Solution Design

I have created this architecture to allow developers generate new web's versions easily just pushing new tag to master branch. The code is this repo's links: https://github.com/lucabem/mms-cloud-skeleton

The componentes are:

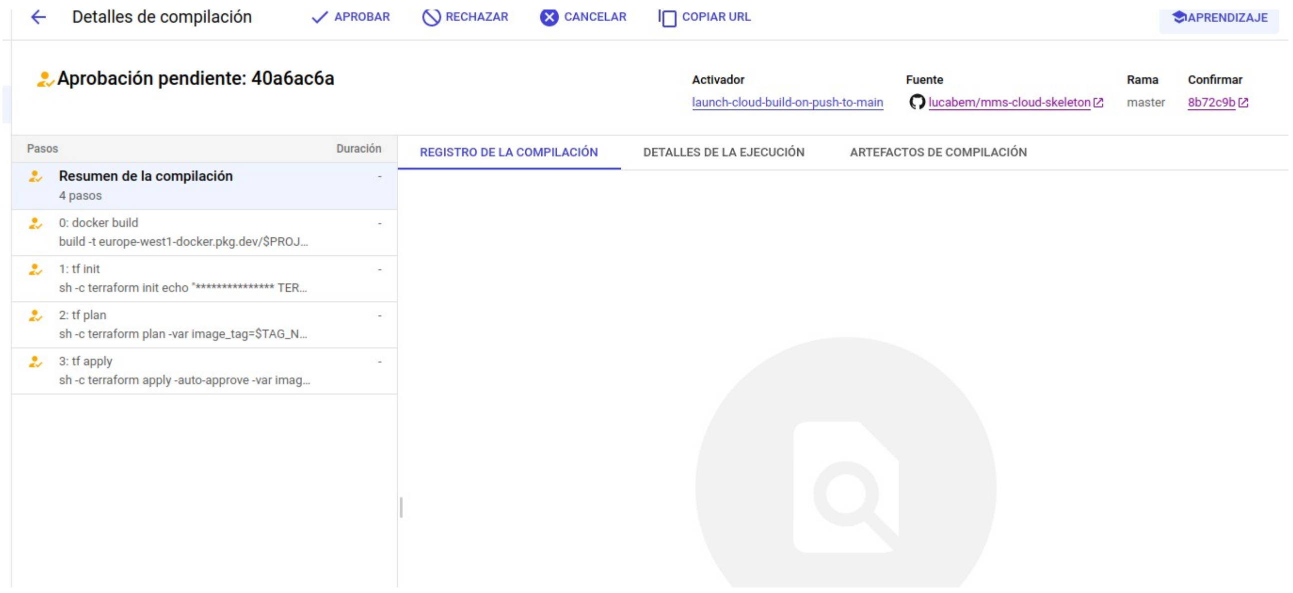
- Cloud Storage Bucket: place where we are going to save logs from CloudBuild (I have use it due to constraint between ServiceAccount attribute in trigger resource and logs).
  In addition, I have created (via CLI) a second bucket where we would save Terraform State, allowing us to execute terraform from all places but keeping the state on a common place

  ```
  gsutil mb gs://${PROJECT_ID}-tfstate
  gsutil versioning set on gs://${PROJECT_ID}-tfstate
  ```

- CloudBuild Triggers: we create a trigger that will generate an docker image when new tag is created on repo, using master branch. This trigger must be approve following best-practices in CI/CD because we just have one branch (master)

- Google Container Cluster: Kubernetes Cluster that will run docker service for deploying web. In a real environment, cluster must be fine-tunning and get opened just needed ports

Each time we publish a new tag on master's branch, on CloudBuild we will see an pending approval as we could see in the image:

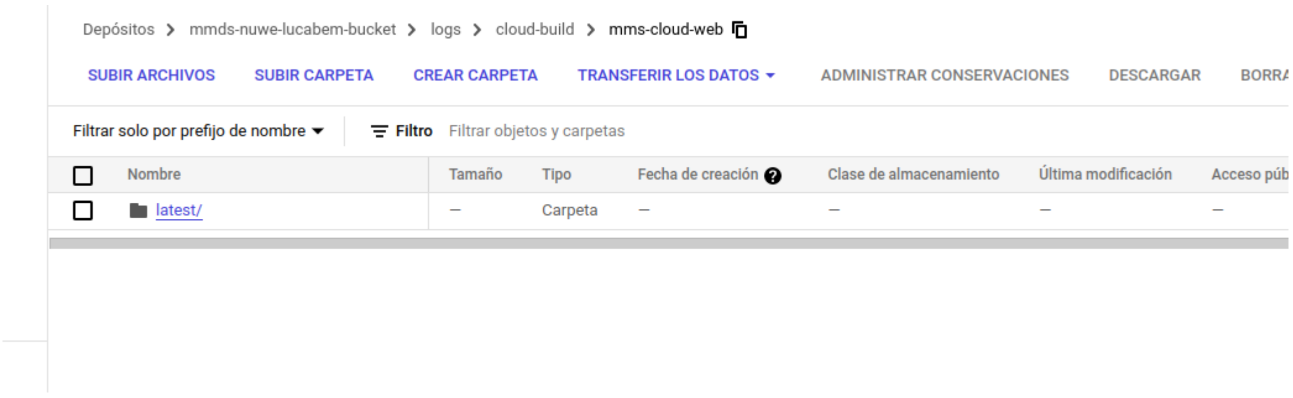


Aprobación pendiente: 40a6ac6a

Activador	Fuente	Rama	Confirmar
launch-cloud-build-on-push-to-main	lucabem/mms-cloud-skeleton	master	8b72c9b

Pasos	Duración
Resumen de la compilación 4 pasos	-
0: docker build build -t europe-west1-docker.pkg.dev/$PROJ...	-
1: tf init sh -c terraform init echo "*************** TER...	-
2: tf plan sh -c terraform plan -var image_tag=$TAG_N...	-
3: tf apply sh -c terraform apply -auto-approve -var imag...	-

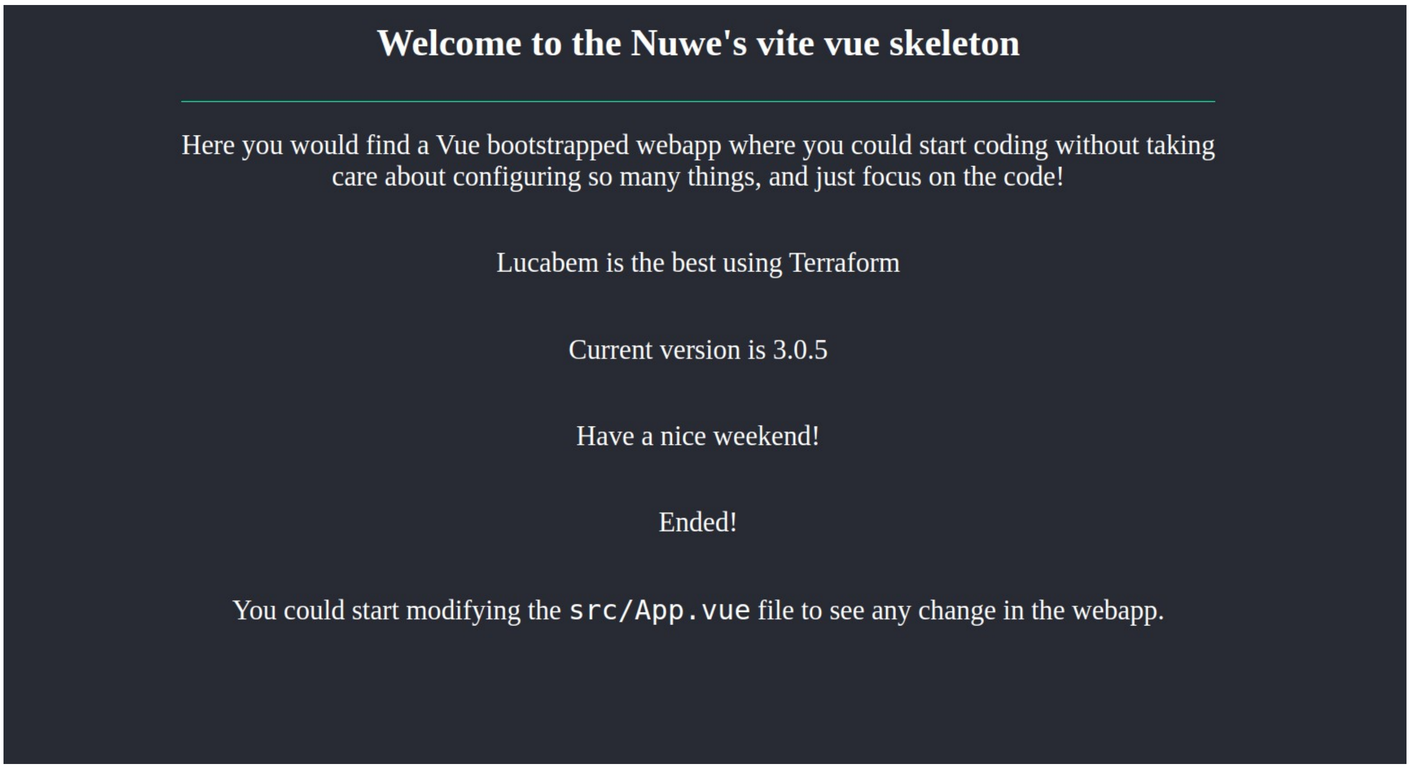
This CodeBuild will load TF-State from bucket and will generate a docker image with tag latest and tag's value from Github repo using env var such as $TAG_NAME. Each tag will generates a folder were logs are saved (only will generate logs if serviceAccount is None on google_cloudbuild_trigger resource)


Depósitos > mmds-nuwe-lucabem-bucket > logs > cloud-build > mms-cloud-web
SUBIR ARCHIVOS
SUBIR CARPETA
CREAR CARPETA
TRANSFERIR LOS DATOS
ADMINISTRAR CONSERVACIONES
DESCARGAR
BORRA
Filtrar solo por prefijo de nombre
Filtro
Filtrar objetos y carpetas

Nombre	Tamaño	Tipo	Fecha de creación	Clase de almacenamiento	Última modificación	Acceso púb
latest/	–	Carpeta	–	–	–	–

Next step is deploying new kubernetes resource via Terraform using var tag_name that will get value from env var $TAG_NAME. This will force deployment to restart his replica and forcing replica to download last image due to propertiy imagePullPolicy setted to Always.

We expose a LoadBalancer's Service to allow reach it via public IP that generates the cluster:


Welcome to the Nuwe's vite vue skeleton

Here you would find a Vue bootstrapped webapp where you could start coding without taking care about configuring so many things, and just focus on the code!

Lucabem is the best using Terraform

Current version is 3.0.5

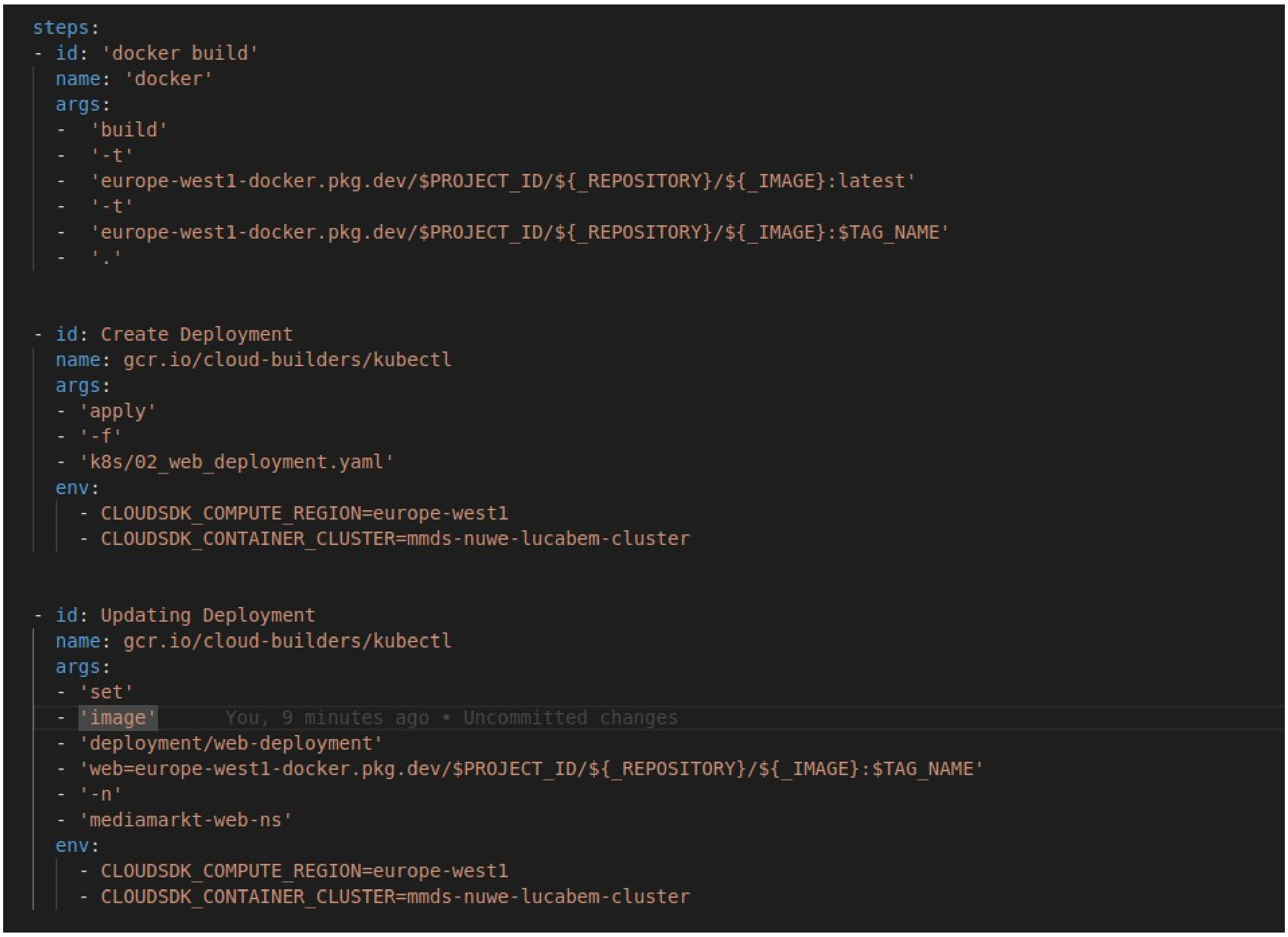
Have a nice weekend!

Ended!

You could start modifying the `src/App.vue` file to see any change in the webapp.

In case more operations are performed on the account, I would not like to execute Terraform commands inside CodeBuild in order to avoid problems with deletion/creation of new resources, we should separate repo in infra-repo and web-repo.

In this case, instead of using Terraform resources, we would use the kubectl client inside CloudBuild adding the YAMLs in the K8S folder and leaving the cloud-build.yaml as follows to update the image with the corresponding tag:



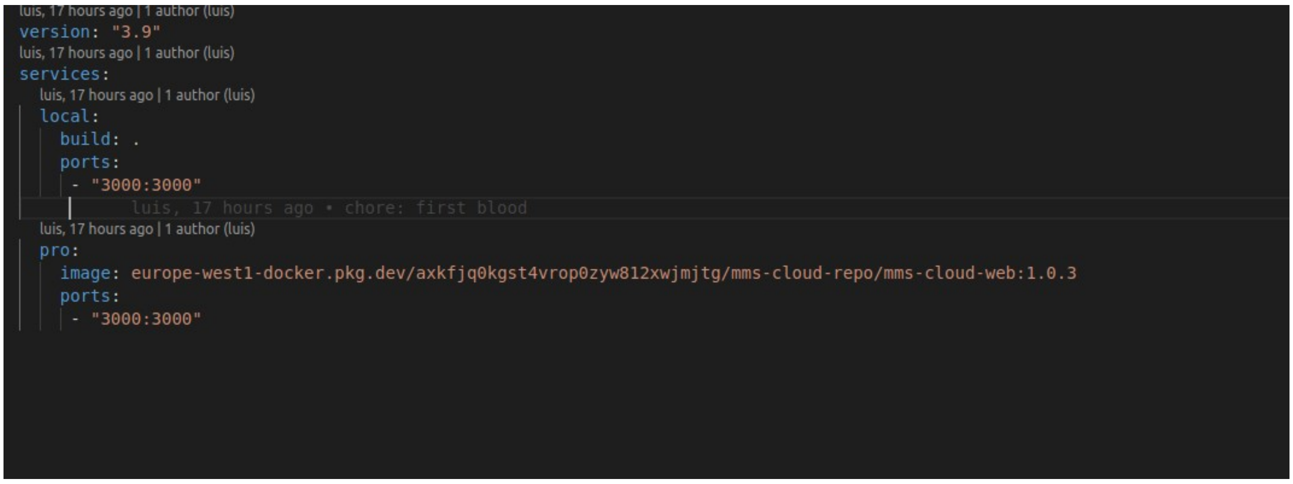
```yaml
steps:
- id: 'docker build'
  name: 'docker'
  args:
  -  'build'
  -  '-t'
  -  'europe-west1-docker.pkg.dev/$PROJECT_ID/${_REPOSITORY}/${_IMAGE}:latest'
  -  '-t'
  -  'europe-west1-docker.pkg.dev/$PROJECT_ID/${_REPOSITORY}/${_IMAGE}:$TAG_NAME'
  -  '.'


- id: Create Deployment
  name: gcr.io/cloud-builders/kubectl
  args:
  - 'apply'
  - '-f'
  - 'k8s/02_web_deployment.yaml'
  env:
    - CLOUDSDK_COMPUTE_REGION=europe-west1
    - CLOUDSDK_CONTAINER_CLUSTER=mmds-nuwe-lucabem-cluster


- id: Updating Deployment
  name: gcr.io/cloud-builders/kubectl
  args:
  - 'set'
  - 'image'      You, 9 minutes ago • Uncommitted changes
  - 'deployment/web-deployment'
  - 'web=europe-west1-docker.pkg.dev/$PROJECT_ID/${_REPOSITORY}/${_IMAGE}:$TAG_NAME'
  - '-n'
  - 'mediamarkt-web-ns'
  env:
    - CLOUDSDK_COMPUTE_REGION=europe-west1
    - CLOUDSDK_CONTAINER_CLUSTER=mmds-nuwe-lucabem-cluster
```

Docker-Compose

The Docker-Compose allow us to execute docker image in a simple way. I have created a docker-compose yaml that allow users to execute current local version of the repo and other service that uses repo from Artifact Registry in order to check again if everythin works.



```yaml
luis, 17 hours ago | 1 author (luis)
version: "3.9"
luis, 17 hours ago | 1 author (luis)
services:
    luis, 17 hours ago | 1 author (luis)
  local:
    build: .
    ports:
      - "3000:3000"
              luis, 17 hours ago • chore: first blood
    luis, 17 hours ago | 1 author (luis)
  pro:
    image: europe-west1-docker.pkg.dev/axkfjq0kgst4vrop0zyw812xwjmjtg/mms-cloud-repo/mms-cloud-web:1.0.3
    ports:
      - "3000:3000"
```

Just launch it using docker-compose up local or docker-compose up pro

IAM Question

For the DevOps team, who will be responsible for creating clusters in Kubernetes, the appropriate role would be the Kubernetes Engine Cluster Admin role. This role allows the team to create and manage Kubernetes clusters, including managing the nodes, pods, and services within the cluster.

To assign the Kubernetes Engine Cluster Admin role to the DevOps team, follow these steps in the IAM GCP Console:

1. Navigate to the IAM & Admin section of the GCP Console
2. Select the project in which the Kubernetes clusters will be created
3. Click on the "Add" button to add a new member to the project
4. Enter the email address or group of the DevOps team members
5. Select the "Kubernetes Engine Cluster Admin" role from the dropdown list
6. Click on the "Save" button to add the role to the DevOps team

For the Finance team, who will be responsible for managing billing in GCP, the appropriate role would be the Billing Account Administrator role. This role allows the team to manage the billing account, including creating and managing budgets, setting up billing alerts, and managing payment methods.

To assign the Billing Account Administrator role to the Finance team, follow these steps in the IAM GCP Console:

1. Navigate to the Billing section of the GCP Console
2. Click on the "Go to linked billing account" button
3. Click on the "Add member" button to add a new member to the billing account
4. Enter the email address or group of the Finance team members
5. Select the "Billing Account Administrator" role from the dropdown list
6. Click on the "Save" button to add the role to the Finance team