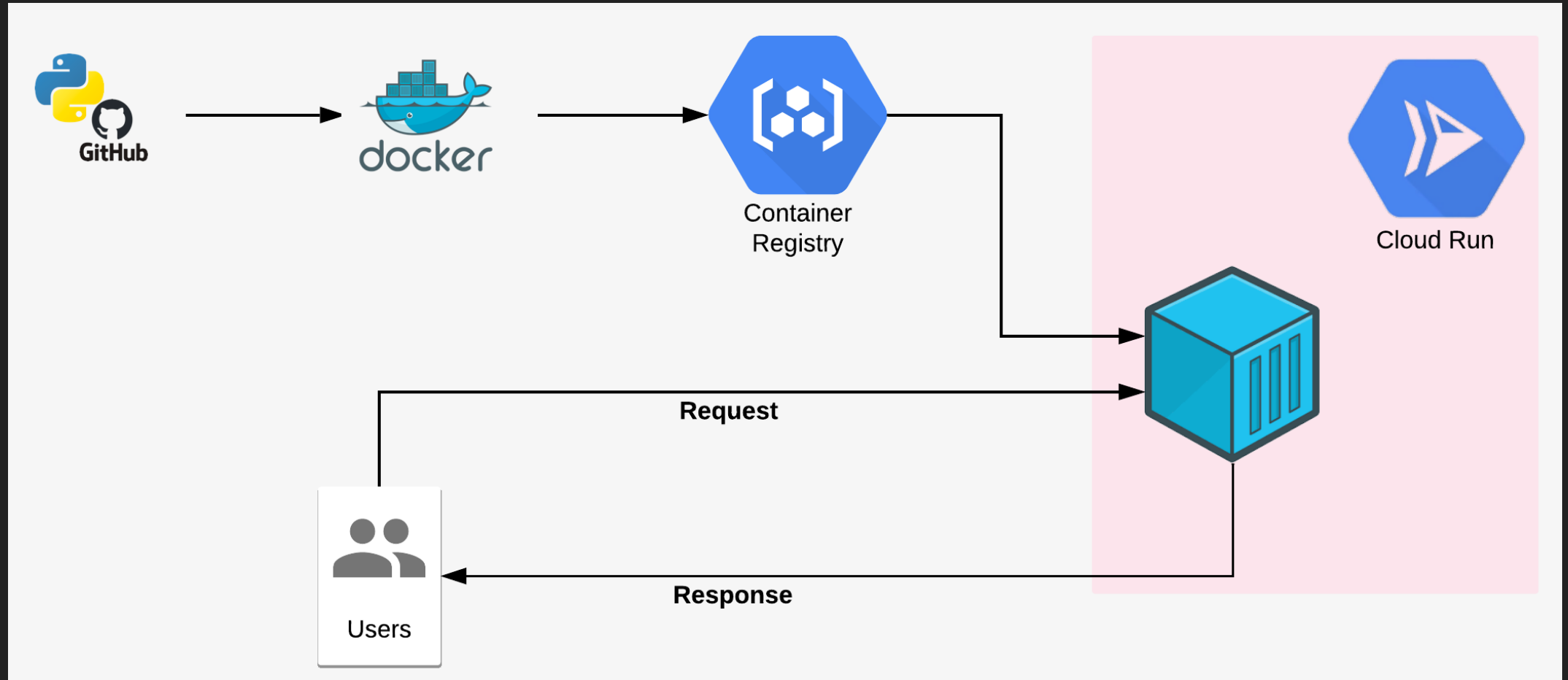


# Introducción a Google Cloud Run.

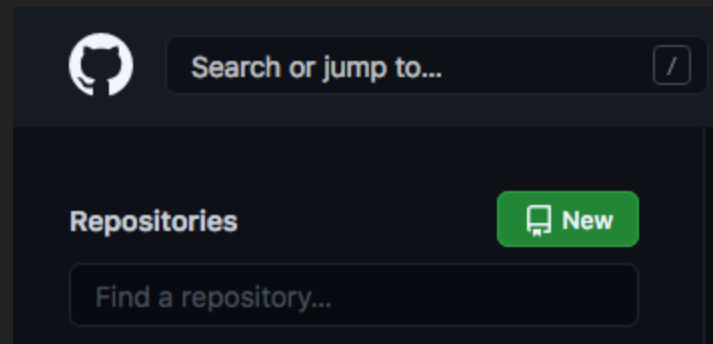
- Para desplegar aplicaciones dockerizadas tenemos varias opciones.
- Las más simples son:
  - Google Cloud Run en Google Cloud
  - Azure Container Instances
  - AWS Fargate

- Vamos a desplegar el API que acabamos de programar.
- Usaremos el código de la carpeta ejemplo\_docker\_3.

- El proceso que seguiremos se resume en:



- Crearemos un nuevo repositorio en github.



## Create a new repository

A repository contains all project files, including the revision history. Already have a project repository elsewhere? [Import a repository](#).

Owner \*



fernandodelacalle ▾

Repository name \*

/ my\_first\_api



Great repository names are short and memorable. Need inspiration? How about **stunning-octo-computing-machine**?

Description (optional)



**Public**

Anyone on the internet can see this repository. You choose who can commit.



**Private**

You choose who can see and commit to this repository.

### Initialize this repository with:

Skip this step if you're importing an existing repository.



**Add a README file**

This is where you can write a long description for your project. [Learn more](#).




**Add .gitignore**

Choose which files not to track from a list of templates. [Learn more](#).



**Choose a license**


A license tells others what they can and can't do with your code. [Learn more](#).

This will set  `main` as the default branch. Change the default name in your [settings](#).



Create repository


- En él pondremos todo el código de nuestra aplicación.
- Clona el repo desde gitkraken.
- Añade los ficheros de la carpeta ejemplo\_docker\_3..


- Haz un commit de los cambios y súbelo al repositorio de origen.

 **fernandodelacalle** / **my\_first\_api**


[Code](#) [Issues](#) [Pull requests](#) [Actions](#) [Projects](#) [Wiki](#) [Security](#) [Insights](#) [Settings](#)


 **main** 

 **1 branch**


 **0 tags**





[Go to file](#)


[Add file](#) 

[Code](#) 

**Fernando de la Calle Silos api example**

0e7df01 1 minute ago  2 commits


 <b>src</b>	api example	1 minute ago
 <b>Dockerfile</b>	api example	1 minute ago
 <b>README.md</b>	Initial commit	5 minutes ago
 <b>requirements.txt</b>	api example	1 minute ago

**README.md** 

# my\_first\_api



- Creamos un proyecto en google cloud

 Google Cloud Platform

Search products and resources

## New Project

Project name \*

My Project 27799

?

Project ID: ninth-terminal-310708. It cannot be changed later. [EDIT](#)


Billing account \*

My Billing Account

▼

Any charges for this project will be billed to the account you select here.

Location \*

 No organization

[BROWSE](#)

Parent organization or folder

CREATE

CANCEL

- Abrimos un cloud shell
- Comprobamos que el shell está en proyecto que acabamos de crear.

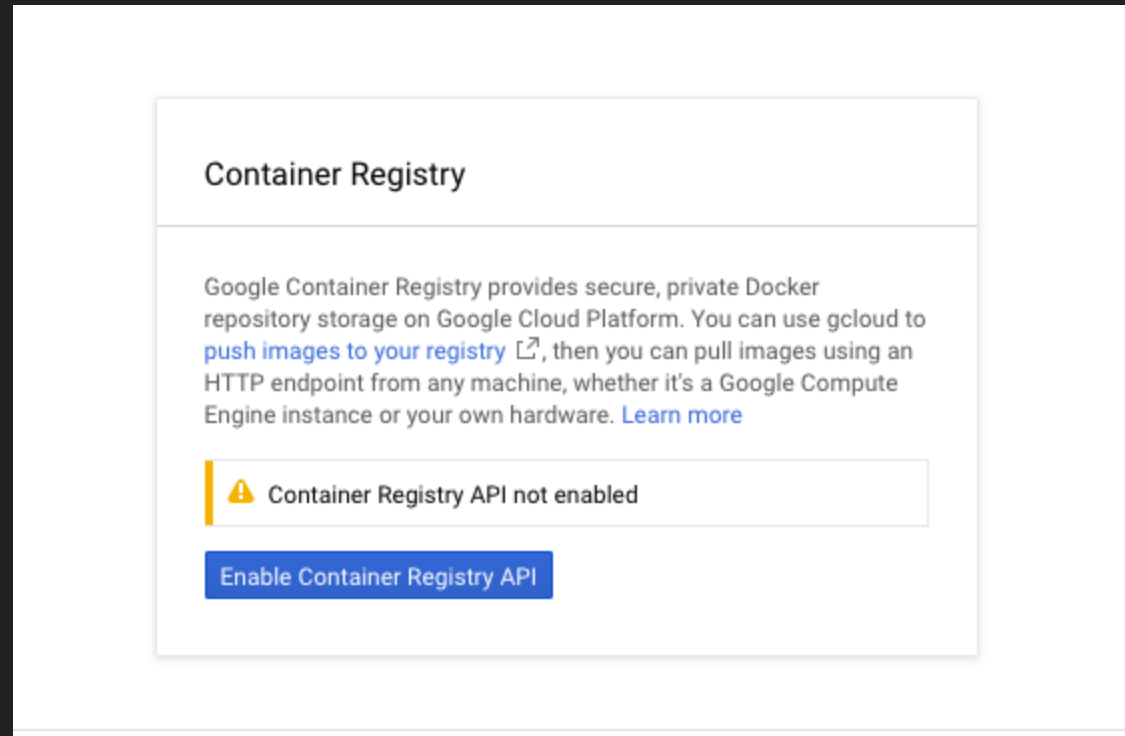
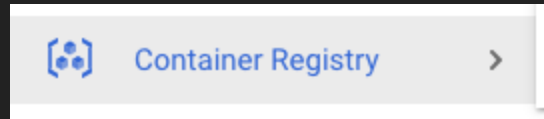
---

```
Welcome to Cloud Shell! Type "help" to get started.  
Your Cloud Platform project in this session is set to my-first-api-310708.  
Use "gcloud config set project [PROJECT_ID]" to change to a different project.  
fernando_decalle@cloudshell:~ (my-first-api-310708)$  
fernando_decalle@cloudshell:~ (my-first-api-310708)$  
fernando_decalle@cloudshell:~ (my-first-api-310708)$
```

- Clonamos el repo que acabamos de crear:

```
git clone https://github.com/----/my_first_api.git  
cd my_first_api
```

- Tenemos que construir la imagen y subirla al registry de google.
- Para ello primero activamos el registry en nuestro proyecto



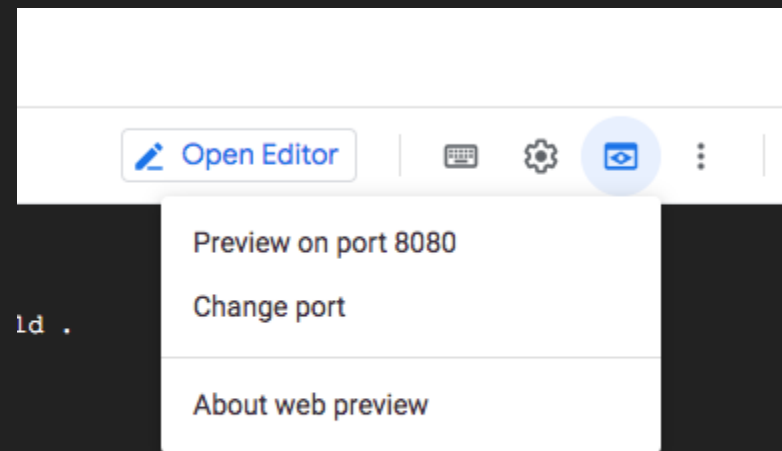
- El nombre de nuestra imagen será: [gcr.io/PROJECT-ID/my-first-api](#)
- Construimos la imagen con:

```
docker build -t gcr.io/PROJECT-ID/my-first-api .
```

- Podemos realizar una prueba ejecutando un contenedor con:

```
docker run -p 8080:8080 gcr.io/PROJECT-ID/my-first-api
```

- Podemos ver si funciona usando web preview:




- Subimos la imagen al registry con:

```
docker push gcr.io/PROJECT-ID/my-first-api
```


- Tendrás que ver en tu registry algo como esto:

my-first-api



Filter All hostnames ?

Name ^	Hostname	Visibility ?
 my-first-api	gcr.io	Private

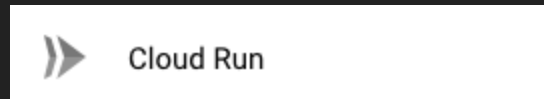
my-first-api

gcr.io / my-first-api-310708 / my-first-api 

Filter by name or tag Columns ?


<input type="checkbox"/> Name	Tags	Created	Uploaded ^
<input type="checkbox"/>  d2089c6811bd	latest	3 minutes ago	Just now 

- Ahora vamos a desplegar la imagen en cloud run.
- Para ello nos dirigimos en la interfaz a:














- Crearemos un nuevo servicio:

 Cloud Run | Services | [+ CREATE SERVICE](#) | [MANAGE CUSTOM DOMAINS](#) | [COPY](#) | [DELETE](#)

Each Cloud Run service has a unique endpoint and autoscales deployed containers. [Learn more](#)

 Clicking "Create service" will enable the Cloud Run API.

 **Filter** Filter services  

<input type="checkbox"/>	 Name 	Req/sec 	Region	Authentication 	Ingress 	Last deployed	Deployed by
No rows to display							

- Configuramos de la siguiente manera:

**1 Service settings**

A service exposes a unique endpoint and automatically scales the underlying infrastructure to handle incoming requests. Deployment platform and service name cannot be changed.

**Service name \***

**Deployment platform ?**

☒ Cloud Run (fully managed)

**Region \***

[How to pick a region?](#)

☐ Cloud Run for Anthos

**NEXT**

- Seleccionamos la imagen que acabamos de crear:

**2 Configure the service's first revision**

A service can have multiple revisions. The configurations of each revision are immutable.


☒ Deploy one revision from an existing container image

Container image URL \*

SELECT

E.g. `us-docker.pkg.dev/cloudrun/container/hello`  
Should listen for HTTP requests on `$PORT` and not rely on local state. [How to build a container?](#)

☐ Continuously deploy new revisions from a source repository

**Advanced settings** 

NEXT

CONTAINER REGISTRY

ARTIFACT REGISTRY

Project: my-first-api-310708 [CHANGE](#)

▶ Demo containers

▼ gcr.io/my-first-api-310708/my-first-api

d2089c6811

latest

2 minutes ago

SELECT

CANCEL

- Seleccionamos por último las siguientes opciones:

### Configure how this service is triggered

A service can be invoked directly or via events. Click "Add Eventarc Trigger" to create a new event-based trigger. [Learn more](#)

**Ingress ?**

☒ Allow all traffic

☐ Allow internal traffic and traffic from Cloud Load Balancing

☐ Allow internal traffic only

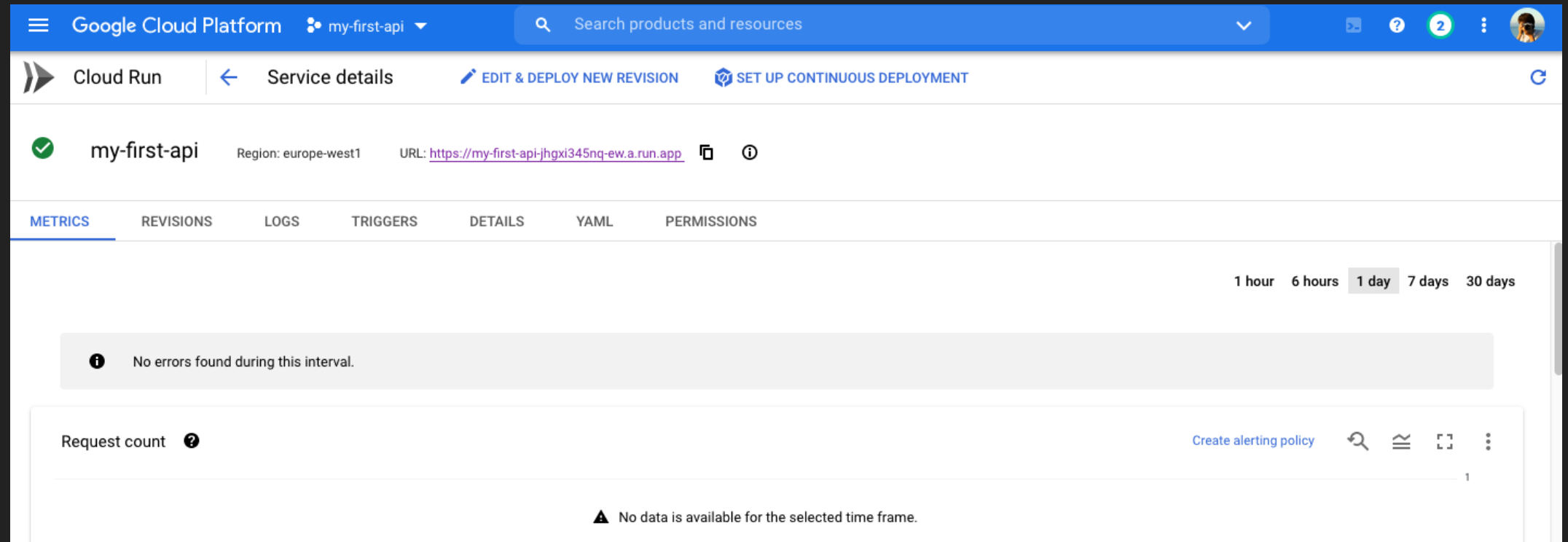
**Authentication \* ?**

☒ Allow unauthenticated invocations  
Check this if you are creating a public API or website.

☐ Require authentication  
Manage authorized users with Cloud IAM.

[+ ADD EVENTARC TRIGGER](#)

- Una vez desplegado veremos:



- Podemos ver nuestra API en la web url superior.

- Este proceso lo podemos realizar desde la linea de comandos con:

```
gcloud run deploy my-api --image gcr.io/PROJECT-ID/my-first-api:0.0.2 \
--allow-unauthenticated \
--platform managed \
--region europe-west1 \
--memory 2G \
```

- Si realizamos algún cambio es importante completar los tags para poder volver a versiones anteriores.
- Realicemos algún cambio en el código en nuestro ordenador.
- Subimos los cambios usando git kraken.



- Desde el cloud shell:
  - Traemos los nuevos cambios:

```
git pull origin main
```

- Construimos la imagen:

```
docker build -t gcr.io/my-first-api-310708/my-first-api:0.0.2 .
```

- Subimos la imagen al registry:

```
docker push gcr.io/my-first-api-310708/my-first-api:0.0.2
```

- Desplegamos desde línea de comandos:

```
gcloud run deploy my-dash --image gcr.io/my-first-api-310708/my-first-api:0.0.2 \
    --allow-unauthenticated \
    --platform managed \
    --region europe-west1 \
    --memory 2G
```

- Es interesante un producto llamado Cloud Scheduler que nos permite realizar invocaciones cron a nuestra API. Por ejemplo para la ejecución de algoritmos o ETLs.

# Ejercicio

- Despliega tu aplicación dash.
- Para ello realiza los siguientes pasos:
  - Usa el repo con el código de la app.
  - Desde la consola de google cloud:
    - Clona el repo.
    - Construye una nueva imagen y subela al registry.
    - Genera el nuevo servicio en cloud run.
  - Comprueba que puedes acceder a la página web.