### **Agenda:** **Key Vault**

### Secrets vs Keys

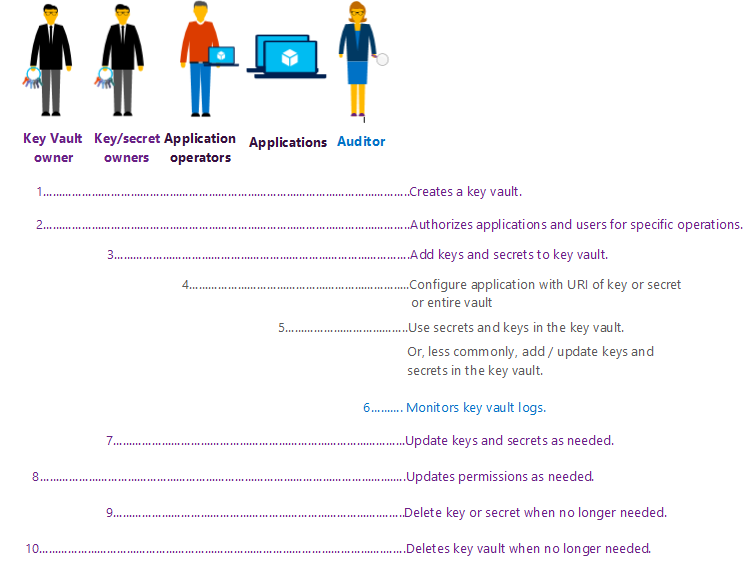
### Accessing Keys and Secrets

### Use cases for Key Vault

### **About Key Vault**

### Key Vault serves as a store of cryptographic keys and secrets, such as authentication keys, storage account keys, data encryption keys, .PFX files, and passwords.

* Developers can create keys for development and testing in minutes, and then seamlessly migrate them to production keys.
* Security administrators can grant (and revoke) permission to keys, as needed.



**Benefits:**

* Increase security and control over keys and passwords.
* Use FIPS 140-2 Level 2 validated hardware security modules (HSMs).
* Reduce latency with cloud scale and global redundancy.
* Create and import encryption keys in minutes.
* Applications have no direct access to keys.
* Simplify and automate tasks for SSL/TLS certificates.

**Secrets vs Keys**

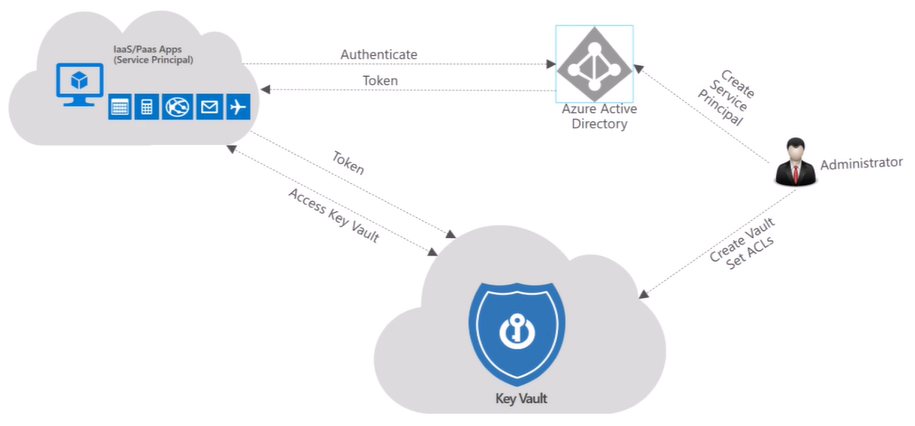
* A **secret** is essentially a small data blob (of up to 10 KB in size) that authorized users and applications **can retrieve from the vault**. To secure access to secrets, you create Azure Active Directory objects representing these users or applications, which they subsequently use to authenticate. Effectively, you avoid potential risk associated with users storing secrets in non-secure locations and eliminate the need to hard-code them into applications.
* Unlike secrets, **keys** stored in a vault **do not leave its boundaries**. Instead, once you add a key to the vault, users and applications must invoke cryptographic functions to perform any operations that require its knowledge. On the other hand, the ability to complete such invocation is also subject to a successful Azure Active Directory-based authentication.

**How to access keys and secrets:**

* To access keys and secrets, users and applications must possess valid Azure Active Directory tokens representing security principal with sufficient permissions to the target vault.
* You can use a REST-based API or Azure PowerShell to retrieve secrets and public parts of keys (in JSON format) from Key Vault.

**How it works:**

1. Applications that need access to Key Vault are registered with Azure Active Directory as Service Principals.
2. Administrator will then create a Key Vault and sets Access Control Lists on the vault so that applications can access it.
3. Applications then authenticates with Azure Active Directory and gets the Token.
4. The application then presents this token to Azure Key Vault
5. Azure Key vault then grants access based on Access Control List (ACL)



Administrator would sign in with an Azure subscription, create a vault for the organization in which to store keys, and then be responsible for operational tasks, such as:

* Create or import a key or secret.
* Revoke or delete a key or secret.
* Authorize users or applications to access the key vault, so they can then manage or use its keys and secrets.
* Configure key usage (for example, sign or encrypt).
* Monitor key usage.

This administrator would then provide developers with URIs to call from their applications, and provide their security administrator with key usage logging information.

**Step 2: Create Key Vaults**

1. Azure Portal 🡪 Search Key Vaults
2. + Create
3. Enter Name = sandeepdemo-keyvault
4. Access configuration tab
   1. Select Vault access policy
   2. Select **Resource access** as required
      1. Enable access to Azure VM for deployment
      2. Enable access to Azure Resource Manager for template deployment
      3. Enable access to Azure Disk Encryption for volume encryption
   3. Note under **Access** **policies** section, your identity is already present with all permissions for Key, Secrets and Certificates
5. Review + create

**Step 3: Add a key and secret to the key vault**

1. Select the Key Valut 🡪 Keys 🡪 + Generate/Import 🡪 Name=DemoKey1 🡪 Create
   1. Select Key 🡪 Current Version 🡪 Copy Key Identifier
2. Select the Key Valut 🡪 Secrets 🡪 + Generate/Import 🡪 Name=DemoSecret1, Secret value = Demo 🡪 Create
   1. Select Select 🡪 Current Version 🡪 Copy Secret Identifier

Note: You can reference a key that you created or uploaded to Azure Key Vault by using its URI.

To get the latest version, you can use <https://dssdemokeyvalut.vault.azure.net/keys/MyFirstKey>

To get this specific version: [<https://dssdemokeyvalut.vault.azure.net/keys/MyFirstKey>/cgacf4f763ar42ffb0a1gca546aygd87](https://contosokeyvault.vault.azure.net/keys/ContosoFirstKey/cgacf4f763ar42ffb0a1gca546aygd87)

**Managed Identity**

**System Assigned Identity**: A system assigned managed identity enables Azure resources to authenticate to cloud services (e.g. Azure Key Vault) without storing credentials in code. Once enabled, all necessary permissions can be granted via Azure role-based-access-control. The lifecycle of this type of managed identity is tied to the lifecycle of this resource. Additionally, each resource (e.g. Virtual Machine) can only have one system assigned managed identity.

**User Assigned:** User assigned managed identities enable Azure resources to authenticate to cloud services (e.g. Azure Key Vault) without storing credentials in code. This type of managed identities are **created as standalone Azure resources**, and have their own lifecycle. A single resource (e.g. Virtual Machine) can utilize multiple user assigned managed identities. Similarly, a single user assigned managed identity can be shared across multiple resources (e.g. Virtual Machine).

|  |  |  |
| --- | --- | --- |
| **Property** | **System-assigned managed identity** | **User-assigned managed identity** |
| Creation | Created as part of an Azure resource | Created as a stand-alone Azure resource |
| Life cycle | Shared life cycle with the Azure resource | Independent life cycle  Must be explicitly deleted |
| Sharing across Azure resources | Cannot be shared  Can only be associated with a single Azure resource | Can be shared  Can be associated with more than one Azure resource |
| Common use cases | Workloads that are contained within a single Azure resource  Workloads for which you need independent identities.  For example, an application that runs on a single virtual machine | Workloads that run on multiple resources and which can share a single identity  Workloads that need pre-authorization to a secure resource as part of a provisioning flow.  Workloads where resources are recycled frequently, but permissions should stay consistent. |