Kubernetes Project

By Mikaela Montaos
Prepared under the direction of Professor Henry Chang

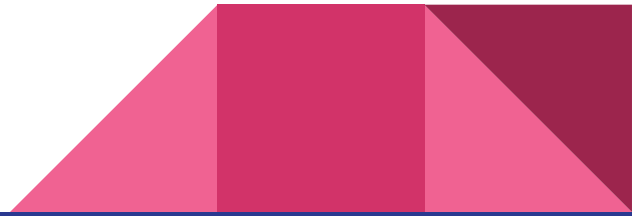
School of Engineering
Northwestern Polytechnic University
117 Fourier Ave, Fremont, CA 94539
April 2021

Table of Contents

1. Introduction
2. Design
3. Implementation
4. Test
5. Enhancement ideas
6. Conclusion
7. Bibliography

Introduction

- The platform for this project is Google Cloud Platform
- The technology used are
  - Google Kubernetes Engine (GKE)
  - MongoDB
  - Python flask web framework
  - REST API
- GKE techniques used are
  - Pod
  - Service
  - Persistent volume
  - Ingress
  - ConfigMaps

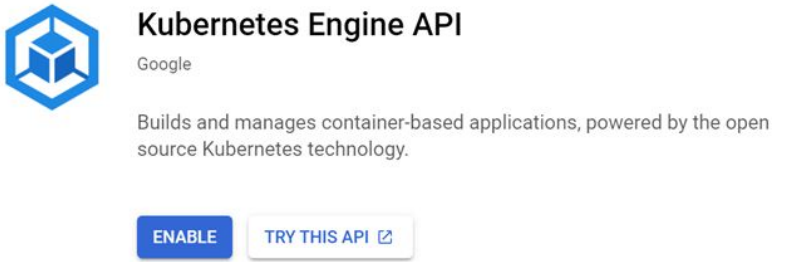


Design

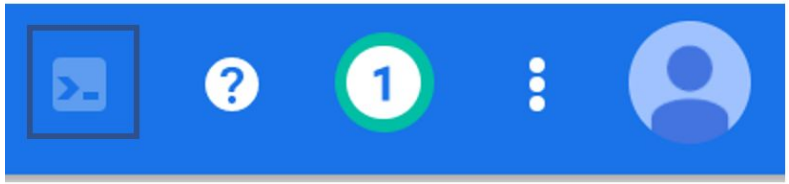
- Kubernetes has the ability to flexibly run on distributed systems
  - Service discovery and load balancing
  - Storage orchestration
  - Automated rollouts and rollbacks
  - Automatic bin packing
  - Self-healing
  - Secret and configuration management
- MongoDB is fast, it supports JSON query language, and supports dynamic queries which is necessary in cloud computing
- Flask is easy to understand, fast and flexible like Kubernetes with the ability to scale up
- REST API is flexible because it can handle multiple calls and return various data formats

Implementation

1. If starting from a new project, enable GKE



2. Activate Cloud Shell and create a cluster



Implementation

3. `gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro --region=us-west1`

```
NAME   LOCATION   MASTER_VERSION   MASTER_IP        MACHINE_TYPE   NODE_VERSION      NUM_NODES   STATUS
kubia  us-west1   1.18.16-gke.502  104.196.236.204  e2-micro       1.18.16-gke.502   3           RUNNING
```

4. Create a persistent volume

   `gcloud compute disks create --size=10GiB --zone=us-west1-a mongodb`

```
NAME     ZONE        SIZE_GB  TYPE          STATUS
mongodb  us-west1-a  10       pd-standard   READY
```

Implementation

5. Create a yaml file named mongodb-deployment.yaml

```yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: mongodb-deployment
spec:
    selector:
        matchLabels:
            app: mongodb
    strategy:
        type: Recreate
    template:
        metadata:
            labels:
                app: mongodb
        spec:
            containers:
                # by default, the image is pulled from docker hub
                - image: mongo
                  name: mongo
                  ports:
                    - containerPort: 27017
                  volumeMounts:
                    - name: mongodb-data
                      mountPath: /data/db
            volumes:
                - name: mongodb-data
                  gcePersistentDisk:
                    pdName: mongodb
                    fsType: ext4
```

Implementation

6. Create a MongoDB deployment pod:

```
kubectl apply -f mongodb-deployment.yaml
```

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f mongodb-deployment.yaml
deployment.apps/mongodb-deployment created
```

7. Check if the pod is running: kubectl get pods

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl get pods
NAME                                 READY   STATUS    RESTARTS   AGE
mongodb-deployment-554cbb9965-lh569  1/1     Running   0          57s
```

8. Create a yaml file named mongodb-service.yaml to create a service to access from the outside

Implementation

```
apiVersion: v1
kind: Service
metadata:
    name: mongodb-service
spec:
    type: LoadBalancer
    ports:
        # service port in cluster
        - port: 27017
        # port to contact inside container
          targetPort: 27017
    selector:
        app: mongodb
```

9. Create a service for MongoDB: `kubectl apply -f mongodb-service.yaml`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl apply -f mongodb-service.yaml
service/mongodb-service created
```

Implementation

10. Check if the service is running: `kubectl get svc`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl get svc
NAME              TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      10.3.240.1     <none>           443/TCP           22m
mongodb-service   LoadBalancer   10.3.245.57    35.230.71.134    27017:32306/TCP   48s
```

11. Check if MongoDB connection with external IP is working

    `kubectl exec -it <mongo-db-deployment pod name> -- bash`

```
montaos19518@cloudshell:~ (cs571-new)$ kubectl exec -it mongodb-deployment-554cbb9965-lh569 -- bash
root@mongodb-deployment-554cbb9965-lh569:/#
```

12. `mongo <external-ip>`

    Output should display info about MongoDB and its connection

13. `exit`
14. Install mongoose: `npm install mongoose`

Implementation

15. Insert records in MongoDB: `node`

```
montaos19518@cloudshell:~ (cs571-306821)$ node
Welcome to Node.js v12.14.1.
Type ".help" for more information.
>
```

Implementation

16.

```
> var MongoClient = require('mongodb').MongoClient;
undefined
> var url = "mongodb://35.230.71.134/mydb"
undefined
> // Connect to db
undefined
>
> MongoClient.connect(url, { useNewUrlParser: true, useUnifiedTopology: true },
... function(err, client) {
.....   if(err)
.....       throw err;
.....
.....           // create a document to be inserted
.....           var db = client.db("studentdb");
.....           const docs = [
.....       { student_id: 11111, student_name: "Bruce Lee", grade: 84 },
.....       { student_id: 22222, student_name: "Jackie Chen", grade: 93 },
.....       { student_id: 33333, student_name: "Jet Li", grade: 88 }
.....     ]
.....     db.collection("students").insertMany(docs, function(err, res) {
.......       if(err) throw err;
.......       console.log(res.insertedCount);
.......       client.close();
.......     });
.....           db.collection("students").findOne({"student_id": 11111},
.......     function(err, result) {
.........       console.log(result);
.........     });
..... });
undefined
> null
3
```

Implementation

17. Create studentServer.js
18. Create Dockerfile

```
montaos19518@cloudshell:~ (cs571-new)$ vi Dockerfile
montaos19518@cloudshell:~ (cs571-new)$ cat Dockerfile
FROM node:7
ADD studentServer.js /studentServer.js
ENTRYPOINT ["node", "studentServer.js"]
RUN npm install mongodb
```

19. Build the studentServer docker image

`docker build -t <Docker Hub ID>/studentserver .`

```
Successfully built ef8c751ff90d
Successfully tagged mva456/studentserver:latest
```

Implementation

20. Push the docker image: `docker push <Docker Hub ID>/studentserver`

21. Create bookshelf.py which contains flask server and API responses
22. Create requirements.txt to include which files to install for the flask server to work

Implementation

```
montaos19518@cloudshell:~ (cs571-306821)$ cat requirements.txt
Flask>=1.1.1,<1.2
flask-restplus>=0.13,<0.14
Flask-SSLify>=0.1.5,<0.2
Flask-Admin>=1.5.3,<1.6
gunicorn>=19,<20
Flask-PyMongo
```

23. Create a Dockerfile

```
montaos19518@cloudshell:~ (cs571-306821)$ cat Dockerfile
FROM python:alpine3.7
COPY . /app
WORKDIR /app
RUN pip install -r requirements.txt
ENV PORT 5000
EXPOSE 5000
ENTRYPOINT ["python3.7"]
CMD ["bookshelf.py"]
```

Implementation

24. Build the bookshelf app into a docker image

    `docker build -t <Docker Hub ID>/bookshelf .`

```
Successfully built f55034334358
Successfully tagged mva456/bookshelf:latest
```

25. Push the docker image to Docker Hub: `docker push <Docker Hub ID>/bookshelf`

```
montaos19518@cloudshell:~ (cs571-new)$ docker push mva456/bookshelf
Using default tag: latest
The push refers to repository [docker.io/mva456/bookshelf]
74a40ac8f6a8: Pushed
43175facf1b0: Pushed
66b6213aba6c: Pushed
5fa31f02caa8: Mounted from library/python
88e61e328a3c: Mounted from library/python
9b77965e1d3f: Mounted from library/python
50f8b07e9421: Mounted from library/python
629164d914fc: Mounted from library/python
latest: digest: sha256:d70b83c78659d13e55792c7f66f5081087750a32fa9f05ff24afd469f5d6c232 size: 2000
```

Implementation

```
montaos19518@cloudshell:~ (cs571-new)$ vi studentserver-configmap.yaml
montaos19518@cloudshell:~ (cs571-new)$ cat studentserver-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
    name: studentserver-config
data:
    MONGO_URL: 35.230.71.134
    MONGO_DATABASE: mydb
```

```
montaos19518@cloudshell:~ (cs571-new)$ vi bookshelf-configmap.yaml
montaos19518@cloudshell:~ (cs571-new)$ cat bookshelf-configmap.yaml
apiVersion: v1
kind: ConfigMap
metadata:
    name: bookshelf-config
data:
    # SERVICE_NAME.NAMESPACE.svc.cluster.local:SERVICE_PORT
    MONGO_URL: 35.230.71.134
    MONGO_DATABASE: mydb
```

Implementation

```
montaos19518@cloudshell:~ (cs571-new)$ vi studentserver-deployment.yaml
montaos19518@cloudshell:~ (cs571-new)$ cat studentserver-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: web
    labels:
        app: studentserver-deploy
spec:
    replicas: 1
    selector:
        matchLabels:
            app: web
    template:
        metadata:
            labels:
                app: web
        spec:
            containers:
                - image: mva456/studentserver
                  imagePullPolicy: Always
                  name: web
                  ports:
                    - containerPort: 8080
                  env:
                    - name: MONGO_URL
                      valueFrom:
                        configMapKeyRef:
                            name: studentserver-config
                            key: MONGO_URL
                    - name: MONGO_DATABASE
                      valueFrom:
                        configMapKeyRef:
                            name: studentserver-config
                            key: MONGO_DATABASE
```

Implementation

```
montaos19518@cloudshell:~ (cs571-new)$ vi bookshelf-deployment.yaml
montaos19518@cloudshell:~ (cs571-new)$ cat bookshelf-deployment.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
    name: bookshelf-deployment
    labels:
        app: bookshelf-deployment
spec:
    replicas: 1
    selector:
        matchLabels:
            app: bookshelf-deployment
    template:
        metadata:
            labels:
                app: bookshelf-deployment
        spec:
            containers:
                - image: mva456/bookshelf
                  imagePullPolicy: Always
                  name: bookshelf-deployment
                  ports:
                    - containerPort: 5000
                  env:
                    - name: MONGO_URL
                      valueFrom:
                        configMapKeyRef:
                            name: bookshelf-config
                            key: MONGO_URL
                    - name: MONGO_DATABASE
                      valueFrom:
                        configMapKeyRef:
                            name: bookshelf-config
                            key: MONGO_DATABASE
```

Implementation

```
montaos19518@cloudshell:~ (cs571-new)$ vi studentserver-service.yaml
montaos19518@cloudshell:~ (cs571-new)$ cat studentserver-service.yaml
apiVersion: v1
kind: Service
metadata:
    name: web
spec:
    type: LoadBalancer
    ports:
        # service port in cluster
        - port: 8080
          # port to contact inside container
          targetPort: 8080
    selector:
        app: web
```

```
montaos19518@cloudshell:~ (cs571-new)$ vi bookshelf-service.yaml
montaos19518@cloudshell:~ (cs571-new)$ cat bookshelf-service.yaml
apiVersion: v1
kind: Service
metadata:
    name: bookshelf-service
spec:
    type: LoadBalancer
    ports:
        # service port in cluster
        - port: 5000
          # port to contact inside container
          targetPort: 5000
    selector:
        app: bookshelf-deployment
```

Implementation

26. Start minikube: `minikube start`
27. Start ingress: `minikube addons enable ingress`

```
montaos19518@cloudshell:~ (cs571-new)$ minikube addons enable ingress
  - Using image us.gcr.io/k8s-artifacts-prod/ingress-nginx/controller:v0.40.2
  - Using image jettech/kube-webhook-certgen:v1.2.2
  - Using image jettech/kube-webhook-certgen:v1.3.0
* Verifying ingress addon...
* The 'ingress' addon is enabled
```

28. Create studentserver pods and service using previously created YAML files

```
kubectl apply -f studentserver-deployment.yaml
kubectl apply -f studentserver-configmap.yaml
kubectl apply -f studentserver-service.yaml
```

Implementation

29. Create bookshelf pods and service using previously create YAML files

```
kubectl apply -f bookshelf-deployment.yaml
kubectl apply -f bookshelf-configmap.yaml
kubectl apply -f bookshelf-service.yaml
```

30. Check if the pods are running: `kubectl get pods`

```
montaos19518@cloudshell:~ (cs571-306821)$ kubectl get pods
NAME                                    READY   STATUS    RESTARTS   AGE
bookshelf-deployment-7bcfdfc44b-xl7kl   1/1     Running   7          8m17s
web-7cc57c5b74-4n4kd                    1/1     Running   6          120m
```

Implementation

```
montaos19518@cloudshell:~ (cs571-306821)$ vi studentServerMongoIngress.yaml
montaos19518@cloudshell:~ (cs571-306821)$ cat studentServerMongoIngress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
    name: server
    annotations:
        nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
    rules:
        - host: cs571.project.com
          http:
            paths:
                - path: /studentserver(/|$)(.*)
                  pathType: Prefix
                  backend:
                    service:
                        name: web
                        port:
                            number: 8080
                - path: /bookshelf(/|$)(.*)
                  pathType: Prefix
                  backend:
                    service:
                        name: bookshelf-service
                        port:
                            number: 5000
```

Implementation

31. Create an ingress service:

    `kubectl apply -f studentServerMongoIngress.yaml`

```
montaos19518@cloudshell:~ (cs571-306821)$ kubectl apply -f studentServerMongoIngress.yaml
ingress.networking.k8s.io/server created
```

32. Get the ingress' address: `kubectl get ingress`

```
montaos19518@cloudshell:~ (cs571-306821)$ kubectl get ingress
NAME     CLASS    HOSTS              ADDRESS        PORTS   AGE
server   <none>   cs571.project.com  192.168.49.2   80      45s
```

33. Enter the command `sudo vi /etc/hosts`
34. In the last line, add `<address> cs571.project.com`

Implementation

```
montaos19518@cloudshell:~ (cs571-306821)$ cat /etc/hosts
# Kubernetes-managed hosts file.
127.0.0.1       localhost
::1     localhost ip6-localhost ip6-loopback
fe00::0 ip6-localnet
fe00::0 ip6-mcastprefix
fe00::1 ip6-allnodes
fe00::2 ip6-allrouters
172.17.0.4      cs-824671623995-default-boost-2mvvh
192.168.49.2 cs571.project.com
```

Test

Display student's score

```
curl cs571.project.com/studentserver/api/score?student_id=11111
```

```
montaos19518@cloudshell:~ (cs571-306821)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"_id":"6074b903888c4370acfc8890","student_id":11111,"student_name":"Bruce Lee","grade":84}
```

Display books: `curl cs571.project.com/bookshelf/books`

```
montaos19518@cloudshell:~ (cs571-306821)$ curl cs571.project.com/bookshelf/books
[]
```

Add book

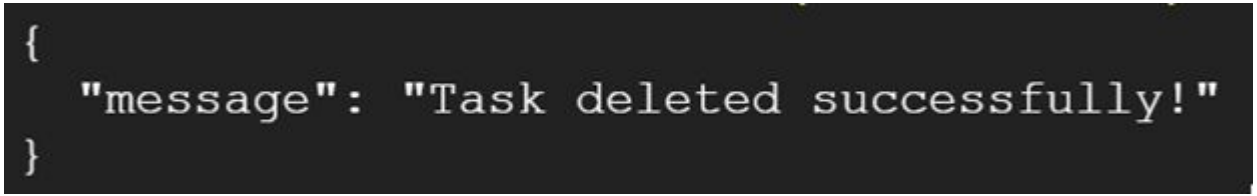
```
curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\":
\"unkown\", \"isbn\": \"123456\" }"
http://cs571.project.com/bookshelf/book
```

```
{
  "message": "Task saved successfully!"
}
```

Test

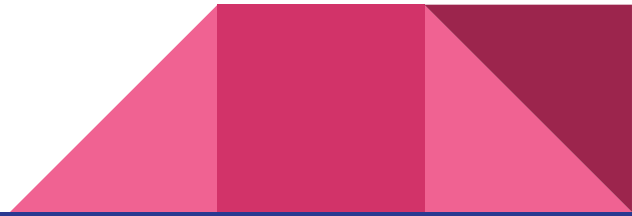
Delete book

```
curl -X DELETE cs571.project.com/bookshelf/book/<id>
```



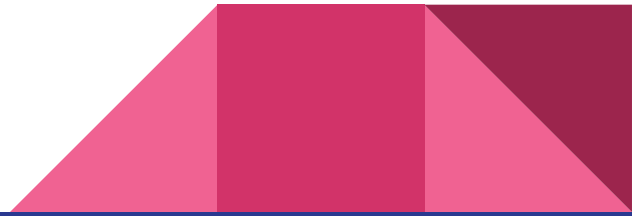
Enhancement ideas

- Implement different apps in GKE
  - Recipe collection
  - To-do list



Conclusion

- It is important to use technologies that are easy to scale for business using cloud computing
- Kubernetes makes a lot of things automated and it eliminated the need to do it manually
- Kubernetes is like an operating system in the cloud



Bibliography

Casey, K. (2020, September 17). *How to explain Kubernetes in plain English*. The Enterprisers Project. https://enterprisersproject.com/article/2017/10/how-explain-kubernetes-plain-english

detimo. (2019, November 18). *Python Flask: pros and cons*. DEV Community. https://dev.to/detimo/python-flask-pros-and-cons-1mlo

Sethi, K. (2019, November 11). *MongoDB vs. RDBMS*. Dzone.Com. https://dzone.com/articles/mongodb-vs-rdbms#:%7E:text=MongoDB%20is%20almost%20100%20times,comparison%20with%20the%20NoSQL%20databases.&text=MongoDB%20supports%20deep%20query%2Dability,nearly%20as%20powerful%20as%20SQL.

What is a RESTful API? (n.d.). MuleSoft. https://www.mulesoft.com/resources/api/restful-api#:%7E:text=One%20of%20the%20key%20advantages,the%20correct%20implementation%20of%20hypermedia.

What is Kubernetes? (2021, February 1). Kubernetes. https://kubernetes.io/docs/concepts/overview/what-is-kubernetes/