Tarea #5  
Servicio con gRPC, Go y Docker

línea corta



**PONDERACIÓN: 100**

**Horas Aproximadas: 8**

Universidad San Carlos de Guatemala

Facultad de ingeniería.

Ingeniería en ciencias y sistemas

# Introducción

# Introducción Esta tarea se enfoca en el desarrollo de un sistema distribuido simple utilizando gRPC, un moderno framework de RPC de alto rendimiento. Los estudiantes implementarán un servicio de "Saludos" en Go, con un componente servidor y un componente cliente. Ambos serán contenerizados con Docker y orquestados con Docker Compose para facilitar su comunicación. Opcionalmente, se explorará el uso de un registro de contenedores privado (Harbor). Esta tarea es crucial para entender la comunicación entre microservicios y las prácticas de contenerización.

# Competencia(s)

Competencia(s) Al finalizar esta tarea, el estudiante será capaz de:

* Definir servicios y mensajes utilizando Protocol Buffers (Protobuf).
* Generar código Go a partir de definiciones Protobuf para gRPC.
* Implementar un servidor gRPC en Go que ofrezca un servicio específico.
* Implementar un cliente gRPC en Go que consuma dicho servicio.
* Contenerizar aplicaciones cliente y servidor utilizando Docker.
* Orquestar la ejecución y comunicación de múltiples contenedores con Docker Compose.
* (Opcional) Publicar y gestionar imágenes de Docker en un registro como Harbor

# Contenido

**Objetivo:** Desarrollar un sistema simple cliente-servidor utilizando gRPC con Go. El servidor ofrecerá un servicio básico de "Saludos" y el cliente consumirá dicho servicio. Ambos componentes deberán ser contenerizados usando Docker y deberán poder comunicarse entre sí dentro de la red de Docker. Opcionalmente, las imágenes de Docker se subirán a un registro de Harbor.

**Tecnologías a Utilizar:**

* Go (Golang)
* gRPC
* Protocol Buffers (Protobuf)
* Docker
* Docker Compose
* (Opcional) Harbor

**Pasos a seguir:**

1. **Definición del servicio (Protocol Buffers):**
   * Crear un archivo .proto que defina un servicio Saludador con un método RPC (e.g., Saludar) que tome un mensaje SolicitudSaludo (conteniendo un nombre) y devuelva un mensaje RespuestaSaludo (conteniendo un mensaje de saludo).
   * Como sugerencia utilizar la misma estructura para su proyecto.
2. **Generar código de Go:**
   * Utilizar protoc con los plugins de Go y gRPC para generar el código stub.
3. **Implementación del servicio gRPC (Servidor en Go):**
   * Crear el proyecto del servidor.
   * Escribir el código del servidor que implemente la interfaz generada del servicio Saludador.
4. **Implementación del Cliente gRPC (Cliente en Go):**
   * Crear el proyecto del cliente.
   * Escribir el código del cliente que se conecte al servidor gRPC y llame al método Saludar.
5. **Contenerización con Docker y docker-compose:**
   * Crear un Dockerfile para el servidor.
   * Crear un Dockerfile para el cliente.
   * Crear un archivo docker-compose.yaml que defina los servicios del servidor y cliente, asegurando que puedan comunicarse.
6. **Ejecución y verificación:**
   * Construir y ejecutar los contenedores usando docker-compose up.
   * Verificar que el cliente pueda enviar una solicitud al servidor y recibir una respuesta correcta (visible en los logs o salida estándar).

**Puntos extras (opcional): Subir Imágenes a Harbor**

1. Crear una VM en GCP (o usar una existente).
2. Desplegar una instancia de Harbor en la VM (seguir documentación oficial).
3. Etiquetar y subir las imágenes Docker del servidor y cliente a su instancia de Harbor.
4. Modificar el docker-compose.yaml para que obtenga las imágenes desde Harbor.
5. Validar el correcto funcionamiento.
   * Referencia para Harbor: <https://goharbor.io/docs/2.12.0/install-config/run-installer-script/>

# Recursos Adicionales

* Documentación oficial de gRPC: <https://grpc.io/docs/>
* Documentación oficial de Protocol Buffers: <https://developers.google.com/protocol-buffers/docs/overview>
* Documentación oficial de Docker Compose: <https://docs.docker.com/compose/>

# Forma de entrega

* Archivo PDF que incluya:
  + Contenido del archivo .proto.
  + Capturas de pantalla del código clave del servidor y cliente Go.
  + Contenido de los Dockerfiles (servidor y cliente).
  + Contenido del archivo docker-compose.yaml.
  + Capturas de pantalla que demuestren la comunicación exitosa entre cliente y servidor.
  + Si se realiza la parte opcional de Harbor: capturas de la instancia de Harbor con las imágenes, docker-compose.yaml modificado y prueba de funcionamiento.
* El archivo debe nombrarse Carnet\_T5.pdf.
* Entregar en Classroom o UEDI con el Auxiliar correspondiente.

# Fecha de entrega

Fecha límite para realizar la entrega de la tarea: 7 días a partir de la asignación de la tarea.

# Rúbrica de Calificación

Establecer de una forma clara los aspectos que se calificarán

|  |  |  |
| --- | --- | --- |
| **Criterio** | **Descripción** | **Puntos Máximos** |
| **Definición Protobuf (.proto)** | El archivo .proto define correctamente el servicio, métodos y mensajes. | **20** |
| **Implementación Servidor gRPC (Go)** | El servidor implementa correctamente el servicio definido y se ejecuta sin errores. | **20** |
| **Implementación Cliente gRPC (Go)** | El cliente se conecta al servidor y consume el servicio correctamente. | **20** |
| **Dockerfiles (Servidor y Cliente)** | Los Dockerfiles son correctos y construyen imágenes funcionales para el servidor y el cliente. | **20** |
| **Docker Compose y Comunicación** | El docker-compose.yaml define y orquesta los servicios correctamente, permitiendo la comunicación entre cliente y servidor. | **20** |
| **(Opcional) Implementación con Harbor** | Se despliega Harbor, se suben las imágenes y se prueba el funcionamiento con docker-compose usando imágenes de Harbor. (Puntos Extra) | **20** |

# Valores

Valores Establecer de una forma clara que pasa si no se cumplen con los valores del laboratorio: El trabajo debe ser individual. Cualquier indicio de copia total o parcial resultará en la anulación de la tarea para todos los involucrados. Se espera que el estudiante realice el desarrollo de manera ética y profesional, consultando los recursos adicionales como guía y no como fuente de copia directa. Las dudas deben realizarse en el foro semanal.

guion corto