**Enterprise Technologies**

**Final Project**

**Report**

745766- Gursharan Preet Kaur

735952 – Sukhpreet Singh Sidhu

745113 – Samkit Shah

**Questions**

1. Describe how this application can be deployed in a cloud environment.
2. Describe how you would be able to implement security features for your app in the cloud.
3. If you need to make your application serverless how it can be done.

**Answers**

1 – There is no need to install MongoDB server and configuring it. Once the data is on the MongoDB connect to Atlas and it is ready to be deployed on to cloud platforms like AWS, Google Cloud, Azure. These platforms give the option of running your cluster on their cloud based platform.

You will need to create an account on MongoDB atlas. Once the required settings options have been selected like region etc., you will be prompted to create a cluster.

Once the cluster is ready you can start adding collections and documents to it.

In Clusters security enable a user and give them read and write access or else give them admin access and next allow access from anywhere.

Next step is deciding which cloud service you would like to install or publish your app on.

Let’s for example take google cloud platform.

Once signed up, start a project and enable the specific API need to run the app, in this case we will need Google App Engine Flexible Environment and Google App Engine Admin API. In the app engine you will have to select the language which is Nodejs for us.

After certain steps like selecting region and creating a Debian Linux GCE VM you will allow for full access to all cloud APIS and SSH the instance.

Next we execute network building code to run node.js and mongoDB client.

After initializing it and installing all dependencies we creating a server.js file we use this file to insert our atlas connection string.

Now you can run the app locally or push the app to GCP.

Once pushed you can visit the application form any web enabled client terminal including mobile devices.

2 – Once the application is connect to the MongoDB Atlas environment, you can whitelist the inbound network access from the application to Atlas. This can be done by either whitelisting public IP addresses or using VPC/VNet peering to whitelist private IP address.

Atlas uses TLS/SSL to encrypt the connections to your databases.

User Authentication/ Authorization, atlas requires client to authenticate to access the clusters. The person trying to access the database has to be a valid user registered to have the permissions he is trying to have access to.

You can use Custom MongoDB roles to provide more security even among users who have access to the database. For example not assigning everyone write permissions to the DB so unauthorized people cannot change the data.

Switch on Two Factor Authentication to help users control access to their Atlas accounts.

Atlas support using AWS KMS, Azure Key Vault and GCP to encrypt storage engines and cloud provider backups. These features are switched on before publishing your app one the respecting cloud platform.

Organization owners can also restrict MongoDB production Support Employees from accessing Atlas backend infrastructure for any.

Owners have the option of allowing a grand of a 24 hour bypass to the access restriction at the Atlas cluster level.

Mongodb allows from Access Tracking

Atlas surfaces authentication logs directly in the UI so that you can easily review successful and unsuccessful authentication attempts made against your clusters.

3-

There are a number of ways to make your Atlas cluster serverless.

For this answer we took the example of MongoDb Stitch.

MongoDB stitch is a backend-as-a service solution. It also supports custom functions similar AWS.

Log in Atlas cluster and link application to create a new Stitch app.

Next click stitch function and create a function. In order to run the function we need to create users in the providers tab.

Next Enable the API keys.

Click on HTTP on under a service. Make sure to save all the changes made.

After ensuring that the stitch function has unrestricted access to all to the cluster and its collections.

Now make and export the function with the code that connects to the atlas cluster and displays the required data from the documents from the cluster.

In order to run the function click on console and run function as.

Now create a read() function. This is used for the API to return values.

Once this function is up and running, Create a webhook using a HTTP client. Make sure the webhook responds to POST requests. Once connected to the Atlas cluster this should be able to post and fetch data out of collections into the app.

GITHUB link –

<https://github.com/sukhpreetsinghsidhu/EnterpriseFinalProject>

Google Drive link –

<https://drive.google.com/open?id=1UxS2k8z-atQmFL2BvH0tbznB-_HKfA0u>

Screenshot of the atlas cluster –

