Documentation for preparing the AMR data

As the Staphopia API does not provide information regarding antimicrobial resistance (AMR) genetics of *Staphylococcus Aureus (S.Aureus)* so the Webscape has decided to prepare AMR genes. This documentation dedicates to how to install necessary tools and use them.

Source to the documentation of Staphopia API:
https://staphopia.emory.edu/docs/api/

1. Install AMRFinderPlus

AMRFinderPlus is a tool to find AMR genes in the bacterial. This tool can generate results so the front-end side can visualize the AMR genes for the website.

Note: this tool is only available on Linux and not on Window and installation via putting command lines in the terminal.

Step 1: Install Miniconda for Linux by using these commands in the terminal:

```
curl -O https://repo.anaconda.com/miniconda/Miniconda3-latest-Linux-x86_64.sh
bash ./Miniconda3-latest-Linux-x86_64.sh
```

Step 2: Install AMRFinder with bioconda

Make sure the conda environment you just created is activated:

```
source ~/miniconda3/bin/activate
```

Install AMRFinder and all of the prerequisites:

```
conda install -y -c bioconda ncbi-amrfinderplus
```

Step 3: Update the tool:

```
conda update -y -c bioconda ncbi-amrfinderplus
```

2. Use AMRFinderPlus

This section show how to use the tool correctly.

Step 1: Create a python file called fasta_generator.py with the code below:

```python
import requests
import sys
import os
import shutil

def main():

    input_file = sys.argv[1]
    with open(input_file,'r') as samples_file:
        samples = samples_file.read()
        samples = samples.split('\n')

    try:
        for sample in samples:
            response = requests.get('https://staphopia.emory.edu/api/sample/' + str(sample) +'/contigs/',
                                    headers={'Authorization': 'Token de28e2ce809de4202d3232bdaab977d0f33a550e'})

            f = open("contig_fasta/" + str(sample) + ".fasta", "w")

            results = response.json()['results']
            for r in results:
                contigID = r['contig']
                sequence = r['sequence']
                f.write(">contig" + contigID + " " + str(sample) + "\n")
                f.write(sequence + "\n")
                f.write("\n")
        f.close()
    except:
        err = sys.exc_info()[0]
        print("Error: " + str(err))


if __name__ == "__main__":
    main()
```

Alternative: There is a python file called fasta_generatot.py inside of ‘datasets’ folder, which is located in ‘code’ folder.

The python file create a file in fasta format (a file only contains genetics sequence) bygetting the Assembled Contigs of each *S.Aureus* sample from Staphopia API.

Step 2: Create a samples.txt and each line put the wanted sample ID of *S.Aureus* available in the Staphopia API.

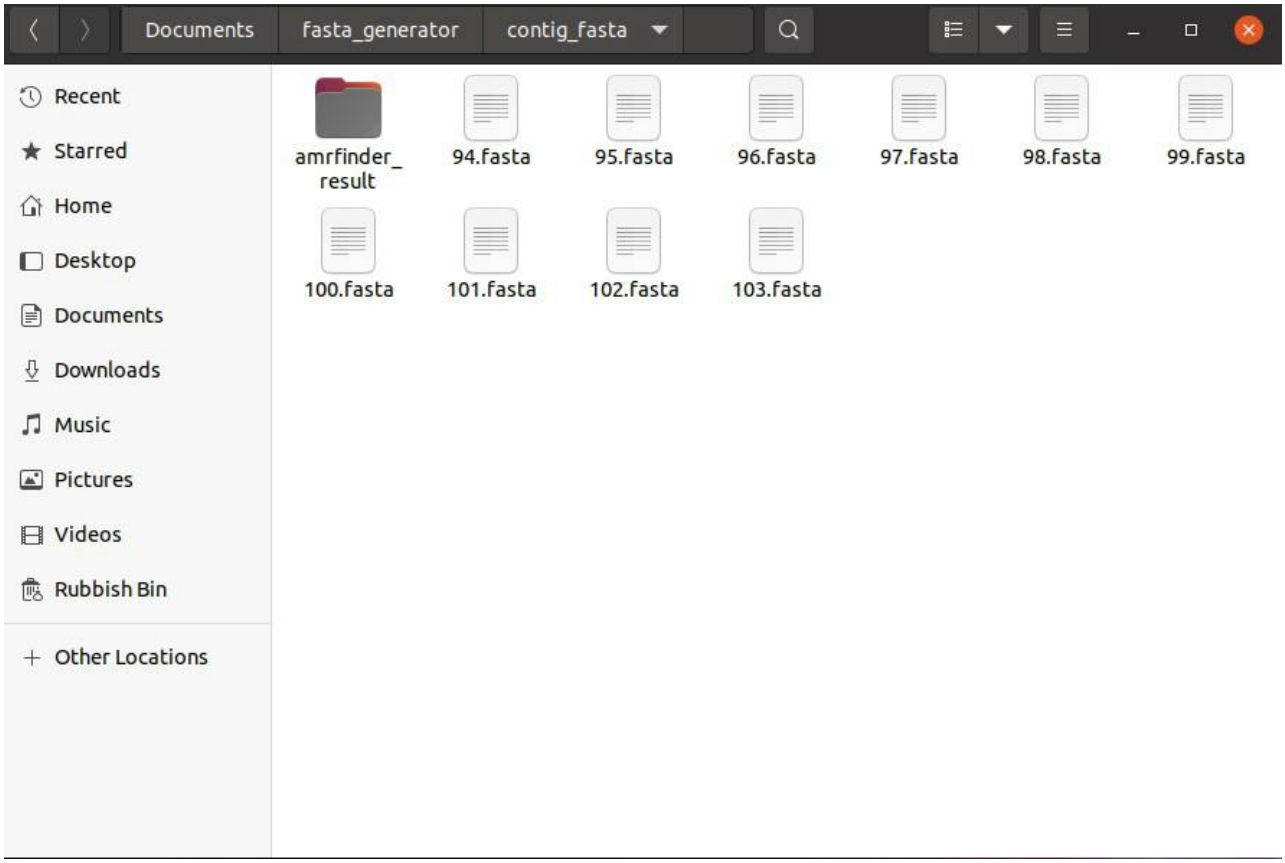
Example of Sample.txt :

Step 3: Using the python file to generate fasta file for each sample inserted in samples.txt by putting this command line:

```
python fasta_generator.py samples.txt
```

All the results is stored in “contig_fasta” folder.

Example of input:



Each fasta is name to each sample ID.

Step 4: Go to “contig_fasta” folder and use AMRFinderPlus for each sample:

Example for file 95.fasta:

amrfinder -n 95.fasta -O Staphylococcus_aureus -o amrfinderresult_95.csv

Explanation for the commandline:

‘amrfinder’ : this is indicating that the tool is used for generating the AMR result is AMRFinderPlus

‘-n’ : this is option command indicating the input fasta file is Nucleotide Fasta file

‘95.fasta’ : the input fasta file

‘-O’ : Taxon used for screening known resistance causing point mutations and blacklisting of common, non-informative genes

‘ Staphylococcus_aureus ‘: indicating that only find AMR genes related to *S.Aureus*

‘-o’: output option

‘amrfinderresult_95.csv’ : name of the output file in csv

format.Example of an output file ‘amrfinderresult_95.csv’:

1	Protein identifier	Contig id	Start	Stop	Strand	Gene symbol	Sequence name	Scope	Element type	Elemen
2	NA	contig26	11299	11676	-	blaI	penicillinase repressor BlaI	core	AMR	AMR
3	NA	contig26	11669	13423	-	blaR1	beta-lactam sensor/signal transducer BlaR1	core	AMR	AMR
4	NA	contig26	13530	14372	+	blaZ	penicillin-hydrolyzing class A beta-lactamase BlaZ	core	AMR	AMR
5	NA	contig4	51412	51828	-	fosB	FosB1/FosB3 family fosfomycin resistance bacillithiol transferase	core	AMR	AMR
6	NA	contig9	35566	36915	-	tet(38)	tetracycline efflux MFS transporter Tet(38)	core	AMR	AMR

Note: This is not the whole result of amrfinderresult_95.csv as there are much morecolumns in the csv file.

Output fields in the output format:

Protein Identifier - This is from the FASTA defline for the protein or DNA sequence.

- Contig id - Contig name.
- Start - 1-based coordinate of first nucleotide coding for protein in DNA sequence on contig.
- Stop - 1-based coordinate of last nucleotide coding for protein in DNA sequence on contig. Note that for protein hits (where the Method is HMM or ends in P) the coordinates are takenfrom the GFF, which means that for circular contigs when the protein spans the contig breakthe stop coordinate may be larger than the contig size
- Gene symbol - Gene or gene-family symbol for protein or nucleotide hit. For point mutations it is a combination of the gene symbol and the SNP definition separated by "_"
- Sequence name - Full-text name for the protein, RNA, or point mutation.

- Scope - The AMRFinderPlus database is split into 'core' AMR proteins that are expected to have an effect on resistance and 'plus' proteins of interest added with less stringent inclusioncriteria. These may or may not be expected to have an effect on phenotype.
- Element type - AMRFinder+ genes are placed into functional categories based onpredominant function AMR, STRESS, or VIRULENCE.
- Element subtype - Further elaboration of functional category into (ANTIGEN, BIOCIDE, HEAT, METAL, PORIN) if more specific category is available, otherwise he element is repeated.
- Class - For AMR genes this is the class of drugs that this gene is known to contribute to resistance of.
- Subclass - If more specificity about drugs within the drug class is known it is elaborated here.
- Method - Type of hit found by AMRFinder. A suffix of 'P' or 'X' is appended to "Methods" that could be found by protein or nucleotide.
    - ALLELE - 100% sequence match over 100% of length to a protein named at the allele level in the AMRFinderPlus database.
    - EXACT - 100% sequence match over 100% of length to a protein in the database that is not a named allele.
    - BLAST - BLAST alignment is > 90% of length and > 90% identity to a protein in the AMRFinderPlus database.
    - PARTIAL - BLAST alignment is > 50% of length, but < 90% of length and > 90% identity to the reference, and does not end at a contig boundary.
    - PARTIAL_CONTIG_END - BLAST alignment is > 50% of length, but < 90% of length and > 90% identity to the reference, and the break occurrs at a contig boundary indicating that this gene is more likely to have been split by an assemblyissue.
    - HMM - HMM was hit above the cutoff, but there was not a BLAST hit that met standards for BLAST or PARTIAL. This does not have a suffix because only protein sequences are searched by HMM.
    - INTERNAL_STOP - Translated blast reveals a stop codon that occurred before the end of the protein. This can only be assessed if the `-n <nucleotide_fasta>` option is used.
    - POINT - Point mutation identified by blast.
- Target length - The length of the query protein or gene. The length will be in amino-acids if the reference sequence is a protein, but nucleotide if the reference sequence is nucleotide.
- Reference sequence length - The length of the Reference protein or nucleotide in thedatabase (NA if HMM-only hit).
- % Coverage of reference sequence - % of reference covered by blast hit (NA if HMM-only hit).
- % Identity to reference sequence - % amino-acid identity to reference protein or nucleotideidentity for nucleotide reference. (NA if HMM-only hit).
- Alignment length - Length of BLAST alignment in amino-acids or nucleotides if nucleotide reference (NA if HMM-only hit).
- Accession of closest protein - RefSeq accession for reference hit by BLAST (NA if HMM-only hit). Note that only one reference will be chosen if the blast hit is equidistant from

multiple references. For point mutations the reference is the sensitive "wild-type" allele, andthe element symbol describes the specific mutation.

- Name of closest protein - Full name assigned to the closest reference hit (NA if HMM-onlyhit).
- HMM id - Accession for the HMM, NA if none.
- HMM description - The family name associated with the HMM, NA if none.

Note: full documetation of AMRFinderPlus can be found in: https://github.com/ncbi/amr/wiki/Running-AMRFinderPlus

Final Note: This documentation only documents the process of data preparation process ofCapstone Project Phase 1 and there will be more added into this documentation.

Documentation for setting up PostgresSQL for StaphBook

Recommendation: this installation guide is for Linux Operating System, ideally Ubuntu 20.04 or later

Step 1: Install PostgresSQL

Run this script which is obtained from: https://www.postgresql.org/download/linux/ubuntu/

```
sudo sh -c 'echo "deb http://apt.postgresql.org/pub/repos/apt $(lsb_release -cs)-pgdg main" > /etc/apt/sources.list.d/pgdg.list'

wget --quiet -O - https://www.postgresql.org/media/keys/ACCC4CF8.asc | sudo apt-key add –

sudo apt-get update

sudo apt-get -y install postgresql
```

Step 2: Create a database in postgres SQL

Change the user to postgres:

- sudo su postgres

Open postgresSQL:

- psql

Setting up the password:

- **ALTER USER** postgres PASSWORD 'myPassword' **ALTER ROLE**

Create a database: (recommend name them to Genomes)

- create database Genomes

Check if the database is created:

- \l

```
postgres=# \l
 Genomes   | postgres | UTF8     | en_AU.UTF-8 | en_AU.UTF-8 |
 hello     | postgres | UTF8     | en_AU.UTF-8 | en_AU.UTF-8 |
 postgres  | postgres | UTF8     | en_AU.UTF-8 | en_AU.UTF-8 |
 staphbook | postgres | UTF8     | en_AU.UTF-8 | en_AU.UTF-8 | =Tc/postgres         +
           |          |          |             |             | postgres=CTc/postgres+
           |          |          |             |             | trongdat=CTc/postgres
 template0 | postgres | UTF8     | en_AU.UTF-8 | en_AU.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
 template1 | postgres | UTF8     | en_AU.UTF-8 | en_AU.UTF-8 | =c/postgres          +
           |          |          |             |             | postgres=CTc/postgres
```

Step 3: Inserting data

Download a SQL script using this link:

https://drive.google.com/file/d/1VM7au292sEmkdNo3D6nw3WTH5Qpr7zvM/view?usp=sharing

Note: this is the link given by our client professor “James Hogan” (j.hogan@qut.edu.au). This sql file is large about 10.9gb. If the above link does not work, please contact with our client to discuss about the data

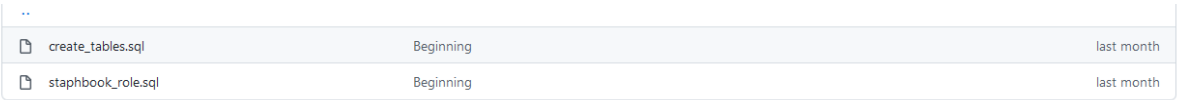
Run the postgresSql in the command line where the sql script is saved in your local machine:

- psql -d {DATABASE_NAME} -a -f {SQL_SCRIPT_NAME}

Example: this is for database “Genomes” and the sql script name is StapBook100.sql

- psql -d Genomes -a -f StapBook100.sql

Run last two sql files which can be found in ./Nodewebsite/sql in the GitHub Repo: https://github.com/NGUYENTRONGDAT123/Webscape-Project



using the similar commandline

- psql -d Genomes -a -f create_tables.sql
- psql -d Genomes -a -f staphbook_role.sql

Step 4: Connecting Database to the application

There is an env template which has explained it which can be found in GitHub repo https://github.com/NGUYENTRONGDAT123/Webscape-Project

but you can follow this if you follow our default installation by creating .env in Nodewebsite folder

```
Nodewebsite > .env
PORT=3001
DB_HOST=localhost
DB_PORT=5432
DB_DB=Genomes
DB_USER=postgres
DB_PASS=password
SECRET="super secret"
```

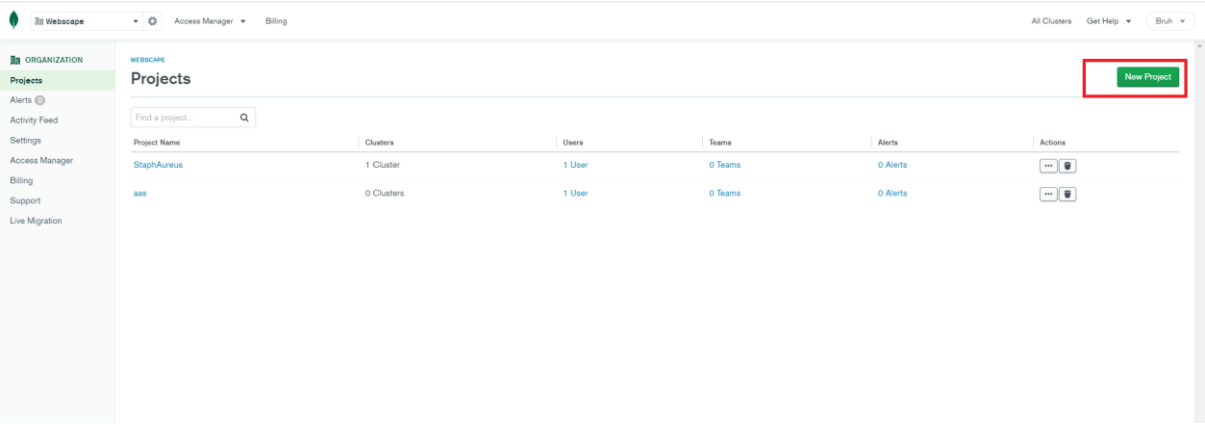
Document for setting MongoDB for AMR features.

MongoDB Atlas is a database on cloud provided by MongoDB. All the prepared data will be stored on cloud and use mongoDB atlas.

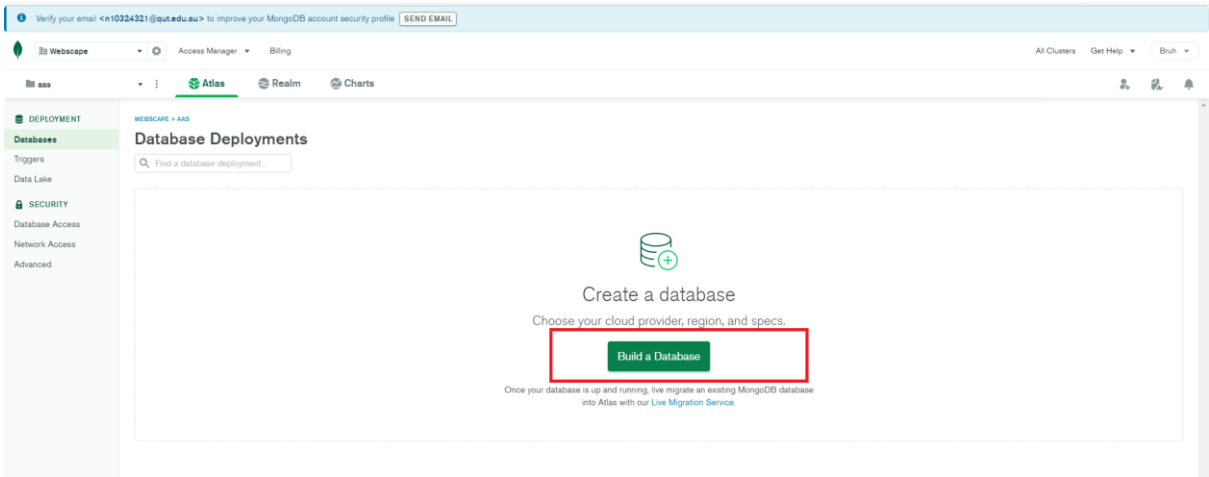
Step 1: Create a MongoDb Atlas using this link https://www.**mongodb**.com/atlas

Note: Create an account using qut account. Note that there is only one free database for one account, which is more than enough for the StaphBook website.

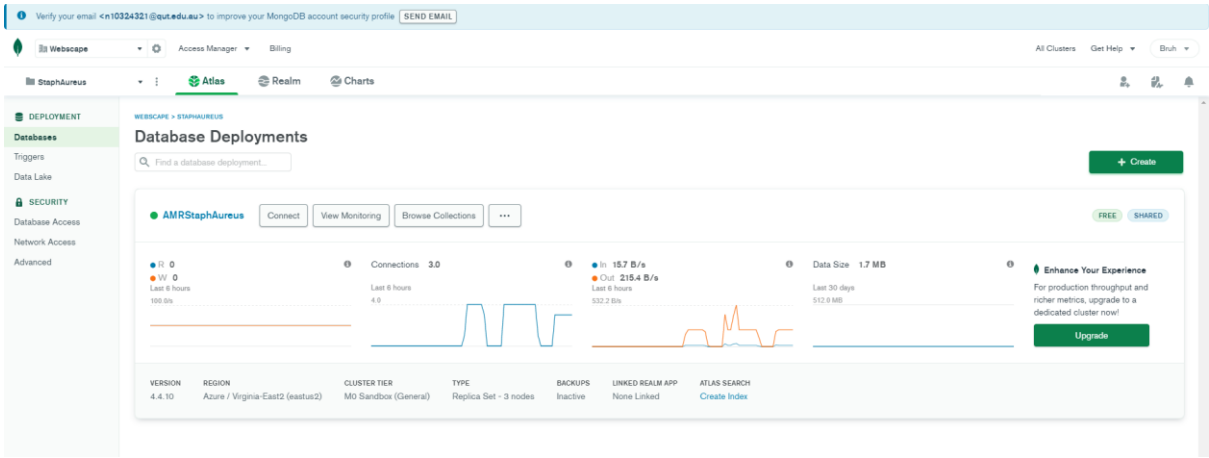
Create a new project



And create a new database:

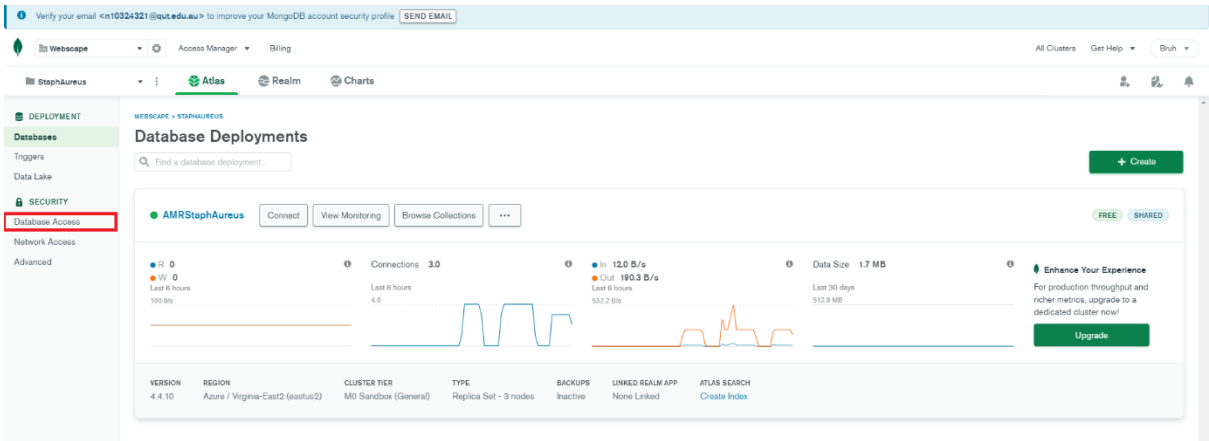


Result would be something like this

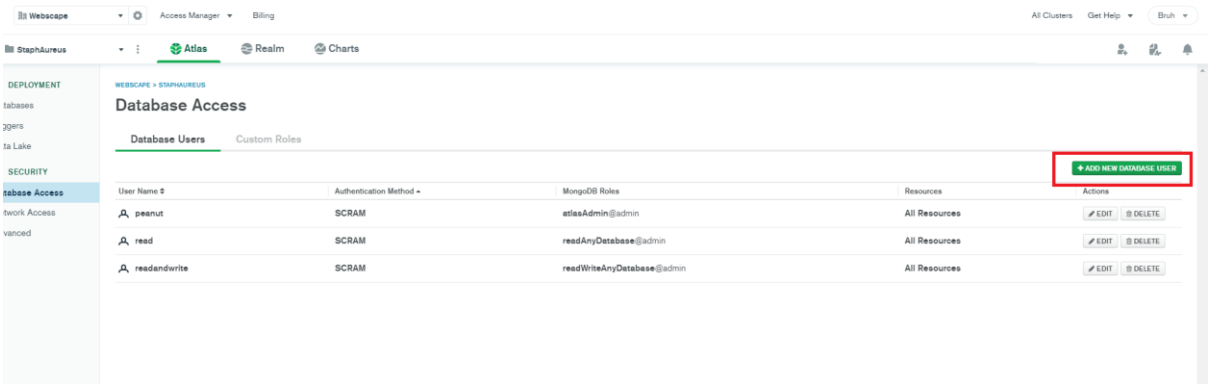


Step 2: Create user to access to the database

Click the Database Access on the left side



Add new Database User and set username and password

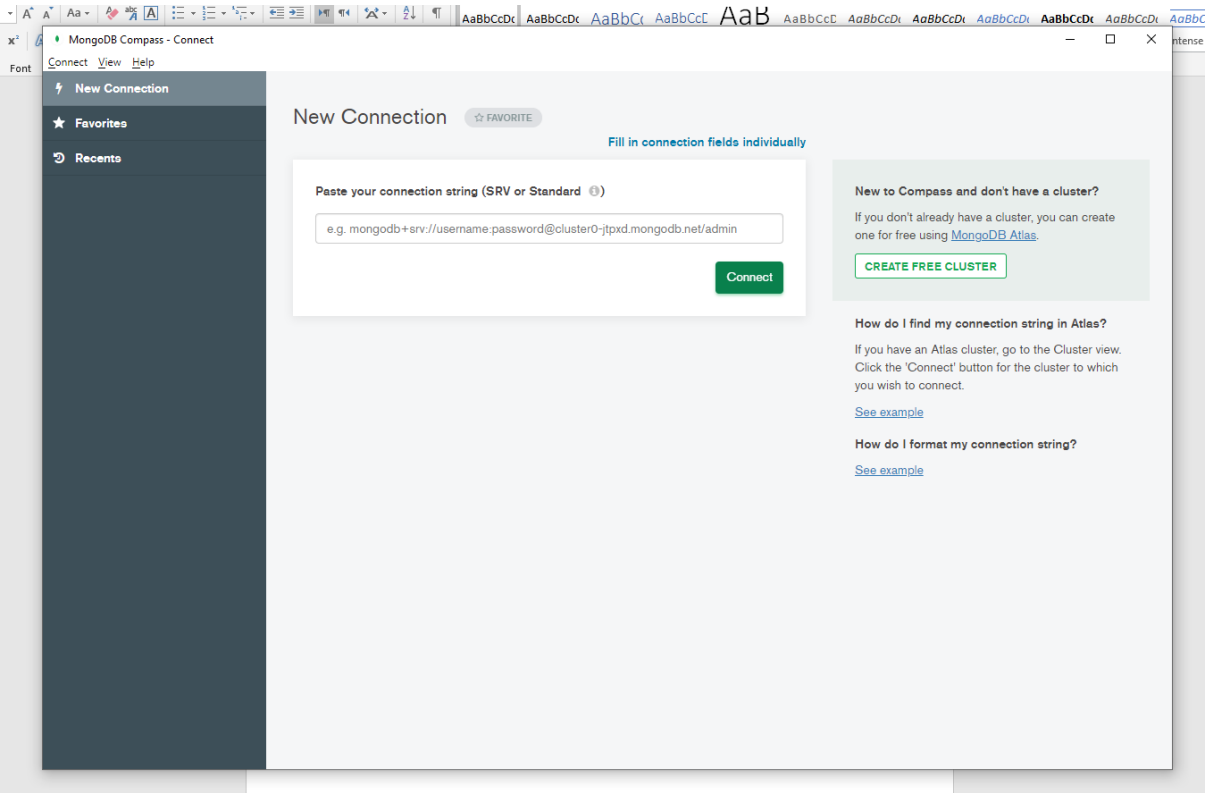


Note: Remember the username and password as it is important for accessing the database

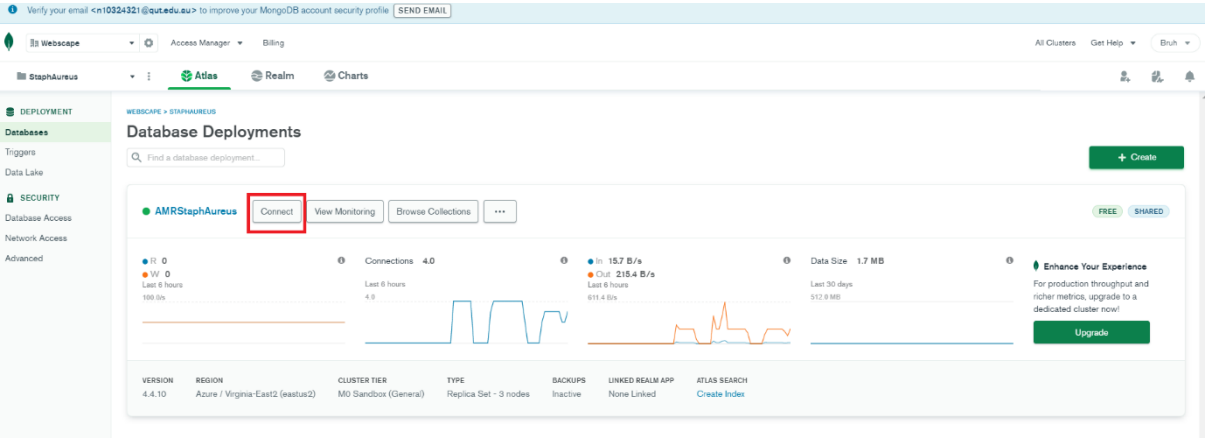
Step 3: Connect to the database

Download MongoDB Compass using this link
https://www.mongodb.com/try/download/compass

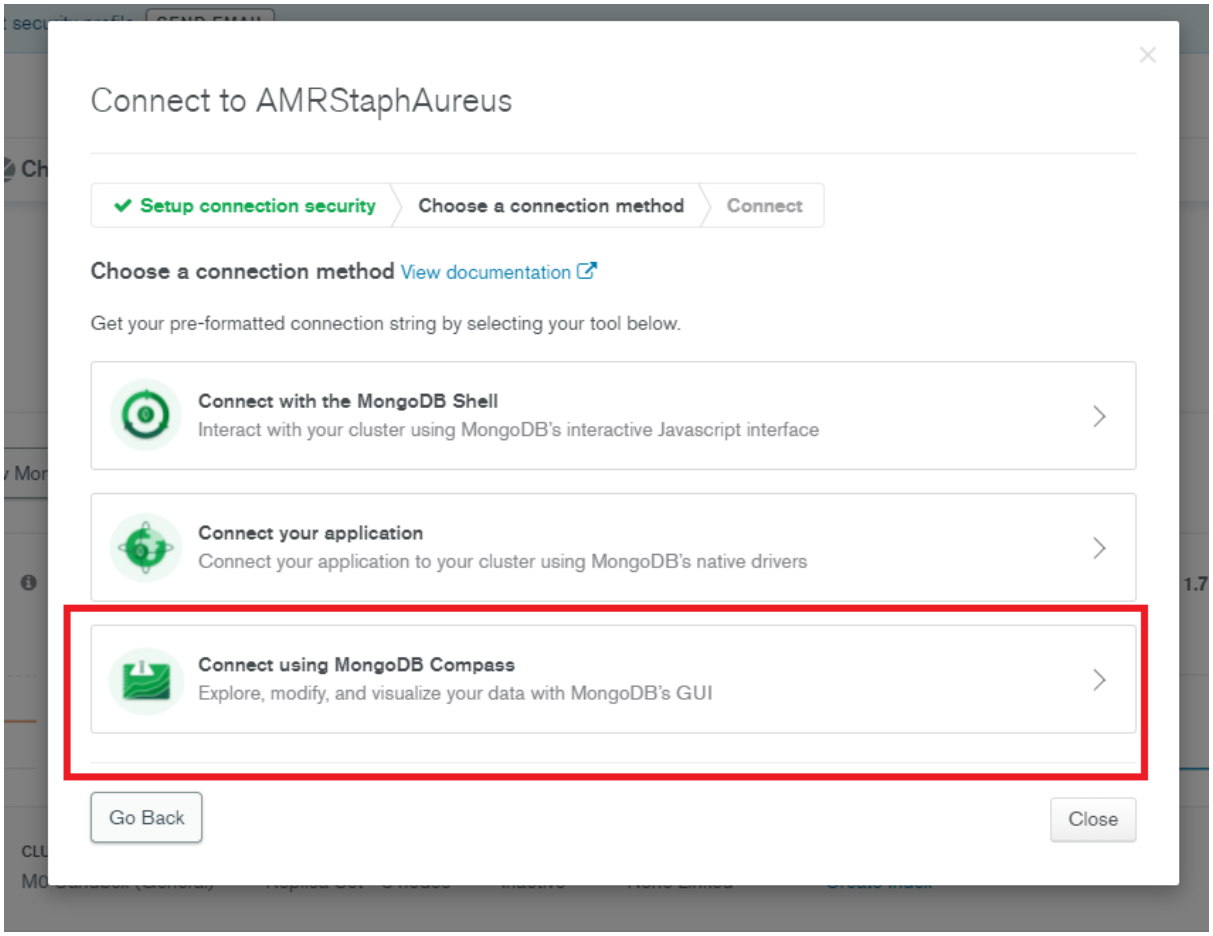
Extract and open it and you should see something like this



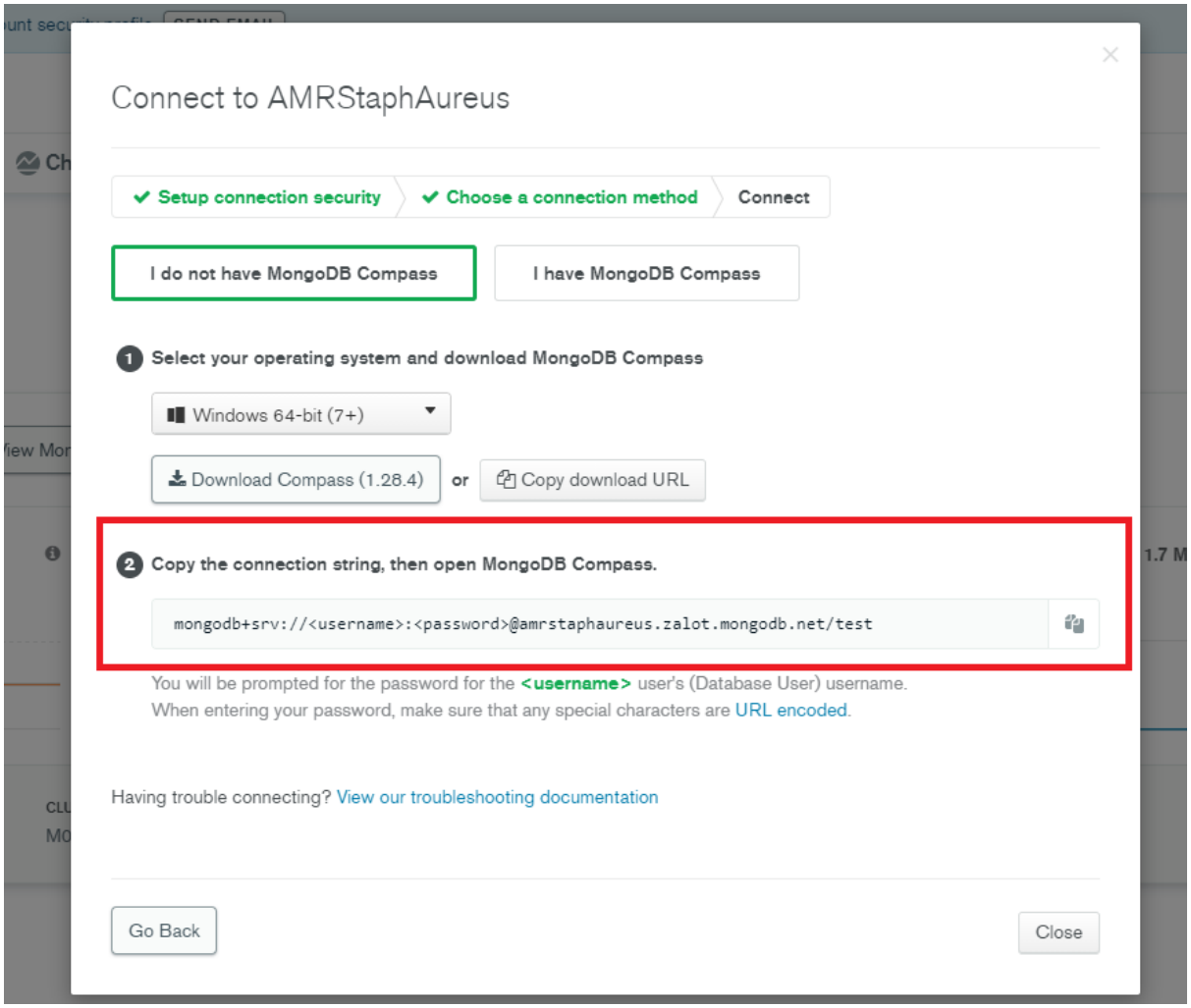
Go back to the website at step and click the Connect button



Next click to connect using MongoDB Compass



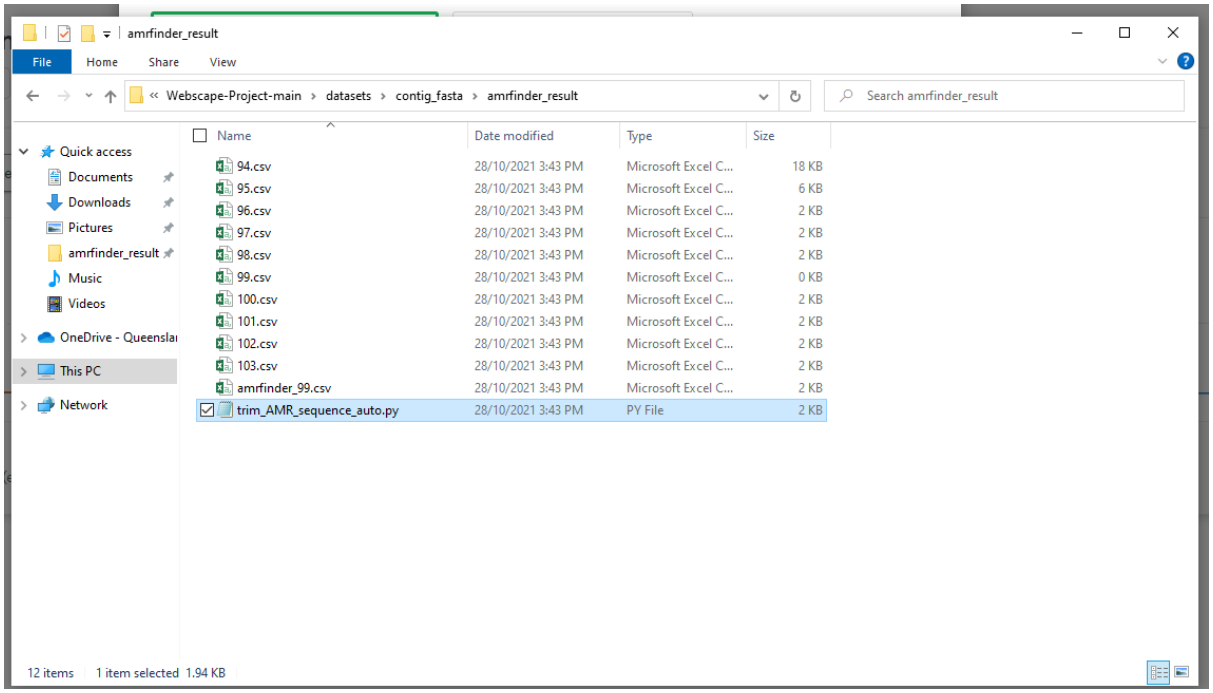
There is a string format that is used for parsing in the compassDB Compass, the username and the password are the users created last step.



Step 4: Inserting the data in the database:

Note: This step can only after you have done prepared data using AMRFinderPlus

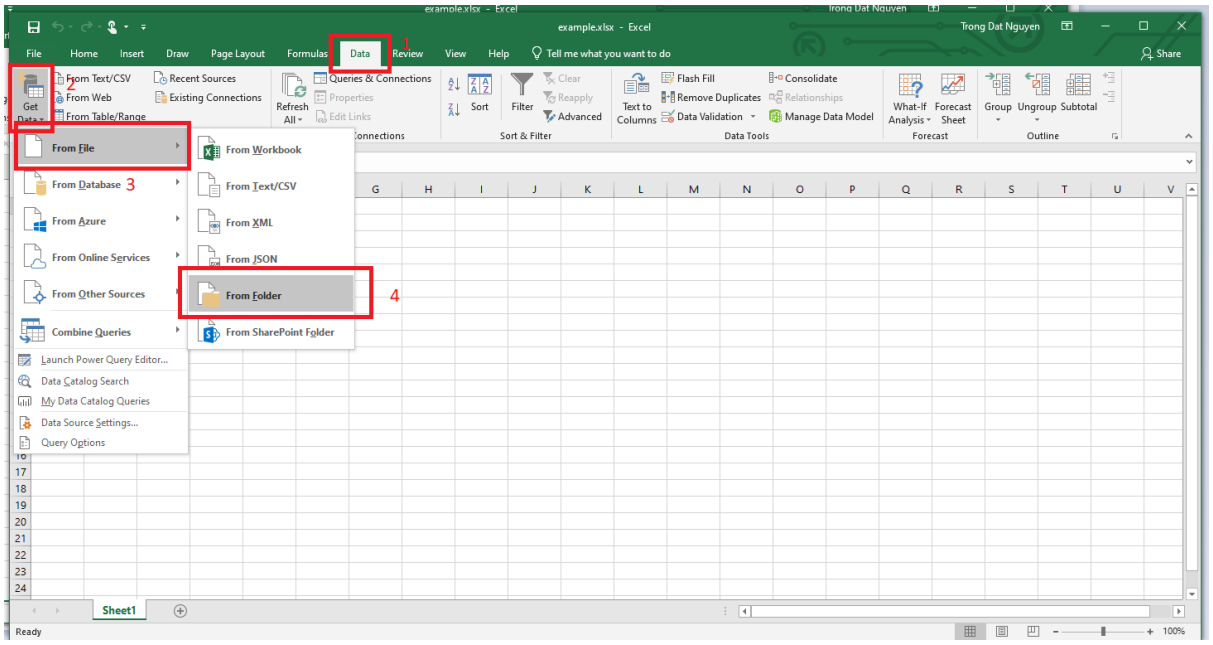
Assume that you have done preparing AMR samples using AMRFinderPlus and you results like this:



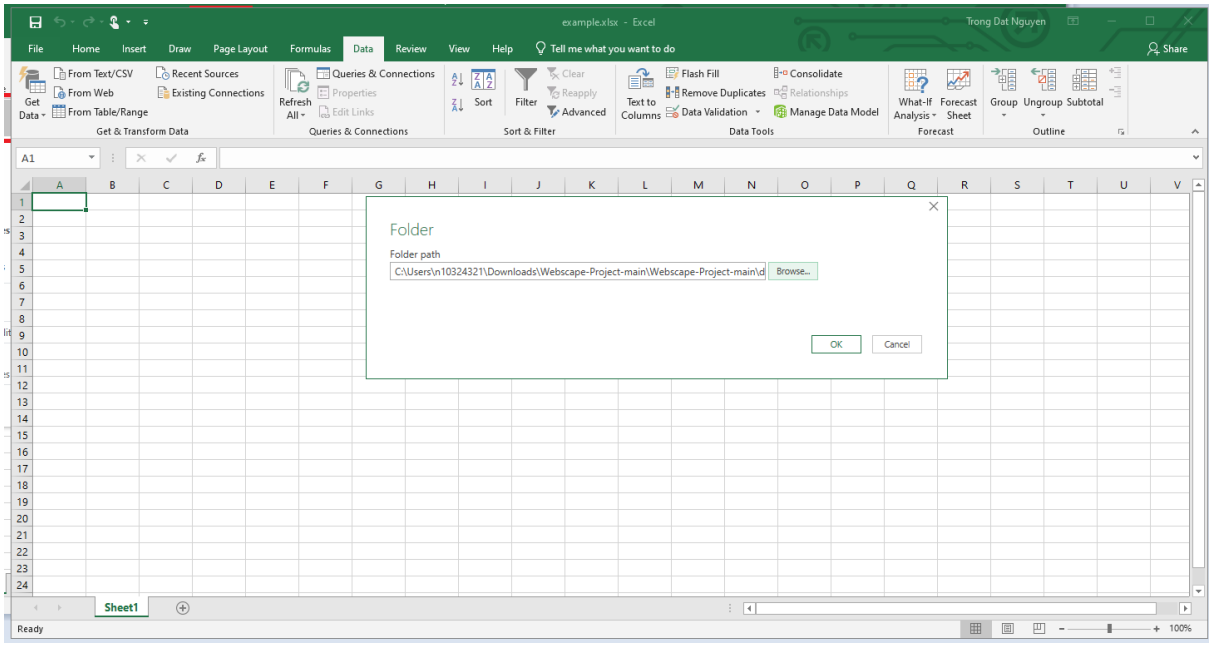
Now you, merge all data using Microsoft Excel

Create an Excel File and open it.

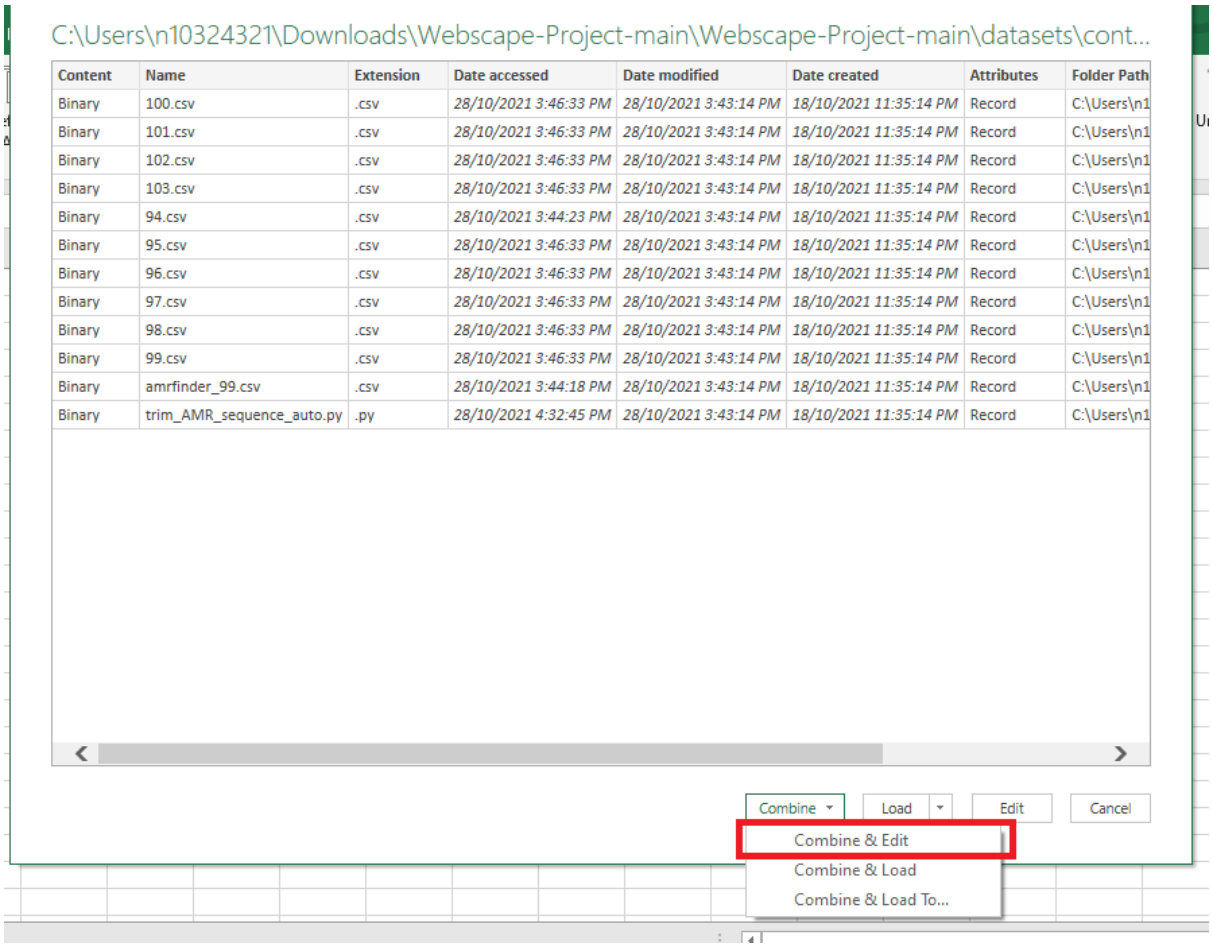
Click the data panel and click the button get Data from the folder



Select the folder where the AMR result is located in your local machine

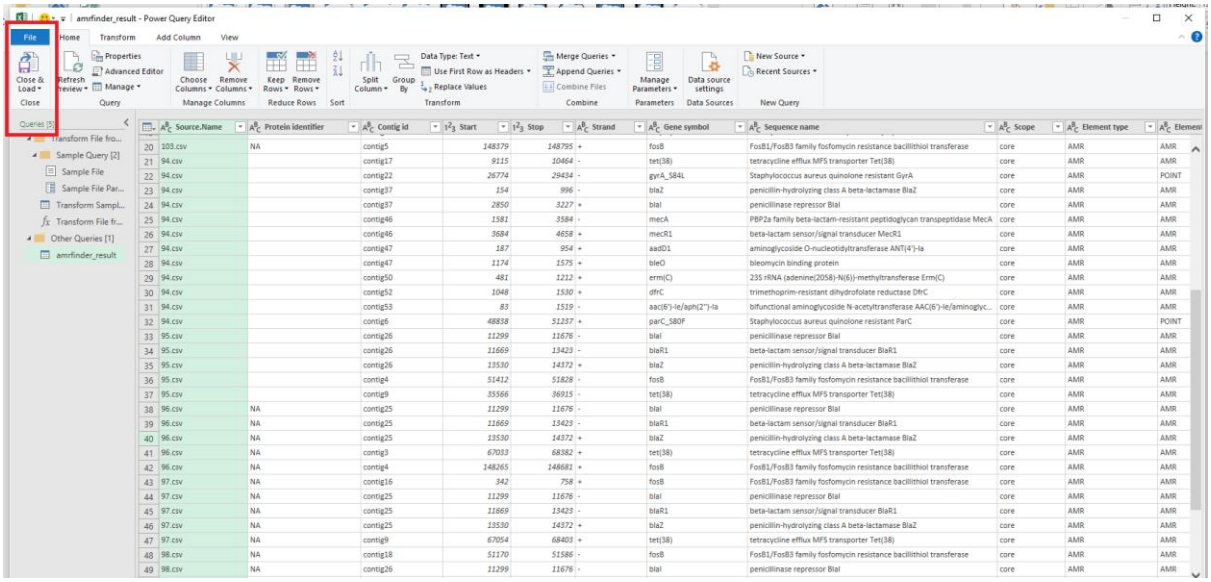


Hit Ok and Select “combine and edit”



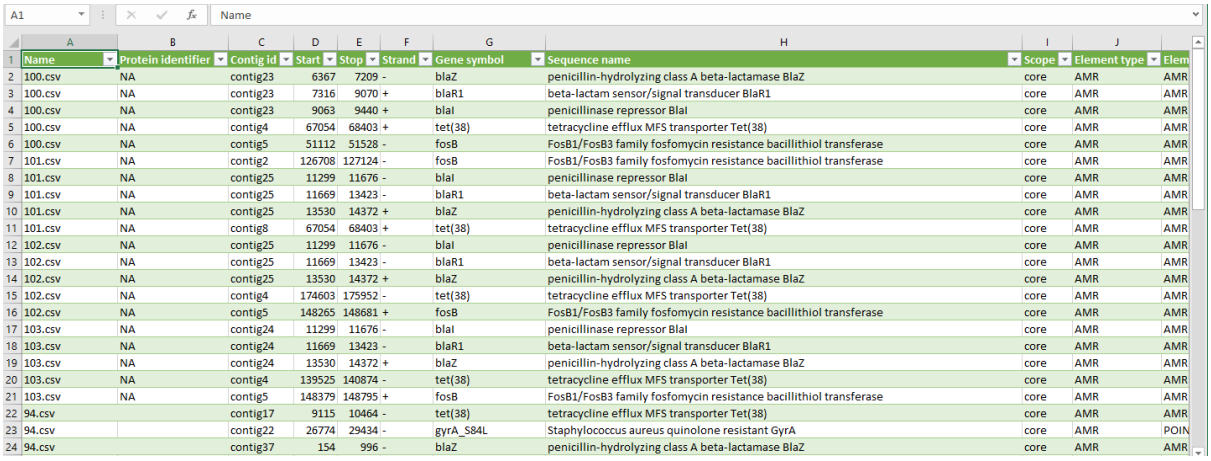
Content	Name	Extension	Date accessed	Date modified	Date created	Attributes	Folder Path
Binary	100.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	101.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	102.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	103.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	94.csv	.csv	28/10/2021 3:44:23 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	95.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	96.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	97.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	98.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	99.csv	.csv	28/10/2021 3:46:33 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	amrfinder_99.csv	.csv	28/10/2021 3:44:18 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1
Binary	trim_AMR_sequence_auto.py	.py	28/10/2021 4:32:45 PM	28/10/2021 3:43:14 PM	18/10/2021 11:35:14 PM	Record	C:\Users\n1

Load the data

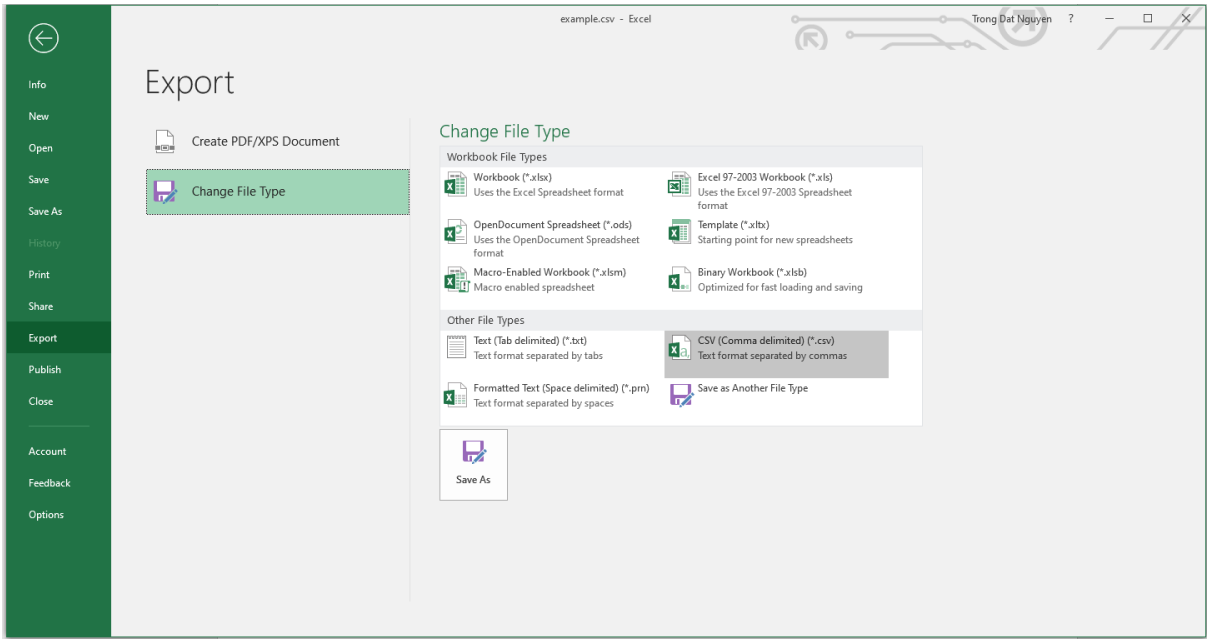


Now you have an excel folder containing all data. You need to export that excel file as csv.

Note: Change Source.Name column to Name

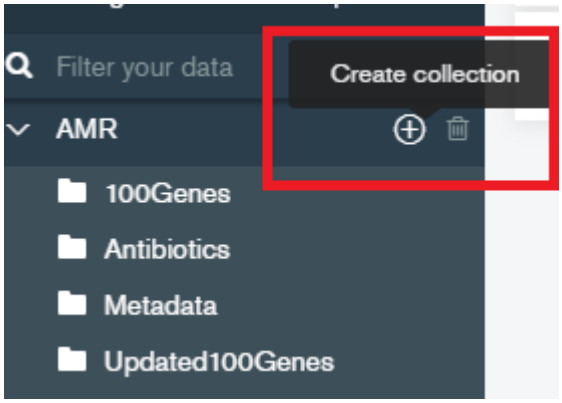


Name	Protein identifier	Contig id	Start	Stop	Strand	Gene symbol	Sequence name	Scope	Element type	Elem
100.csv	NA	contig23	6367	7209	-	blaZ	penicillin-hydrolyzing class A beta-lactamase BlaZ	core	AMR	AMR
100.csv	NA	contig23	7316	9070	+	blaR1	beta-lactam sensor/signal transducer BlaR1	core	AMR	AMR
100.csv	NA	contig23	9063	9440	+	blaI	penicillinase repressor BlaI	core	AMR	AMR
100.csv	NA	contig4	67054	68403	+	tet(38)	tetracycline efflux MFS transporter Tet(38)	core	AMR	AMR
100.csv	NA	contig5	51112	51528	-	fosB	FosB1/FosB3 family fosfomycin resistance bacillithiol transferase	core	AMR	AMR
101.csv	NA	contig2	126708	127124	-	fosB	FosB1/FosB3 family fosfomycin resistance bacillithiol transferase	core	AMR	AMR
101.csv	NA	contig25	11299	11676	-	blaI	penicillinase repressor BlaI	core	AMR	AMR
101.csv	NA	contig25	11669	13423	-	blaR1	beta-lactam sensor/signal transducer BlaR1	core	AMR	AMR
101.csv	NA	contig25	13530	14372	+	blaZ	penicillin-hydrolyzing class A beta-lactamase BlaZ	core	AMR	AMR
101.csv	NA	contig8	67054	68403	+	tet(38)	tetracycline efflux MFS transporter Tet(38)	core	AMR	AMR
102.csv	NA	contig25	11299	11676	-	blaI	penicillinase repressor BlaI	core	AMR	AMR
102.csv	NA	contig25	11669	13423	-	blaR1	beta-lactam sensor/signal transducer BlaR1	core	AMR	AMR
102.csv	NA	contig25	13530	14372	+	blaZ	penicillin-hydrolyzing class A beta-lactamase BlaZ	core	AMR	AMR
102.csv	NA	contig4	174603	175952	-	tet(38)	tetracycline efflux MFS transporter Tet(38)	core	AMR	AMR
102.csv	NA	contig5	148265	148681	+	fosB	FosB1/FosB3 family fosfomycin resistance bacillithiol transferase	core	AMR	AMR
103.csv	NA	contig24	11299	11676	-	blaI	penicillinase repressor BlaI	core	AMR	AMR
103.csv	NA	contig24	11669	13423	-	blaR1	beta-lactam sensor/signal transducer BlaR1	core	AMR	AMR
103.csv	NA	contig24	13530	14372	+	blaZ	penicillin-hydrolyzing class A beta-lactamase BlaZ	core	AMR	AMR
103.csv	NA	contig4	139525	140874	-	tet(38)	tetracycline efflux MFS transporter Tet(38)	core	AMR	AMR
103.csv	NA	contig5	148379	148795	+	fosB	FosB1/FosB3 family fosfomycin resistance bacillithiol transferase	core	AMR	AMR
94.csv		contig17	9115	10464	-	tet(38)	tetracycline efflux MFS transporter Tet(38)	core	AMR	AMR
94.csv		contig22	26774	29434	-	gyrA_S84L	Staphylococcus aureus quinolone resistant GyrA	core	AMR	POIN
94.csv		contig37	154	996	-	blaZ	penicillin-hydrolyzing class A beta-lactamase BlaZ	core	AMR	AMR

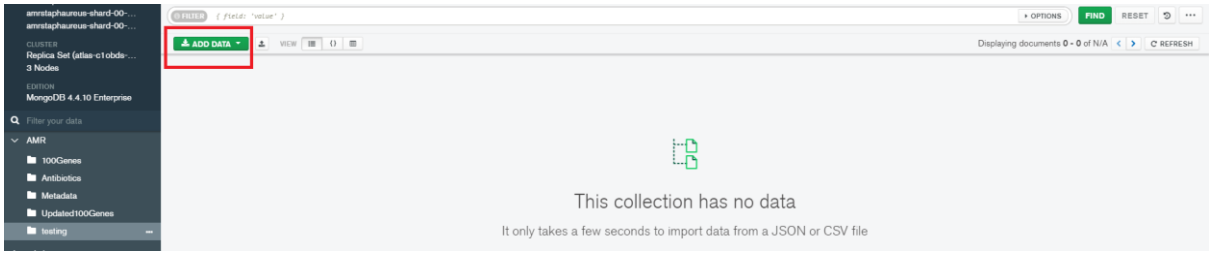


Go back to Mongo Compass and connect to the database in the last step

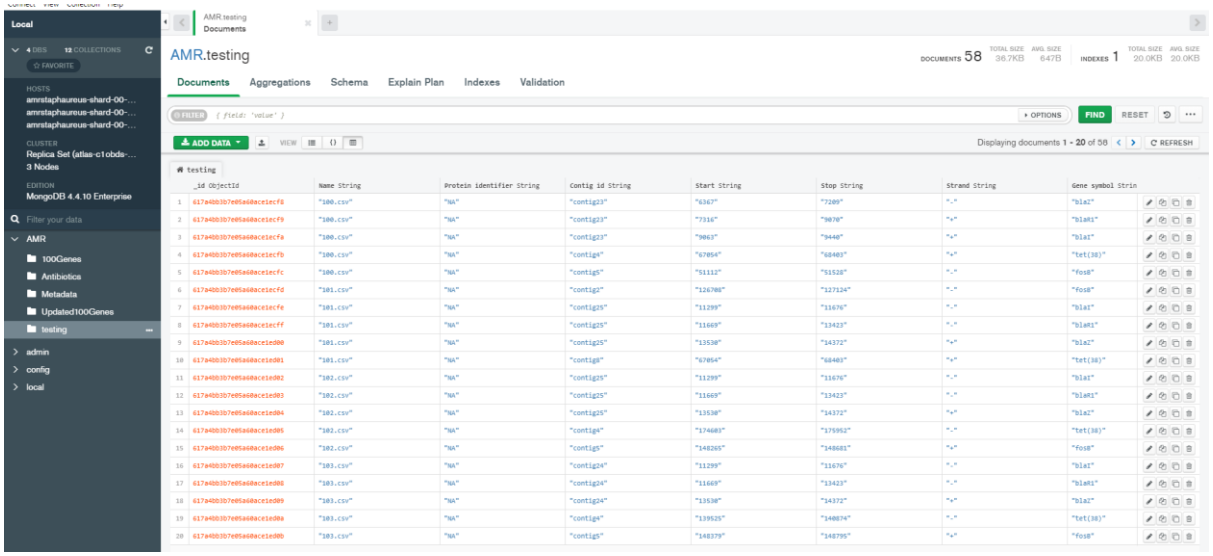
Create a collection and name the collection



Add the data



Select the csv file which was created using Excel and you get the result like this



Those are the steps for inserting the AMR result into the database on Mongo Atlas. This is crucial if the future team wants to add more data to the database

For the purpose of testing the application, these are the collections with these name in order to test it:

- 100Genes: This collection contains AMR genes of 100 samples
- Antibiotics: This collection contains the information of Anti Biotics
- Metadata: This collection contains metadata which is retrieved from Staphopia API and stored them in MongoDB

The Database name that stores 3 collections is named “AMR”

There are 3 json files will be given in GitHub repo:
https://github.com/NGUYENTRONGDAT123/Webscape-Project

which is located in datasets folder

main | Webscape-Project / datasets / | Go to file

NGUYENTRONGDAT123 adding json files ... 2 minutes ago History

..		
contig_fasta	Updated Python file	3 months ago
100Genes	adding json files	2 minutes ago
Antibiotics	adding json files	2 minutes ago
DataPreparationDocs.pdf	Add files via upload	5 months ago
Metadata	adding json files	2 minutes ago
fasta_generator.py	Revert "Revert "Prepare Fasta files""	4 months ago
fasta_generator_auto.py	Revert "Revert "Prepare Fasta files""	4 months ago
samples.txt	datasets	6 months ago
trim_AMR_sequence.py	Updated Python file	3 months ago

Each json files is for each correlated collections' name.

Documentation for installing Redis

Depends on local machines' operating system, installing redis would be different

For Windows: https://github.com/microsoftarchive/redis/releases

Jul 02, 2016
enricogior
win-3.0.504
10a978f
Compare

3.0.504 Latest

This is a critical bug fix release for Redis on Windows 3.0.
If you are running a previous version of 3.0 in a cluster configuration you should upgrade to 3.0.504 urgently.
The fix resolves a problem with the cluster fail-over procedure.

This released is based on antirez/redis 3.0.5 plus Windows-specific fixes.

See the release notes for details.

▾ Assets 4

Redis-x64-3.0.504.msi	6.42 MB
Redis-x64-3.0.504.zip	5.6 MB
Source code (zip)	
Source code (tar.gz)	

15 2 19 2 10

Download the msi file, open and install it

For Ubuntu, run this command line:

```
sudo apt install redis-server
```

Documentation for Running Application

Step 0: Install NodeJS

Step 1: get the repo from: https://github.com/NGUYENTRONGDAT123/Webscape-Project/tree/main/datasets

Step 2: Go to react-client folder

Run this command to install dependencies

- npm install

Run this command to build static page

- npm run build

Step 3: Go to Nodewebsite folder

Run this command to install dependencies

- npm install

Configure .env file

Note: There is .env template on the github repo but you can based on the code here for reference

```
Nodewebsite > .env
PORT=3001
DB_HOST=localhost
DB_PORT=5432
DB_DB=Genomes
DB_USER=postgres
DB_PASS=password
SECRET="super secret"
MONGODB=mongodb+srv://readandwrite:capstone123@amrstaphaureus.zalot.mongodb.net/test
```

And then run the application:

- npm start

Wait for the initialization and go to localhost:PORT (PORT is the port number in .env file)

Note: Application will not run if the databases are not setup correctly

If you have problems, please contact with a previous member of the project datnguyentrong98@gmail.com

Happy Developing Future Developers!