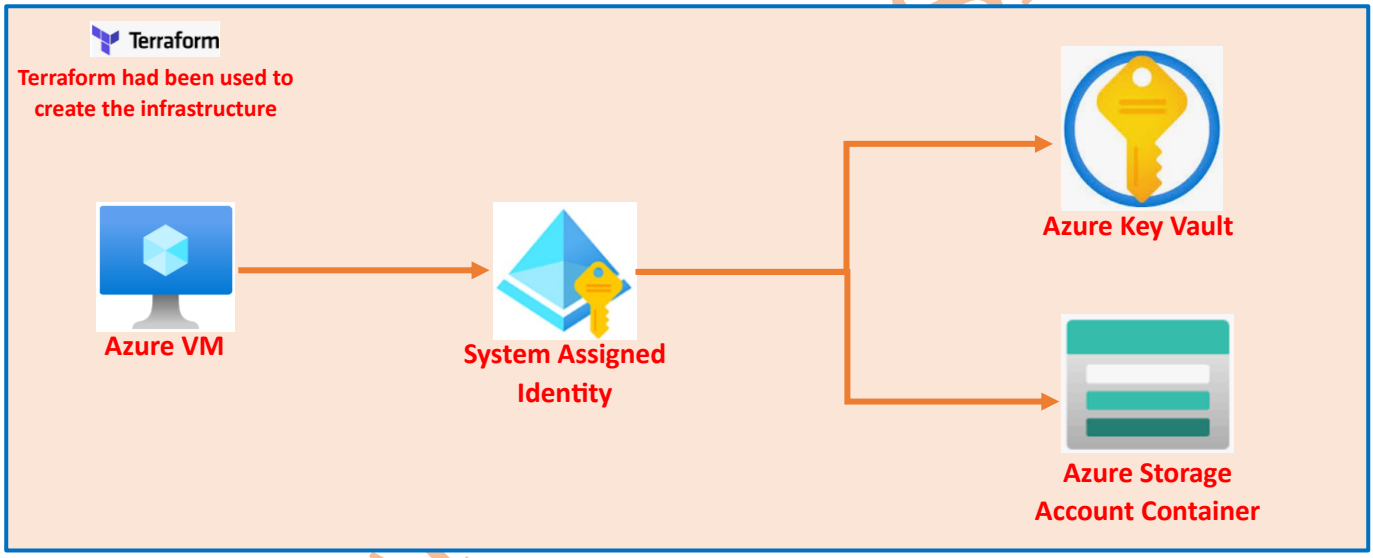
# AKS Workload Identity and Azure VM SystemAssigned Identity

Workload Identity in AKS (Azure Kubernetes Service) allows kubernetes pods to access Azure Resources without storing the credentials in the cluster. As for example Kubernetes pod can access the Azure key vaults to retrieve the secrets stored in the Azure Key Vaults and Kubernetes Pods can access the Azure Storage Account Container to retrieve the files stored in Azure Storage Account Container.

Federated Identity is a method by which a User can access multiple Applications using a single set of Applications.

OIDC (Open ID Connect) is a protocol by which a user sign-in to one Application and can access another Application.




Whenever an application running in Azure Resource tries to Access another Resource then you need Managed Identity, there are two types of Managed Identity one is System Assigned Identity and another is User Assigned Identity.

In the first part of this project, I am explaining Azure VM gets the access of Azure Key Vault or the Azure Storage Account Container using System Assigned Identity. I had created the infrastructure using Terraform. After creation of the Infrastructure I logged into the Azure VM using SSH and tried to test whether I had the Access for Azure Key Vault or Storage Account Container as shown in the Screenshot attached below.

```
curl -s https://dexterkeyvault.vault.azure.net/secrets/username?api-version=2016-10-01 -H "Authorization: Bearer `curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fvault.azure.net' -H Metadata:true| jq -r '.access_token'`"
```

```
[root@[redacted] ~]# curl -s https://dexterkeyvault.vault.azure.net/secrets/username?api-version=2016-10-01 -H "Authorization: Bearer `curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fvault.azure.net' -H Metadata:true| jq -r '.access_token'`"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1845  100  1845    0     0   200k      0 --:--:-- --:--:-- --:--:--  200k
{"value":"dexter","contentType":"","id":"https://dexterkeyvault.vault.azure.net/secrets/username/4[redacted]b","attributes":{"enabled":true,"created":1[redacted]9,"updated":1[redacted]9,"recoveryLevel":"CustomizedRecoverable+Purgeable"},"tags":{}}[root@[redacted] ~]#
```

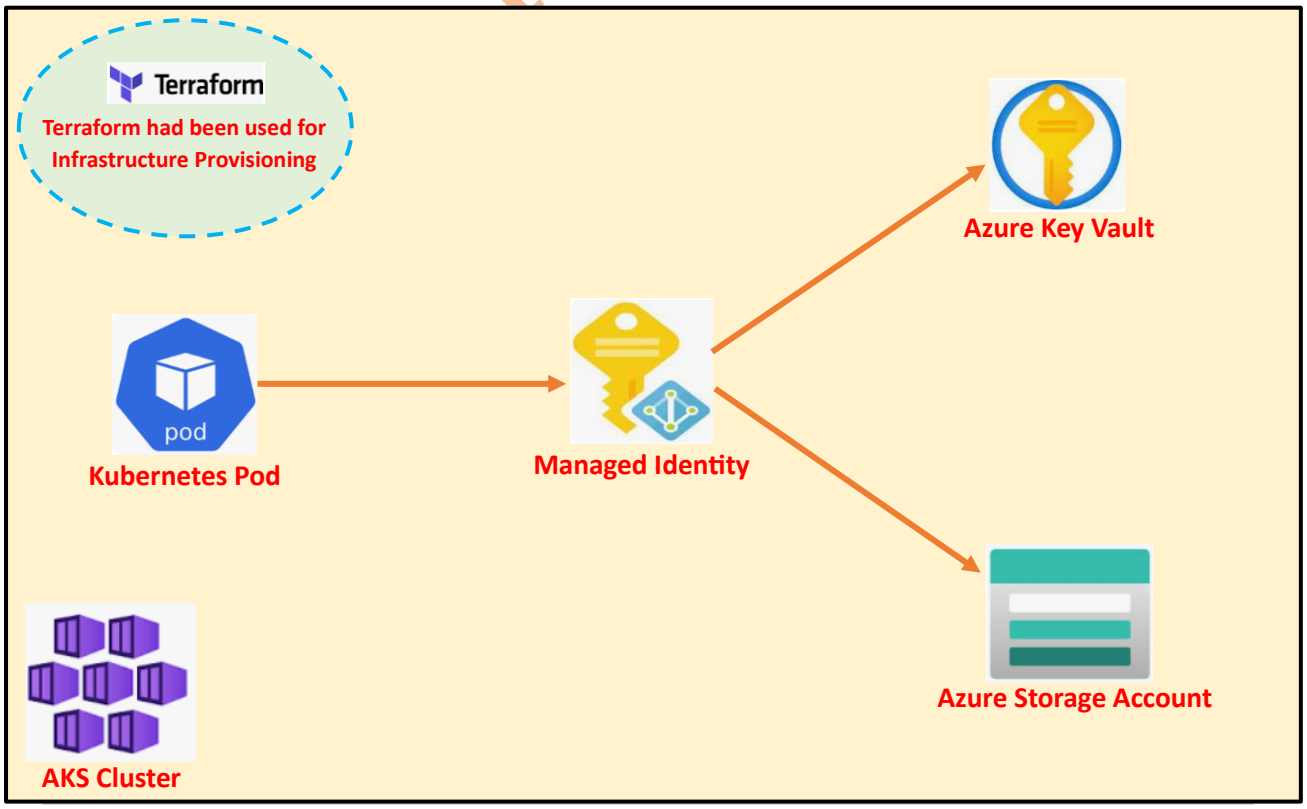
**curl "https://integrateakskvtachotacho.blob.core.windows.net/dexter/demo.txt" -H "x-ms-version: 2017-11-09" -H "Authorization: Bearer `curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fstorage.azure.com%2F' -H Metadata:true | jq -r '.access_token'`"**

```
[root@[redacted] ~]# curl "https://integrateakskvtachotacho.blob.core.windows.net/dexter/demo.txt" -H "x-ms-version: 2017-11-09" -H "Authorization: Bearer `curl 'http://169.254.169.254/metadata/identity/oauth2/token?api-version=2018-02-01&resource=https%3A%2F%2Fstorage.azure.com%2F' -H Metadata:true | jq -r '.access_token'`"
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  1724  100  1724    0     0   187k      0 --:--:-- --:--:-- --:--:--  187k
This is a Demo File.[root@[redacted] ~]#
```

As the result shown in the above two screenshots using the command shown above Azure VM had the privileges to access keyvault and storage account which was verified using the screenshot attached above. The first screenshot revealed the keyvault secret username as **dexter** and second screenshot revealed the file content of demo.txt which was present in the storage account named as integrateakskvtachotacho and container dexter.

In the second part of the project, I used workload Identity using which Kubernetes Pod inside the AKS Cluster had the privilege to access Storage Account Container blob and Azure Key Vault secrets.

I created PostgreSQL pod using deployment and its credentials I provided using the Keyvault. I rotated the credentials of Azure Key Vault and scaled down the replicas to zero and then scaled up the replicas to 1 then found new credentials had been taken place. There was a wait time introduce for 2 minutes after which the secrets had been changed as per the new Azure Key Vault.



The above architecture diagram showed high level architecture diagram of the second part of the project. The kubernetes pod inside the AKS Cluster used Managed Identity to access Azure Storage Account container blob and Azure Key Vault Secrets.

AKS Workload Identity enabled Kubernetes Pod to access Azure Resources without storing the credentials in the Cluster.

**AKS Workload Identity** uses the concept of **Federated Identity** and **OIDC** (Open ID Connect).

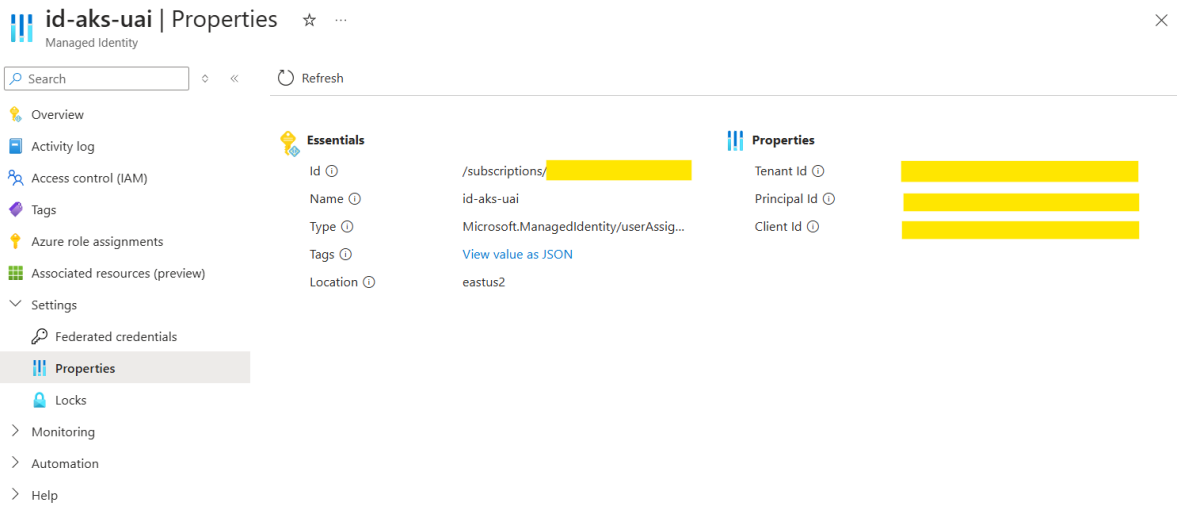
Federated Identity **enable users to access multiple applications with single set of credentials**.

OIDC is a **protocol** which enable users to **sign-in into one application and access another application**.

Created Federated Identity used the command as shown below.

```
az identity federated-credential create --name aksfederatedidentity --identity-name id-aks-uai --resource-group integrateakskv-rg --issuer `az aks show --resource-group integrateakskv-rg --name integrateakskv-cluster --query "oidcIssuerProfile.issuerUrl" -o tsv` --subject system:serviceaccount:dexter:dexter-sa
```

I had created Kubernetes Deployment using the Object ID and Tenant ID which I got from the Azure Managed Identity Console, you can also retrieve these details using the Azure CLI command as shown below.




```
az identity show -g integrateakskv-rg --name id-aks-uai --query 'clientId' -o tsv

az identity show -g integrateakskv-rg --name id-aks-uai --query 'tenantId' -o tsv
```

```
[root@ ~]# kubectl apply -f dexter.yaml
namespace/dexter created
secretproviderclass.secrets-store.csi.x-k8s.io/postgresql-kvname created
serviceaccount/dexter-sa created
deployment.apps/postgreqs-pod-kv created
service/dexter-service created
```

```
apiVersion: v1
kind: Namespace
metadata:
  name: dexter
---
apiVersion: secrets-store.csi.x-k8s.io/v1
kind: SecretProviderClass
metadata:
  name: postgresql-kvname # needs to be unique per namespace
  namespace: dexter
spec:
  provider: azure
  parameters:
    usePodIdentity: "false"
    clientID: "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX" # Setting this to use workload identity
    keyvaultName: "dexterkeyvault"      # Set to the name of your key vault
    cloudName: ""                       # [OPTIONAL for Azure] if not provided, the Azure environment
defaults to AzurePublicCloud
    objects:  |
      array:
        - |
          objectName: username          # Set to the name of your secret
          objectType: secret            # object types: secret, key, or cert
          objectVersion: ""             # [OPTIONAL] object versions, default to latest if empty
        - |
          objectName: password          # Set to the name of your key
          objectType: secret
          objectVersion: ""
    tenantId: "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"       # The tenant ID of the key vault
  secretObjects:                        # [OPTIONAL] SecretObjects defines the desired state of synced
Kubernetes secret objects
  - data:
```

```
  - key: USERNAME                    # data field to populate
    objectName: username             # name of the mounted content to sync; this could be the object name or the object alias
  - key: PASSWORD                    # data field to populate
    objectName: password             # name of the mounted content to sync; this could be the object name or the object alias
  secretName: userpass               # name of the Kubernetes secret object
  type: Opaque
---
apiVersion: v1
kind: ServiceAccount
metadata:
  annotations:
    azure.workload.identity/client-id: "XXXXXXXX-XXXX-XXXX-XXXX-XXXXXXXXXXXX"
  name: dexter-sa
  namespace: dexter
---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: postgreqs-pod-kv
  namespace: dexter
  labels:
    azure.workload.identity/use: "true"
spec:
  selector:
    matchLabels:
      app: PostgreSQL
  replicas: 1
  template:
    metadata:
      labels:
```

```
      app: PostgreSQL
      azure.workload.identity/use: "true"
    spec:
      serviceAccountName: "dexter-sa"
      containers:
      - name: postgresql
        image: postgres:14
        volumeMounts:
        - name: postgresql-store
          mountPath: "/mederma"
          readOnly: true
        env:
        - name: POSTGRES_USER
          valueFrom:
            secretKeyRef:
              name: userpass
              key: USERNAME
        - name: POSTGRES_PASSWORD
          valueFrom:
            secretKeyRef:
              name: userpass
              key: PASSWORD
      volumes:
      - name: postgresql-store
        csi:
          driver: secrets-store.csi.k8s.io
          readOnly: true
          volumeAttributes:
            secretProviderClass: "postgresql-kvname"
---
apiVersion: v1
```

kind: Service

metadata:

  name: dexter-service

  namespace: dexter

spec:

  selector:

    azure.workload.identity/use: "true"

  type: LoadBalancer

  ports:

  - targetPort: 5432

    port: 5432

```
[root@                ~]# kubectl get secrets -n dexter
NAME        TYPE      DATA   AGE
userpass    Opaque    2      3m19s
[root@AzureVM-Server ~]# kubectl get csidriver -n dexter
NAME                      ATTACHREQUIRED   PODINFOONMOUNT   STORAGECAPACITY   TOKENREQUESTS                REQUIRESREPUBLISH   MODES                  AGE
disk.csi.azure.com        true             false            false             <unset>                      false               Persistent             3h27m
file.csi.azure.com        false            true             false             <unset>                      false               Persistent,Ephemeral   3h27m
secrets-store.csi.k8s.io  false            true             false             api://                       false               Ephemeral              3h27m
[root@                ~]# kubectl get sa -n dexter
NAME        SECRETS   AGE
default     0         10m
dexter-sa   0         10m
[root@                ~]# kubectl get pods -n dexter
NAME                                READY   STATUS    RESTARTS   AGE
postgreqs-pod-kv-          -fwb5m   1/1     Running   0          11m
[root@                ~]# kubectl get svc -n dexter
NAME             TYPE           CLUSTER-IP    EXTERNAL-IP    PORT(S)          AGE
dexter-service   LoadBalancer   10.    .158   20.      .51   5432:3    4/TCP   11m
```

Created Kubernetes Pods, Service, Kubernetes Secrets, CSI Driver and mounted that as a volume which was shown in the screenshot attached below.

Initially the credentials were as shown below.

```
[root@                ~]# kubectl get secrets -n dexter
NAME        TYPE      DATA   AGE
userpass    Opaque    2      16m
[root@                ~]# kubectl describe secrets userpass -n dexter
\Name:         userpass
Namespace:    dexter
Labels:       secrets-store.csi.k8s.io/managed=true
Annotations:  <none>

Type:  Opaque

Data
====
PASSWORD:  8 bytes
USERNAME:  6 bytes
[root@                ~]# kubectl get secrets/userpass --template={{.data.PASSWORD}} | base64 -d
Error from server (NotFound): secrets "userpass" not found
[root@                ~]# kubectl get secrets/userpass -n dexter --template={{.data.PASSWORD}} | base64 -d
Admin123[root@                ~]# kubectl get secrets/userpass -n dexter --template={{.data.USERNAME}} | base64 -d
dexter[root@                ~]#
```

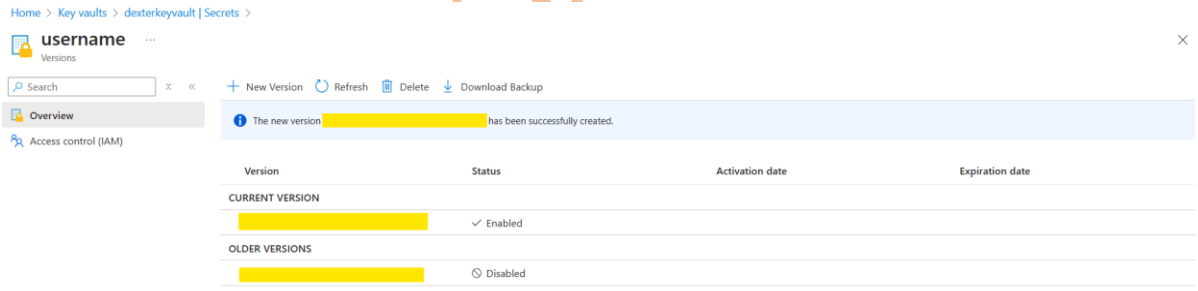
I logged-in into the PostgreSQL using the old credentials as shown below.

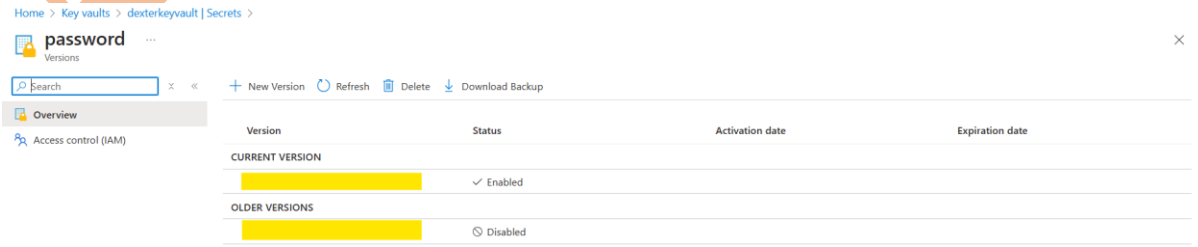
```
[root@                ~]# kubectl exec -it postgreqs-pod-kv-                bash -n dexter -- cat /mederma/username
dexter[root@                ~]# kubectl exec -it postgreqs-pod-kv-                bash -n dexter -- cat /mederma/password
Admin123[root@                ~]#
```

```
[root@                ~]# kubectl get svc -n dexter
NAME             TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
dexter-service   LoadBalancer   10.     .158   20.       .51   5432:31164/TCP   37m
[root@                ~]# psql -h 20.       .51 -U dexter --password
Password:
psql (14.15)
Type "help" for help.

dexter=# \l
                                 List of databases
   Name    | Owner  | Encoding |  Collate   |   Ctype    | Access privileges
-----------+--------+----------+------------+------------+-------------------
 dexter    | dexter | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres  | dexter | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | dexter | UTF8     | en_US.utf8 | en_US.utf8 | =c/dexter        +
           |        |          |            |            | dexter=CTc/dexter
 template1 | dexter | UTF8     | en_US.utf8 | en_US.utf8 | =c/dexter        +
           |        |          |            |            | dexter=CTc/dexter
(4 rows)

dexter=#
```

I changed the credentials from the Azure Key Vault and wait for 2 minutes and the same was reflected in the Kubernetes Secrets then scaled down the pod to replicas equal to zero and finally scaled up the pod to replicas equal to one. Finaly logged-in into the PostgreSQL pod with new credentials and I was able to logged-in.

Created newer version of Azure Key Vault Secrets and disabled older version as shown in the screenshot attached below.




After 2 minutes username and password had been changed in kubernetes secrets as shown below.

```
[root@                ~]# kubectl get secrets/userpass -n dexter --template={{.data.USERNAME}} | base64 -d
therema[root@              kubectl get secrets/userpass -n dexter --template={{.data.PASSWORD}} | base64 -d
Mashroof123[root@                ~]#
```

And in the kubernetes pod, I was able to logged-in with the new username and password as shown below.

```
[root@                ~]# kubectl scale deploy postgreqs-pod-kv --replicas=0 -n dexter
deployment.apps/postgreqs-pod-kv scaled
[root@                ~]# kubectl scale deploy postgreqs-pod-kv --replicas=1 -n dexter
deployment.apps/postgreqs-pod-kv scaled
[root@                ~]# kubectl get pods -n dexter
NAME                                READY   STATUS    RESTARTS   AGE
postgreqs-pod-kv-                   1/1     Running   0          3s
[root@                ~]# kubectl exec -it postgreqs-pod-kv-                bash -n dexter -- cat /mederma/username
therema[root@                ~]# kubectl exec -it postgreqs-pod-kv-                bash -n dexter -- cat /mederma/password
Mashroof123[root@                ~]#
[root@                ~]# kubectl get svc -n dexter
NAME             TYPE           CLUSTER-IP     EXTERNAL-IP     PORT(S)          AGE
dexter-service   LoadBalancer   10.      .85   48.      .87   5432:30828/TCP   6m19s
[root@AzureVM-Server ~]# psql -h 48.      .87 -U therema --password
Password:
psql (14.15)
Type "help" for help.

therema=# \l
                                  List of databases
   Name    |  Owner  | Encoding |  Collate   |   Ctype    |  Access privileges
-----------+---------+----------+------------+------------+---------------------
 postgres  | therema | UTF8     | en_US.utf8 | en_US.utf8 |
 template0 | therema | UTF8     | en_US.utf8 | en_US.utf8 | =c/therema         +
           |         |          |            |            | therema=CTc/therema
 template1 | therema | UTF8     | en_US.utf8 | en_US.utf8 | =c/therema         +
           |         |          |            |            | therema=CTc/therema
 therema   | therema | UTF8     | en_US.utf8 | en_US.utf8 |
(4 rows)

therema=#
```

Created Kubernetes Deployment and hence the Pod using which I accessed the Azure Storage Account Container blob as shown in the screenshot attached below.

```
[root@[redacted] ~]# kubectl apply -f medoko.yaml
deployment.apps/get-storage created
[root@[redacted] ~]# cat medoko.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: get-storage
  namespace: dexter
  labels:
    app: get-storage
spec:
  replicas: 1
  selector:
    matchLabels:
      app: get-storage
  template:
    metadata:
      labels:
        app: get-storage
        azure.workload.identity/use: "true"
    spec:
      serviceAccountName: dexter-sa
      containers:
        - name: get-storage
          image: chester2004/get-storage
          ports:
            - containerPort: 8080
```

```
cat medoko.yaml
apiVersion: apps/v1
kind: Deployment
metadata:
  name: get-storage
  namespace: dexter
  labels:
    app: get-storage
spec:
  replicas: 1
  selector:
    matchLabels:
      app: get-storage
  template:
    metadata:
      labels:
        app: get-storage
        azure.workload.identity/use: "true"
    spec:
      serviceAccountName: dexter-sa
      containers:
        - name: get-storage
          image: chester2004/get-storage
          ports:
            - containerPort: 8080
```

Kubernetes Pod was able to access the Azure Storage Account Container blob as shown in the screenshot attached below.

```
[root@[redacted] ~]# kubectl get pods -n dexter
NAME                                  READY   STATUS    RESTARTS   AGE
get-storage-[redacted]                1/1     Running   0          7m16s
postgreqs-pod-kv-[redacted]           1/1     Running   0          16m
[root@[redacted] ~]# kubectl exec -it get-storage-[redacted] bash -n dexter -- curl http://localhost:8080/api/integrateakskvtachotacho/dexter/demo.txt
 -o myfile
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100    20  100    20    0     0     15      0  0:00:01  0:00:01 --:--:--    15
[root@AzureVM-Server ~]# kubectl exec -it get-storage-[redacted] bash -n dexter -- cat myfile
This is a Demo File.[root@[redacted] ~]#
```

I had created PostgreSQL database using terraform with Username and Password which is taken from the Key Vault username and password. You can reset its credentials either through Azure Console or using Azure CLI.

```
[root@                ~]# psql -h sonarqube-postgresql3.postgres.database.azure.com -d postgres -U dexter --password
Password:
psql (14.15, server 14.13)
SSL connection (protocol: TLSv1.3, cipher: TLS_AES_256_GCM_SHA384, bits: 256, compression: off)
Type "help" for help.

postgres=> \l
                                                  List of databases
       Name        |     Owner      | Encoding |  Collate   |   Ctype    |         Access privileges
-------------------+----------------+----------+------------+------------+-----------------------------------
 azure_maintenance | azuresu        | UTF8     | en_US.utf8 | en_US.utf8 |
 azure_sys         | azuresu        | UTF8     | en_US.utf8 | en_US.utf8 |
 postgres          | azure_pg_admin | UTF8     | en_US.utf8 | en_US.utf8 |
 template0         | azure_pg_admin | UTF8     | en_US.utf8 | en_US.utf8 | =c/azure_pg_admin                +
                   |                |          |            |            | azure_pg_admin=CTc/azure_pg_admin
 template1         | azure_pg_admin | UTF8     | en_US.utf8 | en_US.utf8 | =c/azure_pg_admin                +
                   |                |          |            |            | azure_pg_admin=CTc/azure_pg_admin
(5 rows)

postgres=>
```