How to create .NET8 WebAPI and store the Docker image in my Google Cloud Artifact Registry

1. Create a Google Cloud Artifact Registry repo

To initialize gcloud run the command:

```
gcloud init
```

This will guide you through the initialization process, where you'll log in to your Google account, set your default project, and choose a default compute zone/region if necessary.

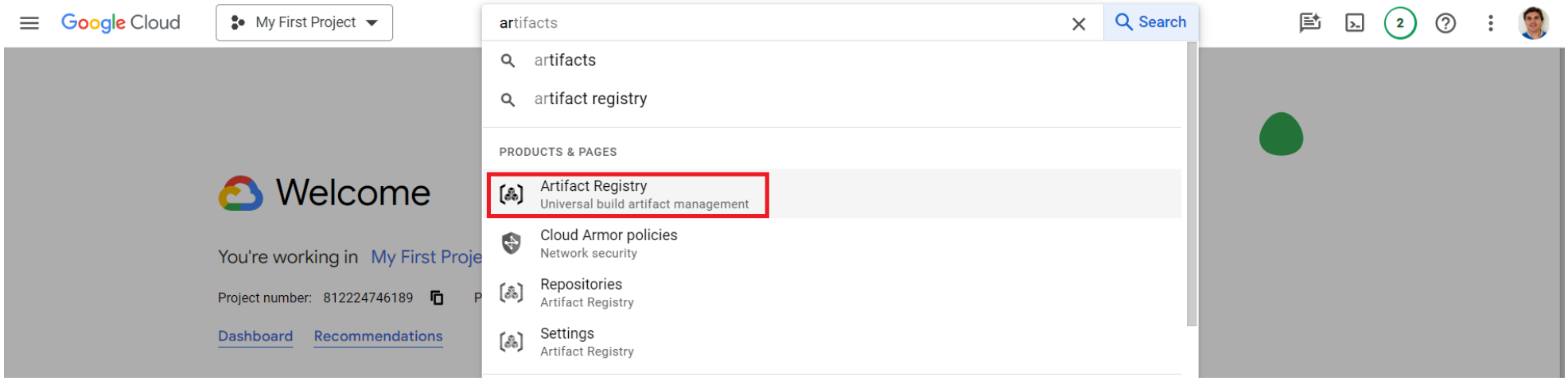
To update gcloud after initialization, you can update the Google Cloud SDK to the latest version by running the command:

```
gcloud components update
```

This command will check for the latest version and update the SDK components as needed.

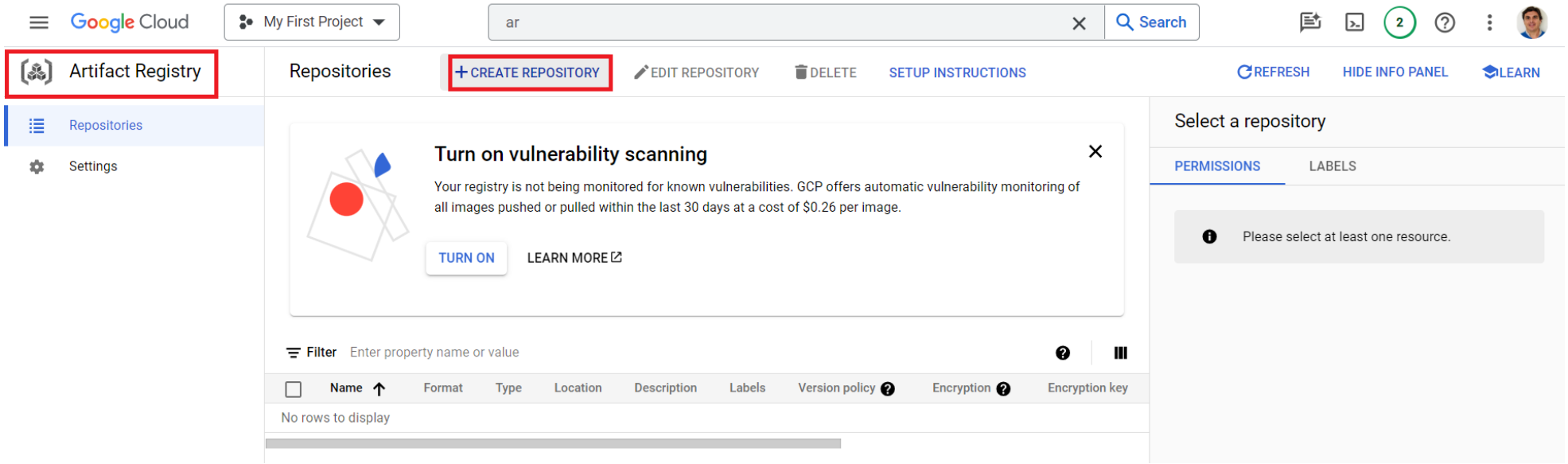
Now we can create a new Artifact Registry repo in Coogle Cloud

We log in to Google Cloud console and we search for **Artifact Registry**

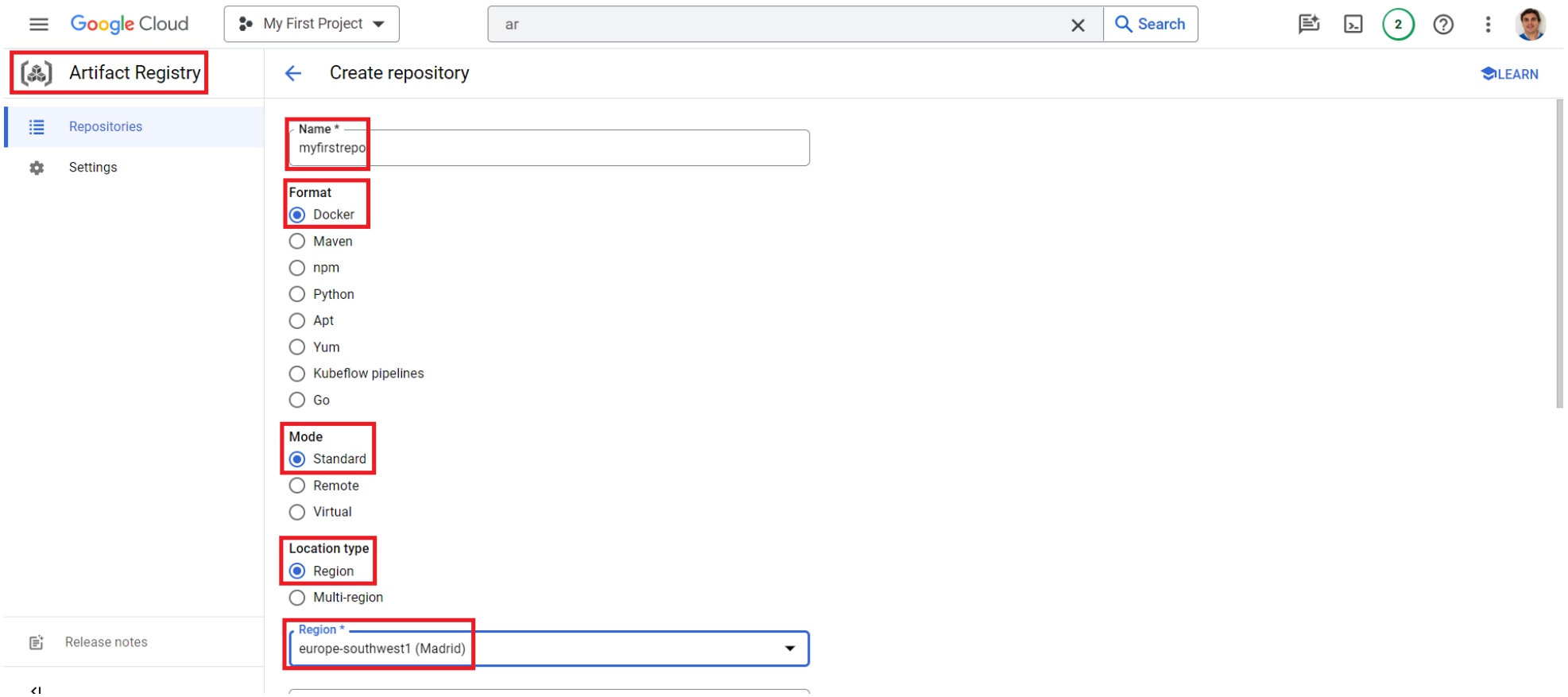


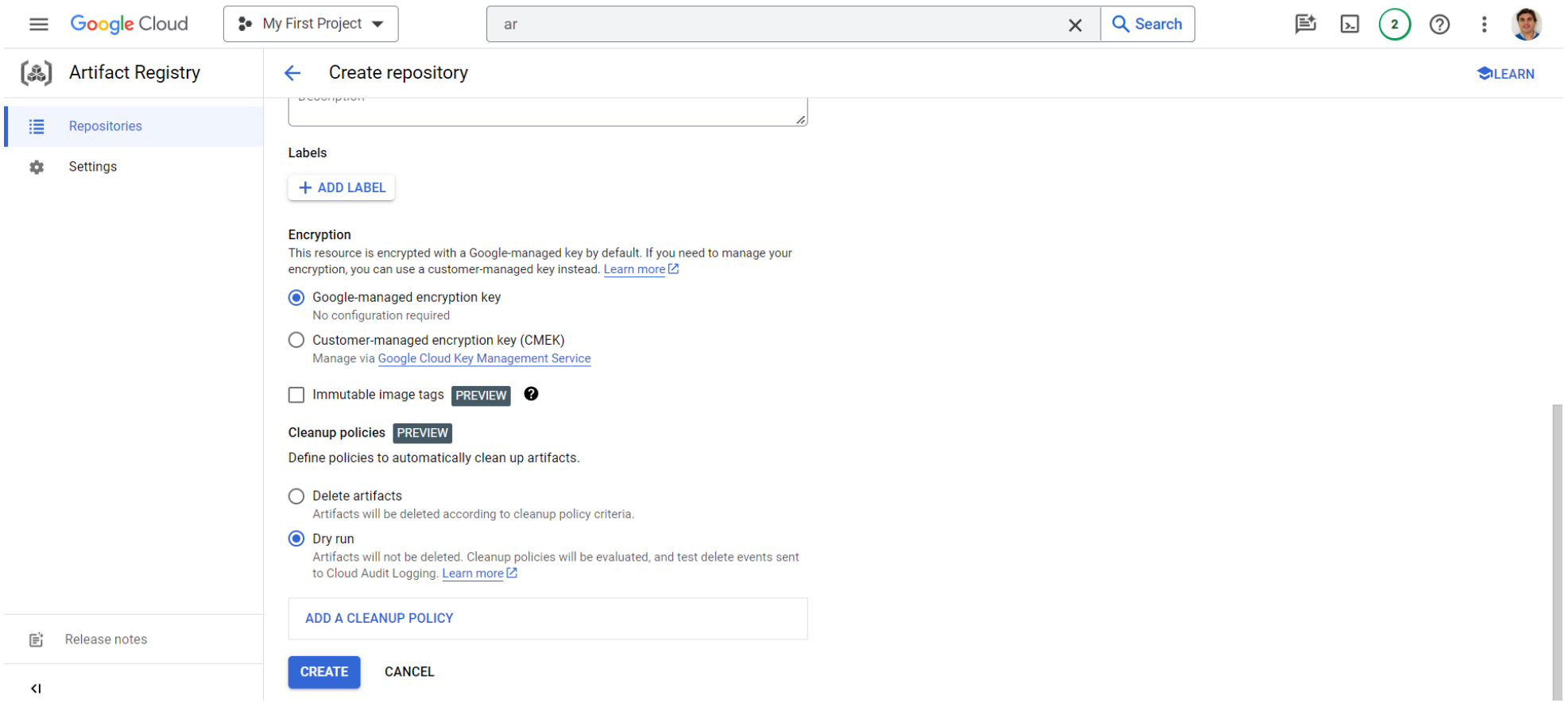
We enable the Artifact Registry API

We create a new repo

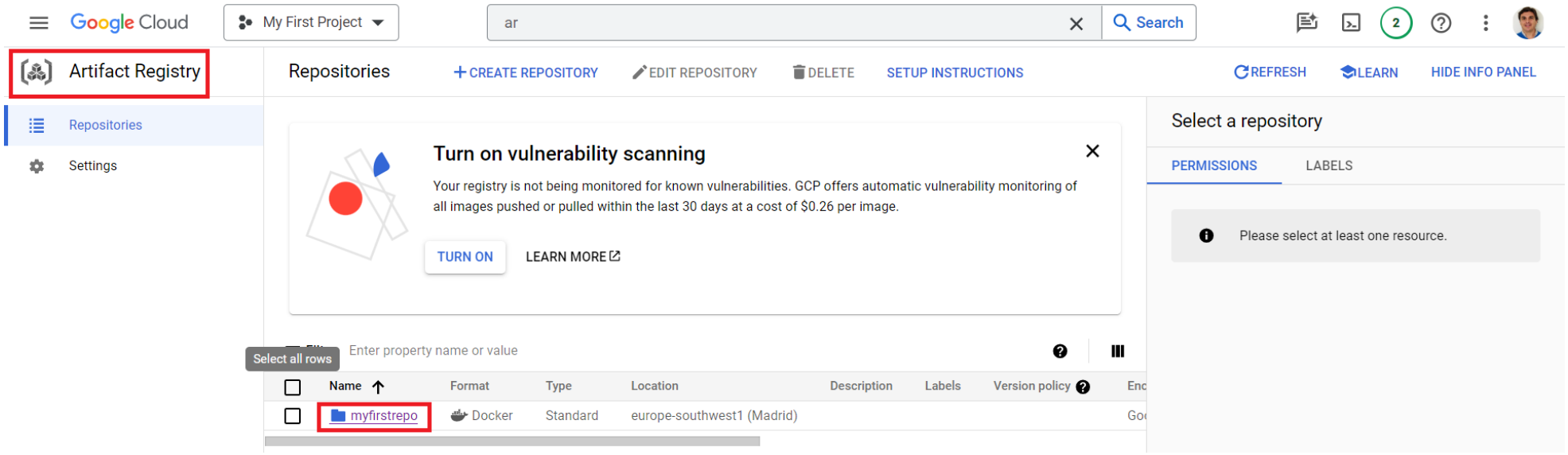


We input the new repo data

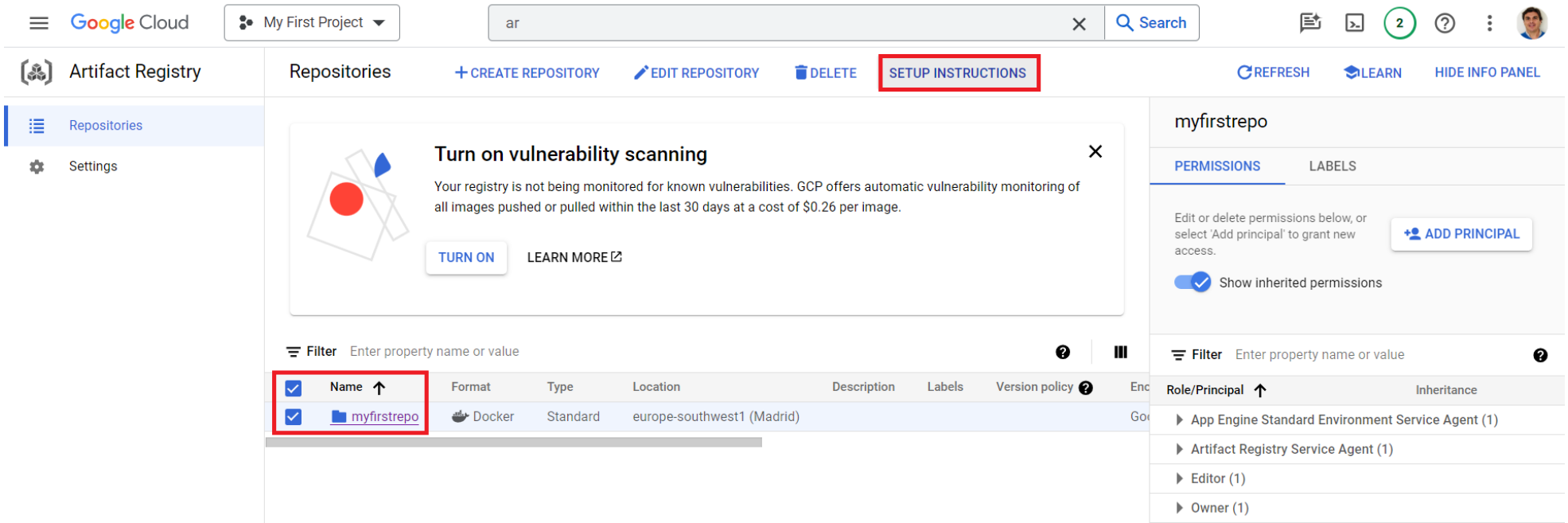

Google Cloud
My First Project
ar
Search
Artifact Registry
Repositories
Settings
Release notes
Create repository
LEARN
Name *
myfirstrepo
Format
Docker
Maven
npm
Python
Apt
Yum
Kubeflow pipelines
Go
Mode
Standard
Remote
Virtual
Location type
Region
Multi-region
Region *
europe-southwest1 (Madrid)



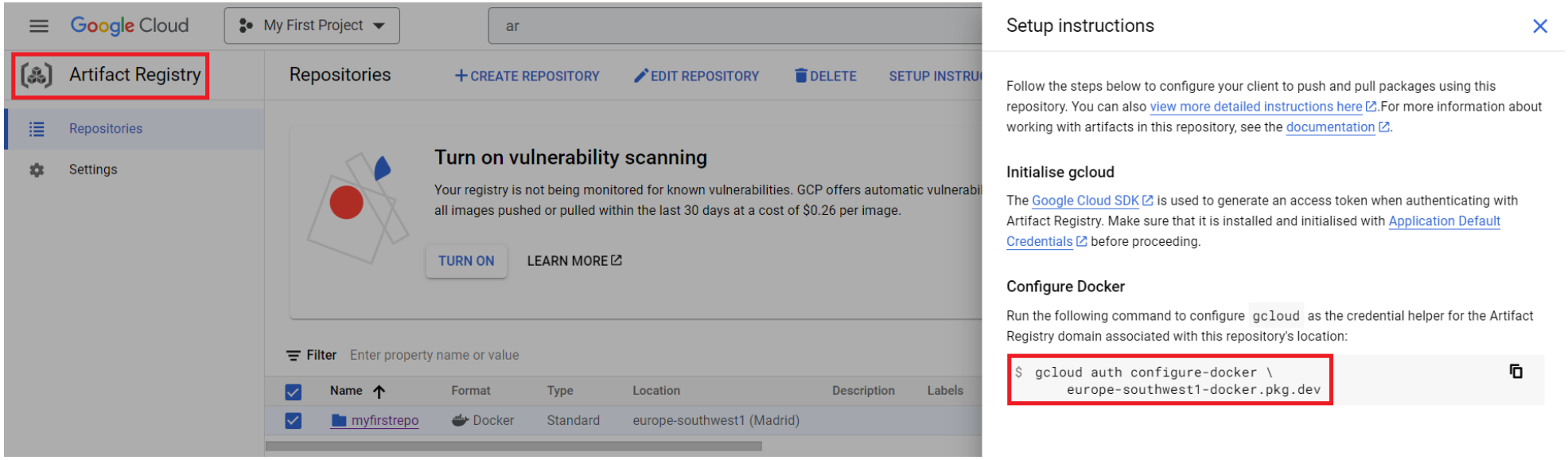
We can verify the new repo in the Artifact Registry list



To Set Up the repo we select it and we press in the **Setup Instructions** button



See the command to Set Up our repo




```
gcloud auth configure-docker europe-southwest1-docker.pkg.dev
```

2. Create a .NET8 WebAPI and create the Dockerfile

Run Visual Studio 2022 Community Edition and create a .NET8 WebAPI

Add Docker support to the application

This is the Dockerfile created automatically:

```
#See https://aka.ms/customizecontainer to learn how to customize your debug container and how Visual Studio uses this Dockerfile

FROM mcr.microsoft.com/dotnet/aspnet:8.0 AS base
USER app
WORKDIR /app
EXPOSE 8080
EXPOSE 8081
```

```
FROM mcr.microsoft.com/dotnet/sdk:8.0 AS build
ARG BUILD_CONFIGURATION=Release
WORKDIR /src
COPY ["GoogleCloudWebAPI.csproj", "."]
RUN dotnet restore "./././GoogleCloudWebAPI.csproj"
COPY . .
WORKDIR "/src/."
RUN dotnet build "./GoogleCloudWebAPI.csproj" -c $BUILD_CONFIGURATION -o /app/build

FROM build AS publish
ARG BUILD_CONFIGURATION=Release
RUN dotnet publish "./GoogleCloudWebAPI.csproj" -c $BUILD_CONFIGURATION -o /app/publish /p:UseAppHost=false

FROM base AS final
WORKDIR /app
COPY --from=publish /app/publish .
ENTRYPOINT ["dotnet", "GoogleCloudWebAPI.dll"]
```

3. Create the Docker image

Open Terminal window and create the Docker image running this command:

```
docker build -t your-webapi-image-name .
```

4. Tag the Docker imge

my-location: europe-southwest1

my-gcloud-project: extreme-axon-381209

my-repo: myfirstrepo

my-imagename: your-webapi-image-name:v1.0

Original imagename:myimagename:latest

Final imagename:my-location-docker.pkg.dev/my-gcloud-project/my-repo/my-imagename:v1.0

```
docker tag myimagename:latest my-location-docker.pkg.dev/my-gcloud-project/my-repo/my-imagename:v1.0
```

```
docker tag your-webapi-image-name:latest europe-southwest1-docker.pkg.dev/extreme-axon-381209/myfirstrepo/your-webapi-image-name:
```

5. Push the Docker image to Google Cloud Artifact Registry repo

5.1. Authenticate gcloud:

Make sure you're authenticated with the Google Cloud SDK and that you're using the correct project. Run:

```
gcloud auth login
```

```
gcloud config set project extreme-axon-381209
```

5.2. Configure Docker to Use gcloud as a Credential Helper:

Run the following command to configure Docker to use gcloud as the credential helper:

```
gcloud auth configure-docker europe-southwest1-docker.pkg.dev
```

This command updates your Docker configuration to use gcloud as the credential helper for repositories in europe-southwest1-docker.pkg.dev

5.3. Push Docker image to Google Cloud Artifact Registry repo

```
docker push europe-southwest1-docker.pkg.dev/extreme-axon-381209/myfirstrepo/your-webapi-image-name:v1.0
```

5.4. Veryfy the Google Cloud Docker image in Docker Desktop

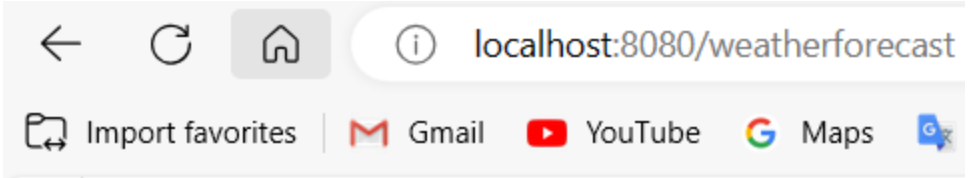
We pull the Docker image from Google Cloud Artifact Registry repo

```
docker pull europe-southwest1-docker.pkg.dev/extreme-axon-381209/myfirstrepo/your-webapi-image-name:v1.0
```

We run the Docker image in Docker Destop

```
docker run -p 8080:8080 -p 8081:8081 europe-southwest1-docker.pkg.dev/extreme-axon-381209/myfirstrepo/your-webapi-image-name:v1.0
```

http://localhost:8080/weatherforecast




```
[
    {
        "date": "2024-01-12",
        "temperatureC": 38,
        "temperatureF": 100,
        "summary": "Hot"
    },
    {
        "date": "2024-01-13",
        "temperatureC": 6,
        "temperatureF": 42,
        "summary": "Hot"
    },
    {
        "date": "2024-01-14",
        "temperatureC": -10,
        "temperatureF": 15,
        "summary": "Warm"
    },
    {
        "date": "2024-01-15",
        "temperatureC": 20,
        "temperatureF": 67,
        "summary": "Warm"
    },
    {
        "date": "2024-01-16",
        "temperatureC": 34,
        "temperatureF": 93,
        "summary": "Warm"
    }
]
```


6. Deploying the Docker image

```
gcloud run deploy --image europe-southwest1-docker.pkg.dev/extreme-axon-381209/myfirstrepo/your-webapi-image-name:v1.0
```