Lab 3 – Run SSIS packages in ADF

In this lab you will provision an Azure-SSIS integration runtime (IR), deploy a SSIS package to an Azure-hosted SSIS catalog and run it in Azure Data Factory.

Before you start the lab you will need:

- Visual Studio 2019
- The SQL Server Integration Services Projects extension for VS 2019
- The Azure Feature Pack for Integration Services. Download and install the 32-bit feature pack for SQL Server 2017 (file name "SsisAzureFeaturePack_2017_**x86**.msi"). Visual Studio is a 32-bit application so requires the 32-bit feature pack.

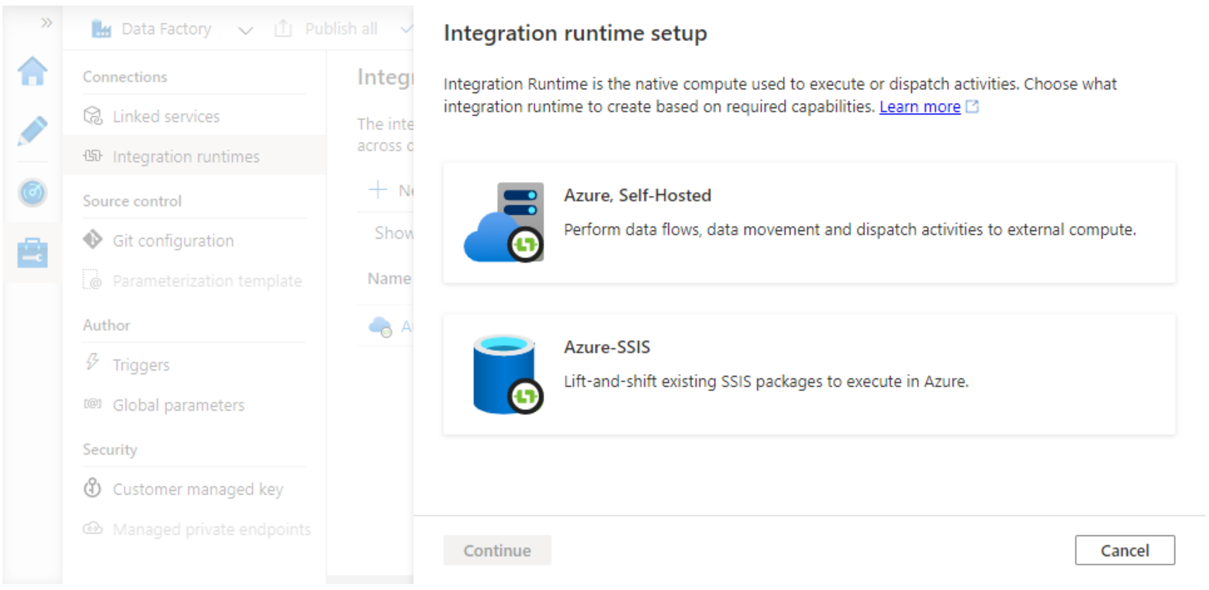
Lab 3.1 – Create SSIS Integration Runtime

ADF runs SSIS packages using a special Azure-SSIS integration runtime – effectively an instance of SQL Server Integration Services running in the cloud. Create an Azure-SSIS IR like this.

1. In the ADF UX, open the Management Hub and select "Integration runtimes" from the "Connections" section of its sidebar.

2. In the main "Integration runtimes" pane click "+ New".

3. Select the "Azure-SSIS" tile and click "Continue". This launches a three-step wizard to create a new Azure-SSIS IR.



Integration runtime setup

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

Azure, Self-Hosted

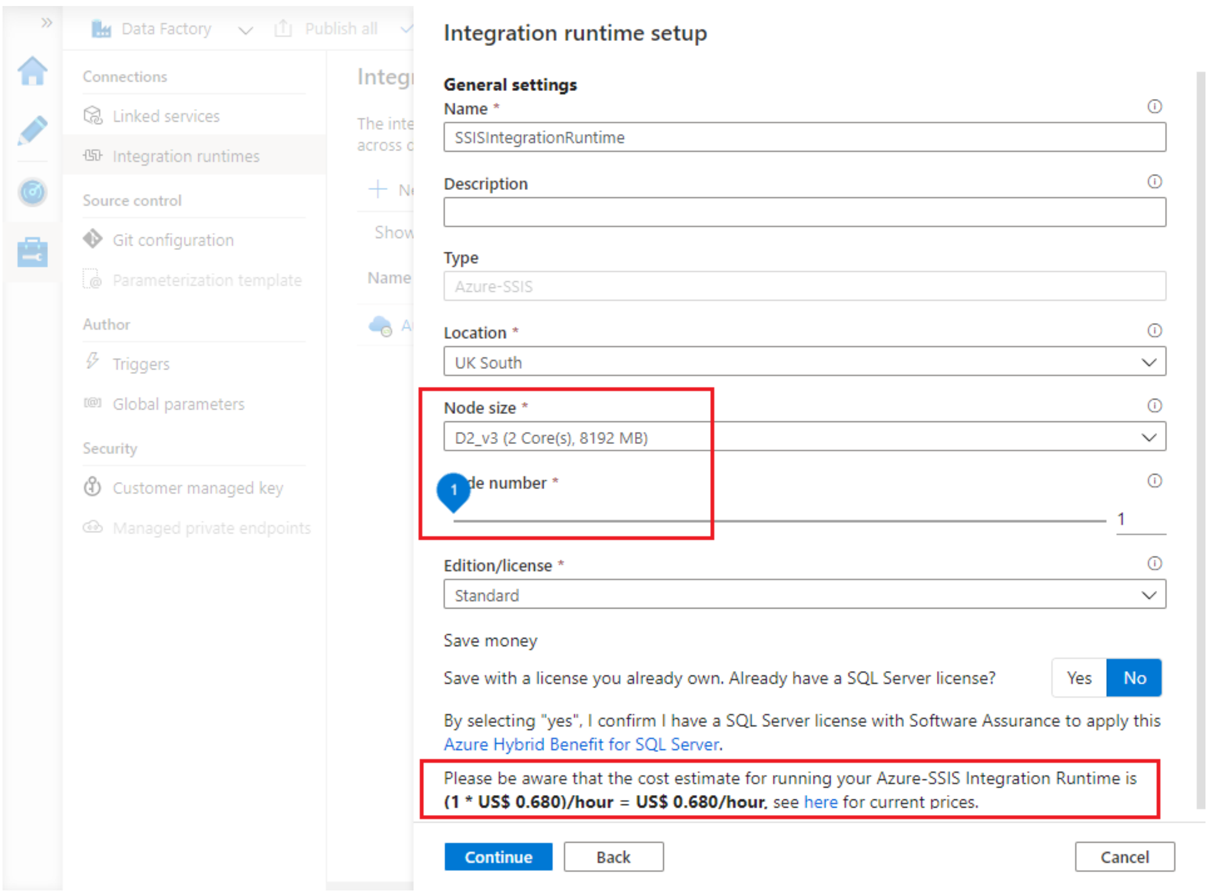
Perform data flows, data movement and dispatch activities to external compute.

Azure-SSIS

Lift-and-shift existing SSIS packages to execute in Azure.

4. In step 1 of the wizard, provide general settings for the IR:

   - Give it a name.
   - Choose the same location as your data factory.
   - Choose the **smallest** option from the "Node size" list. This is just to keep the cost of the lab exercise down – in production environments you will want a node size that gives you the performance you need. Try choosing a few different values to see the effect on the cost estimate displayed at the bottom of the blade.
   - Set "Node number" to 1. Again, this is to keep your lab costs down – try choosing a few different values to see the effect on the cost estimate displayed at the bottom of the blade.
   - Leave other options with their default values and click "Continue".



Integration runtime setup

General settings

Name *: SSISIntegrationRuntime

Description:

Type: Azure-SSIS

Location *: UK South

Node size *: D2_v3 (2 Core(s), 8192 MB)

Node number *: 1

Edition/license *: Standard

Save money

Save with a license you already own. Already have a SQL Server license? Yes [✓ No]

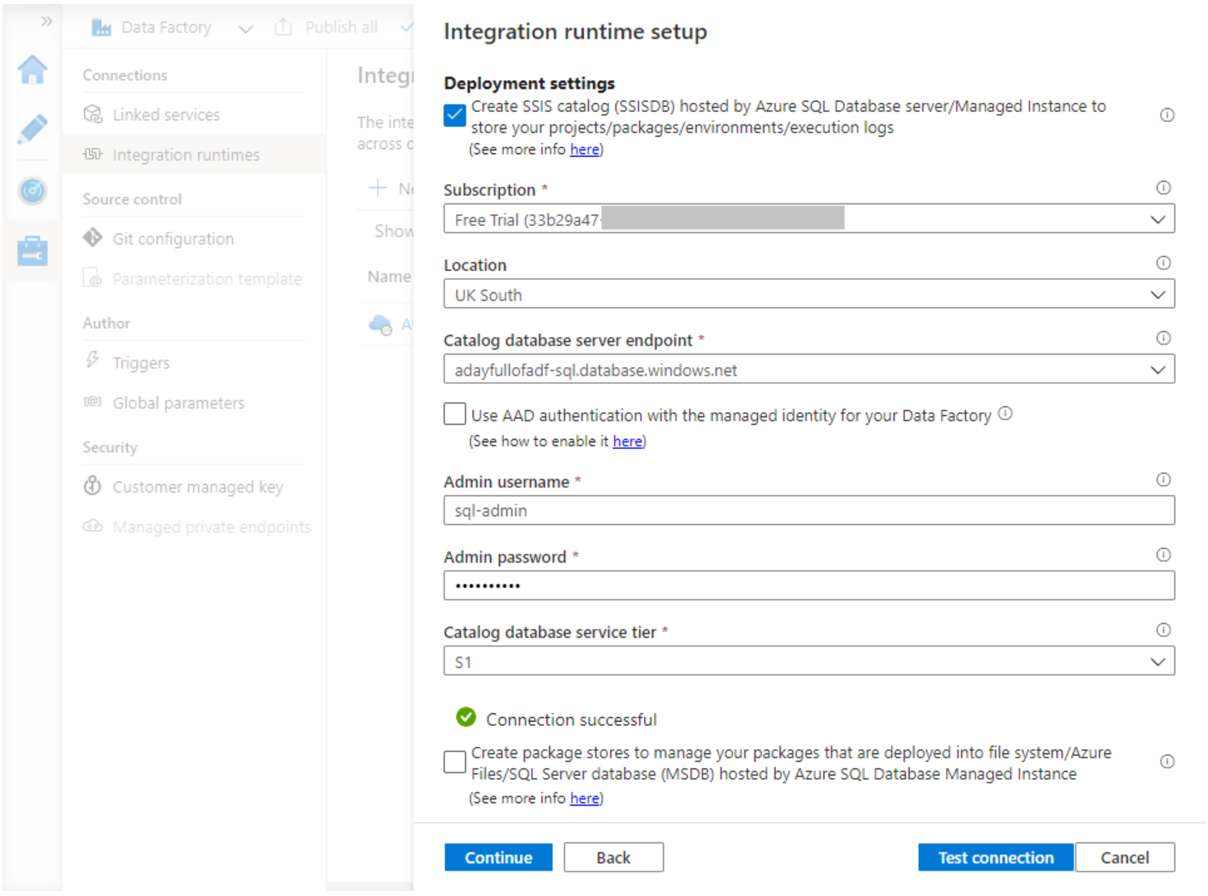
By selecting "yes", I confirm I have a SQL Server license with Software Assurance to apply this Azure Hybrid Benefit for SQL Server.

Please be aware that the cost estimate for running your Azure-SSIS Integration Runtime is **(1 * US$ 0.680)/hour = US$ 0.680/hour**, see here for current prices.

5. In step 2 of the wizard, configure a new SSIS catalog. This needs a SQL server to host the associated SSISDB database – use the server you created in Lab 2.1.

   - Make sure that the "Create SSIS catalog (SSISDB)…" checkbox is ticked.
   - Choose your subscription and the same location you have been using all along.
   - For "Catalog database server endpoint", enter the fully-qualified name of your server. (You found the name in step 5 of Lab 2.1).
   - Enter your SQL admin username and password, and leave other settings with their default values.

   Click "Test connection" to verify that you have configured the SQL server and credentials correctly. When the test succeeds, click Continue.



Integration runtime setup

Deployment settings

✓ Create SSIS catalog (SSISDB) hosted by Azure SQL Database server/Managed Instance to store your projects/packages/environments/execution logs
(See more info here)

Subscription *: Free Trial (33b29a47 [redacted]

Location: UK South

Catalog database server endpoint *: adayfullofadf-sql.database.windows.net

☐ Use AAD authentication with the managed identity for your Data Factory
(See how to enable it here)

Admin username *: sql-admin

Admin password *: ••••••••••

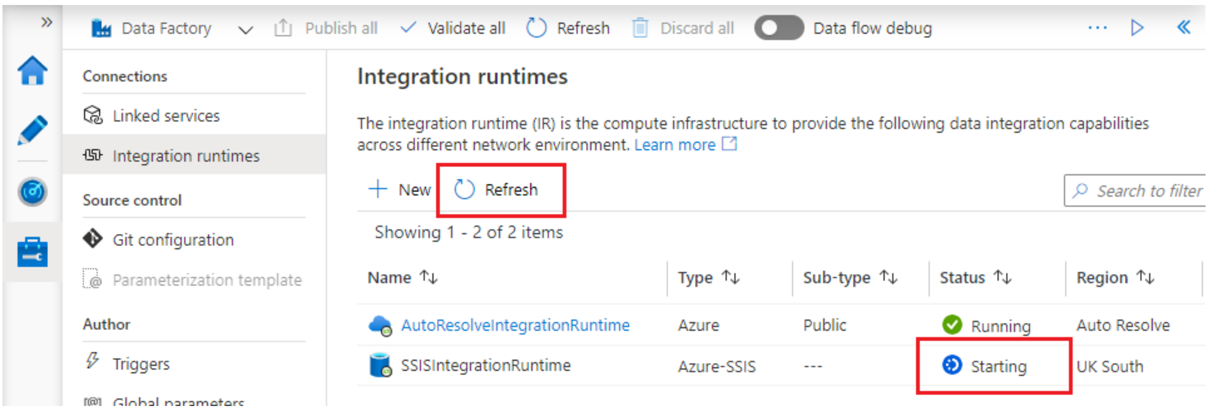
Catalog database service tier *: S1

Connection successful

☐ Create package stores to manage your packages that are deployed into file system/Azure Files/SQL Server database (MSDB) hosted by Azure SQL Database Managed Instance
(See more info here)

6. Step 3 of the wizard provides advanced configuration settings – click “Continue” to accept the defaults and move on.

7. Finally, review the settings you have provided – in particular the cost estimate at the bottom of the blade. Use the “Previous” button to revisit and adjust settings if required, otherwise click “Create” to create and start the integration runtime. The wizard blade closes and your new Azure-SSIS IR appears in the list of integration runtimes with a Status of “Starting”.



Integration runtimes

The integration runtime (IR) is the compute infrastructure to provide the following data integration capabilities across different network environment. Learn more

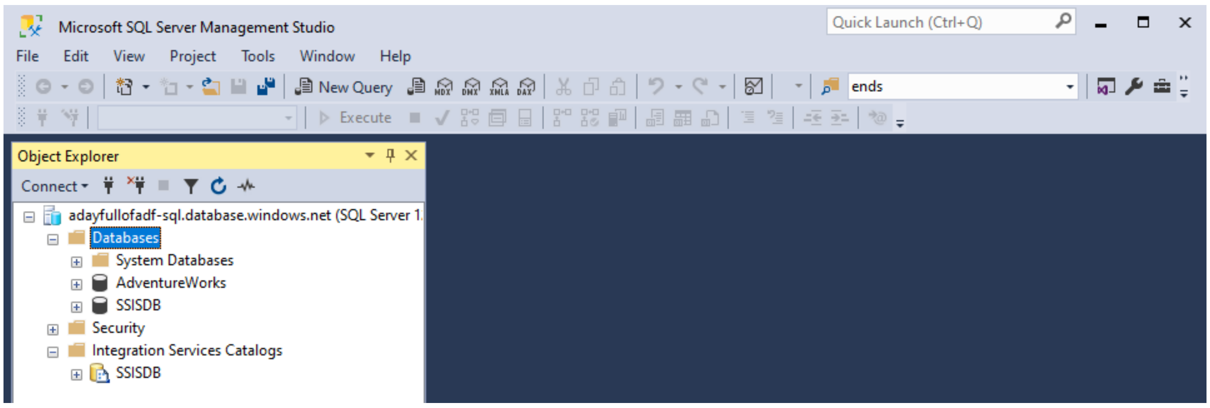
Showing 1 - 2 of 2 items

Name	Type	Sub-type	Status	Region
AutoResolveIntegrationRuntime	Azure	Public	Running	Auto Resolve
SSISIntegrationRuntime	Azure-SSIS	---	Starting	UK South

8. The new integration runtime will take a few minutes to provision and start. Use the “Refresh” button to monitor the IR’s status until it changes to “Running”.

Note: You are charged for any time an Azure-SSIS IR spends running. You can manage costs by starting the runtime only when you need to run SSIS packages and stopping it afterwards. You can do this in the Azure portal or programmatically using ADF's Web activity. Leave it running for this lab, but remember to stop it when you finish.

9. Connect to your SQL server instance using SSMS – you will notice that an SSISDB Integration Services Catalog has now been created, along with its supporting SSISDB database. (If you already have an open SSMS window, you may need to disconnect and reconnect to the server to pick up the change correctly).



Lab 3.2 – Deploy SSIS packages

In this lab you will use your prepared Visual Studio environment to edit and deploy an SSIS project.

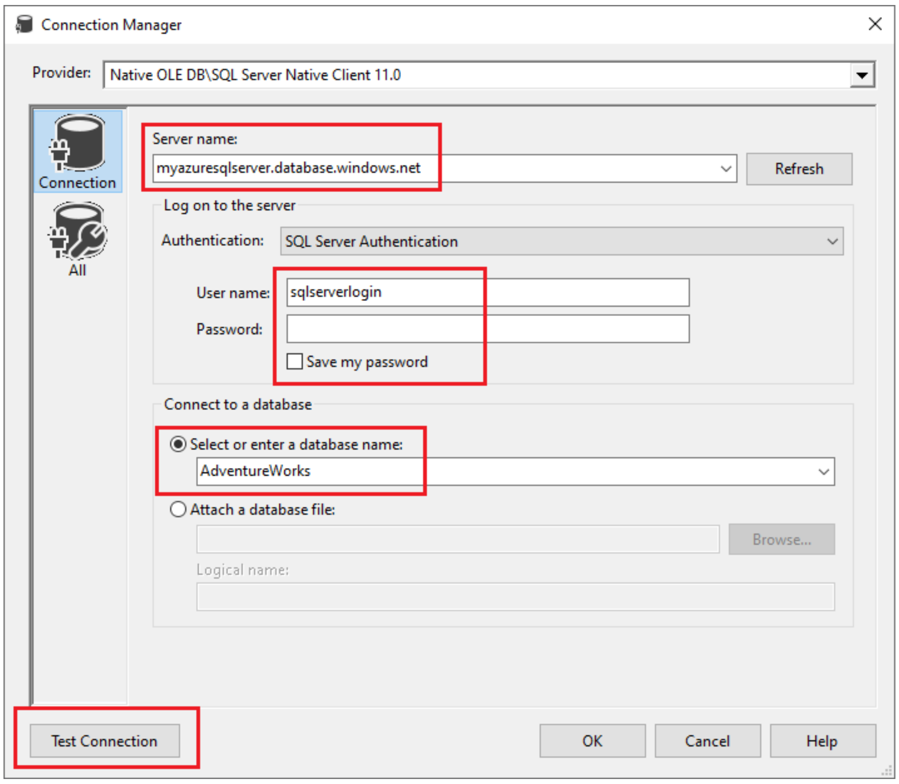
1. Open the SsisPackages.sln Visual Studio solution (stored alongside the other lab files in the GitHub). The solution consists of a single SSIS project containing a single package: "CopyProductCategory.dtsx". Open it.

   The package performs a similar task to the pipeline from Lab 2 – it copies the contents of the [SalesLT].[ProductCategory] table from the [AdventureWorks] database and into Azure Data Lake storage.

2. The "CopyProductCategory.dtsx" package has two connection managers: one for the source database, one for Azure Data Lake storage. Open the connection manager named "AzureSqlDatabase" and update its details to match your [AdventureWorks] database:

   - change the server name to match your Azure SQL database server
   - specify the admin username and password and tick the "Save my password" checkbox
   - check that the database name is correct.

   Test the updated connection manager by clicking "Test Connection" in the bottom left, then click "OK" to save your changes.


Connection Manager

Provider: Native OLE DB\SQL Server Native Client 11.0

Server name: myazuresqlserver.database.windows.net

Log on to the server

Authentication: SQL Server Authentication

User name: sqlserverlogin

Password:

☐ Save my password

Connect to a database

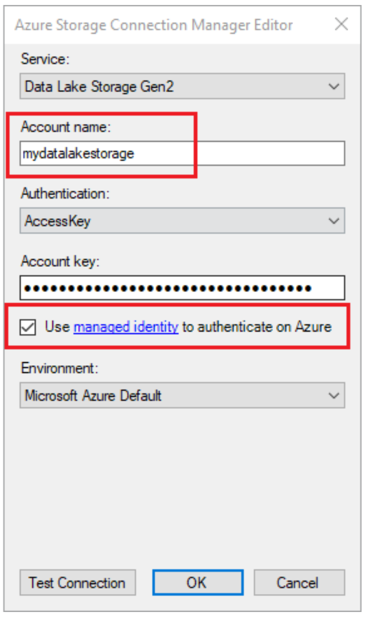
● Select or enter a database name: AdventureWorks

○ Attach a database file:

Logical name:

3. Open the "AzureDataLakeStorage" connection manager and update its settings:

   - replace the saved Account name with the name of your data lake storage account
   - ensure that the "Use managed identity to authenticate on Azure" is ticked – this will allow your data factory to connect to the data lake using its MSI, as in Lab 2.


Azure Storage Connection Manager Editor

Service: Data Lake Storage Gen2

Account name: mydatalakestorage

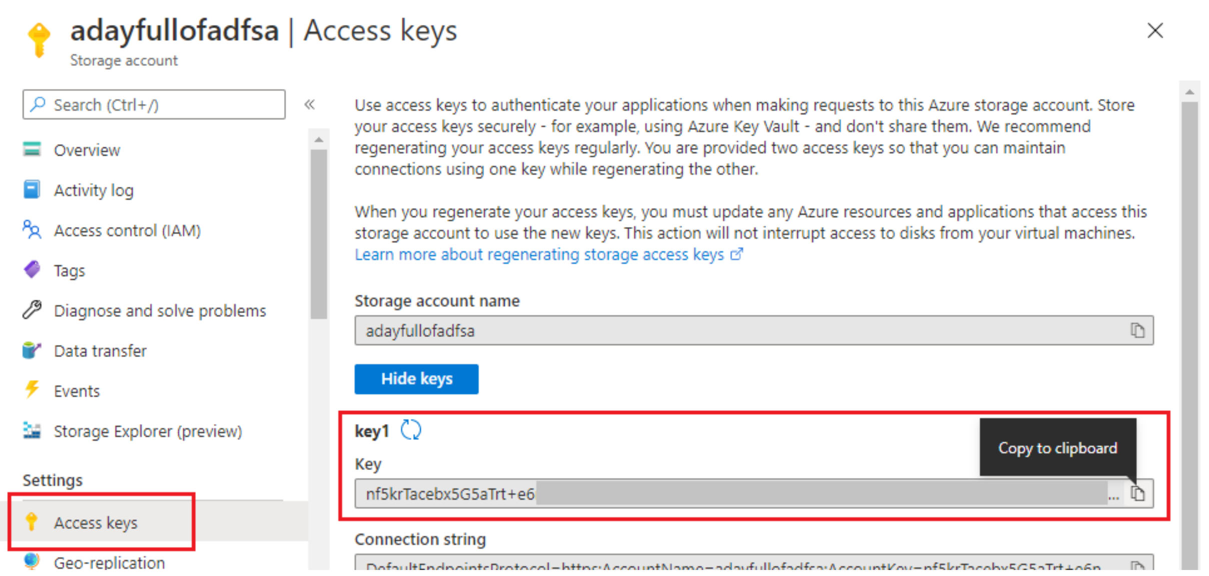
Authentication: AccessKey

Account key: ••••••••••••••••••••••••••••••••••

☑ Use managed identity to authenticate on Azure

Environment: Microsoft Azure Default

If you click "Test Connection" here, Visual Studio will attempt to connect to your data lake using the configured account key, so the test will fail. If you want to verify your account name in this way – or if you want to run the package directly in VS – you must replace the configured account key value with one of the keys from your own storage account. You can find these on the "Access keys" page of the storage account blade in the Azure portal.



adayfullofadfsa | Access keys
Storage account

Use access keys to authenticate your applications when making requests to this Azure storage account. Store your access keys securely - for example, using Azure Key Vault - and don't share them. We recommend regenerating your access keys regularly. You are provided two access keys so that you can maintain connections using one key while regenerating the other.

When you regenerate your access keys, you must update any Azure resources and applications that access this storage account to use the new keys. This action will not interrupt access to disks from your virtual machines.
Learn more about regenerating storage access keys

Storage account name: adayfullofadfsa

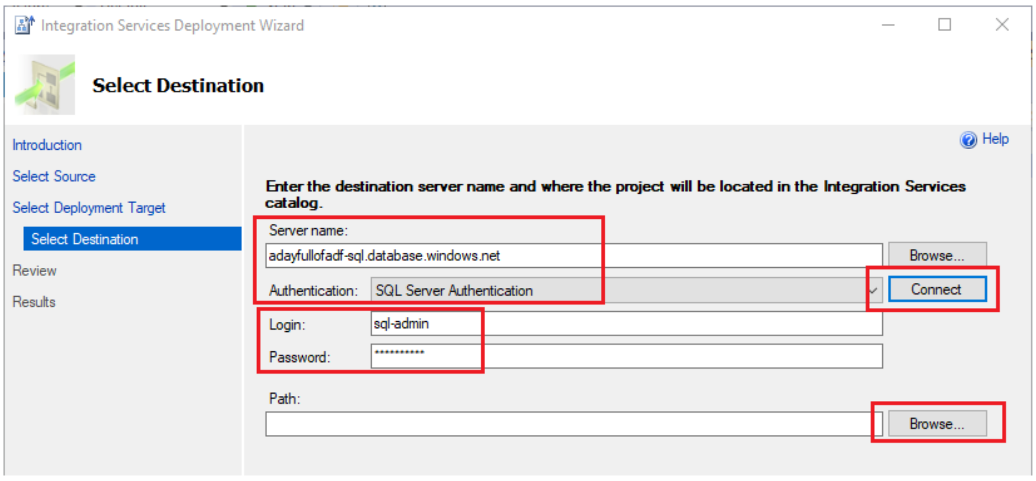
Key: nf5krTacebx5G5aTrt+e6

Connection string: DefaultEndpointsProtocol=https;AccountName=adayfullofadfsa;AccountKey=nf5krTacebx5G5aTrt+e6n

4. Right-click the project in Visual Studio's solution explorer and click "Deploy" to launch the deployment wizard. Click "Next" to skip the introduction page, if displayed.

5. On the "Select Deployment Target" page, choose "SSIS in Azure Data Factory" and click "Next".

6. Update the "Select Destination" page with your Azure SQL Server name and change the "Authentication" option to "SQL Server Authentication". Complete the "Login" and "Password" fields with your SQL admin credentials, then click "Connect". When connection succeeds, the "Path" field is enabled – click the "Browse" button to its right.



Integration Services Deployment Wizard

Select Destination

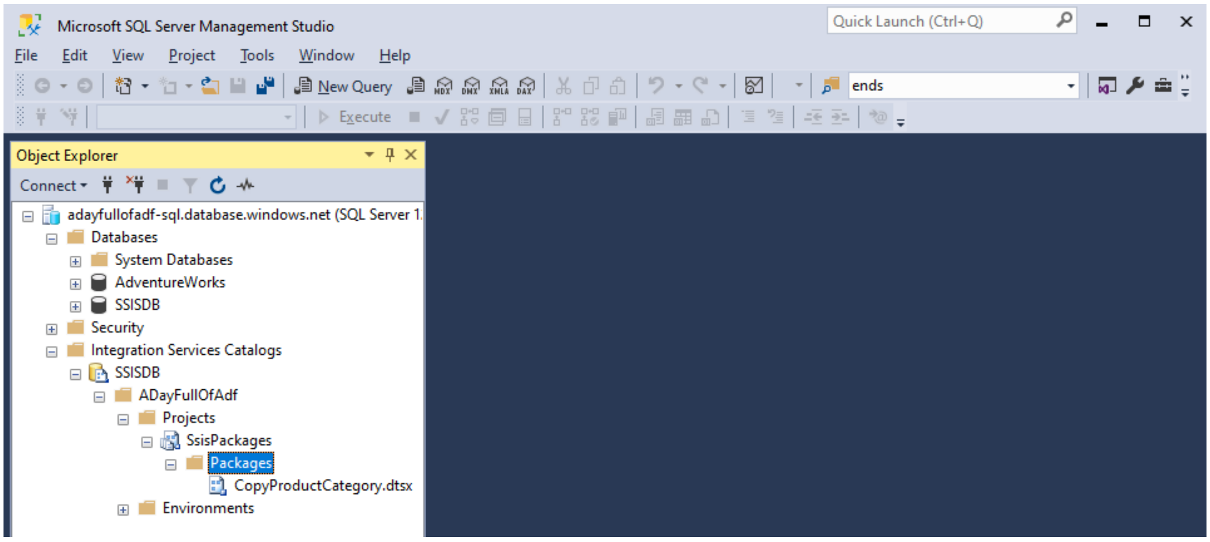
Enter the destination server name and where the project will be located in the Integration Services catalog.

Server name: adayfullofadf-sql.database.windows.net
Authentication: SQL Server Authentication
Login: sql-admin
Password: **********
Path:

7. The "Browse for Folder or Project" dialog is displayed. Click "New folder…" and enter a new folder name of your choice. Click "OK" to create the folder, then "OK" again to select it. The "Select Destination" page's "Path" field will now contain the path "/SSISDB/<your new folder name>/SsisPackages". Click "Next", review your selections, then click "Deploy" to deploy the project to the integration runtime's SSIS catalog.

   You can only make deployments to an Azure-SSIS IR when it is running.

8. Refresh your view of the server's Integration Services Catalog in SSMS. You will now be able to see the deployed project and package.



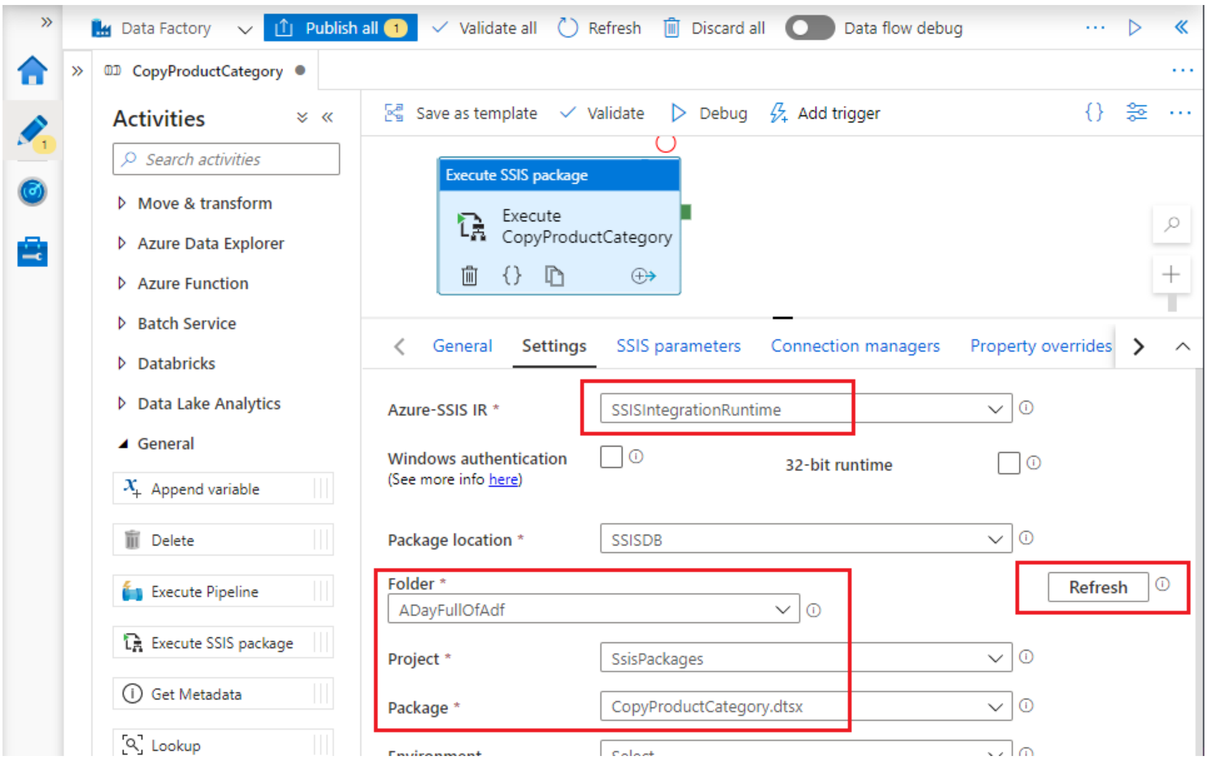
Lab 3.3 – Run the SSIS package in ADF

Now you're ready to run the deployed SSIS package in ADF.

1. Open the ADF authoring canvas and create a new pipeline. Give it a name of your choice.

2. Expand the "General" group in the activity toolbox, then drag an "Execute SSIS package" activity onto the pipeline canvas. Name the activity appropriately and select the "Settings" tab in the activity's configuration pane.

3. Choose your Azure-SSIS IR, then click the "Refresh" button to the right of the "Folder" dropdown. Use the "Folder", "Project" and "Package" dropdowns to choose your deployment folder, the "SsisPackages" project and the "CopyProductCategory.dtsx" package respectively.

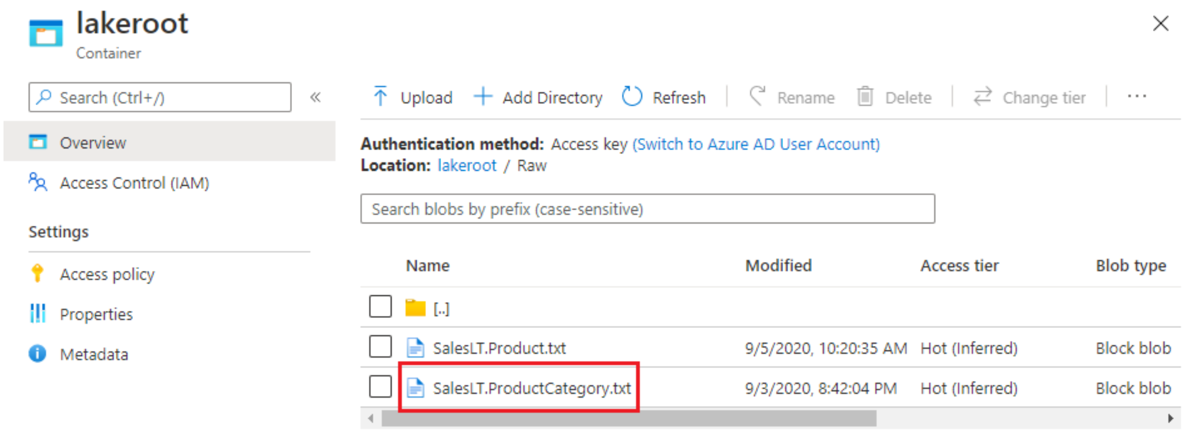


Azure-SSIS IR * SSISIntegrationRuntime
Windows authentication (See more info here)
32-bit runtime
Package location * SSISDB
Folder * ADayFullOfAdf
Project * SsisPackages
Package * CopyProductCategory.dtsx

4. Save/publish your changes, then click “Debug” above the pipeline canvas to run the pipeline in debugging mode. Verify that the SSIS package runs successfully.

> **Note:** At the time of writing my pipeline fails with the message “The TLS version of the connection is not permitted on this storage account”. This appears to be a new issue – I have asked a question about it on Microsoft Q&A. As a workaround, reduce the data lake storage account’s “Minimum TLS Version” to **1.0**. You can do this on the “Configuration” page of the storage account blade in the Azure portal.

5. Inspect the “lakeroot” data lake container’s “Raw” folder in the Azure portal. The file “SalesLT.ProductCategory.txt” has now appeared, copied to the data lake by the SSIS package which was executed by the new ADF pipeline.



Authentication method: Access key (Switch to Azure AD User Account)
Location: lakeroot / Raw

Name	Modified	Access tier	Blob type
[..]			
SalesLT.Product.txt	9/5/2020, 10:20:35 AM	Hot (Inferred)	Block blob
SalesLT.ProductCategory.txt	9/3/2020, 8:42:04 PM	Hot (Inferred)	Block blob

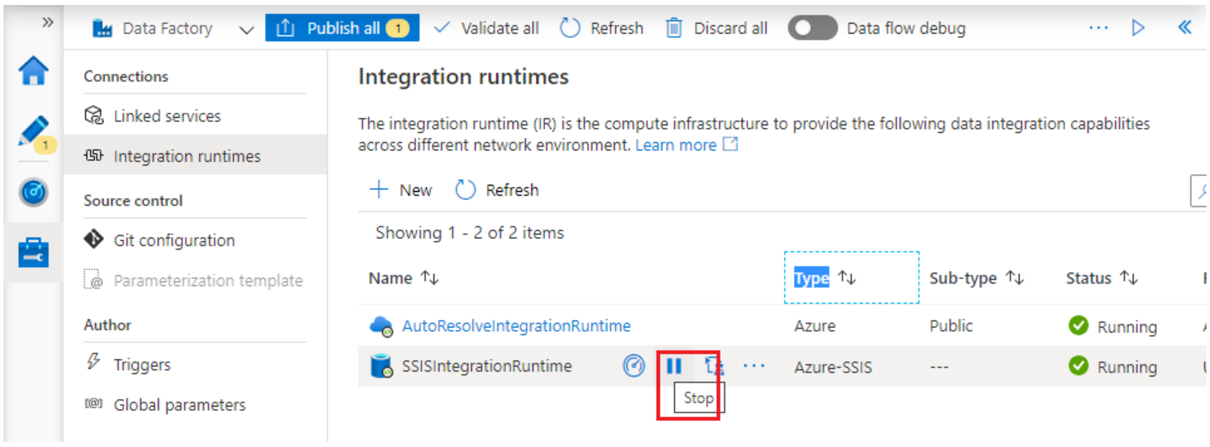
Lab 3.4 – Stop the integration runtime

Did I mention that the Azure-SSIS IR incurs charges whenever it’s running? You can stop it now!

1. In the ADF UX, open the Management Hub and select “Integration runtimes” from the “Connections” section of its sidebar.

2. Hover over the Azure-SSIS IR in the list of integration runtimes. A set of controls appears to the right of the runtime’s name.

3. The “pause” symbol in the set of controls is the IR’s stop button – click it to stop the IR.



Integration runtimes

The integration runtime (IR) is the compute infrastructure to provide the following data integration capabilities across different network environment. Learn more

Showing 1 - 2 of 2 items

Name	Type	Sub-type	Status
AutoResolveIntegrationRuntime	Azure	Public	Running
SSISIntegrationRuntime	Azure-SSIS	---	Running

4. The ADF UX prompts “Are you sure you want to stop the integration runtime?”. Click “Stop”.

5. The confirmation dialog closes, returning you to the Integration runtimes list where the Azure-SSIS IR now has Status “Stopping”. Use the “Refresh” button to monitor its status until it changes to “Stopped”. The IR cannot be used to execute SSIS packages until you restart it.

Recap

In Lab 3 you:

- created an SSIS Integration Runtime to enable ADF to run SSIS packages
- deployed an SSIS project to the SSIS-IR’s Integration Services catalog
- used the Execute SSIS package activity to run a package in the deployed project
- stopped the integration runtime.