



FUNDAMENTOS DE INTERNET DE LA COSAS

2023-2024

Universidad Carlos III de Madrid

Cuaderno 2- Ejercicio - 7

2023/2024

INTRODUCCIÓN A CONTENEDORES Y GEMELO
DIGITAL

Cuaderno 2 - Ejercicio 7

Universidad Carlos III de Madrid. Escuela Politécnica Superior

Objetivos

El propósito de este ejercicio consiste en:

- Introducir las capacidades de la Google Cloud Platform que estará accesible para el cuaderno 2 de la asignatura y canjear los cupones para su uso.
- Poner en práctica los conceptos básicos para la creación de una imagen de contenedor y su lanzamiento.
- Desarrollar un gemelo digital que se comporte como el simulador del vehículo desarrollado en el cuaderno 1 de la asignatura.
- Utilizar la API de Google Maps para calcular la ruta entre dos puntos y trocearla en segmentos de longitud corta para la generación de comandos que guíen al vehículo.

Introducción a Google Cloud Platform y canjeo de los cupones de prácticas

Introducción a Google Cloud Platform

Imagina que eres un arquitecto de software para el portal foo.com, una aplicación accesible en Internet. Hay muchas maneras diferentes de diseñar una aplicación de este tipo. Se va a utilizar este ejemplo para ver cuáles son las capacidades que proporciona la Google Cloud Platform (GCP) para que cuando un usuario abre el navegador y escribe foo.com en la barra de direcciones tenga el comportamiento de cualquier página web.

Para servidores web y de aplicaciones tienes múltiples opciones en Cloud Run, App Engine, GKE y Compute Engine. Echa un vistazo a dónde debería ejecutar mis cosas para más detalles.

- **Serverless:** Si tienes un equipo de desarrolladores, quieres que se centren en la codificación y no se preocupen por la infraestructura y las tareas de escalado. Cloud Run o App Engine serían buenas opciones. Ambos son sin servidor y escalan de bajo a alto tráfico según sea necesario. Si desea ejecutar contenedores sin servidor que sirvan arquitecturas de microservicios web y basadas en eventos, se recomienda Cloud Run. Cloud Run debería funcionar para la mayoría de los casos de uso, echa un vistazo a App Engine si estás desarrollando sitios web con alojamiento de archivos estáticos integrado.
- **Google Kubernetes Engine (GKE):** Si desea ejecutar aplicaciones en contenedores con más opciones de configuración y flexibilidad, puede utilizar GKE. Le ayuda a desplegar fácilmente aplicaciones en contenedores con Kubernetes a la vez que le ofrece control sobre la configuración de los nodos. El escalado también es sencillo; puede definir el número de nodos a escalar a medida que crece el tráfico. GKE también ofrece piloto automático, cuando se necesita flexibilidad y control pero el soporte de operaciones e ingeniería es limitado.

- **Compute Engine:** Se trata directamente de máquinas virtuales (VM), por lo que puede definir con precisión la configuración de sus máquinas en función de la cantidad de memoria y CPU que necesite. Sin embargo, este nivel de control implica una mayor responsabilidad a la hora de escalar, gestionar, parchear y mantener las máquinas virtuales según sea necesario. Compute Engine funciona bien para aplicaciones heredadas con necesidades específicas y en situaciones que realmente requieren un control total.

Por supuesto, foo.com necesita una o varias bases de datos para almacenar la información. Pueden ser bases de datos relacionales o no relacionales, dependiendo del tipo de datos y del caso de uso. (Para obtener información más detallada sobre cómo elegir la base de datos adecuada para tu caso de uso, consulta [Explicación de las opciones de bases de datos de Google Cloud](#)).

Las bases de datos relacionales de Google Cloud incluyen Cloud SQL y Cloud Spanner, ambas gestionadas.

- Cloud SQL es perfecto para necesidades SQL genéricas - MySQL, PostgreSQL y SQL server.
- Spanner es mejor para bases de datos relacionales de escala masiva que necesitan escalabilidad horizontal. (Masiva aquí significa miles de escrituras por segundo y decenas de miles de lecturas por segundo, a la vez que soporta transacciones ACID).
- Para bases de datos no relacionales, Google Cloud tiene tres opciones principales: Firestore, Bigtable y Memorystore.
 - Firestore es una base de datos de documentos sin servidor que proporciona una gran coherencia, admite transacciones ACID y ofrece resultados rápidos a consultas complejas. También admite datos y sincronizaciones sin conexión, lo que la convierte en una gran opción para casos de uso móvil junto con web, IoT y juegos.
 - Bigtable es una base de datos NoSQL de columnas anchas que admite lecturas y escrituras pesadas con una latencia extremadamente baja. Esto la convierte en la elección perfecta para eventos, datos de series temporales de dispositivos IoT, datos de flujo de clics, eventos publicitarios, detección de fraudes, recomendaciones y otros casos de uso relacionados con la personalización.
 - Memorystore es un servicio de almacenamiento de datos en memoria totalmente gestionado para Redis y Memcached. Es ideal para almacenes transitorios y cachés de bases de datos.

A medida que crezca el tráfico, será necesario escalar con él los servidores web y de aplicaciones. Y, a medida que crezca el número de servidores, necesitará un equilibrador de carga para dirigir el tráfico a los servidores web y de aplicaciones. Cloud Load Balancing es un sistema totalmente distribuido y definido por software basado en direcciones IP anycast, lo que significa que puede configurar su frontend con una única dirección IP. También es global, por lo que puede servir contenidos lo más cerca posible de sus usuarios y responder a más de un millón de consultas por segundo. Puede configurar decisiones de enrutamiento basadas en el contenido y en atributos, como el encabezado HTTP y el identificador uniforme de recursos. También ofrece equilibrio de carga interno para los servidores de aplicaciones internos, de modo que puede enrutar el tráfico entre ellos según sea necesario.

Todos los archivos estáticos de foo.es, como archivos multimedia e imágenes, así como CSS y JavaScript, pueden almacenarse en un almacén de objetos. En Google Cloud, Cloud Storage es su almacén de objetos para las necesidades de almacenamiento a largo y corto plazo.

Digamos que foo.com también está disponible en dispositivos móviles, que necesitan imágenes renderizadas en formatos móviles más pequeños. Puede desacoplar esta funcionalidad del servidor web y convertirla en una función como servicio con Cloud Functions. Este enfoque le permite aplicar su lógica de redimensionamiento de imágenes también a otras aplicaciones. Puede activar la función sin servidor tan pronto como se añada un archivo a Cloud Storage y convertir el archivo en múltiples formatos, almacenándolos de nuevo en el almacenamiento, donde son utilizados por el servidor web. También podría utilizar funciones sin servidor para otros casos de uso, como búsquedas de direcciones, chatbots, aprendizaje automático, etc.

Las aplicaciones como foo.com generan datos en tiempo real (por ejemplo, datos de clics) y datos por lotes (por ejemplo, registros). Estos datos deben ser ingeridos, procesados y preparados para los sistemas posteriores en un almacén de datos. A partir de ahí, los analistas de datos, los científicos de datos y los ingenieros de ML pueden seguir analizándolos para obtener información y realizar predicciones. Puede ingerir datos por lotes desde Cloud Storage o BigQuery y datos en tiempo real desde la aplicación mediante Pub/Sub, y escalar hasta ingerir millones de eventos por segundo. Dataflow, basado en Apache Beam de código abierto, puede utilizarse para procesar y enriquecer los datos por lotes y de flujo. Si pertenece al ecosistema Hadoop, puede utilizar Dataproc para el procesamiento; se trata de una plataforma Hadoop y Spark gestionada que le permite centrarse en el análisis en lugar de preocuparse por la gestión y puesta en marcha de su clúster Hadoop.

INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

Para almacenar los datos procesados se necesita un almacén de datos. BigQuery es un almacén de datos sin servidor que admite consultas SQL y puede escalar hasta petabytes de almacenamiento. También puede actuar como almacenamiento a largo plazo y un lago de datos junto con Cloud Storage. Puede utilizar los datos de BigQuery para crear un cuadro de mando en Looker y Data Studio. Con BigQuery ML puedes crear modelos ML y hacer predicciones utilizando consultas SQL estándar.

Para proyectos ML/AI puedes usar los datos en BigQuery para entrenar modelos en Vertex AI. Sus medios de comunicación, imágenes y otros conjuntos de datos de archivos estáticos de almacenamiento en la nube se pueden importar directamente en Vertex AI. Puede crear su propio modelo personalizado o utilizar los modelos preentrenados. Es una buena idea empezar con un modelo pre-entrenado, y ver si funciona para usted. Se cubren los casos de uso más comunes (incluyendo imagen, texto, vídeo y datos tabulares). Si un modelo preentrenado no funciona para su caso de uso, utilice el modelo AutoML en Vertex AI para entrenar un modelo personalizado en su propio conjunto de datos. AutoML admite todos los casos de uso comunes y no requiere código. En caso de que tenga mucha experiencia en ML y ciencia de datos, entonces puede decidir escribir su propio código de modelo personalizado en el marco de su elección.

También tienes que asegurarte de que los equipos de desarrollo y operaciones de foo.com tienen el acceso y las herramientas adecuadas para crear la aplicación y desplegarla. A medida que los desarrolladores escriben el código de la aplicación, pueden utilizar Cloud Code en el IDE para enviar el código a Cloud Build, que lo empaquetará y probará, ejecutará análisis de vulnerabilidades en el código, invocará Binary Authorization para comprobar si hay imágenes de contenedor de confianza y, una vez superadas las pruebas, desplegará el paquete en la fase de montaje. Desde allí se puede crear un proceso para revisar y promover a producción. Las imágenes de contenedor se almacenan en el Registro de artefactos, desde donde pueden desplegarse en GKE o Cloud Run. Las imágenes de Compute Engine se almacenan en su proyecto.

foo.com debe protegerse a nivel de datos, aplicación, usuario/identidad, infraestructura y cumplimiento.

Canjeo de los cupones de prácticas

Para que puedas utilizar los recursos en la Nube de Google Cloud Platform para tus prácticas en alguna asignatura con tu cuenta UC3M, tus profesores han solicitado códigos promocionales, que otorgan 50 USD por alumno y asignatura.

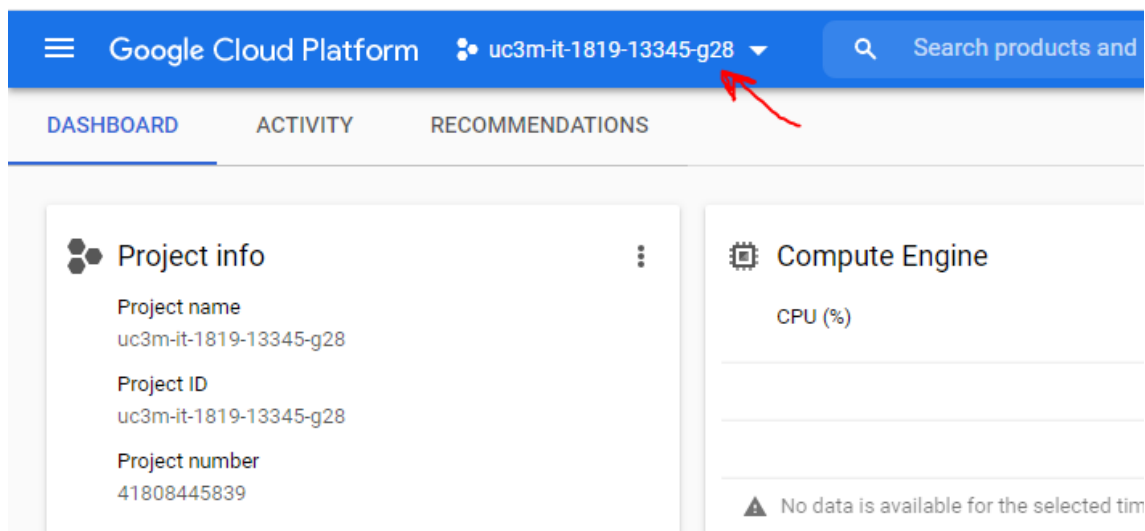
Con estos códigos, no es necesario que introduzcas una tarjeta de crédito, y en este documento te mostramos cómo canjearlos y cómo asignarlo al proyecto que ya debes tener creado usando tu cuenta de Alumno UC3M.

Entra en la consola de Google Cloud Platform con la dirección:

<https://console.cloud.google.com>

Es MUY IMPORTANTE que inicies con tu cuenta de Alumno UC3M.

Dependiendo de si has entrado antes por otras asignaturas o si es la primera vez que entras, tendrás una pantalla diferente. Lo importante es que mires los proyectos que tienes. Para ello, usaremos el selector de proyectos que tenemos en la parte superior superior de la consola:



Del listado de proyectos, tienes que identificar el que se corresponde con tu asignatura. Los profesores te indicarán cual es el código de los proyectos para la asignatura, pero todos siguen el siguiente patrón:

uc3m-<dpto>-<año>-<asig>-g<pract>

Donde:

INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

<dpto> es el código del departamento

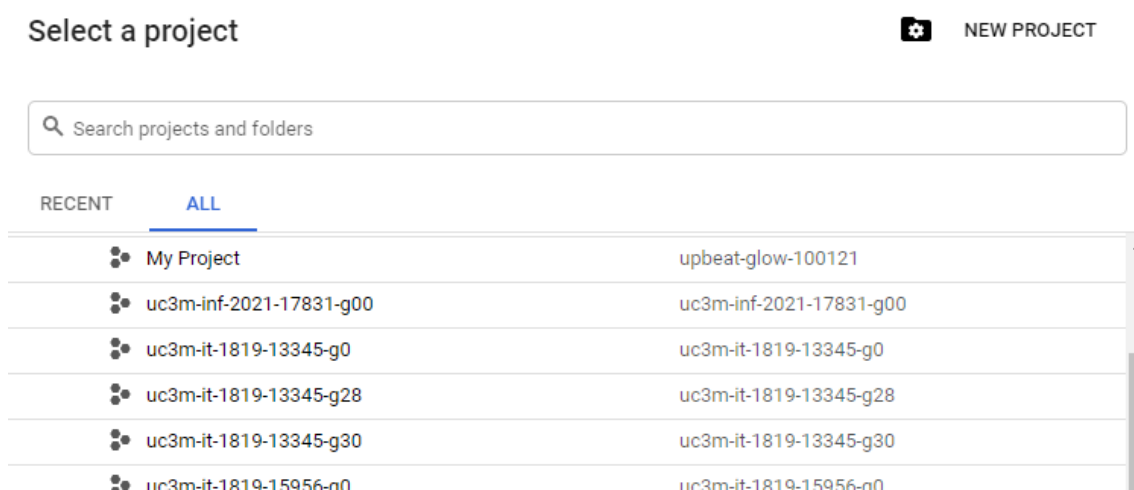
<año> es el año docente de la asignatura

<asig> es el código uc3m para la asignatura

<pract> es el número asignado a tu grupo de prácticas

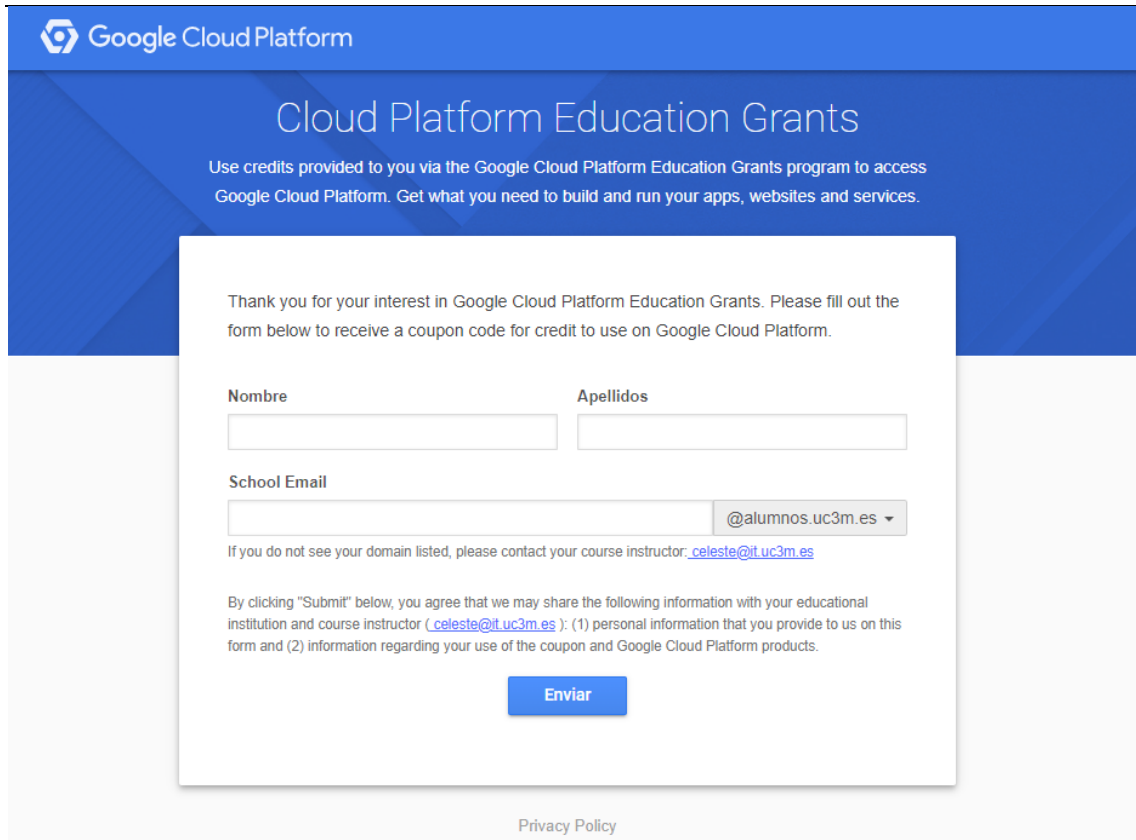
El Servicio de Informática y Comunicaciones gestiona la plataforma de Google Cloud para toda la comunidad Universitaria. Desde SDIC, se crean los proyectos y se asignan a los alumnos para realizar las prácticas.

Durante un periodo de tiempo acordado con los profesores de la asignatura, vuestras cuentas de alumno podrán crear una **Billing Account**, con el crédito promocional que se os asigna. **Una vez terminado el periodo de canjeo de cupones, no podrás crear cuentas de facturación, ni canjear cupones** y tendrás que contactar con tus profesores para resolverlo.



Para canjear tu cupón de acceso a la Google Cloud Platform (GCP) recibirá correo electrónico, un mensaje o anuncio en Aula Global por parte de tus profesores con el enlace para solicitar tu cupón. El proceso es el siguiente:

1. Confirmar tus datos (verificar el correo electrónico).

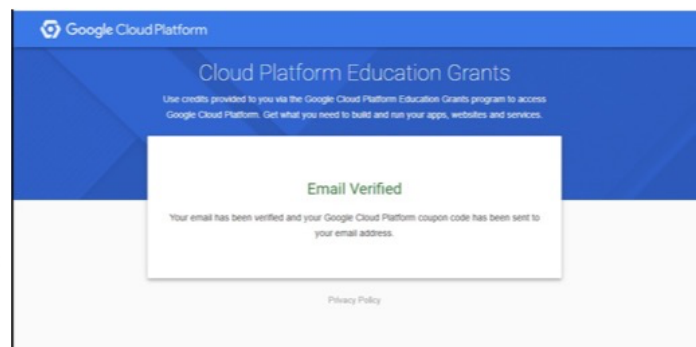
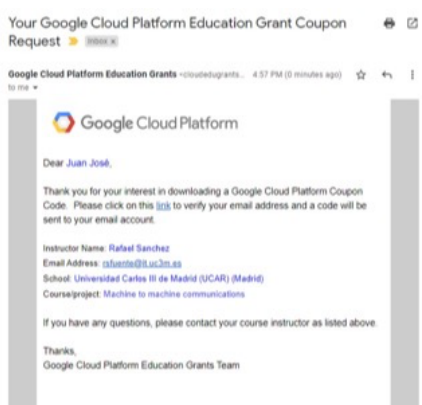


The screenshot shows the Google Cloud Platform Education Grants registration page. At the top, the Google Cloud Platform logo is visible. Below it, the title "Cloud Platform Education Grants" is displayed. A sub-header states: "Use credits provided to you via the Google Cloud Platform Education Grants program to access Google Cloud Platform. Get what you need to build and run your apps, websites and services." The main content area contains a form with the following fields: "Nombre" (First Name), "Apellidos" (Last Name), and "School Email". The "School Email" field has a dropdown menu showing "@alumnos.uc3m.es". Below the form, there is a disclaimer: "By clicking 'Submit' below, you agree that we may share the following information with your educational institution and course instructor (celeste@it.uc3m.es): (1) personal information that you provide to us on this form and (2) information regarding your use of the coupon and Google Cloud Platform products." A blue "Enviar" (Send) button is at the bottom of the form. A "Privacy Policy" link is located at the bottom of the page.

Es imprescindible que utilices la cuenta que finaliza en @alumnos.uc3m.es. En caso de que no dispongas de una cuenta de este tipo, por favor, ponte en contacto con tu profesor de prácticas para que se resuelva tu incidencia.

Una vez que se ha completado el paso anterior, recibirá un correo en tu cuenta de Alumno agradeciendo que solicites un cupón para Google Cloud Platform, y solicitando que pinches en un enlace para verificar tu cuenta de correo.

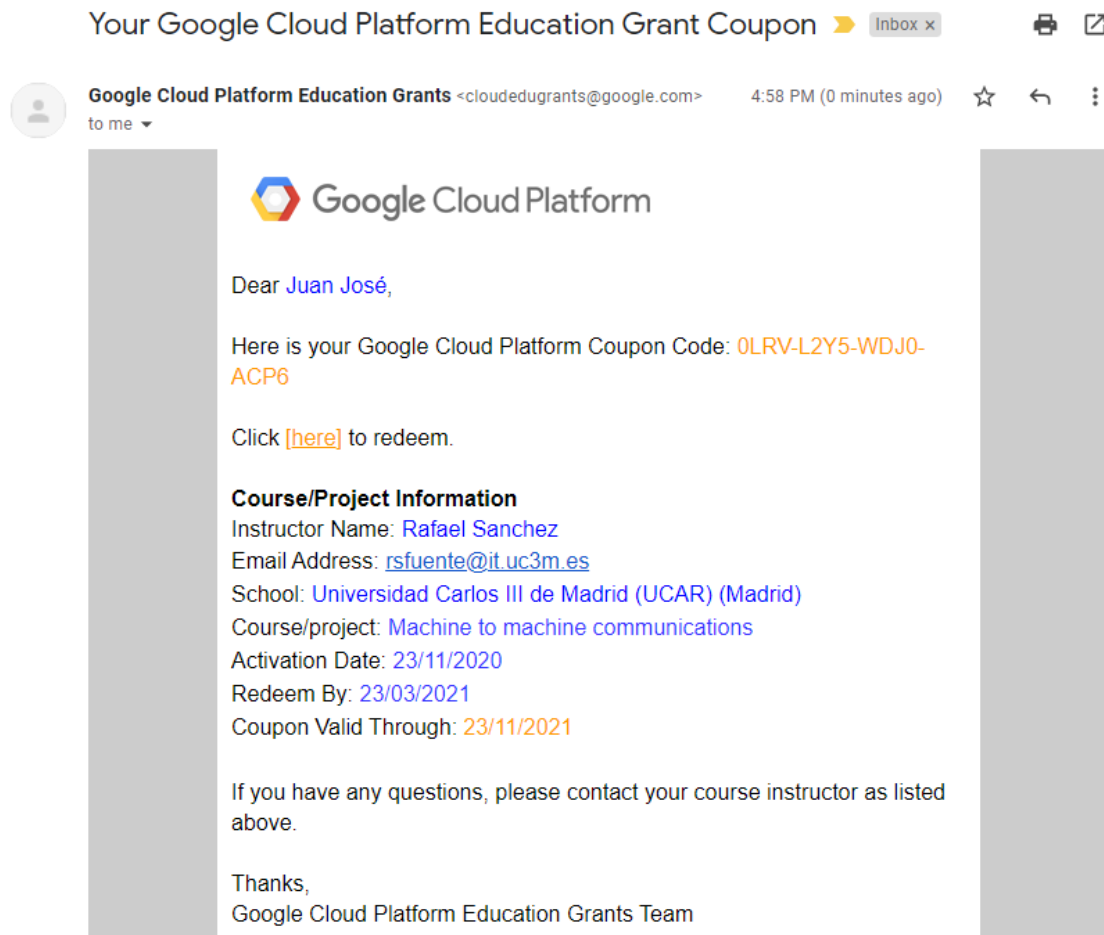
En este momento, confirme los datos y verifica tu correo electrónico para continuar con el proceso.



INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

2. Recibir el correo electrónico de Google Cloud Platform con el código para canjear tu cupón.

Una vez verificada tu dirección de correo, recibirás un correo electrónico en tu cuenta de alumno parecido a este, en el que te dan los datos de la asignatura y el enlace para poder recibir tu cupón:



Pinchamos en el enlace donde pone [here] y nos lleva a GCP con una previsualización del cupón.

3. Confirmar los datos y crear la cuenta de facturación

En la página a la que te envía, te explican que es una Beca educativa, el código de cupón ya introducido, (en algunas ocasiones puede aparecer vacío, si es el caso, introduce el código que te ha llegado al correo) la cantidad de dinero y las condiciones de uso del servicio:

Buscar productos y recursos

Becas educativas

Introduce el código de cupón que te enviamos a través del programa de becas educativas de Google Cloud Platform para recibir crédito de dicha plataforma. Consigue todo lo que necesitas para desarrollar y ejecutar tus aplicaciones, sitios web y servicios.

Código de cupón

OLRV-L2Y5-WDJ0-ACP6

Importe del crédito	Fecha de caducidad	Curso
50,00 \$	22 nov. 2021	Machine to machine communications

Términos del Servicio

País de residencia

España

Quiero recibir periódicamente correos electrónicos con noticias, novedades de productos y ofertas especiales de Google Cloud y de los Google Cloud Partners.

☒ Sí
 ☐ No

Google Cloud Platform education grants credits terms and conditions

By clicking "Accept and continue" below, you, on behalf of yourself and the organization you represent ("You") agree to these terms and conditions:

The credit is valid for Google Cloud Platform products and is subject to Your acceptance of the applicable Google Cloud Platform License Agreement and any other applicable terms of service. The credit is non-transferable and may not be sold or bartered. Unused credit expires on the date indicated on the media conveying the promotion code. The credit may be issued in increments as You use the credit over the period of time during which the credit is valid. Offer void where prohibited by law.

You represent that you are accepting the promotional credit on behalf of your educational institution and the credit can only be used on behalf of the educational entity and not for your personal use. You represent, on behalf of such educational entity, that (i) You are authorized to accept this credit; (ii) the credit is consistent with all applicable laws and regulations, including relevant ethics rules and laws; and (iii) the provision of credits will not negatively impact Google's current or future ability to do business with such educational entity.

You agree that we may share the following information with your educational institution and course instructor: (1) personal information that you provide to us during the coupon redemption process and (2) information regarding your use of the coupon and Google Cloud Platform products.

Aceptar y continuar

Borrar

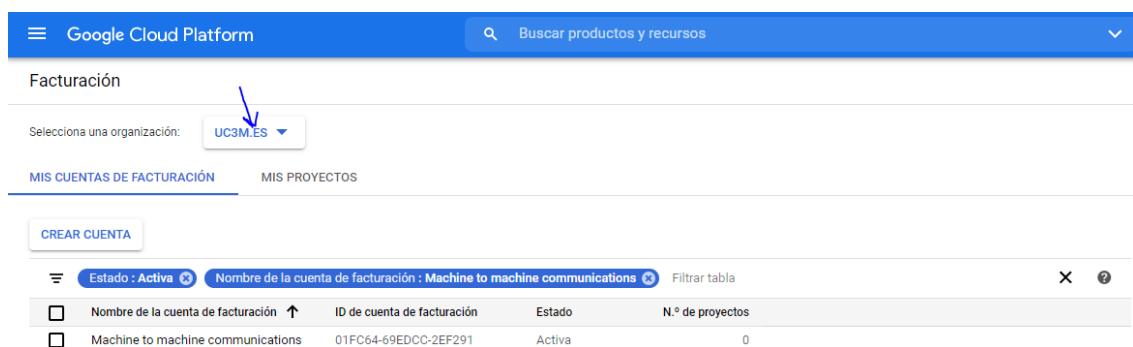
Pincha en Aceptar y continuar. Al hacerlo, se creará la cuenta de facturación con el crédito otorgado.

- Asignar la cuenta de facturación al proyecto correspondiente a tus prácticas de la asignatura.

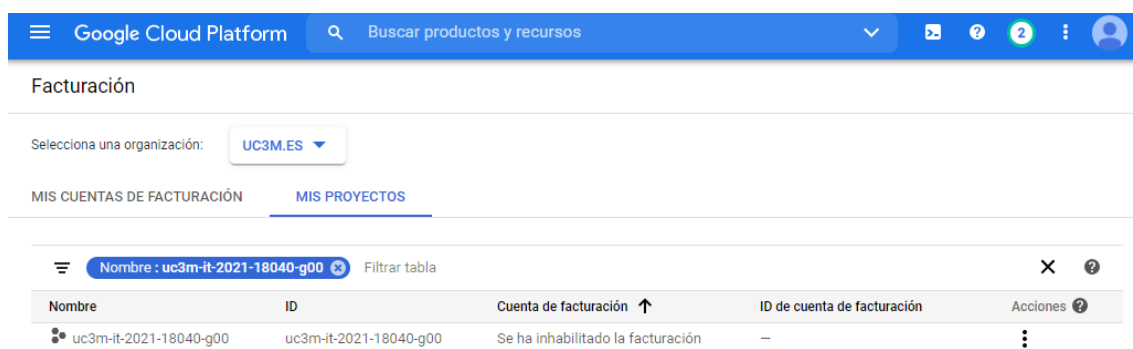
INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

Lo siguiente es, pinchar en el menú lateral, en la opción Facturación. Te mostrará todas las cuentas de facturación a las que tienes alcance. En este caso, existirá una con el nombre de la asignatura, Billing for Education Credits, o un nombre similar.

Es importante que esté seleccionada la organización “uc3m.es”.

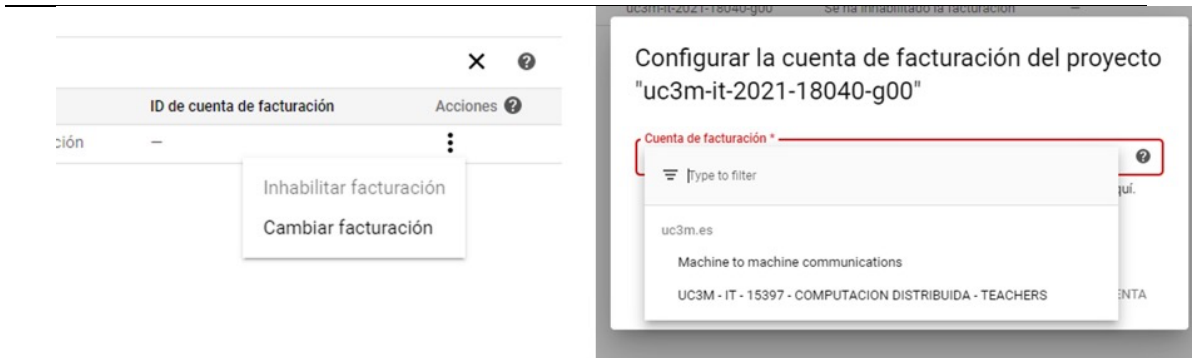


Si pinchas en **Mis Proyectos**, se mostrarán todos los proyectos en los que tienes permisos para operar:



En ese listado de la izquierda debe aparecerte el proyecto asignado a tu grupo de prácticas. Lo siguiente es irse a la columna de Acciones (los 3 puntos verticales) y pinchar en **Cambiar facturación**.

Aparecerá una figura como la de la derecha, allí debes elegir, del listado de cuentas de facturación, la que se llama como el nombre de la asignatura.



Creación de una máquina virtual

1. En la consola de Google Cloud, ve a la página **Instancias de VM**.
2. Selecciona el proyecto y haz clic en **Continuar**.
3. Haz clic en **Crear instancia**.
4. Especifica un **Nombre** para la VM. Se recomienda poner el siguiente nombre: *fic-devices*.
5. Opcional: Cambia la **Zona** para esta VM. Compute Engine aleatoriza la lista de zonas dentro de cada región para fomentar el uso en varias zonas.
6. Selecciona una **Configuración de máquina** para la VM.
7. En la sección **Disco de arranque**, haz clic en **Cambiar** y, luego, haz lo siguiente:
 - a) En la pestaña **Imágenes públicas**, elige lo siguiente:
 - Sistema operativo (Seleccionar el SO que viene por defecto: Debian)
 - Versión del SO
 - Tipo de disco de arranque
 - Tamaño de disco de arranque
 - b) Opcional: Para ver las opciones de configuración avanzadas, haz clic en **Mostrar configuración avanzada**.
 - c) Para confirmar las opciones del disco de arranque, haz clic en **Seleccionar**.
8. Para crear y, también, iniciar la VM, haz clic en **Crear**.

Obtención de la API KEY de Google Maps

Para usar Google Maps Platform, debes habilitar las APIs o los SDKs que planeas usar con tu proyecto:

https://console.cloud.google.com/apis/library/roads.googleapis.com?utm_source=Docs_EnableAPIs&utm_content=Docs_roads&hl=es-

INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

419&_gl=1*1tyrdk1*_ga*MTU2OTg0NDA0MS4xNjkyNjQzNTgy*_ga_NRWSTWS78N*
MTcwMzU5NDA1Mi42LjEuMTcwMzU5NDEzNi4wLjAuMA..

Para generar la API KEY de Google Maps, ver el siguiente vídeo:

https://youtu.be/2_HZObVbe-g

Desarrollo del gemelo digital del simulador del vehículo

En primer lugar, es necesario clonar el proyecto en gitlab (<https://teaching.sel.inf.uc3m.es>) correspondiente a la sesión 7.

A continuación, hay que crear una carpeta que se denomine "VehicleDigitalTwin". Posteriormente, dentro de esa carpeta se creará otra que se denomine "code" donde se almacenará el código en Python correspondiente al gemelo digital del simulador de vehículo autónomo.

Posteriormente, se tiene que crear un fichero en el directorio raíz del proyecto que se denomine VehicleDigitalTwin.py en el que se escribirá el código del gemelo digital del simulador del vehículo.

Se recomienda Copiar/Pegar el código de la solución del ejercicio 6 para comenzar la sesión 7, de esta manera, se podrá aprovechar la estructura del programa, la lógica de control del vehículo y la de gestión de su iluminación.

La estructura general que debe tener el programa a generar en esta sesión debe ser similar a esta:

```
if __name__ == '__main__':
    try:
        my_route = '{"Origin":"Toronto","Destination" : "Montreal"}'
        routes_loader(my_route)
        t2 = threading.Thread(target=environment_simulator,
                               daemon=True)
        t2.start()
        t3 = threading.Thread(target=vehicle_controller, daemon=True)
        t3.start()
        t4 = threading.Thread(target=led_controller, daemon=True)
        t4.start()
        t2.join()
        t3.join()
        t4.join()

    except Exception as e:
        print(e)
        vehicle_stop()
```


INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

En primer lugar, método que sería responsable de determinar la ruta para llegar desde el origen al destino (que en este caso siempre será el mismo: Toronto a Montreal).

Posteriormente, el método `routes_loader` se encargará de generar la ruta entre los puntos indicados y, a partir de la misma, generar la secuencia de comandos que tendrá que procesar el vehículo para completar la ruta requerida.

Posteriormente, se lanzarán tres Daemon Threads diferentes:

- **`environment_simulator`** se encargará de simular las mediciones que el sensor de ultrasonidos y el de luminosidad proporcionan. Este hilo sustituirá a los correspondientes del sensor LDR y del sensor de ultrasonidos. En las siguientes secciones se indicará cuál es el funcionamiento esperado de este hilo
- **`vehicle_controller`** se encargará de ejecutar los comandos asignados al vehículo. Este hilo será exactamente igual al procedente de la sesión 06, aunque se deberá eliminar el código correspondiente al accionamiento del servomotor y el motor de corriente continua.
- **`led_controller`** se encargará de determinar cuál es el comportamiento de las luces en función de las condiciones del contexto. Este hilo será exactamente igual al procedente de la sesión 06, aunque se deberá eliminar el código correspondiente al accionamiento de los LEDs RGB.

Con respecto al programa correspondiente a la sesión 06, también se debe eliminar el funcionamiento del botón porque se considera que cuando se lance el gemelo digital, implícitamente supone que se pretende encender el vehículo.

Se recomienda probar el gemelo digital del simulador del vehículo en el entorno de desarrollo local del estudiante antes de pasar a su containerización y despliegue. De esta manera, se optimizará la detección de los posibles fallos que se puedan introducir en los distintos elementos del código.

Eliminación del código correspondiente al botón de encendido del vehículo

En primer lugar, es necesario eliminar todo el código relacionado con la gestión de la información procedente del botón (`button_manager`).

Por tanto, se tendrán que eliminar todo el código correspondiente con las variables globales *should_run* y *executed_command*.

También se deben eliminar todos los imports correspondientes a la gestión de la interfaz GPIO.

Generador de Rutas

Este método gestiona una lista de rutas, cada ruta se compone por un origen y un destino. Cada vez que se asigna una ruta al vehículo se añade a la lista de rutas pendientes de completar del vehículo. Este funcionamiento permitirá que el código del gemelo digital pueda ser escalable para recibir varias rutas a completar en secuencia.

Controlador del Vehículo (*vehicle_controller*)

Este hilo se ejecutoriará hasta que finalice la ejecución del gemelo digital. Comprobará que el vehículo tiene rutas asignadas pendientes de completar.

1. En caso de que el vehículo tenga rutas pendientes de completar.

a) Se obtendrá su origen y su destino y se pedirá a un método que obtenga la secuencia de pasos a completar y, a partir de ellos, se generen los comandos que el vehículo tiene que ejecutar para completar la ruta.

```
def routes_manager(origin_address="Toronto",
destination_address="Montreal"):
    global currentRouteDetailedSteps
    global vehicleControlCommands

    # print("Asignando una ruta al vehículo")
    url =
    "https://maps.googleapis.com/maps/api/directions/json?origin=" +
    origin_address + "&destination=" + \
        destination_address + "&key=" + google_maps_api_key
    # print("URL: {}".format(url))
    payload = {}
    headers = {}
    response = requests.request("GET", url, headers=headers,
data=payload)
    current_route = response.text
    # print("La ruta es: {}".format(response.text))

    steps = response.json()["routes"][0]["legs"][0]["steps"]
    # print(steps)
    currentRouteDetailedSteps = get_detailed_steps(steps)
    getCommands(currentRouteDetailedSteps)
    # print("He acabado de asignar los comandos al vehículo")
```

Las variables *currentRouteDetailedSteps* y *vehicleControlCommands* se tienen que declarar al principio del código (son variables globales) porque van a actualizar en el ámbito de varios métodos *vehicle_controller*, *routes_manager*, etc.

La API Key de Google Maps se ha obtenido en el ámbito de la ejecución de los pasos de la sección 2 de este ejercicio.

La ruta se calcula ejecutando la correspondiente llamada a la API de Google. Los pasos que componen una ruta se obtiene del resultado de la misma en caso de que no haya habido ningún error con la petición realizada. Los pasos que proporciona la API de Google son los correspondiente a la primera ruta que devuelve como resultado que se obtiene de la siguiente manera:

```
steps = response.json()["routes"][0]["legs"][0]["steps"]
```

En este momento, es necesario obtener pasos más detallados para el vehículo, de tal manera que se puedan determinar los comandos de la misma manera que los procesados en el ámbito de la sesión 6.

Para ello, el método *get_detailed_steps* (paso 1.1) debe obtener los waypoints que debe atravesar el vehículo con el mayor nivel de precisión posible y el método *get_commands* (paso 1.2) será el responsable de generar los comandos a partir de la secuencia de pasos obtenido.

1.1. En el método *get_detailed_steps*, para cada uno de los pasos proporcionados por Google Maps, se debe:

- Determinar la velocidad en escala de 100

```
stepSpeed = (step["distance"]["value"] / 1000) /  
(step["duration"]["value"] / 3600)
```

- Determinar la distancia del paso.

```
stepDistance = step["distance"]["value"]
```

- Determinar el tiempo del paso.

```
stepTime = step["duration"]["value"]
```

- Determinar la maniobra que se tiene que ejecutar con el volante.

```
try:  
    stepManeuver = step["maneuver"]  
except:  
    stepManeuver = "Straight"
```

- Determinar los waypoints que se corresponden con ese paso. Para ello se realizará la llamada al método *decode_polyline*.

```
substeps = decode_polyline(step["polyline"]["points"])
```

Se recomienda copiar/pegar el código completo de este método, que es el siguiente:

```
def decode_polyline(polyline_str):
    '''Pass a Google Maps encoded polyline string; returns list of
    lat/lon pairs'''
    index, lat, lng = 0, 0, 0
    coordinates = []
    changes = {'latitude': 0, 'longitude': 0}

    # Coordinates have variable length when encoded, so just keep
    # track of whether we've hit the end of the string. In each
    # while loop iteration, a single coordinate is decoded.
    while index < len(polyline_str):
        # Gather lat/lon changes, store them in a dictionary to apply
        them later
        for unit in ['latitude', 'longitude']:
            shift, result = 0, 0

            while True:
                byte = ord(polyline_str[index]) - 63
                index += 1
                result |= (byte & 0x1f) << shift
                shift += 5
                if not byte >= 0x20:
                    break

            if (result & 1):
                changes[unit] = ~(result >> 1)
            else:
                changes[unit] = (result >> 1)

            lat += changes['latitude']
            lng += changes['longitude']

        coordinates.append((lat / 100000.0, lng / 100000.0))

    return coordinates
```

Para cada uno de los waypoints generados por este método, se tiene que indicar su origen (p1), su destino (p2), la distancia entre ambos, la duración en segundos, la velocidad (que es la del paso del que procede el waypoint) y la maniobra (que es la del paso del que procede el waypoint) y se tiene que añadir a la lista de pasos detallados que tiene que completar el simulador del vehículo.

```
for substep in substeps:
    if index < len(substeps):
        p1 = {"latitude": substeps[index][0], "longitude":
substeps[index][1]}
        p2 = {"latitude": substeps[index + 1][0], "longitude":
substeps[index + 1][1]}
        # print("Voy a calcular la distancia entre {} y
{}".format(p1, p2))
        points_distance = distance(p1, p2)
        if points_distance > 0.001:
            subStepDuration = points_distance / stepSpeed
            new_detailed_step = {"Origen": p1, "Destination": p2,
```

INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

```
"Speed": stepSpeed, "Time": subStepDuration,
                        "Distance": points_distance,
"Maneuver": stepManeuver}
    # print("Se ha añadido el paso:
{}".format(new_detailed_step))
    detailedSteps.append(new_detailed_step)
    # print("La ruta tiene {}".
pasos".format(len(detailedSteps)))
```

El método que calcula la distancia entre dos puntos es el siguiente:

```
def distance(p1, p2):
    p1Latitude = p1["latitude"]
    p1Longitude = p1["longitude"]
    p2Latitude = p2["latitude"]
    p2Longitude = p2["longitude"]
    # print("Calculando la distancia entre ({} ,{}) y
({} ,{})".format(p1["latitude"], p1["longitude"], p2["latitude"],
p2["longitude"]))
    earth_radius = {"km": 6371.0087714, "mile": 3959}
    result = earth_radius["km"] * acos(
        cos((radians(p1Latitude))) * cos(radians(p2Latitude)) *
cos(radians(p2Longitude) - radians(p1Longitude)) + sin(
        radians(p1Latitude)) * sin(radians(p2Latitude)))
    # print ("La distancia calculada es:{}".format(result))
    return result
```

- 1.2. En el método `get_commands`, para cada uno de los pasos incluidos en la variable global `currentRouteDetailedSteps`, se tiene que generar un comando en el formato procesable por el simulador del vehículo.

Para ello, se tiene que traducir la maniobra a un ángulo para el servomotor que controla la dirección, se tiene que asignar la velocidad y la duración correspondiente al waypoint a completar el vehículo.

```
def getCommands(currentRouteDetailedSteps):
    global vehicleControlCommands
    steeringAngle: float = 90.0

    vehicleControlCommands = []
    index = 0
    for detailedStep in currentRouteDetailedSteps:
        index += 1
        # print("Generando el comando {} para el paso
{}".format(index, detailedStep))
        if (detailedStep["Maneuver"].upper() == "STRAIGHT" or
detailedStep["Maneuver"].upper() == "RAMP_LEFT"
            or detailedStep["Maneuver"].upper() == "RAMP_RIGHT" or
detailedStep["Maneuver"].upper() == "MERGE"
            or detailedStep["Maneuver"].upper() ==
"MANEUVER_UNSPECIFIED"):
            steeringAngle = 90.0
        if detailedStep["Maneuver"].upper() == "TURN_LEFT":
            steeringAngle = 45.0
        if detailedStep["Maneuver"].upper() == "UTURN_LEFT":
            steeringAngle = 0.0
        if detailedStep["Maneuver"].upper() == "TURN_SHARP_LEFT":
```

INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

```
steeringAngle = 15.0
if detailedStep["Maneuver"].upper() == "TURN_SLIGHT_LEFT":
    steeringAngle = 60.0
if detailedStep["Maneuver"].upper() == "TURN_RIGHT":
    steeringAngle = 135.0
if detailedStep["Maneuver"].upper() == "UTURN_RIGHT":
    steeringAngle = 180.0
if detailedStep["Maneuver"].upper() == "TURN_SHARP_RIGHT":
    steeringAngle = 105.0
if detailedStep["Maneuver"].upper() == "TURN_SLIGHT_RIGHT":
    steeringAngle = 150.0
newCommand = {"SteeringAngle": steeringAngle, "Speed":
detailedStep["Speed"], "Time": detailedStep["Time"]}
vehicleControlCommands.append(newCommand)
```

b) Una vez que se han obtenido los comandos que tiene que ejecutar el simulador del vehículo para completar la ruta, se tiene que proceder a su ejecución.

- 1.3. Mientras que haya comandos pendientes de ejecutar, se tiene que proceder a la ejecución del primero. Un ejemplo del método para la ejecución de un comando es el siguiente:

```
def executeCommand(command, step):
    global current_steering
    global current_speed
    global current_position

    current_steering = command["SteeringAngle"]
    current_speed = command["Speed"]
    time.sleep(command["Time"])
    current_position = step["Destination"]
```

- 1.4. Una vez ejecutado un comando, se tiene que proceder a su eliminación de la lista de comandos pendientes de ejecutar.

```
del vehicleControlCommands[0]
```

c) Una vez que se han ejecutado todos los comandos correspondientes a una ruta, ésta se ha finalizado y se tiene que eliminar de las rutas pendientes de completar.

```
if len(routes) > 0:
    del routes[0]
```

2. En caso de que no haya rutas pendientes de completar, ordenará la parada del vehículo y esperará 10 segundos para comprobar si el vehículo ha recibido nuevas peticiones de ruta.

Un ejemplo del método para la parada del vehículo se muestra a continuación.

```
def vehicle_stop():
    global vehicleControlCommands
    global currentRouteDetailedSteps
    global current_steering
    global current_speed
    global current_leds
    global current_ldr
    global current_obstacle_distance

    vehicleControlCommands = []
    currentRouteDetailedSteps = []
    current_steering = 90.0
    current_speed = 0
    current_leds_str = ' [{"Color": "White", "Intensity": 0.0, "Blinking": "False"},' \
                        '{"Color": "White", "Intensity": 0.0, "Blinking": "False"},' \
                        '{"Color": "Red",' "Intensity": 0.0, "Blinking": "False"},' \
                        '{"Color": "Red", "Intensity": 0.0, "Blinking": "False"}]'
    current_leds = json.loads(current_leds_str)
    current_ldr = 0.0
    current_obstacle_distance = 0.0
```

Simulador de Entorno

En primer lugar, si se utiliza como punto de partida el código de la sesión 6, se tiene que eliminar todo el código relacionado con los métodos *ldr_manager* y *ultrasonic_manager*.

El simulador del entorno es un método que se ejecuta continuamente mientras que el gemelo digital del simulador del vehículo esté en ejecución.

- a) En primer lugar, se tiene que simular la luz siguiendo las siguiente reglas:
 - En caso de que el valor de distancia actual sea mayor que 0.0, se generará un número aleatorio entre -300 y 300. El valor obtenido se sumará al valor de luminosidad anterior.
 - En caso de que el valor de luminosidad actual sea menor o igual a 0.0, se generará un valor inicial de luminosidad que se encontrará entre 0.0 y 3000.0.
- b) Posteriormente, se tiene que simular la distancia existente hasta el obstáculo más cercano al vehículo. Las reglas que se tienen que considerar en este momento son:
 - En caso de que el valor de distancia actual al obstáculo sea mayor que 0.0, se generará un número aleatorio entre -5 y 5. El valor obtenido se sumará al valor anterior de distancia al obstáculo.
 - En caso de que el valor de distancia actual al obstáculo sea menor o igual a 0.0, se generará un valor inicial de luminosidad que se encontrará entre 0.0 y 50.0.

INTRODUCCIÓN A CONTENEDORES Y GEMELO DIGITAL

En este momento, el gemelo digital tendrá que determinar si el vehículo debe frenar (debido a la presencia peligrosa de un obstáculo). Se frenará siempre y cuando la distancia al obstáculo sea menor que 10.

Si la distancia es igual o superior a 10m y el vehículo no esté parado, éste dejará de frenar y la velocidad objetivo será la que indique el comando actual.

Controlador de la Iluminación

El hilo correspondiente a la gestión de los LEDs, será exactamente igual desarrollado en la sesión 6.

Sin embargo, será necesario eliminar todo el código correspondiente a la actuación sobre los LEDs RGB, en concreto, los relacionados con los métodos *execute_rear_right_led* y *execute_rear_left_led*.

Una vez finalizado este paso, no olvidar de publicar todo el código generado en el proyecto del Gitlab correspondiente a esta sesión.

Conceptos básicos para la creación de imágenes y lanzamiento de contenedores

Una vez que se ha probado que el código del gemelo digital funciona correctamente en el equipo de desarrollo, se procederá a la preparación para su despliegue con contenedores.

Creación de imágenes con Dockerfile

Para ello, en la carpeta "VehicleDigitalTwin" se creará el fichero Dockerfile.

El contenido de este fichero tendrá que permitir lo siguiente:

- Crear la imagen del contenedor a partir de la correspondiente a python:3.11.1
- Copiar el código que está en la carpeta `./code` a la carpeta `/etc/usr/src/app`
- Establecer esta carpeta como directorio de trabajo.
- Instalar los paquetes necesarios especificados en `requirements.txt`. Los paquetes necesarios son: *requests* y *math*, para ello ejecutar `pip install <paquetes necesarios>`
- Ejecutar el código incluido en el fichero `VehicleDigitalTwin.py`

Un resumen de las sentencias que se pueden incluir en el Dockerfile se muestra en la siguiente figura:

FROM <image_id>
base image to build this image from

RUN <command> *shell form*

RUN ["<executable>",
 "<param1>", *exec form*
 ...,
 "<paramN>"]
executes command to modify container's file system state

MAINTAINER <name>
provides information about image creator

LABEL <key>=<value>
adds searchable **metadata** to image

ARG <name>[=<default value>]
defines overridable **build-time parameter**:
docker build --build-arg <name>=<value> .

ENV <key>=<value>
defines **environment variable** that will be visible during
image build-time and container run-time

ADD <src> <dest>
copies files from <src> (**file, directory or URL**) and adds them
to container file system under <dst> path

COPY <src> <dest> similar to **ADD**, does not support URLs

VOLUME <dest>
defines **mount point** to be shared with host or other containers

EXPOSE <port>
informs Docker engine that container listens to **port** at run-time

WORKDIR <dest>
sets build-time and run-time working directory

USER <user>
defines run-time user to start container process

STOPSIGNAL <signal>
defines signal to use to notify container process to stop

ENTRYPOINT *shell form* or *exec form*
defines run-time **command prefix** that will be added to all
run commands executed by docker run

CMD *shell form* or *exec form*
defines run-time **command** to be executed by default when
docker run command is executed

Una vez finalizado este paso, no olvidar de publicar todo el código generado en el proyecto del Gitlab correspondiente a esta sesión.

Instalación de Docker Engine en la máquina de la GCP

Una vez que hemos completado el paso anterior, es necesario que nos conectemos por SSH a la máquina en la GCP que se ha creado en la sección 2 de este ejercicio.

Para conectarte a las VMs mediante SSH en el navegador desde la consola de Google Cloud, haz lo siguiente:

1. En la consola de Google Cloud, ve a la página **Instancias de VM**.
2. En la lista de instancias de máquinas virtuales, haz clic en **SSH** en la fila de la instancia a la que deseas conectarte.

<input type="checkbox"/>	Name ^	Zone	Recommendation	Internal IP	External IP	Connect
<input type="checkbox"/>	✓ instance-1	us-east1-b		10.142.0.2 (nic0)	35.231.114.114 ↗	SSH ▾ ⋮

Como Docker Engine no está en la distribución por defecto del Debian que se ha utilizado para la creación de la máquina virtual, es necesario instalarlo completando los siguientes pasos:

<https://docs.docker.com/engine/install/debian/>

Como último paso, ejecutar el siguiente comando que permitirá que no sea necesario utilizar sudo para la ejecución de docker engine:

```
sudo usermod -aG docker $USER
```

Posteriormente, cerrar el terminal SSH y volverlo a abrir para que los cambios tengan efecto.

Asimismo, para este y futuros ejercicios, será necesario instalar docker compose. Para ello, es necesario ejecutar los siguientes pasos (consultar la opción *Install the plugin manually*):

<https://docs.docker.com/compose/install/linux/>

Una vez completado este paso, es necesario clonar el proyecto en gitlab (<https://teaching.sel.inf.uc3m.es>) correspondiente a la sesión 7.

<h2>Git Cheat Sheet</h2> <h3>Setup</h3> <p>Set the name and email that will be attached to your commits and tags</p> <pre>\$ git config --global user.name "Danny Adams" \$ git config --global user.email "my-email@gmail.com"</pre> <h3>Start a Project</h3> <p>Create a local repo (omit <directory> to initialise the current directory as a git repo)</p> <pre>\$ git init <directory> Download a remote repo \$ git clone <url></pre> <h3>Make a Change</h3> <p>Add a file to staging</p> <pre>\$ git add <file></pre> <p>Stage all files</p> <pre>\$ git add .</pre> <p>Commit all staged files to git</p> <pre>\$ git commit -m "commit message"</pre> <p>Add all changes made to tracked files & commit</p> <pre>\$ git commit -am "commit message"</pre> <h3>Basic Concepts</h3> <p>main: default development branch origin: default upstream repo HEAD: current branch HEAD~: parent of HEAD HEAD~4: great-great grandparent of HEAD</p> <p><i>By @DoobleDanny</i></p>	<h3>Branches</h3> <p>List all local branches. Add -r flag to show all remote branches. -a flag for all branches.</p> <pre>\$ git branch</pre> <p>Create a new branch</p> <pre>\$ git branch <new-branch></pre> <p>Switch to a branch & update the working directory</p> <pre>\$ git checkout <branch></pre> <p>Create a new branch and switch to it</p> <pre>\$ git checkout -b <new-branch></pre> <p>Delete a merged branch</p> <pre>\$ git branch -d <branch></pre> <p>Delete a branch, whether merged or not</p> <pre>\$ git branch -D <branch></pre> <p>Add a tag to current commit (often used for new version releases)</p> <pre>\$ git tag <tag-name></pre> <h3>Merging</h3> <p>Merge branch a into branch b. Add --no-ff option for no-fast-forward merge</p> <pre>\$ git checkout b \$ git merge a</pre> <p>Merge & squash all commits into one new commit</p> <pre>\$ git merge --squash a</pre>	<h3>Rebasing</h3> <p>Rebase feature branch onto main (to incorporate new changes made to main). Prevents unnecessary merge commits into feature, keeping history clean</p> <pre>\$ git checkout feature \$ git rebase main</pre> <p>Interactively clean up a branches commits before rebasing onto main</p> <pre>\$ git rebase -i main</pre> <p>Interactively rebase the last 3 commits on current branch</p> <pre>\$ git rebase -i HEAD~3</pre> <h3>Undoing Things</h3> <p>Move (&/or rename) a file & stage move</p> <pre>\$ git mv <existing_path> <new_path></pre> <p>Remove a file from working directory & staging area, then stage the removal</p> <pre>\$ git rm <file></pre> <p>Remove from staging area only</p> <pre>\$ git rm --cached <file></pre> <p>View a previous commit (READ only)</p> <pre>\$ git checkout <commit_ID></pre> <p>Create a new commit, reverting the changes from a specified commit</p> <pre>\$ git revert <commit_ID></pre> <p>Go back to a previous commit & delete all commits ahead of it (revert is safer). Add --hard flag to also delete workspace changes (BE VERY CAREFUL)</p> <pre>\$ git reset <commit_ID></pre>	<h3>Review your Repo</h3> <p>List new or modified files not yet committed</p> <pre>\$ git status</pre> <p>List commit history, with respective IDs</p> <pre>\$ git log --oneline</pre> <p>Show changes to unstaged files. For changes to staged files, add --cached option</p> <pre>\$ git diff</pre> <p>Show changes between two commits</p> <pre>\$ git diff commit1_ID commit2_ID</pre> <h3>Stashing</h3> <p>Store modified & staged changes. To include untracked files, add -u flag. For untracked & ignored files, add -a flag.</p> <pre>\$ git stash</pre> <p>As above, but add a comment.</p> <pre>\$ git stash save "comment"</pre> <p>Partial stash. Stash just a single file, a collection of files, or individual changes from within files</p> <pre>\$ git stash -p</pre> <p>List all stashes</p> <pre>\$ git stash list</pre> <p>Re-apply the stash without deleting it</p> <pre>\$ git stash apply</pre> <p>Re-apply the stash at index 2, then delete it from the stash list. Omit stash@{n} to pop the most recent stash.</p> <pre>\$ git stash pop stash@{2}</pre> <p>Show the diff summary of stash 1. Pass the -p flag to see the full diff.</p> <pre>\$ git stash show stash@{1}</pre>	<p>Delete stash at index 1. Omit stash@{n} to delete last stash made</p> <pre>\$ git stash drop stash@{1}</pre> <p>Delete all stashes</p> <pre>\$ git stash clear</pre> <h3>Synchronizing</h3> <p>Add a remote repo</p> <pre>\$ git remote add <alias> <url></pre> <p>View all remote connections. Add -v flag to view urls.</p> <pre>\$ git remote</pre> <p>Remove a connection</p> <pre>\$ git remote remove <alias></pre> <p>Rename a connection</p> <pre>\$ git remote rename <old> <new></pre> <p>Fetch all branches from remote repo (no merge)</p> <pre>\$ git fetch <alias></pre> <p>Fetch a specific branch</p> <pre>\$ git fetch <alias> <branch></pre> <p>Fetch the remote repo's copy of the current branch, then merge</p> <pre>\$ git pull</pre> <p>Move (rebase) your local changes onto the top of new changes made to the remote repo (for clean, linear history)</p> <pre>\$ git pull --rebase <alias></pre> <p>Upload local content to remote repo</p> <pre>\$ git push <alias></pre> <p>Upload to a branch (can then pull request)</p> <pre>\$ git push <alias> <branch></pre>
--	--	--	---	--

Para ayudar a los usuarios no expertos de Git por línea de comandos, la imagen anterior proporciona un resumen de los principales comandos a considerar.

Lanzamiento del contenedor que representa al gemelo digital del simulador del vehículo

Una vez que se ha descargado el código de la sesión 7 en la máquina virtual de la GCP, se puede lanzar el código del gemelo digital del simulador del vehículo utilizando los siguientes comandos:

```
docker build -t vehicle_digital_twin_image ./VehicleDigitalTwin

docker run -d \

    --name vehicle_digital_twin \

    -e PYTHONUNBUFFERED=1 \

    -v $(pwd)/VehicleDigitalTwin/code:/etc/usr/src/app \

    vehicle_digital_twin_image
```

Criterios de evaluación y procedimiento de entrega



Criterios de Evaluación

- Gemelo Digital – Ejecución de rutas y comandos – 4 puntos
- Gemelo Digital – Simulación del entorno – 2 puntos
- Gemelo Digital – Gestión de LEDs – 1 punto
- Gemelo Digital – Ejecución con contenedores – 3 puntos (si no se obtienen estos, el ejercicio se calificará con 0 puntos).



Entrega de la solución del ejercicio guiado

La entrega se realizará de dos maneras:

- 1) **Código Fuente.** En el repositorio de código de la asignatura tendrá en el proyecto Sesión07 correspondiente al grupo de prácticas los ficheros de código con la organización en directorios y nomenclatura de los ficheros que se han indicado a lo largo de este guion.
- 2) **Video demostrativo.** En una tarea Kaltura Capture Video habilitada para este ejercicio se entregará un vídeo que demuestre el correcto funcionamiento de la solución elaborada.

Para este programa, se debe proporcionar una breve explicación del código generado (tanto del código de python como el Dockerfile generado), mostrar el lanzamiento de la ejecución de la solución en la GCP y la visualización de las trazas con los resultados.

La fecha límite para la entrega del ejercicio es 09/05/2024 a las 23:59 h.

De acuerdo con las normas de evaluación continua en esta asignatura, si un estudiante no envía la solución de un ejercicio antes de la fecha límite, el ejercicio será evaluado con 0 puntos.



Cada estudiante debe guardar una copia de la solución entregada hasta la publicación de las calificaciones finales de la asignatura.