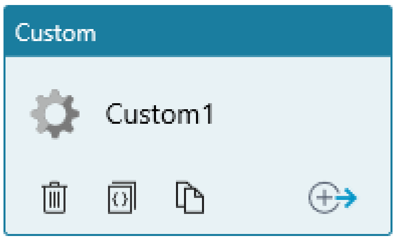
WELCOME TO THE TECHNICAL COMMUNITY BLOG OF PAUL ANDREW

Data Platform Principal Consultant & Solution Architect

Creating an Azure Data Factory v2 Custom Activity

Posted on November 12, 2018July 15, 2020



(https://mrpaulandrew.files.wordpress.com/2018/11/customactivity.png)Many moons ago and in a previous job role I wrote a post for creating an Azure Data Factory v1 Custom Activity here (https://www.purplefrogsystems.com/paul/2016/11/creating-azure-data-factory-custom-activities/). According to Google Analytics this proved to be one of my most popular blog posts on that site. I therefore feel I need to do an update post with the same information for Azure Data Factory (ADF) v2, especially given how this extensibility feature has changed and is implemented in a slightly different way to v1.

If you haven't already been through the Microsoft documents page I would recommend you do so before or after reading the below. It has a great comparison table near the bottom showing ADFv1 vs ADFv2 changes for the Custom Activity. Check it out here: https://docs.microsoft.com/en-us/azure/data-factory/transform-data-using-dotnet-custom-activity (https://docs.microsoft.com/en-us/azure/data-factory/transform-data-using-dotnet-custom-activity)

Let's move onto the none Microsoft version, my version and knowledge 🙂

What is an ADF Custom Activity?

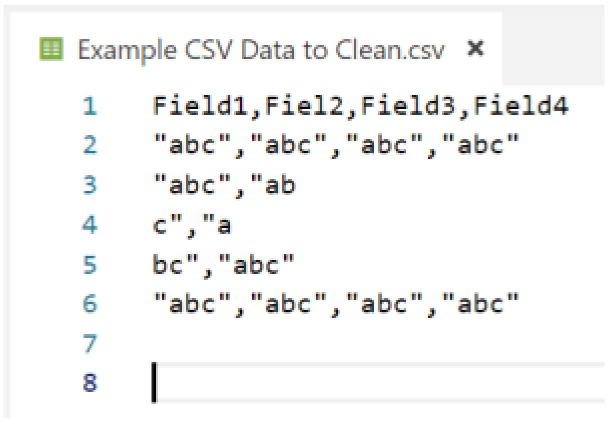
So very quickly, in case you don't know, an Azure Data Factory Custom Activity is simply a bespoke command or application created by you, in your preferred language and wrapped up in an Azure platform compute service that ADF can call as part of an orchestration pipeline. If you want to draw a comparison to SSIS, you can think of an ADF Custom Activity as a Script Task within an SSIS Control Flow.

For the ADF example in this post I'm going to talk about creating a Custom Activity using:

- a console application
- developed in Visual Studio
- coded in C#
- and executed on Windows virtual machines

I hope this is a fairly common approach. If not, things can be switched around as you prefer and I'll highlight alternatives as we go through.

Why do we need an ADF Custom Activity?



Example CSV Data to Clean.csv

```
Field1,Fiel2,Field3,Field4
"abc","abc","abc","abc"
"abc","ab
c","a
bc","abc"
"abc","abc","abc","abc"


```

(https://mrpaulandrew.files.wordpress.com/2018/11/csv-to-clean.png)Typically in our data platform, data analytics or data science solution an element of data cleaning and preparation is required. Sadly, raw source data isn't perfect. My go-to example here is a CSV file, text qualified, but it contains an attribute of free text that some lovely frontend user has added carriage returns to when typing in notes or whatever. These line feeds, as we know, break our CSV structure meaning schema-on-read transformation tools struggle to read the file.

So, given the last paragraph. Why do we need an ADF Custom Activity? To prepare our data using transformation tools that don't require a redefined data structure.

Of course, we can use a Custom Activity for 9000 other reasons as well. We can write some C# to do anything! Or, to answer the question another way. We need an ADF Custom Activity to perform an operation within our orchestration pipeline that isn't yet supported nativity by our ADF Activities.

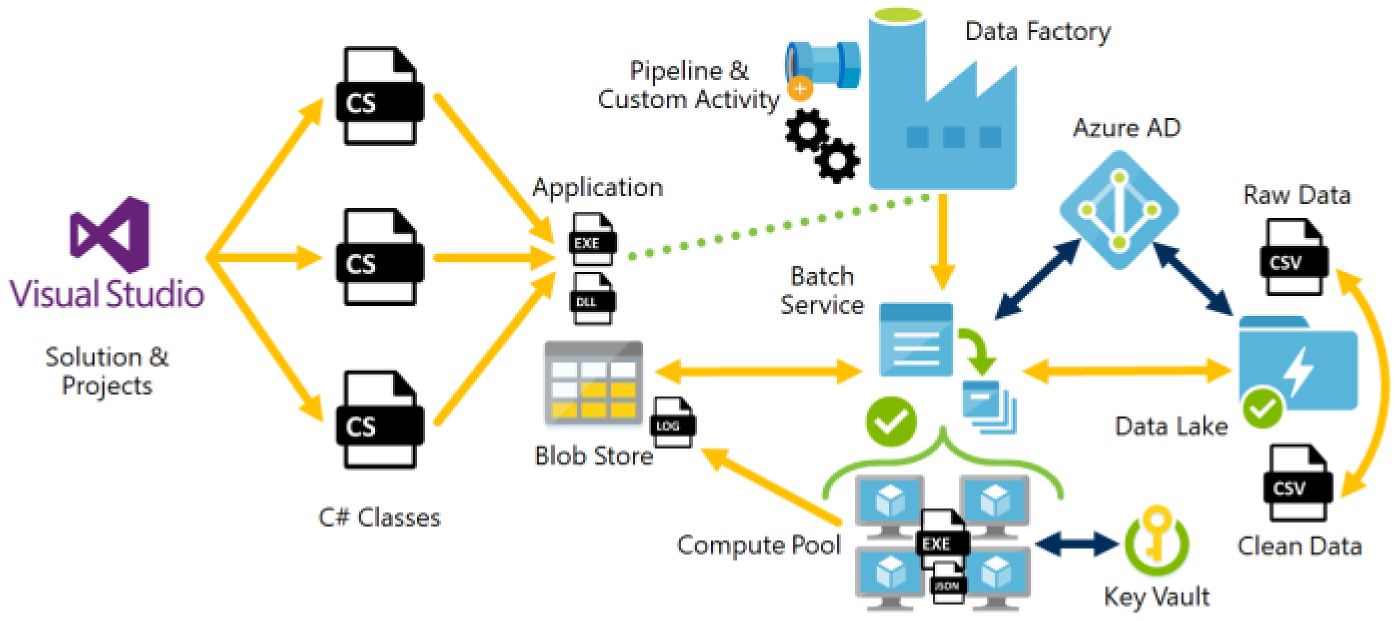
(https://mrpaulandrew.files.wordpress.com/2018/11/function-icon.png)Next, you may think why do we need a Custom Activity for the data preparation example and not handle this with some custom code wrapped up in an Azure Function instead? Well, an Azure Function can only run for 10 minutes and won't scale out in the same way our custom activity will. Simple rules:

- **Azure Function** – quick execution, no scale out abilities.
- **Custom Activity** – long running (if needed), ability to scale out automatically and scale up compute nodes if needed.

Great, why, done. Next.

How does an ADFv2 Custom Activity Work?

Let's start with the high level image I use in my community talks and add some narrative for all the arrows and services! Please do click and enlarge this image. Maybe even save it to view on a second monitor while we go through it.



(https://mrpaulandrew.files.wordpress.com/2018/11/adf-custom-activity.png)

In the image above moving from left to right this is what needs to happens and what we need to under to create our own Custom Activity:

1. We start with a **Visual Studio** solution containing a C# console application project. Or VB, or whatever. Just something that compiles to an executable.
2. Within the project we write our custom code and **classes** for whatever we need. For the file cleaning example above let's say we have an Azure Data Lake file handler class, some file cleaner class with methods for different cleaning operations and an authentication wrapper class for Azure Key Vault, I'll talk more about key vault later. Finally, we have a Program class where everything comes together.
3. Deploy an **Azure Blob Storage** account and create a container where our compiled code can live in the form of a master copy.
4. Next, we build our console application and from the Visual Studio project 'bin' folder we copy the **.exe** file and any NuGet/references **.dll** files into our Azure Blob Storage container. I recommend creating some PowerShell to handle this deployment from Visual Studio.
5. Deploy an **Azure Batch Service**. As Azure Data Factory can't execute the custom code directly with its own Integration Runtime we need something else to do it.
6. Within the Azure Batch Service create a **Compute Pool**. This gives you a pool of virtual machines that can be used in parallel to run our custom code. The pool can have one node or many. Windows or Linux. It can even be given a formula to auto scale out as requests for more tasks are made against the batch service.
7. For this walk through let's assume we have **Azure Data Lake Storage** already deployed with some raw poorly structured data in a CSV file.
8. Deploy an **Azure Data Factory** if you haven't already.
9. Within your data factory you'll need **linked services** to the blob storage, data lake storage, key vault and the batch service as a minimum. For this example.
10. Create an ADF **pipeline** and with a vanilla Custom Activity.
11. In the **Custom Activity** add the batch linked service. Then in settings add the name of your exe file and the resource linked service, which is your Azure Blob Storage. AKA the master copy of the exe.
12. Next, add **Reference Objects** from data factory that can be used at runtime by the Custom Activity console app.

13. We now have everything we need in place for the Custom Activity to function. Let's **trigger** the pipeline and think about the engineering that happens.
14. At runtime, Data Factory logs a **task request** with our Azure Batch Service onto its own internal queue.
15. The batch service copies the master version exe file from blob storage and setups up the task in a special **working directory** on an available compute node within the pool.
16. In addition to the exe file three **JSON** files are created and copied into the same working directory. These contain information about for reference objects set in the data factory Custom Activity.
17. The batch service triggers the console app to run, which executes on compute node virtual machine. If Windows, this can be seen in Task Manager.
18. As the code is running independently on the batch compute node VM it needs to handle its own authentication to other Azure services, in just the same way if the code was running locally on your desktop. This is where we need **Azure Key Vault.** Service principal credentials in the ADF reference objects are no longer shared with the batch service in pain text at runtime. As of September 2018 data factory won't do this.
19. The exe calls and authenticates against Azure Key Vault using its own **Azure Active Directory** service principal and requests a key to authenticate against the Azure Data Lake Store account, in this example.
20. The Raw data is **downloaded** to the batch compute node. A temporary directory will need to be defined in the application where the file can be safely downloaded to.
21. **Cleaning** processes are performed on the compute node using the local disk, RAM and CPU.
22. The cleaned data is **uploaded** back to Azure Data Lake Storage. Depending on the size of the file this may need to be done in chucks and concatenated at the target.
23. Once complete the batch node working directory is cleaned up. The exe and JSON reference object files get deleted. Your downloaded data won't be automatically deleted. You'll need to code that into your application.
24. **Standard out** and **standard error** log files are copied to the same blob storage account as the master exe but under a separate container called 'adfjobs' with subfolders for the Custom Activity run ID from ADF.
25. Data factory reports the state of the activity as successful. We hope.

With an understanding of the above let's now going even deeper into some of these steps, especially where things differ between ADFv1 and ADFv2.

The Console Application

(https://mrpaulandrew.files.wordpress.com/2018/11/dotnetmainorcore.png) (https://mrpaulandrew.files.wordpress.com/2018/11/azure-cli-icon.png)As mentioned above we can use C# or whatever to create our console application. The other consideration here is if you want to use Windows machines or Linux machines for the Batch Service compute pool. If Linux, then you'll need to develop your application using .Net Core rather than the full framework. Yes, the Linux VM's will be cheaper than Windows, but .Net Core has a limited set of libraries compared to the full fat version. It's a trade off you'll need to consider. For the rest of this post I'll be referring to things in the Batch Service assuming we used Windows compute nodes.

Deploying your Application with PowerShell

In point 4 above, a little bit of PowerShell can be helpful to get your compiled application up into Azure Blob Storage. At least in your development environment, maybe not for production. This isn't anything complex, just a copy of everything from your Visual Studio bin folder to a blobs store container.

I've but an example PowerShell script in the below GitHub repository. Please feel free to copy it. I'd normally have a PowerShell helper project in my Visual Studio solution, or just add it directly to the console application project. In both cases the PowerShell extension will be needed to execute the script.

Code in Blog Support **GitHub** repository. (https://mrpaulandrew.files.wordpress.com/2018/11/github-icon.png) https://github.com/mrpaulandrew (https://github.com/mrpaulandrew)/BlobSupportingContent

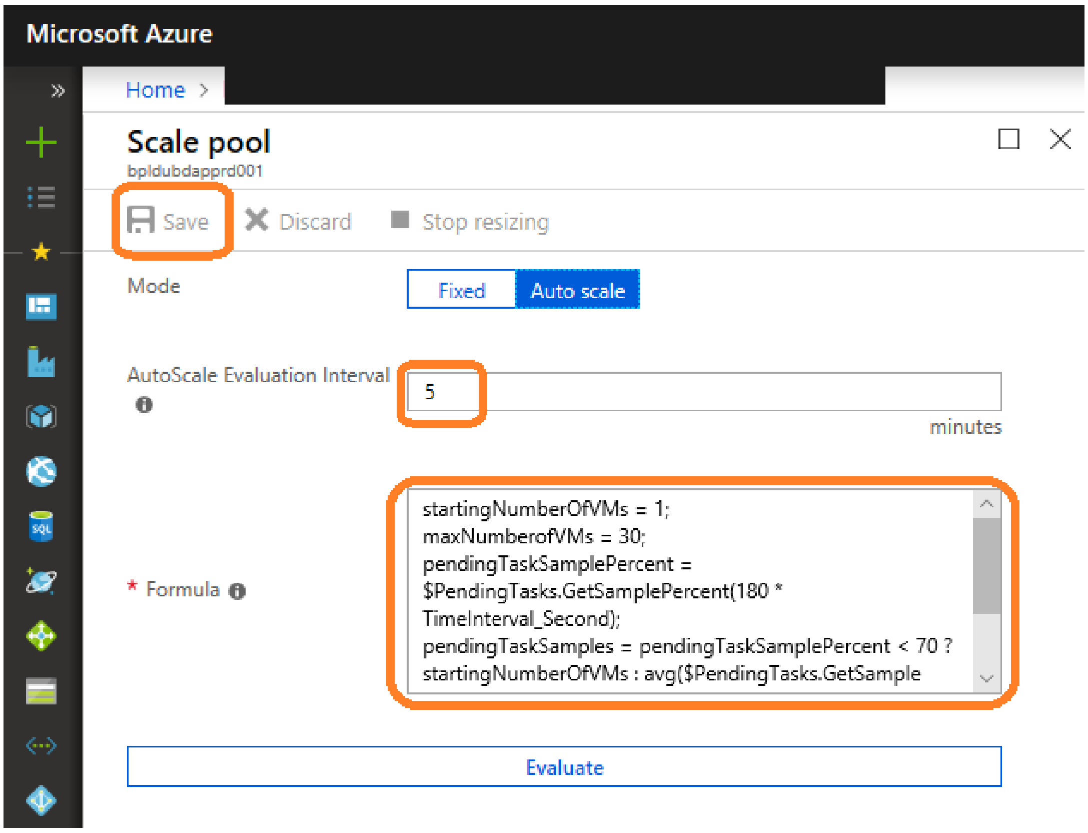
Auto Scaling the Compute Pool

We don't want to be paying for Azure VM's that aren't in use. Therefore, auto scaling the compute pool is a must in production. Microsoft have a great document here (https://docs.microsoft.com/en-us/azure/batch/batch-automatic-scaling) on creating the formula that decides on how scaling occurs. The example formula is a good starting point:

```
startingNumberOfVMs = 1;
maxNumberofVMs = 25;
pendingTaskSamplePercent = $PendingTasks.GetSamplePercent(180 * TimeInterval_Se
pendingTaskSamples = pendingTaskSamplePercent < 70 ? startingNumberOfVMs : avg(
$TargetDedicatedNodes=min(maxNumberofVMs, pendingTaskSamples);
```

Code in Blog Support **GitHub** repository. (https://mrpaulandrew.files.wordpress.com/2018/11/github-icon.png) https://github.com/mrpaulandrew (https://github.com/mrpaulandrew)/BlobSupportingContent

To apply the formula simply go to your **Azure Batch Service** > **Pools** >select the pool > **Scale** > **Auto Scale** > add your formula > **Save**.

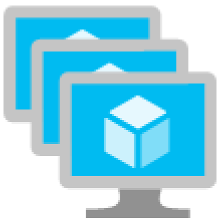


AutoScale Evaluation Interval: 5 minutes

```
startingNumberOfVMs = 1;
maxNumberofVMs = 30;
pendingTaskSamplePercent =
$PendingTasks.GetSamplePercent(180 *
TimeInterval_Second);
pendingTaskSamples = pendingTaskSamplePercent < 70 ?
startingNumberOfVMs : avg($PendingTasks.GetSample
```

The only down side to this option is you can only get the formula to evaluate on a minimum frequency of 5 minutes. This means tasks won't execute as fast compared to a fixed pool size that is always available. This is of course a common pattern with any Azure using a cluster of compute… If you want instant processing, pay for it to be on all the time.

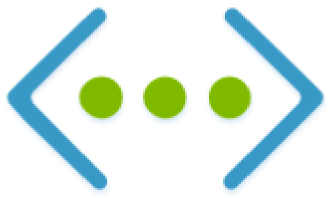
Compute Nodes



(https://mrpaulandrew.files.wordpress.com/2018/11/vm-scale-set-icon.png)In point 15 I mentioned a special working directory. If you RDP to a one of the compute nodes this can be found in at the following location:

C:\user\tasks\workitems\adf-{guid}\job-0000000001\{guid}-{activityname}-\wd

Bear in mind that the C drive is not where the operating system lives. These nodes have a none standard Windows image and special environment variables listed here (https://docs.microsoft.com/en-us/azure/batch/batch-compute-node-environment-variables). That said, custom images can be used if you want to have software preloaded onto a compute node. For example if you data cleaning custom application is really large or has multiple versions it could be loaded into a VM image used when the compute pool is created. Maybe a topic for another post.



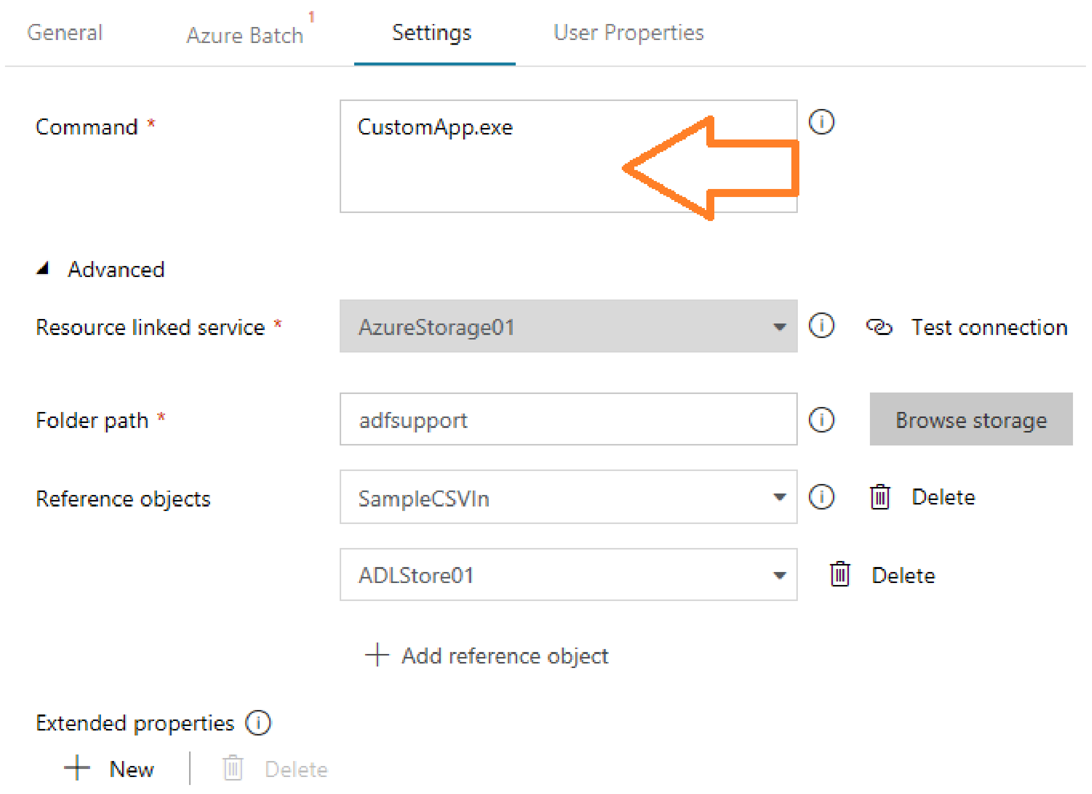
(https://mrpaulandrew.files.wordpress.com/2018/11/vnet-icon.png)Lastly, consider my comment in point 20 about the data. This is downloaded onto the batch service compute nodes for processing and as mentioned we can RDP to them. By default this will be done using a dynamic public IP addresses. I therefore recommend adding the compute pool to an Azure Virtual Network (VNet) meaning access can only be gained via a private network and local address.

If you do decide to attach the compute nodes to an Azure VNet please also consider the amount of addresses available on your chosen SubNet when using the auto scaling pool option mentioned above. It might sound obvious now, but I've seen pools fail to scale because there wasn't enough IP addresses available. #AzureFeatures

The Custom Activity

Now let's think about Azure Data Factory briefly, as it's the main reason for the post 🙂

In version 1 we needed to reference a namespace, class and method to call at runtime. Now in ADF version 2 we can pass a command to the VM compute node, settings screen shot for the ADF developer portal below.



Command * CustomApp.exe

Advanced

Resource linked service * AzureStorage01

Folder path * adfsupport

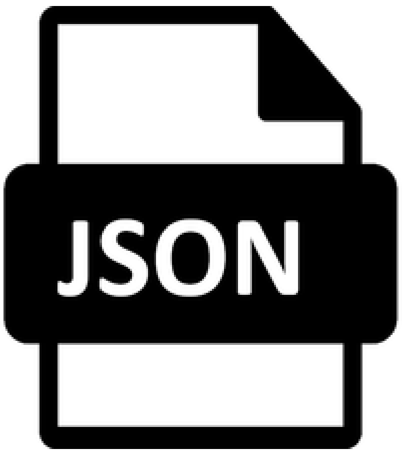
Reference objects SampleCSVIn, ADLStore01

Extended properties

(https://mrpaulandrew.files.wordpress.com/2018/11/adf-custom-activity-settings.png)

This command can be any Windows command. We can change directory and do something else on the VM or whatever, especially if using a custom image for the VM's. Very handy and much more flexible than version 1.

Using the Reference Objects



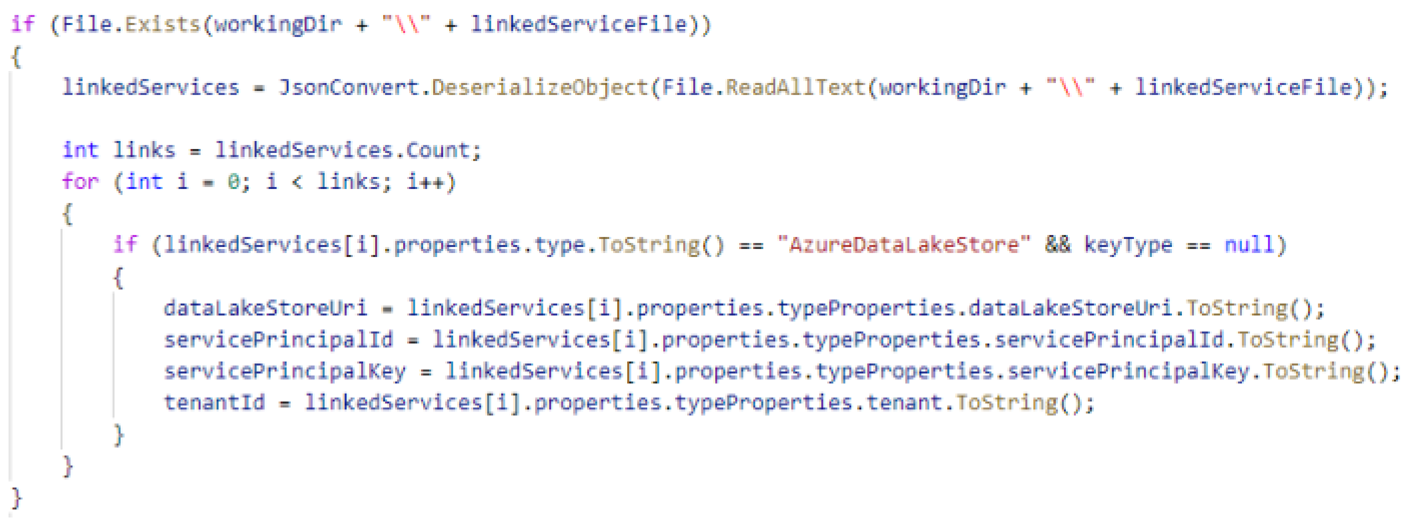
(https://mrpaulandrew.files.wordpress.com/2018/11/jsonfile.png)Now addressing points 12 and 16 above let's focus on our data factory reference objects (seen in the above screen shot as well). This is probably the biggest difference to understand between versions of Azure Data Factory. In version 2, at runtime three JSON files are copied alongside our exe console application and also put into the compute node working directory. These are named:

- activity.json
- datasets.json
- linkedServices.json

There will only ever be three of these JSON files regardless of how many reference objects are added in the data factory settings. To further clarify; if we reference 4 linked services, they will be added to the single JSON reference objects file in an array. The array will have a copy of the JSON used by Data Factory when it calls the linked service in a pipeline.

This gets worse with datasets, especially if there is dynamic content being passed from data factory variables. The dataset name within the reference file will get suffixed with a GUID at runtime. This is nice and safe for the JSON file created on the compute node, but becomes a little more tricky when we want to use the content in our custom application and it has a random name! For example we get this: **AdventureWorksTable81e2671555fc4bc8b8b3955c4581e065**

To access the reference objects we need to code this into our console app, parse the JSON and figure out which key value pairs we want to use. This can get very messy and my C# skills mean I don't yet have a clean way of doing this. Here's an example of how we might find details of our Data Lake Storage account from the linked service reference objects file.



```
if (File.Exists(workingDir + "\\" + linkedServiceFile))
{
    linkedServices = JsonConvert.DeserializeObject(File.ReadAllText(workingDir + "\\" + linkedServiceFile));

    int links = linkedServices.Count;
    for (int i = 0; i < links; i++)
    {
        if (linkedServices[i].properties.type.ToString() == "AzureDataLakeStore" && keyType == null)
        {
            dataLakeStoreUri = linkedServices[i].properties.typeProperties.dataLakeStoreUri.ToString();
            servicePrincipalId = linkedServices[i].properties.typeProperties.servicePrincipalId.ToString();
            servicePrincipalKey = linkedServices[i].properties.typeProperties.servicePrincipalKey.ToString();
            tenantId = linkedServices[i].properties.typeProperties.tenant.ToString();
        }
    }
}
```

(https://mrpaulandrew.files.wordpress.com/2018/11/csharp-parse-adf-ref-object.png)

Code in Blog Support **GitHub** repository.
(https://mrpaulandrew.files.wordpress.com/2018/11/github-icon.png)
https://github.com/mrpaulandrew
(https://github.com/mrpaulandrew)/BlobSupportingContent

I'd be more than happy to accept some C# help on coming up with a better way to traverse these 'Reference Object' files.

Azure Key Vault

(https://mrpaulandrew.files.wordpress.com/2018/11/key-vault-icon.png)With Azure Key Vault if you haven't used it already there is plenty of documentation and examples on working with the SDK, link below. Security is important so Microsoft give us lots of help here.

https://docs.microsoft.com/en-us/azure/key-vault/key-vault-developers-guide
(https://docs.microsoft.com/en-us/azure/key-vault/key-vault-developers-guide)

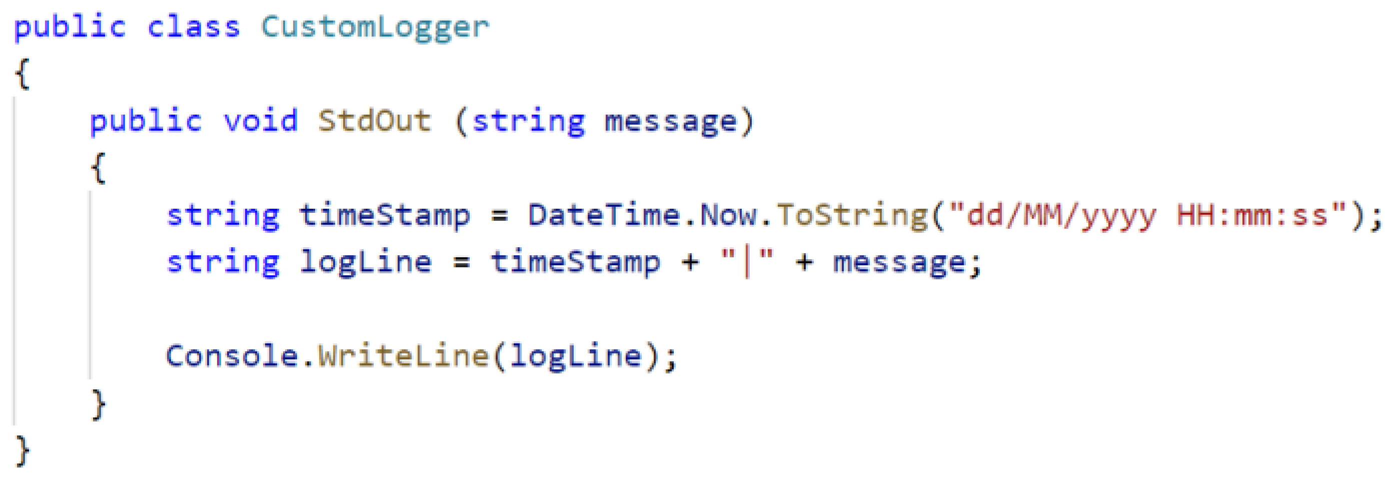
This extra layer of keys being passed around does initially seem a little overkill, but when keys get recycled having a central location is the best thing ever. Do also make the effort to create a comprehensive key vault utilities class that works for you within the console app.

To implement this in an Azure Data Factory Custom activity you'll need to have Key Vault added as its own linked service. Then add the linked service as one of the Reference Objects. When parsed in your app this will give you the URL to reach the Key Vault API. Or, feel free to hard code in the application config, it depends if you plan to change Key Vaults. With the Key Vault URL and the console apps own service principal you'll be able to hit the API and request your Data Lake Store secret, for example.

Logging

If you have 30+ compute nodes in your Batch Service pool creating an RDP connection to each node doesn't really work when trying to check log files. Therefore, as mentioned in point 24 above your console application standard error and standard out log files get copied into Azure Blob Storage. The container created for you called 'adfjobs' will have a sub per data factory activity execution and named with the GUID from the activity run ID. Simply grab the GUID from your Data Factory monitoring and use something like Azure Storage Explorer to search the adjob container.

I also like to extend the normal C# Console.WriteLine output with a little more formatting and date/time information. For example:



```
public class CustomLogger
{
    public void StdOut (string message)
    {
        string timeStamp = DateTime.Now.ToString("dd/MM/yyyy HH:mm:ss");
        string logLine = timeStamp + "|" + message;

        Console.WriteLine(logLine);
    }
}
```

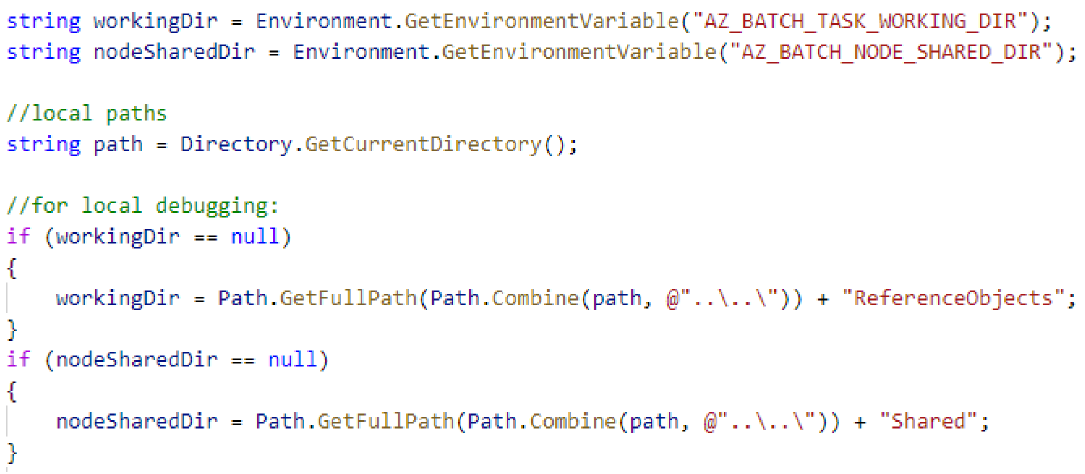
(https://mrpaulandrew.files.wordpress.com/2018/11/adf-custom-logger.png)

Code in Blog Support **GitHub** repository. (https://mrpaulandrew.files.wordpress.com/2018/11/github-icon.png) https://github.com/mrpaulandrew (https://github.com/mrpaulandrew)/BlobSupportingContent

Debugging

Debugging your custom activity just became super easy in ADFv2, mainly because we are using a console application that can be ran locally in Visual Studio with all the usual local debugging perks. The only minor change I recommend you make is imitating the ADF Reference Objects. We of course don't get a copy of these locally when running the application in Visual Studio. An easy way to handle this is by checking for the Batch Service compute node environment variables, which you won't have locally, unless you want to create them. When missing or with NULL values, use copies of the Reference Object JSON files saved into your Visual Studio console application project. You can get them by remoting onto a compute node and copying them out while the Custom Activity is running if you don't want to recreate them by hand. Simples 🙂

Here is those last few sentences in C# form:



```
string workingDir = Environment.GetEnvironmentVariable("AZ_BATCH_TASK_WORKING_DIR");
string nodeSharedDir = Environment.GetEnvironmentVariable("AZ_BATCH_NODE_SHARED_DIR");

//local paths
string path = Directory.GetCurrentDirectory();

//for local debugging:
if (workingDir == null)
{
    workingDir = Path.GetFullPath(Path.Combine(path, @"..\..\")) + "ReferenceObjects";
}
if (nodeSharedDir == null)
{
    nodeSharedDir = Path.GetFullPath(Path.Combine(path, @"..\..\")) + "Shared";
}
```

(https://mrpaulandrew.files.wordpress.com/2018/11/csharp-local-ref-object.png)

Code in Blog Support **GitHub** repository.
(https://mrpaulandrew.files.wordpress.com/2018/11/github-icon.png)
https://github.com/mrpaulandrew
(https://github.com/mrpaulandrew)/BlobSupportingContent

I think that's everything you need! I hope with the above knowledge and links your able to easily understand and create a complete Azure Data Factory v2 Custom Activity.

Many thanks for reading.

Tagged ADFv2, Azure Data Factory, Custom Activity, Dynamic ADF Pipelines

Published by mrpaulandrew

Principal consultant and architect specialising in big data solutions on the Microsoft Azure cloud platform. Data engineering competencies include Azure Data Factory, Data Lake, Databricks, Stream Analytics, Event Hub, IoT Hub, Functions, Automation, Logic Apps and of course the complete SQL Server business intelligence stack. Many years' experience working within healthcare, retail and gaming verticals delivering analytics using industry leading methods and technical design patterns. STEM ambassador and very active member of the data platform community delivering training and technical sessions at conferences both nationally and internationally. Father, husband, swimmer, cyclist, runner, blood donor, geek, Lego and Star Wars fan!
View all posts by mrpaulandrew

19 thoughts on "Creating an Azure Data Factory v2 Custom Activity"

1.
Mehmet says:
January 8, 2019 at 2:33 pm
Hi,

We have an ADF pipeline that runs 3 other pipelines which run the same executable using the same Azure Batch pool. Most of the time the executable runs successfully. However, there are instances where the pipeline hangs before it starts running the executable.

Note that all 3 pipelines are using the same Azure Batch pool.

– Could you let us know what the cause of this hang may be?
– Is there an issue with the pipelines using the same Azure Batch pool? Should they be using different pools?
– The executable does not have any parallelism, so does the number of nodes make a difference?

Thanks

Mehmet

2. Reply
Imran says:
May 2, 2019 at 2:05 am
This is really amazing!
All the content, Thanks a lot for this!

3. Reply
anand says:
June 17, 2019 at 10:28 am
I am trying to get THE CUSTOM ACTIVITY screen where did i get that?

Reply
mrpaulandrew says:
June 18, 2019 at 10:14 am
Hi sorry, I don't understand the question. Which screen are you referring to? The activity settings? Thanks

4. Reply
Andrew Wilder says:
August 27, 2019 at 6:34 pm
Hello,

First off thanks for the detailed article, in particular the "how adfv2 custom activities work" section; I do still have a question though. The article says "With the Key Vault URL and the console apps own service principal you'll be able to hit the API and request your Data Lake Store secret", but the code example uses the service principal details from the ADLS account (not the app itself, or keyvault). In my testing, it had seemed that there isn't any service principal information embedded in activity.json or the keyvault reference object json. Is there a way you know of to securely authenticate with keyvault without using ADLS?

Thanks

5. Reply
Rajat says:
September 19, 2019 at 2:20 pm
Hi Paul,
I am working on a pipeline which copies files(binary) from AWS S3 storage to Azure blob (recursively & maintains hierarchy). The copy activity fails if a file is corrupted and the whole pipeline fails. Is there a way to make the copy activity skip any file if an error occurs during copying it, while continuing with other files in the folder/storage). The copy activity fault tolerance is of no help in case of copying binary files.
Also, do you know where can I get the copy activity code behind. I would need to replicate the copy activity but with a better exception handling (skipping and logging).

Thanks

Reply
mrpaulandrew says:
September 19, 2019 at 2:34 pm

Try using the Validate activity with an IF activity? Depending on how complex your datasets are.

Reply

6. **niec** says:
October 23, 2019 at 7:02 am
Hi Paul,
I am creating a ADF for updating data in Cosmos DB (Mongo API). My Source is csv file which has unique key and value. I have to insert this value in cosmos on the basis of key(key already exists in cosmos db which is unique)
I am doing this using pipeline ,but my old data is getting deleted and only new value is inserted in cosmosdb.
What i want is old_data+ new data in cosmos db.

Is there any way to write custom function which fetch data from cosmosdb using unique key(read from csv and fetch records from cosmos) and then do the update.

Thanks,

Reply

7. **Luoana** says:
May 13, 2020 at 3:53 pm
So glad you made this article, helped me in my discussion with a customer! Thank you very much, keep on posting 🙂

Reply

mrpaulandrew says:
May 13, 2020 at 3:55 pm
Your welcome 🙂

Reply

8. **Candide** says:
May 18, 2020 at 3:59 pm
Hi Paul,
Thanks a lot for your work and your interesting posts!
I was wondering, is it possible to access the files within the batch working directory directly from ADF. Basically, I have a custom activity which produces a custom file on the node pool. In a further step, I would like to process this file via ADF activities. However, I didn't find any way to create a dataset and point it to that file… Do you have an idea? The only thing I am thinking of at the moment, would be to write the file content to "stdout" and then access the content via ADF. This is however not sooo neat 🙂 Thanks!

Reply

mrpaulandrew says:
May 18, 2020 at 4:13 pm
Your custom app needs to write to output dataset to blob storage or data lake as part of the custom process.

Reply

Candide says:
May 18, 2020 at 4:31 pm
Ahh cool – that's my current implementation. But I thought this could have been done differently. Thanks a lot!! 🙂

Reply

9.

Manoj says:
July 1, 2020 at 11:26 am
HI
thanks for the post this was very helpful and informative. can we interact to onprem gateways (IR) from customer activities ?

Thanks in Adavnce,
Manoj

Reply

mrpaulandrew says:
July 1, 2020 at 11:50 am
Hi, not directly, however the batch compute pool could be added to an Azure VNet that then has access to an Express Route connection or VPN allowing the reach down to on prem resources. Cheers

Reply

10. **Sudhanshu dhyani** says:
July 31, 2020 at 5:00 pm
Hi Paul,
We have a similar approach for azure custom activity in our one of the solution that is implemented on azure datafactory version 1, long back in 2016.
Now in the batch account which is in use for linked service in adf for custom activity, we are getting azure advisor notification to upgrade the api version to make the batch services operational.
Whereas i have found special advisory for the user who are using azure batch linked service for custom activity can ignore the advisor notification. Special advisory given in below link.
https://docs.microsoft.com/en-us/rest/api/batchservice/batch-api-status#special-advisory-for-azure-data-factory-custom-activity-users
Could you please help to advise whether we can ignore this notification.
Api verison with batch pool is 2015.1.1.011
Thanks in advance.

Reply

mrpaulandrew says:
August 10, 2020 at 8:33 am
Hi, I suggest you clone the pipeline and repoint the custom activity to a new Batch Account as a simple migration path. Cheers

Reply

11. **Giampaolo** says:
August 14, 2020 at 5:39 pm
Hi Paul, nice article, congrats
is it possible to create a custom activity that makes an API call to get the token to pass to the REST connector? my requirement is to call a protected API and save the content into azure SQL using ADF
thanks for your help

Reply

mrpaulandrew says:
August 15, 2020 at 6:07 am

Hi, anything is possible 🙂
A couple of thoughts…
Maybe hit your API using a web activity first to get the token, then pass the output to your custom activity.
Maybe also land the returned custom activity data into blob storage before copying to the eventual SQLDB.
Basically, my advise it to avoid doing too much work in the custom activity. Break up the steps required.
Maybe even use an Azure Function instead of a custom activity if the data is small. It would be cheaper with an App Service consumption plan.
A few options and ideas.
Cheers

Reply