TRIFACTA

Connection Guide

Version: 7.1.2
Doc Build Date: 11/25/2020

For third-party license information, please select **About Trifacta** from the Help menu.

1. *Connect . 4*
1.1 *Connection Types . . 5*
1.1.1 *Create DB2 Connections 16*
1.1.2 *Create Oracle Connections 18*
1.1.3 *Create PostgreSQL Connections 20*
1.1.4 *Create SQL Server Connections . 22*
1.1.5 *Enable Teradata Connections 24*
1.1.6 *Enable Snowflake Connections 26*
1.1.6.1 *Create Snowflake Connections 28*
1.1.7 *Enable AWS Glue Access . 31*
1.1.8 *Create Azure SQL Database Connections . 36*
1.1.9 *Create SQL DW Connections . 38*
1.1.10 *Create Databricks Tables Connections . 41*
1.1.11 *Create SFTP Connections 44*
1.1.12 *Create Tableau Server Connections . 48*
1.1.13 *Create Salesforce Connections 52*
1.1.14 *Enable Alation Sources . 54*
1.1.15 *Enable Waterline Sources . 57*
1.2 *Configure Connectivity . 59*
1.2.1 *Enable Relational Connections . 61*
1.2.2 *Enable Custom SQL Query . 64*
1.2.3 *Configure JDBC Ingestion . 66*
1.2.4 *Configure Security for Relational Connections . 71*
1.2.5 *Enable SSO for Relational Connections . 74*
1.2.6 *Configure Type Inference . 78*
1.3 *Troubleshooting Relational Connections . 81*

Connect

This section covers how to configure JDBC integration for Trifacta® Wrangler Enterprise and connect your working platform instance to a wide variety of JDBC-based connections.

Many of these connections can be created from the Trifacta application directly. In some cases, additional configuration is required outside of the application.

Not Covered

This guide does not cover connection types that are deeply tied to a specific infrastructure. The following connection types are described in the appropriate Configuration Guide.

Hadoop

- *Create Hive Connections*

AWS

- *Enable S3 Access*
- *Create Redshift Connections*

Connection Types

Contents:

- *Configure*
  - *Disable Creating Connections for Non-Admins*
- *Supported Environments*
  - *Trifacta Wrangler Enterprise*
  - *Amazon AWS*
  - *Microsoft Azure*
- *Default Connections*
  - *Upload*
- *Big Data*
  - *Apache Hadoop HDFS - Cloudera*
  - *Apache Hadoop HDFS - Hortonworks*
  - *Hive*
- *Cloud Platforms*
  - *Amazon S3*
  - *Amazon Redshift*
  - *Snowflake*
  - *AWS Glue*
  - *Microsoft Azure WASB*
  - *Microsoft Azure ADLS Gen1*
  - *Microsoft Azure ADLS Gen2*
  - *Databricks Tables*
- *Relational DBs*
  - *DB2*
  - *Oracle*
  - *PostgreSQL*
  - *SQL Server*
  - *Teradata*
  - *SQL DW*
  - *Azure SQL Database*
- *Applications*
  - *Salesforce*
  - *SFTP*
  - *Tableau Server*
- *Search Integrations*
  - *Alation*
  - *Waterline*
- *Other Connections*
  - *JDBC relational connections*
  - *Cloud connections*

Trifacta® Wrangler Enterprise supports the following types of connections. Use the links below to enable connection to each type and, where applicable, to create new connections to individual instances of the same type.

Notes:

- HDFS and Hive connections can be configured as part of platform configuration.
- Database connections should be configured after you have completed the platform configuration and have validated that it is working for locally uploaded files.

NOTE: Before creating connections to Hive or relational datastores, you must create and deploy an encryption key file. See *Create Encryption Key File*.

Configure

Disable Creating Connections for Non-Admins

By default, all users are permitted to create connections. As needed, you can disable the ability to create connections for non-admin users.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Search for the following parameter, and set it to `false`:

```
"webapp.connectivity.nonAdminManagementEnabled": true,
```

3. Save your changes and restart the platform.

Supported Environments

Trifacta Wrangler Enterprise

Running environment(s): Trifacta Photon and Spark

Base storage layer: HDFS

Amazon AWS

Running environment(s): Trifacta Photon and Spark

Base storage layer: S3

Microsoft Azure

Running environment(s): Trifacta Photon and Spark

Base storage layer: ADLS Gen1, ADLS Gen2, or WASB

For more information, see *Running Environment Options*.

For more information, see *Set Base Storage Layer*.

Default Connections

These connections are automatically enabled and configured with the product.

Upload

Enable: Automatically enabled.

Create New Connection: n/a

Big Data

These connections pertain to the various big data platforms that are supported by the Trifacta platform.

Apache Hadoop HDFS - Cloudera

Enable: *Configure for Hadoop*.

Create New Connection: n/a

Apache Hadoop HDFS - Hortonworks

Enable: *Configure for Hadoop*

Create New Connection: n/a

Hive

Enable: *Configure for Hive*

> **NOTE:** Additional configuration is required.

Create New Connection:

> **NOTE:** A single public Hive connection is supported.

API: *API Reference*

- Type: `hive`
- Vendor: `hive`

For more information, see *Create Hive Connections*.

Cloud Platforms

These connections pertain to cloud platforms with which the Trifacta platform can integrate.

Amazon S3

Supported Versions: n/a

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Not supported	Supported	Not supported

Enable: *Enable S3 Access*

Create New Connection: n/a

> **NOTE:** A single public connection to S3 is supported.

Amazon Redshift

Supported Versions: n/a

Supported Environments:

> **NOTE:** S3 must be set as the base storage layer. See *Set Base Storage Layer*.

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Not supported	Supported	Not supported
Write	Not supported	Supported	Not supported

Enable: *Create Redshift Connections*

Create New Connection:

UI: *Create Redshift Connections*

API: *API Reference*

- Type: `redshift`
- vendor: `redshift`

Snowflake

Supported Versions: n/a

Supported Environments:

> **NOTE:** S3 must be set as the base storage layer. See *Set Base Storage Layer*.

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Not supported	Supported	Not supported
Write	Not supported	Supported	Not supported

Enable: *Create Snowflake Connections*

Create New Connection:

UI: *Create Snowflake Connections*

API: *API Reference*

- Type: `snowflake`
- vendor: `snowflake`

AWS Glue

Supported Versions: n/a

Supported Environments:

> **NOTE:** S3 must be set as the base storage layer, and the platform must be integrated with EMR. See *Set Base Storage Layer*.

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Not supported	Not supported	Not supported

Enable: *Enable AWS Glue Access*

Create New Connection:

UI: *Create Connection Window*

Microsoft Azure WASB

Supported Versions: n/a

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if WASB is base storage layer)

Enable: *Enable WASB Access*

Create New Connection: n/a

> **NOTE:** A single public connection to WASB is supported.

Microsoft Azure ADLS Gen1

Supported Versions: n/a

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if ADLS Gen1 is base storage layer)

Enable: *Enable ADLS Gen1 Access*.

Create New Connection: n/a

> **NOTE:** A single public connection to ADLS Gen1 is supported.

Microsoft Azure ADLS Gen2

Supported Versions: n/a

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if ABFSS is base storage layer)

Enable: *Enable ADLS Gen2 Access*

Create New Connection: n/a

> **NOTE:** A single public connection to ADLS Gen2 is supported.

Databricks Tables

Supported Versions: n/a

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported

Enable: *Create Databricks Tables Connections*

Create New Connection: *Create Databricks Tables Connections*

> **Tip:** It's easier to create a connection of this type through the UI. Typically, only one connection is needed.

API: *API Reference*

- Type: `jdbc`
- Vendor: `databricks`

Relational DBs

These connections pertain to relational database sources.

> **NOTE:** Unless otherwise noted, authentication to a relational connection requires basic authentication (username/password) credentials.

Enable: For more information, see *Enable Relational Connections*.

DB2

- DB2 for Windows and Unix/Linux

Supported Versions: v10.5.5

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Not supported	Not supported	Not supported

Create New Connection:

UI: *Create DB2 Connections*

API:*API Reference*

- Type: `jdbc`
- Vendor: `db2`

Oracle

Supported Versions: 12.1.0.2

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Supported	Supported	Supported

Create New Connection:

UI: *Create Oracle Connections*

API:*API Reference*

- Type: `jdbc`
- Vendor: `oracle`

PostgreSQL

Supported Versions: 9.3.10

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Create New Connection:

UI: *Create Connection Window*

API:*API Reference*

- Type: `jdbc`

- Vendor: `postgres`

SQL Server

Supported Versions: 12.0.4

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Create New Connection:

UI:

- *Create SQL Server Connections*
- *Create Connection Window*

API: *API Reference*

- Type: `jdbc`
- Vendor: `sqlserver`

Teradata

Supported Versions: 14.10+

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Not supported
Write	Supported	Supported	Supported

Enable:

- *Enable Teradata Connections*

Create New Connection:

UI: *Create Connection Window*

API:*API Reference*

- Type: `jdbc`
- Vendor: `teradata`

SQL DW

Supported Versions: n/a

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported **NOTE:** Additional configuration is required.

Enable: *Create SQL DW Connections*

Create New Connection:

UI: *Create Connection Window*

API: *API Reference*

Azure SQL Database

NOTE: This database connection is a specialized version of a SQL Server connection

NOTE: For Azure deployments, some additional configuration properties must be applied. See *Configure for Azure* .

Supported Versions: Azure SQL Database version 12 (other versions are not supported)

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Not supported	Supported
Write	Supported	Not supported	Supported

Enable: *Create Azure SQL Database Connections*

Create New Connection:

UI: *Create Connection Window*

API: Not supported

Applications

Salesforce

Supported Versions: n/a

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Not supported	Not supported	Not supported

Create New Connection:

UI:

- *Create Salesforce Connections*

API:*API Reference*

- Type: `jdbc`
- Vendor: `salesforce`

SFTP

Supported versions: n/a

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Supported	Supported	Supported
Write	Supported	Supported	Supported

Create New Connection:

UI: *Create SFTP Connections*

API:

- Type: `jdbc`
- Vendor: `sftp`

Tableau Server

Supported Versions: 10.5.x and later

Supported Environments:

Operation	Trifacta Wrangler Enterprise	Amazon	Microsoft Azure
Read	Not supported	Not supported	Not supported
Write	Supported	Supported	Supported

Create New Connection:

UI: *Create Tableau Server Connections*

API: *API Reference*

- Type: `jdbc`
- Vendor: `tableau`

Search Integrations

These types of integrations enable you to search catalogs for dataset to import.

> **NOTE:** Search connections cannot be created or modified through the user interface or APIs. They are enabled and configured through parameter.

Search capabilities are available through the application. See *Import Data Page*.

Alation

Enable: For more information on enabling, see *Enable Alation Sources*.

Create New Connection: No additional configuration is required.

For more information on searching Alation sources, see *Using Alation*.

Waterline

Enable: For more information on enabling, see *Enable Waterline Sources*

Create New Connection: No additional configuration is required.

For more information on searching Waterline sources, see *Using Waterline*.

Other Connections

The following connections can be created based on the available drivers.

> **NOTE:** If you cannot create a connection of one of the following types, please contact *Trifacta Customer Success Services*.

JDBC relational connections

- Databricks Spark SQL
- MongoDB
- DataStax Enterprise
- IBM DB2 for z/OS
- IBM DB2 for i/OS
- IBM Informix
- MySQL
- Pivotal Greenplum
- Progress OpenEdge
- SAP Sybase

Cloud connections

- FinancialForce
- Force.com Applications

Create DB2 Connections

Contents:

You can create connections to one or more DB2 databases from

Pre-requisites

Configure

To create this connection:

For additional details on creating an Oracle connection, see

This connection can also be created using the API.

Modify the following properties as needed:

Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Default Column Data Type Inference	Set to `disabled` to prevent the product from applying its own type inference to each column on import. The default value is `enabled`.
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Use

For more information, see *Database Browser*.

Data Conversion

For more information on how values are converted during input and output with this database, see *DB2 Data Type Conversions*.

Create Oracle Connections

Contents:

- *Pre-requisites*
- *Configure*
- *Use*
- *Data Conversion*

You can create connections to one or more Oracle databases from Trifacta® Wrangler Enterprise.

Pre-requisites

> **NOTE:** Dots (.) in the names of Oracle tables or table columns are not supported.

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.
- If you are connecting to the Oracle database using SSL, additional configuration is required. See *Configure Data Service*.

Configure

To create this connection:

- In the Import Data page, click the Plus sign. Then, select the Relational tab. Click the Oracle card.
- You can also create connections through the Connections page. See *Connections Page*.

For additional details on creating an Oracle connection, see *Enable Relational Connections*.

This connection can also be created using the API.

- For details on values to use when creating via API, see *Connection Types*.
- See *API Reference*.

Modify the following properties as needed:

Property	Description
Host	Enter your hostname. Example: `testsql.database.windows.net`
Port	Set this value to `1521`.
Connect String options	Please insert any connection options as a string here.

Enable SSL	Select the option if the connection should use SSL. **NOTE:** Additional configuration is required. See *Configure Data Service*.
Service Name	Enter the name of the Oracle service.
User Name	(basic credential type only) Username to use to connect to the database.
Password	(basic credential type only) Password associated with the above username.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Use

For more information, see *Database Browser*.

Data Conversion

For more information on how values are converted during input and output with this database, see *Oracle Data Type Conversions*.

Create PostgreSQL Connections

Contents:

- *Pre-requisites*
- *Configure*
- *Use*
- *Data Conversion*

You can create connections to one or more PostgreSQL databases from Trifacta® Wrangler Enterprise. For more information on PostgreSQL, see *https://www.postgresql.org/*.

Pre-requisites

> **If the Trifacta databases are hosted on a PostgreSQL server, do not create a connection to this database.**

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Configure

To create this connection:

- In the Import Data page, click the Plus sign. Then, select the Relational tab. Click the PostgreSQL card.
- You can also create connections through the Connections page.
- See *Connections Page*.

For additional details on creating a PostgreSQL connection, see *Enable Relational Connections*.

This connection can also be created using the API.

- For details on values to use when creating via API, see *Connection Types*.
- See *API Reference*.

Modify the following properties as needed:

Property	Description
Host	Enter your fully qualified hostname. Example: `my.postgres.server`
Port	Set this value to `5432`.
Connect String Options	Insert any additional connection parameters, if needed.
Enable SSL	Select the checkbox to enable SSL connections to the database.
Database	Enter the name of the database on the server to which to connect.
User Name	Username to use to connect to the database.

Use

For more information, see

Data Conversion

For more information on how values are converted during input and output with this database, see *Postgres Data Type Conversions*

Create SQL Server Connections

Contents:

- *Pre-requisites*
- *Configure*
- *Use*
- *Data Conversion*

You can create connections to one or more Microsoft SQL Server databases from Trifacta® Wrangler Enterprise.

Pre-requisites

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.
- If you plan to create an SSO connection of this type, additional configuration may be required. See *Enable SSO for Relational Connections*.

Configure

To create this connection:

- In the Import Data page, click the Plus sign. Then, select the Relational tab. Click the SQL Server card.
- You can also create connections through the Connections page.
- See *Connections Page*.

For additional details on creating a SQL Server connection, see *Enable Relational Connections*.

This connection can also be created using the API.

- For details on values to use when creating via API, see *Connection Types*.
- See *API Reference*.

Modify the following properties as needed:

Property	Description
Host	Enter your hostname. Example: `testsql.database.windows.net`
Port	Set this value to `1433`.
Connect String options	Please insert the following as a single string (no line breaks): `;database=<DATABASE_NAME>;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.windows.net;loginTimeout=30;` where: - `<DATABASE_NAME>` is the name of the database to which you are connecting

User Name	(basic credential type only) Username to use to connect to the database.
Password	(basic credential type only) Password associated with the above username.
Default Column Data Type Inference	Set to `disabled` to prevent the platform from applying its own type inference to each column on import. The default value is `enabled`.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Use

For more information, see *Database Browser*.

Data Conversion

For more information on how values are converted during input and output with this database, see *SQL Server Data Type Conversions*.

Enable Teradata Connections

Contents:

- *Limitations*
- *Download and Install Teradata drivers*
- *Increase Read Timeout*
- *Create Teradata Connection*
- *Testing*

This section provides information on how to enable connection to Teradata databases.

- Teradata provides Datawarehousing & Analytics solutions and Marketing applications. The Teradata database supports all of their Datawarehousing solutions. For more information, see *http://www.teradata.com.*
- For more information on supported versions, see *Connection Types*.

This connection supports reading and writing. You can create multiple Teradata connections in the Trifacta application.

Limitations

- By default, Teradata does not permit the publication of datasets containing duplicate rows. Workarounds:
  - Your final statement for any recipe that generates results for Teradata should include a `Remove duplicate rows` transformation.

    > **NOTE:** The above transformation removes exact, case-sensitive duplicate rows. Teradata may still prevent publication for case-insensitive duplicates.

  - It's possible to change the default writing method to Teradata to enable duplicate rows. For more information, contact *Trifacta Support*.
- When creating custom datasets using SQL from Teradata sources, the `ORDER BY` clause in standard SQL does not work. This is a known issue.

Download and Install Teradata drivers

To enable connectivity, you must download and install the Teradata drivers into an accessible location on the Trifacta® node.

> **NOTE:** Please download and install the Teradata driver that corresponds to your version of Teradata. For more information on supported versions, see *Connection Types*.

Steps:

1. If you don't have a Teradata developer account, create one here: *https://downloads.teradata.com/user/register*
2. Log in to the account. Navigate to *http://downloads.teradata.com/download/connectivity/jdbc-driver*
3. Download the JDBC driver in ZIP or TAR form.
4. Copy the downloaded ZIP or TAR file to the Trifacta node.
5. Extract and place the JAR file into a folder accessible to the Trifacta user.
6. Verify that the Trifacta user is the owner of the JAR file and its parent folder.

7. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
8. Locate the `data-service.classpath`. To the classpath value add the folder where you installed the JAR file. For the new entry, remember to add the following to the entry:
    a. Add a prefix of `:`.
    b. Add a suffix of `/*`.
    c. Example:

    ```
    :/opt/trifacta/drivers/*
    ```

    d. Whole classpath example:

    ```
    "data-service": { ...
    "classpath": "%(topOfTree)s/services/data-service/build/libs/data-service.jar:%(topOfTree)s
    /services/data-service/build/conf:%(topOfTree)s/services/data-service/build/dependencies/*:/opt
    /trifacta/drivers/*"
    ```

9. Save your changes and restart the platform.

Increase Read Timeout

Particularly when reading from large Teradata tables, you might experience read timeouts in the Trifacta application.

The default setting is `300` seconds (5 minutes). You should consider raising this limit if you are working with large tables.

For more information, see *Configure Photon Running Environment*.

Create Teradata Connection

For more information on creating a Teradata connection, see *Create Connection Window*.

Testing

Steps:

1. After you create your connection, load a small dataset based on a table in the connected Teradata database. See *Import Data Page*.
2. Perform a few simple transformations to the data. Run the job. See *Transformer Page*.
3. Verify the results.

For more information, see *Verify Operations*.

Enable Snowflake Connections

Contents:

- *Limitations*
- *Pre-requisites*
- *Enable*
- *Configure*
- *Create Stage*
- *Create Snowflake Connection*
- *Testing*

This section describes how to integrate the Trifacta® platform with Snowflake databases.

- Snowflake provides a cloud-database datawarehouse designed for big data processing and analytics. For more information, see *https://www.snowflake.com*.
- For more information on supported versions, see *Connection Types*.

Limitations

> **NOTE:** This integration is supported only for deployments of Trifacta Wrangler Enterprise in customer-managed AWS infrastructures. These deployments must use S3 as the base storage layer. For more information, see *Supported Deployment Scenarios for AWS*.

- SSO connections are not supported.

Pre-requisites

- If you do not provide a stage database, then the Trifacta platform must create one for you in the default database. In this default database, you must include a schema named `PUBLIC`. For more information, please see the Snowflake documentation.

Enable

When relational connections are enabled, this connection type is automatically available. For more information, see *Enable Relational Connections*.

Configure

To create a Snowflake connection, you must enable the following feature. The job manifest feature enables the creation of a manifest file to track the set of temporary files written to S3 before publication to Snowflake.

> **NOTE:** This feature must be enabled when the base storage layer is set to S3. Please verify the following.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

2. Locate the following parameter and set it to `true`:

```
"feature.enableJobOutputManifest": true,
```

3. Save your changes and restart the platform.

Create Stage

In Snowflake terminology, a **stage** is a database object that points to an external location on S3. It must be an external stage containing access credentials.

- If a stage is used, it is typically the default bucket used on S3 for storage.

  > **NOTE:** For read-only connections to Snowflake, you must specify a Database for Stage. The connecting user must have write access to this database.

  > **Tip:** You can specify a separate database to use for your stage.

- If a stage is not specified, a temporary stage is created using the current user's AWS credentials.

  > **NOTE:** Without a defined stage, you must have write permissions to the database from which you import. This database is used to create the temporary stage.

For more information on stages, see *https://docs.snowflake.net/manuals/sql-reference/sql/create-stage.html*.

In the Trifacta platform, the stage location is specified as part of creating the Snowflake connection.

Create Snowflake Connection

For more information, see *Create Snowflake Connections*.

Testing

Steps:

1. After you create your connection, load a small dataset based on a table in the connected Snowflake database.

   > **NOTE:** For Snowflake connections, you must have write access to the database from which you are importing.

   See *Import Data Page*.
2. Perform a few simple transformations to the data. Run the job. See *Transformer Page*.
3. Verify the results.

For more information, see *Verify Operations*.

Create Snowflake Connections

This section describes how to create a connection to your Snowflake datawarehouse.

- Snowflake is an S3-based data warehouse service hosted in the cloud. Auto-scaling, automatic failover, and other features simplify the deployment and management of your enterprise's data warehouse. For more information, see *https://www.snowflake.com*.

Pre-requisites

- **S3 base storage layer:** Snowflake access requires installation of Trifacta software in the AWS infrastructure and use of S3 as the base storage layer, which must be enabled. See *Set Base Storage Layer*.
- **Integration:** Your Trifacta instance is connected to an EMR cluster.

  > **NOTE:** EMRFS Consistency View must be enabled.

  See *Configure for EMR*.
- **Deployment:** Trifacta platform is deployed in EC2.
- **PUBLIC schema:** If you do not create an external staging database:
  - A `PUBLIC` schema is required in your default database.
  - If you do not provide a stage database, then a temporary stage is created for you under the `PUBLIC` schema in the default database.
- **S3 bucket:** The user-created stage must point to the same S3 bucket as the default bucket in use by Trifacta Wrangler Enterprise.
- **Same region:** The Snowflake cluster must be in the same region as the default S3 bucket.

Limitations

- You cannot perform ad-hoc publication to Snowflake.
- SSO connections are not supported.

Create Connection

You can create Snowflake connections through the following methods.

Create through application

Any user can create a Snowflake connection through the application.

Steps:

1. Login to the application.
2. In the menu, click **User menu > Preferences > Connections**.
3. In the Create Connection page, click the Snowflake connection card.
4. Specify the properties for your Snowflake database connection. The following parameters are specific to Snowflake connections:

> **NOTE:** In Snowflake connections, property values are case-sensitive. Snowflake-related locations are typically specified in capital letters.

Property	Description

Account Name	Snowflake account to use. Suppose your hostname is the following:
	`mycompany.snowflakecomputing.com`
	Your account name is the following:
	`mycompany`

Warehouse	The name of the warehouse to which to connect
Stage	If you have deployed a Snowflake stage for managing file conversion to tables, you can enter its name here. A **stage** is a database object that points to an external location on S3. It must be an external stage containing access credentials. If a stage is used, then this value is typically the schema and the name of the stage. Example value: `MY_SCHEMA.MY_STAGE` If a stage is not specified, a temporary stage is created using the current user's AWS credentials. **NOTE:** Without a defined stage, you must have write permissions to the database from which you import. This database is used to create the temporary stage. For more information on stages, see *https://docs.snowflake.net/manuals/sql-reference/sql/create-stage.html*.
Database for Stage	(optional) If you are using a Snowflake stage, you can specify a database other than the default one to host the stage. **NOTE:** If you are creating a read-only connection to Snowflake, this field is required. The accessing user must have write permission to the specified database. If no value is specified, then your stage must be in the default database.

For more information, see *Create Connection Window*.

Disable SSL connections

By default, connections to Snowflake use SSL. To disable, please add the following string to your Connect String Options:

```
;ssl=false
```

Save your changes.

Connect through proxy

If you require connection to Snowflake through a proxy server, additional Connect String Options are required. For more information, see *https://docs.snowflake.net/manuals/user-guide/jdbc-configure.html#specifying-a-proxy-server-in-the-jdbc-connection-string*

.

Create via API

For more information, see *https://api.trifacta.com/ee/es.t/index.html#operation/createConnection*

Testing

Import a dataset from Snowflake. Add it to a flow, and specify a publishing action back to Snowflake. Run a job.

For more information, see *Verify Operations*.

Enable AWS Glue Access

Contents:

- *Supported Deployment*
  - *EMR Settings*
  - *Authentication*
- *Limitations*
- *Enable*
- *Configure*
- *Create Connection*
- *Use*

If you have integrated with an EMR cluster version 5.8.0 or later, you can configure your Hive instance to use AWS Glue Data Catalog for storage and access to Hive metadata.

> **Tip:** For metastores that are used across a set of services, accounts, and applications, AWS Glue is the recommended method of access.

For more information on AWS Glue, see *https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-hive-metastore-glue.html*.

This section describes how to enable integration with your AWS Glue deployment.

Supported Deployment

AWS Glue tables can be read under the following conditions:

- The Trifacta platform uses S3 as the base storage layer.
- The Trifacta platform is integrated with an EMR cluster:
  - EMR version 5.8.0 or later
  - EMR cluster has been configured with HiveServer2
- The Hive deployment must be integrated with AWS Glue.

  > **NOTE:** Hive connections are supported when S3 is the backend datastore.

- For HiveServer2 connectivity, the Trifacta node has direct access to the Master node of the EMR cluster.

EMR Settings

When you create the EMR cluster, please verify the following in the AWS Glue Data Catalog settings:

- **`Use for Hive table metadata`**
- **`Use for Spark table metadata`**

Required Glue table properties

Each Glue table must be created with the following properties specified:

- `InputFormat`
- `OutputFormat`
- `Serde`

These properties must be specified for the Hive JDBC driver to read the Glue tables.

For additional limitations on access Hive tables through Glue, see *https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-hive-metastore-glue.html#emr-hive-glue-considerations-hive* .

Deploy Credentials JAR to S3

To enable integration between the Trifacta platform and AWS Glue, a JAR file for managing the Trifacta credentials for AWS access must be deployed to S3 in a location that is accessible to the EMR cluster.

When the EMR cluster is launched with the followng custom bootstrap action, the cluster does one of the following:

- Interacts with AWS Glue using the credentials specified in `trifacta-conf.json`
- If `aws.mode` = `user`, then the credentials registered by the user are used to connect to AWS Glue.

Steps:

1. From the installation of the Trifacta platform, retrieve the following file:

   ```
   [TRIFACTA_INSTALL_DIR]/aws/glue-credential-provider/build/libs/trifacta-aws-glue-credential-provider.jar
   ```

2. Upload this JAR file to an S3 bucket location where the EMR cluster can access it:
   a. **Via AWS Console S3 UI:** See *http://docs.aws.amazon.com/cli/latest/reference/s3/index.html*.
   b. **Via AWS command line:**

      ```
      aws s3 cp trifacta-aws-glue-credential-provider.jar s3://<YOUR-BUCKET>/
      ```

3. Create a bootstrap action script named `configure_glue_lib.sh`. The contents must be the following:

   ```
   sudo aws s3 cp s3://<YOUR-BUCKET>/trifacta-aws-glue-credential-provider.jar  /usr/share/aws/emr/emrfs/auxlib/
   sudo aws s3 cp s3://<YOUR-BUCKET>/trifacta-aws-glue-credential-provider.jar  /usr/lib/hive/auxlib/
   ```

4. This script must be uploaded into S3 in a location that can be accessed from the EMR cluster. Retain the full path to this location.
5. Add a bootstrap action to EMR cluster configuration.
   a. **Via AWS Console S3 UI:** Create the bootstrap action to point to the script that you uploaded on S3.
   b. **Via AWS command line:**
      i. Upload the `configure_glue_lib.sh` file to the accessible S3 bucket.
      ii. In the command line cluster creation script, add a custom bootstrap action. Example:

      ```
      --bootstrap-actions '[
      {"Path":"s3://<YOUR-BUCKET>/configure_glue_lib.sh","Name":"Custom action"}
      ]'
      ```

Authentication

Authentication methods are required permissions are based on the AWS authentication mode:

```
"aws.mode": "system",
```

aws.mode value	Permissions	Doc
`system`	IAM role assigned to the cluster must provide access to AWS Glue.	See *Configure for AWS*.
`user`	The user role must provide access to AWS Glue.	See below for an example IAM role access control. See *Configure AWS Per-User Authentication*.

Example fine-grain access control for IAM policy:

If you are using IAM roles to provide access to AWS Glue, you can review the following fine-grained access control, which includes the permissions required to access AWS Glue tables. Please add this to the Permissions section of your AWS Glue Catalog Settings page.

> **NOTE:** Please verify that access is granted in the IAM policy to the default database for AWS Glue, as noted below.

```
{
    "Sid" : "accessToAllTables",
    "Effect" : "Allow",
    "Principal" : {
      "AWS" : [  "arn:aws:iam::<accountId>:role/glue-read-all" ]
    },
    "Action" : [ "glue:GetDatabases", "glue:GetDatabase", "glue:GetTables", "glue:GetTable", "glue:
GetUserDefinedFunctions", "glue:GetPartitions" ],
    "Resource" : [ "arn:aws:glue:us-west-2:<accountId>:catalog", "arn:aws:glue:us-west-2:<accountId>:database
/default", "arn:aws:glue:us-west-2:<accountId>:database/global_temp", "arn:aws:glue:us-west-2:<accountId>:
database/mydb", "arn:aws:glue:us-west-2:<accountId>:table/mydb/*" ]
}
```

S3 access

AWS Glue crawls available data that is stored on S3. When you import a dataset through AWS Glue:

- Any samples of your data that are generated by the Trifacta platform are stored in S3. Sample data is read by the platform directly from S3.
- Source data is read through AWS Glue.

> **You should review and, if needed, apply additional read restrictions on your IAM policies so that users are limited to reading data from their own S3 directories. If all users have access to the same areas of the same S3 bucket, then it may be possible for users to access datasets through the platform when it is forbidden through AWS Glue.**

Limitations

- Access is read-only. Publishing to Glue hosted on EMR is not supported.
- When using per-user IAM role-based authentication, EMR Spark jobs on AWS Glue datasources may fail if the job is still running beyond the defined session limit after job submission time for the IAM role.
  - In the AWS Console, this limit is defined in hours as the **Maximum CLI/API session duration** assig ned to the IAM role.
  - In the AWS Glue catalog client for the Hive Metadata store, the temporary credentials generated for the IAM role expire after this limit in hours and cannot be renewed.

Enable

Please verify the following have been enabled and configured.

1. Your deployment has been configured to meet the Supported Deployment guidelines above.
2. You must integrate the platform with Hive.

   > **NOTE:** For the Hive hostname and port number, use the Master public DNS values. For more information, see *https://docs.aws.amazon.com/emr/latest/ReleaseGuide/emr-hive-metastore-glue.html*.

   For more information, see *Configure for Hive*.
3. If you are using it, the custom SQL query feature must be enabled. For more information, see *Enable Custom SQL Query*.

Configure

When accessing Glue using temporary per-user credentials, the credentials are given a duration of `1` hour. As needed, you can modify this duration.

> **NOTE:** This value cannot exceed the Maximum Session Duration value for IAM roles, as configured in the IAM Console.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter. By default, this value is set to `1`:

   ```
   "data-service.sqlOptions.glueTempCredentialTimeoutInHours": 1
   ```

3. Save your changes and restart the platform.

Create Connection

You can create one or more connections to databases in your AWS Glue deployment.

Key fields:

Field	Description
EMR Master Node DNS	This DNS value can be retrieved from the EMR console.
Port	The port number through which to connect to the DNS master node
Connection String Options	No values need to be provided here.

- See *Create Connection Window*.
- See *Connections Page*.

Use

After the integration has been made between the platform and AWS Glue, you can import datasets.

- Browse for datasets through AWS Glue. See *AWS Glue Browser*.

- Import using custom SQL queries. For more information, see *Create Dataset with SQL*.

Create Azure SQL Database Connections

Contents:

- *Limitations*
- *Pre-requisites*
- *Configure*
  - *Configure for SSO*
- *Use*
- *Data Conversion*

You can create a connection to a Microsoft Azure SQL database from the Trifacta platform. This section describes how to create connections of this type.

- This connection type supports data ingestion into ADLS/WASB. When large volumes of data are read from an Azure SQL database during job execution, the data is stored in a temporary location in ADLS/WASB. After the job has been executed, the data is removed from the datastore. This process is transparent to the user.
- For more information on Azure SQL database, see *https://azure.microsoft.com/en-us/services/sql-database/*.

Limitations

- None.

Pre-requisites

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Configure

This connection can also be created using the following property substitutions via API.

- For details on values to use when creating via API, see *Connection Types*.
- For additional details on creating an Azure SQL Database connection, see *Enable Relational Connections*.

Please create an Azure SQL Database connection and then specify the following properties with the listed values:

Property	Description
Host	Enter your hostname. Example: `testsql.database.windows.net`
Port	Set this value to `1433`.

Connect String options	Please insert the following as a single string (no line breaks):
	`;encrypt=true;trustServerCertificate=false;`
	`hostNameInCertificate=*.database.windows.net;loginTimeout=30;`
	
	Tip: If you have access to the Azure SQL database through the Azure Portal, please copy the Connect String from that configuration. You may omit the username and password from that version of the string.

Database	(optional) Name of the Azure SQL database to which you are connecting.
User Name	(for `basic` Credential Type) Username to use to connect to the database.
Password	(for `basic` Credential Type) Password associated with the above username.
Credential Type	• `basic` - Specify username and password as part of the connection • `Azure Token SSO` - Use the SSO principal of the user creating the connection to authenticate to the Azure SQL database. Additional configuration is required. See *Enable SSO for Azure Relational Connections*.
Default Column Data Type Inference	Set to `disabled` to prevent the Trifacta platform from applying its own type inference to each column on import. The default value is `enabled` .

Configure for SSO

If you have enabled Azure AD SSO integration for the Trifacta platform, you can create SSO connections to Azure relational databases. See *Enable SSO for Azure Relational Connections*.

Use

For more information, see *Database Browser*.

Data Conversion

For more information on how values are converted during input and output with this database, see *SQL Server Data Type Conversions*.

Create SQL DW Connections

Contents:

- *Limitations*
- *Pre-requisites*
- *Connection Types*
- *Azure SQL DW permissions*
- *Azure SQL DW External Data Source Name*
- *Configure*
  - *Configure for SSO*
- *Use*
- *Data Conversion*

This section describes how to create connections to Microsoft SQL Datawarehouse (DW).

Limitations

- Microsoft SQL DW connections are available only if you have deployed the Trifacta® platform onto Azure.
- SSL connections to SQL DW are required.

> **NOTE:** In this release, this connection cannot be created through the APIs. Please create connections of this type through the application.

Pre-requisites

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Connection Types

The Trifacta platform supports two types of connections to an Azure SQL DW data warehouse:

Connection Type	Description	Notes
SQL DW Read-Only	Read-only access to the SQL DW data warehouse. This connection is available on the Import Data page only. To create, see *Import Data Page*.	This connection requires fewer permissions on the data warehouse and its databases but is less performant.
SQL DW Read-Write	Read-write access to the SQL DW data warehouse. This connection is available for reading, direct publishing, and ad-hoc publishing. **NOTE:** Under Azure SSO, write operations are not supported through SQL DW connections. To create, see *Connections Page*.	This connection requires more permissions. You must also specify an External Datasource Name. See below. **Tip:** Spark-based jobs that read or write through your SQL DW connection leverage PolyBase for faster performance.

Azure SQL DW permissions

- **Read-Only connection:** The authenticating DB user must have read permissions to any SQL DW databases, schemas and tables to which the user should have access.
- **Read-Write connection:** In addition to the above, the authenticating DB user must have the following permissions:

```
CREATE TABLE**
ALTER ANY SCHEMA
ALTER ANY EXTERNAL DATA SOURCE
ALTER ANY EXTERNAL FILE FORMAT
```

    - The authenticating DB user must also have read access to the external data source.

Azure SQL DW External Data Source Name

When specifying a SQL DW Read-Write connection, you can provide an External Data Source Name value as part of the connection definition. The External Data Source enables publishing and support for large-scale data ingestion.

> **NOTE:** This setting is not used for SQL DW Read-Only connections.

If the External Data Source is not provided:

- The connection is read-only.
- The native ingestion of the Trifacta platform is used.

Requirements:

- The external data source must be created by the database admin on the default database defined in the SQL DW connection. For more information:
    - *https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-data-source-transact-sql#d-create-external-data-source-to-reference-azure-blob-storage*
    - *https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-data-source-transact-sql#g-create-external-data-source-to-reference-azure-data-lake-store*
- The External Data Source must point to the same storage location as the base storage layer for the Trifacta platform. For example, if the base storage layer is WASB, the External Datasource must point to the same storage account defined in Trifacta configuration. If this configuration is incorrect, then publishing and ingestion of data fail.
- For more information on privileges required for the authenticating DB user, see *https://docs.microsoft.com/en-us/sql/t-sql/statements/create-external-table-transact-sql*.

Configure

To create this connection:

- **Read-only connection:** See *Import Data Page*.
- **Read-write connection:** See *Connections Page*.
- For additional details on creating a relational connection, see *Enable Relational Connections*.

Please create a connection of this type in the appropriate page and modify the following properties with the listed values:

Property	Description
Host	Enter your hostname. Example: `testsql.database.windows.net`
Port	Set this value to `1433`.
Database	Set this value to the default database name.
External Data Source Name	For read-write connections, you must provide an External Data Source. Otherwise, the connection is read-only. See above for details.
Connect String options	Include any options required for your environment:
User Name	Username to use to connect to the database.
Password	Password associated with the above username.
Credential Type	• `basic` - Specify username and password as part of the connection • `Azure Token SSO` - Use the SSO principal of the user creating the connection to authenticate to the SQL Server database. Additional configuration is required. See *Enable SSO for Azure Relational Connections*.
Default Column Data Type Inference	Set to `disabled` to prevent the Trifacta platform from applying its own type inference to each column on import. The default value is `enabled`.

Configure for SSO

If you have enabled Azure AD SSO integration for the Trifacta platform, you can create SSO connections to Azure relational databases.

NOTE: When Azure AD SSO is enabled, write operations to SQL Datawarehouse are not supported.

See *Enable SSO for Azure Relational Connections*.

Use

For more information on locating data, see *Database Browser*.

For more information, see *Using SQL DW*.

Data Conversion

For more information on how values are converted during input and output with this database, see *SQL DW Data Type Conversions*.

Create Databricks Tables Connections

Contents:

- *Limitations*
- *Pre-requisites*
  - *Insert Databricks Access Token*
- *Enable*
- *Create Connection*
- *Use*
- *Data Conversion*
- *Troubleshooting*
  - *Failure when importing wide Databricks Tables table*

You can create a connection to Azure Databricks tables from the Trifacta platform. This section describes how to create connections of this type.

- Databricks Tables provides a JDBC-based interface for reading and writing datasets in ADLS or WASB. Using the underlying JDBC connection, you can access your ADLS or WASB data like a relational datastore, run jobs against it, and write results back to the datastore as JDBC tables.
- Your connection to Databricks Tables leverages the SSO authentication that is native to Azure Databricks. For more information on Databricks Tables, see *https://docs.microsoft.com/en-us/azure/databricks/data/tables*.

Limitations

- Ad-hoc publishing of generated results to Databricks Tables is not supported.
- Creation of datasets with custom SQL is not supported.
- Integration with Kerberos or secure impersonation is not supported.
- Some table types and publishing actions are not supported. For more information, see *Using Databricks Tables*.

Pre-requisites

- The Trifacta platform must be installed on Azure and integrated with an Azure Databricks cluster.
  - See *Install for Azure*.
  - See *Configure for Azure Databricks*.

> **NOTE:** For job execution on Spark, the connection must use the Spark instance on the Azure Databricks cluster. No other Spark instance is supported. You can run jobs from this connection through the Photon running environment. For more information, see *Running Environment Options*.

- This connection interacts with Databricks Tables through the Hive metastore that has been installed in Azure Databricks.

> **NOTE:** External Hive metastores are not supported.

Insert Databricks Access Token

Each user must insert a Databricks Personal Access Token into the user profile. For more information, see *Databricks Personal Access Token Page*.

Enable

To enable Databricks Tables connections, please complete the following:

> **NOTE:** Typically, you need only one connection to Databricks Tables, although you can create multiple connections.

> **NOTE:** This connection is created with SSL automatically enabled.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter and set it to `true`:

```
"feature.databricks.connection.enabled": true,
```

3. Save your changes and restart the platform.

Create Connection

This connection can also be created via API. For details on values to use when creating via API, see *Connection Types*.

Please create an Databricks connection and then specify the following properties with the listed values:

> **NOTE:** Host and port number connection information is taken from Azure Databricks and does not need to be re-entered here. See *Configure for Azure Databricks*.

Property	Description
Connect String options	Please insert any connection string options that you need. Connect String options are not required for this connection.
Test Connection	Click this button to test the specified connection.
Default Column Data Type Inference	Set to `disabled` to prevent the Trifacta platform from applying its own type inference to each column on import. The default value is `enabled`.

Use

For more information, see *Using Databricks Tables*.

Data Conversion

For more information on how values are converted during input and output with this database, see *Databricks Tables Data Type Conversions*.

Troubleshooting

Failure when importing wide Databricks Tables table

If you are attempting to import a table containing a large number of columns (>200), you may encounter an error message similar to the following:

```
2019-11-26T10:17:11.439Z [XNIO-3 task-15] ERROR com.trifacta.dataservice.DataServiceController -
[sid=de7f0e78-087d-4922-9251-337c0cc6da71] - [rid=4851532f-3fb1-4746-9884-a1f71c8209ee] - [method=POST] -
[url=/jdbc/datastream] - Stack trace: com.trifacta.dataservice.connect.exception.UnknownJdbcException: A
system error occurred: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in stage
408.0 failed 4 times, most recent failure: Lost task 0.3 in stage 408.0 (TID 1342, 10.139.64.11, executor
11): org.apache.spark.SparkException: Kryo serialization failed: Buffer overflow. Available: 0, required:
1426050. To avoid this, increase spark.kryoserializer.buffer.max value.
```

The problem is that the serializer ran out of memory.

Solution:

To address this issue, you can increase the Kyroserializer buffer size.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the `spark.props` section and add the following setting. Modify `2000` (2GB) depending on whether your import is successful:

   ```
   "spark.kryoserializer.buffer.max.mb": "2000"
   ```

3. Save your changes and restart the platform.
4. Attempt to import the dataset again. If it fails, you can try incrementally raising the above value.

For more information on passing property values into Spark, see *Configure for Spark*.

Create SFTP Connections

Contents:

- *Limitations*
- *Pre-requisites*
  - *SSH Keys*
- *Enable*
  - *Configure file storage protocols and locations*
  - *Enforce authentication methods*
  - *Java VFS service*
- *Create Connection*
  - *Create through application*
  - *Create through APIs*

You can create connections to SFTP servers to upload your datasets to the Trifacta® platform.

- Linux- and Windows-based SFTP servers are supported.

Jobs can be executed from SFTP sources on the following running environments:

- Trifacta Photon
- HDFS-based Spark, which includes Cloudera and Hortonworks

- Spark on EMR
- Azure Databricks

Limitations

- Ingest of over 500 files through SFTP at one time is not supported.
- You cannot run jobs using Avro or Parquet sources uploaded via SFTP.
- When you specify a parameterized output as part of your job execution, the specified output location may include additional unnecessary information about the SFTP connection identifier. None of this information is sensitive. This is a known issue.
- You cannot publish TDE or Hyper format to SFTP destinations.
- You cannot publish compressed Snappy files to SFTP destinations.

Pre-requisites

- Acquire user credentials to access the SFTP server. You can use username/password credentials or SSH keys. See below.
- Verify that the credentials can access the proper locations on the server where your data is stored. Initial directory of the user account must be accessible.

SSH Keys

If preferred, you can use SSH keys to for authentication to the SFTP server.

> **NOTE:** SSH keys must be private RSA keys. If you have OpenSSH keys, you can use the ssh-keygen utility to convert them to private RSA keys.

Enable

By default, this connection type is automatically enabled for use.

> **NOTE:** You must provide the protocol identifier and storage locations for the SMTP server. See below.

Configure file storage protocols and locations

The Trifacta platform must be provided the list of protocols and locations for accessing SFTP.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameters and set their values according to the table below:

```
"fileStorage.whitelist": ["sftp"],
"fileStorage.defaultBaseUris": ["sftp:///"],
```

Parameter	Description
filestorage.whitelist	A comma-separated list of protocols that are permitted to access SFTP. **NOTE:** The protocol identifier `"sftp"` must be included in this list.
filestorage.defaultBaseUris	For each supported protocol, this param must contain a top-level path to the location where Trifacta platform files can be stored. These files include uploads, samples, and temporary storage used during job execution. **NOTE:** A separate base URI is required for each supported protocol. You may only have one base URI for each protocol. **NOTE:** For SFTP, three slashes at the end are required, as the third one is the end of the path value. This value is used as the base URI for all SFTP connections created in Trifacta Wrangler Enterprise. Example: `sftp:////` The above example is the most common example, as it is used as the base URI for all SFTP connections that you create. If you add a server value to the above URI, you limit all SFTP connections that you create to that specified server.

3. Save your changes and restart the platform.

Enforce authentication methods

By default, the Trifacta application enables use of two different authentication mechanisms:

- Basic - use a password to access the SFTP server
- SSHKey - use a public SSHKey and password to access the SFTP server

Along with basic and SSH key, the SFTP servers in your environment may be configured with other authentication methods, and those methods sometimes take precedence. As a result, when using default authentication methods, SFTP connections from the Trifacta platform can fail to connect to the SFTP server.

To eliminate these issues, you can configure the Trifacta application to enforce usage of one of the following authentication schemes. These schemes are passed to the SFTP server during connection time, which forces the server to use the appropriate method of authentication. When the following parameter is specified, SFTP connections can be configured using the listed methods and should work for connecting to the server.

> **NOTE:** Enforcement applies to connections created via the APIs as well. After configuration, please be sure to use one of the enforced authentication methods when configuring your SFTP connections through the application or the APIs.

Steps:

1. To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. Some of these settings may not be available through the *Admin Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter in the configuration file:

   ```
   "batchserver.workers.filewriter.hadoopConfig.sftp.PreferredAuthentications"
   ```

3. Set the parameter value according to the following:

Preferred authentication method	Parameter value	Description
Basic	`"password"`	Basic password authentication method is used to connect to the SFTP server. **NOTE:** You must configure your SFTP server connection in the platform to use the Basic method.
SSHKey	`"publickey"`	SSH Key authentication method is used. **NOTE:** You must configure your SFTP server connection in the platform to use the SSHKey method.
both	`"publickey, password"`	Both methods of authentication are supported.

4. Save your changes and restart the platform.

Java VFS service

Use of SFTP connections requires the Java VFS service in the Trifacta platform.

> **NOTE:** This service is enabled by default.

For more information on configuring this service, see *Configure Java VFS Service*.

Create Connection

Create through application

Any user can create a SFTP connection through the application.

> **NOTE:** Only an administrator can make a connection available for all users.

Steps:

1. Login to the application.
2. In the menu, select **User menu > Preferences > Connections**. See *Connections Page*.
3. In the Connections page, click **Create Connection**. See *Create Connection Window*.
4. In the Create Connection window, click the SFTP connection card.
5. Specify the properties for your SFTP server.

Property	Description
Host	The hostname of the FTP server to which you are connecting. Do not include any protocol identifier (`sftp://`).
Port	The port number to use to connect to the server. Default port number is `22`.
Credential Type	Select one of the following: `basic` - authenticate via username and password `SSH Key` - authenticate via username and SSH key
User Name	The username to use to connect.
Password	(Basic credential type) The password associated with the username.
SSH Key	(SSH Key credential type) The SSH key that applies to the username.
Test Connection	Click this button to test the connection that you have specified.
Default Directory	Absolute path on the SFTP server where users of the connection can begin browsing.
Block Size (Bytes)	Fetch size in bytes for each read from the SFTP server. **NOTE:** Raising this value may increase speed of read operations. However, if it is raised too high, resources can become overwhelmed, and the read can fail.
Connection Name	The name of the connection as you want it to appear in the application.
Description	This description is displayed in the application.

   For more information, see *Create Connection Window*.
6. Click **Save**.

Create through APIs

1. Acquire the vendor and type information. See *Connection Types*.
2. Create the connection through the APIs. See *https://api.trifacta.com/ee/es.t/index.html#operation/createConnection*

Create Tableau Server Connections

Contents:

- *Limitations*
- *Enable Hyper format*
- *Enable TDE format*
  - *Download and Install Tableau SDK*
  - *Enable TDE format*
- *Configure Permissions*
- *Create Tableau Server Connection*
  - *Create through application*
  - *Create through APIs*

This section describes the basics of creating Tableau Server connections from within the application.

> **NOTE:** You can export Tableau files as part of exporting results from the platform. For more information, see *Publishing Dialog*.

Limitations

- This connection type only enables publication.
  - You cannot read data from Tableau Server.
  - When created in the application, publish-only connections must be created through the Connections page.

Enable Hyper format

Hyper format generation is enabled by default. To enable the generation of results into Hyper format, please verify the following:

Steps:

1. Login as an administrator.
2. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
3. Locate the following setting:

   ```
   Hyper output format
   ```

4. Set it to `Enabled`.
5. No other configuration is required.

Enable TDE format

> **NOTE:** The TDE format has been superseded by the Hyper format. Publication to TDE format will be deprecated in a future release. Please switch to using Hyper format.

Download and Install Tableau SDK

To enable generation of TDE files and publication to Tableau Server, the Tableau Server SDK must be licensed, downloaded, and installed in the Trifacta platform.

Steps:

1. Navigate to the following:
   https://onlinehelp.tableau.com/current/api/sdk/en-us/SDK/tableau_sdk_installing.htm#downloading
2. Complete any required licensing steps.
3. Download this version: **Tableau SDK for C/C++/Java (64-bit)**.
4. Transfer the file to the Trifacta node.
   a. Extract it in a directory where the `trifacta` account has read and execute permissions.

      NOTE: The above directory should be located outside of the install directory for the platform software.

   b. Retain the path to this directory. This directory is assumed to have the following name: `<tableau-extract-dir>`.
5. Platform configuration must be updated to point to this SDK. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
6. Update the following property:

   ```
   "batch-job-runner.env.LD_LIBRARY_PATH" = "<tableau-extract-dir>/lib64/tableausdk/"
   ```

7. Add to the Batch Job Runner classpath to the current classpath (`<current_classpath_values>`). You must replace `<tableau-extract-dir>` with the path where you extracted the Tableau Server SDK:

   ```
   "batch-job-runner.classpath" ="<current_classpath_values>:<tableau-extract-dir>/lib64/tableausdk/Java
   /tableaucommon.jar:<tableau-extract-dir>/lib64/tableausdk/Java/tableauserver.jar:<tableau-extract-dir>
   /lib64/tableausdk/Java/tableauextract.jar"
   ```

8. Save your changes and restart the platform.

Enable TDE format

To enable the generation of results into TDE format, please complete the following:

Steps:

1. Login as an administrator.
2. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
3. Locate the following setting:

   ```
   TDE output format
   ```

4. Set it to `Enabled`.

Configure Permissions

The user who is publishing to Tableau Server must have exec permissions on the temporary directory on the backend datastore. This directory is used to write the intermediate file format locally, before it is published to Tableau Server.

> **NOTE:** These permissions must be applied for both TDE and Hyper format.

For more information, see *Supported File Formats*.

Create Tableau Server Connection

Create through application

Any user can create a Tableau Server connection through the application.

> **NOTE:** Only an administrator can make a Tableau Server connection available for all users.

Steps:

1. Login to the application.
2. In the menu, select **User menu > Preferences > Connections**. See *Connections Page*.
3. In the Connections page, click **Create Connection**. See *Create Connection Window*.
4. In the Create Connection window, click the Tableau Server connection card.
5. Specify the properties for your Tableau Server.

Property	Description
Server URL	The URL to the Tableau Server to which you are connecting. To specify an SSL connection, use `https://` for the protocol identifier. **NOTE:** By default, this connection assumes that the port number is `80`. To use a different port, you must specify it as part of the Server name value: `http://<Tableau_Server_URL>:<port_number>`

Site	Enter the value that appears after `/site/` in your target location.
	
	Example target URL:
	
	`https://tableau.example.com/#/site/MyNewTargetSite`
	
	Enter the following for the Site setting:
	
	`MyNewTargetSite`

User Name	The username to use to connect.
Password	The password associated with the username.
Test Connection	Click this button to test the connection that you have specified.

Connection Name	The name of the connection as you want it to appear in the user interface.
Description	This description is displayed in the user interface.

For more information, see *Create Connection Window*.

6. Click **Save**.

Create through APIs

1. Acquire the vendor and type information. See *Connection Types*.
2. Create the connection through the APIs. See *https://api.trifacta.com/ee/es.t/index.html#operation/createConnection*

Create Salesforce Connections

Contents:

- *Limitations*
- *Pre-requisites*
- *Enable*
- *Configure*
  - *Connect string options*
- *Use*

You can create connections to your Salesforce instance from Trifacta® Wrangler Enterprise. This connector is designed as a wrapper around the Salesforce REST API.

Limitations

- This is a read-only connection.
- Single Sign-On (SSO) is not supported.
- Custom domains are not supported.

Pre-requisites

- The account used to login from Trifacta Wrangler Enterprise must access Salesforce through a security token.

  > **NOTE:** Please contact your Salesforce administrator for the Server Name and the Security Token values.

- The logged-in user must have required access to the tables and schema. For more information, see *Using Salesforce*.

- If you haven't done so already, you must create and deploy an encryption key file for the Trifacta node to be shared by all relational connections. For more information, see *Create Encryption Key File*.

Enable

For more information, see *Enable Relational Connections*.

Configure

To create this connection:

- In the Connections page, select the Applications tab. Click the Salesforce card.
- See *Connections Page*.

This connection can also be created using the API.

- For details on values to use when creating via API, see *Connection Types*.
- See *API Reference*.

Modify the following properties as needed:

Property	Description
Server Name	Enter the host name of your Salesforce implementation. Example value: `exampleserver.salesforce.com`
Connect String Options	Apply any connection string options that are part of your authentication to Salesforce. For more information, see below.
User Name	Username to use to connect to the database.
Password	Password associated with the above username.
Security Token generated in account	Paste the security token associated with the account to use for this connection.
Test Connection	After you have defined the connection credentials type, credentials, and connection string, you can validate those credentials.
Default Column Data Type Inference	Set to `disabled` to prevent the platform from applying its own type inference to each column on import. The default value is `enabled` .
Connection Name	Display name of the connection
Connection Description	Description of the connection, which appears in the application.

Connect string options

You can apply the following options to your connection string.

Include system columns

By default, Salesforce does not include system columns generated by Salesforce in any response. To include them, add the following value to the Connect String Options textbox:

```
ConfigOptions=(auditcolumns=all;mapsystemcolumnnames=0)
```

Unlimited number of calls

By default, Salesforce imposes a limit on the number of calls that can be made through the REST APIs by this connector.

You can make the number of calls unlimited by appending the following to the Connect String Options textbox:

```
StmtCallLimit=0
```

Use

You can import datasets from Salesforce through the Import Data page. See *Import Data Page*.

- See *Salesforce Browser*.
- For more information on interacting with Salesforce, see *Using Salesforce*.

Enable Alation Sources

Contents:

- *Limitations*
- *Pre-requisites*
- *Enable Alation Navigation Integration*
  - *Testing Alation browsing integration*
- *Enable Open With Integration*
  - *Testing open with integration*

If you have integrated the Trifacta® platform with Hive, you can integrate it with Alation to simplify finding datasets within Hive for import. The Alation integration supports the following methods:

1. Read directly from Alation through an Alation Navigator integrated into the Import Data page.
2. Locate tables through Alation and then open them with the Trifacta platform.

Alation is a data catalog service for Hive. For more information, see *www.alation.com.*

Limitations

- You can import only tables from Alation.
  - You cannot use queries or select columns for import into the platform.
- Cluster security features such as secure impersonation and Kerberos are supported if both users in the integration are authenticated and impersonated.

Pre-requisites

- Alation version 4.10.0 or later
- Your enterprise environment must have a deployed instance of Hive to which the Trifacta platform has already been integrated. See *Configure for Hive*.
- You must have credentials to access Alation. You can sign up through the Alation Catalog Navigator after the integration is complete.

  > **NOTE:** Your Hive administrator and Alation administrator must ensure that your accounts have the appropriate permissions to search for and access datasets within these separate deployments.

- You must acquire the URL for the host of your Alation deployment.

Enable Alation Navigation Integration

Steps:

> **NOTE:** Although the integration to Alation appears as a connection in the application, the connection cannot be created through the application. Please complete the following steps.

1. Login to the platform as an administrator.
2. From the menu, select **User menu > Admin console > Admin settings**.
3. Search for `alation`.
4. Update the values for the following properties accordingly:

Property	Description
`alation.sdkPath`	This value identifies the path on the Alation server to where their integration SDK is stored. Do not modify this value.
`alation.enabled`	Set this value `true` to enable the integration.
`alation.catalogHost`	Set this value to the URL of the web interface for the Alation deployment.

5. Save your changes.

Testing Alation browsing integration

1. Restart services. See *Start and Stop the Platform*.
2. When the platform has restarted, login.
3. Click **Datasets**. Then, click **Import Data**.
4. In the Import Data page, the Alation connection should appear in the left nav bar. Select it.
5. Click **Launch Alation Catalog**.
6. If prompted, enter your Alation credentials.
7. Navigate to select a Hive table. For more information, see *https://alationhelp.zendesk.com/hc/en-us*.
8. Click **Select**.
9. The table is added to the Import Data page.
10. Import as normal.

Try running a simple job. For more information, see *Verify Operations*.

If ad-hoc publishing to Hive has been enabled, you can export the generated results to Hive and then attempt to re-import through Alation.

> **NOTE:** There may be a delay before the Trifacta results appear in Alation. If necessary, you can manually refresh the catalog from inside Alation.

Enable Open With Integration

Optionally, you can enable Alation users to open Hive tables from Alation in the Trifacta platform.

> **NOTE:** To support this integration, end users must disable popup blockers in the browser. For more information, please see your browser's documentation.

> **NOTE:** If Kerberos is enabled, you must be authenticated into the Trifacta platform and Alation at the same time.

> **NOTE:** HTTPS is not supported.

Steps:

1. Acquire an Alation API token.
    a. Visit the following URL:

```
http://<alation_host>/admin/misc
```

   b. Click **Get API token**.
   c. Copy the generated API token to the clipboard.
2. Paste the API token into the following cURL command and execute it.

   a. HTTP:

```
curl -X POST 'http://<alation_host>/integration/catalog_chooser/register_opener/' -H "Content-
Type: application/json" -H "Token: <token_key>" -d '{"endpoint":"http://<platform_host>:
<platform_port_num>/import/data?uri=${dataSource.jdbcUri}&table=${qualifiedName}","
endpoint_type":"NAVIGATE","name":"Trifacta","accept_object_types":["table"],"
accept_data_source_types":["hive", "hive2"]}'
```

   b. HTTPS: Change the protocol identifier for both URLs to `https` and remove the platform port number.

```
curl -X POST 'https://<alation_host>/integration/catalog_chooser/register_opener/' -H "Content-
Type: application/json" -H "Token: <token_key>" -d '{"endpoint":"https://<platform_host>/import
/data?uri=${dataSource.jdbcUri}&table=${qualifiedName}","endpoint_type":"NAVIGATE","name":"
Trifacta","accept_object_types":["table"],"accept_data_source_types":["hive", "hive2"]}'
```

   where:

Parameter	Description
`<alation_host>`	Hostname of the Alation server
`<token_key>`	The token value that was generated in Alation
`<platform_host>`	Hostname of the Trifacta platform
`<platform_port_num>`	Port number of the Trifacta platform. Default is `3005`.

3. A successful execution of the above command logs the following JSON message:

```
{"id":1,"name":"Trifacta","endpoint":"http://<platform_host>:<platform_port_num>/import/data","
accept_object_types":["table"],"accept_data_source_types":["hive",
"hive2"]}
```

Testing open with integration

Steps:

1. Login to Alation.
2. Search for or navigate to a database table. Click the **Open With...** button. From the drop-down, select Trifacta.
3. The table appears as an imported dataset in the Imported Dataset page.
4. You can import the dataset into a new or existing flow.

For more information, see *Import Data Page*.

Enable Waterline Sources

Contents:

- *Limitations*
- *Pre-requisites*
- *Enable Waterline Integration*
    - *Testing Waterline browsing integration*

You can integrate the Trifacta® platform with the Waterline data catalog to simplify finding datasets within your enterprise data lake. The Waterline integration supports the following methods:

1. Read directly from Waterline through a search box integrated into the Import Data page.
2. Locate assets through Waterline and open them with the Trifacta platform.

Waterline Data is a data catalog service for Hive. For more information, see *www.waterlinedata.com*.

Limitations

> **NOTE:** This integration is not supported in the Wrangler Enterprise desktop application.

Pre-requisites

- Waterline 4.0 and higher
- Waterline must be integrated with your deployment of the Trifacta platform. For more information, please contact your Waterline administrator.
- You must have credentials to access Waterline.

  > **NOTE:** Your Waterline administrator must ensure that your account has the appropriate permissions to search for and access datasets within Waterline and its integrated sources.

- You must acquire the URL for the host of your Waterline deployment.
- You must acquire the hostname and port for the Trifacta platform.

Enable Waterline Integration

Steps:

> **NOTE:** Although the integration appears as a connection in the application, the connection cannot be created through the application. Please complete the following steps.

1. Login to the platform as an administrator.
2. From the menu, select **User menu > Admin console > Admin settings**.
3. Search for `waterline`.
4. Update the values for the following properties accordingly:

Property	Description

`waterline.searchPath`	This value identifies the path on the Waterline server for executing a search. Do not modify this value.
`waterline.enabled`	Set this value `true` to enable the integration.
`waterline.catalogHost`	Set this value to the URL of the Waterline deployment.

5. Save your changes.

Testing Waterline browsing integration

1. Restart services. See *Start and Stop the Platform*.
2. When the platform has restarted, login.
3. Click **Library**. Then, click **Import Data**.
4. In the Import Data page, the Waterline connection should appear in the left nav bar. Select it.
5. Enter your search term, which can be a filename or description of the dataset.
6. Browse and select your dataset.
7. From the Gear menu in Waterline, select **Wrangle**.
8. The dataset is imported into the platform where it can be added to flows. See *Import Data Page*.

Try running a simple job. For more information, see *Verify Operations*.

Configure Connectivity

Contents:

- *Enable*
  - *Data Service*
- *Relational Features*
  - *Custom SQL Query*
  - *JDBC Ingestion*
- *Configure Security*
- *Enable SSO Connections*
- *Type Inference*

This section covers the following areas around general connectivity of the Trifacta® platform.

Additional configuration may be required for individual connection types. For more information, see *Connection Types*.

Enable

The platform automatically enables connectivity to relational databases for reading in datasets and writing results back out.

> **NOTE:** Relational connectivity requires the use of an encryption key file, which must be created and deployed before you create relational connections. For more information, see *Create Encryption Key File* in the Install Guide.

Data Service

The platform streams records from relational sources through the data service. These records are applied to transformation and sampling jobs on the Photon running environment, which is native to the Trifacta node.

> **Tip:** In general, you should not have to modify settings for the data service. However, if you are experiencing general performance issues or issues with specific connection types, you may experiment with settings in the data service.

For more information, see *Configure Data Service* in the Configuration Guide.

Relational Features

Custom SQL Query

To enhance performance of your relational datasets, you can enable the use of custom SQL queries against your relational datasources, which allows you to pre-filter your datasets before you ingest them into the platform. This feature is enabled by default, but additional configuration can be applied. See *Enable Custom SQL Query*.

JDBC Ingestion

As needed, the platform can be configured to ingest data from your relational datasources to the base storage layer for faster execution of Spark-based jobs. See *Configure JDBC Ingestion*.

Configure Security

For more information, see *Configure Security for Relational Connections*.

Enable SSO Connections

If you have enabled Kerberos on the Hadoop cluster, you can leverage the Kerberos global keytab to enable SSO connections to relational sources. See *Enable SSO for Relational Connections*.

Type Inference

By default, the platform applies type inferencing to all imported datasources. However, for schematized sources, you may wish to disable type inferencing from the platform instead relying on the types provided from the source.

> **Tip:** You can also toggle the use of type inferencing for individual connections or for individual imported datasets.

For more information, see *Configure Type Inference*.

Enable Relational Connections

Contents:

- *Supported Relational Databases*
- *Ports*
- *Enable*
- *Limitations*
- *Execution at scale*
- *Password Encryption Key File*

The Trifacta® platform can be configured to access data stored in relational database sources over JDBC protocol. When this connection method is used, individual database tables and views can be imported as datasets.

Supported Relational Databases

The Trifacta platform can natively connect to these relational database platforms. Natively supported versions are the following:

- Oracle 12.1.0.2
- SQL Server 12.0.4
- PostgreSQL 9.3.10
- Teradata 14.10+

> **NOTE:** To enable Teradata connections, you must download and install Teradata drivers first. For more information, see *Enable Teradata Connections*.

Additional relational connections can be enabled and configured for the platform. For more information, see *Connection Types*.

Ports

For any relational source to which you are connecting, the Trifacta node must be able to access it through the specified host and port value.

Please contact your database administrator for the host and port information.

Enable

This feature is enabled automatically.

> **NOTE:** Disabling this feature hides existing relational connections.

Disable relational publishing

By default, relational connections are read/write, which means that users can create connections that enable writing back to source databases.

- When this feature is enabled, writeback is enabled for all natively supported relational connection types. See *Connection Types*.
- Depending on the connection type, the Trifacta platform writes its data to different field types in the target database. For more information, see *Type Conversions*.

- Some limitations apply to relational writeback. See Limitations below.

As needed, you can disable this feature.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter and set it to `false`:

   ```
   "webapp.connectivity.relationalWriteback.enabled": true,
   ```

3. Save changes and restart the platform.

Publishing through relational connections is disabled.

Limitations

> **NOTE:** Unless otherwise noted, authentication to a relational connection requires basic authentication (username/password) credentials.

- You cannot swap relational sources if they are from databases provided by different vendors. See *Flow View Page*.
- There are some differences in behavior between reading tables and views. See *Using Databases*.

Limitations on relational publishing:

> **When the relational publishing feature is enabled, it is automatically enabled for all platform-native connection types. You cannot disable relational publishing for Oracle, SQL Server, PostgreSQL, or Teradata connection types. Before you enable, please verify that all user accounts accessing databases of these types have appropriate permissions.**

> **NOTE:** Writing back to the database utilizes the same user credentials and therefore permissions as reading from it. Please verify that the users who are creating read/write relational connections have appropriate access.

- You cannot ad-hoc publish to a relational target. Relational publishing is only supported through the Run Job page.
- You write to multiple relational outputs from the same job only if they are from the same vendor.
  - For example, if you have two SQL Server connections A and B, you can write one set of results to A and another set of results to B for the same job.
  - If A and B are from different database vendors, you cannot write to them from the same job.

Execution at scale

Jobs for large-scale relational sources can be executed on the Spark running environment. After the data source has been imported and wrangled, no additional configuration is required to execute at scale.

NOTE: End-to-end performance is likely to be impacted by:

- streaming data volumes over 1 TB from the source,
- streaming from multiple concurrent sources,
- overall network bandwidth.

When the job is completed, any temporary files are automatically removed from HDFS.

For more information, see *Run Job Page*.

Password Encryption Key File

Relational database passwords are encrypted using key files:

- **Passwords in transit:** The platform uses a proprietary encryption key that is invoked each time a relational password is shared among platform services.
- **Passwords at rest:** For creating connections to your relational sources, you must create and reference your own encryption key file. This encryption key is accessing your relational connections from the web application. For more information, see *Create Encryption Key File*.

Enable Custom SQL Query

Contents:

- *Limitations*
- *Enable*
- *Use Custom SQL Queries*

To improve performance of your Hive or relational connections, custom SQL queries can be enabled to push the initial filtration of table rows and columns back the database, which is more efficient at performing this task. Instead of loading the entire table into the Trifacta® application and then performing the filtration through the Transformer page, you can insert basic SQL commands as part of your relational queries to collect only the rows and columns of interest from the source.

When enabled, custom SQL query is available for all relational sources.

Limitations

See *Create Dataset with SQL*.

Enable

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following setting:

   ```
   Enable custom SQL Query
   ```

Setting	Description
`enabled`	Set to `true` to enable the SQL pushdown feature. By default, this feature is enabled.
3. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

   Locate the following setting:

   ```
   "webapp.connectivity.customSQLQuery.enableMultiStatement": false,
   ```

Setting	Description
`enableMultiStatement`	When set to `true`, you can insert multi-line statements in your SQL pushdown queries. The default is `false`. **NOTE:** Use of multi-line SQL has limitations. See *Create Dataset with SQL*.

4. Save the file.
5. As needed, you can configure the maximum permitted load time before timeout from the application. See *Configure Application Limits*.
6. Restart the platform. See *Start and Stop the Platform*.

Use Custom SQL Queries

When custom SQL query is enabled, you can enter customized SQL statements in the imported dataset page as part of the import process. See *Import Data Page*.

For examples, see *Create Dataset with SQL*.

After a dataset has been imported using custom SQL, you can edit the SQL as needed. See *Dataset Details Page* .

Configure JDBC Ingestion

Contents:

- *Overview*
  - *Recommended Table Size*
  - *Performance*
  - *Limitations*
- *Enable*
- *Configure*
  - *Configure Ingestion*
  - *Configure Storage*
- *Monitoring Progress*
- *Logging*

This section describes some of the configuration options for the JDBC (relational) ingestion and caching features, which enables execution of large-scale JDBC-based jobs on the Spark running environment.

Overview

Data ingestion works by streaming a JDBC source into a temporary storage space in the base storage layer. The job can then be run on Spark, the default running environment. When the job is complete, the temporary data is removed from base storage or retained in the cache (if it is enabled).

- Data ingestion happens only for Spark jobs. Data ingestion does not apply to Trifacta Photon jobs.

- Data ingestion applies only to JDBC sources that are not native to the running environment. For example, JDBC ingestion is not supported for Hive.
- Supported for HDFS and other large-scale backend datastores.

When data is read from the source, the Trifacta® platform can populate a user-specific **ingest cache**, which is maintained to limit long load times from JDBC sources and to improve overall platform performance.

- The cache is allowed to remain for a predefined expiration limit, after which any new requests for the data are pulled from the source.

  > **NOTE:** Expired cache files are not automatically purged at expiration.

- If the cache fails for some reason, the platform falls back to ingest-only mode, and the related job should complete as expected.

Job Type	JDBC Ingestion Enabled only	JDBC Ingestion and Caching Enabled
transformation job	Data is retrieved from the source and stored in a temporary backend location for use in sampling.	Data is retrieved from the source for the job and refreshes the cache where applicable.

sampling job	See previous.	Cache is first checked for valid data objects. Outdated objects are retrieved from the data source. Retrieved data refreshes the cache. **NOTE:** Caching applies only to full scan sampling jobs. Quick scan sampling is performed in the Trifacta Photon web client. As needed you can force an override of the cache when executing the sample. Data is collected from the source. See *Samples Panel*.

Recommended Table Size

Although there is no absolute limit, you should avoid executing jobs on tables over several 100 GBs. Larger data sources can significantly impact end-to-end performance.

NOTE: This recommendation applies to all JDBC-based jobs.

Performance

Rule of thumb:

- For a single job with 16 ingest jobs occurring in parallel, maximum expected transfer rate is 1 GB/minute.

Scalability:

- Good for 100's of MBs. Not good for tables of GB size.
- 1 ingest job per source, meaning a dataset with 3 sources = 3 ingest jobs.
- Rule of thumb for max concurrent jobs for a similar edge node:

```
max concurrent sources = max cores - cores used for services
```

    - Above is valid until the network becomes a bottleneck. Internally, the above maxed out at about 15 concurrent sources.
    - Default concurrent jobs = 16, pool size of 10, 2 minute timeout on pool. This is to prevent overloading of your database.
    - Adding more concurrent jobs once network has bottleneck will start slow down all the transfer jobs simultaneously.
- If processing is fully saturated (# of workers is maxed):
    - max transfer can drop to 1/3 GB/minute.
    - Ingest waits for two minutes to acquire a connection. If after two minutes a connection cannot be acquired, the job fails.
- When job is queued for processing:
    - Job is silently queued and appears to be in progress.
    - Service waits until other jobs complete.
    - Currently, there is no timeout for queueing based on the maximum number of concurrent ingest jobs.

Limitations

- JDBC ingest caching is not supported for Hive.

Enable

To enable JDBC ingestion and performance caching, both of the following parameters must be enabled.

> **NOTE:** For new installations, this feature is enabled by default. For customers upgrading to Release 5.1 and later, this feature is disabled by default.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Parameter Name	Description
`webapp.connectivity.ingest.enabled`	Enables JDBC ingestion. Default is `true`.
`feature.jdbcIngestionCaching.enabled`	Enables caching of ingested JDBC data. **NOTE:** `webapp.connectivity.ingest.enabled` must be set to `true` to enable JDBC caching. When disabled, no caching of JDBC data sources is performed.
`feature.enableLongLoading`	When enabled, you can monitor the ingestion of long-loading JDBC datasets through the Import Data page. Default is `true`. **Tip:** After a long-loading dataset has been ingested, importing the data and loading it in the Transformer page should perform faster.
`longloading.addToFlow`	When long-loading is enabled, set this value to `false` to enable monitoring of the ingest process when large relational sources are added to a flow. Default is `false`. See *Flow View Page*.
`longloading.addToLibrary`	When long-loading is enabled, set this value to `false` to enable monitoring of the ingest process when large relational sources are added to the library. Default is `false`. See *Library Page*.

Configure

In the following sections, you can review the available configuration parameters for JDBC ingest and JBC performance caching.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Configure Ingestion

Parameter Name	Description
`batchserver.workers.ingest.max`	Maximum number of ingester threads that can run on the Trifacta platform at the same time.
`batchserver.workers.ingest.bufferSizeBytes`	Memory buffer size while copying to backend storage. A larger size for the buffer yields fewer network calls, which in rare cases may speed up ingest.

`batch-job-runner.cleanup.enabled`	Clean up after job, which deletes the ingested data from backend storage. Default is `true`. **NOTE:** If JDBC ingestion is disabled, relational source data is not removed from platform backend storage. This feature can be disabled for debugging and should be re-enabled afterward. **NOTE:** This setting rarely applies if JDBC ingest caching has been enabled.

Configure Storage

When files are ingested, they are stored in one of the following locations:

- **If caching is enabled:**
  - **If the global datasource cache is enabled:** files are stored in a user-specific sub-folder of the path indicated by the following parameter: `hdfs.pathsConfig.globalDatasourceCache`
  - **If the global cache is disabled:** files are stored in a sub-folder of the output area for each user, named: `/.datasourceCache`.
- **If caching is disabled:** files are stored in a sub-folder within the jobs area for the job group. Ingested files are stored in as `.trifacta` files.

NOTE: Whenever a job is run, its source files must be re-ingested. If two or more datasets in the same job run share the same source, only one copy of the source is ingested.

Additional information is provided below.

Global or user cache

Parameter	Description
`datasourceCaching.useGlobalDatasourceCache`	When set to `true`, the platform uses the global data source cache location for storing cached ingest data. **NOTE:** When global caching is enabled, data is still stored individual locations per user. Through the application, users cannot access the cached objects stored for other users. When set to `false`, the platform uses the output directory for each user for storing cached ingest data. Within the output directory, cached data is stored in the `.datasourceCache` directory. **NOTE:** You should verify that there is sufficient storage in each user's output directory to store the maximum cache size as well as any projected uploaded datasets.
`hdfs.pathsConfig.globalDataSourceCache`	Specifies the path of the global datasource cache, if it is enabled. Specify the path from the root folder of HDFS or other backend datastore.

Cache sizing

Parameter	Description
`datasourceCaching.refreshThreshold`	The number of hours that an object can be cached. If the object has not been refreshed in that period of time, the next request for the datasource collects fresh data from the source. By default, this value is set to `168` (one week).

`datasourceCaching.maxSize`	Maximum size in bytes of the datasource cache. This value applies to individual user caches when either global or user-specific caching is enabled.

Logging

Parameter Name	Description
`data-service.systemProperties.logging.level`	When the logging level is set to `debug`, log messages on JDBC caching are recorded in the data service log. **NOTE:** Use this setting for debug purposes only, as the log files can grow quite large. Lower the setting after the issue has been debugged. See Logging below.

Monitoring Progress

You can use the following methods to track progress of ingestion jobs.

- **Through application:** In the Jobs page, you can track progress of all jobs, including ingestion. Where there are errors, you can download logs for further review.
  - See *Jobs Page*.
  - See Logging below.
- **Through APIs:**
  - You can track status of `jobType=ingest` jobs through the API endpoints.
  - From the above endpoint, get the ingest jobId to track progress.
  - See *https://api.trifacta.com/ee/es.t/index.html#operation/getJobGroup*

Logging

Ingest:

During and after an ingest job, you can download the job logs through the Jobs page. Logs include:

- All details including errors
- Progress on ingest transfer
- Record ingestion

See *Jobs Page*.

Caching:

When the logging level is set to `debug` for the data service and caching is enabled, cache messages are logged. These messages include:

- Cache hits and misses
- Cache key generation

Configure Security for Relational Connections

Contents:

- *User Security*
  - *Connection Security Levels*
  - *Credential Sharing*
- *Technical Security*
  - *Encryption Key File*
  - *SSL*
  - *Configure long load timeout limits*
  - *Enable SSO authentication*

You can apply the following Trifacta® platform features to relational connections to ensure compliance with enterprise practices.

User Security

Connection Security Levels

Connection Security Level	Description
Private	Private connections are created by individuals and are by default accessible only to the individual who created them.
Private and shared	Optionally, they can be shared by individuals with other users. **NOTE:** If needed, credential sharing can be disabled. See below.
Global	Global connections are either created by administrators or are private connections promoted to global by administrators.

Credential Sharing

By default, users are permitted to share credentials through the application. Credentials can be shared in the following ways:

- A user can create a private connection to a relational database. Through the application, this private connection can be shared with other users, so that they can access the creator's datasets.
- When sharing a flow with another user, the owner of the flow can choose to share the credentials that are necessary to connect to the datasets that are the sources of the flow.

As needed, credential sharing can be disabled.

NOTE: If enterprise policy is to disable the sharing of credentials, collaborators may need to be permitted to store their source data in shared locations.

Tip: Credential sharing can be disabled by individual users when they share a connection. The connection is shared, but the new user must provide new credentials to use the connection.

Steps:

To disable credential sharing at the global level:

1. Login to the application as an administrator.
2. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Locate the following parameter. Set this property to `false`:

```
"webapp.enableCredentialSharing": true,
```

4. Save your changes and restart the platform.

Technical Security

The following features enhance the security of individual and global relational connections.

Encryption Key File

Relational database passwords are encrypted using key files:

- **Passwords in transit:** The platform uses a proprietary encryption key that is invoked each time a relational password is shared among platform services.
- **Passwords at rest:** For creating connections to your relational sources, you must create and reference your own encryption key file. This encryption key is accessing your relational connections from the web application.

This encryption key file must be created and installed on the Trifacta node. For more information, see *Create Encryption Key File*.

SSL

You can enable SSL by adding the following string to the Connect String Opts field:

```
?ssl=true;
```

> **Tip:** Some connection windows have a Use SSL checkbox, which also works.

Configure long load timeout limits

For long loading relational sources, a timeout is applied to limit the permitted load time. As needed, you can modify this limit to account for larger load times.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

1. Locate and edit the following parameter:

```
"webapp.connectivity.longLoadTimeoutMillis": 120000,
```

2. Save your changes and restart the platform.

Property	Description

`longLoadTimeoutMillis`	Max number of milliseconds to wait for a long-loading data source. The default value is `120000` (2 minutes).

For additional relational configuration settings, see *Configure Data Service*.

Enable SSO authentication

Relational connections can be configured to leverage your enterprise Single Sign-On (SSO) infrastructure for authentication. Additional configuration is required. For more information, see *Enable SSO for Relational Connections*.

Enable SSO for Relational Connections

Contents:

- *Limitations*
- *Pre-requisites*
- *Configure*
  - *Configure JAAS file and path*
  - *JAAS file*
  - *Specify Kerberos configuration file*
  - *Configure vendor definition file*
- *Example Setup*
- *Use*
  - *Sharing*

This section describes how to enable relational connections to leverage your Hadoop Single Sign-On (SSO) infrastructure. When this feature is enabled and properly configured, users can create relational (JDBC) connections that use SSO that you have already configured.

Connections that were created before this feature is enabled continue to operate as expected without modification.

Limitations

- For this release, this feature applies to SQL Server connections only.
- Cross-realm is not supported. As a result, the SQL Server instance, service principal, and Trifacta® principal must be in the same Kerberos realm.

Pre-requisites

- **Kerberos SSO:** You must set up SSO authentication to the Hadoop cluster using Kerberos. This feature uses the global Kerberos keytab. For more information, see *Configure for Kerberos Integration*.

Configure

Configure JAAS file and path

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Parameter	Description
`webapp.connectivity.kerberosDelegateConfigPath`	Path on the Trifacta node to the location of the JAAS configuration file required by the DataDirect driver. **NOTE:** The default location is listed below. You may wish to move this file to a location outside of the Trifacta installation to ensure that the file is not overwritten during upgrades. More information on this file is provided below.

JAAS file

For connections that support Kerberos-delegated authentication, the underlying driver supports a JAAS file in which you can provide environment-specific configuration to the driver. As needed, you can modify this file.

Connection Type	Default path to JAAS file
SQL Server	`%(topOfTree)s/services/data-service/build/conf/kerberosdelegate.config`

Example JAAS file for SQL Server

Below is an example file, where you must apply the Kerberos global keytab and principal values that are to be used to authenticate to use the Kerberos-delegated connections of this type:

```
```

where:

- `keytab` = the absolute path on the Trifacta node where the Kerberos global keytab is located.
- `principal` = Set to the service principal name of the user's service account in LDAP.

Specify Kerberos configuration file

On the Trifacta node, locate the following file:

```
<root>/etc/krb5.conf
```

If it doesn't exist, create it with the following content, some of which you must specify:

```
```

Setting	Description
`default_realm`	Set this value to your default Kerberos realm.

`forwardable`	This value must be set to `true`.
`kdc`	For each realm that you create, you must create an entry in `[realms]`. For the `kdc` entry, apply the KDC domain that the JDBC connection should use.
`my_domain`	For each domain to which the Kerberos delegation applies, you must create an entry in `[domain_realm]` . Entries should look like the following: `example.com = EXAMPLE.COM`

Modify the location of the Kerberos configuration file

If you need to move the location of the file from the default one, please complete the following:

Steps:

1. If you haven't already done so, copy the file from its current location to its preferred location.
2. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Specify the path to the new location in the following parameter:

   ```
   "webapp.connectivity.krb5Path": "/etc/krb5.conf";
   ```

4. Save your changes.

Configure vendor definition file

For each vendor that supports SSO connections you must modify a setting in a configuration file on the Trifacta node. This change can only be applied for vendors that support Kerberized SSO connections.

Steps:

1. On the Trifacta node, navigate to the following directory:

   ```
   /opt/trifacta/services/data-service/build/conf/vendor
   ```

2. In the `vendor` directory, each JDBC vendor has a sub-directory. Open the vendor directory.
3. Edit `connection-metadata.json`.
4. Locate the `credentialType` property. Set the value to `kerberosDelegate`.
5. Save your changes and restart the platform.
6. When you create your connection, select `kerberosDelegate` from the Credential Type drop-down.

Example Setup

The following example uses the default Kerberos realm to set an SSO connection to a SQL Server instance. This example is intended to demonstrate one way in which you can set up your SSO connections.

Steps:

1. Create the Trifacta service principal:
   a. Form: `HTTP/serviceprincipal@REAM`

b. Enable this flag: `ok_to_auth_as_delegate`
c. Example:

```
kadmin -q "addprinc -randkey +ok_to_auth_as_delegate HTTP/serviceprincipal"
kadmin -q "addprinc -randkey +ok_to_auth_as_delegate HTTP/serviceprincipal@REALM"
```

d. For more information on delegation flags, see *https://web.mit.edu/kerberos/krb5-1.12/doc/admin/admin_commands/kadmin_local.html*

2. Generate a keytab for the Trifacta service principal.
3. Register the Trifacta service principal for Microsoft Sql Server instance:

a. Enable this flag: `ok_as_delegated`
b. Example:

```
kadmin -q "addprinc -randkey +ok_as_delegate MSSQLSvc/<FQDN>:<port>"
kadmin -q "addprinc -randkey +ok_as_delegate MSSQLSvc/<FQDN>:<port>@REALM"
kadmin -q "addprinc -randkey +ok_as_delegate MSSQLSvc/<FQDN>"
kadmin -q "addprinc -randkey +ok_as_delegate MSSQLSvc/<FQDN>@REALM"
```

c. For more information on setting this flag, see *https://docs.microsoft.com/en-us/sql/database-engine/configure-windows/register-a-service-principal-name-for-kerberos-connections?view=sql-server-2017* .

4. Create a linked SQL Server account:

a. Account must have the same name as the end-user principal.
b. Account needs connect permissions at least.

> **NOTE:** If you are using LDAP/AD SSO, you can register all of the above SPNs using AD mechanisms. You do not have to use the delegation flags. Delegation can be managed through the UI for the service account.

Use

When you create a new connection of a supported type, you can select the Kerberos Delegate credentials type. When selected, no username or credentials are applied as part of the connection object. Instead, authentication is determined via Kerberos authentication with the cluster.

- *Create SQL Server Connections*

Sharing

When sharing SSO connections, the credentials for the connection cannot be shared for security reasons. The Kerberos principal for the user with whom the connection is shared is applied. That user must have the appropriate permissions to access any required data through the connection. See *Overview of Sharing*.

Configure Type Inference

Contents:

- *Configure Type Inference for Schematized Sources*
- *Enable*
- *Use*
  - *Define for individual connections*
  - *Specify on dataset import*
- *Configure Type Inference in the Data Grid*
- *Type Inference on Export*

By default, the Trifacta® platform applies its own type inference to datasets when they are imported and again when new steps are applied to the data. This section provides information on how you can configure where type inference is applied in the platform.

Data types are inferred by the Trifacta platform when:

- Imported datasets are originally loaded.
- A new transformation step is added in a recipe.
- Non-inferred types are imported as String type.

> **Tip:** You can use the Change Column Type transformation to override the data type inferred for a column. However, if a new transformation step is added, the column data type is re-inferred, which may override your specific typing. You should consider applying Change Column Type transformations as late as possible in your recipes.

For more information on how the Trifacta platform applies data types to specific sources of data on import, see *Type Conversions*.

Configure Type Inference for Schematized Sources

Optionally, you can choose to disable type inference for schematized sources. A **schematized source** includes column data type information as part of the object definition. The following schematized sources are supported for import into the Trifacta platform:

- All JDBC sources

  > **NOTE:** You cannot disable type inference for Oracle sources. This is a known issue.

- Hive
- Redshift
- Avro file format

Enable

Type inference on schematized sources	Setting	Behavior
Enabled	"webapp.connectivity.disableRelationalTypeInference": false,	All imported datasets from schematized sources are automatically inferred by the type system in the Trifacta platform. The inferred data types may be different from those in the source. When the dataset is loaded, data types can be applied to individual columns through the application. Users can apply overrides for: • Individual connections • Individual datasets at time of import
Disabled	"webapp.connectivity.disableRelationalTypeInference": true,	For schematized data sources, type inference is not automatically inferred by Trifacta platform. Data type information is taken from the source schema and applied where applicable to the dataset. If there is no corresponding data type in the Trifacta platform, the data is imported as String type. Users can apply overrides for: • Individual connections • Individual datasets at time of import

Please perform the following configuration change to disable type inference of schematized sources at the global level.

Steps:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Change the following configuration setting to `true`:

```
"webapp.connectivity.disableRelationalTypeInference": false,
```

3. Save your changes.

Use

In the application, type inference can be applied to your imported data through the following mechanisms.

Define for individual connections

You can specify individual connections to apply or not apply Trifacta type inference when the connection is created or edited.

> **NOTE:** When Default Column Data Type Inference is disabled for an individual connection, Trifacta type inference can still be applied on import of individual datasets.

For more information, see *Create Connection Window*.

Specify on dataset import

When type inference has been disabled globally for schematized sources, you can choose to enable or disable it for individual source import.

> **Tip:** To compare how data types are imported from the schematized source or when applied by the Trifact a platform, you can import the same schematized source twice. The first instance of the source can be imported with type inference enabled, and the second can be imported with it disabled.

In the Import Data page, click **Edit Settings** on the data source card.

For more information, see *Import Data Page*.

Configure Type Inference in the Data Grid

Type inference is automatically enabled in the data grid. It cannot be disabled.

> **Tip:** You can override the Trifacta data type by applying a Change Column Type transformation.

When a new transformation step is applied, each column is re-inferred for its Trifacta data type.

Type Inference on Export

When you generate results, the current data types in the data grid are applied to the generated results.

If the publishing destination is a schematized environment, the generated results are written to the target environment based on the environment type. These data type mappings cannot be modified.

For more information on output types, see *Type Conversions*.

Troubleshooting Relational Connections

Contents:

- *Problem - Unable to access customer encryption key*
  - *Solution*
- *Problem - Retrieving sample data for large relational tables is very slow*
  - *Solution*
- *Related articles*

Problem - Unable to access customer encryption key

When trying to create, edit, or test a relational connection, you may receive the following error message:

```
400 - Encryption Key Error. Please Contact Administrator:
Unable to access customer encryption key.
```

You are unable to access the relational source.

Solution

The encryption keyfile is missing from the Trifacta® deployment, or the keyfile has been moved without updating the platform of the new location.

You must create and deploy this keyfile, which is required for ensuring that encrypted usernames and passwords are used in relational connections.

> **NOTE:** This keyfile must be created and deployed before any relational connections are created. Deployment requires access to the file system on the Trifacta node.

After you have deployed the keyfile, you must configure the platform to point to its location. A platform restart is not required.

For more information, see *Enable Relational Connections*.

Problem - Retrieving sample data for large relational tables is very slow

In some cases, you may experience slow performance in reading from database tables, or previews of large imported datasets are timing out.

Solution

In these cases, you can experiment with the number of records that are imported per database read. By default, this value is `25000`.

To improve performance, you can modify the following setting.

You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

```
"data-service.sqlOptions.limitedReadStreamRecords": 25000,
```

To improve performance, you can try lowering this limit incrementally. Avoid raising this limit over `100000`, which can overwhelm the browser.

Related articles

- *Backup and Recovery*
- *Install Databases on Amazon RDS*
- *Using Salesforce*
- *Share Connection Window*
- *SQL Server Data Type Conversions*