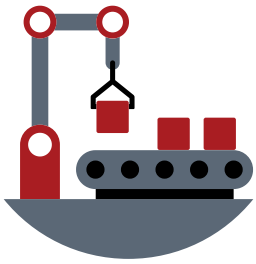
Executing SSIS Packages in Azure Data Factory

This post is part 17 of 25 in the series Beginner's Guide to Azure Data Factory

Two posts ago, we looked at the three types of integration runtimes and created an Azure integration runtime. In the previous post, we created a self-hosted integration runtime for copying SQL Server data. In this post, we will complete the integration runtime part of the series. We will look at what SSIS Lift and Shift is, how to create an Azure-SSIS integration runtime, and how you can start executing SSIS packages in Azure Data Factory.

(*And if you don't work with SSIS, today is an excellent day to take a break from this series. Go do something fun! Like eat some ice cream. I'm totally going to eat ice cream after publishing this post* 🙃)

What does SSIS Lift and Shift mean?



SSIS Lift and Shift means lifting up existing SSIS packages and shifting them to a new location. Basically, moving your projects from on-premises to Azure.

But why would I want to do that?

Well, you could have many reasons, but I'm guessing that the main reason is because you want to migrate to the cloud. It could be because you want to start modernizing your solutions, because you want to get rid of the hardware you're currently paying for, or because you want to reduce maintenance and costs.

But can't I just use a virtual machine running SQL Server?

Sure! But wouldn't it be *nice* to not have to manage that infrastructure yourself? 😎 If your goal is simply to migrate to the cloud, you can absolutely do that by moving from physical servers on-premises to virtual servers in Azure.

However, one of the main reasons to lift and shift your SSIS projects is that you won't *have* to think about SQL Server maintenance, patching, and so on. You can let Microsoft take care of all of that, and you can focus on building good solutions and deliver as much business value as possible.

But what if I use third-party components?

No problem! You can customize the setup to install third-party components.

But can't I just start over in Azure Data Factory?

You most certainly can! But that might not be the best use of your time and resources. If you already have complex projects with thousands of packages, or you have invested heavily in a Biml framework, or you have a large team of brilliant SSIS developers, it might not make sense to *start over*.

In that case, SSIS lift and shift is an excellent strategy. You can continue to use familiar tools, like Visual Studio and SQL Server Management Studio. You can continue to use familiar processes, like generating SSIS packages using Biml. The only thing you need to change is where you deploy your SSIS projects and how to orchestrate them.

Ok, I'm convinced! How does SSIS lift and shift work?

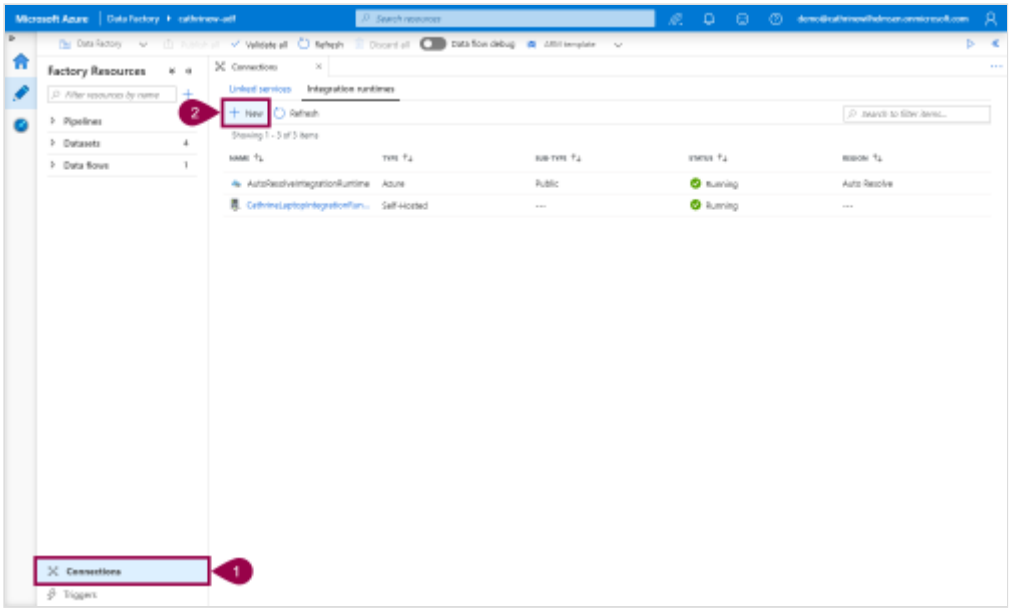
There are three main steps:

1. Creating an Azure-SSIS Integration Runtime
2. Deploying SSIS Packages to SSISDB in Azure
3. Executing SSIS Packages in Azure Data Factory

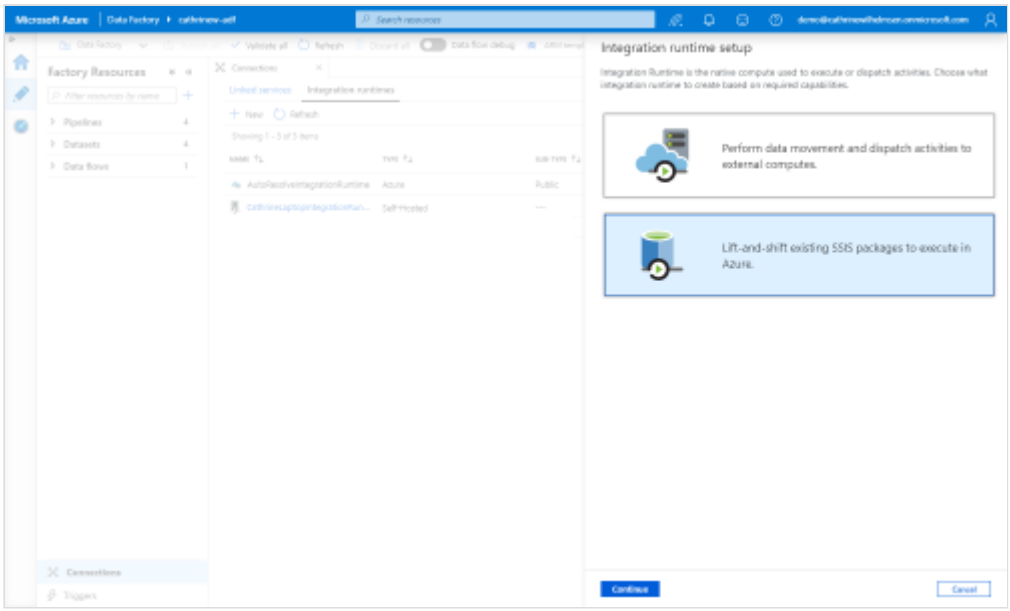
Let's walk through them!

Creating an Azure-SSIS Integration Runtime

Open connections, click on integration runtimes, then click **+ new**:



Select "**lift-and-shift existing SSIS packages**":

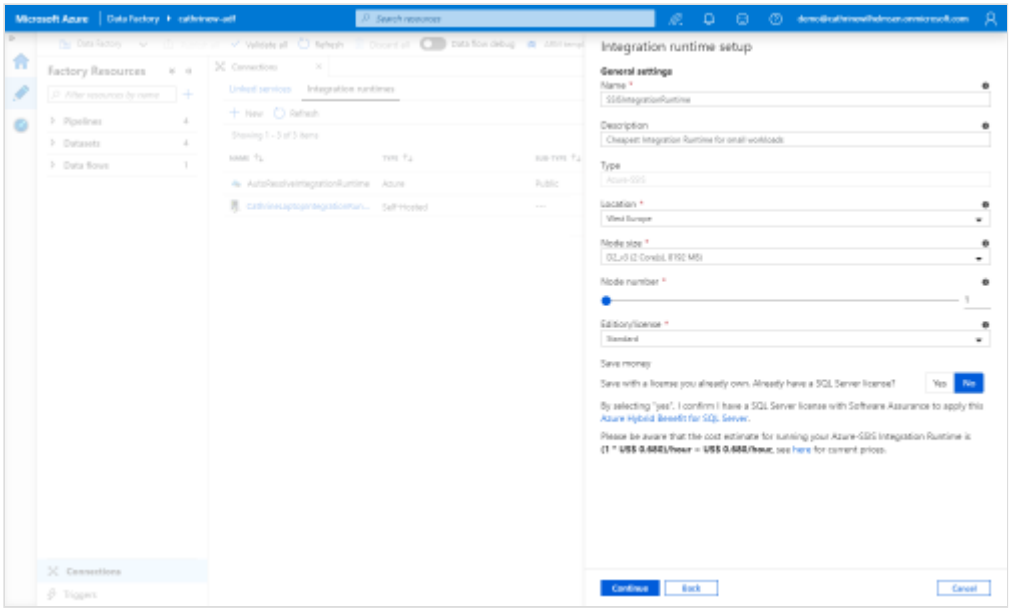


Next, you will go through three pages of settings 😄

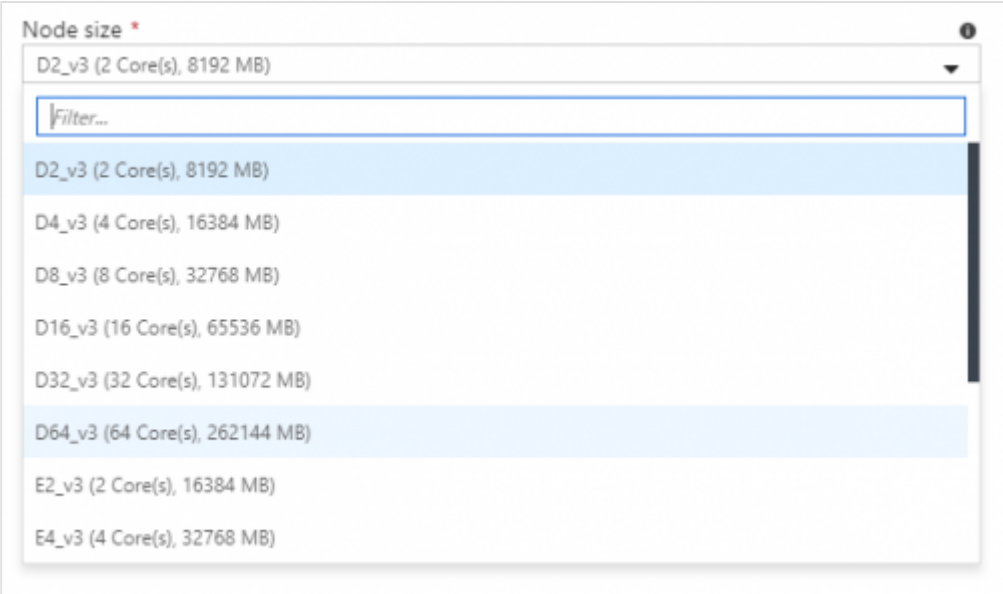
cathrine adf biml speaking Search...

On the first page, you will configure the *general settings*.

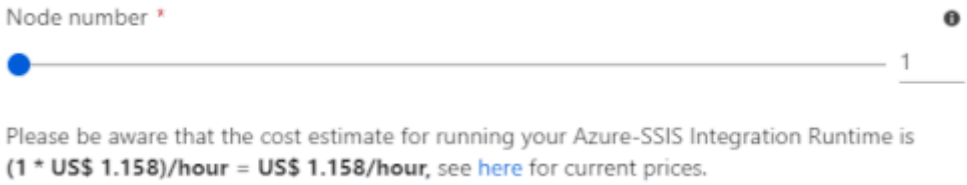
Give the new integration runtime a **name**, **description**, and choose the **location**. Then adjust the **node size**, **node number**, **SQL Server edition**, and **SQL Server license**:



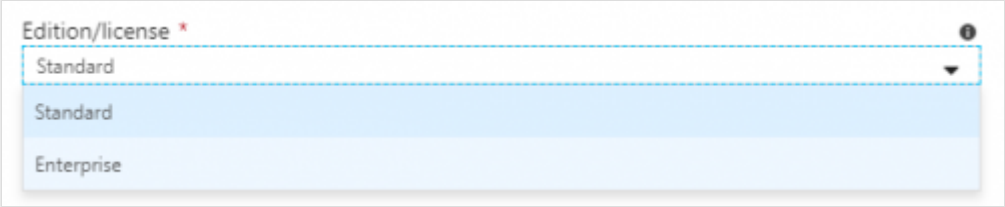
Node size is the *size of the virtual machine* running the SSIS engine. The more power you need, the more you have to pay:



Node number is the *number of virtual machines* running the SSIS engine. The more virtual machines you need, the more you have to pay. It's kind of fun playing with this slider and watching the cost estimate change :D



If you need enterprise features, you can change the **SQL Server edition**. The higher the edition you need, the more you have to pay:



But woohoo! You can also save money :D That is, if you have already paid money for a **SQL Server license**. If you have, make sure you read up on the Azure Hybrid Benefit for SQL Server:

Depending on your configuration and performance needs, you can create an Azure-SSIS integration runtime that's fairly cheap per hour or fairly expensive per hour:

SQL Settings

On the second page, you will configure the *SQL settings* and create the new SSISDB. If you already have an existing SSISB, you can uncheck this and move to the next page.

Choose the **server**, the **admin username and password**, and the **database service tier**.

(I chose a *Basic* database to keep the costs down.)

Advanced Settings

On the third page, you will configure the *advanced settings*:

Choose the number of **maximum parallel executions per node**. This is the number of packages that can run in parallel. The number you see will depend on how you configured the *general settings*. For example, you can only choose 64 parallel executions if you choose a larger node size:

Customize the setup of the integration runtime if you need to install **third-party components** or configure the environment. Read all the details in the official documentation:

You can **select a VNet** to join if you need access to on-premises data sources. Read all the details in the official documentation:

If you don't use a VNet, but still need access to on-premises data sources, you can **set up a self-hosted integration runtime as a proxy**. Read all the details in the official documentation:

Summary

On the *summary* page, you can review all the settings. And while it's tempting to just click *create* at this point, make sure you read it all. It has some really useful tips in there. Especially the part about managing cost ;)

Once you click *create*, the Azure-SSIS integration runtime will be created *and* started.

> **The integration runtime will be created *and* started.**
>
> That means that you will **start paying for the usage**.
>
> That means that if you are not going to start deploying or orchestrating SSIS packages *right away*, you should **stop the integration runtime**.

Did I make that clear enough? :D

Alright, *just in case* I wasn't clear enough: You really, really, really should stop your integration runtime when it's not being used.

Here's why:

https://twitter.com/cathrinew/status/1116291480623173632

You can stop the integration runtime by clicking the **stop button** (that looks like a *pause* button :D)

If you *are* going to use the integration runtime, leave it running. But please, *please*, **please** remember to stop it once you're done ;)

Righto! Let's move on.

Deploying SSIS Packages to SSISDB in Azure

Since this is a series on Azure Data Factory and not SSIS, I'm not going to go into the details of deploying SSIS packages to SSISDB, or how to configure your SSIS catalog. You can read all about that in the official documentation :)

But! I will call out a few things.

I deployed my project from Visual Studio. In the deployment wizard, you get the option to deploy to SSIS in Azure Data Factory:

Connect to the same server that you specified while creating the integration runtime, or to your existing SSISDB in Azure:

After deploying the project, you can connect to your server and browse the new SSISDB. You can only browse this as long as the integration runtime is running:

Ok, last part!

Executing SSIS Packages in Azure Data Factory

Now that we have deployed our SSIS project to Azure, we can start executing SSIS packages in Azure Data Factory! We can use the **execute SSIS package** task.

Use the dropdowns to select the **folder**, **project**, and **package**. If you use environments, you can also specify which environment to use:

(*For more information about all the settings, read the official documentation.*)

Debug the task:

Tadaaa! You have now executed an SSIS package from Azure Data Factory :D

Now… go and stop your integration runtime. I'll wait ;)

Summary

In this post, we completed the integration runtime part of this series. We looked at what SSIS Lift and Shift is, how to create an Azure-SSIS integration runtime, and how you can start executing SSIS packages in Azure Data Factory.

In the next post, we will take a step back from Azure Data Factory development and look at how to make life easier for ourselves… We will set up source control!

(*P.S. Did you remember to stop your Azure-SSIS integration runtime?* )

← Copy SQL Server Data in Azure Data Factory

Source Control in Azure Data Factory →

Share?

Twitter LinkedIn Email More

Related

Integration Runtimes in Azure Data Factory

Overview of Azure Data Factory Components

Copy SQL Server Data in Azure Data Factory

Dec 17, 2019 Data Platform Azure Data Factory

About the Author

Cathrine Wilhelmsen is a Microsoft Data Platform MVP, BimlHero Certified Expert, Microsoft Certified Solutions Expert, international speaker, author, blogger, and chronic volunteer who loves teaching and sharing knowledge. She works as a Senior Business Intelligence Consultant at Inmeta, focusing on Azure Data and the Microsoft Data Platform. She loves sci-fi, chocolate, coffee, craft beers, ciders, cat gifs and smilies :)

Find me!

Subscribe to new posts?

E-mail

Yes, subscribe me!

Recent Posts

Sneaking back in… (2020 edition)

Keyboard shortcuts for moving text lines and windows (T-SQL Tuesday #123)

Speaking at NIC 2020

Azure Data Factory Training Day at SQLBits 2020

Personal Highlights from 2019

Popular Posts

Table Partitioning in SQL Server - The Basics

Parameters in Azure Data Factory

Preparing for and Taking Microsoft Exam DP-200 (Implementing an Azure Data Solution)

Table Partitioning in SQL Server - Partition Switching

Custom Power BI Themes: Page Background Images

Top Tags

Azure Data Factory Biml Certifications Don't Repeat Yourself Microsoft Ignite Notepad++ PASS Summit Personal Precon Speaking SQLBits SQLFamily SQLHangout SQLSatOslo SQLSaturday SQL Server SSIS T-SQL Tuesday Volunteering Webinar

All Categories

Select Category

Full Archive

Select Month