Briksam Kasımoğlu-2102969

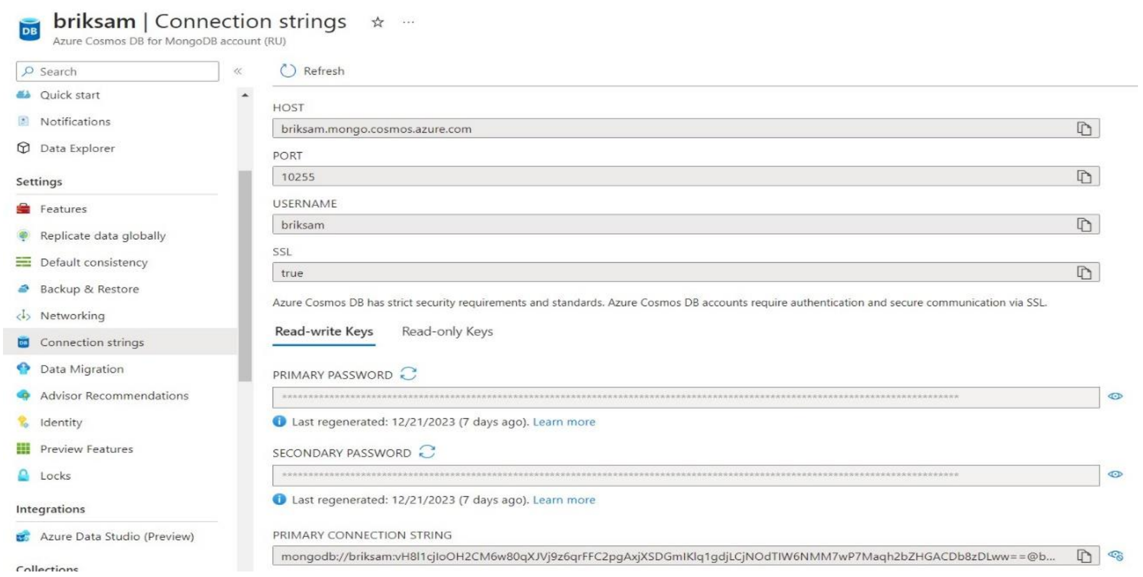
DATABASE SYSTEMS AND CLOUD COMPUTING

Project Proposal #2

Contents

Task-1: Setup MongoDB on Azure:

Open azure portal (https://portal.azure.com/) and click on "create resource" and then find "Azure Cosmos DB." Press on create and then choose "Request unit (RU) database account" which takes us to choose architecture page, choosing the azure for students the resource group and the location, setting up more configurations for the network if needed and then pressing review and create and waiting for the deployment to finish after which we press go to resource. Going to the connection strings tab we find the primary connection string to deploy our database 'Bookstore' from the local device.



HOST
briksam.mongo.cosmos.azure.com

PORT
10255

USERNAME
briksam

SSL
true

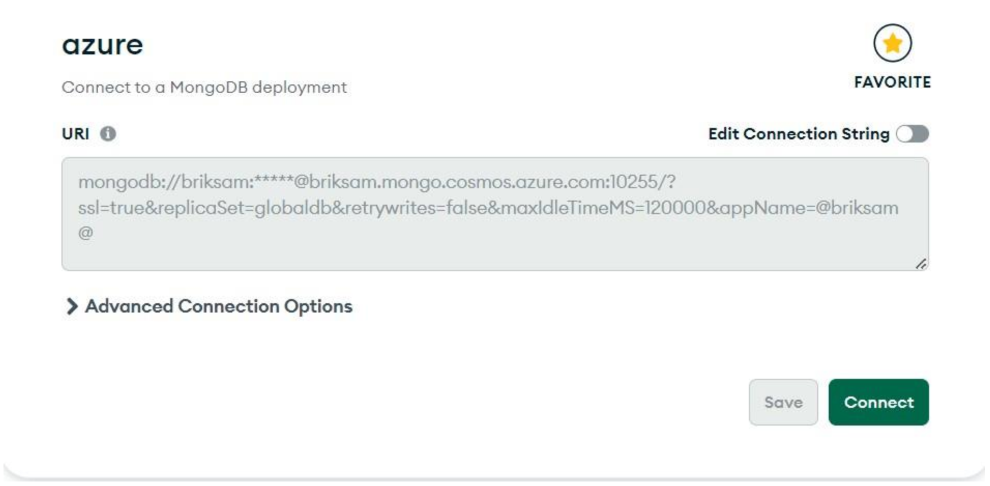
Azure Cosmos DB has strict security requirements and standards. Azure Cosmos DB accounts require authentication and secure communication via SSL.

PRIMARY PASSWORD
****
Last regenerated: 12/21/2023 (7 days ago). Learn more

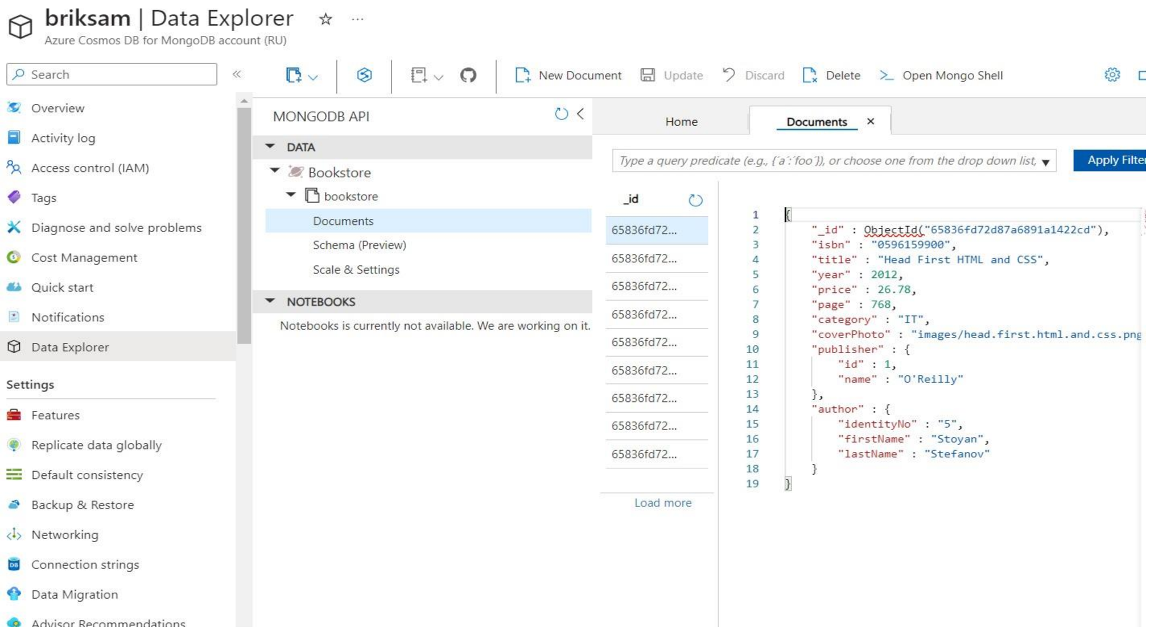
SECONDARY PASSWORD
****
Last regenerated: 12/21/2023 (7 days ago). Learn more

PRIMARY CONNECTION STRING
mongodb://briksam:vH8l1cjloOH2CM6w80qXJVj9z6qrFFC2pgAxjXSDGmIKlq1gdjLCjNOdTIW6NMM7wP7Maqh2bZHGACDb8zDLww==@b...

Using said connection string we connect to the CosmosDB instance from the MongoDB compass, and we import the Json file to it which in return when the azure website is refreshed, we find the data there.



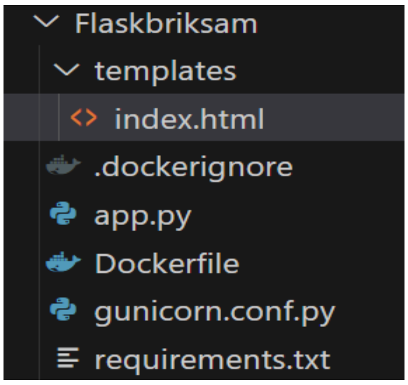
mongodb://briksam:*****@briksam.mongo.cosmos.azure.com:10255/?ssl=true&replicaSet=globaldb&retrywrites=false&maxIdleTimeMS=120000&appName=@briksam@



```
{
    "_id" : ObjectId("65836fd72d87a6891a1422cd"),
    "isbn" : "0596159900",
    "title" : "Head First HTML and CSS",
    "year" : 2012,
    "price" : 26.78,
    "page" : 768,
    "category" : "IT",
    "coverPhoto" : "images/head.first.html.and.css.png
    "publisher" : {
        "id" : 1,
        "name" : "O'Reilly"
    },
    "author" : {
        "identityNo" : "5",
        "firstName" : "Stoyan",
        "lastName" : "Stefanov"
    }
}
```

Task-2: Python Application:

The python script is included in the submission file under the name app.py in the folder Flaskbriksam along with the dockerfile, .dockerignore , requirements file and gunicorn file. Flask is used to ensure RESTFUL application and the dockerfiles to build the docker image and run it.

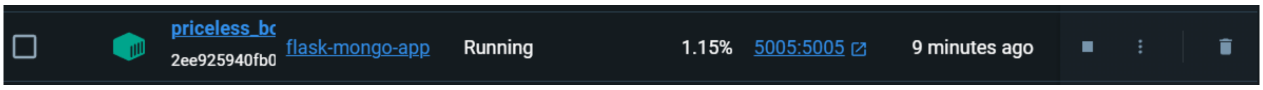


```
Flaskbriksam
  templates
    index.html
  .dockerignore
  app.py
  Dockerfile
  gunicorn.conf.py
  requirements.txt
```

Opening the folder in vs-code and using the terminal there or opening the command prompt and changing the directory to this folder and making sure that the docker engine is running we build and run the docker container and image by running the following two commands one after the other:

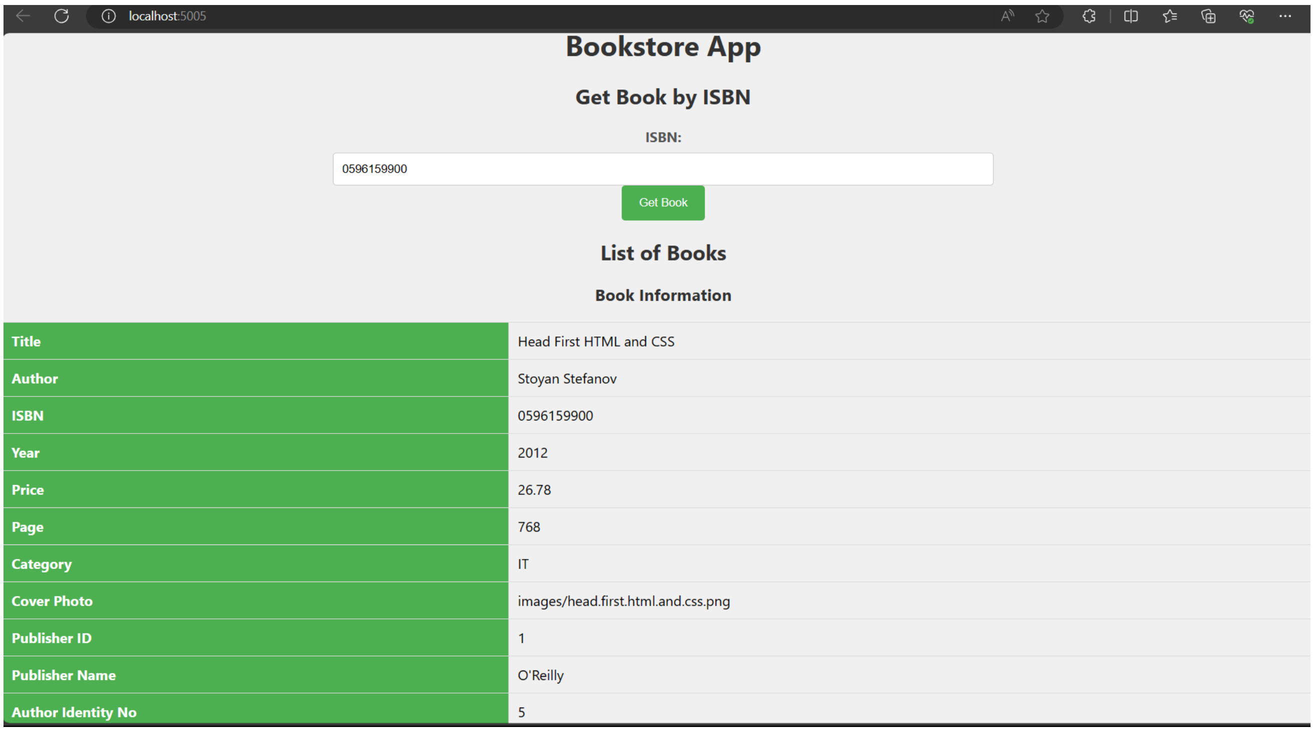
```
docker build -t flask-mongo-app .
docker run -d -p 5005:5005 flask-mongo-app
```

it might take a bit of time but then the app is Containerized using docker and the vs-code docker extension can be of use or opening the docker desktop the container can be seen there and the link to the application can be found and opened from there as well.



```
priceless_bc   flask-mongo-app   Running   1.15%   5005:5005   9 minutes ago
2ee925940fb0
```

To see if the application is running properly, we can open the link on the local device's browser, and it is going to take us immediately to a user interface that supports only one operation which is getting a book's information from the book's isbn:



localhost:5005

Bookstore App

Get Book by ISBN

ISBN:

0596159900

Get Book

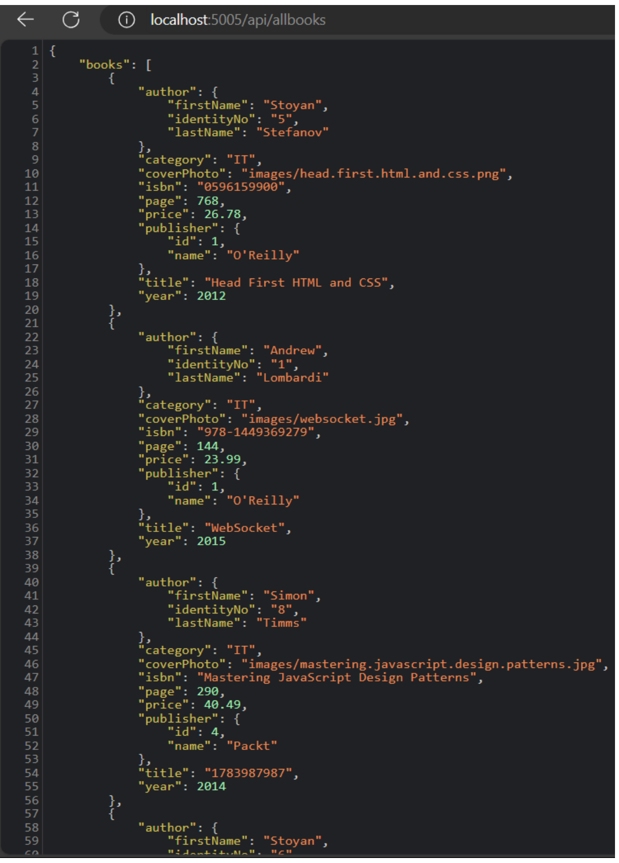
List of Books

Book Information

Title	Head First HTML and CSS
Author	Stoyan Stefanov
ISBN	0596159900
Year	2012
Price	26.78
Page	768
Category	IT
Cover Photo	images/head.first.html.and.css.png
Publisher ID	1
Publisher Name	O'Reilly
Author Identity No	5

If you want to get another book no need to refresh just type the isbn and press get book again and the new book will appear under the first book's information this way multiple book info can be displayed on one page if needed and if not needed then refreshing will be satisfactory.

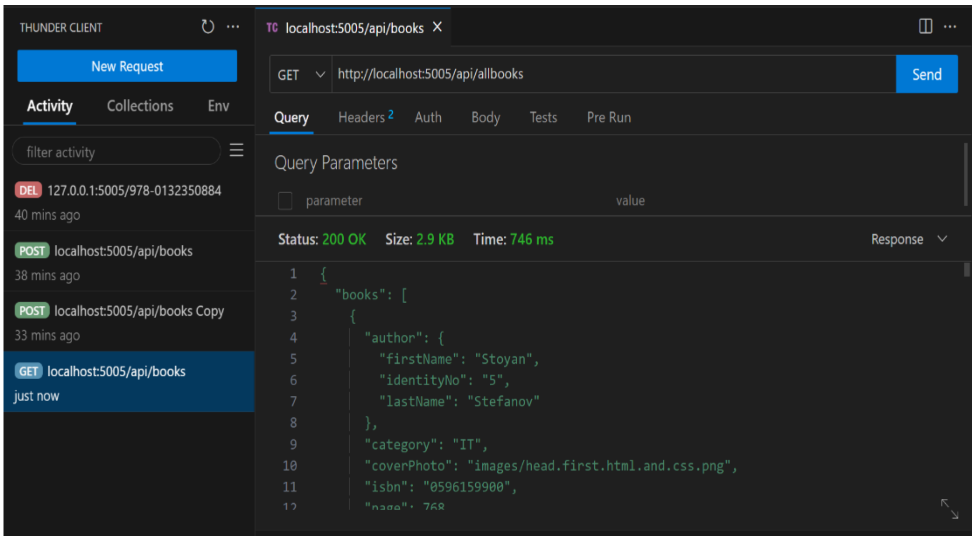
For accessing the other operations, a user interface was not created but it can be done through two ways one is the classic one by adjusting the link according to the route in the app.py and the operation needed ex) localhost:5005/api/allbooks is the link to get all the books in the database not just one based on a condition.



localhost:5005/api/allbooks

```
{
    "books": [
        {
            "author": {
                "firstName": "Stoyan",
                "identityNo": "5",
                "lastName": "Stefanov"
            },
            "category": "IT",
            "coverPhoto": "images/head.first.html.and.css.png",
            "isbn": "0596159900",
            "page": 768,
            "price": 26.78,
            "publisher": {
                "id": 1,
                "name": "O'Reilly"
            },
            "title": "Head First HTML and CSS",
            "year": 2012
        },
        {
            "author": {
                "firstName": "Andrew",
                "identityNo": "1",
                "lastName": "Lombardi"
            },
            "category": "IT",
            "coverPhoto": "images/websocket.jpg",
            "isbn": "978-1449369279",
            "page": 144,
            "price": 23.99,
            "publisher": {
                "id": 1,
                "name": "O'Reilly"
            },
            "title": "WebSocket",
            "year": 2015
        },
        {
            "author": {
                "firstName": "Simon",
                "identityNo": "8",
                "lastName": "Timms"
            },
            "category": "IT",
            "coverPhoto": "images/mastering.javascript.design.patterns.jpg",
            "isbn": "Mastering JavaScript Design Patterns",
            "page": 290,
            "price": 40.49,
            "publisher": {
                "id": 4,
                "name": "Packt"
            },
            "title": "1783987987",
            "year": 2014
        },
        {
            "author": {
                "firstName": "Stoyan",
```

and the other is an extension on vs-code called THUNDER CLIENT which can be used to test all the CRUD operations easily since it is user friendly.



DEL 127.0.0.1:5005/978-0132350884
40 mins ago

POST localhost:5005/api/books
38 mins ago

POST localhost:5005/api/books Copy
33 mins ago

GET localhost:5005/api/books
just now

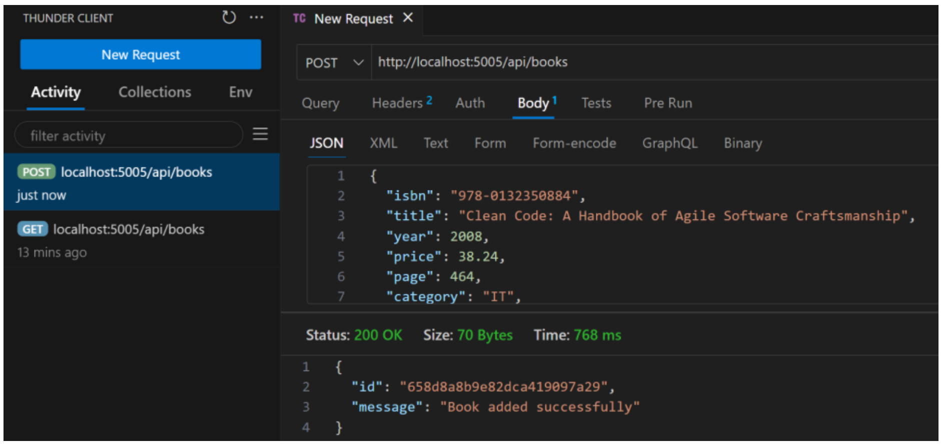
TC localhost:5005/api/books

GET http://localhost:5005/api/allbooks Send

Status: 200 OK Size: 2.9 KB Time: 746 ms

```
{
  "books": [
    {
      "author": {
        "firstName": "Stoyan",
        "identityNo": "5",
        "lastName": "Stefanov"
      },
      "category": "IT",
      "coverPhoto": "images/head.first.html.and.css.png",
      "isbn": "0596159900",
      "page": 768
```

All operations can be tested by changing the GET to the needed operation from the dropdown arrow.

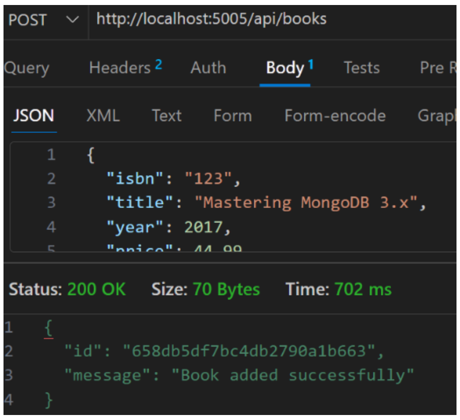


```
POST  http://localhost:5005/api/books
```

```
{
  "isbn": "978-0132350884",
  "title": "Clean Code: A Handbook of Agile Software Craftsmanship",
  "year": 2008,
  "price": 38.24,
  "page": 464,
  "category": "IT",
```

```
Status: 200 OK    Size: 70 Bytes    Time: 768 ms
{
  "id": "658d8a8b9e82dca419097a29",
  "message": "Book added successfully"
}
```

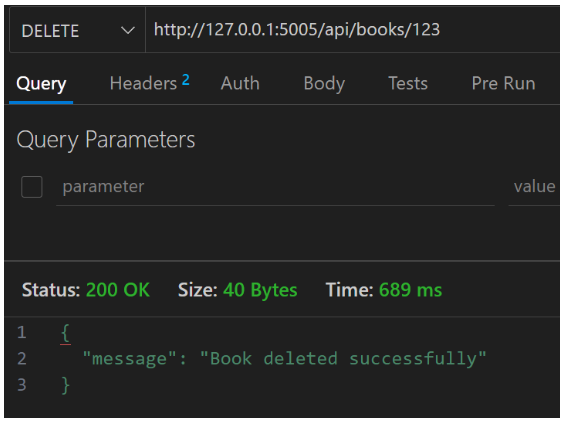
When the database is checked on azure the new document is added successfully there as well.
A dummy document was inserted to test the delete operation:



```
POST  http://localhost:5005/api/books
```

```
{
  "isbn": "123",
  "title": "Mastering MongoDB 3.x",
  "year": 2017,
  "price": 44.99
```

```
Status: 200 OK    Size: 70 Bytes    Time: 702 ms
{
  "id": "658db5df7bc4db2790a1b663",
  "message": "Book added successfully"
}
```

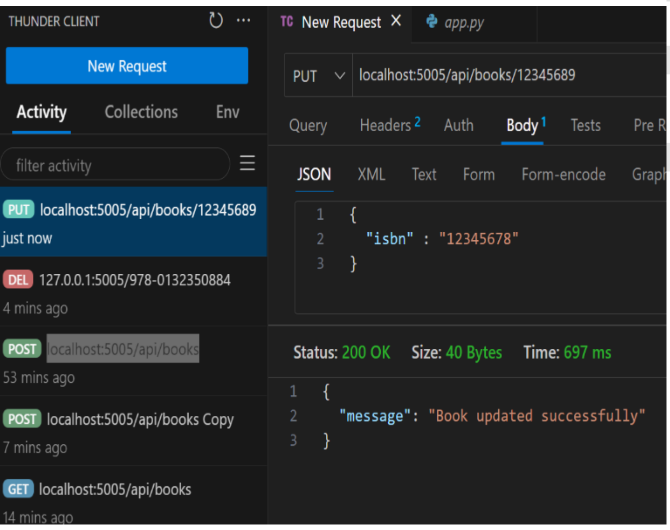


```
DELETE  http://127.0.0.1:5005/api/books/123
```

```
Status: 200 OK    Size: 40 Bytes    Time: 689 ms
{
  "message": "Book deleted successfully"
}
```

The instance appeared and got deleted from CosmosDB API successfully.

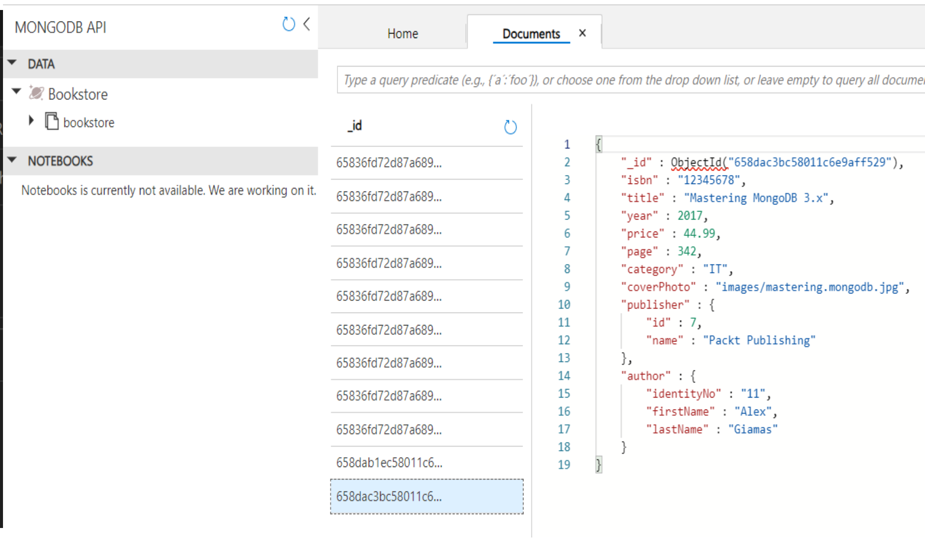
For the update operation a new document was inserted again but this time the isbn was updated successfully to 12345678



```
PUT  localhost:5005/api/books/12345689
```

```
{
  "isbn" : "12345678"
}
```

```
Status: 200 OK    Size: 40 Bytes    Time: 697 ms
{
  "message": "Book updated successfully"
}
```

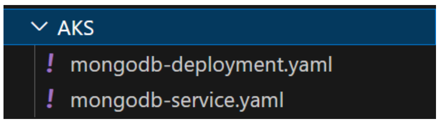


```
{
    "_id" : ObjectId("658dac3bc58011c6e9aff529"),
    "isbn" : "12345678",
    "title" : "Mastering MongoDB 3.x",
    "year" : 2017,
    "price" : 44.99,
    "page" : 342,
    "category" : "IT",
    "coverPhoto" : "images/mastering.mongodb.jpg",
    "publisher" : {
        "id" : 7,
        "name" : "Packt Publishing"
    },
    "author" : {
        "identityNo" : "11",
        "firstName" : "Alex",
        "lastName" : "Giamas"
    }
}
```

Task-3: Azure Kubernetes Service (AKS):

For creating an AKS cluster we again go to the portal (https://portal.azure.com/) and click on "create resource" and find "Azure Kubernetes Service (AKS)" and press create after adding resource group, cluster name, region and setting the AKS pricing tier to free and configuring the rest of the settings based on need press review+create and again wait for deployment then press go to resource, note(do not leave the cluster running because it spends credit from the 100 dollars).

For the deployment of MongoDB and the Kubernetes service the yaml files are included in the submission folder in the folder called AKS as follows:

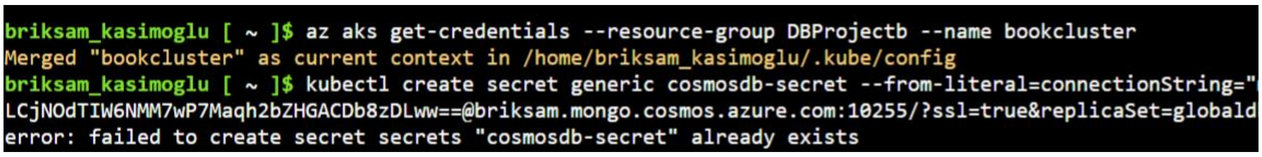


```
∨ AKS
  ! mongodb-deployment.yaml
  ! mongodb-service.yaml
```

Now the way to use these files to execute the deployment and the exposing of the service is by first of all after making sure the cluster is created opening azure cloud shell from the portal website again and waiting for the terminal to be connected.

After which a couple of commands are going to be run through the terminal

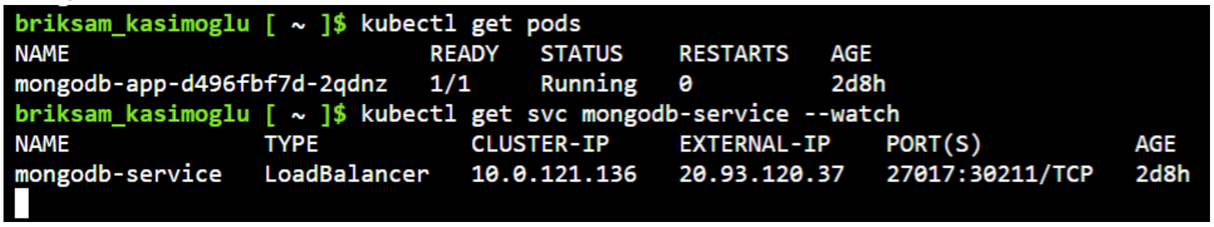
az aks get-credentials --resource-group DBProjectb --name bookcluster
to connect to the cluster
kubectl create secret generic cosmosdb-secret --from-literal=connectionString="mongodb://briksam:vH8l1cjIoOH2CM6w80qXJVj9z6qrFFC2pgAxjXSDGmIKlq1gdjLCjNOdTIW6NMM7wP7Maqh2bZHGACDb8zDLww==@briksam.mongo.cosmos.azure.com:10255/?ssl=true&replicaSet=globaldb&retrywrites=false&maxIdleTimeMS=120000&appName=@briksam@"
to embed the connection string



```
briksam_kasimoglu [ ~ ]$ az aks get-credentials --resource-group DBProjectb --name bookcluster
Merged "bookcluster" as current context in /home/briksam_kasimoglu/.kube/config
briksam_kasimoglu [ ~ ]$ kubectl create secret generic cosmosdb-secret --from-literal=connectionString="
LCjNOdTIW6NMM7wP7Maqh2bZHGACDb8zDLww==@briksam.mongo.cosmos.azure.com:10255/?ssl=true&replicaSet=globald
error: failed to create secret secrets "cosmosdb-secret" already exists
```

kubectl apply -f mongodb-deployment.yaml
kubectl get pods
kubectl apply -f mongodb-service.yaml
kubectl get svc mongodb-service --watch

running the above four commands in that order sequentially is then enough to complete the process get pods is for checking and get svc is for getting the external IP of the service to then access the exposed MongoDB instance.

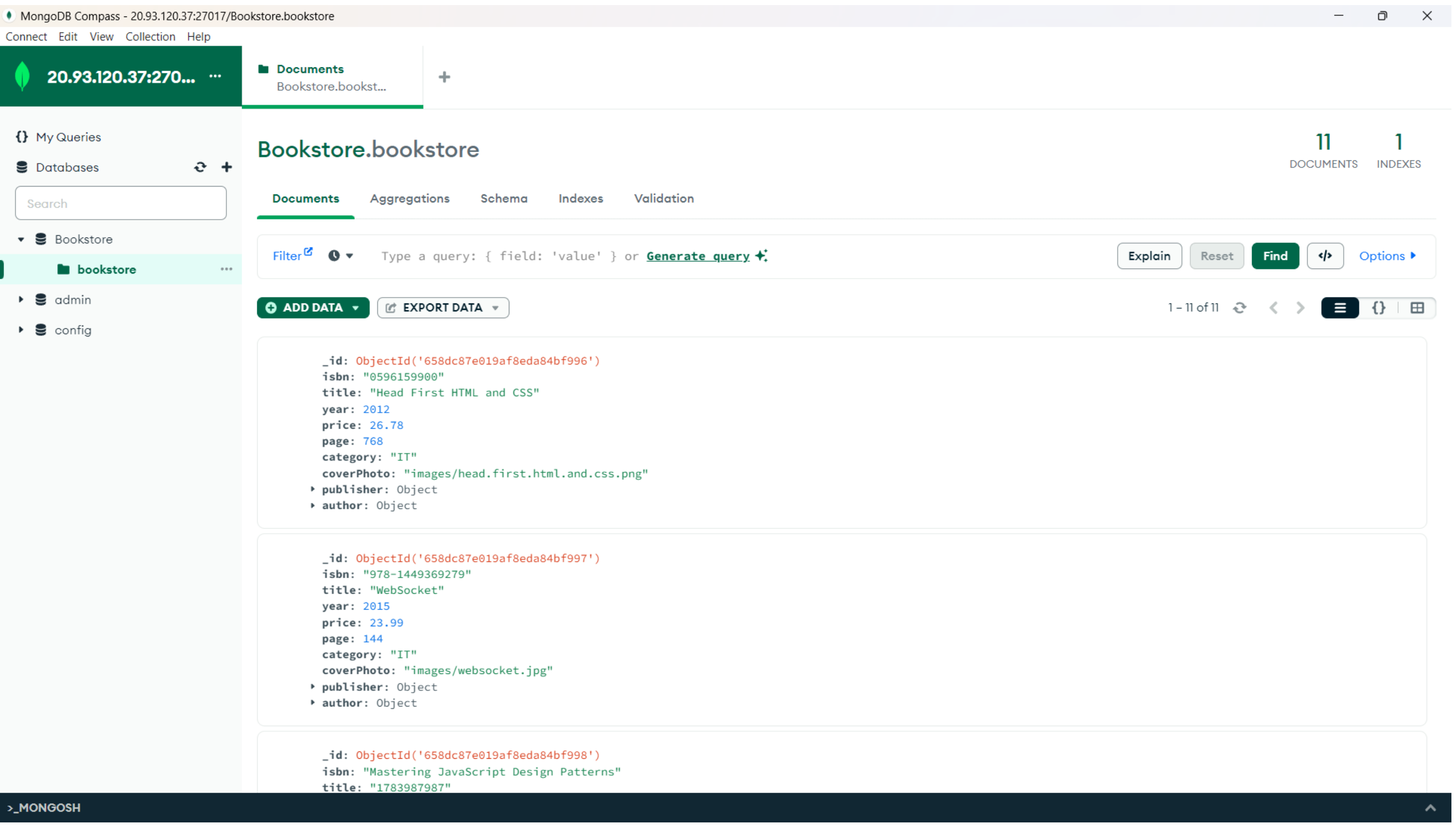


```
briksam_kasimoglu [ ~ ]$ kubectl get pods
NAME                          READY   STATUS    RESTARTS   AGE
mongodb-app-d496fbf7d-2qdnz   1/1     Running   0          2d8h
briksam_kasimoglu [ ~ ]$ kubectl get svc mongodb-service --watch
NAME              TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)           AGE
mongodb-service   LoadBalancer   10.0.121.136  20.93.120.37   27017:30211/TCP   2d8h
```

Now to access the external IP opening a browser and including it preceded by mongodb and followed by the port in the yaml files as the following opens the mongodb instance on mongo compass:

`mongodb://20.93.120.37:27017`

The instance once opened from a browser:


MongoDB Compass - 20.93.120.37:27017/Bookstore.bookstore

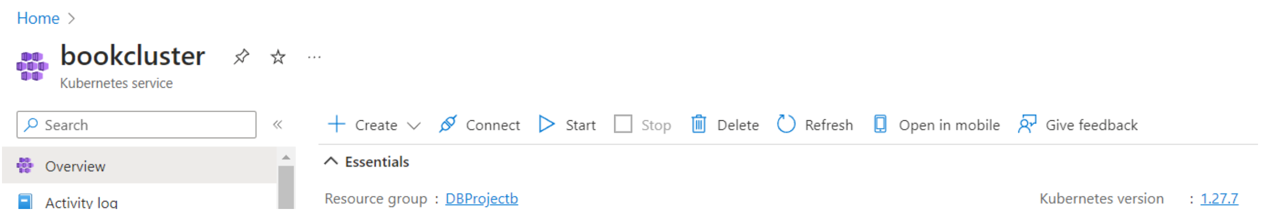
Bookstore.bookstore — 11 DOCUMENTS, 1 INDEXES

```
_id: ObjectId('658dc87e019af8eda84bf996')
isbn: "0596159900"
title: "Head First HTML and CSS"
year: 2012
price: 26.78
page: 768
category: "IT"
coverPhoto: "images/head.first.html.and.css.png"
publisher: Object
author: Object

_id: ObjectId('658dc87e019af8eda84bf997')
isbn: "978-1449369279"
title: "WebSocket"
year: 2015
price: 23.99
page: 144
category: "IT"
coverPhoto: "images/websocket.jpg"
publisher: Object
author: Object

_id: ObjectId('658dc87e019af8eda84bf998')
isbn: "Mastering JavaScript Design Patterns"
title: "1783987987"
```

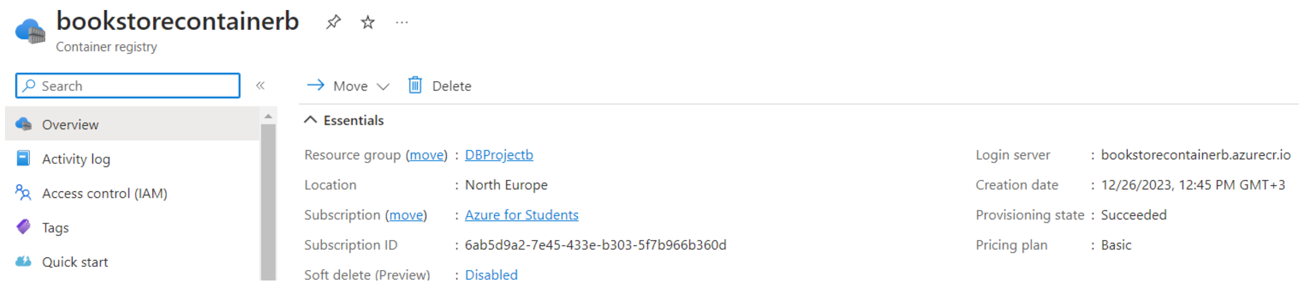
Remember to restart the Kubernetes service when it is needed for use again:


bookcluster
Kubernetes service

Resource group : DBProjectb
Kubernetes version : 1.27.7

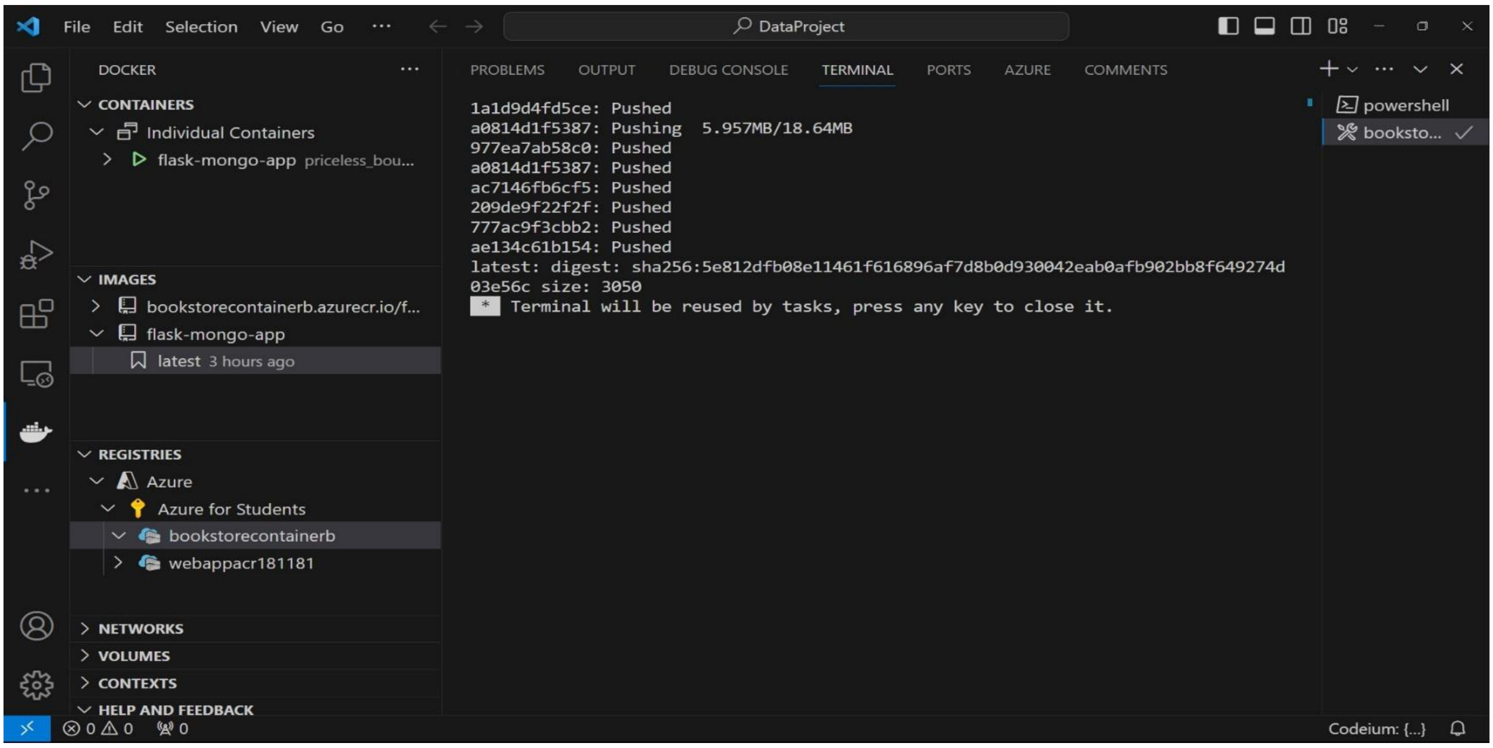
Task-4: Python Application Deployment:

For this we first need to create an ACR azure container registry following similar steps of previous creations from create resource to finding “azure container registry” to filling in information and changing the pricing plan to basic.

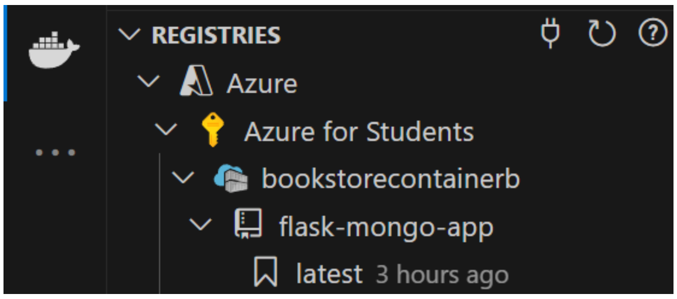

bookstorecontainerb
Container registry

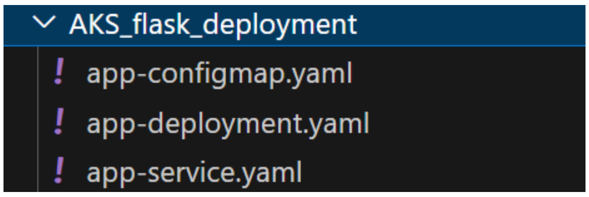
Resource group (move)	DBProjectb	Login server	bookstorecontainerb.azurecr.io
Location	North Europe	Creation date	12/26/2023, 12:45 PM GMT+3
Subscription (move)	Azure for Students	Provisioning state	Succeeded
Subscription ID	6ab5d9a2-7e45-433e-b303-5f7b966b360d	Pricing plan	Basic
Soft delete (Preview)	Disabled		

After which we push the image we created of the flask app on docker to this registry through the help of vs-code and its docker extension, opening the docker extension and signing in the registries part to azure student account and then right clicking on the image in the docker folders and pressing push and following the vs-code's user interface to push it to the wanted azure registry and waiting for it to finish gives the following results:



```
1a1d9d4fd5ce: Pushed
a0814d1f5387: Pushing  5.957MB/18.64MB
977ea7ab58c0: Pushed
a0814d1f5387: Pushed
ac7146fb6cf5: Pushed
209de9f22f2f: Pushed
777ac9f3cbb2: Pushed
ae134c61b154: Pushed
latest: digest: sha256:5e812dfb08e11461f616896af7d8b0d930042eab0afb902bb8f649274d
03e56c size: 3050
 *  Terminal will be reused by tasks, press any key to close it.
```





The image is used in the yaml files for the deployment on Azure Kubernetes.
The yaml files are uploaded in the submission folder under the folder AKS_flask_deployment as shown above.
3 replicas were used in the app-deployment.yaml file for scalability.
The config map part was tried and deployed however it was causing timeout problems when the service was exposed to the internet but removing it from the deployment worked and the configuration variables were added in the flask application explicitly. The way of deploying the configmap is as follows in the code this part will be added:

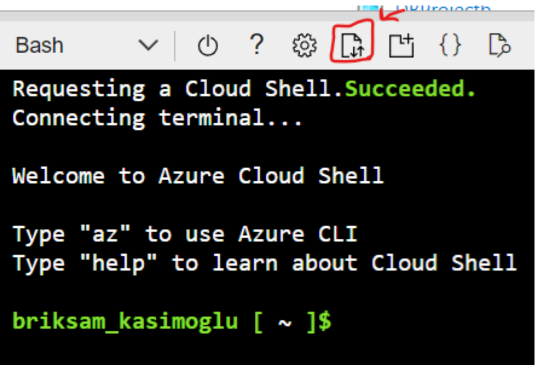
```
'''
import os

app = Flask(__name__)

if 'MONGO_URI' in os.environ and 'APP_ENV' in os.environ:
    mongo_uri = os.environ.get('MONGO_URI')
    app_env = os.environ.get('APP_ENV')
else:
    mongo_uri = "mongodb://briksam:vH8l1cjIoOH2CM6w80qXJVj9z6qrFFC2pgAx
    app_env = "python"
'''
```

the if else is for being able to run this both locally on docker and on the Kubernetes service.
And of course, the rebuilding of the image and pushing the new one to azure container registry before deploying it to Kubernetes is needed.
And then deploying the configmap yaml file is done

through azure cloud shell where the file is first uploaded with the rest of the yaml files through pressing the following :



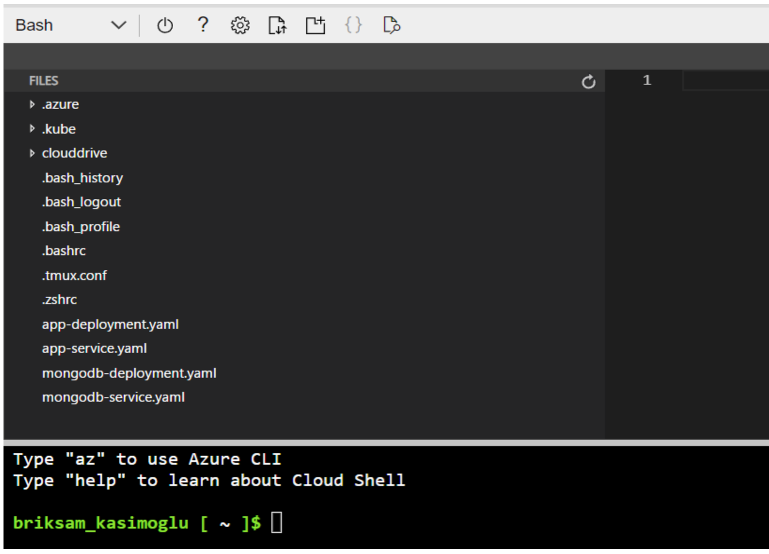
```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

briksam_kasimoglu [ ~ ]$
```

And the button with the curly brackets opens a terminal that shows all uploaded files and more information about this session and previous cloud sessions:



```
Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

briksam_kasimoglu [ ~ ]$
```

The cluster is started again from the azure website, and then this command is run on the terminal for connecting the container registry to the cluster:

```
az aks update --name bookcluster --resource-group DBProjectb --attach-acr bookstorecontainerb
```

after which the following command is run to apply the configmap :
kubectl apply -f app-configmap.yaml

and then the following steps are the same if there was no configmap applied except that the app-deployment will need a modification to reference the configmap yaml file as such:

```
envFrom:
    - configMapRef:
        name: bookstore-config
```

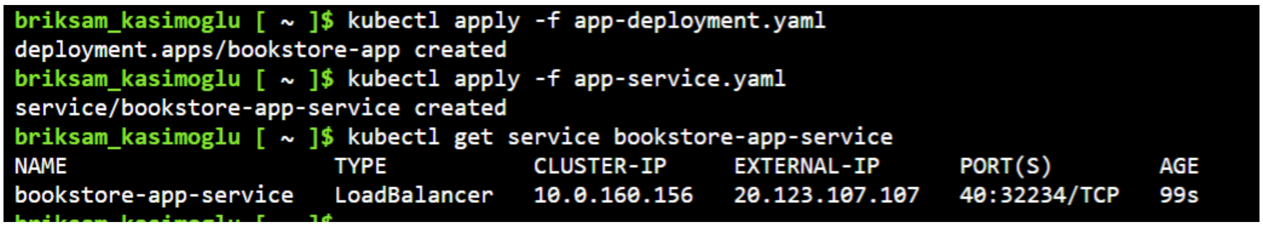
Now the steps without the configmap is the same starting the cluster, uploading the yaml files the same way mentioned before, running the command to connect to the container registry

```
az aks update --name bookcluster --resource-group DBProjectb --attach-acr bookstorecontainerb
```

and then running the following the following commands sequentially for the deployment of the app and the creation of the service
kubectl apply -f app-deployment.yaml

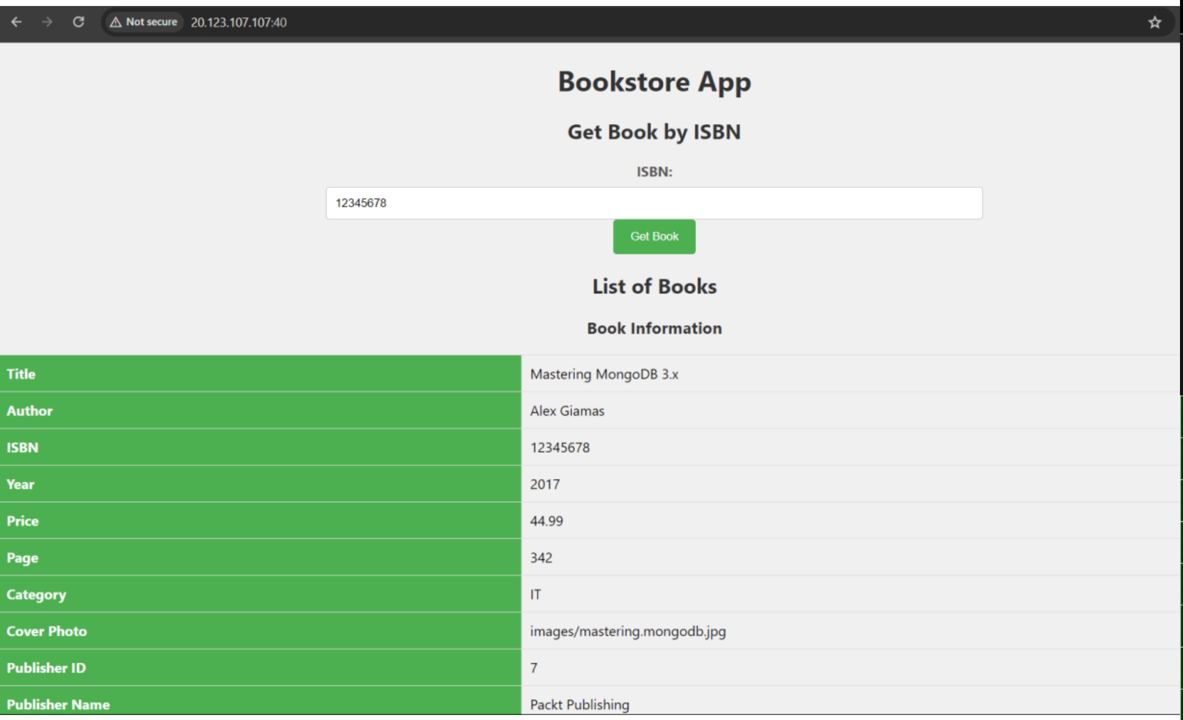
kubectl apply -f app-service.yaml

kubectl get service bookstore-app-service
As the following:



```
briksam_kasimoglu [ ~ ]$ kubectl apply -f app-deployment.yaml
deployment.apps/bookstore-app created
briksam_kasimoglu [ ~ ]$ kubectl apply -f app-service.yaml
service/bookstore-app-service created
briksam_kasimoglu [ ~ ]$ kubectl get service bookstore-app-service
NAME                    TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)         AGE
bookstore-app-service   LoadBalancer   10.0.160.156   20.123.107.107   40:32234/TCP    99s
```

The last command returns the **external IP** that the flask app can be accessed from opening the following link from the browser is enough and it will take us immediately to the user interface built for the get operation:
http://20.123.107.107:40



Bookstore App

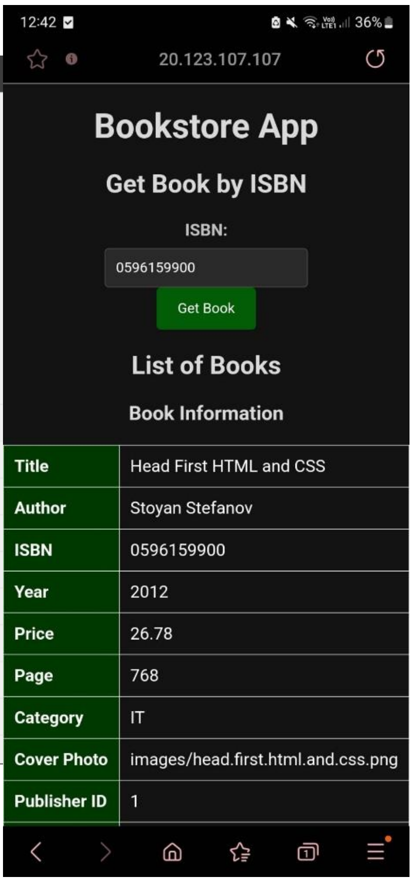
Get Book by ISBN

ISBN: 12345678

List of Books

Book Information

Title	Mastering MongoDB 3.x
Author	Alex Giamas
ISBN	12345678
Year	2017
Price	44.99
Page	342
Category	IT
Cover Photo	images/mastering.mongodb.jpg
Publisher ID	7
Publisher Name	Packt Publishing



Bookstore App

Get Book by ISBN

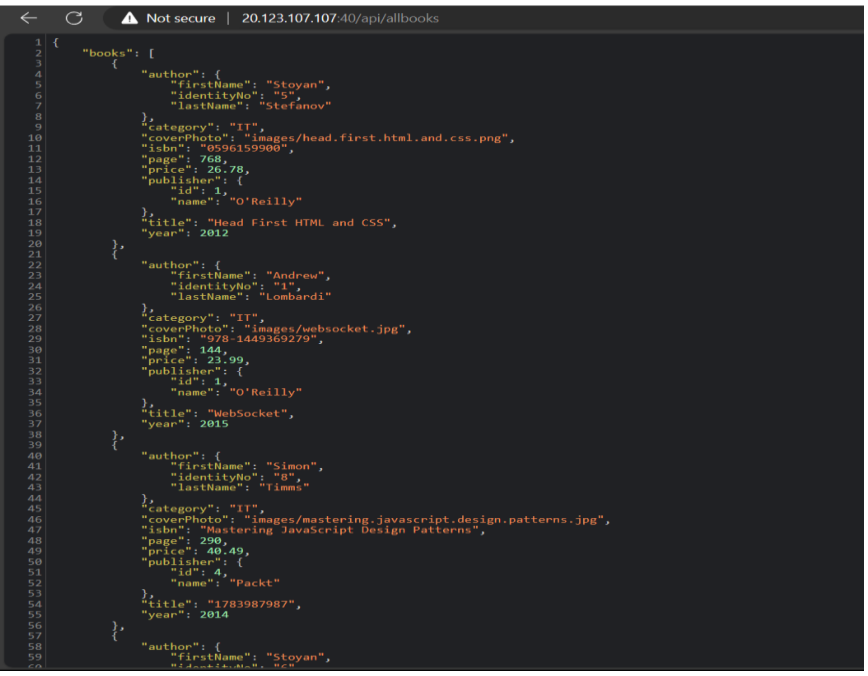
ISBN: 0596159900

List of Books

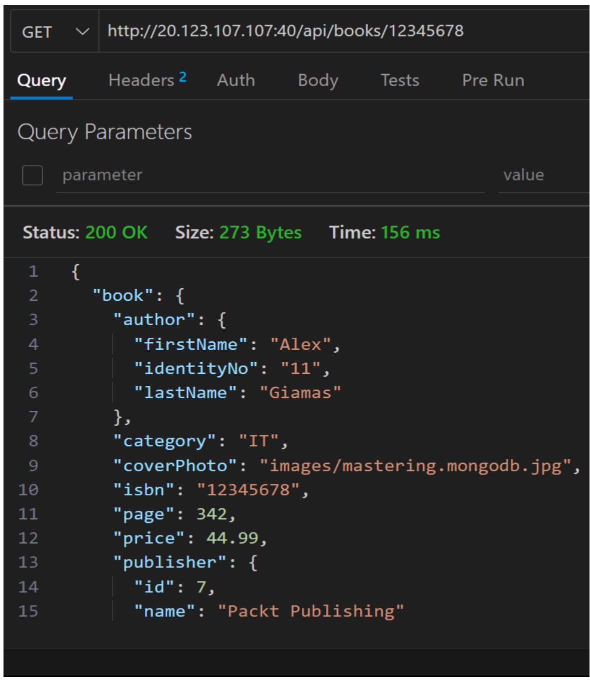
Book Information

Title	Head First HTML and CSS
Author	Stoyan Stefanov
ISBN	0596159900
Year	2012
Price	26.78
Page	768
Category	IT
Cover Photo	images/head.first.html.and.css.png
Publisher ID	1

As seen above the webpage can be accessed even from a non-local device (a mobile phone)
And of course, to access other operations the same things that applied to the local link running on docker apply here so for example to get all books the same routing written in the app.py is applied:



```
{
    "books": [
        {
            "author": {
                "firstName": "Stoyan",
                "identityNo": "5",
                "lastName": "Stefanov"
            },
            "category": "IT",
            "coverPhoto": "images/head.first.html.and.css.png",
            "isbn": "0596159900",
            "page": 768,
            "price": 26.78,
            "publisher": {
                "id": 1,
                "name": "O'Reilly"
            },
            "title": "Head First HTML and CSS",
            "year": 2012
        },
        {
            "author": {
                "firstName": "Andrew",
                "identityNo": "1",
                "lastName": "Lombardi"
            },
            "category": "IT",
            "coverPhoto": "images/websocket.jpg",
            "isbn": "978-1449369279",
            "page": 144,
            "price": 23.99,
            "publisher": {
                "id": 1,
                "name": "O'Reilly"
            },
            "title": "WebSocket",
            "year": 2015
        },
        {
            "author": {
                "firstName": "Simon",
                "identityNo": "8",
                "lastName": "Timms"
            },
            "category": "IT",
            "coverPhoto": "images/mastering.javascript.design.patterns.jpg",
            "isbn": "Mastering JavaScript Design Patterns",
            "page": 290,
            "price": 40.49,
            "publisher": {
                "id": 4,
                "name": "Packt"
            },
            "title": "1783987987",
            "year": 2014
        },
        {
            "author": {
                "firstName": "Stoyan",
```



Status: 200 OK   Size: 273 Bytes   Time: 156 ms

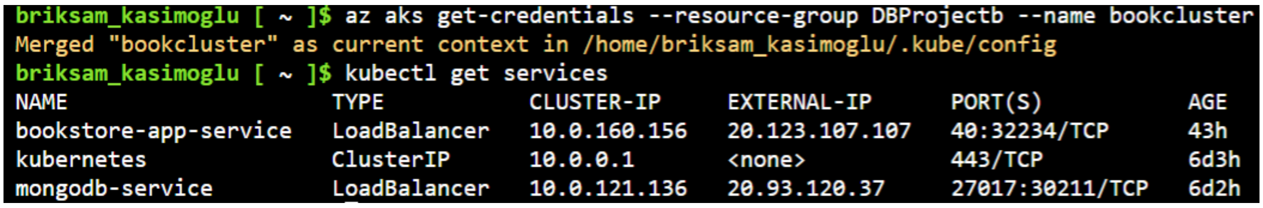
```
{
  "book": {
    "author": {
      "firstName": "Alex",
      "identityNo": "11",
      "lastName": "Giamas"
    },
    "category": "IT",
    "coverPhoto": "images/mastering.mongodb.jpg",
    "isbn": "12345678",
    "page": 342,
    "price": 44.99,
    "publisher": {
      "id": 7,
      "name": "Packt Publishing"
```

Or the thunder Client can be used for more convenient access the same way as explained before as seen above.
The operation can be chosen from the drop-down arrow and the routing is specified in the link area and if any input is needed it's added just like the examples before.
The cluster is stopped again to avoid excessive costs but once restarted the link will automatically work again.

Task-5: Service Discovery and Networking:

Opening the azure cloud shell:

First check existing services:


```
briksam_kasimoglu [ ~ ]$ az aks get-credentials --resource-group DBProjectb --name bookcluster
Merged "bookcluster" as current context in /home/briksam_kasimoglu/.kube/config
briksam_kasimoglu [ ~ ]$ kubectl get services
NAME                   TYPE           CLUSTER-IP     EXTERNAL-IP      PORT(S)           AGE
bookstore-app-service  LoadBalancer   10.0.160.156   20.123.107.107   40:32234/TCP      43h
kubernetes             ClusterIP      10.0.0.1       <none>           443/TCP           6d3h
mongodb-service        LoadBalancer   10.0.121.136   20.93.120.37     27017:30211/TCP   6d2h
```

It looks like both services are running on the same virtual network and can discover each other by their namespace, Now its time to set up the network policy for them with the use of a yaml file again.

If the services were operating on different virtual networks either the following should be applied or the recreation of the whole cluster is required to specify the virtual net configuration:

Getting the virtual network and subnet details from the current deployment of the AKS cluster :

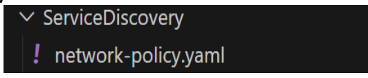
AKS_RESOURCE_GROUP=$(az aks show --resource-group DBProjectb --name bookcluster --query nodeResourceGroup -o tsv)

VNET_NAME=$(az network vnet list --resource-group $AKS_RESOURCE_GROUP --query '[0].name' -o tsv)
SUBNET_NAME=$(az network vnet subnet list --resource-group $AKS_RESOURCE_GROUP --vnet-name $VNET_NAME --query '[0].name' -o tsv)
SUBNET_ID=$(az network vnet subnet show --resource-group $AKS_RESOURCE_GROUP --vnet-name $VNET_NAME --name $SUBNET_NAME --query id -o tsv)

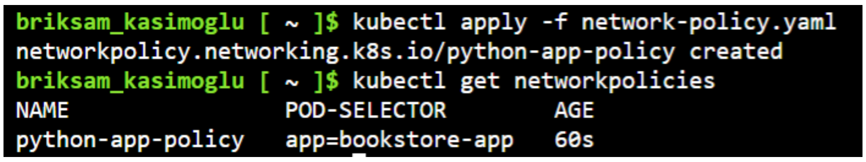
And then updating the cluster to use the existing virtual net and subnet:

az aks update --resource-group DBProjectb --name bookcluster --vnet-subnet-id $SUBNET_ID

The yaml file once again is uploaded in the submission folder in the folder called ServiceDiscovery:

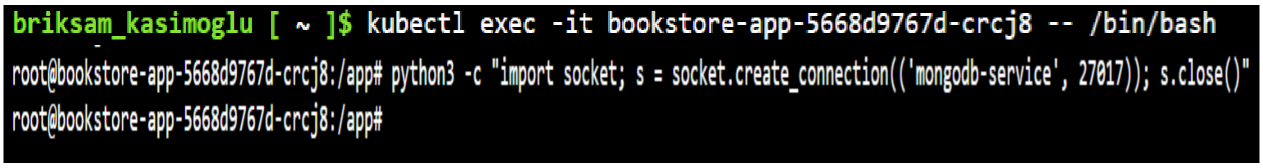


Making sure that the cluster is running and if not starting it again we open the cloud shell and connect to the cluster as shown previously and getting the services making sure they're running we upload the network policy yaml file the same way mentioned in the previous tasks and then applying it with the following command:
kubectl apply -f network-policy.yaml
and then running the following command to check that the policy is applied:
kubectl get networkpolicies
As the following:


```
briksam_kasimoglu [ ~ ]$ kubectl apply -f network-policy.yaml
networkpolicy.networking.k8s.io/python-app-policy created
briksam_kasimoglu [ ~ ]$ kubectl get networkpolicies
NAME                POD-SELECTOR          AGE
python-app-policy   app=bookstore-app     60s
```

Tying to access mongodb pod from inside the flask application pod to check the service discovery between them:

kubectl exec -it bookstore-app-5668d9767d-crcj8 -- /bin/bash
and after:
python3 -c "import socket; s = socket.create_connection(('mongodb-service', 27017)); s.close()"

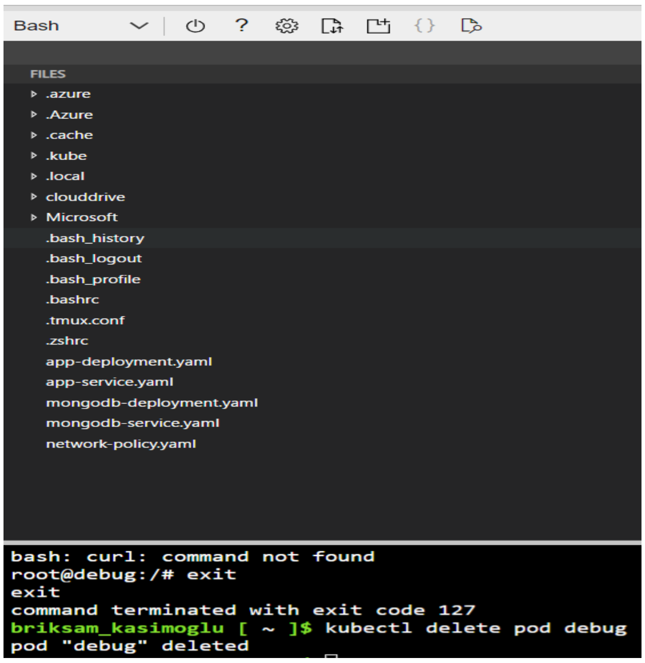


```
briksam_kasimoglu [ ~ ]$ kubectl exec -it bookstore-app-5668d9767d-crcj8 -- /bin/bash
root@bookstore-app-5668d9767d-crcj8:/app# python3 -c "import socket; s = socket.create_connection(('mongodb-service', 27017)); s.close()"
root@bookstore-app-5668d9767d-crcj8:/app#
```

The python script was able to successfully run and create a connection socket which means that the service discovery between the two apps is up and running.

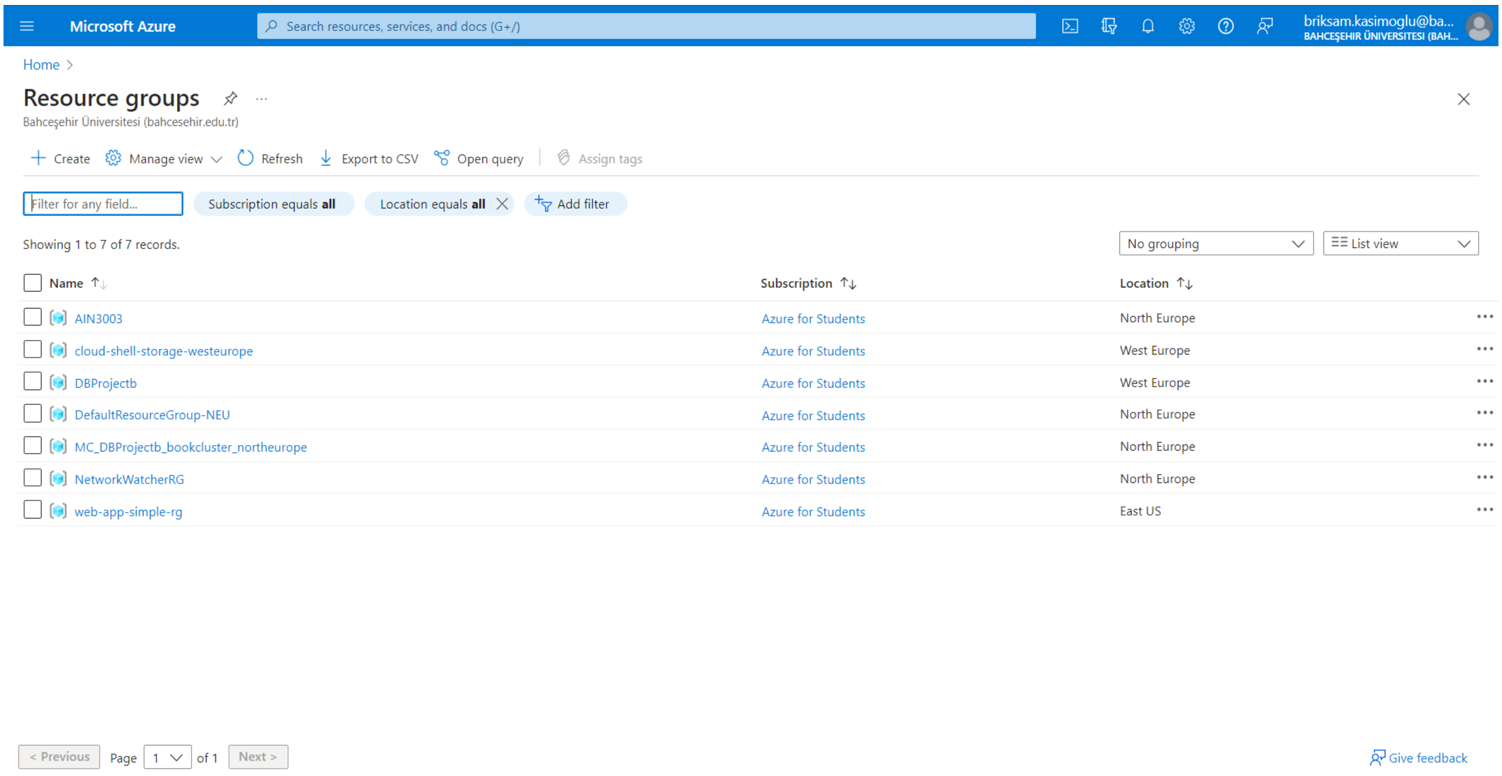
A temporary debugging pod was created and tried accessing the flask app and the mongodb app from within it and the access was denied which means the connection is secure and the network policy is working properly.

Extra documentation:



```
.azure
.Azure
.cache
.kube
.local
clouddrive
Microsoft
.bash_history
.bash_logout
.bash_profile
.bashrc
.tmux.conf
.zshrc
app-deployment.yaml
app-service.yaml
mongodb-deployment.yaml
mongodb-service.yaml
network-policy.yaml
```

```
bash: curl: command not found
root@debug:/# exit
exit
command terminated with exit code 127
briksam_kasimoglu [ ~ ]$ kubectl delete pod debug
pod "debug" deleted
```

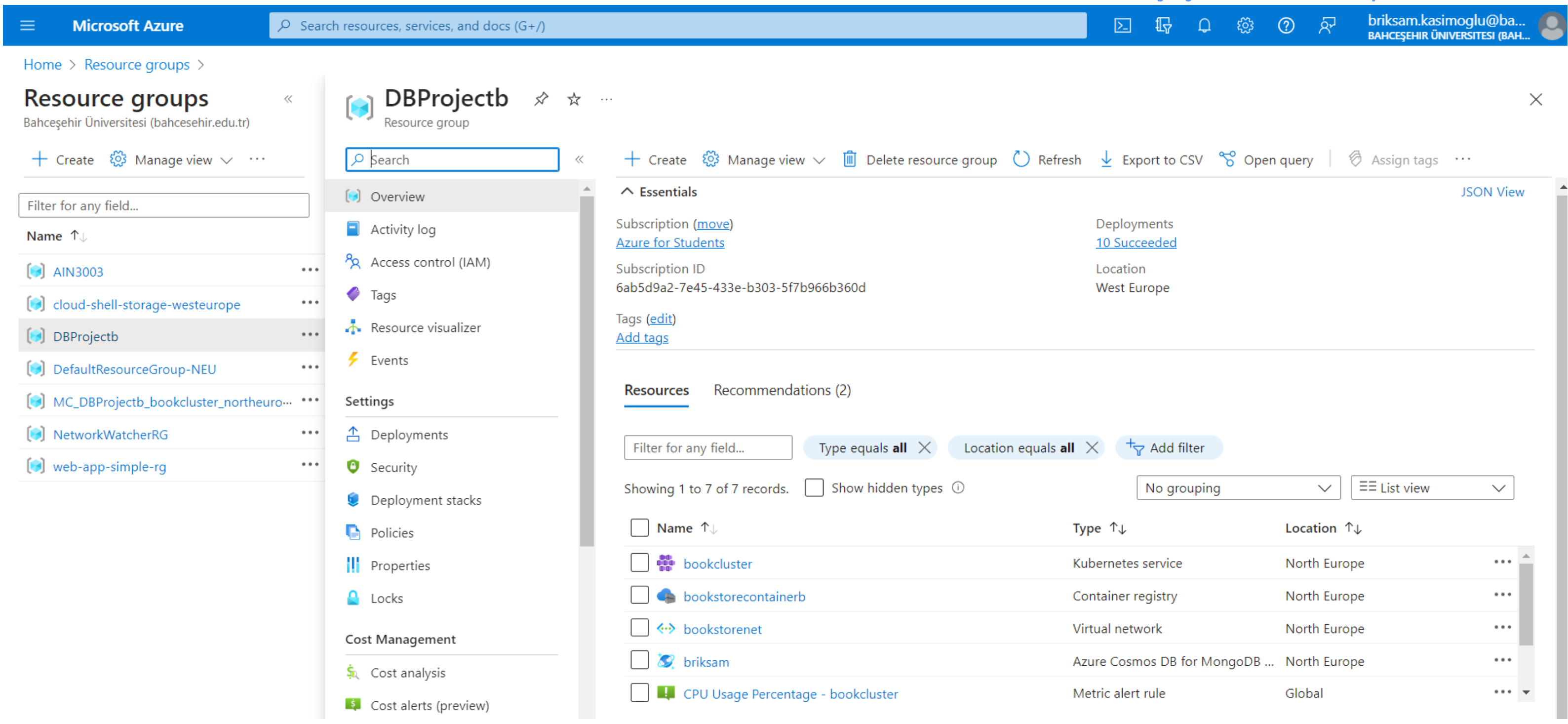


Resource groups
Bahçeşehir Üniversitesi (bahcesehir.edu.tr)

Showing 1 to 7 of 7 records.

Name	Subscription	Location
AIN3003	Azure for Students	North Europe
cloud-shell-storage-westeurope	Azure for Students	West Europe
DBProjectb	Azure for Students	West Europe
DefaultResourceGroup-NEU	Azure for Students	North Europe
MC_DBProjectb_bookcluster_northeurope	Azure for Students	North Europe
NetworkWatcherRG	Azure for Students	North Europe
web-app-simple-rg	Azure for Students	East US

Briksam Kasımoğlu-2102969


Home > Resource groups >

Resource groups
Bahçeşehir Üniversitesi (bahcesehir.edu.tr)

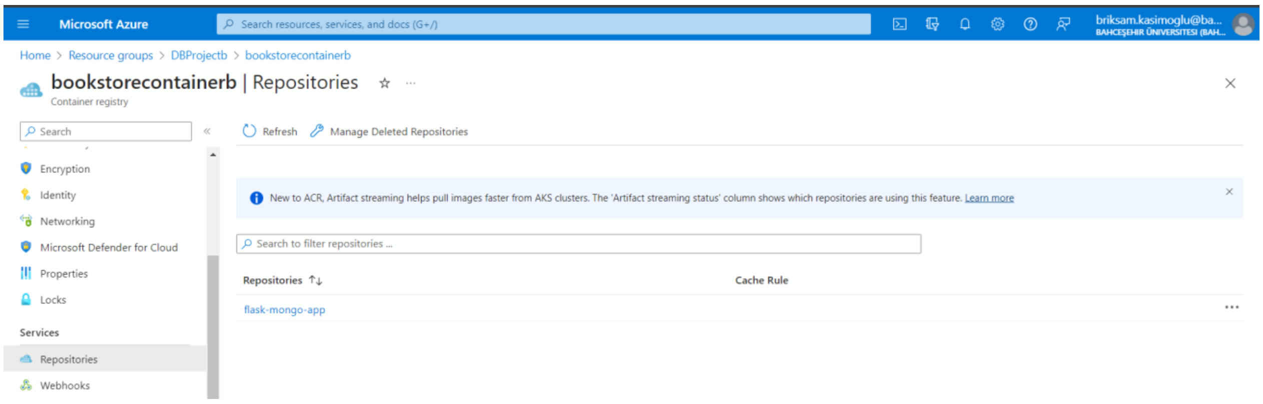
Name
AIN3003
cloud-shell-storage-westeurope
DBProjectb
DefaultResourceGroup-NEU
MC_DBProjectb_bookcluster_northeuro...
NetworkWatcherRG
web-app-simple-rg

DBProjectb
Resource group

Essentials
Subscription (move): Azure for Students
Subscription ID: 6ab5d9a2-7e45-433e-b303-5f7b966b360d
Tags (edit): Add tags
Deployments: 10 Succeeded
Location: West Europe

Resources | Recommendations (2)
Type equals all | Location equals all
Showing 1 to 7 of 7 records.

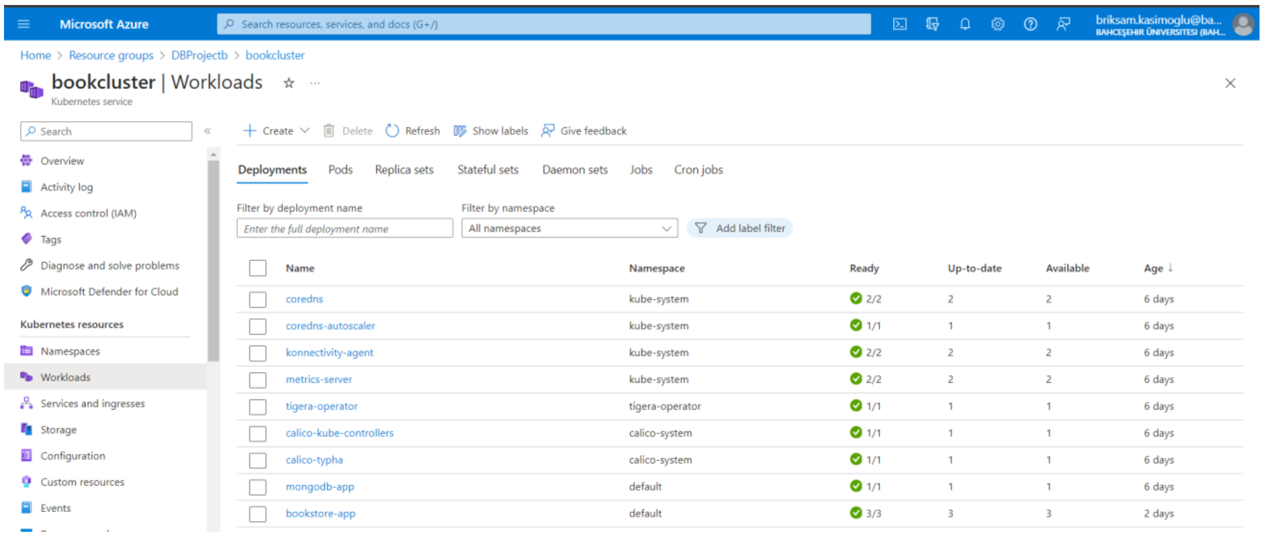
Name	Type	Location
bookcluster	Kubernetes service	North Europe
bookstorecontainerb	Container registry	North Europe
bookstorenet	Virtual network	North Europe
briksam	Azure Cosmos DB for MongoDB ...	North Europe
CPU Usage Percentage - bookcluster	Metric alert rule	Global


Home > Resource groups > DBProjectb > bookstorecontainerb

bookstorecontainerb | Repositories
Container registry

New to ACR, Artifact streaming helps pull images faster from AKS clusters. The 'Artifact streaming status' column shows which repositories are using this feature. Learn more

Repositories	Cache Rule
flask-mongo-app	


Home > Resource groups > DBProjectb > bookcluster

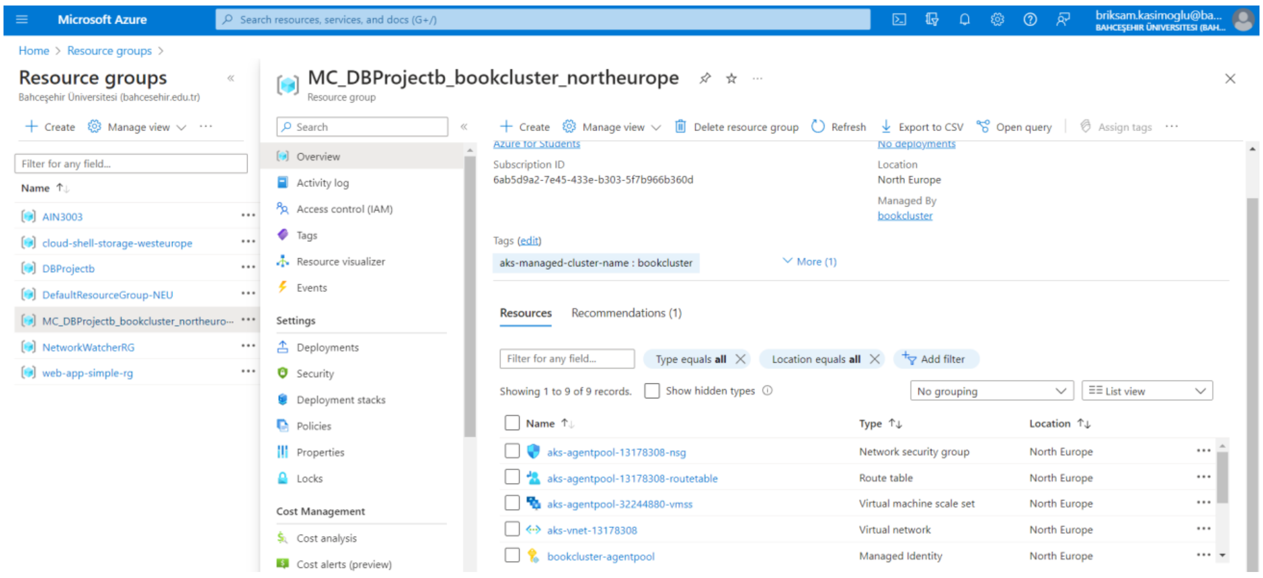
bookcluster | Workloads
Kubernetes service

Deployments | Pods | Replica sets | Stateful sets | Daemon sets | Jobs | Cron jobs

Filter by deployment name: Enter the full deployment name
Filter by namespace: All namespaces

Name	Namespace	Ready	Up-to-date	Available	Age
coredns	kube-system	✓ 2/2	2	2	6 days
coredns-autoscaler	kube-system	✓ 1/1	1	1	6 days
konnectivity-agent	kube-system	✓ 2/2	2	2	6 days
metrics-server	kube-system	✓ 2/2	2	2	6 days
tigera-operator	tigera-operator	✓ 1/1	1	1	6 days
calico-kube-controllers	calico-system	✓ 1/1	1	1	6 days
calico-typha	calico-system	✓ 1/1	1	1	6 days
mongodb-app	default	✓ 1/1	1	1	6 days
bookstore-app	default	✓ 3/3	3	3	2 days

Briksam Kasımoğlu-2102969



Home > Resource groups >

Resource groups
Bahcesehir Üniversitesi (bahcesehir.edu.tr)

Name ↑↓
AIN3003
cloud-shell-storage-westeurope
DBProjectb
DefaultResourceGroup-NEU
MC_DBProjectb_bookcluster_northeuro...
NetworkWatcherRG
web-app-simple-rg

MC_DBProjectb_bookcluster_northeurope
Resource group

Subscription ID
6ab5d9a2-7e45-433e-b303-5f7b966b360d

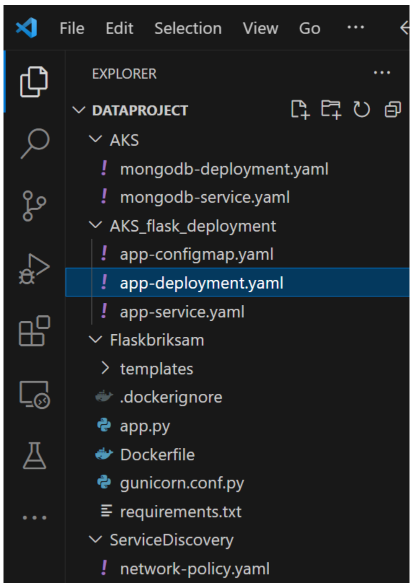
Location
North Europe

Managed By
bookcluster

Tags (edit)
aks-managed-cluster-name : bookcluster

Showing 1 to 9 of 9 records.

Name ↑↓	Type ↑↓	Location ↑↓
aks-agentpool-13178308-nsg	Network security group	North Europe
aks-agentpool-13178308-routetable	Route table	North Europe
aks-agentpool-32244880-vmss	Virtual machine scale set	North Europe
aks-vnet-13178308	Virtual network	North Europe
bookcluster-agentpool	Managed Identity	North Europe



```
EXPLORER
DATAPROJECT
  AKS
    mongodb-deployment.yaml
    mongodb-service.yaml
  AKS_flask_deployment
    app-configmap.yaml
    app-deployment.yaml
    app-service.yaml
  Flaskbriksam
    templates
    .dockerignore
    app.py
    Dockerfile
    gunicorn.conf.py
    requirements.txt
  ServiceDiscovery
    network-policy.yaml
```

With this all tasks are completed and documented and the full python code with the yaml files are uploaded as a zip file in the same architecture shown in the picture above.