Migration from AWS RDS to Cloud SQL by DMS
This doc is used for demonstrating the possibility of using DMS to migrate data from AWS RDS to Cloud SQL.

Find the necessary information from following links anytime:

- DMS: https://cloud.google.com/database-migration/docs/mysql
- MySQL on Amazon RDS: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/CHAP_MySQL.html

[Apply for Joining DMS White List]

1. Email to cloud-dms-pm@google.com, provide project name and ID.
2. After got the approval, follow "before you begin" in this doc https://cloud.google.com/database-migration/docs/mysql/quickstart#before-you-begin, select project, enable API, etc.

[Create AWS RDS]
(Change following information according to your case)
Account:

- `ID: 46479352....`
- `User: demo....`
- `Pass: 11....`

Region:

- `Mumbai`

RDS:

- DB Engine:
  - `Community MySQL 5.7.30`
- Production:
  - `Yes (Master + Slave in multi-AZ)`
- DB Info:
  - `DB Instance Identifier: source (`**`Don't`**` use a long name, because DMS requires the host name of source DB no more than 60 characters. `**`Otherwise`**`, you need to create a DNS redirect, follow this guide` https://cloud.google.com/database-migration/docs/mysql/create-source-connection-profile`)`
  - `User name: admin`
  - `Password: Mid.....`

    - Endpoint: source.cubfukv3pdnl.ap-south-1.rds.amazonaws.com
- Instance size:
    - m5.xlarge (4 vCPU, 16G RAM)
- Storage:
    - Provisioned IOPS (SSD)
    - 100G (1000 IOPS)
- Connectivity:
    - Default VPC
    - Default Subnet
    - Public Access: **NO**
    - Port: 3306
- Configuration:
    - DB Parameter Group: Default MySQL 5.7 (you will need to create a custom parameter group later, see [Configure AWS RDS] part)
    - Option Group: Default MySQL 5.7
    - Backup:
        - Enable automatic backups: **Yes** (for enabling bin-logging)
        - Backup retention period: 7 days
    - Enable Encryption (option): Yes
    - Log Export (option): Enable publishing error/general/slowquery logs to CloudWatch Log

[Import Test Data]

1. Create an EC2:
    a. OS: Amazon Linux
    b. In the same VPC with RDS
    c. Public accessible with public IP (for reliability reason, EIP is better)
    d. Open SSH 22 to anywhere (for security reasons, you can open to specific IPs only) in EC2 security group.
    e. Download PEM file for SSH connection later.
2. Configure RDS security group, allow inbound traffic from EC2 security group, on TCP 3306.
3. SSH to EC2.
4. Install MySQL client:

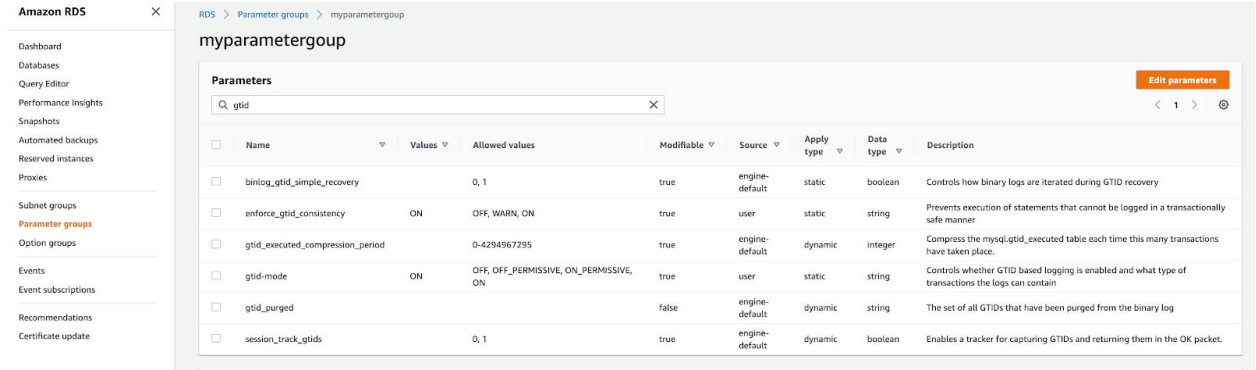
```
$ sudo yum install -y
https://dev.mysql.com/get/mysql57-community-release-el7-11.
noarch.rpm
$ sudo yum install -y mysql-community-client
```

5. Download test data:
```
$ git clone https://github.com/datacharmer/test_db.git
$ cd test_db
```
6. Connect to RDS:
```
$ mysql -h source.cubfukv3pdnl.ap-south-1.rds.amazonaws.com
-P 3306 -u admin -p
(enter your password when prompt)
```
7. Import data:
```
$ mysql> source employees.sql;
```
8. Check result:
```
$ mysql> source test_employees_md5.sql;
```

[Configure AWS RDS]

Follow this guide
https://cloud.google.com/database-migration/docs/mysql/configure-source-database :

1. Stop all DDL operations
2. Set server-id to 1 or larger:
   AWS RDS assigned a server-id automatically, you can find it by:
   ```
   $ mysql> SELECT @@server_id
   ```
3. Set gtid_mode to ON or OFF:
   Find more details here:
   https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/mysql-replication-gtid.html#mysql-replication-gtid.parameters
   Attention 1: Create a new custom parameter group instead of the default one.
   Attention 2: Set both gtid_mode and enforce_gtid_consistency to ON.



   Attention 3: Need to reboot the database after the modification.
4. Check the user used to connect the database has all following privileges:

```
REPLICATION SLAVE, SELECT, SHOW VIEW, REPLICATION CLIENT,
RELOAD, EXECUTE.
```

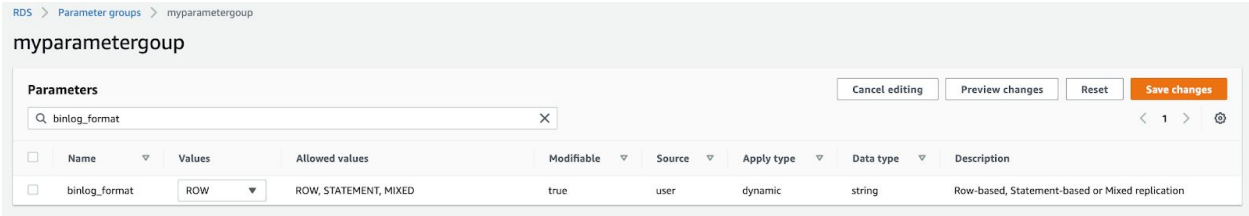
The default RDS user ‘admin’ has had all required privileges:

```
$ mysql> show grants for admin;
| Grants for admin@%
|
+-------------------------------------------------------------+
| GRANT SELECT, INSERT, UPDATE, DELETE, CREATE, DROP,
RELOAD, PROCESS, REFERENCES, INDEX, ALTER, SHOW DATABASES,
CREATE TEMPORARY TABLES, LOCK TABLES, EXECUTE, REPLICATION
SLAVE, REPLICATION CLIENT, CREATE VIEW, SHOW VIEW, CREATE
ROUTINE, ALTER ROUTINE, CREATE USER, EVENT, TRIGGER ON *.*
TO 'admin'@'%' WITH GRANT OPTION |
```

5. Enable binary logging and set retention to a minimum of 2 days:

```
$ mysql> call mysql.rds_set_configuration('binlog retention
hours', 168);
```

6. (**Important!!!**) In the custom parameter group, set binlog_format as ROW:



7. Make sure the InnoDB storage engine is the default:

```
$ mysql> show engines;
+--------------------+---------+-------------------------------------------------------------+--------------+------+------------+
| Engine             | Support | Comment                                                     | Transactions | XA   | Savepoints |
+--------------------+---------+-------------------------------------------------------------+--------------+------+------------+
| InnoDB             | DEFAULT | Supports transactions, row-level locking, and foreign keys  | YES          | YES  | YES        |
| MRG_MYISAM         | YES     | Collection of identical MyISAM tables                       | NO           | NO   | NO         |
| MEMORY             | YES     | Hash based, stored in memory, useful for temporary tables   | NO           | NO   | NO         |
| BLACKHOLE          | YES     | /dev/null storage engine (anything you write to it disappears) | NO           | NO   | NO         |
| MyISAM             | YES     | MyISAM storage engine                                       | NO           | NO   | NO         |
| CSV                | YES     | CSV storage engine                                          | NO           | NO   | NO         |
| ARCHIVE            | YES     | Archive storage engine                                      | NO           | NO   | NO         |
| PERFORMANCE_SCHEMA | YES     | Performance Schema                                          | NO           | NO   | NO         |
| FEDERATED          | NO      | Federated MySQL storage engine                              | NULL         | NULL | NULL       |
+--------------------+---------+-------------------------------------------------------------+--------------+------+------------+
```

[Create a Source Connection Profile]

Follow this guide https://cloud.google.com/database-migration/docs/mysql/create-source-connection-profile :

Source database engine
Amazon RDS for MySQL

Connection profile name *
myconnectionprofile
Less than 60 characters. 19/60

Connection profile ID *
myconnectionprofile
Lowercase letters, numbers, or hyphens. Must be unique in this project. 19/60

Hostname or IP address *
source.cubfukv3pdnl.ap-south-1.rds.am

Port *
3306

Username *
admin

Password
••••••••
CHANGE PASSWORD

Secure your connection

Choose an encryption type, and you'll see the SSL/TLS details needed. Learn more

Encryption type
None
UPDATE CERTIFICATES

SAVE CANCEL

[Create a Migration Job]
Follow this guide:
https://cloud.google.com/database-migration/docs/mysql/create-migration-job

Attention 1:
Due to the limitation from organization, maybe you cannot select the same region as the source database (in my case, AWS RDS in Mumbai) to place the destination database, in that case, choose one of the nearest region (in my case, Cloud SQL in Singapore):

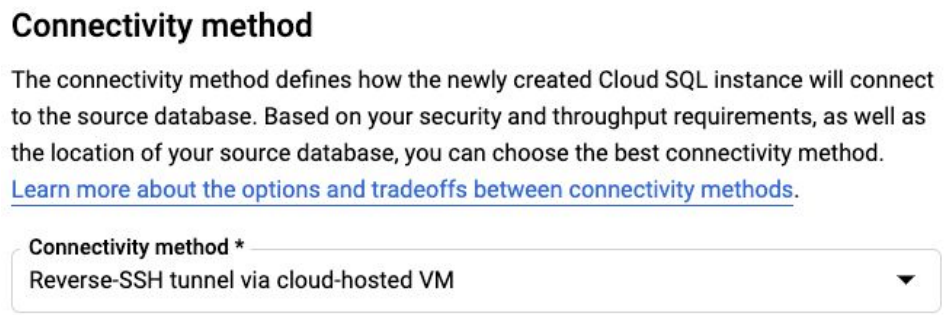
Source database engine
Amazon RDS for MySQL

Destination database engine
Cloud SQL for MySQL

Region
asia-southeast1

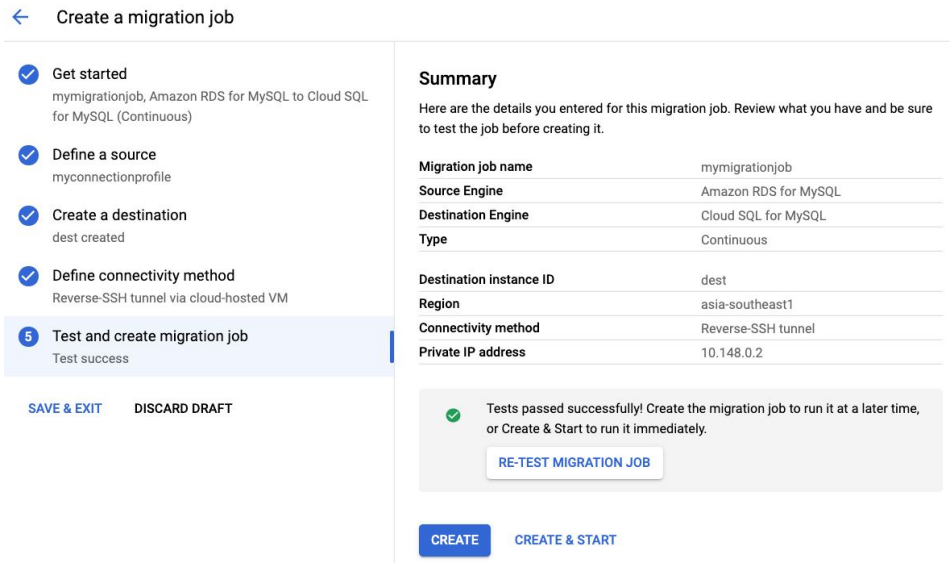
Attention 2:

In my case, because AWS RDS has no public access from internet, I choose reverse SSH tunnel as the connectivity method:



Regarding how to build up a reverse SSH tunnel, see [Configure Connectivity] part below.

Attention 3:
In step 5 (the final step), before clicking the “CREATE” button, suggest doing test by click “TEST MIGRATION JOB” button:

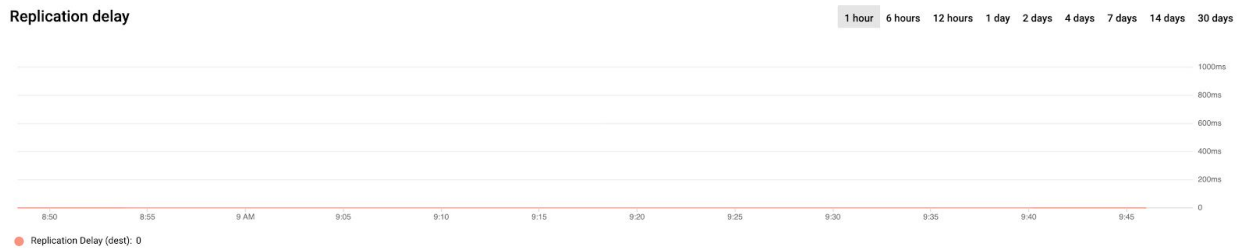


[Start the Migration Job]
After the migration job was created and passed the test, you can start the job anytime from the Migration Jobs page.

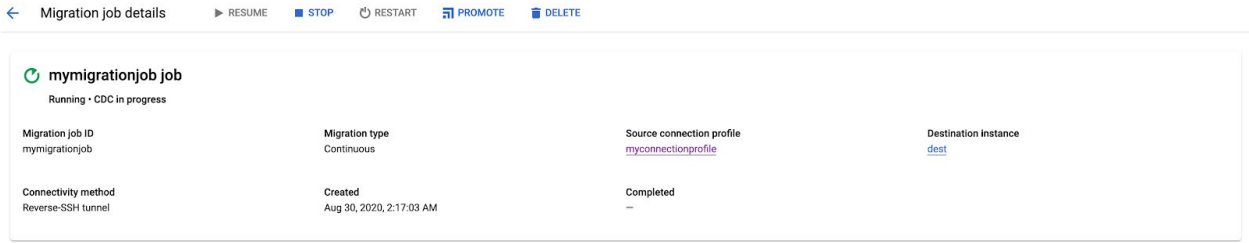
Then, the job status will change to “Running - Full dump in progress”:

After the dump completed, the job status will change to “Running - CDC in progress” (in my case, I chose CDC model):

When the replication delay equals to 0:



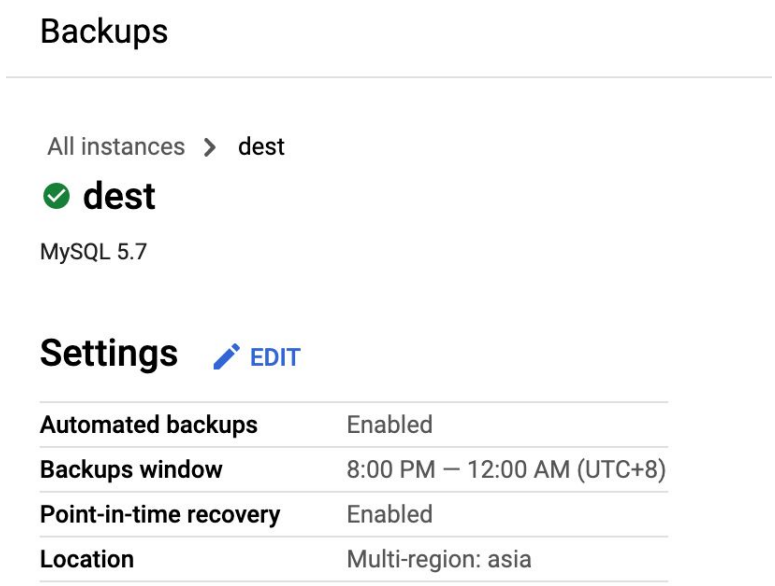
You can promote Cloud SQL to primary database, by click “PROMOTE” button below (this will also disconnect with AWS RDS) :



When you see the migration job status becomes “Migration job is completed”, that means the whole migration has completed. Congrats!

[Verify the Migration]
Follow this guide https://cloud.google.com/database-migration/docs/mysql/verify-migration to verify if the migration is successful.

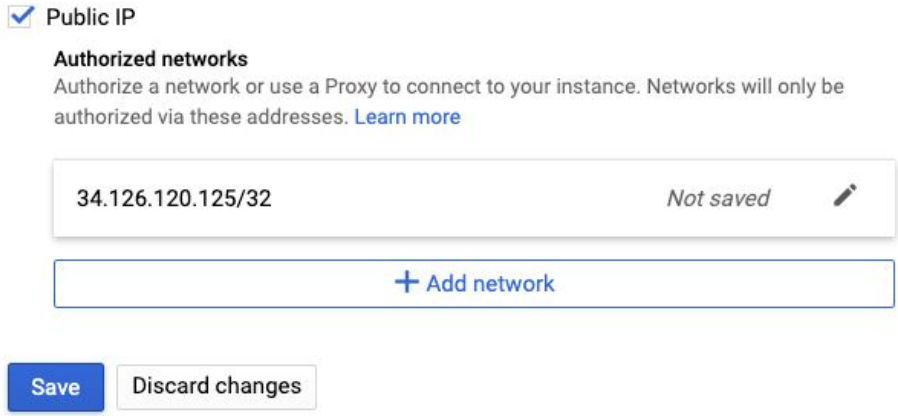
Attention 1:
When connecting to Cloud SQL, if you encountered an `ERROR: (gcloud.sql.connect) HTTPError 400: Invalid request: Binary log must be disabled when backup is disabled`, try to enable Cloud SQL automated backups, then connect again:



Attention 2:
When connecting to Cloud SQL by Cloud Shell, encountered the following error:

```
$ gcloud sql connect dest --user=root --quiet
ERROR: (gcloud.sql.connect) HTTPError 400: Invalid request:
Organization Policy check failure: the authorized networks of
this instance violates the
constraints/sql.restrictAuthorizedNetworks enforced at the
884952457849 project.
```

Changed to use a VM with MySQL shell to access Cloud SQL, but when adding the VM static IP to Cloud SQL authorized networks, and clicked “Save”:



Encountered the following error:

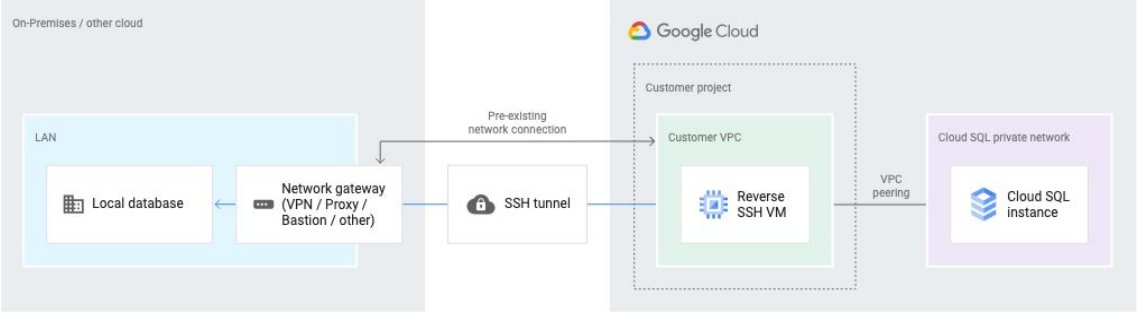
```
Operation failed: Invalid request: Organization Policy check
failure: the authorized networks of this instance violates the
```

`constraints/sql.restrictAuthorizedNetworks enforced at the 884952457849 project.`
Solution: Using a VM with MySQL shell to access Cloud SQL, but use Cloud SQL instance's **private IP** instead of the public ones:
`$ mysql --host=[private IP] --user=root --password`

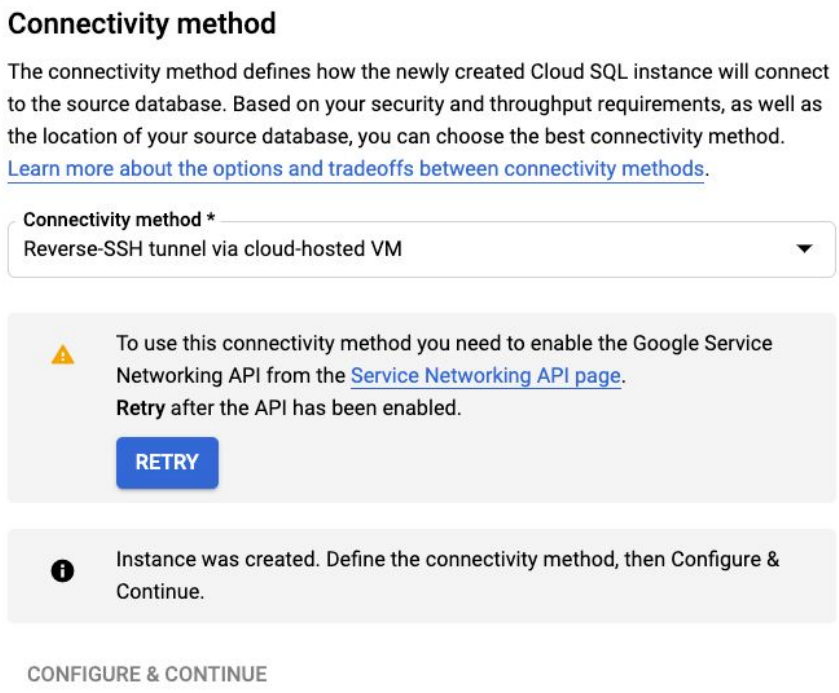
Tips: if you are following the tutorial to install MySQL client on VM, change the command `sudo apt-get install mysql-client` to `sudo apt-get install mariadb-client`.

Note: When you use the root account to access the destination Cloud SQL database instance for the first time, no password is set for the account. Afterward, for security purposes, set a password for the account.

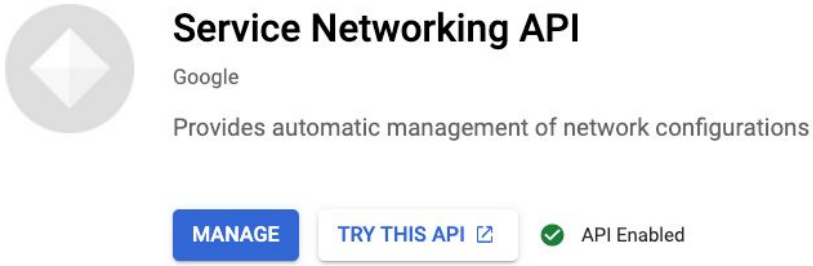
[Configure Connectivity]
Follow this guide:
https://cloud.google.com/database-migration/docs/mysql/configure-connectivity#reverse-ssh-tunnel to establish a reverse SSH tunnel between Cloud SQL and AWS RDS:



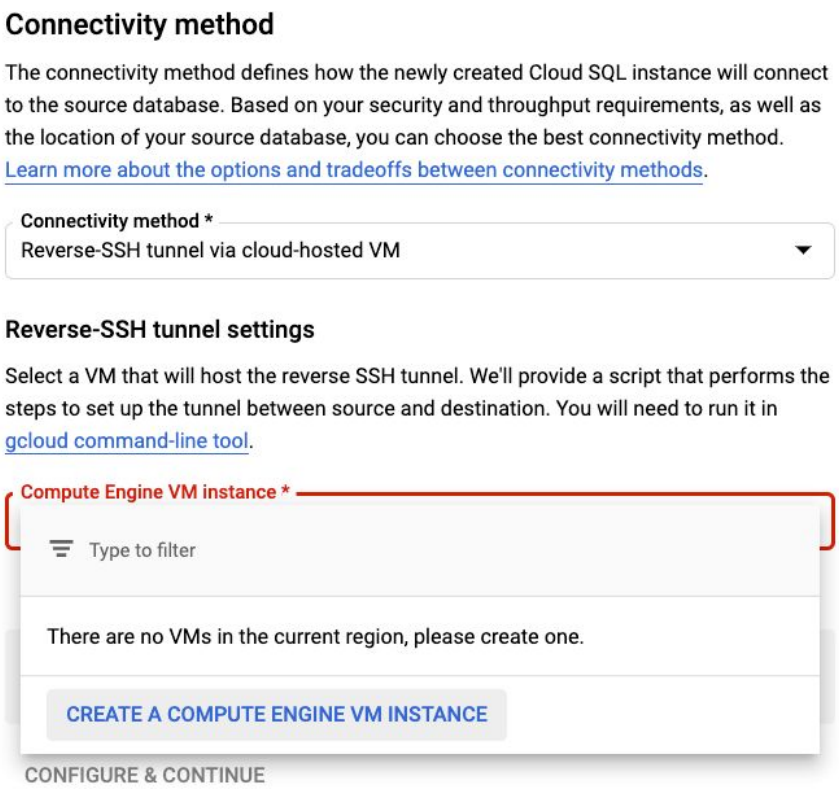
1. Enable Google Service Networking API:
   Click "Service Networking API page" like on Connectivity Method page:



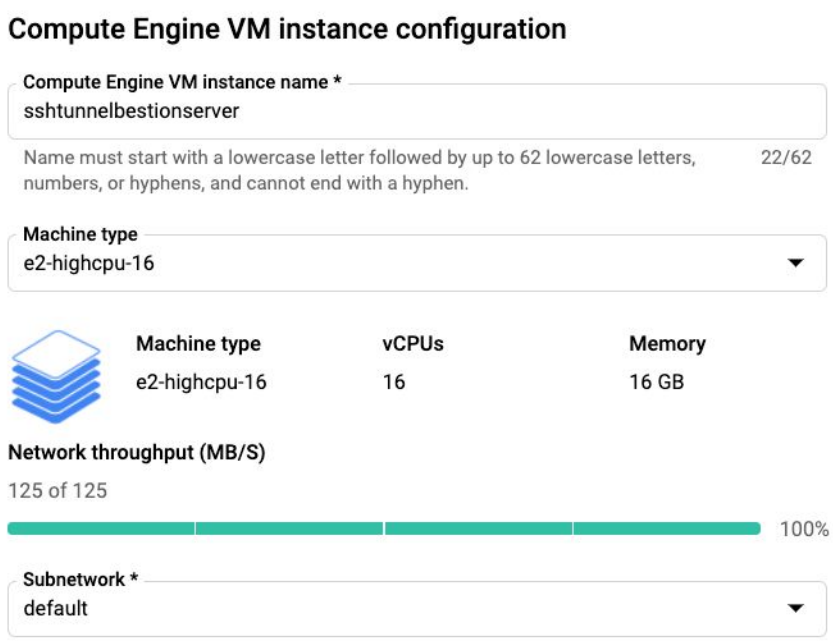
Enable Service Networking API:



2. Reverse-SSH tunnel settings:
   2.1. Click “Create a Compute Engine VM Instance” in the dropdown list, this will create a VM as the SSH tunnel bastion server in GCP side:



2.2. Input instance name, select machine type and VPC:



2.3. Copy the script down:

```
#!/bin/bash
set -ex
```

```
export VM_NAME=sshtunnelbestionserver
export PROJECT_ID=harrishegcgsxonoticdemoingcct
export VM_ZONE=asia-southeast1-b
export VM_MACHINE_TYPE=e2-highcpu-16
export SUBNET_NAME=default
export VM_PORT=3306
export
SOURCE_DB_LOCAL_IP=source.cubfukv3pdnl.ap-south-1.rds.amazo
naws.com
export SOURCE_DB_LOCAL_PORT=3306

echo "Creating a virtual machine (VM) instance named
'${VM_NAME}' in zone '${VM_ZONE}'"

gcloud compute instances create "${VM_NAME}" --machine-type
"${VM_MACHINE_TYPE}" --zone "${VM_ZONE}"
--project="${PROJECT_ID}" --subnet "${SUBNET_NAME}"

gcloud compute ssh "${VM_NAME}" --zone="${VM_ZONE}"
--project="${PROJECT_ID}" -- 'echo "GatewayPorts yes" |
sudo tee -a /etc/ssh/sshd_config && sudo service ssh
restart'

private_ip=$(gcloud compute instances describe "${VM_NAME}"
--zone="${VM_ZONE}" --project="${PROJECT_ID}"
--format='get(networkInterfaces[0].networkIP)')

echo "VM instance '${VM_NAME}' created with private ip
${private_ip}"


echo "Setting up SSH tunnel between the source and the VM
on port '${VM_PORT}'"

gcloud compute ssh "${VM_NAME}" --zone="${VM_ZONE}"
--project="${PROJECT_ID}" -- -f -N -R
"${VM_PORT}:${SOURCE_DB_LOCAL_IP}:${SOURCE_DB_LOCAL_PORT}"

if [[ "$?" -eq 0 ]]; then
  echo "SSH tunnel is ready on port ${VM_PORT}"
fi
```

2.4. SSH to the EC2 instance created in [Import Test Data] part, which will serve as the bastion server on the AWS side.
2.5. Install the gcloud command-line tool on EC2.
2.6. Run the copied script on EC2.
(**Important!!!**) Before executing the command:

```
gcloud compute ssh "${VM_NAME}" --zone="${VM_ZONE}"
--project="${PROJECT_ID}" -- 'echo "GatewayPorts yes" |
sudo tee -a /etc/ssh/sshd_config && sudo service ssh
restart',
```

Make sure you have already set up a such firewall:

```
gcloud compute firewall-rules create [firewall name] \
    --network [VPC name] \
    --action allow \
    --direction ingress \
    --rules tcp:22 \
    --source-ranges [EC2 public IP/32] \
    --priority 1000 \
    --target-tags [network tag of SSH tunnel bastion
server]
```

3. Destination instance settings:
Copy the VM private IP from the script output, enter it below, then click "CONFIGURE & CONTINUE" button. This will establish a private connection from the VM to Cloud SQL.