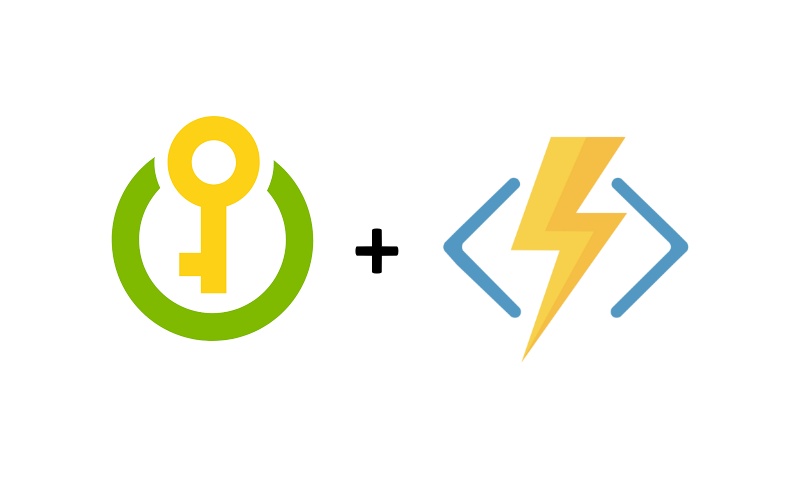
# **Integrate Key Vault Secrets With Azure Functions**



Your application requires service passwords, connection strings, and other secret configuration values to do its job. Storing and handling secret values is risky, and every usage introduces the possibility of leakage. Azure Key Vault, in combination with managed identities for Azure resources, enables your Azure web app to access secret configuration values easily and securely without needing to store any secrets in your source control or configuration.

# **What is Azure Key Vault?**

Azure Key Vault is a *secret store*: a centralized cloud service for storing application secrets - configuration values like passwords and connection strings that must remain secure at all times. Key Vault helps you control your applications' secrets by keeping them in a single central location and providing secure access, permissions control, and access logging.

The main benefits of using Key Vault are:

* Separation of sensitive application information from other configuration and code, reducing risk of accidental leaks
* Restricted secret access with access policies tailored to the applications and individuals that need them
* Centralized secret storage, allowing required changes to happen in only one place
* Access logging and monitoring to help you understand how and when secrets are accessed

## **What is a secret in Key Vault?**

In Key Vault, a secret is a name-value pair of strings. Secret names must be 1-127 characters long, contain only alphanumeric characters and dashes, and must be unique within a vault. A secret value can be any UTF-8 string up to 25 KB in size.

Key Vault supports two additional kinds of secrets beyond strings — *keys* and *certificates* — and provides useful functionality specific to their use cases. This module concentrates on secret strings like passwords and connection strings.

**Vault authentication and permissions**

Azure Key Vault's API uses Azure Active Directory to authenticate users and applications. Vault access policies are based on actions, and are applied across an entire vault. For example, an application with Get (read secret values), List (list names of all secrets), and Set (create or update secret values) permissions to a vault is able to create secrets, list all secret names, and get and set all secret values in that vault.

All actions performed on a vault require authentication and authorization — there is no way to grant any kind of anonymous access.

***Tip:***

When granting vault access to developers and apps, grant only the minimum set of permissions needed. Permissions restrictions help avoid accidents caused by code bugs and reduce the impact of stolen credentials or malicious code injected into your app.

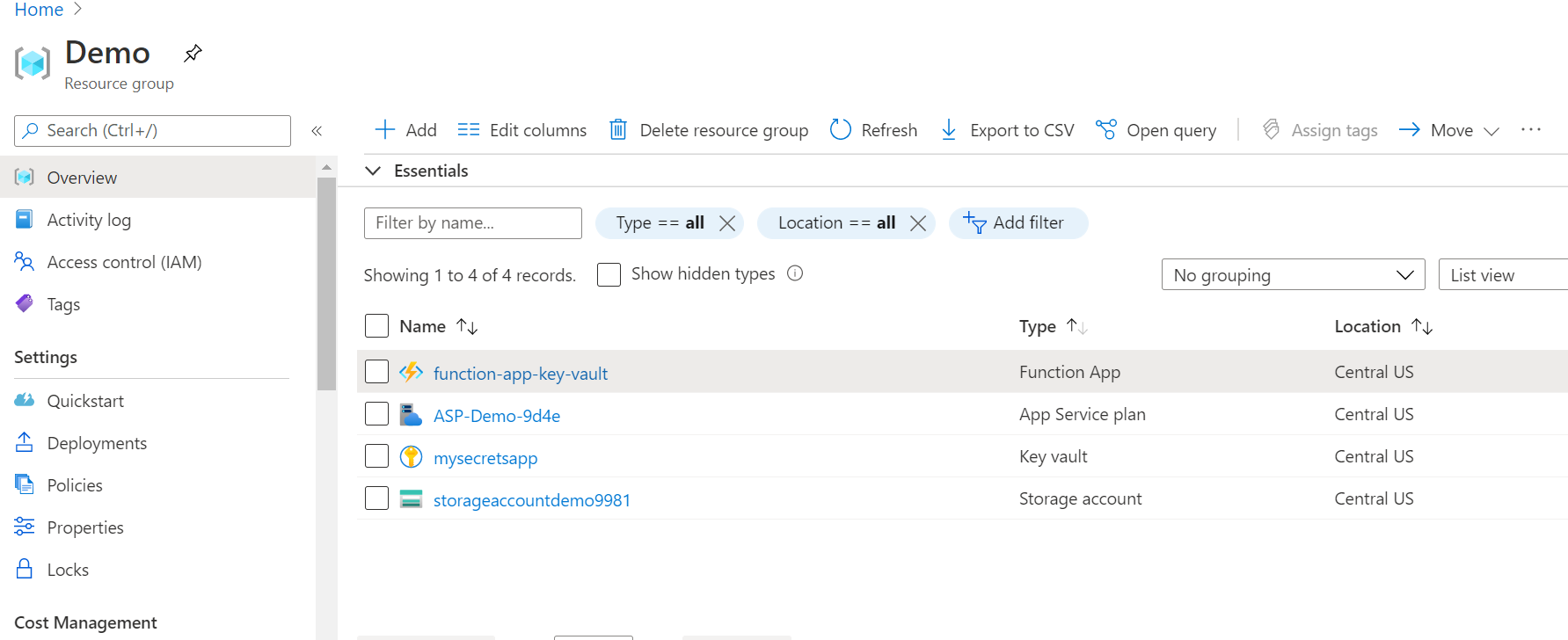
* Developers will usually only need Get and List permissions to a development-environment vault. Some engineers will need full permissions to change and add secrets when necessary.
* For apps, often only Get permissions are required. Some apps may require List depending on the way the app is implemented.

There are a lot of different scenarios where Azure Functions can be used. Interaction with other Azure components is possible but we can also call external services. The question in how to secure credentials and store them in the right place - in this case Azure Key Vault.

***In this Documentation, how to securely pass secrets from the Azure Key Vault to the Azure Function App. With this solution we are not storing credentials directly in the configuration but we inject it from the Key Vault.***

**Create Azure Key Vault and Azure Function App**

First of all we have to create a sample Key Vault and Azure Function App. Below here are my two resources created:

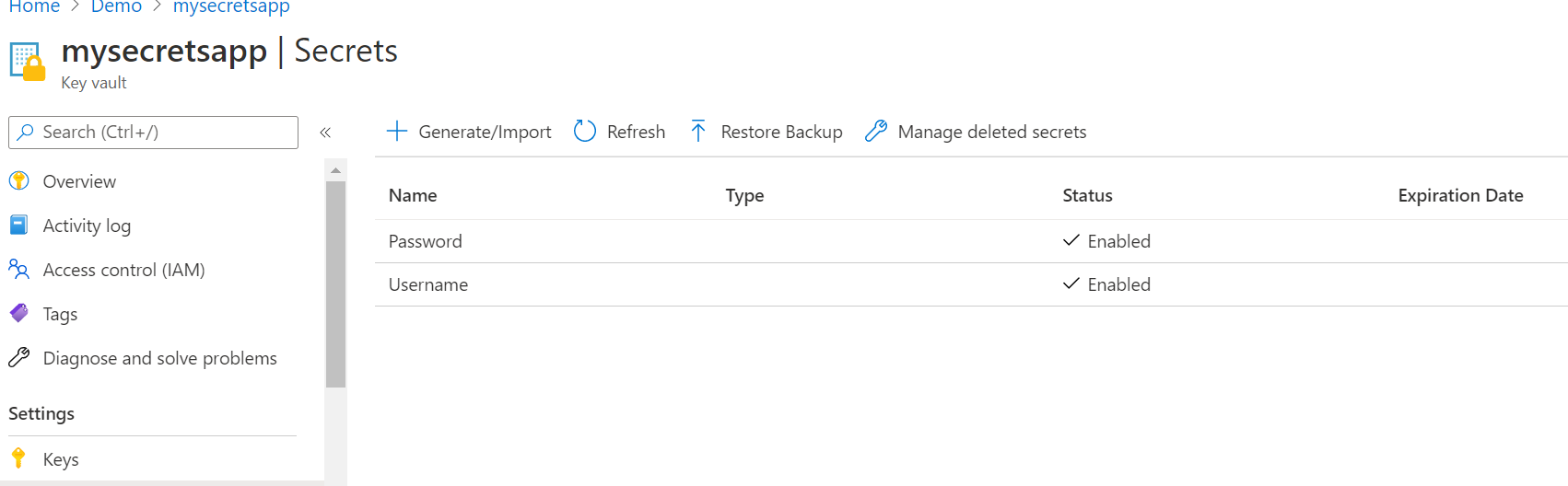


**Add secrets to the Azure Key Vault**

Credentials should be stored in the secure way using Azure Key Vault secrets. Let's add two secrets:

* Username: sampleazure@com
* Password: Test1234@

We will use these two secrets in the Azure Function later.

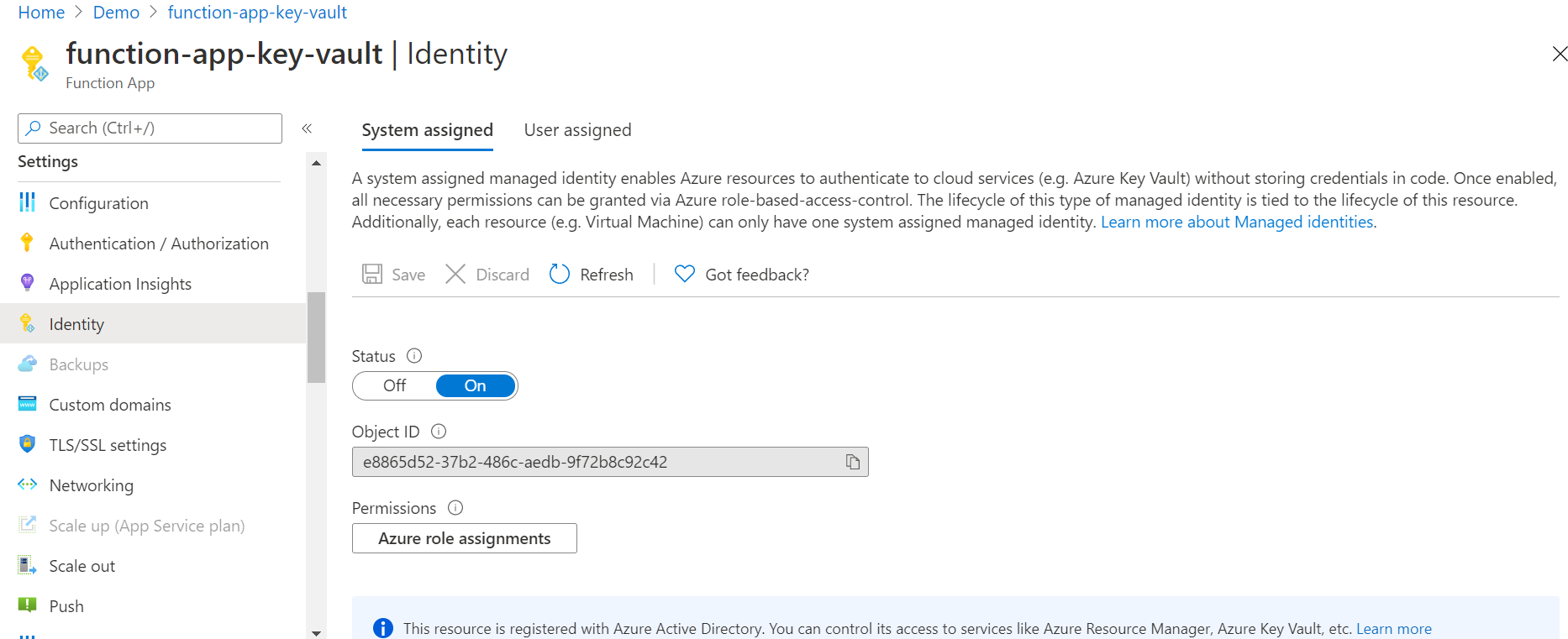


## **Enable system-assigned managed identity for the Function App**

Before we can use Azure Key Vault secrets in the Azure Function code, we have to assign a Managed Identity to it.

Navigate to the “Settings” tab and select “Identity”

Set “Status” to “On” and then click “Save” button - confirm information displayed in the dialog

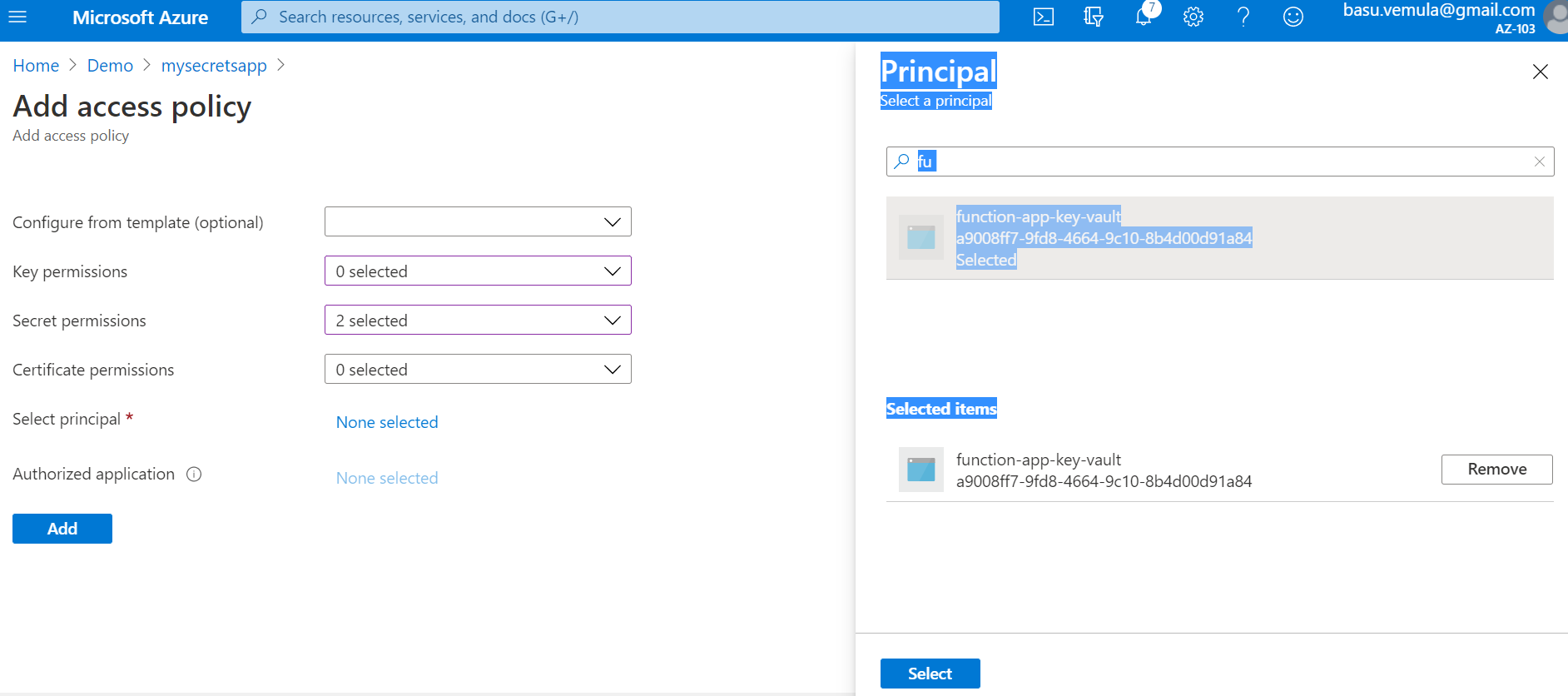


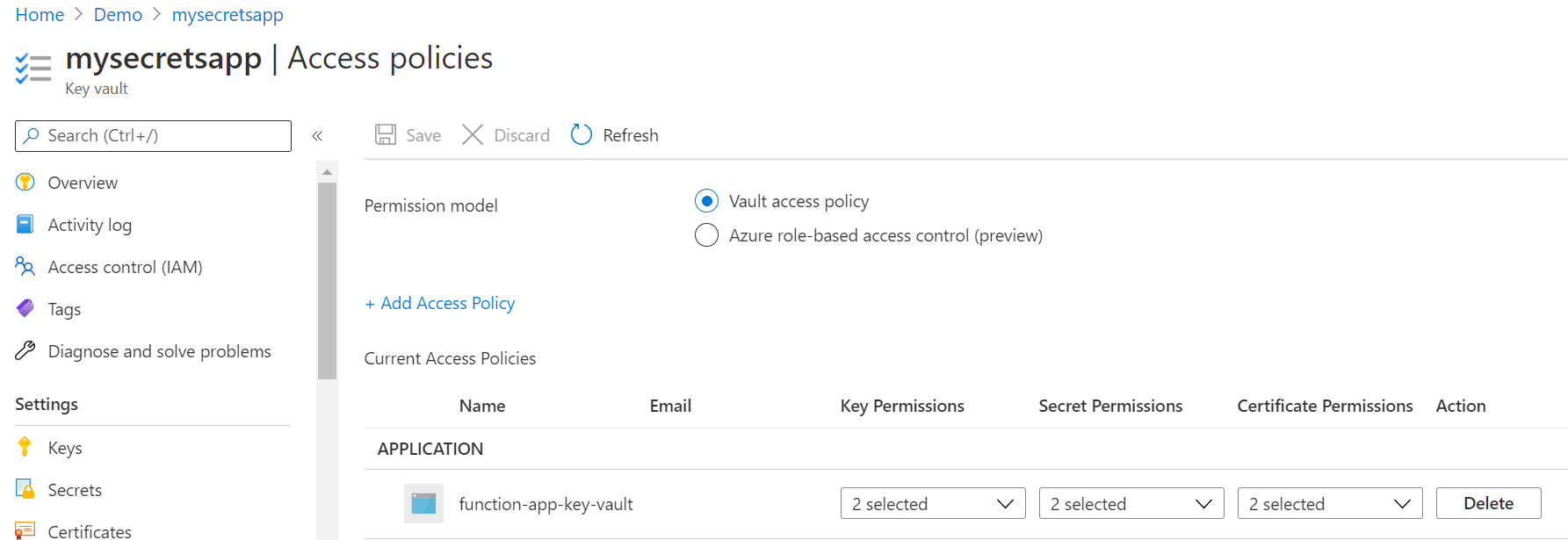
## **Add Key Vault access policy for the Function App**

Now we need to add access policy for the Function App so it will be able to read secrets from the Azure Key Vault.

Select “Access policies” tab:

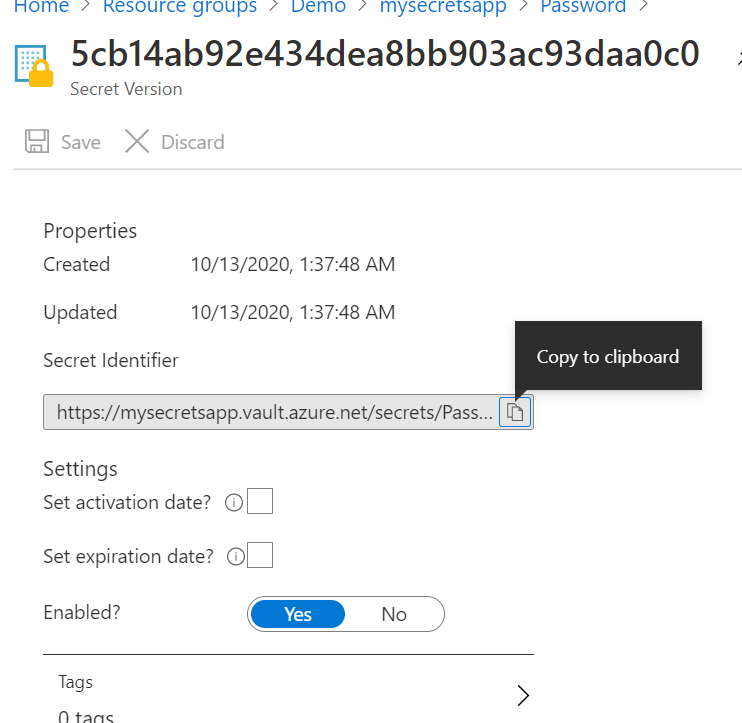
Find Function and select it in the “Service Principal” section. Please note that we need to select “Get” and “List” permissions:





## **Add Key Vault secrets reference in the Function App configuration**

Now we need to refer to the Key Vault secrets in the Function App configuration. Navigate to the “Secrets” tab in the Key Vault and copy “Secret Identifier” for both “Username” and “Password” secrets. We will use them soon:



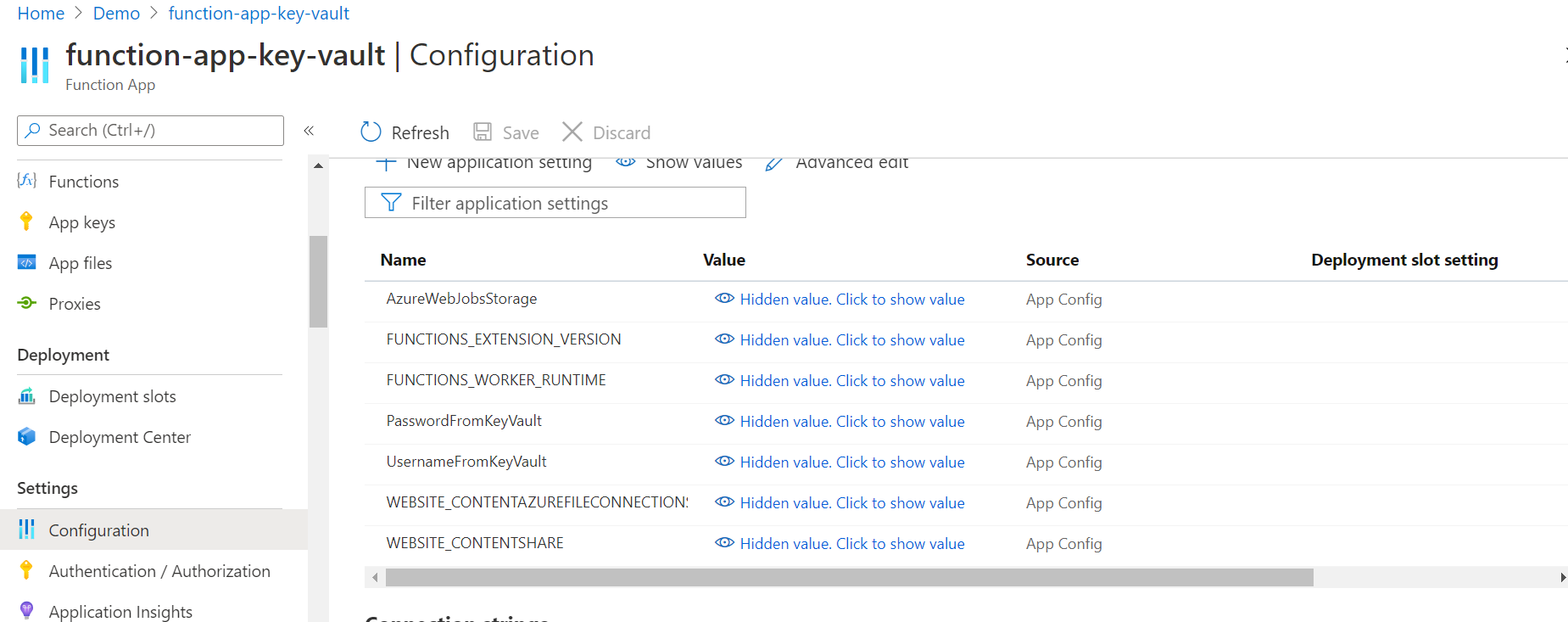
Now we will add these identifiers to the “Configuration” of the Function App. There will be two settings added:

UsernameFromKeyVault:

@Microsoft.KeyVault(SecretUri={copied identifier for the username secret})

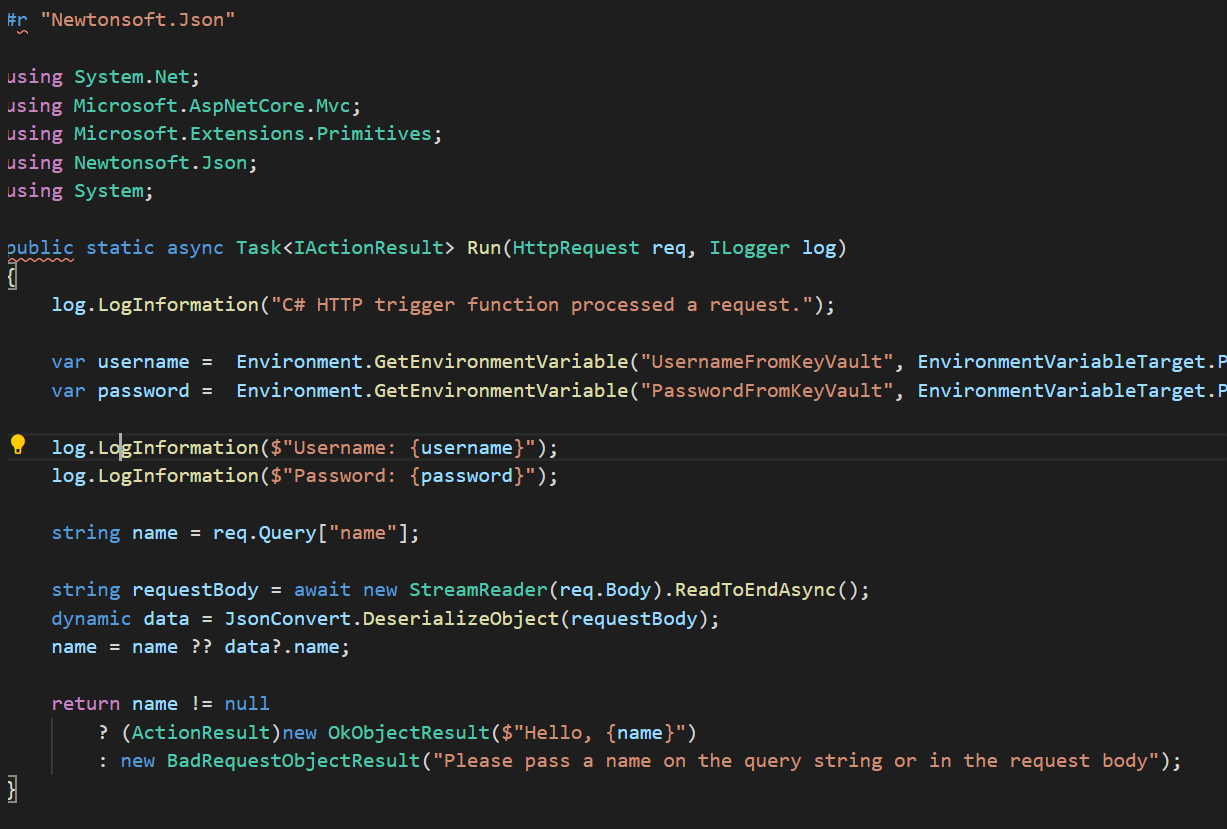
PasswordFromKeyVault:

@Microsoft.KeyVault(SecretUri={copied identifier for the password secret})



## **Test secrets injection**

Now we need to modify the source code of the Function App. The quickest way to check if secrets injection works is to use a logger. We will try to display injected secrets.



Save and run the function. Look at the logs - secrets should be displayed there:

