The Tenant Management Application

The TenantManagement application is a sample .NET 5 application which demonstrates how to manage service principals within a large-scale Power BI embedding environment with 1000's of customer tenants. Let's start by explaining what is meant by a tenant.

If you have worked with Azure AD, the word **"tenant"** might make you think of an Azure AD tenant. However, the concept of a tenant is different for this sample application. In this context, each tenant represents a customer for which you are embedding Power BI reports using the app-owns-data embedding model. In order to manage a multi-tenant environment, you must create a separate tenant for each customer. Provisioning a new customer tenant for Power BI embedding typically involves writing code to create a Power BI workspace, import a PBIX file, patch datasource credentials and start a dataset refresh operation.

The problem that **TenantManagement** application addresses is a Power BI Service limitation which restricts users and service principals from being a member of more than 1000 workspaces. If you are implementing app-owns-data embedding in an application which uses a single service principal, Microsoft will only support you in creating up to 1000 workspaces.

The **TenantManagement** application demonstrates how to work around the 1000 workspace limitation by implementing a service principal pooling scheme. Here is how it works. Each service principal can support up to 1000 workspaces. Therefore, creating a service principal pool of 10 service principals makes it possible to create and manage 10,000 customer tenant workspaces in a fashion that is supported by Microsoft.

In addition to implementing a service principal pooling scheme, the **TenantManagement** application also demonstrates how to create and manage a separate service principal for each customer tenant workspace. An application design which maintains a one-to-one relationship between service principals and customer tenant workspaces is what Microsoft recommends as a best practice because it provides the greatest amount of isolation especially with respect datasource credentials.

You can follow the steps in this document to set up the **TenantManagement** application for testing. To complete these steps, you will require a Microsoft 365 tenant in which you have permissions to create and manage Azure AD applications and security groups. You will also need Power BI Service administrator permissions to configure Power BI settings to give service principals to ability to access the Power BI Service API. If you do not have a Microsoft 365 environment for testing, you can create one for free by following the steps in Create a Development Environment for Power BI Embedding.

Set up your development environment

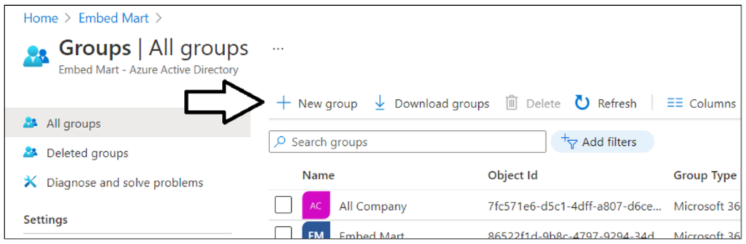
To set up the **TenantManagement** application for testing, you will need to configure a Microsoft 365 environment by completing the following tasks.

1. Create an Azure AD security group named **Power BI Apps**
2. Configure Power BI tenant-level settings for service principal access
3. Create the Azure AD Application for the **TenantManagement** Application

The following three sections will step through each of these setup tasks.

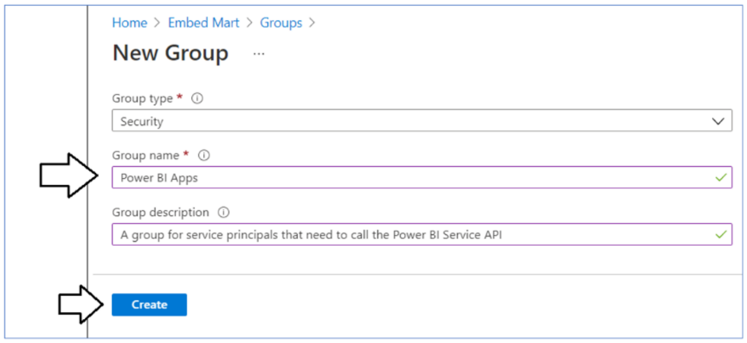
Create an Azure AD security group named Power BI Apps

Begin by navigating to the Groups management page in the Azure portal. Once you get to the **Groups** page in the Azure portal, click the **New group** link.



Name	Object Id	Group Type
AC All Company	7fc571e6-d5c1-4dff-a807-d6ce...	Microsoft 36
EM Embed Mart	86522f1d-9b8c-4797-9294-34d	Microsoft 36

In the **New Group** dialog, Select a **Group type** of **Security** and enter a **Group name** of **Power BI Apps**. Click the **Create** button to create the new Azure AD security group.

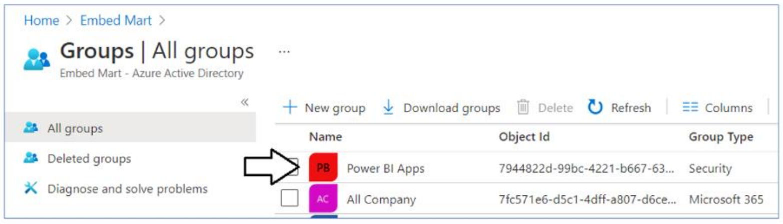


Group type *: Security

Group name *: Power BI Apps

Group description: A group for service principals that need to call the Power BI Service API

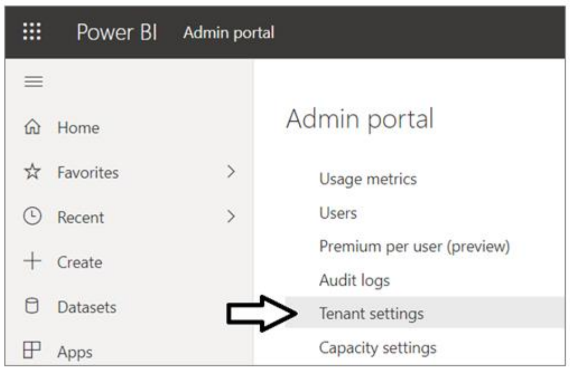
Verify that you can see the new security group named **Power BI Apps** on the Azure portal **Groups** page.



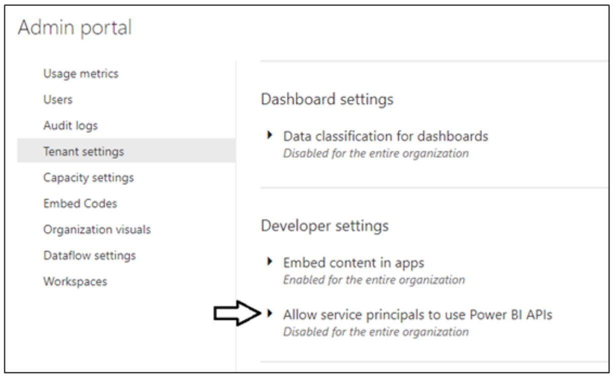
Name	Object Id	Group Type
PB Power BI Apps	7944822d-99bc-4221-b667-63...	Security
AC All Company	7fc571e6-d5c1-4dff-a807-d6ce...	Microsoft 365

Configure Power BI tenant-level settings for service principal access

Next, you need you enable a tenant-level setting for Power BI named **Allow service principals to use Power BI APIs**. Navigate to the Power BI Service admin portal at https://app.powerbi.com/admin-portal. In the Power BI Admin portal, click the **Tenant settings** link on the left.



Move down in the **Developer settings** section and expand the **Allow service principals to use Power BI APIs** section.



Dashboard settings

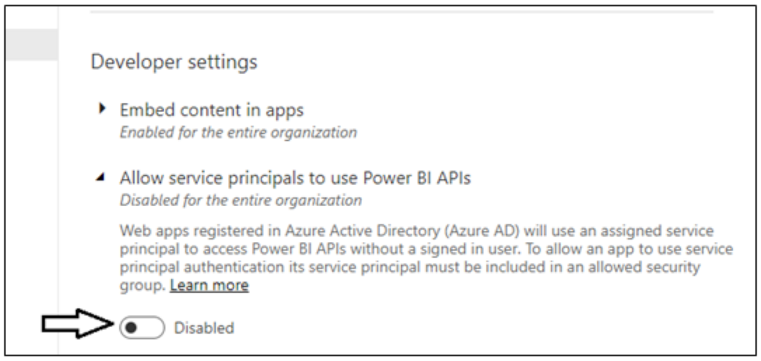
Data classification for dashboards
Disabled for the entire organization

Developer settings

Embed content in apps
Enabled for the entire organization

Allow service principals to use Power BI APIs
Disabled for the entire organization

Note that the **Allow service principals to use Power BI APIs** setting is initially set to **Disabled**.



Developer settings

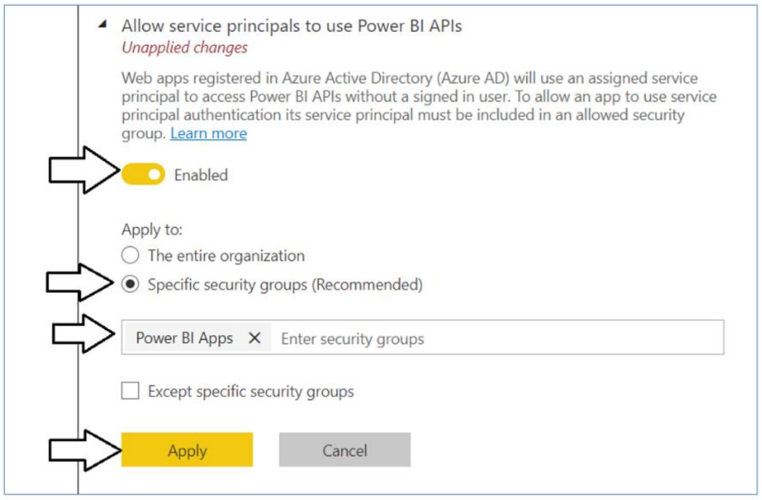
Embed content in apps
Enabled for the entire organization

Allow service principals to use Power BI APIs
Disabled for the entire organization

Web apps registered in Azure Active Directory (Azure AD) will use an assigned service principal to access Power BI APIs without a signed in user. To allow an app to use service principal authentication its service principal must be included in an allowed security group. Learn more

Disabled

Change the setting to **Enabled**. After that, set the **Apply to** setting to **Specific security groups** and add the **Power BI Apps** security group as shown in the screenshot below. Click the **Apply** button to save your configuration changes.



Allow service principals to use Power BI APIs
Unapplied changes

Web apps registered in Azure Active Directory (Azure AD) will use an assigned service principal to access Power BI APIs without a signed in user. To allow an app to use service principal authentication its service principal must be included in an allowed security group. Learn more

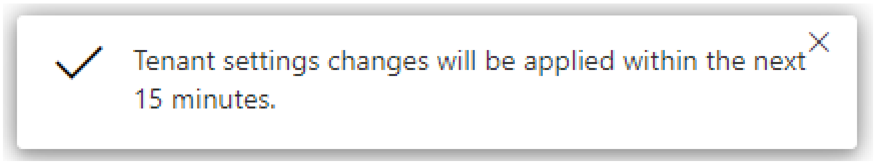
Enabled

Apply to:
- [ ] The entire organization
- [✓] Specific security groups (Recommended)

Power BI Apps ✕ Enter security groups

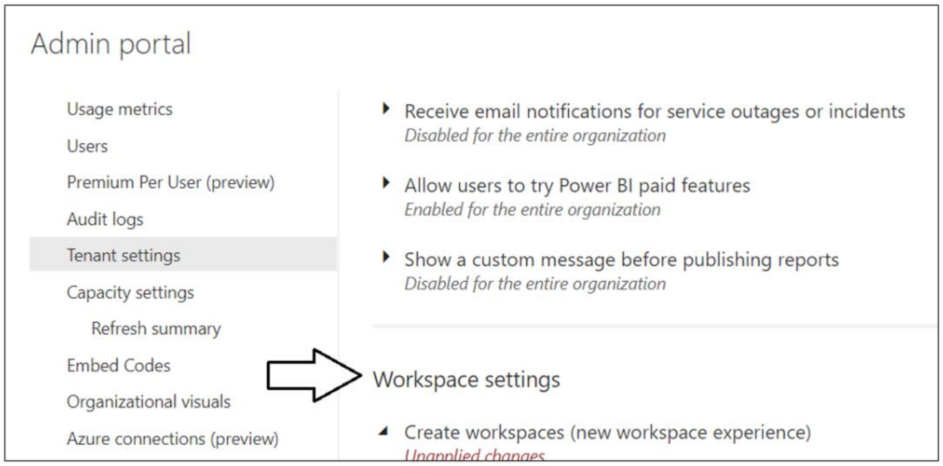
- [ ] Except specific security groups

You will see a notification indicating it might take up to 15 minutes to apply these changes to the organization.



Tenant settings changes will be applied within the next 15 minutes.

Now scroll upward in the **Tenant setting** section of the Power BI admin portal and locate **Workspace settings**.



Admin portal

Usage metrics
Users
Premium Per User (preview)
Audit logs
Tenant settings
Capacity settings
Refresh summary
Embed Codes
Organizational visuals
Azure connections (preview)

Receive email notifications for service outages or incidents
Disabled for the entire organization

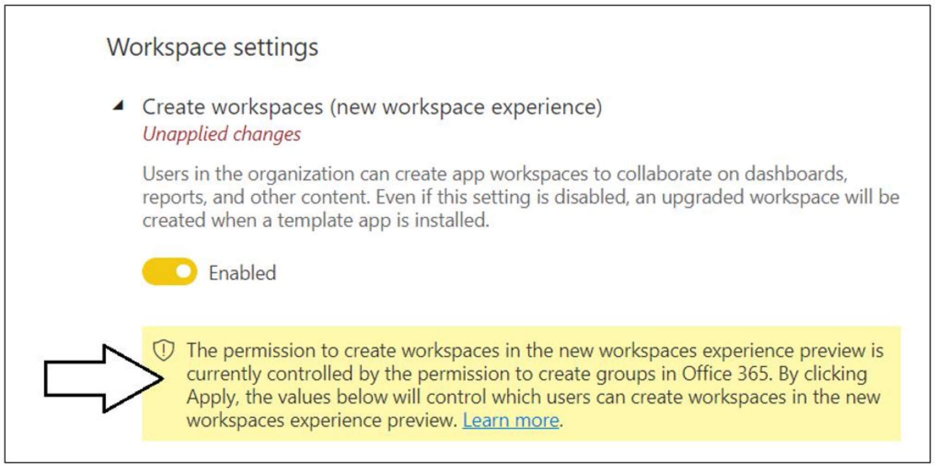
Allow users to try Power BI paid features
Enabled for the entire organization

Show a custom message before publishing reports
Disabled for the entire organization

Workspace settings

Create workspaces (new workspace experience)
Unapplied changes

Note that a new Power BI tenant has an older policy where only users who have the permissions to create Office 365 groups can create new Power BI workspaces. You must reconfigure this setting so that service principals in the **Power BI Apps** group will be able to create new workspaces.



Workspace settings

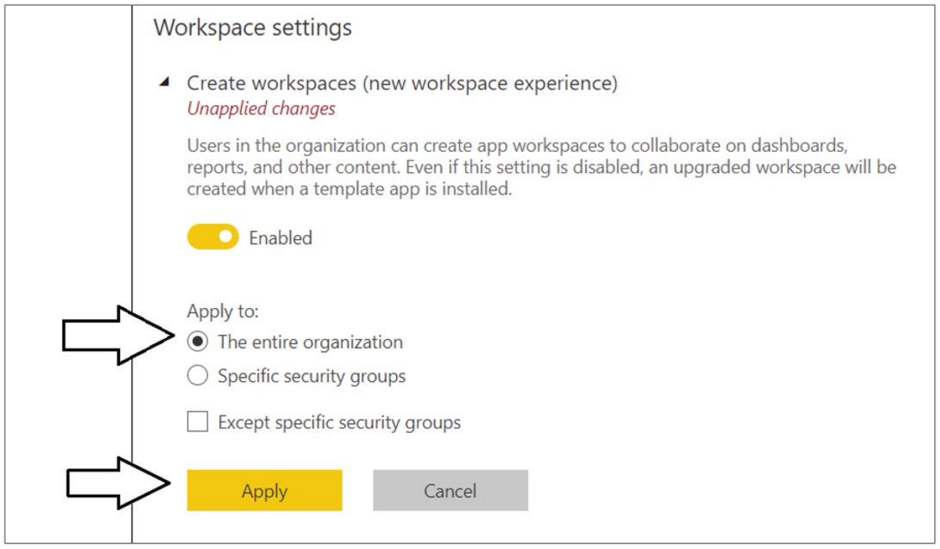
Create workspaces (new workspace experience)
Unapplied changes

Users in the organization can create app workspaces to collaborate on dashboards, reports, and other content. Even if this setting is disabled, an upgraded workspace will be created when a template app is installed.

Enabled

The permission to create workspaces in the new workspaces experience preview is currently controlled by the permission to create groups in Office 365. By clicking Apply, the values below will control which users can create workspaces in the new workspaces experience preview. Learn more.

In **Workspace settings**, set **Apply to** to **The entire organization** and click the **Apply** button to save your changes.



Workspace settings

Create workspaces (new workspace experience)
Unapplied changes

Users in the organization can create app workspaces to collaborate on dashboards, reports, and other content. Even if this setting is disabled, an upgraded workspace will be created when a template app is installed.

Enabled

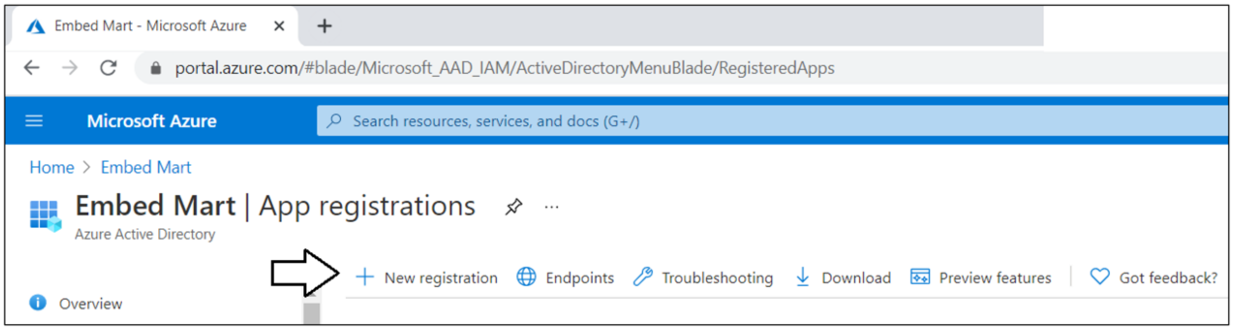
Apply to:
- ● The entire organization
- ○ Specific security groups

☐ Except specific security groups

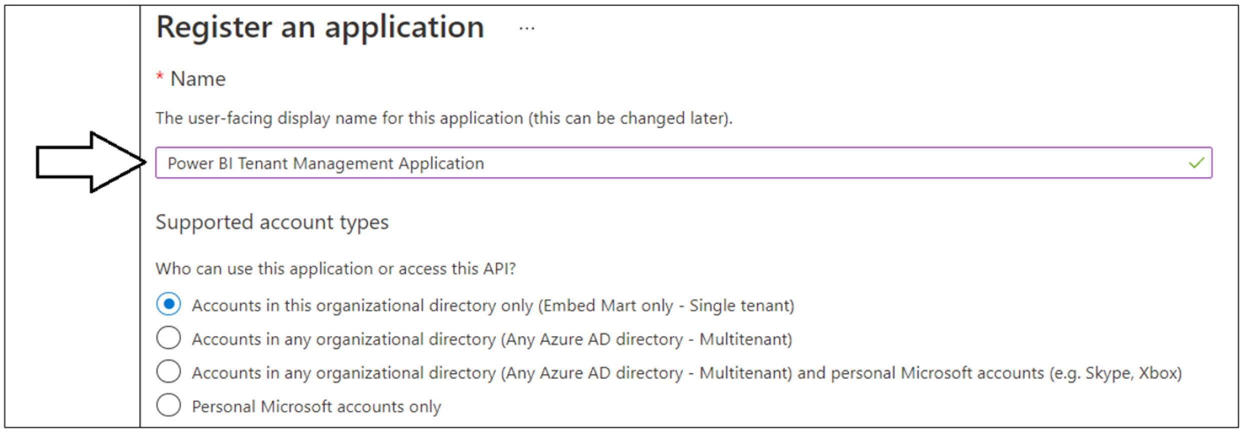
You have now completed the configuration of Power BI tenant-level settings.

Create the Azure AD Application for the TenantManagement Application

Login to the Azure portal to create the new Azure AD application. Begin by navigating to the App registration page in the Azure portal and click the **New registration** link.


Embed Mart - Microsoft Azure
portal.azure.com/#blade/Microsoft_AAD_IAM/ActiveDirectoryMenuBlade/RegisteredApps
Microsoft Azure
Search resources, services, and docs (G+/)
Home > Embed Mart
Embed Mart | App registrations
Azure Active Directory
New registration
Endpoints
Troubleshooting
Download
Preview features
Got feedback?
Overview

On the **Register an application** page, enter an application name such as **Power BI Tenant Management Application** and accept the default selection for **Supported account types** of **Accounts in this organizational directory only**.



Register an application

* Name

The user-facing display name for this application (this can be changed later).

Power BI Tenant Management Application

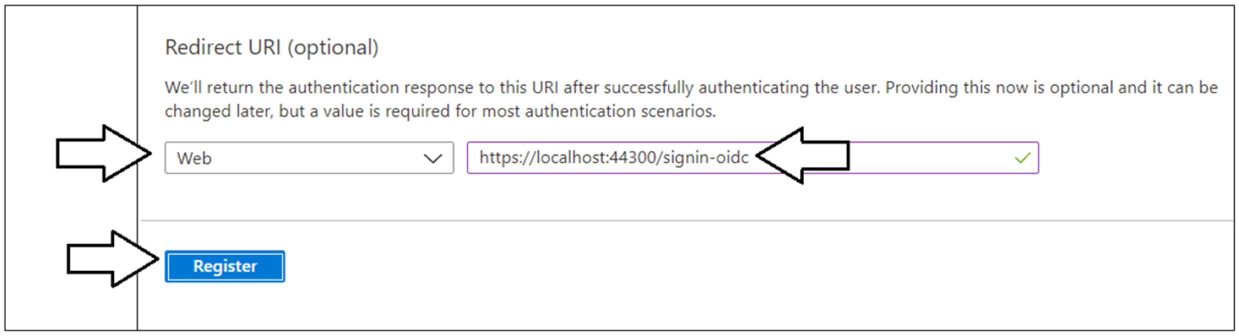
Supported account types

Who can use this application or access this API?

- [✓ Accounts in this organizational directory only (Embed Mart only - Single tenant)]
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

Complete the following steps in the **Redirect URI** section.

1. Leave the default selection of **Web** in the dropdown box
2. Enter a **Redirect URI** of **https://localhost:44300/signin-oidc**
3. Click the **Register** button to create the new Azure AD application.

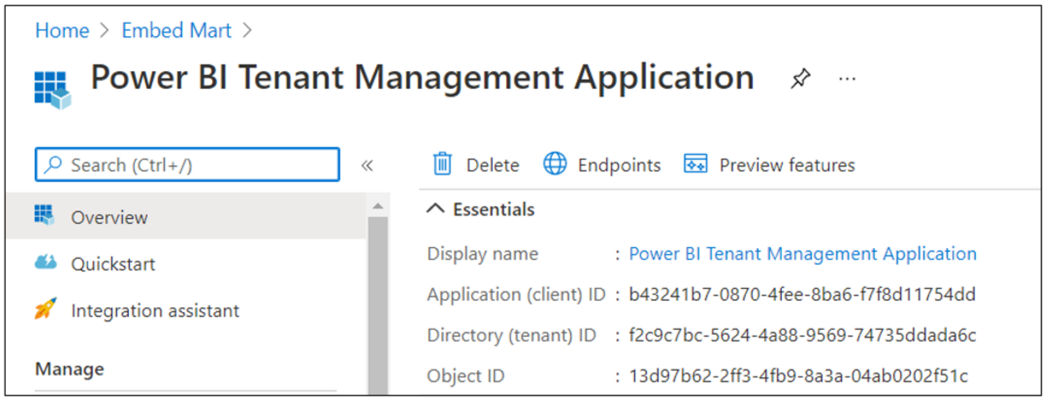


Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

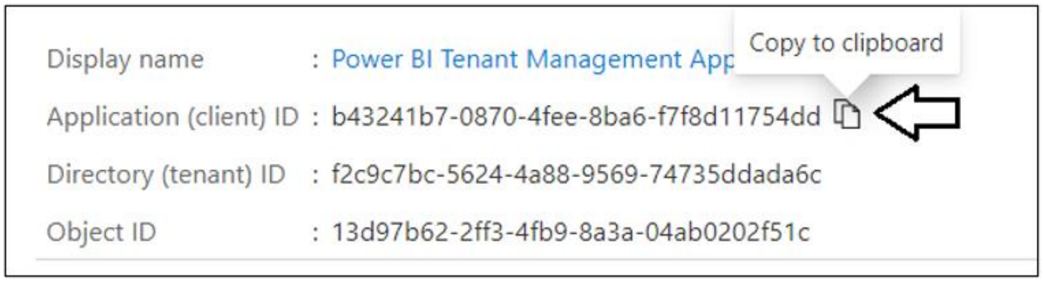
Web | https://localhost:44300/signin-oidc

After creating a new Azure AD application in the Azure portal, you should see the Azure AD application overview page which displays the **Application ID**. Note that the ***Application ID*** is often called the ***Client ID***, so don't let this confuse you. You will need to copy this Application ID and store it so you can use it later to configure the project's support for Client Credentials Flow.



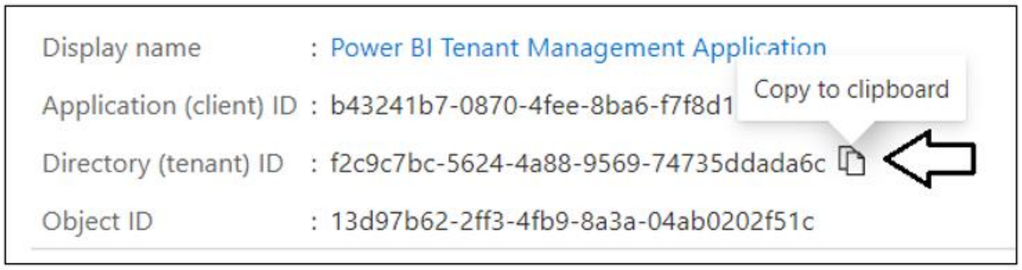
Display name	: Power BI Tenant Management Application
Application (client) ID	: b43241b7-0870-4fee-8ba6-f7f8d11754dd
Directory (tenant) ID	: f2c9c7bc-5624-4a88-9569-74735ddada6c
Object ID	: 13d97b62-2ff3-4fb9-8a3a-04ab0202f51c

Copy the **Client ID** (aka Application ID) and paste it into a text document so you can use it later in the setup process. Note that this **Client ID** value will be used by **TenantManagement** project to configure authentication with Azure AD.



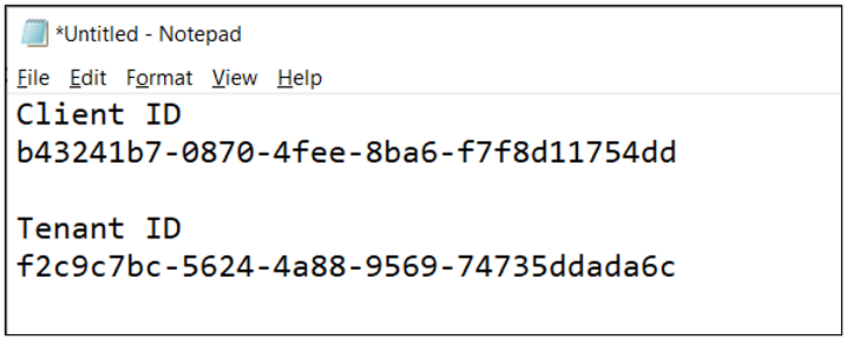
Display name	: Power BI Tenant Management App
Application (client) ID	: b43241b7-0870-4fee-8ba6-f7f8d11754dd
Directory (tenant) ID	: f2c9c7bc-5624-4a88-9569-74735ddada6c
Object ID	: 13d97b62-2ff3-4fb9-8a3a-04ab0202f51c

Next, repeat the same step by copying the **Tenant ID** and copying that into the text document as well.



Display name	: Power BI Tenant Management Application
Application (client) ID	: b43241b7-0870-4fee-8ba6-f7f8d1
Directory (tenant) ID	: f2c9c7bc-5624-4a88-9569-74735ddada6c
Object ID	: 13d97b62-2ff3-4fb9-8a3a-04ab0202f51c

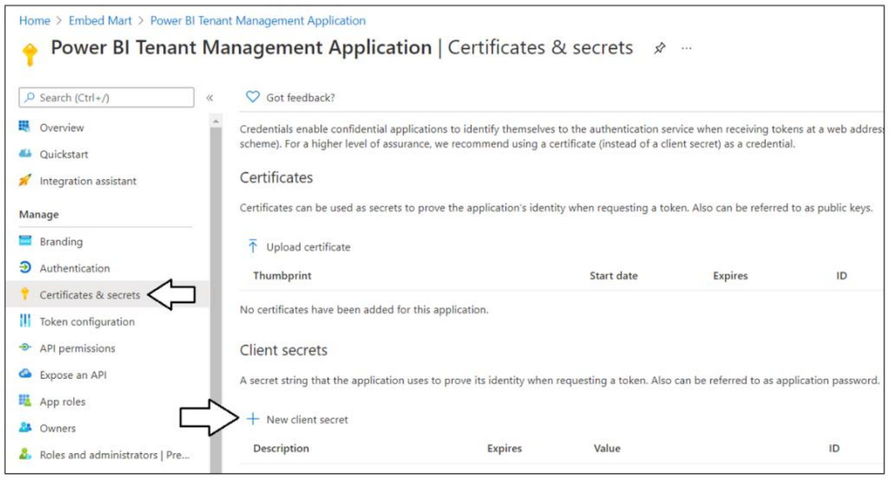
Your text document should now contain the **Client ID** and **Tenant ID** as shown in the following screenshot.



```
Client ID
b43241b7-0870-4fee-8ba6-f7f8d11754dd

Tenant ID
f2c9c7bc-5624-4a88-9569-74735ddada6c
```

Next, you need to create a Client Secret for the application. Click on the **Certificates & secrets** link in the left navigation to move to the **Certificates & secrets** page. On the **Certificates & secrets** page, click the **New client secret** button as shown in the following screenshot.



Home > Embed Mart > Power BI Tenant Management Application

Power BI Tenant Management Application | Certificates & secrets

Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addres scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

Certificates

Certificates can be used as secrets to prove the application's identity when requesting a token. Also can be referred to as public keys.

Thumbprint	Start date	Expires	ID

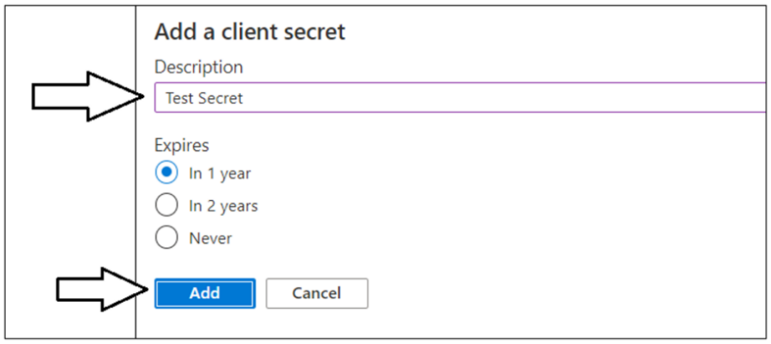
No certificates have been added for this application.

Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

Description	Expires	Value	ID

In the **Add a client secret** dialog, add a text description such as **Test Secret** and then click the **Add** button to create the new Client Secret.



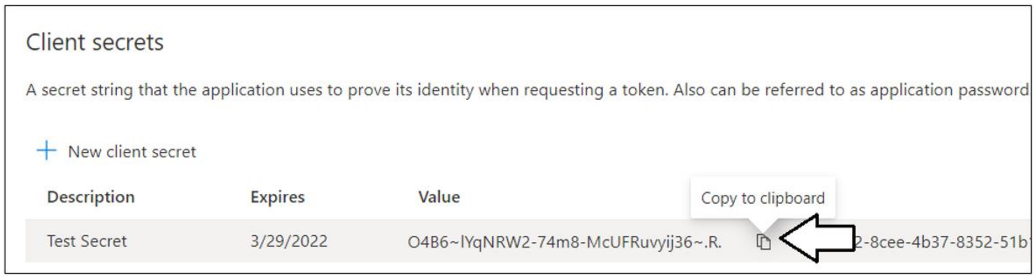
Add a client secret

Description: Test Secret

Expires

[✓ In 1 year]
In 2 years
Never

Once you have created the Client Secret, you should be able to see its **Value** in the **Client secrets** section. Click on the **Copy to clipboard** button to copy the Client Secret into the clipboard.

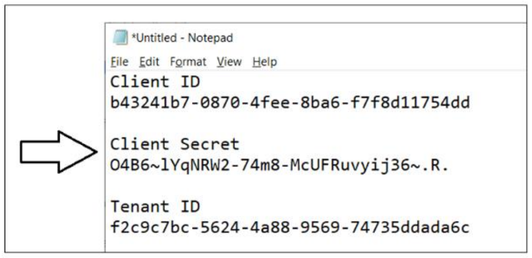


Client secrets

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password

Description	Expires	Value	
Test Secret	3/29/2022	O4B6~lYqNRW2-74m8-McUFRuvyij36~.R.	2-8cee-4b37-8352-51b

Paste the **Client Secret** into the same text document with the **Client ID** and **Tenant ID**.



*Untitled - Notepad

```
Client ID
b43241b7-0870-4fee-8ba6-f7f8d11754dd

Client Secret
O4B6~lYqNRW2-74m8-McUFRuvyij36~.R.

Tenant ID
f2c9c7bc-5624-4a88-9569-74735ddada6c
```

Test the Tenant Management project in Visual Studio 2019

In order to run and test the **TenantManagement** project on a developer workstation, you must install the .NET 5 SDK and Visual Studio 2019. While this document will walk through the steps of opening and running the **TenantManagement** project using Visual Studio 2019, you can also open and run the project using Visual Studio Code if you prefer that IDE. Here are links to download this software if you need them.

1. .NET 5 SDK – [download]
2. Visual Studio 2019 – [download]
3. Visual Studio Code – [download]

Download the Source Code

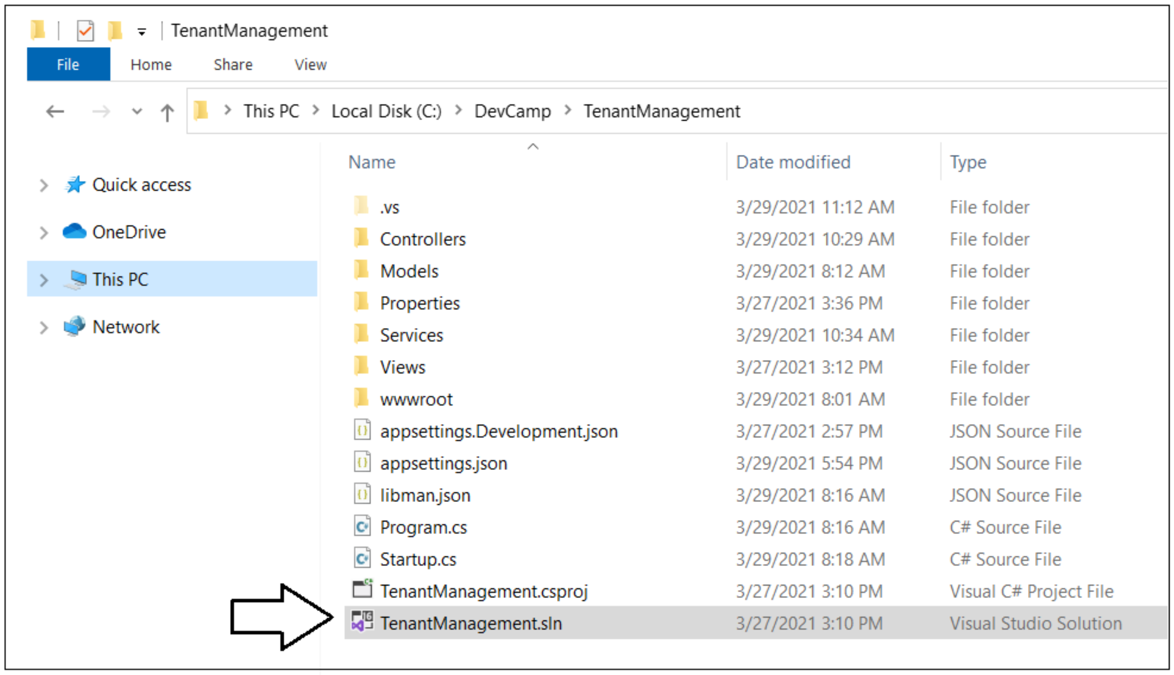
The source code for the **TenantManagement** project is maintained in a GitHub repository at the following URL.

https://github.com/PowerBiDevCamp/TenantManagement

You can download the **TenantManagement** project source files in a single ZIP archive using this link. If you are familiar with the **git** utility, you can clone the project source files to your local developer workstation using the following **git** command.

```
git clone https://github.com/PowerBiDevCamp/TenantManagement.git
```

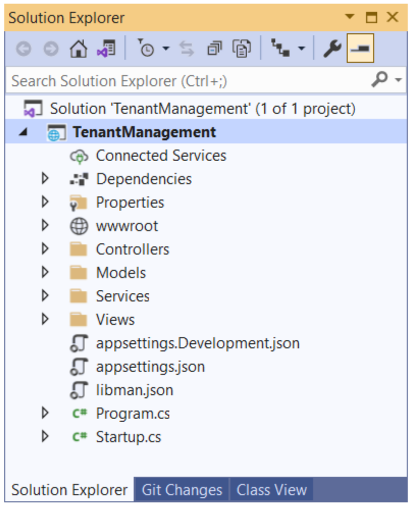
Once you have downloaded the source files for the **TenantManagement** repository to your developer workstation, you will see there is a top-level project folder named **TenantManagement** which contains several files including a solution file named **TenantManagement.sln** and a project file named **TenantManagement.csproj**.



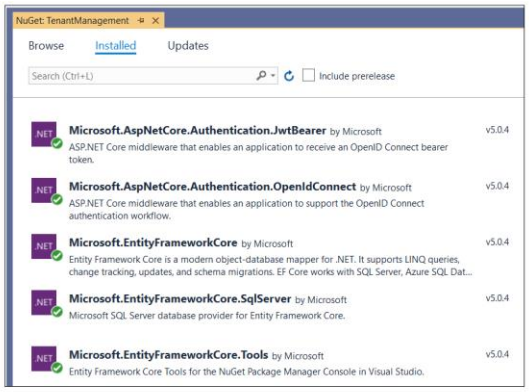
Name	Date modified	Type
.vs	3/29/2021 11:12 AM	File folder
Controllers	3/29/2021 10:29 AM	File folder
Models	3/29/2021 8:12 AM	File folder
Properties	3/27/2021 3:36 PM	File folder
Services	3/29/2021 10:34 AM	File folder
Views	3/27/2021 3:12 PM	File folder
wwwroot	3/29/2021 8:01 AM	File folder
appsettings.Development.json	3/27/2021 2:57 PM	JSON Source File
appsettings.json	3/29/2021 5:54 PM	JSON Source File
libman.json	3/29/2021 8:16 AM	JSON Source File
Program.cs	3/29/2021 8:16 AM	C# Source File
Startup.cs	3/29/2021 8:18 AM	C# Source File
TenantManagement.csproj	3/27/2021 3:10 PM	Visual C# Project File
TenantManagement.sln	3/27/2021 3:10 PM	Visual Studio Solution

Open the Project in Visual Studio 2019

Launch Visual Studio 2019 and use the **File > Open > Project/Solution** menu command to open the solution file named **TenantManagement.sln**. You should note that this development project has been built as a .NET 5 MVC Web Application as shown in the following screenshot.



Let's quickly review the NuGet packages that have been installed in the **TenantManagement** project. There are several NuGet packages which add Entity Framework support which make it possible to quickly create the SQL Server database associated with this project.



Microsoft.AspNetCore.Authentication.JwtBearer by Microsoft v5.0.4
ASP.NET Core middleware that enables an application to receive an OpenID Connect bearer token.

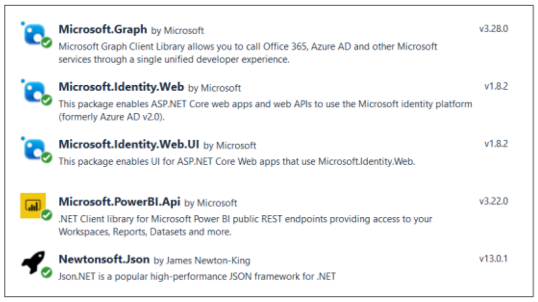
Microsoft.AspNetCore.Authentication.OpenIdConnect by Microsoft v5.0.4
ASP.NET Core middleware that enables an application to support the OpenID Connect authentication workflow.

Microsoft.EntityFrameworkCore by Microsoft v5.0.4
Entity Framework Core is a modern object-database mapper for .NET. It supports LINQ queries, change tracking, updates, and schema migrations. EF Core works with SQL Server, Azure SQL Dat...

Microsoft.EntityFrameworkCore.SqlServer by Microsoft v5.0.4
Microsoft SQL Server database provider for Entity Framework Core.

Microsoft.EntityFrameworkCore.Tools by Microsoft v5.0.4
Entity Framework Core Tools for the NuGet Package Manager Console in Visual Studio.

There are several packages included to add Azure AD authentication support including **Microsoft.Identity.Web** and **Microsoft.Identity.Web.UI**. The package named **Microsoft.Graph** has been included to support .NET programming with the Microsoft Graph API. The package named **Microsoft.PowerBI.Api** has been included to support .NET programming with the Power BI REST API.



Microsoft.Graph by Microsoft v3.28.0
Microsoft Graph Client Library allows you to call Office 365, Azure AD and other Microsoft services through a single unified developer experience.

Microsoft.Identity.Web by Microsoft v1.8.2
This package enables ASP.NET Core web apps and web APIs to use the Microsoft identity platform (formerly Azure AD v2.0).

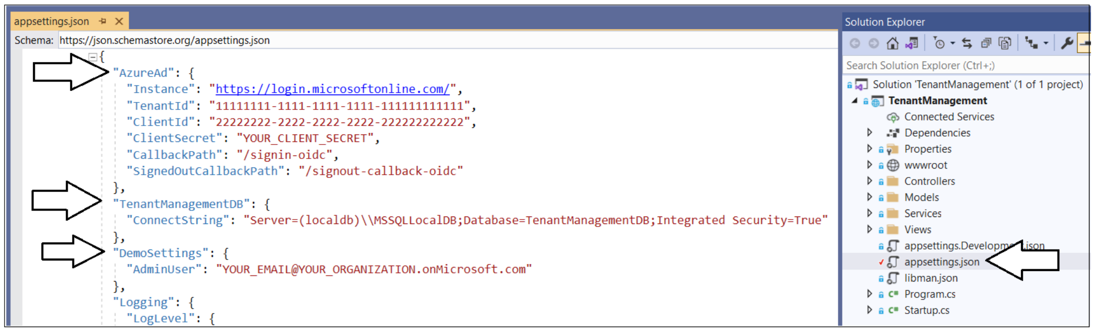
Microsoft.Identity.Web.UI by Microsoft v1.8.2
This package enables UI for ASP.NET Core Web apps that use Microsoft.Identity.Web.

Microsoft.PowerBI.Api by Microsoft v3.22.0
.NET Client library for Microsoft Power BI public REST endpoints providing access to your Workspaces, Reports, Datasets and more.

Newtonsoft.Json by James Newton-King v13.0.1
Json.NET is a popular high-performance JSON framework for .NET

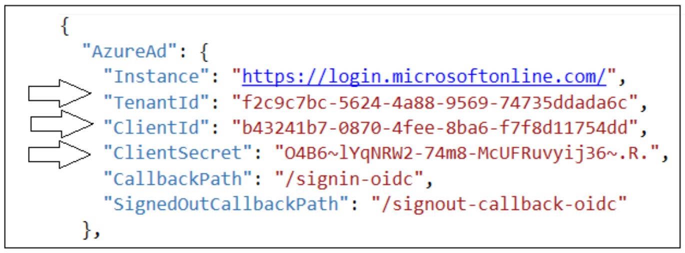
Update application settings in the appsettings.json file

Before you can run the application in the Visual Studio debugger, you must update several critical application settings in the **appsettings.json** file. Open the **appsettings.json** file and examine the JSON content inside. There is three important sections named **AzureAd**, **TenantManagementDB** and **DemoSettings**.



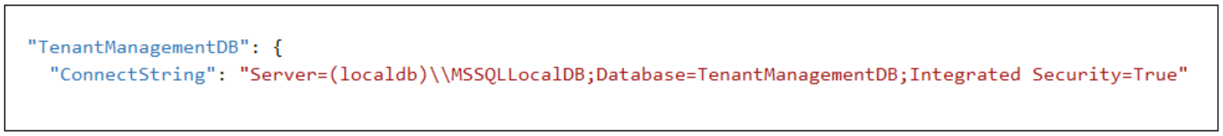
```
{
  "AzureAd": {
    "Instance": "https://login.microsoftonline.com/",
    "TenantId": "11111111-1111-1111-1111-111111111111",
    "ClientId": "22222222-2222-2222-2222-222222222222",
    "ClientSecret": "YOUR_CLIENT_SECRET",
    "CallbackPath": "/signin-oidc",
    "SignedOutCallbackPath": "/signout-callback-oidc"
  },
  "TenantManagementDB": {
    "ConnectString": "Server=(localdb)\\MSSQLLocalDB;Database=TenantManagementDB;Integrated Security=True"
  },
  "DemoSettings": {
    "AdminUser": "YOUR_EMAIL@YOUR_ORGANIZATION.onMicrosoft.com"
  },
  "Logging": {
    "LogLevel": {
```

Inside the **AzureAd** section, update the **TenantId**, **ClientId** and **ClientSecret** with the data you collected when creating the Azure AD application named **Power BI Tenant Management Application.**



```
{
  "AzureAd": {
    "Instance": "https://login.microsoftonline.com/",
    "TenantId": "f2c9c7bc-5624-4a88-9569-74735ddada6c",
    "ClientId": "b43241b7-0870-4fee-8ba6-f7f8d11754dd",
    "ClientSecret": "O4B6~lYqNRW2-74m8-McUFRuvyij36~.R.",
    "CallbackPath": "/signin-oidc",
    "SignedOutCallbackPath": "/signout-callback-oidc"
  },
```

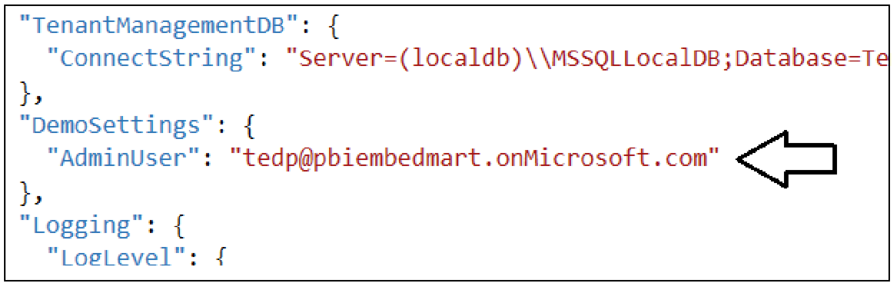
If you are using Visual Studio 2019, you shoukd be able leave the database connection string the way it is with the **Server** setting of **(localdb)\\MSSQLLocalDB**. You can change this connection string to point to a different server if you'd rather create the project database named **TenantManagementDB** in a different location.



```
"TenantManagementDB": {
  "ConnectString": "Server=(localdb)\\MSSQLLocalDB;Database=TenantManagementDB;Integrated Security=True"
```

In the **DemoSettings** section there is a property named **AdminUser**. The reason that this property exists has to with you being able to see Power BI workspaces as they are created by service principals. There is code in the **TenantManagement** application that will add the user specified by the **AdminUser** setting as a workspace admin any times it creates a new Power BI workspace. This just makes things much easier for you to see what's going on when you begin to run and test the application.

Update the **AdminUser** setting with your Azure AD account name so that you will be able to see all the Power BI workspaces created by this application.

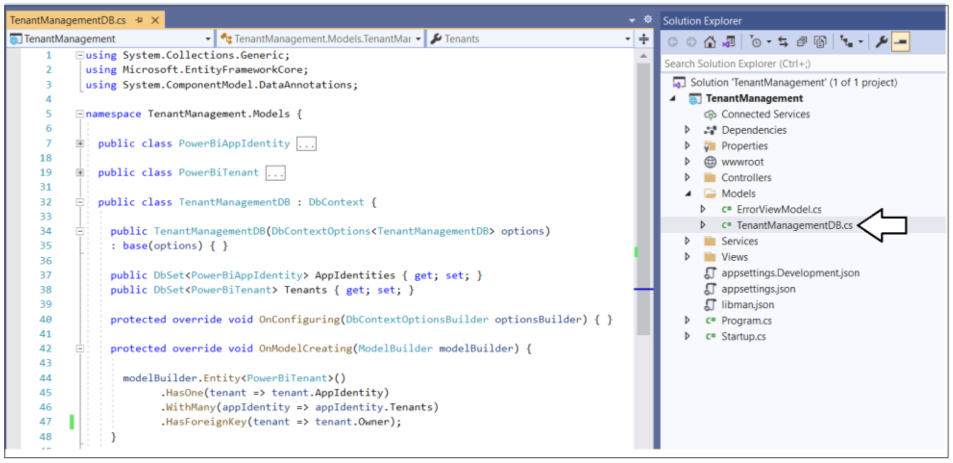


```
"TenantManagementDB": {
  "ConnectString": "Server=(localdb)\\MSSQLLocalDB;Database=Te
},
"DemoSettings": {
  "AdminUser": "tedp@pbiembedmart.onMicrosoft.com"
},
"Logging": {
  "LogLevel": {
```

Create the TenantManagementDB database

Before you can run the application in Visual Studio, you must create the project database named **TenantManagementDB**. This database schema has been created using the .NET 5 version of the Entity Framework. In this step, you will execute two PowerShell cmdlets provided by Entity Framework to create the database.

Before creating the **TenantManagementDB** database, take a moment to understand how it's been structured. Start by opening the file named **TenantManagementDB.cs** in the **Models** folder. Note that you shouldn't make any change to **TenantManagementDB.cs**. You are just going to inspect the file you understand how the **TenantManagementDB** database is generated.



```
using System.Collections.Generic;
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;

namespace TenantManagement.Models {

  public class PowerBiAppIdentity ...

  public class PowerBiTenant ...

  public class TenantManagementDB : DbContext {

    public TenantManagementDB(DbContextOptions<TenantManagementDB> options)
    : base(options) { }

    public DbSet<PowerBiAppIdentity> AppIdentities { get; set; }
    public DbSet<PowerBiTenant> Tenants { get; set; }

    protected override void OnConfiguring(DbContextOptionsBuilder optionsBuilder) { }

    protected override void OnModelCreating(ModelBuilder modelBuilder) {

      modelBuilder.Entity<PowerBiTenant>()
            .HasOne(tenant => tenant.AppIdentity)
            .WithMany(appIdentity => appIdentity.Tenants)
            .HasForeignKey(tenant => tenant.Owner);
    }
```

When you inspect the code inside **TenantManagementDB.cs**, you will see a class named **TenantManagementDB** that derives from **DbContext** to add support for automatic database generation using Entity Framework. The **TenantManagementDB** class serves as the top-level class for the Entity Framework which contains two **DBSet** properties named **AppIdentites** and **Tenants**. When you generate the database, each of these **DBSet** properties will be created as database tables. The **AppIdentites** table is generated using the table schema defined by the **PowerBiAppIdentity** class.

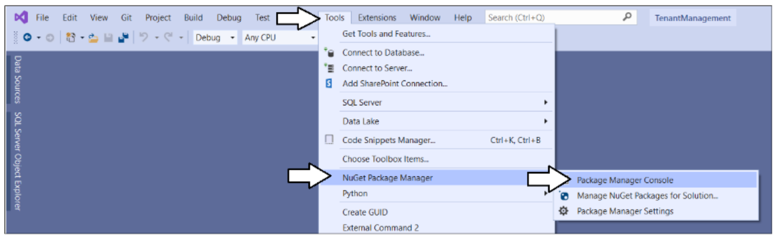
```
public class PowerBiAppIdentity {
  [Key]
  public string Name { get; set; }
  public string ApplicationId { get; set; }
  public string ApplicationObjectId { get; set; }
  public string ServicePrincipalId { get; set; }
  public string ClientSecret { get; set; }
  public string TenantId { get; set; }
  public bool Exclusive { get; set; }
  public virtual List<PowerBiTenant> Tenants { get; set; }
}
```

The **Tenants** table is generated using the table schema defined by the **PowerBiTenant** class.

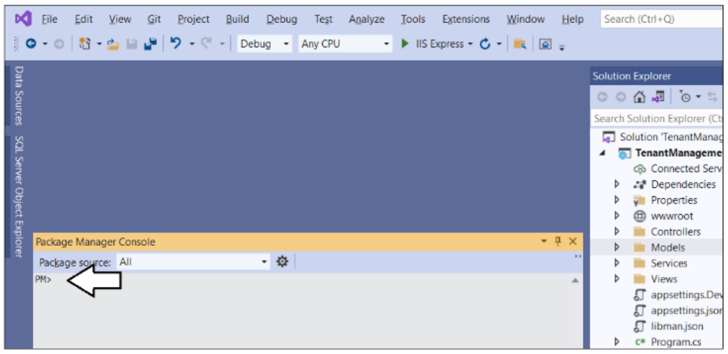
```
public class PowerBiTenant {
  [Key]
  public string Name { get; set; }
  public string WorkspaceId { get; set; }
  public string WorkspaceUrl { get; set; }
  public string DatabaseServer { get; set; }
  public string DatabaseName { get; set; }
  public string DatabaseUserName { get; set; }
  public string DatabaseUserPassword { get; set; }
  public string Owner { get; set; }
  public PowerBiAppIdentity AppIdentity { get; set; }
}
```

After you have inspected the code used to generated the database, close the source file named **TenantManagementDB.cs** without saving any changes. The next step is to run the PowerShell commands to create the project database named **TenantManagementDB**.

Open the Package Manager console using **Tools > NuGet Package Manager > Package Manager Console**.



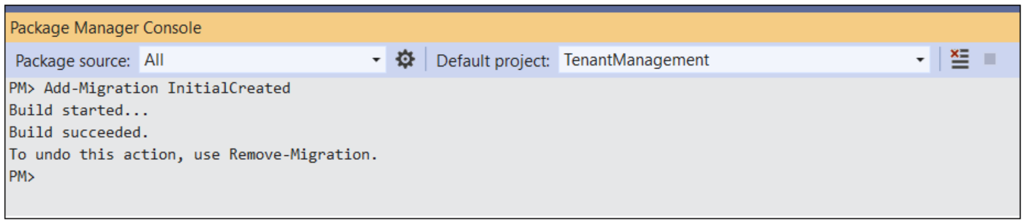
You should see the **Package Manager Console** command prompt where you can execute PowerShell commands.



Type and execute the following **Add-Migration** command to create a new Entity Framework migration in the project.

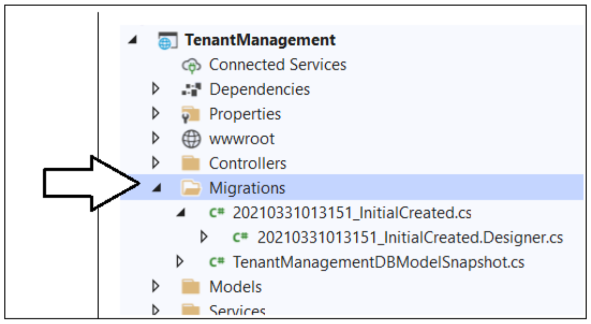
```
Add-Migration InitialCreate
```

The **Add-Migration** command should run without errors. If this command fails you might have to modify the database connection string in **appsettings.json**.



```
PM> Add-Migration InitialCreated
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM>
```

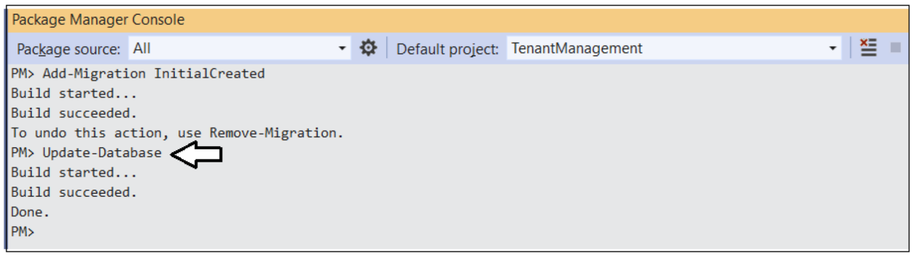
After running the Add-Migration command, you will see a new folder has been added to the project named **Migrations** with several C# source files. There is no need to change anything in thee source files but you can inspect what's inside them if you are curious how the Entity Framework does its work.


TenantManagement
Connected Services
Dependencies
Properties
wwwroot
Controllers
Migrations
20210331013151_InitialCreated.cs
20210331013151_InitialCreated.Designer.cs
TenantManagementDBModelSnapshot.cs
Models
Services

Return to the **Package Manager Console** and run the following **Update-Database** command to generate the database named **TenantManagementDB**.

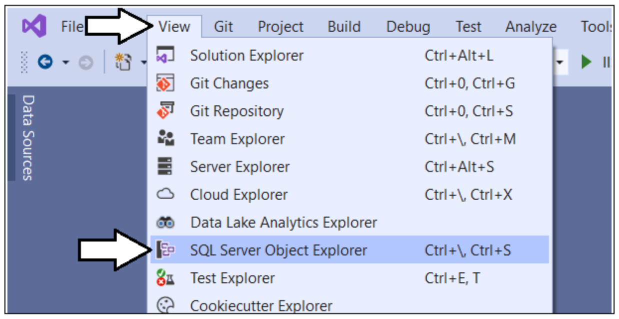
```
Update-Database
```

The **Update-Database** command should run without errors and generate the database named **TenantManagementDB**.

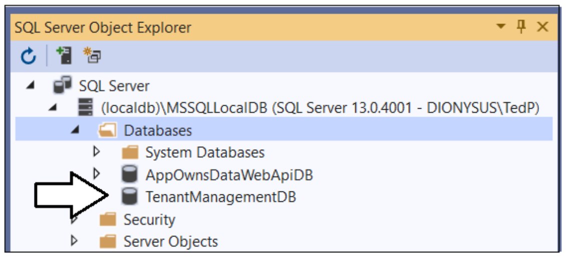


```
PM> Add-Migration InitialCreated
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> Update-Database
Build started...
Build succeeded.
Done.
PM>
```

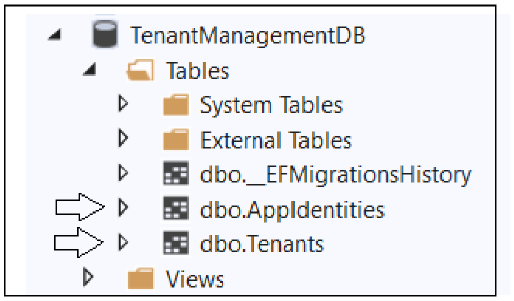
In Visual Studio, you can use the **SQL Server Object Explorer** to see the database that has just been created. Open the **SQL Server Object Explorer** by invoking the **View > SQL Server Object Explorer** menu command.


File
View
Git
Project
Build
Debug
Test
Analyze
Tools
Data Sources
Solution Explorer Ctrl+Alt+L
Git Changes Ctrl+0, Ctrl+G
Git Repository Ctrl+0, Ctrl+S
Team Explorer Ctrl+\, Ctrl+M
Server Explorer Ctrl+Alt+S
Cloud Explorer Ctrl+\, Ctrl+X
Data Lake Analytics Explorer
SQL Server Object Explorer Ctrl+\, Ctrl+S
Test Explorer Ctrl+E, T
Cookiecutter Explorer

Expand the **Databases** node for the server you are using and verify you an see the new database named **TenantManagementDB**.



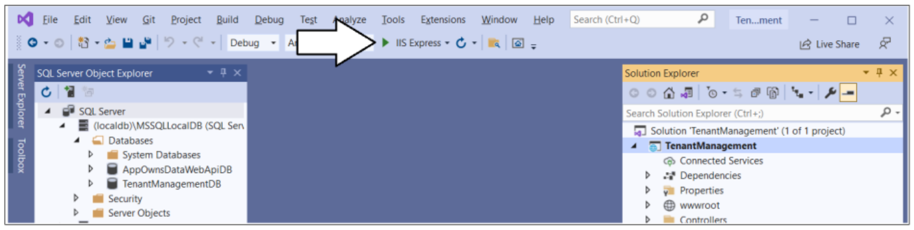
If you expand the **Tables** node for **TenantManagementDB**, you should see the two tables named **AppIdentities** and **Tenants**.



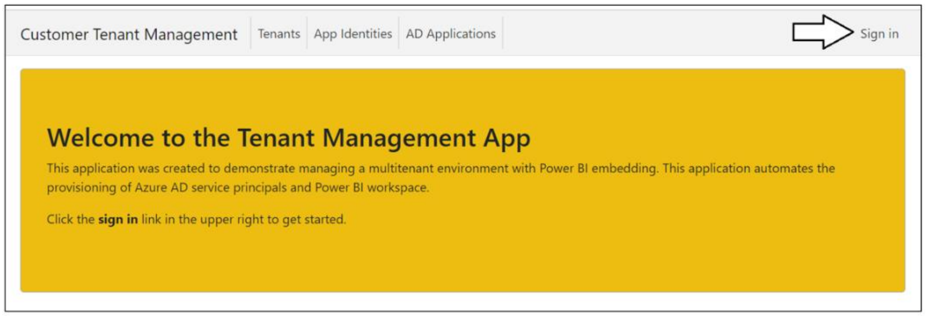
The **TenantManagementDB** database has now been set up and you are ready to run the application in the Visual Studio debugger.

Test the Tenant Management Application

Launch the **TenantManagement** web application in the Visual Studio debugger by pressing the **{F5}** key or clicking the Visual Studio **Play** button with the green arrow and the caption **IIS Express**.



When the application starts, click the **Sign in** link in the upper right corner to begin the user login sequence.

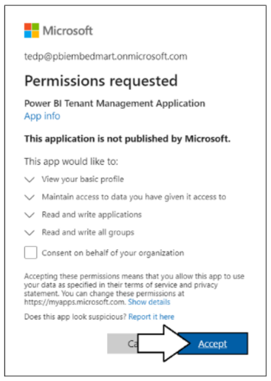


Welcome to the Tenant Management App

This application was created to demonstrate managing a multitenant environment with Power BI embedding. This application automates the provisioning of Azure AD service principals and Power BI workspace.

Click the **sign in** link in the upper right to get started.

The first time you authenticate with Azure AD, you'll be prompted with the **Permissions requested** dialog asking you to accept the delegated permissions for the Microsoft Graph API requested by the application. Click the **Accept** button to grant these permissions and continue.



tedp@pbiembedmart.onmicrosoft.com

Permissions requested

Power BI Tenant Management Application
App info

This application is not published by Microsoft.

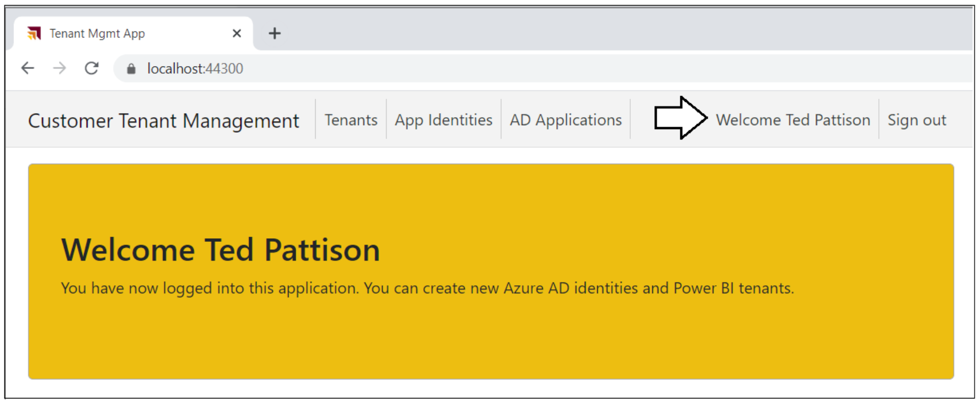
This app would like to:
- View your basic profile
- Maintain access to data you have given it access to
- Read and write applications
- Read and write all groups

☐ Consent on behalf of your organization

Accepting these permissions means that you allow this app to use your data as specified in their terms of service and privacy statement. You can change these permissions at https://myapps.microsoft.com. Show details

Does this app look suspicious? Report it here

Once you have logged in, you should see your name in the welcome message.

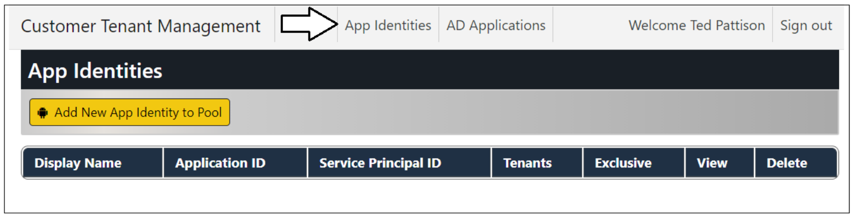


Welcome Ted Pattison

You have now logged into this application. You can create new Azure AD identities and Power BI tenants.

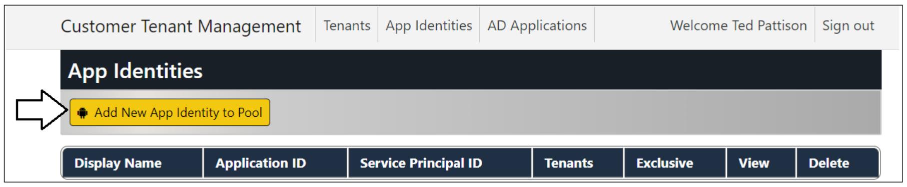
Create App Identities

Start by creating a few new App Identities. Click the **App Identities** link to navigate to the **App Identities** page.



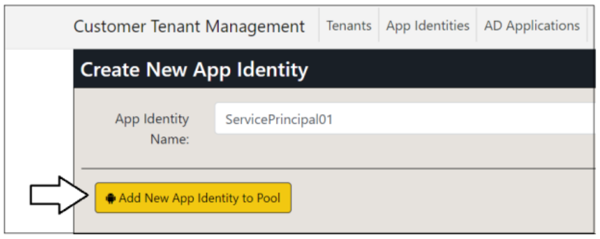
Display Name	Application ID	Service Principal ID	Tenants	Exclusive	View	Delete

Click the **Add New App Identity to Pool** button to display the **Create New App Identity** page.



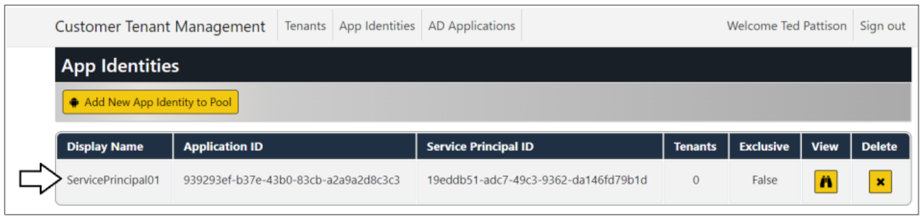
Display Name	Application ID	Service Principal ID	Tenants	Exclusive	View	Delete

When you open the **Create New App Identity** page, it will automatically populate the **App Identity Name** textbox with a value of **ServicePrincipal01**. Click the **Add New App Identity to Pool** button to create the new app identity.



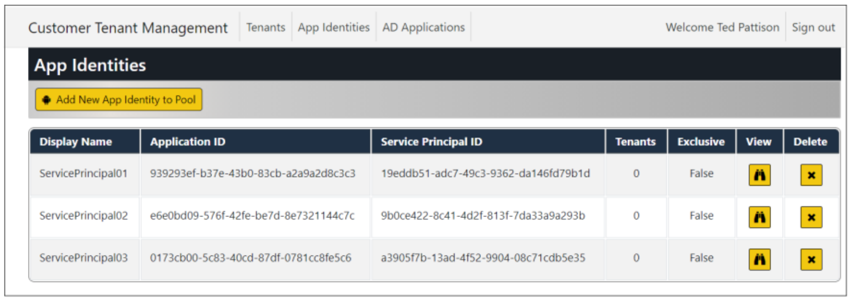
App Identity Name: ServicePrincipal01

After a few second, you should see the new app identity named **ServicePrinicpal01** on the **App Identities** page.



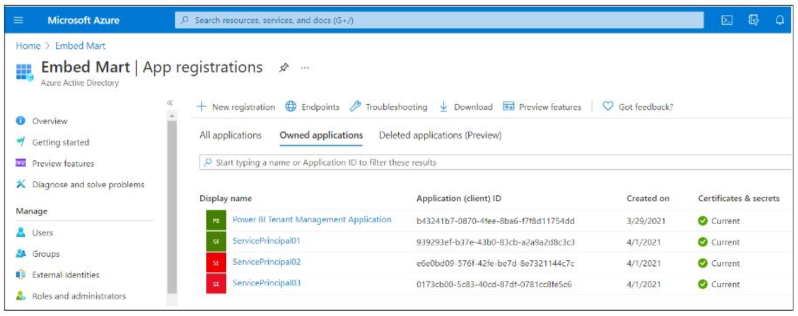
Display Name	Application ID	Service Principal ID	Tenants	Exclusive	View	Delete
ServicePrincipal01	939293ef-b37e-43b0-83cb-a2a9a2d8c3c3	19eddb51-adc7-49c3-9362-da146fd79b1d	0	False		

Follow the same steps to create two more app identities named **ServicePrincipal02** and **ServicePrincipal03**. When you're done, the **App Identities** page should match the following screenshot.



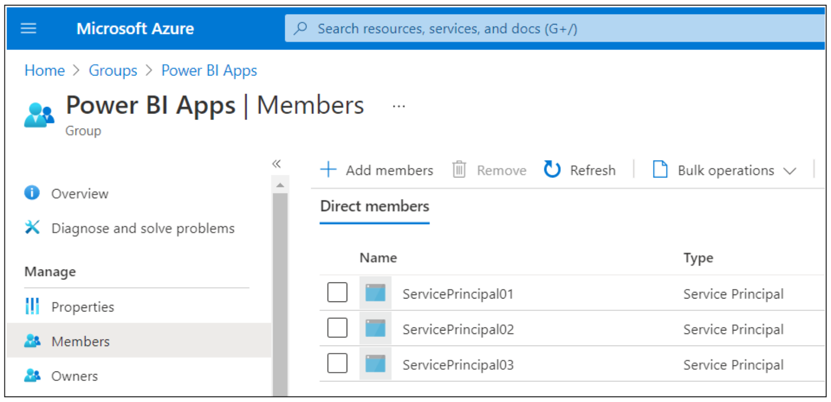
Display Name	Application ID	Service Principal ID	Tenants	Exclusive	View	Delete
ServicePrincipal01	939293ef-b37e-43b0-83cb-a2a9a2d8c3c3	19eddb51-adc7-49c3-9362-da146fd79b1d	0	False		
ServicePrincipal02	e6e0bd09-576f-42fe-be7d-8e7321144c7c	9b0ce422-8c41-4d2f-813f-7da33a9a293b	0	False		
ServicePrincipal03	0173cb00-5c83-40cd-87df-0781cc8fe5c6	a3905f7b-13ad-4f52-9904-08c71cdb5e35	0	False		

Note that behind the scenes the **TenantManagement** application is using the Microsoft Graph API to create new Azure AD application each time you create a new app identity. If you return pack to the App registration page in the Azure portal, you will see that an Azure AD application has been created for each app identity you've created.



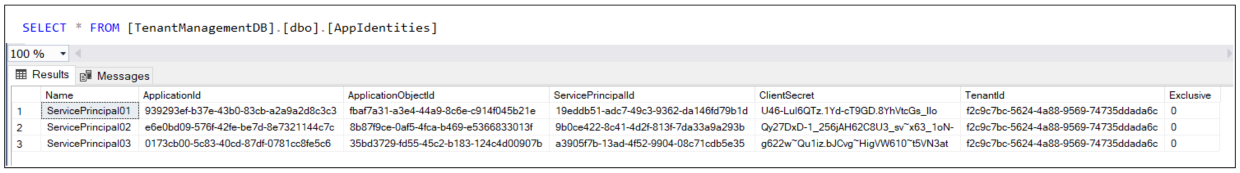
Display name	Application (client) ID	Created on	Certificates & secrets
Power BI Tenant Management Application	b43241b7-0870-4fee-8ba6-f7f8d11754dd	3/29/2021	✓ Current
ServicePrincipal01	939293ef-b37e-43b0-83cb-a2a9a2d8c3c3	4/1/2021	✓ Current
ServicePrincipal02	e6e0bd09-576f-42fe-be7d-8e7321144c7c	4/1/2021	✓ Current
ServicePrincipal03	0173cb00-5c83-40cd-87df-0781cc8fe5c6	4/1/2021	✓ Current

If you return to the Groups page in the Azure portal and drill into the **Members** page of the **Power BI Apps** security group, you will see that the **TenantManagement** application has also added the service principal for each azure AD application as a group member. This is important because these service principals must be added to this Azure AD security group or they will not be able to call the Power BI REST API.



Name	Type
ServicePrincipal01	Service Principal
ServicePrincipal02	Service Principal
ServicePrincipal03	Service Principal

In addition to communicating with Azure AD to create and configure Azure AD application, the **TenantManagement** application also captures application metadata and authentication credentials so it can store them in the **TenantManagementDB** database. Soon you will see how the **TenantManagement** application is able to retrieve these credentials on demand and authenticate with Azure AD under the identity of any of these Azure AD applications.



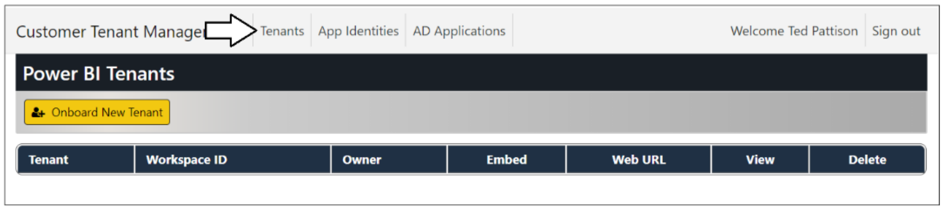
```
SELECT * FROM [TenantManagementDB].[dbo].[AppIdentities]
```

	Name	ApplicationId	ApplicationObjectId	ServicePrincipalId	ClientSecret	TenantId	Exclusive
1	ServicePrincipal01	939293ef-b37e-43b0-83cb-a2a9a2d8c3c3	fbaf7a31-a3e4-44a9-8c6e-c914f045b21e	19eddb51-adc7-49c3-9362-da146fd79b1d	U46-Lul6QTz.1Yd-cT9GD.8YhVtcGs_llo	f2c9c7bc-5624-4a88-9569-74735ddada6c	0
2	ServicePrincipal02	e6e0bd09-576f-42fe-be7d-8e7321144c7c	8b87f9ce-0af5-4fca-b469-e5366833013f	9b0ce422-8c41-4d2f-813f-7da33a9a293b	Qy27DxD-1_256jAH62C8U3_sv~x63_1oN-	f2c9c7bc-5624-4a88-9569-74735ddada6c	0
3	ServicePrincipal03	0173cb00-5c83-40cd-87df-0781cc8fe5c6	35bd3729-fd55-45c2-b183-124c4d00907b	a3905f7b-13ad-4f52-9904-08c71cdb5e35	g622w~Qu1iz.bJCvg~HigVW610~t5VN3at	f2c9c7bc-5624-4a88-9569-74735ddada6c	0

CAVEAT: Keep in mind that the **TenantManagement** application has been designed as a proof-of-concept (POC) application to teach concepts and provide a starting point for other developers. This application does not include certain aspects that are important to include in a real-world applications such as hiding secrets. If you plan to extend this POC sample application into a production application, it will be your responsibility to add support for hiding credentials such as the Client Secret. You can consider an approach such as using the Always Encrypted feature in Azure SQL or extending the **TenantManagement** application to store client secrets or client certificates in Azure Key Vault.

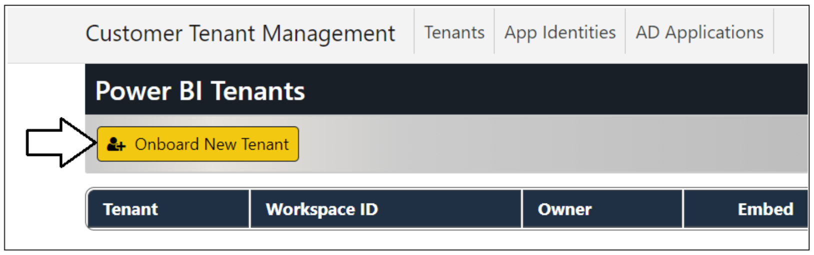
Create New Power BI Tenants

Return to the **TenantManagement** application and navigate to the **Tenants** page.

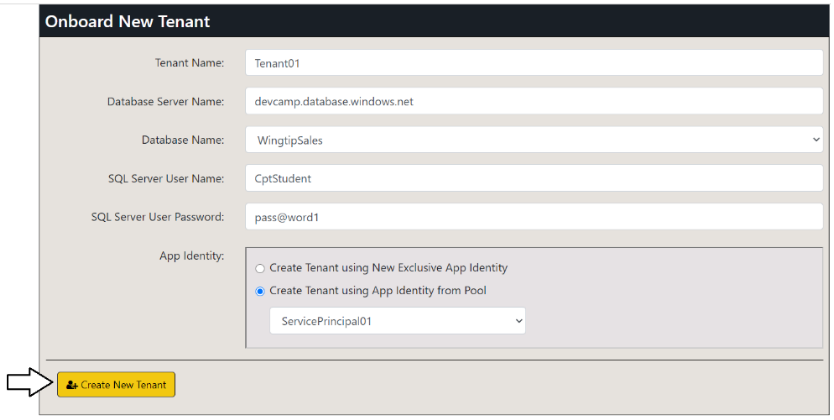


Tenant	Workspace ID	Owner	Embed	Web URL	View	Delete

Click the **Onboard New Tenant** button to display the **Onboard New Tenant** page.

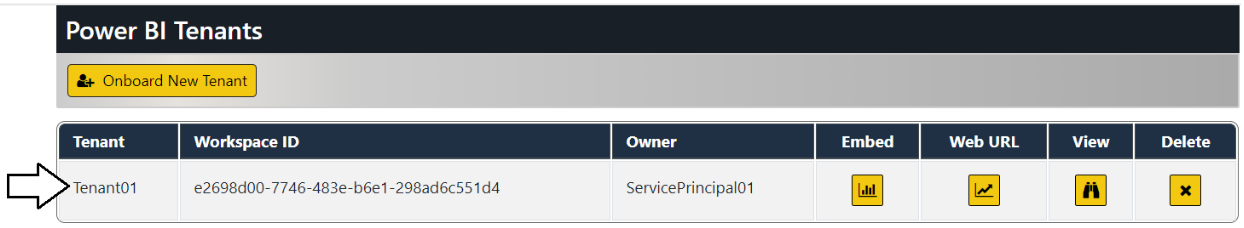


You can create the first tenant using the default values supplied by the **Onboard New Tenant** page. Click to **Create New Tenant** button to begin the process of creating a new customer tenant.


Onboard New Tenant

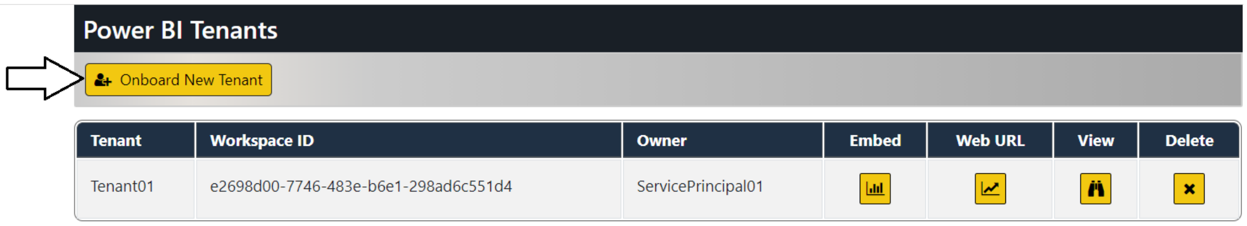
Tenant Name:	Tenant01
Database Server Name:	devcamp.database.windows.net
Database Name:	WingtipSales
SQL Server User Name:	CptStudent
SQL Server User Password:	pass@word1
App Identity:	○ Create Tenant using New Exclusive App Identity ● Create Tenant using App Identity from Pool ServicePrincipal01

After a few seconds, you should see the new customer tenant has been created.



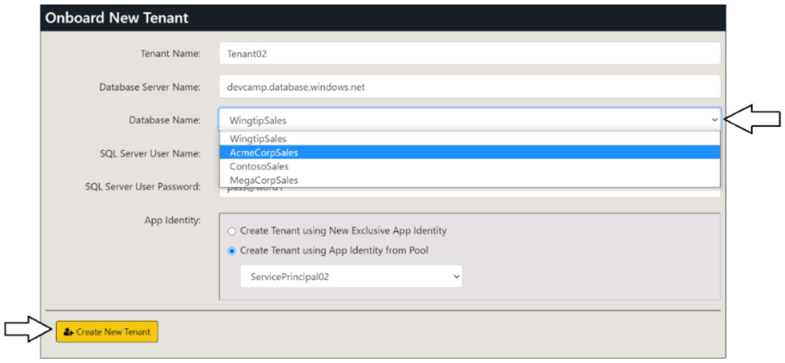
Tenant	Workspace ID	Owner	Embed	Web URL	View	Delete
Tenant01	e2698d00-7746-483e-b6e1-298ad6c551d4	ServicePrincipal01				

Click the **Onboard New Tenant** button again to create a second tenant.



Tenant	Workspace ID	Owner	Embed	Web URL	View	Delete
Tenant01	e2698d00-7746-483e-b6e1-298ad6c551d4	ServicePrincipal01				

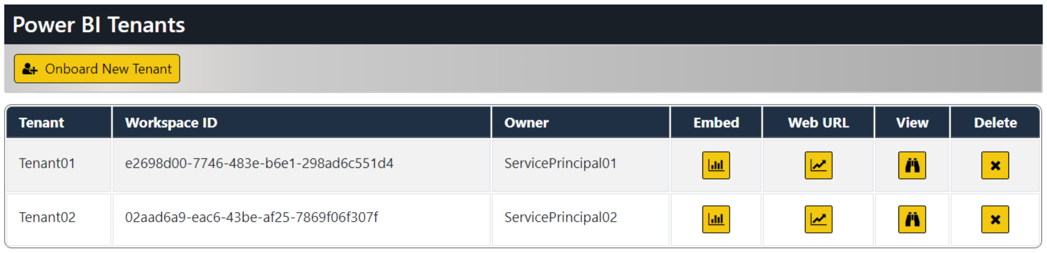
This time, select a different database for **Database Name** and then click **Create New Tenant**.



Onboard New Tenant

Tenant Name:	Tenant02
Database Server Name:	devcamp.database.windows.net
Database Name:	WingtipSales
SQL Server User Name:	
SQL Server User Password:	pass@word1
App Identity:	○ Create Tenant using New Exclusive App Identity ● Create Tenant using App Identity from Pool ServicePrincipal02

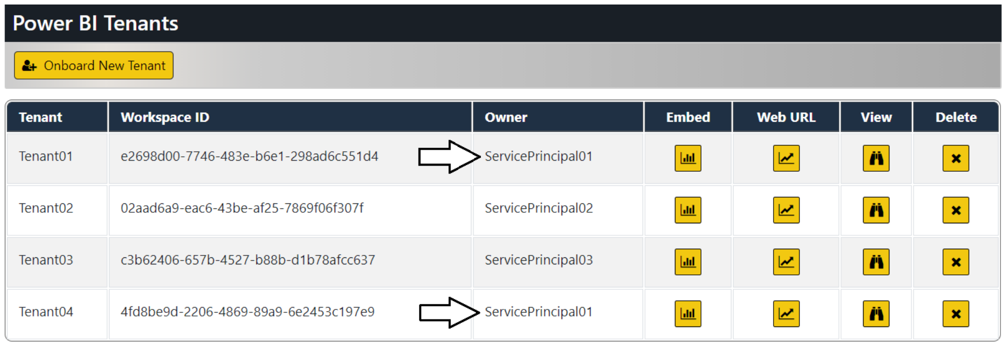
You should now have two customer tenants. Note they each tenant has a different app identity as its **Owner**.



Power BI Tenants

Tenant	Workspace ID	Owner	Embed	Web URL	View	Delete
Tenant01	e2698d00-7746-483e-b6e1-298ad6c551d4	ServicePrincipal01				
Tenant02	02aad6a9-eac6-43be-af25-7869f06f307f	ServicePrincipal02				

Follow the same steps to create two more customer tenants so that there are 3 app identities and 4 customer tenants. Once you have created more tenants then app identities, you should see app identity pooling where multiple customer tenants share the same app identity.



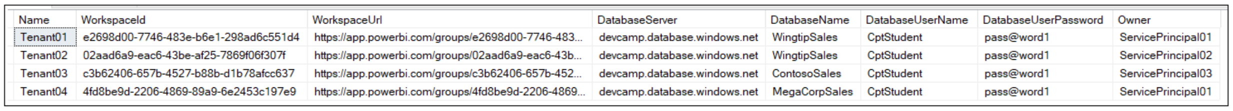
Power BI Tenants

Tenant	Workspace ID	Owner	Embed	Web URL	View	Delete
Tenant01	e2698d00-7746-483e-b6e1-298ad6c551d4	ServicePrincipal01				
Tenant02	02aad6a9-eac6-43be-af25-7869f06f307f	ServicePrincipal02				
Tenant03	c3b62406-657b-4527-b88b-d1b78afcc637	ServicePrincipal03				
Tenant04	4fd8be9d-2206-4869-89a9-6e2453c197e9	ServicePrincipal01				

Now let's discuss what's going on behind the scenes. As you create a new customer tenant, the **TenantManagement** application uses the Power BI REST API to implement the following onboarding logic.

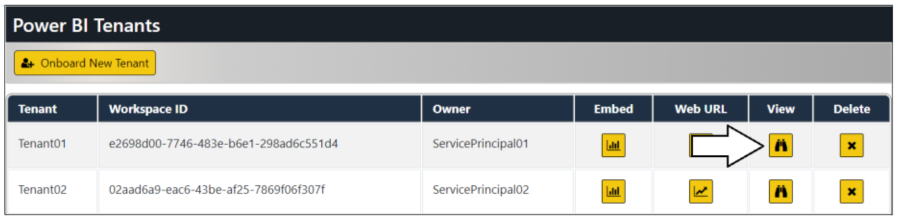
1. Create a new Power BI workspace
2. Upload a template PBIX file to create the **Sales** dataset and the **Sales** report
3. Update dataset parameters on **Sales** dataset to point to this customer's database
4. Patch credentials for the SQL datasource used by the **Sales** dataset
5. Start a refresh operation on the **Sales** database

The **TenantManagement** application also create a new record in the **Tenants** table of the **TenantManagementDB** database. Note that the application identity associated with this customer tenant is tracked in the **Owner** column.



Name	WorkspaceId	WorkspaceUrl	DatabaseServer	DatabaseName	DatabaseUserName	DatabaseUserPassword	Owner
Tenant01	e2698d00-7746-483e-b6e1-298ad6c551d4	https://app.powerbi.com/groups/e2698d00-7746-483...	devcamp.database.windows.net	WingtipSales	CptStudent	pass@word1	ServicePrincipal01
Tenant02	02aad6a9-eac6-43be-af25-7869f06f307f	https://app.powerbi.com/groups/02aad6a9-eac6-43b...	devcamp.database.windows.net	WingtipSales	CptStudent	pass@word1	ServicePrincipal02
Tenant03	c3b62406-657b-4527-b88b-d1b78afcc637	https://app.powerbi.com/groups/c3b62406-657b-452...	devcamp.database.windows.net	ContosoSales	CptStudent	pass@word1	ServicePrincipal03
Tenant04	4fd8be9d-2206-4869-89a9-6e2453c197e9	https://app.powerbi.com/groups/4fd8be9d-2206-4869...	devcamp.database.windows.net	MegaCorpSales	CptStudent	pass@word1	ServicePrincipal01

Click on the **View** button for a specific tenant on the **Power BI Tenants** page to drill into the **Tenant Details** page.

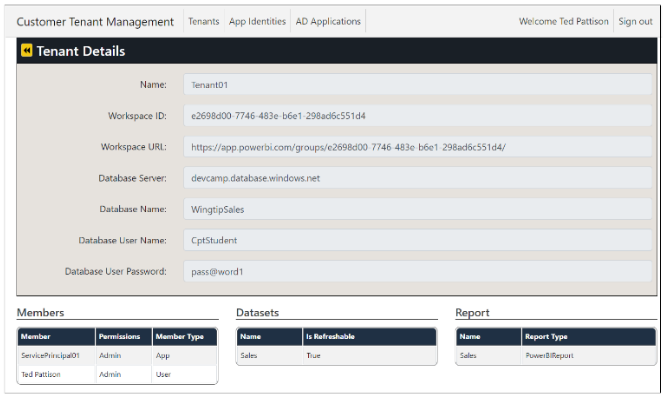


Power BI Tenants

Onboard New Tenant

Tenant	Workspace ID	Owner	Embed	Web URL	View	Delete
Tenant01	e2698d00-7746-483e-b6e1-298ad6c551d4	ServicePrincipal01				
Tenant02	02aad6a9-eac6-43be-af25-7869f06f307f	ServicePrincipal02				

The **Tenant Details** page displays Power BI workspace details including its members, datasets and reports.



Customer Tenant Management | Tenants | App Identities | AD Applications | Welcome Ted Pattison | Sign out

Tenant Details

Name:	Tenant01
Workspace ID:	e2698d00-7746-483e-b6e1-298ad6c551d4
Workspace URL:	https://app.powerbi.com/groups/e2698d00-7746-483e-b6e1-298ad6c551d4/
Database Server:	devcamp.database.windows.net
Database Name:	WingtipSales
Database User Name:	CptStudent
Database User Password:	pass@word1

Members

Member	Permissions	Member Type
ServicePrincipal01	Admin	App
Ted Pattison	Admin	User

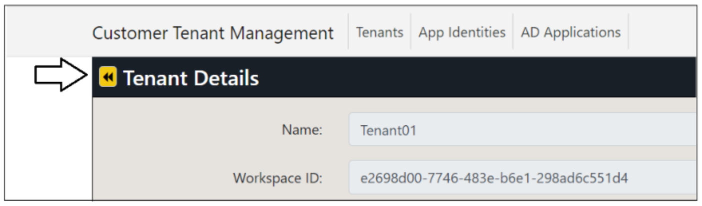
Datasets

Name	Is Refreshable
Sales	True

Report

Name	Report Type
Sales	PowerBIReport

Click on the back arrow to return to the **Power BI Tenants** page.



Customer Tenant Management | Tenants | App Identities | AD Applications

Tenant Details

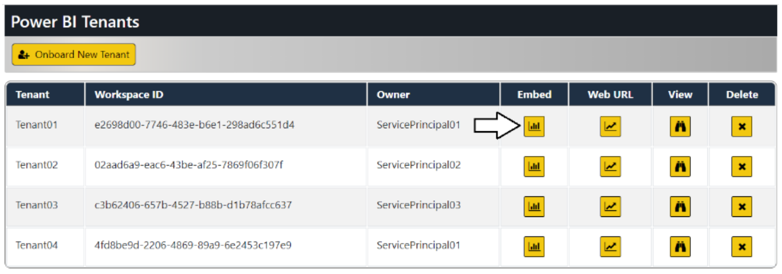
Name:	Tenant01
Workspace ID:	e2698d00-7746-483e-b6e1-298ad6c551d4

If you're interested, you can examine the details of other tenants as well.

Embed Reports

Now it's time to make use of the **TenantManagement** application's ability to embed reports. When navigate to the **Embed** page for a customer tenant, the **TenantManagement** application must acquire an access token for whichever app identity was used to create the customer tenant. The service principal that is configured as the **Owner** of a tenant will be the only service principal who will have access to access the target workspace in Power BI.

Move to the **Power BI Tenants** page and click on the **Embed** button for the first customer tenant.

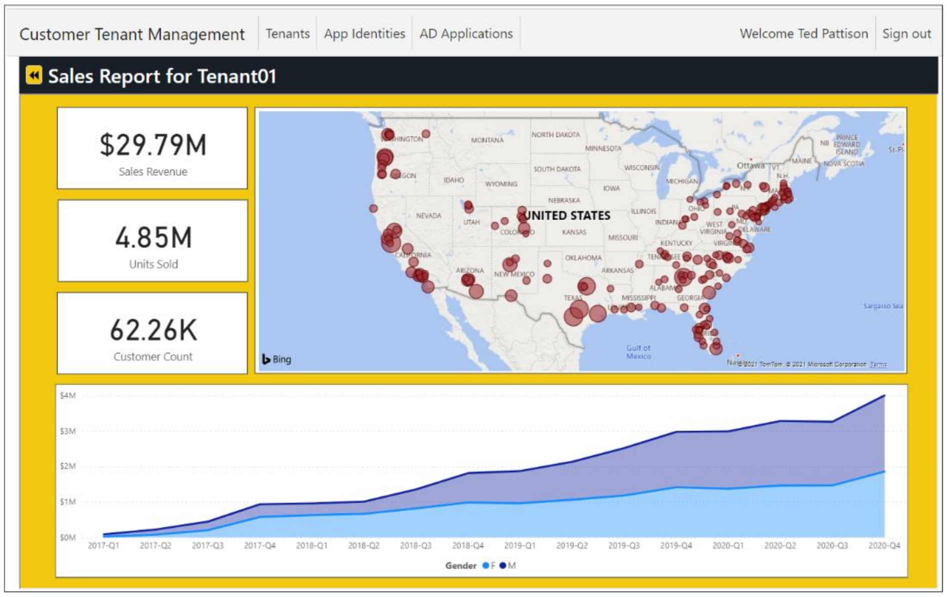


Power BI Tenants

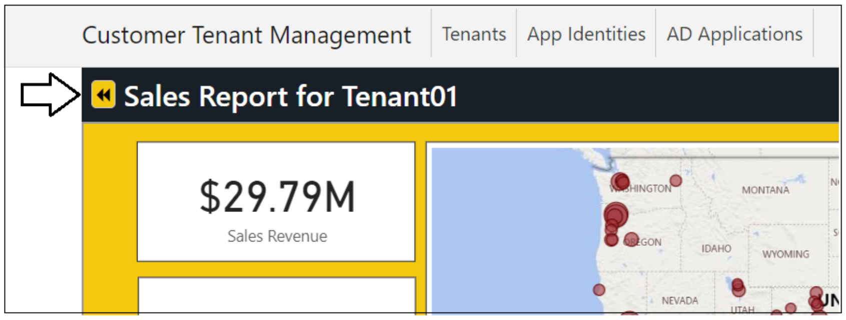
Onboard New Tenant

Tenant	Workspace ID	Owner	Embed	Web URL	View	Delete
Tenant01	e2698d00-7746-483e-b6e1-298ad6c551d4	ServicePrincipal01				
Tenant02	02aad6a9-eac6-43be-af25-7869f06f307f	ServicePrincipal02				
Tenant03	c3b62406-657b-4527-b88b-d1b78afcc637	ServicePrincipal03				
Tenant04	4fd8be9d-2206-4869-89a9-6e2453c197e9	ServicePrincipal01				

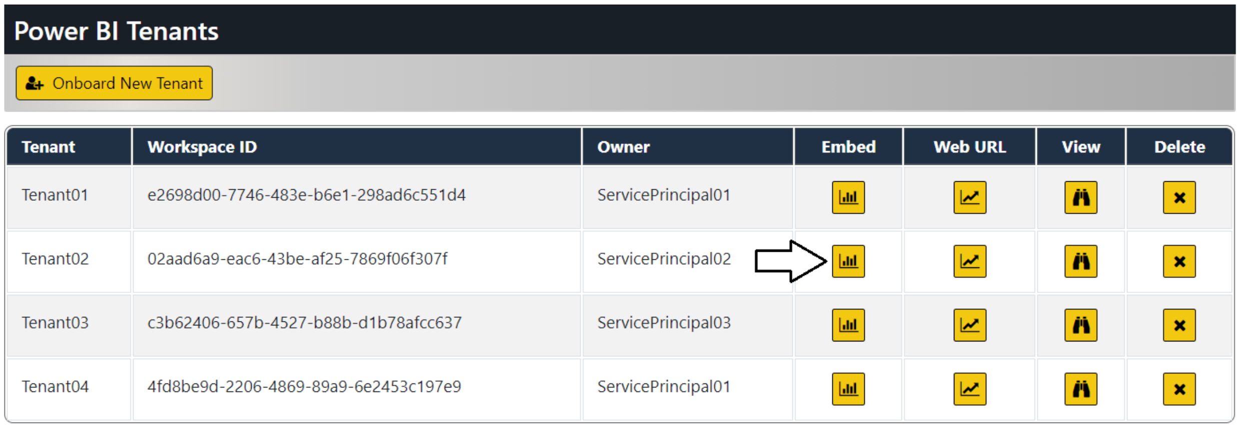
You should now see a page with an embedded report for that tenant. When you click the **Embed** button to embed a report for a customer tenant, the **TenantManagement** application retrieves credentials for the app identity associated with the tenant from the **TenantManagementDB** database. It then uses those credentials to acquire an access token from Azure AD using Client Credentials Flow. That access token is then used to communicate with the Power BI Service to retrieve report metadata and generate an embed token for the embedding process.



Click on the back arrow button to return to the **Tenants** page.



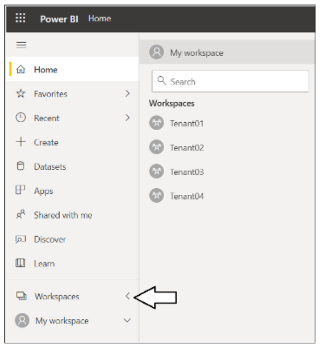
Now test clicking the **Embed** button for other customer tenants. As you can see, the **TenantManagement** application has the ability to acquire access tokens for any of the Azure AD applications that it has created.



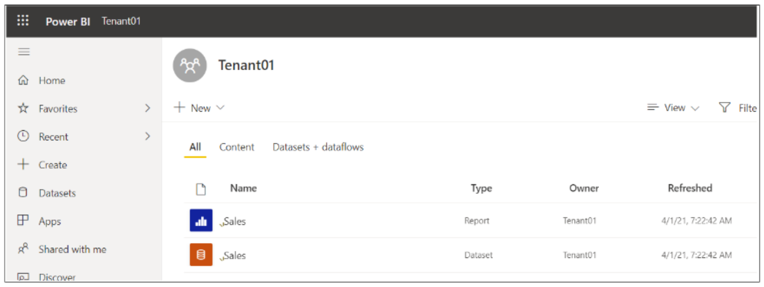
Tenant	Workspace ID	Owner	Embed	Web URL	View	Delete
Tenant01	e2698d00-7746-483e-b6e1-298ad6c551d4	ServicePrincipal01				
Tenant02	02aad6a9-eac6-43be-af25-7869f06f307f	ServicePrincipal02				
Tenant03	c3b62406-657b-4527-b88b-d1b78afcc637	ServicePrincipal03				
Tenant04	4fd8be9d-2206-4869-89a9-6e2453c197e9	ServicePrincipal01				

Inspect the Power BI Workspaces

If you're curious about what's been created in Power BI, you can see for yourself by navigating to the Power BI Service portal at https://app.powerbi.com. You should be able to see and navigate to any of the Power BI workspaces that have been created by the **TenantManagement** application.

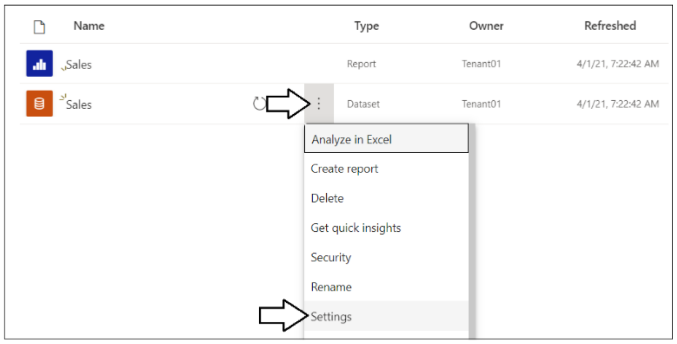


Navigate to one of these workspaces such as **Tenant01**.



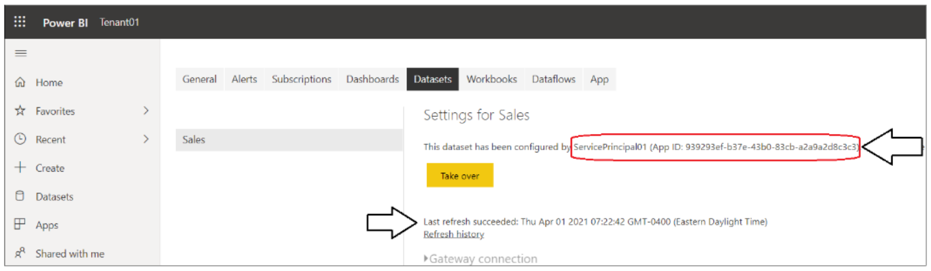
Name	Type	Owner	Refreshed
Sales	Report	Tenant01	4/1/21, 7:22:42 AM
Sales	Dataset	Tenant01	4/1/21, 7:22:42 AM

Drill into the **Setting** page for the dataset named **Sales**.



Name	Type	Owner	Refreshed
Sales	Report	Tenant01	4/1/21, 7:22:42 AM
Sales	Dataset	Tenant01	4/1/21, 7:22:42 AM

You should be able to verify that the **Sales** dataset has been configured by one of the Azure AD applications that was created by the **TenantManagement** application. You should also be able to see the **Last refresh succeeded** message for the dataset refresh operation that was started by the **TenantManagement** as part of its tenant onboarding logic.



Settings for Sales

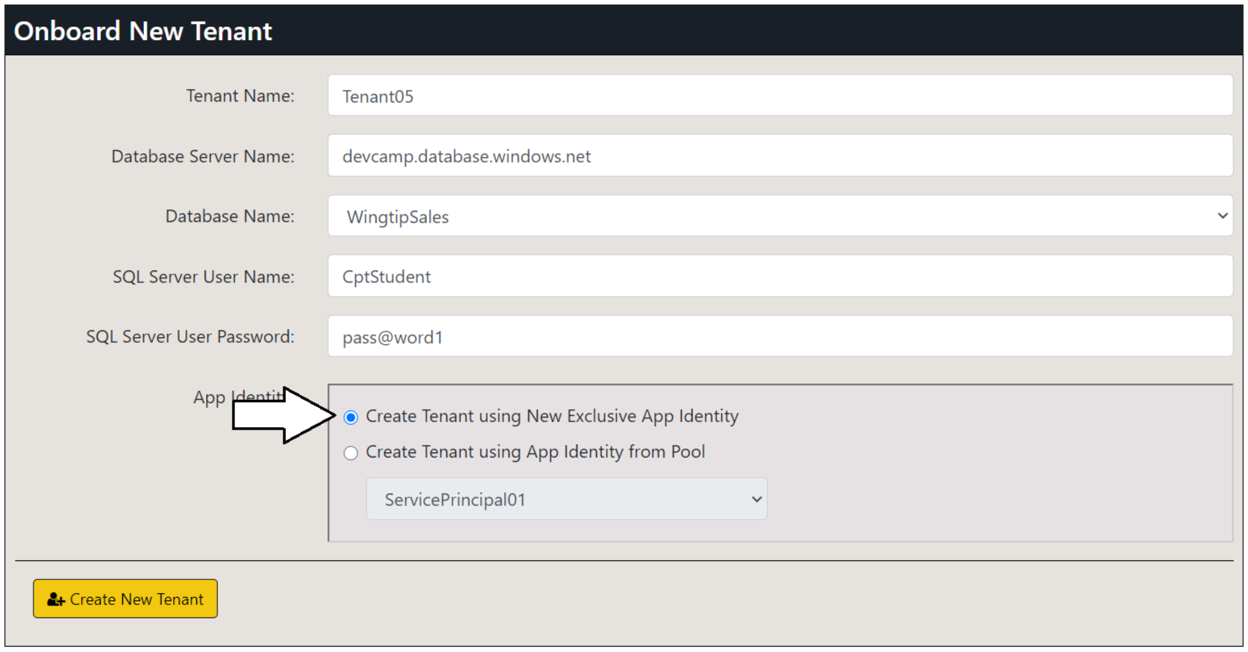
This dataset has been configured by ServicePrincipal01 (App ID: 939293ef-b37e-43b0-83cb-a2a9a2d8c3c3)

Last refresh succeeded: Thu Apr 01 2021 07:22:42 GMT-0400 (Eastern Daylight Time)

Create a Separate Service Principal for Each Tenant

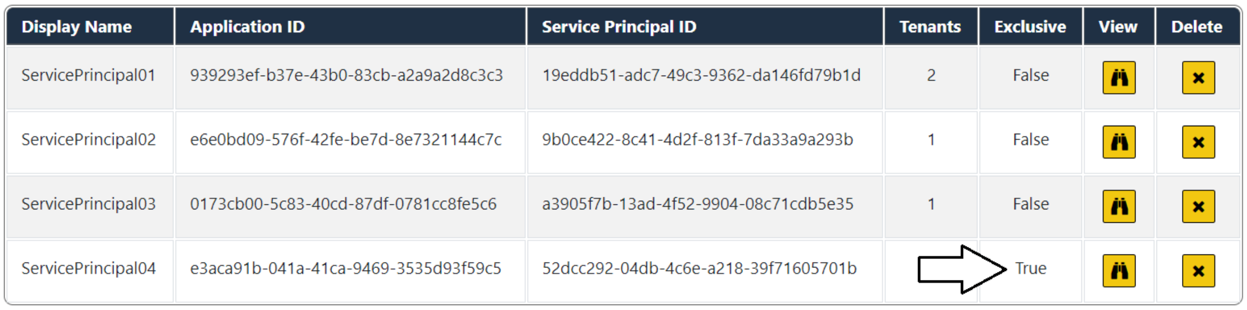
At this point you have used the **TenantManagement** application to pool app identities where one service principal can be the owner of multiple customer tenants. While this approach will work for many organizations and ISVs using Power BI embedding, you can take things one step further by creating a new service principal each time you create a new customer tenant. An application design which maintains a one-to-one relation between service principals and customer tenants will provide the most secure level of isolation. When you pool a service principal across tenants, that service principal will be the owner of datasource credentials for more than a single client.

When creating a new customer tenant using the **Onboard New Tenant** page, you can select the option to **Create Tenant using New Exclusive App Identity**. If you select this option, the **TenantManagement** application will create a new Azure AD application and then use the service principal from that application to create the workspace in Power BI.


Onboard New Tenant

Tenant Name:	Tenant05
Database Server Name:	devcamp.database.windows.net
Database Name:	WingtipSales
SQL Server User Name:	CptStudent
SQL Server User Password:	pass@word1
App Identity	[✓ Create Tenant using New Exclusive App Identity] Create Tenant using App Identity from Pool; ServicePrincipal01

You should take note that The **AppIdentities** table in the **TenantManagementDB** database contains a boolean column named **Exclusive**. When an app identity has an **Exclusive** column value of **true**, there is logic in the application which knows it should not include that app identity in the pool of app identities available on the **Onboard New Tenant** page.



Display Name	Application ID	Service Principal ID	Tenants	Exclusive	View	Delete
ServicePrincipal01	939293ef-b37e-43b0-83cb-a2a9a2d8c3c3	19eddb51-adc7-49c3-9362-da146fd79b1d	2	False		
ServicePrincipal02	e6e0bd09-576f-42fe-be7d-8e7321144c7c	9b0ce422-8c41-4d2f-813f-7da33a9a293b	1	False		
ServicePrincipal03	0173cb00-5c83-40cd-87df-0781cc8fe5c6	a3905f7b-13ad-4f52-9904-08c71cdb5e35	1	False		
ServicePrincipal04	e3aca91b-041a-41ca-9469-3535d93f59c5	52dcc292-04db-4c6e-a218-39f71605701b		True		

This concludes the walkthrough of the **TenantManagement** application.