

Create reference to Azure Key Vault content from function code

Requirements: Basic Azure Function knowledge and access to an Azure Key Vault & Azure Function.

This topic shows you how to work with secrets from Azure Key Vault in your Azure Functions code without requiring any code changes. Azure Key Vault is a service that provides centralized secrets management, with full control over access policies and audit history. This article uses managed identities to access other resources.

Add your secret to Azure Key Vault

First, the secret must be created in the Azure Key Vault. For this, an Azure Key Vault must exist and the permissions to create a new item must be available. There you can insert the secret that you want to use later in the code.

Create a secret ...

Upload options

Manual

Name * ⓘ

TestSecretLNCDOCS

Secret value * ⓘ

.....

Content type (optional)

Set activation date ⓘ

Set expiration date ⓘ

Enabled

Yes

No

Tags

0 tags

Create

Create managed identity of function

To be able to display the secret in the Function code, you have to activate the Managed Identity in the Function App. You can do this via the menu item "Identity" and then switch the status to "On" under "System assigned". Don't forget to save your selection.

Search

- Configuration
- Authentication
- Application Insights
- Identity**
- Backups
- Custom domains
- Certificates
- Networking
- Scale up (App Service plan)

System assigned User assigned

A system assigned managed identity is restricted to one per resource and is tied to the lifecycle of this resource. You can grant permissions to the managed identity by using Azure role-based access control (Azure RBAC). The managed identity is authenticated with Azure AD, so you don't have to store any credentials in code. [Learn more about Managed identities.](#)

Save Discard Refresh | Got feedback?

Status ⓘ

Off On

Create access policy

Then you can go back to the Key Vault and create a new access policy under "Access policies" -> "Create".

Search

+ Create Refresh | Delete Edit

Access policies enable you to have fine grained control over access to vault items. [Learn more](#)

Search Permissions: All X Type: All X

Showing 1 to 4 of 4 records.

<input type="checkbox"/> Name ↑↓	Email ↑↓
> APPLICATION	
> USER	

Objects

- Keys

There you have to select the desired permissions. The Azure Function then connects to the Key Vault with these permissions. To read only the secret content, only the "Get" permission under "Secret permissions" is used.

Configure from a template

Select a template

Key permissions

Key Management Operations

- Select all
- Get
- List
- Update
- Create
- Import
- Delete
- Recover
- Backup
- Restore

Cryptographic Operations

- Select all
- Decrypt
- Encrypt
- Unwrap Key
- Wrap Key
- Verify
- Sign

Privileged Key Operations

- Select all
- Purge
- Release

Rotation Policy Operations

- Select all
- Rotate
- Get Rotation Policy
- Set Rotation Policy

Secret permissions

Secret Management Operations

- Select all
- Get
- List
- Set
- Delete
- Recover
- Backup
- Restore

Privileged Secret Operations

- Select all
- Purge

Certificate permissions

Certificate Management Operations

- Select all
- Get
- List
- Update
- Create
- Import
- Delete
- Recover
- Backup
- Restore
- Manage Contacts
- Manage Certificate Authorities
- Get Certificate Authorities
- List Certificate Authorities
- Set Certificate Authorities
- Delete Certificate Authorities

Privileged Certificate Operations

- Select all
- Purge

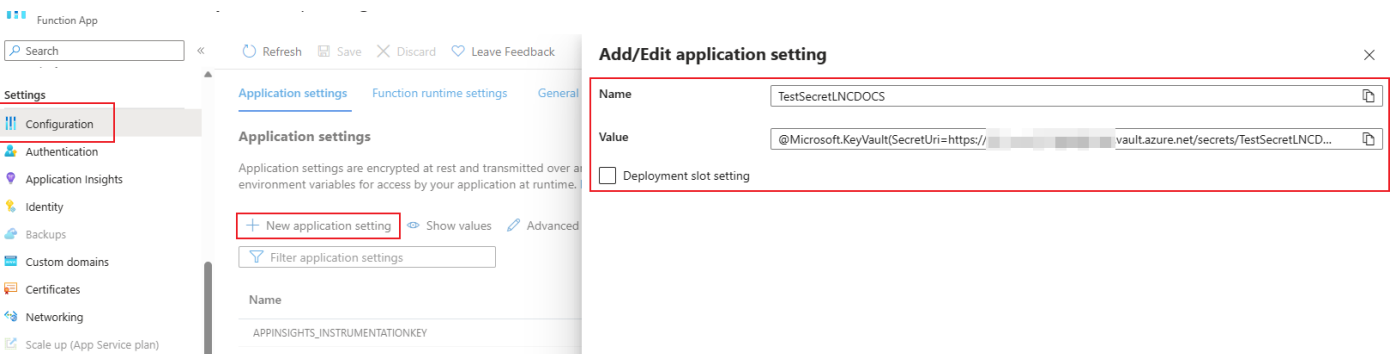
At the end you have to select the managed identity of the Azure Function and save the access policy.

Create environment variable link

Then a link to the Key Vault can be created. To do this, you must create a new application secret under "New application setting" in the Function App under "Configuration".

Here you first enter the name of the variable that you want to address in the code. Under "Value" you have to insert the following content and complete it with your values:

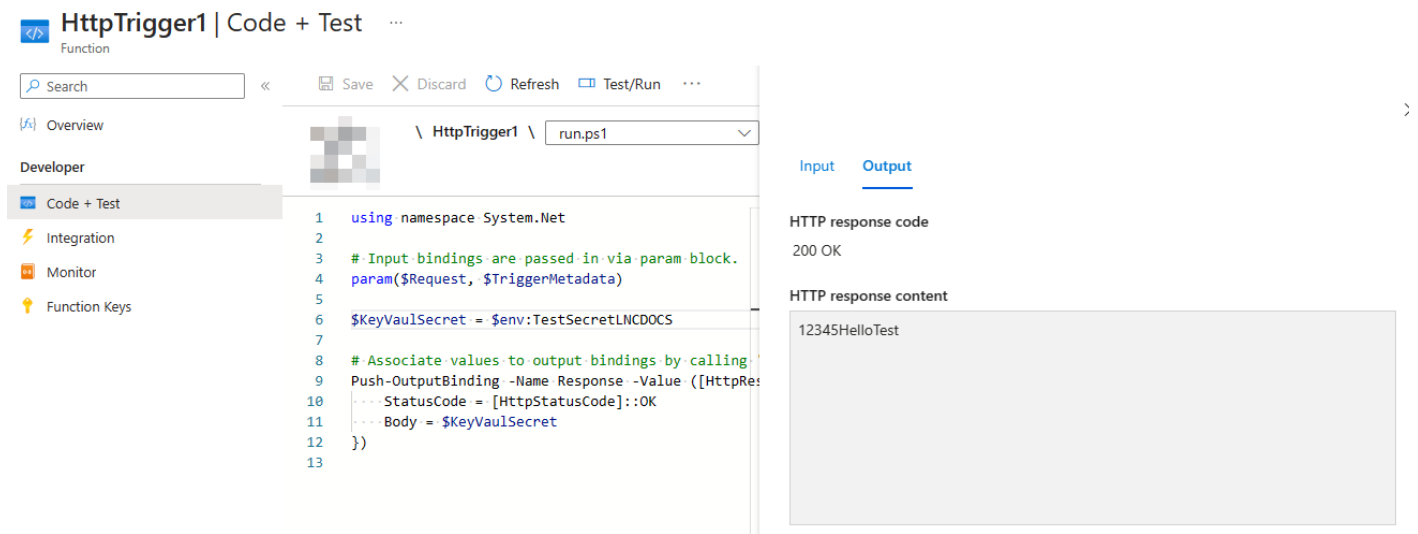
```
@Microsoft.KeyVault(SecretUri=https://<keyvaultname>.vault.azure.net/secrets/<secretname>)
```



Get environment variable content

Then you can read the variable in your function code using the environment as follows:

```
$env:<secretname>
```



Thus, the values can be stored securely without all user accounts needing authorization.

Revision #5

Created 2023-05-24 09:15:31 UTC

Updated 2025-08-06 15:26:32 UTC by Luca Noah Caprez