08.0.2.19

בסיס הנתונים המבוזר - פרויקט סיום קורס

נושא: בניית תהליך (מנגנון) לעדכון מספר טבלאות (collections) בו-זמנית (multi-document transactions)

<u>הצעת הפתרון:</u>

1. שלבים:
   - הקמת שני אתרים (instances) של MongoDB, ראשי ומשני (Primary and Replica).
   - בניית מנגנון עדכון (CRUD) של דוקומנט בטבלאות (collections) שונות
   - פיתוח ממשק משתמש (WEB) לתצוגת התהליך

2. דרישות "must":
   - פיתוח קוד בכל שפה שהיא לבניית התהליך ב- two phase commit module (לא ב-Command Line) – 2 אינסטנסים, אחד פריימרי והשני רפליקה
   - כאשר מכבים שרת אחד, המערכת תעשה roll back
   - יש לבצע את כל תהליכי CRUD (Create/Read/Update/Delete)
   - הצד של MongoDB הינו LINUX
3. דרישות "Nice to Have"
   - בניית שני אתרים בשני שרתים שונים
   - שימוש בשני בסיסי נתונים שונים
   - בניית מנגנון rollback (אם two phase commit לא הצליח)
   - הפרויקט כולו (גם הממשק) פותח בסביבת LINUX

חומר עזר:

https://www.quora.com/How-do-you-perform-multi-document-transactions-in-MongoDB

https://blog.csdn.net/sunbocong/article/details/79962348

/https://docs.mongodb.com/manual/core/transactions

https://docs.mongodb.com/manual/tutorial/deploy-shard-cluster/

mongoimport -h ds221155.mlab.com:21155 -d nofar -c costumers -u nofar -p nofar101 --file costumers.csv --type csv --headerline

mongodb://nofar:nofar101@ds221155.mlab.com:21155/nofar

mongod --dbpath /tmp/data --replSet rs

mongod --dbpath \tmp\data --replSet rs

mongod --dbpath /data/db --port 27017

--bind_ip 127.0.0.1

C:\Program Files\MongoDB\Server\4.0

Deploy Sharded Cluster

Create the Config Server Replica Set

1.Start each member of the config server replica set.
mongod --configsvr --replSet <replica set name> --dbpath <path> --bind_ip localhost,<hostname(s)|ip address(es)>

2.Connect to one of the config servers
mongo --host admin --port 27017

3.Initiate the replica set.
From the mongo shell, run the rs.initiate() method.

Create the Shard Replica Sets

1. Start each member of the shard replica set
mongod --shardsvr --replSet <replSetname> --dbpath <path> --bind_ip localhost,<hostname(s)|ip address(es)>

2.Connect to one member of the shard replica set
mongo --host <hostname> --port <port>

3.Initiate the replica set
From the mongo shell, run the rs.initiate() method.

```
use products
db.createUser( { user: "accountAdmin01",
                 pwd: "changeMe",
                 customData: { employeeId: 12345 },
                 roles: [ { role: "clusterAdmin", db: "admin" },
                          { role: "readAnyDatabase", db: "admin" },
                          "readWrite"] },
               { w: "majority" , wtimeout: 5000 } )

db.getUsers()

db.createCollection(<name>, { capped: <boolean>,
```

autoIndexId: <boolean>,
size: <number>,
max: <number>,
storageEngine: <document>,
validator: <document>,
validationLevel: <string>,
validationAction: <string>,
indexOptionDefaults: <document>,
viewOn: <string>,
pipeline: <pipeline>,
collation: <document>,
writeConcern: <document>})

mongodb+srv://admin:nofar101@cluster0-ehgxt.gcp.mongodb.net/test?retryWrites=true

mongodb-4.0-demos/start-mongo.sh

```
docker run --rm -d -p 27017:27017 --name mongo mongo:4.0.5 --replSet rs

docker exec -it mongo mongo --eval 'rs.initiate'()
```

docker run

--rm : remove container automatically after it exits

-it : connect the container to terminal

--name : name the container

-p 27017:27017 : expose port *27017* externally and map to port *27017*

-v ~/dev:/code : create a host mapped volume inside the container

mongodb-4.0-demos/connect-mongo.sh

```
docker exec -it mongo mongo
```

Create a new mongo process inside the container and connect it to the terminal

mongodb-4.0-demos/compile-docker.sh

```
docker run -it --rm -u $(id -u):$(id -g) -v "$HOME/.m2":/var/maven/.m2 -v
"$(pwd)":/usr/src/mymaven -w /usr/src/mymaven -e MAVEN_CONFIG=/var/maven/.m2 maven:3.5.4-jdk-
10-slim mvn -Duser.home=/var/maven clean package
```

-v : Remove the volumes associated with the container

mongodb-4.0-demos/change-streams-docker.sh

```
docker run --rm -it --network host -v "$(pwd)"/target:/target openjdk:10.0-jre-slim java -cp
/target/mongodb-4.0-demos-1.0.0-jar-with-dependencies.jar com.mongodb.ChangeStreams
mongodb://localhost/test
```

--network : Specify which networks your container should connect to
-cp : Copy files/folders between a container and the local filesystem

mongodb-4.0-demos/transactions-docker.sh

```
docker run --rm -it --network host -v "$(pwd)"/target:/target openjdk:10.0-jre-slim java -cp
/target/mongodb-4.0-demos-1.0.0-jar-with-dependencies.jar com.mongodb.Transactions
mongodb://localhost/test?retryWrites=true
```

BSON is a computer data interchange format. The name "BSON" is based on the term JSON and stands for "Binary JSON". It is a binary form for representing simple or complex data structures including associative arrays (also known as name-value pairs), integer indexed arrays, and a suite of fundamental scalar types. BSON originated in 2009 at MongoDB.

1. Open a docker terminal
2. Go to: "`cd ../../Users/HP/eclipse-workspace/mongodb-nofar-project`"
3. "`docker run --rm -d -p 27017:27017 --name mongo mongo:4.0.2 --replSet rs `"
4. "`docker exec -it mongo mongo --eval 'rs.initiate()'`"