**WHY AZURE KEY VAULT? WHY NOT HARDOCDED?**

**So here is sample(edited my information for security reasons) json format document that I got : I changed the encrypted crredential**

**{**

**"name": "SqlServer1\_hardcoded",**

**"type": "Microsoft.DataFactory/factories/linkedservices",**

**"properties": {**

**"annotations": [],**

**"type": "SqlServer",**

**"typeProperties": {**

**"server": "hello\\SQLEXPRESS",**

**"database": "AdventureWorksLT2022",**

**"encrypt": "mandatory",**

**"trustServerCertificate": true,**

**"authenticationType": "SQL",**

**"userName": "lahari",**

**"encryptedCredential": "hfugfwqvbasbxnabxjAXau** **$2a$10$L.oD3JDF2gNGZ8/J.3WxxRdcY5ErPAlw3V5vqV0Ra2X3IfN9h8Kz6"**

**},**

**"connectVia": {**

**"referenceName": "LinekedintegrationRuntime1",**

**"type": "IntegrationRuntimeReference"**

**}**

**}**

**}**

**The encryptedCredential field in your linked service contains an encrypted version of the password for SQL authentication. Here's a detailed explanation of what encrypted credentials are and what someone can do with them:**

**What is an Encrypted Credential?**

**When you see "encryptedCredential", it refers to a password or sensitive information that has been encrypted by Azure Data Factory before it’s stored. The purpose of encryption is to keep sensitive data, like passwords, secure from unauthorized access. The encrypted value can’t be read directly unless decrypted using the proper keys or credentials.**

**What Can Someone Do with an Encrypted Credential?**

* **Without the Key:  
  If someone only has access to the encryptedCredential field (i.e., the encrypted value), they cannot directly access the password. The encrypted string is not useful on its own without the decryption key that was used to encrypt it.**

**So, if someone tries to view the encrypted credential without the proper access (such as the appropriate encryption keys or access rights), the encrypted value would appear as gibberish, and they would not be able to extract the original password.**

* **With Access to Decryption Keys:  
  If an unauthorized user gains access to the decryption keys or appropriate permissions (e.g., managed identity permissions to access Data Factory resources), they could potentially decrypt the credential and access the original password. This highlights why it's critical to control who has access to your Azure Data Factory instance and the credentials stored in it.**

**If someone has decryption access (e.g., via an Azure role assignment or an access policy that gives them permission to manage the linked service), they could:**

* + **Decrypt the password.**
  + **Use it to connect to the SQL Server.**
  + **Potentially perform actions such as reading, modifying, or deleting data from the SQL database.**

**Why is Encryption Important?**

**Encryption provides an extra layer of security to prevent passwords and other sensitive information from being easily compromised, even if someone gains access to the configuration or JSON files where the credentials are stored. The encrypted form itself, without proper decryption mechanisms, is not usable.**

**Security Measures in Azure Data Factory**

* **Encryption at Rest:  
  All sensitive data, including the encryptedCredential value, is encrypted at rest within Azure Data Factory’s storage systems. This means that the value is stored in an encrypted form, and only authorized systems/services can decrypt it.**
* **Managed Identity Access:  
  To further secure credentials, Azure Data Factory uses Managed Identity to authenticate itself to other Azure services, including Azure Key Vault (if you decide to use Key Vault for secrets). Only the services that have been granted access can retrieve or decrypt the credentials.**

**While using Azure key vault:**

**RBAC vs Access Policies**

1. **RBAC (Role-Based Access Control):**
   * **Applies at the resource level** (e.g., Key Vault, subscription, resource group).
   * Manages permissions across resources, such as giving users roles like **Key Vault Administrator** or **Key Vault Secrets User**.
   * **If a Key Vault is using RBAC**, permissions are applied based on **roles assigned to users/groups** at the **resource level**.
2. **Access Policies:**
   * **Applies at the Key Vault level** and controls **specific permissions for secrets, keys, and certificates** inside the vault.
   * Provides more **granular control** at the **resource level**, where you define who can access individual secrets, keys, or certificates.
   * **If a Key Vault is using Access Policies**, then **RBAC is disabled** for that Key Vault, and the Key Vault's contents are controlled only by the policies you define.
   * **You need to manually set permissions** for each secret/key you want to control.

**What Happens When You Switch Between the Two:**

* If you use **RBAC**, it works across the whole Key Vault and applies permissions at the **resource level** (which is easier for managing high-level access, e.g., allowing full access to all secrets).
* If you switch to **Access Policies**, then **RBAC is no longer in effect** for that Key Vault, and permissions are managed **individually for secrets, keys, and certificates**.

**Key Insight:**

* **RBAC** is simpler for **resource-level control** but less flexible for **fine-grained control** over secrets, keys, and certificates.
* **Access Policies** are more granular but **disable RBAC** in favor of a more detailed configuration for individual resources inside the Key Vault.

**In Short:**

* **Use RBAC** for **high-level resource management** across your Azure resources (like managing access to all secrets in a Key Vault).
* **Use Access Policies** when you need **detailed, per-secret** or **per-key** control inside the Key Vault.

**WHAT HAPPENS IF WE HAVE TWO KEY VALUTS IN A RESOURCE GROUP:**

**so is it possible to have two key vaults fir a single resource group?**

**Here's why and how it works:**

1. **Key Vault as a Resource**: Key Vault is a resource like any other Azure resource (e.g., virtual machines, storage accounts, etc.). You can create as many Key Vaults as needed within a **resource group**, as long as it fits within your Azure subscription limits.
2. **Why Use Multiple Key Vaults**:
   * **Separation of Concerns**: For example, you may want to separate secrets, keys, or certificates for different applications or environments (e.g., Development, Staging, Production).
   * **Security**: By using separate Key Vaults, you can apply different access control policies or encryption settings for each vault.
   * **Resource Organization**: Multiple Key Vaults can help organize resources based on different projects or teams within the same resource group.
3. **Resource Group Limitations**: A **resource group** is just a container for resources. There is no hard limitation on the number of Key Vaults you can have in a resource group, but there are subscription-level limits for resources within a region (e.g., maximum number of resources per subscription, and the number of Key Vaults per subscription).

**Example Use Cases:**

* **Environment Segregation**: You could have one Key Vault for **Development**, another for **Production**, both in the same resource group but with different access controls.
* **Application Segregation**: You might create separate Key Vaults for different applications (e.g., one Key Vault for App A’s secrets and another for App B’s secrets).

**Key Vault Naming:**

Each Key Vault within the same resource group must have a **unique name**, as Key Vault names are globally unique across all of Azure.

So, in conclusion, yes, you can have multiple Key Vaults in a single resource group, and this allows you to organize and manage resources more effectively.

Let me know if you'd like more details on how to set this up or any other questions! 😊

4o mini

**You said:**

so ths will allow the resource group to have only one RBAC but have access policies specific to the keyvalut am i right

**ChatGPT said:**

Yes, you're correct! Here’s a breakdown of what happens:

**Key Vaults and Resource Groups:**

1. **Resource Group**:
   * A **resource group** itself can have **one RBAC** model applied at the **resource group level**.
   * **RBAC roles** can be assigned to users, groups, or managed identities for **all resources** in the resource group, including the **Key Vaults** within that group.
2. **Key Vaults**:
   * For each **Key Vault** within the resource group, you can either use **RBAC** or **Access Policies**, but **Access Policies** are applied **at the Key Vault level**. This means each Key Vault can have **fine-grained control** over who can access specific secrets, keys, or certificates inside it.
   * **Access Policies** will work **independently of RBAC** for that Key Vault once enabled, giving you control over the permissions granted to users for specific contents of the vault.

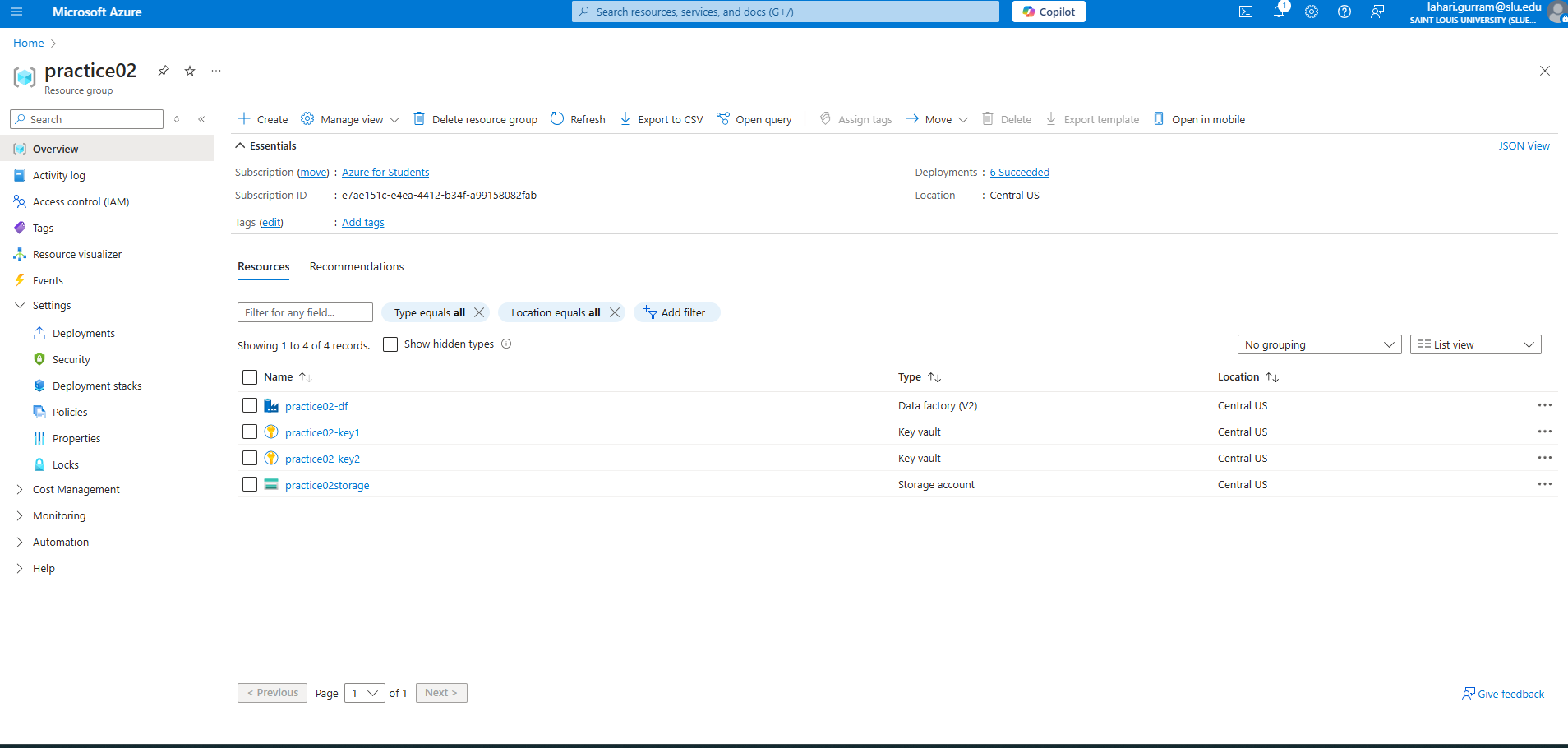
**Scenario:**

* **RBAC at the Resource Group Level**:
  + If you assign a role like **Key Vault Administrator** at the **resource group** level, that role will apply to all Key Vaults in that group, granting users administrative control over those vaults as a whole (but not the contents).
* **Access Policies at the Key Vault Level**:
  + If you have multiple Key Vaults in the same resource group, you can set up **different Access Policies** for each Key Vault, depending on the specific needs for access to secrets, keys, and certificates inside those vaults.
  + For example, one Key Vault might only allow certain users to access secrets, while another might allow more granular control over the keys and certificates.

**In Short:**

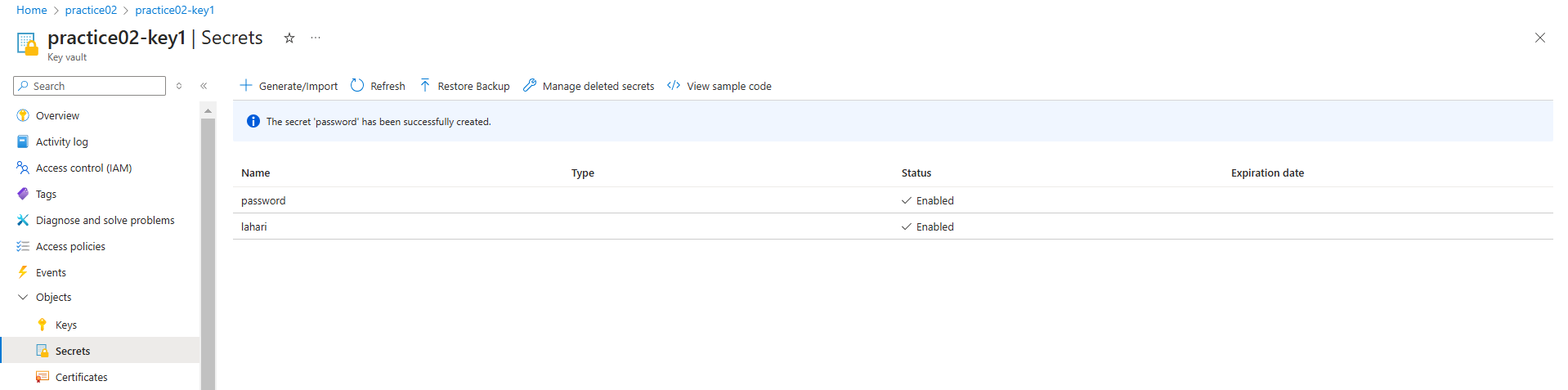
* The **resource group** can use **RBAC** for managing access to all resources within the group (including Key Vaults).
* Each **Key Vault** can have its own **Access Policies** for more granular control of what users can access **inside the vault** (e.g., which secrets/keys they can read, update, or delete).
* The two models (**RBAC** and **Access Policies**) can coexist, and you can mix and match them depending on your needs.

TWO RESOURCE GROUPS IN A SINGLE RESOURCE GROUP:;

****

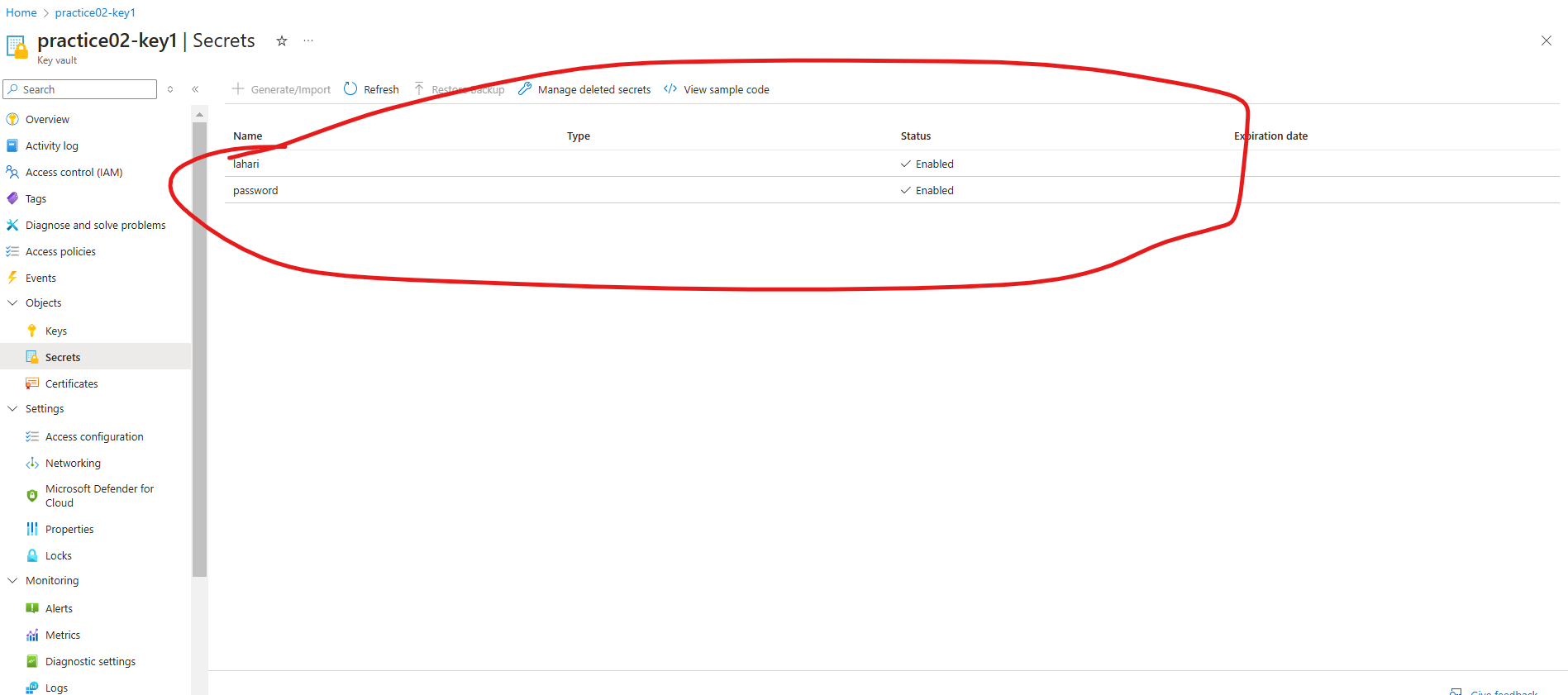
**IN KEY VALUT1:**

When I applied key valut administrator to myself ; access enabled to RBAC;

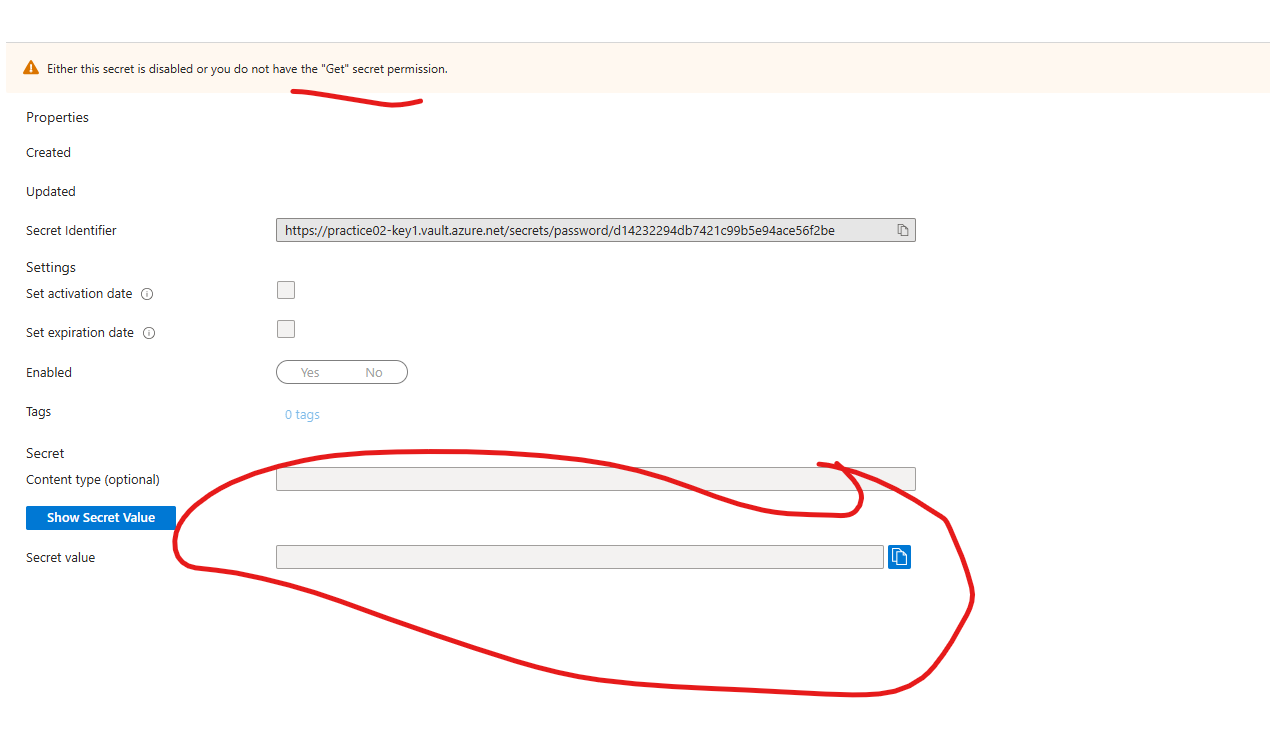
I am able to create delete, and see the secrets , so I created two secrets as follows: 

Now I am changing it to access policies=🡺 where I created an access policy for myself like only permission --🡪 list (ONLY)

Now I am verifying if it works or not

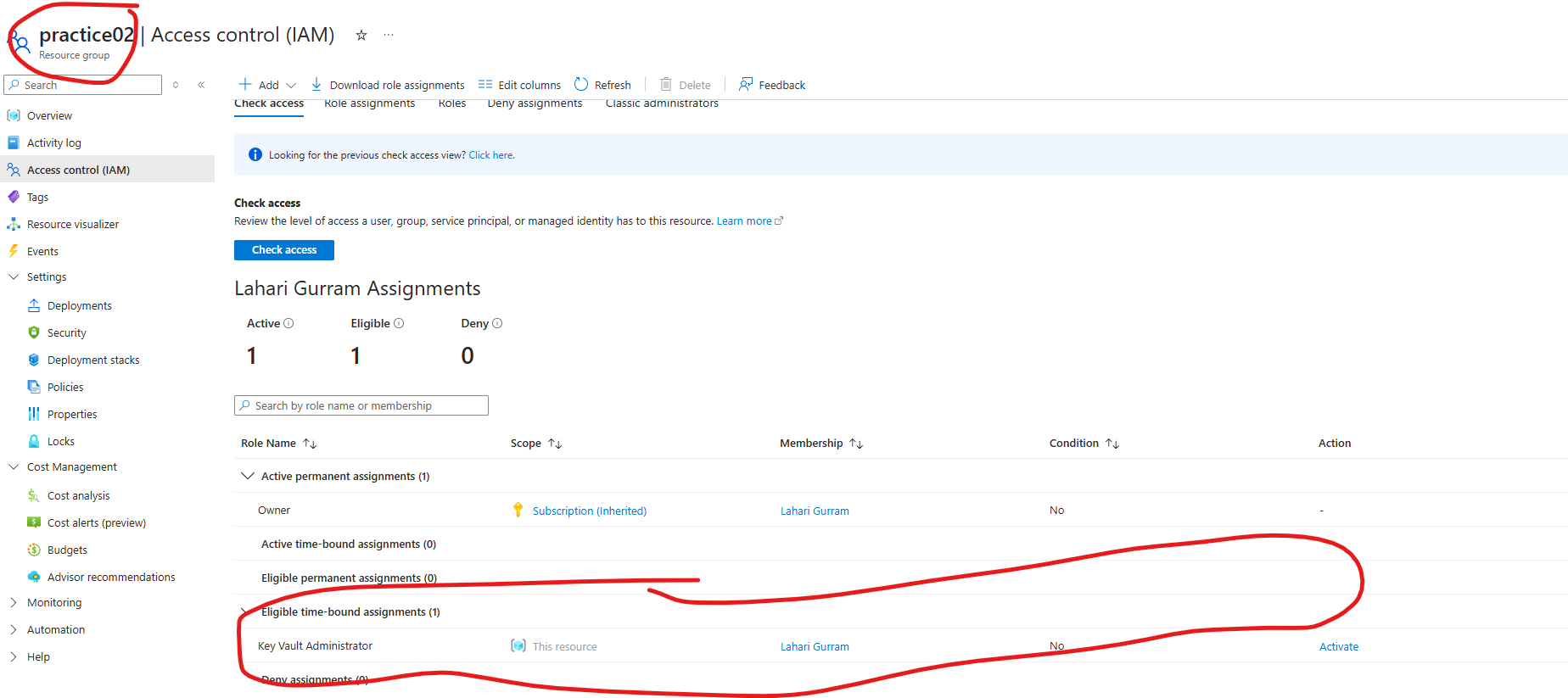


I can see the list but; I cant see the key present inside because there is not GET permission:

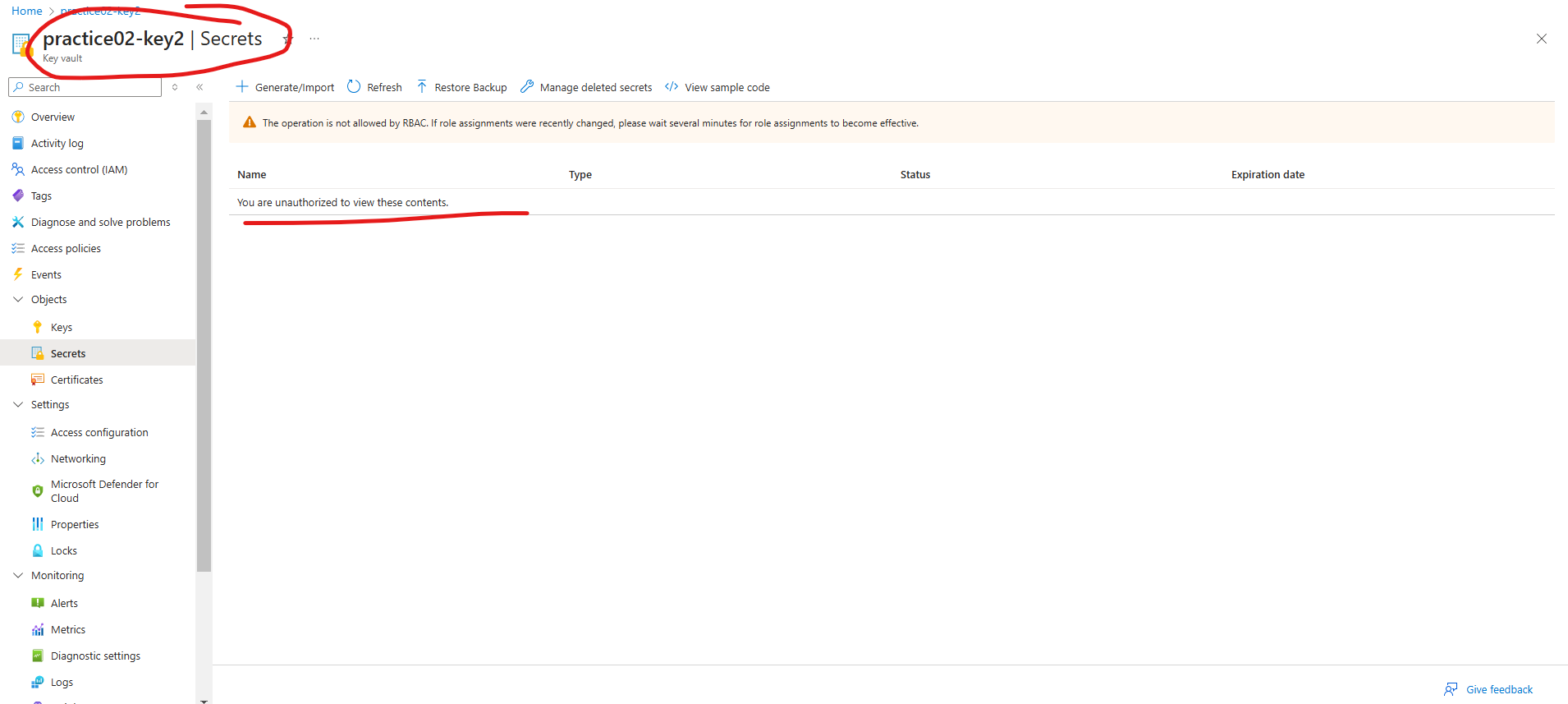
Ok So 

So we understood that when we enable access policies ; RBAC policies will not work

2 KEY VAULTS, KEY VAULT ADMINISTRATOR(RESORCE LEVEL)



Now I wanna see if the RBAC work for the second key vault:

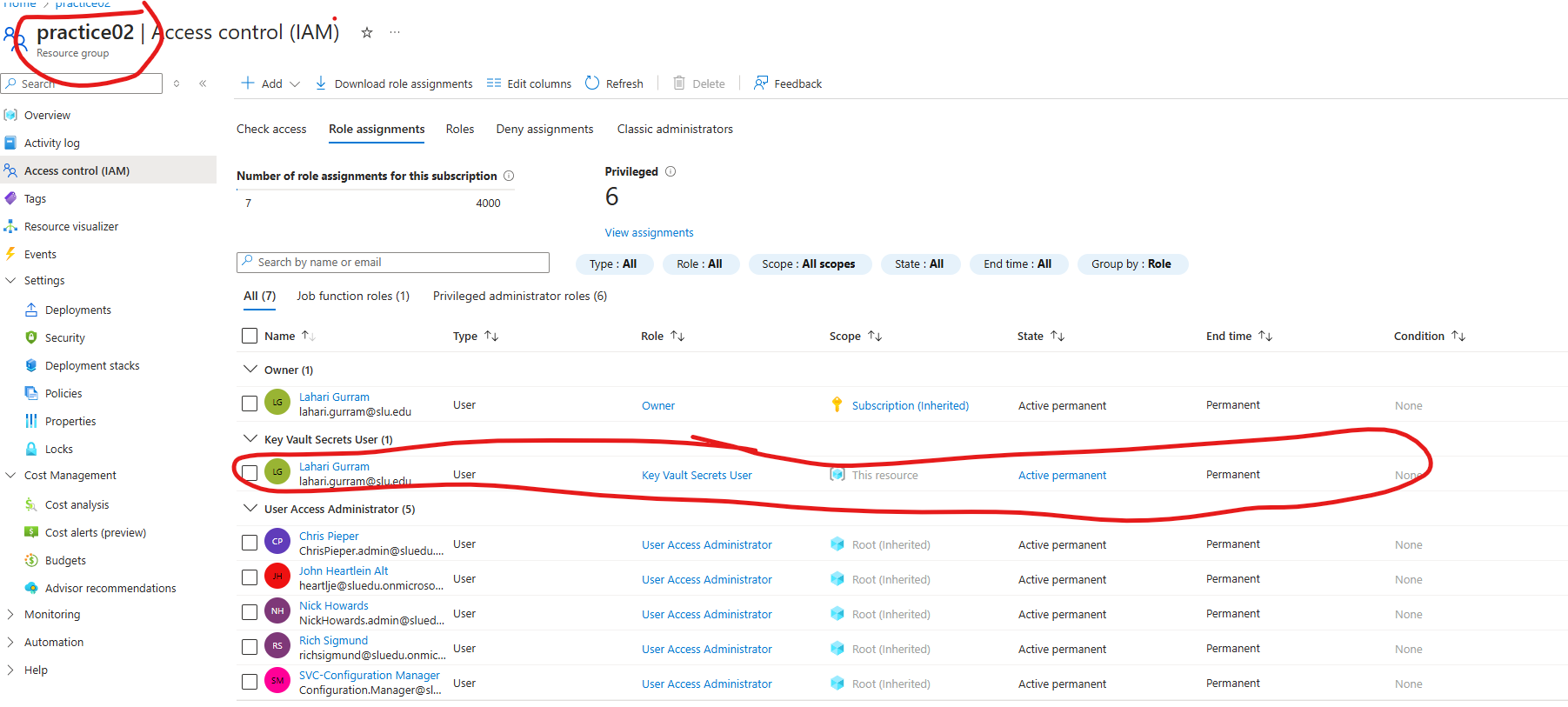


IT WILL NOT BECAUSE:

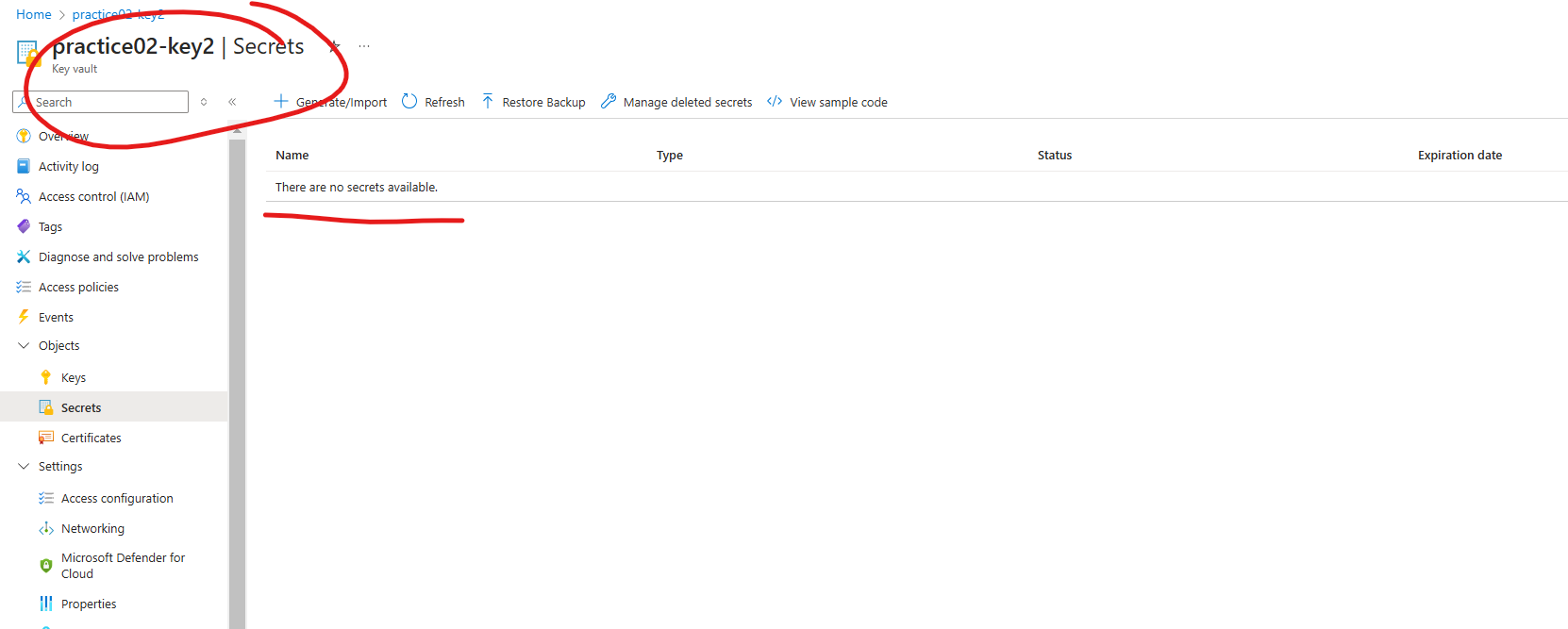
Here are the key points to understand and resolve this issue:

1. **Role Assignment Scope**:
   * The **Key Vault Administrator** role grants permissions to perform management tasks at the **resource** level (like configuring access policies or assigning roles). However, it does not automatically provide access to view or manage the **data plane** objects (secrets, keys, certificates) inside the Key Vault.
   * You need additional permissions, such as **Key Vault Secrets User**, to be able to view secrets within the Key Vault.
2. **Permissions for Data Plane Objects**: To access the **secrets** in Key Vault 2, you need to ensure you have the appropriate role at the **data plane** level. In particular, you might need the following:
   * **Key Vault Secrets User**: Allows you to read secrets.
   * **Key Vault Administrator**: Allows you to manage the Key Vault itself, but **doesn't** grant access to data plane objects like secrets.

So let me change the role to : **Key Vault Secrets User**



Now it is aloowing me to see if any secrets are available:



Therfore:

RBAC applied at resource level works for all the keyvaults( **BASED ON THE ROLE**)