About this document

This document explains the process followed to solve each task of the MMS Hackathon in the cloud engineering track.

Prerequisite

I have created a fork of this repository: nuwe-io/mms-cloud-skeleton
The fork, which also contains the solution to the tasks, is this one: fguendling/ mms-cloud-skeleton-fork

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

Note: everything has been done via GCP Console.

1. Enable Cloud Build API
2. Go to Cloud Build Service in GCP
3. Set up Build Trigger: mms-cloud-skeleton-fork-build-trigger
    a. Go to Triggers
    b. Connect Repository: fguendling/ mms-cloud-skeleton-fork (follow the setup process, including authentication, repository selection)
    c. Create the trigger. Select an appropriate name (mms-cloud-skeleton-fork-trigger) and the right repository, keep default settings during the setup process
4. Run the build, by clicking RUN on the trigger
5. This will build the image and store the artifact in GCP's container registry. I assume this storage location is OK and moved on.

Generation of the Docker Composer YAML (150 points).

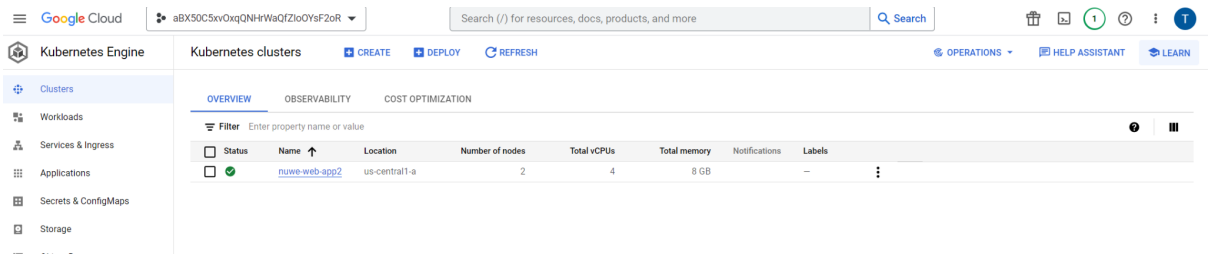
In the forked repository, I created and commited the file docker-compose.yml. It specifies a service called web, which uses the image we have built previously and also maps the ports properly.
I have checked with Michael in the chat and communicated that this docker compose setup would be helpful for local development only (not for deployment), which he confirmed.

Creation of the Terraform Files (400 points).

In order to have automated deployments with Terraform, it is necessary to set up execution of the Terraform workflow in Cloud Build. For that, I have committed the file cloudbuild.yaml to the forked repository. It specifies the basic Terraform workflow (init, plan, apply). I have also specified in Cloud Run that cloudbuild.yaml will be used for builds, because otherwise it would continue to use the Dockerfile from the first task.

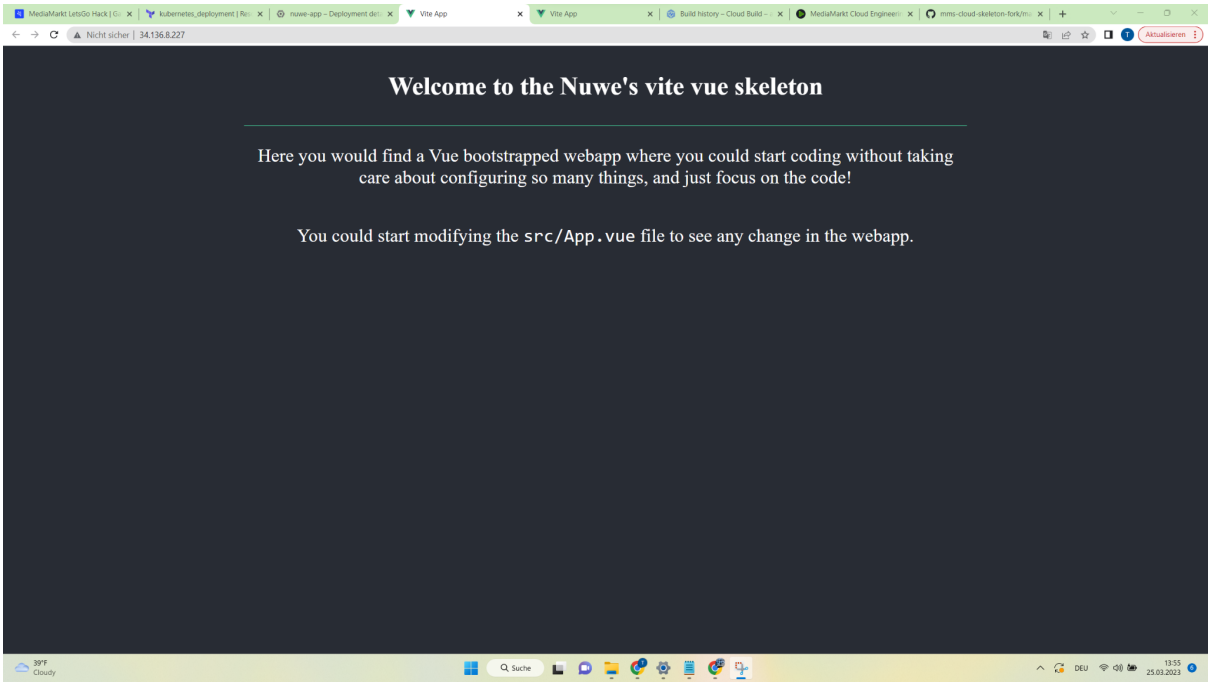
Next, the Terraform configuration itself is required. I have created main.tf which will be used to configure all resources.

The first resource will be the K8s Cluster (google_container_cluster). The “google” provider is required for that. It is also necessary to enable Kubernetes Engine API on GCP. The default Cloud Build service account doesn’t have the container.clusters.create privilege assigned and we are not able to assign it. That’s why I had to change the service account that is used by Cloud Build to the Compute Engine default service account, which has Editor privileges and is hence powerful enough to execute the pipeline and create the K8s cluster. The two pictures below show the output of Cloud Build and the created cluster in GKE.



The next step is to extend the Terraform configuration to not only use the google provider, but the “kubernetes” provider as well. We can then create a kubernetes_deployment resource and a kubernetes_service resource in Terraform. Please refer to main.tf for the details.

After everything deployed, we can access the application via the service endpoint:



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

Kubectl does not really apply for the scenario. Please consider that this setup is deployed via Terraform and Cloud Build only and not by hand via kubectl.

We could alternatively specify the K8s configuration via .yaml files (e.g., deployment.yaml, service.yaml, which would look like something like this (just as an example):

```
apiVersion: v1
kind: Pod
metadata:
  name: our-app
spec:
  containers:
  - name: our-image
    image: our-image:1.28
```

…

We could then make it a reality via the commands (examples only)

kubectl apply -f deployment.yaml

kubectl apply -f service.yaml

Solution of the IAM Role assignation (300 points).

DevOps team for creating clusters in Kubernetes

We could assign the predefined role roles/container.clusterAdmin, but it is also possible to create a custom role and assign only a single permission: container.clusters.create.

Finance team in the managing billing in GCP.

As there is not much detail given, I would check if it is appropriate to grant the powerful roles/billing.admin role to the finance team/ group, as this can manage all aspects of billing accounts. Regarding billing, there are multiple roles available and this can be decided after further consideration of the actual requirements.

How to assign the roles

In the console, these roles can be assigned under IAM by selecting the name of the person or group, while groups are preferable.