* **LAB :** Implement Azure AppObject Configuration
* Go to entra id , >> app registrations >> create an application object blobapp
* Once appregistration is done , go to certifecates and secrets >> create a secret for your appobject

using Azure.Identity;

using Azure.Storage.Blobs;

string containerName="testcont";

string fileName="image1.jpg";

string tenantId="30cc3176-5a12-41ed-b15b-2e34c82f6e8c";

string clientId="1a4f6d58-9f84-4d16-a4eb-d762b09422b4";

string secret="u4b8Q~8j2z3l.lIvSB54tutk2cGuAZQp~Xgv9bJU";

string path=@"C:\Users\HP\Downloads\image1.jpg";

string storageAccountName="testsaaaaaaaaa";

string blobUri=$"https://{storageAccountName}.blob.core.windows.net/{containerName}/{fileName}";

ClientSecretCredential clientSecretCredential=new ClientSecretCredential(tenantId,clientId,secret);

BlobClient blobClient= new BlobClient(new Uri(blobUri),clientSecretCredential);

await blobClient.DownloadToAsync(path);

* Go to azure portal >> ur storage account >> access (iam) >> add role assignment >> blob reader role to our applicationObject blobapp
* Go to the visual studio code >>
* Create a new project “blobapp” and paste above code to test the access .
* In the terminal , :

dotnet add package Azure.Identity

like below :

C:\Users\HP\Desktop\blobapp\blobapp> dotnet add package Azure.Identity

* **Run the above code , on the top right ( run code with associated project )**

**Lab Guide: Using Azure Key Vault to Secure Secrets and Deploy VM**

**Lab 1: Create a Key Vault and Store Secrets**

**Step 1: Create a Key Vault**

1. Navigate to the **Azure Portal**.
2. Search for **Key Vaults** in the search bar.
3. Click **Create**.
   * **Resource Group**: Select or create a resource group (e.g., raman-test-rg).
   * **Key Vault Name**: Enter a unique name (e.g., myKeyVault).
   * **Region**: Select a region close to your location.
   * Click **Review + Create**, then **Create**.

**Step 3: Create a Secret**

1. Navigate to the **Secrets** tab in your Key Vault.
2. Click **+ Generate/Import**.
   * Name: Enter a name for the secret (e.g., vmAdminPassword).
   * Value: Enter a value (e.g., StrongPassword@123).
   * Click **Create**.

**Lab 2: Deploy a VM Using Key Vault Secrets**

**Step 1: Create a Secret for the VM Admin Password**

1. Go to the **Secrets** tab of your Key Vault.
2. Click **+ Generate/Import**.
   * Name: Enter vmAdminPassword.
   * Value: Set the desired VM admin password (e.g., Vm@Password123).
   * Click **Create**.

**Step 2: Assign RBAC Permissions for Key Vault Access**

1. Go to your Key Vault.
2. Navigate to **Access Control (IAM)**.
3. Click **Add role assignment**.
   * Role: Select **Key Vault Secrets User**.
   * Assign access to: Select **User, group, or service principal**.
   * Choose the account or service principal that will deploy the VM.
   * Click **Save**.

**Step 3: Download the VM Template**

1. Navigate to **Virtual Machines** in the Azure Portal.
2. Start creating a new VM.
3. Configure VM settings (e.g., OS, size, etc.).
4. At the **Review + Create** step, download the **template** and **parameters** file.

**Step 4: Modify the Parameters File**

1. Open the downloaded parameters.json file.
2. Locate the adminPassword parameter and replace it with the following:

json

Copy code

"adminPassword": {

"reference": {

"keyVault": {

"id": "/subscriptions/<SubscriptionID>/resourceGroups/<ResourceGroupName>/providers/Microsoft.KeyVault/vaults/<KeyVaultName>"

},

"secretName": "vmAdminPassword"

}

}

* **Mention and replace the keyvalut id ( u will find the id by az keyvault list )**
* **Step 5: Deploy the VM Using CLI**

1. Open **Cloud Shell** in the Azure Portal.
2. Upload the modified parameters.json and template.json files.
3. Deploy the VM using the following command:

bash

Copy code

az deployment group create --resource-group raman-test-rg --template-file template.json --parameters parameters.json

NOTE : MAKE SURE TO CHECK IN KEYVAULTS >> TESTKEYVAULTTT >> configurations >> access configurations >> “ Azure Resource Manager for template deployment “ is enabled …

**Step 6: Verify the VM Deployment**

1. Navigate to **Virtual Machines** in the Azure Portal.
2. Check if the new VM is successfully created , login to it check if ur able to login with the password that secret accesed i.e : StrongPassword@123

**Lab Guide: Test Azure Managed Identity**

**Step 1: Create Azure Key Vault**

1. **Create Key Vault**:
   * In the Azure Portal, navigate to **Key Vaults** > **Create Key Vault**.
   * Provide:
     + **Name**: MyTestKeyVault
     + **Region**: Same as the VM
     + **Pricing Tier**: Standard
   * Click **Create**.

* Become keyvault admintrtaor to do the below steps

1. **Add a Secret**:
   * Go to the Key Vault > **Secrets** > **Generate/Import**.
   * Add:
     + **Name**: TestSecret
     + **Value**: MySecretValue

* **Create an ubuntu vm so that we can test our application code on vm if its able to access vault from vm itself**

**Step 3: Enable System-Assigned Managed Identity on the VM**

1. **Go to VM Security**:
   * In the Azure Portal, navigate to your VM.
   * Go to **Identity** in the left menu.
2. **Enable System-Assigned Managed Identity**:
   * Under the **System-assigned** tab, toggle the status to **On**.
   * Click **Save**.

**Step 4: make sure access is granted to virtual machines in ur azure keyvault under settings >> access configuration**

**Verify Access**:

* + Go to the Key Vault > **Access Control (IAM)**.
  + Ensure the VM's Managed Identity is listed with appropriate permissions. (key vault contributor or vault secrets reader maybe)

**Step 5: Test Access Using Python**

1. **Log in to the VM**:
   * Connect to the VM via SSH.

<https://learn.microsoft.com/en-us/python/api/overview/azure/identity-readme?view=azure-python>

1. **Install Python and Required Packages**:
   * If not already installed, install Python and the required libraries:

bash

Copy code

sudo apt update

sudo apt install python3 python3-pip -y

pip3 install azure-identity azure-keyvault-secrets

1. **Run the Python Script**:
   * Create a Python file (test\_managed\_identity.py) on the VM with the following content:

python

Copy code

from azure.identity import DefaultAzureCredential

from azure.keyvault.secrets import SecretClient

# Azure Key Vault URL

KEY\_VAULT\_URL = "https://<your-key-vault-name>.vault.azure.net/"

# Default Credential (Uses Managed Identity)

credential = DefaultAzureCredential()

# Create a SecretClient

secret\_client = SecretClient(vault\_url=KEY\_VAULT\_URL, credential=credential)

# Retrieve the secret

secret\_name = "TestSecret"

retrieved\_secret = secret\_client.get\_secret(secret\_name)

# Print the secret value

print(f"The value of the secret '{secret\_name}' is: {retrieved\_secret.value}")

1. **Replace Placeholder**:
   * Replace <your-key-vault-name> with the name of your Key Vault.
2. **Run the Script**:
   * Execute the script:

bash

Copy code

python3 test\_managed\_identity.py

**Step 6: Validate the Output**

* If everything is configured correctly, the script should output:

csharp

Copy code

The value of the secret 'TestSecret' is: MySecretValue

**Step 7: Troubleshooting**

1. **Managed Identity Not Detected**:
   * Ensure that the Managed Identity is enabled on the VM.
   * Verify that the VM has the Key Vault Secrets User role.
2. **Network Restrictions**:
   * Ensure that the Key Vault’s networking settings allow access from the VM’s region.

**Step 8: Cleanup**

* **Remove Resources**:
  + Delete the Azure VM and Key Vault if they are no longer needed to avoid unnecessary costs.

This guide provides a clear path to testing Azure Managed Identity in a lab environment. Let me know if you need additional clarification or enhancements!

**LAB : Python Script to Access Blobs Using Managed Identity**

root@testvm:~# cat blob.py

from azure.identity import DefaultAzureCredential

from azure.storage.blob import BlobServiceClient

# Azure Blob Storage details

BLOB\_STORAGE\_URL = "https://testsaaaaaaaaa.blob.core.windows.net/" # Replace with your storage account name

CONTAINER\_NAME = "testcont" # Replace with your container name

# Default Credential (Uses Managed Identity)

credential = DefaultAzureCredential()

# Create a BlobServiceClient

blob\_service\_client = BlobServiceClient(account\_url=BLOB\_STORAGE\_URL, credential=credential)

# List blobs in a container

print("\nBlobs in the container:")

container\_client = blob\_service\_client.get\_container\_client(CONTAINER\_NAME)

for blob in container\_client.list\_blobs():

print(f" - {blob.name}")

1. **Enable Managed Identity on the VM**:
   * Ensure **System-Assigned Managed Identity** is enabled on the VM.
2. **Grant Permissions**:
   * Assign the **Storage Blob Data Reader** role to the Managed Identity for the Storage Account. This allows the VM's Managed Identity to list and download blobs.
3. **Prepare Resources**:
   * Create a **Storage Account**.
   * Add a **container** named testcont.
4. **Install Required Libraries**: Install the Azure SDK libraries:

bash

Copy code

pip install azure-identity azure-storage-blob

1. **Replace Placeholder Values**:
   * Replace <your-storage-account-name> with your actual Storage Account name.
2. **Run the Script**:

bash

Copy code

python3 tets\_blob.py