MongoDB + Python Flask Web Framework + REST API + GKE

Natnael Haile
ID: 20007
August 3, 2024

Table of Contents

Introduction

Design

Implementation

Test

Enhancement Ideas

Conclusion

References

Introduction: Objective

- Deploy and manage a MongoDB instance and two applications (Student Server and Bookshelf API) on Google Kubernetes Engine (GKE) using Persistent Volumes for data persistence.
- **Tools and Technologies:**
  - Google Kubernetes Engine (GKE): For orchestrating containerized applications.
  - MongoDB: NoSQL database for storing data.
  - Node.js: Backend server for Student Server.
  - Python Flask: Framework for Bookshelf API.
  - Docker: Containerization of applications.
  - Nginx Ingress: For managing external access to the services.

Introduction: Key Components

- **Persistent Storage with GKE:**
  - Creating and attaching Persistent Volumes to MongoDB deployment.
- **Application Deployment:**
  - Deploying a Node.js application (Student Server) and a Python Flask application (Bookshelf API) on GKE.
- **Service Exposure:**
  - Using Kubernetes Services and Ingress to expose the applications to external traffic.
- **Data Management:**
  - Connecting applications to MongoDB for data storage and retrieval.
  - Implementing CRUD operations in the Bookshelf API.

Introduction: Highlights

- **Scalable Architecture:** Leveraging Kubernetes for scalable and resilient application deployment**.**
- **Persistent Data:** Ensuring data persistence through Google Cloud persistent disks.
- **Seamless Integration:** Connecting multiple applications with MongoDB for robust data management.
- **User-Friendly Interface:** Providing RESTful APIs for interaction with the applications.

Design: System Architecture

- **Google Kubernetes Engine (GKE):**
  - Manages the deployment, scaling, and operations of containerized applications.
  - Provides high availability and resilience through Kubernetes clusters.
- **Persistent Storage:**
  - Google Cloud Persistent Disks for MongoDB to ensure data persistence across pod restarts.
- **Containerization:**
  - Docker containers for both Student Server (Node.js) and Bookshelf API (Python Flask).
- **Service Exposure:**
  - Kubernetes Services (LoadBalancer) to expose MongoDB, Student Server, and Bookshelf API.
  - Nginx Ingress for managing external HTTP(S) traffic to the applications.

Design: Components

- **MongoDB:**
  - NoSQL database for storing student records and book information.
  - Deployed with persistent storage using Persistent Volume and Persistent Volume Claim.
- **Student Server (Node.js):**
  - Provides an API to fetch student scores from the MongoDB database.
- **Bookshelf API (Python Flask):**
  - Manages CRUD operations for book records in the MongoDB database.

Networking:

- **Internal Communication:**
  - Pods communicate within the cluster using ClusterIP services.
- **External Access:**
  - Nginx Ingress routes external traffic to the appropriate service based on URL paths.

Implementation

- Cluster and Persistent Volume Setup:
  - Create a GKE cluster.
  - Allocate Persistent Volume (PV) for MongoDB.
  - Apply the Persistent Volume, Persistent Volume Claim, mongodb-deployment YAML files to bind storage.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ gcloud compute disks create mongodb --size=10GiB --zone=us-west1-a
WARNING: You have selected a disk size of under [200GB]. This may result in poor I/O performance. For more information, see: https://developers.google.com/compute/docs/
disks#performance.
Created [https://www.googleapis.com/compute/v1/projects/cs570-big-data-analytics/zones/us-west1-a/disks/mongodb].
NAME: mongodb
ZONE: us-west1-a
SIZE_GB: 10
TYPE: pd-standard
STATUS: READY

New disks are unformatted. You must format and mount a disk before it
can be used. You can find instructions on how to do this at:

https://cloud.google.com/compute/docs/disks/add-persistent-disk#formatting

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f mongodb_deployment.yaml
persistentvolume/mongodb-pv created
persistentvolumeclaim/mongodb-pvc created
deployment.apps/mongodb-deployment configured
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Implementation

MongoDB Service Deployment:

- Deploy MongoDB service (a loadbalancer) that exposes your MongoDB deployment to the external network, allowing you to connect to MongoDB using the external IP address assigned by your cloud provider
- Apply the configuration and test the connection.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f mongodb_service.yaml
service/mongodb-service created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl get svc
NAME              TYPE           CLUSTER-IP      EXTERNAL-IP      PORT(S)           AGE
kubernetes        ClusterIP      34.118.224.1    <none>           443/TCP           110m
mongodb-service   LoadBalancer   34.118.227.17   35.227.133.217   27017:31592/TCP   5m55s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl exec -it mongodb-deployment-5c589898cb-vdgw6 -- bash
root@mongodb-deployment-5c589898cb-vdgw6:/#
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl exec -it mongodb-deployment-5c589898cb-vdgw6 -- bash
root@mongodb-deployment-5c589898cb-vdgw6:/# mongosh --host 35.227.133.217 --port 27017
Current Mongosh Log ID: 66addbace96a21036c149f47
Connecting to:          mongodb://35.227.133.217:27017/?directConnection=true&appName=mongosh+2.2.10
Using MongoDB:          7.0.12
Using Mongosh:          2.2.10

For mongosh info see: https://docs.mongodb.com/mongodb-shell/


To help improve our products, anonymous usage data is collected and sent to MongoDB periodically (https://www.mongodb.com/legal/privacy-policy).
You can opt-out by running the disableTelemetry() command.

------
   The server generated these startup warnings when booting
   2024-08-03T07:12:55.717+00:00: Using the XFS filesystem is strongly recommended with the WiredTiger storage engine. See http://dochub.mongodb.org/core/prodnotes-file
system
   2024-08-03T07:12:56.897+00:00: Access control is not enabled for the database. Read and write access to data and configuration is unrestricted
   2024-08-03T07:12:56.898+00:00: vm.max_map_count is too low
------

test>
```

Implementation

- Insert records into MongoDB.

```
undefined
> run().catch(console.dir);
Promise {
  <pending>,
  [Symbol(async_id_symbol)]: 95,
  [Symbol(trigger_async_id_symbol)]: 51
}
> 3 documents were inserted
{
  _id: new ObjectId('66addd0651a2c00fb7b2af49'),
  student_id: 11111,
  student_name: 'Bruce Lee',
  grade: 84
}
```

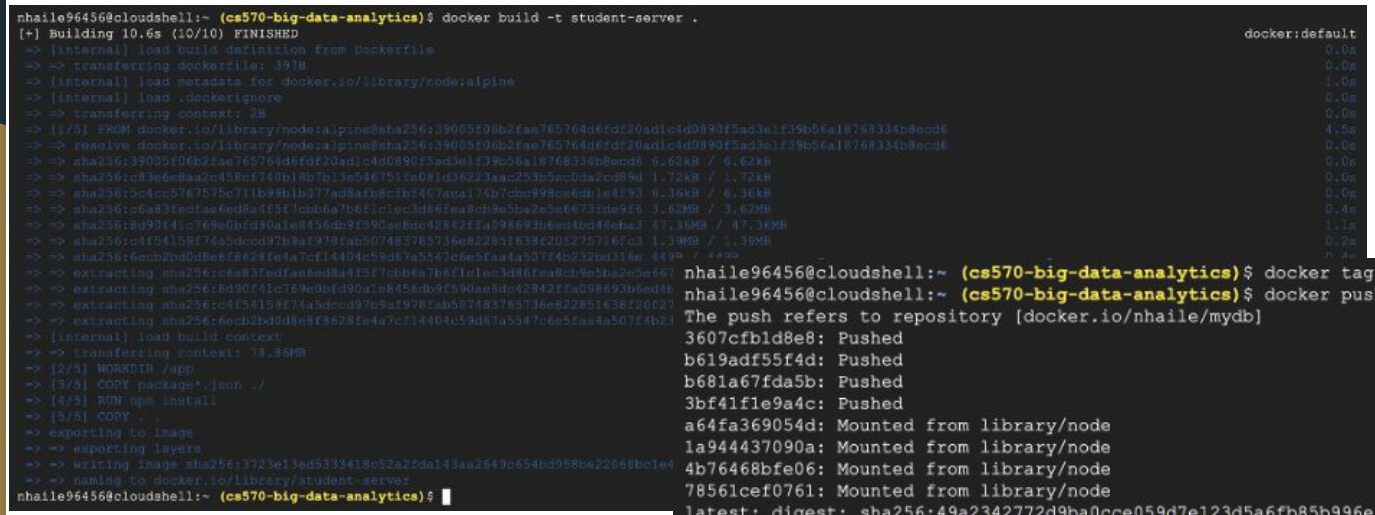
```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ npm install mongodb

added 12 packages in 3s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ node
Welcome to Node.js v20.15.1.
Type ".help" for more information.
> const { MongoClient } = require('mongodb');
undefined
> async function run() {
...   const url = "mongodb://35.227.133.217/studentdb";
...   const client = new MongoClient(url);
...   try {
...     // Connect to the MongoDB cluster
...     await client.connect();
...     // Specify the database and collection
...     const db = client.db("studentdb");
...     const collection = db.collection("students");
...     // Create documents to be inserted
...     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 }
...     ];
...     // Insert the documents
...     const insertResult = await collection.insertMany(docs);
...     console.log(`${insertResult.insertedCount} documents were inserted`);
...     // Find one document
...     const result = await collection.findOne({ student_id: 11111 });
...     console.log(result);
...   } finally {
...     // Close the connection
...     await client.close();
...   }
... }
undefined
> run().catch(console.dir);
Promise {
```

Implementation

Deploy Student Server:

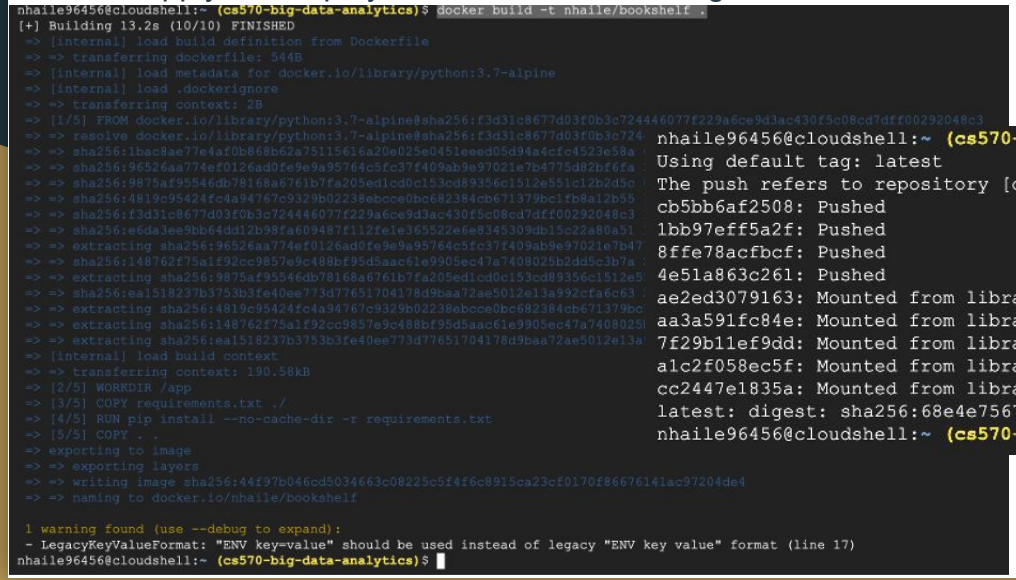
- Create studentServer.js and Dockerfile.
- Build and push Docker image to a registry.
- Apply the Deployment and Service configurations.

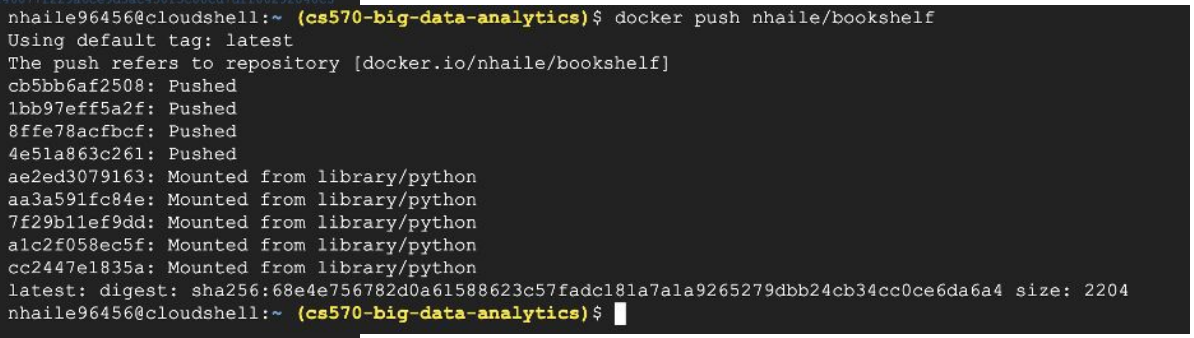


Implementation

Deploy Bookshelf API:

- Create bookshelf.py, Dockerfile, and requirements.txt.
- Build and push Docker image to a registry.
- Apply the Deployment and Service configurations.





Implementation

Build and apply ConfigMap and loadbalancer for Both Applications.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f studentServer_configmap.yaml
configmap/studentserver-config created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f studentServer_deployment.yaml
deployment.apps/web created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f studentServer_service.yaml
service/web created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f bookshelf_configmap.yaml
configmap/bookshelf-config created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f bookshelf_deployment.yaml
deployment.apps/bookshelf-deployment created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f bookshelf_service.yaml
service/bookshelf-service created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Implementation

- Start minikube and enable the Nginx ingress controller in Minikube.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ minikube start
* minikube v1.33.1 on Ubuntu 22.04 (amd64)
  - MINIKUBE_FORCE_SYSTEMD=true
  - MINIKUBE_HOME=/google/minikube
  - MINIKUBE_WANTUPDATENOTIFICATION=false
* Automatically selected the docker driver. Other choices: none, ssh
* Using Docker driver with root privileges
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ minikube addons enable ingress
* ingress is an addon maintained by Kubernetes. For any concerns contact minikube on GitHub.
You can view the list of minikube maintainers at: https://github.com/kubernetes/minikube/blob/master/OWNERS
  - Using image registry.k8s.io/ingress-nginx/controller:v1.10.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
  - Using image registry.k8s.io/ingress-nginx/kube-webhook-certgen:v1.4.1
* Verifying ingress addon...
* The 'ingress' addon is enabled
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
* Configuring bridge CNI (Container Networking Interface) ...
* Verifying Kubernetes components...
  - Using image gcr.io/k8s-minikube/storage-provisioner:v5
* Enabled addons: storage-provisioner, default-storageclass
* Done! kubectl is now configured to use "minikube" cluster and "default" namespace by default
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Implementation

- Build and apply ingress service.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ vi studentservermongoIngress.yaml
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ cat studentservermongoIngress.yaml
apiVersion: networking.k8s.io/v1
kind: Ingress
metadata:
  name: server
  annotations:
    nginx.ingress.kubernetes.io/rewrite-target: /$2
spec:
                number: 8080
      - path: /bookshelf(/|$)(.*)
        pathType: ImplementationSpecific
        backend:
          service:
            name: bookshelf-service
            port:
              number: 5000

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl apply -f studentservermongoIngress.yaml
ingress.networking.k8s.io/server created
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl get ingress
NAME     CLASS    HOSTS               ADDRESS   PORTS   AGE
server   nginx    cs571.project.com             80      32s
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

ver(/|$)(.*)
ntationSpecific

Implementation

Update /etc/hosts

- This step updates the /etc/hosts file on your local machine to map the ingress address to the domain name cs571.project.com.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ minikube ip
192.168.49.2
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

- In the file /etc/hosts append the following line.
  192.168.49.2 cs571.project.com

Test

- Access the application. Try accessing a single students details and all the books in the bookshelf.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl cs571.project.com/studentserver/api/score?student_id=11111
{"student_id":11111,"student_name":"Bruce Lee","student_score":84)nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
Processing request...
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl http://cs571.project.com/bookshelf/books
[]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

- Since the bookshelf is empty, add a book.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl -X POST -H "Content-Type: application/json" -d '{"book_name": "cloud computing", "book_author": "unknown", "isbn": "123456"}' http://cs571.project.com/bookshelf/book
{
  "message": "Book saved successfully!"
}
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
[haile96456@cloudshell:~ (cs570-big-data-analytics)$ curl http://cs571.project.com/bookshelf/books
  {
    "Book Author": "unknown",
    "Book Name": "cloud computing",
    "ISBN": null,
    "id": "66adee3fe1b2a7796453241e"
  }
]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Test

- Update a book.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl -X PUT -H "Content-Type: application/json" -d '{"book_name": "UpdatedBook Name", "book_author": "Updated Autho
r", "isbn": "Updated ISBN"}' http://cs571.project.com/bookshelf/book/66adee3fe1b2a7796453241e
{
  "message": "Book updated successfully!"
}
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl http://cs571.project.com/bookshelf/books
[
  {
    "Book Author": "Updated Author",
    "Book Name": "UpdatedBook Name",
    "ISBN": null,
    "id": "66adee3fe1b2a7796453241e"
  }
]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

- Delete a book.

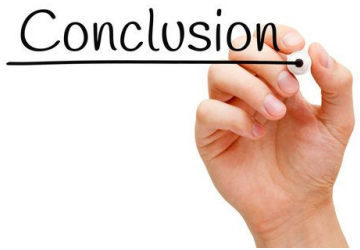
```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl -X DELETE http://cs571.project.com/bookshelf/book/66adee3fe1b2a7796453241e
{
  "message": "Book deleted successfully!"
}
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl http://cs571.project.com/bookshelf/books
[]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

Enhancement Ideas

- Implement auto-scaling for both Student Server and Bookshelf API to handle increased load dynamically
- Integrate tools like Prometheus and Grafana for real-time monitoring and visualization of application performance metrics
- Set up a continuous integration and continuous deployment pipeline using tools like Jenkins or GitHub Actions to automate the build, test, and deployment process
- Implement role-based access control (RBAC) and network policies in Kubernetes to enhance the security of the deployments
- Optimize MongoDB queries and indexing to improve performance and reduce response times for database operations

Conclusion

- Successfully deployed two microservices (Student Server and Bookshelf API) on Google Kubernetes Engine using Docker containers and persistent storage
- Achieved seamless integration of the services with MongoDB for data storage and management
- Effectively utilized Kubernetes Ingress to manage and route external traffic to the appropriate services based on URL paths
- Enhanced operational efficiency through automated deployment and scalability features provided by Kubernetes
- Identified several enhancement opportunities to further improve scalability, security, monitoring, and overall system performance



References

MongoDB docs

Flask Documentation

Nginx Ingress Controller for Kubernetes

Build a RESTful API with Flask, MongoDB, and Python

Python Tutorial: MongoDB RESTful API with Flask

GitHub Link

- https://github.com/cur10usityDrives/Cloud-Computing/tree/main/Kubernetes/MongoDB%2BPython-Flask-Web-Framework%2BREST-API%2BGKE


THANK YOU