Securing Your Credentials for Azure App Development

Presented by Sarah Dutkiewicz
Microsoft MVP, Developer Technologies
Cleveland Tech Consulting, LLC

Thank You to the Code PaLOUsa Sponsors

Progress

WAYSTAR

Bastian SOLUTIONS
a TOYOTA ADVANCED LOGISTICS company

<code> LOUISVILLE

eblu SOLUTIONS

twilio

Scout

aws

cbts

<prosoft>

CGI

Friends of Code PaLOUsa

Asperitas

Couchbase

DATASTAX

mongoDB

redis

ROCKET Mortgage

Agenda

- Problems with Credentials in Azure Development
- Alternatives
- Introducing DefaultAzureCredential
- Demos, demos, and more demos

Where do you store your passwords?

- Config files
  - App config – like here? Deploying passwords and other sensitive data to ASP.NET and Azure App Service - ASP.NET 4.x | Microsoft Docs
  - Secrets config
  - **Add these to .gitignore!**
- Environment Variables
  - How are you loading the values into your environment?
- Secret Manager
  - For development purposes only, not encrypted
- Azure Key Vault

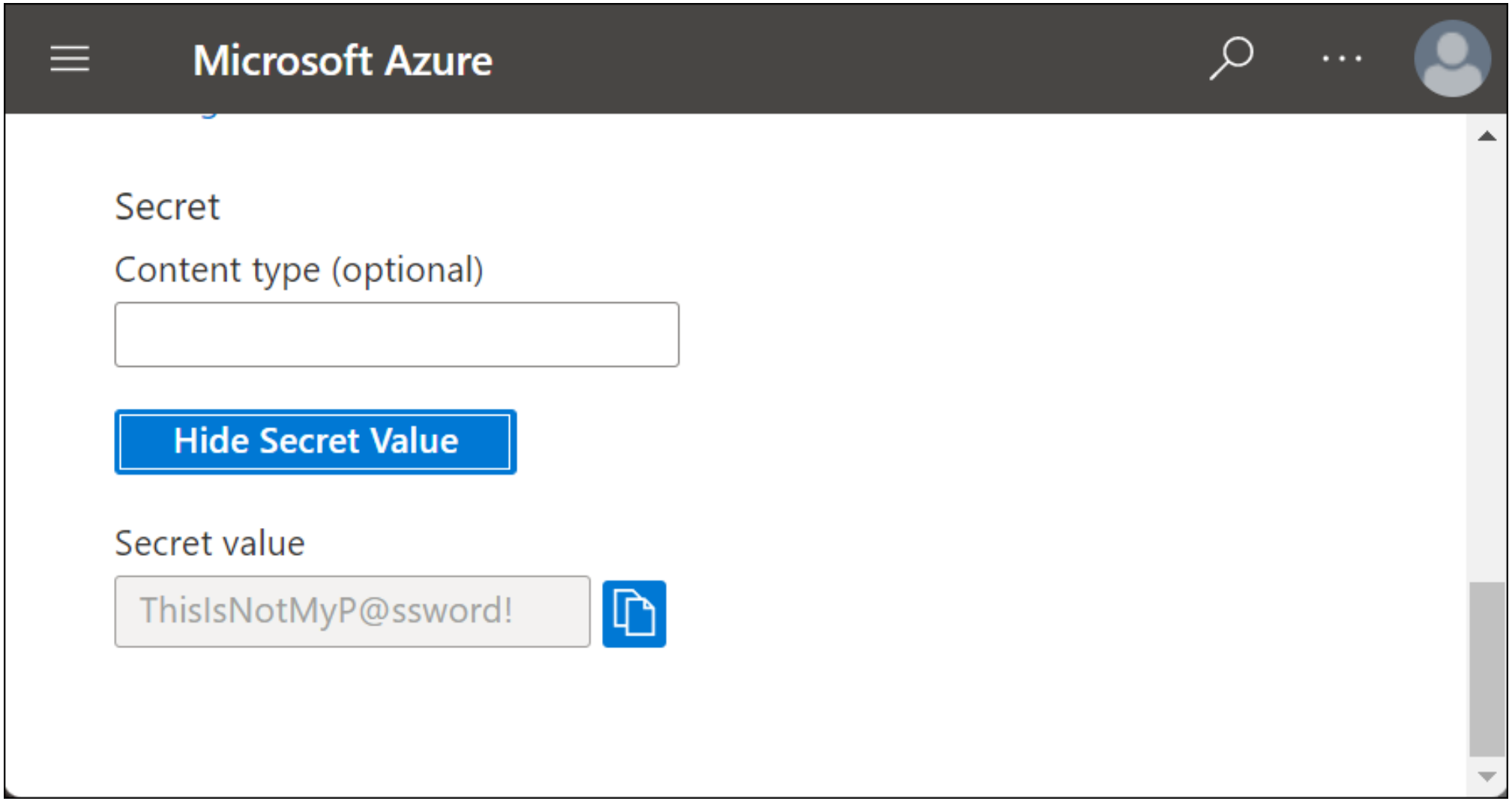
```
<appSettings>
  <!-- SendGrid-->
  <add key="mailAccount" value="My mail account." />
  <add key="mailPassword" value="My mail password." />
  <!-- Twilio-->
  <add key="TwilioSid" value="My Twilio SID." />
  <add key="TwilioToken" value="My Twilio Token." />
  <add key="TwilioFromPhone" value="+12065551234" />

  <add key="GoogClientID"
value="1.apps.googleusercontent.com" />
  <add key="GoogClientSecret" value="My Google client
secret." />
</appSettings>
```

The Problems with Config Files

- Typically in plain text
- Sometimes stored encrypted… with the code to decrypt it 🤦
- Pushed into a public repo or a private repo in the wrong hands = more trouble

This Photo by Unknown Author is licensed under CC BY-NC-ND

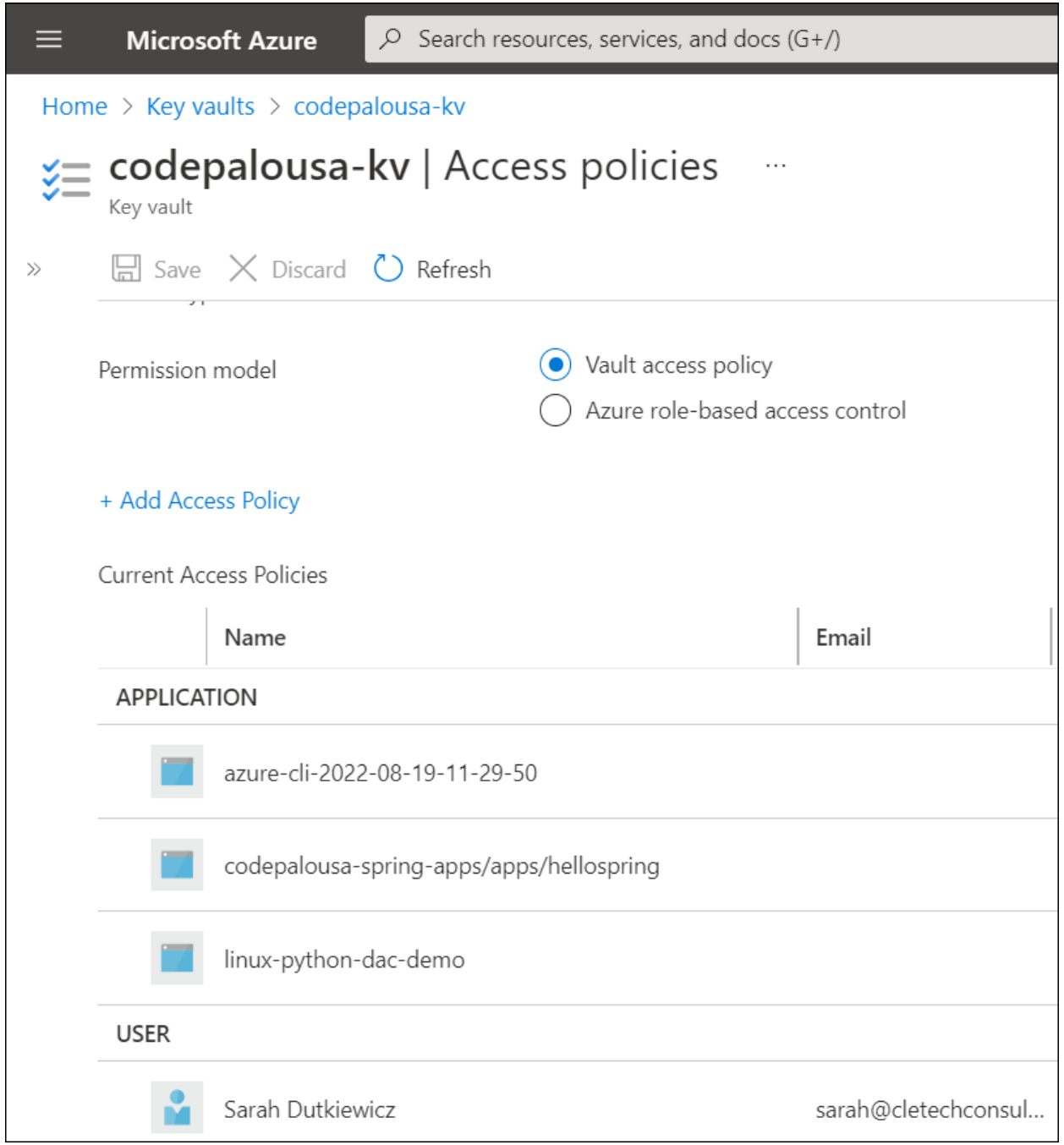
Azure Key Vault - SecretPassword



Azure Key Vault – Access Policies

- RBAC users
- Managed identities
- Service principals

- Can add and revoke permissions



Alternatives to user credentials in Azure

- Managed identities
- X.509 certificates
- Service principals

Managed identities

- A way to authorize access to Azure resources between Azure resources that support Azure AD authentication
- No credentials needed, no tokens to manage
- Can be used in RBAC assignment
- No extra cost!

Two types of managed identities

- System-assigned:
  - Can be enabled on service and assigned by the system
  - Creates identity in Azure AD
  - Automatically deleted when resource is deleted
  - Limited to just that resource
- User-assigned:
  - You create it in Azure AD
  - Assign to one or more instances of Azure service
  - Manual management – does not automatically delete when no longer used

Services that use managed identities

- Not all platforms support managed identities yet
- Some platforms support managed identities in limited cases
- Some platforms have better managed identity support
  - Azure Key Vault
  - Azure App Service
  - Azure Kubernetes Service
  - Azure Data Factory
- Complete, updated list is available at: https://docs.microsoft.com/azure/active-directory/managed-identities-azure-resources/managed-identities-status

X.509 certificates for authentication

- Azure AD certificate based authentication (CBA)
- Currently in public preview (August 2022)
- Recommended as a security best practice – no password secrets
- X.509 certificates verified against Enterprise Public Key Infrastructure (PKI)
- Requires setup in Azure AD
  - Certificate authority
  - Authentication binding policy
  - Username binding policy
  - Enable CBA
- No additional cost!
  - Available in every edition of Azure AD
- Learn more: How to configure Azure AD certificate-based authentication without federation (Preview) - Azure Active Directory - Microsoft Entra | Microsoft Docs

Service principals

- A way to identify an application or user registered in Azure AD
  - Created as part of app registration
- Three types
  - Application
  - Managed identity
  - Legacy – created before app registrations were added to Azure AD
    - Only used in the tenant where created
    - Credentials, service principal names, reply URLs, etc. not in app registration
- Can be used in RBAC assignments
- Learn more: Apps & service principals in Azure AD - Microsoft Entra | Microsoft Docs

Service principal notes

- Originated more similar to Windows Server Active Directory
- Not all service principals have application objects

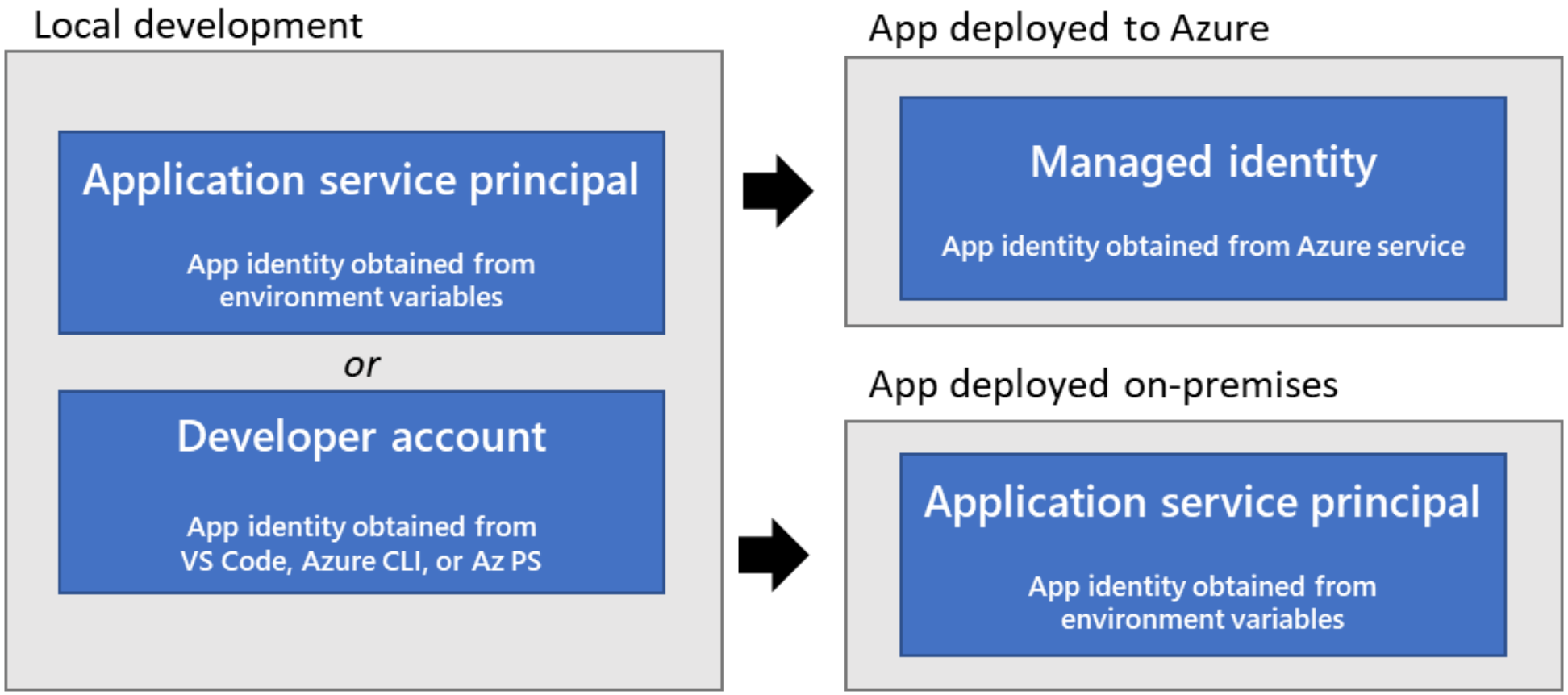
- Uses the command:

az ad sp create-for-rbac –role {ROLE} –scopes /subscriptions/{SUBSCRIPTION_ID}

Output from service principal creation

```
{
  "appId": "AZURE_CLIENT_ID",
  "displayName": "{SERVICE_PRINCIPAL_NAME}",
  "password": "AZURE_CLIENT_SECRET",
  "tenant": "AZURE_TENANT_ID"
}
```

Recommendations on what to choose



Introducing DefaultAzureCredential

DefaultAzureCredential

- Part of Azure Identity
- Available in many of the Azure SDKs including:
  - Java
  - .NET
  - JavaScript
  - Python
- Makes it easier for working with Azure resources and credentials without storing the credentials in config files
- Uses a chain of credentials for authentication

DefaultAzureCredential Order of Checking

1. EnvironmentCredential – environment variables
2. ManagedIdentityCredential – managed identity
3. SharedTokenCacheCredential – local shared token cache from some versions of Visual Studio
4. VisualStudioCredential – auth from Visual Studio
5. VisualStudioCodeCredential – auth from Visual Studio Code
6. AzureCliCredential – auth from Azure CLI
7. AzurePowerShellCredential- auth from Azure PowerShell
8. InteractiveBrowserCredential – auth with a browser prompt; not enabled by default

Failure error with chain

- azure.core.exceptions.ClientAuthenticationError: DefaultAzureCredential failed to retrieve a token from the included credentials.
- Attempted credentials:
- EnvironmentCredential: EnvironmentCredential authentication unavailable. Environment variables are not fully configured.
- Visit https://aka.ms/azsdk/python/identity/environmentcredential/troubleshoot to troubleshoot.this issue.
- ManagedIdentityCredential: ManagedIdentityCredential authentication unavailable, no response from the IMDS endpoint.
- SharedTokenCacheCredential: Shared token cache unavailable
- VisualStudioCodeCredential: Azure Active Directory error '(invalid_grant) AADSTS700082: The refresh token has expired due to inactivity. The token w"ef2cbcdb-b45b-411a-82e8-4da4de87e200","correlation_id":"a5787e3d-7be9-40f9-b6ec-02f5c2d90ae2","error_uri":"https://login.microsoftonline.com/error?code=700082"}
- To mitigate this issue, please refer to the troubleshooting guidelines here at https://aka.ms/azsdk/python/identity/defaultazurecredential/troubleshoot

Notes on the Credential Chain

- Interactive browser option needs to be included explicitly
- Credential types can be excluded in `DefaultAzureCredentialOptions`
  - For example, PowerShell vs Azure CLI preference
- Managed identity is assumed as system-assigned
  - Explicitly set client ID for user-assigned identities in `DefaultAzureCredentialOptions`
- Tenant ID can be overridden in the `DefaultAzureCredentialOptions` object
- By default, targets public cloud. Can change the **authority host** as part of `DefaultAzureCredentialOptions`

Environment Credential

- Works well with service principals and general credentials
- Looks for environment variables with specific names:
  - AZURE_CLIENT_ID
  - AZURE_CLIENT_SECRET
  - AZURE_TENANT_ID
  - AZURE_CLIENT_CERTIFICATE_PATH
  - AZURE_CLIENT_SEND_CERTIFICATE_CHAIN
  - AZURE_USERNAME
  - AZURE_PASSWORD
  - AZURE_AUTHORITY_HOST

Managed Identities

- Used between Azure resources
- Managed in Azure Active Directory
- System-assigned and user-assigned
- No need to store any credentials
- You can’t even access the credentials
- No extra cost

Shared Token Cache

- Stored in an in-memory cache shared between Microsoft applications
- Older versions of Visual Studio might have used this
- Not well documented – not really sure what else may use this

IDE Credentials

- Visual Studio
- Visual Studio Code

Command-line Credentials

- PowerShell
- Azure CLI

Interactive Browser

- Must be enabled when constructing the `DefaultAzureCredential` object
  - Set `includeInteractiveCredentials` to true
  - Set `ExcludeInteractiveBrowserCredential` to false in `DefaultAzureCredentialOptions`

Demos: Java

Demos

- Azure App Service
- Azure Spring Apps

Resources

- Azure Authentication in Java development environments | Microsoft Docs
- Quickstart - Azure Key Vault Secret client library for Java | Microsoft Docs

Python

- Quickstart – Azure Key Vault Python client library – manage secrets | Microsoft Docs
- Azure Identity client library for Python | Microsoft Docs
- Troubleshoot DefaultAzureCredential Authentication Issues (Python)

C#

- Azure Identity client library for .NET - Azure for .NET Developers | Microsoft Docs
- Quickstart - Azure Key Vault keys client library for .NET (SDK version 4) | Microsoft Docs

Tools to assist with monitoring for creds in code

- Microsoft Security Code Analysis
- GitHub secret scanning
- GitLab Secret Detection
- Bitbucket secret scanning
- Azure DevOps
  - Gitleaks extension
  - SARIF SAST Scans Tab
    - SARIF = Static Analysis Results Interchange Format

Resources

- Authentication and the Azure SDK - Azure SDK Blog (microsoft.com)
- DefaultAzureCredential Class
  - Java
  - .NET
  - JavaScript
  - Python
- Azure Key Vault developer's guide | Microsoft Docs

Any Questions?

Twitter: @sadukie
LinkedIn:
https://linkedin.com/in/sadukie