



Overview

This hands-on workshop is designed to get you familiar with all aspects of MongoDB Atlas, including deploying a cluster, querying MongoDB, configuring alerts in Atlas, and managing infrastructure with the Atlas API.

Prerequisites

To successfully complete this workshop:

- You must be able to make outgoing requests from your computer to MongoDB Atlas servers which will be running on port 27017. Please confirm that port 27017 is not blocked by your network by clicking <http://portquiz.net:27017>. If successful, you will see a page load that indicates you can make outgoing requests on port 27017.
- Privileges to install software on your computer. We will be installing MongoDB Compass in this workshop.

Hands-on Labs

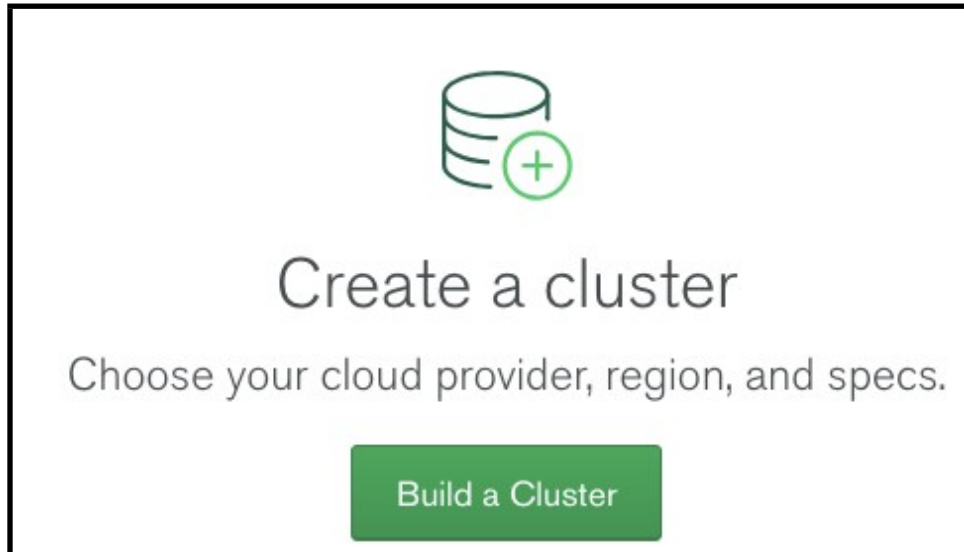
Lab 1 - Create the Cluster

Create an Account or Log In to Atlas

We'll be using [MongoDB Atlas](https://cloud.mongodb.com), 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, select **AWS** as the Cloud Provider:

Cloud Provider & Region

AWS, N. Virginia (us-east-1) ▼



Create a **free tier cluster** by selecting a region with **FREE TIER AVAILABLE** and choosing the **M0** cluster tier below.

★ Recommended region ⓘ

| NORTH AMERICA | EUROPE | ASIA |
|--|--|--------------------------------------|
| <div> N. Virginia (us-east-1) ★ FREE TIER AVAILABLE</div> | <div> Stockholm (eu-north-1) ★</div> | <div> Hong Kong (ap-east-1) ★</div> |
| <div> Ohio (us-east-2) ★</div> | <div> Ireland (eu-west-1) ★ FREE TIER AVAILABLE</div> | <div> Tokyo (ap-northeast-1) ★</div> |

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

Cluster Name

Workshop

One time only: once your cluster is created, you won't be able to change its name.

Workshop

Cluster names can only contain ASCII letters, numbers, and hyphens.

The remaining defaults will suffice.

Click **Create Cluster**:

FREE

Free forever! Your M0 cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.

Cancel

Create Cluster

Continue to Lab 2 while the cluster is provisioning.

Lab 2 - Connect to the Cluster

Install Compass

Compass is the GUI for MongoDB. On the [download](#) page you need to select a Version and Platform:

MongoDB Compass

As the GUI for MongoDB, MongoDB Compass allows you to make smarter decisions about document structure, and subscriptions include technical support for MongoDB Compass.

MongoDB Compass is available in several versions, described below. For more information on version differences

Version

1.19.12 (Stable)

Platforms

RedHat 64-bit (7+)

Download

Make sure you select the “Stable” version, which contains the enterprise features we’ll use in a moment.

Setup Connection Security

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

×

Connect to Workshop

Setup connection security

Choose a connection method

Connect

You need to secure your MongoDB Atlas cluster before you can use it. Set which users and IP addresses can access your cluster now. [Read more](#)

You can't connect yet. Set up your firewall access and user security permission below.

1

Whitelist your connection IP address

Add Your Current IP Address

Add a Different IP Address

2

Create a MongoDB User

This first user will have [atlasAdmin](#) permissions for all clusters in this project.

Keep your credentials handy, you'll need them for the next step.

Username

ex. dbUser

Password

ex. dbUserPassword

SHOW

Autogenerate Secure Password

Create MongoDB User

Close

Choose a connection method >

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

Note: depending on the configuration of the network you're connected to, you may have to whitelist 0.0.0.0/0 (allow connections from everywhere).

You can't connect yet. Set up your firewall access and user security permission below.

1 Whitelist your connection IP address

| IP Address | Description (Optional) |
|--|--|
| <input type="text" value="97.76.196.230"/> | <input type="text" value="Hilton Garden Inn"/> |
| <div>Cancel Add IP Address</div> | |

2 Create a MongoDB User

This first user will have [atlasAdmin](#) permissions for all clusters in this project.

Keep your credentials handy, you'll need them for the next step.

| | | |
|---------------------------------------|--|--|
| Username | Password | Autogenerate Secure Password |
| <input type="text" value="workshop"/> | <input type="password" value="....."/> | SHOW |
| <div>Create MongoDB User</div> | | |

Close

Choose a connection method >

Click **Choose a connection method** and select **Connect with MongoDB Compass**.

Then select **I am using Compass 1.12 or later** and **COPY** the connection string presented:

×

Connect to Workshop

✓ Setup connection security

✓ Choose a connection method

Connect

1

If you have not already, click below to download Compass

Windows

Mac OS X

Other Operating Systems ▾

2

Copy the URI Connection String

[View detailed instructions](#)

I am using Compass 1.12 or later

I am using Compass 1.11 or earlier

mongodb+srv://workshop:<PASSWORD>@workshop-ghuzr.mongodb.net/admin

COPY

Replace **PASSWORD** with the password for the *workshop* user.

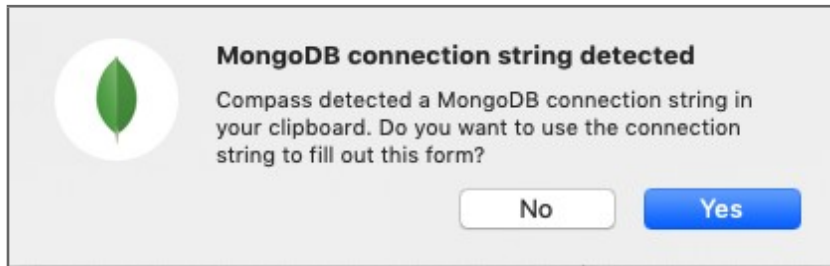
When you open Compass, it should detect the URI string from your clipboard and auto-populate the form.

◀ Choose a connection method

Close

Connect Compass

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

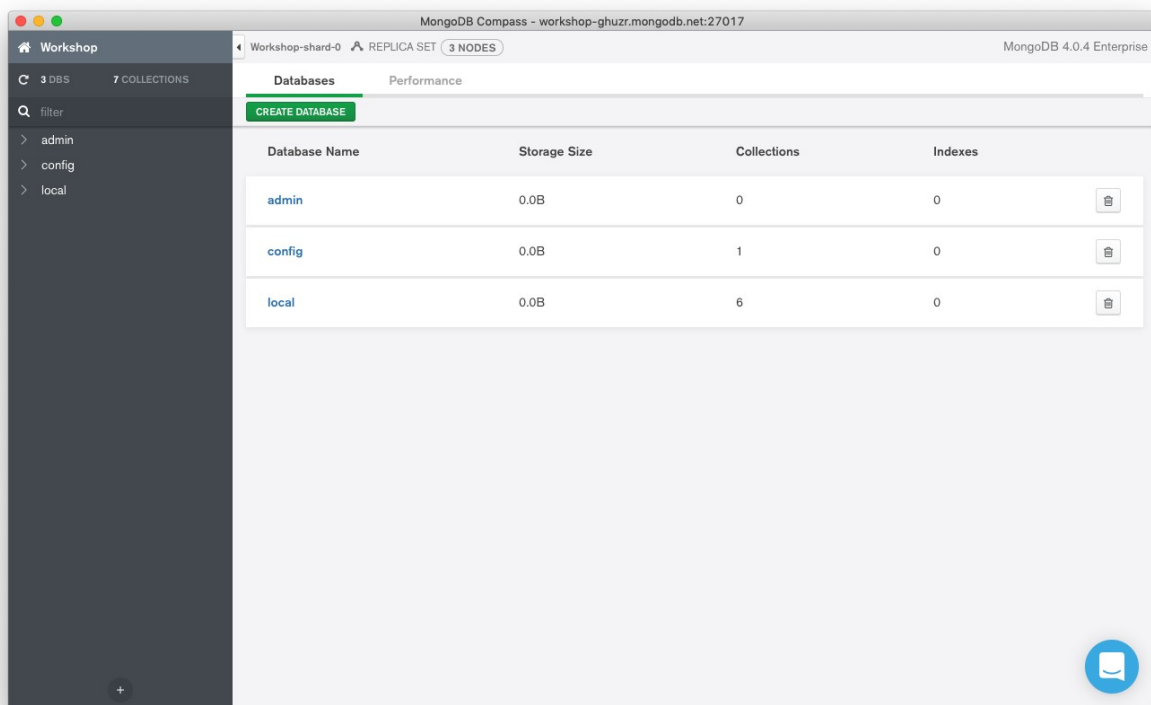


Select **Yes**.

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 this workshop we're going to load a Yelp like collection of New York City restaurants. Download the dataset from Github. If you have the wget utility, you can get the dataset as follows:

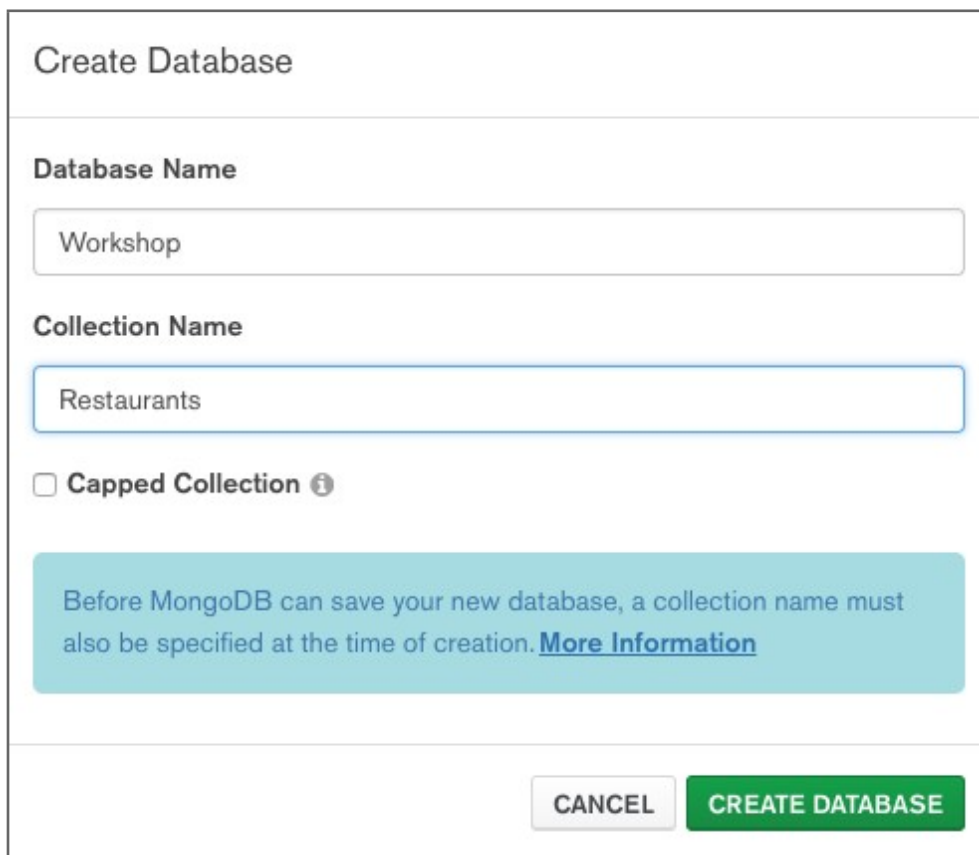
wget <https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json>

Otherwise, just open the link in your browser and once the load completes, save the file.

The dataset is 11.9 MB and has 25K restaurants.

Create a Database and Collection

In Compass, click the **CREATE DATABASE** button and create a **Workshop** database with a **Restaurants** collection:



The screenshot shows the 'Create Database' dialog box in MongoDB Compass. It has a title bar 'Create Database'. Below the title bar, there are two input fields: 'Database Name' with the value 'Workshop' and 'Collection Name' with the value 'Restaurants'. Below these fields is a checkbox labeled 'Capped Collection' with an information icon. At the bottom of the dialog, there are two buttons: 'CANCEL' and 'CREATE DATABASE'. A light blue message box is visible, stating: 'Before MongoDB can save your new database, a collection name must also be specified at the time of creation. [More Information](#)'.

Navigate to the Restaurants collection and select **Import Data** from the menu. Then **BROWSE** to the primer-dataset.json file you downloaded:

Import To Collection Workshop.Restaurants

Select Input File Type

JSON CSV

Select File

/Users/brianleonard/Downloads/primer-dataset.json BROWSE

CANCEL IMPORT

Then select **IMPORT**. You've just imported 25K documents!

Lab 4 - Browse the Documents

Notice how the restaurant documents have a nested subdocument (address) and an array of subdocuments (grades). 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

Wait, I thought MongoDB was 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**.

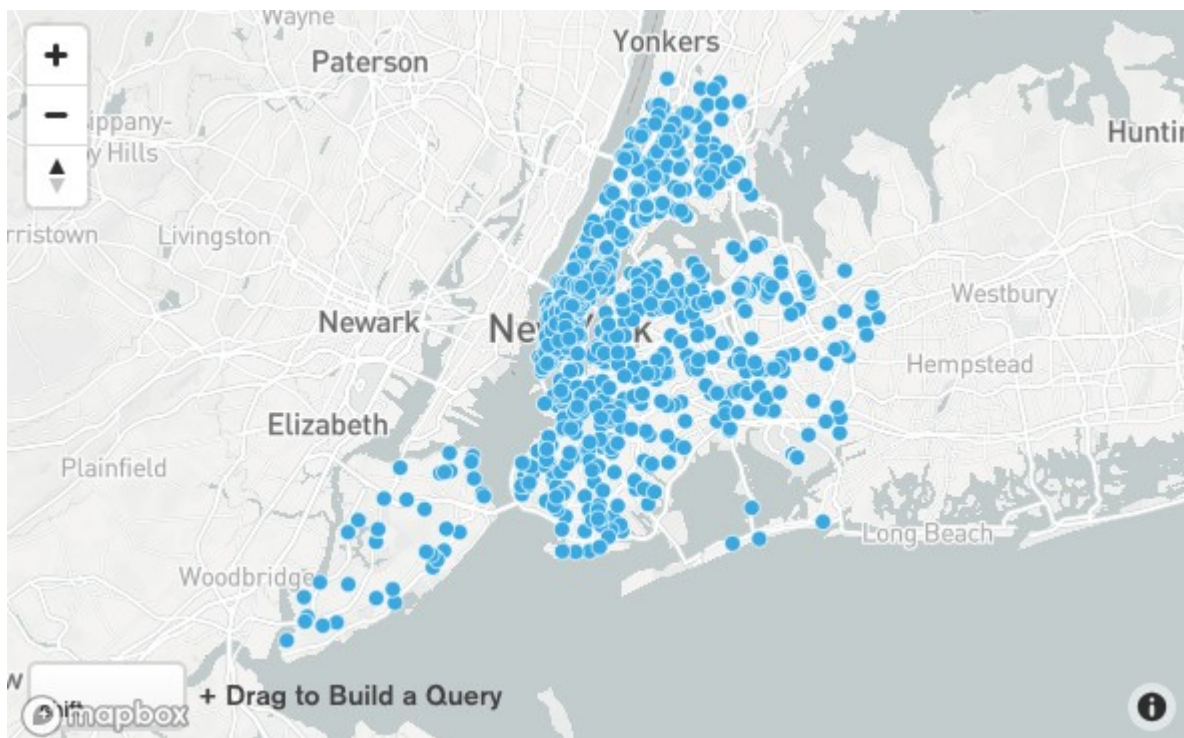
Select the **Schema** tab and select **Analyze Schema**.

*Note, if you don't have a **Schema** tab, you're running the Community Edition of Compass. Revist Lab 2 to get the correct edition.*

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 cuisine, we can see that Chinese is the 2nd most common at 12% (your results may differ slightly based on the sample that was taken):

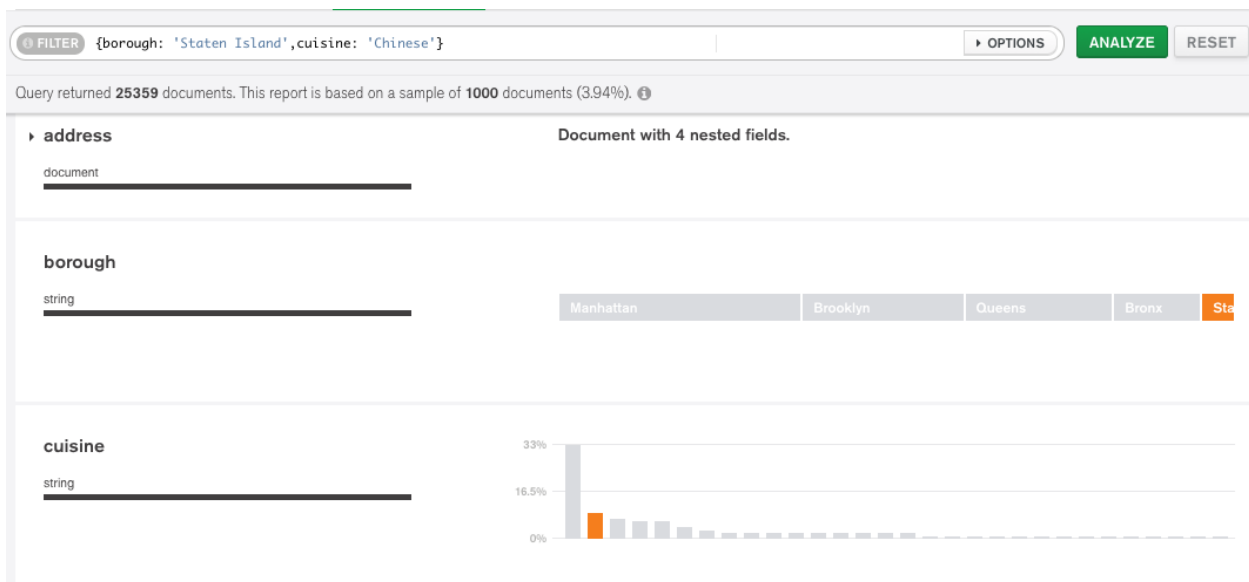


Expand the address field to discover MongoDB's excellent support for [Geospatial Queries](#). As the collection is of restaurants in New York City, zoom the map to NYC:

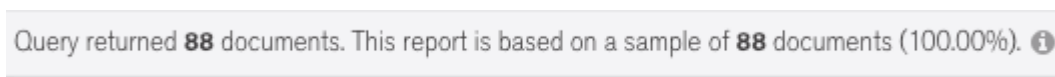


Lab 6 - Query the Data

Unsurprisingly, the MongoDB Query Language (MQL) is also based on JSON. The Schema Analyzer in Compass provides an easy way to learn the language. For example, select **Staten Island** from the borough field (only **Sta** may be showing) and **Chinese** from the cuisine field. Notice as you make selections the FILTER field at the top of the window gets populated:



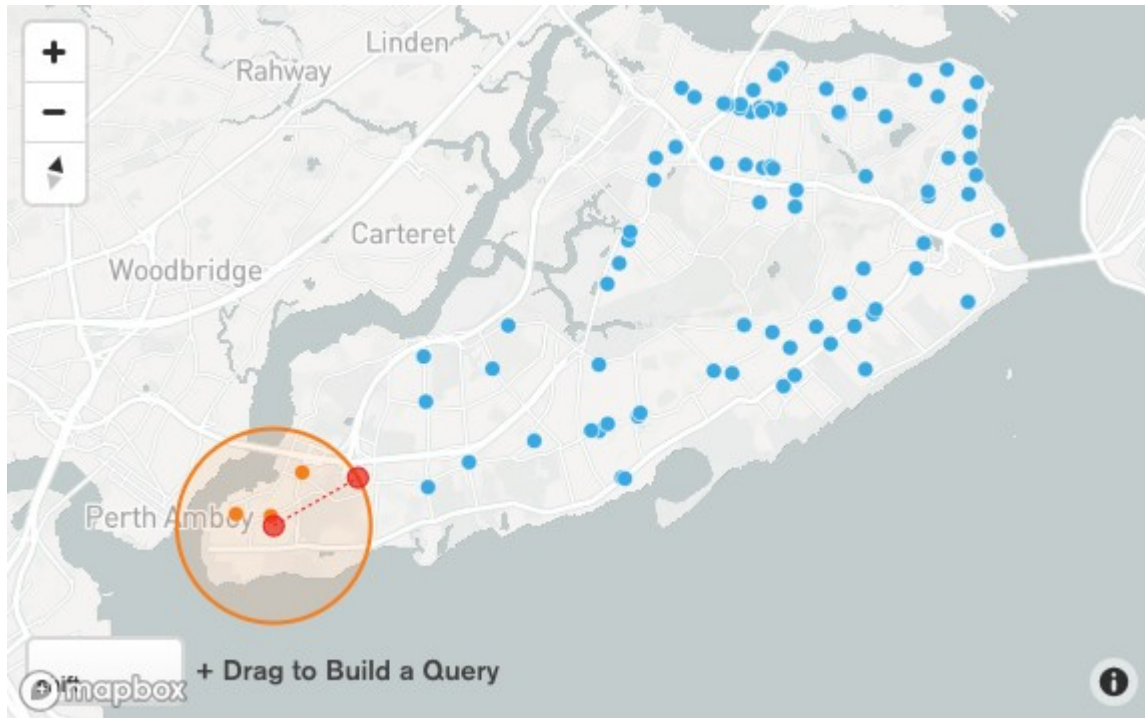
Click the **ANALYZE** button to filter for Chinese restaurants in Staten Island, of which there are 88:



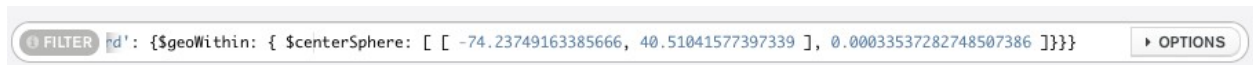
And you can now see this reflected on our map (more dots now appear on Staten Island because our sample now includes all 88 restaurants)



To perform a geospatial query, shift click and drag a circle on the map. Once the circle is in place, it can be moved and resized:



And notice the [\\$geoWithin](#) filter that got added to our query:



Finally, click **ANALYZE** again and click the **Documents** tab to view the Chinese restaurants in our selected radius in Staten Island:

Documents

Aggregations

Schema

FILTER

{borough: 'Staten Island', cuisine: 'Chinese', 'addr

INSERT DOCUMENT

VIEW

LIST

TABLE

_id: ObjectId("5beb3a21af37deb50165abb9")

> address: Object

borough: "Staten Island"

cuisine: "Chinese"

> grades: Array

name: "Kim'S Island Chinese Restaurant"

restaurant_id: "41436344"

_id: ObjectId("5beb3a25af37deb50165caaf")

> address: Object

borough: "Staten Island"

cuisine: "Chinese"

> grades: Array

name: "Loon Chuan Restaurant"

restaurant_id: "41717895"

_id: ObjectId("5beb3a27af37deb50165e0cb")

> address: Object

borough: "Staten Island"

cuisine: "Chinese"

> grades: Array


name: "Chef Hong"

restaurant_id: "50015617"

Lab 7 - Configuring Alerts

Atlas provides a system of alerts that allows users to keep track of a variety of different events that may take place within a cluster. For this lab, we'll set up an alert that will send us an email if the dataset exceeds a certain size.

To get started, navigate to "Alerts" on the sidebar and select "Alert Settings"

 mongoDB

MG-Demo

Alerts

ATLAS **ACTIVE**

Clusters

Data Lake BETA

SECURITY

Database Access

Network Access

Advanced

PROJECT

Access Management

Activity Feed

Alerts 0

| Open Alerts | Closed Alerts | Alert Settings |
|----------------------|---|----------------|
| Active Alerts | Deleted Alerts | |
| Target | Condition | |
| Host of type Primary | Disk space % used on Data Partition above | |
| Host of type Primary | Opcounter: Insert above 100 (avg/sec) | |
| User | User joined the project | |

There are a bunch of alerts that are enabled by default helping us track disk usage, cpu usage, failover events, etc. Default alerts can be modified just like alerts that you create yourself.

To add a new alert, click “Add” in the upper right corner.



Set an alert so that if a host of any type has a logical data size greater than 250 MB, we'll get an email.

Edit Alert

1 Alert if

| TARGET | HOST TYPE | CONDITION/METRIC |
|-------------------|----------------------|--------------------------------------|
| Host > | of any type > | Journaling Write Data Files MB is... |
| Replica Set > | of type Standalone > | Logical Size is... |
| Sharded Cluster > | of type Primary > | Memory: Mapped is... |
| User > | of type Secondary > | Memory: Resident is... |
| Project > | of type Arbiter > | Memory: Virtual is... |
| | | Network: Bytes is... |

2 For ☒ Any Host ☐ Hosts where...

3 Send to

Project: ☒ All Roles send if condition lasts at least mins, resend after mins ☒ Email ☐ SMS

Add ▾

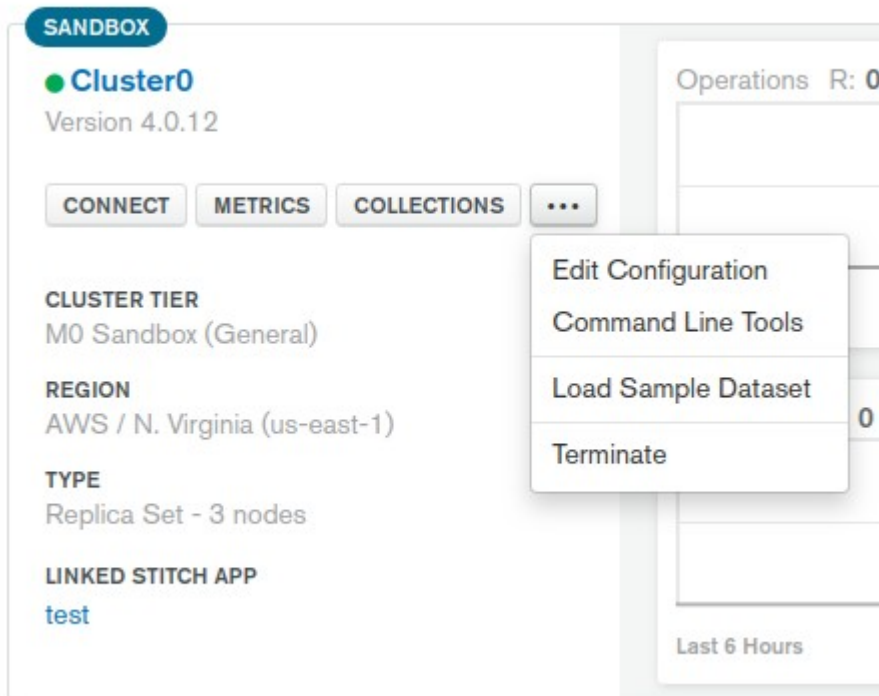
- Atlas Organization
- Atlas User
- Email
- SMS
- HipChat
- Slack
- Flowdock
- VictorOps
- OpsGenie
- PagerDuty
- Webhook
- Datadog

Save

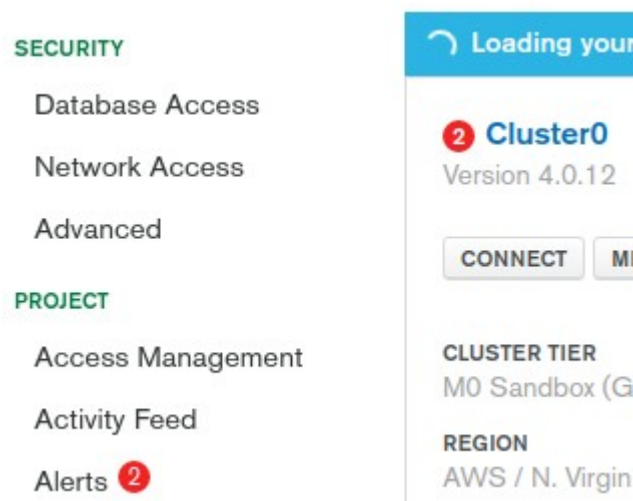
| | | |
|---|-------------------------------|---|
| C | re | Project owner send if condition lasts 0 mins, resend every |
| a | Objects / Returned above 1000 | Project owner send if condition lasts 0 mins, resend every |
| D | ata Partition above 90 | Project owner send if condition lasts 0 mins, resend every |

Set the alert to be sent to “all roles” as an email. Also, browse through the wide variety of other services to which you can send alerts.

To trigger our new alert, navigate back to your cluster and click the “...” and then “Load Sample Dataset”



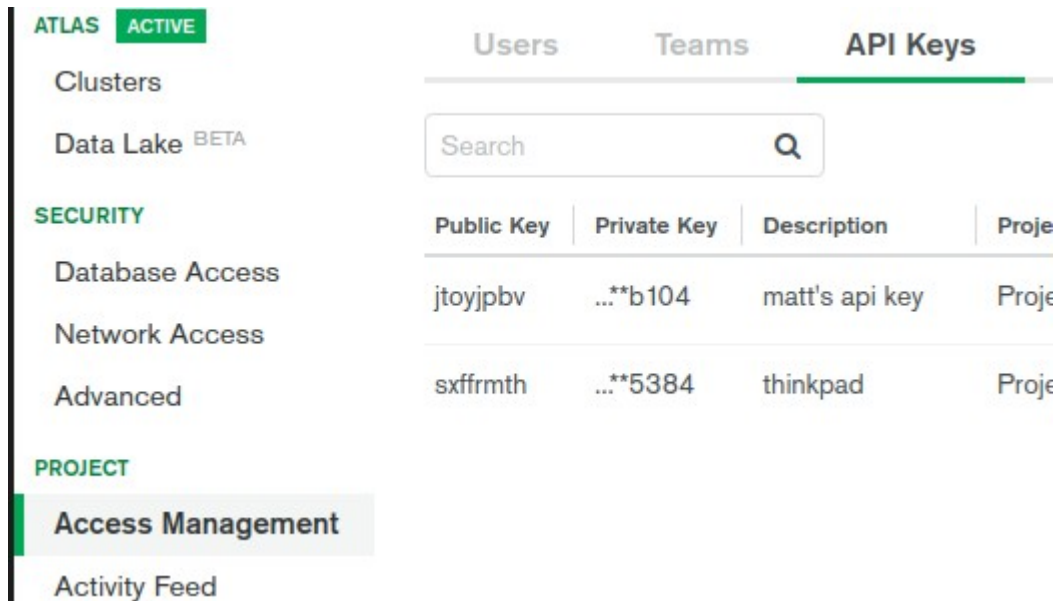
Once the dataset exceeds 250 MB, the alert should be triggered. This will be visible in the Atlas user interface and an email will be sent to the email address you used to create your Atlas account.



Lab 8 - Using the API

Atlas exposes a robust API to allow users to programmatically manage their clusters. Using the API a user can create/terminate clusters, pause and resume clusters, restore a backup, pull database logs, and more. In this lab we'll use the Atlas API to retrieve information about the clusters in our project as well as capture some cluster event data.

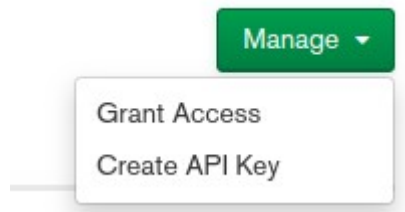
The first step to using the Atlas API is to generate an API key. Navigate to “Access Management” on the sidebar and choose “API Keys”



The screenshot shows the MongoDB Atlas interface. On the left sidebar, under the 'ATLAS ACTIVE' section, 'Access Management' is selected. The main content area has tabs for 'Users', 'Teams', and 'API Keys', with 'API Keys' being the active tab. Below the tabs is a search bar. A table lists the API keys with columns: Public Key, Private Key, Description, and Project. Two keys are visible: one with public key 'jtoyjpbv' and description 'matt's api key', and another with public key 'sxffrmth' and description 'thinkpad'.

| Public Key | Private Key | Description | Project |
|------------|-------------|----------------|---------|
| jtoyjpbv | ...**b104 | matt's api key | Project |
| sxffrmth | ...**5384 | thinkpad | Project |

To create a key, click on “Manage” and select “Create API Key”



The screenshot shows a green 'Manage' button with a dropdown arrow. The dropdown menu is open, showing two options: 'Grant Access' and 'Create API Key'.

Provide a description of the key and make sure permissions include “Project Owner”

Enter API Key Information * required

Public Key

DLHWVZRA

Description*

max 250 characters

my new key

Project Permissions*

Project Owner

Cancel

Next

Click next – the next screen will show your private key. Be sure to copy this key and save it!

✓ API Key Information

Private Key & Whitelist

Private Key

Please copy your full private key now and store it in a secure location.
The full private key will not be available after you leave this page.

17a469ba-3d9a-4b0d-a5ee-76f5c84699d3

Copy

API Whitelist

Each entry can be either a single IP address or a CIDR - notated IP range.

Add Whitelist Entry

Done

The next step is to whitelist your IP address for that particular API key. Click “Add Whitelist Entry” and add your public IP address.

×

Add Whitelist Entry

Creating an API Whitelist ensures that API calls must originate from whitelisted IPs.
[Learn more](#)

Whitelist Entry

USE CURRENT IP ADDRESS

199.255.10.131|

Save

Now that you have an API key, we can use it to retrieve information about clusters in our project. To save some time, let's set the public and private keys as environment variables. In a terminal, run the following commands:

```
export PRIVATEKEY=YourPrivateKeyHere
export PUBLICKEY=YourPublicKeyHere
```

The API calls we are using today both require a project id. To find your project id, navigate to “Settings” on the sidebar.

SECURITY

General

Database Access

Network Access

Advanced

PROJECT

Access Management

Activity Feed

Alerts 3

Integrations

Settings

Project ID

Project id will be here!

Project Name

Project Time Zone

Assign the project id to an environment variable as well.

```
export PROJECTID=YourProjectIDHere
```

Open atlas-api.sh in a text editor and copy the first command (Get information about all clusters in a project) into your terminal and run it. Atlas should respond with information about all the clusters (just one in this case) in your project.

Now run the second command in atlas-api.sh. This should return the same event data that is used to trigger events. In the response data, you should see "OUTSIDE_METRIC_THRESHOLD," this is the data for the alert we configured in the previous lab.

The atlas-api.sh file has two more examples for creating and modifying a cluster. These commands won't work on the free tier, but if you have a credit card associated with your account you can give them a try!