

This document presupposes that the reader possesses a commensurate level of technical expertise with respect to Google Cloud and CI/CD pipelines. It is incumbent upon the reader to have a fundamental comprehension of these concepts to fully appreciate the information contained herein. If the reader is not conversant with Google Cloud or CI/CD pipelines, it is strongly recommended that further research be conducted on these topics prior to proceeding. It's also presupposed that all the necessary API's and products are available and no further explanation on how to enable them is needed; nevertheless, in some instances official documentation will be provided.

In my capacity as a technical support specialist for Google Cloud Platform, the exercises delineated in this document shall be approached as if they were inquiries from a customer. The information offered herein shall be both accurate and congruent with Google policies. It is my express intention to furnish the reader with the most current and dependable information at my disposal and provide factually accurate information and code that can be used in a production environment.

TASKS-OBJECTIVES

- **Register the Container generated by the DockerFile with Cloud Build / Artifacts.**

To do so, I have employed Cloud Build, specifically the triggers [1]. In the project, I've created [MM-github-trigger](#) to take care of any change in the main branch of the GitHub repository, bear in mind that this trigger will be valid for any source repository admitted, be it a GCP one on-prem. For the trigger configuration, please refer to [Img 1]. Please note that the Google recommended image repository is Artifact registry but since I do not have the needed permissions to enable routing to it, I'll be using Container registry instead.

- **Generating a YAML file for Docker Composer.**

The compose file can be found in the root directory of the repository under "compose.yaml" and its contents are as follows.

```
version: "3" # Specifies the version of the Docker Compose file format
services:
  mms2023: # Defines a service named "mms2023"
    image: gcr.io/a37nnunvhmc0oiwh3rx0rwkcl4gr5r/github.com/emanueltud0r/mmcechallenge2023:latest # Specifies the Docker image
    ports:
      - '3000:3000' # Maps port 3000 on the host to port 3000 in the container
    expose:
      - "3000" # Exposes port 3000 to other services in the same network
```

- **Generate the Terraform files to have the infrastructure as code and be able to deploy with Kubernetes.**

In the project, the files are in the root directory. main.tf and providers.tf [Code 1] → create de k8 cluster. deployment.yml [Code 2] → deploys the application to the cluster.

In the cloudBuild.yml [Img 2] file, you can see all the steps of the pipeline.

- **MediaMarkt wants to store sensitive information on Google Cloud Platform (GCP) and uses the principle of least privilege in the assignation of roles. Suppose you oversee assigning roles in GCP, admin of the organization. Your task is to decide which role would be appropriate for each group of people: the DevOps team for creating clusters in Kubernetes and Finance team in the managing billing in GCP. Detail which roles should apply and the steps they applied in the IAM GCP Console.**

The PoLP states that each individual should be given only those privileges needed for a task to be completed. As we have groups of people (namely, DevOps and Billing)

DevOps (Creating clusters)

As the requirements only state creating Clusters and not deleting them, I believe that the best approach is to create a custom IAM role that is a subset of the “Kubernetes Engine Cluster Admin”. As we do not want the team to be able to delete clusters, we must remove the “container.clusters.delete” permission. Thus, with this permission they’ll be able to create view and update clusters but not delete them. As per indications on how to create the custom role, a comprehensive official explanation can be viewed on [2].

Finance team.

As it’s stated that it’s a team, the best course of action would be to preliminary grant “Billing Account Administrator” to all the members. This can be done by creating a group [3]. Nevertheless, it would be highly recommended to isolate further and create other groups within the Finance team which do not have the ability to delete objects as that would cause a problem with GDPR compliance.

In [4] there can be found a tutorial on how to grant specific roles to project-participants.

[1] [Creating and managing build triggers | Cloud Build Documentation | Google Cloud](#)


[2] <https://cloud.google.com/iam/docs/creating-custom-roles#creating>

[3] <https://cloud.google.com/iam/docs/groups-in-cloud-console>

[4] <https://cloud.google.com/iam/docs/grant-role-console>

Media Markt Cloud Engineering Challenge 2023


[Img 1]

Source:  [EmanuelTud0r/MMCEchallenge2023](#) [View triggered builds](#)

Name *
MM-github-trigger
Must be unique within the project's region

Region *
global (non-regional) ▼

Description
Trigger to build a docker image from the source code in.

Tags 

Event

Repository event that invokes trigger

☒ Push to a branch
☐ Push new tag
☐ Pull request
Not available for Cloud Source Repositories

Or in response to

☐ Manual invocation
☐ Pub/Sub message
☐ Webhook event

Source

Repository generation

☒ 1st Generation
☐ 2nd Generation PREVIEW

Repository *
EmanuelTud0r/MMCEchallenge2023 (GitHub App) ▼
Select the repository to watch for events and clone when the trigger is invoked

Branch *
^main\$
Trigger only for a branch that matches the given regular expression [Learn more](#)

<input type="checkbox"/>	Status	Build	Source	Ref	Commit	Trigger Name
<input type="checkbox"/>	✓	0411cbad	EmanuelTud0r/MMCEchallenge2023	main	c9a8678	MM-github-trigger
<input type="checkbox"/>	✓	b8e9957c	EmanuelTud0r/MMCEchallenge2023	main	842629a	MM-github-trigger

[Img 2]

```
You, hace 1 segundo | 1 author (You)
1  steps:
2
3  >    - id: 'build' ...
10 >    - id: 'publish' ...
17
18 >    - id: 'tf init' ...
22
23 >    - id: 'tf plan' ...
27
28 >    - id: 'tf apply' ...
32
33 >    - id: "deploy" ...
37  images:
38    - 'gcr.io/a37nnunvhmc0oiwh3rx0rwkcl4gr5r/g
```

```
- id: 'build'
  name: 'gcr.io/cloud-builders/docker'
  entrypoint: 'bash'
  args:
    - '-c'
    - |
      docker build -t
gcr.io/a37nnunvhmc0oiwh3rx0rwkcl4gr5r/github.com/emanueltud0r/mmcechallenge2023:latest .
  - id: 'publish'
    name: 'gcr.io/cloud-builders/docker'
    entrypoint: 'bash'
    args:
      - '-c'
      - |
        docker push
gcr.io/a37nnunvhmc0oiwh3rx0rwkcl4gr5r/github.com/emanueltud0r/mmcechallenge2023:latest
```

Media Markt Cloud Engineering Challenge 2023

[Code 1]

```
resource "google_container_cluster" "primary" {
  name          = "mm-challenge-2023" # cluster name
  location      = "us-central1-c"
  initial_node_count = 4              # number of node (VMs) for the cluster

  # let's now configure kubectl to talk to the cluster
  provisioner "local-exec" {
    # we will pass the project ID, zone and cluster name here
    # nodejs-demo-319000 | us-central1-c | node-demo-k8s
    command = "gcloud container clusters get-credentials mm-challenge-2023 --zone us-central1-c --project
a37nnunvhmc0oiwh3rx0rwncl4gr5r"
  }
  node_config {
    preemptible = true
    machine_type = "e2-micro"

    oauth_scopes = [
      "https://www.googleapis.com/auth/compute",
      "https://www.googleapis.com/auth/devstorage.read_only",
      "https://www.googleapis.com/auth/logging.write",
      "https://www.googleapis.com/auth/monitoring",
    ]
    metadata = {
      disable-legacy-endpoints = "true"
    }
    tags = ["mm-challenge-2023"]
  }
  timeouts {
    # time out after 45 min if the Kubernetes cluster creation is still not finish
    create = "45m"
    update = "60m"
  }
}
```

Media Markt Cloud Engineering Challenge 2023

[Code 2]

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mm-challenge-2023
  namespace: default
spec:
  replicas: 4          # this is number of pods
  selector:
    matchLabels:
      app: mm-challenge-2023
  template:
    metadata:
      labels:
        app: mm-challenge
    spec:
      # Kubernetes run docker pull pseudo/your-image:latest under the hood
      # Image field in Kubernetes resources is simply the docker image to run.
      containers:
        - name: mm-challenge-app
          # pulling image from my DockerHub
          image: gcr.io/a37nnunvhmc0oiwh3rx0rwkcl4gr5r/github.com/emanueltud0r/mmcechallenge2023:latest
          # using if not present works well with images that has tag
          # if you image is tag:latest - use Always otherwise you
          # wouldn't get the fresh version of your image
          imagePullPolicy: Always
          # -----*
          # It is a good practice to declare resource requests and
          # limits for both memory and cpu for each container.
          # This helps to schedule the container to a node that has
          # available resources for your Pod, and also so that your
          # Pod does not use resources that other Pods needs
          # -----*
      resources:
        limits:
          memory: 6400Mi
          cpu: "250m"
        requests:
          memory: 3200Mi
```

Tudor Emanuel, Techincal Support Analyst
Google Cloud Platform, Barcelona

Media Markt Cloud Engineering Challenge 2023

```
cpu: "200m"  
# specify the container port  
ports:  
- containerPort: 3000
```