

Lab 00: Prerequisites and Setup

Background and Goals

The labs in this Azure Data Factory Deep Dive workshop are intended to give you experience working with a number of key components and data sources that are commonly used in production solutions. Participation in each lab activity is optional, but in order to complete all the labs in this workshop, you will need to configure the following resources.

Prerequisites

Azure Subscription

If you do not have an Azure subscription, create a **free account** on <https://azure.microsoft.com/free/> before you begin.

If you have an existing Azure subscription, ensure that you have **Service Administrator** or **Owner** level of access on the subscription. To view the permissions that you have in the subscription, in the Azure portal, select your username in the upper-right corner, click the ellipsis next to Switch directory, and then select *My Permissions*. If you have access to multiple subscriptions, select the appropriate subscription.

If you do not have Service Administrator or Owner level of access, we recommend that you create a free account. This is because some of the labs may not work if you can't create resources, assign permissions, or make modifications to the Azure Active Directory.

Whitelisted Preview of Azure Data Factory Mapping Data Flows

The Mapping Data Flows feature in Azure Data Factory is still in preview. To participate in these labs, you must sign up for the preview on <http://aka.ms/dataflowpreview>. Make sure you enter "SQLBits" as the Business Name, as this will speed up the approval process.

Local Machine

Currently, the Azure Data Factory UI is only supported in the **Microsoft Edge** and **Google Chrome** browsers. Ensure that you have one of these browsers installed.

We also recommend that you have the following applications installed locally:

- **SQL Server Management Studio (SSMS)**: <https://docs.microsoft.com/en-us/sql/ssms/download-sql-server-management-studio-ssms>
- **SQL Server Data Tools (SSDT)**: <https://docs.microsoft.com/en-us/sql/ssdt/download-sql-server-data-tools-ssdt>
Please note: Make sure you use the **standalone installer** that installs tools for SQL Server Integration Services
- **Azure Storage Explorer**: <https://azure.microsoft.com/en-us/features/storage-explorer/>

Setup: Configure Own Environment

Log into the Azure Portal (<https://portal.azure.com/>). You will create the following resources:

1. Resource Group
2. Azure Data Lake Storage Gen1
3. Azure Storage Account
4. Azure Key Vault
5. Azure SQL Server
6. Azure SQL Data Warehouse
7. Azure Analysis Services
8. Azure Databricks

9. Virtual Machine (Free SQL Server License: SQL Server 2017 Developer on Windows Server 2016) with AdventureWorksLT Demo Database

NAME	PUBLISHER
 Resource group	Microsoft
 Data Lake Storage Gen1	Microsoft
 Storage account	Microsoft
 Key Vault	Microsoft
 SQL server (logical server)	Microsoft
 SQL Database	Microsoft
 SQL Data Warehouse	Microsoft
 Analysis Services	Microsoft
 Azure Databricks	Microsoft
 Free SQL Server License: SQL Server 2017 Developer on Windows Server 2016	Microsoft

1. Create Resource Group

Resource Groups enable you to manage all your resources in an application together. All the remaining resources you create will be created in this Resource Group.

Step	Setting	Value	Notes
1	Resource Group	sqlbitsadf<999>	The name of the Resource Group. Use the prefix sqlbitsadf followed by a unique suffix such as your initials and/or a number. Make note of this name as you will refer back to it frequently in later activities.
2	Region	UK South	Choose UK South . It is important to note that not all Azure resources can be created in each region, but we selected UK South because <i>most</i> of the resources exist within that region.

Unless specified above, use the default settings. It is not necessary to add any tags.

Home > New > Marketplace > My Saved List > Resource group > Create a resource group

Create a resource group

Basics Tags Review + Create

Resource group - A container that holds related resources for an Azure solution. The resource group can include all the resources for the solution, or only those resources that you want to manage as a group. You decide how you want to allocate resources to resource groups based on what makes the most sense for your organization. [Learn more](#)

PROJECT DETAILS

* Subscription: Visual Studio Premium with MSDN

* Resource group: sqlbitsadfcw01 1

RESOURCE DETAILS

* Region: UK South 2

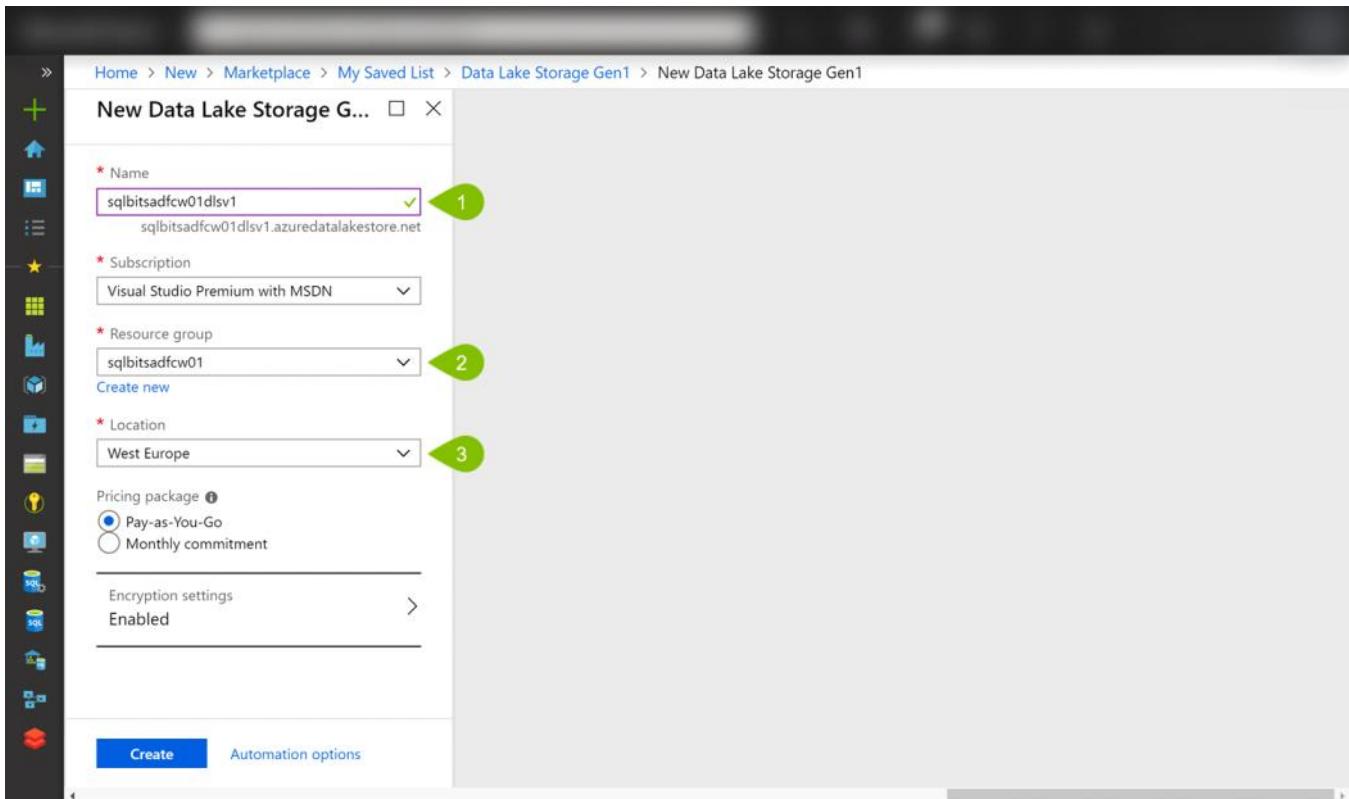
[Review + Create](#) [Next : Tags](#)

2. Create Azure Data Lake Storage Gen1

Please note: Azure Data Lake Storage Gen2 is now generally available. For these labs, you will still use Azure Data Lake Storage Gen1. This is because not all Azure Data Factory activities fully support Gen2 yet.

Step	Setting	Value	Notes
1	Name	sqlbitsadf<999>dlsv1	Use the name of the Resource Group and add dlsv1 as a suffix.
2	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the first step.
3	Location	West Europe	This resource cannot be created in UK South, so choose West Europe .

Unless specified above, use the default settings.



Additional information:

<https://docs.microsoft.com/en-us/azure/data-lake-store/data-lake-store-get-started-portal>

3. Create Azure Storage Account

Storage accounts allow you to store objects like blobs, files, and tables.

Step	Setting	Value	Notes
1	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the first step.
2	Storage Account Name	sqlbitsadf<999>wasb	Use the name of the Resource Group and add wasb as a suffix.
3	Location	UK South	Choose UK South .

Unless specified above, use the default settings.

Create storage account

INSTANCE DETAILS

The default deployment model is Resource Manager, which supports the latest Azure features. You may choose to deploy using the classic deployment model instead. [Choose classic deployment model](#)

Subscription: Visual Studio Premium with MSDN

Resource group: sqlbitsadfcw01 [Create new](#) (Step 1)

Storage account name: sqlbitsadfcw01wasb (Step 2)

Location: UK South (Step 3)

Performance: Standard (radio button selected) Premium

Account kind: StorageV2 (general purpose v2)

Replication: Read-access geo-redundant storage (RA-GRS)

Access tier (default): Cool Hot (radio button selected)

Review + create | Previous | **Next : Advanced >**

Additional information:

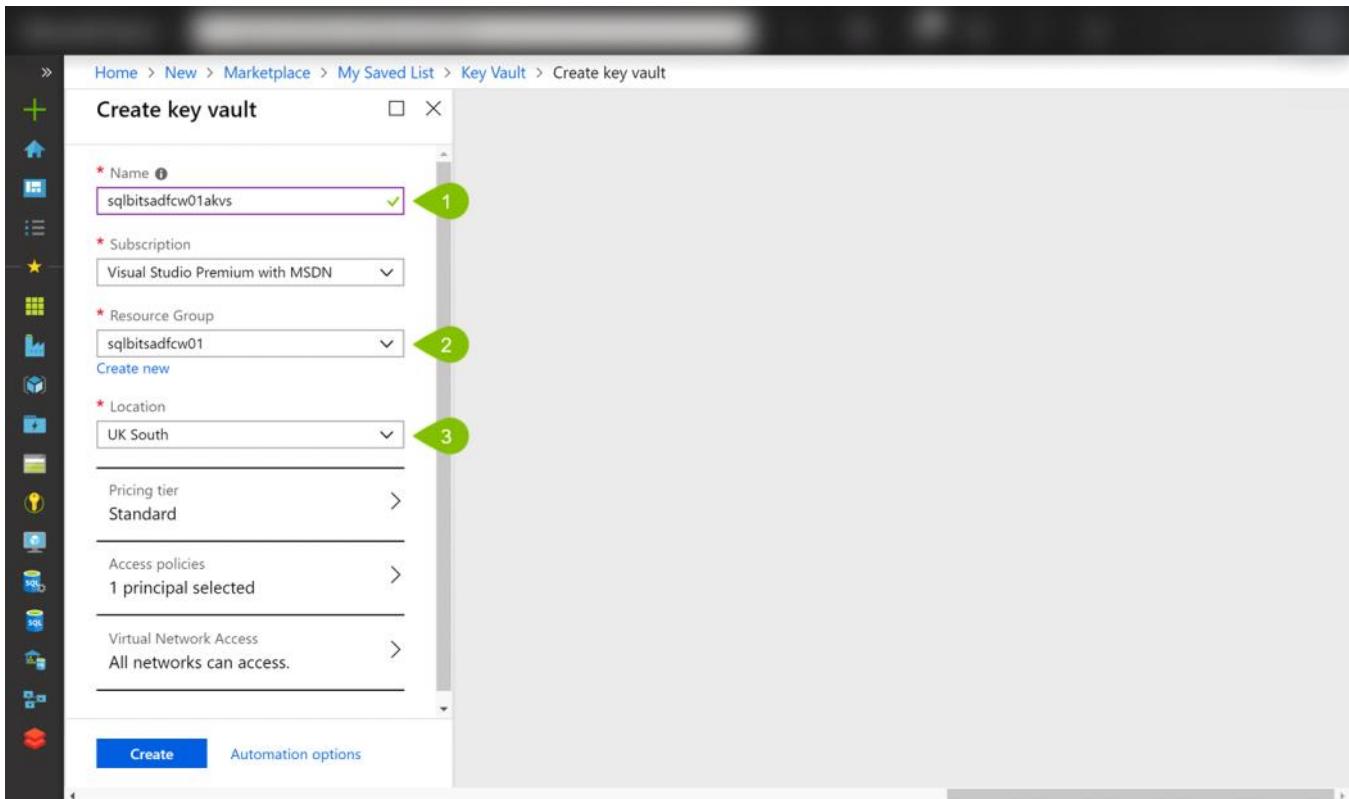
<https://docs.microsoft.com/en-us/azure/storage/common/storage-quickstart-create-account?toc=%2Fazure%2Fstorage%2Fblobs%2Ftoc.json&tabs=azure-portal>

4. Create Azure Key Vault

With Azure Key Vault, you can safely store objects like passwords and encryption keys.

Step	Setting	Value	Notes
1	Name	sqlbitsadf<999>akvs	Use the name of the Resource Group and add akvs as a suffix.
2	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the first step.
3	Location	UK South	Choose UK South .

Unless specified above, use the default settings.



Additional information:

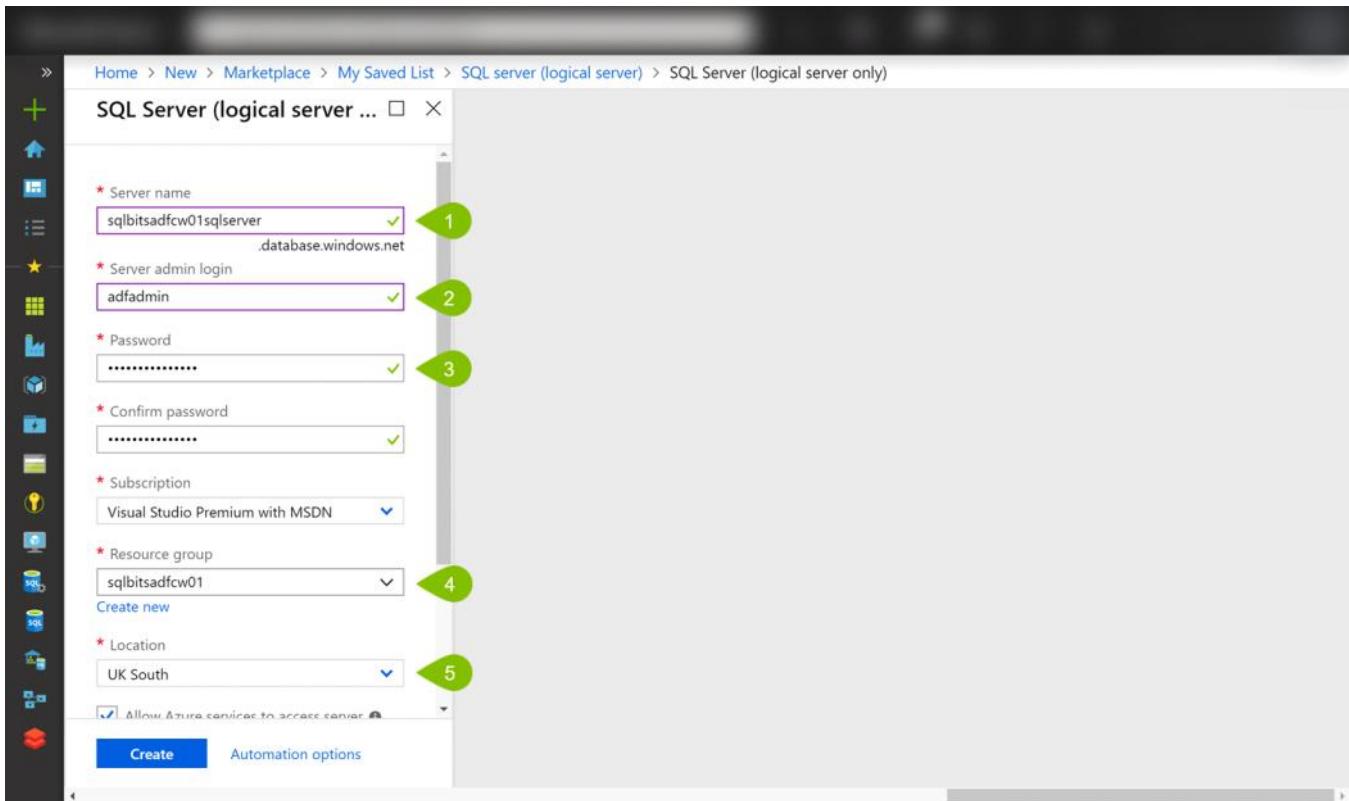
<https://docs.microsoft.com/en-us/azure/key-vault/quick-create-portal>

5. Create Azure SQL Server

The Azure SQL Server will host your SQL Database and SQL Data Warehouse.

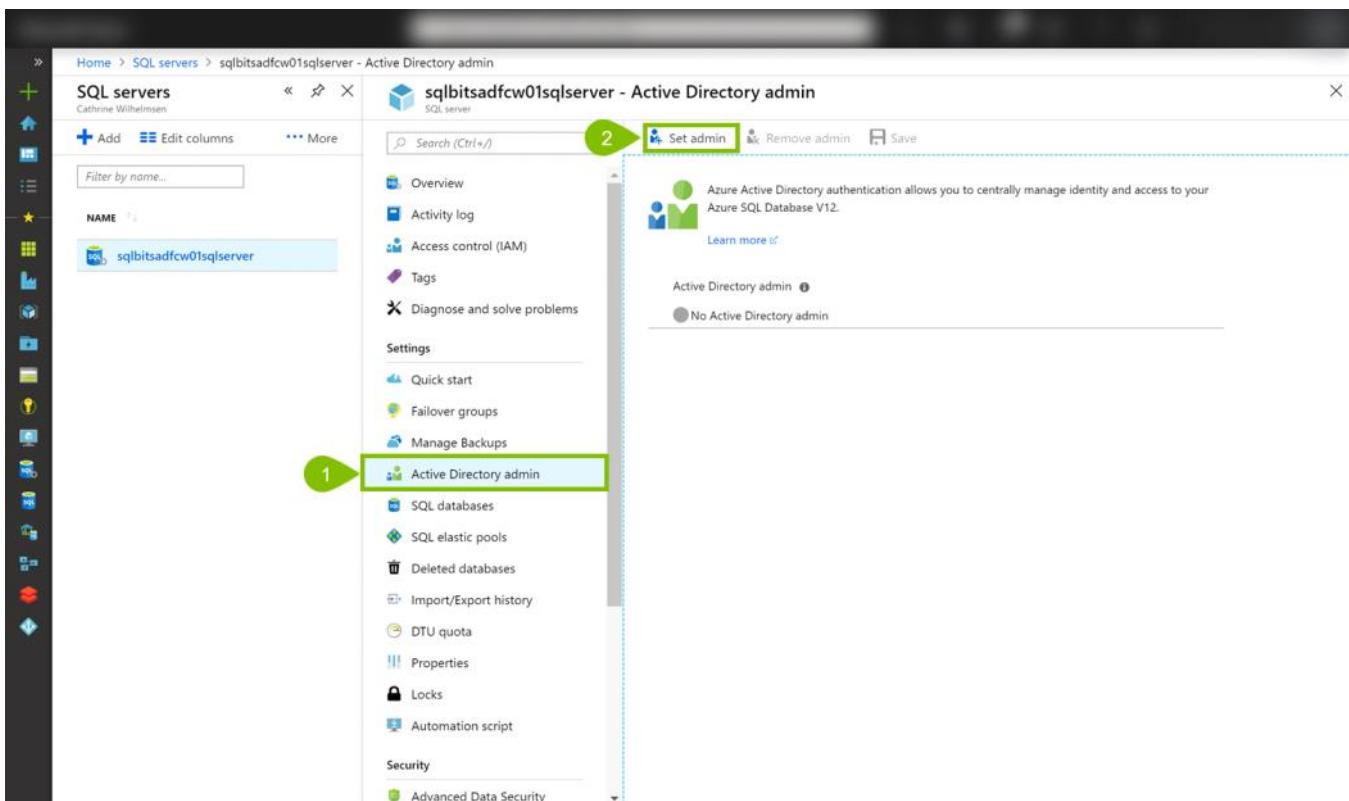
Step	Setting	Value	Notes
1	Server Name	sqlbitsadf<999>sqlserver	Use the name of the Resource Group and add sqlserver as a suffix.
2	SQL Admin Login	adfadmin	Pick a username, for example adfadmin
3	Password	VeryStr0ngPassw0rd!	Pick a password, for example VeryStr0ngPassw0rd!
4	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the first step.
5	Location	UK South	Choose UK South .

Unless specified above, use the default settings.

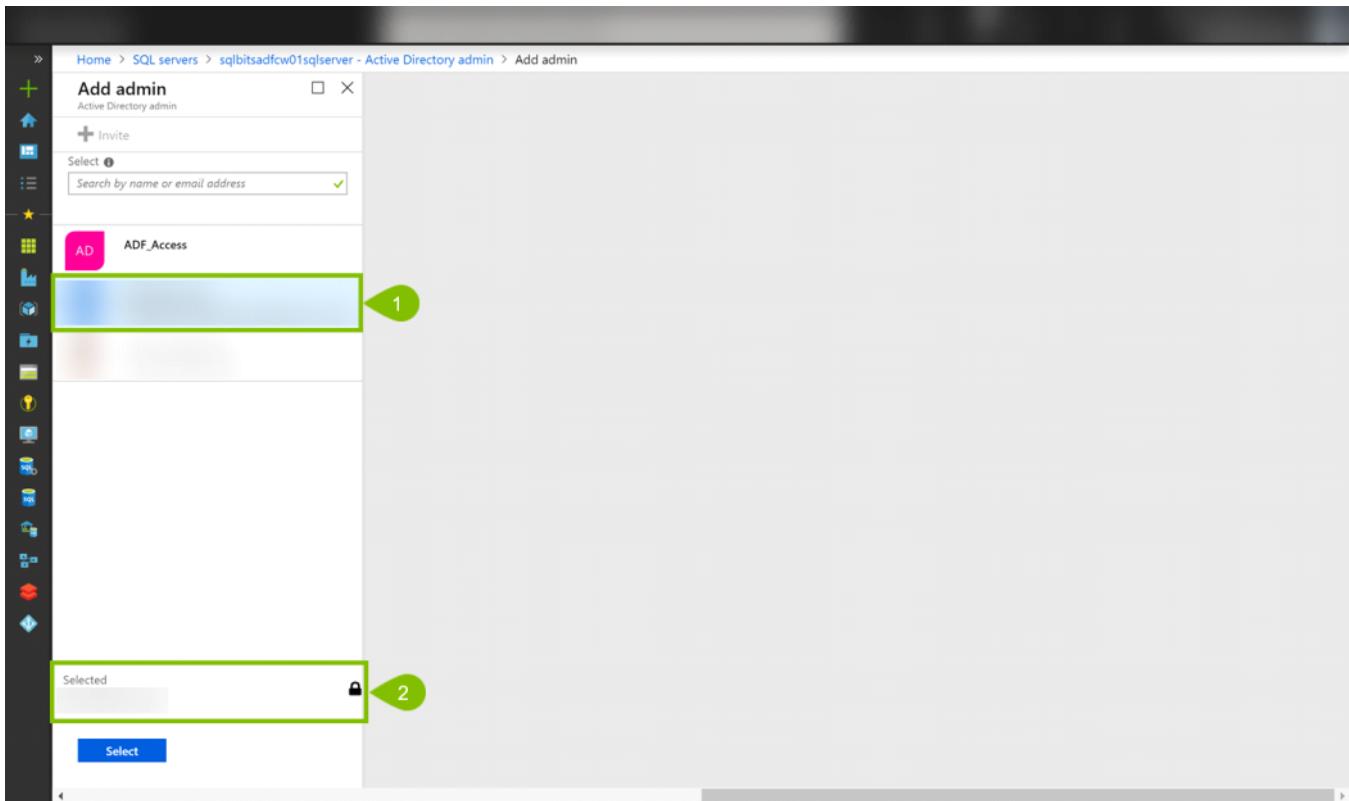


Add yourself as an Active Directory Admin of the Azure SQL Server

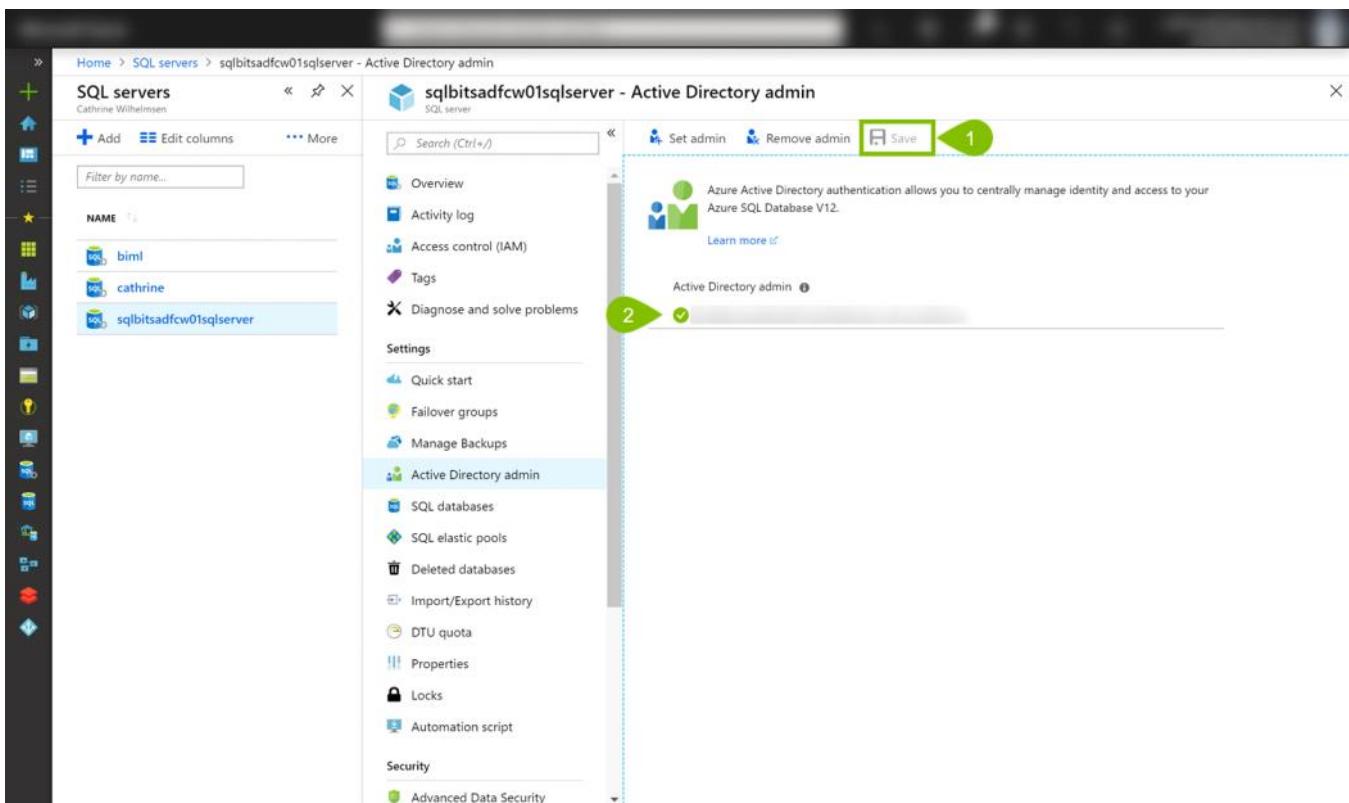
Click on **Active Directory Admin**, then **Set Admin**.



Select your user, then click **Select**.



Ensure that you **Save** the new Active Directory Admin and see the **green checkmark**.



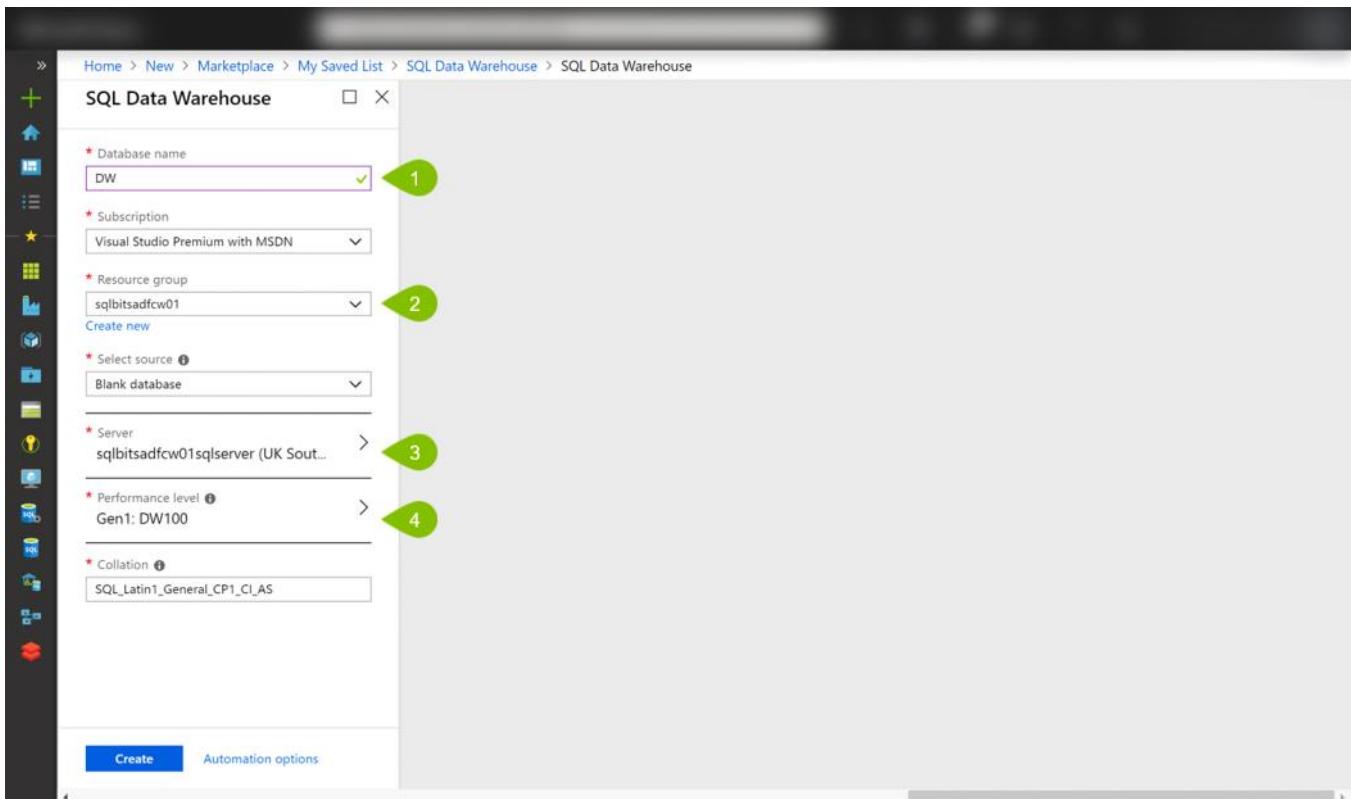
6. Create Azure SQL Data Warehouse

Create the Azure SQL Data Warehouse in the Azure SQL Server created in the previous step.

Please note: It is very important to ensure this resource is paused after creation, as it will incur costs if left running.

Step	Setting	Value	Notes
1	Database Name	DW	
2	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the first step.
3	Server	sqlbitsadf<999>sqlserver	Choose the SQL Server you created in the previous step.
4	Performance Level	Gen1: DW100	Change the default Gen2 DW1000c to the cheaper Gen1 DW100. (The cheapest Gen2 is not available by default in the UK South region.)

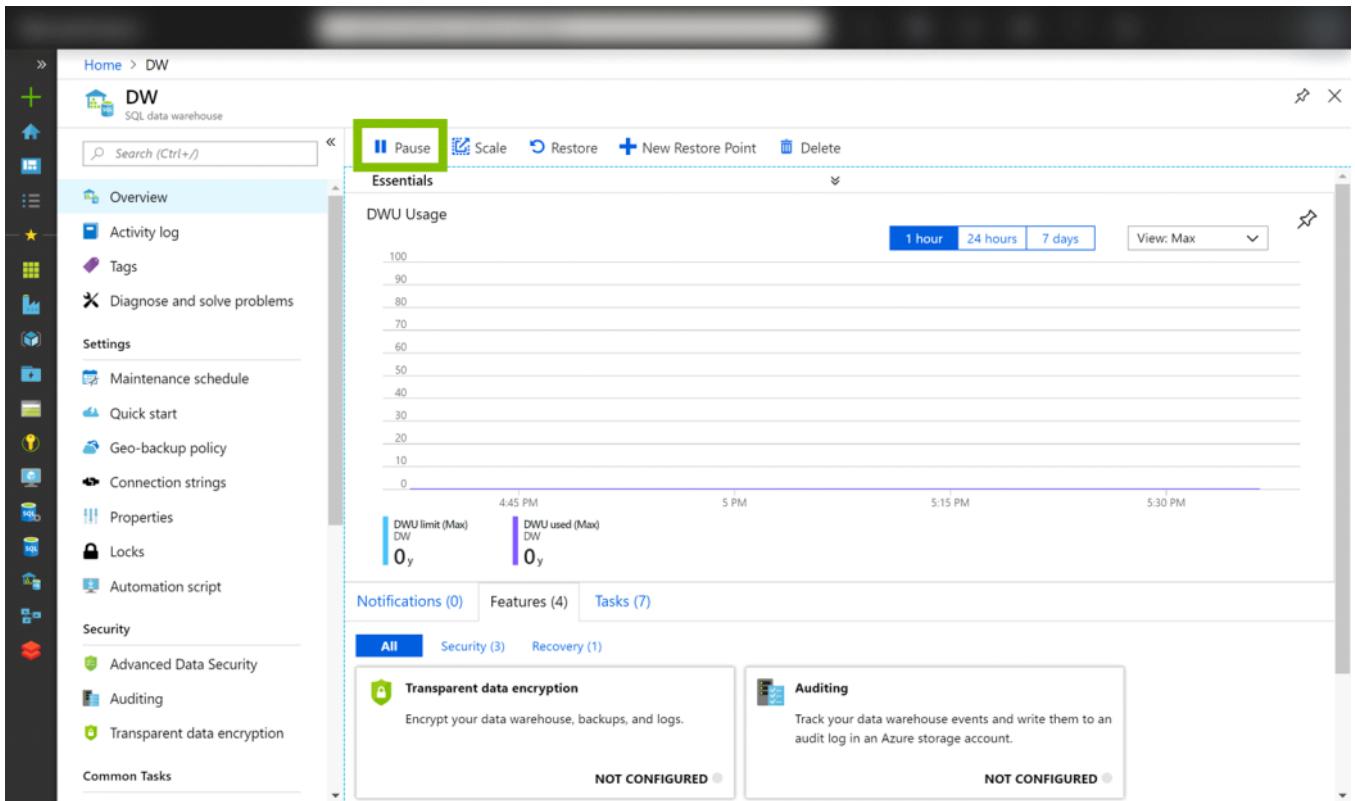
Unless specified above, use the default settings.



Additional information:

<https://docs.microsoft.com/en-us/azure/sql-data-warehouse/create-data-warehouse-portal>

Once the Azure SQL Data Warehouse has been deployed, pause it immediately.



Please note: It is very important to ensure this resource is paused after creation, as it will incur costs if left running.

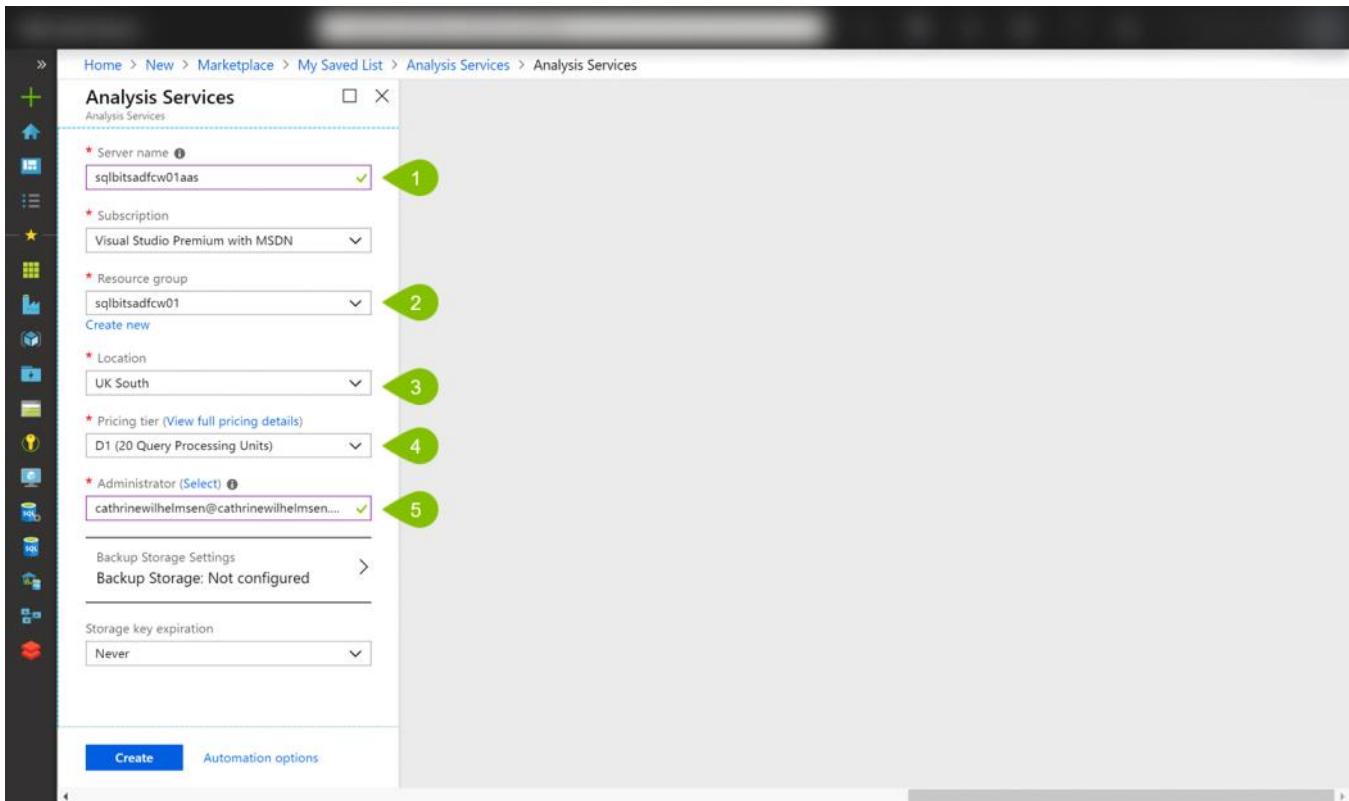
7. Create Azure Analysis Services

Azure Analysis Services integrates with many Azure services enabling you publish Tabular models to Azure.

Please note: It is very important to ensure this resource is paused after creation, as it will incur costs if left running.

Step	Setting	Value	Notes
1	Server Name	sqlbitsadf<999>aas	Use the name of the Resource Group and add aas as a suffix.
2	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the first step.
3	Location	UK South	Choose UK South .
4	Pricing Tier	D1	Choose D1 .
5	Administrator	<User>	Choose your own user.

Unless specified above, use the default settings.



Additional information:

<https://docs.microsoft.com/en-us/azure/analysis-services/analysis-services-create-server>

Once the Azure Analysis Services has been deployed, pause it immediately.

The screenshot shows the Azure portal interface for managing an Analysis Services instance named "sqlbitsadfcw01aas". The left sidebar contains navigation links for Home, Analysis Services, and the specific instance. The main area displays the instance details under the "Essentials" tab. Key information includes:

- Resource group: sqlbitsadfcw01
- Status: Active
- Location: UK South
- Subscription name: Visual Studio Premium with MSDN
- Pricing tier: D1
- Administrator: cathrinewilhelmsen@cathrinewilhelmsen...

Below the essentials, there's a section titled "Models on Analysis Services Server" which currently shows "No results". At the top of the main content area, there are buttons for "New model", "Pause" (which is highlighted with a green box), "Move", and "Delete".

Please note: It is very important to ensure this resource is paused after creation, as it will incur costs if left running.

8. Create Azure Databricks Service

Azure Databricks is an Apache Spark-based analytics platform optimized for the Microsoft Azure cloud services platform.

Step	Setting	Value	Notes
1	Workspace Name	sqlbitsadf<999>adb	Use the name of the Resource Group and add adb as a suffix.
2	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the first step.
3	Location	UK South	Choose UK South .
4	Pricing Tier	Trial (Premium - 14 Days Free)	Choose Trial (Premium - 14 Days Free DBUs) . If this is no longer an option for you, choose Standard .

Unless specified above, use the default settings.

The screenshot shows the Azure portal interface for creating a new service. The path in the top navigation bar is Home > New > Marketplace > My Saved List > Azure Databricks > Azure Databricks Service. The main form has the following fields:

- * Workspace name: sqlbitsadfvcw01adb (highlighted by a green circle with number 1)
- * Subscription: Visual Studio Premium with MSDN (highlighted by a green circle with number 2)
- * Resource group:
 - Create new
 - Use existingsqlbitsadfvcw01 (highlighted by a green circle with number 3)
- * Location: UK South (highlighted by a green circle with number 3)
- * Pricing Tier: Trial (Premium - 14-Days Free DBUs) (highlighted by a green circle with number 4)
- Deploy Azure Databricks workspace in your Virtual Network (preview):
 - Yes
 - No

At the bottom of the form are two buttons: 'Create' and 'Automation options'.

Additional information:

<https://docs.azuredatabricks.net/getting-started/try-databricks.html>

9. Create Virtual Machine (Free SQL Server License: SQL Server 2017 Developer on Windows Server 2016 with AdventureWorksLT Demo Database

Create a virtual machine with SQL Server 2017 and the AdventureWorksLT database that will act as your on-premises source.

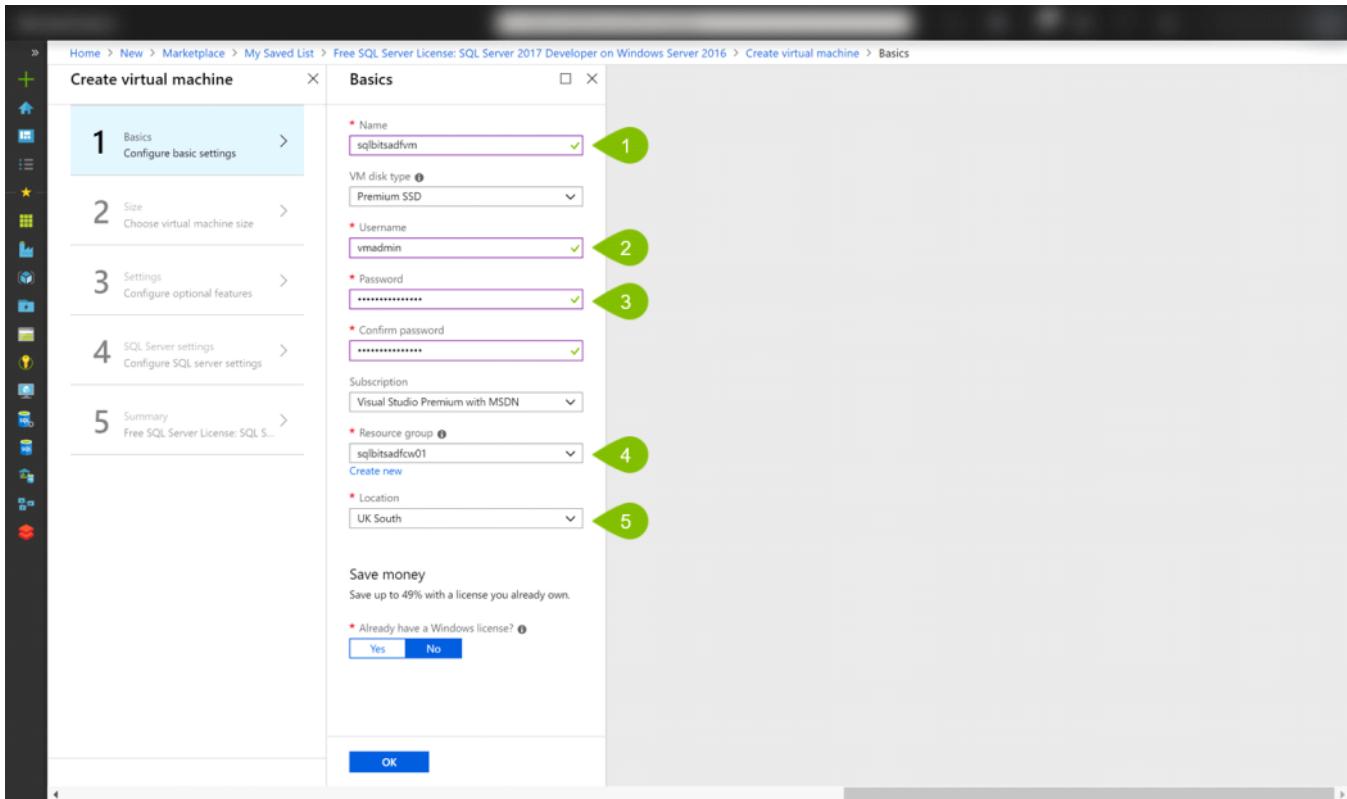
Please note: It is very important to ensure this resource is stopped after creation, as it will incur costs if left running.

This process consists of multiple parts.

Part 1: Basics

Step	Setting	Value	Notes
1	Name	sqlbitsadfvm	Please note that the name has to be between 1-15 characters, so you may not be able to use your chosen suffix.
2	Username	vmadmin	Pick a username, for example vmadmin
3	Password	VeryStr0ngPassw0rd!	Pick a password, for example VeryStr0ngPassw0rd!
4	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the first step.
5	Location	UK South	Choose UK South .

Unless specified above, use the default settings.



Part 2: Size

Choose **D2s_v3**.

Choose a size

Search: Compute type: Current generation Disk type: Premium vCPUs: 1 128

RECOMMENDED	SKU	TYPE	COMPUTE	vCPUS	GB RAM	DATA DISK	MAX IOPS	LOCAL SSDS	Premium	ADDITIONAL	ZONES	NOK/MO
B2ms	Standard	General purpose	2	8	4	2400	16 GB	Yes		1,2,3	NOK 569.88	
B2s	Standard	General purpose	2	4	4	1600	8 GB	Yes		1,2,3	NOK 284.94	
B4ms	Standard	General purpose	4	16	8	3600	32 GB	Yes		1,2,3	NOK 1,140.97	
B8ms	Standard	General purpose	8	32	16	4320	64 GB	Yes		1,2,3	NOK 2,281.94	
D16s_v3	Standard	General purpose	16	64	32	25600	128 GB	Yes		1,2,3	NOK 5,602.23	
D2s_v3	Standard	General purpose	2	8	4	3200	16 GB	Yes		1,2,3	NOK 700.28	
D4s_v3	Standard	General purpose	4	16	8	6400	32 GB	Yes		1,2,3	NOK 1,400.56	
D8s_v3	Standard	General purpose	8	32	16	12800	64 GB	Yes		1,2,3	NOK 2,801.12	
DS1_v2	Standard	General purpose	1	3.5	4	3200	7 GB	Yes		1,2,3	NOK 531.25	
DS11_v2	Standard	Memory optimized	2	14	8	6400	28 GB	Yes		1,2,3	NOK 1,412.63	
DS11-1_v2	Promo	Memory optimized	2	14	8	6400	28 GB	Yes		1,2,3	NOK 1,412.63	
DS12_v2	Promo	Memory optimized	4	28	16	12800	56 GB	Yes		1,2,3	NOK 2,831.30	
DS12_v2	Standard	Memory optimized	4	28	16	12800	56 GB	Yes		1,2,3	NOK 2,837.34	

Prices presented are estimates in your local currency that include only Azure infrastructure costs and any discounts for the subscription and location. The prices don't include any applicable software costs. Final charges will appear in your local currency in cost analysis and billing views. Recommended sizes are determined by the publisher of the selected image based on hardware and software requirements.

Select

Part 3: Settings

Select the RDP public inbound port. Keep all other defaults.

Settings

High availability

Availability zone: None

* Availability set: None

Storage

Use managed disks: Yes

Network

* Virtual network: (new) squbitsadfcw01-vnet

* Subnet: default (10.0.0.0/24)

* Public IP address: (new) squbitsadfvmlp

Network Security Group

Basic [Advanced]

* Select public inbound ports

- RDP (3389)
- No public inbound ports
- HTTP (80)
- HTTPS (443)
- SSH (22)
- RDP (3389)**

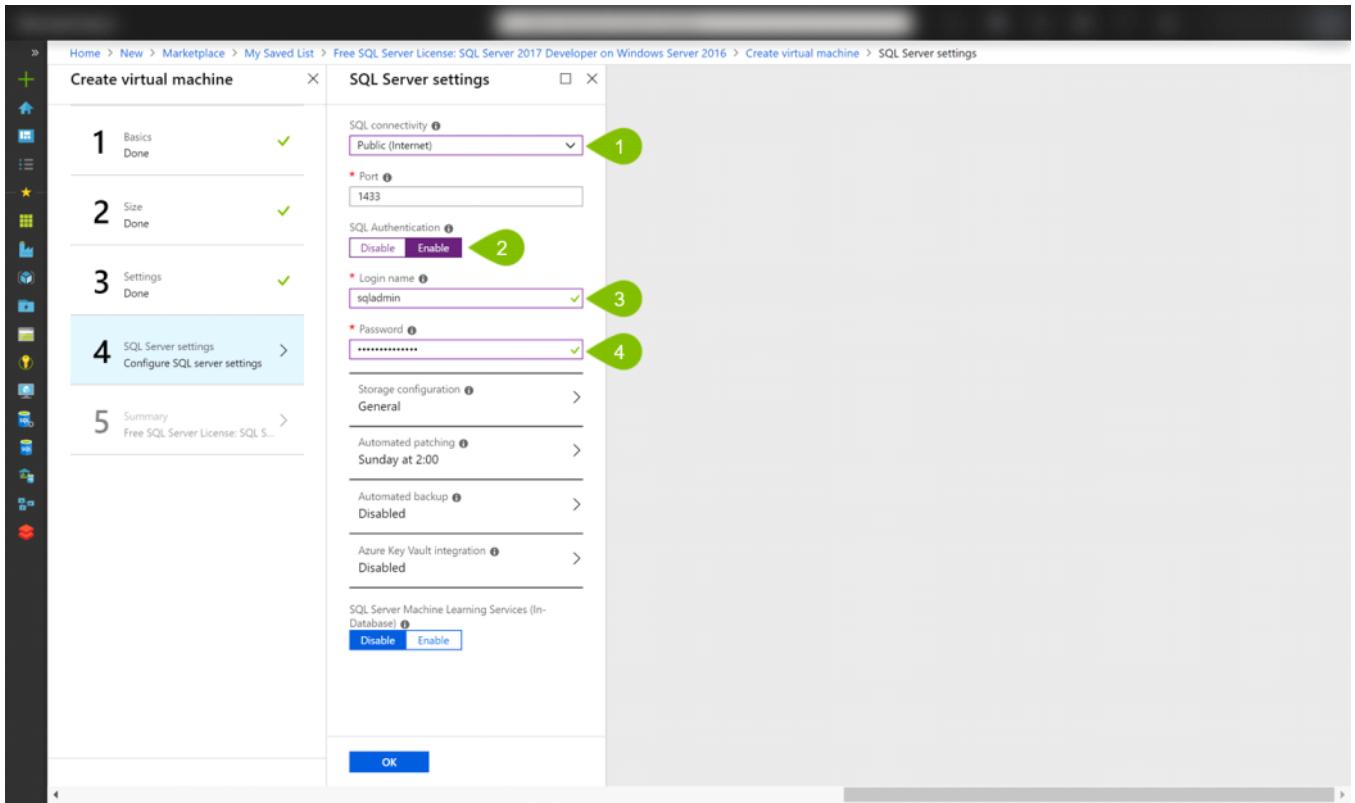
OK

Part 4: SQL Server Settings

Enable public access using SQL authentication.

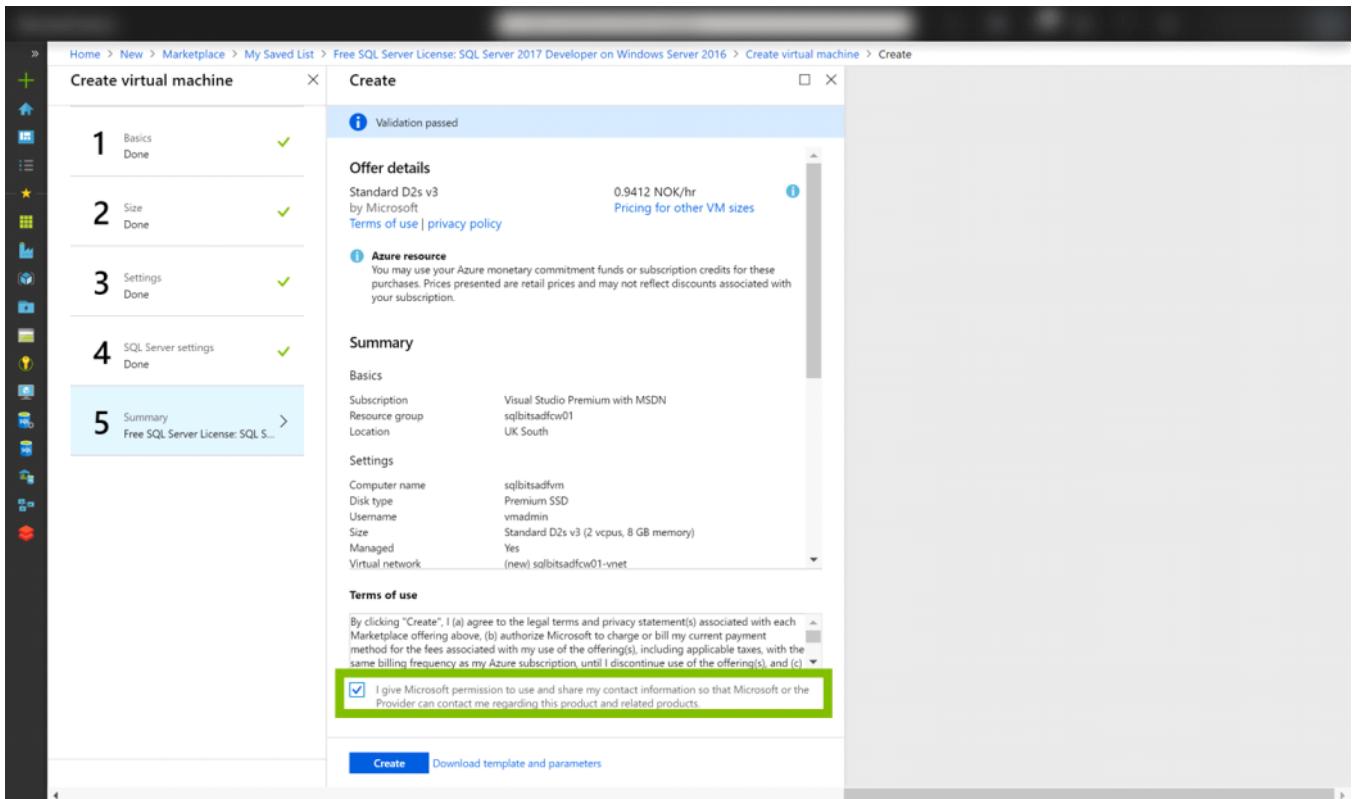
Step	Setting	Value	Notes
------	---------	-------	-------

1	SQL Connectivity	Public (Internet)	We will be using this public setting to simplify demos.
2	SQL Authentication	Enabled	Choose the Resource Group you created in the first step.
3	Username	sqladmin	Pick a username, for example sqladmin
4	Password	VeryStr0ngPassw0rd!	Pick a password, for example VeryStr0ngPassw0rd!



Part 5: Summary

Accept the terms of use and create the VM.



Change the Public IP Address from Dynamic to Static

In this workshop, we don't want to keep our Virtual Machine running all the time as that will incur costs. To be able to start and stop the virtual machine and reconnect to it without having to change the connection settings every time, you need to change the IP address settings. From the Virtual Machine page, click on the **Public IP Address**.

Setting	Value
Computer name	: sqlbitsadfv01
Operating system	: Windows
Size	: Standard D2s v3 (2 vcpus, 8 GB memory)
Public IP address	51.140.105.31
Virtual network/subnet	: sqlbitsadfcw01-vnet/default
DNS name	: Configure

Change the Assignment from **Dynamic** to **Static**. Click **Save**.

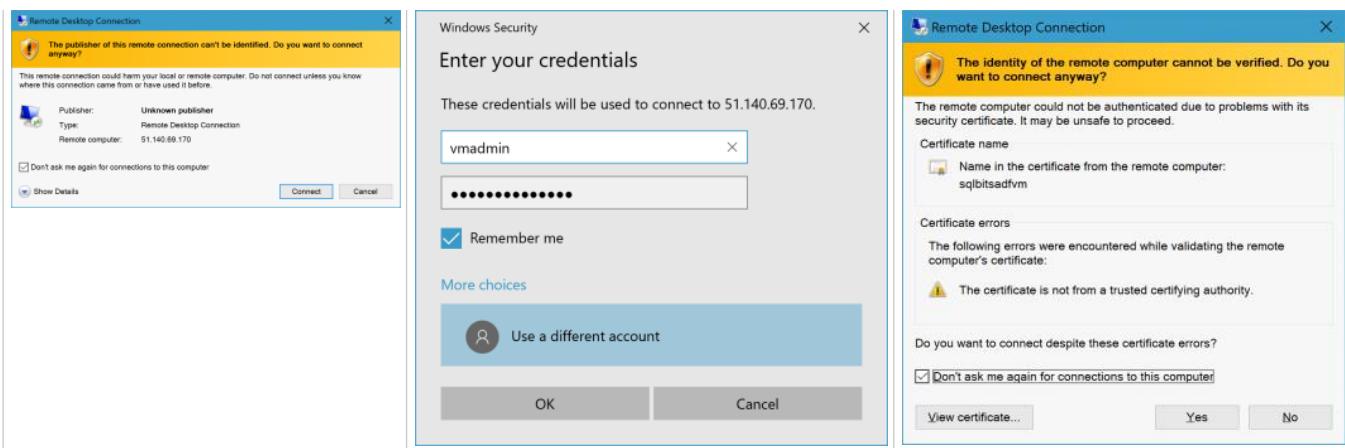
The screenshot shows the Microsoft Azure portal interface. The left sidebar lists various services like Overview, Activity log, Access control (IAM), Tags, Configuration, Properties, Locks, Automation script, Support + troubleshooting, and New support request. The main content area is titled "sqlbitsadfvvm-ip - Configuration". It shows a warning message: "The associated virtual machine 'sqlbitsadfvvm' may be rebooted. Click here to learn more." Below this, the IP address is set to "Static" (radio button selected) with the value 51.140.105.31. There is also an "Idle timeout (minutes)" slider set to 4. A "DNS name label (optional)" field contains ".uksouth.cloudapp.azure.com". Under "Alias record sets", there is a link to "Create alias record". A table shows "No results." for DNS zone entries.

Connect to the Virtual Machine

Go back to the Virtual Machine page. Click **Connect** and download the **RDP** file.

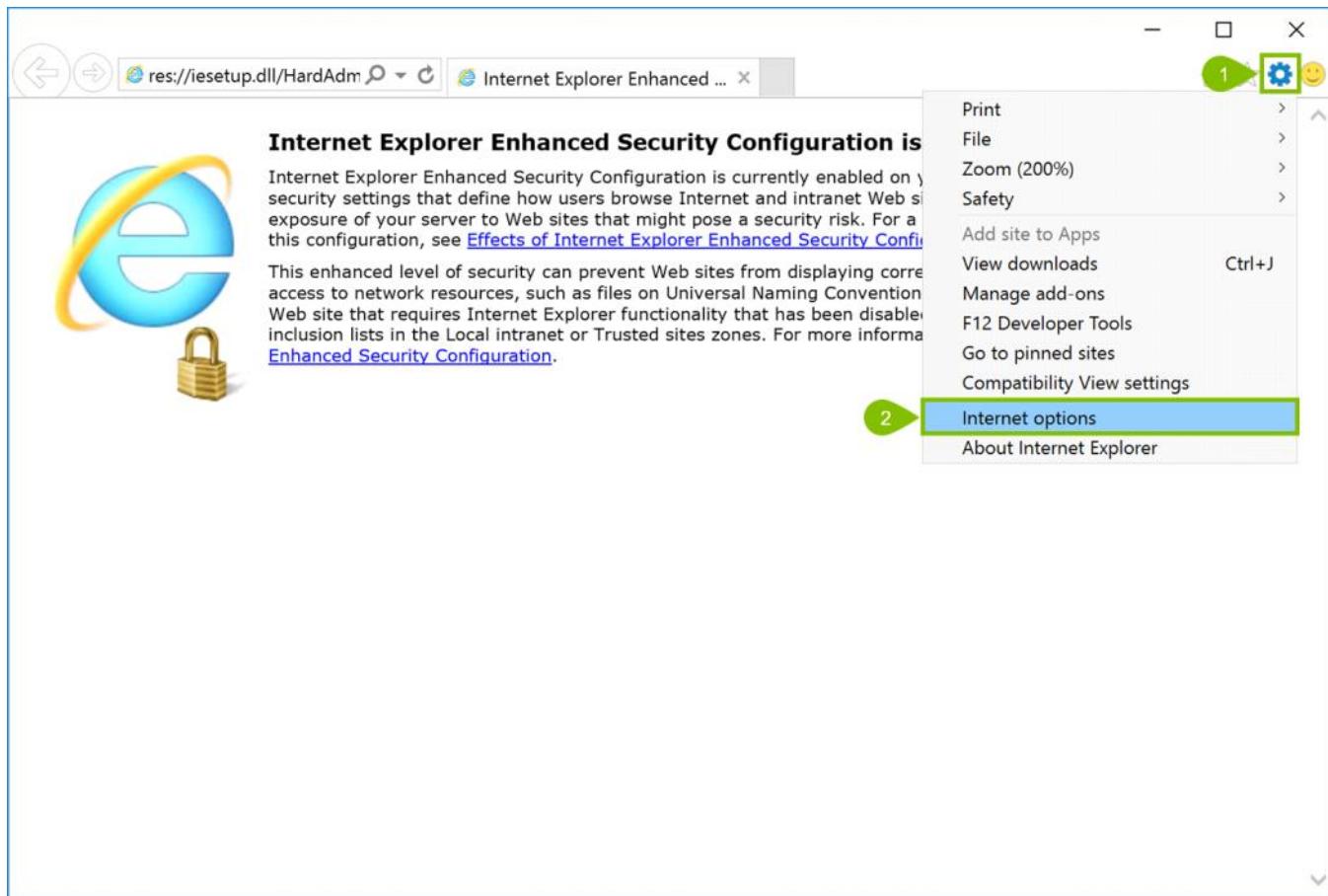
The screenshot shows the Microsoft Azure portal interface. The left sidebar lists Overview, Activity log, Access control (IAM), Tags, Diagnose and solve problems, Networking, Disks, Size, Security, Extensions, Continuous delivery (Preview), Availability set, Configuration, and Identity. The main content area shows details for the virtual machine "sqlbitsadfvvm": Resource group (sqlbitsadfcw01), Status (Running), Location (UK South), Subscription (Visual Studio Premium with MSDN), and Subscription ID. On the right, a modal window titled "Connect to virtual machine" for "sqlbitsadfvvm" is displayed. It contains a warning: "To improve security, enable just-in-time access on this VM." Below this, there are tabs for "RDP" (selected) and "SSH". The RDP section includes fields for "IP address" (Public IP address 51.140.69.170) and "Port number" (3389). A button labeled "Download RDP File" is highlighted with a green circle labeled "2". Below the RDP section, there is a note about inbound traffic and networking, and another note about troubleshooting connection issues.

Open the RDP file, click Connect, then choose **Use a different account** under More choices, enter your username and password (for example **vmadmin / VeryStr0ngPassw0rd!**), and click Yes.

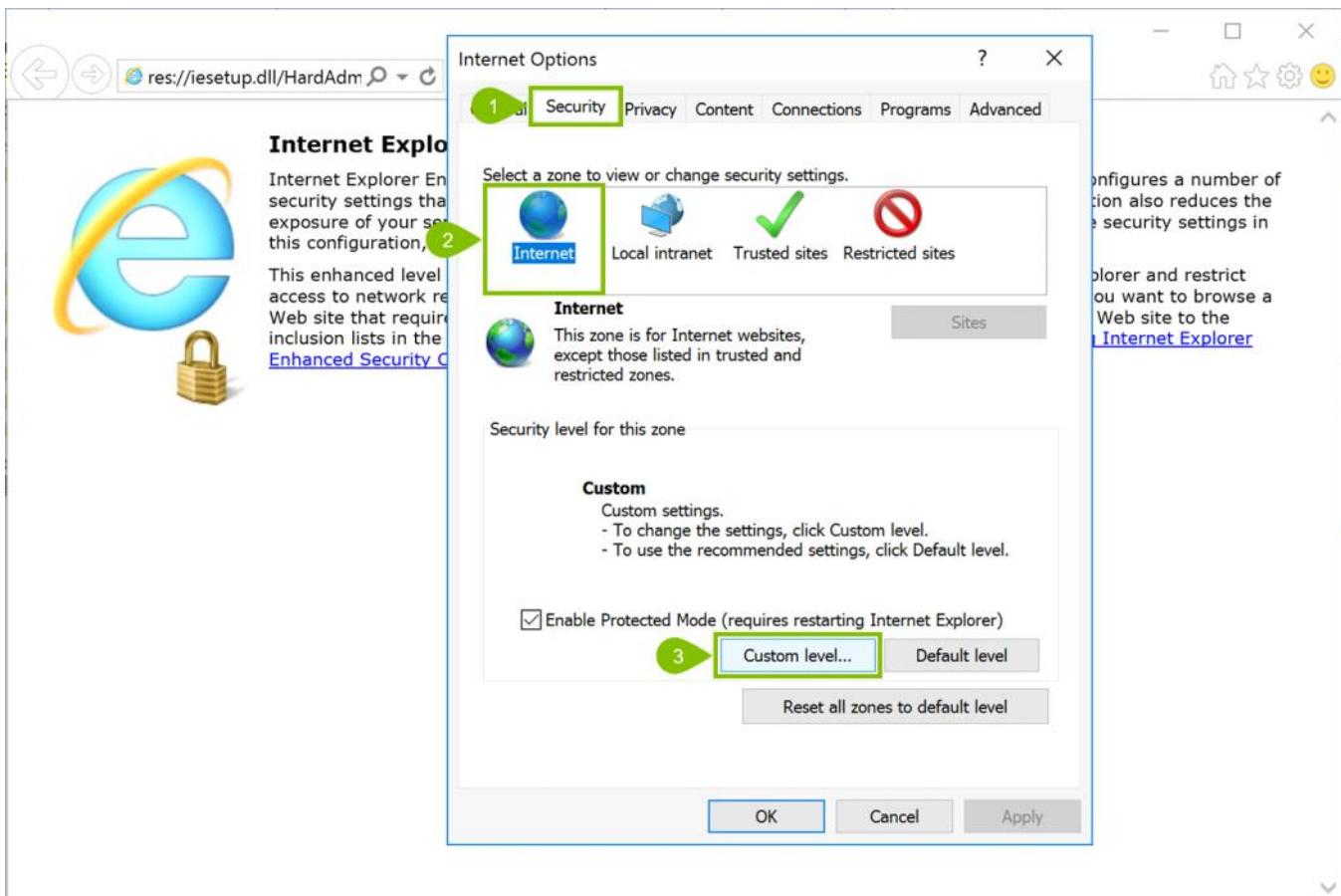


Download the AdventureWorksLT Demo Database

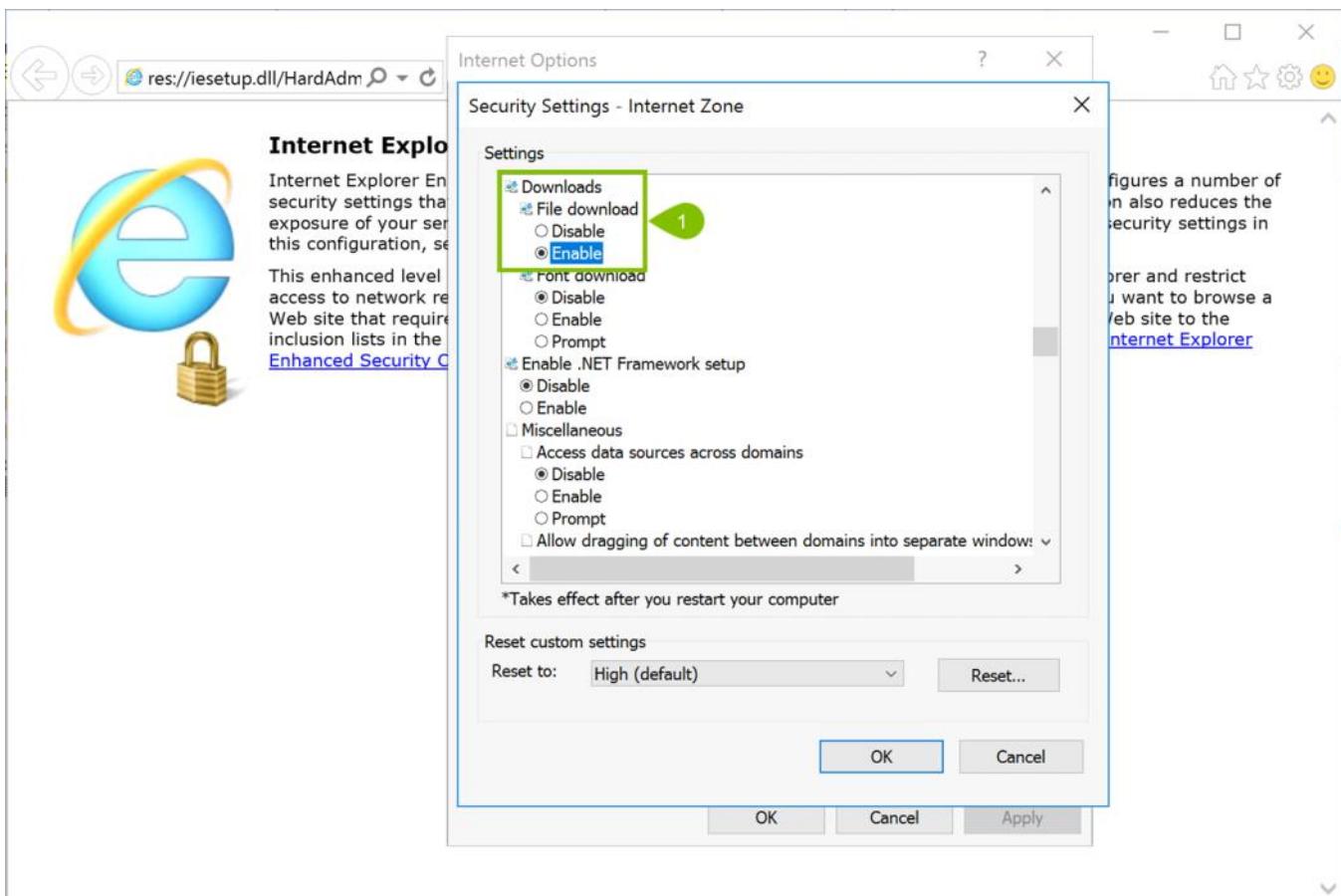
Once inside the Virtual Machine, open Internet Explorer. To be able to download the AdventureWorksLT demo database, you first need to change the security settings. Click **Settings**, then **Internet Options**.



Click the **Security** tab, select **Internet**, then click **Custom level**.



Scroll down to **Downloads**, choose **Enable**.



Scroll further down to **Scripting**, choose **Enable**. Click Yes to confirm you want to change the settings for the zone, click OK to close the Settings windows, and OK again to close the Internet Options window.

Go to <https://github.com/Microsoft/sql-server-samples/releases>, scroll down, and download **AdventureWorksLT2017.bak** to D:\.

The screenshot shows a web browser window displaying the GitHub releases page for the 'sql-server-samples' repository. The main title is 'AdventureWorksLT (Lightweight) full database backups'. Under this section, the file 'AdventureWorksLT2017.bak' is highlighted with a green box and a green arrow labeled '1' pointing to it. Below it are other backup files: 'AdventureWorksLT2016.bak', 'AdventureWorksLT2014.bak', and 'AdventureWorksLT2012.bak'. A second section titled 'AdventureWorksDW (Data Warehouse) full database backups' follows, containing 'AdventureWorksDW2017.bak', 'AdventureWorksDW2016.bak', and 'AdventureWorksDW2016_EXT.bak'. A descriptive note at the bottom states: 'An extended version of AdventureWorksDW2016 designed to showcase SQL Server 2016 features. To see the features in action, run the [SQL Server](#)'.

Open **SQL Server Management Studio (SSMS)**, connect to **localhost**, and restore the **AdventureWorksLT** database.

```
USE [master];
GO

RESTORE DATABASE [AdventureWorksLT]
FROM DISK = N'D:\AdventureWorksLT2017.bak'
WITH FILE = 1,
MOVE N'AdventureWorksLT2012_Data' TO N'F:\Data\AdventureWorksLT2012.mdf',
MOVE N'AdventureWorksLT2012_Log' TO N'F:\Log\AdventureWorksLT2012_log.ldf',
NOUNLOAD,  STATS = 5
GO
```

Open **SQL Server Management Studio (SSMS)**, connect to **localhost**, and run the following query in the **master** database to set up security and permissions.

```
USE [master];
GO

CREATE LOGIN DataLoadUser WITH PASSWORD = 'DataFactoryD3m0!';

USE AdventureWorksLT
GO
```

```

CREATE USER DataLoadUser FOR LOGIN DataLoadUser;

CREATE ROLE Executor;
GRANT EXECUTE TO [Executor]

CREATE ROLE Reader;
GRANT SELECT TO [Reader];

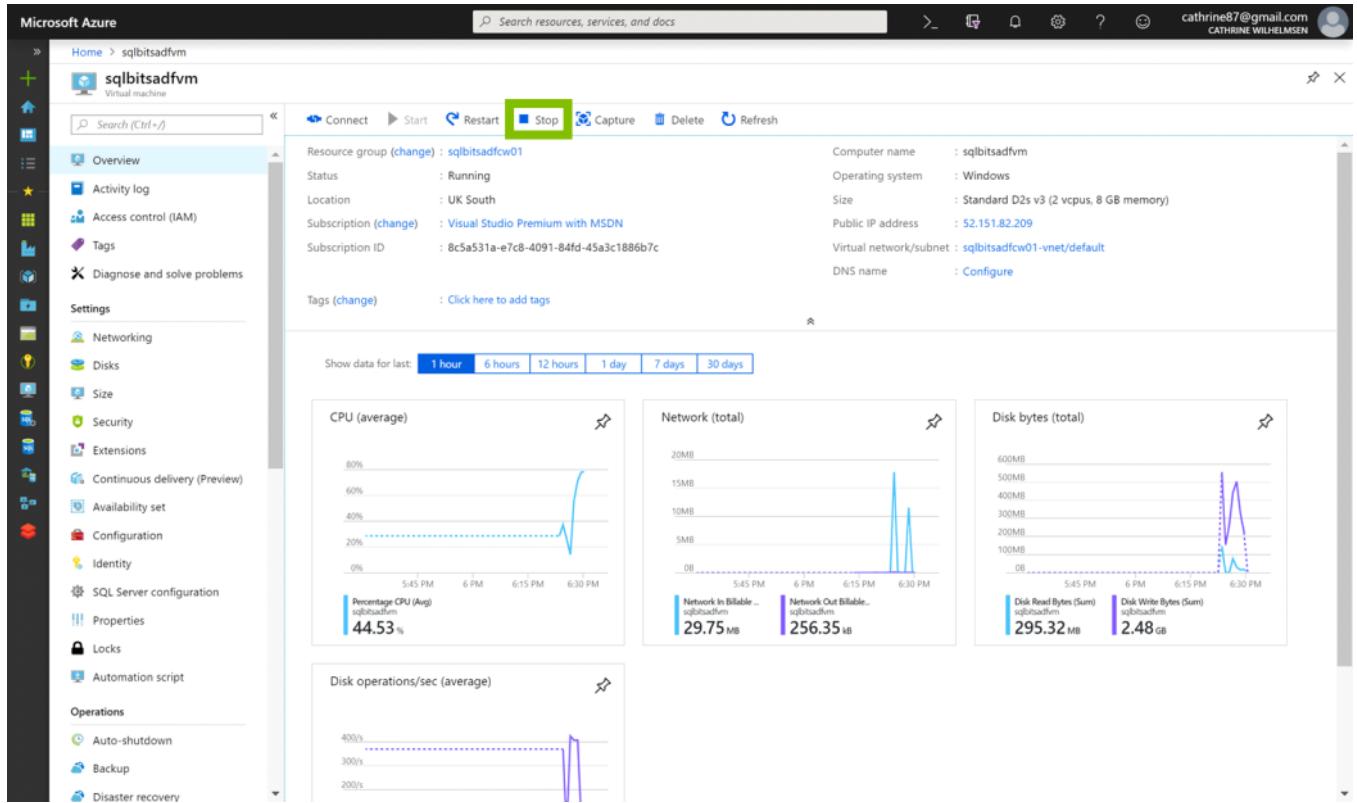
EXEC sp_addrolemember 'Executor', 'Reader';

GRANT VIEW CHANGE TRACKING ON Schema::SalesLT TO [Reader];
GRANT VIEW DEFINITION ON Database::AdventureWorksLT to [Reader];

EXEC sp_addrolemember 'Reader', 'DataLoadUser';

```

Once the AdventureWorksLT database has been restored, exit the Virtual Machine and stop it.



Additional information:

<https://docs.microsoft.com/en-us/azure/virtual-machines/windows/sql/quickstart-sql-vm-create-portal>

Please note: It is very important to ensure this resource is stopped after creation, as it will incur costs if left running.

Post-Setup: Ensure all Resources are Paused or Stopped

	<ul style="list-style-type: none"> • Azure SQL Data Warehouse (<i>Paused</i>) • Azure Analysis Service (<i>Paused</i>) • Virtual Machine (<i>Stopped</i>)
--	--

Lab 01: Azure Data Factory Overview

Background and Goals

In this lab, you will learn how to create your first Azure Data Factory, as well as the foundational Linked Services, Datasets, and Integration Runtimes.

After you create all the necessary resources, you will build a simple pipeline that copies data from a SQL Server source to an Azure Data Lake Storage Gen1 sink. Finally, you will debug and trigger the newly created pipeline.

Prerequisites

In order to complete this lab, you will need to have previously created:

- Resource Group
- Azure Data Lake Storage Gen1
- Azure SQL Server
- Azure SQL Database
- Virtual Machine (Free SQL Server License: SQL Server 2017 Developer on Windows Server 2016) with AdventureWorksLT Demo Database

Overview

Activity 01: Create Azure Data Factory and Configure Security

- Create Azure Data Factory
- Run PowerShell Script to Create Azure Active Directory Group

Activity 02: Setup Integration Runtime

- Create Self-Hosted Integration Runtime in the Azure Data Factory
- Install Integration Runtime on Virtual Machine (simulating an on-premises SQL Server)

Activity 03: Setup Staging Database

- Deploy Staging database from DACPAC
- Setup Master and Staging Security

Activity 04: Create Linked Services

- Create Linked Service to SQL Server in Virtual Machine
- Create Linked Service to Azure SQL Database
- Create Linked Service to Azure Data Lake Storage Gen1

Activity 05: Create Datasets

- Create SQL Server Dataset
- Create Azure Data SQL Database Dataset

Activity 06: Create Pipeline

- Create Pipeline

Activity 07: Execute Pipeline

- Debug Pipeline
- Trigger Pipeline

- Monitor Pipeline

Let's get started with Activity 01!

Activity 01: Create Azure Data Factory and Configure Security

Activity Overview

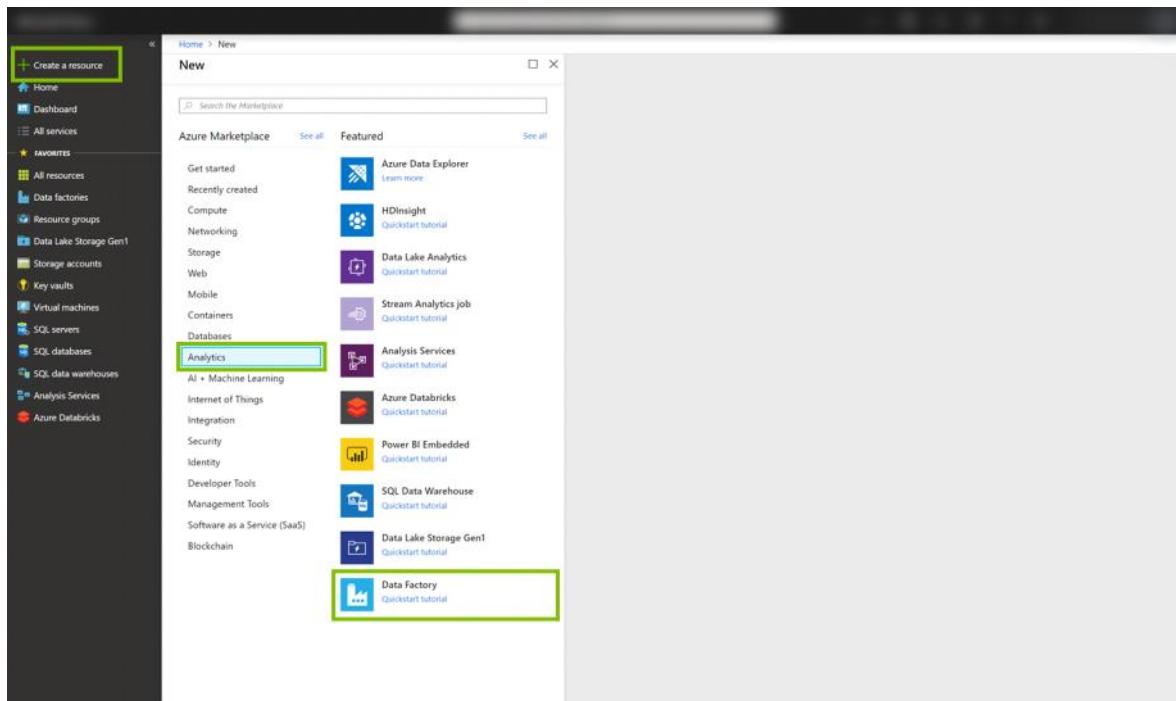
Estimated time to complete activity: **5 minutes**.

In this activity, you will create an Azure Data Factory. This activity is, not surprisingly, required to complete the rest of the labs and activities in the workshop.

Create Azure Data Factory

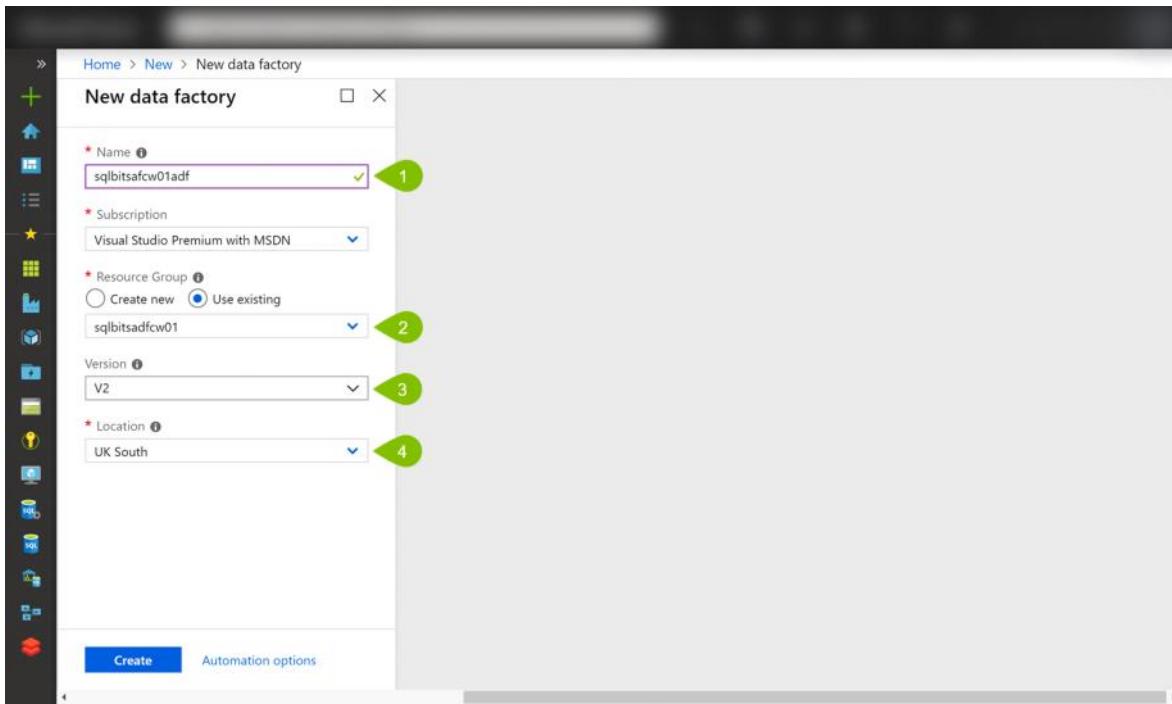
Open **Microsoft Edge** or **Google Chrome** and navigate to <http://portal.azure.com>.

Click **Create a resource** in the left menu, click **Analytics**, and then select **Data Factory**:



On the **New Data Factory** page, provide the following information:

Step	Setting	Value	Notes
1	Name	sqlbitsadf<999>adf	Use the name of the Resource Group and add adf as a suffix.
2	Resource Group	sqlbitsadf<999>	Choose the Resource Group you created in the prerequisites instructions.
3	Version	V2	Select V2 .
4	Location	UK South	Choose UK South .



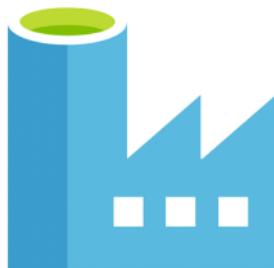
Navigate to the Azure Data Factory

There are two ways to navigate to your new Azure Data Factory:

1. From the Azure Data Factory resource page, click on **Author & Monitor**:

The screenshot shows the Azure Data Factory resource page for 'sqlbitsa fcw01adf'. The 'Author & Monitor' button is highlighted with a green box.

2. Go directly to <https://adf.azure.com> and select your new Azure Data Factory:



Select Data Factory

Microsoft Azure Data Factory is a cloud-based data integration service that automates data movement and transformation. More details can be found [here](#).

Azure Active Directory *

Subscription

Data Factory Name *

Continue

You will now see the **Let's get started** page:

Azure Data Factory

Let's get started

- Create pipeline
- Create pipeline from template
- Copy Data
- Configure SSIS Integration
- Set up Code Repository

Videos

- Overview of Azure Data Factory
- Run Azure Functions from Azure Data Factory pipelines
- Monitor your Azure Data Factory pipelines proactively with alerts

[View All Videos](#)

Run PowerShell Script to Create Azure Active Directory Group

To access your resources from Azure Data Factory, you need to create an Azure Active Directory Group.

Open a **Cloud Shell session** from the Azure Portal and choose **PowerShell**.

The screenshot shows the Microsoft Azure portal homepage. At the top, there's a search bar and a navigation bar with icons for Home, Dashboard, and Save. Below the search bar, a green callout box labeled '1' points to the 'Cloud Shell' icon. The main content area is titled 'Azure services' and 'Make the most out of Azure'. It features several service icons like Virtual machines, Storage accounts, App Services, etc. Below these are five cards: 'Learn Azure with free online courses by Microsoft', 'Monitor your apps and infrastructure', 'Secure your apps and infrastructure', 'Optimize performance, reliability, security, and costs', and 'Connect to Azure via an authenticated browser-based shell'. A green callout box labeled '2' points to the 'PowerShell' tab in the Cloud Shell interface.

If this is the first time you open a Cloud Shell session, you need to **Create Storage**.

The screenshot shows the Microsoft Azure portal homepage. A green callout box labeled '1' points to the 'Cloud Shell' icon in the top navigation bar. The main content area is titled 'Azure services' and 'Make the most out of Azure'. It features several service icons like Virtual machines, Storage accounts, App Services, etc. Below these are five cards: 'Learn Azure with free online courses by Microsoft', 'Monitor your apps and infrastructure', 'Secure your apps and infrastructure', 'Optimize performance, reliability, security, and costs', and 'Connect to Azure via an authenticated browser-based shell'. A green callout box labeled '2' points to the 'Create storage' button in a modal dialog box that says 'You have no storage mounted'.

Once the shell is ready, paste the following script and hit enter.

```
$resourceGroupName = "sqlbitsadf<999>" # Use the name of the Resource Group created in the prerequisites
$dataFactoryName = "sqlbitsadf<999>adf" # Use the name of the Azure Data Factory you just created
$groupName = "ADF_Access" # Leave as-is, this is the default name of the Azure Active Directory Group

$myRg = Get-AzResourceGroup $resourceGroupName
$myDf = Get-AzDataFactoryV2 -ResourceGroupName $myRg.ResourceGroupName -Name $dataFactoryName
$mySp = Get-AzADServicePrincipal -ObjectId $($myDf.Identity.PrincipalId)
$group = New-AzADGroup -DisplayName $groupName -MailNickname $groupName

Add-AzureAdGroupMember -ObjectId $Group.Id -RefObjectId $mySp.Id
```

The screenshot shows the Microsoft Azure portal interface. In the top navigation bar, it says "Microsoft Azure" and "Home > Catherine WilhelmSEN > Groups - All groups". On the left, there's a sidebar with icons for Home, Groups, Storage, Functions, and more. The main content area is titled "Groups - All groups" under "Catherine WilhelmSEN - Azure Active Directory". It shows a table with columns: NAME, GROUP TYPE, and MEMBERSHIP TYPE. A search bar at the top right says "Search groups". Below the table, it says "No groups found". To the left of the table, there's a sidebar with "All groups" and "Deleted groups" options, and "Settings" with "General" and "Expiration" tabs. At the bottom of the screen, a PowerShell window is open with the following command history:

```
PowerShell | ⌂ ? ⌂ ⌂ ⌂ { }  
VERBOSE: Authenticating to Azure ...  
VERBOSE: Building your Azure drive ...  
Azure:/  
PS Azure:\> $resourceGroupName = "sqlbitsadfcw01" # Use the name of the Resource Group created in the prerequisites  
Azure:/  
PS Azure:\> $dataFactoryName = "sqlbitsadfcw01adf" # Use the name of the Azure Data Factory you just created  
Azure:/  
PS Azure:\> $groupName = "ADF_Access" # Leave as-is, this is the default name of the Azure Active Directory Group  
Azure:/  
PS Azure:\>  
Azure:/  
PS Azure:\> $myRg = Get-AzResourceGroup $resourceGroupName  
Azure:/  
PS Azure:\> $myDF = Get-AzDataFactoryV2 -ResourceGroupName $myRg.ResourceGroupName -Name $dataFactoryName  
Azure:/  
PS Azure:\> $mySp = Get-AzADServicePrincipal -ObjectId $($myDF.Identity.PrincipalId)  
Azure:/  
PS Azure:\> $group = New-AzADGroup -DisplayName $groupName -MailNickname $groupName  
Azure:/  
PS Azure:\>  
PS Azure:\> Add-AzureAdGroupMember -ObjectId $Group.Id -RefObjectId $mySp.Id
```

This creates an Azure Active Directory group with one member called ADF_Access.

Activity Summary

You have now created your Azure Data Factory! Next, you will setup an Integration Runtime.

Activity 02: Setup Integration Runtime

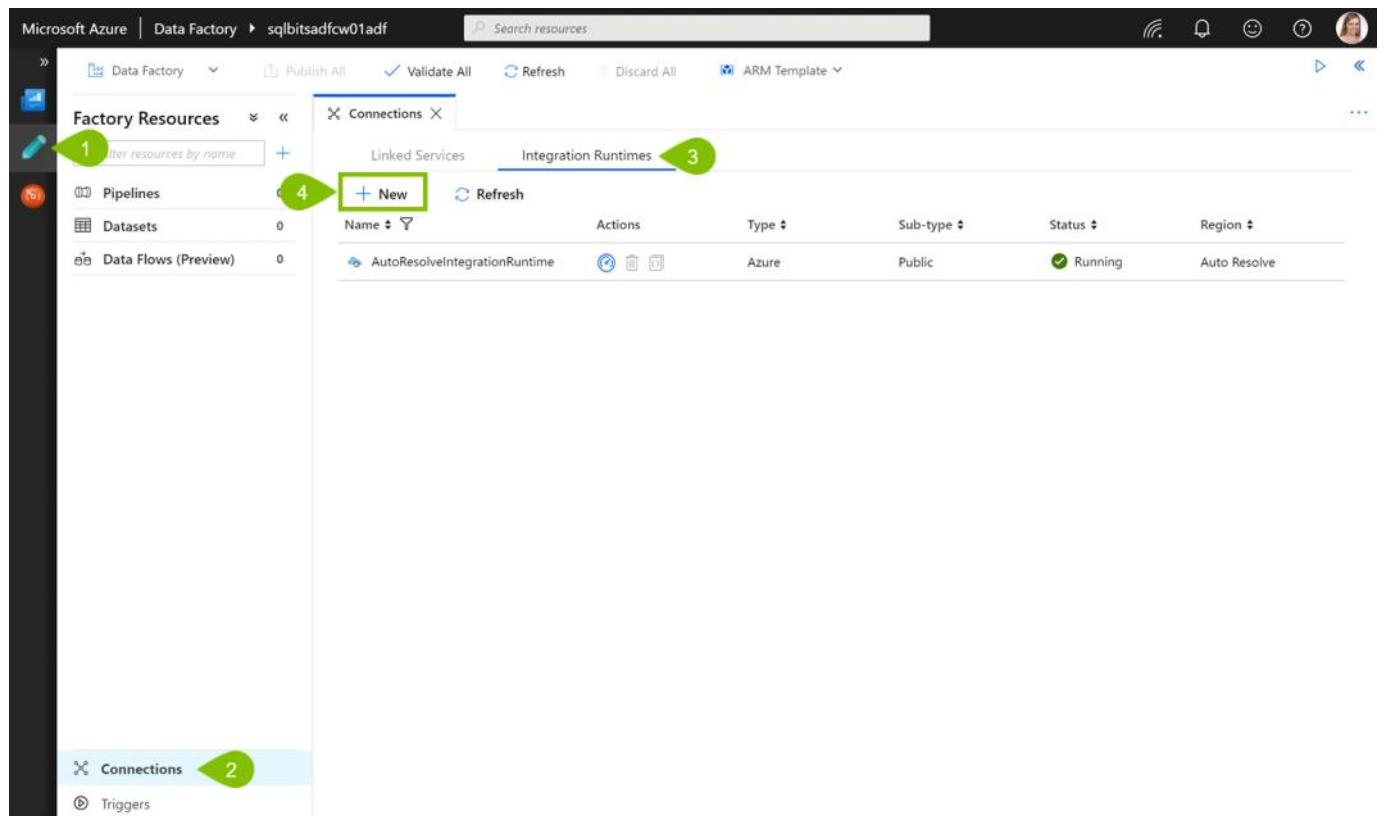
Activity Overview

Estimated time to complete activity: **5-10 minutes**.

In this activity, you will create an Integration Runtime that connects to the Virtual Machine you created in the prerequisites.

Create the Self-Hosted Integration Runtime

1. Click the **Author (pencil)** icon
2. Click on **Connections**
3. Click on **Integration Runtimes**
4. Click **New**



5. Select **Perform data movement and dispatch activities to external computes** and click **Next**

Integration Runtime Setup

Integration Runtime is the native compute used by ADF to execute or dispatch activities. Choose what integration runtime to create based on required capabilities.

- External Compute**: Perform data movement and dispatch activities to external computes.
- SSIS Integration Runtime**: Lift-and-shift existing SSIS packages to execute in Azure.

Cancel **Next →**

6. Select **Self-Hosted** and click **Next**

Integration Runtime Setup

Choose the network environment of the data source/destination or external compute to which the integration runtime will connect to for data movement or dispatch activities:

- Azure Public**
- Self-Hosted** (selected)
- Linked Self-Hosted**

External Resources:
You can use an existing self-hosted integration runtime that exists in another Data Factory. This way you can reuse your existing infrastructure where self-hosted integration runtime is setup.

Cancel **← Previous** **Next →**

7. Enter the name **SelfHostedIntegrationRuntime** and click **Next**

Integration Runtime Setup

Private network support is realized by installing integration runtime to machines in the same on-premises network/VNET as the resource the integration runtime is connecting to. Follow below steps to register and install integration runtime on your self-hosted machines.

Name *
SelfHostedIntegrationRuntime

Description
Enter description here...

Type
Self-Hosted

Cancel <- Previous Next →

- Once the Integration Runtime has been created in the Azure Data Factory, you need to manually download and install the integration runtime in your Virtual Machine. Make note that you will find your **Authentication Keys** here, and **leave this window/tab open**.

Integration Runtime Setup

Settings Nodes Auto update Sharing

Install integration runtime on Windows machine or add further nodes using the Authentication Key.

Name
SelfHostedIntegrationRuntime

Option 1: Express setup
[Click here to launch the express setup for this computer](#)

Option 2: Manual setup

Step 1: Download and install integration runtime
Step 2: Use this key to register your integration runtime

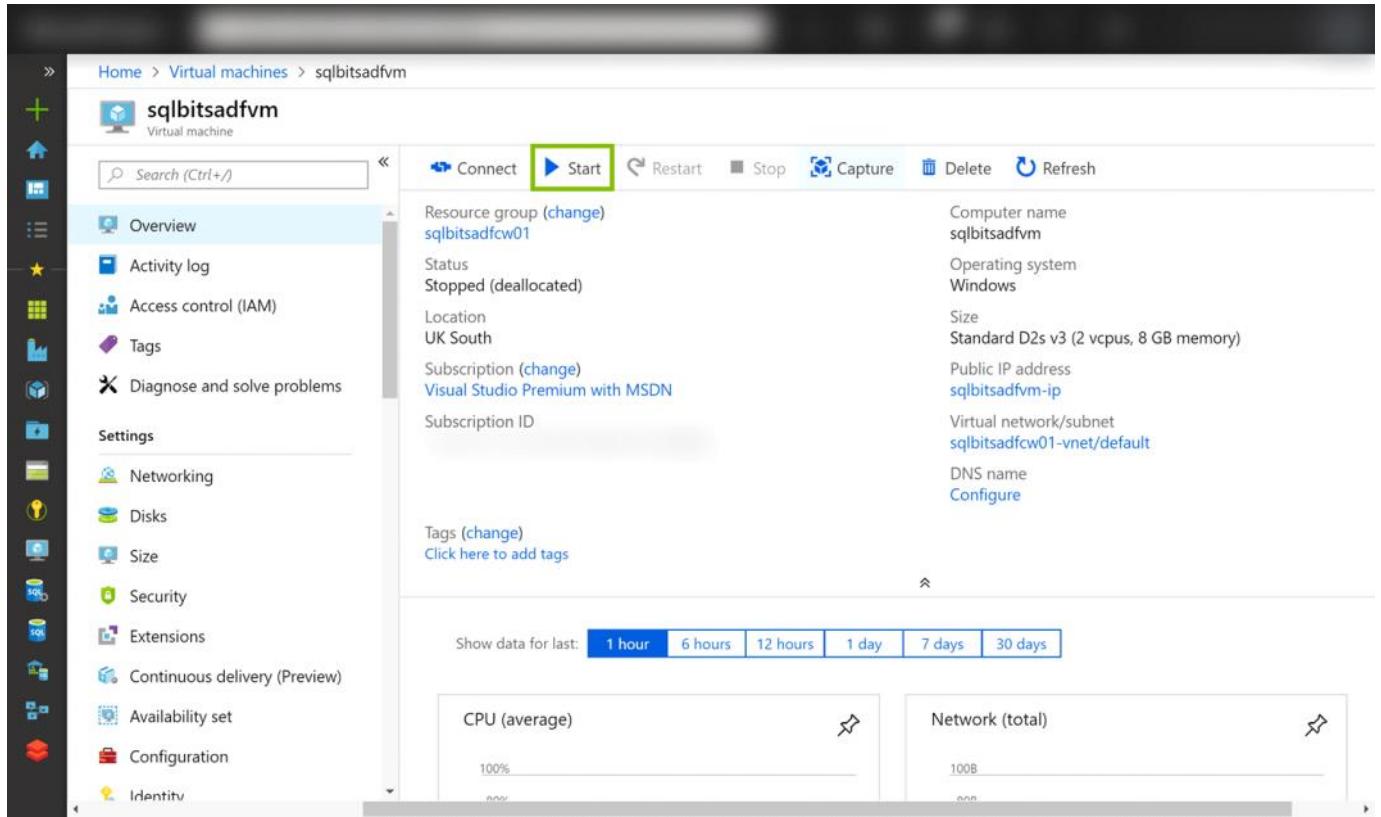
Name	Authentication Key
Key1	IR@acb38cb9-e178-4a32-8845-4ac504c763c9@sqlbitsadfcw01adf@uks@oF
Key2	IR@acb38cb9-e178-4a32-8845-4ac504c763c9@sqlbitsadfcw01adf@uks@QI

Finish

Start the Virtual Machine

In a different window/tab, go to <http://portal.azure.com> and navigate to the Virtual Machine you created in the prerequisites.

Start the Virtual Machine.



The screenshot shows the Azure portal interface for managing a virtual machine named 'sqlbitsadfvm'. The main navigation bar at the top includes 'Home', 'Virtual machines', and the specific machine name 'sqlbitsadfvm'. Below the navigation bar, there are several tabs: 'Overview' (which is selected and highlighted in blue), 'Activity log', 'Access control (IAM)', 'Tags', 'Diagnose and solve problems', 'Settings' (with sub-options like 'Networking', 'Disks', 'Size', 'Security', 'Extensions', 'Continuous delivery (Preview)', 'Availability set', 'Configuration', and 'Identity'), and a search bar labeled 'Search (Ctrl+.)'. To the right of the tabs, there are several management actions: 'Connect', 'Start' (which is highlighted with a green box), 'Restart', 'Stop', 'Capture', 'Delete', and 'Refresh'. Below these actions, detailed information about the VM is listed, including its resource group ('sqlbitsadfcw01'), status ('Stopped (deallocated)'), location ('UK South'), subscription ('Visual Studio Premium with MSDN'), computer name ('sqlbitsadfvm'), operating system ('Windows'), size ('Standard D2s v3 (2 vcpus, 8 GB memory)'), public IP address ('sqlbitsadfvm-ip'), virtual network/subnet ('sqlbitsadfcw01-vnet/default'), and DNS name ('Configure'). A section for 'Tags (change)' allows adding tags, with a link 'Click here to add tags'. At the bottom of the main content area, there is a time range selector for monitoring data: 'Show data for last:' followed by buttons for '1 hour', '6 hours', '12 hours', '1 day', '7 days', and '30 days'. Below this, two performance metrics are displayed: 'CPU (average)' and 'Network (total)'. The CPU chart shows a value of 100%, and the Network chart shows a value of 100B.

Click **Connect** and download the **RDP** file.

The screenshot shows the Azure portal interface for a virtual machine named 'sqlbitsadfvm'. On the left, there's a sidebar with various settings like Overview, Activity log, and Networking. The main area displays basic VM information: Resource group (sqlbitsadfcv01), Status (Running), Location (UK South), Subscription (Visual Studio Premium with MSDN), and Tags. Below this, there's a chart for CPU usage. On the right, a 'Connect to virtual machine' panel is open, specifically the 'RDP' tab. It includes fields for IP address (Public IP address 51.140.69.170) and Port number (3389). A prominent blue button labeled 'Download RDP File' is highlighted with a green circle and labeled '2'.

Open the RDP file, click Connect, then choose **Use a different account** under More choices, enter your username and password (for example **vmadmin / VeryStr0ngPassw0rd!**), and click Yes.

The three screenshots show the sequence of connecting to the remote desktop. The first window is a warning from 'Remote Desktop Connection' about the publisher being unknown. The second window is 'Windows Security' asking for credentials, with 'vmadmin' entered and 'Remember me' checked. The third window is another 'Remote Desktop Connection' dialog asking if you want to connect despite certificate errors, with 'View certificate...' and 'Yes' options available.

Download the Azure Data Factory Integration Runtime

Once inside the Virtual Machine, open Internet Explorer. Navigate to <https://www.microsoft.com/en-us/download/details.aspx?id=39717> and download the Azure Data Factory Integration Runtime.

The screenshot shows a Microsoft web page for downloading the Azure Data Factory Integration Runtime. At the top, there's a navigation bar with back, forward, and search icons, followed by the URL 'https://www.microsoft.com/en-us/download...' and a progress bar indicating the download is 0% complete. Below the header, the title 'Azure Data Factory Integration Runtime' is displayed. A note below it says 'Important! Selecting a language below will dynamically change the complete page content to that language.' A large button labeled 'Choose the download you want' is present. Underneath, there's a table showing two download options:

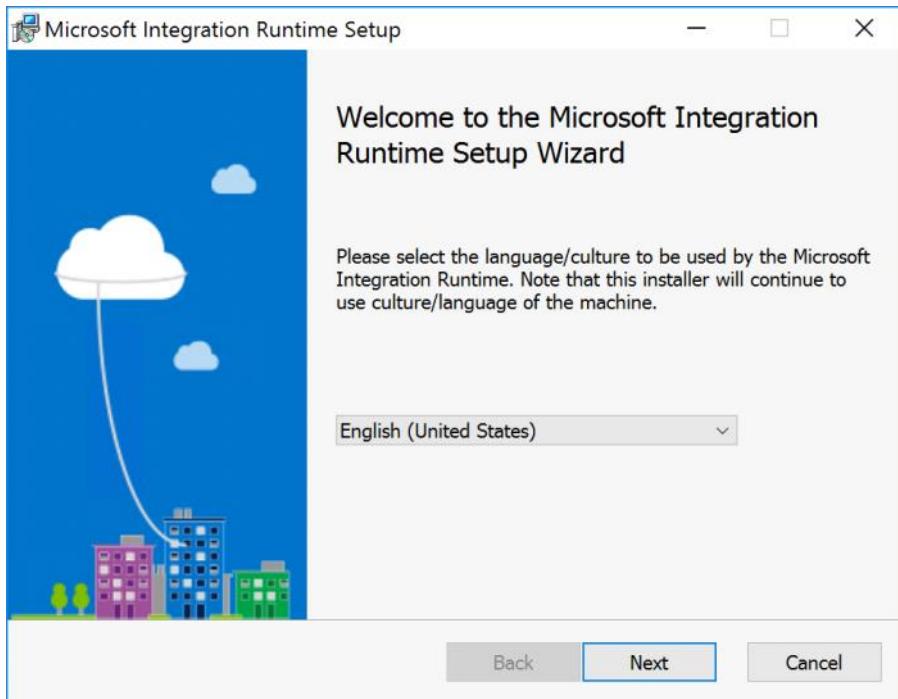
File Name	Size
<input checked="" type="checkbox"/> IntegrationRuntime_3.13.6942.1 (64-bit).msi	242.3 MB
<input type="checkbox"/> Release Notes.doc	123 KB

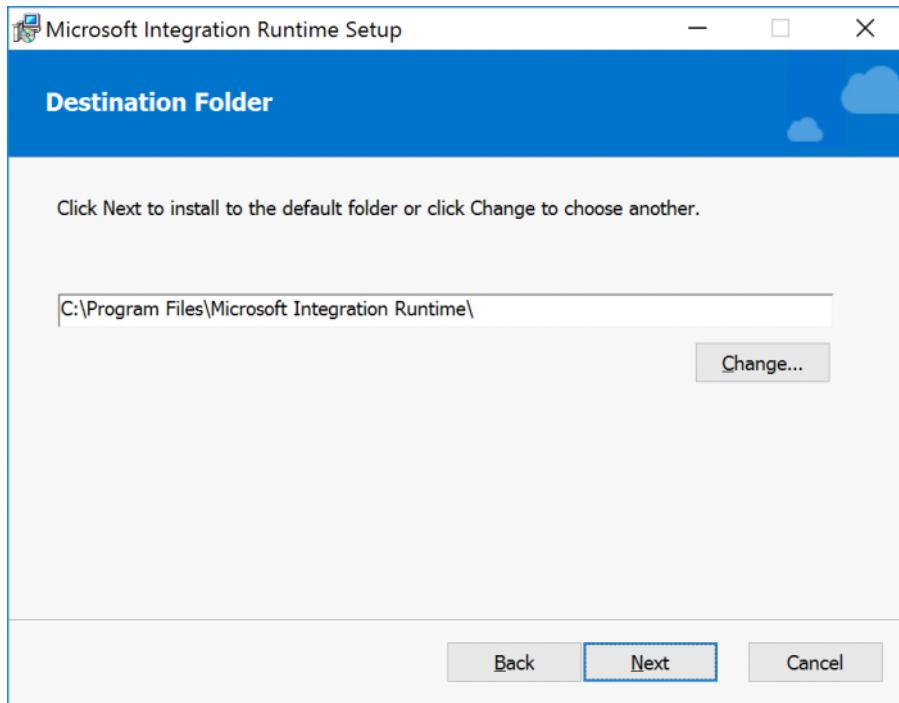
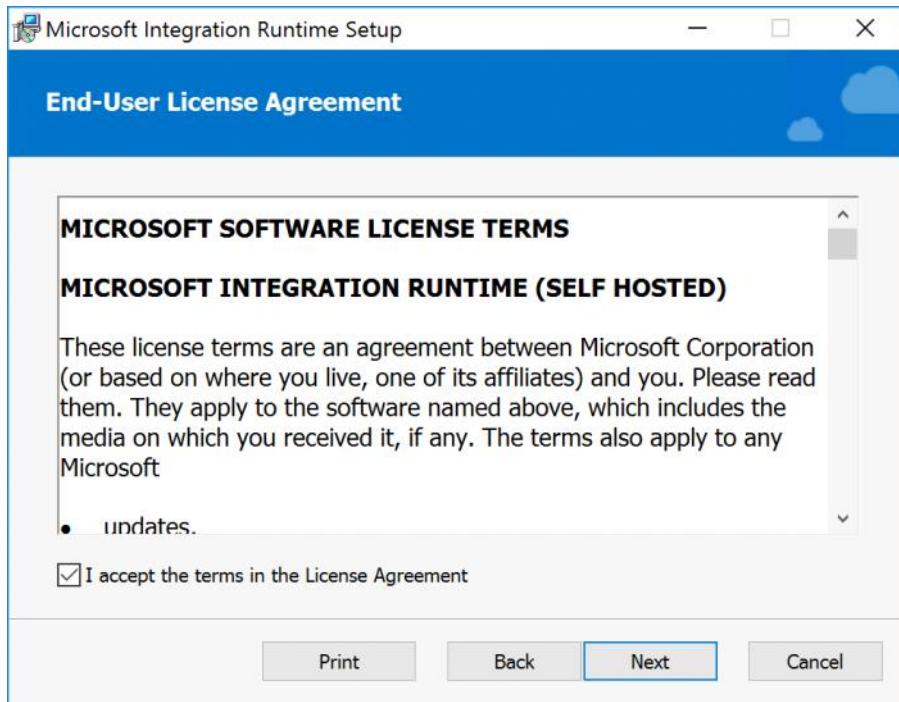
To the right of the table is a 'Download Summary' box containing the file 'IntegrationRuntime_3.13.6942.1 (64-bit).msi' with a size of 'KBMBGB'. Below the summary, it says 'Total Size: 242.3 MB'. At the bottom right of the main content area is a 'Next' button. The footer of the page contains links for various Microsoft products and services.

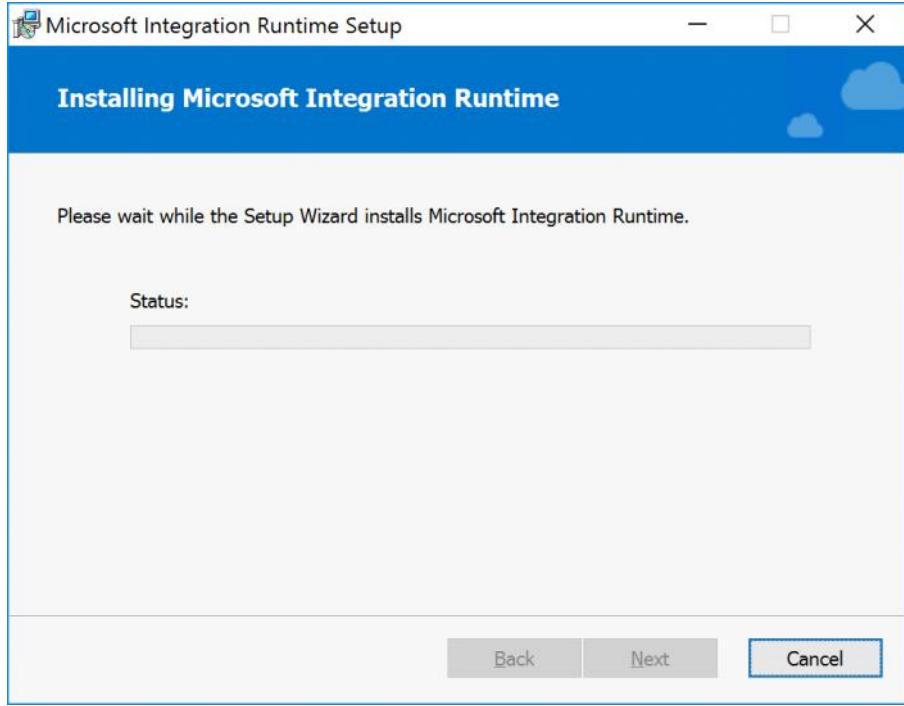
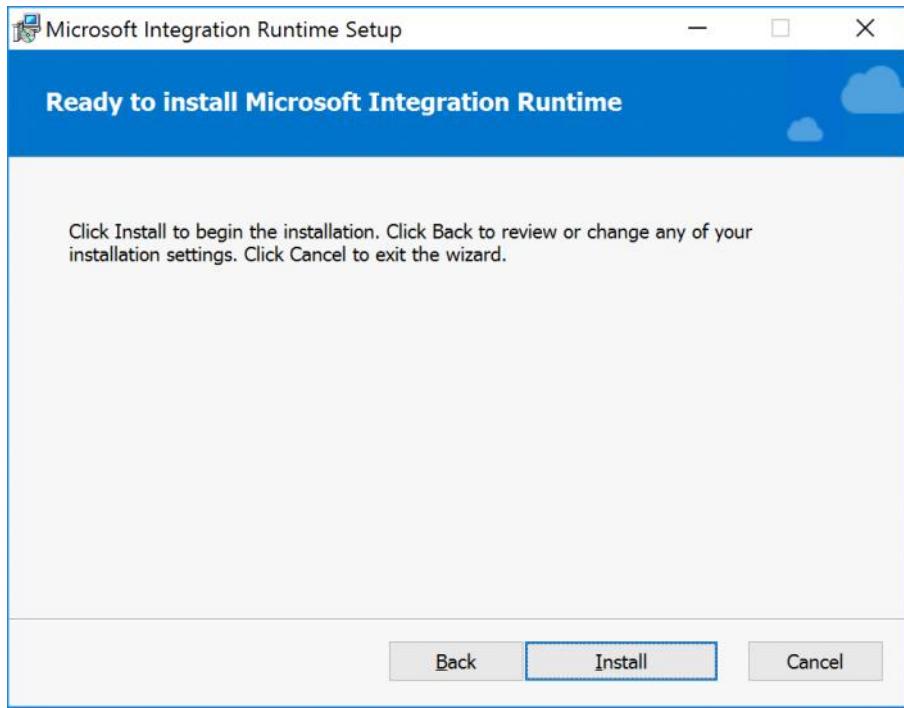
(If you are having problems with the security settings, please refer to the instructions in the prerequisites.)

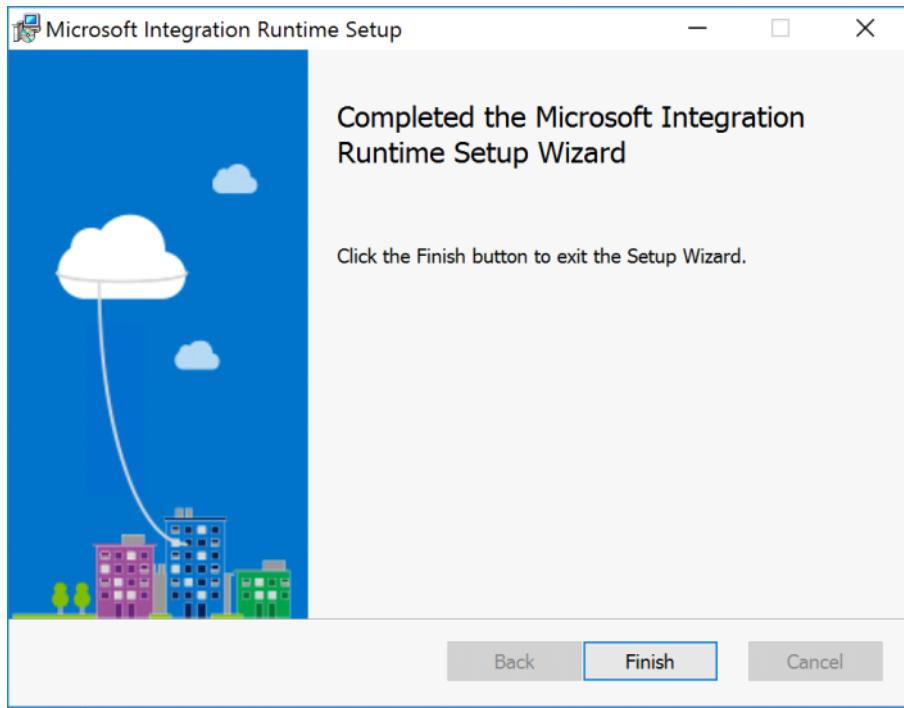
Install the Azure Data Factory Integration Runtime

Run the downloaded Azure Data Factory Integration Runtime Installer.

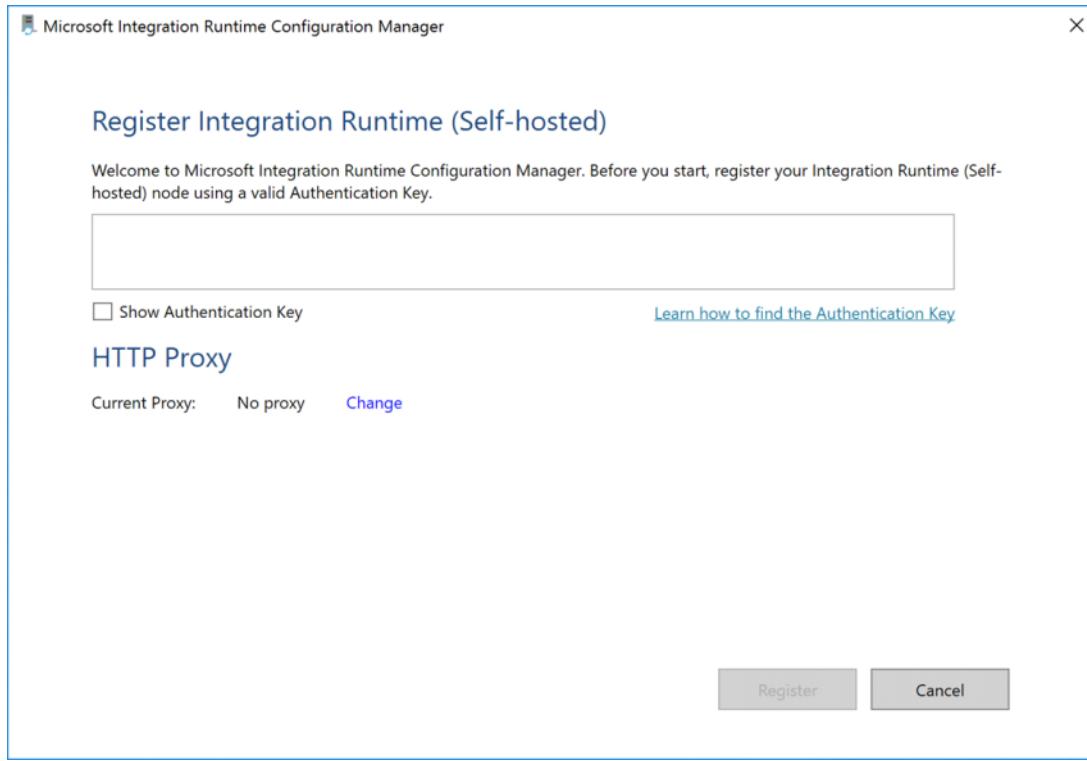








Once the **Microsoft Integration Runtime Configuration Manager** opens, you will need to enter one of the **Authentication Keys** from the Azure Data Factory.



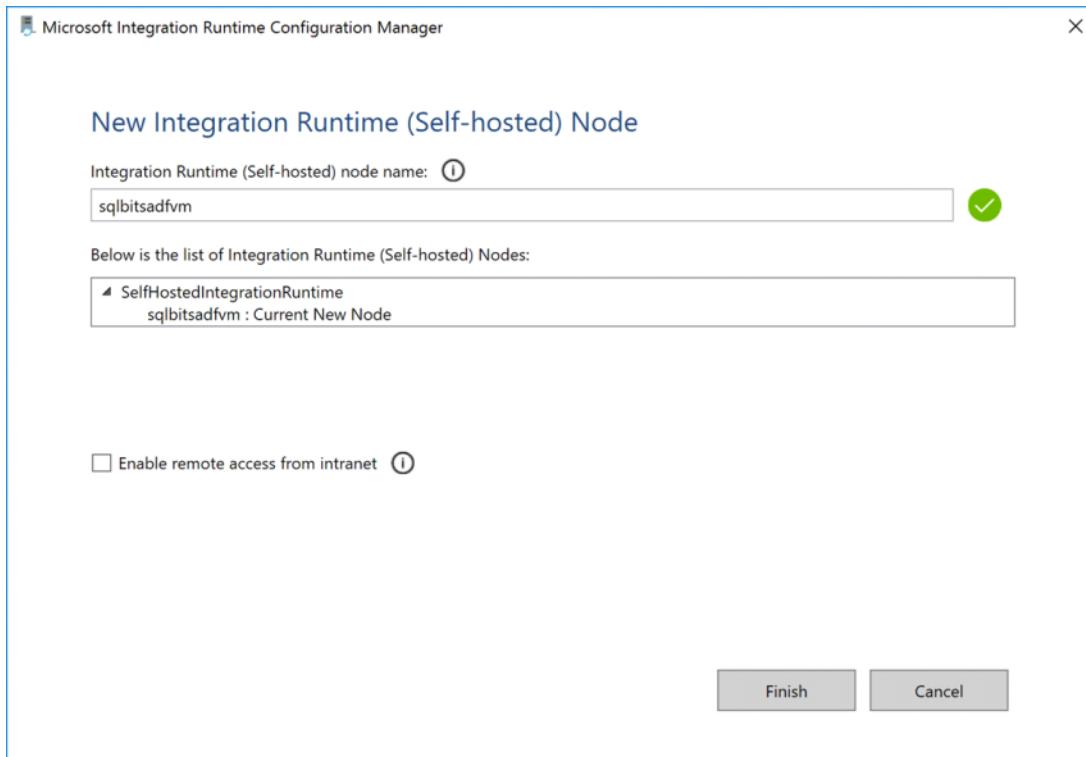
Minimize the Virtual Machine, and go back to the Azure Data Factory window/tab you left open. Copy one of the **Authentication Keys**.

The screenshot shows the Microsoft Azure Data Factory interface. On the left, there's a sidebar with 'Factory Resources' (Pipelines: 0, Datasets: 0, Data Flows (Preview): 0). The main area has tabs for 'Connections' (selected) and 'Integration Runtimes'. Under 'Integration Runtimes', there are two entries: 'AutoResolveIntegrationRuntime' and 'SelfHostedIntegrationRuntime'. A 'Settings' tab is selected in the top right, showing the 'Integration Runtime Setup' dialog. The dialog has sections for 'Name' (SelfHostedIntegrationRuntime), 'Option 1: Express setup' (with a link to launch the express setup), 'Option 2: Manual setup' (with steps to download and install the integration runtime), and a table for 'Name' and 'Authentication Key'. Two rows are listed: 'Key1' with the value 'IR@acb38cb9-e178-4a32-8845-4ac504c763c9@sqlbitsadfcw01adf@uks@oF' and 'Key2' with the value 'IR@acb38cb9-e178-4a32-8845-4ac504c763c9@sqlbitsadfcw01adf@uks@QI'. At the bottom right of the dialog is a 'Finish' button.

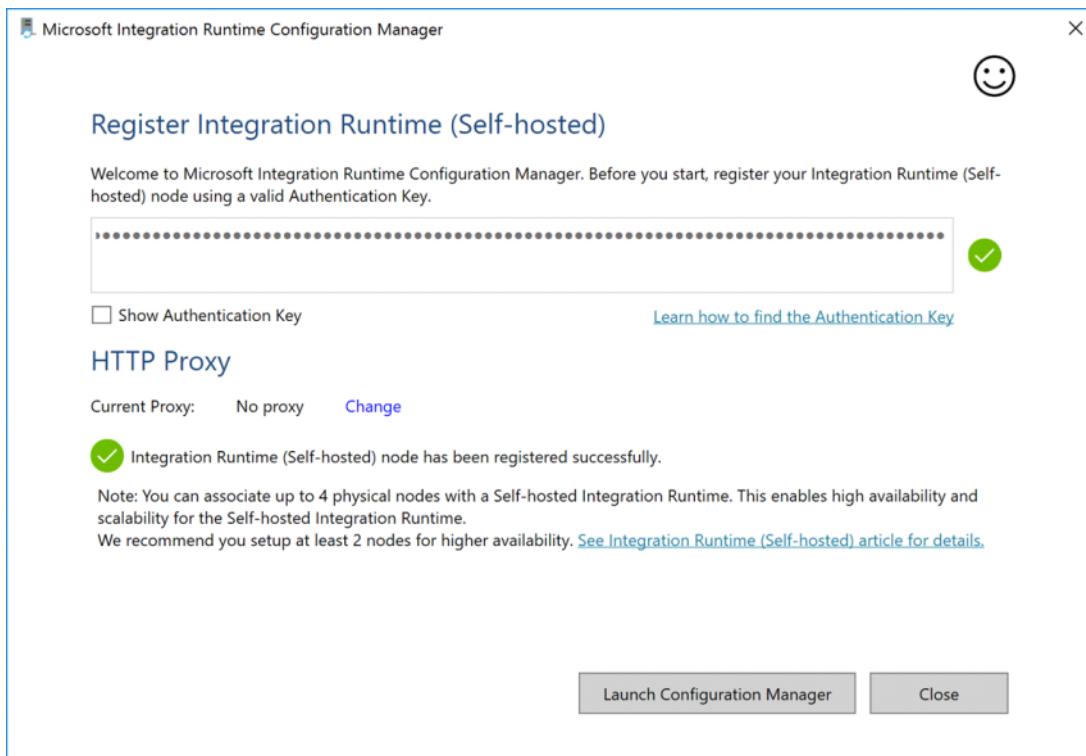
Return to your Virtual Machine and paste the **Authentication Key**. You should see a green checkmark. Click **Register**.

The screenshot shows the 'Microsoft Integration Runtime Configuration Manager' dialog. It has a title bar 'Microsoft Integration Runtime Configuration Manager' and a close button 'X'. The main section is titled 'Register Integration Runtime (Self-hosted)'. It says 'Welcome to Microsoft Integration Runtime Configuration Manager. Before you start, register your Integration Runtime (Self-hosted) node using a valid Authentication Key.' Below this is a large input field containing a long authentication key, followed by a green checkmark icon. There is also a link 'Learn how to find the Authentication Key'. Underneath is an 'HTTP Proxy' section with 'Current Proxy: No proxy' and a 'Change' link. At the bottom are 'Register' and 'Cancel' buttons.

Next, you will set up a Self-Hosted Node. Click **Finish**.



Once the Integration Runtime has been registered successfully, you can click **Close**.



If you click **Launch Configuration Manager** instead, you will see the status of the Self-Hosted Node. It should be connected to the Data Factory V2 cloud service.

Microsoft Integration Runtime Configuration Manager

Home Settings Diagnostics Update Help

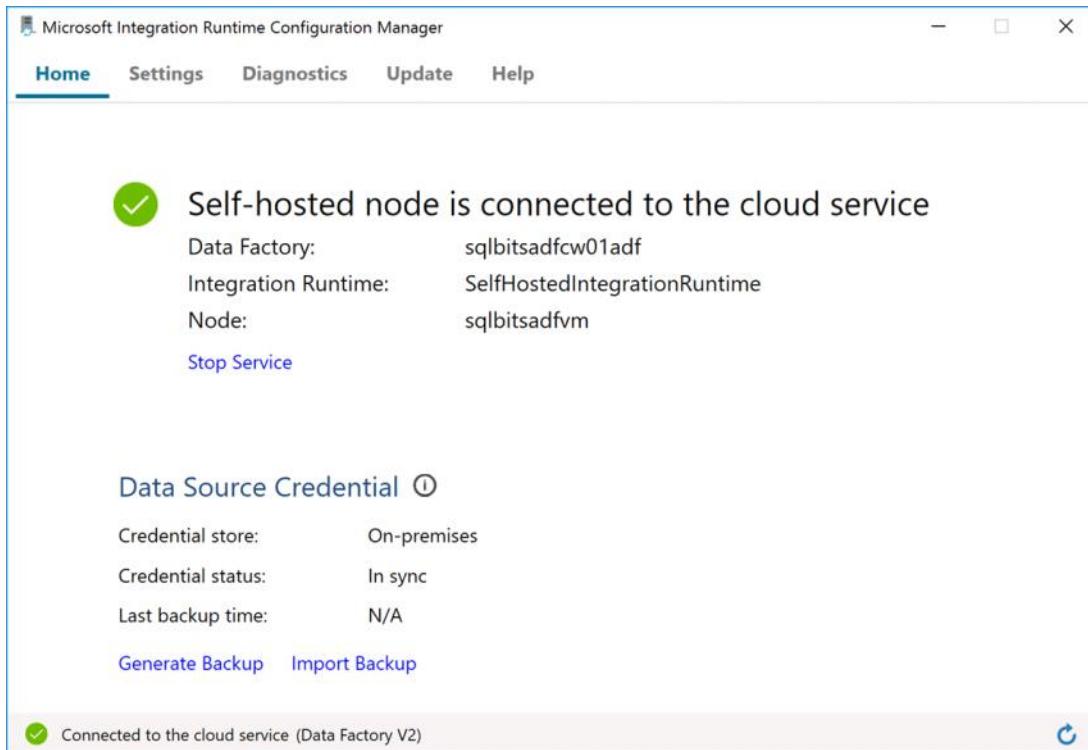
✓ Self-hosted node is connected to the cloud service

Data Factory: sqlbitsadfcw01adf
Integration Runtime: SelfHostedIntegrationRuntime
Node: sqlbitsadfvvm
[Stop Service](#)

Data Source Credential ⓘ

Credential store: On-premises
Credential status: In sync
Last backup time: N/A
[Generate Backup](#) [Import Backup](#)

✓ Connected to the cloud service (Data Factory V2)



(You can always return to this Configuration Manager from the start menu in your Virtual Machine.)

Minimize the Virtual Machine, and go back to the Azure Data Factory window/tab you left open. If the Integration Runtime Menu is still open, click Finish. You will now see that your new **SelfHostedIntegrationRuntime** is **Running**.

Microsoft Azure | Data Factory > sqlbitsadfcw01adf

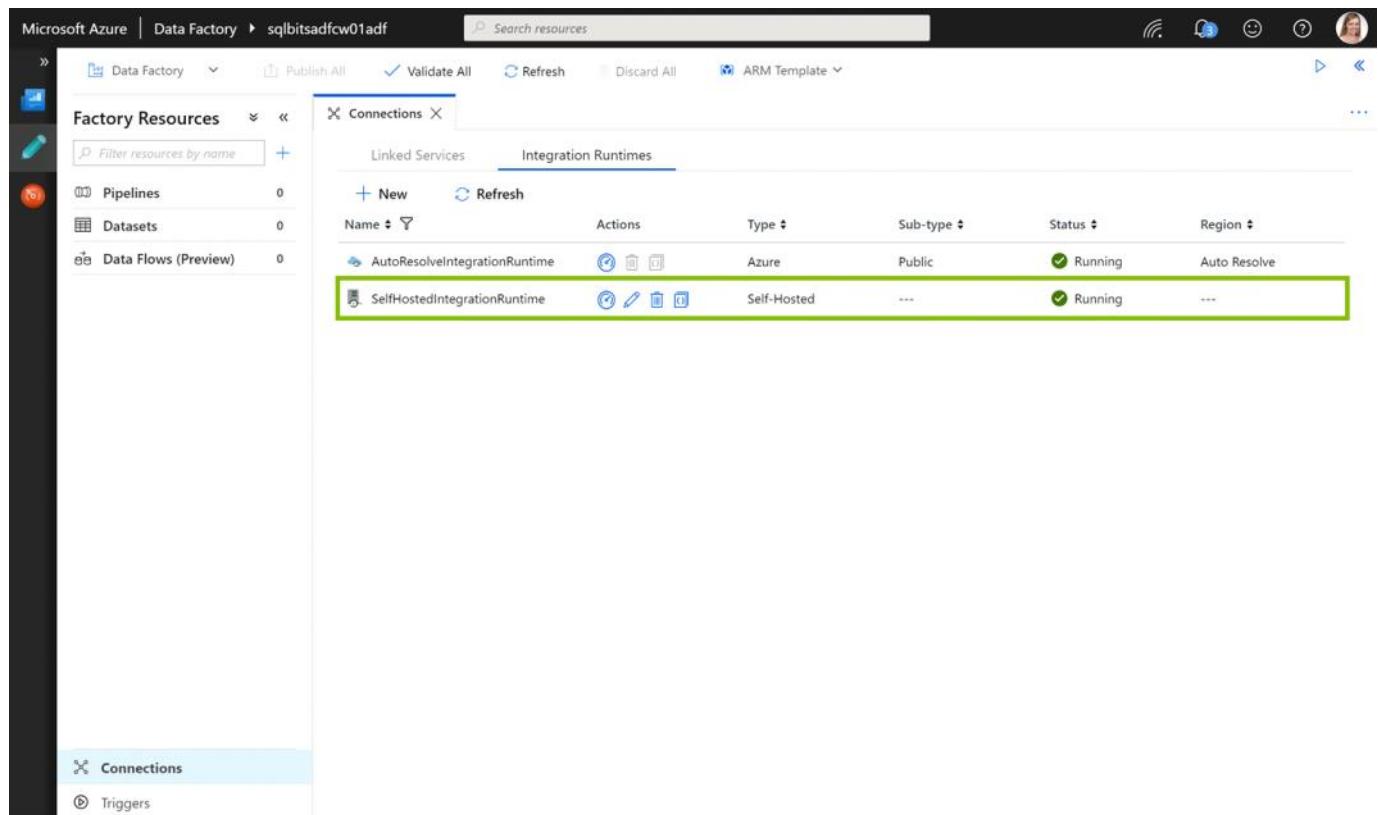
Data Factory Publish All Validate All Refresh Discard All ARM Template

Factory Resources Pipelines 0 Datasets 0 Data Flows (Preview) 0

Connections Filter resources by name +

Name	Actions	Type	Sub-type	Status	Region
AutoResolveIntegrationRuntime		Azure	Public	Running	Auto Resolve
SelfHostedIntegrationRuntime		Self-Hosted	---	Running	---

Connections Triggers



Activity Summary

In this activity, you have created your Self-Hosted Integration Runtime connected to your Virtual Machine, which will be your source in the rest of the lab. Next, you will create the sink (destination) database.

Activity 03: Setup Staging Database

Activity Overview

Estimated time to complete activity: **5-10 minutes.**

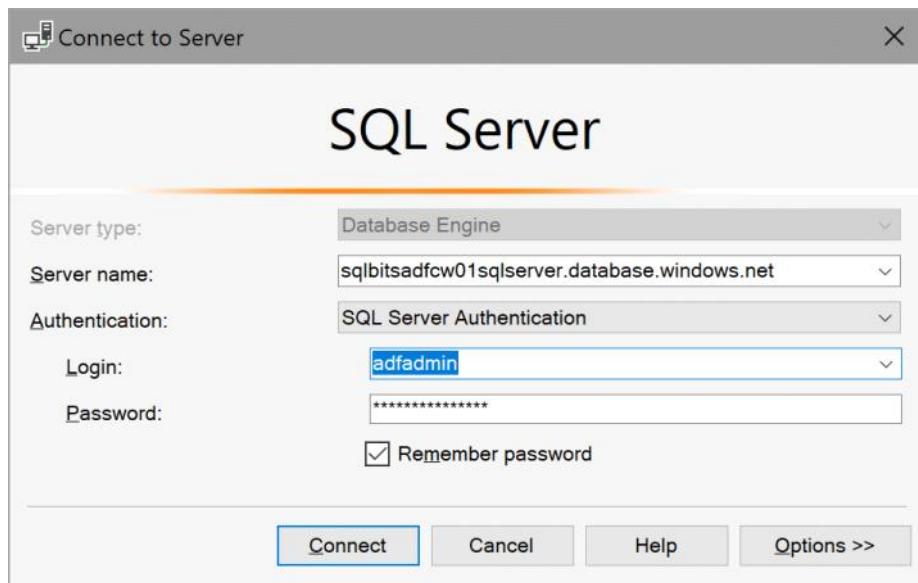
In order to save time during these labs, we have created a DACPAC of the Staging database that includes necessary tables, stored procedures, table types, and more.

If you have already created a blank Staging database - *as instructed in an earlier version of the prerequisites* - we do apologize for this extra step. It will save some time in the end :)

Delete Empty Staging Database

Note: If you do not already have an empty Staging database, skip this step.

Open SQL Server Management Studio (SSMS) and connect to your SQL Server as the **adfadmin** user.



Right-click on the Staging database and choose **Delete**. Wait for the operation to complete, then **shutdown SQL Server Management Studio**.

(Note: We have sometimes experienced a bug where you are unable to deploy a new database with the same name as a database that was just deleted. To avoid running into this issue, make sure you completely close SSMS once the database has been deleted.)

Download Staging.dacpac and StagingSecurity.sql script

Download the latest versions of the following files from:

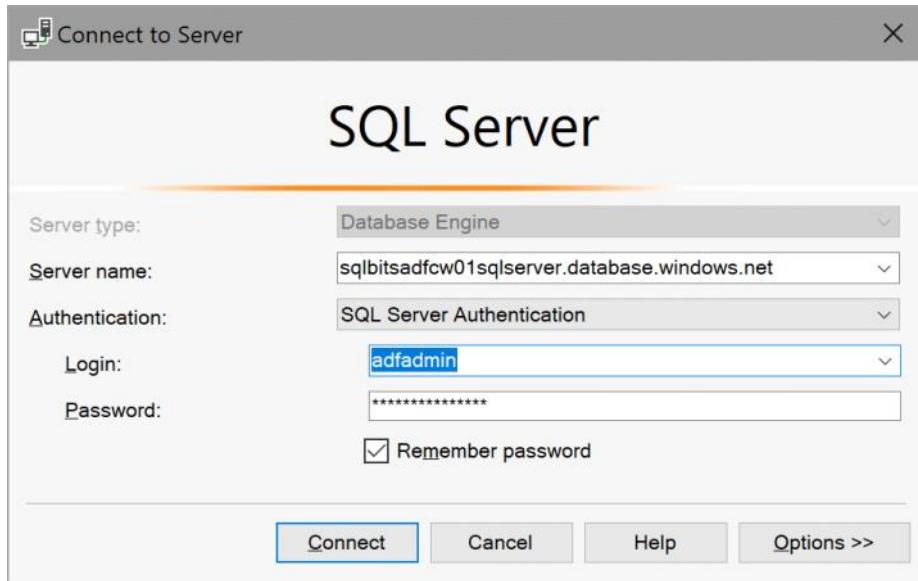
<https://github.com/jasonhorner/adfdeepdive/tree/master/Labs/01%20-%20ADF%20Overview>.

- **Staging.dacpac**
- **Azure SQL Server - Master DB Security.sql**

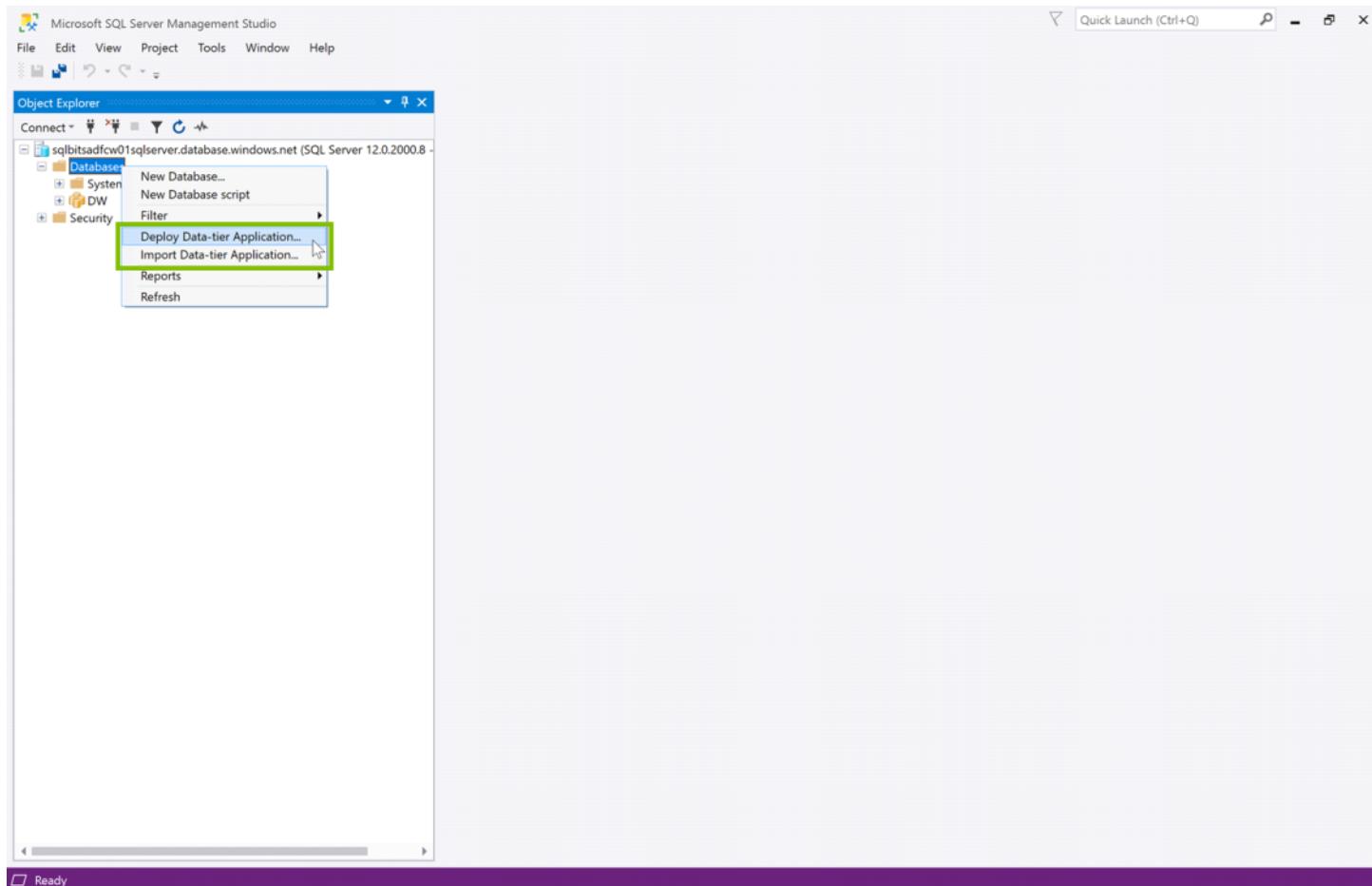
- Azure SQL Server - Staging DB Security.sql

Deploy DACPAC as adfadmin

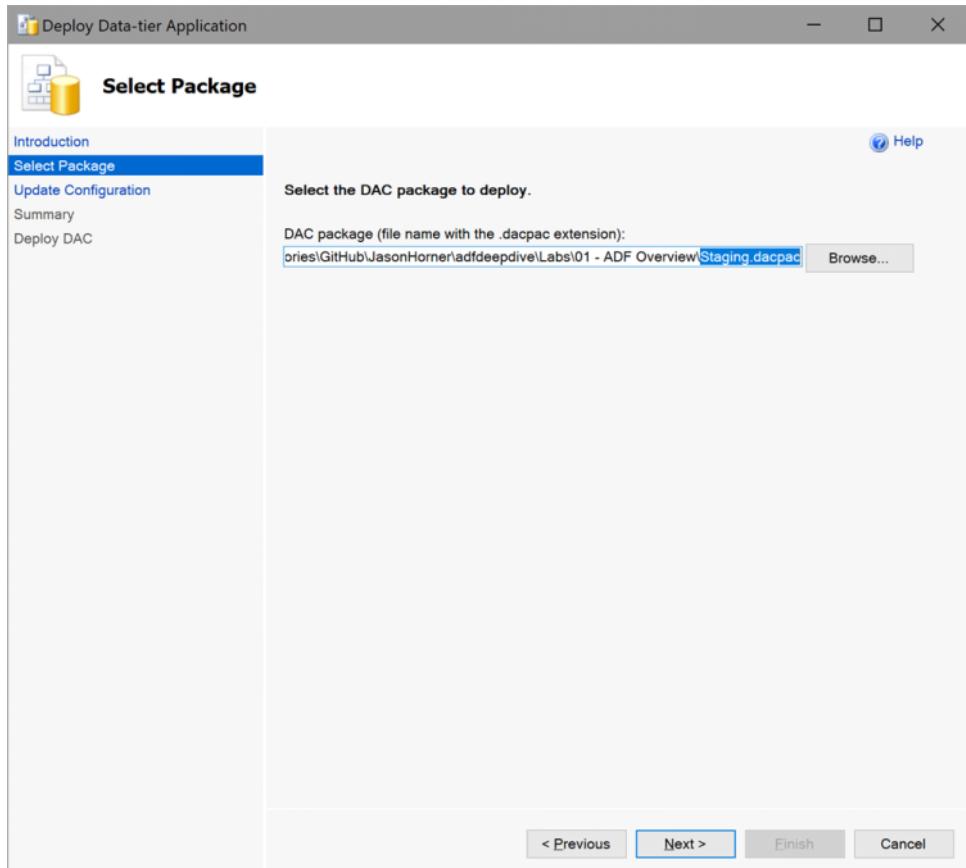
(Re-)Open SQL Server Management Studio (SSMS) and connect to your SQL Server as the **adfadmin** user.
(*This is the SQL Server admin user created in Prerequisites Step 5.*)



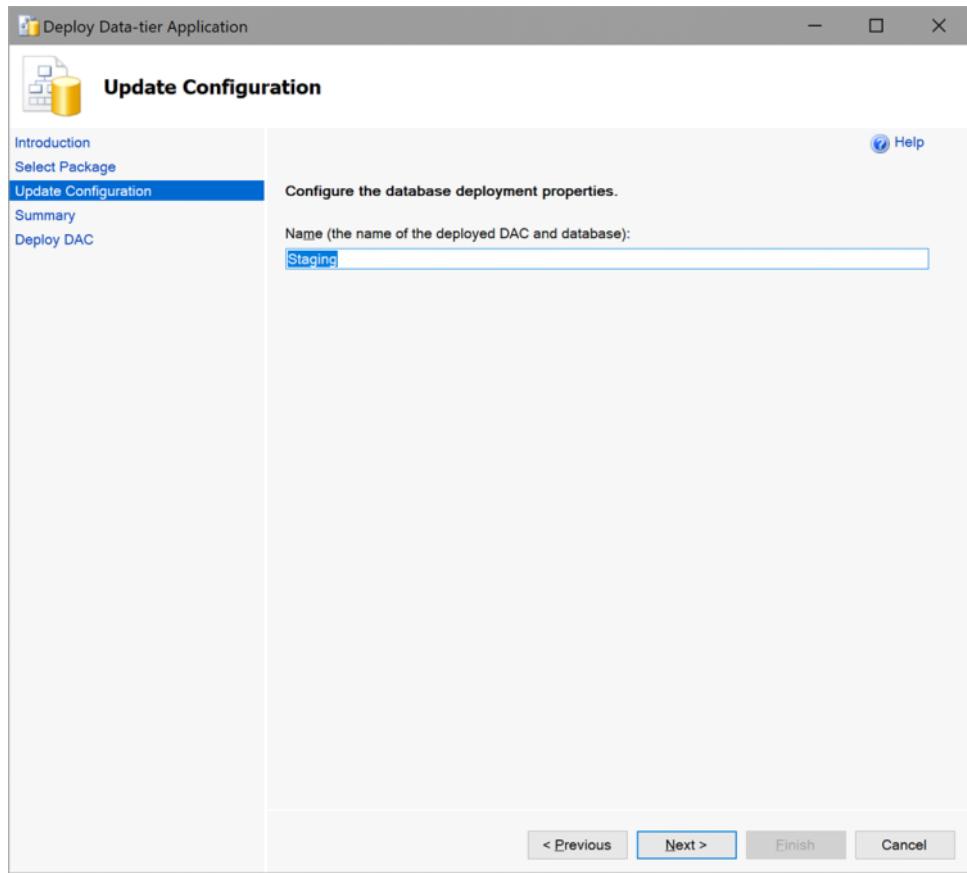
Right-click on **Databases** and select **Deploy Data-tier Application**.



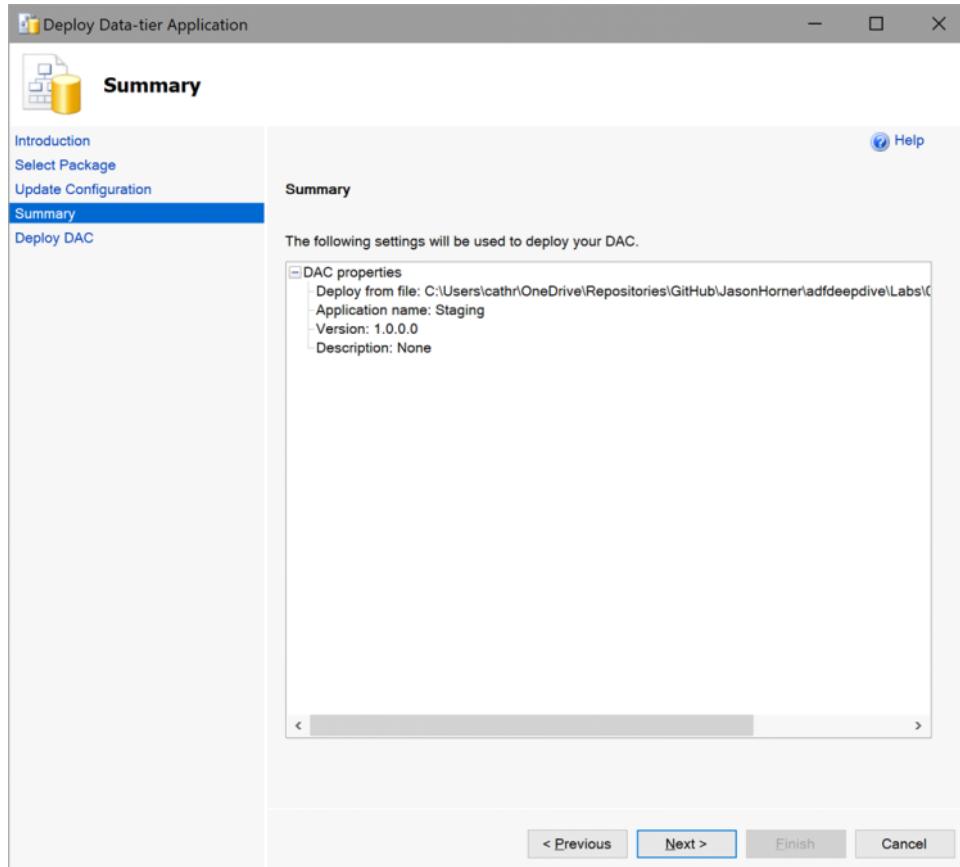
Select the downloaded **Staging.dacpac** file.



Keep the **Staging** name.

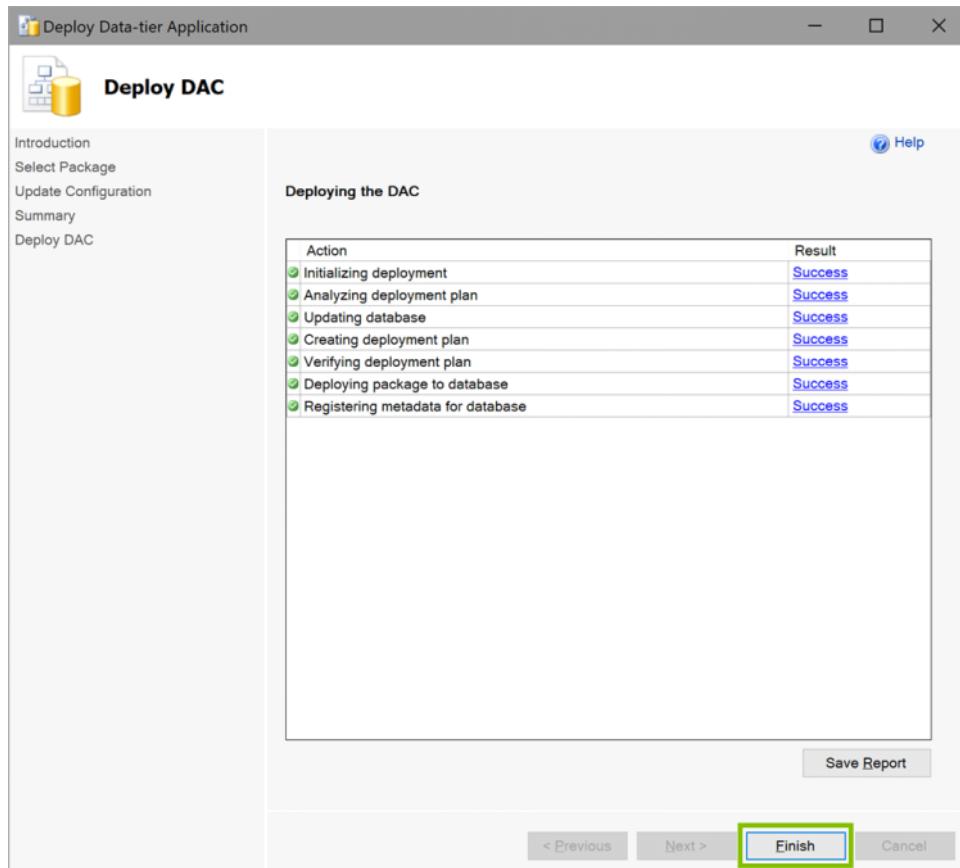


Verify the settings.



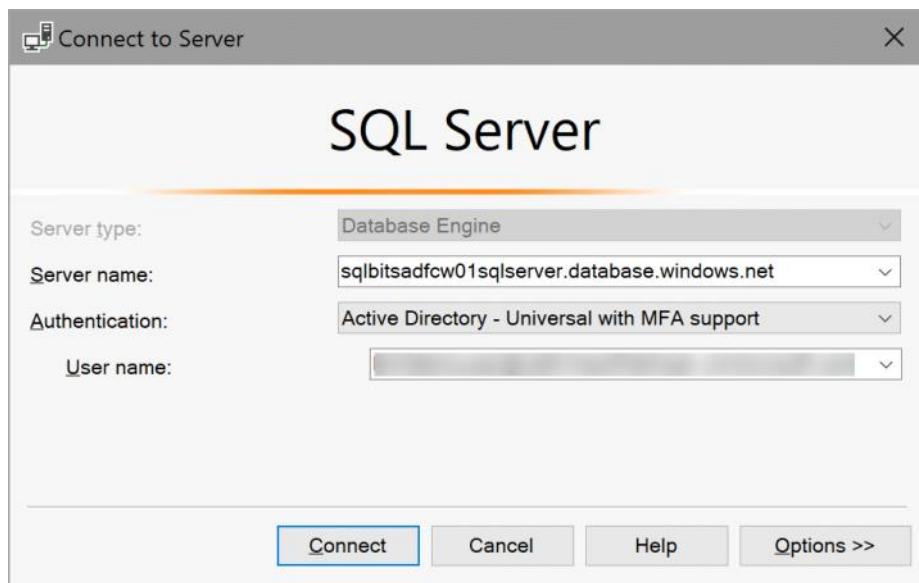
The deployment will usually take 3-5 minutes. Once the database has been deployed successfully, click

Finish.



Setup Master and Staging Security

Open a new connection to your SQL Server as your **Active Directory** user. (*This is the user you specified as the Active Directory Admin in Prerequisites Step 5.*)



Connect to **Master** and open **Azure SQL Server - Master DB Security.sql**. Execute the script.

Connect to **Staging** and open **Azure SQL Server - Staging DB Security.sql**. Ensure that you are connected

to the Staging database and execute the script.

The screenshot shows the Microsoft SQL Server Management Studio interface. The title bar reads "StagingSecurity.sql - Microsoft SQL Server Management Studio". The toolbar includes "File", "Edit", "View", "Query", "Project", "Tools", "Window", and "Help". A dropdown menu is open, showing "Staging" highlighted with a green box. The "Execute" button is also highlighted with a green box. The "Object Explorer" pane on the left shows a connection to "sqlbitsadfcw01sqlserver.database.windows.net (SQL Server 12.0.2000.8)". Under "Databases", there are "System Databases" (DW, Staging), and "Security". The main pane displays a T-SQL script named "StagingSecurity.sql" with the following content:

```
1 CREATE USER [ADF_Access] FROM EXTERNAL PROVIDER;
2
3 CREATE USER [AnalysisServiceUser] FOR LOGIN [AnalysisServiceUser];
4 CREATE USER [DataLoadUser] FOR LOGIN [DataLoadUser];
5 CREATE USER [DeploymentUser] FOR LOGIN [DeploymentUser];
6
7 GRANT CONTROL ON DATABASE::[Staging] TO [DeploymentUser];
8
9 EXEC sp_addrolemember 'Reader', 'AnalysisServiceUser';
10 EXEC sp_addrolemember 'Writer', 'DataLoadUser';
11 EXEC sp_addrolemember 'Writer', 'ADF_Access';
12
13 GRANT EXECUTE TO [Executor];
14 GRANT SELECT TO [Reader];
15
16 GRANT INSERT TO [Writer];
17 GRANT UPDATE TO [Writer];
18 GRANT DELETE TO [Writer];
19
20 GRANT CREATE TABLE TO [Writer];
21 GRANT ALTER ANY SCHEMA TO [Writer];
22 GRANT ALTER ANY DATASPACE TO [Writer];
```

The status bar at the bottom indicates "Connected. (1/1)", "Ln 11", "Col 46", "Ch 46", "INS", and "Staging 00:00:00 0 rows".

Activity Summary

In this activity, you have deployed the Staging database, which will be your sink (destination) in the rest of the lab. Next, you will create Linked Services to connect to the SQL Server in your Virtual Machine, as well as to your Staging Azure SQL Database and Azure Data Lake Storage Gen1.

Activity 04: Create Linked Services

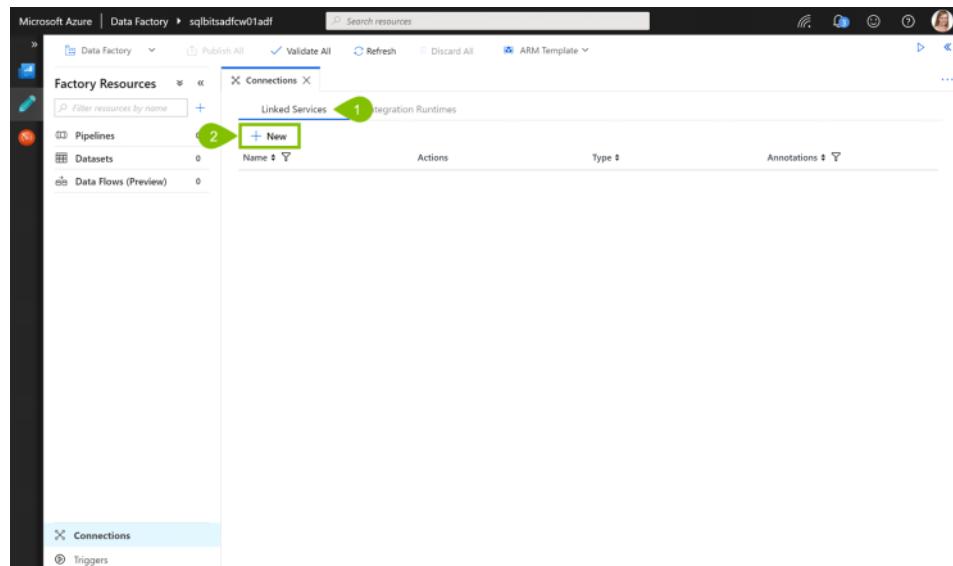
Activity Overview

Estimated time to complete activity: **5 minutes**.

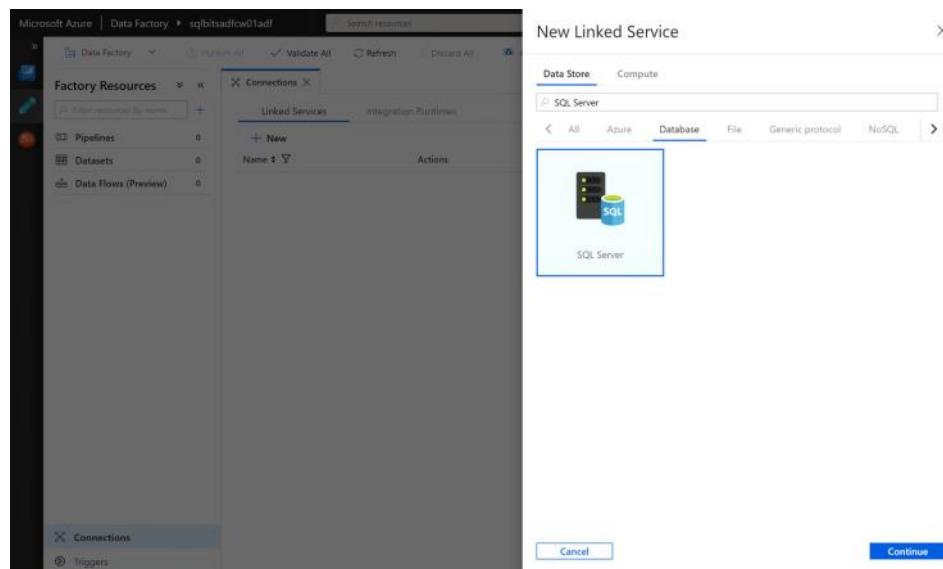
In this activity, you will create Linked Services to connect to the SQL Server in your Virtual Machine, as well as to your Azure SQL Database and Azure Data Lake Storage Gen1.

Create a Linked Service to the SQL Server in the Virtual Machine

In the **Connections** tab, click on **Linked Services**, and click **New**.



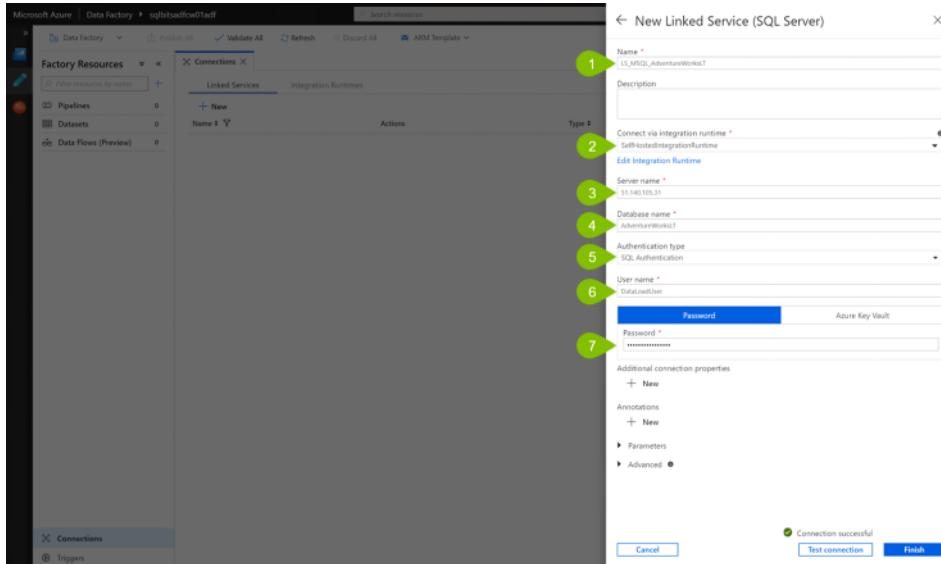
The New Linked Service menu opens. Navigate to Database -> SQL Server, or type in **SQL Server** in the search. Click **Continue**.



In the **New Linked Service (SQL Server)** pane, provide the following information:

Step	Setting	Value	Notes
1	Name	LS_MSQ_AdventureWorksLT	Use the prefix LS_MSQ (for <i>Linked Service, Microsoft SQL Server</i>) and use AdventureworksLT as the name.
2	Connect via Integration Runtime	SelfHostedIntegrationRuntime	Select the Self-Hosted Integration Runtime you created in the previous activity.
3	Server Name	<IP Address>	The Server Name is the IP address of the Virtual Machine you created in the prerequisites. You can find this IP address from the Azure Portal, or from the Integration Runtime settings. (<i>See instructions below.</i>)
4	Database Name	AdventureWorksLT	Use the AdventureWorksLT database you created in the prerequisites.
5	Authentication Type	SQL Authentication	Use SQL Authentication .

6	User Name	DataLoadUser	Use the user you created in the prerequisites, for example DataLoadUser.
7	Password	DataFactoryD3m0!	Use the password you created in the prerequisites, for example DataFactoryD3m0!



Click **Finish**.

Finding the IP Address of the Virtual Machine

Option 1: In the Azure Portal, navigate to the Virtual Machine and copy the **Public IP Address**.

The screenshot shows the Azure portal interface with the URL [https://portal.azure.com/#blade/HubsBlade](#). The left sidebar shows 'Virtual machines' under the 'Compute' category. The main pane displays the details for a VM named 'sqlbitsadfv01'. The 'Public IP address' field is highlighted with a green box, containing the value '51.140.105.31'. Other visible details include the resource group 'sqlbitsadfcw01', status 'Running', location 'UK South', subscription 'Visual Studio Premium with MSDN', and size 'Standard D2s v3 (2 vcpus, 8 GB memory)'. Below the VM details, there are performance monitoring charts for CPU, Network, Disk bytes, and Disk operations/sec.

Option 2: In Azure Data Factory, go to the Integration Runtime **Nodes** settings and click **Get IP Address**.

Microsoft Azure | Data Factory > squbitsadfcw01adf

Factory Resources

Connections

Search resources

Filter resources by name

Pipelines 0

Datasets 0

Data Flows (Preview) 0

Connections

Integration Runtimes

+ New Refresh

Name	Actions	Type	Sub-type	Status	Region
AutoResolveIntegrationRuntime	View Edit Delete	Azure	Public	Running	Auto Resolve
SelfHostedIntegrationRuntime	View Edit Delete	Self-Hosted	...	Running	...

Microsoft Azure | Data Factory > squbitsadfcw01adf

Factory Resources

Connections

Search resources

Filter resources by name

Pipelines 0

Datasets 0

Data Flows (Preview) 0

Connections

Integration Runtimes

+ New Refresh

Name	Status	IP Address	Limit Concurrent Jobs	Actions
AutoResolveIntegrationRuntime	Running	...	6	View Edit Delete
SelfHostedIntegrationRuntime	Running	...	6	View Edit Delete

Integration Runtime: SelfHostedIntegrationRuntime

Setting Nodes Auto update Sharing

View Service URL

1

2

Cancel Finish

Create a Linked Service to the Azure SQL Database

To be able to connect to your Azure SQL Server from your local computer and Azure Data Factory, you first need to add a firewall rule to allow access from the SQLBits WiFi, then configure SQL Server security. In this step, we will also create the objects needed for the next activity.

Add a Firewall Rule for the SQLBits WiFi

The screenshot shows the Azure portal's 'SQL servers' blade. On the left, there's a navigation menu with icons for Home, App Service, Functions, Logic Apps, Container Registry, and more. The main area displays the 'Overview' page for the 'sqlbitsadfcw01sqlserver' database. Key details shown include:

- Resource group: sqlbitsadfcw01
- Status: Available
- Location: UK South
- Subscription: Visual Studio Premium with MSDN
- Server admin: adfadmin
- F火walls and virtual networks: Show firewall settings (highlighted with a green box)

The 'Features' tab is selected, showing two items:

- Active Directory admin: NOT CONFIGURED
- Advanced Data Security: NOT CONFIGURED

This screenshot shows the 'Firewalls and virtual networks' configuration page for the same SQL server. It includes:

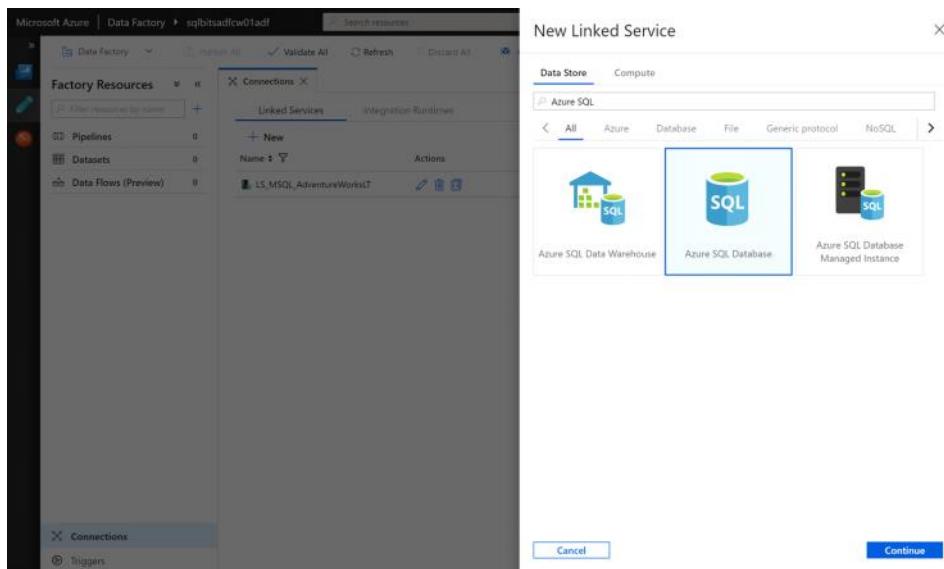
- A message: "Connections from the IPs specified below provides access to all the databases in sqlbitsadfcw01sqlserver."
- An 'Allow access to Azure services' toggle switch set to ON.
- A 'Client IP address' field containing 84.210.96.115.
- A table for defining firewall rules:

Rule Name	Start IP	End IP
SQLBitSF	84.210.96.115	84.210.96.115
- A message: "Connections from the VNET/Subnet specified below provides access to all databases in sqlbitsadfcw01sqlserver."
- A 'Virtual networks' section with buttons for 'Add existing virtual network' and 'Create new virtual network'.

This screenshot shows the 'Firewalls and virtual networks' configuration page again. The 'Save' button is highlighted with a green box. The 'SQLBitSF' rule entry in the table is also highlighted with a green box.

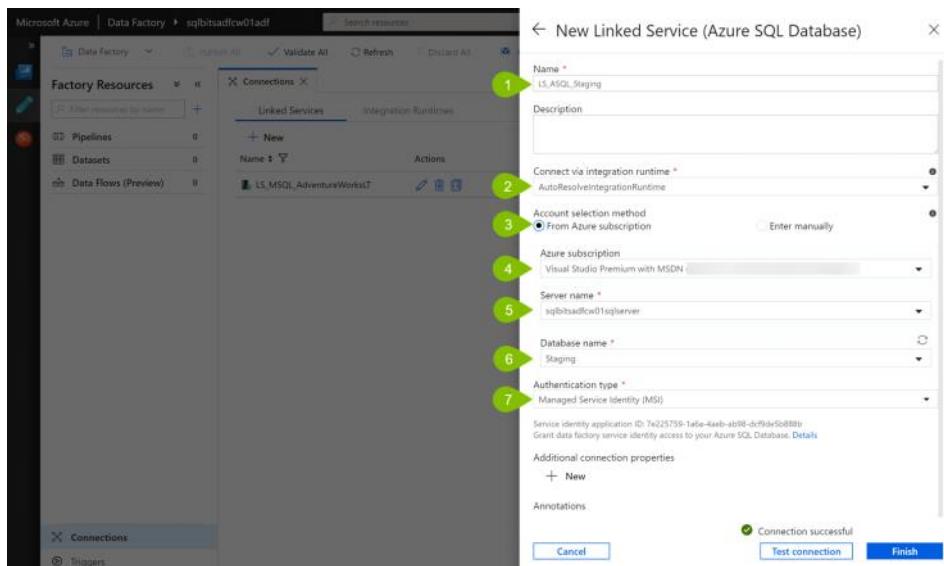
Create the Linked Service

Create a new Linked Service. Navigate to Azure -> Azure SQL Database, or type in **Azure SQL** in the search. Click **Continue**.



In the New Linked Service (SQL Server) pane, provide the following information:

Step	Setting	Value	Notes
1	Name	LS_ASQL_Staging	Use the prefix LS_ASQL (for Linked Service, Azure SQL Database) and use Staging as the name.
2	Connect via Integration Runtime	AutoResolveIntegrationRuntime	Keep the default AutoResolveIntegrationRuntime .
3	Account Selection Method	From Azure Subscription	Keep the default From Azure Subscription .
4	Azure Subscription	<Your Subscription>	Select your Azure Subscription .
5	Server Name	sqlbitsadf<999>sqlserver	Select the SQL Server created in the prerequisites.
6	Database Name	Staging	Select the Staging database created in the prerequisites.
7	Authentication Type	Managed Service Identity (MSI)	Choose Managed Service Identity (MSI) . This connects using the identity of the Azure Data Factory.



Test the connection, then click **Finish**.

Create a Linked Service to the Azure Data Lake Storage Gen1

To be able to connect to your Azure Data Lake Storage Gen1 from Azure Data Factory, you first need to give the ADF_Access group (that you created in Activity 01) access.

Give the ADF_Access Group Permissions

Go to the Azure Data Lake Storage Gen1 and click **Data Explorer**.

Home > Data Lake Storage Gen1 > sqlbitsadfcw01dlsv1

Data Lake Storage Gen1

sqlbitsadfcw01dlsv1 Data Lake Storage Gen1

Essentials

Estimated cost to date: **USD 0.00**

Total storage utilization: **0 bytes** (Last updated 2/17/2019, 9:00 AM)

Read/write requests: **100** (Last updated 2/17/2019, 9:00 AM)

Ingress/Egress: **100B** (Last updated 2/17/2019, 9:00 AM)

NAME

sqlbitsadfcw01dlsv1

Activity log

Access control (IAM)

Tags

Diagnose and solve problems

Events

Settings

Encryption

Firewall and virtual networks

Pricing tier

Properties

Locks

Automation script

Data Lake Storage Gen1

Quick start

Data explorer

Monitoring

Alerts

Metrics

Plan maintenance activities

Click **Access**, then **Add**.

Home > Data Lake Storage Gen1 > sqlbitsadfcw01dlsv1 > Data explorer > sqlbitsadfcw01dlsv1 > Access

sqlbitsadfcw01dlsv1

Access

+ Add

Your permissions

effective permissions on this folder are: Read,Write,Execute.

Owners

Read Write Execute

Assigned permissions

Everyone else

Users not covered above will be limited by these permissions

Click **Select user or group**, then select the **ADF_Access** group.

Home > Data Lake Storage Gen1 > sqlbitsadfcw01dlsv1 > Data explorer > sqlbitsadfcw01dlsv1 > Access > Assign permissions > Select user or group

Access

+ Add

Your permissions

effective permissions on this folder are: Read,Write,Execute.

Owners

Read Write Execute

Assigned permissions

No entries.

Everyone else

Users not covered above will be limited by these permissions

Select user or group

ADF_Access

Add

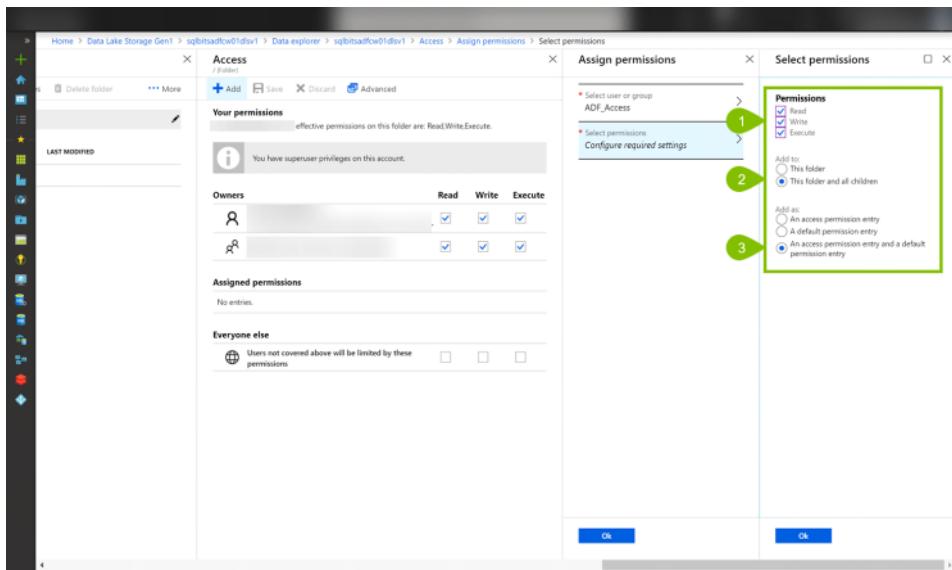
Configure required settings

OK

Change the default settings to:

- Allow **Read**, **Write**, and **Execute**
- Add to: **This folder and all children**

- Add as: An access permission entry and a default permission entry



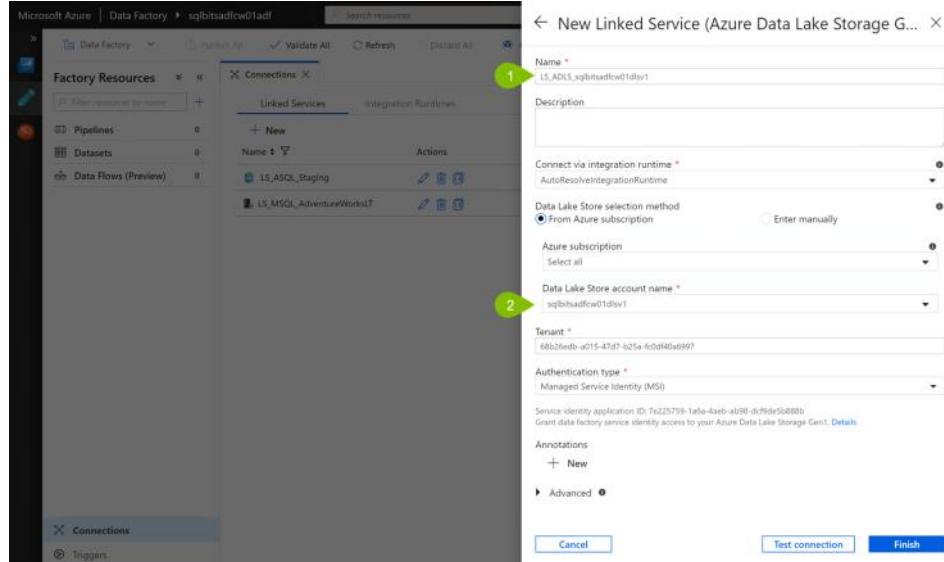
Click **OK** and close out of the other panes.

Create the Linked Service

Create a new Linked Service. Navigate to Azure -> Azure Data Lake Storage Gen1, or type in **Data Lake** in the search. Click **Continue**.

Step	Setting	Value	Notes
1	Name	LS_ADLS_sqlbitsadf<999>dlsv1	Use the prefix LS_ADLS (for Linked Service, Azure Data Lake Storage) and use the name of the Azure Data Lake Storage Gen1 you created in the prerequisites.
2	Data Lake Store Account Name	sqlbitsadf<999>dlsv1	Choose the Azure Data Lake Storage Gen1 you created in the prerequisites.

Unless specified above, use the default settings.



Test the connection, then click **Finish**.

Activity Summary

You have now created your first Linked Services, including configuring connection security. Next, you will create your first Datasets.

Activity 05: Create Datasets

Activity Overview

Estimated time to complete activity: **5 minutes**.

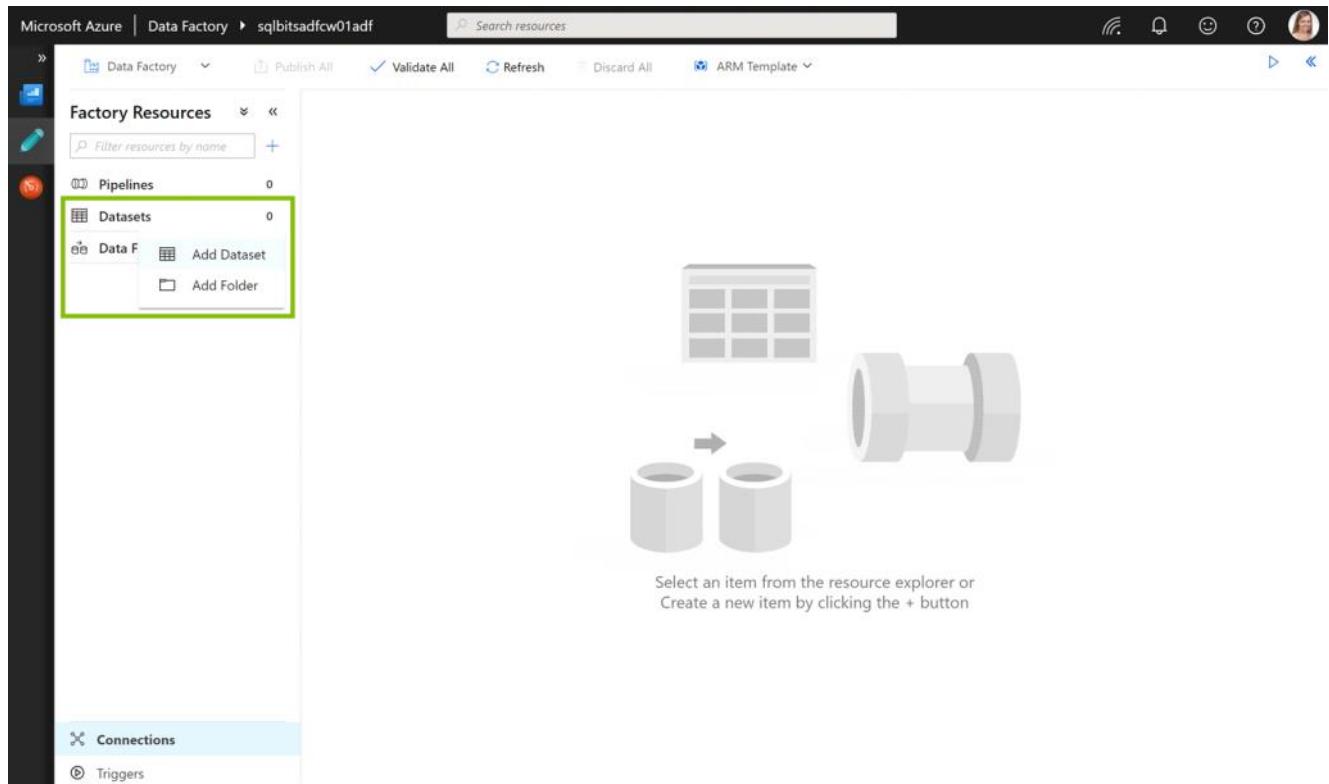
In this activity, you will create the Datasets needed to copy data from the SQL Server in the Virtual Machine to the Azure SQL Database.

Create a SQL Server Dataset

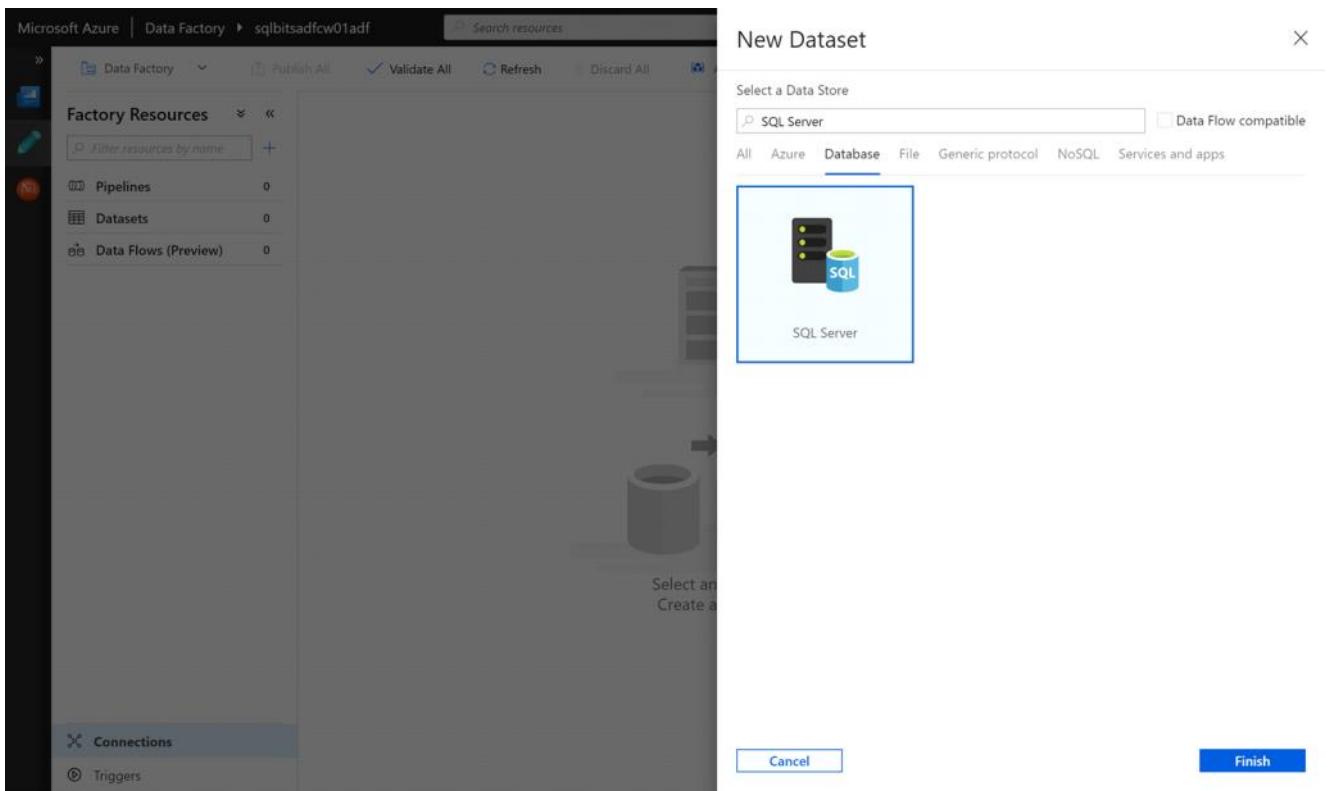
Use the following settings:

Step	Setting	Value	Notes
1	Name	DS_MSQSL_SalesLT_Customer	Use the prefix DS_MSQSL_ (for <i>DataSet</i> , Microsoft SQL Server) and use SalesLT_Customer (the source schema and table) as the name.
2	Linked Service	LS_MSQSL_AdventureWorksLT	Choose the SQL Server Linked Service you created in the previous activity.
3	Table	SalesLT.Customer	Choose the SalesLT.Customer table.

Under Factory Resources, click the ellipsis next to Datasets, then click **Add Dataset**.
(You can also click the + sign and then click **Dataset**.)



The **New Dataset** menu opens. Navigate to Database -> SQL Server, or type in SQL Server in the search. Select **SQL Server** and click **Finish**.



In the **General** tab, name the Dataset **DS_MSQSL_SalesLT_Customer**.

The screenshot shows the dataset configuration page in the Microsoft Azure Data Factory interface. The 'Factory Resources' sidebar on the left shows 'Datasets' (1) selected, with 'DS_MSQSL_SalesLT_Customer' listed. The main area displays the dataset details. The 'General' tab is active, showing the 'Name' field set to 'DS_MSQSL_SalesLT_Customer' (marked with a green arrow). The 'Connection' tab is visible above the form fields. The 'Description' and 'Annotations' fields are empty. The 'Code' button is located at the top right of the page.

In the **Connection** tab, select the **LS_MSQSL_AdventureWorksLT** linked service, and the **SalesLT.Customer** table.

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (0), 'Datasets' (1), and 'Data Flows (Preview)' (0). The 'Datasets' item is selected, showing a list item 'DS_MSQL_SalesLT_Customer'. In the main pane, a dataset named 'DS_MSQL_Sale...' is being edited. The 'Connection' tab is active, displaying a configuration box. Inside the box, the 'Linked service' dropdown is set to 'LS_MSQL_AdventureWorksLT' (highlighted with a green circle labeled '2') and the 'Table' dropdown is set to '[SalesLT].[Customer]' (highlighted with a green circle labeled '3'). Below the dropdowns are 'Test connection', 'Edit', '+ New', 'Refresh', and 'Preview data' buttons. At the bottom of the main pane, there are 'Connections' and 'Triggers' links.

Click Publish All to save.

This screenshot is identical to the one above it, showing the Microsoft Azure Data Factory interface. The 'Factory Resources' sidebar and the dataset configuration in the main pane are the same. The difference is that the 'Publish All' button in the top navigation bar is now highlighted with a yellow box, indicating the next step in the process.

Create an Azure SQL Database Dataset

Repeat the previous steps, but this time, create an **Azure SQL Database** dataset. Use the following settings:

Step	Setting	Value	Notes
1	Name	DS_ASQL_SalesLT_Customer	Use the prefix DS_ASQL_ (for <i>DataSet</i> , <i>Azure SQL Database</i>) and use SalesLT_Customer (<i>the sink schema and table</i>) as the name.
2	Linked Service	LS_ASQL_Staging	Choose the Azure SQL Database Linked Service you created in the previous activity.
3	Table	AWLTSRC.SalesLT_Customer	Choose the AWLTSRC.SalesLT_Customer table.

Activity Summary

You have now created your first Datasets. Next, you will create your first Pipeline to copy data between these Datasets.

Activity 06: Create Pipeline

Activity Overview

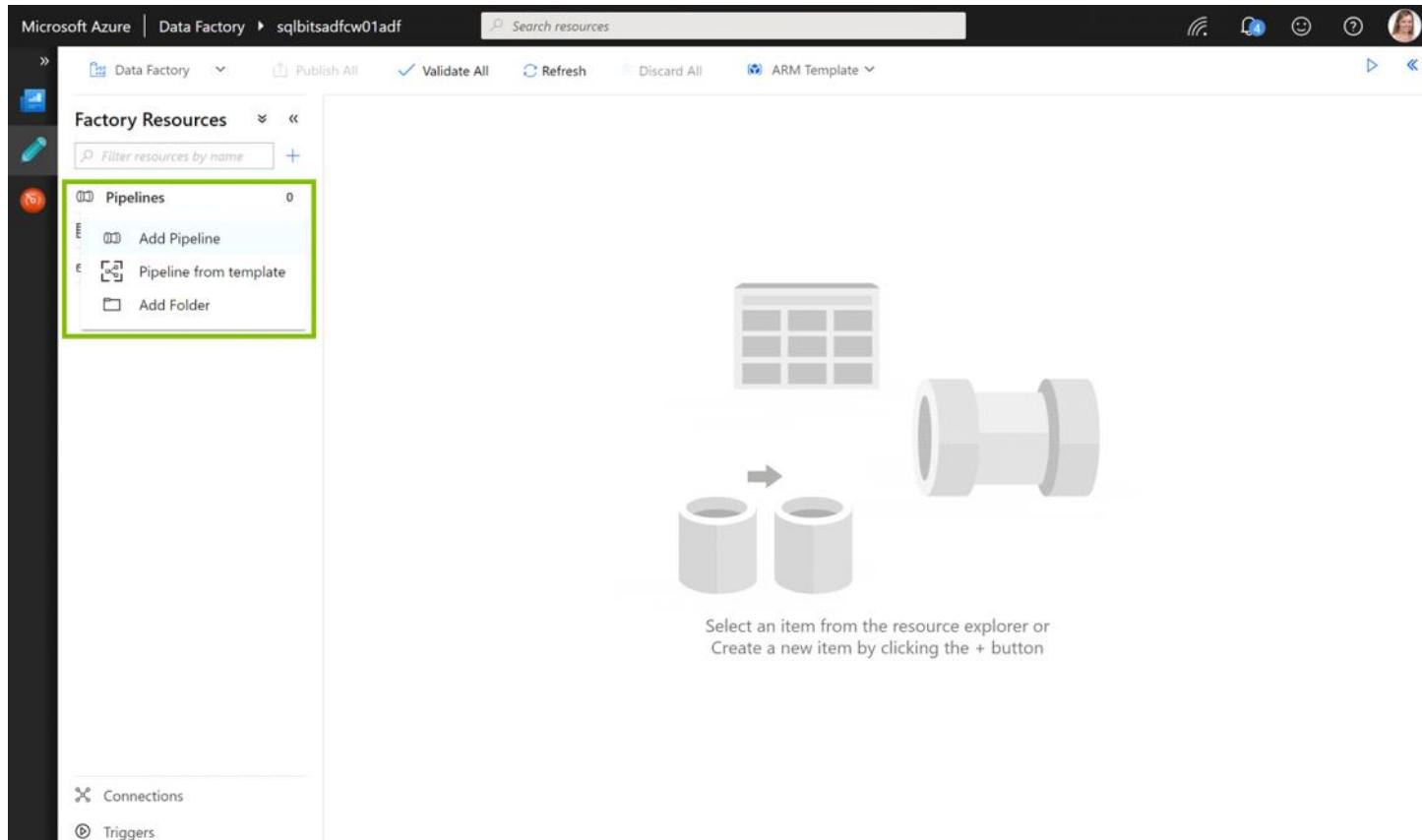
Estimated time to complete activity: **5 minutes.**

In this activity, you will create the Pipeline to copy data between your SQL Server in the Virtual Machine to the Azure SQL Database.

Create the Pipeline

Under Factory Resources, hover over Pipelines until the ellipsis shows up, click it, then click **Add Pipeline**.

(*You can also click the + sign and then click Pipeline.*)



In the **General** tab, name the Pipeline **Copy_SalesLT_Customer**.

The screenshot shows the Microsoft Azure Data Factory pipeline designer interface. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (1), 'Datasets' (2), and 'Data Flows (Preview)' (0). The main area displays the 'Copy_SalesLT...' pipeline. The 'Activities' toolbox on the right is expanded, showing categories like 'Batch Service', 'Databricks', 'Move & Transform', 'Data Lake Analytics', 'General', 'HDInsight', 'Iteration & Conditionals', and 'Machine Learning'. The 'Move & Transform' category is expanded, and the 'Copy Data' activity is selected, highlighted with a green border. The 'General' tab of the properties pane is active, showing the 'Name' field set to 'Copy_SalesLT_Customer'. Other tabs include 'Parameters', 'Variables', and 'Output'.

In the **Activities** toolbox, expand **Move & Transform**. (You can also search for activities by name.) Drag the **Copy Data** activity onto the pipeline designer surface.

The screenshot shows the Microsoft Azure Data Factory pipeline designer interface. The 'Factory Resources' sidebar lists 'Pipelines' (1), 'Datasets' (2), and 'Data Flows (Preview)' (0). The main area displays the 'Copy_SalesLT...' pipeline. The 'Activities' toolbox on the right is expanded, showing categories like 'Batch Service', 'Databricks', 'Move & Transform', 'Data Lake Analytics', 'General', 'HDInsight', 'Iteration & Conditionals', and 'Machine Learning'. The 'Move & Transform' category is expanded, and the 'Copy Data' activity is selected. A green arrow points from the 'Copy Data' activity in the toolbox to its corresponding icon on the pipeline designer surface. The pipeline designer surface shows the 'Copy_SalesLT_Customer' pipeline with the 'Copy Data' activity added.

Click on the Copy Data activity. In the **General** tab, give the activity the name **CopyFromMSQLtoASQL**.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (1), 'Datasets' (2), and 'Data Flows (Preview)' (0). The main area displays a 'Copy Data' activity named 'CopyFromMSQLtoASQL'. The 'General' tab is selected, showing the activity's configuration. The 'Name' field is set to 'CopyFromMSQLtoASQL'. Other settings include 'Description' (empty), 'Timeout' (7.00:00:00), 'Retry' (0), 'Retry interval' (30), and 'Secure output' and 'Secure input' checkboxes (unchecked).

In the **Source** tab, select the **DS_MSQSL_SalesLT_Customer** dataset. Select Query, and use the following:

```
SELECT CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, Suffix,  
CompanyName, SalesPerson, EmailAddress, Phone, PasswordHash, PasswordSalt,  
CAST(rowguid as NVARCHAR(50)) AS rowguid, ModifiedDate FROM SalesLT.Customer
```

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (1), 'Datasets' (2), and 'Data Flows (Preview)' (0). The main area displays a pipeline named 'Copy_SalesLT...'. Under 'Activities', the 'Source' tab is selected for a 'Copy Data' activity. The 'Source dataset' is set to 'DS_MSQL_SalesLT_Customer'. The 'Query' field contains the following SQL script:

```
SELECT CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, Suffix, CompanyName, SalesPerson, EmailAddress, Phone, PasswordHash, PasswordSalt, CAST(rowguid as NVARCHAR(50)) AS rowguid, ModifiedDate FROM SalesLT.Customer
```

In the **Sink** tab, select the **DS_ASQL_SalesLT_Customer** dataset.

In the Pre-copy Script, enter the following:

```
TRUNCATE TABLE [AWLTSRC].[SalesLT_Customer]
```

The screenshot shows the Microsoft Azure Data Factory Pipeline Editor with the same pipeline and activity setup as the previous screenshot, but with the 'Sink' tab selected. In the 'Sink' tab, the 'Sink dataset' is set to 'DS_ASQL_SalesLT_Customer'. The 'Pre-copy script' field contains the following SQL script:

```
TRUNCATE TABLE [AWLTSRC].[SalesLT_Customer]
```

Click **Validate** on the pipeline toolbar above the design surface to validate the pipeline settings. Confirm that the pipeline has been successfully validated. To close the validation output, select the >> (right arrow) button.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (1), 'Datasets' (2), and 'Data Flows (Preview)' (0). The main workspace displays a pipeline named 'Copy_SalesLT...'. A 'Copy Data' activity is selected, showing its configuration under the 'Sink' tab. The 'Sink dataset' is set to 'DS_ASQL_SalesLT_Cust'. The 'Pre-copy script' field contains the SQL command: 'TRUNCATE TABLE [AWLTSRC].[SalesLT_Customer]'. The validation output on the right indicates that the pipeline has been validated successfully with no errors found.

Microsoft Azure | Data Factory > sqlbitsadfcw01adf

Search resources

Data Factory Publish All Validate All Refresh Discard All ARM Template

Factory Resources Pipelines 1 Datasets 2 Data Flows (Preview) 0

Filter resources by name

Activities

Save as temp 1 Validate

Copy Data

CopyFromMSQLtoAS...

General Source Sink Mapping

Sink dataset * DS_ASQL_SalesLT_Cust

Stored Procedure Name Select... Edit

Pre-copy script TRUNCATE TABLE [AWLTSRC].[SalesLT_Customer]

Write batch timeout

Write batch size 10000

Pipeline Validation Output

Your Pipeline has been validated.
No errors were found.

Click **Publish All** to save.

Activity Summary

You have now created your first Pipeline. Next, you will debug and trigger the Pipeline.

Activity 07: Execute Pipeline

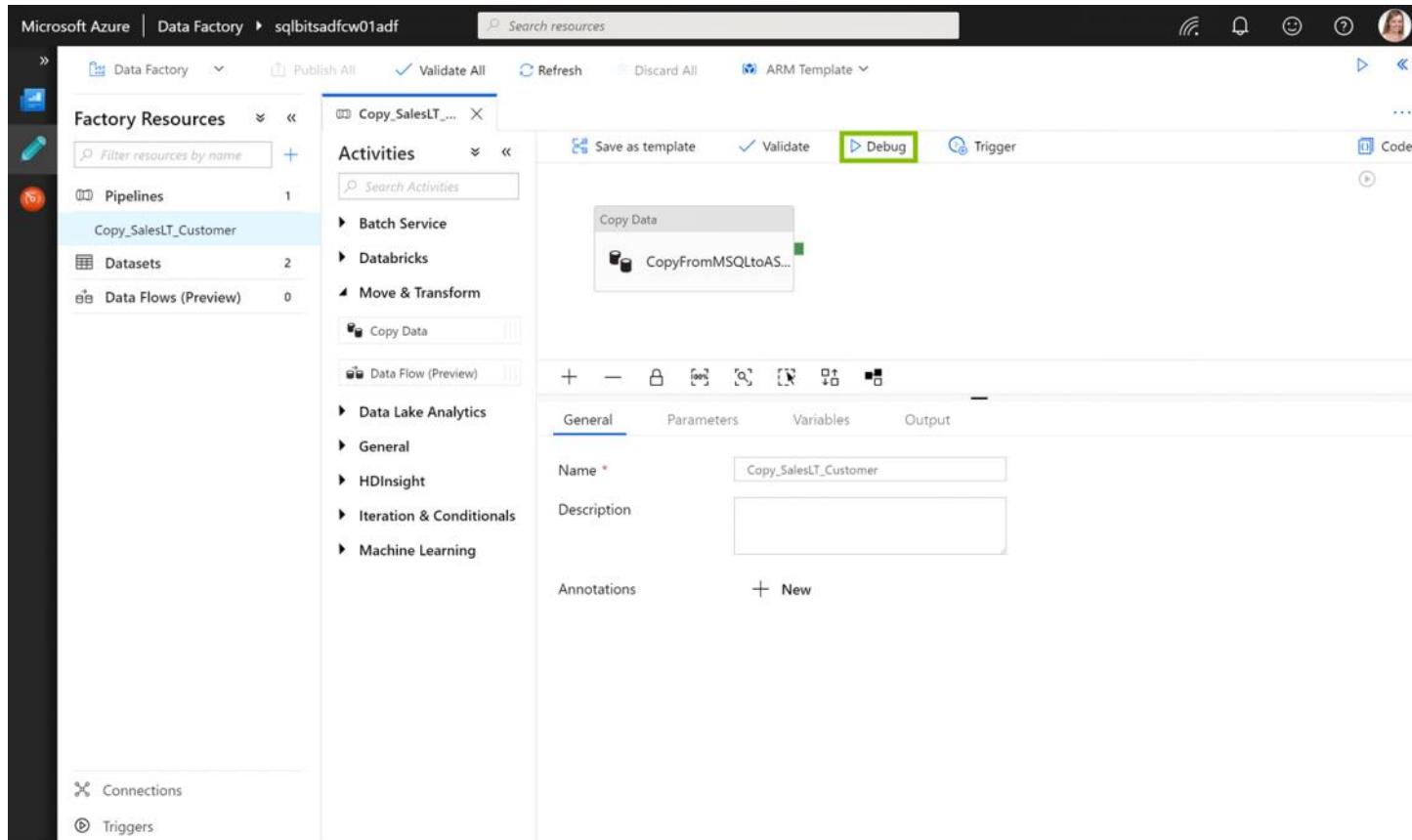
Activity Overview

Estimated time to complete activity: **5 minutes**.

In this activity, you will debug and then trigger the Pipeline created in the previous step.

Debug the Pipeline

Click Debug on the pipeline toolbar above the design surface to trigger a test run.



The **Output** tab will open, and you will see the status of the pipeline run. Also notice the other debug indicators:

1. **Yellow tab bar**
2. **In Progress icon** on Copy Data activity
3. **View Active Debug Runs icon** in Azure Data Factory toolbar

This screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (1), 'Datasets' (2), and 'Data Flows (Preview)'. The main area displays the 'Copy_SalesLT...' pipeline. The 'Activities' section shows a 'Copy Data' activity named 'CopyFromMSQLtoAS...'. A green callout '1' points to the pipeline name in the sidebar. A green callout '2' points to the activity name in the activities list. The 'Output' tab of the pipeline details is highlighted with a green border. It shows a table with one row:

NAME	TYPE	RUN START	DURATION	STATUS	ACTIONS	RUNID
CopyFromMSQLtoAS...	Copy	00:00:05		In Progress	Refresh Cancel	b6adeb48-b61f-424c

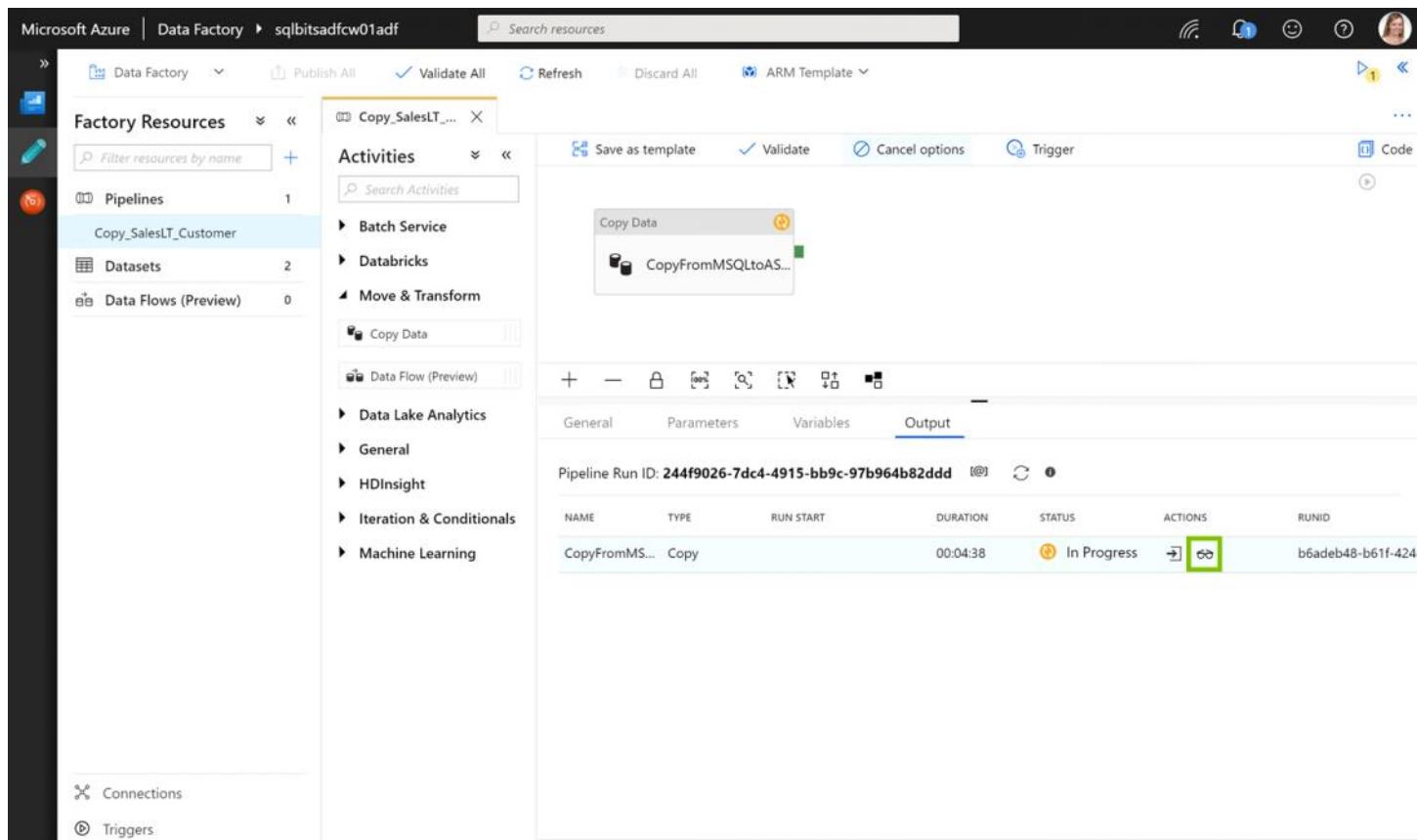
If you hover over the **information icon**, you will see helpful tips and explanations. In this case, it informs you that if the pipeline run takes longer than 5 minutes, you have to **manually refresh** to see the updated status.

This screenshot shows the same Microsoft Azure Data Factory interface as the previous one, but with a tooltip displayed. A green callout points to the information icon in the pipeline run status bar. A larger green callout encloses the status bar, which contains the following text:

Auto refresh every 20 secs for 5 minutes. Click "Refresh" manually for runs taking more than 5 minutes.

If you click the **glasses icon**, you will see the details of the Copy Data activity.

Screenshot of the Microsoft Azure Data Factory pipeline editor showing a 'Copy_SalesLT...' pipeline. The pipeline contains a single 'Copy Data' activity named 'CopyFromMSQLtoAS...'. The pipeline run ID is 244f9026-7dc4-4915-bb9c-97b964b82ddd, and the status is 'In Progress'.



In the **Details** window, you can see information about performance, data and rows read / written, duration, and more. This will give you a good indication of the performance of your Copy Data activity. You can learn more about the Copy Data performance between various sources and sinks by clicking on the link.

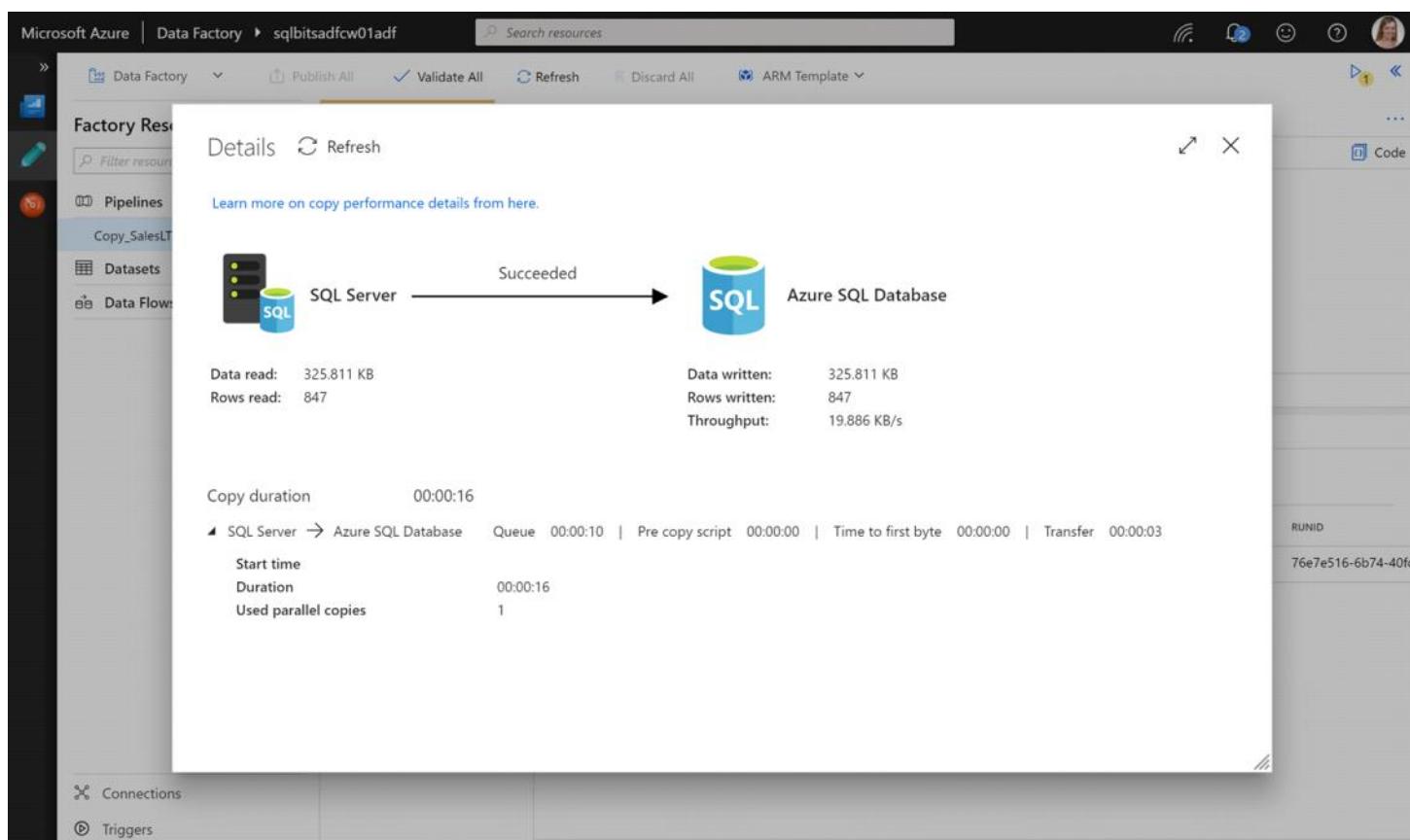
Screenshot of the Microsoft Azure Data Factory pipeline editor showing the 'Details' window for the 'Copy_SalesLT...' pipeline. The activity is named 'CopyFromMSQLtoAS...' and has succeeded. The source is 'SQL Server' and the sink is 'Azure SQL Database'. The copy duration was 00:00:16. The following performance metrics are displayed:

Metric	Value
Data read:	325.811 KB
Rows read:	847
Data written:	325.811 KB
Rows written:	847
Throughput:	19.886 KB/s

The 'Details' window also shows the command line for the activity:

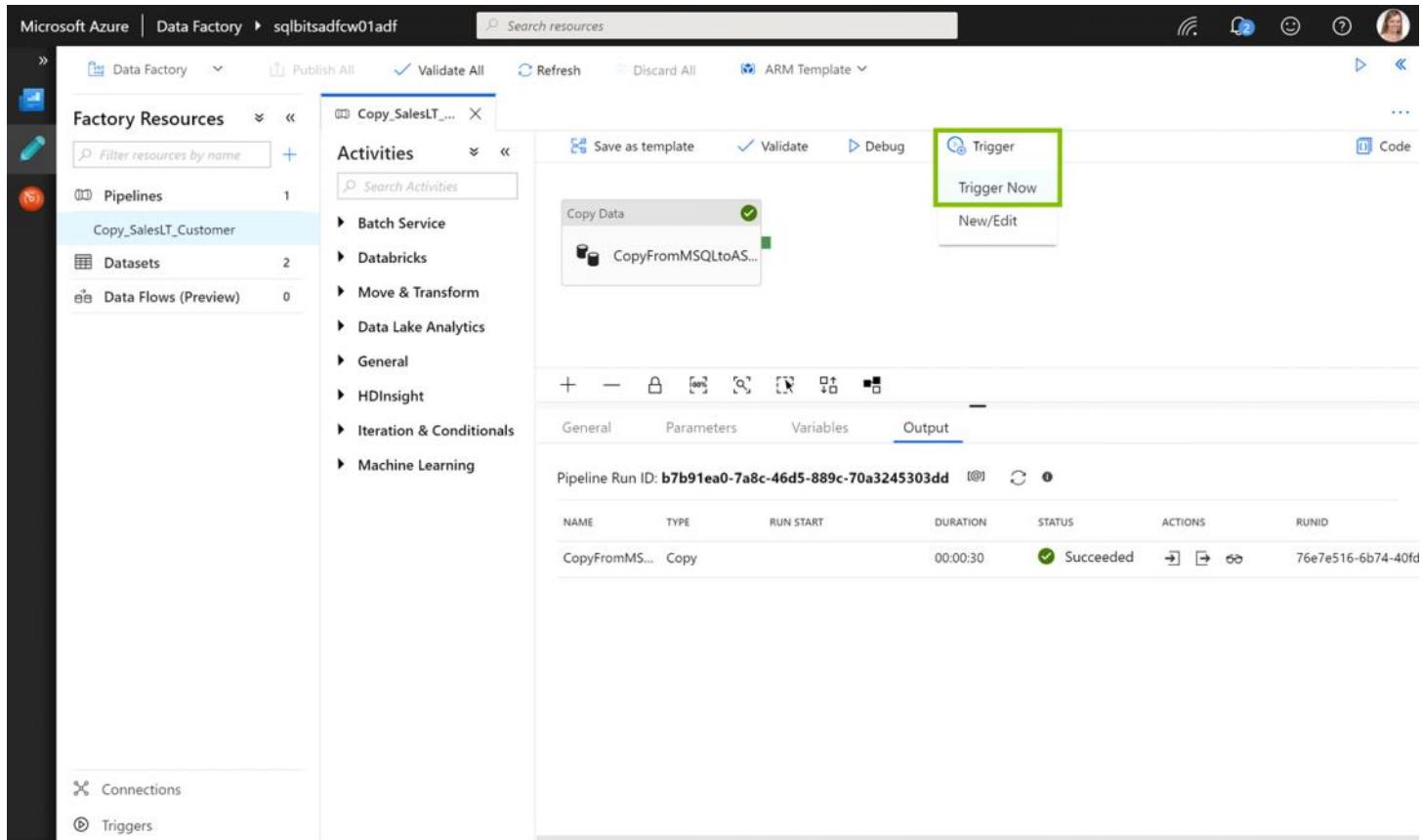
```
SQL Server → Azure SQL Database Queue 00:00:10 | Pre copy script 00:00:00 | Time to first byte 00:00:00 | Transfer 00:00:03
```

Start time: 00:00:16
Duration: 00:00:16
Used parallel copies: 1



Trigger the Pipeline Manually

To trigger the pipeline manually, select Trigger on the pipeline toolbar, and then select Trigger Now.



Notice the warning: "*Trigger pipeline now using last published configuration*". If you have unpublished changes in your pipeline, you must click **Publish All** before you can trigger a run of the new version. Otherwise, it will use the previously published version. Similarly, you have to publish a new pipeline before you can trigger a run of it.

Click **Finish**.

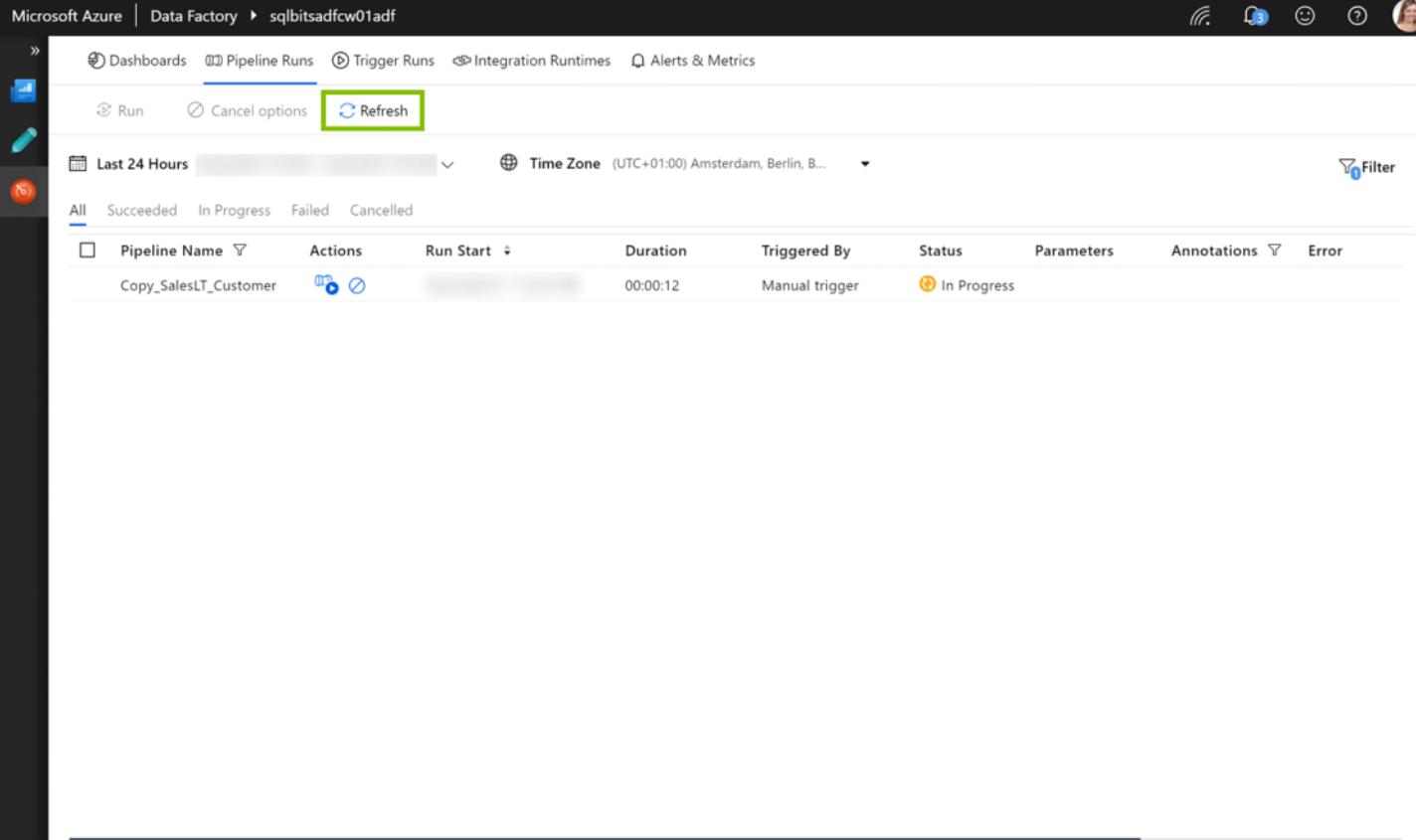
The screenshot shows the 'Pipeline Run' dialog box for a pipeline named 'Copy_SalesLT_Customer'. The dialog includes a warning message: 'Trigger pipeline now using last published configuration.' A 'Parameters' section shows no records found. The main area displays the pipeline's activities: 'Copy Data' and 'CopyFromMSQLtoAS...'. Below this is a toolbar with icons for General, Parameters, and Variables. The pipeline run ID is listed as 'b7b91ea0-7a8c-46d5-889c-70a32...' with a single activity entry: 'CopyFromMSQLtoAS... Copy'. At the bottom are 'Cancel' and 'Finish' buttons, with 'Finish' highlighted.

You will get a notification that the pipeline is **running**. Switch to **Monitor**.

The screenshot shows the Microsoft Azure Data Factory interface with the pipeline 'Copy_SalesLT_Customer' selected. A green callout '1' points to a notification bubble in the top right corner stating 'Running Copy_SalesLT_Customer'. Another green callout '2' points to the 'Pipelines' icon in the left sidebar. The pipeline details pane shows the 'Output' tab selected, displaying the pipeline run ID 'b7b91ea0-7a8c-46d5-889c-70a3245303dd' and a table of activity logs. The table has columns: NAME, TYPE, RUN START, DURATION, STATUS, ACTIONS, and RUNID. One row is shown: 'CopyFromMSQLtoAS... Copy' with status 'Succeeded'. The left sidebar also lists 'Connections' and 'Triggers'.

Monitor the Pipeline

Under Pipeline Runs, you will see the status of the pipeline. Click **Refresh** to see an updated status.



The screenshot shows the Microsoft Azure Data Factory Pipeline Runs page. At the top, there are navigation links: Dashboards, Pipeline Runs (which is selected), Trigger Runs, Integration Runtimes, and Alerts & Metrics. Below the navigation is a toolbar with Run, Cancel options, and Refresh buttons. The Refresh button is highlighted with a green box. A dropdown menu shows 'Last 24 Hours' and a Time Zone setting '(UTC+01:00) Amsterdam, Berlin, B...'. On the right, there is a Filter icon. Below the toolbar, there are tabs for All, Succeeded, In Progress, Failed, and Cancelled. The All tab is selected. The main table lists pipeline runs with columns: Pipeline Name, Actions, Run Start, Duration, Triggered By, Status, Parameters, Annotations, and Error. One run is listed: 'Copy_SalesLT_Customer' with status 'In Progress'. The 'Actions' column for this row contains icons for Stop and Refresh. The 'Status' column shows a blue circle with a white play/pause symbol. The 'Triggered By' column indicates 'Manual trigger'.

Click the **All / Succeeded / In Progress / Failed / Cancelled** tabs to quickly filter the view. Click **View Activity Runs** to see the details.

The screenshot shows the Microsoft Azure Data Factory Pipeline Runs page. At the top, there are navigation links: Dashboards, Pipeline Runs (which is underlined), Trigger Runs, Integration Runtimes, Alerts & Metrics, Run, Cancel options, and Refresh. Below this is a filter bar with 'Last 24 Hours' and 'Time Zone (UTC+01:00) Amsterdam, Berlin, B...'. A 'Filter' icon is also present. The main area displays a table of pipeline runs. The first row shows a run for 'Copy_SalesLT_Customer' with status 'Succeeded'. A green callout with the number '1' points to the 'Actions' column for this row. Another green callout with the number '2' points to the 'Details' link (eyeglasses icon) in the same column.

All	Succeeded	In Progress	Failed	Cancelled				
Pipeline Name	Actions	Run Start	Duration	Triggered By	Status	Parameters	Annotations	Error
Copy_SalesLT_Customer			00:00:18	Manual trigger	Succeeded			

To view details about the copy operation, select the Details (eyeglasses image) link in the Actions column. This will show you the same details as during debugging.

The screenshot shows the Microsoft Azure Data Factory Activity Runs page for the pipeline run 'Copy_SalesLT_Customer'. The title is 'All Pipeline Runs / Copy_SalesLT_Customer - Activity Runs'. The page includes a 'Refresh' button. The main content is titled 'Activity Runs' and shows a table for the pipeline run ID 'b4d0ea80-4f1e-450e-a8fb-bfd30171b1b8'. The table has columns: ACTIVITY NAME, ACTIVITY TYPE, ACTIONS, RUN START, DURATION, STATUS, INTEGRATION RUNTIME, USER PROPERTIES, and RUNID. One row is visible for the activity 'CopyFromMS...', which is of type 'Copy'. The 'Actions' column contains icons for Run, Details (highlighted with a green callout and the number '1'), and Cancel. The 'Status' column shows 'Succeeded' with a green checkmark. The 'INTEGRATION RUNTIME' column shows 'SelfHostedIntegrationRuntime'. The 'USER PROPERTIES' and 'RUNID' columns show their respective values.

ACTIVITY NAME	ACTIVITY TYPE	ACTIONS	RUN START	DURATION	STATUS	INTEGRATION RUNTIME	USER PROPERTIES	RUNID
CopyFromMS...	Copy			00:00:16	Succeeded	SelfHostedIntegrationRuntime	30add0a8-4dc5-4bc5-a0c3-c	

Activity Summary

You have now executed your first Pipeline. In the next lab, you will add parameters to make the pipeline dynamic, and continue building out your solution.

Lab 02: ADF Basics

Background and Goals

In Lab 01, we created a simple pipeline that copied data from a SQL Server source to an Azure SQL Database sink. In a typical big data pipeline, we need to perform this task many times over to copy data from all the source tables. Ideally, we want to eliminate the need to build redundant pipelines.

In this lab, you will learn how to build on the basics to create parameterized, dynamic, and reusable pipelines. These techniques are foundational to applying more advanced patterns in Azure Data Factory.

Prerequisites

In order to complete this lab, you will need to have previously completed Lab 01.

Overview

Activity 01: Adding Pipeline Parameters

- Create Folder
- Clone, Rename, and Move Pipelines
- Create Pipeline Parameters
- Use Pipeline Parameters in Activities
- Validate and Execute Pipelines

Activity 02: Copying Data From File Based Stores

- Upload files to Azure Data Lake Storage Gen1 using Azure Data Explorer
- Create new Datasets
- Create new Pipelines
- Publish and Debug Pipelines

Activity 03: Creating a Master Pipeline

- Create Precedence Constraints
- Execute Other Pipelines
- Trigger Pipelines

Activity 01: Adding Pipeline Parameters

Activity Overview

Estimated time to complete activity: **10 minutes**.

In Lab 01, we created a simple pipeline that copied data from a SQL Server source to an Azure SQL Database sink. In a typical big data pipeline, we need to perform this task many times over to copy data from all the source tables. Ideally, we want to eliminate the need to build redundant pipelines.

In this activity, we will create pipeline parameters to implement dynamic behavior. This technique is foundational to applying more advanced patterns in Azure Data Factory.

Create a New Folder

Under Factory Resources, hover over Pipelines until the ellipsis shows up, click it, then click **Add Folder**.

The screenshot shows the Microsoft Azure Data Factory pipeline editor interface. On the left, the 'Factory Resources' sidebar is open, showing 'Pipelines' selected. Under 'Pipelines', there are three options: 'Add Pipeline', 'Pipeline from template', and 'Add Folder'. The 'Add Folder' option is highlighted with a green box. The main workspace shows a pipeline named 'Copy_SalesLT...' with one activity, 'Copy Data', which has a sub-activity 'CopyFromMSQLtoAS...'. The top navigation bar includes 'Search resources', 'Validate All', 'Refresh', 'Discard All', 'ARM Template', and other standard DevOps tools like 'Save as template', 'Validate', 'Debug', and 'Trigger'. The bottom navigation bar includes tabs for 'General', 'Parameters', 'Variables', and 'Output', with 'General' currently selected. The pipeline name 'Copy_SalesLT...' is set in the 'Name' field.

Name the folder **Lab 02**.

The screenshot shows the Microsoft Azure Data Factory interface. In the left sidebar, under 'Factory Resources', 'Pipelines' is selected, showing one pipeline named 'Copy_SalesLT_Customer'. This pipeline is highlighted with a green border. Below it are 'Datasets' (2) and 'Data Flows (Preview)' (0). On the right, the main workspace displays the 'Copy_SalesLT...' pipeline details. The 'Activities' section lists various options like 'Batch Service', 'Databricks', etc., with 'Move & Transform' expanded to show 'Copy Data' as the active activity. The pipeline configuration pane shows the 'General' tab selected, with the name set to 'Copy_SalesLT_Customer'. Other tabs include 'Parameters', 'Variables', and 'Output'. At the top of the workspace, there are buttons for 'Save as template', 'Validate', 'Debug', 'Trigger', and 'Code'.

Clone, Rename, and Move the Existing Pipeline

Hover over the **Copy_SalesLT_Customer** pipeline until the ellipsis shows up, then click **Clone**.

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' pane is open, showing a list of pipelines, datasets, and data flows. A pipeline named 'Copy_SalesLT_Customer' is selected and highlighted with a green border. To its right, a detailed view of the pipeline's activities is shown. The 'Activities' pane lists various options like Batch Service, Databricks, Move & Transform, etc. Below it, the pipeline's configuration is displayed under the 'General' tab, with fields for 'Name' (set to 'Copy_SalesLT_Customer') and 'Description'. The pipeline has one activity, 'CopyFromMSQLtoAS...', which is a 'Copy Data' type activity.

The new pipeline will get the default name **pipeline1** and will be automatically opened. Rename the pipeline to **Copy_SalesLT_CustomerV2**.

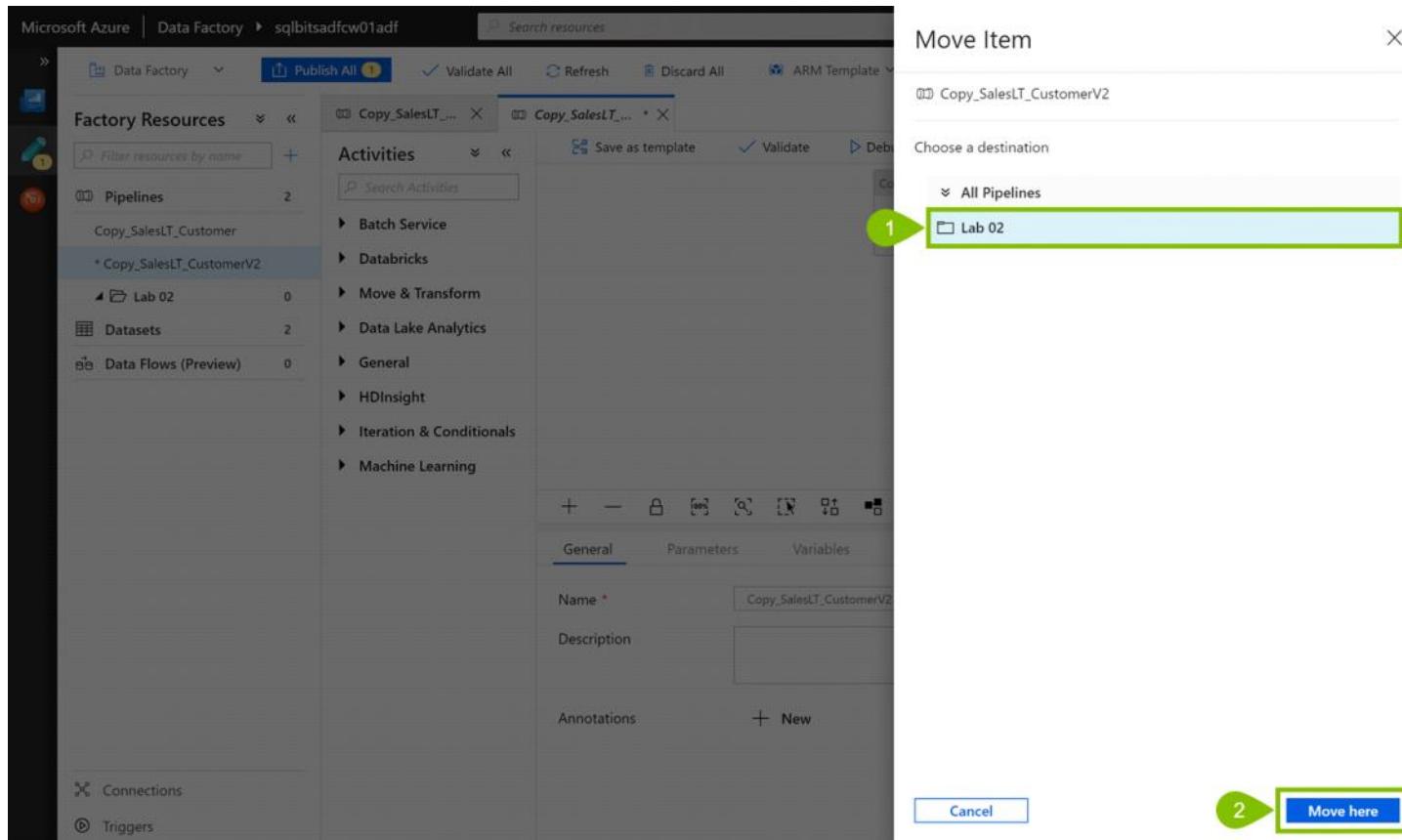
This screenshot shows the same Azure Data Factory interface after the pipeline has been renamed. The 'Factory Resources' pane now lists two pipelines: 'Copy_SalesLT_Customer' and a newly created pipeline named '* Copy_SalesLT_CustomerV2'. The newly created pipeline is selected and highlighted with a green border. In the pipeline's configuration view, the 'Name' field is explicitly set to 'Copy_SalesLT_CustomerV2', with the entire field highlighted by a green dashed border. The rest of the pipeline configuration (Description, Annotations) remains the same.

Move the new pipeline into the **Lab 02** folder by dragging and dropping it onto the folder name.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (2), with 'Copy_SalesLT_CustomerV2' selected and highlighted with a green border. Below it are 'Copy_SalesLT_Customer' and 'Datasets' (2). Other sections like 'Data Flows (Preview)' and 'Connections' are also visible. The main workspace shows a pipeline named 'Copy_SalesLT...'. The 'Activities' pane on the left lists various activity types: Batch Service, Databricks, Move & Transform, Data Lake Analytics, General, HDInsight, Iteration & Conditionals, and Machine Learning. The 'General' tab in the pipeline configuration pane is active, showing the pipeline's name as 'Copy_SalesLT_CustomerV2' and its description as empty. A 'Copy Data' activity is present in the pipeline.

If you prefer, you can use the **Move Item** option instead. This is a very useful option once you add more factory resources, as dragging and dropping *can* be difficult when you have to scroll up and down.

This screenshot is similar to the previous one but shows the 'Move Item' option being used. The 'Copy_SalesLT_CustomerV2' pipeline is selected in the sidebar, and a context menu is open over it, with 'Move Item' highlighted. The rest of the interface is identical to the first screenshot, showing the pipeline editor with its activities and general configuration.



Add Pipeline Parameters

There are two ways to create pipeline parameters. One is fairly straightforward, while the other can be easy to miss. To familiarize you with both approaches, we will demonstrate both in this step. The result should be that your pipeline has the following two parameters:

Parameter	Name	Type	Default Value
1	SourceQuery	String	<code>SELECT CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, Suffix, CompanyName, SalesPerson, EmailAddress, Phone, PasswordHash, PasswordSalt, CAST(rowguid as NVARCHAR(50)) AS rowguid, ModifiedDate FROM SalesLT.Customer</code>
2	SinkPreCopyScript	String	<code>TRUNCATE TABLE [AWLTSRC].[SalesLT_Customer]</code>

Create Pipeline Parameter from Pipeline

We will create the first parameter from the pipeline. Click anywhere on the pipeline design surface (outside the Copy Data activity) and choose the **Parameters** tab. We will create the first parameter this way, so click **+ New** once.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (2), 'Copy_SalesLT_Customer' (1), 'Datasets' (2), and 'Data Flows (Preview)' (0). The main area displays a 'Copy Data' activity named 'CopyFromMSQLtoAS...'. Below the activity, the 'Parameters' tab is selected, indicated by a green box labeled '1'. A green box labeled '2' highlights the '+ New' button under the parameters list.

Add the **SourceQuery** parameter.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. The 'Parameters' tab is selected for the 'Copy Data' activity. A new parameter is being added, with the 'NAME' field set to 'SourceQuery', 'TYPE' set to 'String', and 'DEFAULT VALUE' set to 'SELECT CustomerID, NameStyle, Title, Firs'. A green box highlights the 'SourceQuery' entry in the NAME column.

While you are in the Parameters tab, familiarize yourself with the UI by trying the following:

- Create a new parameter, then delete it. (Use the checkbox.)

- Resize the parameter columns. (Hover over Name / Type / Default Value)

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, and Data Flows (Preview). In the center, a pipeline named 'Copy_SalesLT...' is open, showing a 'Copy Data' activity with the sub-activity 'CopyFromMSQLtoAS...'. Below the activity, the 'Parameters' tab is selected in the ribbon. A table lists parameters: 'SourceQuery' (String, Default Value: 'SELECT CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, ...') and 'Name' (String, Default Value: 'Value'). Three green arrows highlight specific UI elements: one arrow points to the 'Delete' button in the toolbar, another to the 'New' button, and a third to the 'Add dynamic content [Alt+P]' link in the top right corner of the parameter table.

NAME	TYPE	DEFAULT VALUE
SourceQuery	String	SELECT CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, ...
<input checked="" type="checkbox"/> Name	String	Value

Create Pipeline Parameter from Activity

We will create the second parameter from the **Copy Data activity**. Click on the activity and choose the **Sink** tab. Then, click inside the **Pre-copy Script** box and Notice that a new link shows up: **Add dynamic content [Alt+P]**.

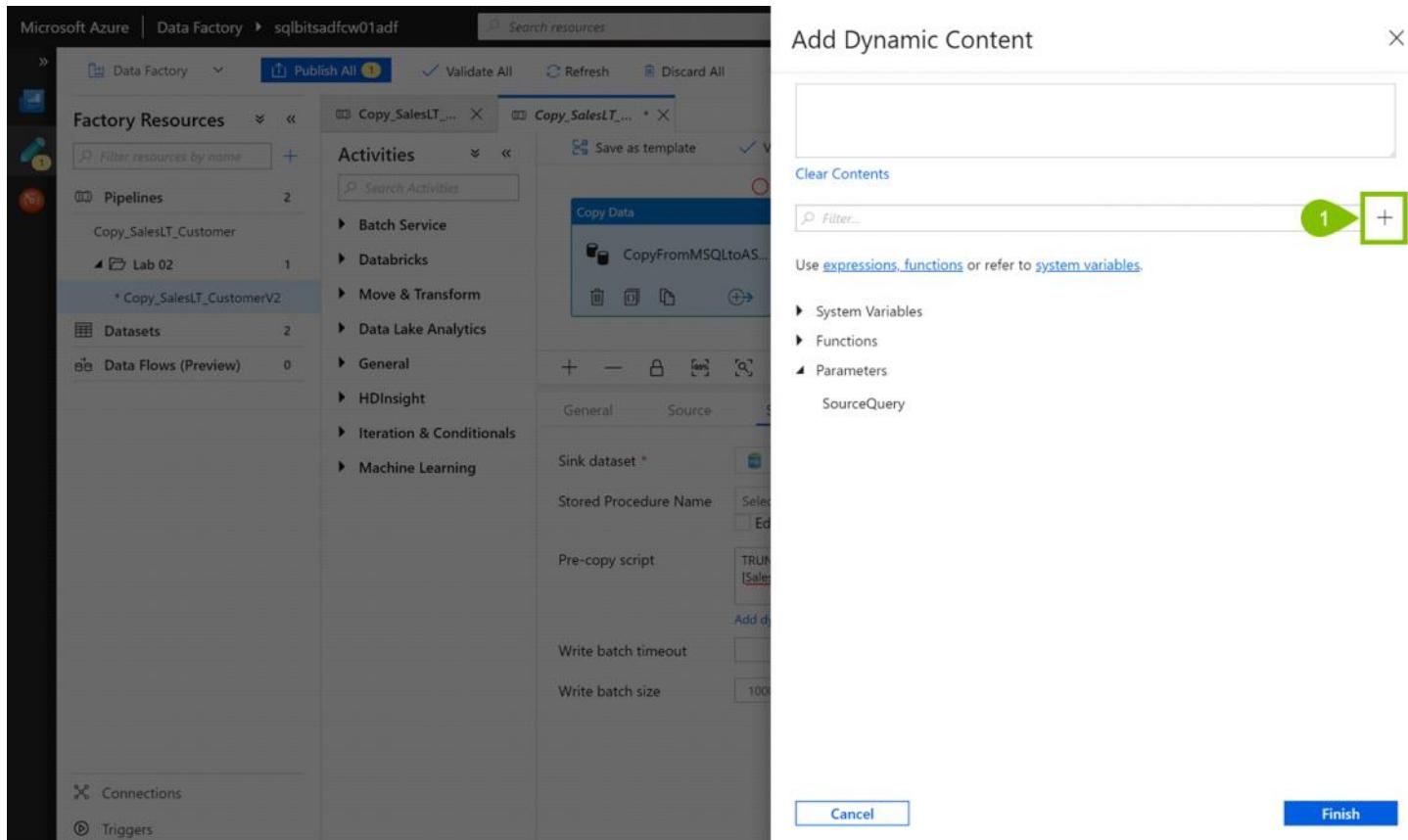
The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, and Data Flows (Preview). In the center, a pipeline named 'Copy_SalesLT...' is selected. A specific activity, 'CopyFromMSQLtoAS...', is highlighted. The 'Sink' tab is active in the activity configuration pane. Under the 'Sink dataset' dropdown, 'DS_ASQL_SalesLT_Customer' is selected. In the 'Pre-copy script' section, the query 'TRUNCATE TABLE [AWLTSRC].[SalesLT].[Customer]' is displayed. A green box highlights the 'Add dynamic content [Alt+P]' link next to the query.

Click on the **Add dynamic content** link, or press **Alt+P**. The Add Dynamic Content pane will open, and you will see the previously created parameter listed.

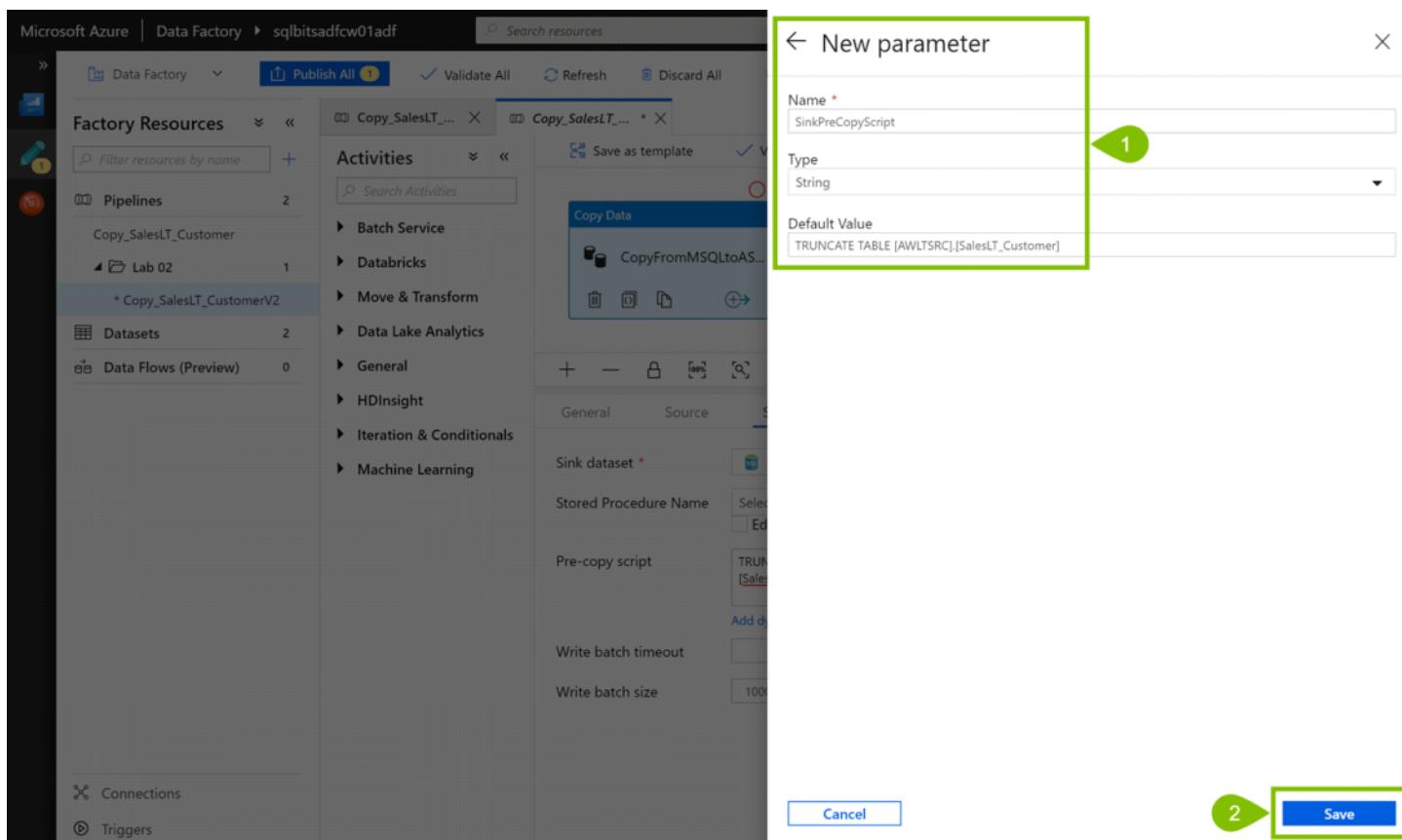
The screenshot shows the Microsoft Azure Data Factory pipeline editor with the 'Add Dynamic Content' pane open. The pane has a search bar at the top and a list of content categories: System Variables, Functions, Parameters, and SourceQuery. A green box highlights the 'SourceQuery' category under 'Parameters'. The main pipeline area shows the same pipeline and activity configuration as the previous screenshot, with the 'Sink' tab active.

Next to Filter, you will see a +. This is very easy to miss, but you can actually create a new parameter

from here! Click the Plus Sign.



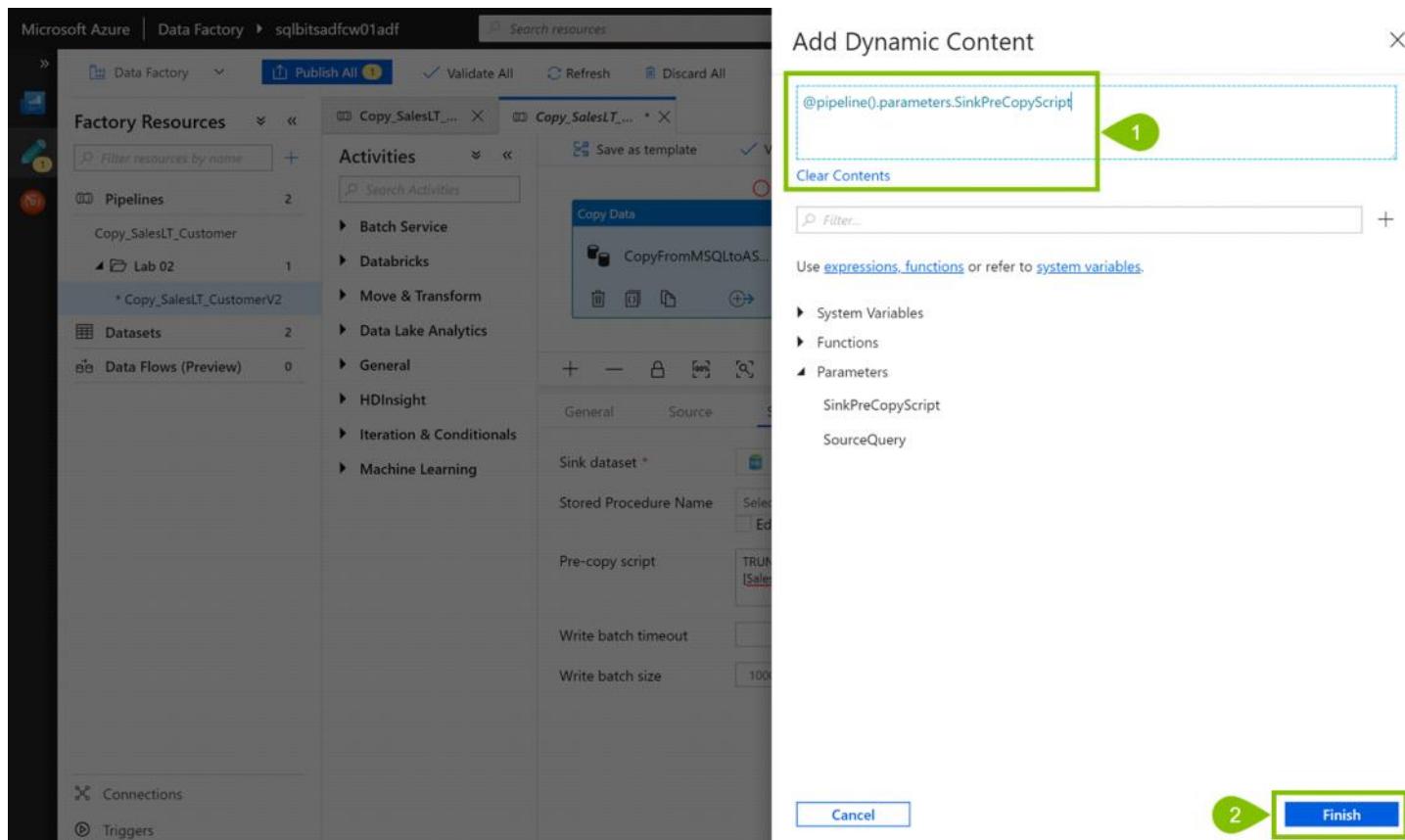
In the New Parameter pane, create the **SinkPreCopyScript** parameter. Click Save.



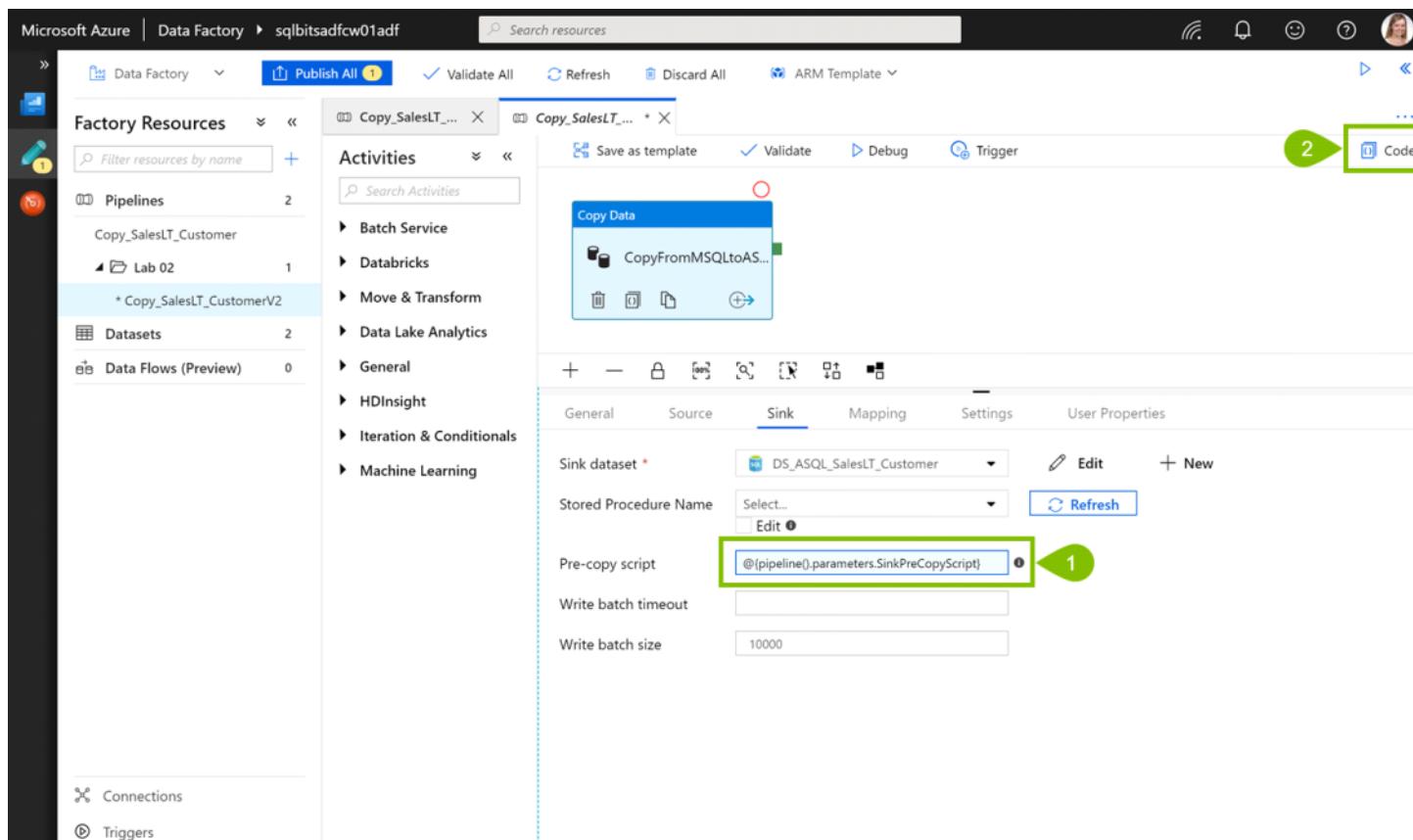
The parameter will get saved, but it will not be added to the dynamic content automatically, so you will see an **Invalid** warning. Click the new **SinkPreCopyScript** parameter to add it.

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists Pipelines, Datasets, and Data Flows. In the center, a pipeline named 'Copy_SalesLT_CustomerV2' is selected. The 'Activities' pane shows a 'Copy Data' activity named 'CopyFromMSQLtoAS...'. On the right, the 'Add Dynamic Content' dialog is open, specifically for the 'Sink dataset' section of the 'Copy Data' activity. The 'Sink dataset' dropdown is expanded, showing 'Stored Procedure Name' and 'Pre-copy script'. The 'Pre-copy script' field contains the value 'TRUNCATE TABLE [SalesLT].[SalesOrderDetail]'. A green callout labeled '1' points to the 'Invalid' status message at the top of the dialog, which reads 'Please specify an expression'. Another green callout labeled '2' points to the 'SinkPreCopyScript' parameter in the 'Parameters' list, which is highlighted with a red border. At the bottom right of the dialog are 'Cancel' and 'Finish' buttons.

You will now see the **@pipeline().parameters.SinkPreCopyScript** syntax, and the text and text box turns blue. Click **Finish**.



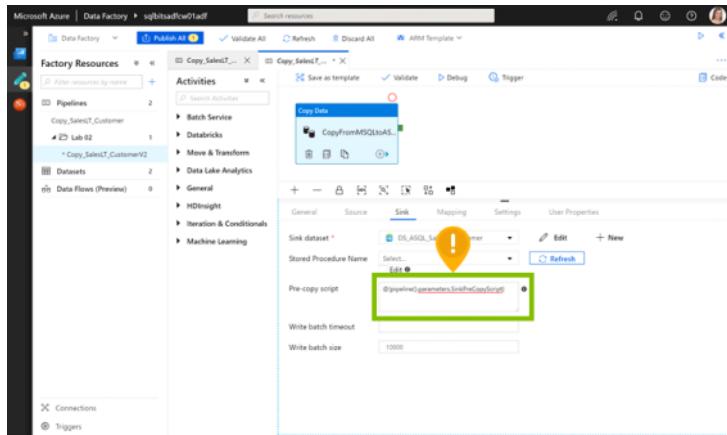
Notice that the Pre-copy script box has now turned **blue**. This means that you are using dynamic content. You can also click on the **Code** to verify that you are using dynamic content.



Under sink, verify that the **preCopyScript** is now **Type: Expression**.

Note: It is **very important** that you **always go through the Add dynamic content [Alt+P] pane when adding parameters**. If you simply type in an expression directly, the content will not be interpreted as dynamic content. You can see this in two ways:

- The text box has not turned blue
 - The property is not set to Type: Expression



```

1
2 "name": "Copy_SalesLT_CustomerV2",
3 "properties": {
4   "activities": [
5     {
6       "name": "CopyFromMySQLtoASQL",
7       "type": "Copy",
8       "policy": {
9         "maxOutput": "7.00:00:00",
10        "retry": 0,
11        "retryIntervalInSeconds": 30,
12        "secureOutput": false,
13        "secureInput": false
14      },
15      "typeProperties": {
16        "source": {
17          "type": "SqlSource",
18          "sqlReaderQuery": "SELECT CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, CompanyName, SalesLT.Customer"
19        }
20      },
21      "sink": {
22        "type": "SqlSink",
23        "writeBatchSize": 10000,
24        "preCopyScript": "@{pipeline().parameters.SinkPreCopyScript}"
25      }
26    }
27  ]
28 }

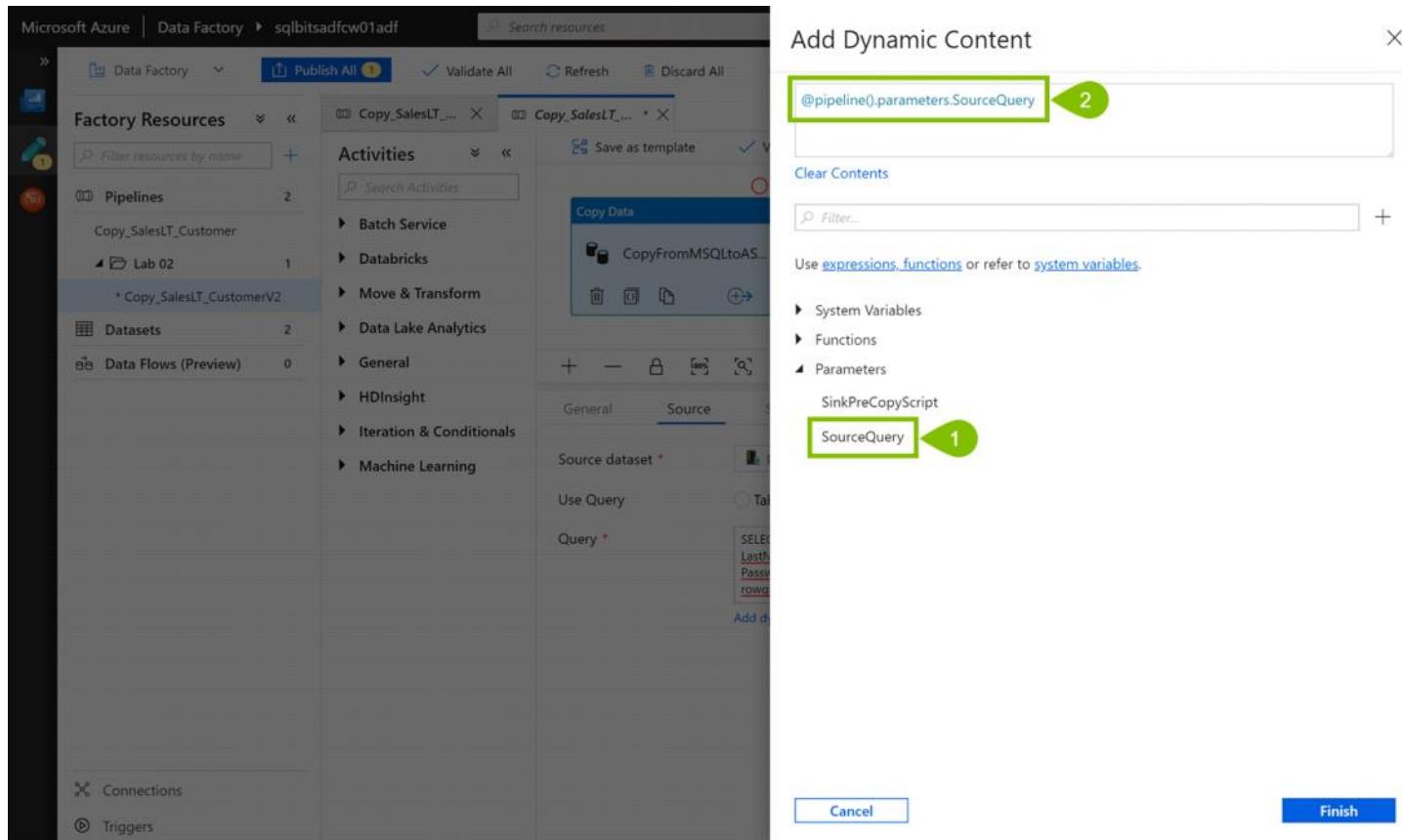
```

Go back to the Pipeline Parameters and verify that both parameters have been added successfully.

NAME	TYPE	DEFAULT VALUE
SourceQuery	String	SELECT CustomerID, NameStyle, Title, FirstName, MiddleName, LastName, CompanyName, SalesLT.Customer
SinkPreCopyScript	String	TRUNCATE TABLE [AWLTSRC].[SalesLT].[Customer]

Use Parameters in Copy Data Activity

Click on the **CopyFromMySQLtoASQL** Copy Data activity and go to the **Source** tab. Click inside the **Query** box, and open the **Add Dynamic Content** pane by clicking the blue link or pressing Alt+P. Click on the **SourceQuery** parameter to add it, and click **Finish**.



Go to the **Sink** tab and verify that it is using the **SinkPreCopyScript** pipeline parameter that we created in the previous step.

Validate and Execute Pipeline

Once you have replaced the hardcoded values with the new pipeline parameters, you can validate and debug / trigger the pipeline.

Click on **Validate** and ensure that you have no errors.

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (2), 'Copy_SalesLT_Customer' (1), 'Lab 02' (1), 'Datasets' (2), and 'Data Flows (Preview)' (0). The main area displays the 'Activities' section for the 'Copy_SalesLT...' pipeline. A green box highlights the 'Validate' button. To the right, the 'Pipeline Validation Output' pane shows a blue bar chart icon with three green bars. A green box highlights the message: 'Your Pipeline has been validated. No errors were found.'

Click on **Debug** and notice that you now have to specify the pipeline parameters before debugging. You can choose to accept the default parameters, or replace them.

The screenshot shows the 'Pipeline Run' dialog for the 'Copy_SalesLT...' pipeline. A green box highlights the 'Parameters' section. It contains two entries: 'SourceQuery' (String) with the value 'SELECT CustomerID, NameStyle,...' and 'SinkPreCopyScript' (String) with the value 'TRUNCATE TABLE [AWLTSRC].[S...]'.

NAME	TYPE	VALUE
SourceQuery	String	SELECT CustomerID, NameStyle,...
SinkPreCopyScript	String	TRUNCATE TABLE [AWLTSRC].[S...]

Verify that the pipeline run **succeeded**, and **Publish** to save your new pipeline.

The screenshot shows the Microsoft Azure Data Factory pipeline editor. On the left, the 'Factory Resources' sidebar lists 'Pipelines' (Copy_SalesLT_Customer, Lab 02, Copy_SalesLT_CustomerV2), 'Datasets', and 'Data Flows (Preview)'. The main area displays the 'Copy_SalesLT...' pipeline. A green callout points to the 'Publish All' button at the top left of the pipeline editor. Another green callout points to the status 'Succeeded' for the 'CopyFromMSQLtoAS...' activity in the 'Output' tab of the pipeline run details.

TIP: We typically recommend debugging the package once with default values to make sure nothing is broken, and then re-running it with new values to verify that the dynamic behavior is working as expected. You can skip the second run in this lab.

Activity Summary

You have now created an updated version of a pipeline that uses pipeline parameters instead of hardcoded values. In the next activity, we will continue to build out our solution and add more data sources. We will copy new data from a JSON file and a text file stored in ADLS v1 to our Azure SQL Database.

Activity 02: Copying Data From File Based Stores

Activity overview

Estimated time to complete activity: **10 minutes.**

In Lab 01, we created a simple pipeline that copied data from a SQL Server source to an Azure SQL Database sink. In the previous activity, we parameterized this pipeline. While copying data from SQL Server to Azure SQL Database is a common use case, you will likely also have to load data from other sources such as an Azure Data Lake.

In this activity, we will load data from a JSON and a Text file in Azure Data Lake Storage Gen2 Data Lake container to Azure SQL Database. The sink dataset will be parameterized and reused for both file copies.

Download Files from GitHub

Download **Address.json** and **CustomerAddress.txt** from
<https://github.com/jasonhorner/adfdeepdive/tree/master/Labs/02%20-%20ADF%20Basics>.

(You can download them directly or work from a clone of the repository. This is entirely up to you and how comfortable you are working with Git clients.)

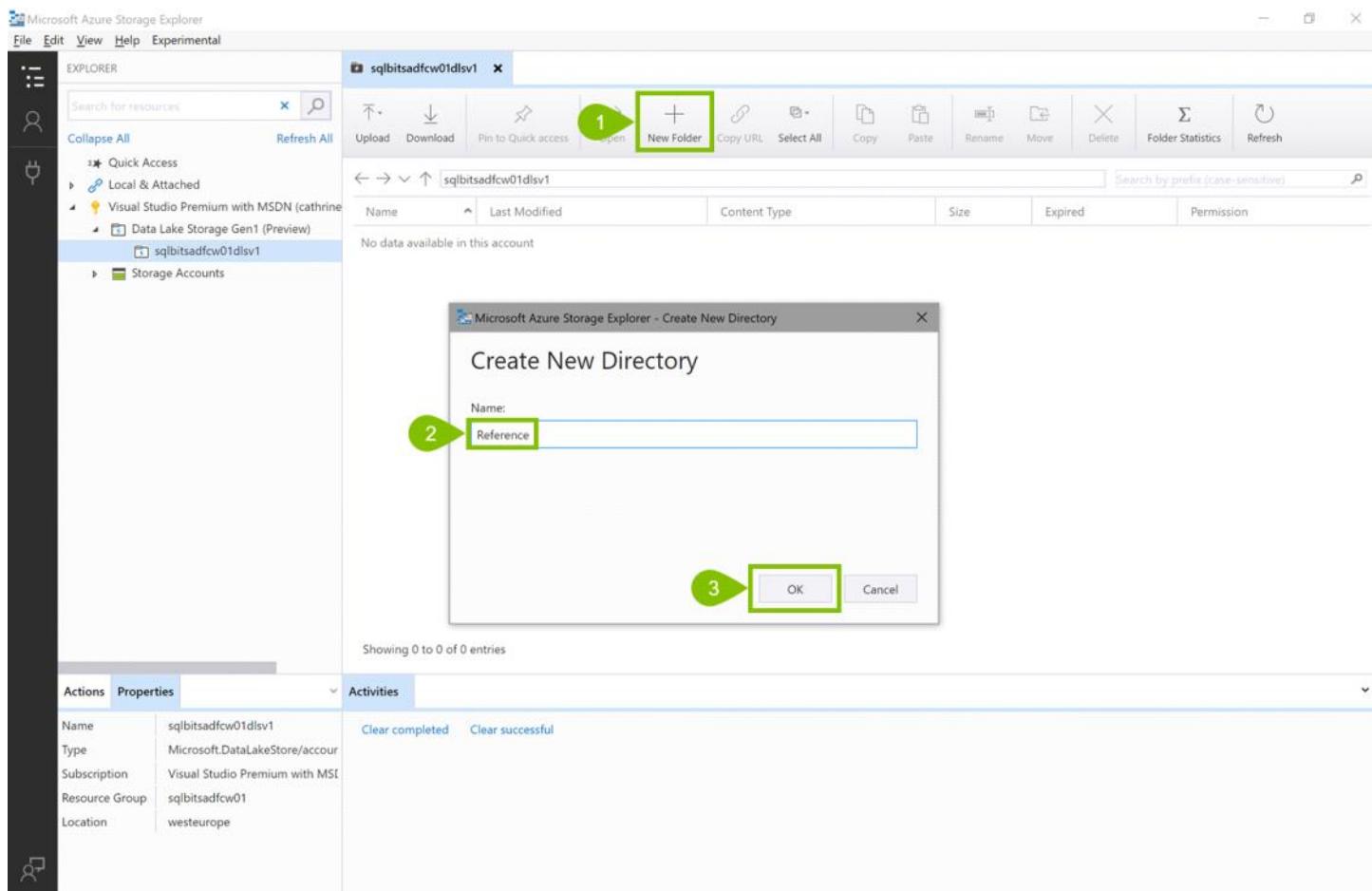
Upload Files to Azure Data Lake Storage Gen1

Open Azure Storage Explorer and navigate to the **sqlbitsadf<999>dlsv1** Azure Data Lake Storage Gen1 you created in the prerequisites.

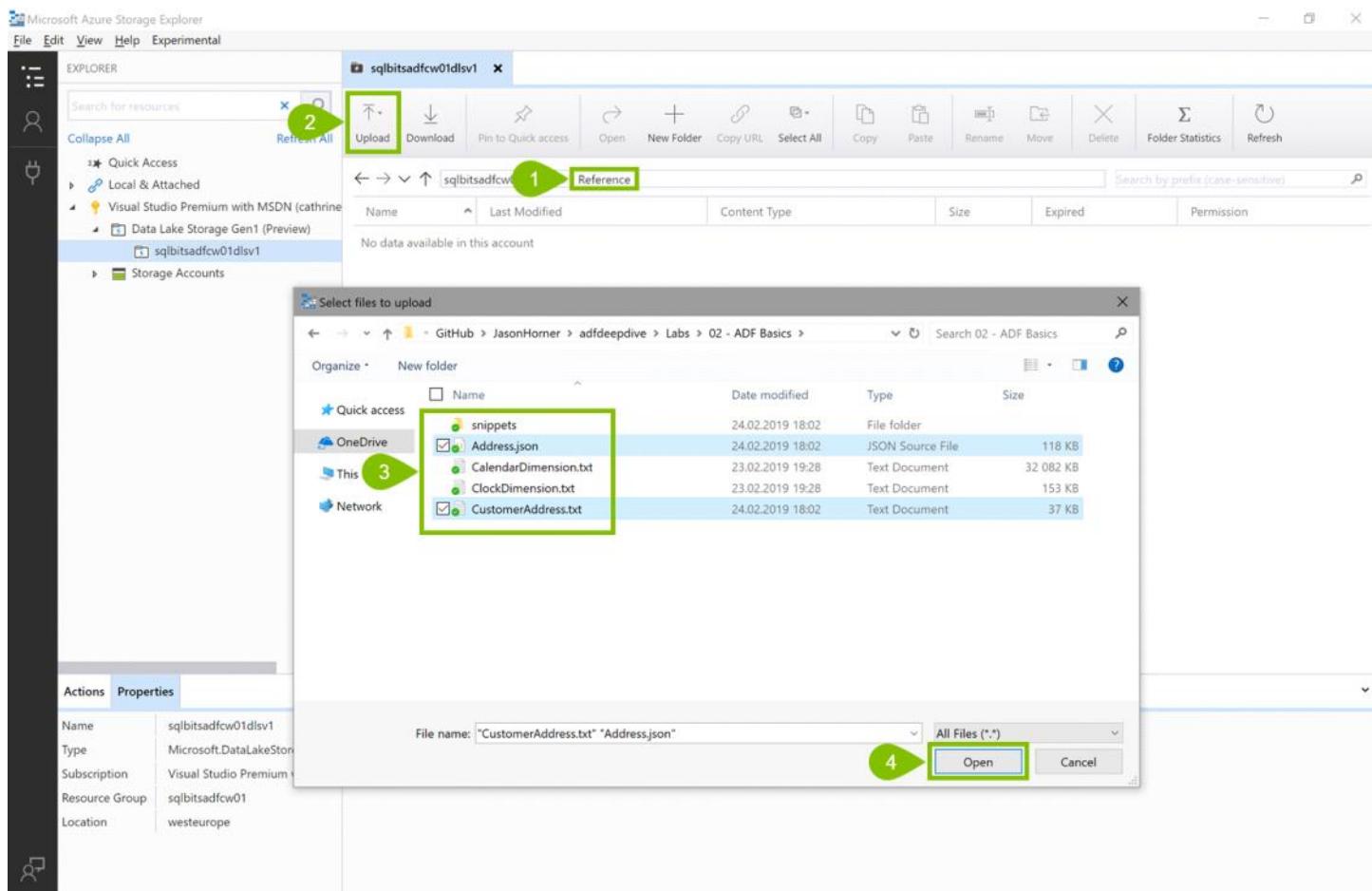
The screenshot shows the Microsoft Azure Storage Explorer interface. On the left, the 'EXPLORER' sidebar lists 'Quick Access', 'Local & Attached', and a 'Visual Studio Premium with MSDN (catherine)' section containing a folder named 'sqlbitsadfcw01dlsv1'. This folder is highlighted with a green box. Below it is a 'Storage Accounts' item. The main pane is titled 'sqlbitsadfcw01dlsv1' and shows a search bar with the query 'sqlbitsadfcw01dlsv1'. It includes standard file operations like Upload, Download, Pin to Quick access, Open, New Folder, Copy URL, Select All, Copy, Paste, Rename, Move, Delete, and Folder Statistics. A 'Search by prefix (case-sensitive)' input field is also present. The table below lists columns for Name, Last Modified, Content Type, Size, Expired, and Permission. A message states 'No data available in this account'. At the bottom, it says 'Showing 0 to 0 of 0 entries'. The 'Properties' tab is active, displaying resource details:

Name	sqlbitsadfcw01dlsv1
Type	Microsoft.DataLakeStore/account
Subscription	Visual Studio Premium with MSI
Resource Group	sqlbitsadfcw01
Location	westeurope

Create a new **Folder** called **Reference**.



Ensure you are navigated to **Reference**. Click **Upload Files**. Select the **Address.json** and **CustomerAddress.txt** files you downloaded, and click **Open**.



Verify that the files have been **uploaded successfully**.

The screenshot shows the Microsoft Azure Storage Explorer interface. On the left, the 'EXPLORER' sidebar lists 'Quick Access', 'Local & Attached', and 'Visual Studio Premium with MSDN (catherine)'. Under 'Data Lake Storage Gen1 (Preview)', the folder 'sqlbitsadfcw01dlsv1' is selected, showing two files: 'Address.json' and 'CustomerAddress.txt'. The 'Properties' tab is active, displaying the storage account details: Name (sqlbitsadfcw01dlsv1), Type (Microsoft.DataLakeStore/account), Subscription (Visual Studio Premium with MSI), Resource Group (sqlbitsadfcw01), and Location (westeurope). The 'Activities' tab shows a list of successful uploads: 'Uploaded Group: Uploaded: 2', 'Uploaded 'Address.json' to sqlbitsadfcw01dlsv1/Reference/Address.json', and 'Uploaded 'CustomerAddress.txt' to sqlbitsadfcw01dlsv1/Reference/CustomerAddress.txt'. The last three items are highlighted with a green border.

Create New Factory Resources

After uploading the files to Azure Data Lake Storage Gen1, you will create two new pipelines, one for each file. To complete this task, you will also need to create new Datasets.

As you have now successfully created Datasets and Pipelines in previous activities, we will mainly provide the settings and values needed in this lab. Some additional information will be provided where instructions differ from previous activities. If you get stuck, review the previous activities, or ask for help from a neighbor or from the instructors.

Create New Pipeline and Dataset for Address.json

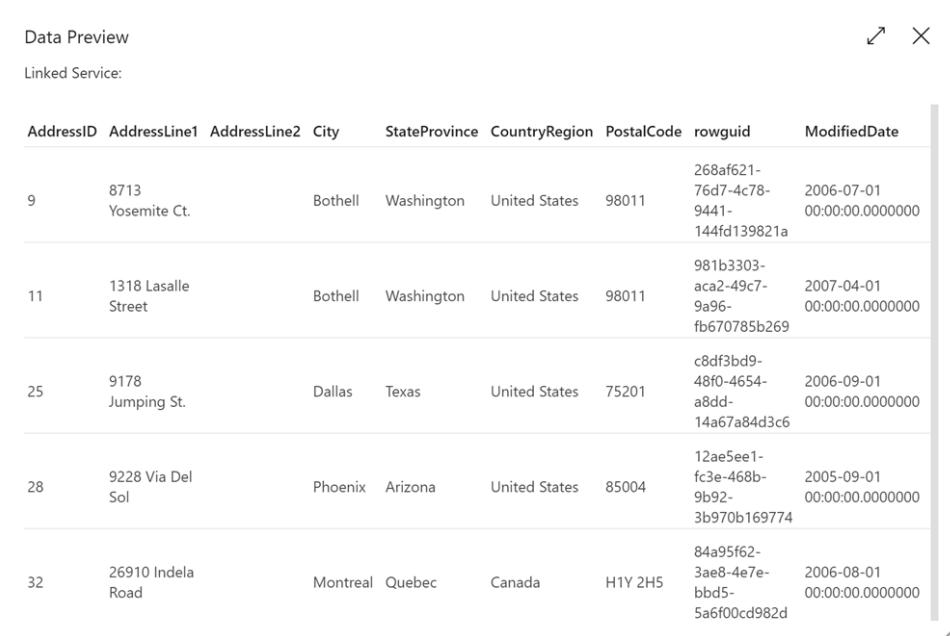
Create a pipeline named **Copy_Address** in the **Lab 02** folder, and add one Copy Data activity named **Copy Address**. In the **Source** tab of the Copy Data activity, click **+ New** and create a dataset with the following properties.

Setting	Value	Notes
Dataset Type	Azure Data Lake Storage Gen1	
Name	DS_ADLS_Address	
Linked	LS_ADLS_sqlbitsadfcw01dlsv1	

Service	
File Path	Reference / Address.json
File Format	JSON Format
Folder	Lab 02

Unless specified above, use the default settings.

Ensure that the **Data Preview** looks ok.



The screenshot shows the 'Data Preview' pane with a green border. At the top left is the title 'Data Preview'. To its right are two small icons: a double-headed arrow and a close (X) button. Below the title, it says 'Linked Service:'. The main area is a table with the following data:

	AddressID	AddressLine1	AddressLine2	City	StateProvince	CountryRegion	PostalCode	rowguid	ModifiedDate
9	8713 Yosemite Ct.			Bothell	Washington	United States	98011	268af621- 76d7-4c78- 9441- 144fd139821a	2006-07-01 00:00:00.0000000
11	1318 Lasalle Street			Bothell	Washington	United States	98011	981b3303- aca2-49c7- 9a96- fb670785b269	2007-04-01 00:00:00.0000000
25	9178 Jumping St.			Dallas	Texas	United States	75201	c8df3bd9- 48f0-4654- a8dd- 14a67a84d3c6	2006-09-01 00:00:00.0000000
28	9228 Via Del Sol			Phoenix	Arizona	United States	85004	12ae5ee1- fc3e-468b- 9b92- 3b970b169774	2005-09-01 00:00:00.0000000
32	26910 Indela Road			Montreal	Quebec	Canada	H1Y 2H5	84a95f62- 3ae8-4e7e- bbd5- 5a6f00cd982d	2006-08-01 00:00:00.0000000

Switch back to the **Copy_Address pipeline**. If the Source dataset refers to a default name like `AzureDataLakeStoreFile1`, click the dropdown and select **DS_ADLS_Address** instead.

Open the **Sink** tab. You will create a new, parameterized dataset to represent a dynamic Azure SQL Database sink that can be reused in other pipelines.

In the **Sink** tab of the Copy Data activity, click **+ New** and choose **Azure Sql Database**. Name the Dataset **DS_ASQL_Staging** and move it to a new **Generic Datasets** folder.

Go to the **Connection** Tab and choose the existing Staging database Linked Service **LS_ASQL_Staging**.

For the **Table** value, check **Edit**, click inside the text box, and then click **Add Dynamic Content [Alt+P]**.

The **Add Dynamic Content** pane will show up on the right hand side of the screen. Click the **+** to the right of the filter text box and create a new parameter.

Setting	Value
Name	TableName
Type	String
Default Value	dbo.TableName

After clicking **Save**, you will be returned to the dynamic content pane. Scroll to the parameters section

at the bottom of the dialog and chose the newly created parameter **TableName**. The expression `@dataset().tableName` should now show up in the first textbox. Click **Finish**.

Verify that the dynamic expression shows up in the Table text box, and that the text box turns blue to indicate that it is using dynamic properties. Switch to the **Parameters** tab to verify that the dataset has the **TableName** parameter.

Switch back to the **Copy_Address pipeline**. If the Sink dataset refers to a default name like `AzureSqlTable1`, click the dropdown and select **DS_ASQL_Staging** instead.

Notice that the Sink dataset now shows the **TableName dataset parameter** under Dataset properties. Next, we want to create a *pipeline parameter* and map it to the *dataset parameter*. This will allow us to pass a table name into the pipeline, which will then be passed into the dataset.

Click inside the **Value** textbox under Dataset Properties and open the Dynamic Content pane. Create a new pipeline parameter.

Setting	Value
Name	SinkTableName
Type	String
Default Value	AWLTSRC.SalesLT_Address

Next, click inside the **Pre-Copy Script** textbox and open the Dynamic Content pane. Create a new pipeline parameter.

Setting	Value
Name	SinkPreCopyScript
Type	String
Default Value	TRUNCATE TABLE AWLTSRC.SalesLT_Address

Once the Pre-Copy Script has been parameterized, open the Pipeline Parameters and verify that the pipeline now has two parameters **SinkTableName** and **SinkPreCopyScript**.

Create New Pipeline and Dataset for CustomerAddress.txt

Follow all the same steps as above, but this time create a **Copy_CustomerAddress** pipeline for the `CustomerAddress.txt` file. Note that this time you can simply reuse the parameterized **DS_ASQL_Staging** Dataset.

Setting	Value	Notes
Dataset Type	Azure Data Lake Storage Gen1	
Name	DS_ADLS_CustomerAddress	
Linked Service	LS_ADLS_sqlbitsadf<999>dlsv1	
File Path	Reference / CustomerAddress.txt	Click Browse to navigate to the file instead of typing in the path manually.

File Format	Text Format	Click Detect Text Format instead of manually choosing the settings.
Folder	Lab 02	<i>Optional, but included as a reminder. Organize your Factory Resources in any way that you prefer.</i>

Setting	Value
Name	SinkTableName
Type	String
Default Value	AWLTSRC.SalesLT_CustomerAddress

Setting	Value
Name	SinkPreCopyScript
Type	String
Default Value	TRUNCATE TABLE AWLTSRC.SalesLT_CustomerAddress

Publish and Debug the New Pipelines

Click **Validate All**. If there are no errors, click **Publish All**.

Trigger both pipelines. Notice the default pipeline properties that will get passed into the generic Sink dataset.

Activity Summary

You have now created two new pipelines that copy a JSON and a Text file from Azure Data Lake Storage Gen1 into Azure SQL Database. We have parameterized the Sink dataset so it can be reused for multiple copy activities.

Activity 03: Creating a Master Pipeline

Activity Overview

Now that we have multiple pipelines, we want to orchestrate them together. We can achieve this by creating a "master" pipeline. (*We will explore this design pattern in more depth in a later lab.*) The master pipeline will first execute the Copy_SalesLT_CustomerV2 pipeline. If that succeeds, it will execute the pipelines Copy_Address and Copy_CustomerAddress. Finally, we will perform some transformations by calling a stored procedure activity to join the data from these datasets and load a final Customer table. This pattern is very similar to the approach you would take with a traditional ELT pipeline in SSIS or another ETL Tool.

Create New Pipeline

Create a new pipeline **Master_Customer** in the **Lab 02** Folder.

Add Execute Pipeline Activity for the Copy_SalesLT_CustomerV2 Pipeline

Click **Activities** -> **General** and drag an **Execute Pipeline** task onto the design surface. Name it **Copy_SalesLT_CustomerV2**.

Click the **Settings** tab. Select the pipeline **Copy_SalesLT_CustomerV2**

Expand **Advanced** and click **Auto-fill parameters**.

The values will get automatically populated with the defaults from the pipeline, but you can modify these or create new ones if needed.

Add Execute Pipeline Activity for the Copy_Address Pipeline

Add a second Execute Pipeline Activity named **Copy_Address** and auto-fill and verify all parameters.

Add Execute Pipeline Activity for the Copy_CustomerAddress Pipeline

Add a third Execute Pipeline Activity named **Copy_CustomerAddress** and auto-fill and verify all parameters.

Add Stored Procedure Activity

Click **Activities** -> **General** and drag a **Stored Procedure** task onto the design surface. Name it **usp_Customer_Load**.

Go to the **SQL Account** tab and select **LS_ASQl_Staging**.

Go to the **Stored Procedure** tab and select **[Stage].[usp_Customer_Load]**.

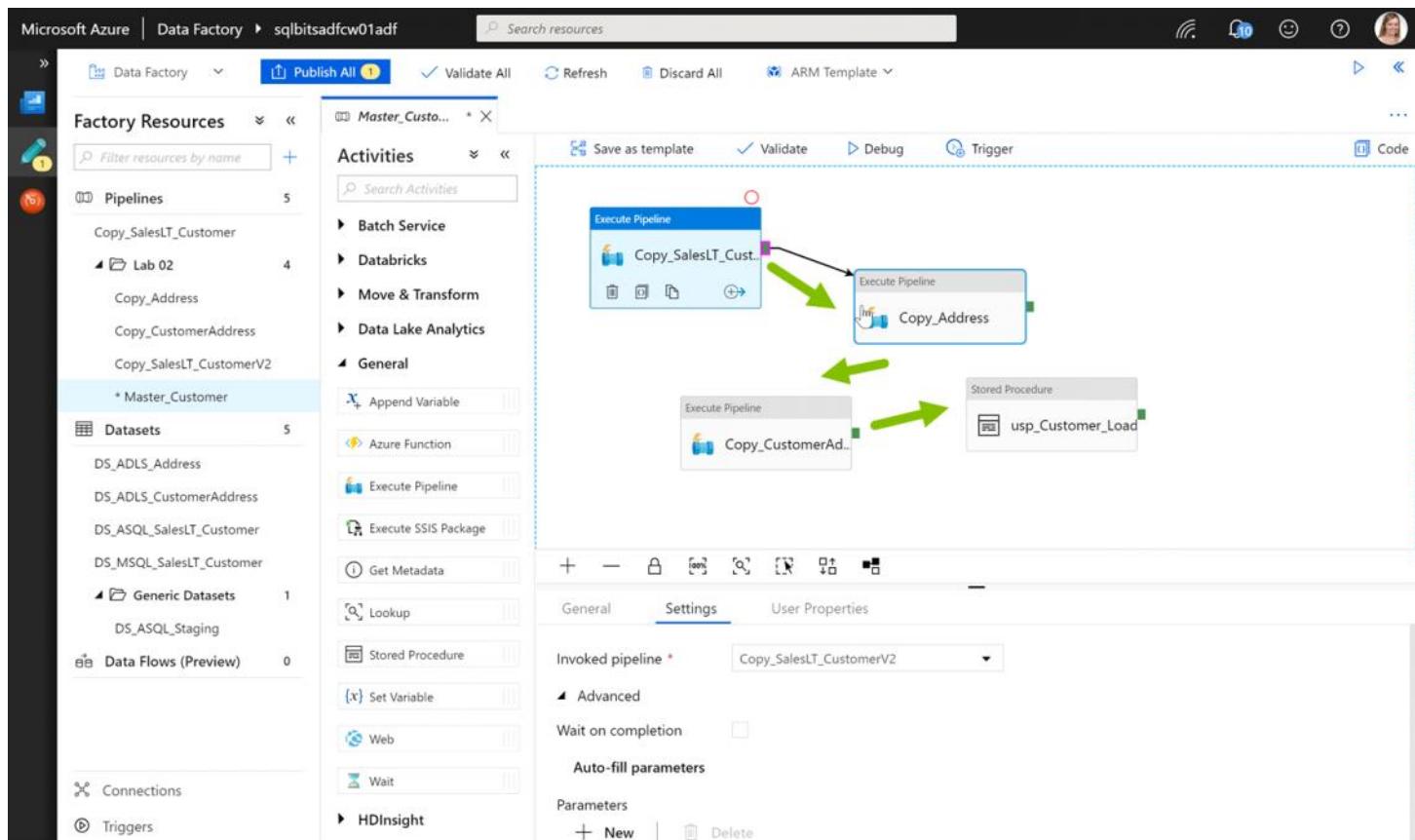
Click the **Import Parameter** button. This will automatically import all of the stored procedure parameters for you. You could also manually create these, but that would be time-consuming and cumbersome.

Set the parameter values for these:

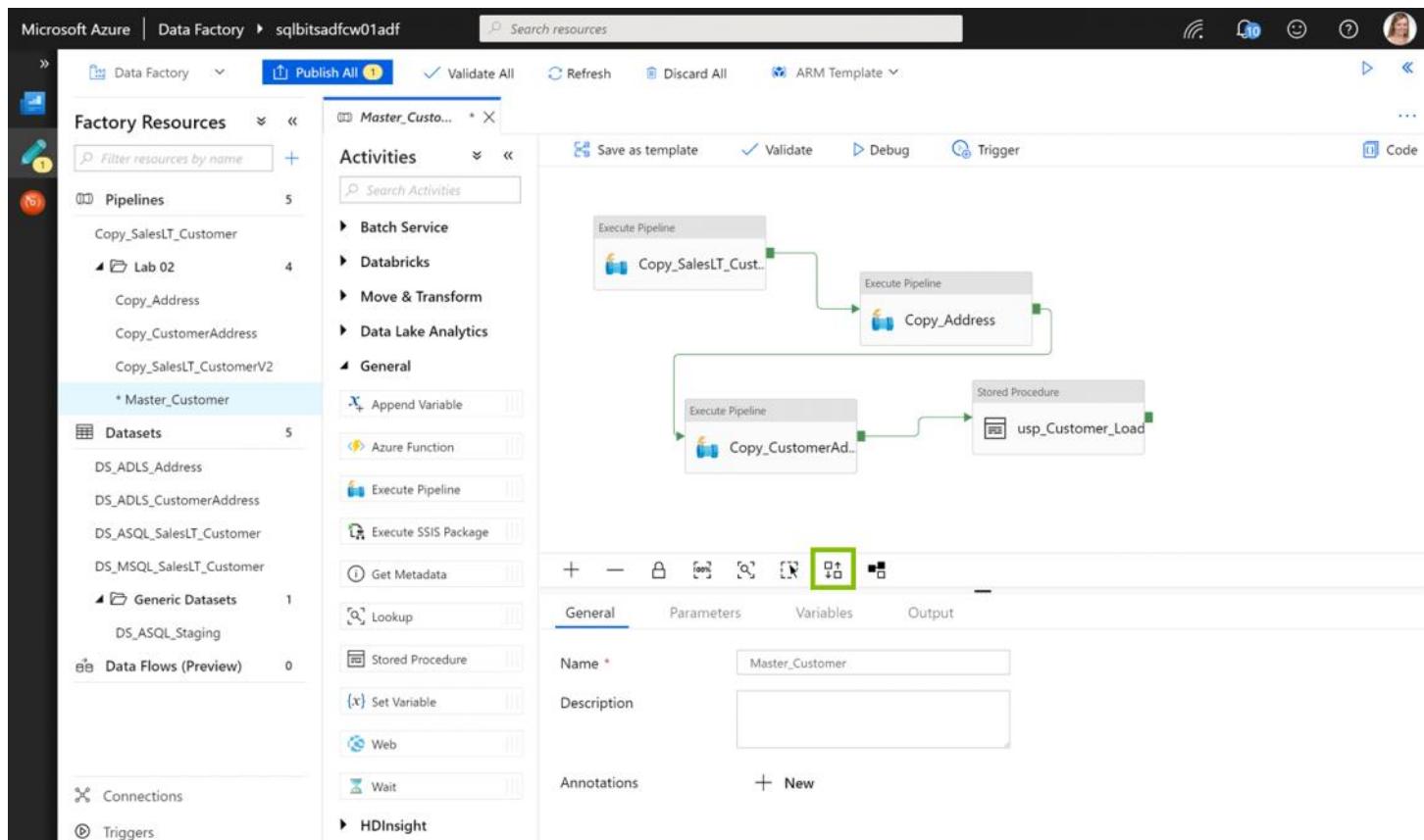
Setting	Value	Notes
BatchId	-1	Enter this as a literal string.
ETLAuditLogId	-1	Enter this as a literal string.
ETLControlId	-1	Enter this as a literal string.
PipelineId	@pipeline().RunId	Add this as Dynamic Content. <i>Hint: Use System Variables in the Add Dynamic Content Pane.</i>
PipelineName	@pipeline().Pipeline	Add this as Dynamic Content. <i>Hint: Use System Variables in the Add Dynamic Content Pane.</i>
PipelineTriggerType	@pipeline().TriggerType	Add this as Dynamic Content. <i>Hint: Use System Variables in the Add Dynamic Content Pane.</i>
SinkName	Stage.Customer	Enter this as a literal string.
SourceName	AWLTSRC.Customer	Enter this as a literal string.
SourceSystem	AWLTSRC	Enter this as a literal string.

Create Precedence Constraints

Add **Success precedence constraints** between each of the Execute Pipeline Activities and the Stored Procedure activity by clicking the green box on each activity and dragging the arrow onto the next task.



Click the Auto-Align button to clean up the design surface.



Screenshot of the Microsoft Azure Data Factory design interface showing a pipeline named "Master_Customer".

The left sidebar lists "Factory Resources" including Pipelines, Datasets, and Activities.

The main area shows a pipeline structure:

```
graph LR; A[Execute Pipeline: Copy_SalesLT_Cust...] --> B[Execute Pipeline: Copy_Address]; B --> C[Execute Pipeline: Copy_CustomerAd...]; C --> D[Stored Procedure: usp_Customer_Load]
```

The "General" tab of the pipeline properties is selected, showing:

- Name: Master_Customer
- Description: (empty)
- Annotations: (empty)

Experiment with the design surface buttons if you can't see all the activities at once.

Screenshot of the Microsoft Azure Data Factory design interface, identical to the previous one but with a green box highlighting the top-left corner of the design surface.

The left sidebar lists "Factory Resources" including Pipelines, Datasets, and Activities.

The main area shows the same pipeline structure:

```
graph LR; A[Execute Pipeline: Copy_SalesLT_Cust...] --> B[Execute Pipeline: Copy_Address]; B --> C[Execute Pipeline: Copy_CustomerAd...]; C --> D[Stored Procedure: usp_Customer_Load]
```

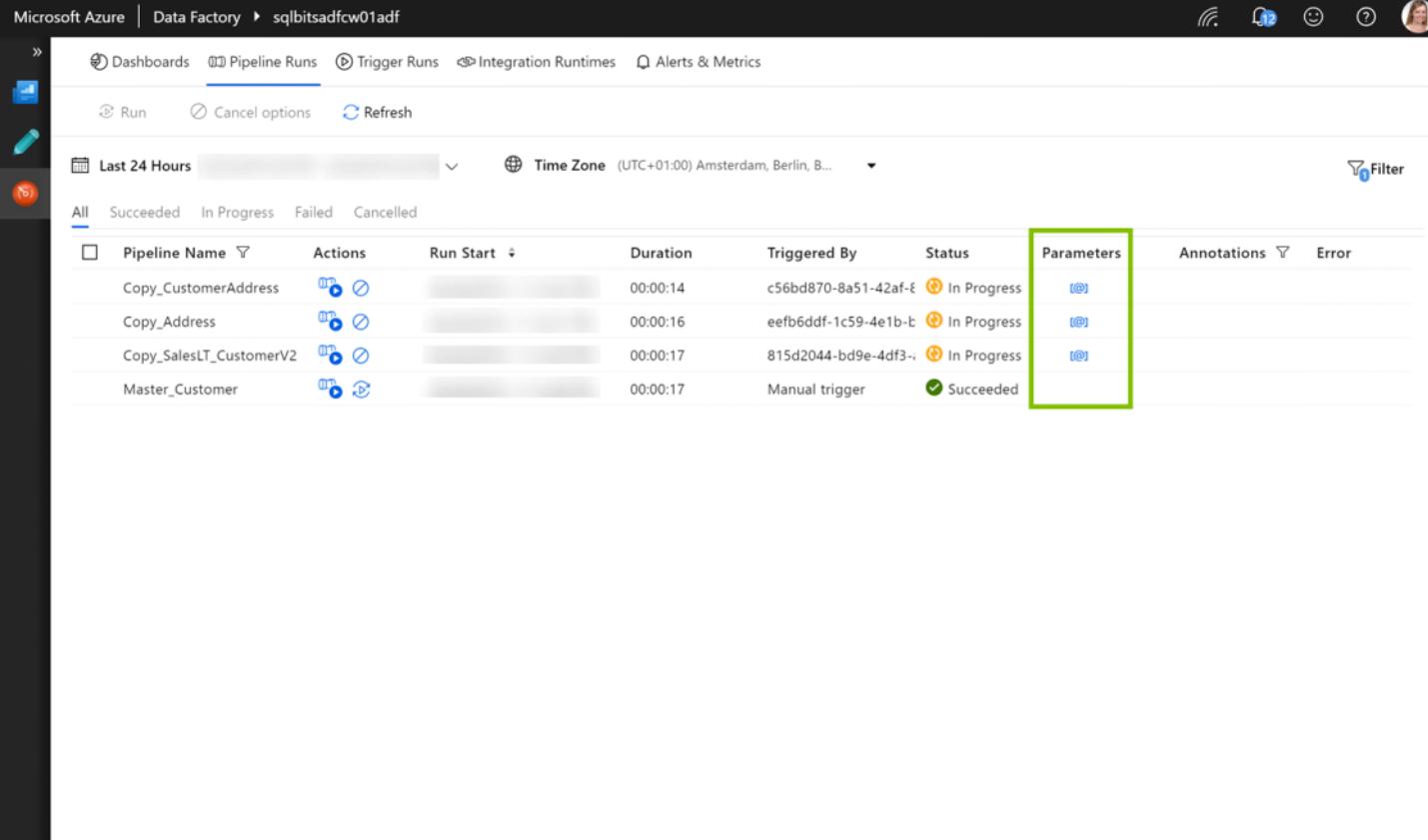
The "General" tab of the pipeline properties is selected, showing:

- Name: Master_Customer
- Description: (empty)
- Annotations: (empty)

Trigger the pipeline

Finally, **Publish All**, then **Trigger** your pipeline to verify that everything is in a working state.

In **Monitor**, click the **Parameters** to see the values used for each activity.



The screenshot shows the Microsoft Azure Data Factory Monitor interface. The top navigation bar includes 'Microsoft Azure', 'Data Factory', and the specific resource name 'sqlbitsadfcw01adf'. Below the navigation are tabs: 'Dashboards', 'Pipeline Runs' (which is selected), 'Trigger Runs', 'Integration Runtimes', and 'Alerts & Metrics'. Under the 'Pipeline Runs' tab, there are buttons for 'Run', 'Cancel options', and 'Refresh'. A time filter dropdown shows 'Last 24 Hours'. The main area displays a table of pipeline runs with the following columns: Pipeline Name, Actions, Run Start, Duration, Triggered By, Status, Parameters, Annotations, and Error. The 'Parameters' column is highlighted with a green box. The table data is as follows:

Pipeline Name	Actions	Run Start	Duration	Triggered By	Status	Parameters	Annotations	Error
Copy_CustomerAddress	[Run] [Stop]	00:00:14	c56bd870-8a51-42af-8 In Progress			[@]		
Copy_Address	[Run] [Stop]	00:00:16	eefb6ddf-1c59-4e1b-b In Progress			[@]		
Copy_SalesLT_CustomerV2	[Run] [Stop]	00:00:17	815d2044-bd9e-4df3-b In Progress			[@]		
Master_Customer	[Run] [Stop]	00:00:17	Manual trigger ✓ Succeeded					

The screenshot shows the Microsoft Azure Data Factory Pipeline Runs page. The top navigation bar includes 'Dashboards', 'Pipeline Runs' (selected), 'Trigger Runs', 'Integration Runtimes', 'Alerts & Metrics', and user profile icons. Below the navigation is a toolbar with 'Run', 'Cancel options', and 'Refresh' buttons. The main area displays pipeline runs for the last 24 hours, filtered by 'Succeeded'. The table columns are: Pipeline Name, Actions, Run Start, Duration, Triggered By, Status, Parameters, Annotations, and Error. Four pipeline runs are listed:

Pipeline Name	Actions	Run Start	Duration	Triggered By	Status	Parameters	Annotations	Error
Copy_CustomerAddress	Run Reschedule Stop	00:00:18	c56bd870-8a51-42af-8 Succeeded					
Copy_Address	Run Reschedule Stop	00:00:18	eefb6ddf-1					
Copy_SalesLT_CustomerV2	Run Reschedule Stop	00:00:18	815d2044-i					
Master_Customer	Run Reschedule Stop	00:00:17	Manual trig					

A callout box highlights the 'Parameters' section for the 'Master_Customer' pipeline run. It shows two parameters:

Name	Value
SinkTableName	AWLTSRC.SalesLT_CustomerAddress
SinkPreCopyScript	TRUNCATE TABLE AWLTSRC.SalesLT_CustomerAddress

Activity Summary

You have now created a master pipeline that performs a basic ELT Routine by loading data from the customer, address and customeraddress datasets. We load data from a relational source and both a JSON and CSV file. We parametrized these connections and pipelines to make them reusable.

Bonus Activity: Parameterize Source Datasets

Activity Overview

In Activity 02, you created a parameterized Sink dataset. If you have spare time in this lab, parameterize the Source datasets reading JSON and Text files so they can be reused for any JSON or Text file.

We have provided some samples json files in the **02 - ADF Basics** folder. See if you can load these to a SQL Table.

What about the Copy_SalesLT_Customer and Copy_SalesLT_CustomerV2 pipelines from Activity 01? What have you learned in this lab that can be implemented to improve these pipelines?

Hints and things to consider:

- Dynamic File Paths
- Dataset Parameters
- Dataset Names
- Folder Names
- Usage of Datasets and Parameters in Pipelines

Lab 04: ADF Design Patterns

Background

In Lab 02, we created some pipelines and demonstrated a rudimentary ELT Pattern, however there is much more needed to build a robust solution. In this lab, we will build upon what we learned in Lab 01 and Lab 02. We will extend the functionality and add some much needed polish.

In this lab, we will take the existing solution and add incremental Load capabilities, we will also make it fully metadata driven so a single master pipeline can drive the extract and load of 1 or many tables from a source system.

Finally we will add the capability to call stored procedures to load a staging area based on our work in Lab 02 Activity 03

At this point you should be getting very familiar with the in's and outs of the Azure Data Factory visual editing environment so we won't always explicitly call out each step in the UI with a screenshot as we did before. But don't worry if you get stuck you can refer back to previous activities, feel free to also ask your lab partner or the instructor for help.

Prerequisites

In order to complete this lab, you will need to have previously created:

- Azure Data Factory From Lab 01
- Copy Activity Pipeline From Lab 01
- Pipelines from Lab 02

Overview

- Add Pipeline Parameters to Copy Activity
 - Parameter 1
 - Parameter 2
- Create New pipeline to copy from file based stored (ADLS Gen1)
- Create a "Master Pipeline"
- Add a stored procedure activity

Let's get started with Activity 1!

In this lab we will take the existing solution and make it fully metadata driven. We will also enable change tracking to allow for incremental load capabilities.

Bulk Tables Transfer with Incremental Load

- 1) Driving the Loop
 - a. Adding Metadata Entries
 - b. Creating stored procedures
- 2) Creating the lookup tasks
- 3) Using the foreach
- 4) Executing the child pipeline

Staged Load - one drawback of the above approach is that the code will perform row by row inserts. In order to maximize performance for large fact table loads and leverage the Sql bulkload we will perform a staged load:

- 1) Create DB Objects
 - a. Staging table
 - b. Post load Sproc
- 2) Create copy activity
 - a. Configure source properties
 - b. Configure pre copy script (Truncate table)
 - c. Configure sink
- 3) Configure post load sproc activity
- 4) Publish and execute pipeline
 - a. Setup a db trace to verify
- 5) Bonus Labs
 - a. Configure Inmemory tables
 - b. Configure temporal tables

Compute execution - you may want to use ADF to orchestrate big data solutions such as hive spark and azure Data lakes analytics or even stored procedures in azure sqldb or sql dw. Often these work flows can be chained together using a simple metadata driven pipeline. The following lab walks us through such a use case:

Streaming processing - it is common to have to ingest data in real time or near realtime From source such as Azure Event hubs, Apache kafka or other message buses. While ADF isn't a great solution for implementing a true real-time solution it can orchestrate micro batch loading

Activity 01: Implement an Incremental Load

Activity Overview (10 minutes)

In the previous lab we used and truncate and load pattern while sometimes this is the only option available. We are going to try and refine the approach by performing an Incremental load. Along the way we will look at an interesting way to add Transformation capabilities in ADF Via SQL Stored procedures this also allows us to get around some mapping issues.

Create a New Folder

Let's kick off this lab by performing an activity you already know how to do creating a new folder: **Lab 03** While we are doing this lets also create a new pipeline by cloning the **Copy_SalesLT_CustomerV2**, changing the name to **Copy_SalesLT_CustomerV3**, and placing that in the new Lab 03 Folder we just created when all of these tasks are complete it should look like this:

Modify the Copy Activity Sink Tab

Now we need to modify the copy Activity Sink tab by making the following changes to support incremental load.

Remove the precopy activity, since we are loading incrementally we won't be needing it.

Set the stored procedure name to **[AWLTSRC].[usp_SalesLT_Customer_Merge]**

Set the Table Type Property

Now we need to also set the Table Type property to a User Defined table type this will be the "Buffer" that Azure Data Factory populates and runs for each set of incoming rows. In this case, the user Defined Table Type has been predefined so we can just enter it: **[AWLTSRC].[SalesLT_CustomerType]**

Import and Wireup the Stored Procedure parameters

Click the **import Parameter** button a list of stored procedure parameters should appear now set the values per the below table:

Setting	Value	Notes
BatchId	-1	Enter this as a literal string
ETLAuditLogId	-1	Enter this as a literal string
ETLControlId	-1	Enter this as a literal string
LogType	Information	Enter this as a literal string
PipelineId	@pipeline().RunId	Enter this as a dynamic Property
PipelineName	@pipeline().Pipeline	Enter this as a dynamic Property
PipelineTriggerType	@pipeline().TriggerType	Enter this as a dynamic Property
SinkName	Stage.Customer	Enter this as a literal string
SourceName	AWLTSRC.Customer	Enter this as a literal string
SourceSystem	AWLTSRC	Enter this as a literal string

Note: the write batch size property is set to 10,000 by default. This means the stored procedure will be called once for every 10,000 rows therefore if there are 100,000 rows in the source table the stored procedure will be executed 10 times.

At this point everything has been wired up now let's test the incremental load.

Testing the incremental Load

First let's delete all of the data in the destination table: **[AWLTSRC].[SalesLT_Customer]**

TSQL

```
TRUNCATE TABLE [AWLTSRC].[SalesLT_Customer];
```

Now debug the pipeline or if you have previously published it run it via a manualy trigger.

Inspect the table results

TSQL

```
SELECT * FROM [AWLTSRC].[SalesLT_Customer];
```

You should have roughly 847 rows returned. Also be sure to note the updatetime and createdatetime

Now rerun the pipeline with no changes and check the row count it should be the same and the rows creatio nand update time should not have changed:

Now on the source Database **AdventureWorksLT** run a query inserting and updating rows (Deletes wont be tracked)

AdventureWorksLT Modification script

```
USE [AdventureWorksLT]
GO

INSERT INTO [SalesLT].[Customer]
SELECT [NameStyle], [Title], [FirstName], [MiddleName], [LastName], [Suffix], [CompanyName],
[SalesPerson], [EmailAddress], [Phone], [PasswordHash], [PasswordSalt], NEWID(), [ModifiedDate]
FROM [SalesLT].[Customer]
WHERE CustomerId < 10
GO

UPDATE [SalesLT].[Customer]
SET
[CompanyName] = [CompanyName] + ':UPD'
WHERE CustomerId < 10
GO
```

Rerun the pipeline and verify the **Staging** database with the below scripts

Staging Verification script

```
-- check for updates, replace with your Datetime  
SELECT *  
FROM [AWLTSRC].[SalesLT_Customer]  
WHERE UpdateDateTime != Createdatetime;  
  
-- check for Inserts, replace with your Datetime  
SELECT *  
FROM [AWLTSRC].[SalesLT_Customer]  
ORDER BY CreateDateTime DESC;
```

Remember to **Publish All** to save.

Activity Summary

You have now have an incremental load process that is working

Activity 02: Bulk Table Transfer

Activity Overview (10 minutes)

In the previous lab we created an incremental load pattern.

We are now going to make this property data driven. via a metadata control table

Execute Database preparation script

In order to get the lab going we need to change the state of some of the table the following script will do that to make sure we are ready.

Create a New Pipeline

create a new pipeline by cloning the **Copy_SalesLT_CustomerV3** and renaming it to: **Copy_SalesLT**

Create a New Parameterized Source Dataset

After cloning the **Copy_SalesLT_CustomerV3** Create a clone of the source dataset

Replace the hard coded table name in the dataset with a new dataset parameter called **tableName**. set the default value to the currently hard coded value.
After you save the changes on the cloned dataset, move the dataset into the **generic datasets** folder.

Create a New Parameterized Sink Dataset

Follow the similar steps from the previous task to create a clone of the sink dataset

Don't forget to set the default value to **Source** and move the dataset to the **generic datasets** folder.

Modify Copy Activity Sink and Source

Modify the copy activity **Source Properties** to match these values:

Setting	Value	Notes
Source dataset		Select the newly created parameterized source dataset from the previous step
TableName (dataset parameter)	@pipeline().parameters.SourceTableName	You will need to add a new pipeline parameter called: SourceTableName set the default value to: [SalesLT].[Customer]
Query	@pipeline().parameters.SourceQuery	You will need to add a new pipeline parameter called: SourceQuery if it doesn't already exist.

Modify the copy activity **Sink Properties** to match these values

Setting	Value	Notes
Sink dataset		Select the newly created parameterized sink dataset from the previous step
TableName (dataset parameter)	Source	You will need to add a new pipeline parameter Called: SinkTableName set the default value to: Source
Stored Procedure Name	@pipeline().parameters.SinkStoredProcedureName	You will need to add a new pipeline parameter Called: SinkStoredProcedure set the default value to: [AWLTSRC].[usp_SalesLT_Customer_Merge] Note: you have to check the edit box to enable this field to be made dynamic
Table type	@pipeline().parameters.SinkTableType	You will need to add a new pipeline parameter Called: SinkTableType set the default value to: [AWLTSRC].[SalesLT_CustomerType]
BatchId	@pipeline().parameters.BatchId	You will need to add a new pipeline parameter Called: BatchId set the default value to: -1 . Consider the data type for this parameter.
ETLAuditLogId	-1	Enter this as a literal string
ETLControlId	-1	Enter this as a literal string
LogType	Information	Enter this as a literal string
PipelineName	@pipeline().Pipeline	Enter this as a dynamic Property
PipelineTriggerType	@pipeline().TriggerType	Enter this as a dynamic Property
SinkName	@pipeline().parameters.SinkTableName	If it doesn't already exist, you will need to add a new pipeline parameter Called: SinkTableName set the default value to:
SourceName	@pipeline().parameters.SourceTableName	If it doesn't already exist, you will need to add a new pipeline parameter Called: SourceTableName set the default value to:
SourceSystem	AWLT	Enter this as a literal string
Write batch size	@pipeline().parameters.SinkWriteBatchSize	You will need to add a new pipeline parameter Called: SinkWriteBatchSize set the default value to: 10000 . Consider the data type for this parameter.

Verify Pipeline Parameters

We wired up a lot of parameters take a look through the screenshot below and verify that they match:

Test the SalesLT Pipeline

Now that we have the pipeline setup lets test it to make sure everything is going according to plan. Publish the pipeline if necessary, correct any errors, then execute the pipeline be either debug or a manual trigger. By default the pipeline should load the **AWLTSRC.SalesLT_Customer** Table (you may want to truncate the table first)

Create a "Master" Pipeline

Create a new Pipeline Called **Copy_SalesLT_Master** move it to Folder **Lab 03**

Add Batch Lookup Parameter

Create a new lookup activity called: **Get BatchId**

Setting	Value	Notes
Linked Service	ds_asql_staging	
TableName	dbo.TableName	Use the default value. This will not be used when we're executing a Stored Procedure.
Stored Procedure	[ETL].[usp_getBatchID]	You need to wireup the inputBatchId parameter by clicking the import parameters and chossing the treat as null option for the parameter

Add Metadata Lookup Activity

Create another lookup activity connected via a success precedence constraint. Call it: **Get Table Metadata**

Configure the activity as below:

Setting	Value	Notes
Linked Service	ds_asql_staging	
TableName	dbo.TableName	Use the default value. This will not be used when we're executing a Stored Procedure.
Query	SELECT * FROM [ETL].[fn_GetTableList_Merge] (NULL,NULL,NULL)	This will return a list of tables from the metadata table

NOTE: Be sure to uncheck first row only so that the activity will return more than one row

Add Foreach Activity

Add a foreach activity called **Load Each Table**

On the settings tab configure the **Items** to be `@activity('Get Table Metadata').output.value` (NOTE: Be sure the name inside the activity matches your)

Note: for the purposes of debugging check the sequential box this ensures that the foreach does not go parallel

On the **Activities** tab configure a execute pipeline activity called: **Merge Load Table** to execute the **Copy_SalesLT** pipeline passing in the following values from the Items

Setting	Value	Notes
SourceReaderQuery	<code>@Item().SourceReaderQuery</code>	This values comes from the output of the Get Table Metadata Lookup activity:
BatchId	<code>@{activity('Get BatchID').output.firstRow.BatchID}</code>	This value comes from the output of the Get Batch Id Lookup activity.
SinkName	Source	This is hard coded because the table type parameter name is consistent across all stored procedures
SourceTableName	<code>@Item().SourceTableName</code>	
SourceSystemName	<code>@Item().SourceSystemName</code>	
SinkWriteBatchSize	<code>@Item().SinkWriteBatchSize</code>	
SinkStoredProcedureName	<code>@item().SinkStoredProcedure</code>	
SinkTableType	<code>@Item().SinkTableType</code>	
SinkPreCopyScript	-1	

Execute the Master Pipeline to test

Now that you have created a parameterized child pipeline and a metadata driven master pipeline, you can execute it if all goes well you should be able to repopulate the tables (you may want to truncate them first to be sure everything is working)

Use the truncate and select scripts we provided in the first activity of this lab if needed.

Activity Summary

You have now have a metadata driven master process that trigger an incremental load. It would be fairly simple to extend this to any of the other tables in the AdventureworkLT Database.

Bonus Activity: Add Product Related Tables

Activity Overview (10 minutes)

In the previous lab we used predefined stored procedures and queries for the Custom Related tables in **AdventureworksLT** if you really want to test your understanding of the pattern you should be able to follow the same pattern for the product related tables:

Table
[SalesLT].[Product]
[SalesLT].[ProductCategory]
[SalesLT].[ProductModel]
[SalesLT].[ProductModelProductDescription]

Keep in mind that for each of the above table you need to create 3 objects(the names below are hard coded to the product table each object name will need to change for each table):

a User Defined TableType - **[AWLTSRC].[SalesLT_ProductType]**

a sink table - **[AWLTSRC].[SalesLT_Product]**

a merge stored procedure - **[AWLTSRC].[sp_SalesLT_Product_Merge]**

Finally you also need to add an entry in the ETL Control table here is an example (For the customer table the values will need to change for the product tables you will need 1 entry per table):

```
INSERT [ETL].[ETLControl_Merge] ([SourceSystem], [SourceDatabase], [SourceSchema], [SourceTable],  
[SinkDatabase], [SinkSchema], [SinkTable], [SinkTableType], [SinkStoredProcedure],  
[SinkPreLoadScript], [SinkWriteBatchSize], [ETLIncrementalDate], [ETLIncrementalID],  
[QueryColumns], [QueryPredicate], [QueryPrimaryKey], [QueryPredicateType], [QueryFrom],  
[LoadType], [TableType], [IsActive], [UpdateDateTime], [UpdateBatchId]) VALUES (N'AWLTSRC',  
N'AdventureWorksLT', N'SalesLT', N'Customer', N'Staging', N'AWLTSRC', N'SalesLT_Customer',  
N'AWLTSRC.SalesLT_CustomerType', N'usp_SalesLT_Customer_Merge', NULL, 10000, NULL, 0,  
N'NameStyle, Title, FirstName, MiddleName, LastName, Suffix, CompanyName, SalesPerson,  
EmailAddress, Phone, PasswordHash, PasswordSalt, rowguid, ModifiedDate', NULL, N'CustomerID',  
NULL, NULL, N'Incremental', N'Stage', 1, CAST(N'2018-09-23T12:47:13.3830000' AS DateTime2), NULL)
```

Bonus Activity: Copy Data incrementally with Change Tracking

Activity Overview

In the previous lab, we used a hash-based delta detection approach to perform a merge load.

Modify our existing pipeline to use change tracking to bulk transfer tables incrementally you will need to make some pretty extensive modification to the existing pipeline and activities to get and set the high watermarks.

We have added some scripts to get you going, and there are also some preexisting objects in the database that should be helpful.

Lab 05: Mapping Data Flows

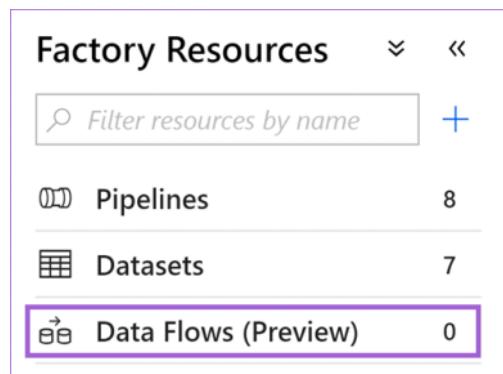
Background and Goals

In this lab, you will get hands-on experience building a Mapping Data Flow, as well as experience working with new Dataset types.

We will create a Pipeline to download public datasets from the Internet Movie Database (<https://imdb.com>) into our Azure Storage Account. Then, we will create a Mapping Data Flow to transform the data and load it into our Azure SQL Data Warehouse.

Prerequisites

You must have already signed up for the Azure Data Factory Mapping Data Flow Preview (<https://aka.ms/dataflowpreview>) and have received confirmation that your subscription has been whitelisted. If you do, you will see **Data Flows (Preview)** under Factory Resources.



If you weren't able to get this done in time for the workshop, team up with another attendee who has.

In addition, you will need to have previously created:

- Resource Group
- Azure Storage Account
- Azure SQL Server
- Azure SQL Data Warehouse

Lab Details

The Internet Movie Database (IMDb) has made several subsets of their data available for personal and non-commercial use. In this lab, we want to get information about movies, their ratings, and the number of votes. We will get this information from the **title.basics** and **title.ratings** files.

Information about the IMDb datasets, including descriptions and details, can be found here:
<https://www.imdb.com/interfaces/>

The actual datasets can be found here: <https://datasets.imdbws.com/>

Source File Format

Each dataset is contained in a gzipped, tab-separated-values (TSV) formatted file in the UTF-8 character set. The first line in each file contains headers that describe what is in each column. A '\N' is used to denote that a particular field is missing or null for that title/name.

Overview

Activity 01: Build Pipeline to Copy Files

- Create Linked Services for HTTP, Azure Blob Storage, and Azure SQL Data Warehouse
- Create Datasets for HTTP, Azure Blob Storage, and Azure SQL Data Warehouse
- Create Pipeline with Parameters and Looping

Activity 02: Create the Mapping Data Flow

- Create Mapping Data Flow

Activity 03: Execute the Mapping Data Flow

- Start Azure SQL Data Warehouse
- Create Destination Table in Azure SQL Data Warehouse
- Create Pipeline to Debug and Execute Mapping Data Flow
- Monitor Mapping Data Flow
- Stop Azure SQL Data Warehouse

Activity 01: Build Pipeline to Copy Files

Activity Overview

Estimated time to complete activity: **10 minutes**.

By now, you should be familiar with creating Linked Services, Datasets, and Pipelines, so we will only provide the high-level details for you.

In this activity, we will create a Pipeline to copy and prepare the files we will use in our Mapping Data Flow. You will get experience with three new Dataset types: **HTTP**, **Parquet**, and **Azure SQL Data Warehouse**.

Create Linked Services

Create three Linked Services. One **HTTP** Linked Service for *getting* the source files, one **Azure Blob Storage** Linked Service for *storing* the source files, and one **Azure SQL Data Warehouse** Linked Service for the transformed output data.

Setting	Value
Type	HTTP
Name	LS_HTTP_IMDb
Base URL	https://datasets.imdbws.com/
Authentication Type	Anonymous

Setting	Value
TYPE	Azure Blob Storage
Name	LS_WASB_IMDb
Authentication Type	Use Account Key
Account Selection Method	From Azure Subscription
Azure Subscription	<Your SQLBits Subscription>
Storage Account Name	sqlbitsadf<999>wasb

Setting	Value
TYPE	Azure SQL Data Warehouse
Name	LS_ASDW_DW
Account Selection Method	From Azure Subscription
Azure Subscription	<Your SQLBits Subscription>
Server Name	Sqlbitsadf<999>sqlserver
Database Name	DW
Authentication Type	SQL Authentication
User Name	Adfadmin
Password	VeryStr0ngPassw0rd!

Create Datasets

Create three Datasets. One **HTTP** Dataset for *getting* the source files, one **Azure Blob Storage** Dataset for *storing* the source files, and one **Azure SQL Data Warehouse** Dataset for the transformed output data.

Setting	Value	Note
Type	HTTP	
Name	DS_HTTP_IMDb	
Linked Service	LS_HTTP_IMDb	
Relative URL	@dataset().FileName	Create and use a Dataset Parameter named FileName with the default value title.basics
Compression Type	GZip	
Compression Level	Fastest	
File Format	Text Format	
Column Delimiter	Tab (\t)	
Row Delimiter	Line Feed (\n)	
Column Names in the First Row	Yes (Checked)	
Quote Character	" (Double Quote)	
NULL Value	\N	

Setting	Value	Note
Type	Azure Blob Storage	
Name	DS_WASB_IMDb	
Linked Service	LS_WASB_IMDb	
File Path: Container	@dataset().FilePathDirectory	Create and use a Dataset Parameter named FilePathDirectory with the default value mappingdataflows/imdb
File Path: File	@dataset().FileName	Create and use a Dataset Parameter named FileName with the default value title.basics
File Format	Parquet	

Setting	Value	Note
Type	Azure SQL Data Warehouse	
Name	DS_ASDW_DW_Movies	
Linked Service	LS_ASDW_DW	
Table	[dbo].[Movies]	Click Edit and type in the name manually

Create the Pipeline

Create a new folder called **Mapping Data Flows**, and add a new pipeline called **PL_IMDb_Copy_HTTP_GZip_to_AdLS_Parquet**.

In previous labs, we have created configuration tables and used the Lookup activity to get the files and tables to work with. Since we want to primarily focus on the Mapping Data Flows feature in this task, we will keep things a little simpler in this pipeline and use an array parameter instead.

Create the following **Pipeline Parameters**:

Name	Type	Default Value
FileNames	Array	["title.basics", "title.ratings"]
SourceFileType	String	.tsv.gz
SinkFilePathDirectory	String	mappingdataflows/imdb
SinkFileType	String	.parquet

Add a **ForEach** task named **Copy HTTP GZip to ADLS Parquet**, and set the **Items** property to the **@pipeline().parameters.FileNames** parameter.

In the ForEach Activities, add a **Copy Data** activity named **HTTP GZip to ADLS Parquet**.

In the Copy Data activity, use the **Source** dataset **DS_HTTP_IMDb**. Set the **FileName** value to **@concat(item(), pipeline().parameters.SourceFileType)**.

In the Copy Data activity, use the **Sink** dataset **DS_WASB_IMDb**. Set the **FilePathDirectory** value to **@pipeline().parameters.SinkFilePathDirectory** and the **FileName** value to **@concat(item(), pipeline().parameters.SinkFileType)**.

Finally, **Validate** and **Publish All**, then **Trigger** the pipeline.

Open **Azure Storage Explorer** and verify that your two new source files have been copied to your Azure Blob Storage.

The screenshot shows the Azure Storage Explorer interface. The left sidebar has a dark theme with icons for upload, download, search, and activities. The main area shows a file structure under 'Active blobs (default)'. The path is 'mappingdataflows > imdb'. A search bar is at the top right. Below it is a table with columns: Name, Size, Access Tier, Last Modified, Blob Type, Content Type, and Status. Two items are listed:

Name	Size	Access Tier	Last Modified	Blob Type	Content Type	Status
title.basics.parquet	215.7 MB	Hot (inferred)		Block Blob	application/octet-stream	Activ
title.ratings.parquet	8.2 MB	Hot (inferred)		Block Blob	application/octet-stream	Activ

At the bottom, it says 'Showing 1 to 2 of 2 cached items'.

Activity Summary

Now that you have your source data, let's build a Data Flow!

Activity 02: Create the Mapping Data Flow

Activity Overview

Estimated time to complete activity: **10 minutes.**

In this activity, we will create a Pipeline to copy and prepare the files we will use in our Mapping Data Flow. You will get experience with two new Dataset types: **HTTP** and **Parquet**. But first, we need to give the ADF_Access Active Directory Group access to the Azure Storage Account.

Create Data Flow

Add a new **Data Flow** named **DF_IMDb_Movies**.

Add Sources

Add **two Sources**, one for each source file. For each source, create a new **Parquet Dataset**.

DS_PARQ_IMDb_Titles:

The screenshot shows the Microsoft Azure Data Factory interface. On the left, the 'Factory Resources' sidebar lists Pipelines (9), Datasets (9), and Data Flows (Preview) (1). The preview section shows a single pipeline named 'DF_IMDb_Movies'. On the right, a 'New Dataset' dialog is open. It has fields for 'Name' (DS_PARQ_IMDb_Titles), 'Linked service' (LS_WASB_IMDb), and 'File path' (mappingdataflows/imdb/title.basics.parquet). Under 'Import schema', the radio button 'From connection/store' is selected. At the bottom right of the dialog are 'Cancel' and 'Finish' buttons.

DS_PARQ_IMDb_Ratings:

New Dataset

Name
DS_PARQ_IMDb_Ratings

Linked service *
LS_WASB_IMDb

Edit Connection

File path *
mappingdataflows / imdb / title.ratings.parquet

Import schema
 From connection/store From local file None

Source Settings Settings Projection Optimization

Output stream name * Ratings

Source dataset * Select...

Options
 Allow schema drift Validate schema

Sampling *
 Enable Disable

Join Sources

Next to the **Titles** source, click the + plus sign. Add a **Join** transformation.

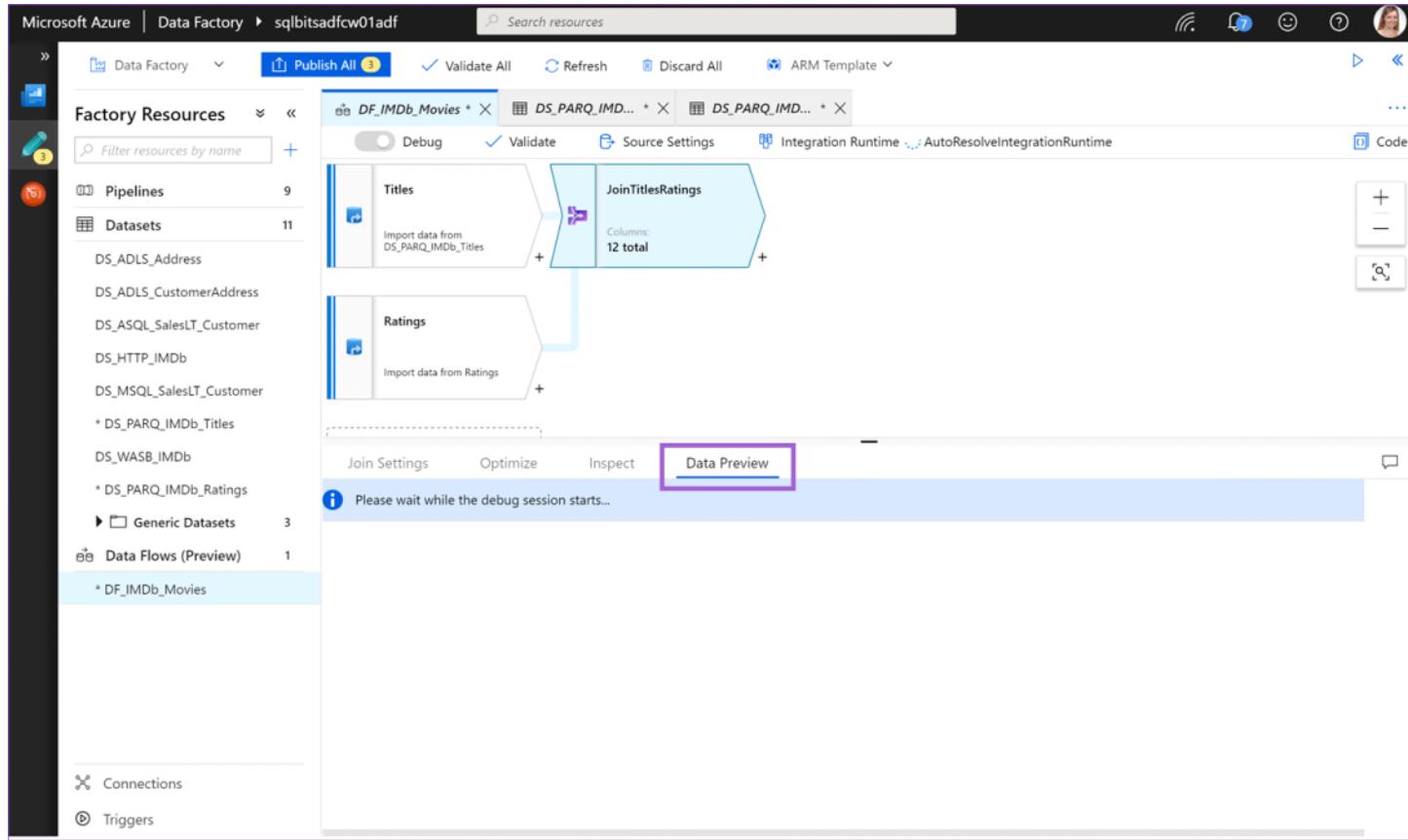
The screenshot shows the Microsoft Azure Data Factory Data Flows interface. On the left, the 'Factory Resources' sidebar lists various datasets and data flows. In the center, two datasets are connected: 'Titles' (9 columns) and 'Ratings' (Import data from Ratings). A purple callout '1' points to the connection line between the datasets. A purple callout '2' points to the 'Join' option in the context menu, which is described as 'Join data from two streams based on a condition'. The 'Source Settings' tab is selected, showing options like Output stream name, Source dataset, Options, and Sampling.

Create an **Inner Join** on the **tconst** columns.

The screenshot shows the Microsoft Azure Data Factory Data Flows interface after the join operation has been configured. The 'Titles' and 'Ratings' datasets are now joined into a single stream named 'JoinTitlesRatings' (12 columns). The 'Join Settings' panel is open, showing the configuration for the join:

- Output stream name:** JoinTitlesRatings
- Left stream:** Titles
- Right stream:** Ratings
- Join type:** Inner (selected)
- Join conditions:**
 - Left: Titles's column: abc tconst
 - Right: Ratings's column: abc tconst

Switch to the **Data Preview** tab and start **Debug**. Wait for the Databricks cluster to get ready. This can take around 5 minutes the first time.



Fetch the latest preview data and verify that you can see both movie titles and their ratings.

Microsoft Azure | Data Factory > sqbitsadfcw01adf

Factory Resources

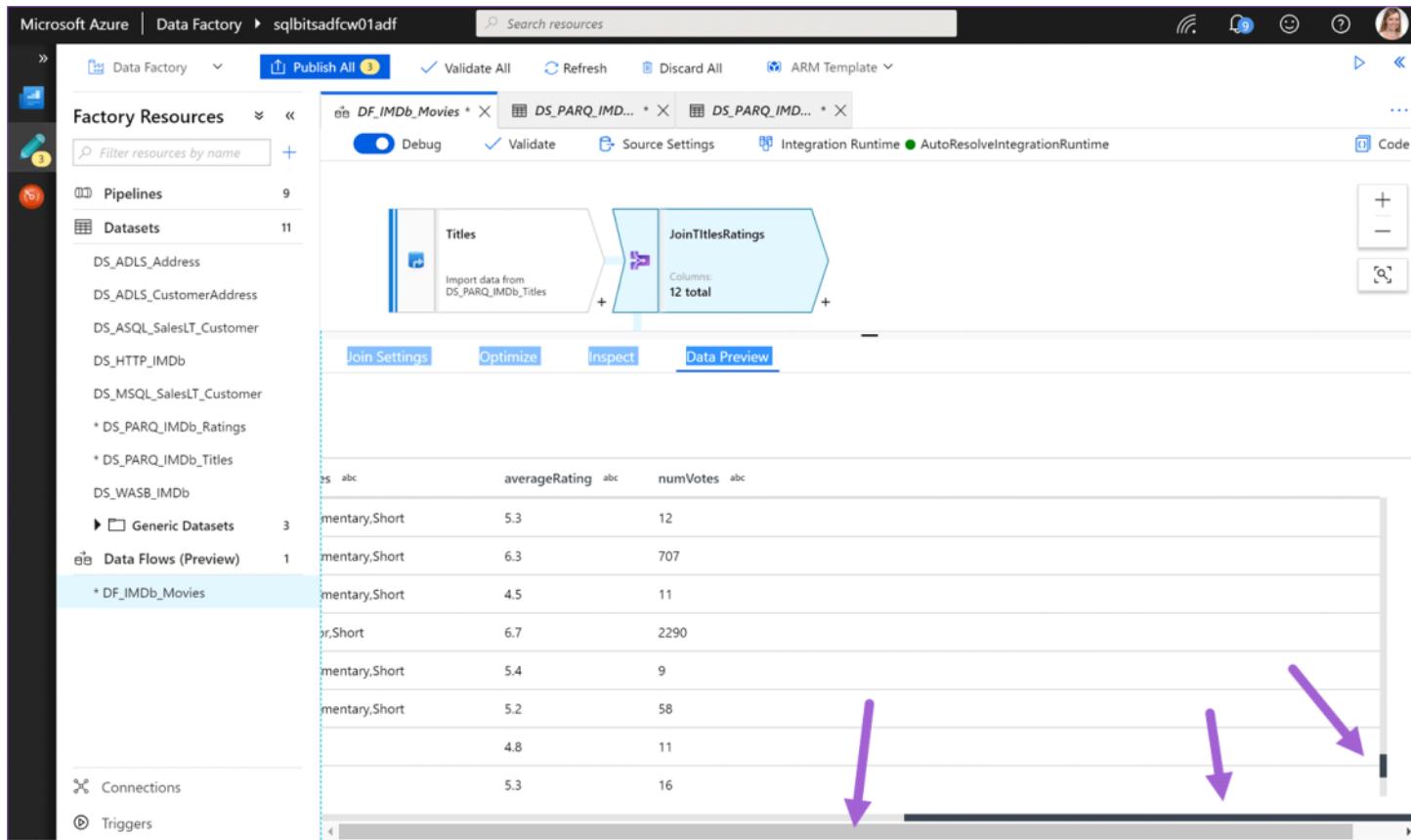
- Pipelines 9
- Datasets 11
- DS_ADLS_Address
- DS_ADLS_CustomerAddress
- DS_ASQL_SalesLT_Customer
- DS_HTTP_IMDb
- DS_MSQl_SalesLT_Customer
- * DS_PARQ_IMDb_Ratings
- * DS_PARQ_IMDb_Titles
- DS_WASB_IMDb
- Generic Datasets 3
- Data Flows (Preview) 1
- * DF_IMDb_Movies

DF_IMDb_Movies * DS_PARQ_IMD... * DS_PARQ_IMD...

Number of rows	Updated*	New*	Unchanged	Total						
tconst	titleType	primaryTitle	originalTitle	isAdult	startYear	endYear	runtimeMinutes	genres	averageRating	numVotes
t10000001	short	Carmencita	Carmencita	0	1894	NULL	1	Documentary,Short	5.8	1467
t10000002	short	Le clown et ses chiens	Le clown et ses chiens	0	1892	NULL	5	Animation,Short	6.4	176
t10000003	short	Fauvre Pierrot	Fauvre Pierrot	0	1892	NULL	4	Animation,Comedy,Romance	6.6	1094
t10000004	short	Un bon bock	Un bon bock	0	1892	NULL	NULL	Animation,Short	6.5	105
t10000005	short	Blacksmith Scene	Blacksmith Scene	0	1893	NULL	1	Comedy,Short	6.2	1800
t10000006	short	Chinese Opium Den	Chinese Opium Den	0	1894	NULL	1	Short	5.6	94
t10000007	short	Corbett and Courtney Be... Edison Kinetoscopic Rec...	Corbett and Courtney Be... Edison Kinetoscopic Rec...	0	1894	NULL	1	Short,Sport	5.5	588
t10000008	short			0	1894	NULL	1	Documentary,Short	5.6	1574

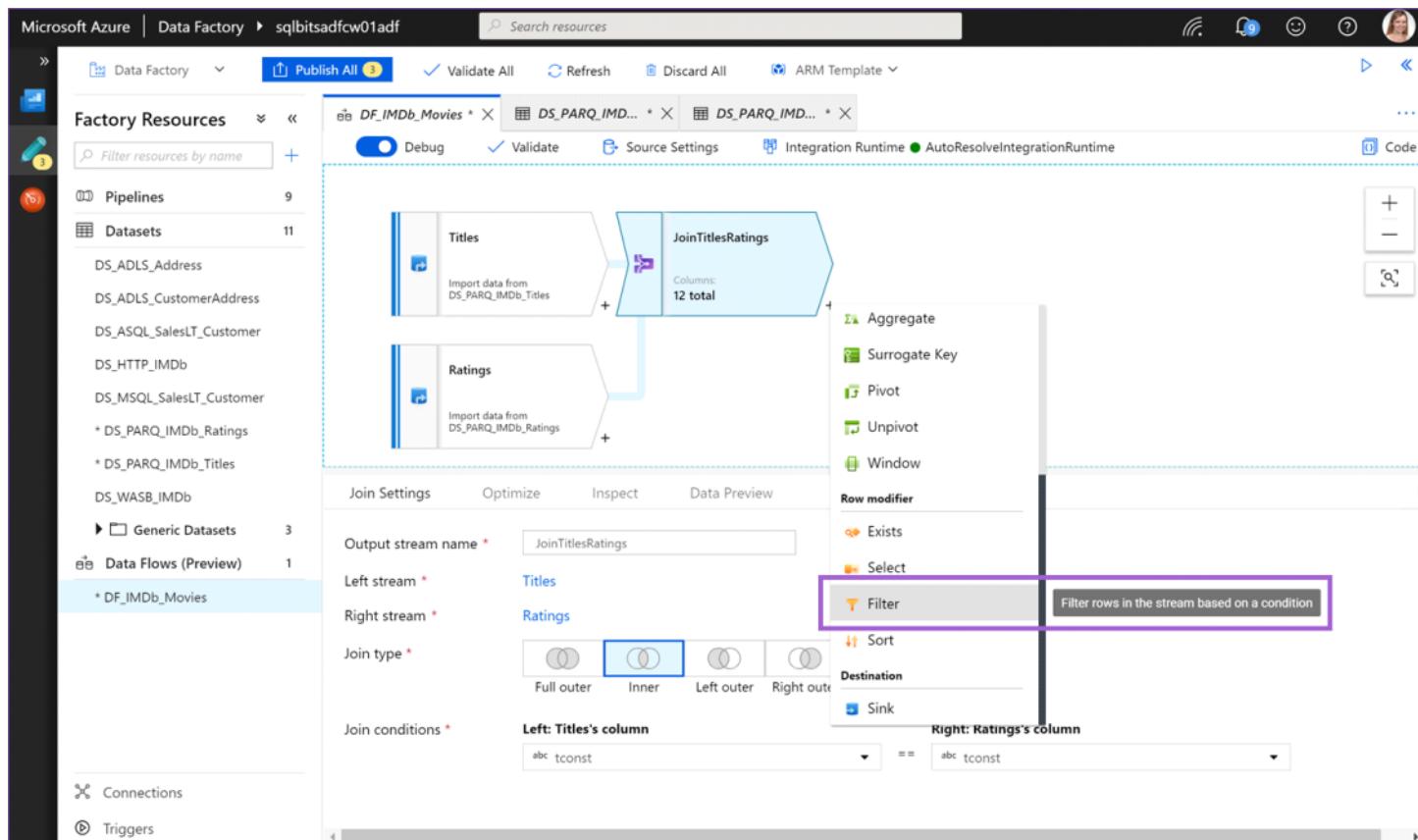
Note: You may run into the "double scrollbars" issue where you have to scroll in both the Data Preview

pane and inside the results grid. This can be rather painful to navigate.

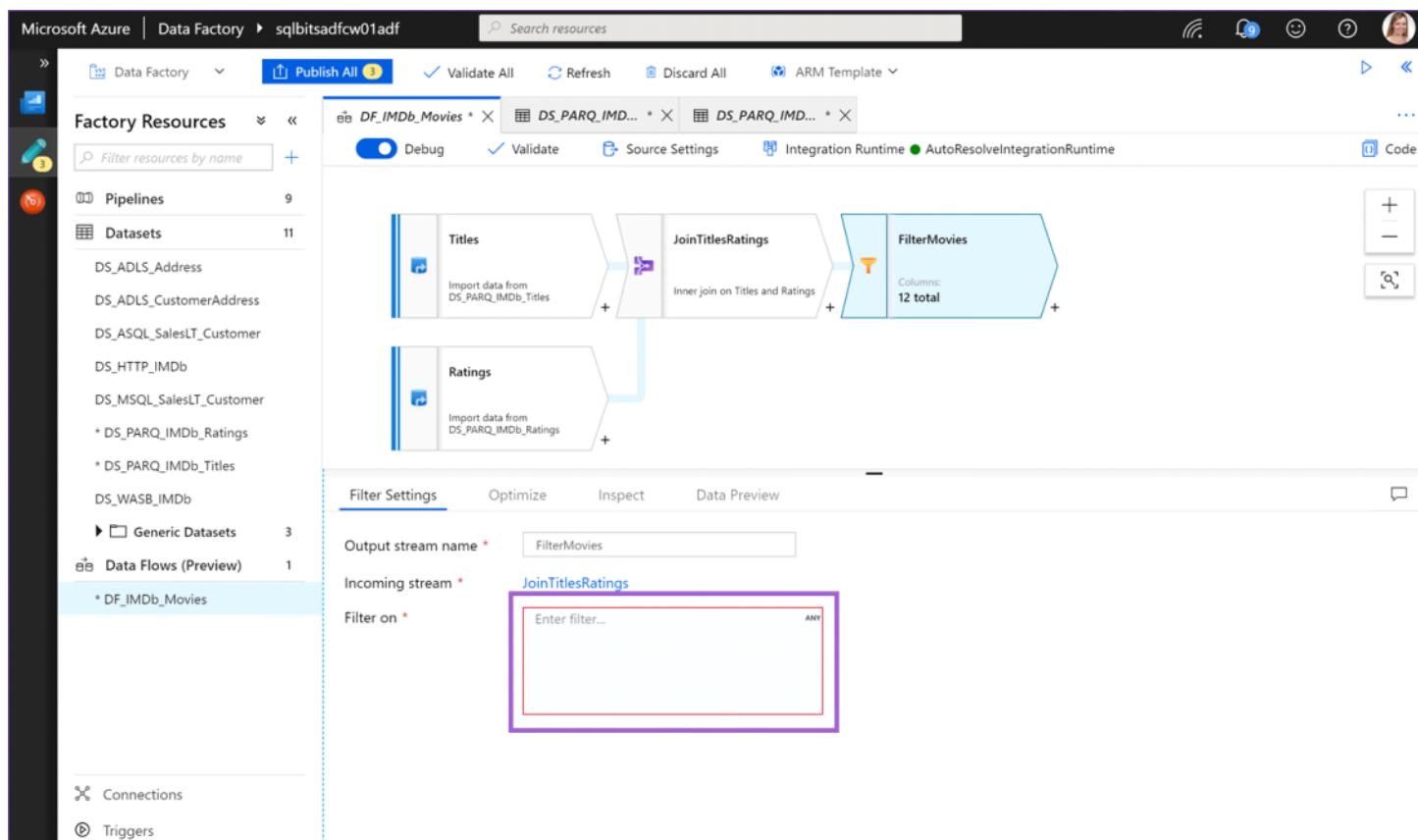


Filter Rows

Next to the **Join** transformation, click the + plus sign. Add a **Filter** transformation.



Open the **Visual Expression Editor** by clicking in the **Filter on** box.



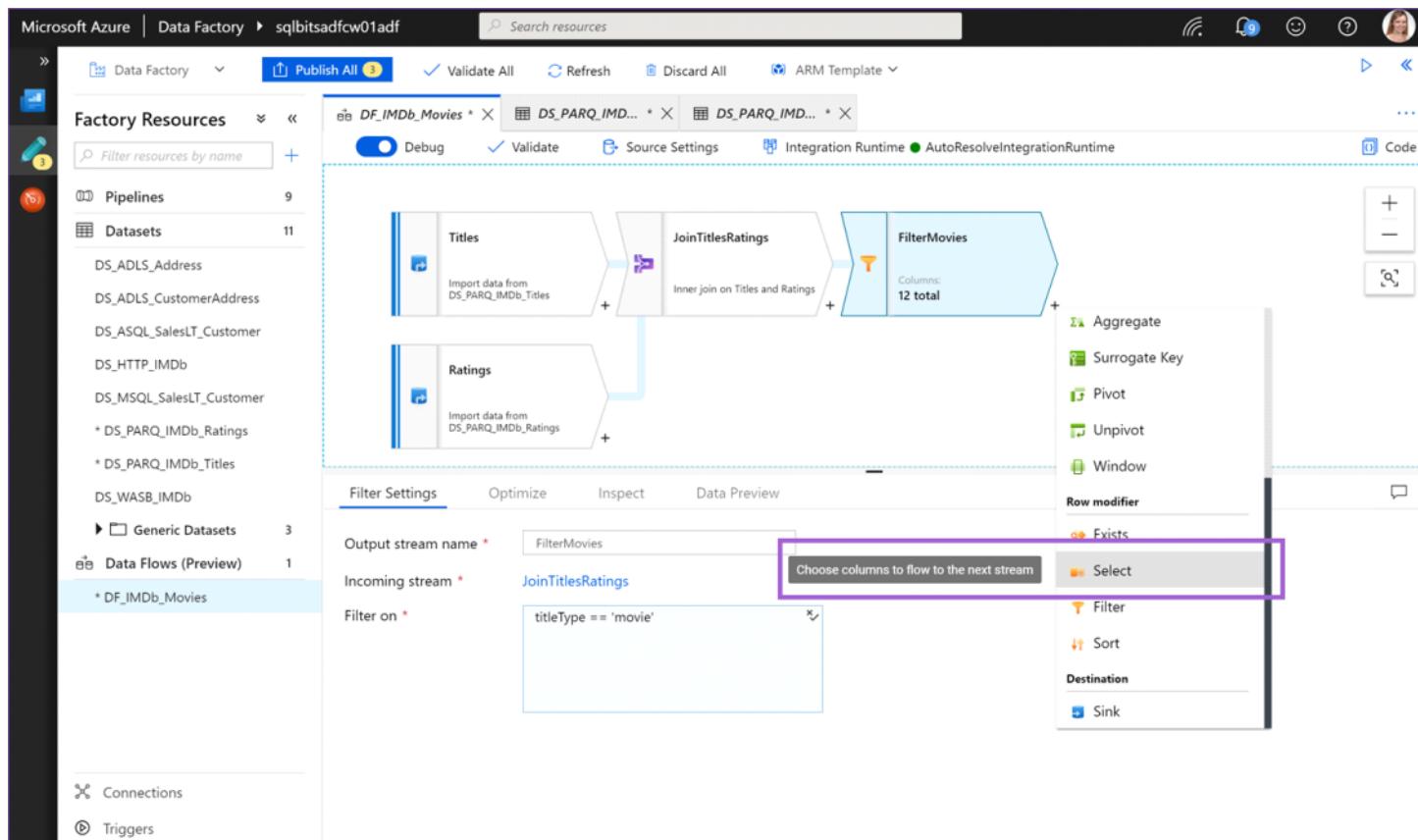
You can click the Input tab to see the input columns, or use the intellisense. Experiment by clicking on the various transformations. Finally, type in the expression `titleType == 'movie'`

Note: These datasets contain the titles of adult movies. If you prefer to remove these, use the following expression instead: `isAdult == '0' && titleType == 'movie'`

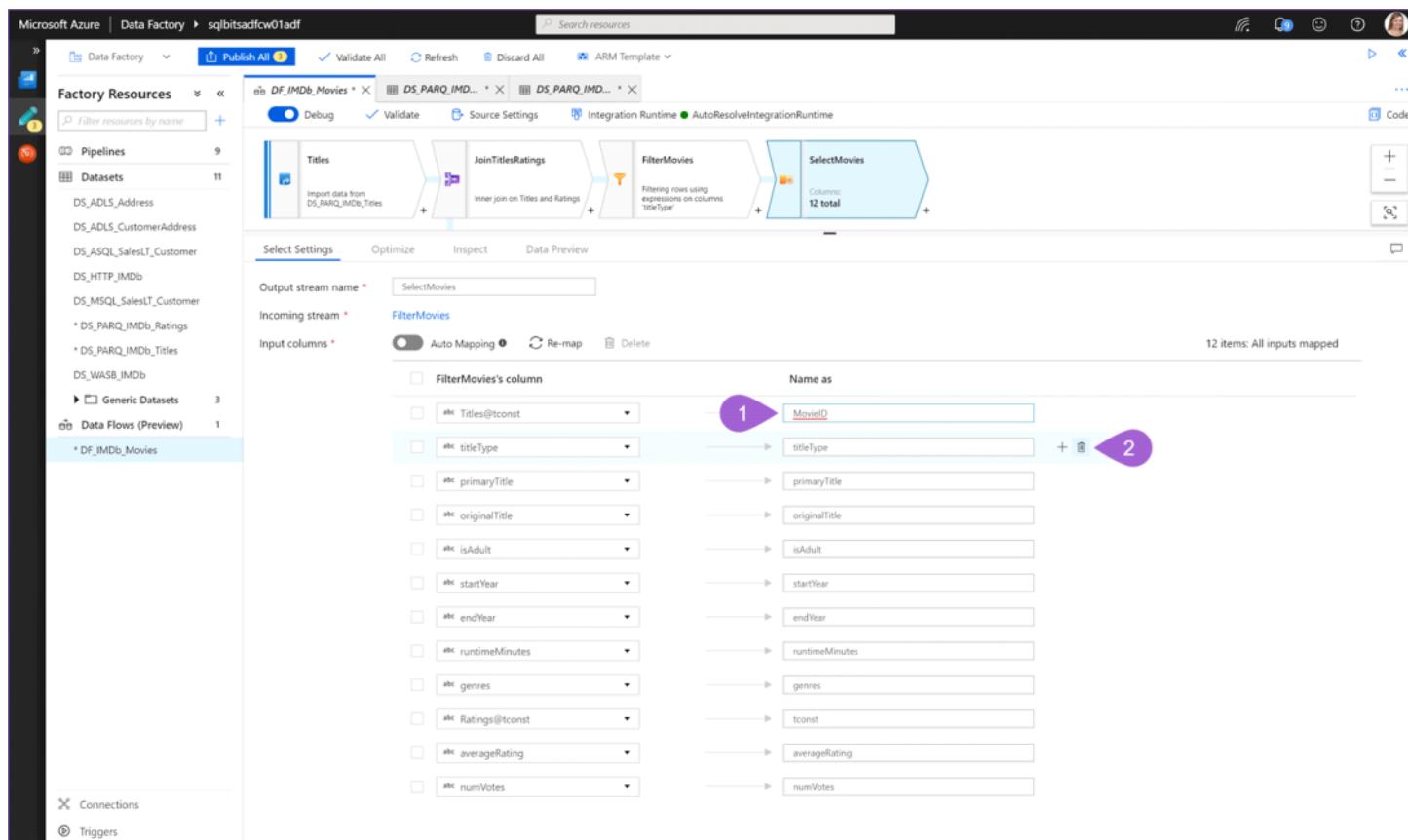
The screenshot shows the Microsoft Azure Data Factory Visual Expression Builder interface. The main area displays a logical expression: `titleType == 'movie'`. This expression is highlighted with a purple rectangular box. Below the expression, there is a list of available variables under the 'Input' tab, including `Titles@tconst`, `titleType`, `primaryTitle`, `originalTitle`, `isAdult`, `startYear`, `endYear`, `runtimeMinutes`, and `genres`. To the right of the expression, there is a toolbar with various operators: +, -, *, /, ||, &&, !, ^, ==, !=, >, <, >=, <=, and []. At the bottom of the builder, there are buttons for 'Cancel', 'Clear Contents', and 'Save and Finish'. A 'Data preview' section shows a table with one row, where the output column contains the value `titleType abc`.

Select and Rename Columns

Next to the **Filter** transformation, click the + plus sign. Add a **Select** transformation.



It can help to zoom out and resize the panes in the interface. Click inside the **Name as** text boxes to rename columns, or hover over and click the icon to delete columns. Notice that when two columns have the same names in the sources, you will get a warning, and they will be prefixed with **Source@Column**.

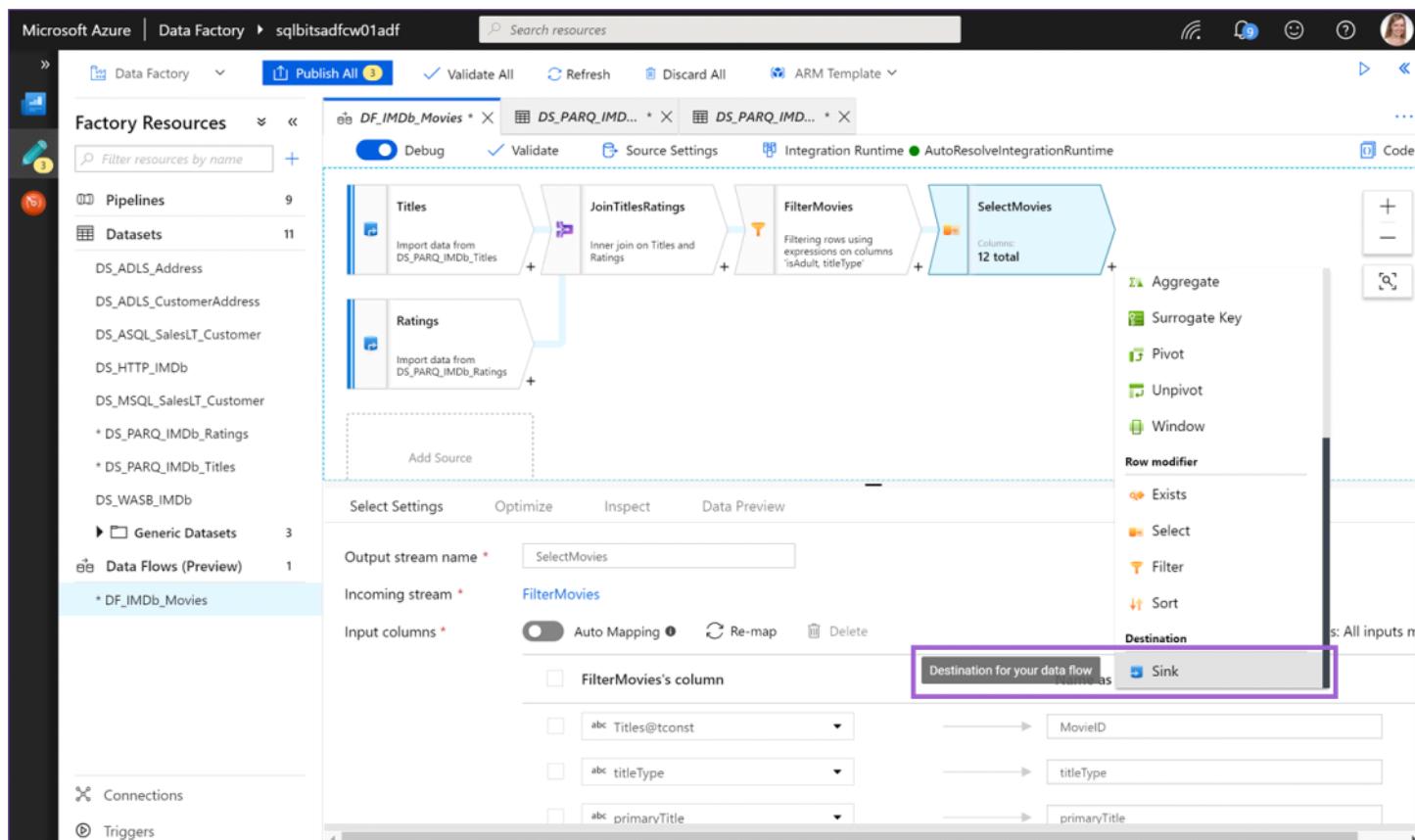


Select and rename the columns.

Column	Name As	Delete
Titles@tconst	MovieID	
titleType		Delete
primaryTitle	Title	
originalTitle	OriginalTitle	
isAdult		Delete
startYear	ReleaseYear	
endYear		Delete
runtimeMinutes	RuntimeMinutes	
genres	Genres	
Ratings@tconst		Delete
averageRating	AverageRating	
numVotes	NumVotes	

Add Sink

Next to the **Select** transformation, click the + plus sign. Add a **Sink** transformation.



For the **Sink dataset**, select the **DS_AS DW_Movies** dataset.

Click **Publish All** to save.

Activity Summary

Now that you have your Data Flow, let's execute it!

Activity 03: Execute the Mapping Data Flow

Activity Overview

Estimated time to complete activity: **5 minutes**.

In this activity, we will create a Pipeline to debug and execute the Data Flow. First, we need to start our Azure SQL Data Warehouse and create our destination table. Then, we will execute the Data Flow and verify that the data has been loaded. Finally, it is important that we stop our Azure SQL Data Warehouse so we don't incur any additional costs.

Start Azure SQL Data Warehouse

From the Azure Portal, **start** your Azure SQL Data Warehouse.

Create Destination Table

Open **SQL Server Management Studio (SSMS)** and connect to your Azure SQL Data Warehouse. Create the **dbo.Movies** table.

```
DROP TABLE IF EXISTS [dbo].[Movies]
GO

CREATE TABLE [dbo].[Movies]
(
    [MovieID] [nvarchar](15) NULL,
    [Title] [nvarchar](200) NULL,
    [OriginalTitle] [nvarchar](200) NULL,
    [ReleaseYear] [smallint] NULL,
    [RuntimeMinutes] [smallint] NULL,
    [Genres] [nvarchar](200) NULL,
    [AverageRating] [decimal](4, 2) NULL,
    [NumVotes] [int] NULL
)
WITH
(
    DISTRIBUTION = HASH ( [ReleaseYear] ),
    CLUSTERED COLUMNSTORE INDEX
)
GO
```

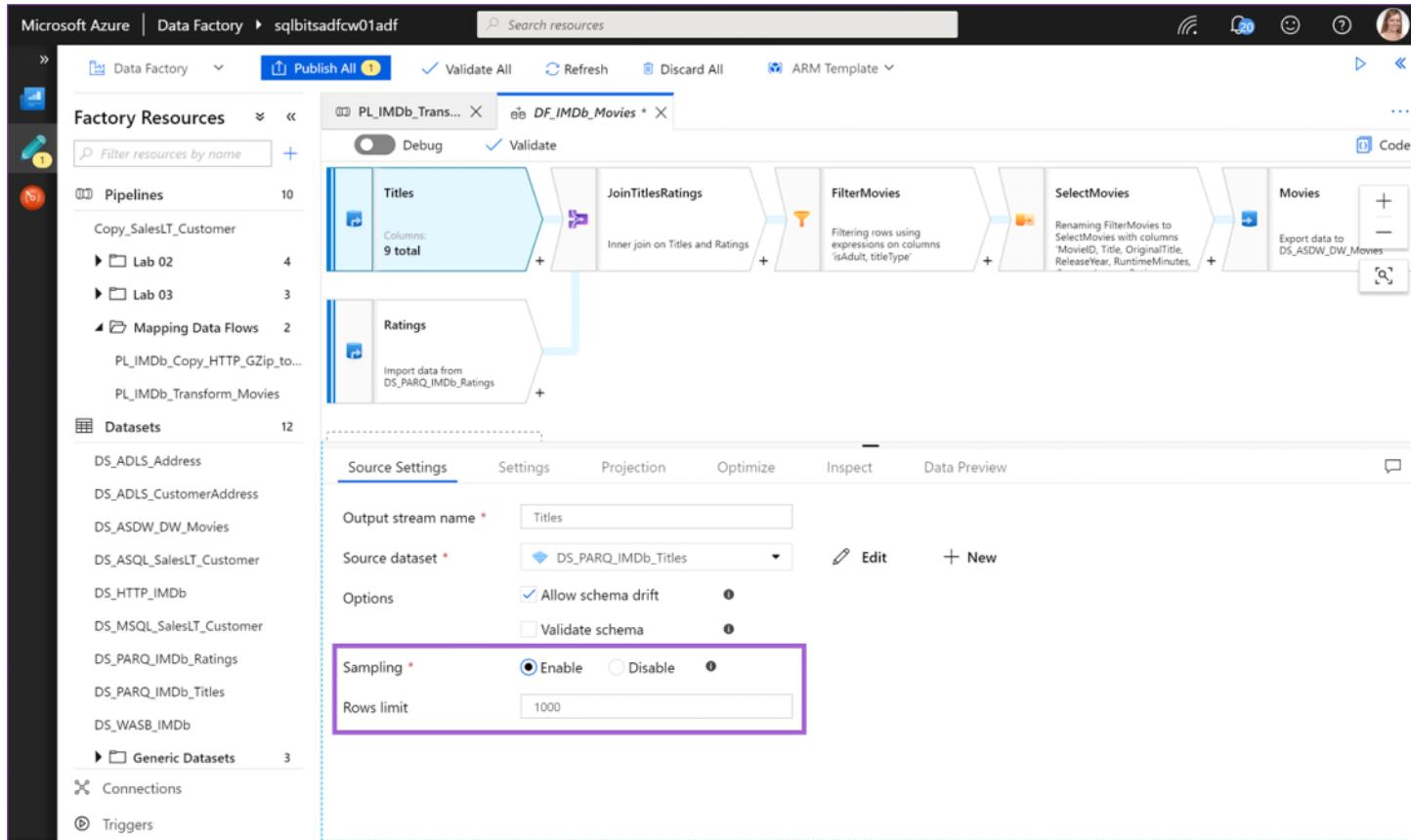
Create Pipeline for Debugging and Executing the Data Flow

To debug a Data Flow, you must run it from a Pipeline. In the **Mapping Data Flows** folder, add a new pipeline called **PL_IMDb_Transform_Movies**.

Add a **Data Flow** activity and choose the **DF_IMDb_Movies** Data Flow. Notice the Settings page. (It is

currently not possible to change the Databricks values.)

Note: To keep the running time to a minimum, we recommend that you enable **Sampling** on the **Titles** source. Set this to 1000 to begin with.



You can always change or remove this later and re-run the pipeline.

Click **Publish All**. Then **Debug** the pipeline.

Review the Execution Plan

Once the Pipeline has executed successfully, review the **Execution Plan** by click on the eyeglasses icon.

Microsoft Azure | Data Factory > sqlbitsadfcw01adf

Search resources

Data Factory Publish All Validate All Refresh Discard All ARM Template

Factory Resources <>

Pipelines 10

- Copy_SalesLT_Customer
 - Lab 02 4
 - Lab 03 3
- Mapping Data Flows 2
 - PL_IMDb_Copy_HTTP_GZip_to...
 - PL_IMDb_Transform_Movies

Datasets 12

Data Flows (Preview) 1

DF_IMDb_Movies

Connections Triggers

Activities <>

Batch Service

Databricks

Move & Transform

Copy Data

Data Flow (Preview)

Data Lake Analytics

General

HDInsight

Iteration & Conditionals

Machine Learning

Data Flow (Preview) Save as template Validate Debug Trigger Code

PL_IMDb_Transform_Movies

Data Flow (Preview) DF_IMDb_Movies

+

General Parameters Variables Output

Pipeline Run ID: 05a6fa7c-0270-40c9-8b8b-da4b706067a1

NAME	TYPE	RUN START	DURATION	STATUS	ACTIONS	RUNID	
DF_IMDb_Movies	ExecuteDataFlow	02/26/2019 7:27 PM	00:06:11	Succeeded			577897c5-3068-49f5

Click around and familiarize yourself with this interface.

Microsoft Azure | Data Factory > sqlbitsadfcw01adf

Search resources

DF_IMDb_Movies Data Flow Number of transforms: 6

Ratings Sink: ●

Titles Sink: ●

JoinTitlesRatings Sink: ●

FilterMovies Sink: ●

SelectMovies Sink: ●

Movies Sink: ●

Ratings Sink: ●

Number of transforms: 6

Movies

TRANSFORM	ROWS	TIME
● Titles	-	-
● SelectMovies	16	
● FilterMovies	16	
● JoinTitlesRatings	570	
● Ratings	922k	3s 671ms

Stop Azure SQL Data Warehouse

From the Azure Portal, **stop** your Azure SQL Data Warehouse.

It is very important to ensure this resource is stopped after this activity is completed, as it will incur costs if left running.

Activity Summary

Congratulations! You have now built and executed your Data Flow.

Bonus Activity: Load Additional Files and Experiment with Transformations

Activity Overview

In this lab, you have worked with just two source files from the available IMDb datasets. If you want to, you can expand on the solution to load the remaining files and experiment with more transformations.

Which other sources and sinks are available in Data Flows? Can you build out a dataset for TV Series in addition to Movies? How can you transform comma-separated columns into rows? How can you get just the top 250 highest-rated movies?

Lab 06: SSIS

In this lab we will cover some of the more advanced features that are needed when the core capabilities of Azure data factory leave you short these are SSIS Lift and shift and Azure Data Flows

- 1) SSIS
- 2) DataFlows

SSIS

you can use SQL Server Data Tools (SSDT) or SQL Server Management Studio (SSMS) to deploy and run SQL Server Integration Services (SSIS) packages in this runtime in Azure.

Provision an Azure-SSIS integration runtime

1. On the Let's get started page, select the Configure SSIS Integration Runtime tile.
2. On the General Settings page of Integration Runtime Setup, complete the following steps:
 - A. For Name, enter the name of your integration runtime.
 - b. For Description, enter the description of your integration runtime.
 - c. For Location, select the location of your integration runtime. Only supported locations are displayed. We recommend that you select the same location of your database server to host SSISDB.
 - d. For Node Size, select the size of node in your integration runtime cluster. Only supported node sizes are displayed. Select a large node size (scale up), if you want to run many compute/memory –intensive packages.
 - e. For Node Number, select the number of nodes in your integration runtime cluster. Only supported node numbers are displayed. Select a large cluster with many nodes (scale out), if you want to run many packages in parallel.
 - f. For Edition/License, select SQL Server edition/license for your integration runtime: Standard or Enterprise. Select Enterprise, if you want to use advanced/premium features on your integration runtime.
 - g. For Save Money, select Azure Hybrid Benefit (AHB) option for your integration runtime: Yes or No. Select Yes, if you want to bring your own SQL Server license with Software Assurance to benefit from cost savings with hybrid use.
 - h. Click Next.

3. On the SQL Settings page, complete the following steps:

For Subscription, select the Azure subscription that has your database server to host SSISDB.

- b. For Location, select the location of your database server to host SSISDB. We recommend that you select

the same location of your integration runtime.

- c. For Catalog Database Server Endpoint, select the endpoint of your database server to host SSISDB. Based on the selected database server, SSISDB can be created on your behalf as a standalone database, part of an elastic pool, or in a Managed Instance and accessible in public network or by joining a virtual

network.

For guidance in choosing the type of database server to host SSISDB, see Compare SQL Database logical server and Managed Instance. If you select Azure SQL Database with virtual network service endpoints/Managed Instance to host SSISDB or require access to on-premises data, you need to join your

Azure-SSIS IR to a virtual network. See Create Azure-SSIS IR in a virtual network.

d. On Use AAD authentication... checkbox, select the authentication method for your database server to host SSISDB: SQL or Azure Active Directory (AAD) with the managed identity for your Azure Data Factory (ADF). If you check it, you need to add the managed identity for your ADF into an AAD group with access permissions to the database server, see Create Azure-SSIS IR with AAD authentication.

e. For Admin Username, enter SQL authentication username for your database server to host SSISDB.

f. For Admin Password, enter SQL authentication password for your database server to host SSISDB.

g. For Catalog Database Service Tier, select the service tier for your database server to host SSISDB: Basic/Standard/Premium tier or elastic pool name.

h. Click Test Connection and if successful, click Next.

4. On the Advanced Settings page, complete the following steps:

a. For Maximum Parallel Executions Per Node, select the maximum number of packages to execute concurrently per node in your integration runtime cluster. Only supported package numbers are displayed.

Select a low number, if you want to use more than one cores to run a single large/heavy-weight package that is compute/memory -intensive. Select a high number, if you want to run one or more small/light-weight

packages in a single core.

b. For Custom Setup Container SAS URI, optionally enter Shared Access Signature (SAS) Uniform Resource Identifier (URI) of your Azure Storage Blob container where your setup script and its associated files are stored, see Custom setup for Azure-SSIS IR.

c. On Select a VNet... checkbox, select whether you want to join your integration runtime to a virtual network. You should check it if you use Azure SQL Database with virtual network service endpoints/Managed Instance to host SSISDB or require access to on-premises data, see Create Azure-SSIS

IR in a virtual network.

Create an Azure-SSIS integration runtime

5. Click Finish to start the creation of your integration runtime.

6. On the Connections tab, switch to Integration Runtimes if needed. Select Refresh to refresh the status.

Use the links in the Actions column to stop/start, edit, or delete the integration runtime. Use the last link to

view JSON code for the integration runtime. The edit and delete buttons are enabled only when the IR is stopped.

[Create an Azure-SSIS integration runtime](#)

[Deploy SSIS package](#)

Now, use SQL Server Data Tools (SSDT) or SQL Server Management Studio (SSMS) to deploy your SSIS packages to Azure. Connect to your Azure SQL Database server that hosts the SSIS Catalog (SSISDB database).

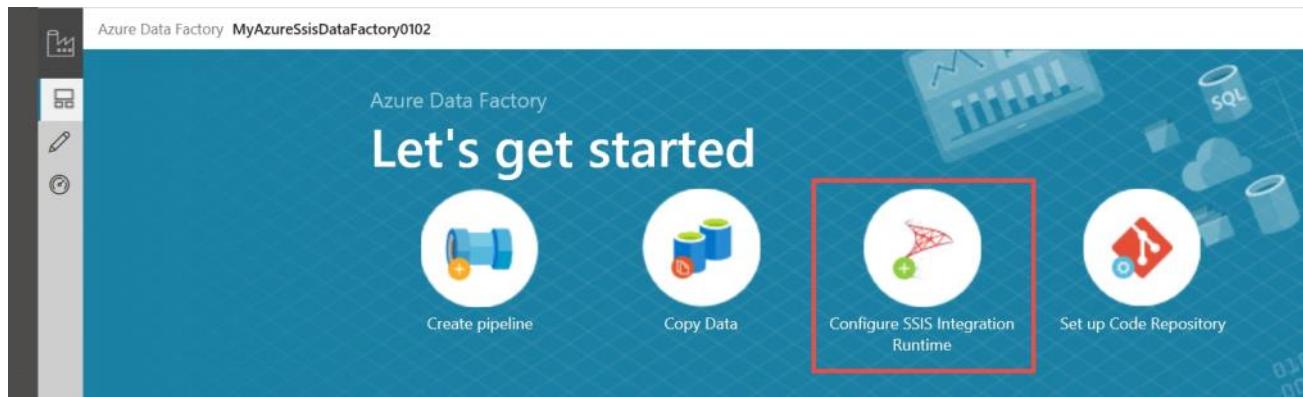
The name of Azure SQL Database server is in the format <servername>.database.windows.net .

Activity 01: Create SSIS Integration runtime

Activity Overview (5 minutes)

Configure SSIS Integration Runtime

In the get started page, click Configure SSIS Integration Runtime tile.



On the General Settings page of Integration Runtime Setup, complete the following steps:

Setting	Value	Notes
Name	SSISIntegrationRuntime	
Location	UK South	Choose UK South Unless you picked a different region when you setup the labs in that case use that region. For example some may have chosen West Europe
Node Size	Standard_D2_v3	Chose the smallest size for demo purposes.
Node Number	1	Note: you can use the slider or the dropdown on the far right.
Edition/License	Enterprise	Note we are choosing enterprise but standard would work this gives you access to the all of the SSIS features that are only available in enterprise edition
Save Money		Choosing this options will reduce the cost but assume you have a valid SQL Server License. Make the best choice for your situation

After configuring your screen should like similar to below:

Integration Runtime Setup

X

General Settings

Name *

SSISIntegrationRuntime

i

Description

i

Type

Azure-SSIS

Location *

UK South

v

Node Size *

Standard_D2_v3 (2 Core(s), 8192 MB)

v

Node Number *

1

Edition/License *

Enterprise

v

Save Money

Save with a license you already own. Already have a SQL Server license?

By selecting "yes", I confirm I have a SQL Server license with Software Assurance to apply this [Azure Hybrid Benefit for SQL Server](#).

Please be aware that the cost estimate for running your Azure-SSIS Integration Runtime is **(1 * US\$ 0.29)/hour = US\$ 0.29/hour**, see [here](#) for current prices.

Now Click Next

On the SQL Settings page, complete the following steps:

Setting	Value	Notes
Subscription	Leave alone	the subscription information should be auto populated
Location	UK South	Choose UK South Unless you picked a different region when you setup the labs in that case use that region. For example some may have chosen West Europe
Catalog Database Server Endpoint		This value should be auto populated
Use AAD authentication		Optional you can set this up but you will have to follow the directions in the link it involves running a PowerShell script. Note: if you check this the other options go away
Admin Username	adfdadmin	This will be the account you chose for the Logical SQL Server during the prereq

		setup tasks
Admin Password		This will be the password you chose for the Logical SQL Server during the prereq setup tasks
Catalog Database Service Tier	S0	

When you are done the screen should look similar to this (be sure to click Test Connection before moving on)

Integration Runtime Setup

SQL Settings

Subscription *

Microsoft Azure Sponsorship (afb8f6f6-a2c0-4a32-ad1d-23054d106390)

Location

UK South

Catalog Database Server Endpoint *

sqlbitsadf01sqlserver.database.windows.net

Use AAD authentication with the managed identity for your ADF ▲
(See how to enable it [here](#))

Catalog Database Service Tier *

S0

Allow Azure services to access ⓘ

✓ Connection successful

Cancel ← Previous Next →

Now click **Next**

On the **Advanced Settings** page, take the defaults

Setting	Value	Notes
Custom Setup Container SAS URI	Leave blank	this property can be used to link to custom setup binaries such as MSI's and Exe's. We won't be using it here but if you want more information: https://docs.microsoft.com/en-us/azure/data-factory/how-to-configure-azure-ssis-ir-custom-setup
Select a VNet for your Azure-SSIS Integration Runtime to join and allow ADF to create certain network resources	unchecked	This property can be used if you need to setup a VNET for advanced scenarios (for example accessing an on-prem data source from within SSIS) if you would like to learn more: https://docs.microsoft.com/en-us/azure/data-factory/join-azure-ssis-integration-runtime-virtual-network

Click **Next** review the settings and then chose **Finish**

Note: it will take 30-40 minutes for the IR to show running periodically check the status. It is critical not to leave it running after the initial deploy as it will incur costs be sure to shut it down when it is done provisioning

In the Connections window, switch to Integration Runtimes if needed. Click Refresh to refresh the status.

Name	Type	Status	Region
defaultIntegrationRuntime	Azure	Running	Auto Resolve
SPAzureSSisIR	Azure-SSIS	Running	East US

Use the links under Actions column to stop/start, edit, or delete the integration runtime. Use the last link to view JSON code for the integration runtime. The edit and delete buttons are enabled only when the IR is stopped.

Name	Type	Status	Region
defaultIntegrationRuntime	Azure	Running	Auto Resolve
SPAzureSSisIR	Azure-SSIS	Running	East US

Note: you could have directly set this up from the Connections Integration Runtime Tab of the visual Authoring U/I.

Summary

In this activity you learned how to setup an Azure SSIS integration runtime. Be sure to stop the integration runtime

Activity 02: Deploy SSIS Project

Activity Overview (5 minutes)

Now, use SQL Server Data Tools (SSDT) or SQL Server Management Studio (SSMS) to deploy your SSIS packages to Azure. Connect to your Azure SQL Database server that hosts the SSIS Catalog (SSISDB database).

The name of Azure SQL Database server is in the format <servername>.database.windows.net .

If you get stuck, see the following articles from the SSIS documentation:

Deploy, run, and monitor an SSIS package on Azure
Connect to the SSIS Catalog on Azure
Schedule package execution on Azure
Connect to on-premises data sources with Windows authentication

Next, we will connect to the Integration Services Catalog to upload the provided package.

Run security scripts

We need to run the following two scripts to ensure that our deployment account has the correct permissions to run the SSIS Package

Master - DBManager Role.sql
SSISDB - DeploymentUser Security.sql

These are located in the folder: **Labs\06 - SSIS Lift and shift**

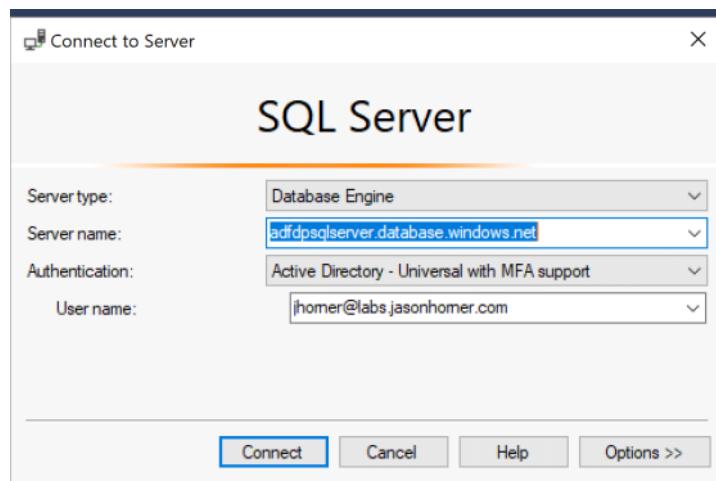
After these scripts run, move on to the next step

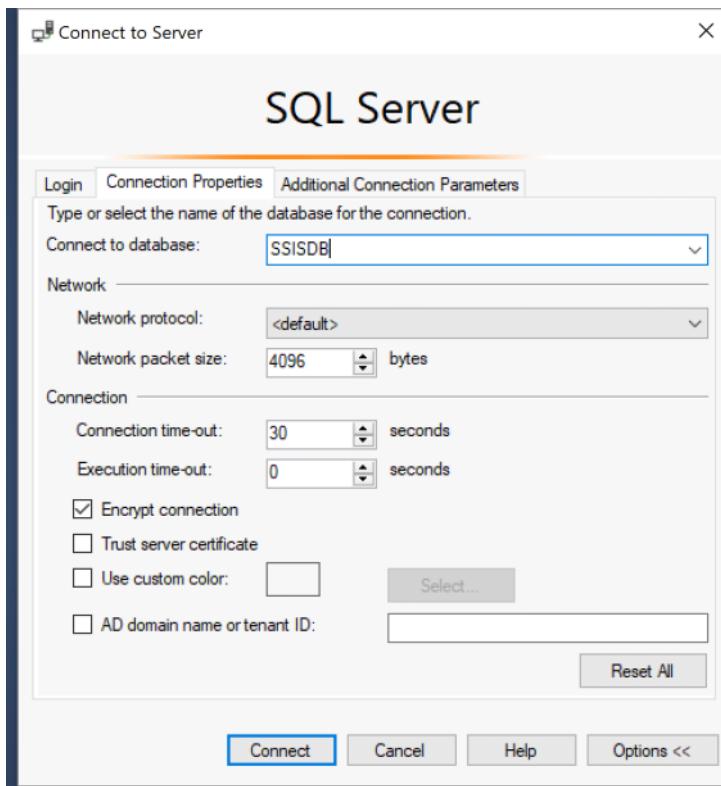
From SQL Server Management Studio, connect to the Integration Services Catalog.

Deploy SSIS Project

- 1) From SQL Server Management Studio, connect to the Integration Services Catalog.

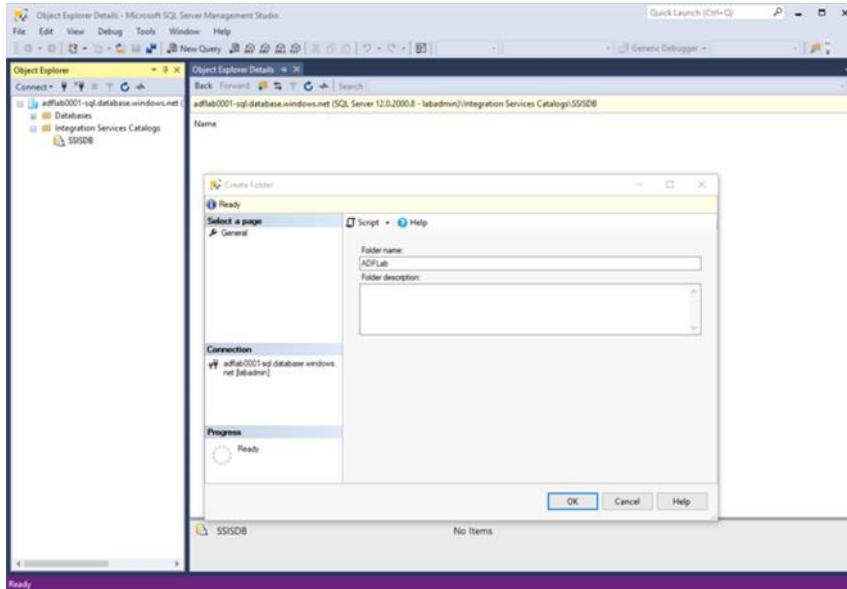
Item	Value	Notes
SQL Server	<Logical SQL Server>.database.windows.net	This will be the name of the azure data factory. It must be unique. Save this value for later as you will refer to it often
Login	DeploymentUser	We use the sql account here, as we experienced problems with the AAD Account
Password	<Your Password>	This is the password that you set
Authentication		
Database	SSISDB	



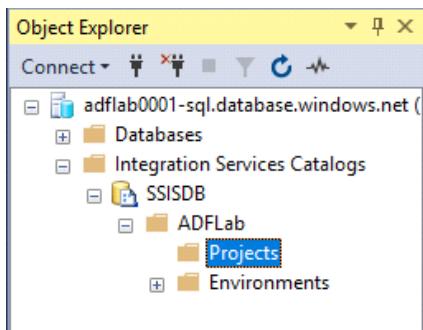


NOTE: It is critical that you set the default database to SSISDB, if you don't you won't get the integration catalog folder below

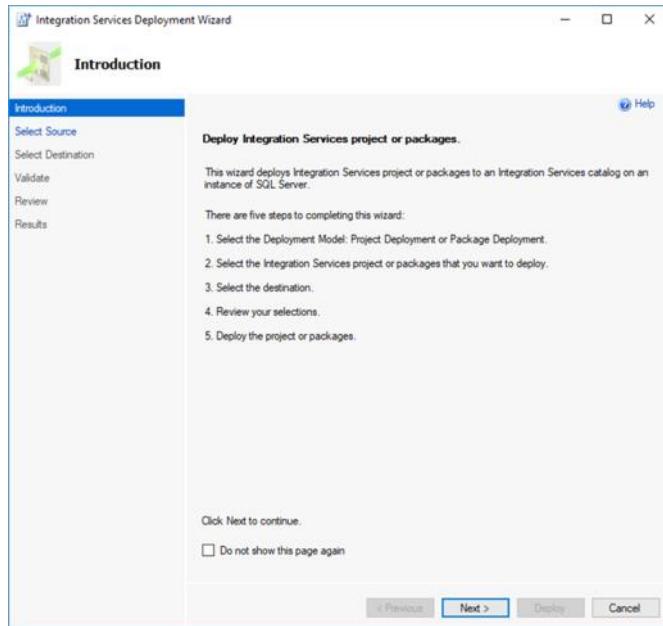
- 2) Right Click on the **SSISDB** in the **Integration Services Catalogs** and Create folder called **<ADFLab>**.



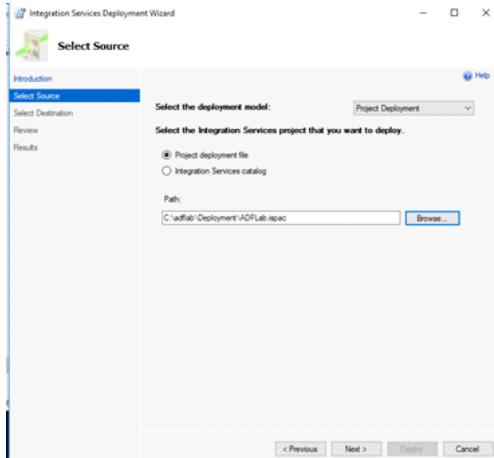
- 3) Navigate to the **Projects** folder beneath the **ADFLab** folder. Right click on the **Projects** folder to Start the Deploy Projects Wizard.



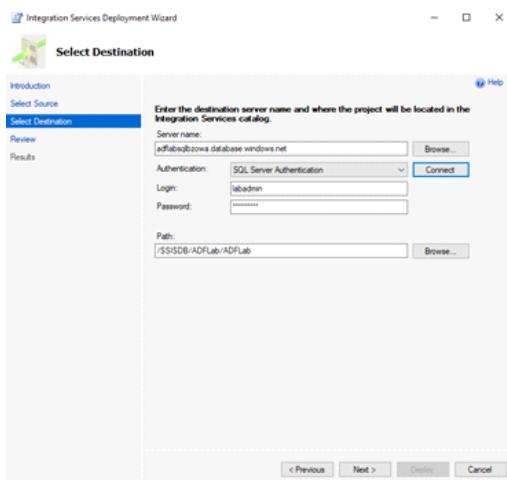
- 1) Click **Next**, if the Introduction Screen is displayed. (Feel Free top check the do not display)



- 1) Select Project Deployment and click Browse locate the **SSISDemo.ispac** file in the folder **Labs\06 - SSIS Lift and shift**.

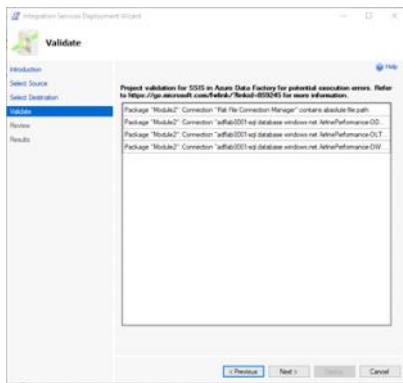


- 2) Select the Destination Integration Services Catalog

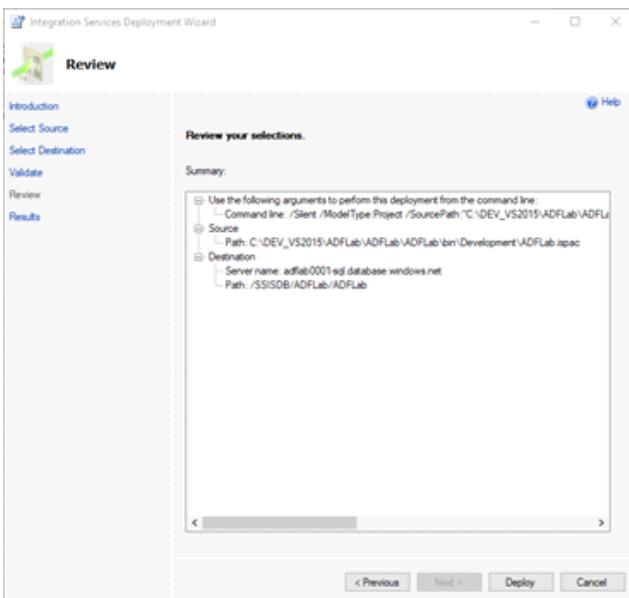


Item	Value	Notes
SQL Server	<Logical SQL Server>.database.windows.net	This will be the name of the azure data factory. It must be unique. Save this value for later as you will refer to it often
Authentication	SQL Server Authentication	
Login	DeploymentUser	This is the deployment user SQL Account
Password	<Your Password>	This is the password that you use to access the azure portal
Path	/SSISDB/ADFLAB	The path that represents the folder you created earlier this should already be populated

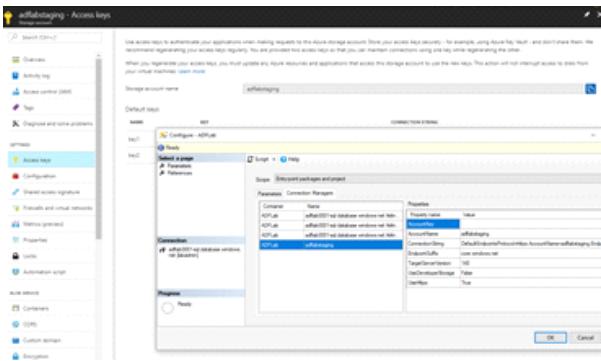
1) Review Validate Results.



1) Review configurations and Deploy the SSIS Project



4) Configure Connection Managers



Connection Manager	Value	Notes
SQL Server	<Logical SQL Server>.database.windows.net	This will be the name of the azure data factory. It must be unique. Save this value for later as you will refer to it often
Username	DataLoadUser	NOTE: this is the dataload user account not the deploymentUser
Password		The password you set

Update the package you should now be able to run and test it.

Summary

In this activity you learned how to Deploy an ispac file to an SSIS Catalog hosted in Azure Sql Database while the package it self was quite trivial and we could have done more with parameters environments and variables. This demonstrated the core principals

Activity 03a: Create a pipeline with an Execute SSIS Package activity

Activity Overview (5 minutes)

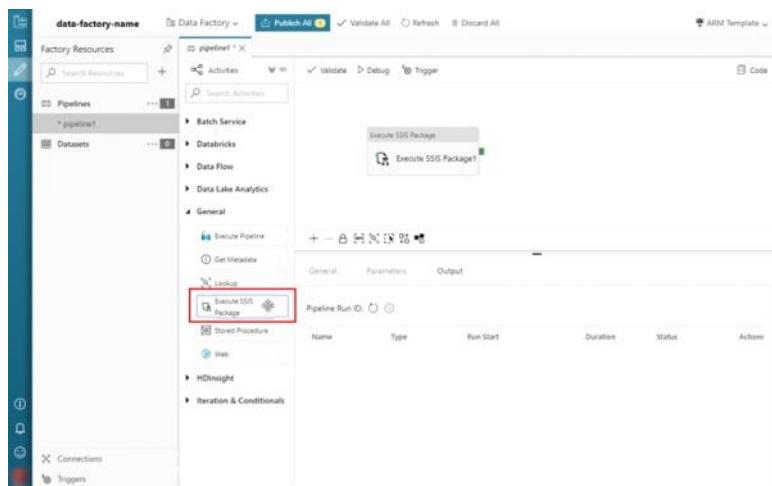
Now that we have created an SSIS Integration Runtime and deployed an SSIS Package to the Catalog, we create a pipeline with a Execute SSIS Package Activity. This activity simplifies the execution of SSIS Packages from Azure Data Factory, however as we will see in the following lab we can also fall back and use a generic stored procedure activity this gives us more fine grained control over the exact TSQL that is used.

Create a pipeline with an Execute SSIS Package activity

In this step, you use the Data Factory UI to create a pipeline. You add an Execute SSIS Package activity to the pipeline and configure it to run the SSIS package.

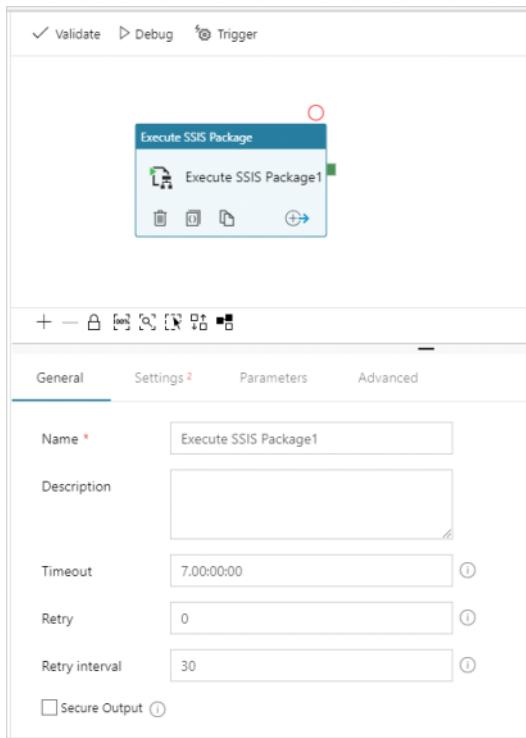
Create A New Pipeline Called **ExecuteSSIS**

In the Activities toolbox, expand General, and drag and drop the Execute SSIS Package activity to the pipeline canvas



On the General tab of properties for the Execute SSIS Package activity, provide a name and description for

the activity. Set optional timeout and retry values.



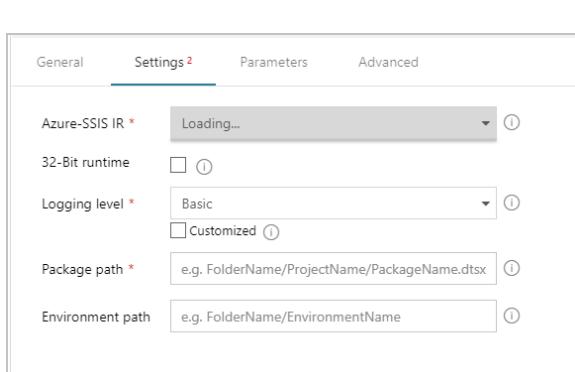
On the Settings tab of properties for the Execute SSIS Package activity, select the Azure-SSIS Integration

Runtime associated with the SSISDB database where the package is deployed. Provide the package path in

the SSISDB database in the format <folder name>/<project name>/<package name>.dtsx .

Optionally specify

32-bit execution and a predefined or custom logging level, and provide an environment path in the format <folder name>/<environment name> .



- 3) To validate the pipeline configuration, click Validate on the toolbar. To close the **Pipeline Validation Report**, click >>.

Activity Summary

After completing this activity we should have a new pipeline that can be used to execute an SSIS Package. We showed how to configure this activity. In the next activity, we will look at an alternative

method to execute the SSIS Package via a stored procedure activity. If you want feel free to skip ahead to **Activity 04: Execute the pipeline** in that activity we will show you how to execute and monitor the pipeline and also cover testing and debugging the deployment.

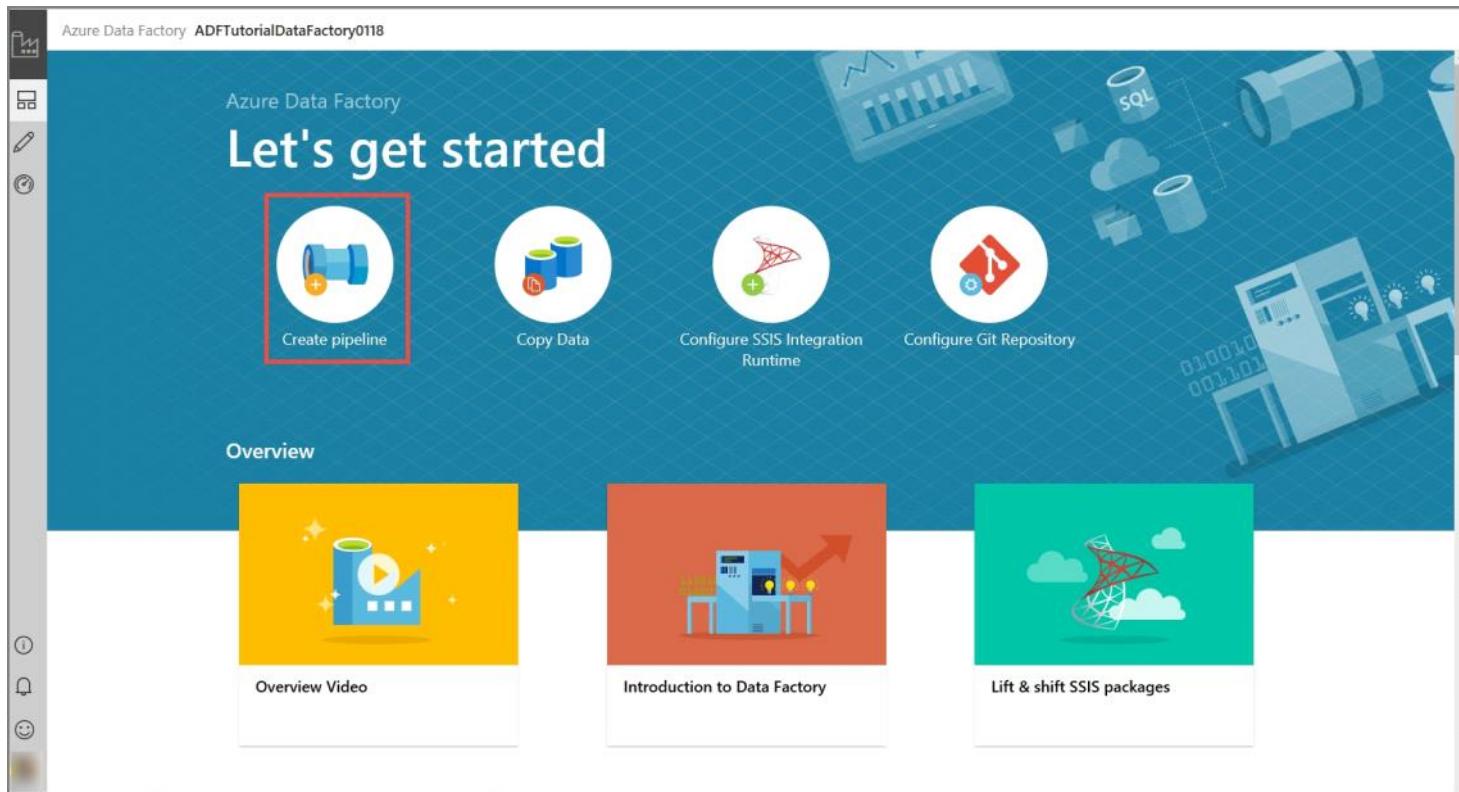
Activity 03b: Create a Pipeline with a Stored Procedure Activity

Activity Overview (5 minutes)

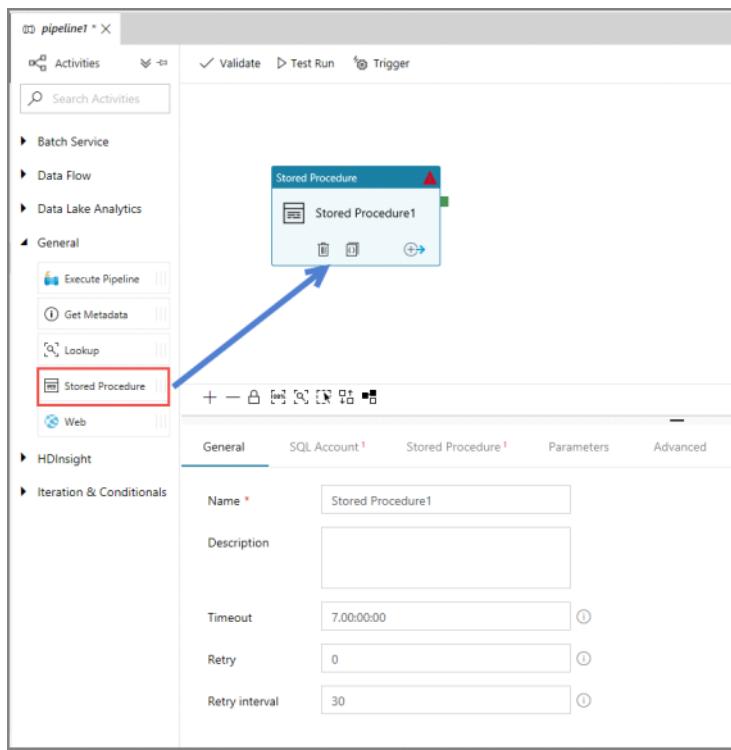
In this activity, we will look at an alternative method for executing an SSIS Package from Azure Data Factory using a Stored Procedure Activity.

Execute SSIS Package Using Stored Procedure Activity

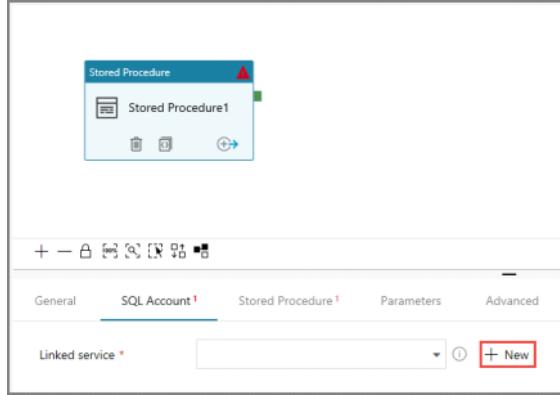
- 1) In the get started page, click **Create pipeline**:



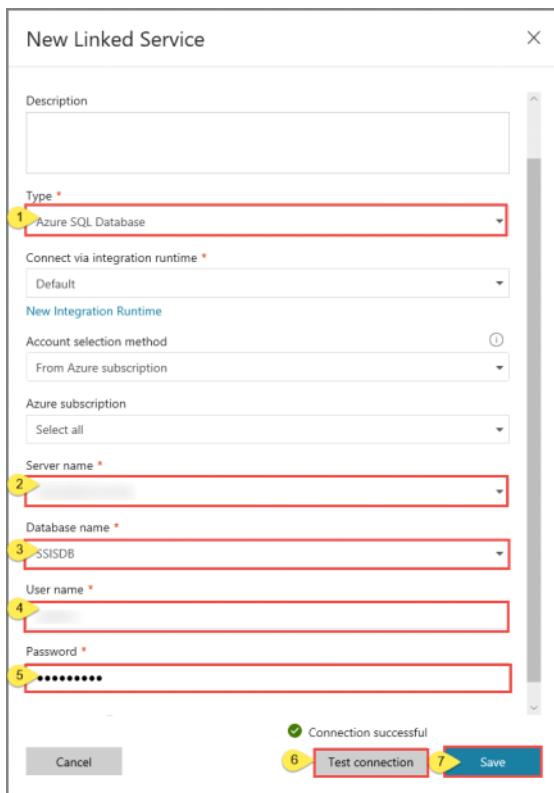
- 1) In the **Activities** toolbox, expand **General**, and drag-drop **Stored Procedure** activity to the pipeline canvas



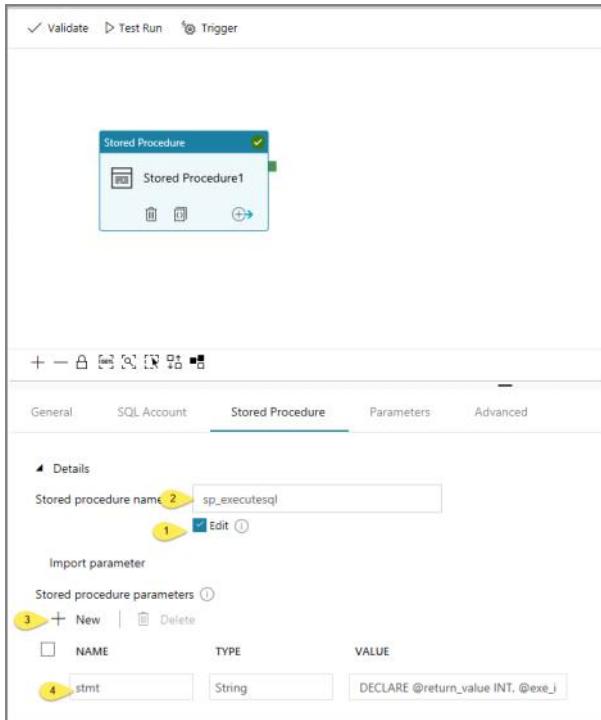
- 3) In the properties window for the stored procedure activity, switch to the **SQL Account** tab, and click **+ New**. You create a connection to the Azure SQL database that hosts the SSIS Catalog (SSIDB database).



- 4) In the **New Linked Service** window, do the following steps:



- 5) In the properties window, switch to the Stored Procedure tab from the SQL Account tab, and do the following steps:



You will need the following snippet for the **stmt** parameter:

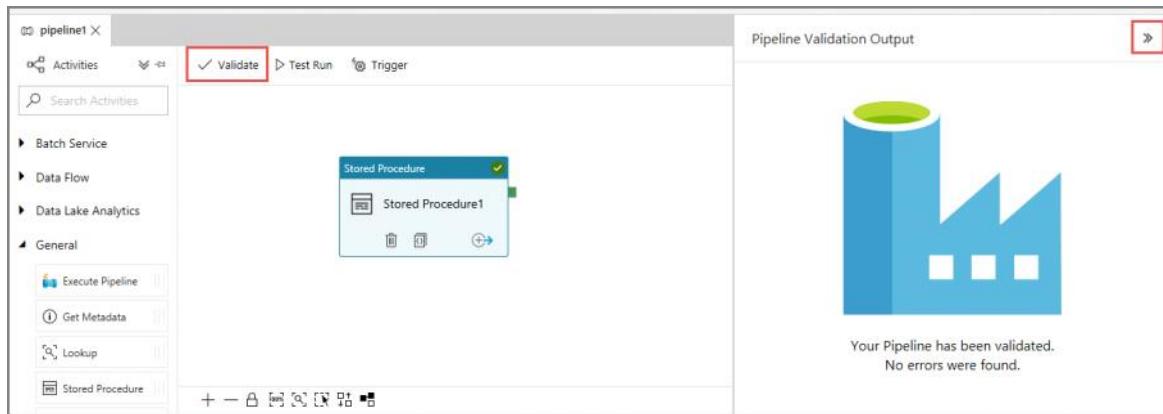
```
DECLARE @return_value INT, @exe_id BIGINT, @err_msg NVARCHAR(150) EXEC
@return_value=[SSISDB].
```

```

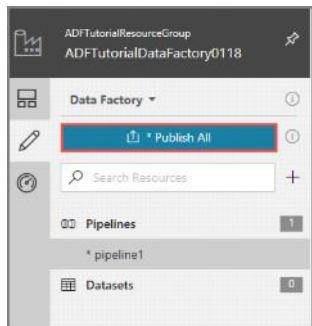
[catalog].[create_execution] @folder_name=N'<FOLDER name in SSIS Catalog>',
@project_name=N'<PROJECT name in SSIS Catalog>', @package_name=N'<PACKAGE name>.dtsx',
@use32bitruntime=0, @runinscaleout=1, @useanyworker=1, @execution_id=@exe_id OUTPUT
EXEC
[SSISDB].[catalog].[set_execution_parameter_value] @exe_id, @object_type=50,
@parameter_name=N'SYNCHRONIZED', @parameter_value=1 EXEC
[SSISDB].[catalog].[start_execution]
@execution_id=@exe_id, @retry_count=0 IF(SELECT [status] FROM
[SSISDB].[catalog].[executions]
WHERE execution_id=@exe_id)<>7 BEGIN SET @err_msg=N'Your package execution did not
succeed for
execution ID: ' + CAST(@exe_id AS NVARCHAR(20)) RAISERROR(@err_msg,15,1) END

```

- 6) To validate the pipeline configuration, click **Validate** on the toolbar. To close the **Pipeline Validation Report**, click >>



- 7) Publish the pipeline to **Data Factory** by clicking **Publish All** button..



Activity Summary

After completing this activity we should have a new pipeline that can be used to execute an SSIS Package. We showed how to configure this activity. In the next activity, we will show you how to execute and monitor the pipeline and also cover testing and debugging the deployment.

Activity 04: Execute the pipeline

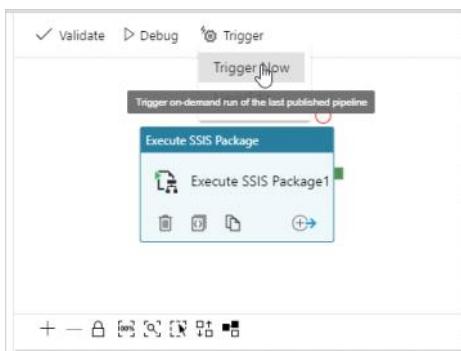
Activity Overview (5 minutes)

In this activity we will execute the pipeline we just published in the previous activity, After executing we will look at how we can monitor the execute and diagnose any failures.

NOTE: these steps should work for either of the pipelines we created in activity 03a or 03b

Trigger the pipeline

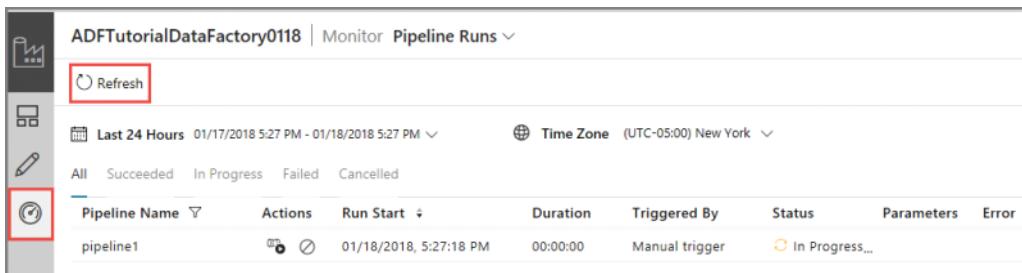
- 1) To trigger a pipeline run, click **Trigger** on the toolbar, and click **Trigger now**.



- 1) In the **Pipeline Run** window, select **Finish**.

Monitor the pipeline

- 1) Switch to the **Monitor** tab on the left. You see the pipeline run and its status along with other information (such as Run Start time). To refresh the view, click **Refresh**.



- 2) Click **View Activity Runs** link in the Actions column. You see only one activity run as the pipeline has only one activity (the Execute SSIS Package activity).

The screenshot shows the 'Pipeline Runs' section of the Azure Data Factory interface. It lists a single pipeline run for 'pipeline1'. The run details are as follows:

Activity Name	Activity Type	Actions	Run Start	Duration	Status	Integration Runtime
Execute SSIS Package1			01/18/2018, 5:27:21 PM	00:00:21	Succeeded	DefaultIntegrationRuntime (East US)

3) You can run the following query against the SSISDB database in your Azure SQL server to verify that the package executed.

```
select * from catalog.executions
```

The screenshot shows the results of the above query in SSMS. The output table has the following columns and data:

execution_id	folder_name	project_name	package_name	reference_id	reference_type	environment_folder_name	environment_name	project_id
1	2	MySSISProjects	SSISTest	Package.dtsx	NULL	NULL	NULL	1
2	3	MySSISProjects	SSISTest	Package.dtsx	NULL	NULL	NULL	1
3	4	MySSISProjects	SSISTest	Package.dtsx	NULL	NULL	NULL	1
4	5	MySSISProjects	SSISTest	Package.dtsx	NULL	NULL	NULL	1

4)

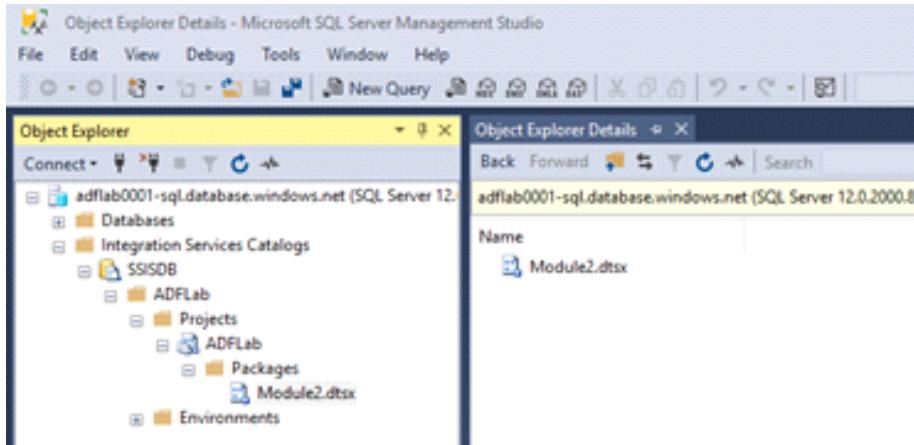
You can also get the SSISDB execution ID from the output of the pipeline activity run, and use the ID to check more comprehensive execution logs and error messages in SSMS.

The screenshot shows two windows side-by-side. On the left is the 'Pipeline Runs' monitor for 'myPipeline'. A specific run is selected, and its 'Output' tab is shown, revealing the execution ID: 'ExecutionID': 'd1fb02bd-0738-4e15-a43d-579747ac3f12'. On the right is Microsoft SQL Server Management Studio (SSMS) displaying the 'All Executions' report. This report shows five successful executions for the package 'ScaleOutProject'. The execution ID 'd1fb02bd-0738-4e15-a43d-579747ac3f12' is highlighted in both the monitor and the SSMS report.

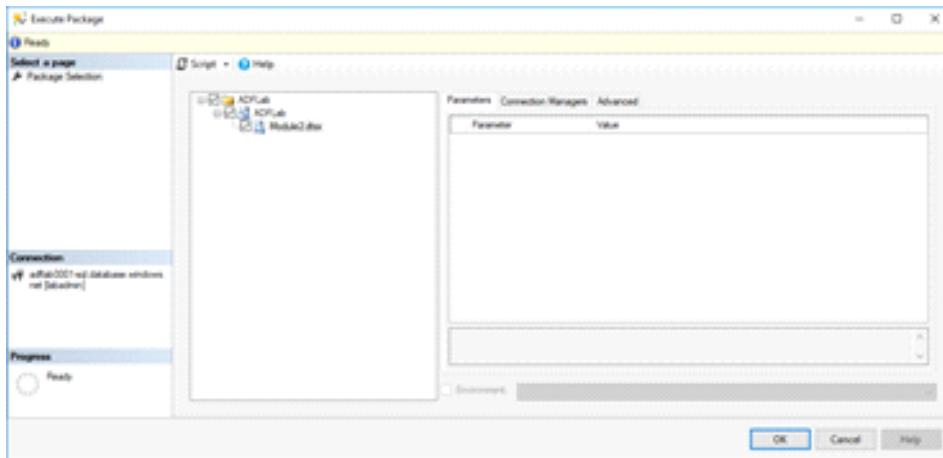
Troubleshooting

If the pipeline fails it is useful to try to execute the SSIS Package from SSMS to see if there is a problem with the deploy, Typical problems could include misconfigured Connection managers, hard coded paths or missing dependencies.

3) Right Click on the Package and choose Execute



4) Execute Package



1) Monitor Package Execution from Integration Services Dashboard

The screenshot shows the Integration Services Dashboard for a SQL Server instance. At the top, it displays 'Integration Services Dashboard' and the connection details 'on adflab0001-sql at 12/19/2017 8:18:58 PM'. A summary section shows the following counts: Failed (0), Running (1), Succeeded (0), and Others (0). To the right, there's a link to 'Other Integration Services Reports' with options for All Executions, All Validations, All Operations, and All Connections. Below the summary, a section titled 'Package Information (Past 24 Hours)' shows that 1 out of 1 packages have executed. A table titled 'Connection Information (Past 24 Hours)' lists failed connections, though none are currently shown. Another table titled 'Packages Detailed Information (Past 24 Hours)' provides a summary of executed packages, including their status, report, package path, failed executions, and last deployed time. The table includes columns for Status, Report, Package Path, Failed/Total Executions, and Last Deployed Time.

Status	Report	Package Path	Failed/Total Executions	Last Deployed Time
Running	Quartz	All Messages	0/1	12/19/2017 8:18:58 PM

If the package fails you can use the integration services dashboard to investigate the error message further.

Bonus Lab: Enhance the SSIS Package

Our SSIS package was pretty useless Create a new SSIS Package to either export the customer data into one of the file based stores (you will need to azure feature pack installed to develop this).

Make the connection managers parameterized

For an extra tough challenge attempt to export the data from adventure works LT Database **SalesLT_SalesOrderDetail** to Either Azure SQL Database or Azure Data Lake.. Did this work? Why or why not? :)

Lab 07: Operationalizing ADF

Background

In the previous labs we have focused on building a robust solutions. Now we want to look at how we can use source control to allow a better team development experience, and also how we can setup monitoring alerting and logging

We will also learn how to import and export the solutions

Prerequisites

In order to complete this lab, you will need to have previously created:

- TBD

Overview

- Create a Schedule trigger
 - Parameter 1
 - Parameter 2
- Configure Pipeline logging
 -
- Create a "Master Pipeline"
- Add a stored procedure activity

Let's get started with Activity 1!

In the previous labs we have focused on building a robust solutions. Now we want to look at how we can use source control to allow a better team development experience, and also how we can setup monitoring alerting and logging

We will also learn how to import and export the solutions

- 1) Configuring Source Control
 - a. Configure with github
- 2) Setting Up Logging
 - a. Configuring the OMS Dashboard
- 3) Setting up an alert
 - a. Create an alert when a pipeline fails
- 4) Import and export ARM
 - a. Export ARM Template
 - b. Import ARM Template
- 5) Bonus Lab: Create a Power BI Dashboard to stream the data in real time

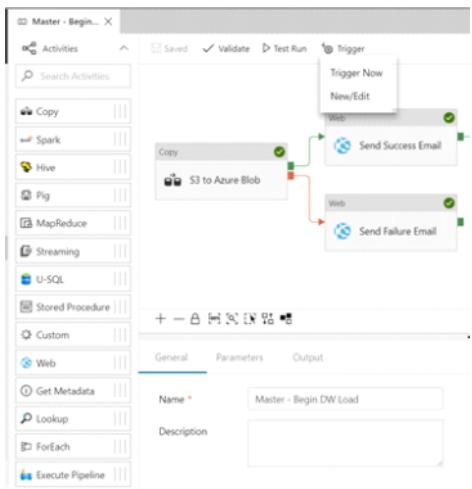
Activity 01: Creating Triggers

Activity Overview (5 minutes)

This activity walks through how to schedule a pipeline run from the Azure Data Factory GUI. We will be showing how to use the Schedule trigger which provides an interface for time-based scheduling of pipelines.

Create A Schedule Trigger

- 1) Click the Trigger Icon and New/Edit to create a new trigger.



- 1) From the Choose Trigger dropdown choose +New to create a new trigger.



- 3) Configure Trigger properties

[← New Trigger](#) [X](#)

Name * [?](#)
Daily - DW Load Trigger

Description

Type * [?](#)
 Schedule Tumbling Window

Start Date * [?](#)
12/18/2017, 11:15 AM

Recurrence * [?](#)
 Daily Every Day(s)

[Advanced recurrence options](#) [?](#)

Execute at these times [?](#)

Hours [X](#)
Minutes

Schedule execution times
12:00

End * [?](#)
 No End On Date

[Cancel](#) [Next](#)

Item	Value	Notes
Name	Trg_Daily	This will be the name of the trigger.
Type	Schedule	
Start Date	Chose Today's Date	This is the Azure Active Directory Account That you login into your Subscription with
Reoccurrence	Daily Every 1 Day	The trigger will execute Once daily
Advanced Reoccurrence options	Hours: 23 Minutes: 59	This configures the reoccurrence time. Th trigger will execute at 11:59 pm
End	No End	The trigger will continue to run until we either delete it or or disable it

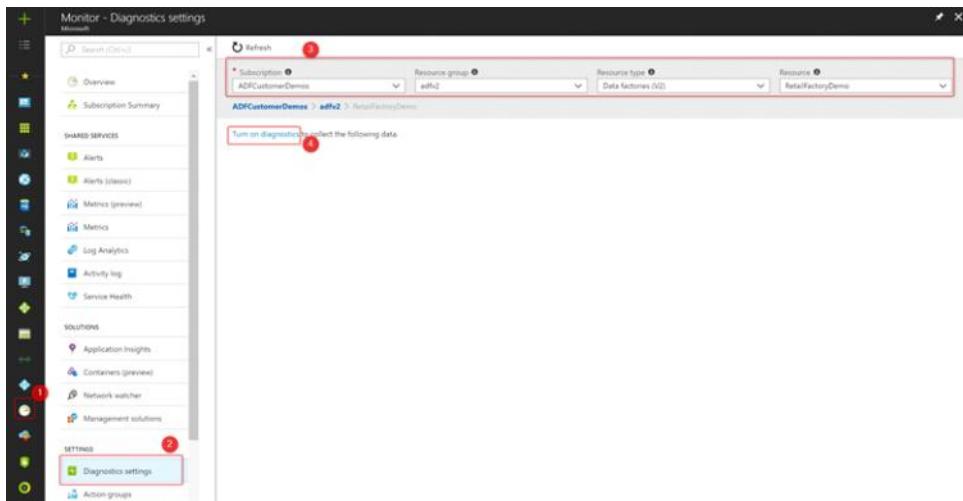
Activity Summary

After completing this activity we should have a new trigger that can be used to schedule pipeline execution. We showed how to configure this trigger for a wall clock schedule. In the next activity, we will look at configuring some of the built-in logging and reporting options from Azure Data Factory.

Activity 02a: Configure Logging

Enable Diagnostic Settings for your data factory.

- 1) Select Azure Monitor -> Diagnostics settings -> Select the data factory -> Turn on diagnostics.



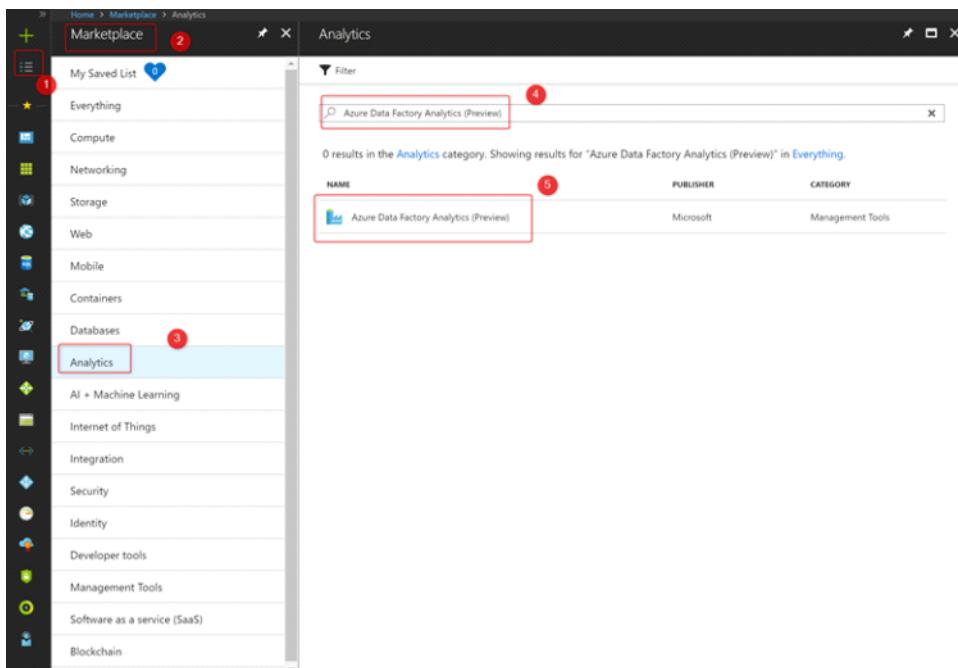
- 1) Provide diagnostic settings including configuration of the workspace.

The screenshot shows the 'Diagnostics settings' blade for creating a new workspace. The 'Name' field is set to 'CustomerDemoDiagnostics'. The 'Send to Log Analytics' checkbox is checked. The 'Log Analytics' configuration section is expanded. The 'LOG' collection section is highlighted with a red box and a red circle with the number 5, showing 'ActivityRuns', 'PipelineRuns', and 'TriggerRuns' selected. The 'METRIC' collection section is also highlighted with a red box. On the right, there's a list of existing workspaces and a 'Create New Workspace' button. A red circle with the number 1 points to the 'Name' field, a red circle with the number 2 points to the 'Send to Log Analytics' checkbox, a red circle with the number 3 points to the 'Log Analytics' configuration section, a red circle with the number 4 points to the 'Create New Workspace' button, and a red circle with the number 5 points to the 'LOG' collection section.

Activity 02b: Install Azure Data Factory Analytics

Install Azure Data Factory Analytics from Azure Marketplace

1)



Provide diagnostic settings including configuration of the workspace.

 Azure Data Factory Analytics (Preview)
Microsoft

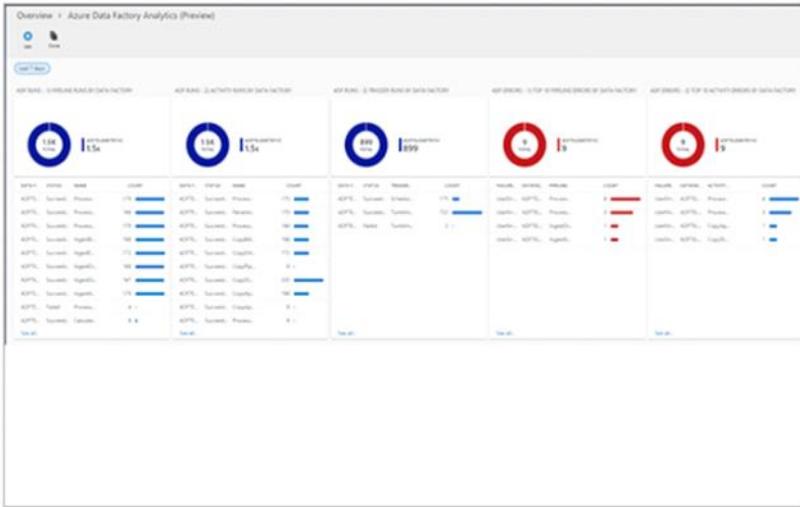
This solution provides you a summary of overall health of your Data Factory, with options to drill into details and to troubleshoot unexpected behavior patterns.

With rich, out of the box views you can get insights into key processing including:

- At a glance summary of data factory pipeline, activity and trigger runs
- Ability to drill into data factory activity runs by type
- Summary of data factory top pipeline, activity errors

Pre-requisite: To take advantage of this solution, Data Factory should enable Log Analytics to push diagnostic data to OMS workspace.

[Save for later](#)

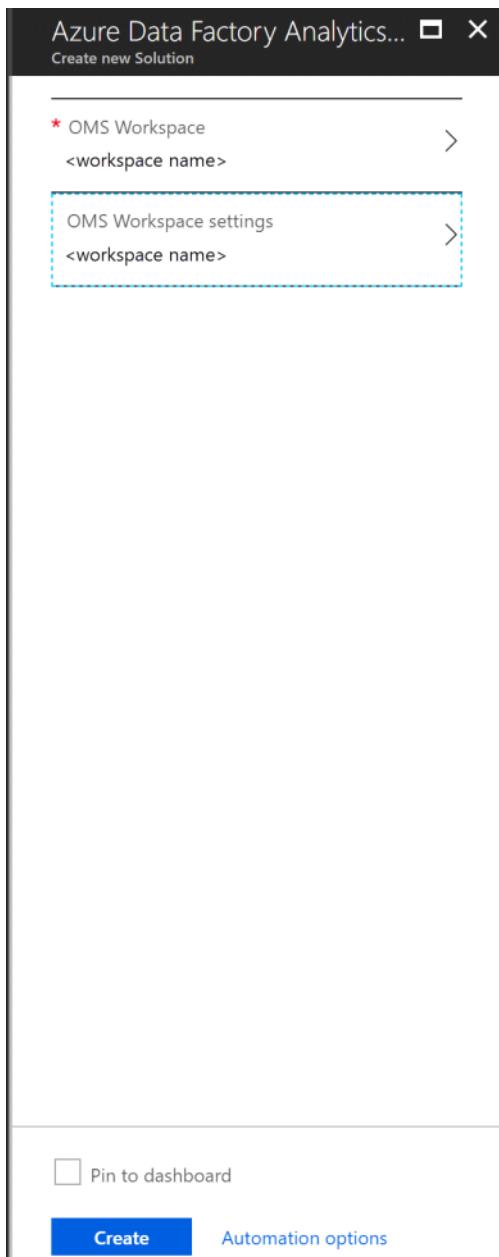


PUBLISHER Microsoft

USEFUL LINKS [Watch Video](#) [Learn More](#)

[Create](#)

Click Create and select the Workspace and Workspace settings.



Installing Azure Data Factory Analytics creates a default set of views that enables the following metrics:

- ADF Runs- 1) Pipeline Runs by Data Factory
- ADF Runs- 2) Activity Runs by Data Factory
- ADF Runs- 3) Trigger Runs by Data Factory
- ADF Errors- 1) Top 10 Pipeline Errors by Data Factory
- ADF Errors- 2) Top 10 Activity Runs by Data Factory
- ADF Errors- 3) Top 10 Trigger Errors by Data Factory
- ADF Statistics- 1) Activity Runs by Type
- ADF Statistics- 2) Trigger Runs by Type
- ADF Statistics- 3) Max Pipeline Runs Duration

From <<https://docs.microsoft.com/en-us/azure/data-factory/monitor-using-azure-monitor>>

Home > AzureDataFactoryAnalytics

AzureDataFactoryAnalytics

Solution

Search (Ctrl+F)

Overview

Activity log

Access control (IAM)

Diagnose and solve problems

SETTINGS

Locks

Automation script

GENERAL

OMS Workspace

Properties

Saved searches

WORKSPACE DATA SOURCES

Virtual machines

Storage accounts logs

Azure Activity log

SUPPORT + TROUBLESHOOTING

Resource health

New support request

Essentials

Resource group: <resource group name>

Status: Active

Location: East US

Subscription name: (change)

Subscription ID: <subscription name>

Subscription ID: <subscription id>

Solution: AzureDataFactoryAnalytics (mflasko)

Type: Microsoft.OperationsManagement/solutions

Workspace Name: <workspace name>

Management services: Operations logs

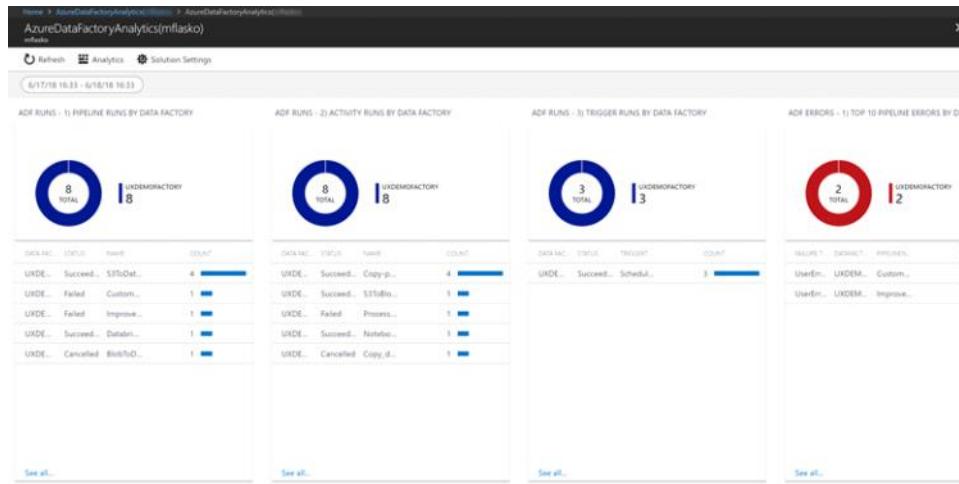
Summary

Azure Data Factory Analytics (Preview)

Run Type	Count
ActivityRuns	8
PipelineRuns	8
TriggerRuns	3
TOTAL	19

Solution Resources

1 AzureDataFactoryAnalytics



Activity 03: Creating an Alert

Log in to the Azure portal and click Monitor -> Alerts to create alerts.

The screenshot shows the Microsoft Azure portal interface. On the left, there is a navigation sidebar with various service icons and links. The 'Monitor' link is highlighted with a red box. The main content area is titled 'Monitor - Alerts' and shows a list of shared services. The 'Alerts' item is also highlighted with a red box. Other items in the list include 'Alerts (Classic)', 'Metrics (preview)', 'Metrics', 'Log Analytics', 'Activity log', and 'Service Health'. Below this is a section titled 'SOLUTIONS' with links to 'Application Insights', 'Network watcher', and 'Management solutions'. At the bottom is a 'SETTINGS' section with a link to 'Diagnostics settings'.

Click + New Alert rule to create a new alert.

The screenshot shows the Microsoft Azure Monitor - Alerts dashboard. At the top, there's a search bar and a 'New Alert Rule' button. Below that, there are sections for 'Subscription' (set to 'ADF + Meshup Central Data Lab'), 'Resource group' (set to 'Type to start filtering...'), and 'Time Range' (set to 'Past 6 Hours'). The main area displays 'Fired Alerts' (0) and 'Total Alert Rules' (0). A table below shows columns for NAME, STATE, ALERT CRITERIA, RESOURCE GROUP, and TARGET RESOURCE, with a note 'No results to display'.

Define the Alert condition.

This screenshot shows the 'Create rule' wizard, step 1: 'Define alert condition'. It has three sections: 'Alert target' (selected), 'Target criteria' (disabled), and 'Signal logic' (disabled). Under 'Alert target', there's a 'Select target...' button (marked 1) and a dropdown menu for 'Filter by subscription' (marked 2) set to 'mfaskoAzure', 'Filter by resource type' (marked 3) set to 'All', and a search input (marked 4) containing 'udemo'. The 'Signal logic' section shows a preview of 'mfaskoAzure > udemo > UxDemoFactory'.

This screenshot continues the 'Create rule' wizard at step 1. The 'Signal logic' section is now open, titled 'Configure signal logic'. It shows a list of available signals under 'All signals (8)'. The first few items in the list are:

SIGNAL NAME	SIGNAL TYPE	MONITOR SERVICE
Failed pipeline runs met...	Metric	Platform
Succeeded pipeline runs...	Metric	Platform
Failed activity runs metri...	Metric	Platform
Succeeded activity run ...	Metric	Platform
Failed trigger runs metric...	Metric	Platform
Succeeded trigger runs ...	Metric	Platform
Integration runtime CPU...	Metric	Platform
Integration runtime ava...	Metric	Platform

Define the Alert details.

Define the Action group.

Home > Monitor - Alerts > Create rule > Add action group > Email/SMS/Push/Voice

Create rule

Rules management

- ▽ 1. Define alert condition
- ▽ 2. Define alert details
- ^ 3. Define action group

Notify your team via email and text messages or automate actions using webhooks, runbooks or integrating with other services.

ACTION GROUP NAME	ACTION GROUP TYPE
No action group selected	

[Select action group](#) [+ New action group](#)

Add action group

* Action group name [i](#) EmailGroup ✓

* Short name [i](#) EmailGroup ✓

* Subscription [i](#) mflaskoAzure ▾

* Resource group [i](#) Default-ActivityLogAlerts (to be created) ▾

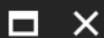
Actions

ACTION NAME	ACTION TYPE	STATUS	DETAILS
SendEmailAlert ✓	Email/SMS/Push/Voice ▾		Edit details
Unique name for the action...		▼	

[Privacy Statement](#)

[Pricing](#)

Email/SMS/Push/Voice



Name

Email

SMS

Country code * Phone number



 Carrier charges may apply.

Azure app Push Notifications

[Learn about the connecting to your Azure resources using the Azure app.](#)

This is the email you use to log into your Azure account.

Voice

Country code * Phone number



OK

Email/SMS/Push/Voice

Name

SendEmailAlert

Email

SMS

Country code ***** Phone number

1

 Carrier charges may apply.

Azure app Push Notifications

Learn about the connecting to your Azure resources using the Azure app.

email@example.com

This is the email you use to log into your Azure account.

Voice

Country code ***** Phone number

1

1234567890

Home > Monitor - Alerts > Create rule

Create rule

Rules management

✓ 1. Define alert condition

✓ 2. Define alert details

✗ 3. Define action group

Notify your team via email and text messages or automate actions using webhooks, runbooks or integrating with external ITSM solutions. Learn more about ITSM integration [here](#)

ACTION GROUP NAME

ACTION GROUP TYPE

REMOVE

EmailGroup

1 Email, 1 SMS

Bonus Activity: Configure Source Control

We walked through the needed steps for source control configuration in the lecture portion
If you have either an azure Devops or github account you can set this up for your data factory

Using the below resources setup your Azure Data Factory with either Azure DevOps Source control or
github

Author with Azure Repos Git integration

<https://docs.microsoft.com/en-us/azure/data-factory/author-visually#author-with-azure-repos-git-integration>

Author with GitHub integration

<https://docs.microsoft.com/en-us/azure/data-factory/author-visually#author-with-github-integration>