Introduction to

Hands-On Workshop

Part 1 - Atlas

Overview

This hands-on workshop is designed to get you familiar with all aspects of MongoDB, from deploying a cluster, to loading and analyzing data, and finally to creating services to access that data.

This workshop includes 11 lab exercises and several more optional lab exercises you can try as time allows. Don't worry about completing all optional lab exercises in this sitting. The free environment you create in this lab will be yours forever.

Prerequisites

To successfully complete this workshop:

- Privileges to install software on your computer. We will be installing MongoDB Compass in this workshop.

Hands-on Labs

Lab 1 - Create the Cluster

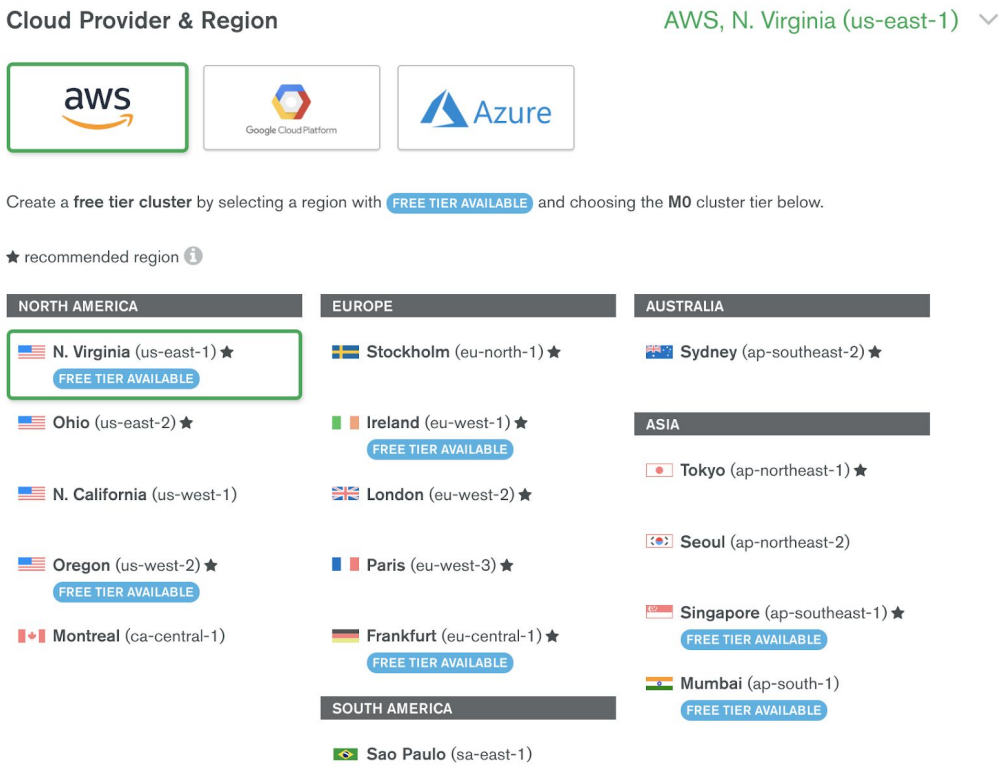
Create an Account or Log In to Atas

We'll be using MongoDB Atlas, our fully managed MongoDB-as-a-service, for this workshop. Go to https://cloud.mongodb.com and either create a new account or log into an existing account you may have previously created.

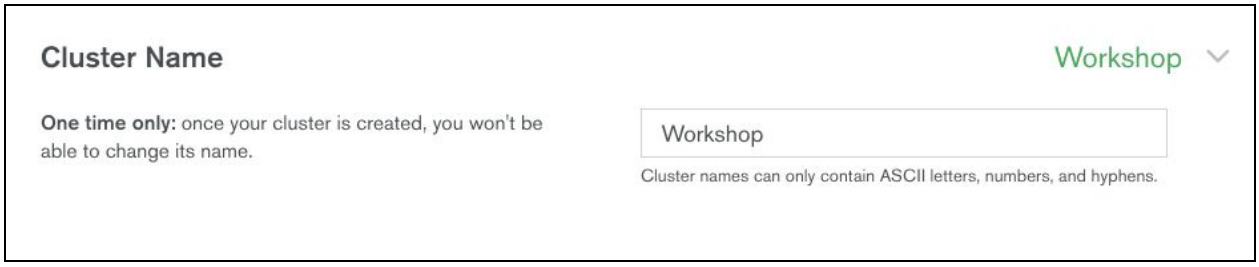
Create a Free Tier Cluster

Click **Build a Cluster**:

Take a moment to browse the options (Provider & Region, Cluster Tier, Version, Backup, …). For our workshop, you can select **ANY** Cloud Provider but let's select AWS for consistency.



and set the Cluster Name to **Workshop**:



The remaining defaults will suffice.

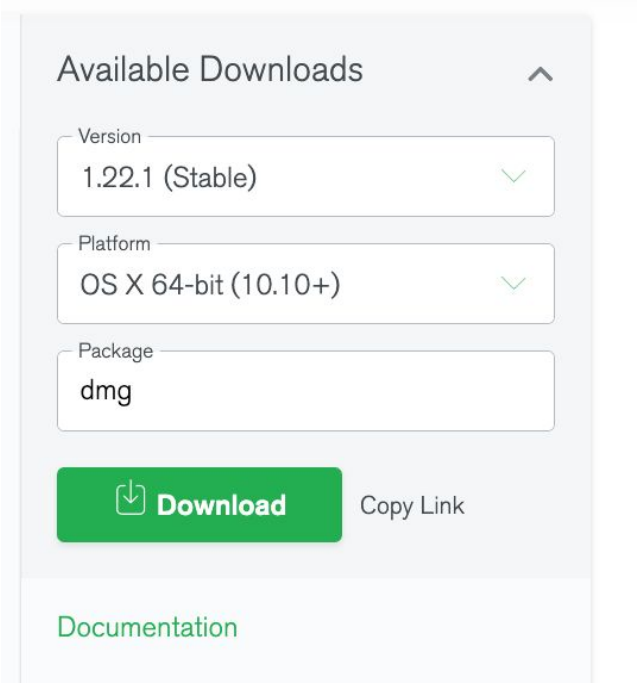
Click **Create Cluster**:

Continue to Lab 2 while the cluster is provisioning.

Lab 2 - Connect to the Cluster

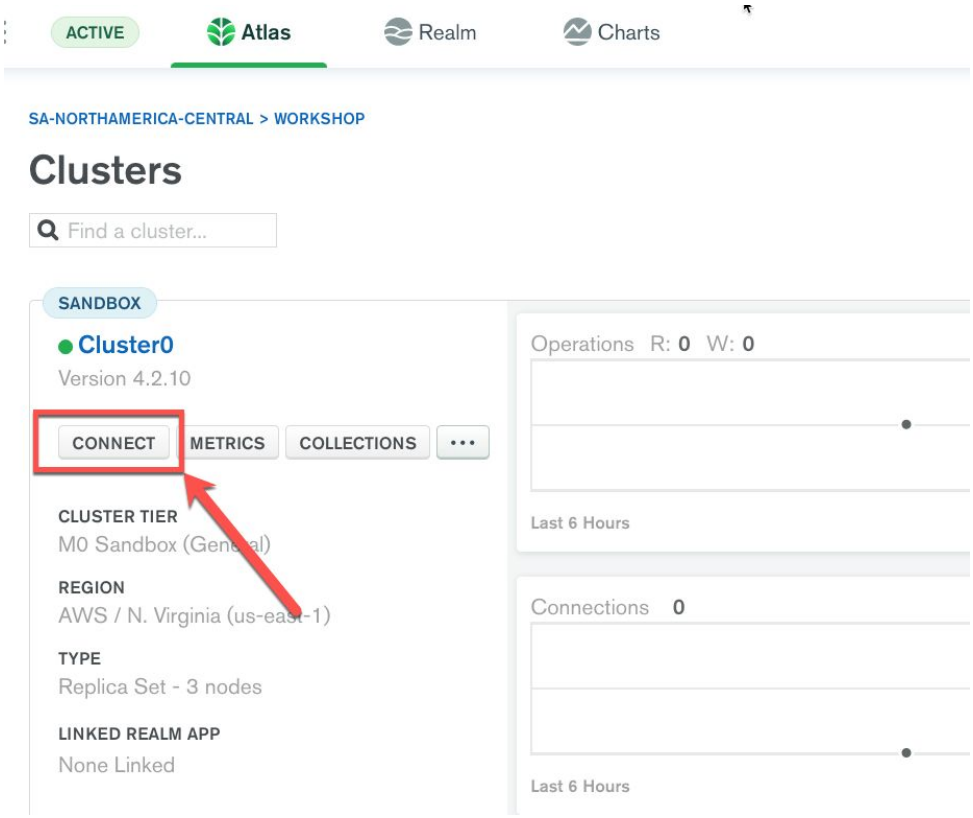
2.A. Install Compass

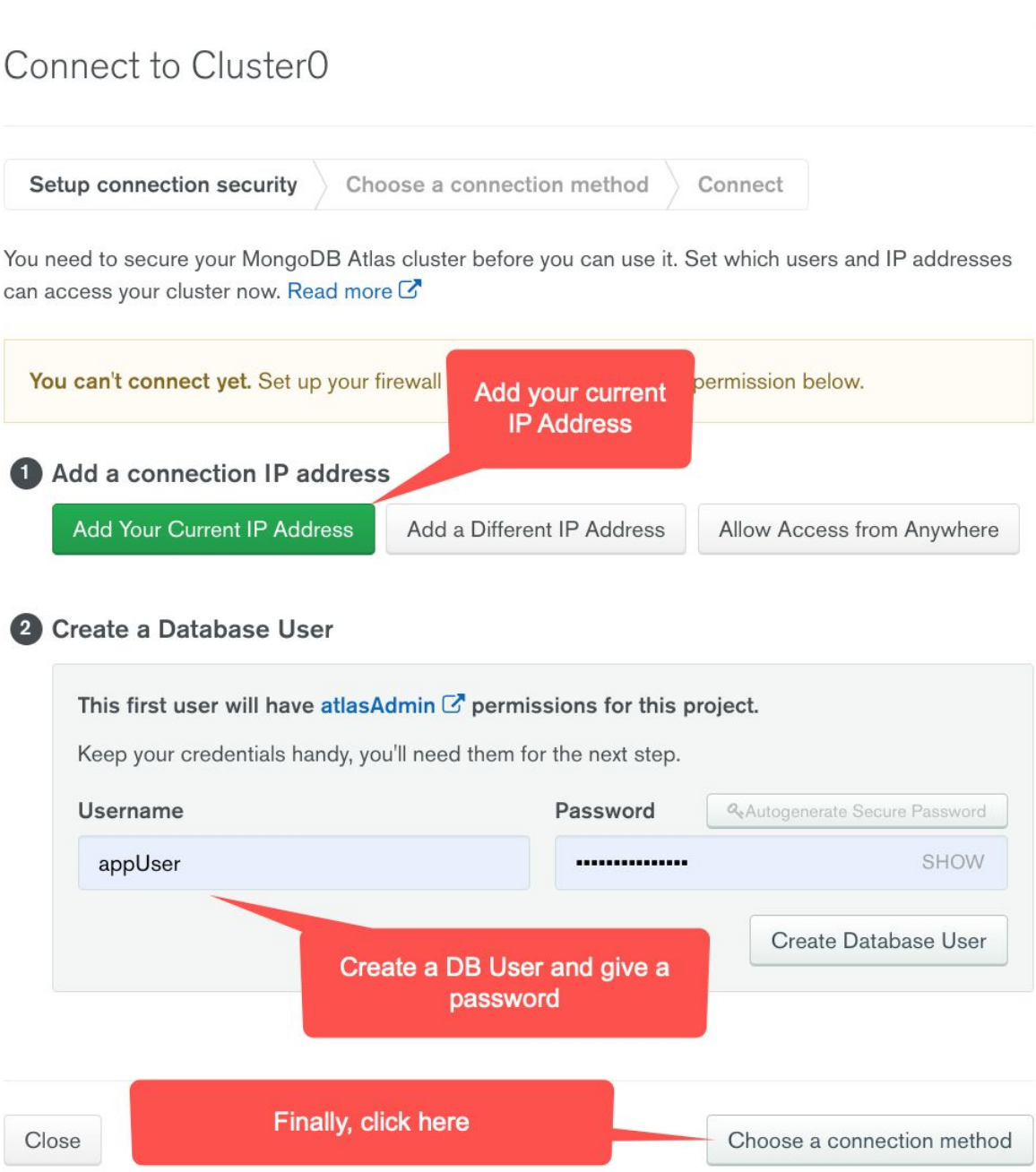
Compass is the GUI for MongoDB. Go to https://www.mongodb.com/download-center/compass to download and install Compass for your platform. Note, there are several editions of Compass. Make sure you download the “Stable” edition:



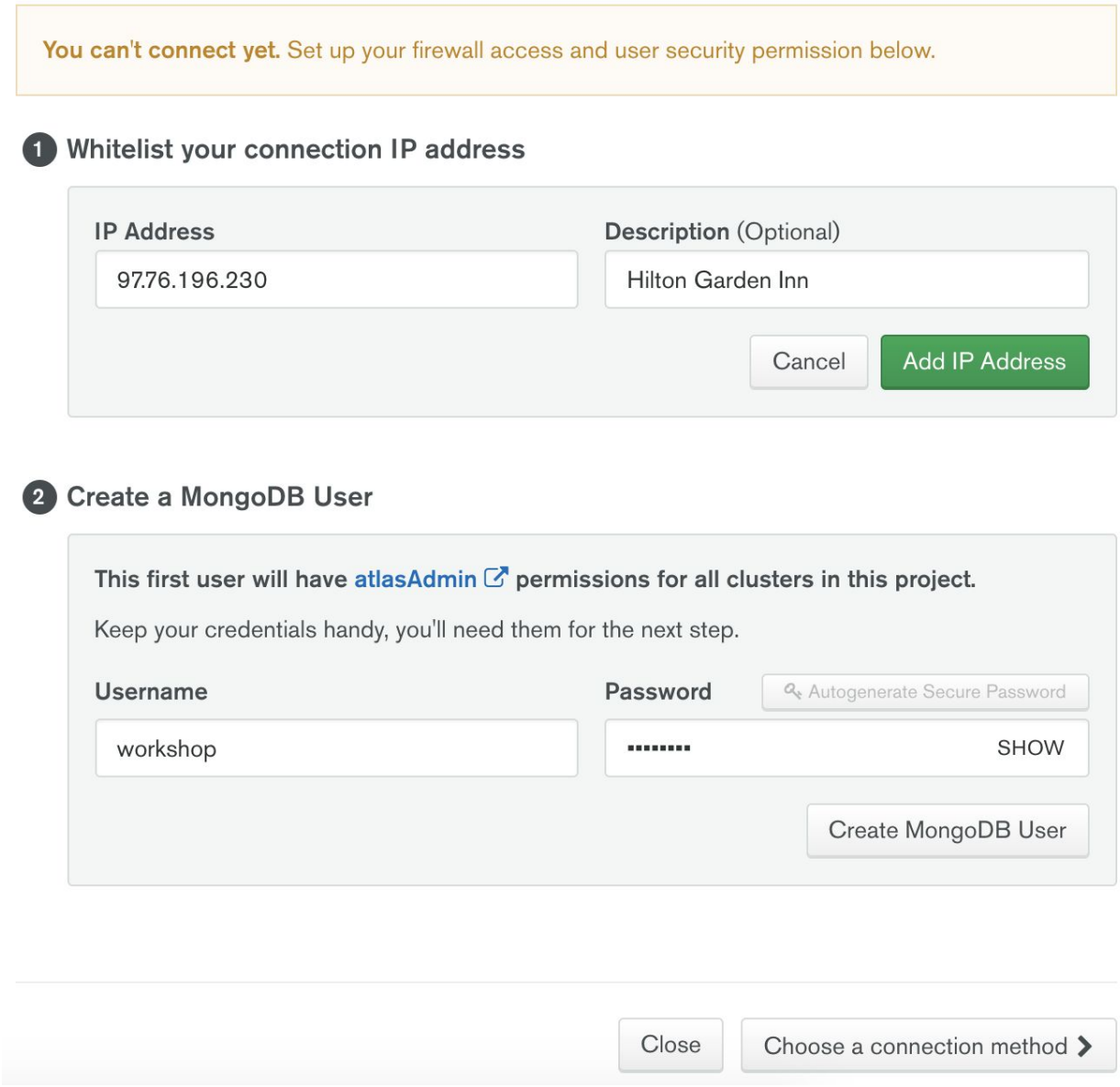
2.B. Setup Connection Security in Atlas

Return to the Atlas UI. Your cluster should now be provisioned. Click the **CONNECT** button, which will prompt you to set up connection security:



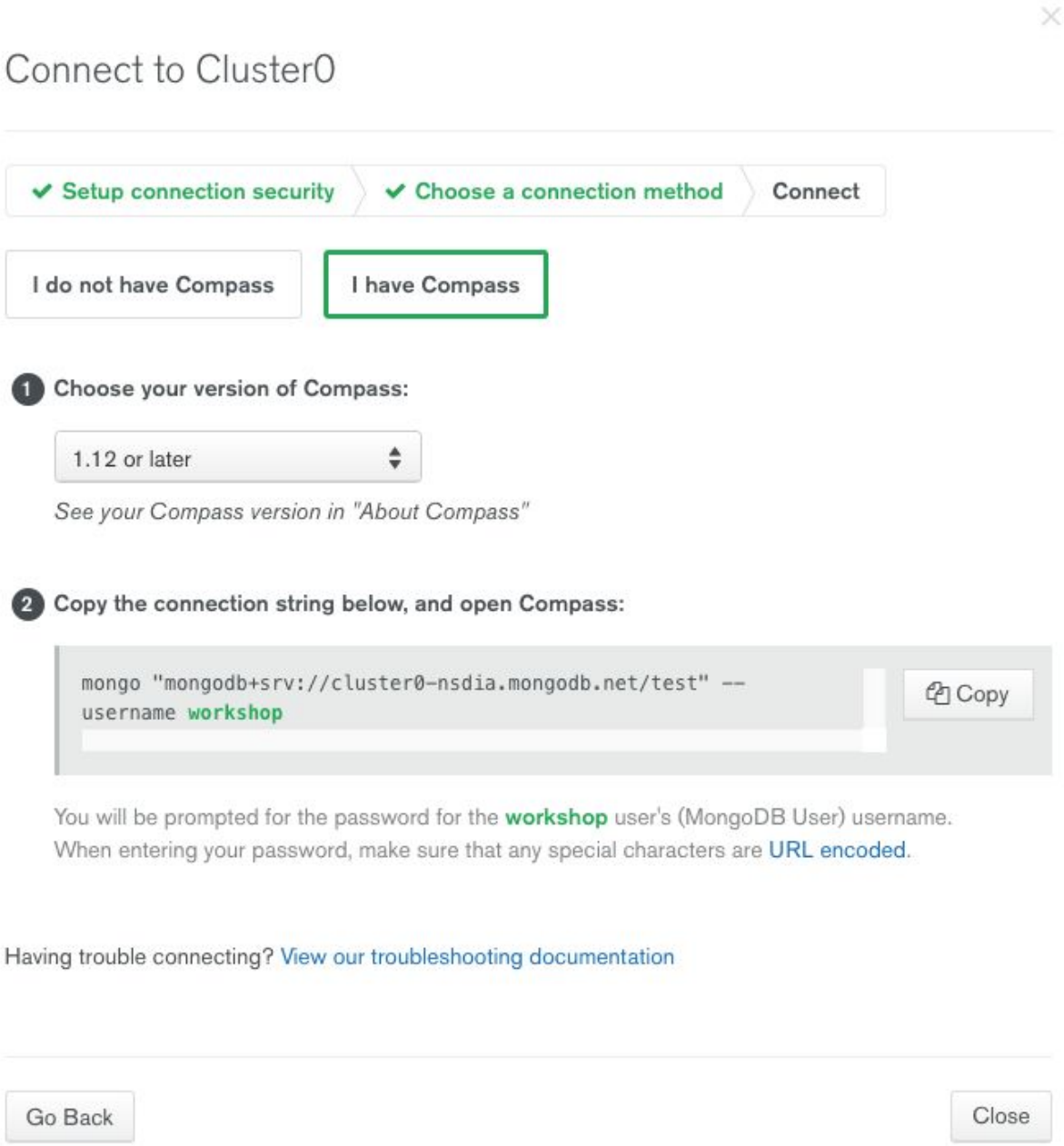


Add Your Current IP Address and **Create a MongoDB User.** I'm using Username **appUser** and password **workshop**:



Click **Choose a connection method** and select **I have Compass**.

Where you choose your version of Compass, select **1.12 or later** and **COPY** the connection string presented:



Connect Compass

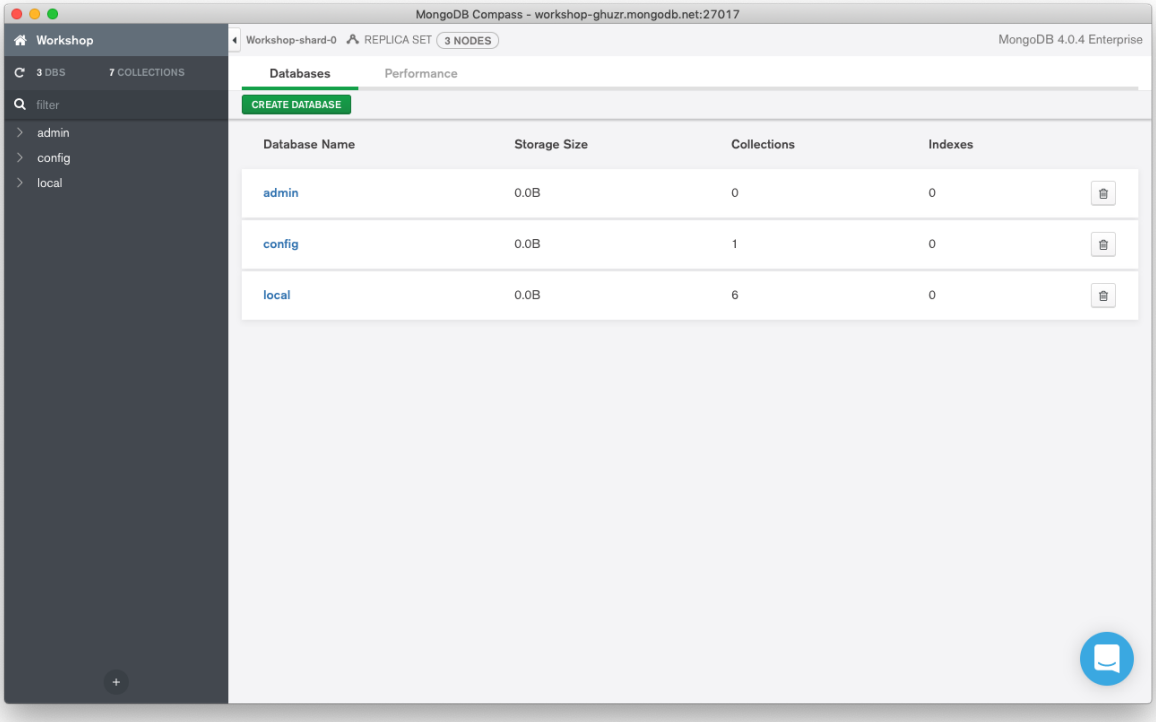
Start Compass and it should detect the connection string in your copy buffer:

Select **Yes**.

Make sure that the **SRV Record** is selected and the authentication is set to username/password. Provide the password (workshop) and *before clicking CONNECT*, **CREATE** a **FAVORITE** named **Workshop**. This will allow us to quickly connect to the cluster in the future.

Click **CONNECT**.

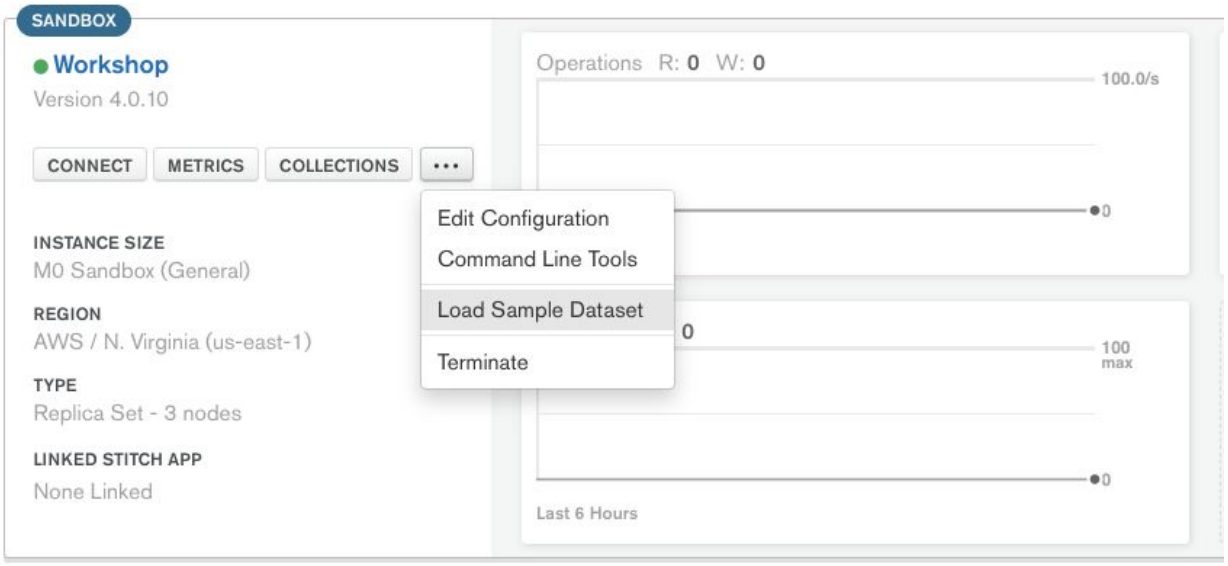
If successful, you'll see some internal databases used by MongoDB:

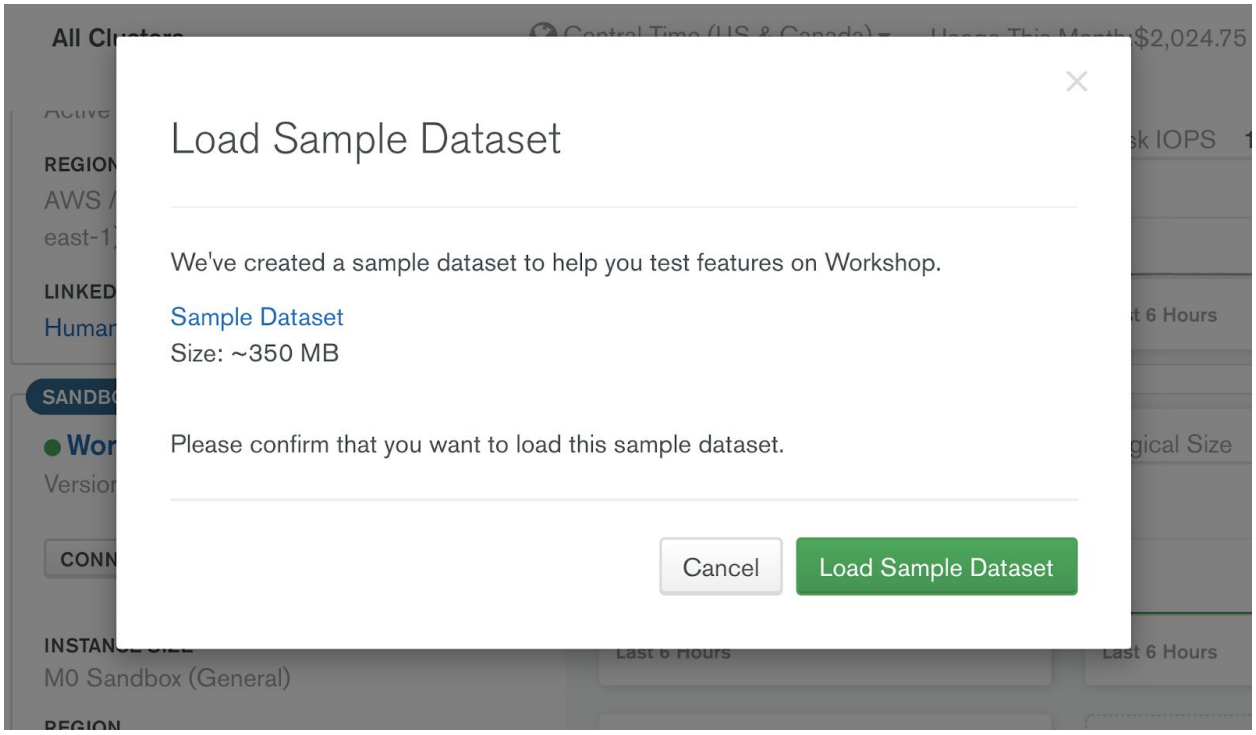


Lab 3 - Load Data

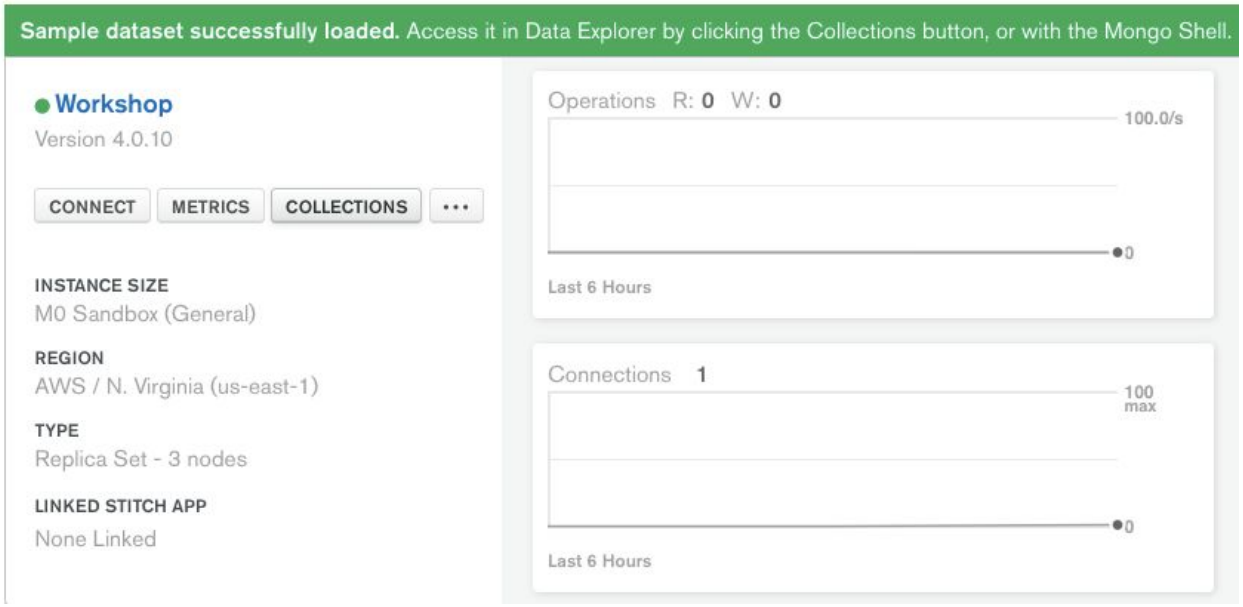
For your convenience, Atlas offers a new “Load Sample Data” feature to allow you to load and explore six different datasets (350MB) with a click of a button.

Return to your Atlas UI, click on the ellipsis … button to Load Sample Dataset.





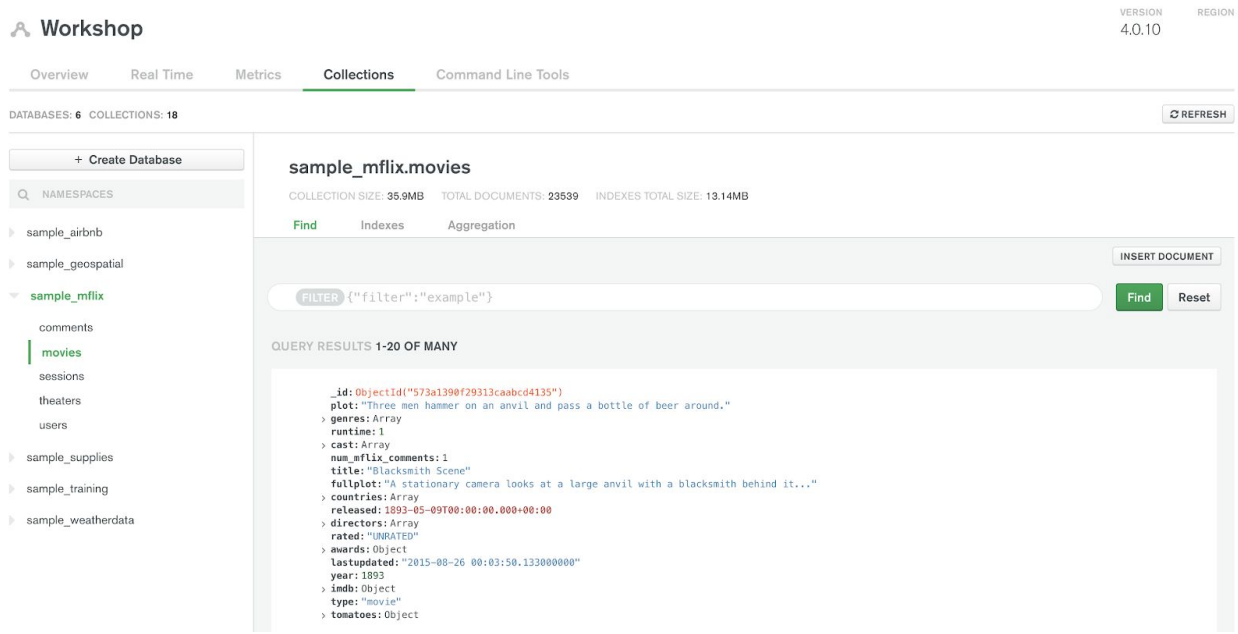
When the loading is done, you can click on the Collections tab inside the Atlas UI to show 6 different databases of 19 different collections.



- sample_airbnb has one collection of AirBnB reviews and listings, already indexed by property type, room type, and bed, name and location.
- sample_geospatial is a collection of shipwrecks and their locations, indexed for searching.
- sample_mflix is a database with five collections all about movies, movie theatres and the metadata like users, comments and sessions. Get visualizing that data with a MongoDB Charts tutorial which already uses the mflix dataset.
- sample_supplies is a typical sales data collection from a mock office supplies company. There's a MongoDB Charts tutorial which shows how to visualize it.
- sample_training is a database with nine collections used in MongoDB's private training. It's based on a range of well known data sources such as Openflights, NYC's OpenData and Twitter's Decahose.
- sample_weatherdata is another collection loaded with geodata, this time for the locations of weather reports on temperature, wind and visibility.

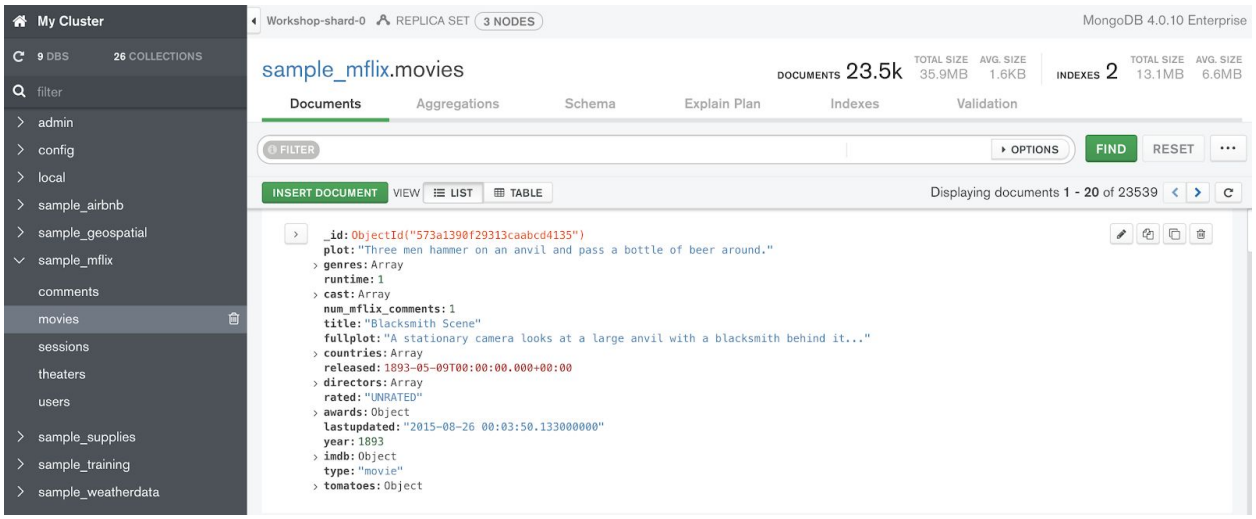
(Feel free to explore these datasets and the various accompanying tutorials.)

For this workshop, we will use the movies collection in the sample_mflix database. This collection contains details on movies. Each document contains a single movie, and information such as its title, release year and cast.



Lab 4 - Browse the Documents in Compass

We will return to Compass to browse and analyze the sample_mflix.movies collection. In Compass, select the **Documents** tab if it is not already selected. You will see this collection has 23539 movie documents.



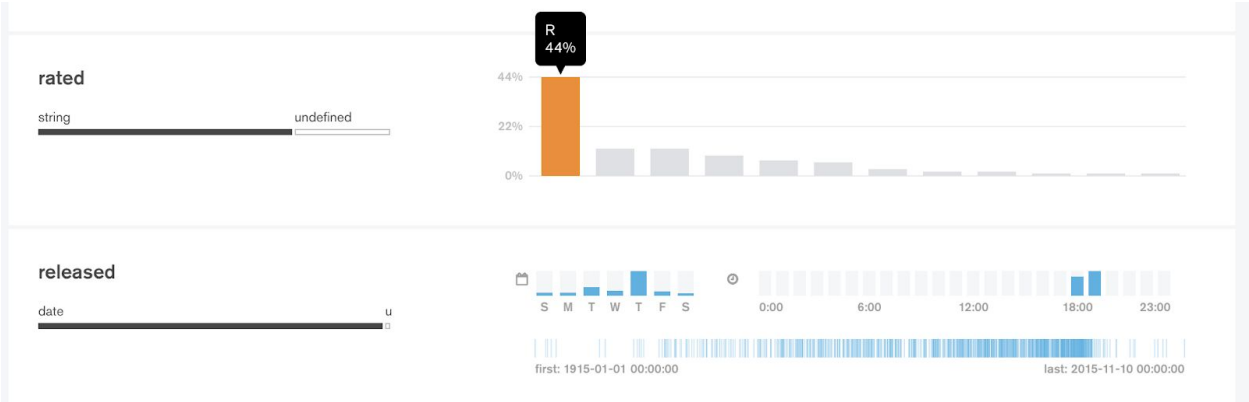
Examine the documents in the collection. Notice how each document has several fields such as **_id**, **title**, and **plot**. The movie documents also have nested subdocuments (**awards, imdb, tomatoes**) and an arrays (**cast, countries, directors**). In a relational database, these fields would most likely be separate tables, but MongoDB allows us to embed this information. Working with data in this natural way is much **easier** than decomposing and composing from relational tables.

Lab 5 - View the Schema

You might be thinking, “Wait, I thought MongoDB is a NoSQL database, and hence, didn’t have a schema?” While that’s technically true, no dataset would be of any use without a schema. So while MongoDB doesn’t enforce a schema, your collections of documents will still always have one. The key difference with MongoDB is that the schema can be **flexible**.

Continuing to work in the **movies** collection, select the **Schema** tab and click **Analyze Schema.** Compass will sample the documents in the collection to derive a schema. In addition to providing field names and types, Compass will also provide a summary of the data values.

For example, for the **rated** field, we can see that **R** is the most common type at 44% and most movies are released on a Thursday (your results may differ slightly based on the sample that was taken) and :



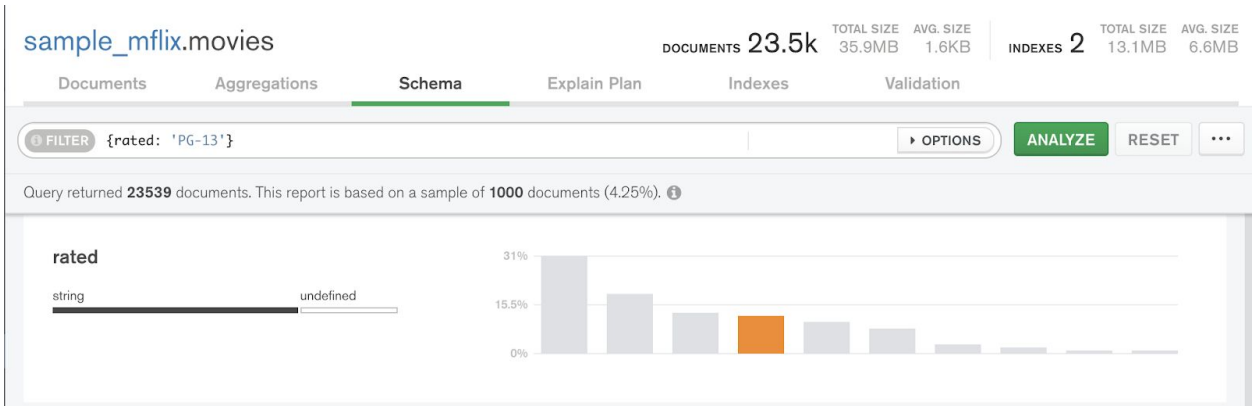
Lab 6 - Query the Data

The MongoDB Query Language (MQL) is based on JSON and provides rich query features for fast, flexible, and highly scalable data. Although you won’t become an expert in every aspect of the MongoDB Query Language today, we will work through a few fun and basic examples to get you started with your MongoDB data in your application.

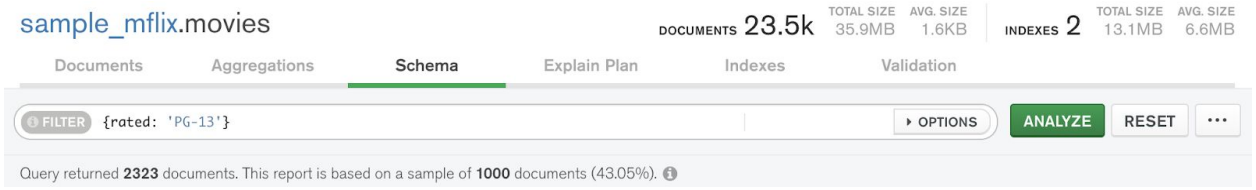
The Schema Analyzer in Compass provides an easy way to learn the language.

6.A. Basic Finds:

For example, when you select the **PG-13** from the **rated** field and notice as you made the selection, the FILTER field at the top of the window gets populated with **{rated: 'PG-13'}**.



Click the **ANALYZE** button to filter for PG-13, of which there are 2323:



To query from the MongoDB shell or your application, MongoDB provides the following methods to read documents from a collection:

db.collection.find()

So in order to find these 2323 MongoDB documents from your movies collection from your application, you will write:

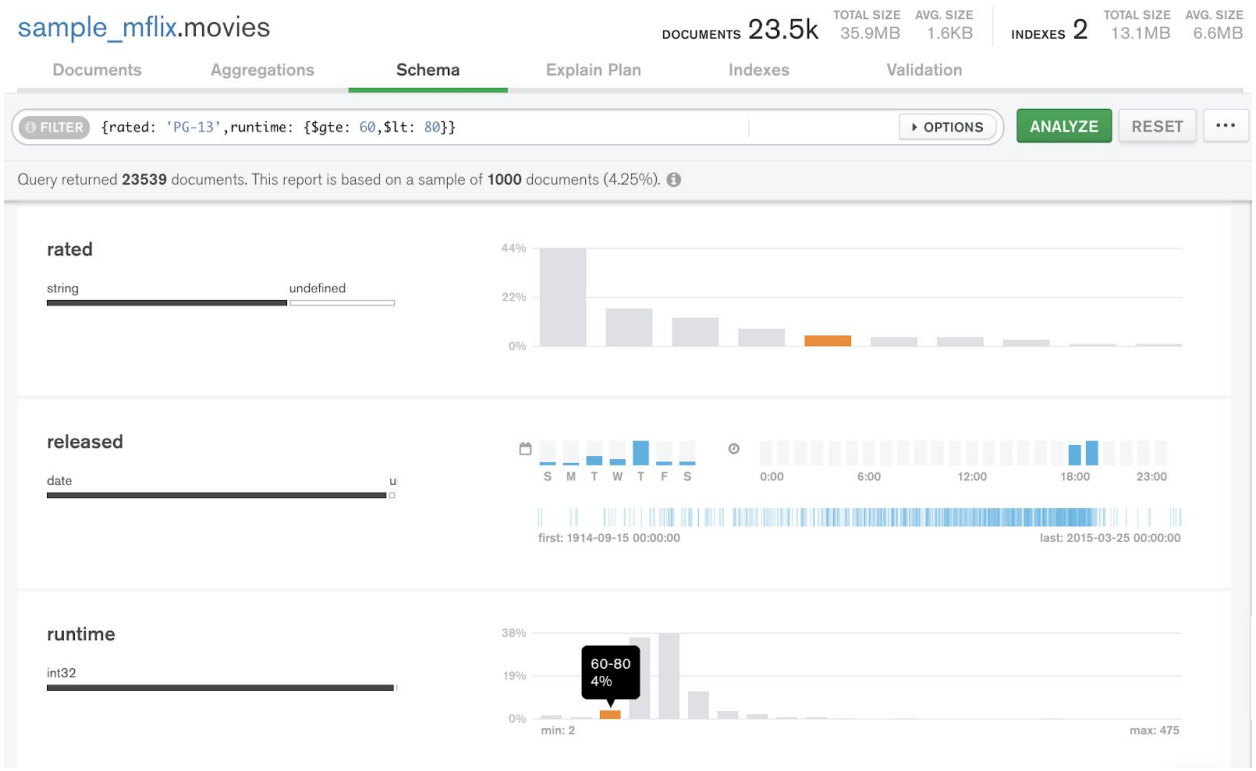
db.movies.find({<<INSERT FILTER>>})

or

db.movies.find({rated: 'PG-13'})

6.B. Multiple Conditions:

You can list multiple conditions in your query by inserting a comma in between conditions. Let's add another condition as a range by making a selection in the **runtime** field:



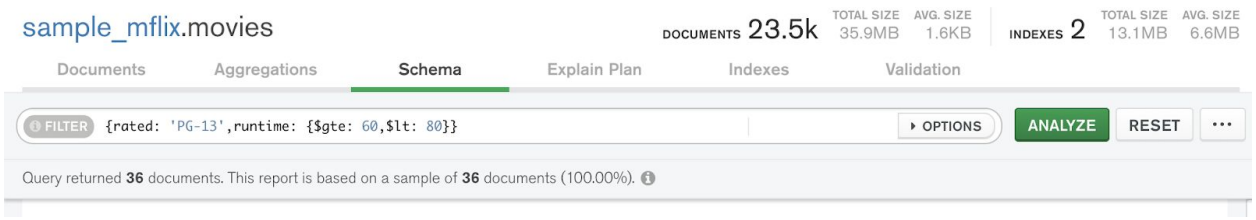
In the screenshot above, to query for movies in the 60-80 minute range, MongoDB query language adds **runtime:{$gte: 60, $lt:80}** to the query. Feel free to play with the values to look for movies for any duration.

Note that selectors in MongoDB are ANDed together by default with a comma. So in the MongoDB Query Language, to find movies that are BOTH rated PG-13 with runtimes between 60 and 80 minutes, you would write:

{ rated:'PG-13', runtime:{ $gte: 60, $lt:80 }

And finding documents that match this query from your application would then be:

db.movies.find({ rated:'PG-13', runtime:{ $gte: 60, $lt:80 })



Switch to the Documents tab in Compass to see these 36 movies.

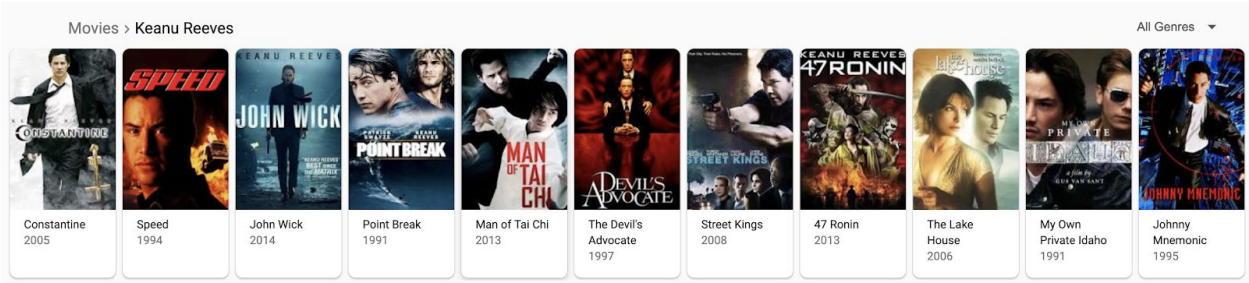
6.C. Embedded Documents:

Now let's discuss how to find equality for embedded documents. Go back to schema and scroll down to tomatoes. Open up the Object and click on any of the **boxOffice** values. You will see that to query for a nested field, you will use the dot notation :

Ex. **{ 'tomatoes.boxOffice': '$14.8M'}**

6.D Querying Arrays:

Finally let's look at exact matches for an array field by looking for our favorite actors in the cast field. I am a Keanu Reeves fan so let's look for some of his movies. 😍💖



First, let's reset our fields in Compass by clearing the filters and clicking the RESET button. You will notice in any movie document opening the **cast** array will list the main actors as strings in an array.

```
_id: ObjectId("573a1390f29313caabcd4135")
plot: "Three men hammer on an anvil and pass a bottle of beer around."
› genres: Array
runtime: 1
⌄ cast: Array
    0: "Charles Kayser"
    1: "John Ott"
num_mflix_comments: 1
title: "Blacksmith Scene"
fullplot: "A stationary camera looks at a large anvil with a blacksmith behind it..."
› countries: Array
released: 1893-05-09T00:00:00.000+00:00
› directors: Array
rated: "UNRATED"
› awards: Object
lastupdated: "2015-08-26 00:03:50.133000000"
year: 1893
› imdb: Object
type: "movie"
› tomatoes: Object
```

The above document for the movie **Blacksmith Scene** has Charles Kayser and John Ott.

To find movies with Keanu Reeves- or any singular actor, simply do a normal find as if cast is a scalar field:

{ cast: "Keanu Reeves" }

and you will find 27 different movie documents:

Now, since it is an array, if you want to find movies with a certain ensemble, **more than 1 actor,** we will use the **$all** operator:

{ cast: { $all:["Keanu Reeves", "Sandra Bullock"]} }

to return 2 movies: ***Speed*** and ***The Lake House*** (which make me completely jealous of Sandra Bullock !)

Now that you have learned some of the fundamentals of the MongoDB query language, play around in Compass or your application to find your favorite movies or ways to impress your friends with movie trivia!