CSCI 5409 Cloud Computing – Summer 2023

Kubernetes Assignment

Building Cloud-Native CI/CD Pipeline and Deploying workload to Google Kubernetes Engine (GKE)

Note: We would like to express our gratitude for being able to reuse the Docker assignment developed by Professor Rob Hawkey in the development of this assignment.

Introduction

This assignment will measure your understanding of concepts such as containerization, CI/CD pipeline, and Google Kubernetes Engine (GKE) by building a cloud-native application on Google Cloud Platform (GCP). This assignment assures us that you have understood the topics of Docker, CI/CD, and Kubernetes.

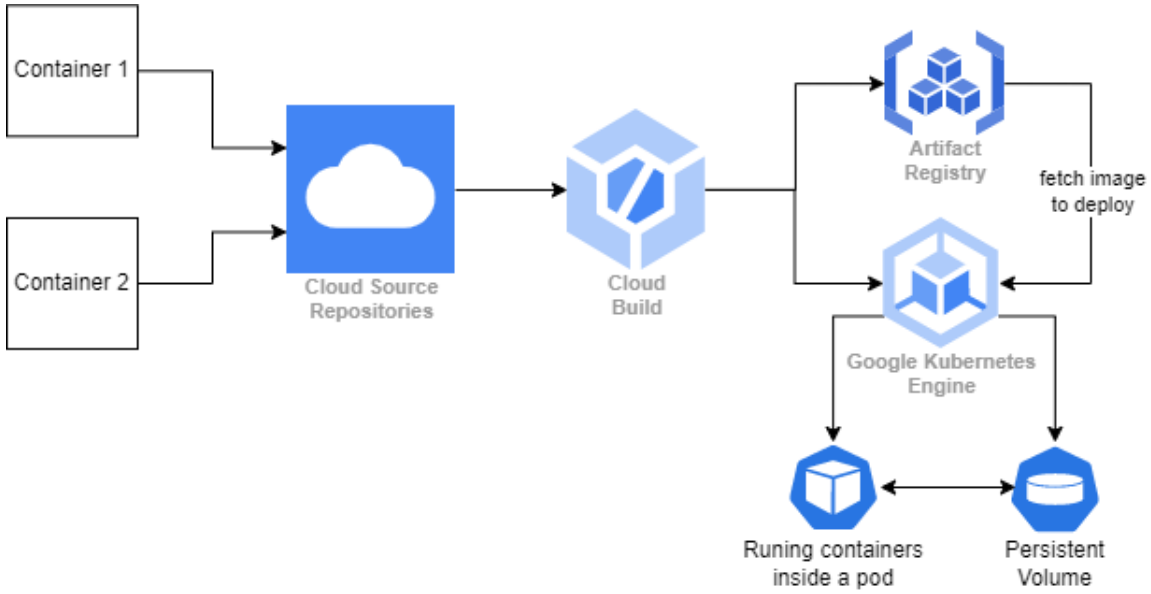
Learning Outcomes

- You understand how to containerize the application using Docker.
- You understand how to build a code deployment pipeline using GCP tools.
- You understand the concepts of Kubernetes and how to use both the GCP console and IaC (i.e. Terraform) to create clusters. Using CI/CD pipeline, you are able to deploy applications on the GKE cluster.
- You are able to attach persistent volumes to the GKE cluster and your applications can read and write data in the persistent volumes.
- You are able to use Kubernetes tools, e.g. kubectl, to interact with containers in the pod and check the status and diagnose the problems of your GKE cluster and components.
- You understand how application update strategies work and how to control the versions of your applications in GKE.

More experience building REST APIs Requirements

You will build two simple microservices in the programming language which you like. These services should be able to interact with each other. You can re-use the services which you have developed in Assignment 1 with minor changes needed for this assignment. To deploy the services on GCP, you will have to create a CI/CD pipeline which deploys the service to GKE. In GKE you will have to create a persistent volume that should be accessed by both containers to store and retrieve file data. You can use GCP services such as Cloud Source Repository (a tool similar to Gitlab) to store the source code, Artifact Registry (a tool similar to Docker Hub) to store the Docker images, and GKE to deploy the images to clusters.

The following diagram shows the GCP services and workflow you will be using in this assignment.



Container 1

The role of the first container is to be able to store the files to persistent volume in GKE and serve as a gatekeeper to calculate the products from the stored file. It must:

1. Be deployed as a service in GKE to be able to communicate with the Internet.
2. Have access to persistent volume in GKE to store and retrieve files.
3. Be able to **communicate with container 2** and vice versa.
4. Validate the input JSON request incoming from the REST API.
5. Send the "file" parameter to container 2 to calculate the product from the "product" parameter in the API and return the response from container 2.

You can re-use the code which you have developed in assignment 1. The modifications which you will have to make are:

1. Create a **POST API** with endpoint: **/store-file**

This API expects container 1 to create a file and store the data given in the API request. The file should be stored in the GKE persistent storage. You must make sure that your container can access the persistent storage.

JSON Input:

```
{
        "file": "file.dat",
        "data": "product, amount \nwheat, 10\nwheat, 20\noats, 5"
}
```

JSON Output:

If the file is stored successfully, this message should be returned:

```
{
        "file": "file.dat",
```

```
    "message": "Success."
}
```

For example, file.dat should look like:

```
product, amount
wheat,10
wheat,20
oats,5
```

If there was an error to store the file in the persistent storage, this message should be returned:

```
{
    "file": "file.dat",
    "error": "Error while storing the file to the storage."
}
```

If the file name is not provided, an error message is returned:

```
{
    "file": null,
    "error": "Invalid JSON input."
}
```

2. Create a POST API with endpoint: **/calculate**

You can re-use this API which you have already developed in Assignment 1.

JSON Input:

```
{
    "file": "file.dat",
    "product": "wheat"
}
```

JSON Output:

If the file exists in persistent storage, the total of product is returned.

Note: The total is calculated by **container 2**

The return response of **/calculate API** should be

```
{
    "file": "file.dat",
    "sum": "30"
}
```

If a filename is provided, but the file contents cannot be parsed due to not following the CSV format described above, this message is returned:

```
{
        “file”: “file.dat”,
        “error”: “Input file not in CSV format.”
}
```

If the filename is provided, but not found in the persistent disk volume, this message is returned:

```
{
        “file”: “file.dat”,
        “error”: “File not found.”
}
```

If the file name is not provided, an error message is returned:

```
{
        “file”: null,
        “error”: “Invalid JSON input.”
}
```

Container 2

The role of container 2 is to listen on another port and endpoint that you define for the service and returns the total of the product. It must:

1. Have access to the persistent volume of GKE.
2. Container 2 should be **able to interact with container 1** and vice versa.
3. Calculate the total of the product by calculating the sum of rows of the same product from the given file.
4. Return the total in the appropriate JSON format, or an error indicating the file is not a proper CSV file in the appropriate JSON format.

Please keep in mind that when I use <>'s I am trying to convey to you that you need to replace the information with the real value, don't include the <>'s or you'll break our code that checks your responses.

JSON your app sends to my app's /start your app will send me the following JSON in your POST to /start

```
{
         "banner": "<Replace with  your Banner ID, e.g., B00XXXX>",
        "ip": "<Replace with the IPv4 of the service (NodePort or Load balancer) of the container 1/>"
}
```

JSON my app sends to your app's /store-file when you POST to my app's /start endpoint with valid JSON my app will immediately interact with yours by sending a POST with the following JSON to your app's /store-file endpoint:

```
{
        “file”: “file.dat”,
        “data”: “product, amount \nwheat, 10\nwheat, 20\noats, 5”
}
```

Your app should respond as per the responses mentioned above.

JSON my app sends to your app's /calculate

```
{
        “file”: “file.dat”,
        “sum”: “30”
}
```

Additional Requirements

1. You must create two repositories for each container in GCP Cloud Source Repository and use GCP Cloud Build to build the pipelines for both services. You must make sure the CI/CD pipeline works well such that the application deployed in the GKE cluster is updated when new code commits check in.
2. You will have to learn the GCP CI/CD tools by yourselves. The tools are Cloud Source Repository, Cloud Build, and Artifact Registry.
3. You will have to learn Terraform by yourselves and create a Terraform script to start your GKE cluster by running the script. You will be launching your GKE cluster by running the Terraform script from GCP Cloud Shell. So we highly recommend you start learning as early as possible to be prepared for the assignment. You will learn Kubernetes in lectures.
4. You must create a yaml file “xxxx.yaml” to deploy your application (workload) to the GKE cluster from the Cloud Shell.
5. In order not to max out your GCP credit and make things consistent for the marking, you need to follow the configurations below to create your GKE cluster:
    a. The GKE cluster is a standard cluster instead of an Autopilot cluster.
    b. You only create 1 node for this cluster.
    c. Choose “container-Optimized OS with containerd (cos_containerd) (default)” as the image type for the node.
    d. Choose “E2” for Series.
    e. Choose “e2-micro (2 vCPU, 1GB memory) for Machine type.
    f. Choose “Standard persistent disk” for Boot disk type.
    g. Enter “10” for Boot disk size.
    h. Leave default for all other options.

The above configuration of the cluster will cost about $79/month. To save money, you don't need to keep your cluster running all the time. You only use it when you test your cluster and applications, and make the video as part of the submission.

6. You must attach a persistent volume (1GB is enough) to your GKE cluster. The persistent volume is mounted to a directory called "/xxxx_PV_dir" where "xxxx" is your first name. The directory is placed in the root directory /. You need to decide what the mode of the PV should be. Your application should be able to read and write files to "/xxxx_PV_dir/".
7. You must expose the application of container 1 as a service to the internet. The yaml file ""xxxx.yaml" not only deploys your workload, but also exposes the service.

After you test and are sure that your GKE cluster works as expected, you can shutdown the cluster. You will only launch your cluster from your Terraform script after your finish testing your GKE cluster.

How To Submit

Video Submission:

We need you to show us the creation of Kubernetes cluster using Terraform. You will be running your Terraform script from GCP Cloud Shell to launch your GKE cluster. The creation of the GKE cluster usually takes about 3-5 minutes. You can pause the recording while you are waiting. After the cluster is created you will show case the CI/CD pipeline which you've built and run the pipeline by making some code changes from your environment. You must make sure you showcase all the process from building the image to image being deployed to the GKE cluster. Again you can do this from the Cloud Shell. You can clone your code from Cloud Source Repository to your home directory in Cloud Shell. Please be aware your Cloud Shell home directory quota is 50G and you are allowed to keep the shell open for 50 hours per week. So you have to close it when you don't use it. You must upload the video on Brightspace under the assignment submission. Videos not submitted to Brightspace will not be considered.

Code Submission:

Create a folder in your Gitlab repository labeled **K8s**. Put all your yaml files and source code in the folder and make sure you commit your files to the **main** branch and push changes to Gitlab. This must be done before the assignment deadline. Code pushed after the deadline will not be graded.

Marking Rubric

Your grades for this assignment will be based on the details you have put in your video submission and the functionality of your application endpoints. Please refer to the above instructions to understand the process of creating and submitting the video. I kindly request you to read it carefully and follow the steps to complete the assignment successfully.

Assignment Rubric	
Cluster creation using Terraform script	20%
CI/CD pipeline on GCP	30%
Successfully running workload on GKE	30%
APIs working as per the requirements. Your code will be marked by the app that I will write, my app will: 1. Listen for request to **/start** to initiate the check process: 30% 2. Records the IP you send to **/start** in the database. 3. Sends a post request to your IP's **/store-file** endpoint: 35% 4. As per your response, sends a post request to **/calculate** endpoint: 35% 5. Verifies that your application is responding back with the correct values. **Note:** Any API not performing operations as per the requirements will be graded as 0.	20%

Suggested Timeline

This is a suggested timeline. Students will have to spend a lot of time on self-directed learning to get started with GCP and Terraform. As a graduate student, you should be able to search and learn some topics by yourselves. I suggest you start as early as possible.

Week	Course Topic	What you should work on the assignment deliverables
Week 3		GCP credit released. Setup your environment on GCP. You should start learning GCP CI/CD tools and Terraform. Get familiar with GCP Cloud Shell because you will interact with a lot of GCP resources from there.
Week 4		Try to import your Docker assignment to GCP and make requested changes following the instructions and requirements. Try to use the GCP CI/CD tools to make the pipeline work for the Docker assignment.
Week 5	Kubernetes and Google Kubernetes Engine	Practice creating GKE clusters from both the console and Terraform. Try to deploy your workload of the Docker assignment to the GKE cluster and make sure it works as requested.
Week 6-7		Refine your assignment, make sure you satisfy all the requirements, record the video, and submit to Brightspace.