MediaMarkt Cloud Engineering Challenge

25th March 2023

Author: Anton Chernysh
Email: anton_chernysh@outlook.es
Linkedin: https://www.linkedin.com/in/anton-chernysh/

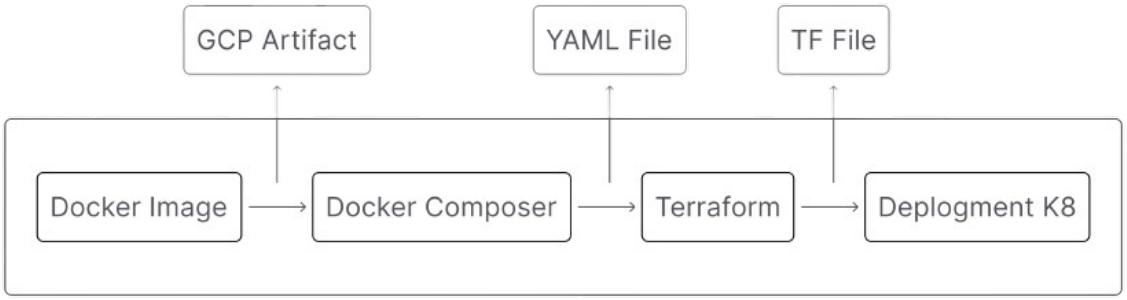
ABOUT

MediaMarkt manages Peta-Bytes of information, and security goes first. In order to ensure the Minimum Least Privilege internal policy we need you to execute it and deploy an application.

In this challenge you can access to the following FrontEnd project that the MediaMarkt developer team has designated: MediaMarkt Github Link Your task as DevOps within the company is to be able to perform the Deployment, CI/CD, of this application in the most automated way possible.

TASKS-OBJECTIVES

- Register the Container generated by the DockerFile with Cloud Build / Artifacts.
- Generating a YAML file for Docker Composer.
- Generate the Terraform files in order to have the infrastructure as code and be able to deploy with Kubernetes.
- Answer the question to check the understanding of the Minimum Least Priviledge in the Roles assignment.



IAM QUESTION TO WRITE

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 are in charge of 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.

EVALUATION

The points will be distributed in the following order (total 1200 points):

- Cloud Build/Artifact to generate the container (150 points).
- Generation of the Docker Composer YAML (150 points).
- Creation of the Terraform Files (400 points).
- Commands for the Deployment through TF files (kubectl) (200 points).
- Solution of the IAM Role assignation (300 points).

In the case of a tie, the quality of the document and the quality of the delivery, as well as its explanation, will be taken into account. If in either case there is still a tie on score, the submission that was made first will be the winner.

SUBMISSION

The submission will be the link to the GitHub repository where your solution is. In this repository you will have the Document explaining the process followed to solve each

task. Also you need to upload the Terraform files (.tf), the Docker Compose (.yaml) and any other file you have used to automate the deployment and assign the roles.

The Document is important to be written in letter size 12, Arial font or similar (understandable typography), DIN A4 page, line spacing 1.5, PDF format and maximum of 10 pages of content. The not use of this rules can be penalized, the worst case if the document is not readable will not be corrected.

Solution

0. Previous steps

During the hackathon I had to install the following dependencies:

Google Cloud SDK: To be able to access our Google Cloud account by console.
Terraform: Being able to configure the cloud resources that we are asked for in the hackathon.
Docker: Being able to compile and run docker images
Kubernetes: Being able to use it in terraform and check if our k8 cluster has been executed correctly.

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

First I initiate the authorization flow between my local machine and Google Could using their SDK. It opens a browser window asking to sign in to our Google account .
gcloud auth login

Then I wanted to set the PROJECT_ID environment variable so I can reuse it everywhere.

```
export PROJECT_ID=<PROJECT_ID>
```

After setting the variable, we can use it to activate the project for our GCloud SDK. This allows us to execute commands against our project.
gcloud config set project $PROJECT_ID
The next step is to create a new Artifact Registry repository in our project for storing docker images.

```
gcloud artifacts repositories create myapp-repo \
--repository-format=docker \
--location=us-west1 \
--description="Docker repository"
```

After creating the repository, we should build the image using the Dockfile located in ./app/ and using -t flag we tag the image with the repository path where we will store it.

```
docker buildx build -t "us-west1-docker.pkg.dev/$PROJECT_ID/myapp-repo/myapp-image" ./app/
```

If you execute `docker images` you should be able to see the new image.

The next command configures Docker to authenticate with the Artifact Registry repository located in the us-west1 region. This is necessary before we can push the container image to the repository.

```
gcloud auth configure-docker us-west1-docker.pkg.dev
```

And now we can push the container image that was built before to the Artifact Registry repository specified in the tag. Once the push is complete, the container image will be stored in the repository and can be used for deployment to a Kubernetes cluster or any other environment that supports Docker container images.

```
docker push us-west1-docker.pkg.dev/$PROJECT_ID/myapp-repo/myapp-image
```

2.Generation of the Docker Composer YAML

This step is simple, we just need to create a docker compose file that uses the image we stored in the artifacts registry.

```
version: '3'
services:
    web:
        image: us-west1-docker.pkg.dev/$PROJECT_ID/myapp-repo/myapp-image
        ports:
             - "3000:3000"
```

3.Creation of the Terraform Files

3.1 Setup Environment

Now I logged again in gcloud and generated application default credentials, which are used by default when you are working with Google Cloud APIs using tools like Terraform.

```
gcloud auth application-default login
```

In order to allow manage Kubernetes clusters on Google Cloud we need to enable a specific service for it. This command enables the Kubernetes Engine API service for the project.

```
gcloud services enable container.googleapis.com
```

In my case, I had to install an additional plugin. This plugin provides authentication and authorization for Google Kubernetes Engine (GKE) clusters using the gcloud command-line tool. It allows Terraform to create and manage GKE clusters using your Google Cloud credentials. Note that this command is only necessary if the plugin is not already installed on your machine.

```
sudo apt-get install google-cloud-sdk-gke-gcloud-auth-plugin
```

3.2 Terraform file

I have not put the code here as it would need too much space. I splited the chapter in different block kinds detailing which one I used and the mandatory arguments for the configuration.

Terraform Block

This block defines the version of Terraform to be used, as well as any required providers for the configuration. I used it to specify the required minimum version of kubernetes.

Providers blocks

Provider blocks define the credentials and configuration settings needed to connect to a cloud provider's API. In our case, I needed two providers: google and kubernetes.

Resource blocks

Resource blocks define the infrastructure objects that will be created by Terraform. In this file, four resources are defined:,

- **google_container_cluster:**

The first creates a Google Kubernetes Engine (GKE) cluster with the name my-gke-cluster. The location parameter specifies the region in which the cluster will be created, and remove_default_node_pool is set to true to ensure that no default node pool is created. The initial_node_count parameter sets the number of initial nodes in the cluster. master_auth block configures the authentication method for accessing the cluster.

- google_container_node_pool:

This block creates a Google Kubernetes Engine (GKE) node pool with preemptible nodes. The name parameter sets the name of the node pool, and cluster and location are taken from the google_container_cluster resource created earlier. node_count sets the number of nodes in the node pool, and node_config sets the configuration for each node, including preemptible and machine_type.

- kubernetes_namespace:

This block creates a Kubernetes namespace with the name kn-app.

- kubernetes_deployment:

This resource defines a Kubernetes Deployment, which specifies how many replicas of a Pod should be created and how they should be managed. The metadata block provides a name and namespace for the Deployment. The spec block specifies that only one replica of the Pod should be created, and defines the Pod template that should be used. The template specifies the Docker image to use, the container name, and the port on which the container listens.

- kubernetes_service:

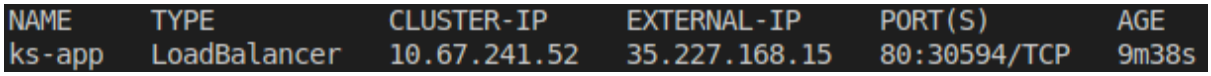
This resource creates a Kubernetes Service object that exposes the Pods managed by the Deployment created in the previous resource block. The metadata block provides a name and namespace for the Service. The spec block specifies that the Service should use a LoadBalancer type, which means that a load balancer will be created to route traffic to the Pods. The selector block specifies which Pods should be targeted by the Service, based on their labels. Finally, the port block defines the port that the Service should listen on and the target port of the Pods.

4. Commands for the Deployment through TF files (kubectl)

```
terraform init # Initialize the terraform project
terraform plan # Check the changes that will be applied
terraform apply # Apply the changes and create the cluster
```

The the first time, I also needed to get the credentials for the cluster so I was able to access the cluster and deploy the kubernetes:
gcloud container clusters get-credentials my-gke-cluster --region us-west1

You can check the service by running:
`kubectl get services -n kn-app`
It should show an external IP address for the service that you can use to access the application using the web browser.



NAME	TYPE	CLUSTER-IP	EXTERNAL-IP	PORT(S)	AGE
ks-app	LoadBalancer	10.67.241.52	35.227.168.15	80:30594/TCP	9m38s

5. Solution of the IAM Role assignation

Description: 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 are in charge of 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.

Response:
The DevOps team is responsible of creating and managing the Kubernetes so it need administration permissions for it. In order to set this permissions, you need to follow the next steps:

1. Navigate to the IAM & Admin section of the GCP Console and select the project.
2. Click on the "ADD" button to add a new member.

3. In the "New Members" field, enter the email addresses of the DevOps team members who need access.
4. In the "Role" drop-down menu, select "Kubernetes Engine Cluster Admin". This role allows the DevOps team to create and manage clusters in Kubernetes, including managing the nodes, pods, and services in the clusters.

The Finance team only needs billing information in GCP. For that you need to follow this steps:

1. Navigate to the IAM & Admin section of the GCP Console and select the project.
2. Click on the "ADD" button to add a new member.
3. In the "New Members" field, enter the email addresses of the Finance team members who need access.
4. In the "Role" drop-down menu, select "Billing Account Administrator". This role allows the Finance team to view billing information, manage billing accounts, and make payments. In case each member of team only need some of those permissions, it could be provided of role with less Billing Account permissions.