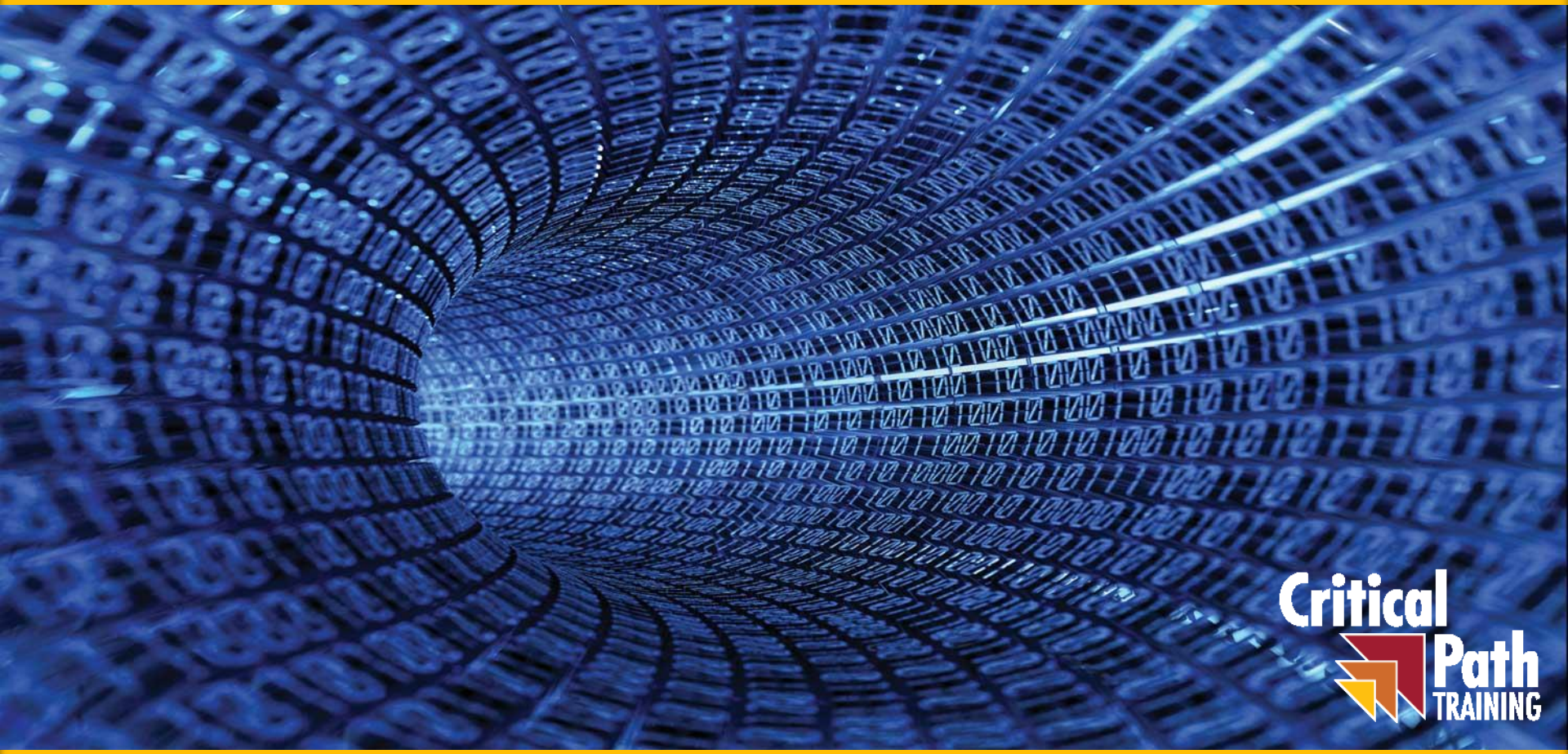
Introduction to Developing Power BI Embedding

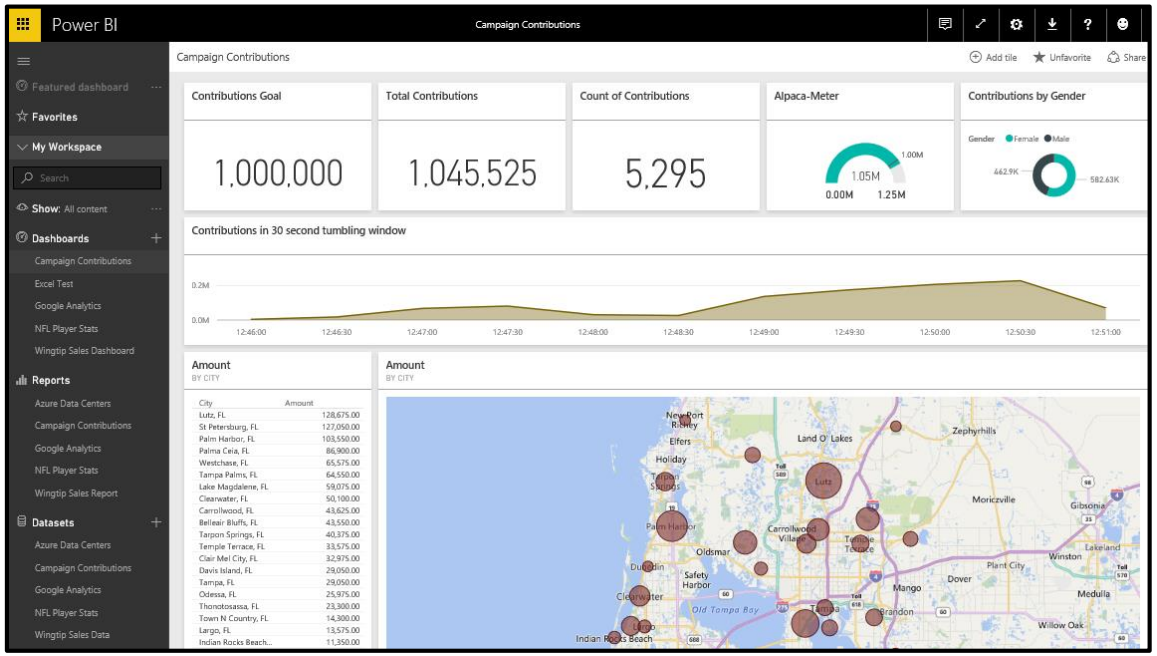


Agenda

- Power BI Embedding Fundamentals
- Understanding Dedicated Capacities
- Setting Up a Development Environment
- Developer Quickstart into Power BI Embedding

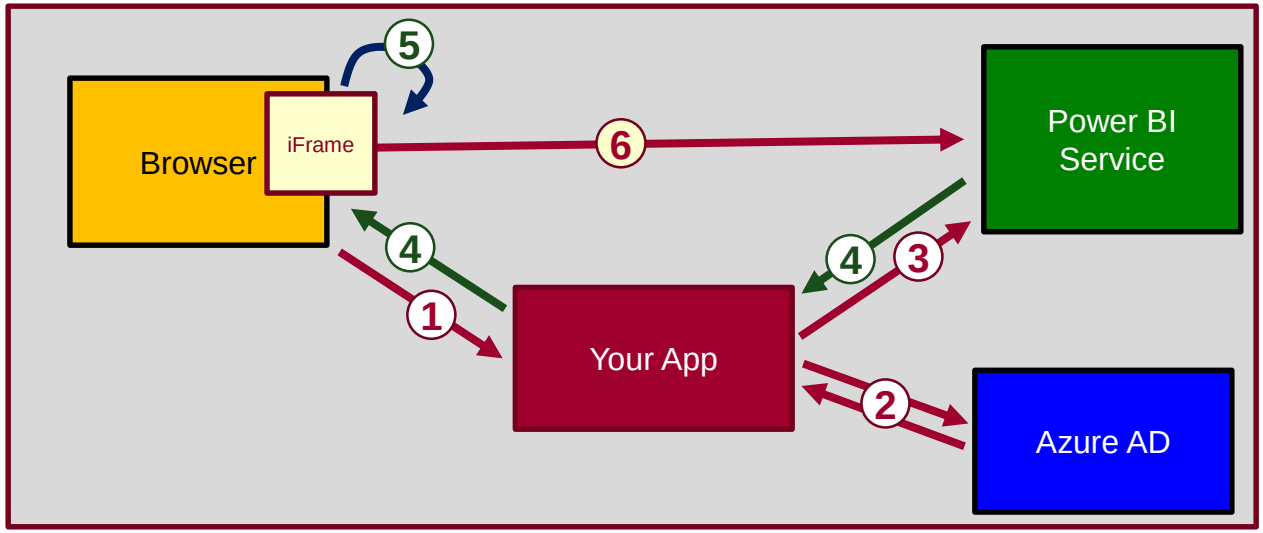
The Power BI Service – Who Is It For?

- Provides SaaS service used by web and mobile users
  - Power BI portal accessible to browsers at https://app.powerbi.com
  - Power BI mobile accessible to users on mobile phones & devices
- Provides PaaS service used by software developers
  - Power BI Service API accessible at https://api.powerbi.com



Power BI Embedding – The Big Picture

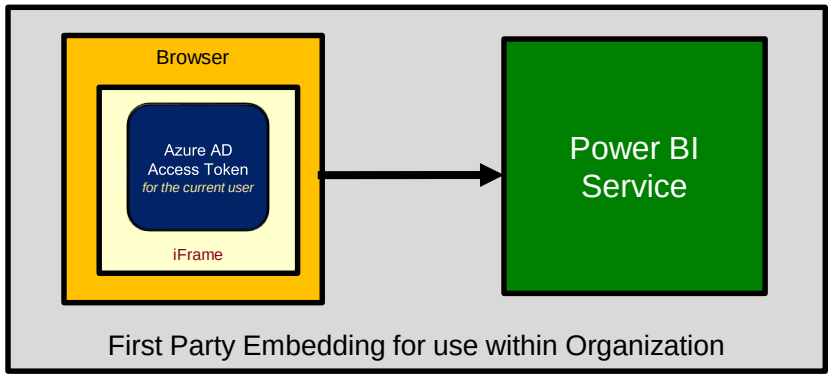
- User launches your app using a browser
- App authenticates with Azure Active Directory and obtains access token
- App uses access token to call to Power BI Service API
- App retrieves data for embedded resource and passes it to browser.
- Client-side code uses Power BI JavaScript API to create embedded resource
- Embedded resource session created between browser and Power BI service



First Party Embedding vs Third Party Embedding

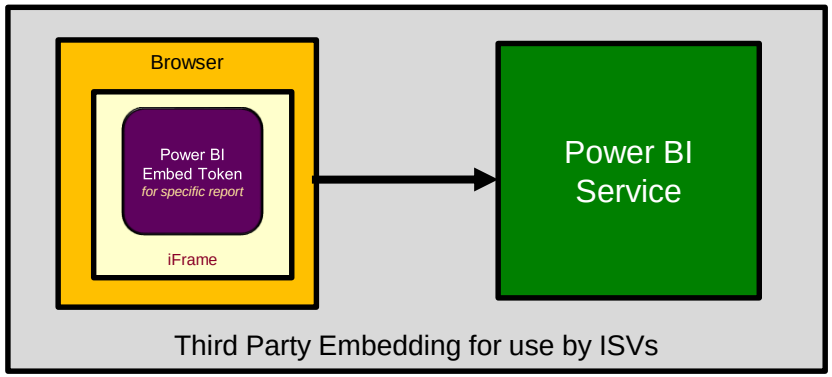
First Party Embedding

- Known as **User-Owns-Data** Model
- All users require a Power BI license
- Useful in corporate environments
- App authenticates as current user
- Your code runs with user's permissions
- User's access token passed to browser



Third Party Embedding

- Known as **App-Owns-Data** Model
- No users require Power BI license
- Useful for commercial applications
- App authenticates with master user account
- Your code runs with admin permissions
- Embed token passed to browser

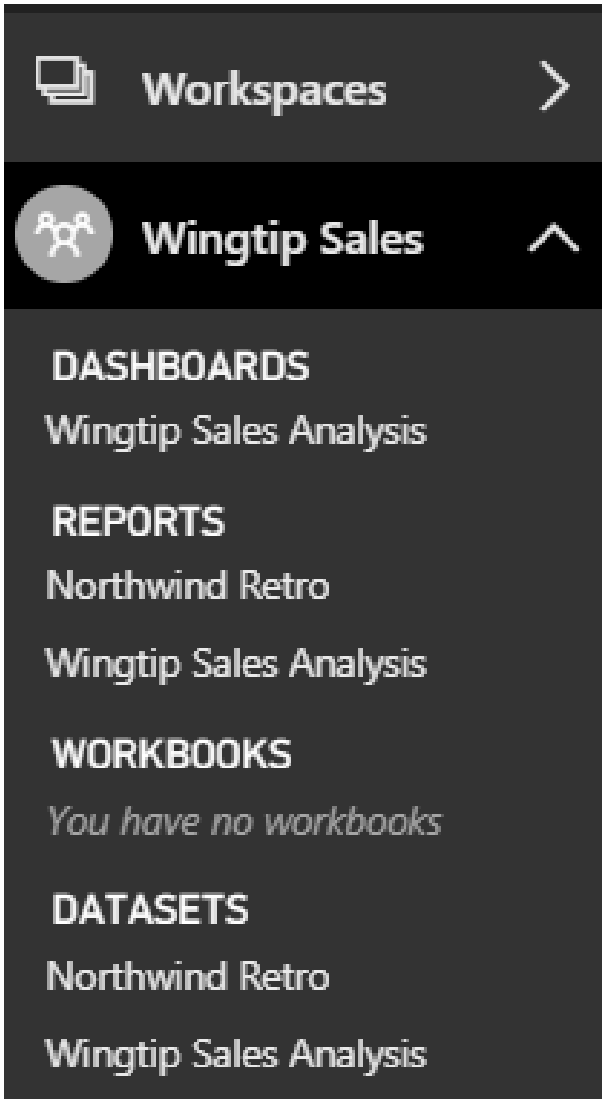


Embeddable Resources

- Reports
- Dashboards
- Dashboard Tiles
- New Reports
- Q&A Experience
- Visuals in custom layout

Embeddable Resources

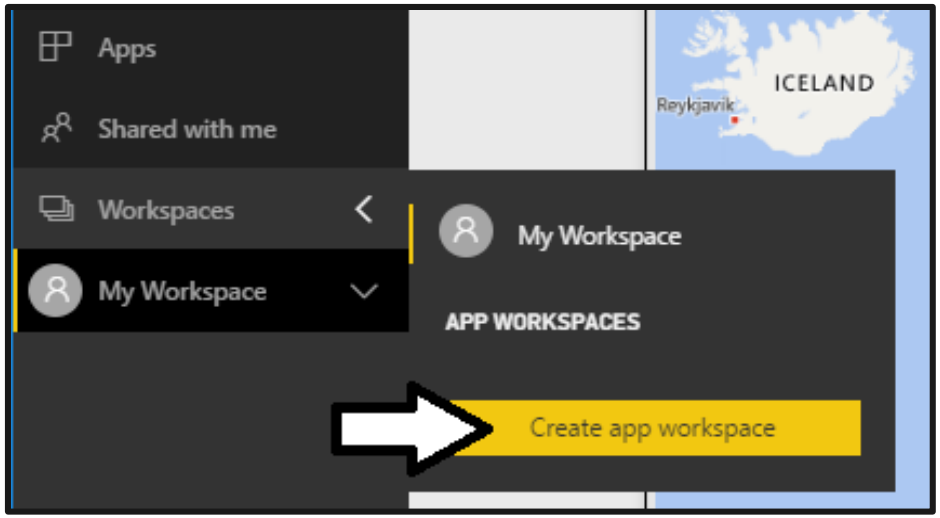
Central Power BI Concepts

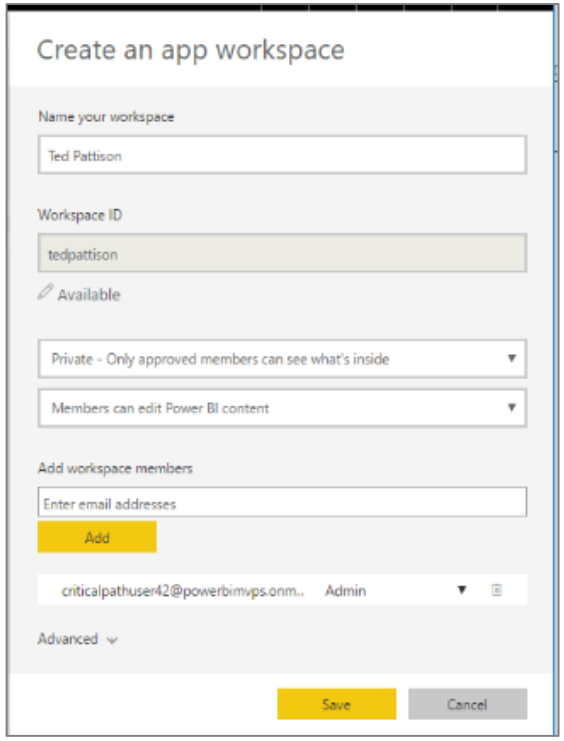


- Workspace
  - Secure container for publishing content
  - Every licensed user gets a personal workspace
  - App workspaces created for custom solutions
- Dashboard
  - Consolidated view into reports and datasets
  - Custom solution entry point for mobile users
- Report
  - Collection of pages with tables & visualizations
  - Provides interactive control of filtering
- Dataset
  - Data model containing one or more tables
  - Can be very simple or very complex

Creating App Workspaces

- Power BI content published in app workspaces
  - Power BI Pro license required to author content in app workspace
  - Datasets & reports created by publishing PBIX project files
  - Dashboards must be created by hand



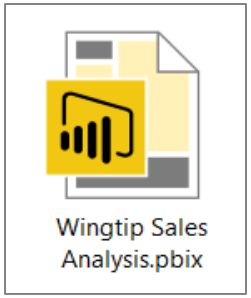


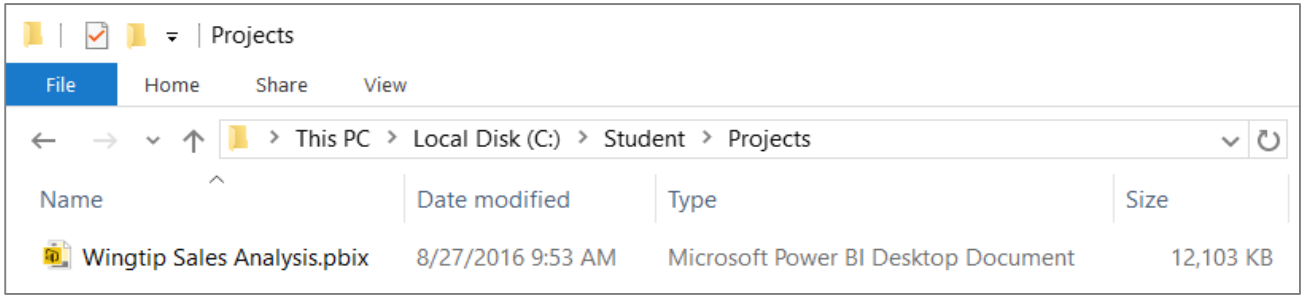
Power BI App Workspaces - V2

- Power BI is transition between workspace models
  - V2 app workspaces are currently in preview
  - Microsoft will soon force the migration from V1 to V2
- V1 workspaces built on top of Office 365 groups
  - Largely managed in Office 365 admin not Power BI
- V2 workspaces independent of Office 365 groups
  - Fully managed within Power BI admin portal
  - Provides more flexibility in assigning members

Power BI Desktop Projects and PBIX Files

- Power BI Desktop projects saved using PBIX files
  - PBIX file contains data source definitions
  - PBIX file contains query definitions
  - PBIX file contains data imported from queries
  - PBIX file contains exactly one data model definition
  - PBIX file contains exactly one report
  - PBIX file never contains data source credentials

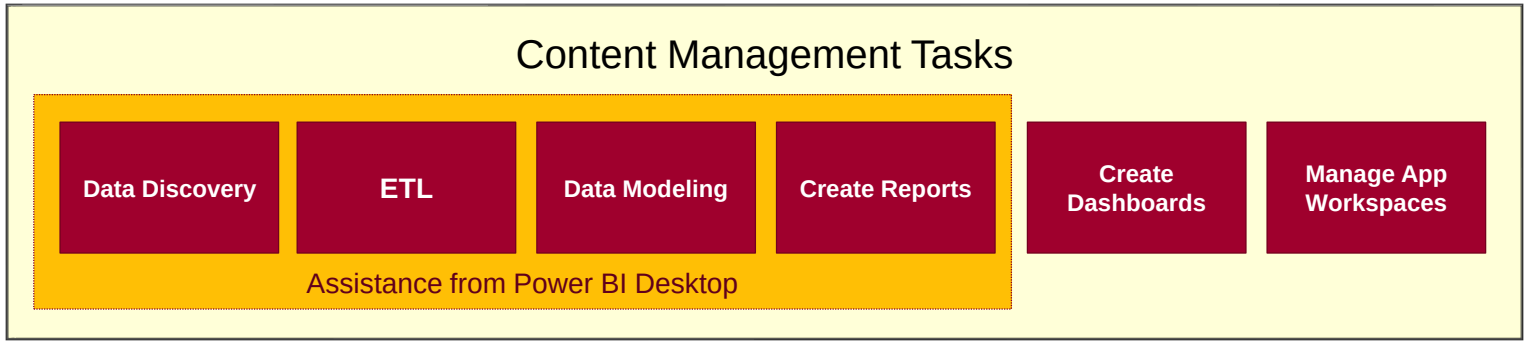




- PBIX files can be tracked in source control (e.g. github.com)
  - PBIX files is a versioned unit of work with 1 dataset and 1 report
  - Some Power BI resources (e.g. Dashboard) have no backing file support

Working with Power BI Desktop

- Power BI Desktop focuses on first four phases
  - Query features for Data Discovery and ETL
  - Data modeling features and DAX language for building data model
  - Report design features for building interactive report
  - No support for building dashboards
  - No support for packaging an entire solution



Division of Labor with Power BI Embedding

- Content Management Team
  - Build Power BI Desktop projects
  - Publish reports & datasets via PBIX files
  - Create dashboards in PBI Service
  - Publish App Workspaces as Apps
  - Monitor Power BI environment

- Application Development Team
  - Develop web apps with PBI embedding
  - Authenticate with Azure Active Directory
  - Retrieve data using Power BI Service API
  - Embed resources using Power BI JavaScript API
  - This team sees PBIX files as black boxes

Capacities and Workspace Associations

- This is

Dedicated Capacities

- Power BI workspaces run in one of two possible environments
  - Shared capacity
  - Dedicated capacity
- Dedicated capacity required for third party embedding
  - You pay Microsoft capacity-based fee for processors cores and RAM
  - No need to pay Microsoft for user licensing
- Dedicated capacity can optionally be used for first party embedding
  - Allows users to run with free license instead of Power BI Pro license ($10/month)
- Dedicated capacities come in two flavors
  - Power BI Premium capacities purchased through Office 365
  - Power BI Embedded capacities purchased through Azure SKU

Power BI Licensing Though Office 365

User-based licensing

Power BI (free)

$0.00 user/month

A cloud-based business analytics service that enables anyone to visualize and analyze data with greater speed, efficiency, and understanding. It ...

Office 2016 desktop & mobile apps
Not included

Office 365 services

Power BI Pro

$9.99 user/month

A cloud-based business analytics service that enables anyone to visualize and analyze data with greater speed, efficiency, and understanding. Power ...

Office 2016 desktop & mobile apps
Not included

Office 365 services

Office 365 Enterprise E5

$35.00 user/month

The Office suite, plus email, instant messaging, HD video conferencing, 1 TB personal file storage and sharing, and advanced security, analytics and PSTN ...

Office 2016 desktop & mobile apps

Office 365 services

Capacity-based Licensing

Power BI Premium P1

$4,995.00 instance/month

Power BI capacity dedicated to your organization, unlocking unlimited content distribution and dependable performance. P1 offers 8 virtual core ...

Office 2016 desktop & mobile apps
Not included

Office 365 services

Power BI Premium P2

$9,995.00 instance/month

Power BI capacity dedicated to your organization, unlocking unlimited content distribution and dependable performance. P1 offers 16 virtual cor ...

Office 2016 desktop & mobile apps
Not included

Office 365 services

Power BI Premium P3

$19,995.00 instance/month

Power BI capacity dedicated to your organization, unlocking unlimited content distribution and dependable performance. P1 offers 32 virtual cor ...

Office 2016 desktop & mobile apps
Not included

Office 365 services

Power BI Premium P4

$39,995.00 instance/month

Power BI capacity dedicated to your organization, unlocking unlimited content distribution and dependable performance. P4 offers 64 virtual cor ...

Office 2016 desktop & mobile apps
Not included

Office 365 services

Power BI Premium P5

$79,995.00 instance/month

Power BI capacity dedicated to your organization, unlocking unlimited content distribution and dependable performance. P5 offers 128 virtual ...

Office 2016 desktop & mobile apps
Not included

Office 365 services

Power BI Premium EM3 (Month to Month)

$2,495.00 instance/month

Embed Power BI content in your custom application, powered by 4 virtual cores of dedicated capacity. Some Premium features are disabled ...

Office 2016 desktop & mobile apps
Not included

Office 365 services

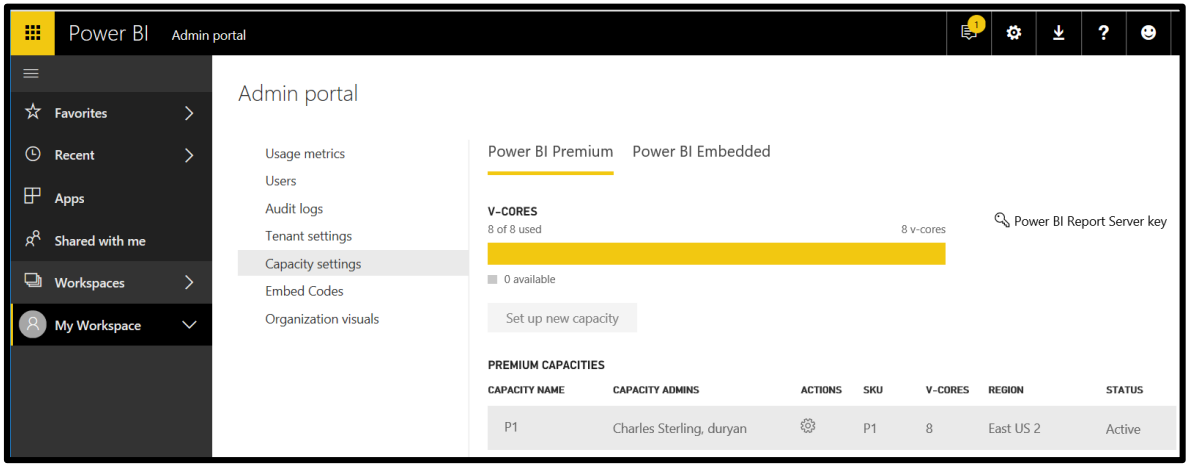
Premium Capacity Nodes

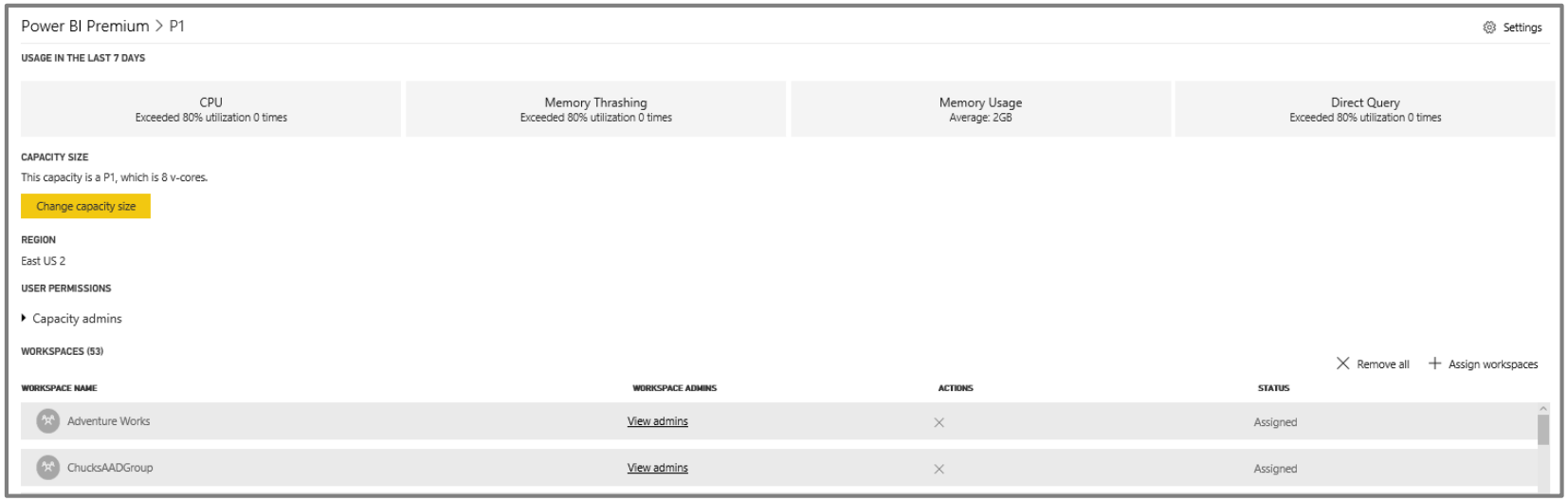
- Power BI Premium Purchased using Nodes
  - Node type defines v-core and RAM capabilities
  - P nodes for Power BI portal deployments and embedded deployments
  - EM nodes only used for embedded deployments

Capacity Node	Total cores	Backend Cores	Frontend Cores	Direct Query Limits	Page renders/hour
EM1	1 v-cores	.5 cores, 3GB RAM	.5 cores		1-300
EM2	2 v-cores	1 core, 5GB RAM	1 core		301-600
EM3	4 v-cores	2 cores, 10GB RAM	2 cores		601-1,200
P1	8 v-cores	4 cores, 25GB RAM	4 cores	30 per second	1,201-2,400
P2	16 v-cores	8 cores, 50GB RAM	8 cores	60 per second	2,401-4,800
P3	32 v-cores	16 cores, 100GB RAM	16 cores	120 per second	4,801-9600
P4	64 v-cores	32 cores, 200GB RAM	32 cores	240 per second	9601-19,200
P5	128 v-cores	64 cores 400GB	64	480 per second	19,201- 38,400

Managing Premium Capacities

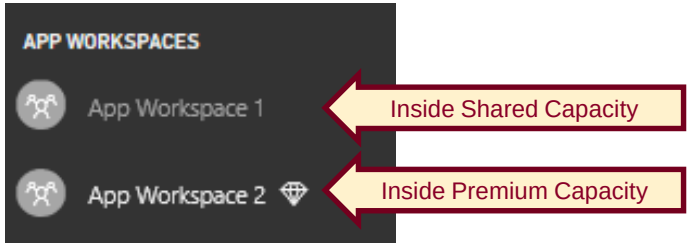
- Dedicated capacities managed in Power BI Admin portal



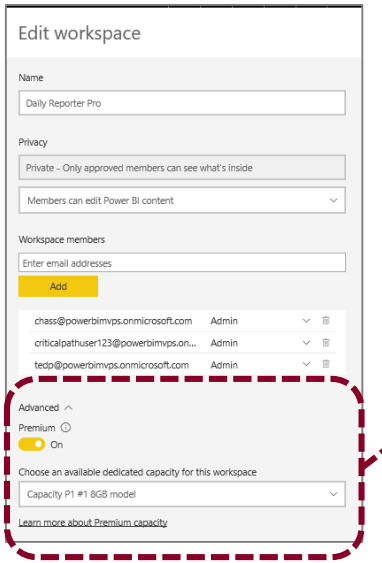


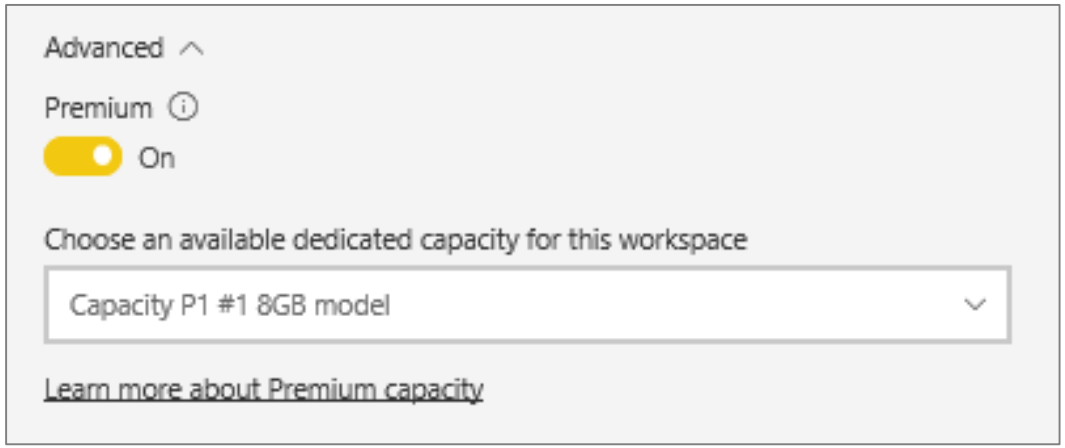
Associating App Workspaces with a Capacity

- App Workspace in Premium Capacity has diamond icon



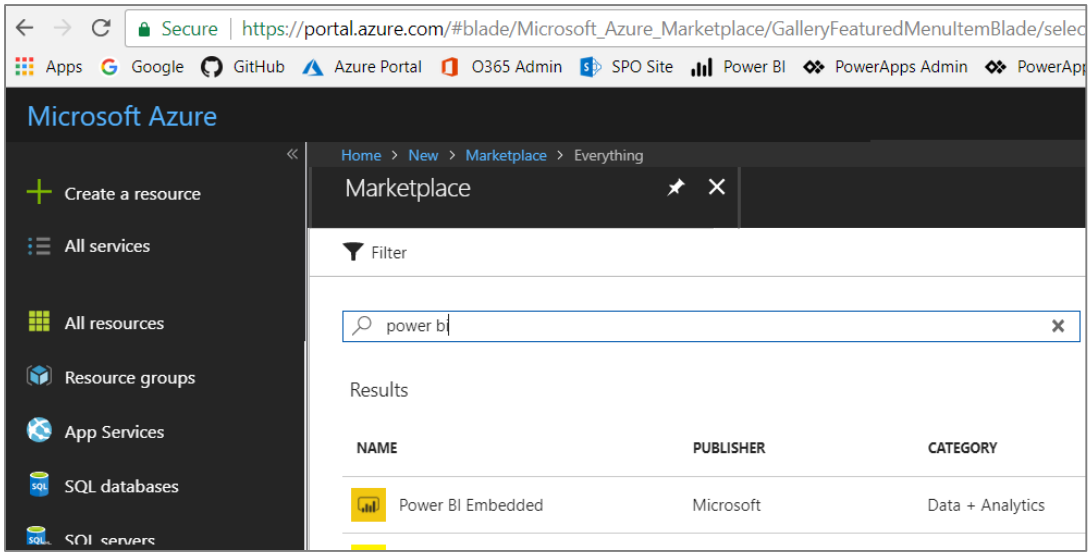
- App workspace moved into Premium capacity in Advanced settings

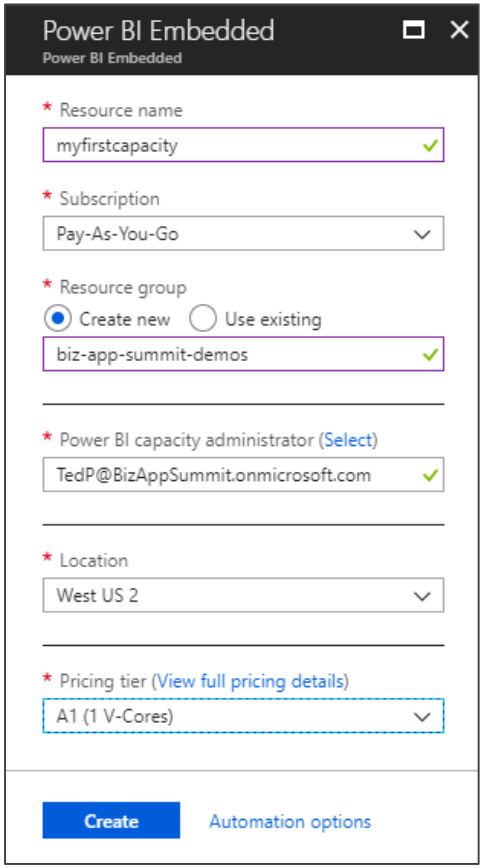




Creating the Power BI Embedded Service

- Power BI Embedded in an Azure on-demand service
  - Must be created in same location as Power BI Service for tenant
  - Can be created manually through the Azure portal
  - Can be created in automated fashion using PowerShell
  - Requires an Azure subscription ***in same tenant***





Embedded Capacity Pricing Tiers

Home > myfirstcapacity

myfirstcapacity
Power BI Embedded

Search (Ctrl+/)

- Overview
- Activity log
- Access control (IAM)
- Tags
- Diagnose and solve problems

SCALE

- Pricing tier

SETTINGS

- Quick Start
- Power BI capacity administrators
- Properties
- Locks
- Automation script

MONITORING

- Diagnostics logs
- Metrics
- Alerts

Pause | Move | Delete

Essentials

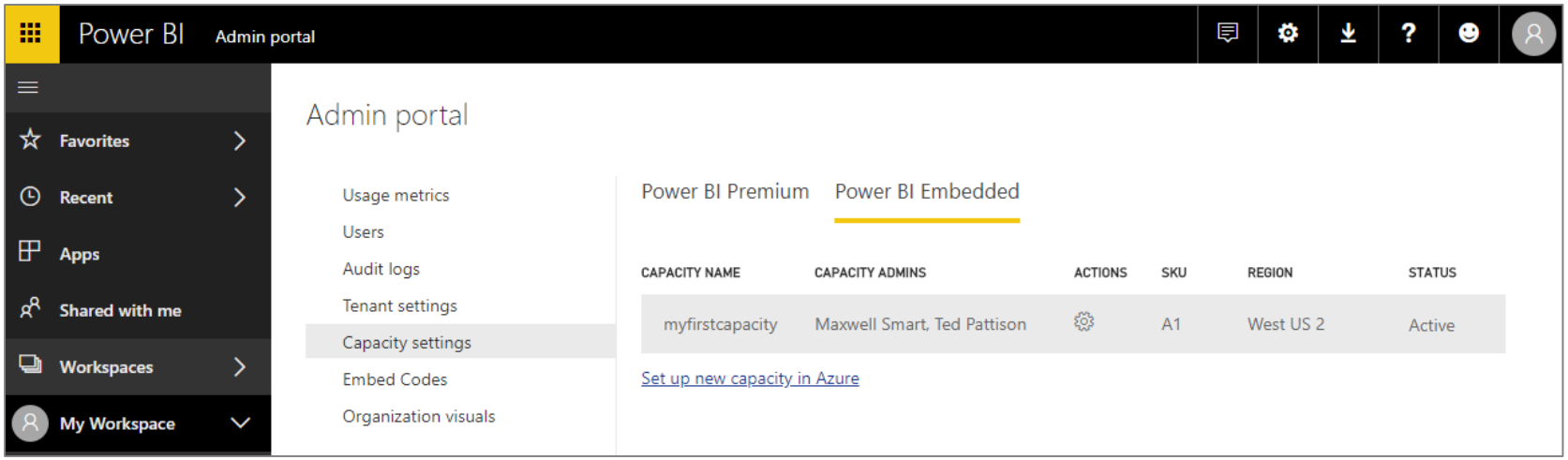
Resource group (change)
biz-app-summit-demos
Status
Active
Location
West US 2
Subscription name (change)
Pay-As-You-Go
Subscription ID
63bf3c07-a58b-4940-ab25-a6a66dda26dc

Resource name
myfirstcapacity
Pricing tier
A1

A1	A2	A3 ★	A4	A5	A6
1 V-Cores	2 V-Cores	4 V-Cores	8 V-Cores	16 V-Cores	32 V-Cores
Up to 3 GB Cache	Up to 5 GB Cache	Up to 10 GB Cache	Up to 25 GB Cache	Up to 50 GB Cache	Up to 100 GB Cache
Shared infrastructure	Shared infrastructure	Dedicated service	Dedicated service	Dedicated service	Dedicated service
750.03 USD/MONTH (ESTIMATED)	1,494.03 USD/MONTH (ESTIMATED)	2,994.00 USD/MONTH (ESTIMATED)	5,994.04 USD/MONTH (ESTIMATED)	11,994.02 USD/MONTH (ESTIMATED)	23,994.45 USD/MONTH (ESTIMATED)

Managing Power BI Embedded Capacities

- Managing Power BI Embedded capacity done Azure portal
  - Use Azure portal to assign administrators & change pricing tier to scale
- Managing Power BI Embedded capacity in Power BI Admin portal
  - Power BI Admin portal to assign workspaces to capacities



PBI Capacity SKU Decoder Ring

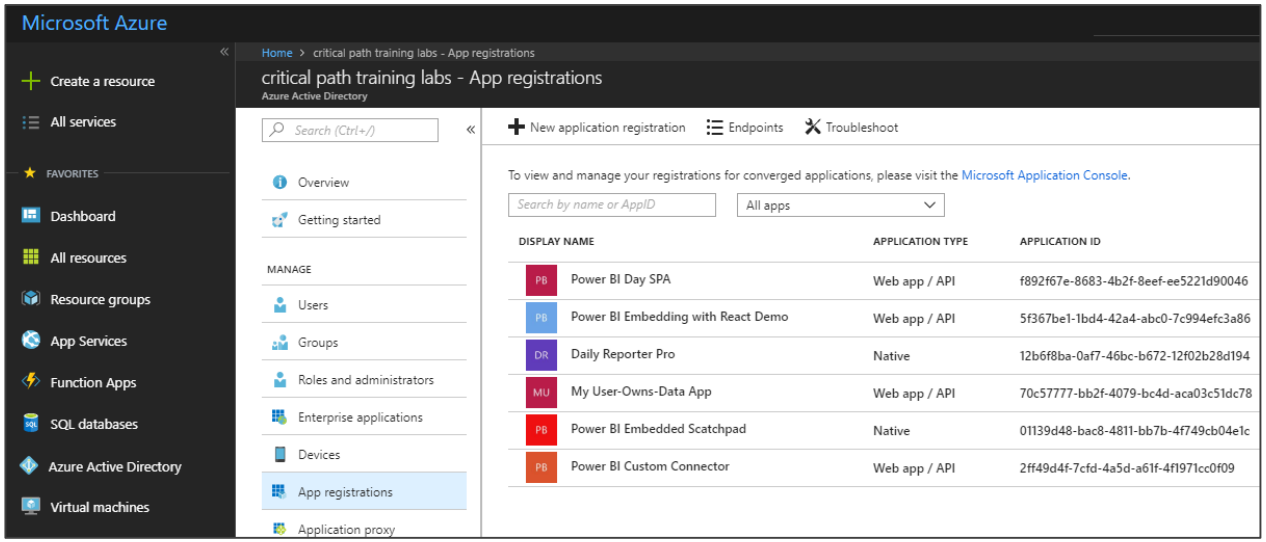
	P SKU	EM SKU	A SKU
Purchased through…	Office 365	Office 365	Azure
Supports Power BI embedding in custom applications	Yes	Yes	Yes
Supports 3rd Party Embedding in custom applications	Yes	Yes	Yes
Supports 1st Party Embedding in custom applications	Yes	Yes	Yes
Supports free users accessing content in Power BI portal	Yes	No	No
Supports free users accessing content in Power BI Mobile	Yes	No	No
Billing cycle	Monthly	Monthly	Hourly
Commitment	Monthly	Monthly/Yearly	None
Turn it off when your not using it	No	No	Yes

Agenda

- Power BI Embedding Fundamentals
- Managing Content using App Workspaces
- Understanding Dedicated Capacities
- **Using the Onboarding Experience Tool**
- Programming with Power BI Service API
- Embedding with the Power BI JavaScript API

Understanding Authentication with Azure AD

- Requirements of calling the Power BI Service API
  - App must be registered with Azure AD with Application ID (aka Client ID)
  - App must be created with app type of as **Web app / API** or **Native**
  - App registration must be configured with required permissions
  - App must implement authentication flow to authenticate user and obtain access token



Demo Time

Getting Started using the Power BI Embedding Onboarding Experience

Agenda

- Power BI Embedding Fundamentals
- Managing Content using App Workspaces
- Understanding Dedicated Capacities
- Using the Onboarding Experience Tool
- **Programming with Power BI Service API**
- Embedding with the Power BI JavaScript API

The Power BI Service API

- Microsoft.PowerBI.Api
  - Microsoft.PowerBI.Api.V1
  - Microsoft.PowerBI.Api.V1.Models
  - Microsoft.PowerBI.Api.V2
    - Dashboards
    - DashboardsExtensions
    - Datasets
    - DatasetsExtensions
    - Gateways
    - GatewaysExtensions
    - Groups
    - GroupsExtensions
    - IDashboards
    - IDatasets
    - IGateways
    - IGroups
    - IImports
    - Imports
    - ImportsExtensions
    - IPowerBIClient
    - IReports
    - ITiles
    - PowerBIClient
    - Reports
    - ReportsExtensions
    - Tiles
    - TilesExtensions

- Microsoft.PowerBI.Api
  - Microsoft.PowerBI.Api.V1
  - Microsoft.PowerBI.Api.V1.Models
  - Microsoft.PowerBI.Api.V2
  - Microsoft.PowerBI.Api.V2.Models
    - BasicCredentials
    - BindToGatewayRequest
    - CloneReportRequest
    - Column
    - ConnectionDetails
    - CredentialDetails
    - Dashboard
    - Dataset
    - DatasetMode
    - Datasource
    - EmbedToken
    - Gateway
    - GatewayDatasource
    - GatewayPublicKey
    - GenerateTokenRequest
    - Group
    - GroupCreationRequest
    - GroupUser
    - GroupUserAccessRight
    - Import
    - ImportConflictHandlerMode
    - ImportInfo
    - MemberAdminAccessRight
    - ODataResponseListDashboard
    - ODataResponseListDataset
    - ODataResponseListDatasource
    - ODataResponseListGateway
    - ODataResponseListGatewayDatasource
    - ODataResponseListGroup
    - ODataResponseListGroupUserAccessRight
    - ODataResponseListImport
    - ODataResponseListRefresh
    - ODataResponseListReport
    - ODataResponseListTable
    - ODataResponseListTile
    - ODataResponseListUserAccessRight
    - PublishDatasourceToGatewayRequest
    - RebindReportRequest
    - Refresh
    - Report
    - Row
    - Table
    - Tile
    - TokenAccessLevel
    - UpdateDatasourceRequest
    - UserAccessRight
    - UserAccessRightEnum

App Configuration Data for AAD Authentication

```xml
<configuration>
  <appSettings>

    <add key="clientId" value="23f6d66f-9a9a-4dba-9b7c-ff8aedadb831" />

    <add key="pbiUserName" value="pbiemasteruser@powerbimvps.onMicrosoft.com" />

    <add key="pbiUserPassword" value="Pa$$word!" />

    <add key="appWorkspaceId" value="4baab6c0-87c5-4a2a-a73e-1f97adcc6bdb" />
```

```csharp
public class PbiEmbeddingManager {

  #region "AAD Authentication Constants"

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

  static string clientId = ConfigurationManager.AppSettings["clientId"];
  static string appWorkspaceId = ConfigurationManager.AppSettings["appWorkspaceId"];
  static string pbiUserName = ConfigurationManager.AppSettings["pbiUserName"];
  static string pbiUserPassword = ConfigurationManager.AppSettings["pbiUserPassword"];

  #endregion
```

Getting an Access Token for the Master User

```
static string GetAccessToken() {
  AuthenticationContext authContext = new AuthenticationContext(aadAuthorizationEndpoint);
  var userCredentials = new UserPasswordCredential(pbiUserName, pbiUserPassword);

  // this call will fail if permission consent has not be granted to master user account
  string aadAccessToken =
    authContext.AcquireTokenAsync(resourceUriPowerBi, clientId, userCredentials).Result.AccessToken;

  // return Azure AD access token for master user account
  return aadAccessToken;
}
```

Initializing an Instance of PowerBIClient

- PowerBIClient object serves as top-level object
  - Used to execute calls against Power BI Service
  - Initialized with function to retrieve AAD access token

```
static string GetAccessToken() ...

static PowerBIClient GetPowerBiClient() {
  var tokenCredentials = new TokenCredentials(GetAccessToken(), "Bearer");
  return new PowerBIClient(new Uri(urlPowerBiRestApiRoot), tokenCredentials);
}

static void Main() {
  PowerBIClient pbiClient = GetPowerBiClient();
  var reports = pbiClient.Reports.GetReports().Value;
  foreach (var report in reports) {
    Console.WriteLine(report.Name);
  }
}
```

Report and Dataset Info

```
// data required for embedding a report
class ReportEmbeddingData {
  public string reportId;
  public string reportName;
  public string embedUrl;
  public string accessToken;
}

// data required for embedding a dashboard
class DashboardEmbeddingData {
  public string dashboardId;
  public string dashboardName;
  public string embedUrl;
  public string accessToken;
}

// data required for embedding a dashboard
class DashboardTileEmbeddingData {
  public string dashboardId;
  public string TileId;
  public string TileTitle;
  public string embedUrl;
  public string accessToken;
}
```

```
// data required for embedding a new report
class NewReportEmbeddingData {
  public string workspaceId;
  public string datasetId;
  public string embedUrl;
  public string accessToken;
}

// data required for embedding QnA experience
class QnaEmbeddingData {
  public string datasetId;
  public string embedUrl;
  public string accessToken;
}
```

Getting the Data for First Party Report Embedding

```
public static ReportEmbeddingData GetReportEmbeddingDataFirstParty() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var report = pbiClient.Reports.GetReportInGroup(workspaceId, reportId);
  var embedUrl = report.EmbedUrl;
  var reportName = report.Name;
  var accessToken = GetAccessToken();

  return new ReportEmbeddingData {
    reportId = reportId,
    reportName = reportName,
    embedUrl = embedUrl,
    accessToken = accessToken
  };

}
```

Embed Tokens

- You can embed reports using master user AAD token, but…
  - You might want embed resource using more restricted tokens
  - You might want stay within the bounds of Power BI licensing terms
- You generate embed tokens with the Power BI Service API
  - Each embed token created for one specific resource
  - Embed token provides restrictions on whether user can view or edit
  - Embed token can only be generated inside dedicated capacity *(semi-enforced)*
  - Embed token can be generated to support row-level security (RLS)

```
Report report = reports.Where(r => r.Id == reportId).First();
var generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "edit");
var token = client.Reports.GenerateTokenInGroupAsync(appWorkspaceId,
                                                     report.Id,
                                                     generateTokenRequestParameters).Result;
```

Getting the Data for Third Party Report Embedding

```
public static ReportEmbeddingData GetReportEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var report = pbiClient.Reports.GetReportInGroup(workspaceId, reportId);
  var embedUrl = report.EmbedUrl;
  var reportName = report.Name;

  // create token request object
  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");

  // call to Power BI Service API and pass GenerateTokenRequest object to generate embed token
  string embedToken = pbiClient.Reports.GenerateTokenInGroup(workspaceId,
                                                             report.Id,
                                                             generateTokenRequestParameters).Token;

  return new ReportEmbeddingData {
    reportId = reportId,
    reportName = reportName,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

Getting the Data for Dashboard Embedding

```
public static DashboardEmbeddingData GetDashboardEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var dashboard = pbiClient.Dashboards.GetDashboardInGroup(workspaceId, dashboardId);
  var embedUrl = dashboard.EmbedUrl;
  var dashboardDisplayName = dashboard.DisplayName;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");
  string embedToken = pbiClient.Dashboards.GenerateTokenInGroup(workspaceId,
                                                                 dashboardId,
                                                                 generateTokenRequestParameters).Token;

  return new DashboardEmbeddingData {
    dashboardId = dashboardId,
    dashboardName = dashboardDisplayName,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

Getting the Data for Dashboard Tile Embedding

```
public static DashboardTileEmbeddingData GetDashboardTileEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var tiles = pbiClient.Dashboards.GetTilesInGroup(workspaceId, dashboardId).Value;

  // retrieve first tile in tiles connection
  var tile = tiles[0];
  var tileId = tile.Id;
  var tileTitle = tile.Title;
  var embedUrl = tile.EmbedUrl;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");
  string embedToken = pbiClient.Tiles.GenerateTokenInGroup(workspaceId,
                                                            dashboardId,
                                                            tileId,
                                                            generateTokenRequestParameters).Token;

  return new DashboardTileEmbeddingData {
    dashboardId = dashboardId,
    TileId = tileId,
    TileTitle = tileTitle,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

Getting the Data for New Report Embedding

```
public static NewReportEmbeddingData GetNewReportEmbeddingData() {

  string embedUrl = "https://app.powerbi.com/reportEmbed?groupId=" + workspaceId;

  PowerBIClient pbiClient = GetPowerBiClient();

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "create",
                                                                                  datasetId: datasetId);

  string embedToken = pbiClient.Reports.GenerateTokenForCreateInGroup(workspaceId,
                                                                      generateTokenRequestParameters).Token;

  return new NewReportEmbeddingData {
    workspaceId = workspaceId,
    datasetId = datasetId,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

Getting the Data for Q&A Experience Embedding

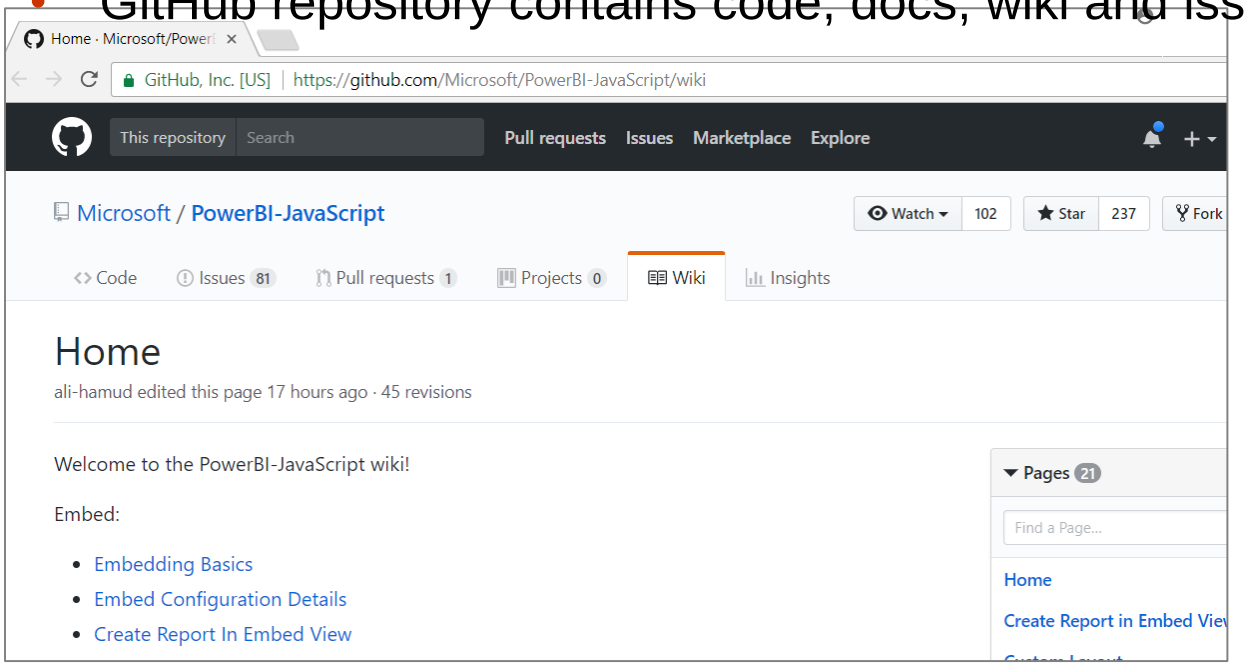
```
public static QnaEmbeddingData GetQnaEmbeddingData() {

  PowerBIClient pbiClient = GetPowerBiClient();

  var dataset = pbiClient.Datasets.GetDatasetByIdInGroup(workspaceId, datasetId);

  string embedUrl = "https://app.powerbi.com/qnaEmbed?groupId=" + workspaceId;
  string datasetID = dataset.Id;

  GenerateTokenRequest generateTokenRequestParameters = new GenerateTokenRequest(accessLevel: "view");

  string embedToken = pbiClient.Datasets.GenerateTokenInGroup(workspaceId,
                                                              dataset.Id,
                                                              generateTokenRequestParameters).Token;

  return new QnaEmbeddingData {
    datasetId = datasetId,
    embedUrl = embedUrl,
    accessToken = embedToken
  };

}
```

Agenda

- Power BI Embedding Fundamentals
- Managing Content using App Workspaces
- Understanding Dedicated Capacities
- Using the Onboarding Experience Tool
- Programming with Power BI Service API
- **Embedding with the Power BI JavaScript API**

Power BI JavaScript API (powerbi.js)

- Power BI JavaScript API used to embed resources in browser
  - Maintained in GitHub at https://github.com/Microsoft/PowerBI-JavaScript/wiki
  - GitHub repository contains code, docs, wiki and issues list

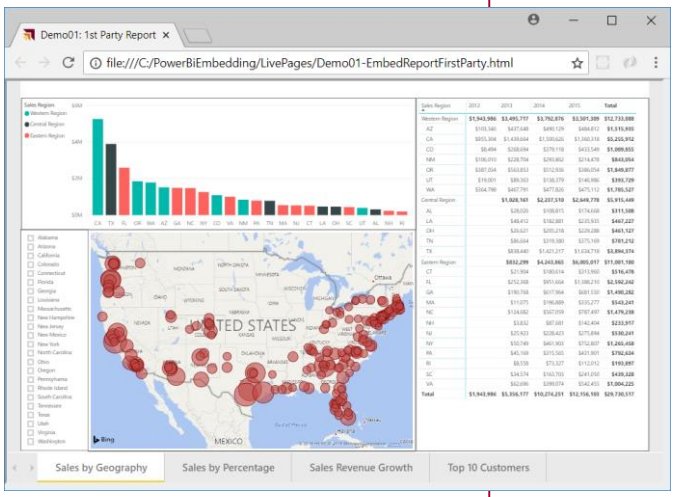


Hello World with Power BI Embedding

- **powerbi.js** library provides **powerbi** as top-level service object
  - You call **powerbi.embed** and pass **configuration** object with access token t
  - **models** object available to supply configuration settings
  - **configuration** object sets **tokenType** to either **models.TokenType.Embed** or **models.TokenType.Aad**

```
// data required for embedding Power BI report
var embedReportId = "ba274ba0-93be-4e53-af65-fdc8a559c557";
var embedUrl = "https://app.powerbi.com/reportEmbed?reportId=ba274ba0-93be-4e53-af65-fdc8a559c557&groupId=7f4
var accessToken = "eyJ0eXAiOiJKV1QiLCJhbGciOiJSUzI1NiIsIng1dCI6IlRpb0d5d3dsaHZkRmJYWjgxM1dwUGF5OUFsVSIsImtpZC

// Get models object to access enums for embed configuration
var models = window['powerbi-client'].models;

// create embed configuration object
var config = {
  type: 'report',
  id: embedReportId,
  embedUrl: embedUrl,
  accessToken: accessToken,
  tokenType: models.TokenType.Aad
};

// Get a reference to the embedded report HTML element
var reportContainer = document.getElementById('embedContainer');

// Embed the report and display it within the div container.
var report = powerbi.embed(reportContainer, config);
```

First party embedding



Embedded Report Configuration Options

- **permissions** determines what permissions are given to user on resource
- **viewMode** determines when report opens in read-only view or edit view
- **pageView** determines how reports scales to fit embed container element

```
var models = window['powerbi-client'].models;

var config: embed.IEmbedConfiguration = {
  type: 'report',
  id: reportId,
  embedUrl: embedUrl,
  accessToken: accessToken,
  tokenType: models.TokenType.Embed,
  permissions: models.Permissions.All,
  viewMode: models.ViewMode.Edit,
  pageView: "fitToWidth",
  settings: {
    filterPaneEnabled: false,
    navContentPaneEnabled: false
  }
};
```

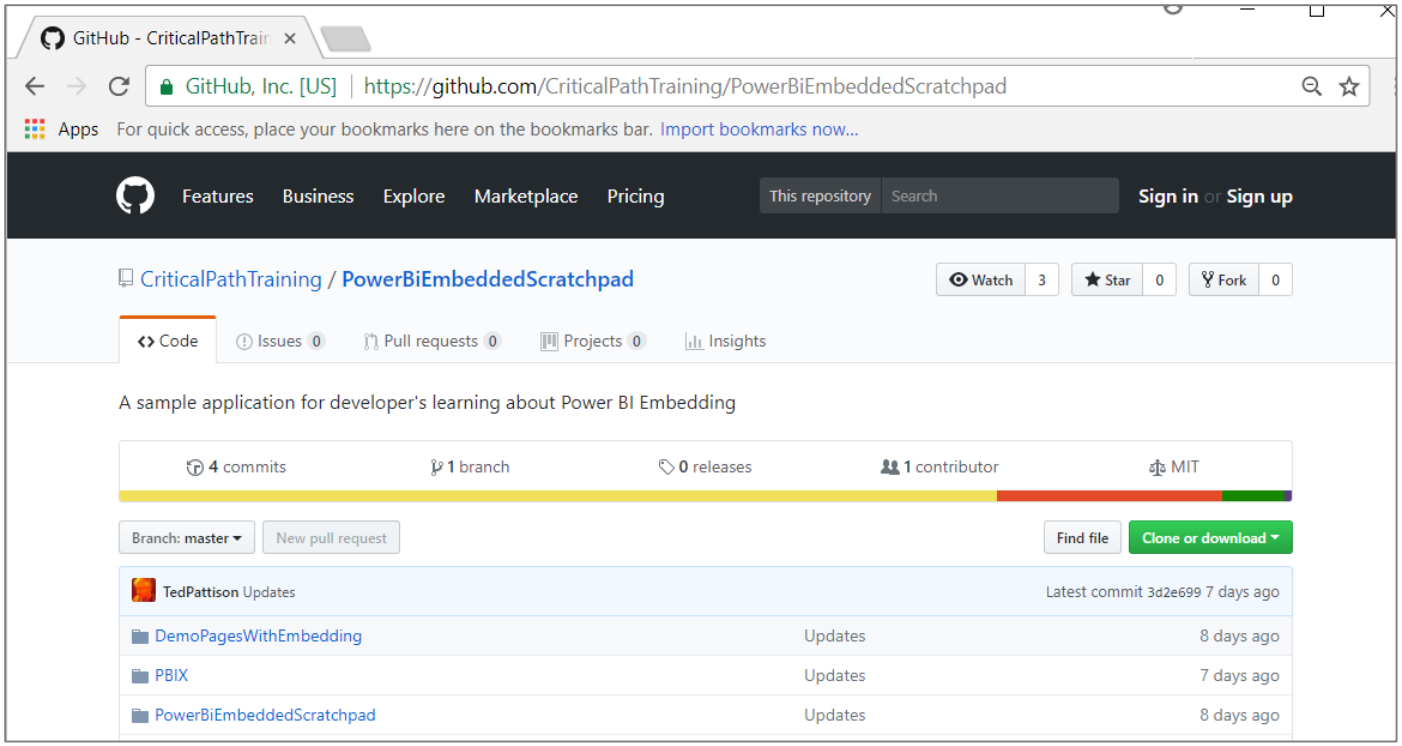
Read: Allows view report only.
ReadWrite: Allows view, Edit and Save report.
Copy: Allows Save a copy of a report using Save As.
Create: Allows creating a new report.
All: Allows everything.

View - Opens report in View mode.
Edit - Opens report in Edit mode.

fitToWidth: Fit to width of host HTML element.
oneColumn: Opens in single column.
actualSize: Actual size as designed in report

PowerBiEmbeddedScratchpad Sample

https://github.com/CriticalPathTraining/PowerBiEmbeddedScratchpad

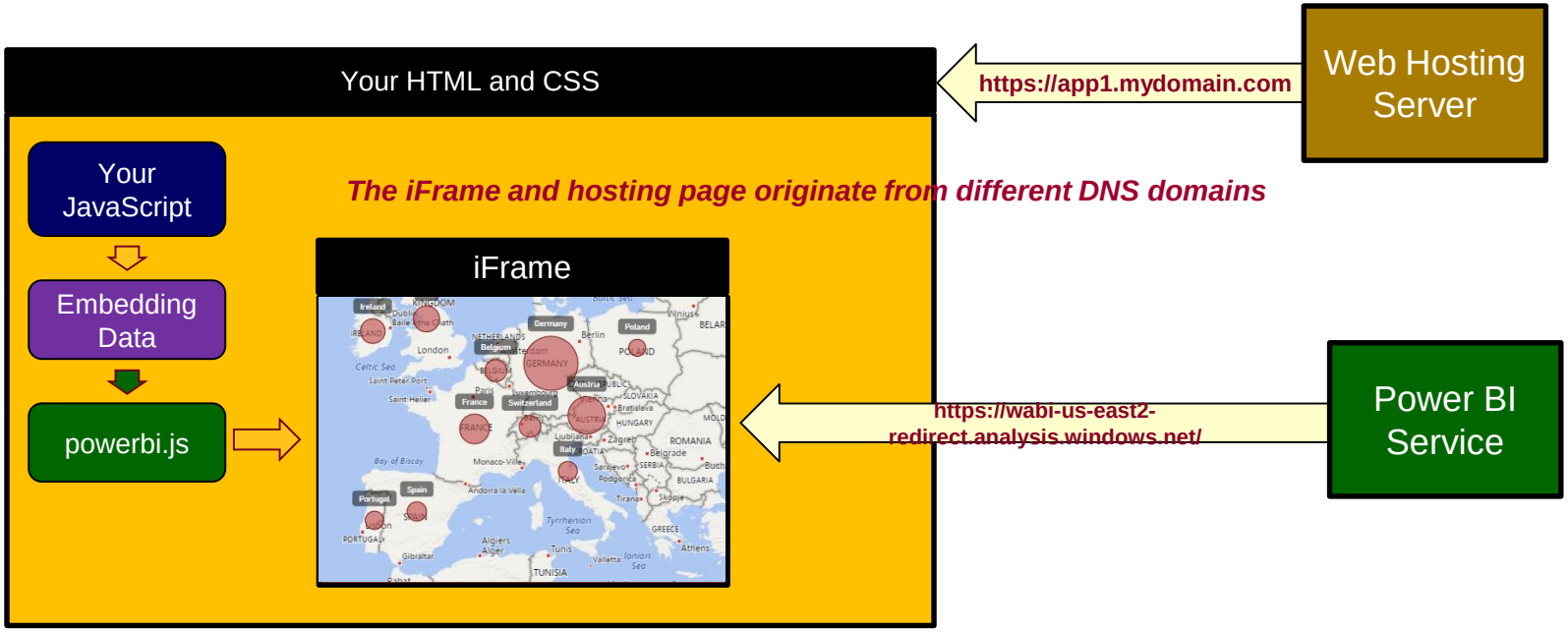


Demo Time

Cooking up something tasty using the Power BI Embedded Scratchpad

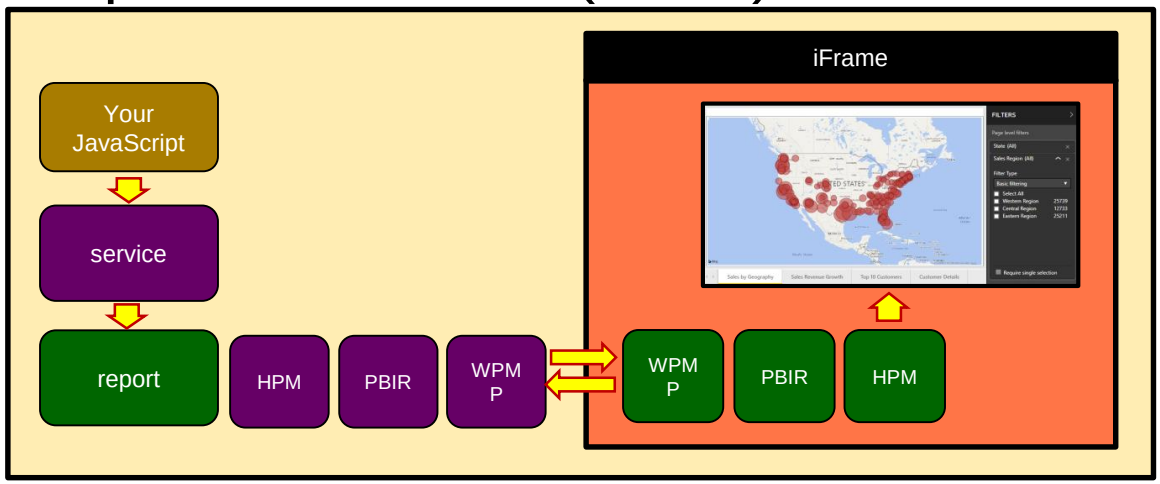
Report Embedding Architecture

- Embedding involves creating an iFrame on the page
  - **powerbi.js** library transparently creates iFrame and sets source to Power BI Service



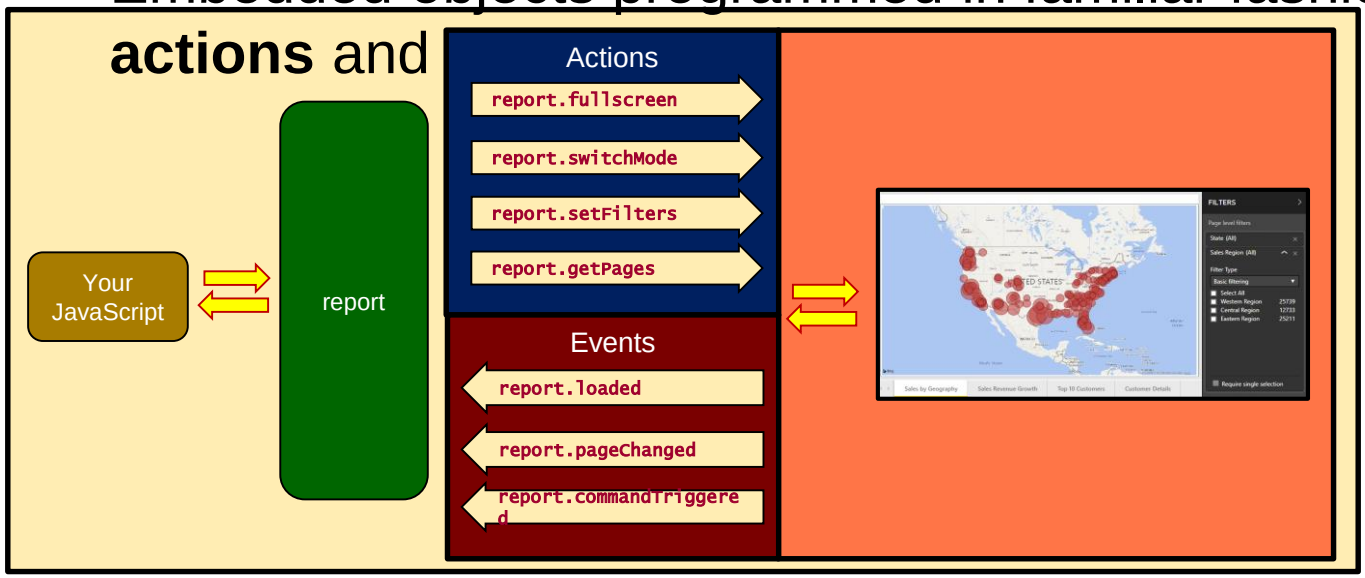
Post Message Communications Flow

- 4 extra libraries used communicate with report in iFrame
  - window-post-message-proxy (WPMP)
  - http-post-message (HPM)
  - powerbi-router (PBIR)
  - powerbi-models (PBIM)



A Promise-based Programming Model

- Design of **powerbi.js** library simulates HTTP protocol
  - Creates more intuitive programming model for developers
  - Programming based on asynchronous requests and promises
  - Embedded objects programmed in familiar fashion using **actions** and



- Report object exposes action methods
  - **report.swtichMode** used to switch report between view mode and edit mode
  - **report.updateSettings** used to hide/show filter pane and page tabs
  - **report.fullscreen** used to bring report into full screen mode
  - **report.print** used to print the report

```
// Embed the report
var report = powerbi.embed(reportContainer, config);

// add variable to track current display mode
var viewMode = "view";

// add command handler to toggle between display mode and edit mode
$("#toggleEdit").click(function () {
  viewMode = (viewMode == "view") ? "edit" : "view";
  // call report.switchMode to change report edit mode
  report.switchMode(viewMode);
  var showFilterPane = (viewMode == "edit");
  // report.updateSettings hide or show filter pane
  report.updateSettings({ "filterPaneEnabled": showFilterPane });
});

// add  command handler to enter full screen mode
$("#fullScreen").click(function () { report.fullscreen(); });

// add  command handler to print report
$("#print").click(function () { report.print(); });
```

Handling Report Events

```
var report = powerbi.embed(embedContainer, config);

var pages;

report.on('loaded', function () {
  // call getPages with callback
  report.getPages().then(
    function (reportPages) {
      pages = reportPages;
      // call method to load pages into nav menu
      loadReportPages(pages);
    });
});

var loadReportPages = function (pages) {
  for (var index = 0; index < pages.length; index++) {
    // determine which pages are visible and not hidden
    if (pages[index].visibility == 0) { // 0 means visible and 1 means hidden
      var reportPageDisplayName = pages[index].displayName;
      pageNavigation.append($("<li>")
        .append($('<a href="javascript:;" >')
        .text(pages[index].displayName))
        .click(function (domEvent) {
          var targetPageName = domEvent.target.textContent;
          // get target page from pages collection
          var targetPage = pages.find(function (page) { return page.displayName === targetPageName; });
          // navigate report to target page
          targetPage.setActive();
        }));
    }
  }
}
```

```
▼ Report
  ▼ allowedEvents: Array(12)
      0: "loaded"
      1: "saved"
      2: "rendered"
      3: "saveAsTriggered"
      4: "error"
      5: "dataSelected"
      6: "filtersApplied"
      7: "pageChanged"
      8: "commandTriggered"
      9: "swipeStart"
      10: "swipeEnd"
      11: "bookmarkApplied"
```

Embedding a New Report

```
// Get data required for embedding
var embedWorkspaceId= "7f4576c7-039a-472f-b998-546a572d5da2";
var embedDatasetId = "b4a48602-71da-42b2-8cf5-44d35b2ac70b";
var embedUrl = "https://app.powerbi.com/reportEmbed?groupId=7f4576c7-039a-472f-b998-546a5
var accessToken = "H4sIAAAAAAAAEAB2Wxw6OCA6E3-W_shIZmpXmQE5NztzIOWdG--7bM3dbsj67qvz3HzN5-i

// Get models object to access enums for embed configuration
var models = window['powerbi-client'].models;

var config = {
  datasetId: embedDatasetId,
  embedUrl: embedUrl,
  accessToken: accessToken,
  tokenType: models.TokenType.Embed,
};

// Get a reference to the embedded report HTML element
var embedContainer = document.getElementById('embedContainer');

// Embed the report and display it within the div container.
var report = powerbi.createReport(embedContainer, config);
```

New Report with SaveAs Redirect

```
// Embed the report and display it within the div container.
var newReport = powerbi.createReport(embedContainer, config);

// this event fires whenever user runs save or SaveAs command on a new report
newReport.on("saved", function (event) {

  // get ID and name of new report
  var newReportId = event.detail.reportObjectId;
  var newReportName = event.detail.reportName;

  // set new title for browser window
  document.title = newReportName;

  // reset report container element
  powerbi.reset(embedContainer);

  config = {
    type: 'report',
    id: newReportId,
    embedUrl: "https://app.powerbi.com/reportEmbed?reportId=" + newReportId + "&groupId=" + embedWorkspaceId,
    accessToken: accessToken,
    tokenType: models.TokenType.Aad,
    permissions: models.Permissions.All,
    viewMode: models.ViewMode.Edit,
  };

  // Embed the report and display it within the div container.
  var savedReport = powerbi.embed(embedContainer, config);
```

Embedding the Q&A Experience

```
// Get data required for embedding
var datasetId = "b4a48602-71da-42b2-8cf5-44d35b2ac70b";
var embedUrl = "https://app.powerbi.com/qnaEmbed?groupId=7f4576c7-039a-472f-b998-546a57
var accessToken = "H4sIAAAAAAAAEAC2Wx6rFDI6E3-XfeuA4h4FeOOecvXPOObuZd58L3XuBpK8Klf79j5W-

// Get models object to access enums for embed configuration
var models = window['powerbi-client'].models;

var config = {
  type: 'qna',
  tokenType: models.TokenType.Embed,
  accessToken: accessToken,
  embedUrl: embedUrl ,
  datasetIds: [ datasetId ],
  viewMode: models.QnaMode.Interactive,
  question: "What is sales revenue by quarter and sales region as stacked area chart"
};

var embedContainer = document.getElementById('embedContainer');

var embeddedObject = powerbi.embed(embedContainer, config);
```

Summary

- ✓ Power BI Embedding Fundamentals
- ✓ Managing Content using App Workspaces
- ✓ Understanding Dedicated Capacities
- ✓ Using the Onboarding Experience Tool
- ✓ Programming with Power BI Service API
- ✓ Embedding with the Power BI JavaScript API