MediaMarkt Cloud Engineering Challenge
By deft241 (Aarón León)

Cloud Build/Artifact to generate the container (150 points).

CI/CD workflow that runs when there are changes ("push") to the "main" branch. The job consists of a series of steps that run in sequence:

- Checkout: Downloads the source code from the repository.

```yaml
- name: Checkout
  uses: actions/checkout@v3
```

- Google Cloud authentication: Authenticates the credentials to connect to Google Cloud Platform

```yaml
- name: Google Cloud authentication.
  uses: google-github-actions/auth@v1
  with:
    credentials_json: ${{ secrets.GCP_SA_KEY }}
```

- Set up Cloud SDK

```yaml
- name: Set up Cloud SDK
  uses: google-github-actions/setup-gcloud@v1
```

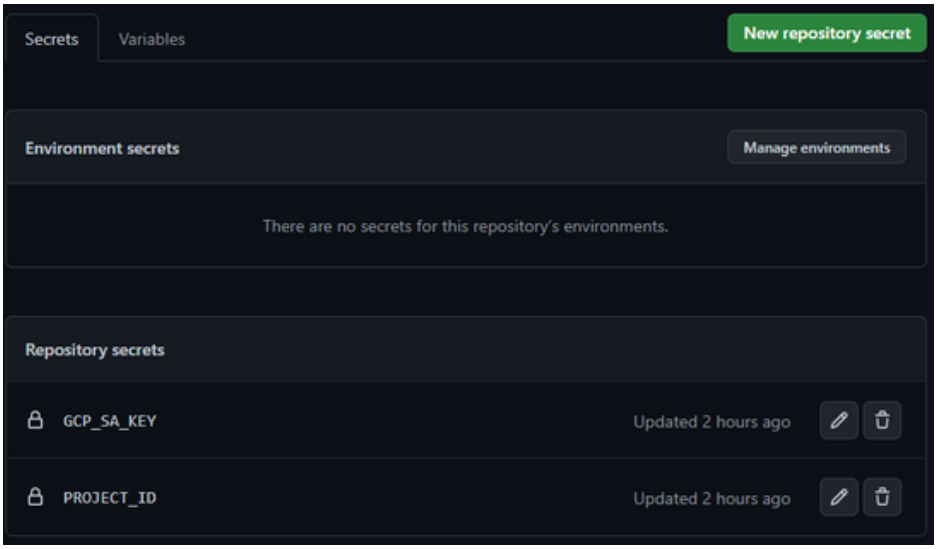
- Set Cloud SDK in docker

```yaml
- name: Set Cloud SDK in docker.
  env:
    ARTIFACT_REGISTRY_ENDPOINT: ${{ env.ARTIFACT_REGISTRY_ENDPOINT }}
  run: |
    gcloud auth configure-docker $ARTIFACT_REGISTRY_ENDPOINT
```

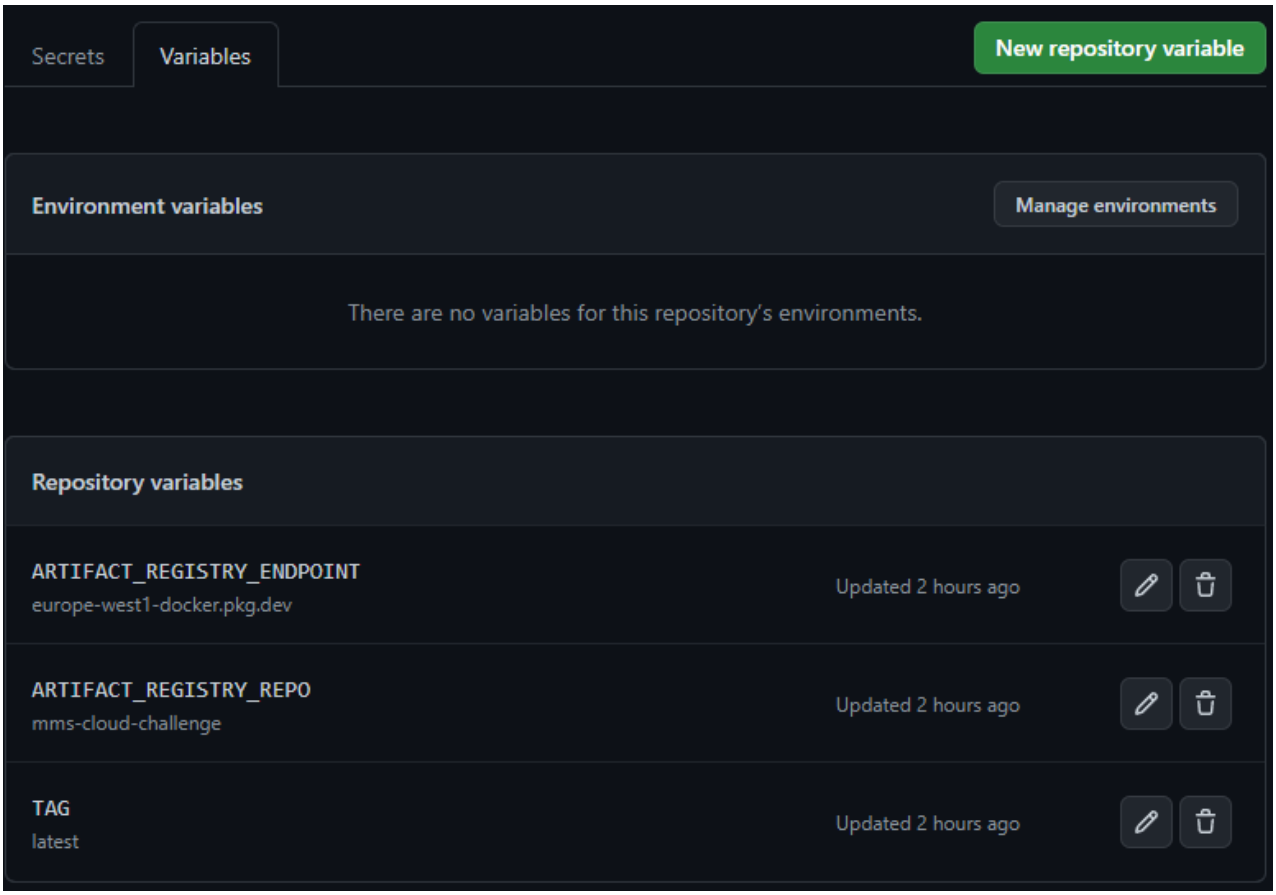
- Build docker image and push it: Builds a Docker image and uploads it to the Google Cloud Artifact Registry.

```yaml
- name: Build docker image and push it.
  env:
    ARTIFACT_REGISTRY_REPO: ${{ env.ARTIFACT_REGISTRY_REPO }}
    TAG: ${{ env.TAG }}
    PROJECT_ID: ${{ env.PROJECT_ID }}
    ARTIFACT_REGISTRY_ENDPOINT: ${{ env.ARTIFACT_REGISTRY_ENDPOINT }}
  run: |
    docker build -t $ARTIFACT_REGISTRY_ENDPOINT/$PROJECT_ID/$ARTIFACT_REGISTRY_REPO/$DOCKER_IMAGE_NAME:$TAG .
    docker push $ARTIFACT_REGISTRY_ENDPOINT/$PROJECT_ID/$ARTIFACT_REGISTRY_REPO/$DOCKER_IMAGE_NAME:$TAG
```

Defining Secrets and Variables on GitHub actions :



GCP_SA_KEY, key from Google Cloud Platform.
PROJECT_ID, the id of the project



ARTIFACT_REGISTRY_ENDPOINT, url to container registry
ARTIFACT_REGISTRY_REPO, repository name
TAG, version name

Generation of the Docker Composer YAML (150 points).

The service uses the specified Docker image "europe-west1-docker.pkg.dev/my-beautiful-project/my-beautiful-repo/mms-cloud-skeleton:latest" from a container registry. The service exposes port **3000** on the host machine, which is mapped to port **3000** on the container.

```yaml
version: "3"
services:
  cloud-challenge:
    image: europe-west1-docker.pkg.dev/my-beautiful-project/my-beautiful-repo/mms-cloud-skeleton:latest
    ports:
      - "3000:3000"
```

Creation of the Terraform Files (400 points).

In the script, we deploy a Google Kubernetes Engine (GKE) cluster in the "europe-west1" region with the following configurations:

1. Sets up the required Google provider with version 4.58.0 and defines sensitive variables for project_id and credentials.
2. Creates a GKE cluster with an initial node count of 1, using the "n1-standard-1" machine type and a 20 GB disk size. It disables the issuance of client certificates for master authentication and enables network policies with Calico as the provider.
3. Configures IP allocation policy for cluster and services secondary ranges.
4. Creates a GKE node pool with 2 nodes, using the same machine type and disk size as the cluster.
5. Sets up a firewall rule for the GKE cluster, allowing traffic on ports 80 and 443 from any source IP address (0.0.0.0/0) and targeting the cluster by its name.
6. Outputs the GKE cluster endpoint and cluster CA certificate for further use.

In summary, this Terraform script automates the deployment of a GKE cluster with a configured node pool and network policies in the "europe-west1" region. It also sets up a firewall rule to allow HTTP and HTTPS traffic, and outputs the cluster endpoint and CA certificate for future use.

File link:
https://github.com/deft241/mms-cloud-skeleton/blob/a127df57801247ceec34cd25c9a0d134c3259d06/main.tf

Commands for the Deployment through TF files (kubectl) (200 points).

First, we need to create the structure, we must create a terraform file.

With "**terraform init",** will download the data and start the resources.

Through the command, "**terraform plan**", it will show us how the deployment will be and what changes will be made. To apply the changes we have to execute "**terraform apply**".

To delete the resources created by terraform, we execute "**terraform destroy**"

Solution of the IAM Role assignation (300 points).

For the DevOps team, assign the role "Kubernetes Engine Admin" (roles/container.admin) to create and manage clusters in Kubernetes.
For the Finance team, assign the role "Billing Account Administrator" (roles/billing.admin).

To apply roles in the IAM GCP Console:

In Google Cloud Console, go to the project and gon to **IAM and Admin,** there, we add a new member, and select the role we want for it.
(Kubernetes Engine Admin to DevOps team, and Billing acount administrator, for the billing team)