**Problem Statement**: Our organization relies on multiple cloud service providers to support various aspects of our operations, including AWS, Microsoft Azure, Google Cloud Platform (GCP), and potentially others. Each cloud provider has its billing system and dashboard, making it challenging to track and manage our overall cloud expenditure efficiently.

**Challenges:**

1. Not knowing how much we're spending on cloud services because we have different billing systems, which can lead to overspending.
2. Spending a lot of time manually collecting and double-checking data from these different billing systems, which is slow and can result in mistakes.
3. Having a hard time making smart decisions about where to spend money on cloud services because we don't have all the spending data in one place.
4. Struggling to follow rules and stick to budgets when using different cloud platforms without a single billing dashboard.
5. Wasting time and money because our current setup doesn't work well for dividing costs, predicting expenses, and managing overall cloud spending.

**Solution Requirement:** To address this issue, we need to develop or implement a single, integrated billing dashboard that consolidates billing data from all cloud providers. This dashboard should provide real-time insights, cost analysis, and reporting capabilities to enable better financial management of our multi-cloud environment.

**Proposed Architecture:**

**A diagram of a data processing process

Description automatically generated**

**Benefits:**

* Improved cost transparency and control.
* Streamlined cloud expense management.
* Enhanced decision-making and resource allocation.

**Project Scope:** This project aims to design, develop, or implement a unified multi-cloud billing dashboard. It should integrate with various cloud provider APIs, securely fetch billing data, and present it in an easy-to-understand format. The dashboard should also offer features such as cost forecasting, trend analysis, and cost allocation capabilities.

**Project Timeline and Resources:** The project timeline, required resources, and budget will be determined as part of the project planning phase.

**Stakeholders:**

* Cloud Administrators
* Finance Team
* IT Management

**Implementation:**

Install MongoDB Community Edition:

Follow these steps to install MongoDB Community Edition using the apt package manager.

**Import the public key used by the package management system**

From a terminal, install gnupg and curl if they are not already available:

|  |
| --- |
| **$ sudo apt-get install gnupg curl** |

To import the MongoDB public GPG key from <https://pgp.mongodb.com/server-7.0.asc>, run the following command:

|  |
| --- |
| **$ curl -fsSL https://pgp.mongodb.com/server-7.0.asc | \** |
| **sudo gpg -o /usr/share/keyrings/mongodb-server-7.0.gpg \** |
| **--dearmor** |

**Create a list file for MongoDB**

Create the /etc/apt/sources.list.d/mongodb-org-7.0.list file for Ubuntu 20.04 (Focal):

|  |
| --- |
| **$ echo "deb [ arch=amd64,arm64 signed-by=/usr/share/keyrings/mongodb-server-7.0.gpg ] https://repo.mongodb.org/apt/ubuntu focal/mongodb-org/7.0 multiverse" | sudo tee /etc/apt/sources.list.d/mongodb-org-7.0.list** |

**Reload local package database**

Issue the following command to reload the local package database:

|  |
| --- |
| **$ sudo apt-get update** |

**Install the MongoDB packages**

You can install either the latest stable version of MongoDB or a specific version of MongoDB. To install the latest stable version, issue the following:

|  |
| --- |
| **$ sudo apt-get install -y mongodb-org** |

When you first install MongoDB, you typically won't have any authentication enabled, which means you can access the MongoDB server without a username and password. This is why there isn't a "first-time login" in the traditional sense. You can connect to MongoDB without specifying authentication credentials.

However, for a production environment or any situation where security is a concern, you should enable authentication and create user accounts with appropriate roles. Here's how you can enable authentication and create your first user:

**Start MongoDB with Authentication**:

To enable authentication, you need to modify your MongoDB server configuration file, typically located at **/etc/mongod.conf**. Edit the configuration file using a text editor, and add or modify the **security** section as follows:

**security:**

**authorization: enabled**

Save the configuration file and restart the MongoDB service:

**$ sudo systemctl restart mongodb**

Enabling authentication will require all clients to provide valid credentials when connecting to the MongoDB server.

**Create an Administrative User**:

To create your first user, you need to connect to the MongoDB server without credentials. You can do this because you have just enabled

**$ mongo**

Once you're in the MongoDB shell, switch to the **admin** database:

**> use admin**

Now you can create an administrative user with roles like **userAdminAnyDatabase** and **dbAdminAnyDatabase**. Replace **admin\_username** and **admin\_password** with your chosen username and password:

**> $db.createUser({ user: "admin\_username", pwd: "admin\_password", roles: ["userAdminAnyDatabase", "dbAdminAnyDatabase"] })**

This user can be used to create other users and manage the MongoDB instance.

**Exit the MongoDB Shell**:

You can exit the MongoDB shell by typing **exit** or pressing Ctrl+C.

After following these steps, your MongoDB instance will be secure, and you will need to provide authentication credentials to connect. Use the **admin\_username** and **admin\_password** you just created to log in, and from there, you can create additional users for your databases with specific roles and permissions.

To create a collection in MongoDB, you can use the **createCollection** method in the MongoDB shell. Here's how you can create a collection:

**Open the MongoDB Shell**: Start by opening the MongoDB shell. You can do this by running the **mongo** command in your terminal or command prompt:

**>mongo**

**Switch to the Database**: If you're not already in the desired database, switch to the database where you want to create the collection using the **use** command:

**>use your\_database\_name**

Replace **your\_database\_name** with the name of the database you want to work with.

**Create the Collection**: To create a collection, use the **createCollection** method. Specify the collection name and any optional options.

**>****db.createCollection("your\_collection\_name", { capped: false, // Whether the collection is capped (limited size) size: 1024, // Maximum size for capped collections (in bytes) max: 1000, // Maximum number of documents for capped collections validator: {}, // Validation rules for the collection validationLevel: "off", // Validation level validationAction: "warn" // Validation action })**

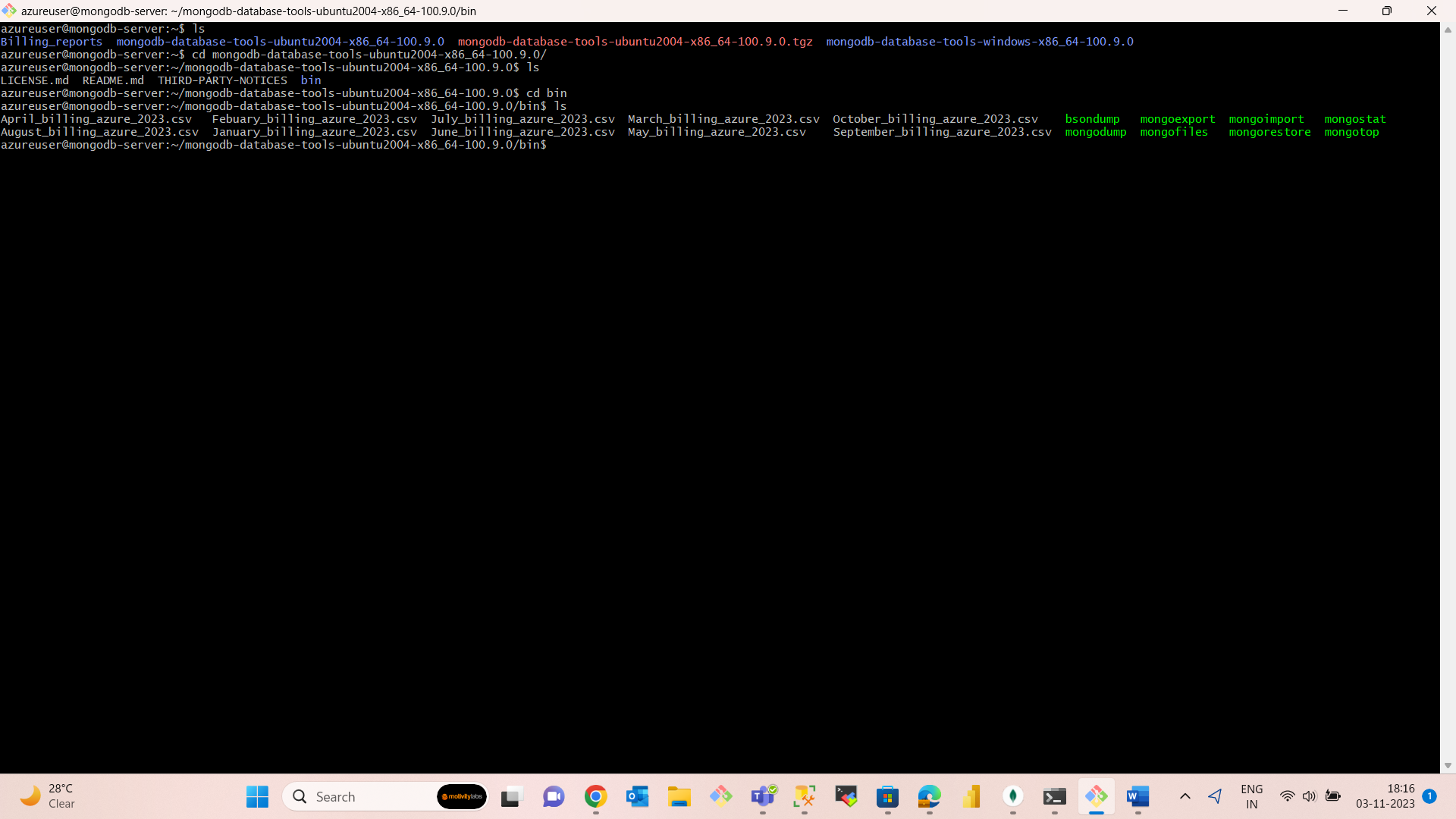
**MongoDB Command Line Database Tools Download:**

Use the following commands to download and extract the MongoDB Command Line Database Tools.

**$ wget https://fastdl.mongodb.org/tools/db/mongodb-database-tools-ubuntu2004-x86\_64-100.9.0.tgz**

**$ tar xvzf** **mongodb-database-tools-ubuntu2004-x86\_64-100.9.0.tgz**

Once the mongodb database tools are unarchived, change the directory to **mongodb-database-tools-ubuntu2004-x86\_64-100.9.0, y**ou will find bin folder inside it. Copy all the billing\_reports into **bin folder** and run the following command **$ ./mongoimport --db database\_name --collection collection\_name --type csv --headerline --file file\_name.csv.** This command will dump the data into mongodb database.



This mongodb command will “**db.your\_collection\_name.find().limit(800)”** will publish all the data that has been dumped into mongodb server.

