



Instituto Tecnológico de Estudios Superiores Monterrey

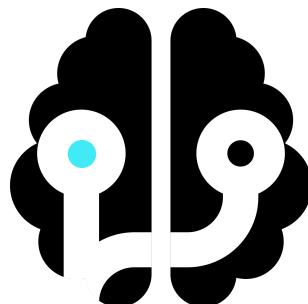
CAMPUS QUERÉTARO

Construcción de software y toma de decisiones

Ricardo Cortés Espinosa

Grupo 402

Manual de despliegue



PRESENTAN

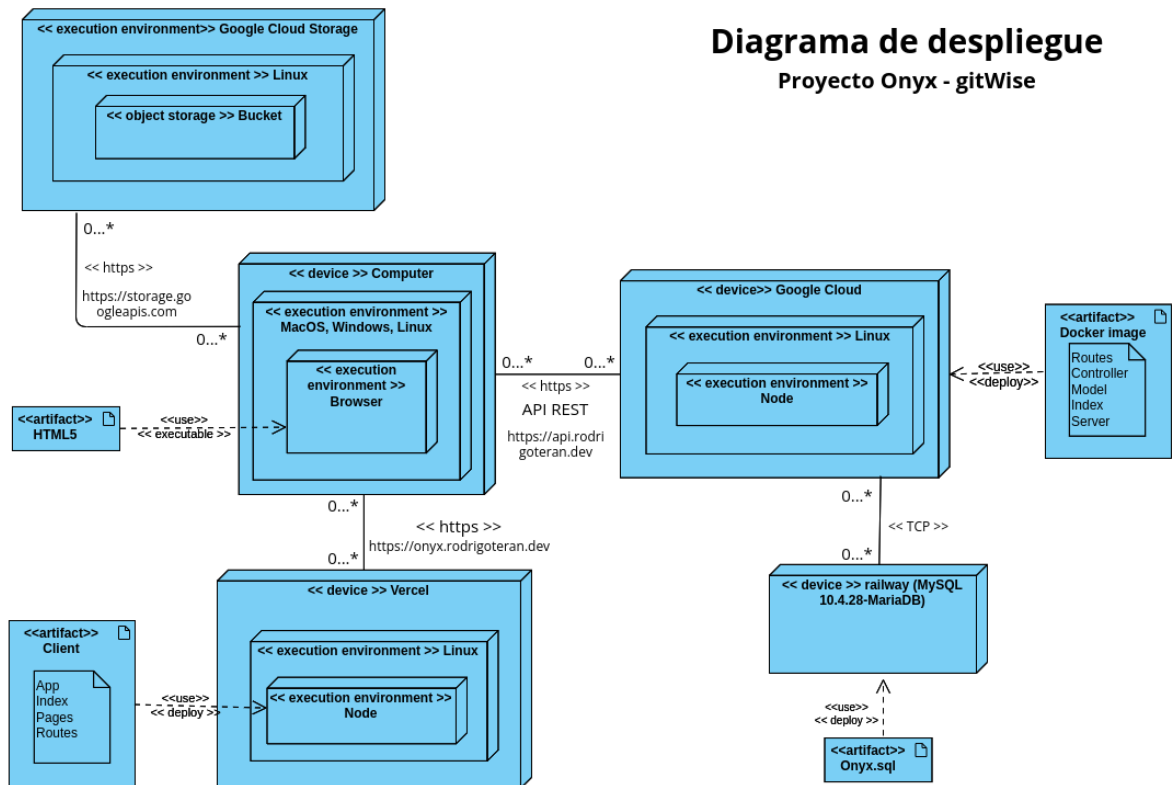
Daniel Felipe Hurtado Giraldo	A01707774
Jose Armando Rosas Balderas	A01704132
Sebastian Armando Flores Lemus	A01709229
Rodrigo Terán Hernández	A01704108
Ramona Nájera Fuentes	A01423596

Fecha:
05/05/2023

Índice

Índice	2
Diagrama de despliegue	3
Instalación	4
Precondiciones	4
Configurar las variables de entorno	4
Despliegue	7
Precondiciones para desplegar	7
Google cloud	7
Habilitar claves de API	8
OAuth	8
Google Cloud Storage	11
Google Container Registry	13
Google Cloud Run	14
Instalar localmente las herramientas para despliegue	14
Docker	14
gCloud	14
Desplegar el servidor	15
Comandos	16
Desplegar la página web	20
Desplegar la base de datos	21

Diagrama de despliegue



Instalación

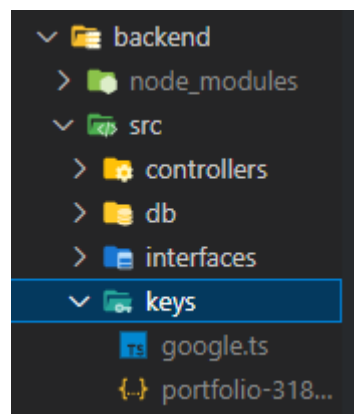
Precondiciones

Tener instalado en el equipo

- Node
- Git
- XAMPP o MAMP

Configurar las variables de entorno

1. Crea la carpeta 'keys' dentro del src en backend para configurar la API de Google Auth



```
const GOOGLE: {
  clientID: string;
  clientSecret: string;
} = {
  clientID: "espacio para la información",
  clientSecret: "espacio para la información"
};

export default GOOGLE;
```

```
// Añade un json con la información necesaria
{
  "type": "",
  "project_id": "",
  "private_key_id": "",
  "private_key": "",
  "client_email": "",
  "client_id": "",
  "auth_uri": "",
  "token_uri": "",
  "auth_provider_x509_cert_url": "",
  "client_x509_cert_url": ""
}
```

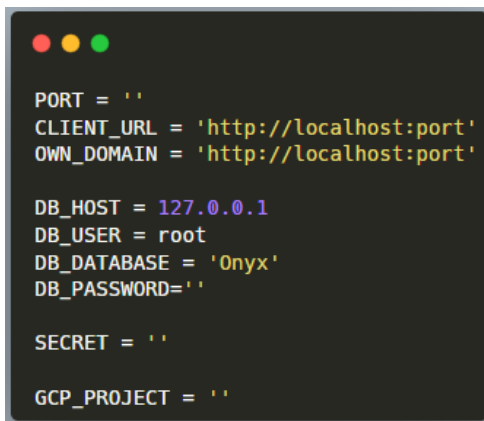
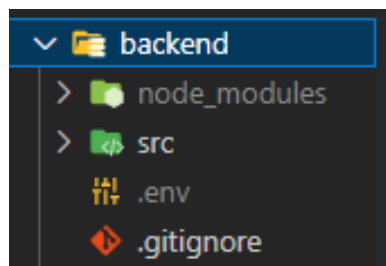
Para saber qué información debes escribir en este apartado y todas las condiciones necesarias ve al paso “OAuth” en la sección de “Despliegue”.

2. En la misma carpeta, agrega un documento de tipo JSON, el cual es el identificador de tu proyecto en Google Cloud Storage, para las imágenes. Para saber como obtener este archivo dirígete a la sección de “Google Cloud Storage” en la sección de “Despliegue”.

3. Archivos .env en la carpeta del backend

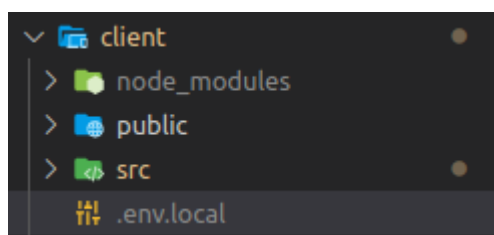
En la carpeta ‘backend’ crea un archivo llamado .env

A continuación ingresa el puerto en el que correrá el servidor, el URL desde el que llegan las peticiones del cliente, el URL del servidor, las información para acceder a la base de datos (host, user, db name y password), el secreto y el id del proyecto en Google Cloud.



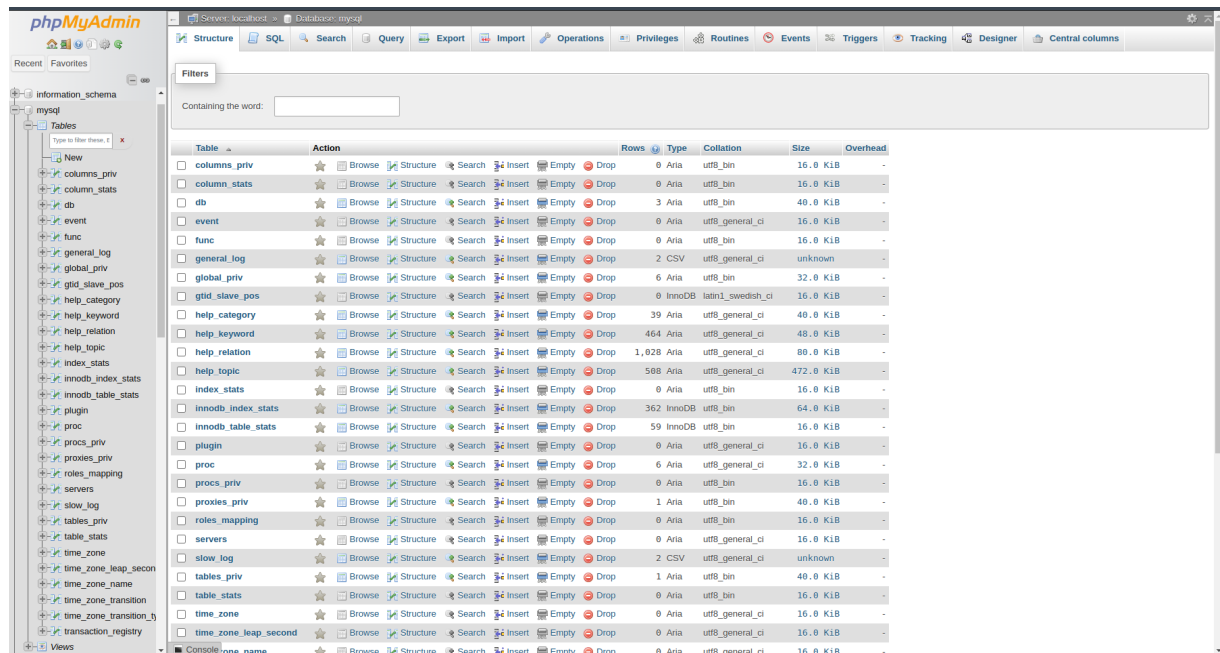
4. Archivos .env en la carpeta del cliente

En la carpeta “client” crea un archivo llamado .env.local. En este archivo introduce la siguiente línea de código, el cual es el URL del servidor, por lo tanto asegúrate de poner, después de “:” el número de puerto que asignaste para el backend.

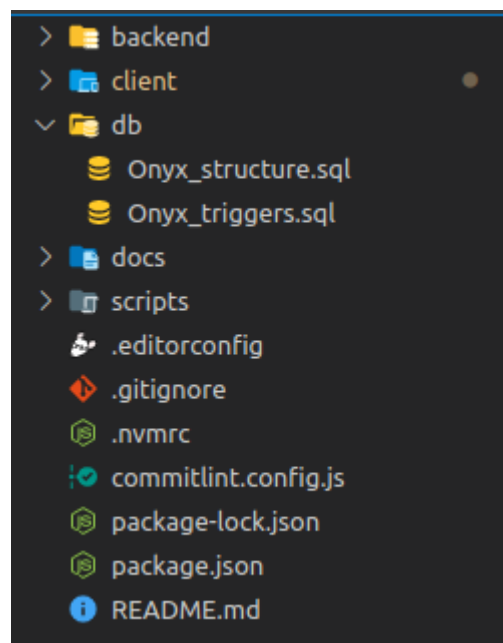


Base de datos

Para instalar la base de datos es necesario que actives MySQL y Apache por medio de XAMPP o MAMP. Ingresa a phpMyAdmin y entra a la sección de SQL.



Posteriormente dirígete a la carpeta de db

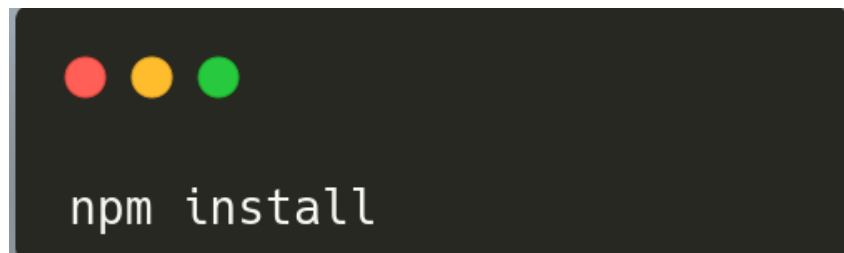


Una vez dentro copia el contenido de Onyx_structure.sql y pegalo en la sección de SQL. Después ejecuta el código con ctrl + ENTER.

Finalmente haz el mismo proceso pero con el contenido de Onyx_triggers.sql.

Instalación de paquetes

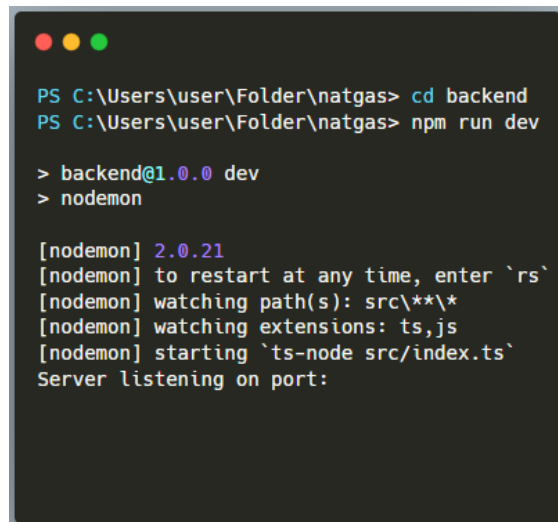
Antes de correr tu proyecto es necesario que tanto en la carpeta “client” como en la carpeta “backend” ejecutes el siguiente comando en la terminal. Esto para que instales todas las librerías de npm utilizadas en el proyecto.



```
npm install
```

Configurando el entorno de ejecución

En tu terminal escribe los comandos ‘cd backend’ y ‘npm run dev’ para iniciar el servidor

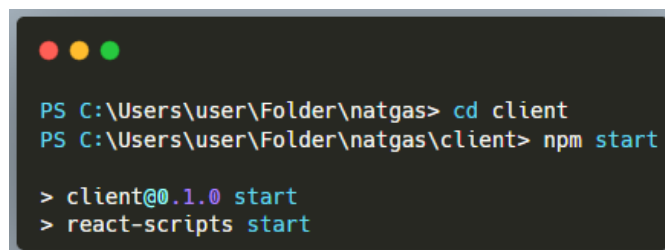


```
PS C:\Users\user\Folder\natgas> cd backend
PS C:\Users\user\Folder\natgas> npm run dev

> backend@1.0.0 dev
> nodemon

[nodemon] 2.0.21
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): src\**\*
[nodemon] watching extensions: ts,js
[nodemon] starting `ts-node src/index.ts`
Server listening on port:
```

En otra terminal escribe los comandos ‘cd client’ y ‘npm start’ para ejecutar la aplicación



```
PS C:\Users\user\Folder\natgas> cd client
PS C:\Users\user\Folder\natgas\client> npm start

> client@0.1.0 start
> react-scripts start
```

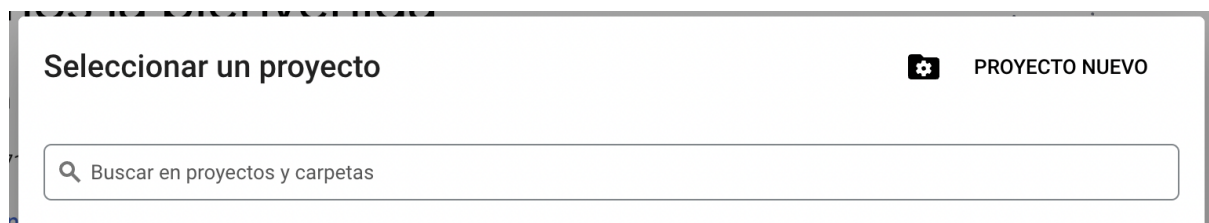
Despliegue

Precondiciones para desplegar

- Tener una cuenta de github
- Tener el repositorio en github

Google cloud

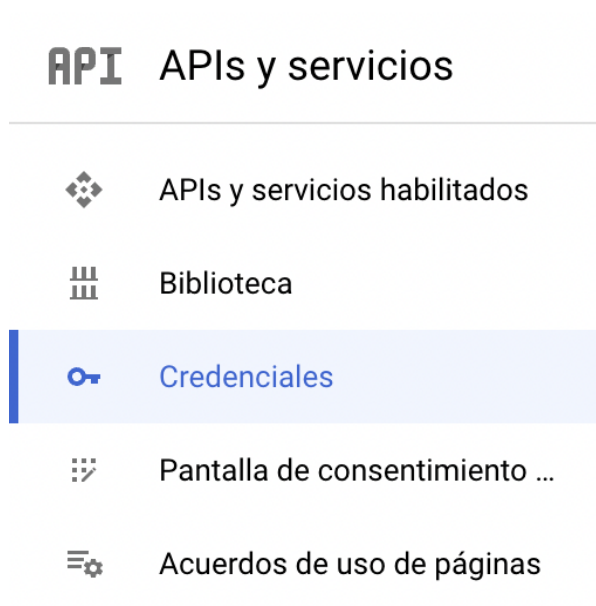
1. Hacerse una cuenta en Google Cloud si no se cuenta con una, <https://cloud.google.com/>
2. Crear un nuevo proyecto:



Habilitar claves de API

OAuth

1. Para habilitar la clave de API para hacer OAuth con Google debe de irse a credenciales:



2. Luego crear una clave de api

Credenciales

[+ CREAR CREDENCIALES](#)

3. Saldrán varias opciones. Debe de escoger la opción de ID de cliente de OAuth:

ID de cliente de OAuth

Solicita el consentimiento del usuario para que tu app pueda acceder a sus datos

4. Ahí debe de poner los orígenes autorizados para el uso de la aplicación. Básicamente el localhost y la url de su servidor (si es que ya la conoce). Luego puede venir a agregar más URI's.

Orígenes autorizados de JavaScript

Para usar con solicitudes de un navegador

URI 1 *

URI 2 *

[+ AGREGAR URI](#)

URI de redireccionamiento autorizados

Para usar con solicitudes de un servidor web

URI 1 *

URI 2 *

URI 3 *

[+ AGREGAR URI](#)

- En la parte de la derecha de esta página van a aparecer las claves para conectarse con este servicio.




Son dos:

- id del cliente
- secreto del cliente

ID de cliente	
Fecha de creación	10 de noviembre de 2021, 22:48:55 GMT-6

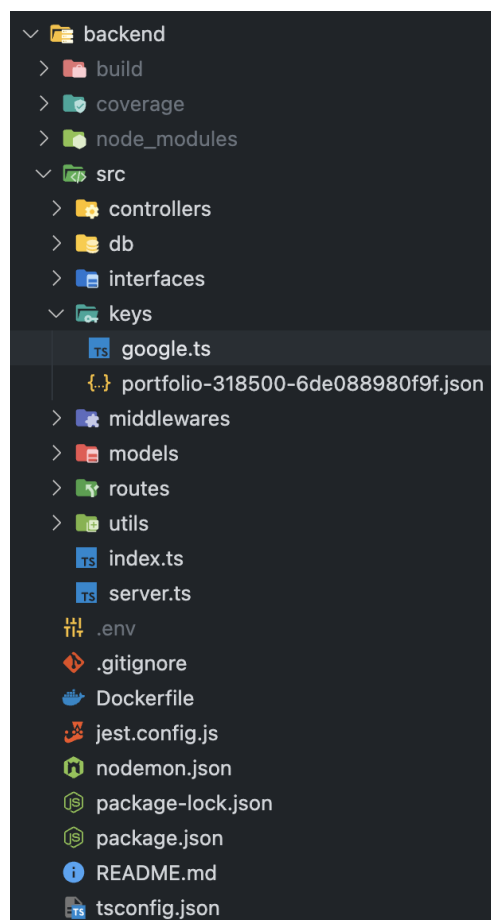
Secretos del cliente

Si estás en proceso de cambiar los secretos del cliente, puedes rotarlos de forma manual sin tiempo de inactividad. [Más información](#)

Secreto del cliente		 
Fecha de creación	10 de noviembre de 2021, 22:48:55 GMT-6	
Estado	 Habilitado	

Esas claves las debes de poner en un archivo llamado google.ts dentro de una carpeta llamada “keys” dentro de backend/src:

Ubicación de google.ts:



Dentro de google.ts:

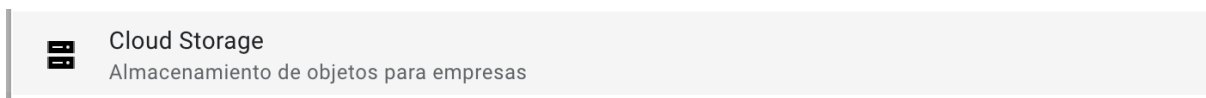
- Aquí sustituye los 3 puntos suspensivos por tus claves.

```
const GOOGLE: {
  clientId: string;
  clientSecret: string;
} = {
  clientId: "...",
  clientSecret: "..."
};

export default GOOGLE;
```

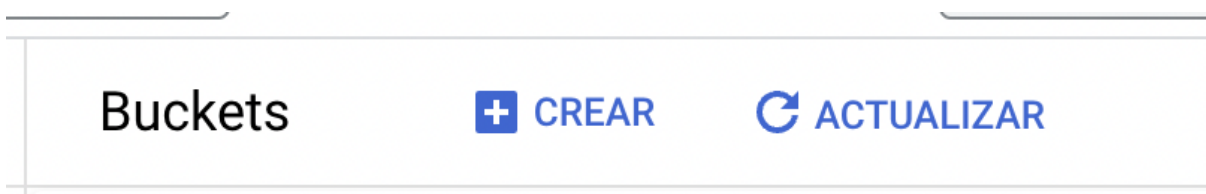
Google Cloud Storage

1. Para usar el Cloud Storage busca Cloud Storage en la barra de navegación dentro de tu proyecto:



2. Da en crear bucket:

Puedes nombrarlo como tu quieras



3. Para autenticar este bucket debes de darle click en agregar cuenta de servicio:



4. Da click en agregar clave:

← teran-onyx

DETALLES

PERMISOS

CLAVES

MÉTRICAS

REGISTROS

Claves



Las claves de cuenta de servicio podrían poner en riesgo la seguridad si se ven comprometidas en el [trabajo](#). Puedes obtener más información sobre cuál es la mejor manera de autenticar en el trabajo.

Agrega un nuevo par de claves o sube un certificado de clave pública de un par de claves existente.

Impide la creación de claves de cuentas de servicio con las [políticas de la organización](#).

[Más información para configurar políticas de la organización en cuentas de servicio](#)

AGREGAR CLAVE ▾

5. Da click en JSON

Crear clave privada para "teran-onyx"

Descarga un archivo que contiene la clave privada. Almacena el archivo en un lugar seguro, ya que no es posible recuperar la clave si se pierde.

Tipo de clave

☒ JSON

Recomendado

☐ P12

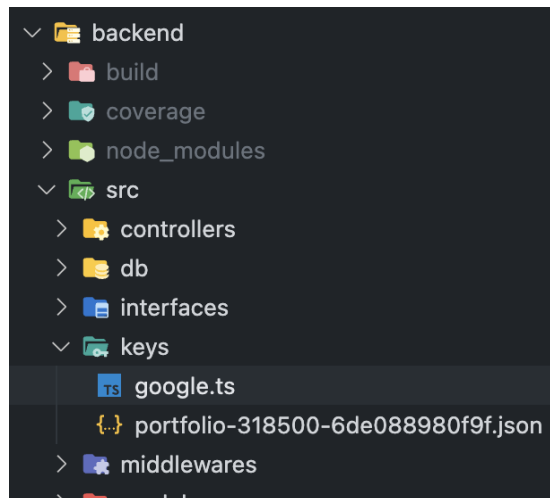
Para compatibilidad inversa con código en formato P12

CANCELAR

CREAR

6. Se va a descargar un archivo .json, guárdalo en esta ubicación:


Justo en la misma carpeta que google.ts. No le cambies el nombre, así déjalo.



Google Container Registry


1. Primero necesitas habilitar esta API:

Solo busca Google Container Registry y dale en habilitar

 Google Cloud

teran ▼

← Detalles del producto



Google Container Registry API






[Google Enterprise API](#)

Google Container Registry provides secure, private Docker image storage on Google Cloud Platform. ...

ADMINISTRAR

✓ API habilitada

2. Dale a crear un registry y te va a crear una carpeta con el nombre que elijas:

Repositorios		
teran		
 Filtro Ingresar el nombre o el valor de la propiedad		
Nombre 	Nombre de host 	Visibilidad 
 onyx-api	gcr.io	Privado

Google Cloud Run

1. Busca google cloud run en la barra de navegación dentro de tu proyecto.
Da click en crear servicio:



Instalar localmente las herramientas para despliegue

Docker

Debes instalar docker localmente en tu computadora.

Link: <https://www.docker.com/>

gCloud

Debes instalar la herramienta de google cloud, el google cloud cli.

Link: <https://cloud.google.com/sdk/docs/install?hl=es-419>

1. Una vez que instales esta herramienta debes de hacer login a tu cuenta de google




```
gcloud auth login
```

2. Luego debes de establecer tu proyecto. Para eso solo le pasas el id de tu proyecto en vez de "project-id"



```
gcloud config set project-id
```

3. Aquí debes de correr este comando una vez que hayas instalado docker. Esto hace que gcloud se configure en tu máquina para usar docker



```
gcloud auth configure-docker
```

Desplegar el servidor

Para esto primero necesitas configurar el Dockerfile que se encuentra en la carpeta de backend. Debes de cambiarlo por este código:

```
FROM node:10-alpine

ENV CLIENT_URL='https://url_del_cliente.com'
ENV OWN_DOMAIN='https://url_del_servidor.com'

ENV DB_HOST='host_de_base_de_datos'
ENV DB_USER='usuario_de_base_de_datos'
ENV DB_DATABASE='nombre_base_de_datos'
ENV DB_PASSWORD='contraseña_base_de_datos'

ENV SECRET='secreto para las sesiones'
ENV GCP_PROJECT='id_del_proyecto_de_google'

# create root application folder
WORKDIR /app

# copy configs to /app folder
COPY package*.json ./
COPY tsconfig.json ./

# copy source code to /app/src folder
COPY src /app/src

# check files list
RUN ls -a

RUN npm ci

RUN npm run build

EXPOSE 3456


CMD [ "node", "build/index.js" ]
```

Ahí debes de poner tus credenciales de la base de datos que escojas y el id de tu proyecto de google.

Comandos

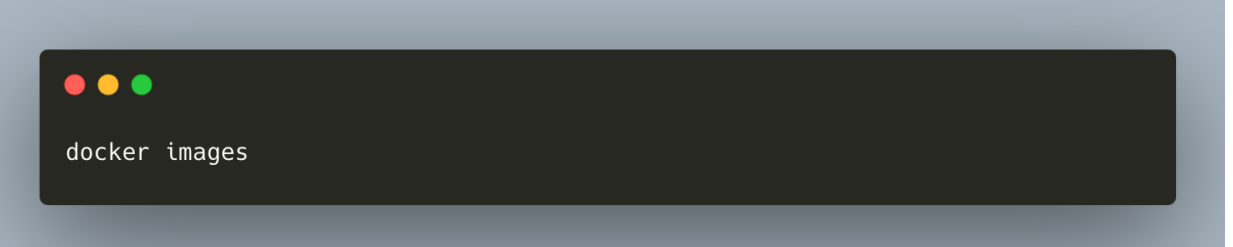
1. Para crear una imagen de docker corre este comando:

Vamos a ponerle por ejemplo a esta imagen el nombre de onyx-api-test, pero puede ser cualquier nombre



```
docker buildx build --platform linux/amd64 -t onyx-api-test .
```

2. Luego checa con este comando su id:




```
docker images
```

Aquí checas el nombre de la imagen que creaste y copias su IMAGE ID:

REPOSITORY	TAG	IMAGE ID	CREATED	SIZE
gcr.io/portfolio-318500/onyx-api	<none>	f98dd5288790	7 hours ago	262MB

3. Luego corres este comando:

Aquí debes de poner la IMAGE_ID de la imagen que acabas de crear, el id de tu proyecto, y el nombre de tu container registry:








```
docker tag IMAGE_ID gcr.io/id_de_tu_proyecto/nombre_de_tu_container_registry
```




- Y al final haces push con la misma url en donde hiciste el tag:

```
docker push gcr.io/id_de_tu_proyecto/nombre_de_tu_container_registry
```

- Aquí vas a ver tus imágenes cuando hagas push:

Filtro Ingresar el nombre o el valor de la propiedad					
<input type="checkbox"/>	Nombre	Etiquetas	Tamaño virtual ?	Fecha de creación	Subido ↓
<input type="checkbox"/>	 2af81f487d35	latest	86.8 MB	hace 2 horas	hace 2 horas
<input type="checkbox"/>	 20eefcad4e20		86.8 MB	hace 3 horas	hace 3 horas
<input type="checkbox"/>	 083a291fca8b		86.8 MB	hace 3 horas	hace 3 horas
<input type="checkbox"/>	 62f3d2bf86eb		86.8 MB	hace 6 horas	hace 6 horas
<input type="checkbox"/>	 096e22b53325		86.8 MB	hace 9 horas	hace 9 horas
<input type="checkbox"/>	 b0e8223df512		86.8 MB	hace 1 día	hace 1 día

- Ahora ve a Google Cloud Run y dale en editar una nueva versión:

 Cloud Run
  Detalles del servicio
  IMPLEMENTAR Y EDITAR UNA NUEVA REVISIÓN

- Selecciona la imagen que acabas de subir:

Selecciona una imagen de contenedor

ARTIFACT REGISTRY

CONTAINER REGISTRY

Proyecto: portfolio-318500

CAMBIAR

Contenedores de demostración

gcr.io/portfolio-318500/onyx-api

2af81f487d	latest	hace 2 horas
20eefcad4e		hace 3 horas
083a291fca		hace 3 horas
62f3d2bf86		hace 6 horas
096e22b533		hace 9 horas
b0e8223df5		hace 1 día
2078a3c8ee		hace 1 día
3bf1d506af		hace 1 día
01c2a7417e		hace 1 día

SELECCIONAR

CANCELAR




8. Y le das en implementar:




- ☒ **Aplicar esta revisión inmediatamente**
El 100% del tráfico se migrará a esta revisión, por lo que se anularán todas las divisiones del tráfico existentes, si las hubiera.

IMPLEMENTAR

CANCELAR

9. En donde dice URL deberías de ser capaz de ver el servidor:

 Cloud Run
  Detalles del servicio
  IMPLEMENTAR Y EDITAR UNA NUEVA REVISIÓN

 **onyx-api**
 Región: us-central1
 URL: <https://onyx-api-v3o5tw7kva-uc.a.run.app>

 (y 1 más)

10. Si visitas la URL te debería salir:
Eso significa que todo salió bien

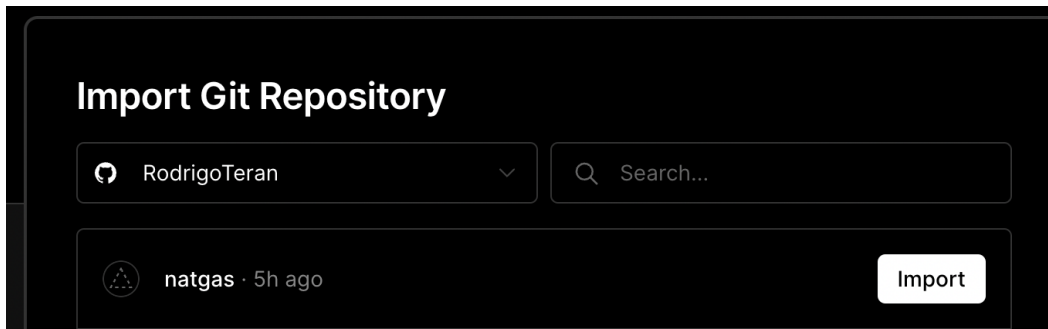
```
{"msg":"La ruta no existe","auth":false,"data":{}}
```

Desplegar la página web

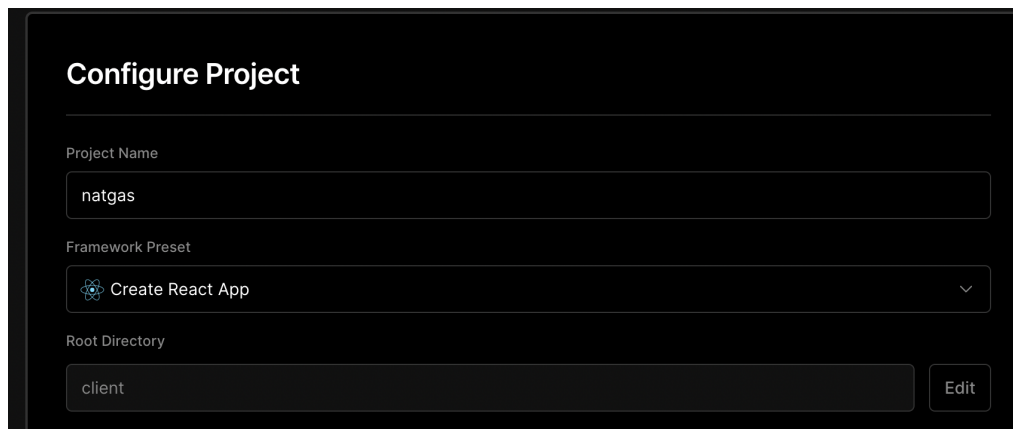
1. Para desplegar la página web tienes que hacerte una cuenta en Vercel
Link: <https://vercel.com/>

Te va a pedir que enlases tu cuenta de github con la de vercel.

2. Cuando enlases tu cuenta dale en importar proyecto, y ahí debes de poner el repositorio en donde tienes el código:



3. Ahí debes de cambiar el directorio raíz al client,



4. Y en donde dice variables de entorno debes de poner la clave valor:



5. Dale en crear

Desplegar la base de datos

1. Para desplegar la base de datos debes de crear una cuenta en Railway
Link: <https://railway.app/>

2. Ahí puedes crear un proyecto de MySQL
Ahí subes el código de SQL y lo ejecutas

3. En el apartado de connect te saldrán estas claves:

Esas las pones en tu Dockerfile:

Available Variables

This plugin exposes the following variables.

MYSQL_URL	*****
-----------	-------

MYSQLDATABASE	*****
---------------	-------

MYSQLHOST	*****
-----------	-------

MYSQLPASSWORD	*****
---------------	-------

MYSQLPORT	*****
-----------	-------

MYSQLUSER	*****
-----------	-------