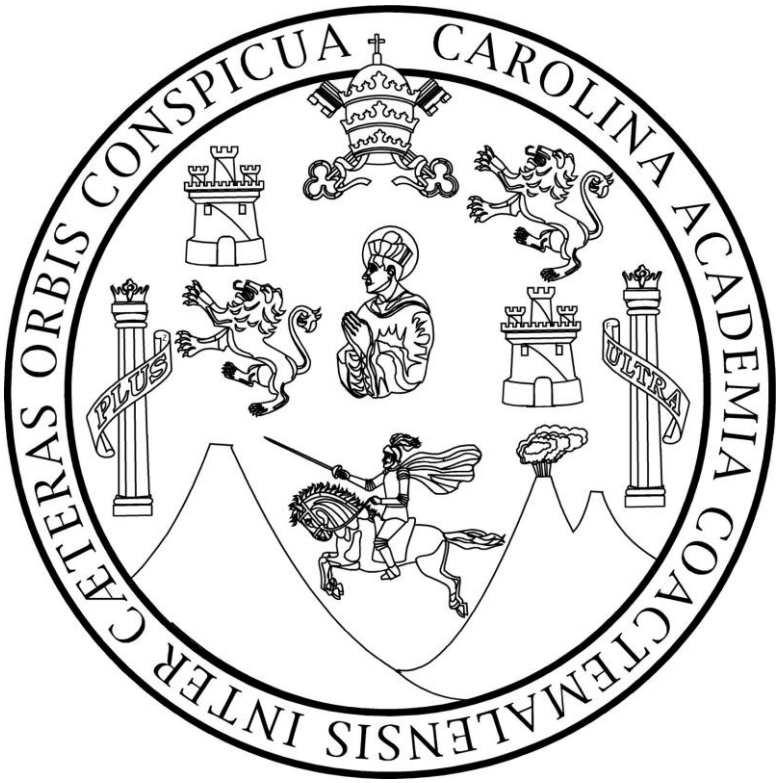
Universidad de San Carlos de Guatemala
Facultad de ingeniería
Sistemas Operativos 1
Sección A



Manual técnico

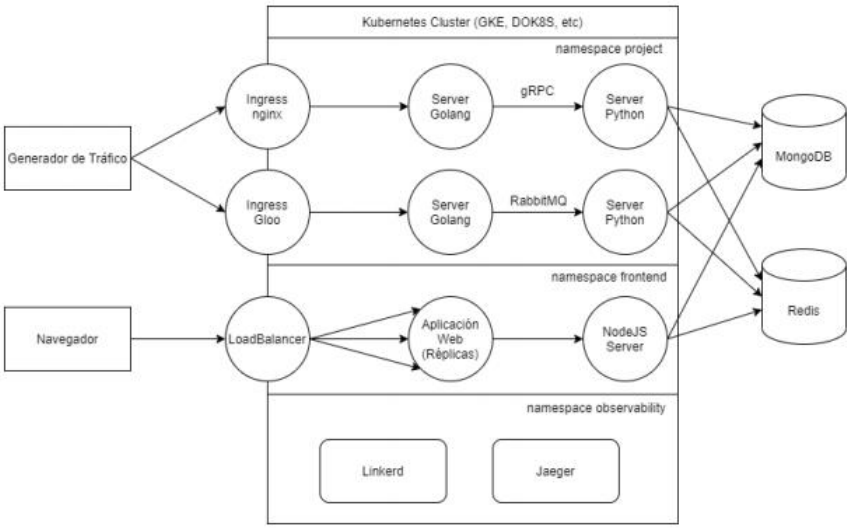
Grupo 6

14 de octubre de 2020

Proyecto

Descripción del problema

Se requiere construir un sistema genérico distribuido que muestre estadísticas en tiempo real utilizando Kubernetes y tecnologías de Cloud Native. Este proyecto se utilizará para analizar los casos de infectados de COVID-19 alrededor del mundo.



Entorno de desarrollo

El proyecto fue desarrollado tomando en cuenta el siguiente entorno y lenguajes de programación.

- Google Cloud
- Go
- Python 3
- HTML
- JavaScript

Fue desarrollado con la utilización de las siguientes herramientas:

- Google Cloud SDK
- Google Kubernetes
- Jaeger
- Linkerd
- Google balancer

Descripción de herramientas

- Google Cloud SDK: Google Cloud Platform, ofrecido por Google, es un conjunto de servicios de computación en la nube que se ejecuta en la misma infraestructura que Google usa internamente para sus productos de usuario final, como Búsqueda de Google, Gmail, almacenamiento de archivos y YouTube.
- Kubernetes es un sistema de orquestación de contenedores de código abierto para automatizar la implementación, el escalado y la administración de aplicaciones informáticas. Fue diseñado originalmente por Google y ahora es mantenido por Cloud Native Computing Foundation

- Flask: Flask es un marco de micro web escrito en Python. Se clasifica como un microframework porque no requiere herramientas o bibliotecas particulares. No tiene una capa de abstracción de base de datos, validación de formularios ni ningún otro componente donde las bibliotecas de terceros preexistentes proporcionan funciones comunes
- MongoDB: MongoDB es un programa de base de datos orientado a documentos multiplataforma. Clasificado como un programa de base de datos NoSQL, MongoDB usa documentos similares a JSON con esquemas opcionales. MongoDB es desarrollado por MongoDB Inc. y tiene licencia de Server Side Public License.

Descripción del código

El código implementado varía dependiendo del servidor que fue configurado. Esto debido a las distintas necesidades que tiene cada uno de estos.

El proyecto consta de tres partes las cuales son:

- Generador de carga

Se utilizo un programa escrito en Golang. Este se encarga de realizar distintas llamadas a servidores mediante la utilización de GoRutinas. Estas son rutinas que se ejecutan simultáneamente junto a otras rutinas.

El código se puede observar a continuación

```
server:= "https://aaddbb.free.beeceptor.com/test"
        solicitudes:= 4;
        archivo:="data.json"

        jsonFile, err := os.Open(archivo)
        if err != nil {
            fmt.Printf("Error leyendo archivo: %v", err)
        }
        defer jsonFile.Close()

        byteValue, _ := ioutil.ReadAll(jsonFile)

        var casos Casos
        json.Unmarshal(byteValue,&casos)
        if solicitudes < len(casos.Casos){
            fmt.Println("It is")
        }else{
            solicitudes = len(casos.Casos)
            fmt.Println("It is not")
        }

        for i := 0; i < solicitudes; i++ {
            fmt.Println("Name: " + casos.Casos[i].Name)
            sendata(server,casos.Casos[i])
        }
    } else{
        fmt.Println("Soy 0")

    }

}

func sendata(server string,sending Caso) {
    requestBody, err := json.Marshal(sending)
    if err != nil {
        fmt.Printf("Error convirtiendo a JSON: %v", err)
    }
    resp, err := http.Post(server,"application/json",bytes.NewBuffer(requ
estBody))
    if err != nil {
        fmt.Printf("Error en petición post: %v", err)
```

```
    }

    defer resp.Body.Close()
    body, err := ioutil.ReadAll(resp.Body)
    if err != nil{
        fmt.Printf("Error leyendo peticion: %v", err)
    }
    fmt.Printf(string(body))
```

- Backend: encargada del funcionamiento y envio de colas mediante RabitMQ. Este cuenta con dos servidores, uno escrito en Python y el otro en Golang que se determinaron por un estándar especifico. Acá se maneja también la comunicación con una base de datos en Redis
- Frontend: se encarga del frontend de la aplicación. Este maneja una pagina web utilizando React que realiza un despliegue para visualizar la información.