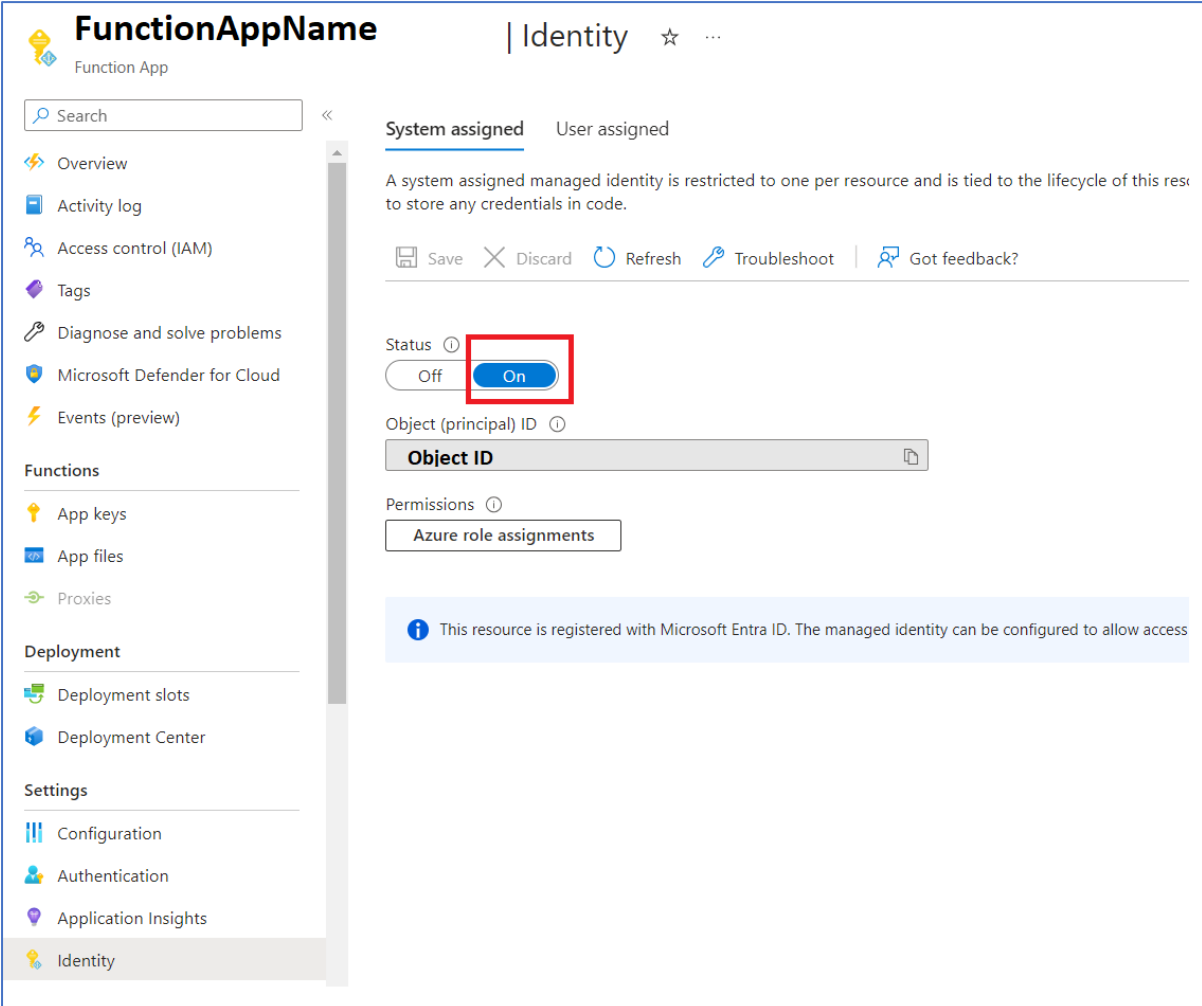


Assuming...

A function app is set up

## Go to FunctionApp > Identity > System assigned = On

Now your FunctionApp is visible in the Key Vault for when you want to attach an Access Policy to the FunctionApp.



The screenshot displays the Azure portal interface for a Function App named 'FunctionAppName'. The 'Identity' section is active, showing the 'System assigned' tab. The 'Status' is set to 'On', which is highlighted with a red box. Below the status, there is a field for 'Object (principal) ID' with a value of 'Object ID'. The 'Permissions' section shows 'Azure role assignments'. A blue information banner at the bottom states: 'This resource is registered with Microsoft Entra ID. The managed identity can be configured to allow access'.

# Create a Key Vault 1

1. Select appropriate subscription and resource group.
2. Give the Key vault a name and select the appropriate settings for your use case.

## Create a key vault ...

**Project details**  
Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription \*

Resource group \*  [Create new](#)

**Instance details**

Key vault name \* ⓘ

Region \*

Pricing tier \* ⓘ

**Recovery options**  
Soft delete protection will automatically be enabled on this key vault. This feature allows you to recover or permanently delete a key vault and secrets for the duration of the retention period. This protection applies to the key vault and the secrets stored within the key vault.

To enforce a mandatory retention period and prevent the permanent deletion of key vaults or secrets prior to the retention period elapsing, you can turn on purge protection. When purge protection is enabled, secrets cannot be purged by users or by Microsoft.

Soft-delete ⓘ  Enabled

Days to retain deleted vaults \* ⓘ

Purge protection ⓘ  Disable purge protection (allow key vault and objects to be purged during retention period)  
 Enable purge protection (enforce a mandatory retention period for deleted vaults and vault objects)

# Create an Access Policy (create a Key Vault 2)

1. We will use “Vault access policy” as a permission model for our specific use case.
2. Create an Access Policy with “Get” and “List” for secret permissions.
3. Assign the principal (in our case, the FunctionApp). It should pop up in the tab “Principal” now that we have turned the system assigned managed identity on.
4. Create the Access Policy.

The screenshot shows the 'Create an access policy' page in the Azure portal. The page is divided into two main sections: 'Create a key vault' on the left and 'Create an access policy' on the right.

**Create a key vault:**

- Access configuration:** The 'Access configuration' tab is active. It includes sections for 'Permission model' (with 'Vault access policy' selected), 'Resource access' (with 'Azure Virtual Machines for deployment', 'Azure Resource Manager for template deployment', and 'Azure Disk Encryption for volume encryption' options), and 'Access policies' (with a '+ Create' button highlighted by a red box).

**Create an access policy:**

- Permissions:** The 'Permissions' step is active. It includes a 'Configure from a template' dropdown menu.
- Key permissions:** Under 'Key Management Operations', 'Get', 'List', 'Update', 'Create', 'Import', and 'Delete' are listed. 'Get' and 'List' are checked.
- Secret permissions:** Under 'Secret Management Operations', 'Get', 'List', 'Set', 'Delete', 'Recover', and 'Backup' are listed. 'Get' and 'List' are checked, and these two options are highlighted with a red box.

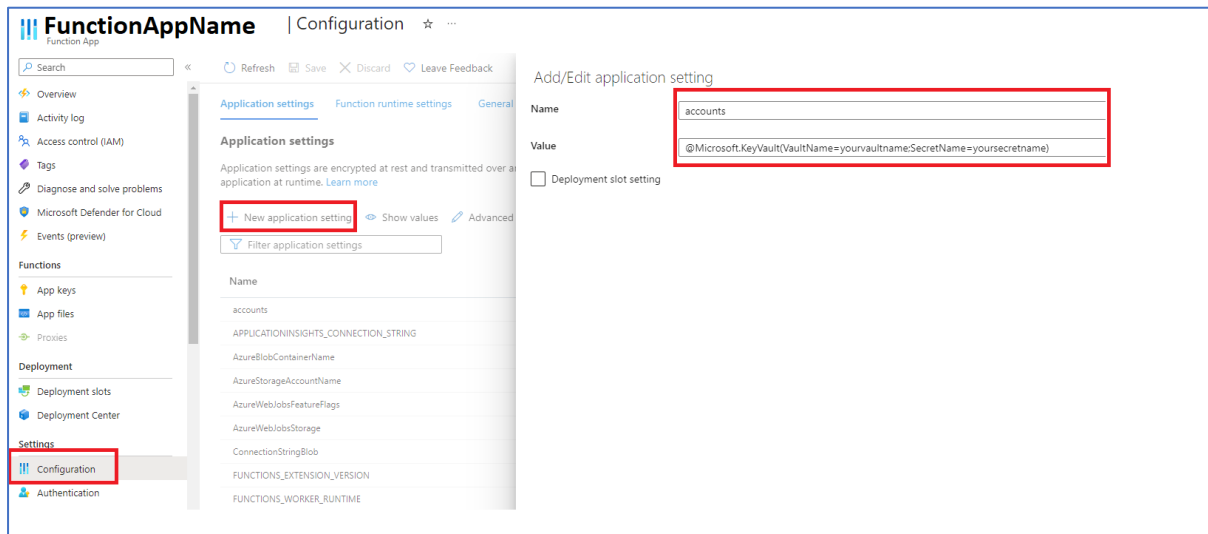
## **Other settings & creating the secret (create a Key Vault 2)**

1. For our specific use case I will leave all other settings on the subsequent tabs as default. Your requirements may vary.
2. Create a secret in the Key Vault.

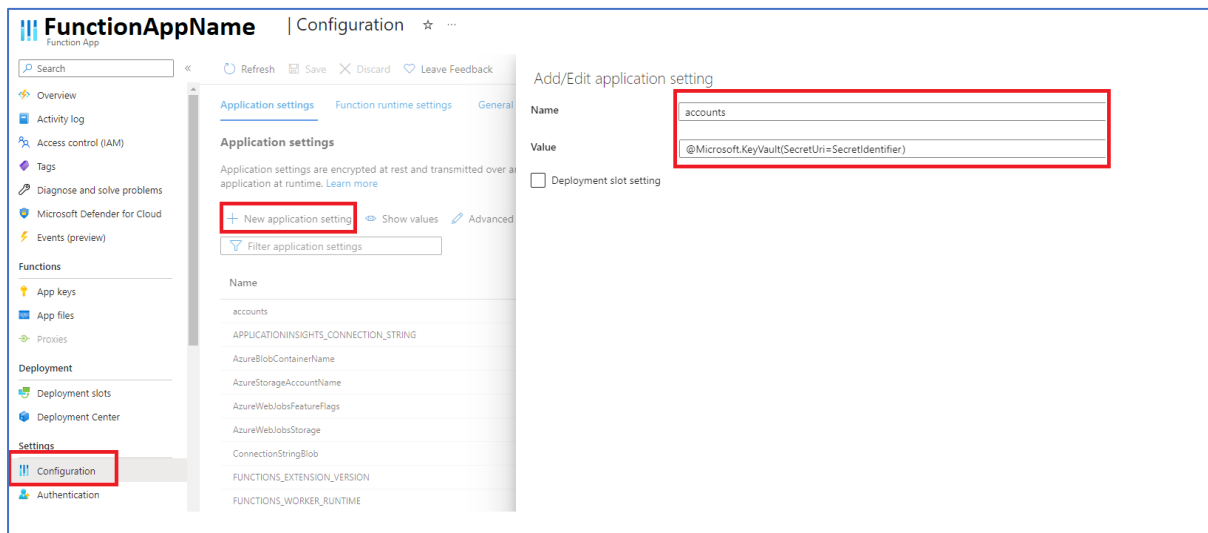
# Linking to Key Vault Secret in FunctionApp environment variables (Application settings)

Refer to the secret in one of two ways

1. With this method it takes some time before changes in the secret are synchronised in Azure Functions.



2. With this method changes in the secret are recognized immediately.



# Retrieving the secret in your script through the set environment variable

You can retrieve the values in this environment variable (named 'accounts' in this specific case) by using the Python code below:

```
os.environ.get('accounts')
```