Secret scopes

learn.microsoft.com/en-us/azure/databricks/security/secrets/secret-scopes

- Article
- 01/21/2023
- 7 minutes to read

Managing secrets begins with creating a secret scope. A secret scope is collection of secrets identified by a name.

A workspace is limited to a maximum of 100 secret scopes. Contact your Azure Databricks representative if you need more.

Note

Databricks recommends aligning secret scopes to roles or applications rather than individuals.

Overview

There are two types of secret scope: Azure Key Vault-backed and Databricks-backed.

Azure Key Vault-backed scopes

To reference secrets stored in an Azure Key Vault, you can create a secret scope backed by Azure Key Vault. You can then leverage all of the secrets in the corresponding Key Vault instance from that secret scope. Because the Azure Key Vault-backed secret scope is a read-only interface to the Key Vault, the `PutSecret` and `DeleteSecret` Secrets API 2.0 operations are not allowed. To manage secrets in Azure Key Vault, you must use the Azure Set Secret REST API or Azure portal UI.

Note

Creating an Azure Key Vault-backed secret scope role grants the **Get** and **List** permissions to the resource ID for the Azure Databricks service using key vault access policies, even if the key vault is using the Azure RBAC permissions model.

Databricks-backed scopes

A Databricks-backed secret scope is stored in (backed by) an encrypted database owned and managed by Azure Databricks. The secret scope name:

- Must be unique within a workspace.
- Must consist of alphanumeric characters, dashes, underscores, `@`, and periods, and may not exceed 128 characters.

The names are considered non-sensitive and are readable by all users in the workspace.

You create a Databricks-backed secret scope using the Databricks CLI setup & documentation (version 0.7.1 and above). Alternatively, you can use the Secrets API 2.0.

Scope permissions

Scopes are created with permissions controlled by ACLs. By default, scopes are created with `MANAGE` permission for the user who created the scope (the "creator"), which lets the creator read secrets in the scope, write secrets to the scope, and change ACLs for the scope. If your account has the Premium Plan, you can assign granular permissions at any time after you create the scope. For details, see Secret access control.

You can also override the default and explicitly grant `MANAGE` permission to all users when you create the scope. In fact, you *must* do this if your account does not have the Premium Plan.

Best practices

As a team lead, you might want to create different scopes for Azure Synapse Analytics and Azure Blob storage credentials and then provide different subgroups in your team access to those scopes. You should consider how to achieve this using the different scope types:

- If you use a Databricks-backed scope and add the secrets in those two scopes, they will be different secrets (Azure Synapse Analytics in scope 1, and Azure Blob storage in scope 2).
- If you use an Azure Key Vault-backed scope with each scope referencing a **different Azure Key Vault** and add your secrets to those two Azure Key Vaults, they will be different sets of secrets (Azure Synapse Analytics ones in scope 1, and Azure Blob storage in scope 2). These will work like Databricks-backed scopes.
- If you use two Azure Key Vault-backed scopes with both scopes referencing the **same Azure Key Vault** and add your secrets to that Azure Key Vault, all Azure Synapse Analytics and Azure Blob storage secrets will be available. Since ACLs are at the scope level, all members across the two subgroups will see all secrets. This arrangement does not satisfy your use case of restricting access to a set of secrets to each group.

Create an Azure Key Vault-backed secret scope

You can create an Azure Key Vault-backed secret scope using the UI or using the Databricks CLI.

Create an Azure Key Vault-backed secret scope using the UI

1. Verify that you have the Directory Readers role in your Azure Active Directory tenant.

2. Verify that you have Contributor or Owner role on the Azure key vault instance that you want to use to back the secret scope.

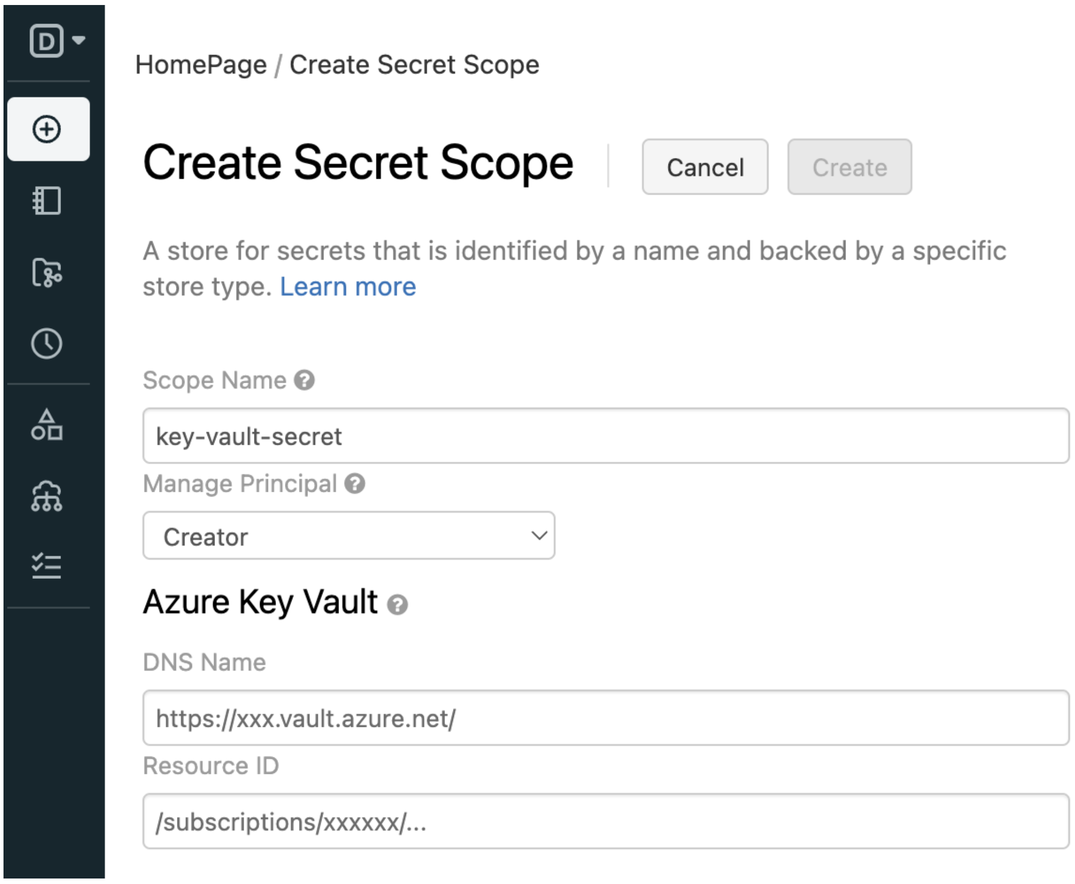
   If you do not have a key vault instance, follow the instructions in Quickstart: Create a Key Vault using the Azure portal.

   Note

   Creating an Azure Key Vault-backed secret scope requires the Contributor or Owner role on the Azure key vault instance even if the Azure Databricks service has previously been granted access to the key vault.

3. If your Azure key vault instance uses the Azure RBAC permissions model, grant the **Key Vault Secrets User** role to the unique resource ID for the Azure Databricks service, which is `2ff814a6-3304-4ab8-85cb-cd0e6f879c1d`.

4. Go to `https://<databricks-instance>#secrets/createScope`. This URL is case sensitive; scope in `createScope` must be uppercase.



Create Secret Scope

A store for secrets that is identified by a name and backed by a specific store type. Learn more

Scope Name: key-vault-secret

Manage Principal: Creator

Azure Key Vault

DNS Name: https://xxx.vault.azure.net/

Resource ID: /subscriptions/xxxxxx/...

5. Enter the name of the secret scope. Secret scope names are case insensitive.

6. Use the **Manage Principal** drop-down to specify whether *All Users* have `MANAGE` permission for this secret scope or only the *Creator* of the secret scope (that is to say, you).

   `MANAGE` permission allows users to read and write to this secret scope, and, in the case of accounts on the Premium Plan, to change permissions for the scope.

   Your account must have the Premium Plan for you to be able to select *Creator*. This is the recommended approach: grant `MANAGE` permission to the *Creator* when you create the secret scope, and then assign more granular access permissions after you have tested the scope. For an example workflow, see Secret workflow example.

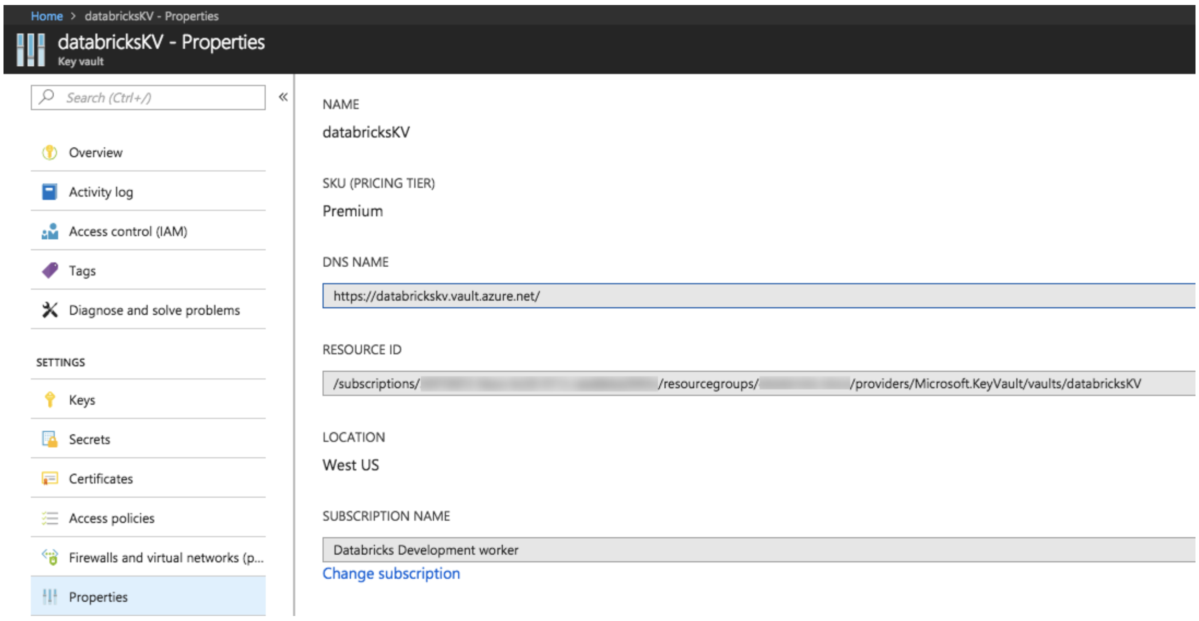
   If your account has the Standard Plan, you must set the `MANAGE` permission to the "All Users" group. If you select *Creator* here, you will see an error message when you try to save the scope.

   For more information about the `MANAGE` permission, see Secret access control.

7. Enter the **DNS Name** (for example, `https://databrickskv.vault.azure.net/`) and **Resource ID**, for example:

   ```
   /subscriptions/xxxxxxxx-xxxx-xxxx-xxxx-
   xxxxxxxxxxxx/resourcegroups/databricks-
   rg/providers/Microsoft.KeyVault/vaults/databricksKV
   ```

   These properties are available from the **Properties** tab of an Azure Key Vault in your Azure portal.



NAME
databricksKV

SKU (PRICING TIER)
Premium

DNS NAME
https://databrickskv.vault.azure.net/

RESOURCE ID
/subscriptions/[redacted]/resourcegroups/[redacted]/providers/Microsoft.KeyVault/vaults/databricksKV

LOCATION
West US

SUBSCRIPTION NAME
Databricks Development worker

8. Click the **Create** button.

9. Use the Databricks CLI setup & documentation `databricks secrets list-scopes` command to verify that the scope was created successfully.

For an example of using secrets when accessing Azure Blob storage, see Mounting cloud object storage on Azure Databricks.

Create an Azure Key Vault-backed secret scope using the Databricks CLI

1. Install the CLI and configure it to use an Azure Active Directory (Azure AD) token for authentication.

   Important

   You need an Azure AD user token to create an Azure Key Vault-backed secret scope with the Databricks CLI. You cannot use an Azure Databricks personal access token or an Azure AD application token that belongs to a service principal.

   If the key vault exists in a different tenant than the Azure Databricks workspace, the Azure AD user who creates the secret scope must have permission to create service principals in the key vault's tenant. Otherwise, the following error occurs:

   ```
   Unable to grant read/list permission to Databricks service principal to
   KeyVault 'https://xxxxx.vault.azure.net/': Status code 403, {"odata.error":
   {"code":"Authorization_RequestDenied","message":
   {"lang":"en","value":"Insufficient privileges to complete the
   operation."},"requestId":"XXXXX","date":"YYYY-MM-DDTHH:MM:SS"}}
   ```

2. Create the Azure Key Vault scope:

   Bash

   ```
   databricks secrets create-scope --scope <scope-name> --scope-backend-type
   AZURE_KEYVAULT --resource-id <azure-keyvault-resource-id> --dns-name <azure-
   keyvault-dns-name>
   ```

   By default, scopes are created with `MANAGE` permission for the user who created the scope. If your account does not have the Premium Plan, you must override that default and explicitly grant the `MANAGE` permission to the `users` (all users) group when you create the scope:

   Bash

   ```
    databricks secrets create-scope --scope <scope-name> --scope-backend-type
   AZURE_KEYVAULT --resource-id <azure-keyvault-resource-id> --dns-name <azure-
   keyvault-dns-name> --initial-manage-principal users
   ```

   If your account in on the Premium Plan, you can change permissions at any time after you create the scope. For details, see Secret access control.

   Once you have created a Databricks-backed secret scope, you can add secrets.

For an example of using secrets when accessing Azure Blob storage, see Mounting cloud object storage on Azure Databricks.

Create a Databricks-backed secret scope

Secret scope names are case insensitive.

To create a scope using the Databricks CLI:

Bash

```bash
databricks secrets create-scope --scope <scope-name>
```

By default, scopes are created with `MANAGE` permission for the user who created the scope. If your account does not have the Premium Plan, you *must* override that default and explicitly grant the `MANAGE` permission to "users" (all users) when you create the scope:

Bash

```bash
databricks secrets create-scope --scope <scope-name> --initial-manage-principal
users
```

You can also create a Databricks-backed secret scope using the Secrets API Put secret operation.

If your account has the Premium Plan, you can change permissions at any time after you create the scope. For details, see Secret access control.

Once you have created a Databricks-backed secret scope, you can add secrets.

List secret scopes

To list the existing scopes in a workspace using the CLI:

Bash

```bash
databricks secrets list-scopes
```

You can also list existing scopes using the Secrets API List secrets operation.

Delete a secret scope

Deleting a secret scope deletes all secrets and ACLs applied to the scope. To delete a scope using the CLI:

Bash

```bash
databricks secrets delete-scope --scope <scope-name>
```

You can also delete a secret scope using the Secrets API Delete secret scope operation.