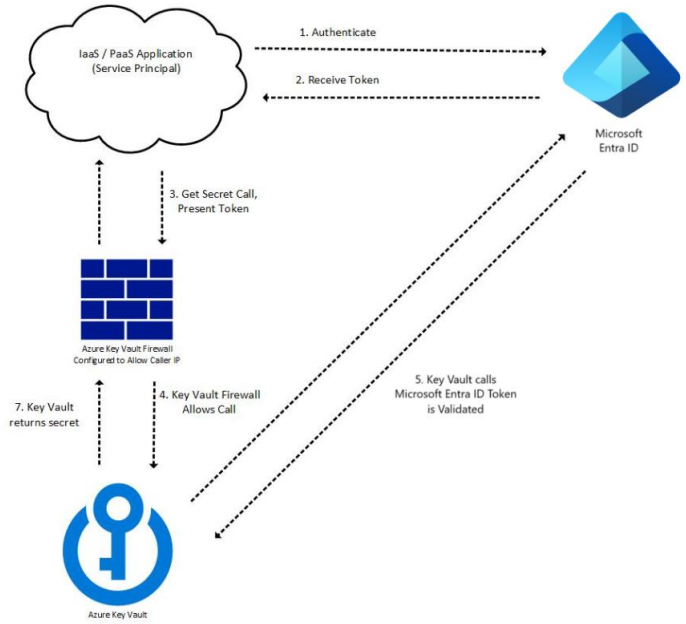
Azure Key Vault & Azure OpenAI: Building Secure AI Workflows in .NET

Referent: Dr. Matthias Liebeck



Azure Cloud Native
KÖLN

LinkedIn

GitHub Copilot
Newsletter

Begriffe

- Secrets = Geheimnisse
- Key = Schlüssel
- Certificate = Zertifikat

Sorry für Denglisch-Mix!

Aufbau des heutigen Vortrags

Einleitung (Wer ist Matthias?)

Passwords + Secrets and und wie wir sie speichern können

Grundzulagen zu Azure Key Vault

Was ist Azure Key Vault?

Warum sollte man ihn verwenden?

Demo: Refactoring einer .NET Anwendung mit Azure OpenAI

Azure Key Vault Best Practices

Über Dr. Matthias Liebeck

- Professioneller Softwareentwickler seit 2009, C# seit 2006
- 2009 – 2015: Studium der Informatik und Mathematik
- 2018: Promotion in Informatik / Künstlicher Intelligenz mit Schwerpunkt Natural Language Processing
- Seit 2019: Tätigkeit bei der anyMOTION GmbH in Düsseldorf / Deutschland als Senior Solution Architect mit den Schwerpunkten
  - ASP.NET Core MVC / Web API / C#
  - SQL Server
  - Azure
- Veranstalter Azure Düsseldorf Meetup

Azure Meetup Düsseldorf

Passwords + Secrets und wie wir sie speichern können

Warum benötigen Anwendungen passwords and secrets?

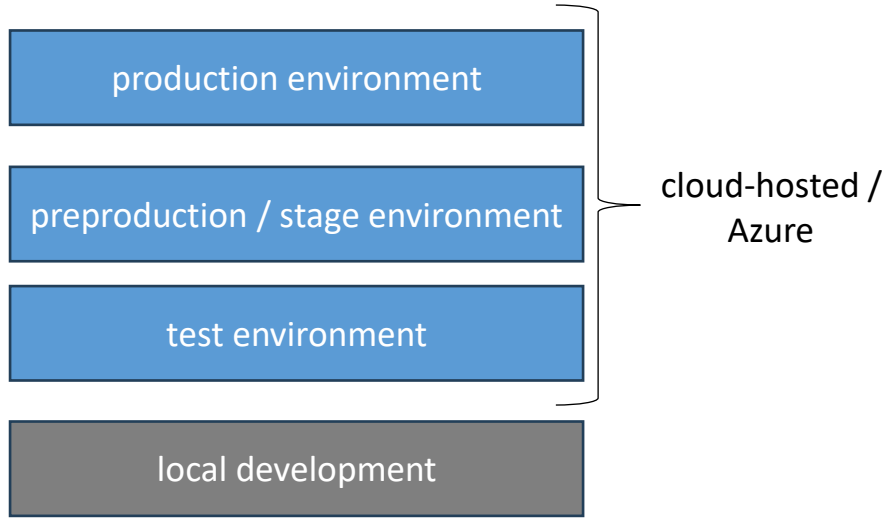
Fast alle realen Softwareanwendungen greifen auf Daten oder Systeme zu

- Datenbanken (connection string / Benutzername, Passwort + Hostname)
  Zweck: Lesen von oder Schreiben von Inhalten in die Datenbank
- Cloud Storage (z. B. Azure Blob Storage, connection string oder SAS-Token)
  Zweck: Sicheres Abrufen von Dateien von Cloud-Speicherdiensten
- APIs und Webdienste
  Zweck: z.B. Facebook API, Discord API, Google Maps API, ...
- SMTP-Server (Hostname, Benutzername + Passwort)
  Zweck: Versand von E-Mails
- Push-Benachrichtigungen (z. B. Firebase, Firebase JSON-Profil)
  Zweck: Senden von Push-Nachrichten an Apps
- Lizenzschlüssel für third party component libraries
  Zweck: ownership authentication

Software Deployment process:

- Der Quellcode wird entwickelt
- …
- Der Quellcode wird deployed
- Es gibt verschiedene Arten secrets / passwords in hosted environments zu hinterlegen, wobei jede environment unterschiedliche Passwörter hat (z.B. unterschiedliche connection strings pro Datenbank)

- Fokus im Vortrag: Passwörter & Secrets für die lokale Entwicklung im verteilten Team => Die Passwörter müssen zur Entwicklungszeit da sein



Wo können wir Passwörter für die lokale Entwicklung speichern? (Visual Studio-Perspektive)?

- Im Code, wo das Passwort benötigt wird

```
var client = new HttpClient();
var authToken :string = Convert.ToBase64String( inArray: Encoding.ASCII.GetBytes( s: $"userName:MySecretPassword"));
client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue( scheme: "Basic", parameter: authToken);
var response = await client.GetAsync( requestUri: "https://www.google.de");
```

  → tracked by source control / Git.

- In einer Konfigurationsdatei, z.B. appsettings.json, tracked by source control

```
"ConnectionStrings": {
  "DefaultConnection": "Server=xxx.database.windows.net;Database=XXX;persist security info=True;user id=sql_admin;password='MyPassword';multipleactiveresultsets=True;"
},
"Settings": {
  "MailSMTP": "smtp.strato.de",
  "MailPort": "587",
  "MailUser": "xxx@mydomain.de",
  "MailPassword": "abcdefg123456"
},
```

- secrets.json → nicht source-code tracked, aber mühevoll im großen Team zu sycen & keine Verlauf / Historie
- environment variables → gleiche Problematik wie secrets.json
- Azure Key Vault 🎉

Keine hard-coded Passwörter verwenden

Keine hard-coded Passwörter nutzen, insbesondere nicht mit source-control

```
var client = new HttpClient();
var authToken :string = Convert.ToBase64String( inArray: Encoding.ASCII.GetBytes( s: $"userName:MySecretPassword"));
client.DefaultRequestHeaders.Authorization = new AuthenticationHeaderValue( scheme: "Basic",  parameter: authToken);
var response = await client.GetAsync( requestUri: "https://www.google.de");
```

Warum nicht?

- Secrets können in öffentlichen Repositories exposed werden
- Password-Rotation benötigt Deployments
- Passwörter sind Teil der Git history und müssen per force push entfernt werden
- versehentliches Teilen ist möglich, z.B. per Screensharing oder Live-Streaming

Datenlecks können teuer werden!

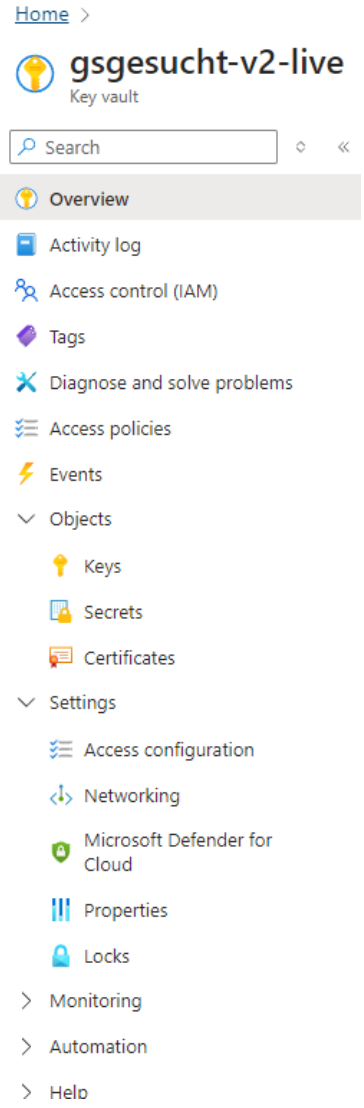
- Im Durchschnitt exposen 3 von 1000 GitHub commits mindestens ein secret. Über 3 Millionen secrets wurden 2022 in GitHub repositories gefunden. [1]
- 2014 Uber: AWS S3 access key in einem public repository gefunden.
  Das Datenleck wurde ursprünglich geheimgehalten → **$ 148 Millionen Dollar Strafe**

[1] https://blog.gitguardian.com/the-state-of-secrets-sprawl-2022/
[2] https://www.breaches.cloud/incidents/uber/

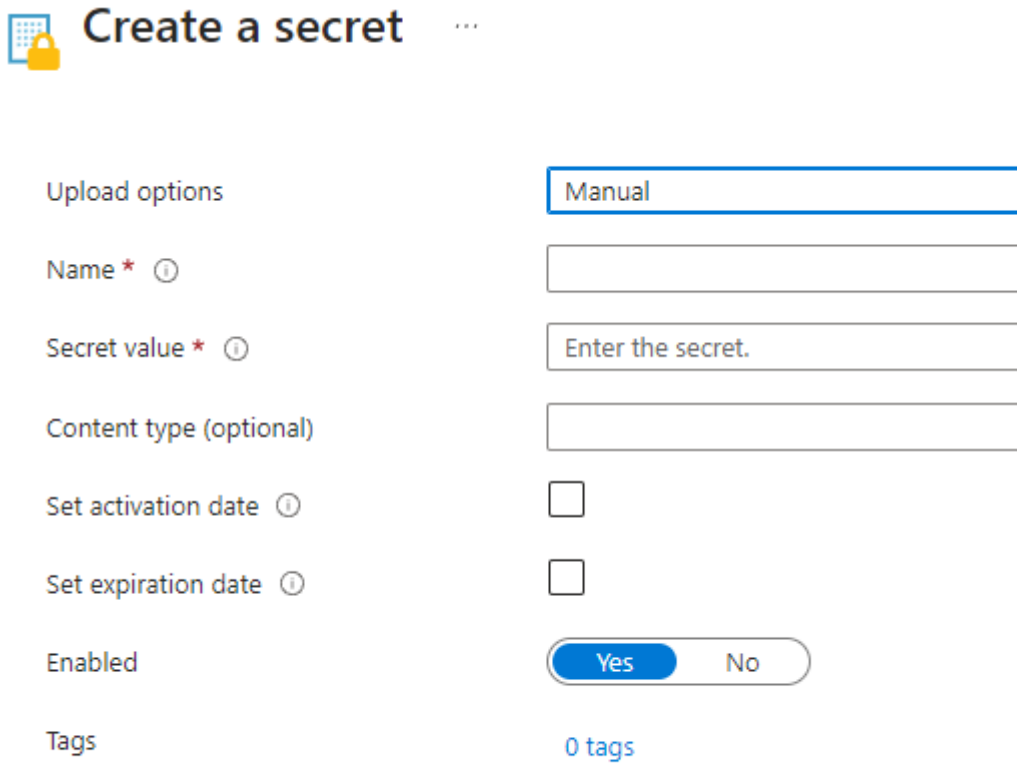
Grundzulagen zu Azure Key Vault

Was ist Azure Key Vault?

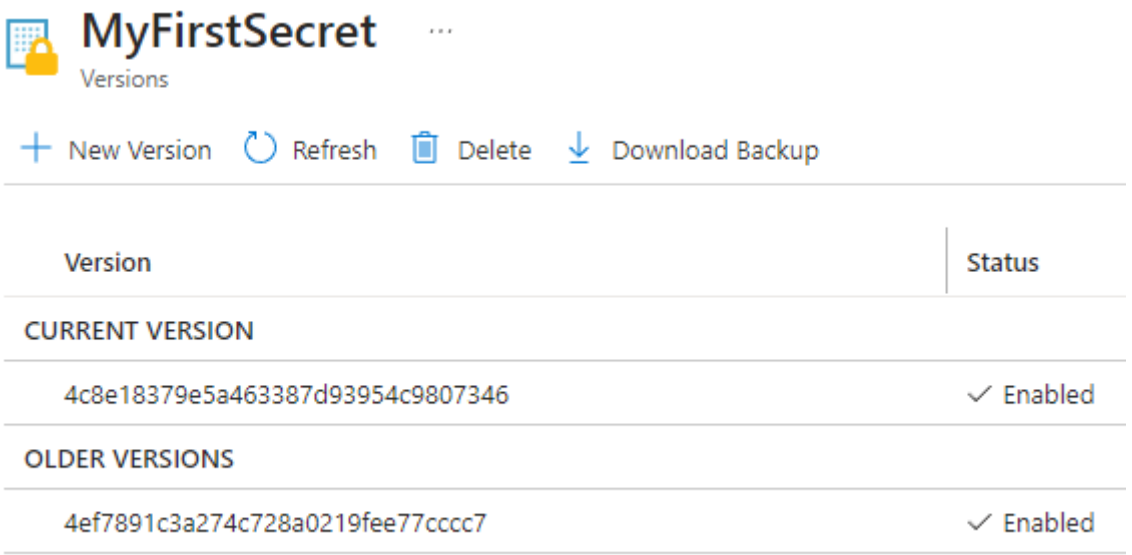
- Azure Key Vault ist eine Azure-Cloudlösung um secrets sicher zu speichern und darauf zuzugreifen
  Er beinhaltet:
  1. Secrets Management (Geheimnisse)
  2. Key Management (Schlüssel)
  3. Certificate Management (Zertifikate)
- Wo kann er verwendet werden?
  - in Anwendungen, z.B. C# Code der Werte aus dem Key Vault liest
  - Azure-Ressourcen
    - Azure App Services
    - Azure Functions
    - Azure Virtual Machines
    - Azure Dev Ops
    - Azure Cosmos DB
    - …



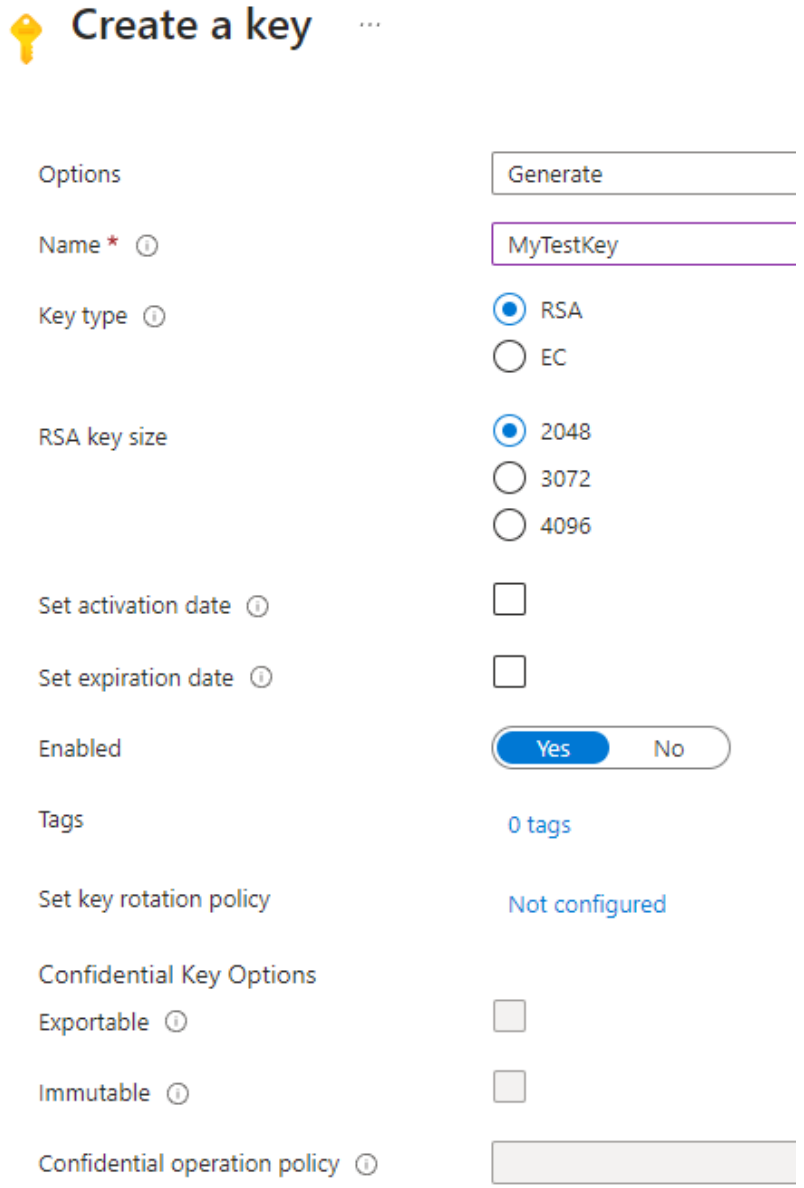
- **Secrets**: Data that you want to secure in the Azure Key Vault, e.g., passwords, API keys, connection strings, …
- **Create a secret:**
  - Name
  - Value
  - optional: activation date
  - optional: expiration date
  - optional: enabled / disabled state



- **View the secret‘s value**
- **Version history**
  - Code can access the current or a specific version
- **Delete secrets**
- **Manage deleted secrets**
  - recover secret from soft-delete
  - or purge

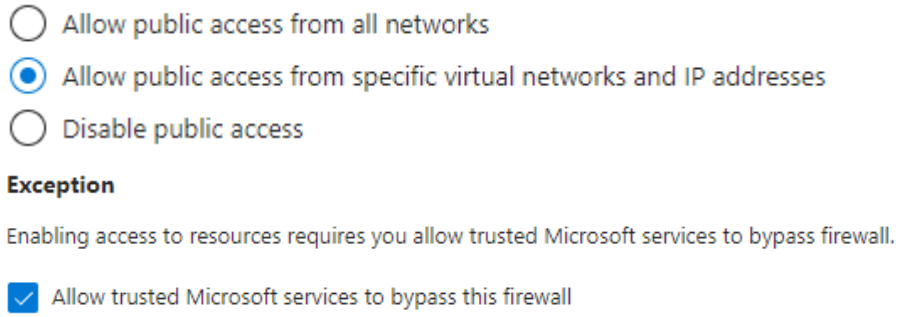


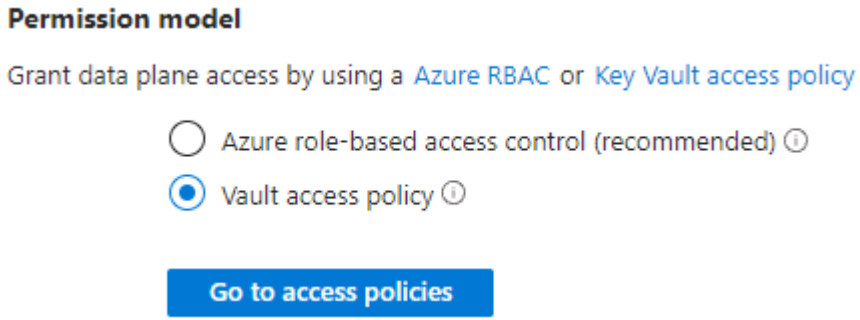
- **Key**: Store symmetric or asymmetric cryptographic keys
- **Create a key:**
  - Name
  - Key type
  - Key size
  - optional: activation date
  - optional: expiration date
  - optional: key rotation policy
  - optional: enabled / disabled state
- **Import key / restore key from backup**
- **Key history**
- **Premium: Bring your own Hardware Security Module (HSM)**



- **Certificate**: Digital certificates to secure the communication over networks
- **Speichern von SSL/TLS-Zertifikaten und -Schlüsseln**
  - Importieren vorhandener Zertifikate
  - Eigene Zertifikate erstellen
- **Certificate history**
- **Export certificates**
- **Autorenew certificates before they expire (self-signed certificates)**

- Um auf einen Azure Key Vault zugreifen zu können, benötigen wir
  1. authentication (muss eingeloggt sein)
  2. authorization (der angemeldete Benutzer muss über eine Zugriffsberechtigung verfügen)
  3. (optional) Zugriff über erlaubtes Netzwerk
- Es gibt zwei Arten von Berechtigungsmodellen
  1. Azure role-based access control (RBAC)
  2. Vault access policy





Unterschied: Access Policies (Legacy) vs. Azure RBAC Access Policies (Legacy)

Vault Access Policies (Legacy)

- **Verwaltung direkt im Key Vault**
- Rechte wie:
  - Get/List Secrets
  - Encrypt/Decrypt Keys
  - Import/Export Certificates
- Gilt nur für diesen **einen Vault**
- Einfach einzurichten
- Wird noch unterstützt, aber nicht mehr empfohlen

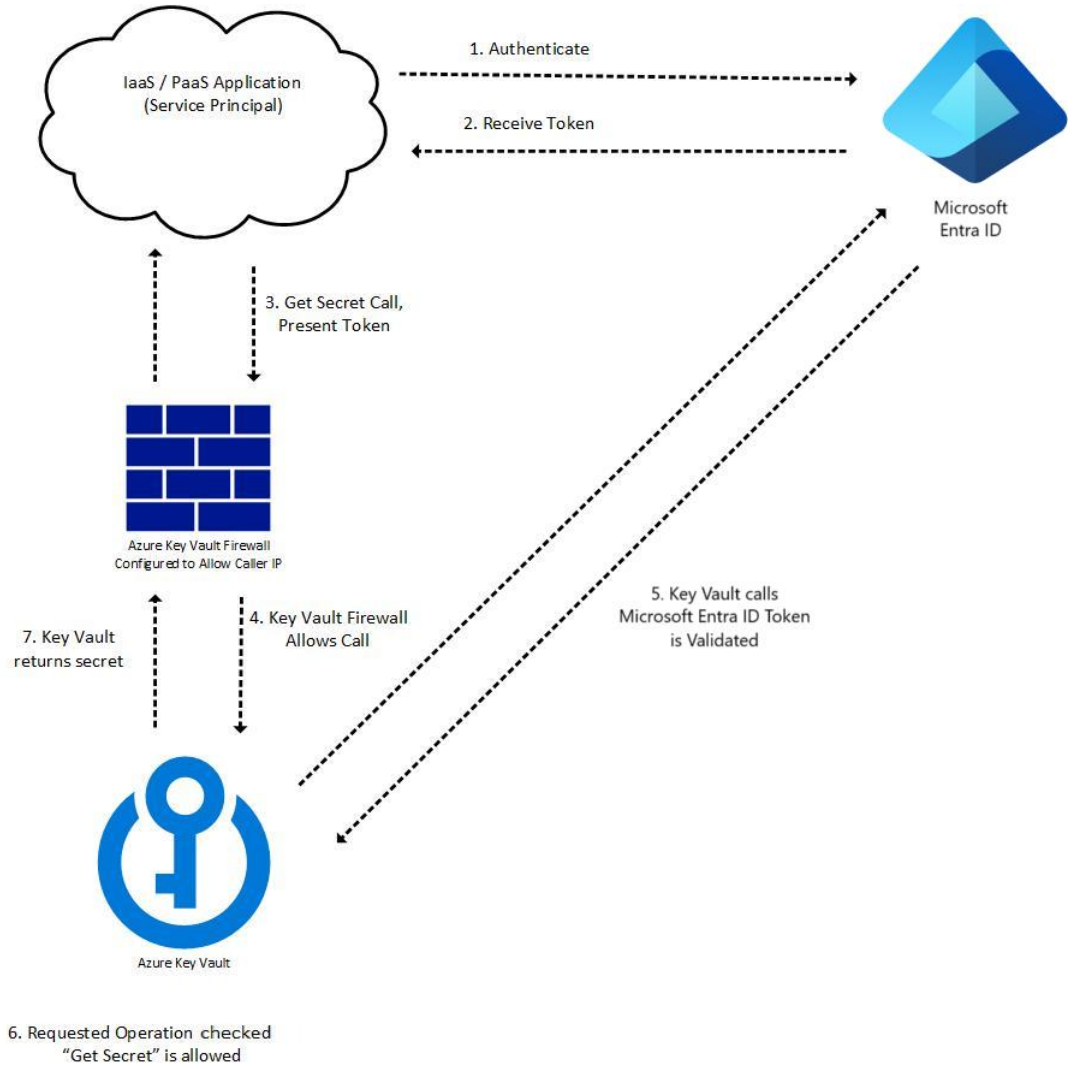
👉 Gut für: kleine Projekte, schnelle Tests, Legacy-Szenarien

Azure RBAC (Role-Based Access Control)

- **Verwaltung über Azure Resource Manager (IAM)**, nicht im Vault selbst
- Standardrollen wie:
  - Key Vault Secrets User / Key Vault Secrets Officer
  - Key Vault Certificate User / Key Vault Certificate Officer
  - Key Vault Crypto User / Key Vault Crypto Officer
- Rollen können **auf Subscription-, Resource Group- oder Vault-Ebene** vergeben werden
- Einheitlich mit dem Rest von Azure IAM

👉 Gut für: Enterprise-Umgebungen, konsistentes IAM

Azure Key Vault: Zugreifen auf Azure Key Vault-Secrets



https://learn.microsoft.com/en-us/azure/key-vault/general/authentication

Welche Identität wird vom Code für den Zugriff auf Azure Key Vault verwendet?


```
new DefaultAzureCredential()
```

DefaultAzureCredential Class

Reference

Feedback

Definition

Namespace: Azure.Identity

Assembly: Azure.Identity.dll

Package: Azure.Identity v1.12.1

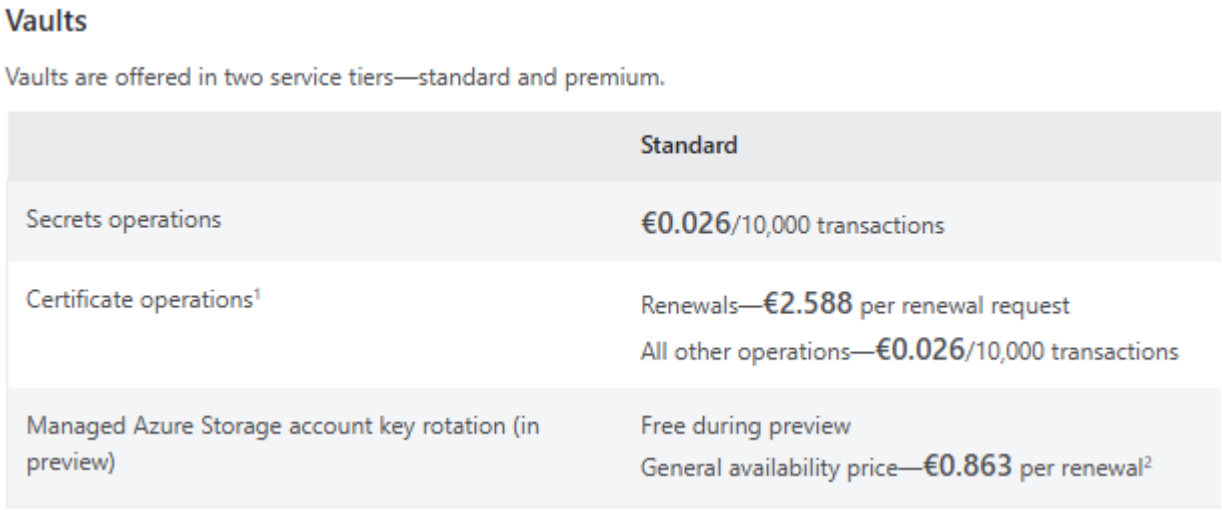
Source: DefaultAzureCredential.cs

Simplifies authentication while developing apps that deploy to Azure by combining credentials used in Azure hosting environments with credentials used in local development. In production, it's better to use something else. See Usage guidance for DefaultAzureCredential.

Attempts to authenticate with each of these credentials, in the following order, stopping when one provides a token:

- EnvironmentCredential
- WorkloadIdentityCredential
- ManagedIdentityCredential
- SharedTokenCacheCredential
- VisualStudioCredential
- VisualStudioCodeCredential
- AzureCliCredential
- AzurePowerShellCredential
- AzureDeveloperCliCredential
- InteractiveBrowserCredential

- **i.d.R. kosten Key Vaults nur wenige Cents im Monat**
- **Abrechnung pro Request:**
  - **Secrets:** Jeder GET, LIST, SET, DELETE = 1 Transaktion
  - **Keys:** Jede Krypto-Operation (Sign, Encrypt, Decrypt) wird einzeln berechnet
  - **Certificates:** Berechnung pro Version & Renewal
- Bei hardware security module (HSM) leicht höhere Kosten



Vaults

Vaults are offered in two service tiers—standard and premium.

	Standard
Secrets operations	€0.026/10,000 transactions
Certificate operations[1]	Renewals—€2.588 per renewal request All other operations—€0.026/10,000 transactions
Managed Azure Storage account key rotation (in preview)	Free during preview General availability price—€0.863 per renewal[2]

Demo: Refactoring einer .NET Anwendung mit Azure OpenAI

Demo: Azure OpenAI

- **ChatGPT** ist ein Produkt von OpenAI
- **Azure OpenAI ist ein Azure Service mit OpenAI Modellen**
- Gleiche Modellfamilien, aber:
  - eigene Azure Ressource
  - eigene Abrechnung
  - eigene Sicherheitsgrenzen
- Zugriff über API Keys

Demo: Azure OpenAI heute und Azure AI Foundry morgen

🕑 Früher: eigenständige Azure OpenAI Ressource

👉 Heute: Integration in Azure AI Foundry

Foundry erweitert:

- weitere Modelle (DeepSeek, xAI, …)
- Agenten
- Datenanbindung

Für diesen Vortrag entscheidend:

- Authentifizierung
- Secret Management
- Die Oberfläche ändert sich, aber das Sicherheitsproblem bleibt identisch.

Demo: Was kann ich mit Azure OpenAI aus .NET machen?

Typische AI Use Cases aus .NET Sicht:

- Textverarbeitung / Natural Language Processing
  - Zusammenfassungen
  - Sentiment Analyse
  - Klassifikation
- Business Use Cases
  - Support-Antworten
  - E-Mail-Entwürfe
  - Auswertung von Feedback
  - …
- Bilder: Bildgenerierung, Bildanalyse
- Audio: Speech to Text, Text to Speech
- …

Demo: Warum der Azure Key Vault hier so wichtig ist

- API Keys sind Kostenfaktor, **da Kosten bei Nutzung entstehen**
- API Keys sind Zugriffsberechtigung
- API Keys gehören nicht:
  - in code
  - in appsettings
  - in repositories / source-control

Key Vault ermöglicht:

- zentrale Verwaltung des API Keys / Zugriffskontrolle
- zentrale Rotation des API Keys

Demo: Vorher - .NET-Konsolenanwendung mit hartcodiertem API-Key in appsettings.json

Program.cs

azurecloudnative20 | Program | <top-level-statements-entry-point>

```
13
14          var settings = new Settings();
15          configuration.GetSection("Settings").Bind(settings);
16
17          var openAiEndpoint = settings.OpenAIEndpoint;
18          var openAiKey = settings.OpenAIApiKey;
19          var deploymentName = settings.OpenAIDeploymentName;
20
21          Console.WriteLine("Calling Azure OpenAI...");
22
23          // Create Azure OpenAI client (2.x style)
24          var azureClient = new AzureOpenAIClient(
25              new Uri(openAiEndpoint),
26              new AzureKeyCredential(openAiKey));
27
28          // Get a chat client for your deployment
29          ChatClient chatClient = azureClient.GetChatClient(deploymentName);
30
31          var text = File.ReadAllText("complaint.txt");
32
33          // Build JSON-structured prompt
34          string prompt =
35          $@"
36          Du bist ein Assistent, der Texte analysiert.
37          Lies den folgenden Beschwerdetext und gib ausschließlich gültiges JSON zurück.
38
39          Das JSON muss genau dieses Format haben:
40
41          {{
42            ""summary"": ""string"",
43            ""sentiment"": ""positive | neutral | negative"",
44            ""sentimentReasons"": [""string""],
45            ""issuesDetected"": [""string""],
46            ""confidence"": 0.0
47          }}
48
49          Hier ist der Text:
50          {text}
51          ";
52
53          var result = await chatClient.CompleteChatAsync(
54          [
55              new UserChatMessage(prompt)
56          ]);
```

appsettings.json

Schema: https://www.schemastore.org/appsettings.json

```
1   {
2       "Settings": {
3           "OpenAIEndpoint": "https://openai-liebeck.c
4           "OpenAIApiKey": "CEyCKykDn6CzqVXJmOM7QUmM8e
5           "OpenAIDeploymentName": "gpt-5-mini"
6       }
7   }
```

Create a key vault ...

Basics | Access configuration | Networking | Tags | Review + create

Azure Key Vault is a cloud service used to manage keys, secrets, and certificates. Key Vault eliminates the need for developers to store security information in their code. It allows you to centralize the storage of your application secrets which greatly reduces the chances that secrets may be leaked. Key Vault also allows you to securely store secrets and keys backed by Hardware Security Modules or HSMs. The HSMs used are Federal Information Processing Standards (FIPS) 140-2 Level 2 validated. In addition, key vault provides logs of all access and usage attempts of your secrets so you have a complete audit trail for compliance.

Project details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Microsoft Azure Sponsorship

Resource group * : LIT-PartnerCredits

Create new

Instance details

Key vault name * : azurecloudnative20

Region * : Germany West Central

Pricing tier * : Standard

Recovery options

Soft delete protection will automatically be enabled on this key vault. This feature allows you to recover or permanently delete a key vault and secrets for the duration of the retention period. This protection applies to the key vault and the secrets stored within the key vault.

To enforce a mandatory retention period and prevent the permanent deletion of key vaults or secrets prior to the retention period elapsing, you can turn on purge protection. When purge protection is enabled, secrets cannot be purged by users or by Microsoft.

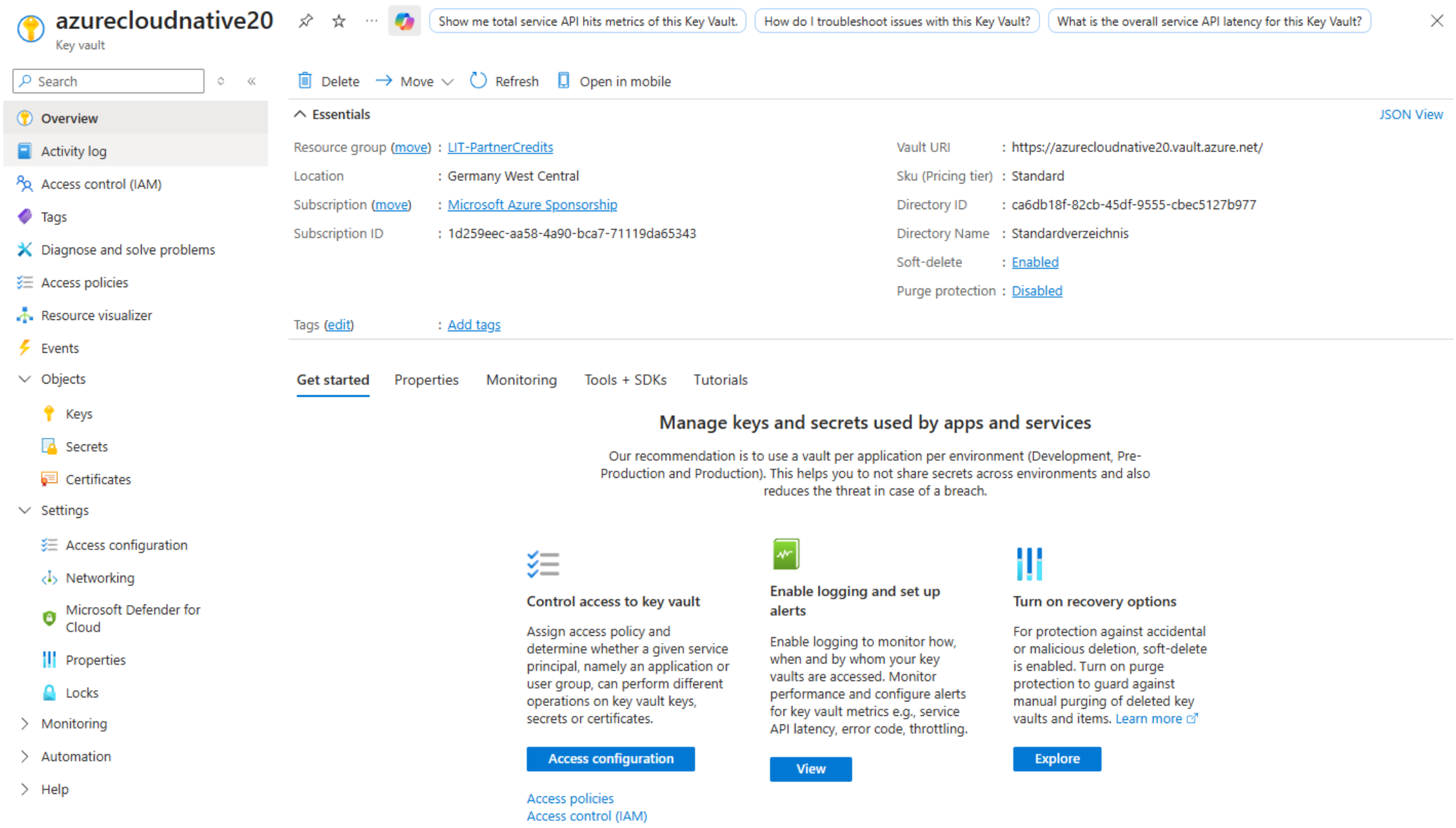
Soft-delete : Enabled

Days to retain deleted vaults * : 90

Purge protection :

- (●) Disable purge protection (allow key vault and objects to be purged during retention period)
- ( ) Enable purge protection (enforce a mandatory retention period for deleted vaults and vault objects)

Demo: https://azurecloudnative20.vault.azure.net/



Demo: Nachher - .NET-Konsolenanwendung mit settings aus Azure Key Vault

Program.cs

azurecloudnative20 | Program | <top-level-statements-entry-poi

```
12
13      var client = new SecretClient(new Uri(keyVaultUrl), new DefaultAzureCredential());
14      KeyVaultSecret secret = client.GetSecret("FirstSecret");
15      Console.WriteLine($"Reading 'FirstSecret' from Key Vault: {secret.Value}");
16
17      // Load configuration
18      var configuration = new ConfigurationBuilder()
19          .SetBasePath(AppContext.BaseDirectory)
20          .AddJsonFile("appsettings.json", optional: false, reloadOnChange: true)
21          .AddAzureKeyVault(
22              new Uri(keyVaultUrl),
23              new DefaultAzureCredential())
24          .Build();
25
26      var settings = new Settings();
27      configuration.GetSection("Settings").Bind(settings);
28
29      var openAiEndpoint = settings.OpenAIEndpoint;
30      var openAiKey = settings.OpenAIApiKey;
31      var deploymentName = settings.OpenAIDeploymentName;
32
33      Console.WriteLine("Calling Azure OpenAI...");
34
35      // Create Azure OpenAI client (2.x style)
36      var azureClient = new AzureOpenAIClient(
37          new Uri(openAiEndpoint),
38          new AzureKeyCredential(openAiKey));
39
40      // Get a chat client for your deployment
41      ChatClient chatClient = azureClient.GetChatClient(deployment);
42
43      var text = File.ReadAllText("complaint.txt");
44
45      // Build JSON-structured prompt
46      string prompt =
47      $@"
48      Du bist ein Assistent, der Texte analysiert.
49      Lies den folgenden Beschwerdetext und gib ausschließlich gültiges JSON zurück.
50
51      Das JSON muss genau dieses Format haben:
52
53      {{
54        ""summary"": ""string"",
55        ""sentiment"": ""positive | neutral | negative"",
```

appsettings.json

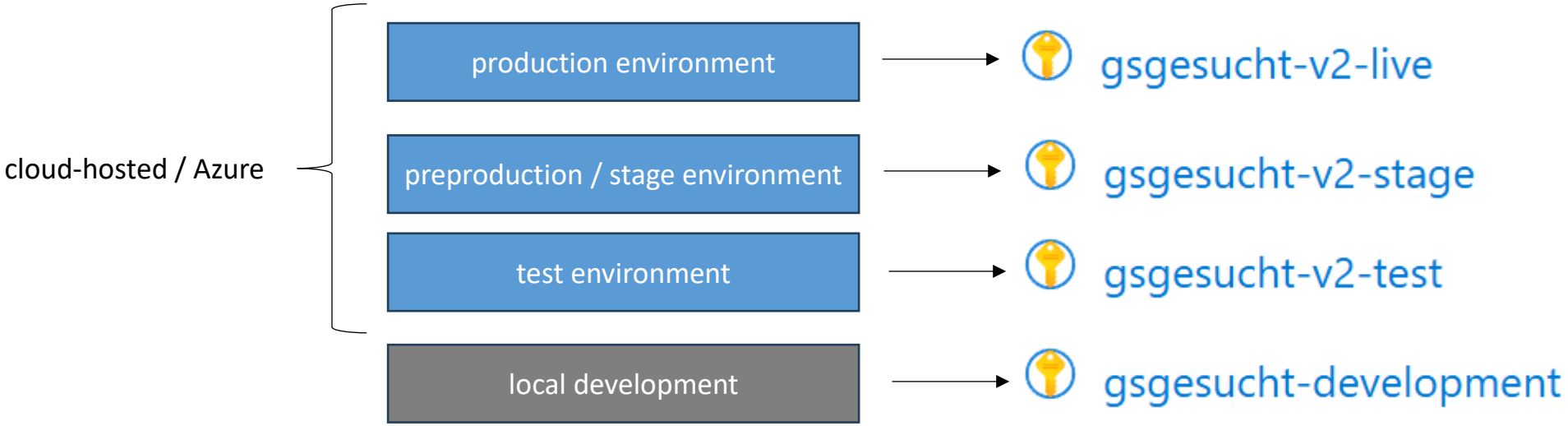
Schema: https://www.schemastore.org/appsettings.json

```
1   {
2       "Settings": {
3           "OpenAIEndpoint": "",
4           "OpenAIApiKey": "",
5           "OpenAIDeploymentName": ""
6       }
7   }
```

Azure Key Vault Best Practices

Trennung von Key Vault nach Environment:

Use a vault per application per environment (development, preproduction, and production), per region. [1]



- Key Vault URL nicht hardcoden sondern als environment variable

[1] https://learn.microsoft.com/en-us/azure/key-vault/general/best-practices

Rate Limiting / Caching

- Azure Key Vault hat ein rate limit und gibt HTTP status code 429 zurück, wenn er zu häufig angefragt wird

- Lösung: Caching!

  Cache the secrets you retrieve from Azure Key Vault in memory, and reuse from memory whenever possible. Re-read from Azure Key Vault only when the cached copy stops working (e.g. because it got rotated at the source). [2]

[2] https://learn.microsoft.com/en-us/azure/key-vault/general/overview-throttling

Soft-delete protection (*nach Löschen in recoverable state)*

- *Soft-delete protection is strongly encouraged.*[3]

Purge protection

- Verhindert die permanente Löschung, bis die retention period abgelaufen ist (auch für Administratoren)
- Schützt vor böswilligen oder versehentlichen Löschungen
- Warum purge protection?
  - schützt vor menschlichen Fehlern
  - eventuell von compliance gefordert, z.B. ISO 27001
  - ohne purge protection kann ein Angreifer den ganzen key vault unwiderruflich löschen

[3] https://learn.microsoft.com/en-us/azure/key-vault/general/soft-delete-overview

Wenn ihr Zugriff auf mehr als einen Azure-Tenant habt

- DefaultAzureCredential mit TenantId

```
var keyVaultEndpoint = new Uri(Environment.GetEnvironmentVariable("GSGesuchtKeyVault"));
builder.Configuration.AddAzureKeyVault(keyVaultEndpoint, new DefaultAzureCredential(
    new DefaultAzureCredentialOptions()
    {
        TenantId = "ca6db18f-82cb-45df-9555-cbec5127b977",
    })
);
```

- LoginHint

```
credential = new InteractiveBrowserCredential(new InteractiveBrowserCredentialOptions
{
    TenantId = "c29e447c-c64c-4812-90c6-c0804fc48653",
    LoginHint = "mail@gmail.com",
});
```

- **Monitoring & Auditing aktivieren**
  - Logging aktivieren (Logging wer & wann auf welches secret zugegriffen wurde)
  - Alerts konfigurieren (z.B. bei delete oder purge-operation)
  - retention policies setzen, z.B. audit logs für 180 Tage
- **Bei lokaler Entwicklung Lese-Reihenfolge festlegen**
  1. Default Werte aus dem Key vault
  2. Überschreiben mit Werten aus secrets (z.B. bei Debugging mit anderen Werten oder LoginHint pro Developer)

- **Managed Identities anstatt client secrets**
  - Überall Entra-ID / RBAC nutzen anstatt client secrets (ID + Passwort)
- **Principle of least privilege** (wie überall in Azure)
  - RBAC passend wählen (z.B. Officer für Senior-Entwickler, Reader für Junior-Entwickler)
  - Bei app registration: nur GET und LIST, nicht SET oder DELETE
  - Use IP restriction
- **IP-Restriktion**
  - falls ein Authentication-Token gestohlen wird, wird trotzdem noch Netzwerkzugriff benötigt
  - ISO 27001 benötigt network-level control

Danke!

LinkedIn

GitHub Copilot Newsletter