Uso de Azure Key Vault con una máquina virtual en Python

Azure Key Vault le ayuda a proteger las claves, los secretos y los certificados, como las claves de API y las cadenas de conexión de base de datos.

En este tutorial, configurará una aplicación de Python para leer información de Azure Key Vault mediante identidades administradas para recursos de Azure. Aprenderá a:

- Crear una bóveda de claves
- Almacenar un secreto en Key Vault
- Crear una máquina virtual Linux de Azure
- Habilitar una identidad administrada para la máquina virtual
- Conceder los permisos necesarios para que la aplicación de consola lea datos de Key Vault
- Recuperar un secreto del almacén de claves

Creación de un grupo de recursos y un almacén de claves

Cada almacén de claves debe tener un nombre único. Reemplace <unique-keyvault-name> por el nombre del almacén de claves en los ejemplos siguientes.

```
az group create --name "myResourceGroup" -l "EastUS"

az keyvault create --name "<your-unique-keyvault-name>" -g "myResourceGroup" --enable-rbac-authorization
```

Llena tu bóveda de claves con un secreto

Vamos a crear un secreto llamado **mySecret** cuyo valor sea **¡Correcto!** . Un secreto puede ser una contraseña, una cadena de conexión SQL o cualquier otra información que necesite mantener segura y disponible para la aplicación.

Para agregar un secreto al almacén de claves recién creado, use el comando siguiente:

```
az keyvault secret set --vault-name "<your-unique-keyvault-name>" --name "mySecret" --value "Success!"
```

Creación de una máquina virtual

Para crear una máquina virtual Linux mediante la CLI de Azure, use el comando az vm create. En el ejemplo siguiente se agrega una cuenta de usuario llamada *azureuser*. El parámetro --generate-ssh-keys se usa para generar automáticamente una clave SSH y colocarla en la ubicación de la clave predeterminada (*~/.ssh*).

```
az vm create \
  --resource-group myResourceGroup \
```

```
  --name myVM \
  --image Ubuntu2204 \
  --admin-username azureuser \
  --generate-ssh-keys
```

Anote el valor de publicIpAddress en la salida.

Asignación de una identidad a la máquina virtual

Cree una identidad asignada por el sistema para la máquina virtual mediante el comando az vm identity assign de la CLI de Azure:

```
az vm identity assign --name "myVM" --resource-group "myResourceGroup"
```

Tenga en cuenta la identidad asignada por el sistema que se muestra en el código siguiente. La salida del comando anterior sería:

```
{
  "systemAssignedIdentity": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "userAssignedIdentities": {}
}
```

Asignación de permisos a la identidad de máquina virtual

Para obtener permisos para el almacén de claves mediante Control de acceso basado en roles (RBAC), asigne un rol a su "Nombre principal de usuario" (UPN) mediante el comando de la CLI de Azure az role assignment create.

```
az role assignment create --role "Key Vault Secrets User" --assignee "<upn>" --scope "/subscriptions/<subscription-id>/resourceGroups/<resource-group-name>/providers/Microsoft.KeyVault/vaults/<your-unique-keyvault-name>"
```

Reemplaza <upn>, <subscription-id>, <resource-group-name> y <your-unique-keyvault-name> con tus valores reales. El UPN normalmente tendrá el formato de una dirección de correo electrónico (por ejemplo, username@domain.com).

Inicio de sesión en la máquina virtual

Para iniciar sesión en una máquina virtual Linux, puede usar el comando ssh con el valor de <publicIpAddress> que se proporciona en el paso Creación de una máquina virtual:

```
ssh azureuser@<PublicIpAddress>
```

Instalación de bibliotecas de Python en la máquina virtual

En la máquina virtual, instale las dos bibliotecas de Python que vamos a usar en el script de Python: azure-keyvault-secrets y azure.identity.

En una máquina virtual Linux, por ejemplo, se puede usar pip3 para instalarlas:

```
pip3 install azure-keyvault-secrets
```

pip3 install azure.identity

Creación y edición del script de Python de ejemplo

En la máquina virtual, cree un archivo de Python llamado **sample.py**. Edite el archivo para que contenga el siguiente código, reemplazando <su-nombre-unico-de-almacen-de-claves> con el nombre de su almacén de claves.

```
from azure.keyvault.secrets import SecretClient

from azure.identity import DefaultAzureCredential


key_vault_name = "<your-unique-keyvault-name>"

key_vault_uri = f"https://{key_vault_name}.vault.azure.net"

secret_name = "mySecret"


credential = DefaultAzureCredential()

client = SecretClient(vault_url=key_vault_uri, credential=credential)

retrieved_secret = client.get_secret(secret_name)


print(f"The value of secret '{secret_name}' in '{key_vault_name}' is:
'{retrieved_secret.value}'")
```

Ejecución de la aplicación de Python de ejemplo

Por último, ejecute **sample.py**. Si todo ha ido bien, debería devolver el valor de su secreto:

```
python3 sample.py


The value of secret 'mySecret' in '<your-unique-keyvault-name>' is: 'Success!'
```

Limpieza de recursos

Cuando ya no sean necesarios, elimine la máquina virtual y el almacén de claves. Puede hacerlo rápidamente eliminando el grupo de recursos al que pertenecen:

```
az group delete -g myResourceGroup
```