**MSDS 7330: Final Project**

From the Top and MongoDB Atlas

From the Top (FTT) is America’s largest national platform dedicated to celebrating the stories, talents, and character of young classically-trained musicians. The organization’s weekly NPR radio broadcast is the most popular weekly one-hour classical music program on public radio and heard on more than 220 stations nationwide by audiences of more than half a million listeners. Each program features five or six performances and interviews that reveal the heart and soul behind the talent of these extraordinary young musicians.

**Situation**

Each year, FTT receives upwards of 800 online applications from solo instrumentalists, vocalists, composers, small ensembles with fewer than 8 musicians, and large ensembles with more than 9 musicians. Applicants, ranging in age from 8 to 18, are asked to complete an extensive application that includes: basic contact and demographic information about students, parents, and teachers; several short answer essay questions; and audio and/or video sample performances. A $70 fee is also collected from applicants.

The system FTT is currently using to power the application process has increased dramatically in price, so it is now necessary to explore alternative, budget-conscious options than do not require significant or ongoing technological, capital, or human resource investments. Key requirements for a new system include:

* Flexibility to collect, store, and retrieve a variety of text-based information;
* Ability to accept, catalogue, and store audio and video files submitted by students as part of the application process;
* Capability to securely accept credit card numbers for processing of application fee;
* Compliant with personally identifiable information protocols and the Massachusetts Data Protection Act, which further stipulates the handling of personal information;
* Solution that balances need for strict security protocols with FTT’s limited technological capacity beyond contract IT support and this consulting data science team; and
* Meeting budget constraints.

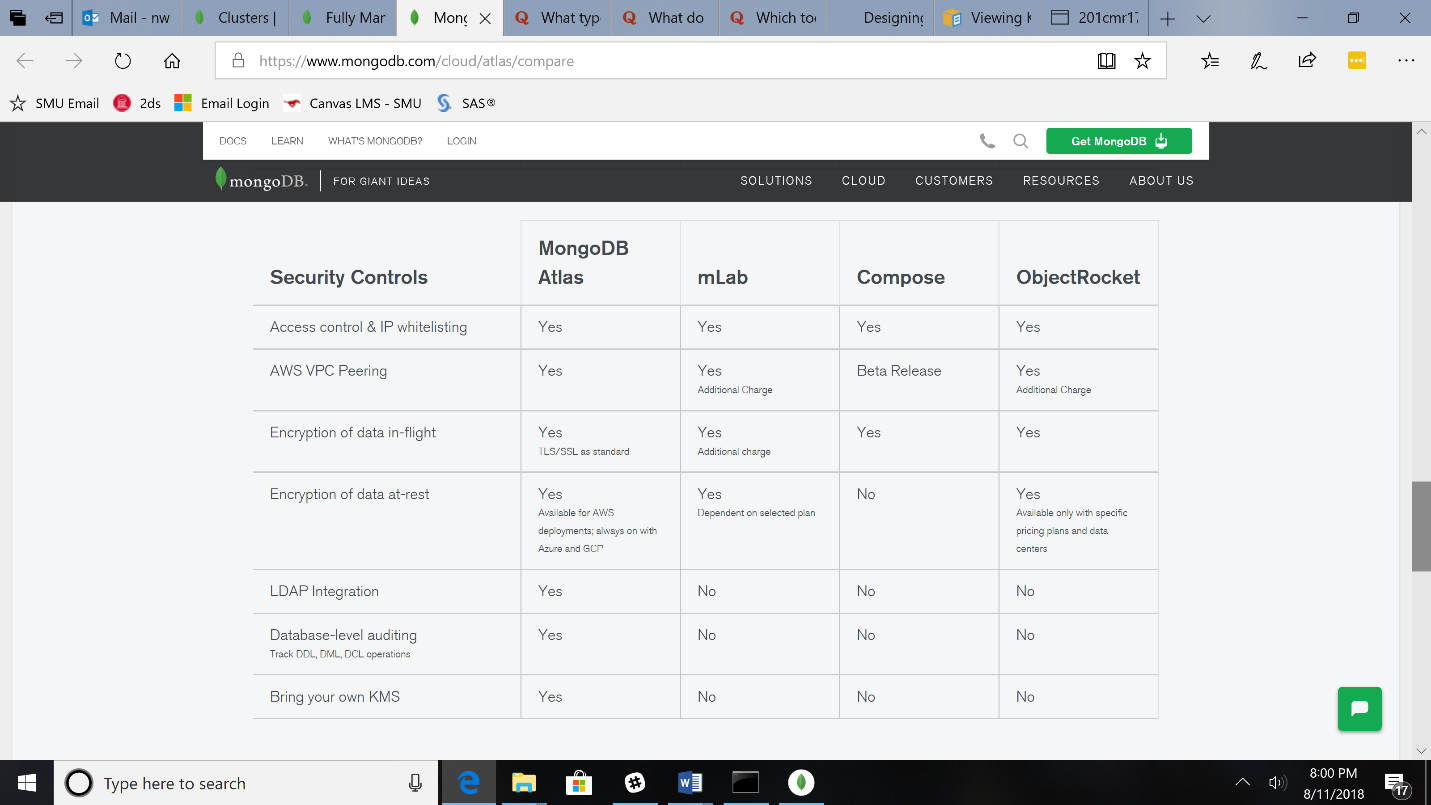
To assist FTT in exploring a new system that meets these requirements, the team took a multi-pronged approach with the scope of this project. The goals were to: research encryption capabilities for a possible new system; suggest and set up a test database for FTT to explore NoSQL options; and recommend next steps to move the project forward.

As a starting place, the team recommends FTT consider the NoSQL database MongoDB to capture application information and materials. MongoDB is noted for being “an open-source document database that provides high performance, high availability, and automatic scaling,” per the company’s website ([link](https://www.mongodb.com/)). MongoDB Atlas is an upgraded version that bring many encryption capabilities already built-in and has compatibility with Amazon Web Services (AWS), Microsoft Azure, and Google Cloud Platform (GCP). This a good option for FTT given the need to collect credit card payments and personally identifiable information, as well as the sensitivity of collecting information about minors. Additionally, the compatibility with AWS offers other resources to consider for the application process.

**Key Tools, Technology, and Resources**

Before setting up the database infrastructure needed as the foundation for FTT’s application process, the team implemented several tools to meet security needs outlined above.

*MongoDB Atlas*

The team’s first step was to install and set up MongoDB Atlas. Once Atlas was running and the project created, an initial decision was to create a cluster to ultimately house data. There are several size options to allow for scalability, a hallmark of MongoDB, and the cost structure is based on what is used. Clusters range from shared options, which are intended as “sandboxes” to get started with MongoDB (shared RAM, shared vCPUs, and 512MB storage) to production environments for large datasets or high-traffic applications (488GB RAM, 64 vCPUs, and 3000GB storage). The team selected the M0 cluster, a free sandbox option (see Technical Appendix pg. 7 for Cluster details).

Another hallmark of MongoDB Atlas is the strong security protocols built into the database as standard options. Most notably, there are protocols that allow for constant, “always-on authentication,” encryption of data both while at rest and in transit, and advanced user access settings based on roles. Additionally, rotating security keys can be enabled. Other features include: TLS/SSL encryption protocols, authentication, and authorization via SCRAM; network isolation and virtual private clouds on AWS; IP whitelists; encrypted storage volumes; and the MongoDB Atlas console to manage users. The team was most interested in the following tools and invested time to set these up initially.

Whitelist IP Addresses

MongoDB Atlas clusters do not have access to the Internet. Additionally, in AWS, the virtual private clouds (VPC) are configured with no inbound Internet access by default. Therefore, an early step in process was to whitelist the IP addresses of the two team members. This setting can be found under the Security tab in MongoDB Atlas under the project’s cluster (see appendix pg. 8).

Encryption in Transit

In the MongoDB Atlas default settings, data is encrypted in transit when communicating back and forth with the server to view or edit data using TLS/SSL cryptographic protocols. TLS uses symmetric key cryptography to encrypt the data transmissions to ensure data privacy, and a “handshake” agreement between the user and database that is unique in every connection to exchange details on the keys and encryption algorithms. These protocols provide assurances on data confidentiality, so that no eavesdropping occurs where outside parties could review the data.

Authentication/Authorization

To access the clusters of data in MongoDB Atlas, user credentials must be created with privileges to access the cluster(s). Each cluster can hold several databases, and each database within MongoDB can have different user logins and permissions. The default authentication process for these logins is SCRAM (Salted Challenge Response Authentication Mechanism), which is based on the Internet Engineering Task Force’s proposed standards from July 2010 (see IETF’s [RFC 5802](https://www.rfc-editor.org/info/rfc5802)). SCRAM in MongoDB version 4.0 uses SHA-256; however, the sample FTT database is built in MongoDB Atlas, where the most recent version is 3.6 and only supports SHA-1 for a hash algorithm (see appendix pg. 9-10).

Two-factor Authentication

Additionally, customers using the Web User Interface (UI) have the option to set up two-factor authentication in MongoDB for data in the GUI. Passwords, keys, and credentials are all stored as encrypted when two-factor authentication is enabled. The team felt this was an important added layer of security to set up.

Mongo Compass

Mongo Compass is the graphical user interface, or GUI, for MongoDB. This is where FTT staff would mostly be interacting with application data, and there are many user-friendly tools to interact and edit data, query for results, and create visual analysis (see appendix pg. 10-11).

*Amazon Web Services*

Of the three cloud computing options that pair with MongoDB Atlas, the team selected AWS, which provides a broad set of infrastructure technology services such as computer power, storage options, networking, and databases. Services are available on-demand and in a “pay-as-you-go” pricing structure, which may be more cost effective for FTT to consider.

Encryption at Rest

To encrypt data at rest, MongoDB Atlas utilizes a symmetric key supplied by AWS based on the algorithm AES256-CBC, which is a 256-bit key using the Advanced Encryption Standard symmetric-key cipher in Cipher Block Chaining mode. This is a very strong cipher and key size to protect against brute-force attacks. There is also regular rotation of the master encryption key, which provides additional levels of security as a moving target of sorts. Thus, hackers would need to repeat cryptanalysis every time the key is rotated. The keys can be set to automatically rotate in AWS once a year, or can be manually done at any time. The encryption though comes at an increased fee in MongoDB Atlas, but given the need to protect personally identifiable information related to the credit card transactions and information about minors, this level of security is warranted (see appendix pg. 12-13).

Multi-Factor Authentication

AWS multi-factor authentication (MFA) provides an extra level of security that can be applied to the AWS environment. MFA can be enabled for the overall AWS account and for individual AWS Identity and Access Management (IAM) users created in the account. This provides a level of security beyond just master username and login credentials for the AWS account holder, which is important since this account has access to the current symmetric key used for encryption (see appendix pg. 13-14).

Amazon Simple Storage Solution (AWS S3)

It is not possible to store the media files directly in MongoDB; therefore, a critical piece of the infrastructure is a complementary system to store the audio and video files uploaded by students during the application. Amazon S3 is a service that provides scalable and highly durable object storage in the cloud. Since AWS in already used as part of the data encryption process, it was logical to consider S3 as an option.

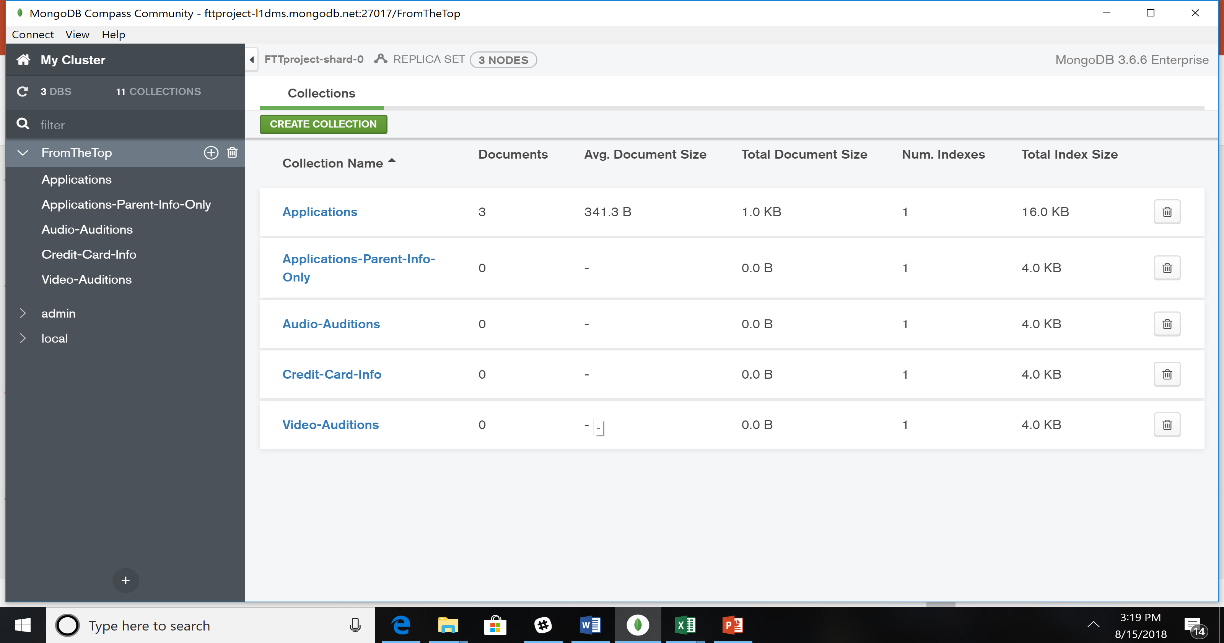
**Database Development Methodology**

MongoDB is a NoSQL database, which has several distinct advantages over a traditional SQL database structure. These types of databases are document-oriented and designed to store semi-structured data. JavaScript Object Notation (JSON) is a common NoSQL language and one of the main formats for data exchange on the Internet. It is both human and machine readable and supports all basic data types. MongoDB stores JSON documents in a binary-encoded format called BSON, which extends the JSON model to manage additional data types, ordered fields, and efficient coding in different languages.

Other advantages are that NoSQL databases such as MongoDB are “schemaless” and rely on key-value pairs to store data. Not needing to adhere to a strict schema (as is the case in a SQL database) allows for more flexibility in collecting data initially and adding additional information to the data collection process at a later point. It is likely that FTT will continue to add required information to the application process, so this is one reason in favor of a NoSQL option.

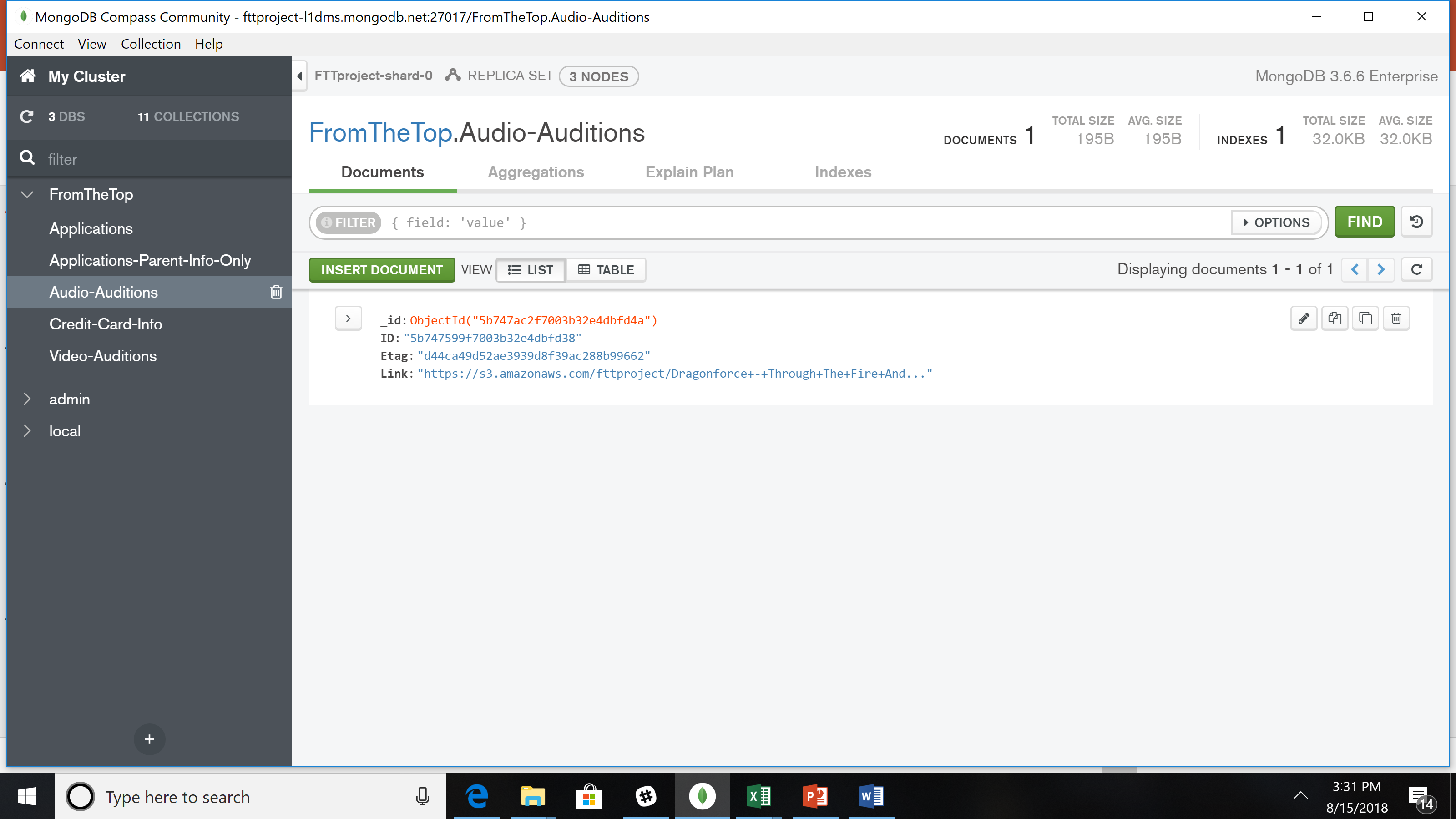
*Sample Database*

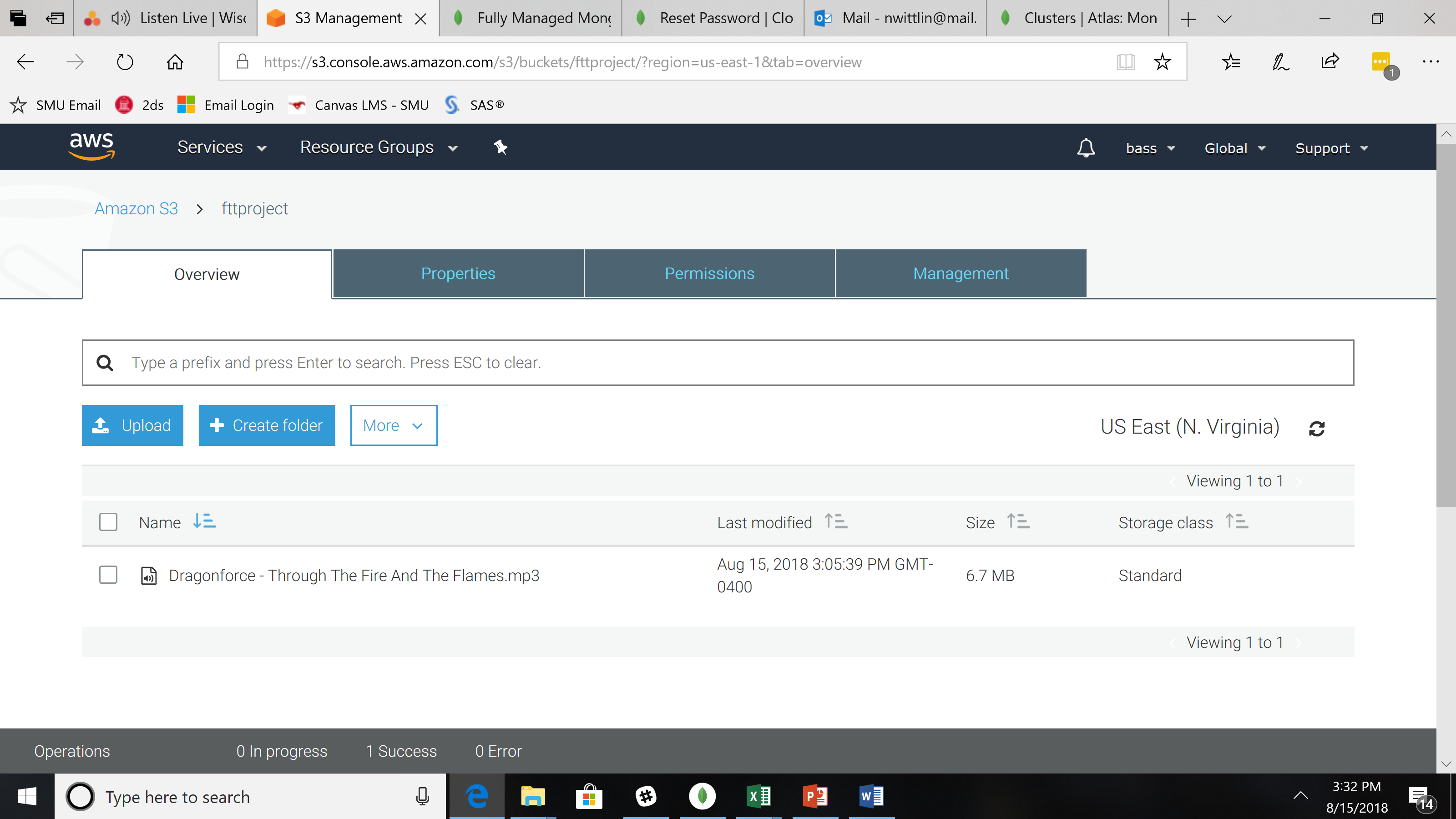
To introduce MongoDB to FTT, the team created a sample database with fictious data about “staff applicants.” This application set up is simpler than FTT’s current application for demonstration purposes. Discussions are underway internally at FTT about the content of the application, and future work on this project will involve finalizing this and the other application types FTT accepts from musicians.

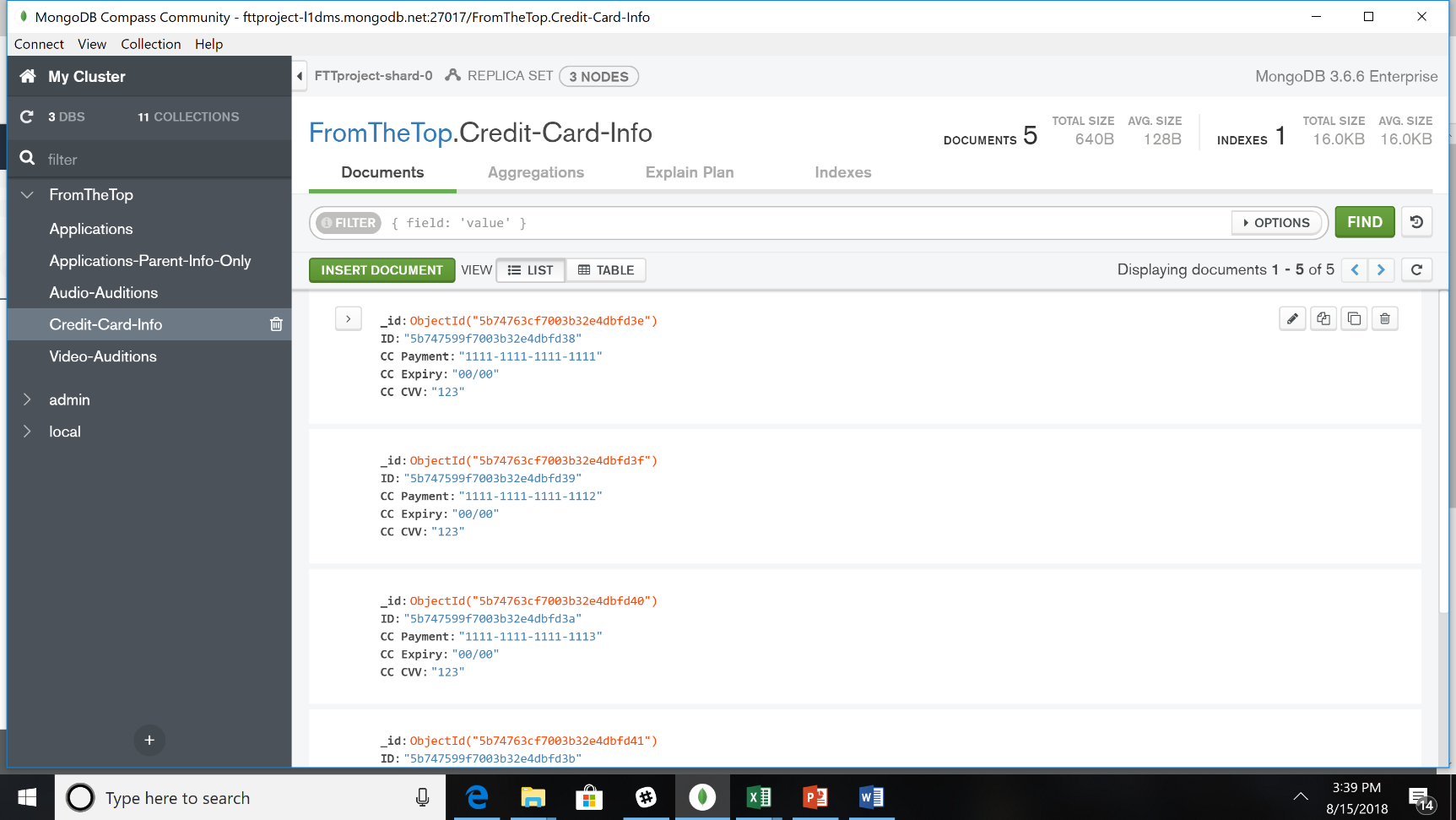


Collections

For application text, the team set up one collection “Applications” to allow for easier and more straightforward querying and analysis. A structural advantage in MongoDB is the ability to embed related data into a single document. This is also known as a “denormalized” data model and stores related pieces of information in the same record. Given the overall simplicity of FTT’s data, this structure makes the most sense and also may make querying data more straightforward once in production.

There are also separate collections for audio files (“Audio-Auditions”) and video files (“Video-Auditions”). It is not possible to store the native audio and video files directly in MongoDB, so the team is recommending FTT store the media files in AWS. The Mongo collections will be an index to capture the unique id of each file to facilitate look up during the review process.



Lastly, the team isolated parent information and credit card information into independent collections so that only FTT’s finance team can process the $70 application fee.

User Profiles

The team set up sample user profiles to mirror key FTT staff roles. For initial demonstration purposes these roles include:

* ftt-master-access: database administrator, contract IT consultants, and data science team; this role manages all the encryption, security protocols, and user access.
* ftt-cc-only-access: for FTT’s finance team to charge the application fee on credit cards; this is a very limited role that performs one function related to the application process.
* ftt-parent-and-video-audio access: for FTT music, admissions, and scholarship teams to review applications and decide on radio show performers; this role is the most comprehensive consumer of the application information, reviewing all text and short answer questions, listening to audio files and watching videos, and eventually casting each FTT radio show. This is a non-technical role.

Media Storage

AWS S3 provides a scalable repository for audio and video media uploaded during the application process. Native files will not be stored directly in MongoDB, but Mongo will serve as the reference guide to connect individual applicants with their music samples in AWS S3. FTT staff can access a weblink directly from the Mongo to review, or they can be logged into AWS.

**Conclusion and Future Work**

It has been a pleasure working with FTT to explore a possible infrastructure for their online application system. The team believes that MongoDB Atlas paired with the security protocols provided in Amazon Web Services meet many of the needs FTT has for the system and strongly recommends the project continue to the next phase of work.

Future work needed to move this project out of testing and into production includes:

* Designing and implementing more nuanced MongoDB Atlas database structure to reflect true use case data for the various types of applications and data that FTT collects;
* Conducting a full cost evaluation to determine if recommended system provides a less expensive option for FTT
* Extensive training with FTT staff to increase comfort level with MongoDB Compass and set up commonly used queries for review process
* Researching, prototyping, pricing, and implementing a secure, web-based form to mine HTML from application and feed to MongoDB Atlas for processing.

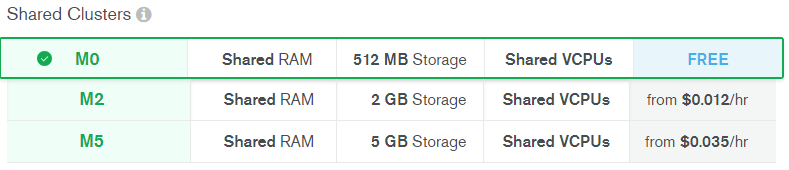
**MSDS 7330: Final Project**

Technical Appendix

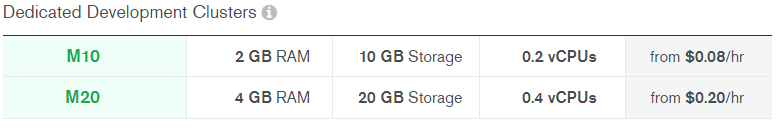
**Screen Shots of Key Tools, Technology, and Resources**

*MongoDB Atlas*

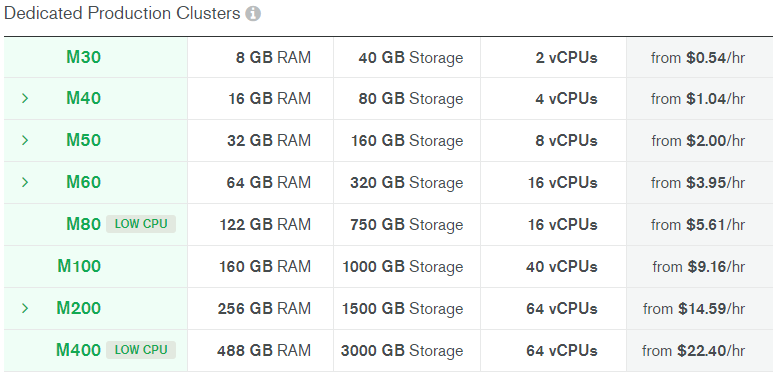
Detailed Information about Cluster Set Up



Shared clusters are “sandbox instances used for getting started with MongoDB.”



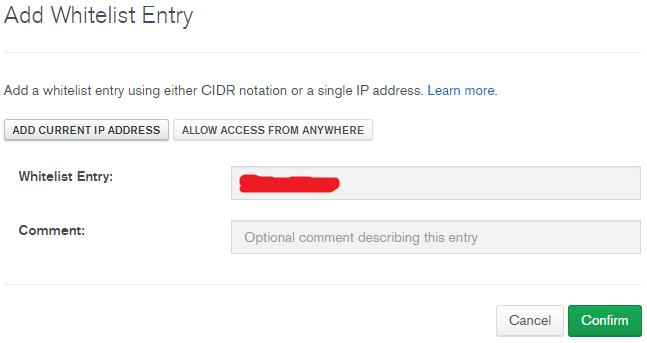
Dedicated development clusters are “best suited for development environments and low-traffic applications.”



Dedicated production clusters are “best suited for production environments, supporting high-traffic applications or large datasets”

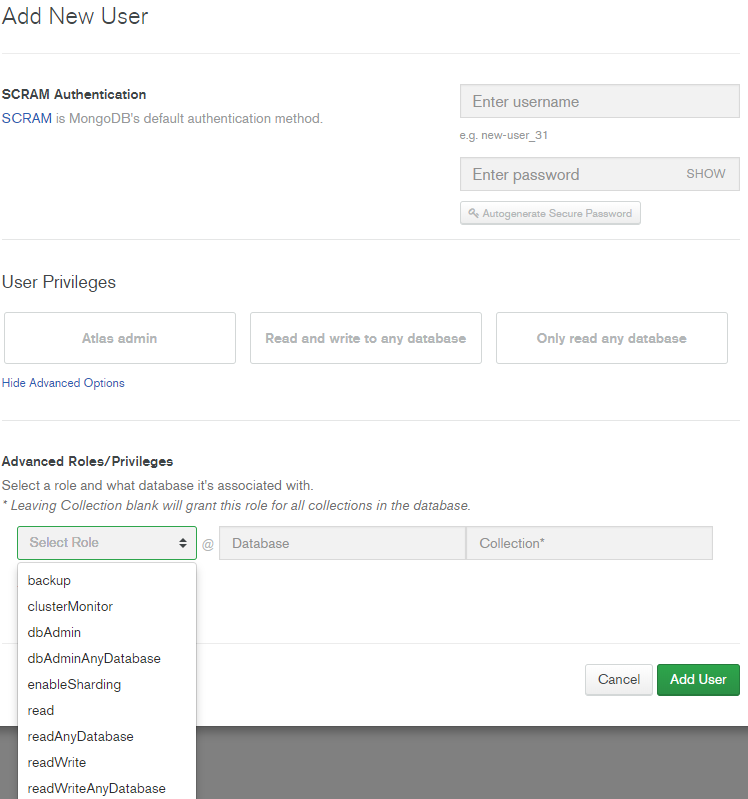
Whitelist IP Addresses

While setting up your initial cluster, MongoDB Atlas will prompt you to whitelist (allow access to) IP addresses that are working on the database. This is available under the Security tab under your project’s cluster.

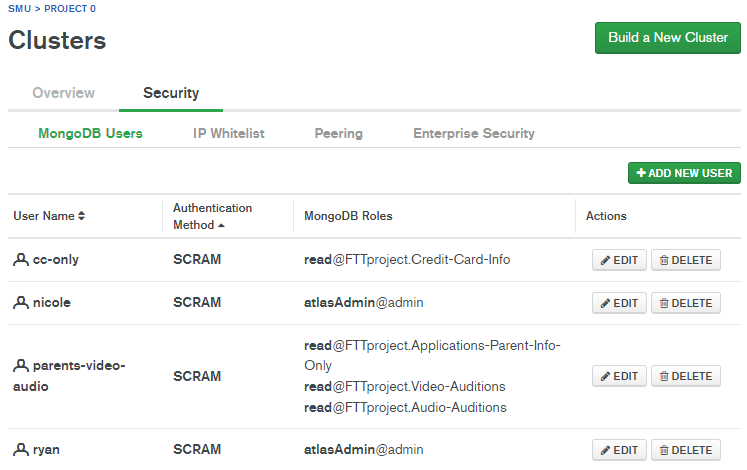


Authentication/Authorization

User access can be selected per user using the options below. This is also available in the Security tab under MongoDB Users when you add a new user. Options are shown to customize the user’s access, including admin per database and collection.

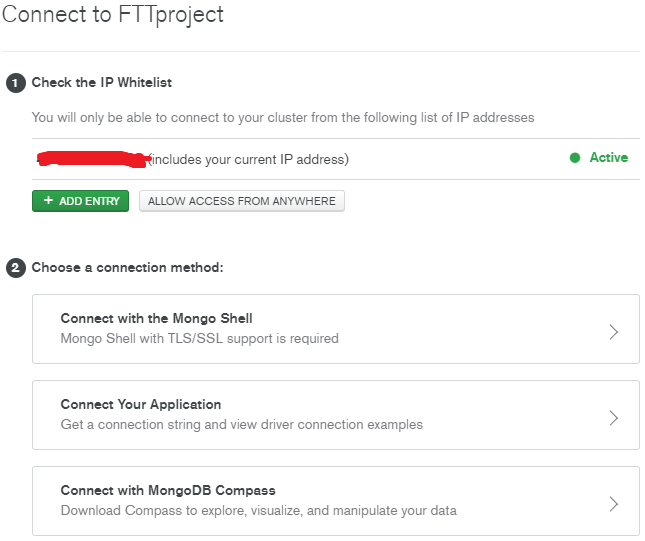


Example logins were created below. Nicole and I were given full admin rights while we made read only rights for logins to have access to the credit card information of our database (a collection called Credit-Card-Info), and read only rights for logins to have access to the audio and video applications as well as their parents’ information for contact purposes.

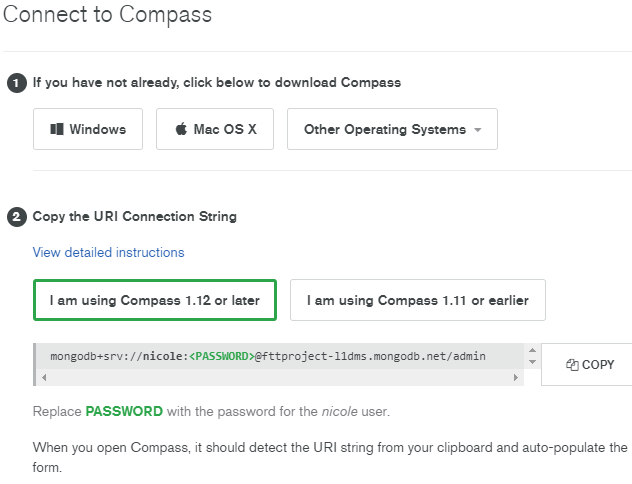


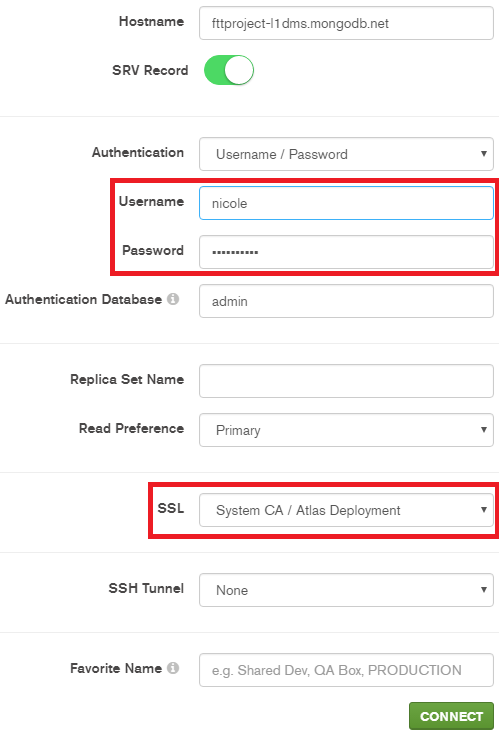
Mongo Compass

Now to connect to cluster you made you have a few options. We recommend Mongo Compass (the GUI application of MongoDB). Here whitelisted home IP addresses so we can connect.



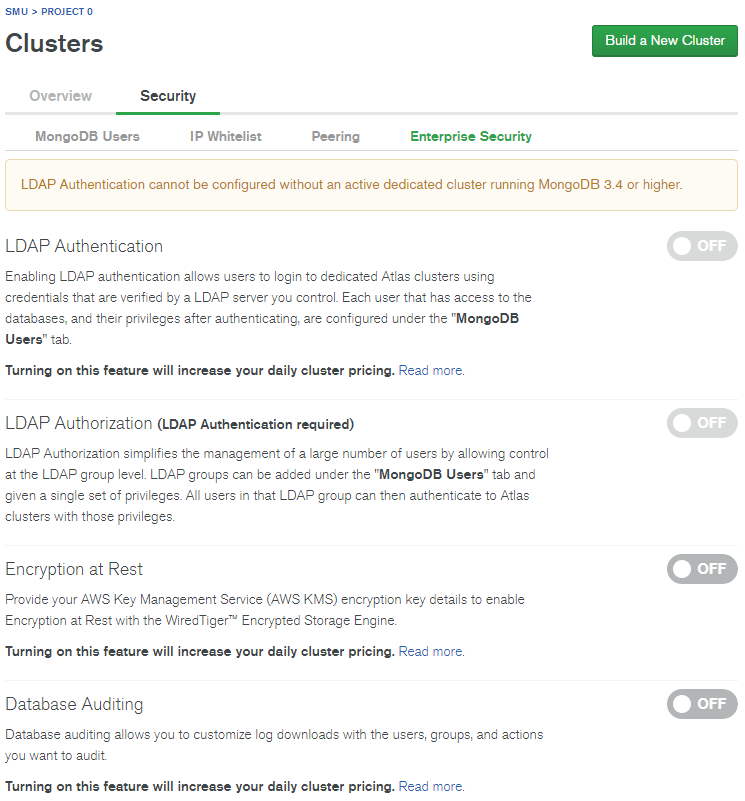
If connecting through Compass it will give you the option to download it if you have not already done so. Then select the version of compass you’re using and will yield a URI to copy into your compass (only need to replace the username/password).

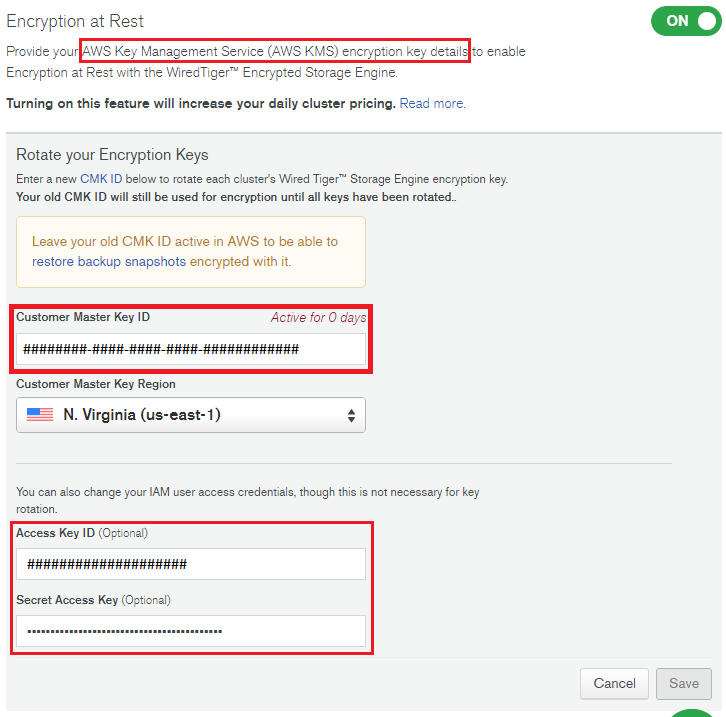


Now when Compass is opened it should auto detect the URI and only need your user’s password. Notice how the SSL defaults to System CA / Atlas Deployment. This is because working with MongoDB Atlas uses TLS/SSL as standard for data in transit (the encryption of communications with the database to prevent packet sniffing, man-in-the-middle attacks, etc.).

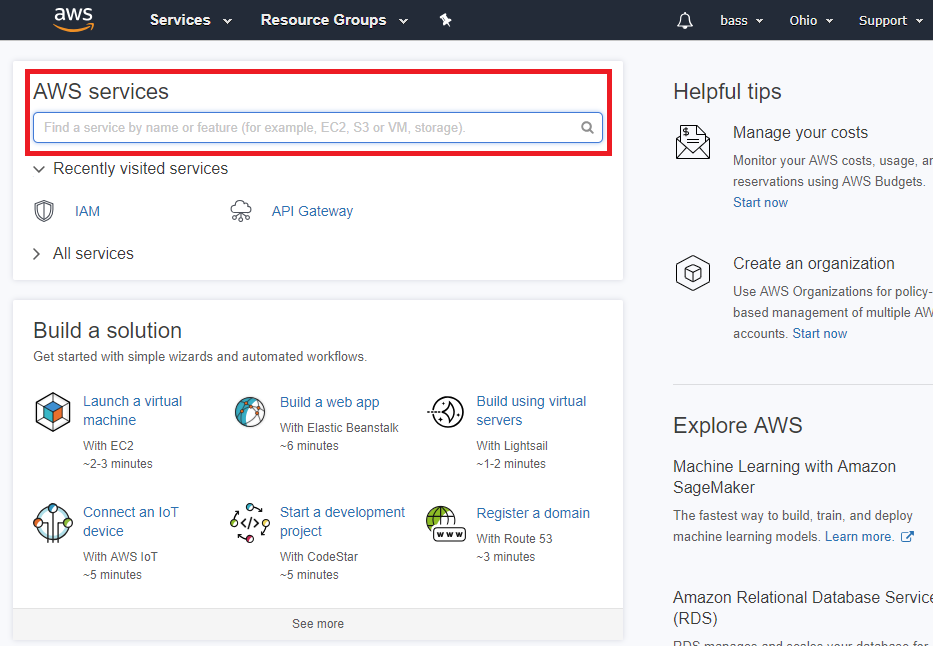
*Amazon Web Services*

Encryption at Rest



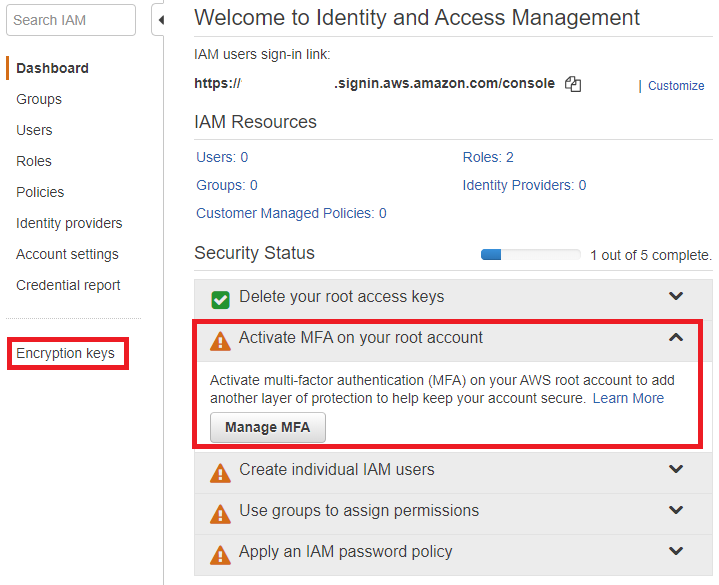


In order to conduct encryption at rest so sensitive data can be stored safely we had to create an Amazon Web Services (AWS) account to create a master key for our database. On the AWS dashboard search for IAM (Identity and Access Management) to be directed to the correct service to generate keys. Although the keys are generated in AWS, the encryption takes place through the Atlas GUI (pictured above) under the Security tab then Enterprise Security sub-tab.

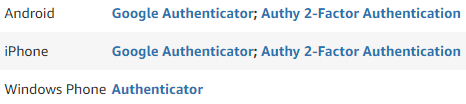


Multi-Factor Authentication

At the IAM screen you can delete your root key to prevent unlimited access to all, activate multi-factor authentication (MFA) as to improve authentication, and set up individual/group IAM users and privileges. On the bottom left portion of the left dashboard is ‘Encryption keys,’ we’ll use this after we set up the IAM. Click manage MFA, we utilized a virtual MFA by downloading the Google Authenticator app on our iPhones. It requires a scanning of the QR code that is presented so you can be authenticated.

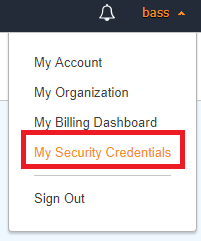


In order to activate our MFA we used a virtual MFA device as suggested by AWS. The options for each type of mobile OS are shown below (source: <https://aws.amazon.com/iam/details/mfa/>).

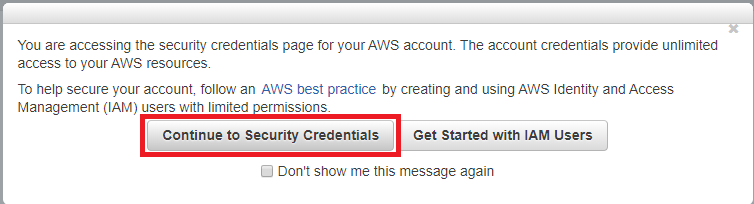


To use Google Authenticator the QR code provided on the AWS IAM screen needs to be scanned through the Google Authenticator app. The first authorization code needs to be entered, and once it expires the second one needs to be entered as well.

To obtain the Access Key ID and Secret Access Key needed to encrypt the database on the Atlas GUI you must go into your security credentials found on your AWS user login in the top right corner of the webpage.



A popup will ask if you want to continue to Security Credentials for the entire AWS account. Creating sub-IAM accounts that have access to only certain aspects of the AWS is more appropriate but since this is a testing of the database I will continue with Security Credentials. A source for the rest of the walk through can be found here: <https://www.cloudberrylab.com/blog/how-to-find-your-aws-access-key-id-and-secret-access-key-and-register-with-cloudberry-s3-explorer/>.



*Resources Consulted*

* [MongoDB Atlas Security Controls](https://webassets.mongodb.com/_com_assets/collateral/Atlas_Security_Controls.pdf?_ga=2.182608547.402734192.1534006027-1880940793.1532485333)
* <https://www.mongodb.com/>
* <https://docs.atlas.mongodb.com/getting-started/?_ga=2.240889478.402734192.1534006027-1880940793.1532485333>
* <https://d1.awsstatic.com/whitepapers/aws-overview.pdf>
* <https://aws.amazon.com/getting-started/>