Azure Key Vault is a cloud service that provides secure storage and access to secrets, keys, and certificates for your Azure applications and services. It helps protect sensitive information like passwords, connection strings, and encryption keys by encrypting them and storing them securely in a dedicated vault.

Here's a more detailed explanation:

* **Secure Storage:**

Key Vault stores secrets, keys, and certificates in a highly secure environment, protecting them from unauthorized access.

* **Centralized Management:**

It provides a centralized location for managing these secrets, simplifying access and versioning.

* **Access Control:**

You can grant specific users or applications access to the secrets they need, ensuring that only authorized entities can retrieve them.

* **Key Vault Vaults:**

Vaults store and manage cryptographic keys, secrets, certificates, and storage account keys.

* **Managed HSM Pools:**

These pools store and manage HSM-backed cryptographic keys, offering enhanced security.

* **Integration:**

Azure Key Vault integrates with other Azure services, making it easy to use with your applications and infrastructure.

* **Monitoring:**

You can use Azure Monitor to track the utilization and operations of your key vaults and identify potential issues.

* **Auditing:**

Key Vault logs activities, allowing you to audit access and identify any unauthorized access attempts.

* **FIPS Validation:**

Keys are stored in FIPS-validated HSMs (Hardware Security Modules) for added assurance.

* **Cost:**

Key Vault offers different pricing tiers and subscription options.

Azure Key Vault is a cloud service for securely storing and accessing secrets. A secret is anything that you want to tightly control access to, such as API keys, passwords, certificates, or cryptographic keys. Key Vault service supports two types of containers: vaults and managed hardware security module(HSM) pools. Vaults support storing software and HSM-backed keys, secrets, and certificates. Managed HSM pools only support HSM-backed keys.

**Managed identities**: Azure Key Vault provides a way to securely store credentials and other keys and secrets, but your code needs to authenticate to Key Vault to retrieve them. Using a managed identity makes solving this problem simpler by giving Azure services an automatically managed identity in Microsoft Entra ID. You can use this identity to authenticate to Key Vault or any service that supports Microsoft Entra authentication, without having any credentials in your code.

Azure Key Vault is one of several [key management solutions in Azure](https://learn.microsoft.com/en-us/azure/security/fundamentals/key-management), and helps solve the following problems:

* **Secrets Management** - Azure Key Vault can be used to Securely store and tightly control access to tokens, passwords, certificates, API keys, and other secrets
* **Key Management** - Azure Key Vault can be used as a Key Management solution. Azure Key Vault makes it easy to create and control the encryption keys used to encrypt your data.
* **Certificate Management** - Azure Key Vault lets you easily provision, manage, and deploy public and private Transport Layer Security/Secure Sockets Layer (TLS/SSL) certificates for use with Azure and your internal connected resources.

Azure Key Vault has two service tiers: Standard, which encrypts with a software key, and a Premium tier, which includes hardware security module(HSM)-protected keys.

**Why use Azure Key Vault?**

1. **Centralize application secrets**

Centralizing storage of application secrets in Azure Key Vault allows you to control their distribution. Key Vault greatly reduces the chances that secrets may be accidentally leaked. When application developers use Key Vault, they no longer need to store security information in their application. Not having to store security information in applications eliminates the need to make this information part of the code. For example, an application may need to connect to a database. Instead of storing the connection string in the app's code, you can store it securely in Key Vault.

Your applications can securely access the information they need by using URIs. These URIs allow the applications to retrieve specific versions of a secret. There's no need to write custom code to protect any of the secret information stored in Key Vault.

1. **Securely store secrets and keys**

Access to a key vault requires proper authentication and authorization before a caller (user or application) can get access. Authentication establishes the identity of the caller, while authorization determines the operations that they're allowed to perform.

Authentication is done via Microsoft Entra ID. Authorization may be done via Azure role-based access control (Azure RBAC) or Key Vault access policy. Azure RBAC can be used for both management of the vaults and to access data stored in a vault, while key vault access policy can only be used when attempting to access data stored in a vault.

Azure key vaults are encrypted at rest with a key stored in hardware security modules (HSMs). Azure safeguards keys, secrets, and certificates using industry-standard algorithms, key lengths, and software cryptographic modules. For added assurance, you can generate or import keys in HSMs (type RSA-HSM, EC-HSM, or OCT-HSM) that never leave the HSM boundary.

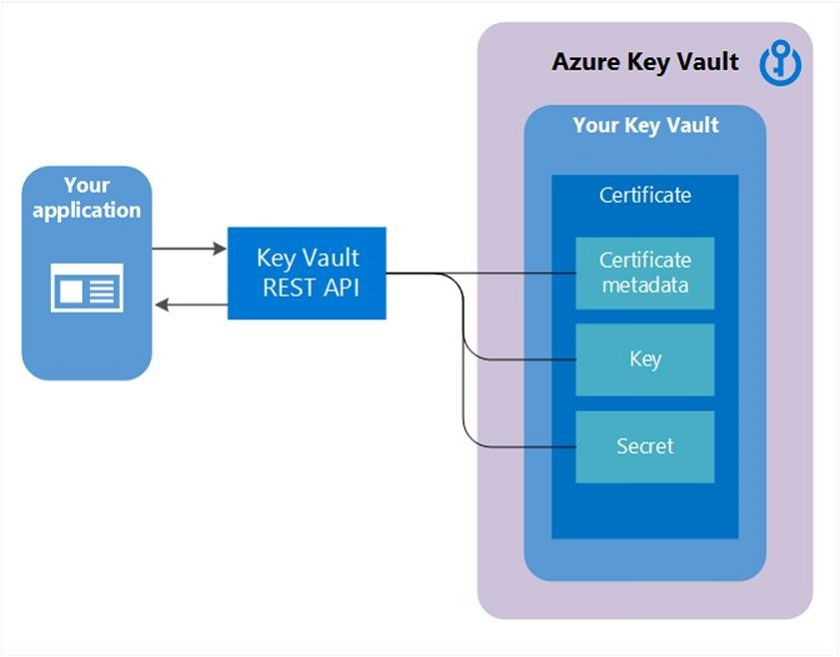
1. **Monitor access and use**

Once you've created a couple of Key Vaults, you'll want to monitor how and when your keys and secrets are being accessed. You can monitor activity by enabling logging for your vaults. You can configure Azure Key Vault to:

* Archive to a storage account.
* Stream to an event hub.
* Send the logs to Azure Monitor logs.

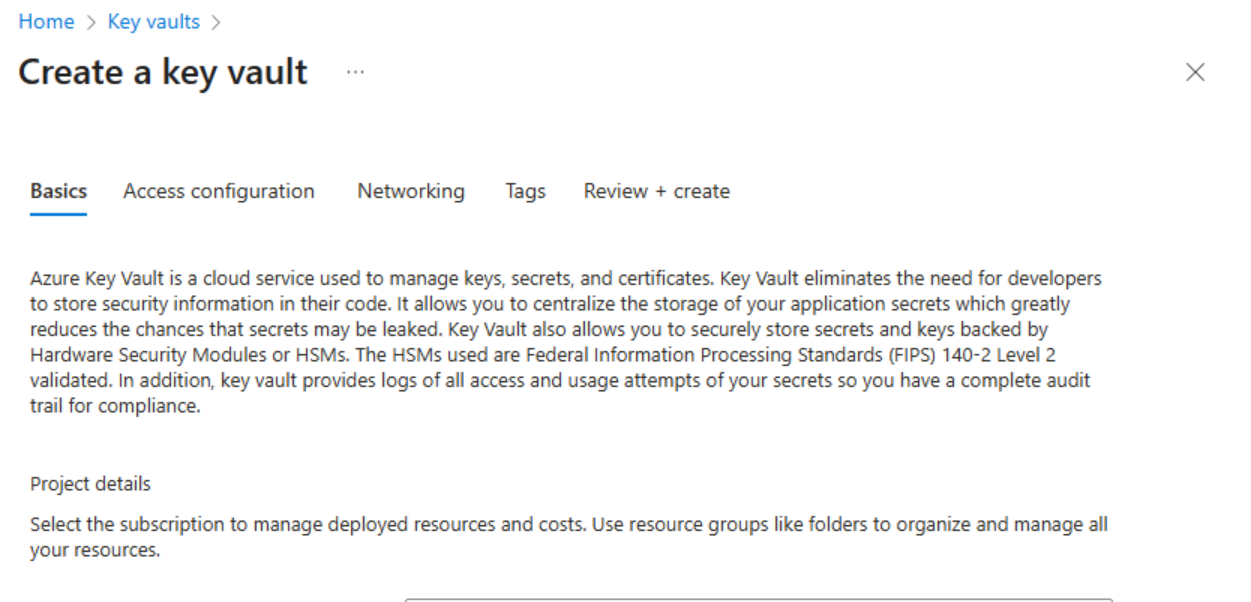
Azure Key Vault enables Azure subscribers to safeguard and control cryptographic keys and other secrets used by cloud apps and services. Azure Key Vault provides two types of containers:

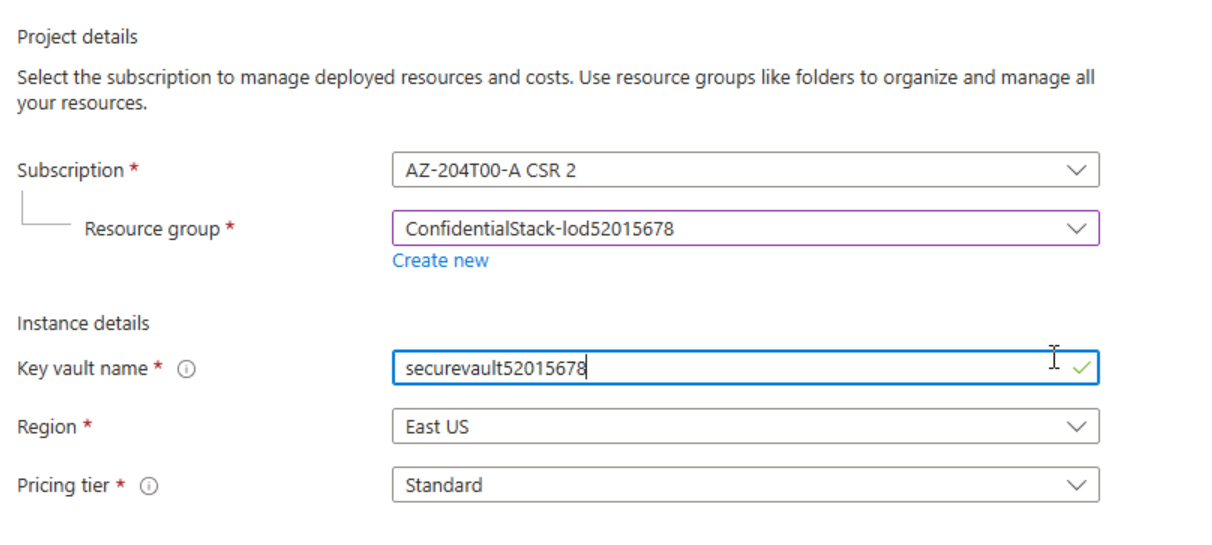
* Vaults for storing and managing cryptographic keys, secrets, certificates, and storage account keys.
* Managed HSM pool for storing and managing HSM-backed cryptographic keys.

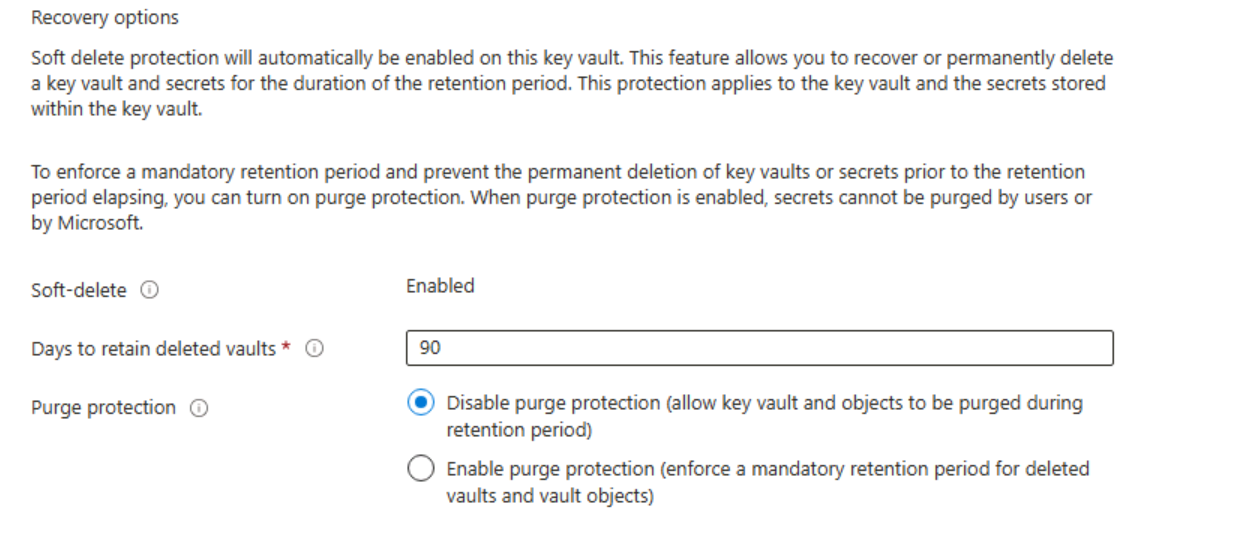


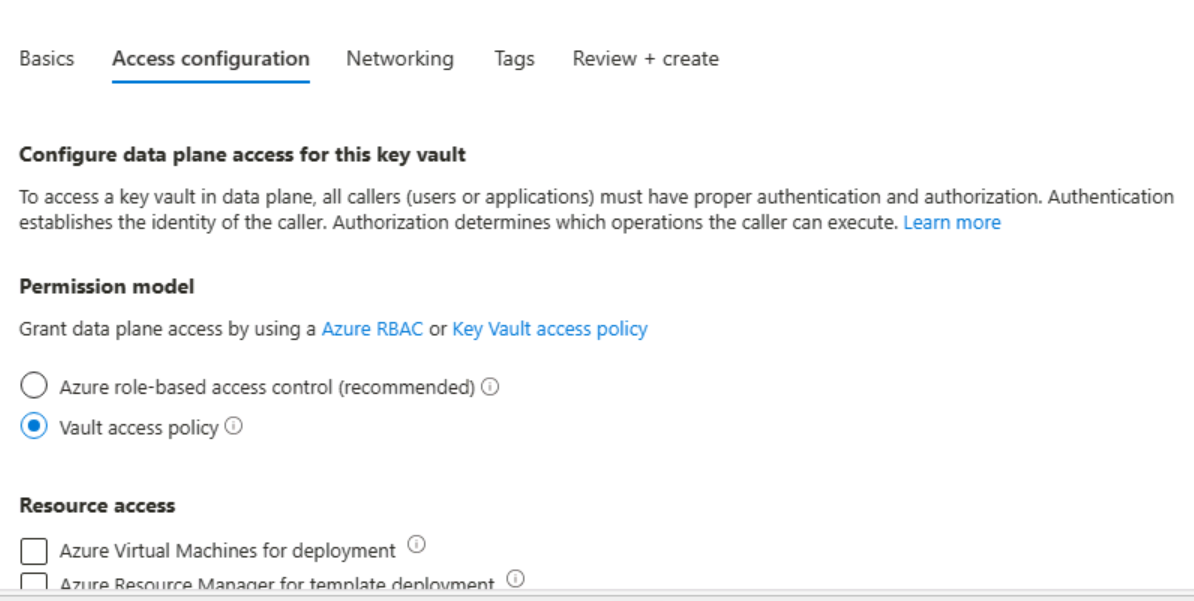
DefaultEndpointsProtocol=https;AccountName=securestor52015678;AccountKey=dAHk14/RmhMiemVqOHJte36Yk6d/Ra7G7HYCYVbwEK0dmZQc6+BHckHpfltK6G4QsaBIeZ6hikQa+ASt2bbxmA==;EndpointSuffix=core.windows.net

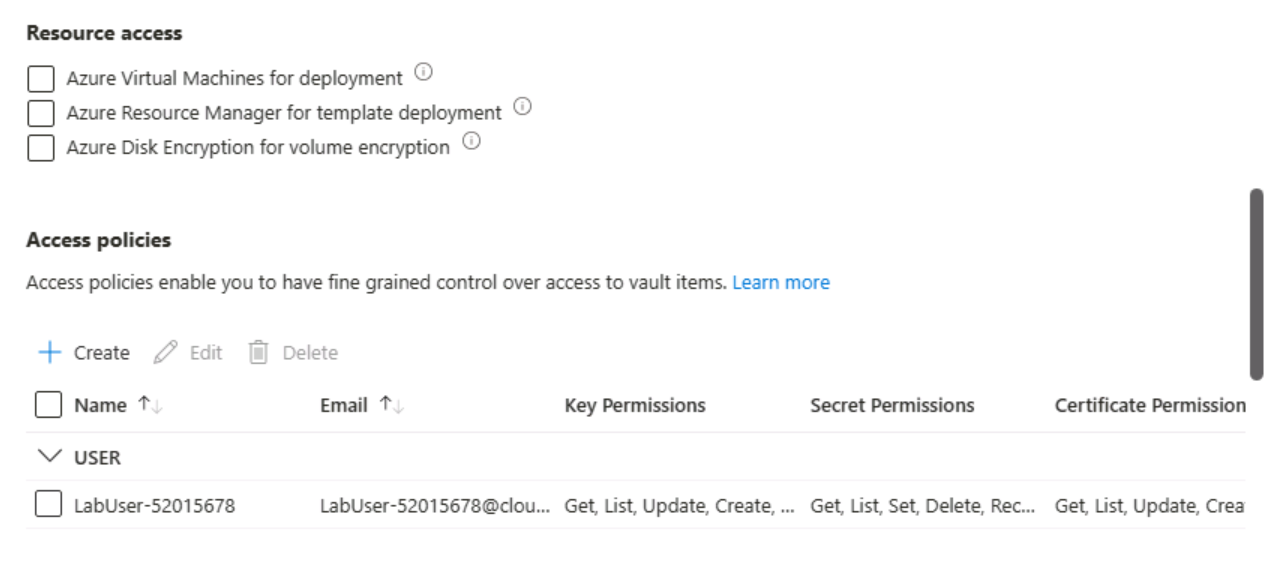
dAHk14/RmhMiemVqOHJte36Yk6d/Ra7G7HYCYVbwEK0dmZQc6+BHckHpfltK6G4QsaBIeZ6hikQa+ASt2bbxmA==

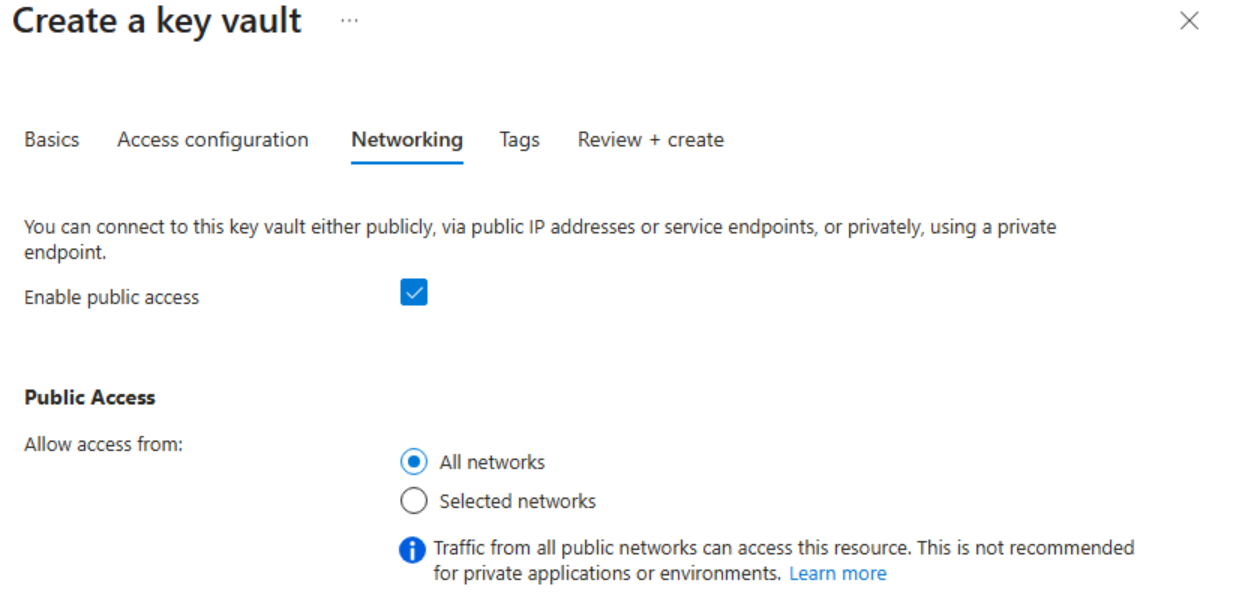


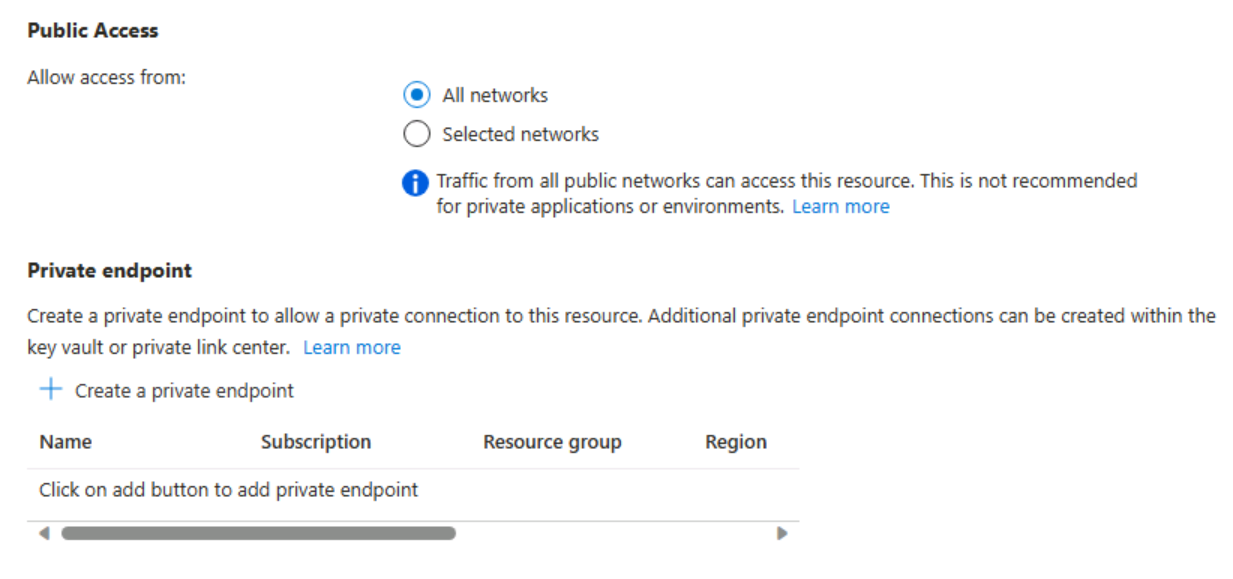


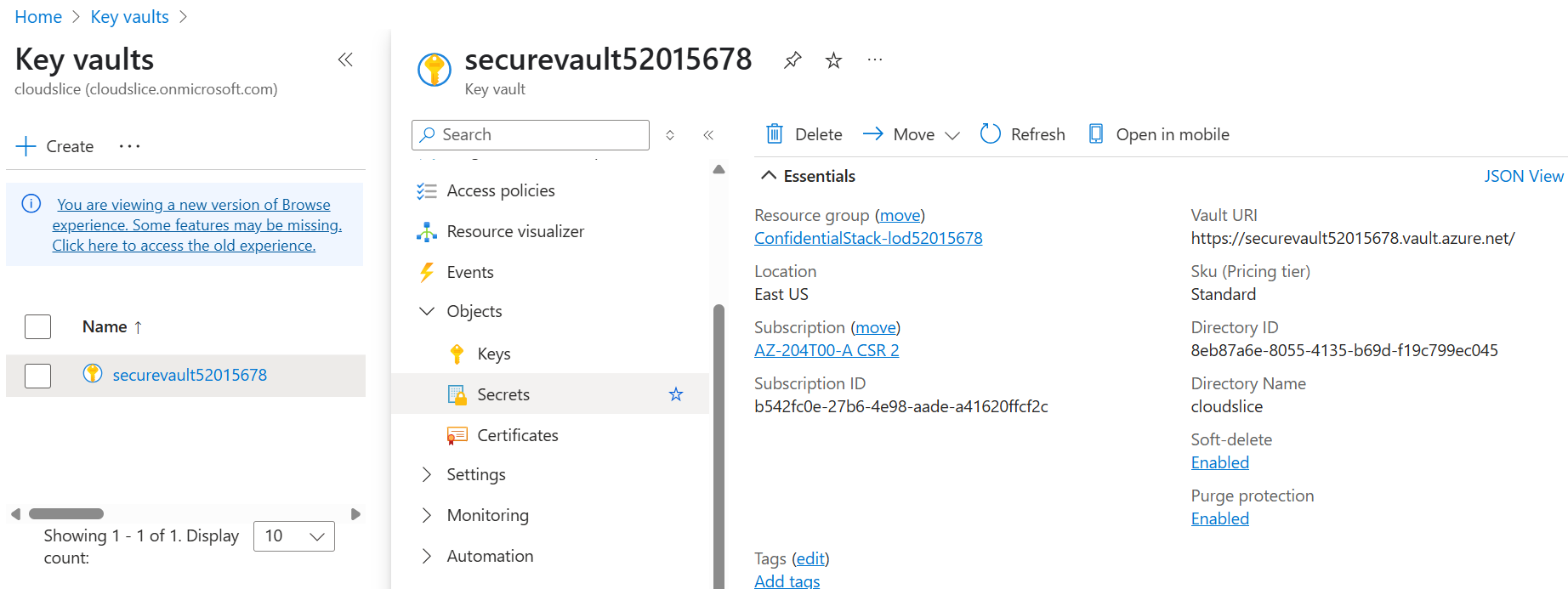


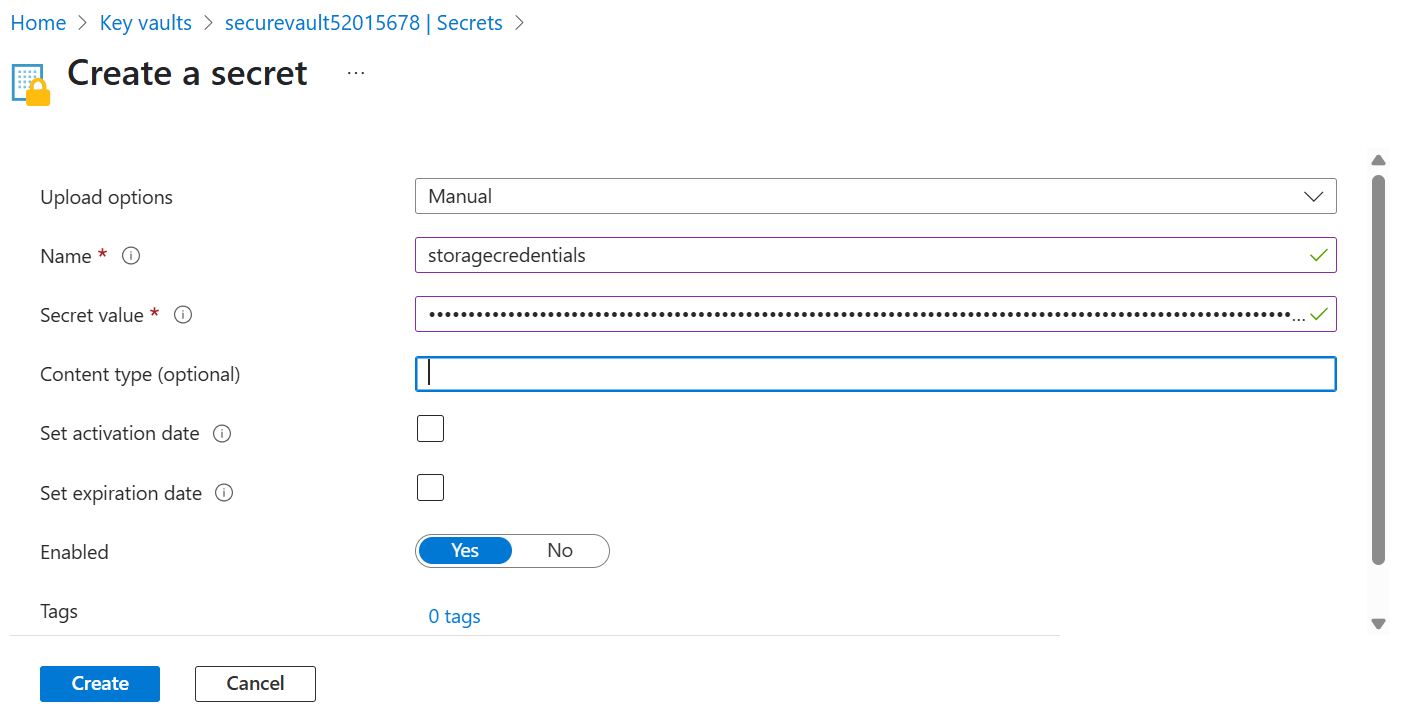




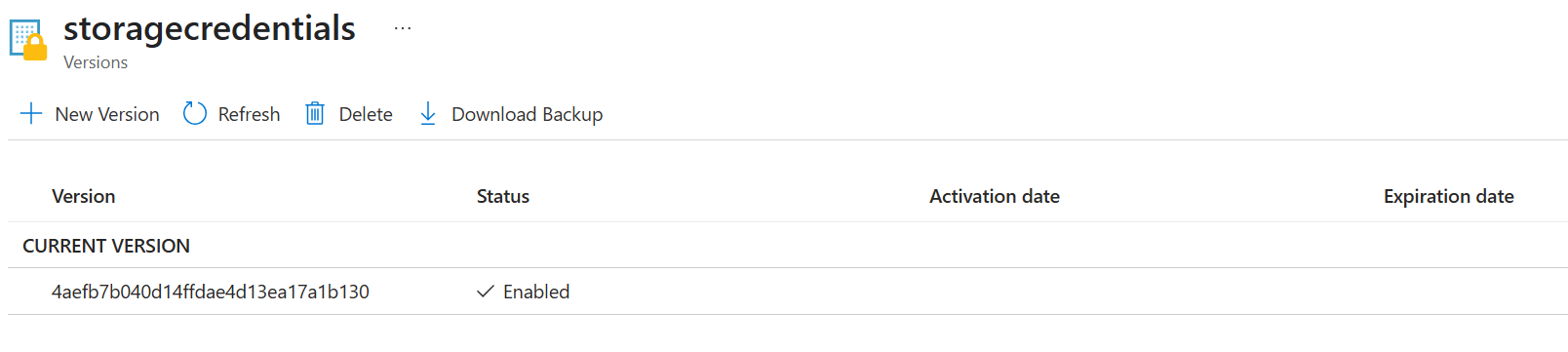








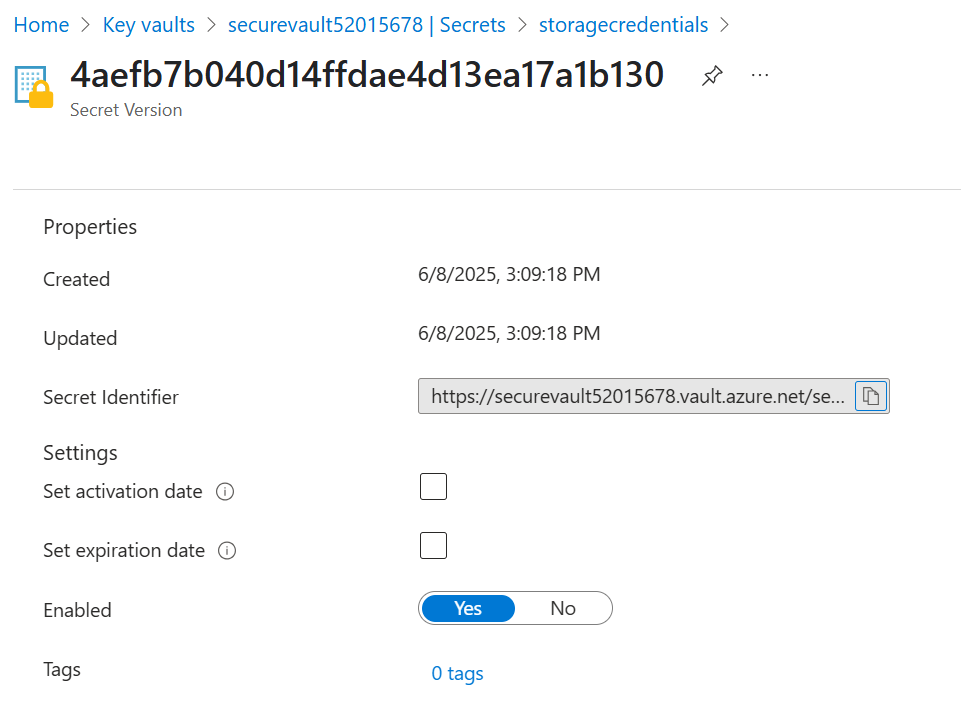
Secret value is blob storage connection string value.

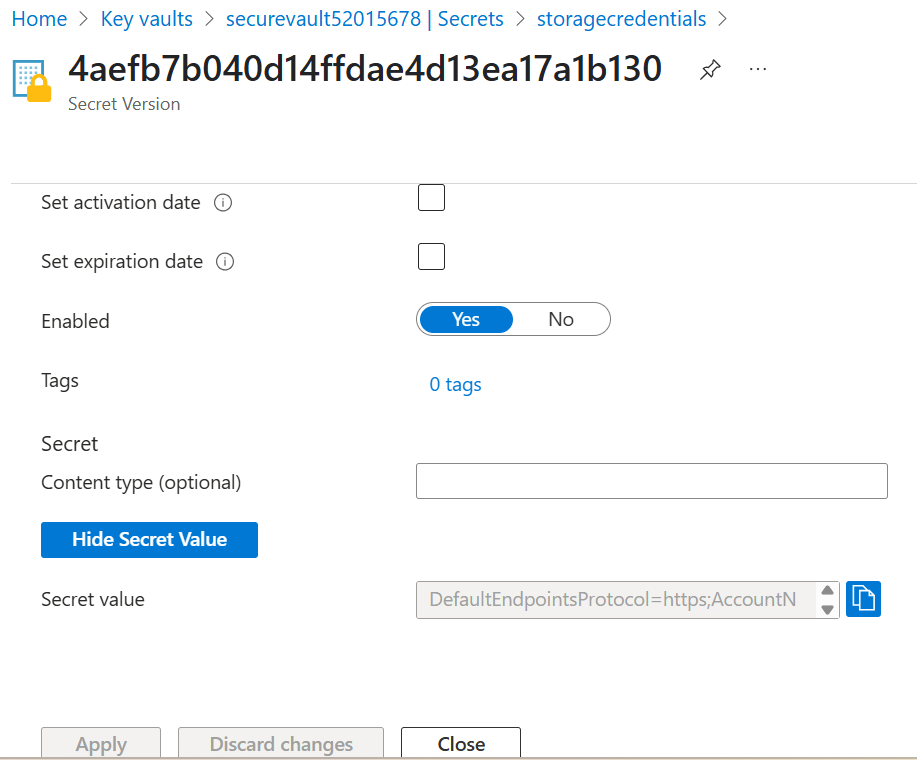


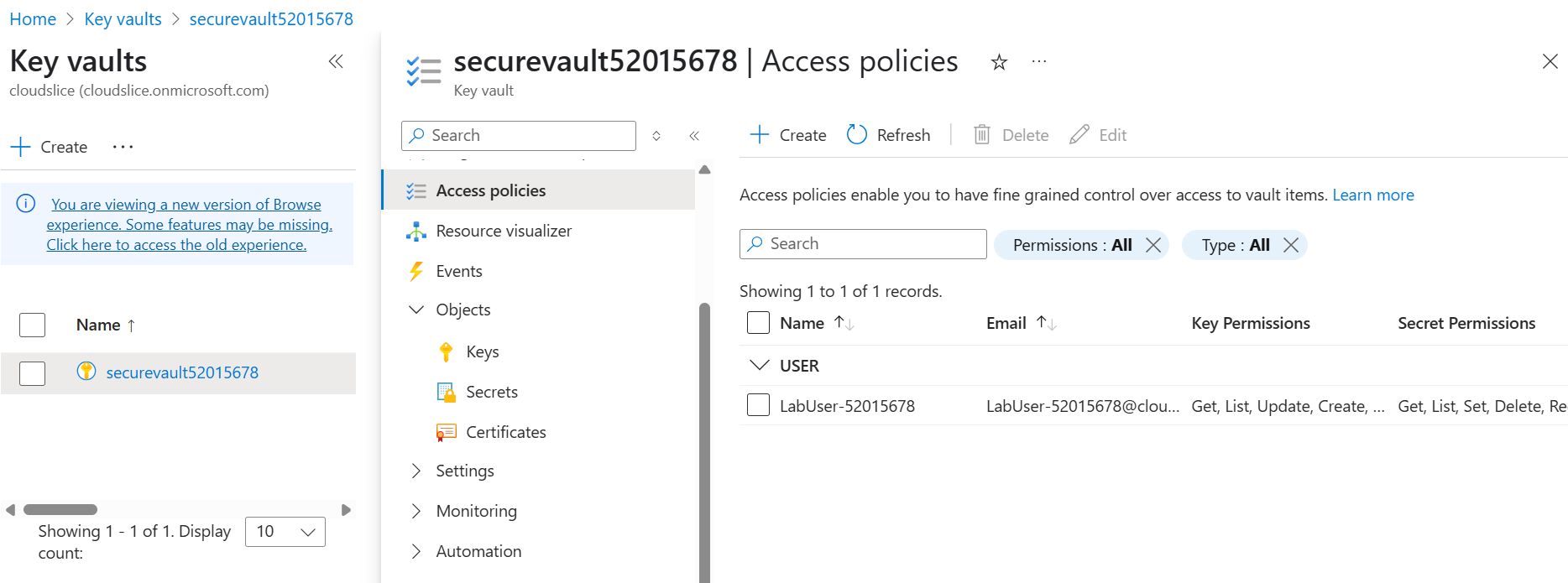
DefaultEndpointsProtocol=https;AccountName=securestor52015678;AccountKey=dAHk14/RmhMiemVqOHJte36Yk6d/Ra7G7HYCYVbwEK0dmZQc6+BHckHpfltK6G4QsaBIeZ6hikQa+ASt2bbxmA==;EndpointSuffix=core.windows.net

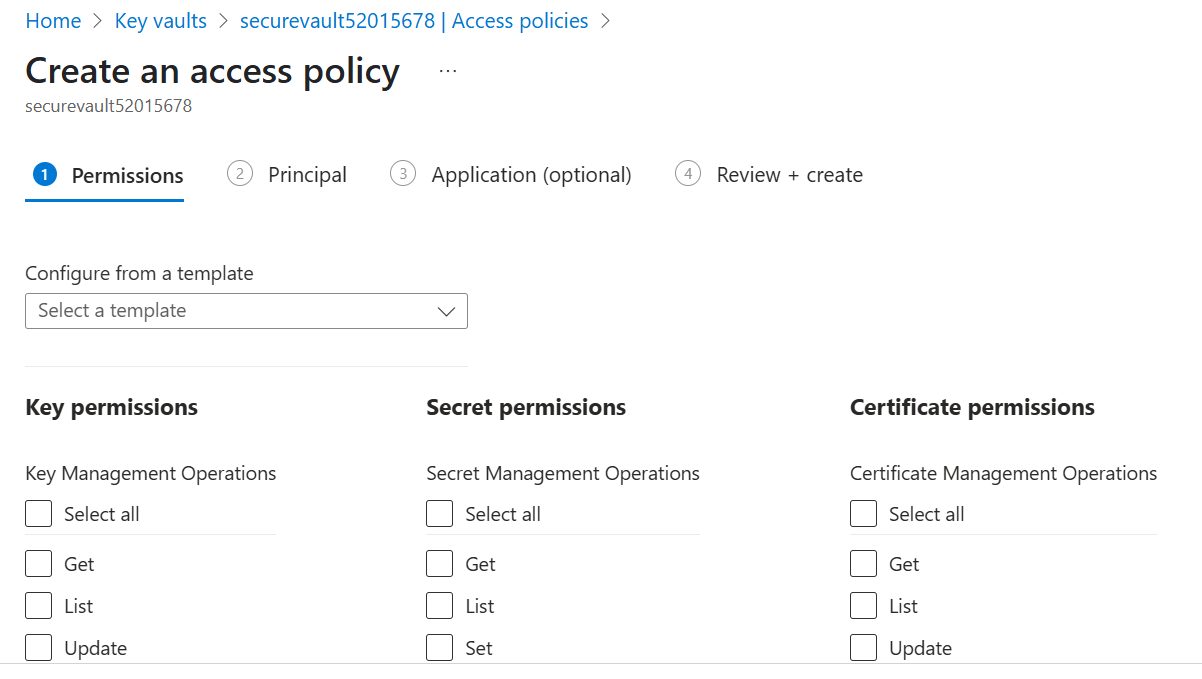
Secret Identifier

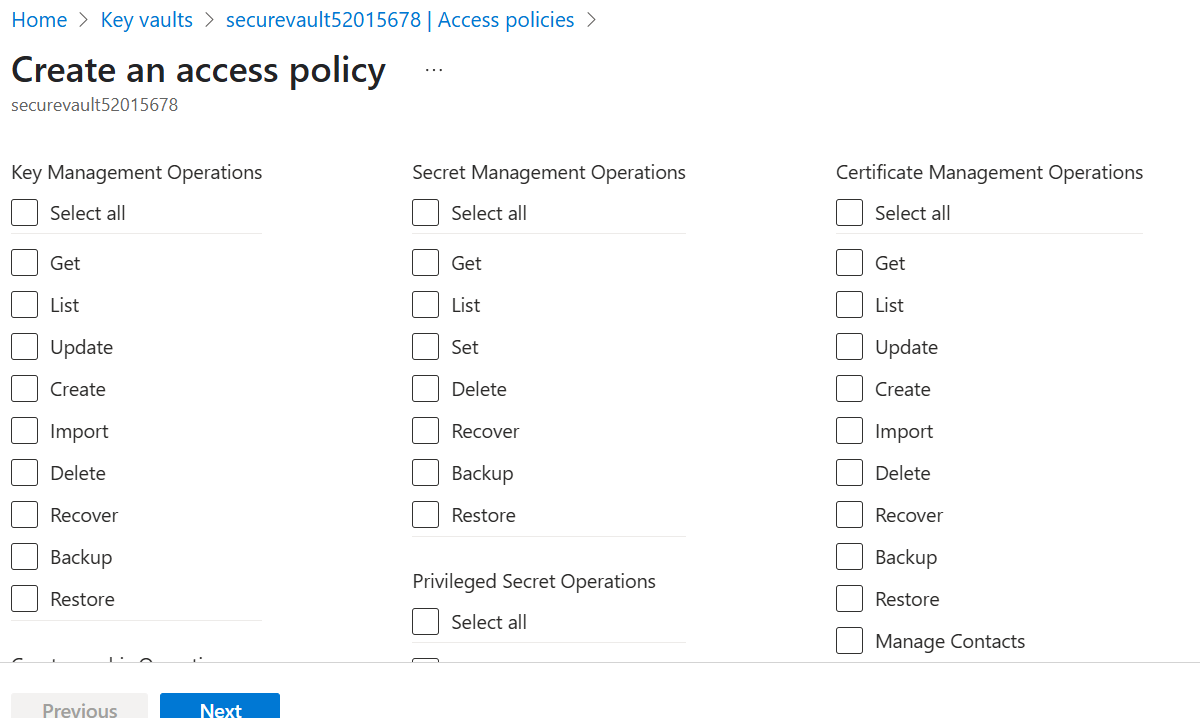
<https://securevault52015678.vault.azure.net/secrets/storagecredentials/4aefb7b040d14ffdae4d13ea17a1b130>

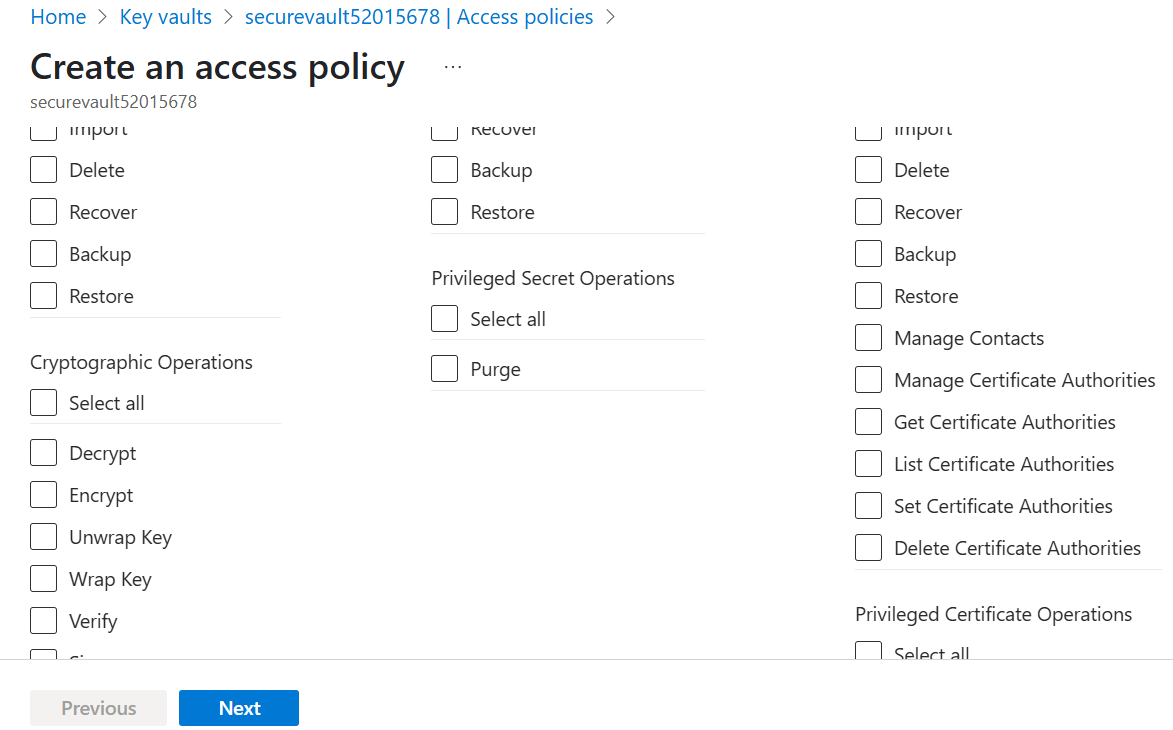


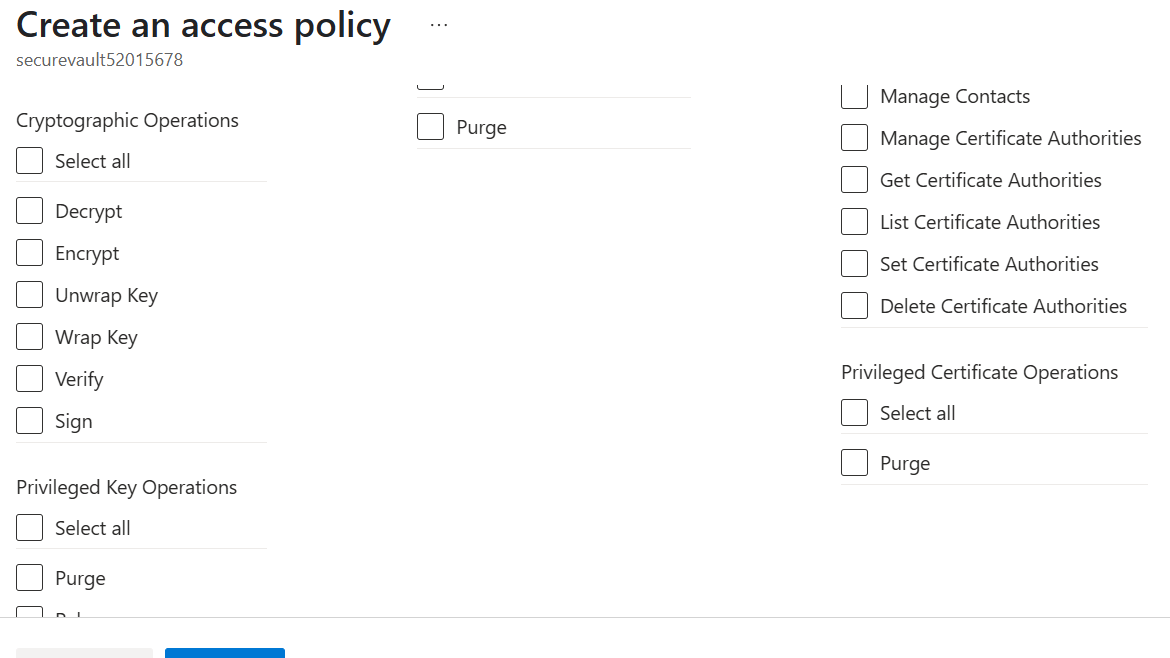


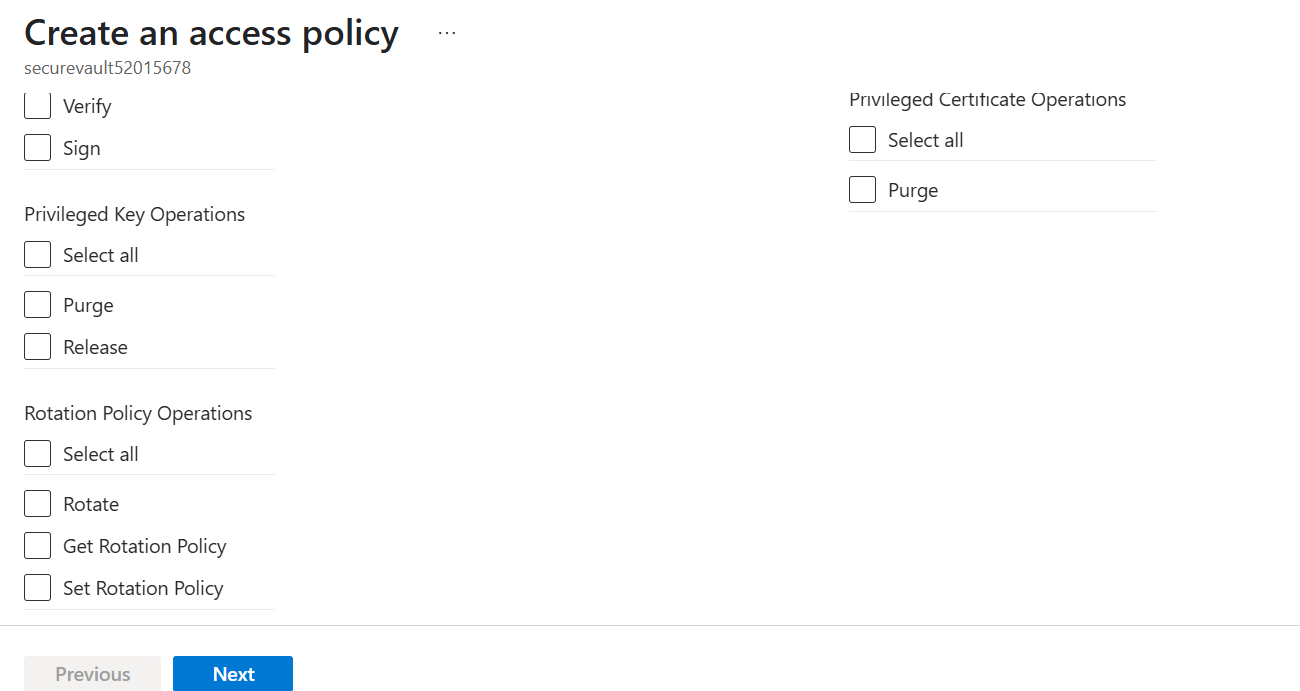


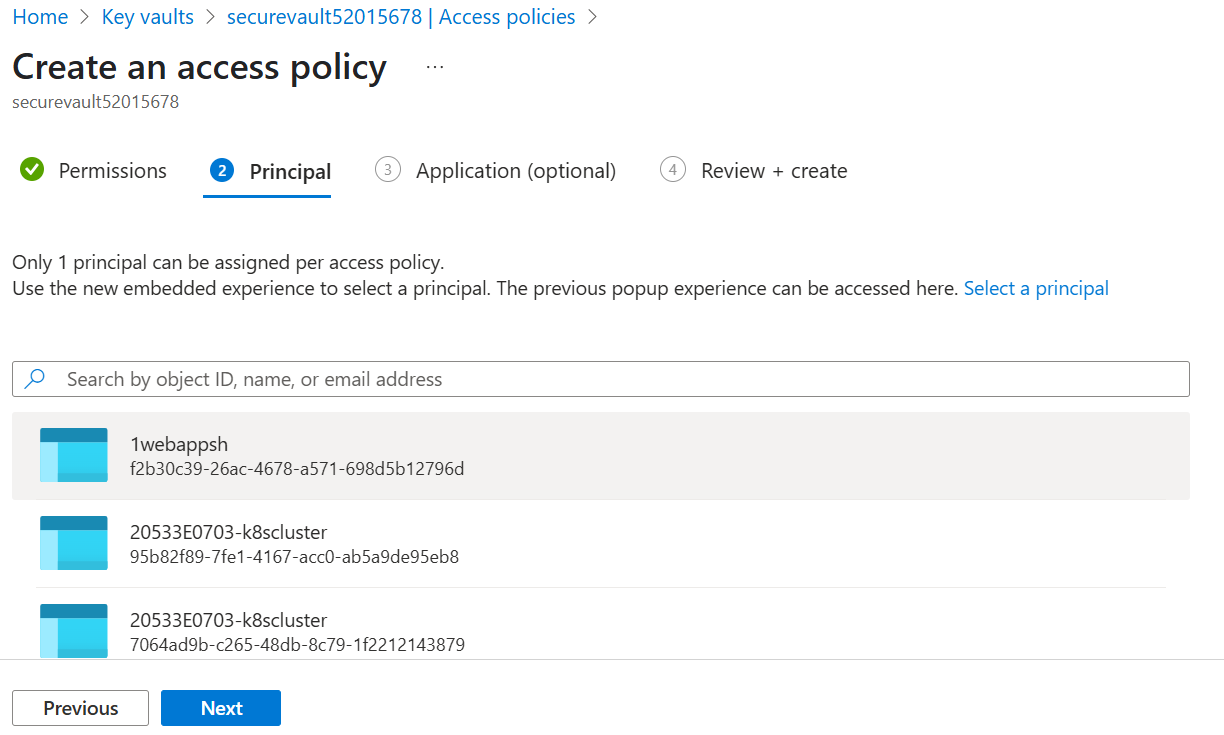


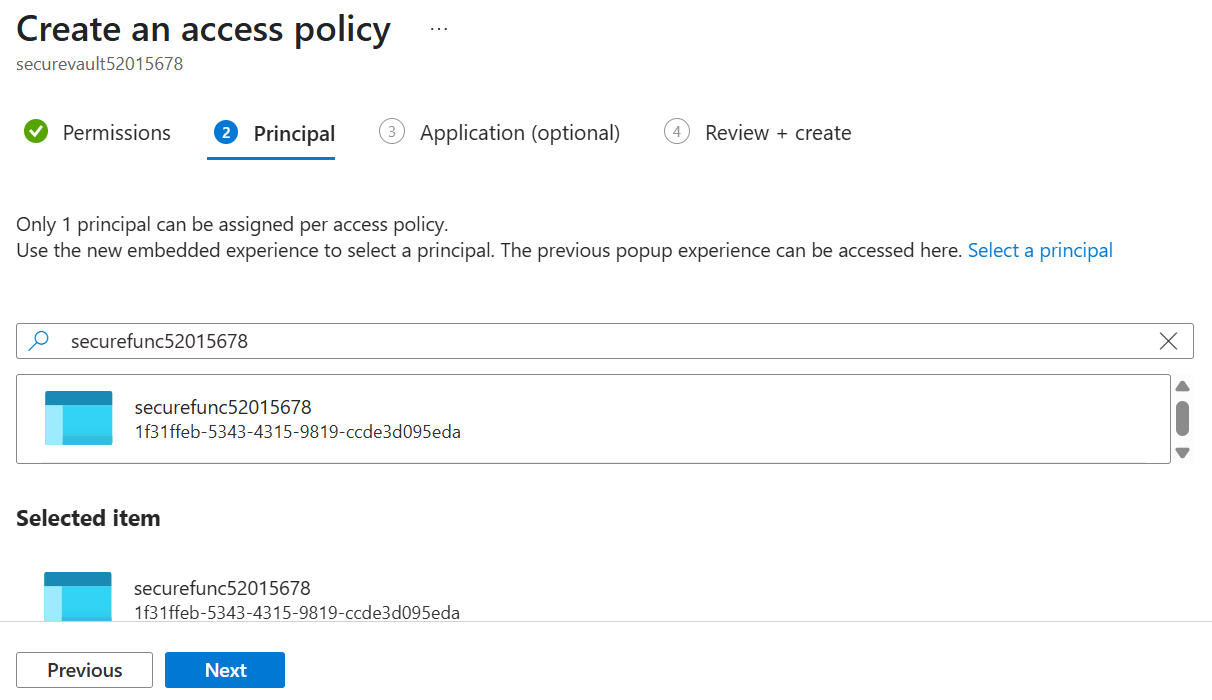












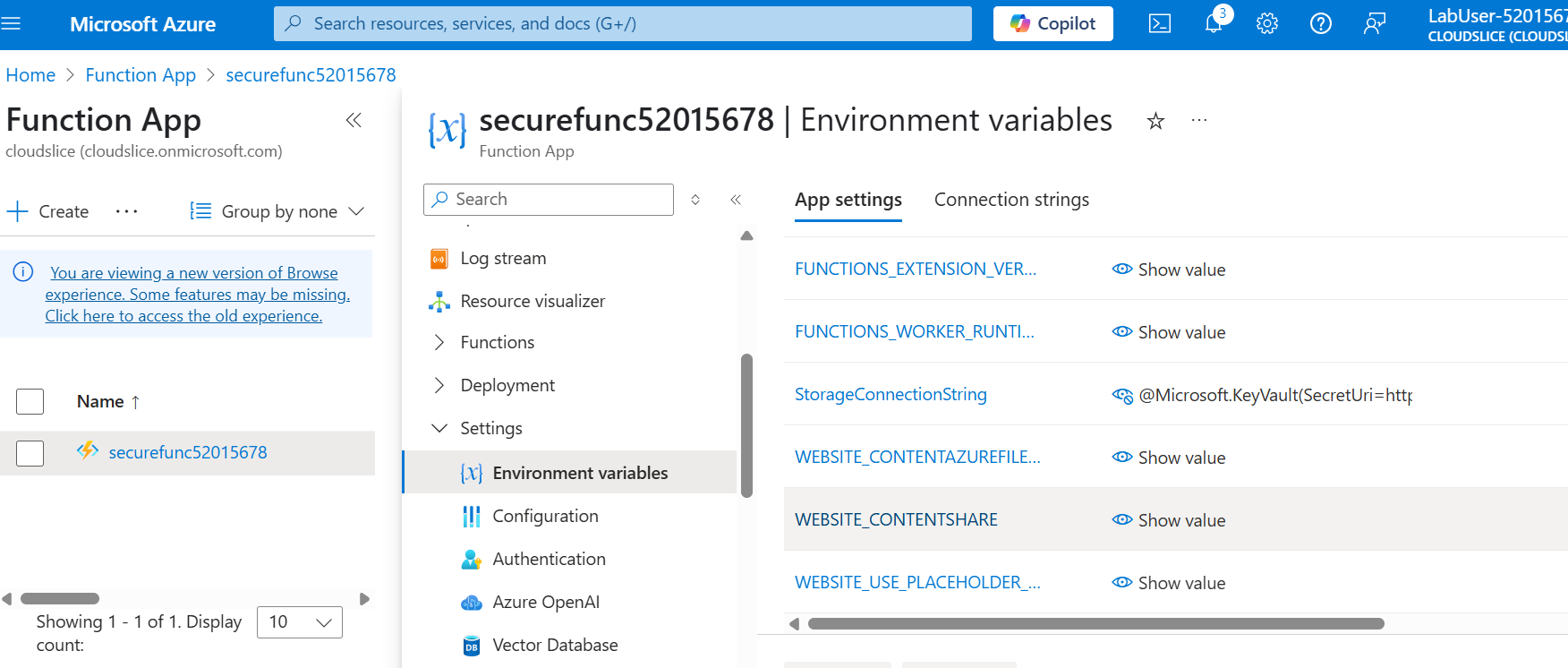
@Microsoft.KeyVault(SecretUri=<https://securevault52015678.vault.azure.net/secrets/storagecredentials/4aefb7b040d14ffdae4d13ea17a1b130>)

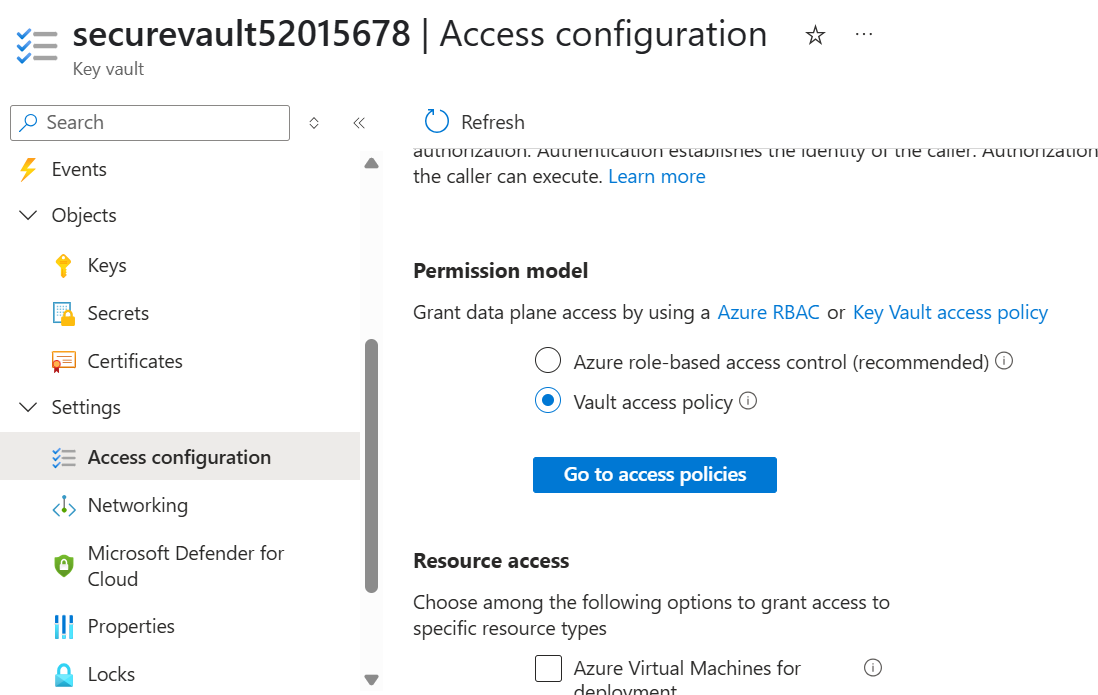
**Note**: For example, if your secret identifier is https://securevaultstudent.vault.azure.net/secrets/storagecredentials/17b41386df3e4191b92f089f5efb4cbf, the resulting value would be @Microsoft.KeyVault(SecretUri=https://securevaultstudent.vault.azure.net/secrets/storagecredentials/17b41386df3e4191b92f089f5efb4cbf).

1. On the **Function App** blade, in the **Settings** section, select the **Environment variables** link.
2. In the **App settings** tab, select **+ Add**. Enter the following information in the **Add/Edit application setting** pop-up dialog:

| **Setting** | **Action** |
| --- | --- |
| **Name** text box | enter StorageConnectionString |
| **Value** text box | Construct a value by using the following syntax: @Microsoft.KeyVault(SecretUri=<Secret Identifier>), where the <Secret Identifier> placeholder represents the secret identifier you recorded earlier in this exercise. |
| **Deployment slot setting** check box | Retain the default value |

1. **Note**: For example, if your secret identifier is https://securevaultstudent.vault.azure.net/secrets/storagecredentials/17b41386df3e4191b92f089f5efb4cbf, the resulting value would be @Microsoft.KeyVault(SecretUri=https://securevaultstudent.vault.azure.net/secrets/storagecredentials/17b41386df3e4191b92f089f5efb4cbf).
2. Select **Apply** to close the pop-up dialog and return to the **App settings** section.
3. At the bottom of the **App settings** section, select **Apply**.





**Required NuGet packages📦**1. Azure.Identity2**.**Azure.Extensions.AspNetCore.Configuration.Secrets  
3. Azure.Security.KeyVault.Secrets  
4. Microsoft.Extensions.Configuration.AzureKeyVault

**{**

**"$id": "1",**

**"id": "https://securevault52015678.vault.azure.net/secrets/storagecredentials/4aefb7b040d14ffdae4d13ea17a1b130",**

**"name": "storagecredentials",**

**"properties": {**

**"$id": "2",**

**"id": "https://securevault52015678.vault.azure.net/secrets/storagecredentials/4aefb7b040d14ffdae4d13ea17a1b130",**

**"vaultUri": "https://securevault52015678.vault.azure.net",**

**"name": "storagecredentials",**

**"version": "4aefb7b040d14ffdae4d13ea17a1b130",**

**"contentType": null,**

**"managed": false,**

**"keyId": null,**

**"enabled": true,**

**"notBefore": null,**

**"expiresOn": null,**

**"createdOn": "2025-06-08T09:39:18+00:00",**

**"updatedOn": "2025-06-08T09:39:18+00:00",**

**"recoverableDays": 90,**

**"recoveryLevel": "Recoverable",**

**"tags": {**

**"$id": "3"**

**}**

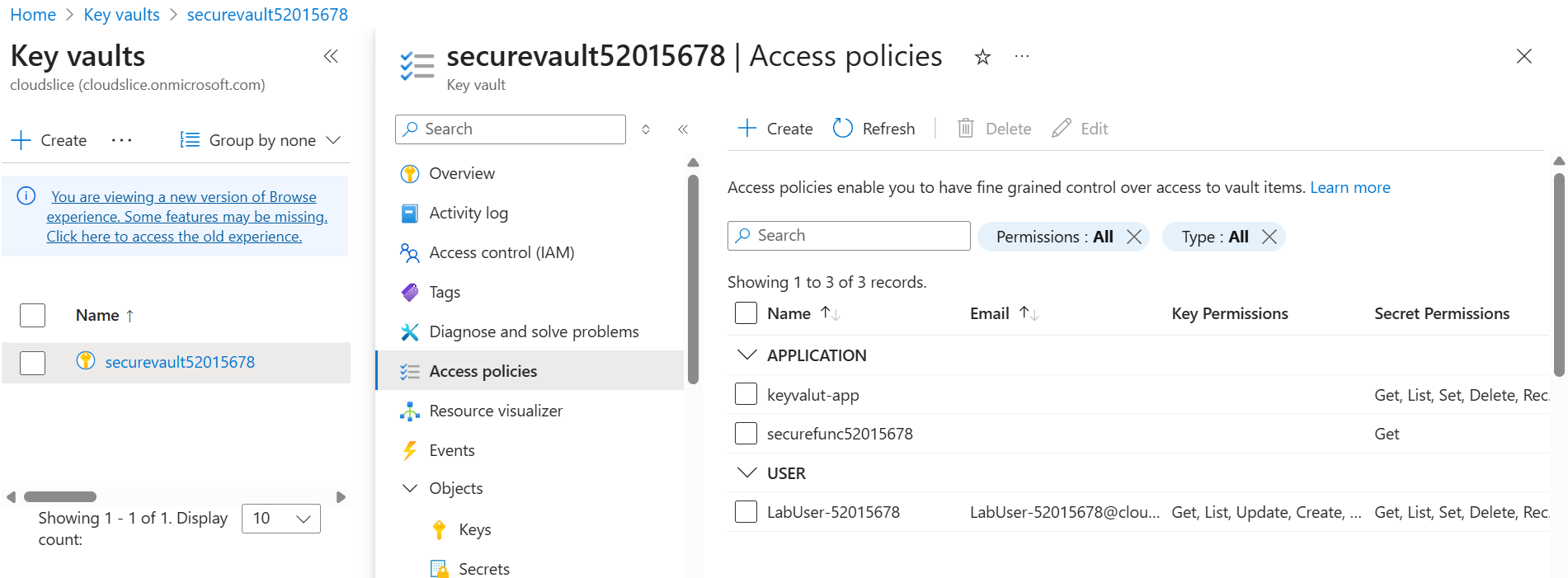
**},**

**"value": "DefaultEndpointsProtocol=https;AccountName=securestor52015678;AccountKey=dAHk14/RmhMiemVqOHJte36Yk6d/Ra7G7HYCYVbwEK0dmZQc6+BHckHpfltK6G4QsaBIeZ6hikQa+ASt2bbxmA==;EndpointSuffix=core.windows.net"**

**}**

For builder.Configuration.AddAzureKeyVault

We need to create app registration.



Here keyvault-app is app registration name.

(This is an alternative option that simplifies authentication).

1. **1. Configure Authentication:**

You need to configure your application to authenticate with Azure Key Vault. Common methods include:

* + **Managed Identities:** If your application is running in an Azure environment, you can use managed identities for seamless authentication without storing credentials.
  + **Client Secrets:** If you're running locally or in an environment without managed identities, you'll need to store a client secret (and optionally, a client ID and tenant ID).

DefaultAzureCredential automatically handles authentication using the managed identity.

To access a Key Vault key value in an Azure Web App using the secret identifier, you need to grant your web app access to the Key Vault and then retrieve the secret value using the secret identifier. You'll do this through Azure App Service Key Vault references or programmatically with the Azure Key Vault client library.

Steps to access a Key Vault secret in an Azure Web App:

1. **1. Grant Web App Access:**
   * In the Azure portal, navigate to your Key Vault.
   * Go to "Access policies" and add a new policy.
   * Select your web app as the "Principal" (choose the Managed Identity if your app uses one).
   * Grant "Get" and "List" permissions for secrets.
2. **2. Retrieve the Secret Identifier (URI):**
   * In the Key Vault, find the secret you want to access.
   * Copy the "Secret Identifier" or the secret's URI.
   * The URI is structured like this: {vaultBaseUrl}/secrets/{secret-name}/{secret-version}. You can omit the version for the latest version.
3. **3. Use the Secret in Your Web App:**
   * **Key Vault References (Recommended):**
     + In your web app's application settings, add a new app setting named after your secret (e.g., "MySecret").
     + Set the value of this app setting to a Key Vault reference. This will look like: @Microsoft.KeyVault(VaultName=your-keyvault-name,SecretName=your-secret-name) or a similar syntax with the full secret URI if you prefer to include the version.