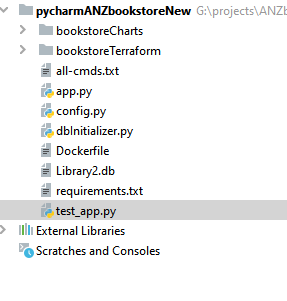
1. **Set up the project on local for testing and initial validation before deployment to GCP cloud**:

Download to local the repo from github and import to editor like pycharm and select virtual env or conda env with python 3.6 or 3.7:

git clone <https://github.com/debatosh99/ANZbookstoreNew.git>

The project structure should look similar to this :-

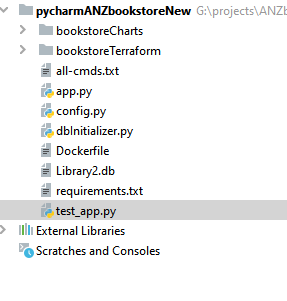


Change directory to /app and you will find requirements.txt.

Run below command to install the dependencies.

python.exe -m pip install -r requirements.txt

The Flask app files should look similar to below:-



**app.py** -> This is the main Flask app

**config.py** -> This contains the config for gunicorn, which is a light weight webserver compatible with flask framework.

**dbInitialiazer.py** -> This has some sqllite db functionalities like create db, table, list, insert sample records etc. This basically sets up the DB and inserts sample records.

**Library2.db** -> This is the sqllite DB which was pre created with sample records. Feel free to reinitialize this DB by running the dbInitializer.py code.

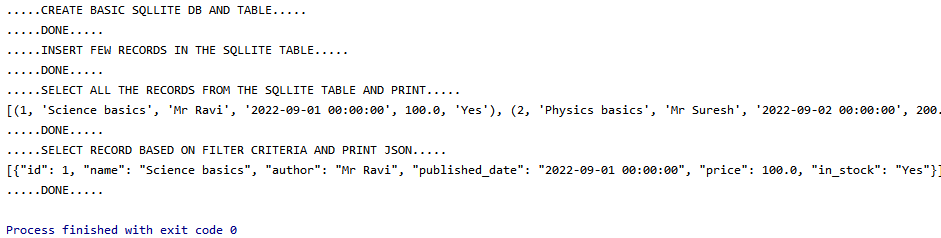
**requirements.txt** -> This contains all the code dependencies.

**test\_app.py** -> This is the unit test code.

**Dockerfile** -> This is the dockerfile to containerize the app.

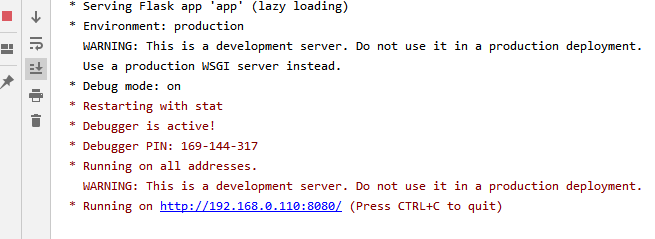
1. **Run the code locally on the system and run tests and validate.**

Run the dbInitializer.py to reinitialize the DB and create tables and insert sample records and run sample query.



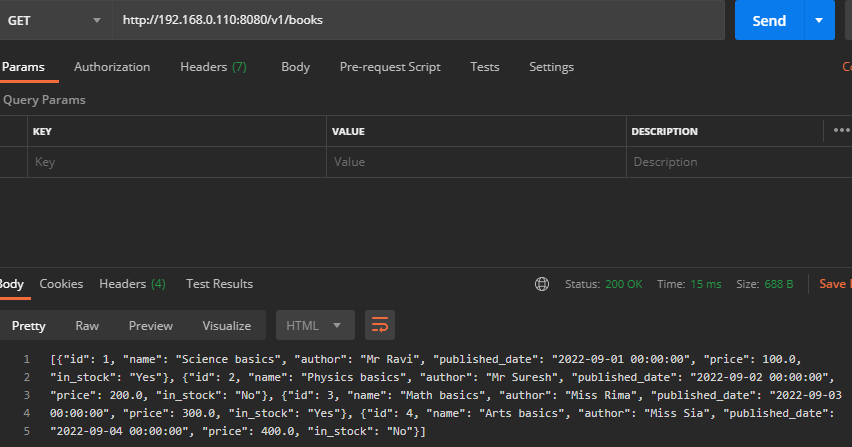
Once the DB and Tables are initialized, run the main Flask app.

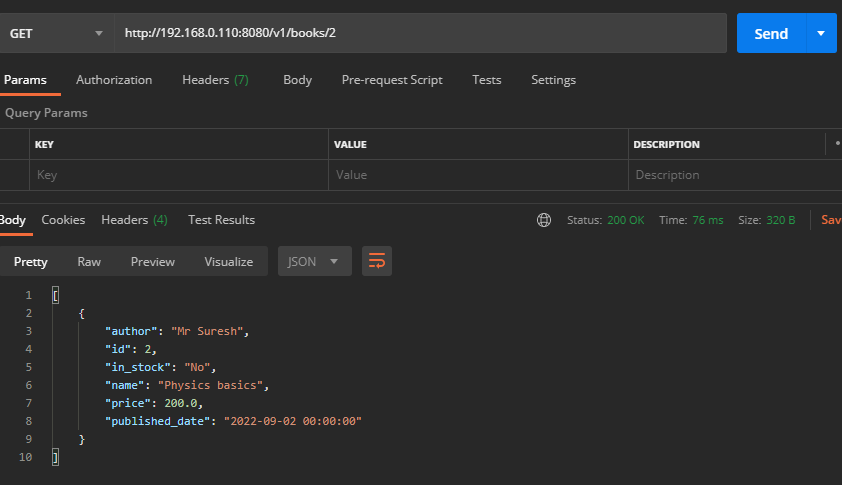
Make sure to turn off debug mode before deploying to formal environment.

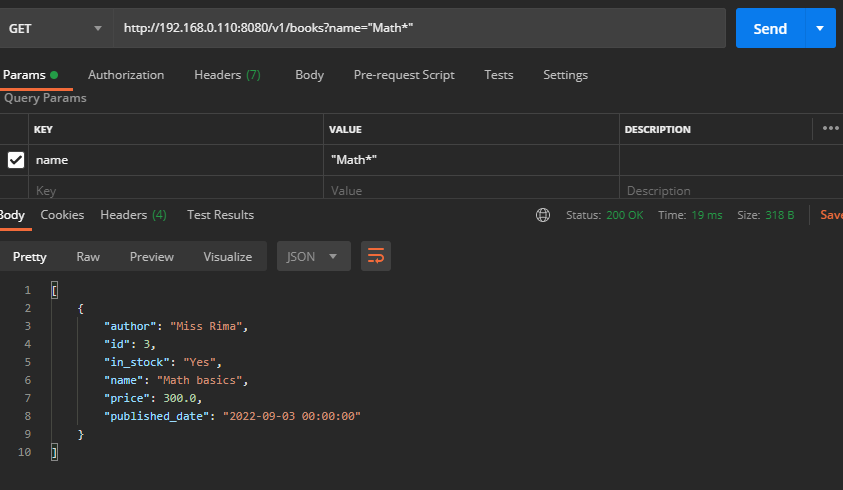


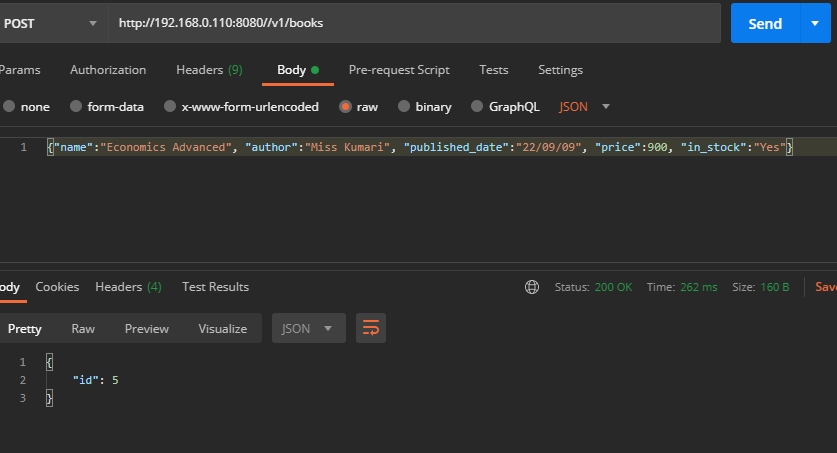
Once the web server is up, test out the rest end points using Postman and check for below end points with expected response.

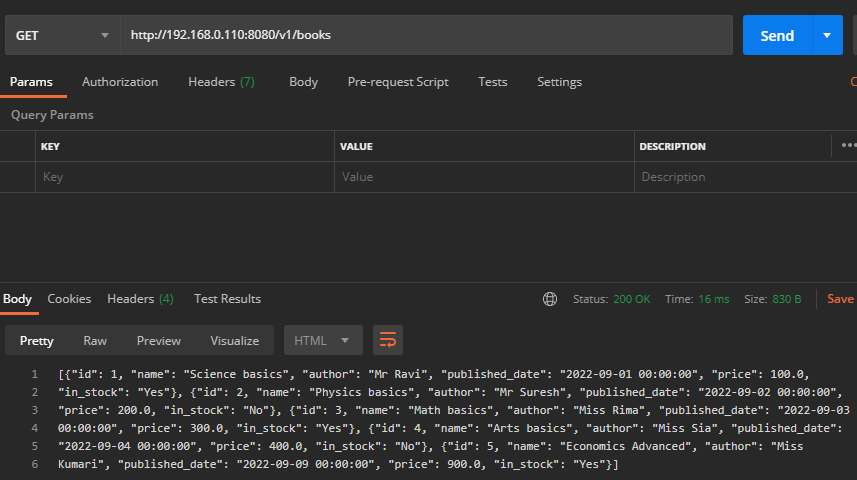
|  |  |  |
| --- | --- | --- |
| End point | HTTP Method | Description |
| /app-url/v1/books/ | GET | Return List of books in the Book Store |
| /app-url/v1/books/{book-id} | GET | Returns the book detail |
| /app-url/v1/books | POST | Insert a new book using the json data in the http body. |
| /app-url/v1/books?name=”abc\*” | GET | Partial Search: Returns all books with matching name URL parameter value. |



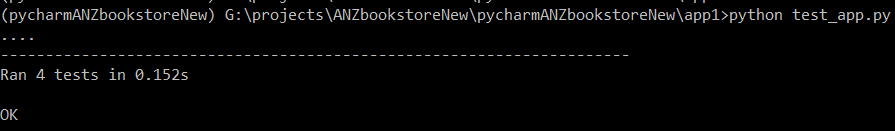


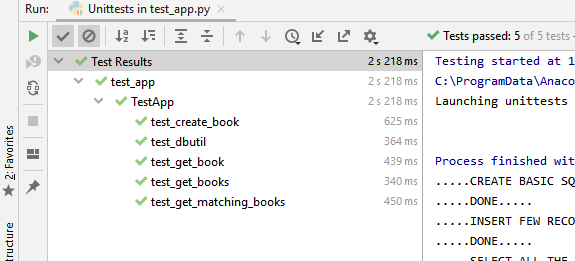






Run the unit tests:-

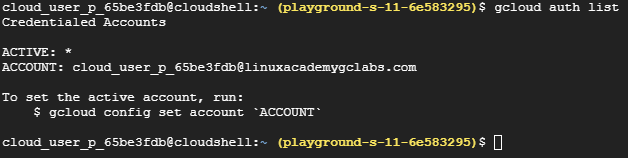




1. **Deployment to cloud GCP App Engine**

Login to GCP console and open cloud shell and run “gcloud auth list”

This should open up the browser and ask to authenticate.



Then to verify check the project(pre created with default vpc and iam) run “gcloud projects list”



We need to enable the cloud build api.

“gcloud services enable cloudbuild.googleapis.com”



Run git clone to get the code repo:-

git clone <https://github.com/debatosh99/ANZbookstoreNew.git>

Run below commands:-

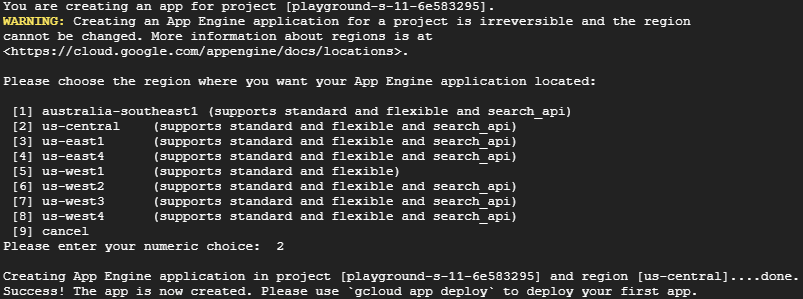
mkdir app

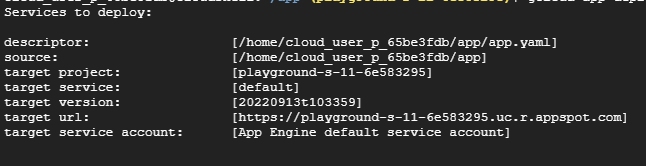
cd ANZbookstoreNew

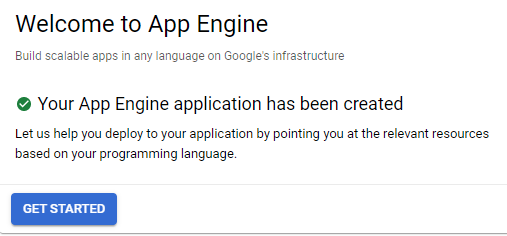
cp app.py app.yaml Dockerfile config.py dbInitializer.py Library2.db requirements.txt test\_app.py ~/app/

Create the App Engine application:-

gcloud app create --project=[project id]

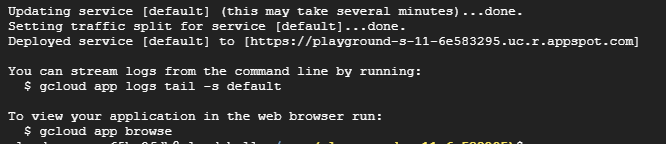






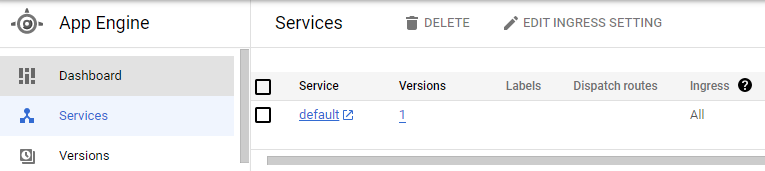
Deploy to App Engine:

gcloud app deploy

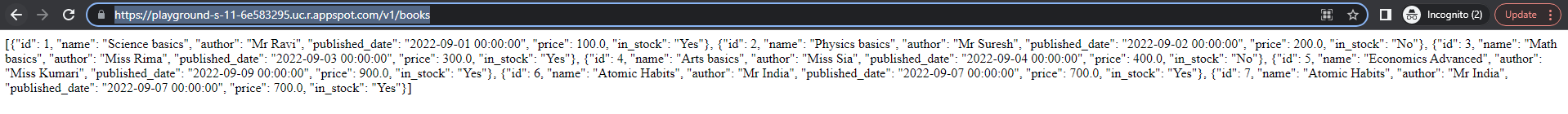


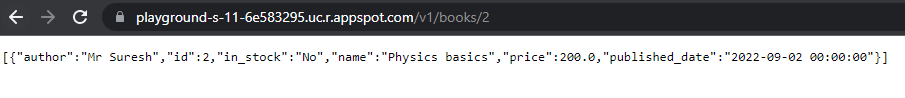
You can reach the App Engine deployed service at :-

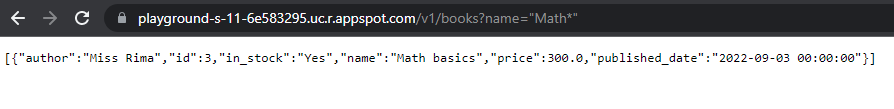
<https://playground-s-11-6e583295.uc.r.appspot.com>



Then verify the rest endpoints :-





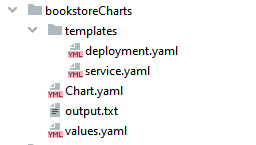


1. **Deployment to cloud GKE cluster using Terraform and Helm**

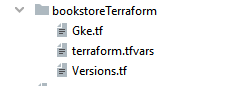
I have the Terraform code and Helm charts in separate folder in the project repo:-



The helm charts repo is simple with main two kubernetes resources includes the deployment and service and the values.yaml with configurable parameters.



The Terraform folder also contains the main terraform script along with the variables and versions separately.



**Next we move to provision the GKE Auto Pilot cluster using Terraform:-**

PROJECT\_ID: playground-s-11-6e583295

NAME: playground-s-11-6e583295

Change directory to Terraform scripts dir inside the cloned repo:

cd ANZbookstoreNew/bookstoreTerraform

Enable the container api before running Terraform:-

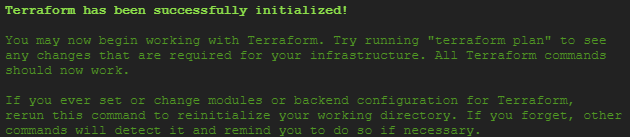
gcloud services enable container.googleapis.com

gcloud services enable compute.googleapis.com

Use below command so Terraform can use the credentials:-

gcloud auth application-default login

Then run (terraform init) to initialize Terraform and download the providers.



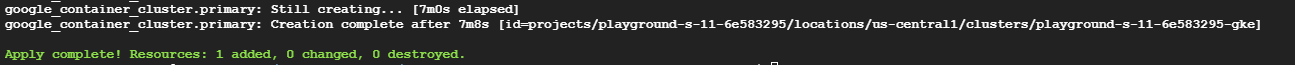
Then run (terraform validate) to validate.

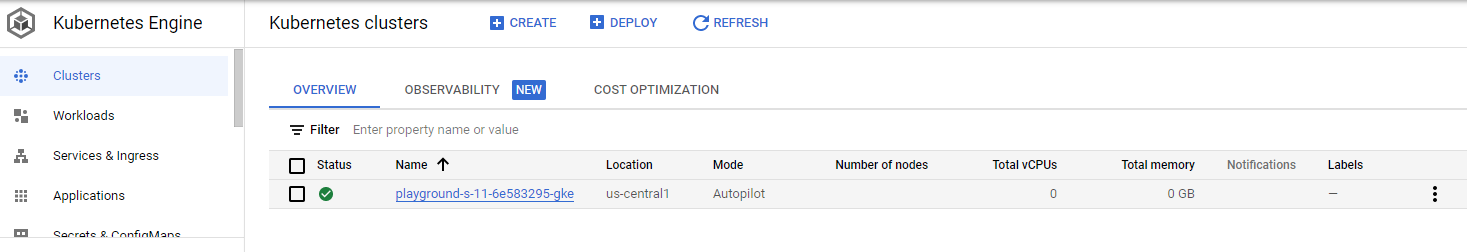


Then run (terraform plan) and (terraform apply)

This should try to create a GKE Auto Pilot cluster.

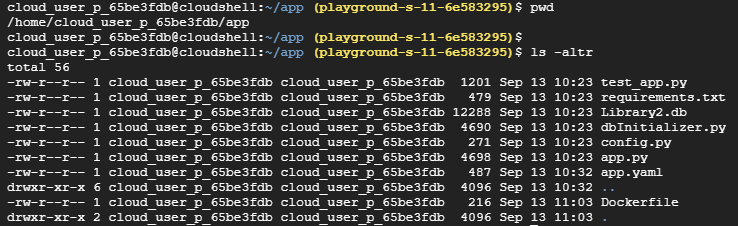
Creation complete after 7m8s [id=projects/playground-s-11-6e583295/locations/us-central1/clusters/playground-s-11-6e583295-gke]





1. **Creation of app docker image.**

Now its time to create the docker image of the project and push it to gcr container repo.



Once again enable the cloud build api:-

gcloud services enable cloudbuild.googleapis.com

gcloud builds --project playground-s-11-6e583295 submit --tag gcr.io/playground-s-11-6e583295/bookstore:v1 .

Once the build is successful the image is pushed to gcr and can be viewed in gcs store as well.

SOURCE: gs://playground-s-11-6e583295\_cloudbuild/source/1663067247.434076-b129f059c7c3425b9d2875e354f3ce5c.tgz

IMAGES: gcr.io/playground-s-11-6e583295/bookstore:v1



1. **Helm installation of the charts for app deploy to GKE**

Then run below command which enables kubectl to get credentials and address of the GKE cluster just created and cache it in kubeconfig.

gcloud container clusters get-credentials playground-s-11-6e583295-gke --region us-central1

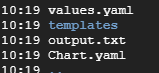


When run “kubectl get pods” there are no resources yet since we have not yet deployed our containerized app.



Change directory to the helm charts directory.

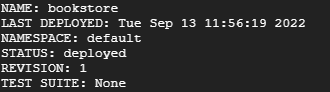
cd ANZbookstoreNew/bookstoreCharts



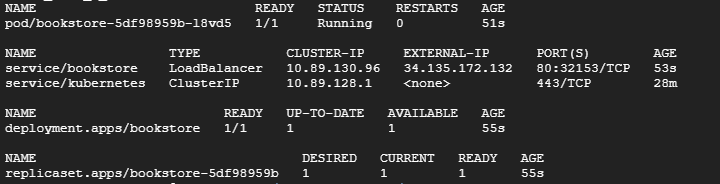
Change the image tag in the values.yaml to use the app image just pushed to gcr.

Then run the helm install command to deploy the app to GKE:-

helm install bookstore .



Then run the (kubectl get all) to verify all resources are created.

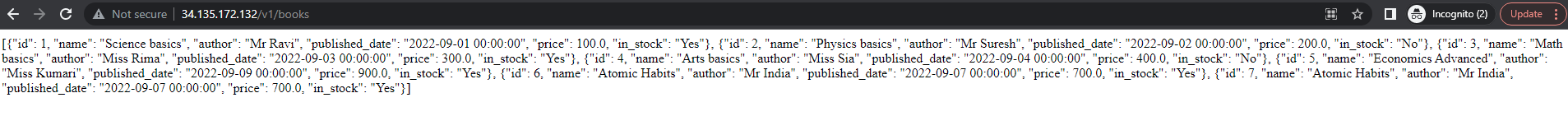


Then run (kubectl get svc) to list all services and check the EXTERNAL-IP of the bookstore load balancer type service to expose our app.

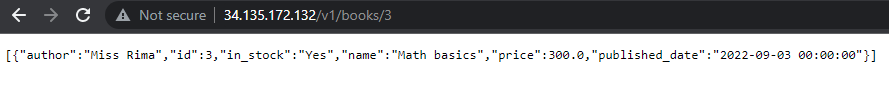


1. **The app can now be accessed at :**

<http://34.135.172.132/v1/books>



<http://34.135.172.132/v1/books/3>



[http://34.135.172.132/v1/books?name=%22Math\*%22](http://34.135.172.132/v1/books?name=%22Math*%22)

