Introduction to

Hands-On Workshop

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

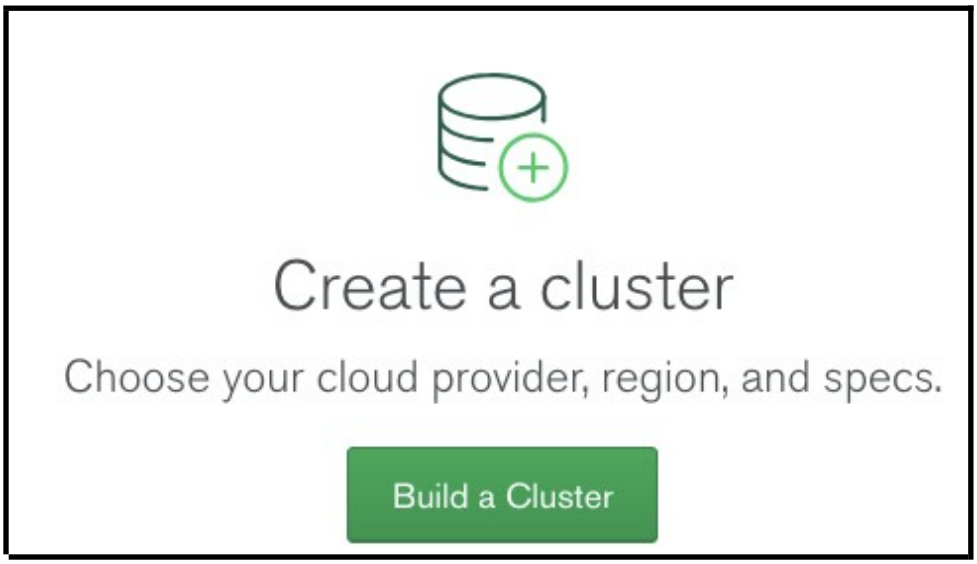
Lab 1 - Create the Cluster

Create an Account or Log In to Atas

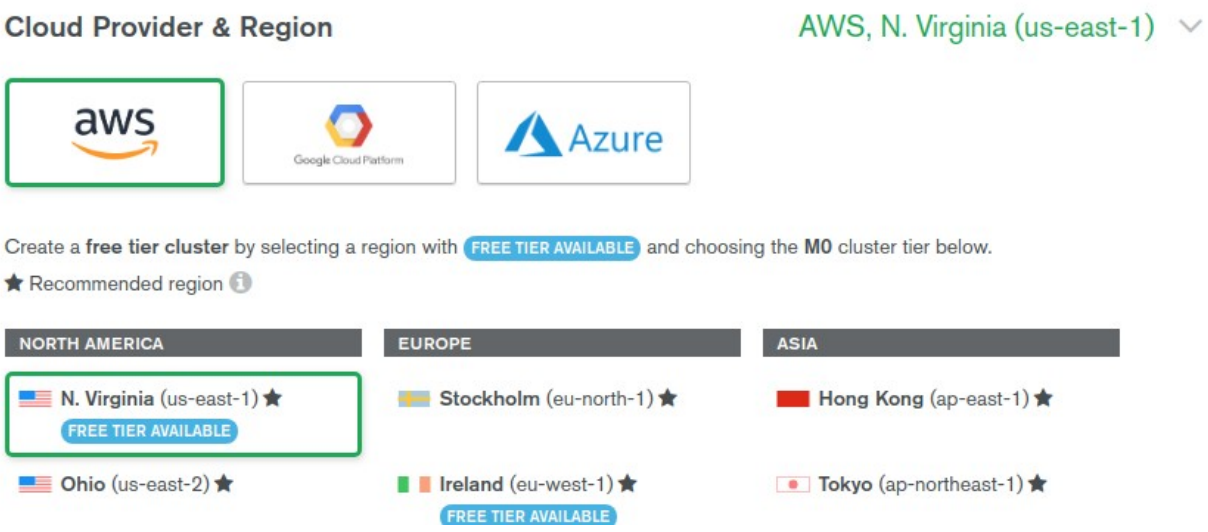
We'll be using MongoDB Atlas, our fully managed MongoDB-as-a-service, for this workshop. Go to https://cloud.mongodb.com and either create a new account or log into an existing account you may have previously created.

Create a Free Tier Cluster

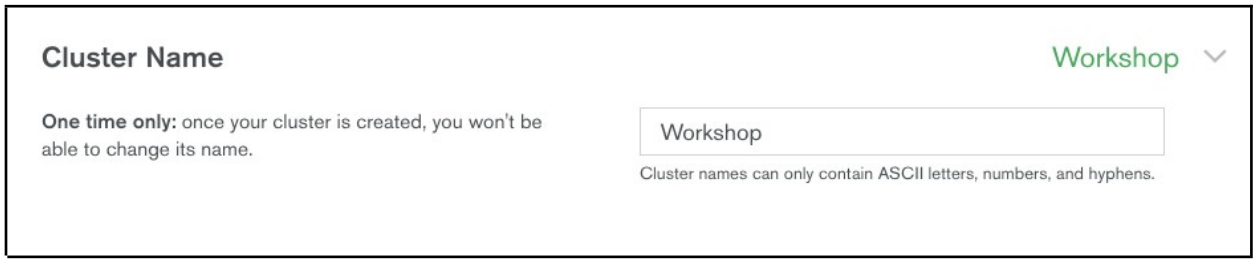
Click **Build a Cluster**:



Take a moment to browse the options (Provider & Region, Cluster Tier, Version, Backup, …). For our workshop, select **AWS** as the Cloud Provider:



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



The remaining defaults will suffice.

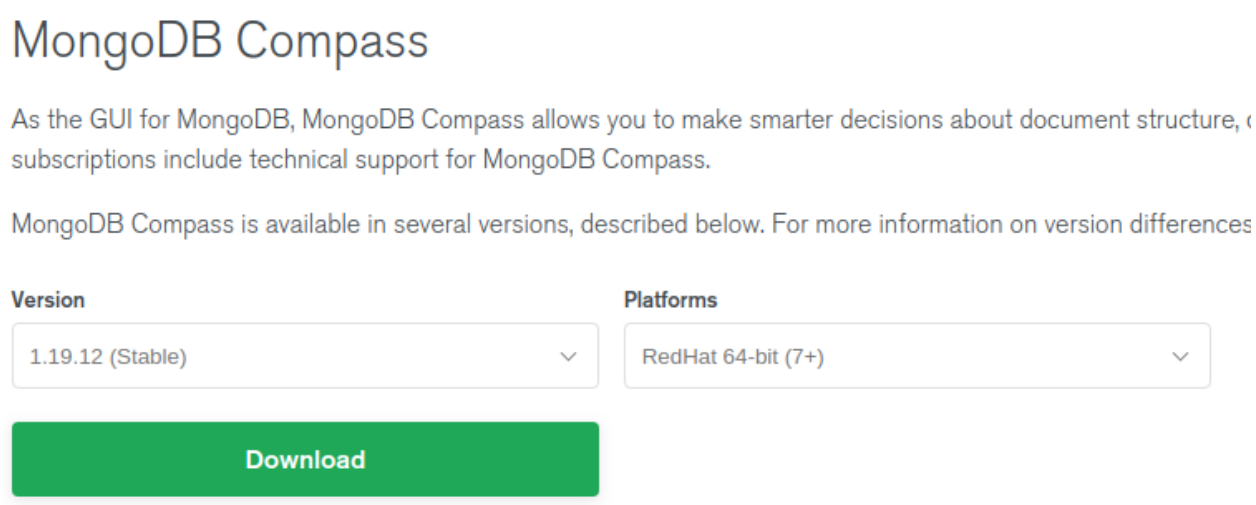
Click **Create Cluster**:

Continue to Lab 2 while the cluster is provisioning.

Lab 2 - Connect to the Cluster

Install Compass

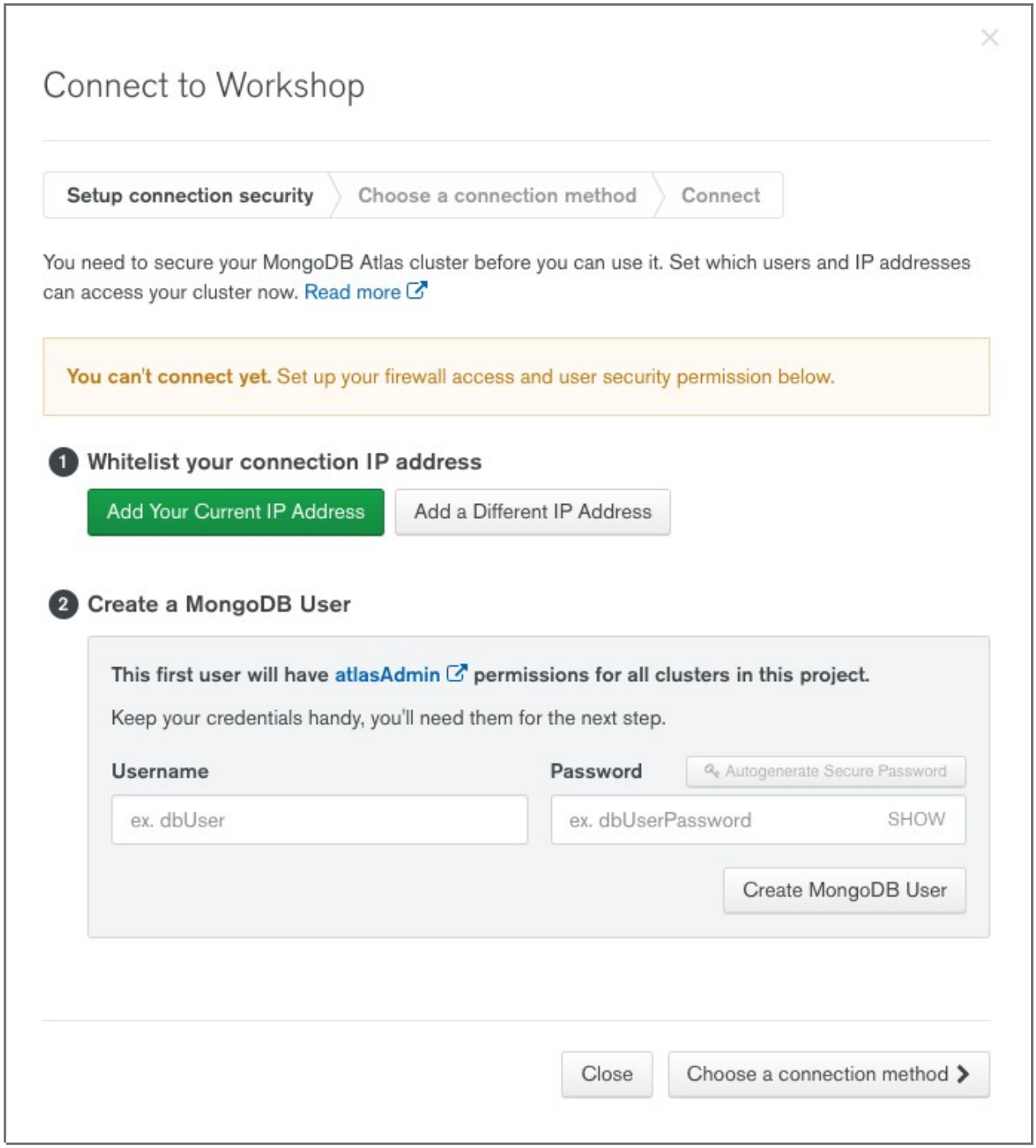
Compass is the GUI for MongoDB. On the download page you need to select a Version and Platform:



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

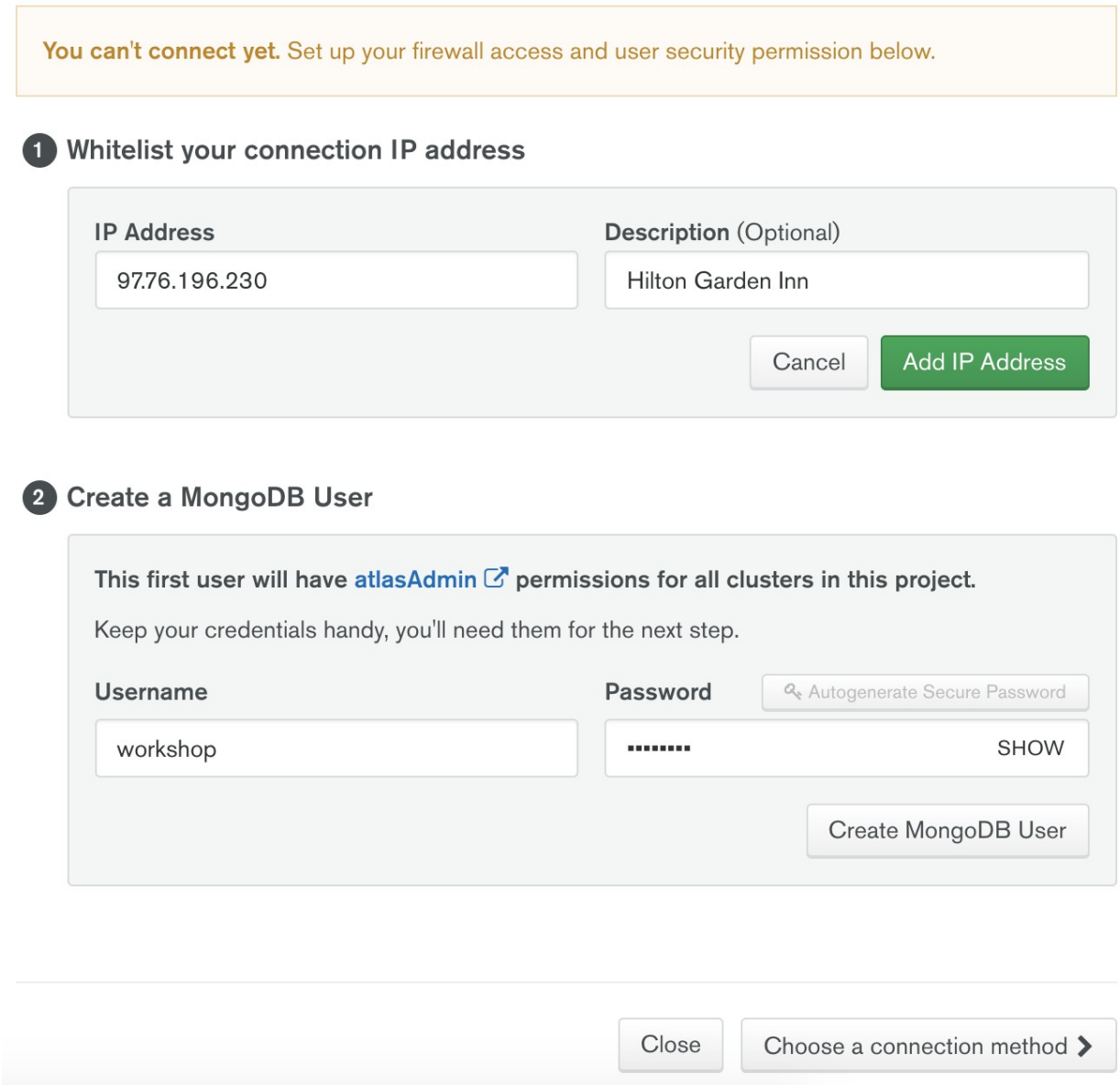
Setup Connection Security

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



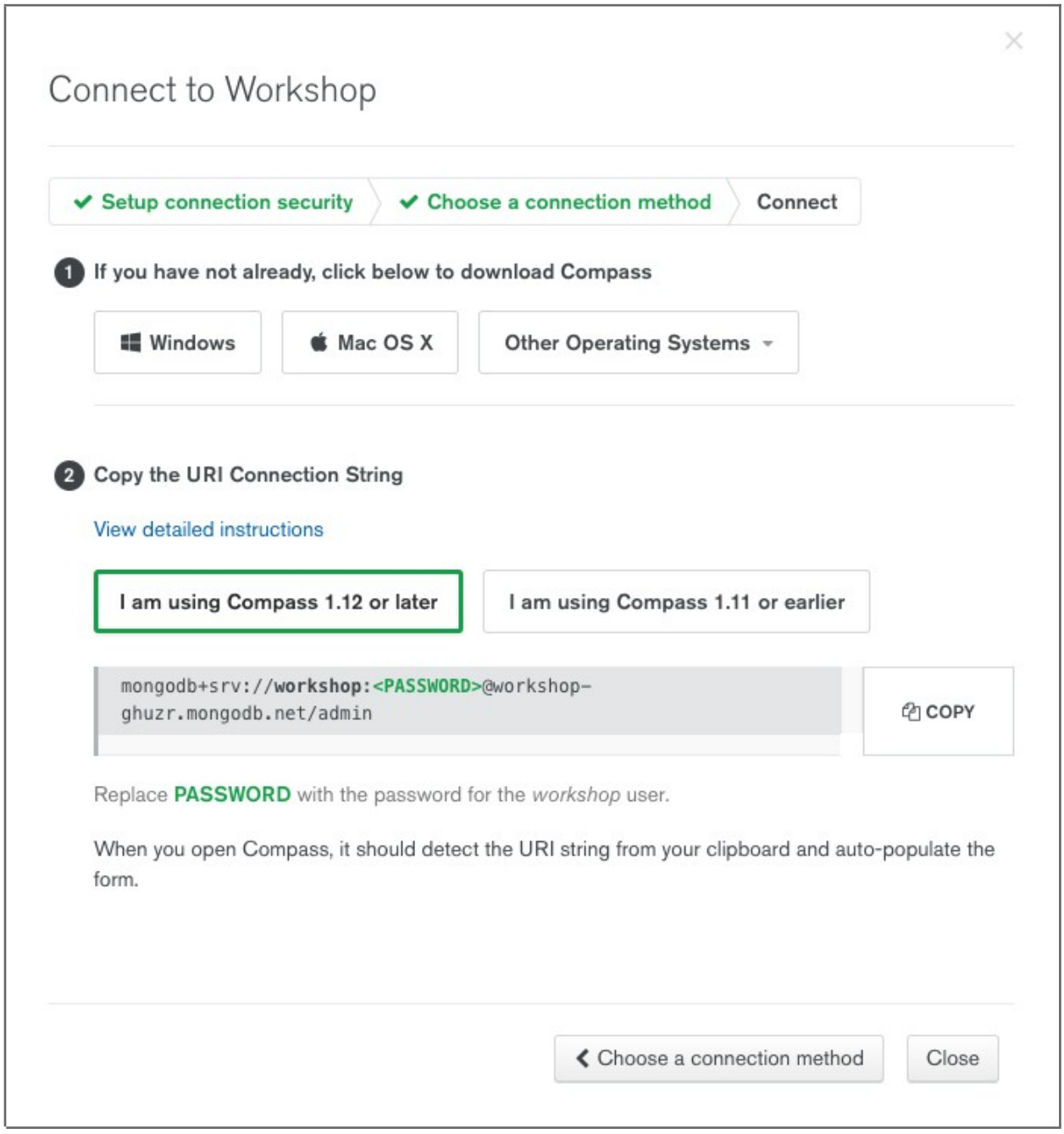
Add Your Current IP Address and **Create a MongoDB User.** I'm using Username **workshop** and password **workshop**:

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).



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 Compass

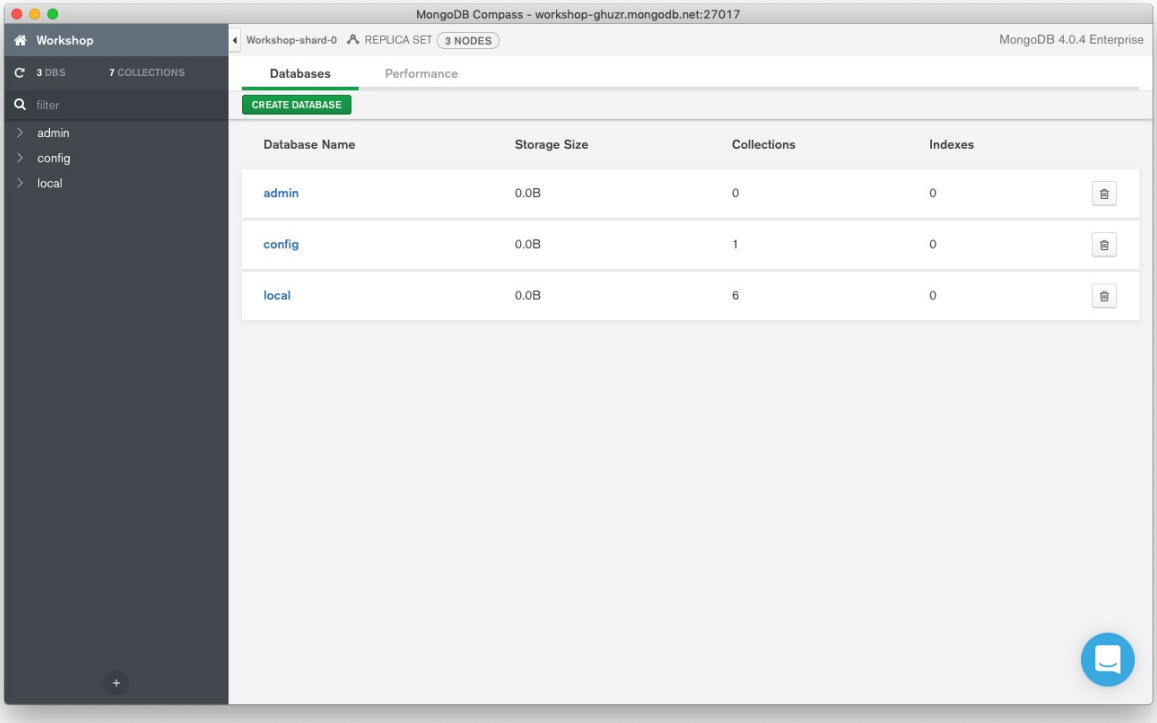
Start Compass and it should detect the connection string in your copy buffer:

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:

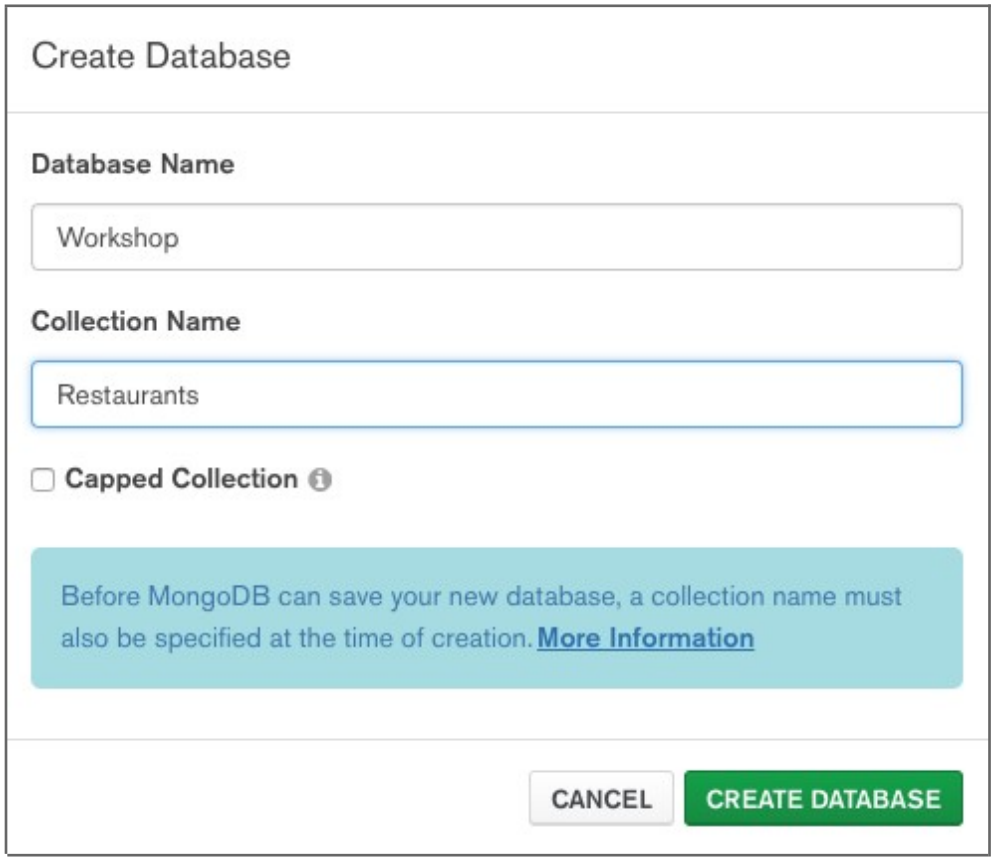
`wget https://raw.githubusercontent.com/mongodb/docs-assets/primer-dataset/primer-dataset.json`

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

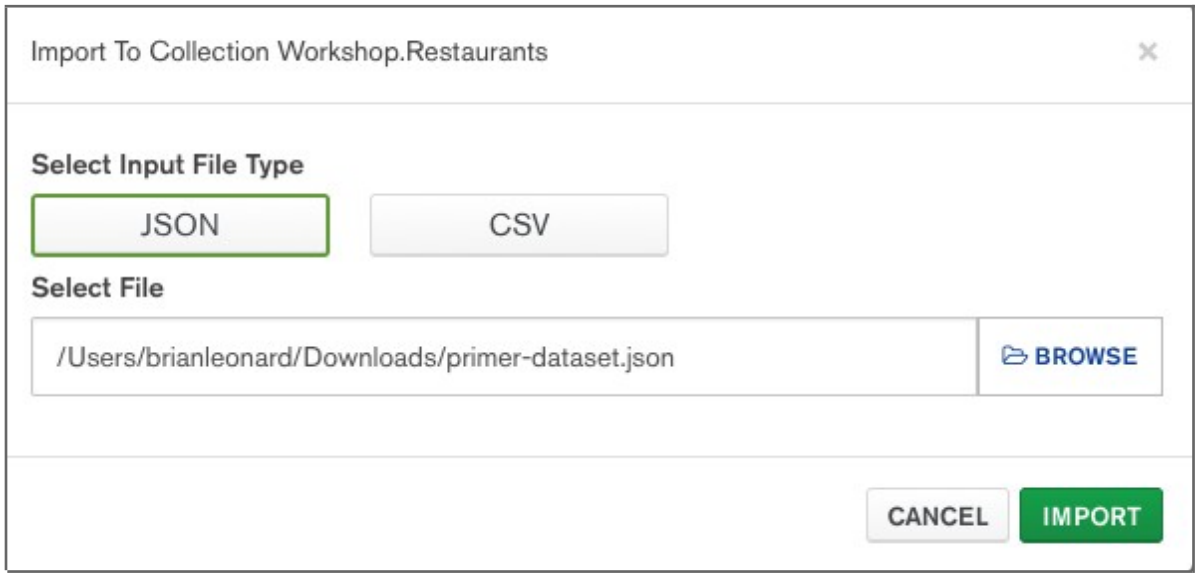
The dataset is 11.9 MB and has 25K restaurants.

Create a Database and Collection

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



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



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.

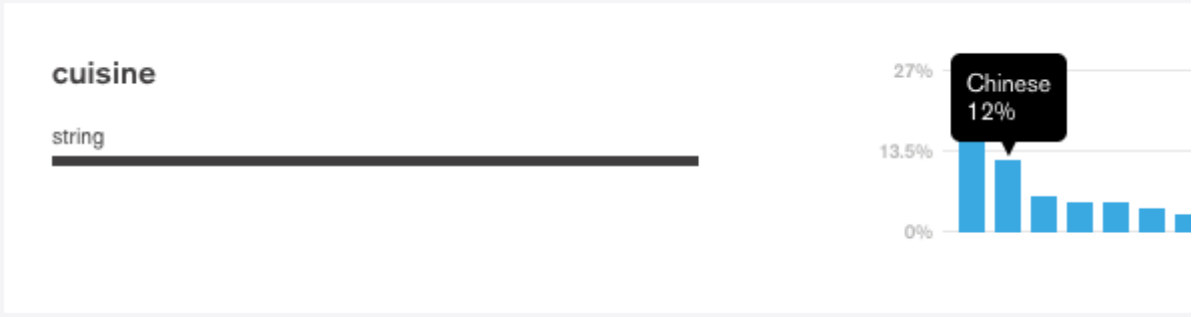
Lab 5 - View the Schema

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**.

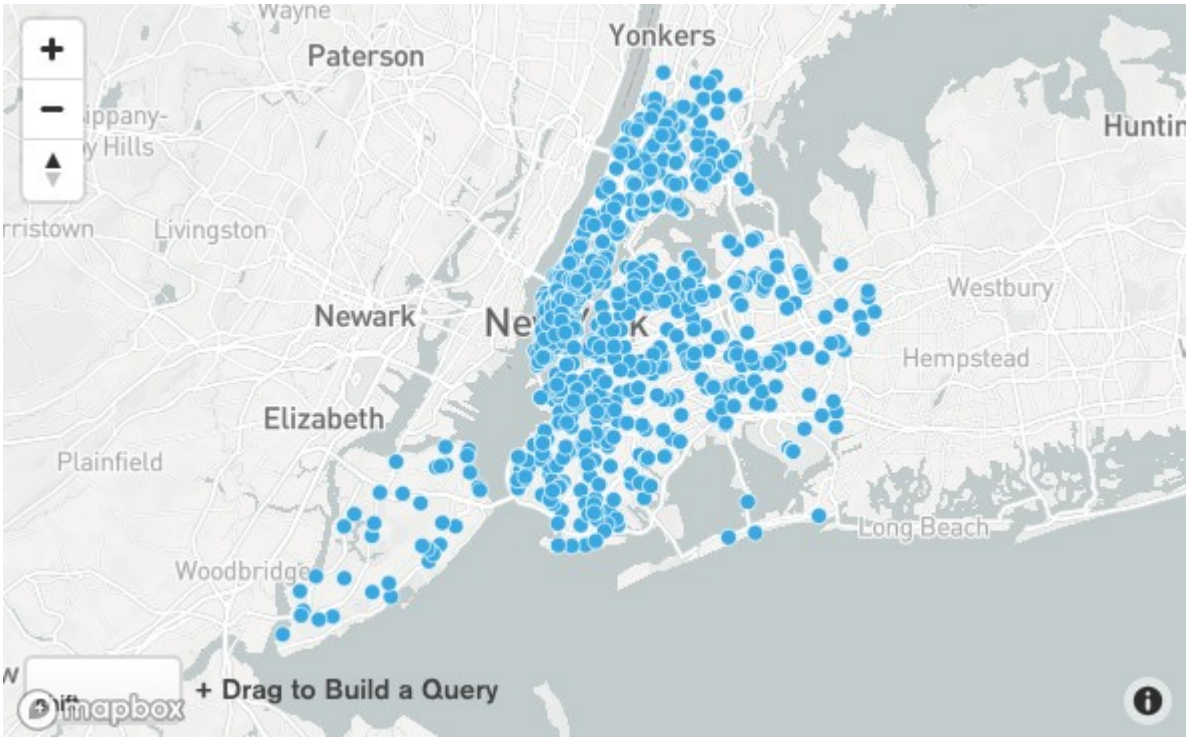
Select the **Schema** tab and select **Analyze Schema.**

> *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):

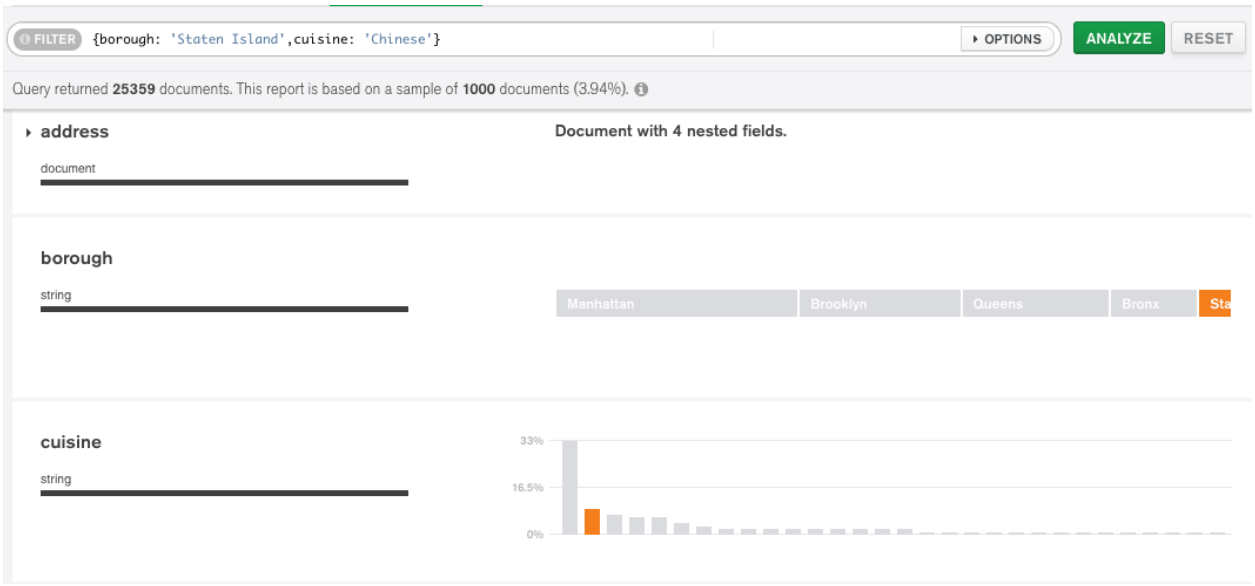


Expand the address field to discover MongoDB's excellent support for Geospatial Queries. As the collection is of restaurants in New York City, zoom the map to NYC:

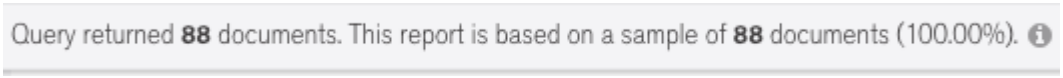


Lab 6 - Query the Data

Unsurprisingly, the MongoDB Query Language (MQL) is also based on JSON. The Schama 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:

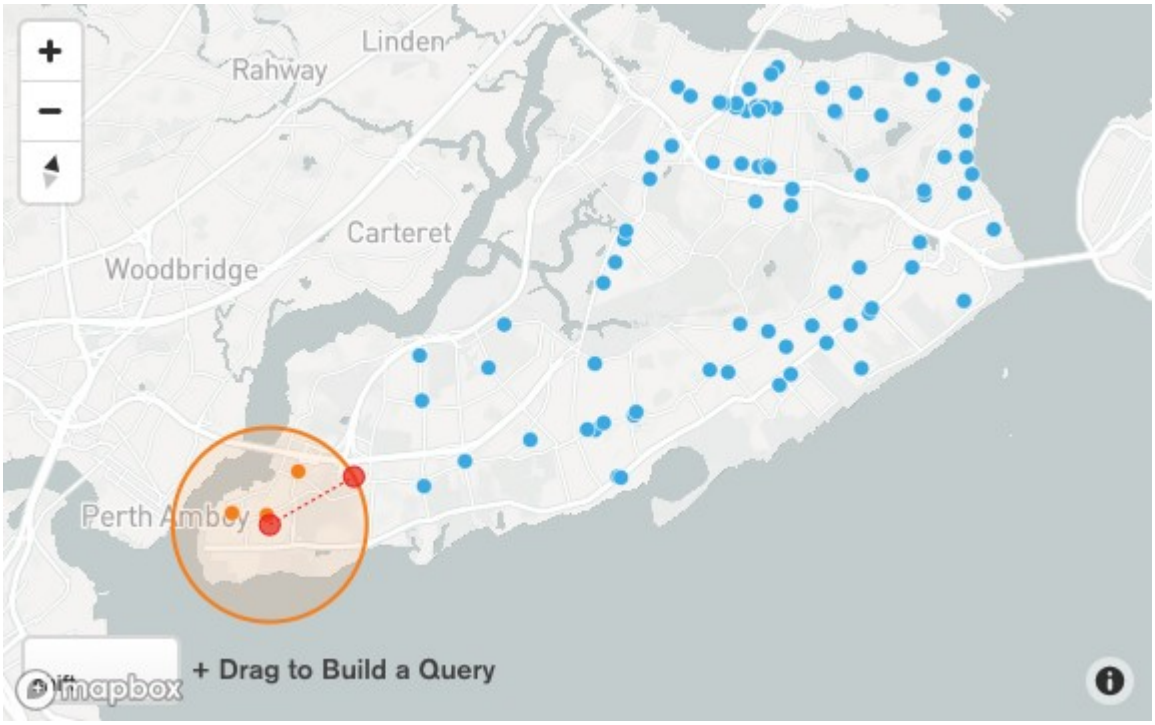


Click the **ANALYZE** button to filter for Chinese restaurants in Staten Island, of which there a 88:



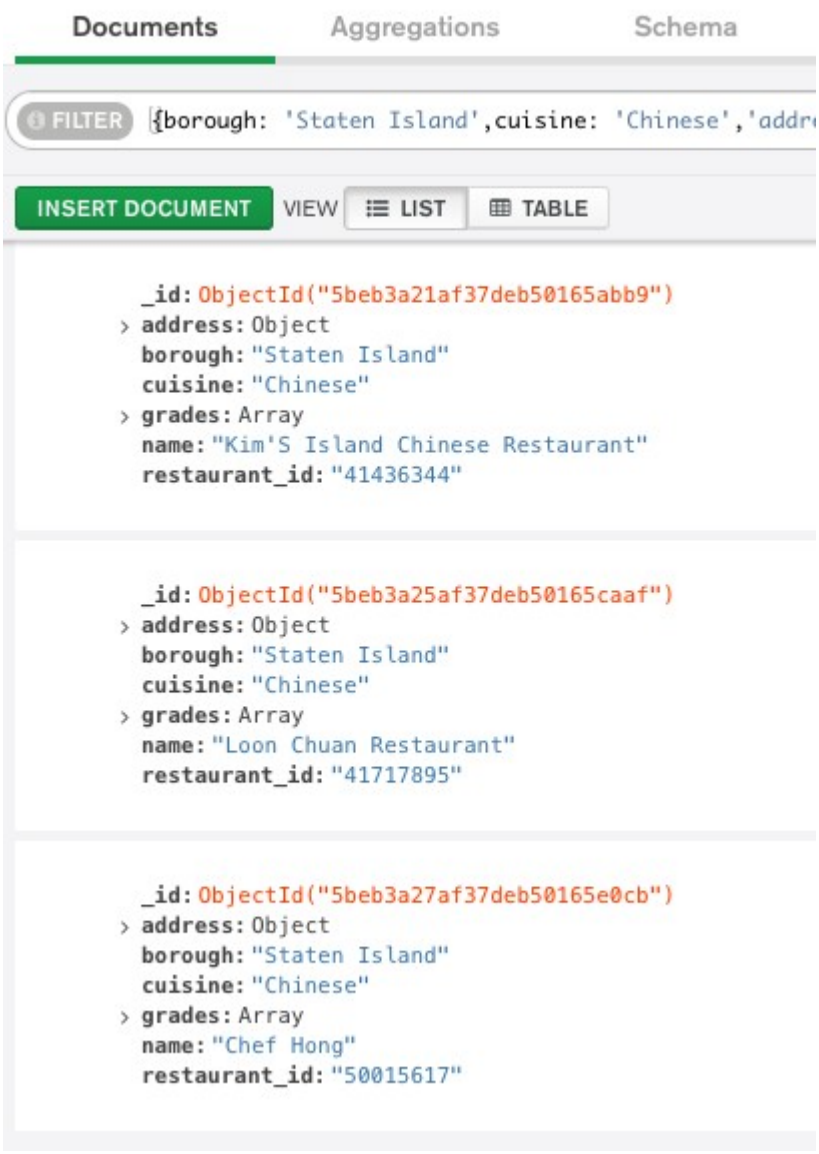
And you can now see this reflected on our map (more dots now appear on Staten Island because our sample now includes all 88 restaurants)

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:

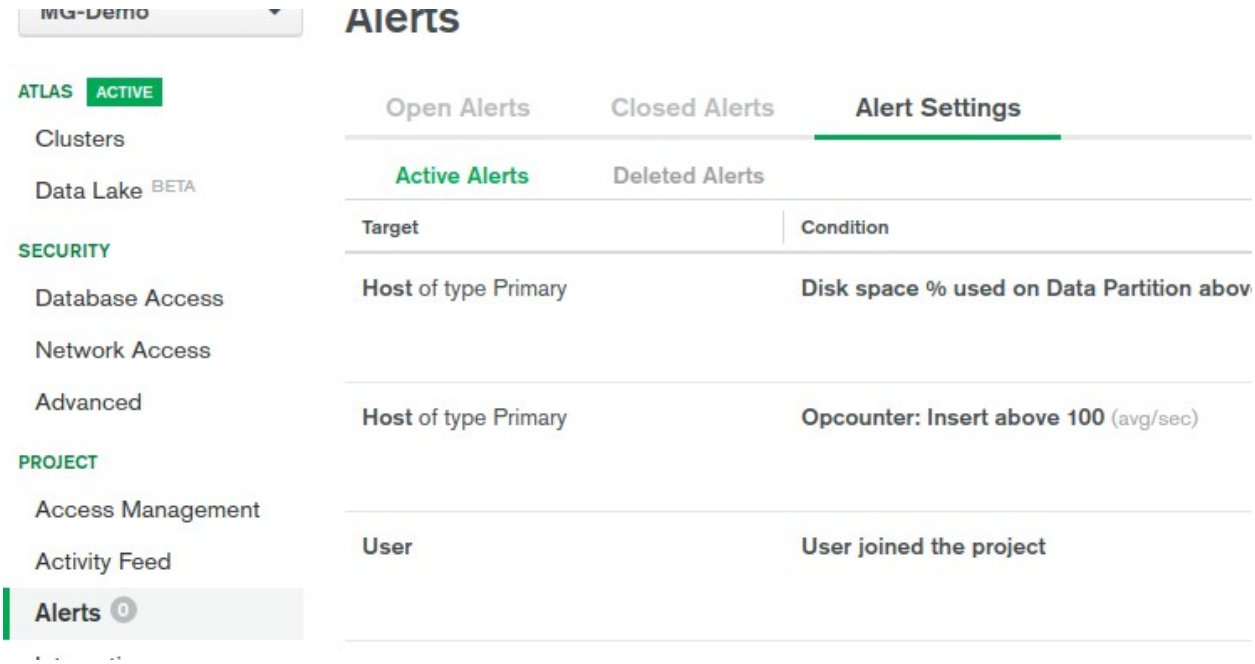
Finally, click ANALYZE again and click the **Documents** tab to view the Chinese restaurants in our selected radius in Staten Island:



Lab 7 - Configuring Alerts

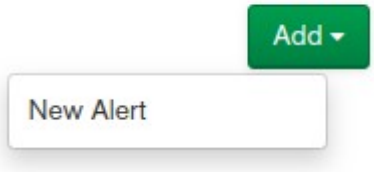
Atlas provides a system of alerts that allows users to keep track of a variety of different events that may take place within a cluster. For this lab, we'll set up an alert that will send us an email if the dataset exceeds a certain size.

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

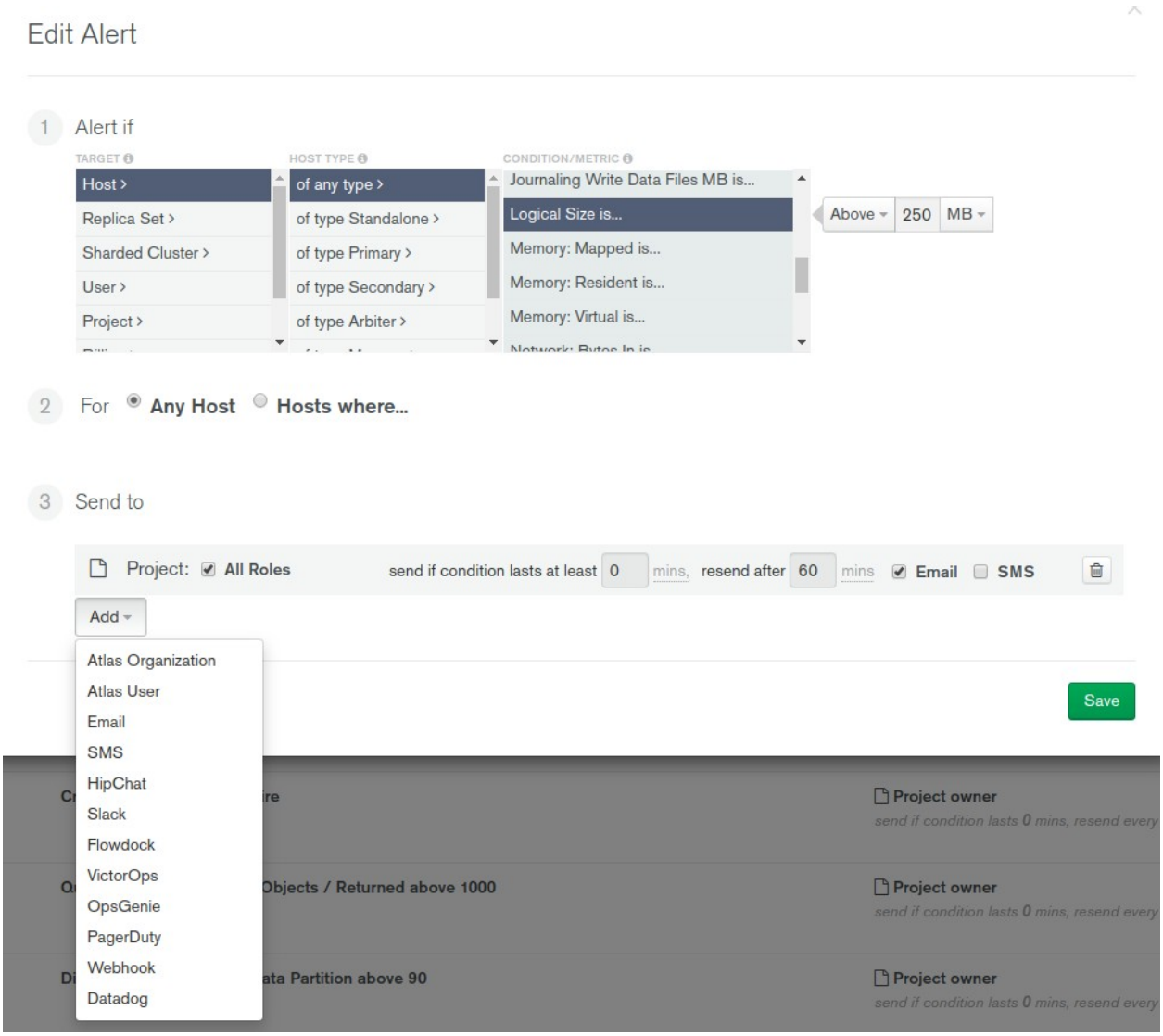


There are a bunch of alerts that are enabled by default helping us track disk usage, cpu usage, failover events, etc. Default alerts can be modified just like alerts that you create yourself.

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

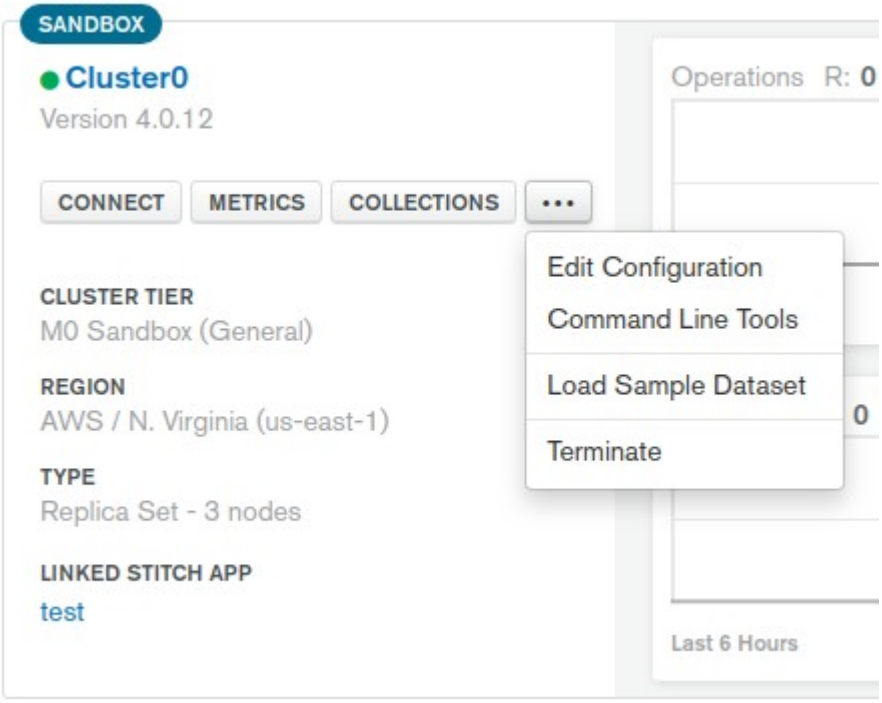


Set an alert so that if a host of any type has a logical data size greater than 250 MB, we’ll get an email.

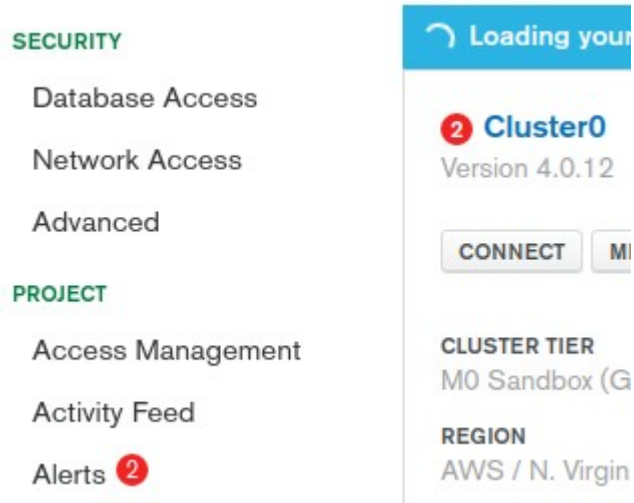

Edit Alert
1 Alert if
TARGET
HOST TYPE
CONDITION/METRIC
Host >
Replica Set >
Sharded Cluster >
User >
Project >
of any type >
of type Standalone >
of type Primary >
of type Secondary >
of type Arbiter >
Journaling Write Data Files MB is...
Logical Size is...
Memory: Mapped is...
Memory: Resident is...
Memory: Virtual is...
Above
250
MB
2 For Any Host Hosts where...
3 Send to
Project: All Roles
send if condition lasts at least 0 mins, resend after 60 mins
Email
SMS
Add
Atlas Organization
Atlas User
Email
SMS
HipChat
Slack
Flowdock
VictorOps
OpsGenie
PagerDuty
Webhook
Datadog
Save
Project owner
send if condition lasts 0 mins, resend every
Objects / Returned above 1000
ata Partition above 90

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”



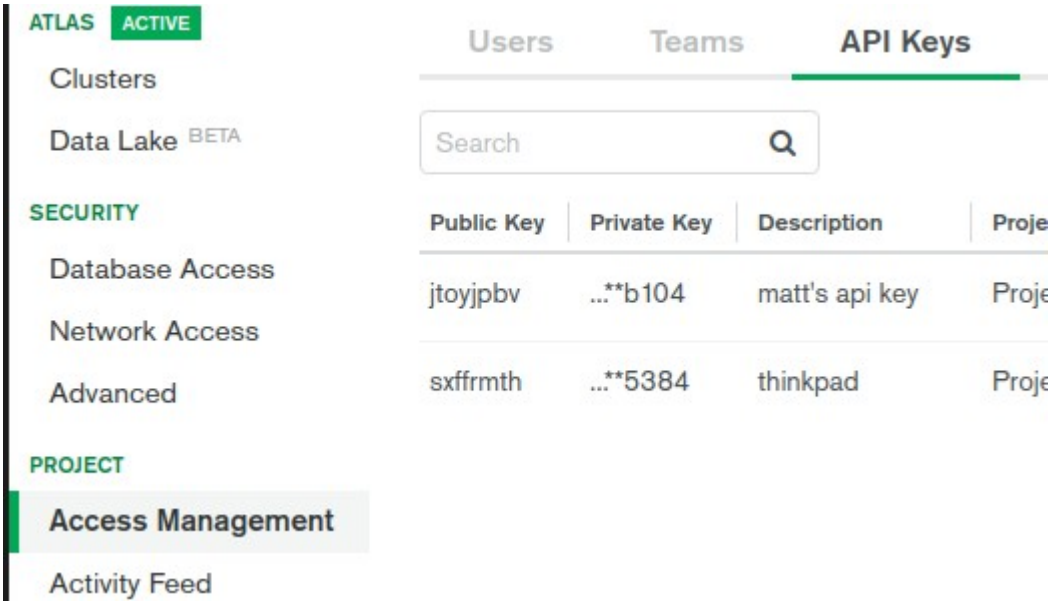
Once the dataset exceeds 250 MB, the alert should be triggered. This will be visible in the Atlas user interface and an email will be sent to the email address you used to create your Atlas account.



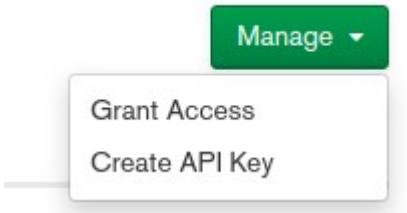
Lab 8 - Using the API

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.

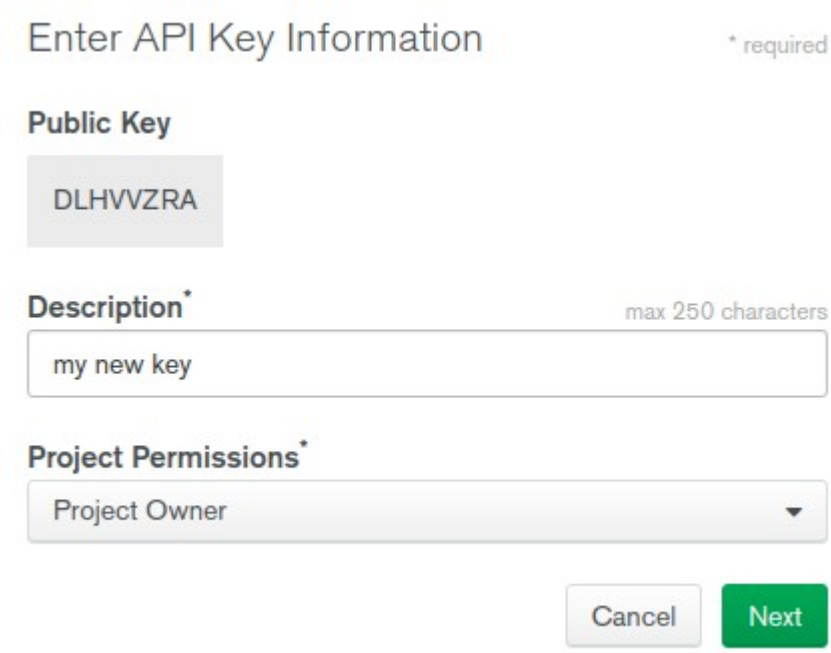
The first step to using the Atlas API is to generate an API key. Navigate to "Access Management" on the sidebar and choose "API Keys"



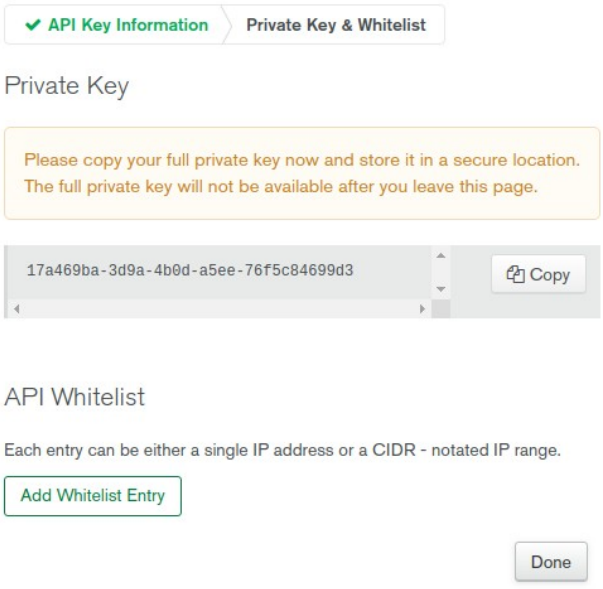
To create a key, click on "Manage" and select "Create API Key"



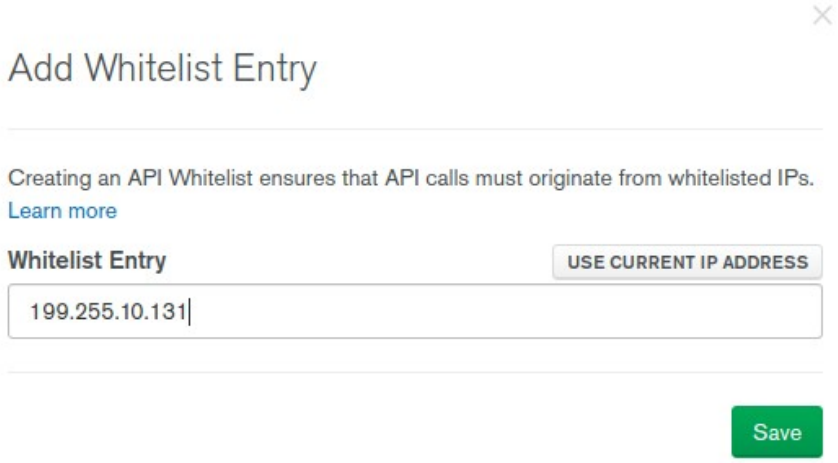
Provide a description of the key and make sure permissions include “Project Owner”



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



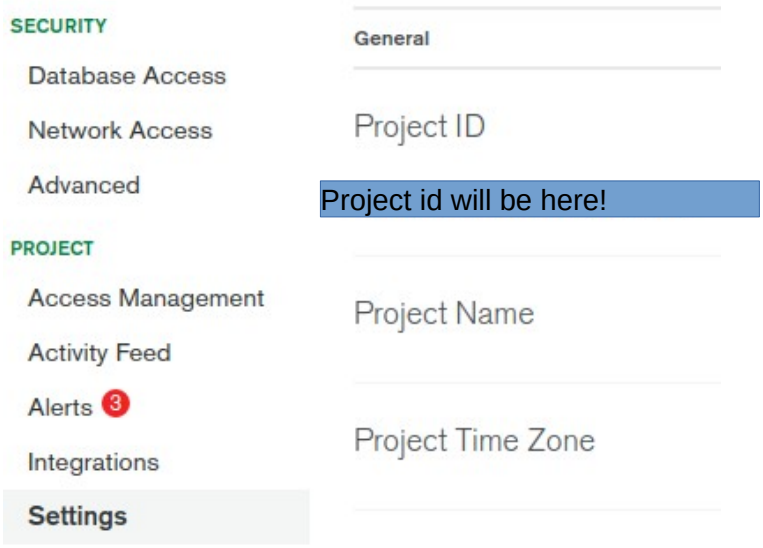
The next step is to whitelist your IP address for that particular API key. Click “Add Whitelist Entry” and add your public IP address.



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.



Assign the project id to an environemnt 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!