

Module 07: Implement secure cloud solutions



Topics

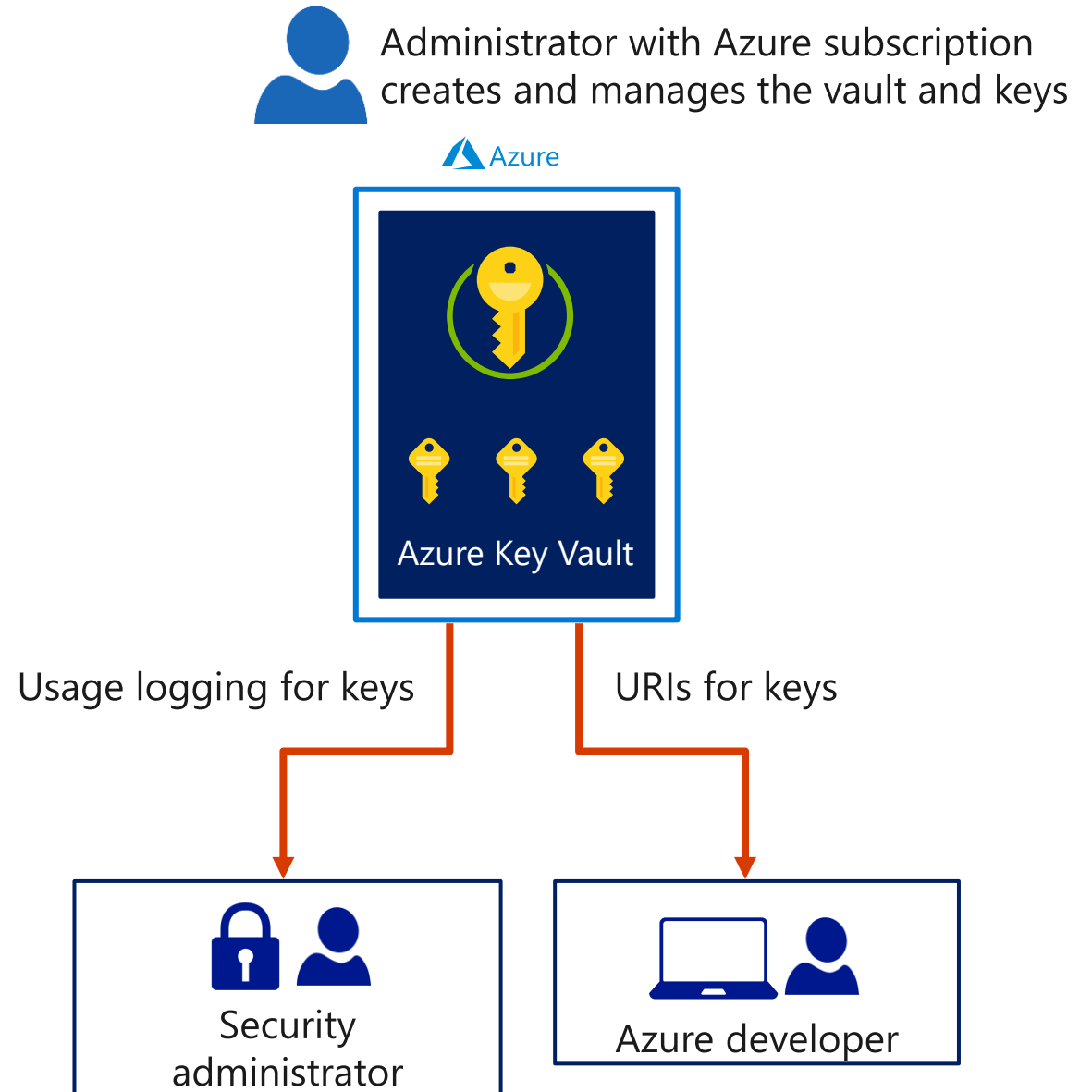
- Manage keys, secrets, and certificates by using the KeyVault API
- Implement Managed Identities for Azure resources
- Secure app configuration data by using Azure App Configuration

Lesson 01: Manage keys, secrets, and certificates by using the KeyVault API



Azure Key Vault

- Safeguard cryptographic keys and other secrets that cloud apps and services use:
 - Increase security and control over keys and passwords
 - Applications have no direct access to keys
 - Use FIPS 140-2 Level 2 validated hardware security modules (HSMs)
 - Create and import:
 - Encryption keys
 - API keys
 - Secrets
 - Passwords
 - SSL/TLS certificates



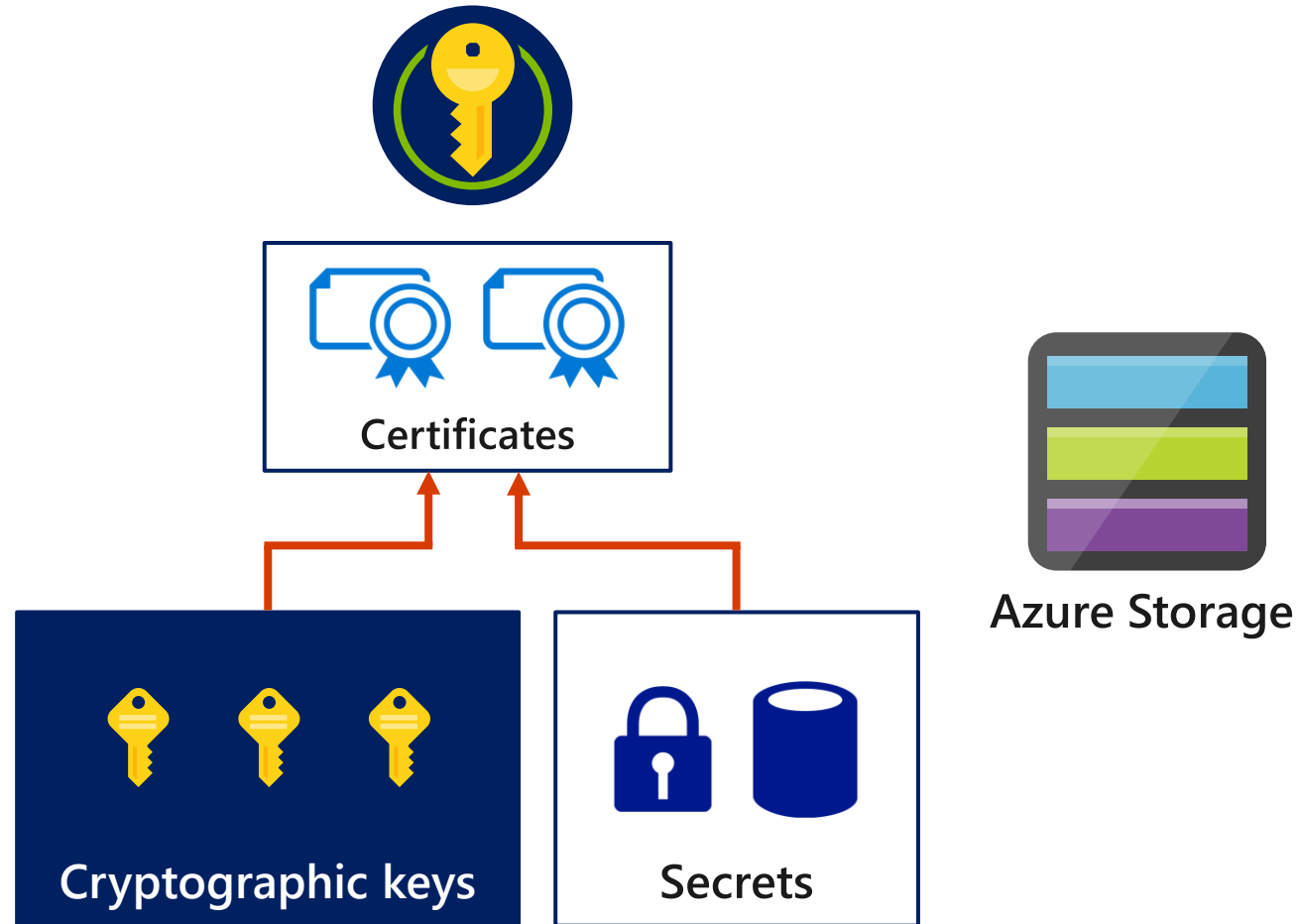
Key Vault concepts

- Vault:
 - Logical group of secrets
- Vault owner:
 - Identity that has full control over the vault
 - Can grant other identities consumer (scoped) access to the vault
- Vault consumer:
 - Identity that can perform actions on the assets inside the vault (with permission)

Key Vault authentication

- Managed identity:
 - Assigned identity for an Azure resource
 - Fastest way to access the vault from a service without sharing or exposing credentials
- Service principal:
 - Can provide certificate or secret
 - Not recommended as it's difficult to rotate

Key Vault secret types



Create Key Vault secret by using Azure CLI

Create resource group

```
az group create --name SecurityGroup --location westus
```

Create Key Vault resource

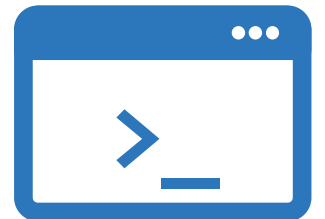
```
az keyvault create --name contosovault --resource-group SecurityGroup --location westus
```

Set secret in Key Vault

```
az keyvault secret set --vault-name contosovault --name DatabasePassword --value  
'Pa55w.rd'
```

Show value of secret in Key Vault

```
az keyvault secret show --vault-name contosovault --name DatabasePassword
```



Demonstration: Setting and retrieving a secret from Azure Key Vault by using Azure CLI



Get Key Vault secret by using C#

```
string secretUri = "https://contoso-vault2.vault.azure.net/secrets/example/0932840309";  
var securityToken = "...";  
  
// Create Key Vault client  
var client = new KeyVaultClient(  
    new KeyVaultClient.AuthenticationCallback(securityToken)  
);  
  
// Get secret  
var secretBundle = await client.GetSecretAsync(secretUri);  
  
// Get value of secret  
var secret = secretBundle.Value;
```

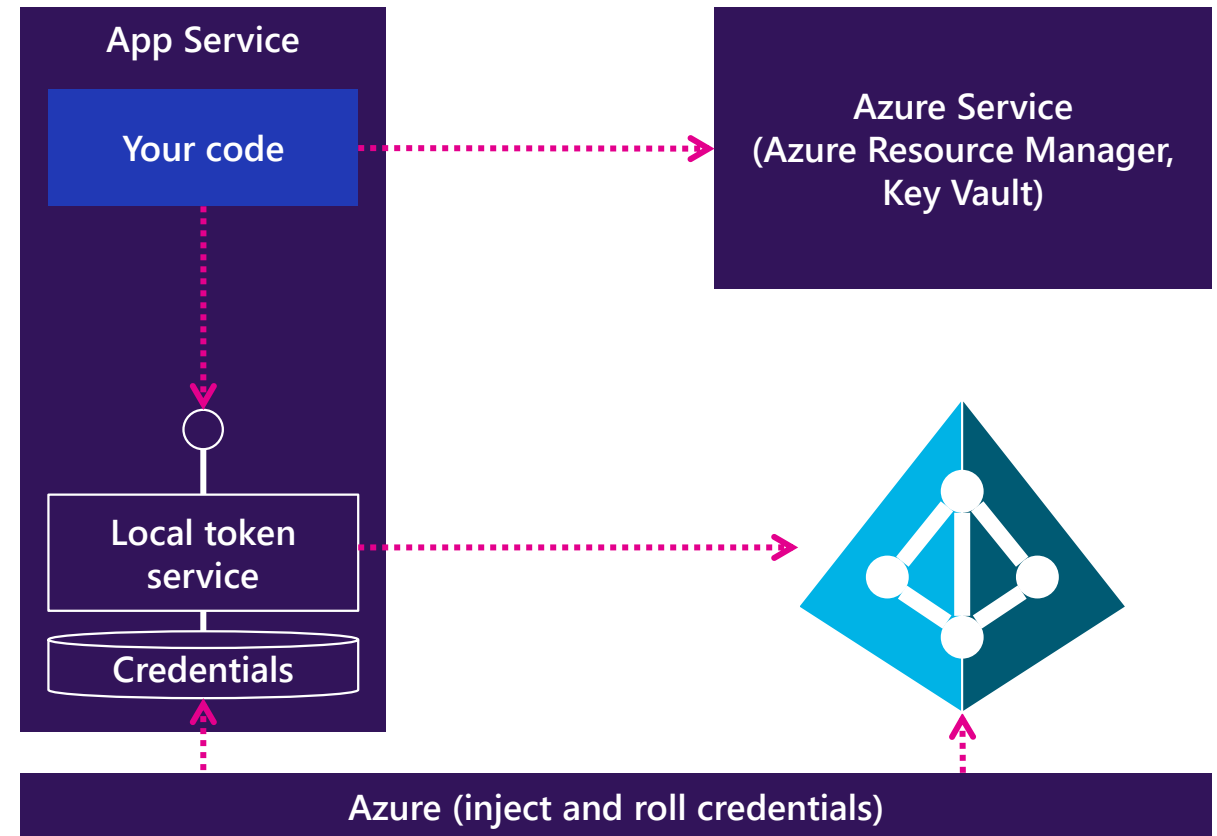


Lesson 02: Implement Managed Identities for Azure resources

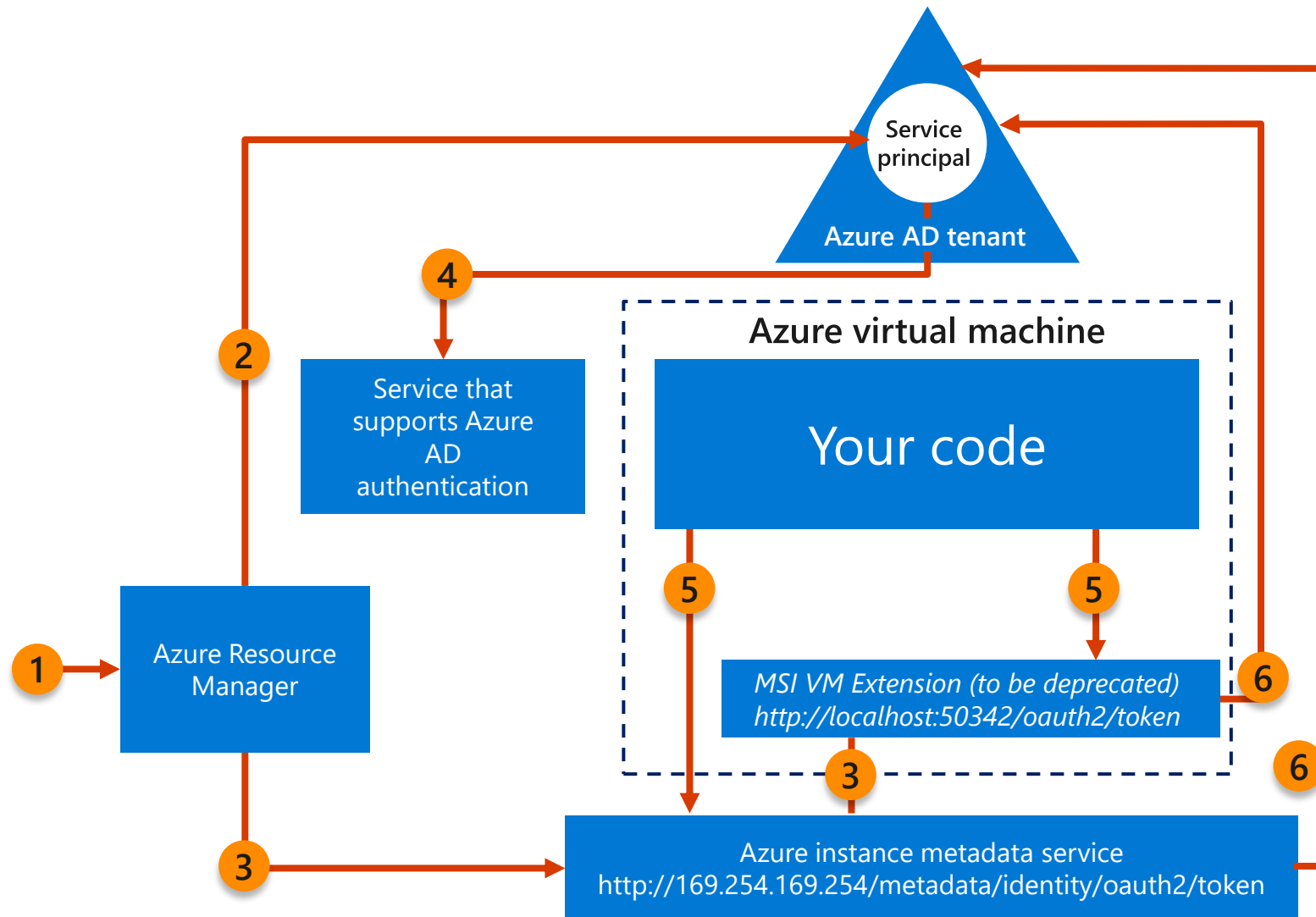


Azure AD–managed identity

- Keeps credentials out of code
- Identity automatically managed in Azure AD for Azure resources
- Uses a local MSI endpoint to get access tokens from Azure AD
- Direct authentication with services or retrieve credentials from Azure Key Vault

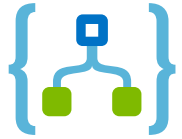


Managed identities implementation



System-assigned managed identity

Shares a lifecycle with the associated Azure resource



Logic Apps



Data Factory



Container Instances



API Management



Container Registries



Managed identity



Managed identity



Managed identity



Managed identity



Managed identity

User-assigned managed identity

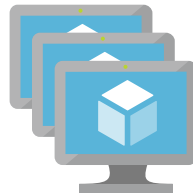
Can be shared among multiple Azure resources



Blueprints



Virtual Machines



Virtual Machine Scale Sets



Functions



App Service



Managed identity



Types of managed identities

	System-assigned	User-assigned
Creation	<ul style="list-style-type: none">Created as part of an Azure resource (for example, an Azure virtual machine or Azure App Service)	<ul style="list-style-type: none">Created as a standalone Azure resource
Lifecycle	<ul style="list-style-type: none">Shares a lifecycle with the assigned Azure resourceWhen the parent resource is deleted, the managed identity is deleted as well	<ul style="list-style-type: none">Has an independent lifecycleMust be explicitly deleted
Sharing across Azure resources	<ul style="list-style-type: none">Cannot be sharedCan only be associated with a single Azure resource	<ul style="list-style-type: none">Can be sharedThe same user-assigned managed identity can be associated with more than one Azure resource

Managed identities use cases

- **System-assigned managed identity:**

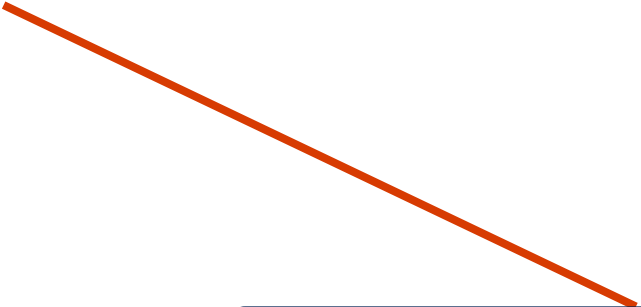
- Workloads that are contained within a single Azure resource
- Workloads for which you need independent identities

- **User-assigned managed identity:**

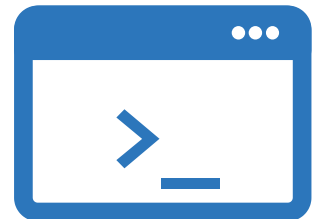
- Workloads that run on multiple resources and which can share a single identity
- Workloads that need preauthorization to a secure resource as part of a provisioning flow
- Workloads where resources are recycled frequently, but permissions should stay consistent

Configure managed identities for Web Apps by using Azure CLI

```
az web app identity assign --name <app-name> --resource-group <resource-group>
```



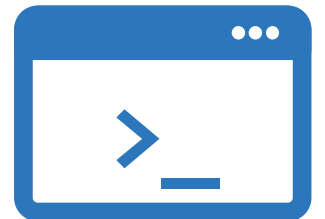
Enable system-assigned
managed identity



Configure managed identities for Azure Container Instances by using Azure CLI

```
az container create --resource-group <resource-group> --name <container-name> --  
image microsoft/azure-cli --assign-identity --command-line "tail -f /dev/null"
```

Create container with system-
assigned managed identity



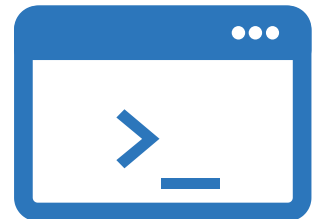
Create user-assigned managed identity by using Azure CLI

```
az identity create --resource-group <resource-group> --name <identity-name>
```

Create user-assigned managed identity

```
resourceID=$(az identity show --resource-group <resource-group> --name <identity-name> --query id --output tsv)
```

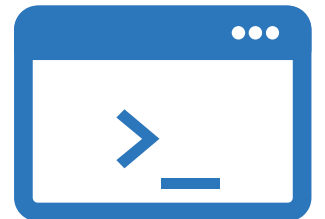
Get unique resource identifier



Assign managed identity by using Azure CLI

```
az container create --resource-group <resource-group> `  
  --name <container-name> --assign-identity $resourceID `  
  --image microsoft/azure-cli
```

Create container with user-
assigned managed identity



Lesson 03: Secure app configuration data by using Azure App Configuration



Azure App Configuration

- Service to centrally manage application settings and feature flags
- Dedicated UI for feature flag management
- Enhanced security through Azure-managed identities
- Complete data encryptions at rest or in transit

Programming languages 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
Others	App Configuration REST API

Keys and values

- Keys serve as the name for key-value pairs
- Keys stored in App Configuration are case-sensitive, unicode-based strings
- Up to 10,000 Unicode characters allowed

Scenario	Examples
Based on component services	AppName:Service1:ApiEndpoint AppName:Service2:ApiEndpoint
Based on deployment regions	AppName:Region1:DbEndpoint AppName:Region2:DbEndpoint

Labels

- Labels are used to differentiate key values with the same key
- Can be used to specify multiple environments

```
Key = AppName:DbEndpoint & Label = Test  
Key = AppName:DbEndpoint & Label = Staging  
Key = AppName:DbEndpoint & Label = Production
```

- Or different versions

```
Key = AppName:DbEndpoint & Label = 1  
Key = AppName:DbEndpoint & Label = 2  
Key = AppName:DbEndpoint & Label = 3
```

Feature Management

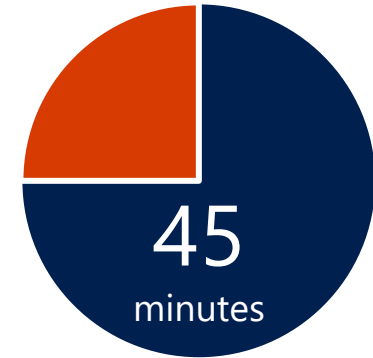
- A feature flag is a variable with a binary state of *on* or *off*
- Feature flags have an associated code block
- The state of the feature flag triggers whether the code block runs or not
- Features can be filtered based on:
 - Simple true/false flag
 - Percentages of requests (sticky sessions)
 - Timeframes (valid from/until)
 - Custom filters

**Lab: Access resource
secrets more
securely across
services**



Lab: Access resource secrets more securely across services

Duration



Lab sign-in information



