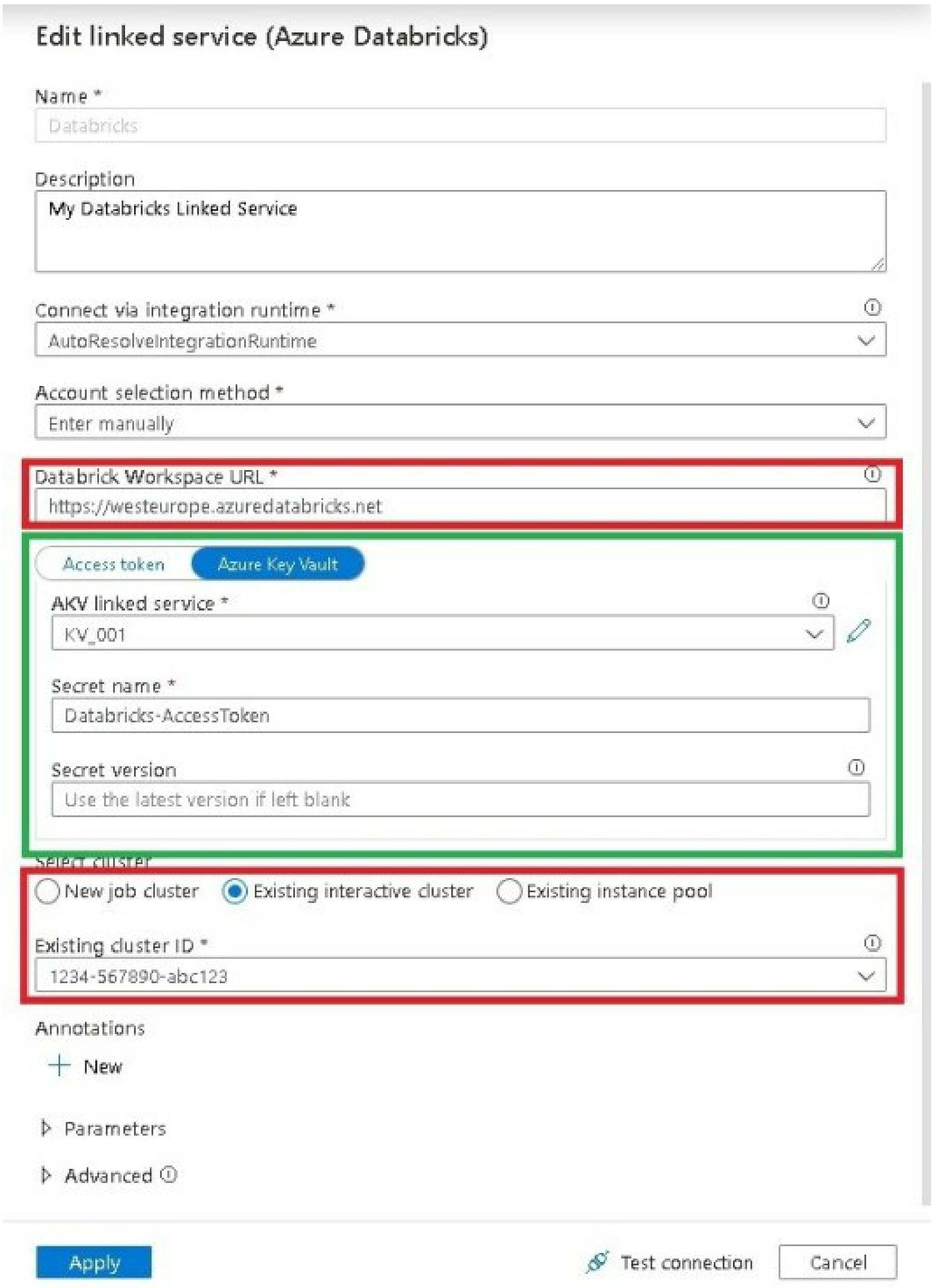
Azure Data Factory, dynamic JSON and Key Vault references

blog.gbrueckl.at/2020/07/azure-data-factory-dynamic-json-and-key-vault-references

By Gerhard Brueckl 2020-07-20

Paul Andrews (b, t) recently blogged about HOW TO USE 'SPECIFY DYNAMIC CONTENTS IN JSON FORMAT' IN AZURE DATA FACTORY LINKED SERVICES. He shows how you can modify the JSON of a given Azure Data Factory linked service and inject parameters into settings which do not support dynamic content in the GUI. What he shows with Linked Services and parameters also applies to Key Vault references – sometimes the GUI allows you to reference a value from the Key Vault instead of hard-coding it but for other settings the GUI only offers a simple text box:


Edit linked service (Azure Databricks)

Name *
Databricks

Description
My Databricks Linked Service

Connect via integration runtime *
AutoResolveIntegrationRuntime

Account selection method *
Enter manually

Databrick Workspace URL *
https://westeurope.azuredatabricks.net

Access token [✓ Azure Key Vault]

AKV linked service *
KV_001

Secret name *
Databricks-AccessToken

Secret version
Use the latest version if left blank

Select cluster
New job cluster [✓ Existing interactive cluster] Existing instance pool

Existing cluster ID *
1234-567890-abc123

As You can see, the setting “AccessToken” can use a Key Vault reference whereas settings like “Databricks Workspace URL” and “Cluster” do not support them. This is usually fine because the guys at Microsoft also thought about this and support Key Vault references for the settings that are actually security relevant or sensitive. Also, providing the option to use Key Vault references everywhere would flood the GUI. So this is just fine.

But there can be good reasons where you want to get values from the Key Vault also for non-sensitive settings, especially when it comes to CI/CD and multiple environments. From my experience, when you implement a bigger ADF project, you will probably have a Key Vault for your sensitive settings and all other values are provided during the deployment via ARM parameters.

So you will end up with a mix of Key Vault references and ARM template parameters which very likely will be derived from the Key Vault at some point anyway. To solve this, you can modify the JSON of an ADF linked service directly and inject KeyVault references into almost every property!
Lets have a look at the JSON of the Databricks linked service from above:

```
1   {
2       "name": "Databricks",
3       "properties": {
4           "annotations": [],
5           "type": "AzureDatabricks",
6           "typeProperties": {
7               "domain": "https://westeurope.azuredatabricks.net",
8               "accessToken": {
9                   "type": "AzureKeyVaultSecret",
10                  "store": {
11                      "referenceName": "KV_001",
12                      "type": "LinkedServiceReference"
13                  },
14                  "secretName": "Databricks-AccessToken"
15              },
16              "existingClusterId": "0717-094253-sir805"
17          },
18          "description": "My Databricks Linked Service"
19      },
20      "type": "Microsoft.DataFactory/factories/linkedservices"
21  }
```

As you can see in lines 8-15, the property “accessToken” references the secret “Databricks-Accesstoken” from the Key Vault linked service “KV_001” and the actual value is populated at runtime.

After reading all this, you can probably guess what we are going to do next –
We also replace the other properties by Key Vault references:

```
{
    "name": "Databricks",
    "properties": {
        "type": "AzureDatabricks",
        "annotations": [],
        "typeProperties": {
            "domain": {
                "type": "AzureKeyVaultSecret",
                "store": {
                    "referenceName": "KV_001",
                    "type": "LinkedServiceReference"
                },
                "secretName": "Databricks-Workspace-URL"
            },
            "accessToken": {
                "type": "AzureKeyVaultSecret",
                "store": {
                    "referenceName": "KV_001",
                    "type": "LinkedServiceReference"
                },
                "secretName": "Databricks-AccessToken"
            },
            "existingClusterId": {
                "type": "AzureKeyVaultSecret",
                "store": {
                    "referenceName": "KV_001",
                    "type": "LinkedServiceReference"
                },
                "secretName": "Databricks-ClusterID"
            }
        }
    }
}
```

You now have a linked service that is configured solely by the Key Vault. If you think one step further, you can replace all values which are usually sourced by ARM parameters with Key Vault references instead and you will end up with an ARM template that only has two parameters – the name of the Data Factory and the URI of the Key Vault linked service! (you may even be able to derive the Key Vaults URI from the Data Factory name if the names are aligned!)

The only drawback I could find so far was that you cannot use the GUI anymore but need to work with the JSON from now on – or at least until you remove the Key Vault references again so that the GUI can display the JSON properly again. But this is just a minor thing as linked services usually do not change very often.

I also tried using the same approach to inject Key Vault references into Pipelines and Dataset but unfortunately this did not work
This is probably because Pipelines and Datasets are evaluated at a different stage and hence cannot dynamically reference the Key Vault.