

alteryx | TRIFACTA

Configuration Guide

Version: 9.7

Doc Build Date: 12/31/2022

Disclaimers

Except as otherwise provided in an express written agreement, Alteryx Inc. (“ Alteryx”) makes no representations or warranties with respect to the software and documentation contained herein and specifically disclaims any implied warranties of merchantability or fitness for any particular purpose. Furthermore, Alteryx reserves the right to revise the software and documentation from time to time without the obligation of Alteryx to notify any person of such revisions or changes.

Copyright and Trademark Notices

© 2013 - 2022 Alteryx Inc. All Rights Reserved.

Alteryx Inc.
17200 Laguna Canyon Road
Irvine, CA 92618
Phone: +1 888 836 4274

Alteryx and Trifacta are registered trademarks of Alteryx Inc.

Credits

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

See also [Third-Party License Information](#).

1. <i>Configure</i>	5
1.1 <i>Platform Configuration Methods</i>	6
1.2 <i>Required Platform Configuration</i>	9
1.2.1 <i>Create Encryption Key File</i>	12
1.2.2 <i>Storage Deployment Options</i>	13
1.2.2.1 <i>Set Base Storage Layer</i>	16
1.2.3 <i>Running Environment Options</i>	19
1.2.3.1 <i>Configure Photon Client</i>	21
1.2.3.2 <i>Configure Photon Running Environment</i>	24
1.2.3.3 <i>Configure Spark Running Environment</i>	30
1.2.4 <i>Profiling Options</i>	32
1.2.5 <i>Configure for Spark</i>	34
1.2.5.1 <i>Enable Spark Job Overrides</i>	54
1.3 <i>Configure for High Availability</i>	58
1.4 <i>Configure for Hadoop</i>	67
1.4.1 <i>Configure for Cloudera</i>	75
1.4.2 <i>Configure Hadoop Authentication</i>	77
1.4.2.1 <i>Configure for Kerberos Integration</i>	79
1.4.2.2 <i>Configure for Secure Impersonation</i>	82
1.4.3 <i>Enable HttpFS</i>	88
1.4.4 <i>Enable Integration with Compressed Clusters</i>	91
1.4.5 <i>Enable Integration with Cluster High Availability</i>	94
1.4.6 <i>Configure for Hive</i>	98
1.4.6.1 <i>Configure for Hive with Sentry</i>	104
1.4.6.2 <i>Hive Connections</i>	108
1.4.7 <i>Configure for KMS</i>	114
1.4.7.1 <i>Configure for KMS for Sentry</i>	115
1.5 <i>Configure for AWS</i>	117
1.5.1 <i>Configure for AWS Authentication</i>	121
1.5.1.1 <i>Configure AWS Per-User Auth for Temporary Credentials</i>	125
1.5.1.2 <i>Configure for AWS SAML Passthrough Authentication</i>	130
1.5.1.3 <i>Configure for EC2 Role-Based Authentication</i>	132
1.5.1.4 <i>Configure for AWS Secrets Manager</i>	134
1.5.2 <i>S3 Access</i>	137
1.5.3 <i>AWS Glue Access</i>	148
1.5.4 <i>Snowflake Access</i>	152
1.5.5 <i>Configure for EMR</i>	155
1.5.6 <i>Configure for AWS Databricks</i>	174
1.6 <i>Configure for Azure</i>	192
1.6.1 <i>Configure Azure Key Vault</i>	196
1.6.2 <i>Configure SSO for Azure AD</i>	200
1.6.2.1 <i>Enable SSO for Azure Relational Connections</i>	205
1.6.3 <i>ADLS Gen2 Access</i>	208
1.6.4 <i>ADLS Gen1 Access</i>	216
1.6.5 <i>WASB Access</i>	224
1.6.6 <i>Configure for Azure Databricks</i>	232
1.7 <i>Configure Security</i>	249
1.8 <i>Configure SSO for AD-LDAP</i>	255
1.9 <i>Configure SSO for SAML</i>	265
1.10 <i>Configuring Platform Users</i>	271
1.10.1 <i>Change Admin Password</i>	272
1.10.2 <i>Enable SMTP Email Server Integration</i>	273
1.10.3 <i>Configure User Self-Registration</i>	276
1.10.4 <i>Enable Self-Service Password Reset</i>	277
1.10.5 <i>Configure User-Specific Props for Cluster Jobs</i>	279
1.10.6 <i>Configure Users and Groups</i>	281
1.11 <i>Configure Features</i>	285
1.11.1 <i>Configure Deployment Manager</i>	288
1.11.2 <i>Configure Scheduling</i>	291
1.11.3 <i>Configure Sharing</i>	293

- 1.11.4 *Configure Webhooks* 295
- 1.11.5 *Enable API Access Tokens* 298
- 1.12 *Configure Services* 300
 - 1.12.1 *Configure Batch Job Runner* 301
 - 1.12.2 *Configure Data Service* 306
 - 1.12.3 *Configure Java VFS Service* 311
 - 1.12.4 *Configure VFS Service* 314
 - 1.12.5 *Configure Connector Configuration Service* 315
 - 1.12.6 *Configure Optimizer Service* 316
 - 1.12.7 *Configure Orchestration Service* 318
 - 1.12.8 *Configure Secure Token Service* 320
 - 1.12.9 *Configure Logging for Services* 321
 - 1.12.10 *Configure Support Bundling* 330
 - 1.12.11 *Configure for Redis* 333
- 1.13 *Miscellaneous Configuration* 336
 - 1.13.1 *Configure Application Limits* 339
 - 1.13.2 *Configure Global File Encoding Type* 348
 - 1.13.3 *Enable User Analytics* 350
- 1.14 *Tune Application Performance* 353

Configure

The following topics describe how to configure Designer Cloud Powered by Trifacta® Enterprise Edition for initial deployment and continued use.

- For more information on administration tasks and admin resources, see *Admin*.

Platform Configuration Methods

Contents:

- *Admin Settings page*
 - *Minimum requirements*
 - *Access*
 - *Limitations*
 - *Workspace Settings Page*
 - *trifacta-conf.json*
 - *Database Configuration*
-

The Designer Cloud powered by Trifacta® platform supports multiple methods of configuration.

Do not make changes through multiple configuration interfaces at the same time. Saved changes can completely overwrite the platform configuration file, removing any unsaved changes from another method.

Admin Settings page

Through the Admin Settings page in the application, platform administrators can perform most configuration tasks.

NOTE: The Admin Settings page is the recommended method for applying configuration changes to the Designer Cloud powered by Trifacta platform. Entries are validated against known types, and the platform is automatically restarted when you save your changes.

Minimum requirements

To use the Admin Settings page, please verify that the following have been completed:

1. Install and initialize the databases. See *Install Databases*.
2. Install or upgrade the Trifacta software on the node. See *Install*.
3. Start the platform. See *.Start and Stop the Platform*.
4. See below for access.

Access

1. Login to the application as an administrator.
2. Select **User menu > Admin console > Admin settings**.
3. Perform your configuration tasks as needed.

In the documentation, JSON configuration items that are available through this page are described using dot notation, as in the following example:

```
"webapp.bodyParser.json.limit": "10mb",
```

Tip: In the Admin Settings page, you can paste the property name without quotes into the search box: `webapp.bodyParser.json.limit`

Limitations

The Admin Settings page is not useful for the following configuration scenarios:

- Changing the configuration when the application is not available.
- You cannot add or delete parameters through this interface.
- Some platform integrations require configuration changes to files other than `trifacta-conf.json`. These changes are described later in the documentation.

In the above scenarios, you should apply your changes through `trifacta-conf.json`.

NOTE: When a change is saved through the Admin Settings page, the entire `trifacta-conf.json` file is overwritten.

For more information, see *Admin Settings Page*.

Workspace Settings Page

In the Workspace Settings page, you can review a set of feature enablement and Designer Cloud application settings that can be modified on a per-workspace basis.

Tip: Over time, more settings from the Admin Settings page and `trifacta-conf.json` will be migrated to this page. For more information, see *Changes to Configuration*.

For more information, see *Workspace Settings Page*.

`trifacta-conf.json`

On the Trifacta node, you can edit the configuration file directly if necessary.

1. Login to the Trifacta node as an administrator.
2. Edit the following file: `/opt/trifacta/conf/trifacta-conf.json`.
3. Make changes as necessary.
4. In the documentation, configuration items may be listed in either JSON dot notation or in hierarchical JSON notation, as in the following:

```
"webapp": {  
  ...  
  "timeoutMilliseconds": 2000,  
  ...  
},
```

NOTE: The ellipses in the configuration indicate that there are unlisted parameters in the actual configuration. Typically, you should search for the final parameter name.

NOTE: In `trifacta-conf.json`, all values in the `spark.props` section or `hadoopConfig` section must be quoted values, as these values are passed to services external to the platform.

5. Save the file and restart the platform. See *Start and Stop the Platform*.

Database Configuration

Where possible, you should use the Admin Settings page for making configuration changes to the databases. See *Admin Settings Page*.

For more information, see *Install Databases*.

Required Platform Configuration

Contents:

- *Configuring by File*
- *Required Platform Configuration Steps*
 - *Change admin password*
 - *Review self-registration*
 - *Configure shared secret*
 - *Other Required Configuration*

This section contains a set of configuration steps required to enable basic functionality in the Designer Cloud powered by Trifacta® platform, as well as the methods by which you can apply the configuration.

Before you begin any configuration or modification to a working configuration, you should back up the `/opt/trifacta/conf` directory.

Admin Settings page: If the software has been installed and the databases have been initialized, you should be able to start the platform and access the Admin Settings page. For more information, see *Platform Configuration Methods*.

Tip: Whenever possible, you should use the Admin Settings page of the application for platform configuration.

Configuration by file: If the application is not available, you can perform configuration changes using the platform files. See below.

Configuring by File

Please make backups of any configuration files that you modify and apply changes with caution.

Tip: If you have not used a Linux text editor, please enter one of the following strings at the command line to see which is available in your environment. nano may be the easiest to use:

- vi
- vim
- emacs
- nano

The Trifacta configuration files are stored in the following directory:

`/opt/trifacta/conf`

Filename	Description
<code>hadoop-site/*</code>	(Hadoop only) Directory for configuration files from the Hadoop cluster to which the platform connects. See <i>Prepare Hadoop for Integration with the Platform</i> .
<code>nginx.conf</code>	Configuration of the platform's HTTP access.

trifacta-conf.json	Most customer-facing configuration and product options for all components are stored here.
--------------------	--

NOTE: After saving your changes to the config files, you must restart the Designer Cloud powered by Trifacta platform to apply them. See *Start and Stop the Platform*.

Required Platform Configuration Steps

Change admin password

As part of the install process, an admin user account is created. For security, this password should be changed.

Review self-registration

By default, any visitor to the Login page can create an account in the Designer Cloud powered by Trifacta platform .

If the Designer Cloud powered by Trifacta platform is available on the public Internet or is otherwise vulnerable to unauthorized access, unauthorized users can register and use the product. If this level of access is unsatisfactory, you should disable self-registration.

Disabling self-registration means that a Trifacta administrator must enable all users. For more information, see *Configure User Self-Registration*.

Configure shared secret

To manage cookie signing, the platform deploys a shared secret, which is used for guaranteeing data transfer between the web client and the platform.

At install time, the platform inserts a default shared secret. The default 64-character shared secret for the platform is the same for all instances of the platform of the same version. This secret should not be used across multiple deployments of the platform.

NOTE: If your instance of the platform is available on the public Internet or if you have deployed multiple instances of the same release of the platform, cookies can become insecure across instances when the secret is shared across instances. You should get in the habit of changing this value for each installation of the platform.

Please complete the following steps to change the shared secret.

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:

```
"sharedSecret": <64_character_value>
```

3. Modify the current value. The new value can be any 64-character string.
4. Save your changes.

Other Required Configuration

The following configuration steps must be reviewed and completed for all deployments of the Designer Cloud powered by Trifacta platform :

NOTE: You must define and configure your backend datastore before the platform is operational.

NOTE: If High Availability is enabled on the cluster, it must be enabled on the Designer Cloud powered by Trifacta platform , even if you are not planning to rely on it. Do this step after completing the preceding steps. See *Enable Integration with Cluster High Availability*.

Create Encryption Key File

The platform utilizes a key file to encrypt and decrypt usernames and passwords for use in connecting to your relational datastores. This keyfile provides an extra layer of security through symmetric encryption.

NOTE: You must create and deploy this keyfile in order to create and use relational connections.

Credentials are encrypted using the AES-128-CBC algorithm.

Requirements for the keyfile:

- This file is a plain text file stored within the Trifacta® platform.
- This file must be deployed before any database connection is created.
- This file must contain a text string that is the key to use.
- The text string can be any string. It should be randomized and not easy to guess.
- After creation, this file cannot be modified.
- This file is shared for all JDBC connections. It does not need to be shared with any database server.

You, the customer, are responsible for the security of this file. It should be secured such that 1) only the root user has read/write access and 2) the Trifacta user has read only access. After the file has been created, it cannot be modified. If it needs to be moved, use the steps below to indicate its new location for the platform.

You must store this file within the Trifacta deployment and reference it through the platform configuration.

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

1. Locate the following configuration. Specify the path to the keyfile relative to the top-level deployment location. Include the filename:

```
"encryption.keyFile": "/opt/trifacta/conf/.key/customerKey",
```

2. Save your changes.

A platform restart is required.

Storage Deployment Options

Contents:

- *Definitions*
 - *HDFS Only*
 - *Hybrid Hadoop-based Deployment*
 - *Amazon-based Deployment*
 - *S3 without Browse or Access*
 - *Microsoft Azure with ADLS Access*
 - *ADLS Gen2*
 - *ADLS Gen1*
 - *Microsoft Azure with WASB Access*
 - *Configuration for Storage Deployments*
-

The Designer Cloud powered by Trifacta® platform can be configured to read and write data from multiple environments at the same time. This page provides information on the supported options.

After you have configured the base storage layer and access and browsing capabilities, you cannot switch them for your Trifacta deployment.

Definitions

Base Storage Layer:

The base storage layer defines where job results are written by default.

NOTE: The base storage layer should be enabled and configured during initial installation. After the base storage layer has been configured, it cannot be switched to another environment.

Tip: The Designer Cloud powered by Trifacta platform can enable connectivity to both S3 and HDFS at the same time. Note that `webapp.storageProtocol=s3` should still be specified to write results to S3.

Access and Browse data - S3:

Optionally, you can enable access and the ability to browse your S3 datastore.

JDBC Sources:

Independent of these storage options, you can access database table data through JDBC datastores. See *Relational Access*.

HDFS Only

Base Storage Layer: HDFS

Access and Browse data - S3: Off

Notes:

The default configuration, this deployment should be used for most on-premise Hadoop environments. In this case, the Designer Cloud powered by Trifacta platform only has access to HDFS and Hive as sources on a single Hadoop cluster.

Hybrid Hadoop-based Deployment

Base Storage Layer: HDFS

Access and Browse data - S3: On

Notes:

This deployment is recommended for the following:

- On-premises Hadoop clusters that require access to remote S3 data
- Hadoop clusters hosted in the cloud that require access to remote S3 data and want to continue to use HDFS as an output location

In this scenario, the Designer Cloud powered by Trifacta platform has access to HDFS and Hive data on the same cluster, as well as access to the remote S3 buckets that have been enabled for the platform.

- HDFS remains the output location for all job results, profiles, and uploads.

Amazon-based Deployment

Base Storage Layer: S3

Access and Browse data - S3: On

Notes:

This deployment is recommended for Hadoop clusters that are completely hosted in AWS and must use S3 as the base storage for all data including job results, profiles, and uploads.

S3 without Browse or Access

NOTE: Before you select your deployment options, you should review additional Amazon information on running Hadoop on S3. For more information, see <https://wiki.apache.org/hadoop/AmazonS3>.

Base Storage Layer: S3

Access and Browse data - S3: Off

Notes:

This configuration is not supported. For more information, please contact *Alteryx Support*.

Microsoft Azure with ADLS Access

ADLS Gen2

Base Storage Layer: ABFSS

Access and Browse data:

- Azure Databricks: Enabled
- ADLS Gen2: Read-write
- WASB: (optional) Read-only

ADLS Gen1

Base Storage Layer: HDFS

Access and Browse data:

- Azure Databricks: Enabled
- ADLS: Read-write
- WASB: (optional) Read-only

Microsoft Azure with WASB Access

Base Storage Layer: WASB

Access and Browse data:

- Azure Databricks: Enabled
- ADLS: (optional) Read-only
- WASB: Read-write

Configuration for Storage Deployments

Base Storage Layer: *Set Base Storage Layer*

Storage Deployments:

- *Configure for Hadoop*
- *S3 Access*
- *WASB Access*
- *ADLS Gen1 Access*

Set Base Storage Layer

Contents:

- *Base Storage Layer Options*
 - *HDFS*
 - *S3*
 - *WASBS*
 - *ADL*
 - *ABFSS*
 - *Base storage layer port options*
 - *Set Storage Layer*
 - *Disable Hadoop Access*
-

In your platform configuration, you must specify the storage platform that is your base storage layer. The **base storage layer** defines the primary storage integration for the Designer Cloud powered by Trifacta® platform . In some cases, integration with other storage layers is supported.

After you define the base storage layer and restart the platform, you cannot change the base storage layer to another option. Please consider your options carefully before you define the base storage layer.

If S3 is the base storage layer, you must also define the default storage bucket to use during initial installation, which cannot be changed at a later time. For additional requirements, see [S3 Access](#).

NOTE: If HDFS is specified as your base storage layer, you cannot publish to Redshift.

Base Storage Layer Options

HDFS

If you are integrating with a Hadoop cluster, you can use HDFS for base storage.

Tip: For Designer Cloud Powered by Trifacta Enterprise Edition, HDFS is the default base storage layer.

S3

If you have installed the product on-premises or on an EC2 instance in AWS, you can set the base storage layer to S3.

Read access to S3 is supported if HDFS is the base storage layer.

For more information, see [S3 Access](#).

Required for:

- Enable write access to S3
- Publish to Redshift

WASBS

If you have installed the product from the Azure Marketplace and are integrating with WASB, you must set to the base storage layer to WASBS.

For more information, see *WASB Access*.

Required for:

- Access to WASB (Azure deployments only)

ADL

Set the base storage layer to `adl` if you are integrating with ADLS Gen1 for read/write access.

NOTE: ADLS Gen1 storage requires an Azure Databricks cluster for execution.

For more information, see *ADLS Gen1 Access*.

Required for:

- Access to ADLS Gen1 (Azure deployments only)

ABFSS

Set the base storage layer to `abfss` if you are integrating with ADLS Gen2.

NOTE: ADLS Gen2 storage requires an Azure Databricks cluster for execution.

For more information, see *ADLS Gen2 Access*.

Required for:

- Access to ADLS Gen2 (Azure deployments only)

For more information on options, see *Storage Deployment Options*.

Base storage layer port options

When you configure your base storage layer, you must also define the port number to use for access.

NOTE: If you change the port number of the base storage layer in the future, all results from previous jobs are lost. Please choose the port number with care.

Set Storage Layer

When you have decided on the final base storage layer, set the following property to one of the above values in platform configuration.

The platform requires that one backend datastore be configured as the base storage layer. This base storage layer is used for storing uploaded data and writing results and profiles. Please complete the following steps to set the base storage layer for the Designer Cloud powered by Trifacta platform .

You cannot change the base storage layer after it has been set. You must uninstall and reinstall the platform to change it.

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 the value for your base storage layer:

```
"webapp.storageProtocol": "hdfs",
```

3. Save your changes and restart the platform.

NOTE: To complete the integration with the base storage layer, additional configuration is required.

Disable Hadoop Access

If you are not using Hadoop at all, please complete the following configuration change.

Steps:

1. Login to the Trifacta node.
2. Edit the following files:

```
site-config-*-s3.json  
site-config.installer-*-edge.json
```

3. In these files, set the following property value to `hostname`:

```
"hdfs.namenode.host": "hostname",
```

4. Save the files and restart the platform.

Running Environment Options

Contents:

- *Available Running Environments*
- *Configure*
 - *Configure Running Environments*
 - *Configure Default Running Environment*

The Designer Cloud powered by Trifacta® platform can be configured to integrate with a variety of environments for processing transformation jobs. When you run a job through the application, you have the option of selecting the running environment on which you wish to run the job.

Tip: In general, you should accept the default environment that is presented for job execution. The application attempts to match the scope of your job to the most appropriate running environment.

This section applies to execution of transform jobs. For more information on options for profiling jobs, see *Profiling Options*.

Available Running Environments

For more information, see *Overview of Job Execution*.

Configure

Configure Running Environments

To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

The following parameters define the available running environments:

```
"webapp.runWithSparkSubmit": true,  
"webapp.runinEMR": false,  
"webapp.runInDatabricks": true,  
"webapp.runInDataflow": false,
```

For more information on configuring the running environment for EMR, see *Configure for EMR*.

Below, you can see the configuration settings required to enable each running environment.

- The Spark running environment requires a Hadoop cluster as the backend job execution environment.
 - In the Run Job page, select **Spark**.
- The Trifacta Photon running environment executes on the Trifacta node and provide processing to the front-end client and at execution time.
 - In the Run Job page, select **Photon**.
 - For more information on disabling the Trifacta Photon running environment, see *Configure Photon Running Environment*.

Type	Running Environment	Configuration Parameters	Notes
------	---------------------	--------------------------	-------

Hadoop Backend	Spark	<code>webapp.runWithSparkSubmit = true</code>	The Spark running environment is the default configuration.
Client Front-end and non-Hadoop Backend	Trifacta Photon	In the Workspace Settings page, set Photon Execution to Enabled.	Trifacta Photon is the default running environment for the front-end of the application. It is enabled by default. For more information, see <i>Workspace Settings Page</i> .

NOTE: Do not modify the `runInDataflow` setting.

Configure Default Running Environment

When you specify a job, the default running environment is pre-configured for you, based on the following parameter:

NOTE: If your environment has no running environment such as Spark for running large-scale jobs, this parameter is not used. All jobs are run on the Trifacta node.

```
"webapp.client.maxExecutionBytes.photon": 1000000000,
```

The default environment presented to you is based on the size of the primary datasource. For the above setting of 1 GB:

Running Environment	Default Condition
Trifacta Photon	Size of primary datasource is less than or equal to the above value in bytes.
Spark	Size of primary datasource is greater than the above value in bytes.

NOTE: This setting defines only the environment that is recommended to you as a predefined selection. If a second running environment is available, you can choose to select it, although it is not recommended to choose an environment other than the default. See *Run Job Page*.

Setting this value too high forces more jobs onto the Trifacta Photon running environment, which may cause slow performance and can potentially overwhelm the server.

Tip: To force the default setting to always be a Hadoop or bulk running environment, set this value to 0. All users are recommended to use the bulk option instead of the Trifacta Photon running environment. However, smaller jobs may take longer than expected to execute.

Configure Photon Client

Contents:

- *Limitations*
- *Recommended Photon Configuration by Core Count*
- *Modify Limits*
 - *Sample Size Limits*
 - *Maximum Data in the Client*
- *Use Photon Client*
- *Configure VFS Service*

Trifacta Photon client is the in-browser engine for management of sampling and transformation in the Transformer page. This section describes configuration options for Trifacta Photon.

Trifacta Photon is automatically downloaded and updated in each user's web browser when accessing the Transformer page. Within the browser, Photon is used to manage sampling and to process transformations on those samples on the local web client, which limits required interactions with the Trifacta node.

Features:

- Larger sample sizes (up to 10MB by default).

NOTE: Since Trifacta Photon supports larger sample sizes, some interactions may be slightly impacted. Loading states have been introduced to enable faster responsiveness from the application.

Tip: For datasets that are smaller than the sample size limit, the Transformer Page displays the entire dataset in its transformed state. So, you can download the dataset from the Recipe Panel in Transformer Page without having to execute it on a remote server. This expanded capability allows for faster and more immediate local use of the product. See *Recipe Panel*.

This section contains the user-facing configuration for the Trifacta Photon client. Except as noted below, these configuration changes are applied to the Trifacta node, which then applies the configuration to each instance of the Trifacta Photon client and its interactions with the node.

NOTE: Some configuration is shared with the Trifacta Photon running environment. For more information, see *Configure Photon Running Environment*.

Limitations

- None

Recommended Photon Configuration by Core Count

On the Trifacta node, you can make adjustments to the resources claimed by the Photon running environment based on the number of cores on the machine. The following table identifies the recommended settings for a node with 8, 16, or 32 cores. The default settings assume 16 cores.

Parameter	8 cores	16 cores (default)	32 cores
-----------	---------	--------------------	----------

webapp.numProcesses	2	2	5
vfs-service.numProcesses	2	2	3
photon.numThreads	2	4	4
batchserver.workers.photon.max	2	2	4

The number of simultaneous users is a competing factor.

- For a high number, more resources should be reserved for the webapp and the VFS services.
- For a low number, more resources for Photon should improve performance for sampling and job execution on the Trifacta Photon running environment.

The following table illustrates some adjustments for a 16-core system:

Parameter	16 cores (default)	Low number of simultaneous users	High number of simultaneous users
webapp.numProcesses	2	1	4
vfs-service.numProcesses	2	1	4
photon.numThreads	4	4	4
batchserver.workers.photon.max	2	2	2

Modify Limits

NOTE: Increasing these values can have a significant impact on load times and performance. Change these values only if you are experiencing difficulties. Make incremental changes.

Sample Size Limits

You can configure the default and maximum permitted size of samples downloaded to the browser. For more information, see *Configure Application Limits*.

Maximum Data in the Client

The following parameter sets the upper limit on the number of bytes that can be delivered to the browser, which includes samples, metadata, and other data. Adding some kinds of transformations, such as joins or adding new columns, can grow the volume of data in the browser. This parameter represents the upper volume of data that can be displayed for a sample + transformations.

```
"webapp.client.maxResultsBytes": 41943040,
```

Tip: By default, the value for this parameter is four times the value of permitted sample sizes. If a user changes the permitted maximum size of samples delivered to the browser, then the total limit of maximum data in the client is set to 4x of the maximum permitted sample size.

NOTE: If the data volume of the same grows beyond this size, the number of rows available in the sample is reduced until the sample volume is below this threshold. This truncation happens automatically and without warning.

Use Photon Client

The Trifacta Photon client is an embedded component of the web client. Access to it is transparent to the user.

Configure VFS Service

Photon interacts with backend datastores through the VFS service.

NOTE: The VFS service is enabled by default and must be enabled when Trifacta Photon running environment is in use.

For more information, see *Configure VFS Service*.

Configure Photon Running Environment

Contents:

- *Limitations*
- *Disable Trifacta Photon Running Environment*
- *Example Configuration*
- *Modify Limits*
 - *Runtime job timeout*
 - *Trifacta Photon running environment memory timeout*
 - *Batch FileSystem Access Timeout Settings*
 - *Dynamic chunk size for Parquet reads*
 - *Tuning Photon*
- *Configure VFS Service*
- *Use Trifacta Photon Running Environment*

The Designer Cloud® application can connect to a high-performance environment embedded in the Trifacta node for execution of jobs against small- to medium-sized datasets, called the **Trifacta Photon running environment**.

- The Trifacta Photon running environment can be selected in the Run Job page.

By default, the Trifacta Photon running environment is enabled for new installations.

Features:

- Faster execution times for transform and profiling jobs
- Better consistency with typecasting done in Spark jobs

This section provides information on how to enable and configure the Trifacta Photon running environment.

NOTE: Some configuration is shared with the Trifacta Photon client. For more information, see *Configure Photon Client*.

Limitations

NOTE: For profiles executed in the Trifacta Photon running environment, percentages for valid, missing, or mismatched column values may not add up to 100% due to rounding. See *Overview of Visual Profiling*.

Disable Trifacta Photon Running Environment

The Trifacta Photon running environment is enabled by default. Please complete the following configuration to disable the running environment.

NOTE: A cluster-based running environment, such as Spark, must be available for processing jobs when this one is disabled.

NOTE: When Trifacta Photon is disabled, quick scan sampling is not available.

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. To disable the Trifacta Photon running environment, locate the following setting, and set it to `Disabled`:

```
Photon execution
```

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

```
"feature.enableSamplingScanOptions": false,  
"feature.enableFirstRowsSample": false,
```

5. Save your changes and restart the platform.

Example Configuration

The following configuration includes the default values.

```
"photon": {  
  "cacheEnabled": true,  
  "numThreads": 4,  
  "distroPath": "/photon/dist/centos6/photon",  
  "traceExecution": false,  
  "websocket": {  
    "host": "localhost",  
    "port": 8082  
  },  
  "mode": "wasm"  
},
```

Some of these values apply to the Trifacta Photon client. For more information, see *Configure Photon Client*.

Parameter	Description
<code>cacheEnabled</code>	Debugging setting. Leave the default value.
<code>numThreads</code>	Maximum number of threads permitted to the Trifacta Photon process. For recommended values, see <i>Configure Photon Client</i> .
<code>distroPath</code>	Please verify that this property is set to the following value, which works for all operating system distributions: <pre>"distroPath": "/photon/dist/centos6/photon",</pre>
<code>traceExecution</code>	Debugging setting. Leave the default value.
<code>websocket.host</code>	Internal parameter. Do not modify.

websocket . port	Internal parameter. Do not modify.
mode	Set this value is wasm.

Modify Limits

Runtime job timeout

By default, the Designer Cloud powered by Trifacta platform imposes no limit on execution of a Trifacta Photon job. As needed, you can enable and configure a limit.

Steps:

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

```
"batchserver.workers.photon.timeoutEnabled": false,
"batchserver.workers.photon.timeoutMinutes": 180,
```

Setting	Description
timeoutEnabled	Set to <code>false</code> to disable job limiting. Set to <code>true</code> to enable the timeout specified below.
timeoutMinutes	Defines the number of minutes that a Trifacta Photon job is permitted to run. Default value is 180 (three hours).

2. Save your changes and restart the platform.

When a job has failed due to exceeding a timeout, additional information is available in the job logs. The following is a good search term for this type of error:

```
java.lang.Exception: Photon job '<jobId>' timeout
```

where `<jobId>` is the internal job identifier.

Job logs can be downloaded from the Job page. See *Job History Page*.

Trifacta Photon running environment memory timeout

To prevent crashes, the Trifacta Photon running environment imposes a memory consumption limit for each job. If this memory timeout is exceeded, the job is automatically killed. As needed, you can disable this memory protection (not recommended) or change the memory threshold when jobs are killed.

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 setting:

```
"batchserver.workers.photon.memoryMonitorEnabled": false,
```

Setting	Description
memoryMonitorEnabled	Set to <code>false</code> to disable memory monitoring. Set to <code>true</code> to enable the threshold specified below.

3. Save your changes and restart the platform.

Additional information is available in the job logs. The following is a good search term for this type of error:

```
java.lang.Exception: Photon job '<jobId>' failed with memory consumption over threshold
```

where <jobId> is the internal job identifier.

Below this line item, you may see the following entries, which can provide additional information to adjust the memory settings:

```
2017-05-04T02:26:40.549Z [job-id 740] com.trifacta.joblaunch.util.ProcessMonitorUtil [Thread-20] INFO com.trifacta.joblaunch.util.ProcessMonitorUtil - Global memory size: 8373186560 bytes
2017-05-04T02:26:40.555Z [job-id 740] com.trifacta.joblaunch.util.ProcessMonitorUtil [Thread-20] INFO com.trifacta.joblaunch.util.ProcessMonitorUtil - Available global memory size at process start: 2672959488 bytes
...
2017-05-04T02:29:15.690Z [job-id 740] com.trifacta.joblaunch.util.ProcessMonitorUtil [Thread-20] INFO com.trifacta.joblaunch.util.ProcessMonitorUtil - Current memory consumption: 5.614080429077148%
2017-05-04T02:29:15.691Z [job-id 740] com.trifacta.joblaunch.util.ProcessMonitorUtil [Thread-20] ERROR com.trifacta.joblaunch.util.ProcessMonitorUtil - Average memory consumption for the past 15 seconds over 5% threshold: 5.174326801300049 %. Current available global memory: 2244628480 bytes
```

Item	Description
Global memory size	Total available global memory in bytes
Available global memory size at process start	Total available memory in bytes when the job is launched
Current memory consumption	Current memory usage for the job process as a percentage of the total. This metric is posted to the log every 30 seconds and can be used to debug memory leaks.
Average memory consumption for the past 15 seconds over x% threshold	When the job fails due to the memory threshold, this metric identifies the average memory consumption percentage over the past 15 seconds. The defined threshold percentage is included.
Current available global memory	When the job fails, this metric identifies the total available memory at the time of failure.

Job logs can be downloaded from the Job page. See [Job History Page](#).

Batch FileSystem Access Timeout Settings

The default timeout settings for reading and writing of data from the client browser through Trifacta Photon running environment to the Trifacta node should work in most cases.

Particularly when reading from large tables, you might discover errors similar to the following:

```
06:21:21.365 [Job 23] INFO com.trifacta.hadoopdata.photon.BatchPhotonRunner - terminating with uncaught exception of type Poco::TimeoutException: Timeout
06:21:21.375 [Job 23] INFO com.trifacta.hadoopdata.photon.BatchPhotonRunner - /vagrant/photons/dist/centos6/photons/bin/photons-cli: line 22: 15639 Aborted $
Unknown macro: {command[@]}
```

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 `photons.extraCliArgs` node.
3. Add the following values to the `extraCliArgs` entry:

```
"photon.extraCliArgs" : "-batch_vfs_read_timeout <300> -batch_vfs_write_timeout <300>"
```

Argument	Description
-batch_vfs_read_timeout	Timeout limit in seconds of read operations from the datastore. Default value is 300 seconds (5 minutes). <div style="border: 1px solid green; padding: 5px; margin-top: 10px;">Tip: Raising the value to 3600 seconds should be fine in most environments. Avoid setting this value above 7200 seconds (2 hours).</div>
-batch_vfs_write_timeout	Timeout limit in seconds of write operations to the datastore. Default value is 300 seconds (5 minutes). <div style="border: 1px solid blue; padding: 5px; margin-top: 10px;">NOTE: Do not modify unless specifically instructed by <i>Alteryx Support</i>.</div>

4. To reduce timeouts, raise the above settings.
5. Save your changes and restart the platform.

Dynamic chunk size for Parquet reads

In some environments, quick scan sampling and data preview of Parquet files may have performance issues. To improve performance in these areas, you can enable reading of chunks of dynamic size from Parquet. When enabled, the Designer Cloud application adjusts the size of each read chunk based on the Row Group Size of the Parquet file. These chunk sizes are calculated for each file and can vary between 1 MB and 20 MB.

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`:

```
"photon.useDynamicChunkSizeForParquet": true,
```

3. Save your changes and restart the platform.

Tuning Photon

For more information on tuning the performance of Photon, see *Tune Application Performance*.

Configure VFS Service

The Trifacta Photon running environment interacts with backend datastores through the VFS service.

NOTE: The VFS service does not often need non-default configuration.

For more information, see *Configure VFS Service*.

Use Trifacta Photon Running Environment

When executing a job, select the **Photon** option.

NOTE: Before you test, please be sure to complete all steps of *Required Platform Configuration*.

Configure Spark Running Environment

Contents:

- *Enable Spark Execution Environment*
- *Use Spark Execution Environment*
- *Change Limits*
 - *Change Spark settings per job*
 - *Enable user-specific overrides*

This section provides information on how to enable and configure the Spark running environment, which leverages Spark's faster in-memory processing to deliver better execution performance.

Limitations

NOTE: When a recipe containing a user-defined function is applied to text data, any non-printing (control) characters cause records to be truncated by the Spark running environment during job execution. In these cases, please execute the job on the Trifacta® Photon running environment.

- You cannot publish through Cloudera Navigator for Spark jobs.

Enable Spark Execution Environment

The Spark execution environment is enabled by default.

NOTE: If you have not done so already, please enable and configure the Spark Job Service. See *Configure for Spark*.

Use Spark Execution Environment

When Spark execution is enabled, it is available like any other execution environment in the application. When executing a job, select the **Spark** option from the drop-down in the Run Job page. See *Run Job Page*.

Change Limits

For more information on changing limits and other tuning parameters, see *Configure for Spark*.

Change Spark settings per job

You can enable a set of Spark properties that users are permitted to override on individual jobs. For more information, see *Enable Spark Job Overrides*.

Enable user-specific overrides

You can also enable user-specific overrides for Spark jobs executed on the cluster.

NOTE: The user-specific overrides take precedence over the Spark settings applied to the output objects.

For more information, see *Configure User-Specific Props for Cluster Jobs*.

Profiling Options

Contents:

- [Overview](#)
- [Configure](#)
 - [Run Profiling as a Second Job in Spark](#)
- [Disable Profiling Option](#)

A **visual profile** can be optionally generated as part of execution of a job. While profiling can extend the time it takes to complete the job, a visual profile can provide key statistical information on the columns of your data and the overall dataset itself. This section provide information on each of the available profilers and how to configure them.

NOTE: To enable use of a profiler, the running environment on which it is hosted must also be enabled. See [Running Environment Options](#).

The Designer Cloud powered by Trifacta® platform supports the following options for profiling your data.

Overview

Type of Profiler	Requires Hadoop cluster?	Supported Running Environment	Description
Scala Spark Profiler	Yes	all	Generates a profile on the Trifacta Photon running environment using the Spark Job Service. Does not require any Spark installation on the cluster. <div style="border: 1px solid green; padding: 5px; text-align: center;">Tip: This is the best-performing profiler in most use cases.</div> See Configure for Spark .
Trifacta Photon Profiler	No	Trifacta Photon	Default profiler when Trifacta Photon running environment is enabled, and the Scala Spark Profiler has not been enabled.

Configure

Making changes to your profiling type:

1. Apply the configuration changes listed below.
2. Save your changes and restart the platform.
3. Restart your browser and login again.

Run Profiling as a Second Job in Spark

By default, profiling jobs execute on the running environment where the transformation job was executed. You can configure the Trifacta Photon running environment to execute profiling jobs as a separate job in the Spark running environment. This separation allows users to begin working with the transformed data while waiting for the profiling job to complete.

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 property and set its value to `true`:

```
"photon.runProfileWithSpark": false,
```

3. Save your changes and restart the platform.

When profiling is selected for jobs executing on the Trifacta Photon running environment, their jobs are executed on the Spark running environment as a separate job.

Disable Profiling Option

Profiling is invoked at job execution time by the user. See *Run Job Page*.

To disable user choice through the UI, set the following parameter:

```
"profiler.userOption": false,
```

When the above setting is disabled:

- Any available checkbox no longer works. User cannot choose whether to profile or not.
- Profiles are always executed.

NOTE: This setting does not affect profiling through the APIs. Profiling can always be enabled or disabled for jobs that are executed via API.

Configure for Spark

Contents:

- *Supported Versions*
- *Prerequisites*
- *Configure the Designer Cloud powered by Trifacta platform*
 - *Configure Spark Job Service*
 - *Configure Spark*
 - *Specify YARN queue for Spark jobs*
 - *Spark tuning properties*
- *Configure Spark Version*
 - *Set Spark version*
 - *Acquire native libraries from the cluster*
 - *Use native libraries on Cloudera 6.0 and later*
 - *Configure Spark for Cloudera Data Platform*
 - *Modify Spark version and Java JDK version*
 - *Combine transform and profiling in Spark jobs*
- *Additional Configuration*
- *Restart Platform*
- *Verify Operations*
- *Logs*
- *Troubleshooting*
 - Problem - Spark job succeeds on the cluster but is reported as failed in the application. Spark Job Service is constantly dying.*
 - Problem - Spark Job Service fails to start with a "Exception in thread "main" com.fasterxml.jackson.databind.JsonMappingException: Jackson version is too old 2.5.3" error.*
 - Problem - Spark jobs fail with "Unknown message type: -22" error*
 - Problem - Spark jobs fail when Spark Authentication is enabled on the Hadoop cluster*
 - Problem - Job fails with "Required executor memory MB is above the max threshold MB of this cluster" error*
 - Problem - Job fails with ask timed out error*
 - Problem - Spark fails with ClassNotFoundException error in Spark job service log*
 - Problem - Spark fails with PermGen OutOfMemory error in the Spark job service log*
 - Problem - Spark fails with "token (HDFS_DELEGATION_TOKEN token) can't be found in cache" error in the Spark job service log on a Kerberized cluster when Impersonation is enabled*
 - Problem - Spark fails with "job aborted due to stage failure" error*
 - Problem - Spark job fails with "Error while instantiating 'org.apache.spark.sql.hive.HiveSessionState'" error*
 - Problem - Spark job fails with "No Avro files found. Hadoop option "avro.mapred.ignore.inputs.without.extension" is set to true. Do all input files have ".avro" extension?" error*
 - Problem - Spark job fails in the platform but successfully executes on Spark*

The Designer Cloud powered by Trifacta® platform can be configured to work with Spark to execute job results, a visual profile of transform job results, or both.

- A **visual profile** is a visual summary of a dataset. It visually identifies areas of interest, including valid, missing, or mismatched values, as well as useful statistics on column data.
 - Visual profiles can be optionally generated using Spark.
 - In the Designer Cloud application, visual profiles appear in the Job Details page when a job has successfully executed and a profile has been requested for it. See *Job Details Page*.
 - For more information, see *Overview of Visual Profiling*.
- **Apache Spark** provides in-memory processing capabilities for a Hadoop cluster. In Spark, the processing of the large volume of computations to generate this information is performed in-memory. This method reduces disk access and significantly improves overall performance. For more information, see <https://spark.apache.org/>.

The Spark Job Service is a Scala-based capability for executing jobs and profiling your job results as an extension of job execution. This feature leverages the computing power of your existing Hadoop cluster to increase job execution and profiling performance. Features:

- Requires no additional installation on the Trifacta node.
- Support for yarn-cluster mode ensures that all Spark processing is handled on the Hadoop cluster.
- Exact bin counts appear for profile results, except for Top-N counts.

Supported Versions

The following versions of Spark are supported:

NOTE: Depending on the version of Spark and your Hadoop distribution, additional configuration may be required. See [Configure Spark Version](#) below.

- Spark 3.2.0, Spark 3.2.1

NOTE: Spark 3.2.x is supported only on specific deployments and versions of the following environments:

Azure Databricks 10.x

AWS Databricks 10.x

- Spark 3.0.1

NOTE: Spark 3.0.1 is supported only on specific deployments and versions of the following environments:

AWS Databricks 7.3 LTS (**Recommended**)

EMR 6.2.1, EMR 6.3

- Spark 2.4.6 **Recommended**
- Spark 2.3.x

Prerequisites

NOTE: Spark History Server is not supported. It should be used only for short-term debugging tasks, as it requires considerable resources.

Before you begin, please verify the following:

1. For additional prerequisites for a kerberized environment, see [Configure for Kerberos Integration](#).
2. Additional configuration is required for secure impersonation. See [Configure for Secure Impersonation](#).

Configure the Designer Cloud powered by Trifacta platform

Configure Spark Job Service

The Spark Job Service must be enabled for both execution and profiling jobs to work in Spark.

Below is a sample configuration and description of each property. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

```
"spark-job-service" : {
  "systemProperties" : {
    "java.net.preferIPv4Stack": "true",
    "SPARK_YARN_MODE": "true"
  },
  "sparkImpersonationOn": false,
  "optimizeLocalization": true,
  "mainClass": "com.trifacta.jobserver.SparkJobServer",
  "jvmOptions": [
    "-Xmx128m"
  ],
  "hiveDependenciesLocation": "%(topOfTree)s/hadoop-deps/cdh-6.2/build/libs",
  "env": {
    "SPARK_JOB_SERVICE_PORT": "4007",
    "SPARK_DIST_CLASSPATH": "",
    "MAPR_TICKETFILE_LOCATION": "<MAPR_TICKETFILE_LOCATION>",
    "MAPR_IMPERSONATION_ENABLED": "0",
    "HADOOP_USER_NAME": "trifacta",
    "HADOOP_CONF_DIR": "%(topOfTree)s/conf/hadoop-site/"
  },
  "enabled": true,
  "enableHiveSupport": true,
  "enableHistoryServer": false,
  "classpath": "%(topOfTree)s/services/spark-job-server/server/build/install/server/lib/*:%(topOfTree)s/conf/hadoop-site/%(topOfTree)s/services/spark-job-server/build/bundle/*:%(topOfTree)s/%(hadoopBundleJar)s",
  "autoRestart": false,
}
```

The following properties can be modified based on your needs:

NOTE: Unless explicitly told to do so, do not modify any of the above properties that are not listed below.

Property	Description
sparkImpersonationOn	Set this value to <code>true</code> , if secure impersonation is enabled on your cluster. See <i>Configure for Secure Impersonation</i> .
jvmOptions	This array of values can be used to pass parameters to the JVM that manages Spark Job Service.
hiveDependenciesLocation	If Spark is integrated with a Hive instance, set this value to the path to the location where Hive dependencies are installed on the Trifacta node. For more information, see <i>Configure for Hive</i> .
env.SPARK_JOB_SERVICE_PORT	Set this value to the listening port number on the cluster for Spark. Default value is 4007. For more information, see <i>System Ports</i> .
env.HADOOP_USER_NAME	The username of the Hadoop principal used by the platform. By default, this value is <code>trifacta</code> .

env. HADOOP_CONF_DIR	The directory on the Trifacta node where the Hadoop cluster configuration files are stored. Do not modify unless necessary.
enabled	Set this value to <code>true</code> to enable the Spark Job Service.
enableHiveSupport	See below.

After making any changes, save the file and restart the platform. See *Start and Stop the Platform*.

Configure service for Hive

Depending on the environment, please apply the following configuration changes to manage Spark interactions with Hive:

Environment	spark.enableHiveSupport
Hive is not present	<code>false</code>
Hive is present but not enabled.	<code>false</code>
Hive is present and enabled	<code>true</code>

If Hive is present on the cluster and either enabled or disabled: the `hive-site.xml` file must be copied to the correct directory:

```
cp /etc/hive/conf/hive-site.xml /opt/trifacta/conf/hadoop-site/hive-site.xml
```

At this point, the platform only expects that a `hive-site.xml` file has been installed on the Trifacta node . A valid connection is not required. For more information, see *Configure for Hive* .

Configure Spark

After the Spark Job Service has been enabled, please complete the following sections to configure it for the Designer Cloud powered by Trifacta platform .

Yarn cluster mode

All jobs submitted to the Spark Job Service are executed in YARN cluster mode. No other cluster mode is supported for the Spark Job Service.

Configure access for secure impersonation

The Spark Job Service can run under secure impersonation. For more information, see *Configure for Secure Impersonation*.

When running under secure impersonation, the Spark Job Service requires access to the following folders. Read, write, and execute access must be provided to the Trifacta user and the impersonated user.

Folder Name	Platform Configuration Property	Default Value	Description
Trifacta Libraries folder	" <code>hdfs.pathsConfig.libraries</code> "	<code>/trifacta/libraries</code>	Maintains JAR files and other libraries required by Spark. No sensitive information is written to this location.
Trifacta Temp files folder	" <code>hdfs.pathsConfig.tempFiles</code> "	<code>/trifacta/tempfiles</code>	Holds temporary progress information files for YARN applications. Each file contains a number indicating the progress percentage. No sensitive information is written to this location.

Trifacta Dictionaries folder	"hdfs.pathsConfig.dictionaries"	/trifacta/dictionaries	Contains definitions of dictionaries created for the platform.
------------------------------	---------------------------------	------------------------	--

Identify Hadoop libraries on the cluster

The Spark Job Service does not require additional installation on the Trifacta node or on the Hadoop cluster. Instead, it references the spark-assembly JAR that is provided with the Trifacta distribution.

This JAR file does not include the Hadoop client libraries. You must point the Designer Cloud powered by Trifacta platform to the appropriate libraries.

Steps:

1. In platform configuration, locate the `spark-job-service` configuration block.
2. Set the following property:

```
"spark-job-service.env.HADOOP_CONF_DIR": "<path_to_Hadoop_conf_dir_on_Hadoop_cluster>",
```

Property	Description
<code>spark-job-service.env.HADOOP_CONF_DIR</code>	Path to the Hadoop configuration directory on the Hadoop cluster.

3. In the same block, the `SPARK_DIST_CLASSPATH` property must be set depending on your Hadoop distribution.
4. Save your changes.

Locate Hive dependencies location

If the Designer Cloud powered by Trifacta platform is also connected to a Hive instance, please verify the location of the Hive dependencies on the Trifacta node. The following example is from Cloudera 6.2:

NOTE: This parameter value is distribution-specific. Please update based on your Hadoop distribution.

```
"spark-job-service.hiveDependenciesLocation": "%(topOfTree)s/hadoop-deps/cdh-6.2/build/libs",
```

Specify YARN queue for Spark jobs

Through the Admin Settings page, you can specify the YARN queue to which to submit your Spark jobs. All Spark jobs from the Designer Cloud powered by Trifacta platform are submitted to this queue.

Steps:

1. In platform configuration, locate the following:

```
"spark.props.spark.yarn.queue": "default",
```

2. Replace `default` with the name of the queue.
3. Save your changes.

Spark tuning properties

In addition to the specific Spark properties that are exposed in platform configuration, you can pass properties and their values to the Spark running environment, which are interpreted and applied during the execution of your jobs. In platform configuration, these properties and values are passed in through the `spark.props` area, where you specify the property name and value in JSON format.

For example, you can pass additional properties to Spark such as number of cores, executors, and memory allocation. Some examples are below.

NOTE: The following values are default values. If you are experiencing performance issues, you can modify the values. If you require further assistance, please contact *Alteryx Support*.

- In Admin Settings:

```
"spark.props.spark.driver.maxResultSize": "0",
```

- In `trifacta-conf.json`:

```
"spark": {  
  ...  
  "props": {  
    "spark.executor.memory": "6GB",  
    "spark.executor.cores": "2",  
    "spark.driver.memory": "2GB"  
    ...  
  }  
},
```

If you have sufficient cluster resources, you should pass the following values:

- In Admin Settings:

```
"spark.props.spark.driver.maxResultSize": "0",
```

- In `trifacta-conf.json`:

```
"spark": {  
  ...  
  "props": {  
    "spark.executor.memory": "16GB",  
    "spark.executor.cores": "5",  
    "spark.driver.memory": "16GB"  
    ...  
  }  
},
```

Notes:

- The above values must be below the per-container thresholds set by YARN. Please verify your settings against the following parameters in `yarn-site.xml`:

```
yarn.scheduler.maximum-allocation-mb  
yarn.scheduler.maximum-allocation-vcores  
yarn.nodemanager.resource.memory-mb  
yarn.nodemanager.resource.cpu-vcores
```

- If you are using YARN queues, please verify that these values are set below max queue thresholds.
- For more information on these properties, see <https://spark.apache.org/docs/2.2.0/configuration.html>.

Save your changes.

Configure Batch Job Runner for Spark service

You can modify the following Batch Job Runner configuration settings for the Spark service.

NOTE: Avoid modifying these settings unless you are experiencing issues with the user interface reporting jobs as having failed while the Spark job continues to execute on YARN.

Setting	Description	Default
batchserver.spark.requestTimeoutMillis	Specifies the number of milliseconds that the Batch Job Runner service should wait for a response from Spark. If this timeout is exceeded, the UI changes the job status to failed. The YARN job may continue.	600000 (600 seconds)

Configure Spark Version

Set Spark version

Review and set the following parameter.

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. Verify that the Spark master property is set accordingly:

```
"spark.master": "yarn",
```

3. Review and set the following parameter based on your Hadoop distribution:

NOTE: This setting is ignored for EMR, Azure Databricks and AWS Databricks, which always use the vendor libraries.

Hadoop Distribution	Parameter Value	Value is required?
Cloudera Data Platform 7.1	"spark.useVendorSparkLibraries" : true,	Yes. Additional configuration is required.
CDH 6.x	"spark.useVendorSparkLibraries" : true,	Yes. Additional configuration is in the next section.

4. Locate the following setting:

```
"spark.version"
```

5. Set the above value based on your Hadoop distribution in use:

Hadoop Distribution	spark.version	Notes
	3.0.1	NOTE: This version of Spark is available for selection through the Designer Cloud application . It is supported for a limited number of running environments. Additional information is provided later.
Cloudera Data Platform 7.1	2.4.cdh6.3.3.plus	NOTE: Please set the Spark version to the value indicated. This special value accounts for unexpected changes to filenames in the CDH packages.
CDH 6.3.3	2.4.cdh6.3.3.plus	NOTE: Please set the Spark version to the value indicated. This special value accounts for unexpected changes to filenames in the CDH packages.

Acquire native libraries from the cluster

NOTE: If the Trifacta node is installed on an edge node of the cluster, you may skip this section.

You must acquire native Hadoop libraries from the cluster if you are using any of the following versions:

Hadoop version	Library location on cluster	Trifacta node location
Cloudera Data Platform 7.1	/opt/cloudera/parcels/CDH-7.1.1-1.cdh7.1.1.* The last directory name may vary between minor distributions.	See section below.
Cloudera 6.0 or later	/opt/cloudera/parcels/CDH	See section below.

NOTE: Whenever the Hadoop distribution is upgraded on the cluster, the new versions of these libraries must be recopied to the following locations on the Trifacta node. This maintenance tasks is not required in the Trifacta node is an edge node of the cluster.

For more information on acquiring these libraries, please see the documentation provided with your Hadoop distribution.

Use native libraries on Cloudera 6.0 and later

To integrate with CDH 6.x, the platform must use the native Spark libraries. Please add the following properties to your configuration.

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. Set `sparkBundleJar` to the following:
 - a. For Cloudera 6.x:

```
"sparkBundleJar": "/opt/cloudera/parcels/CDH/lib/spark/jars/*:/opt/cloudera/parcels/CDH/lib/spark/hive/*"
```

- b. For Cloudera Data Platform, see "Configure Spark for Cloudera Data Platform."

3. For the Spark Job Service, the Spark bundle JAR must be added to the classpath:

NOTE: The key modification is to remove the `topOfTree` element from the `sparkBundleJar` entry.

- a. For Cloudera 6.x:

```
"spark-job-service": {  
  "classpath": "%(topOfTree)s/services/spark-job-server/server/build/install/server/lib/*:%  
(topOfTree)s/conf/hadoop-site/:/usr/lib/hdinsight-datalake/*:%(sparkBundleJar)s:%(topOfTree)s/%  
(hadoopBundleJar)s"  
},
```

- b. For Cloudera Data Platform, see "Configure Spark for Cloudera Data Platform."

4. In the `spark.props` section, add the following property:

- a. For Cloudera 6.x:

```
"spark.yarn.jars": "local:/opt/cloudera/parcels/CDH/lib/spark/jars/*,local:/opt/cloudera/parcels  
/CDH/lib/spark/hive/*",
```

- b. For Cloudera Data Platform, see "Configure Spark for Cloudera Data Platform."

5. Save your changes.

Configure Spark for Cloudera Data Platform

The Designer Cloud powered by Trifacta platform can integrate with Spark on Cloudera Data Platform (CDP) to run jobs on ACID tables in the following deployment:

- CDP Private Cloud 7.1.5
- Spark 2.4.x
- Hive Warehouse Connector

NOTE: Spark Direct Reader mode only is supported for Hive Warehouse Connector.

Steps:

Please complete the following additional steps to enable this integration.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. CDP 7.1.x requires the use of Spark 2.4.x, which requires a special Scala version. Please review and change if needed the following settings:

```
"spark.version": "2.4.cdh6.3.3.plus",  
"spark.scalaVersion": "2.11",  
"spark.useVendorSparkLibraries": true,  
"sparkBundleJar": "/opt/cloudera/parcels/CDH/lib/spark/jars/*:/opt/cloudera/parcels/CDH/lib/spark/hive  
/*",
```

3. Please insert the following settings in the `spark.props` area of configuration:

```

"spark": {
  ...
  "props": {
    "spark.yarn.jars": "local:/opt/cloudera/parcels/CDH/lib/spark/jars/*,local:/opt/cloudera/parcels/CDH/lib/spark/hive/*,local:/opt/cloudera/parcels/CDH/jars/hive-warehouse-connector-assembly-1.0.0.7.1.4.0-203.jar",
    "spark.sql.extensions": "com.qubole.spark.hiveacid.HiveAcidAutoConvertExtension",
    "spark.datasource.hive.warehouse.read.via.llap": "false",
    "spark.sql.hive.hwc.execution.mode": "spark",
    "spark.hadoop.hive.metastore.uris": "thrift://example.com:9083",
    "spark.kryo.registrator": "com.qubole.spark.hiveacid.util.HiveAcidKryoRegistrator",
    "spark.sql.hive.hiveserver2.jdbc.url": "jdbc:hive2://example.com:2181/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2"
  }
},

```

NOTE: All paths listed in the above properties must be verified against the CDP environment.

Property	Description
"spark.yarn.jars"	For the <code>hive-warehouse-connector-assembly</code> JAR file, the path and version information depends on your installation. Please use a value that matches your environment.
"spark.hadoop.hive.metastore.uris"	This value can be obtained in the following file: <code>/etc/hive/conf/hive-site.xml</code> .
"spark.sql.hive.hiveserver2.jdbc.url"	This value can be obtained in the following file: <code>"/etc/hive/conf.cloudera.hive_on_tez/beeline-site.xml"</code> .

4. Update the classpath value for the Spark job service:

```

"spark-job-service": {
  "classpath": "%(topOfTree)s/services/spark-job-server/server/build/install/server/lib/*:%(topOfTree)s/conf/hadoop-site/:(sparkBundleJar)s:%(topOfTree)s/(hadoopBundleJar)s",
},

```

5. Save your changes and restart the platform.

Modify Spark version and Java JDK version

NOTE: The Spark version settings in this section do not apply to Databricks, which has a dedicated Spark version property: `databricks.sparkVersion`.

- For more information, see *Configure for AWS Databricks*.
- For more information, see *Configure for Azure Databricks*.

The Designer Cloud powered by Trifacta platform defaults to using Spark 2.3.0. Depending on the version of your Hadoop distribution, you may need to modify the version of Spark that is used by the platform.

In the following table, you can review the Spark/Java version requirements for the Trifacta node hosting Designer Cloud Powered by Trifacta Enterprise Edition.

To change the version of Spark in use by the Designer Cloud powered by Trifacta platform, you change the value of the `spark.version` property, as listed below. No additional installation is required.

NOTE: If you are integrating with an EMR cluster, the version of Spark to configure for use depends on the version of EMR. Additional configuration is required. See *Configure for EMR*.

NOTE: The value for `spark.version` does not need to be set for Databricks. The version of Spark for Databricks is controlled by a different setting.

Additional requirements:

- The supported cluster must use Java JDK 8 or 11.
- If the platform is connected to an EMR cluster, you must set the local version of Spark (`spark.version` property) to match the version of Spark that is used on the EMR cluster.

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

Spark 2.3.0

Required Java JDK Versions	Java JDK 8 or 11
Spark for Designer Cloud Powered by Trifacta Enterprise Edition	<pre>"spark.version": "2.3.0",</pre>

Spark 2.4.6

Required Java JDK Version	Java JDK 8 or 11
Spark for Designer Cloud Powered by Trifacta Enterprise Edition	<pre>"spark.version": "2.4.6",</pre> <p>NOTE: For CDH 6.3.3, please set the <code>spark.version</code> property value to the following: <code>2.4cdh6.3.3.plus</code>.</p> <p>NOTE: For Spark 2.4.0 and later, please verify that the following is set:</p> <pre>"spark.useVendorSparkLibraries": true,</pre> <p>This configuration is ignored for EMR and Azure Databricks.</p>

Spark 3.0.1

Required Java JDK Version	Java JDK 8 or 11
----------------------------------	------------------

Spark for Designer Cloud Powered by Trifacta Enterprise Edition

```
"spark.version": "3.0.1",
```

NOTE: Support for this version of Spark is limited. Additional configuration is required. See below.

Additional configuration requirements for this version:

- For this version of Spark, please set the following:

```
"spark.scalaVersion": "2.12",
```

- For Spark 2.4.0 and later, please verify that the following is set:

```
"spark.useVendorSparkLibraries": true,
```

This setting is ignored for EMR and Azure Databricks.

- Additional configuration is required. For more information:
 - [Configure for Azure Databricks](#)
 - [Configure for AWS Databricks](#)
 - [Configure for EMR](#)

Spark 3.2.0, Spark 3.2.1

Required Java JDK Version

Java JDK 8 or 11

Spark for Designer Cloud Powered by Trifacta Enterprise Edition

```
"spark.version": "3.2.0",
```

or

```
"spark.version": "3.2.1",
```

NOTE: Support for this version of Spark is limited. Additional configuration is required. See below.

Additional configuration requirements for this version:

- For this version of Spark, please set the following:

```
"spark.scalaVersion": "2.12",
```

- For Spark 2.4.0 and later, please verify that the following is set:

```
"spark.useVendorSparkLibraries": true,
```

This setting is ignored for EMR and Azure Databricks.

- Additional configuration is required. For more information:
 - [Configure for Azure Databricks](#)
 - [Configure for AWS Databricks](#)

Combine transform and profiling in Spark jobs

When profiling is enabled for a job, the Spark running environment executes the transformation and profiling tasks of these jobs together by default. This combination of jobs is faster and more efficient than separating the jobs. As needed, these tasks can be separated. The following behaviors vary depending on whether these tasks are separated or combined:

When transform and profiling are combined:

- If the transform task succeeds and profiling fails, then the job is shown as failed in the Job Details page. The generated datasets may be available for download.
- Since the job failed, no publishing actions are launched.

When transform and profiling are separated:

- If the transform task succeeds and profiling fails, then the the generated datasets may be available for download.
- When the transform task succeeds, any defined publishing actions are launched.

For more information on configuring these options, see *Workspace Settings Page*.

For more information, see *Job Details Page*.

Additional Configuration

- For more information on executing jobs on Spark, see *Configure Spark Running Environment*.
- For more information on visual profiling, see *Overview of Visual Profiling*.

Restart Platform

You can restart the platform now. See *Start and Stop the Platform*.

Verify Operations

At this point, you should be able to run a job in the platform, which launches a Spark execution job and a profiling. Results appear normally in the Designer Cloud application .

Steps:

To verify that the Spark running environment is working:

1. After you have applied changes to your configuration, you must restart services. See *Start and Stop the Platform*.
2. Through the application, run a simple job, including visual profiling. Be sure to select **Spark** as the running environment.
3. The job should appear as normal in the Job Status page.
4. To verify that it ran on Spark, open the following file:
`/opt/trifacta/logs/batch-job-runnnner.log`
5. Search the log file for a `SPARK JOB INFO` block with a timestamp corresponding to your job execution.
6. See below for information on how to check the job-specific logs.
7. Review any errors.

For more information, see *Verify Operations*.

Logs

Service logs:

Logs for the Spark Job Service are located in the following location:

```
/opt/trifacta/logs/spark-job-service.log
```

Additional log information on the launching of profile jobs is located here:

```
/opt/trifacta/logs/batch-job-runner.log
```

Job logs:

When profiling jobs fail, additional log information is written to the following:

```
/opt/trifacta/logs/jobs/<job-id>/spark-job.log
```

Troubleshooting

Below is a list of common errors in the log files and their likely causes.

Problem - Spark job succeeds on the cluster but is reported as failed in the application. Spark Job Service is constantly dying.

Whenever a Spark job is executed, it is reported back as having failed. On the cluster, the job appears to have succeeded. However, in the Spark Job Service logs, the Spark Job Service cannot find any of the applications that it has submitted to resource manager.

In this case, the root problem is that Spark is unable to delete temporary files after the job has completed execution. During job execution, a set of ephemeral files may be written to the designated temporary directory on the cluster, which is typically `/trifacta/tempfiles`. In most cases, these files are removed transparent to the user.

- This location is defined in the `hdfs.pathsConfig.tempFiles` parameter in `trifacta-conf.json`.

In some cases, those files may be left behind. To account for this accumulation in the directory, the Designer Cloud powered by Trifacta platform performs a periodic cleanup operation to remove temp files that are over a specified age.

- The age in days is defined in the `job.tempfiles.cleanup.age` parameter in `trifacta-conf.json`.

This cleanup operation can fail if HDFS is configured to send Trash to an encrypted zone. The HDFS API does not support the `skipTrash` option, which is available through the HDFS CLI. In this scenario, the temp files are not successfully removed, and the files continue to accumulate without limit in the temporary directory. Eventually, this accumulation of files can cause the Spark Job Service to crash with Out of Memory errors.

Solution

The following are possible solutions:

1. Solution 1: Configure HDFS to use an unencrypted zone for Trash files.
2. Solution 2:
 - a. Disable temp file cleanup in `trifacta-conf.json`:

```
"job.tempfiles.cleanup.age": 0,
```

- b. Clean up the `tempfiles` directory using an external process.

Problem - Spark Job Service fails to start with a "Exception in thread "main" com.fasterxml.jackson.databind.JsonMappingException: Jackson version is too old 2.5.3" error.

Spark job service fails to start with an error similar to the following in the `spark-job-service.log` file:

```
Exception in thread "main" com.fasterxml.jackson.databind.JsonMappingException: Jackson version is too old 2.5.3
```

Solution

Some versions of the `hadoopBundleJar` contain older versions of the Jackson dependencies, which break the `spark-job-service`.

To ensure that the `spark-job-service` is provided the correct Jackson dependency versions, the `sparkBundleJar` must be listed before the `hadoopBundleJar` in the `spark-job-service.classpath`, which is inserted as a parameter in `trifacta-conf.json`. Example:

```
"spark-job-service.classpath": "%(topOfTree)s/services/spark-job-server/server/build/install/server/lib/*:%(topOfTree)s/conf/hadoop-site/:%(topOfTree)s/%(sparkBundleJar)s:%(topOfTree)s/%(hadoopBundleJar)s"
```

Problem - Spark jobs fail with "Unknown message type: -22" error

Spark jobs may fail with the following error in the YARN application logs:

```
ExecutorLostFailure (executor 4 exited caused by one of the running tasks) Reason: Unable to create executor due to Unable to register with external shuffle server due to : java.lang.IllegalArgumentException: Unknown message type: -22  
at org.apache.spark.network.shuffle.protocol.BlockTransferMessage$Decoder.fromByteBuffer(BlockTransferMessage.java:67)
```

Solution

This problem may occur if Spark authentication is disabled on the Hadoop cluster but enabled in the Designer Cloud powered by Trifacta platform . Spark authentication must match on the cluster and the platform.

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 `spark.props` entry.
3. Insert the following property and value:

```
"spark.authenticate": "false"
```

4. Save your changes and restart the platform.

Problem - Spark jobs fail when Spark Authentication is enabled on the Hadoop cluster

When Spark authentication is enabled on the Hadoop cluster, Spark jobs can fail. The YARN log file message looks something like the following:

```
17/09/22 16:55:42 INFO yarn.ApplicationMaster: Unregistering ApplicationMaster with FAILED (diag message:
User class threw exception: org.apache.spark.SparkException: Job aborted due to stage failure: Task 0 in
stage 0.0 failed 4 times, most recent failure: Lost task 0.3 in stage 0.0 (TID 3, example.com, executor 4):
ExecutorLostFailure (executor 4 exited caused by one of the running tasks) Reason: Unable to create executor
due to Unable to register with external shuffle server due to : java.lang.IllegalStateException: Expected
SaslMessage, received something else (maybe your client does not have SASL enabled?)
at org.apache.spark.network.sasl.SaslMessage.decode(SaslMessage.java:69)
```

Solution

When Spark authentication is enabled on the Hadoop cluster, the Designer Cloud powered by Trifacta platform must also be configured with Spark authentication enabled.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Inside the `spark.props` entry, insert the following property value:

```
"spark.authenticate": "true"
```

3. Save your changes and restart the platform.

Problem - Job fails with "Required executor memory MB is above the max threshold MB of this cluster" error

When executing a job through Spark, the job may fail with the following error in the `spark-job-service.log`:

```
Required executor memory (6144+614 MB) is above the max threshold (1615 MB) of this cluster! Please check the
values of 'yarn.scheduler.maximum-allocation-mb' and/or 'yarn.nodemanager.resource.memory-mb'.
```

Solution

The per-container memory allocation in Spark (`spark.executor.memory` and `spark.driver.memory`) must not exceed the YARN thresholds. See *Spark tuning properties* above.

Problem - Job fails with ask timed out error

When executing a job through Spark, the job may fail with the following error in the `spark-job-service.log` file:

```
Job submission failed
akka.pattern.AskTimeoutException: Ask timed out on [Actor[akka://SparkJobServer/user
/ProfileLauncher#1213485950]] after [20000 ms]
```

There is a 20-second timeout on the attempt to submit a Profiling job to Yarn. If the initial upload of the spark libraries to the cluster takes longer than 20 seconds, the `spark-job-service` times out and returns an error to the UI. However, the libraries do finish uploading successfully to the cluster.

The library upload is a one-time operation for each install/upgrade. Despite the error, the libraries are uploaded successfully the first time. This error does not affect subsequent profiler job runs.

Solution:

Try running the job again.

Problem - Spark fails with ClassNotFoundException error in Spark job service log

When executing a job through Spark, the job may fail with the following error in the `spark-job-service.log` file:

```
java.lang.ClassNotFoundException: com.trifacta.jobserver.profiler.Profiler
```

By default, the Spark job service attempts to optimize the distribution of the Spark JAR files across the cluster. This optimization involves a one-time upload of the `spark-assembly` and `profiler-bundle` JAR files to HDFS. Then, YARN distributes these JARs to the worker nodes of the cluster, where they are cached for future use.

In some cases, the localized JAR files can get corrupted on the worker nodes, causing this `ClassNotFoundException` error to occur.

Solution:

The solution is to disable this optimization through platform configuration.

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 `spark-job-service` configuration node.
3. Set the following property to `false`:

```
"spark-job-service.optimizeLocalization" : true
```

4. Save your changes and restart the platform.

Problem - Spark fails with PermGen OutOfMemory error in the Spark job service log

When executing a job through Spark, the job may fail with the following error in the `spark-job-service.log` file:

```
Exception in thread "LeaseRenewer:trifacta@nameservice1" java.lang.OutOfMemoryError: PermGen space
```

Solution:

The solution is to configure the PermGen space for the Spark driver:

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` configuration node.
3. Set the following property to the given value:

```
"spark.props.spark.driver.extraJavaOptions" : "-XX:MaxPermSize=1024m -XX:PermSize=256m",
```

4. Save your changes and restart the platform.

Problem - Spark fails with "token (HDFS_DELEGATION_TOKEN token) can't be found in cache" error in the Spark job service log on a Kerberized cluster when Impersonation is enabled

When executing a job through Spark, the job may fail with the following error in the `spark-job-service.log` file:

```
org.apache.hadoop.ipc.RemoteException(org.apache.hadoop.security.token.SecretManager$InvalidToken): token (HDFS_DELEGATION_TOKEN token x for trifacta) can't be found in cache
```

Solution:

The solution is to set Spark impersonation to `true`:

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-job-service` configuration node.
3. Set the following property to the given value:

```
"spark-job-service.sparkImpersonationOn" : true,
```

4. Save your changes and restart the platform.

Problem - Spark fails with "job aborted due to stage failure" error

Issue:

Spark fails with an error similar to the following in the `spark-job-service.log`:

```
"Job aborted due to stage failure: Total size of serialized results of 208 tasks (1025.4 MB) is bigger than spark.driver.maxResultSize (1024.0 MB)"
```

Explanation:

The `spark.driver.maxResultSize` parameter determines the limit of the total size of serialized results of all partitions for each Spark action (e.g. `collect`). If the total size of the serialized results exceeds this limit, the job is aborted.

To enable serialized results of unlimited size, set this parameter to zero (0).

Solution:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. To the `spark.props` section of the file, remove the size limit by setting this value to zero:

```
"spark.driver.maxResultSize": "0"
```

3. Save your changes and restart the platform.

Problem - Spark job fails with "Error while instantiating 'org.apache.spark.sql.hive.HiveSessionState'" error

Issue:

Spark job fails with an error similar to the following in either the `spark-job.log` or the `yarn-app.log` file:

```
"java.lang.IllegalArgumentException: Error while instantiating 'org.apache.spark.sql.hive.HiveSessionState'"
```

Explanation:

By default, the Spark running environment attempts to connect to Hive when it creates the Spark Context. This connection attempt may fail if Hive connectivity (in `conf/hadoop-site/hive-site.xml`) is not configured correctly on the Trifacta node.

Solution:

This issue can be fixed by configuring Hive connectivity on the edge node.

If Hive connectivity is not required, the Spark running environment's default behavior can be changed as follows:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. In the `spark-job-service` section of the file, disable Hive connectivity by setting this value to `false`:

```
"spark-job-service.enableHiveSupport": false
```

3. Save your changes and restart the platform.

Problem - Spark job fails with "No Avro files found. Hadoop option "avro.mapred.ignore.inputs.without.extension" is set to true. Do all input files have ".avro" extension?" error

Issue:

Spark job fails with an error similar to the following in either the `spark-job.log` or the `yarn-app.log` file:

```
java.io.FileNotFoundException: No Avro files found. Hadoop option "avro.mapred.ignore.inputs.without.extension" is set to true. Do all input files have ".avro" extension?
```

Explanation:

By default, Spark-Avro requires all Avro files to have the `.avro` extension, which includes all part files in a source directory. Spark-Avro ignores any files that do not have the `.avro` extension.

If a directory contains part files without an extension (e.g. `part-00001`, `part-00002`), Spark-Avro ignores these files and throws the "No Avro files found" error.

Solution:

This issue can be fixed by setting the `spark.hadoop.avro.mapred.ignore.inputs.without.extension` property to `false`:

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. To the `spark.props` section of the file, add the following setting if it does not already exist. Set its value to `false`:

```
"spark.hadoop.avro.mapred.ignore.inputs.without.extension": "false"
```

3. Save your changes and restart the platform.

Problem - Spark job fails in the platform but successfully executes on Spark

Issue:

After you have submitted a job to be executed on the Spark cluster, the job may fail in the Designer Cloud powered by Trifacta platform after 30 minutes. However, on the busy cluster, the job remains enqueued and is eventually collected and executed. Since the job was canceled in the platform, results are not returned.

Explanation:

This issue is caused by a timeout setting for Batch Job Runner, which cancels management of jobs after a predefined number of seconds. Since these jobs are already queued on the cluster, they may be executed independent of the platform.

Solution:

This issue can be fixed by increasing the Batch Job Runner Spark timeout setting:

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 property. By default, it is set to 172800, which is 48 hours:

```
"batchserver.spark.progressTimeoutSeconds": 172800,
```

3. If your value is lower than the default, you can increase this value high enough for your job to succeed.
4. Save your changes and restart the platform.
5. Re-run the job.

Enable Spark Job Overrides

Contents:

- *Limitations*
 - *Default Overrides*
 - *Enable*
 - *Configure Available Parameters to Override*
 - *Apply Overrides*
 - *For scheduled jobs*
 - *Via API*
-

By default, the Designer Cloud Powered by Trifacta® Enterprise Edition applies configuration to Spark at the global level. All jobs submitted to the connected instance of Spark utilize the same set of Spark properties and settings. As needed, the set of properties can be modified by administrators through the Admin console.

Optionally, flow owners can configure overrides to the default Spark properties at the output object level.

- Overrides are defined as part of the output object. When you configure these values, they are applied each time that the output object is used to generate a set of results.
- These overrides can be applied to ad-hoc or scheduled jobs.

User-specific Spark overrides: If you have enabled user-specific overrides for Spark jobs, those settings take precedence over the settings that are applied through this feature. For more information, see *Configure User-Specific Props for Cluster Jobs*.

Limitations

This feature allows administrators to enable the passthrough of properties to Spark, and users can submit any value of an enabled property. Please be careful in choosing the properties that you enable for users to override.

Property validation:

- You can enable properties that are destructive.
- Property names of whitelisted properties:
 - No validation of property names is provided by the Designer Cloud application .
 - No check is performed on property names. Invalid property names are ignored.
- Property values:
 - Property values are not compared against the operating environment. For example, a user can input 1000 for the number of `spark.executor.cores` , which causes job failures.
 - Property values are validated against expected type.

Default Overrides

When this feature is enabled, the following properties are available for users to override at job execution time with their preferred values.

NOTE: These properties are always available for override when the feature is enabled.

Spark parameter	Description
spark.driver.memory	Amount of RAM in GB on each Spark node that is made available for the Spark drivers.
spark.executor.memory	Amount of RAM in GB on each Spark node that is made available for the Spark executors.
spark.executor.cores	Number of cores on each Spark executor that is made available to Spark.
transformer.dataframe.checkpoint.threshold	<p>When checkpointing is enabled, the Spark DAG is checkpointed when the approximate number of expressions in this parameter has been added to the DAG. Checkpointing assists in managing the volume of work that is processed through Spark at one time; by checkpointing after a set of steps, the Designer Cloud powered by Trifacta platform can reduce the chances of execution errors for your jobs.</p> <p>By raising this number:</p> <ul style="list-style-type: none"> You increase the upper limit of steps between checkpoints. You may reduce processing time. It may result in a higher number of job failures.

Spark jobs on Azure Databricks:

For Spark jobs executed on Azure Databricks, only the following default override parameters are supported:

Spark parameter	Description
transformer.dataframe.checkpoint.threshold	See above.

During Spark job execution on Azure Databricks:

Whenever overrides are applied to an Azure Databricks cluster, the overrides must be applied at the time of cluster creation. As a result, a new Azure Databricks cluster is spun up for the job execution, which may cause the following:

- **Delay in job execution as the cluster is spun up.**
 - **Increased usage and costs**
 - **After a new Azure Databricks cluster has been created using updated Spark properties for the job, any existing clusters complete executing any in-progress jobs and gracefully terminate based on the idle timeout setting for Azure Databricks clusters.**
- You cannot apply overrides to the Spark dynamic allocation settings on Azure Databricks, including the following parameters. At job execution time, these parameters are discarded, even if they have been added to the whitelisted properties:
 - spark.dynamicAllocation.enabled
 - spark.shuffle.service.enabled
 - spark.executor.instances

For more details on setting these parameters, see *Tune Cluster Performance*.

Enable

Workspace administrators can enable Spark job overrides.

Steps:

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

```
Enable Custom Spark Options Feature
```

4. Set this parameter to `Enabled`.

Configure Available Parameters to Override

After enabling the feature, workspace administrators can define the Spark properties that are available for override.

Steps:

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

```
Spark Whitelist Properties
```

4. Enter a comma-separated list of Spark properties. For example, the entry for adding the following two properties looks like the following:

```
spark.driver.extraJavaOptions,spark.executor.extraJavaOptions
```

5. Save your changes.
6. After saving, please reload the page for the feature to be enabled.
7. When users configure ad-hoc or scheduled Spark jobs, the above properties are now available for overriding, in addition to the default override properties.

Apply Overrides

Overrides are applied to output objects associated with a flow.

Run Job page

When you configure an on-demand job to run:

1. Select the output object in the flow. Then, click **Run**.
2. In the Run Job page, select **Spark** for the Running Environment.
3. Click the Advanced environment options caret.
4. The Spark Execution Properties are available for override.

NOTE: No validation of the property values is performed against possible values or the connected running environment.

For more information, see *Spark Execution Properties Settings*.

For scheduled jobs

When you are scheduling a job:

1. Select the output object in the flow. In the context panel, select the Destinations tab.
2. Under Scheduled destinations, click **Add** to add a new destination or **Edit** to modify an existing one.
3. In the Scheduled Publishing settings page, select **Spark** for the Running Environment.
4. Click the Advanced environment options caret.
5. The Spark Execution Properties are available for override.

NOTE: No validation of the property values is performed against possible values or the connected running environment.

For more information, see *Spark Execution Properties Settings*.

Via API

You can submit Spark property overrides as part of the request body for an output object. See *API Task - Manage Outputs*.

Configure for High Availability

Contents:

- *Limitations*
- *Log File Setup*
- *Platform Configuration*
 - *Disable restart*
 - *Disable job cancellation*
- *NFS Mount Setup*
- *Example NFS Mount*
 - *Server-side Configuration*
 - *Client-side Configuration*
 - *Mount NFS Shared Directories on Client*
 - *Verifying NFS Shared Directories On Clients*
 - *Auto-mount NFS Shared Directories*
- *Integrate with HAProxy Load Balancer*
- *Platform Start*
- *Additional Configuration*
 - *Enable Redis Sentinel*
- *Troubleshooting*
 - *When mounting the shared directories on the client, a Connection timeout error occurs*

This section contains additional configuration tasks for setting up high availability on the Trifacta® node.

- The Designer Cloud powered by Trifacta platform can also integrate with a highly available Hadoop cluster. For more information, see *Enable Integration with Cluster High Availability*.

Limitations

For more information on deploying high availability, see *Install for High Availability*.

Log File Setup

Since the platform nodes are mounting to the same directory, a configuration flag forces log files to be put into a subdirectory containing the hostname of the node from which the file was generated. The resulting directory structure:

```
/opt/trifacta/  
--- conf  
--- bin  
--- logs/  
    --- node1/webapp.log  
    --- node1/data-service.log  
    --- node1/spark-job-service.log  
    --- node1/...  
    --- node2/webapp.log  
    --- node2/data-service.log  
    --- node2/spark-job-service.log  
    --- node2/...  
    ---- .....
```

- Job logs are already prefixed with the job-group id, so there is no need to prefix them with the hostname.
- The Download Logs should work independent of the node serving the endpoint, so there is no hostname prefixing required for the job logs.

To enable separation of log files into separate directories by node, please complete the following:

Steps:

1. Login as admin to the server (main) node in your HA environment.
2. To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Set the following parameter to `true`:

```
"feature.highAvailability.separateLogFilesByName": false,
```

4. Save the file.
5. Before restarting the server, please complete any remaining configuration changes.

Platform Configuration

Disable restart

The preferred method of restarting the platform is to use the Restart button in the Admin Settings page. When a Trifacta node is part of a high availability environment, the node cannot be restarted through the Admin Settings page, which only restarts the node. Other nodes in the environment are not restarted, and any configuration changes are not propagated.

NOTE: In an HA environment, each node must be started, stopped, and restarted through the command line. See *Start and Stop the Platform*.

To disable use of the Restart button in the Admin Settings page, please complete the following steps.

Steps:

1. Login as admin to the server (main) node in your HA environment.
2. To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Set the following parameter to `false`:

```
"feature.highAvailability.enableRestartFromAdminUI": true,
```

4. Save the file.
5. Before restarting the server, please complete any remaining configuration changes.

Disable job cancellation

In a high-availability environment, job cancellation is not supported. To disable, please complete the following steps.

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. Verify that the following parameter is set to `false`:

```
"feature.highAvailability.jobCancellation": false,
```

3. Before restarting the server, please complete any remaining configuration changes.

NFS Mount Setup

Each node shares the following resources:

- Trifacta databases
- Directories shared via NFS mount:

```
/opt/trifacta/logs  
/opt/trifacta/conf
```

Example NFS Mount

The following sections contain an example configuration for the NFS mount to be shared across all nodes.

NOTE: There are many ways to configure an NFS mount. The following example applies to CentOS 7. Please modify these commands for different operating systems and versions as needed.

Server-side Configuration

NFS service

Use the following command to install the required NFS packages on the server:

```
yum install nfs-utils nfs-utils-lib
```

Enable and start NFS services:

```
systemctl enable rpcbind  
systemctl enable nfs-server  
systemctl enable nfs-lock  
systemctl enable nfs-idmap  
  
systemctl start rpcbind  
systemctl start nfs-server  
systemctl start nfs-lock  
systemctl start nfs-idmap
```

Shared directories

The following directories shared across all nodes:

```
/opt/trifacta/logs  
/opt/trifacta/conf
```

1. To export the shared directories from the server, edit the following file:

```
/etc/exports
```

2. Append the following lines to the end of the file:

```
/opt/trifacta/conf 192.168.1.102(rw,sync,no_root_squash,no_all_squash)  
/opt/trifacta/logs 192.168.1.102(rw,sync,no_root_squash,no_all_squash)
```

Property	Description
/opt/trifacta /conf /opt/trifacta /logs	Directories to share across NFS
192.168.1.102	IP address for client. <div style="border: 1px solid black; padding: 5px; text-align: center;">NOTE: Insert the above commands once for each client IP address.</div>
rw	Read/write permissions on the shared folder.
sync	Enables synchronizing shared directories.
no_root_squash	Enables root privilege.
no_all_squash	Enables user's authority.

3. Restart the nfs-server service:

```
systemctl restart nfs-server
```

Client-side Configuration

On each client node, install NFS packages:

```
yum install nfs-utils nfs-utils-lib
```

Enable and start the NFS services:

```
systemctl enable rpcbind
systemctl enable nfs-server
systemctl enable nfs-lock
systemctl enable nfs-idmap

systemctl start rpcbind
systemctl start nfs-server
systemctl start nfs-lock
systemctl start nfs-idmap
```

Mount NFS Shared Directories on Client

Mount the shared directories from the server to the client as shown below:

```
mount -t nfs 192.168.1.101:/opt/trifacta/logs /opt/trifacta/logs
mount -t nfs 192.168.1.101:/opt/trifacta/conf /opt/trifacta/conf
```

NOTE: Repeat the above on each client node.

If you encounter a Connection timed out error, please see the Troubleshooting section below.

Verifying NFS Shared Directories On Clients

Verify that the shared directories from the server are mounted:

```
mount | grep nfs
```

Example output:

```
rpc_pipefs on /var/lib/nfs/rpc_pipefs type rpc_pipefs (rw,relatime)
nfsd on /proc/fs/nfsd type nfsd (rw,relatime)
192.168.1.101:/opt/trifacta/logs on /opt/trifacta/logs type nfs4 (rw,relatime,vers=4.1,rsize=1048576,
wsize=1048576,namlen=255,hard,proto=tcp,port=0,timeo=600,retrans=2,sec=sys,clientaddr=10.0.9.167,
local_lock=none,addr=10.0.6.77)
```

Auto-mount NFS Shared Directories

The above manual steps must be repeated whenever the platform is rebooted.

To automatically mount the shared directories on reboot:

1. Edit the following file:

```
/etc/fstab
```

2. Add the following lines:

```
192.168.1.101:/opt/trifacta/logs /opt/trifacta/logs nfs rw,sync,hard,intr 0 0
192.168.1.101:/opt/trifacta/conf /opt/trifacta/conf nfs rw,sync,hard,intr 0 0
```

3. Reboot the client system and check the whether the NFS shared directories were mounted or not:

```
mount | grep nfs
```

Integrate with HAProxy Load Balancer

After you have installed the software on each node in your high availability environment and configured your NFS mount, please complete the following instructions to integrate each node with a load balancer.

NOTE: HAProxy is the recommended load balancer. These instructions apply to HAProxy. If you are using a different load balancer, please consult the documentation that came with your product.

Configuration example:

```
#-----
# Example configuration for a possible web application. See the
# full configuration options online.
#
# http://haproxy.1wt.eu/download/1.4/doc/configuration.txt
#
#-----
# Global settings
#-----
global
```

```

# to have these messages end up in /var/log/haproxy.log you will
# need to:
#
# 1) configure syslog to accept network log events.  This is done
#    by adding the '-r' option to the SYSLOGD_OPTIONS in
#    /etc/sysconfig/syslog
#
# 2) configure local2 events to go to the /var/log/haproxy.log
#    file. A line like the following can be added to
#    /etc/sysconfig/syslog
#
#    local2.*                /var/log/haproxy.log
#
log        127.0.0.1 local1 notice

chroot    /var/lib/haproxy
pidfile   /var/run/haproxy.pid
maxconn   4000
user      haproxy
group     haproxy
daemon

# turn on stats unix socket
stats socket /var/lib/haproxy/stats

#-----
# common defaults that all the 'listen' and 'backend' sections will
# use if not designated in their block
#-----
defaults
    mode                http
    log                 global
    option              httplog
    option              dontlognull
    option http-server-close
    option forwardfor   except 127.0.0.0/8
    option              redispatch
    option              log-health-checks
    retries             3
    timeout http-request 10s
    timeout queue       1m
    timeout connect     10s
    timeout client      1m
    timeout server      1m
    timeout http-keep-alive 10s
    timeout check       1s
    maxconn             3000

#-----
# main frontend which proxys to the backends
#-----
frontend main *:8088
#    acl url_static     path_beg       -i /static /images /javascript /stylesheets
#    acl url_static     path_end       -i .jpg .gif .png .css .js
    mode http
#    use_backend static      if url_static
    default_backend     app

#-----
# static backend for serving up images, stylesheets and such
#-----
#backend static
#    balance roundrobin
#    server      static 127.0.0.1:4331 check

#-----
# round robin balancing between the various backends
#-----
backend app
    balance roundrobin
    option httpchk GET /v2/ping

```

```
server app1 192.168.1.101:3005 check
server app2 192.168.1.102:3005 check
```

Key Settings:

Setting	Description
balance	Set this value to the method of load balancing. Round robin is recommended.
option httpchk	Please set the value to the following: <pre>GET /v2/ping</pre>
server app1	Set this value to the IP address of the server node.
server app2 server app3	Please insert separate entries for the IP addresses of each client node.

Platform Start

To restart all nodes of the platform, please complete the following steps:

NOTE: When the platform is restarted in a high availability environment, all scheduled jobs that are in-progress or have been triggered during downtime may be assigned to a single node. To distribute across multiple nodes, restart all services except for the time-based service on each node. After all other services have restarted, start the time-based trigger service on each node.

1. Apply any configuration changes to `trifacta-conf.json` on the server. Since the configuration directory is now shared across all nodes, these changes are automatically applied to all client nodes.
2. Verify that all nodes are pointing to the same instances of the Trifacta databases.
3. Start the master node.

NOTE: Nodes in an HA environment must be restarted at the command line. See *Start and Stop the Platform*.

4. Wait for it to complete restart.
5. Start each client node one at a time.

NOTE: Do not start multiple nodes at the same time, which can cause migration errors.

6. When all nodes are restarted, you may continue with any necessary configuration steps through the (main) server node.

Additional Configuration

Enable Redis Sentinel

You can configure the Trifacta node to connect to a high availability Redis deployment using Redis Sentinel. For more information, see *Configure for Redis*.

Troubleshooting

When mounting the shared directories on the client, a Connection timeout error occurs

If you encounter the error:

```
mount.nfs: Connection timed out
```

This error may be due to the firewall blocking the NFS server.

Solution:

1. To access NFS shared directories from remote clients, the following ports in the NFS server iptables / firewall must be allowed:

```
rpcinfo -p
Sample output:

  program vers proto  port  service
  100000   4   tcp   111   portmapper
  100000   3   tcp   111   portmapper
  100000   2   tcp   111   portmapper
  100000   4   udp   111   portmapper
  100000   3   udp   111   portmapper
  100000   2   udp   111   portmapper
  100024   1   udp   60985 status
  100024   1   tcp   54302 status
  100005   1   udp   20048 mountd
  100005   1   tcp   20048 mountd
  100005   2   udp   20048 mountd
  100005   2   tcp   20048 mountd
  100005   3   udp   20048 mountd
  100005   3   tcp   20048 mountd
  100003   3   tcp   2049  nfs
  100003   4   tcp   2049  nfs
  100227   3   tcp   2049  nfs_acl
  100003   3   udp   2049  nfs
  100003   4   udp   2049  nfs
  100227   3   udp   2049  nfs_acl
  100021   1   udp   46666 nlockmgr
  100021   3   udp   46666 nlockmgr
  100021   4   udp   46666 nlockmgr
  100021   1   tcp   42955 nlockmgr
  100021   3   tcp   42955 nlockmgr
  100021   4   tcp   42955 nlockmgr
  100011   1   udp   875   rquotad
  100011   2   udp   875   rquotad
  100011   1   tcp   875   rquotad
  100011   2   tcp   875   rquotad
```

2. Run the following commands on the NFS server:

```
firewall-cmd --permanent --add-port=111/tcp
firewall-cmd --permanent --add-port=54302/tcp
firewall-cmd --permanent --add-port=20048/tcp
firewall-cmd --permanent --add-port=2049/tcp
firewall-cmd --permanent --add-port=46666/tcp
firewall-cmd --permanent --add-port=42955/tcp
firewall-cmd --permanent --add-port=875/tcp
```

3. To apply the change, restart the firewall service:

```
firewall-cmd --reload
```

4. Mount the shared directories on each client with the following commands:

```
mount -t nfs 192.168.1.101:/opt/trifacta/logs /opt/trifacta/logs  
mount -t nfs 192.168.1.101:/opt/trifacta/conf /opt/trifacta/conf
```

5. The NFS shared directories should mount without error.

Configure for Hadoop

Contents:

- *Before You Begin*
 - *Key deployment considerations*
 - *Platform configuration*
 - *Required Configuration for Hadoop*
 - *Specify Trifacta user*
 - *Data storage*
 - *Configure ResourceManager settings*
 - *Specify distribution client bundle*
 - *Default Hadoop job results format*
 - *Additional Configuration for Hadoop*
 - *Authentication*
 - *Hadoop KMS*
 - *Hive access*
 - *High availability environment*
 - *Apply cluster configuration files via symlink*
 - *Modify Trifacta configuration changes*
 - *Configure Snappy publication*
 - *Debugging*
-

The Designer Cloud powered by Trifacta® platform supports integration with a number of Hadoop distributions, using a range of components within each distribution. This page provides information on the set of configuration tasks that you need to complete to integrate the platform with your Hadoop environment.

Before You Begin

Key deployment considerations

1. **Hadoop cluster:** The Hadoop cluster should already be installed and operational. As part of the install preparation, you should have prepared the Hadoop platform for integration with the Designer Cloud powered by Trifacta platform . See *Prepare Hadoop for Integration with the Platform*.
 - a. For more information on the components supported in your Hadoop distribution, See *Install Reference*.
2. **Storage:** on-premises, cloud, or hybrid.
 - a. The Designer Cloud powered by Trifacta platform can interact with storage that is in the local environment, in the cloud, or in some combination. How your storage is deployed affects your configuration scenarios. See *Storage Deployment Options*.
3. **Base storage layer:** You must configure one storage platform to be the base storage layer. Details are described later.

NOTE: Some deployments require that you select a specific base storage layer.

After you have defined the base storage layer, it cannot be changed. Please review your *Storage Deployment Options* carefully. The required configuration is described later.

Hadoop versions

The Designer Cloud powered by Trifacta platform supports integration only with the versions of Hadoop that are supported for your version of the platform.

NOTE: The versions of your Hadoop software and the libraries in use by the Designer Cloud powered by Trifacta platform must match. Unless specifically directed by *Alteryx Support*, integration with your Hadoop cluster using a set of Hadoop libraries from a different version of Hadoop is not supported.

For more information, see *Product Support Matrix*.

Platform configuration

After the Designer Cloud powered by Trifacta platform and its databases have been installed, you can perform platform configuration. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

NOTE: Some platform configuration is required, regardless of your deployment. See *Required Platform Configuration*.

Required Configuration for Hadoop

Please complete the following sections to configure the platform to work with Hadoop.

Specify Trifacta user

NOTE: Where possible, you should define or select a user with a `userID` value greater than 1000. In some environments, lower `userID` values can result in failures when running jobs on Hadoop.

Set the Hadoop username [`hadoop.user` (default=`trifacta`)] for the Designer Cloud powered by Trifacta platform to use for executing jobs:

```
"hdfs.username": [hadoop.user],
```

If the Trifacta software is installed in a Kerberos environment, additional steps are required, which are described later.

Data storage

The Designer Cloud powered by Trifacta platform supports access to the following Hadoop storage layers:

- HDFS
- S3

Set the base storage layer

At this time, you should define the base storage layer from the platform.

The platform requires that one backend datastore be configured as the base storage layer. This base storage layer is used for storing uploaded data and writing results and profiles. Please complete the following steps to set the base storage layer for the Designer Cloud powered by Trifacta platform .

You cannot change the base storage layer after it has been set. You must uninstall and reinstall the platform to change it.

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 the value for your base storage layer:

```
"webapp.storageProtocol": "hdfs",
```

3. Save your changes and restart the platform.

NOTE: To complete the integration with the base storage layer, additional configuration is required.

Required configuration for each type of storage is described below.

S3

The Designer Cloud powered by Trifacta platform can integrate with an S3 bucket:

- If you are using HDFS as the base storage layer, you can integrate with S3 for read-only access.
- Base storage layer requires read-write access.

NOTE: If you are integrating with S3, additional configuration is required. Instead of completing the HDFS configuration below, please enable read-write access to S3. See *S3 Access* in the *Configuration Guide*.

HDFS

If output files are to be written to an HDFS environment, you must configure the Designer Cloud powered by Trifacta platform to interact with HDFS.

- Hadoop Distributed File Service (HDFS) is a distributed file system that provides read-write access to large datasets in a Hadoop cluster. For more information, see http://hadoop.apache.org/docs/r1.2.1/hdfs_design.html.

If your deployment is using HDFS, do not use the `trifacta/uploads` directory. This directory is used for storing uploads and metadata, which may be used by multiple users. Manipulating files outside of the Designer Cloud application can destroy other users' data. Please use the tools provided through the interface for managing uploads from HDFS.

NOTE: Use of HDFS in safe mode is not supported.

Below, replace the value for `[hadoop.user (default=trifacta)]` with the value appropriate for your environment. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

```
"hdfs": {
  "username": "[hadoop.user]",
  ...
  "namenode": {
    "host": "hdfs.example.com",
    "port": 8080
  },
},
```

Parameter	Description
username	Username in the Hadoop cluster to be used by the Designer Cloud powered by Trifacta platform for executing jobs.
namenode.host	Host name of namenode in the Hadoop cluster. You may reference multiple namenodes.
namenode.port	Port to use to access the namenode. You may reference multiple namenodes. NOTE: Default values for the port number depend on your Hadoop distribution. See <i>System Ports</i> in the Planning Guide.

Individual users can configure the HDFS directory where exported results are stored.

NOTE: Multiple users cannot share the same home directory.

See *Storage Config Page* in the User Guide.

Access to HDFS is supported over one of the following protocols:

- See *WebHDFS* below.
- See *HttpFS* below.

WebHDFS

If you are using HDFS, it is assumed that WebHDFS has been enabled on the cluster. Apache WebHDFS enables access to an HDFS instance over HTTP REST APIs. For more information, see <https://hadoop.apache.org/docs/r1.0.4/webhdfs.html>.

The following properties can be modified:

```
"webhdfs": {
  ...
  "version": "/webhdfs/v1",
  "host": "",
  "port": 50070,
  "https": false
},
```

Parameter	Description
version	Path to locally installed version of WebHDFS. NOTE: For <i>version</i> , please leave the default value unless instructed to do otherwise.

host	Hostname for the WebHDFS service. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> NOTE: If this value is not specified, then the expected host must be defined in <code>hdfs.namenode.host</code>. </div>
port	Port number for WebHDFS. The default value is 50070. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> NOTE: The default port number for SSL to WebHDFS is 50470 . </div>
httpfs	To use HttpFS instead of WebHDFS, set this value to <code>true</code> . The port number must be changed. See <i>HttpFS</i> below.

Steps:

1. Set `webhdfs.host` to be the hostname of the node that hosts WebHDFS.
2. Set `webhdfs.port` to be the port number over which WebHDFS communicates. The default value is 50070. For SSL, the default value is 50470.
3. Set `webhdfs.httpfs` to `false`.
4. For `hdfs.namenodes`, you must set the `host` and `port` values to point to the active namenode for WebHDFS.

HttpFS

You can configure the Designer Cloud powered by Trifacta platform to use the HttpFS service to communicate with HDFS, in addition to WebHDFS.

NOTE: HttpFS serves as a proxy to WebHDFS. When HttpFS is enabled, both services are required.

In some cases, HttpFS is required:

- High availability requires HttpFS.
- Your secured HDFS user account has access restrictions.

If your environment meets any of the above requirements, you must enable HttpFS. For more information, see *Enable HttpFS* in the Configuration Guide.

Configure ResourceManager settings

Configure the following:

```
"yarn.resourcemanager.host": "hadoop",
"yarn.resourcemanager.port": 8032,
```

NOTE: Do not modify the other host/port settings unless you have specific information requiring the modifications.

For more information, see *System Ports* in the Planning Guide.

Specify distribution client bundle

The Designer Cloud powered by Trifacta platform ships with client bundles supporting a number of major Hadoop distributions. You must configure the `jarfile` for the distribution to use. These distributions are stored in the following directory:

/opt/trifacta/hadoop-deps

Configure the bundle distribution property (`hadoopBundleJar`) in platform configuration. Examples:

Hadoop Distribution	<code>hadoopBundleJar</code> property value
Cloudera	"hadoop-deps/cdh-x.y/build/libs/cdh-x.y-bundle.jar"
Cloudera Data Platform	"hadoop-deps/cdp-x.y.z/build/libs/cdp-x.y.z-bundle.jar"

where:

`x.y` is the major-minor build number (e.g. 5.4)

NOTE: The path must be specified relative to the install directory.

Tip: If there is no bundle for the distribution you need, you might try the one that is the closest match in terms of Apache Hadoop baseline. For example, CDH5 is based on Apache 2.3.0, so that client bundle will probably run ok against a vanilla Apache Hadoop 2.3.0 installation. For more information, see *Alteryx Support*.

Cloudera distribution

Some additional configuration is required. See *Configure for Cloudera* in the Configuration Guide.

Default Hadoop job results format

For smaller datasets, the platform recommends using the Trifacta Photon running environment.

For larger datasets, if the size information is unavailable, the platform recommends by default that you run the job on the Hadoop cluster. For these jobs, the default publishing action for the job is specified to run on the Hadoop cluster, generating the output format defined by this parameter. Publishing actions, including output format, can always be changed as part of the job specification.

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

```
"webapp.defaultHadoopFileFormat": "csv",
```

Accepted values: `csv`, `json`, `avro`, `pqt`

For more information, see *Run Job Page* in the User Guide.

Additional Configuration for Hadoop

Authentication

Kerberos

The Designer Cloud powered by Trifacta platform supports integration with Kerberos security. The platform can utilize Kerberos' secure impersonation to broker interactions with the Hadoop environment.

See *Configure for Kerberos Integration*.

See *Configure for Secure Impersonation*.

Single Sign-On

The Designer Cloud powered by Trifacta platform can integrate with your SSO platform to manage authentication to the Designer Cloud application. See *Configure SSO for AD-LDAP*.

Hadoop KMS

If you are using Hadoop KMS to encrypt data transfers to and from the Hadoop cluster, additional configuration is required. See *Configure for KMS*.

Hive access

Apache Hive is a data warehouse service for querying and managing large datasets in a Hadoop environment using a SQL-like querying language. For more information, see <https://hive.apache.org/>.

See *Configure for Hive*.

High availability environment

You can integrate the platform with the Hadoop cluster's high availability configuration, so that the Designer Cloud powered by Trifacta platform can match the failover configuration for the cluster.

NOTE: If you are deploying high availability failover, you must use HttpFS, instead of WebHDFS, for communicating with HDFS, which is described in a previous section.

For more information, see *Enable Integration with Cluster High Availability*.

After you have performed the base installation of the Designer Cloud powered by Trifacta® platform, please complete the following steps if you are integrating with a Hadoop cluster.

Apply cluster configuration files via symlink

If the Designer Cloud powered by Trifacta platform is being installed on an edge node of the cluster, you can create a symlink from a local directory to the source cluster files so that they are automatically updated as needed.

1. Navigate to the following directory on the Trifacta node:

```
cd /opt/trifacta/conf/hadoop-site
```

2. Create a symlink for each of the Hadoop Client Configuration files referenced in the previous steps. Example:

```
ln -s /etc/hadoop/conf/core-site.xml core-site.xml
```

3. Repeat the above steps for each of the Hadoop Client Configuration files.

Modify Trifacta configuration changes

1. To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. **HDFS:** Change the host and port information for HDFS as needed. Please apply the port numbers for your distribution:

```
"hdfs.namenode.host": "<namenode>",
"hdfs.namenode.port": <hdfs_port_num>
"hdfs.yarn.resourcemanager": {
  "hdfs.yarn.webappPort": 8088,
  "hdfs.yarn.adminPort": 8033,
  "hdfs.yarn.host": "<resourcemanager_host>",
  "hdfs.yarn.port": <resourcemanager_port>,
  "hdfs.yarn.schedulerPort": 8030
```

3. Save your changes and restart the platform.

Configure Snappy publication

If you are publishing using Snappy compression, you may need to perform the following additional configuration.

Steps:

1. Verify that the `snappy` and `snappy-devel` packages have been installed on the Trifacta node. For more information, see <https://hadoop.apache.org/docs/r2.7.1/hadoop-project-dist/hadoop-common/NativeLibraries.html>.
2. From the Trifacta node, execute the following command:

```
hadoop checknative
```

3. The above command identifies where the native libraries are located on the Trifacta node.
4. **Cloudera:**
 - a. On the cluster, locate the `libsnapappy.so` file. Verify that this file has been installed on all nodes of the cluster, including the Trifacta node. Retain the path to the file on the Trifacta node.
 - b. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
 - c. Locate the `spark.props` configuration block. Insert the following properties and values inside the block:

```
"spark.driver.extraLibraryPath": "/path/to/file",
"spark.executor.extraLibraryPath": "/path/to/file",
```

5. Save your changes and restart the platform.
6. Verify that the `/tmp` directory has the proper permissions for publication. For more information, see *Supported File Formats*.

Debugging

You can review system services and download log files through the Designer Cloud application .

See *System Services and Logs*.

Configure for Cloudera

Contents:

- *Prerequisites*
- *Configure for Cloudera Data Platform*
- *Configure Designer Cloud powered by Trifacta platform*
 - *Configure Hive Locations*
 - *Configure SSL for Hive*
- *Restart*

This section provides additional configuration requirements for integrating the Designer Cloud powered by Trifacta® platform with the Cloudera platform.

NOTE: Except as noted, the following configuration items apply to the latest supported version of Cloudera platform.

Prerequisites

Before you begin, it is assumed that you have completed the following tasks:

1. Successfully installed a supported version of Cloudera platform into your enterprise infrastructure.
2. Installed the Trifacta software in your environment. For more information, see *Install Software*.
3. Reviewed the mechanics of platform configuration. See *Required Platform Configuration*.
4. Configured access to the Trifacta database. See *Configure the Databases*.
5. Performed the basic cluster integration configuration. See *Configure for Hadoop*.
6. You have access to platform configuration through the Trifacta node or through the Admin Settings page. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Configure for Cloudera Data Platform

Cloudera Data Platform (CDP) offers a broad set of cloud services for enterprise data, including data analytics and artificial intelligence. For more information, see <https://docs.cloudera.com/cdp/latest/overview/topics/cdp-overview.html>.

The configuration required to integrate the Designer Cloud powered by Trifacta platform with Cloudera Data Platform (CDP) is very similar to the base Cloudera configuration with the following exceptions:

- You must reference the correct Hadoop dependencies bundle JAR for the platform to use, which should already be specified. For more information, see *Configure for Hadoop*.
- For Spark:
 - You must specify the use of native libraries.
 - You must specify a specific Spark version for the platform to use.

These differences in configuration are described inline with the base Cloudera and Spark configuration.

Configure Designer Cloud powered by Trifacta platform

Configure Hive Locations

If you are enabling an integration with Hive on the cluster, there are some distribution-specific parameters that must be set. For more information, see *Configure for Hive*.

Configure SSL for Hive

CDH supports two methods of enabling SSL communications with Hive:

1. **SASL-QOP method:** Enable encryption between Hive JDBC and HiveServer 2 using SASL-QOP. This method is available by default with the Designer Cloud powered by Trifacta platform .
2. **TLS/SSL method:** Use TLS/SSL encryption for JDBC connections to HiveServer 2.

To determine the method in use:

1. In Cloudera Manager configuration, search for: `tls`.
2. If the options for TLS/SSL are enabled, please complete the following configuration steps.
3. If these options are not enabled, the cluster can still use the SASL-QOP method. For more information on this method, see *Configure for Hive*.

Enable TLS/SSL Method

Steps:

1. The default Hive JDBC driver provided with your Trifacta installation must be replaced with the drive provided by Cloudera. Please complete the following commands, noting the wildcards (*) in the JAR path:

NOTE: The current driver must be removed or replaced in the working directory. Do not leave it in the directory.

```
cd /opt/trifacta/services/data-service/build/dependencies
rm *hive*jdbc*
cp /opt/cloudera/parcels/CDH-6.2*/jars/hive-jdbc-1.1.0-cdh6.2.0*jar .
```

2. Enable the Hive connection. For the Hive connection string options, you must specify something like the following:

```
"connectStrOpts": ";ssl=true;sslTrustStore=</path/to/truststore>;
trustStorePassword=<storePassword>"
```

NOTE: The truststore specified above must exist on the Trifacta node and be accessible to the Trifacta user through the listed password. This truststore must contain the certificate for the Hive server.

3. Save the parameters file. For more information on creating the connection, see *Configure for Hive*.
4. Restart the platform. See *Start and Stop the Platform*.
5. Verify that you can read from a Hive source through the application. See *Hive Connections*.

Restart

To apply your changes, restart the platform. See *Start and Stop the Platform*.

Configure Hadoop Authentication

Contents:

- *End-User Authorization*
 - *Security Scenarios for HDFS Access*
 - *Security Scenarios for Hive Access*

Depending on your Hadoop security environment, the following sections describe implications for the platform and provide links to appropriate documentation.

End-User Authentication

Depending on use of Single Sign On, Trifacta users access the application using the following credentials.

Security Features	Implications
Single Sign On (SSO)	Users access application using the LDAP/AD principal associated with their account. For more information, see <i>Configure SSO for AD-LDAP</i> .
All other security scenarios	Users access application using their Trifacta userId.

End-User Authorization

The following security scenarios apply to accessing Hadoop-based data storage.

Security Scenarios for HDFS Access

Depending on the following security features implemented in your Hadoop environment, your interactions with HDFS may have different implications.

Security Features	Implications
No Kerberos authentication	<ul style="list-style-type: none">• All Trifacta users use the <code>[hadoop.user (default=trifacta)]</code> Hadoop user to access HDFS.• No security is applied.
<ul style="list-style-type: none">• Kerberos authentication• No secure impersonation	<ul style="list-style-type: none">• All Trifacta users authenticate and then use delegation token for all requests to HDFS.<ul style="list-style-type: none">• If you receive an error when attempting to contact HDFS, your delegation token may have failed due to configuration error. Please contact your Trifacta administrator.• All Trifacta users use the <code>[hadoop.user]</code> Hadoop user to access HDFS.
<ul style="list-style-type: none">• Kerberos authentication• Secure impersonation	<ul style="list-style-type: none">• All Trifacta users authenticate with the <code>[hadoop.user]</code> user keytab. The <code>[hadoop.user]</code> user retrieves a delegation token on behalf of the user's Hadoop principal.<ul style="list-style-type: none">• If you receive an error when attempting to contact HDFS, your delegation token may have failed due to a configuration error. Please contact your Trifacta administrator.• Trifacta users securely impersonate using their assigned Hadoop principal on HDFS.

For more technical information:

- See *Configure for Kerberos Integration*.
- See *Configure for Secure Impersonation*.

Security Scenarios for Hive Access

Depending on the following security features implemented in your Hadoop environment, your interactions with Hive may have different implications.

Security Features	Implications
No additional security features	<ul style="list-style-type: none"> • All Trifacta users use the [<code>hadoop.user</code>] Hadoop user to access Hive. • No security is applied.
<ul style="list-style-type: none"> • Kerberos authentication • No secure impersonation 	<ul style="list-style-type: none"> • Trifacta users authenticate with the [<code>hadoop.user</code>] user keytab for all requests to Hive. <ul style="list-style-type: none"> • If you receive an error when attempting to contact Hive, authentication likely failed due to a configuration error. Please contact your Trifacta administrator.
<ul style="list-style-type: none"> • Kerberos authentication • Secure impersonation 	<ul style="list-style-type: none"> • Trifacta users authenticate with the [<code>hadoop.user</code>] user keytab and then send proxying requests on behalf of the user's Hadoop principal. <ul style="list-style-type: none"> • If you receive an error when attempting to contact Hive, authentication likely failed due to a configuration error. Please contact your Trifacta administrator. • Hive is responsible for respecting proxy permissions, with the <code>hive</code> user itself proxying as [<code>hadoop.user</code>] proxying as the user's Hadoop principal.
<ul style="list-style-type: none"> • Kerberos authentication • Secure authentication • Sentry role-based access (Cloudera only) 	<ul style="list-style-type: none"> • Trifacta users authenticate with the [<code>hadoop.user</code>] user keytab and then send proxying requests on behalf of the user's Hadoop principal. <ul style="list-style-type: none"> • If you receive an error when attempting to contact Hive, authentication likely failed due to a configuration error. Please contact your Trifacta administrator. • Hive executes access to the physical data file on HDFS as the Unix or LDAP user <code>hive</code>, which should be part of the group [<code>hadoop.group</code> (default=<code>trifactausers</code>)].
<ul style="list-style-type: none"> • Sentry role-based access (Cloudera only) 	<ul style="list-style-type: none"> • Hive authorizes access with a Sentry lookaside. The [<code>hadoop.user</code>] user as well as the user's Hadoop principal should be configured with appropriate privileges and roles in Sentry.
<ul style="list-style-type: none"> • Kerberos authentication • No secure authentication • Sentry role-based access (Cloudera only) 	<ul style="list-style-type: none"> • Trifacta users authenticate with the [<code>hadoop.user</code>] user keytab. <ul style="list-style-type: none"> • If you receive an error when attempting to contact Hive, authentication likely failed due to a configuration error. Please contact your Trifacta administrator. • Hive executes access to the physical data file on HDFS as the Unix or LDAP user <code>hive</code>, which should be part of the group [<code>hadoop.group</code> (default=<code>trifactausers</code>)].

Configure for Kerberos Integration

Contents:

- *Prerequisites for Kerberos integration*
- *Configure the KDC*
- *Create keytab in Active Directory environments*
- *Configure the Designer Cloud powered by Trifacta platform for Kerberos*
- *Configure Kerberos-delegated relational connections*

This document describes how to set up a Trifacta® user in Kerberos.

- Kerberos provides authentication services across a wide variety of platforms. See <http://www.kerberos.org/>.

Prerequisites for Kerberos integration

Before you begin, please verify the following:

1. The `[hadoop.user (default=trifacta)]` user is created and enabled on each node in the Hadoop cluster.

NOTE: If LDAP is enabled, the `trifacta` user should be created in the same realm as the cluster.

2. On the Trifacta host, the directory `/opt/trifacta` is owned by the `[hadoop.user]` user .
3. The `[hadoop.user]` user exists on each node in the Hadoop cluster.

NOTE: The `[hadoop.user]` must have the same user ID and group ID on each node in the cluster. Depending on your cluster's configuration, this requirement may require an LDAP command. Configuring LDAP is beyond the scope of this document.

4. The `[hadoop.user]` user must be a member of any special group that is permitted to access HDFS or to run Hadoop jobs.

condit

Configure the KDC

Steps:

1. On your KDC node, configure a Kerberos principal for the Designer Cloud powered by Trifacta platform :
 - a. The principal's identifier has two parts: its **name** and its **realm**. For example, the principal `trifacta@HADOOPVAL.MSSVC.LOCAL` has the name `trifacta` and the realm `HADOOPVAL.MSSVC.LOCAL`.
 - b. Retain the name and principal for later configuration.
2. Create a keytab file for the Trifacta principal. Command:

```
kadmin xst -k trifacta.keytab <full_principal_identifier>
```

where:

`<full_principal_identifier>` is the principal identifier in Kerberos.

On the KDC, you may have to run `kadmin.local` instead of `kadmin`. The rest of the arguments should remain the same.

NOTE: If you're creating a keytab file in an AD environment, alternative instructions may need to be applied. See below.

3. Verify that the keytab is working. Command:

```
klist -e -k -t trifacta.keytab
```

4. Copy the keytab to the Trifacta node in the following directory:
`/opt/trifacta/conf/trifacta.keytab`
5. Configure the keytab file so that it is owned by the `[hadoop.user]` user. It should only be readable by that user.

NOTE: Verify that all user principals that use the platform are also members of the group of the keytab user.

Create keytab in Active Directory environments

Some additional instructions are provided for the following environments.

For MIT Kerberos

See <https://kb.iu.edu/d/aumh> :

```
> ktutil
ktutil: addent -password -p username@EXAMPLE.COM -k 1 -e rc4-hmac
Password for username@EXAMPLE.COM: [enter your password]
ktutil: addent -password -p username@EXAMPLE.COM -k 1 -e aes256-cts
Password for username@EXAMPLE.COM: [enter your password]
ktutil: wkt username.keytab
ktutil: quit
```

For Heimdal Kerberos

```
> ktutil -k username.keytab add -p username@EXAMPLE.COM -e arcfour-hmac-md5 -V 1
```

If the keytab created in Heimdal does not work, you may need an `aes256-cts` entry. In this case, locate a machine with MIT Kerberos, and use the MIT Kerberos method instead.

Configure the Designer Cloud powered by Trifacta platform for Kerberos

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

Locate the `kerberos` section, which controls Kerberos authentication.

Example configuration:

Substitute your own values in place of the example values as appropriate.

```

"kerberos.enabled": true,
"kerberos.principal": "trifacta",
"kerberos.kdc": "kdc.mssvc.local",
"kerberos.realm": "HADOOPVAL.MSSVC.LOCAL",
"kerberos.keytab": "/opt/trifacta/conf/trifacta.keytab"
"kerberos.principals.hive": "<UNUSED>",
"kerberos.principals.namenode": "nn/_HOST@EXAMPLE.COM"
"kerberos.principals.resourcemanager": "<YOUR_VALUE_HERE>",

```

Parameter	Description
enabled	To enable Kerberos authentication, set this value to <code>true</code> .
principal	The name part of the principal you created in the KDC
kdc	The host of the KDC
realm	Realm of the KDC
keytab	Directory in the Trifacta deployment where the Kerberos keytab file is stored
principals	<p>List of jobtrackers and namenodes that are governed by Kerberos</p> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <p>NOTE: <code>kerberos.principals.hive</code> is unused. This value must be inserted into the Hive connection definition. See <i>Hive Connections</i>.</p> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-bottom: 5px;"> <p>NOTE: If you don't know the values to use here, see <i>Set principal values</i> below.</p> </div> <div style="border: 1px solid #ccc; padding: 5px;"> <p>NOTE: If you don't specify principal names in the <code>principals</code> definition section, the default names are used: <code>mapred/<jobtracker host>@<realm></code>. You should specify the principals explicitly.</p> </div>

At this point, you should be able to load files from HDFS and run jobs against the kerberized Hadoop cluster.

Set principal values for YARN

Check the following Hadoop config properties in `yarn-site.xml` :

```

principals.jobtracker = yarn.resourcemanager.principal
principals.namenode = dfs.namenode.kerberos.principal

```

Configure Kerberos-delegated relational connections

When Kerberos has been enabled in the platform, you can apply the global keytab to be used for SSO connections to relational sources of data. For more information, see *Enable SSO for Relational Connections*.

Configure for Secure Impersonation

Contents:

- *Users and groups for secure impersonation*
- *Hadoop configuration for secure impersonation*
- *HDFS Directories*
- *Trifacta configuration for secure impersonation*
- *Provisioning impersonated users*

In a Hadoop environment, **secure impersonation** enables the Designer Cloud powered by Trifacta® platform and its users to act as the signed-in user when performing actions on Hadoop. When enabled, you can leverage the permissions infrastructure in your Hadoop cluster to control privacy level, collaboration, and data sharing for your user base. For the Trifacta user, their jobs and job outputs are owned by the specified Trifacta user, instead of the Hadoop user [`hadoop.user`]. The Trifacta system account is required even in secure impersonation mode.

- This configuration is optional.
- For more information on Hadoop secure impersonation, see <http://hadoop.apache.org/docs/current/hadoop-project-dist/hadoop-common/Superusers.html>.

Please complete these steps to to enable secure impersonation.

NOTE: Trifacta secure impersonation requires Kerberos to be applied to the Hadoop cluster. However, you can use Kerberos without enabling secure impersonation, if desired. See *Configure for Kerberos Integration*.

Users and groups for secure impersonation

On the Hadoop cluster, the Designer Cloud powered by Trifacta platform requires a common Unix or LDAP group containing the [`hadoop.user (default=trifacta)`] and all Trifacta users.

NOTE: In UNIX environments, usernames and group names are case-sensitive. Please be sure to use the case-sensitive names for users and groups in your Hadoop configuration and Trifacta configuration file.

NOTE: If the HDFS user has restrictions on its use, it is not suitable for use with secure impersonation. Instead, you should enable HttpFS and use a separate HttpFS-specific user account instead. For more information, see *Configure for Hadoop*.

Assuming this group is named `trifactausers`:

- Create a Unix or LDAP group `trifactausers`
- Make user [`hadoop.user`] a member of `trifactausers`
- Verify that all user principals that use the platform are also members of the `trifactausers` group.

Hadoop configuration for secure impersonation

In your Kerberos configuration, you must configure the user [`hadoop.user`] as a secure impersonation proxy user for Trifacta users.

NOTE: The following addition must be made to your Hadoop cluster configuration file. This file must be copied to the Trifacta node with the required other cluster configuration files. See *Configure for Hadoop*.

In `core-site.xml` on the Hadoop cluster, add the following configuration, replacing the values for `[hadoop.user]` and `[hadoop.group (default=trifactausers)]` with the values appropriate for your environment:

```
<!-- Trifacta secure impersonation -->
<property>
  <name>hadoop.proxyuser.[hadoop.user].hosts</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.proxyuser.[hadoop.user].groups</name>
  <value>[hadoop.group]</value>
</property>
```

HDFS Directories

Verify that the shared upload and job results directories are owned and writeable to the `trifactausers` group.

For more information on HDFS directories and their permissions, see *Prepare Hadoop for Integration with the Platform*.

Stricter directory permissions in an impersonated environment

By default, the directories and sub-directories of the locations for uploaded data and job results are set to 730 in an environment with secure impersonation enabled. This configuration allows impersonated users to do the following:

NOTE: Stricter permissions sets can adversely affect users' ability to access shared flows.

- The 7 user-permission implies that individual users have full permissions over their own directories.
 - Individual users can read only data in their own upload directory below `/trifacta/uploads`.
- The 3 user-permission is used because the top-level directory is owned by the `[hadoop.user]` user. Each impersonated user in the `trifactausers` group requires write and execute permissions on their own directory to create it and manage it. This permission set implies that the `trifactausers` group has read and execute permissions over a user's upload directory.
- Without access-level controls, these permissions are inherited from the parent directory and have the following implications:
 - Since impersonated group users have execute permissions, they can list all directories in this area.
 - Since impersonated group users have write permissions, they can theoretically write to any other user's upload directory, although this directory is not configurable.

Within the upload area, each user of the Designer Cloud powered by Trifacta platform is assigned an individual directory. For simplicity, the permissions on these directories are automatically applied to the sub-directories. In an impersonated environment, an individual directory is owned by the Hadoop principal for the user, so if two or more users share the same Hadoop principal, they have theoretical access to each others' directories. This simple scheme can be replaced by a more secure method using access-level controls.

NOTE: To enable these stricter permissions, access-level controls must be enabled on your Hadoop cluster. For more information, please see the documentation for your Hadoop distribution.

If access-level controls are enabled for your impersonated environment, you can apply stricter permissions on these sub-directories for additional security.

Steps:

The following steps apply 730 permissions to the top-level directory and 700 to all user sub-directories. With these stricter permissions on sub-directories, no one other than the user, including the `trifacta` user, can access the user's sub-directory.

1. The `/trifacta/uploads` value is the default value for upload location in HDFS.
 - a. In an individual deployment, the directory setting is defined in platform configuration . You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*. Locate the value for `hdfs.pathsConfig.fileUpload`.

NOTE: The following steps can also be applied to the directory where job results are written. By default, this directory is `/trifacta/queryResults`. For more secure controls over job results, you should also retrieve the value for `hdfs.pathsConfig.batchResults`.

- b. Replace the values in the following steps with the value from your configuration.
2. Before you begin, you should consider resetting all access-level controls on the upload directories and sub-directories:

```
hdfs dfs -setfacl -b /trifacta/uploads
```

3. The following command removes the application of the permissions from the `uploads` directory and any sub-directory to members of the default group. So, an individual group member's permissions are not automatically shared with the group:

```
hdfs dfs -setfacl -R -m default:group:--- /trifacta/uploads
```

4. The following command is required to enable all users to access dictionaries:

```
hdfs dfs -setfacl -R -m default:group:rwX /trifacta/uploads/0
```

5. The following step sets the permissions at the top level to 730 :

```
hdfs dfs -chmod 730 /trifacta/uploads
```

6. Sub-directory permissions are a combination of these permissions and any relevant access-level controls.
7. **Apply to queryResults directory:** Repeat the above steps for the `/trifacta/queryResults` directory as needed.
8. **ACL for Hive:** If you need to apply access controls to Hive, you can use the following:

```
hdfs dfs -setfacl -R -m default:user:hive:rwX /trifacta/queryResults
```

User directories for YARN

For YARN deployments, each Hadoop user must have a home directory for which the user has write permissions. This directory must be located in the following location within HDFS:

```
/user/<username>
```

where:

- <username> is the Hadoop principal to use.

NOTE: For jobs executed on the default Trifacta Photon running environment, user output directories must be created with the same permissions as you want for the transform and sampling jobs executed on the server. Users may be able to see the output directories of other users, but output job files are created with the user umask setting (`hdfs.permissions.userUMask`), as defined in platform configuration.

Example for Hadoop principal `myUser`:

```
hdfs dfs -mkdir /user/myUser
hdfs dfs -chown myUser /user/myUser
```

Optional:

```
hdfs dfs -chmod -R 700 /user/myUser
```

Trifacta configuration for secure impersonation

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

Set the following parameter to `true`.

```
"hadoopImpersonation" : true,
```

If you have enabled the Spark running environment for job execution, you must enable the following parameter as well:

```
"spark-job-service.sparkImpersonationOn" : true,
```

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

Umask permissions

Under secure impersonation, the Designer Cloud powered by Trifacta platform utilizes two separate umask permission sets. If secure impersonation is not enabled, the Designer Cloud powered by Trifacta platform utilizes the `systemUmask` for all operations.

NOTE: Umask settings are three-digit codes for defining the bit switches for read, write, and execute permissions for users, groups, and others (in that order) for a file or directory. These settings are inverse settings. For example, the umask value of 077 enables read, write, and execute permissions for users and disables all permissions for groups and others. For more information, see <https://en.wikipedia.org/wiki/Umask>.

Name	Property	Description
userUmask	hdfs.permissions.userUMask	Controls the output permissions of files and directories that are created by impersonated users. These permissions define private permission settings for individual users.
systemUmask	hdfs.permissions.systemUMask	Controls the output permissions of files and directories that are created by the Trifacta system user. These permissions also control resources for the admin user and resources that are shared between Trifacta users.

Notes:

- In a secure impersonation environment, systemUmask should be defined as 027 (the default value), which enables read access to shared resources for all users in the Trifacta group.
 - For greater security, it is possible to set the userUmask to 077, which locks down individual user directories under /trifacta/queryResults. However, secure impersonation requires more permissions on the systemUMask to enable sharing of resources.
 - Please note that the permission settings for the admin user are controlled by systemUmask.

Provisioning impersonated users

NOTE: A newly created user in the platform cannot log in unless provisioned by a platform administrator, even if self-registration is enabled. The administrator must apply the Hadoop principal to the account.

To provision users as admin, log in and visit the Admin Settings page from the drop-down menu on the top right. Locate the Users section and click **Edit Users**. If you have the Secure Hadoop Impersonation flag on with Kerberos enabled, you should see a Hadoop Principal column. From here, you can assign each user a Hadoop principal. Multiple users can share the same Hadoop principal, but each Trifacta user must have a Hadoop principal assigned to them.

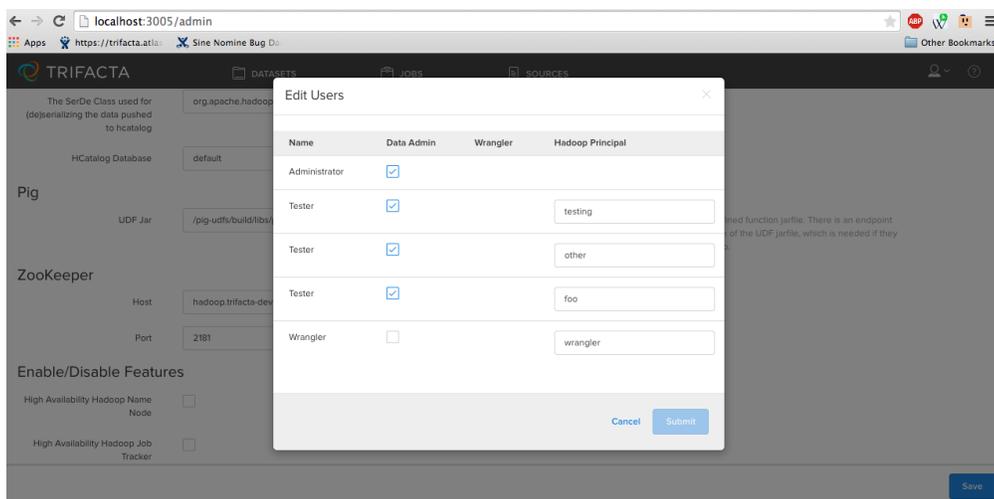


Figure: Editing users

Enable HttpFS

Contents:

- *Prerequisites*
 - *Configuration*
 - *Enable SSL*
-

This section describes how to enable the Designer Cloud powered by Trifacta® platform to use the HttpFS service for communicating with Hadoop HDFS. HttpFS is commonly used in the following scenarios:

1. **High Availability.** WebHDFS does not support High Availability failover. You must use HttpFS instead.
2. **HDFS user is not available for secure impersonation.** If you have enabled secure impersonation in an environment where the HDFS superuser is restricted from use, you can enable HttpFS and use the HttpFS superuser for secure impersonation.

Prerequisites

Before you begin, please verify that you have done the following in your environment:

- Enabled HDFS in your Hadoop cluster.
- Installed `hadoop-httpfs` into your Hadoop cluster.
- HttpFS has been enabled on a known port on the cluster.

NOTE: If you are enabling HttpFS for use with High Availability, you should avoid enabling the HttpFS service on the primary namenode of the cluster. For more information, see *Enable Integration with Cluster High Availability*.

NOTE: By default, HttpFS is available on port 14000. Please verify the port number in use for your cluster.

- Started HttpFS service on the cluster.

Configuration

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

Steps:

1. The configuration settings for HttpFS are within the HDFS configuration area:

```
"hdfs.webhdfs.host": "",  
"hdfs.webhdfs.port": 14000,  
"hdfs.webhdfs.httpfs": true,
```

2. Set `hdfs.webhdfs.httpfs` to `true`.
3. Specify the host and port for the HttpFS service. You can use one of the following methods:
 - a. Specify `hdfs.webhdfs.host` and `hdfs.webhdfs.port` values to point to the node hosting HttpFS.

- b. Leave the `hdfs.webhdfs.host` value empty, in which case the platform falls back to using the namenode host as the WebHDFS host. Modify that value if required.

NOTE: By default, the platform expects this service to be available on port 14000. Please apply the value that matches your cluster environment.

4. Save your changes and restart the platform.

Enable SSL

Optionally, you can enable secure (SSL) communications between the platform and HttpFS.

NOTE: The most secure method requires the creation and deployment of an SSL certificate for the HDFS instance. These steps provide instructions for how to do so.

If this certificate is not available, you can still enable communication over SSL over WebHDFS or HttpFS. Please skip steps 1 and 2 and complete the secure configuration without certificate export.

Steps:

1. Deploy a PEM file certificate that can be read by the `[os.user (default=trifacta)]` user account on the Trifacta node.

NOTE: The following security configuration requires export of and access to an SSL certificate in PEM file format for the HDFS instance. Creation and deployment of this certificate exceeds the scope of this document. Please see the documentation provided with your Hadoop distribution.

Certificates are commonly stored in Java keystores. They can be exported to PEM file format using the following command:

```
keytool -exportcert -rfc -alias <node_alias> -storepass <pwd> -keystore cacerts -file <filename.pem>
```

where:

- <pwd> is the keystore password.
- <filename.pem> is the output filename for the certificate.
- <node_alias> is the alias for the certificate in the keystore.

2. Place this generated certificate on the Trifacta node in a place where it is readable by the `[os.user (default=trifacta)]` user. The following location is suitable:

```
/opt/trifacta
```

3. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
4. Locate the following setting and enable it:

Setting	Description
"hdfs.webhdfs.ssl.enabled" : true,	Set to true to enable SSL communications with WebHDFS or (if enabled) HttpFS.

5. There is no need to update the port number. Port 14000 applies to HTTP and HTTPS.
6. **Security Level:** The level of security is determined by the following configuration options:

a. Secure without certificate export:

Setting	Description
<code>"hdfs.webhdfs.ssl.certificateValidationRequired": false,</code>	Set to <code>false</code> to disable use of trusted certificate validation.
<code>"hdfs.webhdfs.ssl.certificatePath": " ",</code>	Leave this value empty.

b. Secure with certificate:

Setting	Description
<code>"hdfs.webhdfs.ssl.certificateValidationRequired": false,</code>	Set to <code>true</code> to require SSL use of trusted certificate validation.
<code>"hdfs.webhdfs.ssl.certificatePath": " ",</code>	Configure the path on the Trifacta node to the location where you stored the certificate.

7. Save your changes and restart the platform.

Enable Integration with Compressed Clusters

Contents:

- *Prerequisites*
 - *Enable integration with compression*
 - *Specify codecs*
 - *Configure platform*
-

The Designer Cloud powered by Trifacta® platform can be configured to integrate with fully compressed Hadoop clusters. The following cluster compression methods are supported:

- Gzip
- Bzip2
- Snappy

Supported compressed running environments:

- Spark

For more information, see *Running Environment Options*.

Hadoop clusters can be configured to enable compression of intermediate and/or final output data by default. The settings that are usually used to do so can be found in `mapred-site.xml` and `core-site.xml`.

Prerequisites

NOTE: If you have not done so already, you must retrieve cluster configuration files and store them on the Trifacta node. For more information, see *Configure for Hadoop*.

Enable integration with compression

Steps:

1. Edit the local version of `mapred-site.xml`. This file is typically located in `/etc/conf/hadoop`.
2. Add the following properties:

```

<configuration>
  ...
  <property>
    <name>mapreduce.map.output.compress</name>
    <value>true</value>
  </property>

  <property>
    <name>mapreduce.map.output.compress.codec</name>
    <value>org.apache.hadoop.io.compress.SnappyCodec</value>
  </property>

  <property>
    <name>mapreduce.output.fileoutputformat.compress</name>
    <value>true</value>
  </property>

  <property>
    <name>mapreduce.output.fileoutputformat.compress.type</name>
    <value>BLOCK</value>
  </property>

  <property>
    <name>mapreduce.output.fileoutputformat.compress.codec</name>
    <value>org.apache.hadoop.io.compress.SnappyCodec</value>
  </property>
  ...
</configuration>

```

3. Save the file and complete the following steps.

Specify codecs

One or more compression/decompression methods (codecs) must be specified in `core-site.xml`.

Steps:

1. Edit the local version of `mapred-site.xml`. This file is typically located in `/etc/conf/hadoop`.
2. Specify the codecs to use in the `io.compression.codecs` property. Supported values:

Code	Value
Gzip	<code>org.apache.hadoop.io.compress.GzipCodec</code>
Bzip2	<code>org.apache.hadoop.io.compress.BZip2Codec</code>
Snappy	<code>org.apache.hadoop.io.compress.SnappyCodec</code>

3. In the following example, all three codecs have been specified:

```

<configuration>
  ...
  <property>
    <name>io.compression.codecs</name>
    <value>org.apache.hadoop.io.compress.GzipCodec,org.apache.hadoop.io.compress.BZip2Codec,org.apache.
hadoop.io.compress.SnappyCodec</value>
  </property>
  ...
</configuration>

```

4. Save the file.

Configure platform

Apply the following changes from within the application to enable the Designer Cloud powered by Trifacta platform to communicate with the compressed cluster.

Steps:

1. Login to the application.
2. In the Admin Settings page, set the following settings:

Setting	Description
<code>hadoopDefaultClusterCompression.enabled</code>	To enable integration with a compressed cluster, set this value to <code>true</code> .
<code>hadoopDefaultClusterCompression.compression</code>	Set this value to the type of compression applied on the cluster: <code>none</code> - (default) no cluster compression <code>gzip</code> <code>bzip2</code> <code>snappy</code>

3. Save your changes and restart the platform.

Enable Integration with Cluster High Availability

Contents:

- *Enable HttpFS*
- *Enable HA Service*
- *Configure HA for Individual Components*
- *Update Active Namenode*
- *Configure HA in a Kerberized Environment*
- *Platform Restart*

In a Hadoop cluster, **high availability** provides failover support for one or more configured nodes. This section describes how to enable the Designer Cloud powered by Trifacta® platform to utilize the highly available set of nodes within the Hadoop cluster. High availability enables access to each node of the cluster configured for it, in the event of machine crash or software installation or upgrade.

If high availability is enabled on the Hadoop cluster, you must enable it on the Designer Cloud powered by Trifacta platform, which prevents integration conflicts between Hadoop-specific components in the platform and their cluster equivalents.

Enable HttpFS

The WebHDFS service does not directly support high availability. You must enable the related HttpFS service and specify a WebHDFS namenode to point to the server hosting the HttpFS service.

NOTE: Avoid enabling the HttpFS service on the primary namenode of the cluster. In the event that the node hosting the namenode fails over, the HttpFS service is no longer available. You may be required to manually set the active namenode and restart the Designer Cloud powered by Trifacta platform.

For more information, see *Enable HttpFS*.

Enable HA Service

To begin, you must enable the High Availability service in the Designer Cloud powered by Trifacta platform for the supported components. In platform configuration, each component has its own feature flag under `feature.highAvailability`.

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

In the following example configuration, high availability has been disabled for `resourcemanagers` and enabled for `namenodes`:

NOTE: In almost all cases, `feature.highAvailability.resourceManager` should be set to `false`. For more information, see *Example - Configure resource manager* below.

```
"feature.highAvailability.namenode": true,  
"feature.highAvailability.resourceManager": false,
```

Configure HA for Individual Components

High availability in Hadoop works by specifying a **nameservice** for a highly available component and then enumerating the hosts and ports as **children** of that nameservice node. These values must be explicitly specified in the platform configuration.

Service names and child names should be specified in the file as they appear in the cluster's configuration files.

Example - Configure namenode

In the following example, the nameservice `namenodeha` provides high availability through two namenodes: `nn1` and `nn2`. In a high availability environment, these hosts are used for submitting jobs and writing data to HDFS.

```
"hdfs": {
  ...
  "highAvailability": {
    "serviceName": "namenodeha",
    "namenodes": {
      "nn1": {
        "host": "nn1.hadoop.mycompany.org",
        "port": 8020
      },
      "nn2": {
        "host": "nn2.hadoop.mycompany.org",
        "port": 8020
      }
    }
  }
},
```

Example - Configure resource manager

NOTE: Set `feature.highAvailability.resourcemanager=true` only if the cluster file `yarn-site.xml` enables `yarn.resourcemanager.hostname.highlyavailableyarn`. This setting enables the cluster high availability for resourcemanager.

Otherwise, set `feature.highAvailability.resourcemanager=false` for all environments. For HA environments, the resourcemanager hosts specified in the configuration below set the HA servers that are used by the Designer Cloud powered by Trifacta platform .

The following example specifies two failover nodes for the resource manager: `rm1` and `rm2`.

```

"yarn": {
  "resourceManagers": {
    "rm1": {
      "host": "rm1.yarn.mycompany.org",
      "port": 8032,
      "schedulerPort": 8030,
      "adminPort": 8033,
      "webappPort": 8042
    },
    "rm2": {
      "host": "rm2.yarn.mycompany.org",
      "port": 8032,
      "schedulerPort": 8030,
      "adminPort": 8033,
      "webappPort": 8042
    }
  }
}

```

Update Active Namenode

The active namenode used by the service must be configured explicitly. This value must be updated whenever the active namenode changes. Otherwise, HDFS becomes unavailable.

NOTE: If the HttpFS service has been tied to the primary namenode of the cluster and that node fails, this setting must be manually configured to the new node and the platform must be restarted. Avoid tying HttpFS to the primary namenode.

In this example, the active namenode has been set to the `nn1` value in the previous configuration:

```

"webhdfs": {
  "proxy": { ... },
  "version": "/webhdfs/v1",
  "port": 14000,
  "httpfs": true
},
...
"namenode": {
  "host": "nn1.hadoop.mycompany.org",
  "port": 8020
},
},

```

Configure HA in a Kerberized Environment

If you are enabling high availability in a Kerberized environment, additional configuration is required.

NOTE: WebHDFS does not support high availability/failover. You must enable HttpFS instead. For more information, see *Enable HttpFS*.

Steps:

1. If you have not done so already, acquire `httpfs-site.xml` from your Hadoop cluster.
2. Add the following settings to the file, replacing `[hadoop.user (default=trifacta)]` with the value appropriate for your environment:

```
<property>
  <name>https.authentication.type</name>
  <value>org.apache.hadoop.security.token.delegation.web.KerberosDelegationTokenAuthenticationHandler<
/property>
</property>
<property>
  <name>https.authentication.delegation-token.token-kind</name>
  <value>WEBHDFS delegation</value>
</property>
<property>
  <name>https.proxyuser.[hadoop.user].hosts</name>
  <value>*</value>
</property>
<property>
  <name>https.proxyuser.[hadoop.user].groups</name>
  <value>*</value>
</property>
```

3. The above change must also be applied to the `https-site.xml` configuration file for the cluster.

Save your changes and restart the platform.

Platform Restart

When high availability has been enabled, you must restart the platform from the command line. For more information, see *Start and Stop the Platform*.

Configure for Hive

Contents:

- *Prerequisites*
 - *Limitations*
 - *Configure for Hive*
 - *Enable appending to Hive tables without full permissions*
 - *Validate Configuration*
-

This section describes how to enable the Designer Cloud powered by Trifacta® platform to read sources in Hive and write results back to Hive.

- A Hive source is a single table in a selected Hive database.
- Apache Hive is a data warehouse system for managing queries against large datasets distributed across a Hadoop cluster. Queries are managed using HiveQL, a SQL-like querying language. See <https://hive.apache.org/>.
- The platform can publish results to Hive as part of any normal job or on an ad-hoc basis for supported output formats.
- Hive is also used by the Designer Cloud powered by Trifacta platform to publish metadata results. This capability shares the same configuration described below.
- ORC tables managed through Hive can be used as datasources for the platform.

Prerequisites

1. HiveServer2 and your Hive databases must already be installed in your Hadoop cluster.

NOTE: For JDBC interactions, the Designer Cloud powered by Trifacta platform supports HiveServer2 only.

2. You have verified that Hive is working correctly.
3. You have acquired and deployed the `hive-site.xml` configuration file into your Trifacta deployment. See *Configure for Hadoop*.

Limitations

1. Only one global connection to Hive is supported.
2. Changes to the underlying Hive schema are not picked up by the Designer Cloud powered by Trifacta platform and will break the source and datasets that use it.
3. During import, the JDBC data types from Hive are converted to Trifacta data types. When data is written back to Hive, the original Hive data types may not be preserved. For more information, see *Type Conversions*.
4. Publish to unmanaged tables in Hive is supported, except for the following actions:
 - a. Create table
 - b. Drop & load table
5. Publish to partitioned tables in Hive is supported.
 - a. The schema of the results and the partitioned table must be the same.
 - b. If they do not match, you may see an SchemaMismatched Exception error in the UI. You can try a drop and republish action on the data. However, the newly generated table does not have partitions.
 - c. For errors publishing to partitioned columns, additional information may be available in the logs.
6. Writing or publishing to ORC tables through Hive is not supported.

NOTE: Running user-defined functions for an external service, such as Hive, is not supported from within a recipe step. As a workaround, you may be able to execute recipes containing such external UDFs on the Photon running environment. Performance issues should be expected on larger datasets.

Configure for Hive

Hive user

The user with which Hive connects to read from the backend datastore should be a member of the user group [`hive.group` (default=`trifactausers`)] or whatever group is used to access storage from the Designer Cloud powered by Trifacta platform .

Verify that the Unix or LDAP group [`os.group` (default=`trifacta`)] has read access to the Hive warehouse directory.

Hive user for Spark:

NOTE: If you are executing jobs in the Spark running environment, additional permissions may be required. If the Hive source is a reference or references to files stored elsewhere in backend storage, the Hive user or its group must have read and execute permissions on the source directories or files.

Enable Data Service

In platform configuration, the Trifacta data service must be enabled. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Please verify the following:

```
"data-service.enabled": true,
```

Locate the Hive JDBC Jar

In platform configuration, you must verify that the following parameter is pointing to the proper location for the Hive JDBC JAR file. The example below identifies the location for Cloudera 6.2:

NOTE: This parameter varies for each supported distribution and version.

```
"data-service.hiveJdbcJar": "hadoop-deps/cdh-6.2/build/libs/cdh-6.2-hive-jdbc.jar",
```

Enable Hive Support for Spark Job Service

If you are using the Spark running environment for execution and profiling jobs, you must enable Hive support within the Spark Job Service configuration block.

NOTE: The Spark running environment is the default running environment. When this change is made, the platform requires that a valid `hive-site.xml` cluster configuration file be installed on the Trifacta node.

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 setting and verify that it is set to `true`:

```
"spark-job-service.enableHiveSupport" : true,
```

3. Modify the following parameter to point to the location where Hive dependencies are installed. This example points to the location for Cloudera 6.2:

NOTE: This parameter value is distribution-specific. Please update based on your specific distribution.

```
"spark-job-service.hiveDependenciesLocation": "%(topOfTree)s/hadoop-deps/cdh-6.2/build/libs",
```

4. Save your changes.

Enable Hive Database Access for Spark Job Service

The Spark Job Service requires read access to the Hive databases. Please verify that the Spark user can access the required Hive databases and tables.

For more information, please contact your Hive administrator.

Configure managed table format

The Designer Cloud powered by Trifacta platform publishes to Hive using managed tables. When writing to Hive, the platform pushes to an externally staged table. Then, from this staging table, the platform selects and inserts into a managed table.

By default, the platform publishes to managed tables in Parquet format. As needed, you can apply the following values into platform configuration to change the format to which the platform writes when publishing a managed table:

- PARQUET (default)
- AVRO

To change the format, please modify the following parameter.

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 modify it using one of the above values:

```
"data-service.hiveManagedTableFormat": "PARQUET",
```

3. Save your changes and restart the platform.

Create Hive Connection

NOTE: High availability for Hive is supported through configuration of the Hive connection.

For more information, see *Hive Connections*.

Optional Configuration

Depending on your Hadoop environment, you may need to perform additional configuration to enable connectivity with your Hadoop cluster.

Additional Configuration for Secure Environments

For secure impersonation

NOTE: You should have already configured the Designer Cloud powered by Trifacta platform to use secure impersonation. For more information on basic configuration, see *Configure for Secure Impersonation*.

You must add the Hive principal value to your Hive connection. Add the following principal value to the Connect String Options textbox.

```
"connectStrOpts": ";principal=<principal_value>",
```

For Kerberos with secure impersonation

NOTE: You should have already enabled basic Kerberos integration. For more information, see *Configure for Kerberos Integration*.

NOTE: If you are enabling Hive in a Kerberized environment, you must also enable secure impersonation. When connecting to Hive, Kerberos without secure impersonation is not supported. You should have already configured the Designer Cloud powered by Trifacta platform to use secure impersonation. For more information on basic configuration, see *Configure for Secure Impersonation*.

Additional Configuration for Sentry

The Designer Cloud powered by Trifacta platform can be configured to use Sentry to authorize access to Hive. See *Configure for Hive with Sentry*.

Enable appending to Hive tables without full permissions

Optionally, you can enable users to publish to Hive tables for which they do not have CREATE or DROP permissions.

If they have read (SELECT) and append (INSERT) permissions on a Hive schema, they can be permitted to append to the production schema using a separate schema that matches the production one. The Designer Cloud powered by Trifacta platform does the following:

1. CREATE a staging table in the schema specified in the User Profile.

NOTE: These schemas must be created. Users must be given CREATE and DROP permissions on them.

2. INSERT the output data into the staging table.
3. Via INSERT, copy over data from the staging table to the production schema, effectively performing an append on the production table.

Steps:

NOTE: This feature must be enabled.

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 setting:

```
"feature.showHiveStagingDB": true,
```

3. Save your changes and restart the platform.

Use:

1. Staging schemas must be created in Hive.
2. Each user must insert the name of the staging schema in their user profile once. For more information, see *User Profile Page*.
3. When users generate results to Hive, they choose to publish to the production schema as an `append` operation.
 - a. For more information, see *Run Job Page*.
 - b. For more information, see *Publishing Dialog*.
4. Because this feature is enabled, the platform uses the specified staging schema and publishing mechanism to perform the `append` to the production schema.

Validate Configuration

NOTE: The platform cannot publish to a default database in Hive that is empty. Please create at least one table in your default database.

Build example Hive dataset

Steps:

1. Download and unzip the following dataset: *Dataset-HiveExampleData*.
2. Store the dataset in the following example directory:

```
/tmp/hiveTest_5mb
```

3. Use the following command to create your table:

```
create table test (name string, id bigint, id2 bigint, randomName string, description string, dob string, title string, corp string, fixedOne bigint, fixedTwo int) row format delimited fields terminated by ',' STORED AS TEXTFILE;
```

4. Add the example dataset to the above test table:

```
load data local inpath '/tmp/hiveTest_5mb' into table test;
```

Check basic connectivity

Steps:

1. After restart, login to the Designer Cloud application . See *Login*.

2. If the platform is able to connect to Hive, you should see a Hive tab in the Import Data page. Create a new dataset and verify that the data from the Hive data has been ingested in the Transformer page.
3. If not, please check `/opt/trifacta/logs/data-service.log` for errors.
4. For more information, see *Verify Operations*.

Configure for Hive with Sentry

Contents:

- *Prerequisites*
- *Secure Impersonation with Designer Cloud powered by Trifacta platform and Hive with Sentry*
- *Users and Groups for Sentry*
- *Configuration*
- *Basic Authentication*
- *Verify Operations*
- *Troubleshooting*

This section describes how to ensure that the Designer Cloud powered by Trifacta® platform is configured correctly to connect to Hive when Sentry is enabled for Hive. Sentry provides role-based authorization for Hive and other Hadoop components on the Cloudera platform.

- For more information, see <http://www.cloudera.com>.

Prerequisites

Before you begin, please verify that your enterprise has deployed both Hive and Sentry according to recommended configuration practices. For more information, please consult the documentation that was provided with your Hadoop distribution.

NOTE: Before you begin, you must integrate the Designer Cloud powered by Trifacta platform with Hive. See *Configure for Hive*.

1. Enable the Sentry Service. Then, configure Hive to use the Sentry Service. See http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/sg_sentry_service_config.html#concept_amg_2l2_xq_unique_2
2. (recommended) Enable secure impersonation. See below.

Secure Impersonation with Designer Cloud powered by Trifacta platform and Hive with Sentry

Secure impersonation ensures consistent and easily traceable security access to the data stored within your Hadoop cluster.

NOTE: Although not required, secure impersonation is highly recommended for connecting the platform with Hive.

The Designer Cloud powered by Trifacta platform requires the following additional configuration changes to maintain secure impersonation and work with Hive data:

1. Enable the platform with secure impersonation. See *Configure for Secure Impersonation* for details.
2. Give the local Hive user access to the Unix or LDAP group [`ldap.group (default=trifactausers)`].
3. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
4. Set the following `umask` in `trifacta-conf.json`:

```
"hdfs.permissions.userUmask" = 027,
```

5. Verify that the Unix or LDAP group [`ldap.group`] has read access to the hive warehouse directory as specified in the following section. For more information, see

http://www.cloudera.com/content/www/en-us/documentation/enterprise/latest/topics/sg_sentry_service_config.html#concept_mlr_qxm_vq_unique_1

6. (Optional) Configuring Sentry to sync HDFS permissions will maintain user level access control for the underlying data files. For more information, see http://www.cloudera.com/documentation/enterprise/5-4-x/topics/sg_hdfs_sentry_sync.html

Users and Groups for Sentry

For Sentry, the following definitions and relationships apply.

Definition	Description
User	Individual account, as identified by the underlying authentication system
Group	A set of users maintained by the authentication system
Role	A set of privileges stored as a template to combine multiple access rules
Privilege	An instruction or rule allowing access to an object. Examples of Privileges include access to databases, tables, and the operations that can be executed.

In Sentry:

- Privileges can only be granted to Roles.
- A Group can be assigned to one or more Roles.
 - Users are assigned to a Group through the underlying authentication mechanism (e.g. operating system or LDAP).

Configuration

NOTE: Before you begin, you should determine the privileges that must be granted to Trifacta users based on your environment and needs.

NOTE: If you are publishing back to Hive, please verify that one of the following is enabled:

1. The `hive:hive` user has `rwX` permissions on the publish directory.
2. The Hive users must be members of the group that owns the directory.

Steps:

1. Start Beeline as an administrative user.
2. Create a role for users of the Designer Cloud powered by Trifacta platform :

```
CREATE ROLE trifactaUserRole;
```

3. Grant that role to the [`ldap.group` (default=`trifactausers`)] group associated with the platform:

```
GRANT ROLE trifactaUserRole TO GROUP trifacta;
```

4. Grant all privileges to this role for the filesystem area under which platform output is generated. The full URI is required. Example:

NOTE: Modify the grants as needed for your environment.

```
GRANT ALL ON URI 'hdfs://domain_example:8020/trifacta/queryResults/user1@example.com/' to ROLE
trifactaUserRole;
```

NOTE: If the above URI changes, the above grant must be reapplied to the new URI.

Basic Authentication

When the Designer Cloud powered by Trifacta platform is enabled with secure impersonation and submits requests to Hive, the following steps occur:

1. The platform authenticates as the [hadoop.user] user through Kerberos.
2. The Hive server authorizes access to the underlying table through Sentry as the Hadoop principal user assigned to the Trifacta user.

NOTE: This Hadoop principal is the user that should be configured with appropriate privileges and roles in Sentry.

3. The Hive server executes access to the physical data file on HDFS as the Unix or LDAP user `hive`, which should be part of the designated group [hadoop.group (default=trifactausers)].

NOTE: Since Sentry assigns privileges and roles to Unix groups, a common practice is to assign the Hadoop principal users (used by Trifacta users) to dedicated Unix groups that are **separate** from the Unix group [os.group (default=trifacta)] to use within Sentry. Sentry should not grant any privileges and roles to the Unix group `trifacta`.

NOTE: In UNIX environments, usernames and group names are case-sensitive. Please verify that you are using the case-sensitive names for users and groups in your Hadoop configuration and Trifacta configuration file.

Verify Operations

After you have completed your configuration changes, you should restart the platform. See *Start and Stop the Platform*.

To verify platform operations, run a simple job. For more information, see *Verify Operations*.

Troubleshooting

Cannot publish to Hive in a Kerberized environment with secure impersonation using Sentry

If you have deployed Sentry to manage access to a Kerberized environment using secure impersonation, you may encounter the following error when trying to write your results back to the Hadoop cluster:

NOTE: This issue is known to appear in Cloudera 5.7. It may not appear in later releases.

```
2015-09-02 20:49:54.111Z - WARN : com.trifacta.dataservice.Controller : Bad Request: org.springframework.
jdbc.BadSqlGrammarException: StatementCallback; bad SQL grammar [CREATE TABLE `test_trifacta` ROW FORMAT
SERDE 'org.apache.hadoop.hive.serde2.avro.AvroSerDe' STORED AS INPUTFORMAT 'org.apache.hadoop.hive ql.io.avro.
AvroContainerInputFormat' OUTPUTFORMAT 'org.apache.hadoop.hive.ql.io.avro.AvroContainerOutputFormat' LOCATION
'hdfs://domain_example:8020/trifacta/queryResults/user1@example.com/test/143/original_98.avro' TBLPROPERTIES
```

```
('avro.schema.literal'='{ "type": "record", "name": "GenericTrifactaRecord", "fields": [{ "name": "name", "type": ["null", "string"] }, { "name": "id", "type": ["null", "string"] }, { "name": "id2", "type": ["null", "long"] }, { "name": "randomname", "type": ["null", "string"] }, { "name": "description", "type": ["null", "string"] }, { "name": "dob", "type": ["null", "string"] }, { "name": "title", "type": ["null", "string"] }, { "name": "corp", "type": ["null", "string"] }, { "name": "fixedone", "type": ["null", "long"] }, { "name": "fixedtwo", "type": ["null", "long"] } ] }'); nested exception is org.apache.hive.service.cli.HiveSQLException: Error while compiling statement: FAILED: SemanticException No valid privileges
```

```
Required privileges for this query: Server=server1->URI=hdfs://domain_example:8020/trifacta/queryResults/user1@example.com/test/143/original_98.avro->action=*
```

In this case, Sentry is failing to validate the URI permissions to allow the user (`user1@example.com`) to access the HDFS path, as the permissions have not been specifically granted to the required role. Sentry queries for authorization, fails, and throws the above exception.

The solution is to grant all access privileges for the user's Sentry role to Trifacta results directory for the target user. In the following example, access is granted to the `role2` role:

```
GRANT ALL ON URI 'hdfs://domain_example:8020/trifacta/queryResults/user1@example.com/' to ROLE role2;
```

Since permissions in Sentry are recursive through the directories, the target directory for the specific job is covered. For more information on Sentry permissions, See Terminologies section in http://www.cloudera.com/content/cloudera/en/documentation/core/latest/topics/cdh_sg_sentry.html.

Hive Connections

Contents:

- *Enable*
 - *Limitations*
 - *Create Encryption Key File*
 - *Create Hive Connection*
 - *Create through application*
 - *Create via API*
 - *Additional Configuration Options*
 - *Configure for HTTP*
 - *Configure for SASL*
 - *Configure for Kerberos*
 - *Run Hive jobs on a specific YARN queue*
 - *Using Hive Connections*
 - *Uses of Hive*
 - *Before you begin using Hive*
 - *Reading partitioned data*
 - *Storing data in Hive*
 - *Reading from Hive*
 - *Writing to Hive*
 - *SQL Syntax*
 - *Reference*
-

This section describes the basics of creating a connection to Hive from Designer Cloud Powered by Trifacta® Enterprise Edition and modifying it based on your cluster environment.

For more information on supported versions of Hive , see *Configure for Hive*.

Enable

After you have configured your HDFS base storage layer, you can enable and configure your connection to the associated instance of Hive.

NOTE: Additional configuration is required.

For more information, see *Configure for Hive*.

Limitations

NOTE: The platform supports a single, global connection to Hive . All users must use this connection.

Create Encryption Key File

Before you create your Hive connection, you must create and reference an encryption key file. If you have not created one already, see *Create Encryption Key File*.

Create Hive Connection

You can create the connection through the application or the APIs.

NOTE: The following configuration methods apply to creation of an insecure connection to a Hive 1.x instance. If you are applying security, using HTTP, or connecting to Hive 2.x, additional configuration is required. Before creating these connections, please review the Additional Configuration Options section below.

Create through application

A Trifacta administrator can create the Hive connection through the application.

Steps:

1. Login to the application.
2. In the menu, select **User menu > Preferences > Connections**.
3. In the Create Connection page, click the Hive connection card.
4. Specify the properties for your Hive database.
 - a. Before you create the connection, you should review the rest of this section to verify that you are using the proper connection string options for the Hive connection to work in your environment.

NOTE: If secure impersonation is enabled on your cluster, you must include the Hive principal as part of your connection string. For more information, see *Configure for Hive*.

- b. For more information, see *Create Connection Window*.
5. Click **Save**.

Create via API

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/createConnectionAPI>:
API Reference

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

Additional Configuration Options

Configure for HTTP

By default, the Designer Cloud powered by Trifacta platform utilizes TCP to connect to Hive. As needed, the platform can be configured to interact with Hive Server2 over HTTP.

NOTE: HTTP connections are enabled by inserting additional parameters as part of the connection string. In some environments, the JDBC drive may automatically insert these parameters, based on configuration in `hive-site.xml`. If these parameters are not inserted, the Designer Cloud powered by Trifacta platform does not need additional configuration. For more information, see <https://cwiki.apache.org/confluence/display/Hive/HiveServer2+Clients#HiveServer2Clients-ConnectionURLWhenHiveServer2IsRunninginHTTPMode>.

NOTE: If you are planning to use SSL, additional configuration is required. See below.

Steps:

To enable the Designer Cloud powered by Trifacta platform to use HTTP when connecting to Hive, please do the following.

1. Create a params file for your Hive connection. This file must contain at least the following entry:

```
"connectStrOpts": ";transportMode=http;httpPath=cliservice"
```

2. Save the file.
3. In your command, set the connection port value to 10001.
4. Execute the command.
5. Test the Hive connection.
6. If it's not working, delete the connection and try again.

Configure for SASL

NOTE: Cloudera supports an additional method for communicating over SSL with Hive . For more information on how to identify the method used by your Cloudera cluster, see *Configure for Cloudera*.

The steps below describe how to enable the SASL-QOP method of SASL (Simple Authentication and Security Layer) communications. To enable, you must add an additional connection string parameter.

Steps:

1. Create or edit the params file for your connection to Hive. This file must include the following setting for `connectStrOpts`:

```
{
  "connectStrOpts": ";sasl.qop=auth-conf",
  "defaultDatabase": "default",
  "jdbc": "hive2"
}
```

2. Save the file.
3. In your command, set the connection port value to 10001.
4. Execute the command.
5. Test the Hive connection.
6. If it's not working, delete the connection and try again.

Configure for Kerberos

For a Kerberos-enabled cluster, you must include the Kerberos principal value as part of the Hive connection string options.

Please complete the following steps to update the Kerberos principal value for Hive into your Hive connection.

Steps:

1. Retrieve the Kerberos principal value for your deployment.
2. Apply the Kerberos principal to your Hive connection. Add the Hive principal in the above format to the Connect String Options in your Hive connection. See *Create Connection Window*.
3. Save the file.
4. Test the connection.

Run Hive jobs on a specific YARN queue

If needed, you can route any Spark jobs sourced from Hive to a specific YARN job queue.

Steps:

1. In your Hive connection, add the following option to your Connection String Options (`connectStrOpts`) in your Hive connection:

```
"connectStrOpts": ";sess_var_list?mapred.job.queue.name=<your_queue>",
```

where:

<your_queue> is the name of the YARN job queue.

2. Save the change.
3. Test the connection.
4. Run a job, and verify that it is routed to the appropriate YARN job queue.

Using Hive Connections

This section describes how you interact through the Designer Cloud powered by Trifacta® platform with your Hive data warehouse.

- Hive is an open-source, scalable data warehouse built on top of the Hadoop infrastructure to enable SQL-like access to a datastore where processing is converted to Hadoop map/reduce tasks. Hive users can interact directly with the databases and tables using HiveQL, a querying language similar to SQL. For more information, see https://en.wikipedia.org/wiki/Apache_Hive.
- Hive sources can be stored as Hive database tables and views, or you can select or create tables to which to publish job results.

Uses of Hive

The Designer Cloud powered by Trifacta platform can use Hive for the following tasks:

1. Create datasets by reading from Hive tables.
2. Write data to Hive.

Before you begin using Hive

- **Read Access:** Your Hadoop administrator must configure read permissions to Hive databases.
 - Your Hadoop administrator should provide a database table or tables for data upload to your Hive datastore.
- **Write Access:** You can write jobs directly to Hive or ad-hoc publish jobs results to Hive at a later time. See Writing to Hive below.

Secure Access

Depending on the security features you've enabled, the technical methods by which Trifacta users access Hive may vary. For more information, see *Configure Hadoop Authentication*.

Reading partitioned data

The Designer Cloud powered by Trifacta platform can read in partitioned tables. However, it cannot read individual partitions of partitioned tables.

NOTE: If you are using custom SQL to read from a partitioned Hive table, performance may be impacted.

Tip: If you are reading data from a partitioned table, one of your early recipe steps in the Transformer page should filter out the unneeded table data so that you are reading only the records of the individual partition.

Storing data in Hive

Your Hadoop administrator should provide datasets or locations and access for storing datasets within Hive.

- Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

NOTE: The Designer Cloud powered by Trifacta platform does not modify source data in Hive. Datasets sourced from Hive are read without modification from their source locations.

Reading from Hive

You can create a Trifacta dataset from a table or view stored in Hive.

For more information on how data types are imported from Hive, see *Hive Data Type Conversions*.

Notes on reading from Hive views using custom SQL

If you have enabled custom SQL and are reading data from a Hive view, nested functions are written to a temporary filename, unless they are explicitly aliased.

Tip: If your custom SQL uses nested functions, you should create an explicit alias from the results. Otherwise, the job is likely to fail.

Problematic example:

```
SELECT
  UPPER(`t1`.`column1`),
  TRIM(`t1`.`column2`),...
```

When these are read from a Hive view, the temporary column names are: `_c0`, `_c1`, etc. During job execution, Spark ignores the `column1` and `column2` reference.

Improved Example:

```
SELECT
  UPPER(`t1`.`column1`) as col1,
  TRIM(`t1`.`column2`) as col2,...
```

In this improved example, the two Hive view columns are aliased to the explicit column names, which are correctly interpreted and used by the Spark running environment during job execution.

Writing to Hive

You can write data back to Hive using one of the following methods:

NOTE: You cannot publish to a Hive database that is empty. The database must contain at least one table.

NOTE: If you are writing to unmanaged tables in Hive, create and drop & load actions are not supported.

- Job results can be written directly to Hive as part of the normal job execution. Create a new publishing action to write to Hive. See *Run Job Page*.
- As needed, you can publish results to Hive for previously executed jobs. See *Publishing Dialog*.
- For more information on how data is converted to Hive, see *Hive Data Type Conversions*.

SQL Syntax

The following syntax requirements apply to this connection.

Object delimiter: backtick

Example syntax:

```
SELECT `column1`,`column2` FROM `databaseName`.`tableName`;
```

For more information on SQL in general, see *Supported SQL Syntax*.

Reference

Create New Connection:

NOTE: A single public Hive connection is supported.

Configure for KMS

Hadoop KMS is a key management system that enables encrypted transport to and from the Hadoop cluster. This section describes how to configure the Designer Cloud powered by Trifacta® platform for integration with KMS.

NOTE: The Designer Cloud powered by Trifacta platform supports encryption at rest only through the KMS solution provided with the Hadoop distribution. Generic encryption at rest is not supported.

NOTE: If KMS is enabled on the cluster, you must configure KMS for the Designer Cloud powered by Trifacta platform regardless of other security features enabled on the cluster.

- For more information on KMS, see <https://hadoop.apache.org/docs/stable/hadoop-kms/index.html>.

NOTE: The required configuration for integrating with each Hadoop distribution may vary. Please be sure to review the details.

Prerequisites

1. You have installed the Trifacta software. See *Install*.
2. You have performed the basic configuration steps for Hadoop. See *Configure for Hadoop*.
3. You have enabled any required secure authentication services.
 - a. See *Configure for Kerberos Integration*.
 - b. See *Configure for Secure Impersonation*.

Configure by Distribution Type

KMS is a cluster-wide configuration. If you are enabling Kerberos, secure impersonation, or encryption at rest on the cluster, you must perform the KMS site configuration changes in the pages for your specific Hadoop distribution.

Cloudera/Sentry: See *Configure for KMS for Sentry*.

Configure for KMS for Sentry

Contents:

- *Configure Hadoop Cluster*
 - *Enable HDFS Encryption*
 - *Java KMS Configuration*
 - *Java KeyStore KMS Configuration*
 - *HDFS Configuration*
- *Validate*

This section describes how to configure the Designer Cloud powered by Trifacta® platform for integration with KMS system for Cloudera. It assumes that access to the cluster is gated by Sentry.

Before you begin, please verify the prerequisites. See *Configure for KMS*.

Configure Hadoop Cluster

NOTE: These changes should be applied through the management console for the Hadoop cluster before pushing the client configuration files to the nodes of the cluster.

In the following sections:

- [hadoop.user (default=trifacta)] - the userID accessing the cluster component
- [hadoop.group (default=trifactausers)] -the appropriate group of user accessing the cluster component

Enable HDFS Encryption

On the Cloudera cluster, you may enable HDFS encryption using a designated Java KeyStore. For more information, see

http://www.cloudera.com/documentation/enterprise/latest/topics/sg_hdfs_encryption_wizard.html?scroll=concept_n2p_5vq_vt#concept_fcq_phr_wt_unique_1

Java KMS Configuration

Additional configuration for the Java KMS is required. See

http://www.cloudera.com/documentation/enterprise/latest/topics/cdh_sg_kms.html.

Java KeyStore KMS Configuration

In the `kms-site.xml` configuration file, please locate the following properties:

NOTE: If you have deployed Cloudera Manager for your cluster, do not modify these properties in the file. Make any modifications through the Cloudera Manager console.

```
<property>
  <name>hadoop.kms.authentication.kerberos.keytab</name>
  <value>${user.home}/kms.keytab</value>
</property>
```

In Cloudera Manager, you may wish to change the following safety value value. Navigate to **KMS service > Configuration > Advanced > Key Management Server Proxy Advanced Configuration Snippet (Safety Valve) for kms-site.xml**. Modify the following:

```
<property>
  <name>hadoop.kms.aggregation.delay.ms</name>
  <value>10000</value>
</property>
```

In the `kms-site.xml` file, insert the following properties, which are required properties for the Key Management Server Advanced Configuration safety value:

```
<property>
  <name>hadoop.kms.authentication.kerberos.principal</name>
  <value>*</value>
</property>
<property>
  <name>hadoop.kms.proxyuser.[hadoop.user].groups</name>
  <value>[hadoop.group]</value>
</property>
<property>
  <name>hadoop.kms.proxyuser.[hadoop.user].hosts</name>
  <value>*</value>
</property>
```

HDFS Configuration

In `httpfs-site.xml`, please insert the following properties, which are the safety value for HttpFS Advanced Configuration:

```
<property>
  <name>httpfs.proxyuser.[hadoop.user].groups</name>
  <value>[hadoop.group]</value>
</property>
<property>
  <name>httpfs.proxyuser.[hadoop.user].hosts</name>
  <value>*</value>
</property>
```

Save the files.

Validate

After the configuration is complete, you can try to import a dataset from a source stored in a cluster location managed by KMS, assuming that any required authentication configuration has been completed. See *Import Data Page*.

For more information, see *Configure Hadoop Authentication*.

Configure for AWS

Contents:

- *Internet Access*
- *Database Installation*
- *Base AWS Configuration*
 - *Base Storage Layer*
 - *Configure AWS Region*
- *AWS Authentication*
- *AWS Storage*
 - *S3 Layer*
 - *S3 Connections*
 - *Redshift Connections*
 - *Snowflake Connections*
- *Supported AWS Clusters*
 - *EMR*

This documentation applies to installation from a supported Marketplace. Please use the installation instructions provided with your deployment.

If you are installing or upgrading a Marketplace deployment, please use the available PDF content. You must use the install and configuration PDF available through the Marketplace listing.

The Designer Cloud powered by Trifacta® platform can be hosted within Amazon and supports integrations with multiple services from Amazon Web Services, including combinations of services for hybrid deployments. This section provides an overview of the integration options, as well as links to related configuration topics.

For an overview of AWS deployment scenarios, see *Supported Deployment Scenarios for AWS*.

Internet Access

From AWS, the Designer Cloud powered by Trifacta platform requires Internet access for the following services:

NOTE: Depending on your AWS deployment, some of these services may not be required.

- AWS S3
- Key Management System [KMS] (if sse-kms server side encryption is enabled)
- Secure Token Service [STS] (if temporary credential provider is used)
- EMR (if integration with EMR cluster is enabled)

NOTE: If the Designer Cloud powered by Trifacta platform is hosted in a VPC where Internet access is restricted, access to S3, KMS and STS services must be provided by creating a VPC endpoint. If the platform is accessing an EMR cluster, a proxy server can be configured to provide access to the AWS ElasticMapReduce regional endpoint.

Database Installation

The following database scenarios are supported.

Database Host	Description
Cluster node	By default, the Trifacta databases are installed on PostgreSQL instances in the Trifacta node or another accessible node in the enterprise environment. For more information, see <i>Install Databases</i> .
Amazon RDS	For Amazon-based installations, you can install the Trifacta databases on PostgreSQL instances on Amazon RDS. For more information, see <i>Install Databases on Amazon RDS</i> .

Base AWS Configuration

The following configuration topics apply to AWS in general.

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

Base Storage Layer

NOTE: The base storage layer must be set during initial configuration and cannot be modified after it is set.

S3: Most of these integrations require use of S3 as the base storage layer, which means that data uploads, default location of writing results, and sample generation all occur on S3. When base storage layer is set to S3, the Designer Cloud powered by Trifacta platform can:

- read and write to S3
- read and write to Redshift
- connect to an EMR cluster

HDFS: In on-premises installations, it is possible to use S3 as a read-only option for a Hadoop-based cluster when the base storage layer is HDFS. You can configure the platform to read from and write to S3 buckets during job execution and sampling. For more information, see *S3 Access*.

For more information on setting the base storage layer, see *Set Base Storage Layer*.

For more information, see *Storage Deployment Options*.

Configure AWS Region

For Amazon integrations, you can configure the Trifacta node to connect to Amazon datastores located in different regions.

NOTE: This configuration is required under any of the following deployment conditions:

1. The Trifacta node is installed on-premises, and you are integrating with Amazon resources.
2. The EC2 instance hosting the Trifacta node is located in a different AWS region than your Amazon datastores.
3. The Trifacta node or the EC2 instance does not have access to `s3.amazonaws.com`.

1. In the AWS console, please identify the location of your datastores in other regions. For more information, see the Amazon documentation.
2. Login to the Designer Cloud application .
3. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
4. Set the value of the following property to the region where your S3 datastores are located:

```
aws.region
```

If the above value is not set, then the Designer Cloud powered by Trifacta platform attempts to infer the region based on default S3 bucket location.

5. Save your changes.

Configure region for AWS GovCloud

If your instance of the Designer Cloud powered by Trifacta platform is deployed in the AWS GovCloud, additional configuration is required.

NOTE: GovCloud authentication is completely isolated from Amazon.com. Key-secret combinations for AWS do not apply to AWS GovCloud.

For more information, see <https://docs.aws.amazon.com/govcloud-us/latest/UserGuide/govcloud-differences.html>.

NOTE: In AWS GovCloud, the AWS S3 endpoint must be configured in the private subnet or VPC to be able to make outbound requests to S3 resources.

You must configure the region and endpoint settings to communicate with GovCloud S3 resources.

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. In the Admin Settings page, specify the following parameters as follows:

Parameter	Description
<code>aws.region</code>	Specify the AWS GovCloud region where your S3 resources are located.
<code>aws.s3.endpoint</code>	Specify the private subnet or VPC endpoint through which the Designer Cloud powered by Trifacta platform can reach S3 resources.

3. Save your changes.
4. Login to the Trifacta node as an administrator.
5. Edit the following file:

```
/opt/trifacta/conf/env.sh
```

6. Add the following line:

```
AWS_REGION=<aws.region>
```

where:

<aws.region> is the value you inserted for the AWS GovCloud region in the Admin Settings page.

7. Save the file.
8. If the following does not apply, restart the platform.

Accessing AWS secret region using an emulation platform:

1. If you are connecting to the AWS secret region using an emulation platform with custom TLS certificates, you must install and reference a valid TLS/SSL certificate for the emulation platform.
 - a. To the `env.sh` file, please add the following line:

```
NODE_EXTRA_CA_CERTS=/home/trifacta/ca-chain.cert.pem
```

NOTE: The above certificate must be a valid TLS/SSL certificate for the emulation platform.

- b. For AWS requests from Java services, you must add the custom CA certificate to the CA certificates for the system. For more information, please see the documentation for your emulation platform.
2. Save your changes.
3. Restart the platform.

AWS Authentication

For more information, see *Configure for AWS Authentication*.

AWS Storage

S3 Layer

To integrate with S3, additional configuration is required. See *S3 Access*.

S3 Connections

Users can create secondary connections to specific S3 buckets. For more information, see *External S3 Connections*.

Redshift Connections

You can create connections to one or more Redshift databases, from which you can read database sources and to which you can write job results. Samples are still generated on S3.

For more information, see *Amazon Redshift Connections*.

Snowflake Connections

Through your AWS deployment, you can access your Snowflake databases. For more information, see *Snowflake Connections*.

Supported AWS Clusters

EMR

When Designer Cloud Powered by Trifacta Enterprise Edition is installed through AWS, you can integrate with an EMR cluster for Spark-based job execution. For more information, see *Configure for EMR*.

Configure for AWS Authentication

Contents:

- *Overview*
 - *AWS authentication mode*
 - *AWS credential provider type*
 - *EMR authentication mode*
 - *SSO support*
- *Basic Configuration*
 - *Before you configure*
 - *Configure AWS mode and credential provider*
 - *Configure authentication for EMR*
- *AWS Authentication Topics*

This section provides high-level information on the different configuration methods by which Designer Cloud Powered by Trifacta® Enterprise Edition authenticates to AWS resources. From here, you can jump to:

- **Configuration Tasks:** Step-by-step tasks for configuring the product for a specific AWS authentication method.
- **AWS Authentication Topics:** Detailed documentation on various authentication methods.

Overview

Designer Cloud Powered by Trifacta Enterprise Edition provides the following methods of authenticating to AWS.

AWS authentication mode

When connecting to AWS, Designer Cloud Powered by Trifacta Enterprise Edition supports the following basic authentication modes.

AWS Mode	Description
System	All users of the workspace use the same set of credentials to authenticate to AWS. Access to AWS resources is managed through a single, system account. The type of account that you specify is based on the credential provider selected below.
User	Each user of the workspace uses a personal set of credentials to authenticate. Authentication must be specified for individual users. Tip: Although the steps are more involved to set up and manage per-user authentication, this method provides superior security, data governance, and overall management.

AWS credential provider type

For each access mode, Designer Cloud Powered by Trifacta Enterprise Edition supports the following types of credential providers:

Credential Provider Type	Description
default	Credentials are provided in the form of AWS key/secret combinations.
instance	Credentials are provided in the form of roles associated with the EC2 instance for the product.
temporary	Credentials are provided in the form of IAM roles.

Tip: This method is recommended.

EMR authentication mode

Similar to general AWS access, Designer Cloud Powered by Trifacta Enterprise Edition supports the following modes for providing credentials for EMR for running jobs.

- **EMR system mode:** All workspace users use the same AWS key/secret combination to access EMR.
- **EMR user mode:** Each workspace user submits a personal set of credentials to access EMR.

The following table illustrates how AWS mode and EMR mode work together:

AWS mode	System	User
EMR mode		
System	AWS and EMR use a single key-secret combination.	<ul style="list-style-type: none">• AWS access uses a single key-secret combination.• EMR access is governed by per-user credentials. Per-user credentials can be provided from one of several different providers.
User	Not supported.	AWS and EMR use the same per-user credentials for access. Per-user credentials can be provided from one of several different providers.

SSO support

Designer Cloud Powered by Trifacta Enterprise Edition supports integration with a SAML SSO credential provider for AWS resources. Additional details are provided below.

Basic Configuration

Before you configure

Tip: If you prefer, you can review the available authentication tasks to see if one matches your environment.

Before you configure the product, please verify the following:

1. You have chosen the AWS mode to use.
2. You have chosen the credential provider type to use.
3. You have defined and enabled the credentials required to support the above configuration choices.

Configure AWS mode and credential provider

The following table breaks down the configuration of credentials based on the credential type and the AWS mode based on the setting of two key parameters. These two basic parameters can be configured at the same time.

- credential provider - source of credentials: platform (default), instance (EC2 instance only), or temporary
- AWS mode - the method of authentication from platform to AWS: system-wide or by-user

NOTE: If you are using AWS user mode or SSO, additional configuration is required.

To configure:

1. Login to the Designer Cloud 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. Apply the following configuration to the platform.

AWS Mode	System	User
Credential Provider		
Default	One system-wide key/secret combo is inserted in the platform for use	Each user provides key/secret combo.
	Config: <pre>"aws.credentialProvider": "default", "aws.mode": "system", "aws.s3.key": <key>, "aws.s3.secret": <secret>,"</pre>	Config: <pre>"aws.credentialProvider": "default", "aws.mode": "user",</pre> User: <i>Configure Your Access to S3</i>
Instance	Platform uses roles from the EC2 instance where the platform is running.	Not supported.
	Config: <pre>"aws.credentialProvider": "instance", "aws.mode": "system",</pre>	Config: n/a
Temporary	Temporary credentials are issued based on system IAM roles.	Per-user authentication when using IAM role.
	Config: <pre>"aws.credentialProvider": "temporary", "aws.mode": "system", "aws.systemIAMRole": "<IAMRole>,"</pre>	Config: <pre>"aws.credentialProvider": "temporary", "aws.mode": "user",</pre>

Default credential provider

Whether the AWS access mode is set to system or user, the default credential provider for AWS and S3 resources is the Designer Cloud powered by Trifacta platform .

Mode	Description	Configuration
<pre>"aws.mode": "system",</pre>	A single AWS Key and Secret is inserted into platform configuration. This account is used to access all resources and must have the appropriate permissions to do so.	<pre>"aws.s3.key": "<your_key_value>", "aws.s3.secret": "<your_key_value>,"</pre>
<pre>"aws.mode": "user",</pre>	Each user must specify an AWS Key and Secret into the account to access resources.	For more information on configuring individual user accounts, see <i>Configure Your Access to S3</i> .

Default credential provider with EMR:

If you are using this method and integrating with an EMR cluster:

- Copying the custom credential JAR file must be added as a bootstrap action to the EMR cluster definition. See *Configure for EMR*.
- As an alternative to copying the JAR file, you can use the EMR EC2 instance-based roles to govern access. In this case, you must set the following parameter:

```
"aws.emr.forceInstanceRole": true,
```

For more information, see *Configure for EC2 Role-Based Authentication*.

Instance credential provider

When the platform is running on an EC2 instance, you can manage permissions through pre-defined IAM roles.

NOTE: AWS mode must be set to `system`.

NOTE: If the Designer Cloud powered by Trifacta platform is connected to an EMR cluster, you can force authentication to the EMR cluster to use the specified IAM instance role. See *Configure for EMR*.

For more information, see *Configure for EC2 Role-Based Authentication*.

Temporary credential provider

For even better security, you can enable use temporary credentials provided from your AWS resources based on an IAM role specified per user.

Tip: This method is recommended by AWS.

Set the following properties.

Property	Description
"aws.credentialProvider"	<ul style="list-style-type: none"> • If <code>aws.mode = system</code>, set this value to <code>temporary</code>. • If <code>aws.mode = user</code> and you are using per-user authentication, then this setting is ignored and should stay as <code>default</code>.

Per-user authentication

Individual users can be configured to provide temporary credentials for access to AWS resources, which is a more secure authentication solution.

- Optionally, you can leverage your existing S3 permission scheme by modifying the IAM role or roles used to access S3.
- For more information, see *Configure AWS Per-User Auth for Temporary Credentials*.

Configure authentication for EMR

For more information, see *Configure for EMR*.

AWS Authentication Topics

Configure AWS Per-User Auth for Temporary Credentials

Contents:

- *Before You Begin*
- *Enable*
- *Configure Per-User Authentication using IAM Role*
- *Enable Attribute-Based Access to S3*
 - *Prerequisites*
 - *Specify general Hadoop bundle JAR file*
 - *Modify IAM policy*
 - *Enable*
- *User Access*

For Designer Cloud Powered by Trifacta® Enterprise Edition, you can configure AWS authentication on a per-user basis, using temporary credentials for superior security.

Before You Begin

You must configure your AWS mode of access: `system` or `user`. For more information, see *Configure for AWS*.

Enable

To enable per-user authentication using temporary credentials, the following parameters must be set:

Property	Description
<pre>"aws.readFromConfigurationService": false,</pre>	Set this value to <code>false</code> for Designer Cloud Powered by Trifacta Enterprise Edition, which prevents the product from retrieving AWS-related configuration information from the incorrect source.
<pre>"aws.mode": "user",</pre>	Each user can specify credentials.

To authenticate to AWS services from the Designer Cloud powered by Trifacta platform using an IAM role:

Property	Description
<pre>"aws.ec2InstanceRoleForAssumeRole": true,</pre>	<ul style="list-style-type: none">• If <code>true</code>, then all users use the EC2 instance role for authenticating to the AWS STS service for their temporary credentials. <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"><p>NOTE: You must ensure that the role provides adequate access to STS. Details are below.</p></div> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"><p>Tip: This method is recommended.</p></div> <ul style="list-style-type: none">• If <code>false</code>, then a system-wide set of AWS key/secret credentials must be inserted into platform configuration in the Admin Settings page as the master set of credentials to access STS for temporary

credentials:

Properties to set:

```
"aws.s3.key"  
"aws.s3.secret"
```

NOTE: After specifying the above key/secret combination, you can skip to the User Access section below.

Configure Per-User Authentication using IAM Role

Please complete the following general steps.

Steps:

1. Instance role: Create an IAM role and link it to the EC2 instance where the Trifacta node is hosted.
 - a. Include the following IAM policy:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "VisualEditor0",  
      "Effect": "Allow",  
      "Action": "sts:AssumeRole",  
      "Resource": "arn:aws:iam::*:role/*"  
    }  
  ]  
}
```

- b. For more information, see <https://aws.amazon.com/premiumsupport/knowledge-center/assign-iam-role-ec2-instance/>.
2. User role: Create another IAM role and provides required access to the S3 buckets. Example:

```
{  
  "Version": "2012-10-17",  
  "Statement": [  
    {  
      "Sid": "MyBucketAndObjectPermissions",  
      "Action": [  
        "s3:GetBucketLocation",  
        "s3:ListBucket",  
        "s3:DeleteObject",  
        "s3:GetObject",  
        "s3:PutObject"  
      ],  
      "Effect": "Allow",  
      "Resource": [  
        "arn:aws:s3:::<my_s3_bucket>",  
        "arn:aws:s3:::<my_s3_bucket>/*"  
      ]  
    },  
    {  
      "Sid": "TrifactaPublicDatasets",  
      "Effect": "Allow",  
      "Action": [  

```

```

        "s3:GetObject",
        "s3:ListBucket"
    ],
    "Resource": [
        "arn:aws:s3:::trifacta-public-datasets/*",
        "arn:aws:s3:::trifacta-public-datasets"
    ]
}
]
}

```

where:

<my_s3_bucket> is the name of your bucket.

3. Under the user role definition, edit the Trust relationship. Add the instance role to Principal:

```

{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::<awsAccountId>:role/instanceRole"
        ]
      },
      "Action": "sts:AssumeRole"
    }
  ]
}

```

- a. For more information, see *Insert Trust Relationship in AWS IAM Role*.
 - b. For more granular control over the Trust relationship, see https://docs.aws.amazon.com/IAM/latest/UserGuide/reference_policies_elements_principal.html.
4. AWS Glue: If you are integrating with AWS Glue, additional permissions must be set. For more information, see *AWS Glue Access*.
 5. Log in the Designer Cloud powered by Trifacta platform as a Trifacta admin.
 6. Click the link to specify storage settings. Populate the values for:
 - a. IAM role
 - b. Role ARN
 - c. S3 Bucket Name
 - d. For more information, see *AWS Settings Page*.
 7. Save your changes.

Enable Attribute-Based Access to S3

When IAM roles are used for per-user authentication, Designer Cloud Powered by Trifacta Enterprise Edition can be configured to pass an additional attribute as part of any request for S3 resources through AWS Secure Token Service. This attribute, called a **session tag**, contains the Trifacta user identifier, which is the username part of the user's email address. This `userId` is used as the key within S3 to identify the permissions available to the user on S3. In this manner, you can leverage your existing enterprise S3 permissioning for more precise access, without having to replicate the permissioning in Designer Cloud Powered by Trifacta Enterprise Edition.

For more information on session tags, see

https://docs.aws.amazon.com/IAM/latest/UserGuide/tutorial_attribute-based-access-control.html.

Prerequisites

- S3 must be set as the base storage layer. For more information, see *Set Base Storage Layer*.
- Designer Cloud Powered by Trifacta Enterprise Edition must be configured to use IAM roles through the temporary credential provider mechanism for per-user authentication to AWS. See above.

- A `userId` must be matched to the identifier that is used within the enterprise infrastructure to define S3 access.
- If you are running jobs on EMR, EMR 5.29.0 and later is supported.

NOTE: After enabling the use of session tags, you must spin up a new EMR cluster, which forces EMR to use the newly deployed credential provider JAR file.

Specify general Hadoop bundle JAR file

This feature requires that you deploy the generic Hadoop bundle JAR file for use when running Spark jobs. Version-specific bundle JARs, which are used by default, do not have the latest AWS SDK binaries, which are required for this feature. There are no functional issues with using the generic bundle JAR, which includes these binaries.

Please complete the following steps.

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 the value listed below:

```
"hadoopBundleJar": "hadoop-deps/generic-hadoop/build/libs/generic-hadoop-bundle.jar"
```

3. Save your changes and restart the platform.

Modify IAM policy

The IAM policy used for S3 access must be modified to include the request permissions. When using session tags, any trust policies must have the `sts:TagSession` permission. Below, the previous policy has been modified to include the required elements:

NOTE: The `sts:TagSession` permission must be added to all IAM roles that are used to connect to S3 or S3-related resources.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": [
          "arn:aws:iam::<awsAccountId>:role/instanceRole"
        ]
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Enable

When the above change has been applied, you can enable the feature.

Steps:

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

```
Session Tags: Enable the use of session tags when assuming an IAM Role
```

3. In the following setting, specify the value that the Designer Cloud application should insert for the tag when requesting AWS resources:

```
Session Tags: The name of the session tag that holds the username as its value
```

4. A restart is not required.

NOTE: Users should log out and login again to experience the changes in permissions due to the session tags.

User Access

After per-user authentication has been enabled, each user must provide or be provided the credentials and S3 bucket to use.

Users can insert a default S3 bucket and credentials to use in their profiles. See *Configure Your Access to S3*.

Configure for AWS SAML Passthrough Authentication

Contents:

- *Prerequisites*
 - *Enable*
 - *Configure*
 - *List of Roles*
 - *Per-User Assignments*
 - *Assignment per API*
-

Optionally for single sign-on, the Designer Cloud powered by Trifacta® platform can leverage the AWS user/role mappings that are managed by a SAML authentication provider. In this authentication scenario:

- The Designer Cloud powered by Trifacta platform uses its native SAML support for SSO authentication.
- Access to AWS resources is governed by the set of permissions and IAM roles that are managed by your AWS admins.
- A third-party SAML Identity Provider provides IAM role ARNs for each authenticating user in a SAML assertion.
- Users of the Designer Cloud powered by Trifacta platform are mapped to one or more IAM roles. These IAM roles can be selected at the workspace (admin) or individual user level.
- The Designer Cloud powered by Trifacta platform does not allow admins or users to edit the list of IAM roles for use.

Usage:

When this feature is enabled, a user's available IAM roles are automatically synched via SAML. When a user signs in to the Designer Cloud application, the user can select a role to use.

Prerequisites

- Per-user authentication to AWS has been enabled.

NOTE: Please be sure to verify that you have deployed the required policies as part of any IAM roles in use.

For more information, see *Configure AWS Per-User Auth for Temporary Credentials*.

- This feature is supported only for the SAML authentication method of SSO authentication native to the Designer Cloud powered by Trifacta platform. It is not supported for any other SSO auth method. For more information, see *Configure SSO for SAML*.
- AWS permissions must be defined via IAM role and made available to an identity provider that adheres to SAML standards. The SAML identity provider must be configured with a list of SAML assertions containing the IAM roles that an external user may assume.

NOTE: When this feature is enabled and the platform is restarted, users of the Designer Cloud powered by Trifacta platform cannot authenticate to AWS resources until IAM roles have been assigned to their accounts. Where possible, you should enable this feature on an unused instance of the platform.

Enable

To enable, the following configuration change must be applied.

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.importAwsRoles.saml.enabled": true,
```

3. Save your changes and restart the platform.

Configure

After the feature has been enabled, the Designer Cloud powered by Trifacta platform assigns IAM roles to users automatically when they sign in.

List of Roles

The list of available roles is passed from the SAML identity provider to the Designer Cloud powered by Trifacta platform in a SAML assertion attribute. From this list of roles, each user can select the one to apply to the account.

Use the steps below to review and modify the SAML attribute that contains the list of role ARNs for users.

NOTE: Modify this value only if necessary.

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 the case-sensitive name of the SAML attribute:

```
"feature.importAwsRoles.saml.rolesAttribute": "https://aws.amazon.com/SAML/Attributes/Role",
```

Tip: For convenience, the default value for this SAML assertion attribute name matches the value used in AWS documentation:

https://docs.aws.amazon.com/IAM/latest/UserGuide/id_roles_providers_create_saml_assertions.html#saml_role-attribute

3. Save your changes and restart the platform.

Per-User Assignments

Individual users must select the IAM role ARN to assume from the list exposed by Trifacta administrator.

NOTE: Before a user is permitted to complete login to the application, the user must select a role from the provided list.

For more information, see *Configure Your Access to S3*.

Assignment per API

You can use the platform APIs to create platform AWS roles and assign them to users. For more information, see *API Task - Manage AWS Configurations*.

Configure for EC2 Role-Based Authentication

Contents:

- *IAM roles*
- *AWS System Mode*
- *Additional AWS Configuration*
- *Use of S3 Sources*

When you are running the Designer Cloud powered by Trifacta platform on an EC2 instance, you can leverage your enterprise IAM roles to manage permissions on the instance for the Designer Cloud powered by Trifacta platform. When this type of authentication is enabled, Trifacta administrators can apply a role to the EC2 instance where the platform is running. That role's permissions apply to all users of the platform.

IAM roles

Before you begin, your IAM roles should be defined and attached to the associated EC2 instance.

NOTE: The IAM instance role used for S3 access should have access to resources at the bucket level.

For more information, see

<http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html>.

AWS System Mode

To enable role-based instance authentication, the following parameter must be enabled.

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

Additional AWS Configuration

The following additional parameters must be specified:

Parameter	Description
<code>aws.credentialProvider</code>	Set this value to <code>instance</code> . IAM instance role is used for providing access.
<code>aws.hadoopFsUseSharedInstanceProvider</code>	Set this value to <code>true</code> for CDH. The class information is provided below.

Shared instance provider class information

Pre-Cloudera 6.0.0:

```
"org.apache.hadoop.fs.s3a.SharedInstanceProfileCredentialsProvider"
```

Cloudera 6.0.0 and later:

Set the above parameters as follows:

```
"aws.credentialProvider": "instance",  
"aws.hadoopFSUseSharedInstanceProvider": false,
```

Use of S3 Sources

To access S3 for storage, additional configuration for S3 may be required.

NOTE: Do not configure the properties that apply to `user` mode.

Output sizing recommendations:

- Single-file output: If you are generating a single file, you should try to keep its size under 1 GB.
- Multi-part output: For multiple-file outputs, each part file should be under 1 GB in size.
- For more information, see https://docs.aws.amazon.com/redshift/latest/dg/c_best-practices-use-multiple-files.html

For more information, see [S3 Access](#).

Configure for AWS Secrets Manager

Contents:

- *Limitations*
 - *Supported AWS resources*
 - *Configure*
 - *Configure region and metadata settings*
 - *Configure roles*
-

Secrets Manager is a secure AWS service that enables you to store and manage Personal Access Tokens, which are used by the Designer Cloud powered by Trifacta® platform to access AWS resources. When a user requests access to an applicable AWS resource, the Designer Cloud powered by Trifacta platform queries the Secrets Manager for the user's Personal Access Token, which is then used to access the resource. Individual users do not have access to the Secrets Manager.

For storing Personal Access Tokens, the Designer Cloud powered by Trifacta platform uses the following attributes to distinguish platform secrets from secrets stored by other applications or users in the customer's AWS account:

- Application name
- Environment

Permissions to use secrets can be controlled based on custom Tags or Paths.

For more information, see <https://docs.databricks.com/dev-tools/api/latest/authentication.html>.

Limitations

NOTE: You can have only one AWS account per Secrets Manager and per region. See below for configuring region-related settings.

NOTE: Customer-managed encryption keys are not supported for use in AWS Secrets Manager by the Designer Cloud powered by Trifacta platform .

Supported AWS resources

AWS Secrets Manager is supported for use in accessing the Personal Access Tokens for platform users for the following AWS resources:

- AWS Databricks

Configure

Please complete the following configuration steps to enable integration with AWS Secrets Manager.

Configure region and metadata settings

Configure the following region-related settings.

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

Property	Description
<code>aws.region</code>	This value should already be defined for your AWS integration.

Configure roles

Through the AWS console, you can define and manage the policies for the IAM roles including those associated with your EC2 instance.

Attach policy to EC2 instance role

When you are running the Designer Cloud powered by Trifacta platform on an EC2 instance, you can leverage your enterprise IAM roles to manage permissions on the EC2 instance. When this type of authentication is enabled, Designer Cloud powered by Trifacta platform administrators can apply a role to the EC2 instance where the platform is running. The instance profile that is attached to the EC2 instances must have the following secrets permission so that Designer Cloud powered by Trifacta platform can read and store secrets in Secrets Manager.

For more information on IAM roles for EC2, see <http://docs.aws.amazon.com/AWSEC2/latest/UserGuide/iam-roles-for-amazon-ec2.html>.

Steps:

The following permissions should be set in the EC2 instance:

1. Log in to the AWS console.
2. Add the following policy to the instance profile corresponding to the EC2 instance.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Sid": "VisualEditor0",
      "Effect": "Allow",
      "Action": [
        "secretsmanager:PutSecretValue",
        "secretsmanager:CreateSecret",
        "secretsmanager>DeleteSecret",
        "secretsmanager:GetSecretValue",
        "secretsmanager:TagResource"
      ],
      "Resource": "arn:aws:secretsmanager:<aws.region>:<your_aws_account_id>:secret:Trifacta/"
    }
  ]
}
```

Resource value	Description
<code><aws.region></code>	The region for which the access is provided. Verify that this value is set to <code>aws</code> .
<code><metadata.environment></code>	Differentiates the secrets between a test and production environment. You can set this value as per your requirements.

3. Save the IAM role definition.

NOTE : Designer Cloud powered by Trifacta platform recommends to avoid deleting secrets from the AWS console, as they cannot be restored for a period of 7 days. During this period, you cannot create secrets using the same name. As a result, existing jobs may fail without secrets available for a specific

user, and Designer Cloud powered by Trifacta platform cannot create a new secret with the same name. However, you can restore secrets through AWS CLI.

For more information on the AWS CLI for Secrets Manager, see <https://awscli.amazonaws.com/v2/documentation/api/latest/reference/secretsmanager/index.html>.

S3 Access

Contents:

- *Base Storage Layer*
- *Limitations*
- *Prerequisites*
 - *Required AWS Account Permissions*
- *Configuration*
 - *Define base storage layer*
 - *Enable read access to S3*
 - *Configure file storage protocols and locations*
 - *Java VFS service*
 - *S3 access modes*
 - *System mode - additional configuration*
- *S3 Configuration*
 - *Configuration reference*
 - *Enable use of server-side encryption*
 - *Configure S3 filewriter*
- *Create Redshift Connection*
- *Create Additional S3 Connections*
- *Additional Configuration for S3*
- *Testing*
- *Troubleshooting*
 - *Profiling consistently fails for S3 sources of data*
 - *Spark local directory has no space*

Below are instructions on how to configure Designer Cloud Powered by Trifacta® Enterprise Edition to point to S3.

Simple Storage Service (S3) is an online data storage service provided by Amazon, which provides low-latency access through web services. For more information, see <https://aws.amazon.com/s3/>.

Base Storage Layer

- **If base storage layer is S3:** you can enable read/write access to S3.
- **If base storage layer is not S3:** you can enable read-only access to S3.

Limitations

- The Designer Cloud powered by Trifacta platform only supports running S3-enabled instances over AWS.
- Access to AWS S3 Regional Endpoints through internet protocol is required. If the machine hosting the Designer Cloud powered by Trifacta platform is in a VPC with no internet access, a VPC endpoint enabled for S3 services is required. The Designer Cloud powered by Trifacta platform does not support access to S3 through a proxy server.
- Write access requires using S3 as the base storage layer. See *Set Base Storage Layer*.

NOTE: Spark 2.3.0 jobs may fail on S3-based datasets due to a known incompatibility. For details, see <https://github.com/apache/incubator-druid/issues/4456>.

If you encounter this issue, please set `spark.version` to `2.1.0` in platform configuration. For more information, see *Admin Settings Page*.

Prerequisites

- On the Trifacta node, you must install the Oracle Java Runtime Environment for Java 8. Other versions of the JRE are not supported. For more information on the JRE, see <http://www.oracle.com/technetwork/java/javase/downloads/index.html>.
- If IAM instance role is used for S3 access, it must have access to resources at the bucket level.

Required AWS Account Permissions

For more information, see *Required AWS Account Permissions*.

Configuration

Depending on your S3 environment, you can define:

- read access to S3
- access to additional S3 buckets
- S3 as base storage layer
- Write access to S3
 - S3 bucket that is the default write destination

Define base storage layer

The base storage layer is the default platform for storing results.

Required for:

- Write access to S3
- Connectivity to Redshift

The base storage layer for your Trifacta instance is defined during initial installation and cannot be changed afterward.

If S3 is the base storage layer, you must also define the default storage bucket to use during initial installation, which cannot be changed at a later time. See *Define default S3 write bucket* below.

For more information on the various options for storage, see *Storage Deployment Options*.

For more information on setting the base storage layer, see *Set Base Storage Layer*.

Enable read access to S3

When read access is enabled, Trifacta users can explore S3 buckets for creating datasets.

NOTE: When read access is enabled, Trifacta users have automatic access to all buckets to which the specified S3 user has access. You may want to create a specific user account for S3 access.

NOTE: Data that is mirrored from one S3 bucket to another might inherit the permissions from the bucket where it is owned.

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Set the following property to enabled:

```
Enable S3 Connectivity
```

3. Save your changes.
4. In the S3 configuration section, set `enabled=true`, which allows Trifacta users to browse S3 buckets through the Designer Cloud application .
5. Specify the AWS `key` and `secret` values for the user to access S3 storage.

Configure file storage protocols and locations

The Designer Cloud powered by Trifacta platform must be provided the list of protocols and locations for accessing S3.

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": ["s3"],
"fileStorage.defaultBaseUri": ["s3:///"],
```

Parameter	Description
filestorage.whitelist	<p>A comma-separated list of protocols that are permitted to access S3.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>NOTE: The protocol identifier "s3" must be included in this list.</p> </div>
filestorage.defaultBaseUri	<p>For each supported protocol, this parameter must contain a top-level path to the location where Designer Cloud powered by Trifacta platform files can be stored. These files include uploads, samples, and temporary storage used during job execution.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>NOTE: A separate base URI is required for each supported protocol. You may only have one base URI for each protocol.</p> </div> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>NOTE: For S3, 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 S3 connections created in Designer Cloud Powered by Trifacta Enterprise Edition.</p> </div> <p>Example:</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>s3:///</p> </div> <p>The above example is the most common example, as it is used as the base URI for all S3 connections that you create. Do not add a bucket value to the above URI.</p>

3. Save your changes and restart the platform.

Java VFS service

Use of SFTP connections requires the Java VFS service in the Designer Cloud powered by Trifacta platform .

NOTE: This service is enabled by default.

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

S3 access modes

The Designer Cloud powered by Trifacta platform supports the following modes for access S3. You must choose one access mode and then complete the related configuration steps.

NOTE: Avoid switching between user mode and system mode, which can disable user access to data. At install mode, you should choose your preferred mode.

System mode

(default) Access to S3 buckets is enabled and defined for all users of the platform. All users use the same AWS access key, secret, and default bucket.

System mode - read-only access

For read-only access, the key, secret, and default bucket must be specified in configuration.

NOTE: Please verify that the AWS account has all required permissions to access the S3 buckets in use. The account must have the ListAllMyBuckets ACL among its permissions.

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:

Parameters	Description
<code>aws.s3.key</code>	Set this value to the AWS key to use to access S3.
<code>aws.s3.secret</code>	Set this value to the secret corresponding to the AWS key provided.
<code>aws.s3.bucket.name</code>	Set this value to the name of the S3 bucket from which users may read data. NOTE: Bucket names are not validated. NOTE: Additional buckets may be specified. See below.

3. Save your changes.

User mode

Optionally, access to S3 can be defined on a per-user basis. This mode allows administrators to define access to specific buckets using various key/secret combinations as a means of controlling permissions.

NOTE: When this mode is enabled, individual users must have AWS configuration settings applied to their account, either by an administrator or by themselves. The global settings in this section do not apply in this mode.

To enable:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Verify that the following setting has been set to enabled:

```
Enable S3 Connectivity
```

3. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
4. Please verify that the setting below has been configured:

```
"aws.mode": "user",
```

5. Additional configuration is required for per-user authentication.
 - a. You can choose to enable session tags to leverage your existing S3 permission scheme.
 - b. For more information, see *Configure AWS Per-User Authentication*.

NOTE: If you have enabled user mode for S3 access, you must create and deploy an encryption key file. For more information, see *Create Encryption Key File*.

NOTE: If you have enabled `user` access mode, you can skip the following sections, which pertain to the `system` access mode, and jump to the *Enable Redshift Connection* section below.

System mode - additional configuration

The following sections apply only to `system` access mode.

Define default S3 write bucket

When S3 is defined as the base storage layer, write access to S3 is enabled. The Designer Cloud powered by Trifacta platform attempts to store outputs in the designated default S3 bucket.

NOTE: This bucket must be set during initial installation. Modifying it at a later time is not recommended and can result in inaccessible data in the platform.

NOTE: Bucket names cannot have underscores in them. See <http://docs.aws.amazon.com/AmazonS3/latest/dev/BucketRestrictions.html>.

Steps:

1. Define S3 to be the base storage layer. See *Set Base Storage Layer*.
2. Enable read access. See *Enable read access*.
3. Specify a value for `aws.s3.bucket.name` which defines the S3 bucket where data is written. Do not include a protocol identifier. For example, if your bucket address is `s3://MyOutputBucket`, the value to specify is the following:

```
MyOutputBucket
```

NOTE: Bucket names are not validated.

NOTE: Specify the top-level bucket name only. There should not be any backslashes in your entry.

NOTE: This bucket also appears as a read-access bucket if the specified S3 user has access.

Adding additional S3 buckets

When read access is enabled, all S3 buckets of which the specified user is the owner appear in the Designer Cloud application . You can also add additional S3 buckets from which to read.

NOTE: Additional buckets are accessible only if the specified S3 user has read privileges.

NOTE: Bucket names cannot have underscores in them.

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: `aws.s3.extraBuckets`:
 - a. In the Admin Settings page, specify the extra buckets as a comma-separated string of additional S3 buckets that are available for storage. Do not put any quotes around the string. Whitespace between string values is ignored.
 - b. In `trifacta-conf.json`, specify the `extraBuckets` array as a comma-separated list of buckets as in the following:

```
"extraBuckets": ["MyExtraBucket01", "MyExtraBucket02", "MyExtraBucket03"]
```

NOTE: Specify the top-level bucket name only. There should not be any backslashes in your entry.

NOTE: Bucket names are not validated.

3. Specify the `extraBuckets` array as a comma-separated list of buckets as in the following:

```
"extraBuckets": ["MyExtraBucket01", "MyExtraBucket02", "MyExtraBucket03"]
```

4. These values are mapped to the following bucket addresses:

```
s3://MyExtraBucket01
s3://MyExtraBucket02
s3://MyExtraBucket03
```

S3 Configuration

Configuration reference

You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.

Setting	Description
Enable S3 Connectivity	When set to enabled, the S3 file browser is displayed in the GUI for locating files.

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

```
"aws.s3.bucket.name": "<BUCKET_FOR_OUTPUT_IF_WRITING_TO_S3>"
"aws.s3.key": "<AWS_KEY>",
"aws.s3.secret": "<AWS_SECRET>",
"aws.s3.extraBuckets": [ "<ADDITIONAL_BUCKETS_TO_SHOW_IN_FILE_BROWSER>" ]
```

Setting	Description
bucket.name	Set this value to the name of the S3 bucket to which you are writing. <ul style="list-style-type: none">When <code>webapp.storageProtocol</code> is set to <code>s3</code>, the output is delivered to <code>aws.s3.bucket.name</code>.
key	Access Key ID for the AWS account to use. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">NOTE: This value cannot contain a slash (/).</div>
secret	Secret Access Key for the AWS account.
extraBuckets	Add references to any additional S3 buckets to this comma-separated array of values. The S3 user must have read access to these buckets.

Enable use of server-side encryption

You can configure the Designer Cloud powered by Trifacta platform to publish data on S3 when a server-side encryption policy is enabled. SSE-S3 and SSE-KMS methods are supported. For more information, see <http://docs.aws.amazon.com/AmazonS3/latest/dev/serv-side-encryption.html>.

Notes:

- When encryption is enabled, all buckets to which you are writing must share the same encryption policy. Read operations are unaffected.

To enable, please specify the following parameters.

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

Server-side encryption method

```
"aws.s3.serverSideEncryption": "none",
```

Set this value to the method of encryption used by the S3 server. Supported values:

NOTE: Lower case values are required.

- sse-s3
- sse-kms
- none

Server-side KMS key identifier

When KMS encryption is enabled, you must specify the AWS KMS key ID to use for the server-side encryption.

```
"aws.s3.serverSideKmsKeyId": "",
```

Notes:

- Access to the key:
 - Access must be provided to the authenticating user.
 - The AWS IAM role must be assigned to this key.
- Encrypt/Decrypt permissions for the specified KMS key ID:
 - Permissions must be assigned to the authenticating user.
 - The AWS IAM role must be given these permissions.
 - For more information, see <https://docs.aws.amazon.com/kms/latest/developerguide/key-policy-modifying.html> .

The format for referencing this key is the following:

```
"arn:aws:kms:<regionId>:<acctId>:key/<keyId>"
```

You can use an AWS alias in the following formats. The format of the AWS-managed alias is the following:

```
"alias/aws/s3"
```

The format for a custom alias is the following:

```
"alias/<FSR>"
```

where:

<FSR> is the name of the alias for the entire key.

Save your changes and restart the platform.

Configure S3 filewriter

The following configuration can be applied through the Hadoop `site-config.xml` file. If your installation does not have a copy of this file, you can insert the properties listed in the steps below into `trifacta-conf.json` to configure the behavior of the S3 filewriter.

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 `filewriter.hadoopConfig` block, where you can insert the following Hadoop configuration properties:

```
"filewriter": {  
  max: 16,  
  "hadoopConfig": {  
    "fs.s3a.buffer.dir": "/tmp",  
    "fs.s3a.fast.upload": "true"  
  },  
  ...  
}
```

Property	Description
<code>fs.s3a.buffer.dir</code>	Specifies the temporary directory on the Trifacta node to use for buffering when uploading to S3. If <code>fs.s3a.fast.upload</code> is set to <code>false</code> , this parameter is unused. NOTE: This directory must be accessible to the Batch Job Runner process during job execution.
<code>fs.s3a.fast.upload</code>	Set to <code>true</code> to enable buffering in blocks. When set to <code>false</code> , buffering in blocks is disabled. For a given file, the entire object is buffered to the disk of the Trifacta node. Depending on the size and volume of your datasets, the node can run out of disk space.

3. Save your changes and restart the platform.

Create Redshift Connection

For more information, see *Amazon Redshift Connections*.

Create Additional S3 Connections

Creating additional S3 connections is not required. After you define S3 as your base storage layer, you can create user-specific access to S3 buckets through *Designer Cloud application*. For more information, see *External S3 Connections*.

Additional Configuration for S3

The following parameters can be configured through the *Designer Cloud powered by Trifacta platform* to affect the integration with S3. You may or may not need to modify these values for your deployment.

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

Parameter	Description
<code>aws.s3.endpoint</code>	This value should be the S3 endpoint DNS name value.

NOTE: Do not include the protocol identifier.

Example value:

```
s3.us-east-1.amazonaws.com
```

If your S3 deployment is either of the following:

- located in a region that does not support the default endpoint, or
- v4-only signature is enabled in the region

Then, you can specify this setting to point to the S3 endpoint for Java/Spark services.

For more information on this location, see https://docs.aws.amazon.com/general/latest/gr/rande.html#s3_region.

Testing

Restart services. See *Start and Stop the Platform*.

Try running a simple job from the Designer Cloud application . For more information, see *Verify Operations*.

Troubleshooting

Profiling consistently fails for S3 sources of data

If you are executing visual profiles of datasets sourced from S3, you may see an error similar to the following in the `batch-job-runner.log` file:

```
01:19:52.297 [Job 3] ERROR com.trifacta.hadoopdata.joblaunch.server.BatchFileWriterWorker -  
BatchFileWriterException: Batch File Writer unknown error:  
{jobId=3, why=bound must be positive}  
01:19:52.298 [Job 3] INFO com.trifacta.hadoopdata.joblaunch.server.BatchFileWriterWorker - Notifying monitor  
for job 3 with status code FAILURE
```

This issue is caused by improperly configuring buffering when writing to S3 jobs. The specified local buffer cannot be accessed as part of the batch job running process, and the job fails to write results to S3.

Solution:

You may do one of the following:

- Use a valid temp directory when buffering to S3.
- Disable buffering to directory completely.

Steps:

- 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, where you can insert either of the following Hadoop configuration properties:

```
"filewriter": {  
  max: 16,  
  "hadoopConfig": {  
    "fs.s3a.buffer.dir": "/tmp",  
    "fs.s3a.fast.upload": false  
  },  
  ...  
}
```

Property	Description
<code>fs.s3a.buffer.dir</code>	Specifies the temporary directory on the Trifacta node to use for buffering when uploading to S3. If <code>fs.s3a.fast.upload</code> is set to <code>false</code> , this parameter is unused.
<code>fs.s3a.fast.upload</code>	When set to <code>false</code> , buffering is disabled.

- Save your changes and restart the platform.

Spark local directory has no space

During execution of a Spark job, you may encounter the following error:

```
org.apache.hadoop.util.DiskChecker$DiskErrorException: No space available in any of the local directories.
```

Solution:

- Restart Trifacta services, which may free up some temporary space.
- Use the steps in the preceding solution to reassign a temporary directory for Spark to use (`fs.s3a.buffer.dir`).

AWS Glue Access

Contents:

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

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

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

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

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

Supported Deployment

Amazon Glue tables can be read under the following conditions:

- The Designer Cloud powered by Trifacta platform uses S3 as the base storage layer.
- The Designer Cloud powered by 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 Amazon 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 Amazon 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 Amazon Glue tables.

For additional limitations on access Hive tables through Amazon 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 Designer Cloud powered by Trifacta platform and Amazon 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 following custom bootstrap action, the cluster does one of the following:

- Interacts with Amazon 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 Amazon Glue.

Steps:

1. From the installation of the Designer Cloud powered by 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 and 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 Amazon Glue .	See <i>Configure for AWS</i> .
user	The user role must provide access to Amazon Glue .	See below for an example IAM role access control. See <i>Configure AWS Per-User Auth for Temporary Credentials</i>

Example fine-grain access control for IAM policy:

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

NOTE: Please verify that access is granted in the IAM policy to the default database for Amazon 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

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

- Any samples of your data that are generated by the Designer Cloud powered by Trifacta platform are stored in S3. Sample data is read by the platform directly from S3.
- Source data is read through Amazon 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 Amazon 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 Amazon 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** assigned to the IAM role.
 - In the Amazon 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 Amazon 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

See *AWS Glue Connections*.

Snowflake Access

Contents:

- *Limitations*
 - *Prerequisites*
 - *General*
 - *AWS permissions*
 - *OAuth2 requirements*
 - *Enable*
 - *Create Stage*
 - *Create Snowflake Connection*
 - *Testing*
-

This section describes how to integrate the Designer Cloud powered by Trifacta® platform with Snowflake data bases.

- Snowflake provides a cloud-database data warehouse 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 Designer Cloud Powered by Trifacta Enterprise Edition 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.

Prerequisites

General

- If you do not provide a stage database, then the Designer Cloud powered by Trifacta platform must create one for you under the `PUBLIC` schema in the default database.
 - In this default database, you must include a schema named `PUBLIC`.
 - For more information, please see the Snowflake documentation.
- The user-created stage must point to the same S3 bucket that is the default S3 bucket in use by Designer Cloud Powered by Trifacta Enterprise Edition.

AWS permissions

Your Snowflake stage and other databases must be permitted to use S3 resources for your users.

NOTE: If users in your deployment are using IAM roles in user mode for AWS access, then the Snowflake stage must have permissions to write to the user's S3 bucket.

For more information, see *Required AWS Account Permissions*.

OAuth2 requirements

If you are integrating with Snowflake using OAuth2, additional configuration is required:

- OAuth2 must be enabled in the product. For more information, see *Enable OAuth 2.0 Authentication*.
- You must create a client app and client to manage authentication between the Designer Cloud application and your Snowflake deployment. For more information, see *OAuth 2.0 for Snowflake*.

Enable

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

Create Stage

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

- If a stage is used, it should be in 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. Please verify the Prerequisites again.

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 Designer Cloud Powered by Trifacta Enterprise Edition, the stage location is specified as part of creating the Snowflake connection.

Create Snowflake Connection

For more information, see *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*.

Configure for EMR

Contents:

- *Supported Versions*
 - *EMR 6*
 - *EMR 5*
 - *Supported Spark Versions*
- *Limitations*
 - *Limitations for Kerberos*
- *Create EMR Cluster*
 - *Cluster options*
 - *Specify cluster roles*
 - *Authentication*
- *Set up S3 Buckets*
 - *Bucket setup*
 - *Set up EMR resources buckets*
- *Access Policies*
 - *EC2 instance profile*
 - *EMR roles*
 - *General configuration for Designer Cloud powered by Trifacta platform*
 - *Change admin password*
 - *Verify S3 as base storage layer*
 - *Set up S3 integration*
 - *Configure EMR authentication mode*
- *Configure Designer Cloud powered by Trifacta platform for EMR*
 - *Enable EMR integration*
 - *Apply EMR cluster ID*
 - *Extract IP address of master node in private sub-net*
 - *Configure Spark for EMR*
- *Configure Designer Cloud powered by Trifacta platform for EMR with Kerberos*
 - *Disable standard EMR integration*
 - *Enable YARN*
 - *Acquire site config files*
 - *Unused properties for EMR with Kerberos*
- *Additional Configuration for EMR*
 - *Enable EMR job cancellation*
 - *Default Hadoop job results format*
 - *Configure Snappy publication*
 - *Additional parameters*
- *Optional Configuration*
 - *Configure for Redshift*
 - *Configure for EMR high availability*
 - *Switch EMR Cluster*
 - *Configure Batch Job Runner*
 - *Modify Job Tag Prefix*
- *Testing*

You can configure your instance of the Designer Cloud powered by Trifacta platform to integrate with Amazon Elastic MapReduce (EMR), a highly scalable Hadoop-based execution environment.

NOTE: This section applies only to installations of Designer Cloud Powered by Trifacta Enterprise Edition where a valid license key file has been applied to the platform.

- **Amazon EMR** (Elastic MapReduce) provides a managed Hadoop framework that makes it easy, fast, and cost-effective to process vast amounts of data across dynamically scalable Amazon EC2 instances. For more information on EMR, see <http://docs.aws.amazon.com/cli/latest/reference/emr/>.

This section can be used to integrate with the following cluster deployments:

- **EMR:** default
- **EMR with Kerberos:** a separate configuration path is documented in this section

This section outlines how to create a new EMR cluster and integrate the Designer Cloud powered by Trifacta platform with it. The platform can be integrated with existing EMR clusters.

Supported Versions

EMR 6

Supported Versions: EMR 6.2.1, 6.3

NOTE: EMR 6.2.1 and EMR 6.3 require Spark 3.0.1. For more information, see *Configure for Spark*.

NOTE: Do not use EMR 6.2.0. Use EMR 6.2.1.

NOTE: EMR 6.0 and EMR 6.1 are not supported.

EMR 5

Supported Versions: EMR 5.13 - EMR 5.30.2

NOTE: EMR 5.28.0 is not supported, due to *Spark compatibility issues*. Please use 5.28.1 or later.

NOTE: Do not use EMR 5.30.0 or EMR 5.30.1. Use EMR 5.30.2.

NOTE: EMR 5.20 - EMR 5.30 requires Spark 2.4. For more information, see *Configure for Spark*.

Supported Spark Versions

Depending on the version of EMR in use, you must configure the Designer Cloud powered by Trifacta platform to use the appropriate version of Spark. Please note the appropriate configuration settings below for later use.

NOTE: The version of Spark to use for the platform is defined in the `spark.version` property. This configuration step is covered later.

EMR versions	Spark version	Additional configuration and notes
EMR 6.2.1, EMR 6.3	<code>"spark.version": "3.0.1",</code>	Please also set the following:

		"spark.scalaVersion": "2.12",
EMR 5.20 - EMR 5.30.2	"spark.version": "2.4.6",	
EMR 5.13 - EMR 5.19	"spark.version": "2.3.0",	

Limitations

- The Designer Cloud powered by Trifacta platform must be installed on AWS.

Limitations for Kerberos

If you are integrating with a kerberized EMR cluster, the following additional limitations apply:

- This feature is supported for Designer Cloud Powered by Trifacta Enterprise Edition only.
- The Designer Cloud powered by Trifacta platform must be deployed on an AWS EC2 Instance that is joined to the same domain as the EMR cluster.
- The EMR cluster must be kerberized using the Cross-Realm Trust method. Additional information is below.

Create EMR Cluster

Use the following section to set up your EMR cluster for use with the Designer Cloud powered by Trifacta platform .

- **Via AWS EMR UI:** This method is assumed in this documentation.
- **Via AWS command line interface:** For this method, it is assumed that you know the required steps to perform the basic configuration. For custom configuration steps, additional documentation is provided below.

NOTE: If you are integrating with a kerberized EMR cluster, the cluster must be kerberized using the Cross-Realm Trust method. The KDC on the EMR cluster must establish a cross-realm trust with the external KDC. No other Kerberos method is supported.

For more information, see

https://docs.amazonaws.cn/en_us/emr/latest/ManagementGuide/emr-kerberos-options.html.

NOTE: It is recommended that you set up your cluster for exclusive use by the Designer Cloud powered by Trifacta platform .

Cluster options

NOTE: If you are deploying EMR in a highly available environment, you must create each EMR cluster from the command line. For more information, see "Configure for EMR High Availability" below.

In the Amazon EMR console, click **Create Cluster**. Click **Go to advanced options**. Complete the sections listed below.

NOTE: Please be sure to read all of the cluster options before setting up your EMR cluster.

NOTE: Please perform your configuration through the Advanced Options task.

For more information on setting up your EMR cluster, see <http://docs.aws.amazon.com/cli/latest/reference/emr/create-cluster.html>.

Advanced Options

In the Advanced Options screen, please select the following:

- Software Configuration:
 - Release: EMR version to select.
 - Select:
 - Hadoop 2.8.3
 - Hue 3.12.0
 - Ganglia 3.7.2

Tip: Although it is optional, Ganglia is recommended for monitoring cluster performance.

- Spark version should be set accordingly. See "Supported Spark Versions" above.
- Deselect everything else.
- Multiple master nodes (optional):
 - To deploy your EMR cluster in a highly available configuration, select the "Use multiple master nodes" checkbox.
 - When selected, this option creates a total of three master nodes. Two of the nodes are configured as failover options if the first one fails.

Deploying EMR with multiple master nodes may incur additional costs.

Additional configuration is required. For more information, see "Configure for EMR High Availability" below.

- Edit the software settings:
 - Copy and paste the following into **Enter Configuration**:

```
[
  {
    "Classification": "capacity-scheduler",
    "Properties": {
      "yarn.scheduler.capacity.resource-calculator": "org.apache.hadoop.yarn.util.resource.
      DominantResourceCalculator"
    }
  }
]
```

- **EMR job cancellation:** The following configuration entry is required if you wish to enable EMR job cancellation:

```
{
  "Classification": "core-site",
```

```
"Properties": {
  "hadoop.http.filter.initializers": "org.apache.hadoop.security.
HttpCrossOriginFilterInitializer,org.apache.hadoop.http.lib.StaticUserWebFilter"
}
```

Additional configuration is required later to enable job cancellation. For more information, see "Enable EMR job cancellation" below.

- Auto-terminate cluster after the last step is completed: **Leave this option disabled.**

Hardware configuration

NOTE: Please apply the sizing information for your EMR cluster that was recommended for you. If you have not done so, please contact your Trifacta representative.

General Options

- Cluster name: Provide a descriptive name.
- Logging: Enable logging on the cluster.
 - S3 folder: Please specify the S3 bucket and path to the logging folder.

NOTE: Please verify that this location is read accessible to all users of the platform. See below for details.

- Debugging: Enable.
- Termination protection: Enable.
- Tags:
 - No options required.
- Additional Options:
 - EMRFS consistent view: Do not enable.
 - Custom AMI ID: None.
 - Bootstrap Actions:
 - If you are using a custom credential provider JAR, you must create a bootstrap action.

NOTE: This configuration must be completed before you create the EMR cluster. For more information, see *Authentication* below.

Security Options

- EC2 key pair: Please select a key/pair to use if you wish to access EMR nodes via SSH.
- Permissions: Set to Custom to reduce the scope of permissions. For more information, see "Access Policies" below.

NOTE: Default permissions give access to everything in the cluster.

- Encryption Options
 - No requirements.
- EC2 Security Groups:
 - The selected security group for the master node on the cluster must allow TCP traffic from the Trifacta instance on port 8088. For more information, see *System Ports*.

Create cluster and acquire cluster ID

If you performed all of the configuration, including the sections below, you can create the cluster.

NOTE: You must acquire your EMR cluster ID for use in configuration of the Designer Cloud powered by Trifacta platform .

Specify cluster roles

The following cluster roles and their permissions are required. For more information on the specifics of these policies, see "Access Policies" below.

- **EMR Role:**
 - Read/write access to log bucket
 - Read access to resource bucket
- **EC2 instance profile:**
 - If using instance mode:
 - EC2 profile should have read/write access for all users.
 - EC2 profile should have same permissions as EC2 Edge node role.
 - Read/write access to log bucket
 - Read access to resource bucket
- **Auto-scaling role:**
 - Read/write access to log bucket
 - Read access to resource bucket
 - Standard auto-scaling permissions

Authentication

You can use one of two methods for authenticating the EMR cluster's access to S3:

- **Role-based IAM authentication (recommended):** This method leverages your IAM roles on the EC2 instance.
- **Custom credential provider:** This method utilizes a custom credential provider JAR file provided with the platform. This custom credential provider is automatically deployed by the Designer Cloud powered by Trifacta platform during job submission.

Role-based IAM authentication

You can leverage your IAM roles to provide role-based authentication to the S3 buckets.

NOTE: The IAM role that is assigned to the EMR cluster and to the EC2 instances on the cluster must have access to the data of all users on S3.

For more information, see *Configure for EC2 Role-Based Authentication*.

Custom credential provider

If you are not using IAM roles for access, you can manage access using either of the following:

- AWS key and secret values specified in `trifacta-conf.json`
- AWS user mode

In either scenario, the Designer Cloud powered by Trifacta platform deploys a custom credential provider JAR file to the EMR before the job is executed.

NOTE: If you are also integrating with AWS Glue, you must provide a separate custom credential JAR file as part of that integration. For more information, see *AWS Glue Access*.

Set up S3 Buckets

Bucket setup

You must set up S3 buckets for read and write access.

NOTE: Within the Designer Cloud powered by Trifacta platform , you must enable use of S3 as the default storage layer. This configuration is described later.

For more information, see [S3 Access](#).

Set up EMR resources buckets

NOTE: If you are connecting to a kerberized EMR cluster, please skip to the next section. This section is not required.

On the EMR cluster, all users of the platform must have access to the following locations:

Location	Description	Required Access
EMR Resources bucket and path	The S3 bucket and path where resources can be stored by the Designer Cloud powered by Trifacta platform for execution of Spark jobs on the cluster. The locations are configured separately in the Designer Cloud powered by Trifacta platform .	Read/Write
EMR Logs bucket and path	The S3 bucket and path where logs are written for cluster job execution.	Read

These locations are configured on the Designer Cloud powered by Trifacta platform later.

Access Policies

EC2 instance profile

Trifacta users require the following policies to run jobs on the EMR cluster:

```
{
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "elasticmapreduce:AddJobFlowSteps",
        "elasticmapreduce:DescribeStep",
        "elasticmapreduce:DescribeCluster",
        "elasticmapreduce:ListInstanceGroups",
        "elasticmapreduce:CancelSteps"
      ],
      "Resource": [
        "*"
      ]
    },
    {
      "Effect": "Allow",
      "Action": [
        "s3:*"
      ],
      "Resource": [
        "arn:aws:s3:::__EMR_LOG_BUCKET__",
        "arn:aws:s3:::__EMR_LOG_BUCKET__/*"
      ]
    }
  ]
}
```

```

        "arn:aws:s3:::__EMR_RESOURCE_BUCKET__",
        "arn:aws:s3:::__EMR_RESOURCE_BUCKET__/*"
    ]
}
]
}

```

Permissions for EMR high availability

If the Designer Cloud powered by Trifacta platform is integrating with a highly available EMR cluster through multiple master nodes, the following permission must be included in the ARN used for accessing EMR:

```
"elasticmapreduce:listInstances",
```

Additional configuration is required. See "Configure for EMR High Availability" below.

EMR roles

The following policies should be assigned to the EMR roles listed below for read/write access:

```

{
    "Effect": "Allow",
    "Action": [
        "s3:*"
    ],
    "Resource": [
        "arn:aws:s3:::__EMR_LOG_BUCKET__",
        "arn:aws:s3:::__EMR_LOG_BUCKET__/*",
        "arn:aws:s3:::__EMR_RESOURCE_BUCKET__",
        "arn:aws:s3:::__EMR_RESOURCE_BUCKET__/*"
    ]
}

```

General configuration for Designer Cloud powered by Trifacta platform

Please complete the following sections to configure the Designer Cloud powered by Trifacta platform to communicate with the EMR cluster.

Change admin password

As soon as you have installed the software, you should login to the application and change the admin password. The initial admin password is the instanceId for the EC2 instance. For more information, see *Change Password*.

Verify S3 as base storage layer

EMR integrations requires use of S3 as the base storage layer.

NOTE: The base storage layer must be set during initial installation and set up of the Trifacta node.

See *Set Base Storage Layer*.

Set up S3 integration

To integrate with S3, additional configuration is required. See *S3 Access*.

Configure EMR authentication mode

Authentication to AWS and to EMR supports the following basic modes:

- **System:** A single set of credentials is used to connect to resources.
- **User:** Each user has a separate set of credentials. The user can choose to submit key-secret combinations or role-based authentication.

NOTE: Your method of authentication to AWS should already be configured. For more information, see *Configure for AWS*.

The authentication mode for your access to EMR can be configured independently from the base authentication mode for AWS, with the following exception:

NOTE: If `aws.emr.authMode` is set to `user`, then `aws.mode` must also be set to `user`.

Authentication mode configuration matrix:

AWS mode (aws.mode)	system	user
EMR mode (aws.emr.authMode)		
system	<p>AWS and EMR use a single key-secret combination. Parameters to set:</p> <pre>"aws.s3.key" "aws.s3.secret"</pre> <p>See <i>Configure for AWS</i>.</p>	<p>AWS access uses a single key-secret combination.</p> <p>EMR access is governed by per-user credentials. Per-user credentials can be provided from one of several different providers.</p> <p>NOTE: Per-user access requires additional configuration for EMR. See the following section.</p> <p>For more information on configuring per-user access, see <i>Configure for AWS</i>.</p>
user	Not supported	<p>AWS and EMR use the same per-user credentials for access. Per-user credentials can be provided from one of several different providers.</p> <p>NOTE: Per-user access requires additional configuration for EMR. See the following section.</p> <p>For more information on configuring per-user access, see <i>Configure AWS Per-User Auth for Temporary Credentials</i>.</p>

Please apply the following configuration to set the EMR authentication mode:

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 settings and apply the appropriate values. See the table below:

```
"aws.emr.authMode": "user",
```

Setting	Description
aws.emr.authMode	<p>Configure the mode to use to authenticate to the EMR cluster:</p> <p><code>system</code> - In system mode, the specified AWS key and secret combination are used to authenticate to the EMR cluster. These credentials are used for all users.</p> <p><code>user</code> - In user mode, user configuration is retrieved from the database.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>NOTE: User mode for EMR authentication requires that <code>aws.mode</code> be set to <code>user</code>. Additional configuration for EMR is below.</p> </div>

3. Save your changes.

EMR per-user authentication for the Designer Cloud powered by Trifacta platform

If you have enabled per-user authentication for EMR (`aws.emr.authMode=user`), you must set the following properties based on the credential provider for your AWS per-user credentials.

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

Authentication method	Properties and values
<p>Use default credential provider for all Trifacta access including EMR.</p> <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>NOTE: This method requires the deployment of a custom credential provider JAR.</p> </div>	<pre>"aws.credentialProvider": " default", "aws.emr.forceInstanceRole": false,</pre>
<p>Use default credential provider for all Trifacta access. However, EC2 role-based IAM authentication is used for EMR.</p>	<pre>"aws.credentialProvider": " default", "aws.emr.forceInstanceRole": true,</pre>
<p>EC2 role-based IAM authentication for all Trifacta access</p>	<pre>"aws.credentialProvider": " instance",</pre>

Configure Designer Cloud powered by Trifacta platform for EMR

NOTE: This section assumes that you are integrating with an EMR cluster that has not been kerberized. If you are integrating with a Kerberized cluster, please skip to "Configure for EMR with Kerberos".

Enable EMR integration

After you have configured S3 to be the base storage layer, you must enable EMR integration.

Steps:

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

1. Set the following value:

```
"webapp.runInEMR": true,
```

2. Set the following values:

```
"webapp.runWithSparkSubmit": false,
```

3. Verify the following property values:

```
"webapp.runWithSparkSubmit": false,  
"webapp.runInDataflow": false,
```

4. Save your changes and restart the platform.

5. To enable Trifacta Photon, an in-memory running environment for small- to medium-sized jobs, please do the following:

- In the Designer Cloud application, navigate to **User menu > Admin console > Workspace settings**.
- In the Workspace Settings page, set `Photon execution` to `Enabled`.
- Trifacta Photon is available for job execution when you next log in to the Designer Cloud application.

Apply EMR cluster ID

The Designer Cloud powered by Trifacta platform must be aware of the EMR cluster to which to connection.

Steps:

- Administrators can apply this configuration change through the *Admin Settings Page* in the application. If the application is not available, the settings are available in `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
- Under External Service Settings, enter your AWS EMR Cluster ID. Click the Save button below the textbox.

For more information, see *Admin Settings Page*.

Extract IP address of master node in private sub-net

If you have deployed your EMR cluster on a private sub-net that is accessible outside of AWS, you must enable this property, which permits the extraction of the IP address of the master cluster node through DNS.

NOTE: This feature must be enabled if your EMR is accessible outside of AWS on a private network.

Steps:

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

```
"emr.extractIPFromDNS": false,
```

3. Save your changes and restart the platform.

Configure Spark for EMR

For EMR, you can configure a set of Spark-related properties to manage the integration and its performance.

Configure Spark version

Depending on the version of EMR with which you are integrating, the Designer Cloud powered by Trifacta platform must be modified to use the appropriate version of Spark to connect to EMR.

NOTE: You should have already acquired the value to apply. See "Supported Spark Versions" above.

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:

```
"spark.version": "<SparkVersionForMyEMRVersion>",
```

3. This setting is ignored for EMR: `spark.useVendorSparkLibraries`
4. Save your changes.

Disable Spark job service

The Spark job service is not used for EMR job execution. Please complete the following to disable it:

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 and set it to `false`:

```
"spark-job-service.enabled": false,
```

3. Locate the following and set it to `false`:

```
"spark-job-service.enableHiveSupport": false,
```

4. Save your changes.

Specify YARN queue for Spark jobs

Through the Admin Settings page, you can specify the YARN queue to which to submit your Spark jobs. All Spark jobs from the Designer Cloud powered by Trifacta platform are submitted to this queue.

Steps:

1. In platform configuration, locate the following:

```
"spark.props.spark.yarn.queue"
```

2. Specify the name of the queue.
3. Save your changes.

Allocation properties

The following properties must be passed from the Designer Cloud powered by Trifacta platform to Spark for proper execution on the EMR cluster.

To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

NOTE: Do not modify these properties through the Admin Settings page. These properties must be added as extra properties through the Spark configuration block. Ignore any references in `trifacta-conf.json` to these properties and their settings.

```
"spark": {
  ...
  "props": {
    "spark.dynamicAllocation.enabled": "true",
    "spark.shuffle.service.enabled": "true",
    "spark.executor.instances": "0",
    "spark.executor.memory": "2048M",
    "spark.executor.cores": "2",
    "spark.driver.maxResultSize": "0"
  }
  ...
}
```

Property	Description	Value
<code>spark.dynamicAllocation.enabled</code>	Enable dynamic allocation on the Spark cluster, which allows Spark to dynamically adjust the number of executors.	true
<code>spark.shuffle.service.enabled</code>	Enable Spark shuffle service, which manages the shuffle data for jobs, instead of the executors.	true
<code>spark.executor.instances</code>	Default count of executor instances.	See Sizing Guide.
<code>spark.executor.memory</code>	Default memory allocation of executor instances.	See Sizing Guide.
<code>spark.executor.cores</code>	Default count of executor cores.	See Sizing Guide.
<code>spark.driver.maxResultSize</code>	Enable serialized results of unlimited size by setting this parameter to zero (0).	0

Combine transform and profiling for Spark jobs

When profiling is enabled for a Spark job, the transform and profiling tasks are combined by default. As needed, you can separate these two tasks. Publishing behaviors vary depending on the approach. For more information, see *Configure for Spark*.

Configure Designer Cloud powered by Trifacta platform for EMR with Kerberos

NOTE: This section applies only if you are integrating with a kerberized EMR cluster. If you are not, please skip to "Additional Configuration for EMR".

Disable standard EMR integration

When running jobs against a kerberized EMR cluster, you utilize the Spark-submit method of job submission. You must disable the standard EMR integration.

Steps:

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

1. Search for the following setting and set it `false`:

```
"webapp.runInEMR": false,
```

2. Set the following value:

```
"webapp.runWithSparkSubmit": true,
```

3. Disable use of Hive, which is not supported with EMR:

```
"spark-job-service.enableHiveSupport": false,
```

4. Verify the following property value:

```
"webapp.runInDataflow": false,
```

5. Save your changes.

Enable YARN

To use Spark-submit, the Spark master must be set to use YARN.

Steps:

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

1. Search for the following setting and set it `yarn`:

```
"spark.master": "yarn",
```

2. Save your changes.

Acquire site config files

For integrating with an EMR cluster with Kerberos, the EMR cluster site XML configuration files must be downloaded from the EMR master node to the Trifacta node.

NOTE: This step is not required for non-Kerberized EMR clusters.

NOTE: When these files change, you must update the local copies.

1. Download the Hadoop Client Configuration files from the EMR master node. The required files are the following:
 - a. `core-site.xml`
 - b. `hdfs-site.xml`
 - c. `mapred-site.xml`
 - d. `yarn-site.xml`

2. These configuration files must be moved to the Trifacta deployment. By default, these files are in `/etc/hadoop/conf`:

```
sudo cp <location>/*.xml /opt/trifacta/conf/hadoop-site/  
sudo chown trifacta:trifacta /opt/trifacta/conf/hadoop-site/*.xml
```

3. (Option) If we want to support impersonate, we also need copy `*.keytab` from the EMR master node under `/etc` folder to EC2 instance under same folder.

Unused properties for EMR with Kerberos

When integrating with a kerberized EMR cluster, the following Trifacta settings are unused:

1. **External Service Settings:** In the Admin Settings page, this section of configuration does not apply to EMR with Kerberos.
2. **Unused EMR settings:** In the Admin Settings page, the following EMR settings do not apply to EMR with Kerberos:

```
aws.emr.tempfilesCleanupAge  
aws.emr.proxyUser  
aws.emr.maxLogPollingRetries  
aws.emr.jobTagPrefix  
aws.emr.getLogOnFailure  
aws.emr.getLogForAllJobs  
aws.emr.extractIPFromDNS  
aws.emr.connectToRmEnabled
```

Additional Configuration for EMR

Enable EMR job cancellation

By default, a job that is started on EMR cannot be canceled through the application. Optionally, you can enable users to cancel their EMR jobs in progress.

Prerequisites:

1. The following permission must be added to the IAM role that is used to interact with EMR:

```
elasticmapreduce:CancelSteps
```

For more information, See "EC2 instance profile" above.

2. You should add an additional software setting to the cluster definition through the EMR console. For more information, see "Advanced Options" above.

Steps:

Please complete the following configuration changes to enable job cancellation on EMR.

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

NOTE: Enabled by default, this parameter is optional for basic EMR connectivity. This parameter must be enabled for EMR job cancellation.

```
"aws.emr.connectToRmEnabled": true,
```

3. Please verify that the Trifacta node can connect to the EMR master node on port 8088. For more information, see "Create EMR Cluster" above.
4. Please locate the following parameter and verify that it is set to `true`:

```
"aws.emr.cancelEnabled": true,
```

5. Save your changes and restart the platform.
6. To verify, launch an EMR job. In the Flow View context menu or in the Job History page, you should see a **Cancel job** option.

Default Hadoop job results format

For smaller datasets, the platform recommends using the Trifacta Photon running environment.

For larger datasets, if the size information is unavailable, the platform recommends by default that you run the job on the Hadoop cluster. For these jobs, the default publishing action for the job is specified to run on the Hadoop cluster, generating the output format defined by this parameter. Publishing actions, including output format, can always be changed as part of the job specification.

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

```
"webapp.defaultHadoopFileFormat": "csv",
```

Accepted values: `csv`, `json`, `avro`, `pqt`

For more information, see *Run Job Page*.

Configure Snappy publication

If you are publishing using Snappy compression for jobs run on an EMR cluster, you may need to perform the following additional configuration.

Steps:

1. SSH into EMR cluster (master) node:

```
ssh <EMR master node>
```

2. Create tarball of native Hadoop libraries:

```
tar -C /usr/lib/hadoop/lib -czvf emr-hadoop-native.tar.gz native
```

3. Copy the tarball to the Trifacta EC2 instance used by the into the `/tmp` directory:

```
scp -p emr-hadoop-native.tar.gz <EC2 instance>:/tmp
```

4. SSH to Trifacta EC2 instance:

```
ssh <EC2 instance>
```

5. Create path values for libraries:

```
sudo -u trifacta mkdir -p /opt/trifacta/services/batch-job-runner/build/libs
```

6. Untar the tarball to the Trifacta installation path:

```
sudo -u trifacta tar -C /opt/trifacta/services/batch-job-runner/build/libs -xzf /tmp/emr-hadoop-native.tar.gz
```

7. Verify `libhadoop.so*` and `libsnappy.so*` libraries exist and are owned by the Trifacta user:

```
ls -l /opt/trifacta/services/batch-job-runner/build/libs/native/
```

8. Verify that the `/tmp` directory has the proper permissions for publication. For more information, see *Supported File Formats*.

9. A platform restart is not required.

Additional parameters

You can set the following parameters as needed:

Steps:

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

Property	Required	Description
<code>aws.emr.resource.bucket</code>	Y	S3 bucket name where Trifacta executables, libraries, and other resources can be stored that are required for Spark execution.
<code>aws.emr.resource.path</code>	Y	S3 path within the bucket where resources can be stored for job execution on the EMR cluster. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">NOTE: Do not include leading or trailing slashes for the path value.</div>
<code>aws.emr.proxyUser</code>	Y	This value defines the user for the Trifacta users to use for connecting to the cluster. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">NOTE: Do not modify this value.</div>
<code>aws.emr.maxLogPollingRetries</code>	N	Configure maximum number of retries when polling for log files from EMR after job success or failure. Minimum value is 5.
<code>aws.emr.tempfilesCleanupAge</code>	N	Defines the number of days that temporary files in the <code>/trifacta/tempfiles</code> directory on EMR HDFS are permitted to age. By default, this value is set to 0, which means that cleanup is disabled. If needed, you can set this to a positive integer value. During each job run, the platform scans this directory for temp files older than the specified number of days and removes any that are found. This cleanup provides an additional level of system hygiene. Before enabling this secondary cleanup process, please execute the following command to clear the <code>tempfiles</code> directory: <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"><pre>hdfs dfs -rm -r -skipTrash /trifacta/tempfiles</pre></div>

Optional Configuration

Configure for Redshift

For more information on configuring the platform to integrate with Redshift, see *Amazon Redshift Connections*.

Configure for EMR high availability

The Designer Cloud powered by Trifacta platform can be configured to integrate with multiple master EMR nodes, which are deployed in a highly available environment.

Deploying additional instances of EMR may result in increased costs.

Versions

Integration with EMR high availability is supported for EMR 5.23.0 and later.

Permissions

An additional permission must be added to the ARN used to access EMR. For more information, see "Permissions for EMR high availability" above.

For more information, see <https://docs.aws.amazon.com/emr/latest/ManagementGuide/emr-plan-ha-launch.html>.

Create cluster

When you create the cluster, you must select the "Use multiple master nodes to improve cluster availability" checkbox. For more information, see "Create Clusters" above.

Switch EMR Cluster

If needed, you can switch to a different EMR cluster through the application. For example, if the original cluster suffers a prolonged outage, you can switch clusters by entering the cluster ID of a new cluster. For more information, see *Admin Settings Page*.

Configure Batch Job Runner

Batch Job Runner manages jobs executed on the EMR cluster. You can modify aspects of how jobs are executed and how logs are collected. For more information, see *Configure Batch Job Runner*.

Modify Job Tag Prefix

In environments where the EMR cluster is shared with other job-executing applications, you can review and specify the job tag prefix, which is prepended to job identifiers to avoid conflicts with other applications.

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 and modify if needed:

```
"aws.emr.jobTagPrefix": "TRIFACTA_JOB_",
```

3. Save your changes and restart the platform.

Testing

1. Load a dataset from the EMR cluster.
2. Perform a few simple steps on the dataset.
3. Click **Run** in the Transformer page.
4. When specifying the job:
 - a. Click the Profile Results checkbox.
 - b. Select **Spark**.
5. When the job completes, verify that the results have been written to the appropriate location.

Configure for AWS Databricks

Contents:

- *Prerequisites*
 - *Limitations*
 - *Supported versions of Databricks*
 - *Job Limits*
 - *Managing Limits*
 - *Enable*
 - *Configure*
 - *Configure cluster mode*
 - *Configure use of cluster policies*
 - *Configure Instance Profiles in AWS Databricks*
 - *Configure instance pooling*
 - *Configure Platform*
 - *Configure Databricks Job Management*
 - *Configure for Databricks Secrets Management*
 - *Enable shared cluster for Databricks Tables*
 - *Configure for Secrets Manager*
 - *Configure for Users*
 - *Configure AWS Databricks workspace overrides*
 - *Configure Databricks job throttling*
 - *Configure personal access token*
 - *Specify Databricks Tables cluster name*
 - *Configure maximum retries for REST API*
 - *Use*
 - *Run Job From Application*
 - *Run Job via API*
 - *Troubleshooting*
 - *Spark job on AWS Databricks fails with "Invalid spark version" error*
 - *Spark job fails with "spark scheduler cannot be cast" error*
-

This section provides high-level information on how to configure the Designer Cloud powered by Trifacta platform to integrate with Databricks hosted on AWS.

AWS Databricks is a unified data analytics platform that has been optimized for use on the AWS infrastructure.

- For more information, see <https://databricks.com/aws>.
- For documentation on AWS Databricks, see <https://databricks.com/documentation>.

Additional Databricks features supported by the platform:

- Credential passthrough (AWS Databricks only):
<https://docs.databricks.com/security/credential-passthrough/iam-passthrough.html>
- Table access control: <https://docs.databricks.com/security/access-control/table-acls/object-privileges.html>

Prerequisites

- The Designer Cloud powered by Trifacta platform must be installed in a customer-managed AWS environment.
- The base storage layer must be set to S3. For more information, see *Set Base Storage Layer*.
- AWS Secrets Manager is required for AWS Databricks use. For more information, see *Configure for AWS Secrets Manager*.

Limitations

- Import datasets created from nested folders is not supported for running jobs from AWS Databricks.
- If the job is submitted using the User cluster mode and no cluster is available, the following are the launch times for a new cluster with and without instance pools:
 - Without instance pools: Up to 5 minutes to launch
 - With instance pools : Up to 30 seconds to launch
- If the job is canceled during cluster startup:
 - The cluster startup continues. After the cluster is running, the job is terminated, and the cluster remains.
 - As a result, there is a delay in reporting the job cancellation in the Job Details page. The job should be reported as canceled not failed.
- AWS Databricks integration works with Spark 2.4.x, Spark 3.0.1, Spark 3.2.0, and Spark 3.2.1.

NOTE: The version of Spark for AWS Databricks must be applied to the platform configuration through the `databricks.sparkVersion` property. Details are provided later.

Supported versions of Databricks

- AWS Databricks 10.x
- AWS Databricks 9.1 LTS (**Recommended**)
- AWS Databricks 7.3 LTS

Job Limits

By default, the number of jobs permitted on an AWS workspace is set to 1000.

- The number of jobs that can be created per workspace in an hour is limited to 1000.
- The number of jobs a workspace can create in an hour is limited to 5000 when using the run-submit API. This limit also affects jobs created by the REST API and notebook tasks. For more information, see "Configure Databricks job management" below.
- The number of actively concurrent job runs in a workspace is limited to 150.

These limits apply to any jobs that use workspace data on the cluster.

Managing Limits

To enable retrieval and auditing of job information after a job has been completed, the Designer Cloud powered by Trifacta platform does not delete jobs from the cluster. As a result, jobs can accumulate over time to exceed the number of jobs permitted on the cluster. If you reach these limits, you may receive a `Quota for number of jobs has been reached limit`. For more information, see <https://docs.databricks.com/user-guide/jobs.html>.

Optionally, you can allow the Designer Cloud powered by Trifacta platform to manage your jobs to avoid these limitations. For more information, see "Configure Databricks job management" below.

Enable

To enable AWS Databricks, perform the following configuration changes:

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter, which enables Trifacta Photon for smaller job execution. Set it to `Enabled`:

Photon execution

- You do not need to save to enable the above configuration change.
- 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 parameters. Set them to the values listed below, which enables AWS Databricks (small to extra-large jobs) running environments:

```
"webapp.runInDatabricks": true,
"webapp.runWithSparkSubmit": false,
"webapp.runInDataflow": false,
```

- Do not save your changes until you have completed the following configuration section.

Configure

Configure cluster mode

When a user submits a job, the Designer Cloud Powered by Trifacta Enterprise Edition provides all the cluster specifications in the Databricks API and it creates cluster only for per-user or per-job, that means once the job is complete, the cluster is terminated. Cluster creation may take less than 30 seconds if instance pools are used. If the instance pools are not used, it may take 10-15 minutes.

For more information on job clusters, see <https://docs.databricks.com/clusters/configure.html>.

The job clusters automatically terminate after the job is completed. A new cluster is automatically created when the user next requests access to AWS Databricks access.

Cluster Mode	Description
USER	When a user submits a job, Designer Cloud Powered by Trifacta Enterprise Edition creates a new cluster and persists the cluster ID in Designer Cloud Powered by Trifacta Enterprise Edition metadata for the user if the cluster does not exist or invalid. If the user already has an existing interactive valid cluster, then the existing cluster is reused when submitting the job. Reset to JOB mode to run jobs in AWS Databricks.
JOB	When a user submits a job, Designer Cloud Powered by Trifacta Enterprise Edition provides all the cluster specifications in the Databricks API. Databricks creates a cluster only for this job and terminates it as soon as the job completes. Default cluster mode to run jobs in AWS Databricks.

Configure use of cluster policies

Optionally, you can configure the Designer Cloud powered by Trifacta platform to use the Databricks cluster policies that have been specified by your Databricks administrator for creating and using clusters. These policies are effectively templates for creation and use of Databricks clusters and govern aspects of clusters such as the type and count of nodes the resources that can be accessed via the cluster, and other settings. For more information on Databricks cluster policies, see <https://docs.databricks.com/administration-guide/clusters/policies.html>.

Prerequisites

NOTE: Your Databricks administrator must create and deploy the Databricks cluster policies from which Trifacta users can select for their personal use.

Notes:

- When this feature is enabled, each user may select the appropriate Databricks cluster policy to use for jobs. If none is selected by a user, jobs are launched without a cluster policy for the user using the Databricks properties set in platform configuration.

NOTE: Except for Spark version and cluster policy identifier in job-level overrides, other Databricks cluster configuration in the Designer Cloud powered by Trifacta platform is ignored when this feature is in use. Other job-level overrides are also ignored.

- If a cluster policy is modified and existing clusters are using it, then subsequent job executions using that policy attempt to use the same cluster. This can cause issues in performance and even job failures.

Tip: Avoid editing cluster policies that are in use, as these changed policies may be applied to clusters generated under the old policies. Instead, you should create a new policy and assign it for use.

- If the cluster policy references a Databricks instance pool that does not exist, the job fails.

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter and set it to `Enabled`:

```
Databricks Cluster Policies
```

3. Save your changes and restart the platform.

NOTE: Each user must select a cluster policy to use. For more information, see *Databricks Settings Page*.

Job overrides:

A user's cluster policy can be overridden when a job is executed via API. Set the request attribute for the `clusterPolicyId`.

NOTE: If a Databricks cluster policy is used, all job-level overrides except for `clusterPolicyId` are ignored.

For more information, see *API Task - Run Job*.

Policy template for AWS - without instance pools:

The following example cluster policy can provide a basis for creating your own AWS cluster policies when instance pools are not in use:

```
{
  "autoscale.max_workers": {
    "type": "fixed",
    "value": 3,
    "hidden": true
  },
  "autoscale.min_workers": {
    "type": "fixed",
    "value": 1,
    "hidden": true
  },
  "autotermination_minutes": {
    "type": "fixed",
    "value": 10,
  }
}
```

```

    "hidden": true
  },
  "aws_attributes.availability": {
    "type": "fixed",
    "value": "SPOT_WITH_FALLBACK",
    "hidden": false
  },
  "aws_attributes.ebs_volume_count": {
    "type": "fixed",
    "value": 0,
    "hidden": false
  },
  "aws_attributes.ebs_volume_size": {
    "type": "fixed",
    "value": 0,
    "hidden": false
  },
  "aws_attributes.first_on_demand": {
    "type": "fixed",
    "value": 1,
    "hidden": false
  },
  "aws_attributes.spot_bid_price_percent": {
    "type": "fixed",
    "value": 100,
    "hidden": false
  },
  "aws_attributes.instance_profile_arn": {
    "type": "fixed",
    "value": "arn:aws:iam::999999999999:instance-profile/SOME_Role_ARN",
    "hidden": false
  },
  "driver_node_type_id": {
    "type": "fixed",
    "value": "i3.xlarge",
    "hidden": true
  },
  "enable_local_disk_encryption": {
    "type": "fixed",
    "value": false
  },
  "node_type_id": {
    "type": "fixed",
    "value": "i3.xlarge",
    "hidden": true
  }
}

```

Policy template for AWS - without instance pools:

The following example cluster policy can provide a basis for creating your own AWS cluster policies when instance pools are in use:

```

{
  "autoscale.max_workers": {
    "type": "fixed",
    "value": 3,
    "hidden": true
  },
  "autoscale.min_workers": {
    "type": "fixed",
    "value": 1,
    "hidden": true
  },
  "aws_attributes.instance_profile_arn": {
    "type": "fixed",
    "value": "arn:aws:iam::999999999999:instance-profile/SOME_POLICY",
    "hidden": false
  }
}

```

```

},
"enable_local_disk_encryption": {
  "type": "fixed",
  "value": false
},
"instance_pool_id": {
  "type": "fixed",
  "value": "SOME_POOL",
  "hidden": true
},
"driver_instance_pool_id": {
  "type": "fixed",
  "value": "SOME_POOL",
  "hidden": true
},
"autotermination_minutes": {
  "type": "fixed",
  "value": 10,
  "hidden": true
},
}
}

```

Configure Instance Profiles in AWS Databricks

Designer Cloud powered by Trifacta platform EC2 instances can be configured with permissions to access AWS resources like S3 by attaching an IAM instance profile. Similarly, instance profiles can be attached to EC2 instances for use with AWS Databricks clusters.

NOTE: You must register the instance profiles in the Databricks workspace, or your Databricks clusters reject the instance profile ARNs and display an error. For more information, see <https://docs.databricks.com/administration-guide/cloud-configurations/aws/instance-profiles.html#step-5-add-the-instance-profile-to-databricks>.

To configure the instance profile for AWS Databricks, you must provide an IAM instance profile ARN in `awsAttributes.instanceProfileArn` parameter.

NOTE: For AWS Databricks, instance profiles are supported when the `aws.credentialProvider` is set to `instance` or `temporary`.

<code>aws.credentialProvider</code>	AWS Databricks permissions
<code>instance</code>	Designer Cloud powered by Trifacta platform or Databricks jobs gets all permissions directly from the instance profile.
<code>temporary</code>	Designer Cloud powered by Trifacta platform or Databricks jobs use temporary credentials that are issued based on system or user IAM roles. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>NOTE: The instance profile must have policies that allow Designer Cloud powered by Trifacta platform or Databricks to assume those roles.</p> </div>
<code>default</code>	n/a

NOTE: If the `aws.credentialProvider` is set to `temporary` or `instance` while using AWS Databricks:

- `databricks.awsAttributes.instanceProfileArn` must be set to a valid value for Databricks jobs to run successfully.
- `aws.ec2InstanceRoleForAssumeRole` flag is ignored for Databricks jobs.

For more information, see *Configure for AWS Authentication*.

Configure instance pooling

Instance pooling reduces cluster node spin-up time by maintaining a set of idle and ready instances. The Designer Cloud powered by Trifacta platform can be configured to leverage instance pooling on the AWS Databricks cluster for both worker and driver nodes.

NOTE: When instance pooling is enabled, the following parameters are not used:

```
databricks.driverNodeType
```

```
databricks.workerNodeType
```

For more information, see <https://docs.databricks.com/clusters/instance-pools/configure.html>.

Instance pooling for worker nodes

Prerequisites:

- All cluster nodes used by the Designer Cloud powered by Trifacta platform are taken from the pool. If the pool has an insufficient number of nodes, cluster creation fails.
- Each user must have access to the pool and must have at least the `ATTACH_TO` permission.
- Each user must have a personal access token from the same AWS Databricks workspace. See *Configure personal access token* below.

To enable:

1. Acquire your pool identifier or pool name from AWS Databricks.

NOTE: You can use either the Databricks pool identifier or pool name. If both `poolId` and `poolName` are specified, `poolId` is used first. If that fails to find a matching identifier, then the `poolName` value is checked.

Tip: If you specify a `poolName` value only, then you can run your Databricks jobs against the available clusters across multiple Trifacta workspaces. This mechanism allows for better resource allocation and broader execution options.

2. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Set either of the following parameters:
 - a. Set the following parameter to the AWS Databricks pool identifier:

```
"databricks.poolId": "<my_pool_id>",
```

- b. Or, you can set the following parameter to the AWS Databricks pool name:

```
"databricks.poolName": "<my_pool_name>",
```

4. Save your changes and restart the platform.

Instance pooling for driver nodes

The Designer Cloud powered by Trifacta platform can be configured to use Databricks instance pooling for driver pools.

To enable:

1. Acquire your driver pool identifier or driver pool name from Databricks.

NOTE: You can use either the Databricks driver pool identifier or driver pool name. If both `driverPoolId` and `driverPoolName` are specified, `driverPoolId` is used first. If that fails to find a matching identifier, then the `driverPoolName` value is checked.

Tip: If you specify a `driverPoolName` value only, then you can run your Databricks jobs against the available clusters across multiple Trifacta workspaces. This mechanism allows for better resource allocation and broader execution options.

2. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Set either of the following parameters:
 - a. Set the following parameter to the Databricks driver pool identifier:

```
"databricks.driverPoolId": "<my_pool_id>",
```

- b. Or, you can set the following parameter to the Databricks driver pool name:

```
"databricks.driverPoolName": "<my_pool_name>",
```

4. Save your changes and restart the platform.

Configure Platform

Review and modify the following configuration settings, as required:

NOTE: Restart the platform after you modify the configuration settings for the system to take affect.

Following is the list of parameters that have to be set to integrate the AWS Databricks with Designer Cloud powered by Trifacta platform :

Required Parameters

Parameter	Description	Value
<code>databricks.serviceUrl</code>	URL to the AWS Databricks Service where Spark jobs will be run	-
<code>metadata.cloud</code>	Must be set to <code>aws</code> and should not be changed to any other value while using AWS Databricks	Default: <code>aws</code>

Following is the list of parameters that can be reviewed or modified based on your requirements:

Optional Parameters

Parameter	Description	Value
<code>databricks.awsAttributes.firstOnDemandInstances</code>	Number of initial cluster nodes to be placed on on-demand instances. The remainder is placed on availability instances	Default: 1
<code>databricks.awsAttributes.availability</code>	Availability type used for all subsequent nodes past the <code>firstOnDemandInstances</code> .	Default: SPOT_WITH_FALLBACK
<code>databricks.awsAttributes.availabilityZone</code>	Identifier for the availability zone/datacenter in which the cluster resides. The provided availability zone must be in the same region as the Databricks deployment.	
<code>databricks.awsAttributes.spotBidPricePercent</code>	The max price for AWS spot instances, as a percentage of the corresponding instance type's on-demand price. When spot instances are requested for this cluster, only spot instances whose max price percentage matches this field will be considered.	Default: 100
<code>databricks.awsAttributes.ebsVolume</code>	The type of EBS volumes that will be launched with this cluster.	Default: None
<code>databricks.awsAttributes.instanceProfileArn</code>	EC2 instance profile ARN for the cluster nodes. This is only used when AWS credential provider is set to temporary/instance. The instance profile must have previously been added to the Databricks environment by an account administrator.	For more information, see <i>Configure for AWS Authentication</i> .
<code>databricks.clusterMode</code>	Determines the cluster mode for running a Databricks job.	Default: JOB
<code>feature.parameterization.matchLimitOnSampling.databricksSpark</code>	Maximum number of parameterized source files that are permitted for matching in a single dataset with parameters.	Default: 0
<code>databricks.workerNodeType</code>	Type of node to use for the AWS Databricks Workers/Executors. There are 1 or more Worker nodes per cluster.	Default: <code>i3.xlarge</code>
<code>databricks.sparkVersion</code>	AWS Databricks runtime version which also references the appropriate version of Spark.	Depending on your version of AWS Databricks, please set this property according to the following: <ul style="list-style-type: none"> AWS Databricks 10.x: <code>10.0.x-scala2.12</code> AWS Databricks 9.1 LTS: <code>9.1.x-scala2.12</code> AWS Databricks 7.3 LTS: <code>7.3.x-scala2.12</code> Please do not use other values.
<code>databricks.minWorkers</code>	Initial number of Worker nodes in the cluster, and also the minimum number of Worker nodes that the cluster can scale down to during auto-scale-down.	Minimum value: 1 Increasing this value can increase compute costs.
<code>databricks.maxWorkers</code>	Maximum number of Worker nodes the cluster can create during auto scaling.	Minimum value: Not less than <code>databricks.minWorkers</code> . Increasing this value can increase compute costs.
<code>databricks.poolId</code>	If you have enabled instance pooling in AWS Databricks, you can specify the pool identifier here.	<div style="border: 1px solid gray; padding: 5px;"> <p>NOTE: If both <code>poolId</code> and <code>poolName</code> are specified, <code>poolId</code> is used first. If that fails to find a matching identifier, then the <code>poolName</code> value is checked.</p> </div>

databricks.poolName	If you have enabled instance pooling in AWS Databricks, you can specify the pool name here.	See previous. Tip: If you specify a poolName value only, then you can use the instance pools with the same poolName available across multiple Databricks workspaces when you create a new cluster.
databricks.driverNodeType	Type of node to use for the AWS Databricks Driver. There is only one Driver node per cluster.	Default: <code>i3.xlarge</code> For more information, see the sizing guide for Databricks. NOTE: This property is unused when instance pooling is enabled. For more information, see Configure instance pooling below.
databricks.driverPoolId	If you have enabled instance pooling in AWS Databricks, you can specify the driver node pool identifier here. For more information, see Configure instance pooling below.	NOTE: If both driverPoolId and driverPoolName are specified, driverPoolId is used first. If that fails to find a matching identifier, then the driverPoolName value is checked.
databricks.driverPoolName	If you have enabled instance pooling in AWS Databricks, you can specify the driver node pool name here. For more information, see Configure instance pooling below.	See previous. Tip: If you specify a driverPoolName value only, then you can use the instance pools with the same driverPoolName available across multiple Databricks workspaces when you create a new cluster.
databricks.logsDestination	DBFS location that cluster logs will be sent to every 5 minutes	Leave this value as <code>/trifacta/logs</code> .
databricks.enableAutotermination	Set to true to enable auto-termination of a user cluster after N minutes of idle time, where N is the value of the autoterminationMinutes property.	Unless otherwise required, leave this value as <code>true</code> .
databricks.clusterStatePollerDelayInSeconds	Number of seconds to wait between polls for AWS Databricks cluster status when a cluster is starting up	
databricks.clusterStartupWaitTimeInMinutes	Maximum time in minutes to wait for a Cluster to get to Running state before aborting and failing an AWS Databricks job.	Default: 60
databricks.clusterLogSyncWaitTimeInMinutes	Maximum time in minutes to wait for a Cluster to complete syncing its logs to DBFS before giving up on pulling the cluster logs to the Trifacta node.	Set this to 0 to disable cluster log pulls.
databricks.clusterLogSyncPollerDelayInSeconds	Number of seconds to wait between polls for a Databricks cluster to sync its logs to DBFS after job completion.	Default: 20
databricks.autoterminationMinutes	Idle time in minutes before a user cluster will auto-terminate.	Do not set this value to less than the cluster startup wait time value.

databricks. maxAPICallRetries	Maximum number of retries to perform in case of 429 error code response	Default: 5. For more information, see Configure Maximum Retries for REST API section below.
databricks. enableLocalDiskEncryption	Enables encryption of data like shuffle data that is temporarily stored on cluster's local disk.	-
databricks. patCacheTTLInMinutes	Lifespan in minutes for the Databricks personal access token in-memory cache	Default: 10
spark. useVendorSparkLibraries	When <code>true</code> , the platform bypasses shipping its installed Spark libraries to the cluster with each job's execution.	<div style="border: 1px solid black; padding: 5px;"> <p>NOTE: This setting is ignored. The vendor Spark libraries are always used for AWS Databricks.</p> </div>

Configure Databricks Job Management

AWS Databricks enforces a hard limit of 1000 created jobs per workspace, and by default cluster jobs are not deleted. To support jobs more than 1000 jobs per cluster, you can enable job management for AWS Databricks.

NOTE: This feature covers the deletion of the job definition on the cluster, which counts toward the enforced limits. The Designer Cloud powered by Trifacta platform never deletes the outputs of a job or the job definition stored in the platform. When cluster job definitions are removed, the jobs remain listed in the Job History page, and job metadata is still available. There is no record of the job on the AWS Databricks cluster. Jobs continue to run, but users on the cluster may not be aware of them.

Tip: Regardless of your job management option, when you hit the limit for the number of job definitions that can be created on the Databricks workspace, the platform by default falls back to using the `runs /submit` API, if the Databricks Job Runs Submit Fallback setting has been enabled.

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following property and set it to one of the values listed below:

Databricks Job Management

Property Value	Description
Never Delete	(default) Job definitions are never deleted from the AWS Databricks cluster.
Always Delete	The AWS Databricks job definition is deleted during the clean-up phase, which occurs after a job completes.
Delete Successful Only	When a job completes successfully, the AWS Databricks job definition is deleted during the clean-up phase. Failed or canceled jobs are not deleted, which allows you to debug as needed.
Skip Job Creation	For jobs that are to be executed only one time, the Designer Cloud powered by Trifacta platform can be configured to use a different mechanism for submitting the job. When this option is enabled, the Designer Cloud powered by Trifacta platform submits jobs using the <code>run-submit</code> API, instead of the <code>run-now</code> API. The <code>run-submit</code> API does not create an AWS Databricks job definition. Therefore the submitted job does not count toward the enforced job limit.

Default	Inherits the default system-wide setting.
---------	---

- When this feature is enabled, the platform falls back to use the runs/submit API as a fallback when the job limit for the Databricks workspace has been reached:

```
Databricks Job Runs Submit Fallback
```

- Save your changes and restart the platform.

Configure for Databricks Secrets Management

Optionally, you can leverage Databricks Secrets Management to store sensitive Databricks configuration properties. When this feature is enabled and a set of properties are specified, those properties and their values are stored in masked form. For more information on Databricks Secrets Management, see <https://docs.databricks.com/security/secrets/index.html>.

Steps:

- 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 properties and set accordingly:

Setting	Description
<code>databricks.secretNamespace</code>	If multiple instances of Designer Cloud Powered by Trifacta Enterprise Edition are using the same Databricks cluster, you can specify the Databricks namespace to which these properties apply.
<code>databricks.secrets</code>	An array containing strings representing the properties that you wish to store in Databricks Secrets Management. For example, the default value stores a recommended set of Spark and Databricks properties: <pre>["spark.hadoop.dfs.adls.oauth2.client.id", "spark.hadoop.dfs.adls.oauth2.credential", "dfs.adls.oauth2.client.id", "dfs.adls.oauth2.credential", "fs.azure.account.oauth2.client.id", "fs.azure.account.oauth2.client.secret"],</pre> You can add or remove properties from this array list as needed.

- Save your changes and restart the platform.

Enable shared cluster for Databricks Tables

Optionally, you can provide to the Designer Cloud powered by Trifacta platform the name of a shared Databricks cluster to be used to access Databricks Tables.

Prerequisites:

NOTE: Any shared cluster must be maintained by the customer.

- Shared clusters are not provisioned per-user and cannot be used to run Spark jobs.
- If you have enabled table access control on your high-concurrency cluster, you must configure access to data objects for users. For more information, see <https://docs.databricks.com/security/access-control/table-acls/object-privileges.html>.
- If only a limited number of users are using the shared cluster, you must configure the attach permission on the shared cluster in the Databricks workspace.

Configure cluster

Depending on the credential provider type, the following properties must be specified in the Spark configuration for the shared cluster.

Default credential provider:

```
"fs.s3a.access.key"  
"fs.s3a.secret.key"  
"spark.hadoop.fs.s3a.access.key"  
"spark.hadoop.fs.s3a.secret.key"
```

For more information, see <https://docs.databricks.com/data/data-sources/aws/amazon-s3.html>.

Instance credential provider:

```
"fs.s3a.credentialsType"  
"fs.s3a.stsAssumeRole.arn"  
"fs.s3a.canned.acl"  
"fs.s3a.acl.default"  
"spark.hadoop.fs.s3a.credentialsType"  
"spark.hadoop.fs.s3a.stsAssumeRole.arn"  
"spark.hadoop.fs.s3a.canned.acl"  
"spark.hadoop.fs.s3a.acl.default"
```

For more information, see <https://docs.databricks.com/administration-guide/cloud-configurations/aws/instance-profiles.html>.

Temporary credential provider:

```
"fs.s3a.canned.acl"  
"fs.s3a.acl.default"  
"spark.hadoop.fs.s3a.canned.acl"  
"spark.hadoop.fs.s3a.acl.default"
```

For more information, see <https://docs.databricks.com/administration-guide/cloud-configurations/aws/assume-role.html>.

Enable

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 add the name of the Databricks cluster to use to browse Databricks Table:

```
"feature.databricks.connection.clusterName": "<your_cluster_name>",
```

3. Save your changes and restart the cluster.

When a cluster name is provided:

- If the cluster is available, all users of the Designer Cloud powered by Trifacta platform attempt to connect to Databricks Tables through the listed cluster.
- If the cluster has been terminated, a message indicates that the cluster must be restarted. After it has been restarted, you can try again.

- If the cluster name is invalid, the Designer Cloud powered by Trifacta platform fails any read/write operations to Databricks Tables.

NOTE: While using a single cluster shared across users to access Databricks Tables, each user must have a valid Databricks Personal Access Token to the shared cluster.

If a cluster name is not provided:

- In USER mode, the default behavior is to create one interactive cluster for each user to browse Databricks Tables.
- In JOB mode, the interactive cluster is created only when a user needs to browse Databricks Tables.

Configure user-level override for Databricks Tables access cluster

Individual users can specify the name of the cluster that accesses Databricks Tables through the Databricks settings. See "Specify Databricks Tables cluster name" below.

Configure for Secrets Manager

The AWS Secrets Manager is a secure vault for storing access credentials to AWS resources.

NOTE: AWS Secrets Manager is mandatory to use with AWS Databricks.

For more information, see *Configure for AWS Secrets Manager*.

Configure for Users

Configure AWS Databricks workspace overrides

A single AWS Databricks account can have access to multiple Databricks workspaces. You can create more than one workspace by using Account API if you are account is on the E2 version of the platform or on a selected custom plan that allows multiple workspaces per account.

For more information, see <https://docs.databricks.com/administration-guide/account-api/new-workspace.html>

Each workspace has a unique deployment name associated with it that defines the workspace URL. For example: `https://<deployment-name>.cloud.databricks.com` .

NOTE:

- The existing property `databricks.serviceUrl` is used to configure the URL to the Databricks Service to run Spark jobs.
- The `databricks.serviceUrl` defines the default Databricks workspace for all user in the Designer Cloud Powered by Trifacta Enterprise Edition workspace.
- Individual user can override this setting in the User Preferences in the Databricks Personal Access Token page.

For more information, see *Databricks Settings Page*.

For more information, see *Configure Platform* section above.

Configure Databricks job throttling

By default, Databricks workspaces apply limits on the number of jobs that can be submitted before the cluster begins to fail jobs. These limits are the following:

- Maximum number of concurrent jobs per cluster
- Max number of concurrent jobs per workspace
- Max number of concurrent clusters per workspace

Depending on how your clusters are configured, these limits can vary. For example, if the maximum number of concurrent jobs per cluster is set to 20, then the 21st concurrent job submitted to the cluster fails.

To prevent unnecessary job failure, the Designer Cloud powered by Trifacta platform submits the throttling of jobs to Databricks. When job throttling is enabled and the 21 concurrent job is submitted, the Designer Cloud powered by Trifacta platform holds that job internally the first of any of the following events happens:

- An active job on the cluster completes, and space is available for submitting a new job. The job is then submitted.
- The user chooses to cancel the job.
- One of the timeout limits described below is reached.

NOTE: The Designer Cloud powered by Trifacta platform supports throttling of jobs based on the maximum number of concurrent jobs per cluster. Throttling against the other limits listed above is not supported at this time.

Steps:

Please complete the following steps to enable job throttling.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. In the Designer Cloud application, select **User menu > Admin console > Admin settings**.
3. Locate the following settings and set their values accordingly:

Setting	Description
<code>databricks.userClusterThrottling.enabled</code>	When set to <code>true</code> , job throttling per Databricks cluster is enabled. Please specify the following settings.
<code>databricks.userClusterthrottling.maxTokensAllottedPerUserCluster</code>	Set this value to the maximum number of concurrent jobs that can run on one user cluster. Default value is 20.
<code>databricks.userClusterthrottling.tokenExpiryInMinutes</code>	The time in minutes after which tokens reserved by a job are revoked, irrespective of the job status. If a job is in progress and this limit is reached, then the Databricks token is expired, and the token is revoked under the assumption that it is stale. Default value is 120 (2 hours). <div style="border: 1px solid #c6e0b4; padding: 5px; margin-top: 10px;">Tip: Set this value to 0 to prevent token expiration. However, this setting is not recommended, as jobs can remain in the queue indefinitely.</div>
<code>jobMonitoring.queuedJobTimeoutMinutes</code>	The maximum time in minutes in which a job is permitted to remain in the queue for a slot on Databricks cluster. If this limit is reached, the job is marked as failed.
<code>batch-job-runner.cleanup.enabled</code>	When set to <code>true</code> , the Batch Job Runner service is permitted to clean up throttling tokens and job-level personal access tokens. <div style="border: 1px solid #c6e0b4; padding: 5px; margin-top: 10px;">Tip: Unless you have reason to do otherwise, you should leave this setting to <code>true</code>.</div>

4. Save your changes and restart the platform.

Configure personal access token

Each user must insert a Databricks Personal Access Token to access Databricks resources. For more information, see *Databricks Settings Page*.

Specify Databricks Tables cluster name

Individual users can specify the name of the cluster to which they are permitted to access Databricks Tables. This cluster can also be shared among users. For more information, see *Databricks Settings Page*.

Configure maximum retries for REST API

There is a limit of 30 requests per second per workspace on the Databricks REST APIs. If this limit is reached, then a HTTP status code 429 error is returned, indicating that rate limiting is being applied by the server. By default, the Designer Cloud powered by Trifacta platform re-attempts to submit a request 5 times and then fails the job if the request is not accepted.

If you want to change the number of retries, change the value for the `databricks.maxAPICallRetries` flag.

Value	Description
5	(default) When a request is submitted through the AWS Databricks REST APIs, up to 5 retries can be performed in the case of failures. <ul style="list-style-type: none">The waiting period increases exponentially for every retry. For example, for the 1st retry, the wait time is 10 seconds, 20 seconds for the next retry, 40 seconds for the third retry and so on.You can set the values accordingly based on number of minutes /seconds you want to try.
0	When an API call fails, the request fails. As the number of concurrent jobs increases, more jobs may fail. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;">NOTE: This setting is not recommended.</div>
5+	Increasing this setting above the default value may result in more requests eventually getting processed. However, increasing the value may consume additional system resources in a high concurrency environment and jobs might take longer to run due to exponentially increasing waiting time.

Use

Run Job From Application

When the above configuration has been completed, you can select the running environment through the application.

NOTE: When a Databricks job fails, the failure is reported immediately in the Designer Cloud application . In the background, the job logs are collected from Databricks and may not be immediately available.

See *Run Job Page*.

Run Job via API

You can use API calls to execute jobs.

Make sure that the request body contains the following:

```
"execution": "databricksSpark",
```

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/runJobGroup>

Troubleshooting

Spark job on AWS Databricks fails with "Invalid spark version" error

When running a job using Spark on AWS Databricks, the job may fail with the above invalid version error. In this case, the Databricks version of Spark has been deprecated.

Solution:

Since an AWS Databricks cluster is created for each user, the solution is to identify the cluster version to use, configure the platform to use it, and then restart the platform.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Acquire the value for `databricks.sparkVersion`.
3. In AWS Databricks, compare your value to the list of supported AWS Databricks version. If your version is unsupported, identify a new version to use.

NOTE: Ensure to note the version of Spark supported for the version of AWS Databricks that you have chosen.

4. In the Designer Cloud powered by Trifacta platform configuration, Set `databricks.sparkVersion` to the new version to use.

NOTE: The value for `spark.version` does not apply to Databricks.

5. Restart the Designer Cloud powered by Trifacta platform .
6. The platform is restarted. A new AWS Databricks cluster is created for each user using the specified values, when the user runs a job.

Spark job fails with "spark scheduler cannot be cast" error

When you run a job on Databricks, the job may fail with the following error:

```
java.lang.ClassCastException: org.apache.spark.scheduler.ResultTask cannot be cast to org.apache.spark.scheduler.Task
at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:616)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at java.lang.Thread.run(Thread.java:748)
```

The `job.log` file may contain something similar to the following:

```
2022-07-19T15:41:24.832Z - [sid=0cf0cff5-2729-4742-a7b9-4607ca287a98] - [rid=83eb9826-fc3b-4359-8e8f-7fbf77300878] - [Async-Task-9] INFO com.trifacta.databricks.spark.JobHelper - Got error org.apache.spark.SparkException: Stage 0 failed. Error: Job aborted due to stage failure: Task 0 in stage 0.0 failed 4 times, most recent failure: Lost task 0.3 in stage 0.0 (TID 3, ip-10-243-149-238.eu-west-1.compute.internal, executor driver): java.lang.ClassCastException: org.apache.spark.scheduler.ResultTask cannot be cast to org.apache.spark.scheduler.Task
at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:616)
...
```

This error is due to a class mismatch between the Designer Cloud powered by Trifacta platform and Databricks.

Solution:

The solution is to disable the precedence of using the Spark JARs provided from the Designer Cloud powered by Trifacta platform over the Databricks Spark JARs. Please perform the following steps:

1. To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the `spark.props` section and add the following configuration elements:

```
"spark": {  
  ...  
  "props": {  
    "spark.driver.userClassPathFirst": false,  
    "spark.executor.userClassPathFirst": false,  
    ...  
  }  
},
```

3. Save your changes and restart the platform.

Configure for Azure

Contents:

- Prerequisites
- Configure Azure
 - Create registered application
- Configure the Platform
 - Configure for Azure Databricks
 - Configure base storage layer
 - Configure for Key Vault
 - Configure for SSO
 - Configure for ADLS Gen2
 - Configure for ADLS Gen1
 - Configure for WASB
 - Configure for Azure Gov Cloud
 - Configure relational connections
- Testing

Please complete the following steps in the listed order to configure your installed instance of the Designer Cloud powered by Trifacta® platform to integrate with the running environment cluster.

Prerequisites

1. Deploy running environment cluster and Trifacta node.

NOTE: The running environment cluster can be deployed as part of the installation process. You can also integrate the platform with a pre-existing cluster. Details are below.

2. Install Designer Cloud powered by Trifacta platform on the node.

For more information, see *Install for Azure*.

Configure Azure

Create registered application

You must create an Azure Active Directory (AAD) application and grant it the desired access permissions, such as read/write access to resources and read/write access to the Azure Key Vault secrets.

NOTE: If you are integrating with Azure Databricks and are Managed Identities for authentication, please skip this section. That configuration is covered in a later step.

This service principal is used by the Designer Cloud powered by Trifacta platform for access to all Azure resources. For more information, see <https://docs.microsoft.com/en-us/azure/azure-resource-manager/resource-group-create-service-principal-portal>.

After you have registered, acquire the following information:

Azure Property	Location	Use
Application		

ID	Acquire this value from the Registered app blade of the Azure Portal.	Applied to Designer Cloud powered by Trifacta platform configuration: <code>azure.applicationid</code> .
Service User Key	Create a key for the Registered app in the Azure Portal.	Applied to Designer Cloud powered by Trifacta platform configuration: <code>azure.secret</code> . NOTE: If you are using Azure AD to integrate with an Azure Databricks cluster, the Azure AD secret value stored in <code>azure.secret</code> must begin with an alphanumeric character. This is a known issue.
Directory ID	Copy the Directory ID from the Properties blade of Azure Active Directory.	Applied to Designer Cloud powered by Trifacta platform configuration: <code>azure.directoryId</code> .

To create an Azure Active Directory (AAD) application, please complete the following steps in the Azure console.

Steps:

1. Create registered application:
 - a. In the Azure console, navigate to **Azure Active Directory > App Registrations**.
 - b. Create a New App. Name it `trifacta`.

NOTE: Retain the Application ID and Directory ID for configuration in the Designer Cloud powered by Trifacta platform .

2. Create a client secret:
 - a. Navigate to **Certificates & secrets**.
 - b. Create a new Client secret.

NOTE: Retain the value of the Client secret for configuration in the Designer Cloud powered by Trifacta platform .

3. Add API permissions:
 - a. Navigate to **API Permissions**.
 - b. Add Azure Key Vault with the `user_impersonation` permission.

These properties are applied later in the configuration process.

Configure the Platform

Configure for Azure Databricks

You can integrate the Designer Cloud powered by Trifacta platform with Azure Databricks. For more information, see *Configure for Azure Databricks*.

Configure base storage layer

For Azure installations, you can set your base storage layer to be HDFS or WASB.

NOTE: The base storage layer must be set after installation. After it has been configured, it cannot be modified.

Azure storage	webapp.storageProtocol setting

WASB	wasbs
ADLS Gen2	abfss
ADLS Gen1	adl

See *Set Base Storage Layer*.

Configure for Key Vault

For authentication purposes, the Designer Cloud powered by Trifacta platform must be integrated with an Azure Key Vault keystore. See *Configure Azure Key Vault*.

Configure for SSO

If needed, you can integrate the Designer Cloud powered by Trifacta platform with Azure AD for Single-Sign On to the platform. See *Configure SSO for Azure AD*.

Configure for ADLS Gen2

Enable read-only or read-write access to ADLS Gen2. For more information, see *ADLS Gen2 Access*.

Configure for ADLS Gen1

Enable read-only or read-write access to ADLS Gen1. For more information, see *ADLS Gen1 Access*.

Configure for WASB

Enable read-only or read-write access to WASB. For more information on integrating with WASB, see *WASB Access*.

Configure for Azure Gov Cloud

To enable use of the Azure Gov Cloud, please perform the following configuration steps.

NOTE: Managed Identities is not supported for Azure Gov Cloud.

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 `US_GOV`:

```
"azure.environment": "US_GOV",
```

3. Save your changes and restart the platform.

Configure relational connections

If you are integrating Designer Cloud Powered by Trifacta Enterprise Edition with relational datastores, please complete the following configuration sections.

Create encryption key file

An encryption key file must be created on the Trifacta node. This key file is shared across all relational connections. See *Create Encryption Key File*.

Create Azure SQL Database connection

For more information, see *Azure SQL Database Connections*.

Create Azure SQL DW connection

For more information, see *Microsoft SQL Data Warehouse Connections*.

Testing

1. Load a dataset from the cluster.
2. Perform a few simple steps on the dataset.
3. Click **Run** in the Transformer page.
4. When specifying the job:
 - a. Click the Profile Results checkbox.
 - b. Select **Spark**.
5. When the job completes, verify that the results have been written to the appropriate location.

Configure Azure Key Vault

Contents:

- *Create a Key Vault resource in Azure*
 - *Create Key Vault in Azure*
 - *Enable Key Vault access for the Designer Cloud powered by Trifacta platform*
- *Configure Key Vault for WASB*
 - *Create WASB access token*
 - *Configure Key Vault key and secret for WASB*
- *Configure Key Vault for ADLS*
- *Configure the Platform*
 - *Configure Key Vault location*
 - *Apply SAS token identifier for WASB*
 - *Configure Secure Token Service*

For authentication purposes, the Designer Cloud powered by Trifacta® platform must be integrated with an Azure Key Vault keystore.

- For more information, see <https://azure.microsoft.com/en-us/services/key-vault/>.

Please complete the following sections to create and configure your Azure Key Vault.

Create a Key Vault resource in Azure

Please complete the following steps in the Azure portal to create a Key Vault and to associate it with the Trifacta registered application.

NOTE: A Key Vault is required for use with the Designer Cloud powered by Trifacta platform .

Create Key Vault in Azure

Steps:

1. Log into the Azure portal.
2. Goto: <https://portal.azure.com/#create/Microsoft.KeyVault>
3. Complete the form for creating a new Key Vault resource:
 - a. Name: Provide a reasonable name for the resource. Example:

```
<clusterName>-<applicationName>-<group/organizationName>
```

Or, you can use `trifacta`.

- b. Location: Pick the location used by the cluster.
 - c. For other fields, add appropriate information based on your enterprise's preferences.
4. To create the resource, click **Create**.

NOTE: Retain the DNS Name value for later use.

Enable Key Vault access for the Designer Cloud powered by Trifacta platform

Steps:

In the Azure portal, you must assign access policies for application principal of the Trifacta registered application to access the Key Vault.

Steps:

1. In the Azure portal, select the Key Vault you created. Then, select **Access Policies**.
2. In the Access Policies window, select the Trifacta registered application.
3. Click **Add Access Policy**.
4. Select the following secret permissions (at a minimum):
 - a. Get
 - b. Set
 - c. Delete
 - d. Recover
5. Select the Trifacta application principal.
6. Assign the policy you just created to that principal.

Configure Key Vault for WASB

Please complete the following steps for using the Key Vault with WASB.

Create WASB access token

If you are enabling access to WASB, you must create this token within the Azure Portal.

For more information, see

<https://docs.microsoft.com/en-us/rest/api/storageservices/delegating-access-with-a-shared-access-signature>.

You must specify the storage protocol (`wasbs`) used by the Designer Cloud powered by Trifacta platform .

Configure Key Vault key and secret for WASB

For WASB, you must create key and secret values in the Key Vault that match other values in your Azure configuration. To enable access to the Key Vault, you must specify your key and secret values as follows:

Item	Applicable Configuration
key	The value of the key must be specified as the <code>sasTokenId</code> in the Designer Cloud powered by Trifacta platform .
secret	The value of the secret should match the shared access signature for your storage. This value is specified as <code>sasToken</code> in the Designer Cloud powered by Trifacta platform .

Acquire shared access signature value:

In the Azure portal, please do the following:

1. Open your storage account.
2. Select **Shared Access Signature** .
3. Generate or view existing signatures.
4. For a new or existing signature, copy the SAS token value. Omit the leading question mark (?).
5. Paste this value into a text file for safekeeping.

Create a custom key:

To create a custom key and secret pair for WASB use by the Designer Cloud powered by Trifacta platform , please complete the following steps:

1. On an existing or newly created Azure Key Vault resource, click **Secrets**.
2. At the top of the menu, click **Generate/Import**.
3. In the Create a secret menu:
 - a. Select **Manual** for upload options.
 - b. Chose an appropriate name for the key.

NOTE: Please retain the name of the key for later use, when it is applied through the Designer Cloud powered by Trifacta platform as the `sasTokenId` value. Instructions are provided later.

- c. Paste the SAS token value for the key into the secret field.
- d. Click **Create**.

Configure Key Vault for ADLS

For ADLS Gen1 and ADLS Gen2, the Designer Cloud powered by Trifacta platform creates its own key-secret combinations in the Key Vault. No additional configuration is required.

Configure the Platform

Configure Key Vault location

The location of the Azure Key Vault must be specified for the Designer Cloud powered by Trifacta platform . The location can be found in the properties section of the Key Vault resource in the Azure portal.

Steps:

1. Log in to the Azure portal.
2. Select the Key Vault resource.
3. Click **Properties**.
4. Locate the DNS Name field. Copy the field value.

This value is the location for the Key Vault. It must be applied in the Designer Cloud powered by Trifacta platform .

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. Specify the URL in the following parameter:

```
"azure.keyVaultURL": "<your key value URL>",
```

Apply SAS token identifier for WASB

If you are using WASB as your base storage layer, you must apply the SAS token value into the configuration of the Designer Cloud powered by Trifacta platform .

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. Paste the value of the SAS Token for the key you created in the Key Vault as the following value:

```
"azure.wasb.defaultStore.sasTokenId": "<your Sas Token Id>",
```

3. Save your changes.

Configure Secure Token Service

Access to the Key Vault requires use of the secure token service (STS) from the Designer Cloud powered by Trifacta platform . To use STS with Azure, several properties must be specified. For more information, see *Configure Secure Token Service*.

Configure SSO for Azure AD

Contents:

- *Prerequisites*
 - *Limitations*
 - *Configure Azure AD for Designer Cloud powered by Trifacta platform*
 - *Azure Key Vault Permissions*
 - *Configure Designer Cloud powered by Trifacta platform for Azure AD*
 - *Azure AD Properties*
 - *Configure session timeout page*
 - *User Management*
 - *Configure auto-registration*
 - *Provision new users under SSO without auto-registration*
 - *Disable user*
 - *User Access*
 - *SSO Relational Connections*
-

When the Designer Cloud powered by Trifacta® platform is deployed on Azure, it can be configured to provide single sign-on (SSO) with Azure AD (Active Directory) authentication management. Use this section to enable auto-logins for Azure users.

- If auto-provisioning is not desired, after completing the basic configuration, you can disable auto-provisioning using the steps listed in the Advanced Configuration section.
- Single Sign-On (SSO) authentication enables users to authenticate one time to access multiple systems. The SSO platform must translate its authentication into authentication methods executed against each system under SSO control. For more information, see https://en.wikipedia.org/wiki/Single_sign-on.

Supported authentication models:

Users can authenticate with the Designer Cloud powered by Trifacta platform using Azure AD accounts in the following scenarios:

- Azure AD is the identity provider,
- Azure AD is federated through a trust setup with a supported external identity provider,
- Azure AD is federated with on-premises Active Directory and Active Directory Federation Services (ADFS).

Azure Data Lake Store: Users can obtain OAuth access and refresh tokens from AzureAD and use the tokens to access.

Azure Databricks Clusters: If you have integrated with an Azure Databricks cluster, please complete this configuration to enable SSO authentication for Azure. No additional configuration is required to enable SSO for Azure Databricks.

Prerequisites

1. You have installed the Designer Cloud powered by Trifacta platform on Microsoft Azure. See *Install for Azure*.
2. You have performed the basic configuration for Azure integration. See *Configure for Azure*.
3. Your enterprise uses Azure SSO for User Identity and Authentication.
4. The Designer Cloud powered by Trifacta platform must be registered as a Service Provider in your Azure AD tenant.
5. Please acquire the following Service Provider properties:
 - a. The Service Provider Application ID (Client ID) and Key (Secret) are used for user authentication to the Azure Key Vault, Azure AD, and Azure Data Lake Store (if connected). These properties are

specified in the Designer Cloud powered by Trifacta platform as part of the basic Azure configuration.

NOTE: The Designer Cloud powered by Trifacta platform must be assigned the Reader role for the Azure Key Vault. Other permissions are also required. See the Azure Key Vault Permissions section below.

- b. The Service Provider Reply URL provides the redirect URL after the user has authenticated with Azure AD.
- c. The Service Provider should be granted Delegated permissions to the Windows Azure Service Management API so it can access Azure Service Management as organization users.

Limitations

Scheduled jobs are run under the access keys for the user who initially created the schedule. They continue to run as scheduled until those keys are explicitly revoked by an admin.

NOTE: With Azure SSO enabled, use of custom dictionaries is not supported.

Configure Azure AD for Designer Cloud powered by Trifacta platform

Please verify or perform the following configurations through Azure.

Azure Key Vault Permissions

For the Azure Key Vault:

- The Trifacta application must be assigned the Reader permission to the key vault.
- For the Key Vault Secrets, the application must be assigned the Set, Get, and Delete permissions.

Configure Designer Cloud powered by Trifacta platform for Azure AD

Azure AD Properties

Please configure the following properties.

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

Property	Description
<code>azure.sso.enabled</code>	Set this value to <code>true</code> to enable Azure AD Single Sign-On. The Designer Cloud powered by Trifacta platform authenticates users through enterprise Azure AD.
<code>azure.sso.redirectUrl</code>	Set this value to the redirect URL callback configured for this Azure AD application in the Azure portal. The URL is in the following format: <pre>https://<trifacta-app-host>/sign-in/azureCallback</pre>
<code>azure.sso.allowHttpForRedirectUrl</code>	When <code>true</code> , the <code>redirectUrl</code> can be specified as an insecure, non-HTTPS value. Default is <code>false</code> .
<code>azure.sso.enableAutoRegistration</code>	

	Set this value to <code>true</code> to enable SSO users to automatically register and login to the Trifacta application when they connect.
azure.resourceURL	<p>This value defines the Azure AD resource for which to obtain an access token.</p> <div style="border: 1px solid #ccc; padding: 10px; margin: 10px 0;"> <p>NOTE: By default, this value is <code>https://graph.windows.net/</code>. You can select other values from the drop-down in the Admin Settings page.</p> </div> <p>When using Azure Data Lake:</p> <ol style="list-style-type: none"> 1. In the Azure Portal, grant to the Trifacta application ID the Azure Data Lake API permission. 2. Set this value to <code>https://datalake.azure.net/</code>. 3. Sign out of the Designer Cloud application and sign in again.

Configure Azure Active Directory endpoint and authority

By default, the Designer Cloud powered by Trifacta platform uses the public Azure AD endpoint for brokering single sign-on requests. As needed, you can configure the platform to submit these requests to a different authority., such as Azure Gov Cloud.

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

Property	Description
azure.AADEndpoint	<p>Azure Active Directory endpoint and authority.</p> <ul style="list-style-type: none"> • Defaults to the Azure public commercial endpoint: <code>login.microsoftonline.com</code>. • Value must include a hostname with no protocol, port, or path. • Do not include any slashes.

Configure session timeout page

By default under SSO, manual logout and session expiration logout redirect to different pages. Manual logout directs you to SAML sign out, and session expiry produces a session expired page.

If desired, you can redirect the user to a different URL on session expiry:

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. Specify the URL of the page to which you wish to redirect users after a session has timed out:

```
"webapp.session.redirectUriOnExpiry": "<myPreferredSessionExpiryURL>",
```

3. Save your changes and restart the platform.

User Management

Tip: After SSO is enabled, the first AD user to connect to the platform is automatically registered as an admin user.

Configure auto-registration

Enabling auto-registration:

Auto-registration must be enabled for the Designer Cloud powered by Trifacta platform and for Azure AD SSO specifically.

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

Property	Description
<code>webapp.sso.enableAutoRegistration</code>	This property has no effect in Azure.
<code>azure.sso.enableAutoRegistration</code>	Set this value to <code>true</code> . For more information, see Azure AD Properties above.

How users are managed depends on whether auto-registration is enabled:

- If auto-registration is enabled, after users provide their credentials, the account is automatically created for them.
- If auto-registration is disabled, a Trifacta administrator must still provision a user account before it is available. See below.

Enabled:

After SSO with auto-registration has been enabled, you can still manage users through the Users page, with the following provisions:

- The Designer Cloud powered by Trifacta platform does not recheck for attribute values on each login. If attribute values change in LDAP, they must be updated in the Designer Cloud application, or the user must be deleted and recreated through auto-provisioning.
- If the user has been removed from AD, the user cannot sign in to the platform.
- If you need to remove a user from the platform, you should consider just disabling the user.

For more information, see *Users Page*.

Disabled:

To disable auto-provisioning in the platform, please verify the following property:

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

```
"webapp.sso.enableAutoRegistration" : false,
```

3. Save your changes and restart the platform.
4. New users of the Designer Cloud powered by Trifacta platform must be provisioned by a Trifacta administrator. See below.

Provision new users under SSO without auto-registration

If SSO auto-registration is disabled, admin users can provision new users of the platform through the following URL:

```
https://<hostname>/register
```

```
http://<host_name>:<port_number>/register
```

- The user's password is unnecessary in an SSO environment. You must provide the SSO principal value, which is typically the Active Directory login for the user.
- If you are connected to a Hadoop cluster, you must provision the Hadoop principal value.

- See *Create User Account*.
- Admin accounts can be created through the application. See *Create Admin Account*.

Disable user

If a user has been disabled in Azure AD, a Trifacta administrator must disable the user in the Designer Cloud application . Otherwise, the user can still access the Designer Cloud application until the user's access token expires.

For more information on disabling user accounts, see *Users Page*.

User Access

Users access the application through the Trifacta login page:

```
https://<hostname>
```

SSO Relational Connections

For more information, see *Enable SSO for Azure Relational Connections*.

Enable SSO for Azure Relational Connections

Contents:

- *Prerequisites*
 - *Limitations*
 - *Configure Azure AD for Designer Cloud powered by Trifacta platform*
 - *Configure Designer Cloud powered by Trifacta platform for Azure AD*
 - *Define scope*
 - *Enable SSO credential type*
 - *Create Connections*
 - *User Access*
-

You can extend the basic SSO integration between the Designer Cloud powered by Trifacta® platform and the Azure infrastructure to include SSO connections to Azure-based relational sources.

Supported relational connection types:

- Azure SQL Database
- SQL Datawarehouse

Prerequisites

- SSO integration to Azure AD must be enabled. See *Configure SSO for Azure AD*.

Limitations

1. Sharing of Azure connections is supported in the following manner:
 - a. Non-SSO Azure connections: Shared normally, with or without credentials.
 - b. SSO Azure connections:
 - i. The connection can be shared, but the credentials cannot.
 - ii. If the user who is shared the connection attempts to use it, that user's SSO principal is used. If that SSO principal has the same permissions as the original user, then the connection is fully operational. If not, then some data may not be accessible.
2. Write operations to SQL Datawarehouse are not supported for Azure SSO connections.

Configure Azure AD for Designer Cloud powered by Trifacta platform

Your Azure admin must enable the following:

1. Your SQL Server database must have an Active Directory Admin assigned to it.
 - a. This assignment must be applied for SQL DB and SQL DW connections.
2. Each user that is creating and using SQL Server connections over SSO must have a corresponding account in the SQL Server database.
3. To the Azure AD application, the "Azure SQL Database - user impersonation" permissions must be added.

For more information, please contact your Azure administrator.

Configure Designer Cloud powered by Trifacta platform for Azure AD

Define scope

You can define the scope of access in either of the following ways:

1. The Azure admin can manually define access for individual databases, or:
2. You can do the following on the Trifacta node:
 - a. SSH to the Trifacta node. Login as an administrator.
 - b. Navigate to the following:

```
/opt/trifacta/conf/
```

- c. Open `trifacta-conf.json`.
- d. Locate the `azure.sso.scope` property. Add this value to the property:

```
"https://database.windows.net/user_impersonation"
```

It is the second line in the following:

NOTE: If there are now multiple values in the entry, a comma must be placed after every line except for the last one.

```
{
  "azure": {
    "sso": {
      "scope": [
        "https://datalake.azure.net/user_impersonation",
        "https://database.windows.net/user_impersonation"
      ]
    }
  }
}
```

- e. Save the file.

Enable SSO credential type

NOTE: This configuration applies only for SQL DW connections. However, even if you are not creating these connections immediately, you should perform this configuration change.

When you create Azure SSO relational connections, you must select `azureTokenSso` for the credential type.

- For SQL DB connections, this selection is automatically enabled.
- For SQL DW connections, you must specify that this option is available by making a manual edit to a file on the Trifacta node.

Steps:

1. SSH to the Trifacta node. Login as an administrator.
2. Navigate to the following directory:

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

3. Edit `connection-metadata.json`.
4. Locate the `credentialType` property. Set the value to `azureTokenSso`.
5. Save your changes and restart the platform.

Create Connections

When you create a relational connection where Azure SSO has been enabled, select `Azure Token SSO` from the `Credential Type` drop-down.

NOTE: The SSO principal of the user who is creating or accessing the connection is used to connect to the specified database.

- See *Azure SQL Database Connections*.
- See *Microsoft SQL Data Warehouse Connections*.

User Access

Users can access the connections through the Import Data page. See *Import Data Page*.

ADLS Gen2 Access

Contents:

- *Limitations*
 - *Read-only access*
- *Prerequisites*
 - *General*
 - *Create a registered application*
 - *Azure properties*
 - *Key Vault Setup*
- *Configure the Designer Cloud powered by Trifacta platform*
 - *Define base storage layer*
 - *Review Java VFS Service*
 - *Configure file storage protocols and locations*
 - *Configure access mode*
- *Testing*
- *Troubleshooting*
 - *Problem: SSLHandshakeException : Unsupported curvel: 29 error when retrieving Databricks token*
- *Using ADLS Gen 2 Connection*
 - *Uses of ADLS*
 - *Before you begin using ADLS*
 - *Secure access*
 - *Storing data in ADLS*
 - *Reading from sources in ADLS*
 - *Creating datasets*
 - *Writing job results*
 - *Creating a new dataset from results*
- *Reference*

Deployments of the Designer Cloud powered by Trifacta® platform on Microsoft Azure can integrate with the next generation of Azure Data Lake Store (ADLS Gen2).

- **Microsoft Azure Data Lake Store Gen2 (ADLS Gen2)** combines the power of a high-performance file system with massive scale and economy. Azure Data Lake Storage Gen2 extends Azure Blob Storage capabilities and is optimized for analytics workloads.
- For more information, see <https://azure.microsoft.com/en-us/services/storage/data-lake-storage/>.

Supported Environments:

Operation	Designer Cloud Powered by Trifacta Enterprise Edition	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if ABFSS is base storage layer)

Limitations

A single public connection to ADLS Gen2 is supported.

Read-only access

If the base storage layer has been set to WASB, you can follow these instructions to set up read-only access to ADLS Gen2.

NOTE: To enable read-only access to ADLS Gen2, do not set the base storage layer to `abfss`.

Prerequisites

General

- The Designer Cloud powered by Trifacta platform has already been installed and integrated with an Azure Databricks cluster. See *Configure for Azure Databricks*.
- For each combination of blob host and container, a separate Azure Key Vault Store entry must be created. For more information, please contact your Azure admin.
- When running against ADLS Gen2, the product requires that you create a filesystem object in your ADLS Gen2 storage account.
- ABFSS must be set as the base storage layer for the Designer Cloud powered by Trifacta platform instance. See *Set Base Storage Layer*.

Create a registered application

Before you integrate with Azure ADLS Gen2, you must create the Designer Cloud powered by Trifacta platform as a registered application. See *Configure for Azure*.

Azure properties

The following properties should already be specified in the Admin Settings page. Please verify that the following have been set:

- `azure.applicationId`
- `azure.secret`
- `azure.directoryId`

The above properties are needed for this configuration.

Tip: ADLS Gen2 also works if you are using Azure Managed Identity.

Registered application role

NOTE: The Storage Blob Data Contributor role or its equivalent roles must be assigned in the ADLS Gen2 storage account.

For more information, see *Configure for Azure*.

Key Vault Setup

An Azure Key Vault has already been set up and configured for use by the Designer Cloud powered by Trifacta platform. Properties must be specified in the platform, if they have not been configured already.

For more information on configuration for Azure key vault, see *Configure for Azure*.

Configure the Designer Cloud powered by Trifacta platform

Define base storage layer

Per earlier configuration:

- `webapp.storageProtocol` must be set to `abfss`.

NOTE: Base storage layer must be configured when the platform is first installed and cannot be modified later.

NOTE: To enable read-only access to ADLS Gen2, do not set the base storage layer to `abfss`.

- `hdfs.protocolOverride` is ignored.

See *Set Base Storage Layer*.

Review Java VFS Service

Use of ADLS Gen2 requires the Java VFS service in the Designer Cloud powered by Trifacta platform .

NOTE: This service is enabled by default.

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

Configure file storage protocols and locations

The Designer Cloud powered by Trifacta platform must be provided the list of protocols and locations for accessing ADLS Gen2 blob storage.

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": ["abfss"],  
"fileStorage.defaultBaseUri": ["abfss://filesystem@storageaccount.dfs.core.windows.net/"],
```

Parameter	Description
<code>fileStorage.whitelist</code>	A comma-separated list of protocols that are permitted to read and write with ADLS Gen2 storage. NOTE: The protocol identifier "abfss" must be included in this list.
<code>fileStorage.defaultBaseUri</code>	For each supported protocol, this param must contain a top-level path to the location where Designer Cloud powered by 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.

3. Save your changes and restart the platform.

Configure access mode

Mode	Description
System	<p>All users authenticate to ADLS using a single system key/secret combination. This combination is specified in the following parameters, which you should have already defined:</p> <ul style="list-style-type: none">• <code>azure.applicationId</code>• <code>azure.secret</code>• <code>azure.directoryId</code> <p>These properties define the registered application in Azure Active Directory. System authentication mode uses the registered application identifier as the service principal for authentication to ADLS. All users have the same permissions in ADLS.</p> <p>For more information on these settings, see <i>Configure for Azure</i>.</p>
User	<p>In user mode, per-user access is governed by Azure AD SSO. A set of tokens is acquired during SSO login for the user and is stored in the Azure Key Vault against the user's masked identifier.</p> <p>Additional configuration is required. See below.</p>

System mode access

When access to ADLS Gen2 is requested, the platform uses the combination of Azure directory ID, Azure application ID, and Azure secret to complete access.

Steps:

Please verify the following steps to specify the ADLS access mode.

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

```
"azure.adlsgen2.mode": "system",
```

3. Save your changes.

User mode access

In user mode, a set of tokens is acquired during SSO login for the user and is stored in the Azure Key Vault against the user's masked identifier.

Prerequisites:

- Designer Cloud powered by Trifacta platform must be integrated with a Databricks 10.x cluster. For more information, see *Configure for Azure Databricks*.
- User mode access to ADLS requires Single Sign On (SSO) to be enabled for integration with Azure Active Directory. For more information, see *Configure SSO for Azure AD*.

Steps:

Please verify the following steps to specify the ADLS access mode.

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

```
"azure.adlsgen2.mode": "user",
```

3. Save your changes.

Testing

Restart services. See *Start and Stop the Platform*.

After the configuration has been specified, an ADLS Gen2 connection appears in the Import Data page. Select it to begin navigating for data sources.

NOTE: If you have multiple ADLS Gen2 file systems or storage accounts, you can access the secondary ones through the ADLS Gen2 browser. Edit the URL path in the browser and paste in the URI for other locations.

Try running a simple job from the Designer Cloud application . For more information, see *Verify Operations*.

Troubleshooting

Problem: SSLHandshakeException : Unsupported curveId: 29 error when retrieving Databricks token

This issue is caused by the Designer Cloud powered by Trifacta platform sending a known set of elliptic curve algorithms to Microsoft during SSL handshake, but an unsupported curve algorithm is being negotiated and used by the Microsoft server.

- This issue applies only when SSL is enabled when accessing the base storage layer.
- This issue applies to all Azure-based base storage layers supported by the Designer Cloud powered by Trifacta platform .

A similar issue is described here: <https://bugs.openjdk.java.net/browse/JDK-8171279>

Solution:

Microsoft should fix the problem.

Within the Designer Cloud powered by Trifacta platform , you can apply the following workaround:

NOTE: This solution disables the use of the listed algorithms for all Java services installed on the Trifacta node and is satisfactory for all Java services of the Designer Cloud powered by Trifacta platform .

1. Login to the Trifacta node as an administrator.
2. Edit the following file:

```
$JAVA_HOME/jre/lib/security/java.security
```

3. Locate the following parameter: `jdk.tls.disabledAlgorithms`.
4. To the above parameter, add the following algorithm references to disable them:

```
TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA, TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA
```

5. Save your changes and restart the platform.

Using ADLS Gen 2 Connection

Uses of ADLS

The Designer Cloud powered by Trifacta platform can use ADLS for the following reading and writing tasks:

1. **Creating Datasets from ADLS Files:** You can read in from a data source stored in ADLS. A source may be a single ADLS file or a folder of identically structured files. See *Reading from Sources in ADLS* below.
2. **Reading Datasets:** When creating a dataset, you can pull your data from another dataset defined in ADLS. See *Creating Datasets* below.
3. **Writing Job Results:** After a job has been executed, you can write the results back to ADLS. See *Writing Job Results* below.

In the Designer Cloud application, ADLS is accessed through the ADLS browser.

NOTE: When the Designer Cloud powered by Trifacta platform executes a job on a dataset, the source data is untouched. Results are written to a new location, so that no data is disturbed by the process.

Before you begin using ADLS

- **Read/Write Access:** Your cluster administrator must configure read/write permissions to locations in ADLS. Please see the ADLS documentation.

Avoid using `/trifacta/uploads` for reading and writing data. This directory is used by the Designer Cloud application.

- Your cluster administrator should provide a place or mechanism for raw data to be uploaded to your datastore.
- Your cluster administrator should provide a writeable home output directory for you, which you can review. See *Storage Config Page*.

Secure access

Depending on the security features you've enabled, the technical methods by which Trifacta users access ADLS may vary. For more information, see *ADLS Gen1 Access*.

Storing data in ADLS

Your cluster administrator should provide raw data or locations and access for storing raw data within ADLS. All Trifacta users should have a clear understanding of the folder structure within ADLS where each individual can read from and write their job results.

- Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

NOTE: The Designer Cloud powered by Trifacta platform does not modify source data in ADLS. Sources stored in ADLS are read without modification from their source locations, and sources that are uploaded to the platform are stored in `/trifacta/uploads`.

Reading from sources in ADLS

You can create a dataset from one or more files stored in ADLS..

Folder selection:

When you select a folder in ADLS to create your dataset, you select all files in the folder to be included. Notes:

- This option selects all files in all sub-folders. If your sub-folders contain separate datasets, you should be more specific in your folder selection.
- All files used in a single dataset must be of the same format and have the same structure. For example, you cannot mix and match CSV and JSON files if you are reading from a single directory.
- When a folder is selected from ADLS, the following file types are ignored:
 - *_SUCCESS and *_FAILED files, which may be present if the folder has been populated by the running environment.
 - If you have stored files in ADLS that begin with an underscore (_), these files cannot be read during batch transformation and are ignored. Please rename these files through ADLS so that they do not begin with an underscore.

Parameters:

You can parameterize parts of your paths to source files for your imported dataset. Parameters can be applied to:

- user info value
- host value
- path values

For more information:

- See *Parameterize Files for Import*.
- See *Overview of Parameterization*.

Creating datasets

When creating a dataset, you can choose to read data in from a source stored from ADLS or from a local file.

- ADLS sources are not moved or changed.
- Local file sources are uploaded to `/trifacta/uploads` where they remain and are not changed.

Data may be individual files or all of the files in a folder. For more information, see *Reading from Sources in ADLS* above.

In the Import Data page, click the ADLS tab. See *Import Data Page*.

Writing job results

When your job results are generated, they can be stored back in ADLS for you at the location defined for your user account.

- The ADLS location is available through the Publishing dialog in the Output Destinations tab of the Job Details page. See *Publishing Dialog*.
- Each set of job results must be stored in a separate folder within your ADLS output home directory.
- For more information on your output home directory, see *Storage Config Page*.

If your deployment is using ADLS, do not use the `trifacta/uploads` directory. This directory is used for storing uploads and metadata, which may be used by multiple users. Manipulating files outside of the Designer Cloud application can destroy other users' data. Please use the tools provided through the interface for managing uploads from ADLS.

Users can specify a default output home directory and, during job execution, an output directory for the current job.

Access to results:

Depending on how the platform is integrated with ADLS, other users may or may not be able to access your job results.

- If user mode is enabled, results are written to ADLS through the ADLS account configured for your use. Depending on the permissions of your ADLS account, you may be the only person who can access these results.
- If user mode is not enabled, then each Trifacta user writes results to ADLS using a shared account. Depending on the permissions of that account, your results may be visible to all platform users.

Creating a new dataset from results

As part of writing job results, you can choose to create a new dataset, so that you can chain together data wrangling tasks.

NOTE: When you create a new dataset as part of your job results, the file or files are written to the designated output location for your user account. Depending on how your cluster permissions are configured, this location may not be accessible to other users.

Reference

Supported Versions: n/a

Supported Environments:

Operation	Designer Cloud Powered by Trifacta Enterprise Edition	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if ABFSS is base storage layer)

Create New Connection: n/a

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

ADLS Gen1 Access

Contents:

- *Limitations*
 - *Read-only access*
- *Prerequisites*
 - *General*
 - *Create a registered application*
 - *Azure properties*
 - *Key Vault Setup*
- *Configure ADLS Authentication*
 - *System mode access*
 - *User mode access*
- *Configure the Designer Cloud powered by Trifacta platform*
 - *Configure storage protocol*
 - *Define storage locations*
- *Testing*
- *Troubleshooting*
- *Using ADLS Gen 1 Connection*
 - *Uses of ADLS*
 - *Before you begin using ADLS*
 - *Secure access*
 - *Storing data in ADLS*
 - *Reading from sources in ADLS*
- *Creating Datasets*
 - *Writing job results*
 - *Creating a new dataset from results*
- *Reference*

By default, Microsoft Azure deployments integrate with Azure Data Lake Store (ADLS). Optionally, you can configure your deployment to integrate with WASB.

- **Microsoft Azure Data Lake Store (ADLS Gen1)** is a scalable repository for big data analytics.
- ADLS Gen1 is accessible from Azure Databricks.
- For more information, see <https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-overview>.
- For more information on the newer version of ADLS, see [ADLS Gen2 Access](#).

Supported Environments:

Operation	Designer Cloud Powered by Trifacta Enterprise Edition	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if ADLS Gen1 is base storage layer)

Limitations

- A single public connection to ADLS Gen1 is supported.
- In this release, the Designer Cloud powered by Trifacta platform supports integration with the default store only. Extra stores are not supported.

Read-only access

If the base storage layer has been set to WASB, you can follow these instructions to set up read-only access to ADLS Gen1.

NOTE: To enable read-only access to ADLS Gen1, do not set the base storage layer to `adl`. The base storage layer for ADLS read-write access must remain `wasbs`.

Prerequisites

General

- The Designer Cloud powered by Trifacta platform has already been installed and integrated with an Azure Databricks cluster. See *Configure for Azure Databricks*.
- ADL must be set as the base storage layer for the Designer Cloud powered by Trifacta platform instance. See *Set Base Storage Layer*.

Create a registered application

Before you integrate with Azure ADLS Gen1, you must create the Designer Cloud powered by Trifacta platform as a registered application. See *Configure for Azure*.

Azure properties

The following properties should already be specified in the Admin Settings page. Please verify that the following have been set:

- `azure.applicationId`
- `azure.secret`
- `azure.directoryId`

The above properties are needed for this configuration. For more information, see *Configure for Azure*.

Key Vault Setup

An Azure Key Vault has already been set up and configured for use by the Designer Cloud powered by Trifacta platform. For more information, see *Configure for Azure*.

Configure ADLS Authentication

Authentication to ADLS storage is supported for the following modes, which are described in the following section.

Mode	Description
System	<p>All users authenticate to ADLS using a single system key/secret combination. This combination is specified in the following parameters, which you should have already defined:</p> <ul style="list-style-type: none">• <code>azure.applicationId</code>• <code>azure.secret</code>• <code>azure.directoryId</code> <p>These properties define the registered application in Azure Active Directory. System authentication mode uses the registered application identifier as the service principal for authentication to ADLS Gen1. All users have the same permissions in ADLS Gen1.</p> <p>For more information on these settings, see <i>Configure for Azure</i>.</p>
User	<p>Per-user mode allows individual users to authenticate to ADLS Gen1 through their Azure Active Directory login.</p>

NOTE: Additional configuration for AD SSO is required. Details are below.

Steps:

Please complete the following steps to specify the ADLS Gen1 access mode.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Set the following parameter to the preferred mode (`system` or `user`):

```
"azure.adl.mode": "<your_preferred_mode>",
```

3. Save your changes.

System mode access

When access to ADLS is requested, the platform uses the combination of Azure directory ID, Azure application ID, and Azure secret to complete access.

After defining the properties in the Designer Cloud powered by Trifacta platform, system mode access requires no additional configuration.

User mode access

In user mode, a user ID hash is generated from the Key Vault key/secret and the user's AD login. This hash is used to generate the access token, which is stored in the Key Vault.

Set up for Azure AD SSO

NOTE: User mode access to ADLS requires Single Sign On (SSO) to be enabled for integration with Azure Active Directory. For more information, see *Configure SSO for Azure AD*.

Configure the Designer Cloud powered by Trifacta platform

Configure storage protocol

You must configure the platform to use the ADL storage protocol when accessing.

NOTE: Per earlier configuration, base storage layer must be set to `adl` for read/write access to ADLS. See *Set Base Storage Layer*.

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 change its value to `adl`:

```
"webapp.storageProtocol": "adl",
```

3. Set the following parameter to `false`:

```
"hdfs.enabled": false,
```

4. Save your changes and restart the platform.

Define storage locations

You must define the base storage location and supported protocol for storing data on ADLS.

NOTE: You can specify only one storage location for ADLS.

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 configuration block. Specify the listed changes:

```
"fileStorage": {
  "defaultBaseUri": [
    "<baseURIOfYourLocation>"
  ],
  "whitelist": ["adl"]
}
```

Parameter	Description
defaultBaseUri	<p>A comma-separated list of protocols that are permitted to read and write with ADLS storage.</p> <p>NOTE: The <code>adl://</code> protocol identifier must be included.</p> <p>Example value:</p> <pre>adl://<YOUR_STORE_NAME>.azuredatalakestore.net</pre>
whitelist	<p>For each supported protocol, this array must contain a top-level path to the location where Designer Cloud powered by Trifacta platform files can be stored. These files include uploads, samples, and temporary storage used during job execution.</p> <p>NOTE: This array of values must include <code>adl</code>.</p>

3. Save your changes and restart the platform.

Testing

Restart services. See *Start and Stop the Platform*.

After the configuration has been specified, an ADLS connection appears in the Import Data page. Select it to begin navigating for data sources.

Specify ADLS Gen1 Path:

In the ADLS Gen1 browser, you can specify an explicit path to resources. Click the Pencil icon, paste the path value, and click **Go**.

```
/trifacta/input/username@example.com
```

You should paste the following in the Path textbox:

```
hdfs://trifacta/input/username@example.com
```

NOTE: When inserting values directly into the Path textbox, you must use the `hdfs://` protocol identifier. Do not use the `adl://` protocol identifier.

Tip: You can retrieve your home directory from your profile. See *Storage Config Page*.

Try running a simple job from the Designer Cloud application . For more information, see *Verify Operations*.

Troubleshooting

For additional troubleshooting information, see *ADLS Gen2 Access*.

Using ADLS Gen 1 Connection

Uses of ADLS

The Designer Cloud powered by Trifacta platform can use ADLS for the following reading and writing tasks:

1. **Creating Datasets from ADLS Files:** You can read in from a data source stored in ADLS. A source may be a single ADLS file or a folder of identically structured files. See *Reading from Sources in ADLS* below.
2. **Reading Datasets:** When creating a dataset, you can pull your data from another dataset defined in ADLS. See *Creating Datasets* below.
3. **Writing Job Results:** After a job has been executed, you can write the results back to ADLS. See *Writing Job Results* below.

In the Designer Cloud application , ADLS is accessed through the ADLS browser.

NOTE: When the Designer Cloud powered by Trifacta platform executes a job on a dataset, the source data is untouched. Results are written to a new location, so that no data is disturbed by the process.

Before you begin using ADLS

- **Read/Write Access:** Your cluster administrator must configure read/write permissions to locations in ADLS. Please see the ADLS documentation.

Avoid using `/trifacta/uploads` for reading and writing data. This directory is used by the Designer Cloud application .

- Your cluster administrator should provide a place or mechanism for raw data to be uploaded to your datastore.
- Your cluster administrator should provide a writeable home output directory for you, which you can review. See *Storage Config Page*.

Secure access

Depending on the security features you've enabled, the technical methods by which Trifacta users access ADLS may vary. For more information, see *ADLS Gen1 Access*.

Storing data in ADLS

Your cluster administrator should provide raw data or locations and access for storing raw data within ADLS. All Trifacta users should have a clear understanding of the folder structure within ADLS where each individual can read from and write their job results.

- Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.

NOTE: The Designer Cloud powered by Trifacta platform does not modify source data in ADLS. Sources stored in ADLS are read without modification from their source locations, and sources that are uploaded to the platform are stored in `/trifacta/uploads`.

Reading from sources in ADLS

You can create a dataset from one or more files stored in ADLS.

Wildcards:

You can parameterize your input paths to import source files as part of the same imported dataset. For more information, see *Overview of Parameterization*.

Folder selection:

When you select a folder in ADLS to create your dataset, you select all files in the folder to be included. Notes:

- This option selects all files in all sub-folders. If your sub-folders contain separate datasets, you should be more specific in your folder selection.
- All files used in a single dataset must be of the same format and have the same structure. For example, you cannot mix and match CSV and JSON files if you are reading from a single directory.
- When a folder is selected from ADLS, the following file types are ignored:
 - `*_SUCCESS` and `*_FAILED` files, which may be present if the folder has been populated by the running environment.
 - If you have stored files in ADLS that begin with an underscore (`_`), these files cannot be read during batch transformation and are ignored. Please rename these files through ADLS so that they do not begin with an underscore.

Creating Datasets

When creating a dataset, you can choose to read data in from a source stored from ADLS or from a local file.

- ADLS sources are not moved or changed.
- Local file sources are uploaded to `/trifacta/uploads` where they remain and are not changed.

Data may be individual files or all of the files in a folder. For more information, see *Reading from Sources in ADLS* above.

In the Import Data page, click the ADLS tab. See *Import Data Page*.

Writing job results

When your job results are generated, they can be stored back in ADLS for you at the location defined for your user account.

- The ADLS location is available through the Publishing dialog in the Output Destinations tab of the Job Details page. See *Publishing Dialog*.
- Each set of job results must be stored in a separate folder within your ADLS output home directory.
- For more information on your output home directory, see *Storage Config Page*.

If your deployment is using ADLS, do not use the `trifacta/uploads` directory. This directory is used for storing uploads and metadata, which may be used by multiple users. Manipulating files outside of the Designer Cloud application can destroy other users' data. Please use the tools provided through the interface for managing uploads from ADLS.

Users can specify a default output home directory and, during job execution, an output directory for the current job.

Access to results:

Depending on how the platform is integrated with ADLS, other users may or may not be able to access your job results.

- If user mode is enabled, results are written to ADLS through the ADLS account configured for your use. Depending on the permissions of your ADLS account, you may be the only person who can access these results.
- If user mode is not enabled, then each Trifacta user writes results to ADLS using a shared account. Depending on the permissions of that account, your results may be visible to all platform users.

Creating a new dataset from results

As part of writing job results, you can choose to create a new dataset, so that you can chain together data wrangling tasks.

NOTE: When you create a new dataset as part of your job results, the file or files are written to the designated output location for your user account. Depending on how your cluster permissions are configured, this location may not be accessible to other users.

Reference

Supported Versions: n/a

Supported Environments:

Operation	Designer Cloud Powered by Trifacta Enterprise Edition	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if ADLS Gen1 is base storage layer)

Create New Connection: n/a

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

WASB Access

Contents:

- *Limitations*
 - *Read-only access*
- *Prerequisites*
 - *General*
 - *Create a registered application*
 - *Other Azure properties*
 - *Key Vault Setup*
- *Configure WASB Authentication*
- *Configure the Designer Cloud powered by Trifacta platform*
 - *Define location of SAS token*
 - *Define WASB stores*
 - *Configure storage protocol*
 - *Define storage locations*
- *Testing*
- *Troubleshooting*
- *Using WASB Connection*
 - *Uses of WASB*
 - *Before you begin using WASB*
 - *Secure access*
 - *Storing data in WASB*
 - *Reading from sources in WASB*
 - *Creating datasets*
 - *Writing job results*
 - *Creating a new dataset from results*
- *Reference*

By default, Microsoft Azure deployments integrate with Azure Data Lake Store (ADLS). Optionally, you can configure your deployment to integrate with WASB.

- **Windows Azure Storage Blob (WASB)** is an abstraction layer on top of HDFS, which enables persistence of storage, access without a Hadoop cluster presence, and access from multiple Hadoop clusters.

Supported Environments:

Operation	Designer Cloud Powered by Trifacta Enterprise Edition	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if WASB is base storage layer)

Limitations

- A single public connection to WASB is supported.
- If a directory is created on the cluster through WASB, the directory includes a Size=0 blob. The Designer Cloud powered by Trifacta platform does not list them and does not support interaction with Size=0 blobs.

Read-only access

If the base storage layer has been set to ADLS Gen1 or ADLS Gen2, you can follow these instructions to set up read-only access to WASB.

NOTE: If you are adding WASB as a secondary integration, your WASB blob container or containers must contain at least one folder. This is a known issue.

NOTE: To enable read-only access to WASB, do not set the base storage layer to `wasbs`. The base storage layer must remain set for ADLS Gen 1 or ADLS Gen 2.

Prerequisites

General

- The Designer Cloud powered by Trifacta platform has already been installed and integrated with an Azure Databricks cluster.
- WASB must be set as the base storage layer for the Designer Cloud powered by Trifacta platform instance. See *Set Base Storage Layer*.
- For each combination of blob host and container, a separate Azure Key Vault Store entry must be created. For more information, please contact your Azure admin.

Create a registered application

Before you integrate with Azure WASB, you must create the Designer Cloud powered by Trifacta platform as a registered application. See *Configure for Azure*.

Other Azure properties

The following properties should already be specified in the Admin Settings page. Please verify that the following have been set:

- `azure.applicationId`
- `azure.secret`
- `azure.directoryId`

The above properties are needed for this configuration. For more information, see *Configure for Azure*.

Key Vault Setup

For new installs, an Azure Key Vault has already been set up and configured for use by the Designer Cloud powered by Trifacta platform.

NOTE: An Azure Key Vault is required. Upgrading customers who do not have a Key Vault in their environment must create one.

For more information, see *Configure for Azure*.

Configure WASB Authentication

Authentication to WASB storage is managed by specifying the appropriate host, container, and token ID in the Designer Cloud powered by Trifacta platform configuration. When access to WASB is requested, the platform passes the information through the Secure Token Service to query the specified Azure Key Vault Store using the provided values. The keystore returns the value for the secret. The combination of the key (token ID) and secret is used to access WASB.

NOTE: Per-user authentication is not supported for WASB.

For more information on creating the Key Vault Store and accessing it through the Secure Token Service, see *Configure for Azure*.

Configure the Designer Cloud powered by Trifacta platform

Enable

For more information, see *WASB Access*.

Define location of SAS token

The SAS token required for accessing Azure can be accessed from either of the following locations:

1. Key Vault
2. Trifacta configuration

SAS token from Key Vault

To store the SAS token in the key vault, specify the following parameters in platform configuration. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.

Parameter	Description
<code>"azure.wasb.fetchSasTokensFromKeyVault": true,</code>	Instructs the Designer Cloud powered by Trifacta platform to query the Key Vault for SAS tokens NOTE: The Key Vault must already be set up. See "Key Vault Setup" above.

SAS token from Trifacta configuration

To specify the SAS token in the Designer Cloud powered by Trifacta platform configuration, set the following flag to `false`:

Parameter	Description
<code>"azure.wasb.fetchSasTokensFromKeyVault": false,</code>	Instructs the Designer Cloud powered by Trifacta platform to acquire per-container SAS tokens from the platform configuration.

Define WASB stores

The WASB stores that users can access are specified as an array of configuration values. Users of the platform can use all of them for reading sources and writing results.

Steps:

1. To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the `azure.wasb.stores` configuration block.
3. Apply the appropriate configuration as specified below.

Tip: The default container must be specified as the first set of elements in the array. All containers listed after the first one are treated as extra stores.

```

"azure.wasb.stores":
[
  {
    "sasToken": "<DEFAULT_VALUE1_HERE>",
    "keyVaultSasTokenSecretName": "<DEFAULT_VALUE1_HERE>",
    "container": "<DEFAULT_VALUE1_HERE>",
    "blobHost": "<DEFAULT_VALUE1_HERE>"
  },
  {
    "sasToken": "<VALUE2_HERE>",
    "keyVaultSasTokenSecretName": "<VALUE2_HERE>",
    "container": "<VALUE2_HERE>",
    "blobHost": "<VALUE2_HERE>"
  }
]
},

```

Parameter	Description	SAS Token from Azure Key Vault	SAS Token from Platform Configuration
sasToken	<p>Set this value to the SAS token to use, if applicable.</p> <p>Example value:</p> <pre>?sv=2019-02-02&ss=bfqt&srt=sco&sp=rwdlacup&se=2022-02-13T00:00:00Z&st=2020-02-13T00:00:00Z&spr=https&sig=<redacted></pre>	<p>Set this value to an empty string.</p> <p>NOTE: Do not delete the entire line. Leave the value as empty.</p>	<p>See below for the command to execute to generate a SAS token.</p>
keyVaultSasTokenSecretName	<p>Set this value to the secret name of the SAS token in the Azure key vault to use for the specified blob host and container.</p> <p>If needed, you can generate and apply a per-container SAS token for use in this field for this specific store. Details are below.</p>		<p>Set this value to an empty string.</p> <p>NOTE: Do not delete the entire line. Leave the value as empty.</p>
container	<p>Apply the name of the WASB container.</p> <p>NOTE: If you are specifying different blob host and container combinations for your extra stores, you must create a new Key Vault store. See above for details.</p>		
blobHost	<p>Specify the blob host of the container.</p> <p>Example value:</p> <pre>storage-account.blob.core.windows.net</pre> <p>NOTE: If you are specifying different blob host and container combinations for your extra stores, you must create a new Key Vault store. See above for details.</p>		

4. Save your changes and restart the platform.

Generate per-container SAS token

Execute the appropriate command at the command line to generate a SAS token for a specific container. The following Windows PowerShell command generates a SAS token that is valid for a full year:

```
Set-AzureRmStorageAccount -Name 'name'  
$sasToken = New-AzureStorageContainerSASToken -Permission r -ExpiryTime (Get-Date).AddYears(1) -Name  
'<container_name>'
```

Tip: You can also generate a Shared Access Signature token for your Storage Account and Container from the Azure Portal.

Configure storage protocol

You must configure the platform to use the WASBS (secure) storage protocol when accessing.

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 change its value `wasbs` for secure access:

```
"webapp.storageProtocol": "wasbs",
```

3. Set the following:

```
"hdfs.enabled": false,
```

4. Save your changes and restart the platform.

Define storage locations

You must define the base blob locations and supported protocol for storing data on WASB.

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 configuration block. Specify the listed changes:

```
"fileStorage": {  
  "defaultBaseUri": [  
    "<baseURIOfYourBlob>"  
  ],  
  "whitelist": ["wasbs"]  
}
```

Parameter	Description
defaultBaseUri	A comma-separated list of protocols that are permitted to read and write with WASB storage. NOTE: The <code>wasbs : //</code> protocol identifier must be included. WASB protocol is not supported.

	<p>Example value:</p> <pre>wasbs://container@storage-account.blob.core.windows.net/</pre>
whitelist	<p>For each supported protocol, this array must contain a top-level path to the location where Designer Cloud powered by Trifacta platform files can be stored. These files include uploads, samples, and temporary storage used during job execution.</p> <p>NOTE: This array of values must include wasbs.</p>

3. Save your changes and restart the platform.

Testing

After the configuration has been specified, a WASB connection appears in the Import Data page. Select it to begin navigating through the WASB Browser for data sources.

Try running a simple job from the Designer Cloud application . For more information, see *Verify Operations*.

Troubleshooting

For additional troubleshooting information, see *ADLS Gen2 Access*.

Using WASB Connection

Uses of WASB

The Designer Cloud powered by Trifacta platform can use WASB for the following tasks for reading and writing data:

1. **Importing datasets from WASB Files:** You can read in from source data stored in WASB. An imported dataset may be a single WASB file or a folder of identically structured files. See *Reading from Sources in WASB* below.
2. **Reading Datasets:** When creating a dataset, you can pull your data from a source in WASB. See *Creating Datasets* below.
3. **Writing Results:** You can publish your results directly to WASB. See *Writing Job Results* below.
4. **Publishing Job Results:** After a job has been executed, you can write the results back to WASB. See *Writing Job Results* below.

In the Designer Cloud application , WASB is accessed through the WASB browser. See *File System Browser*.

NOTE: When the Designer Cloud powered by Trifacta platform executes a job on a dataset, the source data is untouched. Results are written to a new location, so that no data is disturbed by the process.

Before you begin using WASB

- **Read/Write Access:** Your administrator must configure read/write permissions to locations in WASB. Please see the WASB documentation provided with your cluster software distribution.

Avoid using `/trifacta/uploads` for reading and writing data. This directory is used by the Designer Cloud application .

NOTE: If a directory is created on the cluster through WASB, the directory includes a Size=0 blob. The Designer Cloud powered by Trifacta platform does not list them and does not support interaction with Size=0 blobs.

- Your cluster administrator should provide a place or mechanism for raw data to be uploaded to your datastore.
- Your cluster administrator should provide a writeable home output directory for you, which you can review. See *Storage Config Page*.

Secure access

Client-side encryption is not supported. Through WASBS, HTTPS is supported.

Storing data in WASB

Your cluster administrator should provide raw data or locations and access for storing raw data within WASB. All Trifacta users should have a clear understanding of the folder structure within WASB where each individual can read from and write their job results.

- Users should know where shared data is located and where personal data can be saved without interfering with or confusing other users.
- The Designer Cloud application stores the results of each job in a separate folder in WASB.

NOTE: The Designer Cloud powered by Trifacta platform does not modify source data in WASB. Data stored in WASB is read without modification from source locations, and source data that is uploaded to the platform are stored in `/trifacta/uploads`.

Reading from sources in WASB

You can import a dataset from one or more files stored in WASB.

Wildcards:

You can parameterize your input paths to import source files as part of the same imported dataset. For more information, see *Overview of Parameterization*.

Folder selection:

When you select a folder in WASB for your imported dataset, you select all files in the folder to be included. Notes:

- This option selects all files in all sub-folders. If your sub-folders contain separate datasets, you should be more specific in your folder selection.
- All files used in a single imported dataset must be of the same format and have the same structure. For example, you cannot mix and match CSV and JSON files if you are reading from a single directory. Files of the same format must have identical column structures.
- When a folder is selected from WASB, the following file types are ignored:
 - `*_SUCCESS` and `*_FAILED` files, which may be present if the folder has been populated from the cluster.
 - If you have stored files in WASB that begin with an underscore (`_`), these files cannot be read during batch transformation and are ignored. Please rename these files through WASB so that they do not begin with an underscore.

Creating datasets

When creating a dataset, you can choose to read data in source data stored from WASB or from a local file.

- WASB sources are not moved or changed.
- Local file sources are uploaded to `/trifacta/uploads` where they remain and are not changed.

Data may be individual files or all of the files in a folder.

- For more information, see *Reading from Sources in WASB*.
- In the Import Data page, click the WASB tab. See *Import Data Page*.

Writing job results

When your job results are generated, they can be stored back in WASB at the location defined for your user account.

- As part of the job specification, you can create a publishing target in WASB. See *Run Job Page*.
- For ad-hoc publication to WASB, the target location is available through the Job Details page. See *Job Details Page*.
- Each set of job results must be stored in a separate folder within your WASB output home directory.
- For more information on your output home directory, see *Storage Config Page*.

If your deployment is using WASB, do not use the `trifacta/uploads` directory. This directory is used for storing uploads and metadata, which may be used by multiple users. Manipulating files outside of the Designer Cloud application can destroy other users' data. Please use the tools provided through the interface for managing uploads from WASB.

Creating a new dataset from results

As part of writing job results, you can choose to create a new dataset, so that you can chain together data wrangling tasks.

NOTE: When you create a new dataset as part of your job results, the file or files are written to the designated output location for your user account. Depending on how your WASB permissions are configured, this location might not be accessible to other users.

Reference

Supported Versions: n/a

Supported Environments:

Operation	Designer Cloud Powered by Trifacta Enterprise Edition	Amazon	Microsoft Azure
Read	Not supported	Not supported	Supported
Write	Not supported	Not supported	Supported (only if WASB is base storage layer)

Create New Connection: n/a

NOTE: A single public connection to WASB is supported.

Configure for Azure Databricks

Contents:

- *Prerequisites*
 - *Simba driver*
- *Limitations*
 - *Supported versions of Azure Databricks*
 - *Job limits*
- *Create Cluster*
- *Enable*
- *Configure*
 - *Configure cluster mode*
 - *Configure use of cluster policies*
 - *Configure Platform*
 - *Configure instance pooling*
 - *Configure Databricks job management*
 - *Configure Databricks job throttling*
 - *Configure for Databricks Secrets Management*
 - *Configure personal access token*
 - *Combine transform and profiling for Spark jobs*
- *Additional Configuration*
 - *Enable SSO for Azure Databricks*
 - *Enable Azure Managed Identity access*
 - *Enable shared cluster for Databricks Tables*
 - *Specify Databricks Tables cluster name*
 - *Pass additional Spark properties*
 - *Configure Maximum Retries for REST API*
- *Use*
 - *Run job from application*
 - *Run job via API*
- *Troubleshooting*
 - *Spark job on Azure Databricks fails with "Invalid spark version" error*
 - *Spark job fails with "spark scheduler cannot be cast" error*

This section describes how to configure the Designer Cloud powered by Trifacta® platform to integrate with Databricks hosted in Azure.

- Azure Databricks is an Apache Spark implementation that has been optimized for use on the Azure platform. For more information, see <https://databricks.com/product/azure>.

NOTE: You cannot integrate with existing Azure Databricks clusters.

Additional Databricks features supported by the platform:

- Table access control: <https://docs.databricks.com/security/access-control/table-acls/object-privileges.html>

Prerequisites

- The Designer Cloud powered by Trifacta platform must be deployed in Microsoft Azure.

NOTE: If you are using Azure Databricks as a datasource, please verify that openJDKv1.8.0_302 or earlier is installed on the Trifacta node. Java 8 is required. If necessary, downgrade the Java version and restart the platform. There is a known issue with TLS v1.3.

Simba driver

Beginning in Release 7.1, the integration with Azure Databricks switched from using a Hive-based driver to a Simba driver for the integration with Spark.

NOTE: If you have upgraded from a release before Release 7.1, you should review the Connect String Options in your Azure Databricks connections, such as Databricks Tables. These options may not work with the Simba driver.

No installation is required to use this new driver.

For more information on the Simba driver, see <https://databricks.com/spark/odbc-driver-download>.

Limitations

NOTE: If you are using Azure AD to integrate with an Azure Databricks cluster, the Azure AD secret value stored in `azure.secret` must begin with an alphanumeric character. This is a known issue.

- The Designer Cloud powered by Trifacta platform must be installed on Microsoft Azure.
- Import from nested folders is not supported for running jobs from Azure Databricks.
- When a job is started and no cluster is available, a cluster is initiated, which can take up to four minutes. If the job is canceled during cluster startup:
 - The job is terminated, and the cluster remains.
 - The job is reported in the application as Failed, instead of Canceled.
- Azure Databricks integration works with Spark 2.4.x, Spark 3.0.1, Spark 3.2.0, and Spark 3.2.1.

NOTE: The version of Spark for Azure Databricks must be applied to the platform configuration through the `databricks.sparkVersion` property. Details are provided later.

- Azure Databricks integration does not work with Hive.

Supported versions of Azure Databricks

Supported versions:

- Azure Databricks 10.x
- Azure Databricks 9.1 LTS (**Recommended**)
- Azure Databricks 7.3 LTS

NOTE: When an Azure Databricks version is deprecated, it is no longer available for selection when creating a new cluster. As a result, if a new cluster needs to be created for whatever reason, it must be created using a version supported by Azure Databricks, which may require you to change the value of the Spark version settings in the Designer Cloud powered by Trifacta platform. See "Troubleshooting" for more information.

For more information on supported versions, see <https://docs.databricks.com/release-notes/runtime/releases.html#supported-databricks-runtime-releases-and-support-schedule>

Job limits

By default, the number of jobs permitted on an Azure Databricks cluster is set to 1000.

- The number of jobs that can be created per workspace in an hour is limited to 1000 .
 - If Databricks Job Management is enabled in the platform, then this limit is raised to 5000 by using the run-submit API. For more information, see "Configure Databricks job management" below.
 - For more information, see <https://docs.databricks.com/dev-tools/api/latest/jobs.html#run-now>.
- These limits apply to any jobs run for workspace data on the cluster.
- The number of actively concurrent job runs in a workspace is limited to 150.

Managing limits:

To enable retrieval and auditing of job information after a job has been completed, the Designer Cloud powered by Trifacta platform does not delete jobs from the cluster. As a result, jobs can accumulate over time to exceeded the number of jobs permitted on the cluster. If you reach these limits, you may receive a `Quota for number of jobs has been reached limit`. For more information, see <https://docs.databricks.com/user-guide/jobs.html>.

Optionally, you can allow the Designer Cloud powered by Trifacta platform to manage your jobs to avoid these limitations. For more information, see "Configure Databricks job management" below.

Create Cluster

NOTE: Integration with pre-existing Azure Databricks clusters is not supported.

When a user first requests access to Azure Databricks, a new Azure Databricks cluster is created for the user. Access can include a request to run a job or to browse Databricks Tables. Cluster creation may take a few minutes.

A new cluster is also created when a user launches a job after:

- The Azure Databricks configuration properties or Spark properties are changed in platform configuration.
- A JAR file is updated on the Trifacta node

A user's cluster automatically terminates after a configurable time period. A new cluster is automatically created when the user next requests access to Azure Databricks access. See "Configure Platform" below.

Enable

To enable Azure Databricks, please perform the following configuration changes.

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter, which enables Trifacta Photon for smaller job execution. Set it to `Enabled`:

```
Photon execution
```

3. You do not need to save to enable the above configuration change.
4. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
5. Locate the following parameters. Set them to the values listed below, which enables Azure Databricks (small to extra-large jobs) running environments:

```
"webapp.runInDatabricks": true,  
"webapp.runWithSparkSubmit": false,
```

```
"webapp.runinEMR": false,
"webapp.runInDataflow": false,
```

6. Locate the following parameter and set it to `azure`:

```
"metadata.cloud": "azure",
```

NOTE: Do not set this parameter to any value other than `azure` while using Azure Databricks.

7. Do not save your changes until you have completed the following configuration section.

Configure

Configure cluster mode

When a user submits a job, the Designer Cloud Powered by Trifacta Enterprise Edition provides all the cluster specifications in the Databricks API and it creates cluster only for per-user or per-job. that means once the job is complete, the cluster is terminated. Cluster creation may take less than 30 seconds if instance pools are used. If the instance pools are not used, it may take 10-15 minutes.

For more information on job clusters, see <https://docs.databricks.com/clusters/configure.html> .

The job clusters automatically terminate after the job is completed. A new cluster is automatically created when the user next requests access to Azure Databricks access.

Cluster Mode	Description
USER	When a user submits a job, Designer Cloud Powered by Trifacta Enterprise Edition creates a new cluster and persists the cluster ID in Designer Cloud Powered by Trifacta Enterprise Edition metadata for the user if the cluster does not exist or invalid. If the user already has an existing interactive valid cluster, then the existing cluster is reused when submitting the job. Default cluster mode to run jobs in Azure Databricks.
JOB	When a user submits a job, Designer Cloud Powered by Trifacta Enterprise Edition provides all the cluster specifications in the Databricks API. Databricks creates a cluster only for this job and terminates it as soon as the job completes.

Configure use of cluster policies

Optionally, you can configure the Designer Cloud powered by Trifacta platform to use the Databricks cluster policies that have been specified by your Databricks administrator for creating and using clusters. These policies are effectively templates for creation and use of Databricks clusters and govern aspects of clusters such as the type and count of nodes the resources that can be accessed via the cluster, and other settings. For more information on Databricks cluster policies, see <https://docs.databricks.com/administration-guide/clusters/policies.html>.

Prerequisites

NOTE: Your Databricks administrator must create and deploy the Databricks cluster policies from which Trifacta users can select for their personal use.

Notes:

- When this feature is enabled, each user may select the appropriate Databricks cluster policy to use for jobs. If none is selected by a user, jobs are launched without a cluster policy for the user using the Databricks properties set in platform configuration.

NOTE: Except for Spark version and cluster policy identifier in job-level overrides, other Databricks cluster configuration in the Designer Cloud powered by Trifacta platform is ignored when this feature is in use. Other job-level overrides are also ignored.

- If a cluster policy is modified and existing clusters are using it, then subsequent job executions using that policy attempt to use the same cluster. This can cause issues in performance and even job failures.

Tip: Avoid editing cluster policies that are in use, as these changed policies may be applied to clusters generated under the old policies. Instead, you should create a new policy and assign it for use.

- If the cluster policy references a Databricks instance pool that does not exist, the job fails.

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following parameter and set it to `Enabled`:

```
Databricks Cluster Policies
```

3. Save your changes and restart the platform.

NOTE: Each user must select a cluster policy to use. For more information, see *Databricks Settings Page*.

Job overrides:

A user's cluster policy can be overridden when a job is executed via API. Set the request attribute for the `clusterPolicyId`.

NOTE: If a Databricks cluster policy is used, all job-level overrides except for `clusterPolicyId` are ignored.

For more information, see *API Task - Run Job*.

Policy template for Azure - without instance pools:

The following example cluster policy can provide a basis for creating your own Azure cluster policies when instance pools are not in use:

```
{
  "autoscale.max_workers": {
    "type": "fixed",
    "value": 3,
    "hidden": true
  },
  "autoscale.min_workers": {
    "type": "fixed",
    "value": 1,
    "hidden": true
  },
  "autotermination_minutes": {
    "type": "fixed",
    "value": 10,
    "hidden": true
  },
}
```

```

"driver_node_type_id": {
  "type": "fixed",
  "value": "Standard_D3_v2",
  "hidden": true
},
"enable_local_disk_encryption": {
  "type": "fixed",
  "value": false
},
"node_type_id": {
  "type": "fixed",
  "value": "Standard_D3_v2",
  "hidden": true
}
}

```

Policy template for Azure - with instance pools:

The following example cluster policy can provide a basis for creating your own Azure cluster policies when instance pools are in use:

```

{
  "autoscale.max_workers": {
    "type": "fixed",
    "value": 3,
    "hidden": true
  },
  "autoscale.min_workers": {
    "type": "fixed",
    "value": 1,
    "hidden": true
  },
  "enable_local_disk_encryption": {
    "type": "fixed",
    "value": false
  },
  "instance_pool_id": {
    "type": "fixed",
    "value": "SOME_POOL",
    "hidden": true
  },
  "driver_instance_pool_id": {
    "type": "fixed",
    "value": "SOME_POOL",
    "hidden": true
  },
  "autotermination_minutes": {
    "type": "fixed",
    "value": 10,
    "hidden": true
  },
}

```

Configure Platform

Please review and modify the following configuration settings.

NOTE: When you have finished modifying these settings, save them and restart the platform to apply.

Parameter	Description	Value
databricks.clusterMode	Determines the cluster mode for running a Databricks job.	Default: USER

feature. parameterization. matchLimitOnSampling. databricksSpark	Maximum number of parameterized source files that are permitted for matching in a single dataset with parameters.	
databricks. workerNodeType	Type of node to use for the Azure Databricks Workers/Executors. There are 1 or more Worker nodes per cluster.	<p>Default: <code>Standard_D3_v2</code></p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p>NOTE: This property is unused when instance pooling is enabled. For more information, see Configure instance pooling below .</p> </div> <p>For more information, see the sizing guide for Azure Databricks.</p>
databricks. sparkVersion	Azure Databricks cluster version which also includes the version of Spark.	<p>Depending on your version of Azure Databricks, please set this property according to the following:</p> <ul style="list-style-type: none"> Azure Databricks 10.x: <code>10.0.x-scala2.12</code> Azure Databricks 9.1 LTS: <code>9.1.x-scala2.12</code> Azure Databricks 7.3 LTS: <code>7.3.x-scala2.12</code> <p>Please do not use other values.</p>
databricks. serviceUrl	URL to the Azure Databricks Service where Spark jobs will be run (Example: <i>https://westus2.azuredatabricks.net</i>)	
databricks. minWorkers	Initial number of Worker nodes in the cluster, and also the minimum number of Worker nodes that the cluster can scale down to during auto-scale-down	<p>Minimum value: 1</p> <p>Increasing this value can increase compute costs.</p>
databricks. maxWorkers	Maximum number of Worker nodes the cluster can create during auto scaling	<p>Minimum value: Not less than <code>databricks.minWorkers</code>.</p> <p>Increasing this value can increase compute costs.</p>
databricks.poolId	If you have enabled instance pooling in Azure Databricks, you can specify the worker node pool identifier here. For more information, see Configure instance pooling below.	<div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p>NOTE: If both <code>poolId</code> and <code>poolName</code> are specified, <code>poolId</code> is used first. If that fails to find a matching identifier, then the <code>poolName</code> value is checked.</p> </div>
databricks. poolName	If you have enabled instance pooling in Azure Databricks, you can specify the worker node pool name here. For more information, see Configure instance pooling below.	<p>See previous.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0; background-color: #e6ffe6;"> <p>Tip: If you specify a <code>poolName</code> value only, then you can use the instance pools with the same <code>poolName</code> available across multiple Databricks workspaces when you create a new cluster.</p> </div>
databricks. driverNodeType	Type of node to use for the Azure Databricks Driver. There is only 1 Driver node per cluster.	<p>Default: <code>Standard_D3_v2</code></p> <p>For more information, see the sizing guide for Databricks.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p>NOTE: This property is unused when instance pooling is enabled. For more information, see Configure instance pooling below .</p> </div>
databricks. driverPoolId	If you have enabled instance pooling in Azure Databricks, you can specify the driver node pool identifier here. For more information, see Configure instance pooling below.	<div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p>NOTE: If both <code>driverPoolId</code> and <code>driverPoolName</code> are specified, <code>driverPoolId</code> is used first. If that fails to find a matching identifier, then the <code>driverPoolName</code> value is checked.</p> </div>
databricks. driverPoolName	If you have enabled instance pooling in Azure Databricks, you can specify the driver node pool	<p>See previous.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0; background-color: #e6ffe6;"> </div>

	name here. For more information, see Configure instance pooling below.	Tip: If you specify a <code>driverPoolName</code> value only, then you can use the instance pools with the same <code>driverPoolName</code> available across multiple Databricks workspaces when you create a new cluster.
<code>databricks.logsDestination</code>	DBFS location that cluster logs will be sent to every 5 minutes	Leave this value as <code>/trifacta/logs</code> .
<code>databricks.enableAutotermination</code>	Set to true to enable auto-termination of a user cluster after N minutes of idle time, where N is the value of the <code>autoterminationMinutes</code> property.	Unless otherwise required, leave this value as <code>true</code> .
<code>databricks.clusterStatePollerDelayInSeconds</code>	Number of seconds to wait between polls for Azure Databricks cluster status when a cluster is starting up	
<code>databricks.clusterStartupWaitTimeInMinutes</code>	Maximum time in minutes to wait for a Cluster to get to Running state before aborting and failing an Azure Databricks job	
<code>databricks.clusterLogSyncWaitTimeInMinutes</code>	Maximum time in minutes to wait for a Cluster to complete syncing its logs to DBFS before giving up on pulling the cluster logs to the Trifacta node.	Set this to 0 to disable cluster log pulls.
<code>databricks.clusterLogSyncPollerDelayInSeconds</code>	Number of seconds to wait between polls for a Databricks cluster to sync its logs to DBFS after job completion	
<code>databricks.autoterminationMinutes</code>	Idle time in minutes before a user cluster will auto-terminate.	Do not set this value to less than the cluster startup wait time value.
<code>databricks.enableLocalDiskEncryption</code>	Enables encryption of data like shuffle data that is temporarily stored on cluster's local disk.	-
<code>databricks.patCacheTTLInMinutes</code>	Lifespan in minutes for the Databricks personal access token in-memory cache	Default: 10
<code>databricks.maxAPICallRetries</code>	Maximum number of retries to perform in case of 429 error code response	Default: 5. For more information, see Configure Maximum Retries for REST API section below.
<code>spark.useVendorSparkLibraries</code>	When <code>true</code> , the platform bypasses shipping its installed Spark libraries to the cluster with each job's execution.	NOTE: This setting is ignored. The vendor Spark libraries are always used for Azure Databricks.

Configure instance pooling

Instance pooling reduces cluster node spin-up time by maintaining a set of idle and ready instances. The Designer Cloud powered by Trifacta platform can be configured to leverage instance pooling on the Azure Databricks cluster for both worker and driver nodes.

NOTE: When instance pooling is enabled, the following parameters are not used:

`databricks.driverNodeType`

`databricks.workerNodeType`

For more information, see <https://docs.azuredatabricks.net/clusters/instance-pools/index.html>.

Instance pooling for worker nodes

Prerequisites:

- All cluster nodes used by the Designer Cloud powered by Trifacta platform are taken from the pool. If the pool has an insufficient number of nodes, cluster creation fails.
- Each user must have access to the pool and must have at least the `ATTACH_TO` permission.
- Each user must have a personal access token from the same Azure Databricks workspace. See Configure personal access token below.

To enable:

1. Acquire your pool identifier or pool name from Azure Databricks.

NOTE: You can use either the Databricks pool identifier or pool name. If both `poolId` and `poolName` are specified, `poolId` is used first. If that fails to find a matching identifier, then the `poolName` value is checked.

Tip: If you specify a `poolName` value only, then you can run your Databricks jobs against the available clusters across multiple Trifacta workspaces. This mechanism allows for better resource allocation and broader execution options.

2. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Set either of the following parameters:
 - a. Set the following parameter to the Azure Databricks pool identifier:

```
"databricks.poolId": "<my_pool_id>",
```

- b. Or, you can set the following parameter to the Azure Databricks pool name:

```
"databricks.poolName": "<my_pool_name>",
```

4. Save your changes and restart the platform.

Instance pooling for driver nodes

The Designer Cloud powered by Trifacta platform can be configured to use Databricks instance pooling for driver pools.

To enable:

1. Acquire your driver pool identifier or driver pool name from Databricks.

NOTE: You can use either the Databricks driver pool identifier or driver pool name. If both `driverPoolId` and `driverPoolName` are specified, `driverPoolId` is used first. If that fails to find a matching identifier, then the `driverPoolName` value is checked.

Tip: If you specify a `driverPoolName` value only, then you can run your Databricks jobs against the available clusters across multiple Trifacta workspaces. This mechanism allows for better resource allocation and broader execution options.

2. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
3. Set either of the following parameters:
 - a. Set the following parameter to the Databricks driver pool identifier:

```
"databricks.driverPoolId": "<my_pool_id>",
```

b. Or, you can set the following parameter to the Databricks driver pool name:

```
"databricks.driverPoolName": "<my_pool_name>",
```

4. Save your changes and restart the platform.

Configure Databricks job management

Azure Databricks enforces a hard limit of 1000 created jobs per workspace, and by default cluster jobs are not deleted. To support jobs more than 1000 jobs per cluster, you can enable job management for Azure Databricks.

NOTE: This feature covers the deletion of the job definition on the cluster, which counts toward the enforced limits. The Designer Cloud powered by Trifacta platform never deletes the outputs of a job or the job definition stored in the platform. When cluster job definitions are removed, the jobs remain listed in the Job History page, and job metadata is still available. There is no record of the job on the Azure Databricks cluster. Jobs continue to run, but users on the cluster may not be aware of them.

Tip: Regardless of your job management option, when you hit the limit for the number of job definitions that can be created on the Databricks workspace, the platform by default falls back to using the runs /submit API, if the Databricks Job Runs Submit Fallback setting has been enabled.

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following property and set it to one of the values listed below:

```
Databricks Job Management
```

Property Value	Description
Never Delete	(default) Job definitions are never deleted from the Azure Databricks cluster.
Always Delete	The Azure Databricks job definition is deleted during the clean-up phase, which occurs after a job completes.
Delete Successful Only	When a job completes successfully, the Azure Databricks job definition is deleted during the clean-up phase. Failed or canceled jobs are not deleted, which allows you to debug as needed.
Skip Job Creation	For jobs that are to be executed only one time, the Designer Cloud powered by Trifacta platform can be configured to use a different mechanism for submitting the job. When this option is enabled, the Designer Cloud powered by Trifacta platform submits jobs using the run-submit API, instead of the run-now API. The run-submit API does not create an Azure Databricks job definition. Therefore the submitted job does not count toward the enforced job limit.
Default	Inherits the default system-wide setting.

3. When this feature is enabled, the platform falls back to use the runs/submit API as a fallback when the job limit for the Databricks workspace has been reached:

```
Databricks Job Runs Submit Fallback
```

4. Save your changes and restart the platform.

Configure Databricks job throttling

By default, Databricks workspaces apply limits on the number of jobs that can be submitted before the cluster begins to fail jobs. These limits are the following:

- Maximum number of concurrent jobs per cluster
- Max number of concurrent jobs per workspace
- Max number of concurrent clusters per workspace

Depending on how your clusters are configured, these limits can vary. For example, if the maximum number of concurrent jobs per cluster is set to 20, then the 21st concurrent job submitted to the cluster fails.

To prevent unnecessary job failure, the Designer Cloud powered by Trifacta platform submits the throttling of jobs to Databricks. When job throttling is enabled and the 21 concurrent job is submitted, the Designer Cloud powered by Trifacta platform holds that job internally the first of any of the following events happens:

- An active job on the cluster completes, and space is available for submitting a new job. The job is then submitted.
- The user chooses to cancel the job.
- One of the timeout limits described below is reached.

NOTE: The Designer Cloud powered by Trifacta platform supports throttling of jobs based on the maximum number of concurrent jobs per cluster. Throttling against the other limits listed above is not supported at this time.

Steps:

Please complete the following steps to enable job throttling.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. In the Designer Cloud application, select **User menu > Admin console > Admin settings**.
3. Locate the following settings and set their values accordingly:

Setting	Description
<code>databricks.userClusterThrottling.enabled</code>	When set to <code>true</code> , job throttling per Databricks cluster is enabled. Please specify the following settings.
<code>databricks.userClusterthrottling.maxTokensAllottedPerUserCluster</code>	Set this value to the maximum number of concurrent jobs that can run on one user cluster. Default value is 20.
<code>databricks.userClusterthrottling.tokenExpiryInMinutes</code>	The time in minutes after which tokens reserved by a job are revoked, irrespective of the job status. If a job is in progress and this limit is reached, then the Databricks token is expired, and the token is revoked under the assumption that it is stale. Default value is 120 (2 hours). <div style="border: 1px solid #c8e6c9; border-radius: 10px; padding: 5px; text-align: center;">Tip: Set this value to 0 to prevent token expiration. However, this setting is not recommended, as jobs can remain in the queue indefinitely.</div>
<code>jobMonitoring.queuedJobTimeoutMinutes</code>	The maximum time in minutes in which a job is permitted to remain in the queue for a slot on Databricks cluster. If this limit is reached, the job is marked as failed.
<code>batch-job-runner.cleanup.enabled</code>	When set to <code>true</code> , the Batch Job Runner service is permitted to clean up throttling tokens and job-level personal access tokens.

Tip: Unless you have reason to do otherwise, you should leave this setting to `true`.

4. Save your changes and restart the platform.

Configure for Databricks Secrets Management

Optionally, you can leverage Databricks Secrets Management to store sensitive Databricks configuration properties. When this feature is enabled and a set of properties are specified, those properties and their values are stored in masked form. For more information on Databricks Secrets Management, see <https://docs.databricks.com/security/secrets/index.html>.

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 properties and set accordingly:

Setting	Description
<code>databricks.secretNamespace</code>	If multiple instances of Designer Cloud Powered by Trifacta Enterprise Edition are using the same Databricks cluster, you can specify the Databricks namespace to which these properties apply.
<code>databricks.secrets</code>	An array containing strings representing the properties that you wish to store in Databricks Secrets Management. For example, the default value stores a recommended set of Spark and Databricks properties: <pre>["spark.hadoop.dfs.adls.oauth2.client.id", "spark.hadoop.dfs.adls.oauth2.credential", "dfs.adls.oauth2.client.id", "dfs.adls.oauth2.credential", "fs.azure.account.oauth2.client.id", "fs.azure.account.oauth2.client.secret"],</pre> You can add or remove properties from this array list as needed.

3. Save your changes and restart the platform.

Configure personal access token

Each user must insert a Databricks Personal Access Token to access Databricks resources. For more information, see *Databricks Settings Page*.

Combine transform and profiling for Spark jobs

When profiling is enabled for a Spark job, the transform and profiling tasks are combined by default. As needed, you can separate these two tasks. Publishing behaviors vary depending on the approach. For more information, see *Configure for Spark*.

Additional Configuration

Enable SSO for Azure Databricks

To enable SSO authentication with Azure Databricks, you enable SSO integration with Azure AD. For more information, see *Configure SSO for Azure AD*.

Enable Azure Managed Identity access

For enhanced security, you can configure the Designer Cloud powered by Trifacta platform to use an Azure Managed Identity. When this feature is enabled, the platform queries the Key Vault for the secret holding the applicationId and secret to the service principal that provides access to the Azure services.

NOTE: This feature is supported for Azure Databricks only.

NOTE: Your Azure Key Vault must already be configured, and the applicationId and secret must be available in the Key Vault. See *Configure for Azure*.

To enable, the following parameters for the Designer Cloud powered by Trifacta platform must be specified.

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

Parameter	Description
azure.managedIdentities.enabled	Set to <code>true</code> to enable use of Azure managed identities.
azure.managedIdentities. keyVaultApplicationIdSecretName	Specify the name of the Azure Key Vault secret that holds the service principal Application Id.
azure.managedIdentities. keyVaultApplicationSecretName	Specify the name of the Key Vault secret that holds the service principal secret.

Save your changes.

Enable shared cluster for Databricks Tables

Optionally, you can provide to the Designer Cloud powered by Trifacta platform the name of a shared Databricks cluster to be used to access Databricks Tables.

Prerequisites:

NOTE: Any shared cluster must be maintained by the customer.

- Shared clusters are not provisioned per-user and cannot be used to run Spark jobs.
- If you have enabled table access control on your high-concurrency cluster, you must configure access to data objects for users. For more information, see <https://docs.databricks.com/security/access-control/table-acls/object-privileges.html>.
- If only a limited number of users are using the shared cluster, you must configure the attach permission on the shared cluster in the Databricks workspace.

Configure cluster

Depending on the credential provider type, the following properties must be specified in the Spark configuration for the shared cluster.

ADLS Gen1 Credentials:

```
"dfs.adls.oauth2.client.id"  
"dfs.adls.oauth2.credential"  
"dfs.adls.oauth2.refresh.url"  
"dfs.adls.oauth2.access.token.provider.type"  
"dfs.adls.oauth2.access.token.provider"  
"fs.azure.account.oauth2.client.endpoint"  
"spark.hadoop.dfs.adls.oauth2.client.id"  
"spark.hadoop.dfs.adls.oauth2.credential"  
"spark.hadoop.dfs.adls.oauth2.refresh.url"  
"spark.hadoop.fs.azure.account.oauth2.client.endpoint"  
"spark.hadoop.dfs.adls.oauth2.access.token.provider.type"  
"spark.hadoop.dfs.adls.oauth2.access.token.provider"
```

For more information, see <https://docs.databricks.com/data/data-sources/azure/azure-datalake.html>.

ADLS Gen2 Credentials:

```
"fs.azure.account.auth.type"  
"fs.azure.account.oauth.provider.type"  
"fs.azure.account.oauth2.client.id"  
"fs.azure.account.oauth2.client.secret"  
"fs.azure.account.oauth2.client.endpoint"  
"spark.hadoop.fs.azure.account.auth.type"  
"spark.hadoop.fs.azure.account.oauth.provider.type"  
"spark.hadoop.fs.azure.account.oauth2.client.id"  
"spark.hadoop.fs.azure.account.oauth2.client.secret"  
"spark.hadoop.fs.azure.account.oauth2.client.endpoint"
```

For more information, see <https://docs.databricks.com/data/data-sources/azure/azure-datalake-gen2.html>.

WASB Credentials:

```
fs.azure.sas.<container-name>.<storage-account-name>.blob.core.windows.net  
spark.hadoop.fs.azure.sas.<container-name>.<storage-account-name>.blob.core.windows.net  
azure.wasb.stores
```

For more information, see <https://docs.databricks.com/data/data-sources/azure/azure-storage.html>.

Enable

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 add the name of the Databricks cluster to use to browse Databricks Table:

```
"feature.databricks.connection.clusterName": "<your_cluster_name>",
```

3. Save your changes and restart the cluster.

When a cluster name is provided:

- If the cluster is available, all users of the Designer Cloud powered by Trifacta platform attempt to connect to Databricks Tables through the listed cluster.
- If the cluster has been terminated, a message indicates that the cluster must be restarted. After it has been restarted, you can try again.
- If the cluster name is invalid, the Designer Cloud powered by Trifacta platform fails any read/write operations to Databricks Tables.

NOTE: While using a single cluster shared across users to access Databricks Tables, each user must have a valid Databricks Personal Access Token to the shared cluster.

If a cluster name is not provided:

- In USER mode, the default behavior is to create one interactive cluster for each user to browse Databricks Tables.
- In JOB mode, the interactive cluster is created only when a user needs to browse Databricks Tables.

Configure user-level override for Databricks Tables access cluster

Individual users can specify the name of the cluster that accesses Databricks Tables through the Databricks settings. See "Specify Databricks Tables cluster name" below.

Specify Databricks Tables cluster name

Individual users can specify the name of the cluster to which they are permitted to access Databricks Tables. This cluster can also be shared among users. For more information, see *Databricks Settings Page*.

Pass additional Spark properties

As needed, you can pass additional properties to the Spark running environment through the `spark.props` configuration area.

NOTE: These properties are passed to Spark for all jobs.

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 property: `spark.props`.
3. Insert new Spark properties. For example, you can specify the `spark.props.spark.executor.memory` property, which changes the memory allocated to the Spark executor on each node by using the following in the `spark.props` area:

```
"spark": {  
  ...  
  "props": {  
    "spark.executor.memory": "6GB"  
  }  
  ...  
}
```

4. Save your changes and restart the platform.

For more information on modifying these settings, see *Configure for Spark*.

Configure Maximum Retries for REST API

There is a limit of 30 requests per second per workspace on the Databricks REST APIs. If this limit is reached, then a HTTP status code 429 error is returned, indicating that rate limiting is being applied by the server. By default, the Designer Cloud powered by Trifacta platform re-attempts to submit a request 5 times and then fails the job if the request is not accepted.

If you want to change the number of retries, change the value for the `databricks.maxAPICallRetries` flag.

Value	Description
5	<p>(default) When a request is submitted through the Azure Databricks REST APIs, up to 5 retries can be performed in the case of failures.</p> <ul style="list-style-type: none">• The waiting period increases exponentially for every retry. For example, for the 1st retry, the wait time is 10 seconds, 20 seconds for the next retry, 40 seconds for the third retry and so on.• You can set the values accordingly based on number of minutes /seconds you want to try.

0	<p>When an API call fails, the request fails. As the number of concurrent jobs increases, more jobs may fail.</p> <div data-bbox="237 195 1455 275" style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"> <p>NOTE: This setting is not recommended.</p> </div>
5+	<p>Increasing this setting above the default value may result in more requests eventually getting processed. However, increasing the value may consume additional system resources in a high concurrency environment and jobs might take longer to run due to exponentially increasing waiting time.</p>

Use

Run job from application

When the above configuration has been completed, you can select the running environment through the application.

NOTE: When a Databricks job fails, the failure is reported immediately in the Designer Cloud application . In the background, the job logs are collected from Databricks and may not be immediately available.

See *Run Job Page*.

Run job via API

You can use API calls to execute jobs.

Please make sure that the request body contains the following:

```
"execution": "databricksSpark",
```

For more information, see <https://api.trifacta.com/ee/es.t/index.html#operation/runJobGroup>

Troubleshooting

Spark job on Azure Databricks fails with "Invalid spark version" error

When running a job using Spark on Azure Databricks, the job may fail with the above invalid version error. In this case, the Databricks version of Spark has been deprecated.

Solution:

Since an Azure Databricks cluster is created for each user, the solution is to identify the cluster version to use, configure the platform to use it, and then restart the platform.

1. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Acquire the value for `databricks.sparkVersion`.
3. In Azure Databricks, compare your value to the list of supported Azure Databricks version. If your version is unsupported, identify a new version to use.

NOTE: Please make note of the version of Spark supported for the version of Azure Databricks that you have chosen.

4. In the Designer Cloud powered by Trifacta platform configuration, set `databricks.sparkVersion` to the new version to use.

NOTE: The value for `spark.version` does not apply to Databricks.

5. Restart the Designer Cloud powered by Trifacta platform .
6. The platform is restarted. A new Azure Databricks cluster is created for each user using the specified values, when the user runs a job.

Spark job fails with "spark scheduler cannot be cast" error

When you run a job on Databricks, the job may fail with the following error:

```
java.lang.ClassCastException: org.apache.spark.scheduler.ResultTask cannot be cast to org.apache.spark.scheduler.Task
at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:616)
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1149)
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:624)
at java.lang.Thread.run(Thread.java:748)
```

The `job.log` file may contain something similar to the following:

```
2022-07-19T15:41:24.832Z - [sid=0cf0cff5-2729-4742-a7b9-4607ca287a98] - [rid=83eb9826-fc3b-4359-8e8f-7fbf77300878] - [Async-Task-9] INFO com.trifacta.databricks.spark.JobHelper - Got error org.apache.spark.SparkException: Stage 0 failed. Error: Job aborted due to stage failure: Task 0 in stage 0.0 failed 4 times, most recent failure: Lost task 0.3 in stage 0.0 (TID 3, ip-10-243-149-238.eu-west-1.compute.internal, executor driver): java.lang.ClassCastException: org.apache.spark.scheduler.ResultTask cannot be cast to org.apache.spark.scheduler.Task
at org.apache.spark.executor.Executor$TaskRunner.run(Executor.scala:616)
...
```

This error is due to a class mismatch between the Designer Cloud powered by Trifacta platform and Databricks.

Solution:

The solution is to disable the precedence of using the Spark JARs provided from the Designer Cloud powered by Trifacta platform over the Databricks Spark JARs. Please perform the following steps:

1. To apply this configuration change, login as an administrator to the Trifacta node. Then, edit `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Locate the `spark.props` section and add the following configuration elements:

```
"spark": {
  ...
  "props": {
    "spark.driver.userClassPathFirst": false,
    "spark.executor.userClassPathFirst": false,
    ...
  }
},
```

3. Save your changes and restart the platform.

Configure Security

Contents:

- *Harden Trifacta node*
 - *User Access*
 - *Client Security*
 - *Enable SSL*
 - *Session timeouts*
 - *Access logs*
 - *Databases*
 - *SSL for Trifacta databases*
 - *Configure Secure Access for Relational Connections*
 - *Enhance Cluster Security*
 - *Configure for secure impersonation*
 - *Configure for Kerberos Integration*
 - *Configure for KMS*
 - *Enable SSL for HttpFS*
 - *Enable SSL for Hive*
-

This section provides an overview of security features of the Designer Cloud powered by Trifacta® platform and links to configuration tasks for each area.

Harden Trifacta node

The following sections cover how to enhance security for the Trifacta node.

User Access

Configure Password Criteria

By default, the Designer Cloud application enforces very limited requirements on password strength.

By default, a password can be a single character with no other requirements. Please configure password requirements.

For more information, see *Configure Password Criteria*.

Change Admin Password

As soon as the Designer Cloud powered by Trifacta platform is operational, you should change the password on the admin account.

See *Change Admin Password*.

Single Sign-On

The Designer Cloud powered by Trifacta platform can integrate with Active Directory at the KDC/Kerberos level or directory level.

NOTE: SSO integration requires set up of an Apache server as a reverse proxy. Instructions are provided in the link below.

See *Configure SSO for AD-LDAP*.

Disable User Self-Register

Whether you use SSO or not, you should consider disabling user self-registration. When self-registration is disabled, an admin must provision individual users. See *Configure User Self-Registration*.

Application Timeouts

As needed, you can review and modify various application timeouts, which may need modification to meet your enterprise standards. For more information, see *Configure Application Limits*.

Client Security

Enable HTTP Strict-Transport security headers

HTTP Strict-Transport security headers force web browsers to use secure communications when interacting with the server and prevent any communications over insecure HTTP protocol.

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. Set the following setting to true:

```
"proxy.securityHeaders.httpsHeaders": true,
```

3. Save changes and restart the platform.

Enable Secure cookies

The web application requires use of cookies. Set the following flag to ensure use of secure cookies.

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. Set the following setting to true:

```
"webapp.session.cookieSecureFlag": true,
```

3. Save changes and restart the platform.

Enable Content Security Policy

Content Security Policy (CSP) is an added layer of security that helps to detect and mitigate some types of attacks, including cross-site scripting(XSS) and content injection attacks. CSP directives can be used to specify an allow-list of locations from which web applications may load resources.

NOTE: CSP is considered an optional, additional security measure for most applications. For the Designer Cloud application, additional measures including thorough input validation are deployed as the first line of defense.

Optionally, you can choose to enable Content Security Policy and the following directives in your web server configuration.

For more information on Content Security Policy, see <https://www.w3.org/TR/CSP2/>.

Enable CSP:

You can enable the use of Content Security Policy as needed.

- Login to the Trifacta node as an administrator.
- Edit:
 - For Nginx (default) web server: `/etc/nginx/conf.d/trifacta.conf`.
 - For Tripache web server: `/opt/trifacta/pkg3p/tripache/conf/httpd.conf`.
- Locate or insert the following header, which enables basic Content Security Policy.

```
# Content-Security-Policy: default-src 'self';
```

- The part after the colon is the directive. See below for additional information on the directives that you can insert.
- Remove the hashmark (#) at the front of the header to enable the policy.
- Save the file and restart the platform.

Additional options are described below.

Frame ancestors:

To prevent unwanted framing of the Designer Cloud application in other applications that may capture clicks and other behaviors, you can enable a content security policy that defines the accepted frame ancestors for use by the Tripache web server.

By default, the application sets the `X-frame` header to `DENY`. For additional security, you can enable one of the following content security policies for the Designer Cloud application to use.

NOTE: This solution applies only to the Tripache web application server. This solution is not applicable to the default Nginx web server.

Steps:

1. Edit the following file:

```
/opt/trifacta/pkg3p/tripache/conf/httpd.conf
```

2. Search for the `Content-Security-Policy` header in the file. If it is present, it may look like the following:

```
# Content-Security-Policy: frame-ancestors <source>;
```

- a. If it is not present, insert the above line.
3. Remove any leading hash marks (#) to enable it.
 4. Replace the `<source>` value with one of the following:

Value	Description
a host source	<p>Internet host by name or IP address, plus an optional URL scheme and/or port number, separated by spaces.</p> <ul style="list-style-type: none">• The host address may include an optional asterisk character (*) for a wildcard. You may use a wildcard again (*) for the port number to indicate that all legal ports are valid for the source.• Single quotes surrounding the host are not allowed. <p>Example:</p>

	<ul style="list-style-type: none"> • <code>http://*.example.com</code>: Matches all attempts to load from any subdomain of example.com using the http: URL scheme. • <code>mail.example.com:443</code>: Matches all attempts to access port 443 on mail.example.com. • <code>https://store.example.com</code>: Matches all attempts to access store.example.com using https:.
a scheme source	<p>A scheme such as <code>http:</code> or <code>https:</code>.</p> <ul style="list-style-type: none"> • The colon is required and scheme should not be quoted. • You can also specify data schemes (not recommended). <p>Other values:</p> <ul style="list-style-type: none"> • <code>data</code>: Allows data: URLs to be used as a content source. This is insecure; an attacker can also inject arbitrary data: URLs. Use this sparingly and definitely not for scripts. • <code>mediastream</code>: Allows mediastream: URIs to be used as a content source. • <code>blob</code>: Allows blob: URIs to be used as a content source. • <code>filesystem</code>: Allows filesystem: URIs to be used as a content source.
'self'	<p>Refers to the origin from which the protected document is being served, including the same URL scheme and port number. You must include the single quotes.</p> <p>Some browsers specifically exclude blob and filesystem from source directives. Sites needing to allow these content types can specify them using the <code>data</code> attribute.</p> <p>Example:</p> <pre>Content-Security-Policy: frame-ancestors 'self' https://www.example.org;</pre>
'none'	<p>Refers to the empty set of no URLs matches. The single quotes are required.</p> <p>Example:</p> <pre>Content-Security-Policy: frame-ancestors 'none';</pre>

For more information, see

<https://developer.mozilla.org/en-US/docs/Web/HTTP/Headers/Content-Security-Policy/frame-ancestors>.

5. Save your changes and restart the platform.

Enable SSL

Deploy Platform SSL Certificate

To enable HTTPS communications with the web application of the Designer Cloud powered by Trifacta platform, you must create and install an SSL certificate for use by the platform.

NOTE: After you have deployed an SSL certificate, you can enable secure headers and secure cookies to be used by the web application.

See *Install SSL Certificate*.

SSL for SMTP Server

If the platform is integrated with an SMTP email server, by default it assumes that the server supports SSL. If not, this capability must be disabled.

NOTE: Access to SMTP server is required for password reset communications.

See *Enable SMTP Email Server Integration*.

Session timeouts

For more information on these parameters, see *Configure Application Limits*.

Access logs

For some access logs, you can configure the fields that are included, which permits you to remove sensitive information like IP addresses. For more information, see *Configure Logging for Services*.

Databases

SSL for Trifacta databases

You can apply SSL secure access for connections to the Trifacta databases. For more information, see *Enable SSL for Databases*.

Configure Secure Access for Relational Connections

If you are enabling connections to relational databases, you must create and deploy a key file containing the credentials to use for your JDBC sources. These credentials are then used for encrypted access.

NOTE: Encrypted authentication with your JDBC resources is required.

- For more information on enablement, see *Relational Access*.
- For more information on security, see *Configure Security for Relational Connections*.

Enhance Cluster Security

These options security options enhance the security of communications between the Trifacta node and the integrated cluster.

Configure for secure impersonation

Secure impersonation enables users to securely access the Hadoop cluster through a dedicated user or set of users, which enables use of cluster security features and permissions structures.

NOTE: Secure impersonation requires Kerberos applied to the cluster.

See *Configure for Secure Impersonation*.

Configure for Kerberos Integration

If user access on your Hadoop cluster is secured via Kerberos, you can configure the platform to leverage this cluster security feature.

See *Configure for Kerberos Integration*.

Configure for KMS

Hadoop supports the use of encrypted transport to and from the cluster KMS system. Depending on the software distribution, configuration steps may vary.

NOTE: If KMS is enabled on the cluster, you must configure KMS for the Designer Cloud powered by Trifacta platform regardless of other security features enabled on the cluster.

See *Configure for KMS*.

Enable SSL for HttpFS

Optionally, you can enable SSL connections between the Designer Cloud powered by Trifacta platform and the cluster's instance of HttpFS. See *Enable HttpFS*.

Enable SSL for Hive

You can configure SSL access to Hive. See *Configure for Hive*.

Configure SSO for AD-LDAP

Contents:

- *Prerequisites*
 - *Verification for LDAP*
 - *Attribute Mapping*
 - *SSO Auth Methods for AD-LDAP*
 - *Configure for Native AD-LDAP SSO*
 - *Use SSL for Native SSO Auth*
 - *Listening Port*
 - *Enable initial bind as user*
 - *Configure for Apache Reverse Proxy*
 - *Set up the reverse proxy*
 - *Platform configuration for reverse proxy*
 - *Use SSL for AD-LDAP SSO Auth*
 - *Additional Configuration*
 - *Managing Principal Case*
 - *User Management*
-

The Designer Cloud powered by Trifacta® platform can be configured to provide single sign-on (SSO) logins with Active Directory/Lightweight Directory Access Protocol (AD/LDAP). These steps allow you to enable auto-provisioning of new users to the platform if they can authenticate through LDAP.

- If auto-provisioning is not desired, after completing the basic configuration, you can disable auto-provisioning using the steps listed in the Advanced Configuration section.
- Single Sign-On (SSO) authentication enables users to authenticate one time to access multiple systems. The SSO platform must translate its authentication into authentication methods executed against each system under SSO control. For more information, see https://en.wikipedia.org/wiki/Single_sign-on.

Do not use this configuration for the following:

- **SAML 2.0:** See *Configure SSO for SAML*.
- **Azure deployments:** See *Configure SSO for Azure AD*.

Prerequisites

1. You have already installed the Designer Cloud powered by Trifacta platform .
 - a. See *System Requirements*.
 - b. See *Install Software*.
2. Your enterprise uses AD/LDAP for User Identity and Authentication.
3. You have the following required pieces of LDAP information available:
 - a. The host and port of the AD/LDAP server against which to authenticate
 - b. The base DN
 - c. The bind DN and password
4. You have installed LDAP utilities (ldap-utils) on the Trifacta node.

Verification for LDAP

Before you enable SSO to LDAP, you can use the following steps to verify that the Designer Cloud powered by Trifacta platform is able to communicate with LDAP with the information provided by your LDAP administrator.

Steps:

1. If you have not done so already, verify that you have the required pieces of LDAP information. See the previous section.
2. To verify the above information, execute the following command:

```
ldapsearch -H ldaps://<LDAP_SERVER_HOSTNAME>:<PORT> -x -b '<BASE_DN>' -D '<BIND_DN>' -w '<BIND_PASSWORD>' '(objectClass=*)' ['<attribute-name>'] | less
```

NOTE: Be sure to use single quotes (') in your command.

where:

Parameter	Description
ldap (s)://<LDAP_SERVER_HOSTNAME>:<PORT>	Hostname and port number of the LDAP server, as provided by your LDAP administrator. Please use ldaps as the protocol.
<BASE_DN>	Base DN value provided by your LDAP administrator
<BIND_DN>	Bind DN value provided by your LDAP administrator
<BIND_PASSWORD>	Bind password value provided by your LDAP administrator
(objectClass=*)	Please include this string value.
<attribute-name>	(Optional) Comma-separated list of attributes to include back from the LDAP server.
less	(Optional) Pipes lengthy output to page-by-page display tool.

3. Anonymous binding: In the unlikely event that the LDAP server supports anonymous binding, the BIND_DN and BIND_PASSWORD values are not required, as in the following command:

```
ldapsearch -H ldaps://<LDAP_SERVER_HOSTNAME>:<PORT> -x -b '<BASE_DN>' '(objectClass=*)' ['<attribute-name>'] | less
```

4. Output:
 - a. Success: Entire LDAP sub-tree is piped to the less tool for display.
 - b. Failure: You have entered incorrect LDAP connection information. Please review the values and contact your LDAP administrator if needed.

Attribute Mapping

To provision users, the Designer Cloud powered by Trifacta platform requires user profile information from attributes. The following are the default mappings:

Platform User Profile Field	Default AD attribute	Default LDAP attribute
email address	userPrincipalName	mail
sso-principal	uid	uid
hadoop-principal	uid	uid
userName	sAMAccountName	cn

The following is an example output of ldapsearch:

```
# jguy, Users, 56bb81bb6782d97b5c37b0cb, example.com
dn: uid=jguy,ou=Users,o=56bb81bb6782d97b5c37b0cb,dc=example,dc=com
sn: Joe
cn: Joe Guy
```

```
objectClass: top
loginShell: /bin/bash
homeDirectory: /home/jguy
uid: jguy
gidNumber: 5143
mail: jguy@example.com
givenName: Joe
```

In the above output, the `cn`, `uid` and `mail` attributes from LDAP are used by the platform. In this case, the sso-principal, email address, and name attribute are correct for Designer Cloud powered by Trifacta platform use.

NOTE: If your output from `ldapsearch` indicates that you must use non-default attributes, you must modify the configuration based on your SSO method. Details are below.

SSO Auth Methods for AD-LDAP

The Designer Cloud powered by Trifacta platform supports the following methods for integrating with AD-LDAP SSO:

1. **Native AD-LDAP support:** Integrate to the enterprise SSO using platform-native configuration.

Tip: This method is recommended.

2. **Apache reverse proxy:** Integrate the platform with an Apache reverse proxy server hosted on the Trifacta node. This node provides the connection to enterprise SSO.

NOTE: This older version of SSO integration is likely to be deprecated in a future release.

Configure for Native AD-LDAP SSO

This section covers setting up platform-native SSO integration with enterprise AD-LDAP.

Limitations:

The following limitations apply to the platform-native version of SSO for AD-LDAP. If these limitations apply to your environment, please use the reverse proxy version instead.

- No support for multiple LDAP servers
- No support for LDAP with SASL
- No support for custom authentication form

Please complete the following steps to enable native platform integration with your enterprise LDAP provider.

Steps:

1. Administrators can apply this configuration change through the *Admin Settings Page* in the application. If the application is not available, the settings are available in `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. The following settings do not apply to this method of SSO integration. However:

NOTE: If you are switching from the reverse proxy method to this method, please verify that these settings are set to the values listed in the New Value column.

Setting	Description	New Value
"webapp.sso.enable"	Enables use of reverse proxy SSO by the Designer Cloud application .	If changing SSO methods, set this value to <code>false</code> .
"webapp.sso.disableAuthGateway"	When set to <code>true</code> , the reverse proxy server is disabled.	If changing SSO methods, set this value to <code>true</code> .
"webapp.sso.enableAutoRegistration"	Enables users to auto-register an account with the platform when they connect to the login page.	To enable automatic access with SSO-authenticated users, set this value to <code>true</code> . To require administrator provisioning of user accounts, set this value to <code>false</code> . For more information, see <i>Manage Users under SSO</i> .

3. Enable LDAP for the platform:

trifacta-conf.json setting	Description
"webapp.ldap.enabled"	Set this value to <code>true</code> to enable LDAP for the Designer Cloud application .

4. Configure location and properties of the enterprise LDAP server:

trifacta-conf.json setting	Description
"webapp.ldap.server.url"	The URL of the LDAP server
"webapp.ldap.server.searchFilter"	The search filter to use when querying the LDAP server for users. Default value is: <pre>(uid={{username}})</pre>
"webapp.ldap.server.searchBase"	The starting point on the LDAP server to begin the search for users. NOTE: This value must be populated. Example value: <pre>dc=example,dc=org</pre>
"webapp.ldap.server.searchAttributes"	Array of attributes to retrieve from the LDAP server for a user. Modify this value only if your identity provider is sending different attributes.
"webapp.ldap.server.internalCACertificatePath"	Path to the CA certificate to use when connecting to the LDAP server over <code>ldaps://</code> .
"webapp.ldap.server.bindDN"	The distinguished name used to bind to the LDAP directory. NOTE: By default, the Designer Cloud powered by Trifacta platform uses the admin credentials listed here to perform the initial bind to the active directory. If each user's account is configured to be able to browse directory objects, you can configure the platform to use the individual accounts to perform the full authentication. See "Enable initial bind as user" below.

	Example value: cn=admin,dc=example,dc=org
"webapp.ldap.server.bindCredentials"	Password for simple authentication to the LDAP server.

5. Configure the LDAP property mappings:

trifacta-conf.json setting	Description	LDAP Property
"webapp.ldap.mapping.ssoPrincipal"	LDAP user property defining a user's ssoPrincipal.	uid
"webapp.ldap.mapping.name"	LDAP user property defining a user's name.	cn
"webapp.ldap.mapping.hadoopPrincipal"	LDAP user property defining a user's hadoopPrincipal. NOTE: This value must be a case-sensitive match to the value of the LDAP attribute.	uid
"webapp.ldap.mapping.email"	LDAP user property defining a user's email.	mail

6. Save the file.
7. Restart the platform.
8. Test authentication.

Use SSL for Native SSO Auth

To enforce SSL connections to the platform, you can create and install an SSL certificate. This certificate is also used when platform-native LDAP integration is enabled. For more information, see *Install SSL Certificate*.

Listening Port

Defaults:

- If enabled, SSL utilizes port number 443.
- By default, the platform is configured to use 3005.

If needed, you can change the listening port for the platform to match the port required for your deployment. For more information, see *Change Listening Port*.

Enable initial bind as user

By default, the Designer Cloud powered by Trifacta platform utilizes a two-tiered mechanism for binding credentials to the LDAP server.

- The initial bind performs a Distinguished Name (DN) lookup using an admin account that you specify in the following parameters:

```
"webapp.ldap.server.bindDN"
"webapp.ldap.server.bindCredentials"
```

- The secondary bind uses the user's credentials.

Optionally, the Designer Cloud powered by Trifacta platform can be configured to send the user's credentials for the initial bind to the active directory.

NOTE: Each user's credentials must be configured to be able to browse directory objects.

To enable initial binding using the user's credentials, please set the following configuration flag.

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 the value to `true`:

```
"webapp.ldap.server.initialBindAsUser.enabled": false,
```

3. In the following parameter, add the distinguished name pattern to use to complete the bind. Example:

```
"webapp.ldap.server.initialBindAsUser.bindDnPattern": "uid={{username}},dc=example,dc=org",
```

4. Save your changes and restart the platform.

Configure for Apache Reverse Proxy

Set up the reverse proxy

Please complete the following steps to enable and configure the Apache reverse proxy server on the Trifacta node.

Steps:

1. Edit the following file:

```
/opt/trifacta/pkg3p/tripache/conf/conf.d/trifacta.conf
```

2. Add values for your LDAP environment for the following settings. For an Active Directory configuration, remove the comment from the first `Define` line and specify appropriate values for the following:

```
#####  
# Basic : LDAP Configuration  
#  
# Active Directory Mode. Uncomment below to enable Active Directory compatibility  
# Define ACTIVE_DIRECTORY "1"  
Define TF_LDAP_SERVER "ldap://SERVER:PORT"  
Define TF_LDAP_BASE_DN "<BASE_DN>"  
Define TF_LDAP_BIND_DN "<BIND_DN>"  
Define TF_LDAP_BIND_PASSWORD "<BIND_PASSWORD>"
```

3. By default, SSO access occurs over port 2443. If needed, you can set the port number for user access in the following setting:

NOTE: This value should not be set to any value that conflicts with other ports in use by the Trifacta node. For more information, see *System Ports*.

```
<VirtualHost *:2443>
```

NOTE: To complete the configuration to change the listening port, additional configuration is required after you complete this section. Instructions are in the following section.

4. Save the file.
5. Create an admin account. You can either:
 - a. Connect to the application using an AD-linked ID. The first AD account to connect to the application is auto-registered as an admin account.
 - b. Define an admin user under SSO. For more information, see *Create Admin Account*.
6. Save your changes.
7. Save the file and restart the platform. See *Start and Stop the Platform*.

Change Listening Port for Reverse Proxy SSO

By default, SSO access uses port number 2443. If you need to change that listening port, please complete the following steps.

NOTE: Please make sure that the listening port is not set to a value that conflicts with any of the other listening ports in use by the platform. For more information, see *System Ports*.

Steps:

1. If you have not done so already, in `trifacta.conf`, please set the `VirtualHost` value to the appropriate port number. See previous instructions.
2. Edit the following file:

```
/opt/trifacta/pkg3p/tripache/conf/conf.d/httpd.conf
```

3. Locate the following entries:

```
#Listen 12.34.56.78:80
Listen 2443
```

4. Change the value for 2443 to be the value you set in `trifacta.conf`.
5. Save the file.
6. Perform any other necessary configuration before restarting the platform.

Platform configuration for reverse proxy

Please complete the following steps in the platform.

Steps:

1. Administrators can apply this configuration change through the *Admin Settings Page* in the application. If the application is not available, the settings are available in `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
2. Configure the following settings:

Setting	Description	Value
"webapp.sso.enable"	Enables use of SSO by the Designer Cloud application .	Set this value to <code>true</code> .
"webapp."	This setting determines the SSO	Set this value to <code>false</code> .

sso.disableAuthGateway"	method to use.	
"webapp.sso.enableAutoRegistration"	Enables users to auto-register an account with the platform when they connect to the login page.	To enable automatic access with SSO-authenticated users, set this value to true . To require administrator provisioning of user accounts, set this value to false. For more information, see <i>Manage Users under SSO</i> .

3. Save the file and restart the platform.

Use SSL for AD-LDAP SSO Auth

To enforce SSL connections to the platform when the AD-LDAP method of SSO authentication is in use, please complete the following additional steps:

NOTE: By default, the Tripache proxy server runs on port 2443. Standard SSL port communications is 443. However, since the proxy runs as the `trifacta` user, instead of root user, port numbers below 1000 are not available. The default port number for the proxy server is fine.

Steps:

1. You must create an SSL certificate.

NOTE: In the following, complete the steps to generate the certificate only.

For more information, see *Install SSL Certificate* .

2. Install the certificate and key file in an area that is accessible to the `trifacta` user. Recommended location:

```
/opt/trifacta/conf/
```

3. Retain the paths to these two files.
4. Edit the following file:

```
opt/trifacta/pkg3p/tripache/conf/conf.d/trifacta.conf
```

5. Locate the following configuration block:

```
#####
# Basic: SSL for Trifacta Webapp
# Uncomment below after generating certificate & key.
# SSLEngine on
# SSLCertificateFile "/opt/trifacta/server.crt"
# SSLCertificateKeyFile "/opt/trifacta/server.key"
#####
```

6. Uncomment the three configuration lines above.
7. Insert the full path to the certificate file and the key file in the space provided.
8. Save the file.
9. Restart the platform.

Additional Configuration

Customize authentication form

When SSO is enabled, the default presented to users who are authenticating is very plain. If desired, you can customize the form using the following steps.

Steps:

1. Modify the following file on the Trifacta node to suit your style for the login screen:

```
/opt/trifacta/pkg3p/tripache/htdocs/login.html
```

NOTE: Do not modify the names of the form fields or the form action.

2. Edit the following file:

```
/opt/trifacta/pkg3p/tripache/conf/conf.d/trifacta.conf
```

3. Uncomment the line that contains the following:

```
Define FORM_AUTH "1"
```

4. Restart the platform.
5. Test the login page.

AD-LDAP filtering for Reverse Proxy SSO

User access can be limited based on AD/LDAP attributes. Typical scenarios restrict access based on membership of a group or value of an attribute.

- The SSO template configuration contains example filters, which are commented out by default.
- For more information, see https://httpd.apache.org/docs/2.4/mod/mod_authnz_ldap.html#requiredirectives.

Managing Principal Case

As needed, you can configure the Designer Cloud powered by Trifacta platform to force captured principal values to lowercase. This standardization is applied throughout the platform, which may prevent connectivity or impersonation issues due to case mismatches.

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, which govern case conversion in the platform of the SSO and Hadoop principals for LDAP SSO:

```
"webapp.ldap.mapping.ssoPrincipalToLowerCase": false,  
"webapp.ldap.mapping.hadoopPrincipalToLowerCase": true,
```

3. To force conversion to lowercase, set these values to `true`.
4. Save changes and restart the platform.

User Management

For more information, see *Manage Users under SSO*.

Configure SSO for SAML

Contents:

- *Limitations*
- *Prep*
 - *(Optional) Enable HTTPS*
 - *Encryption key requirements*
 - *Acquire IDP metadata file*
 - *Acquire SAML claims from your identity provider*
- *Configure SAML for the platform*
- *Managing Principal Case*
- *Configure Session Expiration Page*
- *User Management*

The Designer Cloud powered by Trifacta® platform can be configured to provide single sign-on (SSO) logins with Active Directory/Lightweight Directory Access Protocol (AD/LDAP). These steps allow you to enable auto-provisioning of new users to the platform if they can authenticate through LDAP.

NOTE: This SAML 2.0 solution applies to Designer Cloud Powered by Trifacta Enterprise Edition. You cannot apply this solution to cloud-based product editions.

- If auto-provisioning is not desired, after completing the basic configuration, you can disable auto-provisioning using the steps listed in the Advanced Configuration section.
- Single Sign-On (SSO) authentication enables users to authenticate one time to access multiple systems. The SSO platform must translate its authentication into authentication methods executed against each system under SSO control. For more information, see https://en.wikipedia.org/wiki/Single_sign-on.

Do not use this configuration for the following:

- **Enterprise LDAP-AD:** See *Configure SSO for AD-LDAP*.
- **Azure deployments:** See *Configure SSO for Azure AD*.

Limitations

NOTE: This solution is not supported on CentOS/RHEL 8.

- SAML 2.0 only. SAML 1.1 is not supported.
- By default, our SP uses transient NameID format. Not all SAML providers will accept transient. You may have to change the metadata file to use something like the following:

```
<NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</NameIDFormat>
```

NOTE: Spaces are not supported in values for `hadoopPrincipal` and `ssoPrincipal`. Suggested format: `lastName.firstName`.

Prep

Before you begin, you should create a backup of your `trifacta-conf.json` file.

(Optional) Enable HTTPS

If you prefer to have users connect to the platform over HTTPS, you should enable it before completing the SAML setup. For more information, see *Install SSL Certificate*.

Encryption key requirements

To enable secure auth using SAML, you must deploy the following keys to the Trifacta node.

NOTE: When the SAML setup script is executed, the following keys and certs are created for your use and stored in the default locations listed below. If you prefer, you can copy in your own keys and certificates for the platform to use. If the paths or filenames differ from the defaults listed below, you must modify the configuration, which is described later.

SAML key	Default path on Trifacta node
Public signing certificate	<code>/opt/trifacta/conf/.key/saml-signing-public-key.cert</code>
Private signing key	<code>/opt/trifacta/conf/.key/saml-signing-private-key.key</code>
Public decryption certificate	<code>/opt/trifacta/conf/.key/saml-decryption-public-key.cert</code>
Private decryption key	<code>/opt/trifacta/conf/.key/saml-decryption-private-key.key</code>

Acquire IDP metadata file

From your identity provider, please acquire the public metadata file and transfer it to the Trifacta node.

Notes:

- This file must be hosted on the node.
- If there are changes to the source of this file, a new version of the file must be transferred to the Trifacta node.

Please store the file in the following location:

```
/opt/trifacta/conf/idp-metadata.xml
```

After the file is transferred to the Trifacta node, the platform must be made aware of it. These steps are covered below.

Acquire SAML claims from your identity provider

Check the following SAML claims in your identify provider. Verify that it is sending the following pieces of information. Below are the default attributes that are expected by the platform:

NOTE: Please note any differences between the expected default attribute names below and the values in your identity provider. These values must be updated in the platform, as described later.

Information	Default SAML Attribute Name
Email address	mail
Last name	name
User ID	userPrincipalName

Tip: If you do not have access to the IDP configuration, you can search the response back from your IDP for the following:

```
<saml: Attribute Name="
```

The full value inside the double quotes must be set as the second value in the MellonSetEnv properties file. To see the decoded SAML response, you can use a Chrome plugin like 'rcFederation SAML, WS-Federation and OAuth tracer'.

Configure SAML for the platform

Please complete the following steps to configure the platform to use your enterprise SAML authentication:

Steps:

1. The following script must be run as the `root` user.
2. On the Trifacta node, navigate to the following directory:

```
cd /opt/trifacta/webapp/bin
```

3. Execute the following script:

```
$ ./saml-sp-metadata-generator.js <hostname>
```

where `<hostname>` is the host value for your Trifacta node. Do not include the protocol identifier (e.g. `http://`) or the port number as part of this value.

4. The above script outputs the following:

Tip: The objects, paths, and filenames generated by this script are automatically in place for use by the platform. To use other objects, you must configure the paths in the platform, as described later in this section.

Item	Description	How to Use
Signing Private key	Path to generated private key for signing	If the path is the default one and no asset exists there, then the setup script generates the asset for you.
Signing Certificate	Path to generated certificate for signing	See previous.
Encryption Private key	Path to generated private key for encryption	See previous.
Encryption Certificate	Path to generated certificate for encryption	See previous.

Metadata	Metadata file <code>saml-sp-metadata.xml</code> for your identity provider	See next step.
----------	--	----------------

- The `saml-sp-metadata.xml` file in the same directory where you executed the script can be uploaded to your identity provider.
- Administrators can apply this configuration change through the *Admin Settings Page* in the application. If the application is not available, the settings are available in `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
- Configure the following settings:

Setting	Description	Value
"webapp.sso.enable"	Enables use of SSO by the Designer Cloud application .	Set this value to <code>false</code> .
"webapp.sso.disableAuthGateway"	When SSO is enabled, this value should be set to <code>true</code> to disable the use of the reverse proxy server, which is not used in SAML authentication.	Set this value to <code>true</code> .
"webapp.sso.enableAutoRegistration"	Enables users to auto-register an account with the platform when they connect to the login page.	To enable automatic access with SSO-authenticated users, set this value to <code>true</code> . To require administrator provisioning of user accounts, set this value to <code>false</code> . For more information, see <i>Manage Users under SSO</i> .

- Enable use of SAML by the Designer Cloud application :

<code>trifacta-conf.json</code> setting	Description
"webapp.saml.enabled"	Set this value to <code>true</code> .
"webapp.saml.server.entityId"	Set this value to the URI of the enterprise SAML server.

- If your identity provider is sending attribute values that differ from the values expected by the platform, please configure those values in the following properties:

<code>trifacta-conf.json</code> setting	Description	SAML attribute
"webapp.saml.mapping.ssoPrincipal"	SAML profile attribute that defines a user's SSO principal. Spaces are not supported.	userPrincipalName
"webapp.saml.mapping.name"	SAML profile attribute that defines a user's name.	name
"webapp.saml.mapping.hadoopPrincipal"	SAML profile attribute that defines a user's Hadoop principal. Spaces are not supported.	userPrincipalName
"webapp.saml.mapping.email"	SAML profile attribute that defines a user's email.	mail

- Configure the path to IDP metadata file, which you should have already downloaded to the Trifacta node.

Tip: Unless you wish to move the file to a different directory, this value does not need to be changed.

<code>trifacta-conf.json</code> setting	Description
"webapp.saml.idpMetadataPath"	Path to the IDP metadata file that you downloaded to the Trifacta node.

NOTE: This value is required and should already be specified to the default location previously listed.

- Configure SAML call back URLs, if needed. These values do not require modifying in most cases.

--	--

trifacta-conf.json setting	Description
"webapp.saml.server.logoutCallbackUrl"	URL to which user is redirected after logout. This value must end with /saml/logout/callback.
"webapp.saml.server.callbackUrl"	URL to which user is redirected after authentication. This value must end with /saml/login/callback?redirect_to=/.

12. Configure paths to security certificates. Modify only if you have stored your keys in non-default locations or filenames:

trifacta-conf.json setting	Description	Default path
"webapp.saml.security.signingCertPath"	This signing certificate must be a public certificate that matches the private key.	/opt/trifacta/conf/.key/saml-signing-public-key.cert
"webapp.saml.security.privateCertPath"	This private key must match the public signing certificate. Authentication requests can be signed using RSA-SHA1. The private key must be in PEM format. Authentication requests can be signed using RSA-SHA1. To sign them you need to provide a private key in the PEM format.	/opt/trifacta/conf/.key/saml-signing-private-key.key
"webapp.saml.security.decryptionPvkPath"	This private key is used for decrypting any encrypted assertions received by the platform.	/opt/trifacta/conf/.key/saml-decryption-private-key.key
"webapp.saml.security.decryptionCertPath"	This public certificate must match the private key for decryption.	/opt/trifacta/conf/.key/saml-decryption-public-key.cert

13. (optional) By default, the Designer Cloud application applies the sha1 algorithm to transactions with the SAML identity provider. If needed, you can apply the following configuration changes to use a different supported algorithm:

Tip: Although it's not required, these values should match.

trifacta-conf.json setting	Description	Default value
"webapp.saml.security.signatureAlgorithm"	This algorithm is used for signing SAML requests. Supported algorithms: sha1 sha256 sha512	sha1
"webapp.saml.security.digestAlgorithm"	This algorithm is used for provided a digest of the signed data object retrieved from the identity provider. Supported algorithms: sha1 sha256 sha512	sha1

14. (optional) In some SAML environments, such as Active Directory Federation Services (AD FS), the SAML Identity Provider makes its own choice for what authentication factors to use when authenticating a user. In these environments, you may wish to disable a request for the authentication context with the identity provider. Users may encounter an error similar to the following:

"Authentication method 'X509, MultiFactor, MultiFactorFederated' by which the user authenticated with the service doesn't match requested authentication method 'Password, ProtectedTransport'"

trifacta-conf.json setting	Description
"webapp.saml.disableRequestedAuthnContext"	<p>When set to <code>true</code>, the Designer Cloud application does not include a request for a specific authentication context, which is unnecessary because the identity provider has already determined the authentication method.</p> <p>(default) When set to <code>false</code>, the Designer Cloud application requests the authentication context from the identity provider.</p> <div style="border: 1px solid green; padding: 5px; margin-top: 10px;"> <p>Tip: In most environments, this setting should be <code>false</code>, which is the default.</p> </div>

15. Save the file.

Managing Principal Case

As needed, you can configure the Designer Cloud powered by Trifacta platform to force captured principal values to lowercase. This standardization is applied throughout the platform, which may prevent connectivity or impersonation issues due to case mismatches.

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, which govern case conversion in the platform of the SSO and Hadoop principals for SAML SSO:

```
"webapp.saml.mapping.ssoPrincipalToLowerCase": false,
"webapp.saml.mapping.hadoopPrincipalToLowerCase": true,
```

3. To force conversion to lowercase, set these values to `true`.
4. Save changes and restart the platform.

Configure Session Expiration Page

By default under SSO, manual logout and session expiration logout redirect to different pages. Manual logout directs you to SAML sign out, and session expiry produces a session expired page.

If desired, you can redirect the user to a different URL on session expiry:

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. Specify the URL of the page to which you wish to redirect users after a session has timed out:

```
"webapp.session.redirectUriOnExpiry": "<myPreferredSessionExpiryURL>",
```

3. Save your changes and restart the platform.

User Management

For more information, see *Manage Users under SSO*.

Configuring Platform Users

These topics cover setting up user accounts during initial installation of the Designer Cloud powered by Trifacta® platform .

Change Admin Password

As part of the install process, an admin user account is created. For security, this password should be changed.

NOTE: Some platform functions cannot be executed without an admin account. Your deployment should always have an admin account.

After the Trifacta software has been installed, the administrator of the system should immediately change the password for the admin account through the Designer Cloud application . If you do not know the admin account credentials, please contact *Alteryx Support*.

Steps:

1. Login to the application using the admin account.
2. In the menu bar, click **User menu > Preferences**.
3. Click **Profile**.
4. Enter a new password, and click **Save**.
5. Logout and login again using the new password.

Enable SMTP Email Server Integration

Contents:

- *Limitations*
- *Configure SMTP*
 - *Configure email sender*
- *Test SMTP*
- *Enable Email Notifications*
 - *Configure Notifications*

The Designer Cloud powered by Trifacta® platform can be integrated with an existing SMTP server for delivering emails to users of the platform. This configuration is necessary if the following features are enabled:

- Users are required to validate their email addresses after self-registration. Both of these options must be enabled. See *Configure User Self-Registration*.
- Users are permitted to reset their passwords. See *Enable Self-Service Password Reset*.

Limitations

NOTE: Installation and use of custom certificates on the Trifacta node is not supported.

Configure SMTP

To enable, please complete the following.

NOTE: If you are planning to enable SSL for communicating with the SMTP server, you must generate first a certificate for the server that is not self-signed.

NOTE: For more information on the settings to use in your environment, please contact your email administrator.

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. Edit the SMTP settings:

```
"smtp.host": "<SMTP_HOST>",  
"smtp.port": "587"  
"smtp.username": "<EMAIL_ACCT_USERNAME>",  
"smtp.password": "<EMAIL_ACCT_PWD>",  
"smtp.enableSSL": true  
"smtp.validateRegistrationEmail": false,  
"smtp.rejectUnauthorized": true,  
"smtp.authenticated": true,
```

Setting	Description
host	Hostname of the SMTP server

port	Port number to use to contact the SMTP server; the server listens for new email on this port. 587 is the default value. Tip: SMTP port 25 can be used as a listening port, too.
username	Username of the email account to use to send emails to users. NOTE: This username appears in the email message.
password	Password of the email account to use.
enableSSL	Set to true, if the SMTP server is configured to use SSL. Default is true. NOTE: When SSL is enabled for the email server, its certificate cannot be self-signed.
validateRegistrationEmail	When set to true, an email is sent to confirm registration to newly registered users, based on the configured SMTP server connection.
rejectUnauthorized	When set to true, the email server rejects any connection that is not authorized from the list of supported certificate authorities.
authenticated	If you use an unauthenticated SMTP connection, set this value to false.

3. Save your changes and restart the platform.

Configure email sender

You can configure the email address and display name of all emails sent from the Trifacta node.

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. Edit the following settings:

```
"webapp.emailSender": "<SENDER_EMAIL_ADDRESS>",
"webapp.emailSenderName": "<SENDER_DISPLAY_NAME>",
```

Setting	Description
emailSender	Enter an email account to use as the displayed sender of these emails. Tip: You should enter a value here to mask the real user account that is specified for the SMTP server connection.
emailSenderName	Display name of the sender of emails.

3. Save your changes and restart the platform.

Test SMTP

Trifacta administrators can send a test email to a specified email address using the configured SMTP settings. For more information, see *Admin Settings Page*.

Other email integration tests:

- If it's been enabled, you can reset user password. See *Enable Self-Service Password Reset*.
- If email registration validation has been enabled, you can create a new user account to verify that the SMTP server is working.
- If neither of the above is enabled, you can use an external testing method to verify that the SMTP server is working properly.

Enable Email Notifications

Optionally, you can enable the Designer Cloud powered by Trifacta platform to send email notifications on the status of job executions using the configured SMTP server.

NOTE: This feature requires access to an SMTP server to send emails.

Steps:

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

Parameter	Setting
Email notifications	Set this parameter to <code>enabled</code> to permit the platform to send email notifications. Default is <code>disabled</code> .
Email notifications: on Job Failure	Set this parameter to the default types of jobs that generate emails when they fail. Default is <code>scheduled</code> .
Email notifications: on Job Success	Set this parameter to the default types of jobs that generate emails when they succeed. Default is <code>never</code> .

Tip: The above defaults can be overridden for individual flows. See below.

- For more information on the available settings, see *Workspace Settings Page*.
3. To display correct images and links in the emails, please do the following:
 - a. You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
 - b. Locate the following parameter. Verify that its value is set to the host and port number of the Designer Cloud powered by Trifacta platform :

```
"webapp.hostUrl": "http://example.com:3005"
```

- c. Save your changes and restart the platform.

Configure Notifications

You can configure email notifications to be sent based on job executions from individual flows. Non-collaborating watchers can be informed, as well. These settings override the workspace defaults. See *Manage Flow Notifications Dialog*.

Configure User Self-Registration

By default, Trifacta® users can register themselves for an account, which immediately grants access to the platform. If preferred, registration can be gated by the admin user, so that all users are verified by an admin prior to having access to the platform.

- When single sign on (SSO) is enabled, self-registration is disabled.

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. Set the following parameter:

```
"webapp.selfRegistration" = true,
```

3. If you want users to validate their email addresses after registering, set the following parameter:

```
"webapp.validateRegistrationEmail" = true,
```

4. Save your changes and restart the platform.

NOTE: If your environment includes SSO or Kerberos for authentication, a workspace administrator must apply a principal value to user account before it can be used. See *Users Page*.

Enable Self-Service Password Reset

Contents:

- [Prerequisites](#)
- [Configuration](#)
- [Validate](#)

By default, the Designer Cloud powered by Trifacta® platform provides controls for administrators to reset passwords for themselves and for other users.

- For more information, see [Users Page](#).
- After users have logged in, they can reset their passwords via their Profile page. See [User Profile Page](#).

Optionally, Designer Cloud Powered by Trifacta Enterprise Edition customers can enable users to reset their passwords via email, which provides an extra measure of validation and security.

Prerequisites

1. Your enterprise must have an accessible SMTP email server that the Designer Cloud powered by Trifacta platform can use to send emails.
2. You should create a dedicated email account for issuing the password resets. The username and password for this account must be stored in platform configuration.

Configuration

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. Edit the following settings:

```
"webapp.emailTokenExpiryInMinutes": 1440,
```

Setting	Description
<code>emailTokenExpiryInMinutes</code>	Set the expiration time for the password reset email. Users must click through to reset their password before this time limit expires. The default value is 1440, which is 24 hours.

3. You can change the email address and display name of the sender of all emails from the Trifacta node. For more information, see [Enable SMTP Email Server Integration](#).
4. You apply this change through the [Workspace Settings Page](#). For more information, see [Platform Configuration Methods](#).

Setting	Description
Enable self service password reset	Set this value to <code>true</code> to enable this feature.

5. Save your changes and restart the platform.

Validate

1. Visit the login page for your Trifacta deployment. See [Login](#).

2. Click the **Forgot password?** link.
3. Reset your password.
4. When the email arrives, click the link to complete the password reset.

Configure User-Specific Props for Cluster Jobs

Contents:

- *Enable Java properties directory*
 - *Required Permissions*
- *Define user-specific properties files*

For individual users of the Designer Cloud powered by Trifacta platform, an administrator may like to submit a custom set of properties to the Hadoop cluster for each job execution. For example, you may wish to change the available Java heap size for users who submit large jobs. This section describes how to define and deploy user-specific Java properties files for cluster jobs.

NOTE: User-specific custom properties override or append any other custom properties that are passed to Hadoop. Suppose the Java properties file contains a single property.

- If the property is not specified elsewhere in the job definition, it is appended to any other properties that are passed.
- If the property is specified elsewhere in the job definition, the Java properties file overrides the other custom property value.

NOTE: You cannot specify user-specific properties for S3 jobs.

Enable Java properties directory

This feature is enabled by defining the values for the Java properties directory for Spark in the platform configuration.

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

Steps:

1. Set the following properties to the directories where the user-specific property files can be stored on the Trifacta node. Example:

```
"spark.userPropsDirectory": "/opt/trifacta/conf/usr",
```

2. Save your changes and restart the platform.

Required Permissions

For the above locations, the Trifacta service requires the following permissions:

- Execute permission on the directory defined in `spark.userPropsDirectory`.

- Read permission on the files in them.

Define user-specific properties files

For each user that is passing in custom properties, a separate file must be created in the appropriate directory with the following filename pattern:

```
userEmail-user.properties
```

where:

`userEmail` is the email address for the user that is registered with the Designer Cloud powered by Trifacta platform. For example, for `userId joe@example.com`, the filename is `joe@example.com-user.properties`.

File Format:

Each file must follow Java Properties file format, which is the following:

```
property.a=value.a  
property.b=value.b
```

NOTE: Do not include `spark.props` in your property names. For example, if you are modifying the Spark YARN queue (`spark.props.spark.yarn.queue`) for the user, the property name must be `spark.yarn.queue`.

For more information on this format, see

https://docs.oracle.com/cd/E23095_01/Platform.93/ATGProgGuide/html/s0204propertiesfileformat01.html.

Configure Users and Groups

Contents:

- *Enable*
 - *Enable and configure SSO*
 - *Configure platform*
 - *Create users*
- *Sync Users and Groups via API*
- *Testing - Share Flows and Connections*

The Designer Cloud powered by Trifacta® platform can be configured to support the use of groups for users.

NOTE: This feature is in Beta release.

Limitations:

- Group definitions must be pulled in from LDAP through a supported SSO integration.
 - You cannot create and manage groups from within the product.
 - You cannot import groups from other identity providers.
- Supported SSO integrations:
 - *Configure SSO for AD-LDAP* - platform native method
- Untested SSO integrations:
 - *Configure SSO for AD-LDAP* - reverse proxy method
 - *Configure SSO for SAML*
 - *Configure SSO for Azure AD*
- In this release, groups apply only to the sharing of connections and flows.

Enable

Enable and configure SSO

You must enable and configure one of the supported SSO integration methods.

Configure platform

Please review and set the following platform settings.

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 settings and apply values as needed:

Setting	Description
"feature.groups.enabled"	Set this value to <code>true</code> to enable use of LDAP groups in the platform.
"feature.groups.mapping"	Set this value to the LDAP search result parameter containing the value to be used as the name of a group in the Designer Cloud application. This value must have unique values, since groups in the Designer Cloud powered by Trifacta platform must have unique names.

groupName"	<p>Tip: "cn" is a good choice.</p>
"feature. groups. ldapServers"	<p>(optional) An array of parameters, listing LDAP servers to use for syncing of groups. If this parameter is not specified, then the LDAP server specified in the parameter <code>webapp.ldap.server</code> is used for syncing.</p>
"feature. groups. defaultGroupF ilters"	<p>(optional) You must provide at least one search filter string to use to query the LDAP servers for groups. The following example searches for all groups named <code>foo</code> and <code>bar</code>. In the UI:</p> <pre>(ou=foo),(ou=bar)</pre> <p>If editing this parameter through <code>trifacta-conf.json</code>, this value must be stored as an array with appropriate syntax:</p> <pre>["(ou=foo)", "(ou=bar)"]</pre> <p>Notes:</p> <p>A search filter doesn't need to use the <code>ou</code> parameter. Any valid LDAP search filter can be used.</p> <p>Each search filter must include parentheses at the beginning and the end.</p> <p>Each filter string is expected to return a single item. If the search results include multiple items, only the first item is used.</p> <p>If this value is empty, no groups are synced.</p>

3. Save your changes and restart the platform.

Create users

All users must be created in the Designer Cloud powered by Trifacta platform .

NOTE: The email address for the user in the Designer Cloud powered by Trifacta platform must match the LDAP email attribute.

- See *Users Page*.
- For more information on creating users via API, see <https://api.trifacta.com/ee/es.t/index.html#operation/createPerson>

Syncing:

After the platform users and groups have been synced with the LDAP identity provider:

- Any objects shared to a group are shared to individual users of the group as collaborators.
- If an LDAP user has no corresponding Designer Cloud powered by Trifacta platform user at the time of syncing, the platform user is automatically added to the group and inherits the group's permissions when the account is created.

NOTE: If a Designer Cloud powered by Trifacta platform user is removed from an LDAP group, the user remains a member of the platform group until groups are synced again. When groups are synced, the user is removed from the group and loses access to any objects shared with the group.

Sync Users and Groups via API

You can use the following endpoint to sync the platform with the configured LDAP servers for their groups.

NOTE: This endpoint must be triggered using an admin account.

Endpoint	http://www.example.com:3005/v4/groups/syncGroups
Authentication	Required
Method	POST
Request Body	Empty.
Response Status Code	200 - OK
Response Body	The response body contains the list of groups that have been added or removed based on the syncing: <pre>{ "data": [{ "ldap": "LDAP://www.ldap.example.com", "updatedGroups": [{ "id": 55, "members": [{ "id": 94, "email": "guest1@example.com", "name": "Guest Number One" }, { "id": 95, "email": "guest2@example.com", "name": "guest2 NumberTwo" }] }, { "name": "deptGRP1" }], "deletedGroups": [{ "id": 54, "name": "deptGRP2" }] }] }</pre>

cURL example:

- The backslash \ character indicates that the line continues on the following line.
- The following example references the use of an API token generated for the admin user. For more information, see *Manage API Access Tokens*.

```
curl -X POST \  
  http://www.example.com:3005/v4/groups/syncGroups \  
  -H 'authorization: Basic <auth_token>' \  
  -H 'cache-control: no-cache'
```

Testing - Share Flows and Connections

- **Flows:** You can share a flow to an imported group like you share with individual users. For more information, see *Share Flow Dialog*.
- **Connections:** You can share your connection to an imported group. For more information, see *Share Connection Dialog*.

Configure Features

Contents:

- *Platform Features*
 - *Enable deletion of jobs*
 - *Configure machine-learning transformation suggestions*
 - *Enable uploading profile pictures*
 - *Disable column lineage recipe highlighting*
 - *Configure publishing access controls*
 - *Enable publication of Datetime/Timestamp values to Parquet outputs*
 - *Disable custom types*
- *Additional Features*

In this section, you can review configuration options for specific features of Designer Cloud Powered by Trifacta® Enterprise Edition.

Feature flags are used to enable or disable features in the Designer Cloud powered by Trifacta platform . You can apply these changes through the Admin Settings page in the Designer Cloud application .

Platform Features

Through the Admin Settings page, platform administrators can search for the required feature flag by entering some or all of a feature flag name and modify values as needed.

NOTE: You must be an administrator to access this feature.

Tip: You can copy setting names from the documentation to search the available feature flags. Do not paste in double quotes from documentation samples.

NOTE: Except as noted, platform feature flags can be modified through the Admin Settings page in the Designer Cloud application by a platform administrator. For more information, see *Platform Configuration Methods* .

Do not modify settings through Admin Settings page and `trifacta-conf.json` at the same time. Saving changes in one interface wipes out any unsaved changes in the other interface.

As required, you can enable or disable a feature flag by performing the following steps:

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. Set the value as required:
 - a. Feature flags are used to enable or disable features. Typically, these are `true` or `false` values.

```
"<feature.flag>": true or false
```

where:

`<feature.flag>` is the name of the flag to modify. .

- b. Some features are enabled or modified using values other than `true` or `false`.
 - c. See below.
3. To save your changes, click **Save**.
 4. A platform restart is automatically executed when you save the changes.

For more information, see *Admin Settings Page*.

Feature	Description
Enable deletion of jobs	<p>When enabled, jobs can be deleted from Flow View page or from the Job History page through the context menu.</p> <ul style="list-style-type: none">• See <i>Job History Page</i>.• See <i>Flow View Page</i>. <p>Feature flag: <code>feature.enableJobDeletion</code></p> <p>Default value: <code>false</code></p>
Configure machine-learning transformation suggestions	<p>The Designer Cloud powered by Trifacta platform supports two methods of applying machine learning to the ranking of transformation suggestions in the Designer Cloud application .</p> <p>Enables the newer method of ranking transform suggestions based on machine learning. This method is executed in the client and is therefore faster.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"><p>NOTE: The older, server-side method has been deprecated and should not be used.</p></div> <p>Feature flag: <code>feature.mlTransformSuggestions.suggestionRankingModel.enabled</code></p> <p>Default value: <code>true</code></p>
	<p>Enables the old server-side method of ranking transform suggestions based on machine learning.</p> <div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"><p>NOTE: This parameter should not be enabled and will be deprecated in a future release.</p></div> <p>Feature flag: <code>feature.mlTransformSuggestions.enabled</code></p> <p>Default value : <code>false</code></p>
	<div style="border: 1px solid #ccc; padding: 5px; margin: 10px 0;"><p>NOTE: This parameter is applied only when <code>feature.mlTransformSuggestions.enabled</code> is <code>true</code> . Do not use this parameter.</p></div> <p>Feature flag: <code>feature.mlTransformSuggestions.delayThreshold</code></p> <p>Default value : <code>80 milliseconds</code></p>
Enable uploading profile pictures	<p>You can configure the Designer Cloud powered by Trifacta platform to all users to upload images to be used as their profile pictures.</p> <p>Feature flag: <code>webapp.enableProfilePicture</code></p> <p>Default value : <code>true</code></p>
Disable column lineage recipe highlighting	<p>By default, the Designer Cloud application can display lineage highlighting of applicable steps for a selected column.</p> <p>Feature flag: <code>webapp.enableColumnLineageScriptHighlighting</code></p> <p>Default value : <code>true</code></p>

Configure publishing access controls	<p>When this feature is enabled, job results are always written with permissions inherited from the parent directory of the target location.</p> <p>Feature flag: feature.copyFileToDestination</p> <p>Default value : false</p>
Enable publication of Datetime/Timestamp values to Parquet outputs	<p>By default, Designer Cloud Powered by Trifacta Enterprise Edition publishes Datetime values as String values in Parquet format.</p> <p>Optionally, you can enable the generation of Datetime/Timestamp values from Datetime values.</p> <p>Feature flag: feature.outputParquetTimestamp</p> <p>Default value: false</p>
Disable custom types	<p>By default, you can create custom data types in Designer Cloud Powered by Trifacta Enterprise Edition. When a column is set to a custom data type, the values in the column are validated against the type specification.</p> <div data-bbox="557 636 1456 728" style="border: 1px solid red; padding: 5px; text-align: center;"> <p>After a custom data type has been created, it cannot be removed from the platform.</p> </div> <p>Feature flag: feature.enableCustomTypes</p> <p>Default value: true</p>

Additional Features

In addition to enabling the feature, you can configure additional settings for the following features.

Configure Deployment Manager

Contents:

- *Enable Export and Import*
- *Enable Dev-Only Environment*
- *Enable Prod-Only Environment*
 - *User Management for Prod-only*
- *Enable All-in-One Environment*
 - *User management for All-in-One environment*
 - *Switching roles*

The Designer Cloud powered by Trifacta® platform supports multiple types of platform environments. Through the Deployment Manager, you can deploy your flows from a development instance to a production instance.

Tip: When you initially set up a platform instance, you should decide whether it is a Dev instance, a Prod instance or both. Details are below.

NOTE: Assignment of roles must be executed through the Admin Settings page. You cannot assign roles through API commands.

Platform Instance Type	Description	User Management	Default?
Development (Dev) only	A Development instance of the platform is used to build and test your flows and recipes until they are ready for operational use in production.	All users can build and execute flows. No users can access the Deployments area.	Yes
Production (Prod) only	A Production instance serves to host production versions of your flows, to manage the versions that are in use, and to execute jobs. It is primarily a read-only instance of the platform. You cannot access the Transformer page to modify your recipes in a Production instance. When a flow is ready for production use, you can export the flow from the Dev instance and import it into the Prod instance. For more information, see <i>Overview of Deployment Manager</i> .	No users can build and execute flows. All users access the Deployments area.	No
Both (All-in-One)	A Development environment can be configured to serve as both instance types.	Users can access the Deployments area only if the Deployments role has been added to their accounts.	No

Enable Export and Import

The ability to export and import flow packages must be enabled in your environment.

Steps:

1. Login as a workspace administrator.
2. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.

3. Locate the following settings, and verify that they are set to `Enabled`.

```
Export
Import
```

4. Test the export of a flow.

Enable Dev-Only Environment

Deployment Manager configuration is required.

NOTE: Do not include the Deployment role in any users accounts. See *Users Page*.

Enable Prod-Only Environment

If you are installing separate instances of the Designer Cloud powered by Trifacta platform to serve as Dev/Test and Prod environments, you can configure the Prod environment to serve only production purposes. Users who are permitted access to this environment can create and manage deployments, releases within them, and jobs triggered for these releases.

Tip: Separate Dev and Prod platform instances is recommended.

By default, the installed instance of the platform is configured as a Development instance. To configure the installed platform to operate as a Production instance, please complete the following steps.

NOTE: If you are enabling a Production-only instance of the platform, you should verify that you have deployed sufficient cluster resources for executing jobs and have sufficient nodes and users in your Trifacta license to support it. For more information, see *Overview of Deployment Manager*.

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. Configure the following setting to be `true`:

```
"deploymentManagement.enabled" : true,
```

3. Save your changes and restart the platform.

User Management for Prod-only

You must create accounts in the Prod instance for users who are to be permitted to create and manage deployments.

Tip: You should limit the number of users who can access a Production environment.

Tip: A deployment user should be assigned the flow author role for the workspace. Lesser flow roles may prevent the deployment user from properly importing and managing flows. See *Roles Page*.

NOTE: Any user who has access to a Production-only instance of the platform can perform all deployment-related actions in the environment. The Deployment role does not apply. For more information, see *Users Page*.

Enable All-in-One Environment

In this environment, individual user accounts may access development and testing features of the platform or the Deployment Manager, but not both. A user is a development user or a production user, based upon roles in the user's account.

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. Configure the following setting to be `false`:

```
"deploymentManagement.enabled" : false,
```

3. Save your changes and restart the platform.

User management for All-in-One environment

In this environment, access to Deployment Manager is determined by the presence of the Deployment role in a user's account:

When `deploymentManagement.enabled=false`:

Deployment role	Description
Not present in the account	User experiences the platform instance as a default Dev experience. User can create flows, recipes, and datasets, as well as run jobs on both a scheduled and ad-hoc basis.
Present in the account	User experiences the platform instance as a Prod environment. User can create and manage deployments and their releases. User can review connections and flows, although interaction may be limited. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">NOTE: Users should avoid making changes to flows in a Production environment. When a new release of a flow is imported, those changes are lost.</div>

Switching roles

In an All-in-One environment, administrators can change account permissions to enable or disable access to Prod features.

- Administrators should not apply these permission changes to admin accounts; use a separate account instead.
- If you switch the Deployment role on a single account, changes that you make to a Dev version of a flow are not automatically applied to a Prod version of the same flow, and vice-versa. You must still export the flow from one environment and import into the other to see any changes.

Configure Scheduling

Contents:

- *Limitations*
- *Prerequisites*
- *Enable*
- *Configure*
 - *Database configuration*

You can create scheduled executions of your flows in the Designer Cloud® application . When the schedule is triggered, all datasets that have scheduled output destinations are generated. Results are written to the specified output locations.

Limitations

- You can create one schedule per flow.

Prerequisites

You must install the two scheduling databases. See *Install Databases for PostgreSQL*.

Enable

By default, scheduling is enabled. Use the following steps to enable or disable.

Steps:

To enable, please specify the following parameters.

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

```
"feature.scheduling.protobufDefinitionsFolder": "%(topOfTree)s/services/scheduling-service/protobuf/build/install/scheduling-service/api",  
"feature.scheduling.baseUrl": "http://localhost:43143",
```

Parameter	Description
"feature.scheduling.protobufDefinitionsFolder"	Defines the location on the Trifacta node where the protobuf definitions are stored. <div style="border: 1px solid #ccc; padding: 5px; text-align: center;">NOTE: Do not modify this value.</div>
"feature.scheduling.baseUrl"	Specify the base URL for the scheduling service. You must specify the full URL: <div style="border: 1px solid #ccc; padding: 5px; margin-top: 5px;"><code>http://<base_platform_host>:<scheduling_port></code></div> <ul style="list-style-type: none">• <code><base_platform_host></code> is the host of the Trifacta node. Default value is <code>localhost</code>.• <code><scheduling_port></code> is the listening port for the scheduling service. Default value is <code>43143</code>.

NOTE: The value for `<scheduling_port>` must match the value that you specify for the scheduling service port. For more information, see *System Ports*.

You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.

Parameter	Description
Enable Scheduling feature	Set this value to <code>true</code> to enable scheduling.

Configure

Database configuration

You can make changes to the database configuration as needed.

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

For more information, see *Configure the Databases*.

Configure Sharing

You can share objects that you have created with other users or all users of Designer Cloud Powered by Trifacta® Enterprise Edition. This section provides information on how to configure it. For more information on sharing, see *Overview of Sharing*.

Enable

By default, sharing is enabled. The following parameters can be modified to enable/disable aspects of sharing.

You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.

Parameter	Default	Description
Enable Flow Sharing feature	true	Enable/disable the ability to share your flows with other users.

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

Parameter	Default	Description
<code>webapp.enableCredentialsSharing</code>	true	Enable/disable the sharing of credentials. <div style="border: 1px solid #ccc; padding: 10px; margin-top: 10px;"><p>NOTE: If this parameter is set to <code>false</code> and flow sharing is enabled, all users must provide their own credentials for each owned or shared connection.</p><ul style="list-style-type: none">If this parameter is set to <code>true</code>, then a connection's credentials can be shared at the connection owner's discretion.Later, if this parameter is set to <code>false</code>, the credentials of a previously shared connection are still shared, and you cannot choose whether or not to share credentials for pre-existing connections. You also cannot share credentials for any newly created connections.</div>

Access

NOTE: In Designer Cloud Powered by Trifacta Enterprise Edition, any user who is granted the admin role is also granted the workspace admin role, which enables owner-level access to some types of user-created objects in the workspace. See *Workspace Admin Permissions*.

Enable Flow Sharing feature	<code>webapp.enableCredentialsSharing</code>	Description
true	true	Flows: Users can share flows and connections. When flows are shared, the connections and their associated credentials are automatically shared. Shared connections can be used to import and/or publish data like owned connections. Connections: Connections can be shared.
true	false	

		<p>Flows: Users can share flows. When flows are shared, their connections are also shared, but new users must provide their own credentials to access the data.</p> <p>When new users provide credentials, the shared connections can be used to import and/or publish data like owned connections.</p> <p>Connections: When connections are shared, credentials must be provided by the new user.</p>
false	true	<p>Flows: Flows cannot be shared.</p> <p>Connections: Connections can be shared. The sharing user can choose whether or not to share credentials.</p>
false	false	<p>Flows: Flows cannot be shared.</p> <p>Connections: When connections are shared, credentials must be provided by the new user.</p>

Configure

Flows are shared through the Flow View page. See *Flow View Page*.

Credentials are shared through the Connections page. See *Connections Page*.

Configure Webhooks

Contents:

- *Limitations*
 - *Prerequisites*
 - *Enable*
 - *Configure*
 - *Limit URLs*
 - *Configure retries*
 - *Configure maximum number of webhooks per flow*
 - *Add signature header to webhook requests*
 - *Disable test button*
 - *Use*
-

A **webhook** is a notification sent from one application to another using HTTP requests. The Designer Cloud powered by Trifacta® platform can be configured to send webhook notifications to other applications based on the outcome of job executions. For example, you can configure the Designer Cloud powered by Trifacta platform to send a text message to a channel in your enterprise's messaging platform when jobs from a flow succeed or fail or both.

Limitations

- You cannot deploy an SSL certificate of a third-party application for use when sending webhook notifications from the Trifacta node.

Prerequisites

For more information, see *Create Flow Webhook Task*.

Enable

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the **Webhooks** settings.
3. Set this value to `true`.

See *Workspace Settings Page*.

Configure

Limit URLs

If necessary, you can restrict the IP address and URLs to which webhook requests can be sent.

NOTE: If you are permitting webhook requests to be submitted back to the Designer Cloud powered by Trifacta platform, you must verify that the IP address for the Trifacta node is not forbidden. See Security considerations below.

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. Modify the following settings:

Setting	Description
<code>webapp.webhookSettings.forbiddenIPs</code>	<p>This list contains the set of IP addresses that are not permitted to be sent webhook requests.</p> <div style="border: 1px solid #ccc; padding: 5px;"><p>NOTE: Do not remove the default values. These values prevent requests being sent to the Trifacta node.</p></div> <p>Addresses can be submitted in IPv4 or IPv6 format.</p> <p>Values are delimited by semi-colons.</p>
<code>webapp.webhookSettings.allowedUrlRegexes</code>	<p>As needed, you can create a regular expression to limit the permitted URLs to which webhooks can be sent.</p>

3. Save your settings and restart the platform.

Security considerations when whitelisting the platform

Webhooks can be configured to submit requests back to the Designer Cloud powered by Trifacta platform . For example, you can configure a webhook request to run a job after completion of a job. However, since a webhook request could be any REST API request to a target platform, it is potentially risky to enable webhook requests from the Designer Cloud powered by Trifacta platform to itself. To limit security risks:

- Where possible, specify URLs instead of IPs, which are more likely to change.
- The allowed URL regexes are processed first. For security, you can restrict access to the Trifacta node to only the permitted endpoint version `v4`:

```
webapp.webhookSettings.allowedUrlRegexes: ['^http:\\/\\/my\\.trifacta-instance\\.com\\/v4.*$']
```

- The Trifacta node must not be on the forbidden IP addresses.
- Webhooks should not be configured to use admin accounts.
- Accounts used for webhook requests should be limited in scope.

Tip: You can create a `webhook` user account, with which flows are shared. Then, jobs can be executed under this account, which is limited in scope.

- Webhook accounts should always be authenticated. Use of access tokens for these accounts is recommended.

Configure retries

If a webhook request fails to be received by the target application, the Designer Cloud powered by Trifacta platform retries sending the request. You can configure the following parameters pertaining to these retries:

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

Parameter	Description
<code>webapp.webhookQueue.maxRetries</code>	The maximum number of times that a failed webhook request is sent. Default is 1.

webapp.webhookQueue.retryDelayMs	The number of milliseconds between a failure and retrying to send the request. Default is 1000 (one second).
webapp.webhookSettings.timeoutMs	Global timeout setting for webhooks in milliseconds.

Configure maximum number of webhooks per flow

By default, users are permitted to create a maximum of 50 webhooks for each flow. If needed, you can change this limit.

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.webhookSettings.maxWebhooksPerFlow	The maximum number of webhooks that can be created in a flow. Default is 50.

Add signature header to webhook requests

As needed, you can sign the webhook requests that are submitted to a target platform.

NOTE: For each target application that requires a signed request, you must deploy a secret key in the request, and the target application must be defined to expect this key.

NOTE: Adding signatures may require custom coding or configuration in the target application.

For more information, see *Create Flow Webhook Task*.

Disable test button

When you create a new webhook task, you can optionally send a test webhook message to verify that the task is properly configured. As needed, this Test button can be disabled.

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

```
"webapp.webhookSettings.testWebhook": false,
```

3. Save your changes and restart the platform.

Use

Webhook notifications can be defined to be sent on the success or failure of executed jobs. These notifications are defined on a per-flow basis but can be restricted to individual outputs as needed. For more information, see *Create Flow Webhook Task*.

Enable API Access Tokens

Contents:

- *Enable*
- *Create and Use*
 - *Via UI*
 - *Via API*
- *Delete*
- *Disable*

For secure and flexible access to the REST APIs of the Designer Cloud powered by Trifacta® platform , you can enable access tokens. Each request via API requires some form of authentication. By using API access tokens, you can ensure that transfer of authentication information is minimized and obscured, and you can control the lifespan of these tokens.

Tip: Access tokens are the recommended method for managing access to the REST APIs.

- For more information on all supported forms of authentication via API, see <https://api.trifacta.com/ee/es.t/index.html#section/Authentication>
- For more information on how to use access tokens, see *Manage API Access Tokens*.

Enable

Steps:

1. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the following settings and configure them as needed:

Workspace Setting	Description
API Access Token	Set this value to Enabled to allow use of API access tokens.
Allow users to generate access tokens	Set this value to Enabled to allow non-admin users to generate access tokens. Set this value to Disabled to reserve access token generation for workspace administrators only.
Maximum lifetime for user generated access tokens (days)	Defines the maximum number of days that a user-generated access token is permitted to access the workspace. Set this value to -1 to enable unlimited lifetime tokens.

Create and Use

Via UI

You can create and delete tokens for personal use through the Settings area in the Designer Cloud application . For more information, see *Access Tokens Page*.

Via API

You can manage your access tokens through a specified set of REST APIs. See *Manage API Access Tokens*.

Delete

Workspace administrators can delete tokens of any user. For more information, see *Access Tokens Page*.

Disable

To disable this feature, please set the workspace setting to `Disabled`.

NOTE: Disabling this feature prevents all API users from using their tokens to access any endpoints.

Configure Services

This section contains information on how to configure the listed services of the Designer Cloud powered by Trifacta® platform .

NOTE: Some services are not configurable.

For more information on configuring logging, see *Configure Logging for Services*.

Configure Batch Job Runner

Contents:

- *Configure Timeout*
 - *Configure polling intervals*
- *Configure Thread Sizing*
 - *Configure job threads*
 - *Configure connection pool sizing*
- *Configure BJR for EMR*
 - *Multiple BJR instances*
 - *YARN logs from EMR*
- *Configure Database*
 - *Configure database cleanup*
 - *Configure Jobs database*
- *Logging*
- *Troubleshooting*
 - *Job fails with "Could not open JDBC Connection for transaction" error*

The Designer Cloud powered by Trifacta® platform utilizes the batch job runner service to orchestrate jobs that are executed on the selected backend running environment. This service passes jobs to the backend and tracks their progress until success or failure. This service is enabled by default.

Configure Timeout

Setting	Default Value	Description
<code>batchserver.spark.requestTimeoutMillis</code>	120000 (2 minutes)	Maximum number of milliseconds that the Batch Job Runner service should wait for a response from the Spark Job service during job execution. Default is 2 minutes.

Configure polling intervals

The following parameters can be modified to change the Batch Job Runner polling intervals for various types of jobs.

Setting	Default Value	Description
<code>jobMonitoring.ingestPollFrequencySeconds</code>	3	Polling interval in seconds for Batch Job Runner to check for status of ingest jobs.
<code>jobMonitoring.publishPollFrequencySeconds</code>	3	Polling interval in seconds for Batch Job Runner to check for status of publishing jobs.
<code>jobMonitoring.wranglePollFrequencySeconds</code>	3	Polling interval in seconds for Batch Job Runner to check for status of wrangling jobs.
<code>jobMonitoring.maxHeartbeatDelayMilliseconds</code>	7200000 (2 hours)	Duration in milliseconds for the service to wait for a job heartbeat before failing.

Configure Thread Sizing

Configure job threads

As needed, you can configure the number of worker threads assigned to each process that is managed by the batch job runner. Depending on the volume and complexity of jobs that you run of each type, you may choose to modify these settings to improve performance for key job types.

Tip: These settings can be configured through the Admin Settings page in the Designer Cloud application. See *Admin Settings Page*.

By running environment:

Setting	Default Value	Description
<code>batchserver.workers.photon.max</code>	2	Number of worker threads for running Trifacta Photon jobs. This value corresponds to the maximum number of photon jobs that can be queued at the same time. For more information, see <i>Configure Photon Running Environment</i> .
<code>batchserver.workers.spark.max</code>	16	Number of worker threads for running Spark jobs. For more information, see <i>Configure Spark Running Environment</i> .
<code>batchserver.workers.wrangle.max</code>	16	Number of worker threads for running transformation jobs.

By job type:

Setting	Default Value	Description
<code>batchserver.workers.ingest.max</code>	16	Maximum number of worker threads for running ingest jobs, which are used for loading relational data into the platform. After this maximum number has been reached, subsequent requests are queued.
<code>batchserver.workers.profile.max</code>	16	Maximum number of worker threads for running profile jobs, which provide summary and detail statistics on job results.
<code>batchserver.workers.publish.max</code>	16	Maximum number of worker threads for running publish jobs, which deliver pre-generated job results to other datastores.
<code>batchserver.workers.fileconverter.max</code>	16	Maximum number of worker threads for running fileconverter jobs, which are used to convert source formats into output formats.
<code>batchserver.workers.filewriter.max</code>	16	Maximum number of worker threads for running filewriter jobs, which are used for writing file-based outputs to a specified storage location.

Depending on your running environment, there may be additional parameters that you can configure to affect Batch Job Runner for that specific environment:

- *Configure for Spark*
- *Configure Photon Running Environment*

Configure connection pool sizing

The following thread pool sizing parameters should be configured in conjunction with each other:

```
"batch-job-runner.database.poolMaxSize": 32,
"batch-job-runner.asyncTaskScheduler.threadPoolSize": 32,
```

NOTE: If `batch-job-runner.database.poolMaxSize` is modified, please ensure that the new value is greater than the thread pool size for the task scheduler.

Parameter Name	Description
<code>batch-job-runner.database.poolMaxSize</code>	Maximum size of the database connection pool
<code>batch-job-runner.asyncTaskScheduler.threadPoolSize</code>	Maximum size of the task scheduler thread pool

Configure BJR for EMR

Multiple BJR instances

If the Designer Cloud powered by Trifacta platform is connected to an EMR cluster, multiple instances of the batch job runner are deployed to manage jobs across the cluster so that if one fails, YARN jobs are still tracked. No configuration is required.

YARN logs from EMR

The following properties below can be modified for batch job runner:

Setting	Default Value	Description
<code>aws.emr.getLogsOnFailure</code>	false	When set to <code>true</code> , YARN logs from all nodes in the EMR cluster are collected from S3 and stored on the Trifacta node in the following location: <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <code>/opt/trifacta/logs/jobs/<jobId>/container</code> </div> where: <code><jobId></code> is the Designer Cloud powered by Trifacta platform internal identifier for the job that failed.
<code>aws.emr.getLogsForAllJobs</code>	false	When set to <code>true</code> , YARN logs from nodes in the EMR cluster are collected and stored in the above location for all jobs, whether they succeed or fail. <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <p>NOTE: This parameter is intended for debugging purposes only.</p> </div>

Configure Database

Configure database cleanup

By default, the Jobs database, which is used by the batch job runner, does not remove information about jobs after they have been executed.

Logging:

- Batch Job Runner activities are surfaced in `batch-job-runner.log`. For more information, see *Configure Logging for Services*.
- Logging information for individual jobs is available in the `job.log` file written in the job directory. For more information, see *Diagnose Failed Jobs*.

As needed, you can enable the Designer Cloud powered by Trifacta platform to perform cleanup operations on the Jobs database.

NOTE: If cleanup is not enabled, the Jobs database continues to grow. You should perform periodic cleanups in conjunction with your enterprise database policies.

Steps:

To enable cleanup of the Jobs database, please complete 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 settings and set them accordingly:

Settings	Description
<code>batch-job-runner.cleanup.enabled</code>	Set this value to <code>true</code> to enable this feature.
<code>batch-job-runner.cleanup.interval</code>	The interval in ISO-8601 repeating intervals format at which batch-job-runner should clean outdated information about jobs. Default value is <code>R/PT1H</code> , which means that the job is executed once per hour.
<code>batch-job-runner.cleanup.maxAge</code>	The retention time for deployments in ISO-8601 interval format after which information about jobs is considered outdated. Default value is <code>P14D</code> , which means that the jobs information is cleaned out after 14 days.
<code>batch-job-runner.cleanup.maxDelete</code>	Maximum number of jobs whose information can be deleted per cleanup pass. Default value is <code>1000</code> .

For more information on ISO-8601 interval format, see https://en.wikipedia.org/wiki/ISO_8601#Repeating_intervals.

3. Save your changes and restart the platform.

Configure Jobs database

The Batch Job Runner utilizes its own Jobs database. For more information, see *Configure the Databases*.

Logging

For more information on logging for the service, see *Configure Logging for Services*.

Troubleshooting

Job fails with "Could not open JDBC Connection for transaction" error

When executing a job, you may encounter an error similar to the following:

NOTE: This issue can apply to jobs of various types: transformation, sampling, or schema refresh.

```
2022-01-30T15:40:19.418Z - [sid=] - [rid=] - [activiti-acquire-timer-jobs] ERROR org.activiti.engine.impl.  
asyncexecutor.AcquireTimerJobsRunnable - [method=] - [url=] - exception during timer job acquisition: Could  
not open JDBC Connection for transaction; nested exception is java.sql.SQLTransientConnectionException:  
HikariPool-1 - Connection is not available, request timed out after 30000ms.  
org.springframework.transaction.CannotCreateTransactionException: Could not open JDBC Connection for  
transaction; nested exception is java.sql.SQLTransientConnectionException: HikariPool-1 - Connection is not  
available, request timed out after 30000ms.  
    at org.springframework.jdbc.datasource.DataSourceTransactionManager.doBegin(DataSourceTransactionManager.  
java:309) ~[spring-jdbc-5.3.8.jar:5.3.8]  
    ....
```

Solution:

In this case, the Batch Job Runner service is unable to open a JDBC connection due to a lack of available connections in the database connection pool for the service. The solution is to review the connection pool settings for available threads and to ensure that the value for `batch-job-runner.database.poolMaxSize` is greater than or equal to `batch-job-runner.asyncTaskScheduler.threadPoolSize`.

For more information, see "Configure connection pool sizing" above.

Configure Data Service

Contents:

- [Configure SQL Options](#)
- [Configure relational read stream limits](#)

[Configure Caching](#)

[Enable Connection Pooling](#)

[Configure for Specific Integrations](#)

- [Configure Data Service for Hive](#)
- [Configure Data Service for Tableau Server](#)

[Additional Configuration](#)

- [Data service shutdown timeout](#)
- [Additional configuration areas](#)

[Logging](#)

[Other Topics](#)

The Data Service enables the Designer Cloud powered by Trifacta® platform to stream metadata and records from JDBC sources for sampling and job execution in the Trifacta Photon running environment. This section describes how to enable and configure the service, including performance tweaks and connection-specific configuration.

Configure Service

The following basic properties enable the service and specify basic location for it.

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

Property	Description
"data-service.enabled"	When true, the data service is enabled. <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;">NOTE: When set to false, access to any relational connection is prevented.</div> Default is true.
"data-service.host"	Hostname for the service. Default is localhost.
"data-service.port"	Port number used by the service. Default is 41912. <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;">NOTE: If you are changing the port number, avoid creating conflicts with existing ports in use. For more information, see System Ports.</div>
"data-service.classpath"	The Java class path for the data service.

"data-service. autoRestart"	When true, the data service is automatically restarted if it crashes. Default is true .
"data-service. vendorPath"	Path to the vendor configuration files for relational connections. Default value: <pre>%(topOfTree)s/services/data-service/build/conf/vendor</pre>

Configure SQL Options

Configure relational read stream limits

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

The Data Service reads data from relational sources in streams of records. You can modify the following parameters to configure the limits of SQL record streaming during read operations. The size of these streams are defined by the following parameters:

```
"data-service.sqlOptions.maxReadStreamRecords": -1,
"data-service.sqlOptions.limitedReadStreamRecords": 1000000,
"data-service.sqlOptions.initialReadStreamRecords": 25,
"data-service.sqlOptions.hiveReadStreamRecords": 100000000,
```

Property	Description
"data-service.sqlOptions.maxReadStreamRecords"	The maximum number of JDBC records pulled in per stream read during batch execution. If this value is set to <code>-1</code> , then no limit is applied.
"data-service.sqlOptions.limitedReadStreamRecords"	Max number of records read for the initial sample and quick scan sampling. Setting to <code>-1</code> means there is no limit.
"data-service.sqlOptions.initialReadStreamRecords"	Initial number of records to read for client-side preview and for client-side transform. Set to <code>-1</code> to apply no limit.
"data-service.sqlOptions.hiveReadStreamRecords"	Max number of records that can be read from Hive, if <code>maxReadStreamRecords</code> is <code>-1</code> . <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>NOTE: This value cannot be set to <code>-1</code>, which results in a Data Service error. Hive reads must be limited.</p> </div>

Configure Caching

The data service maintains a cache of JDBC objects that have been retrieved for use. You can configure the following properties to tune settings of the cache.

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

Property	Description
"data-service.cacheOptions.validationDelayMilliseconds"	Number of milliseconds to wait between checks validating cached pools. Default is 3600000 (1 hour).
"data-service.cacheOptions.maxSize"	Maximum number of objects in the cache. Default is 100.

	NOTE: Set this value to 0 to disable data service caching.
"data-service.cacheOptions.expirySeconds"	Objects in the cache that are older than this number of seconds are automatically expired. Default is 86400 (1 day).

Enable Connection Pooling

By default, JDBC connection pooling is disabled. Optionally, you can choose to enable the feature, although this is not recommended.

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

Property	Description
"data-service.connectionPooling.enabled"	<p>When set to <code>true</code>, connection pooling is enabled for JDBC-based connections. Connections are returned from a C3P0 connection pool. No other configuration is required.</p> <p>When set to <code>false</code>, connection pooling is disabled. Connections are established on-demand for specific jobs.</p> <p>By default, this flag is set to <code>false</code>.</p>

Configure for Specific Integrations

Configure Data Service for Hive

The following properties apply to how the platform connects to Hive.

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

Property	Description
"data-service.hiveManagedTableFormat"	Managed table format for your Hive deployment. Default is <code>PARQUET</code> .
"data-service.hiveJdbcJar"	Path to the JAR to use for JDBC connectivity to Hive. Default path depends on your Hadoop distribution.

Configure Data Service for Tableau Server

The following properties apply to how the platform publishes to Tableau Server.

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

Property	Description
"data-service.tableauBufferSizeInBytes"	<p>Number of bytes of data to include in each HTTP request chunk when publishing to Tableau Server. When the Designer Cloud application publishes a file to Tableau Server, the file is divided into chunks, and each chunk is attached as part of a HTTP request payload. This flag controls the chunk size in bytes.</p> <p>When the chunk size is large, the number of HTTP requests required to send the whole file to Tableau Server is smaller. However, a large chunk size increases the risk of a <code>RequestTimeoutException</code>, which causes the publishing job to fail.</p> <p>Default is 3000000 bytes.</p>

Additional Configuration

Data service shutdown timeout

If a shutdown or restart command is issued for the platform, the data service may be in the process of waiting for pending requests from various services before it can gracefully shut down. In some cases, the `supervisord` process, which governs platform starting and stopping, fails to restart while waiting for the data service to shut down.

By default, the data service waits for five seconds (5000 milliseconds) before automatically shutting down, regardless of the pending requests. If needed, you can adjust this shutdown timeout setting.

Do not modify this setting unless you are experiencing problems with how data service is interacting with other services during platform stop, start, or restart operations.

Notes:

- This value should be set to a higher value than the `stop-wait` setting for the `supervisord` service. Otherwise, the `supervisord` service may not wait long enough for the data service to complete its shutdown.
- To configure the `supervisord` setting:
 - Edit the following file:

```
/conf/supervisord.conf
```

- Locate the `stopwaitsecs` attribute.
- Edit the attribute to be higher than the value you are configuring for the setting below. The `supervisord` value is in seconds.

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 setting:

```
"data-service.shutdownTimeout": 5000,
```

3. Modify the value in milliseconds.
4. Save your changes and restart the platform.

Additional configuration areas

The following aspects of the data service can be configured outside of the application:

- Connection pool size and retry parameters
- Vendor field mappings
- Oracle ciphers for SSL connections
- JDBC fetch size by vendor

For more information, please contact *Alteryx Customer Success and Services*.

Logging

For more information on logging for the service, see *Configure Logging for Services*.

Other Topics

- If you are reading large datasets from relational sources, you can enable JDBC ingestion, which reads source data in the background and stages on the backend datastore for execution. For more information, see *Configure JDBC Ingestion*.
- Optionally, SSO authentication can be applied to relational connections. For more information, see *Enable SSO for Relational Connections*.

Configure Java VFS Service

Contents:

- *Configure*
- *Additional Configuration*
 - *Change memory for Java VFS service*
 - *Whitelist file storage protocols*
- *Logging*

The Java VFS Service is the next generation service for the virtual file system for the Designer Cloud powered by Trifacta® platform . This service is required for the following integrations:

- S3 Connections
- SFTP Connections
- ADLS Gen1
- ADLS Gen2
- WASB

In future releases, it may be applied to other components of the platform, instead of VFS Service. For more information on the earlier version, see *Configure VFS Service*.

Configure

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 configuration:

```
"java-vfs-service.enabled":true,  
"java-vfs-service.host":127.0.0.1,  
"java-vfs-service.port":41917,
```

Parameter	Description
<code>enabled</code>	Set this value to <code>true</code> to enable the Java VFS Service.
<code>host</code>	Host of the Java VFS Service. This host is used to listen to requests from the Designer Cloud application . Leave this value as <code>127.0.0.1</code> .
<code>port</code>	Port number that Java VFS Service uses to communicate. This port number is used to listen to requests from the Designer Cloud application . Default value is <code>41917</code> . This port must be opened on the Trifacta node. See <i>System Ports</i> .

3. Verify that the `enabled` parameter is set to `true`.
4. Locate the following properties. These properties define the address and port used by other services in the Designer Cloud powered by Trifacta platform to communicate with the Java VFS Service:

NOTE: These property values can differ with the `host` and `port` values specified above.

```
"java-vfs-service.systemProperties.server.host": 127.0.0.1,  
"java-vfs-service.systemProperties.server.port": 41917,
```

Parameter	Description
-----------	-------------

host	Internal address of the Java VFS Service. This host is used to listen to requests from other platform services. Leave this value as 127.0.0.1.
port	Port number that Java VFS Service uses to communicate with other platform services. Default value is 41917.

5. Save your changes and restart the platform.

Additional Configuration

Change memory for Java VFS service

In some cases, you may notice errors like the following in the Java VFS service logs:

```
java.lang.OutOfMemoryError: Java heap space
```

In this case, the service is running out of allocated memory. As needed, you can expand the memory allocated to the service:

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, which is set to 512 MB by default:

```
"java-vfs-service.jvmOptions": "-Xmx512m",
```

3. You can experiment with raising this value to 1024 MB (`-Xmx1024m`).
4. Save your changes and restart the platform.

Whitelist file storage protocols

When interacting with file storage systems, the Java VFS service checks the filestorage whitelist to verify that a particular file system is permitted. As needed, you can add other filesystem protocols to this list.

For more information on the values to insert here:

- S3 Access
- SFTP Connections
- ADLS Gen2 Access
- ADLS Gen1 Access
- WASB Access

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, which is set to 512 MB by default:

```
"filestorage.whitelist": "sftp,s3",
```

3. For each file system, you can configure the starting location for that system's protocol. In the following the root directory (`/`) is the starting directory for SFTP and S3 systems:

```
"fileStorage.defaultBaseUri": "sftp:///,s3:///",
```

-
4. Save your changes and restart the platform.

Logging

You can configure how logging is managed for the VFS service. For more information on configuring logging for the VFS service, see *Configure Logging for Services*.

Configure VFS Service

The VFS Service serves the front-end interface and brokers connections with the backend datastores. The VFS service is required when the Trifacta Photon client or the Trifacta® Photon running environment is enabled.

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

Configure

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 configuration:

```
"vfs-service.enabled":true,  
"vfs-service.host":"localhost",  
"vfs-service.port":41913,  
"vfs-service.bindHost":"0.0.0.0",  
"vfs-service.autoRestart":true,  
"vfs-service.timeoutMilliseconds":7200000,  
"vfs-service.numProcesses":2,
```

3. Verify that the `enabled` parameter is set to `true`.
4. Additional configuration settings are described below.
5. Save your changes and restart the platform.

Parameter	Description
<code>enabled</code>	Set this value to <code>true</code> to enable the VFS Service.
<code>host</code>	Host of the VFS Service. Leave this value as <code>localhost</code> .
<code>port</code>	Port number that VFS service uses to communicate. Default value is 41913. This port must be opened on the Trifacta node. See <i>System Ports</i> .
<code>bindHost</code>	Do not modify this value.
<code>autoRestart</code>	When set to <code>true</code> , the VFS Service automatically restarts and attempts to return to its pre-restart state. This value should be set to <code>false</code> for debugging purposes only.
<code>timeoutMilliseconds</code>	Timeout for requests to the VFS service. Default is 7200000 (2 hours).
<code>numProcesses</code>	Number of processes on the Trifacta node used by the VFS service. Default is 2.

Logging

You can configure how logging is managed for the VFS service. For more information on configuring logging for the VFS service, see *Configure Logging for Services*.

Configure Connector Configuration Service

The Connector Configuration Service manages the configuration of connections, their defaults, and their overrides.

Configure

NOTE: Except in rare cases, the other properties for this service do not need to be modified.

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

Property	Description
"connector-configuration-service.enabled"	Set this value to <code>true</code> . It is enabled by default.
"connector-configuration-service.autorestart"	Set this value to <code>true</code> to enable auto-restarting of the service.
"connector-configuration-service.host"	Host IP address of the Trifacta node. In most cases, this value should be <code>127.0.0.1</code> .
"connector-configuration-service.port"	Set this value to <code>41925</code> . See <i>System Ports</i> .
"connector-configuration-service.connectTimeoutMilliseconds"	Maximum time in milliseconds that a connection to the Connector Configuration Service database is permitted, before the connection is timed out. Default value is <code>30000</code> (30 seconds).
"connector-configuration-service.maxElapsedRetryTimeMilliseconds"	Maximum time in milliseconds that the service is permitted to retry if an <code>IOException</code> is encountered. Default value is <code>180000</code> (180 seconds).

3. Save your changes and restart the platform.

Logging

You can configure how logging is managed for the service. For more information on configuring logging, see *Configure Logging for Services*.

Configure Optimizer Service

Contents:

- *Configure Service*
 - *Additional configuration*
- *Use*
 - *Enable for workspace users*
 - *Enable optimizations for individual flows*
- *Logging*

The Optimizer service manages optimizations of job executions that involve queries to relational sources. Based on the enabled optimizations, the service can tune the SQL to optimize queries for better performance and less data transfer.

Prerequisites

The Optimizer service database must be installed. This database is installed as part of normal install and upgrade operations. For more information, see *Install Databases*.

Configure Service

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 configuration:

```
"optimizer-service.enabled":true,  
"optimizer-service.host":"localhost",  
"optimizer-service.port":41913,  
"optimizer-service.autoRestart":true,
```

3. Verify that the `enabled` parameter is set to `true`.
4. Additional configuration settings are described below.

Parameter	Description
<code>enabled</code>	Set this value to <code>true</code> to enable the Optimizer service.
<code>host</code>	Host of the Optimizer service. Leave this value as <code>localhost</code> .
<code>port</code>	Port number that Optimizer service uses to communicate. Default value is 41922. This port must be opened on the Trifacta node.
<code>autoRestart</code>	When set to <code>true</code> , the Optimizer service automatically restarts and attempts to return to its pre-restart state. This value should be set to <code>false</code> for debugging purposes only.

Additional configuration

The following parameters can be modified as needed.

Parameter	Description
<code>optimizer-service.maximumOptimizationTimeInSeconds</code>	Maximum time in seconds that the service is permitted to optimize a job. If this time is exceeded, the service errors out, and the job proceeds without optimization. Default value is 45.

	<p>This value may be increased if optimization is desired for any of the following:</p> <ul style="list-style-type: none"> • Very complex recipes • Extremely wide datasets • Jobs reading from several relational sources and pushing down filters to them
optimizer-service. maximumNumberOfWalksPerBatch	<p>Maximum number of passes the service is permitted to make on a recipe to optimize a job. If this number of passes is exceeded, the job is partially optimized. Default value is 300.</p> <p>This value may be increased if the full suite of optimizations is desired for complex recipes with more than 100 steps.</p>

Use

Enable for workspace users

A workspace administrator must enable optimizations for workspace users.

Steps:

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

```
Logical and physical optimizations of jobs
```

3. Users can now optimize jobs at the flow level. See below.

Enable optimizations for individual flows

For an individual flow, you can specify the set of optimizations to apply to the job. In Flow View, select **Optimization settings** from the flow context menu. For more information, see *Flow Optimization Settings Dialog*.

Logging

You can configure how logging is managed for the Optimizer service. For more information on configuring logging for the Optimizer service, see *Configure Logging for Services*.

Configure Orchestration Service

Contents:

- *Database Install*
- *Enable*
- *Plan Triggering*
- *Sizing*
- *Task Polling*
- *Logging*

The orchestration service orchestrates the execution of plans on the Designer Cloud powered by Trifacta® platform . This service manages the sequencing and delivery of specific tasks of a plan to the appropriate service for execution and monitors progress until success or failure. This service is enabled by default and interacts with a dedicated database.

Tip: The orchestration service is used only for execution of plans. If you are not using plans in your deployment of the Designer Cloud powered by Trifacta platform , this service is not required. You may want to install the database for potential use in the future.

Database Install

The Orchestration Service database is installed and upgraded inline with the other Trifacta databases. For more information, see *Install Databases*.

Enable

The following parameters are used to enable the service and perform basic configuration.

Setting	Default Value	Description
orchestration-service.enabled	true	Enables the service. NOTE: To use plans, this service must be enabled.
orchestration-service.host	127.0.0.1	IP address of the host of the service.
orchestration-service.port	42424	Port number on the host of the service.
orchestration-service.autoRestart	true	When true, the service automatically restarts if it crashes.
orchestration-service.classpath	%(topOfTree)s/services/orchestration-service/build/install/orchestration-service/orchestration-service.jar: %(topOfTree)s/services/orchestration-service/build/install/orchestration-service/lib/*	Modify the classpath only if necessary.

Plan Triggering

You can configure these parameters to set the retry time limits for the initial triggering events of a plan.

Setting	Default	Description
---------	---------	-------------

	Value	
orchestration-service.triggerCheckJobContinue.minTimeoutMillis	2000	Number of milliseconds of delay before starting the first retry.
orchestration-service.triggerCheckJobContinue.maxRetryTimeMillis	3000	Maximum time in milliseconds that the retried operation is permitted to run.

Sizing

These settings define the queue sizes and parallel execution of tasks for the orchestration service.

Setting	Default Value	Description
orchestration-service.taskExecutor.queueSize	15	Number of jobs that can be queued in in-memory queue for an execution thread to become available. This parameter defines the size of the queue which is used for holding tasks to be executed.
orchestration-service.taskExecutor.maxPoolSize	20	Maximum number of parallel threads executing jobs. Tip: This value should be less than or equal to the value for <code>orchestration-service.database.poolMaxSize</code> , which defines the maximum number of threads in the database pool.
orchestration-service.taskExecutor.corePoolSize	12	Number of threads that execute jobs. Tip: Set this value to more than 1 to activate parallel job execution.

Task Polling

These settings define polling intervals for the orchestration service.

Setting	Default Value	Description
orchestration-service.runTaskLockExtensionTimeInMillis	180000	Duration in milliseconds to lock the worker for long-running tasks.
orchestration-service.runTaskLockExtensionFrequencyInMillis	120000	Frequency in milliseconds for how often to extend the worker lock for long-running tasks.
orchestration-service.deployWorkflow.minTimeoutMillis	3000	Number of milliseconds before starting the first retry for a deploy action.
orchestration-service.deployWorkflow.maxRetryTimeMillis	30000	Maximum time in milliseconds that the retried operation is permitted to run for a deploy action.
orchestration-service.deleteWorkflow.minTimeoutMillis	3000	Number of milliseconds before starting the first retry for a delete action.
orchestration-service.deleteWorkflow.maxRetryTimeMillis	30000	Maximum time in milliseconds that the retried operation is permitted to run for a delete action.

Logging

Enable and specify aspects of logging for the orchestration service. For more information, see *Configure Logging for Services*.

Configure Secure Token Service

The Secure Token Service manages the use of secure tokens in the Designer Cloud powered by Trifacta platform for use with third-party systems. This service is used for:

- Authentication with Azure Key Vault. For more information, see *Configure Azure Key Vault*.
- OAuth2 authentication with third-party systems. For more information, see *Enable OAuth 2.0 Authentication*.

Configure

NOTE: Except in rare cases, the other properties for secure token service do not need to be modified.

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

Property	Description
"secure-token-service.enabled"	Set this value to <code>true</code> . It is enabled by default.
"secure-token-service.autorestart"	Set this value to <code>true</code> to enable auto-restarting of the secure token service.
"secure-token-service.port"	Set this value to <code>41921</code> .
"com.trifacta.services.secure_token_service.refresh_token_encryption_key"	Enter a base64 string to serve as your encryption key for the refresh token of the secure token service. A default encryption key is inserted for you. NOTE: If a valid base64 string value is not provided here, the platform fails to start. For more information, see <i>Create Encryption Key File</i> in the Configuration Guide.
"com.trifacta.services.secure_token_service.user_id_salt"	Enter a base64 string to serve as an encryption key for user IDs that are passed to the secure token service to return tokens. NOTE: This value must be specified with a non-empty string.
"secure-token-service.userIdHashingPepper"	Enter a base64 string.

3. Save your changes and restart the platform.

Logging

You can configure how logging is managed for the secure token service. For more information on configuring logging, see *Configure Logging for Services*.

Configure Logging for Services

Contents:

- Access
 - Support logs
- Configure
 - Mask PII
 - Configure log rotation
 - Configure log format for access logs
 - Log format for application logs
 - Configure output format
- Service Log Configuration
 - WebApp
 - Authorization Service
 - Batch Job Runner
 - Orchestration Service
 - Secure Token Service
 - Java VFS Service
 - VFS Service
 - Data Service
 - Connector Configuration Service
 - ML Service
 - JSData

As needed, you can modify via parameter the logging levels for the following services of the platform. These settings should only be modified when you are debugging an issue. After the issue is resolved, you should set the logging level back to its original value.

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

For more information on these services, see *System Services and Logs*.

For more information on logging levels, see <https://logging.apache.org/log4j/log4j-2.12.0/manual/customloglevels.html>.

Access

Support logs

For support use, the most meaningful logs and configuration files can be downloaded from the application. Select **Resources menu > Download logs**.

NOTE: If you are submitting an issue to *Alteryx Support*, please download these files through the application.

The admin version of this dialog enables downloading logs by timeframe, job ID, or session ID. For more information, see *Admin Download Logs Dialog*.

All Logs

Trifacta® administrators can access the logs through the Designer Cloud application . Use the following URL:

```
<hostname>:<port_number>/logs
```

For more information on the available logs, see *System Services and Logs*.

Configure

Mask PII

You can use the following settings to mask personal information in the log files. However, doing so may complicate debugging:

Logging Parameter and Default Value	Description
<pre>"logging.piiMask.enabled": false,</pre>	Set this value to <code>true</code> to enable masking of personally identifiable information (PII) in log files. In the Designer Cloud powered by Trifacta platform, this feature masks email addresses (userIds) with hashed values.
<pre>"logging.piiMask.salt": "this is a grain of salt",</pre>	When PII masking is enabled, this value can be used as a randomizing element for generating hashed values.

Configure log rotation

By default, log files for each service is automatically rotated for you.

As needed, you can enable and configure log rotation for additional log files. When enabled, the following logs are subject to rotation:

- `proxy_access.log`
- `proxy_error.log`
- `batch-job-runner.access.log`
- `data-service.access.log`
- `ml_service.access.log`
- `scheduling-service.access.log`
- `time-based-trigger-service.access.log`

NOTE: By default, these log files are rotated by `supervisord`. For more information on default values for `stdout_logfile_maxbytes`, `stdout_logfile_backups`, and `stderr` equivalents, see <http://supervisord.org/configuration.html#program-x-section-values>.

The following logs are excluded from log rotation and are not rotated when this feature is enabled:

- `join-inf.log`
- `join-sel.log`
- `protobuf-events.log`
- `webapp.analytics.log`
- `segment-proto.log`

To enable log rotation, you must copy the default configuration file into the appropriate working directory and set permissions:

```
sudo cp /opt/trifacta/conf/trifacta-logrotate.conf /etc/logrotate.d/trifacta-logrotate.conf
sudo chown root: /etc/logrotate.d/trifacta-logrotate.conf
sudo chmod 644 /etc/logrotate.d/trifacta-logrotate.conf
```

You can test the configuration using the following command:

```
sudo logrotate --debug /etc/logrotate.d/trifacta-logrotate.conf
```

Configure log format for access logs

The following are examples of access logs:

Service	Applicable Log Files	Parameter
VFS Service	vfs-service.access.log	vfs-service.loggerOptions.format
WebApp	webapp.access.log	webapp.loggerOptions.format

You can configure the fields and format of them.

NOTE: This field configuration applies only the log files listed above.

For each of the above parameters, you can use a token-based method of configuring the fields to include in a log entry in the listed logs. You can modify:

- The sequence of fields
- The fields to include
- For some fields, you can specify specific subsets of the available information.

Entries are specified as space-delimited fields, each of which begins with a colon:

```
:field1 :field2 :field3
```

For more information on these log format options, see <https://github.com/expressjs/morgan#tokens>.

Log format for application logs

Application log files have the following structure:

```
2019-07-09T11:46:54.667Z - warn: [EXPRESS] received extra query parameter(s): limit,offset,numPageLinks [url=/v4/connections/2/query] [method=GET] [sid=9133aa58-9cbb-4e0a-bfb4-2c59c44ffe2d] [rid=e4319f35-82a1-4c0b-a2d3-2e75d224178a]
```

The following fields are included in these files:

NOTE: These fields cannot be modified.

Field	Description
Timestamp	Date and timestamp in UTC when the event occurred

NOTE: Log values are timestamped in UTC time zone.

Level	Logging level for the event: <code>info</code> , <code>warning</code> , <code>error</code> , or <code>debug</code>
sid	The user's session identifier. Session Ids may be preserved across logins, but they may occasionally change if the user logs out of the application.
rid	The request identifier is unique to the specific request and is preserved across all services.
method	HTTP method that was used to invoke the request
url	URL of the endpoint that triggered the event. Hostname and port number are not included. NOTE: Query parameters are not included in the URL.

Configure output format

By default, logs are in flat text format and exported as `.log` files.

Some loggers enable output in JSON format.

NOTE: JSON format is not recommended for Designer Cloud Powered by Trifacta Enterprise Edition. You may not be able to download logs in JSON format.

NOTE: Log files that are configured for JSON output format cannot be included in the support bundle. See *Support Bundle Contents*.

To enable JSON format, locate the appropriate parameter, where the wildcard below represents the service. Set the value to `true`:

```
**.loggerOptions.json": true,
```

Save your changes and restart the platform or service.

More examples are listed below.

Service Log Configuration

WebApp

The WebApp manages loading of data from the supported connections into the front-end web interface.

NOTE: After changing these settings, the platform must be restarted.

Logging Parameter and Default Value	Description
<code>"webapp.loggerOptions.silent": false,</code>	When set to <code>true</code> , log messages are silent.
<code>"webapp.loggerOptions.level": "info",</code>	Supported levels (in decreasing order of verbosity): <code>silly</code> , <code>debug</code> , <code>info</code> , <code>warning</code> , <code>error</code>

<code>"webapp.loggerOptions.json": false,</code>	If set to <code>true</code> , logging output is in JSON format.
<code>"webapp.loggerOptions.format": ":method :url :status :res[content-length] :response-time :referrer :remote-addr :trifacta-user :user-agent",</code>	String containing list of fields to include in each log message.

NOTE: JSON format is not recommended for Designer Cloud Powered by Trifacta Enterprise Edition. You may not be able to download logs in JSON format.

Authorization Service

The authorization service manages access permissions to workspace objects.

Logging Parameter and Default Value	Description
<code>"authorization-service.systemProperties.logging.type": "regular",</code>	Set the type of output log: <ul style="list-style-type: none"> <code>regular</code> - regular files written to the Trifacta node <code>json</code> - JSON files should be used with cloud deployments only.
<code>"authorization-service.systemProperties.logging.level": "info",</code>	Supported levels (in decreasing order of verbosity): <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warning</code> , <code>error</code> , <code>fatal</code> , <code>off</code>
<code>"authorization-service.systemProperties.logging.fileName": "logs/authorization-service.log"</code>	Filename to which log files are written to the Trifacta node

Batch Job Runner

The Batch Job Runner service manages the execution of batch jobs on the backend running environment.

Logging Parameter and Default Value	Description
<code>"batch-job-runner.systemProperties.batch.rootLogLevel": "info",</code>	Supported levels (in decreasing order of verbosity): <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> , <code>fatal</code> , <code>off</code>

Orchestration Service

The orchestration service manages plan, snapshot, trigger, and task execution.

Tip: This service log captures issues related to plan execution. Please provide this log if you are experiencing issues with plan execution.

For more information, see *Overview of Operationalization*.

Logging Parameter and Default Value	Description
<code>"orchestration-service.systemProperties.logging.type": "regular",</code>	Set the type of output log: <ul style="list-style-type: none"> <code>regular</code> - regular files written to the Trifacta node

	<ul style="list-style-type: none"> • <code>json</code> - JSON files should be used with cloud deployments only.
<pre>"orchestration-service.systemProperties.logging.level": "info",</pre>	Supported levels (in decreasing order of verbosity): <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warning</code> , <code>error</code> , <code>fatal</code> , <code>off</code>
<pre>"orchestration-service.systemProperties.logging.fileName": "logs/orchestration-service.log"</pre>	Filename to which log files are written to the Trifacta node

Secure Token Service

The Secure Token Service manages authentication and returned tokens delivered from the authentication service.

Logging Parameter and Default Value	Description
<pre>"secure-token-service.systemProperties.log.level": "info",</pre>	Supported levels (in decreasing order of verbosity): <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warn</code> , <code>error</code> , <code>fatal</code> , <code>off</code>

See *Configure Secure Token Service*.

Java VFS Service

Loads data from various filesystems supported by the platform. For more information, see *Configure Java VFS Service*.

Logging Parameter and Default Value	Description
<pre>"java-vfs-service.systemProperties.logging.type": "regular",</pre>	Set the type of output log: <ul style="list-style-type: none"> • <code>regular</code> - regular files written to the Trifacta node • <code>json</code> - JSON files should be used with cloud deployments only.
<pre>"java-vfs-service.systemProperties.logging.level": "info",</pre>	Supported levels (in decreasing order of verbosity): <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warning</code> , <code>error</code> , <code>fatal</code> , <code>off</code>
<pre>"java-vfs-service.systemProperties.logging.fileName": "logs/java-vfs-service.log"</pre>	Filename to which log files are written to the Trifacta node
<pre>"java-vfs-service.systemProperties.access.log.path": "%(topOfTree)s/logs"</pre>	Path to the access log file for this service.
<pre>"java-vfs-service.systemProperties.log4j.configurationFile": "log4j2.xml"</pre>	Path to the Log4J2 configuration file on the Trifacta node. <div style="border: 1px solid black; padding: 5px; margin-top: 10px;"> <p>NOTE: Do not modify this path or file unless necessary.</p> </div>

VFS Service

Loads data from the various filesystems supported by the platform, both in the front-end user interface and in batch mode when the Trifacta Photon running environment or Trifacta Photon web client is enabled. Both are enabled by default. For more information, see *Configure VFS Service*.

Logging Parameter and Default Value	Description
<pre>"vfs-service.loggerOptions.silent": false,</pre>	When set to true , log messages are silent.
<pre>"vfs-service.loggerOptions.level": "info",</pre>	Supported levels (in decreasing order of verbosity): silly , debug , info , warning , error
<pre>"vfs-service.loggerOptions.json" : false,</pre>	If set to true , logging output is in JSON format. NOTE: JSON format is not recommended for Designer Cloud Powered by Trifacta Enterprise Edition. You may not be able to download logs in JSON format.
<pre>"vfs-service.loggerOptions.format": ":method :url :status :res[content-length] :response-time :referrer :remote-addr :trifacta-user :user-agent",</pre>	String containing list of fields to include in each log message.

Data Service

Service prepares queries against JDBC interfaces, using internal REST API calls.

Tip: Optionally, you can enable the logging of events from the underlying driver for each relational connection in `data-service.log`. For more information, see *Configure Connectivity*.

Logging Parameter and Default Value	Description
<pre>"data-service.systemProperties.logging.level": "info",</pre>	Supported levels (in decreasing order of verbosity): trace, debug , info , warn , error , fatal , off

Connector Configuration Service

Manages the metadata for connector types and their overrides. For more information, see *Configure Connector Configuration Service*.

Logging Parameter and Default Value	Description
<pre>"connector-configuration-service.systemProperties.logging.type": "regular",</pre>	Set the type of output log: <ul style="list-style-type: none"> regular - regular files written to the Trifacta node json - JSON files should be used with cloud deployments only.

<code>"connector-configuration-service.systemProperties.logging.level": "info",</code>	Supported levels (in decreasing order of verbosity): <code>trace</code> , <code>debug</code> , <code>info</code> , <code>warning</code> , <code>error</code> , <code>fatal</code> , <code>off</code>
<code>"connector-configuration-service.systemProperties.logging.fileName": "logs/java-vfs-service.log"</code>	Filename to which log files are written to the Trifacta node
<code>"connector-configuration-service.systemProperties.access.log.path": "%(topOfTree)s/logs"</code>	Path to the access log file for this service.
<code>"connector-configuration-service.systemProperties.log4j.configurationFile": "log4j2.xml"</code>	Path to the Log4J2 configuration file on the Trifacta node. <div style="border: 1px solid black; padding: 5px; text-align: center;">NOTE: Do not modify this path or file unless necessary.</div>

ML Service

For the ML service, logging level can be modified at the command line:

```
/services/ml-service/app.py --port <ml-service.port> --bindHost <ml-service.bindHost> --log-level <log_level>
```

Logging Parameter and Default Value	Description
<code>"ml-service.port": 5000,</code>	The <code>ml-service.port</code> setting in Admin Settings defines the port for the machine learning service. Default is 5000. For more information, see <i>System Ports</i> .
<code>"ml-service.bindHost": "127.0.0.1",</code>	The <code>ml-service.bindHost</code> setting in Admin Settings defines the host to which the ML service binds. Default is 127.0.0.1 (localhost).
<code>"ml-service.loggerOptions.json": false,</code>	The <code>ml-service.loggerOptions.json</code> setting in Admin Settings defines the output format. If set to <code>true</code> , logging output is in JSON format. <div style="border: 1px solid black; padding: 5px; text-align: center;">NOTE: JSON format is not recommended for Designer Cloud Powered by Trifacta Enterprise Edition. You may not be able to download logs in JSON format.</div>
Command Line parameter: <code>--log-level 'DEBUG'</code>	At the command line, you can use the <code>--log-level</code> parameter to specify the log level. Supported log levels: <code>DEBUG</code> , <code>INFO</code> , <code>WARNING</code> , <code>ERROR</code> , <code>CRITICAL</code>

JSData

The JSData logging options do not apply to a specific service. Instead, they are used by various services to log activities related to Wrangle and its interactions with various connections and running environments.

Logging Parameter and Default Value	Description
<pre>"jsdata.loggerOptions.silent": false,</pre>	When set to <code>true</code> , log messages are silent.
<pre>"jsdata.loggerOptions.level": "info",</pre>	Supported levels (in decreasing order of verbosity): <code>silly</code> , <code>debug</code> , <code>info</code> , <code>error</code> , <code>warn</code>
<pre>"jsdata.loggerOptions.json": false,</pre>	If set to <code>true</code> , logging output is in JSON format. <div data-bbox="456 541 1455 632" style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;"> <p>NOTE: JSON format is not recommended for Designer Cloud Powered by Trifacta Enterprise Edition. You may not be able to download logs in JSON format.</p> </div>

Configure Support Bundling

Contents:

- *Enable*
 - *Disable encryption*
 - *Disable support bundle UI*
 - *Configure Bundle Contents*
 - *Logging*
-

When a job succeeds or fails, you can optionally download the log files associated with the job. Optionally, this download bundle can include configuration files and service logs to assist in debugging job issues.

Tip: This feature is recommended. The optional support bundle includes files that *Alteryx Support* is likely to require during the support process. See below details related to security.

Security:

- Optionally, you can enable the application of encryption to sensitive data in the configuration files. See *Enable encryption* below.
- You can configure the specific files that you wish to include or exclude.

Enable

This feature is enabled by default.

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`:

```
"supportBundle.enabled": true,
```

3. Save your changes and restart the platform.

NOTE: Log files that are configured for JSON output format cannot be included in the support bundle. For more information on disabling JSON output in service logs, see *Configure Logging for Services*.

Disable encryption

By default:

- For end users, encryption is enabled.
- For admin users, encryption is always disabled.

When encryption is enabled:

- Administrators still download a clear-text version of the support bundle.

- End-users can download an encrypted version of the support bundle. This encrypted version can be delivered to *Alteryx Support* to assist in debugging of issues.

As needed, you can insert the public key to apply to the encryption.

NOTE: If you do not have the Trifacta public key, please contact *Alteryx Support*.

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. Set the following parameters:

Parameter	Description
<code>supportBundle.encryption.enabled</code>	Set to <code>false</code> to disable encryption of downloaded files for end-users.
<code>supportBundle.encryption.publicKey</code>	Click Edit to paste in the public key value to use for encryption.

3. Save your changes and restart the platform.

Disable support bundle UI

By default, Designer Cloud Powered by Trifacta Enterprise Edition includes two dialogs for downloading logs. To access, select **Resources menu > Download logs**.

- For more information on the admin version, see *Admin Download Logs Dialog*.
- For more information on the user version, see *Download Logs Dialog*.

NOTE: Data in the logs downloaded through the admin version is always unencrypted. Data downloaded through the user version is encrypted but can be unencrypted (see below).

As needed, you can disable one or both of these dialogs:

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 one or both of them to `false`:

```
"supportBundle.endUserUI.enabled": true,
"supportBundle.adminUI.enabled": true,
```

3. Save your changes and restart the platform.

Configure Bundle Contents

You can configure the contents of the support bundle using the following parameters.

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. Set the following sizing parameters:

Parameter	Description
supportBundle.files.maxNumberOfArchiveFiles	Specifies the maximum number of archive files of the primary platform configuration file <code>trifacta-conf.json</code> to include. Whenever this configuration file is saved, a new archive version is created.
supportBundle.files.maxFileSizeInBytes	Limits the maximum size in bytes of each file in the bundle. Default is 1000000 (about 1MB). NOTE: Files that exceed this maximum size are truncated at the limit. For log files, older entries are lost.

- Review and configure the files to include in the bundle. For more information on these files, see *Support Bundle Contents*.

Parameter	Description
supportBundle.files.staticFiles	Comma-separated list of paths to the static configuration files to include. Paths are relative to <code>topOfTree</code> .
supportBundle.files.serviceLogs	Comma-separated list of service log files to include. Paths are relative to the following directory: <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <code>/opt/trifacta/logs</code> </div> For more information on these files, see <i>Configure Logging for Services</i> .
supportBundle.files.confFiles	Comma-separated list of configuration files to include in the bundle. Paths are relative to the following directory: <div style="border: 1px solid #ccc; padding: 5px; margin: 5px 0;"> <code>/opt/trifacta/conf</code> </div> Some of these files pertain to the web server that serves the Designer Cloud application . If the Designer Cloud powered by Trifacta platform is connected to a Hadoop-based running environment, Hadoop configuration files can be included as well. See <i>Support Bundle Contents</i> .

- Save your changes and restart the platform.

Logging

- Unable to download logs:** See `webapp.log`.
- Problems generating the support bundle:** If there are issues generating a support bundle, the ZIP file contains an `errors.txt` file to provide additional information.

Configure for Redis

Contents:

- *Prerequisites*
 - *Install Redis server*
 - *Configure Designer Cloud powered by Trifacta platform*
 - *Enable*
 - *Configure for TLS*
 - *Configure for high availability*
-

Redis is an in-memory datastore, which can be used as a high-performance database, cache, and message broker. Redis is available for on-premises and cloud deployments. For more information, see <https://redis.io/>.

Uses:

The Designer Cloud powered by Trifacta platform can be configured to use an accessible Redis server in your enterprise infrastructure. Uses include the following:

- sending emails and webhook requests
- caching authorization service requests
- Managing concurrency issues and file locking while running conversion jobs

Prerequisites

- Redis 6.2 or higher
- Redis server must be accessible to the Trifacta node.

Install Redis server

Please install Redis server on an accessible node in your enterprise infrastructure. For more information, see <https://redis.io/download>.

Configure Designer Cloud powered by Trifacta platform

Enable

Please complete the following steps to enable use of Redis in the Designer Cloud powered by Trifacta platform .

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 settings and set them to `true`:

Setting	Description
"redis.enabled"	Set to <code>true</code> to enable use of the specified Redis server as a distributed queue.
"feature.enableRedisLocking"	Set to <code>true</code> to enable use of the specified Redis server for file locking during job execution.

3. Configure the host and port number of the Redis server:

Setting	Description
---------	-------------

"redis.host"	Hostname of the Redis server. If the server is hosted on the Trifacta node, enter the following: <div style="border: 1px solid #ccc; padding: 5px; width: fit-content; margin: 5px auto;">127.0.0.1</div>
"redis.port"	Port number for the Redis server. Default value is 6379.
"redis.password"	Password to use for the Designer Cloud powered by Trifacta platform to access the Redis server.

4. Save your changes and restart the platform.

Configure for TLS

You can configure the Designer Cloud powered by Trifacta platform to use TLS (SSL) when communicating with the Redis server.

Prerequisites:

- You must install an SSL certificate on the Trifacta node. For more information, see *Enable SSL for Databases*.

NOTE: If you have already configured SSL for use with the Trifacta databases, then you have already installed the SSL certificate and configured its use. In this case, you only need to set the enable flag below for use of SSL with the Redis flag and provide any additional NodeJS options. The other settings are shared between these two features.

Steps:

- You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*.
- To enable TLS for Redis, locate the following setting and set it to `true`:

```
"redis.tls.enabled": true,
```

- If you have not configured TLS for the Trifacta databases, please complete the following configuration:

Setting	Description
"redis.tls.serverCertificateAuthorityFile"	(optional) Path on the Trifacta node to the certificate authority verification file, which is used to verify the presented server certificate.
"redis.tls.clientKeyFile"	(optional) Path on the Trifacta node to the client key file, which is used for client authentication.
"redis.tls.clientCertificateFile"	(optional) Path on the Trifacta node to the SSL certificate to use for client authentication.
"redis.tls.additionalOptions"	JSON string of additional TLS properties. Properties should be the same ones accepted by NodeJS's <code>tls.connect()</code> method. For more information, see https://nodejs.org/api/tls.html#tls_connect_options_callback .

4. Save your changes and restart the platform.

Configure for high availability

If you have deployed the Designer Cloud powered by Trifacta platform in a high availability environment, you can configure the platform to work with Redis Sentinel to manage master and failover Redis instances. Please complete the following configuration steps.

NOTE: These settings must be applied on each instance of the Trifacta node.

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. If you have not done so already, enable the Redis parameters.

NOTE: When Redis sentinel is enabled for use by the Designer Cloud powered by Trifacta platform, the `host` and `port` options for Redis are ignored. Please enable the service and complete the following configuration.

3. Configure the following to use Sentinel:

Setting	Description
<code>redis.useSentinel</code>	Set this value to <code>true</code> to enable Redis Sentinel.
<code>redis.SentinelOpts.sentinel</code> s	Enter a comma-separated list of available sentinels to which the Trifacta node can connect.
<code>redis.sentinelOpts.name</code>	Enter a comma-separated list of Redis hosts to which you can connection.

4. Save your changes and restart the platform.

Miscellaneous Configuration

Contents:

- *Configure File Format Support*
 - *Configure CSV field delimiters*
 - *Supported Filename Extensions*
 - *Infer compression scheme*
 - *Enabling Features*
 - *Other Configuration Topics*
-

This section contains miscellaneous configuration topics to enable minor features of the platform.

Limits

For more information, see *Configure Application Limits*.

Configure File Format Support

Configure CSV field delimiters

When you publish a CSV file, by default the fields in the file are comma-separated. Optionally, you can configure a different field delimiter.

Tip: During publication, you can specify whether the output file includes double quote marks around each field. For more information, see *Run Job Page*.

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. To enable the feature, verify that the following parameter has been set to `true`:

```
"feature.publishDelimiterQuoteOption.enabled": true,
```

3. If you wish to specify the delimiter character using a Unicode value, please set the following to `true`:

```
"feature.publishDelimiterQuoteOption.enableUnicodeCSVDelimiters": true,
```

4. Search for the following parameter:

```
"webapp.outputCsvDelimiter"
```

5. In the textbox, enter the string that is used as the delimiter.
 - a. Strings can be more than one character in length.
 - b. If you have enabled the Unicode CSV delimiters feature, you can enter Unicode characters in the following format:

```
\uXXXX
```

where `xxxx` is the Unicode value for the character.

NOTE: The delimiter value is used as the default value for each job. The default can be overridden on a per-job basis. For more information, see *Run Job Page*.

6. Save your changes.

Supported Filename Extensions

You cannot upload a file with an unsupported or unlisted file extension. As needed, you can add extensions to the list of supported file formats, which enables the file to be uploaded.

- Extensions are case-sensitive. `.xml` and `.XML` must both be listed.
- A leading period is required.

NOTE: Even if the file extension is added to this list, the platform may not be able to process the file. For more information, see *Supported File Formats*.

NOTE: The Designer Cloud powered by Trifacta platform prevents the uploading of files with extensions that are disabled through specific configuration flags. For example, if you disable BZIP2 or Avro files, you cannot re-enable them by adding their extensions to this configuration list.

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 add file extensions. In the example, two new extensions have been added to the list:

```
"webapp.client.allowedFileExtensions": [".log", ".LOG"],
```

Tip: To allow the uploading of files that do not have a file extension, enter a value of `"` (empty string) in the array. Files without extensions are treated as TXT files by the Designer Cloud powered by Trifacta platform .

3. Save your changes and restart the platform.

Infer compression scheme

For imported files, the Designer Cloud application infers compression based on the filename extension. For example, all files that have the `.gz` extension at the end of the filename are decompressed as GZIP files. This is the default behavior.

As needed, the Designer Cloud application can be configured to infer any compression that has been applied to the file based on reading in the few bytes of the file. Based on that data signature, the file is passed to the appropriate internal service for decompression.

Tip: This method is preferred if it's possible for filename extension to be improperly specified for the compression scheme.

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`:

```
"eature.inferCompressionFromFileSignature.enabled": false,
```

3. Save your changes and restart the platform.

Enabling Features

For more information on the feature flags to enable or disable features, see *Configure Features*.

Other Configuration Topics

Configure Application Limits

Contents:

- *Operating System Limits*
 - *Raise ulimit setting*
- *Browser Limits*
 - *Change body limits*
 - *Change maximum number of rows displayed in browser per join key*
 - *Change page preview limit*
- *Ingestion*
 - *Maximum record length*
- *Timeouts*
 - *Change application timeout limits*
 - *Session timeout*
 - *Timeout for suggestion card suggestions*
- *Jobs*
 - *Maximum number of flow jobs launched in parallel*
 - *Job status polling interval*
- *Sampling*
 - *Size of stored samples*
 - *Sample size load limit*
 - *Photon random sample load limit*
- *Wrangling Limits*
 - *Maximum split limits*
- *Relational limits*
- *Miscellaneous limits*
 - *Date range limit*

This section provides information on various settings that you can specify to apply minimum and maximum limits on the Designer Cloud® application .

Operating System Limits

Raise ulimit setting

To perform normal operations, the Designer Cloud powered by Trifacta platform may need to maintain a high number of simultaneously open files, the count of which may exceed the default setting for the operating system (the ulimit).

NOTE: If the Designer Cloud powered by Trifacta platform hits the ulimit and is unable to open additional files, jobs may fail, or the platform may be unable to access content. The log may contain something similar to the following error: `Failed on local exception: java.net.SocketException: Too many open files.`

By default, the operating system sets the limit on the number of open files at 1024. Please complete the following steps to raise this limit.

Tip: The ulimit should be raised to 64000 depending on the quality of your hardware.

Steps:

1. If it is running, stop the Designer Cloud powered by Trifacta platform . See *Start and Stop the Platform*.
2. Verify the current ulimit:

```
ulimit -Hn
```

3. Edit the following file: `/etc/security/limits.conf`.
4. At the bottom of the file, add the following entry, which overrides the defined limit with a value of 16000:

```
* hard nofile 16000
```

5. Please add the following line after the previous one if this error is encountered: `"java.lang.OutOfMemoryError: unable to create new native thread"`. This exception means the ulimit for processes must be increased, too:

```
* hard nproc 16000
```

6. Save the file and restart the platform. See *Start and Stop the Platform*.

Browser Limits

Change body limits

If you are encountering log message where the request submitted from the client is too large, you can try to raise the limit on the size of body objects submitted from the client.

NOTE: Raising these values too high can overload the browser.

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

Setting	Description
<pre>"webapp.bodyParser.urlencoded.limit": "10mb",</pre>	Maximum permitted size of the URL-encoded body of a request submitted from the client. Size is in MB.
<pre>"webapp.bodyParser.json.limit": "10mb",</pre>	Maximum permitted size of a JSON object submitted from the client. Size is in MB.

Change maximum number of rows displayed in browser per join key

For each matching join key value, the Designer Cloud application displays a maximum of three rows in the browser for the current sample. So, when you join a dataset with repeating key values, you may see a fewer number of rows of data than you would expect.

NOTE: This issue is limited only to the sampled data that is displayed in the browser. When you run a job across the entire dataset, the proper number of rows are generated in the output.

For some users, this simplification may be confusing. As needed, you can use the following steps to change the maximum number of rows displayed in the browser for each join key.

Steps:

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

1. Search for and modify the following parameter:

```
"webapp.client.sampleOutputTuplesPerJoinKey": 3,
```

2. Save your changes and restart the platform.

Change page preview limit

In the Flow and Dataset pages, you can preview the data in datasets that you have imported or are importing. For example, when you click the Eye icon next to a dataset's name, you can see a preview of the data in the dataset, which is useful for ensuring that you have the correct data.

Depending on the size of the datasets, you may wish to increase the limit on the size of preview data. If you are working with wide datasets, you may need to increase the limit so that you can get a solid preview of the contents.

NOTE: Increasing this preview size may have performance impacts, particularly on lower-quality desktops. You should make adjustments with caution.

Steps:

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

1. Locate the following setting, which defines the number of bytes that are loaded by default in a preview. Maximum permitted value is 1024000 (1 MB).

```
"webapp.client.previewLoadLimit": 128000,
```

2. Save your changes and restart the platform.
3. After the platform has restarted, you should preview a large dataset to verify that performance is acceptable.

Ingestion

Maximum record length

By default, the maximum length for an individual record is 20 MB. After rows have been split, individual records can be up to this limit in length.

As needed, you can modify this limit.

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, which reflects the maximum record length in bytes:

```
"webapp.maxRecordLength": 209715200,
```

3. Modify as needed.

NOTE: Be careful when you raise this value, which can cause out of memory conditions, empty data grids, and browser crashes. You should raise the value incrementally.

4. Save your changes and restart the platform.

Timeouts

Change application timeout limits

The front-end application respects the following timeout settings for queries issued to back-end datastores, including the Trifacta database.

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

Settings	Description
<code>webapp.timeoutMilliseconds</code>	Overall timeout limit in milliseconds for the front-end application. Default value is 120000 (2 minutes).
<code>jsdata.remoteTransformTimeoutMilliseconds</code>	Timeout limit in milliseconds for the Transformer Page. This setting is an override of the previous one. Default value is 180000 (3 minutes).

You can change the timeout settings if you are experiencing timeouts or other errors because of long-running queries to external data connections.

NOTE: In most environments, these settings should not be changed. Lowering them can cause reasonable queries to fail, and raising them too high can cause performance issues. Please adjust them only if you are experiencing very long query times to external sources, especially for database views.

Steps:

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

1. Locate the following configuration. Specify new timeout values in milliseconds:

```
"webapp.timeoutMilliseconds": 120000,
```

```
"jsdata.remoteTransformTimeoutMilliseconds": 180000,
```

2. Save your changes and restart the platform.

Session timeout

By default, the maximum session duration is set to be one month. If needed, you can change the maximum session duration, as well as other session parameter values.

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. Modify the following parameters, as needed:

Parameter	Description	Default
<code>webapp.session.refreshEmbeddedExpirationDateAfterMinutes</code>	Refresh interval in minutes for the expiration date embedded in the session cookie	5
<code>webapp.session.cookieSecureFlag</code>	Set a secure cookie in the client application.	false

3. You apply this change through the *Workspace Settings Page*. For more information, see *Platform Configuration Methods*.

Setting	Description	Default
Session duration	Maximum session duration in minutes	10080 (one week)

4. Save your changes and restart the platform.

Timeout for suggestion card suggestions

By default, the platform waits a specified length of time for the machine learning service to return suggestion cards. When more time is enabled, the service may be able to discover better suggestions based on the currently selected data.

If needed, you can change the delay limit from its default value of 80 milliseconds.

Steps:

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

1. Locate the following setting and change its value:

```
"feature.mlTransformSuggestions.delayThreshold": 80,
```

2. Save your changes and restart the platform.

Jobs

Maximum number of flow jobs launched in parallel

By default, the Designer Cloud powered by Trifacta platform permits up to four jobs from the same flow to be launched in parallel for execution. If there are more flow job launches than this limit, the additional jobs are queued for execution after one or more of the launched jobs has completed.

Tip: This limit is most relevant when you are running a scheduled job, which can execute all jobs in a flow at the same time.

Max parallel jobs setting	Description
4	(Default) Up to four jobs from the same flow can be launched and in the process of execution at the same time. <ul style="list-style-type: none"> • Additional jobs are queued for execution.

1	Jobs from the flow are executed sequentially.
0	No limit. All jobs from a flow can be executed at the same time.

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. Modify it according to your needs:

```
"webapp.jobLaunchingBatchSize": 4,
```

3. Save your changes and restart the platform.

Job status polling interval

Periodically, the application polls the running environment to check the status of jobs in transit. This polling occurs in the following areas of the application:

- Job History page - Checks to see if running jobs have been resolved.
- Flow View page - Checks to see if running jobs have been resolved.
- Transformer page - Checks to see if sampling jobs have been resolved.

NOTE: This setting does not apply to the initial sample which is derived from the first N rows of the dataset.

As needed, you can modify the interval at which the application polls for job status from these area. The default value is 5000 milliseconds (5 seconds).

NOTE: If this setting is lowered too much, polling requests can overlap, resulting in no updates to the application. Application performance can be impeded.

Steps:

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

1. Locate the following setting and change its value:

```
"webapp.polling.jobStatusInMillis" : 5000,
```

2. Save your changes and restart the platform.

Sampling

The following configuration settings define the size of samples stored in the base storage layer and transmitted to the user's web browser for display through the Transformer page.

Size of stored samples

By default, samples are generated and stored in the base storage layer up to 40 MB in size.

NOTE: This size is applied to all user-generated samples. Modifications to this size can significantly change the volume of data stored in the backend.

NOTE: If the datasource is compressed or must be converted during ingestion, the stored size of the sample on the base storage layer can exceed this limit.

Tip: This size should be modified in conjunction with any changes to the maximum size of transferred samples, which is described in the following section.

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

Setting	Default value	Description
<code>webapp.sampleOutputLimit</code>	41943040	<p>Sets the requested size of samples in bytes that are stored in the base storage layer for each sample.</p> <p>NOTE: This parameter defines the storage size of samples on the backend. Default storage size is four times larger than in previous releases.</p> <p>NOTE: For datasources that must be decompressed or converted during ingest, the actual storage volume may be larger than this limit.</p>

Sample size load limit

By default, samples that are transferred to the client in the web browser for users are 10 MB in maximum size. If desired, users can increase or decrease this sample size on a per-recipe basis.

As needed, you can configure the following:

- setting the default size of samples displayed in the browser (default is 10 MB)
- setting the maximum size of samples displayed in browser (default is 40 MB)
 - users can override the actual size of the sample downloaded to their browser based on their own experience

Notes:

Increasing the sample size may degrade the user experience in the Transformer page in the following ways:

- Generation of column details and data grid histograms
- Preview card loading time
- Time required to complete brushing and linking in histograms

NOTE: If you increase the sample size above the default setting and encounter unacceptable performance in the above areas, you should reduce the sample size settings.

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

Setting	Default value	Description and Notes
<code>webapp.client.</code>	104857	Sets the default maximum number of bytes that can be loaded into the browser for samples.

defaultLoadLimit	60	NOTE: On a per-recipe basis, users can override this setting through the Transformer page. See <i>Change Recipe Sample Size</i> .
webapp.client.maxLoadLimit	41943040	Sets the maximum number of bytes that can be loaded into the browser for samples. NOTE: This value cannot be overridden by users. Users can set the sample size in their browser up to this limit and no higher.

Photon random sample load limit

Unless it is not available for some reason, Trifacta Photon is used to generate random samples. By default, the Trifacta Photon running environment loads a maximum of 1 GB (1024 MB) of data from the imported dataset for generating a new random sample. This data comes from the top of the file, meaning that rows that are deeper than 1 GB in the source data cannot be included in any generated random sample.

From this selection of data, a sample of the data is derived for display in the data grid. As needed, you can configure the random sample limit to include a larger or smaller volume of maximum data.

NOTE: Be careful making adjustments to this setting. If the volume of data is too large, you can crash the running environment.

Steps:

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

1. Locate the following setting, which is listed in terms of bytes. The default value listed below corresponds to 1 GB of data:

```
"webapp.sampleLoadLimit": 1073741824,
```

2. Save your changes and restart the platform.

Wrangling Limits

Maximum split limits

By default, a single split operation can break up a single column into 250 separate columns. As needed, you can change this maximum value.

NOTE: Increasing this limit consumes more resources and may overload the Designer Cloud application or your browser. Adjust with caution.

Tip: This limit also to extraction of keys and values from Objects and Arrays.

Steps:

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

1. Locate the following setting, which represents the maximum number of columns that can be generated by one of the applicable steps:

```
"feature.delimSplitColumnLimit": 250,
```

2. Save your changes and restart the platform.

Relational limits

See *Configure Security for Relational Connections*.

Miscellaneous limits

Date range limit

By default, the Designer Cloud powered by Trifacta platform supports the following date range for Datetime data type validation:

```
January 1, 1400 - December 31, 2599
```

This date range is validated against the following default regular expression:

```
((?:1[4-9]|2[0-5])\d{2})
```

As needed, you can change the above regular expression to define your preferred date range for the Datetime data type. Your regular expression must be in the following format:

```
(<your_regular_expression>)
```

For example, the following regular expression allows dates up to December 31, 9999:

```
((?:1[4-9]|[0-9][0-9])\d{2})
```

NOTE: Use of Trifacta patterns in this field is not supported. The entry must be a valid regular expression.

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:

```
webapp.yearFourDigitRegex
```

3. Insert your regular expression in the required format.
4. Save your changes and restart the platform.
5. You should check your new Datetime date range validation against some sample data.

Configure Global File Encoding Type

Contents:

- [Supported File Encoding Types](#)
- [Configure Global File Encoding Type](#)
- [Validate](#)
- [Update Sources](#)

The Designer Cloud powered by Trifacta® platform supports a single global file encoding type, which is set to UTF-8 by default. This file encoding type applies to all text files for the following operations:

- Loading the default sample and any subsequent random samples
- Running text-based jobs

NOTE: This setting applies only to text files. Binary types, such as Avro, are not affected by the global file encoding type.

NOTE: If you change this setting, datasets that were imported under the former encoding type are no longer valid. Instructions are provided below for updating them.

Supported File Encoding Types

For more information, see [Supported File Encoding Types](#).

Configure Global File Encoding Type

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

1. Set the following parameter to the appropriate file encoding type:

```
"inputFileEncoding": "UTF-8",
```

2. Save your changes and restart the platform.

NOTE: After you change the global encoding type, datasets that were imported under the old encoding type must be reloaded to the platform. For more information, see [Update Sources](#).

Validate

After you have changed the global file encoding type, restart services. See [Start and Stop the Platform](#).

You should try to create a dataset from source data of the selected encoding type.

Update Sources

After you have changed the global encoding type, datasets that were imported under the former encoding type are no longer valid.

Steps:

1. For each dataset imported under the old encoding type, upload a new version.
2. For each recipe that used the old version of the imported dataset:
 - a. Edit the recipe in the Transformer Page.
 - b. Swap the source from the old version to the new one. For more information, see *Flow View Page*.
3. Repeat for each imported and recipe combination.

Enable User Analytics

Contents:

- *Configuration Steps*
- *Customer Requests*
- *Configure for Platform Analytics*
- *Configure for Segment Analytics*
- *Configure for Amplitude Analytics client-side SDK*
- *Open user logging port*
- *Create credentials file*
- *Generate cron job*
- *Disable*

This section describes how to enable or disable logging of user activities and transfer of the logs to Alteryx®. When this feature is enabled, user activities are captured locally on the Trifacta node in a series of log files. Periodically, these log files are uploaded to a predefined S3 bucket, where Alteryx can analyze the logging activity to improve the product and assist in troubleshooting.

Tip: This feature is useful for providing better suggestions and machine-based learning to the Designer Cloud powered by Trifacta platform instance.

NOTE: During initial deployment, this service may be enabled for you. You can use the information below to disable the service.

Alteryx captures the following types of usage information, which are available in different releases.

- These can be separately enabled.
- For more information on the data that is captured, see <https://community.trifacta.com/s/article/Trifacta-Usage-Data-Collection-1515802070895>.

Analytics Type	Description
Trifacta Analytics	Alteryx proprietary capture of information about the Designer Cloud powered by Trifacta platform
Segment Analytics	Analytics for various common data segments, such as Google and Marketo.

Configuration Steps

The following configuration steps must be completed:

1. Customer must file a request with *Alteryx Support* when this service is to be enabled for the first time.
2. Services must be enabled on the Trifacta node.
3. Open user logging port, if not already opened.
4. Generate and publish credentials.
5. Define cron job to upload logs.

Customer Requests

To enable this service, customers must file a support ticket with *Alteryx Support*. In your request, please include a request for the appropriate API write key values to insert in the configuration. Details are below.

Configure for Platform Analytics

The platform's custom-built telemetry system is controlled by the following config field in You can apply this change through the *Admin Settings Page* (recommended) or `trifacta-conf.json`. For more information, see *Platform Configuration Methods*:

Property	Description
<code>"webapp.client.enableUserEventLogging"</code>	When set to <code>true</code> , enable logging of user events via client-side telemetry.

Configure for Segment Analytics

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

The following settings apply to segment analytics.

Property	Description
<code>"webapp.enableAnalytics": false,</code>	When <code>true</code> , segment analytics are globally enabled. When <code>false</code> , no data is recorded for any segment or forwarded to any channel.
<code>"webapp.analytics.segmentWriteKey": "<YOUR_VALUE_HERE>",</code>	For remote analytics, this property identifies the segment API writeKey matching the project to which to push. Within the Segment project lives the configuration for each sink (e.g Google Analytics, Marketo). NOTE: For more information on the key value to insert, please contact <i>Alteryx Support</i> .
<code>"webapp.analytics.enabledChannels": ["Log", "Amplitude"],</code>	The channels for which to record data: <code>Log Google Analytics Marketo Amplitude <future sink></code> . <code>Log</code> is the default channel. For remote analytics, this value and the <code>segmentWriteKey</code> are independent, enabling two points of control. For example, if you wanted Marketo to receive analytics, you'd need to include it in <code>enabledChannels</code> and also hook up that integration for your project in Segment.

1. Make changes to the above properties as needed.
2. Save your changes and restart the platform.

Configure for Amplitude Analytics client-side SDK

Property	Description
----------	-------------

<pre>"telemetry.amplitude.client.enabled"</pre>	<p>When set to <code>true</code>, enable logging of user events via Amplitude's client-side SDK.</p> <p>NOTE: This property is not intended for customer modification. If you believe that you need to make changes, please contact <i>Alteryx Support</i>.</p>
<pre>"telemetry.amplitude.client.writeKey"</pre>	<p>NOTE: For more information on the key value to insert, please contact <i>Alteryx Support</i>.</p>

Open user logging port

NOTE: To receive the full benefits of this feature, the Trifacta node must be able to connect to the public Internet.

On the server hosting the Designer Cloud powered by Trifacta platform , the following port must be opened:

- Port 80 (HTTP) and/or
- Port 443 (HTTPS)

Create credentials file

To connect to S3, the Designer Cloud powered by Trifacta platform requires that a set of credentials be generated and stored in the following directory. This credentials is provided by Alteryx.

```
/opt/trifacta/bin/log-forwarding
```

Generate cron job

To regularly upload the generated logs to Alteryx, you can configure a cron job to transfer the files.

Steps:

1. Create an agent or script to periodically run the node process `log-forwarding.js`. You should run this once per day.
2. An example command to run this script from the deployment directory is the following:

```
node bin/log-forwarding/src/log-forwarding.js protobuf-events.log segment-proto.log cleaned-join-logs.txt
```

Disable

To disable the service, set `webapp.enableAnalytics` to `false`. Then, restart the platform.

Tune Application Performance

Contents:

- *Application Performance*
 - *Other applicable parameters*
- *Limit Application Memory Utilization*
- *Trifacta Photon Running Environment Performance*

This section provides general guidelines for tuning the configuration parameters of the Trifacta® node for varying loads.

NOTE: These guidelines are estimates of what should provide satisfactory performance. You should review particulars of the variables listed below in detail prior to making recommendations or purchasing decisions.

For more information on tuning the performance of the connected cluster, see *Tune Cluster Performance*.

Application Performance

Some Designer Cloud powered by Trifacta platform services running on the Trifacta node can use multiple processes to serve more requests in parallel (e.g., `webapp` and `vfs-service`). By increasing the number of processes, these services are able to serve more requests in parallel and improve the application's response time under load.

Other services, such as `batch-job-runner`, `data-service`, and `scheduling-service` use multiple threads within a single process as needed to serve concurrent requests. Each service may have tuning parameters that can be adjusted according to load.

Tip: Unless specific recommendations apply below, you should use the default configuration.

For an expected number of peak concurrent (active) users, P , set the following parameters.

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

Parameter	Default Value	Recommended Value	Example
<code>webapp.numProcesses</code>	2	$P / 15$, rounded up to the nearest integer, minimum of 2	for $P = 40$, set to 3
<code>webapp.database.pool.maxConnections</code>	10	$5 * \text{webapp.numProcesses}$	for <code>webapp.numProcesses = 3</code> , set to 15
<code>vfs-service.numProcesses</code>	2	$P / 50$, rounded up to the nearest integer, minimum of 2	for $P = 225$, set to 5

Other applicable parameters

The following configuration parameters also affect application performance.

NOTE: Avoid modifying these parameters unless instructed by *Alteryx Support*.

Parameter	Default Value
<code>batch-job-runner.systemProperties.httpMaxConnectionsPerDestination</code>	50

Limit Application Memory Utilization

Several Trifacta node services allow limitations on memory utilization by varying their JVM configuration's `-Xmx` value. These can be limited by modifying the following parameters:

Parameter	Default Configuration
<code>batch-job-runner.jvmOptions</code>	<code>-Xmx1024m</code>
<code>data-service.jvmOptions</code>	<code>-Xmx128m</code>
<code>spark-job-service.jvmOptions</code>	<code>-Xmx128m</code>

Other services have low memory requirements.

Trifacta Photon Running Environment Performance

Jobs run on the Trifacta node and "Quick Scan" samples are executed by the Trifacta Photon running environment embedded on the Trifacta node, running alongside the application itself. Two main parameters can be used to tune concurrency of job execution and throughput of individual jobs:

Parameter	Description	Default
<code>batchserver.workers.photon.max</code>	Maximum number of simultaneous Trifacta Photon processes; once exceeded, jobs are queued	2
<code>photon.numThreads</code>	Number of threads used by each Trifacta Photon process	4

Increasing the number of concurrent processes allows more users' jobs to execute concurrently. However, it also leads to resource contention among the jobs and the application services.

Trifacta Photon execution is purely in memory. It does not spill to disk when the total data size exceeds available memory. As a result, you should configure limits on memory utilization by Trifacta Photon. If a job exceeds the configured memory threshold, it is killed by a parent process tasked with monitoring the job.

Parameter	Description	Default
<code>batchserver.workers.photon.memoryMonitorEnabled</code>	Set <code>true</code> to enable the monitor, and set to <code>false</code> (limited only by operating system) otherwise	<code>true</code>
<code>batchserver.workers.photon.memoryPercentageThreshold</code>	Percentage of available system memory that each Photon process can use (if <code>photon.memoryMonitorEnabled</code> is <code>true</code>)	50

A reasonable rule of thumb: the input data size should not exceed one tenth of the job's memory limit. This rule of thumb accounts for joins and pivots and other operations that can increase memory usage over the data size. However, this parameter is intended as a safeguard; it is unlikely that all running jobs would approach the memory limit simultaneously. So you should "oversubscribe" and use slightly more than $(100 / \text{batchserver.workers.photon.max})$ for this threshold.

In addition to a memory threshold, execution time of any Photon job can also be limited via the following parameters:

Parameter	Description	Default
<code>batchserver.workers.photon.timeoutEnabled</code>	Set <code>true</code> to enable the timeout, and set to <code>false</code> (unlimited) otherwise	<code>true</code>
<code>batchserver.workers.photon.timeoutMinutes</code>	Time in minutes after which to kill the Photon process (if <code>batchserver.workers.photon.timeoutEnabled</code> is <code>true</code>)	180

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



Copyright © 2022 - Trifacta, Inc.
All rights reserved.