MediaMarkt Hackathon - Solution to the Cloud Challenge

By Mario Valentin Ochoa Mota (@mvochoa)

All the files and instructions for this challenge are in the repository https://github.com/mvochoa/mediamarkt-hackathon-cloud.

Configurations

It is necessary to configure the working environment, the application repository to be CI/CD and the google cloud console.

Configuration of the working environment

It is necessary to make a clone of the repository https://github.com/mvochoa/mediamarkt-hackathon-cloud if you open it with the visual studio code editor and have installed the Dev Containers extension, the editor is configured with everything that is required to work.

In case you do not use the editor and extension, the binaries required are:

- Terraform: `=> v1.4.2`
- Google Cloud CLI (gcloud): `The latest version`

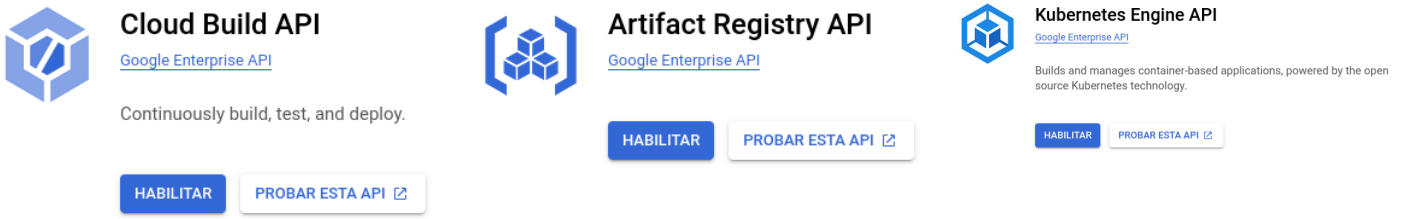
Once the environment is configured, it is required to login to Google Cloud with the command `gcloud auth application-default login` and follow the instructions.

Application configuration

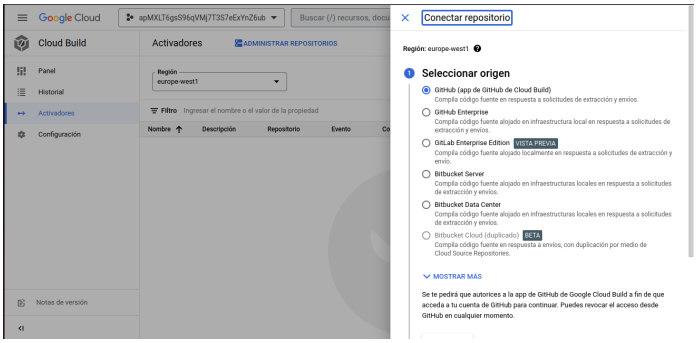
The application to which the CI/CD will be created is https://github.com/nuwe-io/mms-cloud-skeleton, as I do not have permissions in the repository, I made a fork to my github account https://github.com/mvochoa/mms-cloud-skeleton.

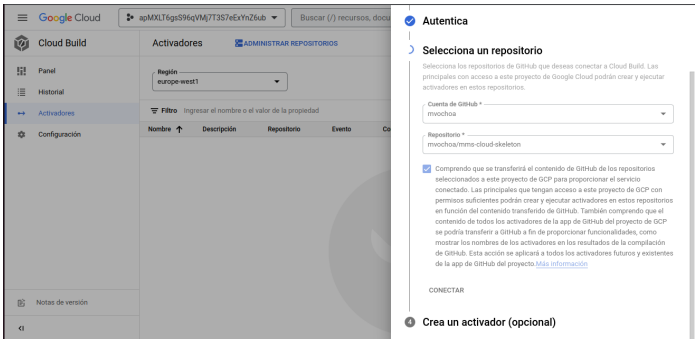
Google Cloud Configuration

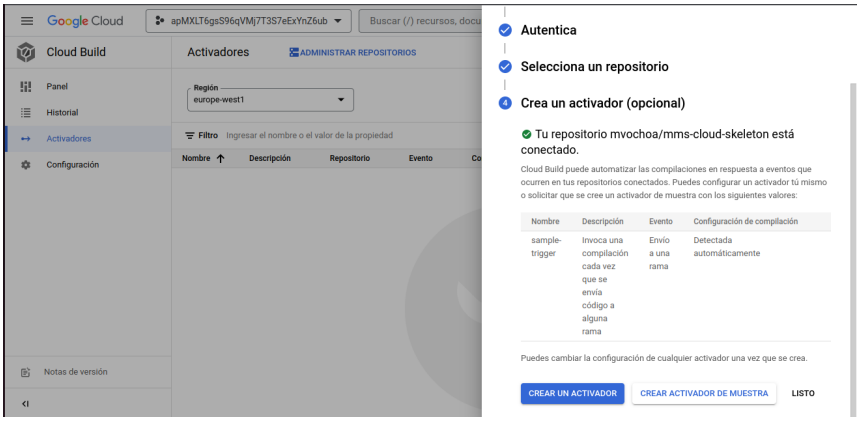
It is necessary to access the google cloud console and the project `apMXLT6gsS96qVMj7T3S7eExYnZ6ub` once inside it is necessary to enable the following APIs:



Finally it is necessary to connect my GitHub account with Cloud Build so I can access the repository, the process is as follows:







Application of terraform manifests

All the creation and configuration of Google Cloud resources is done with terraform files, also the Kubernetes manifest (deployments, services, etc) is done with the terraform provider for kubernetes.

First you need to create a bucket to store the terraform `.tfstate` files by executing the following commands:

```
$ cd src/init/
$ terraform init
$ terraform apply
...
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.

  Enter a value: yes

...

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

bucket = "08fe34899c02123a-tfstate"
```

The resources created are:

Name	Type	Description
google_storage_bucket.default	resource	Bucket where the terraform `.tfstate` files will be stored
random_id.bucket_prefix	resource	It is used to generate a random number for the bucket name

Now once the bucket is created we can apply the rest of the manifests with the commands:

```
$ cd ..
$ terraform init
$ terraform apply
...
Do you want to perform these actions?
  Terraform will perform the actions described above.
  Only 'yes' will be accepted to approve.
```

```
  Enter a value: yes

...

Apply complete! Resources: 7 added, 0 changed, 0 destroyed.

Outputs:

service = "http://130.211.57.127"
```

The resources created are:

Name	Type	Description
google_artifact_registry_repository.mediamarkt-cloud	resource	Repository where the docker images built by the cloud build will be stored
google_cloudbuild_trigger.mms-cloud-skeleton	resource	The Cloud Build that is executed each time the `main` branch of the repository is pushed
google_container_cluster.mediamarkt-cloud	resource	
google_container_node_pool.mediamarkt-cloud	resource	
kubernetes_config_map_v1.script	resource	Kubernetes manifest containing the script to force the change of the image when a new one is uploaded to the artifact registry
kubernetes_deployment_v1.mms-cloud-skeleton	resource	Kubernetes Manifest for application deployment will only generate a 1 pod
kubernetes_service_v1.mms-cloud-skeleton	resource	Kubernetes Manifest for the service that will generate a Load Balancer with a Public IP so that the application can be accessed.
google_client_config.default	data source	
google_project.project	data source	

(IMPORTANT) Problem with permissions in Google Cloud

Due to lack of permissions for the google cloud IAM with the user assigned to me by the Nuwe team, the cloud build cannot update the deployment image in kubernetes. The resources that were not applied, are commented in the `src/cloudbuild.tf` file.

The resources that cannot be created are the assignment of the kubernetes role to the cloud build service account and the commands to make the image change in the kubernetes deployment:

```
resource "google_project_iam_member" "cloudbuild-kubernetes" {
  project = local.project
  role    = "roles/container.developer"
  member  = "serviceAccount:${data.google_project.project.number}@cloudbuild.gserviceaccount.com"
}

resource "google_cloudbuild_trigger" "mms-cloud-skeleton" {
  ...

  build {
    ...

    step {
      name       = "gcr.io/google.com/cloudsdktool/cloud-sdk"
      entrypoint = "gcloud"
      args       = ["container", "clusters", "get-credentials", local.cluster_name,
"--region=${local.region}", "--project=${local.project}"]
    }

    step {
      name       = "gcr.io/google.com/cloudsdktool/cloud-sdk"
      entrypoint = "kubectl"
      args       = ["set", "image", "deployment/${local.repository_name}",
"${local.repository_name}=${local.docker_image_name}"]
    }
  }
}
```

The commands that the cloud build will not be able to execute are:

```
$ gcloud container clusters get-credentials mediamarkt-cloud --region=europe-west1
--project=apmxlt6gss96qvmj7t3s7eexynz6ub
$ kubectl set image deployment/mms-cloud-skeleton
mms-cloud-skeleton=europe-west1-docker.pkg.dev/apmxlt6gss96qvmj7t3s7eexynz6ub/mediamarkt-cloud/mm
s-cloud-skeleton:$SHORT_SHA
```

Solution for the lack of permissions

I found this solution by reviewing the service account of the kubernetes nodes, I found that you have the permission to access the cluster, so internally in the cluster you can execute the commands that can not be executed in the cloud build for lack of permissions.

The solution was to add a container to the kubernetes deployment of the application, which is responsible for checking regularly (every 5 min) if a new image was uploaded to the artifact registry and when it detects a new image updates the deployment with that new image. The script that runs the container is the following:

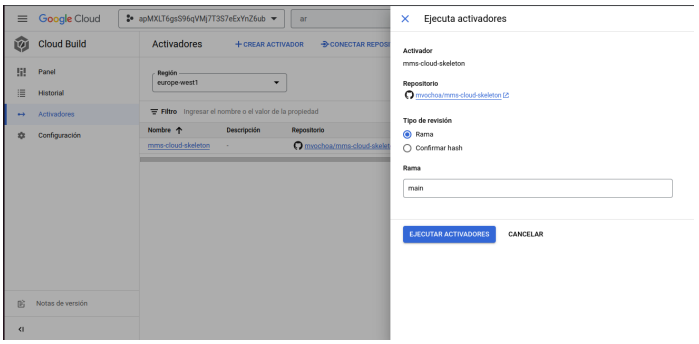
```
#!/bin/bash

function GetTag() {
  TAG=$(gcloud artifacts docker images list --include-tags
europe-west1-docker.pkg.dev/apmxlt6gss96qvmj7t3s7eexynz6ub/mediamarkt-cloud/mms-cloud-skeleton |
grep latest | awk '{printf $3$4}')
  TAG="${TAG/latest/}"
  TAG="${TAG/,/}"
  printf $TAG
}

gcloud container clusters get-credentials mediamarkt-cloud --region=europe-west1
--project=apmxlt6gss96qvmj7t3s7eexynz6ub

IMAGE=$(kubectl get deployment mms-cloud-skeleton
-o=jsonpath='{.spec.template.spec.containers[0].image}')

while true
do
  TAG=$(GetTag)

  if [[ "$IMAGE" !=
"europe-west1-docker.pkg.dev/apmxlt6gss96qvmj7t3s7eexynz6ub/mediamarkt-cloud/mms-cloud-skeleton:$
TAG" ]];
  then
    kubectl set image deployment/mms-cloud-skeleton
mms-cloud-skeleton=europe-west1-docker.pkg.dev/apmxlt6gss96qvmj7t3s7eexynz6ub/mediamarkt-cloud/mm
s-cloud-skeleton:$TAG
  fi
```

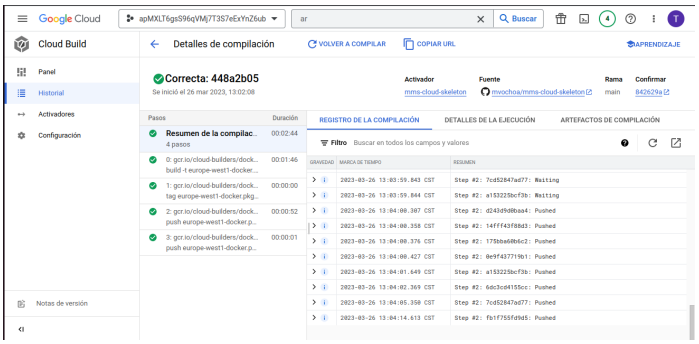
```
  sleep 300
done
```

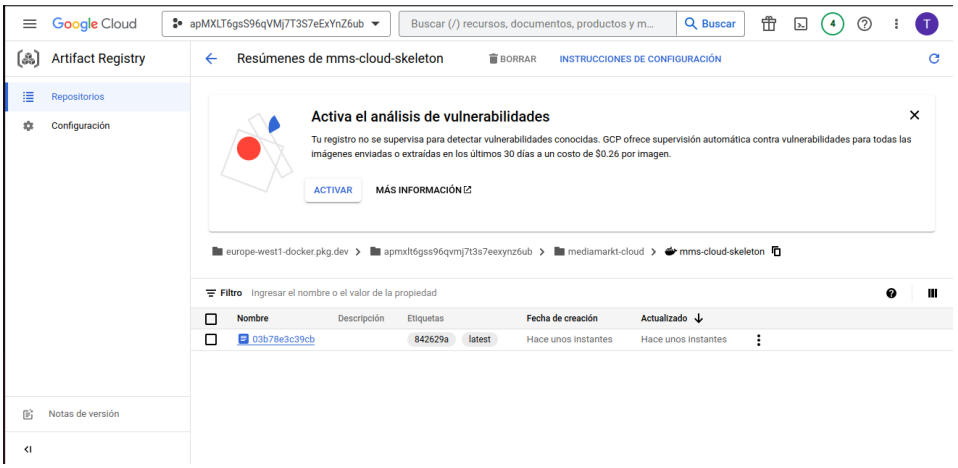
This way the CI/CD is complete 🎉🎉🎉. I know it's not a very clean way to solve the problem but since I couldn't assign permissions correctly, it was the only way I found to get the CI/CD to work.

Testing the operation of the cloud build

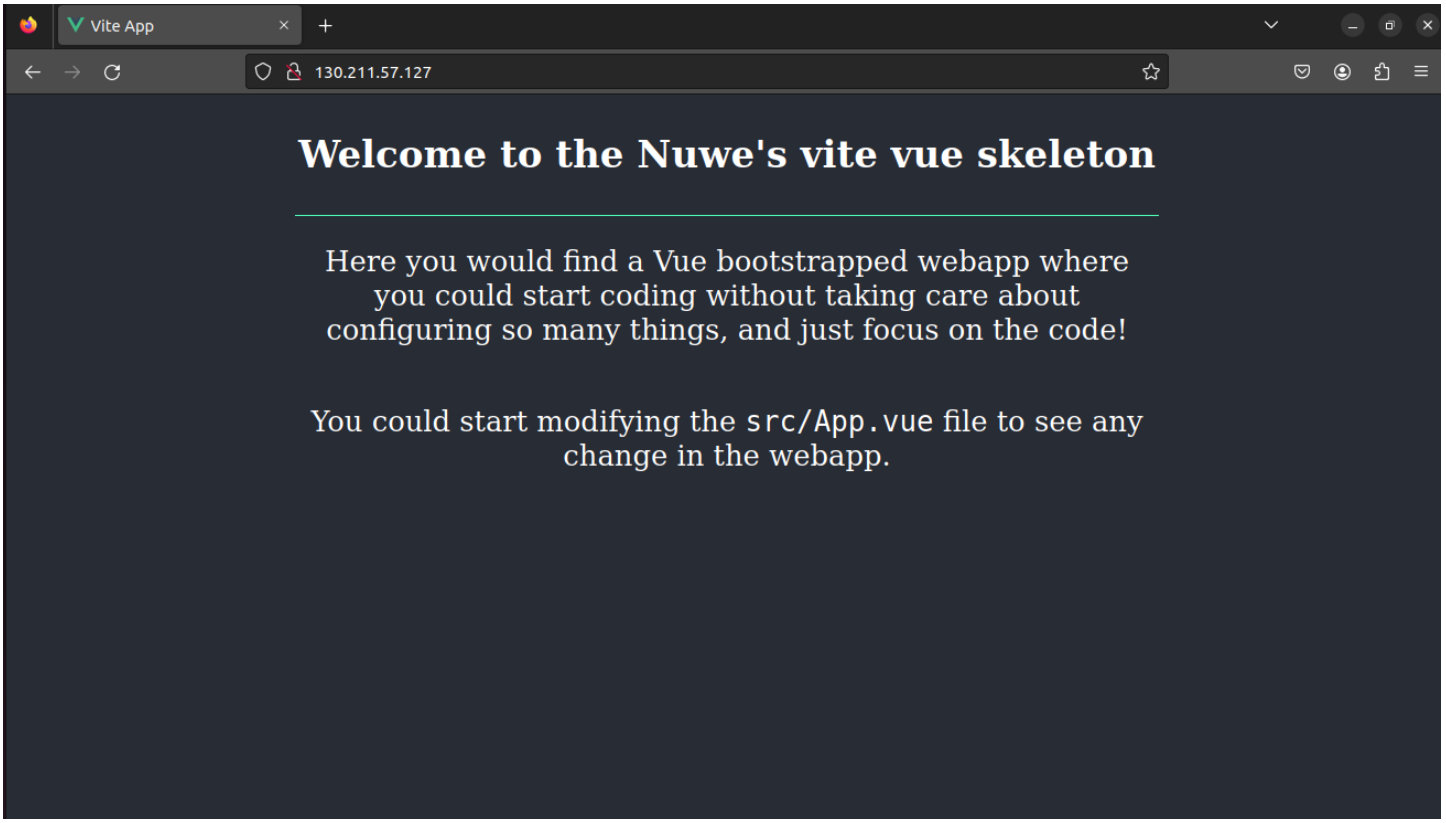
A manual execution of the cloud build is going to be done to see the application as it is at the moment.



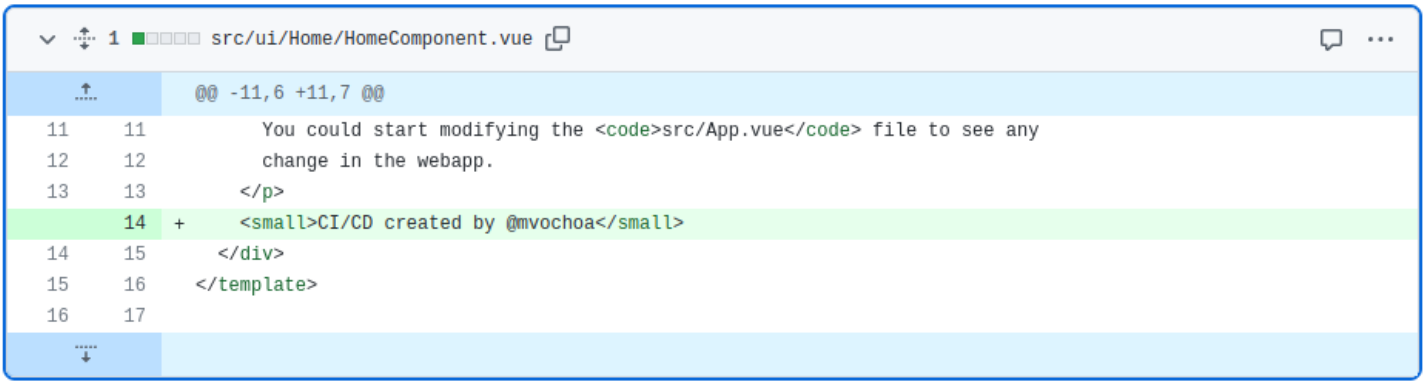


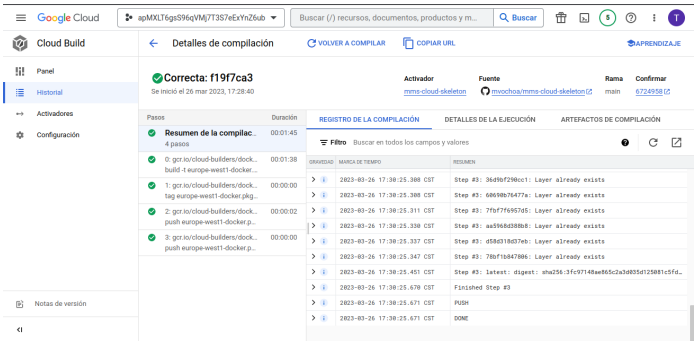


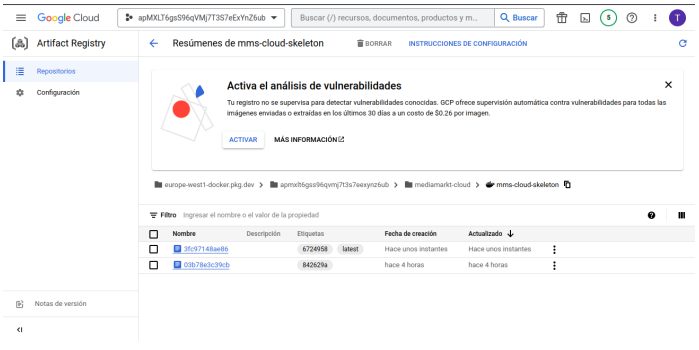
A few minutes later, the web can be viewed on the IP 130.211.57.127 of the kubernetes service.

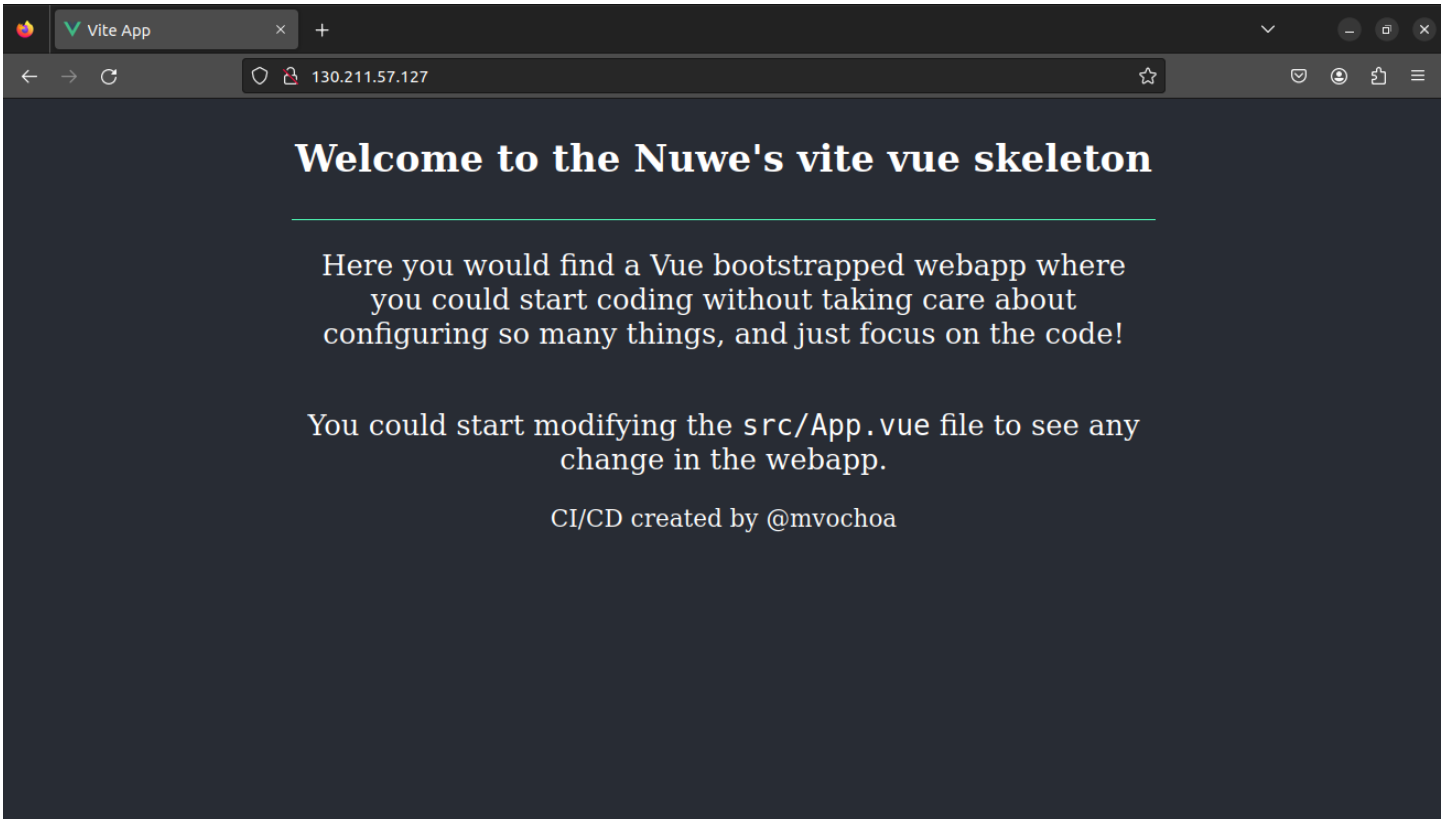


Now we test uploading changes to the repository to see if the whole pipeline works correctly, the change that was added was:









Docker Compose file

To the github repository https://github.com/mvochoa/mms-cloud-skeleton the `docker-compose.yaml` file was added and the `Dockerfile` was improved.

The `docker-compose.yaml` file was optimized so that developers can work faster and more comfortably since a development server is running and the `node_modules` are stored in volume.

```
version: '3.8'
services:
  app:
    build:
      context: .
      target: builder
    ports:
      - 3000:3000
    volumes:
      - .:/app
      - node_modules:/app/node_modules
    command: sh -c 'yarn install && yarn run dev'

volumes:
  node_modules:
```

The `Dockerfile` was also adjusted to make the final image lighter and with better performance.

19 Dockerfile

```
@@ -1,11 +1,11 @@
- FROM node:alpine
-
- ADD package.json /app/package.json
+ FROM node:alpine AS builder

  WORKDIR /app

  RUN yarn add vite

+ ADD package.json /app/package.json
+
  # Installing packages
  RUN yarn install

@@ -16,4 +16,15 @@ RUN yarn build-only

  EXPOSE 3000

- CMD [ "yarn", "preview" ]
+ CMD [ "yarn", "dev" ]
+
+ FROM nginx:stable-alpine
+
+ ENV NGINX_PORT=3000
+
+ COPY --from=builder /app/dist /usr/share/nginx/html
+
+ RUN mkdir /etc/nginx/templates/ \
+     && sed 's/listen       80/listen       ${NGINX_PORT}/g' /etc/nginx/conf.d/default.conf > /etc/nginx/templates/default.conf.template
+
+ EXPOSE 3000
```

Questions about roles

DevOps

For the management of the kubernetes cluster in Google Cloud you have to assign to the group, the role of **Kubernetes Engine Developer** this role allows the management of the different services that the cluster has.

Finances

To manage the billing in Google Cloud, the **Billing Account Viewer** role must be assigned to the group, this role only allows the management of billing.