CONTENTS

RELATED

How To Install MongoDB on Ubuntu 12.04

View

How To Securely Configure a Production MongoDB Server

View

Tutorial Series: How To Manage Data with MongoDB

// Tutorial //

How To Use MongoDB Compass

Published on October 26, 2021 · Updated on October 26, 2021

Databases MongoDB NoSQL Applications

By Mateusz Papiernik
Software Engineer, CTO @Makimo

The author selected the Open Internet/Free Speech Fund to receive a donation as part of the Write for DOnations program.

Introduction

their data. To this end, the MongoDB project offers an official graphical user interface called *MongoDB Compass*.

With MongoDB Compass, sometimes shortened to *Compass*, you can access most of the features the MongoDB database engine offers through an intuitive visual display. You can glance through databases, collections and individual documents, interactively create queries, manipulate existing documents, and design aggregation pipelines through a dedicated interface.

In this tutorial, you'll install MongoDB Compass on a local machine and familiarize yourself with how to perform various database administration using the graphical tool.

Prerequisites

To follow this tutorial, you will need:

- A server with a regular, non-root user with `sudo` privileges and a firewall configured with UFW. This tutorial was validated using a server running Ubuntu 20.04, and you can prepare your server by following this initial server setup tutorial for Ubuntu 20.04.
- MongoDB installed on your server. To set this up, follow our tutorial on How to Install MongoDB on Ubuntu 20.04.
- Your server's MongoDB instance secured by enabling authentication and creating an administrative user. To secure MongoDB like this, follow our tutorial on How To Secure MongoDB on Ubuntu 20.04.
- Your MongoDB instance configured to allow connections from your local machine. Follow our guide on How To Configure Remote Access for MongoDB on Ubuntu 20.04 to set this up. As you follow this guide, **be sure to use your local machine in place of a second remote Ubuntu server**.
- A local machine on which you can install MongoDB Compass. This tutorial has instructions for how to install Compass on machines running Ubuntu and RHEL-based operating systems, but it also includes links to MongoDB's instructions for installing Compass on Windows and MacOS.

Note: The linked tutorials on how to configure your server, install, and then secure MongoDB installation refer to Ubuntu 20.04. This tutorial concentrates on MongoDB itself, not the underlying operating system. It will generally work with any MongoDB installation regardless of the operating system as long as authentication has been enabled and you've allowed

To use MongoDB Compass, you must install it on your local computer. MongoDB provides official packages for the graphical tool for Ubuntu and RHEL-based Linux distributions, as well as Windows and MacOS.

To find the appropriate package for your system, navigate to the MongoDB Compass Downloads page in your web browser. There, find the **Available Downloads** section on the right-hand side of the page and select your desired **Version** and **Platform** from the drop-down menus there. This tutorial's examples will install version `1.28.4`, the latest stable version at the time of this writing.

After making your choices, click the **Copy Link** button which will copy the download link to your clipboard. If you selected Ubuntu as your platform this link will download a `.deb` package, but if you selected RedHat the link will download a `.rpm` package.

Then, open up a terminal session **on your local machine**.

If your local machine is running Ubuntu and the link you copied is for a `.deb` package, run a `wget` command and pass the link you just copied to it as an argument. This will download the package to your working directory:

```
sammy@ubuntu wget https://downloads.mongodb.com/compass/mongodb-compass_1.28.4_
```
Copy de

Then install the `.deb` package with `apt`:

```
sammy@ubuntu sudo apt install ./mongodb-compass_1.28.4_amd64.deb
```
Copy

This command will install the Compass package along with all necessary dependencies.

If, however, you're using a RHEL-based distribution like CentOS, Fedora, or Rocky Linux, you can download and install the `.rpm` package directly from the web with a single command. Run the following `dnf` command which will also install Compass along with all of its dependencies:

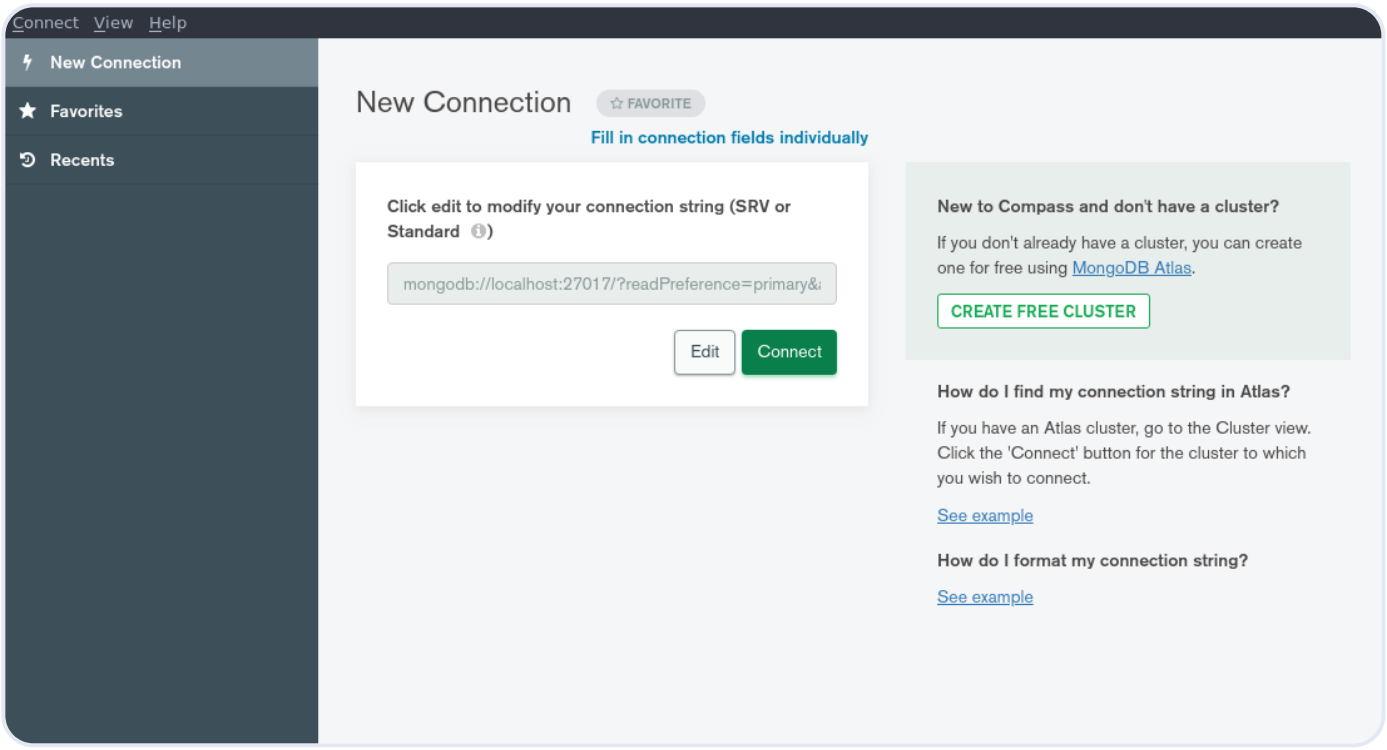
```
sammy@rhel sudo dnf install -y https://downloads.mongodb.com/compass/mongodb-co
```
Copy .

After installing the MongoDB Compass package, you can run the installed software by executing:

```
$ mongodb-compass
```

Copy

Compass will greet you with a welcome screen:



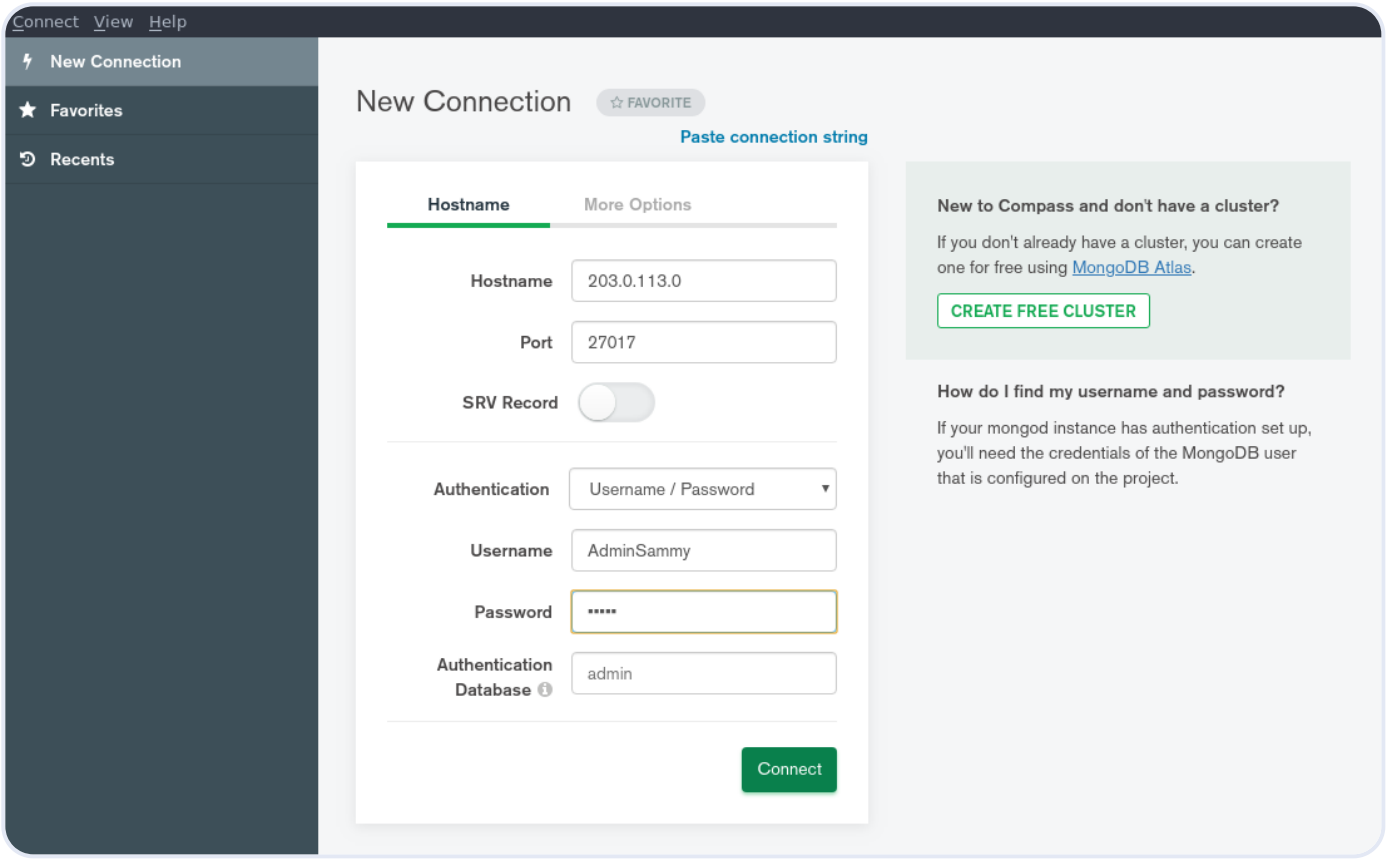
Now that you've installed MongoDB Compass on your local machine, you can connect it to the MongoDB instance running on your remote server.

Step 2 – Connecting to The MongoDB Server

To use MongoDB Compass with the MongoDB instance running on your remote server, you must first connect to it like you would if you were accessing the database through the shell. Assuming you completed the prerequisite tutorial on How To Configure Remote Access for MongoDB on Ubuntu 20.04, you will have already configured your MongoDB server to allow remote connections from your local machine.

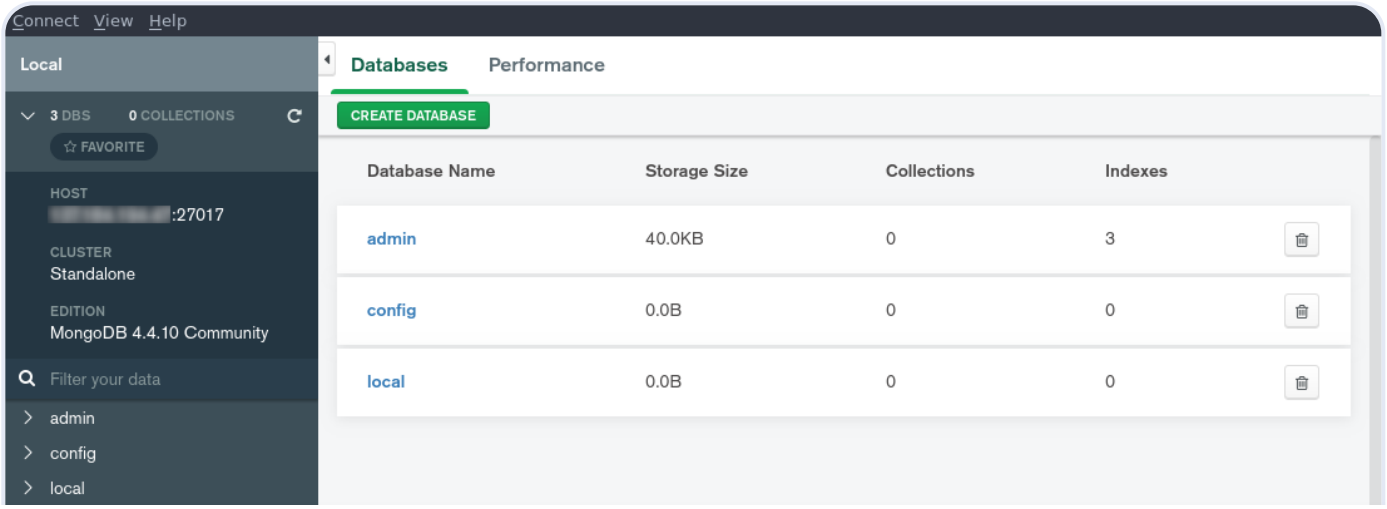
Compass allows you to connect using either a *connection string* — a single line of text

Click **Fill in connection fields individually** at the top of the welcome screen. The **New Connection** screen will appear with a list of empty fields:



Enter the IP address of the remote server on which your MongoDB instance is running into the **Hostname** field. Leave the default **Port** value unless you've changed the port on which your MongoDB instance is listening for connections. Because the server was secured with authentication enabled, you also need to switch the **Authentication** option to **Username / Password**. After selecting this option, enter your administrative MongoDB user's username, the password associated with this account, and this user's authentication database in the three new fields.

After clicking the **Connect** button, Compass will attempt to connect to the MongoDB instance. If it succeeds, you'll be taken to the Home screen showing the list of all the databases on the instance. They will also appear in the left panel along with high-level information like the database server's IP address and what version of MongoDB it's running:



If the connection attempt fails, make sure that you entered all the connection details correctly.

Once you've successfully connected your local Compass installation to your remote MongoDB server instance you can move on to creating a new test database and inserting test data into a new collection.

Step 3 – Preparing the Test Data

To illustrate the different features of MongoDB Compass, this guide will use a set of sample documents in its examples. This step involves creating a collection and inserting this set of sample data into it.

This sample collection contains documents representing the twenty most populated cities in the world. A sample document for Tokyo will follow this structure:

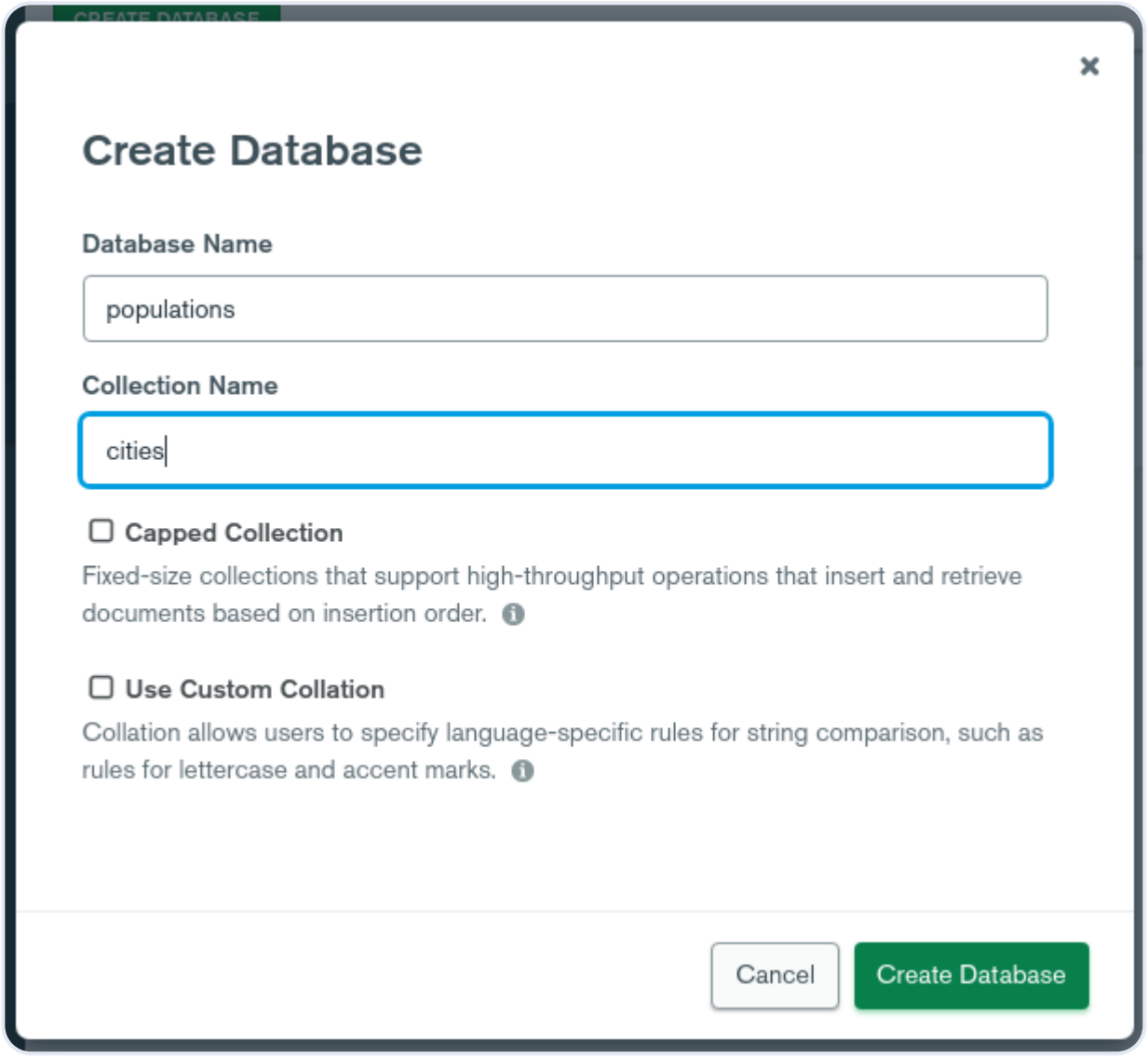
```
    "population": 37.400
}
```

The document contains information about the city's name, the country where it's located, the continent, and its population represented in millions. This guide will name the sample database `populations` and the collection which will store the documents will be named `cities`.

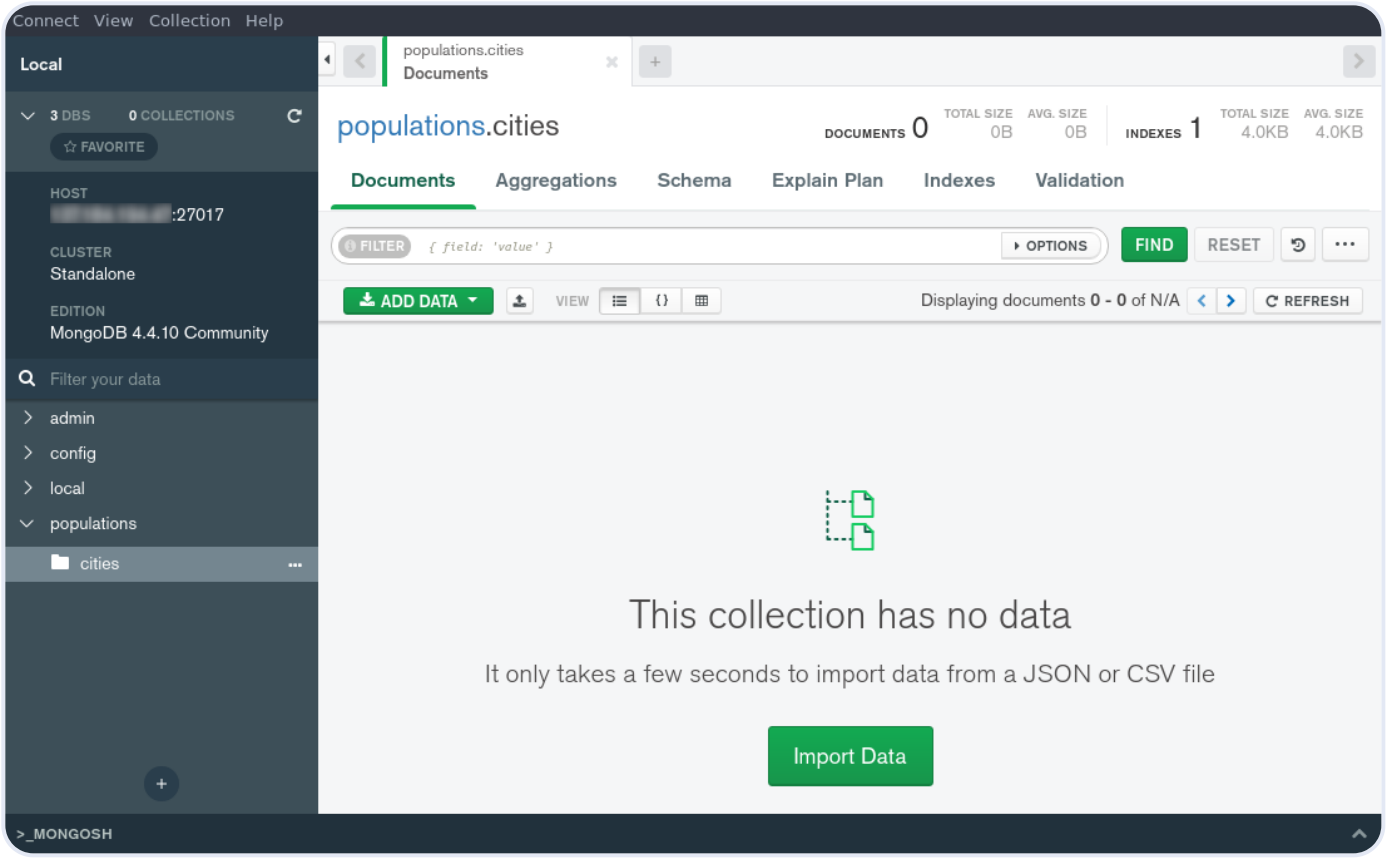
To begin, click the **CREATE DATABASE** button at the top of the Home screen. Alternatively, you can click the plus sign (**+**) at the bottom of the left panel.

In MongoDB, a database and collection are usually created when the first document is inserted into the collection without any need for an explicit creation operation for these structures. However, it is possible to create a new database explicitly and that's how you'll do it in MongoDB Compass in this tutorial.

Type `populations` into the **Database Name** field and `cities` into the **Collection Name** field, leaving all other fields with their default values and then click **Create Database**:



Compass will create the database. Click the `populations` database to reach the database view. Then, click `cities` to reach the empty collection view:



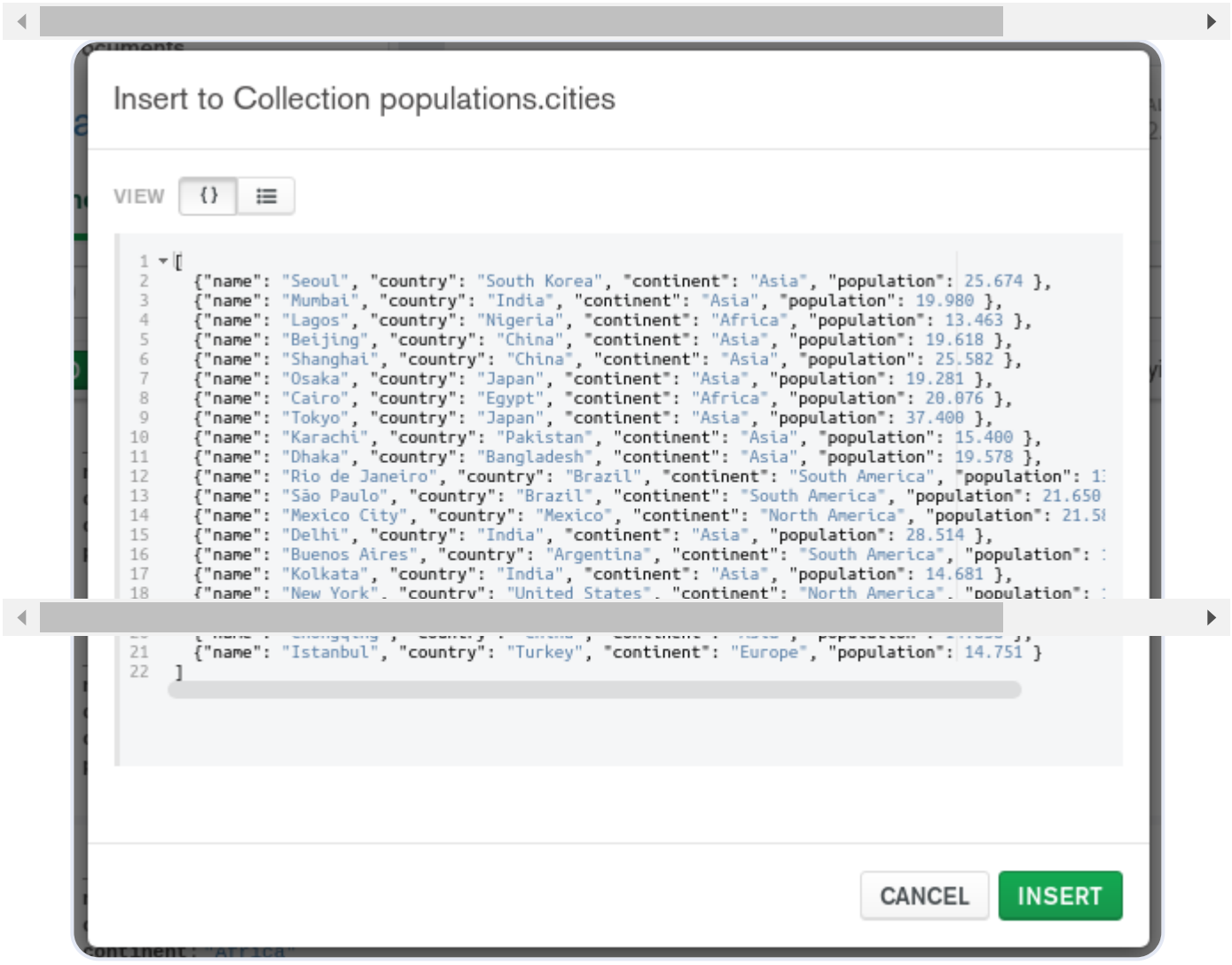
Now that the database and collection have been created, you can insert a list of unsorted documents into the `cities` collection. Click the **ADD DATA** button and then select the **Insert Document** option.

A window will appear in which you can enter one or more data documents, in JSON format, as you would do when using the shell command. Enter the following set of documents into the field, replacing any content that was there by default:

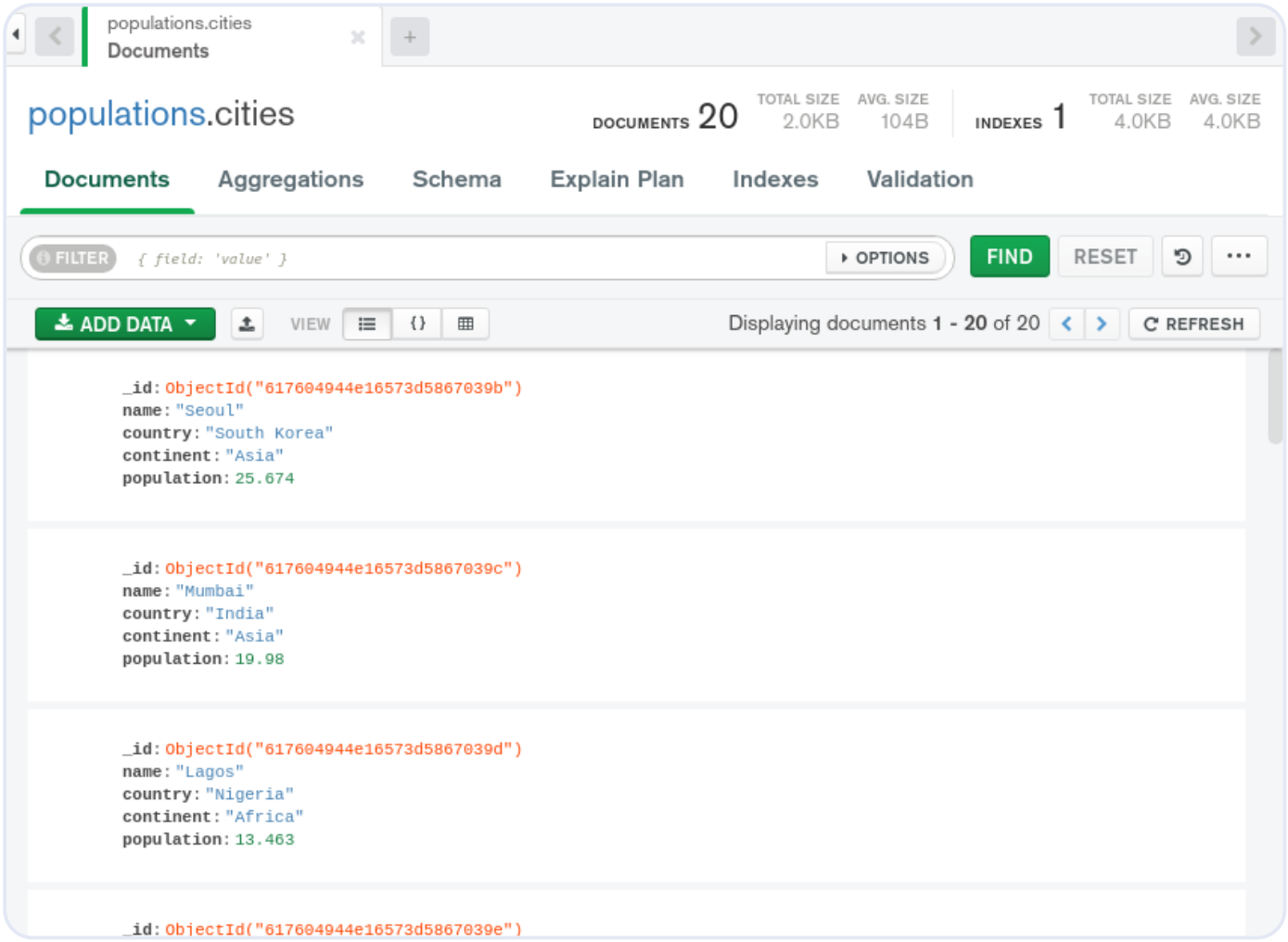
Copy

```
[
  {"name": "Seoul", "country": "South Korea", "continent": "Asia", "population": 25.674
  {"name": "Mumbai", "country": "India", "continent": "Asia", "population": 19.980 },
  {"name": "Lagos", "country": "Nigeria", "continent": "Africa", "population": 13.463 },
  {"name": "Beijing", "country": "China", "continent": "Asia", "population": 19.618 },
  {"name": "Shanghai", "country": "China", "continent": "Asia", "population": 25.582 },
  {"name": "Osaka", "country": "Japan", "continent": "Asia", "population": 19.281 },
  {"name": "Cairo", "country": "Egypt", "continent": "Africa", "population": 20.076 },
  {"name": "Tokyo", "country": "Japan", "continent": "Asia", "population": 37.400 },
  {"name": "Karachi", "country": "Pakistan", "continent": "Asia", "population": 15.400 }
```

```
  {"name": "Kolkata", "country": "India", "continent": "Asia", "population": 14.681 },
  {"name": "New York", "country": "United States", "continent": "North America", "popula
  {"name": "Manila", "country": "Philippines", "continent": "Asia", "population": 13.482
  {"name": "Chongqing", "country": "China", "continent": "Asia", "population": 14.838 },
  {"name": "Istanbul", "country": "Turkey", "continent": "Europe", "population": 14.751
]
```



Click the **INSERT** button, and Compass will insert the list of documents and then automatically display them in the collection browser:



With that, you've successfully created a set of sample documents representing the world's most populated cities. This collection will serve as this guide's example data as you explore MongoDB Compass. Next, you'll learn how to browse data with the graphical interface.

Step 4 – Navigating and Filtering the Data

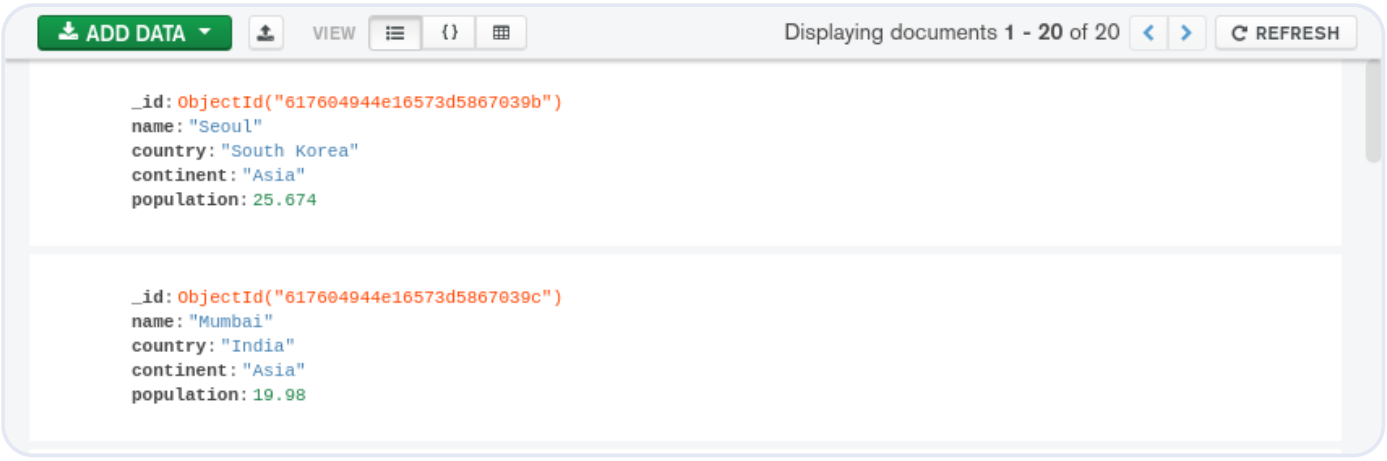
MongoDB Compass is a convenient tool for browsing through the data stored in a MongoDB database through a graphical interface. It removes the burden of having to remember the names of obscure databases or collections, and you can navigate to any database or collection on your MongoDB server with just a few clicks.

The primary navigation tool in Compass is the left panel, which works like a tree showing the database's contents. The top-level nodes are the databases, which you can click on to

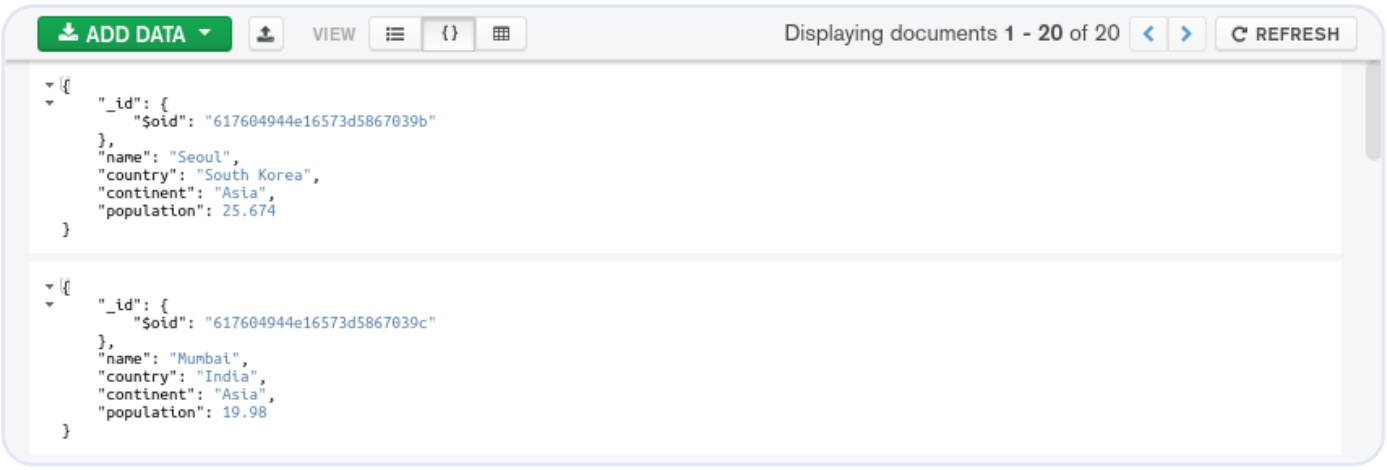
preparing the test data, a single collection will be available. By clicking on the collection name, you'll load the data browser screen.

By default, Compass will show the first 20 unfiltered results returned by an empty query on the selected collection. To the right off the **ADD DATA** button you used in the previous step, you'll find a **View** section with three separate display modes you can choose from:

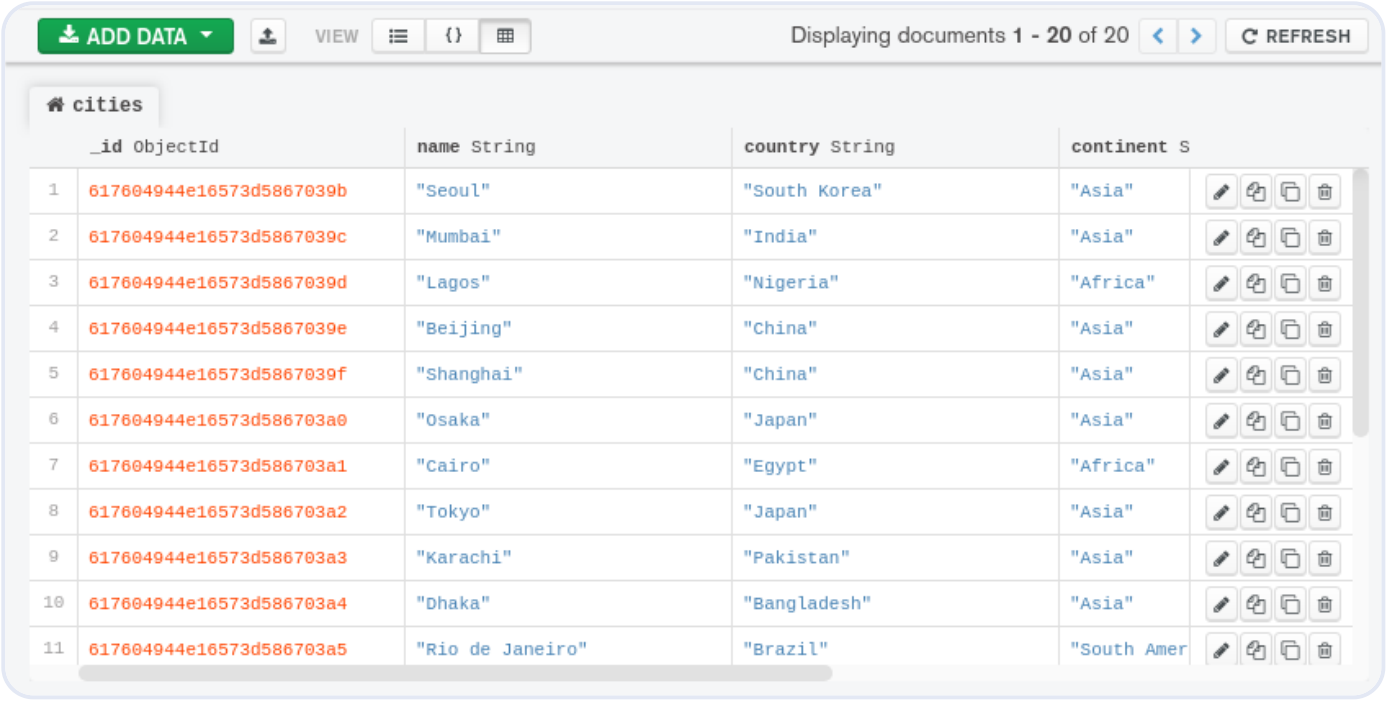
- **List view**: the default view showing documents as key-value pairs shown one in a row. This display mode resembles the JSON document format, but its syntax coloring and additional interface features, such as collapsible nested documents, help make it more readable:



- **JSON view**: this view shows the actual document structure as represented in JSON:



- **Table view**: showing the data in a tabular interface, similar to that found in relational databases. This can be handy if the documents follow a well-defined flat structure but



	_id ObjectId	name String	country String	continent S
1	617604944e16573d5867039b	"Seoul"	"South Korea"	"Asia"
2	617604944e16573d5867039c	"Mumbai"	"India"	"Asia"
3	617604944e16573d5867039d	"Lagos"	"Nigeria"	"Africa"
4	617604944e16573d5867039e	"Beijing"	"China"	"Asia"
5	617604944e16573d5867039f	"Shanghai"	"China"	"Asia"
6	617604944e16573d586703a0	"Osaka"	"Japan"	"Asia"
7	617604944e16573d586703a1	"Cairo"	"Egypt"	"Africa"
8	617604944e16573d586703a2	"Tokyo"	"Japan"	"Asia"
9	617604944e16573d586703a3	"Karachi"	"Pakistan"	"Asia"
10	617604944e16573d586703a4	"Dhaka"	"Bangladesh"	"Asia"
11	617604944e16573d586703a5	"Rio de Janeiro"	"Brazil"	"South Amer

Regardless of what view you use, you can use the data browser screen to quickly query your data just like you would with the `find()` method in the MongoDB shell. As an example, you might query your collection to find all the documents representing cities in North America by running the following operation in the MongoDB shell:

```
> db.cities.find({ "continent": "North America" })
```

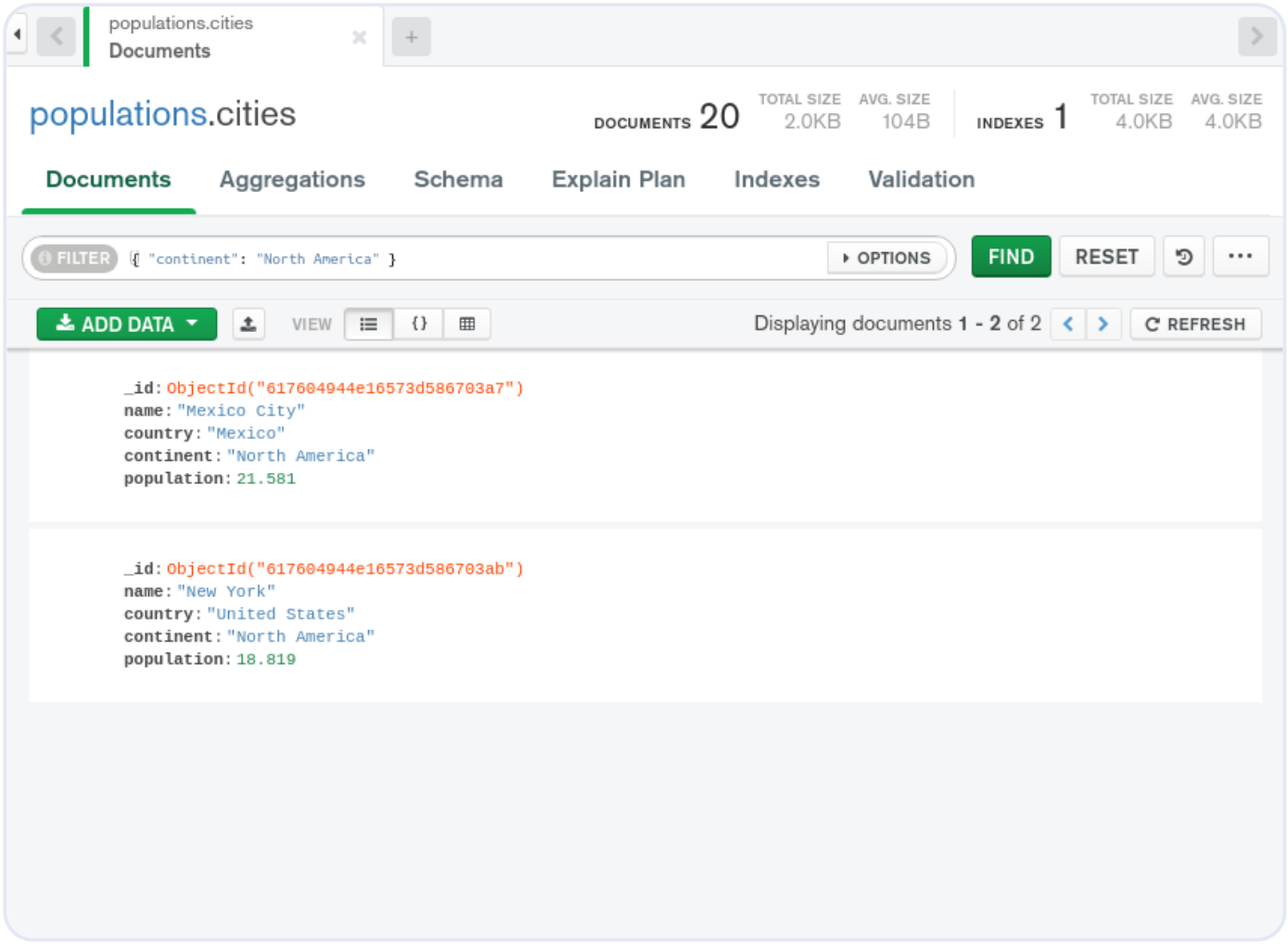
Copy

In this `find()` method, `{ "continent": "North America" }` is the *query document*. This is the part of the command that tells MongoDB how to filter the data. Likewise, you can enter this, or any other query document, into MongoDB Compass's **FILTER** field at the top of the collection window to filter your data.

Go ahead and enter this query document into the **FILTER** field, and then press **FIND**:

```
{ "continent": "North America" }
```

Copy



MongoDB Compass will narrow the list of documents to the two entries matching the filtering criteria. You can use any valid query document that you would use in a `find()` command to filter date in the **FILTER** field. If you make a syntax error, Compass will turn the **FILTER** badge to red, indicating there is a problem with the query.

You can also sort the results and apply projections to return only a limited subset of fields using the data browser interface. Click on the **OPTIONS** button near the filtering query bar to reveal further options. The **PROJECT** and **SORT** fields will appear below the **FILTER** field. A number of other fields will also appear, but it would be beyond the scope of this guide to discuss all of them in detail.

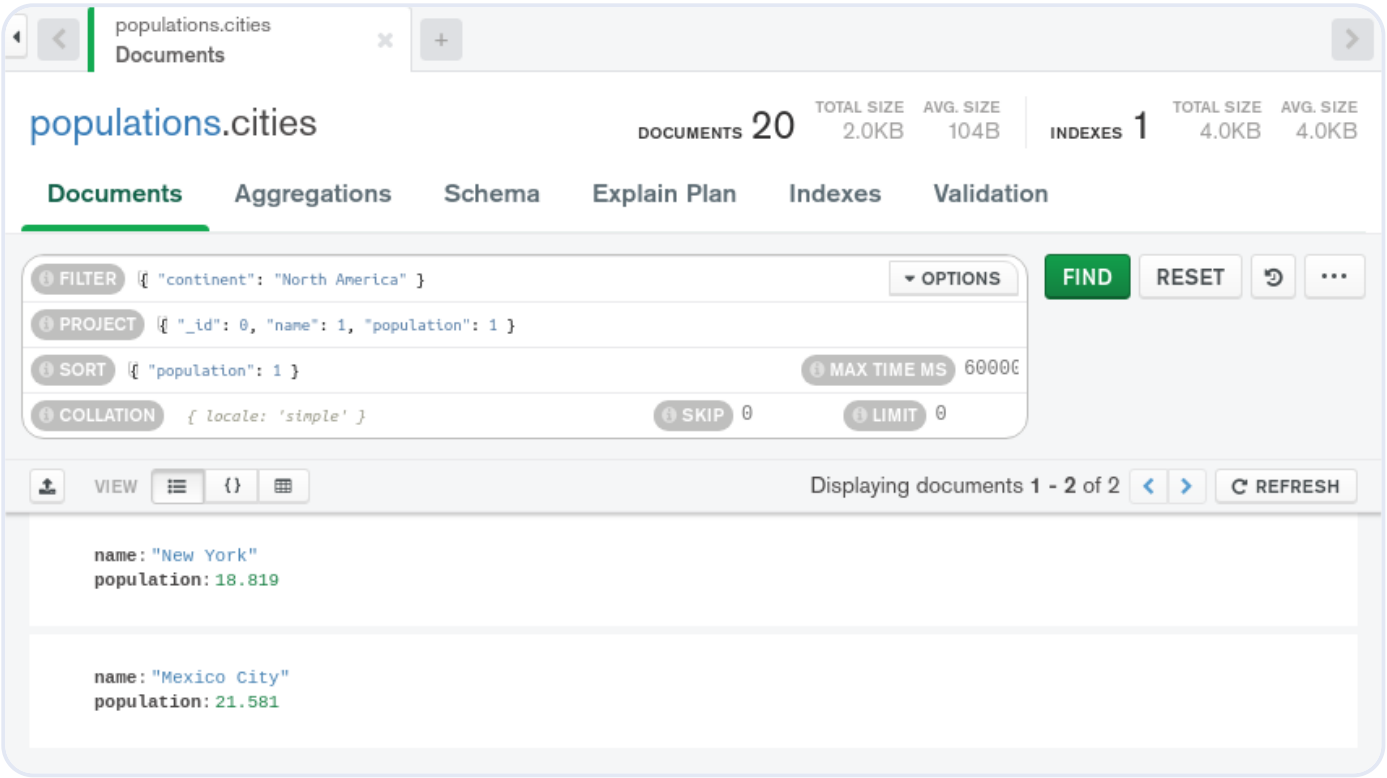
Both the **PROJECT** and **SORT** fields accept the same documents you would pass to the `find()` and `sort()` methods in the shell. As an example, try limiting the returned fields to

```
{ "_id": 0, "name": 1, "population": 1 }
```
Copy

To sort the list by population in ascending order, add the following sort document to the **SORT** field:

```
{ "population": 1 }
```
Copy

Click the **FIND** button again to apply these projection and sorting documents:

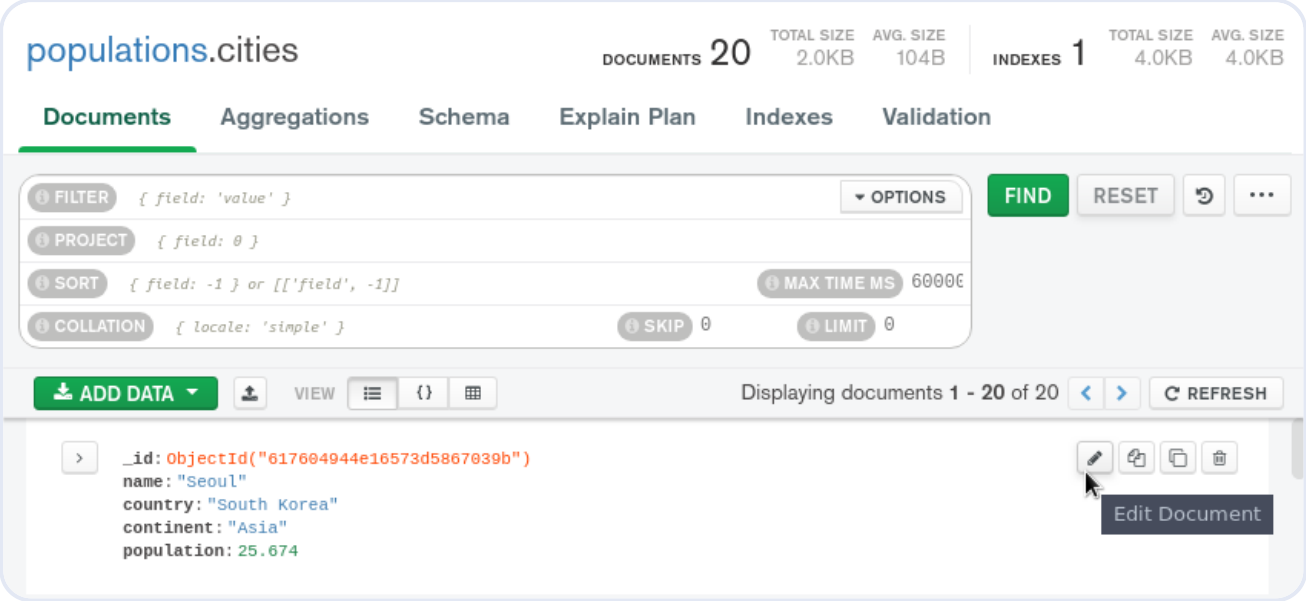


Compass now shows two simplified documents representing New York and Mexico City. The results are equivalent to running the following query in the MongoDB shell:

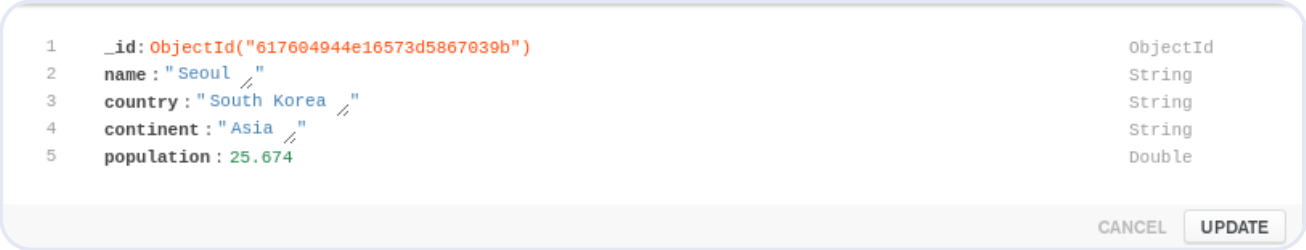
```
> db.cities.find(
>     { "continent": "North America" },
>     { "_id": 0, "name": 1, "population": 1 }
> ).sort(
>     { "population": 1 }
> )
```
Copy

To clear the filter, projection, and sort documents you've applied, you can click the **RESET** button.

Note: Using the data browser, you can also modify an individual document's contents. From the List view, hover your cursor over a document in the list and clicking on the pencil icon within the contextual menu that appears:



The static display will change into the list of editable fields that you can freely modify to quickly alter the document:



After making your desired changes, click the **UPDATE** button and they will be written to the database.

Step 5 – Using the Interactive Aggregation Pipeline Builder

> **Note:** The example in this step follows the aggregation pipeline from the How to Use Aggregations tutorial that uses similar test data. If you are not familiar with aggregation pipelines, you can follow that tutorial to learn about their principles first.

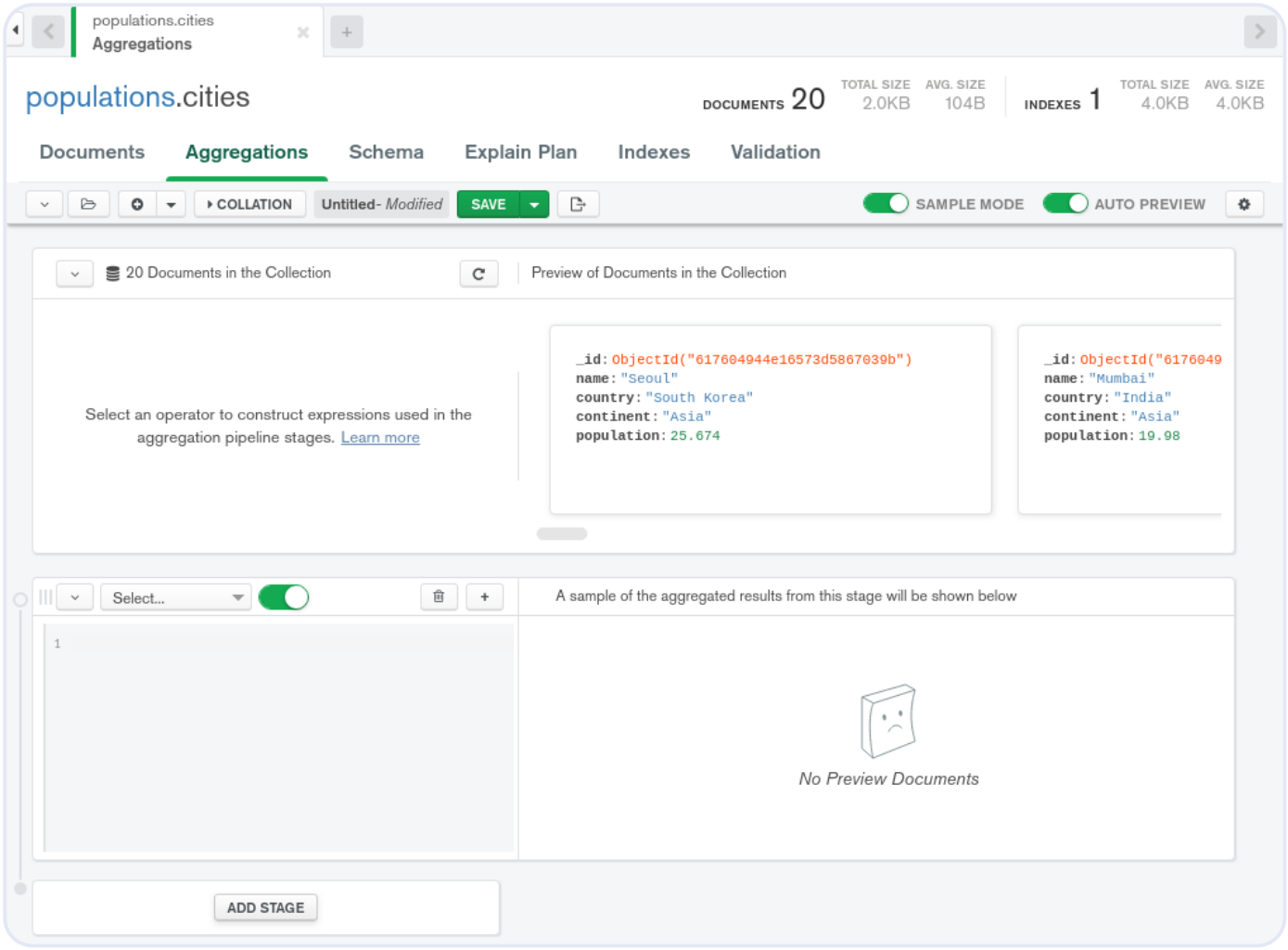
As an example, say that the task at hand is to list the most populated cities of countries represented in the collection, but only those found in Asia and North America. The pipeline should only return the cities' names and populations. The results should be sorted in descending order by population, returning countries with the largest cities first. Lastly, the documents' structure should replicate the following:

Example document structure

```
{
    "location" : {
        "country" : "Japan",
        "continent" : "Asia"
    },
    "most_populated_city" : {
        "name" : "Tokyo",
        "population" : 37.4
    }
}
```

To start building an aggregation pipeline that will satisfy these requirements, open up the **Aggregations** tab for the `cities` collection.

Begin by filtering the initial documents coming from the `cities` collection to contain only countries in Asia and North America. Open the **Aggregations** tab for the `cities` collection:



The top section of the aggregation pipeline builder view shows the source documents in the collection. This allows you to quickly glance through the documents that will be used as input for the aggregation pipeline. Note that if the documents don't immediately appear, you may need to press the refresh button (⟳) to make these documents appear if they aren't there by default.

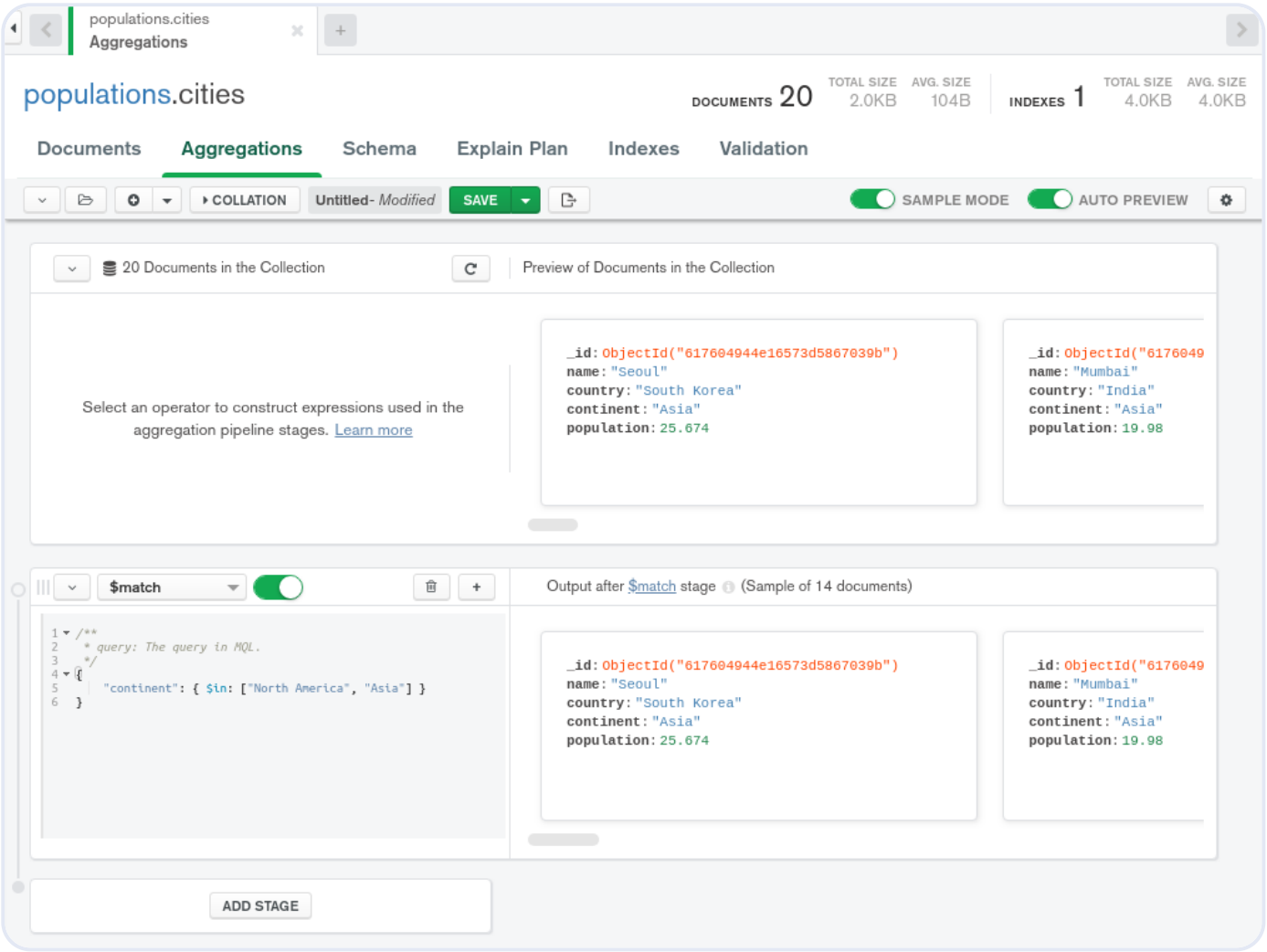
In the case of this example these are the `cities` documents with their original, unaltered structure. The second row, initially empty, is the first aggregation stage.

The first step to building this example pipeline is to filter the initial documents coming from the `cities` collection so they only contain documents representing cities in North America and Asia. To this end, use the drop-down **Select...** menu and choose the `$match` stage. MongoDB Compass will provide you with an example showing the syntax it expects. The

```
{
    "continent": { $in: ["North America", "Asia"] }
}
```

Copy

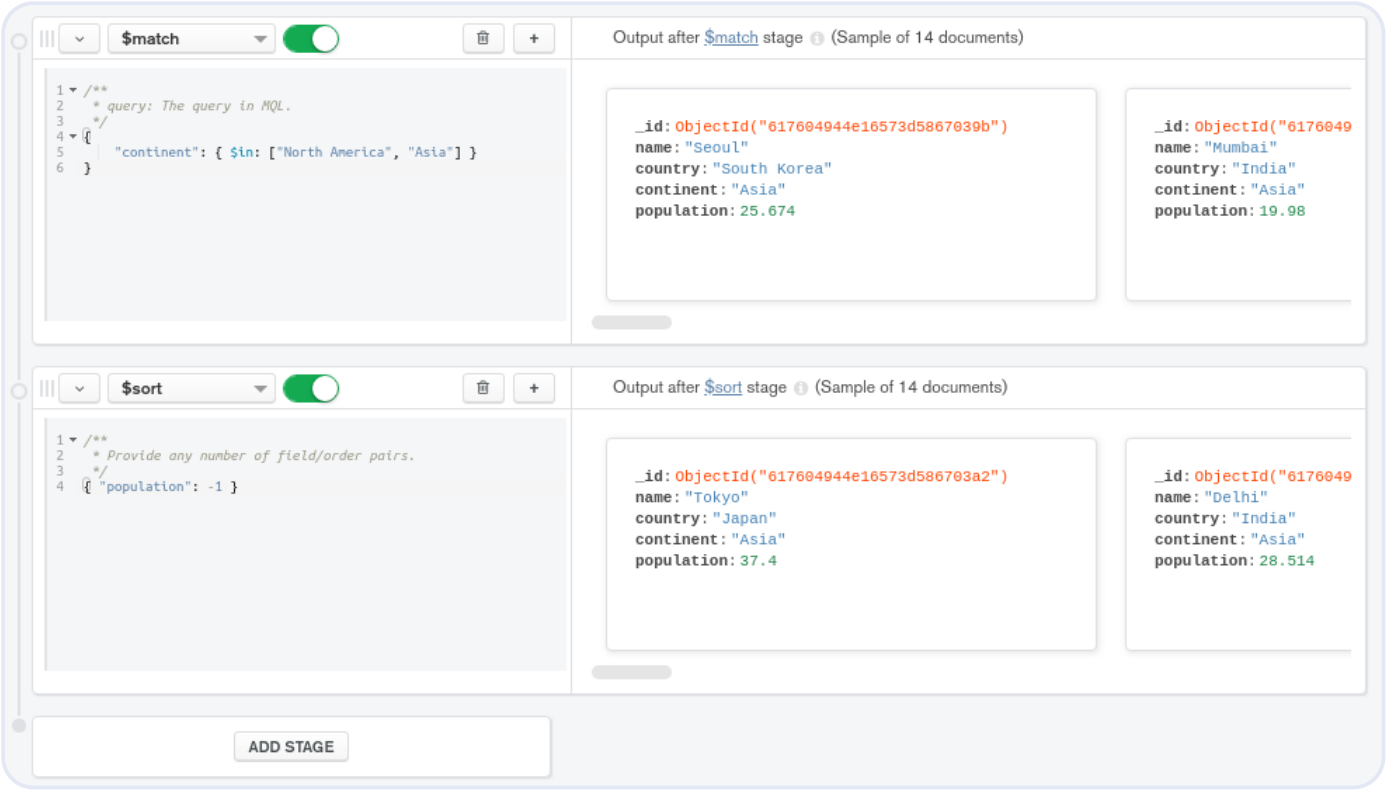
Compass will automatically preview the results of applying the first stage, displaying the filtered documents on the right. This way, you can quickly verify that each stage works as intended:



The second step is to order the documents by population in descending order. Add an additional stage using the **ADD STAGE** button. Another empty stage row will appear. There, select the `$sort` stage and enter the following as the stage settings:

```
{ "population": -1 }
```

Copy



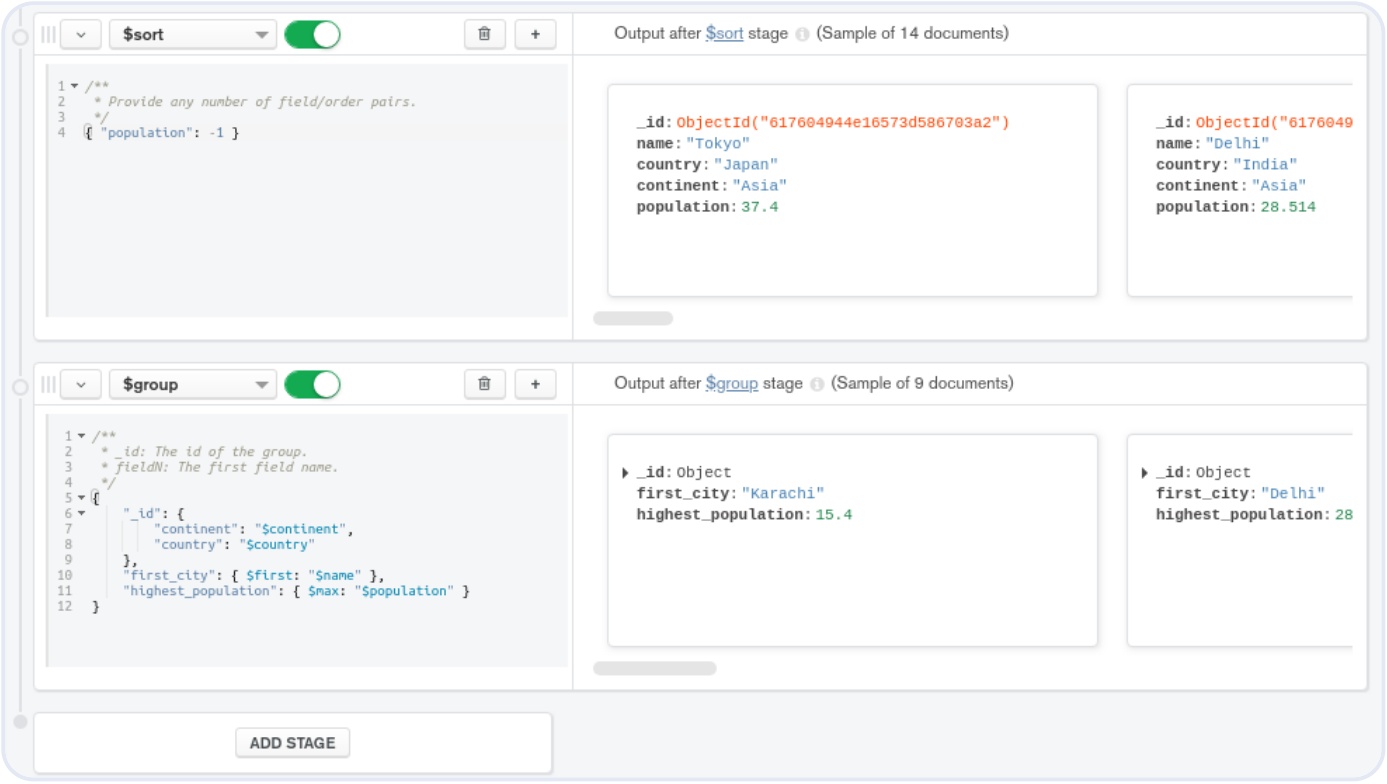
Since the list of cities is now sorted by the population coming from the expected continents, the next necessary step is to group cities by their countries, choosing only the most populated city from each group. Add another stage, this time selecting `$group` as the stage type.

To group cities by unique continent and country pairs and summarize these pairs by only showing the name and population of their most populated cities, enter the following grouping settings:

```
{
    "_id": {
        "continent": "$continent",
        "country": "$country"
    },
    "first_city": { $first: "$name" },
    "highest_population": { $max: "$population" }
}
```

Copy

The `highest_population` value uses the `$max` accumulator operator to find the highest

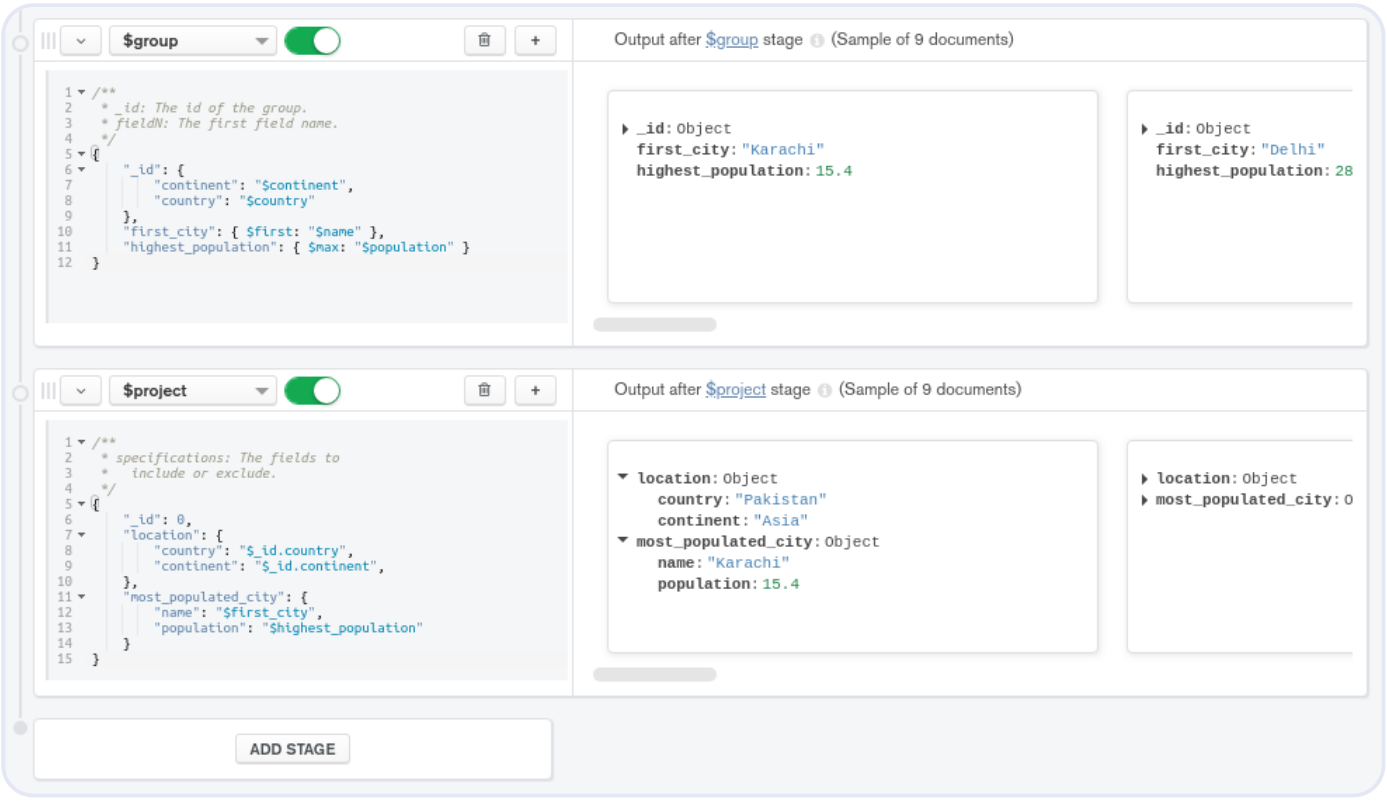


After applying the grouping stage, the list of documents is reduced to 9 entries, but the document structure is now different than before. Compass previews the output from the group stage on the right, showing the newly computed `highest_population` and `first_city` values as well as the grouping expression value in the `_id` field.

The last step to satisfying the requirements described at the beginning of this section is to transform the document's structure. You can do that by adding another stage row and selecting `$project` as the stage. To achieve the specified document structure, enter the following projection document:

```
{                                                                    Copy
    "_id": 0,
    "location": {
        "country": "$_id.country",
        "continent": "$_id.continent",
    },
    "most_populated_city": {
        "name": "$first_city",
        "population": "$highest_population"
```

`continent`. Each of these refers to the values from the input document. `most_populated_city` follows a similar principle, nesting the `name` and `population` fields inside. Both of these fields refer to the top-level fields `first_city` and `highest_population`. This projection stage effectively constructs an entirely new structure for the output:



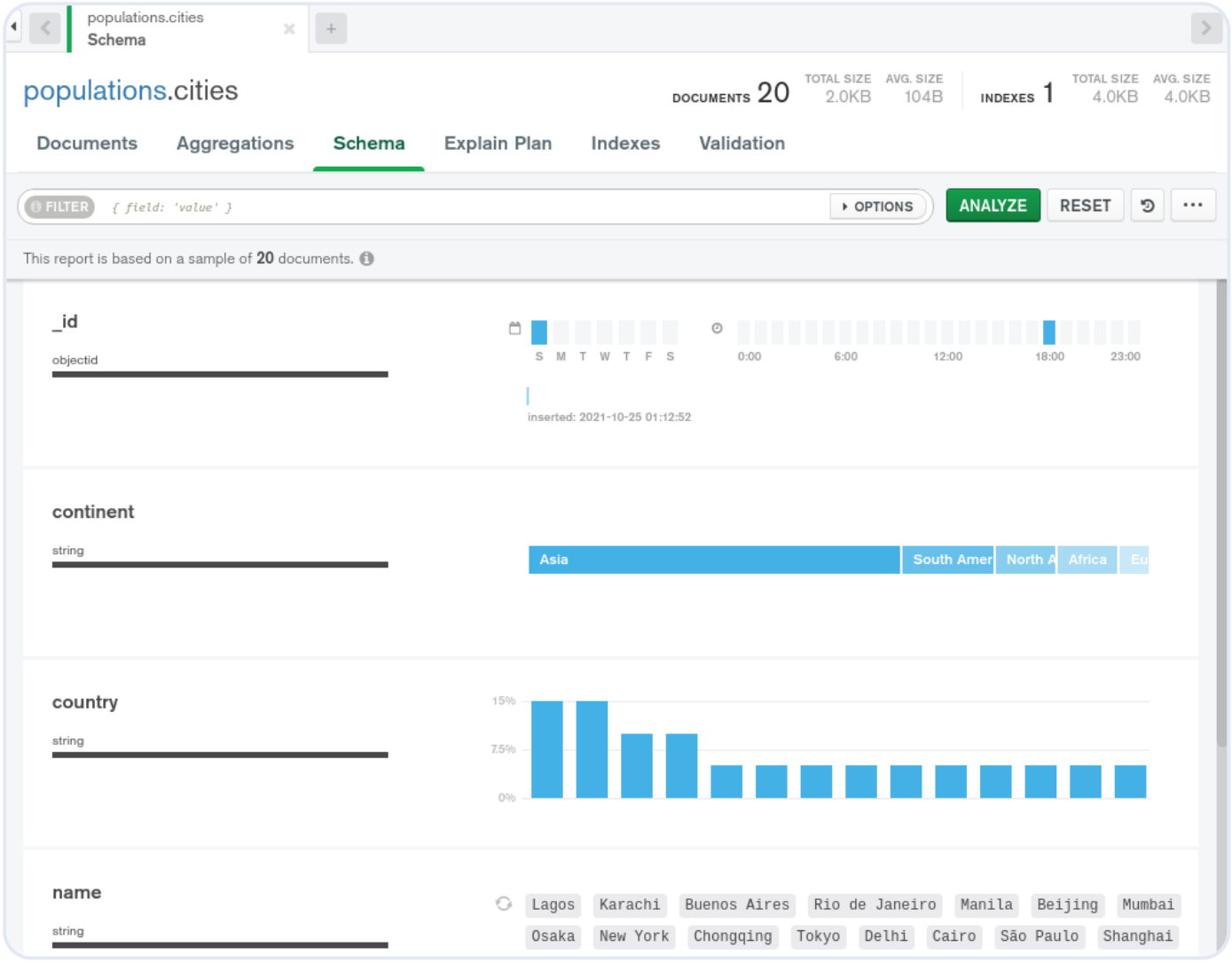
Notice that the preview pane on the right for this aggregation stage shows the sample output after transformation. With this, you can quickly confirm the projection stage resulted in the expected transformation.

By using the aggregation pipeline builder, you can conveniently build aggregations stage by stage without worrying about the complex syntax of a single aggregation command. Toggling on the **SAMPLE MODE** switch ensures that MongoDB Compass will use only a subset of input documents, which makes it quicker to develop pipelines when the source data set is large.

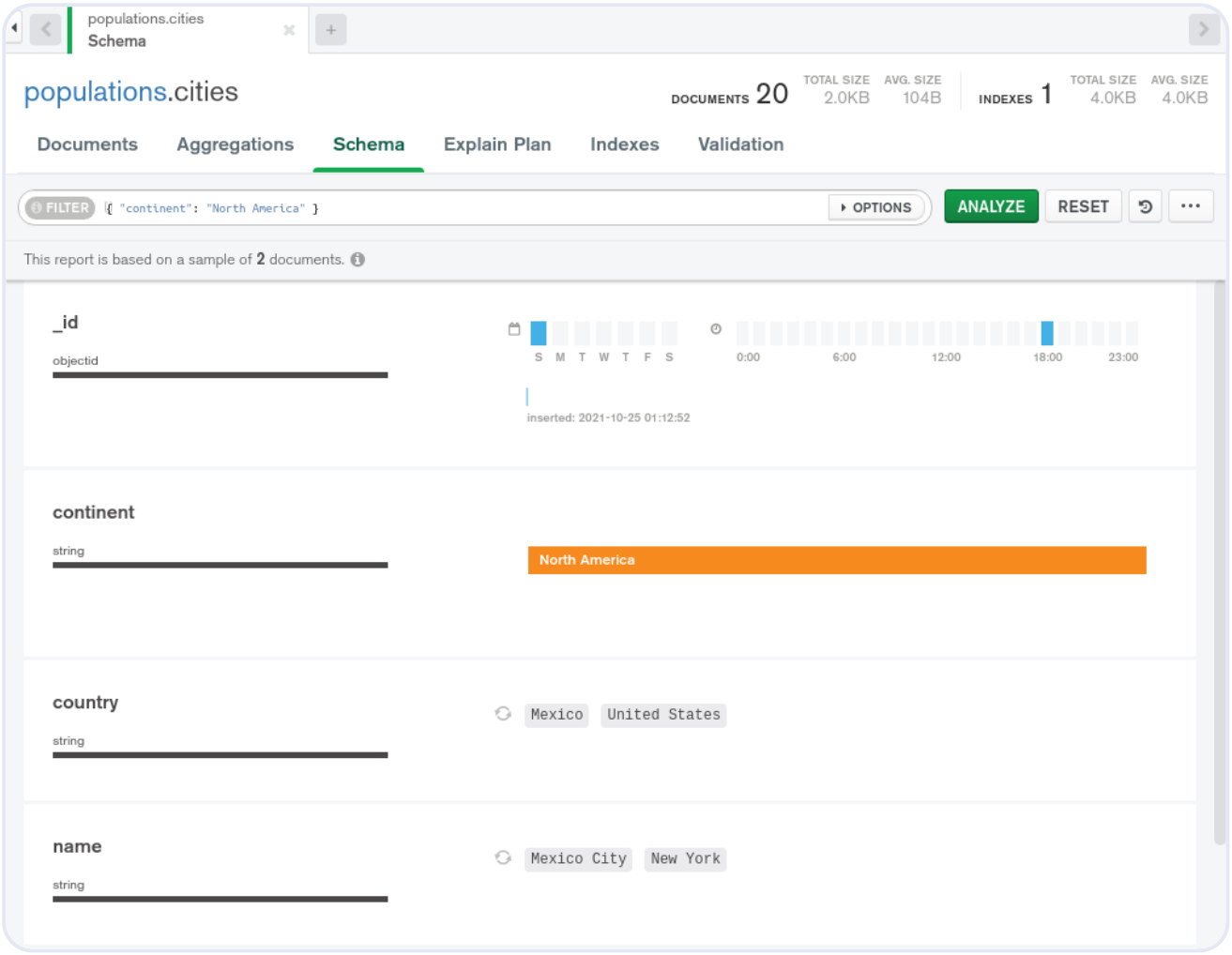
Thanks to the preview pane for each of the stages, you can verify that each stage provides expected results. You can also disable and enable individual stages with the flip-switches to understand how the aggregation pipeline will work without some of the stages activated.

functions in MongoDB, but in this step you'll explore a feature that's unique to Compass: its schema visualizer interface. This tool can help you understand the structure of data within your collections.

To use it, first select the **Schema** tab in the `cities` collection view. The view will initially be empty, but when you press **Analyze** button, Compass will churn the data to reveal insights about its form, size, and contents:



Note: Similarly to just browsing the data, you can use filter query documents to narrow down the selection and force MongoDB Compass to provide insights on a subset of collection documents:



For each of the document fields, the schema visualizer will provide insights into the data found in the database.

For instance, notice the `_id` field within the schema visualizer. MongoDB requires that every document has an `_id` field to be used as a primary key. In Compass's schema visualizer, it shows when documents were inserted into the database. In this example, all the documents were inserted at a single point of time on Sunday evening. With a living database, however, the entries will be spread throughout the course of the database's use. This knowledge could be helpful for gauging how much the database is used throughout the course of a typical week.

For the `continent` and `country` fields, which both contain string values but with values that

The `name` field is also a string value field, but this time each one is unique. In this case, Compass shows a sample set of values.

By combining Compass's schema visualizer with MongoDB's filtering capabilities you can quickly scan your data and the generated visualizations, allowing you to analyze your data without having to write out complex queries. This comes in handy not only to visualize the contents of the database, but also to understand the data to aid decisions about creating indexes or sharded clusters.

Conclusion

In this article, you familiarized yourself with MongoDB Compass, a GUI that allows you to manage your MongoDB data through a convenient visual display. You used the tool to create a new collection, insert new documents, filter and navigate the data, create a multi-stage aggregation pipeline, and visualize the collection's schema using the schema visualization tool.

The tutorial described only a subset of MongoDB Compass features, which encompass many other capabilities related to MongoDB administration, such as index management, query execution plan visualization, and a view of the server's real-time performance metrics. To learn more about how Compass can help you, we encourage you to study the official official MongoDB Compass documentation.

Next in series: How To Use Transactions in MongoDB →

Tutorial Series: How To Manage Data with MongoDB

MongoDB is a document-oriented NoSQL database management system (DBMS). Unlike traditional relational DBMSs, which store data in tables consisting of rows and columns, MongoDB stores data in JSON-like structures referred to as *documents*.

This series provides an overview of MongoDB's features and how you can use them to manage and interact with your data.

Subscribe

Databases MongoDB NoSQL Applications

Browse Series: 16 articles

Expand to view all

Mateusz Papiernik Author

Software Engineer, CTO @Makimo

Creating bespoke software ◦ CTO & co-founder at Makimo. I'm a software enginner & a geek. I like making impossible things possible. And I need tea.

Mark Drake Editor

Manager, Developer Education

Technical Writer @ DigitalOcean

Still looking for an answer?

Ask a question | Search for more help

Was this helpful? Yes No

Comments

Leave a comment

Leave a comment...

This textbox defaults to using **Markdown** to format your answer.

You can type **!ref** in this text area to quickly search our full set of tutorials, documentation & marketplace offerings and insert the link!

Sign In or Sign Up to Comment

Popular Topics

Ubuntu

Python

Security

MySQL

Docker

Kubernetes

Browse all topic tags

Free Managed Hosting →

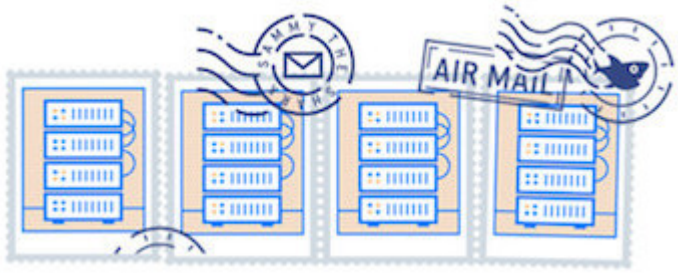
All tutorials →

Questions

Q&A Forum

Ask a question

DigitalOcean Support



GET OUR BIWEEKLY NEWSLETTER

Sign up for Infrastructure as a Newsletter.

Uptime