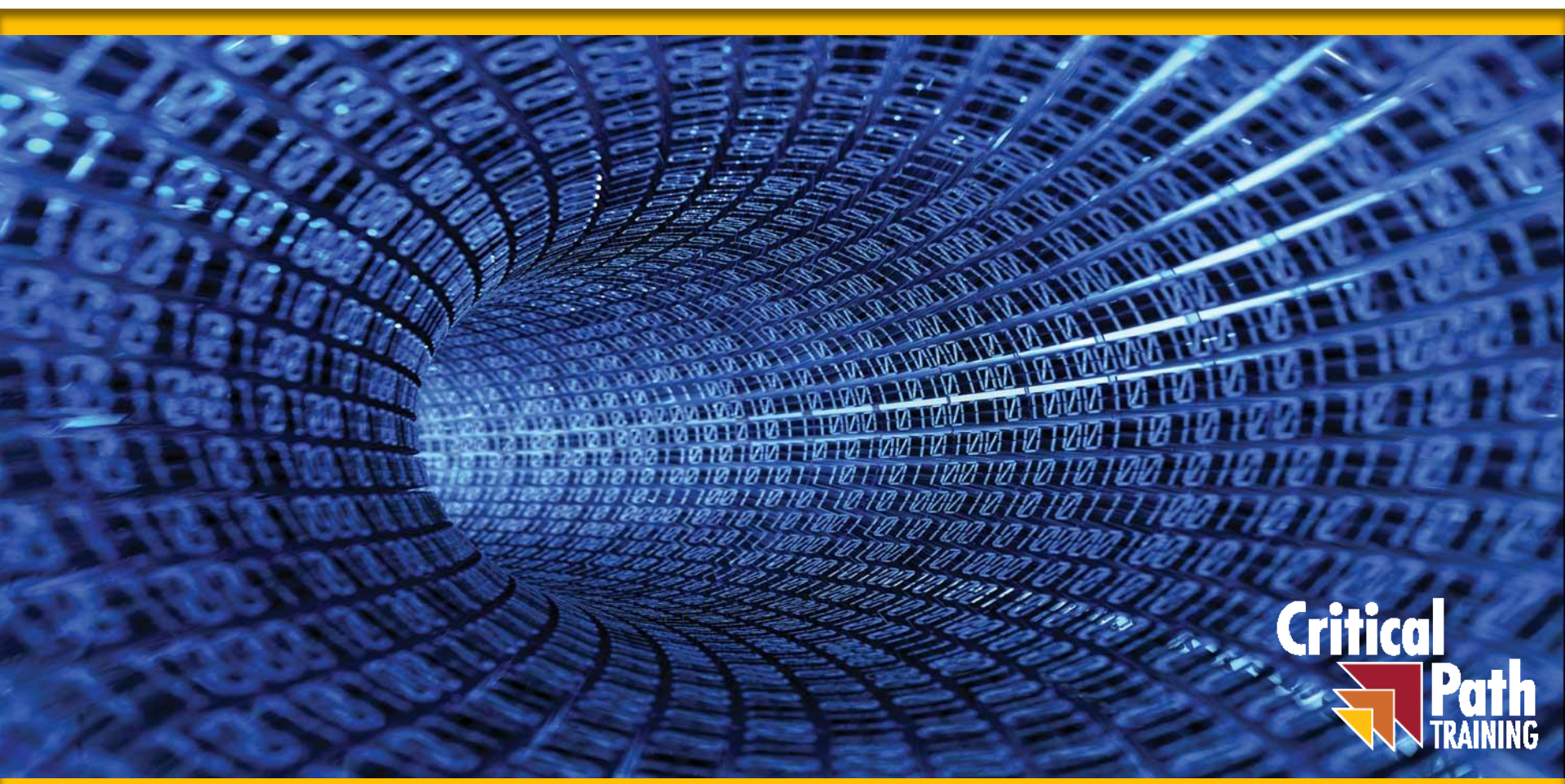


Developing Secure Applications using Azure AD



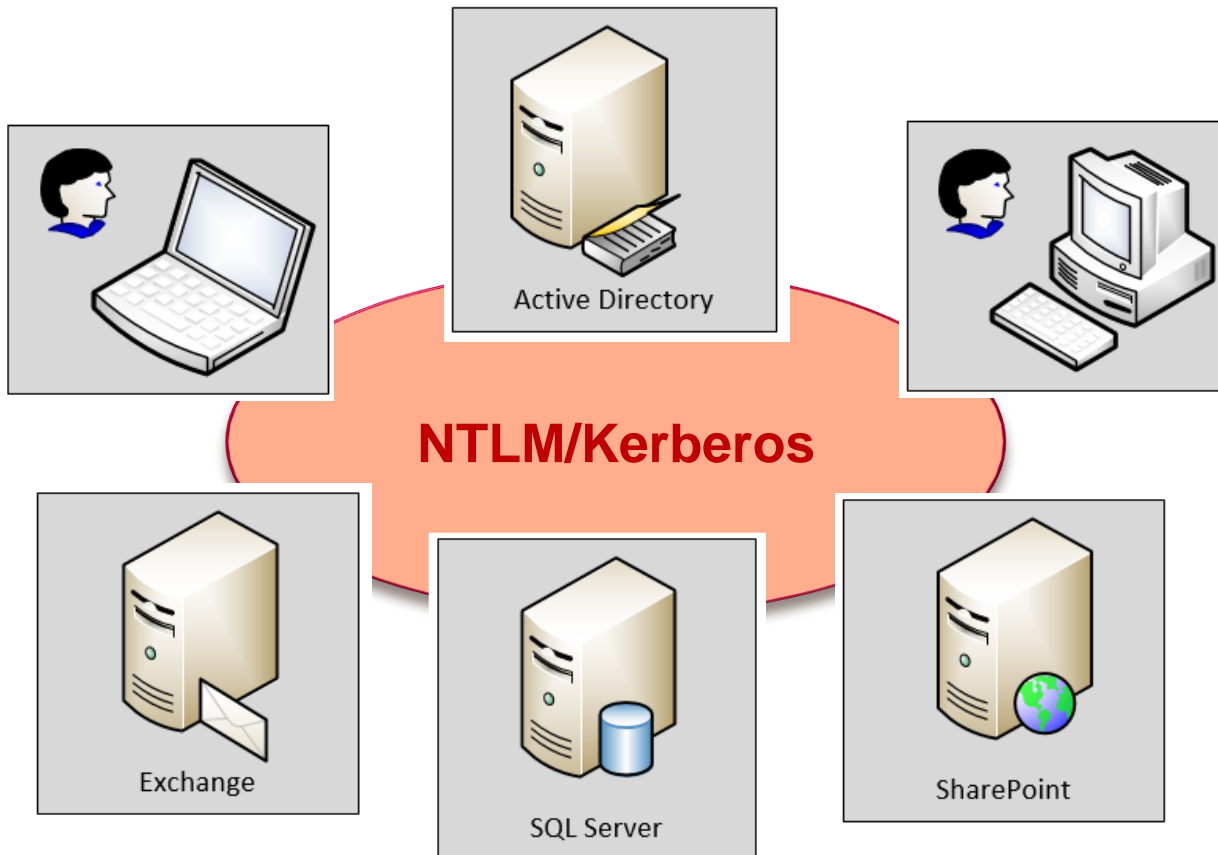
Agenda

- Understanding OAuth 2.0 and OpenID Connect
- The Role of Azure Active Directory
- Creating & Configuring Azure AD Applications
- Securing MVC Applications using ADAL and OWIN
- Securing SPAs using ADAL.js & Implicit Grant Flow

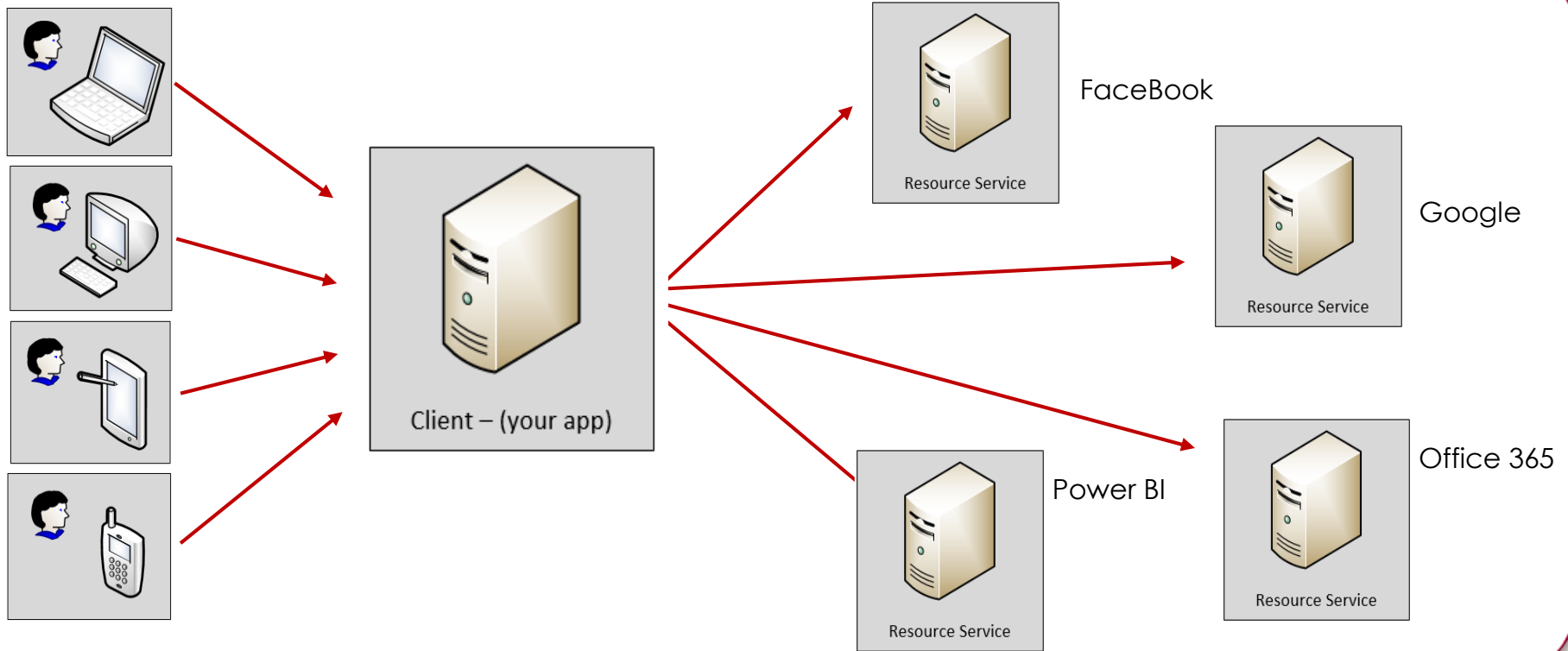


Old-school Enterprise Security

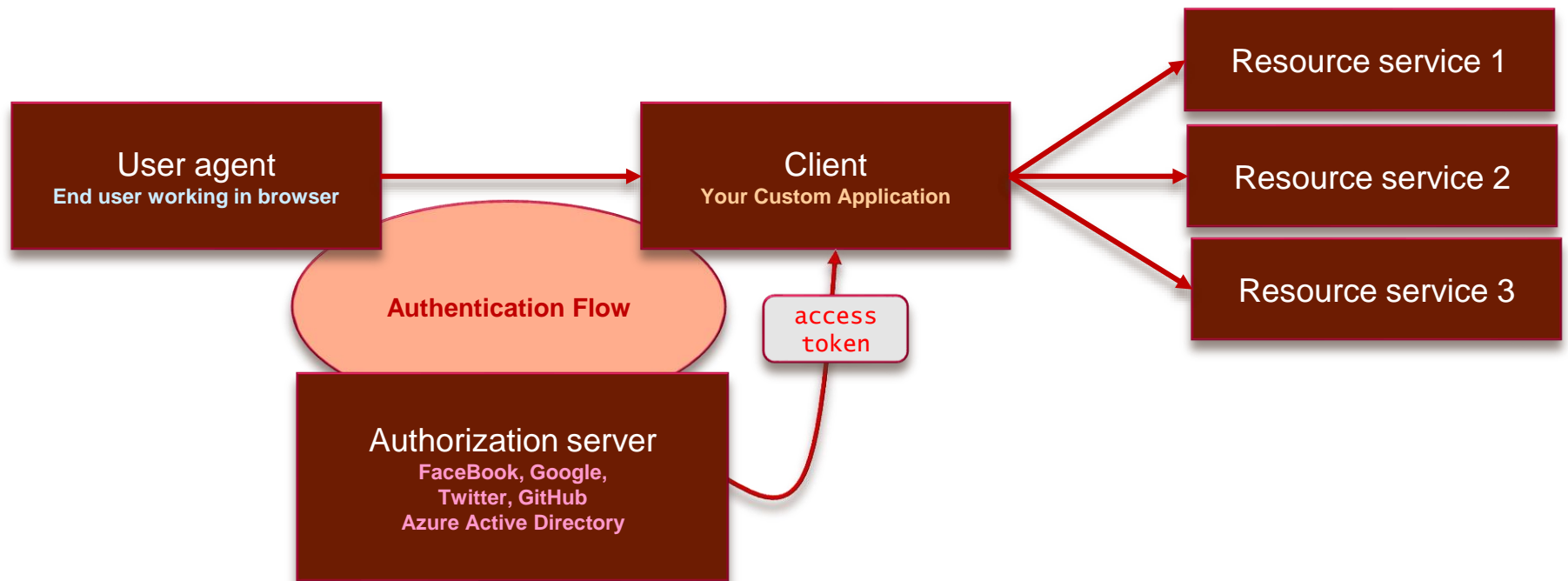
Local AD Domain: WINGTIP.COM



Internet Security



OAuth 2.0



View into an Access Token

```
{
  "aud": "https://outlook.office365.com",
  "iss": "https://sts.windows.net/f995267b-5b7d-4e65-b929-d3d3e11784f9/",
  "iat": 1427935797,
  "nbf": 1427935797,
  "exp": 1427939697,
  "ver": "1.0",
  "tid": "f995267b-5b7d-4e65-b929-d3d3e11784f9",
  "amr": ["pwd"],
  "oid": "eb679998-e8b9-40c9-b61e-4198b02b3ade",
  "upn": "TedP@sharepointconfessions.onmicrosoft.com",
  "puid": "1003BFFD85265F3D",
  "sub": "CI3lh-1kN6YD_JVKoSPjmFLTd8GyOMtgMsrvdJJdaUw",
  "given_name": "Ted",
  "family_name": "Pattison",
  "name": "Ted Pattison",
  "groups": ["a5fa8ce1-abdf-44e4-9f84-158da6ec38d0"],
  "unique_name": "TedP@sharepointconfessions.onmicrosoft.com",
  "appid": "33d561fb-59a7-4817-bddf-2117193d62e0",
  "appidacr": "1",
  "scp": "Calendars.Read Contacts.Read Contacts.Write Mail.Read Mail.Send",
  "acr": "1"
}
```



OAuth Client Registration

- Client must be registered with authorization server
 - Authorization server tracks each client with unique Client ID
 - Client should be registered with one or more Reply URLs
 - Reply URL should be fixed endpoint on Internet
 - Reply URL used to transmit security tokens to clients
 - Client registration tracks permissions and other attributes



Authentication Flows

- User Credentials Flow (*public client*)
 - Used in Native clients to obtain access code
 - Requires passing user name and password
- Authorization Code Grant Flow (*confidential client*)
 - Client first obtains authorization code then access token
 - Server-side application code never sees user's password
- Client Credentials Grant Flow (*confidential client*)
 - Authentication based on SSL certificate with public-private key pair
 - Used to obtain access token when using app-only permissions
- Implicit Grant Flow (*public client*)
 - Used in SPAs built with JavaScript and AngularJS
 - Application obtains access token w/o acquiring authorization code

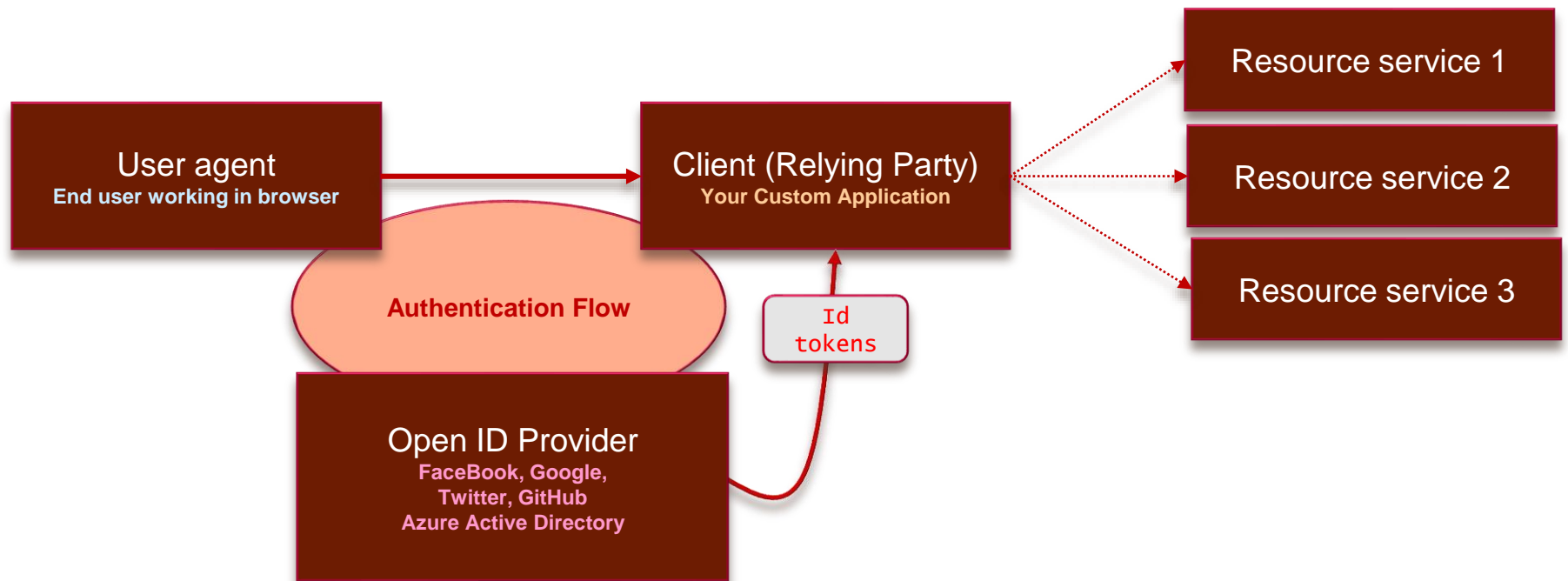


OAuth 2.0 and Authentication

- OAuth 2.0 was designed for authorization
 - Creation of access token requires authentication
 - Authorization server passes access token to client
 - Client passes access token when calling resource services
 - Access token serves as app credentials for authorization
- Access token not intended for user authentication
 - Access token not designed to carry user identity data
 - OAuth 2.0 doesn't require validation of access token
 - Naïve OAuth 2.0 implementations subject to attack



Open ID Connect



Agenda

- ✓ OAuth 2.0 and OpenID Connect
- Azure Active Directory
 - Creating Azure AD applications
 - Active Directory Authentication Library for .NET
 - Programming Web Clients



Tenants and Organizational Accounts

- Azure AD used to authenticate users and apps
 - PBI licenses are assigned to Azure AD user accounts
 - Organization owns a tenant (i.e. directory)
 - AAD tenant contains user accounts and groups
 - AAD tenant contains set of registered applications
- You must register your application with Azure AD
 - Requirement of calling to Power BI service API
 - Applications registered as Web app or Native app
 - Registered applications are assigned GUID for client ID
 - Application is configured with permissions



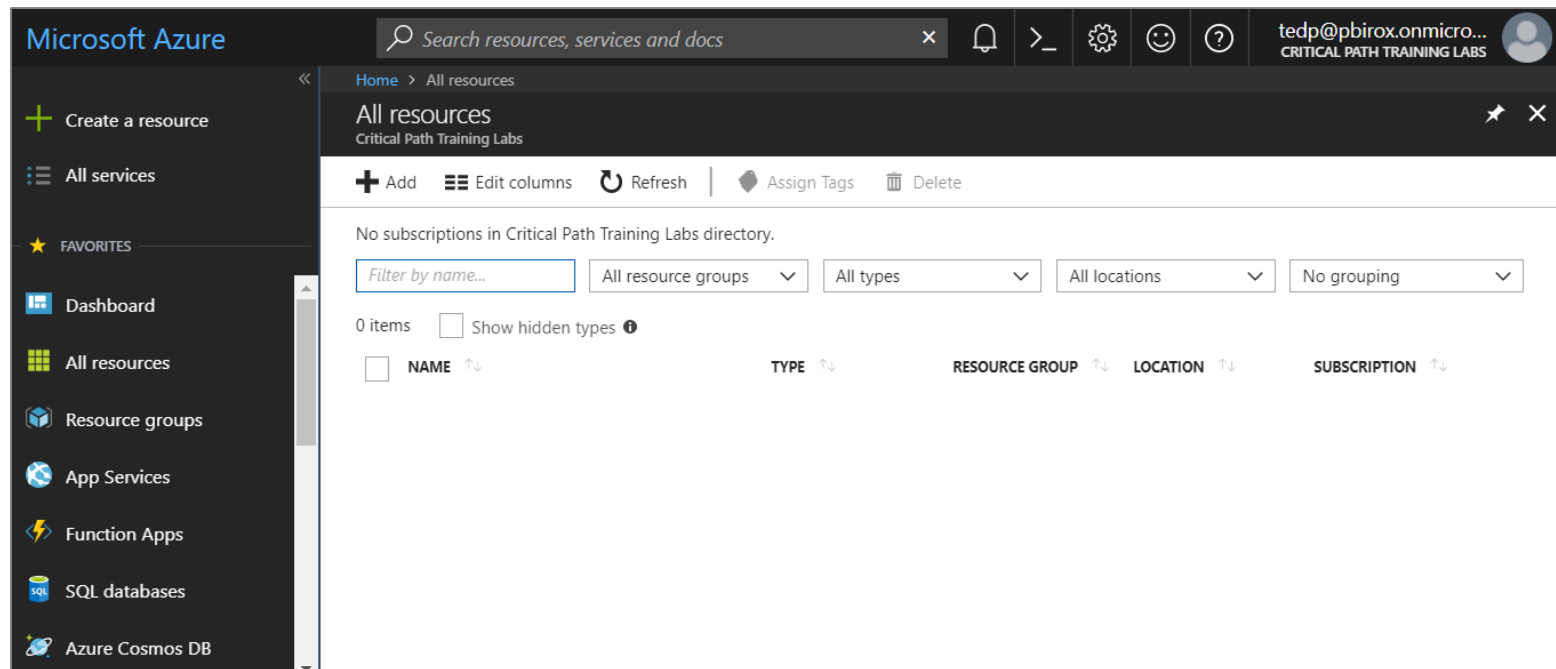
Agenda

- ✓ OAuth 2.0 and OpenID Connect
- ✓ Azure Active Directory
- Creating Azure AD applications
 - Active Directory Authentication Library for .NET
 - Programming Web Clients



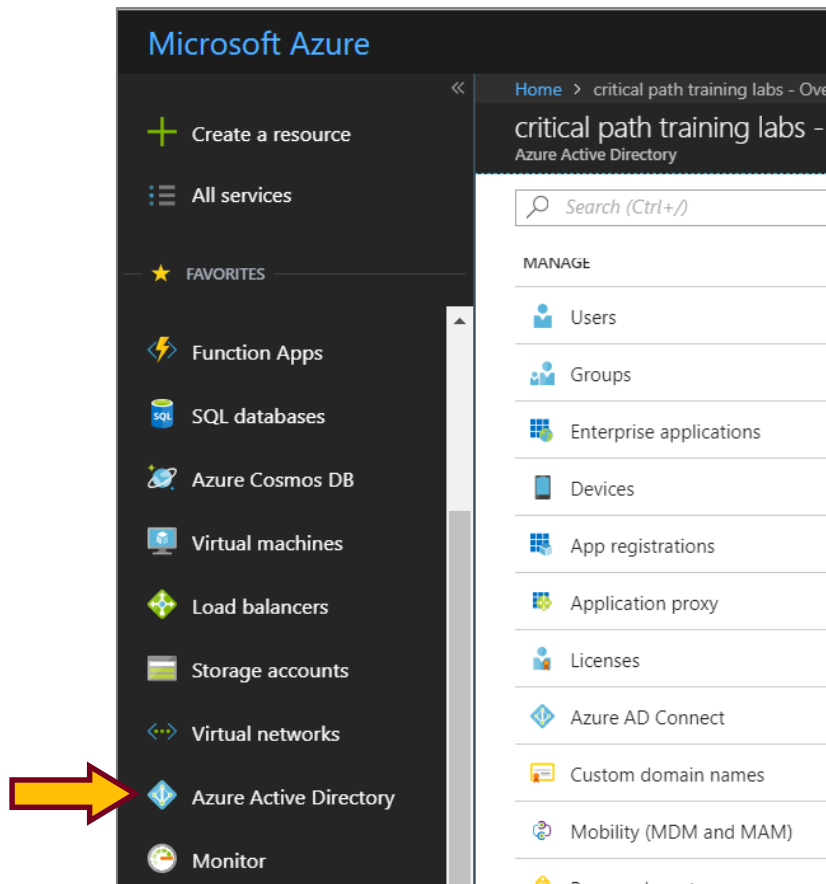
The Azure Portal

- Azure portal allows to create application
 - Azure Portal accessible at <https://portal.azure.com>
 - Azure subscription required to create resources (e.g. VMs)
 - No Azure subscription required to manage users or applications



Azure Active Directory

- Azure portal access to Access Azure Active Directory
 - Provides ability to configure users, groups and application



Azure AD Applications

- Creating applications required for AAU authentication
 - Applications are as Native application or Web Applications
 - Application identified using GUID known as application ID
 - Application ID often referred to as client ID or app ID

Home > critical path training labs - App registrations

critical path training labs - App registrations
Azure Active Directory

Search (Ctrl+/)

Users
Groups
Enterprise applications
Devices
App registrations
Application proxy

+ New application registration Endpoints Troubleshoot

To view and manage your registrations for converged applications, please visit the [Microsoft Application Console](#).

Search by name or AppID All apps

| | DISPLAY NAME | APPLICATION TYPE | APPLICATION ID |
|----|-----------------------|------------------|--------------------------------------|
| MN | My Native Application | Native | a96d73bf-ed85-4829-bd70-904each1e933 |
| MW | My Web Application | Web app / API | 17c41b86-853a-41ca-8446-92035a3699b3 |



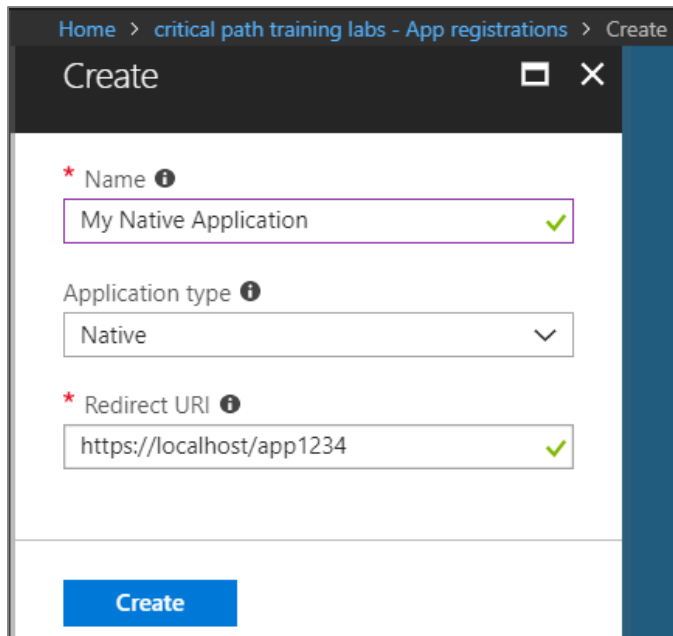
Application Types

- Single tenant application vs Multi-tenant application
 - Single tenant application intended for use within one organization
 - Multi-tenant application intended by use across organizations
 - Multi-tenant application are SaaS applications written by ISVs
- Azure AD Application Types
 - Native applications
 - Web app
 - Web app configured to allow Implicit Flow



Creating a Native Application

- Power BI supports Native applications
 - Can be used for desktop applications and Console applications
 - Used for third party embedding (known as App Owns Data model)
 - Application type should be configured as Native
 - Requires Redirect URI with unique string - not an actual URL



Home > critical path training labs - App registrations > Create

Create

* Name ⓘ
My Native Application ✓

Application type ⓘ
Native ✓

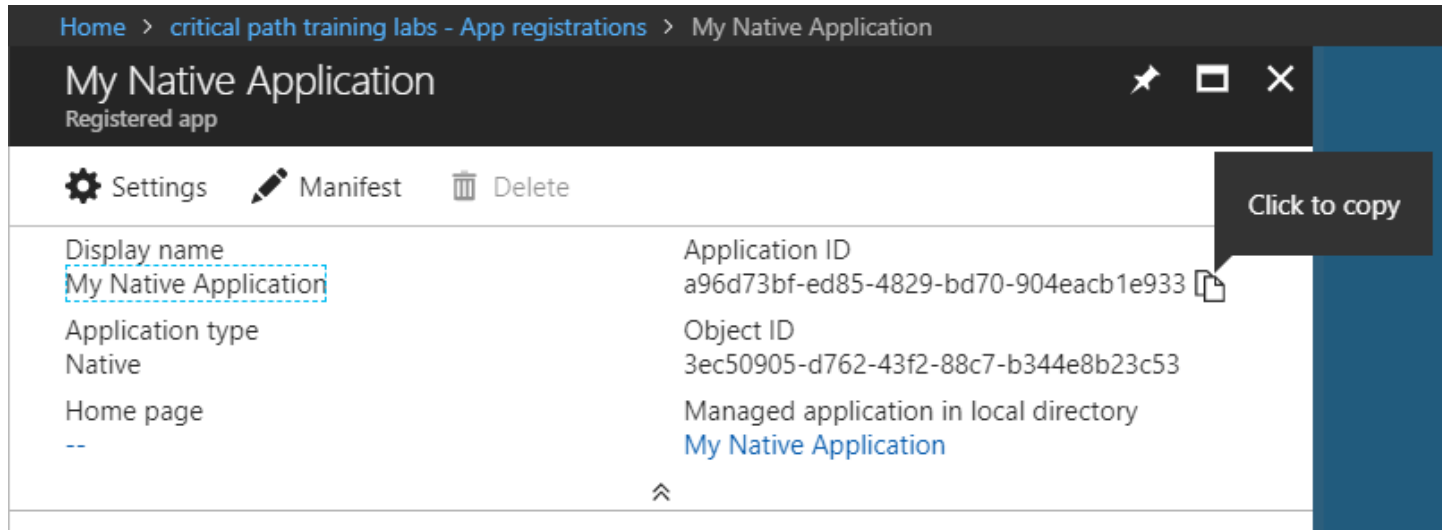
* Redirect URI ⓘ
https://localhost/app1234 ✓

Create



Copying the Application ID

- Each new application created with Application ID
 - You cannot supply your own GUID for application ID
 - Azure AD will always create this GUID
 - You can copy the application ID from the azure portal



The screenshot shows the Azure portal interface for a registered application named 'My Native Application'. The breadcrumb navigation at the top reads 'Home > critical path training labs - App registrations > My Native Application'. Below the application name, there are three tabs: 'Settings' (selected), 'Manifest', and 'Delete'. The 'Settings' tab displays the following information:

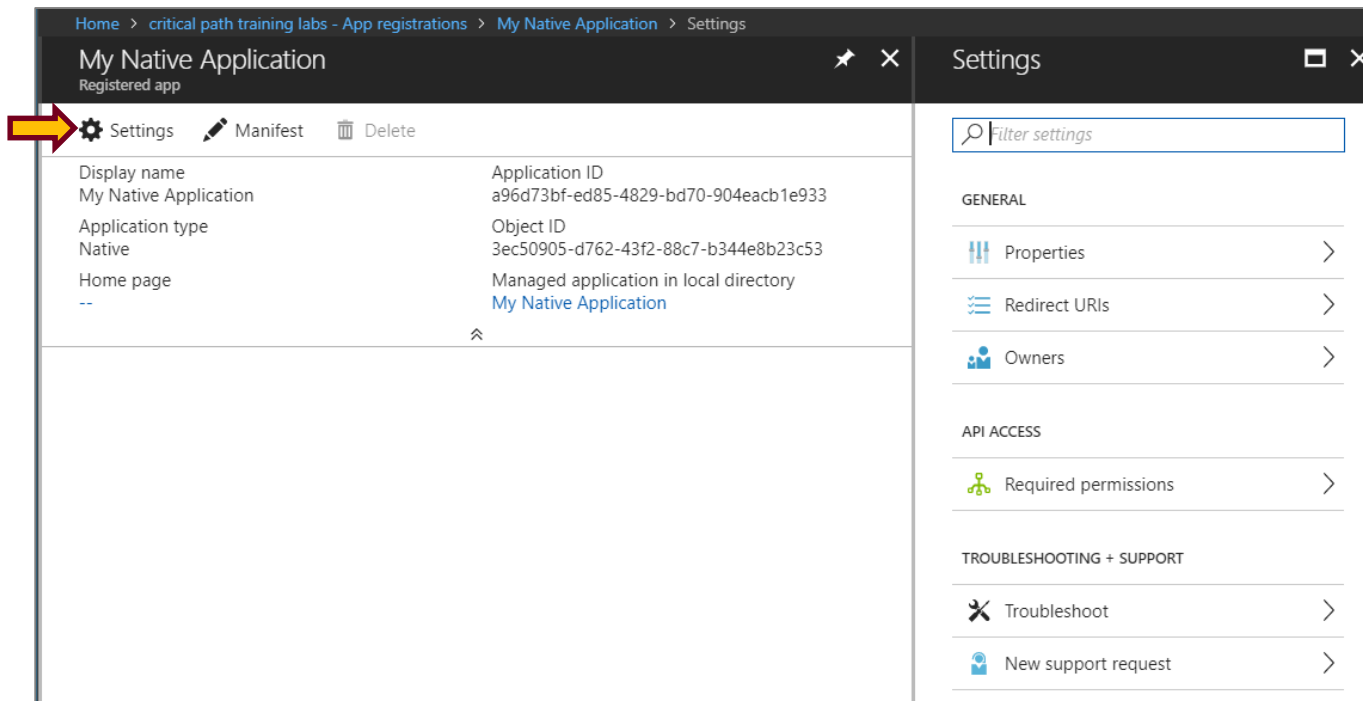
| | |
|-----------------------|--|
| Display name | Application ID |
| My Native Application | a96d73bf-ed85-4829-bd70-904eacb1e933 |
| Application type | Object ID |
| Native | 3ec50905-d762-43f2-88c7-b344e8b23c53 |
| Home page | Managed application in local directory |
| -- | My Native Application |

A tooltip with the text 'Click to copy' is positioned over the Application ID value.



Native Application Settings

- Properties
- Redirect URLs
- Owners
- Required Permissions



The screenshot displays the Azure portal interface for managing a native application. The breadcrumb navigation at the top reads: Home > critical path training labs - App registrations > My Native Application > Settings. The main header shows 'My Native Application' with a star icon and a close button, and below it, 'Registered app'. The left sidebar contains three buttons: 'Settings' (highlighted with a red arrow), 'Manifest' (pencil icon), and 'Delete' (trash icon). The main content area is divided into two panels. The left panel lists application properties: Display name (My Native Application), Application ID (a96d73bf-ed85-4829-bd70-904eacb1e933), Application type (Native), and Home page (Managed application in local directory, with a link to 'My Native Application'). The right panel, titled 'Settings', features a search bar 'Filter settings' and three sections: 'GENERAL' with links for Properties, Redirect URLs, and Owners; 'API ACCESS' with a link for Required permissions; and 'TROUBLESHOOTING + SUPPORT' with links for Troubleshoot and New support request.

| Property | Value |
|------------------|---|
| Display name | My Native Application |
| Application ID | a96d73bf-ed85-4829-bd70-904eacb1e933 |
| Application type | Native |
| Home page | Managed application in local directory My Native Application |

Settings

Filter settings

GENERAL

- Properties
- Redirect URLs
- Owners

API ACCESS

- Required permissions

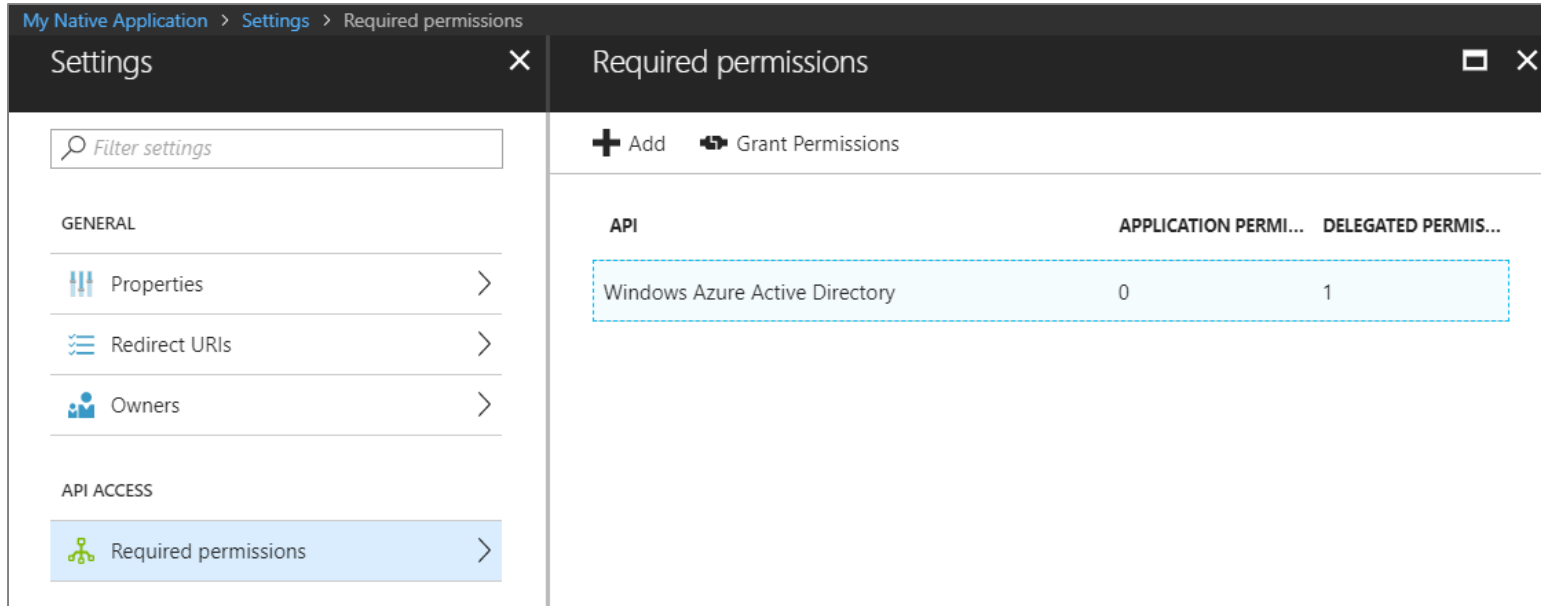
TROUBLESHOOTING + SUPPORT

- Troubleshoot
- New support request



Configuring Required Permissions

- Application configured with permissions
 - Default permissions allows user authentication – but that's it
 - To use APIs, you must assign permissions to the application



The screenshot shows the 'Required permissions' settings page for a native application. The left sidebar contains a 'Settings' menu with a search bar and two sections: 'GENERAL' (Properties, Redirect URIs, Owners) and 'API ACCESS' (Required permissions). The main area is titled 'Required permissions' and includes '+ Add' and 'Grant Permissions' buttons. Below these is a table with columns 'API', 'APPLICATION PERMI...', and 'DELEGATED PERMIS...'. A single row is visible, representing 'Windows Azure Active Directory' with values '0' and '1'.

| API | APPLICATION PERMI... | DELEGATED PERMIS... |
|--------------------------------|----------------------|---------------------|
| Windows Azure Active Directory | 0 | 1 |



Choosing APIs

- There are lots of APIs to choose from
 - Microsoft Graph
 - Office 365 SharePoint Online
 - Power BI Service

The screenshot shows the 'Add API access' dialog in the Azure portal. The breadcrumb navigation at the top reads: 'My Native Application > Settings > Required permissions > Add API access > Select an API'. The dialog is split into two panes. The left pane, titled 'Add API access', contains a numbered list of steps: '1 Select an API' (with 'Power BI Service' listed below it) and '2 Select permissions'. The right pane, titled 'Select an API', features a search bar with the placeholder text 'Search for other applications with Service Principal name'. Below the search bar is a list of available APIs: 'Windows Azure Active Directory', 'Office 365 Exchange Online', 'Microsoft Graph', 'Office 365 SharePoint Online', 'Skype for Business Online', 'Office 365 Yammer', 'Power BI Service' (which is highlighted with a dashed blue border), 'Microsoft Rights Management Services', and 'Microsoft Intune API'. At the bottom of the left pane is a 'Done' button, and at the bottom of the right pane is a 'Select' button.



Delegated Permissions vs Application Permissions

- Permissions categorized into two basic types
 - Delegated permissions are (app + user) permissions
 - Application permissions are app-only permissions (far more powerful)
 - Not all application types and APIs support application permissions
 - Power BI Service API does not yet support application permissions
- Example permissions for Office 365 SharePoint Online
 - Some delegated permissions requires administrative permissions

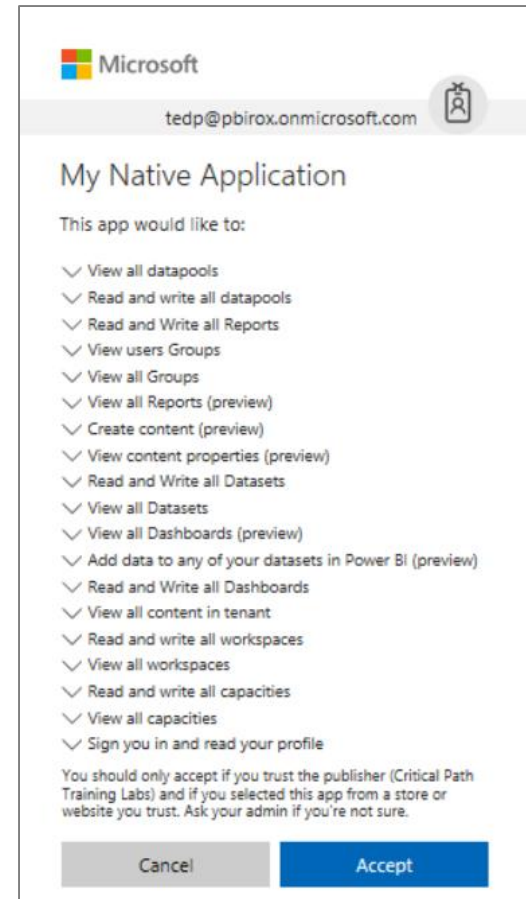
| <input type="checkbox"/> DELEGATED PERMISSIONS | REQUIRES ADMIN |
|--|----------------|
| Run search queries as a user | ✓ Yes |
| Read user profiles | ✓ Yes |
| <input checked="" type="checkbox"/> Read user files | ✗ No |
| Read managed metadata | ✓ Yes |
| <input checked="" type="checkbox"/> Read items in all site collections | ✗ No |
| Read and write user profiles | ✓ Yes |
| <input checked="" type="checkbox"/> Read and write user files | ✗ No |
| Read and write managed metadata | ✓ Yes |
| <input checked="" type="checkbox"/> Read and write items in all site collections | ✗ No |
| <input checked="" type="checkbox"/> Read and write items and lists in all site collections | ✗ No |
| Have full control of all site collections | ✓ Yes |

| APPLICATION PERMISSIONS | REQUIRES ADMIN |
|--|----------------|
| Read user profiles | ✓ Yes |
| Read and write user profiles | ✓ Yes |
| Read and write managed metadata | ✓ Yes |
| Read managed metadata | ✓ Yes |
| Read and write items and lists in all site collections | ✓ Yes |
| Have full control of all site collections | ✓ Yes |
| Read items in all site collections | ✓ Yes |
| Read and write items in all site collections | ✓ Yes |



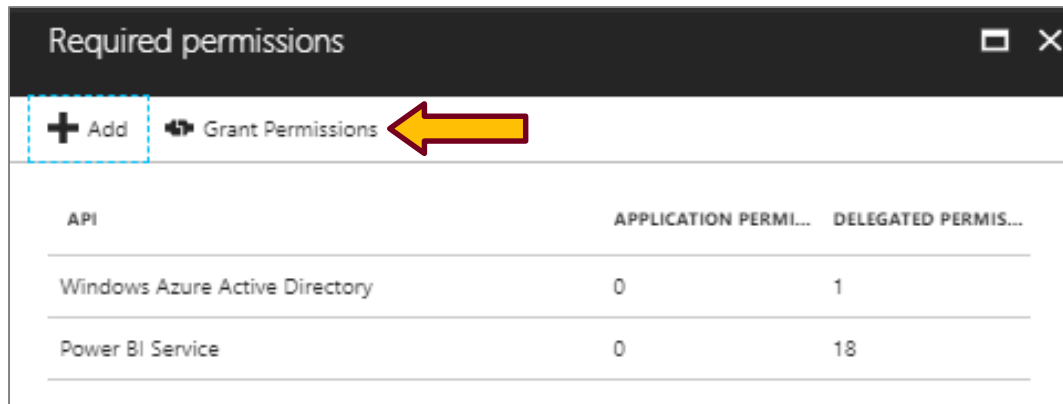
Interactive Consent for Delegated Permissions

- Users must consent to delegated permissions
 - User prompted during first log in
 - User must click Accept
 - Only occurs once for each user



Granting Delegated Permissions

- It can be helpful to Grant Permissions in Azure portal
 - Prevents the need for interactive granting of application by user
 - Might be required when authenticating in non-interactive fashion



| Required permissions | | |
|--|----------------------|---------------------|
| <div>+ Add 🔑 Grant Permissions</div> | | |
| API | APPLICATION PERMI... | DELEGATED PERMIS... |
| Windows Azure Active Directory | 0 | 1 |
| Power BI Service | 0 | 18 |





DEMO

Creating an AAD Application

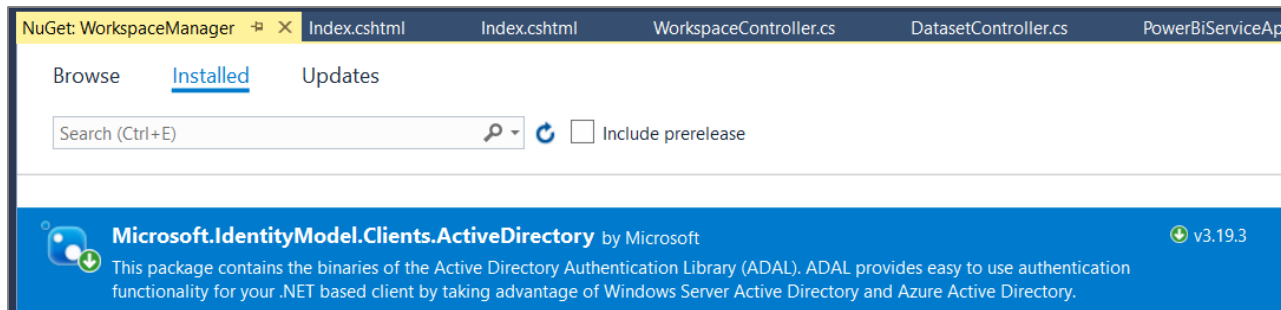
Agenda

- ✓ OAuth 2.0 and OpenID Connect
- ✓ Azure Active Directory
- ✓ Creating Azure AD applications
- Active Directory Authentication Library for .NET
- Programming Web Clients



ADAL for .NET

- Active Directory Authentication Library for .NET
 - Used in Native Clients and in Web Clients
 - Handles authentication flow behind the scenes
 - Provides caching for access tokens and refresh tokens



- ADAL .NET installs as a NuGet Package
 - Package name is **Microsoft.IdentityModel.Clients.ActiveDirectory**



Access Token Acquisition (Native Client)

- With interactive login

```
static string aadAuthorizationEndpoint = "https://login.windows.net/common/oauth2/authorize";
static string resourceUriPowerBi = "https://analysis.windows.net/powerbi/api";
static string urlPowerBiRestApiRoot = "https://api.powerbi.com/";

public const string clientId = "315e87eb-a6a0-4886-9b20-9f7ecdaca888";
public const string redirectUrl = "https://localhost/app1234";

static string GetAccessToken() {
    // create new authentication context
    var authenticationContext = new AuthenticationContext(aadAuthorizationEndpoint);

    // use authentication context to trigger user sign-in and return access token
    var userAuthnResult = authenticationContext.AcquireTokenAsync(resourceUriPowerBi,
                                                                clientId,
                                                                new Uri(redirectUrl),
                                                                new PlatformParameters(PromptBehavior.Auto)).Result;

    // return access token to caller
    return userAuthnResult.AccessToken;
}
```

- With Direct User Credentials (non-interactive)

```
string userName = "tedp@sharepointconfessions.onmicrosoft.com";
string userPassword = "Dublin@1234";

UserPasswordCredential creds = new UserPasswordCredential(userName, userPassword);
var userAuthnResult = authenticationContext.AcquireTokenAsync(PowerBiServiceResourceUri,
                                                                ClientID,
                                                                creds).Result;

// cache access token in AccessToken field
AccessToken = userAuthnResult.AccessToken;
```





DEMO

Using ADAL in a Native Client

Agenda

- ✓ OAuth 2.0 and OpenID Connect
- ✓ Azure Active Directory
- ✓ Creating Azure AD applications
- ✓ Active Directory Authentication Library for .NET
- Programming Web Clients



Authorization Code Grant Flow

- Provides Highest Levels of Security
 - User credentials never seen by client
 - Access token passed to client with Reply URL
 - Access token not passed through user agent
- Refresh tokens used to get new access tokens
 - Access token lifetime is about 1 hour
 - Refresh token lifetime is 14 days
 - AAD supports multi-resource refresh tokens (MRRTs)



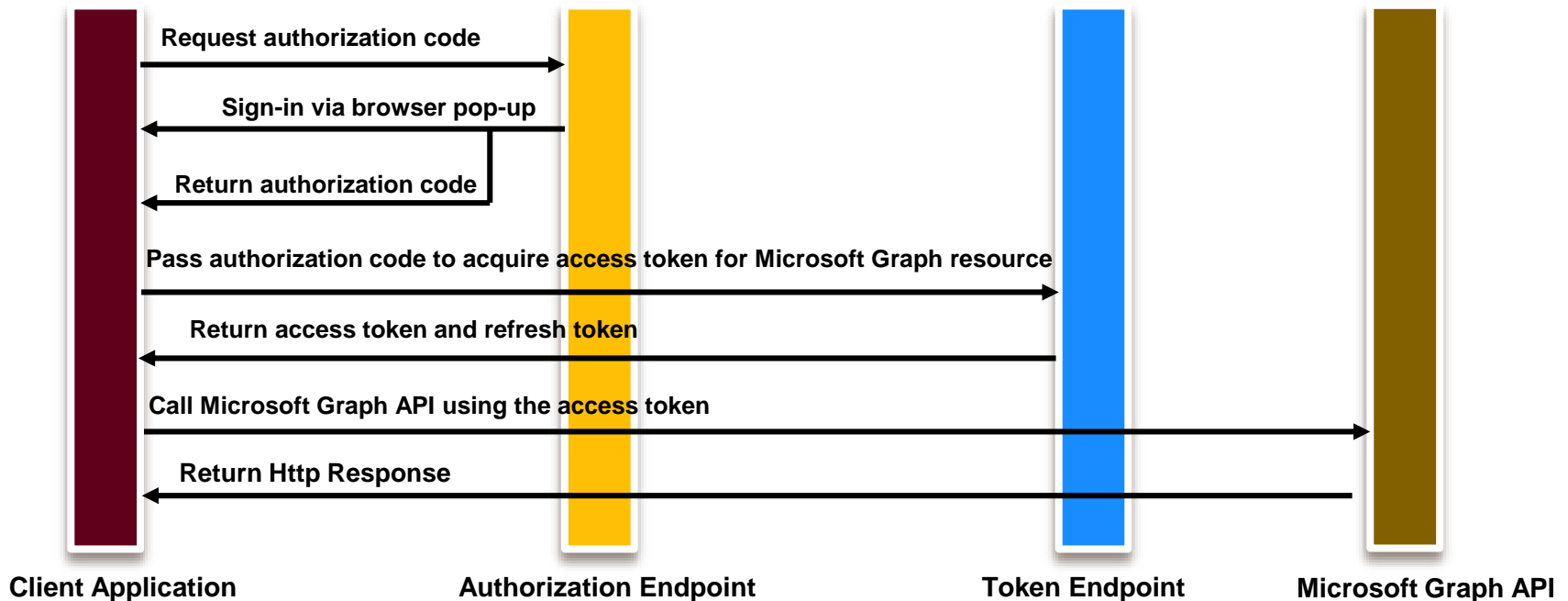
Authorization Code Grant Flow Example

- **Sign-on URL**
 - Development: <https://localhost:44300/>
 - Production: <https://www.MyDomain.com/>
- **Reply URL**
 - Development: <https://localhost:44300/AcceptDirect>
 - Production: <https://www.MyDomain.com/AcceptDirect>
- **Client ID**
 - GUID-based identifier for a specific AAD application
 - [33d561fb-59a7-4817-bddf-2117193d62e0](#)
- **Key** (aka Client Secret)
 - Key that acts as a secret password between Azure AD and application
 - [ouWdhd2LxDI0Pcu2SKlujEiQ5GmSbKRbBM24nETb5dw=](#)



Authorization Code Grant Flow

- Sequence of Requests in Authorization Code Grant Flow
 - Application redirects to AAD authorization endpoint
 - User prompted to log on at Windows logon page
 - User prompted to consent to permissions (first access)
 - AAD redirects to application with authorization code
 - Application redirects to AAD access token endpoint





DEMO

Using ADAL in a Web Client

Summary of OAuth Client Types

| | Web Client SPA | Hybrid Native Client | Web Application Client | Web Service Client |
|-----------------------------------|----------------|------------------------|------------------------|--------------------|
| Client Type | Public | Public or Confidential | Confidential | Confidential |
| Verifiable Reply URL | Yes | No | Yes | Yes |
| Authenticates Client | No | It Depends | Yes | Yes |
| Token from Authorization Endpoint | Yes | Yes | No | No |
| Access Token from URI Fragment | Yes | No | No | No |
| Token from Token Endpoint | No | Yes | Yes | Yes |
| Can use refresh tokens | No | Yes | Yes | Yes |
| Permissions | Delegated | Delegated + App | Delegated + App | Delegated + App |



Summary

- ✓ OAuth 2.0 and OpenID Connect
- ✓ Azure Active Directory
- ✓ Creating Azure AD applications
- ✓ Active Directory Authentication Library for .NET
- ✓ Programming Web Clients

