TRIFACTA

Install Guide for Azure

Version: 6.4.1
Doc Build Date: 08/30/2019

Install Overview

Contents:

Basic Install Workflow

1. Review the pre-installation checklist and other system requirements. See *Install Preparation*.
2. Review the requirements for your specific installation scenario in the following sections.
3. Install the software. See *Install Software*.
4. Install the databases. See *Install Databases*.
5. Configure your installation.
6. Verify operations.

Notation

In this guide, JSON settings are provided in dot notation. For example, `webapp.selfRegistration` refers to a JSON block `selfRegistration` under `webapp`:

```
{
...
  "webapp": {
    "selfRegistration": true,
    ...
  }
...
}
```

Install for High Availability

Contents:

The Trifacta® platform can be installed across multiple nodes for high availability failover. This section describes the general process for installing the platform across multiple, highly available nodes.

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

Limitations

The following limitations apply to this feature:

- This form of high availability is not supported for Marketplace installations.

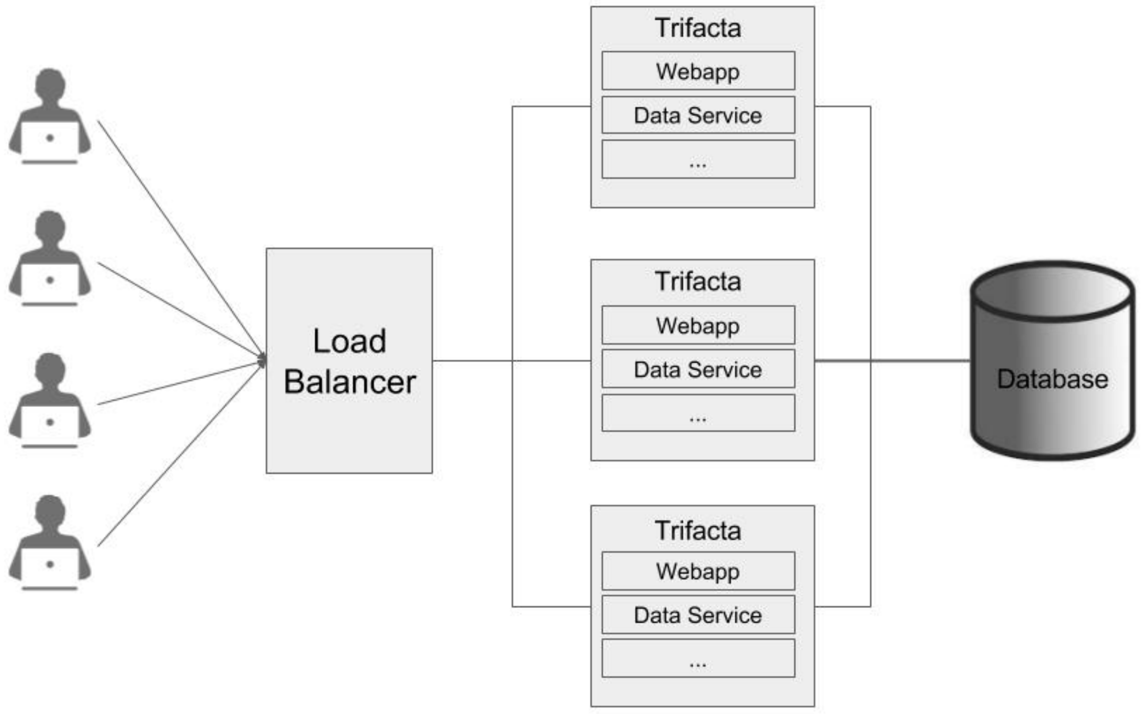
- Job canceling does not work.
- When HA is enabled, the restart feature in the Admin Settings page does not work. You must restart using the command line.
- The platform must be installed on `/opt/trifacta` on every failover node.
- This feature does not apply to the following components:
    - Hadoop cluster (See previous link.)
    - webhdfs/httpfs
    - Sentry
    - Navigator
    - Atlas
    - any other application/infrastructure with which the Trifacta platform can integrate

For more information, see *Configure for High Availability*.

Overview

The Trifacta platform supports an Active-Active HA deployment model, which works well at scale. The architecture features a single load balancer sitting in front of multiple nodes running the Trifacta platform. Each node:

- communicates with the same database
- shares the `/opt/trifacta/conf` and `/opt/trifacta/logs` directories through NFS.



- **Database:** PostGreSQL supports HA. The HA-enabled database runs outside of the cluster of platform nodes and appears to each node as a single database. No application code changes are required.
- **Load balancer:** HAProxy is used for its capabilities on health checking the other HA nodes. This load balance periodically checks the health of the other nodes in the setup.
  - If the health for a given node fails, then the load balancer stops routing traffic to that node while continuing to poll its health.
  - If the node recovers, the load balancer resumes sending traffic to it.
  - Node health is described below.
- **Synchronized configuration:** All nodes share the `/opt/trifacta/conf` mount point, which allows the same configuration files to be visible and accessible on each node.

Job interruption

In case of a failover event, any in-progress job should be marked as failed.

Failover events/scenarios around jobs:

#	Job	Event	Resulting job state
1	In progress	The batch job runner is fine, but executor running the job fails.	Failed ✓
2	In progress	The batch job runner or the node dies.	In Progress ✗
3	Queued	The batch job runner or the node dies.	In Progress[1] ✗
4	Pending	The batch job runner or the node dies.	In Progress[1] [2] ✗

[1] It may not be "In Progress". However, the job has not truly failed.

2 A nuance around #3. There is a feature flag that can be enabled and is enabled by default, which causes pending jobs to be marked as failed on (re)start of batch job runner. However, because this feature indiscriminately marks *all* pending jobs as failed, it cannot be safely enabled in an environment that has multiple running batch job runners.

Installation Topography

The Trifacta platform supports a single load balancer placed in front of multiple nodes, each of which runs the same version of Trifacta Wrangler Enterprise. Content between nodes is shared using an NFS resource mount.

- **master node:** This node is the default one used for hosting and serving the Trifacta platform. Example node information:

  ```
  NFS Server Hostname: server.local
  NFS Server IP Address: 192.168.1.101
  ```

- **client node(s):** These nodes are failover nodes in case the master node is unavailable. Example node information:

  ```
  NFS Client Hostname: client.local
  NFS Client IP Address: 192.168.1.102
  ```

- **load balancer:** This documentation references set up for HAProxy as an example. If you are using a different load balancer, please consult the documentation that came with your product.

Shared resources:

Each node shares the following resources:

- Trifacta databases
- Directories shared via NFS mount:

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

Order of Installation

Steps:

1. All nodes must meet the system requirements. See *System Requirements*.
2. All nodes must have the appropriate ports opened. See *System Ports*.
3. Install the databases.

   NOTE: The databases must be installed in a location that is accessible to all nodes.

   NOTE: When installing databases for high availability access, you should deploy standard access and replication techniques that are consistent with the policies of your enterprise.

See *Install Databases*.

4. Complete the installation process for the server node.

> **NOTE:** After install, do not start the Trifacta node.

See *Install Software*.

5. Repeat the above process for each of the client nodes.
6. The software is installed on all nodes. No node is running the software.

Configuration

Additional configuration is required.

> **NOTE:** Starting and stopping the platform in high availability mode requires additional steps.

For more information, see *Configure for High Availability*.

Install On-Premises

Contents:

To install Trifacta® Wrangler Enterprise inside your enterprise infrastructure, please review and complete the following sections in the order listed below.

Scenario Description

- Installation of Trifacta Wrangler Enterprise on a server on-premises

- Installation of Trifacta databases on a server on-premises
- Integration with a supported Hadoop cluster on premises.
- Base storage layer of HDFS

Preparation

1. **Review Planning Guide:** Please review and verify *Install Preparation* and sub-topics.
2. **Acquire Assets:** Acquire the installation package for your operating system and your license key. For more information, contact *Trifacta Support*.
    1. If you are completing the installation without Internet access, you must also acquire the offline versions of the system dependencies. See *Install Dependencies without Internet Access*.
3. **Deploy Hadoop cluster:** In this scenario, the Trifacta platform does not create a Hadoop cluster. See below.

    > **NOTE:** Installation and maintenance of a working Hadoop cluster is the responsibility of the Trifacta Wrangler Enterprise customer. Guidance is provided below on the requirements for integrating the platform with the cluster.

4. **Deploy Trifacta node:**Trifacta Wrangler Enterprise must be installed on an edge node of the cluster. Details are below.

Limitations: For more information on limitations of this scenario, see *Product Limitations* in the *Install Preparation* area.

Deploy the Cluster

In your enterprise infrastructure, you must deploy a cluster using a supported version of Hadoop to manage the expected data volumes of your Trifacta jobs. For more information on suggested sizing, see *Sizing Guidelines* in the *Install Preparation* area.

When you configure the platform to integrate with the cluster, you must acquire information about the cluster configuration. For more information on the set of information to collect, see *Pre-Install Checklist* in the *Install Preparation* area.

> **NOTE:** By default, smaller jobs are executed on the Trifacta Photon running environment . Larger jobs are executed using Spark on the integrated Hadoop cluster. Spark must be installed on the cluster. For more information, see *System Requirements* in the *Install Preparation* area.

The Trifacta platform supports integration with the following cluster types. For more information on the supported versions, please see the listed sections below.

- See *Supported Deployment Scenarios for Cloudera*.
- See *Supported Deployment Scenarios for Hortonworks*.

Prepare the cluster

Before installing software, please complete the following steps if you are integrating with a Hadoop cluster.

Before you begin, please verify or complete the following:

1. On the Hadoop cluster:
    1. Create a user [`hadoop.user` (default=`trifacta`)] and a group for it [`hadoop.group` (default=`trifactausers`)].
    2. Create the following directories:
        1. `/trifacta`

        2. `/user/trifacta`
    3. Change the ownership of `/trifacta` and `/user/trifacta` to `trifacta:trifacta` or the corresponding values for the Hadoop user in your environment.

       NOTE: You must verify that the `[hadoop.user]` user has complete ownership and full access to Read, Write and Execute on these directories recursively.

2. Verify that WebHDFS is configured and running on the cluster.

3. Software installation is completed on a dedicated node in the cluster. The user installing the Trifacta software must have sudo access.

4. If you are installing on a server with an older instance of Postgres, you should remove the older version or change the default ports.

For more information, see *Prepare Hadoop for Integration with the Platform*.

Additional users may be required. For more information, see *Required Users and Groups* in the *Install Preparation* area.

Deploy the Trifacta node

An edge node of the cluster is required to host the Trifacta platform software. For more information on the requirements of this node, see *System Requirements*.

Install Workflow

Please complete these steps listed in order:

1. **Install software:** Install the Trifacta platform software on the cluster edge node. See *Install Software*.
2. **Install databases:** The platform requires several databases for storage.

   NOTE: The default configuration assumes that you are installing the databases on a PostgreSQL server on the same edge node as the software using the default ports. If you are changing the default configuration, additional configuration is required as part of this installation process.

   For more information, see *Install Databases*.
3. **Start the platform:** For more information, see *Start and Stop the Platform*.
4. **Login to the application:** After software and databases are installed, you can login to the application to complete configuration:
    1. See *Login*.
    2. As soon as you login, you should change the password on the admin account. In the left menu bar, select **Settings > Admin Settings**. Scroll down to Manage Users. For more information, see *Change Admin Password*.

Tip: At this point, you can access the online documentation through the application. In the left menu bar, select **Help menu > Product Docs**. All of the following content, plus updates, is available online. See Documentation below.

Configure for Hadoop

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

Apply cluster configuration files - non-edge node

> **NOTE:** Installation on a non-edge node is not supported. Legacy customers can continue to use a non-edge node, but this deployment is not recommended.

If the Trifacta platform is being installed on a non-edge node, you must copy over the Hadoop Client Configuration files from the cluster.

> **NOTE:** When these files change, you must update the local copies. For this reason, it is best to install on an edge node.

1. Download the Hadoop Client Configuration files from the Hadoop cluster. The required files are the following:
    1. `core-site.xml`
    2. `hdfs-site.xml`
    3. `mapred-site.xml`
    4. `yarn-site.xml`
    5. `hive-site.xml` (if you are using Hive)
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
    ```

For more information, see *Configure for Hadoop*.

Apply cluster configuration files - edge node

If the 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.

For more information, see *Configure for Hadoop*.

Modify Trifacta configuration changes

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

   For more information, see *Configure for Hadoop*.

3. Save your changes and restart the 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-5.4/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/libs
/spark-job-server-bundle.jar:%(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 `true`, if secure impersonation is enabled on your cluster. See *Configure for Secure Impersonation*.
`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 T rifacta node. For more information, see *Configure for Hive*.
`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 *System Ports*.
`env.HADOOP_USER_NAME`	The username of the Hadoop principal used by the platform. By default, this value is `trifacta`.

`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 `true` 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	`false`
Hive is present but not enabled.	`false`
Hive is present and enabled	`true`

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 Trifact a 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	`"hdfs.pathsConfig.libraries"`	`/trifacta/libraries`	Maintains JAR files and other libraries required by Spark. No sensitive information is written to this location.

Trifacta Temp files folder	`"hdfs.pathsConfig.tempFiles"`	`/trifacta/tempfiles`	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 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
spark-job-service.env.HADOOP_CONF_DIR	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.
    1. **For Cloudera 5.x:** This property can be left blank.
    2. **For Hortonworks 2.x:** This property configuration is covered later in this section.
4. Save your changes.

Locate Hive dependencies location

If the 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 5.10:

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

```
"spark-job-service.hiveDependenciesLocation":"%(topOfTree)s/hadoop-deps
/cdh-5.10/build/libs",
```

For more information, see *Configure for Spark*.

Enable High Availability

NOTE: If high availability is enabled on the Hadoop cluster, it must be enabled on the Trifacta platform, even if you are not planning to rely on it. See *Enable Integration with Cluster High Availability*.

Configure for Trifacta platform

Set base storage layer

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.

NOTE: By default, the base storage layer for Trifacta Wrangler Enterprise is set to HDFS. You can change it now, if needed. After this base storage layer is defined, it cannot be changed again.

See *Set Base Storage Layer*.

Verify Operations

NOTE: You can try to verify operations using the Trifacta Photon running environment at this time. While you can also try to run a job on the Hadoop cluster, additional configuration may be required to complete the integration. These steps are listed under Next Steps below.

Prepare Your Sample Dataset

To complete this test, you should locate or create a simple dataset. Your dataset should be created in the format that you wish to test.

Characteristics:

- Two or more columns.
- If there are specific data types that you would like to test, please be sure to include them in the dataset.
- A minimum of 25 rows is required for best results of type inference.
- Ideally, your dataset is a single file or sheet.

Store Your Dataset

If you are testing an integration, you should store your dataset in the datastore with which the product is integrated.

Tip: Uploading datasets is always available as a means of importing datasets.

- You may need to create a connection between the platform and the datastore.
- Read and write permissions must be enabled for the connecting user to the datastore.

- For more information, see *Connections Page*.

Verification Steps

Steps:

1. Login to the application. See *Login*.
2. In the application menu bar, click **Library**.
3. Click **Import Data**. See *Import Data Page*.
    1. Select the connection where the dataset is stored. For datasets stored on your local desktop, click **U pload**.
    2. Select the dataset.
    3. In the right panel, click the Add Dataset to a Flow checkbox. Enter a name for the new flow.
    4. Click **Import and Add to Flow**.
    5.
       Troubleshooting: At this point, you have read access to your datastore from the platform. If not, please check the logs, permissions, and your Trifacta® configuration.
4. In the left menu bar, click the Flows icon. Flows page, open the flow you just created. See *Flows Page*.
5. In the Flows page, click the dataset you just imported. Click **Add new Recipe**.
6. Select the recipe. Click **Edit Recipe**.
7. The initial sample of the dataset is opened in the Transformer page, where you can edit your recipe to transform the dataset.
    1. In the Transformer page, some steps are automatically added to the recipe for you. So, you can run the job immediately.
    2. You can add additional steps if desired. See *Transformer Page*.
8. Click **Run Job**.
    1.
    2. If options are presented, select the defaults.
    3. To generate results in other formats or output locations, click **Add Publishing Destination.** Configur e the output formats and locations.
    4. To test dataset profiling, click the Profile Results checkbox. Note that profiling runs as a separate job and may take considerably longer.
    5. See *Run Job Page*.
    6.
       Troubleshooting: Later, you can re-run this job on a different running environment. Some formats are not available across all running environments.
9. When the job completes, you should see a success message under the Jobs tab in the Flow View page.
    1. **Troubleshooting:** Either the Transform job or the Profiling job may break. To localize the problem, try re-running a job by deselecting the broken job type or running the job on a different running environment (if available). You can also download the log files to try to identify the problem. See *Job Details Page*.
10. Click **View Results** from the context menu for the job listing. In the Job Details page, you can see a visual profile of the generated results. See *Job Details Page*.
11. In the Output Destinations tab, click a link to download the results to your local desktop.
12. Load these results into a local application to verify that the content looks ok.

Checkpoint: You have verified importing from the selected datastore and transforming a dataset. If your job was successfully executed, you have verified that the product is connected to the job running environment and can write results to the defined output location. Optionally, you may have tested profiling of job results. If all of the above tasks completed, the product is operational end-to-end.

Documentation

Tip: You should access online documentation through the product. Online content may receive updates that are not present in PDF content.

You can access complete product documentation online and in PDF format. From within the Trifacta application, select **Help menu > Product Docs**.

Next Steps

After you have accessed the documentation, the following topics are relevant to on-premises deployments. Please review them in order.

NOTE: These materials are located in the *Configuration Guide*.

Topic	Description
Required Platform Configuration	This section covers the following topics, some of which should already be completed: • *Set Base Storage Layer* - The base storage layer must be set once and never changed. • *Create Encryption Key File* - If you plan to integrate the platform with any relational sources, you must create an encryption key file and store it on the Trifacta node • *Running Environment Options* - Depending on your scenario, you may need to perform additional configuration for your available running environment(s) for executing jobs. • *Profiling Options* - In some environments, tweaks to the settings for visual profiling may be required. You can disable visual profiling if needed. • *Configure for Spark* - If you are enabling the Spark running environment, please review and verify the configuration for integrating the platform with the Hadoop cluster instance of Spark.

Configure for Hadoop	• *Configuration by Hadoop Distribution*:
	  ◦ *Configure for Cloudera*
	  ◦ *Configure for Hortonworks*
	• *Configure Hadoop Authentication*

Enable Integration with Compressed Clusters	If the Hadoop cluster uses compression, additional configuration is required.
Enable Integration with Cluster High Availability	If you are integrating with high availability on the Hadoop cluster, please complete these steps. • If you are integrating with high availability on the Hadoop cluster, HttpFS must be enabled in the platform. HttpFS is required in other, less-common cases. See *Enable HttpFS*.
Configure for Hive	Integration with the Hadoop cluster's instance of Hive.
Configure for KMS	Integration with the Hadoop cluster's key management system (KMS) for encrypted transport. Instructions are provided for distribution-specific versions of Hadoop.

Configure Security	A list of topics on applying additional security measures to the Trifa cta platform and how integrates with Hadoop.
Configure SSO for AD-LDAP	Please complete these steps if you are integrating with your enterprise's AD/LDAP Single Sign-On (SSO) system.

Install for Azure

Contents:

This install process applies to installing Trifacta® Wrangler Enterprise on an Azure infrastructure that you manage.

Azure Marketplace deployments:

NOTE: Content in this section does not apply to deployments from the Azure Marketplace, which provide fewer deployment and configuration options. For more information, see the Azure Marketplace.

Scenario Description

NOTE: All hardware in use for supporting the platform is maintained within the enterprise infrastructure on Azure.

- Installation of Trifacta Wrangler Enterprise on a node in Microsoft Azure
- Installation of Trifacta databases on the same node
- Integration with a supported cluster for running jobs.
- Base storage layer and backend datastore of ADLS or WASB
- High availability or failover of the Trifacta node is not supported in Azure.
- High availability of cluster components is automatically managed by the HDI cluster.
  - Auto-management does not apply to non-Hadoop clusters, such as Azure Databricks.

For more information on deployment scenarios, see *Supported Deployment Scenarios for Azure*.

Product Limitations

- The application user credentials are used to access to the HDI cluster. Details are provided below.
- ADLS/Storage Blob access is only for the HDInsight cluster's primary storage. Additional storage accounts are not supported.
- HDFS must be set as the base storage layer of the Trifacta platform. Details are provided later.
    - S3 integration and AWS-based integrations such as Redshift are not supported.
- Use of HttpFS is not supported.
- Security features such as Kerberos and secure impersonation are not supported.

For more information on the limitations of this deployment scenario, see *Product Limitations*.

Pre-requisities

Desktop Requirements

- All desktop users of the platform should have a supported version of Google Chrome installed on their desktops.
    - For more information. see *Desktop Requirements*.
    - If a supported browser is not available within your enterprise, desktop users can install the Trifacta enterprise application as a separate application. For more information, see *Install Desktop Application*.
- All desktop users must be able to connect to the EC2 instance through the enterprise infrastructure.

Azure Pre-requisites

Depending on which of the following Azure components you are deploying, additional pre-requisites and limitations may apply. Please review these sections as well.

- Cluster:
    - *Configure for HDInsight*
    - *Configure for Azure Databricks*
- Storage:
    - *Enable ADLS Access*
    - *Enable WASB Access*
- *Configure SSO for Azure AD*

Preparation

Before you begin, please verify that you have completed the following:

1. **Review Planning Guide:** Please review and verify *Install Preparation* and sub-topics.
2. **Read:** Please read this entire document before you create the EMR cluster or install the Trifacta platform.
3. **Acquire Assets:** Acquire the installation package for your operating system and your license key. For more information, contact *Trifacta Support*.
    1. If you are completing the installation without Internet access, you must also acquire the offline versions of the system dependencies. See *Install Dependencies without Internet Access*.
4. **Cluster sizing:** Before you begin, you should allocate sufficient resources for the cluster. For guidance, please contact your Trifacta representative.
5. **Node:** Review the system requirements for the node hosting the Trifacta platform. See *System Requirements*.
    1. The required set of ports must be enabled for listening. See *System Ports*.
    2. This node should be dedicated for Trifacta use.
6. **Databases:**
    1. The platform utilizes a set of databases that must be accessed from the Trifacta node. Databases are installed as part of the workflow described later.

2. For more information on the supported databases and versions, see *System Requirements*.
3. For more information on database installation requirements, see *Install Databases*.

Limitations: For more information on limitations of this scenario, see *Product Limitations* in the *Install Preparation* area.

Deploy the Cluster

Deploy and provision a cluster of one of the supported types. The Trifacta platform supports integrations with multiple cluster types.

> **NOTE:** Before you deploy, you should review cluster sizing options. For guidance, please contact your Trifacta representative.

Primary storage of the cluster may be set to an existing Azure Data Lake Store or Blob Storage.

- Any additional storage associated with the cluster is not available through the Trifacta application.

For more information, see *Supported Deployment Scenarios for Azure*.

Deploy the Trifacta node

In your Azure infrastructure, you must deploy a suitable VM for the installation of the Trifacta platform.

The operating system requirements for the VM for installing the platform vary depending on the type of job execution cluster with which you are running.

Cluster Type	Supported O/S for VM	Notes
HDInsight	Ubuntu only	Trifacta platform must be installed on an edge node of the HDInsight cluster.
Azure Databricks	CentOS and Ubuntu	

- When you configure the platform to integrate with the cluster, you must acquire some information about the cluster resources. For more information on the set of information to collect, see *Pre-Install Checklist* in the *Install Preparation* area.
- For more information, see *System Requirements* in the *Install Preparation* area.
- A set of ports must be opened on the VM for the platform. For more information, see *System Ports* in the *Install Preparation* area.

For more information on the supported EMR distributions, see *Supported Deployment Scenarios for Azure*.

Prepare the cluster

1. Create the following directories, which are specified by parameter in the platform.

Default HDFS path	Platform configuration property
/user/trifacta	
/trifacta	
/trifacta/dictionaries	hdfs.pathsConfig.dictionaries
/trifacta/libraries	hdfs.pathsConfig.libraries

/trifacta/queryResults	hdfs.pathsConfig.batchResults
/trifacta/tempfiles	hdfs.pathsConfig.tempFiles
/trifacta/uploads	hdfs.pathsConfig.fileUpload
/trifacta/.datasourceCache	hdfs.pathsConfig.globalDatasourceCache

2. Change the ownership of the above directories to `trifacta:trifacta` or the corresponding values for the S3 user in your environment.

Additional users may be required. For more information, see *Required Users and Groups* in the *Install Preparation* area.

Install Workflow

Please complete these steps listed in order.

1 - Install Software

Install the Trifacta platform software on the node you created.

NOTE: You must follow the instructions provided for Ubuntu installation.

See *Install Software*.

2 - Install Databases

The platform requires several databases for storing metadata.

NOTE: The software assumes that you are installing the databases on a PostgreSQL server on the same node as the software. If you are not or are changing database names or ports, additional configuration is required as part of this installation process.

For more information, see *Install Databases* in the Databases Guide.

3 - Start the platform

For more information, see *Start and Stop the Platform*.

4 - Login to the Application

After software and databases are installed, you can login to the application to complete configuration. See *Login*.

As soon as you login, you should change the password on the admin account. In the left menu bar, select **Setting s > Admin Settings**. Scroll down to Manage Users. For more information, see *Change Admin Password*.

Tip: At this point, you can access the online documentation through the application. In the left menu bar, select **Help menu > Product Docs**. All of the following content, plus updates, is available online. See Documentation below.

Configuration Workflow

After you have completed the above topics, you can complete the configuration for your deployment below.

NOTE: The following configuration topics are not part of this installation guide. You should log in to the application and access the links below.

1. **Configure for Azure:** Configure the platform to work with Azure.
2. **Integrate with cluster:** If the application is up and running, you can configure to the backend cluster for running jobs. Choose one of the following:
    1. HDInsight
    2. Azure Databricks
3. **Integrate with backend storage:**
    1. **Set base storage layer:** The base storage layer must be set at the time of install and cannot be changed. See *Set Base Storage Layer*.
    2. ADLS
    3. WASB
4. **Verify operations:** At this point, you should be able to run a job. See *Verify Operations*.
5. **Create additional connections:** Through connections, you can access other sources of data and, optionally, publish job results.

Documentation

You can access complete product documentation online and in PDF format. From within the product, select **Help menu > Product Docs**.The following configuration topics are relevant to Azure deployments. Please review them in order.

Topic	Description
Supported Deployment Scenarios for Azure	Matrix of supported Azure components.
Configure for Azure	Top-level configuration topic on integrating the platform with Azure. **Tip:** You should review this page.
Configure for HDInsight	Review this section if you are integrating the Trifacta platform with a pre-existing HDI cluster.
Configure for Azure Databricks	Review this section if you are integrating with a pre-existing Azure Databricks cluster.
Enable ADLS Access	Configuration to enable access to ADLS.
Enable WASB Access	Configuration to enable access to WASB.
Verify Operations	You should be able to verify platform operations by running a simple job at this time.
Relational Connections	To enable, see *Enable Relational Connections*. Azure-specific relational connections: • *Create SQL DW Connections* • *Create Azure SQL Database Connections*
Configure SSO for Azure AD	How to integrate the Trifacta platform with Azure Active Directory for Single Sign-On.

Configure Server Access through Proxy

When you attempt to launch the application, you may receive an error message similar to the following:

```
No internet connection
Remote server timed out.
```

In some environments, your desktop machine may need to connect to the Internet through a proxy server. If you are using Wrangler Enterprise desktop application, it needs to know the proxy server to which to connect in order to access the Trifacta® node.

Please complete the following configuration steps to access the Trifacta servers.

Steps:

1. In the No internet connection dialog, click **Configure Proxy Settings**.
2. Please provide the following configuration information for your proxy server:
    1. **Proxy Host:** The URL of the proxy server. Please include the protocol identifier (e.g. `http://` or `https://`).
    2. **Proxy Port:** The port number to use to connect to the proxy server. In a URL, this value appears after a colon (e.g. `http://myproxy.example.com:8080`).
    3. **Username:** (optional) If your proxy requires a username to access, please enter it here.
    4. **Password:** (optional) Password associated with the user name.
3. Click **Save Proxy Settings and Restart**.

When the application restarts, you should be able to connect to the login screen.

> **NOTE:** If you continue to have difficulties connecting to the Internet, please contact your network administrator or Internet provider.

Install Software

To install Trifacta® Wrangler Enterprise, please review and complete the following sections in the order listed below.

Topics:

- *Install Dependencies without Internet Access*
- *Install on CentOS and RHEL*
- *Install on Ubuntu*
- *Install for Docker*
- *License Key*
- *Install Desktop Application*
- *Start and Stop the Platform*
- *Login*

Install Dependencies without Internet Access

Offline dependencies should be included in the URL location that Trifacta® provided to you. Please use the `\*deps\*` file.

> **NOTE:** If your installation server is connected to the Internet, the required dependencies are automatically downloaded and installed for you. You may skip this section.

Use the steps below to acquire and install dependencies required by the Trifacta platform. If you need further assistance, please contact *Trifacta Support*.

Install software dependencies without Internet access for CentOS or RHEL:

1. In a CentOS or RHEL environment, the dependencies repository must be installed into the following directory:

   ```
   /var/local/trifacta
   ```

2. The following commands configure Yum to point to the repository in `/var/local/trifacta`, which yum knows as `local`. Repo permissions are set appropriately. Commands:

   ```
   tar xvzf <DEPENDENCIES_ARCHIVE>.tar.gz
   mv local.repo /etc/yum.repos.d
   mv trifacta /var/local
   chown -R root:root /var/local/trifacta
   chmod -R o-w+r /var/local/trifacta
   ```

3. The following command installs the RPM while disable all repos other than local, which prevents the installer from reaching out to the Internet for package updates:

   NOTE: The disabling of repositories only applies to this command.

   ```
   sudo yum --disablerepo=* --enablerepo=local install <INSTALLER>.rpm
   ```

4. If the above command fails and complains about a missing repo, you can add the missing repo to the `enablerepo` list. For example, if the `centos-base` repo is reported as missing, then the command would be the following:

   ```
   sudo yum --disablerepo=* --enablerepo=local,centos-base install
   <INSTALLER>.rpm
   ```

5. If you do not have a supported version of a Java Developer Kit installed on the Trifacta node, you can use the following command to install OpenJDK, which is included in the offline dependencies:

   ```
   sudo yum --disablerepo=* --enablerepo=local,centos-base install
   java-1.8.0-openjdk-1.8.0 java-1.8.0-openjdk-devel
   ```

Install database dependencies without Internet access:

If you are installing the databases on a node without Internet access, you can install the dependencies using either of the following commands:

NOTE: This step is only required if you are installing the databases on the same node where the software is installed.

For PostgreSQL:

```
sudo yum --disablerepo=* --enablerepo=local install postgresql96-server
```

For MySQL:

```
sudo yum --disablerepo=* --enablerepo=local install mysql-community-
server
```

NOTE: You must also install the MySQL JARs on the Trifacta node. These instructions are provided later.

Database are installed after the software is installed. For more information, see *Install Databases*.

Install dependencies without Internet access in Ubuntu:

If you are trying to perform a manual installation of dependencies in Ubuntu, please contact *Trifacta Support*.

Install on CentOS and RHEL

Contents:

This guide takes you through the steps for installing Trifacta® Wrangler Enterprise software on CentOS or Red Hat.

For more information on supported operating system versions, see *System Requirements*.

Preparation

Before you begin, please complete the following.

NOTE: Except for database installation and configuration, all install commands should be run as the root user or a user with similar privileges. For database installation, you will be asked to switch the database user account.

Steps:

1. Set the node where Trifacta Wrangler Enterprise is to be installed.
    1. Review the *System Requirements* and verify that all required components have been installed.
    2. Verify that all required system ports are opened on the node. See *System Ports*.
2. Review the *Desktop Requirements*.

    NOTE: Trifacta Wrangler Enterprise requires the installation of Google Chrome on each desktop. For more information, see *Desktop Requirements*.

3. Review the *System Dependencies*.

    NOTE: If you are installing on node without access to the Internet, you must download the offline dependencies before you begin. See *Install Dependencies without Internet Access*.

4. Acquire your *License Key*.
5. Install and verify operations of the datastore, if used.

    NOTE: Access to the Spark cluster is required.

6. Verify access to the server where the Trifacta platform is to be installed.
7. **Cluster Configuration:** Additional steps are required to integrate the Trifacta platform with the cluster. See *Prepare Hadoop for Integration with the Platform*.

Installation

1. Install Dependencies

Without Internet access

If you have not done so already, you may download the dependency bundle with your release directly from Trifacta . For more information, see *Install Dependencies without Internet Access*.

With Internet access

Use the following to add the hosted package repository for CentOS/RHEL, which will automatically install the proper packages for your environment.

```
# If the client has curl installed ...
curl https://packagecloud.io/install/repositories/trifacta/dependencies
/script.rpm.sh | sudo bash

# Otherwise, you can also use wget ...
wget -qO- https://packagecloud.io/install/repositories/trifacta
/dependencies/script.rpm.sh | sudo bash
```

2. Install JDK

By default, the Trifacta node uses OpenJDK for accessing Java libraries and components. In some environments, basic setup of the node may include installation of a JDK. Please review your environment to verify that an appropriate JDK version has been installed on the node.

NOTE: Use of Java Development Kits other than OpenJDK is not currently supported. However, the platform may work with the Java Development Kit of your choice, as long as it is compatible with the supported version(s) of Java. See *System Requirements*.

NOTE: OpenJDK is included in the offline dependencies, which can be used to install the platform without Internet access. For more information, see *Install Dependencies without Internet Access*.

The following commands can be used to install OpenJDK. These commands can be modified to install a separate compatible version of the JDK.

```
sudo yum install java-1.8.0-openjdk-1.8.0 java-1.8.0-openjdk-devel
```

NOTE: If `java-1.8.0-openjdk-devel` is not included, the batch job runner service, which is required, fails to start.

JAVA_HOME:

By default, the `JAVA_HOME` environment variable is configured to point to a default install location for the OpenJDK package.

NOTE: If you have installed a JDK other than the OpenJDK version provided with the software, you must set the `JAVA_HOME` environment variable on the Trifacta node to point to the correct install location.

The property value must be updated in the following locations:

1. Edit the following file: `/opt/trifacta/conf/env.sh`
2. Save changes.

3. Install Trifacta package

NOTE: If you are installing without Internet access, you must reference the local repository. The command to execute the installer is slightly different. See *Install Dependencies without Internet Access*.

NOTE: Installing the Trifacta platform in a directory other than the default one is not supported or recommended.

Install the package with yum, using root:

```
sudo yum install <rpm file>
```

4. Verify Install

The product is installed in the following directory:

```
/opt/trifacta
```

JAVA_HOME:

The platform must be made aware of the location of Java.

Steps:

1. Edit the following file: `/opt/trifacta/conf/trifacta-conf.json`
2. Update the following parameter value:

   ```
   "env": {
     "JAVA_HOME": "/usr/lib/jvm/java-1.8.0-openjdk.x86_64"
   },
   ```

3. Save changes.

5. Install License Key

Please install the license key provided to you by Trifacta. See *License Key*.

6. Store install packages

For safekeeping, you should retain all install packages that have been installed with this Trifacta deployment.

7. Install and configure Trifacta databases

The Trifacta platform requires installation of several databases. If you have not done so already, you must install and configure the databases used to store Trifacta metadata. See *Install Databases*.

Configuration

After installation is complete, additional configuration is required.

The Trifacta platform requires additional configuration for a successful integration with the datastore. Please review and complete the necessary configuration steps. For more information, see *Configure*.

Install on Ubuntu

Contents:

This guide takes you through the steps for installing Trifacta® Wrangler Enterprise software on Ubuntu.

For more information on supported operating system versions, see *System Requirements*.

Preparation

Before you begin, please complete the following.

NOTE: Except for database installation and configuration, all install commands should be run as the root user or a user with similar privileges. For database installation, you will be asked to switch the database user account.

Steps:

1. Set the node where Trifacta Wrangler Enterprise is to be installed.
   1. Review the *System Requirements* and verify that all required components have been installed.
   2. Verify that all required system ports are opened on the node. See *System Ports*.
2. Review the *Desktop Requirements*.

   NOTE: Trifacta Wrangler Enterprise requires the installation of Google Chrome on each desktop.

3. Review the *System Dependencies*.

   NOTE: If you are installing on node without access to the Internet, you must download the offline dependencies before you begin. See *Install Dependencies without Internet Access*.

4. Acquire your *License Key*.
5. Install and verify operations of the datastore, if used.

NOTE: Access to the cluster may be required.

6. Verify access to the server where the Trifacta platform is to be installed.
7. **Cluster configuration:** Additional steps are required to integrate the Trifacta platform with the cluster. See *Prepare Hadoop for Integration with the Platform*.

Installation

1. Install Dependencies

Without Internet access

If you have not done so already, you may download the dependency bundle with your release directly from Trifacta . For more information, see *Install Dependencies without Internet Access*.

With Internet access

Use the following to add the hosted package repository for Ubuntu, which will automatically install the proper packages for your environment.

NOTE: Install curl if not present on your system.

Then, execute the following command:

NOTE: Run the following command as the root user. In proxied environments, the script may encounter issues with detecting proxy settings.

```
curl https://packagecloud.io/install/repositories/trifacta/dependencies
/script.deb.sh | sudo bash
```

Special instructions for Ubuntu installs

These steps manually install the correct and supported version of the following:

- nodeJS
- nginX

Due to a known issue resolving package dependencies on Ubuntu, please complete the following steps prior to installation of other dependencies or software.

1. Login to the Trifacta node as an administrator.
2. Execute the following command to install the appropriate versions of nodeJS and nginX.
    1. Ubuntu 14.04:

    ```
    sudo apt-get install nginx=1.12.2-1~trusty nodejs=10.13.0-
    1nodesource1
    ```

    2. Ubuntu 16.04

```
sudo apt-get install nginx=1.12.2-1~xenial nodejs=10.13.0-
1nodesource1
```

3. Continue with the installation process.

2. Install JDK

By default, the Trifacta node uses OpenJDK for accessing Java libraries and components. In some environments, basic setup of the node may include installation of a JDK. Please review your environment to verify that an appropriate JDK version has been installed on the node.

NOTE: Use of Java Development Kits other than OpenJDK is not currently supported. However, the platform may work with the Java Development Kit of your choice, as long as it is compatible with the supported version(s) of Java. See *System Requirements*.

NOTE: OpenJDK is included in the offline dependencies, which can be used to install the platform without Internet access. For more information, see *Install Dependencies without Internet Access*.

The following commands can be used to install OpenJDK. These commands can be modified to install a separate compatible version of the JDK.

```
sudo apt-get install openjdk-8-jre-headless
```

JAVA_HOME:

By default, the `JAVA_HOME` environment variable is configured to point to a default install location for the OpenJDK package.

NOTE: If you have installed a JDK other than the OpenJDK version provided with the software, you must set the `JAVA_HOME` environment variable on the Trifacta node to point to the correct install location.

The property value must be updated in the following locations:

1. Edit the following file: `/opt/trifacta/conf/env.sh`
2. Save changes.

3. Install Trifacta package

NOTE: If you are installing without Internet access, you must reference the local repository. The command to execute the installer is slightly different. See *Install Dependencies without Internet Access*.

NOTE: Installing the Trifacta platform in a directory other than the default one is not supported or recommended.

Install the package with apt, using root:

```
sudo dpkg -i <deb file>
```

The previous line may return an error message, which you may ignore. Continue with the following command:

```
sudo apt-get -f -y install
```

4. Verify Install

The product is installed in the following directory:

```
/opt/trifacta
```

JAVA_HOME:

The platform must be made aware of the location of Java.

Steps:

1. Edit the following file: `/opt/trifacta/conf/trifacta-conf.json`
2. Update the following parameter value:

   ```
   "env": {
     "JAVA_HOME": "/usr/lib/jvm/java-1.8.0-openjdk.x86_64"
   },
   ```

3. Save changes.

5. Install License Key

Please install the license key provided to you by Trifacta. See *License Key*.

6. Store install packages

For safekeeping, you should retain all install packages that have been installed with this Trifacta deployment.

7. Install and configure Trifacta databases

The Trifacta platform requires installation of several databases. If you have not done so already, you must install and configure the databases used to store Trifacta metadata. See *Install Databases*.

Configuration

After installation is complete, additional configuration is required.

The Trifacta platform requires additional configuration for a successful integration with the datastore. Please review and complete the necessary configuration steps. For more information, see *Configure*.

Install for Docker

Contents:

This guide steps through the process of acquiring and deploying a Docker image of the Trifacta® platform in your Docker environment. Optionally, you can build the Docker image locally, which enables further configuration options.

Deployment Scenario

- Trifacta Wrangler Enterprise deployed into a customer-managed environment: On-premises, AWS, or Azure.
- PostgreSQL 9.6 installed either:
  - Locally
  - Remote server

Limitations

- You cannot upgrade to a Docker image from a non-Docker deployment.
- You cannot switch an existing installation to a Docker image.
- Supported distributions of Cloudera or Hortonworks:
  - *Supported Deployment Scenarios for Cloudera*
  - *Supported Deployment Scenarios for Hortonworks*
- The base storage layer of the platform must be HDFS. Base storage of S3 is not supported.
- High availability for the Trifacta platform in Docker is not supported.
- SSO integration is not supported.

Requirements

Support for orchestration through Docker Compose only

- Docker version 17.12 or later
- Docker-Compose 1.11.2 or later. Version must be compatible with your version of Docker.

Docker Daemon

	Minimum	Recommended
CPU Cores	2 CPU	4 CPU
Available RAM	8 GB RAM	10+ GB RAM

Preparation

1. Review the *Desktop Requirements*.

   > **NOTE:** Trifacta Wrangler Enterprise requires the installation of Google Chrome on each desktop. For more information, see *Desktop Requirements*.

2. Acquire your *License Key*.

Acquire Image

You can acquire the latest Docker image using one of the following methods:

1. Acquire from FTP site.
2. Build your own Docker image.

Acquire from FTP site

Steps:

1. Download the following files from the FTP site:
   1. `trifacta-docker-setup-bundle-x.y.z.tar`
   2. `trifacta-docker-image-x.y.z.tar`

   > **NOTE:** `x.y.z` refers to the version number (e.g. `6.4.0`).

2. Untar the `setup-bundle` file:

   ```
   tar xvf trifacta-docker-setup-bundle-x.y.z.tar
   ```

3. Files are extracted into a `docker` folder. Key files:

File	Description
docker-compose-local-postgres.yaml	Runtime configuration file for the Docker image when PostgreSQL is to be running on the same machine. More information is provided below.
docker-compose-local-mysql.yaml	Runtime configuration file for the Docker image when MySQL is to be running on the same machine. More information is provided below.
docker-compose-remote-db.yaml	Runtime configuration file for the Docker image when the database is to be accessed from a remote server. **NOTE:** You must manage this instance of the database. More information is provided below.
README-running-trifacta-container.md	Instructions for running the Trifacta container **NOTE:** These instructions are referenced later in this workflow.
README-building-trifacta-container.md	Instructions for building the Trifacta container **NOTE:** This file does not apply if you are using the provided Docker image.

4. Load the Docker image into your local Docker environment:

```
docker load < trifacta-docker-image-x.y.z.tar
```

5. Confirm that the image has been loaded. Execute the following command, which should list the Docker image:

```
docker images
```

6. You can now configure the Docker image. Please skip that section.

Build your own Docker image

As needed, you can build your own Docker image.

Requirements

- Docker version 17.12 or later
- Docker Compose 1.11.2 or newer. It should be compatible with above version of Docker.

Build steps

1. Acquire the RPM file from the FTP site:

    NOTE: You must acquire the el7 RPM file for this release.

2. In your Docker environment, copy the `trifacta-server\*.rpm` file to the same level as the `Dockerfile`.
3. Verify that the `docker-files` folder and its contents are present.
4. Use the following command to build the image:

```
docker build -t trifacta/server-enterprise:latest .
```

5. This process could take about 10 minutes. When it is completed, you should see the build image in the Docker list of local images.

    NOTE: To reduce the size of the Docker image, the Dockerfile installs the trifacta-server RPM file in one stage and then copies over the results to the final stage. The RPM is not actually installed in the final stage. All of the files are properly located.

6. You can now configure the Docker image.

Configure Docker Image

Before you start the Docker container, you should review the properties for the Docker image. In the provided image, please open the appropriate `docker-compose` file:

File	Description
docker-compose-local-postgres.yaml	Database properties in this file are pre-configured to work with the installed instance of PostgreSQL, although you may wish to change some of the properties for security reasons.
docker-compose-local-mysql.yaml	Database properties in this file are pre-configured to work with the installed instance of MySQL, although you may wish to change some of the properties for security reasons.
docker-compose-remote-db.yaml	The Trifacta databases are to be installed on a remote server that you manage. **NOTE:** Additional configuration is required.

NOTE: You may want to create a backup of this file first.

Key general properties:

NOTE: Avoid modifying properties that are not listed below.

Property	Description
image	This reference must match the name of the image that you have acquired.
container_name	Name of container in your Docker environment.
ports	Defines the listening port for the Trifacta application. Default is `3005`. **NOTE:** If you must change the listening port, additional configuration is required after the image is deployed. See *Change Listening Port* For more information, see *System Ports*.

Database properties:

These properties pertain to the installation of the database to which the Trifacta application connects.

Property	Description
DB_INIT	If set to `true`, database initialization steps are performed at startup. **NOTE:** This step applies only if you are starting the container for the first time, and PostgreSQL databases will be installed locally.
DB_TYPE	Set this value to `postgresql` or `mysql`.
DB_HOST_NAME	Hostname of the machine hosting the databases. Leave value as `localhost` for local installation.
DB_HOST_PORT	(Remote only) Port number to use to connect to the databases. Default is `5432`. **NOTE:** If you are modifying, additional configuration is required after installation is complete. See *Change Database Port*.
DB_ADMIN_USERNAME	Admin username to be used to create DB roles/databases. Modify this value for remote installation. **NOTE:** If you are modifying this value, additional configuration is required. Please see the documentation for your database version.
DB_ADMIN_PASSWORD	Admin password to be used to create DB roles/databases. Modify this value for remote installation.

Kerberos properties:

If your Hadoop cluster is protected by Kerberos, please review the following properties.

Property	Description
KERBEROS_KEYTAB_FILE	Full path inside of the container where the Kerberos keytab file is located. Default value: `/opt/trifacta/conf/trifacta.keytab` **NOTE:** The keytab file must be imported and mounted to this location. Configuration details are provided later.
KERBEROS_KRB5_CONF	Full path inside of the container where the Kerberos `krb5.conf file` is located. Default: `/opt/krb-config/krb5.conf`

Hadoop distribution client JARs:

Please enable the appropriate path to the client JAR files for your Hadoop distribution. In the following example, the Cloudera path has been enabled, and the Hortonworks path has been disabled:

```
      # Mount folder from outside for necessary hadoop client jars
      # For CDH
      - /opt/cloudera:/opt/cloudera
      # For HDP
      #- /usr/hdp:/usr/hdp
```

Please modify these lines if you are using Hortonworks.

Volume properties:

These properties govern where volumes are mounted in the container.

NOTE: These values should not be modified unless necessary.

Property	Description
volumes.conf	Full path in container to the Trifacta configuration directory. Default: `/opt/trifacta/conf`

volumes.logs	Full path in container to the Trifacta logs directory. Default: `/opt/trifacta/logs`
volumes.license	Full path in container to the Trifacta license directory. Default: `/trifacta-license`

Start Server Container

After you have performed the above configuration, execute the following to initialize the Docker container:

```
docker-compose -f <docker-compose-filename>.yaml run trifacta initfiles
```

When the above is started for the first time, the following directories are created on the localhost:

Directory	Description
./trifacta-data	Used by the Trifacta container to expose the `conf` and `logs` directories.

Import Additional Configuration Files

After you have started the new container, additional configuration files must be imported.

Import license key file

The Trifacta license file must be staged for use by the platform. Stage the file in the following location in the container:

NOTE: If you are using a non-default path or filename, you must update the `<docker-compose-filename>.yaml` file.

```
trifacta-license/license.json
```

Import Hadoop distribution libraries

If the container you are creating is on the edge node of your Hadoop cluster, you must provide the Hadoop libraries.

1. You must mount the Hadoop distribution libraries into the container. For more information on the libraries, see the documentation for your Hadoop distribution.
2. The Docker Compose file must be made aware of these libraries. Details are below.

Import Hadoop cluster configuration files

Some core cluster configuration files from your Hadoop distribution must be provided to the container. These files must be copied into the following directory within the container:

```
./trifacta-data/conf/hadoop-site
```

For more information, see *Configure for Hadoop* in the Configuration Guide.

Install Kerberos client

If Kerberos is enabled, you must install the Kerberos client and keytab on the node container. Copy the keytab file to the following stage location:

```
/trifacta-data/conf/trifacta.keytab
```

See *Configure for Kerberos Integration* in the Configuration Guide.

Perform configuration changes as necessary

The primary configuration file for the platform is in the following location in the launched container:

```
/opt/trifacta/conf/trifacta-conf.json
```

NOTE: Unless you are comfortable working with this file, you should avoid direct edits to it. All subsequent configuration can be applied from within the application, which supports some forms of data validation. It is possible to corrupt the file using direct edits.

Configuration topics are covered later.

Start and Stop the Container

Stop container

Stops the container but does not destroy it.

NOTE: Application and local database data is not destroyed. As long as the `<docker-compose-filename>.yaml` properties point to the correct location of the `*-data` files, data should be preserved. You can start new containers to use this data, too. Do not change ownership on these directories.

```
docker-compose -f <docker-compose-filename>.yaml stop
```

Restart container

Restarts an existing container.

```
docker-compose -f <docker-compose-filename>.yaml start
```

Recreate container

Recreates a container using existing local data.

```
docker-compose -f <docker-compose-filename>.yaml up --force-recreate -d
```

Stop and destroy the container

Stops the container and destroys it.

The following also destroys all application configuration, logs, and database data. You may want to back up these directories first.

```
docker-compose -f <docker-compose-filename>.yaml down
```

Local PostgreSQL:

```
sudo rm -rf trifacta-data/ postgres-data/
```

Local MySQL or remote database:

```
sudo rm -rf trifacta-data/
```

Verify Deployment

1. Verify access to the server where the Trifacta platform is to be installed.
2. **Cluster Configuration:** Additional steps are required to integrate the Trifacta platform with the cluster. See *Prepare Hadoop for Integration with the Platform*.
3. Start the platform within the container. See *Start and Stop the Platform*.

Configuration

After installation is complete, additional configuration is required. You can complete this configuration from within the application.

Steps:

1. Login to the application. See *Login*.
2. The primary configuration interface is the Admin Settings page. From the left menu, select **Settings menu > Settings > Admin Settings**. For more information, see *Admin Settings Page* in the Admin Guide.
3. Workspace-level configuration can also be applied. From the left menu, select **Settings menu > Settings > Workspace Admin**. For more information, see *Workspace Admin Page* in the Admin Guide.

The Trifacta platform requires additional configuration for a successful integration with the datastore. Please review and complete the necessary configuration steps. For more information, see *Configure* in the Configuration Guide.

License Key

Contents:

Download license key file

If you have not done so already, the license key file is available where you have acquired the installation package. Please download `license.json`.

Acquire license key

A valid license key (`license.json`) is provided to each customer prior to installation. Your license key file is a JSON file that contains important information on your license.

NOTE: If your license key has expired, please contact *Trifacta Support*.

Install your license key

If you are updating your license, you may want to save your previous license key to a new location before overwriting.

NOTE: Do not maintain multiple license key files in this directory.

To apply your license key, copy the key file to the following location in the Trifacta® deployment:

```
/opt/trifacta/license
```

Update your license key

After you have installed your license key, you can update your license with a new one through the Admin Settings page. See *Admin Settings Page*.

Changing the license key location

By default, the license key file in use must be named: `license.json`.

If needed, you can change the path and filename of the license key. The property is the following:

```
"license.location"
```

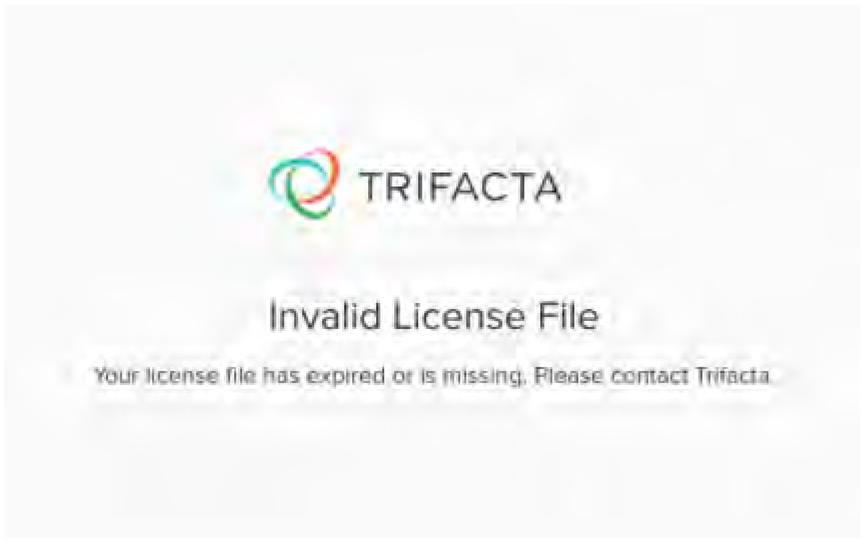
See *Admin Settings Page*.

Expired license

NOTE: If your license expires, you cannot use the product until a new and valid license key file has been applied. When administrators attempt to login to the application, they are automatically redirected to a location from which they can upload a new license key file.

Invalid license key file

When you start the Trifacta platform, you may see the following:



Invalid License File

Your license file has expired or is missing. Please contact Trifacta

Your license key is missing or has expired. Please contact *Trifacta Support*.

Install Desktop Application

Contents:

If your environment does not support the use of Chrome, you can install the Wrangler Enterprise desktop application to provide the same access and functionality as the Trifacta® application. This desktop application connects to the enterprise Trifacta instance and provides the same capabilities without requiring a locally installed version of Chrome browser.

Wrangler Enterprise desktop application is a hybrid desktop application. Your local application instance accesses registered data files located in the datastore to which the Trifacta node is connected.

Install Process

NOTE: The Wrangler Enterprise desktop application is a 64-bit Microsoft Windows application. It requires a 64-bit version of Windows to execute. The application also supports Single Sign On (SSO), if it is enabled.

Download

To begin, you must download the following Windows MSI file (`TrifactaEnterpriseSetup.msi`) from the location where your software was provided.

If you are planning to automate installation to desktops in your environment, please also download `setTrifactaServer.ps1`.

Setup

Before you begin, you should perform any necessary configuration of the Trifacta node before deploying the instances of the application. See *Configure for Desktop Application*.

Install for Windows

Steps:

1. On your Windows desktop, double-click the MSI file.
2. Follow the on-screen instructions to install the software.

Windows Command Line Installation and Configuration

As an alternative, you can perform installation and initial configuration from the command line. Download the MSI and the PS1 files to a local directory that is accessible.

NOTE: For command line install, you must download from the `setTrifactaServer.ps1` from the download location.

Install software:

```
msiexec /i <path_to_TrifactaEnterpriseSetup.msi> /passive
```

Configure URL of Trifacta node:

```
setTrifactaServer.ps1 -trifactaServer <server_url> -installDir
<local_dir>
```

Parameter	Description
`trifactaServer`	(Required) URL of the server hosting the Trifacta platform. Format: `<http|https>://<host>:<port>`
`installDir`	(Optional) Specifies the installation directory in the local environment. If not specified, installation directory defaults to use the same path as the installer.
common installer parameters	This command supports the following Windows installer parameters: Verbose, Debug, ErrorAction, ErrorVariable, WarningAction, WarningVariable, OutBuffer, PipelineVariable, and OutVariable. For more information, see *about_CommonParameters* here: *http://go.microsoft.com/fwlink/?LinkID=113216*.

After this install is completed, desktop users should be able to use the application normally.

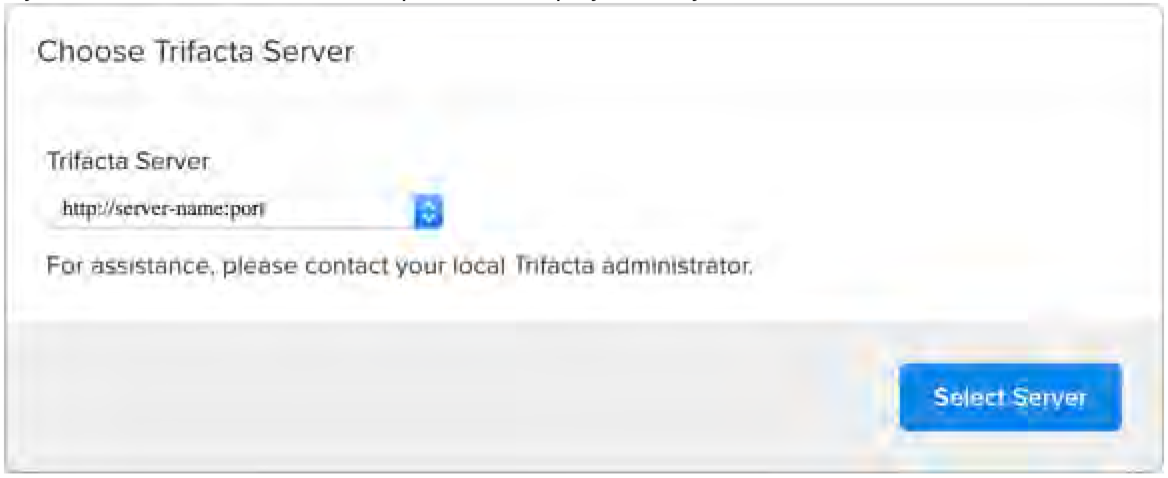
Launch the Application

Steps:

1. When installation is complete, double-click the application icon.
2. For the server, please enter the full URL including port number of the Trifacta instance to which you are connecting.
    1. By default, the server is available over port `3005`. For more information, please contact your IT administrator.
    2. If you connect to the Internet through a proxy, additional configuration is required. See *Configure Server Access through Proxy*.

        NOTE: If you make a mistake in specifying the URL to the server, please uninstall and reinstall the MSI. This step clears the local application cache, and you can enter the appropriate path through the application. See *Uninstall* below.
3. When the proper URL and port number are provided, you may launch the application.

4. If your environment contains multiple server deployments, you can select the one to which to connect:



For assistance, please contact your local Trifacta administrator.

Figure: Choose Server

5. Login with your Trifacta account. See *Login*.

Documentation Note

Unless specifically noted, all features described for Trifacta Wrangler Enterprise or the Trifacta application apply to the Wrangler Enterprise desktop application.

Uninstall

To uninstall from your Windows machine, use the Add or Remove Programs control panel.

Troubleshooting

Cannot connect to server

If you are unable to connect to the server, please do the following:

1. Verify that you are connecting to the appropriate URL.
    1. If you are connecting to the incorrect URL, please uninstall the application and re-install using the MSI file. See *Uninstall* above.
2. Verify if you need to connect to the server through a proxy server. If so, additional configuration is required. See *Configure Server Access through Proxy*.
3. Check your firewall settings.

"Does Not Support Your Browser" error

This error message indicates that you are trying to connect to an instance of the server that does not support the Wrangler Enterprise desktop application. Please verify that your connection URL is pointed to a supported instance of the server.

Start and Stop the Platform

Contents:

- *Debugging*
- *Troubleshooting*
    - *Error - SequelizeConnectionRefusedError: connect ECONNREFUSED*

Tip: The Restart Trifacta button in the Admin Settings page is the preferred method for restarting the platform.

NOTE: The restart button is not available when high availability is enabled for the Trifacta® node.

See *Admin Settings Page*.

Start

NOTE: These operations must be executed under the root user.

Command:

```
service trifacta start
```

Verify operations

Steps:

1. Check logs for errors:

    ```
    /opt/trifacta/logs/*.log
    ```

    1. You can also access logs through the Trifacta® application for each service. See *System Services and Logs*.
2. Login to the Trifacta application. If available, perform a simple transformation operation. See *Login*.
3. Run a simple job. See *Verify Operations*.

Restart

Command:

```
service trifacta restart
```

When the login page is available, the system has been restarted. See *Login*.

Tip: If you have made any configuration changes, you should verify operations. See *Verify Operations*.

Stop

Command:

```
service trifacta stop
```

Debugging

You can verify operations of WebHDFS. Command:

```
curl -i  "http://<hadoop_node>:<port_number>/webhdfs/v1/?
op=LISTSTATUS&user.name=trifacta"
```

Troubleshooting

Error - SequelizeConnectionRefusedError: connect ECONNREFUSED

If you have attempted to start the platform after an operating system reboot, you may receive the following error message, and the platform start fails to complete:

```
2016-10-04T14:03:17.883Z - error: [ENVIRONMENT] Environment Sanity Test
Failed
2016-10-04T14:03:17.883Z - error: [ENVIRONMENT] Exception Type: Error
2016-10-04T14:03:17.883Z - error: [ENVIRONMENT] Exception Message:
SequelizeConnectionRefusedError: connect ECONNREFUSED
```

Solution:

NOTE: This solution applies to PostgreSQL 9.6 only. Please modify for your installed database version.

This error can occur when the operating system is restarted. Please execute the following commands to check the PostgreSQL configuration and restart the databases.

```
chkconfig postgresql-9.6 on
```

Then, restart the platform as normal.

```
service trifacta restart
```

Login

NOTE: Administrators of the platform should change the default password for the admin account. See *Change Admin Password*.

To login to the Trifacta® application, navigate to the following in your browser:

`http://<host_name>:<port_number>`

where:

- `<host_name>` is the host of the Trifacta application.
- `<port_number>` is the port number to use. Default is `3005`.

If you do not have an account, click **Register**.

- If self-registration is enabled, you you may be able to immediately login after registering.
- If Kerberos or secure impersonation is enabled, an administrator must apply a Hadoop principal value to the account before you can login. Please contact your Trifacta administrator.
- System administrators can enable self-registration. See *Configure User Self-Registration*.

After you login, you are placed in the Flows page, where you can create and manage your datasets and flows. See *Flows Page*.

- If you are using S3 as your base storage layer and per-user authentication has been enabled, you must provide the AWS credentials to connect to your storage. From the left navigation bar, select **Settings > Storage** and then select the AWS option. See *Configure Your Access to S3*.

- For a basic walkthrough of the Trifacta application, see *Workflow Basics*.

To logout:

From the Settings menu, select **Logout**.

Install Reference

These appendices provide additional information during installation of Trifacta® Wrangler Enterprise.

Topics:

- *Install SSL Certificate*
- *Change Listening Port*
- *Supported Deployment Scenarios for Cloudera*
- *Supported Deployment Scenarios for Hortonworks*
- *Supported Deployment Scenarios for AWS*

- *Supported Deployment Scenarios for Azure*
- *Uninstall*

Install SSL Certificate

Contents:

You may optionally configure an SSL certificate to secure connections to the web application of the Trifacta® platform.

Pre-requisites

1. A valid SSL certificate for the FQDN where the Trifacta application is hosted
2. Root access to the Trifacta server
3. Trifacta platform is up and running

Configure nginx

There are two separate Nginx services on the server: one service for internal application use, and one service that functions as a proxy between users and the Trifacta application. To install the SSL certificate, all configuration are applied to the proxy process only.

Steps:

1. Log into the Trifacta server as the **centos** user. Switch to the **root** user:

   ```
   sudo su
   ```

2. Enable the proxy nginx service so that it starts on boot:

   ```
   systemctl enable nginx
   ```

3. Create a folder for the private key and limit access to it:

   ```
   sudo mkdir /etc/ssl/private/ && sudo chmod 700 /etc/ssl/private
   ```

4. Copy the following files to the server. If you copy and paste the content, please ensure that you do not miss characters or insert unwanted characters.

1. The `.key` file should go into the `/etc/ssl/private/` directory.
2. The .crt file and the CA bundle/intermediate certificate bundle should go into the `/etc/ssl/certs/` directory.

   > **NOTE:** The delivery name and format of these files varies by provider. Please verify with your provider's documentation if this is unclear.

3. Your certificate and the intermediate/authority certificate must be combined into one file for nginx. Here is an example of how to combine them together:

   ```
   cat example_com.crt bundle.crt >> ssl-bundle.crt
   ```

5. Update the permissions on these files. Modify the following filenames as necessary:

   ```
   sudo chmod 600 /etc/ssl/certs/ssl-bundle.crt
   sudo chmod 600 /etc/ssl/private/your-private-cert.key
   ```

6. Use the following commands to deploy the example SSL configuration file provided on the server:

   > **NOTE:** Below, some values are too long for a single line. Single lines that overflow to additional lines are marked with a \. The backslash should not be included if the line is used as input.

   ```
   cp /opt/trifacta/conf/ssl-nginx.conf.sample /etc/nginx/conf.d
   /trifacta.conf && \
   rm /etc/nginx/conf.d/default.conf
   ```

7. Edit the following file:

   ```
   /etc/nginx/conf.d/trifacta.conf
   ```

8. Please modify the following key directives at least:

Directive	Description
`server_name`	FQDN of the host, which must match the SSL certificate's Common Name
`ssl_certificate`	Path to the file of the certificate bundle that you created on the server. This value may not require modification.
`ssl_certificate_key`	Path to the .key file on the server.

Example file:

```
server {
  listen          443;
  ssl             on;
  server_name     EXAMPLE.CUSTOMER.COM;
  # Don't limit the size of client uploads.
  client_max_body_size 0;
  access_log      /var/log/nginx/ssl-access.log;
  error_log       /var/log/nginx/ssl-error.log;
  ssl_certificate      /etc/ssl/certs/ssl-bundle.crt;
  ssl_certificate_key  /etc/ssl/certs/EXAMPLE-NAME.key;
  ssl_protocols        SSLv3 TLSv1 TLSv1.1 TLSv1.2;
  ssl_ciphers RC4:HIGH:!aNULL:!MD5;
  ssl_prefer_server_ciphers on;
  keepalive_timeout    60;
  ssl_session_cache    shared:SSL:10m;
  ssl_session_timeout  10m;
  location / {
    proxy_pass  http://localhost:3005;
    proxy_next_upstream error timeout invalid_header http_500
http_502 http_503 http_504;
    proxy_set_header        Accept-Encoding   "";
    proxy_set_header        Host            $host;
    proxy_set_header        X-Real-IP       $remote_addr;
    proxy_set_header        X-Forwarded-For
$proxy_add_x_forwarded_for;
    proxy_set_header        X-Forwarded-Proto $scheme;
    add_header              Front-End-Https   on;
    proxy_http_version 1.1;
    proxy_set_header Upgrade $http_upgrade;
    proxy_set_header Connection "upgrade";
    proxy_set_header Host $host;
    proxy_redirect     off;
  }
  proxy_connect_timeout       6000;
  proxy_send_timeout          6000;
  proxy_read_timeout          6000;
  send_timeout                6000;
}
server {
  listen          80;
  return 301 https://$host$request_uri;
}
```

9. Save the file.
10. To apply the new configuration, start or restart the nginx service:

```
service nginx restart
```

Modify listening port for Trifacta platform

If you have changed the listening port as part of the above configuration change, then the `proxy.port` setting in Trifacta platform configuration must be updated. See *Change Listening Port*.

Add secure HTTP headers

If you have enabled SSL on the platform, you can optionally insert the following additional headers to all requests to the Trifacta node:

Header	Protocol	Required Parameters
X-XSS-Protection	HTTP and HTTPS	`proxy.securityHeaders.enabled=true`
X-Frame-Options	HTTP and HTTPS	`proxy.securityHeaders.enabled=true`
Strict-Transport-Security	HTTPS	`proxy.securityHeaders.enabled=true` and `proxy.securityHeaders.httpsHeaders=true`

NOTE: SSL must be enabled to apply these security headers.

Steps:

To add these headers to all requests, please apply the following change:

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 change its value to `true`:

```
"proxy.securityHeaders.httpsHeaders": false,
```

3. Save your changes and restart the platform.

Enable secure cookies

If you have enabled SSL on the platform, you can optionally enable the use of secure cookies.

NOTE: SSL must be enabled.

Steps:

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

```
"webapp.session.cookieSecureFlag": false,
```

3. Save your changes and restart the platform.

Troubleshooting

Problem - SELinux blocks proxy service from communicating with internal app service

If the Trifacta platform is installed on SELinux, the operating system blocks communications between the service that manages the proxy between users and the application and the service that manages internal application communications.

To determine if this problem is present, execute the following command:

```
sudo cat /var/log/audit/audit.log | grep nginx | grep denied
```

The problem is present if an error similar to the following is returned:

```
type=AVC msg=audit(1555533990.045:1826142): avc:  denied  { name_connect
} for  pid=25516 comm="nginx" dest=3005 scontext=system_u:system_r:
httpd_t:s0
```

For more information on this issue, see *https://www.nginx.com/blog/using-nginx-plus-with-selinux*.

Solution:

The solution is to enable the following network connection through the operating system:

```
sudo setsebool -P httpd_can_network_connect 1
```

Restart the platform.

Change Listening Port

If you need to change the listening port for the Trifacta® platform, please complete the following instructions.

Tip: This change most typically applies if you are enabling use of SSL. For more information, see *Install SSL Certificate*.

NOTE: By default, the platform listens on port `3005`. All client browsing devices must be configured to enable use of this port or any port number that you choose to use.

Steps:

1. Login to the Trifacta node as an admin.
2. Edit the following file:

   ```
   /opt/trifacta/conf/nginx.conf
   ```

3. Edit the following setting:

```
server {
  listen 3005;
  ...
```

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

```
"proxy.port": 3005,
```

7. Set this value to the same value you applied in `nginx.conf`.
8. Save your changes and restart the platform.

Supported Deployment Scenarios for Azure

Contents:

Azure Deployment Scenarios

The following are the Azure deployment scenarios.

Deployment Scenario	Trifacta node installation	**Base Storage Layer**	**Storage - WASB**	**Storage - ADLS**	**Storage - Azure SQL DB**	**Storage - SQL DW**	**Cluster**
Trifacta® Wrangler Enterprise install for WASB	Azure	WASB	read/write	read only	read	read/write	• HDI • Azure Databricks
Trifacta Wrangler Enterprise install for ADLS	Azure	HDFS	read only	read/write	read	read/write	• HDI • Azure Databricks

Legend and Notes:

Column	**Notes**
Deployment Scenario	Description of the Azure-connected deployment
Trifacta node installation	Location where the Trifacta node is installed in this scenario.

Base Storage Layer	When the Trifacta platform is first installed, the base storage layer must be set. **NOTE:** After you have begun using the product, you cannot change the base storage layer. For more information, see *Set Base Storage Layer*.
Storage - WASB	For read/write access to WASB, the base storage layer must be set to WASB. For more information, see *Enable WASB Access*.
Storage - ADLS	For read/write access to ADLS, the base storage layer must be set to HDFS. For more information, see *Enable ADLS Access*.
Storage - Azure SQL DB	For Azure installs, you can optionally create a connection to an Azure SQL Database. For more information, see *Create Azure SQL Database Connections*.
Storage - SQL DW	For Azure installs, you can optionally create a connection to an Azure-hosted instance of SQL DW. For more information, see *Create SQL DW Connections*.
Cluster	List of Hadoop cluster types that are supported for integration and job execution at scale. • The Trifacta platform can integrate with at most one cluster. It cannot integrate with two different clusters at the same time. • Smaller jobs can be executed on the Trifacta Photon running environment, which is hosted on the Trifacta node itself. • For more information, see *Running Environment Options*.

Azure Installations

For more information, see *Install for Azure*.

Azure Integrations

The following table describes the different Azure components that can host or integrate with the Trifacta platform. Combinations of one or more of these items constitute one of the deployment scenarios listed in the following section.

Azure Service	Description	Base Storage Layer	Other Required Azure Services
HDI	Microsoft Azure deployments can integrate with an HDI cluster, which can be pre-existing or created at the time of deployment. See *Configure for HDInsight*.	Base storage layer can be HDFS (for ADLS) or WASB.	
Azure Databricks	Optionally, you can integrate the Trifacta platform with an Azure Databricks cluster. See *Configure for Azure Databricks*.	Base storage layer can be HDFS (for ADLS) or WASB.	
WASB	Windows Azure Storage Blobs (WASB) extends HDFS to enable access to storage blobs that have not been deployed into the HDI cluster. See *Enable WASB Access*.	Base Storage Layer = WASB	HDI cluster WASB Key Vault

ADLS	Active Data Lake Store (ADLS) provides a highly scalable file-based storage system within HDI cluster. See *Enable ADLS Access*.	Base Storage Layer = HDFS	HDI cluster ADLS Key Vault

The following database connections are optional.

Database Name	Description
Hive	You can read from and write to Hive, a data warehouse built on top of HDI. • For usage information, see *Using Hive*. • To create the connection, see *Create Hive Connections*.
SQL DW	You can read from and write to SQL Data Warehouse, a scalable data warehouse solution for Azure. • For usage information, see *Using SQL DW*. • To create the connection, see *Create SQL DW Connections*.
SQL DB	You can read from SQL DB, a SQL Server variant for Azure. • For usage information, see *Using Databases*. • To create the connection, see *Create Azure SQL Database Connections*.

Uninstall

To remove Trifacta® Wrangler Enterprise, execute as root user one of the following commands on the Trifacta node.

NOTE: All platform and cluster configuration files are preserved. User metadata is preserved in the Trifact a database.

CentOS/RHEL:

```
sudo rpm -e trifacta
```

Ubuntu:

```
sudo apt-get remove trifacta
```

Configure for Azure

Contents:

Please complete the following steps in the listed order to configure your installed instance of the Trifacta® platform to integrate with an HDInsight cluster.

Pre-requisites

1. Deploy HDI cluster and Trifacta node.

    NOTE: The HDI 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 Trifacta platform on the node.

For more information, see *Install for Azure*.

Configure Azure

Create registered application

You must create a Azure Active Directory (AAD) application and grant it the desired access permissions, such as read/write access to the ADLS resource and read/write access to the Azure Key Vault secrets .

This service principal is used by the 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 Trifacta platform configuration: `azure.applicationid`.
Service User Key	Create a key for the Registered app in the Azure Portal.	Applied to Trifacta platform configuration: `azure.secret`.
Directory ID	Copy the Directory ID from the Properties blade of Azure Active Directory.	Applied to Trifacta platform configuration: `azure.directoryId`.

These properties are applied later in the configuration process.

Configure the Platform

Configure for HDI

If you are integrating the Trifacta platform with a pre-existing HDI cluster, additional configuration is required. See *Configure for HDInsight*.

> **NOTE:** If you created a new HDI cluster as part of the installation, all required is listed below.

Configure for Azure Databricks

Optionally, you can integrate the Trifacta platform with Azure Databricks, instead of HDI.

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	hdfs.protocolOverride setting
WASB	`wasbs`	(empty)
ADLS	`hdfs`	`adl`

See *Set Base Storage Layer*.

Configure for Key Vault

For authentication purposes, the 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

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:
    1. Name: Provide a reasonable name for the resource. Example:

        ```
        <clusterName>-<applicationName>-<group/organizationName>
        ```

    2. Location: Pick the location used by the HDI cluster.
    3. For other fields, add appropriate information based on your enterprise's preferences.
4. To create the resource, click **Create**.

Enable Key Vault access for the Trifacta platform

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 New**.
4. For Secret permissions, select the following:
    1. Get
    2. Set
    3. Delete
5. Do not select any other options.
6. Click **OK**.

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 Trifacta platform.

Configure Key Vault key and secret for WASB

In the Key Vault, you can create key and secret pairs for use.

Base Storage Layer	Description
ADLS	The Trifacta platform creates its own key-secret combinations in the Key Vault. No additional configuration is required. Please skip this section and populate the Key Vault URL into the Tr ifacta platform.
WASB	For WASB, you must create key and secret values that match other values in your Azure configuration. Instructions are below.

WASB: 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 `sasTokenId` in the Trifacta platform.
secret	The value of the secret should match the shared access signature for your storage. This value is specified as `sasToken` in the Trif acta 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 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:
    1. Select **Manual** for upload options.
    2. Chose an appropriate name for the key.

        NOTE: Please retain the name of the key for later use, when it is applied through the Trifact a platform as the `sasTokenId` value. Instructions are provided later.

    3. Paste the SAS token value for the key into the secret field.
    4. Click **Create**.

Configure Key Vault location

For ADLS or WASB, the location of the Azure Key Vault must be specified for the 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 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 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 Trifacta platform. To use STS with Azure, the following properties must be specified.

> **NOTE:** Except in rare cases, the other properties for secure token service do not need to be modified.

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

Property	Description
" secure-token-service .autorestart"	Set this value to `true` to enable auto-restarting of the secure token service.
"secure-token-service.port"	Set this value to `8090`.
"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.
"secure-token-service.userIdHashingPepper"	Enter a base64 string.

Configure for SSO

If needed, you can integrate the Trifacta platform with Azure AD for Single-Sign On to the platform. See *Configure SSO for Azure AD*.

Configure for ADLS

Enable read-only or read-write access to ADLS. For more information, see *Enable ADLS Access*.

Configure for WASB

Enable read-only or read-write access to WASB. For more information on integrating with WASB, see *Enable WASB Access*.

Configure relational connections

If you are integrating Trifacta Wrangler Enterprise 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 Hive connection

You can create a connection to the Hive instance on the HDI cluster with some modifications.

- **High Availability:** Natively, Azure supports high availability for HiveServer2 via Zookeeper. Host and port information in the JDBC URL must be replaced with a Zookeeper quorum.

In addition to the other Hive connection properties, please specify the following values for the properties listed below:

Property	Description
Host	Use your Zookeeper quorum value. For the final node of the list, omit the port number. Example: `zk1.cloudapp.net:2181,zk2.cloudapp.net:2181,zk3.cloudapp.net`
Port	Set this value to `2181`.
Connect String options	In addition to any options required for your environment, include the following option: `/;serviceDiscoveryMode=zooKeeper;zooKeeperNamespace=hiveserver2`
Database	Enter your Hive database name.

Connections are created through the Connections page. See *Connections Page*.

For additional details on creating a connection to Hive, see *Create Hive Connections*.

A Hive connection can also be created using the above property substitutions via programmatic methods.

- For details on values to use, see *Connection Types*.
- See *API Connections Create v4*.

Create Azure SQL Database connection

For more information, see *Create Azure SQL Database Connections*.

Create Azure SQL DW connection

For more information, see *Create SQL DW Connections*.

Testing

1. Load a dataset from the HDI cluster through either ADLS or WASB.
2. Perform a few simple steps on the dataset.
3. Click **Run Job** in the Transformer page.

4. When specifying the job:
    1. Click the Profile Results checkbox.
    2. Select **Spark**.
5. When the job completes, verify that the results have been written to the appropriate location.

Configure for HDInsight

Contents:

When deployed to Microsoft Azure, the Trifacta® platform must be integrated with Microsoft HDInsight, a Hadoop-based platform for data storage and analytics. This section describes the configuration steps required to integrate with a pre-existing HDI cluster. This section applies only if you have installed the Trifacta platform onto a node of a pre-existing HDI cluster.

If you created a new HDI cluster as part of your deployment of the platform from the Azure Marketplace, please skip this section. You may use it as reference in the future.

Supported Versions

Supported Versions: This release supports integration with HDI 3.5 and HDI 3.6.

Limitations

For this release, the following limitations apply:

- HDI does not support the client-server web sockets configuration used by the platform. This limitation results in diminished suggestions prompted by platform activities.

Pre-requisites

This section makes the following assumptions:

1. You have installed and configured the Trifacta platform onto an edge node of a pre-existing HDI cluster.
2. You have installed WebWASB on the platform edge node.

Before You Begin

Create Trifacta user account on HDI cluster

The Trifacta platform interacts with the cluster through a single system user account. A user for the platform must be added to the cluster.

UserID:

If possible, please create the user ID (`[hdi.user]`) as: `trifacta`.

This user must be created on each data node in the cluster.

This user should belong to the group (`[hdi.group]`): `trifactausers`.

User requirements:

- (if integrating with WASB) Access to WASB
- Permission to run YARN jobs on the cluster.

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 cluster paths in the following locations:

   ```
   "hdfs.pathsConfig.fileUpload": "/trifacta/uploads",
   "hdfs.pathsConfig.dictionaries": "/trifacta/dictionaries",
   "hdfs.pathsConfig.batchResults": "/trifacta/queryResults",
   ```

   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 Trifact a application can destroy other users' data. Please use the tools provided through the interface for managing uploads to WASB.

   Individual users can configure the output directory where exported results are stored. See *Storage Config Page*.
3. Save your changes.

Acquire cluster configuration files

You must acquire the configuration files from the HDI cluster and apply them to the Trifacta node.

Tip: Later, you configure the platform settings for accessing various components in the cluster. This host, port, and other information is available through these cluster configuration files.

Steps:

1. On any edge node of the cluster, acquire the .XML files from the following directory:

   ```
   /etc/hadoop/conf
   ```

   > **NOTE:** If you are integrating with an instance of Hive on the cluster, you must also acquire the Hive configuration file: `/etc/hive/conf/hive-site.xml`.

2. These files must be copied to the following directory on the Trifacta node:

   ```
   /trifacta/conf/hadoop-site
   ```

3. Replace any existing files with these files.

Acquire build build number

You must acquire the full version and build number of the underlying Hortonworks distribution. On any of the cluster nodes, navigate to `/usr/hdp`. The version and build number is referenced as a directory in this location, named in the following form:

```
A.B.C.D-X
```

For the rest of the configuration, the sample values for HDI 3.6 are referenced. Use the appropriate values for your distribution.

Supported HDInsight Distribution	Short Hortonworks Version	Example Full Hortonworks Version
3.5	2.5	2.5.6.2-9
3.6	2.6	2.6.2.2-5

Configure the HDI Cluster

The following configuration sections must be reviewed and completed.

Specify Storage Layer

In the Azure console, you must specify and retain the type of storage to apply. In the Storage tab of the cluster specification, the following storage layers are supported.

> **NOTE:** After the base storage layer has been defined in the Trifacta platform, it cannot be changed. Reinstallation is required.

NOTE: If possible, you should reserve a dedicated cluster for the Trifacta platform processing. If there is a mismatch between the storage layer of your existing HDI cluster and the required storage for your Trifacta deployment, you can create a new HDI cluster as part of the installation process. For more information, see *Install for Azure*.

Tip: During installation of the Trifacta platform, you must define the base storage layer. Retain your selection of the Azure Storage Layer and its mapped based storage layer for the Trifacta platform installation.

Azure Storage Layer	**Description**	Trifacta Base Storage Layer
`Azure Storage`	Azure storage leverages WASB, an astraction layer on top of HDFS.	`wasbs`
`Data Lake Store`	Data Lake Store maps to ADLS in the Trifacta platform.	`hdfs`

Specify Protocol

In the Ambari console, you must specify the communication protocol to use in the cluster.

NOTE: The cluster protocol must match the protocol in use by the Trifacta platform.

Steps:

1. In the Ambari console, please migrate to the following location: **HDFS > Configs > Advanced > Advanced Core Site > fs.defaultFS**.
2. Set the value according to the following table:

Azure Storage Layer	**Protocol (fs.defaultFS) value**	Trifacta platform config value	**Link**
`Azure Storage`	`wasbs://<containername>@<accountname>.blob.core.windows.net`	`"webapp.storageProtocol": "wasbs",`	See *Set Base Storage Layer*.
`Data Lake Store`	`adl://home`	`"webapp.storageProtocol": "hdfs",`	See *Set Base Storage Layer*.

3. Save your changes.

Define Script Action for domain-joined clusters

If you are integrating with a domain-joined cluster, you must specify a script action to set some permissions on cluster directories.

For more information, see *https://docs.microsoft.com/en-us/azure/hdinsight/domain-joined/apache-domain-joined-configure-using-azure-adds*.

Steps:

1. In the Advanced Settings tab of the cluster specification, click **Script actions**.
2. In the textbox, insert the following URL:

```
https://raw.githubusercontent.com/trifacta/azure-deploy/master/bin
/set-key-permissions.sh
```

3. Save your changes.

Install Software

If you haven't done so already, you can install the Trifacta software into an edge node of the HDI cluster. For more information, see *Install on CentOS and RHEL*.

Configure the Platform

These changes must be applied after the Trifacta platformhas been installed.

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

Specify Trifacta user

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

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

Specify location of client distribution bundle JAR

The 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:

`/trifacta/hadoop-deps`

Configure the bundle distribution property (`hadoopBundleJar`):

```
  "hadoopBundleJar": "hadoop-deps/hdp-2.6/build/libs/hdp-2.6-bundle.jar"
```

Configure component settings

For each of the following components, please explicitly set the following settings.

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 Batch Job Runner:

```
  "batch-job-runner": {
   "autoRestart": true,
    ...
    "classpath": "%(topOfTree)s/hadoop-data/build/install/hadoop-
data/hadoop-data.jar:%(topOfTree)s/hadoop-data/build/install/hadoop-
data/lib/*:%(topOfTree)s/conf/hadoop-site:/usr/hdp/current/hadoop-
client/hadoop-azure.jar:/usr/hdp/current/hadoop-client/lib/azure-
storage-2.2.0.jar"
  },
```

3. Configure the following environment variables:

```
"env.PATH": "${HOME}/bin:$PATH:/usr/local/bin:/usr/lib/zookeeper
/bin",
"env.TRIFACTA_CONF": "/opt/trifacta/conf"
"env.JAVA_HOME": "/usr/lib/jvm/java-1.8.0-openjdk-amd64",
```

4. Configure the following properties for various Trifacta components:

```
  "ml-service": {
   "autoRestart": true
  },
  "monitor": {
   "autoRestart": true,
    ...
   "port": <your_cluster_monitor_port>
  },
  "proxy": {
   "autoRestart": true
  },
  "udf-service": {
   "autoRestart": true
  },
  "webapp": {
    "autoRestart": true
  },
```

5. Disable S3 access:

```
"aws.s3.enabled": false,
```

6. Configure the following Spark Job Service properties:

```
"spark-job-service.classpath": "%(topOfTree)s/services/spark-job-
server/server/build/libs/spark-job-server-bundle.jar:%(topOfTree)s
/conf/hadoop-site/:%(topOfTree)s/services/spark-job-server/build
/bundle/*:/usr/hdp/current/hadoop-client/hadoop-azure.jar:/usr/hdp
/current/hadoop-client/lib/azure-storage-2.2.0.jar",
"spark-job-service.env.SPARK_DIST_CLASSPATH": "/usr/hdp/current
/hadoop-client/*:/usr/hdp/current/hadoop-mapreduce-client/*",
```

7. Save your changes.

Configure High Availability

If you are integrating the platform the HDI cluster with high availability enabled, please complete the following steps so that the platform is aware of the failover nodes

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. Enable high availability feature on the namenode and resourceManager nodes:

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

3. For each YARN resource manager, you must configure its high availability settings. The following are two example node configurations, including the default port numbers for HDI:

> **Tip:** Host and port settings should be available in the cluster configuration files you copied to the Trifacta node. Or you can acquire the settings through the cluster's admin console.

```
    "yarn": {
      "highAvailability": {
        "resourceManagers": {
          "rm1": {
            "port": <your_cluster_rm1_port>,
            "schedulerPort": <your_cluster_rm1_scheduler_port>,
            "adminPort": <your_cluster_rm1_admin_port>,
            "webappPort": <your_cluster_rm1_webapp_port>
          },
          "rm2": {
            "port": <your_cluster_rm2_port>,
            "schedulerPort": <your_cluster_rm2_scheduler_port>,
            "adminPort": <your_cluster_rm2_admin_port>,
            "webappPort": <your_cluster_rm2_webapp_port>
          }
        }
      }
    },
```

4. Configure the high availability namenodes. The following example configures two namenodes (nn1 and nn2), including the default port numbers for HDI:

> **Tip:** Host and port settings should be available in the cluster configuration files you copied to the Tr ifacta node. Or you can acquire the settings through the cluster's admin console.

```
    "hdfs": {
      ...
      "highAvailability": {
        "namenodes": {
          "nn1": {
            "port": <your_cluster_namenode1_port>
          },
          "nn2": {
            "port": <your_cluster_namenode2_port>
          }
        }
      ...
```

5. Save your changes.

> **NOTE:** If you are deploying high availability failover, you must use HttpFS, instead of WebHDFS, for communicating with HDI. Additional configuration is required. HA in a Kerberized environment for HDI is not supported. See *Enable Integration with Cluster High Availability*.

Create Hive connection

Limitations:

1. The platform only supports HTTP connections to Hive on HDI. TCP connections are not supported.
2. The Hive port must be set to `10001` for HTTP.

For more information, see *Create Hive Connections*.

Hive integration requires additional configuration.

> **NOTE:** Natively, HDI supports high availability for Hive via a Zookeeper quorum.

For more information, see *Configure for Hive* .

Configure for Spark Profiling

If you are using Spark for profiling, you must add environment properties to your cluster configuration. See *Configure for Spark*.

Configure for UDFs

If you are using user-defined functions (UDFs) on your HDInsight cluster, additional configuration is required. See *Java UDFs*.

Configure Storage

Before you begin running jobs, you must specify your base storage layer, which can be WASB or ADLS. For more information, see *Set Base Storage Layer*.

Additional configuration is required:

- For more information, see *Enable WASB Access*.
- For more information, see *Enable ADLS Access*.

Configure for Azure Databricks

Contents:

This section describes how to configure the 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:** For each user, a separate cluster is created. It may take a few minutes to spin up a new cluster.

Pre-requisites

- The Trifacta platform must be deployed in Microsoft Azure.

Limitations

- Nested folders are not supported when 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 only.

  > **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.

Job counts

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`.
- 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`.

> **NOTE:** To enable retrieval and auditing of job information after a job has been completed, the 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. You should periodically delete jobs on your Azure Databricks cluster to prevent reaching these limits and receiving a `Quota for number of jobs has been reached` limit.

For more information, see *https://docs.databricks.com/user-guide/jobs.html*.

Enable

To enable Azure Databricks, please perform the following configuration changes.

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. Set them to the values listed below, which enable the Trifacta Photon (sm aller jobs) and Azure Databricks (small to extra-large jobs) running environments:

```
"webapp.runInTrifactaServer": true,
"webapp.runInDatabricks": true,
"webapp.runInHadoop": false,
"webapp.runinEMR": false,
"webapp.runInDataflow": false,
"photon.enabled": true,
```

3. Do not save your changes until you have completed the following configuration section.

Configure

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
feature.parameterization. maxNumberOfFilesForExecution. databricksSpark	Maximum number of parameterized source files that are permitted to be executed as part of an Azure Databricks job.	
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.	Default: `Standard_D3_v2` For more information, see the sizing guide for Azure Databricks.
databricks.sparkVersion	Azure Databricks cluster version which also includes the Spark Version.	Please do not change unless you are using a non-default version of Spark. For more information, see *Configure for Spark*.
databricks.serviceUrl	URL to the Azure Databricks Service where Spark jobs will be run (Example: *https://westus2.azuredatabricks.net*)	
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	Minimum value: `1` Increasing this value can increase compute costs.
databricks.maxWorkers	Maximum number of Worker nodes the cluster can create during auto scaling	Minimum value: Not less than `databricks.minWorkers`. Increasing this value can increase compute costs.
databricks.logsDestination	DBFS location that cluster logs will be sent to every 5 minutes	Leave this value as `/trifacta/logs`.
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 `true`.

databricks.driverNodeType	Type of node to use for the Azure Databricks Driver. There is only 1 Driver node per cluster.	Default: `Standard_D3_v2` For more information, see the sizing guide for Databricks.
databricks. clusterStatePollerDelayInSeconds	Number of seconds to wait between polls for Azure 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 Azure Databricks job	
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	
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.
spark.useVendorSparkLibraries	When `true`, the platform bypasses shipping its installed Spark libraries to the cluster with each job's execution.	Default is `false`. Do not modify unless you are experiencing failures in Azure Databricks job execution. For more information, see Troubleshooting below.

Configure Personal Access Token

After the above configuration has been performed, each user must insert their personal access token into their User Settings page. This configuration enables the user to authenticate using the Azure Databricks REST APIs, which enables the execution of jobs.

NOTE: Each user must apply a personal access token to their User Profile. Users that do not provide a personal authentication token cannot run jobs on Azure Databricks, including transformation, sampling, and profiling jobs.

Steps:

1. Acquire your Azure Databricks personal access token. For more information, see *https://docs.azuredatabricks.net/api/latest/authentication.html#requirements*.
2. Login to the application. From the menu bar, select **Settings menu > Settings**.
3. Click **Databricks**.

4. In the Databricks Personal Access Token field, paste your token.

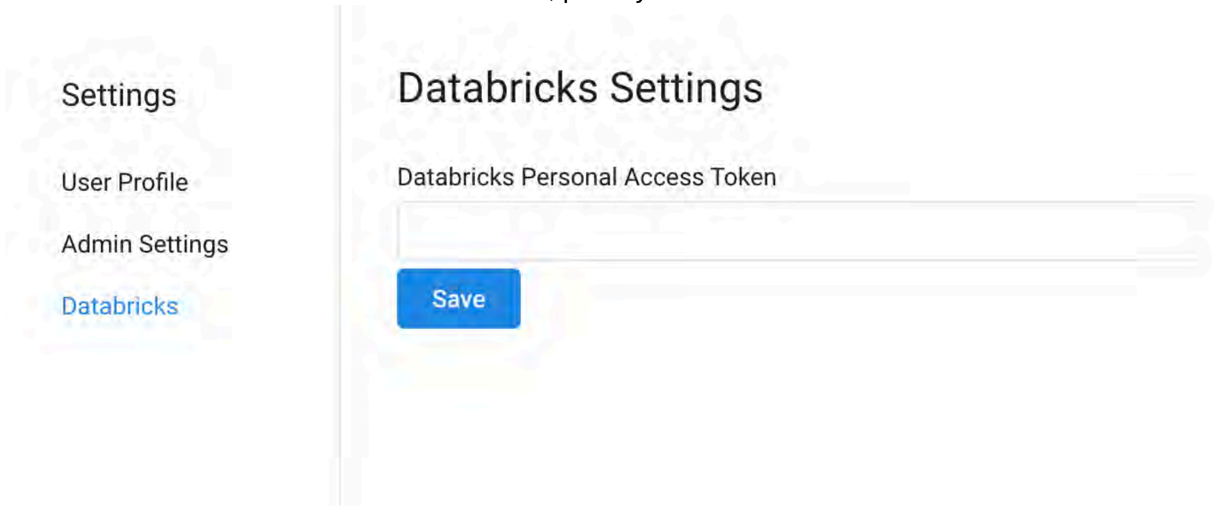


Figure: Databricks user configuration

5. Click **Save.**

Azure Databricks personal access tokens are saved in the Azure key vault.

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 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 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 `true` 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.keyVaultApplicationSecretSecretName	Specify the name of the Key Vault secret that holds the service principal secret.

Save your changes.

Use

Run job from application

When the above configuration has been completed, you can select the running environment through the application. See *Run Job Page*.

Run job via CLI

You can run jobs on Azure Databricks via CLI. When executing, the `job_type` parameter must be set to `databricksSpark`. See *CLI for Jobs*.

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 *API JobGroups Create v4*.

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 Trifacta platform configuration:
   1. Set `databricks.sparkVersion` to the new version to use.
   2. Set `spark.version` to the appropriate version of Spark to use.
5. Restart the 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.

Enable ADLS Access

Contents:

By default, Microsoft Azure deployments integrate with Azure Data Lake Store (ADLS). Optionally, you can configure your deployment to integrate with WASB.

- **Microsof Azure Data Lake Store (ADLS)** is a scalable repository for big data analytics. ADLS is accessible from Microsoft HDI. For more information, see *https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-overview*.

Limitations of ADLS Integration

- In this release, the 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.

NOTE: To enable read-only access to ADLS, do not set the base storage layer to `hdfs`. The base storage layer for ADLS read-write access must remain `wasbs`.

Pre-requisites

General

- The Trifacta platform has already been installed and integrated with an Azure HDI cluster. See *Configure for HDInsight*.
- HDFS must be set as the base storage layer for the 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 ADLS, you must create the 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 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	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: • `azure.applicationId` • `azure.secret` • `azure.directoryId` 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. For more information on these settings, see *Configure for Azure*.
User	Per-user mode allows individual users to authenticate to ADLS through their Azure Active Directory login. **NOTE:** Additional configuration for AD SSO is required. Details are below.

Steps:

Please complete 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 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 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 Trifacta platform

Define default storage location and access key

In platform configuration, you must define the following properties:

```
"azure.adl.store": "<your_value_here>",
```

This property defines the ADLS storage to which all output data is delivered. Example:

```
adl://<YOUR_STORE_NAME>.azuredatalakestore.net
```

Per earlier configuration:

- `webapp.storageProtocol` must be set to `hdfs`.
- `hdfs.protocolOverride` must be set to `adl`.

Configure HDFS properties

In the Trifacta platform, you must configure the following properties for effective communication with HDFS.

```
"hdfs": {
 "username": "[hadoop.user]",
 "enabled": true,
 "webhdfs": {
  "httpfs": false,
  "maprCompatibilityMode": false,
  "ssl": {
   "enabled": true,
   "certificateValidationRequired": false,
   "certificatePath": "<YOUR_PATH_HERE>"
  },
  "host": "[ADLS].azuredatalakestore.net",
  "version": "/webhdfs/v1",
  "proxy": {
   "host": "proxy",
   "enabled": false,
   "port": 8080
  },
  "credentials": {
   "username": "[hadoop.user]",
   "password": ""
  },
  "port": 443
 },
 "protocolOverride": "adl",
 "highAvailability": {
  "serviceName": "[ADLS].azuredatalakestore.net",
  "namenodes": {}
 },
 "namenode": {
  "host": "[ADLS].azuredatalakestore.net",
  "port": 443
 }
}
```

Property	Description
`hdfs.username`	Set this value to the name of the user that the Trifacta platform uses to access the cluster.
`hdfs.enabled`	Set to `true`.
`hdfs.webhdfs.httpfs`	Use of HttpFS in this integration is not supported. Set this value to `false`.
`hdfs.webhdfs.maprCompatibilityMode`	This setting does not apply to ADLS. Set this value to `false`.
`hdfs.webhdfs.ssl.enabled`	SSL is always used for ADLS. Set this value to `true`.
`hdfs.webhdfs.ssl.certificateValidationRequired`	Set this value to `false` .
`hdfs.webhdfs.ssl.certificatePath`	This value is not used for ADLS.
`hdfs.webhdfs.host`	Set this value to the address of your ADLS datastore.

`hdfs.webhdfs.version`	Set this value to `/webhdfs/v1`.
`hdfs.webhdfs.proxy.host`	This value is not used for ADLS.
`hdfs.webhdfs.proxy.enabled`	A proxy is not used for ADLS. Set this value to `false` .
`hdfs.webhdfs.proxy.port`	This value is not used for ADLS.
`hdfs.webhdfs.credentials.username`	Set this value to the name of the user that the Trifacta platform uses to access the cluster.
`hdfs.webhdfs.credentials.password`	Leave this value empty for ADLS.
`hdfs.webhdfs.port`	Set this value to `443`.
`hdfs.protocolOverride`	Set this value to `adl`.
`hdfs.highAvailability.serviceName`	Set this value to the address of your ADLS datastore.
`hdfs.highAvailability.namenodes`	Set this value to an empty value.
`hdfs.namenode.host`	Set this value to the address of your ADLS datastore.
`hdfs.namenode.port`	Set this value to `443`.

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 change its value to `true`:

```
"azure.adl.enabled": true,
```

3. Save your changes.

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.

Try running a simple job from the Trifacta application. For more information, see *Verify Operations*.

- See *ADLS Browser*.
- See *Using ADLS*.

Enable WASB Access

Contents:

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.

Limitations of WASB Integration

- If a directory is created on the HDI cluster through WASB, the directory includes a Size=0 blob. The Trifact a 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, you can follow these instructions to set up read-only access to WASB.

NOTE: If you are adding WASB as a secondary integration to ADLS, 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 for ADLS read-write access must remain `hdfs`.

Pre-requisites

General

- The Trifacta platform has already been installed and integrated with an Azure HDI cluster. See *Configure for HDInsight*.
- WASB must be set as the base storage layer for the 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 ADLS, you must create the 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 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 Trif acta 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 Trifacta platform

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

whether SAS token is to be retrieved from Azure Key Vault or 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*.

Secret names used for extra stores

If you are enabling extra WASB stores, specify the secret name to be used to access the SAS token from the Key Vault per extra Store.

> **NOTE:** Additional configuration is required for enabling extra WASB stores. See below.

```
"azure.wasb.extraStores": [ {
        ...
        "keyVaultSasTokenSecretName": "<secret_name>"
  }, {
        ...
        "keyVaultSasTokenSecretName": "<secret_name>"
  }
]
```

Parameter	Description
`"azure.wasb.fetchSasTokensFromKeyVault": true,`	Instructs the Trifacta platform to query the Key Vault for SAS tokens **NOTE:** The Key Vault must already be set up. See "Key Vault Setup" above.
`"azure.wasb.defaultStore.keyVaultSasTokenSecretName": "<your_value_here>",`	The default store's SAS token secret name to retrieve the SAS token for the default store from the Azure Key Value Store.

SAS token from Trifacta configuration

To specify the SAS token in the Trifacta platform configuration, set the following flag to false and then specify the SAS token per container.

Parameter	Description
`"azure.wasb.fetchSasTokensFromKeyVault": false,`	Instructs the Trifacta platform to acquire per-container SAS tokens from the platform configuration.

`"azure.wasb.defaultStore.sasToken": "<your_value_here>",`	Specify the SAS token here for the default store, if `azure.wasb.fetchSasTokensFromKeyVault` is set to `false`.

SAS token for extra WASB stores

If you are enabling extra WASB stores and `azure.wasb.fetchSasTokensFromKeyVault` is set to `false`, specify the sasToken for each extra store.

NOTE: Additional configuration is required for enabling extra WASB stores. See below.

```
"azure.wasb.extraStores": [ {
        ...
        "sasToken": "<your_value_here>"
  }, {
        ...
        "sasToken": "<your_value_here>"
  }
]
```

Define default storage location and access key

In platform configuration, you must define the following properties. When these properties are specified, the platform acquires the secret for the specified token ID, which is used to gain access to WASB.

Storage account

Azure path to the location where your data is to be stored.

```
"azure.wasb.defaultStore.blobHost": "<your_value_here>",
```

Container

Within your storage location, this value defines the default container for storing data.

```
"azure.wasb.defaultStore.container": "<your_value_here>",
```

Define extra stores

If you have additional WASB stores, you can specify access to them for the Trifacta platform. Users of the platform can use 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`. Some of these settings may not be available through the *Admin Settings Page*. For more information, see *Platform Configuration Methods*.
2. Locate the `azure.wasb.extraStores` configuration block and add the following parameters:

```
"azure.wasb.extraStores":
    "extraStores": [
     {
      "sasToken": "<VALUE1_HERE",
      "keyVaultSasTokenSecretName": "<VALUE1_HERE>",
      "container": "<VALUE1_HERE>",
      "blobHost": "<VALUE1_HERE>"
     },
     {
      "sasToken": "VALUE2_HERE",
      "keyVaultSasTokenSecretName": "<VALUE2_HERE>",
      "container": "<VALUE2_HERE>",
      "blobHost": "<VALUE2_HERE>"
     }
    ]
   },
 },
```

Parameter	Description
`sasToken`	Set this value to `SAS token`, if applicable.
`keyVaultSasTokenSecretName`	To use the same SAS token as used in default storage, set this value to the same SAS token ID. If needed, you can generate and apply a per-container SAS token for use in this field for this specific store. Details are below.
`container`	Apply the name of the WASB container. **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.
`blobHost`	Specify the blob host for the extra store. **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.

3. Save your changes and restart the platform.

Generate per-container SAS token

Execute the following command at the command line to generate a SAS token for a specific container:

```
Set-AzureRmStorageAccount -Name 'name'
$sasToken = New-AzureStorageContainerSASToken -Permission r -ExpiryTime
(Get-Date).AddHours(2.0) -Name '<container_name>'
```

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. Save your changes and restart the platform.

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 change its value to `true`:

   ```
   "azure.wasb.enabled": true,
   ```

3. Save your changes and restart the platform.

Testing

Restart services. See *Start and Stop the Platform*.

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 Trifacta application. For more information, see *Verify Operations*.

- See *WASB Browser*.
- See *Using WASB*.

Configure SSO for Azure AD

Contents:

When the 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*.
- When enabled, SSO also applies to the Wrangler Enterprise desktop application, if it is installed.

Supported authentication models:

Users can authenticate with the 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 ADLS.

Domain-Joined Clusters: Using Azure AD, the Trifacta platform can be deployed to a domain-joined HDInsight cluster and can run jobs as the authenticated AD user via secure impersonation. For more information, see *Configure for HDInsight*.

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.

Pre-Requisites

1. You have installed the 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 Trifacta platform must be registered as a Service Provider in your Azure AD tenant.
5. Please acquire the following Service Provider properties:
    1. 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 Trifacta platform as part of the basic Azure configuration.

        NOTE: The 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.

    2. The Service Provider Reply URL provides the redirect URL after the user has authenticated with Azure AD.

3. 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 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 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
azure.sso.enabled	Set this value to `true` to enable Azure AD Single Sign-On. The Trifacta platform authenticates users through enterprise Azure AD.
azure.sso.redirectUrl	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: `https://<trifacta-app-host>/sign-in/azureCallback`
azure.sso.allowHttpForRedirectUrl	When `true`, the `redirectUrl` can be specified as an insecure, non-HTTPS value. Default is `false`.
azure.sso.enableAutoRegistration	Set this value to `true` to enable SSO users to automatically register and login to the Trifacta application when they connect.

azure.resourceURL	This value defines the Azure AD resource for which to obtain an access token. **NOTE:** By default, this value is `https://graph.windows.net/` . You can select other values from the drop-down in the Admin Settings page. When using Azure Data Lake: 1. In the Azure Portal, grant to the Trifacta application ID the Azure Data Lake API permission. 2. Set this value to `https://datalake.azure.net/` . 3. Sign out of the Trifacta application and sign in again.

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 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
`webapp.sso.enableAutoRegistration`	This property has no effect in Azure.
`azure.sso.enableAutoRegistration`	Set this value to `true`. 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 Admin Settings page, with the following provisions:

- The Trifacta platform does not recheck for attribute values on each login. If attribute values change in LDAP, they must be updated in the User Management page, 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 through the User Management area.

For more information, see *Manage Users*.

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 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 Trifac ta application. Otherwise, the user can still access the Trifacta application until the user's access token expires.

For more information on disabling user accounts, see *Manage Users*.

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:

You can extend the basic SSO integration between the Trifacta® platform and the Azure infrastructure to include SSO connections to Azure-based relational sources.

Supported relational connection types:

- Azure SQL Database
- SQL Datawarehouse

Pre-Requisites

- 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:
    1. Non-SSO Azure connections: Shared normally, with or without credentials.
    2. SSO Azure connections:
        1. The connection can be shared, but the credentials cannot.
        2. 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 Trifacta platform

Your Azure admin must enable the following:

1. Your SQL Server database must have an Active Directory Admin assigned to it.
    1. 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 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:
    1. SSH to the Trifacta node. Login as an administrator.
    2. Navigate to the following:

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

3. Open `trifacta-conf.json`.
4. Locate the `azure.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": {
           "scope": [
               "https://datalake.azure.net/user_impersonation",
               "https://database.windows.net/user_impersonation"
           ]
       }
   }
   ```

5. 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/service/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 *Create Azure SQL Database Connections*.
- See *Create SQL DW Connections*.

User Access

Users can access the connections through the Import Data page. See *Import Data Page*.

TRIFACTA