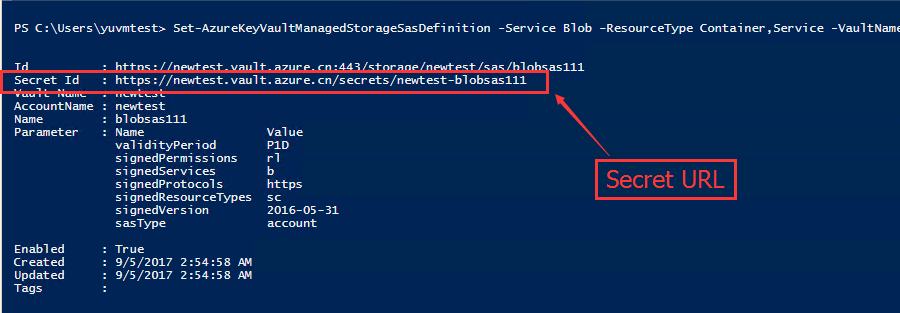
**题目：**Azure Key Vault 管理存储帐户密钥

在 Azure Key Vault 存储帐户密钥推出之前，开发人员必须管理其自己的 Azure 存储帐户 (ASA) 密钥，并手动或通过外部自动化功能轮换这些密钥。 现在，Key Vault 存储帐户密钥已实现为 Key Vault 机密，可用于通过 Azure 存储帐户进行身份验证。本文主要介绍如何使用Azure Powershell创建基于Key Vault的存储账户SAS信息，然后结合C#基于AD Application认证的方式使用Key Vault获取存储对应的SAS，进而创建StorageClient。

* **Powershell code**

|  |
| --- |
| # 参数设置  $subscriptionId = 'e0fbea86-6cf2-4b2d-81e2-9c59f4f96bcb'  $keyVaultName = 'newtest'  $resourceGroupName = 'yuvmtest'  $keyVaultObjectId = '3be6a1e9-8a8a-4613-8e4e-b974593d011e'  $location = 'china north'  $adApplicationId = '16a94c84-c6ba-4f19-aa48-3353f8ffe18e' # 通过后面：获取Key Vault的ObjectId 脚本获取  $storageAccountName = 'yunewstoragetest'  # Scope参数获取：Protal -> 对应的存储账户 -> 设置 -> 属性 -> 资源ID  $scope = '/subscriptions/e0fbea86-6cf2-4b2d-81e2-9c59f4f96bcb/resourceGroups/yuvmtest/providers/Microsoft.Storage/storageAccounts/yunewstoragetest'  # 使用用户名和密码登录到订阅  $cred = New-Object System.Management.Automation.PSCredential("cietest03@microsoftinternal.partner.onmschina.cn",(ConvertTo-SecureString "DEV@!321" -AsPlainText -Force))  Add-AzureRmAccount -EnvironmentName AzureChinaCloud -Credential $cred  Get-AzureRmSubscription | select SubscriptionName,SubscriptionID | sort SubscriptionName,SubscriptionID  Set-AzureRmContext -SubscriptionId $subscriptionId  Get-AzureRmStorageAccount -Name yutest  # 获取Key Vault的ObjectId：  Get-AzureRmADServicePrincipal -SearchString "Azure Key Vault"  # 创建密钥保管库  New-AzureRmKeyVault -VaultName $keyVaultName -ResourceGroupName $resourceGroupName -Location $location  Get-AzureRmKeyVault -VaultName $keyVaultName    # 根据获取的ObjectId将存储密钥操作员角色分配给 Azure Key Vault 标识  New-AzureRmRoleAssignment -ObjectId $keyVaultObjectId -RoleDefinitionName Owner -Scope $scope  # 为key-vault分配操作storage的权限范围  Set-AzureRmKeyVaultAccessPolicy -VaultName $keyVaultName -ObjectId $keyVaultObjectId -PermissionsToStorage all  # 授权AD应用程序使用密钥或机密，关于AD应用程序的创建请参考链接：http://arui.me/index.php/archives/114/  Set-AzureRmKeyVaultAccessPolicy -VaultName $keyVaultName -ServicePrincipalName $adApplicationId -PermissionsToSecrets all -PermissionsToStorage all -PermissionsToCertificates all  # 添加storage Account  $regenerationPeriod = [System.Timespan]::FromDays(90)  Add-AzureKeyVaultManagedStorageAccount -AccountName $keyVaultName -VaultName $keyVaultName -AccountResourceId $scope -ActiveKeyName 'key1' -RegenerationPeriod $regenerationPeriod  # 获取 Secret URL 对应secret id  Set-AzureKeyVaultManagedStorageSasDefinition -Service Blob -ResourceType Container,Service -VaultName $keyVaultName -AccountName $keyVaultName -Name blobsas111 -Protocol HttpsOnly -ValidityPeriod ([System.Timespan]::FromDays(1)) -Permission Read,List,Write,Delete |

获取的Secret URL：



* **C#Code调用**

|  |
| --- |
| using Microsoft.Azure.KeyVault;  using Microsoft.IdentityModel.Clients.ActiveDirectory;  using Microsoft.WindowsAzure.Storage;  using Microsoft.WindowsAzure.Storage.Auth;  using Microsoft.WindowsAzure.Storage.Blob;  using System;  using System.Net.Http;  using System.Threading;  using System.Threading.Tasks;  namespace storagekeyvault  {  class Program  {  static KeyVaultClient kv;  static void Main(string[] args)  {  AsyncMethod();  Console.WriteLine("success!");  Console.ReadKey(true);  }  static async void AsyncMethod()  {  kv = new KeyVaultClient(new KeyVaultClient.AuthenticationCallback((authority, resource, scope) => GetToken(authority, resource, scope)), new InjectHostHeaderHttpMessageHandler());  var secret1 = await kv.GetSecretAsync("https://newtest.vault.azure.cn/secrets/newtest-blobsas111");  var accountSasCredential = new StorageCredentials(secret1.Value);  // Use credentials and the Blob storage endpoint to create a new Blob service client.  var accountWithSas = new CloudStorageAccount(accountSasCredential, new Uri("https://yutest.blob.core.chinacloudapi.cn/"), null, null, null);  var blobClientWithSas = accountWithSas.CreateCloudBlobClient();  // Retrieve a reference to a container.  CloudBlobContainer container1 = blobClientWithSas.GetContainerReference("aaa111");  // Create the container if it doesn't already exist.  container1.CreateIfNotExists();  Console.WriteLine(container1.Properties);  Console.WriteLine("----------------------------------------------------------------");  Console.ReadKey(true);  }  /// <summary>  /// 获取认证token  /// </summary>  /// <param name="authority"></param>  /// <param name="resource"></param>  /// <param name="scope"></param>  /// <returns></returns>  private static async Task<string> GetToken(string authority, string resource, string scope)  {  var authContext = new AuthenticationContext(authority);  //对应AD Application的信息  ClientCredential clientCred = new ClientCredential("16a94c84-c6ba-4f19-aa48-3353f8ffe18e", "123456");  var result = await authContext.AcquireTokenAsync(resource, clientCred);  if (result == null)  throw new InvalidOperationException("Failed to obtain the JWT token");  return result.AccessToken;  }  }  public class InjectHostHeaderHttpMessageHandler : DelegatingHandler  {  protected override Task<HttpResponseMessage> SendAsync(HttpRequestMessage request,  CancellationToken cancellationToken)  {  var requestUri = request.RequestUri;  var authority = string.Empty;  var targetUri = requestUri;  string networkUrl = "";  if (!string.IsNullOrEmpty(networkUrl))  {  authority = targetUri.Authority;  targetUri = new Uri(new Uri(networkUrl), targetUri.PathAndQuery);  request.Headers.Add("Host", authority);  request.RequestUri = targetUri;  }  return base.SendAsync(request, cancellationToken).ContinueWith<HttpResponseMessage>(response =>  {  return response.Result;  });  }  }  } |

* **更多信息参考链接：**

Azure Key Vault 存储帐户密钥：<https://docs.microsoft.com/zh-cn/azure/key-vault/key-vault-ovw-storage-keys>

Azure 密钥保管库入门：<https://docs.azure.cn/zh-cn/key-vault/key-vault-get-started#authorize>

Azure​RM.​Key​Vault：<https://docs.microsoft.com/zh-cn/powershell/module/azurerm.keyvault/?view=azurermps-4.3.1>