Microsoft

AZ-204T00A

Learning Path 07: Implement secure cloud solutions

Agenda

- Implement Azure Key Vault
- Implement managed identities
- Implement Azure App Configuration

Module 1: Implement Azure Key Vault

Learning objectives

- Describe the benefits of using Azure Key Vault
- Explain how to authenticate to Azure Key Vault
- Set and retrieve a secret from Azure Key Vault by using the Azure CLI

Introduction

- Azure Key Vault is a cloud service for securely storing and accessing secrets.
- A secret is anything that you want to tightly control access to, such as API keys, passwords, certificates, or cryptographic keys.

Explore Azure Key Vault

Azure Key Vault provides

- **Secrets Management:** For safe storage and strict control
- **Key Management:** Used as a key management solution
- **Certificate Management:** Provision, manage, and deploy public and private SSL/TLS certificates

Key benefits of using Azure Key Vault

- **Centralized application secrets:** Control their distribution.
- **Securely store secrets and keys:** Requires proper authentication and authorization to gain access
- **Monitor access and use:** Enable logging to monitor activity.
- **Simplified administration of application secrets:** Follow the life cycle and have high availability.

Discover Azure Key Vault best practices (1 of 2)

Authenticating to Azure Key Vault

Three ways to authenticate

- **Managed identities for Azure resources**: When you deploy an app on a virtual machine in Azure, you can assign an identity to your virtual machine that has access to Key Vault.
- **Service principal and certificate**: You can use a service principal and an associated certificate that has access to Key Vault.
- **Service principal and secret**: Although you can use a service principal and a secret to authenticate to Key Vault, we don't recommend it.

Discover Azure Key Vault best practices (2 of 2)

Best practices

- **Use separate key vaults:** Recommended to use a vault per application per environment.
- **Control access to your vault:** Key Vault data is sensitive and business critical.
- **Backup:** Create regular back ups of your vault on update/delete/create of objects within a Vault.
- **Logging:** Be sure to turn on logging and alerts.
- **Recovery options**: Turn on soft-delete and purge protection if you want to guard against force deletion of the secret.

Authenticate to Azure Key Vault (1 of 2)

Two ways to obtain a service principal

- Enable a system-assigned managed identity for the application. deployed to a variety of services.
- If you cannot use managed identity, you instead register the application with your Azure AD tenant.

Note: It is recommended to use a system-assigned managed identity.

Authenticate to Azure Key Vault (2 of 2)

Authentication to Key Vault in application code

Key Vault SDK is using Azure Identity client library, which allows seamless authentication to Key Vault across environments.

.NET	Python	Java	JavaScript
Azure Identity SDK .NET	Azure Identity SDK Python	Azure Identity SDK Java	Azure Identity SDK JavaScript

Authentication to Key Vault with REST

Access tokens must be sent to the service using the HTTP Authorization header:

```
PUT /keys/MYKEY?api-version=<api_version> HTTP/1.1
Authorization: Bearer <access_token>
```

Exercise: Set and retrieve a secret from Azure Key Vault by using Azure CLI

In this exercise you learn how to use Azure CLI to create Azure Key Vault resources and create and retrieve a key.

Objectives

- Create a Key Vault
- Add and retrieve a secret
- Clean up resources

Summary and knowledge check

In this module, you learned how to:

- Describe the benefits of using Azure Key Vault
- Explain how to authenticate to Azure Key Vault
- Set and retrieve a secret from Azure Key Vault by using the Azure CLI

1. What method of authenticating to Azure Key Vault is recommended for most scenarios?

2. Azure Key Vault protects data when it's traveling between Azure Key Vault and clients. What protocol does it use for encryption?

Module 2: Implement managed identities

Learning objectives

- Explain the differences between the two types of managed identities
- Describe the flows for user- and system-assigned managed identities
- Configure managed identities
- Acquire access tokens by using REST and code

Introduction

- A common challenge for developers is the management of secrets and credentials used to secure communication between different components making up a solution.
- Managed identities eliminate the need for developers to manage credentials.

Explore managed identities (1 of 2)

Types of managed identities

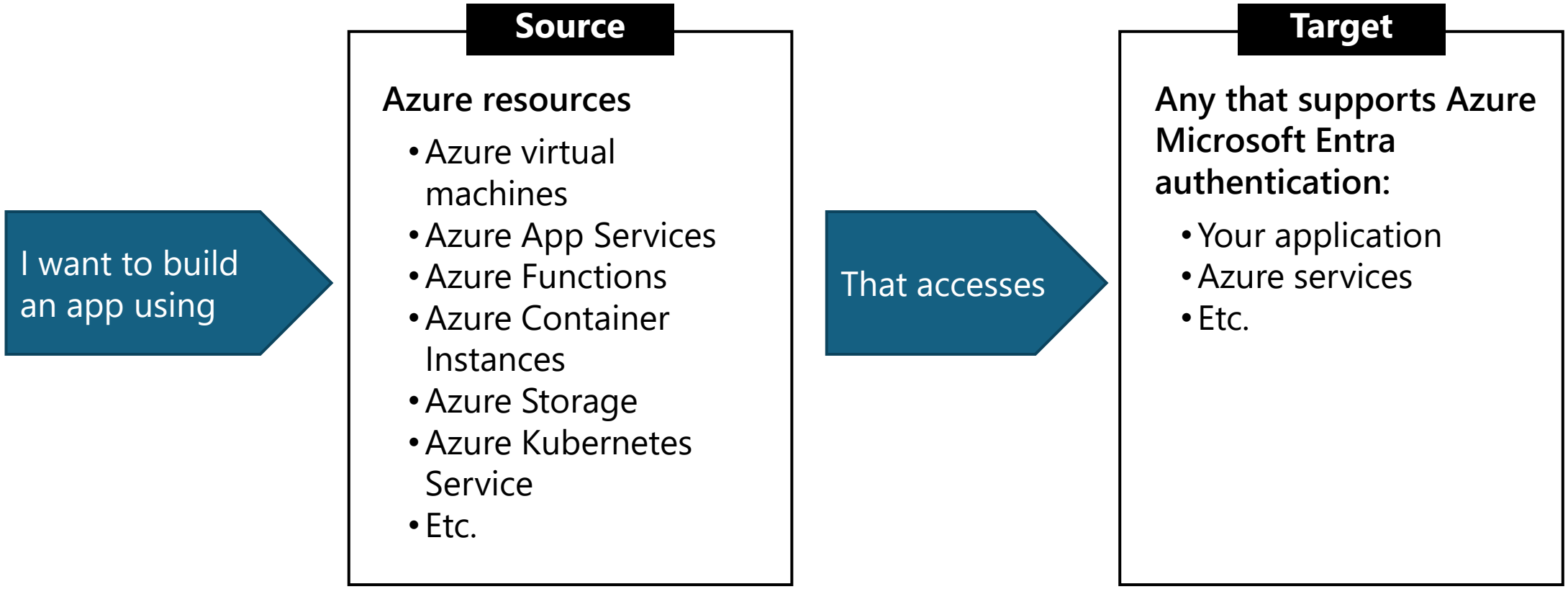
- A system-assigned managed identity is enabled directly on an Azure service instance.
- A user-assigned managed identity is created as a standalone Azure resource.

Characteristics of managed identities

Characteristic	System-assigned managed identity	User-assigned managed identity
Creation	Created as part of an Azure resource (for example, an Azure virtual machine or Azure App Service)	Created as a stand-alone Azure resource
Lifecycle	Shared lifecycle with the Azure resource that the managed identity is created with. When the parent resource is deleted, the managed identity is deleted as well.	Independent life-cycle. Must be explicitly deleted.
Sharing across Azure resources	Cannot be shared, it can only be associated with a single Azure resource.	Can be shared, the same user-assigned managed identity can be associated with more than one Azure resource.

Explore managed identities (2 of 2)

When to use managed identities



Discover the managed identities authentication flow

How a system-assigned managed identity works with an Azure virtual machine:

1. Azure Resource Manager (ARM) receives a request to enable the system-assigned managed identity on a virtual machine.
2. Azure Resource Manager creates a service principal in Microsoft Entra ID for the identity of the virtual machine.
3. ARM configures the identity on the virtual machine by updating the Azure Instance Metadata Service identity endpoint with the service principal client ID and certificate.
4. After the virtual machine has an identity, use the service principal information to grant the virtual machine access to Azure resources.
5. Your code that's running on the virtual machine can request a token from the Azure Instance Metadata service endpoint, accessible only from within the virtual machine.
6. A call is made to Microsoft Entra ID to request an access token by using the client ID and certificate configured. Azure Active Directory returns a JSON Web Token access token.
7. Your code sends the access token on a call to a service that supports Microsoft Entra authentication.

Configure managed identities (1 of 3)

System-assigned managed identity:

- To create, or enable, an Azure virtual machine with the system-assigned managed identity your account needs the Virtual Machine Contributor role assignment.
- No additional Microsoft Entra directory role assignments are required.

User-assigned managed identity:

- **Create the user-assigned identity**: Create a user-assigned managed identity using `az identity create`.
- **Assign the identity to a virtual machine:** Creates a virtual machine associated with the new user-assigned identity.

Configure managed identities (2 of 3)

Enable system-assigned identity during resource creation

```
az vm create --resource-group myResourceGroup \
    --name myVM --image win2016datacenter \
    --generate-ssh-keys \
    --assign-identity \
    --admin-username azureuser \
    --admin-password myPassword12
```

Enable user-assigned identity during resource creation

```
# Create the identity
az identity create -g myResourceGroup \
    -n myUserAssignedIdentity

# Assign identity during creation
az vm create \
    --resource-group <RESOURCE GROUP> \
    --name <VM NAME> \
    --image UbuntuLTS \
    --admin-username azureuser \
    --admin-password myPassword12 \
    --assign-identity <USER ASSIGNED IDENTITY NAME>
```

Configure managed identities (3 of 3)

Azure SDKs with managed identities for Azure resources support:

- Azure supports multiple programming platforms through a series of Azure SDKs.
- Several of them have been updated to support managed identities for Azure resources.

SDK	Sample
.NET	Manage resource from a virtual machine enabled with managed identities for Azure resources enabled
Java	Manage storage from a virtual machine enabled with managed identities for Azure resources
Node.js	Create a virtual machine with system-assigned managed identity enabled
Python	Create a virtual machine with system-assigned managed identity enabled
Ruby	Create Azure virtual machine with an system-assigned identity enabled

Acquire an access token

- A client application can request managed identities for Azure resources app-only access token for accessing a given resource.
- The token is based on the managed identities for Azure resources service principal.
- Recommended to use the `DefaultAzureCredential`

```
// When deployed to an azure host, the default azure credential will authenticate the
specified user assigned managed identity.


string userAssignedClientId = "<your managed identity client Id>";

var credential = new DefaultAzureCredential(new DefaultAzureCredentialOptions {
        ManagedIdentityClientId = userAssignedClientId });

var blobClient = new BlobClient(new Uri("URI"), credential);
```

Summary and knowledge check

In this module, you learned how to:

- Explain the differences between the two types of managed identities
- Describe the flows for user- and system-assigned managed identities
- Configure managed identities
- Acquire access tokens by using REST and code

1 A client app requests managed identities for an access token for a given resource. What is is the basis for the token?

Module 3: Implement Azure App Configuration

Learning objectives

- Explain the benefits of using Azure App Configuration
- Describe how Azure App Configuration stores information
- Implement feature management
- Securely access your app configuration information

Introduction

- Azure App Configuration provides a service to centrally manage application settings and feature flags.
- Programs running in a cloud, generally have many components that are distributed in nature.
- Spreading configuration settings across these components can lead to hard-to-troubleshoot errors during an application deployment.
- Use App Configuration to store all the settings for your application and secure their access in one place.

Explore the Azure App Configuration service (1 of 2)

Application configuration provides benefits:

- A fully managed service that can be set up in minutes
- Flexible key representations and mappings
- Tagging with labels
- Point-in-time replay of settings
- Dedicated UI for feature flag management
- Comparison of two sets of configurations on custom-defined dimensions
- Enhanced security through Azure-managed identities
- Complete data encryptions, at rest or in transit
- Native integration with popular frameworks

App Configuration implementation scenarios:

- Centralize management and distribution of hierarchical configuration data for different environments and geographies
- Dynamically change application settings without the need to redeploy or restart an application
- Control feature availability in real-time

Explore the Azure App Configuration service (2 of 2)

The easiest way to add an App Configuration store to your application is through a client library that Microsoft provides.

Programming language and framework	How to connect
.NET Core and ASP.NET Core	App Configuration provider for .NET Core
.NET Framework and ASP.NET	App Configuration builder for .NET
Java Spring	App Configuration client for Spring Cloud
JavaScript/Node.js	App Configuration client for JavaScript
Python	App Configuration client for Python
Others	App Configuration REST API

Create paired keys and values

Keys

- **Design key namespaces** – There are two general approaches to naming keys used for configuration data: flat or hierarchical.
- **Label keys** – Key values in App Configuration can optionally have a label attribute.
- **Version key values** – App Configuration doesn't version key values automatically as they're modified.
- **Query key values** – Each key value is uniquely identified by its key plus a label that can be null.

Values

- Values assigned to keys are also unicode strings.
- There's an optional user-defined content type associated with each value.
- Configuration data stored in an App Configuration store, which includes all keys and values, is encrypted at rest and in transit.

Manage application features (1 of 2)

Feature management is a modern software-development practice that decouples feature release from code deployment and enables quick changes to feature availability on demand.

Basic concepts	
Feature flag: A feature flag is a variable with a binary state of on or off. **Feature manager:** A feature manager is an application package that handles the lifecycle of all the feature flags in an application. **Filter:** A filter is a rule for evaluating the state of a feature flag.	**Components that implement effective feature management** • An application that makes use of feature flags. • A separate repository that stores the feature flags and their current states.

Manage application features (2 of 2)

Feature flag declaration

- Each feature flag has two parts: a name, and a list of one or more filters that are used to evaluate if a feature's state is on.
- When a feature flag has multiple filters, the filter list is traversed in order until one of the filters determines the feature should be enabled.
- The feature manager supports *appsettings.json* as a configuration source for feature flags.

Feature flag repository

- To use feature flags effectively, you need to externalize all the feature flags used in an application.
- Azure App Configuration is designed to be a centralized repository for feature flags.

Secure app configuration data (1 of 2)

Encrypt configuration data by using customer-managed keys

Enable customer-managed key capability:

- Standard tier Azure App Configuration instance
- Azure Key Vault with soft-delete and purge-protection features enabled
- An RSA or RSA-HSM key within the Key Vault

Allow Azure application configuration to use Key Vault keys:

- Assign a managed identity to the Azure App Configuration instance
- Grant the identity permissions in the target Key Vault's access policy.

Secure app configuration data (2 of 2)

Private endpoints and managed identities

Use private endpoints for Azure App Configuration

- Secure your application configuration details by configuring the firewall to block all connections to App Configuration on the public endpoint.
- Increase security for the virtual network (VNet) ensuring data doesn't escape from the VNet.
- Securely connect to the App Configuration store from on-premises networks that connect to the VNet using VPN or ExpressRoutes with private-peering.

Managed identities

- A system-assigned identity is tied to your configuration store. It's deleted if your configuration store is deleted. A configuration store can only have one system-assigned identity.
- A user-assigned identity is a standalone Azure resource that can be assigned to your configuration store. A configuration store can have multiple user-assigned identities.

Summary and knowledge check

In this module, you learned how to:

- Explain the benefits of using Azure App Configuration
- Describe how Azure App Configuration stores information
- Implement feature management
- Securely access your app configuration information

1. Which type of encryption does Azure App Configuration use to encrypt data at rest?

2. Which option evaluates the state of a feature flag?

Discussion and lab

Group discussion questions

- Contoso Inc has a web app where each developer has contributor access. What can be done to prevent developers from accessing application secrets?
- Managed identities are generally recommended to handle authentication between Azure resources in a solution. In what situations would you want to use a different authentication solution?
- Contoso Inc wants to perform A/B testing on a new feature for their app. What service(s) should they use and what kind of changes do they need to make to their code?

Lab 07: Access resource secrets more securely across services

In this lab, you will create a storage account and an Azure Function app that will access the storage account. To demonstrate the secure storage of connection string information, you will provision a Key Vault resource and manage the appropriate secrets to store the connection string information. You will also manage the service identity to gain secure access to the connection string information for the storage account.

http://aka.ms/az204labs

- Exercise 1: Create Azure resources
- Exercise 2: Configure secrets and identities
- Exercise 3: Build an Azure Functions app
- Exercise 4: Access Azure Blob Storage data

End of presentation