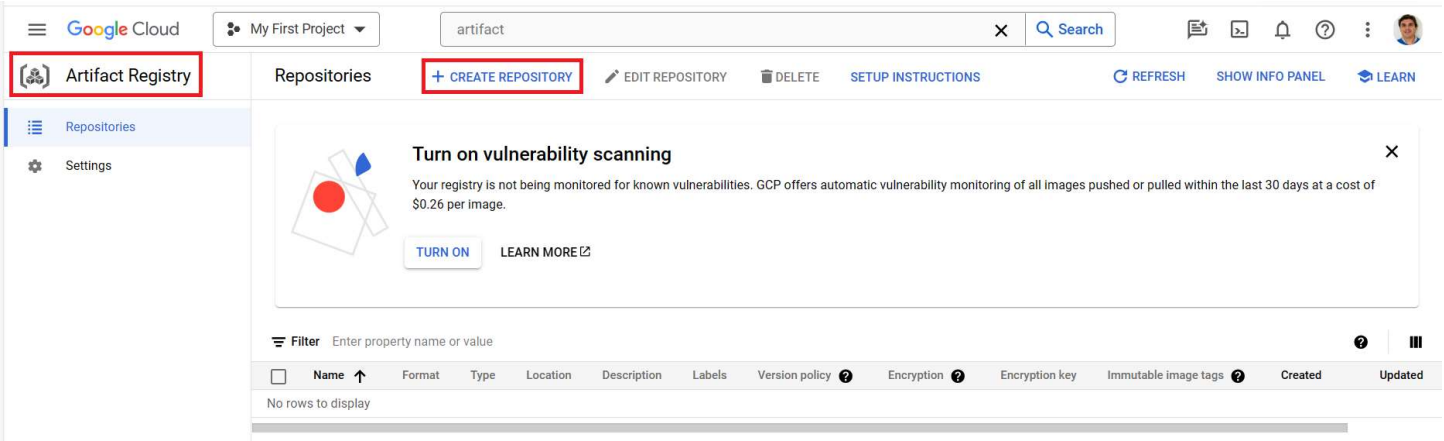
GithubActions: How to create .NET8 WebAPI Docker image and upload to Google Cloud Artifacts Registry

You can see this example source code in this github repo:

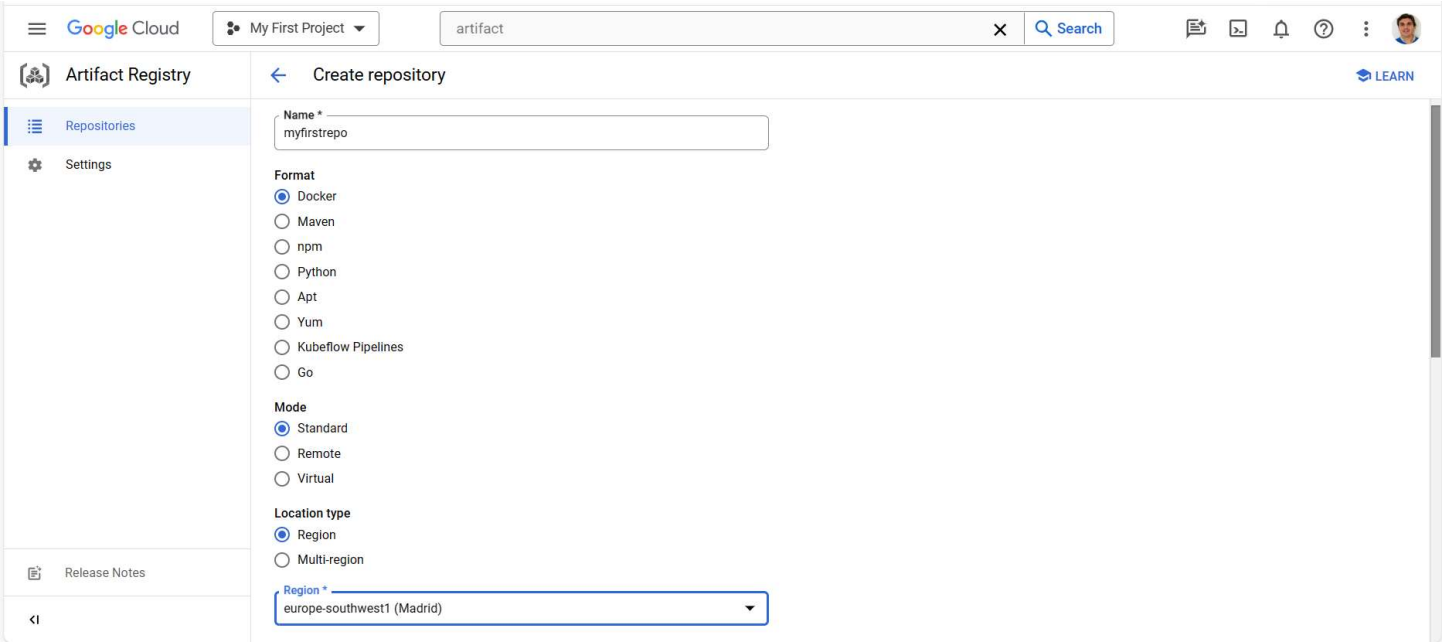
https://github.com/luiscoco/GithubActions_dotNET8WebAPI_Create_DockerImage_Upload_to_Google Cloud_Artifacts_Registry

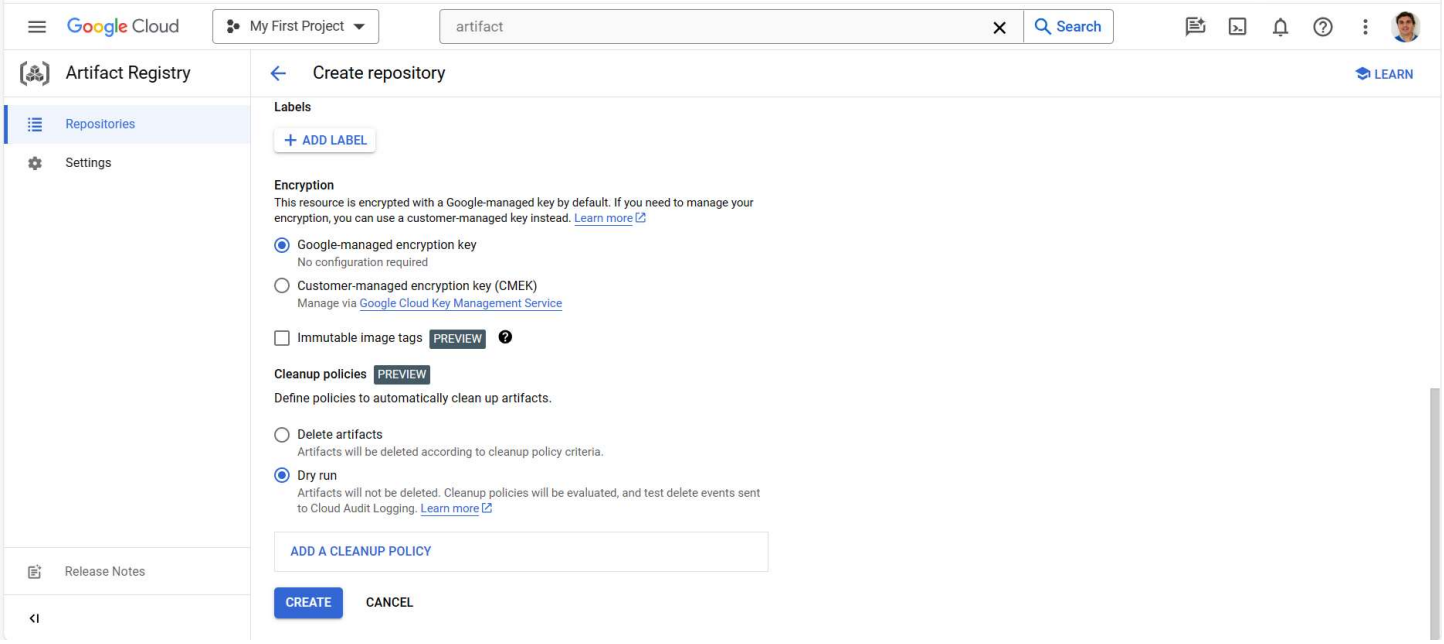
0. Prerequisite

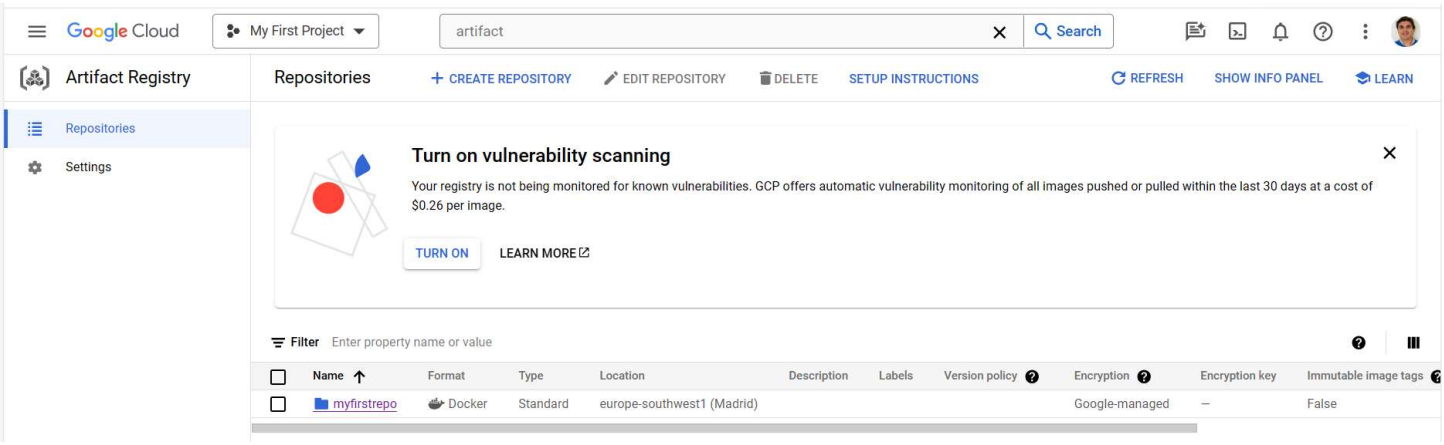
We have to create a new repo in Google Cloud Artifacts Registry



We set the input data

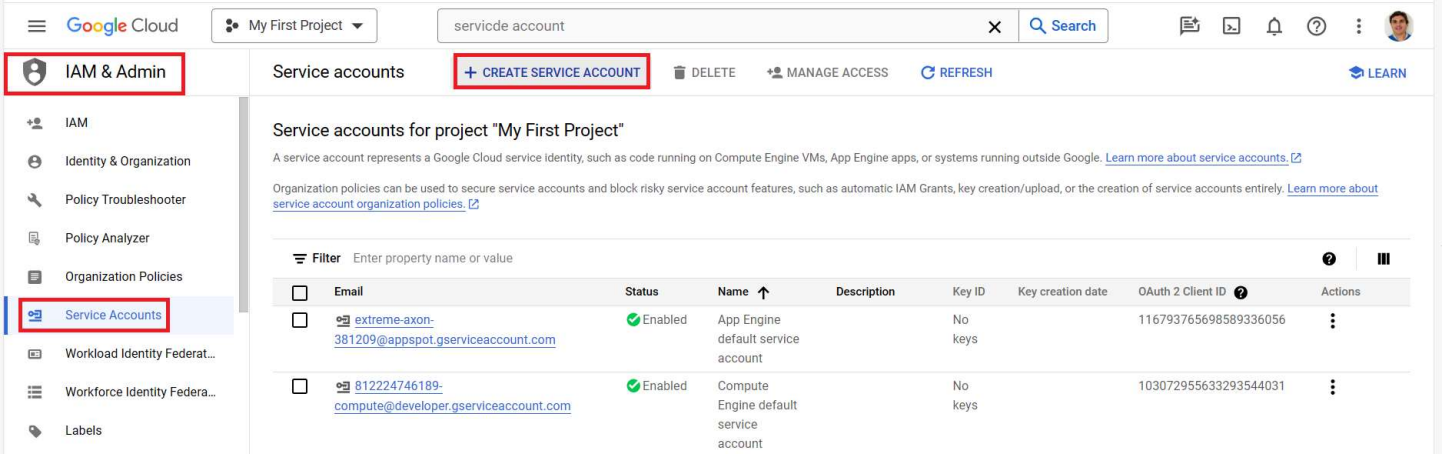






1. Create a Service Account in Google Cloud Platform

Go to the GCP Console: Open the Google Cloud Console and log in to your account.



Select Your Project: Make sure you have the correct project selected in which you want to create the service account.

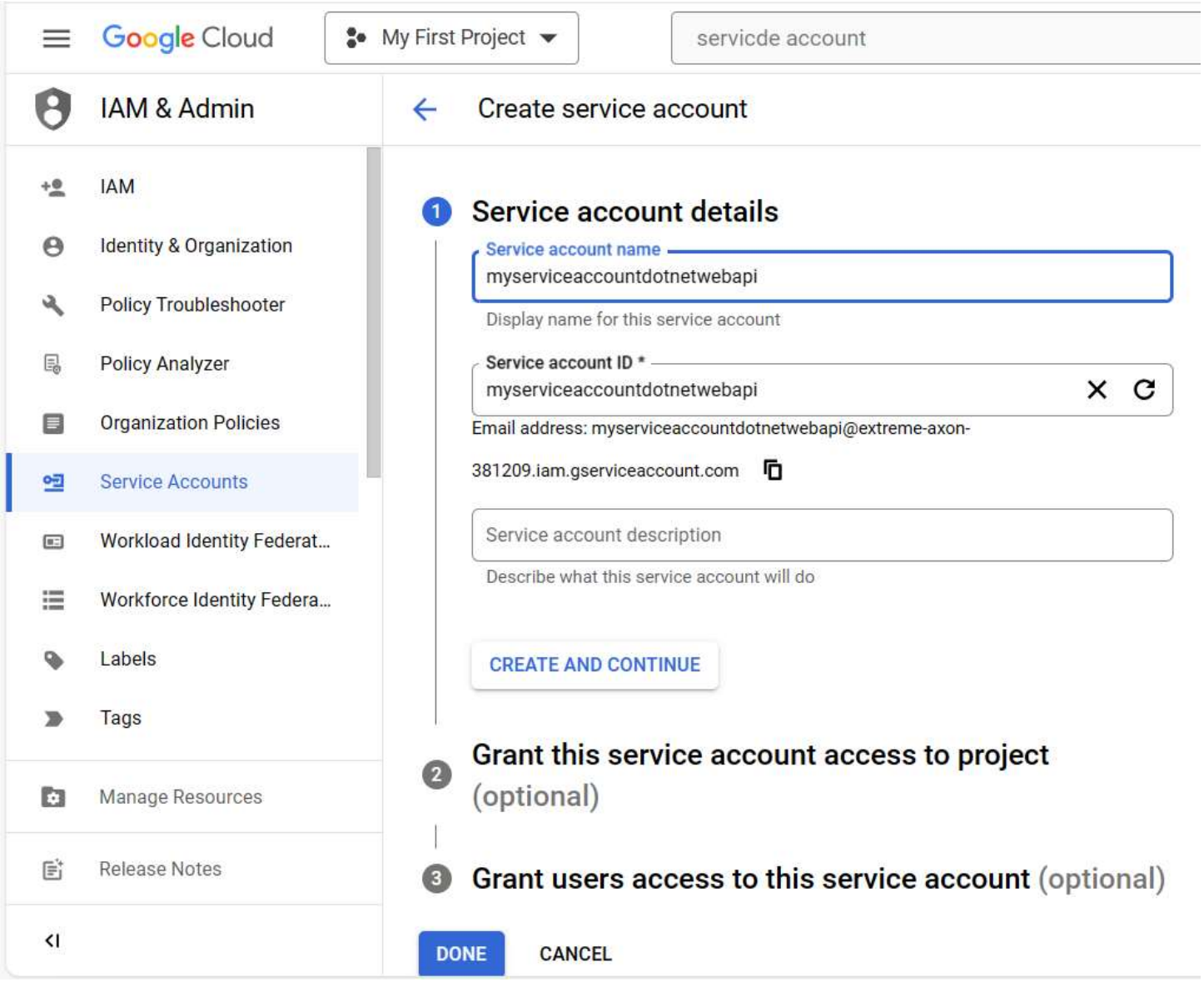
Navigate to IAM & Admin: In the left-hand menu, click on "IAM & Admin", then select "Service Accounts".

Create Service Account: Click on "Create Service Account" and fill in the necessary details:

Name: Give your service account a name.

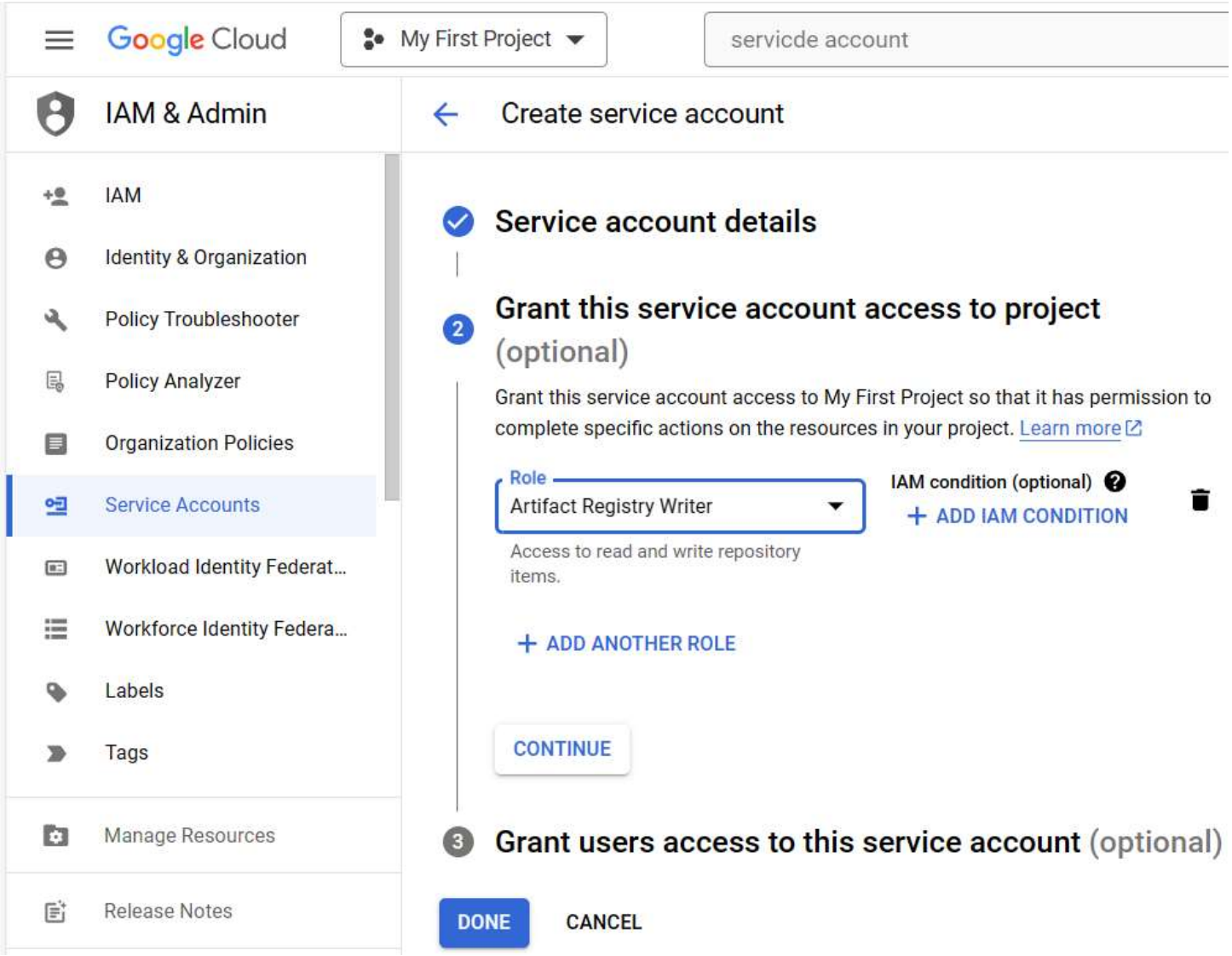
ID: This is filled automatically based on the name but can be customized.

Description: (Optional) Add a description for your service account.

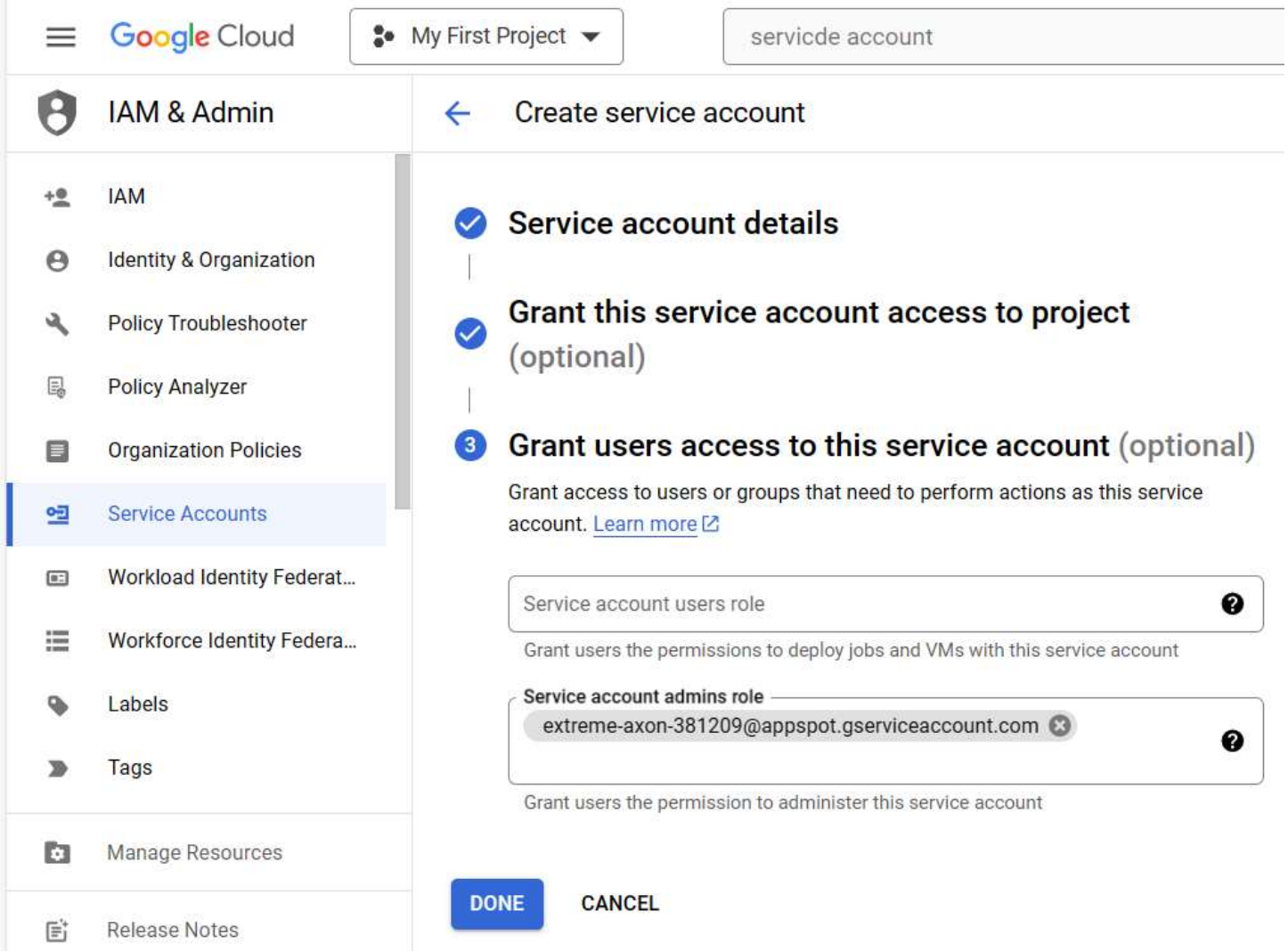


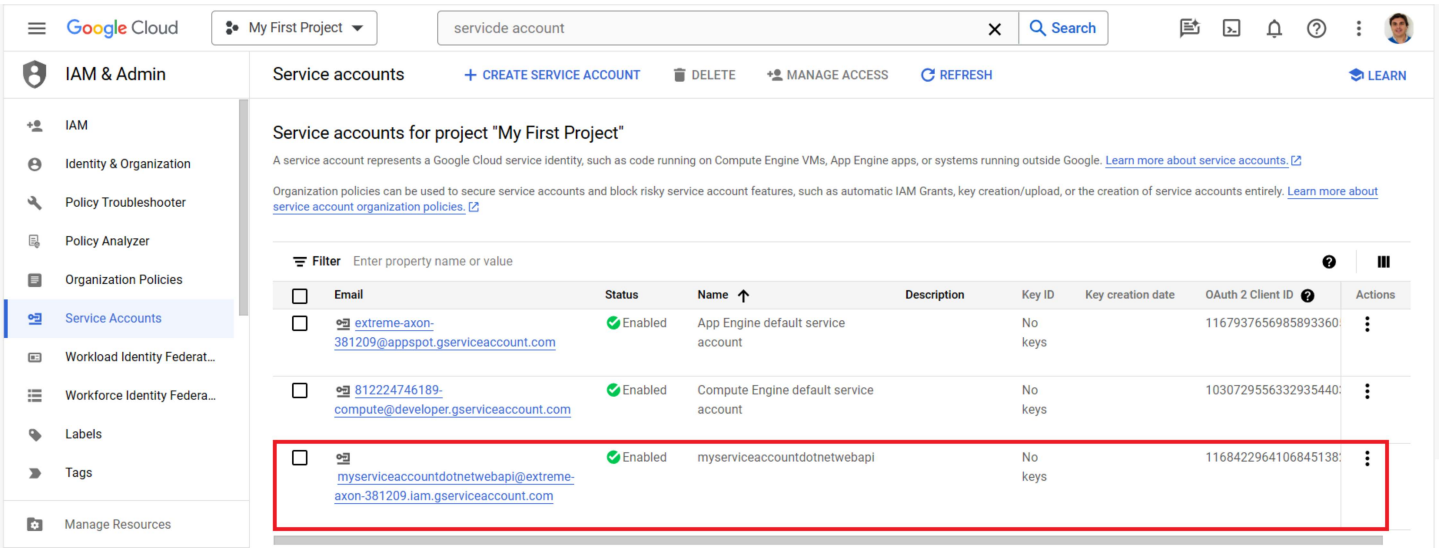
Grant Access: Assign the service account appropriate roles. For Docker images push we can assing the role: "**Artifact Registry Writer**"

Other similar roles could be: "**Storage Admin**" or "**Artifact Registry Administrator**"



Do not forget to set the project ID (for this example: extreme-axon-381209) in the **Service account admin role**: extreme-axon-381209@appspot.gserviceaccount.com App Engine default service account

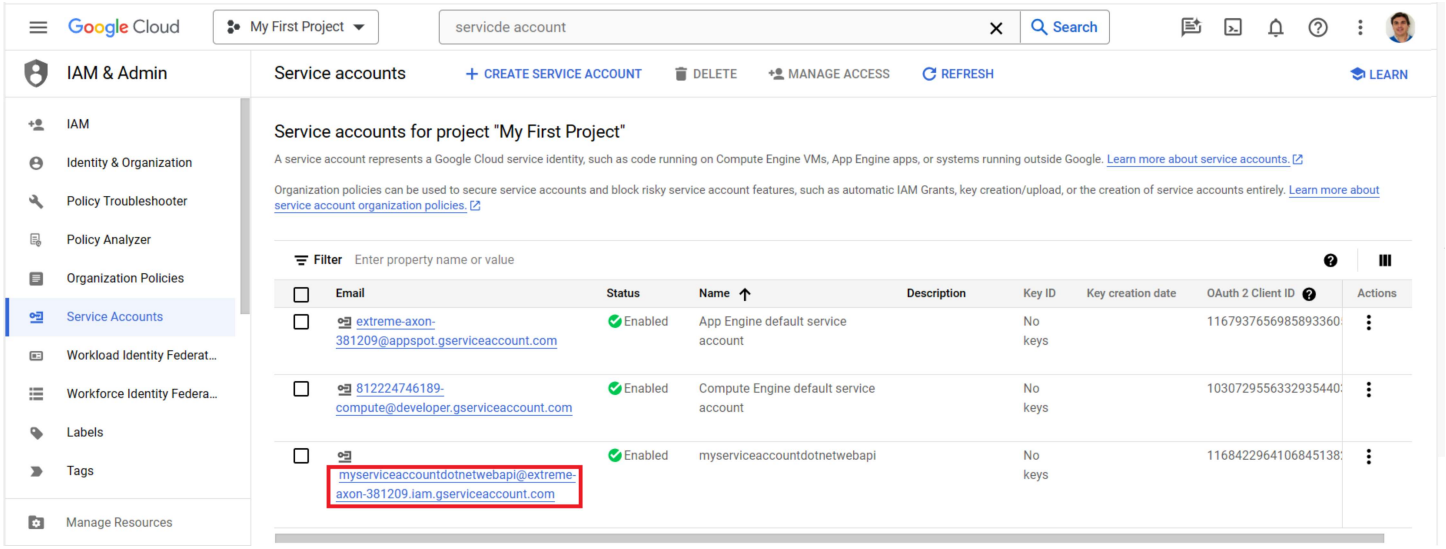


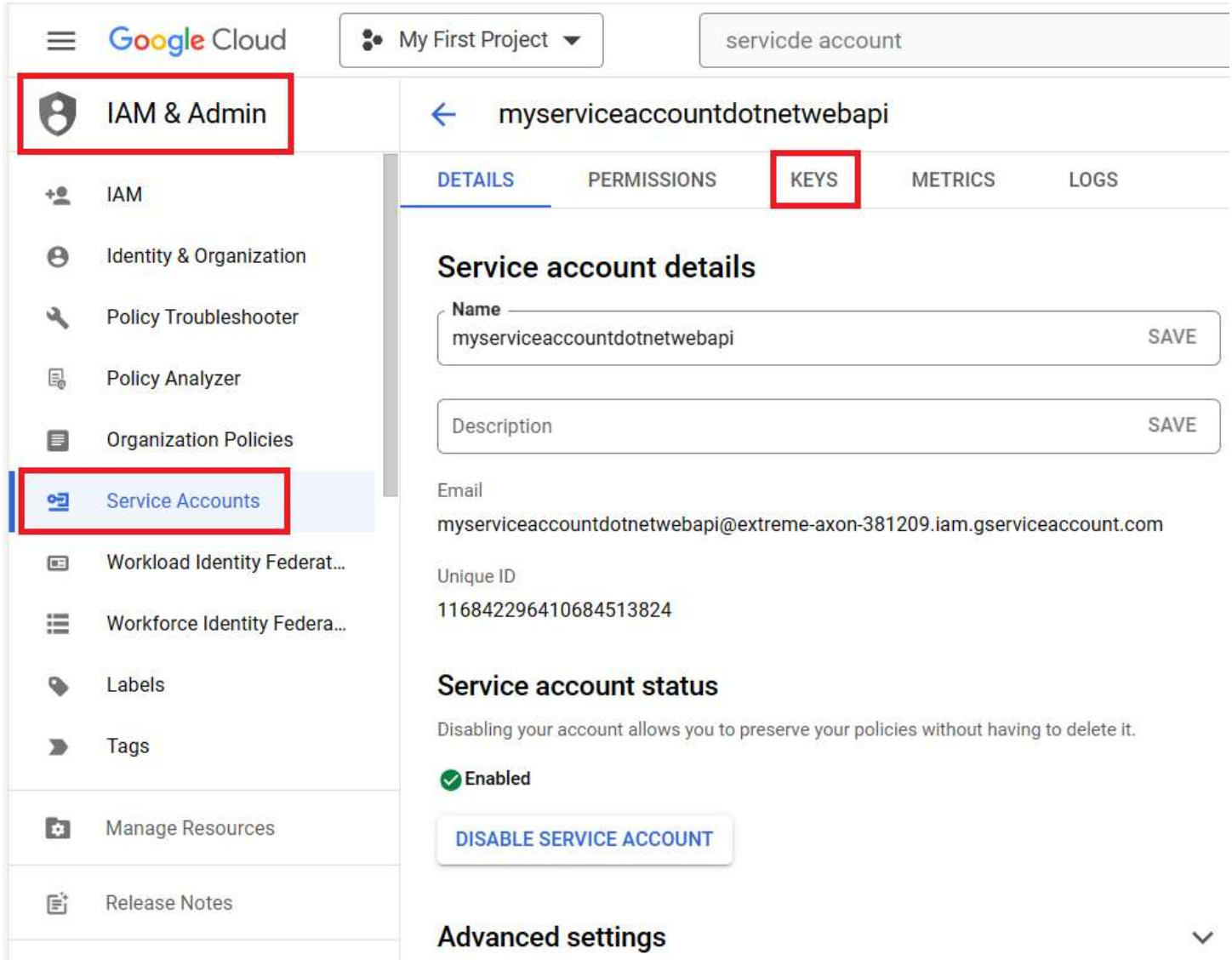


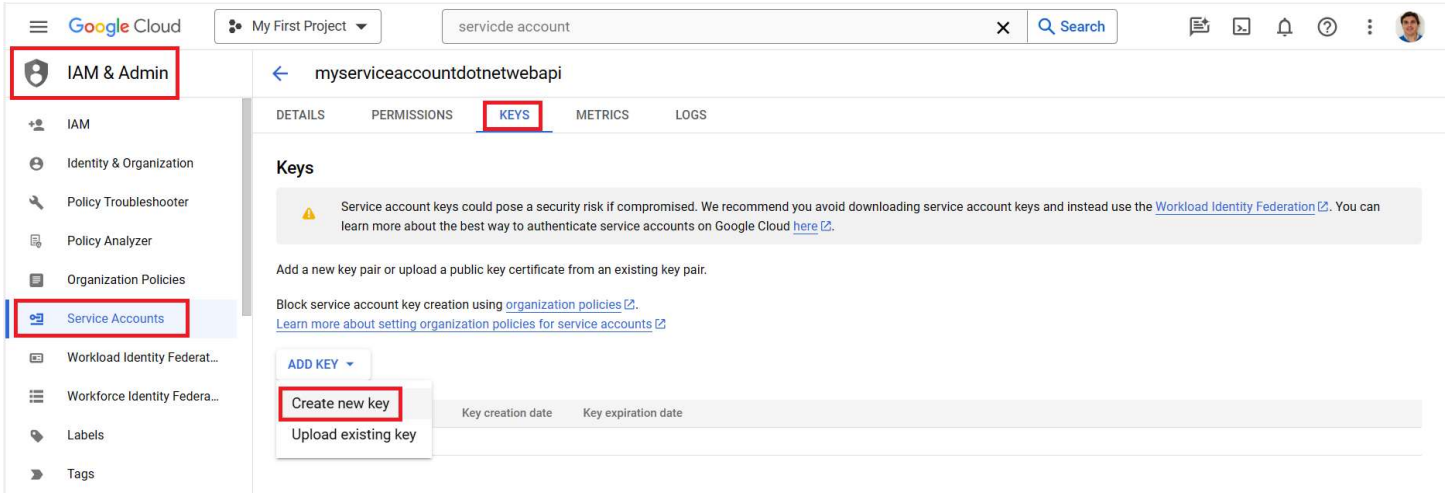
Be cautious with permissions to follow the principle of least privilege.

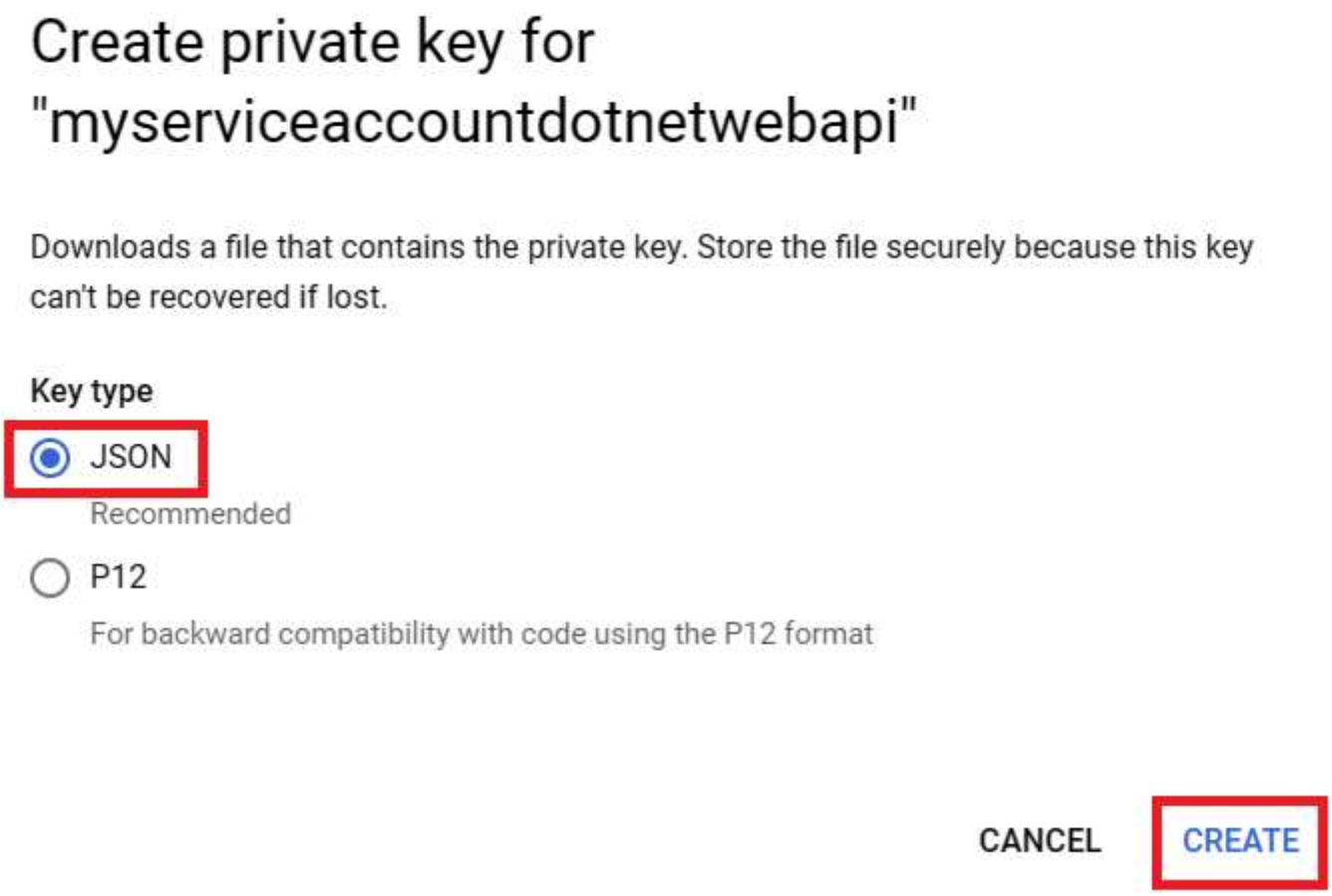
Create Key: After creating the service account, click on it to open its details. Under the "Keys" tab, click "Add Key", then select "Create new key".

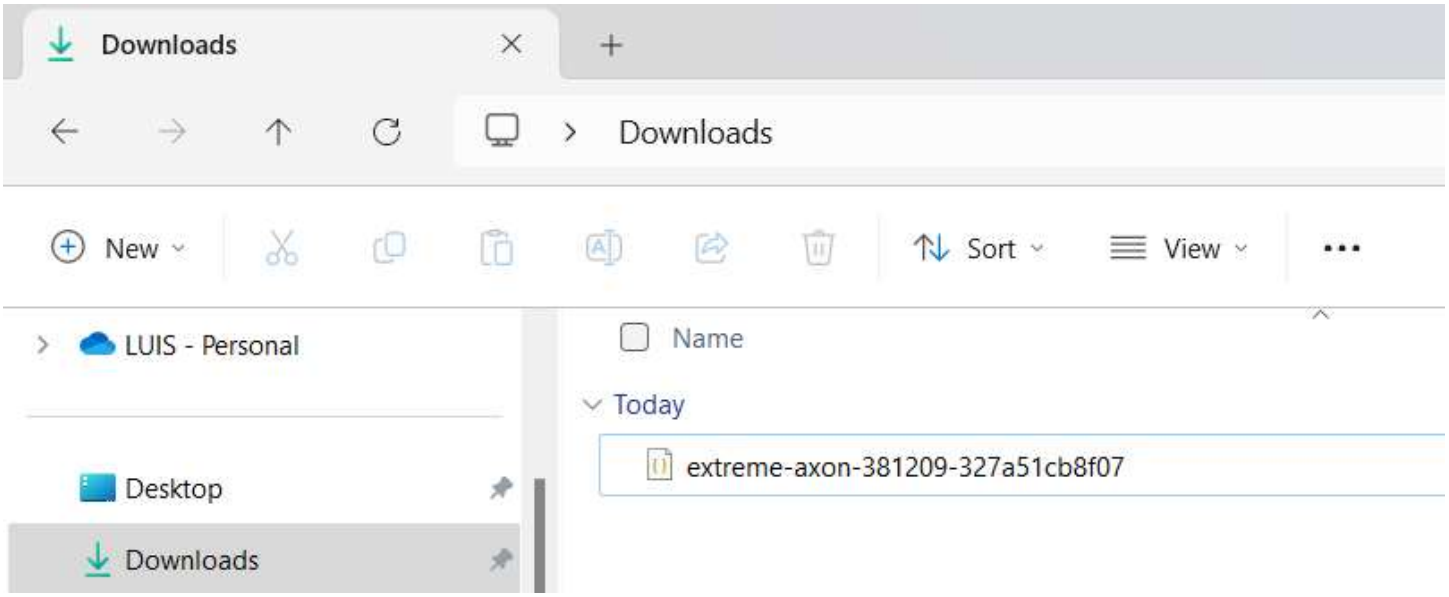
Choose "JSON" as the key type and click "Create". This will download the JSON key file to your computer.


Google Cloud
My First Project
servicde account
Search
IAM & Admin
Service accounts
+ CREATE SERVICE ACCOUNT
DELETE
MANAGE ACCESS
REFRESH
LEARN
IAM
Identity & Organization
Policy Troubleshooter
Policy Analyzer
Organization Policies
Service Accounts
Workload Identity Federat...
Workforce Identity Federa...
Labels
Tags
Manage Resources
Service accounts for project "My First Project"
A service account represents a Google Cloud service identity, such as code running on Compute Engine VMs, App Engine apps, or systems running outside Google. Learn more about service accounts.
Organization policies can be used to secure service accounts and block risky service account features, such as automatic IAM Grants, key creation/upload, or the creation of service accounts entirely. Learn more about service account organization policies.
Filter Enter property name or value
Email
Status
Name
Description
Key ID
Key creation date
OAuth 2 Client ID
Actions
extreme-axon-381209@appspot.gserviceaccount.com
Enabled
App Engine default service account
No keys
812224746189-compute@developer.gserviceaccount.com
Enabled
Compute Engine default service account
No keys
myserviceaccountdotnetwebapi@extreme-axon-381209.iam.gserviceaccount.com
Enabled
myserviceaccountdotnetwebapi
No keys


Google Cloud
My First Project
servicde account
IAM & Admin
myserviceaccountdotnetwebapi
IAM
Identity & Organization
Policy Troubleshooter
Policy Analyzer
Organization Policies
Service Accounts
Workload Identity Federat...
Workforce Identity Federa...
Labels
Tags
Manage Resources
Release Notes
DETAILS
PERMISSIONS
KEYS
METRICS
LOGS
Service account details
Name
myserviceaccountdotnetwebapi
SAVE
Description
SAVE
Email
myserviceaccountdotnetwebapi@extreme-axon-381209.iam.gserviceaccount.com
Unique ID
116842296410684513824
Service account status
Disabling your account allows you to preserve your policies without having to delete it.
Enabled
DISABLE SERVICE ACCOUNT
Advanced settings


Google Cloud
My First Project
servicde account
Search
IAM & Admin
IAM
Identity & Organization
Policy Troubleshooter
Policy Analyzer
Organization Policies
Service Accounts
Workload Identity Federat...
Workforce Identity Federa...
Labels
Tags
myserviceaccountdotnetwebapi
DETAILS
PERMISSIONS
KEYS
METRICS
LOGS
Keys
Service account keys could pose a security risk if compromised. We recommend you avoid downloading service account keys and instead use the Workload Identity Federation. You can learn more about the best way to authenticate service accounts on Google Cloud here.
Add a new key pair or upload a public key certificate from an existing key pair.
Block service account key creation using organization policies.
Learn more about setting organization policies for service accounts
ADD KEY
Create new key
Upload existing key
Key creation date
Key expiration date


Create private key for "myserviceaccountdotnetwebapi"
Downloads a file that contains the private key. Store the file securely because this key can't be recovered if lost.
Key type
JSON
Recommended
P12
For backward compatibility with code using the P12 format
CANCEL
CREATE



2. Add the Key as a Secret in your GitHub Repository

Go to Your GitHub Repository: Open your GitHub repository in a web browser.

Navigate to Settings: Click on "**Settings**" in the top menu of your repository.

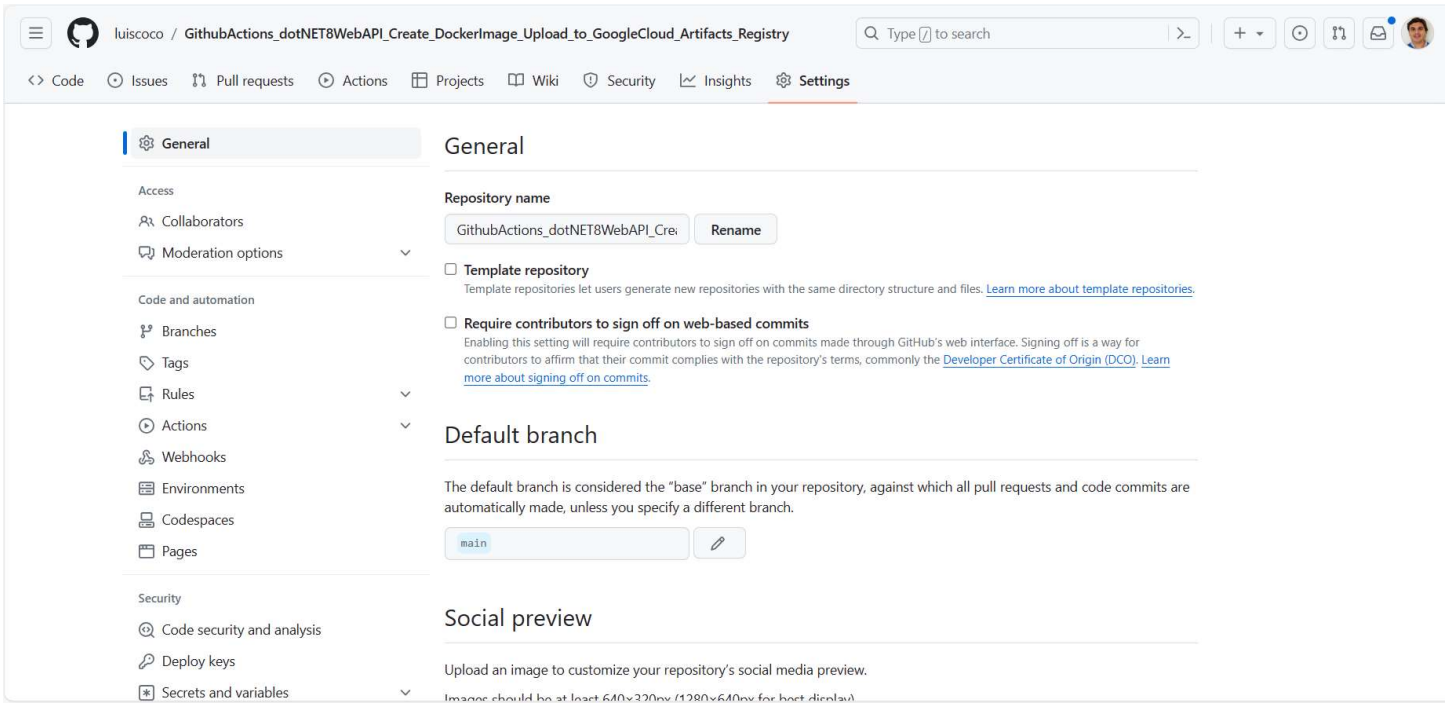
Access Secrets: In the left-hand sidebar, click on "**Secrets**", then select "**Actions**".

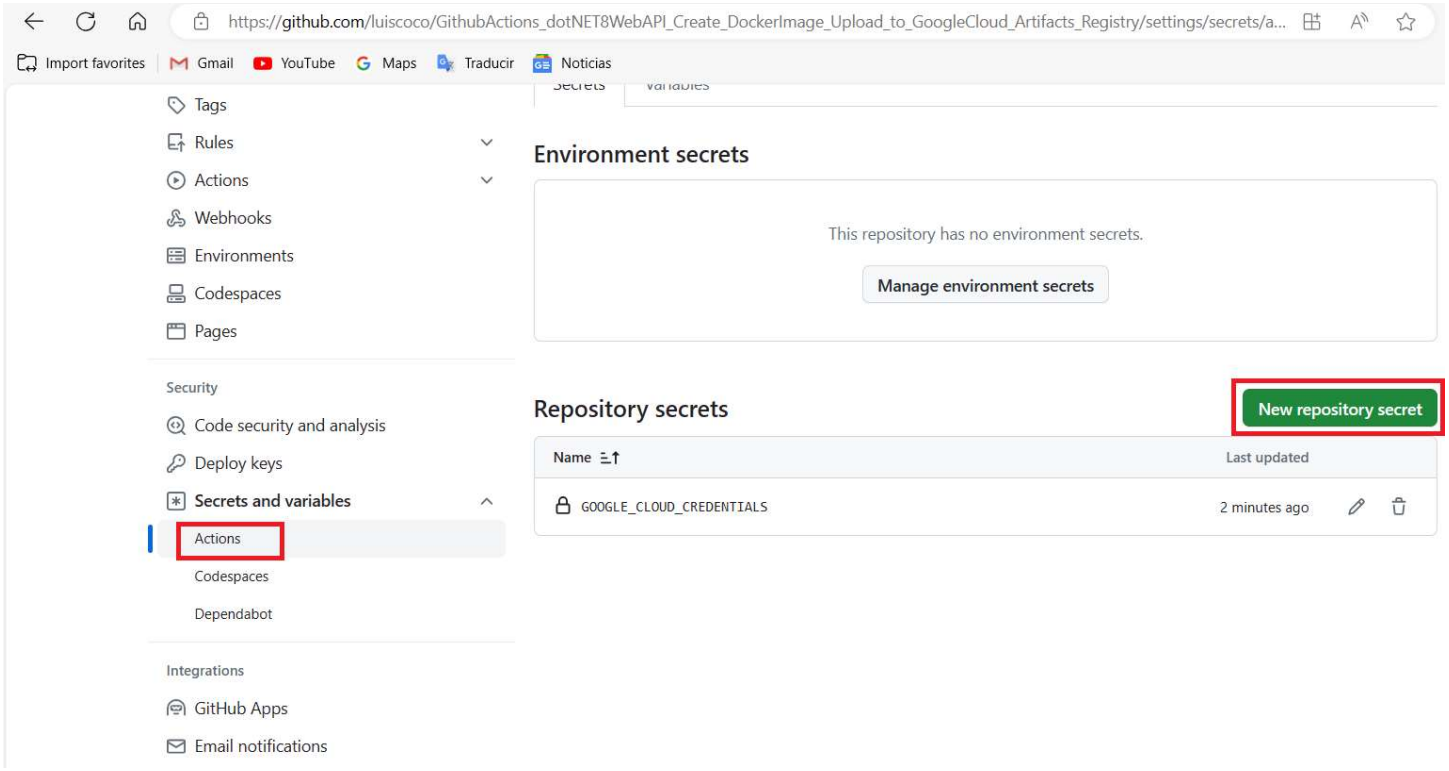
Add a New Secret: Click on "**New repository secret**".

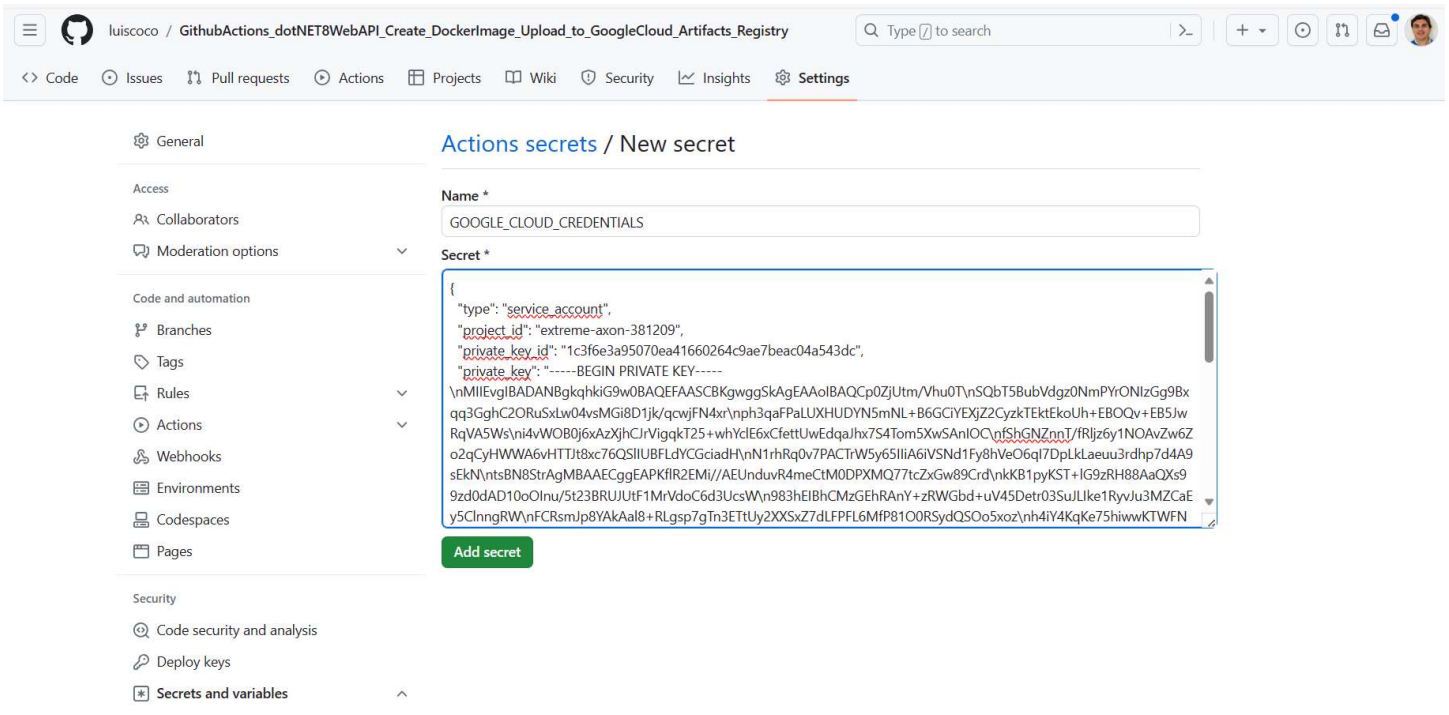
Name Your Secret: Enter **GOOGLE_CLOUD_CREDENTIALS** as the name.

Paste the Key Content: Open the **JSON** key file you downloaded from GCP in a text editor, copy all its contents, and paste them into the secret's value field in GitHub.

Save the Secret: Click "Add secret" to save your new secret.







Now, your GitHub Actions workflow can use this **secret** to authenticate with Google Cloud services. In your workflow file, you can reference this secret as ${{ secrets.GOOGLE_CLOUD_CREDENTIALS }}.

3. Create the main.yml file for Github actions workflow

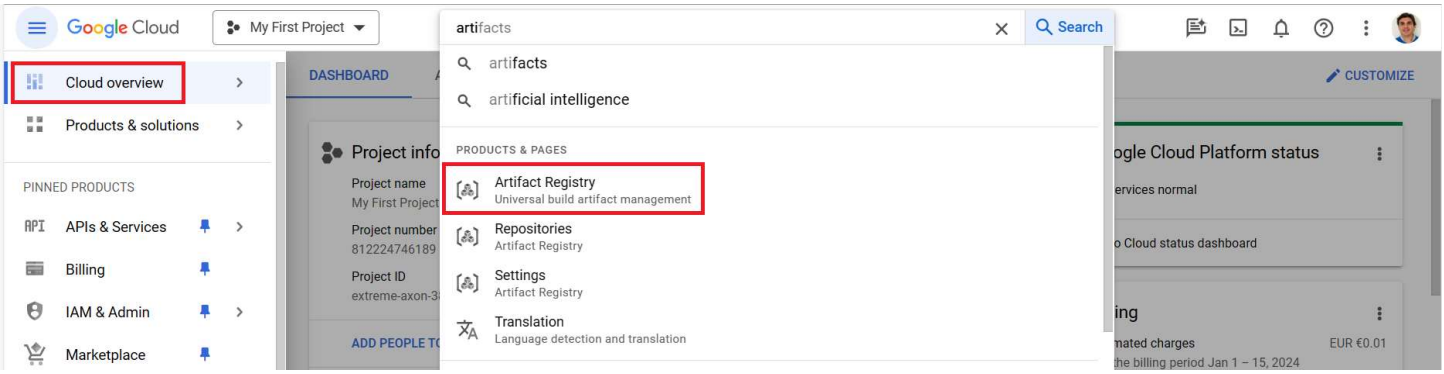
Below is the **main.yml** file tailored for your requirements.

This workflow assumes you have already set up Google Cloud credentials as secrets in your GitHub repository

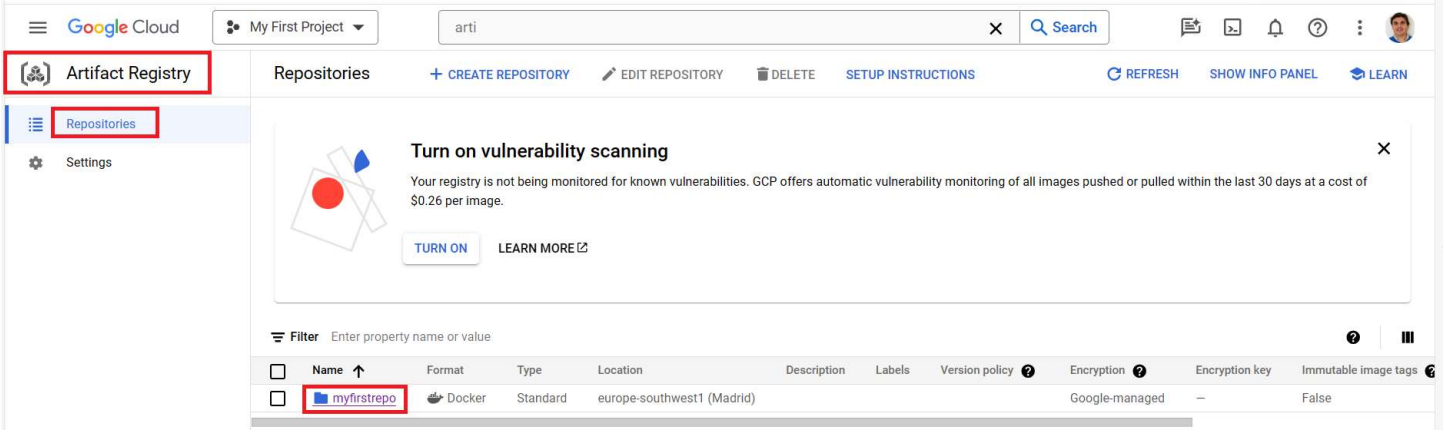
```yaml
name: Build and Push Docker Image

on:
  push:
    branches:
      - main

env:
  PROJECT_ID: extreme-axon-381209
  IMAGE_NAME: my-dotnetwebapi
  REPOSITORY: europe-southwest1-docker.pkg.dev/extreme-axon-381209/myfirstrepo
  TAG: latest

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Authenticate to Google Cloud
      uses: google-github-actions/auth@v2
      with:
        credentials_json: ${{ secrets.GOOGLE_CLOUD_CREDENTIALS }}

    - name: Configure Docker for Google Cloud Artifact Registry
      run: |
        echo '${{ secrets.GOOGLE_CLOUD_CREDENTIALS }}' | gcloud auth activate-service-account
        gcloud auth configure-docker europe-southwest1-docker.pkg.dev --quiet
    - name: Build Docker image
      run: |
        docker build -t ${{ env.REPOSITORY }}/${{ env.IMAGE_NAME }}:${{ env.TAG }} .
    - name: Push Docker image
      run: |
        docker push ${{ env.REPOSITORY }}/${{ env.IMAGE_NAME }}:${{ env.TAG }}
    - name: Verify the image was pushed
      run: |
        gcloud artifacts docker images list ${{ env.REPOSITORY }}/${{ env.IMAGE_NAME }}
```

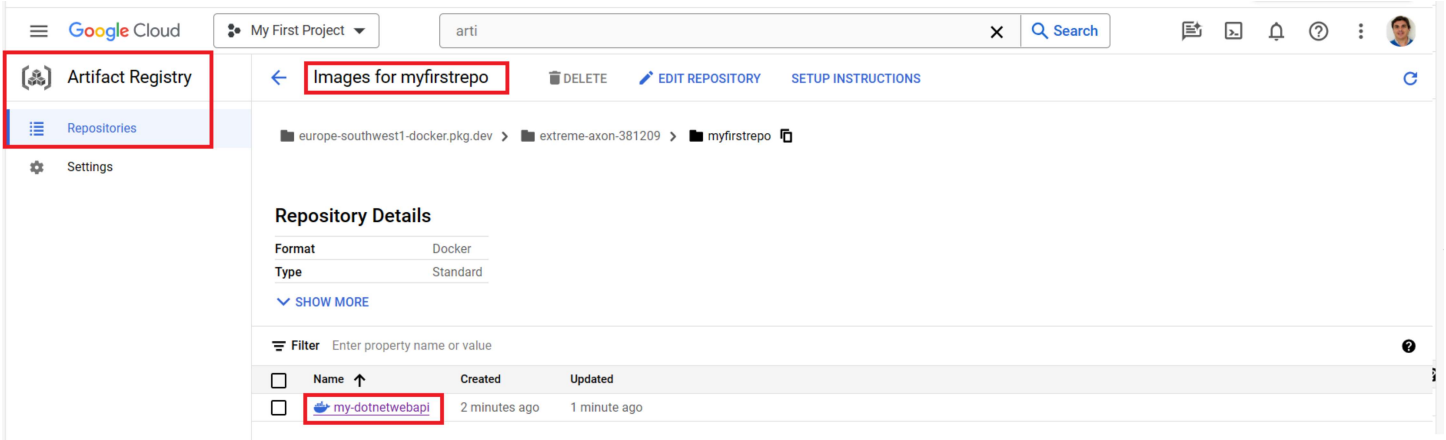
4. Verify the Docker image uploaded to Google Cloud

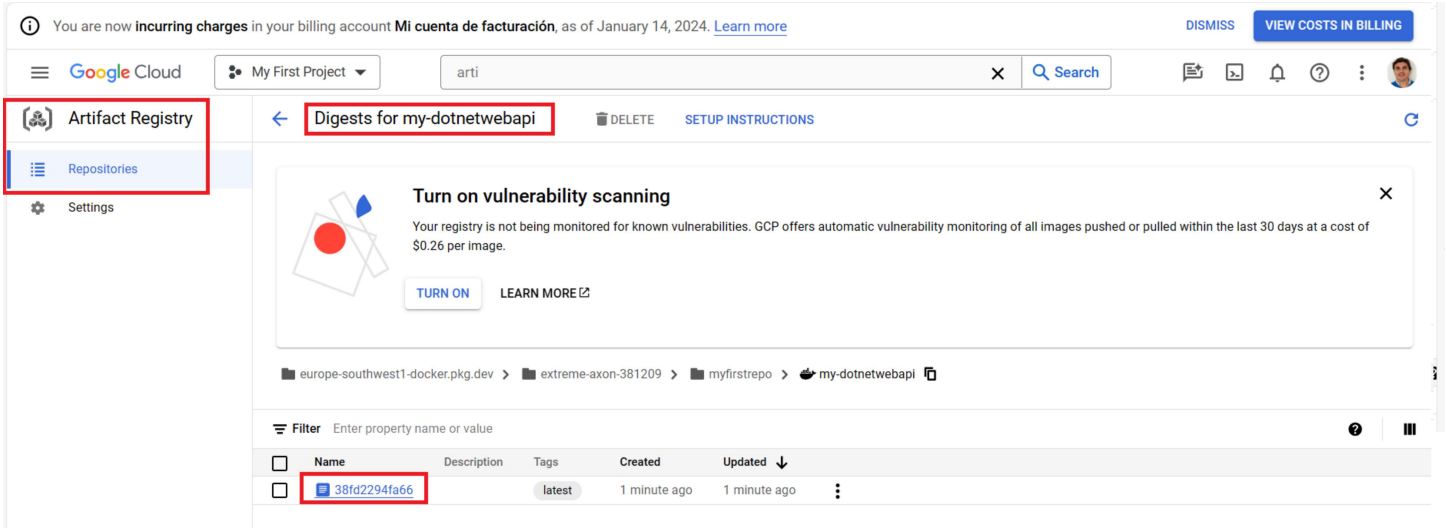
Navigate to Google Cloud Artifacts Registry repo

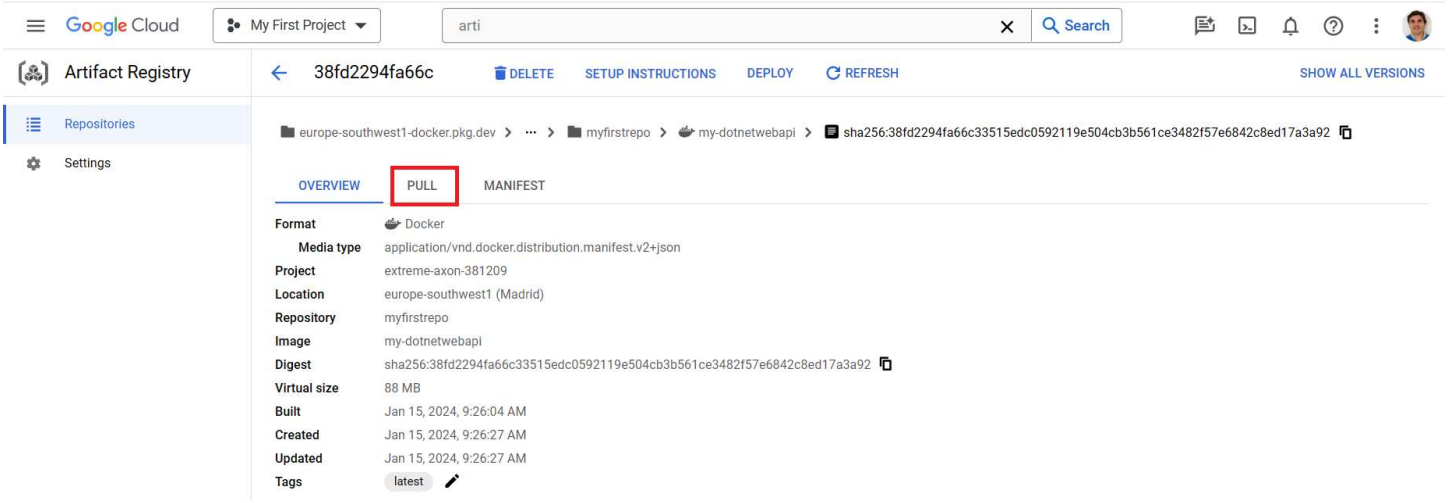


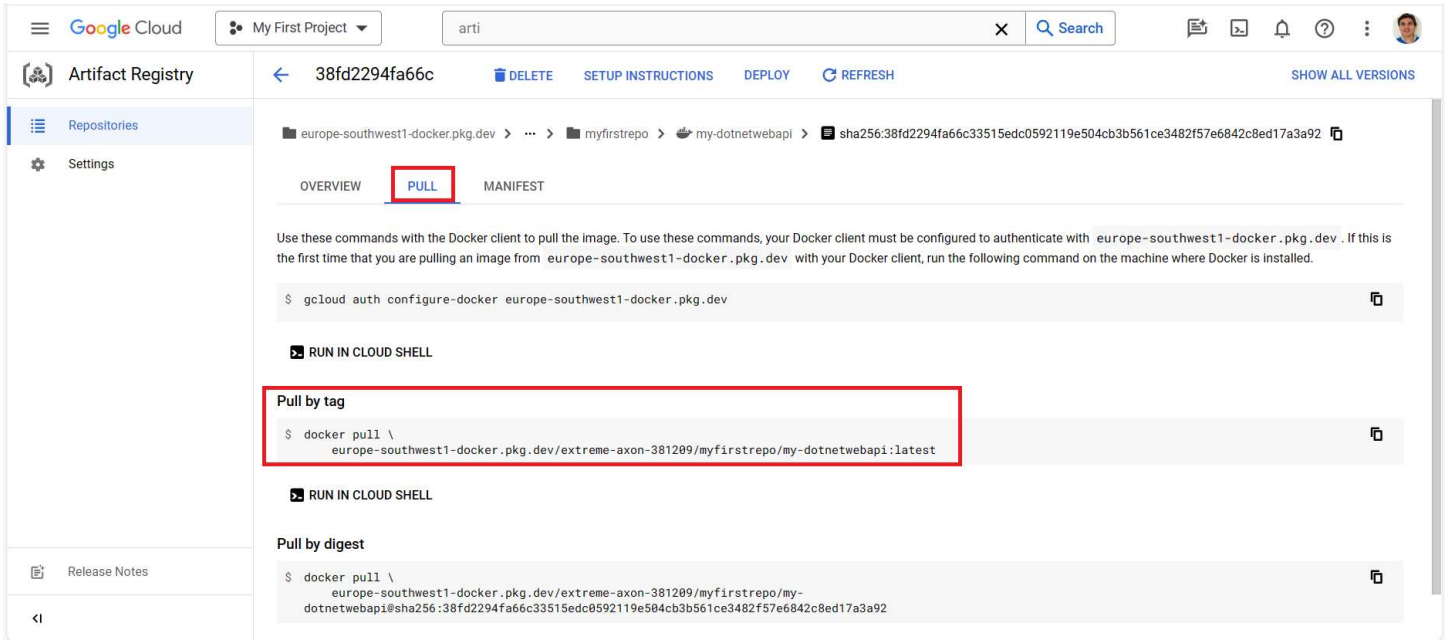
We can see inside the repo the uploaded Docker image







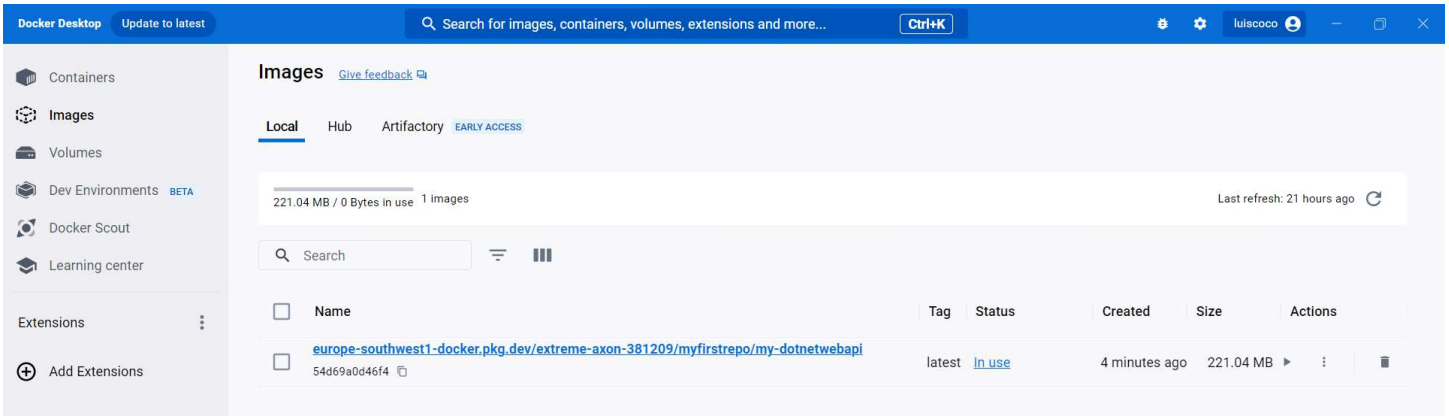




We run this command to pull the image and to

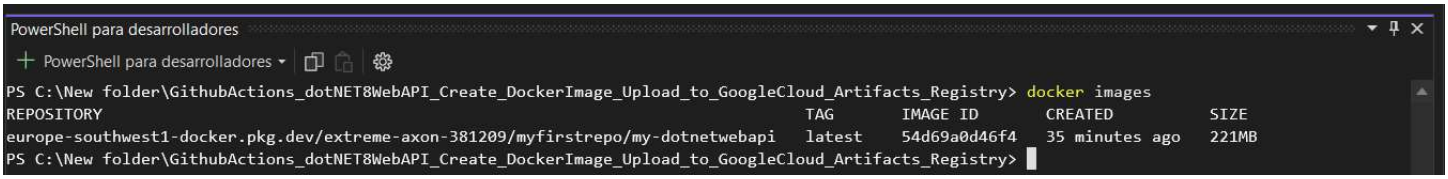
```
docker pull europe-southwest1-docker.pkg.dev/extreme-axon-381209/myfirstrepo/my-dotnetwebapi:l
```

We verified the downloaded image in Docker Desktop



Also we can see the image with the command

```
docker images
```



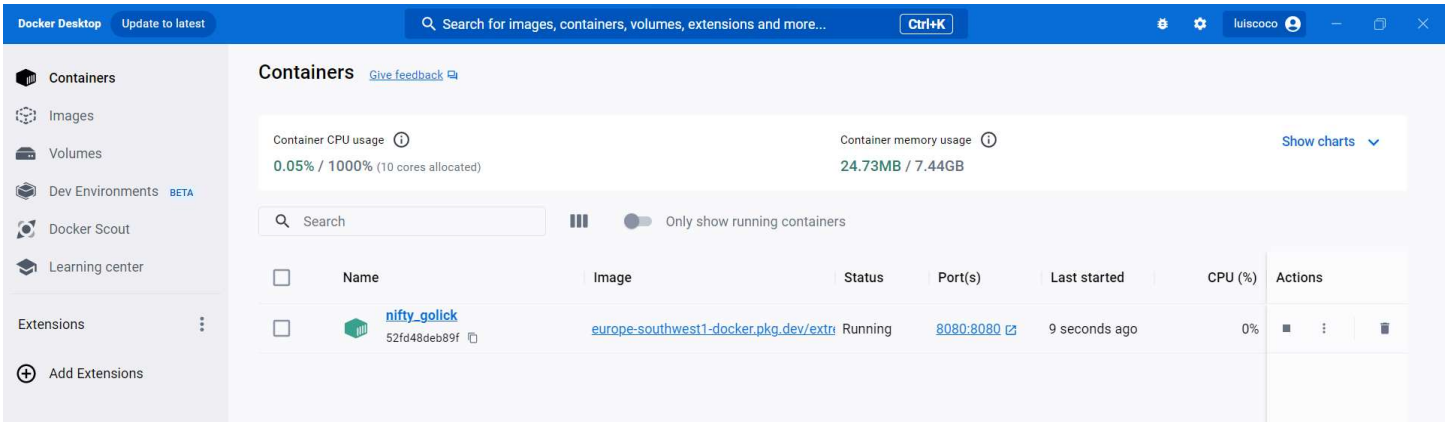
We run the image in our Docker Desktop

```
docker run -p 8080:8080 europe-southwest1-docker.pkg.dev/extreme-axon-381209/myfirstrepo/my-do
```

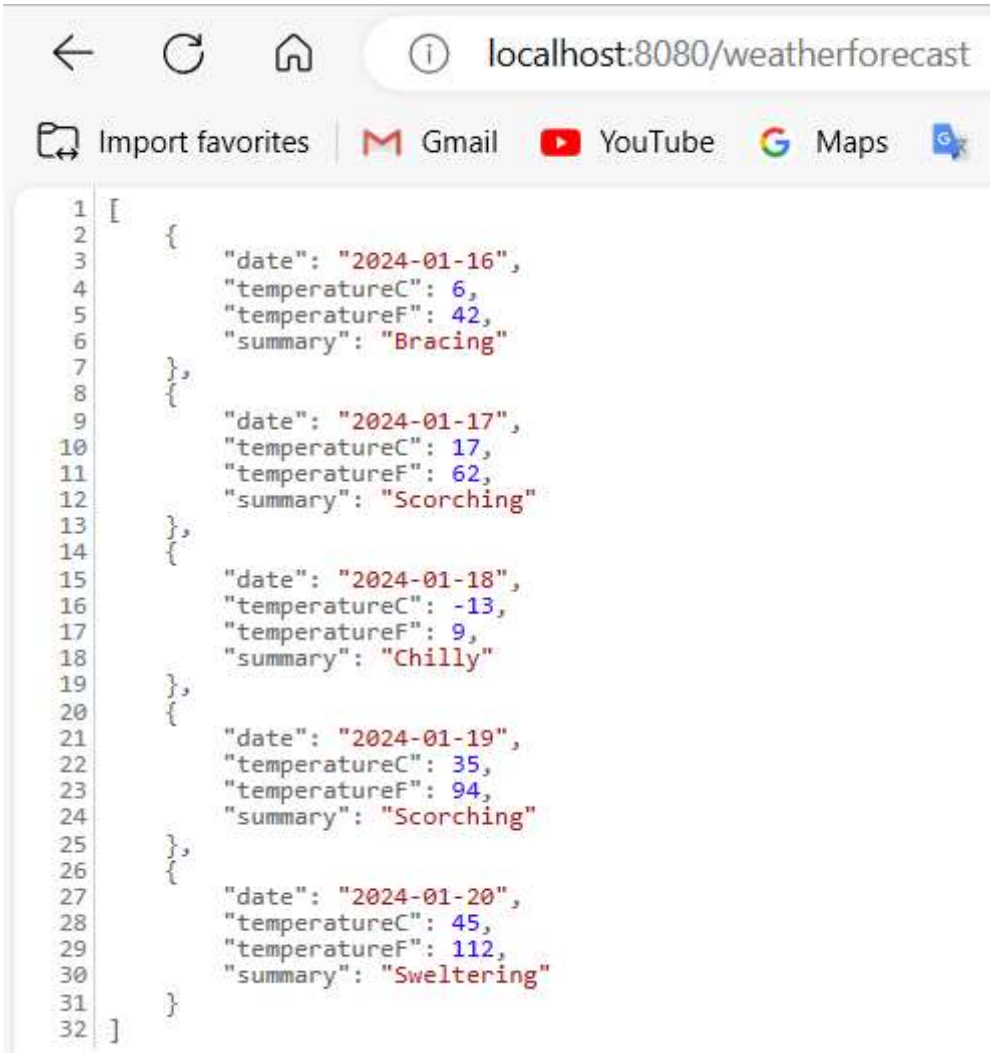
We see the running image with the command

```
docker ps
```

And also we can see the image in Docker Desktop



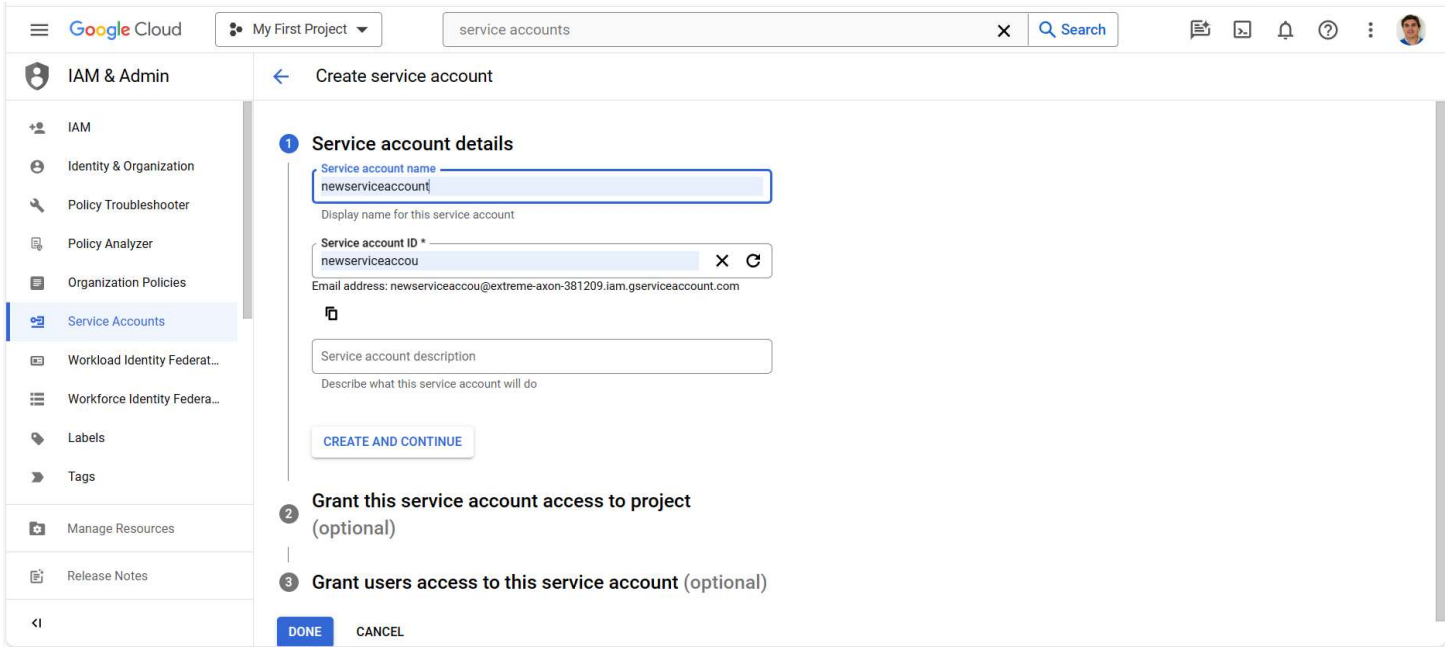
We can verify the running Docker container

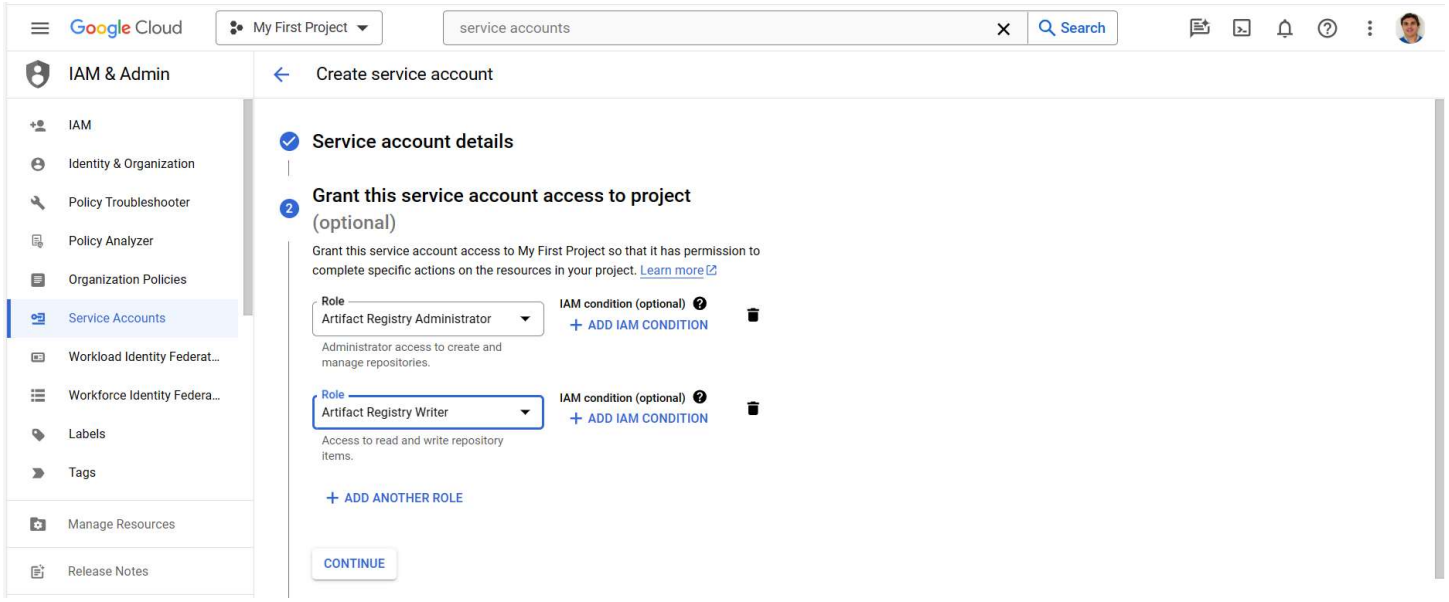


5. Modify the main.yml for creating a new repo if it doesn't exist

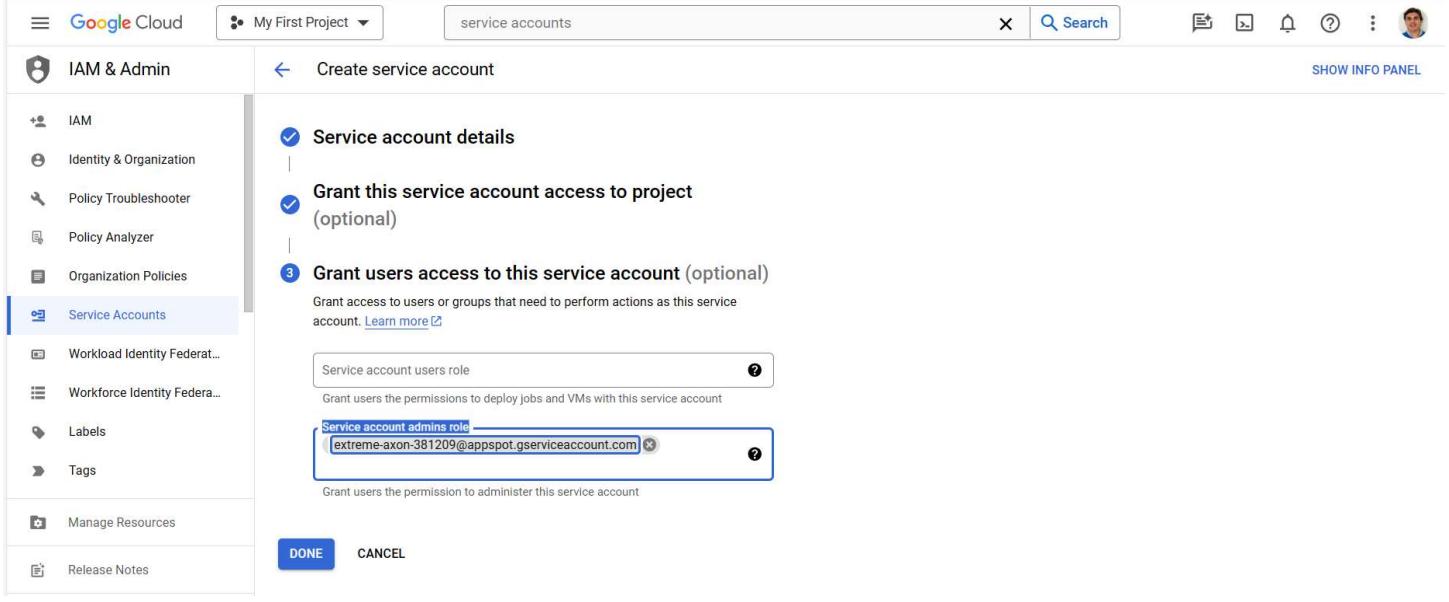
As prerequisite for creating a new Artifact Registry repo it is mandatory to assign to the Service Account the permission **Artifact Registry Administrator**

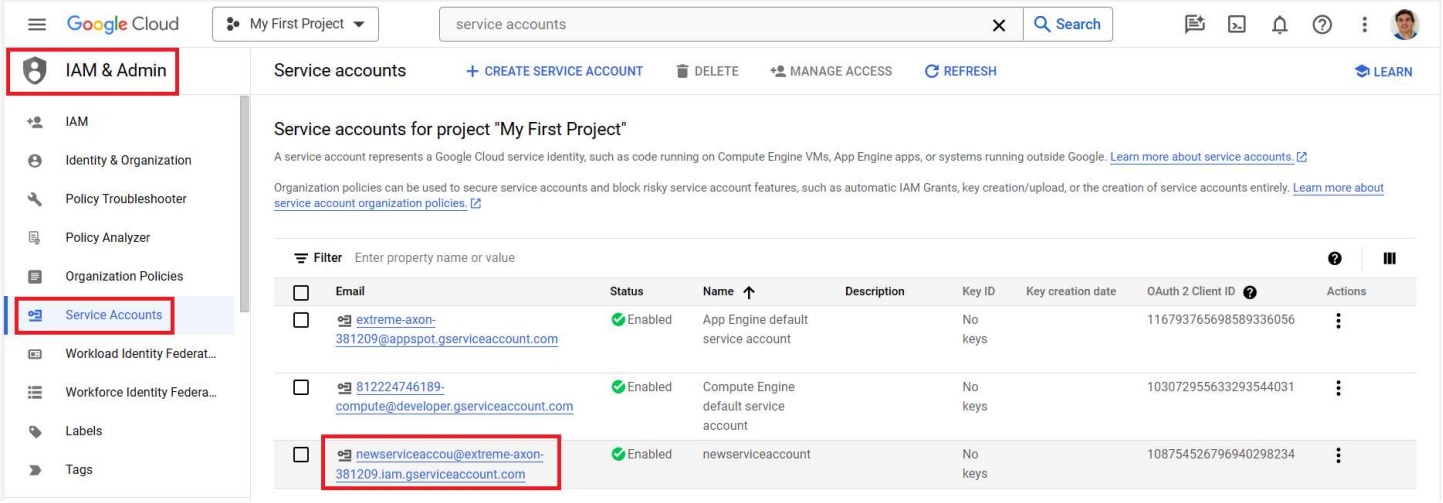
We also set the permission **Artifact Registry Writer** to push Docker image to the Artificat Registry rerpo





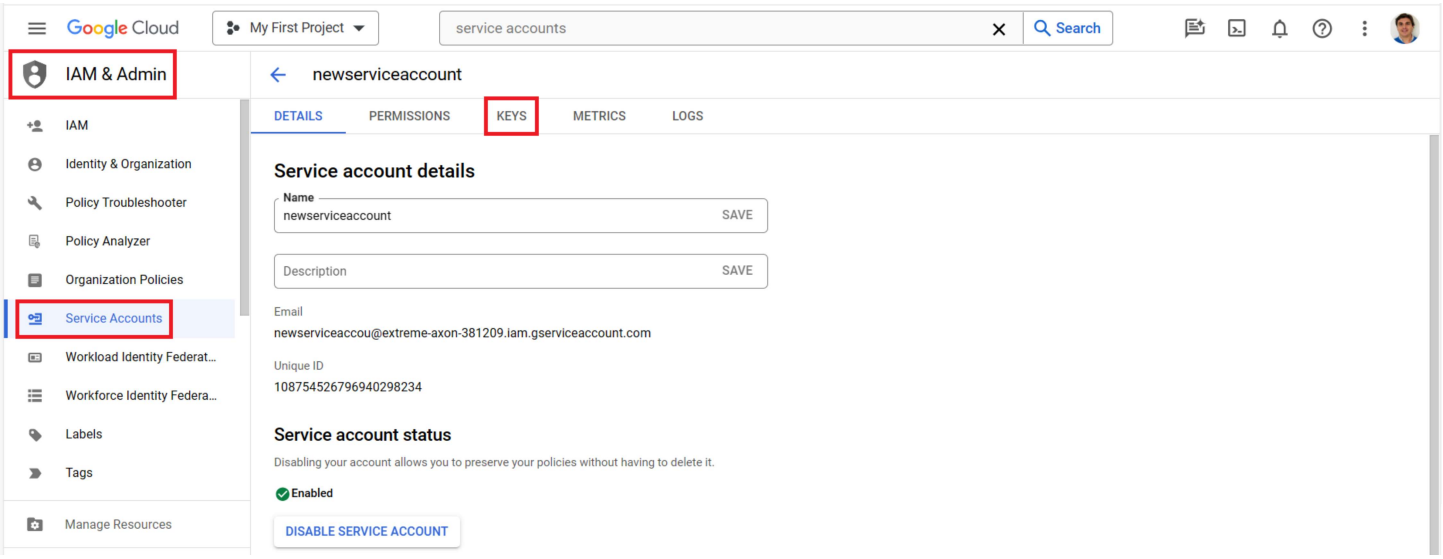
We also set the Service account admins role, we select our Projec ID role

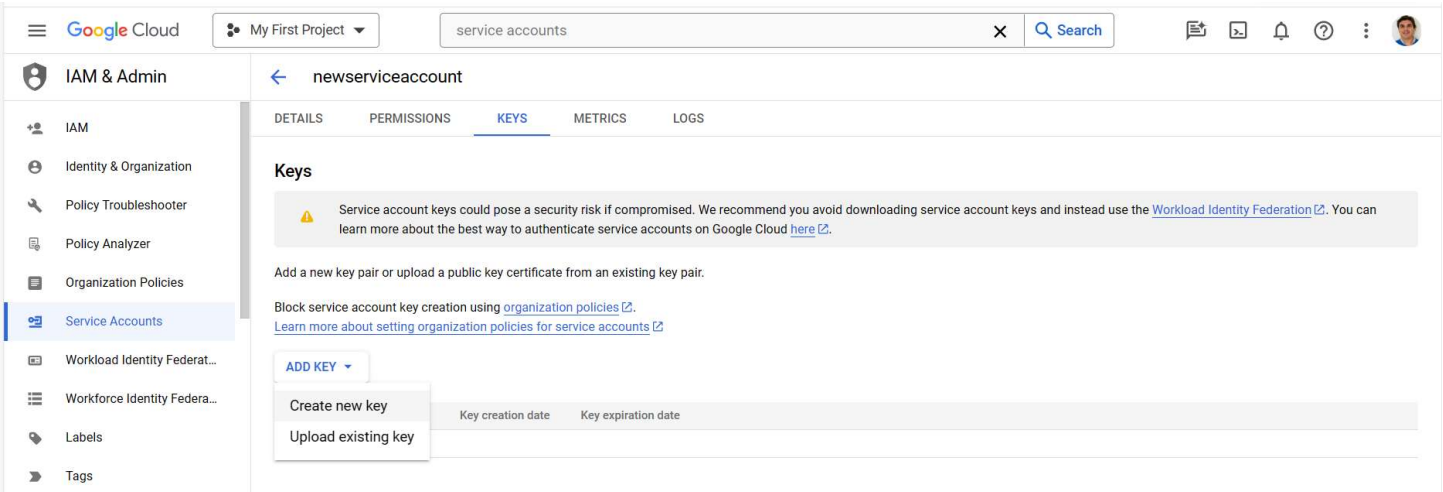


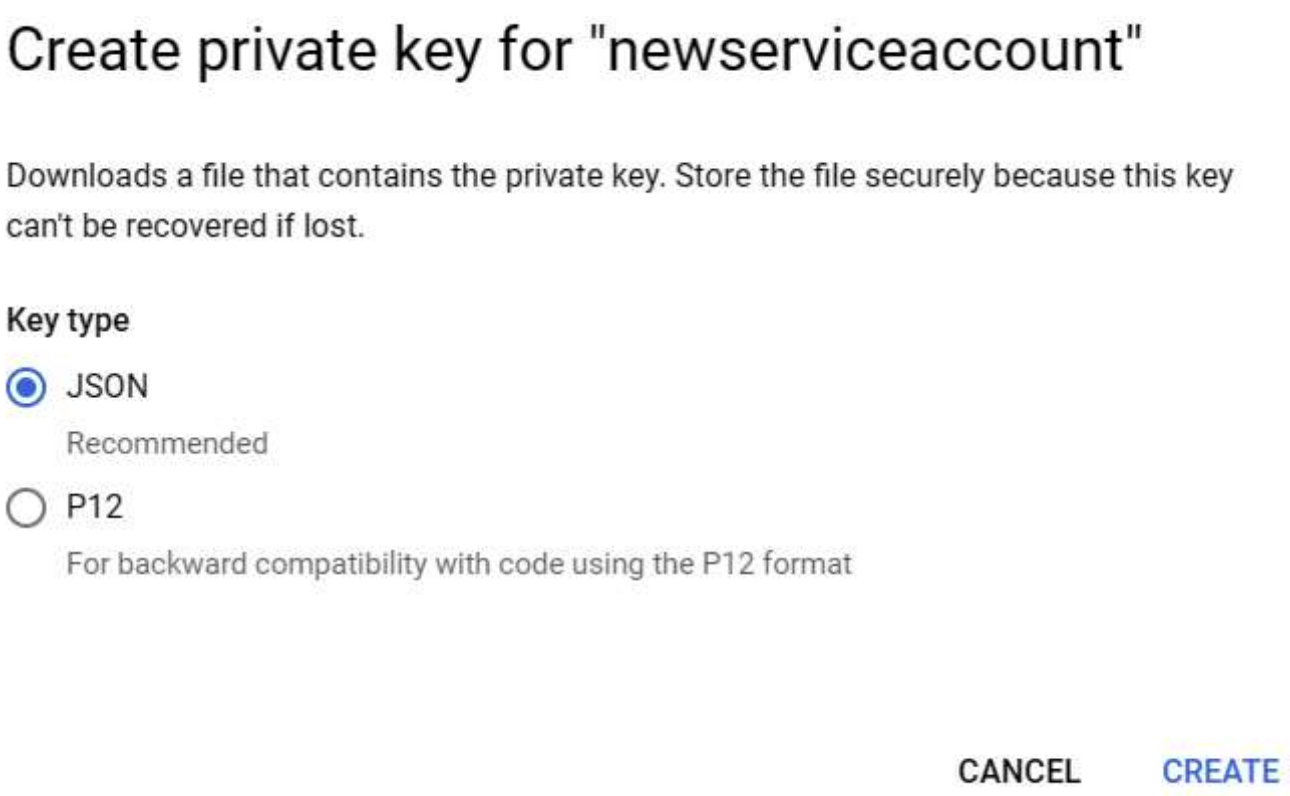


Then We create a new Key to copy it in our Github Secrets and store it with the variable name **GOOGLE_CLOUD_CREDENTIALS**

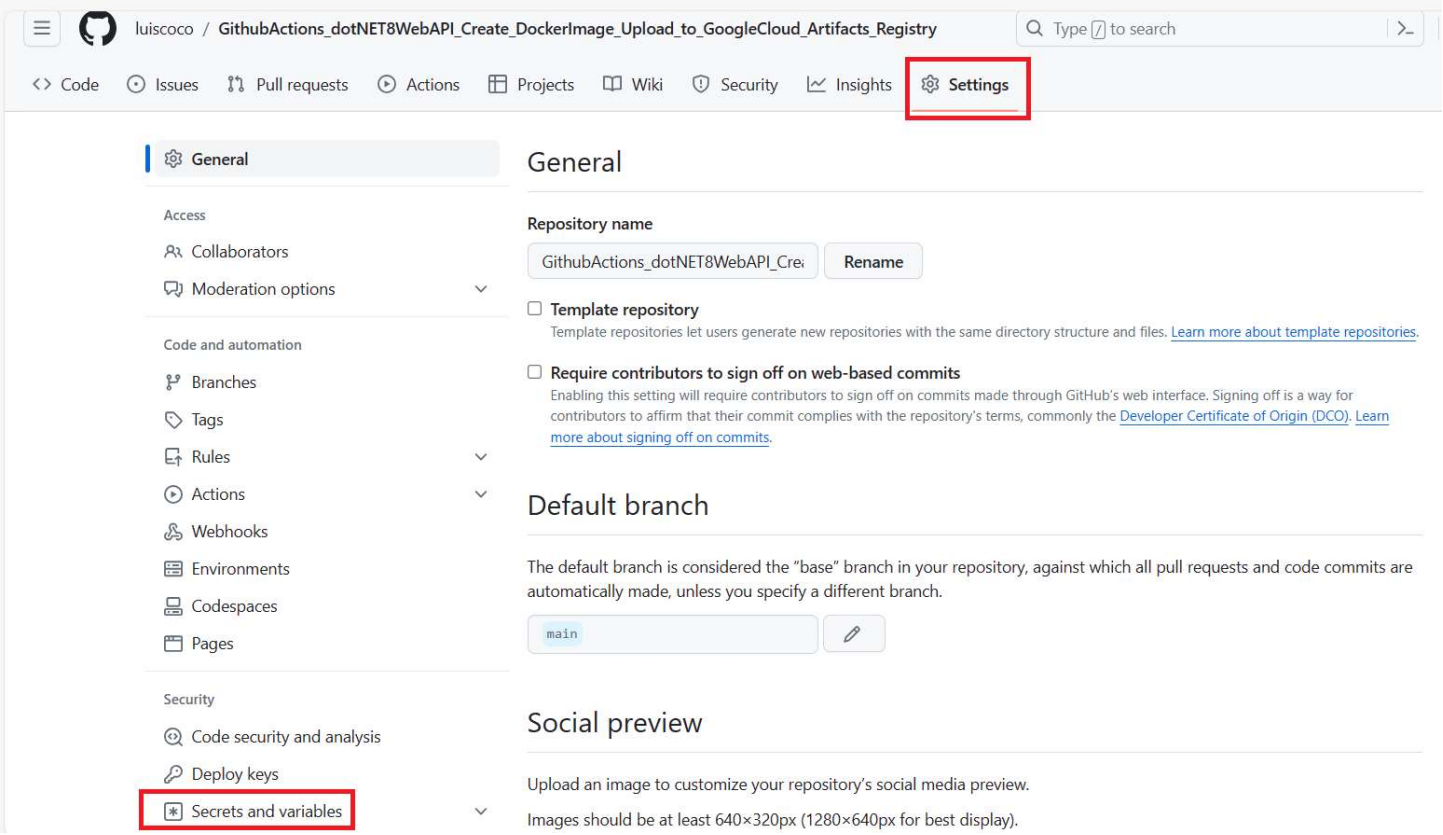
We click on the Key tab and we create a new JSON key file

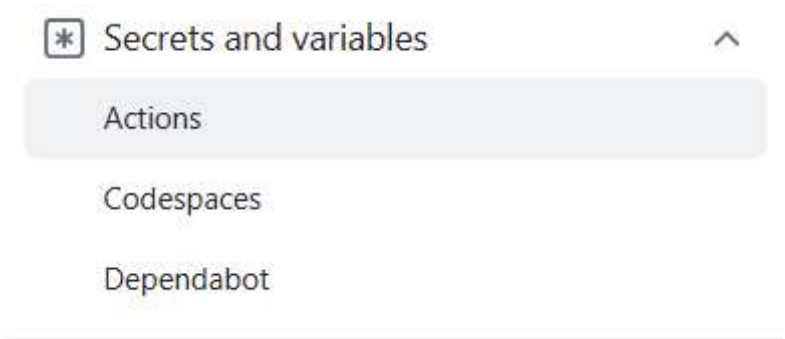


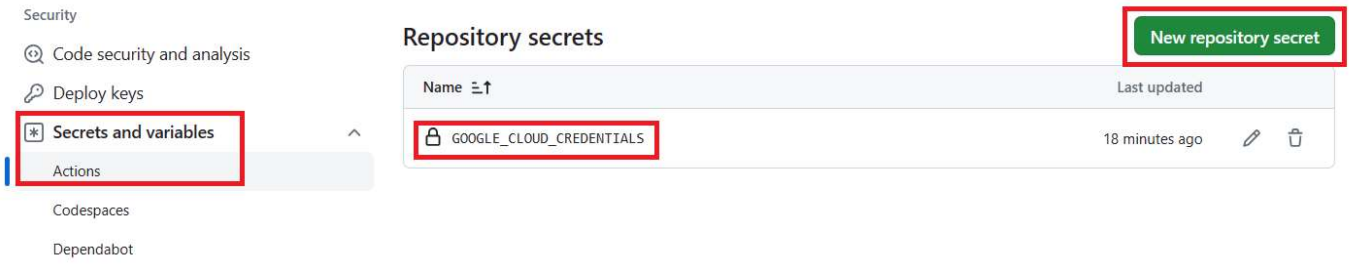




In our github repo we nagivate to the **Settings->Secrets option->Actions**







Below is the complete **main.yml** file based on your initial workflow, with the added functionality to check for the existence of a Google Cloud Artifact Registry repository and create it if it does not exist.

This workflow includes steps for authentication, checking and creating the repository, building and pushing the Docker image, and verifying the push.

```yaml
name: Build and Push Docker Image

on:
  push:
    branches:
      - main

env:
  PROJECT_ID: extreme-axon-381209
  IMAGE_NAME: my-dotnetwebapi
  REPOSITORY: myfirstrepo
  TAG: latest

jobs:
  build-and-push:
    runs-on: ubuntu-latest

    steps:
    - name: Checkout code
      uses: actions/checkout@v4

    - name: Authenticate to Google Cloud
      uses: google-github-actions/auth@v2
      with:
        credentials_json: ${{ secrets.GOOGLE_CLOUD_CREDENTIALS }}

    - name: Configure Docker for Google Cloud Artifact Registry
```

```
    run: |
      echo '${{ secrets.GOOGLE_CLOUD_CREDENTIALS }}' | gcloud auth activate-service-account
      gcloud auth configure-docker europe-southwest1-docker.pkg.dev --quiet

  - name: Check if Google Cloud Artifact Registry repository exists
    id: check-repo
    run: |
      if gcloud artifacts repositories describe ${{ env.REPOSITORY }} --location=europe-sout
        echo "Repository exists"
      else
        echo "::set-output name=repo_exists::false"
      fi

  - name: Create Google Cloud Artifact Registry repository if it does not exist
    if: steps.check-repo.outputs.repo_exists == 'false'
    run: |
      gcloud artifacts repositories create ${{ env.REPOSITORY }} --repository-format=docker
      echo "Repository created"

  - name: Build Docker image
    run: |
      docker build -t europe-southwest1-docker.pkg.dev/${{ env.PROJECT_ID }}/${{ env.REPOSIT

  - name: Push Docker image
    run: |
      docker push europe-southwest1-docker.pkg.dev/${{ env.PROJECT_ID }}/${{ env.REPOSITORY

  - name: Verify the image was pushed
    run: |
      gcloud artifacts docker images list europe-southwest1-docker.pkg.dev/${{ env.PROJECT_I
```