**REPORT**

**ON**

**Configuration of Azure Data factory and create pipelines to take data from Azure Blob and insert it into Azure SQL**

Submitted in partial fulfilment of the requirements

for the award of the Degree of

**Bachelor of Technology**

**of**

**Poornima University, Jaipur**

****

**Submitted By:**

**AKANSHA IMMANUEL ………………………………………2017PUSETBCCX**

**WASIM AKRAM…………………………………………………2017PUSETBCCX**

**AMAN GOYAL……………………………………………………2017PUSETBCCX**

**ABHISHEK KHARRA…...………………………………………2017PUSETBCCX**

**AKHILESH PRAJAPAT………………………………………….2017PUSETBCCX**

**Submitted To:**

**Dr. Nitesh Kaushik AND Akansha Jain ( Asst. Professor)**

**Department of Computer Engineering** **School of Engineering & Technology, Poornima University Ramchandrapura, Sitapura Ext., Jaipur, Rajasthan**

**Session: 2019-20**

Table of Contents

[**What is Azure Data Factory?** 2](#_Toc40559010)

[**How does it work?** 3](#_Toc40559011)

[**Connect and collect** 3](#_Toc40559012)

[**Transform and enrich** 4](#_Toc40559013)

[**CI/CD and publish** 4](#_Toc40559014)

[**Monitor** 4](#_Toc40559015)

[**Top-level concepts** 4](#_Toc40559016)

[**Pipeline** 4](#_Toc40559017)

[**Mapping data flows** 5](#_Toc40559018)

[**Datasets** 5](#_Toc40559019)

[**Linked services** 5](#_Toc40559020)

[**Triggers** 5](#_Toc40559021)

[**Pipeline runs** 6](#_Toc40559022)

[**Parameters** 6](#_Toc40559023)

[**Control flow** 6](#_Toc40559024)

[**Variables** 6](#_Toc40559025)

[Prerequisites 7](#_Toc40559026)

[Azure subscription 7](#_Toc40559027)

[Azure roles 7](#_Toc40559028)

[SQL Server 7](#_Toc40559029)

[Azure storage account 8](#_Toc40559030)

[Create a data factory 9](#_Toc40559031)

[Create a pipeline 11](#_Toc40559032)

[Trigger a pipeline run 17](#_Toc40559033)

[Monitor the pipeline run 17](#_Toc40559034)

**Configuration of Azure data factory and create pipelines to take data from Azure Blob and insert it into Azure SQL**

**What is Azure Data Factory?**

In the world of big data, raw, unorganized data is often stored in relational, non-relational, and other storage systems. However, on its own, raw data doesn't have the proper context or meaning to provide meaningful insights to analysts, data scientists, or business decision makers.

Big data requires service that can orchestrate and operationalize processes to refine these enormous stores of raw data into actionable business insights. Azure Data Factory is a managed cloud service that's built for these complex hybrid extract-transform-load (ETL), extract-load-transform (ELT), and data integration projects.

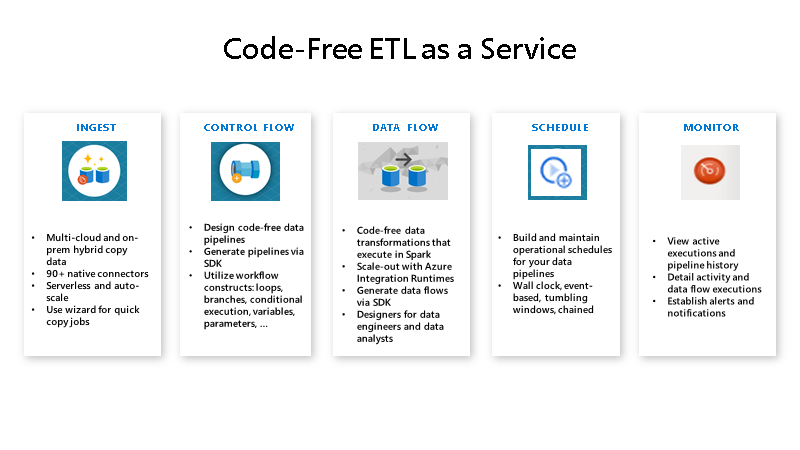
For example, imagine a gaming company that collects petabytes of game logs that are produced by games in the cloud. The company wants to analyze these logs to gain insights into customer preferences, demographics, and usage behavior. It also wants to identify up-sell and cross-sell opportunities, develop compelling new features, drive business growth, and provide a better experience to its customers.

To analyze these logs, the company needs to use reference data such as customer information, game information, and marketing campaign information that is in an on-premises data store. The company wants to utilize this data from the on-premises data store, combining it with additional log data that it has in a cloud data store.

To extract insights, it hopes to process the joined data by using a Spark cluster in the cloud (Azure HDInsight), and publish the transformed data into a cloud data warehouse such as Azure SQL Data Warehouse to easily build a report on top of it. They want to automate this workflow, and monitor and manage it on a daily schedule. They also want to execute it when files land in a blob store container.

Azure Data Factory is the platform that solves such data scenarios. It is the *cloud-based ETL and data integration service that allows you to create data-driven workflows for orchestrating data movement and transforming data at scale*. Using Azure Data Factory, you can create and schedule data-driven workflows (called pipelines) that can ingest data from disparate data stores. You can build complex ETL processes that transform data visually with data flows or by using compute services such as Azure HDInsight Hadoop, Azure Databricks, and Azure SQL Database.

Additionally, you can publish your transformed data to data stores such as Azure SQL Data Warehouse for business intelligence (BI) applications to consume. Ultimately, through Azure Data Factory, raw data can be organized into meaningful data stores and data lakes for better business decisions.

****

**How does it work?**

Data Factory contains a series of interconnected systems that provide a complete end-to-end platform for data engineers.

**Connect and collect**

Enterprises have data of various types that are located in disparate sources on-premises, in the cloud, structured, unstructured, and semi-structured, all arriving at different intervals and speeds.

The first step in building an information production system is to connect to all the required sources of data and processing, such as software-as-a-service (SaaS) services, databases, file shares, and FTP web services. The next step is to move the data as needed to a centralized location for subsequent processing.

Without Data Factory, enterprises must build custom data movement components or write custom services to integrate these data sources and processing. It's expensive and hard to integrate and maintain such systems. In addition, they often lack the enterprise-grade monitoring, alerting, and the controls that a fully managed service can offer.

With Data Factory, you can use the [Copy Activity](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview) in a data pipeline to move data from both on-premises and cloud source data stores to a centralization data store in the cloud for further analysis. For example, you can collect data in Azure Data Lake Storage and transform the data later by using an Azure Data Lake Analytics compute service. You can also collect data in Azure Blob storage and transform it later by using an Azure HDInsight Hadoop cluster.

**Transform and enrich**

After data is present in a centralized data store in the cloud, process or transform the collected data by using ADF mapping data flows. Data flows enable data engineers to build and maintain data transformation graphs that execute on Spark without needing to understand Spark clusters or Spark programming.

If you prefer to code transformations by hand, ADF supports external activities for executing your transformations on compute services such as HDInsight Hadoop, Spark, Data Lake Analytics, and Machine Learning.

**CI/CD and publish**

Data Factory offers full support for CI/CD of your data pipelines using Azure DevOps and GitHub. This allows you to incrementally develop and deliver your ETL processes before publishing the finished product. After the raw data has been refined into a business-ready consumable form, load the data into Azure Data Warehouse, Azure SQL Database, Azure CosmosDB, or whichever analytics engine your business users can point to from their business intelligence tools.

**Monitor**

After you have successfully built and deployed your data integration pipeline, providing business value from refined data, monitor the scheduled activities and pipelines for success and failure rates. Azure Data Factory has built-in support for pipeline monitoring via Azure Monitor, API, PowerShell, Azure Monitor logs, and health panels on the Azure portal.

**Top-level concepts**

An Azure subscription might have one or more Azure Data Factory instances (or data factories). Azure Data Factory is composed of four key components. These components work together to provide the platform on which you can compose data-driven workflows with steps to move and transform data.

**Pipeline**

A data factory might have one or more pipelines. A pipeline is a logical grouping of activities that performs a unit of work. Together, the activities in a pipeline perform a task. For example, a pipeline can contain a group of activities that ingests data from an Azure blob, and then runs a Hive query on an HDInsight cluster to partition the data.

The benefit of this is that the pipeline allows you to manage the activities as a set instead of managing each one individually. The activities in a pipeline can be chained together to operate sequentially, or they can operate independently in parallel.

**Mapping data flows**

Create and manage graphs of data transformation logic that you can use to transform any-sized data. You can build-up a reusable library of data transformation routines and execute those processes in a scaled-out manner from your ADF pipelines. Data Factory will execute your logic on a Spark cluster that spins-up and spins-down when you need it. You won't ever have to manage or maintain clusters.

**Activity**

Activities represent a processing step in a pipeline. For example, you might use a copy activity to copy data from one data store to another data store. Similarly, you might use a Hive activity, which runs a Hive query on an Azure HDInsight cluster, to transform or analyze your data. Data Factory supports three types of activities: data movement activities, data transformation activities, and control activities.

**Datasets**

Datasets represent data structures within the data stores, which simply point to or reference the data you want to use in your activities as inputs or outputs.

**Linked services**

Linked services are much like connection strings, which define the connection information that's needed for Data Factory to connect to external resources. Think of it this way: a linked service defines the connection to the data source, and a dataset represents the structure of the data. For example, an Azure Storage-linked service specifies a connection string to connect to the Azure Storage account. Additionally, an Azure blob dataset specifies the blob container and the folder that contains the data.

Linked services are used for two purposes in Data Factory:

* To represent a **data store** that includes, but isn't limited to, an on-premises SQL Server database, Oracle database, file share, or Azure blob storage account. For a list of supported data stores, see the [copy activity](https://docs.microsoft.com/en-us/azure/data-factory/copy-activity-overview) article.
* To represent a **compute resource** that can host the execution of an activity. For example, the HDInsightHive activity runs on an HDInsight Hadoop cluster. For a list of transformation activities and supported compute environments, see the [transform data](https://docs.microsoft.com/en-us/azure/data-factory/transform-data) article.

**Triggers**

Triggers represent the unit of processing that determines when a pipeline execution needs to be kicked off. There are different types of triggers for different types of events.

**Pipeline runs**

A pipeline run is an instance of the pipeline execution. Pipeline runs are typically instantiated by passing the arguments to the parameters that are defined in pipelines. The arguments can be passed manually or within the trigger definition.

**Parameters**

Parameters are key-value pairs of read-only configuration.  Parameters are defined in the pipeline. The arguments for the defined parameters are passed during execution from the run context that was created by a trigger or a pipeline that was executed manually. Activities within the pipeline consume the parameter values.

A dataset is a strongly typed parameter and a reusable/referenceable entity. An activity can reference datasets and can consume the properties that are defined in the dataset definition.

A linked service is also a strongly typed parameter that contains the connection information to either a data store or a compute environment. It is also a reusable/referenceable entity.

**Control flow**

Control flow is an orchestration of pipeline activities that includes chaining activities in a sequence, branching, defining parameters at the pipeline level, and passing arguments while invoking the pipeline on-demand or from a trigger. It also includes custom-state passing and looping containers, that is, For-each iterators.

**Variables**

Variables can be used inside of pipelines to store temporary values and can also be used in conjunction with parameters to enable passing values between pipelines, data flows, and other activities.

The Azure Data Factory user interface (UI) create a data factory pipeline that copies data from an on-premises SQL Server database to Azure Blob storage. You create and use a self-hosted integration runtime, which moves data between on-premises and cloud data stores.

you perform the following steps:

* Create a data factory.
* Create a self-hosted integration runtime.
* Create SQL Server and Azure Storage linked services.
* Create SQL Server and Azure Blob datasets.
* Create a pipeline with a copy activity to move the data.
* Start a pipeline run.
* Monitor the pipeline run.

## Prerequisites

### Azure subscription

Before you begin, create an azure account.

### Azure roles

To create data factory instances, the user account you use to sign in to Azure must be assigned a Contributor or Owner role or must be an administrator of the Azure subscription.

To view the permissions, you have in the subscription, go to the Azure portal. In the upper-right corner, select your user name, and then select **Permissions**. If you have access to multiple subscriptions, select the appropriate subscription. For sample instructions on how to add a user to a role, see [Manage access using RBAC and the Azure portal](https://docs.microsoft.com/en-us/azure/role-based-access-control/role-assignments-portal).

### SQL Server

In this tutorial, you use an on-premises SQL Server database as a source data store. The pipeline in the data factory you create in this tutorial copies data from this on-premises SQL Server database (source) to Blob storage (sink). You then create a table named **emp** in your SQL Server database and insert a couple of sample entries into the table.

1. Start SQL Server Management Studio. If it's not already installed on your machine, go to [Download SQL Server Management Studio](https://docs.microsoft.com/sql/ssms/download-sql-server-management-studio-ssms).
2. Connect to your SQL Server instance by using your credentials.
3. Create a sample database. In the tree view, right-click **Databases**, and then select **New Database**.
4. In the **New Database** window, enter a name for the database, and then select **OK**.
5. To create the **emp** table and insert some sample data into it, run the following query script against the database. In the tree view, right-click the database that you created, and then select **New Query**.

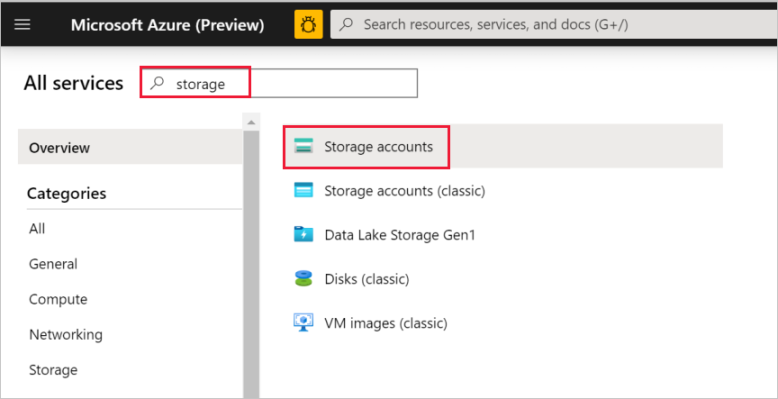
### Azure storage account

In this tutorial, you use a general-purpose Azure storage account (specifically, Blob storage) as a destination/sink data store. If you don't have a general-purpose Azure storage account, see [Create a storage account](https://docs.microsoft.com/en-us/azure/storage/common/storage-account-create). The pipeline in the data factory that you create in this tutorial copies data from the on-premises SQL Server database (source) to Blob storage (sink).

#### Get the storage account name and account key

You use the name and key of your storage account in this tutorial. To get the name and key of your storage account, take the following steps:

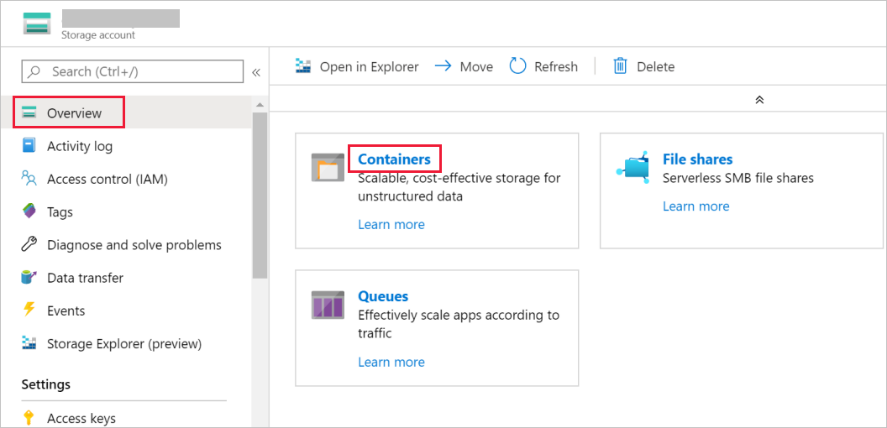
1. Sign in to the [Azure portal](https://portal.azure.com/) with your Azure user name and password.
2. In the left pane, select **All services**. Filter by using the **Storage** keyword, and then select **Storage accounts**.
3. In the list of storage accounts, filter for your storage account if needed. Then select your storage account.
4. In the **Storage account** window, select **Access keys**.
5. In the **Storage account name** and **key1** boxes, copy the values, and then paste them into Notepad or another editor for later use in the tutorial.



**The adftutorial container**

In this section, you create a blob container named **adftutorial** in your Blob storage.

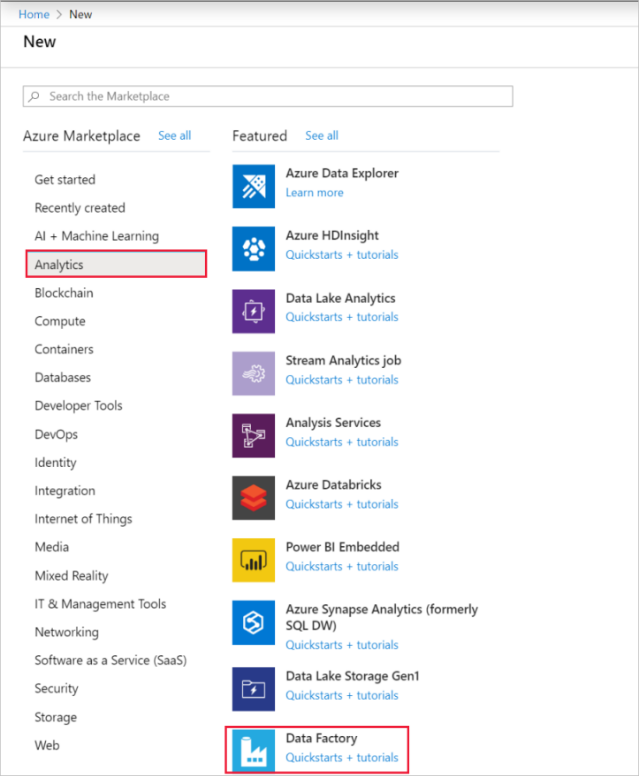
1. In the **Storage account** window, go to **Overview**, and then select **Containers**.
2. In the **Containers** window, select **+ Container** to create a new one.
3. In the **New container** window, under **Name**, enter **adftutorial**. Then select **Create**.
4. In the list of containers, select **adftutorial** you just created.
5. Keep the **container** window for **adftutorial** open. You use it to verify the output at the end of the tutorial. Data Factory automatically creates the output folder in this container, so you don't need to create one.

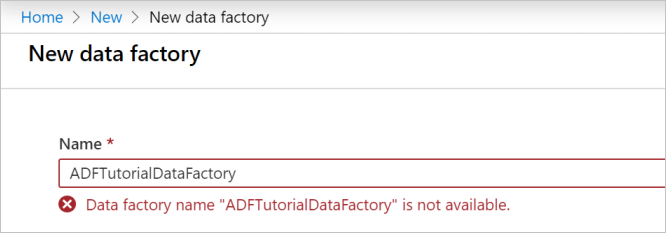
****

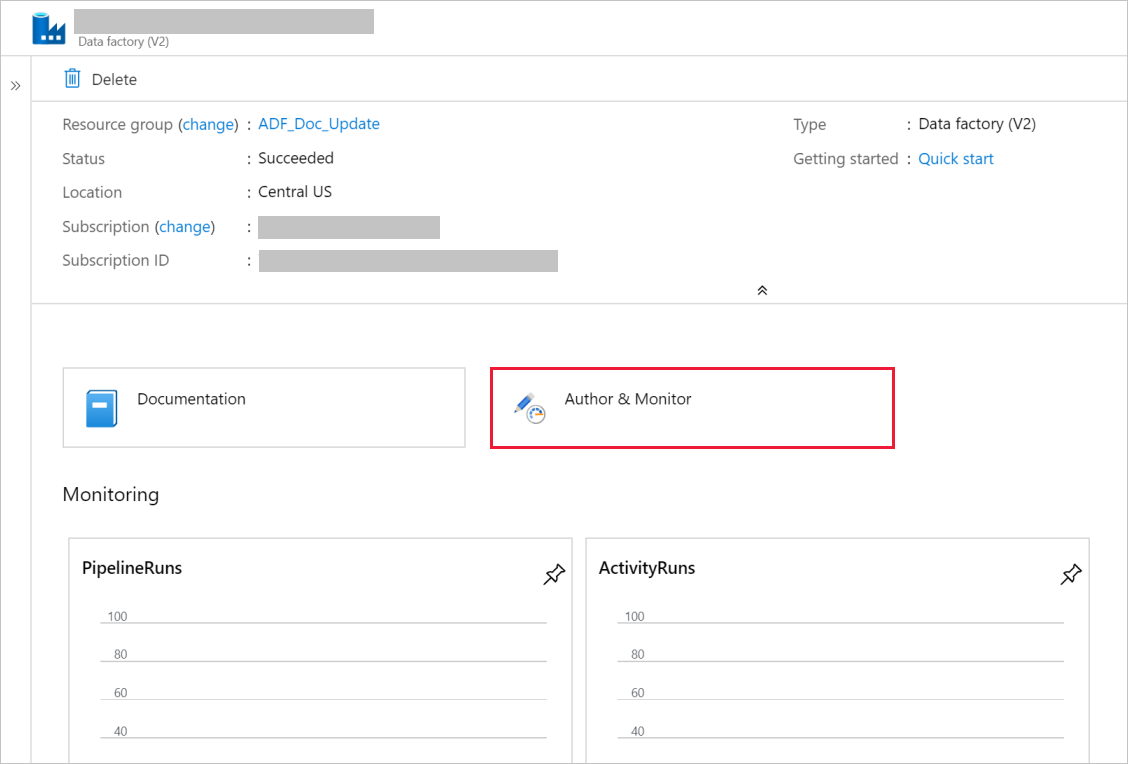
## Create a data factory

In this step, you create a data factory and start the Data Factory UI to create a pipeline in the data factory.

1. Open the **Microsoft Edge** or **Google Chrome** web browser. Currently, Data Factory UI is supported only in Microsoft Edge and Google Chrome web browsers.
2. On the left menu, select **Create a resource** > **Analytics** > **Data Factory**.
3. On the **New data factory** page, under **Name**, enter **ADFTutorialDataFactory**.
4. The name of the data factory must be globally unique. If you see the following error message for the name field, change the name of the data factory (for example, yournameADFTutorialDataFactory). For naming rules for Data Factory artifacts, see [Data Factory naming rules](https://docs.microsoft.com/en-us/azure/data-factory/naming-rules).
5. Select the Azure **subscription** in which you want to create the data factory.
6. For **Resource Group**, take one of the following steps:
   1. Select **Use existing**, and select an existing resource group from the drop-down list.
   2. Select **Create new**, and enter the name of a resource group.
      1. To learn about resource groups, see [Use resource groups to manage your Azure resources](https://docs.microsoft.com/en-us/azure/azure-resource-manager/management/overview).
7. Under **Version**, select **V2**.
8. Under **Location**, select the location for the data factory. Only locations that are supported are displayed in the drop-down list. The data stores (for example, Storage and SQL Database) and computes (for example, Azure HDInsight) used by Data Factory can be in other regions.
9. Select **Create**.
10. After the creation is finished, you see the **Data Factory** page as shown in the image.
11. Select the **Author & Monitor** tile to launch the Data Factory UI in a separate tab.

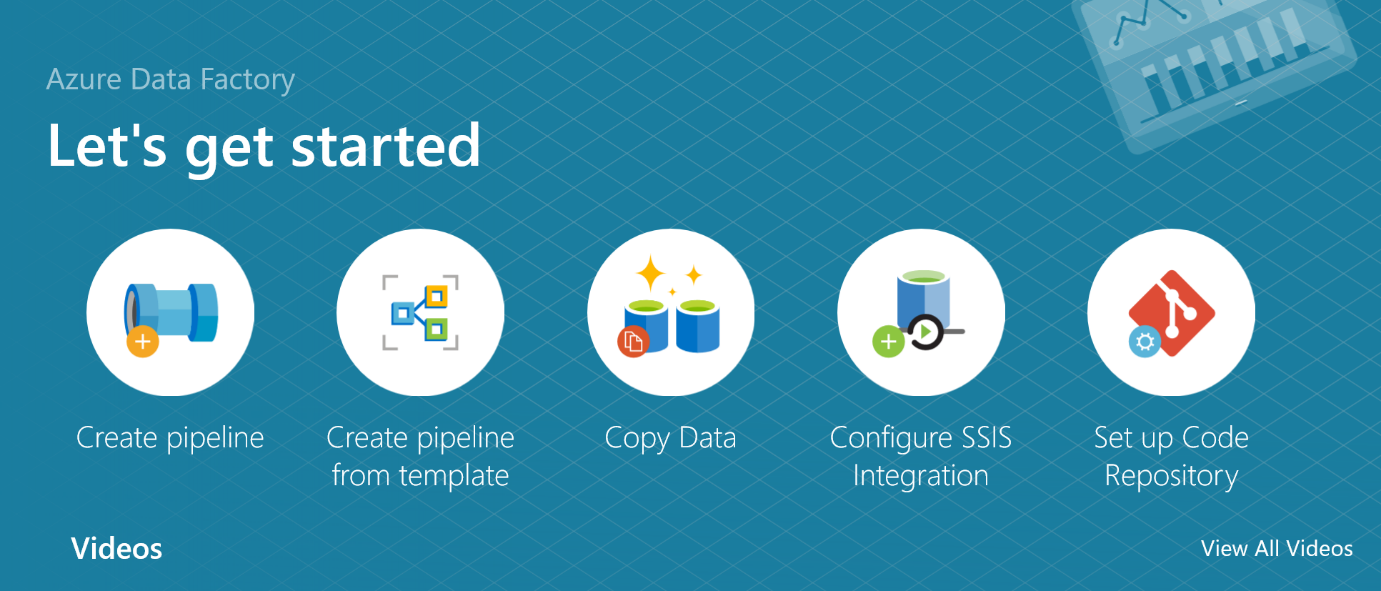




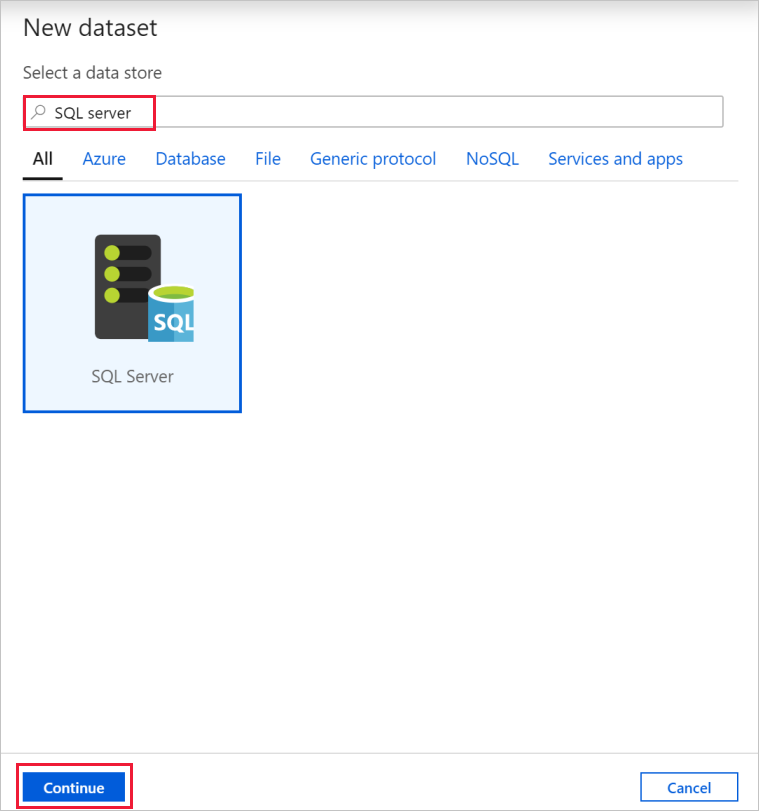
****

## Create a pipeline

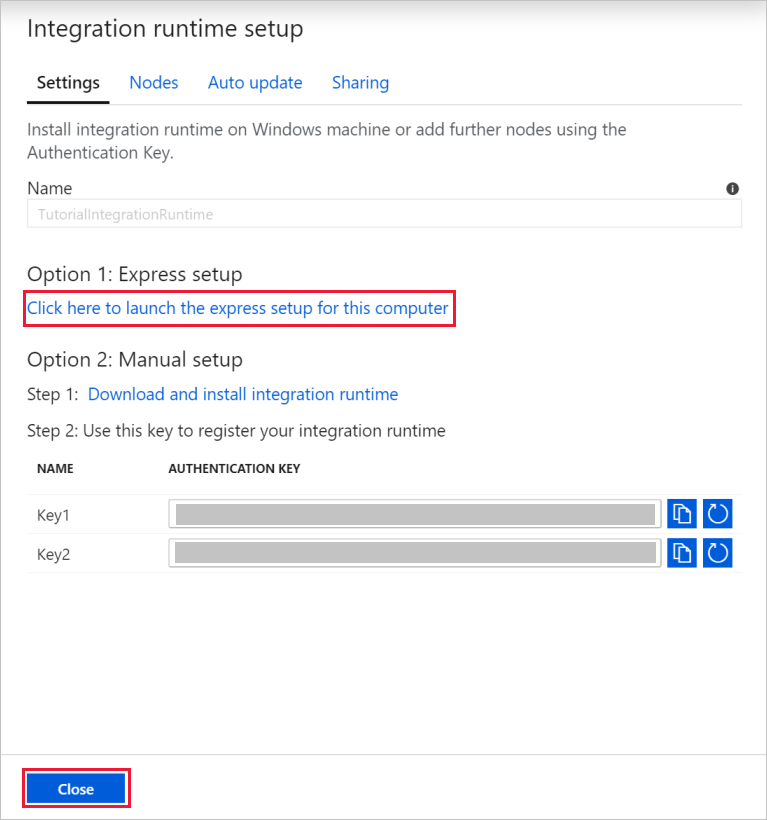
1. On the **Let's get started** page, select **Create pipeline**. A pipeline is automatically created for you. You see the pipeline in the tree view, and its editor opens.



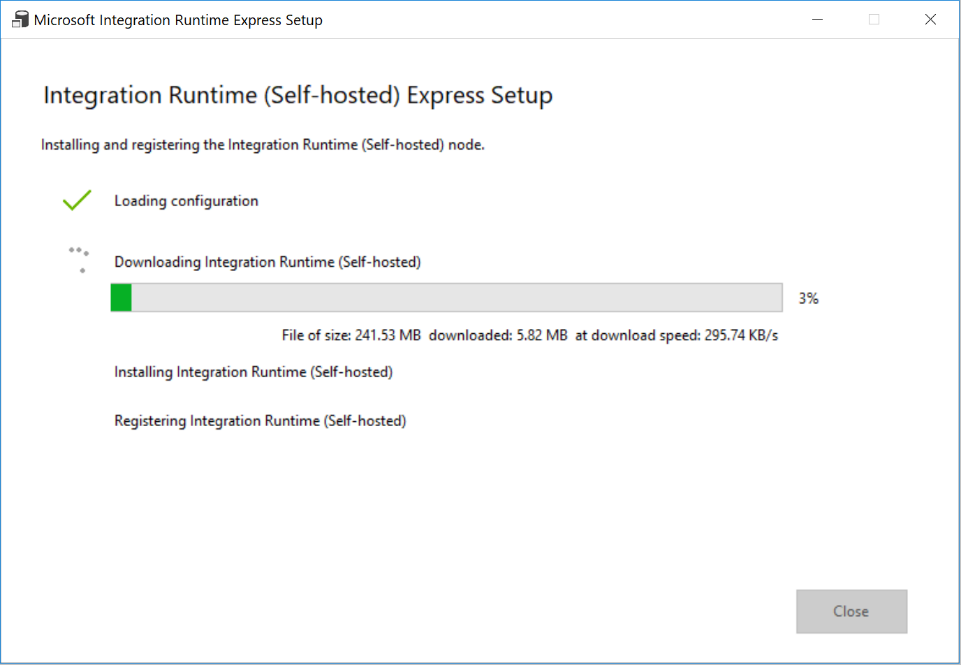
1. On the **General** tab at the bottom of the **Properties** window, for **Name**, enter **SQLServerToBlobPipeline**.
2. In the **Activities** tool box, expand **Move & Transform**. Drag and drop the **Copy** activity to the pipeline design surface. Set the name of the activity to **CopySqlServerToAzureBlobActivity**.
3. In the **Properties** window, go to the **Source** tab, and select **+ New**.
4. In the **New Dataset** dialog box, search for **SQL Server**. Select **SQL Server**, and then select **Continue**.



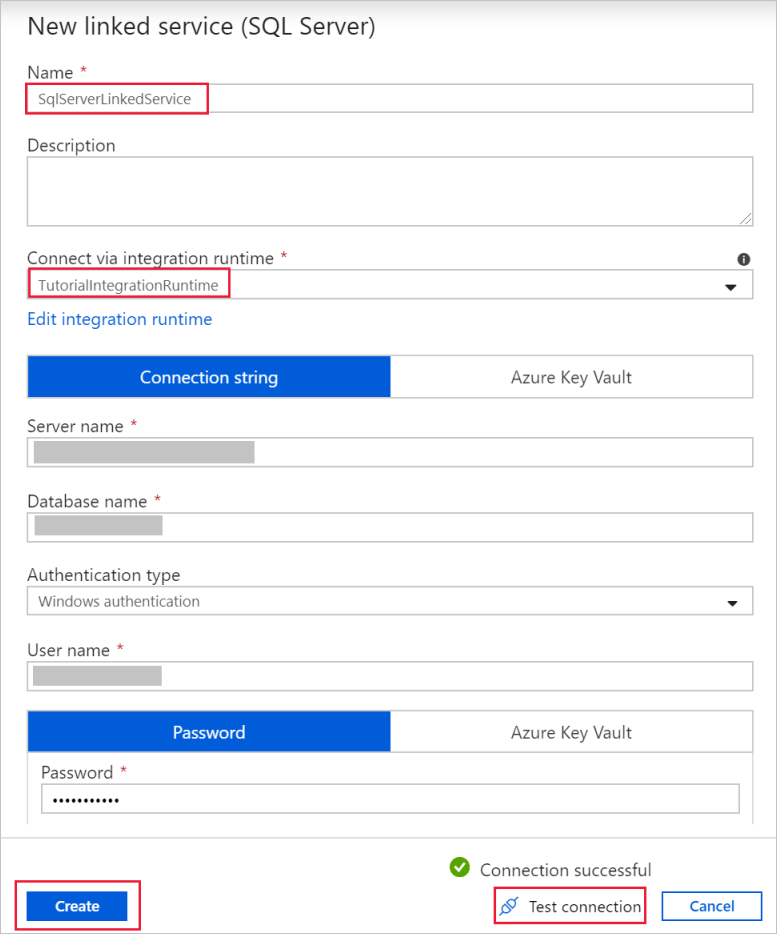
1. In the **Set Properties** dialog box, under **Name**, enter **SqlServerDataset**. Under **Linked service**, select **+ New**. You create a connection to the source data store (SQL Server database) in this step.
2. In the **New Linked Service** dialog box, add **Name** as **SqlServerLinkedService**. Under **Connect via integration runtime**, select **+New**. In this section, you create a self-hosted integration runtime and associate it with an on-premises machine with the SQL Server database. The self-hosted integration runtime is the component that copies data from the SQL Server database on your machine to Blob storage.
3. In the **Integration Runtime Setup** dialog box, select **Self-Hosted**, and then select **Continue**.
4. Under name, enter **TutorialIntegrationRuntime**. Then select **Create**.
5. For Settings, select **Click here to launch the express setup for this computer**. This action installs the integration runtime on your machine and registers it with Data Factory. Alternatively, you can use the manual setup option to download the installation file, run it, and use the key to register the integration runtime.



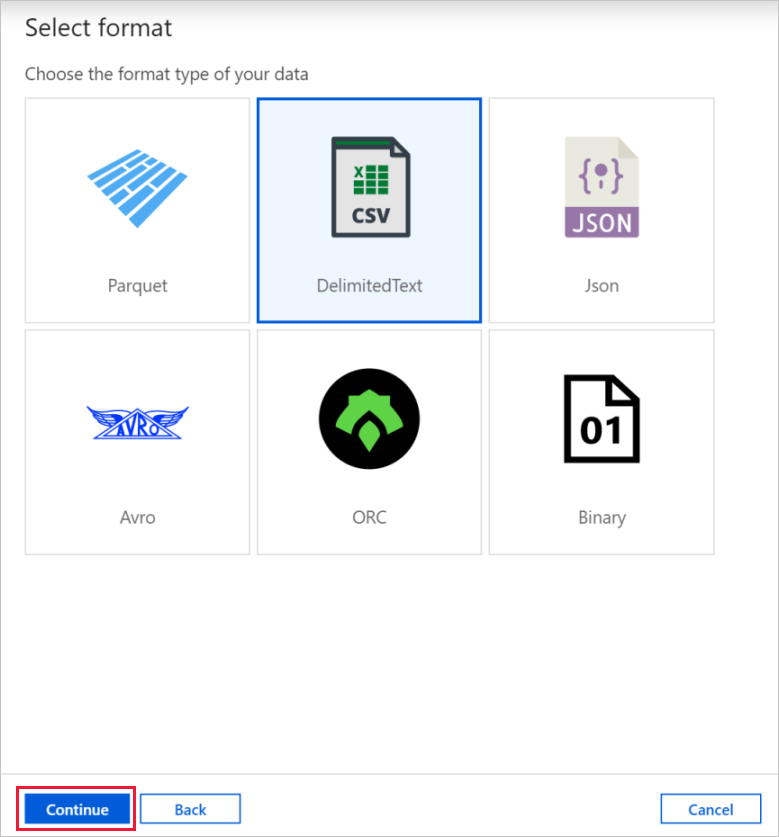
1. In the **Integration Runtime (Self-hosted) Express Setup** window, select **Close** when the process is finished.



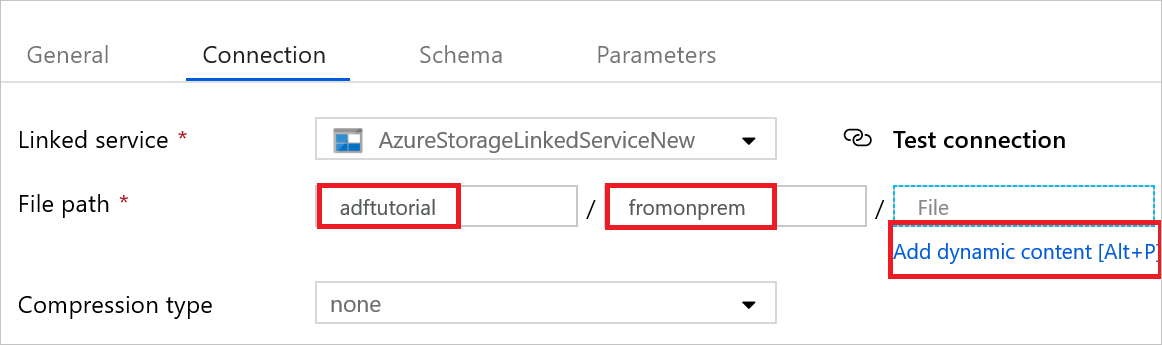
1. In the **New linked service (SQL Server)** dialog box, confirm that **TutorialIntegrationRuntime** is selected under **Connect via integration runtime**. Then, take the following steps:
   1. Under **Name**, enter **SqlServerLinkedService**.
   2. Under **Server name**, enter the name of your SQL Server instance.
   3. Under **Database name**, enter the name of the database with the **emp** table.
   4. Under **Authentication type**, select the appropriate authentication type that Data Factory should use to connect to your SQL Server database.
   5. Under **User name** and **Password**, enter the user name and password. If you need to use a backslash (\) in your user account or server name, precede it with the escape character (\). For example, use *mydomain\\myuser*.
   6. Select **Test connection**. This step is to confirm that Data Factory can connect to your SQL Server database by using the self-hosted integration runtime you created.
   7. To save the linked service, select **Create**.



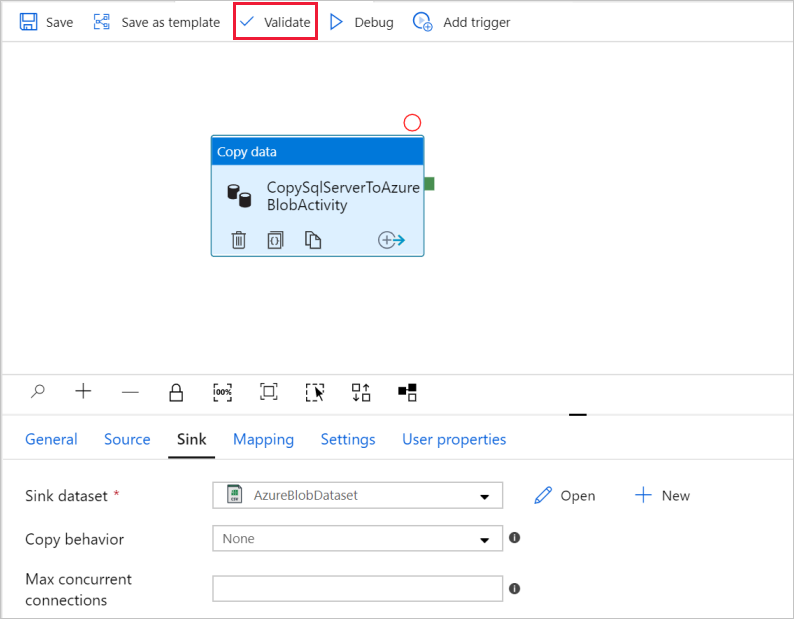
1. After the linked service is created, you're back to the **Set properties** page for the SqlServerDataset. Take the following steps:
   1. In **Linked service**, confirm that you see **SqlServerLinkedService**.
   2. Under **Table name**, select **[dbo].[emp]**.
   3. Select **OK**.
2. Go to the tab with **SQLServerToBlobPipeline**, or select **SQLServerToBlobPipeline** in the tree view.
3. Go to the **Sink** tab at the bottom of the **Properties** window, and select **+ New**.
4. In the **New Dataset** dialog box, select **Azure Blob Storage**. Then select **Continue**.
5. In **Select Format** dialog box, choose the format type of your data. Then select **Continue**



1. In the **Set Properties** dialog box, enter **AzureBlobDataset** for Name. Next to the **Linked service** text box, select **+ New**.
2. In the **New Linked Service (Azure Blob Storage)** dialog box, enter **AzureStorageLinkedService** as name, select your storage account from the **Storage account** name list. Test connection, and then select **Create** to deploy the linked service.
3. After the linked service is created, you're back to the **Set properties** page. Select **OK**.
4. Open the sink dataset. On the **Connection** tab, take the following steps:
   1. In **Linked service**, confirm that **AzureStorageLinkedService** is selected.
   2. In **File path**, enter **adftutorial/fromonprem** for the **Container/ Directory** part. If the output folder doesn't exist in the adftutorial container, Data Factory automatically creates the output folder.
   3. For the **File** part, select **Add dynamic content**.



1. d. Add @CONCAT(pipeline().RunId, '.txt'), and then select **Finish**. This action will rename the file with PipelineRunID.txt.
2. Go to the tab with the pipeline opened, or select the pipeline in the tree view. In **Sink Dataset**, confirm that **AzureBlobDataset** is selected.
3. To validate the pipeline settings, select **Validate** on the toolbar for the pipeline. To close the **Pipe validation output**, select the **>>** icon.



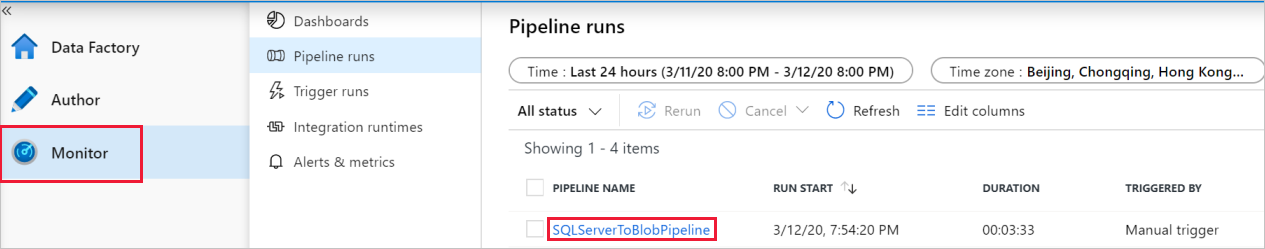
1. To publish entities you created to Data Factory, select **Publish all**.
2. Wait until you see the **Publishing completed** pop-up. To check the status of publishing, select the **Show notifications** link on the top of the window. To close the notification window, select **Close**.

## Trigger a pipeline run

Select **Add Trigger** on the toolbar for the pipeline, and then select **Trigger Now**.

## Monitor the pipeline run

1. Go to the **Monitor** tab. You see the pipeline that you manually triggered in the previous step.
2. To view activity runs associated with the pipeline run, select the **SQLServerToBlobPipeline** link under PIPELINE NAME.
3. On the **Activity runs** page, select the Details (eyeglasses image) link to see details about the copy operation. To go back to the Pipeline Runs view, select **All pipeline runs** at the top.



# **Resources**

Azure Documentation, Copy data from an on-premises SQL Server database to Azure Blob storage: <https://docs.microsoft.com/en-us/azure/data-factory/tutorial-hybrid-copy-portal>

SQL Server 2017: <https://www.microsoft.com/en-in/sql-server/sql-server-downloads>

Azure Storage Explorer: <https://azure.microsoft.com/en-gb/features/storage-explorer/>