



App-Owns-Data Starter Kit

Ted Pattison
Power BI Customer Advisory Team (PBICAT)

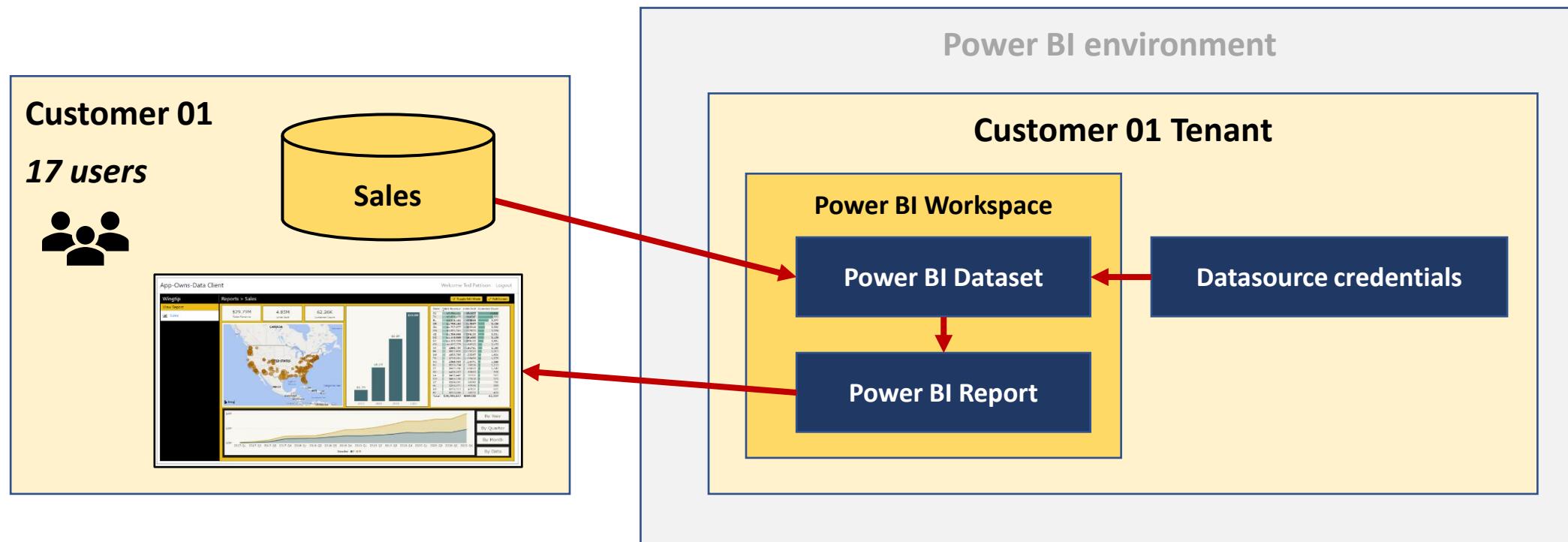
Agenda

➤ Solution Architecture

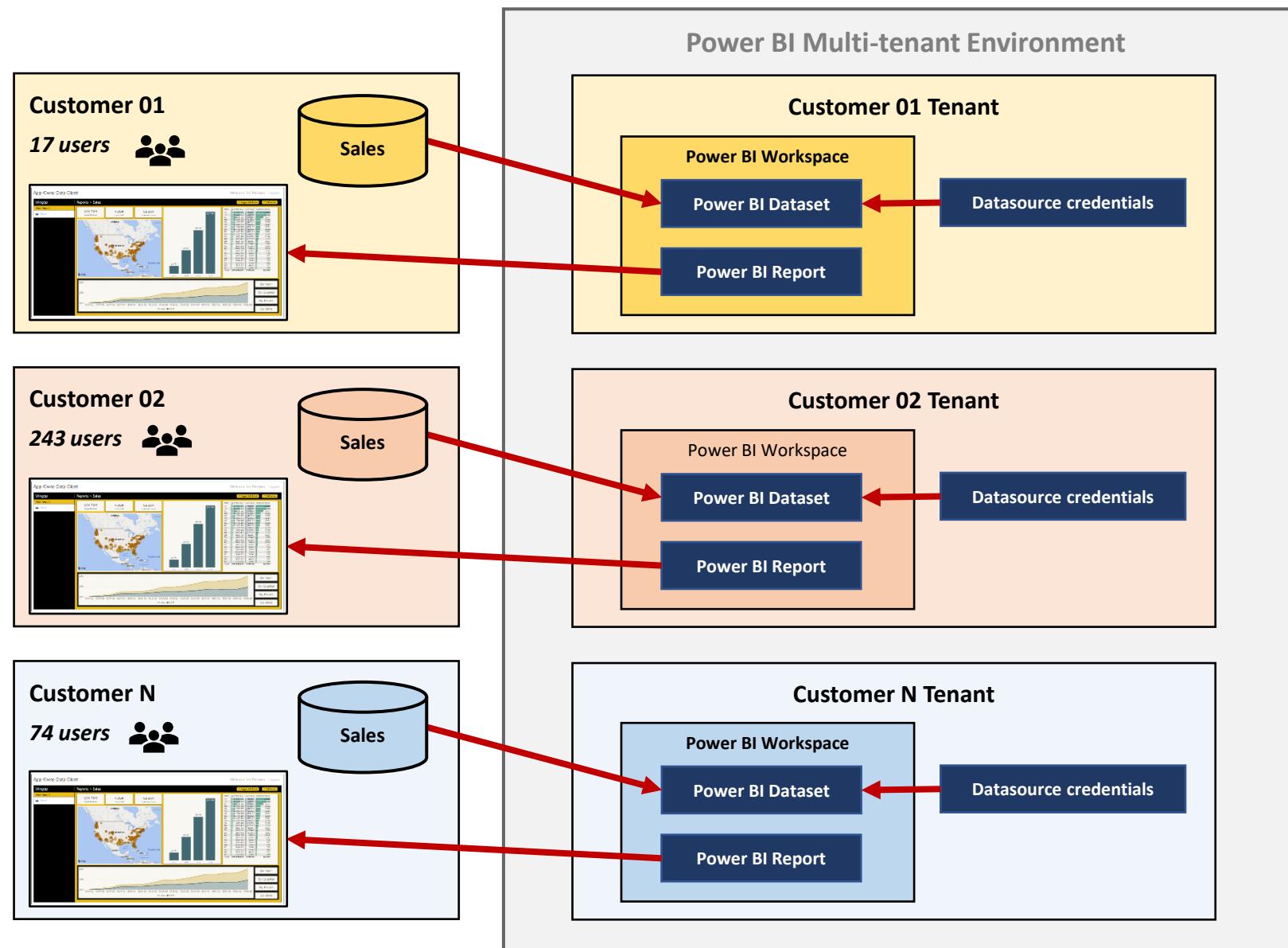
- Set up your development environment
- Open the App-Owns-Data Starter Kit in Visual Studio
- Test the AppOwnsDataAdmin application
- Test the AppOwnsDataClient application
- Use activity logs to monitor user activity and report performance

Customer Tenants in App-Owns-Data Embedding

- Tenant represents a customer with one or more users
 - You must create a separate tenant footprint in Power BI for each customer



Designing a Multi-tenant Environment



App-Owns-Data Embedding & Authorization

- Power BI doesn't know anything about your users
 - Power BI cannot provide assistance when it comes to authorization
 - Not an issue in scenario where all users see all content with the same access
 - Your application must add explicit support for any authorization scheme
- Key observations about App-Owns-Data embedding
 - you have **flexibility** to design the authorization scheme any way you'd like
 - you have **responsibility** to build authorization scheme from the ground up
 - you should use **custom database** to track the authorization data you need

App-Owns-Data Starter Kit - Value Proposition

- Provides guidance for the following tasks and procedures
 - Onboarding new **customer tenants**
 - Managing **user permissions**
 - Implementing customer-facing client as **Single Page Application (SPA)**
 - Creating **custom telemetry layer** to log user activity
 - Monitoring **user actions** such as viewing, editing, and creating reports
 - Monitoring **report performance** across all customer tenants

App-Owns-Data Starter Kit on GitHub

<https://github.com/PowerBiDevCamp/App-Owns-Data-Starter-Kit>

The screenshot shows the GitHub repository page for 'PowerBiDevCamp/App-Owns-Data-Starter-Kit'. The repository has 1 branch and 0 tags. The main branch was updated 9 hours ago by user 'TedPattison' with 186 commits. The commit history includes updates to 'AppOwnsDataAdmin', 'AppOwnsDataClient', 'AppOwnsDataShared', 'AppOwnsDataWebApi', 'Docs', 'AppOwnsDataStarterKit.sln', and 'AppOwsDataUsageReporting.pbix'. The repository description states: 'App-Owns-Data Starter Kit is a developer sample demonstrating common design techniques used in App-Owns-Data embedding.' It also links to 'Readme' and 'MIT License'. There are no releases published.

GitHub - PowerBiDevCamp/App-Owns-Data-Starter-Kit

github.com/PowerBiDevCamp/App-Owns-Data-Starter-Kit

Why GitHub? Team Enterprise Explore Marketplace Pricing

Search Sign in Sign up

PowerBiDevCamp / App-Owns-Data-Starter-Kit

Code Issues Pull requests Actions Projects Wiki Security Insights

main 1 branch 0 tags Go to file Code

TedPattison Updates 8bada81 9 hours ago 186 commits

- AppOwnsDataAdmin Updates 7 days ago
- AppOwnsDataClient Updates 7 days ago
- AppOwnsDataShared Updates 7 days ago
- AppOwnsDataWebApi Delete AppOwnsDataWebApi.csproj.user 7 days ago
- Docs Updates 9 hours ago
- AppOwnsDataStarterKit.sln Updates 15 days ago
- AppOwsDataUsageReporting.pbix Updates 15 days ago

About

App-Owns-Data Starter Kit is a developer sample demonstrating common design techniques used in App-Owns-Data embedding.

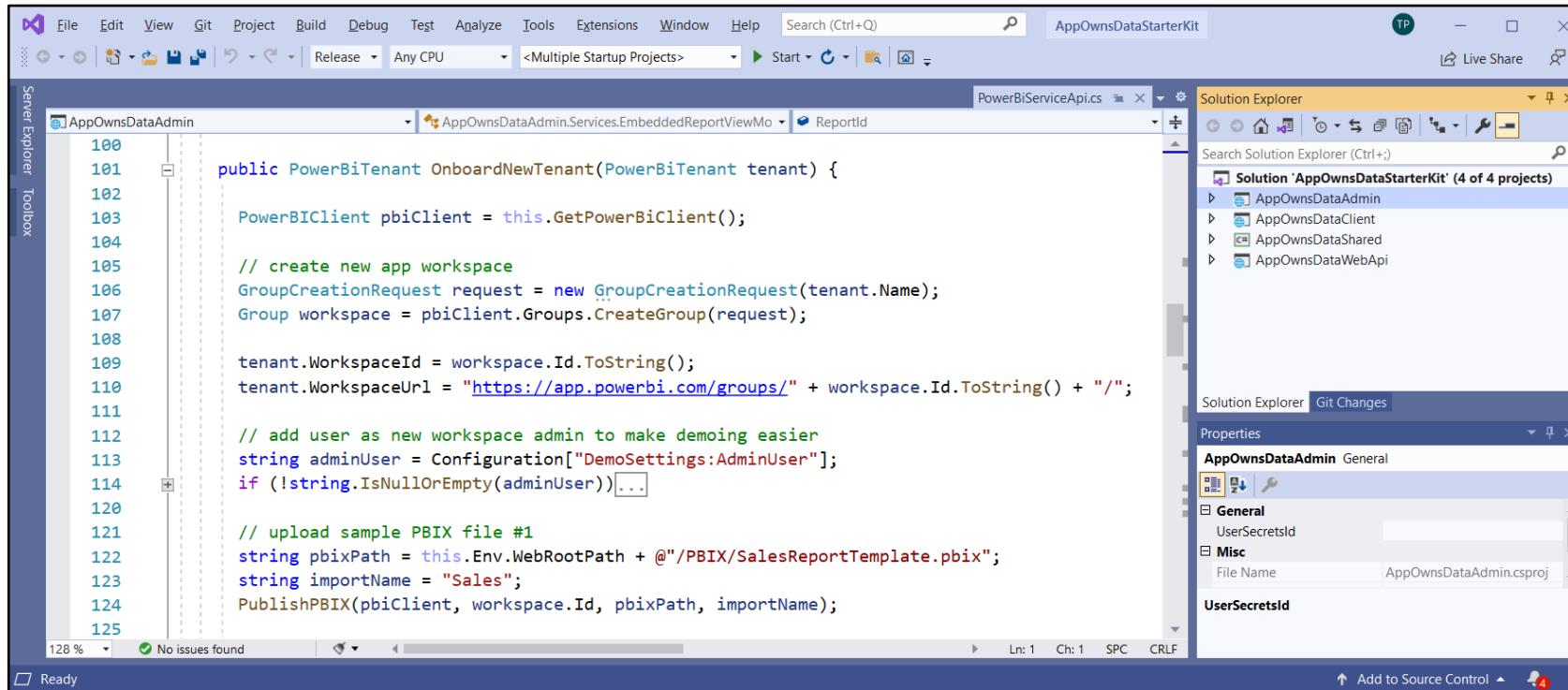
Readme MIT License

Releases

No releases published

The App-Owns-Data Starter Kit Solution

- Created as POC to provide starting point for ISVs and large organizations
 - Built using **.NET 5** and **ASP.NET CORE**
 - Can be opened and tested with **Visual Studio 2019** or **Visual Studio Code**
 - Designed for simple App-Owns-Data scenario – extend it to meet your requirements

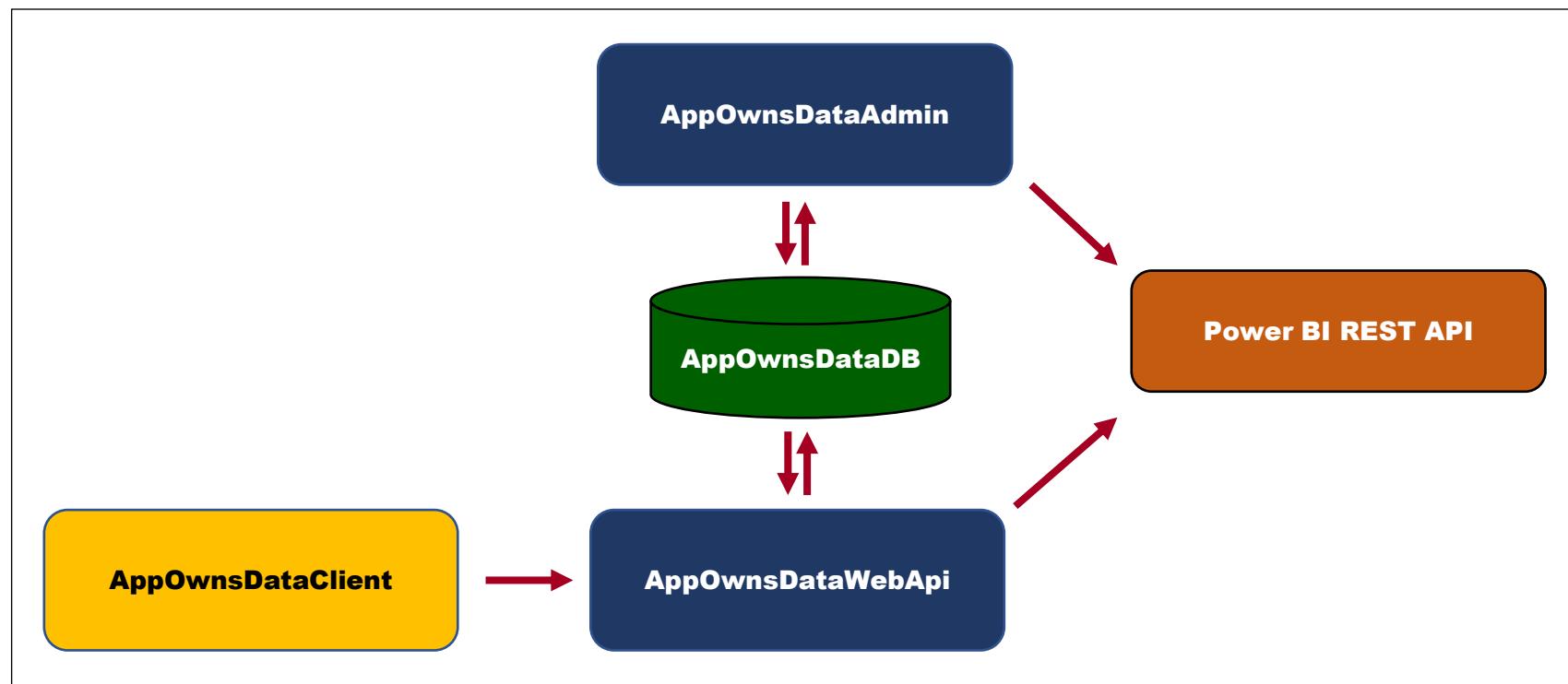


The screenshot shows the Visual Studio 2019 interface with the 'AppOwnsDataStarterKit' solution open. The 'PowerBiServiceApi.cs' file is the active code editor, displaying C# code for managing Power BI tenants. The 'Solution Explorer' window on the right lists four projects: 'AppOwnsDataAdmin', 'AppOwnsDataClient', 'AppOwnsDataShared', and 'AppOwnsDataWebApi'. The 'Properties' window is also visible, showing settings for the 'AppOwnsDataAdmin' project. The status bar at the bottom indicates 'Ready'.

```
100 public PowerBiTenant OnboardNewTenant(PowerBiTenant tenant) {
101
102     PowerBIClient pbiClient = this.GetPowerBiClient();
103
104     // create new app workspace
105     GroupCreationRequest request = new GroupCreationRequest(tenant.Name);
106     Group workspace = pbiClient.Groups.CreateGroup(request);
107
108     tenant.WorkspaceId = workspace.Id.ToString();
109     tenant.WorkspaceUrl = "https://app.powerbi.com/groups/" + workspace.Id.ToString() + "/";
110
111     // add user as new workspace admin to make demoing easier
112     string adminUser = Configuration["DemoSettings:AdminUser"];
113     if (!string.IsNullOrEmpty(adminUser))...
114
115
116     // upload sample PBIX file #1
117     string pbixPath = this.Env.WebRootPath + @"/PBIX/SalesReportTemplate.pbix";
118     string importName = "Sales";
119     PublishPBIX(pbiClient, workspace.Id, pbixPath, importName);
120
121
122
123
124
125 }
```

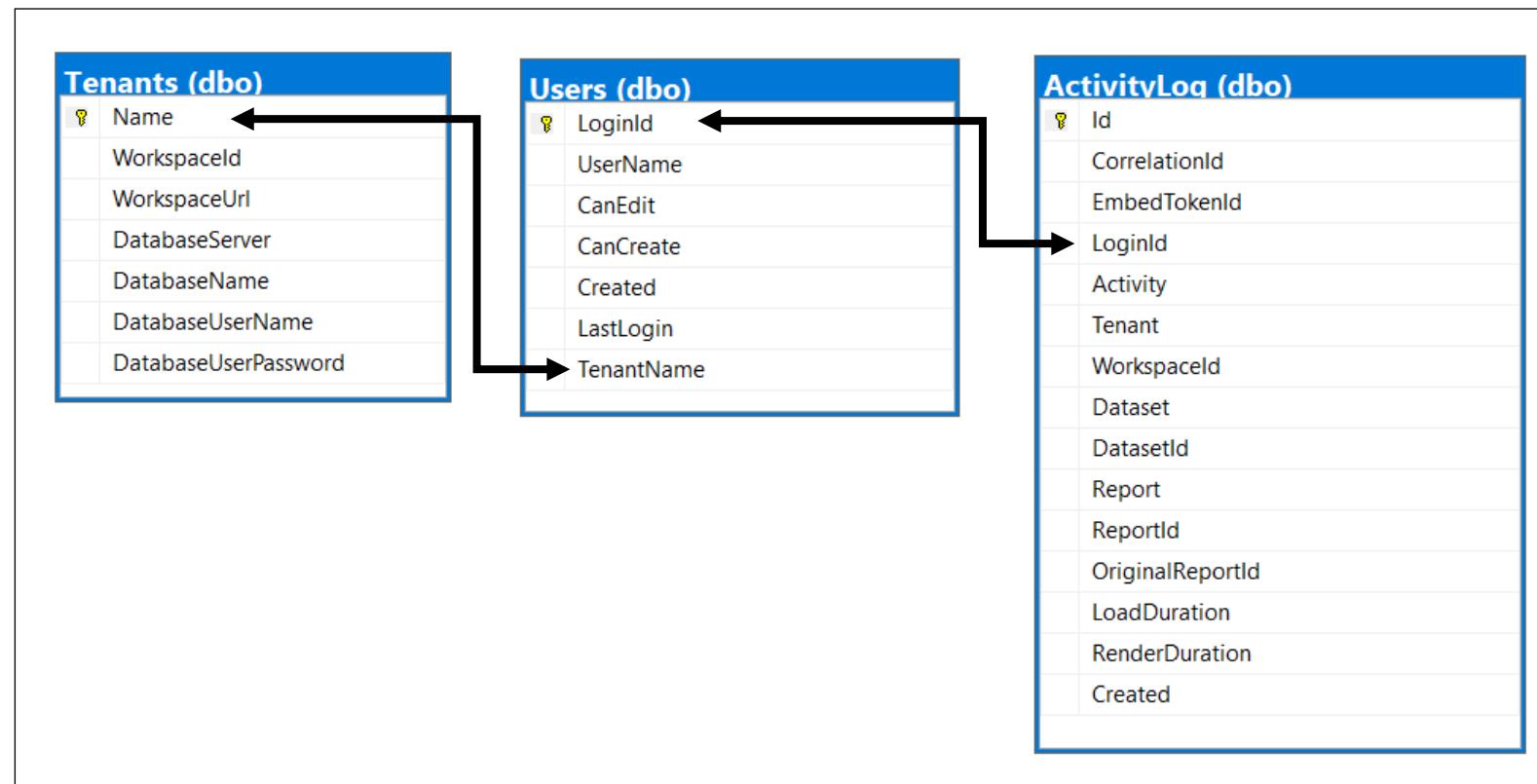
Solution Architecture

- **AppOwnsDataDB**: custom database to track tenants, user permissions and user activity
- **AppOwnsDataAdmin**: administrative app to create tenants and manage user permissions
- **AppOwnsDataClient**: customer-facing SPA used to view and author reports
- **AppOwnsDataWebApi**: custom Web API used by the AppOwnsDataClient application



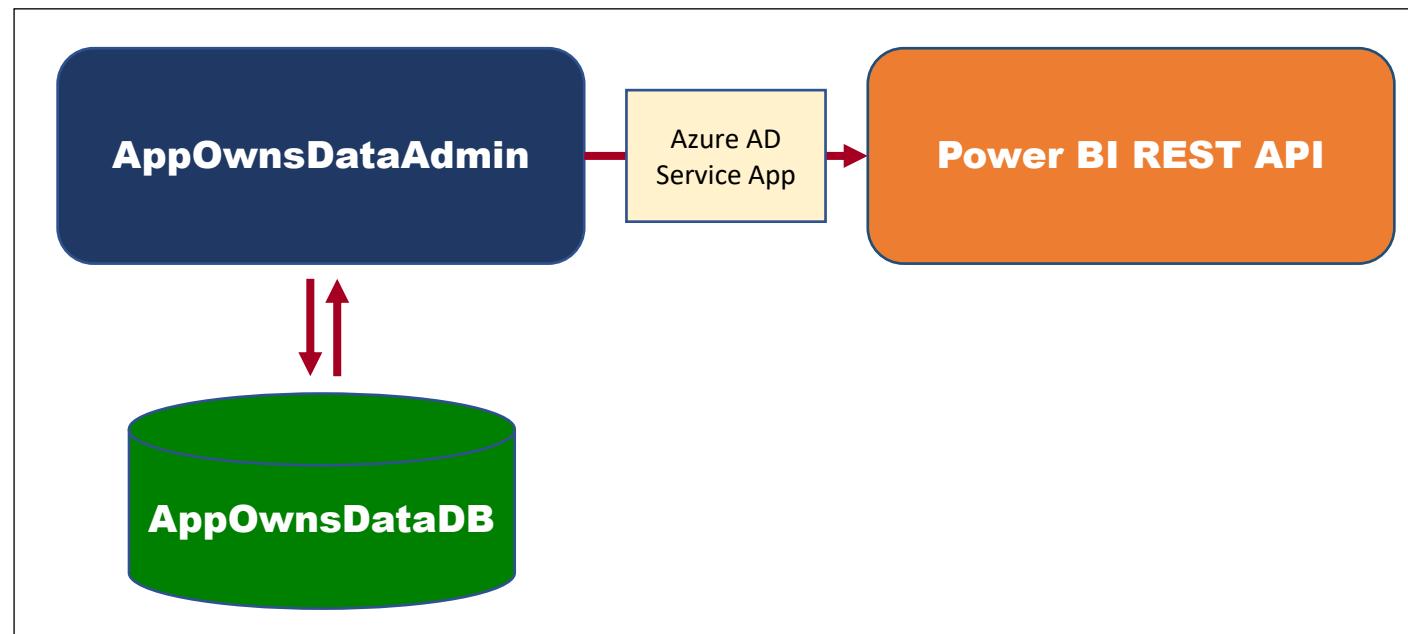
Database Schema for AppOwnsDataDB

- **Tenants** table tracks Power BI workspace Id and customer database connection info
- **Users** table tracks user profile with tenant assignment and permissions within tenant
- **ActivityLog** table tracks telemetry data about user activity and report performance



AppOwnsDataAdmin

- Uses Power BI REST API to provision workspaces for customer tenants
- Calls to Power BI REST API as a service principal – not as a user
- Adds a record to **AppOwnsDataDB** for each customer tenant



Creating Customer Tenants

- **AppOwnsDataAdmin** provides **Onboard New Tenant** page
 - Behind the scenes, **AppOwnsDataAdmin** creates and configures Power BI workspace
 - Makes it possible to configure details for connection to customer database
 - Tenants can be viewed or deleted

Onboard New Tenant

Tenant Name:	Wingtip
Database Server Name:	devcamp.database.windows.net
Database Name:	WingtipSales
SQL Server User Name:	CptStudent
SQL Server User Password:	*****

Create New Tenant



Power BI Tenants

Tenant	Workspace ID	Embed	Web URL	View	Delete
Acme Corp	16a1acfd-e1c1-40ec-9b1a-d00b228fd5cd				
Contoso	cee25f0e-3d3a-4dee-bdfa-8821423761e1				
Mega Corp	a6be93a4-65b1-4019-81cc-46791ede9db4				
Wingtip	7e45fc73-edd5-4af8-ae87-82d04481c7d6				

Sequence of Tenant Provisioning Operations

- Create a new Power BI workspace
- Assign workspace to a dedicated capacity
- Import template PBIX files to create datasets and reports
- Update connection string to redirect dataset to customer's database
- Patch datasource credentials
- Start dataset refresh operations

Managing User Permissions

- **Edit User** page make it possible to assign user to a customer tenant
- Users can optionally be assigned **edit permissions** and **create permissions**

Users								
 Create New User								
Login ID	User Name	Last Login	Tenant	Can Edit	Can Create	View	Edit	Delete
AustinP@powerbidevcamp.net	Austin Powers	5/2/2021 4:26 PM	unassigned	False	False			
ted@tedpattison.net	Ted Pattison	4/29/2021 7:14 PM	Wingtip	True	True			

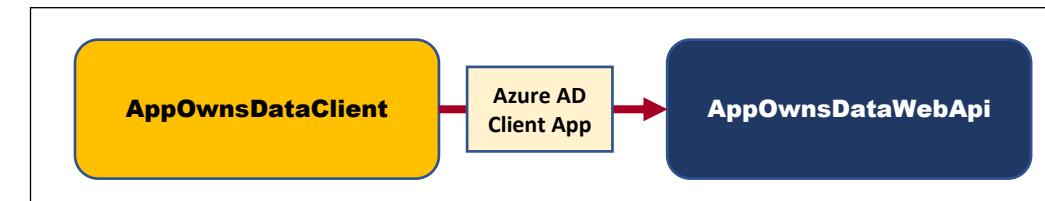
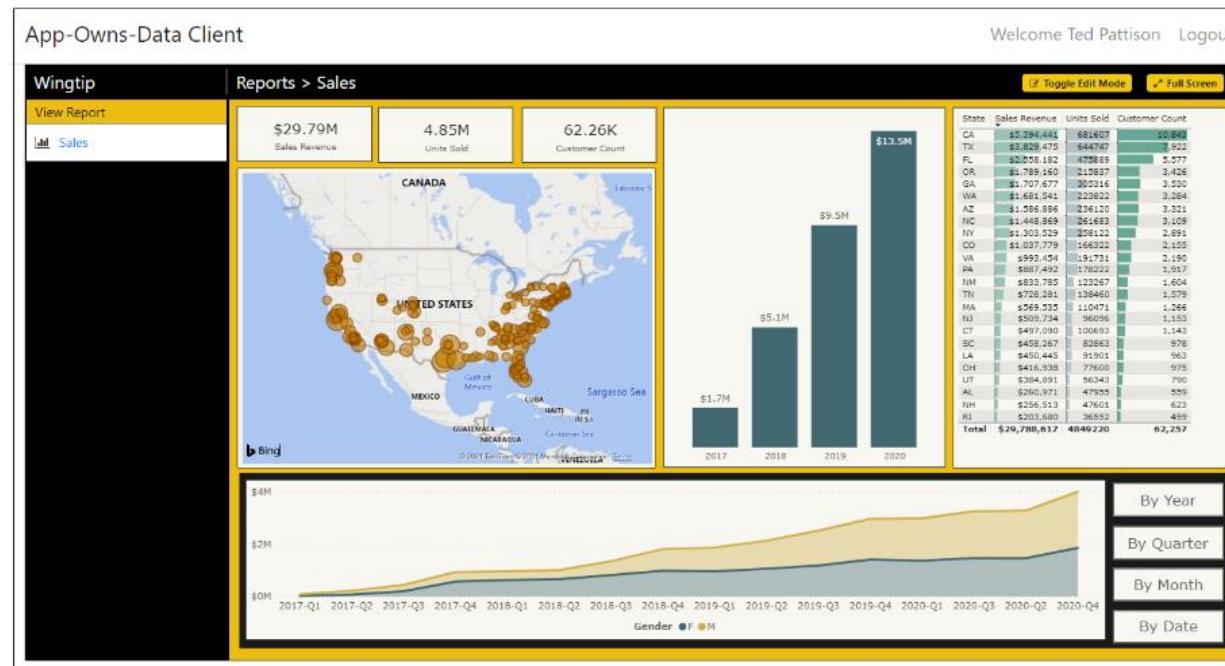
 **Edit User**

Login Id:	AustinP@powerbidevcamp.net
Last Login:	5/2/2021 4:26:14 PM
User Name:	Austin Powers
Home Tenant:	Wingtip
Can Edit:	<input checked="" type="checkbox"/>
Can Create:	<input type="checkbox"/>

 Save

AppOwnsDataClient

- Built as simple **Single Page Application (SPA)**
- Client **authenticates user** and **acquires access tokens** to call **AppOwnsDataWebApi**
- Project configured with **node.js** and **webpack** to support **TypeScript** compilation
- Project includes node.js packages for **jQuery** and **Power BI JavaScript API**
- Project easily extended to support development with **React** or **Angular**



Report Authoring

- **AppOwnsDataClient** demonstrates report authoring features
 - Users with **edit permissions** can move report into edit view and customize layout
 - Users with **create permissions** can copy existing reports and create new reports

The screenshot shows the 'App-Owning Data Client' interface. On the left, there's a sidebar with a dark background labeled 'Wingtip' and a yellow 'View Report' button. The main area is titled 'Reports > Sales' and displays three visualizations: a map of North America with sales points, a bar chart of sales by year (2017-2020), and a line chart of sales over time by gender. Top-level metrics are shown in boxes: '\$29.79M Sales Revenue', '4.85M Units Sold', and '62.25K Customer Count'. To the right, a large white arrow points from the top right towards the 'Toggle Edit Mode' button in the top right corner of the main content area. The 'Edit Mode' button has a checked checkbox icon. The interface includes a 'Welcome Ted Pattison Logout' header, a toolbar with various icons, and a sidebar on the right containing sections for 'Visualizations', 'Fields', 'Filters', 'Values', 'Drill through', 'Cross-report', 'Keep all filters', and 'Add drill-through fields here'.

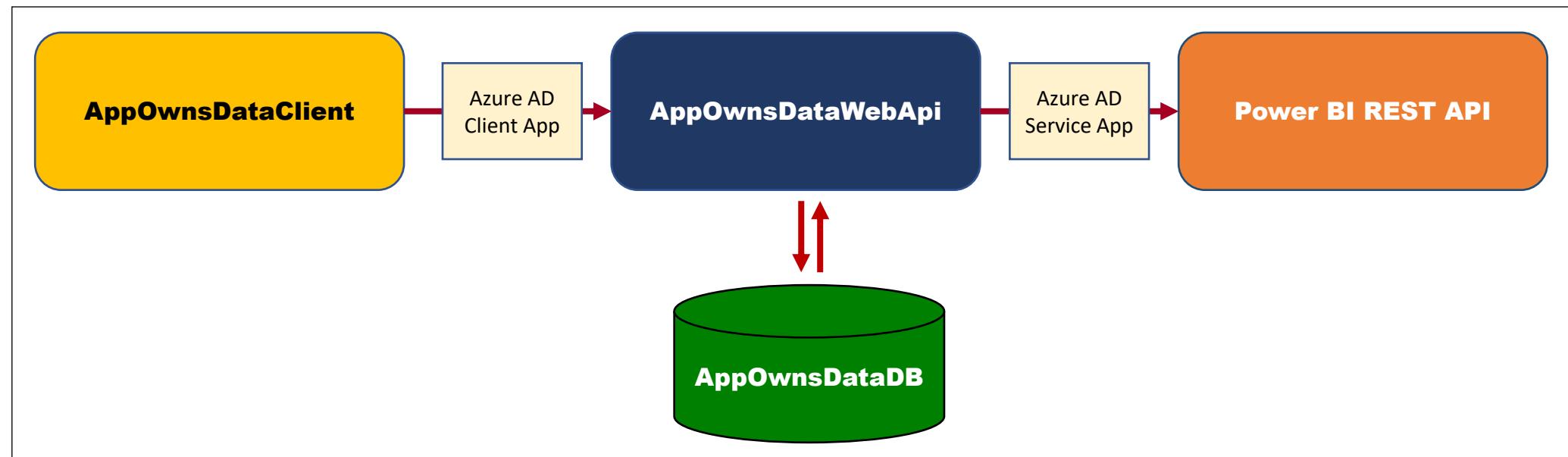
Responsive Design

- Supporting mobile devices is a common application requirement
 - **AppOwnsDataClient** demonstrates **responsive design strategy**
 - Power BI report template designed with **master view** and **mobile view**
 - Client app has logic to switch between master view and mobile view



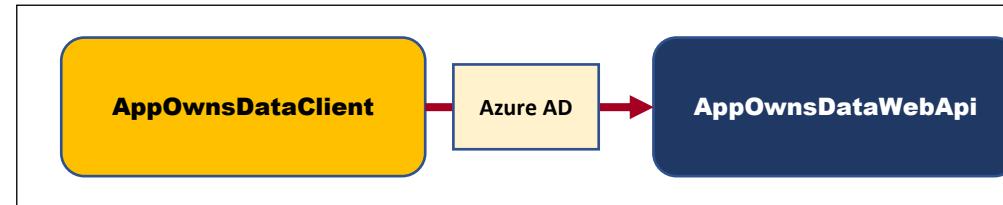
AppOwnsDataWebApi

- Provides service-oriented architecture endpoints for **AppOwnsDataClient**
 - Provides endpoint to **process user login**
 - Provides endpoint to **retrieve embedding data and embed tokens**
 - Provides endpoint to **log activity events** to be recorded in **AppOwnsDataDB**

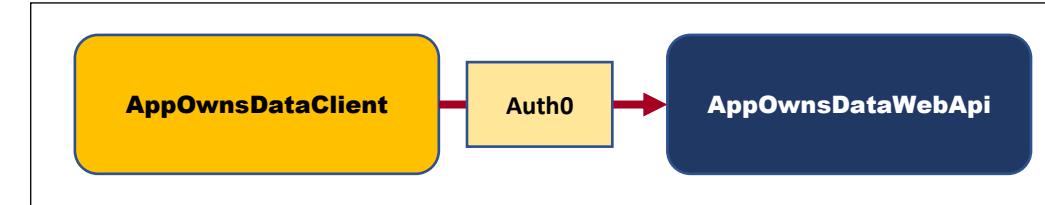
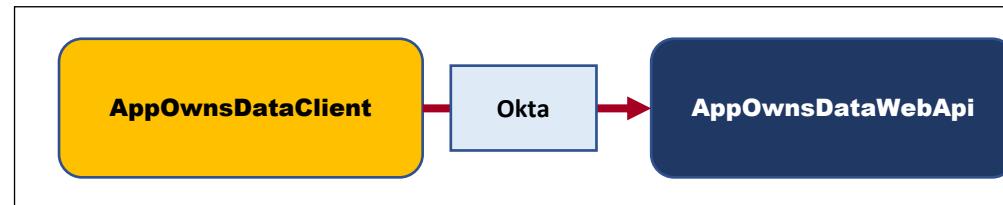
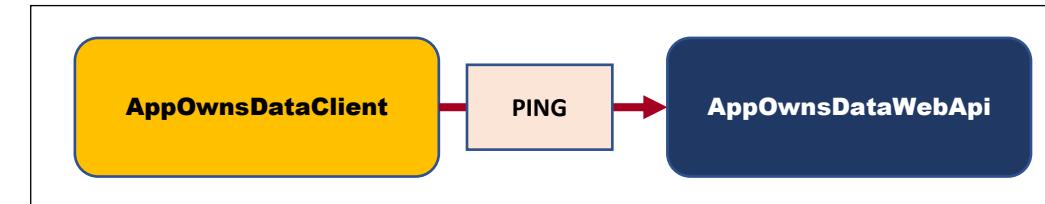
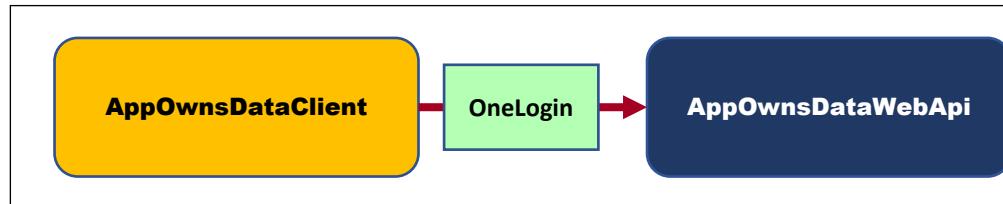


Identity Providers and App-Owns-Data Development

- App-Owns-Data development allows you to pick any identity provider
 - Identity provider responsible for authenticating user and validating **LoginId**
 - App-Owns-Data Starter Kit uses Azure AD as client app identity provider

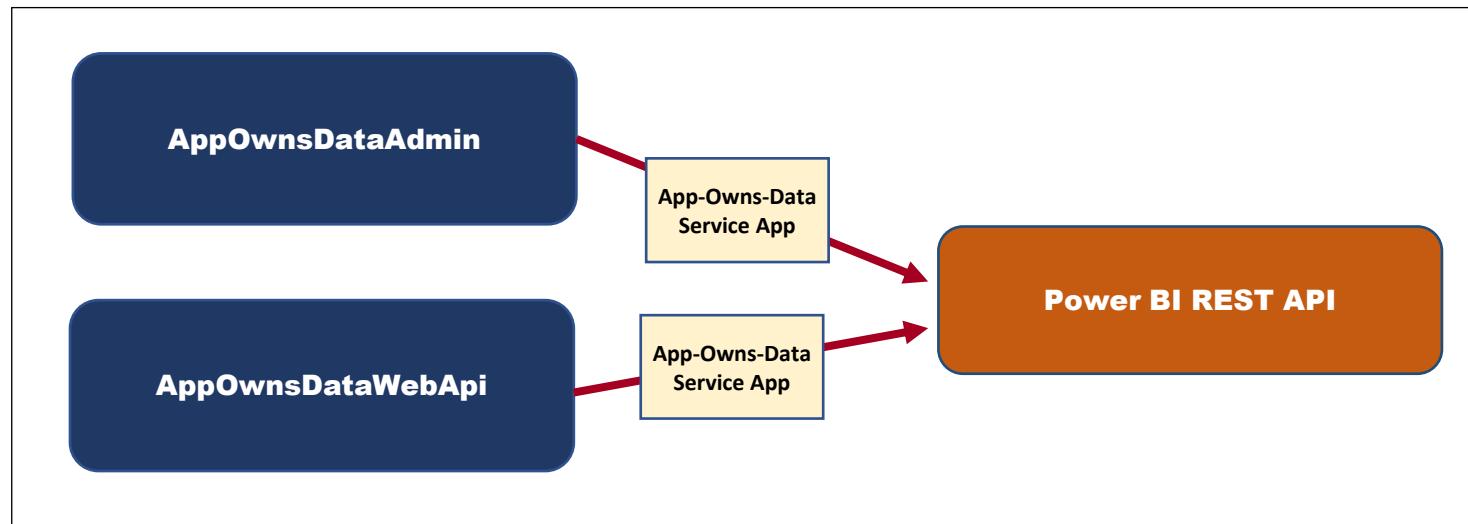


- Solution architecture designed to facilitate switching out the identity provider



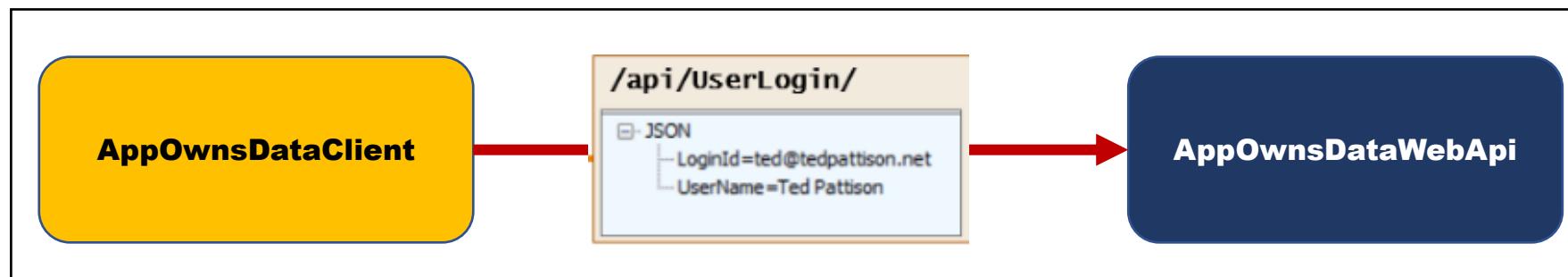
Using a single service principal

- **AppOwnsDataAdmin** calls Power BI REST API to **create and configure workspaces**
- **AppOwnsDataWebApi** calls Power BI REST API to generate **embed data & embed tokens**
- Both applications run under identity of single service principal
- Service principal has admin permissions on any workspaces that it creates



User Login

- **AppOwnsDataClient** calls **UserLogin** endpoint after authenticating user
- **AppOwnsDataWebApi** updates last login time for existing users
- **AppOwnsDataWebApi** creates new record for first-time users
- New users have no access to content until they are assigned to a customer tenant



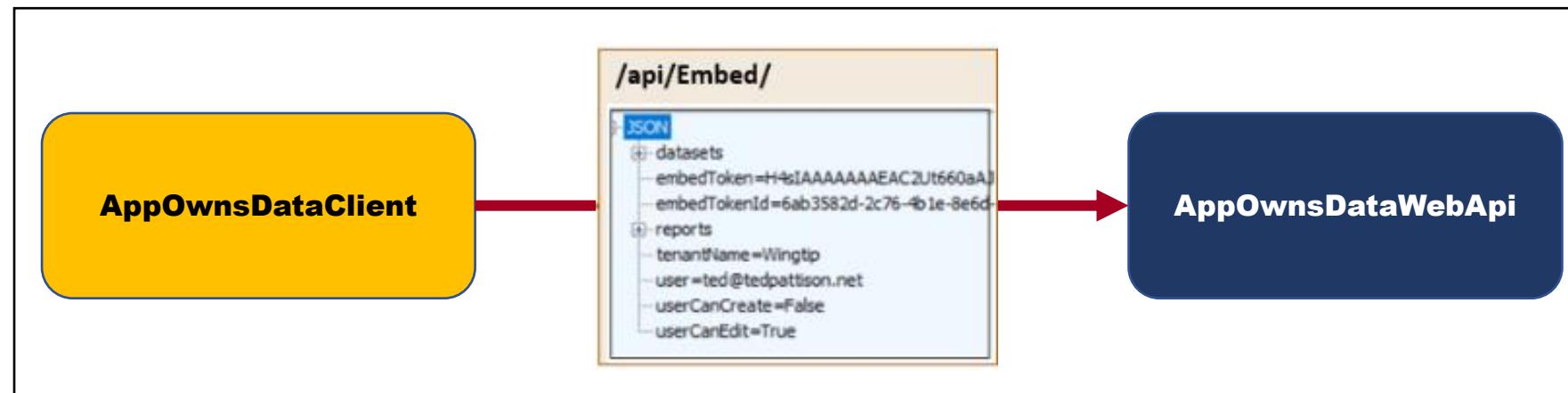
Unassigned users have no access to content

- New users created without being assigned to tenant
- Users see message shown below until they are assigned to tenant

The screenshot shows a web application interface. At the top left, it says "App-Owns-Data Client". At the top right, it says "Welcome Ted Pattison" and "Logout". The main content area has a large orange background. In the center, the text "Thanks for logging in" is displayed in a dark blue font. Below it, a message in a smaller dark blue font reads: "Your user account has no assigned tenant. Once your user account has been assigned to a tenant, you will be able to use this application and interact with embedded report from Power BI."

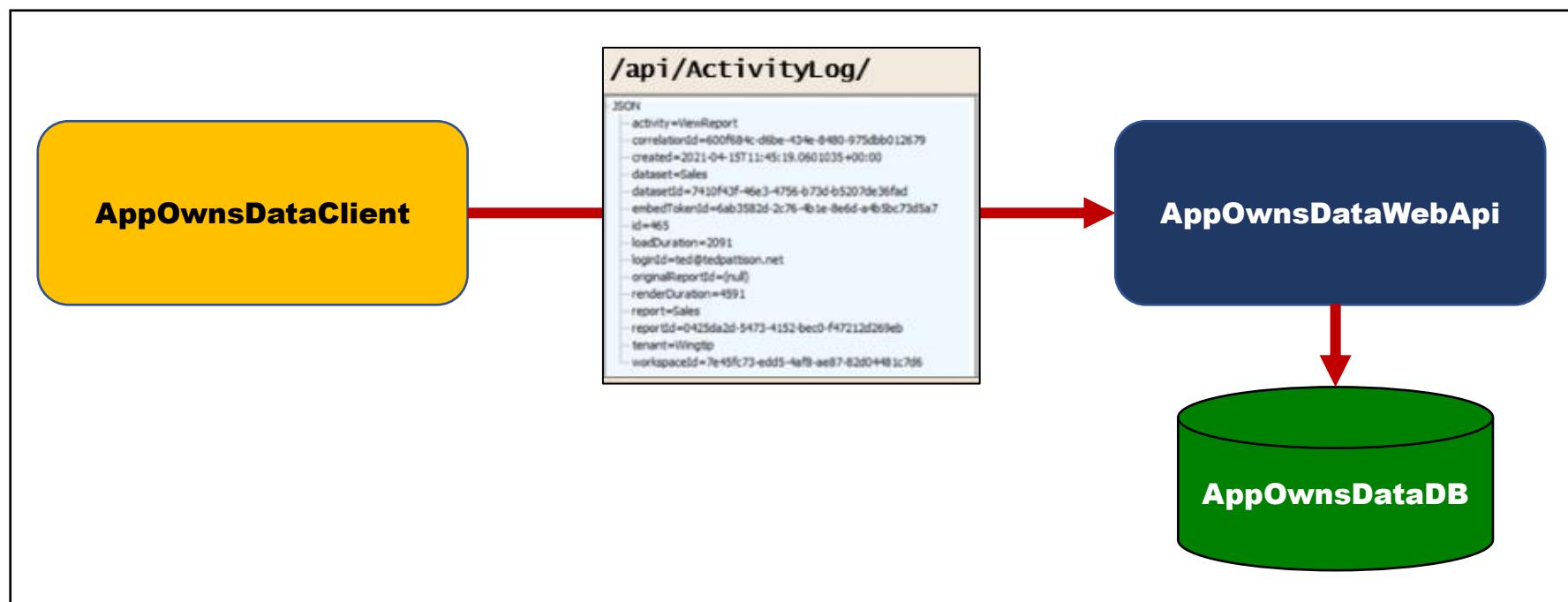
Embedding Data View Model

- **AppOwnsDataWebApi** returns view model to **AppOwnsDataClient**
 - View model contains embedding data for reports and datasets
 - View model contains embed token used to embed reports
 - Embed tokens contains set of permissions generated for current user



Adding a custom telemetry layer

- App-Owns-Data Starter Kit demonstrates adding custom telemetry
 - **AppOwnsDataWebApi** exposes **ActivityLog** endpoint
 - **AppOwnsDataClient** logs custom events by posting to **ActivityLog** endpoint
 - **AppOwnsDataWebApi** records event activity in **AppOwnsDataDB**



Monitoring User Activity

- Activity log data stored in **ActivityLog** table of **AppOwnsDataDB**
 - Makes it possible to monitor user activity with Power BI Desktop project
 - App-Owns-Data Starter Kit provides PBIX template for monitoring & analysis

UserName	Created	Tenant	WorkspaceId	LoginId	Activity	Report
Austin Powers	2021-04-19 6:44:06 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales
Benny Austin	2021-04-19 6:47:38 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales
Slava Trofimov	2021-04-19 6:54:54 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	EditReport	Sales
Ted Pattison	2021-04-19 6:55:00 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales
	2021-04-19 7:07:14 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	CreateReport	Sales by Year and Quarter
	2021-04-19 7:07:17 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales by Year and Quarter
	2021-04-19 7:09:30 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales
	2021-04-19 7:09:48 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales by Year and Quarter
	2021-04-19 7:09:52 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales
	2021-04-20 8:57:11 AM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales
	2021-04-20 8:57:33 AM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales by Year and Quarter

Monitoring Report Performance

- **ViewReport** activity logged with perf data for **LoadDuration** and **RenderDuration**
- Allows for monitoring/analysis of report performance across multi-tenant environment
- Allows for early detection of performance problems affecting user experience

Tenant	Dataset	Report	Report Views	Avg Load Time	Avg Render Time
Customer 123	Sales	Sales	18	1.40	2.69
Wingtip	Sales	Sales	317	1.56	2.63
Ofer INC	Sales	Sales	8	1.42	2.56
Mega Corp	Sales	Sales	11	1.29	2.14
Acme Corp	Sales	Sales	74	1.06	1.78
Contoso	Sales	Sales	1	0.82	1.68
Wingtip	Sales	Report 2	1	0.90	1.64
Wingtip	Sales	Sales in FL	8	0.73	1.58
Acme Corp	Sales	Sales in FL	5	0.85	1.52
Customer 123	Sales	Report	3	1.07	1.36

Agenda

- ✓ Solution Architecture
- Set up your development environment
 - Open the App-Owns-Data Starter Kit in Visual Studio
 - Test the AppOwnsDataAdmin application
 - Test the AppOwnsDataClient application
 - Use activity logs to monitor user activity and report performance

Setting up your development environment

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 named App-Owns-Data Service App
4. Create the Azure AD Application named App-Owns-Data Client App

Create the Power BI Apps Group in Azure AD

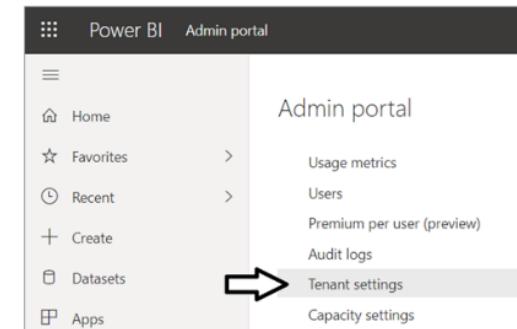
- Create a new Azure AD security group named **Power BI Apps**
 - Power BI Apps group used to configure Power BI access for service principal

The screenshot shows the 'Groups | All groups' page in the Azure Active Directory portal for the 'Embed Mart - Azure Active Directory' tenant. The left sidebar has links for 'All groups', 'Deleted groups', and 'Diagnose and solve problems'. The main area displays a table with columns: Name, Object Id, and Group Type. A red arrow points to the 'Power BI Apps' row, which is highlighted with a red background. The 'Power BI Apps' row contains the following data:

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

Configure service principal support in Power BI

- Update two tenant-level settings in Power BI Admin portal
 - Configure **Allow service principals to use Power BI APIs**
 - Configure **Workspace settings > Create workspaces**



The screenshot shows the 'Allow service principals to use Power BI APIs' configuration page. It includes:

- A title section with a toggle switch labeled 'Enabled'.
- A 'Apply to:' section with a radio button selected for 'Specific security groups (Recommended)'.
- A 'Power BI Apps' input field with a 'Enter security groups' placeholder, preceded by a red arrow.
- A checkbox for 'Except specific security groups'.
- Buttons for 'Apply' and 'Cancel' at the bottom.

Red arrows point from the 'Enabled' switch, the 'Specific security groups' radio button, and the 'Enter security groups' input field towards the corresponding sections in the second screenshot.

The screenshot shows the 'Create workspaces' configuration page. It includes:

- A title section with a toggle switch labeled 'Enabled'.
- A note about workspace creation permissions in a yellow box.
- A 'Apply to:' section with a radio button selected for 'Specific security groups'.
- A 'Power BI Apps' input field with a 'Enter security groups' placeholder, preceded by a red arrow.
- A checkbox for 'Except specific security groups'.
- Buttons for 'Apply' and 'Cancel' at the bottom.

Red arrows point from the 'Enabled' switch, the 'Specific security groups' radio button, and the 'Enter security groups' input field towards the corresponding sections in the first screenshot.

Create the App-Owns-Data Service App

- Understanding the **App-Owns-Data Service App**
 - Azure AD application used to authenticate as service principal
 - Used by both **AppOwnsDataAdmin** and **AppOwnsDataWebApi**
 - You need to record **Tenant ID**, **Client ID** and **Client Secret** when creating application
 - **Tenant ID**, **Client ID** and **Client Secret** used to implement client credentials flow

The screenshot shows the Azure portal interface for managing an application. The title bar says "App-Owns-Data Service App". Below it is a search bar and navigation links for "Delete", "Endpoints", and "Preview features". On the left, there's a sidebar with "Overview", "Quickstart", "Integration assistant", "Manage", and "Branding". The main content area displays the "Essentials" section with the following details:

- Display name : App-Owns-Data Service App
- Application (client) ID : c25919d8-1906-4382-a168-b49dbf476154
- Directory (tenant) ID : 9fa52d4b-075e-41ad-bba7-a85e9bddad18
- Object ID : 48f7a647-bf77-4579-9538-9e7fcf8f5230

The screenshot shows a "Notepad" window titled "*Untitled - Notepad" with the following text:

```
File Edit Format View Help
App-Owns-Data Service App
-----
Tenant ID:
9fa52d4b-075e-41ad-bba7-a85e9bddad18

Client ID:
c25919d8-1906-4382-a168-b49dbf476154

Client Secret:
w0595Td7wFGP[REDACTED]F0
```

A large white arrow points from the bottom-left towards the "Client Secret" line.

TIP: Use client secret when developing – move to client certificates for production

Add the Azure AD App to Power BI Apps Group

- Add **App-Owns-Data Service App** to **Power BI Apps** group
 - This gives service principal ability to call Power BI Service API
 - This also gives service principal ability to create new Power BI workspaces
 - Service principal is automatically admin of any workspace it creates

The screenshot shows the 'Power BI Apps | Members' page in the Microsoft Azure portal. The left sidebar has a 'Members' tab selected. The main area displays a table titled 'Direct members' with one row:

Name	Type	Email
App-Owns-Data Service App	Service Principal	

A large white arrow points from the text 'Add the Azure AD App to Power BI Apps Group' in the slide content to the 'Members' tab in the screenshot.

Create the App-Owns-Data Client App

- Azure AD application used to authenticate user of **AppOwnsDataClient**
 - Users can authenticate using any Microsoft organizational or personal account
 - Authentication allows **AppOwnsDataWebApi** to validate **LoginId** of current user
 - **AppOwnsDataWebApi** and **AppOwnsDataClient** both configured with this **Client ID**

The image shows two side-by-side screenshots. The left screenshot is the 'Register an application' wizard in the Azure portal. It has three main sections: 1) General settings where 'Name' is set to 'App-Owns-Data Client App' and 'Supported account types' is set to 'Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)'. 2) Redirect URI (optional) where 'Single-page application (SPA)' is selected and the 'Redirect URI' field contains 'https://localhost:44301/'. 3) A footer with a 'Register' button. The right screenshot is a Notepad file containing the following app registration details:

```
*Untitled - Notepad
File Edit Format View Help
App-Owns-Data Service App
-----
Tenant ID: 9fa52d4b-075e-41ad-bba7-a85e9bddad18
Client ID: c25919d8-1906-4382-a168-b49dbf476154
Client Secret: w0595Td7wFGP[REDACTED]F0
-----
App-Owns-Data Client App
-----
Client ID 046a89da-c98e-40f0-a11b-a3d71289556f
```

Remember – you don't have to use Azure AD as the identity provider

Creating a Custom Scope for AppOwnsDataWebApi

- **App-Owns-Data Client App** used to secure a custom Web API
 - Securing Web API with Azure AD application requires creating custom scope
 - Set up requires creating **custom scope** in **App-Owns-Data Client App** named **Reports.Embed**
 - Scope identifier created in the format of **api://[CLIENT_ID]/Reports.Embed**

Add a scope

Scope name * ⓘ
Reports.Embed
api://046a89da-c98e-40f0-a11b-a3d71289556f/Reports.Embed

Who can consent? ⓘ
Admins and users Admins only

Admin consent display name * ⓘ
Allow this app to embed reports

Admin consent description * ⓘ
Allow this app to embed reports

User consent display name ⓘ
Allow this app to embed reports

User consent description ⓘ
Allow this app to embed reports

State ⓘ
Enabled Disabled

Add scope Cancel

Application ID URI api://046a89da-c98e-40f0-a11b-a3d71289556f

Scopes defined by this API

Define custom scopes to restrict access to data and functionality protected by the API. An application that requires access to parts of this API can request that a user or admin consent to one or more of these.

Adding a scope here creates only delegated permissions. If you are looking to create application-only scopes, use 'App roles' and define app roles assignable to application type. [Go to App roles](#).

+ Add a scope

Scopes	Can consent	Admin consent display ...	User consent display na...	State
api://046a89da-c98e-40f0-a11b-a3d71289556f/Report... 	Admins and users	Allow this app to embed r...	Allow this app to embed r...	Enabled

Agenda

- ✓ Solution Architecture
- ✓ Set up your development environment
- Open the App-Owns-Data Starter Kit in Visual Studio
 - Test the AppOwnsDataAdmin application
 - Test the AppOwnsDataClient application
 - Use activity logs to monitor user activity and report performance

Install the Required Developer Software

- .NET 5 SDK

<https://dotnet.microsoft.com/download/dotnet/5.0>

- Node.js

<https://nodejs.org/en/download/>

- Visual Studio 2019

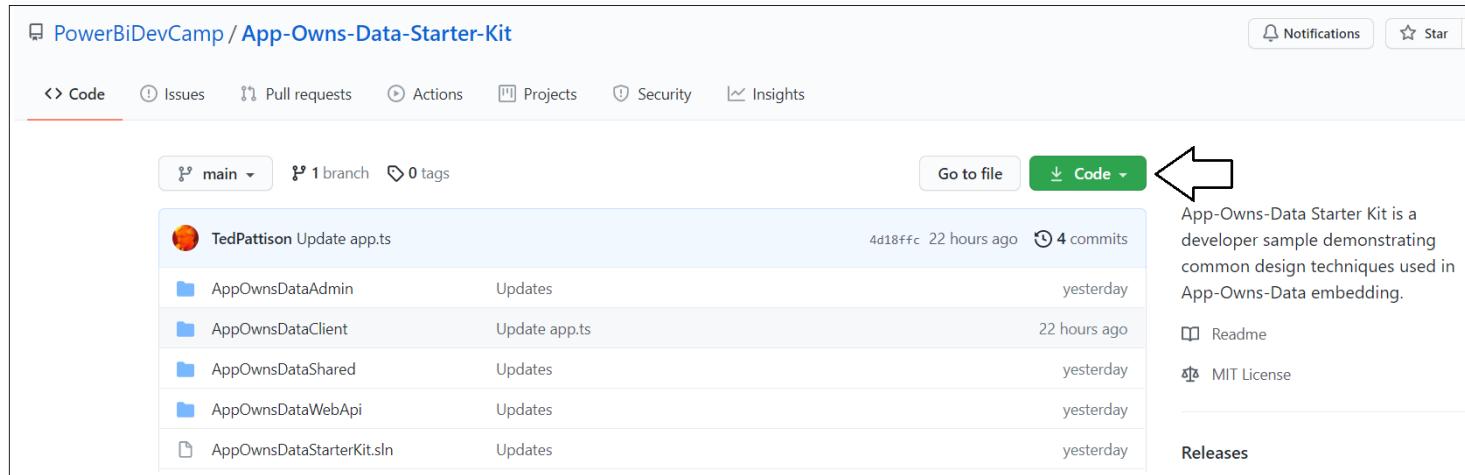
<https://visualstudio.microsoft.com/downloads/>

- Visual Studio Code

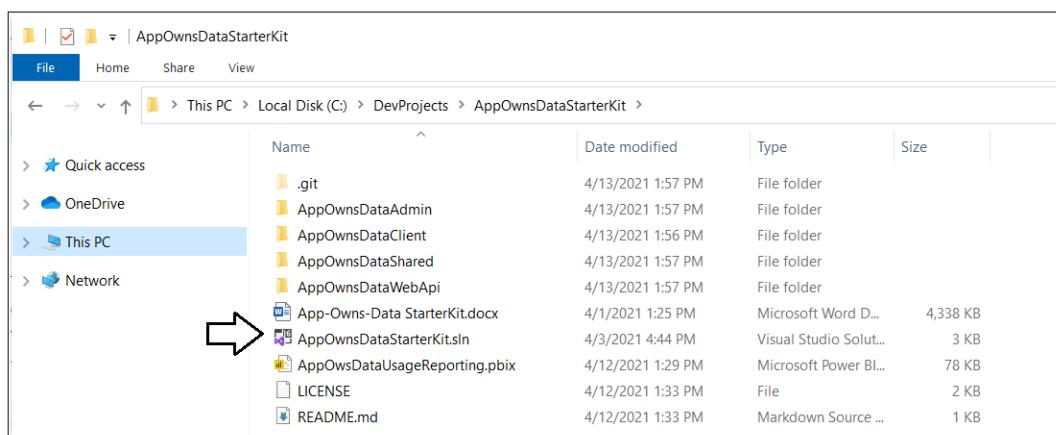
<https://code.visualstudio.com/Download>

Download the source code

- Download the **App-Owns-Data Starter Kit source code** from **GitHub**

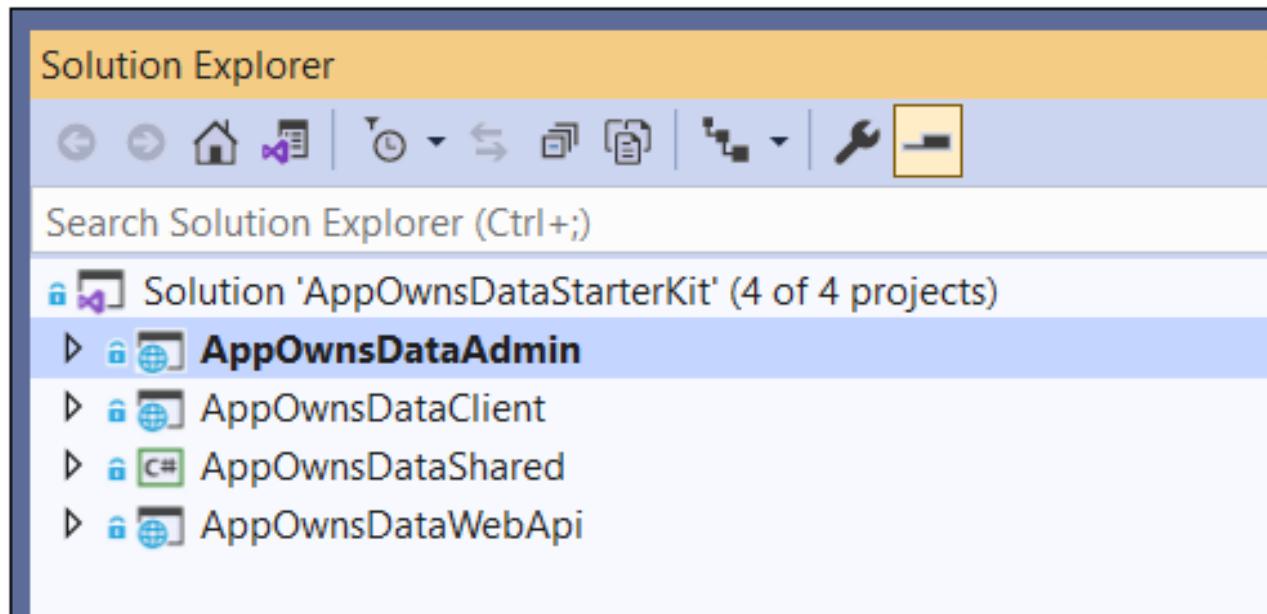


- Top-level folder contains **AppOwnsDataStarterKit.sln** and child folders for each project



Open AppOwnsDataStarterKit.sln in Visual Studio

- **AppOwnsDataAdmin**: ASP.NET MVC Web Application built using .NET 5
- **AppOwnsClient**: SPA built using HTML, CSS and Typescript
- **AppOwnsDataShared**: Class library project used to generate **AppOwnsDataDB**
- **AppOwnsDataWebApi**: ASP.NET Web API used by **AppOwnsDataClient**



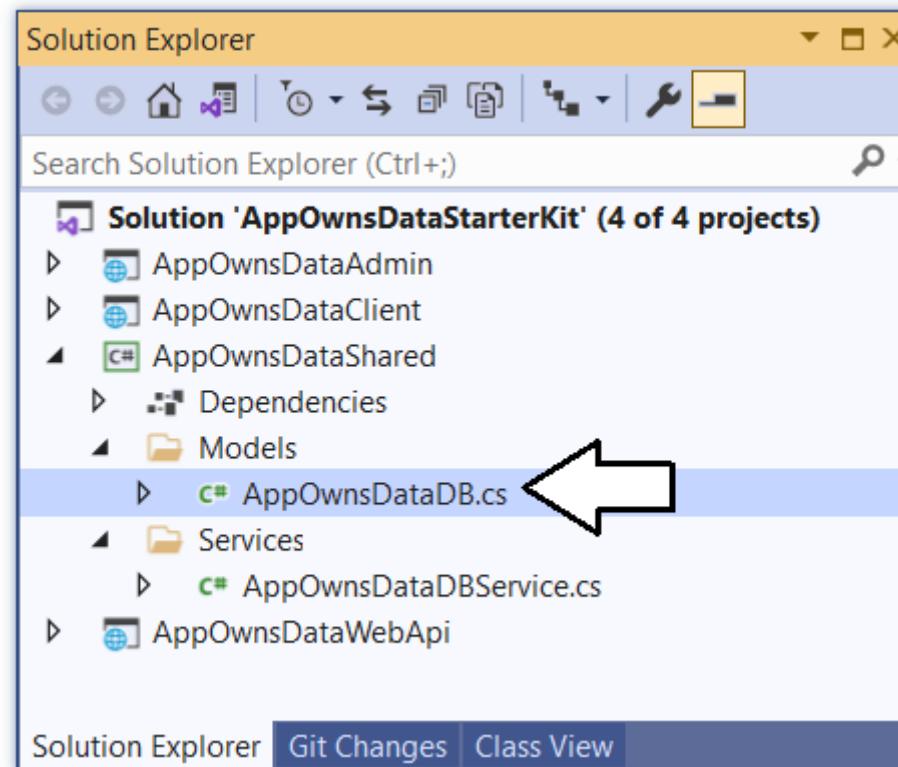
Update appsettings.json in AppOwnsDataAdmin

- Update **AzureAD** section with data from **App-Owns-Data Service App**
- Leave **PowerBi** section with default setting when using Power BI public cloud
- Ensure **AppOwnsDataDB:ConnectionString** is valid for your development environment
- Update **DemoSettings:AdminUser** with your user account name in Power BI tenant



AppOwnsDataShared class library

- Contains shared code for **AppOwnsDataAdmin** and **AppOwnsDataWebApi**
 - Uses Entity Framework Core and C# code-first approach to generate **AppOwnsDataDB**
 - Contains all data access code for executing read/write operations to **AppOwnsDataDB**



Using Code-First to Generate Tables from C# Classes

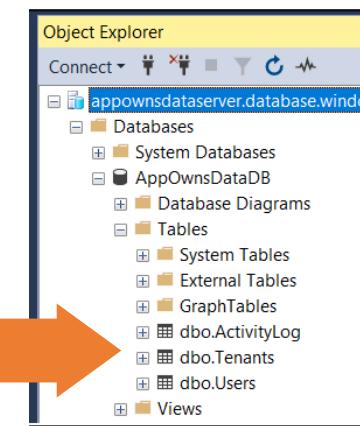
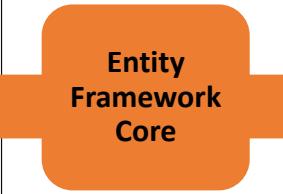
- **AppOwnsDataDB** generated using Entity Framework Core
 - C# classes used to model **Tenants** table, **Users** table and **ActivityLog** table
 - **AppOwnsDataDB** class derives from **DbContext** to define top-level database class

```
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 class User {  
    [Key]  
    public string LoginId { get; set; }  
    public string UserName { get; set; }  
    public bool CanEdit { get; set; }  
    public bool CanCreate { get; set; }  
    public DateTime Created { get; set; }  
    public DateTime LastLogin { get; set; }  
    public string TenantName { get; set; }  
}
```

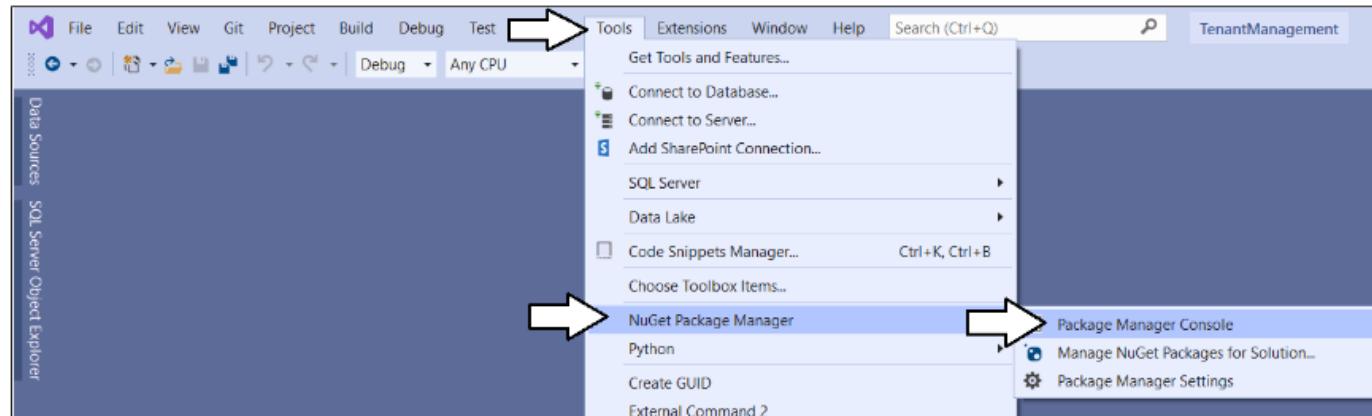
```
public class ActivityLogEntry {  
    public int Id { get; set; }  
    public string CorrelationId { get; set; }  
    public string EmbedTokenId { get; set; }  
    public string LoginId { get; set; }  
    public string Activity { get; set; }  
    public string Tenant { get; set; }  
    public string WorkspaceId { get; set; }  
    public string Dataset { get; set; }  
    public string DatasetId { get; set; }  
    public string Report { get; set; }  
    public string ReportId { get; set; }  
    public string OriginalReportId { get; set; }  
    public int? LoadDuration { get; set; }  
    public int? RenderDuration { get; set; }  
    public DateTime Created { get; set; }  
}
```

```
public class AppOwnsDataDB : DbContext {  
  
    public AppOwnsDataDB(DbContextOptions<AppOwnsDataDB> options) :  
        base(options) {}  
  
    public DbSet<PowerBiTenant> Tenants { get; set; }  
    public DbSet<User> Users { get; set; }  
    public DbSet<ActivityLogEntry> ActivityLog { get; set; }  
}
```

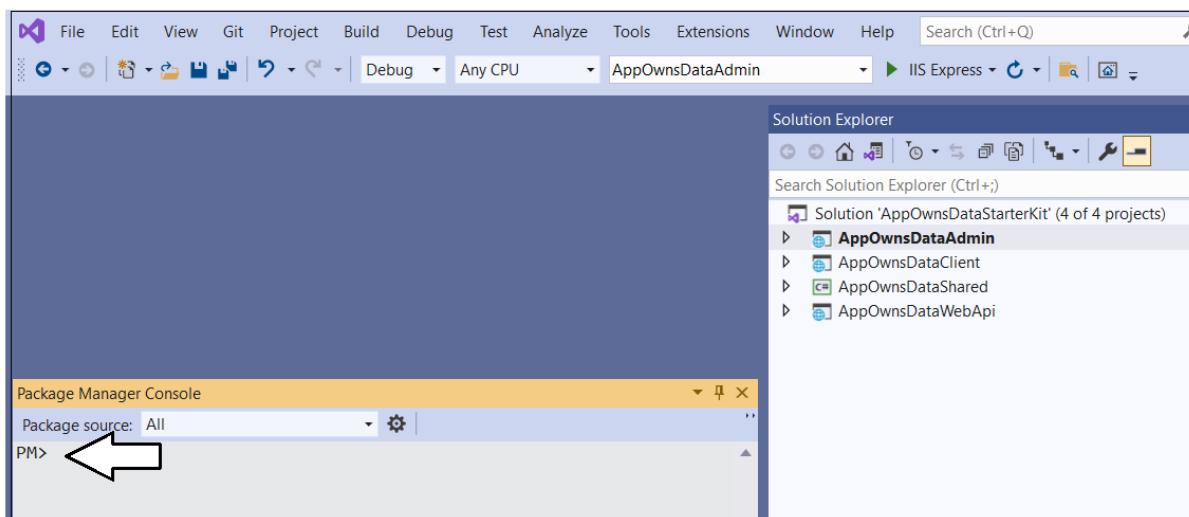


Package Manager Console

- Open the **NuGet Manager Console**



- NuGet Manager Console provides a prompt to type and execute PowerShell commands



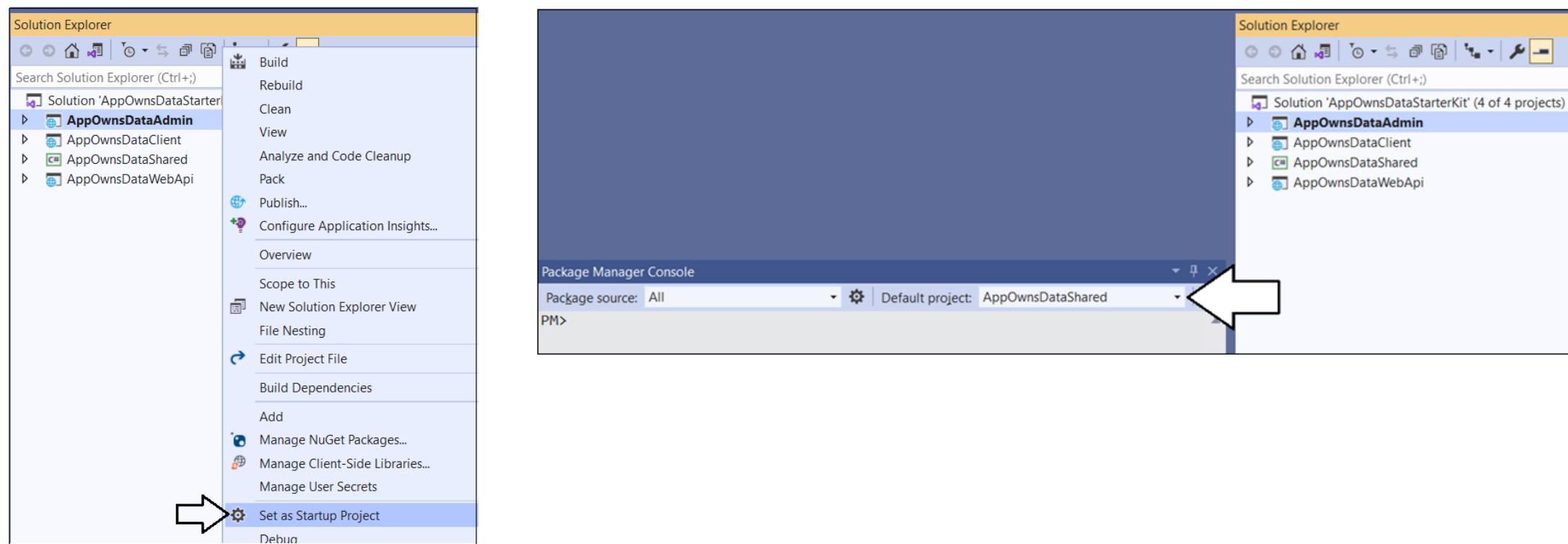
Getting Ready to create the database

1. Set **AppOwnsDataAdmin** as **Startup Project**

- This is required so Entity Framework uses connection string from **appsettings.json**

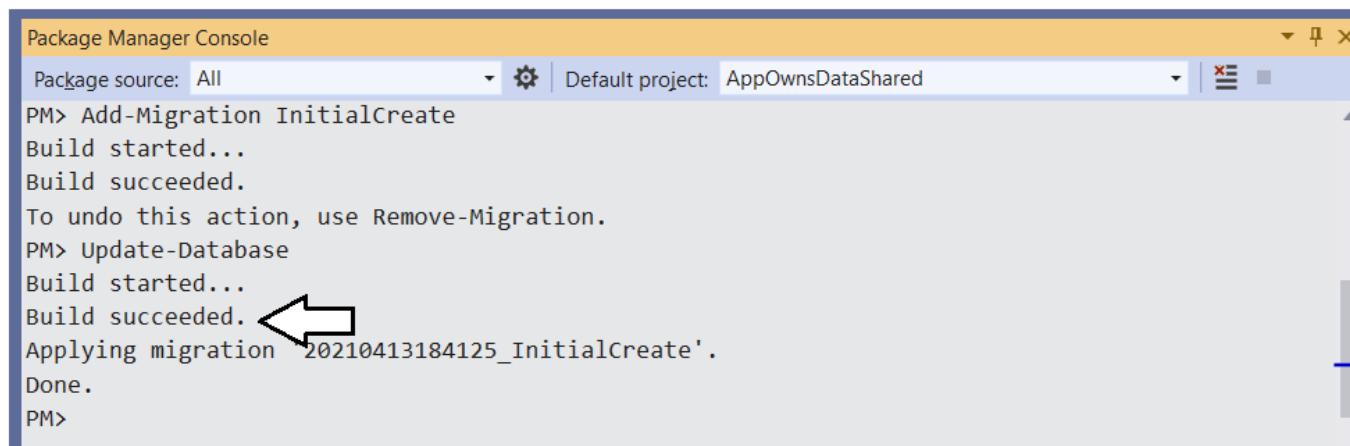
2. Set **AppOwnsDataShared** as **Default project** in Nuget Package console

- This is required so Entity Framework can determine which project contains **AppOwnsDataDB** class

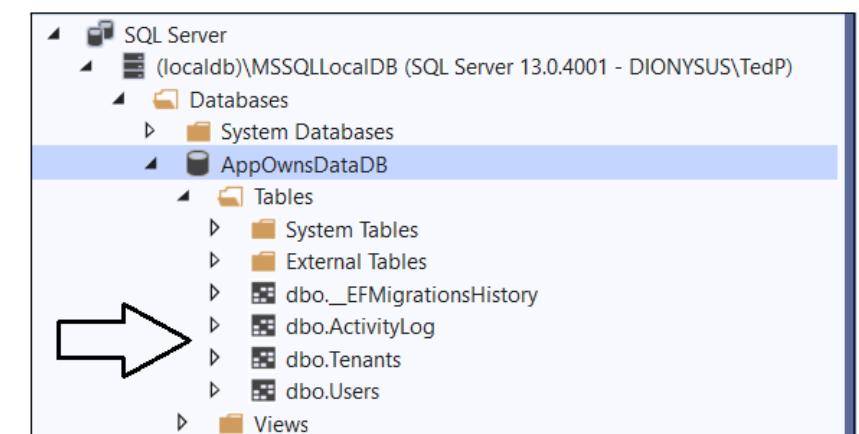


Create the AppOwnsDataDB database

- Execute PowerShell command to generate code to create database
Add-Migration InitialCreate
- Execute PowerShell command to execute code to create the database
Update-Database



```
Package Manager Console
Package source: All | Default project: AppOwnsDataShared
PM> Add-Migration InitialCreate
Build started...
Build succeeded.
To undo this action, use Remove-Migration.
PM> Update-Database
Build started...
Build succeeded. ←
Applying migration '20210413184125_InitialCreate'.
Done.
PM>
```



Agenda

- ✓ Solution Architecture
- ✓ Set up your development environment
- ✓ Open the App-Owns-Data Starter Kit in Visual Studio
- Test the AppOwnsDataAdmin application
 - Test the AppOwnsDataClient application
 - Use activity logs to monitor user activity and report performance

Create New Customer Tenants

- Create a new customer tenant

The screenshot shows the 'Onboard New Tenant' form. It includes fields for Tenant Name (Tenant01), Database Server Name (devcamp.database.windows.net), Database Name (WingtipSales), SQL Server User Name (CptStudent), and SQL Server User Password (pass@word1). A yellow 'Create New Tenant' button is at the bottom right. Two arrows point to the 'Tenant Name' field and the 'Create New Tenant' button.

Tenant Name:	Tenant01
Database Server Name:	devcamp.database.windows.net
Database Name:	WingtipSales
SQL Server User Name:	CptStudent
SQL Server User Password:	pass@word1

Create New Tenant

- You should see the next tenant in the **Power BI Tenants** page

The screenshot shows the 'Power BI Tenants' page with a table. Tenant01 has a Workspace ID of 775ada0a-b353-45e3-a070-8277a557ea17 and Tenant02 has a Workspace ID of b35b6342-6563-46c1-b3e0-0592926e976b. Each row has icons for Embed, Web URL, View, and Delete. Two arrows point to the 'Onboard New Tenant' button and the Tenant01 row.

Power BI Tenants					
Onboard New Tenant					
Tenant	Workspace ID	Embed	Web URL	View	Delete
Tenant01	775ada0a-b353-45e3-a070-8277a557ea17				
Tenant02	b35b6342-6563-46c1-b3e0-0592926e976b				

- Behind the scenes, **AppOwnsDataApp** adds record in **Tenants** table in **AppOwnsDataDB**

The screenshot shows the 'Tenants' table in the AppOwnsDataDB. It contains columns for Name, WorkspaceId, WorkspaceUrl, DatabaseServer, DatabaseName, DatabaseUserName, and DatabaseUserPassword. Tenant01 and Tenant02 are listed with their respective details.

Name	WorkspaceId	WorkspaceUrl	DatabaseServer	DatabaseName	DatabaseUserName	DatabaseUserPassword
Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	https://app.powerbi.com/groups/775ada0a-b353-45e3...	devcamp.database.windows.net	WingtipSales	CptStudent	pass@word1
Tenant02	b35b6342-6563-46c1-b3e0-0592926e976b	https://app.powerbi.com/groups/b35b6342-6563-46c1...	devcamp.database.windows.net	ContosoSales	CptStudent	pass@word1

Tenant Details

- **Tenant Details** page lists members, datasets and reports
 - **AppOwnsDataAdmin** calls Power BI REST API to discover workspace artifacts

Tenant Details

Name: Contoso

Workspace ID: edee34e3-201e-4824-b369-921acdf36583

Workspace URL: <https://app.powerbi.com/groups/edee34e3-201e-4824-b369-921acdf36583/>

Database Server: devcamp.database.windows.net

Database Name: ContosoSales

Database User Name: CptStudent

Database User Password:

Members

Member	Permissions	Member Type
App-Owns-Data Service App	Admin	App
Ted Pattison	Admin	User

Datasets

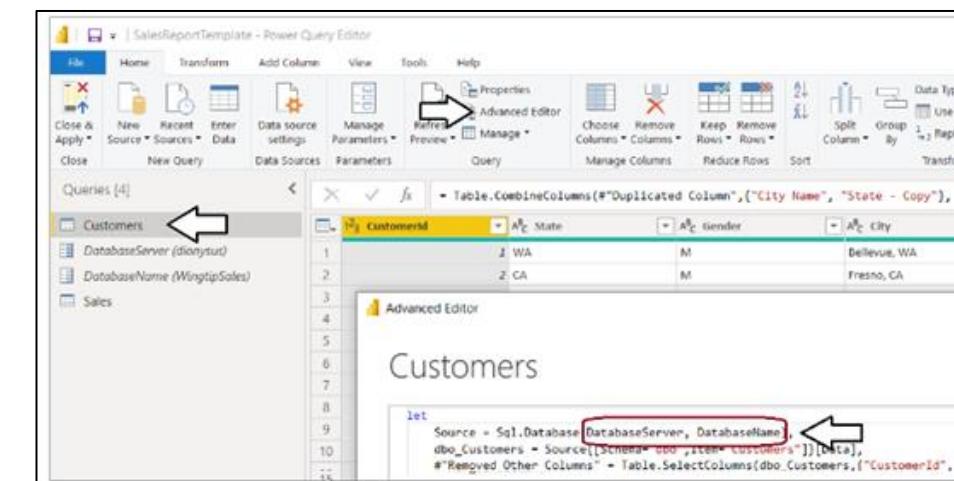
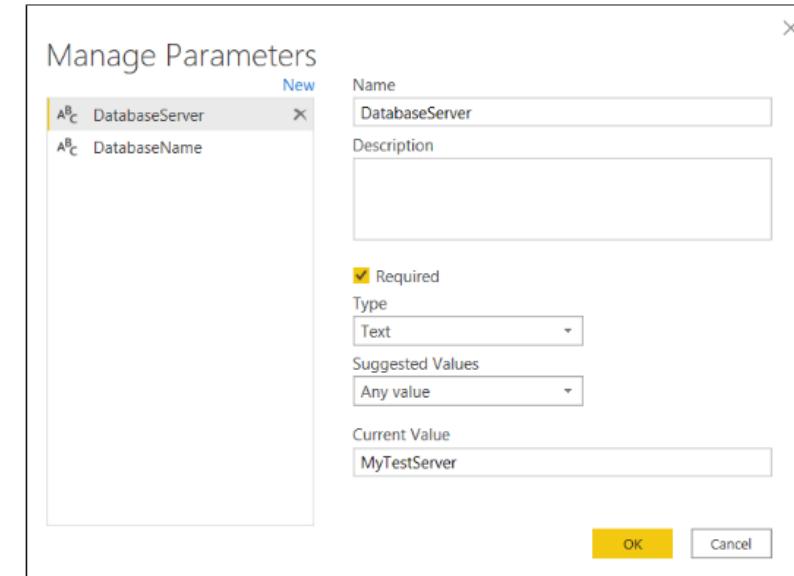
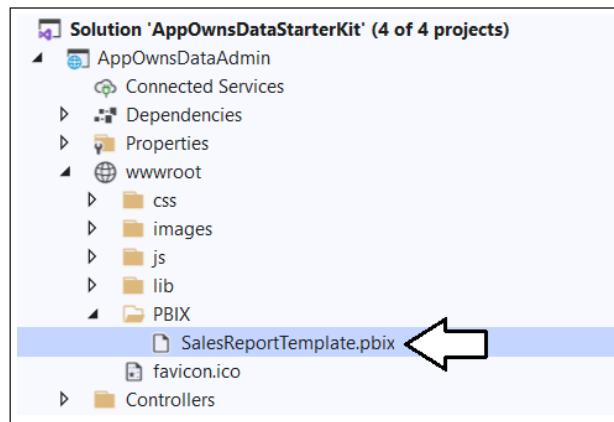
Name	Is Refreshable
Sales	True

Report

Name	Report Type
Sales	PowerBIReport
CA Sales	PowerBIReport

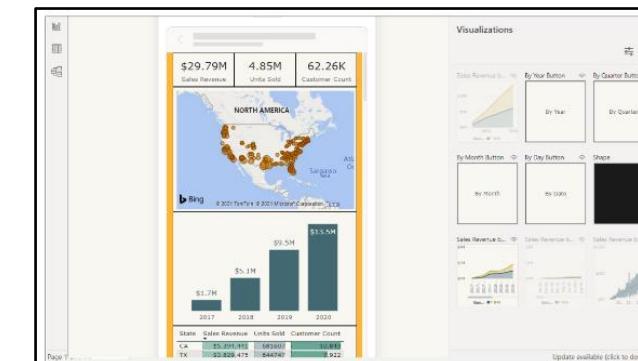
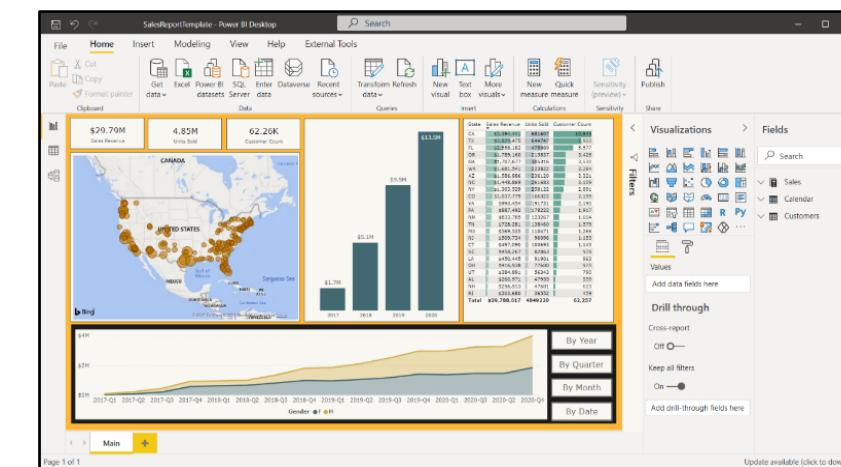
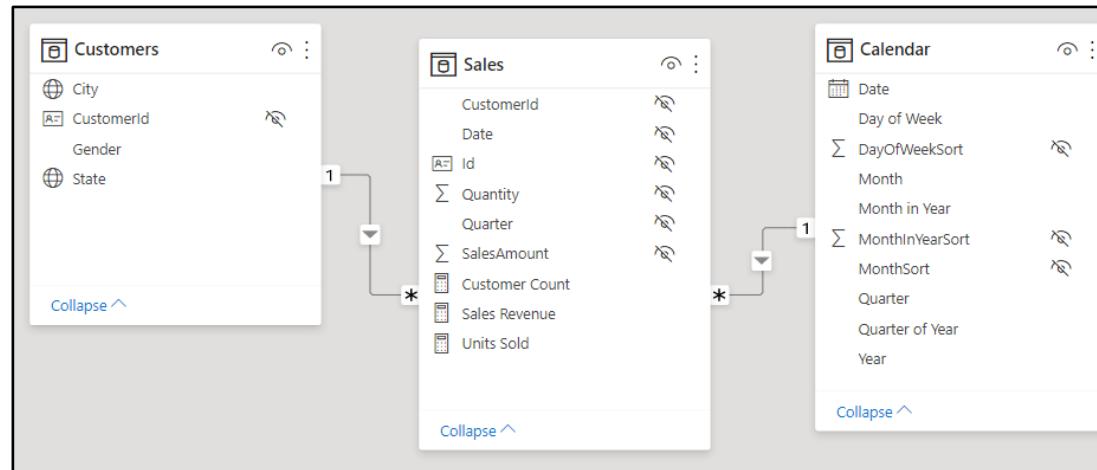
Understanding SalesReportTemplate.pbix

- **AppOwnsDataAdmin** uses **SalesReportTemplate.pbix** as PBIX template file
- PBIX import process creates **Sales** dataset and **Sales** report
- **Sales** dataset parameterized with **DatabaseServer** and **DatabaseName**
- Dataset parameters updated after import to redirect **Sales** dataset to customer database



Dataset and Report Design

- **SalesReportTemplate.pbix** data model designed with three tables
- **SalesReportTemplate.pbix** report designed with master view & mobile view



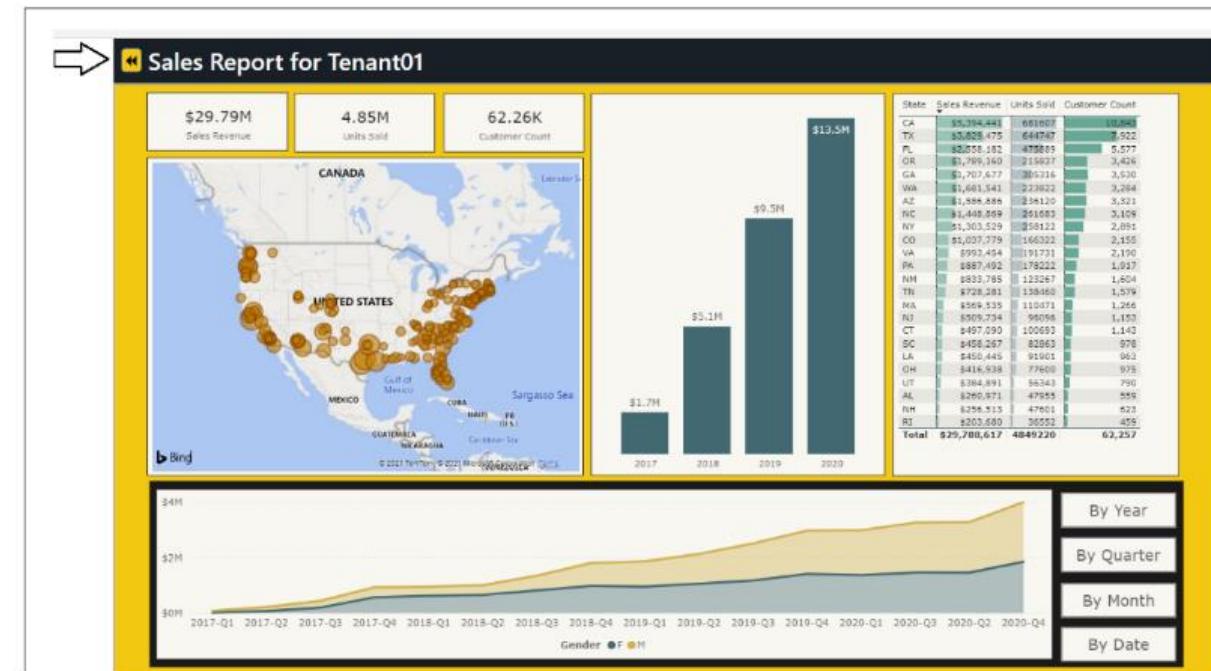
Embedding Reports in AppOwnsDataAdmin

- **AppOwnsDataAdmin** provides **Embed** view for each tenant
 - Allows administrative user to view the **Sales** report from any tenant

Power BI Tenants

[+ Onboard New Tenant]

Tenant	Workspace ID	Embed	Web URL	View	Delete
Tenant01	775ada0a-b353-45e3-a070-8277a557ea17				
Tenant02	b35b6342-6563-46c1-b3e0-0592926e976b				



What's been created in Power BI?

- You can see the workspaces created in Power BI
 - There should be a Power BI workspace for each tenant that's been created
 - **ConfiguredBy** property of **Sales** dataset should be set to **App-Owns-Data Service App**
 - **Last refresh** time should reflect refresh operation at end of onboarding process

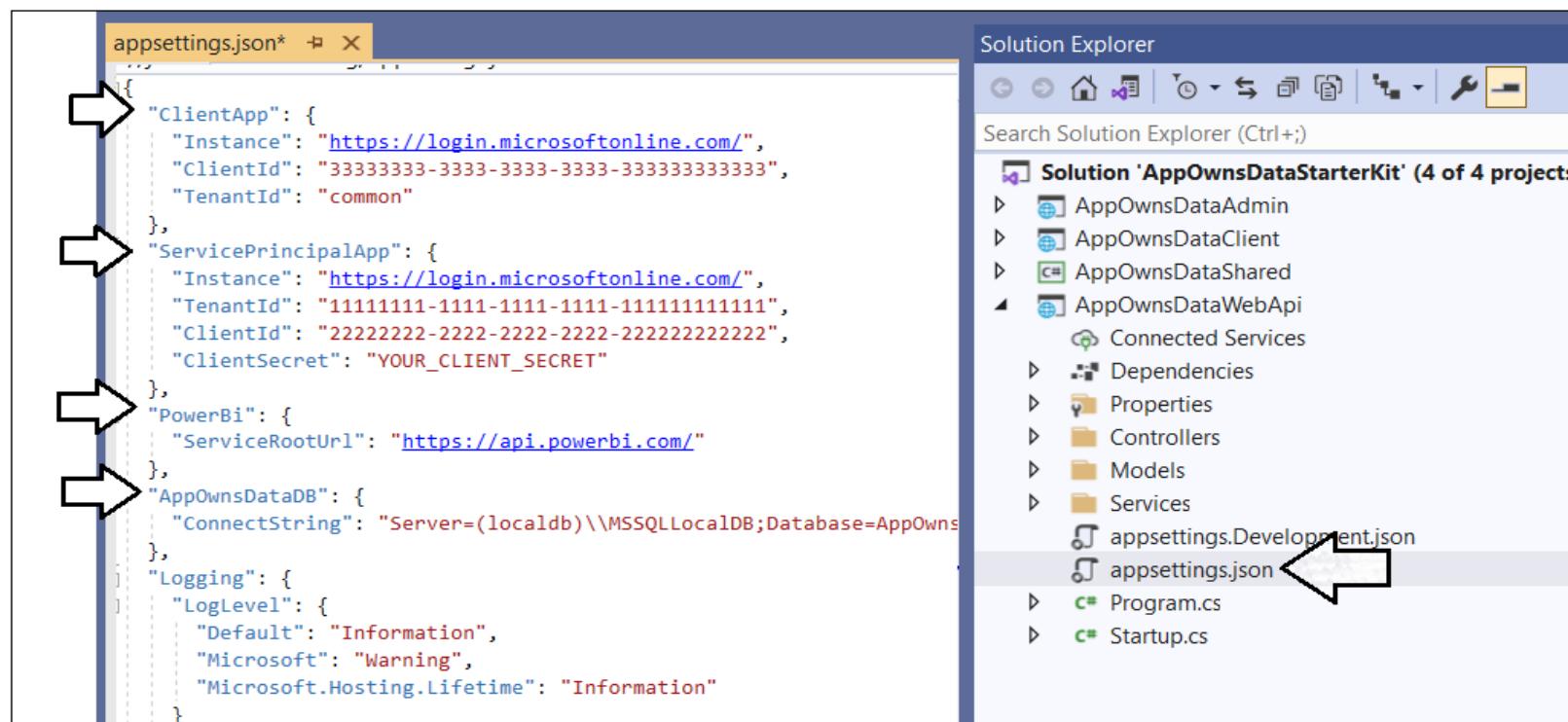
The image consists of two screenshots of the Power BI interface. The left screenshot shows the main 'Home' page with a sidebar containing 'Workspaces' (highlighted with a yellow arrow), 'Tenant01', and 'Tenant02'. The right screenshot shows the 'Datasets' tab for the 'Sales' dataset in the 'Tenant01' workspace. The 'ConfiguredBy' property is highlighted with a red box and an arrow pointing to it. Other visible elements include 'General', 'Alerts', 'Subscriptions', 'Dashboards', 'Workbooks', 'Dataflows', and 'App' tabs. A message 'Settings for Sales' and 'This dataset has been configured by App-Owns-Data Service App (App ID: 22b111f4-5046-4d48-b210-1862a633be95). What settings?' is also present.

Agenda

- ✓ Solution Architecture
- ✓ Set up your development environment
- ✓ Open the App-Owns-Data Starter Kit in Visual Studio
- ✓ Test the AppOwnsDataAdmin application
- Test the AppOwnsDataClient application
- Use activity logs to monitor user activity and report performance

Update appsettings.json in AppOwnsDataWebApi

- Update **ClientApp** section with data from **App-Owns-Data Client App**
- Update **ServicePrincipalApp** section with data from **App-Owns-Data Service App**
- Leave **PowerBi** section with default setting when using Power BI public cloud
- Ensure **AppOwnsDataDB:ConnectionString** is valid for your development environment



Configure and build AppOwnsDataClient

- AppOwnsDataClient needs to be configured with Client ID
 - Use **Client ID** from **App-Owns-Data Client App**
 - Compile Typescript in project after updating **Client ID**
 - **Node.js** and **webpack** used to compile Typescript for distribution

The screenshot shows the Visual Studio IDE interface. On the left is the code editor window titled "appSettings.ts*" containing the following TypeScript code:

```
export default class AppSettings {
    public static clientId: string = "33333333-3333-3333-3333-333333333333";
    public static tenant: string = "common";
    public static apiRoot: string = "https://localhost:44302/api/";
    public static apiScopes: string[] = [
        "api://" + AppSettings.clientId + "/Reports.Embed"
    ];
}
```

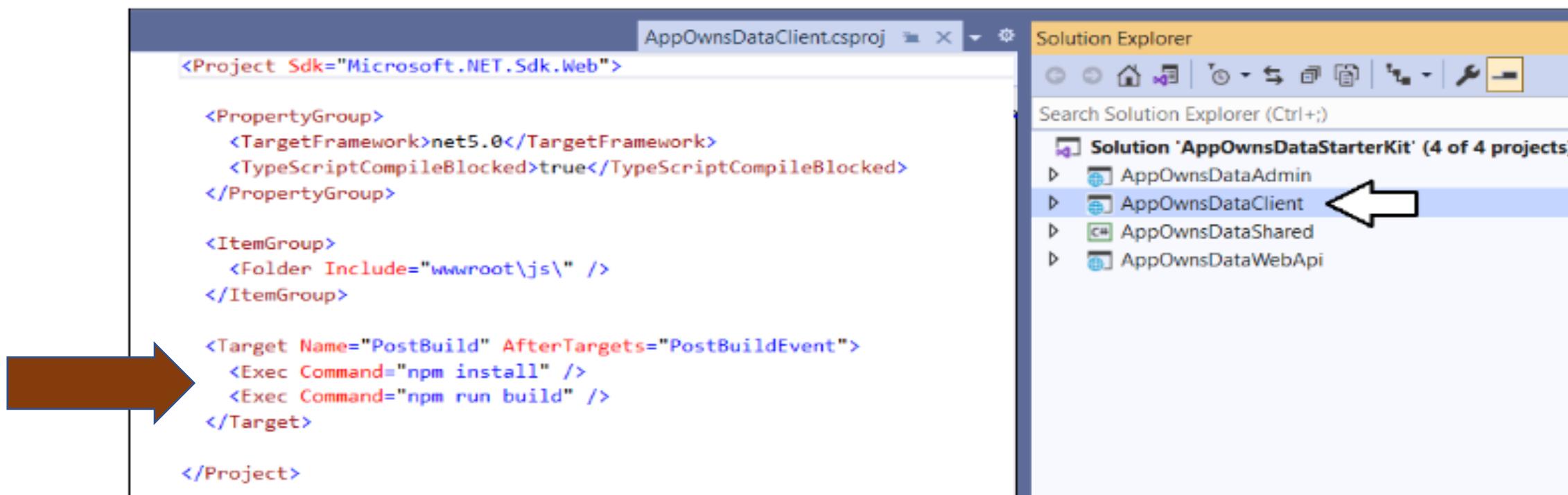
On the right is the "Solution Explorer" window, which lists the projects in the solution:

- Solution 'AppOwnsDataStarterKit' (4 of 4 projects)
 - AppOwnsDataAdmin
 - AppOwnsDataClient
 - Connected Services
 - Dependencies
 - Properties
 - wwwroot
 - App
 - models
 - services
 - app.ts
 - appSettings.ts
 - Controllers
 - Models

A black arrow points from the "appSettings.ts" file in the Solution Explorer towards the code editor.

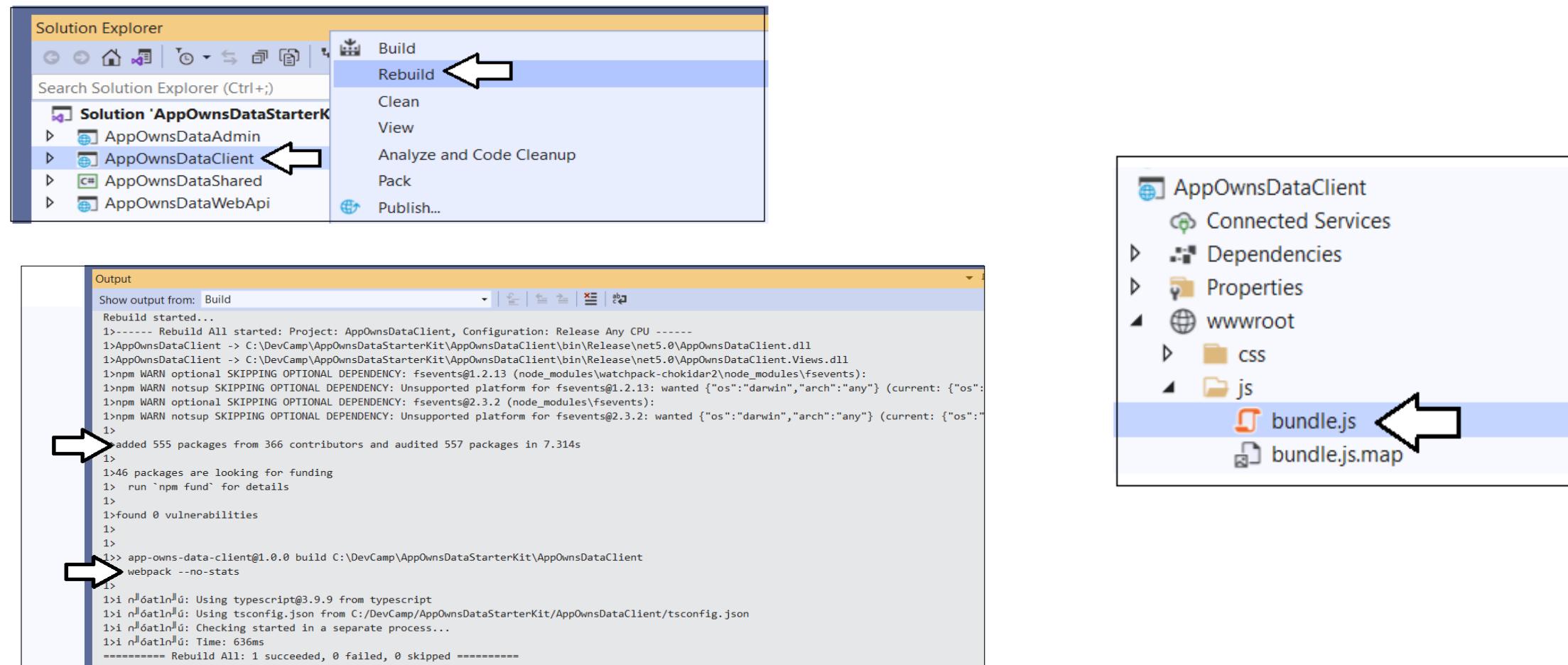
Node.js integration

- **AppOwnsDataClient** project built as a Node.js project
 - You must install **Node.js** in order to build the project
 - **npm install** command used to restore Node.js packages
 - **npm run build** command triggers webpack to compile Typescript into JavaScript



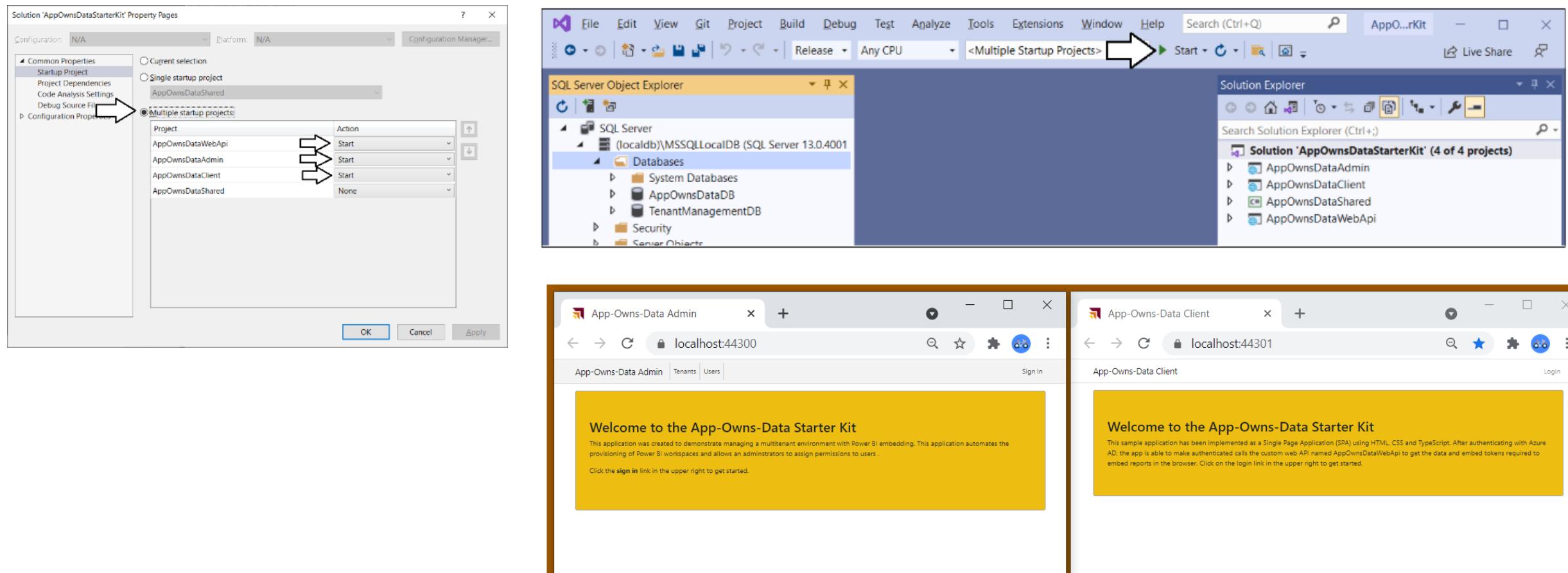
Building the AppOwnsDataClient Project

- Rebuilding project executed required Node.js commands
 - **webpack** utility used to compile Typescript into **bundle.js** file for distribution



Test AppOwnsDataClient in Visual Studio

- Configure the **AppOwnsDataStarterKit** solution with multiple Startup projects
- Allows testing **AppOwnsDataClient** along side by side with **AppOwnsDataAdmin**



Testing the App-Owns-Data Applications Side by Side

- Use **AppOwnsDataAdmin** to move **AppOwnsDataClient** user through 4 possible states
 1. Test **AppOwnsDataClient** when user is **unassigned**
 2. Test **AppOwnsDataClient** when user is assigned to tenant with **read-only permissions**
 3. Test **AppOwnsDataClient** when user is assigned to tenant with **edit permissions**
 4. Test **AppOwnsDataClient** when user is assigned to tenant with **edit + create permissions**

Edit User

Login Id:	AustinP@powerbidevcamp.net
Last Login:	4/19/2021 6:36:11 PM
User Name:	Austin Powers
Home Tenant:	[unassigned]
Can Edit:	<input type="checkbox"/>
Can Create:	<input type="checkbox"/>

→ Home Tenant: [unassigned] [unassigned] Tenant01 Tenant02

Edit User

Login Id:	AustinP@powerbidevcamp.net
Last Login:	4/19/2021 6:36:11 PM
User Name:	Austin Powers
Home Tenant:	Tenant01
Can Edit:	<input type="checkbox"/>
Can Create:	<input type="checkbox"/>

→ Save

Users

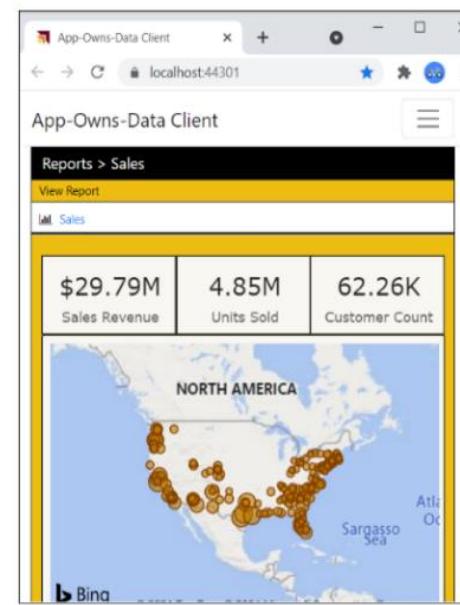
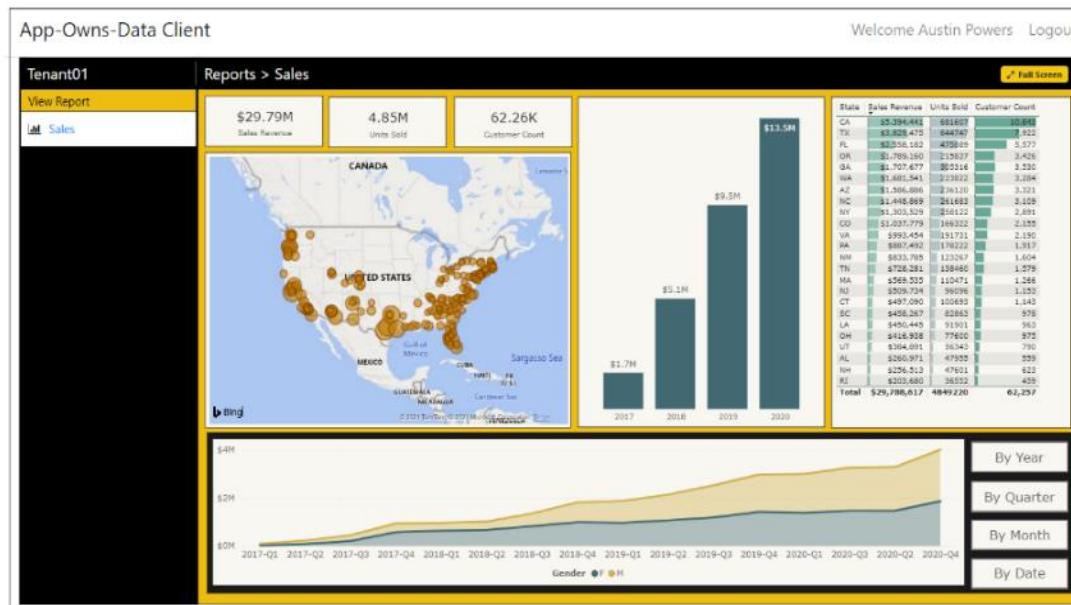
Users							
Create New User							
Login ID	User Name	Tenant	Can Edit	Can Create	View	Edit	Delete
AustinP@powerbidevcamp.net	Austin Powers	Tenant01	False	False			

AppOwnsDataClient User Experience

- Test user experience when user is unassigned

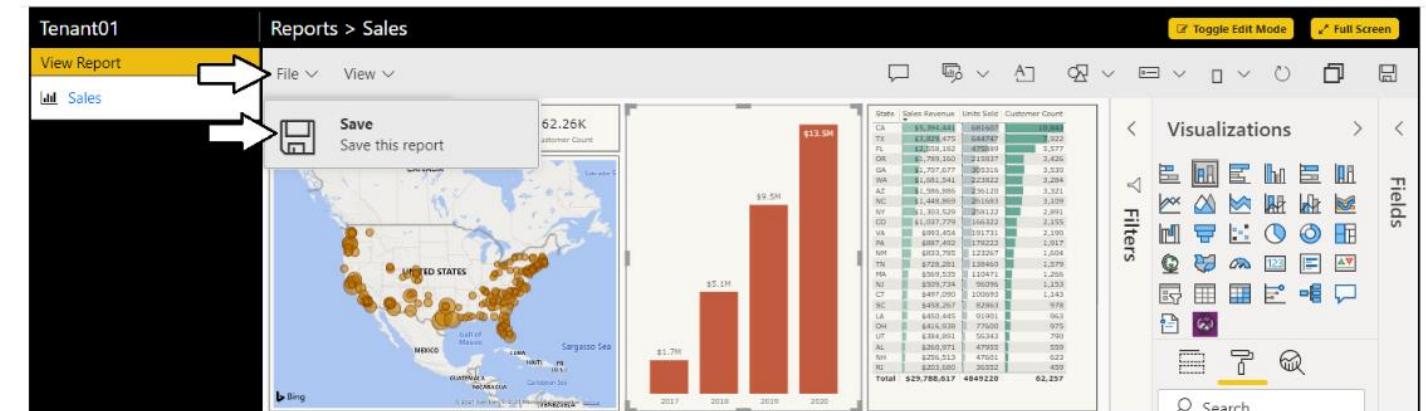
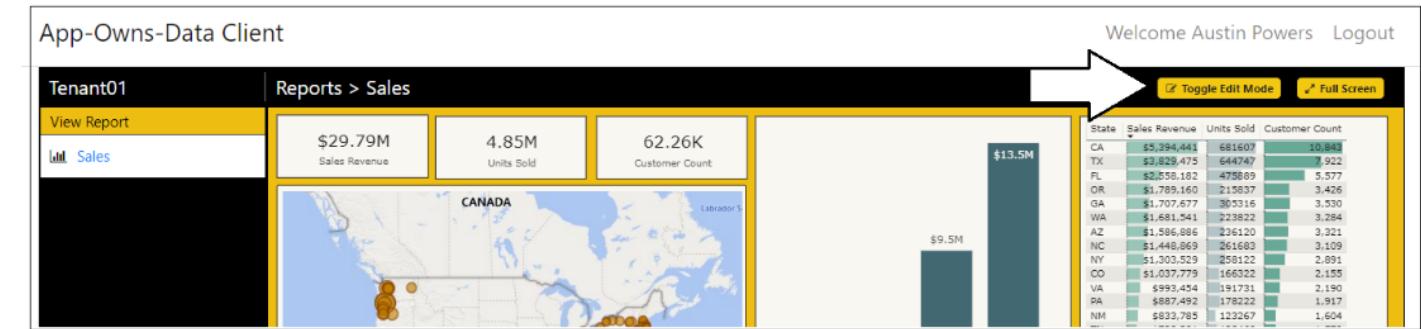
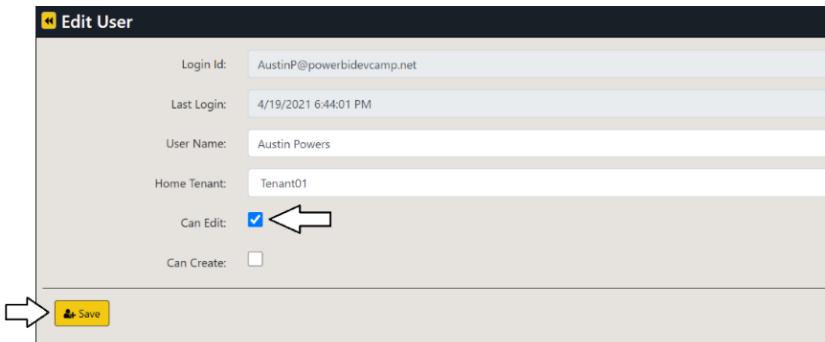
The screenshot shows a login page with a large orange message box in the center. The message reads: "Thanks for logging in. Your user account has no assigned tenant. Once your user account has been assigned to a tenant, you will be able to use this application and interact with embedded report from Power BI."

- Test experience with user assigned - make browser window narrow to see mobile view



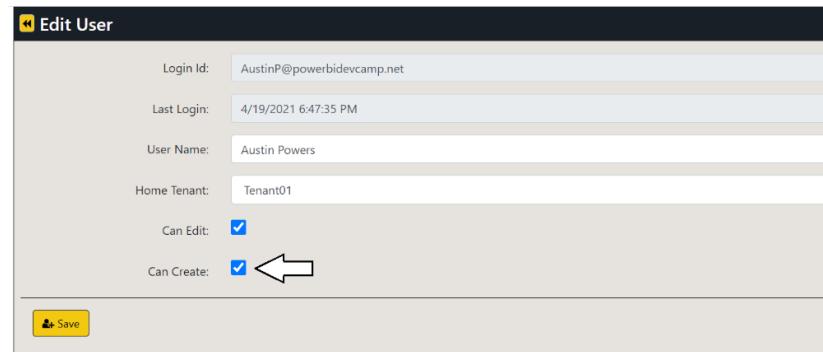
Edit reports using AppOwnsDataClient

- Test user experience when user has edit permissions
 - User can move report into edit mode
 - Once in edit mode, you can customize report and save changes

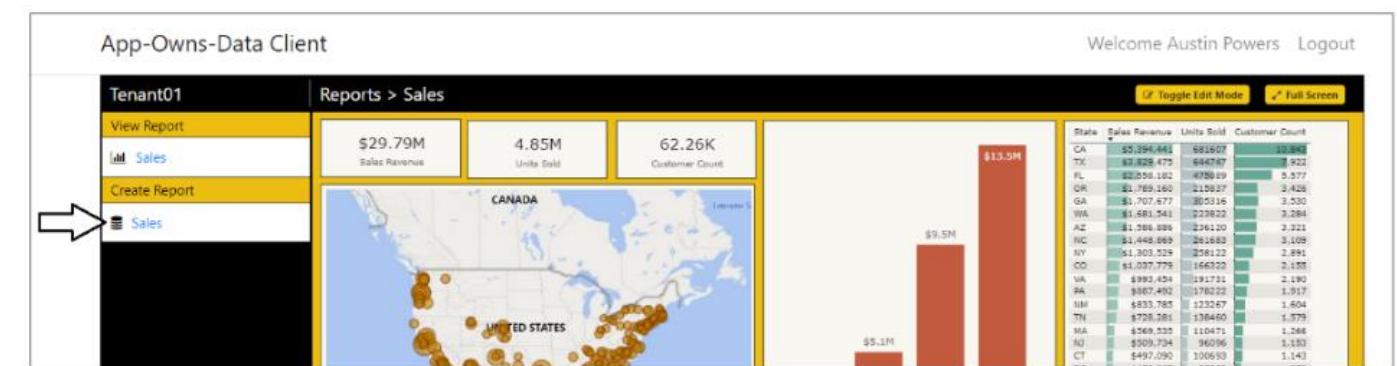


Create New Content using AppOwnsDataClient

- Test user experience when user has **edit + create permissions**
 - User can use **Save As** command on report in edit mode to copy a report
 - User can click dataset link in **Create Report** section to create new report



The screenshot shows the 'App-Owns-Data Client' interface. On the left, the 'Tenant01' sidebar has a 'Create Report' option highlighted with a yellow arrow. The main area shows a 'Sales' dataset with a 'Save as' dialog open, and a second yellow arrow points to the 'Create Report' button in the dialog.



Agenda

- ✓ Solution Architecture
- ✓ Set up your development environment
- ✓ Open the App-Owns-Data Starter Kit in Visual Studio
- ✓ Test the AppOwnsDataAdmin application
- ✓ Test the AppOwnsDataClient application
- Use activity logs to monitor user activity and report performance

Monitor user activity data from ActivityLog table

- **AppOwnsDataClient** constantly logs user activity
 - Activity is logged whenever a user views a report
 - Activity is logged whenever a user edits, copies or creates a report
 - Activity is logged by **AppOwnsDataWebApi** by adding record into **ActivityLog** table

The screenshot shows a SQL Server Management Studio (SSMS) interface. At the top, there is a query window containing the following T-SQL code:

```
SELECT [LoginId],[Activity],[Tenant],[WorkspaceId],[Dataset],
       [Report],[LoadDuration],[RenderDuration],[Created]
  FROM ActivityLog
 ORDER BY [Created] desc
```

Below the query window, there are two tabs: "Results" and "Messages". The "Results" tab is selected and displays a table of activity log data. The table has the following columns:

	LoginId	Activity	Tenant	WorkspaceId	Dataset	Report	LoadDuration	RenderDuration	Created
1	AustinP@powerbidevcamp.net	ViewReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales	559	1256	2021-04-19 19:09:52.1372042
2	AustinP@powerbidevcamp.net	ViewReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales by Year and Quarter	627	1233	2021-04-19 19:09:48.1554560
3	AustinP@powerbidevcamp.net	ViewReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales	611	1330	2021-04-19 19:09:30.1482556
4	AustinP@powerbidevcamp.net	ViewReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales by Year and Quarter	1457	2137	2021-04-19 19:07:17.7397512
5	AustinP@powerbidevcamp.net	CreateReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales by Year and Quarter	NULL	NULL	2021-04-19 19:07:14.9479799
6	AustinP@powerbidevcamp.net	ViewReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales	1023	2093	2021-04-19 18:55:00.1997032
7	AustinP@powerbidevcamp.net	EditReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales	NULL	NULL	2021-04-19 18:54:54.7320409
8	AustinP@powerbidevcamp.net	ViewReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales	1172	2065	2021-04-19 18:47:38.8572691
9	AustinP@powerbidevcamp.net	ViewReport	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	Sales	Sales	1309	2318	2021-04-19 18:44:06.6375980

AppOwsDataUsageReporting.pbix

- App-Owns-Data Starter Kit includes report template **AppOwsDataUsageReporting.pbix**
 - **AppOwsDataUsageReporting.pbix** imports data from **AppOwnsDataDB**
 - Allows for monitoring and analysis of usage data

The screenshot shows the Power BI Desktop interface with the title bar "AppOwsDataUsageReporting - Power BI Desktop". The ribbon is visible with the "Home" tab selected. The main area displays a table visualization titled "Activity Log" with the following columns: Created, Tenant, WorkspaceId, LoginId, Activity, Report, Dataset, LoadDuration, and RenderDuration. The table contains 12 rows of data. The "Fields" pane on the right lists "UserName" and "Activity Log". The "Visualizations" pane on the far right shows a single chart. The bottom navigation bar includes tabs for "Activity Log" (which is selected), "Report Authoring", "Slow Reports", "Users", and a plus sign for new content.

Created	Tenant	WorkspaceId	LoginId	Activity	Report	Dataset	LoadDuration	RenderDuration
2021-04-19 6:44:06 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales	Sales	1309	2318
2021-04-19 6:47:38 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales	Sales	1172	2065
2021-04-19 6:54:54 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	EditReport	Sales	Sales		
2021-04-19 6:55:00 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales	Sales	1023	2093
2021-04-19 7:07:14 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	CreateReport	Sales by Year and Quarter	Sales		
2021-04-19 7:07:17 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales by Year and Quarter	Sales	1457	2137
2021-04-19 7:09:30 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales	Sales	611	1330
2021-04-19 7:09:48 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales by Year and Quarter	Sales	627	1233
2021-04-19 7:09:52 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales	Sales	559	1256
2021-04-20 8:57:11 AM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales	Sales	3084	4557
2021-04-20 8:57:33 AM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales by Year and Quarter	Sales	1933	3333

Capturing Report Performance Data

- Steps to capture performance data when embedding a report
 - Capture time before starting the embedding process
 - Determine time duration between start of embedding and the **Loaded** event
 - Determine time duration between start of embedding and the **rendered** event
 - **LoadDuration** and **RenderDuration** recorded in milliseconds

```
var timerStart: number = Date.now();
var initialLoadComplete: boolean = false;
var loadDuration: number;
var renderDuration: number;

App.currentReport = <powerbi.Report>App.powerbi.embed(App.embedContainer[0], config);

App.currentReport.on("loaded", async (event: any) => {
    loadDuration = Date.now() - timerStart;
    // other code for loaded omitted for brevity
});

App.currentReport.on("rendered", async (event: any) => {
    if (!initialLoadComplete) {
        renderDuration = Date.now() - timerStart;
        var correlationId: string = await App.currentReport.getCorrelationId();
        await App.logViewReportActivity(correlationId, App.viewModel.embedTokenId, report, loadDuration, renderDuration);
        initialLoadComplete = true;
    }
});
```

Activity Event Payload	
JSON	Activity=ViewReport
	CorrelationId=d40b3d7f-ce2f-4f83-9f8d-0d57efcb2b50
	Dataset=Sales
	DatasetId=85e47446-0abb-4d6f-a6cb-fce430a50ad5
	EmbedTokenId=66ad4d0c-42a2-4f8c-a6f2-8c6e06517d6
	LoadDuration=2361
	LoginId=AustinP@powerbidevcamp.net
	RenderDuration=4043
	Report=Sales
	ReportId=8dc87d5c-e183-4a20-9083-9d7c38e3fc6e
	Tenant=Wingtip

Monitor Report Performance

- Monitor report performance across multi-tenant environment
 - Long load times could be evidence of report with slow connection
 - Long render times could be evidence of report with too many visuals

Tenant	Dataset	Report	Report Views	Avg Load Time	Avg Render Time
Customer 123	Sales	Sales	18	1.40	2.69
Wingtip	Sales	Sales	317	1.56	2.63
Ofer INC	Sales	Sales	8	1.42	2.56
Mega Corp	Sales	Sales	11	1.29	2.14
Acme Corp	Sales	Sales	74	1.06	1.78
Contoso	Sales	Sales	1	0.82	1.68
Wingtip	Sales	Report 2	1	0.90	1.64
Wingtip	Sales	Sales in FL	8	0.73	1.58
Acme Corp	Sales	Sales in FL	5	0.85	1.52
Customer 123	Sales	Report	3	1.07	1.36

Next Steps

- Use a different authentication provider
- Create a more granular permissions scheme
- Add integration with row-level security (RLS)
- Redesign AppOwnsDataClient using React.js or Angular
- Learn scaling techniques to support more than 1000 tenants

Summary

- ✓ Solution Architecture
- ✓ Set up your development environment
- ✓ Open the App-Owns-Data Starter Kit in Visual Studio
- ✓ Test the AppOwnsDataAdmin application
- ✓ Test the AppOwnsDataClient application
- ✓ Use activity logs to monitor user activity and report performance