

# Dropit Shopping

## Home assignment

### 1. Prerequisite

- Create Atlas account
- Get the connection string of the atlas account

Connect to Cluster1

✓ Setup connection security   ✓ Choose a connection method   Connect

1 Select your driver and version

| DRIVER  | VERSION      |
|---------|--------------|
| Node.js | 4.1 or later |

2 Add your connection string into your application code

Include full driver code example

```
mongodb+srv://hazak:<password>@cluster1.7yjzumh.mongodb.net/?retryWrites=true&w=majority
```

Replace **<password>** with the password for the **hazak** user. Ensure any option params are **URL encoded**.

Having trouble connecting? [View our troubleshooting documentation](#)

Go Back   Close

- Use Mongodb Compass for downloading the database locally
  - Mongodb Compass connection

## New Connection

Connect to a MongoDB deployment



FAVORITE

URI i

Edit Connection String

mongodb+srv://hazak:kcyillfq@cluster1.7yzumh.mongodb.net/?retryWrites=true&w=majority

### ▼ Advanced Connection Options

[General](#) [Authentication](#) [TLS/SSL](#) [Proxy/SSH](#) [In-Use Encryption](#) [Advanced](#)

#### Authentication Method

None  Username/Password  X.509  Kerberos  LDAP  AWS IAM

#### Username

hazak

#### Password

.....

#### Authentication Database i

*Optional*

#### Authentication Mechanism

- Exporting the sample\_airbnb locally

```

_id: "10038496"
listing_url: "https://www.airbnb.com/rooms/10038496"
name: "Copacabana Apartment Posto 6"
summary: "The Apartment has a living room, toilet, bedroom (suite) and American ..."
space: "The apartment has a living room, wash room, suite and an American kitc..."
description: "The Apartment has a living room, toilet, bedroom (suite) and American ..."
neighborhood_overview: "Copacabana in the South zone is the district that offers the greatest ..."
notes: ""
transit: "On the street there is plenty of transport and the subway station is n..."
access: "todo o espaço."
interaction: "Contact telephone numbers if needed: Valeria ((PHONE NUMBER HIDDEN)) (U.."
house_rules: "Entreguem o imóvel conforme receberam e respeitem as regras do condomi.."
property_type: "Apartment"
room_type: "Entire home/apt"
bed_type: "Real Bed"
minimum_nights: "3"
maximum_nights: "75"
cancellation_policy: "strict_14_with_grace_period"
last_scraped: 2019-02-11T05:00:00.000+00:00
calendar_last_scraped: 2019-02-11T05:00:00.000+00:00
first_review: 2016-01-18T05:00:00.000+00:00
last_review: 2019-01-28T05:00:00.000+00:00
accommodates: 4
bedrooms: 1
beds: 3
  
```

## 2. Project Tree

- **deploymentAndService/**
  - Mongodb-depl.yaml
  - Mongo-express-depl.yaml
- **secret/**
  - mong-secret.yaml
- **jobs/**
  - mongo-dump.yaml
  - mongo-import.yaml
- Mongoldb-express-rest-api-example
- listingsAndReviews.csv

“**deploymentAndService**” folder will create a deployment and a service for MongoDB and Express application

“**secret**” folder will store confidential information like username and password

“**jobs**” folder contains jobs for importing and dumping the Database. The particularity of a job is too create a specific pod, execute a task and delete the pod.

“**Mongoldb-express-rest-api-example**” folder will contain our express application. We will use it for building an image and using this image in our Kubernetes deployment

“**listingsAndReviews.csv**” file represent our local database in a CSV format

### 3. Node initialization

This project is running on a Minikube environment.  
So first we are going to create our Node with minikube

```
juliencohen@juliens-MacBook-Air dropit % minikube start
🕒 minikube v1.26.1 on Darwin 12.5 (arm64)
💡 Using the docker driver based on existing profile
👍 Starting control plane node minikube in cluster minikube
🌐 Pulling base image ...
🏃 Updating the running docker "minikube" container ...
🐳 Preparing Kubernetes v1.24.3 on Docker 20.10.17 ...
🔎 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
```

Then we check that our minikube node has been created successfully

```
[juliencohen@juliens-MacBook-Air dropit % kubectl get node
NAME      STATUS    ROLES      AGE      VERSION
minikube  Ready     <none>    100s    v1.24.3
```

## 4. Import the provided db dump into the local DB

### A. Deployment and Service of the MongoDB application

- mong-secret.yaml

```
apiVersion: v1
kind: Secret
metadata:
  name: mongodb-secret
type: Opaque
data:
  mongo-root-username: dXNlcj5hbWU=
  mongo-root-password: cGFzc3dvcmQ=
```

Here we added mongo username and password in a base64 encoded format

mongo-root-username: username

mongo-root-password: password

- Apply the secret

```
julienco@juliens-MacBook-Air dropit % kubectl apply -f secret/mong-secret.yaml
secret/mongodb-secret created
julienco@juliens-MacBook-Air dropit % kubectl get secret
NAME          TYPE      DATA   AGE
mongodb-secret Opaque    2      6s
```

- mongodb-depl.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: mongodb-deployment
  labels:
    app: mongodb
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb
  template:
    metadata:
      labels:
        app: mongodb
    spec:
      containers:
        - name: mongodb
          image: mongo
          ports:
            - containerPort: 27017
          env:
            - name: MONGO_INITDB_ROOT_USERNAME
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: mongo-root-username
            - name: MONGO_INITDB_ROOT_PASSWORD
              valueFrom:
                secretKeyRef:
                  name: mongodb-secret
                  key: mongo-root-password
```

Here we can see that we will create a deployment of a mongodb image and we will add username and password env variable in our pod.

- Mongodb service

```
apiVersion: v1
kind: Service
metadata:
  name: mongodb-service
spec:
  selector:
    app: mongodb
  type: LoadBalancer
  ports:
    - protocol: TCP
      port: 27017
      targetPort: 27017
      nodePort: 30001
~
```

Here the service will create an external IP that is useful for the communication between pods.

- Lets apply these configurations

## Information about our mongo deployment

```
juliencohen@juliens-MacBook-Air dropit % kubectl get deploy
NAME           READY   UP-TO-DATE   AVAILABLE   AGE
mongodb-deployment  1/1     1          1          6s
juliencohen@juliens-MacBook-Air dropit % kubectl describe deploy mongodb-deployment
Name:           mongodb-deployment
Namespace:      default
CreationTimestamp:  Wed, 17 Aug 2022 12:32:12 +0300
Labels:          app=mongodb
Annotations:    deployment.kubernetes.io/revision: 1
Selector:        app=mongodb
Replicas:       1 desired | 1 updated | 1 total | 1 available | 0 unavailable
StrategyType:   RollingUpdate
MinReadySeconds: 0
RollingUpdateStrategy: 25% max unavailable, 25% max surge
Pod Template:
  Labels:  app=mongodb
  Containers:
    mongodb:
      Image:      mongo
      Port:       27017/TCP
      Host Port:  0/TCP
      Environment:
        MONGO_INITDB_ROOT_USERNAME: <set to the key 'mongo-root-username' in secret 'mongodb-secret'> Optional: false
        MONGO_INITDB_ROOT_PASSWORD: <set to the key 'mongo-root-password' in secret 'mongodb-secret'> Optional: false
      Mounts:
        <none>
      Volumes:
        <none>
  Conditions:
    Type        Status  Reason
    ----        ----   -----
    Available   True    MinimumReplicasAvailable
    Progressing True    NewReplicaSetAvailable
  OldReplicaSets: <none>
  NewReplicaSet:  mongodb-deployment-67dcfb9c9f (1/1 replicas created)
Events:
  Type      Reason          Age      From            Message
  ----      ----          ----      ----            -----
  Normal   ScalingReplicaSet 21s     deployment-controller  Scaled up replica set mongodb-deployment-67dcfb9c9f to 1
```

Here is a good practice to check our deployment with a “kubectl describe” command line  
We can see information about the image, the replicaset and the environment variables that we created.

### Information about our mongo service

```
juliencohen@juliens-MacBook-Air dropit % kubectl get svc
NAME          TYPE      CLUSTER-IP      EXTERNAL-IP     PORT(S)        AGE
kubernetes    ClusterIP  10.96.0.1       <none>        443/TCP       13d
mongodb-service LoadBalancer  10.108.104.10  <pending>     27017:30001/TCP  10m
juliencohen@juliens-MacBook-Air dropit % kubectl describe svc mongodb-service
error: unknown command "describe" for "kubectl"

Did you mean this?
  describe
juliencohen@juliens-MacBook-Air dropit % kubectl describe svc mongodb-service
Name:            mongodb-service
Namespace:       default
Labels:          <none>
Annotations:    <none>
Selector:        app=mongodb
Type:           LoadBalancer
IP Family Policy: SingleStack
IP Families:    IPv4
IP:              10.108.104.10
IPs:             10.108.104.10
Port:            <unset>  27017/TCP
TargetPort:      27017/TCP
NodePort:        <unset>  30001/TCP
Endpoints:      172.17.0.3:27017
Session Affinity: None
External Traffic Policy: Cluster
Events:          <none>
```

We can see here on our mongodb-service that we have a **CLUSTER-IP**  
The **EXTERNAL-IP** is set to pending because we are working on a **minikube** environment  
on a real environment an **EXTERNAL-IP** will be set.

On the “**kubectl describe**” command output we can see interesting information about the name of the service the IP address, the port, the type of the service.

## Information about the pod that has been created by our deployment

```
Start Time: Wed, 17 Aug 2022 12:32:13 +0300
Labels: app=mongodb
        pod-template-hash=67dcfb9c9f
Annotations: <none>
Status: Running
IP: 172.17.0.3
IPs:
  IP: 172.17.0.3
Controlled By: ReplicaSet/mongodb-deployment-67dcfb9c9f
Containers:
  mongodb:
    Container ID: docker://835ffe87e4b6c195fd205b561387a38655d403c764a120abc9c4690e2fe977e9
    Image: mongo
    Image ID: docker-pullable://mongo@sha256:4200c3073389d5b303070e53ff8f5e4472efb534340d28599458ccc24f378025
    Port: 27017/TCP
    Host Port: 0/TCP
    State: Running
      Started: Wed, 17 Aug 2022 12:32:17 +0300
    Ready: True
    Restart Count: 0
    Environment:
      MONGO_INITDB_ROOT_USERNAME: <set to the key 'mongo-root-username' in secret 'mongodb-secret'> Optional: false
      MONGO_INITDB_ROOT_PASSWORD: <set to the key 'mongo-root-password' in secret 'mongodb-secret'> Optional: false
    Mounts:
      /var/run/secrets/kubernetes.io/serviceaccount from kube-api-access-dfxkg (ro)
Conditions:
  Type Status
  Initialized True
  Ready True
  ContainersReady True
  PodScheduled True
Volumes:
  kube-api-access-dfxkg:
    Type: Projected (a volume that contains injected data from multiple sources)
    TokenExpirationSeconds: 3607
    ConfigMapName: kube-root-ca.crt
    ConfigMapOptional: <nil>
    DownwardAPI: true
QoS Class: BestEffort
Node-Selectors: <none>
Tolerations: node.kubernetes.io/not-ready:NoExecute op=Exists for 300s
node.kubernetes.io/unreachable:NoExecute op=Exists for 300s
Events:
  Type Reason Age From Message
  ---- ---- -- -- -----
  Normal Scheduled 15m default-scheduler Successfully assigned default/mongodb-deployment-67dcfb9c9f-5rdr to minikube
  Normal Pulling 15m kubelet   Pulling image "mongo"
  Normal Pulled 15m kubelet   Successfully pulled image "mongo" in 3.510281335s
  Normal Created 15m kubelet   Created container mongodb
  Normal Started 15m kubelet   Started container mongodb
```

The Events part is interesting here because we can see that the pod has been created, the mongo image has been pulled and the container has been started

## B. MongoDB import job

We are going to create a job that will create a pod and execute a mongoimport command line

- Mongo-import.yaml

```
apiVersion: batch/v1
kind: Job
metadata:
  name: mongodb-import
  labels:
    app: mongodb
spec:
  backoffLimit: 5
  activeDeadlineSeconds: 100
  template:
    spec:
      containers:
        - name: mongoimport
          image: mongo
          args:
            - /bin/sh
            - '-c'
            - mongoimport --host mongodb-service -u username -p password --authenticationDatabase admin --type csv -d sample_airbnb -c listingsAndReviews --headerline --drop /db/listingsAndReviews.csv
      volumes:
        - name: db-volume
          mountPath: /db
  volumes:
    - name: db-volume
      hostPath:
        path: /mnt/vpath
  restartPolicy: OnFailure
```

The interesting information here is the mongoimport command line that will import the csv file in the /db folder.

A volume will be created “/db” that will refer to the /mnt/vpath from our minikube node.

Before apply this configuration file we have to create the vpath folder and add our csv file

```
minikube cp listingsAndReviews.csv /mnt/vpath/listingAndReviews.csv
```

Now that our csv file exist in our Node on /mnt/vpath we can apply the job file

```
juliencohen@juliens-MacBook-Air dropit % kubectl logs mongodb-import-k754f
2022-08-17T10:07:33.308+0000    connected to: mongodb://mongodb-service/
2022-08-17T10:07:33.309+0000    dropping: sample_airbnb.listingsAndReviews
2022-08-17T10:07:35.442+0000    5555 document(s) imported successfully. 0 document(s) failed to import.
```

When we are checking the logs of the pod created by the job we can see that 5555 documents have been imported

Lets use mongodb compass for checking that our database has been added successfully

First we create a tunnel for accessing to our mongodb locally

```
juliencohen@juliens-MacBook-Air dropit % minikube service mongodb-service
|-----+-----+-----+-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----+-----+-----+-----|
| default | mongodb-service | 27017 | http://192.168.49.2:30001 |
|-----+-----+-----+-----|
🏃 Starting tunnel for service mongodb-service.
|-----+-----+-----+-----|
| NAMESPACE | NAME | TARGET PORT | URL |
|-----+-----+-----+-----|
| default | mongodb-service | | http://127.0.0.1:60265 |
|-----+-----+-----+-----|
⚠️ Opening service default/mongodb-service in default browser...
⚠️ Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

Then we use the url for connecting on compass

The screenshot shows the MongoDB Compass interface for connecting to a MongoDB deployment. At the top, there's a header with 'Connect to a MongoDB deployment' and a 'FAVORITE' button. Below it is a 'URI' field containing the URL: `mongodb://username:password@127.0.0.1:60265/?authMechanism=DEFAULT`. To the right of the URI is an 'Edit Connection String' button with a toggle switch.

Below the URI is a section titled 'Advanced Connection Options' with tabs for General, Authentication (which is selected), TLS/SSL, Proxy/SSH, In-Use Encryption, and Advanced.

The 'Authentication Method' section shows several options: None, Username/Password (which is highlighted with a green border), X.509, Kerberos, LDAP, and AWS IAM.

The 'Username' field contains 'username' and the 'Password' field contains '.....' (redacted).

The 'Authentication Database' field is labeled 'Optional'.

The 'Authentication Mechanism' section shows three options: Default (which is highlighted with a green border), SCRAM-SHA-1, and SCRAM-SHA-256.

At the bottom are three buttons: 'Save' (gray), 'Save & Connect' (green), and 'Connect' (green).

In the picture below we can see that the database exist on our mongodb pod

The screenshot shows the MongoDB Compass interface displaying a document from the 'sample\_airbnb.listingsAndReviews' collection. The document has an ID of `_id: 10006546`. The document details a listing with the following fields:

- `access: "We are always available to help guests. The house is fully available t..."`
- `accommodates: 8`
- `address: Object`
- `amenities: ["TV", "Cable TV", "Wifi", "Kitchen", "Paid parking off premises", "Smoking..."`
- `availability: Object`
- `bathrooms: 1`
- `bed_type: "Real Bed"`
- `bedrooms: 3`
- `beds: 5`
- `calendar_last_scraped: "2019-02-16T05:00:00.000Z"`
- `cancellation_policy: "moderate"`
- `cleaning_fee: 35`
- `description: "Fantastic duplex apartment with three bedrooms, located in the histori..."`
- `extra_people: 15`
- `first_review: "2016-01-03T05:00:00.000Z"`
- `guests_included: 6`
- `host: Object`
- `house_rules: "Make the house your home..."`
- `images: Object`
- `interaction: "Cot - 10 € / night Dog - € 7,5 / night"`
- `last_review: "2019-01-20T05:00:00.000Z"`
- `last_scraped: "2019-02-16T05:00:00.000Z"`
- `listing_url: "https://www.airbnb.com/rooms/10006546"`
- `maximum_nights: 30`

The document is displayed in a JSON-like format with collapsible sections indicated by arrows. At the top right, there are statistics: 5.6k DOCUMENTS and 1 INDEXES. Below the document, there are buttons for 'OPTIONS', 'FIND', 'RESET', and 'REFRESH'.

## 5. API calls and saving result in CSV

### A. Set up the mongodb-express-rest-api-example

The purpose is to do API calls for inserting and listings records.  
So only the server part is interesting we will not use the frontend part of the project.

Below the tree of the server folder:

- config.env
- **db/**
  - conn.js
- **node\_modules/**
- package-lock.json
- package.json
- **routes/**
  - record.js
- server.js

The config.env file will have important information about the db url and the port

```
ATLAS_URI=mongodb://username:password@mongodb-service/?authMechanism=DEFAULT  
PORT=5001
```

The conn.js file will use the connection string url and the connection to the sample\_airbnb database

The record.js file define the routes that we will use for inserting a record and to get all the records

The server.js file is the main file for running the nodejs server

Once all the project is configured we are going to build an image with Docker and push it in a docker hub repository

## B. Creation of an image for our deployment

In the root of the project we will create a Dockerfile:

```
FROM node

RUN mkdir -p /home/app

COPY . /home/app

WORKDIR /home/app/server

CMD ["node", "server.js"]
~
```

Then we will create a build and push in a docker hub repo

```
docker build -t mongodb-express:1.0 .
docker tag d98844cf2a2c cohenj55/mongodb-express
docker push cohenj55/mongodb-express
```

## C. Deployment and service express application

- Express-mongo-depl.yaml

```
apiVersion: apps/v1
kind: Deployment
metadata:
  name: express-deployment
  labels:
    app: mongodb-express
spec:
  replicas: 1
  selector:
    matchLabels:
      app: mongodb-express
  template:
    metadata:
      labels:
        app: mongodb-express
    spec:
      containers:
        - name: mongodb-express
          image: cohenj55/mongodb-express
          ports:
            - containerPort: 5001
```

So here we can see that we are going to deploy the image that we have created where “cohenj55” is my docker hub username

- Lets have a look now on the express service configuration

```
---  
apiVersion: v1  
kind: Service  
metadata:  
  name: mongodb-express-service  
spec:  
  selector:  
    app: mongodb-express  
  type: LoadBalancer  
  ports:  
    - protocol: TCP  
      port: 5001  
      targetPort: 5001  
      nodePort: 30002
```

- Information about the pod that has been created by the deployment

| node.kubernetes.io/unreachable:NoExecute op=Exists for 300s |        |           |       |                   |
|---|--------|-----------|-------|-------------------|
| Events:   | Type   | Reason    | Age   | From              |
|   | Normal | Scheduled | 2m13s | default-scheduler |
|   | Normal | Pulling   | 2m12s | kubelet           |
|   | Normal | Pulled    | 2m7s  | kubelet           |
|   | Normal | Created   | 2m7s  | kubelet           |
|   | Normal | Started   | 2m7s  | kubelet           |

So here we can see that the image has been pulled correctly and the container start.

```
juliencohen@juliens-MacBook-Air deploymentAndServices % kubectl logs express-deployment-6855787d7b-255v9  
Successfully connected to MongoDB.  
Server is running on port: 5001
```

In the picture above we can see that the server is running on the port 5001.

Now we are going to create a tunnel for accessing to our express pod locally

```
juliencohen@juliens-MacBook-Air dropit % minikube service mongodb-express-service  
|-----|-----|-----|-----|  
| NAMESPACE | NAME | TARGET PORT | URL |  
|-----|-----|-----|-----|  
| default | mongodb-express-service | 5001 | http://192.168.49.2:30002 |  
|-----|-----|-----|-----|  
🏃 Starting tunnel for service mongodb-express-service.  
|-----|-----|-----|-----|  
| NAMESPACE | NAME | TARGET PORT | URL |  
|-----|-----|-----|-----|  
| default | mongodb-express-service | | http://127.0.0.1:51125 |  
|-----|-----|-----|-----|  
💡 Opening service default/mongodb-express-service in default browser...  
❗ Because you are using a Docker driver on darwin, the terminal needs to be open to run it.
```

## D. API call - adding records and listings records

- Postman for adding a new record

The screenshot shows the Postman interface for a POST request. The URL is `http://127.0.0.1:51125/listings/recordSwipe`. The Body tab is selected, containing the following JSON payload:

```
1 { "id": "10266175", "direction": "right" }
```

The response status is 204 No Content, with a duration of 32 ms and a size of 166 B.

- Checking in mongodb compass if a new record has been added

The screenshot shows the MongoDB Compass interface for the `sample_airbnb.matches` collection. There is 1 document and 1 index. The document details are as follows:

```
_id: ObjectId('62fcfd649a702c62902c692d7')
listing_id: "10266175"
last_modified: 2022-08-17T11:51:37.220+00:00
session_id: null
direction: "right"
```

- Postman for listings all records

The screenshot shows the Postman interface with a GET request to `http://127.0.0.1:51125/listings`. The response status is 200 OK, 47 ms, 818.86 KB. The response body is a JSON object representing an Airbnb listing:

```

1
2   {
3     "_id": "10006546",
4     "access": "We are always available to help guests. The house is fully available to guests. We are
      always ready to assist guests. when possible we pick the guests at the airport. This service
      transfer have a cost per person. We will also have service \"meal at home\" with a diverse
      menu and the taste of each. Enjoy the moment!",
5     "accommodates": 8,
6     "address": {
7       "country": "Portugal",
8       "country_code": "PT",
9       "government_area": "Cedofeita, Ildefonso, Sé, Miragaia, Nicolau, Vitória",
10      "location": "",
11      "market": "Porto",
12      "street": "Porto, Porto, Portugal",
13      "suburb": ""
14    },
15    "amenities": "[\"TV\",\"Cable TV\",\"Wifi\",\"Kitchen\",\"Paid parking off premises\",\"Smoking
      allowed\",\"Pets allowed\",\"Buzzer/wireless intercom\",\"Heating\",\"Family/kid friendly\",
      \"Washer\",\"First aid kit\",\"Fire extinguisher\",\"Essentials\",\"Hangers\",\"Hair dryer\",
      \"Iron\",\"Pack 'n Play/travel crib\",\"Room-darkening shades\",\"Hot water\",\"Bed linens\",
      \"Extra pillows and blankets\",\"Microwave\",\"Coffee maker\",\"Refrigerator\",\"Dishwasher\"]"
  }

```

Now that we added a new record and listed all the records we are going to export it

- Mongo-dump.yaml

```

apiVersion: batch/v1
kind: Job
metadata:
  name: mongodb-dump
  labels:
    app: mongo
spec:
  backoffLimit: 5
  activeDeadlineSeconds: 100
  template:
    spec:
      containers:
        - name: mongodump
          image: mongo
          args:
            - /bin/sh
            - '-c'
            - mongodump --host mongodb-service -u username -p password --authenticationDatabase admin --db sample_airbnb --out /db
          volumeMounts:
            - name: db-volume
              mountPath: /db
        volumes:
          - name: db-volume
            hostPath:
              path: /mnt/vpath
  restartPolicy: OnFailure

```

This  
job

will create a pod and perform a mongodump and save the export in the /db folder of the pod. Here we can see that a volume is attached where /mnt/vpath will be the host path So the db will be saved in the /mnt/vpath of my minikube node.

- logs pod created by the job

```
juliencohen@juliens-MacBook-Air:~$ kubectl logs mongojob-mongodump-g2nf/
2022-08-17T13:20:54.128+0000    writing sample_airbnb.listingsAndReviews to /db/sample_airbnb/listingsAndReviews.bson
2022-08-17T13:20:54.130+0000    writing sample_airbnb.matches to /db/sample_airbnb/matches.bson
2022-08-17T13:20:54.131+0000    done dumping sample_airbnb.matches (1 document)
2022-08-17T13:20:54.401+0000    done dumping sample_airbnb.listingsAndReviews (5555 documents)
```

Lets check in minikube if the dump exist in the /mnt/vpath folder

```
docker@minikube:~$ ls -la /mnt/vpath/sample_airbnb/
total 93984
drwxr-xr-x 2 root root      4096 Aug 17 13:20 .
drwxr-xr-x 3 root root      4096 Aug 17 13:20 ..
-rw-r--r-- 1 root root  96215462 Aug 17 13:20 listingsAndReviews.bson
-rw-r--r-- 1 root root     185 Aug 17 13:20 listingsAndReviews.metadata.json
-rw-r--r-- 1 root root     103 Aug 17 13:20 matches.bson
-rw-r--r-- 1 root root     174 Aug 17 13:20 matches.metadata.json
```

So here we can see that the data has been successfully exported

## Conclusion

Here we create a node with a mongodb and express pod  
The pods can communicate between them for doing the API calls.  
We can still improve it with adding an nginx reverse proxy because here the API calls are done directly on the express pod. If we add a reverse proxy server we could add some filtering rules That will be a good practice for avoiding users requesting directly on the express pod.

For the mongoimport and the mongoexport we decide to create a job, but in a production environment it will makes more sense to do a cron job that will backup the database every day for example.