Microsoft | Power BI

PBICAT

App-Owns-Data Starter Kit

Ted Pattison

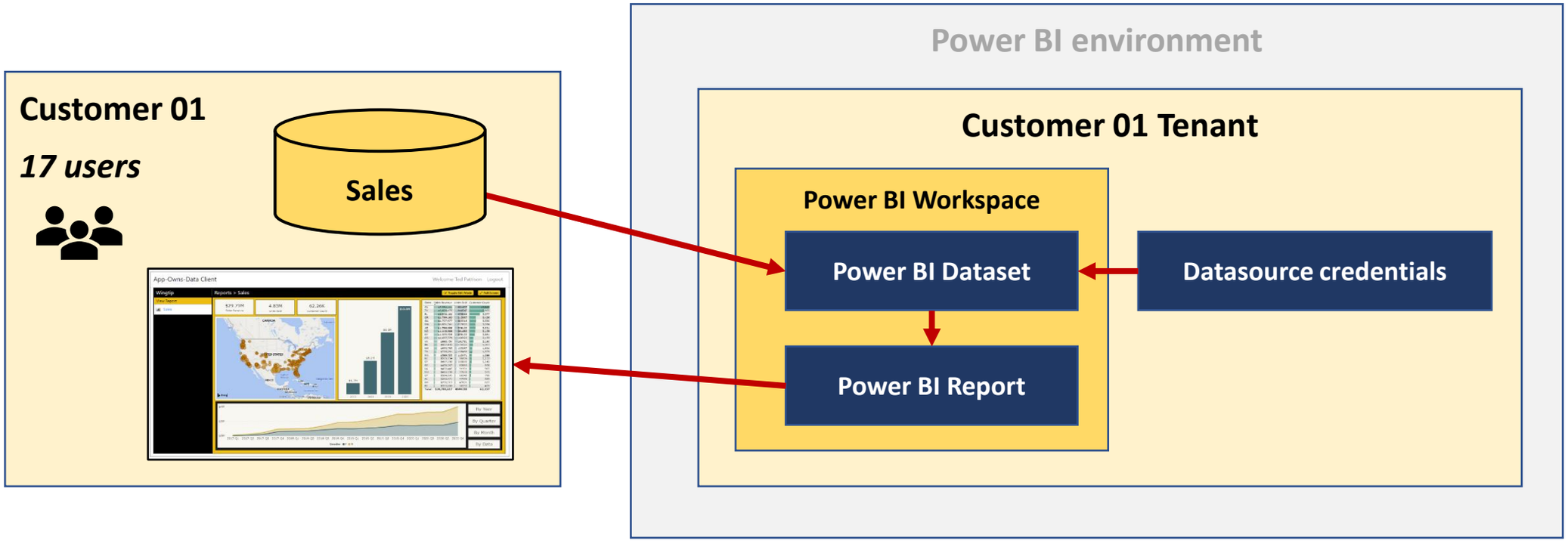
Power BI Customer Advisory Team (PBICAT)

Agenda

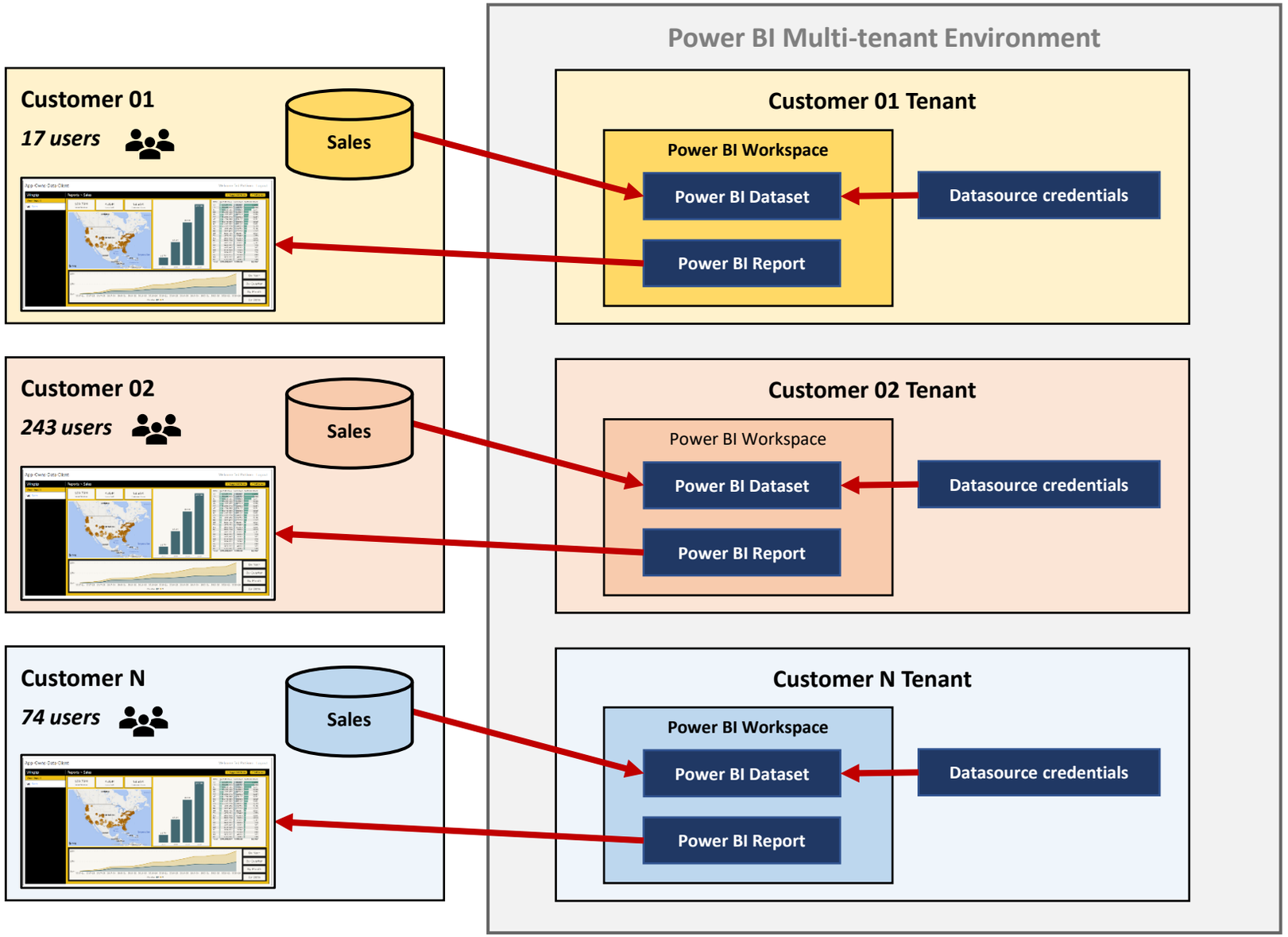
- Solution Architecture
- Set up your development environment
- Open the App-Owns-Data Starter Kit in Visual Studio
- Test the AppOwnsDataAdmin application
- Test the AppOwnsDataClient application
- Use activity logs to monitor user activity and report performance

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

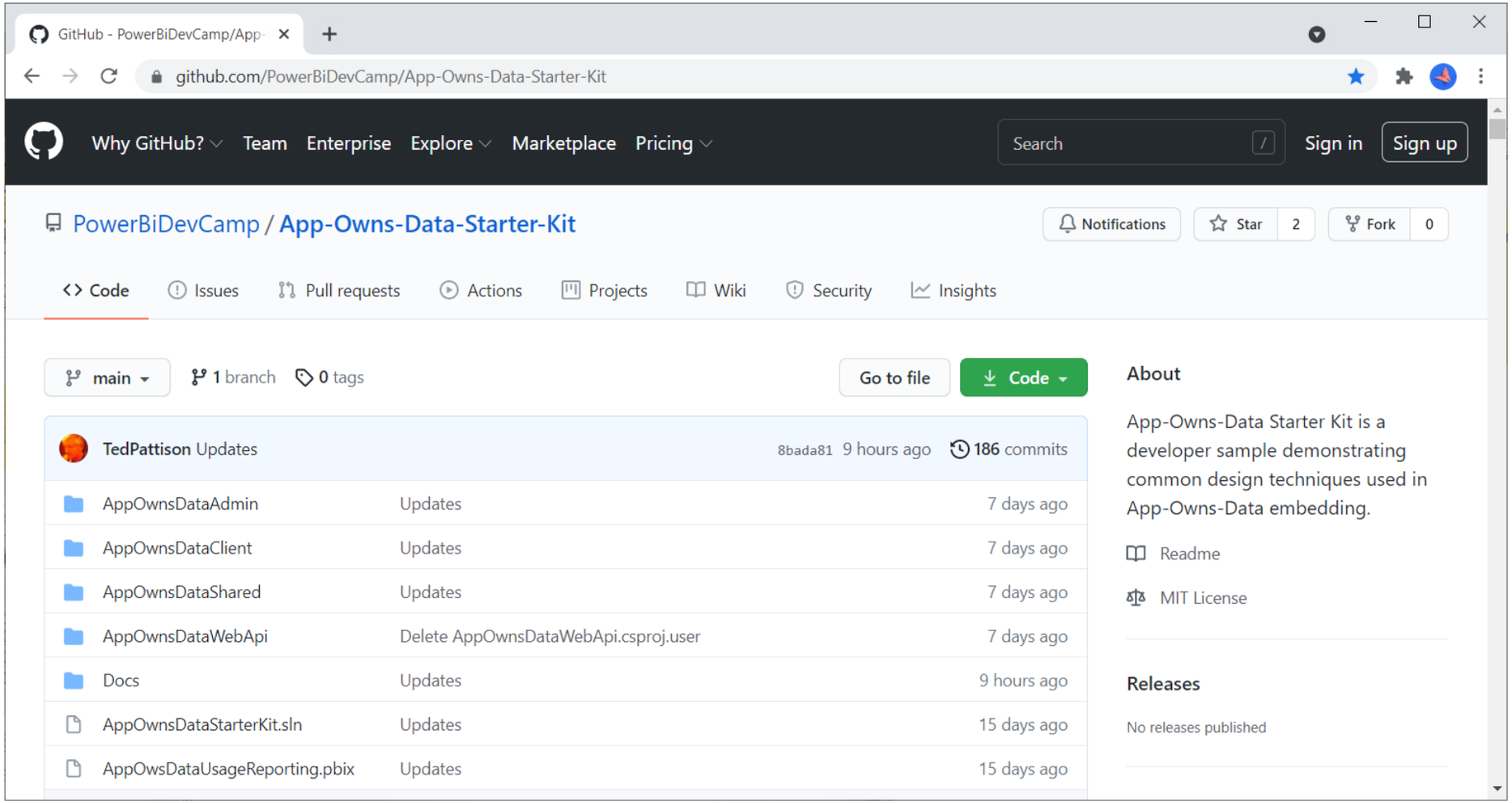
- Key observations about App-Owns-Data embedding
  - you have **flexibility** to design the authorization scheme any way you'd like
  - you have **responsibility** to build authorization scheme from the ground up
  - you should use **custom database** to track the authorization data you need

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

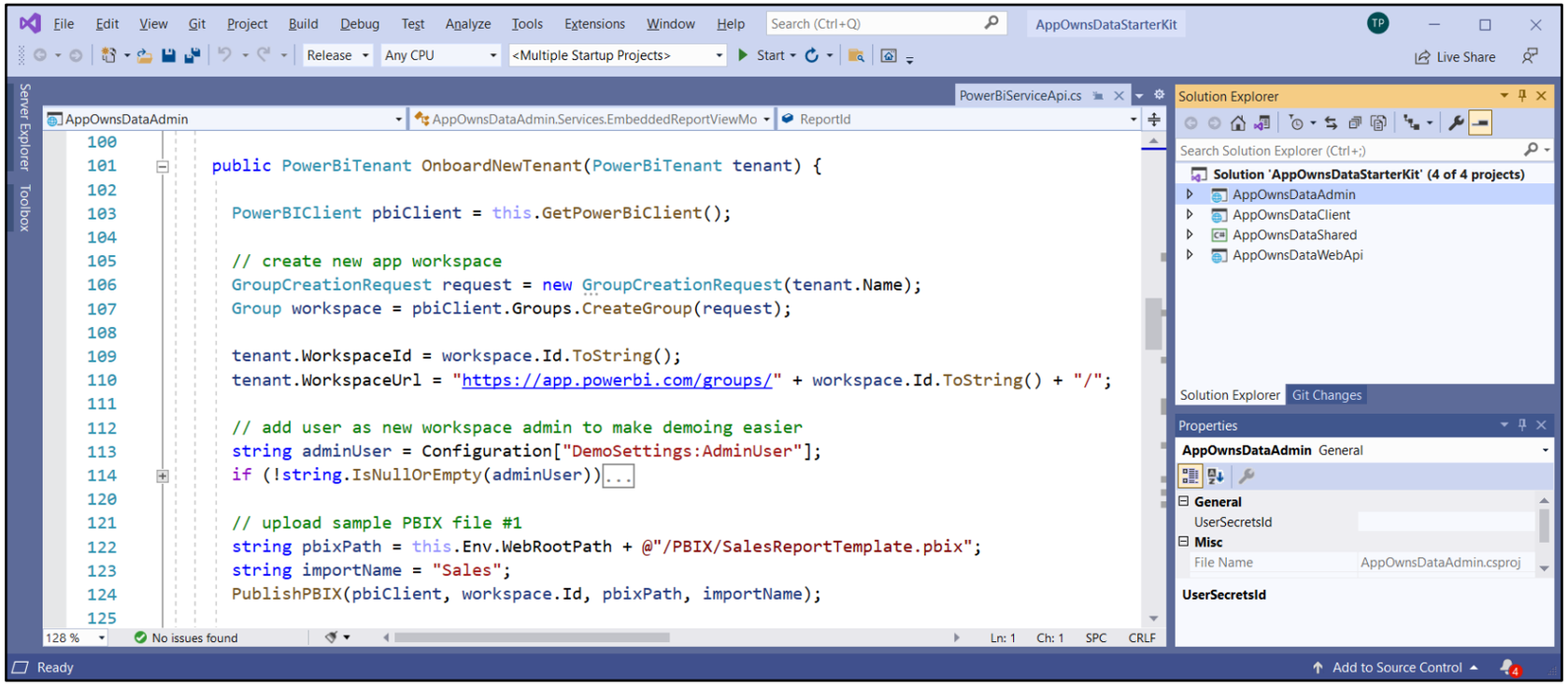
App-Owns-Data Starter Kit on GitHub

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



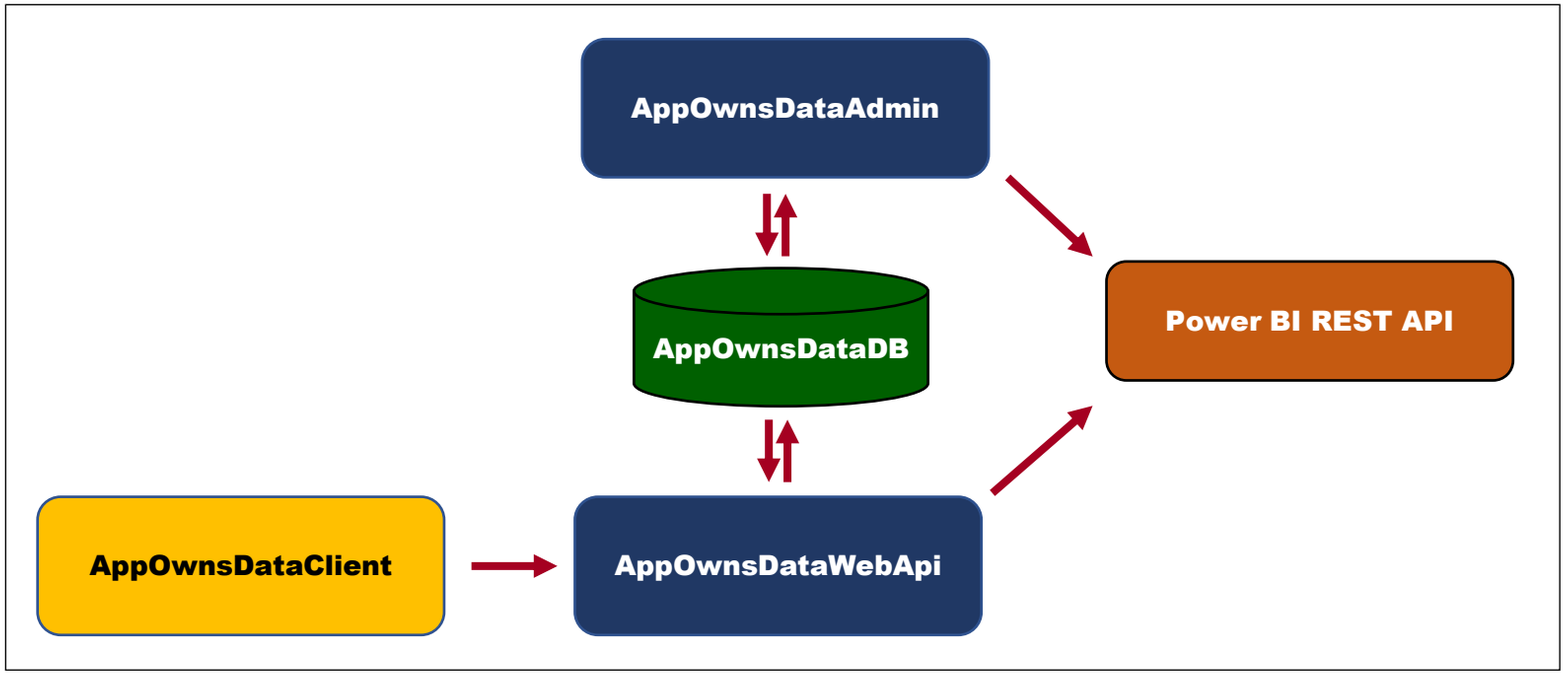
The App-Owns-Data Starter Kit Solution

- 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



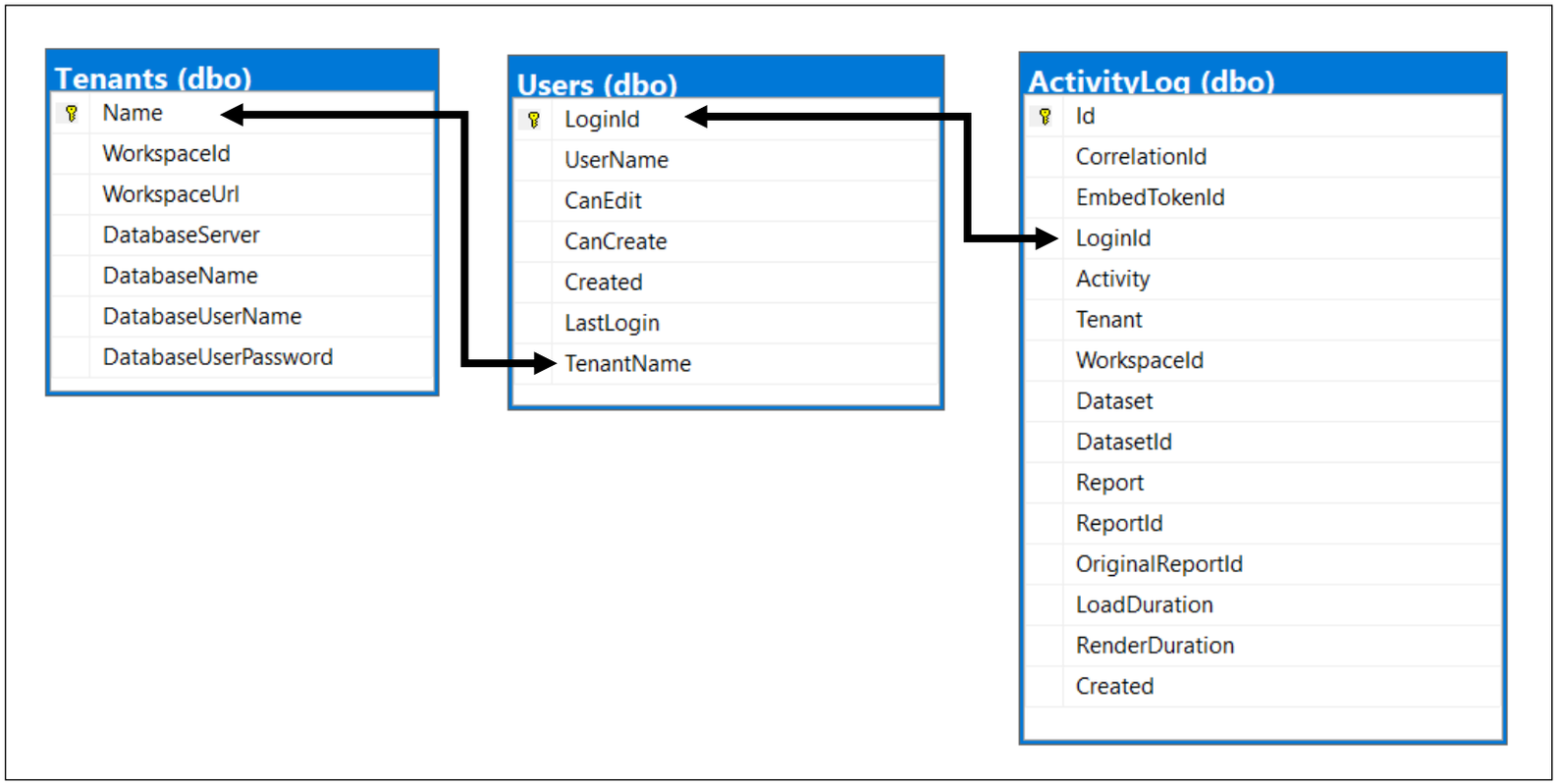
Solution Architecture

- **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



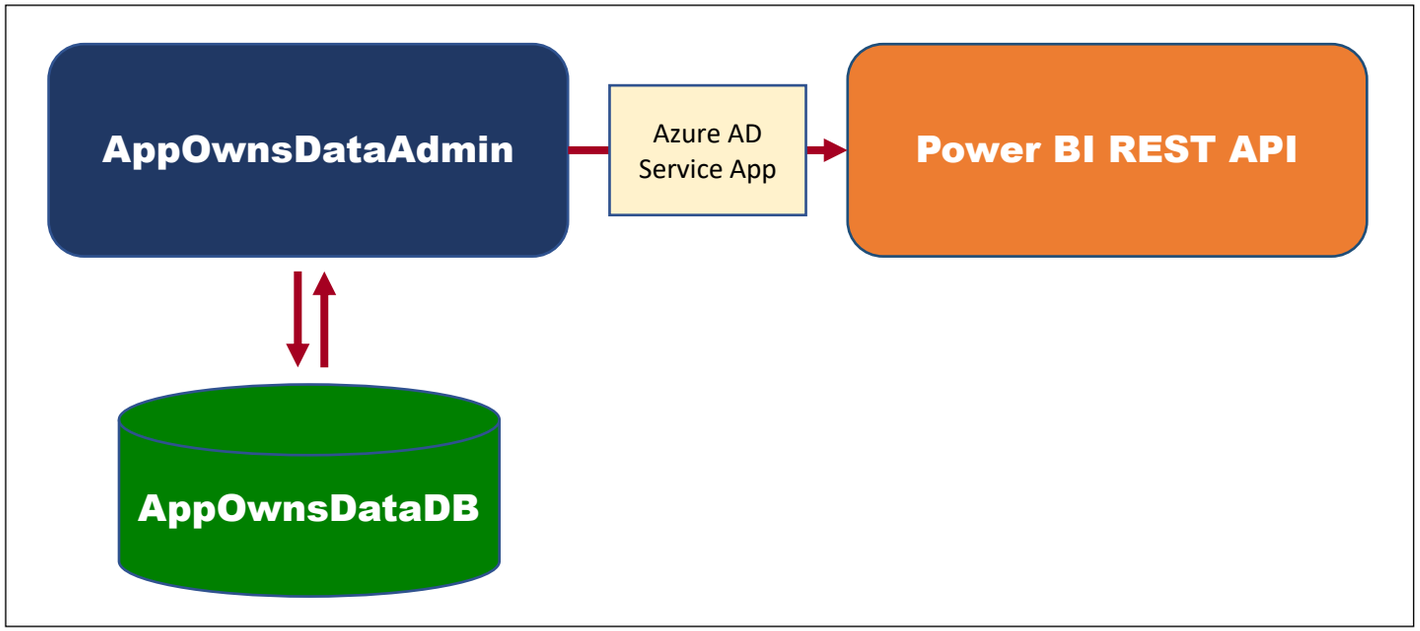
Database Schema for AppOwnsDataDB

- **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



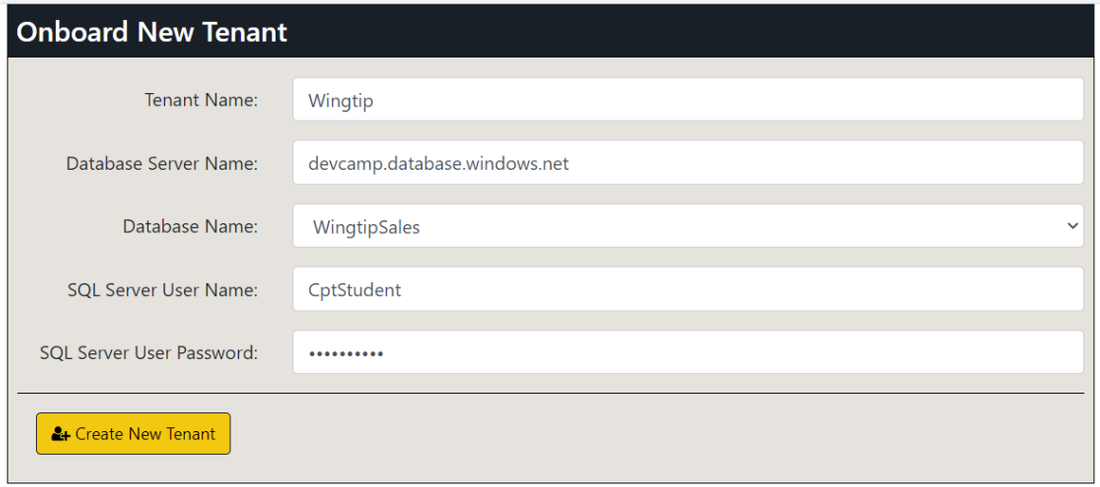
AppOwnsDataAdmin

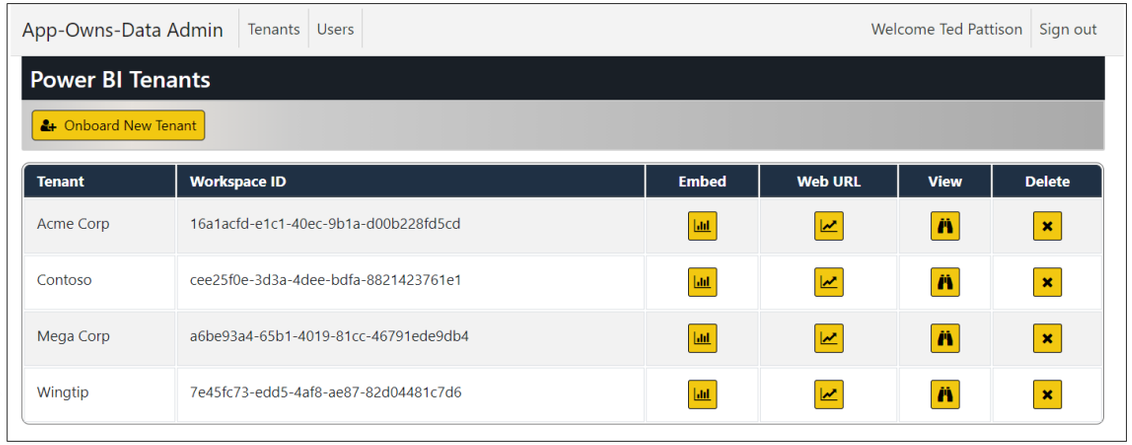
- Uses Power BI REST API to provision workspaces for customer tenants
- Calls to Power BI REST API as a service principal – not as a user
- Adds a record to AppOwnsDataDB for each customer tenant



Creating Customer Tenants

- AppOwnsDataAdmin provides **Onboard New Tenant** page
  - Makes is possible to configure details for connection to customer database
  - Tenants can be viewed or deleted



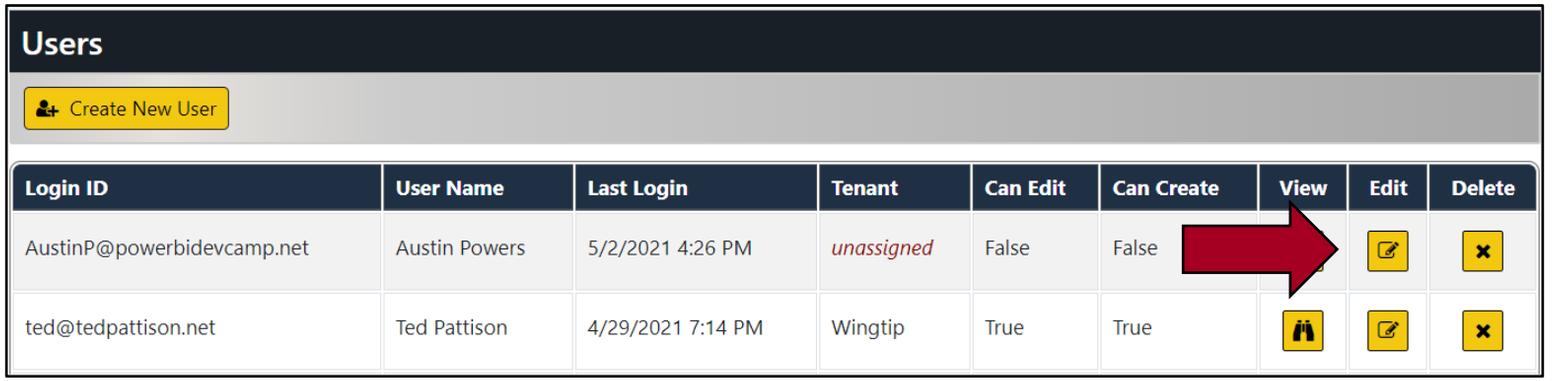


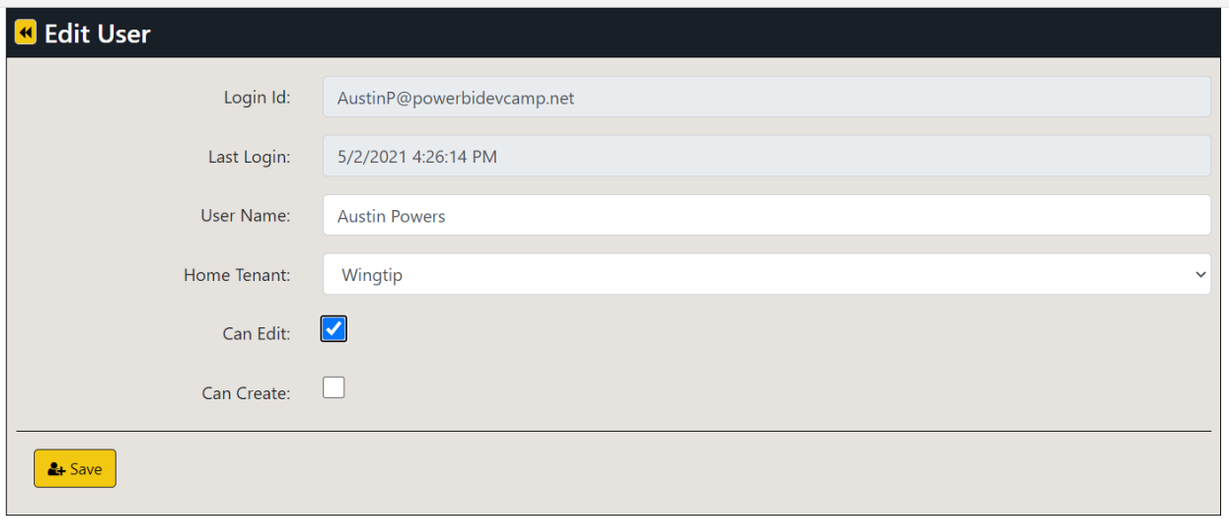
Sequence of Tenant Provisioning Operations

- 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

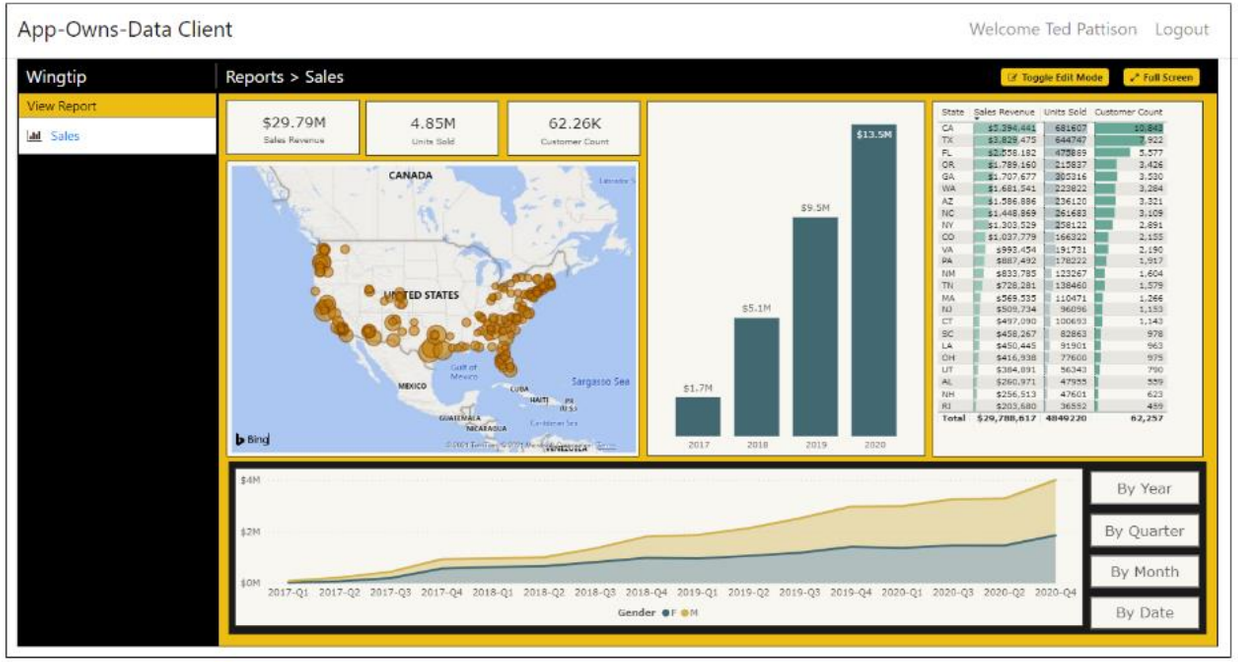
- **Edit User** page make it possible to assign user to a customer tenant
- Users can optionally be assigned permissions for Edit and Create

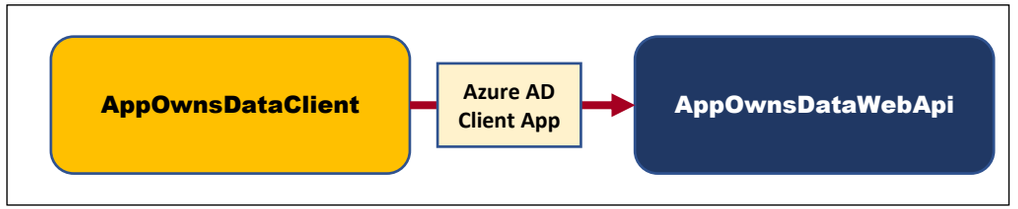




AppOwnsDataClient

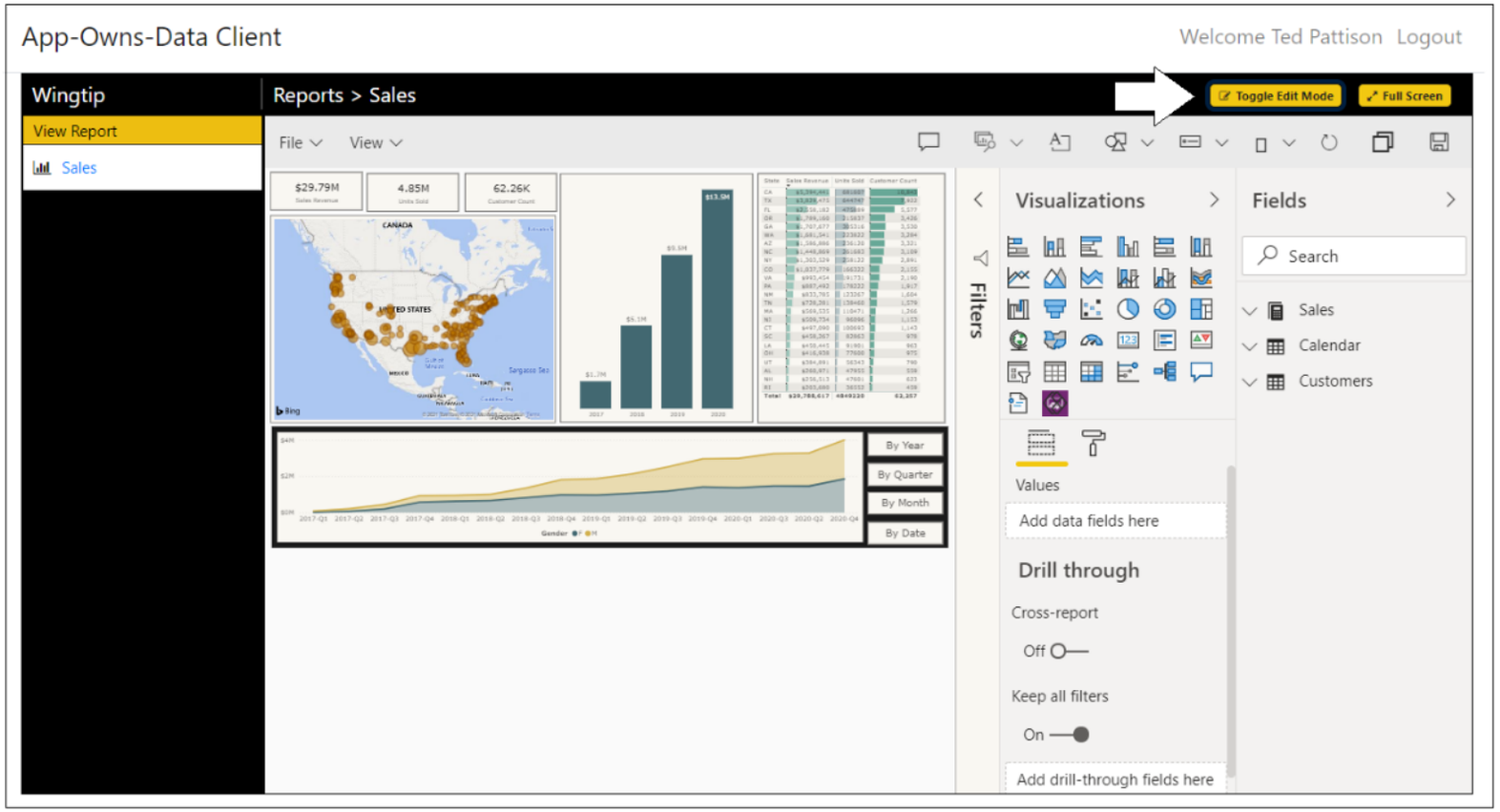
- Built as simple **Single Page Application (SPA)**
- Client **authenticates user** and acquires **access tokens** to call AppOwnsDataWebApi
- Project configured with **node.js** and **webpack** to support **Typescript** compilation
- Project includes node.js packages for **jQuery** and **Power BI JavaScript 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 create new reports

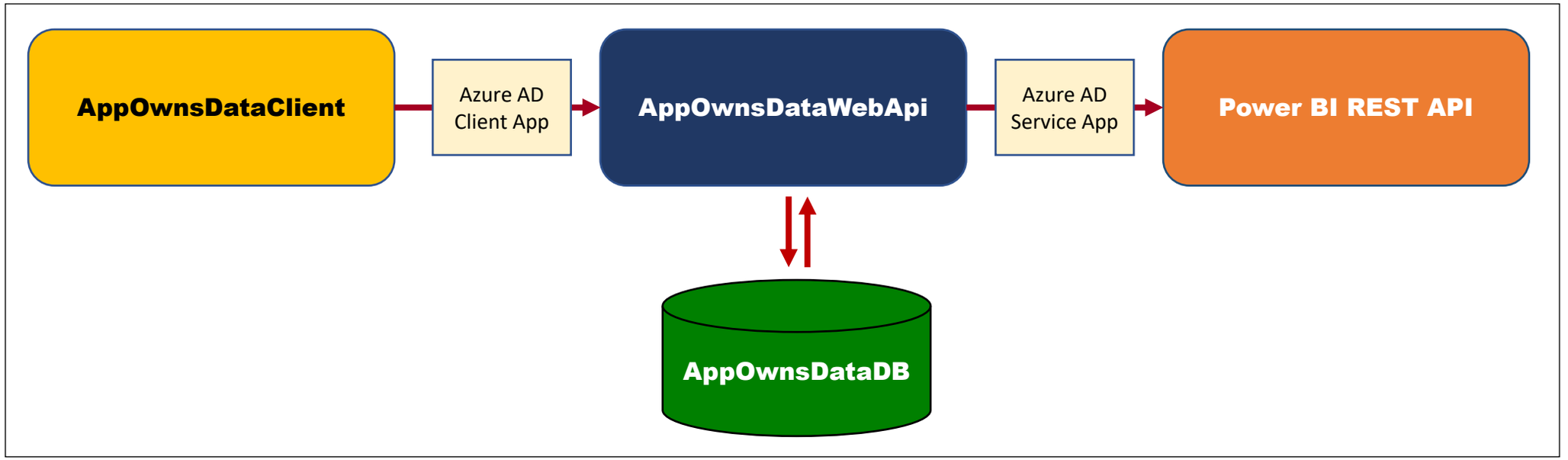


Responsive Design

- 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

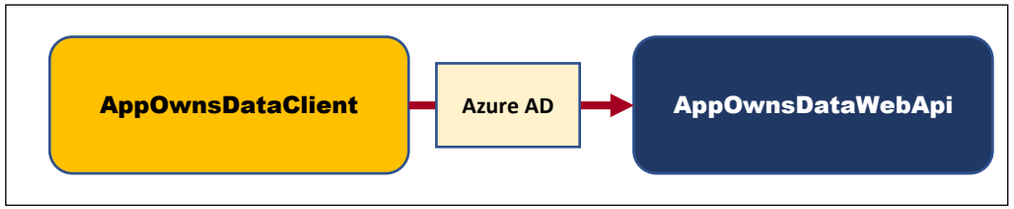
AppOwnsDataWebApi

- 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

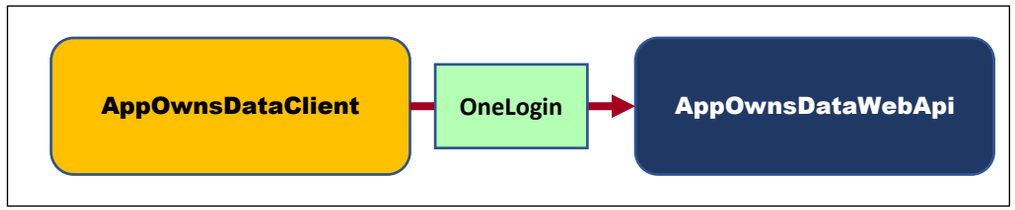


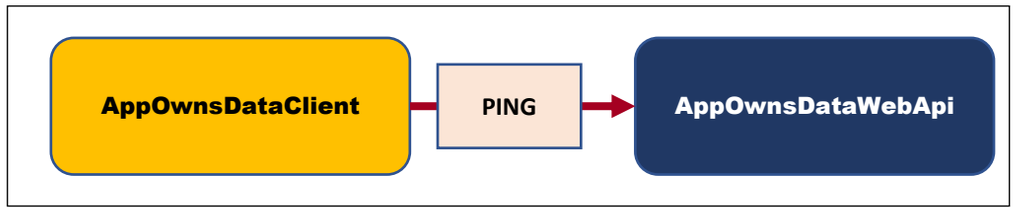
Identity Providers and App-Owns-Data Development

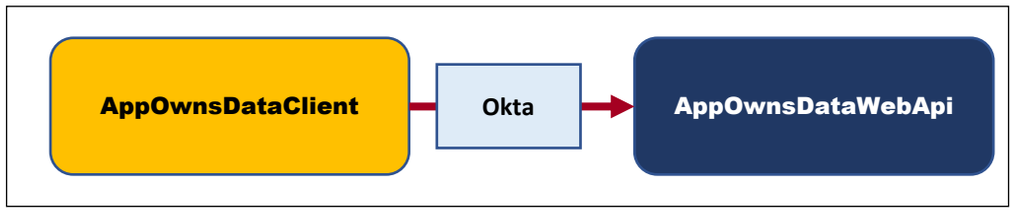
- App-Owns-Data development allows you to pick any identity provider
  - Identity provider responsible for authenticating user and validating LoginID
  - App-Owns-Data Starter Kit uses Azure AD as client app identity provider

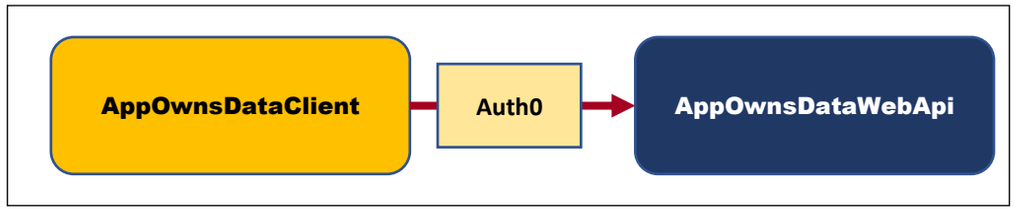


  - Solution architecture designed to facilitate switching out the identity provider



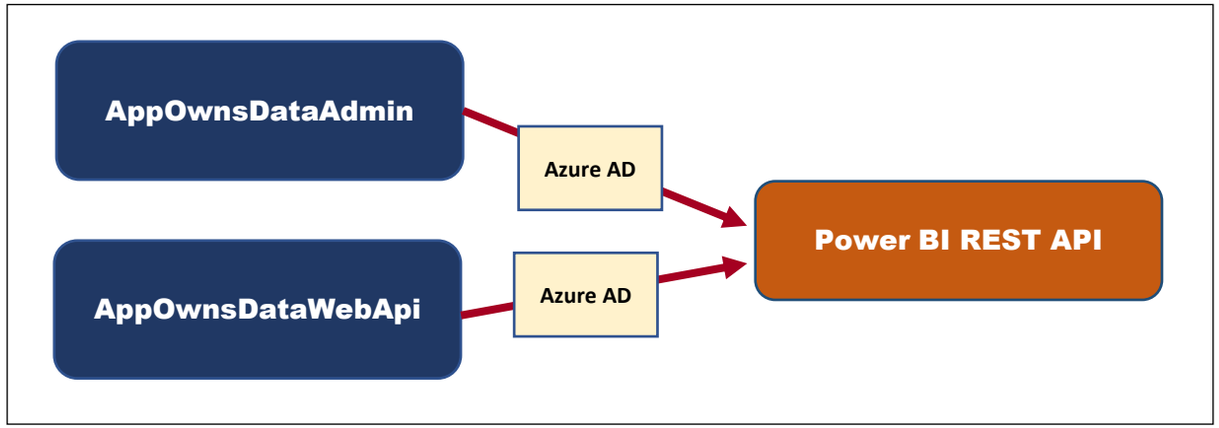






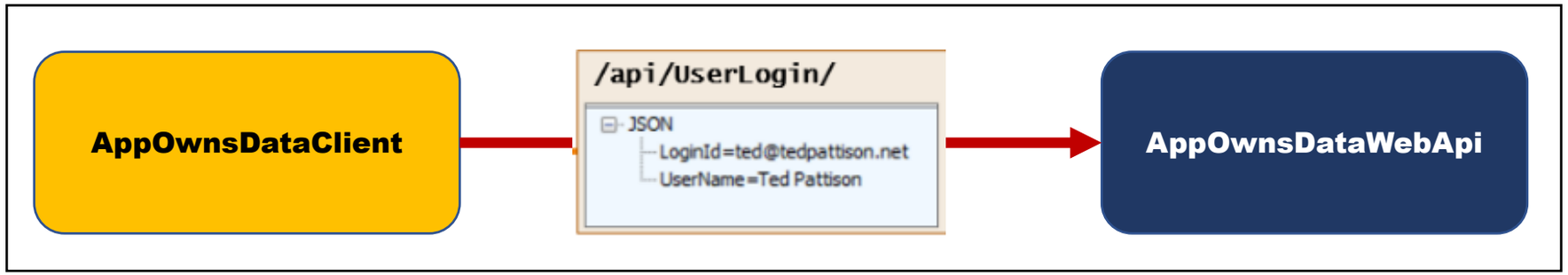
Using a single service principal

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



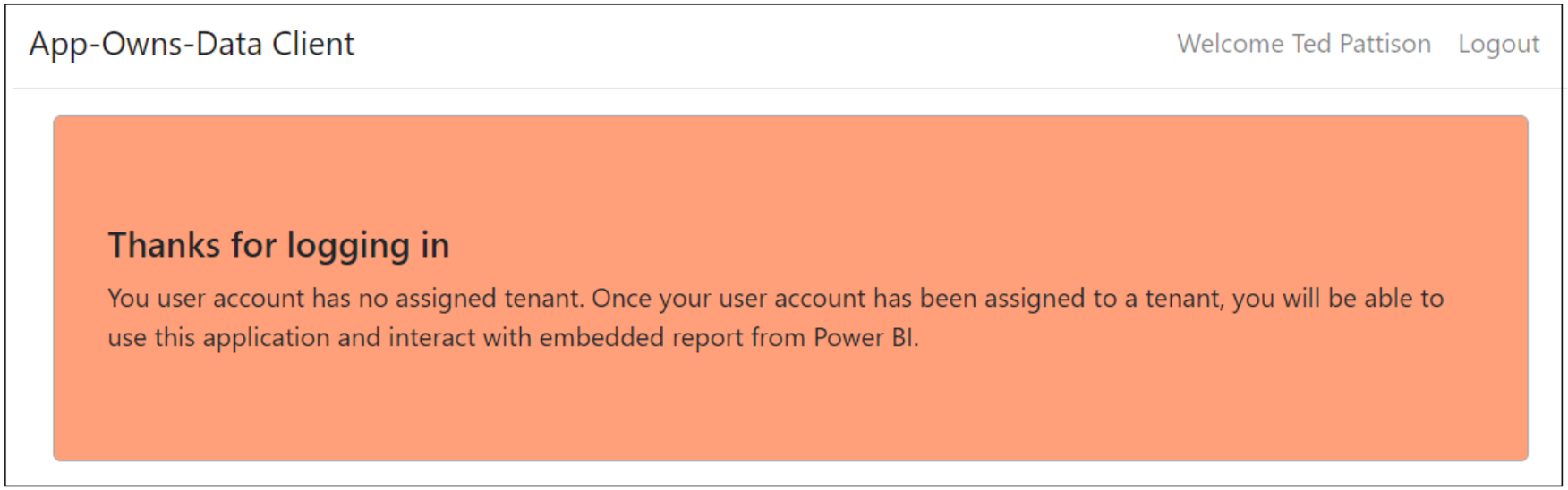
User Login

- 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



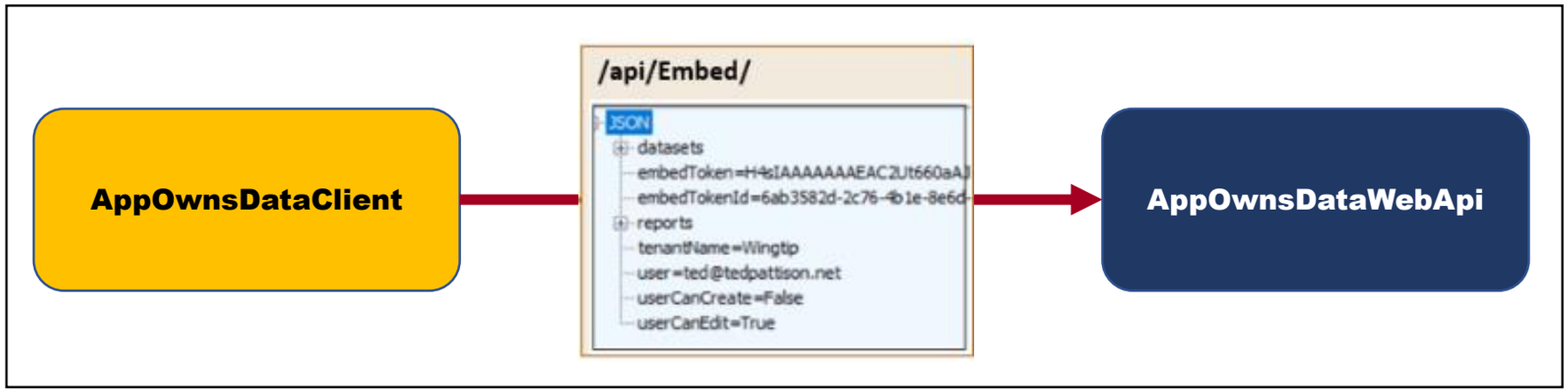
Unassigned users have no access to content

- New users created without being assigned to tenant
- Users cannot see content until they are assigned to tenant



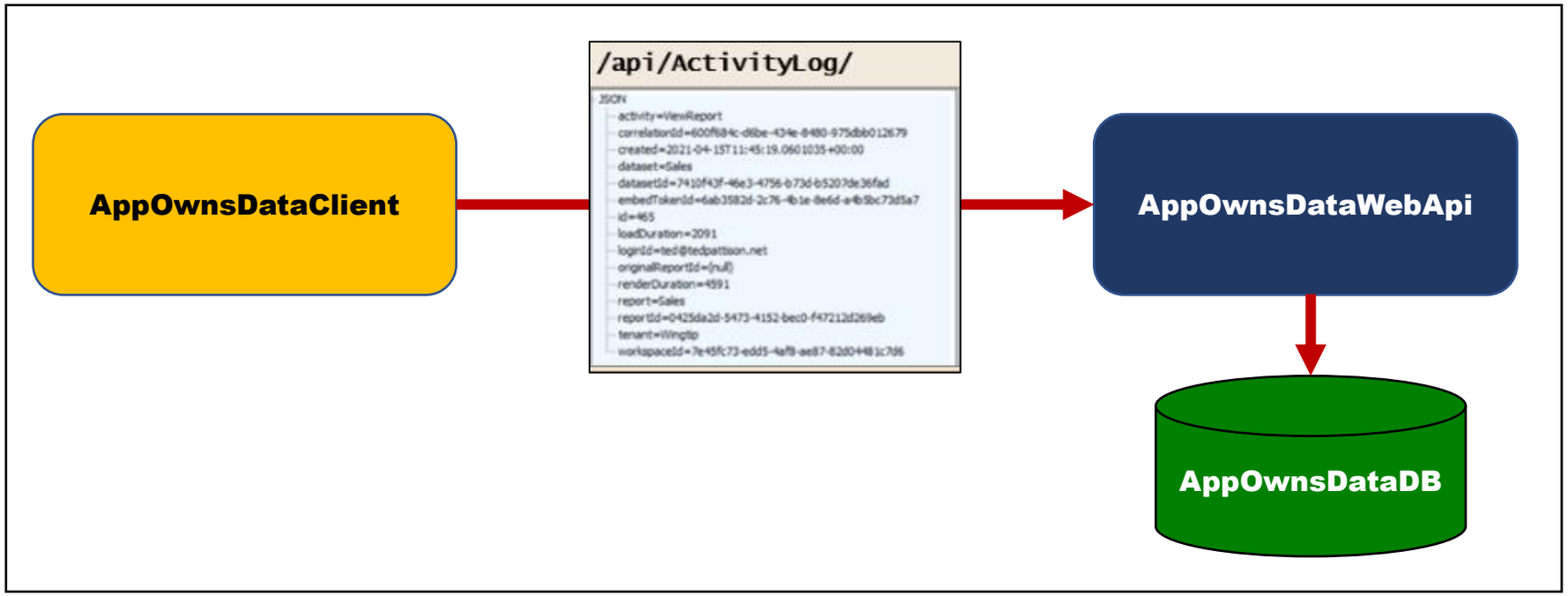
Embedding Data View Model

- 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



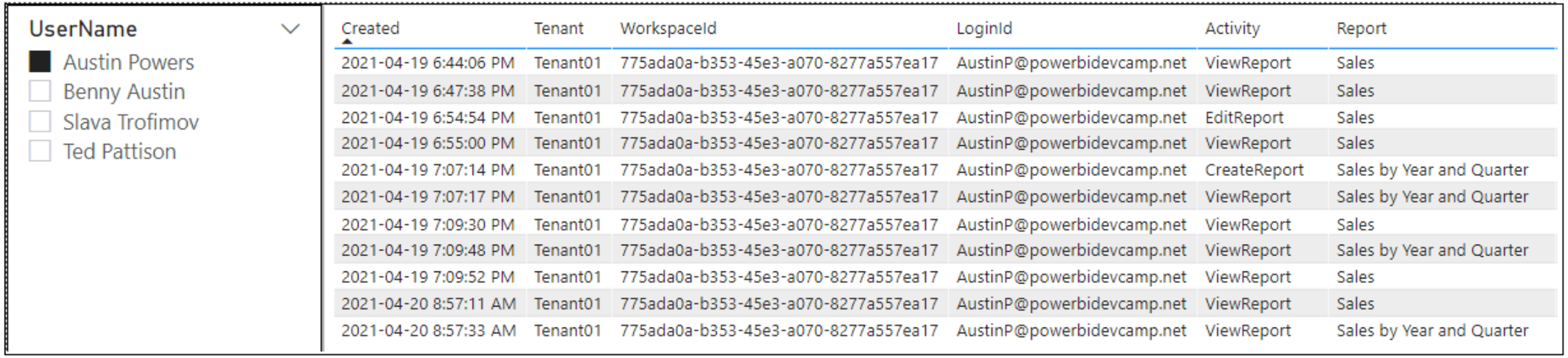
Adding a custom telemetry layer

- 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



UserName

- Austin Powers
- Benny Austin
- Slava Trofimov
- Ted Pattison

Created	Tenant	WorkspaceId	LoginId	Activity	Report
2021-04-19 6:44:06 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales
2021-04-19 6:47:38 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	ViewReport	Sales
2021-04-19 6:54:54 PM	Tenant01	775ada0a-b353-45e3-a070-8277a557ea17	AustinP@powerbidevcamp.net	EditReport	Sales
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

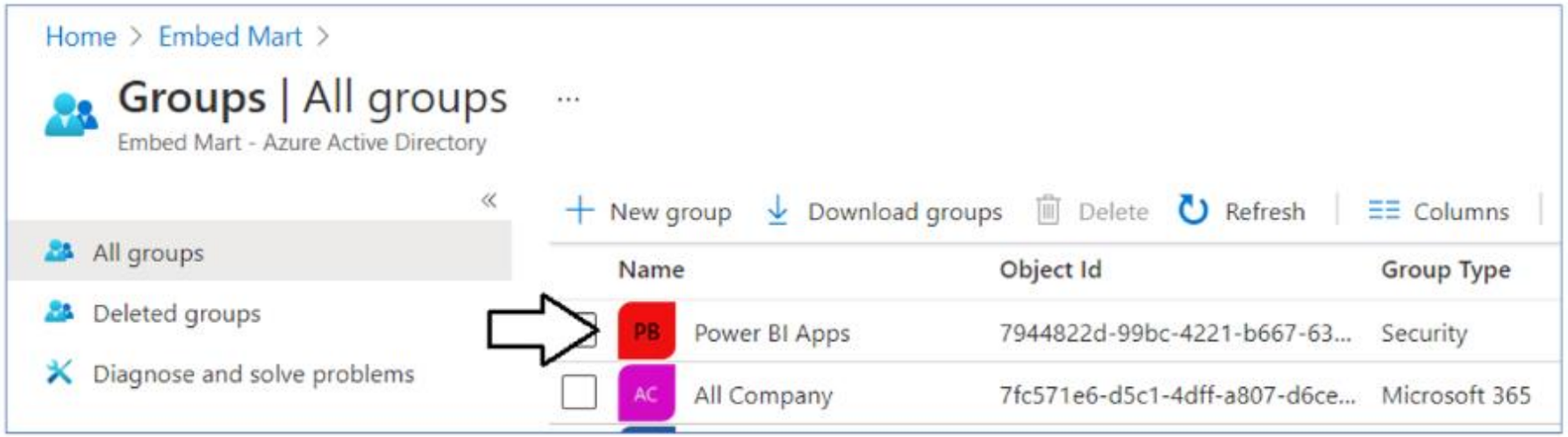
- ✓ Solution Architecture
- ➢ Set up your development environment
- Open the App-Owns-Data Starter Kit in Visual Studio
- Test the AppOwnsDataAdmin application
- Test the AppOwnsDataClient application
- Use activity logs to monitor user activity and report performance

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

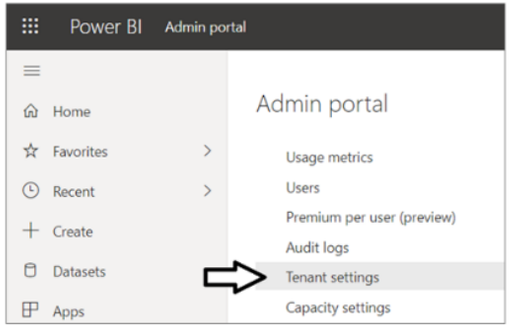
Create the Power BI Apps Group in Azure AD

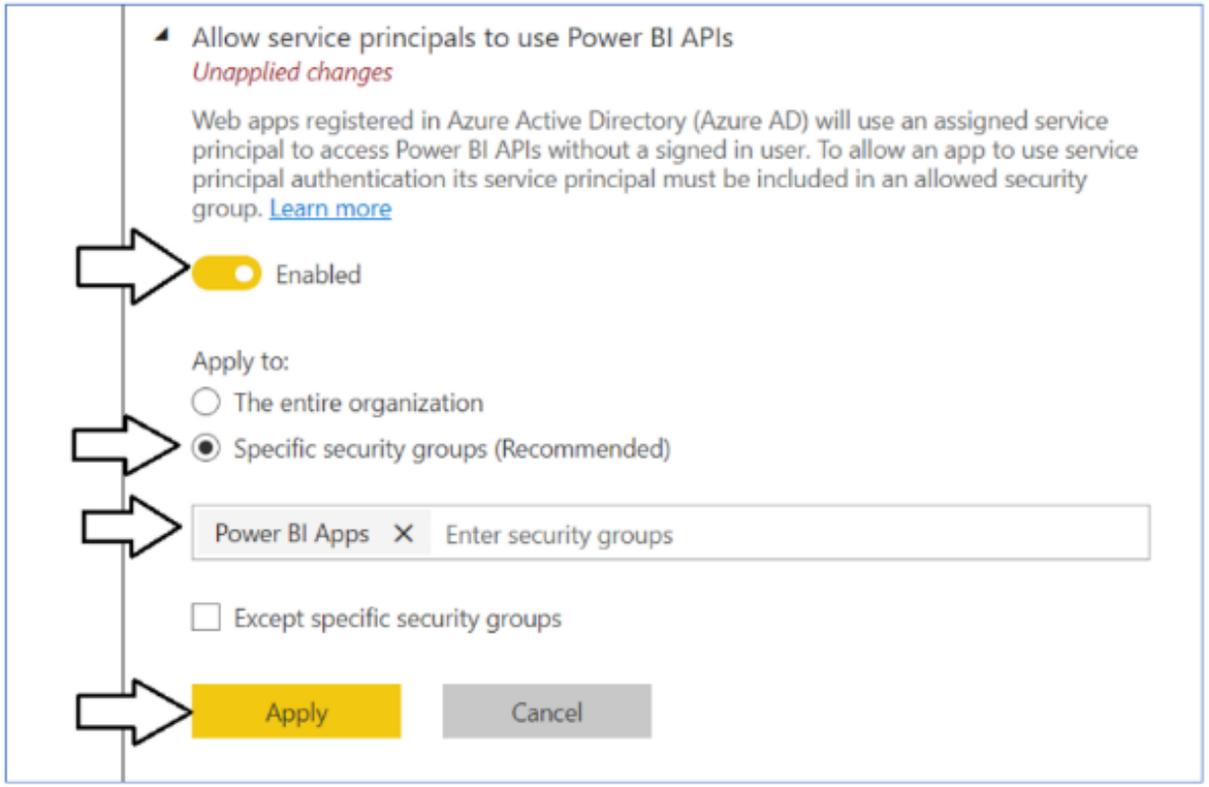
- Create a new Azure AD security group named **Power BI Apps**
  - Power BI Apps group used to configure Power BI access for service principal

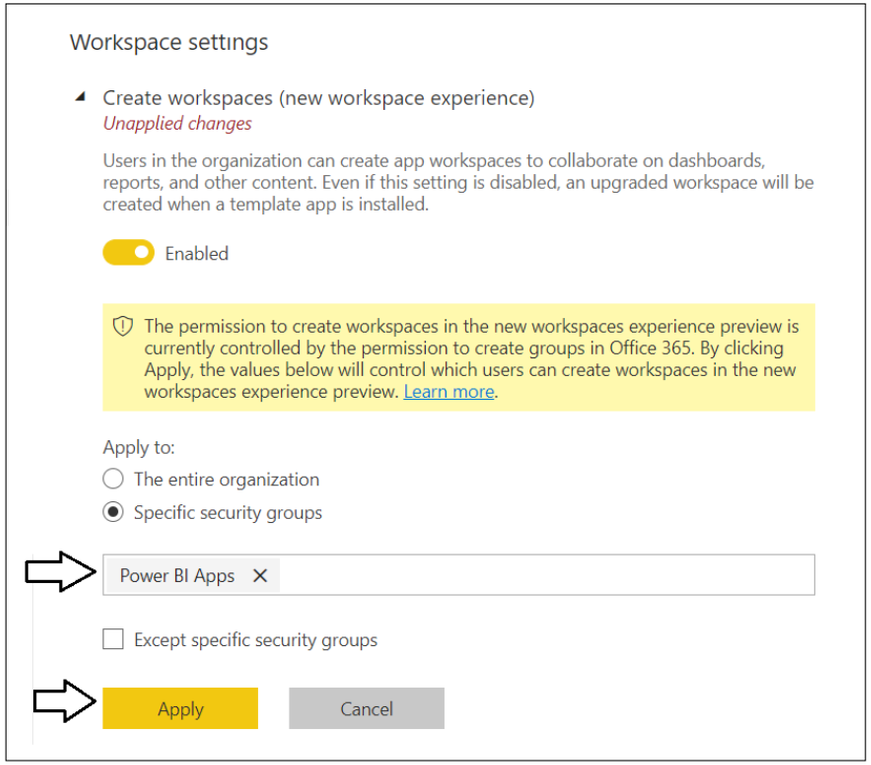


Configure service principal support in Power BI

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

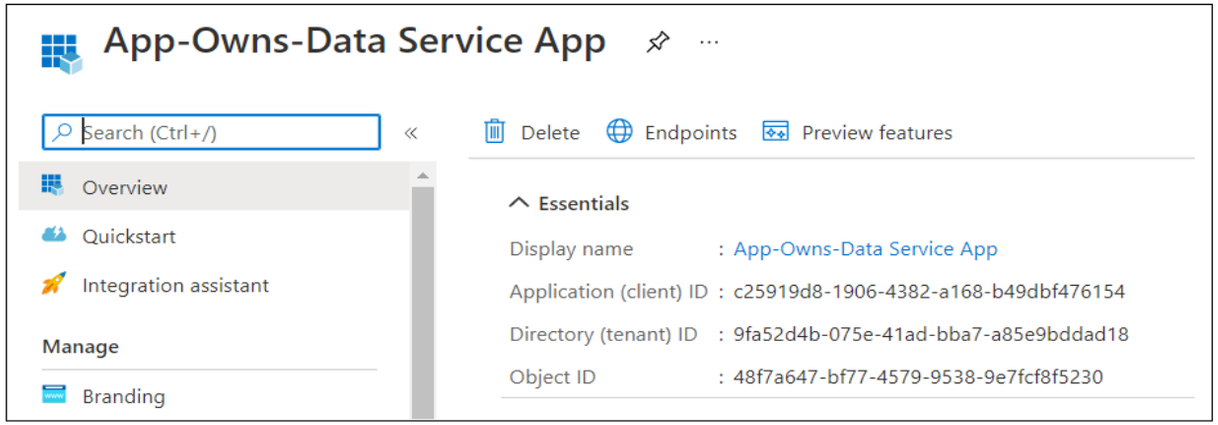


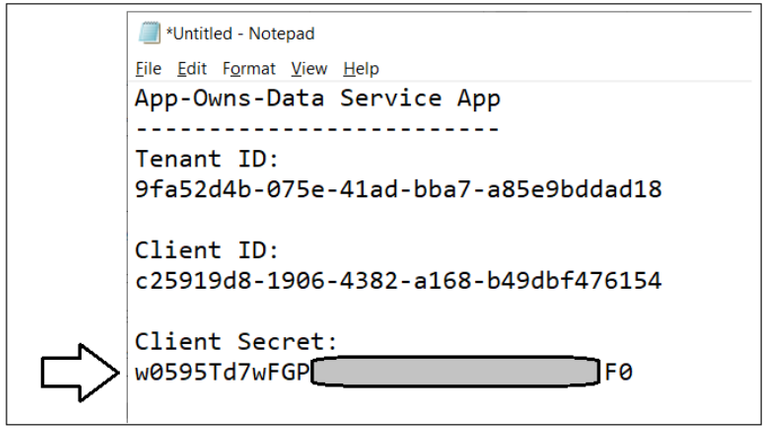




Create the App-Owns-Data Service App

- 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
  - Tenant ID, Client ID and Client Secret used to implement client credentials flow

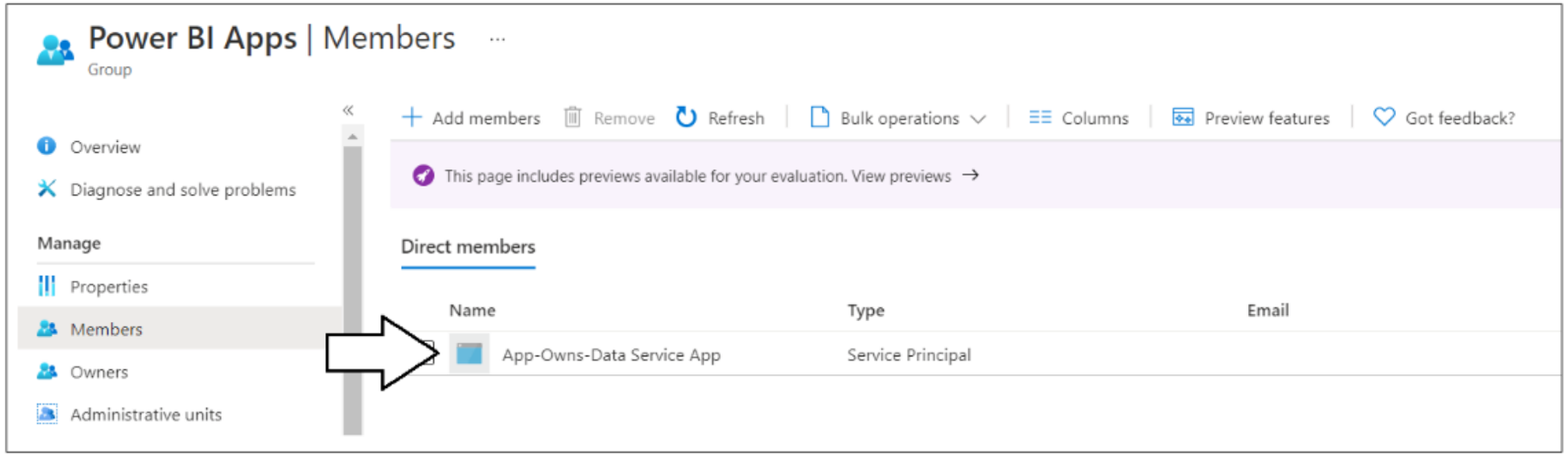




TIP: Use client secret when developing – move to client certificates for production

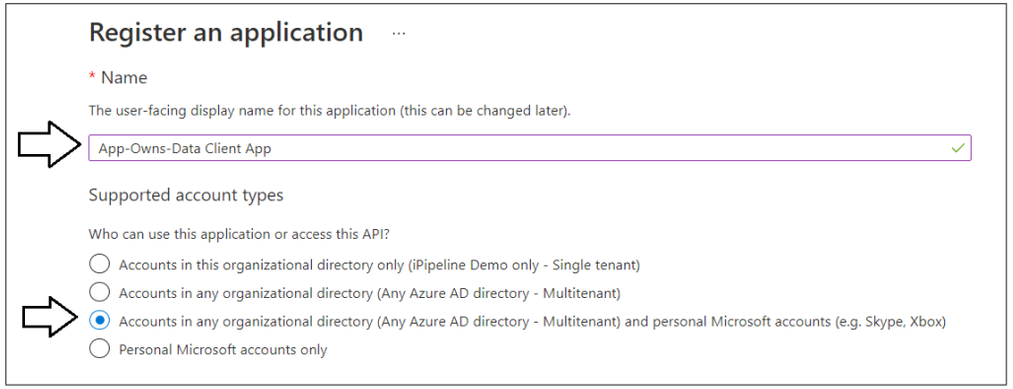
Add the Azure AD App to Power BI Apps Group

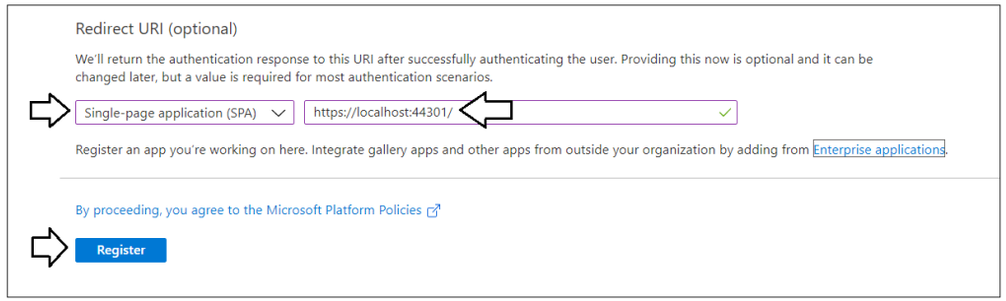
- Add **App-Owns-Data Service App** to **Power BI Apps** group
  - This gives service principal ability to call Power BI Service API
  - This also gives service principal ability to create new Power BI workspaces
  - Service principal is automatically admin of any workspace it creates

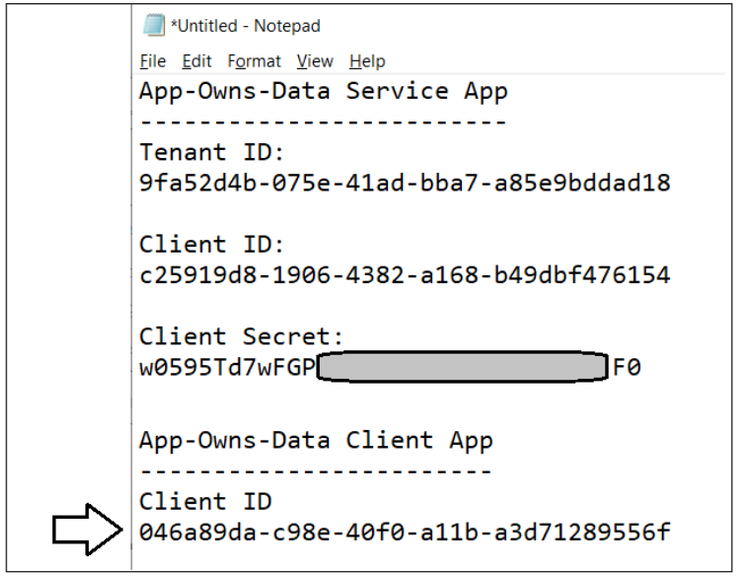


Create the App-Owns-Data Client App

- Azure AD application used to authenticated 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**



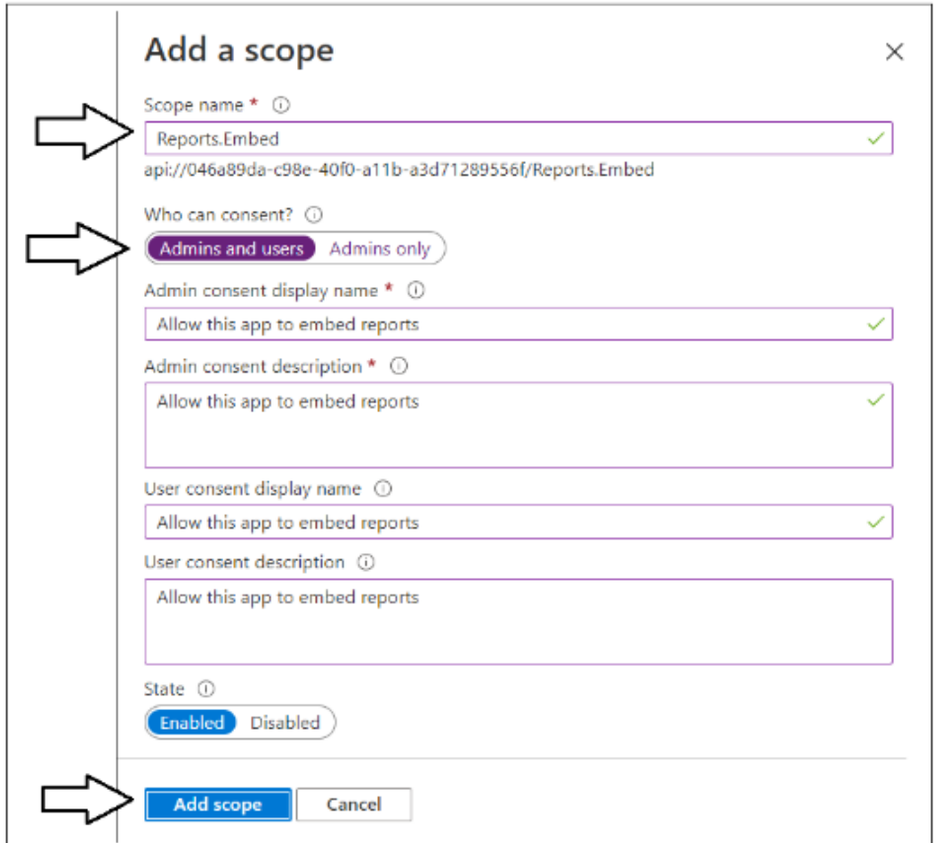


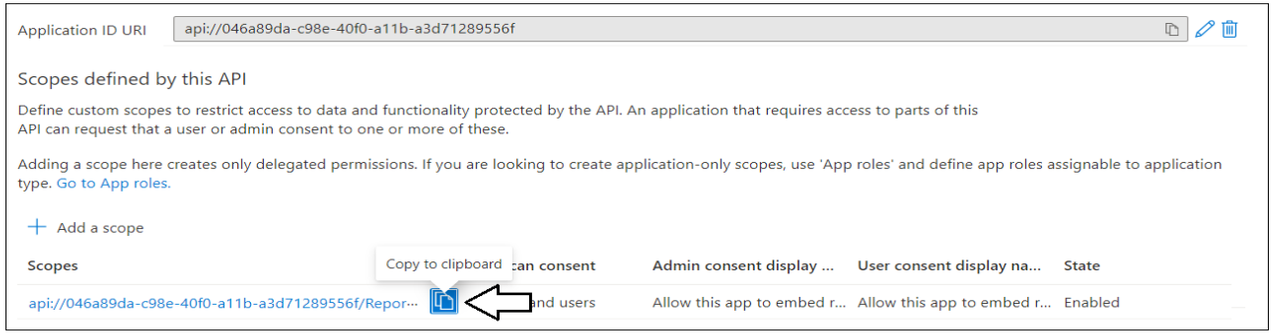


Remember – you don't have to use Azure AD as the identity provider

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`**





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

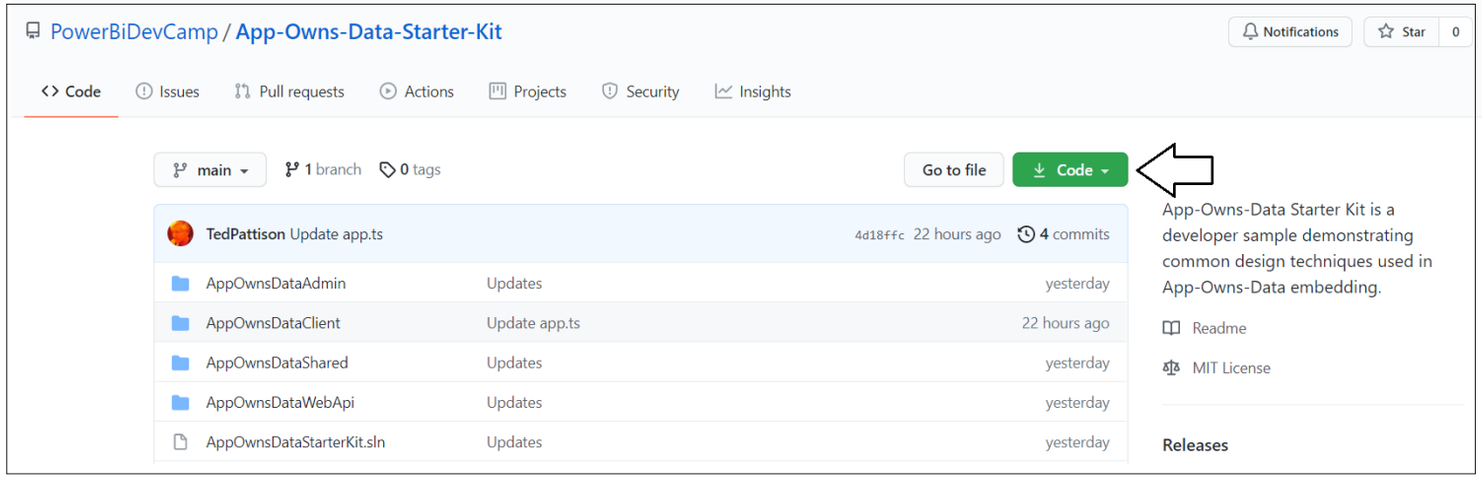
  https://visualstudio.microsoft.com/downloads/

- Visual Studio Code

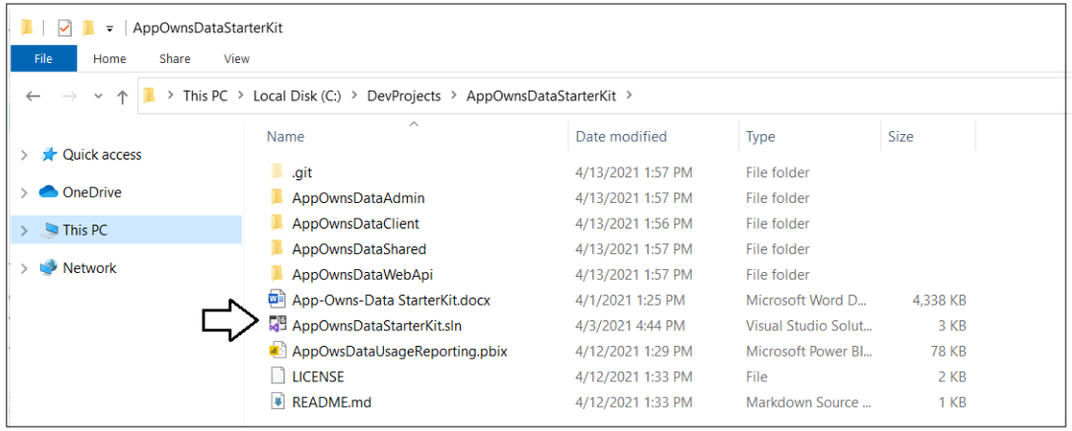
  https://code.visualstudio.com/Download

Download the source code

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

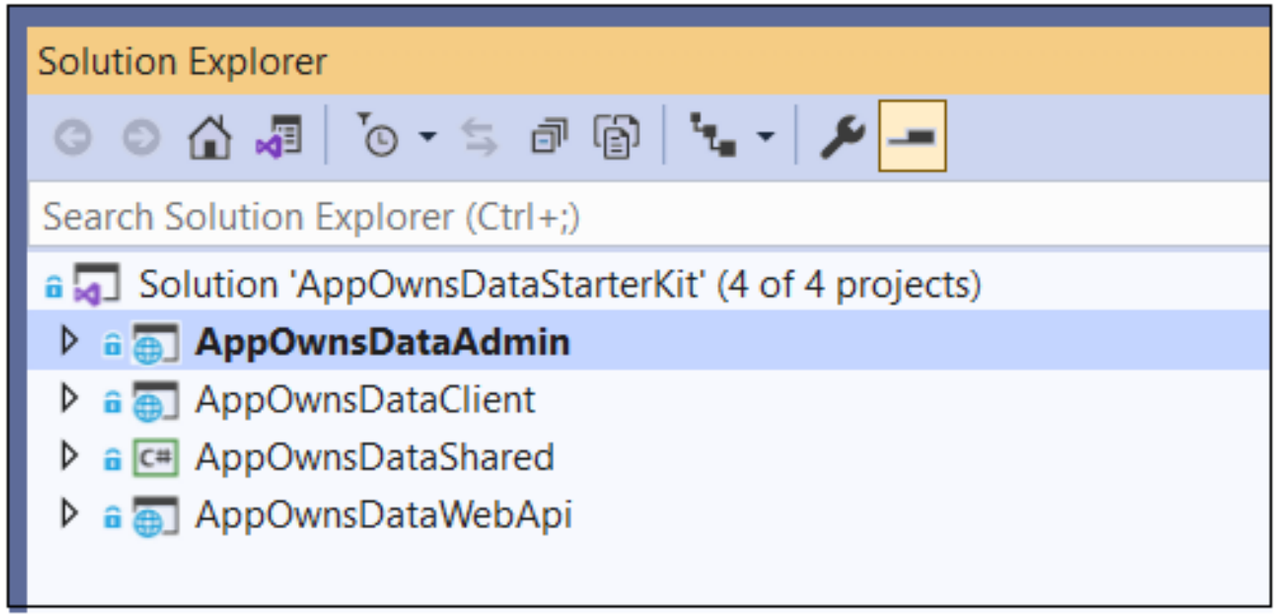


- Top-level folder contains .SLN file and four child folders for each project



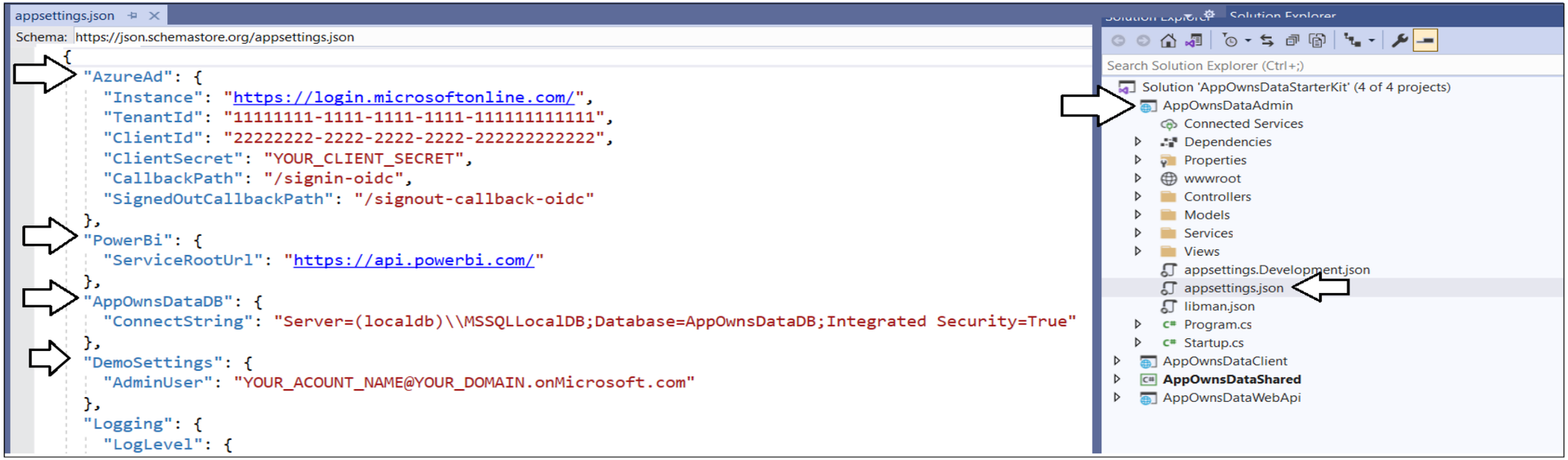
Open AppOwnsDataStarterKit.sln in Visual Studio

- **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



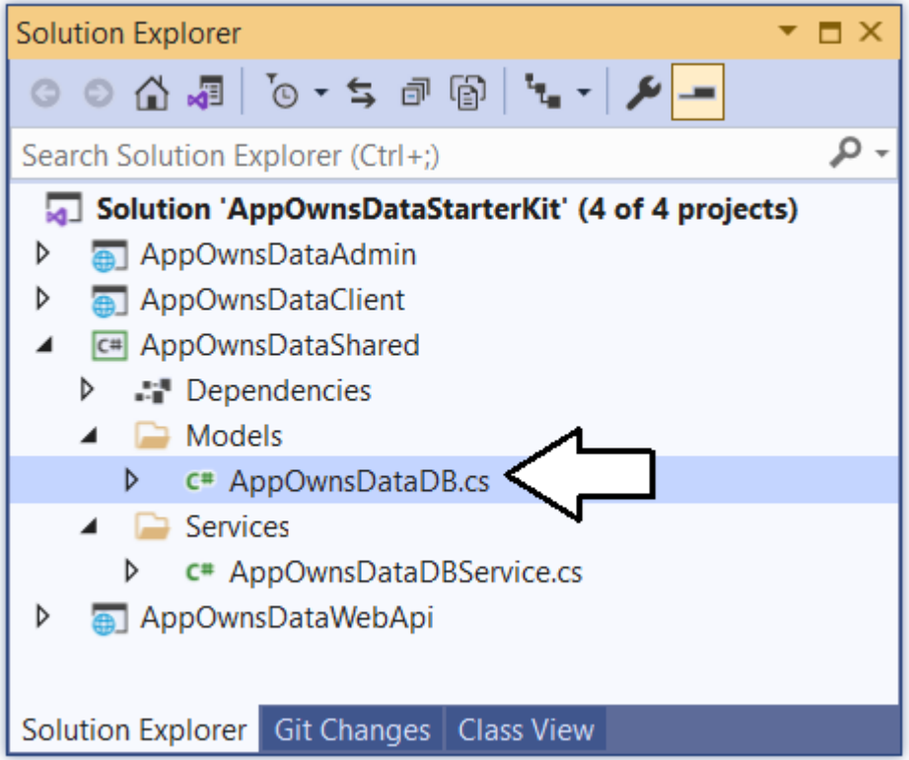
Update appsettings.json in AppOwnsDataAdmin

- 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:ConnectString`** 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

- AppOwsDataDB 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

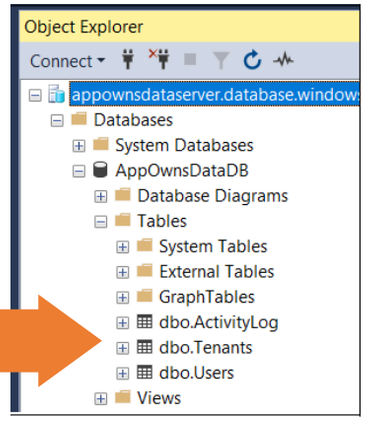
```
public class PowerBiTenant {
  [Key]
  public string Name { get; set; }
  public string WorkspaceId { get; set; }
  public string WorkspaceUrl { get; set; }
  public string DatabaseServer { get; set; }
  public string DatabaseName { get; set; }
  public string DatabaseUserName { get; set; }
  public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

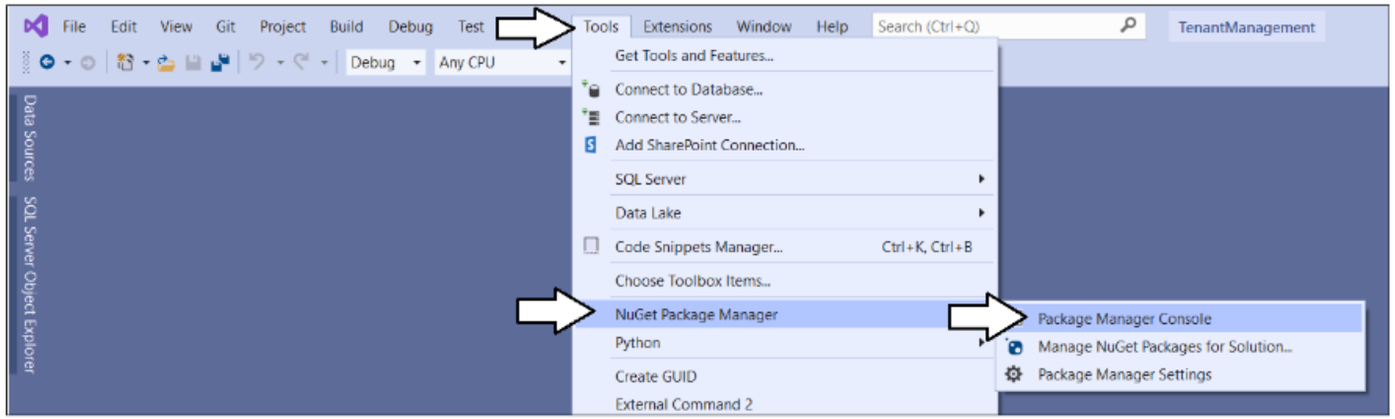
```
public class AppOwnsDataDB : DbContext {

  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; }
}
```

Entity Framework Core

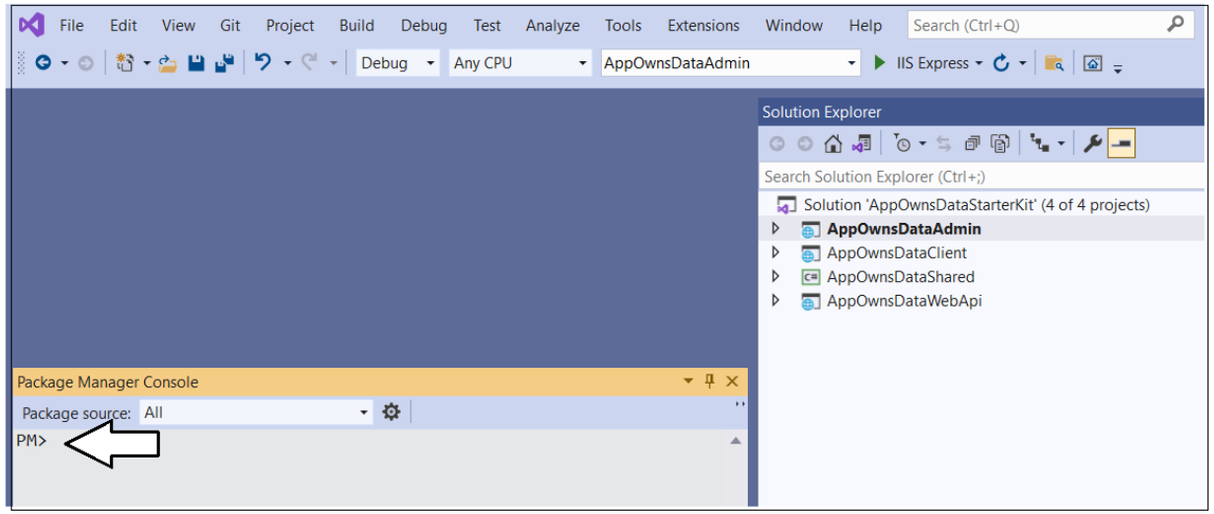


Package Manager Console

- Open the NuGet Manager Console

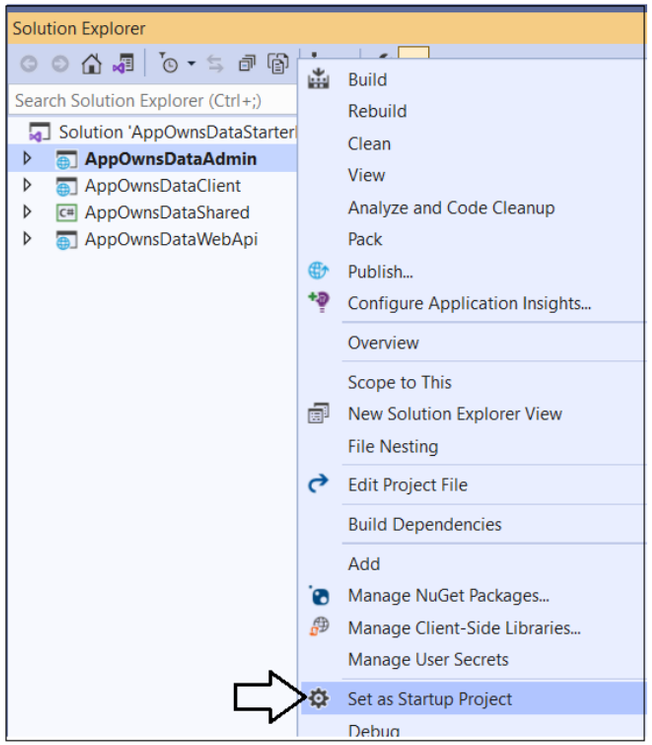


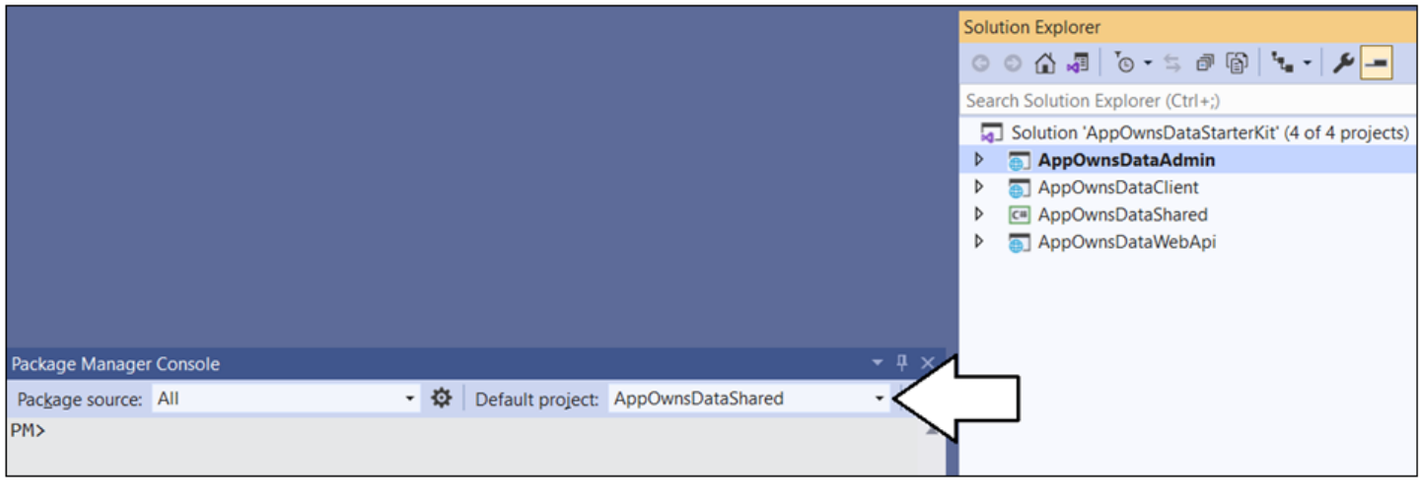
- NuGet Manager Console provides a prompt to type and execute PowerShell commands



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





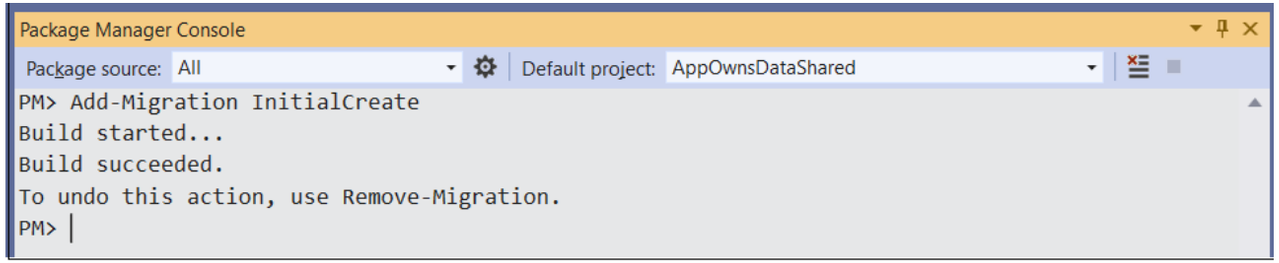
Create the AppOwnsDataDB database

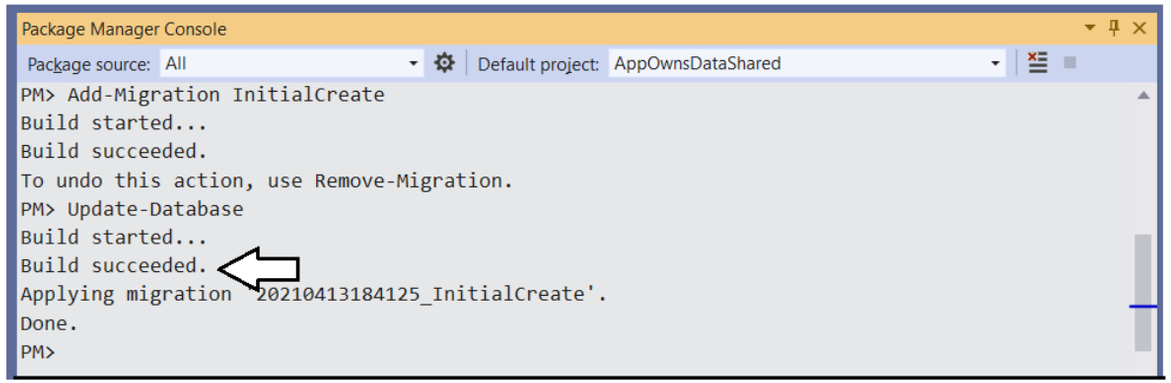
- Execute PowerShell command to generate code to create database

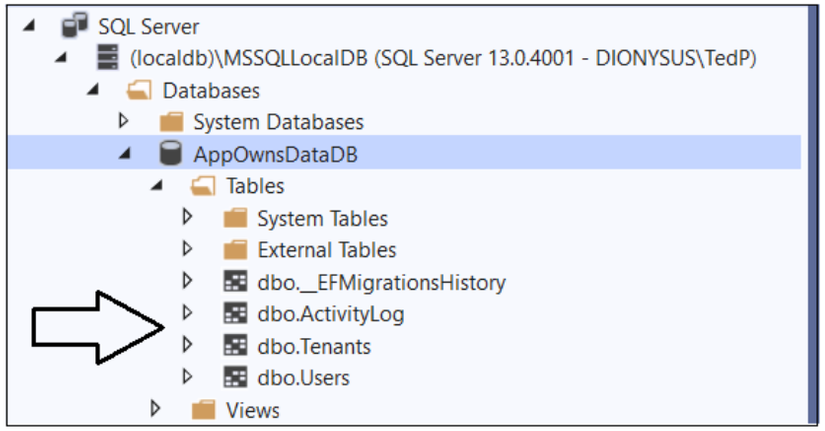
```
Add-Migration InitialCreate
```

- Execute PowerShell command to execute code to create the database

```
Update-Database
```





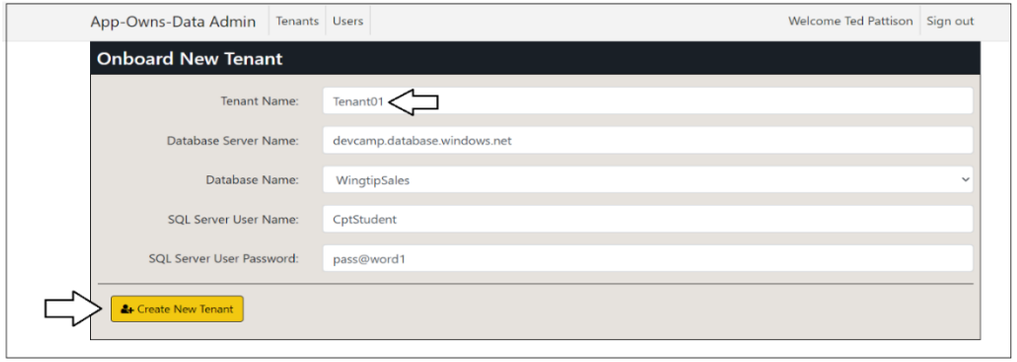


Agenda

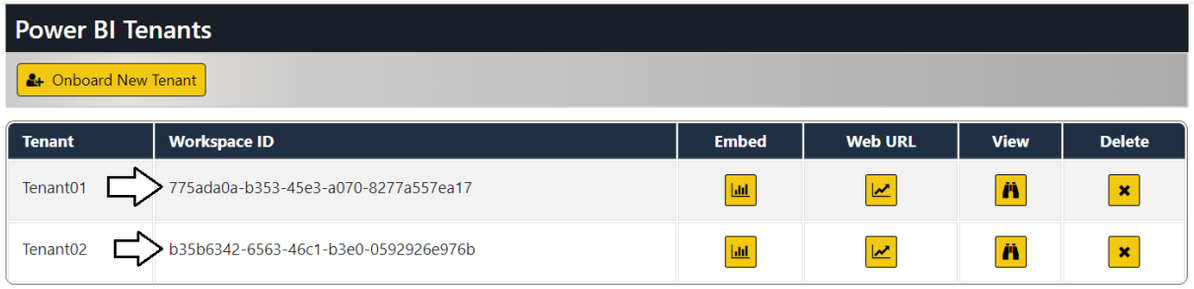
- ✓ Solution Architecture
- ✓ Set up your development environment
- ✓ Open the App-Owns-Data Starter Kit in Visual Studio
- ➢ Test the AppOwnsDataAdmin application
- Test the AppOwnsDataClient application
- Use activity logs to monitor user activity and report performance

Create New Customer Tenants

- Create a new customer tenant



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

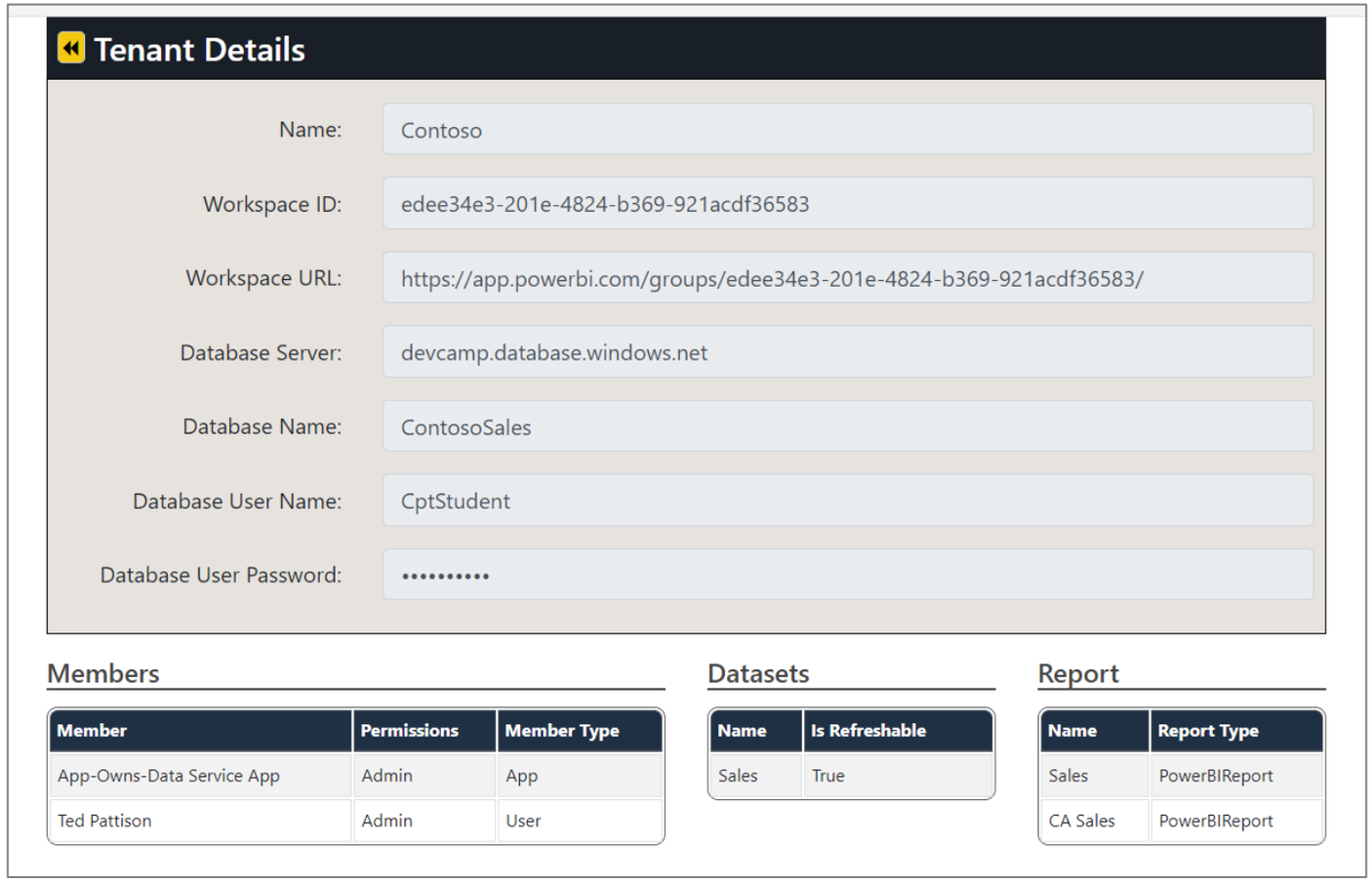


- Behind the scenes, **AppOwnsDataApp** adds record in **Tenants** table in **AppOwnsDataDB**

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

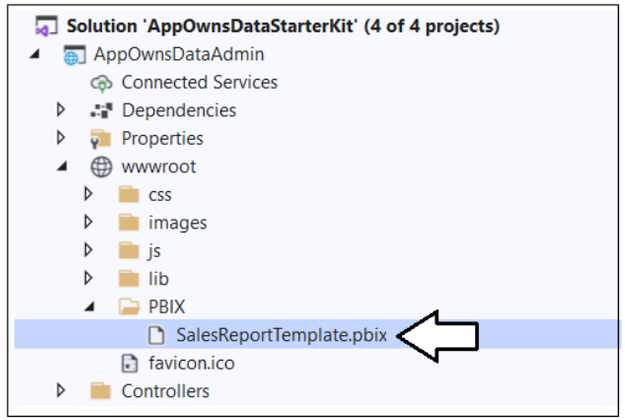
Tenant Details

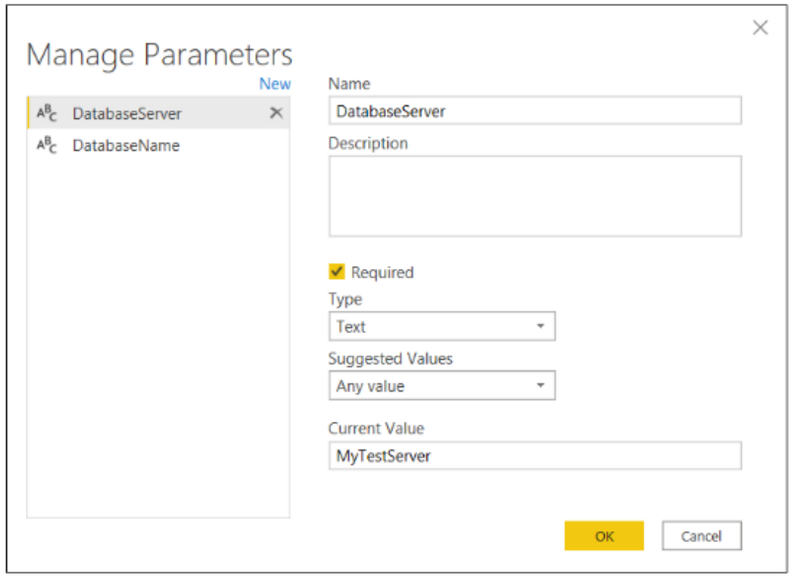
- Tenant Details page lists members, datasets and reports
  - AppOwnsDataAdmin calls Power BI REST API to discover workspace artifacts

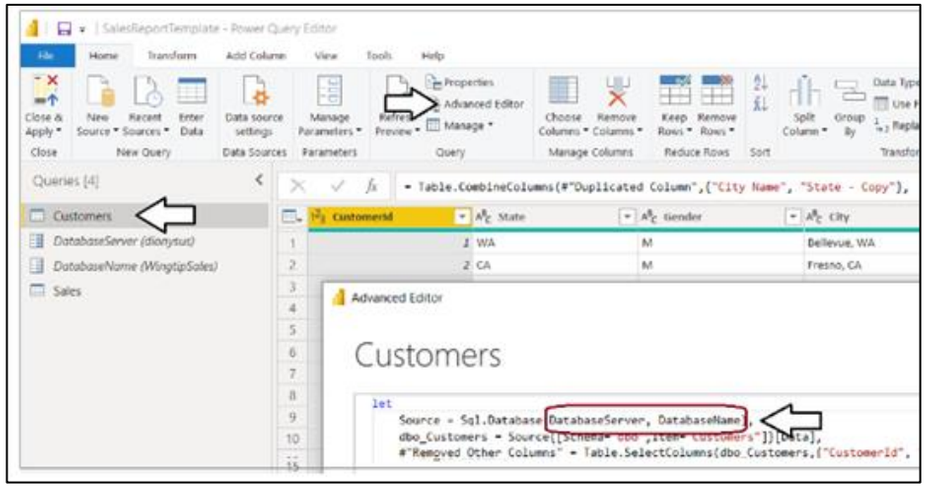


Understanding SalesReportTemplate.pbix

- **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 to customer database

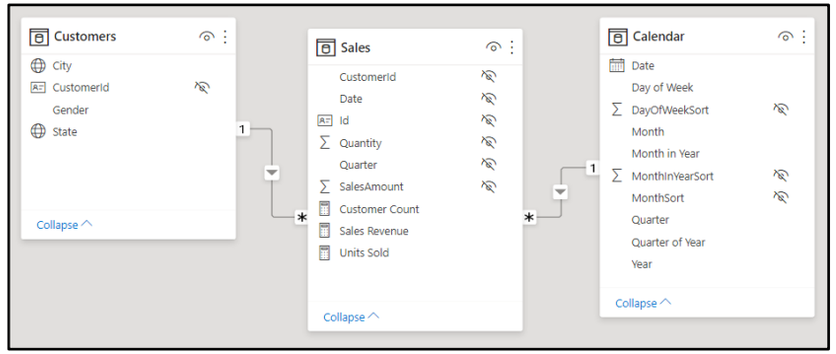


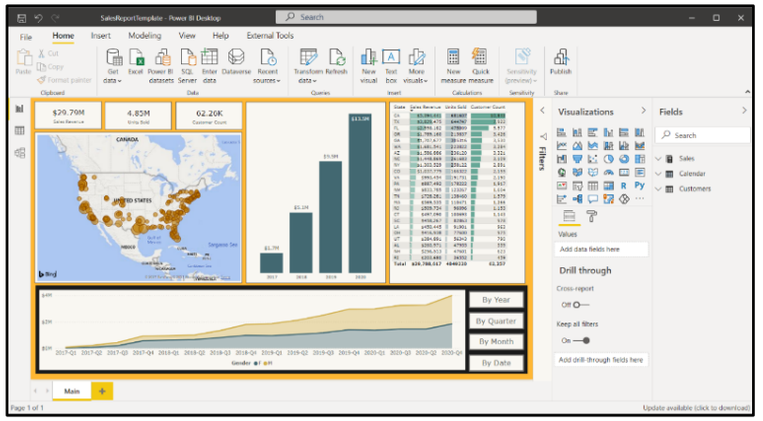


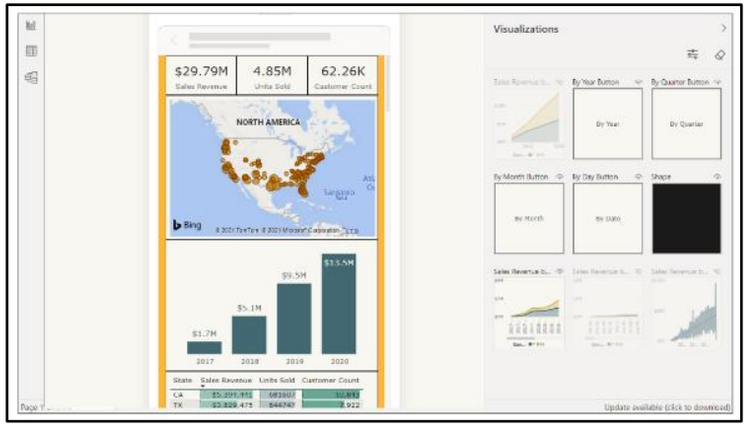


Dataset and Report Design

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

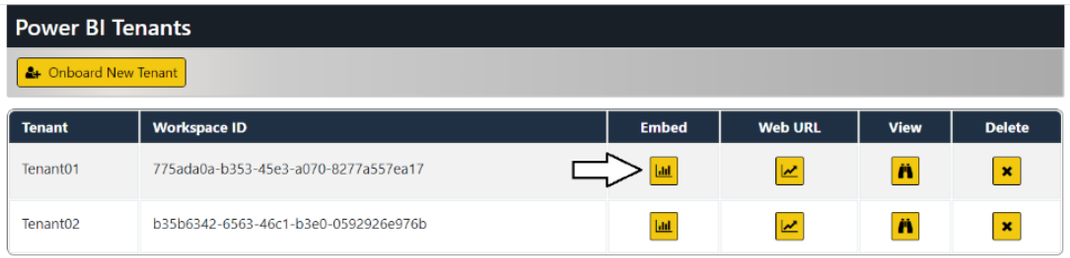


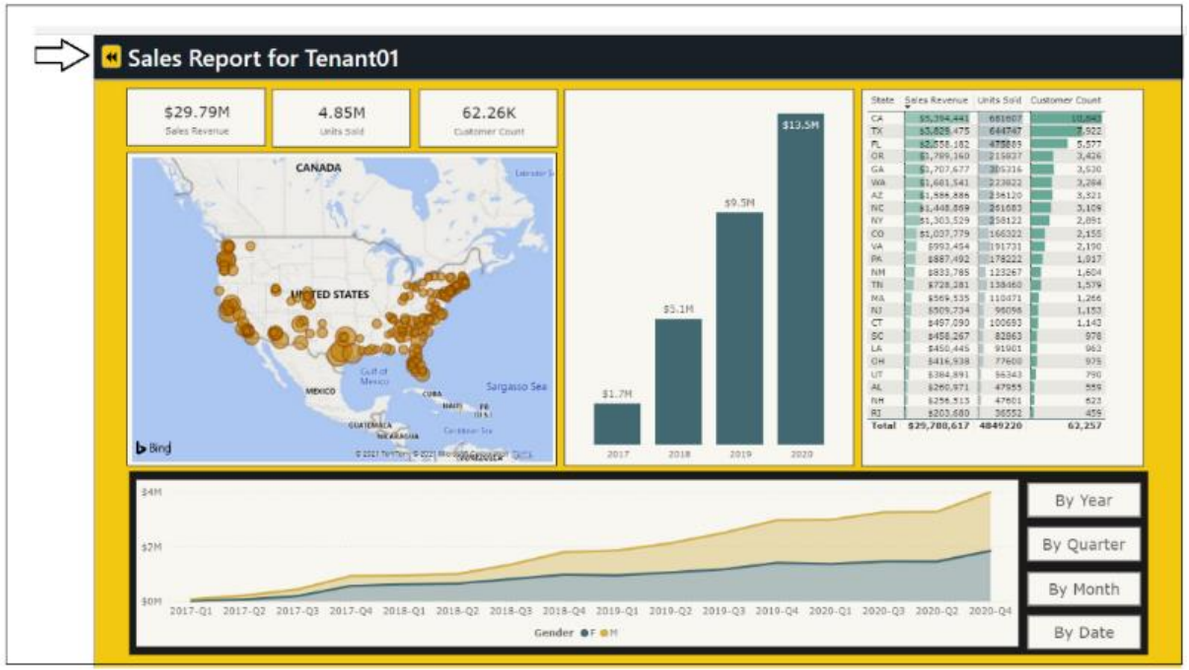




Embedding Reports in AppOwnsDataAdmin

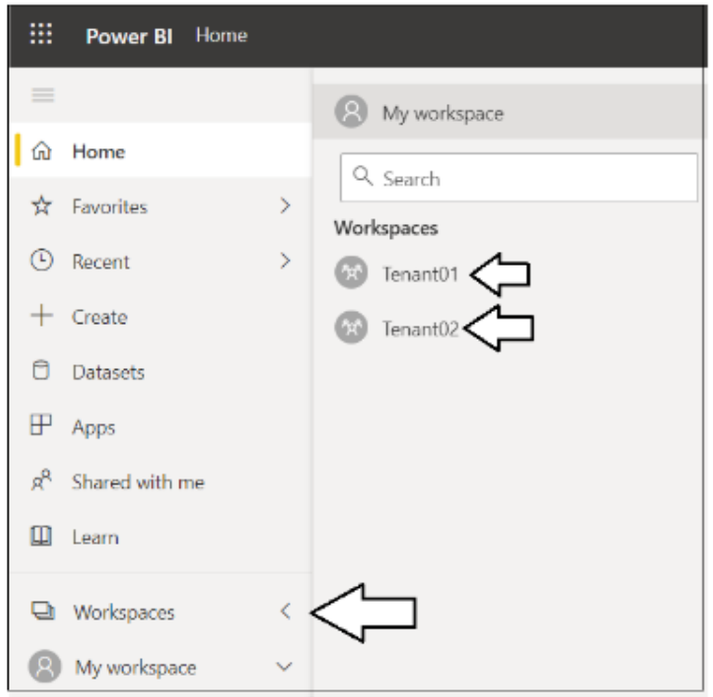
- **AppOwnsDataAdmin** provides **Embed** view for each tenant
  - Allows user to view the **Sales** report from any tenant

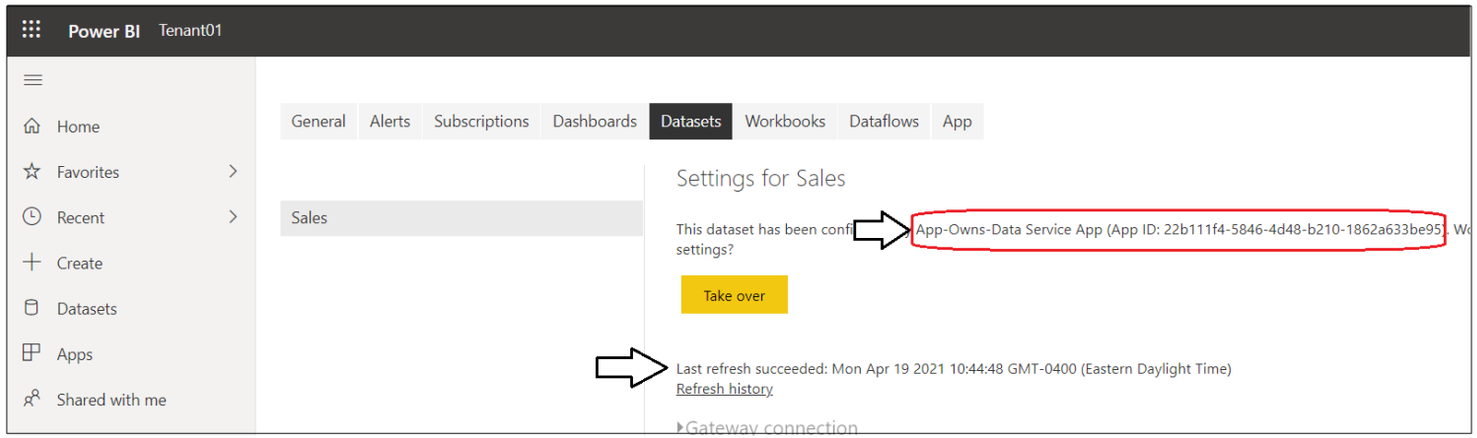




What's been created in Power BI?

- 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



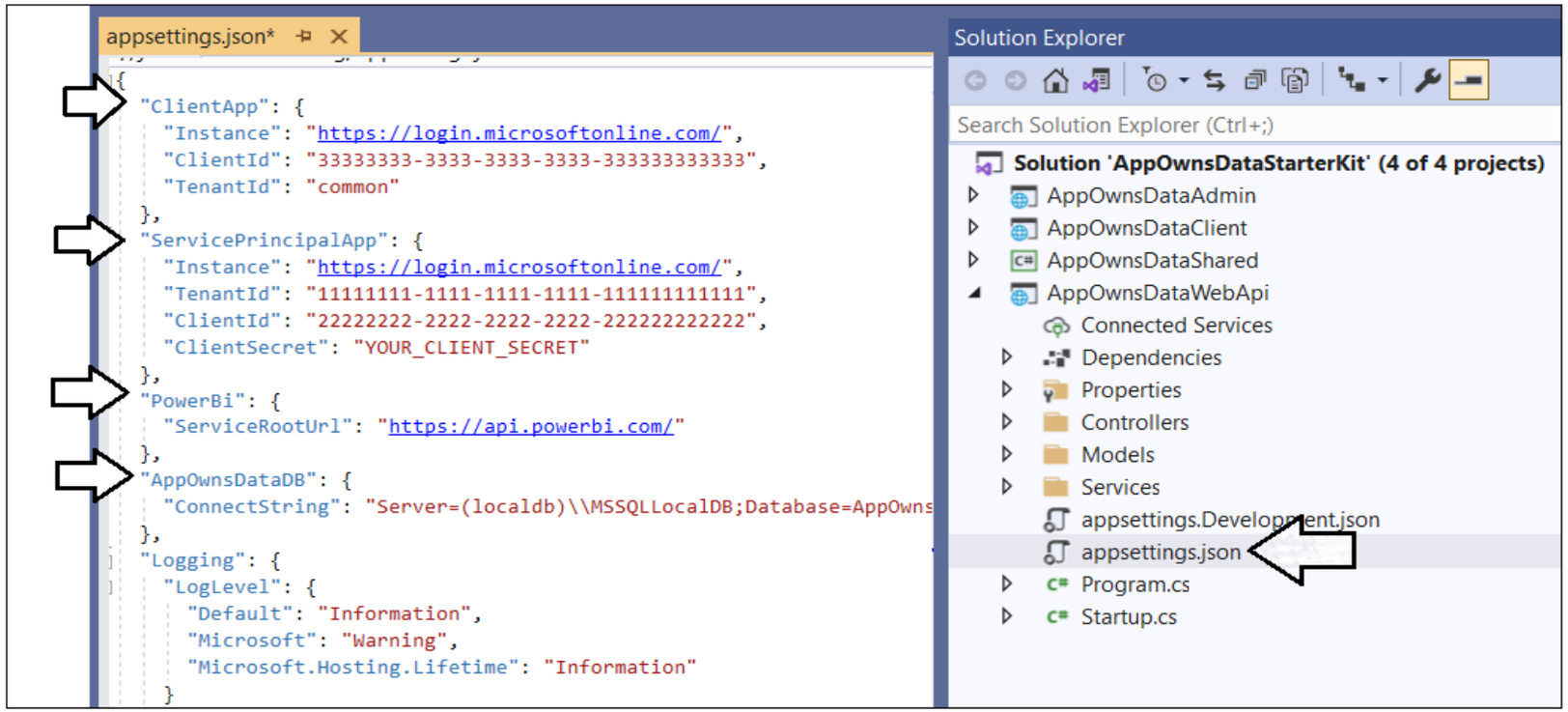


Agenda

- ✓ 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

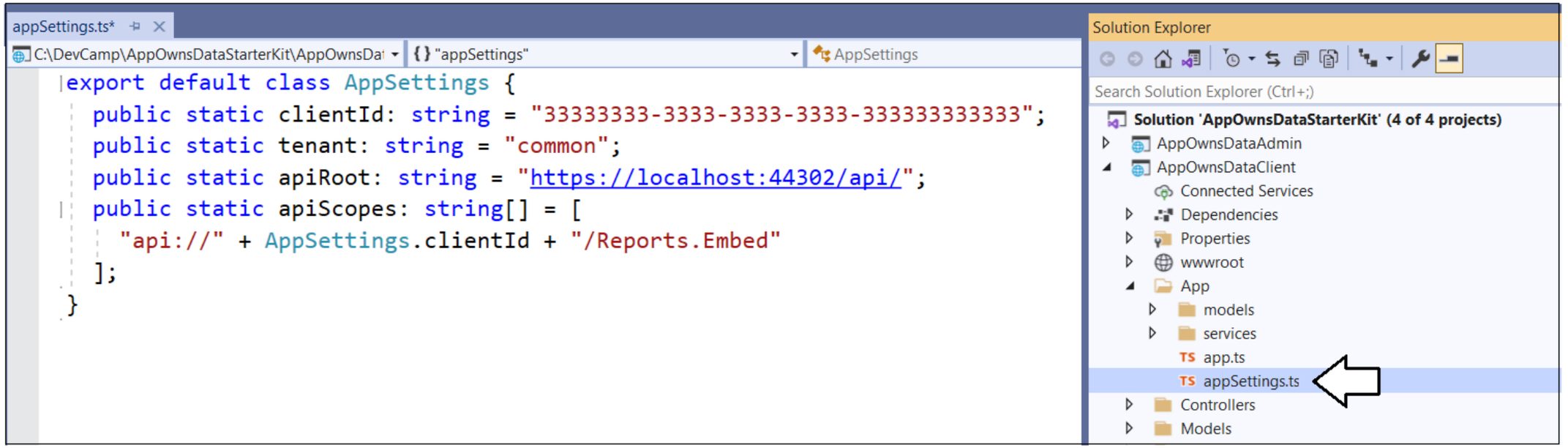
Update appsettings.json in AppOwnsDataWebApi

- 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:ConnectString`** is valid for your development environment



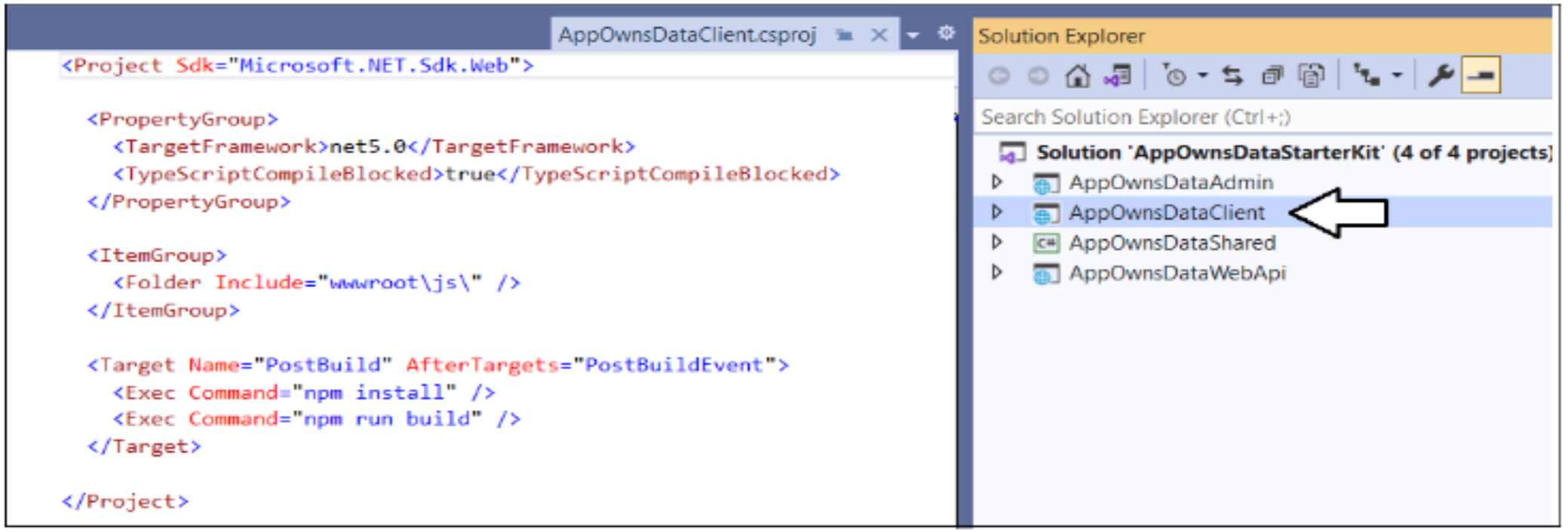
Configure and build AppOwnsDataClient

- 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



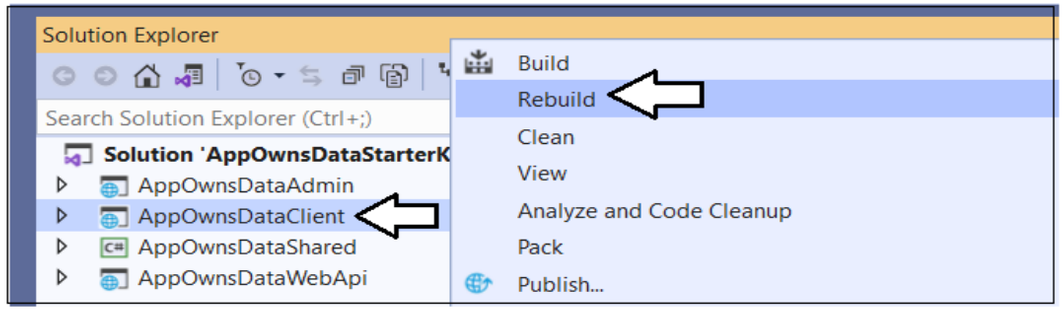
Node.js integration

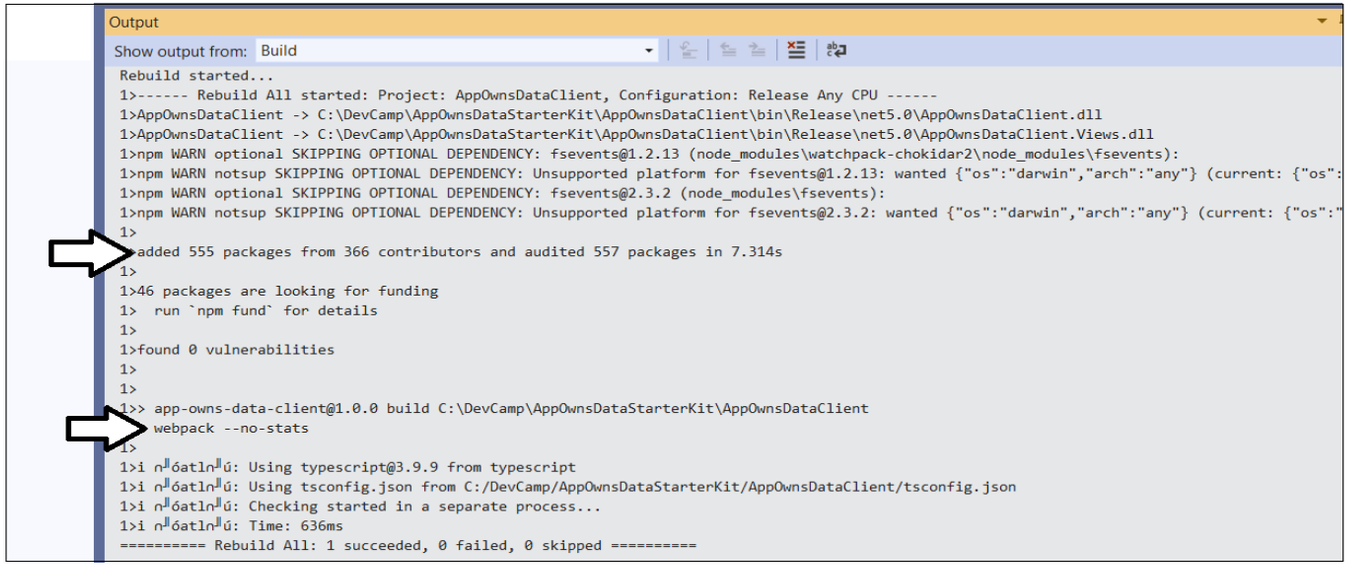
- AppOwnsDataClient project built as a Node.js project
  - You must install Node.js in order to build the project
  - `npm install` command used to restore Node.js packages
  - `npm run build` command triggers webpack to compile Typescript into JavaScript

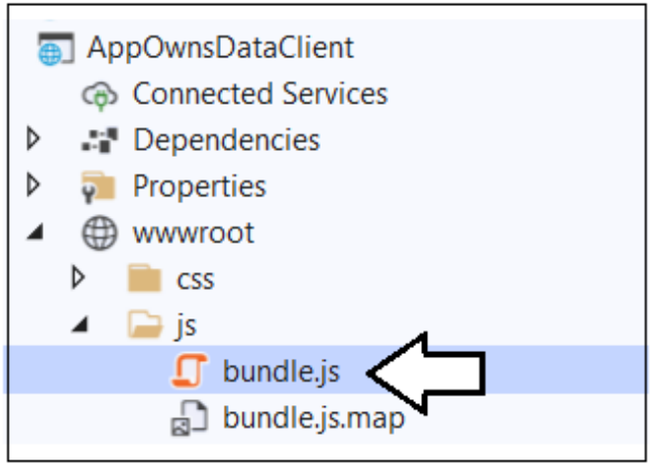


Building the AppOwnsDataClient Project

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

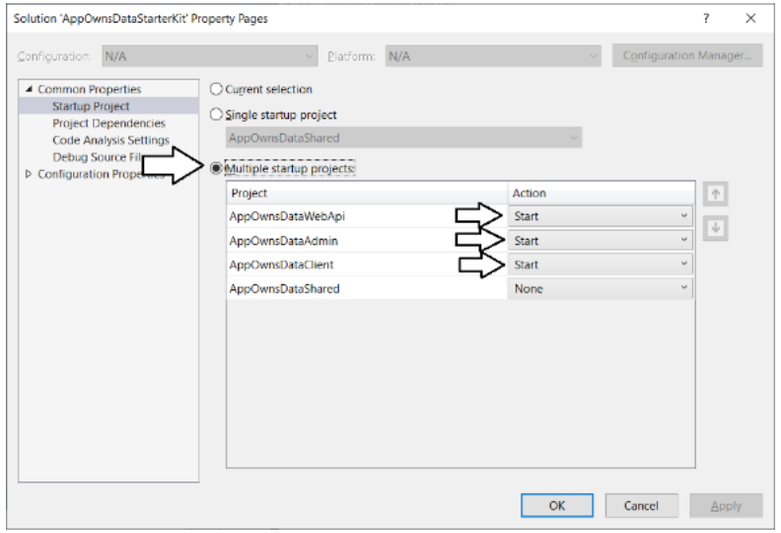


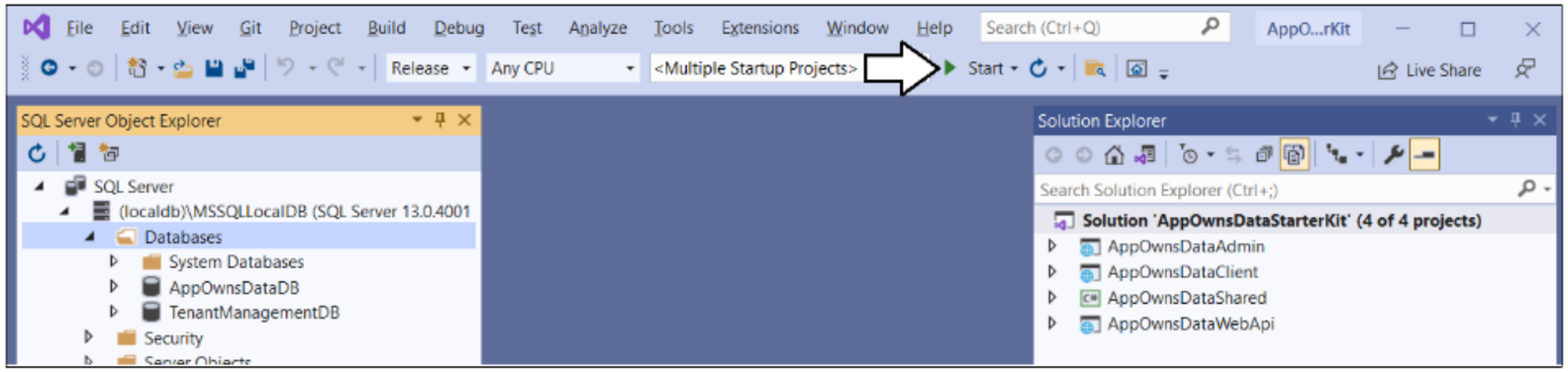


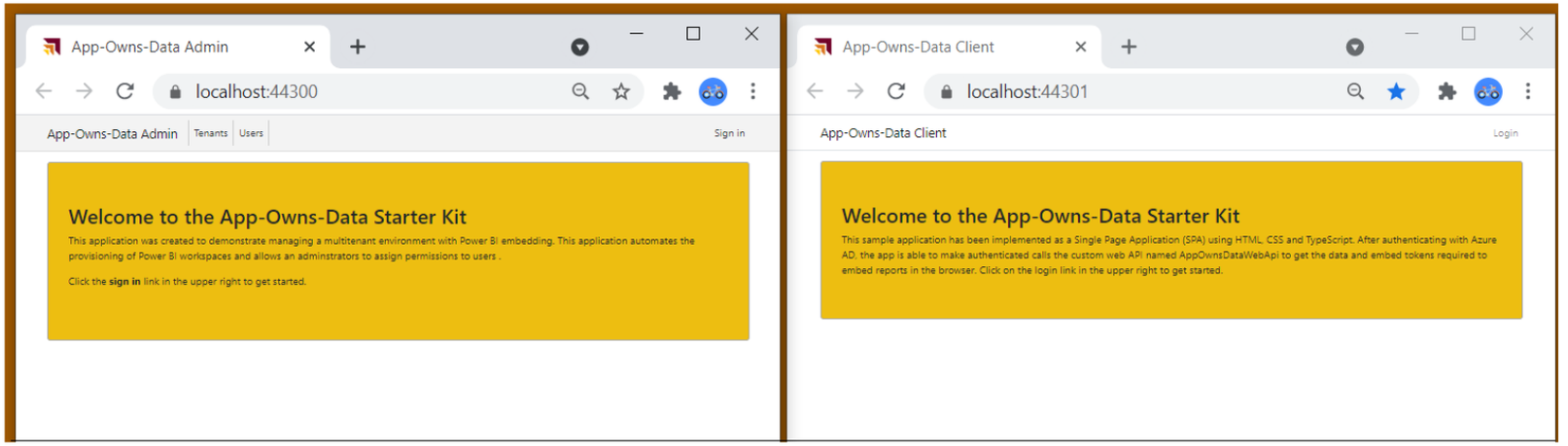


Test AppOwnsDataClient in Visual Studio

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

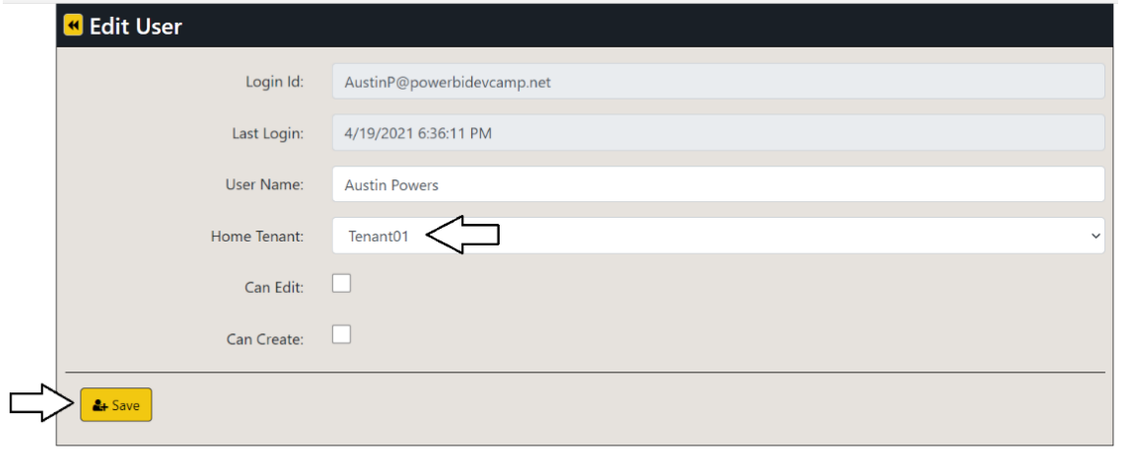


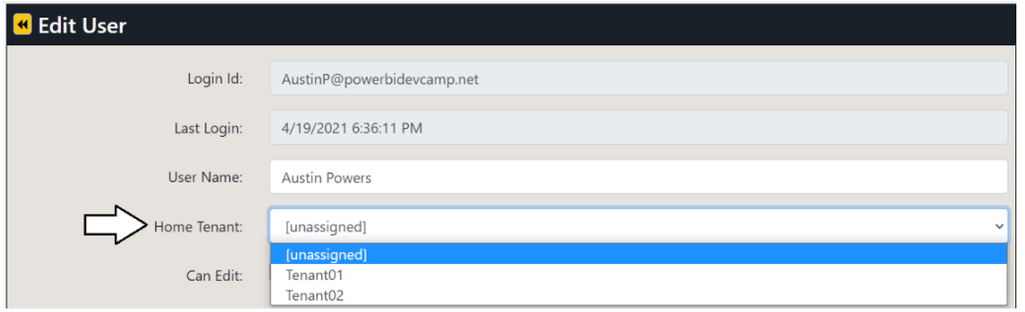


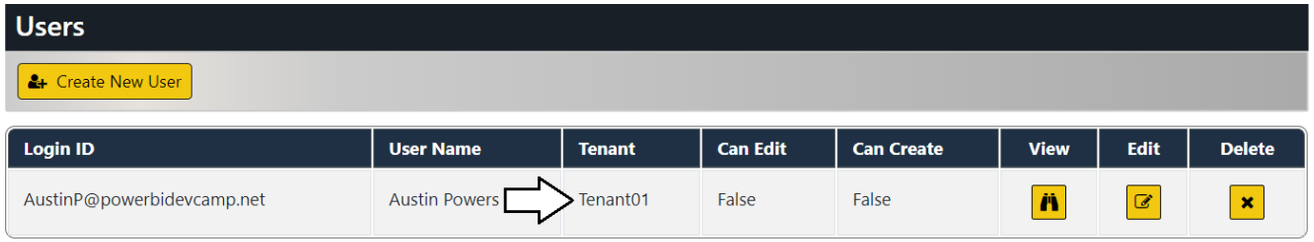


Testing the App-Owns-Data Applications Side by Side

- 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**

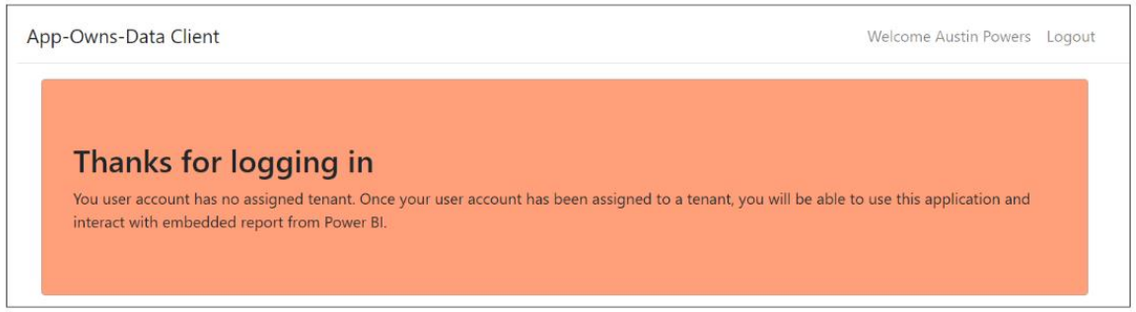




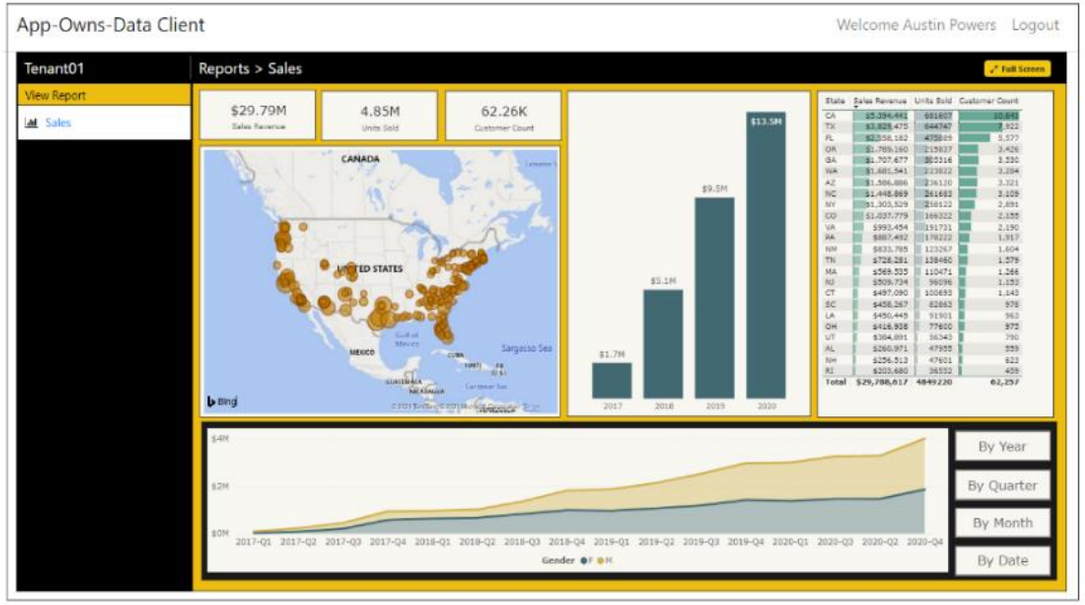


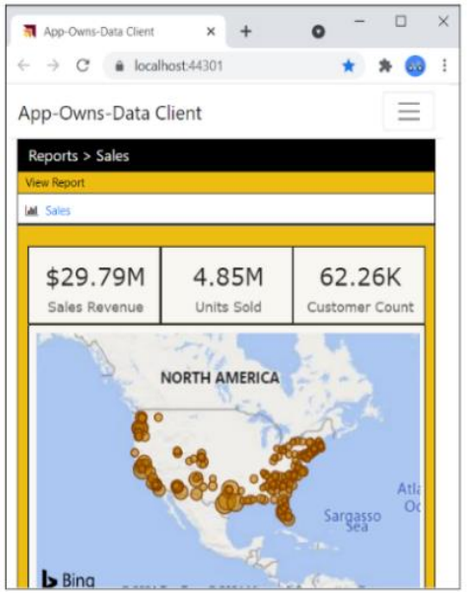
AppOwnsDataClient User Experience

- Test user experience when user is unassigned



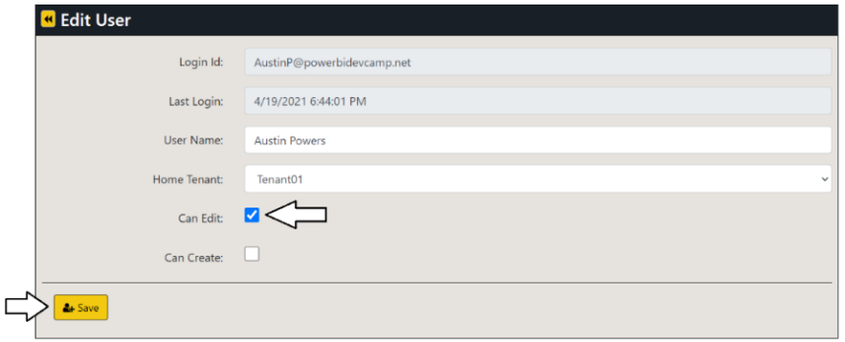
- Test experience with user assigned - make browser window narrow to see mobile view

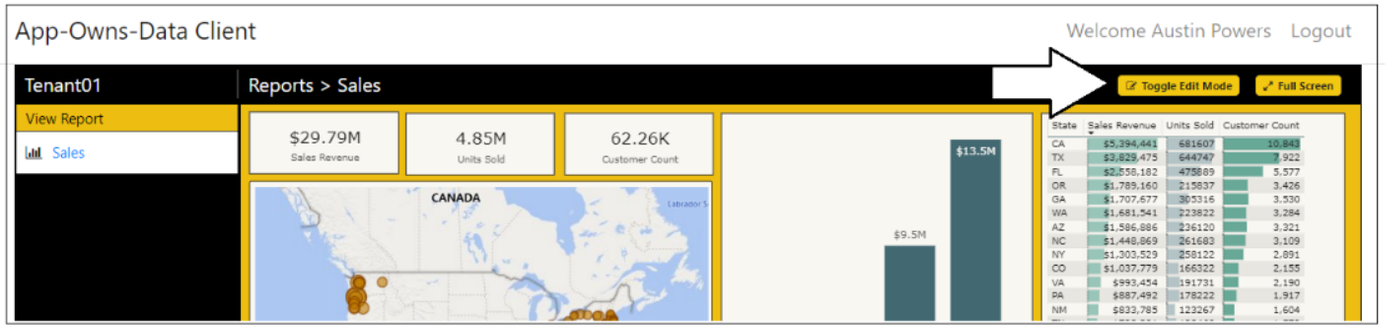




Edit reports using AppOwnsDataClient

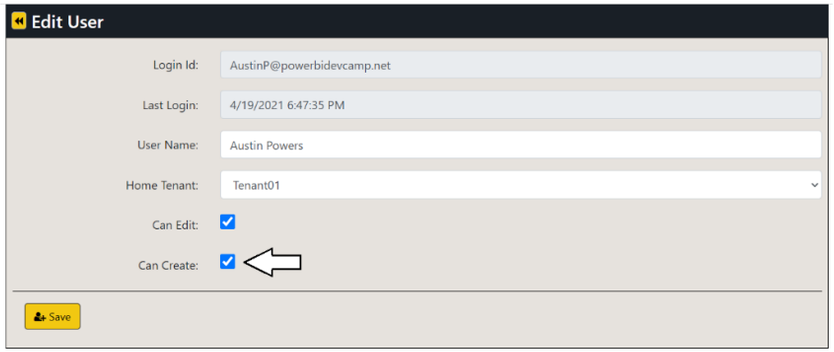
- Test user experience when user has edit permissions
  - User can move report into edit mode
  - Once in edit mode, you can customize report and save changes

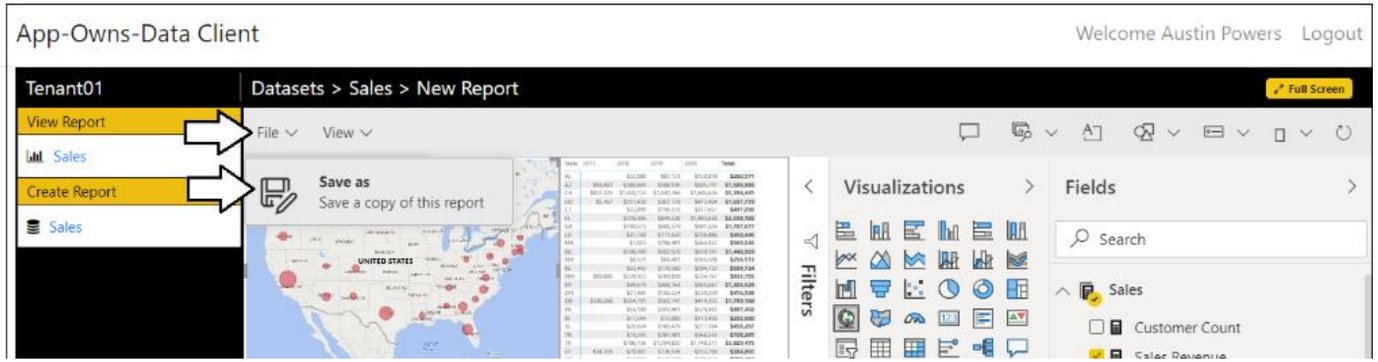


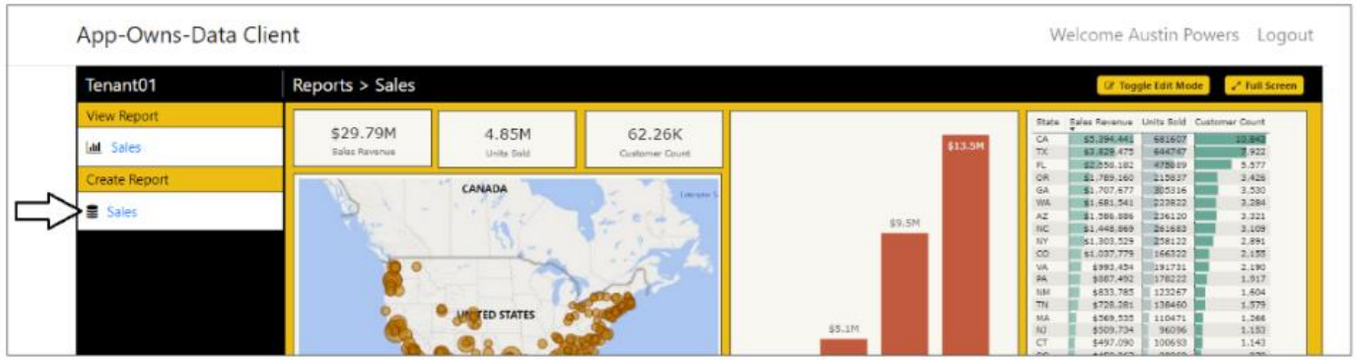


Create New Content using AppOwnsDataClient

- 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 in Create Report section to create new report







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

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

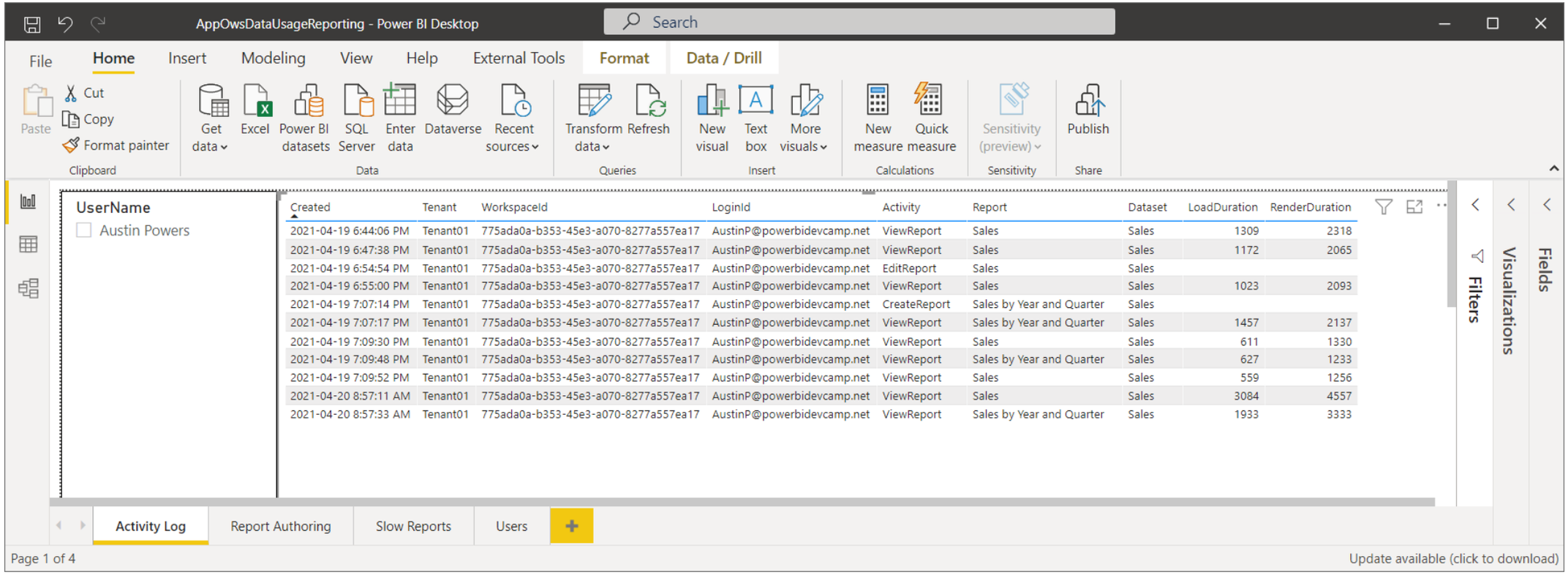
100 %

Results | Messages

	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

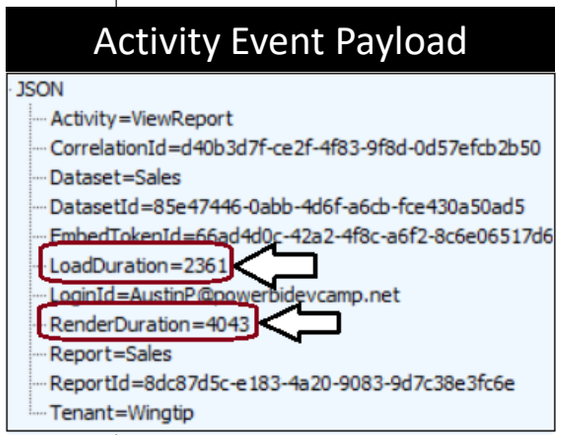
- App-Owns-Data Starter Kit includes report template **AppOwsDataUsageReporting.pbix**
  - **AppOwsDataUsageReporting.pbix** imports data from **AppOwnsDataDB**
  - Allows for monitoring and analysis of usage data



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 fo 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;
  }
});
```



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