Description of Application Architecture

My Crypto Price Searcher application takes in the user's crypto search string and return the current crypto price. Here is how my meet the task requirement:

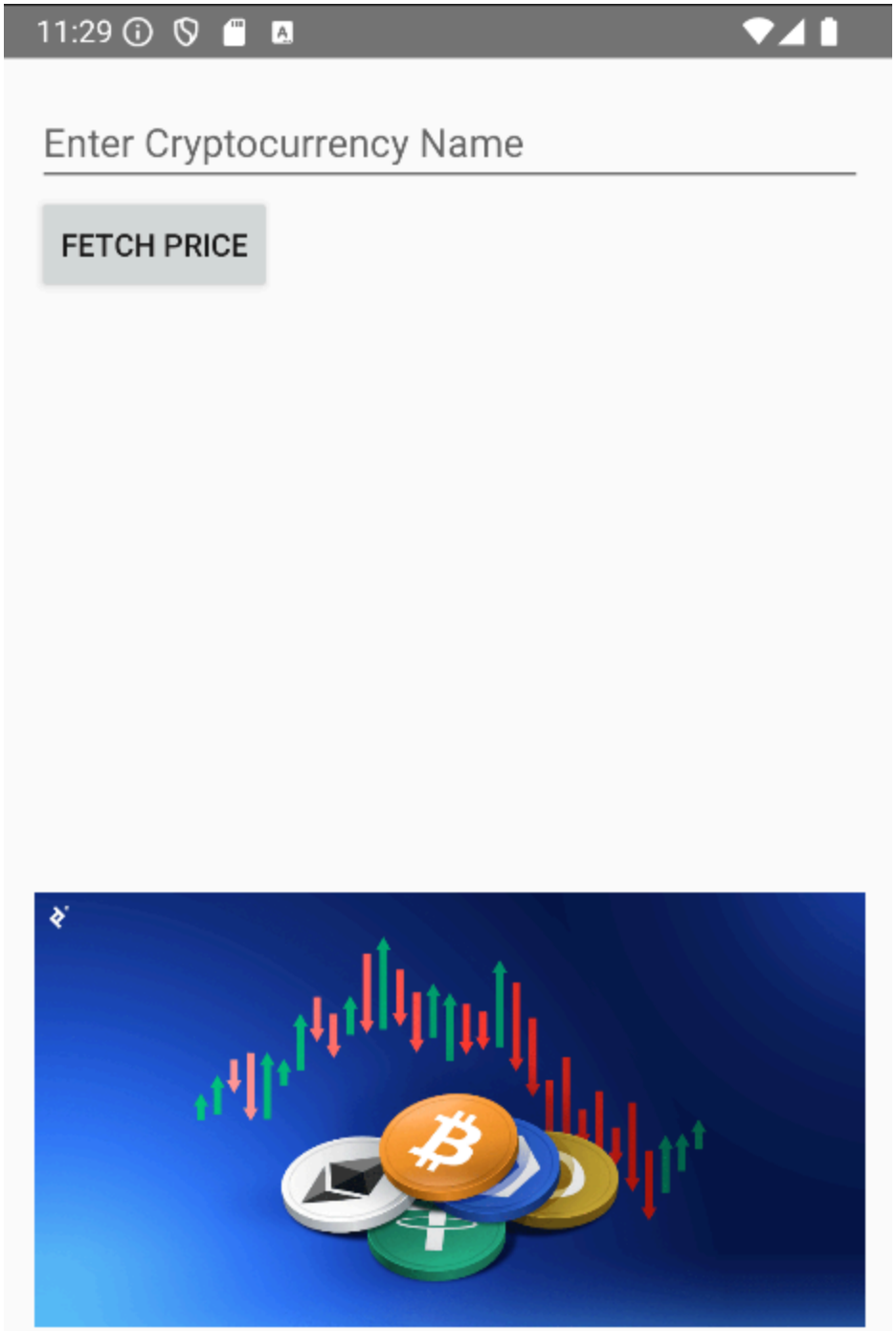
1. Implement a native Android Application:

The name of my application is CryptoApplication.

a. Has at least three different kinds of Views in your Layout (TextView, EditText, ImageView, or anything that extends android.view.View).

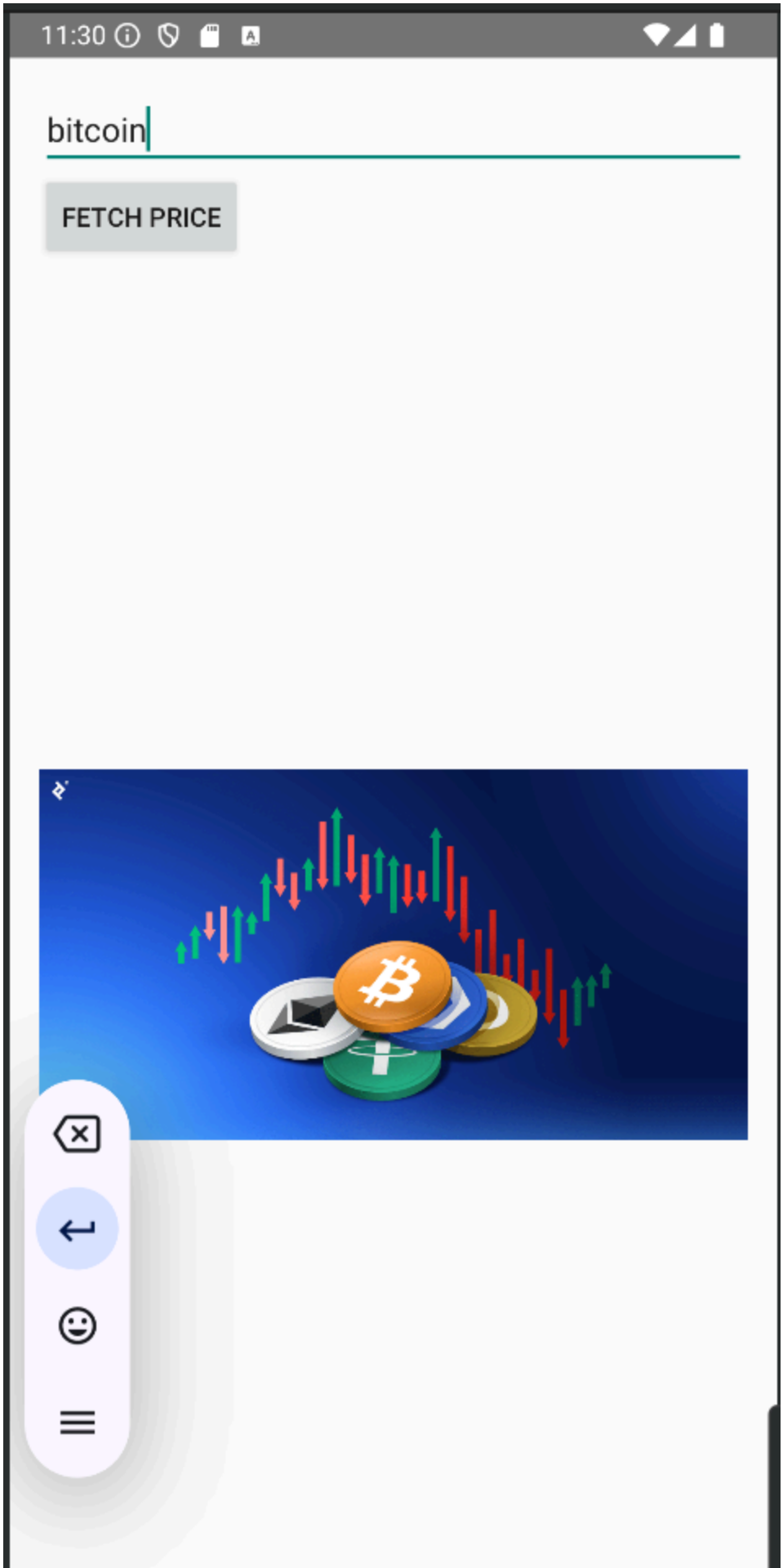
My application uses TextView, EditText, Button, and ImageView. See activity_main.xml and activity_crypto_price.xml for details of how they are incorporated into the LinearLayout.

Here is a screenshot of the layout when just launch the application:

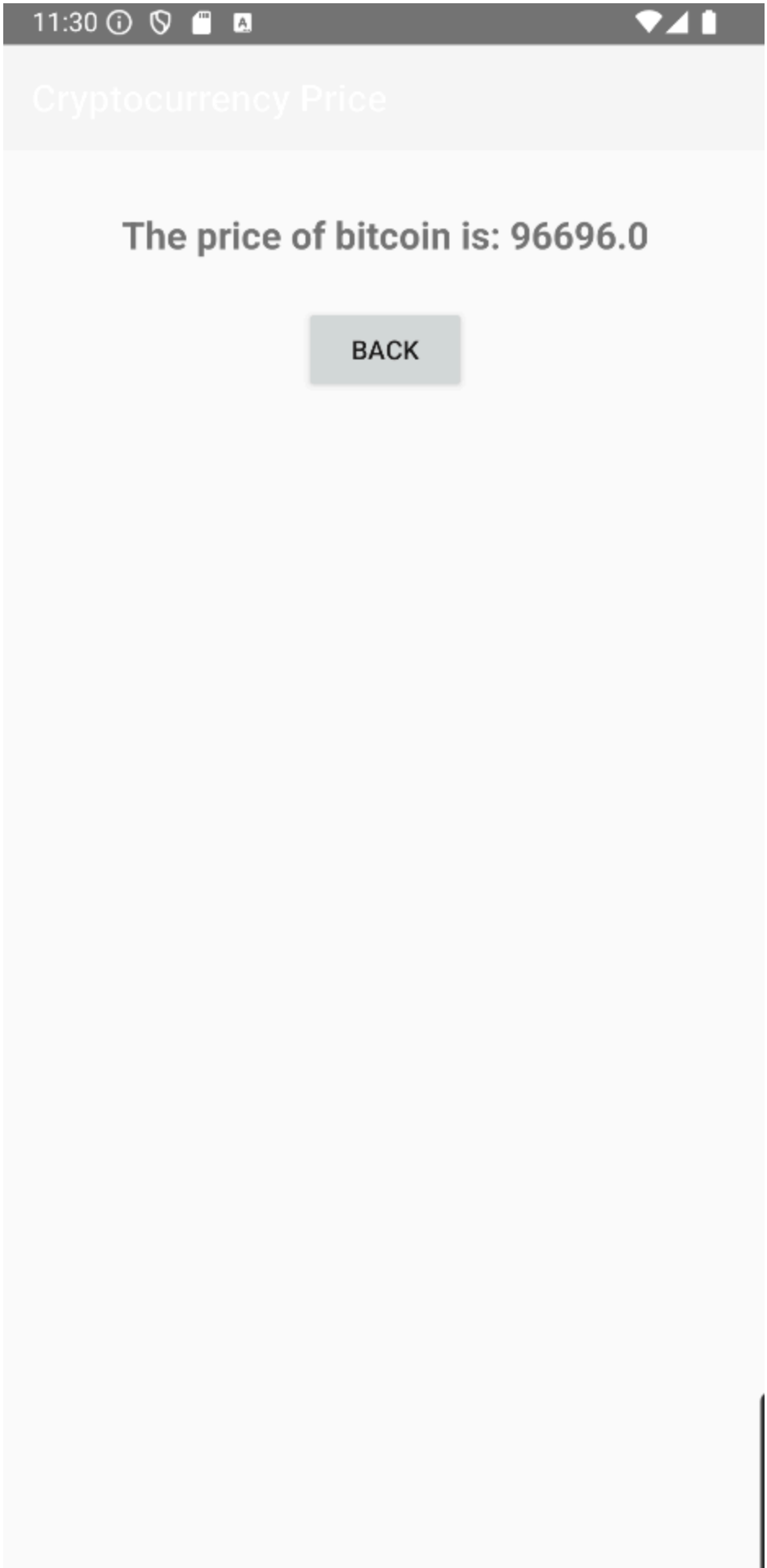

11:29
Enter Cryptocurrency Name
FETCH PRICE

b. Requires Input from user:

Here is a screenshot of a sample user input:


11:30
bitcoin
FETCH PRICE

c. Makes an HTTP request (using an appropriate HTTP method) to your web service
The CryptoServlet makes a HTTP GET request to reach a CoinGecko API endpoint to fetch current price information. The request URL has the pattern like "https://api.coingecko.com/api/v3/simple/price?ids=" + cryptoName + "&vs_currencies=usd", where cryptoName is provided by the user. The request reaches your web application, which processes the request and returns JSON data containing relevant information.
d. Receives and parses an XML or JSON formatted reply from your web service
The application receives a JSON response from the CoinGecko API. The response contains details like the name and current price. Then, the app parses this JSON response in MongoDB.java, which then extracts and stores the details in the database. An example of the received JSON format might look like:
{ "crypto": "bitcoin", "price": 34000,"timestamp": "2024-11-21T12:00:00Z"}
e. Display new information to user:
A sample screenshot is:


11:30
Cryptocurrency Price
The price of bitcoin is: 96696.0
BACK

f. Is repeatable (I.e. the user can repeatedly reuse the application without restarting it.)

The user can hit the back button and go back to the search page and start a new price search.

2. Implement a Web Service

a. Implement a Web Service

The URL of my web service deployed to Codespace is

http://supreme-zebra-9v64997vjqwfvw5-8080.app.github.dev/api/crypto

b. Receives an HTTP request from the native Android application

The application sends an HTTP GET request to your web service via a specific endpoint, /cryptoName=bitcoin.

c. Executes business logic and processes XML or JSON from a third-party API

The service processes incoming requests by making a GET call to the cryptocurrency price API, parses the JSON response, and stores information in MongoDB.

d. Replies with an XML or JSON formatted response

Once the data is processed, the service sends back a JSON-formatted response to the Android app containing details like the crypto name, price, and timestamp.

The distributed aspect is handled using the MongoDB Atlas cloud service. MongoDB stores cryptocurrency request logs, user IPs, and analytics, which are queried by the servlet to update the dashboard. GitHub Codespaces was used as the development environment.

4. Logging Information

In the application, I log the following pieces of information for each request and reply:

1. Timestamp: The time when the request was made, ensuring we can track user activity.
2. Cryptocurrency Name: The name of the cryptocurrency requested, to understand trends in user preferences.
3. Price: The current price of the requested cryptocurrency, useful for auditing accuracy.
4. User Agent: The client's browser or application version, aiding in understanding platform usage.
5. Client IP: The IP address from which the request was made, which helps in user geolocation analysis.
6. Response Time: The time taken for each request, used to ensure the application performs efficiently.

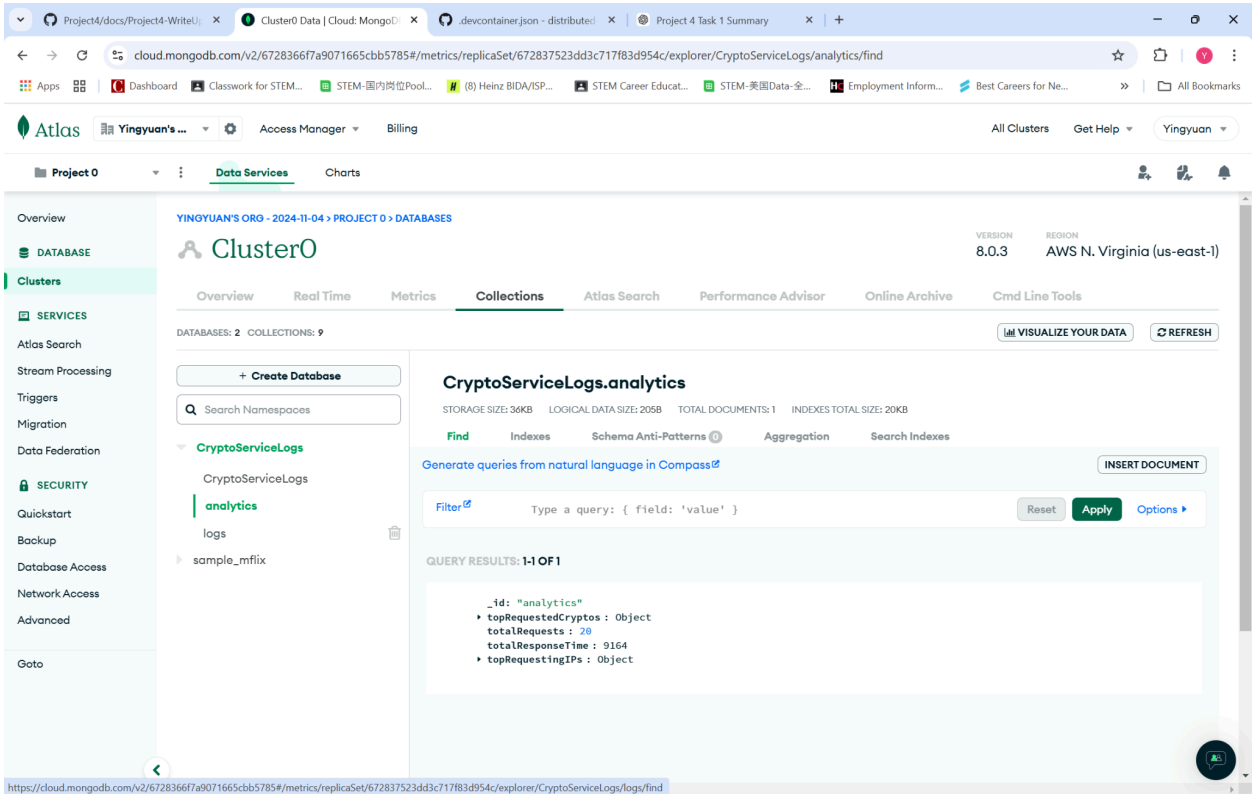
5. Atlas Database Logging

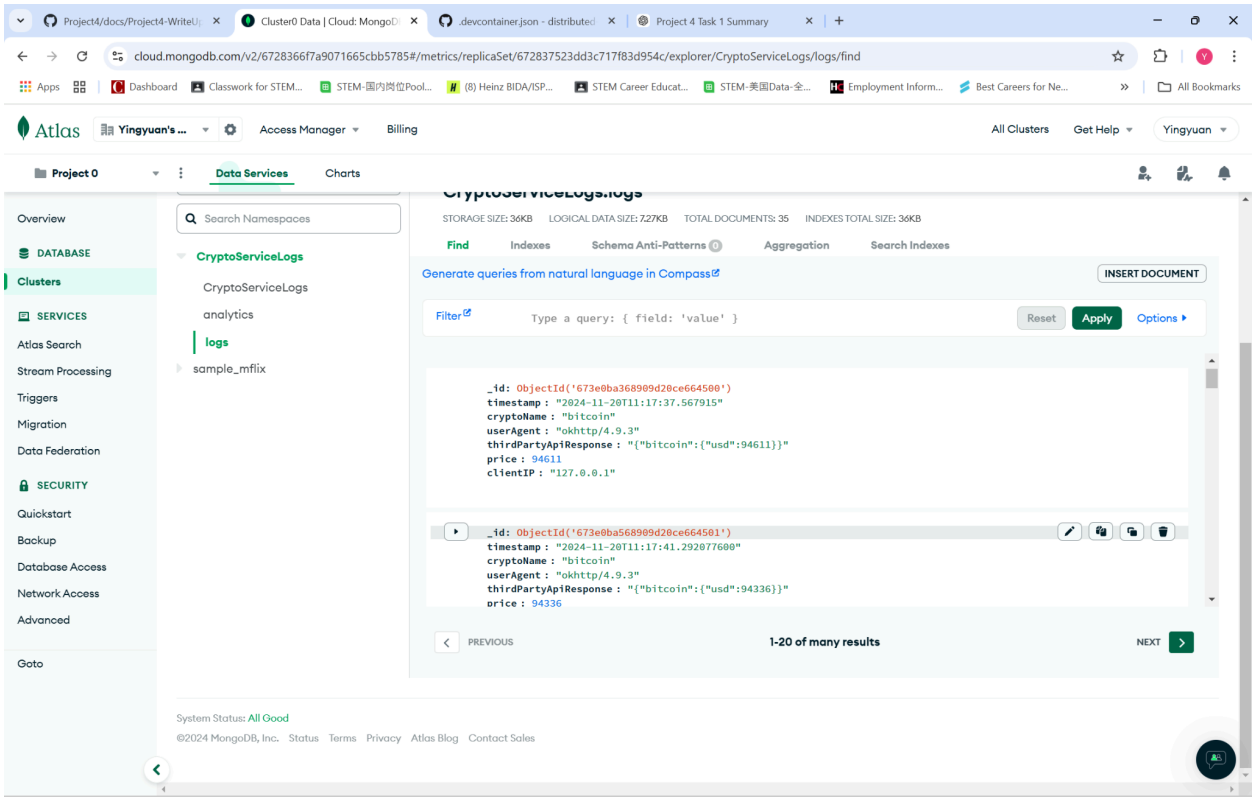
The MongoDB Atlas connection string used is:

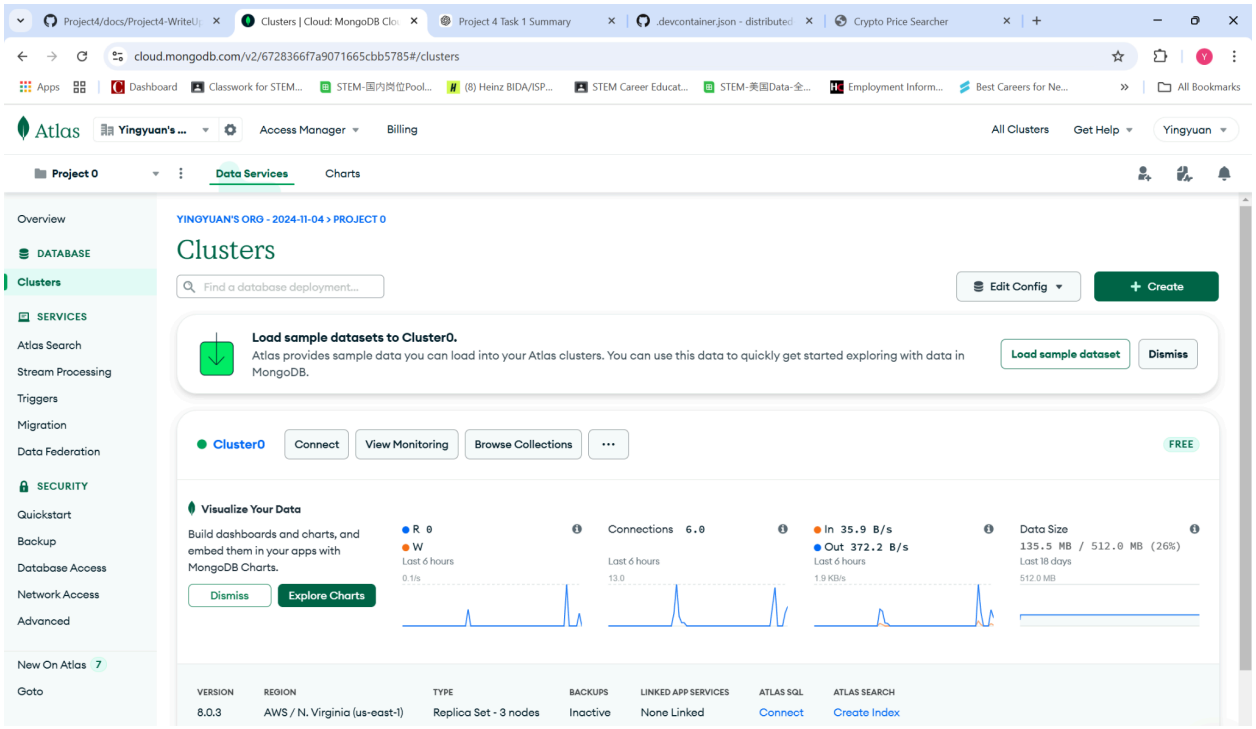
mongodb+srv://yingyual:1652043681aA@cluster0.jk5a5.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0

My MongoDB Atlas cluster uses a Replica Set configuration with three shards, ensuring scalability and data distribution.

Here is screenshot of my MongoDB Structure:


cloud.mongodb.com/v2/6728366f7a9071665cbb5785#/metrics/replicaSet/672837523dd3c717f83d954c/explorer/CryptoServiceLogs/analytics/find
Atlas
Yingyuan's ...
Access Manager
Billing
All Clusters
Get Help
Yingyuan
Project 0
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
Goto
YINGYUAN'S ORG - 2024-11-04 > PROJECT 0 > DATABASES
Cluster0
VERSION
8.0.3
REGION
AWS N. Virginia (us-east-1)
Overview
Real Time
Metrics
Collections
Atlas Search
Performance Advisor
Online Archive
Cmd Line Tools
DATABASES: 2 COLLECTIONS: 9
VISUALIZE YOUR DATA
REFRESH
+ Create Database
Search Namespaces
CryptoServiceLogs
CryptoServiceLogs
analytics
logs
sample_mflix
CryptoServiceLogs.analytics
STORAGE SIZE: 36KB
LOGICAL DATA SIZE: 205B
TOTAL DOCUMENTS: 1
INDEXES TOTAL SIZE: 20KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
QUERY RESULTS: 1-1 OF 1
_id: "analytics"
topRequestedCryptos : Object
totalRequests : 20
totalResponseTime : 9164
topRequestingIPs : Object
https://cloud.mongodb.com/v2/6728366f7a9071665cbb5785#/metrics/replicaSet/672837523dd3c717f83d954c/explorer/CryptoServiceLogs/logs/find


cloud.mongodb.com/v2/6728366f7a9071665cbb5785#/metrics/replicaSet/672837523dd3c717f83d954c/explorer/CryptoServiceLogs/logs/find
Atlas
Yingyuan's ...
Access Manager
Billing
All Clusters
Get Help
Yingyuan
Project 0
Data Services
Charts
Overview
DATABASE
Clusters
SERVICES
Atlas Search
Stream Processing
Triggers
Migration
Data Federation
SECURITY
Quickstart
Backup
Database Access
Network Access
Advanced
Goto
Search Namespaces
CryptoServiceLogs
CryptoServiceLogs
analytics
logs
sample_mflix
STORAGE SIZE: 36KB
LOGICAL DATA SIZE: 7.27KB
TOTAL DOCUMENTS: 35
INDEXES TOTAL SIZE: 36KB
Find
Indexes
Schema Anti-Patterns
Aggregation
Search Indexes
Generate queries from natural language in Compass
INSERT DOCUMENT
Filter
Type a query: { field: 'value' }
Reset
Apply
Options
_id: ObjectId('673e0ba568909d20ce664500')
timestamp : "2024-11-20T11:17:37.567915"
cryptoName : "bitcoin"
userAgent : "okhttp/4.9.3"
thirdPartyApiResponse : "{"bitcoin":{"usd":94611}}"
price : 94611
clientIP : "127.0.0.1"
_id: ObjectId('673e0ba568909d20ce664501')
timestamp : "2024-11-20T11:17:41.292077600"
cryptoName : "bitcoin"
userAgent : "okhttp/4.9.3"
thirdPartyApiResponse : "{"bitcoin":{"usd":94336}}"
price : 94336
PREVIOUS
1-20 of many results
NEXT
System Status: All Good
©2024 MongoDB, Inc. Status Terms Privacy Atlas Blog Contact Sales



This connects to my three-node MongoDB replica set where all logging data is stored.

6. Web Dashboard for Analytics and Logs

Here is a screenshot of my dashboard: