Datometry® Hyper-Q™ for Google BigQuery Installation and Set-up

Version 3.40 and later releases
June 9, 2021

Part No. DTMY-2002.1-2106

Table of Contents

1 About Datometry Hyper-Q Installation for Goggle BigQuery

Scope of Document

Datometry Hyper-Q for Goggle BigQuery Installation and Set-up describes how to install Datometry® Hyper-Q™ for use with the Goggle Cloud BigQuery database. This document describes:

- How to create a virtual machine (VM) on Goggle Cloud Platform.
- Install all required software packages for Hyper-Q.
- Configure a Goggle Cloud load balancer for use with Hyper-Q.
- Configure Hyper-Q to work with Goggle Cloud and BigQuery.

Intended Audience

This document is intended for experienced Linux system administrators who are familiar with Goggle Cloud, virtual machine technology, and data center operations.

2 What is Datometry Hyper-Q?

Datometry Hyper-Q is a database virtualization platform that makes Teradata applications instantly interoperable with Google Cloud Platform's BigQuery. Applications originally written or configured for Teradata can run directly on BigQuery without changing SQL code or APIs. To do so, Hyper-Q is situated between your database applications and BigQuery. Sitting in the data path, Hyper-Q accepts all communication from the various applications and translates Teradata-specific SQL statements into BigQuery SQL.

Hyper-Q provides full functional and accurate emulation of all commonly used Teradata features. Even complex features such as recursion or Global Temporary Tables, which do not have a direct corresponding implementation on BigQuery, are emulated by Hyper-Q. Additionally, Hyper-Q supports loaders and utilities commonly used in Extract, Transform, and Load (ETL) or Extract, Load, and Transform (ELT) processing.

Hyper-Q Solution Architecture

Hyper-Q, together with BigQuery, provides a complete replacement of legacy Teradata appliances. Figure 1 depicts the architecture of the Hyper-Q for Google Cloud BigQuery solution.

Hyper-Q connects client applications with BigQuery. In a production environment, two redundant Hyper-Q instances in conjunction with a load balancer provide high-availability and fault-tolerance.

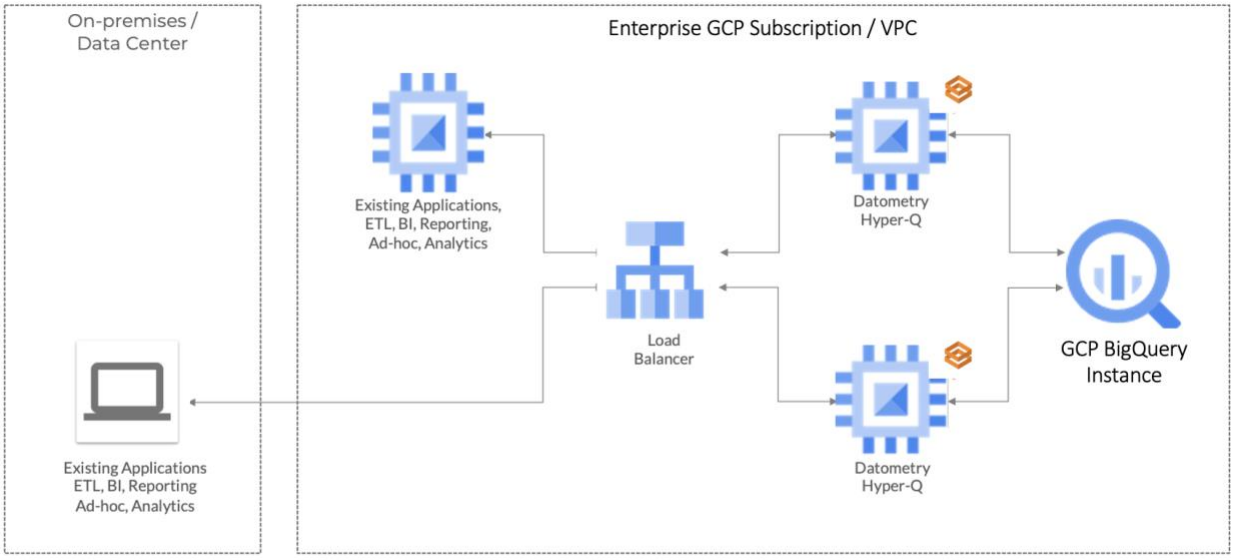

Figure 1. Hyper-Q for BigQuery architecture with multiple Hyper-Q instances

Terminology

We use the following terminology in this guide:

- *Service* refers to the Google Cloud BigQuery service in your project.
- *Google Cloud BigQuery project* or *project* refers to the actual data warehouse.
- *Hyper-Q VM* is the virtual machine in which you install the Hyper-Q software.

To learn more about Google Cloud and BigQuery, see the BigQuery documentation.

3 Prerequisites

Verify that the Google Cloud environment into which you want to deploy Hyper-Q meets the following requirements.

Datometry Hyper-Q Software and License

Datometry makes available the following items needed for the installation. These files need to be copied to the Hyper-Q VM for installation.

Filename	Description
`hyperq-`*`version`*`-bigquery.tar.gz`	RPM packages with which to install the Hyper-Q software.
`dtm.lic`	License file specific to your Google Cloud subscription plan.

Google Cloud BigQuery Project

Ensure you have an active Google Cloud project, and have available the project name, project ID, and project number. To learn more, see Projects in the Google Cloud documentation.

Google Cloud Credentials and Privileges

You must have the following privileges within the Google Cloud project and credentials on the BigQuery instance to which you are connecting Hyper-Q. You need these credentials to install, not to operate, the system. The installation automatically creates an account with which to operate Hyper-Q.

Note: Avoid using highly privileged user IDs such as an overall Google Cloud administrative account. Password requirements are as recommended by the customer.

Google Cloud Permissions within the Project

The person performing the installation must have an account in the Google Cloud project with permissions to provision virtual machines using Compute Engine within the project. To learn more, see the Google Cloud Compute Engine documentation.

BigQuery Permissions

You must create several database objects in the BigQuery project. Therefore, you must have sufficient permissions to perform the following operations in the BigQuery project that you are connecting to Hyper-Q:

- create a schema
- create a user
- create a role
- add user to the newly created role

Create the Metadata Store Schema

Metadata Store contains annotations, a type of metadata for name resolution that identifies complex SQL constructs that Hyper-Q emulates. Hyper-Q stores this metadata in a separate schema named __DTM_MDSTORE. The __DTM_MDSTORE schema contains information such as if a column is case sensitive or not, and the date format to use with a given column among many other characteristics of the SQL being emulated. Hyper-Q references this metadata when emulating queries from Teradata clients for the use with Google BigQuery.

Using a query editor, create the following objects in the BigQuery project. You issue this SQL statement to BigQuery to BigQuery from a SQL client tool that routes the statement through the Google Cloud load balancer through a Datometry Hyper-Q VM.

Note: Be aware that schema and table names in BigQuery are quoted identifiers and must be enclosed by backtick (`) characters, not single quotes.

IMPORTANT: You must grant read access to every user of `MDSTORE_TBL`.

```
CREATE SCHEMA `__DTM_MDSTORE`;

CREATE TABLE `__DTM_MDSTORE`.`MDSTORE_TBL` (

    schemaname string,

    objectname string,

    colname string,

    propname string,

    propvalue string,

    seqno int64

);
```

Create a Dedicated User to Access Metadata Store

Metadata Store is maintained and accessed using a specific user ID. The user ID to be used for this purpose is set using the `"mdstore".user` property in the `dtm.ini` configuration file. To learn more, see "Example dtm.ini Configuration File" on page 21.

Note: Metadata Store does not require a separate user account to be created as shown below. However, it is best practice to create a dedicated user whose authorization is limited only to Metadata Store. If you decide not to create a dedicated user, skip the remainder of this section.

In order to set up a dedicated user ID for accessing Metadata Store you must create a login with a valid password. Hyper-Q uses Google Identity and Access Management (IAM) to authenticate client applications connecting to BigQuery.

The steps to create a Hyper-Q user account for use with BigQuery are:

1. Create an OAuth 2.0 client ID.
2. Authorize the application to access BigQuery.
3. Generate a refresh token.
4. Create client credentials.
5. Ensure the user you create has the BigQuery IAM role `roles/bigquery.user`. To learn more, see BigQuery permissions and predefined IAM roles in the Google documentation.

Create an OAuth 2.0 Client ID

To use OAuth 2.0 in your application, you need create an OAuth 2.0 client ID, which your application uses when requesting an OAuth 2.0 access token.

1. Open a web browser and go to the Google Cloud Platform Console.
2. From the projects list, select a project or create a new one.
3. If the APIs & Services page is not already open, open the console left-side menu and select **APIs & Services**.
4. Click **Credentials** from the APIs & Services page.
5. Click **+ Create Credentials**, and select **OAuth client ID**.
6. If this is your first time creating a client ID, you can also configure your consent screen by clicking **Consent Screen**.
7. Click **Create client ID**.
8. Save the OAuth2 client ID and client secret displayed for the application.

You use the OAuth2 client ID is subsequent steps in the installation procedure, so make note of it for future reference.

Authorize the Application to Access BigQuery

1. Open a web browser and go to the following URL, replacing *`oauth2_client_id`* with the OAuth 2.0 client ID you created in the previous procedure.

   ```
   https://accounts.google.com/o/oauth2/v2/auth?client_id= oauth2_client_id&redirect_uri=
   urn:ietf:wg:oauth:2.0:oob&state=GBQAUthTest&access_type=offline&scope=
   https://www.googleapis.com/auth/bigquery&response_type=code
   ```

2. Log in using a Google account that has privileges to enable the application to view and manage data in BigQuery.
3. Allow these permissions.
4. Copy the authorization code displayed by Google to your clipboard and save it somewhere safe.

   Note: The authorization code is a single use code. If you encounter an error when generating the refresh token, you must generate a new authorization code.

Generate a Refresh Token

OAuth2 access expires after a limited time. An OAuth2 refresh token allows your application to automatically obtain new access tokens.

IMPORTANT: If you receive an "`invalid_grant`" error when generating the refresh token, you must generate a new authorization code. The authorization code is a single use code.

1. Using an HTTP client that can submit arbitrary POST requests, generate a refresh token for the application to access BigQuery.
2. Submit a POST request using the `curl` command with the following values obtained in the previous procedures.

   ```
   https://www.googleapis.com/oauth2/v4/token?code=authorization_code&client_id=
   oauth2_client_id&client_secret=oauth2_client_secret&redirect_uri=urn:ietf:wg:oauth:2.0:o
   ob&grant_type=authorization_code
   ```

where:

Parameter	Description
authorization_code	The code that your application uses to obtain an access token. **Note**: The authorization code is a single use code. If you encounter an error when generating the refresh token, you must generate a new authorization code.
oauth2_client_id	The OAuth client ID.
oauth2_client_secret	The OAuth2 client secret. In this context, the client secret is obviously not treated as a secret.

The following example submits the request using `curl`:

```
curl --data 'code=authorization_code&client_id=
oauth2_client_id&client_secret=oauth2_client_secret&redirect_uri=
urn:ietf:wg:oauth:2.0:oob&grant_type=authorization_code'
'https://www.googleapis.com/oauth2/v4/token'
```

3. Save the value of the refresh token you receive in the response.

   Secure the refresh token according to your organization's policy on storing and maintaining authentication credentials. For example, using a password manger.

Create Client Credentials

You create credentials to use in the client applications using:

- A username, which is the OAuth2 Client ID. For example: `123456789012-1234567890abcdef1234567890abcdef.apps.googleusercontent.com`
- A password, which is the OAuth2 client secret in combination with the refresh token separated by four vertical bars. For example: `'client_secret||||refresh_token'`

Create a BigQuery User Role

Ensure the user you create has the BigQuery IAM role `roles/bigquery.user`. To learn more, see BigQuery permissions and predefined IAM roles in the Google documentation.

Required Ports for Hyper-Q

Port 1025 is the default TCP port for all Teradata clients. Configure the Google Cloud firewall to allow access to Hyper-Q on port 1025. For more information, see Google Cloud firewalls.

If you manage network components outside of Google Cloud that are between your Teradata clients and Hyper-Q, you may need to configure those firewalls to allow access on port 1025.

4 Installation and Setup

This procedure describes how to install and configure an individual Hyper-Q instance. For a deployment that includes multiple instances of Hyper-Q VM, repeat the following procedure for each instance you want to create.

Create a Hyper-Q Virtual Machine

Create a virtual machine for Hyper-Q following the procedures for provisioning virtual machines in Goggle Cloud. The virtual machine you create must be an instance template. Name the instance template such that it is easily identifiable as being used for Hyper-Q. For example: **dtm-hyper-q**

The Hyper-Q virtual machine must meet the following system requirements.

Resource	Minimum	Recommended
Series	N1	N1
VM type	n1-standard-16 (16 vCPU with 60 GB memory)	n1-standard-16 (16 vCPU with 60 GB memory)
Disk	Standard SSD with 500GB	Premium SSD, with 1TB
OS	CentOS 7.x	CentOS 7.x

To learn more, see Creating and starting a VM instance in the Google Cloud documentation.

Create a Virtual Machine Instance Group

An *instance group* is a collection of virtual machine instances that you can manage as a single entity. To learn how to create an instance group, see Creating groups of unmanaged instances in the Google Cloud documentation.

Specify the following values for the Hyper-Q instance group.

Parameter	Value
Instance template	The name of the instance template you specified. For example: **dtm-hyper-q**
Number of instances	2
Autoscaling mode	Do not autoscale.

Create a Network Load Balancer

In a production environment, two redundant Hyper-Q instances in conjunction with a load balancer provide high-availability and fault-tolerance. Google Cloud provides several different load balancer options. For Hyper-Q, you must use either an external or internal TCP/UDP load balancer. To learn about the different load balancers, and how to create and configure one for use with Hyper-Q see the following articles in the Google Cloud documentation:

- Choosing a load balancer
- External TCP/UDP Network Load Balancing overview
- Internal TCP/UDP Load Balancing overview

The following procedure summarizes how to configure a Google Cloud load balancer for use with Hyper-Q VM. Refer to the Google Cloud documentation for information specific to the type of load balancer you are using for your deployment.

1. Specify the following values for the backend configuration of the Hyper-Q load balancer.

Parameter	Value
TCP Load Balancing	Specify TCP Load Balancing
Name	The name of the template you specified. For example: **dtm-hyper-q**
Region and Zone	Select your preferred region and zone.
Backends	Select the existing instance group you created in the previous procedure. For example: **dtm-hyper-q**

2. Specify the following values for the frontend IP and port configuration of the Hyper-Q load balancer.

Parameter	Value
Name (Optional)	Enter a name to identify the fronted IP and port number configuration.
Network Service Tier	Select **Premium**.
IP	Select **Create IP address**.
Port	Choose port 1025, the default port for all existing Teradata clients. **IMPORTANT**: Configure the Google Cloud firewall to allow access to Hyper-Q on port 1025. For more information, see Google Cloud firewalls.

	If you manage network components outside of Google Cloud that are between your Teradata clients and Hyper-Q, you may need to configure those firewalls to allow access on port 1025.

3. Perform a health check on the load balancer to confirm clients can connect to the Hyper-Q VMs. To learn how to perform a health check, see Creating health checks in the Google Cloud documentation.

   Specify the following configuration values for the health check.

Parameter	Value
Protocol	TCP
Port	1025
Unhealthy threshold	6
Check interval	15 seconds

4. Review the health check results and confirm the load balancer is performing properly.

Required Ports for Hyper-Q Virtual Machine

Hyper-Q uses designated ports for communication. Configure the Hyper-Q VM so that it permits inbound connections using the following ports.

Port	Description
22	SSH connection for maintenance during operations.
1025	Default port for all existing Teradata clients.
5432	Default port for Datometry Client Tools and Utilities.

Important: Configure all ports that are not required for your deployment to drop connection requests and not be open for any application or system traffic. As an additional security measure, restrictions should be placed on the available ports to only allow traffic from IP addresses known to be acceptable associated with your corporate network.

Install Extra Packages for Enterprise Linux on Cent OS

The Extra Packages for Enterprise Linux (EPEL) repository are an additional package repository that provides access to install packages for commonly used software.

To install the EPEL packages, perform the following steps:

1. Connect to the virtual machine on which you intend to install Hyper-Q using SSH as the root user.
2. Install the EPEL repository with the following `yum install` command.

   ```
   $ sudo yum install epel-release
   ```

3. Confirm that the EPEL packages are installed using the `yum repolist` command.

   ```
   $ sudo yum repolist
   ```

Install the Hyper-Q Software

To install the software, perform the following steps:

1. Perform the following steps in a directory other than `/opt/datometry/*`.
2. Copy the `hyperq-`*`version`*`-bigquery.tar.gz` to the Hyper-Q VM.
3. Un-compress the file.

   ```
   $ tar xvf hyperq-version-bigquery.tar.gz
   ```

   This creates the following RPM files:

   `hyperq-`*`version`*`.x86_64.rpm`

   `hyperq-monitors-`*`version`*`.x86_64.rpm`

   `hyperq-bigquery-`*`version`*`.x86_64.rpm`

   `hyperq-system-`*`version`*`.x86_64.rpm`

4. Install the software packages as superuser, replacing the *`version`* placeholder with the actual version number.

   ```
   $ sudo yum install hyperq-version.x86_64.rpm
     hyperq-monitors-version.x86_64.rpm
     hyperq-bigquery-version.x86_64.rpm
     hyperq-system-version.x86_64.rpm
   ```

Installation Results

The installation process performs the following steps:

- Creates the group `dtm` and a user ID `dtm` at the OS level.
- Installs all binaries and support files under `/opt/datometry`.
- Creates a service `dtm` within the system.
- Adjusts system controls by installing the file `/etc/sysctl.d/dtm.conf` and applies its configuration settings to the Hyper-Q VM. Changes take effect immediately without requiring a reboot.
- Installs Google BigQuery Python 2 modules. A list of modules can be found at: `/opt/datometry/dtm/lib/dtm-1.0/priv/requirements.txt`

Note: The installation creates the service, however, it does not enable automatically starting the service after a reboot. To enable automatically starting the service, see "Configure Hyper-Q to Start Automatically at Boot Time" on page 19.

Configure Hyper-Q for Use with Google BigQuery

The configuration parameters for Hyper-Q are in the file `/opt/datometry/config/dtm.ini`. Modify the following lines to configure Hyper-Q for use with the BigQuery environment.

Note: All string values on the left-hand side of an assignment in `dtm.ini` must be enclosed in double quotes. For example: `"DTM_MDS_USER".`

For an example of a `dtm.ini` configuration file, see "Example dtm.ini Configuration File" on page 21.

1. As super user, log in as the dtm user.

   ```
   $ sudo su -dtm
   ```

2. Open the `dtm.ini` configuration file in a text editor.
3. In the `[endpoints]`section, locate the line below and replace *`bigquery_project_id`* with your BigQuery project ID.

   ```
   "endpoint".database = bigquery_project_id
   ```

4. In the `[gateways] section`, locate the line below and replace *`bigquery_project_id`* with your BigQuery project ID.

   ```
   "gateway".gpc_project_id = bigquery_project_id
   ```

5. In the `[metadata_stores]` section, locate the lines below and replace *`bigquery_project_id`*, *`bigquery_username`*, and *`bigquery_user_password`* with the name of your Goggle cloud BigQuery project ID, and the username and password to be used to access Metadata Store.

```
"mdstore".database = bigquery_project_id
"mdstore".user = bigquery_username
"mdstore".password = bigquery_user_password
```

Install Hyper-Q User Defined Functions

Hyper-Q utilizes user defined functions (UDFs) to implement standard functionality. These are referred to as Standard UDFs. The UDFs must be installed in the Google BigQuery instance and updated with each new Hyper-Q release.

1. Locate the UDF source files in the directory `/opt/datometry/dtm-`*`version`*`/udf`.
2. Execute the SQL contained in all UDF source files directly on the Google BigQuery instance using a database client to connect directly to the instance. For example, the BigQuery editor that is part of Google Cloud Console.

Install the Hyper-Q License File

Datometry provides you with a unique license file, `dtm.lic`, which is subject to your license agreement. Ensure that:

- The license is valid. The expiration date in the file corresponds to the terms of your license agreement.
- The license file is in `/opt/datometry/config,` and is readable by the user `dtm:dtm`.
- There is only one file ending in the `.lic` extension in the directory `/opt/datometry/config`.

Hyper-Q services will not start if any of the above conditions are not met.

5 After You Install Hyper-Q

After you install Hyper-Q, you start and manage the deployment using the `systemctl` utility. To do so, log into each Hyper-Q VM instance, and run the `systemctl` utility to perform the desired action.

Start the Hyper-Q Service

1. Log into the Hyper-Q VM.
2. Run the `systemctl` command to start the Hyper-Q service.

```
$ sudo systemctl start dtm
```

The service is now operational, and applications can connect to the Hyper-Q service and query BigQuery.

Stop the Hyper-Q Service

1. Log into the Hyper-Q VM.
2. Run the `systemctl` command to stop the Hyper-Q service.

```
$ sudo systemctl stop dtm
```

Check the Status of the Hyper-Q Service

1. Log into the Hyper-Q VM.
2. Run the `systemctl` command to perform a status check for the Hyper-Q service.

```
$ systemctl status dtm
```

Validate the Hyper-Q Installation

You can validate that Hyper-Q is correctly installed and fully operational.

1. Log into Hyper-Q using a Teradata client and valid credentials for the BigQuery project Hyper-Q is associated with.
2. Execute the command `dtm show version`.

   ```
   dtm show version
   ```

3. Using the Teradata SQL `SHOW VIEW` *`view_name`* statement to display a view in your BigQuery project.

   Confirm the definition of the view displays correctly.

   ```
   SHOW VIEW view_name
   ```

Configure Hyper-Q to Start Automatically at Boot Time

The installation registers Hyper-Q as a service with the system but does not enable it to start at boot time. You can configure Hyper-Q to automatically start when you boot the Hyper-Q VM.

After rebooting the Hyper-Q VM, run the command:

```
$ sudo systemctl enable dtm
```

6 Troubleshooting the Hyper-Q Installation

The Hyper-Q installation troubleshooting topics provide solutions to problems that you might encounter during the Hyper-Q deployment process.

For support FAQs and troubleshooting information, visit the Datometry Help Center at: `https://support.datometry.com/`

Collecting Log Files for Troubleshooting a Hyper-Q Installation

Hyper-Q logs critical startup messages as well as errors at runtime to the file `error_log-`*`DATE_TIME`*`.csv` located in the directory `/opt/datometry/logs`.

At startup and at midnight a new log file is created with the current date and time as part of the filename.

Hyper-Q Fails to Start

Hyper-Q will fail to start for the following reasons.

Missing License File

- Copy a valid license file to /opt/datometry/config/dtm.lic.

Duplicate License Files

- Ensure only one file ending in .lic is in the directory /opt/datometry/config.

Expired License

1. Request a new license file from Datometry Support.
2. Ensure the new license has not expired by inspecting the file, then install license file.

To learn more, see "Install the Hyper-Q License File" on page 14.

Client Cannot Connect to Hyper-Q

If your client cannot connect to Hyper-Q verify that the endpoint is correctly configured.

- Ensure the database name and port configuration in the `"endpoints"` section of the `dtm.ini` file is correct.

Logon fails with DTM3102 Error

If your attempt to logon fails with the error message DTM3102: Failed to connect to the underlying database: "failed to connect: password malformed" verify that the password is in the form `'client_secret||||refresh_token',.`

Client Connects to Hyper-Q but Cannot Connect to BigQuery

The connection to the BigQuery project is not configured correctly.

- Ensure database name and port configuration in `"gateways"` section of the `dtm.ini` file is correct.

The Metadata Store is Not Accessible

If the Metadata Store is not accessible, the client receives the error "DTM5306: Invalid MDStore table".

Ensure project ID, database, username, and password configuration in the `"metadata_stores"` section of the `dtm.ini` file is correct, and that the credentials being used can access Metadata Store in BigQuery.

Missing Metadata Store Permissions

If permissions have not been assigned to Metadata Store, queries involving the catalog (Information Schema, DBC) will fail.

- Grant `SELECT` permissions on the Metadata Store schema to `PUBLIC`. See "Create the Metadata Store Schema" on page 7.

Disable data encryption

Either the client cannot log in the database or executing queries on the session fails with "Cannot call a method on closed connection" or a similar error message.

- Ensure the client is not requesting data encryption.

7 Example dtm.ini Configuration File

The following is an example of the `dtm.ini` filed used to configure Hyper-Q.

Placeholder Name	Replacement Value
bigquery_project_id	Name of the project ID
bigquery_username	User ID for Metadata Store access
bigquery_user_password	Password for the Metadata Store user

```
version = "3.41.0"

[endpoints]

; tdc-tdc-bq
"endpoint".port = 1025
"endpoint".database = "bigquery_project_id"
"endpoint".type = teradata"endpoint".version = "160000"
"endpoint".is_case_sensitive = false
"endpoint".spill_dir = "/tmp" ;location to store large result sets

[gateways]

"gateway".name = "gateway_bigquery"
"gateway".is_odbc_gateway = true
"gateway".connection_string =
"Driver=/opt/simba/googlebigqueryodbc/lib/64/libgooglebigqueryodbc_sb64.so;OAuthMechanism=0;
KeyFilePath=/my-path/to-key/account.json;Catalog=bigquery_project_id;Email=my-
email;UseNativeQuery=1;SQLDialect=1;"
"gateway".host = ""
"gateway".port = 1234
"gateway".tls = prefer
"gateway".ipv = 4
"gateway".type = bigquery
"gateway".version = "010000"
"gateway".default_schema = "public"
"gateway".temp_table_schema = "dtm_temp"
"gateway".auth_backend = "bigquery_oauth"
"gateway".gcp_project_id = "bigquery_project_id" ;specified by customer
"gateway".parallel_conversion_available_memory_size = 3200MB
"gateway".max_tdf_rows = 1000000

[metadata_stores]
"mdstore".mdstore_workers = 8
"mdstore".name = "MDSTORE"
"mdstore".gateway = "gateway_bigquery"
"mdstore".user = "bigquery_username.apps.googleusercontent.com” ;specified by customer
"mdstore".password = "bigquery_user_password" ;specified by customer
"mdstore".database = "bigquery_project_id" ;specified by customer
"mdstore".mdstore_schema = "__DTM_MDSTORE"
"mdstore".mdstore_table = "MDSTORE_TBL"
"mdstore".mdstore_worktable = "MDSTORE_WRKTBL"

[pools]
"pool".gateway = ["gateway_bigquery"]
"pool".name = "pool_bigquery"
"pool".backlog = 10
"pool".capacity = 1000
"pool".active = 1000
;"pool".tx_sync = shared

[policies]
```

```
"policy".protocol = all
"policy".database = all
"policy".user = all
"policy".ip = "0.0.0.0"
"policy".ip_prefix = 0
"policy".pool = "pool_bigquery"
"policy".auth_method = passthru

"general_setting1".memory_limit = 2GB
;"general_setting1".total_memory = 2GB
```