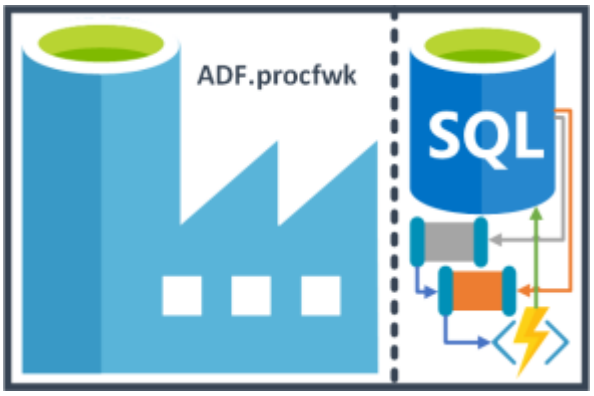
WELCOME TO THE TECHNICAL COMMUNITY BLOG OF PAUL ANDREW

Data Platform Principal Consultant & Solution Architect

Creating a Simple Staged Metadata Driven Processing Framework for Azure Data Factory Pipelines – Part 1 of 4

Posted on February 25, 2020July 15, 2020 (https://mrpaulandrew.files.wordpress.com/2020/02/adfprocfwk.png)In this four part blog series I want to share my approach to delivering a metadata driven processing framework in Azure Data Factory. This is very much version 1 of what is possible and where can we build on this to deliver a hardened solution for our platform orchestrator. For the agenda of this blog series I'm planning to break things down as follows:



1. Design, concepts, service coupling, caveats, problems. Post link (https://mrpaulandrew.com/2020/02/25/creating-a-simple-staged-metadata-driven-processing-framework-for-azure-data-factory-pipelines-part-1-of-4/).
2. Database build and metadata. Post link (https://mrpaulandrew.com/2020/03/02/creating-a-simple-staged-metadata-driven-processing-framework-for-azure-data-factory-pipelines-part-2-of-4/).
3. Data Factory build. Post link (https://mrpaulandrew.com/2020/03/05/creating-a-simple-staged-metadata-driven-processing-framework-for-azure-data-factory-pipelines-part-3-of-4/).
4. Execution, conclusions, enhancements. Post link (https://mrpaulandrew.com/2020/03/09/creating-a-simple-staged-metadata-driven-processing-framework-for-azure-data-factory-pipelines-part-4-of-4/).

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

Part 1

The concept of having a processing framework to manage our Data Platform solutions isn't a new one. However, overtime changes in the technology we use means the way we now deliver this orchestration has to change as well, especially in Azure. On that basis and using my favourite Azure

orchestration service; Azure Data Factory (ADF) I've created an alpha metadata driven framework that could be used to execute all our platform processes. Furthermore, at various community events I've talked about bootstrapping solutions with Azure Data Factory so now as a technical exercise I've rolled my own simple processing framework. Mainly, to understand how easily we can make it with the latest cloud tools and fully exploiting just how dynamic you can get a set of generational pipelines.

Design Thoughts/Questions

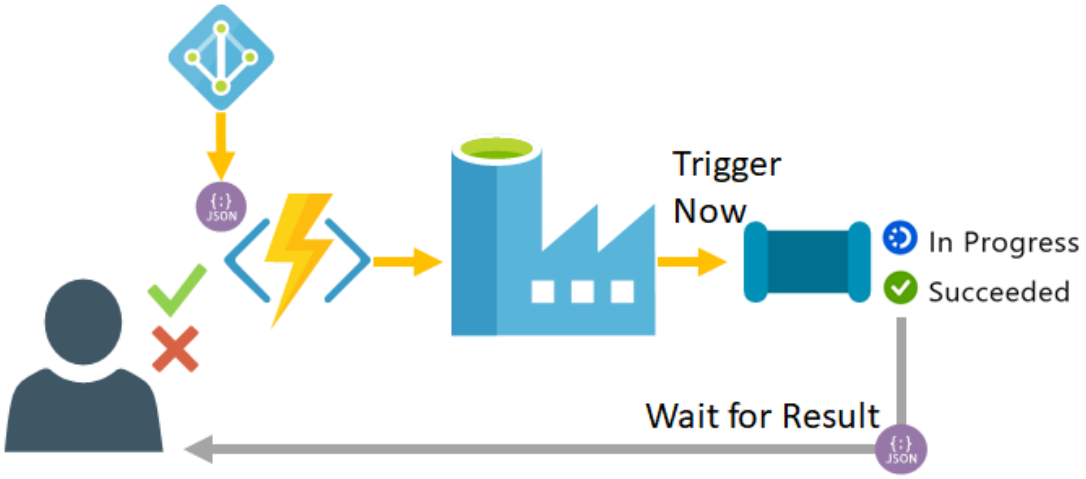
Leading up the build of such a framework a few key factors design decisions had to be thought about. I think its worth sharing these to begin with for context and because as the title suggests this is a simple staged approach to processing. Here's why:

1. **Questions;** How should the framework call the execution of the work at the lowest level? Or to ask the question in more practical terms for Azure Data Factory. Should we define a particular Data Factory Activity type in our metadata, then dynamically switch between different activities for the child pipeline calls? I did consider using the new Data Factory Switch activity for exactly this, but then decided against it (for now).

My Answer; To keep things simple the lowest level executor will be the call to another Data Factory pipeline, affectively using the Execute Pipeline Activity. Doing this means specifics of the actual work can be wrapped up using a tailored activity type, with tailored values (eg. Using a Databricks Notebooks Activity referencing a particular Linked Service and Workspace). Hitting a child pipeline as the lowest level of execution in the framework caller offers an easier abstraction over the actual work being done.

2. **(https://mrpaulandrew.files.wordpress.com/2020/02/execute-pipeline-1.png)Questions;** Given our understanding in point 1 of the child level call from the framework. How are we technically going to manage this? The Execute Pipeline Activity (https://docs.microsoft.com/en-us/azure/data-factory/control-flow-execute-pipeline-activity) in ADF does not allow dynamic values. The child pipeline referenced must be hard coded. Also, what if our solution orchestration needs to span multiple Data Factory resources?

My Answer; Use an Azure Function to execute any Data Factory Pipeline in any Data Factory. You may recall I created this in my last blog post (https://mrpaulandrew.com/2020/02/18/execute-any-azure-data-factory-pipeline-with-an-azure-function/) 😉



(https://mrpaulandrew.files.wordpress.com/2020/02/execute-adf-pipeline-with-azure-function.png)

3. **Questions;** What should be used to house our orchestration metadata? Would a set of JSON file artefacts be sufficient? Or, at the other end of the complexity spectrum, maybe we use the Gremlin API within an Azure Cosmos Database to create a node and edges based solution for connecting dependencies between processes.

My Answer; I settled on an Azure SQLDB as a middle point between a file and Cosmos. This also made development time easier as I do love a bit of T-SQL.

4. **Questions;** Building on point 3 relating to our up stream and down stream dependencies. How could/should these be handled/connected without resulting in a single threaded, sequential set of processes. Then, how can be make the metadata simple enough to edit without requiring a complete overhaul of many to many connections.

My Answer; I've introduced the concept of execution stages and child processes within a stage. As stated in point 1, the lowest level of execution will be done using ADF pipelines. Doing this means stages can execute sequentially to support dependencies. But child processes can execute in parallel to offer the ability of scaling out the lowest level executions. Therefore, child processes should have not inter-dependencies.

5. **Questions;** Should we be concerned about tightly coupling Azure Data Factory to an Azure SQLDB for metadata and an Azure Function App for child process execution? Is this good practice?

My Answer; Yes, I'm concerned. This isn't great, I'm fully aware. We could add others layer of abstraction into the framework, but for now keeping things simple is my preference. Also, in the medium term I really hope Microsoft will allow the Execute Pipeline Activity to be dynamic, avoiding the need to call from ADF to a Function and back.

6. **Questions;** How should the metadata be used, toggled and controlled at runtime?

My Answer; To offer a small amount of disconnect in the overall execution I'm copying how the SQL Server Agent behaves for a given run of tasks. If you aren't familiar with the SQL Agent, basically I'll take a static copy of the processing metadata and use this as a fixed set of things during runtime. This means any changes made to the metadata during processing won't compromise the execution run in progress. Finally, to address the control part of this question, bit fields will be added in the metadata for the stage level and child process level. Meaning anything can easily be disabled pre execution.

I think that is a good point to conclude the first part of this blog series. To recap:

- Idea established.
- Design work done.
- Concepts defined.
- Low level function executor already built.

In part 2 of **Creating a Simple Staged Metadata Driven Processing Framework for Azure Data Factory Pipelines** we'll build the database schema required to drive the orchestration framework.

Stay tuned.

Many thanks for reading.

Tagged Azure Data Factory, Azure Functions, Azure SQLDB, Dynamic ADF Pipelines, Processing Framework

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

15 thoughts on "Creating a Simple Staged Metadata Driven Processing Framework for Azure Data Factory Pipelines – Part 1 of 4"

1. **suhail ali** says:
   February 26, 2020 at 7:18 pm
   Could you use LogicApps to dynamically call the pipeline instead of Azure Functions?

   Reply

   mrpaulandrew says:
   February 27, 2020 at 12:40 pm
   Hi, yes you certainly could. I just prefer the flexibility and ease of creating a Function over the action of a Logic App. Use a Logic App with a Data Factory Web Hook activity as the low level caller if you like.

   Reply

3. Pingback: A Metadata-Driven Framework for ADF Pipelines – Curated SQL
Martin G says:
February 28, 2020 at 2:36 pm
Hey Andrew, thank you very much for these insights. I am be curious to see a poc regarding your answer of question 3, how to load meta data from an AzSQL table within a pipeline.
One way i can think of to load the metadata is
– to define variables in the pipeline for each meta data setting …
– …and then use a lookup activity to select the data as a row from a DB table, ..
– …and then use the output of that activity in set_variable activities, one for each variable

However this feels very clumsy and hard to maintain, resulting in a fan of set_variable activities in the pipeline GUI. Do you know of a more elegant way (other than not using the GUI at all :P)?
Thanks, KR

Reply

mrpaulandrew says:
February 28, 2020 at 4:39 pm
All will be revealed.
Paul Andrew

Reply

4. Pingback: Creating a Simple Staged Metadata Driven Processing Framework for Azure Data
5. Factory Pipelines – Part 2 of 4 – Welcome to the Technical Community Blog of Paul Andrew
Martin says:
March 2, 2020 at 12:29 pm
I don't understand, could you elaborate pls?

Reply

mrpaulandrew says:
March 2, 2020 at 12:30 pm
Hi, which bit?

Reply

Martin says:
March 2, 2020 at 7:26 pm
ah, sry i misunderstood. I will stay tuned to your follow ups.

6. Pingback: Creating a Simple Staged Metadata Driven Processing Framework for Azure Data Factory Pipelines – Part 3 of 4 – Welcome to the Technical Community Blog of Paul Andrew
7. Pingback: Creating a Simple Staged Metadata Driven Processing Framework for Azure Data Factory Pipelines – Part 4 of 4 – Welcome to the Technical Community Blog of Paul Andrew
8. Pingback: ADF.procfwk v1.1 – Service Principal Handling via Metadata – Welcome to the
9. Technical Community Blog of Paul Andrew
Mallik says:
May 1, 2020 at 1:20 pm
Hi Andrew, I liked the idea using Azure functions to invoke Data Factory pipeline to overcome the limitation in Execute Pipeline Activity.

Have you considered using Web Activity to submit REST calls to ADF api to create a pipeline run. The idea is that this way we don't have to switch to azure functions back and forth. All Child pipelines to be invoked by WebActivity.

Do you see any shortfalls in this approach? One problem would be to get the AUTH token to submit the requests. Do you see this approach to be possible at all?

Documentation for API call:
https://docs.microsoft.com/en-us/rest/api/datafactory/pipelines/createrun

Sample post call to invoke a pipeline:
POST
https://management.azure.com/subscriptions/{subscriptionId}/resourceGroups/{resourceGroupName}/providers/Microsoft.DataFactory/factories/{factoryName}/pipelines/{pipelineName}/createRun?api-version=2018-06-01

Reply

mrpaulandrew says:
May 1, 2020 at 2:12 pm
Hi, yes, I did consider it. Authentication was a concern and also capturing when the pipeline finished. As I understand the Web activity would just create an async call (fire and forget).
In v1.4 of my processing framework I handle this with multiple Azure Functions in a more controlled way and am working on capturing fails. v1.4 also handles restarts.
https://github.com/mrpaulandrew/ADF.procfwk
Thanks
Paul, not Andrew

Reply

10. Pingback: Using Logic Apps in a Data Factory Execution Framework – Part 1 – Data Savvy