CS571 Signature Project

Presented by
Namrata Waybhase
19551

Subjects

1. Introduction
2. Steps
3. Results
4. Conclusion

1. Introduction

Topics

Technologies used in this project

- MongoDB
- Python Flask Web Framework
- REST API
- GKE

2. Objectives

Objective

1. To demonstrate deployment of full-stack application using various components offered by Google Cloud

2. In this project, we have used micro - service architecture because it is scalable, easy to maintain architecture

3. Instead of monolithic application we have used 3-tier system.

3. Implementation

Implementation

- Step 1: Pods

  Pods are used to run the applications. In this project, we worked on Student record and bookstore

- Step 2: Services

  We are using services to access external applications.

- Step 3: Persistent Volume

  Created Persistent Volume to store data using Mongodb database

- Step 4: Ingress

  Ingress is used to expose both applications under same domain but on different path

- Step 5: ConfigMaps

  It is used store MongoDB service address, in case MongoDB is down and restarts with a different service address, and with ConfigMaps, we don't need to build the docker image again with the new address

3. Test

Step 1 Create MongoDB using Persistent Volume on GKE, and insert records into it

Following commands are used:

1. **Create a cluster** on GKE : gcloud container clusters create kubia --num-nodes=1 --machine-type=e2-micro —region=us-central1
2. **Persistent Volume:** gcloud compute disks create --size=10GiB --zone=us-central1-a mongodb
3. Created mongodb-deployment.yaml file, mongodb-service.yaml
4. **Create a deployment for the mongoDB:** kubectl apply -f mongodb-deployment.yaml
5. **Create a service for the mongoDB:** kubectl apply -f mongodb-service.yaml
6. **Check pods are running:**kubectl get pods
7. **Check services are up: k**ubectl get svc
8. **Mongodb is functional**: kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash
9. **Create student.js**
10. **Install mongodb:** npm install mongodb
11. Run: **node student.js**
12. Result:

Step 2 Modify our studentServer to get records from MongoDB and deploy to GKE

Following commands are used:

1. **Create a studentServer**
2. **Create Dockerfile**
3. **Persistent Volume:** gcloud compute disks create --size=10GiB --zone=us-central1-a mongodb
4. Created mongodb-deployment.yaml file, mongodb-service.yaml
5. **Create a deployment for the mongoDB:** kubectl apply -f mongodb-deployment.yaml
6. **Create a service for the mongoDB:** kubectl apply -f mongodb-service.yaml
7. **Check pods are running:**kubectl get pods
8. **Check services are up: k**ubectl get svc
9. **Mongodb is functional**: kubectl exec -it mongodb-deployment-replace-with-your-pod-name -- bash
10. **Create student.js**
11. **Install mongodb:** npm install mongodb
12. Run: **node student.js**
13. Result:

Step 3 Create a python Flask bookshelf REST API and deploy on GKE

Following commands are used:

1. **Create a python bookshelf.py**
2. **Create Dockerfile**
3. **docker build -t YourDockerHubID/bookshelf .**

```
Successfully built 8589a45ecff9
Successfully tagged 19551/bookshelf:latest
```

4. Push the docker image to your dockerhub: **$ docker push 19551/bookshelf**

```
namrata_waybhase@cloudshell:~/Project (bold-bastion-309120)$ docker push 19551/bookshelf
Using default tag: latest
The push refers to repository [docker.io/19551/bookshelf]
16f8651b6643: Pushed
9c93888c76d6: 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:b927d8588a3eed5b5e2c9b88e4151b22ea3742cf73a8f215e94e28020d135cde size: 1789
```

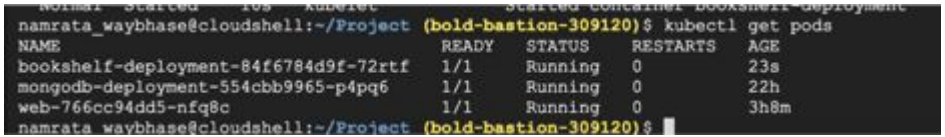
Step4 Create ConfigMap for both applications to store MongoDB URL and MongoDB name

Following commands are used:

1. Create a file studentserver-configmap.yaml
2. Create a file bookshelf-configmap.yaml

Step5 Expose 2 application using ingress with Nginx, so we can put them on the same Domain but different PATH

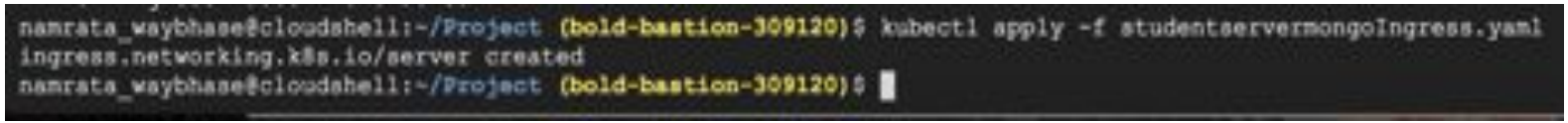
1. Create studentserver-deployment.yaml: **$ vi studentserver-deployment.yaml**
2. Create bookshelf-deployment.yaml: **$ vi bookshelf-deployment.yaml**
3. Create studentserver-service.yaml: **$ vi studentserver-service.yaml**
4. Create bookshelf-service.yaml: **$ vi bookshelf-service.yaml**
5. **minikube start**
6. **minikube addons enable ingress**
7. kubectl get pods



Create an ingress service

Following commands are used:

1. vi studentservermongoIngress.yaml
2. kubectl apply -f studentservermongoIngress.yaml



3. Add Addreee to /etc/hosts

   vi /etc/hosts

4. curl cs571.project.com/studentserver/api/score?student_id=11111
5. curl cs571.project.com/bookshelf/books
6. Add a book curl -X POST -d "{\"book_name\": \"cloud computing\",\"book_author\": \"unkown\", \"isbn\": \"123456\" }" http://cs571.project.com/bookshelf/book
7. Delete a book curl -X DELETE cs571.project.com/bookshelf/book/id

You can try all the above command to test.

Conclusion

Conclusion

- Using different technologies like
- MongoDB + Python Flask Web Framework + REST API + GKE , we were able access student record and bookstore application
- We stored our data in Mongodb database
- We were able to access a student's score from student application
- We were able to list all the books, add book, update book and delete book from bookstore.

https://www.shmoop.com/the-hours/

References

References

1.https://npu85.npu.edu/~henry/npu/classes/kubernetes_in_action/configmap/slide/exercise_configmap.html

2.https://npu85.npu.edu/~henry/npu/classes/kubernetes_in_action/configmap/hw/q5/2021_spring/CS571_Signature_Project_Quan_Zhou.pdf