

MongoDB Atlas Free + Google Cloud Run

CS351 | Illustrated student setup and database-management guide

Understand the two separate services before starting. MongoDB Atlas is your database hosting provider. When you select Google Cloud in Atlas, MongoDB provisions and manages the database on Google Cloud infrastructure controlled by MongoDB - **not in your own Google Cloud project**. You manage this database entirely through your MongoDB Atlas account; the Atlas Free cluster does not use your Google Cloud credits or require a Google Cloud login.

Separately, you will log into your own Google Cloud account to deploy a Node.js/Express web application on **Cloud Run**. The application connects to Atlas using a MongoDB connection URI. Eligible Google Cloud educational credits can cover Cloud Run charges, subject to program rules.

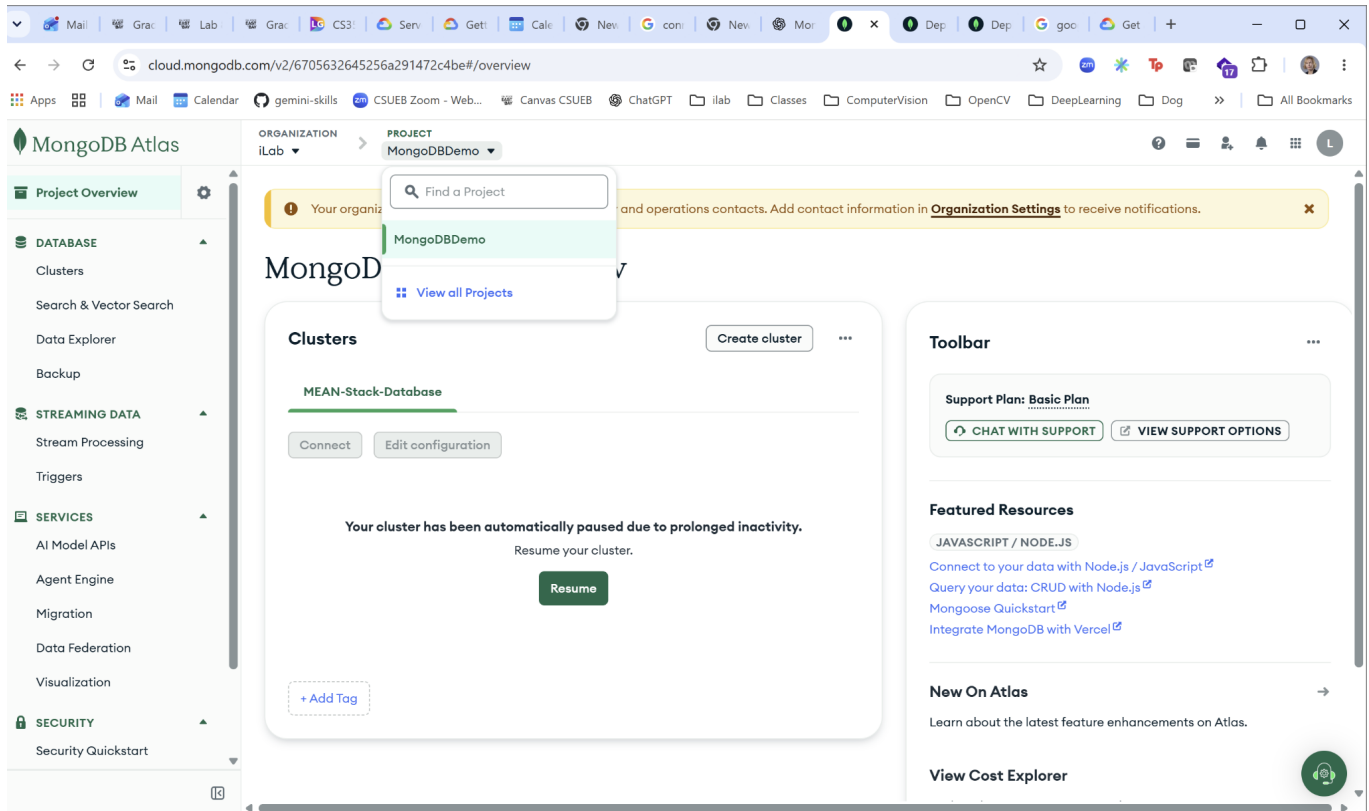
Outcome: A free Atlas cluster, a CustomerDB database, a customers collection, a manually inserted document, a secure Node.js connection URI, and temporary network access for development.

Security: Some original demonstration screenshots contained credentials. Sensitive portions are obscured in this handout. Never publish database passwords or .env files. Reset any password previously shown in a screenshot.

1 | Sign in and create a separate Atlas project

Go to [MongoDB Atlas registration](#). Sign up or sign in. From the project selector, click **View all Projects**, then create a new project for your work. The example project is **CS351_GreweDemo**.

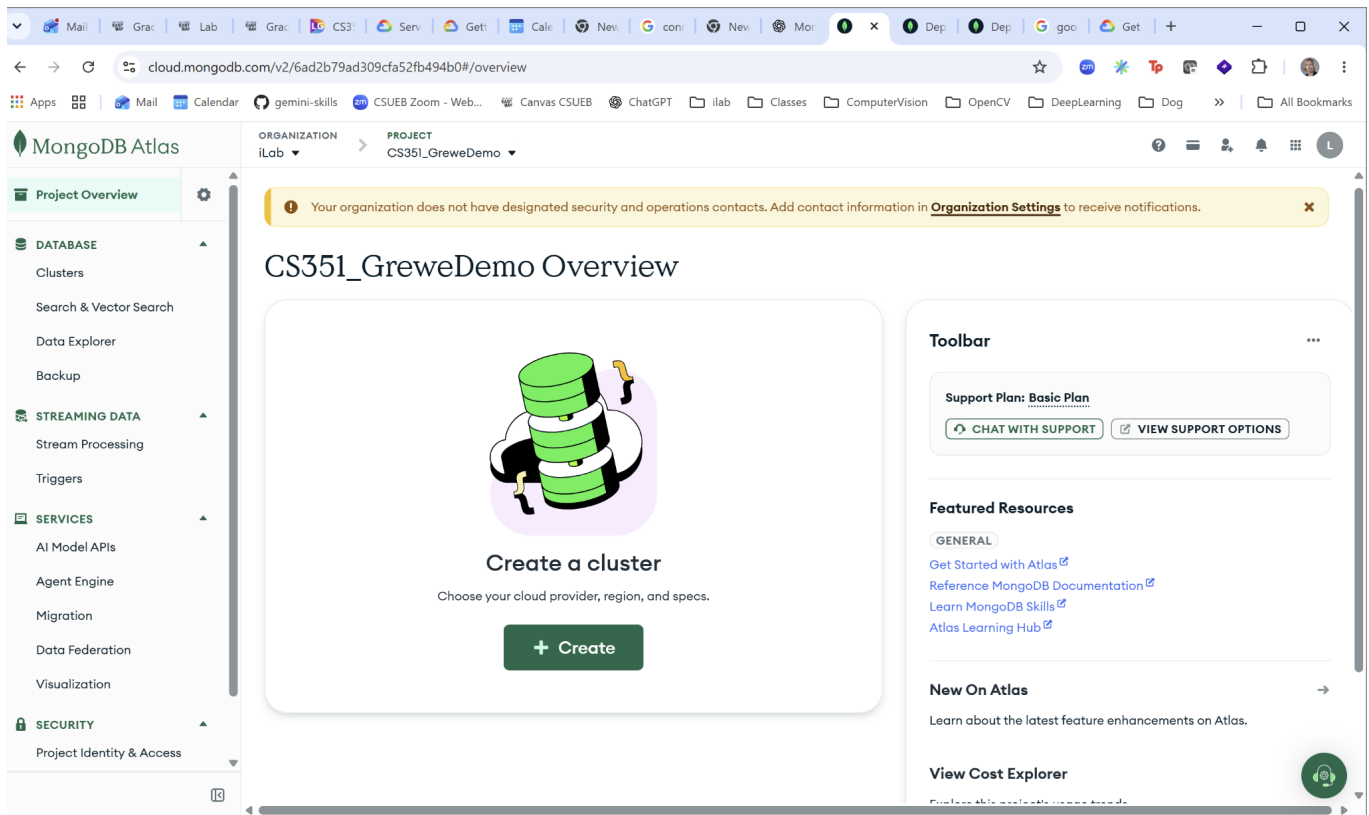
A project organizes Atlas clusters, users, and network settings. It is not the same as a Google Cloud project.



The project selector includes View all Projects.

2 | Start a new cluster

In the empty project overview, click the large green **Create** button.



An empty Atlas project prompts you to create a cluster.

3 | Choose the truly free tier

On Deploy your cluster, choose the card labeled **Free (\$0/hour)**. Do **not** select **M10** (\$0.08/hour) or **Flex** (paid). The Free option shown in this interface provides 512 MB of storage.

Keep **Automate security setup** checked. Leave **Preload sample dataset** unchecked for this exercise.

The screenshot shows the MongoDB Atlas 'Deploy your cluster' page. At the top, there's a navigation bar with 'ORGANIZATION' (ILab) and 'PROJECT' (CS351_GreweDemo). The main heading is 'Deploy your cluster'. Below it, a subtext says: 'Use a template below or set up advanced configuration options. You can also edit these configuration options once the cluster is created.'

There are three tier cards: 'M10' (\$0.08/hour), 'Flex' (From \$0.011/hour), and 'Free' (\$0/hour). The 'Free' card is selected and highlighted with a green border. Each card lists specifications: STORAGE, RAM, vCPU, and MAX CONN. Below each card is a 'Compare features' link.

Below the tier cards are three sections: 'Configurations', 'Quick setup', and 'Pricing'.
- 'Configurations' has a 'Name' field with the value 'Cluster0'.
- 'Quick setup' has two checkboxes: 'Automate security setup' (checked) and 'Preload sample dataset' (unchecked).
- 'Pricing' shows '\$0/hour' and a note: 'Free forever! Your free cluster is ideal for experimenting in a limited sandbox. You can upgrade to a production cluster anytime.'

At the bottom, there are three buttons: 'I'll do this later', 'Go to Advanced Configuration', and 'Create Deployment'.

The current Atlas interface distinguishes paid M10 and Flex from Free.

4 | Select Google Cloud and region

Give the cluster a name, for example **CS351DemoCustomers**. Select **Google Cloud** as provider and **Iowa (us-central1)** as region. Using the same region for Cloud Run later can reduce latency.

Verify the Pricing panel still reads **\$0/hour**; click **Create Deployment**.

The screenshot displays the MongoDB Atlas cluster creation wizard. At the top, three pricing tiers are shown: M10 (\$0.08/hour), Flex (From \$0.011/hour), and Free. The Free tier is selected. Below the pricing options, the 'Configurations' section includes a text input for the cluster name (CS351DemoCustomers), a dropdown for the provider (Google Cloud is selected), and a dropdown for the region (Iowa (us-central1)). The 'Quick setup' section has two checkboxes: 'Automate security setup' (checked) and 'Preload sample dataset' (unchecked). The 'Pricing' section shows '\$0/hour' for the Free tier. At the bottom, there is a 'Create Deployment' button.

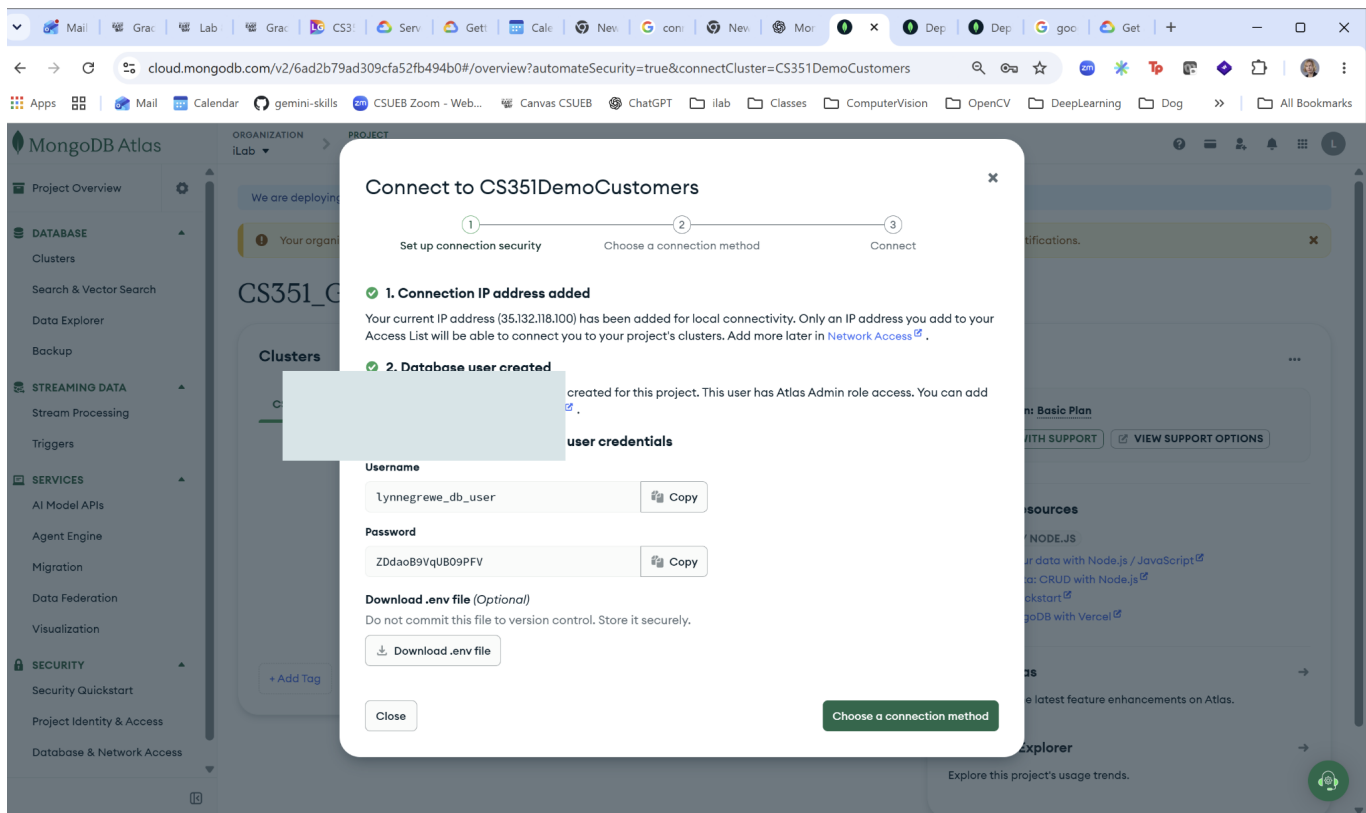
Free, Google Cloud, us-central1, and the deployment button.

5 | Save database credentials

Atlas automatic security setup adds your current public IP address and creates a database user. Save the username and password securely; optionally download the .env file. **Never commit the .env file to GitHub.**

Important: the demonstration password appeared in a screenshot. Change it under **Security > Database & Network Access > Database Users > Edit > Edit Password** before using the database. For an application, prefer a separate user with **readWrite** access to CustomerDB rather than broad atlasAdmin privileges.

Click **Choose a connection method**.

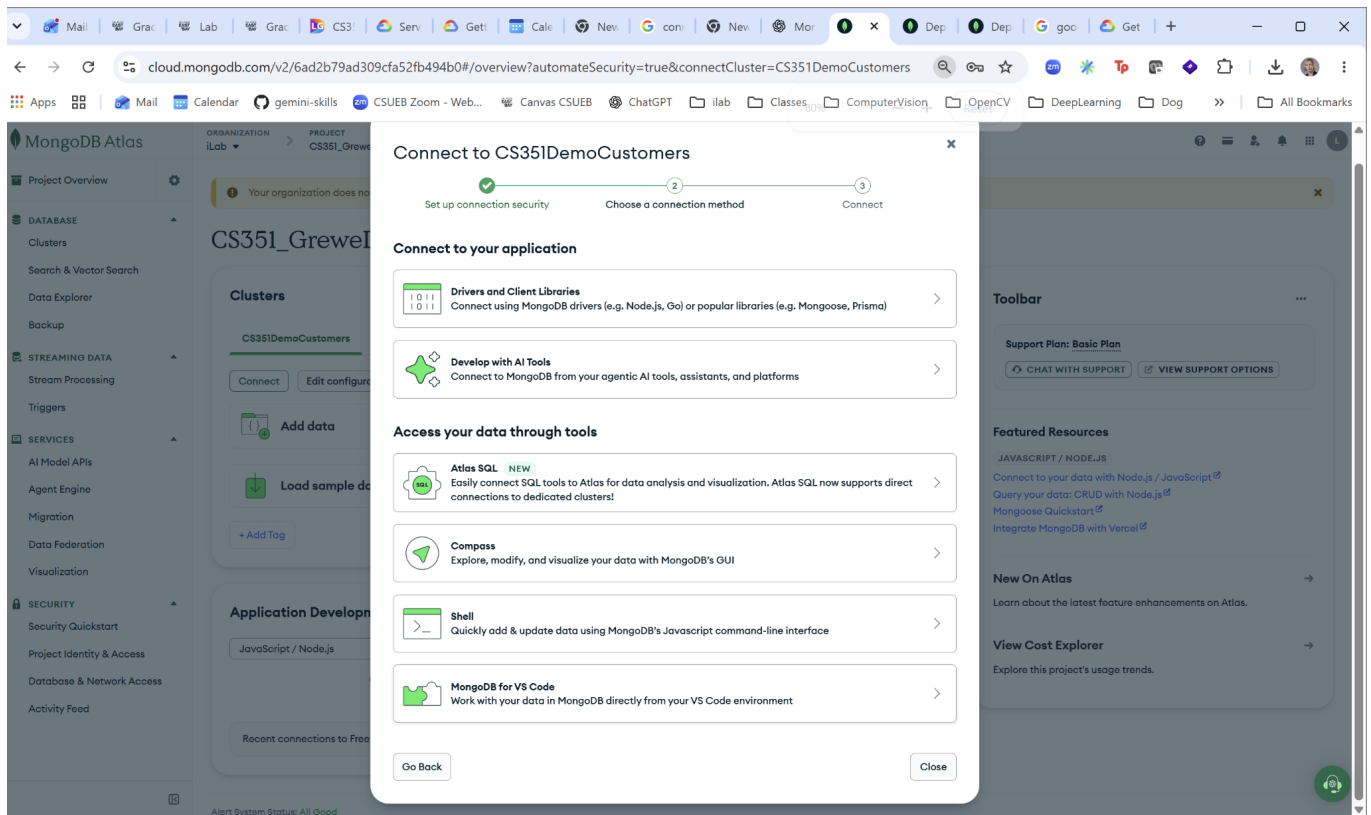


Atlas automatically creates a database user and offers an .env download. Credentials obscured.

6 | Choose the Node.js application connection

Select the first option, **Drivers and Client Libraries**. This is the only connection-method choice required for the Node.js/Express application.

Other options: Compass is an optional desktop database GUI; Shell is a command-line interface; MongoDB for VS Code is an optional editor extension; Atlas SQL and AI Tools are not needed. Manual document management is available in **Atlas Data Explorer**, outside this dialog.



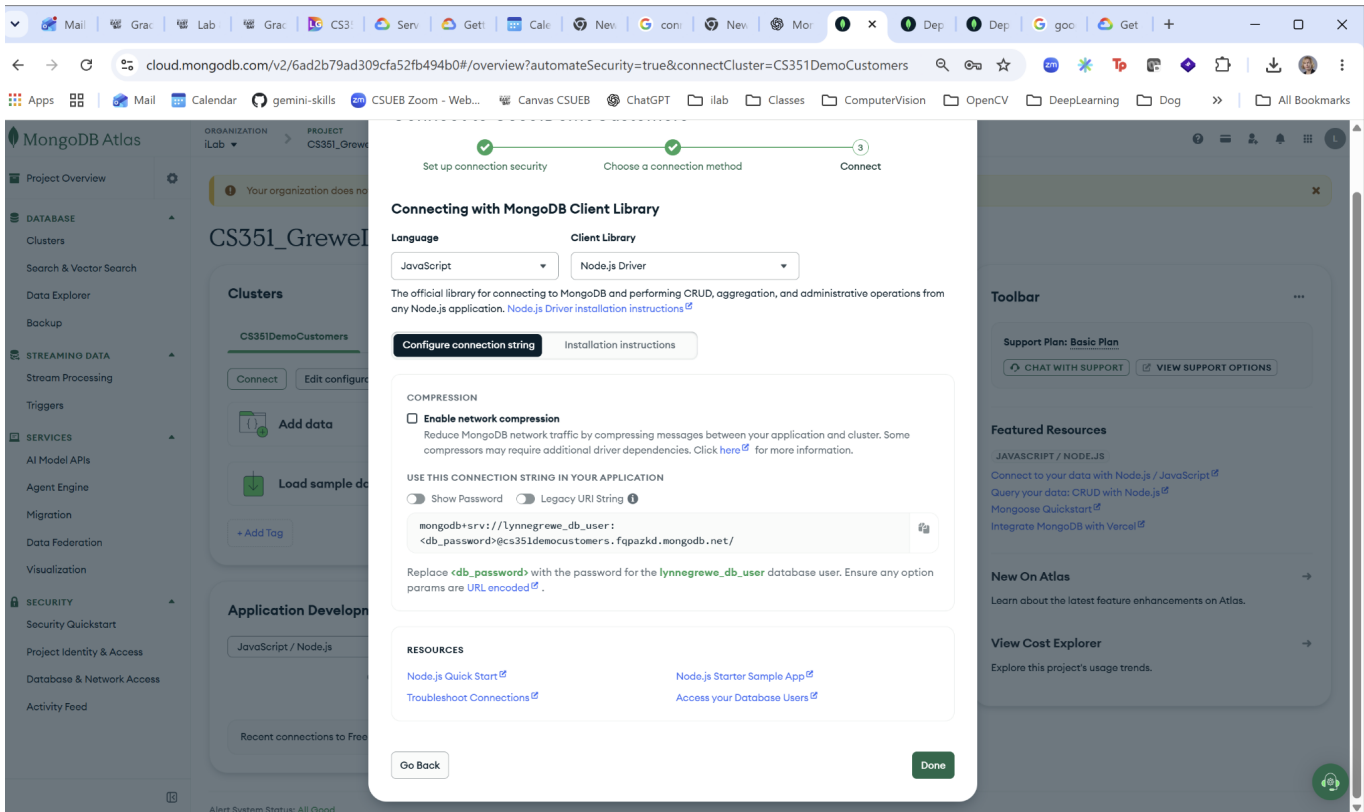
Select Drivers and Client Libraries for Express.

7 | Copy the MongoDB connection URI

Set **Language** = **JavaScript** and **Client Library** = **Node.js Driver**. Use **Configure connection string** and copy the URI. **Installation instructions** shows how to install the driver. Leave network compression and Legacy URI String disabled for this exercise.

A URI has the general form `mongodb+srv://USERNAME:<db_password>@CLUSTER-HOST/`. Replace the placeholder with your actual password in secure application configuration; URL-encode special characters when required. Do not share a completed URI containing credentials.

The official driver is installed using `npm install mongodb`. Mongoose is an optional higher-level alternative.

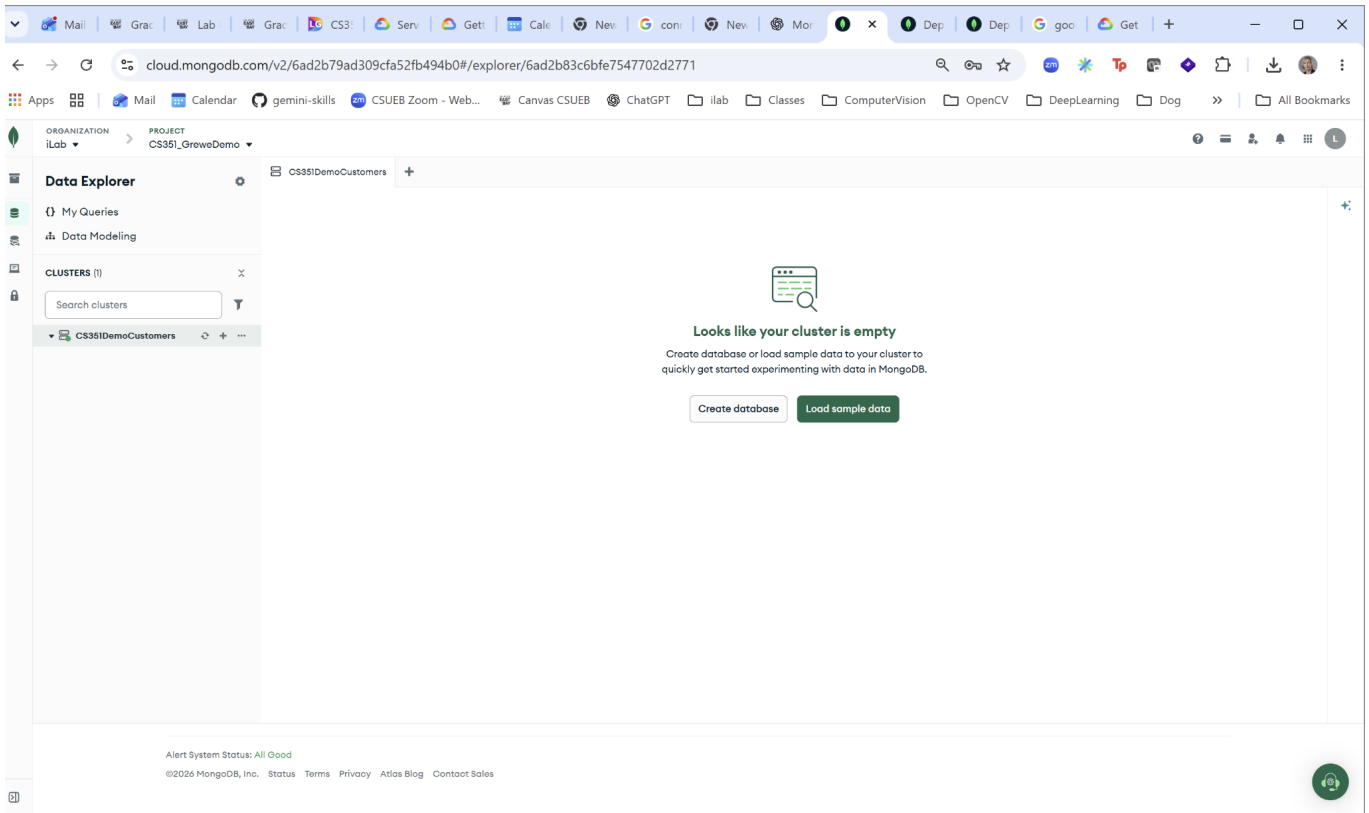


Node.js Driver connection settings and connection-string copy control.

8 | Open Data Explorer

Close the connection dialog and choose **Data Explorer** from the Atlas sidebar. This browser-based interface lets you create databases, collections, and documents and inspect changes made by Express.

An empty cluster shows a **Create database** button. Click it; do not load sample data.

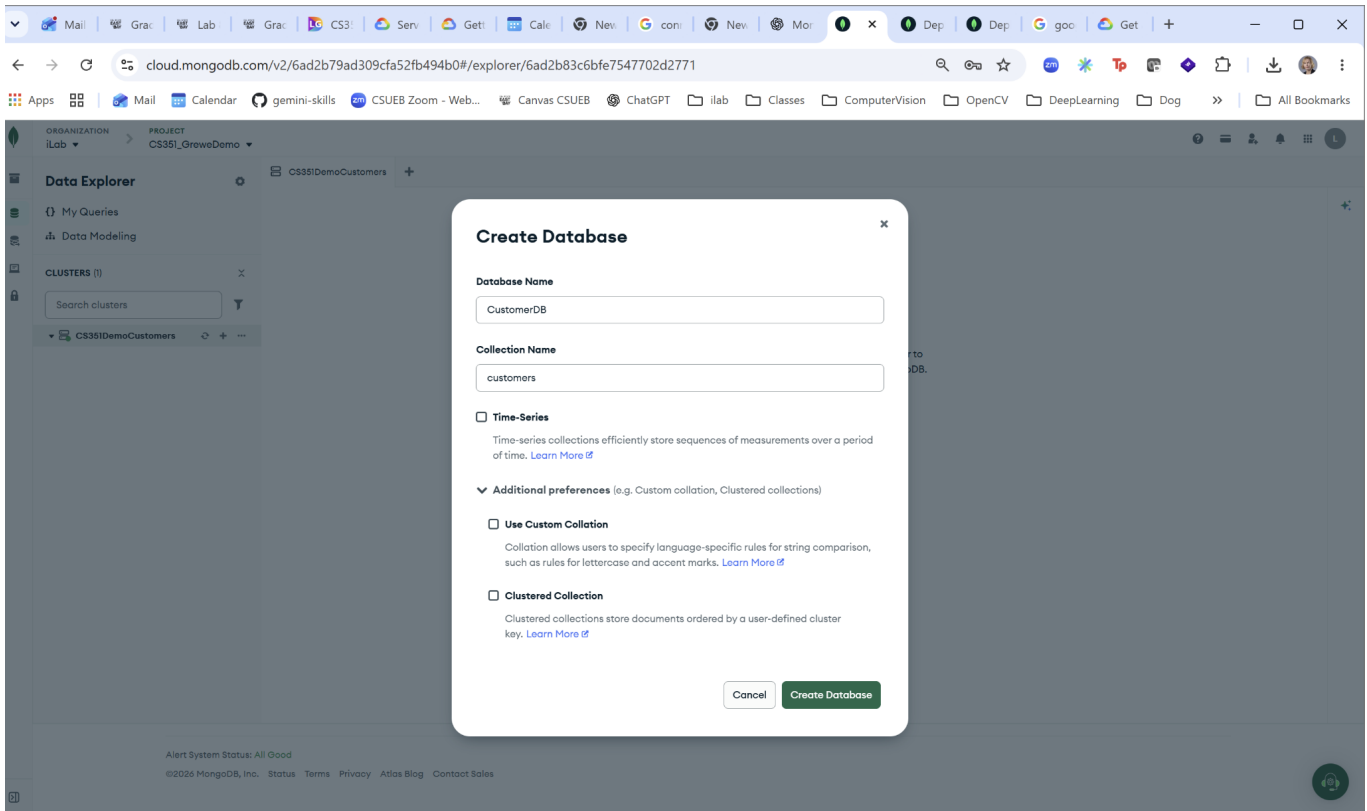


Empty Data Explorer showing Create database.

9 | Create CustomerDB and customers

Set **Database Name = CustomerDB** and **Collection Name = customers**. Leave Time-Series, Custom Collation, and Clustered Collection unchecked.

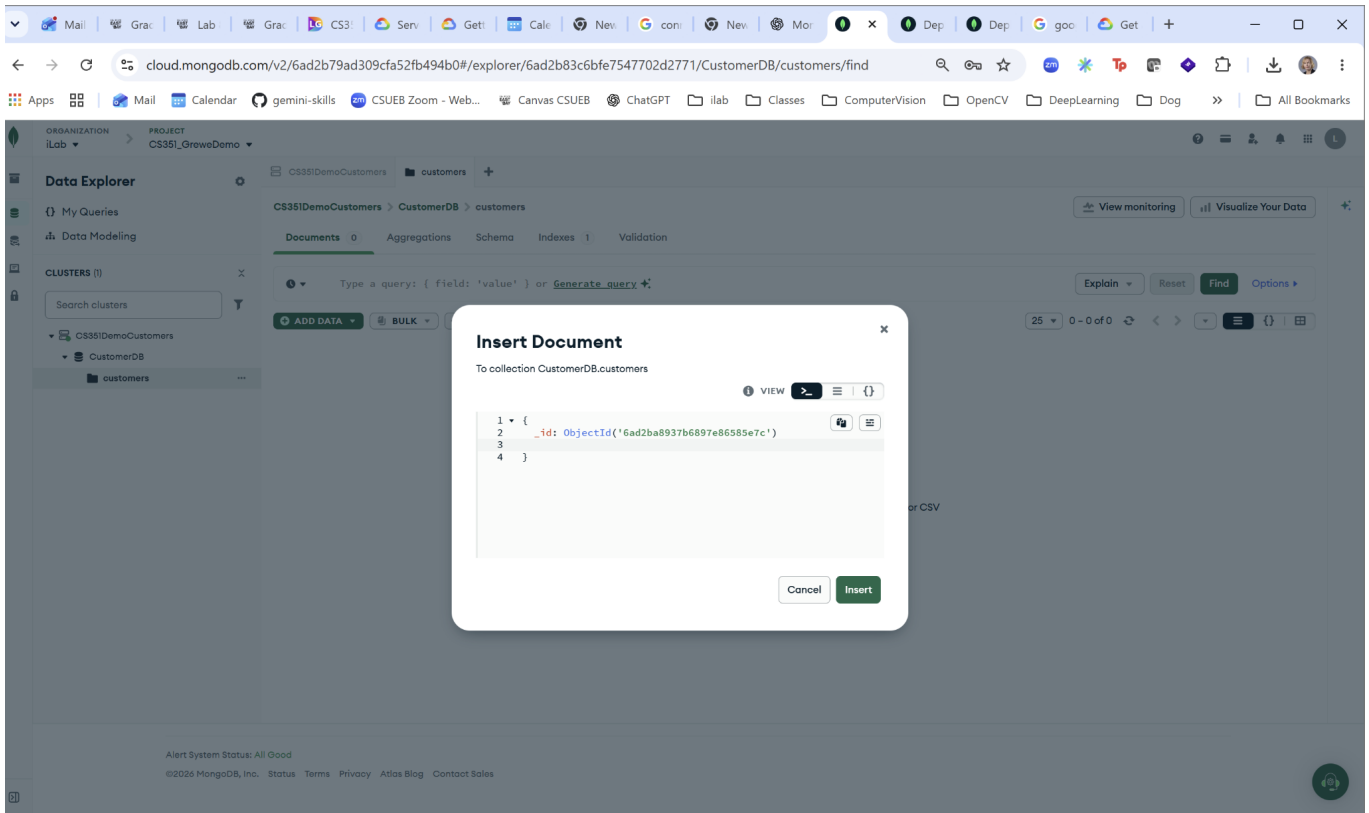
Time-Series is for timestamped measurements; Custom Collation changes string-comparison rules; Clustered Collection changes document storage organization. None is needed here. Click **Create Database**.



Create Database form with the recommended names and defaults.

10 | Insert a customer document

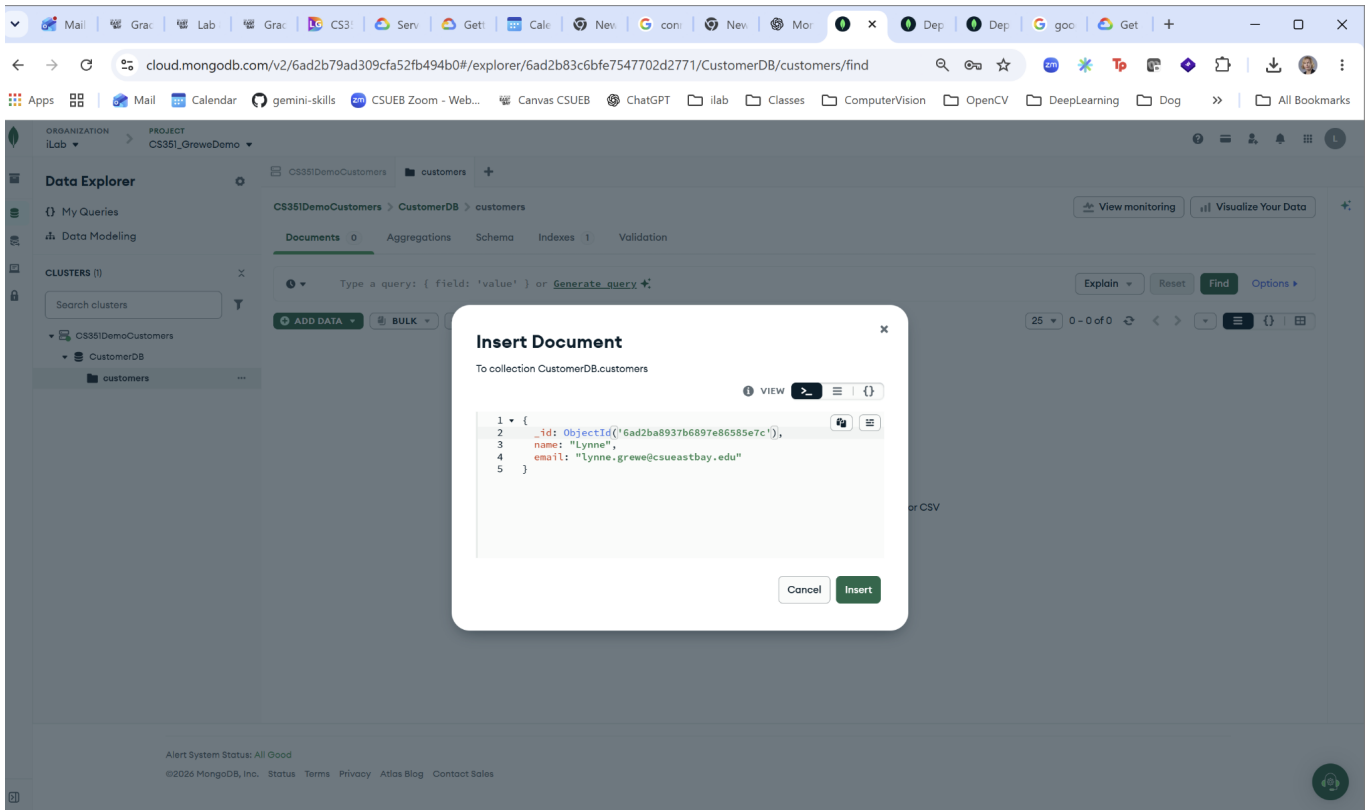
In **CustomerDB > customers**, click **+ ADD DATA > Insert Document**. The editor starts with an automatically generated `_id` of type `ObjectId`. Keep it and add fields, separating entries with commas.



The Insert Document editor starts with a generated ObjectId.

11 | Add fields and save

Enter fields such as name: "Lynne" and email: "lynne.grewe@csueastbay.edu". The editor uses a MongoDB-friendly document syntax. Leave the generated `_id` unchanged. Click **Insert**.



Example customer document ready to insert.

12 | Inspect, edit, query, and delete

The Documents tab should now show **1** document. Hover over a document to reveal editing controls. Use **Edit** to change name or add a city field; save the change. Use the document delete control to remove records when appropriate.

To filter records, type a query such as `{ name: "Lynne" }` in the query bar and click **Find**. The same collection can later be read and modified by Express.

cloud.mongodb.com/v2/6ad2b79ad309cfa52fb494b0#/explorer/6ad2b83c6bfe7547702d2771/CustomerDB/customers/find

ORGANIZATION: iLab PROJECT: CS351_GroweDemo

Data Explorer

- My Queries
- Data Modeling
- CLUSTERS (1)
 - CS351DemoCustomers
 - CustomerDB
 - customers

CS351DemoCustomers > CustomerDB > customers

Documents 1 Aggregations Schema Indexes 1 Validation

Type a query: { field: 'value' } or [Generate query](#)

[ADD DATA](#) [BULK](#) [EXPORT CODE](#)

25 1 of 1

```
{ "_id": ObjectId("6ad2ba8937b6897e86585e7c"),  
  "name": "Lynne",  
  "email": "Lynne.growe@csueastbay.edu" }
```

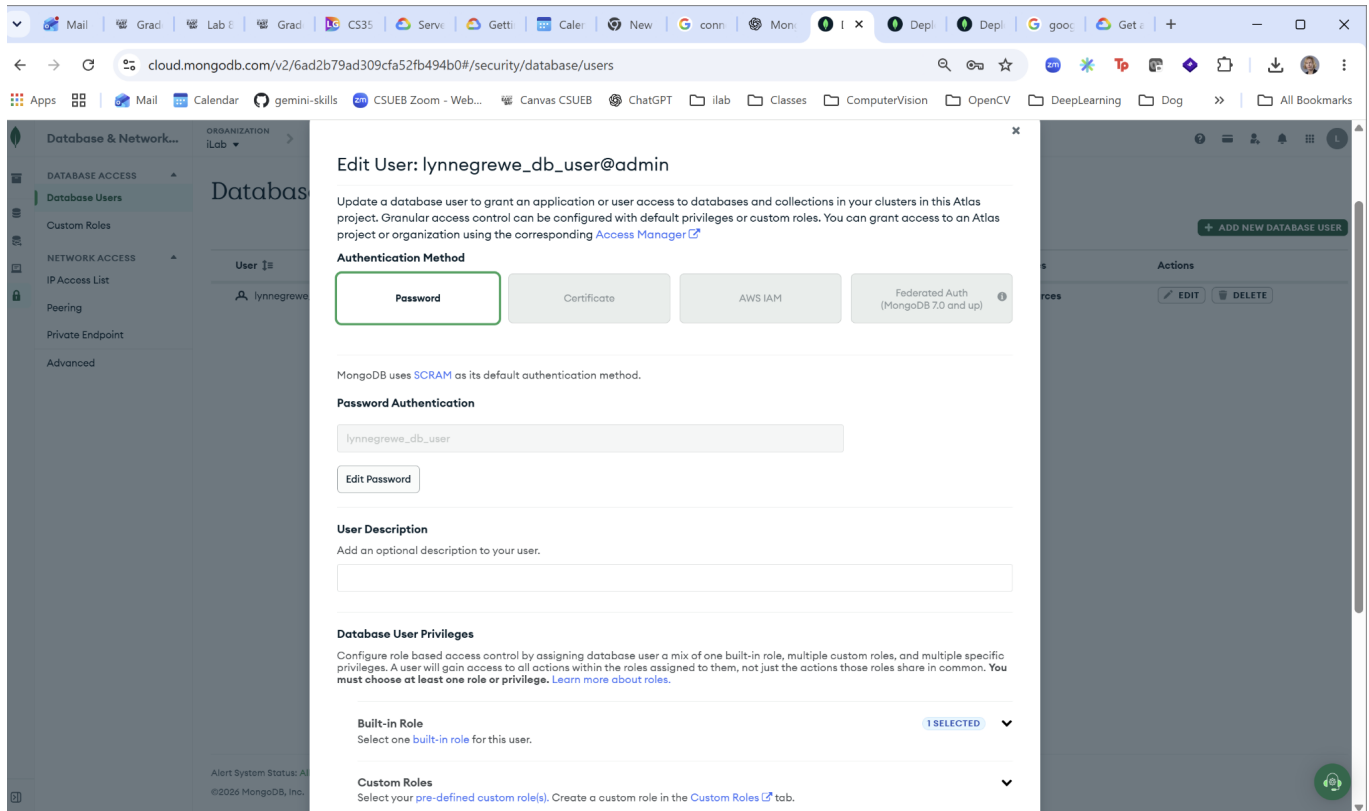
Alert System Status: All Good
©2026 MongoDB, Inc. [Status](#) [Terms](#) [Privacy](#) [Atlas Blog](#) [Contact Sales](#)

One saved customer record in Data Explorer.

13 | Locate database users and network access

Under **Security > Database & Network Access**, the **Database Users** area manages usernames, passwords, and roles. It does not control allowed source IPs.

To rotate a password, open the user, click **Edit**, and use **Edit Password**. Create an application-specific user with the least privilege necessary.



Database user editor and left-side navigation to IP Access List.

14 | Open the IP Access List

In the left sidebar, under **NETWORK ACCESS**, click **IP Access List**. Atlas initially lists the IP address added during automatic setup. Click **+ ADD IP ADDRESS** to allow another source.

Database & Network...

DATABASE ACCESS

Database Users

Custom Roles

NETWORK ACCESS

IP Access List

Peering

Private Endpoint

Advanced

ORGANIZATION
ilab

PROJECT
CS351_GreweDemo

IP Access List

You will only be able to connect to your cluster from the following list of IP Addresses:

IP Address	Comment	Status	Actions
35.192.118.100/32 (includes your current IP address)	Created as part of the Auto Setup process	Active	EDITDELETE

Alert System Status: All Good

©2025 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales

The original IP Access List contains only the setup computer address.

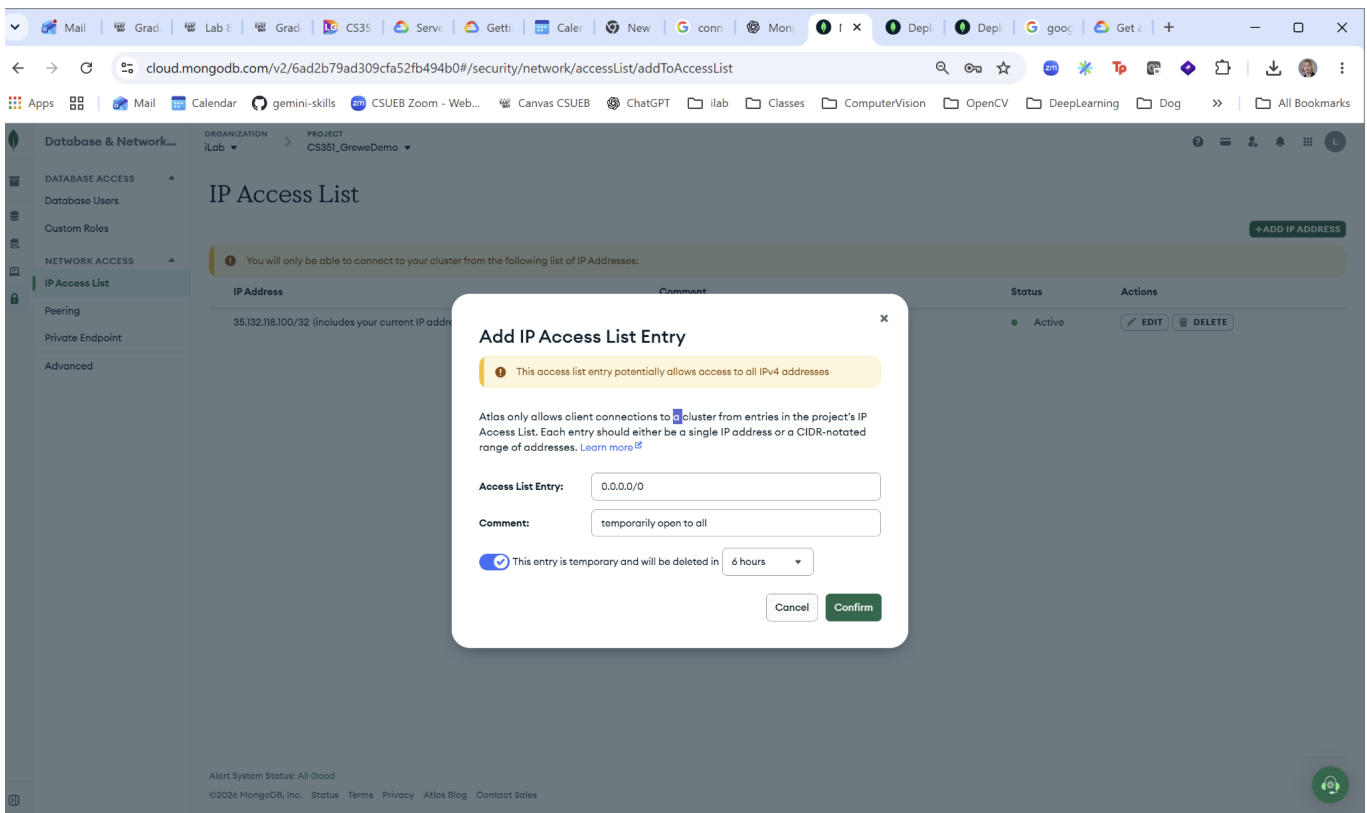
15 | Choose individual IPs or temporary anywhere access

Preferred for coworkers: Add each coworker's stable public IPv4 address with a /32 suffix. For example, 203.0.113.25/32. If their public IP changes, update the entry. Everyone still needs valid MongoDB database credentials and privileges.

Temporary Cloud Run testing: Enter 0.0.0.0/0 to permit connection attempts from any IPv4 address. Add a comment such as "temporarily open to all". **Enable the temporary entry checkbox.** Use 1 hour for a quick test or **6 hours** for a lab. Click **Confirm**.

Security warning: 0.0.0.0/0 exposes the endpoint to internet-wide connection attempts. Authentication remains required, but stolen credentials or weak passwords could permit unauthorized access. Never leave this rule indefinitely enabled.

Expiration: Atlas automatically deletes the temporary rule when time runs out. Your data remains, but later Cloud Run connections may fail until a new permitted entry is added. For ongoing deployments, configure Cloud Run static outbound IPs and allowlist those addresses.



Temporary 0.0.0.0/0 rule with six-hour expiration.

16 | Connect the Node.js/Express application

Install the driver in your Node.js project: `npm install mongodb`. Store the full connection URI in the `MONGODB_URI` environment variable, not in source code. In Cloud Run, use Secret Manager to supply the URI securely.

Minimal server-side example:

```
const { MongoClient } = require("mongodb");
const client = new MongoClient(process.env.MONGODB_URI);

async function getCustomers() {
  await client.connect();
  return client.db("CustomerDB")
    .collection("customers").find({}).toArray();
}
```

Deploy the Express application separately to **Google Cloud Run** in your Google Cloud project. Atlas remains managed in your MongoDB account. A successful test should retrieve the customer document inserted manually in Data Explorer.

Final student checklist

- Free cluster on Google Cloud (Atlas-managed)
- CustomerDB database and customers collection created
- At least one document inserted and visible
- Node.js URI copied and stored securely
- Previously exposed password rotated; app user has limited privileges
- Temporary IP access confirmed and its expiration understood
- Express connection tested; Cloud Run deployment is a separate step

Official documentation: [MongoDB Atlas Free cluster](#) | [MongoDB Node.js driver](#) | [Google Cloud Run Node.js](#)