```
$KVRGname = 'KeyVaultRG';

 $VMRGName = 'SecurityDemoRG';

 $vmName = 'TestMachine';

 $KeyVaultName = 'RudraDiskEncryptionVault';


 $KeyVault = Get-AzKeyVault -VaultName $KeyVaultName -ResourceGroupName
$KVRGname;

 $diskEncryptionKeyVaultUrl = $KeyVault.VaultUri;

 $KeyVaultResourceId = $KeyVault.ResourceId;

 Set-AzVMDiskEncryptionExtension -ResourceGroupName $VMRGname -VMName $vmName
-DiskEncryptionKeyVaultUrl $diskEncryptionKeyVaultUrl -
DiskEncryptionKeyVaultId $KeyVaultResourceId;


Get-AzVmDiskEncryptionStatus -ResourceGroupName 'SecurityDemoRG' -VMName
'TestMachine'
```

```
ng System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using Microsoft.WindowsAzure.Storage;
using Microsoft.WindowsAzure.Storage.Blob;


namespace ConsoleApp3
{
    class Program
    {
        static void Main(string[] args)
        {
            //Return a reference to the container using the SAS URI.
            CloudBlobContainer container = new CloudBlobContainer(new Uri("<Container
SAS key here>"));

            try
            {
                foreach (ICloudBlob blob in container.ListBlobs())
                {
                    Console.WriteLine(blob.Name);
                }

                Console.ReadKey();
            }
            catch (StorageException e)
            {
                Console.WriteLine("List operation failed ");

            }



        }
    }
}
```

- Login into Azure
  - Connect-AzAccount
- Get existing role definition
  - Get-AzRoleDefinition -Name "Storage Account Contributor" | ConvertTo-Json | Out-File <Filepath>
- To get Azure subscription ID
  - Get-AzSubscription
- To create new role
  - New-AzRoleDefinition -InputFile <FilePath>

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Client] "DisabledByDefault"=dword:00000000
"Enabled"=dword:00000001

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\SecurityProviders\SCHANNEL\Protocols\TLS 1.2\Server] "DisabledByDefault"=dword:00000000
"Enabled"=dword:00000001

[HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\.NETFramework\v4.0.30319]
"SchUseStrongCrypto"=dword:00000001

```
$response = Invoke-WebRequest -Uri
'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-
01&resource=https://storage.azure.com/' -Method GET -Headers @{Metadata="true"}
```

```
$content = $response.Content | ConvertFrom-Json

$ArmToken = $content.access_token

$head = New-Object "System.Collections.Generic.Dictionary[[String],[String]]"
$head = @{}
$head.Add("Authorization", "Bearer $ArmToken")
$head.Add("x-ms-version","2017-11-09")

$byte = (Invoke-WebRequest -Uri
https://rudrateststorageaccount.blob.core.windows.net/test/CustomStorageAccountContributor.jso
n -Method GET -Headers $head).content

$string = [System.Text.Encoding]::UTF8.GetString($byte)

$string
```

```
        {
          "mode": "all",
          "policyRule": {
            "if": {
              "field": "[concat('tags[', parameters('tagName'), ']')]",
              "exists": "false"
            },
            "then": {
              "effect": "append",
              "details": [
                {
                  "field": "[concat('tags[', parameters('tagName'), ']')]",
                  "value": "[parameters('tagValue')]"
                }
              ]
            }
          },
          "parameters": {
            "tagName": {
              "type": "String",
              "metadata": {
                "description": "Name of the tag, such as costCenter"
              }
            },
            "tagValue": {
              "type": "String",
              "metadata": {
                "description": "Value of the tag, such as headquarter"
              }
            }
          }
        }
```

```
{
  "mode": "indexed",
  "policyRule": {
    "if": {
      "allOf": [
        {
          "field": "location",
          "notIn": "[parameters('listOfAllowedLocations')]"
        },
        {
          "field": "location",
          "notEquals": "global"
        },
        {
          "field": "type",
          "notEquals": "Microsoft.AzureActiveDirectory/b2cDirectories"
        }
      ]
```

```
    },
    "then": {
      "effect": "deny"
    }
  },
  "parameters": {
    "listOfAllowedLocations": {
      "type": "Array",
      "metadata": {
        "displayName": "Allowed locations",
        "description": "The list of locations that can be specified when deploying resources.",
        "strongType": "location"
      }
    }
  }
}
```

- Microsoft.IdentityModel.Clients.ActiveDirectory
- WindowsAzure.Storage

```
using Microsoft.IdentityModel.Clients.ActiveDirectory;
using Microsoft.WindowsAzure.Storage.Auth;
using System.Globalization;
using Microsoft.WindowsAzure.Storage.Blob;
using System.IO;

 static void Main(string[] args)
     {
        Task<AuthenticationResult> t = GetUserOAuthToken();
        t.Wait();
        string accessToken = t.Result.AccessToken;
        Console.WriteLine( "ACCESS TOKEN \n\n" + accessToken);
        Console.WriteLine("\n\n Please any key to display content of the blob");
        Console.ReadKey();

        // Use the access token to create the storage credentials.
        TokenCredential tokenCredential = new TokenCredential(accessToken);
        StorageCredentials storageCredentials = new StorageCredentials(tokenCredential);

        // Create a block blob using those credentials
        CloudBlockBlob blob = new CloudBlockBlob(new
Uri("https://rudrateststorageaccount.blob.core.windows.net/testcontainer/SalesOrder.json"),
storageCredentials);
        using (var stream = blob.OpenRead())
        {
           using (StreamReader reader = new StreamReader(stream))
           {
              while (!reader.EndOfStream)
              {
                 Console.WriteLine(reader.ReadLine());

              }
           }
        }
        Console.WriteLine("\n\n Please any key to terminate the program");
        Console.ReadKey();
     }
     static async Task<AuthenticationResult> GetUserOAuthToken()
     {
        const string ResourceId = "https://storage.azure.com/";
        const string AuthInstance = "https://login.microsoftonline.com/{0}/";
        const string TenantId = "<TenantID>"; // Tenant or directory ID

        // Construct the authority string from the Azure AD OAuth endpoint and the tenant ID.
        string authority = string.Format(CultureInfo.InvariantCulture, AuthInstance, TenantId);
        AuthenticationContext authContext = new AuthenticationContext(authority);
        ClientCredential cc = new ClientCredential("<CLIENT ID>", "<CLIENT SECRET>");

        // Acquire an access token from Azure AD.
```

```
    AuthenticationResult result = await authContext.AcquireTokenAsync(ResourceId, cc);

    return result;
}
```

```
$cert = New-SelfSignedCertificate -Type Custom -KeySpec Signature -Subject "CN=trainings.rudra9.com" -KeyExportPolicy Exportable -HashAlgorithm sha256 -KeyLength 2048 -CertStoreLocation "Cert:\CurrentUser\My" -KeyUsageProperty Sign -KeyUsage CertSign -TextExtension @("2.5.29.37={text}1.3.6.1.5.5.7.3.1")
```