Project: Display products from an API on the browser using GKE

Client: Paymentsense

Author: Charles Victor

Agency: NDC Tek

Table of Contents

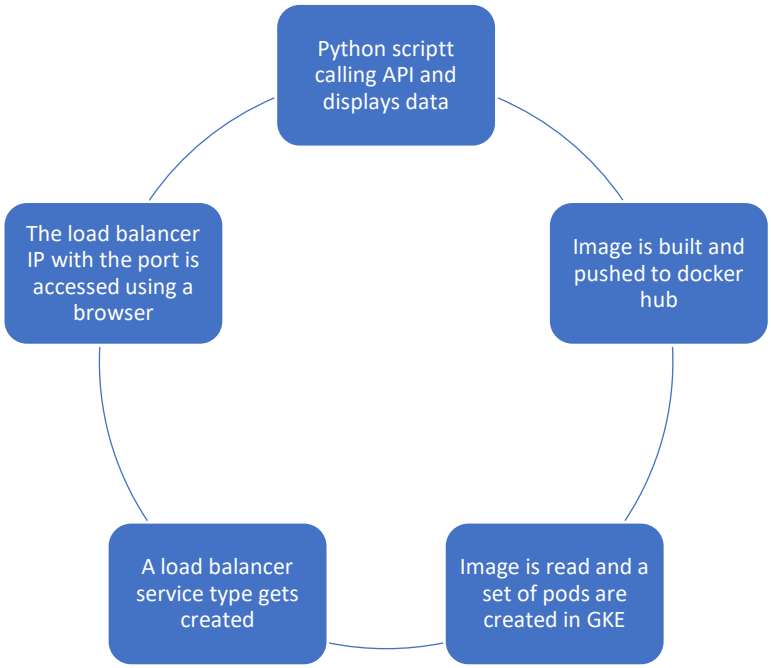
1) Project requirement and Approach ---------- 3
2) Terraform code ---------- 4
3) Python code ---------- 5
4) Docker image building ---------- 6
5) Kubernetes yaml files for Deployment and Service ---------- 7
6) Pod and Service Status ---------- 8
7) Accessing the service on the browser ---------- 9
8) Horizontal Pod Autoscaling ---------- 10
9) References ---------- 11

Project requirement

1. Create a GKE Cluster using Terraform
2. Create an app that accepts response from : https://reqres.in/api/products/, and displays complete list of products. (you can use a language of your choice)
3. Deploy the app using cloudBuilder and Helm3 to GKE. You may use any other tooling if you wish.
4. Enable Auto scaling, and test the same/ provide us with a test script or documentation.

Approach and Architecture

1) Create a GKE cluster using Terraform.
2) A python script is written to read the product details from an API page.
3) A python docker image is created using docker build
4) The image is pushed to public repository in docker hub
5) Kubernetes deployment is created using the docker image built in step 3
6) A service of type Load Balancer which connects to the pods gets created
7) A horizontal pod autoscaling object is created to scale if the cpu utilization goes above 70%
8) The load balancer IP with the port number is accessed to get the product details on the browser



Terraform code

1) VPC and subnet details are mentioned in VPC.tf
2) Google provider and terraform versions are mentioned in versions.tf
3) Project name and region details are mentioned in terraform.tfvars. *Project name has to be changed*
4) Output.tf has the details to present to the user once terraform has created the infrastructure.
5) gke.tf has the cluster, node pool and machine type details are written in them

After running terraform plan followed by apply, the following output details are presented:

…..

Apply complete! Resources: 2 added, 0 changed, 0 destroyed.

Outputs:

kubernetes_cluster_host = "<HOST IP>"

kubernetes_cluster_name = "<Project Name -gke>"

project_id = "<Project Name>"

region = "us-central1"

GKE nodes:

We can check the provisioned nodes as below:

```
C:\Users\cvictor\AppData\Local\Google\Cloud SDK>kubectl get nodes
NAME                                                  STATUS   ROLES    AGE   VERSION
gke-my-project-15211-my-project-15211-058f259b-csvw   Ready    <none>   40h   v1.18.15-gke.1501
gke-my-project-15211-my-project-15211-058f259b-fbck   Ready    <none>   40h   v1.18.15-gke.1501
gke-my-project-15211-my-project-15211-15c4f2d9-0hsv   Ready    <none>   40h   v1.18.15-gke.1501
gke-my-project-15211-my-project-15211-15c4f2d9-l77t   Ready    <none>   40h   v1.18.15-gke.1501
gke-my-project-15211-my-project-15211-3374fb62-43w7   Ready    <none>   40h   v1.18.15-gke.1501
gke-my-project-15211-my-project-15211-3374fb62-btmp   Ready    <none>   40h   v1.18.15-gke.1501
```

Python code

The following python code uses flask, pandas and requests libraries to process the api data and present on the browser. It also uses a template file called product.html to convert the dataframe.

```
import requests
import pandas as pd
import json
from pandas.io.json import json_normalize
from flask import Flask, request, render_template, session, redirect
app = Flask(__name__)
@app.route("/")
def test():
    global result
    dataget = requests.get('https://reqres.in/api/products')
    s = dataget.json()
    data=list(s.items())[4][1]
    result = pd.json_normalize(data)
    return render_template('product.html', column_names=result.columns.values,
 row_data=list(result.values.tolist()), zip=zip)
if __name__ == "__main__":
    app.run(debug=True, host="127.0.0.1")
```

Docker image

The docker image is built using this command:

docker build -f Dockerfile -t pyw:latest .

```
PS C:\Docker\Final\app> docker build -f Dockerfile -t pyw:latest .
[+] Building 19.4s (14/14) FINISHED
 => [internal] load build definition from Dockerfile                                                0.0s
 => => transferring dockerfile: 275B                                                                0.0s
 => [internal] load .dockerignore                                                                   0.0s
 => => transferring context: 2B                                                                     0.0s
 => [internal] load metadata for docker.io/library/python:3.7                                       1.9s
 => [auth] library/python:pull token for registry-1.docker.io                                       0.0s
 => [1/8] FROM docker.io/library/python:3.7@sha256:8c5d7a9110c22ebf261d382fe14d46a13a4b938b32a7189  0.0s
 => [internal] load build context                                                                   0.0s
 => => transferring context: 991B                                                                   0.0s
 => CACHED [2/8] RUN mkdir /app                                                                     0.0s
 => CACHED [3/8] RUN mkdir /app/templates                                                           0.0s
 => CACHED [4/8] WORKDIR /app                                                                       0.0s
 => [5/8] ADD . /app/                                                                               0.0s
 => [6/8] ADD main.py /app/                                                                         0.0s
 => [7/8] ADD /templates/product.html /app/templates/                                               0.1s
 => [8/8] RUN pip install -r requirements.txt                                                      15.9s
 => exporting to image                                                                              1.4s
 => => exporting layers                                                                             1.3s
 => => writing image sha256:84240e2373e25ccb24cd48572809c45ba1a8c734b5bb9fd766ab622d5978b7b0        0.0s
 => => naming to docker.io/library/pyw:latest                                                       0.0s
```

Docker image is tagged and built as below:

PS C:\Docker\Final\app> docker tag pyw charlesvictor/pyapi:latest

PS C:\Docker\Final\app> docker push charlesvictor/pyapi:latest

Kubernetes

Now the docker image and the GKE clusters are available. We can provision the deployment by using the following YAML file:

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: pyapi-deploy
spec:
  selector:
    matchLabels:
      app: products
      department: sales
  replicas: 3
  template:
    metadata:
      labels:
        app: products
        department: sales
    spec:
      containers:
      - name: hello
        image: "charlesvictor/pyapi"
        ports:
        - containerPort: 5000
        resources:
          limits:
            cpu: 500m
          requests:
            cpu: 200m
```

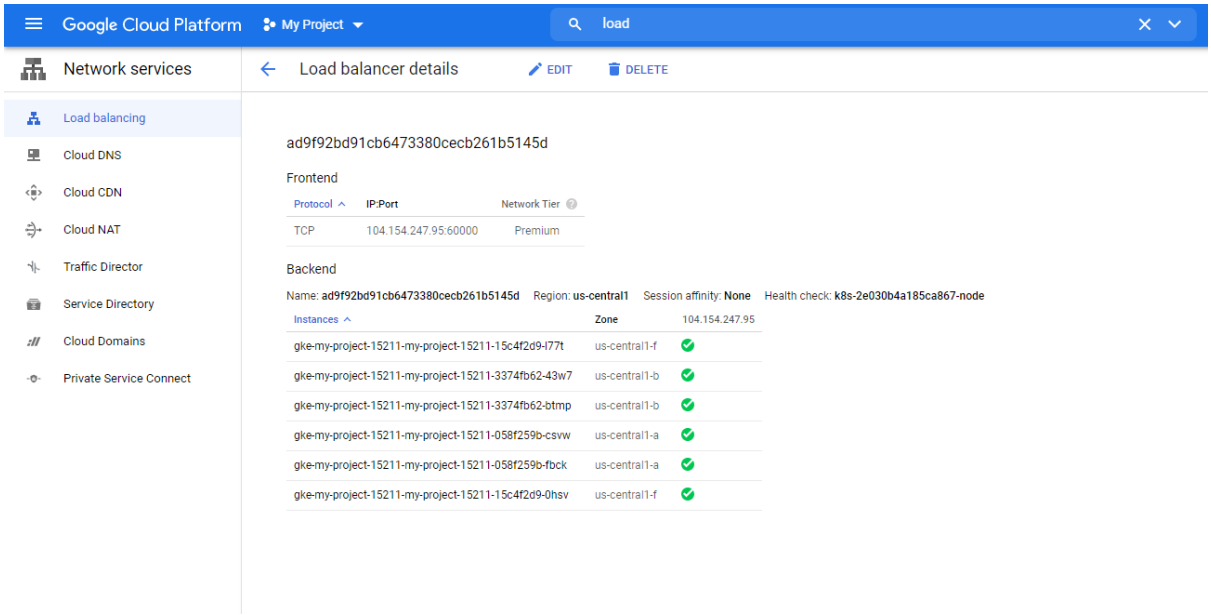
Expose the deployment via a load balancer as below:

```yaml
apiVersion: v1
kind: Service
metadata:
  name: my-lb-service
spec:
  type: LoadBalancer
  selector:
    app: products
    department: sales
  ports:
  - protocol: TCP
    port: 60000
    targetPort: 5000
```

The deployment and the service can be created as below:

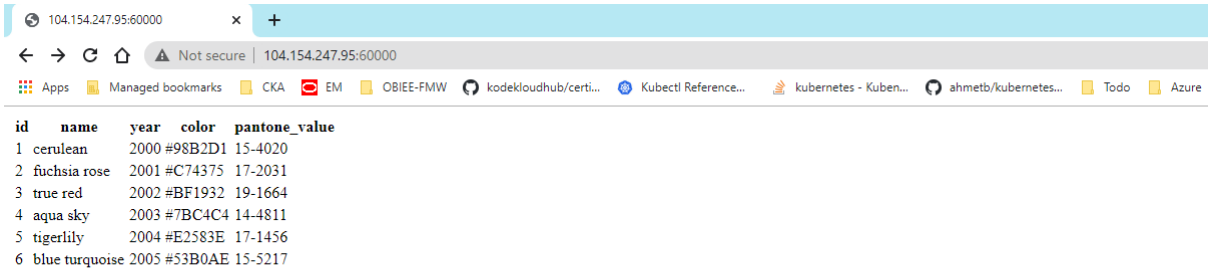
```
C:\Users\cvictor\AppData\Local\Google\Cloud SDK>kubectl apply -f my-python-api2.yaml
deployment.apps/pyapi-deploy created

C:\Users\cvictor\AppData\Local\Google\Cloud SDK>kubectl apply -f dep_lb.yaml
service/my-lb-service created

C:\Users\cvictor\AppData\Local\Google\Cloud SDK>kubectl get pods
NAME                           READY   STATUS    RESTARTS   AGE
nginx                          1/1     Running   0          28h
pyapi-deploy-bc67b59b-46vm2    1/1     Running   0          63s
pyapi-deploy-bc67b59b-lj69w    1/1     Running   0          63s
pyapi-deploy-bc67b59b-ntfs9    1/1     Running   0          63s

C:\Users\cvictor\AppData\Local\Google\Cloud SDK>kubectl get svc
NAME            TYPE           CLUSTER-IP       EXTERNAL-IP      PORT(S)           AGE
kubernetes      ClusterIP      10.187.240.1     <none>           443/TCP           42h
my-lb-service   LoadBalancer   10.187.255.154   104.154.247.95   60000:31913/TCP   59s
```

The external IP and the port has to be used to access the service on the browser. Here the external IP is **104.154.247.95 and the port is 60000**. This external load balancer is created as an NLB in GCP:



Accessing the service on the browser

The product details from the API are read and displayed on the browser as below:



id	name	year	color	pantone_value
1	cerulean	2000	#98B2D1	15-4020
2	fuchsia rose	2001	#C74375	17-2031
3	true red	2002	#BF1932	19-1664
4	aqua sky	2003	#7BC4C4	14-4811
5	tigerlily	2004	#E2583E	17-1456
6	blue turquoise	2005	#53B0AE	15-5217

Horizontal Pod Autoscaling

We can also set up horizontal pod autoscaling as below:

kubectl autoscale deployment pyapi-deploy --cpu-percent=50 --min=1 --max=10

```
C:\Users\cvictor\AppData\Local\Google\Cloud SDK>kubectl autoscale deployment pyapi-deploy --cpu-percent=50 --min=1 --max=10
horizontalpodautoscaler.autoscaling/pyapi-deploy autoscaled

C:\Users\cvictor\AppData\Local\Google\Cloud SDK>kubectl describe hpa
Name:                                                  pyapi-deploy
Namespace:                                             default
Labels:                                                <none>
Annotations:                                           <none>
CreationTimestamp:                                     Sat, 03 Apr 2021 12:36:34 +0100
Reference:                                             Deployment/pyapi-deploy
Metrics:                                               ( current / target )
  resource cpu on pods  (as a percentage of request):  5% (11m) / 50%
Min replicas:                                          1
Max replicas:                                          10
Deployment pods:                                       3 current / 3 desired
Conditions:
  Type            Status  Reason              Message
  ----            ------  ------              -------
  AbleToScale     True    ScaleDownStabilized  recent recommendations were higher than current one, applying the highest recent recommendation
  ScalingActive   True    ValidMetricFound     the HPA was able to successfully calculate a replica count from cpu resource utilization (percentage of request)
  ScalingLimited  False   DesiredWithinRange   the desired count is within the acceptable range
Events:           <none>

C:\Users\cvictor\AppData\Local\Google\Cloud SDK>
```

References

terraform.io

kubernetes.io

https://cloud.google.com/kubernetes-engine