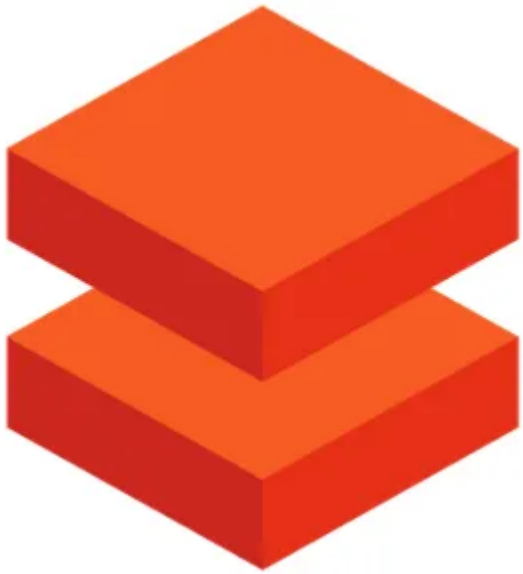
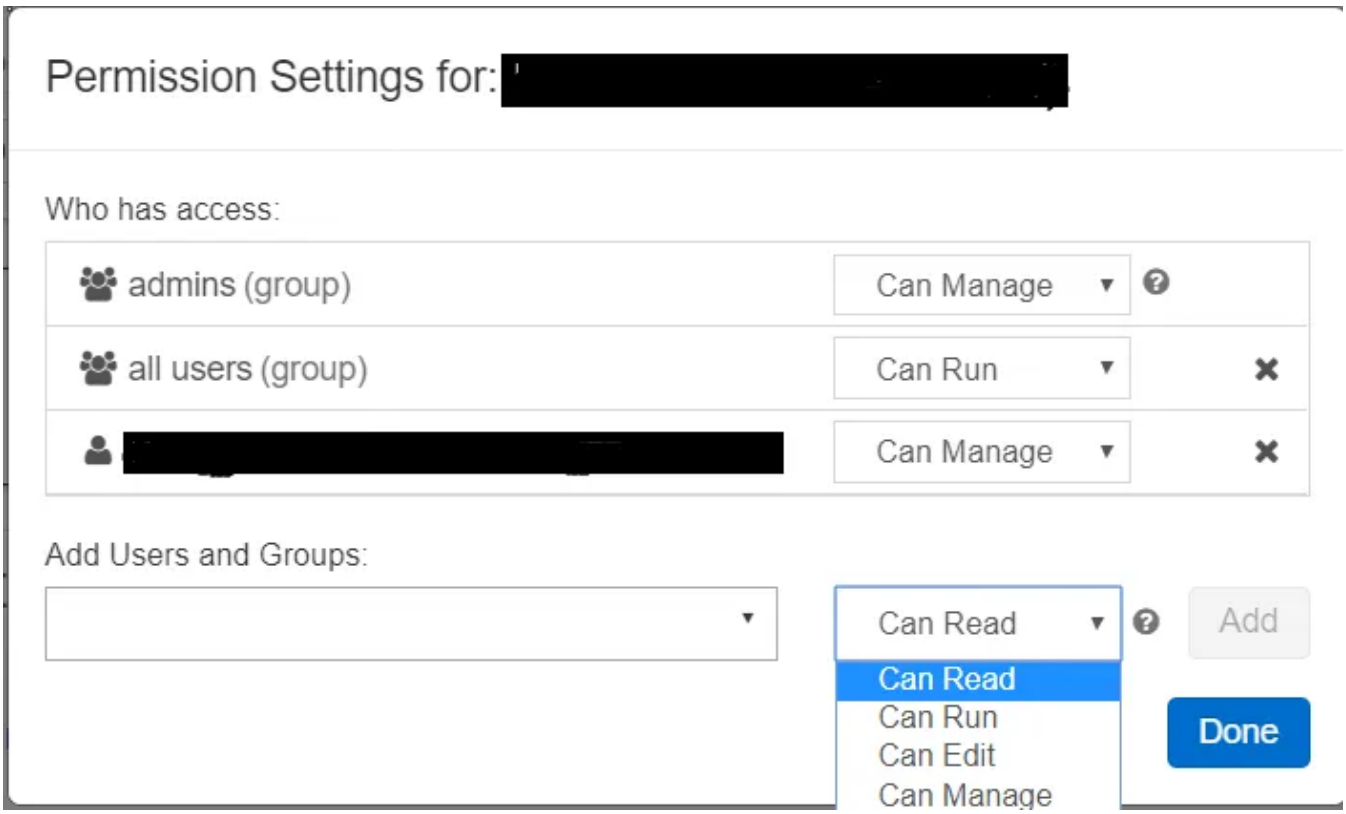
Prosenjit Chakraborty Follow
Dec 5, 2018 · 4 min read · ✦ · Listen

Azure Databricks with Azure Key Vaults



Why?

While connecting to any output storage/systems from Databricks we need to provide user ids/passwords or access keys. These secrets are in clear texts and whoever is having the Databricks workspace access, can see these!



We can't restrict a user to view a particular notebook if she/he has access to the workspace.

Few examples below –

1. Connection setting to Azure Blob Storage

```
%scala
spark.conf.set("fs.azure.account.key.<storage_account>.blob.core.windows.net",
"<storage_account_access_key in clear text>")
```

2. Connection setting to Azure SQL DW

```
%scala
val df = spark.read
.format("com.databricks.spark.sqldw")
.option("url", "jdbc:sqlserver://<server-name>:1433;database=
<database_name>;user=<user>;password=<password in clear
text>;encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.database.
windows.net;loginTimeout=30;")
.option("tempdir",
"wasbs://<container>@<storage_account>.blob.core.windows.net/<container>")
.option("forward_spark_azure_storage_credentials", "true")
.option("query", "SELECT * FROM MyTable WHERE PrimaryKey = 123456")
.load()
```

What is the Solution?

There are couple of solutions:

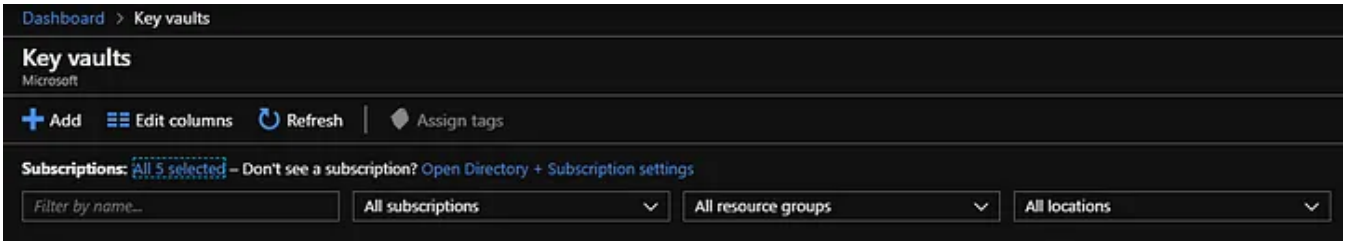
1. **Create a secret in a Databricks-backed scope**

Refer: https://docs.azuredatabricks.net/user-guide/secrets/secrets.html for further details.

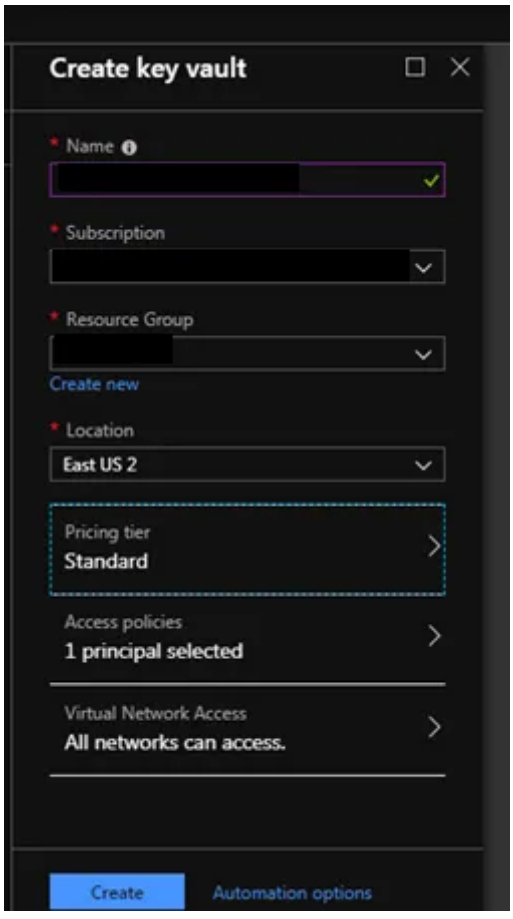
2. Create a secret in an Azure Key Vault-backed scope

Azure Key Vault integration with Azure Databricks is in preview mode at the time of this writing.

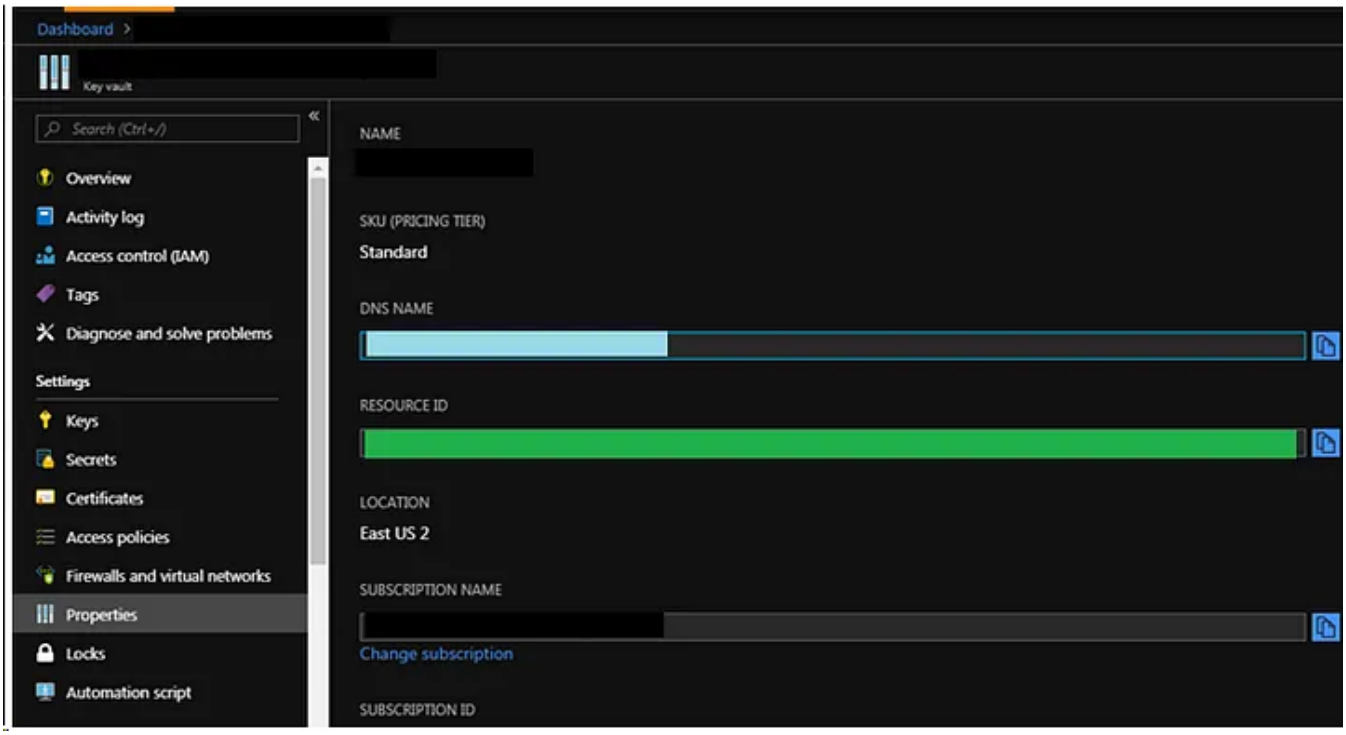
Step 1: Create a new Key Vault



Open Azure Key Vault, click on 'Add'

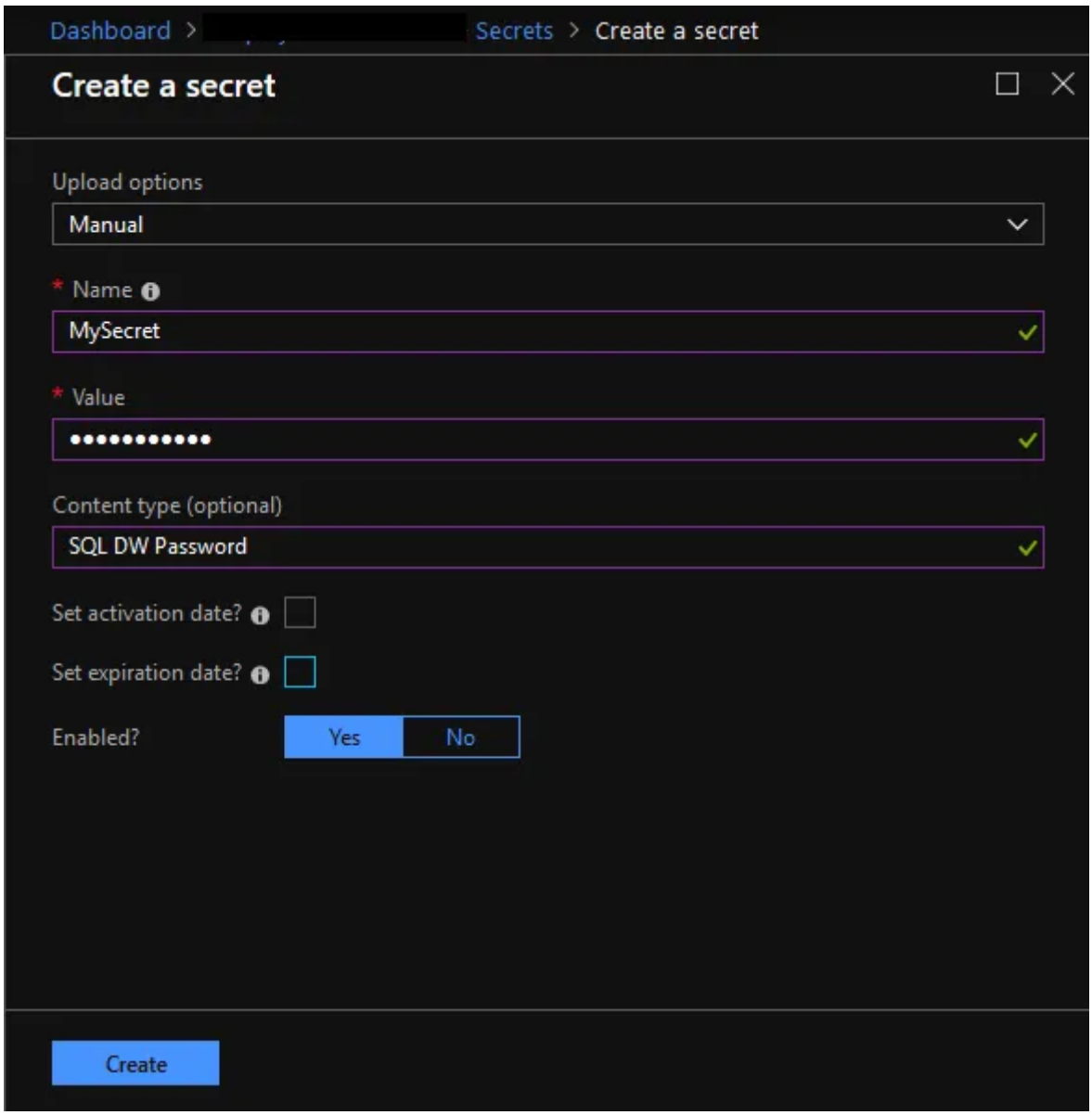


Input Name, select Subscription, Resource Group and other settings

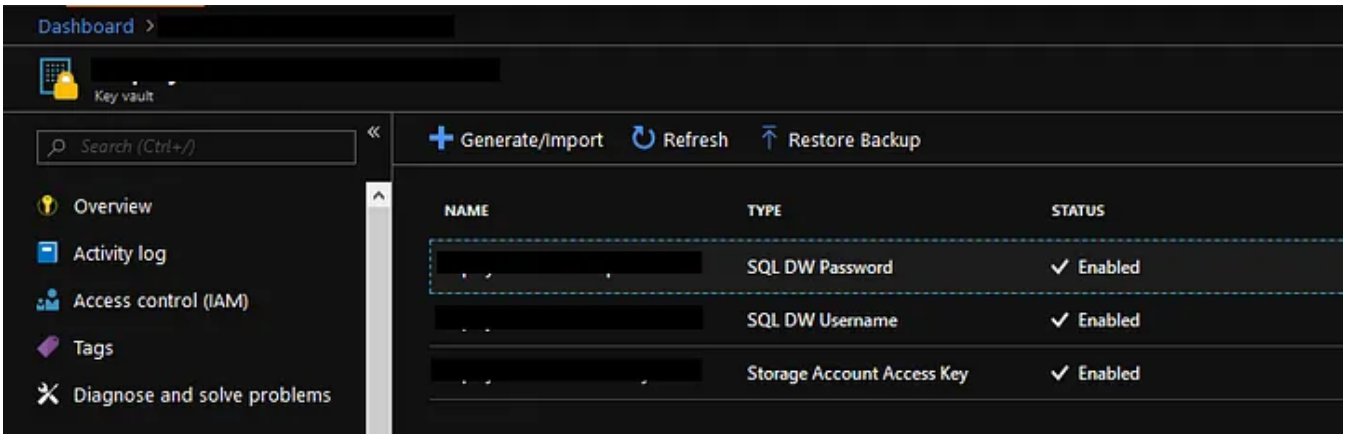


Note the DNS Name & Resource Id of the newly created Key Vault

Step 2: Create a Secret



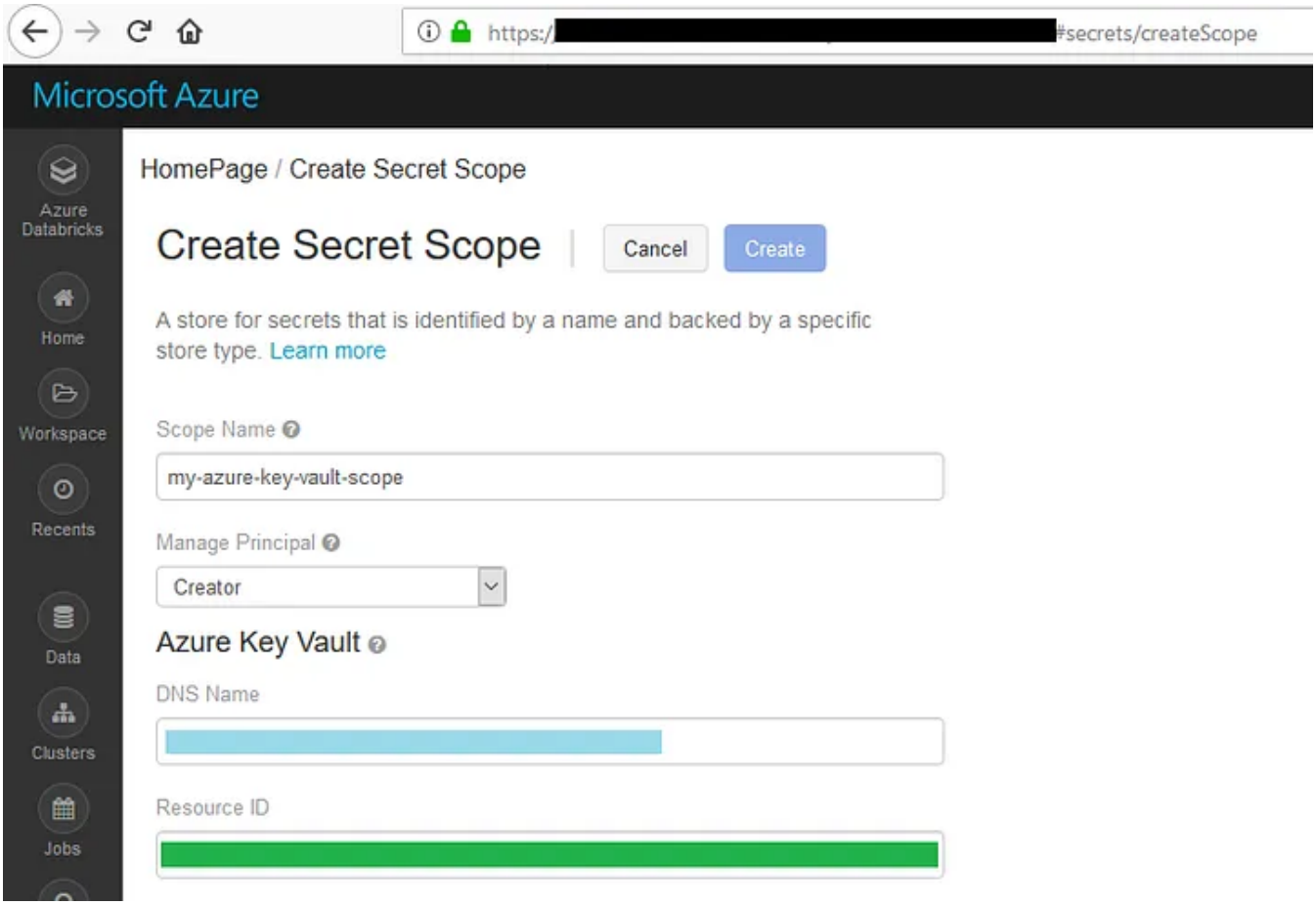
Select Secrets under Settings at the left blade & create a new secret



We can store the userid, password or access keys inside the Azure Key Vault

Step 3: Create a Secret Scope at Databricks

Open the Databricks Create Secret Scope link: https://<Databricks link>#secrets/createScope



Input a Scope Name, add the DNS Name & Resource ID taken from Step 1

Azure Databricks is now linked with the Azure Key Vault!

Step 4: Use the Secrets from Azure Databricks

We can access the secret value from Azure Key Vault by using the following:

dbutils.secrets.get(scope = “my-azure-key-vault-scope”, key = “MySecret”)

So, the connections to the external systems can be changed as follows:

1. **Azure Blob Storage:**

%scala
spark.conf.set(“fs.azure.account.key.<storage_account>.blob.core.windows.net”, **dbutils.secrets.get(scope = “my-azure-key-vault-scope”, key = “secret-access-key”)**)

2. **Azure SQL Data Warehouse:**

%scala
val df = spark.read
.format(“com.databricks.spark.sqldw”)

```
.option("url", s"""jdbc:sqlserver://<server-name>:1433;database=
<database_name>;user=${dbutils.secrets.get(scope = "my-azure-key-vault-scope",
key = "username")};password=${dbutils.secrets.get(scope = "my-azure-key-vault-
scope", key =
"MySecret")};encrypt=true;trustServerCertificate=false;hostNameInCertificate=*.da
tabase.windows.net;loginTimeout=30;""")
.option("tempdir",
"wasbs://<container>@<storage_account>.blob.core.windows.net/<container>")
.option("forward_spark_azure_storage_credentials", "true")
.option("query", "SELECT * FROM MyTable WHERE PrimaryKey = 123456")
.load()
```

Now, users having access to Databricks notebooks can only see the Azure Key Vault secret names but not the actual secrets! They can only use it to access the external system from other notebooks. Databricks blocks printing the actual value in notebook execution output.

```
1 dbutils.secrets.get(scope = "my-azure-key-vault-scope", key = "MySecret")
```

```
res5: String = [REDACTED]
```

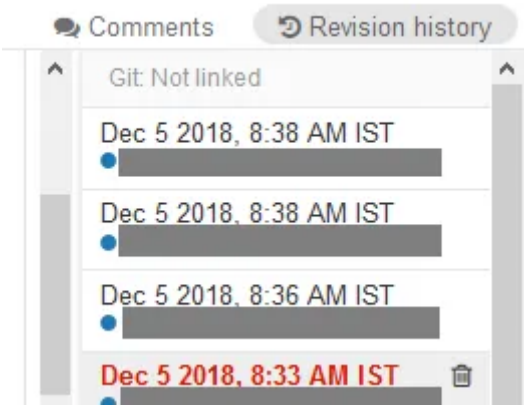
The actual output is obscured

If we want to apply role based access control to restrict read/access a notebook at all, we should follow: https://docs.microsoft.com/en-us/azure/role-based-access-control/role-assignments-portal

That's all?

Wait! Databricks maintains revision history. So, we need to clean this up to remove any trace of passwords. Take some care, as this could remove any important version changes.

If we have already integrated Git with Databricks, we need to remove passwords/access keys from our the version management system as well.



Databricks notebook — Revision history

Cautions

Azure Key Vault comes with some extra cost, for details follow:
https://azure.microsoft.com/en-us/pricing/details/key-vault/

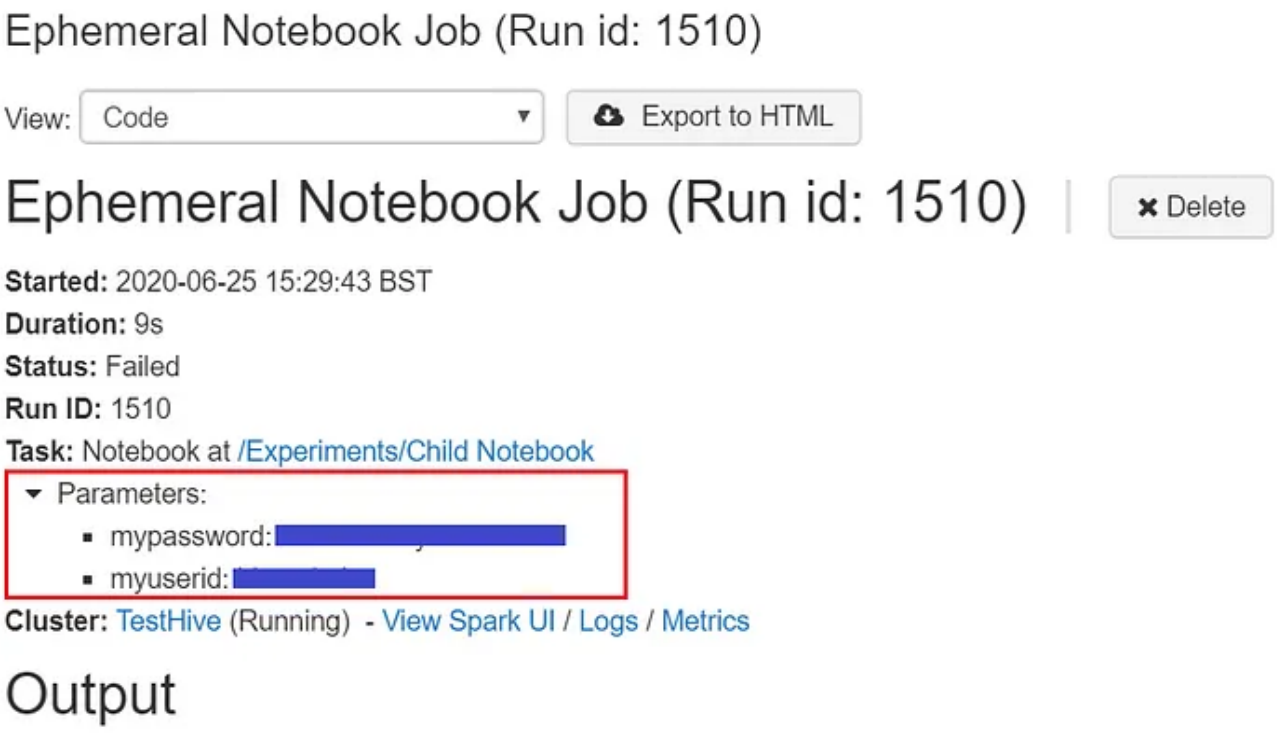
Update — 25th June 2020

Never pass the secrets to another notebook; fetch & use them locally!

```
dbutils.notebook.run("Child Notebook", 0,
   Map(
    "myuserid" -> dbutils.secrets.get(scope = "mysecretscope", key = "myuserid"),
    "mypassword" -> dbutils.secrets.get(scope = "mysecretscope", key = "mypassword")
   )
  )
```

Notebook job #1513

Pass the secrets from parent notebook to a called child notebook.



Once run, the secrets will be revealed in the Parameters section!

The clear text parameters can be retrieved from the job execution history.

Thanks for reading!! If you have enjoyed, Clap & Share it!! To see similar posts, follow me on Medium & LinkedIn.

Azure Databricks Big Data Apache Spark