

# How Do You Get an Access Token in Azure?

## The .NET Developer's Guide to Token Credentials



# Why Did I Specialize In This Topic?



AZ-204 Azure  
Developer Associate

## How Did I Approach This?

# My Philosophy for Learning

You Don't Really Understand It Until You Can Explain It!



# My Study Process

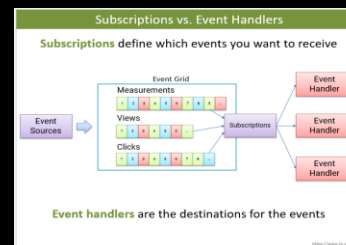


## Cloud Debugger

- Explore and learn Azure
- Open-source



CONTENT



## Azure for Developers Training Workshop



Blog Posts



Presentations

# Did I Make it?

# Passing the Certificate



**Tore Nestenius**

has successfully passed all requirements for

**Microsoft Certified: Azure Developer Associate**

Credential ID: 7B1F1B97C26520BD

Certification number: B2KAE5-F97537

Earned on: 23 August 2024

Expires on: 24 August 2026

A handwritten signature in black ink, appearing to read "Satya N.".

Satya Narayana Nadella

✓ Online Verifiable

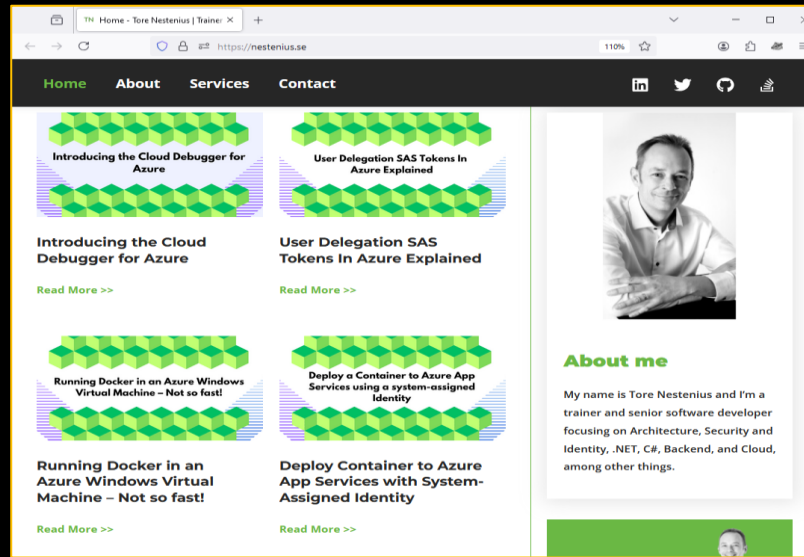
Who Am I?



# About Tore Nestenius



Work: [tn-data.se](https://tn-data.se)



Blog: [nstenius.se](https://nstenius.se)

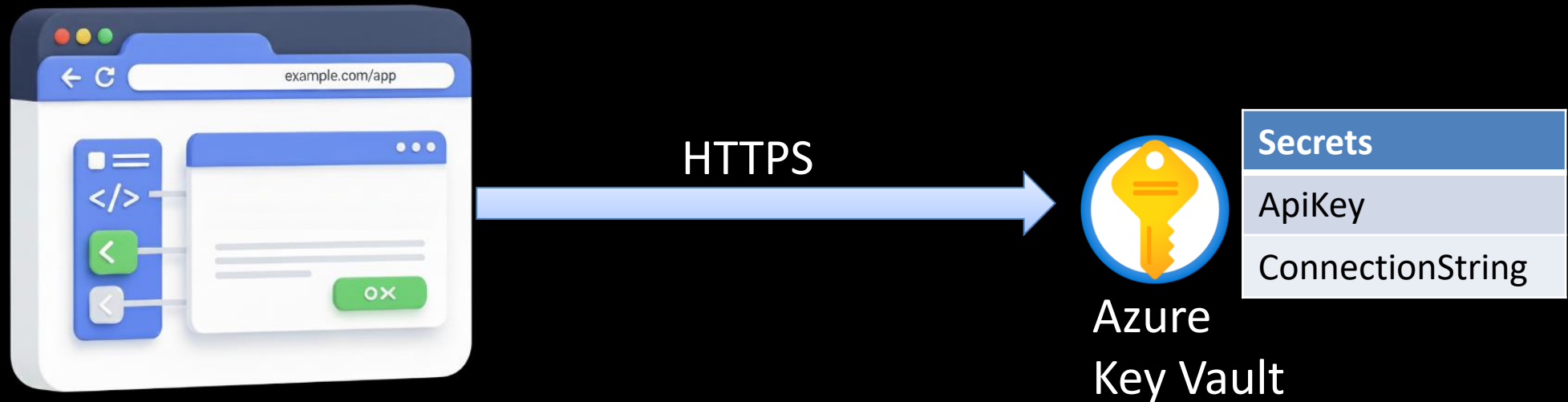


[meetup.com/net-skane](https://meetup.com/net-skane)

# Azure Authentication Fundamentals



# Our App Needs Secrets From Azure Key Vault



Let's Try The Most Obvious Approach First

# Live Coding #1

```
var url = "https://my-demo-keyvault-633.vault.azure.net";

var secret = await $"{url}/secrets/ApiKey"
    .SetQueryParam("api-version", "7.4")
    .GetJsonAsync<Secret>();

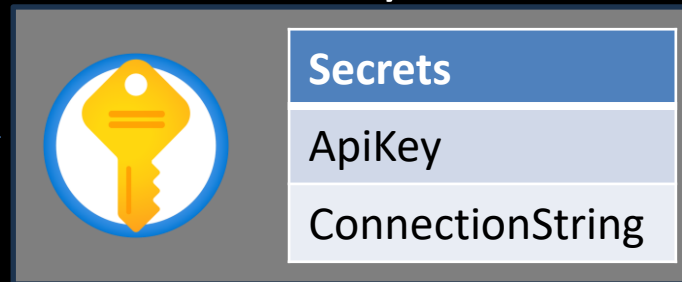
Console.WriteLine($"Secret value: {secret.value}");

public record Secret(string value, string id, object attributes);
```



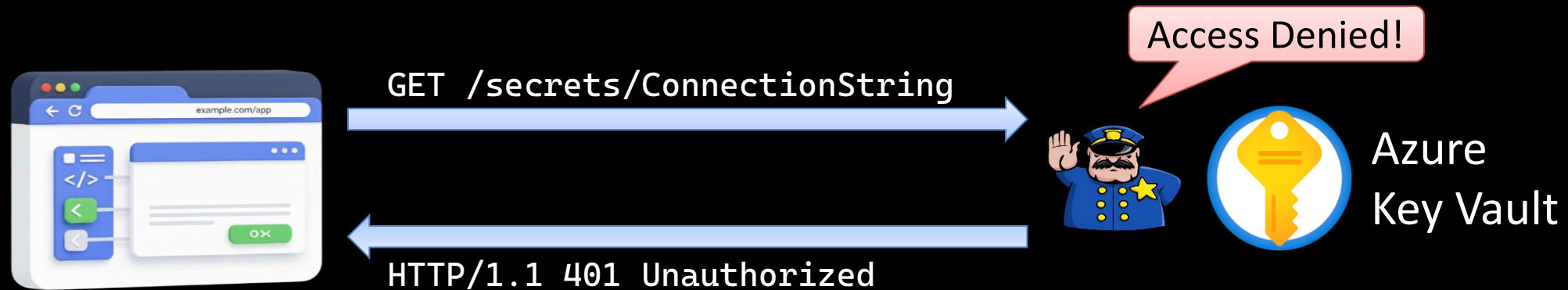
GET /secrets/ConnectionString

## Azure Key Vault



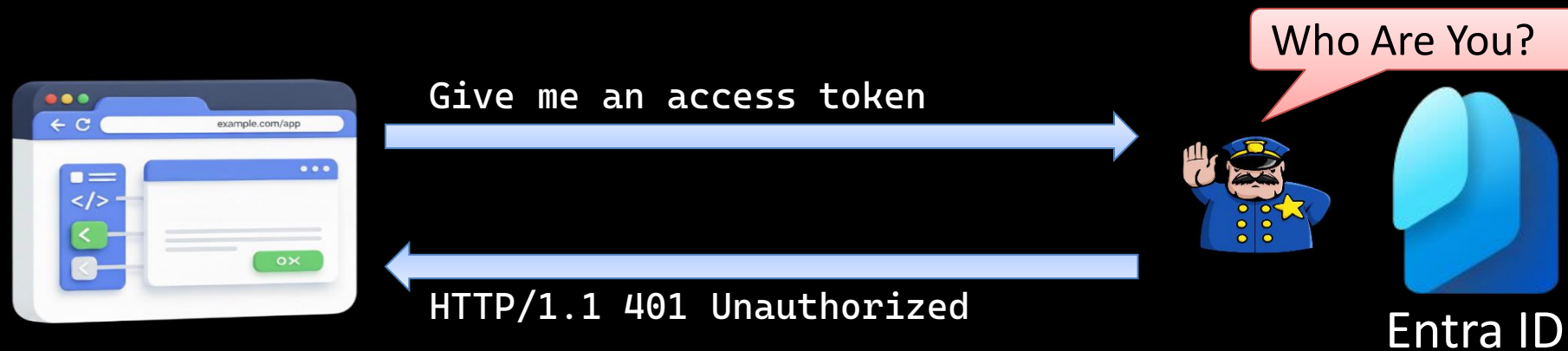
# Authentication In Azure

# We Need An Access Token To Access Key Vault



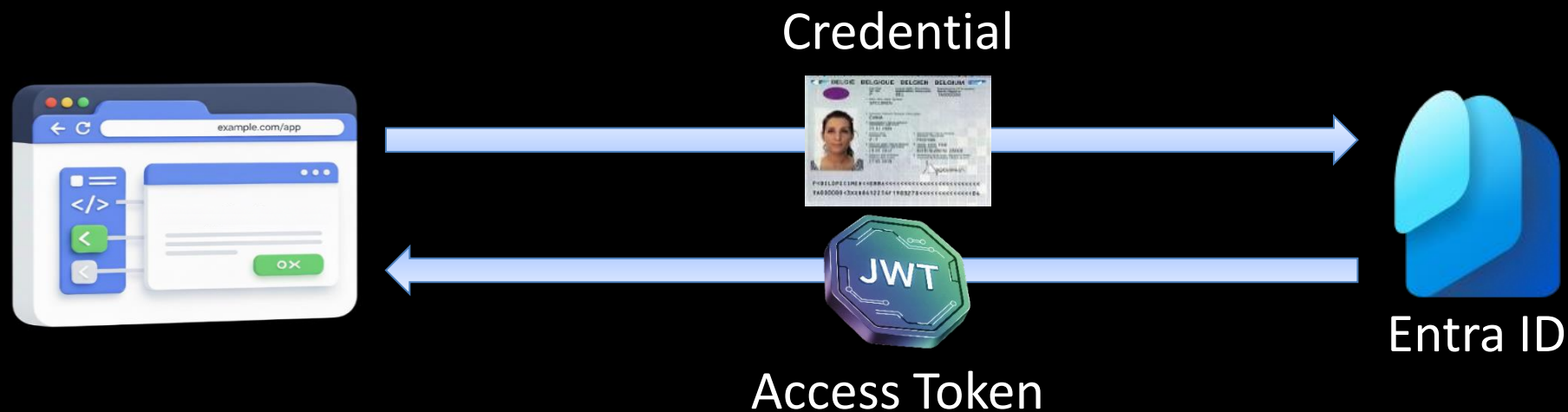
## How Do I Get An Access Token?

# The App Must Authenticate With Entra ID First



## How Does My App Authenticate?

# The App Provides A Valid Credential



## Let's See This In Practice



## Live Coding #2

```
// Get Access Token
var options = new AzureCliCredentialOptions
{ TenantId = "567d82a1-7f61-4da2-b955-d3244ea6e976" };

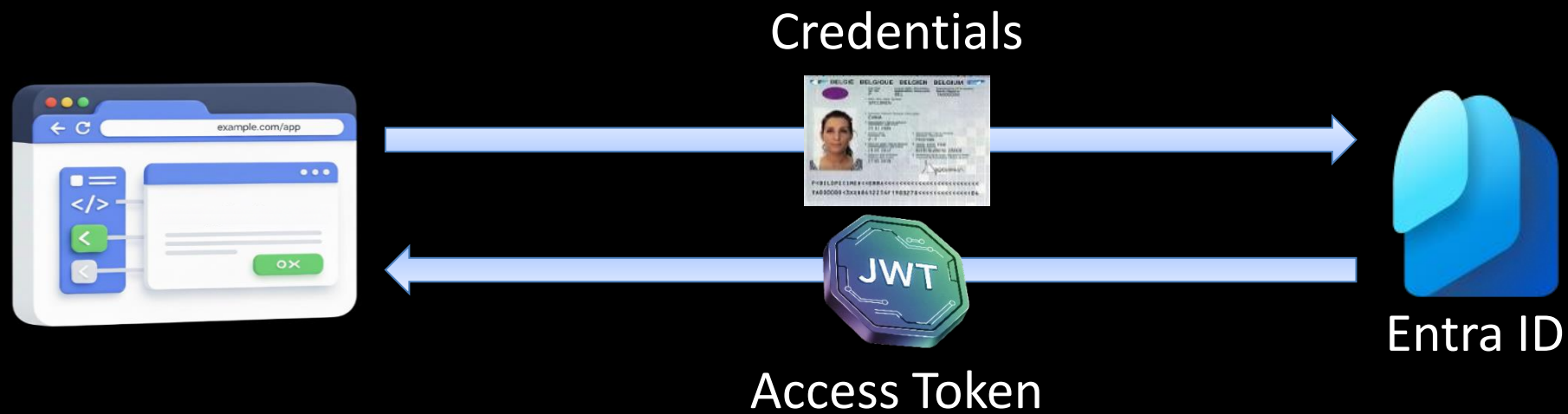
var credential = new AzureCliCredential(options);
var context = new TokenRequestContext(["https://vault.azure.net/.default"]);

var accessToken = await credential.GetTokenAsync(context, default);

Console.WriteLine($"Token acquired:\r\n{accessToken.Token}\r\n\r\n");
```



# What Is Inside The Access Token?



# Exploring the Access Token

```
var accessToken = await credential.GetTokenAsync(tokenContext);
```



The screenshot shows the 'Text Visualizer' application with the following content:

- Expression:** token.Token
- String manipulation:** JWT Decode
- Header:**

```
{
  "typ": "JWT",
  "alg": "RS256",
  "x5t": "yEUwmXWL107Cc-7QZ2W5be0b3sQ",
```
- Payload:**

```
{
  "aud": "https://management.azure.com",
  "iss": "https://sts.windows.net/567d82a1-7f61-4da2-b955-d3244ea6e976/",
  "iat": 1761760935,
  "nbf": 1761760935,
  "exp": 1761766174,
  "acr": "1",
  "acrs": [
    "p1"
  ],
  "aio": "
  "AZQAa/8aAAAAHLbEmTOWM9BbHPyTTwJjJszfuHYnjRJ/63cvXE8S72SSeM0cdS2l0g4b8HuXsgD
  BkC-GC5F4F4M9N-8DF-1D3-4W410-7-1-4W4-1-0C25C63-1-1W4-1-1"

```
- Signature:**

```
ShMkq5fwaIIwUlxbvMEp-7VcO2_9qe0RNS5qRIY0QawHz9RRuYA5AXRr3NESQfPG0YhL8KN8F4b
SEb83Qm31fRbwHD3vmfToDs00QmksYr8MJbq2_UHj0dsNEyTstXiLSkgts0Q4kZxesnXqTZkft
GKzBLe7A36t2XntTu8wAiAXXLf1yTy0AMQgGMfwYHrVRF9JxCh2H3BK1wj1EPtRhptUYS2Xo1sr
xW-BEcpwhsdqykVuMdppj0kNjQsClY2IgKuWrmWfo_HSDQP2y3b12Y0Hf0oUsU0xwoTp1-
```
- Word Wrap:** ☒ Word Wrap

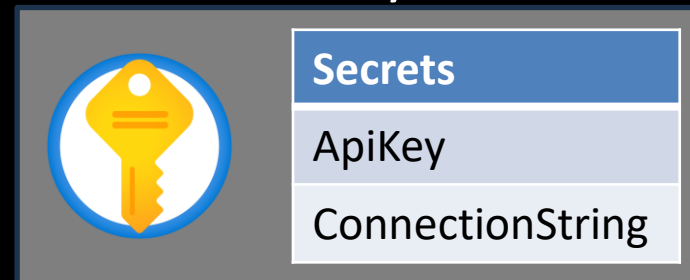
# Using The Access Token

# Live Coding #4

```
var secret1 = await $"{url}/secrets/ApiKey"  
    .SetQueryParam("api-version", "7.4")  
    .WithOAuthBearerToken(accessToken.Token)  
    .GetJsonAsync<Secret>();  
  
var secret2 = await $"{url}/secrets/ConnectionString"  
    .SetQueryParam("api-version", "7.4")  
    .WithOAuthBearerToken(accessToken.Token)  
    .GetJsonAsync<Secret>();  
  
Console.WriteLine($"ApiKey: {secret1.value}");  
Console.WriteLine($"ConnectionString: {secret2.value}");
```



## Azure Key Vault



# What Is A Token Credential?



## It Knows How To Get An Access Token

```
var credential = new AzureCliCredential();
```



## What Credentials Exist?

# The Many Types Of Credentials



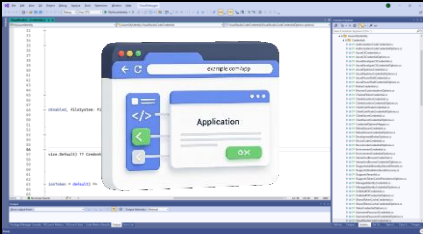
AuthorizationCodeCredential  
AzureCliCredential  
AzureDeveloperCliCredential  
AzurePipelinesCredential  
AzurePowerShellCredential  
BrokerCredential  
ChainedTokenCredential  
ClientAssertionCredential  
ClientCertificateCredential  
ClientSecretCredential  
DefaultAzureCredential

DeviceCodeCredential  
EnvironmentCredential  
InteractiveBrowserCredential  
ManagedIdentityCredential  
OnBehalfOfCredential  
SharedTokenCacheCredential  
UsernamePasswordCredential  
VisualStudioCodeCredential  
VisualStudioCredential  
WorkloadIdentityCredential  
...

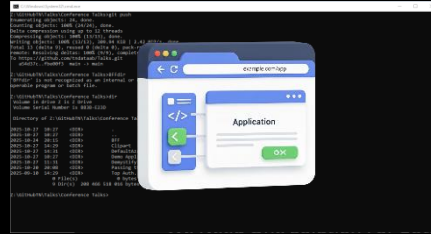
## Why Do We Need So Many?

# We Need To Support Many Environments

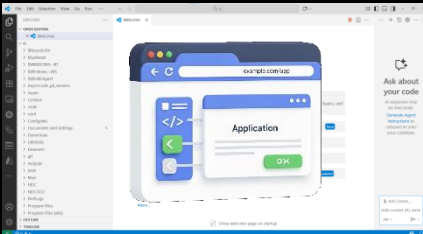
## Outside Azure



Visual Studio



Command line

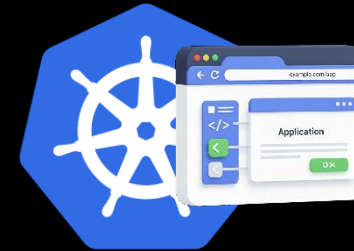


Visual Studio Code



Kubernetes

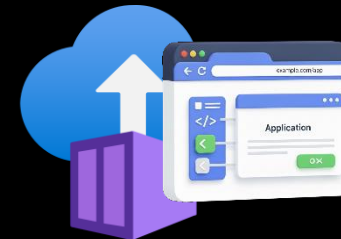
## Inside Azure



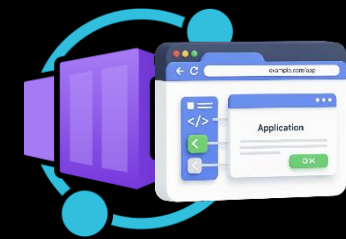
Kubernetes



App Services



Container  
Instances

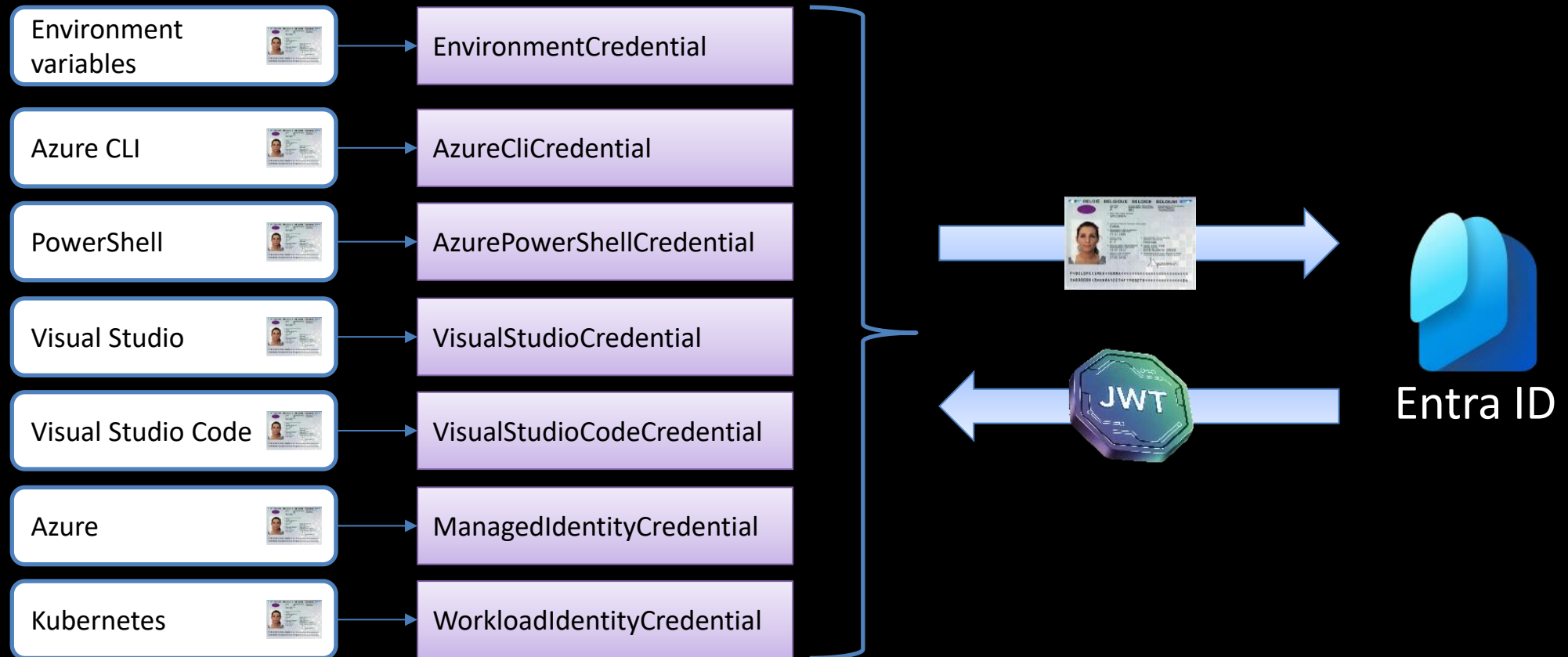


Container  
Apps

# We Need One Solution For All!

# Credential Sources

## Where Do The Credentials Come From?



## What Does This Mean For Our Apps?

# Different Environments Need Different Credentials

Joe's Machine



**VisualStudioCredential**

Build Server



**EnvironmentCredential**

Anna's Machine



**AzureCliCredential**

Azure



**ManagedIdentityCredential**

## What Can This Look Like In Code?



# Live Coding #5

```
var environment = "Anna";

TokenCredential? credential = null;

switch (environment)
{
    case "Joe":
        credential = new VisualStudioCredential();
        break;
    case "Anna":
        credential = new AzureCliCredential();
        break;
    case "BuildServer":
        credential = new EnvironmentCredential();
        break;
    case "Azure":
        credential = new ManagedIdentityCredential();
        break;
}

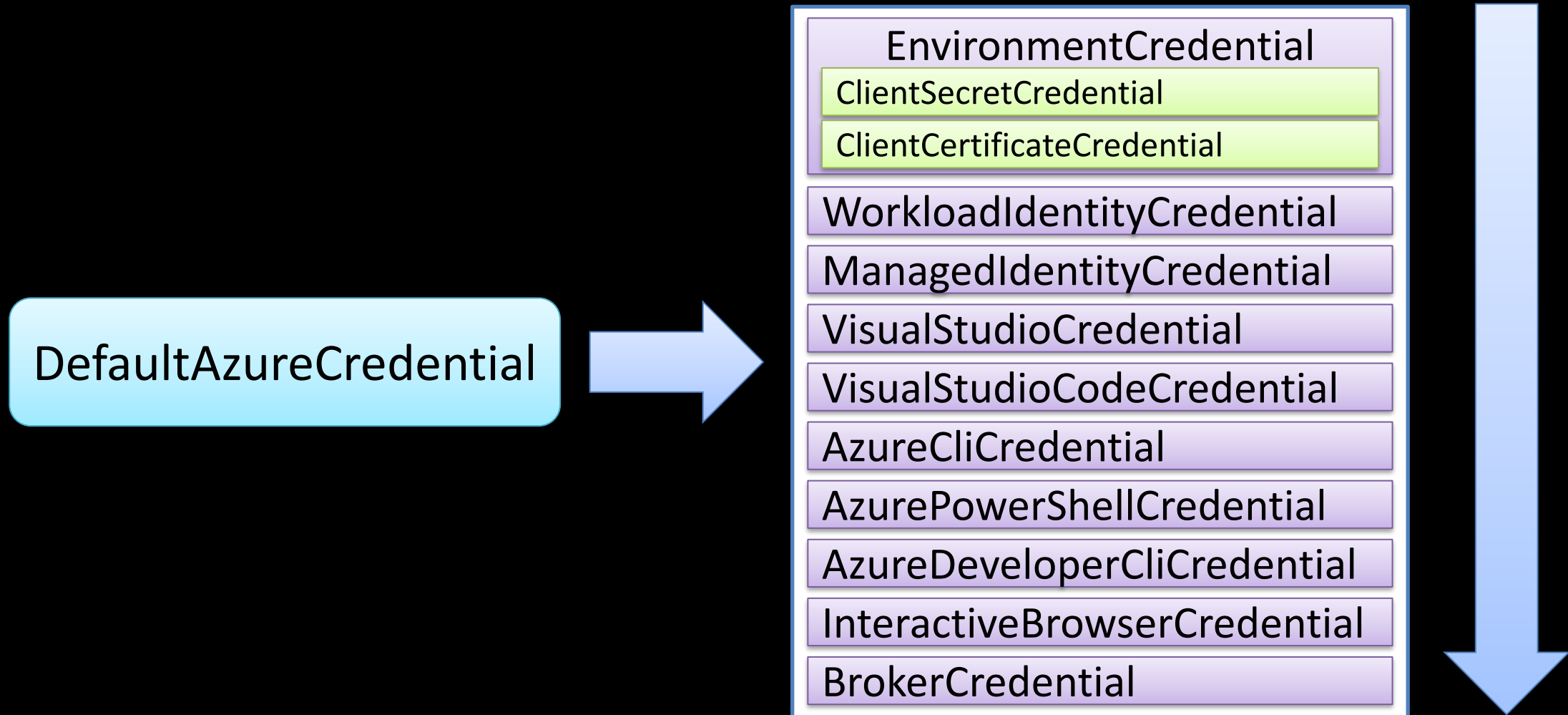
//Use credential to get token
```



There must be  
a better way?

# DefaultAzureCredential

## DefaultAzureCredential To The Rescue



# Live Coding #6

## Joe's Machine



VisualStudioCredential

## Anna's Machine



AzureCliCredential

## Build Server



EnvironmentCredential

## Azure



ManagedIdentity  
Credential

```
var options = new DefaultAzureCredentialOptions
{
    TenantId = "567d82a1-7f61-4da2-b955-d3244ea6e976"
};

var credential = new DefaultAzureCredential(options);

//Use credential to get token
```



Specifying TenantId  
is a best practice!

# DefaultAzureCredential Trouble!



## You Might Get The Wrong Credential

### DefaultAzureCredential

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

You expected credentials here



Read/Write

But you received this one!



Read Only

Can be hard to troubleshoot



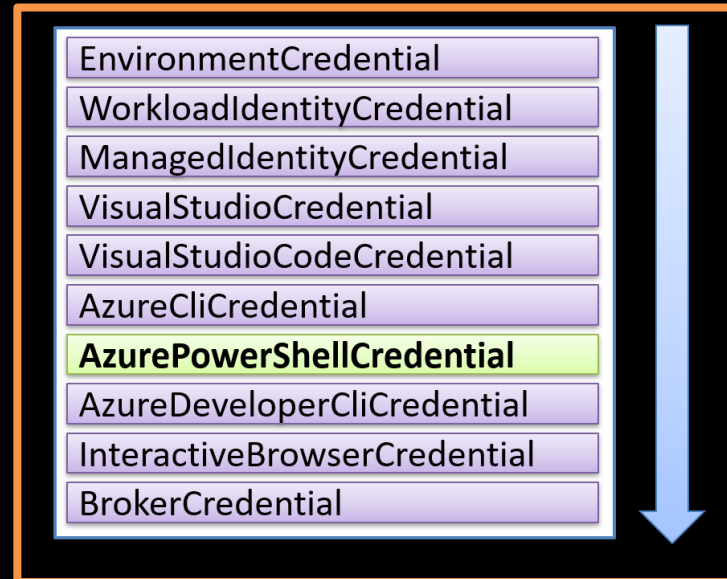
## What Else Is A Problem?



## How Can We Find Out Which Credential Was Chosen?



### DefaultAzureCredential



## What Else Is A Problem?

# Exploring Timeouts When No Credentials Are Found

```
// Benchmark DefaultAzureCredentials

var credential = new DefaultAzureCredential();

var context = new TokenRequestContext(["https://vault.azure.net/.default"]);

var accessToken = await credential.GetTokenAsync(context);

Console.WriteLine($"Token acquired:\r\n {accessToken.Token}\r\n\r\n");
```

# One Solution: Disabling Unwanted Credentials

```
// Exclude certain credentials
var options = new DefaultAzureCredentialOptions
{
    ExcludeEnvironmentCredential = false,
    ExcludeManagedIdentityCredential = false,
    ExcludeVisualStudioCredential = true,
    ExcludeVisualStudioCodeCredential = true,
    ExcludeAzureCliCredential = false,
    ExcludeAzurePowerShellCredential = true,
    ExcludeAzureDeveloperCliCredential = false,
    ExcludeWorkloadIdentityCredential = true,
    ExcludeBrokerCredential = true,
    ExcludeInteractiveBrowserCredential = true
};

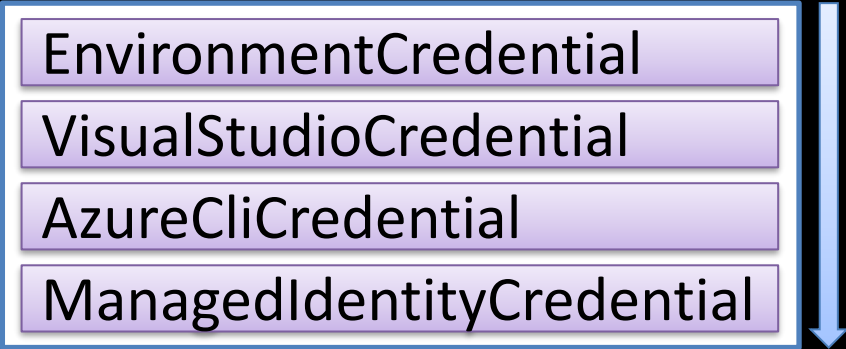
var credential = new DefaultAzureCredential(options);
```

# ChainedTokenCredential

## Custom Control When You Need It

```
var credential = new ChainedTokenCredential(  
    new EnvironmentCredential(),  
    new VisualStudioCredential(),  
    new AzureCliCredential(),  
    new ManagedIdentityCredential());
```

### ChainedTokenCredential



The diagram illustrates the ChainedTokenCredential as a vertical stack of four credential types. From top to bottom, the stack consists of: EnvironmentCredential, VisualStudioCredential, AzureCliCredential, and ManagedIdentityCredential. A large blue arrow on the right side of the stack points downwards, indicating the sequence of attempts.

- EnvironmentCredential
- VisualStudioCredential
- AzureCliCredential
- ManagedIdentityCredential

**Best practice:** Provide TenantId to each credential for production



# Who Uses These Token Credentials?

# The Main Consumer Is The Azure SDK For .NET

Token Credential

Default Azure  
Credentials



Azure SDK Libraries



**Azure.Security.KeyVault.Secrets** by azure-sdk  
This is the Microsoft Azure Key Vault Secrets client library

Direct HTTP Requests to Azure APIs

```
// Using the Access Token and some refactoring
var vaultUrl = "https://my-demo-keyvault-633.vault.azure.net";

var secret1 = await $"{{vaultUrl}}/secrets/ApiKey"
    .SetQueryParam("api-version", "7.4")
    .WithOAuthBearerToken(accessToken.Token)
    .GetJsonAsync<KeyVaultSecret>();
```

## Used By Over 100+ Azure SDK Client Libraries!

# Using The Azure SDK for .NET

```
// Step 1: Create Key Vault client
var client = new SecretClient(vaultUri: new Uri(url),
                             credential: credential);

// Step 2: Get secrets from Key Vault
var response1 = await client.GetSecretAsync("ApiKey");
var response2 = await client.GetSecretAsync("ConnectionString");

// Step 3: Extract secrets from the response
KeyVaultSecret secret1 = response1.Value;
KeyVaultSecret secret2 = response2.Value;

Console.WriteLine($"ApiKey: {secret1.Value}");
Console.WriteLine($"ConnectionString: {secret2.Value}");
```

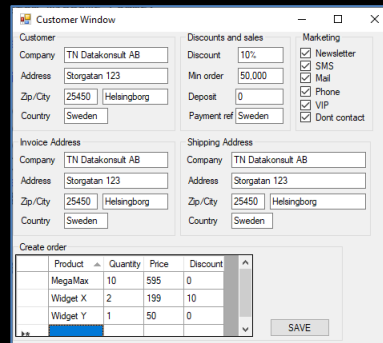


# Interactive Browser Credential

# How Can Joe Access His Secrets?



Joe



Customer Window

Customer

Company: TN Datakonsult AB

Address: Storgatan 123

Zip/City: 25450 Helsingborg

Country: Sweden

Discounts and sales

Discount: 10%

Min order: 50,000

Deposit: 0

Payment ref: Sweden

Marketing

☒ Newsletter

☒ SMS

☒ Mail

☒ Phone

☒ VIP

☒ Dont contact

Invoice Address

Company: TN Datakonsult AB

Address: Storgatan 123

Zip/City: 25450 Helsingborg

Country: Sweden

Shipping Address

Company: TN Datakonsult AB

Address: Storgatan 123

Zip/City: 25450 Helsingborg

Country: Sweden

Create order

Product	Quantity	Price	Discount
MegaMax	10	595	0
Widget X	2	199	10
Widget Y	1	50	0

SAVE

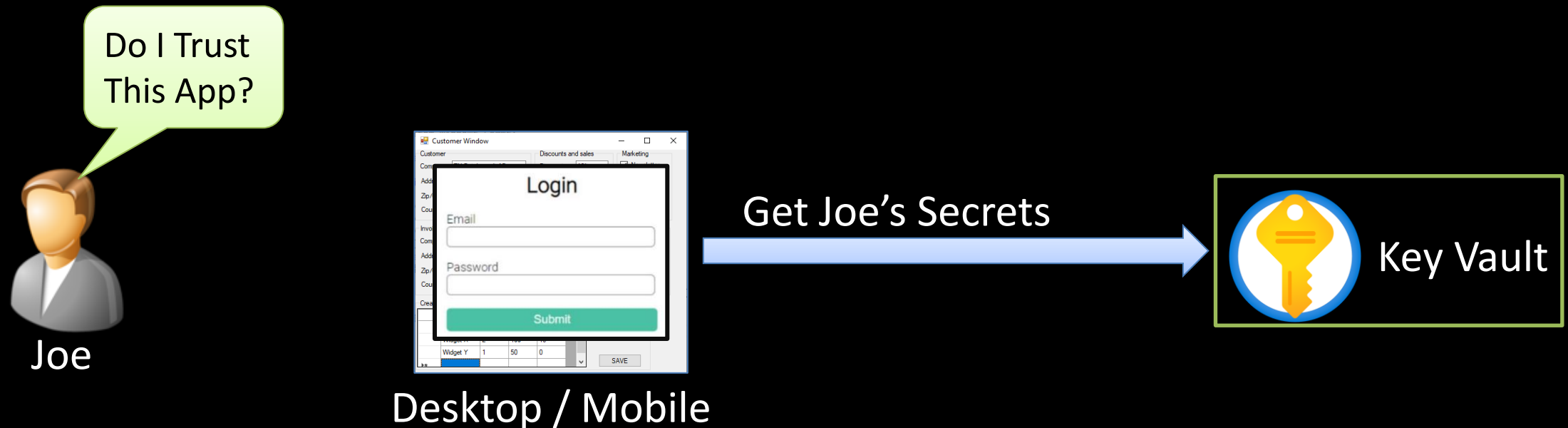
Desktop / Mobile

Get Joe's Secrets



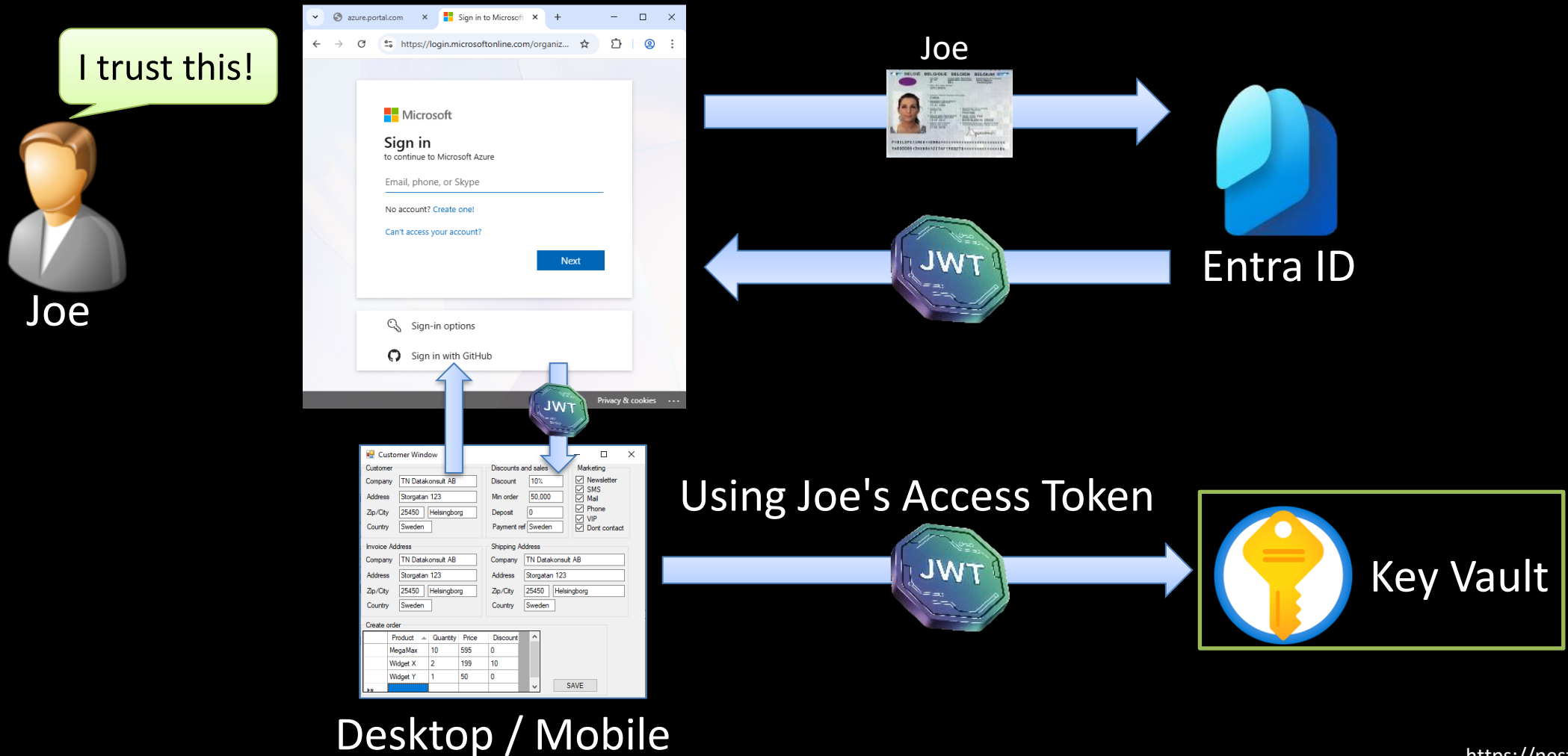
## How Should We Not Do It?

# Don't Build Login Forms Into Your App



## What Is The Correct Way?

## Delegate Authentication To The Browser

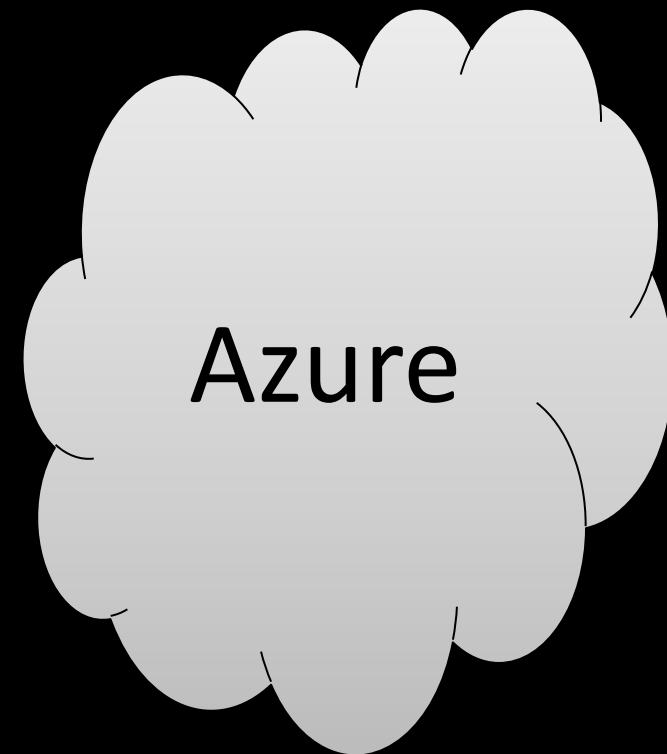
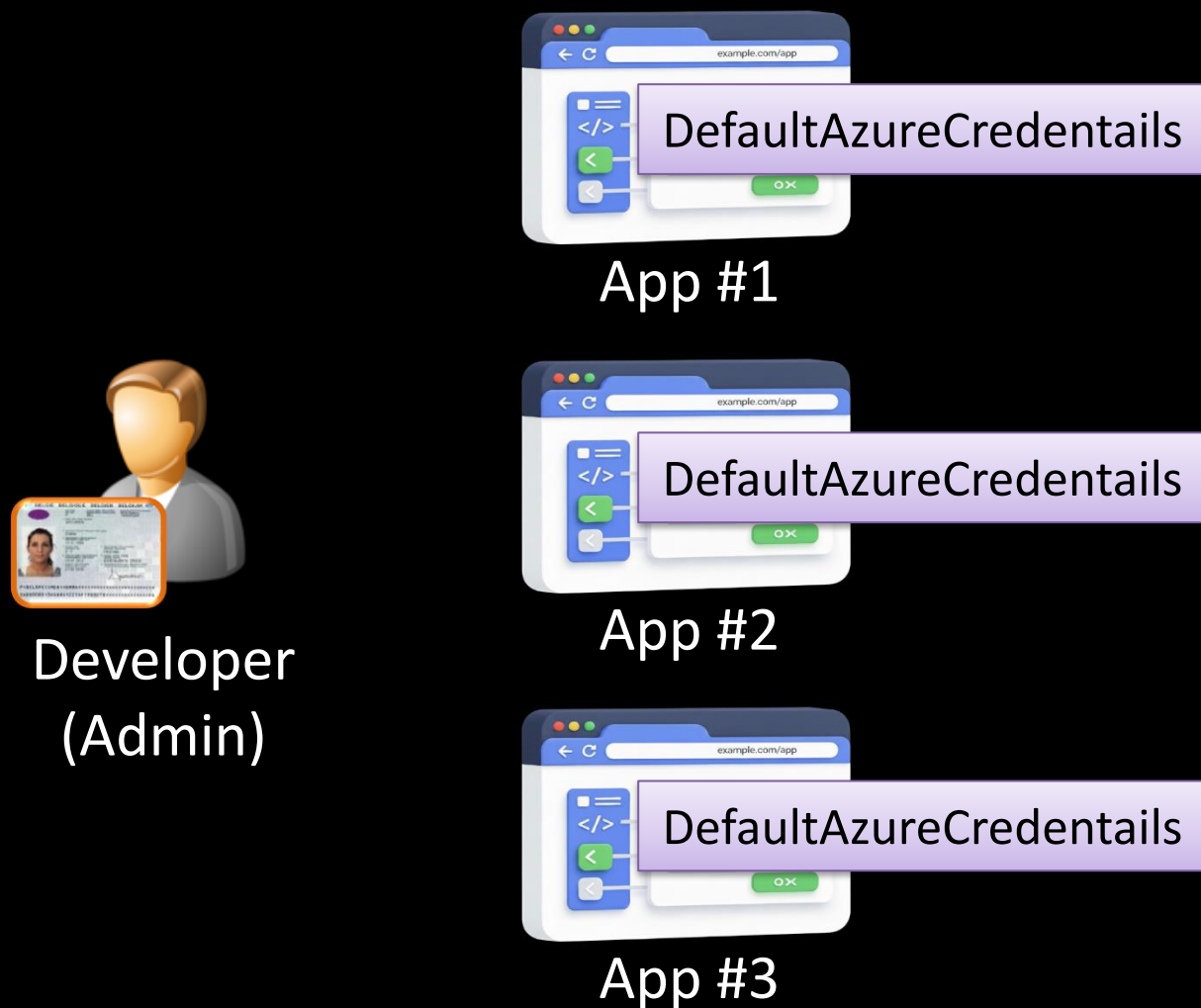


# Exploring Interactive Browser Credential

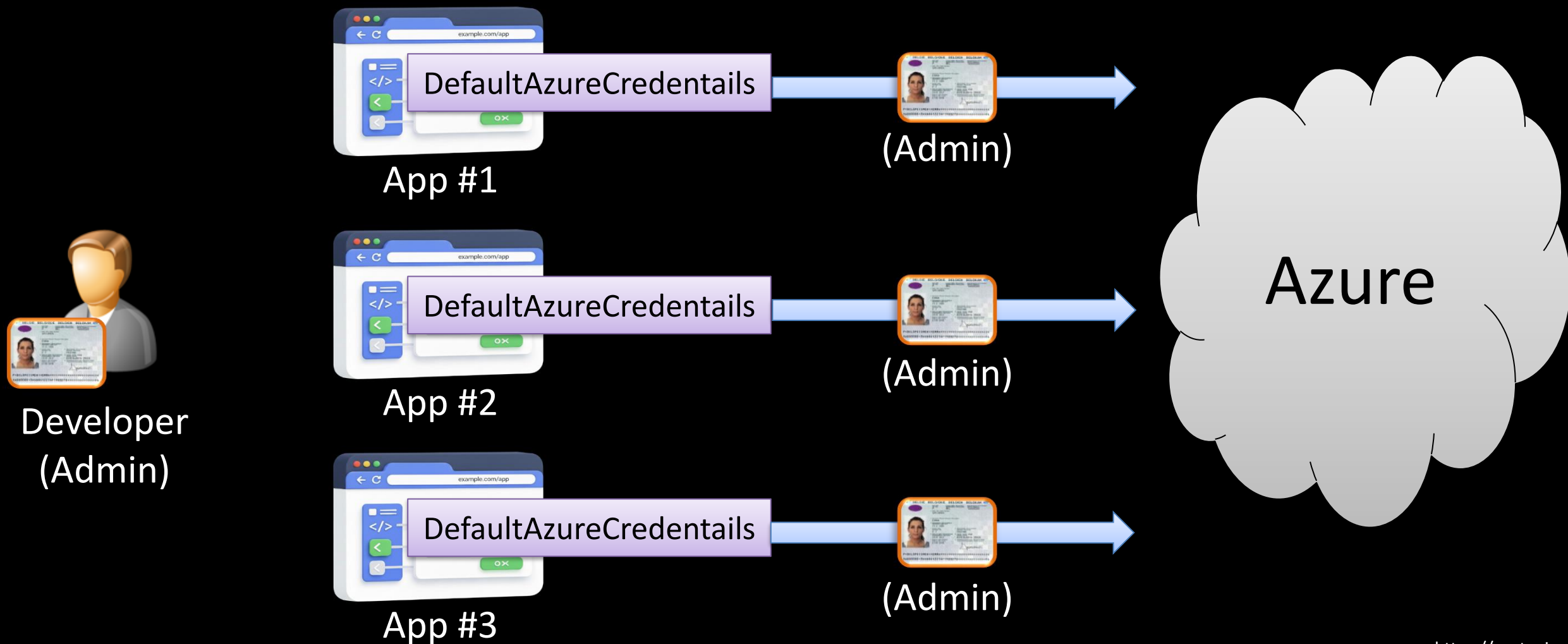
A screenshot of a web browser window titled "MainWindow". The browser's address bar is empty. The main content area has a light gray background with a large, bold, black heading "Get Secrets" centered at the top. Below the heading is a large block of text consisting of a single line of base64-encoded data. At the bottom of the page, there are two input fields. The first field contains the text "Server=mysqlserver;Database=mydb;User Id=myuser;Password=mypassword;". The second field contains the text "MyApiKey-1234567890abcdef".

# Local Development

## Should We Use DefaultAzureCredentials For Local Dev?

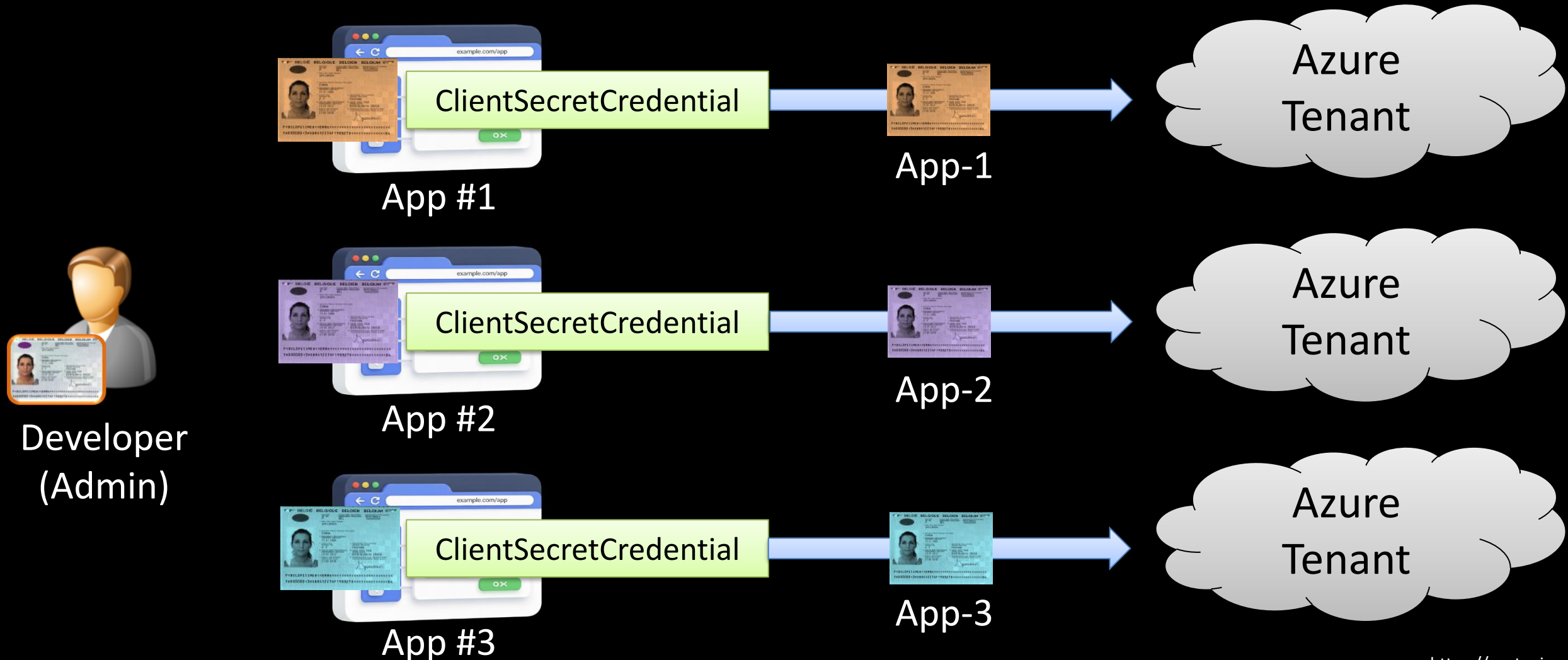


## No, It Hides Authentication Issues in Production



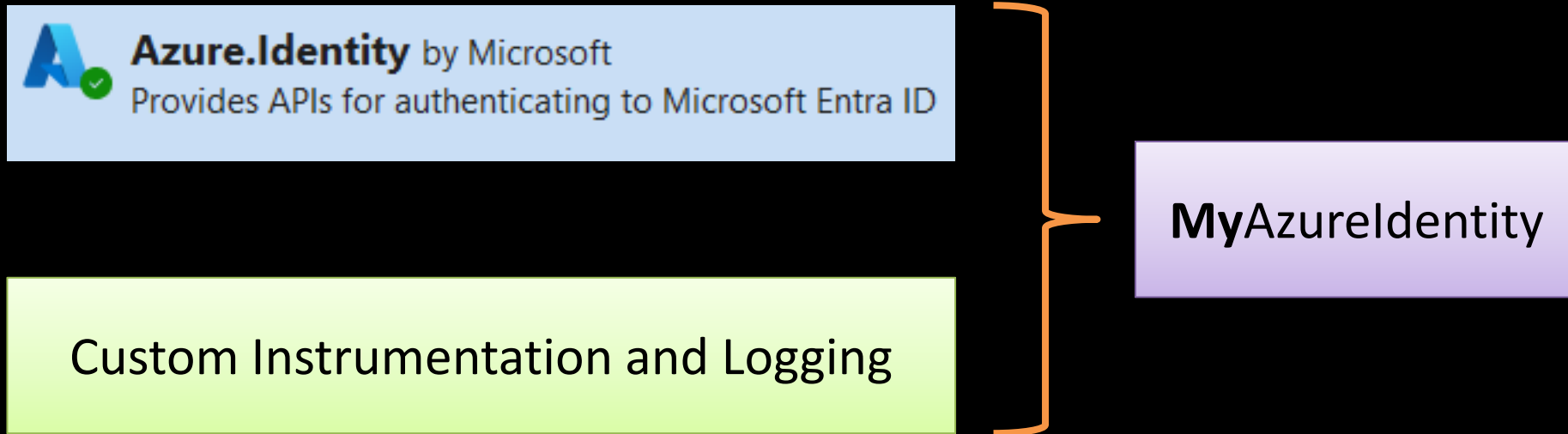


## Better: Give Each App Its Own Identity



# Frequently Asked Questions

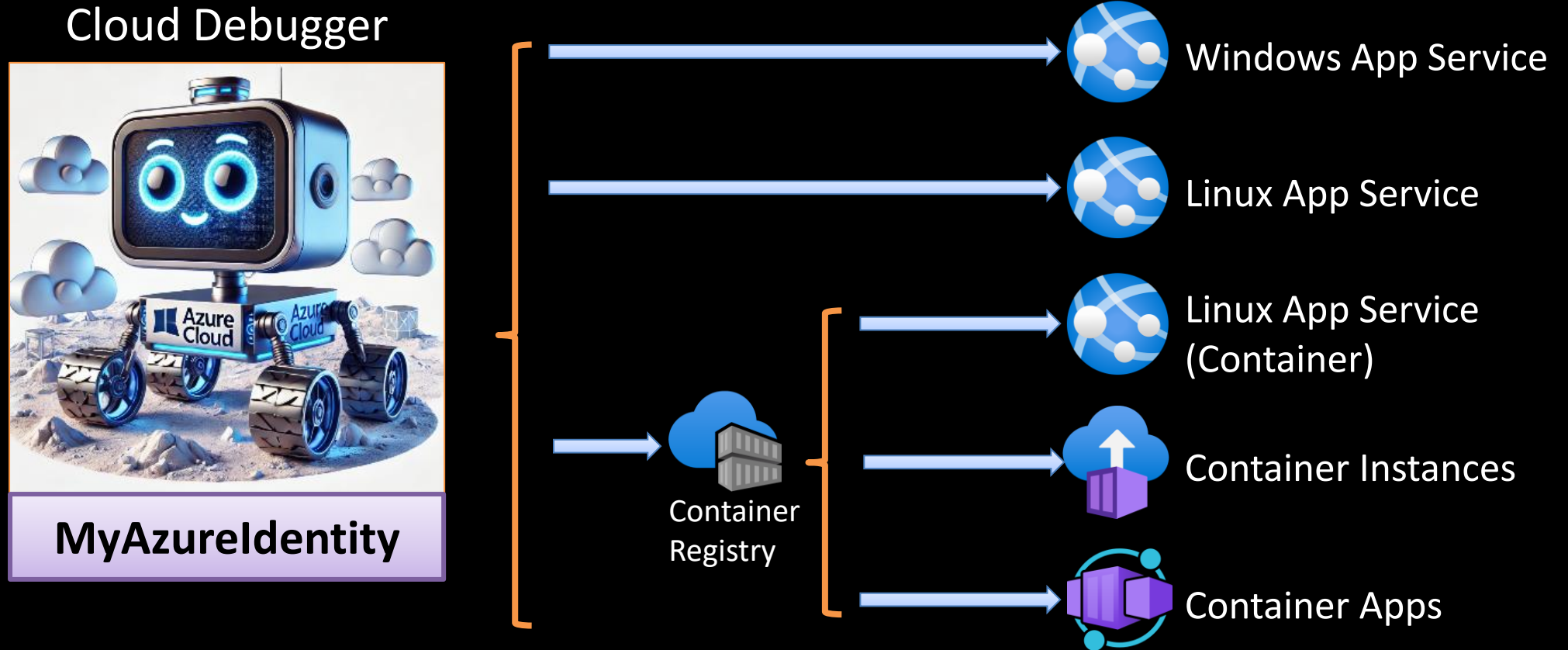
## How Can We Inspect The Internals In Even More Detail?



**Warning:** Only for learning purposes!

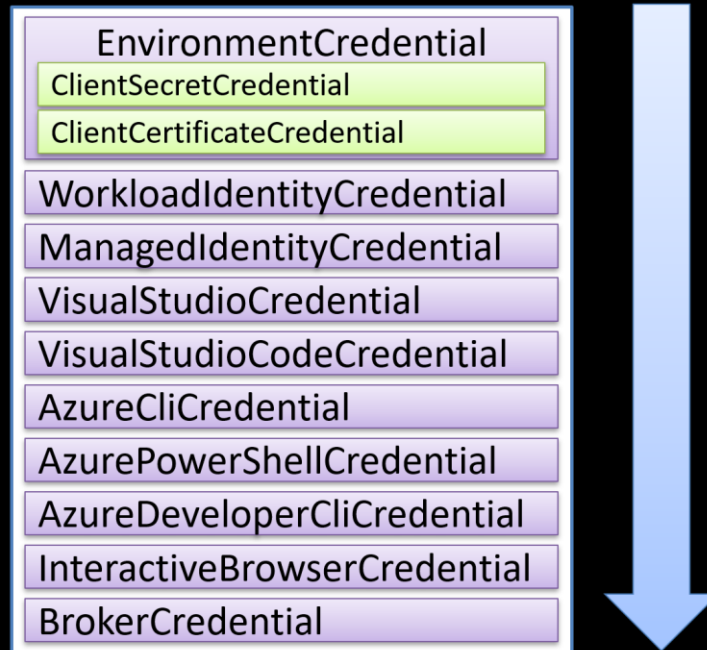


# Exploring The Internals Of Token Credentials



# Advanced DefaultAzureCredential

## By Default, It Will Try These Credentials



## However, This Can Be Customized!

# Advanced DefaultAzureCredential

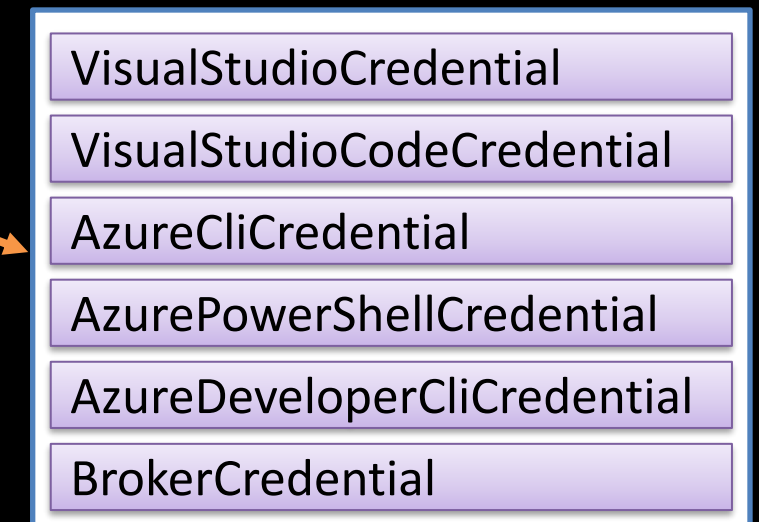
## DefaultAzureCredential



**AZURE\_TOKEN\_CREDENTIALS=prod**



**AZURE\_TOKEN\_CREDENTIALS=dev**



## A Specific Credential Can Also Be Selected

Environment Variable

`AZURE_TOKEN_CREDENTIALS=ManagedIdentityCredential`

### DefaultAzureCredential

EnvironmentCredential
ClientSecretCredential
ClientCertificateCredential
WorkloadIdentityCredential
<b>ManagedIdentityCredential</b>
VisualStudioCredential
VisualStudioCodeCredential
AzureCliCredential
AzurePowerShellCredential
AzureDeveloperCliCredential
InteractiveBrowserCredential
BrokerCredential



# QUESTIONS?



Cloud Debugger

<https://github.com/tndata/CloudDebugger>

Presentation

<https://github.com/tndataab/PublicBlogContent>

Blog: [nstenius.se](https://nstenius.se)

Work: [tn-data.se](https://tn-data.se)