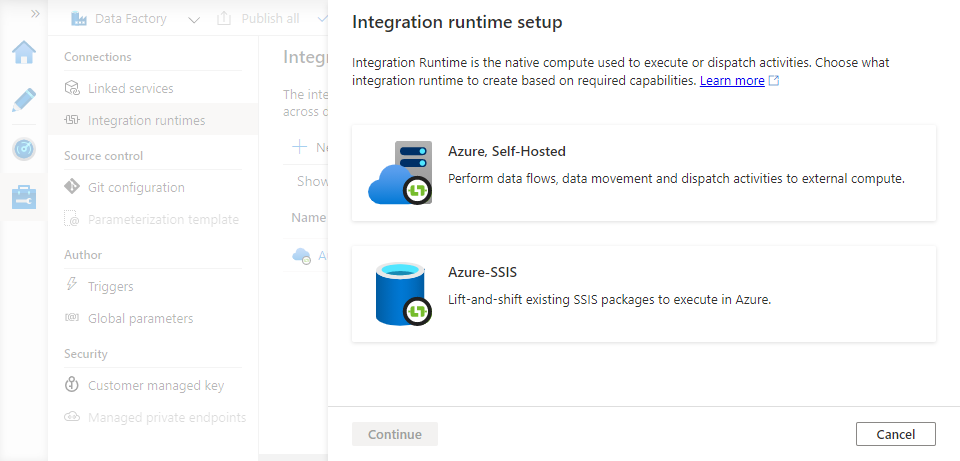
Lab 3 – Run SSIS packages in ADF

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.

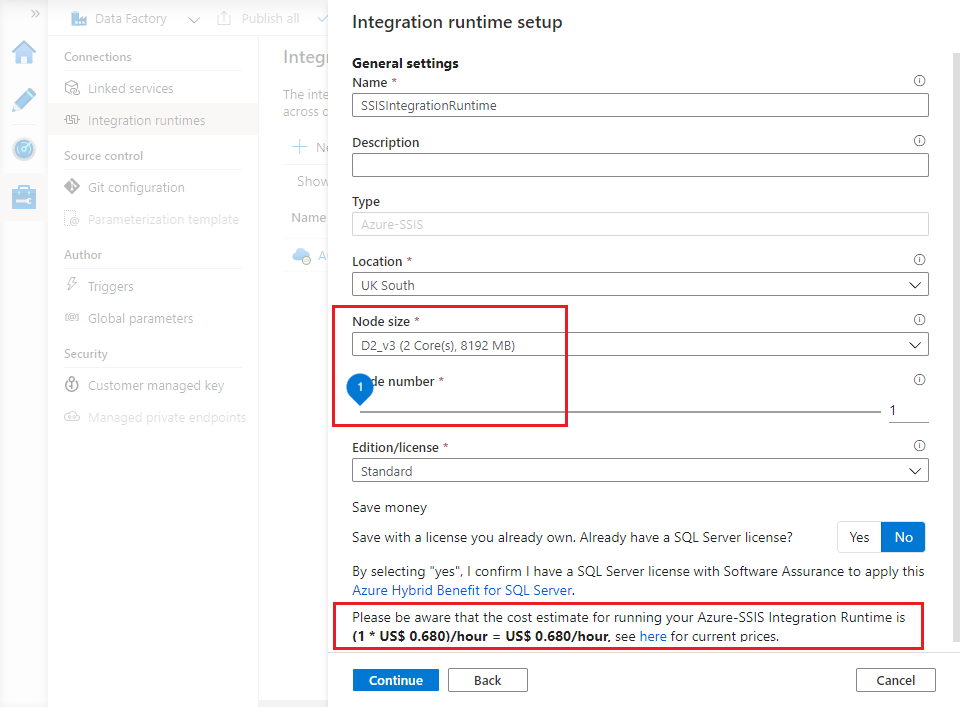
# Lab 3.1 – Create SSIS Integration Runtime

ADF runs SSIS packages using a special Azure-SSIS integration runtime – basically 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.

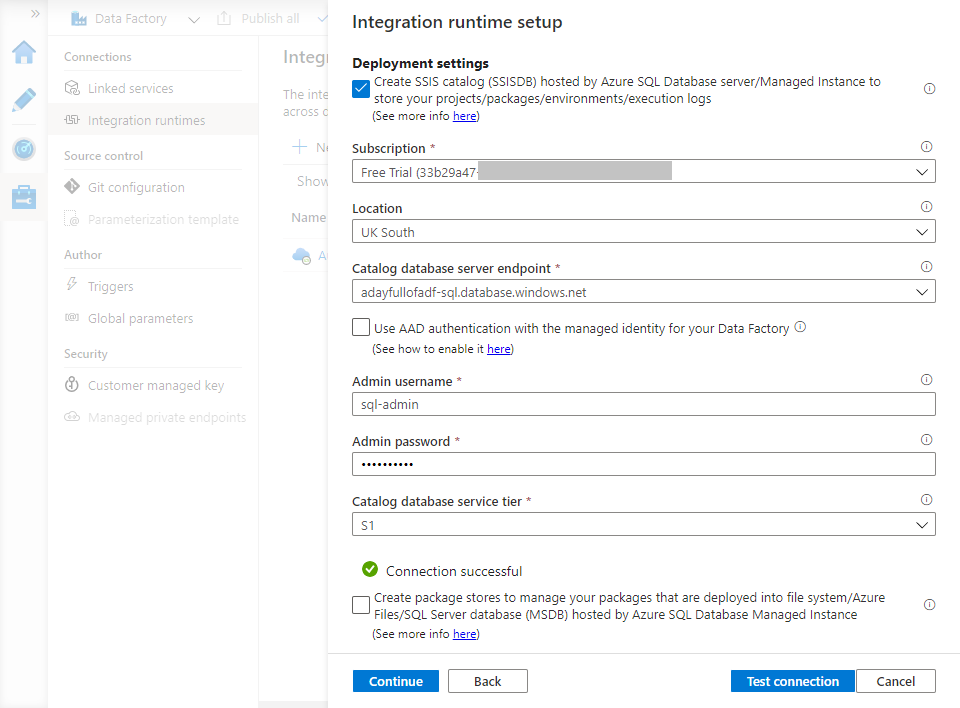


1. In step 1 of the wizard, provide general settings for the IR:
   * Give it a name
   * Choose the same location as your data factory
   * Choose the **smallest** option from the “Node size” list. This is just to keep the cost of the lab exercise down – in production environments you will want a node 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.
   * Accept the other option defaults and click “Continue”.

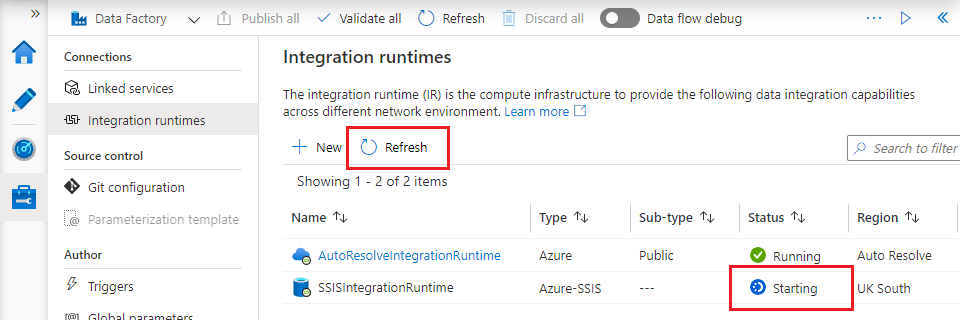


1. In step 2 of the wizard, configure a new SSIS catalog. This needs a SQL server to host the associated SSISDB database – use the server you created in lab 2.1.
   * Make sure that the “Create SSIS catalog (SSISDB)…” checkbox is ticked.
   * Choose your subscription and the same location you have been 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 of lab 2.1).
   * Enter your SQL admin username and password, and accept other default settings.

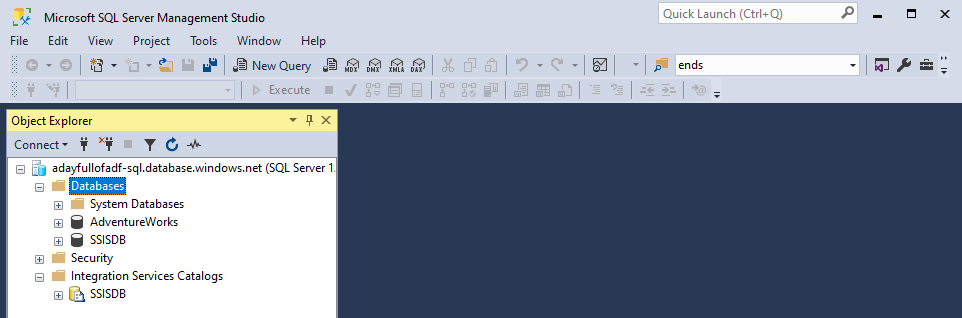
Click “Test connection” to verify that you have configured the SQL server and credentials correctly. When the test succeeds, click Continue.



1. Step 3 of the wizard provides advance configuration settings – click “Continue” to accept the defaults and move on.
2. 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”.



1. 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”.
2. 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.



**Note:** A running Azure-SSIS IR **costs money** – you are charged for the time it 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 portal or programmatically [using ADF’s Web activity](https://docs.microsoft.com/en-us/azure/data-factory/how-to-schedule-azure-ssis-integration-runtime). Leave it running for this lab, but remember to stop it when you finish.

# Lab 3.2 – Deploy SSIS packages

For this lab you will need:

* Visual Studio 2019
* The [SQL Server Integration Services Projects extension](SQL%20Server%20Integration%20Services%20Projects) for VS 2019
* The [Azure Feature Pack for Integration Services](https://docs.microsoft.com/en-us/sql/integration-services/azure-feature-pack-for-integration-services-ssis). 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.

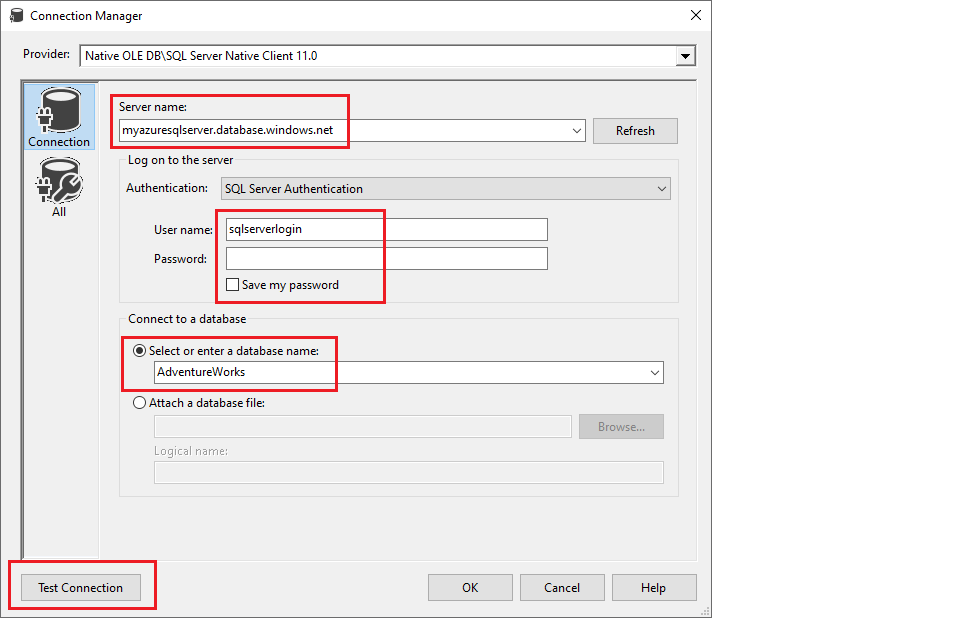
After installing the necessary VS components:

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: “CopySalesOrderDetail.dtsx”. Open it.

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

1. The “CopySalesOrderDetail.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.



1. Open the “” connection manager

Edit connection managers

* AzureSqlDatabase: Server name, User name, Password (+DB if you called it something different). test connection.
* AzureDataLakeStorage

Right click project, deploy. Wizard:

* Overview
* Choose “SSIS in Azure Data Factory”, click “Next
* Enter server name, login, password, click “Connect”. Path un-greys.
* Browse for Folder or Project dialog – New folder…
* Path populates, click “Next”.
* Review, click “Deploy”. SSIS IR must be runnibg!
* Close

Can see deployed package in catalog

# Lab 3.3 – Run the SSIS package in ADF

1. New pipeline
2. Add activity
3. Settings
   1. Azure-SSIS IR
   2. Package location pre-selected. Click Refersj
   3. Choose folder, project, package
4. Debug, look in lake

# Lab 3.4 – Stop the integration runtime

Stop SSIS IR!

Running -> Stopping -> Stopped