Lab 13: Creating Secret Scopes in Databricks (All Methods: A, B, and C)

Author: Dr. Sandeep Kumar Sharma

Introduction

In this lab, we will learn **ALL** three methods to create secret scopes in Databricks. This is one of the most confusing topics for beginners, so we will break it down into very simple, step-by-step instructions.

Most participants struggle because: - They do not know where the Databricks CLI runs - They are not sure which method is required in production - They don't understand the difference between *internal secret scopes* and *Key Vault–backed secret scopes*

After this lab, your confidence on secrets + authentication will become very strong.

Learning Objective

By the end of this lab, you will understand: - Method A: How to create a **Key Vault–backed secret scope** using the Databricks UI (NO CLI required) - Method B: How to install and use the **Databricks CLI** to create an **internal secret scope** - Method C: How to choose the right method using comparisons, and how to read these secrets inside a Databricks notebook

---

Learning Outcome

You will be able to: - Create secure secret scopes in real production environments - Store sensitive credentials (SP secrets, SAS tokens, access keys) - Access secrets from a Databricks notebook using `dbutils.secrets.get()` - Decide which method is best for your architecture

---

METHOD A – Azure Key Vault–Backed Secret Scope (Recommended, No CLI Required)

Why use this?

- Most secure for production
- No CLI needed
- Secrets remain in Azure Key Vault
- Rotation and auditing are automatic

Step A1 — Create a Key Vault (If not already created)

1. Go to **Azure Portal → Create Resource → Key Vault**
2. Name your Key Vault: `mykeyvault-demo`
3. Choose region, resource group
4. Click **Review + Create**

Step A2 — Add the secret inside Key Vault

1. Open your Key Vault
2. Go to **Secrets → Generate/Import**
3. Name: `sp-client-secret`
4. Value: *your real service principal secret*
5. Save

Your secret now lives in Azure Key Vault.

Step A3 — Open Databricks Workspace and Create a Scope

1. Go to your Databricks Workspace
2. On the left-panel click **Settings → User Settings**
3. Go to the tab **Access Tokens / Advanced → Secret Scopes** (depending on workspace version)
4. Click **Create Secret Scope**
5. Provide a name: `my-adls-scope`
6. Select **Azure Key Vault**

7. Choose the Key Vault created earlier
8. Click **Create**

Your Key Vault is now linked to Databricks!

Step A4 — Access the secret inside a Databricks notebook

```
client_secret = dbutils.secrets.get("my-adls-scope", "sp-client-secret")
print("Secret retrieved successfully")
```

This confirms your configuration is correct.

METHOD B – Create Secret Scope Using Databricks CLI (Internal Secrets)

Why use this?

- Good for development or automation pipelines
- Secrets are stored **inside Databricks itself**, not Azure Key Vault
- Not recommended for highly secure production environments

Step B1 — Install Databricks CLI (on your laptop or server)

You **cannot** run these commands inside Databricks notebooks.

On Windows (PowerShell):

```
pip install databricks-cli
```

On Mac/Linux:

```
pip install databricks-cli
```

Step B2 — Generate a Databricks Access Token

1. Open Databricks workspace
2. Click **Settings → User Settings → Access Tokens**
3. Generate New Token
4. Copy the token (save it safely)

Step B3 — Configure CLI

Run on your local PC:

```
databricks configure --token
```

It will ask: - Databricks Host (your workspace URL) - Token

Enter values.

Step B4 — Create the secret scope

```
databricks secrets create-scope --scope my-internal-scope
```

Step B5 — Add secrets to the scope

```
databricks secrets put --scope my-internal-scope --key sp-client-secret
# It will open a prompt. Paste your secret.
```

Now your secret is stored in Databricks internal vault.

Step B6 — Access the secret in a notebook

```
client_secret = dbutils.secrets.get("my-internal-scope", "sp-client-secret")
display(client_secret)
```

METHOD C – Understanding Differences + Which One Should You Use?

Key Vault–Backed Scope (Method A)

- Secrets live in **Azure Key Vault**
- Best for production
- Automatic rotation
- Role-based access control
- NO CLI needed

Internal Secret Scope (Method B)

- Secrets live **inside Databricks**
- Good for test environments
- Requires CLI setup
- Not ideal for enterprise security

Notebook Retrieval (Common for Both)

Use the same command for both:

```
client_secret = dbutils.secrets.get("scope-name", "key-name")
```

Verification Test

Run this test in a notebook:

```
try:
    value = dbutils.secrets.get("my-adls-scope", "sp-client-secret")
    print("Secret access successful: OK ✔️")
except Exception as e:
    print("Error:", e)
```

End of Lab 13

You now understand: - How to create a Secret Scope using **Azure Key Vault** (No CLI) - How to create a Secret Scope using **Databricks CLI** (Internal method) - How to compare production vs non-production usage - How to retrieve secrets securely inside notebooks

If you want, I can now create: **Lab 14 — Using Managed Identity with Databricks (Most Secure, No Secrets Needed)**.