Summary

3 minutes

In this module, you secured an app's secret configuration in Azure Key Vault. Our app code authenticates to the vault with a managed identity and automatically loads the secrets from the vault into memory at startup.

Clean up

The sandbox automatically cleans up your resources when you're finished with this module.

When you're working in your own subscription, it's a good idea at the end of a project to identify whether you still need the resources you created. Resources left running can cost you money. You can delete resources individually or delete the resource group to delete the entire set of resources.

To cleanup your Cloud Shell storage, delete the `KeyVaultDemoApp` directory.

Next steps

If this was a real app, what would come next?

- Put all your app secrets in your vaults! There's no longer any reason to have them in configuration files.
- Continue to develop the application. Your production environment is all set up, so for future deployments to it you don't need to repeat all the setup.
- To support development, create a development-environment vault that contains secrets with the same names but different values. Grant permissions to the development team and configure the vault name in the app's development-environment configuration file. Configuration depends on your implementation: for ASP.NET Core, `AddAzureKeyVault` will automatically detect local installations of Visual Studio and the Azure CLI and use Azure credentials configured in those applications to sign in and access the vault. For Node.js, you can create a development-environment service principal with permissions to the vault and have the app authenticate using `loginWithServicePrincipalSecret`.
- Create additional environments for purposes like user acceptance testing.
- Separate vaults across different subscriptions and/or resource groups to isolate them.
- Grant access to other environment vaults to the appropriate people.

Further reading

- Key Vault documentation
- More about AddAzureKeyVault and its advanced options
- This tutorial walks through using `KeyVaultClient`, including manually authenticating it to Azure Active Directory using a client secret instead of using a managed identity.
- Managed identities for Azure resources token service documentation, for implementing the authentication workflow yourself.

Check your knowledge

1. Which of the following is **not** a benefit of Azure Key Vault?

- (◉) Secure storage of private user information. ✓

  Key Vault is intended for storing application secrets, not user secrets.

- ( ) Synchronizing application secrets among multiple instances of an application.
- ( ) Reducing the need for application developers to directly handle application secrets.
- ( ) Controlling access to application secrets with assignable permissions.

2. Which of these statements best describes Azure Key Vault's authentication and authorization process?

- ( ) Applications authenticate to a vault with the username and password of the lead developer and have full access to all secrets in the vault.
- ( ) Applications and users authenticate to a vault with a Microsoft account and are authorized to access specific secrets.
- (◉) Applications and users authenticate to a vault with their Azure Active Directory identities and are authorized to perform actions on all secrets in the vault. ✓

  Authentication to Key Vault uses Azure Active Directory identities. Access policies are used to provide authorization for actions that apply to every secret in the vault.

- ( ) Applications authenticate to a vault with the username and password of a user that signs in to the web app, and is granted access to secrets owned by that user.

3. How does Azure Key Vault help protect your secrets after they have been loaded by your app?

- ○ Azure Key Vault automatically generates a new secret after every use.
- ○ The Azure Key Vault client library protects regions of memory used by your application to prevent accidental secret exposure.
- ○ Azure Key Vault double-encrypts secrets, requiring your app to decrypt them locally every time they're used.
- ◉ It doesn't protect your secrets. Secrets are unprotected once they're loaded by your application. ✓

Once secrets have been loaded by an app, they are unprotected. Make sure to not log them, store them, or return them in client responses.

Module complete:

Unlock achievement