Google Cloud MSSQL Document

This doc included learning MSSQL with Google cloud, together. We can document everything here so that everyone has access to it and can learn and grow together.

## **Creating a MSSQL cloud Instance – by Gourav**

1. Enable the compute engine API.
2. GO to the Navigation menu and choose SQL, it will ask us to create new instance click on create new.
3. Choose SQL Server
4. It will ask us to choose the Cloud SQL edition which we want.

—>Cloud SQL Enterprise edition: provides all core capabilities of Cloud SQL and is suitable for applications requiring a balance of performance, availability, and cost.

—>Cloud SQL Enterprise Plus edition provides the best performance and availability to run applications requiring the highest level of availability and performance in addition to the capabilities of the Cloud SQL Enterprise edition.

To understand the best of these edition here is the official documentation below:

—> [Edition](https://cloud.google.com/sql/docs/editions-intro?authuser=2&hl=en&_gl=1*p7zd4f*_ga*MzA1ODI0NzQ3LjE3MjIxMDM5NTY.*_ga_WH2QY8WWF5*MTczMjY5OTcxMS41NS4xLjE3MzI3MDExMjAuNjAuMC4w#edition-features)-Doc

1. Choose the Database version – E.g. SQL server 2017,2019,2022.

—> Different MSSQL versions which are available on cloud SQL are:

SQL Server Standard: Balanced for general-purpose workloads.

SQL Server Enterprise: Optimized for large-scale and critical applications.

SQL Server Express: Lightweight, suitable for small applications.

SQL Server Web: Cost-effective for web hosting providers.

* Proceeding for Instance Configuration

1. Provide the Instance ID

The Instance ID is the unique name assigned to the Cloud SQL instance.

example———— GouravAppTest

1. Choose a strong password, It will be the password for the admin user.

Note: The default admin username in here Cloud will be ‘**sqlserver**’.

example————mssqlgourav

1. Region and Zone, Now let’s choose a region for our database.

—> Region: A region represents a geographical location where Google Cloud resources are hosted. Examples include us-central1, asia-south1, europe-west1. Each region contains multiple zones.

—> Zone: A zone is a specific data center within a region. Examples include us-central1-a, asia-south1-b, and so on.

Go through this official [Region&ZoneDoc](https://cloud.google.com/compute/docs/regions-zones) for more info.

* CLOUD MACHINE CONFIGURATION

1. MACHINE TYPE: The machine type defines the virtual hardware configuration (CPU and memory) allocated to our Cloud SQL instance. It directly impacts the performance, scalability, and cost of your database.  
   Here is a detailed documentation on [MACHINES TYPES](https://cloud.google.com/compute/docs/machine-resource) .
2. STORAGE TYPE: Storage is the disk space allocated to our Cloud SQL instance. It holds our databases, logs, and temporary files required for operations.

Choosing correct Storage type is very important for improved performance.

For now, here we are choosing 250 GB SSD.

Also, there is an option of automatic storage increase which will increase the storage when capacity is near to the threshold.

Since discussing on Cloud Storage here is also a slight overview of the [cloud storage bucket types](https://cloud.google.com/storage/docs/storage-classes?authuser=2&hl=en&_gl=1*1r7t182*_ga*MTE0MzkyNzIxMi4xNzMyNjI1MDMy*_ga_WH2QY8WWF5*MTczMzQyNjQyMC4xNi4xLjE3MzM0Mjc5OTAuNjAuMC4w), will be useful when creating buckets inside the cloud storage.

| Feature | Standard | Nearline | Coldline | Archive |
| --- | --- | --- | --- | --- |
| Access Frequency | Frequent | Monthly or less | Quarterly or less | Yearly or less |
| Storage Cost | High | Moderate | Low | Lowest |
| Retrieval Cost | None | Moderate | High | Highest |
| Minimum Duration | None | 30 days | 90 days | 365 days |
| Latency | Milliseconds | Milliseconds | Milliseconds | Higher (hours) |

1. Encryption:

Encryption ensures that our data is stored securely by converting it into a format that can only be read by someone with the correct decryption key. This protects the database content from unauthorized access.

Cloud SQL instances are encrypted using a Google-managed encryption key (GMEK) or Customer managed Encryption Key (CMEK)

Note: There is no TDE encryption supported for MSSQL cloud instances.

**GMEK vs CMEK**

|  |  |  |
| --- | --- | --- |
| **Feature** | **Google-Managed Key (GMEK)** | **Customer-Managed Key (CMEK)** |
| **Key Management** | Fully handled by Google | Managed by the customer using Cloud KMS |
| **Ease of Use** | Very simple, no additional setup | Requires manual setup and management |
| **Compliance Control** | Limited control over keys | Full control, suitable for strict compliance |
| **Risk of Data Loss** | Minimal, as Google ensures key availability | High, if keys are lost, disabled, or deleted |

1. Connections/Connectivity Settings:

Connectivity settings define how our application or client interacts with the Cloud SQL instance.

—> Public IP Address:

Assigns an external IP address to our instance, making it accessible over the internet.

Best suited for applications hosted outside Google Cloud.

For Example: Add the client machine’s IP address or corporate network range to the Authorized Networks list.

—> Private IP Address:

Assigns an internal IP address within our Virtual Private Cloud (VPC).

Recommended for applications hosted on Google Cloud or within the same VPC.

For brief info please go through this Official [MSSQLConnectivityDocument .](https://cloud.google.com/sql/docs/sqlserver/connect-overview?authuser=2&hl=en&_gl=1*mpkfzt*_ga*MzA1ODI0NzQ3LjE3MjIxMDM5NTY.*_ga_WH2QY8WWF5*MTczMjc5OTI4NS42NC4xLjE3MzI4MDEyMjAuNjAuMC4w)

1. Data protection: By default, it is always enabled when HA is there which includes automated backups and point in time recovery.

We can also choose the multi region feature to store the backup in case of disaster recovery.

1. Maintenance:

This option basically only takes place once every few months and requires the instance to be restarted while updates are made, which disrupts service briefly.

So, we should choose the week and window for that.

There is also an advanced option of deny maintenance period which allows zero maintenance and downtime for that window.

1. Database Flags and Parameters:

Database flags are configuration settings that modify the behavior of the MSSQL database instance. These flags allow us to fine-tune our instance for specific use cases, optimize performance, Enable, or disable specific SQL Server features.

[Documentation for database flags](https://cloud.google.com/sql/docs/sqlserver/flags?authuser=2&hl=en&_gl=1*1py673n*_ga*MzA1ODI0NzQ3LjE3MjIxMDM5NTY.*_ga_WH2QY8WWF5*MTczMjgwNzE2NS42NS4wLjE3MzI4MDcxNjUuNjAuMC4w)

Parameters include default settings like collation and time zone, which define the character set, sorting rules, and time configuration for the database. These settings are applied globally unless overridden at the database or query level.

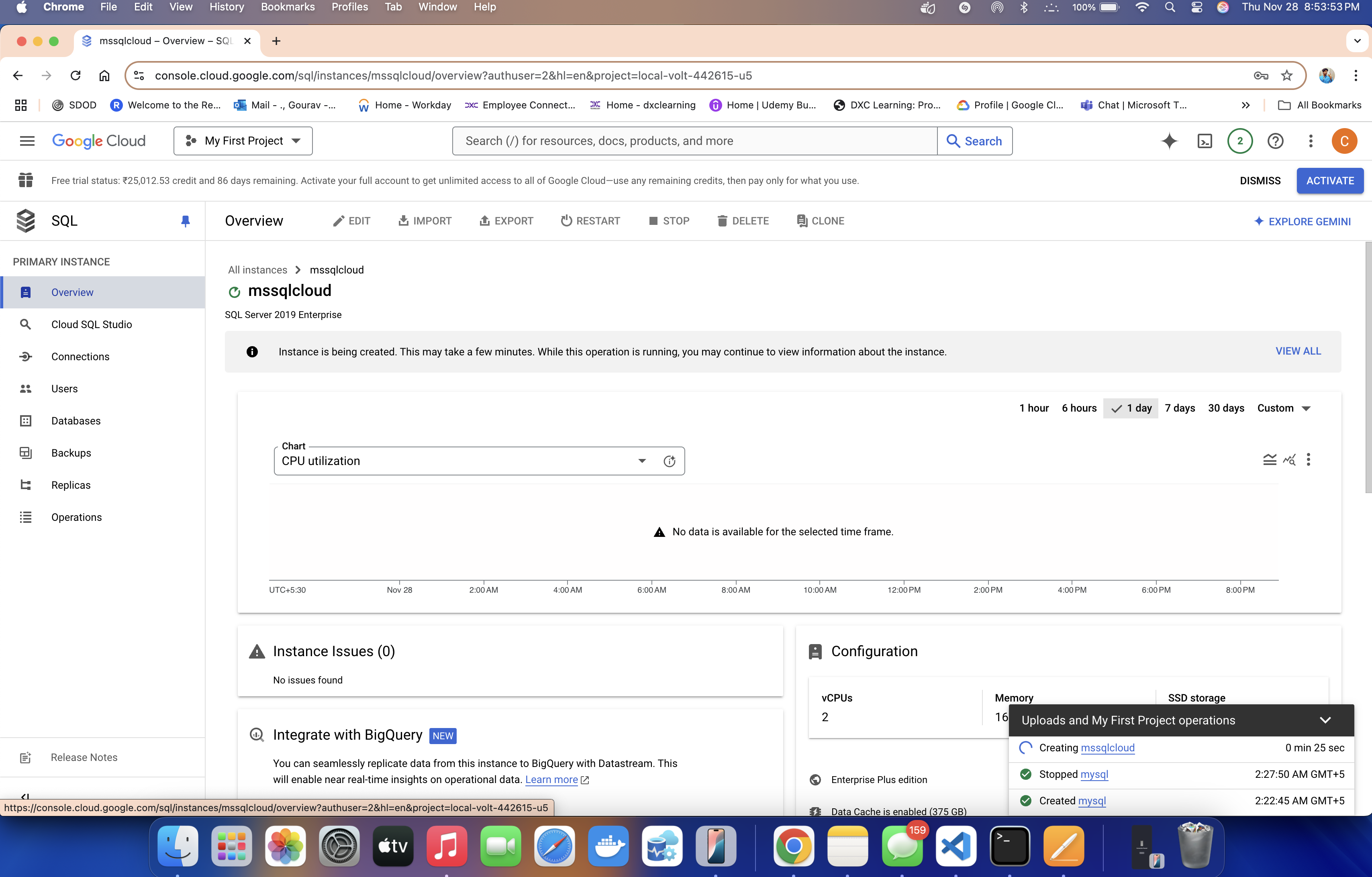
[Documentation of SQL server Instance Configuration](https://cloud.google.com/sql/docs/sqlserver/instance-settings?authuser=2&hl=en&_gl=1*2c1fi2*_ga*MzA1ODI0NzQ3LjE3MjIxMDM5NTY.*_ga_WH2QY8WWF5*MTczMjgwNzE2NS42NS4wLjE3MzI4MDcxNzMuNTIuMC4w#collation)

We can also enable [SQL server audit logs](https://cloud.google.com/sql/docs/sqlserver/db-audit?authuser=2&hl=en&_gl=1*18klwox*_ga*MzA1ODI0NzQ3LjE3MjIxMDM5NTY.*_ga_WH2QY8WWF5*MTczMjc5OTI4NS42NC4xLjE3MzI4MDI1MTEuNTkuMC4w) and store it on a cloud storage bucket .

1. Labels: (Optional)- Labels are key-value pairs used to organize, categorize, and manage Google Cloud resources. They act as metadata for your resources, making it easier to filter, search, and apply policies across multiple resources.

As for example key: value

env: production

env: staging

## **Connect to MSSQL cloud Instance using local client – by Gourav**

1. To connect the cloud instance from our local machine, SSMS or Azure Data studio or Visual studio code or any other client application which is not on same cloud, there are few network configurations to be made.
2. Enable Public IP for on Cloud SQL Instance
3. This can be done while creating the instance or we can edit the instance and enable it.
4. Retrieve the Local Machine's Public IP Address
5. Open a web browser and visit a site like WhatIsMyIP or search for "What is my IP" in Google and Noted own the public IPv4 address (e.g., 203.0.113.25).
6. Google Cloud SQL uses connection whitelisting to control which IPs can access the instance.
7. Inside our Cloud SQL console page go to Connections
8. Click on the Networking tab.
9. Add the IP Address to the Authorized Networks: (Preferred to give CIDR notation IP address).
10. Provide a name for the IP (e.g., Local Machine Access).
11. Save Changes
12. Now once instance settings are updated go back to the SQL cloud console page overview.
13. Get the Public IP address. Copy it
14. Open the Client (SSMS/VS code / Azure Data studio) provide the IP in instance name.
15. Provide the credentials, default login name - sqlserver
16. And then connect.

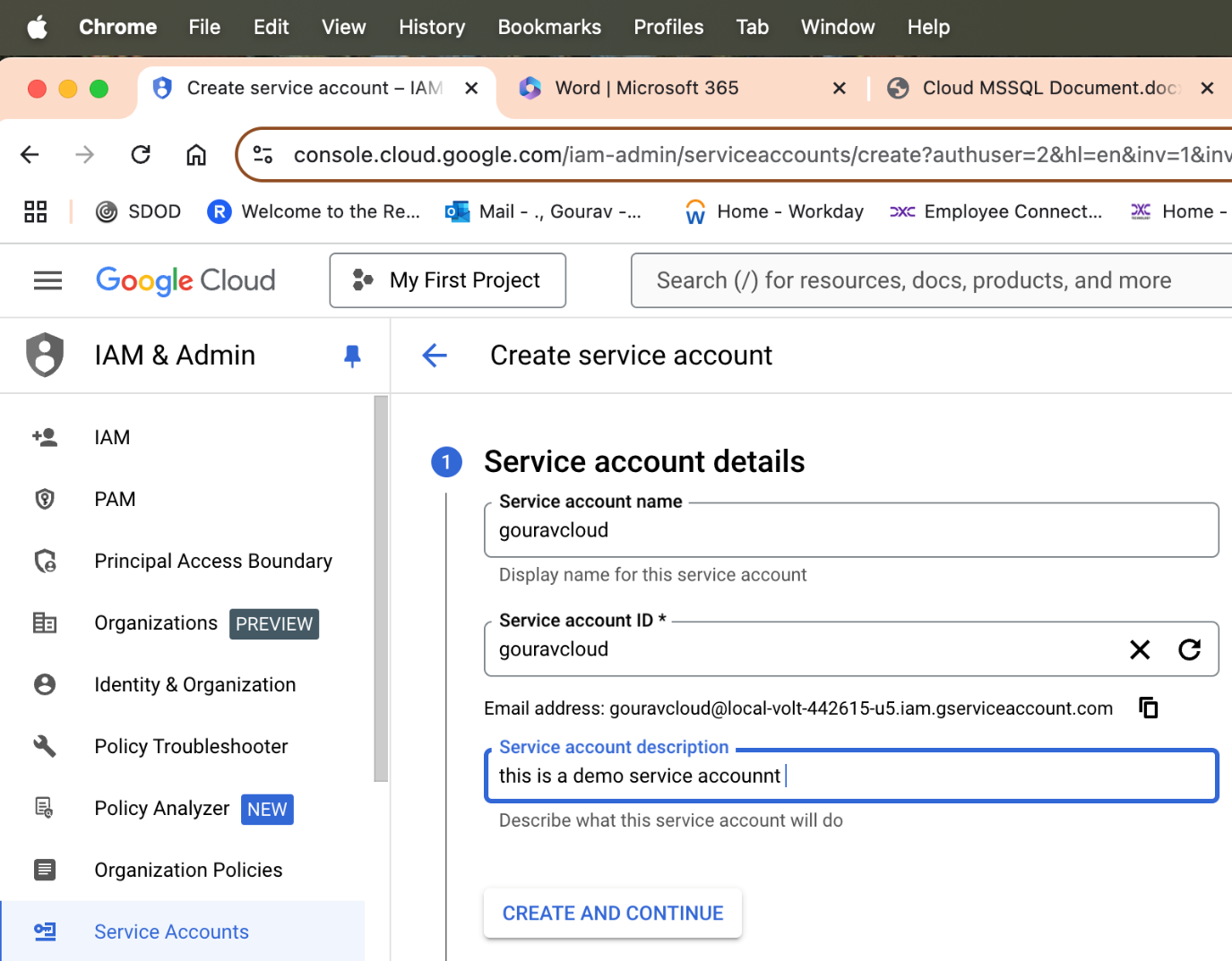
## **Connect to MSSQL cloud Instance using Cloud SQL auth Proxy – by Gourav**

1. This method involves connecting to the cloud SQL instance from our local machine using Cloud SQL Auth Proxy method.
2. This eliminates the process of whitelisting our IP in the Cloud SQL network tab, which we did in above steps which makes it a secure way of connecting .
3. The first step is creating a service account on our google cloud console.

Creating a service account is important not only for this step but also for other uses so let us break down the creation and understand it in a detailed way.

**SERVICE ACCOUNT CREATION:**

When we create a service account, it is kind of a "robot account" used by applications or automated systems to access Google Cloud resources. Without roles, the service account has no permissions by default. Assigning roles allows the service account to perform specific actions, like reading data from a database, uploading files to Cloud Storage, or managing resources.

* Go to Console page, Overview --> IAM & Admin --> Service accounts
* Click on Create a service account
* Fill in the below details.
* 
* Next after clicking create and continue,
* The next dialog box is Grant this Service Account Access to Project (Optional) -- this is Recommended to fill.
* What this means is Here, we assign roles to the service account, defining its permissions.

Roles are sets of permissions grouped together to make it easier to assign access. Instead of manually granting individual permissions, we assign roles, which contain pre-defined permissions.

Roles in Google Cloud are classified into three main types:

1. **Primitive Roles (Broad, Deprecated for Most Cases)**

These roles apply to the entire project and are too broad for most scenarios:

--Viewer: Read-only access to all resources.

--Editor: Read and write access to all resources, except permissions and roles.

--Owner: Full control of all resources, including managing roles and permissions.

We should Avoid using these roles unless necessary. They grant excessive permissions, violating the principle of least privilege.

1. **Predefined Roles (Granular and Specific)**

Predefined roles are tailored for specific Google Cloud services, giving access to only the necessary permissions. Here are some examples based on common use cases:

|  |  |  |
| --- | --- | --- |
| **Use Case** | **Predefined Role** | **Permissions Included** |
| **Cloud SQL (Databases)** | Cloud SQL Client | Read/write data in SQL instances (but not manage the instances). |
|  | Cloud SQL Admin | Full management of SQL instances, backups, and data (administrative access). |
|  | Cloud SQL Viewer | Read-only access to instance configurations (cannot view data or make changes). |
| **Storage (Files)** | Storage Object Viewer | Read-only access to Cloud Storage buckets. |
|  | Storage Admin | Full control of storage buckets and files. |
| **Compute Engine (VMs)** | Compute Viewer | Read-only access to virtual machine instances and networks. |
|  | Compute Admin | Full control over virtual machines, disks, and networks. |
| **Kubernetes** | Kubernetes Engine Viewer | Read-only access to Kubernetes clusters. |
|  | Kubernetes Engine Admin | Full administrative access to Kubernetes clusters. |
| **BigQuery** | BigQuery Data Viewer | Read-only access to BigQuery datasets. |
|  | BigQuery Admin | Full access to manage datasets, tables, and queries. |
| **IAM and Security** | IAM Viewer | View all IAM roles and permissions but cannot edit. |
|  | Security Admin | Manage security policies, firewalls, and rules. |

1. **Custom Roles (Highly Specific to particular use case and security policies)**

If predefined roles do not fit our needs, we can create a **custom role** with exactly the permissions we need. This is especially useful for applications with extremely specific requirements.

* Now next there is also an option for --- IAM condition (optional), this is a very enhanced security features for the service account and its roles we define.

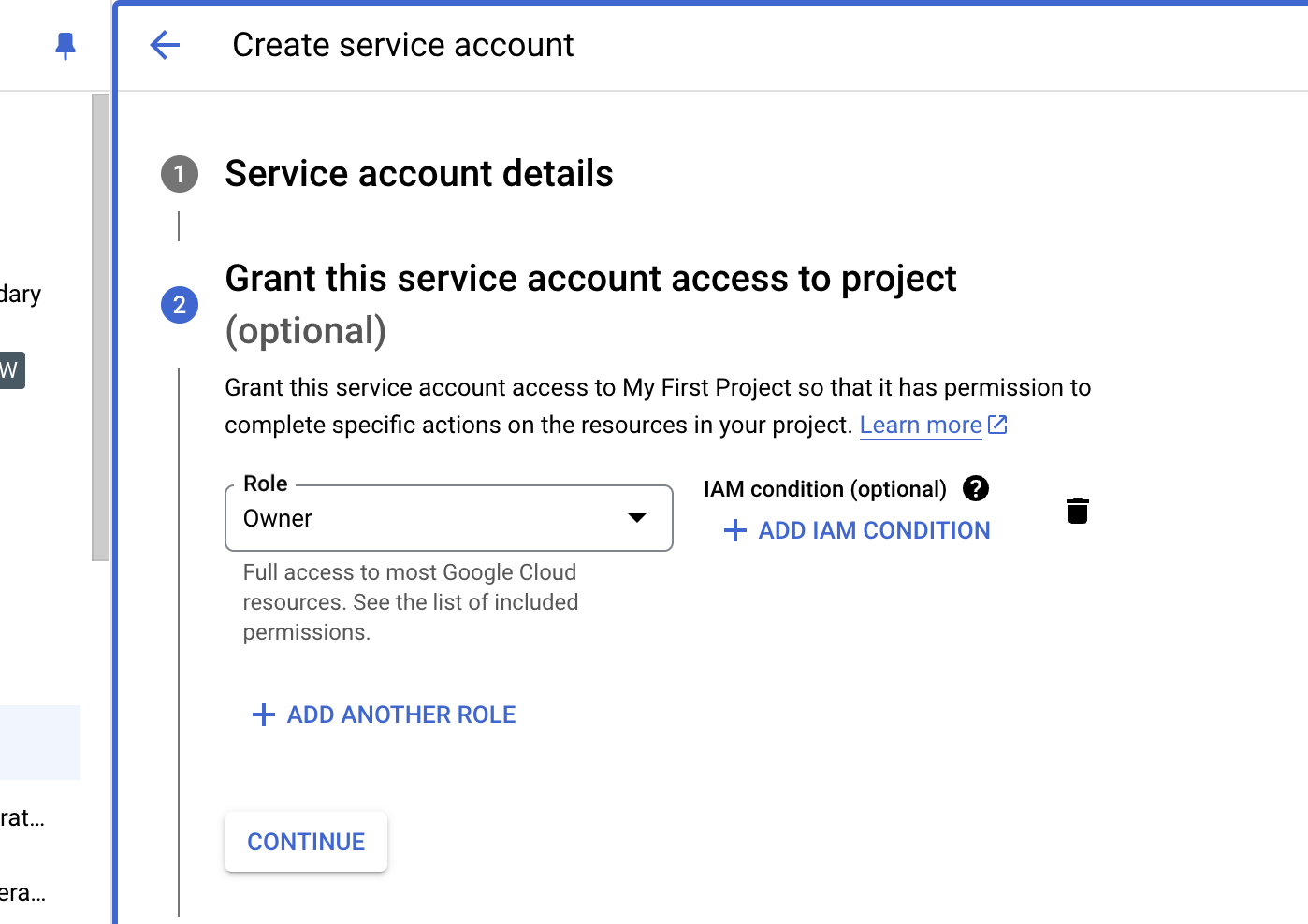
It allows us to create fine-grained access controls for our service account (or any IAM principal). It lets us apply conditional logic to the roles or permissions we assign to restrict their usage based on certain criteria.

For example:

Grant access only during specific timeframes (e.g., working hours).

Restrict access to specific resources or operations.

Ensure access is granted only when certain request attributes are satisfied (e.g., IP address or geographic location).



Since this is a demo purpose account i choose to give owner role, but its highly recommended to go with the principle of Lease privileges to ensure better security and integrity of the resources on cloud.

* Grant users access to this service account (optional)

This step is about giving specific users or groups the ability to use or manage our service account.

For example: We might want **other team members** to use this service account for running scripts or jobs. This step is **optional**, but it can be useful if we are working in a team or need others to help with managing or using the service account.

* Main Things we Can Let Others Do use this feature is

-- Use the Service Account (Run things as the service account).

Example: If someone is running a job that needs to connect to a database, they can impersonate (act as) this service account.

-- Manage the Service Account (Full control).

Example: If an admin wants to create or delete this service account or manage its keys.

* Enter the Gmail of the user and then enter the role. These roles can as below:

|  |  |  |
| --- | --- | --- |
| **Role Name** | **Purpose** | **When to Use** |
| Service Account User | Allows impersonation of the account | When users need to run jobs or scripts using the service account. |
| Service Account Admin | Full management of the service account | For administrators who need to create, delete, or modify service accounts. |
| Service Account Key Admin | Manage keys for the service account | For secure key creation, rotation, and deletion. |
| Viewer | View service account details | When someone needs read-only access to the account. |

* Click next and continue, it is created successfully. PFB reference
* Now service account is created let's create up a key for the account.

The key is used to use this service account in scripts or applications (e.g., for authentication), we need to generate a key.

We can download the key in JSON format, which will be used to authenticate your applications.

* To generate a key:

Go to the Service Accounts section in Google Cloud.

Select our newly created service account.

Click Key and choose Add key ---> create new key and choose file type as json .

Download and securely store the key in a private folder.

Coming back to our topic of Connecting to MSSQL cloud Instance using Cloud SQL auth Proxy.

1. Make sure could sql proxy is installed along with Google cloud CLI and SDK tools.
2. Please go through google and YouTube videos to install these tools on your OS.
3. After that confirm it by running below command: cloud-sql-proxy –version
4. Once it confirms the version, we can proceed with establishing the connection.
5. NOTE: This method does not involve IP whitelisting
6. Give below command to establish the connection local machine and cloud instance.

cloud-sql-proxy my-project:us-central1:my-instance --credentials-file=/path/to/keyfile.json

1. If everything and setup was done without any issue, we will see a message as below :  
   2024/12/14 12:28:58 The proxy has started successfully and is ready for new connections!
2. After that open, the IDE: VS code or azure data studio or SSMS or SQLCMD or any other software to connect to the databases on cloud.
3. After giving the required credentials, we can connect to the database. We dont have to be IP whitelisted or dependent for this, and it allows secure connection too.
4. NOTE: This connection will work till our terminal window is active, once we close the terminal window the connections drop.

## **Migrating a database from local instance to MSSQL cloud – BY GOURAV**

| **Method** | **Downtime** | **Complexity** | **Use Case** | **Tools** |
| --- | --- | --- | --- | --- |
| **Backup and restore** | High | Medium | Best for full database migration with downtime allowed. | Native MSSQL tools (SSMS, T-SQL) |
| **Data Migration Assistant** | Medium | Low | Great for schema + data migration with compatibility checks. | Data Migration Assistant (free by Microsoft) |
| **SSMS Export/Import** | Low | Low | Quick, manual approach for smaller databases. | SQL Server Management Studio (SSMS) |
| **Third-Party Tools** | Low | Medium | Ideal for large/complex databases with fine control. | **Redgate SQL Compare/SQL Data Compare** (paid), **DBForge Studio for SQL Server** (paid), **Quest Toad for SQL Server** (paid), **Azure Data Studio** (free) |

1. **Conventional backup and restore method:**
2. Let’s do the most easy and conventional way for migration.
3. Firstly, take the full backup of the database on Source instance (local/on premises)
4. Then connect to your Google cloud console page.
5. Go to navigation menu—Cloud Storage
6. Choose Buckets – Create new!
7. Pick a globally unique, permanent name for the bucket. [Naming guidelines](https://cloud.google.com/storage/docs/buckets?authuser=2&hl=en&_gl=1*1gjf65y*_ga*MTE0MzkyNzIxMi4xNzMyNjI1MDMy*_ga_WH2QY8WWF5*MTczMzQyNjQyMC4xNi4xLjE3MzM0MjczMTIuNTguMC4w#naming)
8. Tick the box and Enable Hierarchical namespace on this bucket. This fulfills the need of logical folder structures. (folders/subfolders)
9. Click continue and move to Choose where to store the data.

This decision affects cost, performance, and data availability and cannot be changed later for the bucket.

à Here is a detailed in this [official document](https://cloud.google.com/storage/docs/locations?authuser=2&hl=en&_gl=1*1yhepqu*_ga*MTE0MzkyNzIxMi4xNzMyNjI1MDMy*_ga_WH2QY8WWF5*MTczMzQyNjQyMC4xNi4xLjE3MzM0Mjc2NTQuNjAuMC4w) .

For test and learn purpose here we are choosing single region.

1. Choose a storage class, explained, and attached details and documentation in the installation part. Refer above. (STEP 10).

Click continue.

1. Choose Uniform Access—this is easy to manage the access over folders and subfolders in the bucket.

It is conceptually like how Windows OS manages access when we apply permissions to a folder and choose to propagate those permissions to all its subfolders and files.

1. Next choose the data protection, keep it same as predefined and click create.
2. Once the bucket in created in few seconds, let’s open the bucket which we created now.
3. Crete a folder with any name.
4. Open the folder (/localbackup). Click on upload.
5. Upload Files --- Chose the place where the backup is placed. And upload it
6. Note: its only suitable for smaller datasets as the larger datasets require more time to upload to bucket and thus will cause more downtime,
7. Once the backup is placed, Go to Navigation menu in console page.
8. Open SQL --- Click on SQL instance (devapp).
9. If not started, start the SQL server.
10. In the overview page, on top choose Import.
11. Source chose the bucket, open bucket, open folder and choose the .bak file (fullbackup).
12. In Destination give the name of a new database which will be created, and we will restore the .bak file.
13. Click Import,
14. Wait for some time depending on size of databases it can take time too.
15. Once import Is successful, connect to the instance from SSMS, Cloud SQL studio or any other client and check if new database is created.
16. Congrats we have Successfully migrated our database from on premises to cloud.

1. **Using SSMS Export/Import Wizard:**
2. Before proceeding for this migration setup, we must ensure that our local machine has the required access to establish a connection to Cloud MSSQL instance.
3. Open SSMS, connect to the local instance i.e. we will refer to it here as Source instance
4. Right-click the database that needs to be migrated, then select:

Tasks > Export Data... (for exporting schema and data)

1. In the SQL Server Import and Export Wizard, click Next.

Under Choose a Data Source:

Data Source: Microsoft OLE DB Driver for SQL Server.

Server Name: Enter the name of local MSSQL instance or enter localhost.

Authentication: Use the appropriate credentials (Windows/SQL Server authentication).

Database: Select the database which has to be migrated.

Click Next.

1. Under Choose a Destination:

Destination: Microsoft OLE DB Driver for SQL Server

Server Name: Enter the Public IP address or hostname of the Google Cloud MSSQL instance.

Authentication: Use the appropriate credentials for the cloud instance.

Database: Choose the target database or create a new one.

Click Next.

1. Specify Table Copy or Query

Under Specify Table Copy or Query, choose one of the following options:

Copy data from one or more tables or views: Select this to migrate entire tables.

Write a query to specify the data to transfer: Use this to migrate specific data subsets.

Click Next.

1. Select Tables and Views

The wizard displays all tables and views in the source database.

1. Select the tables/views which ever need to be migrated.

Optionally, click on Edit Mappings:

Review and adjust column mappings if necessary.

Ensure primary keys and constraints are correctly mapped.

If needed, enable the option to delete existing data in the target table before inserting.

Click OK and then Next.

1. Configure the Data Transfer: - This Chooses how to run the data transfer:

Run immediately: Executes the transfer right away.

Save SSIS Package: Saves the configuration as an SSIS package for reuse.

1. Review the summary of migration settings.
2. Click Finish to start the data migration process.