TLS Proxy: Custom domain support for Key Vault using Application Gateway

Contents

Overview

KeyVault supports KeyVault URLs like https://tlsproxy-poc-keyvault.vault.azure.net. It does not support custom domain URLs like https://tlsproxy-poc-keyvault.cloudapp.net (or https://https://tlsproxy-poc-keyvault.contoso.com). This feature can be supported by adding Application Gateway as a frontend for KeyVault which maps custom domain URL to Application Gateway URL.

In this document, all the components are configured using Azure portal. A secret is retrieved using a SDK client from a KeyVault. Client sends HTTP request using custom domain URL and KeyVault supports only KeyVault URL. AppGw sits between them and routes the traffic coming from client to the KeyVault endpoint.

Pre-requisites

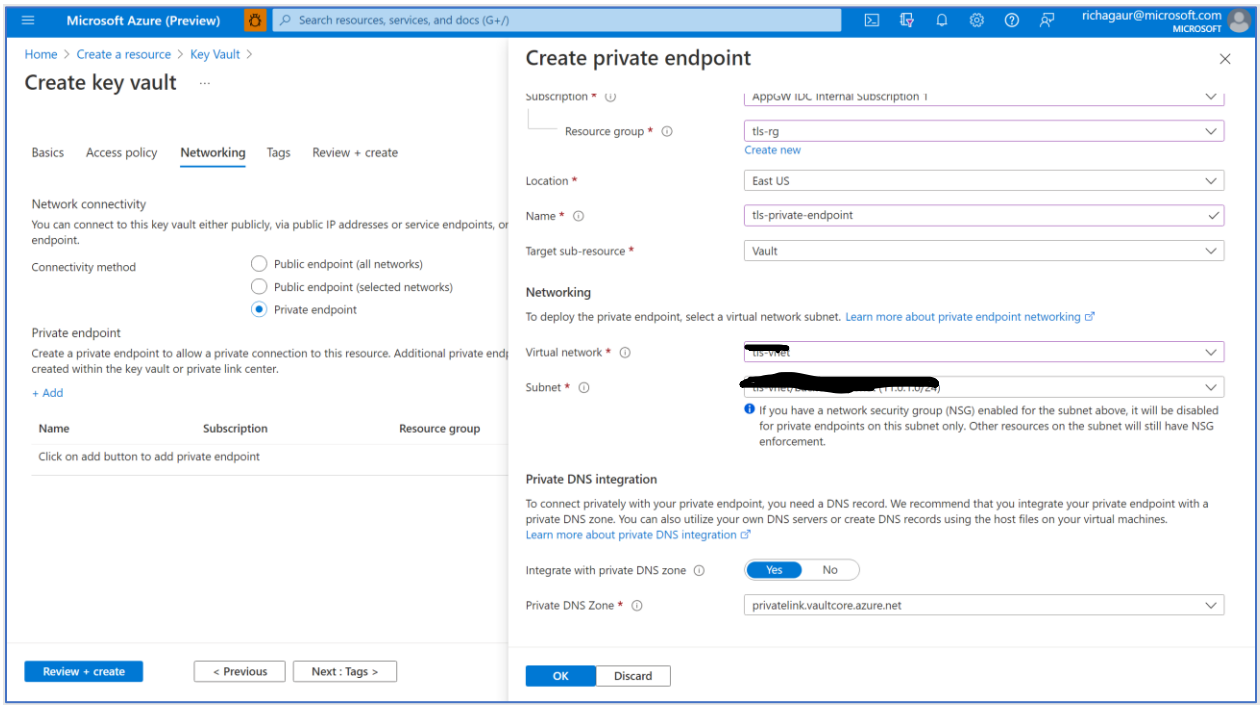
1. Azure Key Vault with a Secret
2. Application Gateway using L7 capabilities (HTTP)
3. Client SDK (Documentation)

Solution – Detailed Process

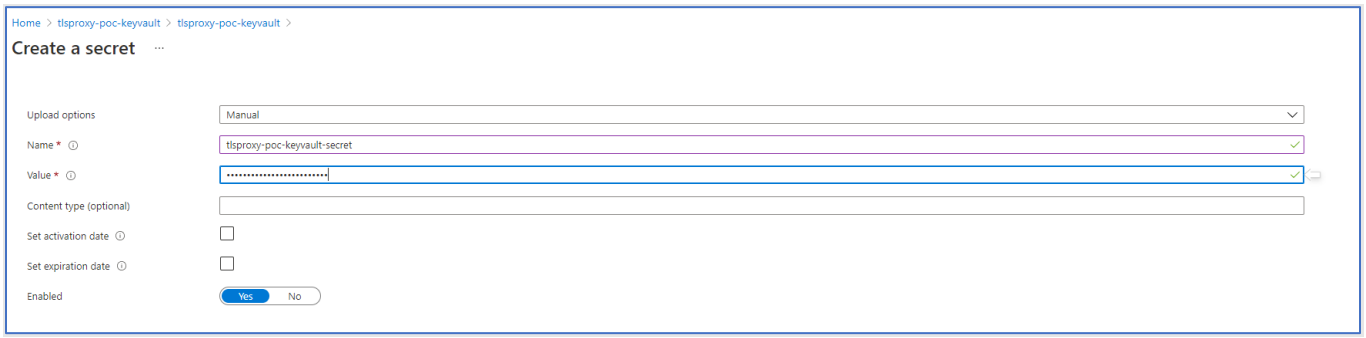
Configuration of Key Vault

1. Create a KeyVault with its private endpoint added to Application Gateway Virtual Network.

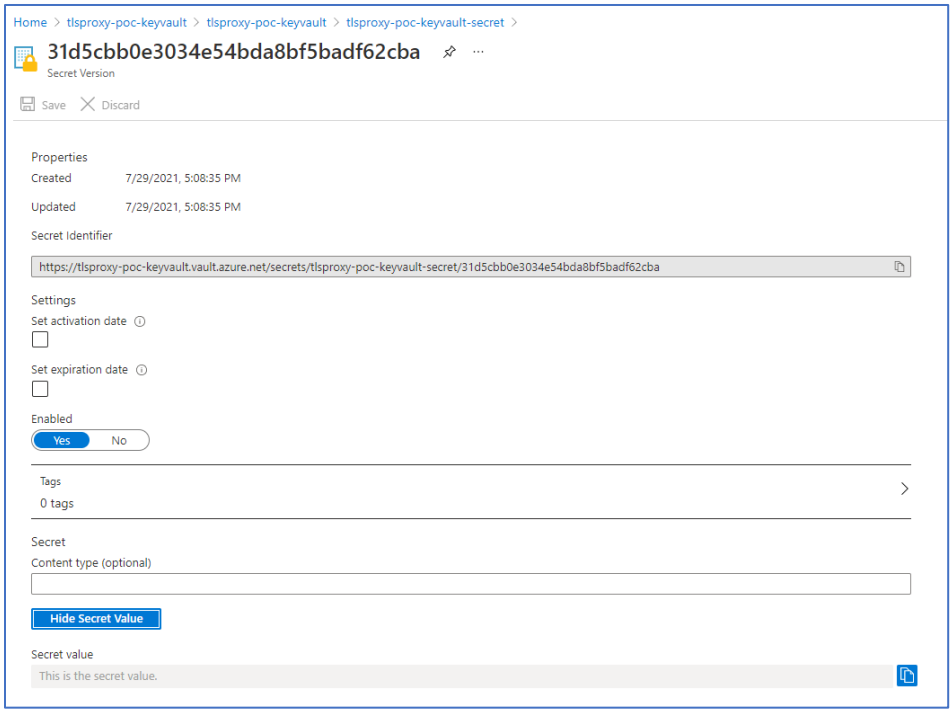
While selecting the network, use private endpoint option and create a new private endpoint. Choose the same resource group and vnet as application gateway and subnet should part of the same Vnet as the Application Gateway.



2. Add a Secret in Key Vault

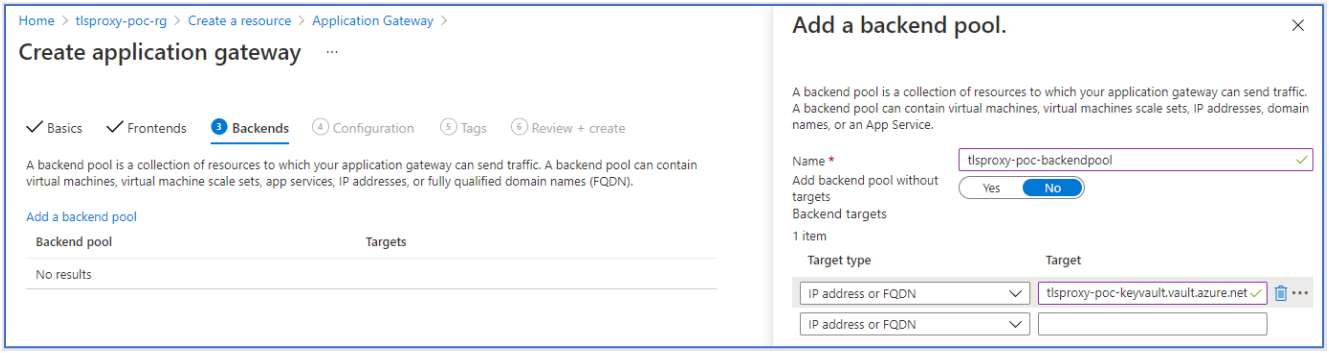


After secret addition, it will look like below.

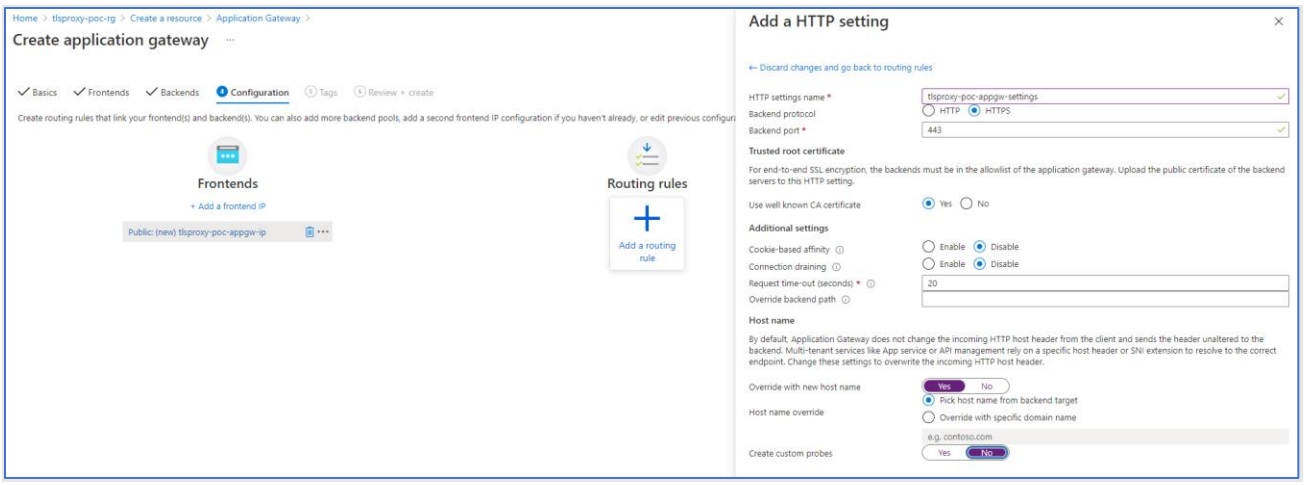


Configuration of Application Gateway

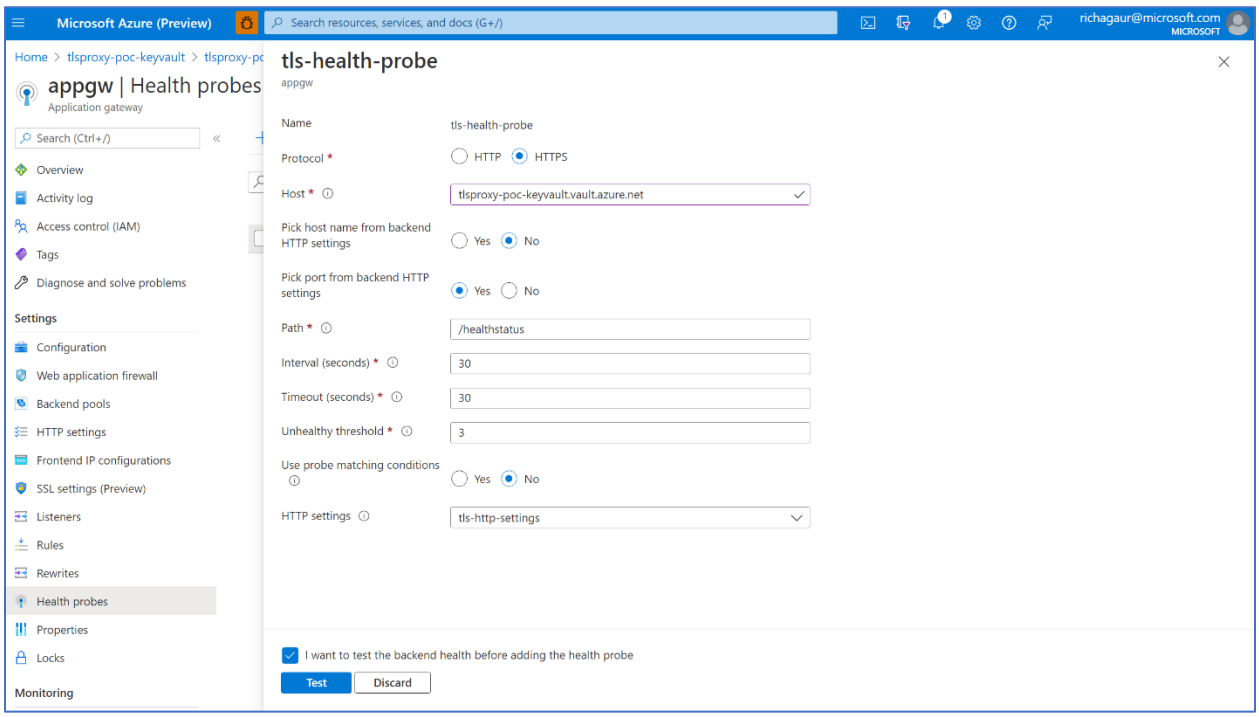
1. In backend pool, add target as default KeyVault URI.



2. For HTTP Setting, choose “well-known CA certificate” and override the hostname by picking it from the backend target. Custom probe can be added later.



3. Add a Custom Health Probe with HTTPS protocol and Path as "/healthstatus".



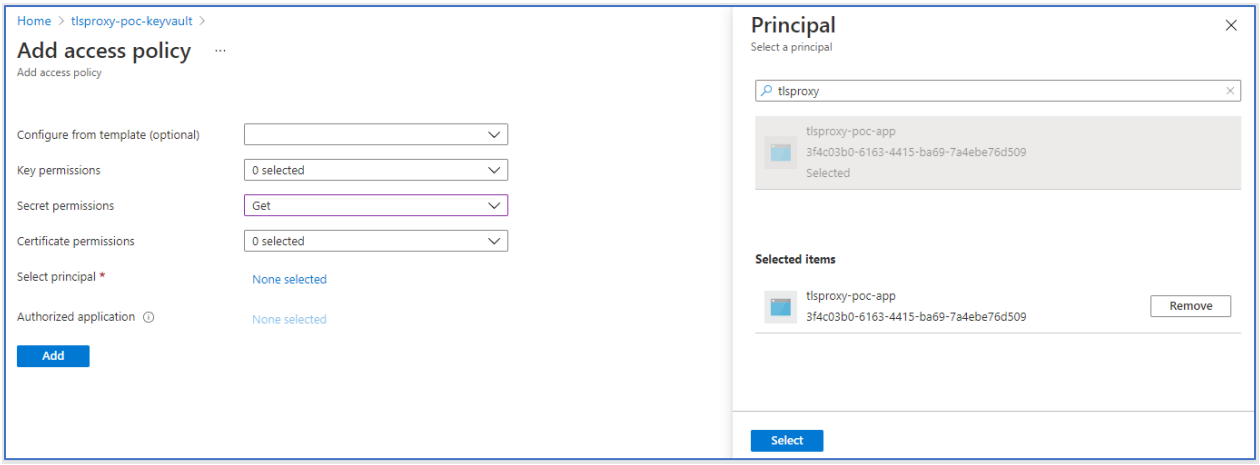
Configure Key Vault for authentication

1. Using PowerShell, create a service principal using the following command.

```
az ad sp create-for-rbac -n <your-application-name> --skip-assignment"
```

```
PS C:\WINDOWS\system32> az ad sp create-for-rbac -n tlsproxy-poc-app --skip-assignment
Changing "tlsproxy-poc-app" to a valid URI of "http://tlsproxy-poc-app", which is the required format used for service principal names
{
  "appId": "3f4c03b0-6163-4415-ba69-7a4ebe76d509",
  "displayName": "tlsproxy-poc-app",
  "name": "http://tlsproxy-poc-app",
  "password": "n7mtb8Q3oL1GBVWuKKVxeVZ-JTIJ0TAs-m",
  "tenant": "72f988bf-86f1-41af-91ab-2d7cd011db47"
}
PS C:\WINDOWS\system32>
```

2. In the Key Vault, add an access policy with Get Secret permissions and Service Principal created above.



Before we proceed, the custom domain's DNS should be setup to resolve to Application Gateway IP.

Verifying using an SDK Client

Create a client (C# Program) using Key Vault SDK. We will send a request through both default and custom URLs to check the outputs.

Default URL - GetSecret request is sent using the default Key Vault URL https://tlsproxy-poc-keyvault.vault.azure.net

```
Program.cs
KeyVaultClient                                  KeyVaultClient.Program
1   namespace KeyVaultClient
2   {
3       using System;
4       using Azure.Security.KeyVault.Secrets;
5       using Azure.Identity;
6
        0 references
7       class Program
8       {
9           private const string keyVaultUrl = "https://tlsproxy-poc-keyvault.vault.azure.net";
10          private const string customUrl = "https://appgwdevtest.appgwtest.cloudapp.net";
11          private const string tenantId = "72f988bf-86f1-41af-91ab-2d7cd011db47";
12          private const string clientId = "3f4c03b0-6163-4415-ba69-7a4ebe76d509";
13          private const string clientSecret = "n7mtb8Q3oL1GBVWuKKVxeVZ-JTIJ0TAs-m";
14          private const string secretName = "tlsproxy-poc-keyvault-secret";
15          private const string secretVersion = "31d5cbb0e3034e54bda8bf5badf62cba";
16
            0 references
17          static void Main(string[] args)
18          {
19              string url = keyVaultUrl;
20              SecretClient client = new SecretClient(vaultUri: new Uri(url),
21                  credential: new ClientSecretCredential(tenantId: tenantId, clientId: clientId, clientSecret: clientSecret));
22              KeyVaultSecret secret = null;
23
24              try
25              {
26                  secret = client.GetSecret(name: secretName, version: secretVersion);
27                  Console.WriteLine($"GetSecret request sent for secret {secretName} and version {secretVersion} with URL {url}");
28              }
29              catch (Exception ex)
30              {
31                  Console.WriteLine(ex);
32                  return;
33              }
34
35              Console.WriteLine($"Secret name: {secret.Name}");
36              Console.WriteLine($"Secret value: {secret.Value}");
37              Console.WriteLine($"Secret id: {secret.Id}");
38              Console.WriteLine($"Secret version: {secret.Properties.Version}");
39              Console.WriteLine($"Secret enabled: {secret.Properties.Enabled}");
40          }
41      }
42  }
43
146 %
Error List  Output

C:\WINDOWS\system32\cmd.exe
GetSecret request sent for secret tlsproxy-poc-keyvault-secret and version 31d5cbb0e3034e54bda8bf5badf62cba with URL https://tlsproxy-poc-keyvault.vault.azure.net
Secret name: tlsproxy-poc-keyvault-secret
Secret value: This is the secret value.
Secret id: https://tlsproxy-poc-keyvault.vault.azure.net/secrets/tlsproxy-poc-keyvault-secret/31d5cbb0e3034e54bda8bf5badf62cba
Secret version: 31d5cbb0e3034e54bda8bf5badf62cba
Secret enabled: True
Press any key to continue . . .
```

Output

```
GetSecret request sent for secret tlsproxy-poc-keyvault-secret and version 31d5cbb0e3034e54bda8bf5badf62cba with URL https://tlsproxy-poc-keyvault.vault.azure.net
Secret name: tlsproxy-poc-keyvault-secret
Secret value: This is the secret value.
Secret id: https://tlsproxy-poc-keyvault.vault.azure.net/secrets/tlsproxy-poc-keyvault-secret/31d5cbb0e3034e54bda8bf5badf62cba
Secret version: 31d5cbb0e3034e54bda8bf5badf62cba
Secret enabled: True
```

Custom URL - GetSecret request is sent using the Custom URL https://appgwdevtest.appgwtest.cloudapp.net

```
Program.cs
KeyVaultClient                                  KeyVaultClient.Program
1   namespace KeyVaultClient
2   {
3       using System;
4       using Azure.Security.KeyVault.Secrets;
5       using Azure.Identity;
6
        0 references
7       class Program
8       {
9           private const string keyVaultUrl = "https://tlsproxy-poc-keyvault.vault.azure.net";
10          private const string customUrl = "https://appgwdevtest.appgwtest.cloudapp.net";
11          private const string tenantId = "72f988bf-86f1-41af-91ab-2d7cd011db47";
12          private const string clientId = "3f4c03b0-6163-4415-ba69-7a4ebe76d509";
13          private const string clientSecret = "n7mtb8Q3oL1GBVWuKKVxeVZ-JTIJ0TAs-m";
14          private const string secretName = "tlsproxy-poc-keyvault-secret";
15          private const string secretVersion = "31d5cbb0e3034e54bda8bf5badf62cba";
16
            0 references
17          static void Main(string[] args)
18          {
19              string url = customUrl;
20              SecretClient client = new SecretClient(vaultUri: new Uri(url),
21                  credential: new ClientSecretCredential(tenantId: tenantId, clientId: clientId, clientSecret: clientSecret));
22              KeyVaultSecret secret = null;
23
24              try
25              {
26                  secret = client.GetSecret(name: secretName, version: secretVersion);
27                  Console.WriteLine($"GetSecret request sent for secret {secretName} and version {secretVersion} with URL {url}");
28              }
29              catch (Exception ex)
30              {
31                  Console.WriteLine(ex);
32                  return;
33              }
34
35              Console.WriteLine($"Secret name: {secret.Name}");
36              Console.WriteLine($"Secret value: {secret.Value}");
37              Console.WriteLine($"Secret id: {secret.Id}");
38              Console.WriteLine($"Secret version: {secret.Properties.Version}");
39              Console.WriteLine($"Secret enabled: {secret.Properties.Enabled}");
40          }
41      }
42  }
43
146 %
Error List  Output

C:\WINDOWS\system32\cmd.exe
GetSecret request sent for secret tlsproxy-poc-keyvault-secret and version 31d5cbb0e3034e54bda8bf5badf62cba with URL https://appgwdevtest.appgwtest.cloudapp.net
Secret name: tlsproxy-poc-keyvault-secret
Secret value: This is the secret value.
Secret id: https://tlsproxy-poc-keyvault.vault.azure.net/secrets/tlsproxy-poc-keyvault-secret/31d5cbb0e3034e54bda8bf5badf62cba
Secret version: 31d5cbb0e3034e54bda8bf5badf62cba
Secret enabled: True
Press any key to continue . . .
```

Output

```
GetSecret request sent for secret tlsproxy-poc-keyvault-secret and version 31d5cbb0e3034e54bda8bf5badf62cba with URL https://appgwdevtest.appgwtest.cloudapp.net
Secret name: tlsproxy-poc-keyvault-secret
Secret value: This is the secret value.
Secret id: https://tlsproxy-poc-keyvault.vault.azure.net/secrets/tlsproxy-poc-keyvault-secret/31d5cbb0e3034e54bda8bf5badf62cba
Secret version: 31d5cbb0e3034e54bda8bf5badf62cba
Secret enabled: True
```

Client Code

```
namespace KeyVaultClient
{
    using System;
    using Azure.Security.KeyVault.Secrets;
    using Azure.Identity;

    class Program
    {
        private const string keyVaultUrl = "https://tlsproxy-poc-keyvault.vault.azure.net";
        private const string customUrl = "https://appgwdevtest.appgwtest.cloudapp.net";
        private const string tenantId = "72f988bf-86f1-41af-91ab-2d7cd011db47";
        private const string clientId = "3f4c03b0-6163-4415-ba69-7a4ebe76d509";
        private const string clientSecret = "n7mtb8Q3oL1GBVWuKKVxeVZ-JTIJ0TAs-m";
        private const string secretName = "tlsproxy-poc-keyvault-secret";
        private const string secretVersion = "31d5cbb0e3034e54bda8bf5badf62cba";

        static void Main(string[] args)
        {
            string url = customUrl;
            SecretClient client = new SecretClient(vaultUri: new Uri(keyVaultUrl), credential: new
ClientSecretCredential(tenantId: tenantId, clientId: clientId, clientSecret: clientSecret));
            KeyVaultSecret secret = null;

            try
            {
                secret = client.GetSecret(name: secretName, version: secretVersion);
                Console.WriteLine($"GetSecret request sent for secret {secretName} and version
{secretVersion} with URL {url}");
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex);
                return;
            }

            Console.WriteLine($"Secret name: {secret.Name}");
            Console.WriteLine($"Secret value: {secret.Value}");
            Console.WriteLine($"Secret id: {secret.Id}");
            Console.WriteLine($"Secret version: {secret.Properties.Version}");
            Console.WriteLine($"Secret enabled: {secret.Properties.Enabled}");
        }
    }
}
```

Callouts/Caveats

1. Some API's return the ID/URI of resource having the actual/default hostname of keyvault even while calling the api's from custom domain. The response in these api's would need to be modified to replace the actual hostname with custom domain. Example: In API Get Keys there is nextLink field which points to the next url for getting the paginated response for keys. The hostname in next url would be replaced to custom domain, and everything should work fine as is.

Request

```
GET https://myvault.vault.azure.net//keys?maxresults=1&api-version=7.2
```

Response


```
{
  "value": [
    {
      "kid": "https://myvault.vault.azure.net/keys/sdktestkey",
      "attributes": {
        "enabled": true,
        "created": 1493937656,
        "updated": 1493937656,
        "recoveryLevel": "Recoverable+Purgeable"
      }
    }
  ],
  "nextLink":"https://myvault.vault.azure.net:443/keys?api-
version=7.2&$skiptoken=eyJOZXh0TWFya2VyIjoiMiE5NiFNREF3TURJM0lXdGxlUzlWVUVSQlZF
VkxSVmxCVkZSU1NVSlZWRVZUVkVWVFZDRXdNREF3TWpnaE1q
QXhOeTB3TlMwd05GUXdNVG94TVRveE5pNDNNekE0TnpReVdpRS0iLCJUYXJnZXRMb2NhdGlv
biI6MH0&maxresults=1"
}
```


2. The health probe must contain the path **/healthstatus** else the backend health check would respond with 404.

3. The Pick Host Name from Backend should be set to true in HTTP Setting to work correctly.

4. While configuring Private Link from clients other than portal, the subnet in which the endpoint would reside must have “privateLinkServiceNetworkPolicies” property disabled. More on this here.

5. The DNS of the custom domain should point to the Application Gateway public IP.

6. Due to current limitation with Application Gateway, any PaaS service’s default FQDN should be added to the backend pool **after** configuring the private link, otherwise the default FQDN continues to point to its public IP. [ETA for this fix: Early 2022. As a workaround, you can even perform any PUT operation on gateway which will refresh its DNS.]

7. This document assumes that the Listener is created with HTTPS protocol and an appropriate certificate.

ETA

Private preview – 15th February 2022

----------End of File----------