MediaMarkt Cloud Engineering challenge

Summary

This document resolves the *TASKS-OBJECTIVES* and explains how to deploy the *mms-cloud-skeleton* app in a *Google Cloud Platform* project in a *Google Kubernetes Engine*.

Pre-requisites

For deploying all, you need the following:

- Install the gcloud CLI
- Install terraform

Tasks-objectives

Cloud Build/Artifact to generate the container

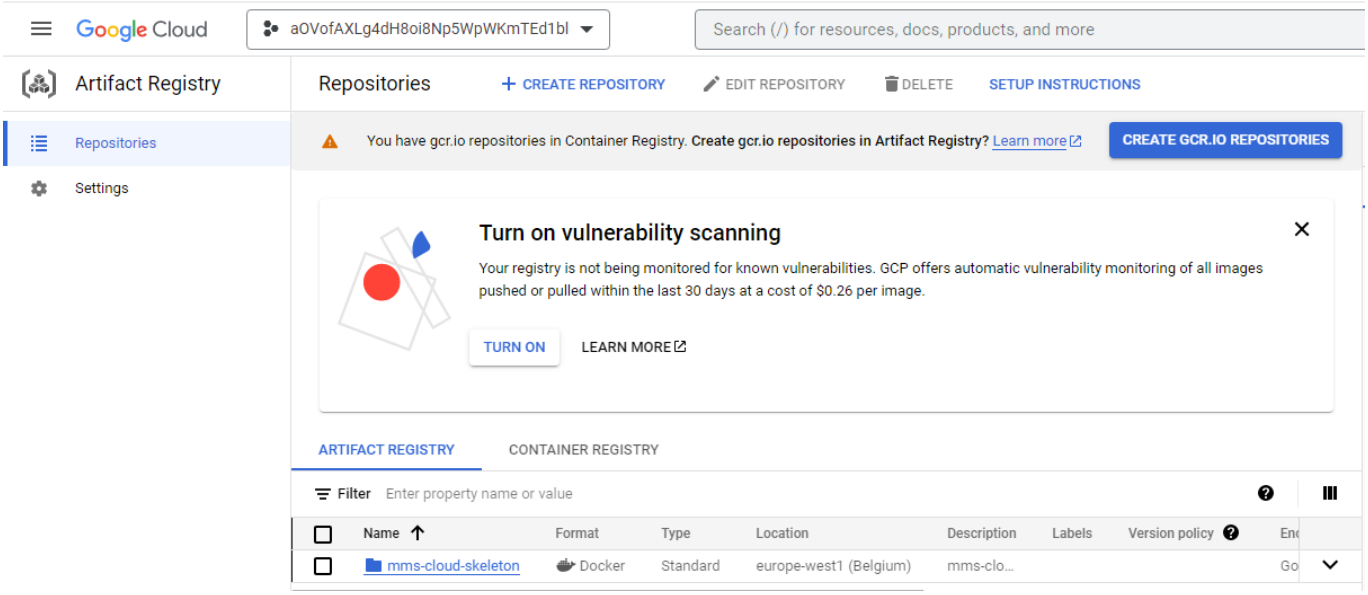
For pushing the Cloud Build/Artifact to generate the container, please execute the following lines (tested on Ubuntu 20.04) on the root directory. Please change the value of *project_id* and *location* with yours

```
location=europe-west1
project_id=aovofaxlg4dh8oi8np5wpwkmted1bl

gcloud artifacts repositories create mms-cloud-skeleton--repository-format=docker
--location=$location --description="mms-cloud-skeleton"

gcloud builds submit --region=$location --tag $location-
docker.pkg.dev/$project_id/mms-cloud-skeleton/mms-cloud-skeleton-image:latest
```

Once the *gcloud builds submit* is finished we can check it with the following command:

```
gcloud artifacts repositories list

                            ARTIFACT_REGISTRY
REPOSITORY          FORMAT  MODE                 DESCRIPTION         LOCATION
LABELS  ENCRYPTION          CREATE_TIME          UPDATE_TIME          SIZE (MB)
mms-cloud-skeleton  DOCKER  STANDARD_REPOSITORY  mms-cloud-skeleton  europe-west1
Google-managed key  2023-03-27T18:37:29  2023-03-27T18:42:59  444.649
```

Or in the *GCP* console in *Artifact Registry* menu:



Generation of the Docker Composer YAML

The Docker Compose YAML is stored in the root of the repository with the name compose.yaml and is the following:

```
services:
  mms-cloud-skeleton:
    image: "europe-west1-docker.pkg.dev/aovofaxlg4dh8oi8np5wpwkmted1bl/mms-cloud-skeleton/mms-cloud-skeleton-image"
    ports:
      - "3000:3000"
```

This file is only for local deployment because *GKE* can't use *Docker-compose* files.

Creation of the Terraform Files

In the folder terraform are stored all the files for deploying the resources and the kubectl deployment to *GCP*. Please update the terraform.tfvars file with your values. The provided file is working fine for my environment.

This terraform deploys a private *GKE* cluster with the resources needed for having connection to internet and also makes a *kubectl apply* from a terraform module.

Commands for the Deployment through TF files (kubectl)

First steps

First you have to prepare your environment and create the bucket for tfstates and the artifact registry. This step must be executed for manual and CI deployment.

```
# Environment variable for bucket name
project_id=aovofaxlg4dh8oi8np5wpwkmted1bl

# Log in GCP
```

```
gcloud auth login

# Create environment variable needed for executing from Terraform
export GOOGLE_OAUTH_ACCESS_TOKEN=$(gcloud auth print-access-token)

# Enable required services on GCP
gcloud services enable serviceusage.googleapis.com
gcloud services enable compute.googleapis.com
gcloud services enable container.googleapis.com
gcloud services enable cloudresourcemanager.googleapis.com
gcloud services enable cloudbuild.googleapis.com
gcloud services enable artifactregistry.googleapis.com

# Create storage bucket on GCP for storing TFStates
bucket_name="mms-cloud-skeleton-$project_id-tfstate"
gcloud storage buckets create gs://$bucket_name
```

Manual deployment

For manual deployment, the Terraform must be executed for the first time. This would create all the resources, including the *Cloud build* and its trigger for be prepared for CI.

```
bucket_name="mms-cloud-skeleton-$project_id-tfstate"
project_id=aovofaxlg4dh8oi8np5wpwkmted1bl
CLOUDBUILD_SA="$(gcloud projects describe $project_id \
    --format 'value(projectNumber)')@cloudbuild.gserviceaccount.com"
terraform init -backend-config="bucket=$bucket_name"
terraform plan -var="project_id=$project_id" -
var="service_account_email=$CLOUDBUILD_SA"
terraform apply -var="project_id=$project_id" -
var="service_account_email=$CLOUDBUILD_SA"
```

This may take up to 20 minutes. Now, you have to execute the *kubectl* command for deploying the image and the service to *GKE*.

```
# Environment variables
location=europe-west1
project_id=aovofaxlg4dh8oi8np5wpwkmted1bl
cluster_name=mms-cluster-gke
docker_image=$location-docker.pkg.dev/$project_id/mms-cloud-skeleton/mms-cloud-
skeleton-image

# Get credentials from the GKE for executing kubectl
gcloud container clusters get-credentials $cluster_name --region $location --
project $project_id

# Deploy the yaml to the GKE
cat ../kubernetes/deployment.yaml | sed "s|{{DOCKER_IMAGE}}|$docker_image|g" |
```

```
kubectl apply -f

# Get the public IP of the service deployed. It can take some time until EXTERNAL-IP appears
kubectl get service mms-cloud-skeleton-srv
```

Now, if you open a browser to the IP that appears on *EXTERNAL-IP* the webpage of *mms-cloud-skeleton* should appear and you are ready for CI.

Cloud Build for CI/CD

The CI integration is going to be done with GCP Cloud Build. In order to have everything up for the deployment, we have to give editor role to the Cloud Build service account:

```
CLOUDBUILD_SA="$(gcloud projects describe $PROJECT_ID \
    --format 'value(projectNumber)')@cloudbuild.gserviceaccount.com"
gcloud projects add-iam-policy-binding $PROJECT_ID \
    --member serviceAccount:$CLOUDBUILD_SA --role roles/editor
```

Then, go to the GCP console and create a *Cloud Build Project* and a trigger from your repository. This step is not done by Terraform because we are also deploying the resources from there with Terraform. Once the *Cloud Build* is created, make a push to your repository and all the environment is going to be created and also the *Kubernetes* yaml will be pushed to *GKE*.

Remove all resources

You can remove all the resources executing the following command:

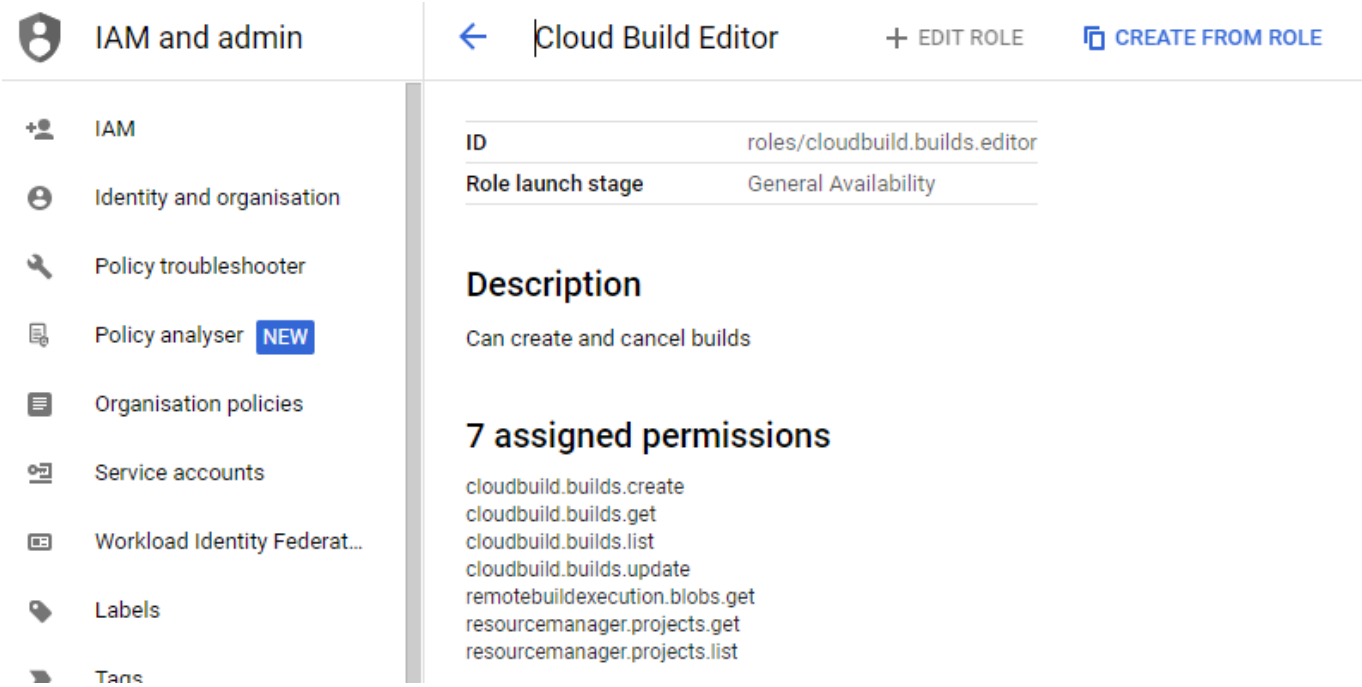
```
terraform destroy
```

Solution of the IAM Role assignation

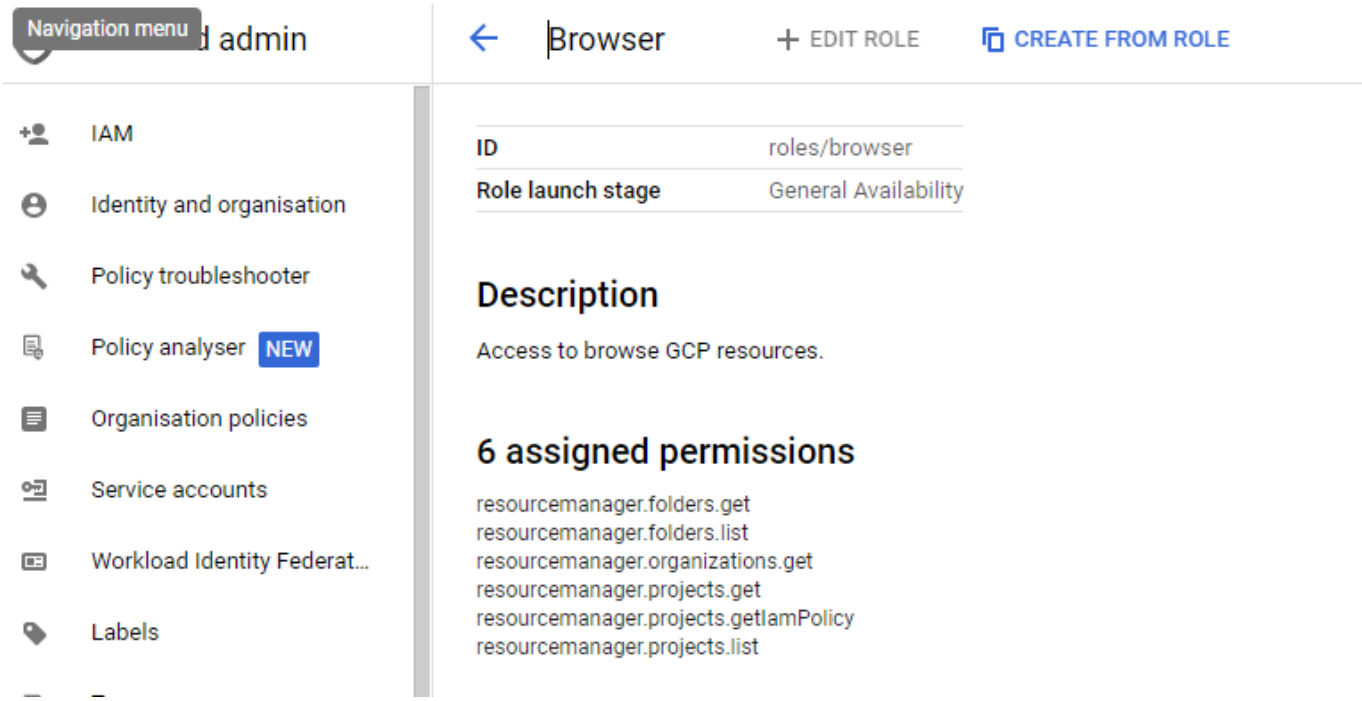
Devops Team Group

The Devops Team Group must have the possibility to deploy clusters in Kubernetes. As per our example it is done by *Cloud Build*, they only have to have the permissions for manage *Cloud Build*, so their roles would be:

1. Cloud Build Editor - For being able to manage the Cloud Builds



2. Browser - For navigate between the resources without editing them.



Finance team

The Finance Team have to be able to manage all related to billing accounts, so their roles would be:

1. Billing Account Administrator - For being authorized to see and manage all aspects of billing account

IAM and admin

- IAM
- Identity and organisation
- Policy troubleshooter
- Policy analyser NEW
- Organisation policies
- Service accounts
- Workload Identity Federat...
- Labels
- Tags
- Settings
- Privacy and security
- Identity-Aware Proxy

Billing Account Administrator + EDIT ROLE CREATE FROM ROLE

ID	roles/billing.admin
Role launch stage	General Availability

Description

Authorized to see and manage all aspects of billing accounts.

90 assigned permissions

billing.accounts.close
billing.accounts.get
billing.accounts.getCarbonInformation
billing.accounts.getIamPolicy
billing.accounts.getPaymentInfo
billing.accounts.getPricing
billing.accounts.getSpendingInformation
billing.accounts.getUsageExportSpec
billing.accounts.list
billing.accounts.move
billing.accounts.redeemPromotion
billing.accounts.removeFromOrganization
billing.accounts.reopen
billing.accounts.setIamPolicy
billing.accounts.update
billing.accounts.updatePaymentInfo
billing.accounts.updateUsageExportSpec