

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-vdgv6 -- bash
root@mongodb-deployment-5c589898cb-vdgv6:/#
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ kubectl exec -it mongodb-deployment-5c589898cb-vdgv6 -- bash
root@mongodb-deployment-5c589898cb-vdgv6:/# mongosh --host 35.227.133.217 --port 27017
Current Mongosh Log ID: 66addbacc96a21036c149f47
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-filesystem
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('66add0651a2c00fb7b2af49'),
  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.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker build -t student-server .
[+] Building 10.6s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 397B
=> [internal] load metadata for docker.io/library/node:alpine
=> [internal] load .dockerignore
=> transferring context: 2B
=> [1/5] FROM docker.io/library/node:alpine@sha256:39005f06b2fae765764d6fd20ad1c4d0890f5ad3e1f39b56a18768334b8ecd6
=> resolve docker.io/library/node:alpine@sha256:39005f06b2fae765764d6fd20ad1c4d0890f5ad3e1f39b56a18768334b8ecd6
=> sha256:39005f06b2fae765764d6fd20ad1c4d0890f5ad3e1f39b56a18768334b8ecd6 6.62kB / 6.62kB
=> sha256:c83e6e8aa2c458cf740b18b7b13e546751fe081d36223aac253b5ec0da2cd89d 1.72kB / 1.72kB
=> sha256:5cdcc5767575c711b99b1b077ad8af08c1b54c7aca174p7cbe998ce6d81e4f93 6.36kB / 6.36kB
=> sha256:c8a83fcd6a6ed8a415f7cbb6a7b6f1c1ec3d86fa9cb9e5ba2c3e66731de9f5 3.62MB / 3.62MB
=> sha256:8d90f41c769e0bf490a1e8456db9f596aebdc42842ffa98693bted48c44e6a3 47.39MB / 47.39MB
=> sha256:c8f54159f74a5dcd97b9af978fab507483785736e822851638f20f275716fd3 1.39MB / 1.39MB
=> sha256:6e6cb2nd0d8e8f8628feda7c1f4404c59d67a5547c6e5faa4a507f4b232ba1316e 44.9B / 44.9B
=> extracting sha256:c8a83fcd6a6ed8a415f7cbb6a7b6f1c1ec3d86fa9cb9e5ba2c3e66731de9f5
=> extracting sha256:8d90f41c769e0bf490a1e8456db9f596aebdc42842ffa98693bted48c44e6a3
=> extracting sha256:c8f54159f74a5dcd97b9af978fab507483785736e822851638f20f275716fd3
=> extracting sha256:6e6cb2nd0d8e8f8628feda7c1f4404c59d67a5547c6e5faa4a507f4b23
=> [internal] load build context
=> transferring context: 78.86MB
=> [2/5] WORKDIR /app
=> [3/5] COPY package*.json ./
=> [4/5] RUN npm install
=> [5/5] COPY ...
=> exporting to image
=> exporting layers
=> writing image sha256:3723e13d333418c52a2fda143aa2649c654bd958ba22068b0e1e
=> naming to docker.io/library/student-server
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
docker:default
0.0s
0.0s
1.0s
0.0s
0.0s
4.5s
0.0s
0.0s
0.0s
0.0s
0.0s
0.4s
1.1s
0.2s
76.4s

nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker tag student-server:latest nhaile/mydb:latest
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker push nhaile/mydb:latest
The push refers to repository [docker.io/nhaile/mydb]
3607cfb1d8e8: Pushed
b619adf55f4d: Pushed
b681a67fda5b: Pushed
3bf41f1e9a4c: Pushed
a64fa369054d: Mounted from library/node
1a944437090a: Mounted from library/node
4b76468bfe06: Mounted from library/node
78561cef0761: Mounted from library/node
latest: digest: sha256:49a2342772d9ba0cce059d7e123d5a6fb85b996e5a775a2352809d5bad5ee4d6 size: 1995
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

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.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker build -t nhaile/bookshelf .
```

```
[+] Building 13.2s (10/10) FINISHED
=> [internal] load build definition from Dockerfile
=> => transferring dockerfile: 544B
=> [internal] load metadata for docker.io/library/python:3.7-alpine
=> [internal] load .dockerignore
=> => transferring context: 2B
=> [1/5] FROM docker.io/library/python:3.7-alpine@sha256:f3d31c8677d03f0b3c724446077f229a6ce9d3ac430f5c08cd7dff00292048c3
=> resolve docker.io/library/python:3.7-alpine@sha256:f3d31c8677d03f0b3c724
=> sha256:1bac8ae77e4af0b868b62a75115616a20e025e0451eed05d94a4cfc4523e58a
=> sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e97021e7b4775d82bf6fa
=> sha256:9875af95546db78168a6761b7fa205ed1cd0c153cd89356c1512e551c12b2d5c
=> sha256:4819c95424fc4a94767c9329b02238ebc0e0bc682384cb671379bcbfb8a12b55
=> sha256:f3d31c8677d03f0b3c724446077f229a6ce9d3ac430f5c08cd7dff00292048c3
=> sha256:e6da3ee3bb64dd12b98fa609487f112fe1e365522e6e8345309db15c22a80a51
=> extracting sha256:96526aa774ef0126ad0fe9e9a95764c5fc37f409ab9e97021e7b47
=> sha256:148762f75a1f92cc9857e9c488bf95d5aac61e9905ec47a7408025b2dd5c3b7a
=> extracting sha256:9875af95546db78168a6761b7fa205ed1cd0c153cd89356c1512e5
=> sha256:ea1518237b3753b3fe40ee773d77651704178d9baa72ae5012e13a992cfa6c63
=> extracting sha256:4819c95424fc4a94767c9329b02238ebc0e0bc682384cb671379bcb
=> extracting sha256:148762f75a1f92cc9857e9c488bf95d5aac61e9905ec47a7408025b
=> extracting sha256:ea1518237b3753b3fe40ee773d77651704178d9baa72ae5012e13a
=> [internal] load build context
=> => transferring context: 190.58kB
=> [2/5] WORKDIR /app
=> [3/5] COPY requirements.txt ./
=> [4/5] RUN pip install --no-cache-dir -r requirements.txt
=> [5/5] COPY . .
=> exporting to image
=> => exporting layers
=> => writing image sha256:44f97b046cd5034663c08225c5f4f6c8915ca23cf0170f86676141ac97204de4
=> => naming to docker.io/nhaile/bookshelf
```

```
1 warning found (use --debug to expand):
- LegacyKeyValueFormat: "ENV key=value" should be used instead of legacy "ENV key value" format (line 17)
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ docker push nhaile/bookshelf
```

Using default tag: latest

The push refers to repository [docker.io/nhaile/bookshelf]

cb5bb6af2508: Pushed

1bb97eff5a2f: Pushed

8ffe78acfbcf: Pushed

4e51a863c261: Pushed

ae2ed3079163: Mounted from library/python

aa3a591fc84e: Mounted from library/python

7f29b11ef99d: Mounted from library/python

alc2f058ec5f: Mounted from library/python

cc2447e1835a: Mounted from library/python

latest: digest: sha256:68e4e756782d0a61588623c57fadcl81a7ala9265279dbb24cb34cc0ce6da6a4 size: 2204

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

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
* Creating "minikube" virtual machine box using code in "minikube" cluster
```

```
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)$ 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)$
```

```
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:
  ingressClassName: nginx
```

```
    number: 8080
- path: /bookshelf(/|$)(.*)
  pathType: ImplementationSpecific
  backend:
    service:
      name: bookshelf-service
      port:
        number: 5000
```

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

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)$
```

```
[nhaile96456@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 Author", "isbn": "Updated ISBN"}' http://cs571.project.com/bookshelf/book/66adee3felb2a7796453241e
{
  "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": "66adee3felb2a7796453241e"
  }
]
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$
```

- Delete a book.

```
nhaile96456@cloudshell:~ (cs570-big-data-analytics)$ curl -X DELETE http://cs571.project.com/bookshelf/book/66adee3felb2a7796453241e
{
  "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

Conclusion



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>

