# Microsoft Cloud Development in a Day: Power Platform + Azure

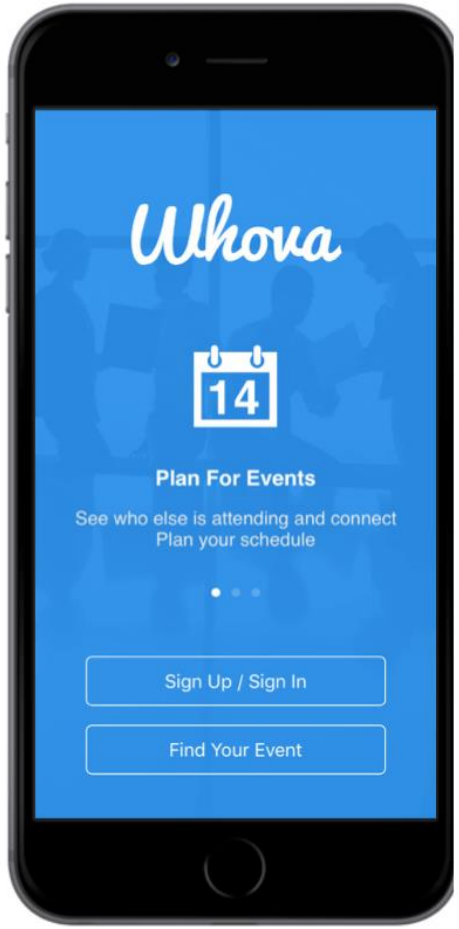
April Dunnam

Dan Wahlin

intersection

POWERED BY
Microsoft & NextGen

# Get Whova for DEVintersection, Microsoft Azure + AI Conference, Azure Data and M365 Conference - Fall 2022




## DEVintersection, Microsoft Azure + AI Conference, Azure Data and M365 Conference - Fall 2022 Official Event App

- Explore the **professional profiles** of event speakers and attendees
- Send **in-app messages** and **exchange contact info**
- **Network and find attendees** with common affiliations, educations, shared networks, and social profiles
- Receive **update notifications** from organizers
- Access the **event agenda**, GPS guidance, maps, and parking directions at your fingertips

Available on the App Store  ANDROID APP ON Google play

The event invitation code is:
**FallConf22**

**DEC 5-7**
**2023**
**WORKSHOPS**
**DEC 3, 4 & 8**

## DEVintersection Conference

**DEVintersection.com**

## Microsoft Azure + AI Conference

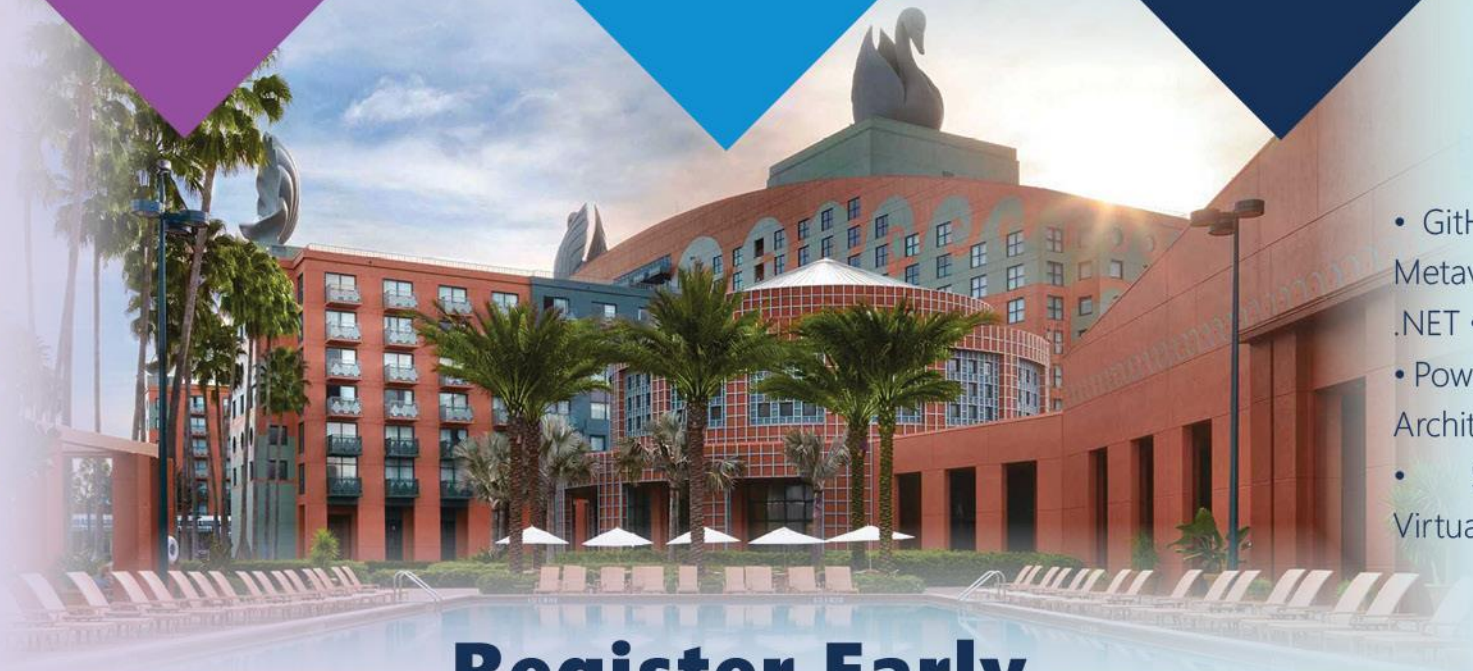
**AzureAIConf.com**

## Azure Data Conference

**AzureDataConf.com**

**DISNEY WORLD**
**SWAN & DOLPHIN RESORT**

AI • Angular • Architecture • ASP.NET • Azure • Azure Data • Azure Databricks • Azure Functions • Azure IoT • Azure SQL• Azure Synapse C# • Blazor • Cloud Security • Codespaces • Cognitive Services • CosmosDB • Dapr • Data Science • DevOps • Docker



• GitHub Actions • Machine Learning • Metaverse • Microservices • MySQL • .NET • .NET MAUI • Node.js • PostgreSQL • Power Apps • Power BI • React • Scalable Architectures • Security & Compliance • SQL Server 2022 TypeScript • Virtual Machines • Visual Studio

# Register Early
## for Optional Workshops

**and Receive Your Option of Hardware**

# Goals for today

- Get you familiar with the Power Platform.
- Extend it using your existing skills.
- Understand how you can integrate Power Platform with Microsoft Cloud services.

Not trying to convert you to a Citizen Developer.

# About Us

## April Dunnam

https://youtube.com/@aprildunnam

@AprilDunnam

## Dan Wahlin

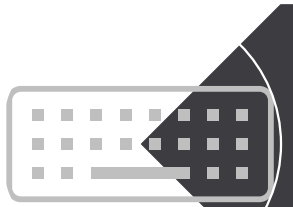
https://blog.codewithdan.com

@DanWahlin

# Get the Content

https://aka.ms/DevIntersectionPowerAzureWorkshop

# Who are you?

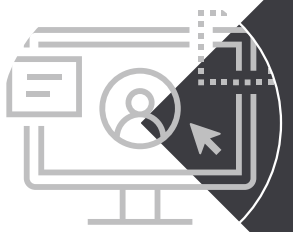


## Coder

- .NET dev

## Automation specialist

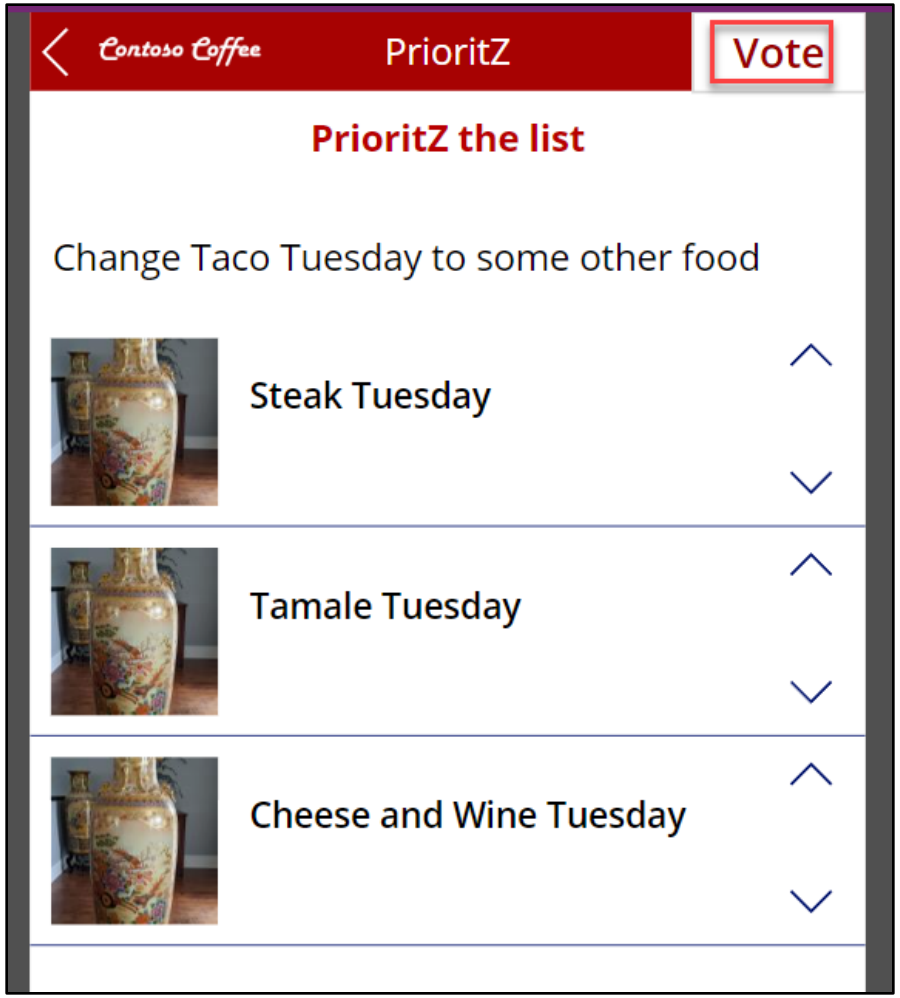
- Ready to code



## Power Platform dev

- Time to learn more

# You are developing for Contoso Coffee today



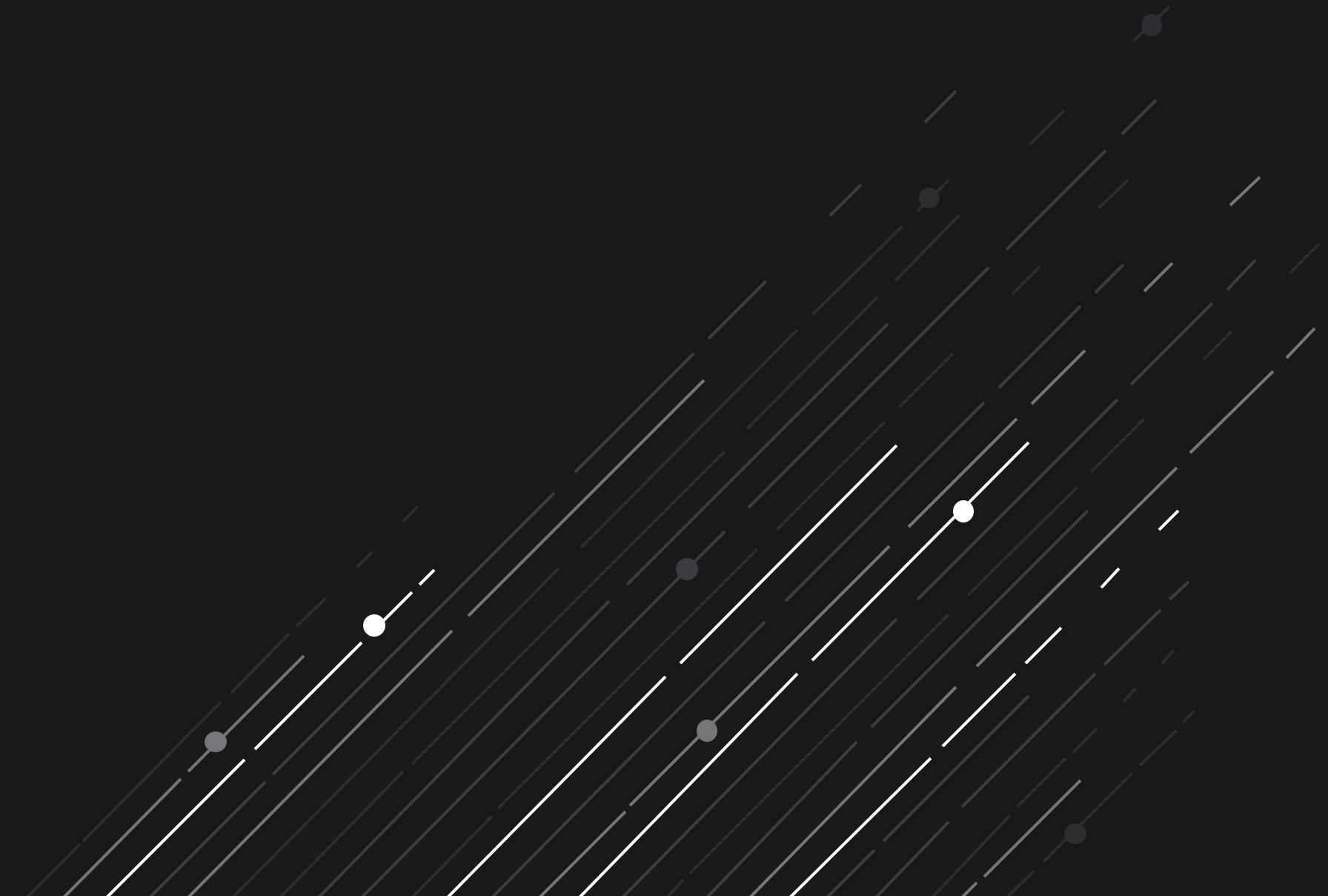

## Steps

| | | | | |
|---|---|---|---|---|
| Set up | Build a code component | Build a custom connector for an existing API | Use an Azure Function | Application Lifecycle Management |

# Questions before we get started?

Microsoft Power Platform

# Overview



# Power Platform: Everyone's platform for transformation

## No Code

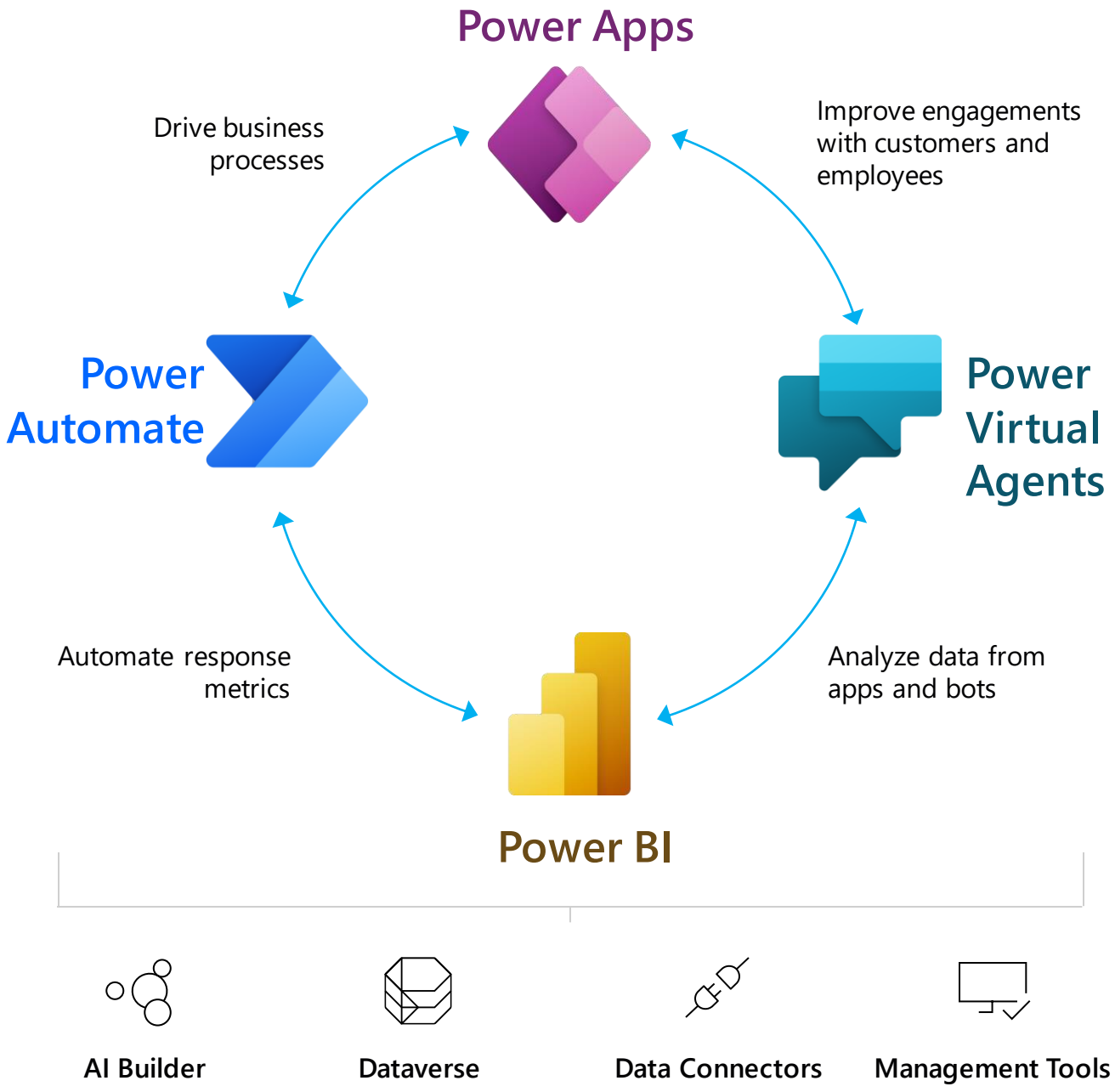
Drag and drop experiences that are like PowerPoint

## Low Code

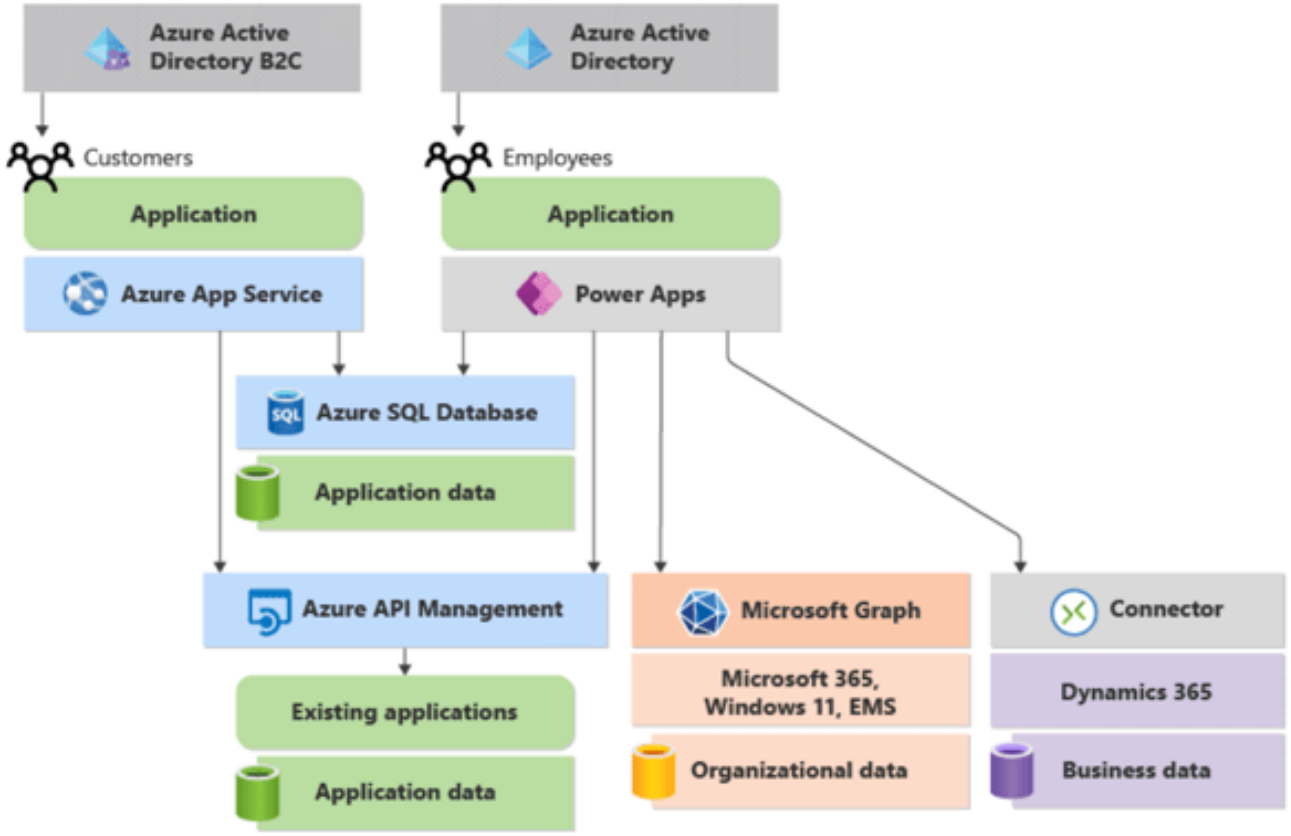
Drag and drop plus Excel-like formulas for functionality

## Code First

Professional development and extensibility across Microsoft



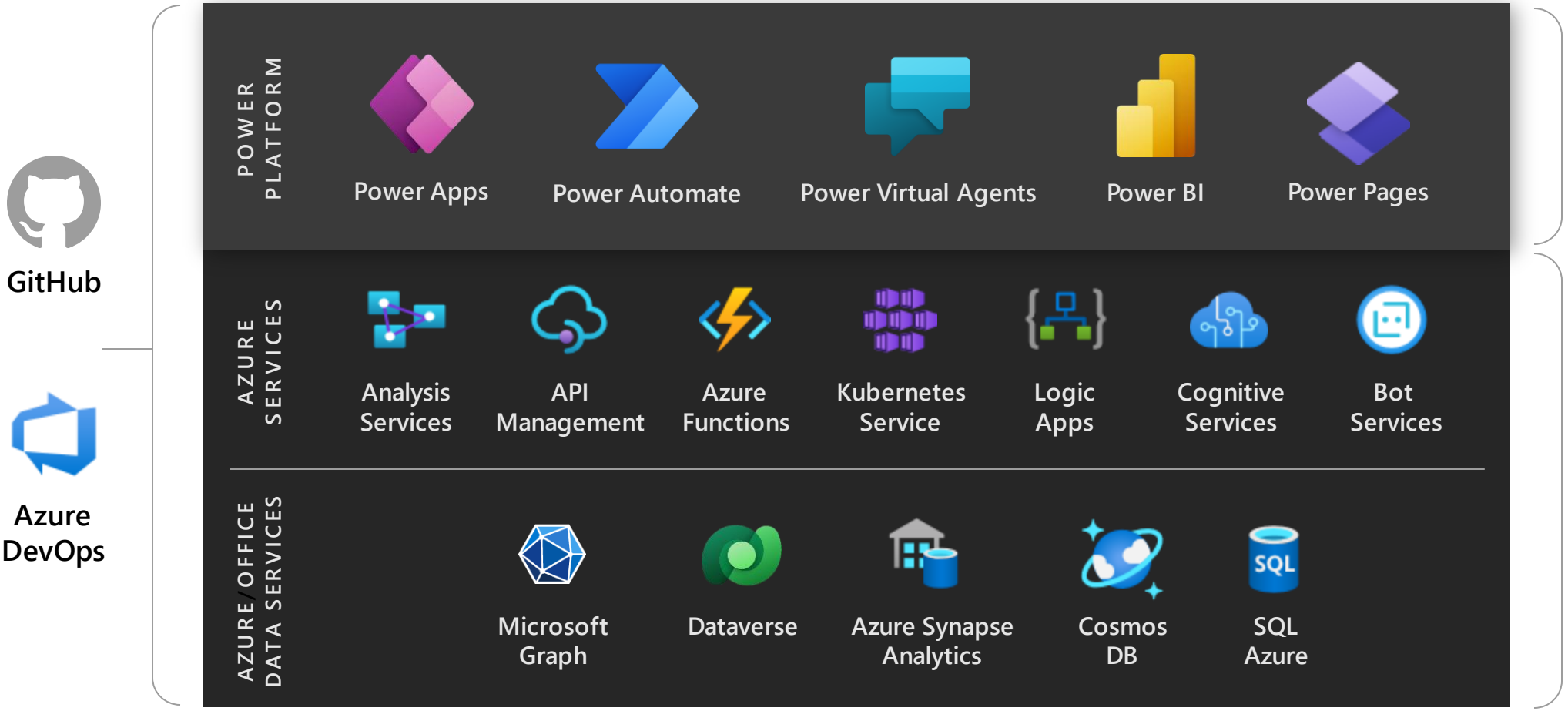
# Building on the Microsoft Cloud



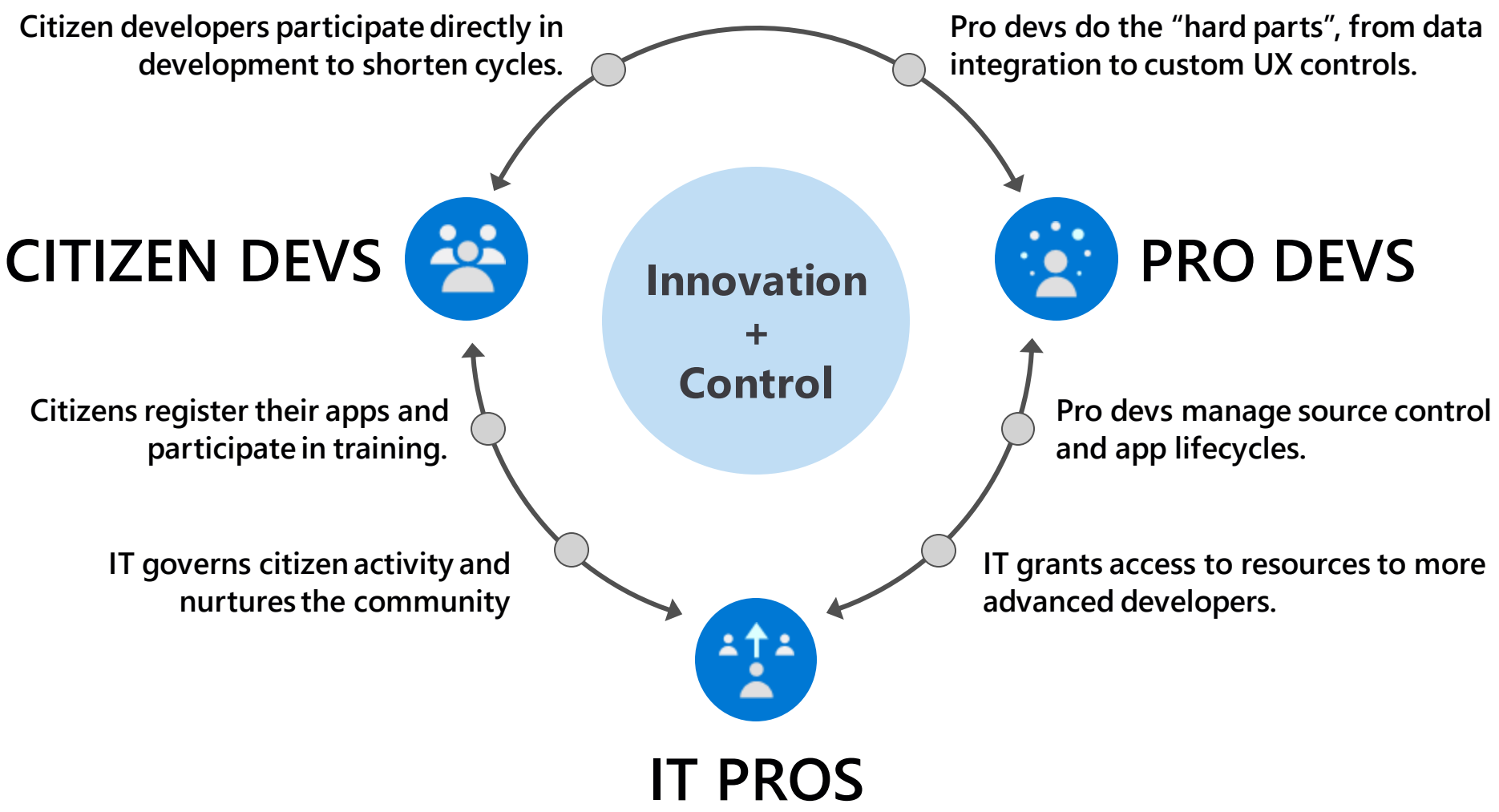

https://aka.ms/microsoft-cloud-solutions

# Extend the Power Platform with Azure Services

Integrate Azure services with pre-built and custom APIs

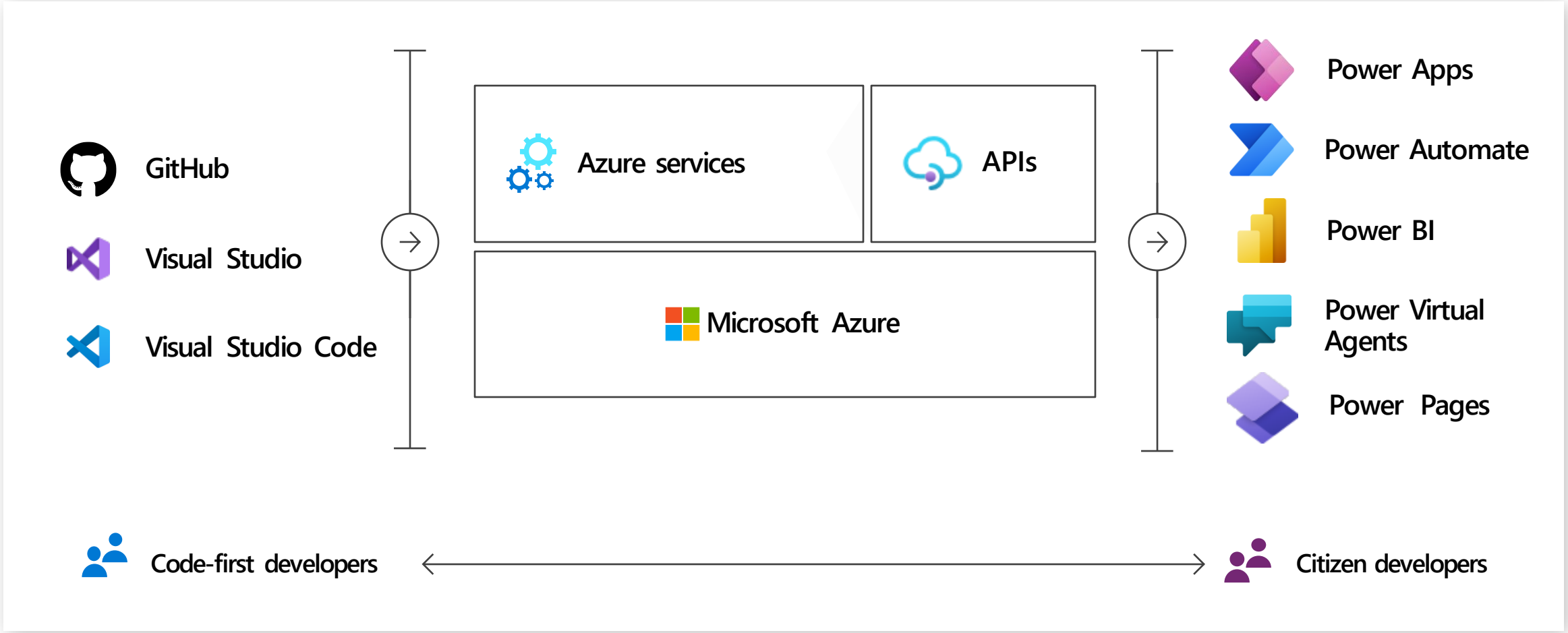


# Power Platform development is a team sport



# Fusion Team Development

Low-code and pro-dev application development extensibility with Power Apps + Azure



# FUSION DEV TEAMS

EMPOWER DEV + BUSINESS TEAMS (TO) CREATE BETTER SOFTWARE FASTER

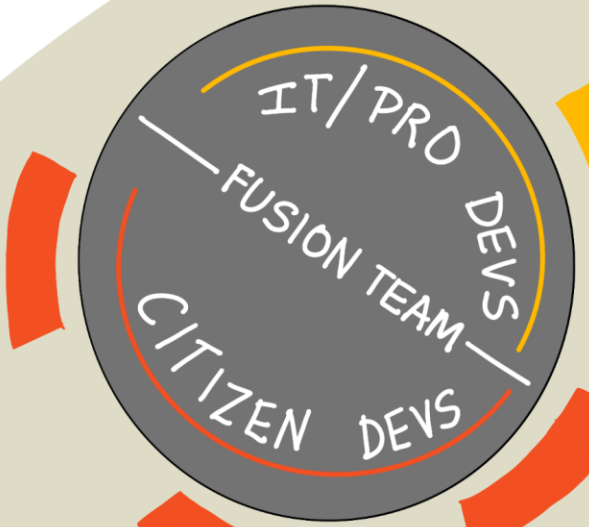
**PREETI**
IT SYSTEMS ADMIN

**MALIK**
OFFICE MANAGER

**KIANA**
FULL STACK DEV

**MARIA**
INVENTORY MANAGEMENT

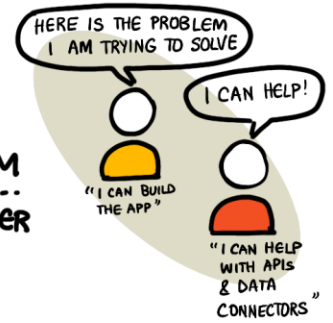
**CALEB**
FIELD TECHNICIAN




# TEAM PROCESS

EVERYONE CONTRIBUTES WITHIN THEIR COMFORT ZONE

**A KICKOFF**
CITIZEN DEVELOPER IDENTIFIES A PROBLEM
BRINGS A TEAM TOGETHER TO SOLVE IT

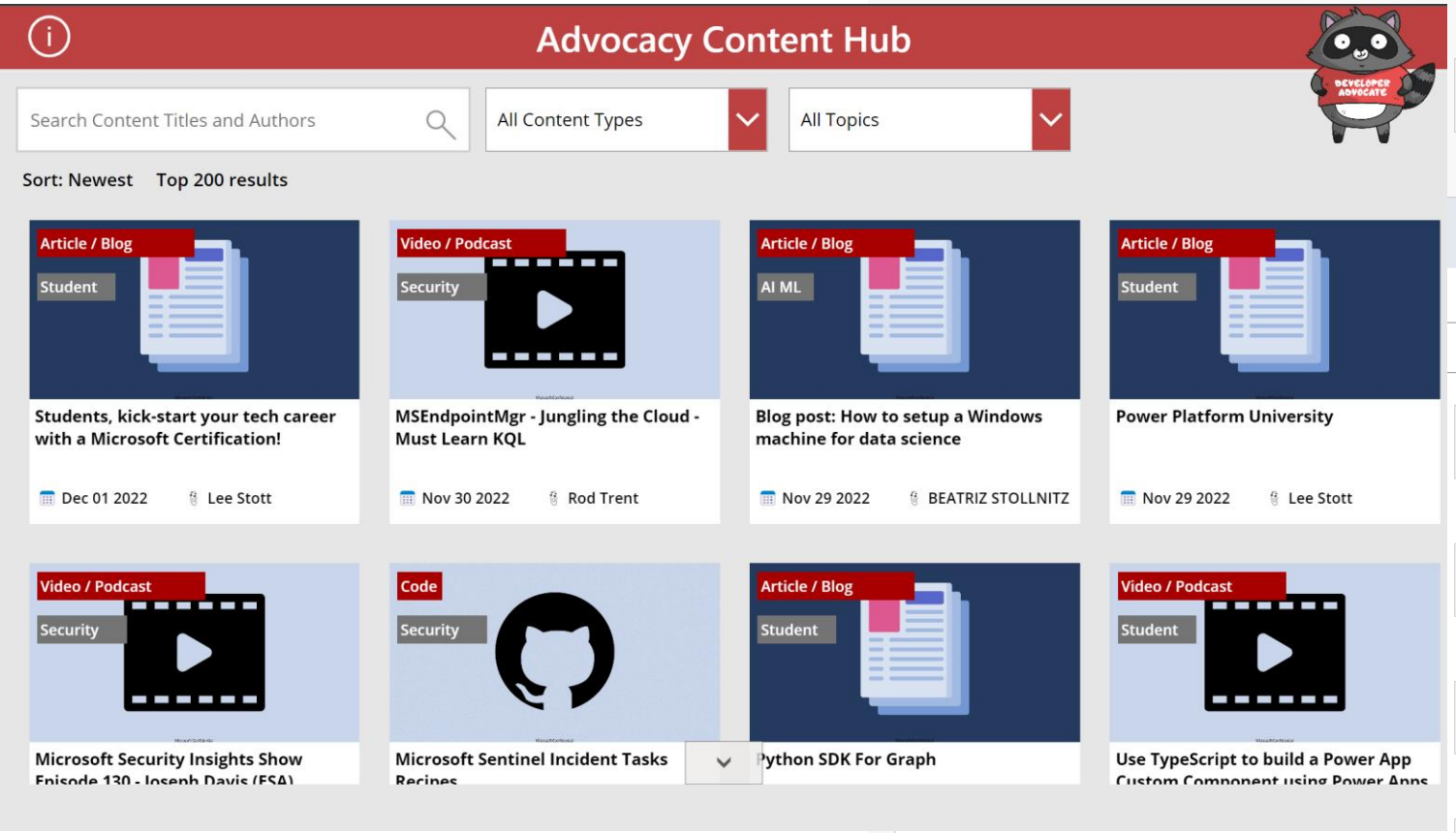


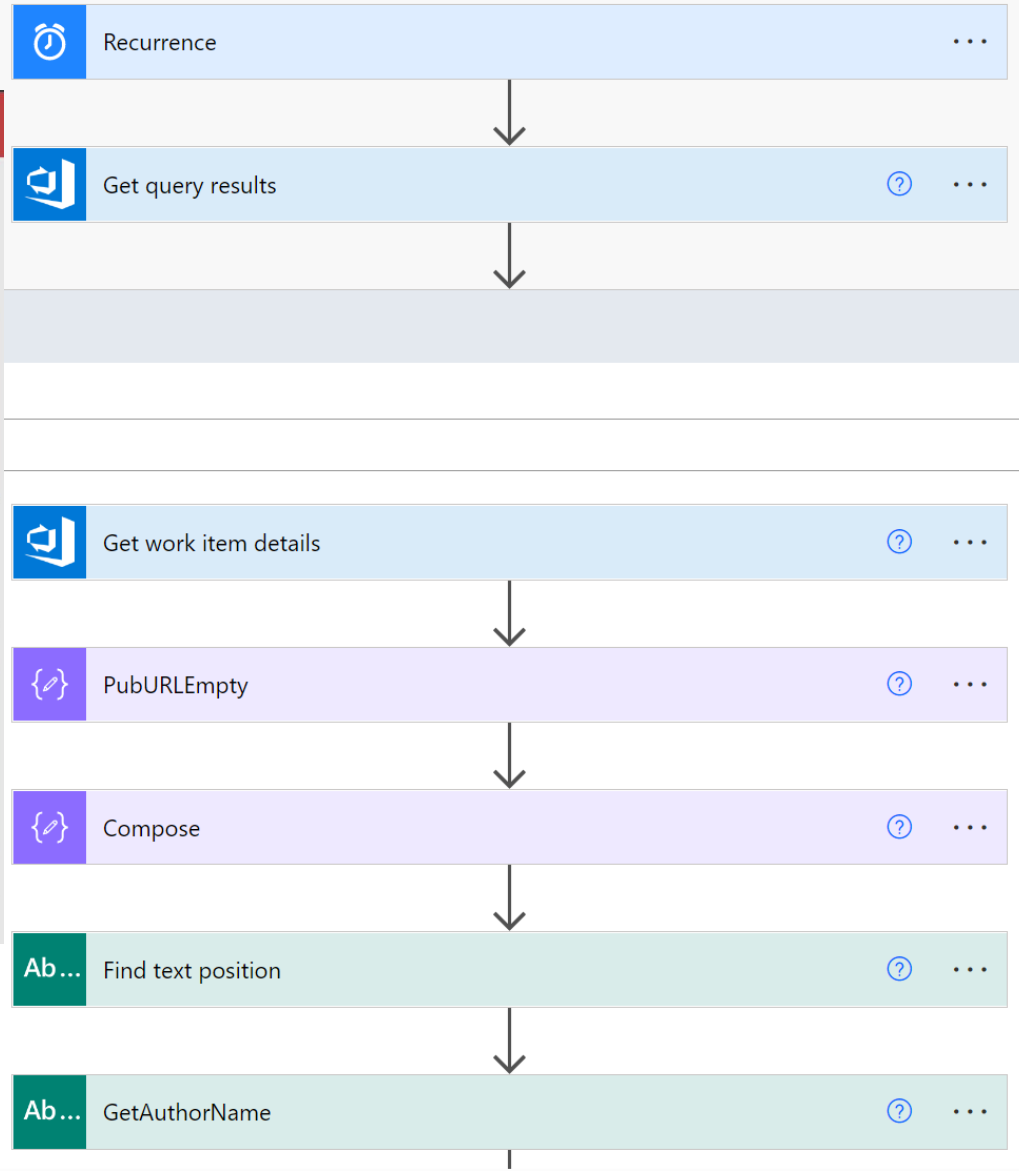

**B PLANNING**
IT/PRO DEVELOPERS IDENTIFY ADMIN/DATA NEEDS
CITIZEN DEVS FOCUS ON APP/USER EXPERIENCE

**C EXECUTION**
TEAM CHECKS IN REGULARLY TO REVIEW BLOCKERS, EVALUATE PROGRESS

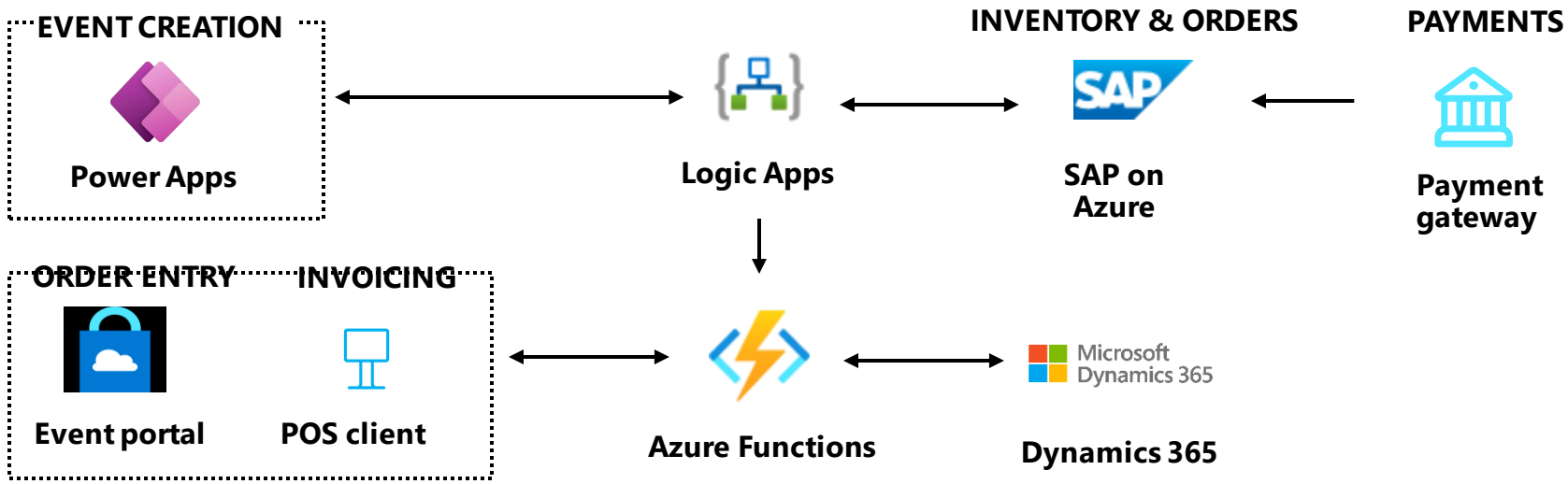
# Fusion Development Process

# Fusion Dev at Microsoft – Advocacy Content Hub



# Streamline event management, like Coca-Cola UNITED




"Power Apps has enabled us to create a more modern and efficient method of charitable event fundraising for our customers with Azure providing the ERP connections for convenient payment processing."

Bob Means
Director of Business Solutions
Coca-Cola UNITED

## Situation

Coca-Cola UNITED works directly with consumers to provide products for events, such as charity fundraisers. Previously, business development managers would manually collect order sheets from customers and manage invoicing in Excel. Payment collection was managed separately by event organizers.

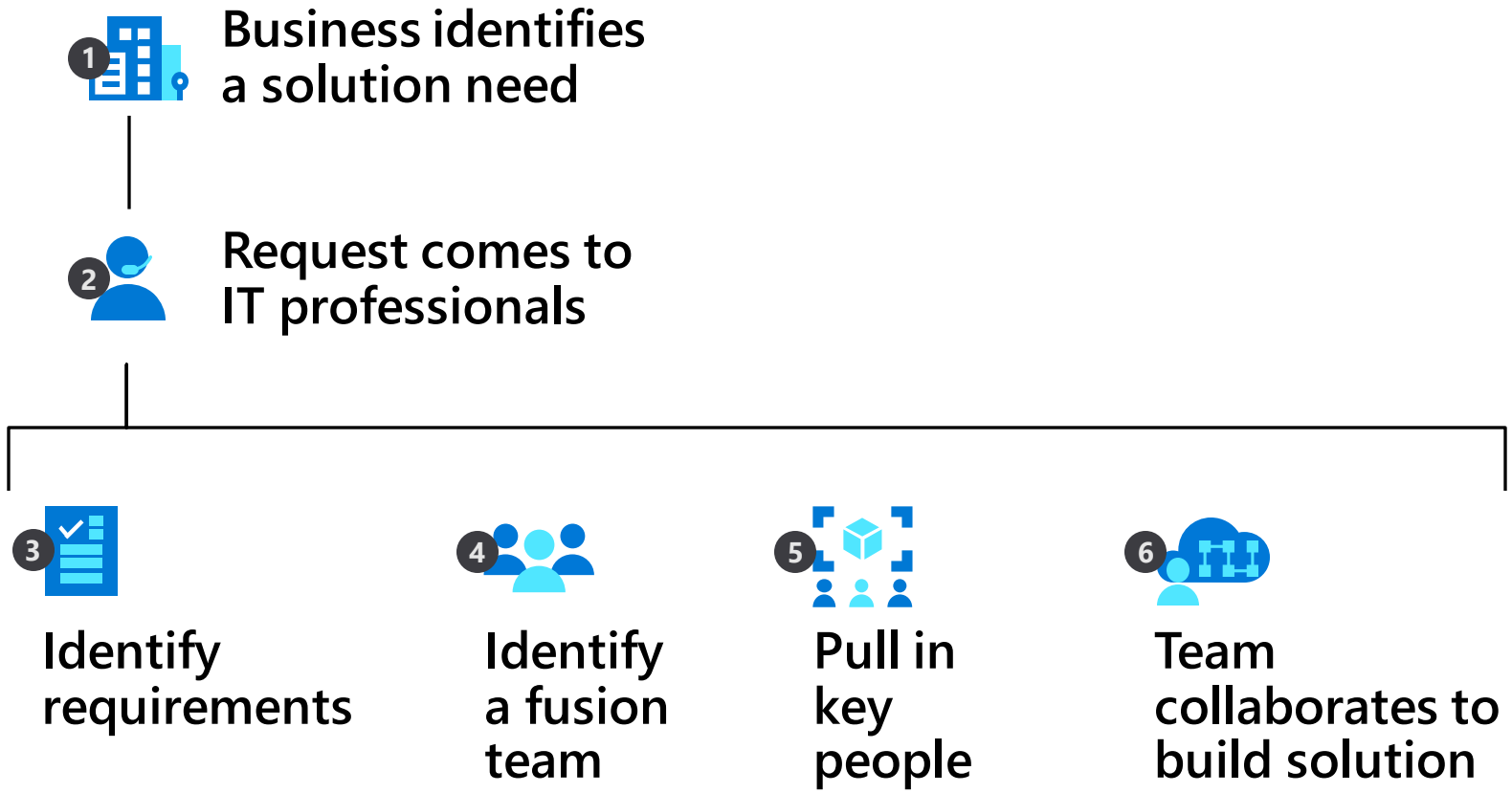
## Solution

A manager uses the Power Apps app to enter an event which triggers creation of a consumer website for orders. These orders are added to Dynamics 365 and SAP on Azure. Invoices are sent to a POS client and credit card payments are managed through a payment gateway.

## Impact

The new digital system provides customers with contactless event management, eliminating the need to submit paper order forms and collect payments from attendees. Now, customers simply submit an order online and wait for a check from Coca-Cola UNITED once the event has been completed.

# Building a flagship app using fusion teams



# The Pro Developer

## Building a flagship app—professional developers

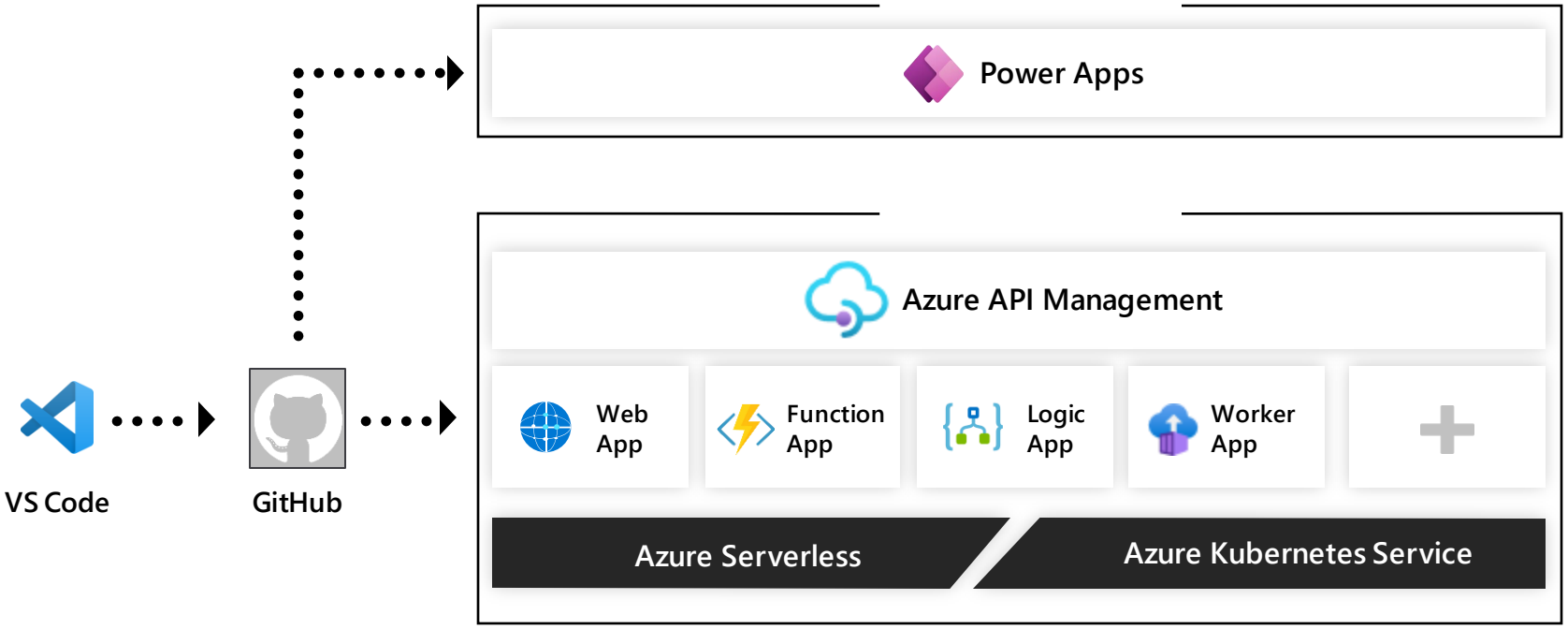
1. The pro developer starts building the service
2. Interact with legacy data
3. Builds API and publish as an Azure function to API Management
4. Leverage API Management to create Connector component for Power Platform
5. Collaborates with citizen dev to use and solve problems

# Low code to serverless for modern app dev

**Power Apps** provides a low code and rapid development environment for LOB experiences

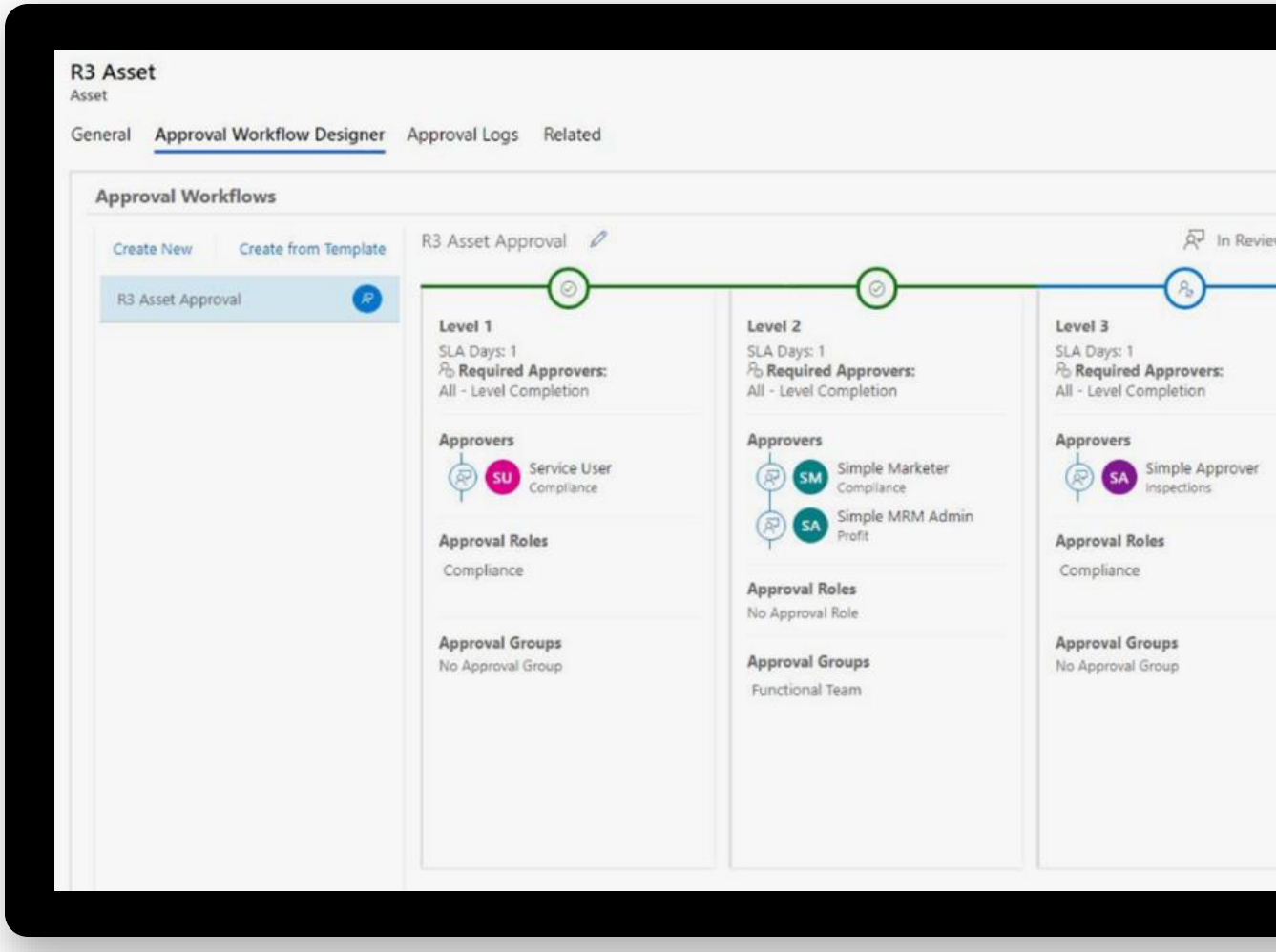
**Azure API Management** acts as the gateway to microservices and code deployed in Azure

**Azure PaaS** provides easy serverless and Kubernetes based hosting models



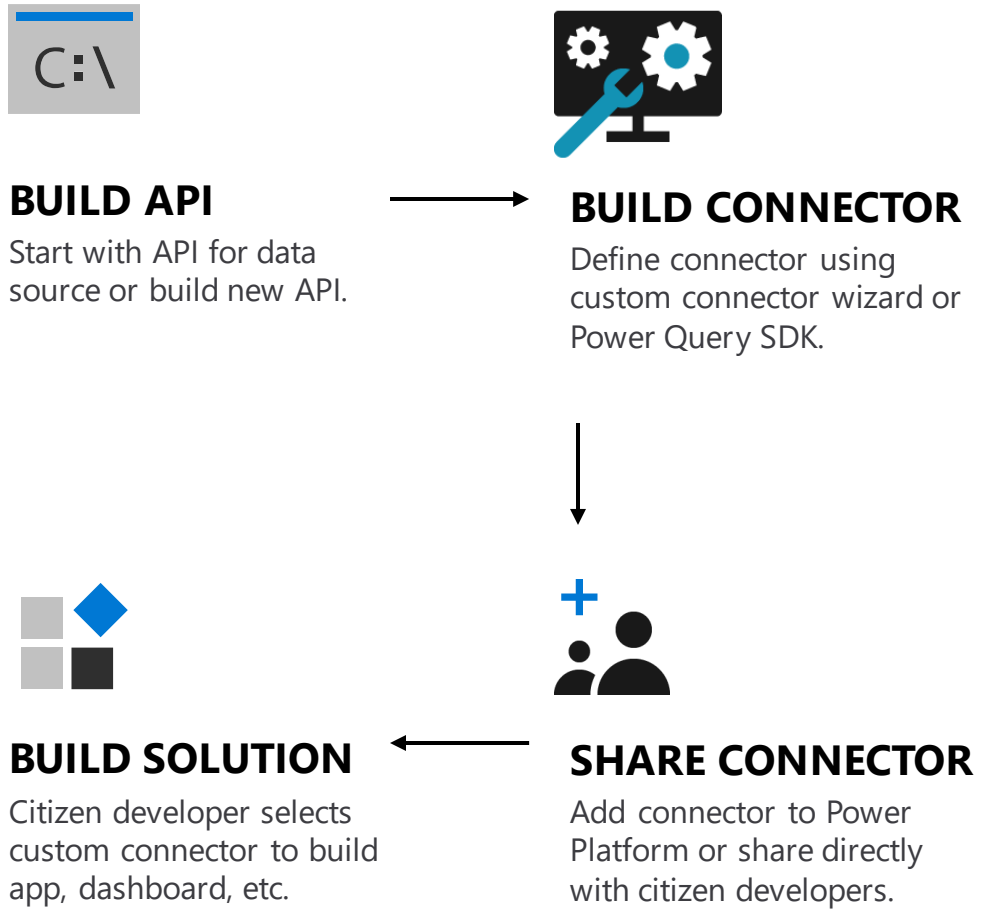
# Build custom components with the Power Apps component framework

- Developers can build compelling visual components for Power Apps.
- Custom components are responsive, reusable and support multiple form factors.
- Reuse your current IP or build using Visual Studio or Visual Studio Code
- Use common frameworks like React and command-line tools for building and testing.

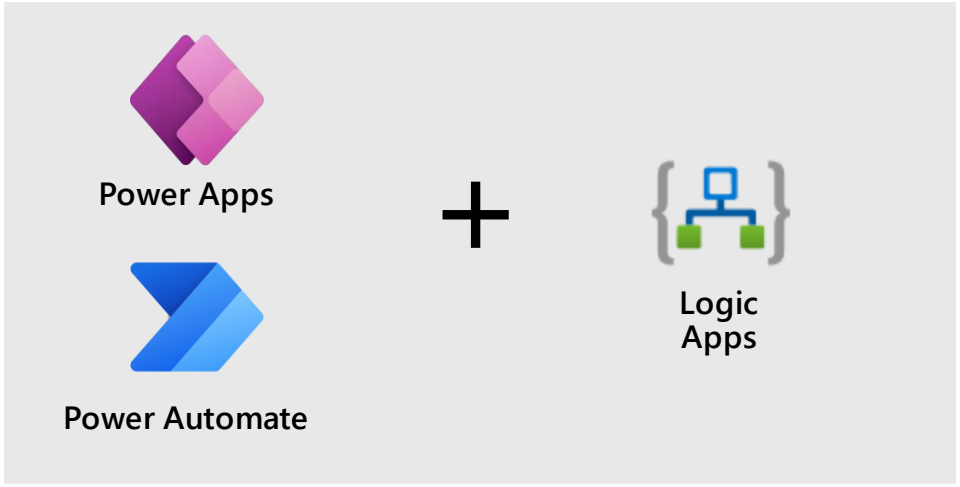


# Unlock access to any data source with custom connectors

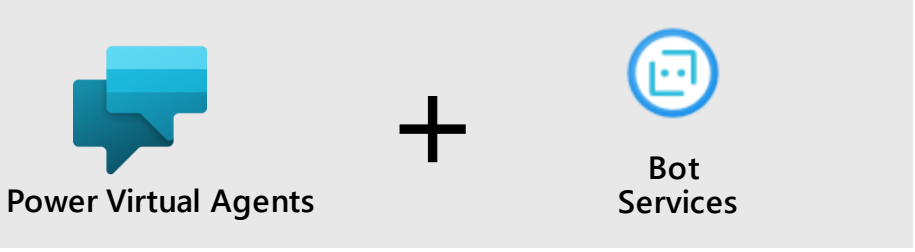
- Communicate with services that are not available as prebuilt connectors.
- Create a connector using in-product wizards in Power Platform
- Connectors in Power Automate can include your own triggers and actions.
- Connectors in Power BI can include your own logic and credential options.



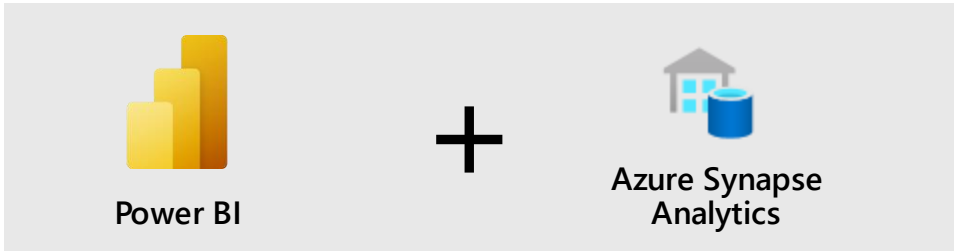
# Power Platform + Azure services in action




- Extend and expand a flow's capabilities by leveraging logic apps unique capabilities like B2B integration.
- Call your logic apps from Power Automate and Power Apps by exporting your logic apps as connectors.

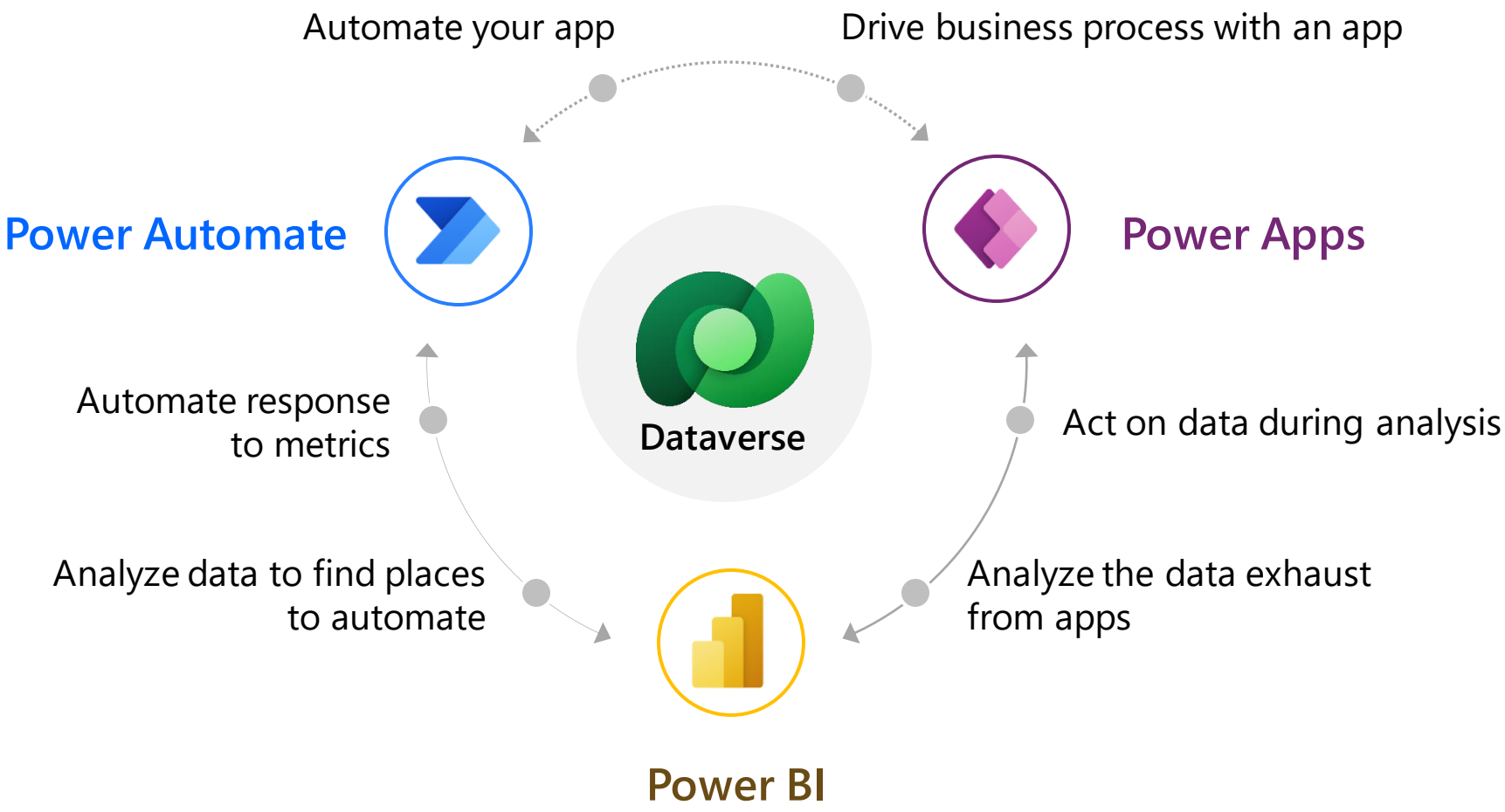



- Convert custom bots built with pro-code tools to a Skill and embed the Skill within a Power Virtual Agents bot.

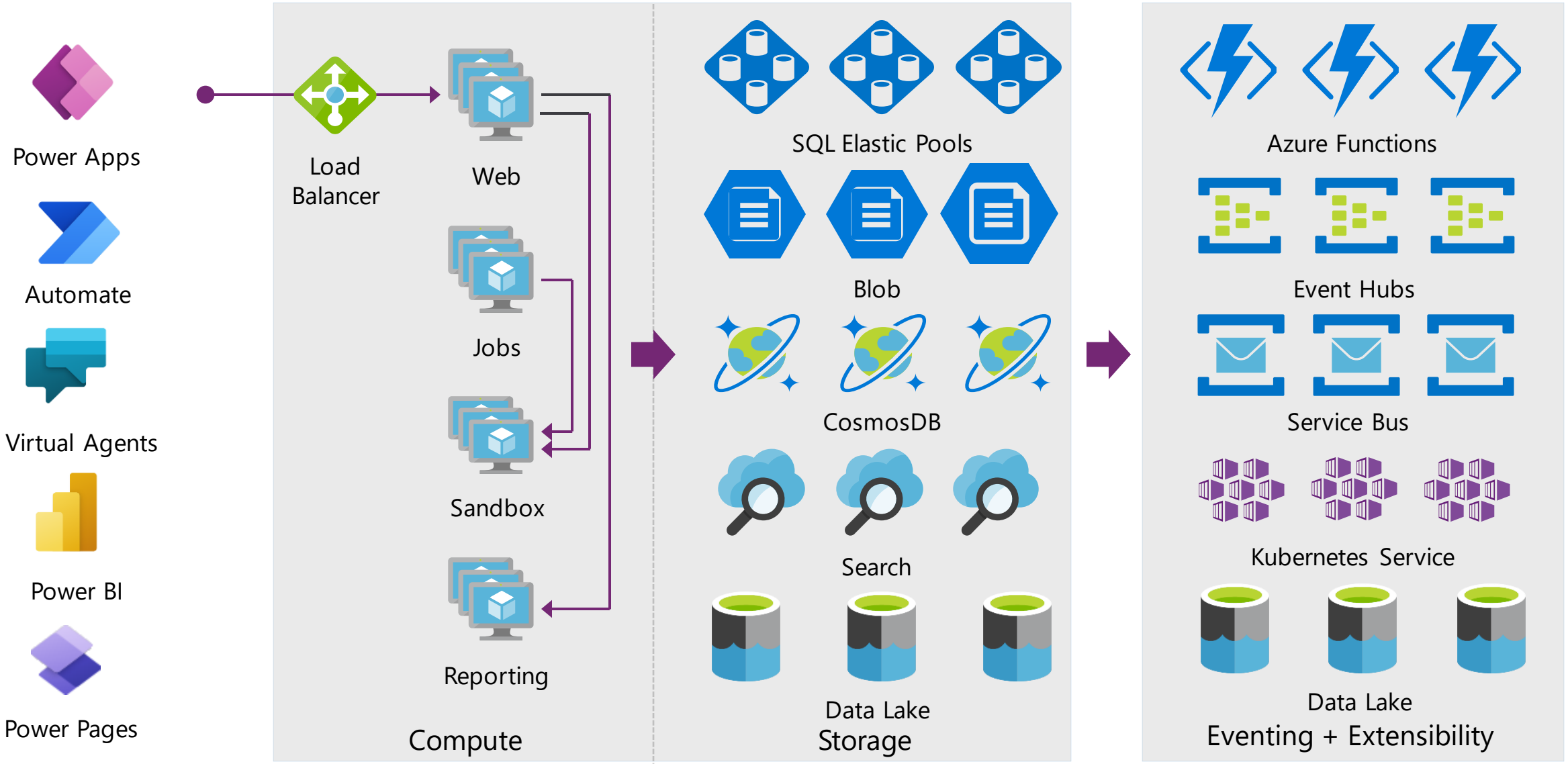



- Expand discovery of insights from all your data and reduce project development time with a limitless analytics service.

# Microsoft Dataverse is fully integrated into the Power Platform

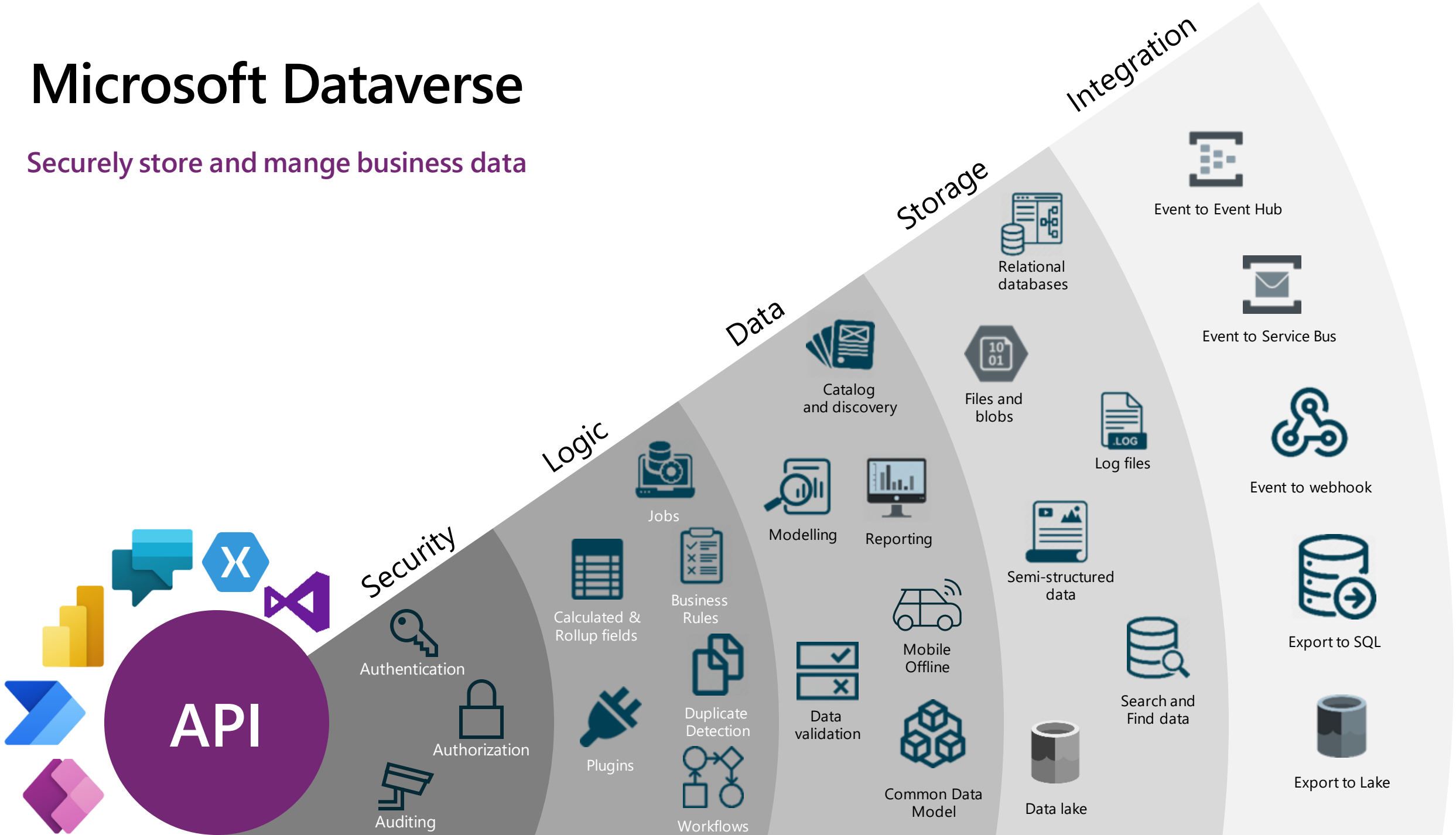


# Microsoft Dataverse... Runs on Azure and Extends with Azure



# Microsoft Dataverse

## Securely store and mange business data



# Easily integrate with Microsoft Teams

Depth of coding required

## Microsoft Teams integrations

**Ready-made apps**

*+ hundreds of partner apps*

**Low-code apps & workflows (Power Platform)**

| Power Apps | Power Automate | Power Virtual Agents | Power BI |
|---|---|---|---|
|  |  |  |  |
| Build custom apps | Automate repetitive tasks | Create tailored chatbots | Visualize and discuss data |

**Full-code custom solutions**

 Departmental tools

 Employee resources

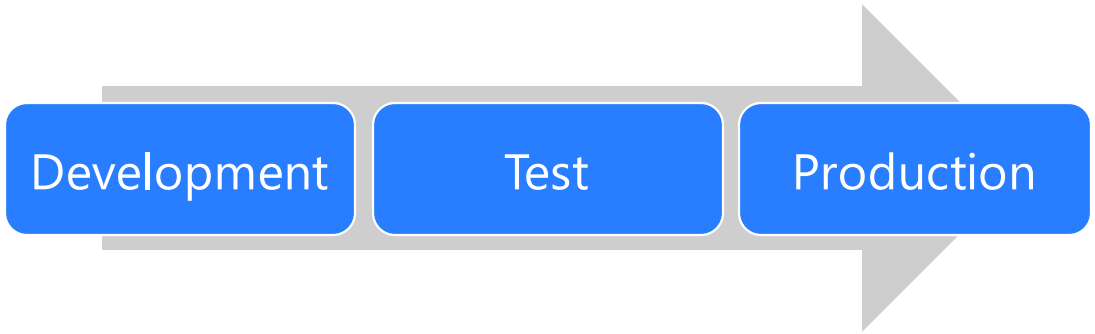
 Processes and workflow

 Support and info

# Using Solutions with Power Platform

- Solutions are a container to track the changes you make to the Microsoft Dataverse, Power Apps and Power Automate flows

- Solutions are how you transport and install changes to target environments




- Microsoft Dynamics 365 apps are installed using solutions
  - 3rd party apps provided by independent software vendors (ISVs) also use solutions

# Contoso Device Ordering Solution Example

## Solution – Device Ordering

### Apps

- Device Ordering
- Device Procurement

### Security Roles

- Device Ordering App role
- Device Procurement App role
- Device Procurement Flow role

### Data (Tables)

- Device Order

Solutions > Contoso Device Order Management

| Display name | Name | Type |
|---|---|---|
| Calculate Ship Date | Calculate Ship Date | Process |
| Contoso CDS Model Driven User | Contoso CDS Model Driven User | Security Role |
| Contoso Device Order App | Contoso Device Order App | Security Role |
| Contoso Device Procurement App | Contoso Device Procurement App | Security Role |
| Contoso Device Procurement Flow SP | Contoso Device Procurement Flow SP | Security Role |
| Approval Status | contoso_approvalstatus | Option Set |
| Device Order | contoso_deviceorder | Entity |
| Device Ordering App | contoso_deviceorderingapp_87ca2 | Canvas App |
| Device Procurement | contoso_DeviceProcurement | Model-driven App |
| Device Procurement | contoso_DeviceProcurement | Client Extension |
| Device Procurement Process | contoso_deviceprocurementprocess | Entity |
| Device approval request | Device approval request | Process |
| Device Procurement Process | Device Procurement Process | Process |

### Processes

- Order Approval Flow
- Device Procurement Process

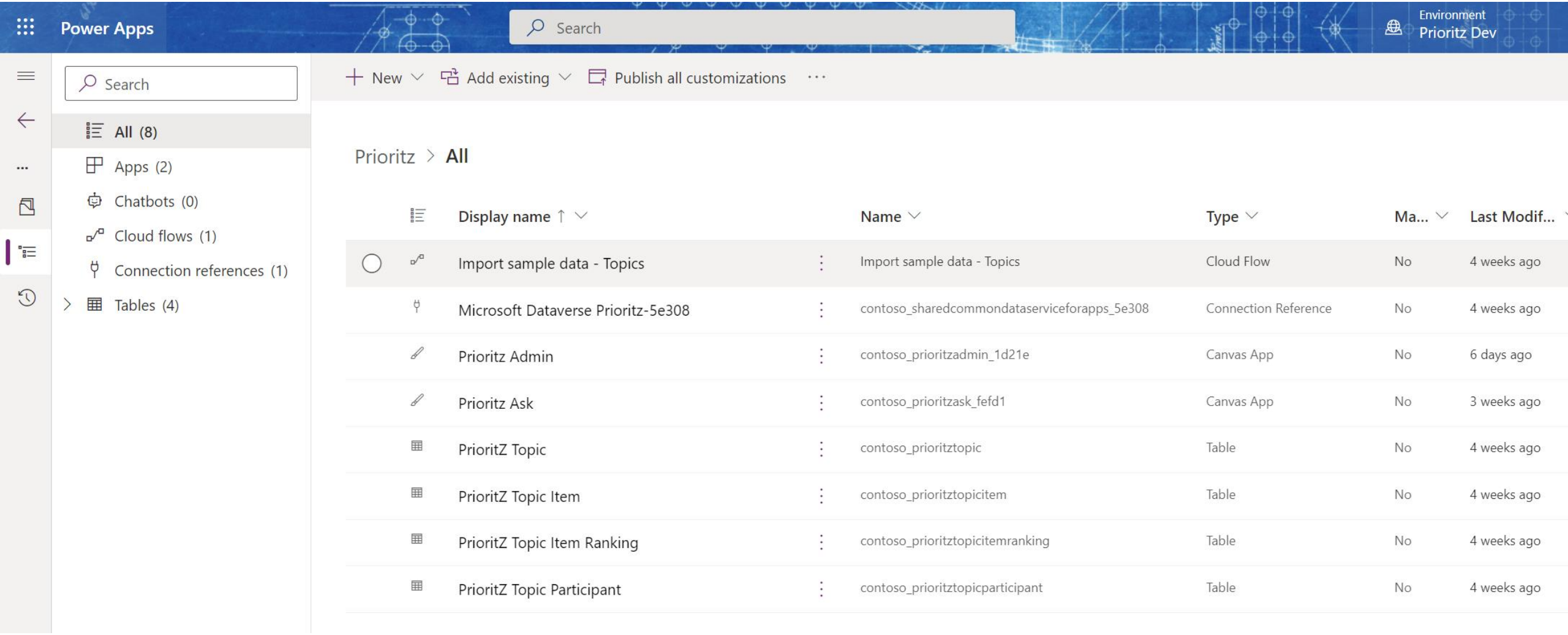
### Custom Connectors

- Computer Vendor Ordering

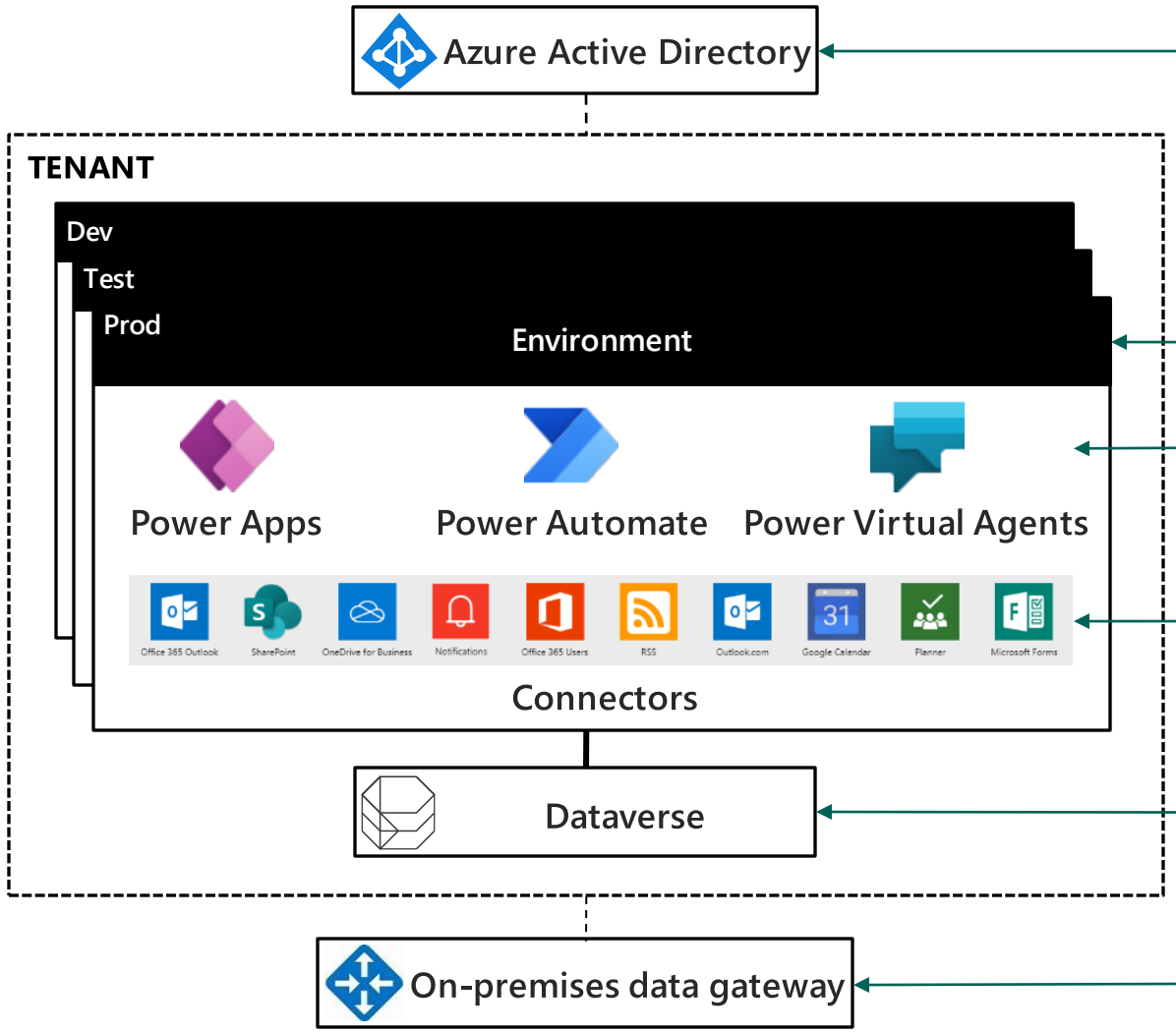
### Code Components

- Order Status Visual

# Manage solution components with Solution Explorer



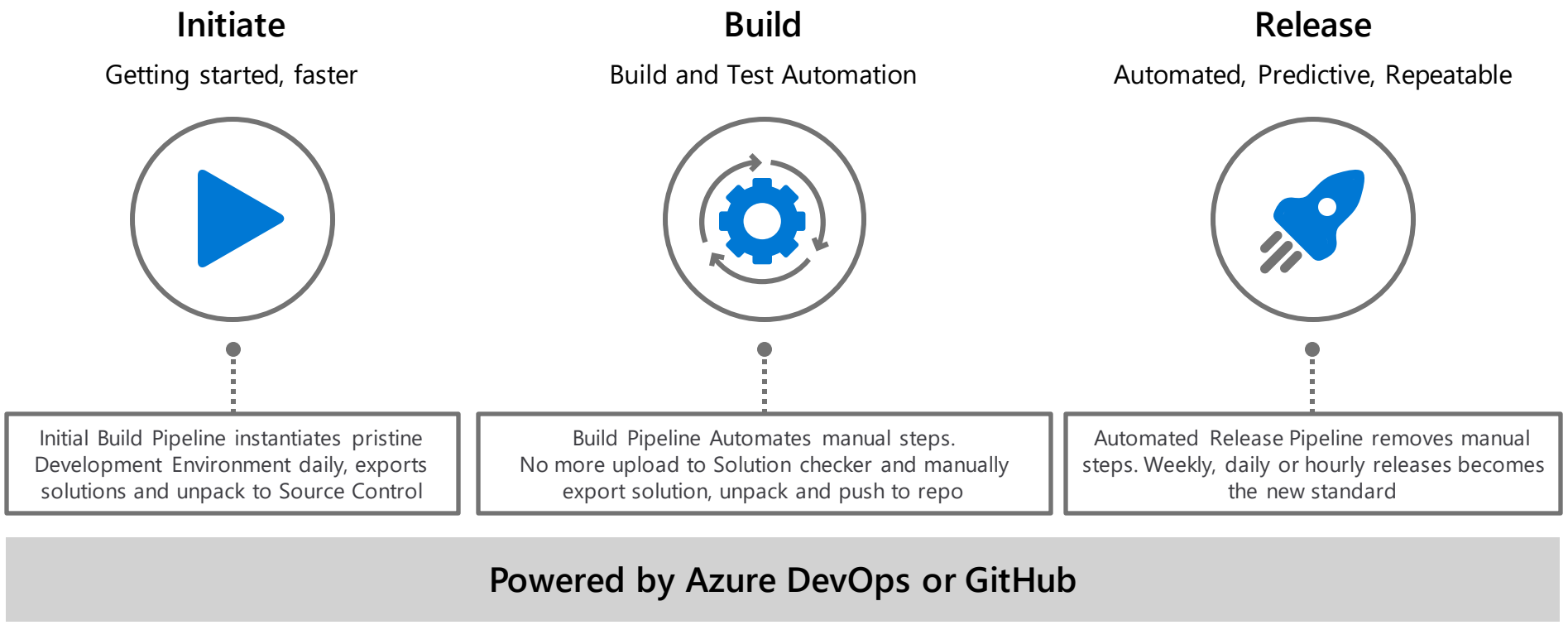
# A look at a tenant using Power Platform




1. Tenant access & isolation
2. Environment access & strategy
3. Resource permissions
4. Connector access and data loss prevention policies
5. Dataverse data security
6. On-premises data gateway

# Automate the build process with DevOps

The Microsoft Power Platform Build Tools allows anyone to setup DevOps for low-code and pro-code application development for Power Apps, Power Automate, Power Virtual Agents



# Power Platform Command Line

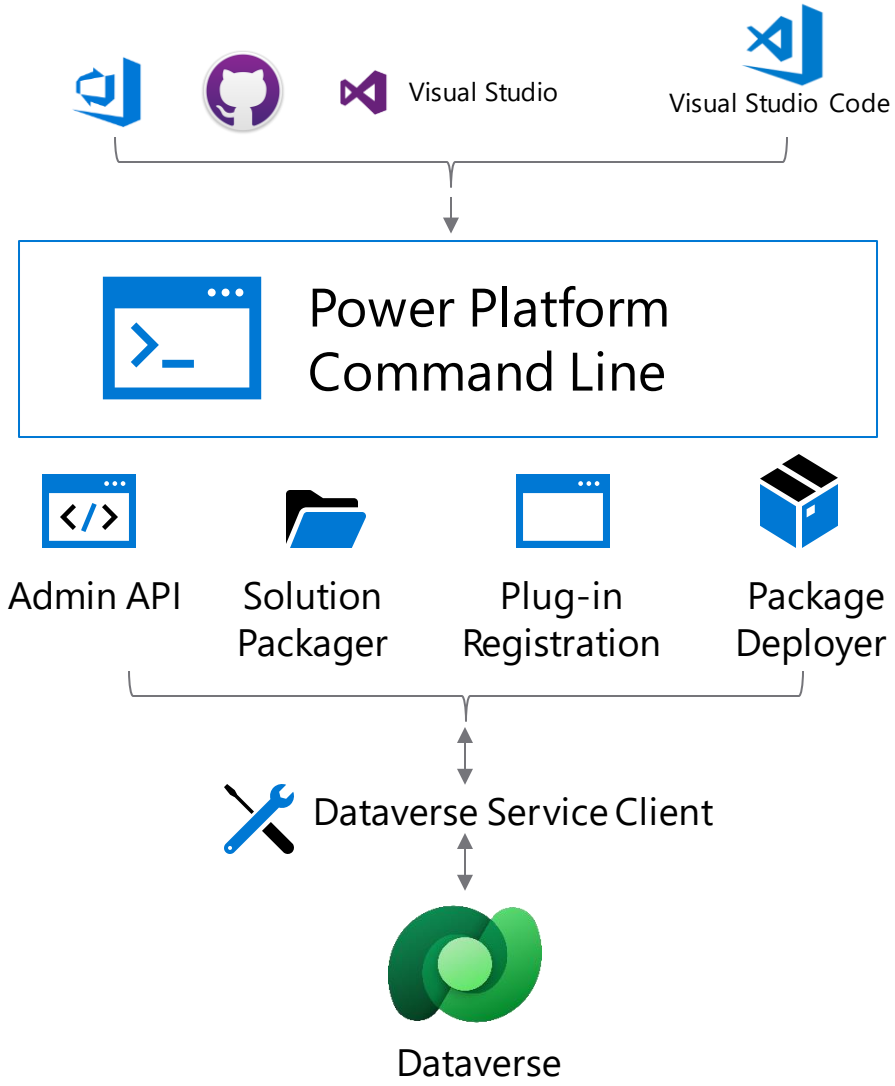
Single simplified interface

Common classes and interactions

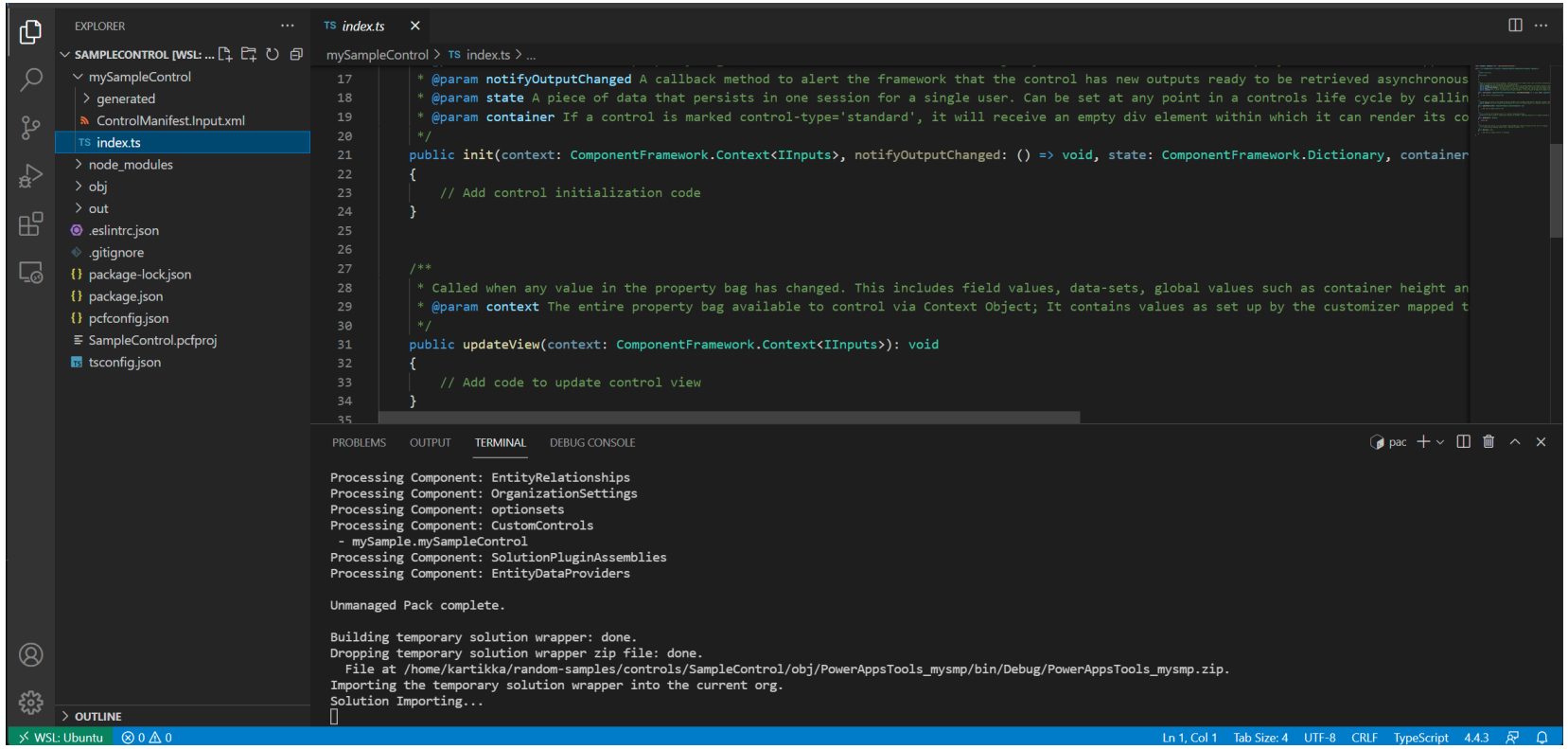
Ease of management for developer functions

Consistent CI/CD actions regardless of platform

Multi-Platform support for code-first developers and DevOps engineers



# Cross platform tooling support




- Visual Studio Code support
- Available on Windows, Linux, and macOS
- Including on WSL
- GitHub CodeSpaces

# Hands on lab

Import existing solution

Explore the solution assets

Make a small change to a table and use new column in an app

Install tooling

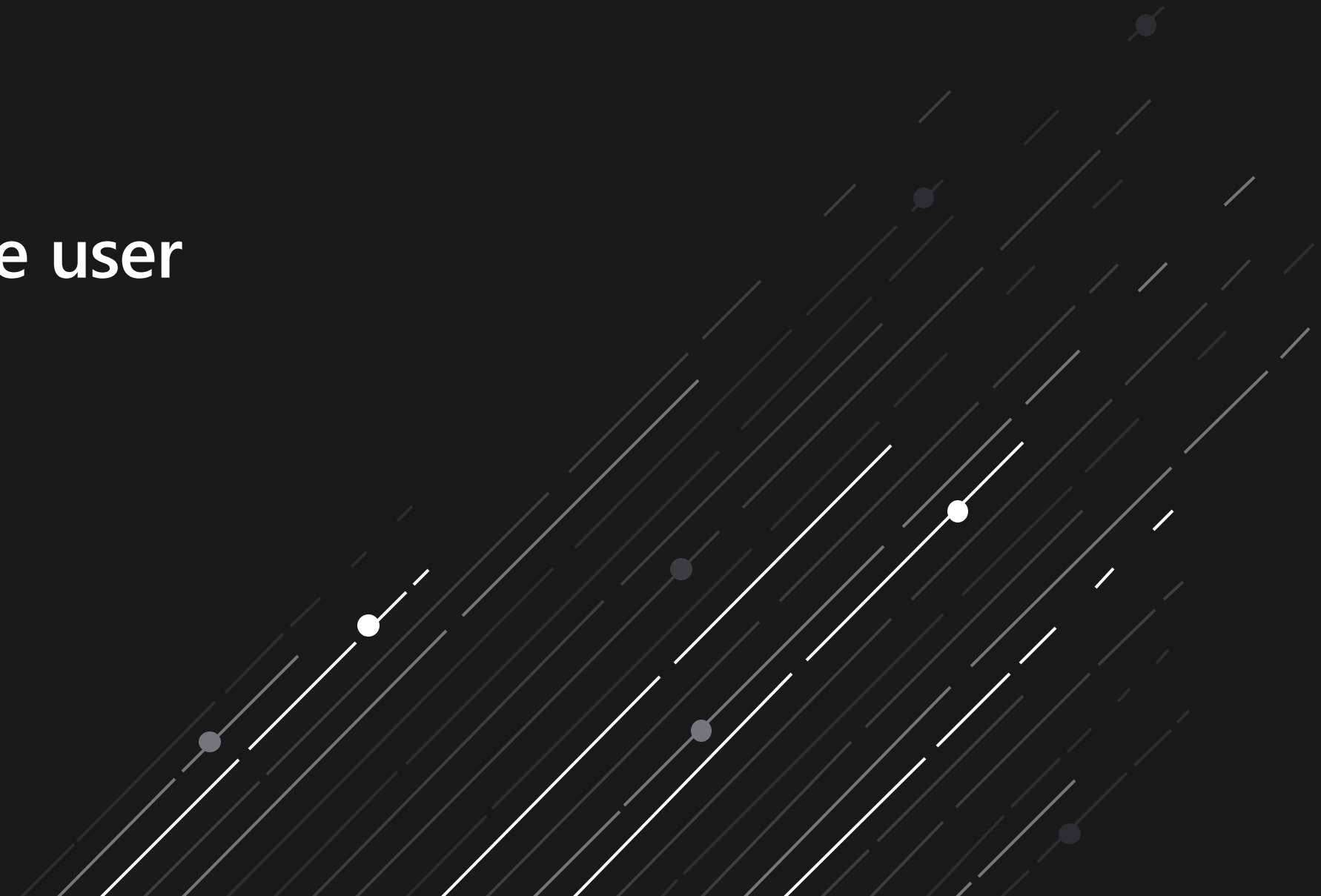
# Wrapping up

- Extend Power Platform services to fill gaps using the development skills you already have

- Capabilities like Power Apps code components and custom connectors make what you build easy to consume by fusion teams

Microsoft Power Platform

# Extending the user experience



# Types of components

## Canvas Components

Targeted for canvas app makers

Built in Power Apps Studio using Power Fx

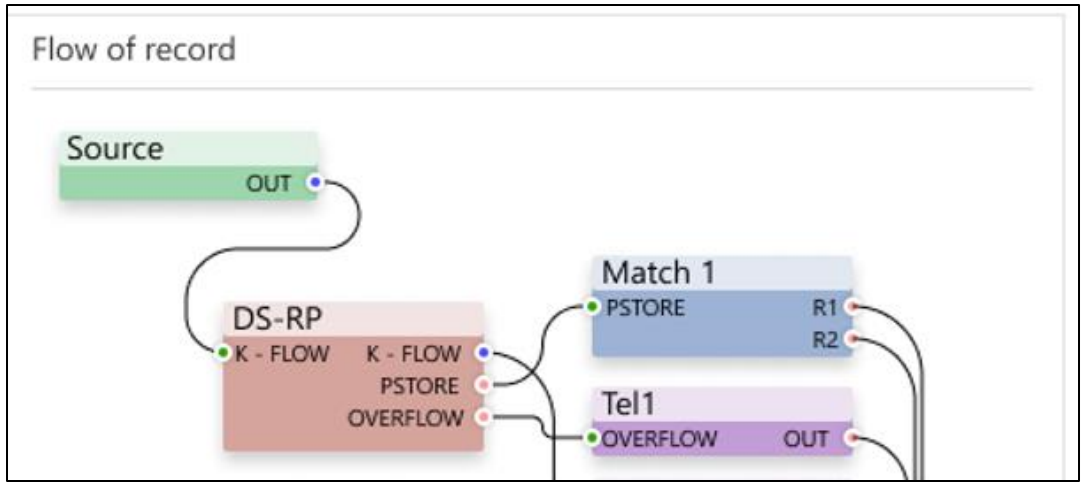
Works in canvas apps


## Power Apps Component Framework (PCF)

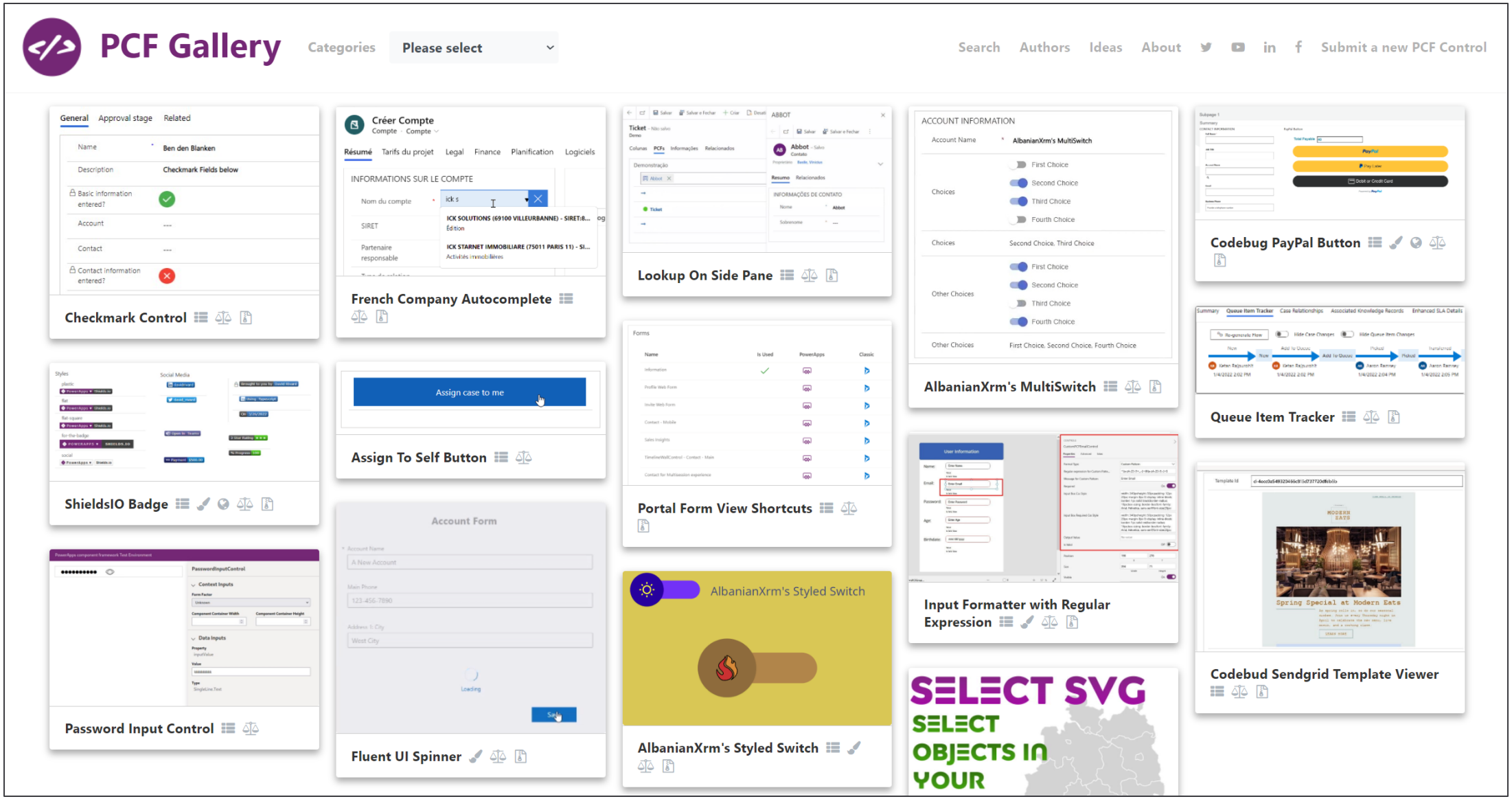
Targeted for professional developers to build

Built in TypeScript using pro tooling

Works in model-driven, canvas and portal apps

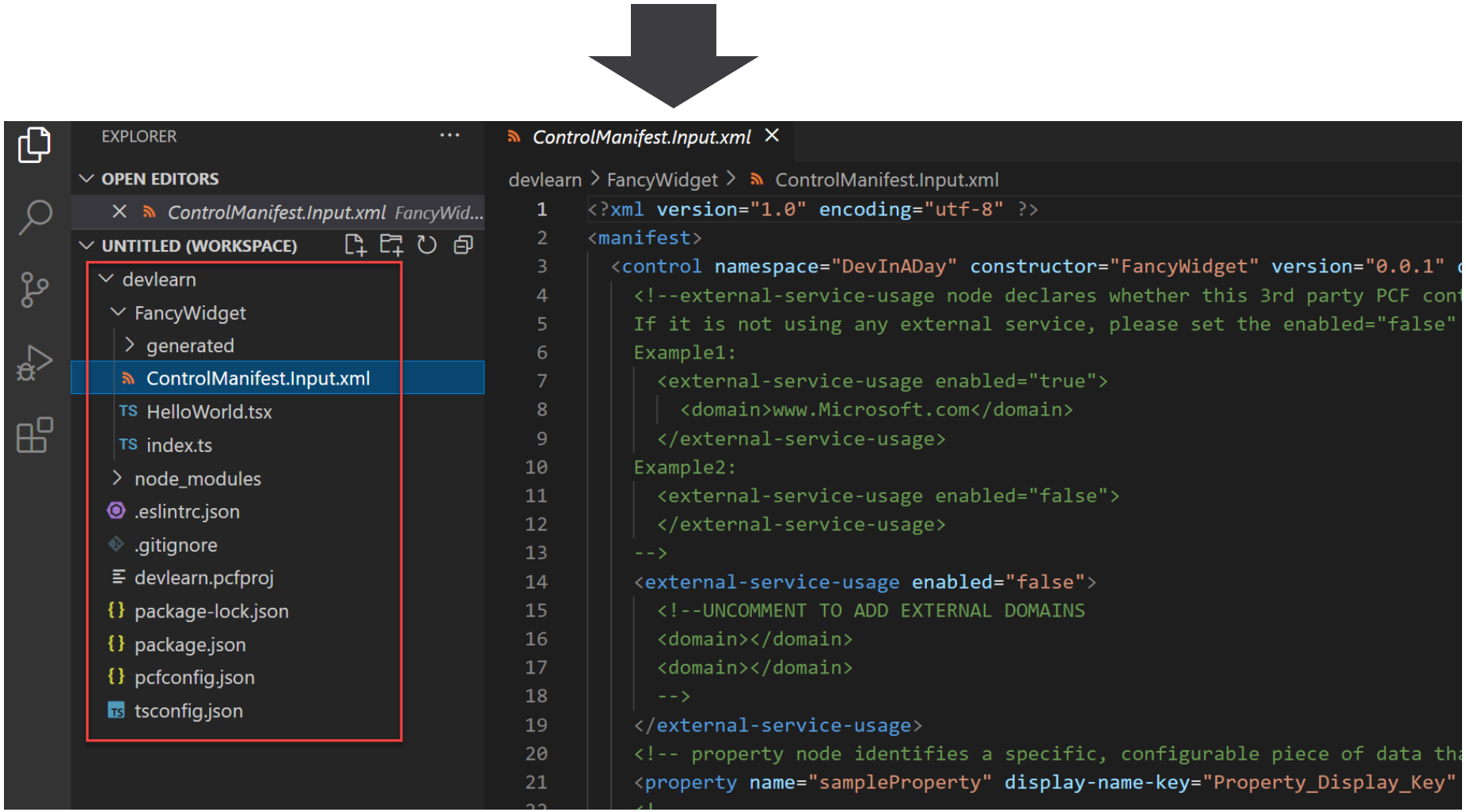


# Community PCF Gallery – https://pcf.gallery



# Scaffold code component using Power Platform CLI

pac pcf init -n FancyWidget -ns DevInADay -t field -fw react -npm



# Manifest defines the code component

Name, version and type

devlearn > FancyWidget > ControlManifest.Input.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<manifest>
  <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1"
    display-name-key="FancyWidget" description-key="FancyWidget description" control-type="virtual" >

    <external-service-usage enabled="false"></external-service-usage>
    <!-- property node identifies a specific, configurable piece of data that the control expects from
    <property name="sampleProperty" display-name-key="Property_Display_Key"
      description-key="Property_Desc_Key" of-type="SingleLine.Text" usage="bound" required="true" />

    <resources>
      <code path="index.ts" order="1"/>
      <platform-library name="React" version="16.8.6" />
      <platform-library name="Fluent" version="8.29.0" />
      <css path="css/FancyWidget.css" order="1" />
      <resx path="strings/FancyWidget.1033.resx" version="1.0.0" />
    </resources>

    <feature-usage>
      <uses-feature name="Device.captureAudio" required="true" />
      <uses-feature name="Device.captureImage" required="true" />
      <uses-feature name="Device.captureVideo" required="true" />
      <uses-feature name="Device.getBarcodeValue" required="true" />
      <uses-feature name="Device.getCurrentPosition" required="true" />
      <uses-feature name="Device.pickFile" required="true" />
      <uses-feature name="Utility" required="true" />
      <uses-feature name="WebAPI" required="true" />
    </feature-usage>
  </control>
</manifest>
```

# Manifest defines the code component - Properties

Properties exposed to hosting app

devlearn > FancyWidget > ControlManifest.Input.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<manifest>
  <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1"
    display-name-key="FancyWidget" description-key="FancyWidget description" control-type="virtual" >

    <external-service-usage enabled="false"></external-service-usage>
    <!-- property node identifies a specific, configurable piece of data that the control expects from
    <property name="sampleProperty" display-name-key="Property_Display_Key"
      description-key="Property_Desc_Key" of-type="SingleLine.Text" usage="bound" required="true" />

    <resources>
      <code path="index.ts" order="1"/>
      <platform-library name="React" version="16.8.6" />
      <platform-library name="Fluent" version="8.29.0" />
      <css path="css/FancyWidget.css" order="1" />
      <resx path="strings/FancyWidget.1033.resx" version="1.0.0" />
    </resources>

    <feature-usage>
      <uses-feature name="Device.captureAudio" required="true" />
      <uses-feature name="Device.captureImage" required="true" />
      <uses-feature name="Device.captureVideo" required="true" />
      <uses-feature name="Device.getBarcodeValue" required="true" />
      <uses-feature name="Device.getCurrentPosition" required="true" />
      <uses-feature name="Device.pickFile" required="true" />
      <uses-feature name="Utility" required="true" />
      <uses-feature name="WebAPI" required="true" />
    </feature-usage>
  </control>
</manifest>
```

# Manifest defines the code component - Resources

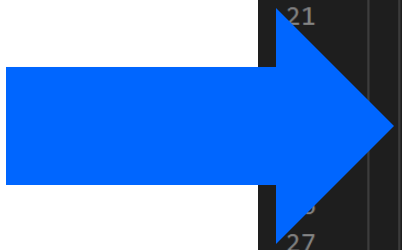
Resource files that the component needs

devlearn > FancyWidget > ControlManifest.Input.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<manifest>
  <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1"
    display-name-key="FancyWidget" description-key="FancyWidget description" control-type="virtual" >

    <external-service-usage enabled="false"></external-service-usage>
    <!-- property node identifies a specific, configurable piece of data that the control expects from
    <property name="sampleProperty" display-name-key="Property_Display_Key"
      description-key="Property_Desc_Key" of-type="SingleLine.Text" usage="bound" required="true" />

    <resources>
      <code path="index.ts" order="1"/>
      <platform-library name="React" version="16.8.6" />
      <platform-library name="Fluent" version="8.29.0" />
      <css path="css/FancyWidget.css" order="1" />
      <resx path="strings/FancyWidget.1033.resx" version="1.0.0" />
    </resources>

    <feature-usage>
      <uses-feature name="Device.captureAudio" required="true" />
      <uses-feature name="Device.captureImage" required="true" />
      <uses-feature name="Device.captureVideo" required="true" />
      <uses-feature name="Device.getBarcodeValue" required="true" />
      <uses-feature name="Device.getCurrentPosition" required="true" />
      <uses-feature name="Device.pickFile" required="true" />
      <uses-feature name="Utility" required="true" />
      <uses-feature name="WebAPI" required="true" />
    </feature-usage>
  </control>
</manifest>
```

# Manifest defines the code component - Features

Enable component features



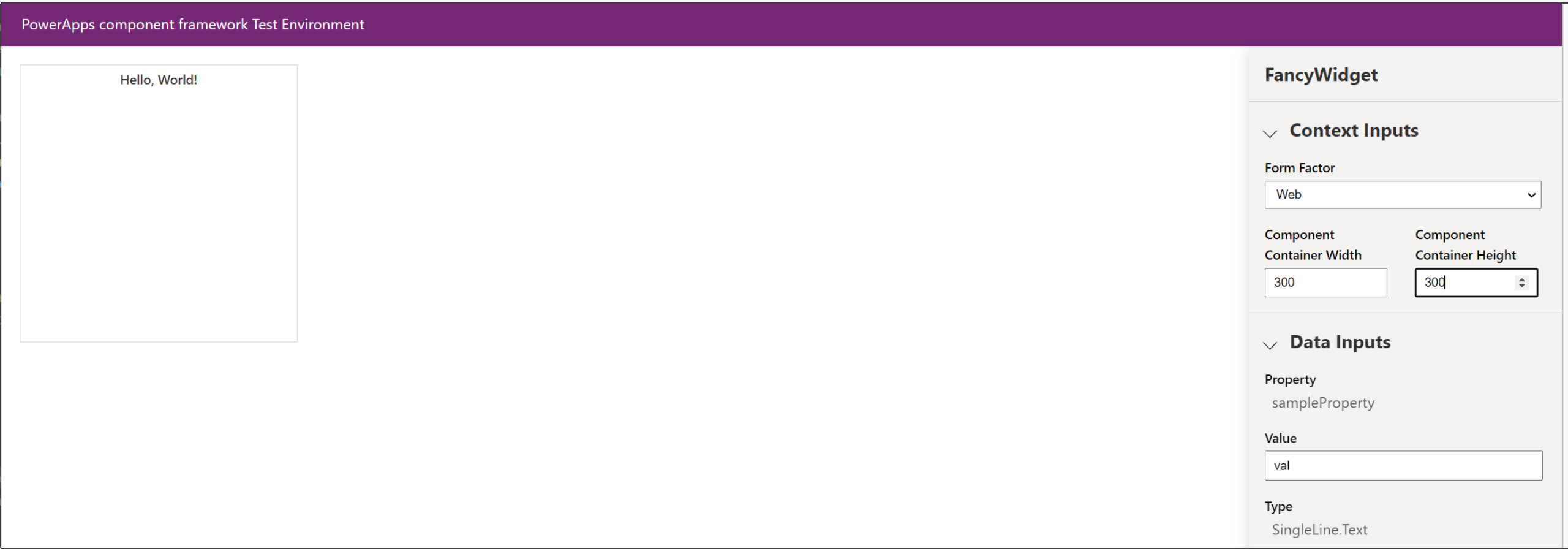
devlearn > FancyWidget > ControlManifest.Input.xml

```xml
<?xml version="1.0" encoding="utf-8" ?>
<manifest>
  <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1"
    display-name-key="FancyWidget" description-key="FancyWidget description" control-type="virtual" >

    <external-service-usage enabled="false"></external-service-usage>
    <!-- property node identifies a specific, configurable piece of data that the control expects from
    <property name="sampleProperty" display-name-key="Property_Display_Key"
      description-key="Property_Desc_Key" of-type="SingleLine.Text" usage="bound" required="true" />

    <resources>
      <code path="index.ts" order="1"/>
      <platform-library name="React" version="16.8.6" />
      <platform-library name="Fluent" version="8.29.0" />
      <css path="css/FancyWidget.css" order="1" />
      <resx path="strings/FancyWidget.1033.resx" version="1.0.0" />
    </resources>

    <feature-usage>
      <uses-feature name="Device.captureAudio" required="true" />
      <uses-feature name="Device.captureImage" required="true" />
      <uses-feature name="Device.captureVideo" required="true" />
      <uses-feature name="Device.getBarcodeValue" required="true" />
      <uses-feature name="Device.getCurrentPosition" required="true" />
      <uses-feature name="Device.pickFile" required="true" />
      <uses-feature name="Utility" required="true" />
      <uses-feature name="WebAPI" required="true" />
    </feature-usage>
  </control>
</manifest>
```

# Component implementation

| Method | Description |
| --- | --- |
| Init | Used to initialize the control instance. Controls can kick off remote server calls and other initialization actions here. |
| updateView | Called when any value in the property bag has changed.<br><br>This method should be implemented to update the values displayed in the component. |
| getOutputs | Called by the framework prior to a control receiving new data.<br><br>The getOutputs method returns values that have the changes made by the user. For a field component, this would typically be the new value for the component. |
| destroy | Called when the control is to be removed from the DOM tree. Controls should use this call for cleanup. |

# Basic testing

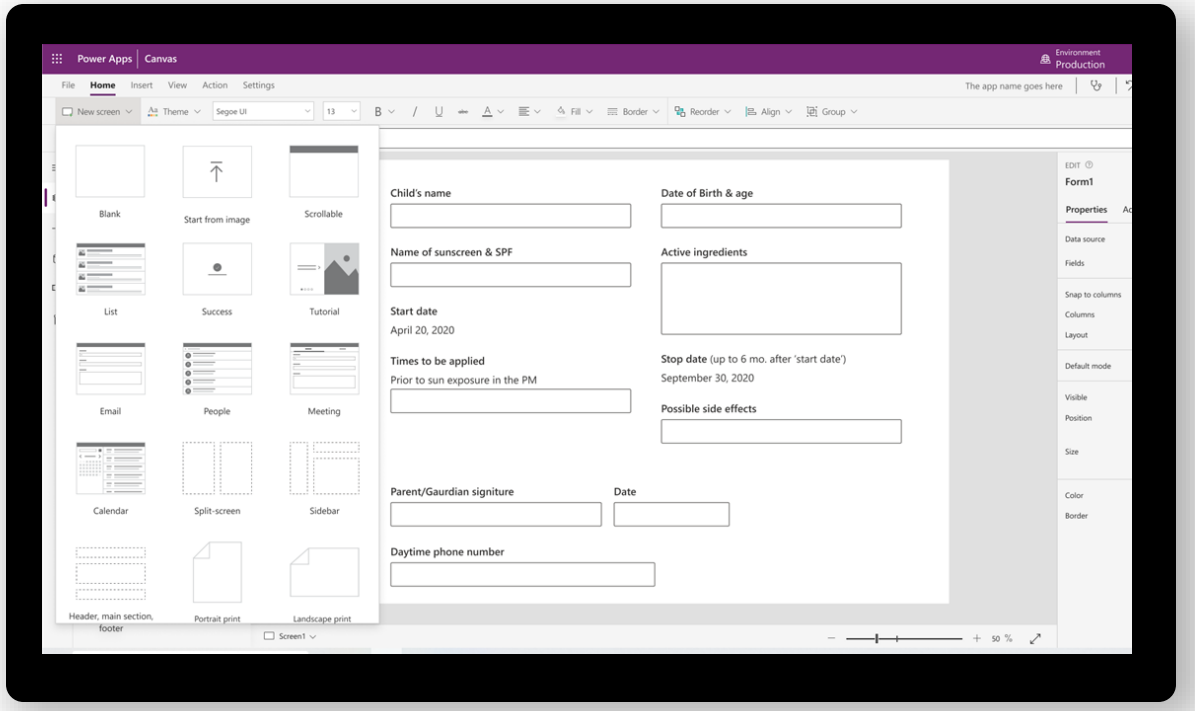


# Innovate at speed with express design

Jumpstart the app development process and empower anyone to easily create new apps

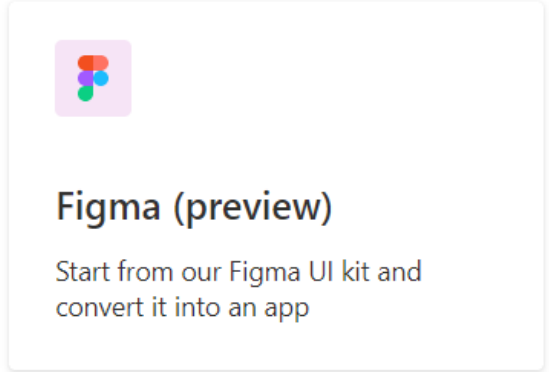
Start from visual design such as a paper form, a Figma design, or even a whiteboard sketch

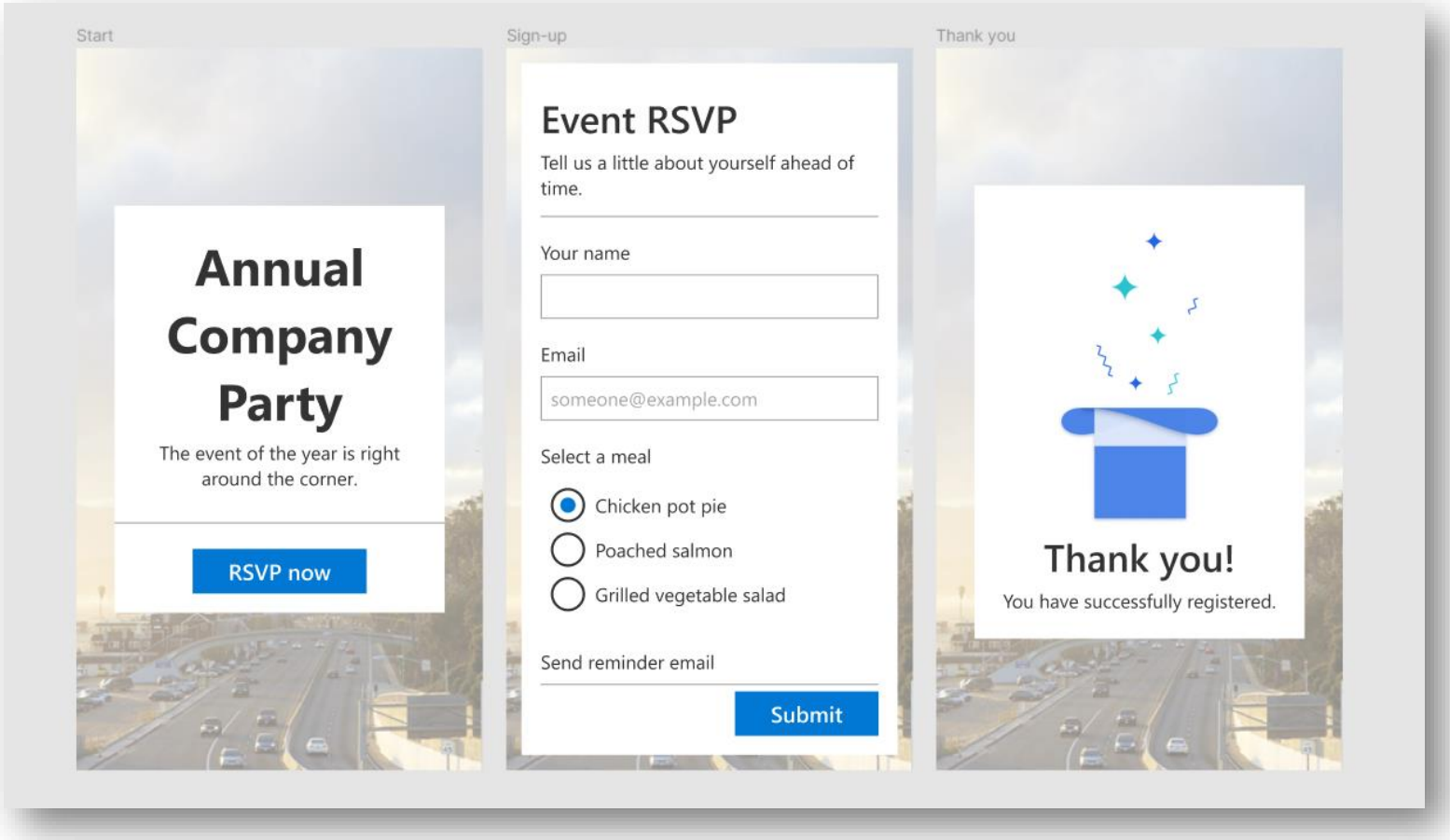
Build an app in a few steps by uploading your content and connecting to your data



# Figma to app

1. Duplicate the **Create Apps from Figma UI Kit**
2. Design your app and customize the styles
3. Convert your design using **Figma to App**



# Demo

Build a code component to allow drag and drop prioritization of a list

Test component in app and prepare for how citizen dev would consume component

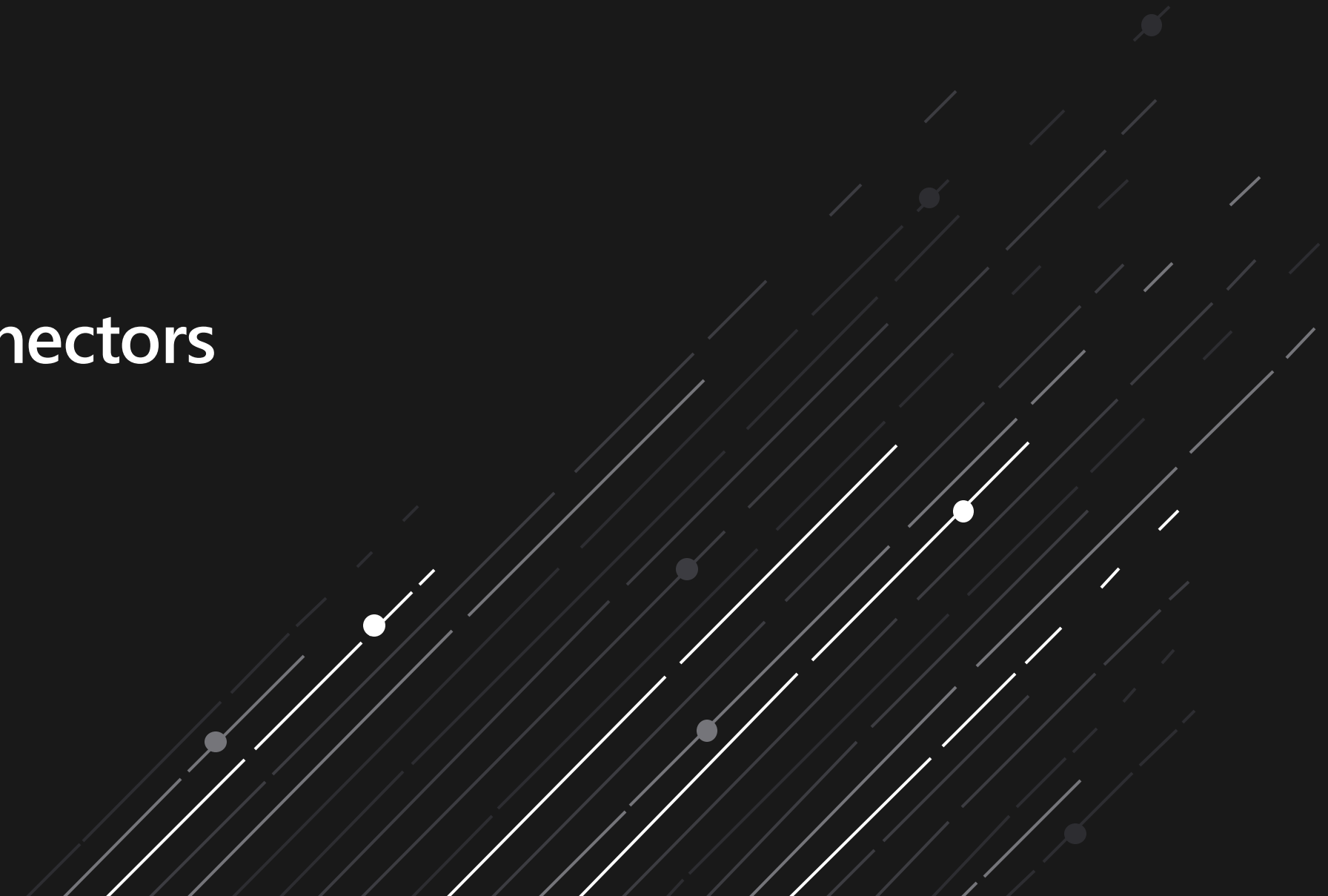
# Wrapping up

- Power Apps component framework gives developers the ability to extend the visualizations in apps

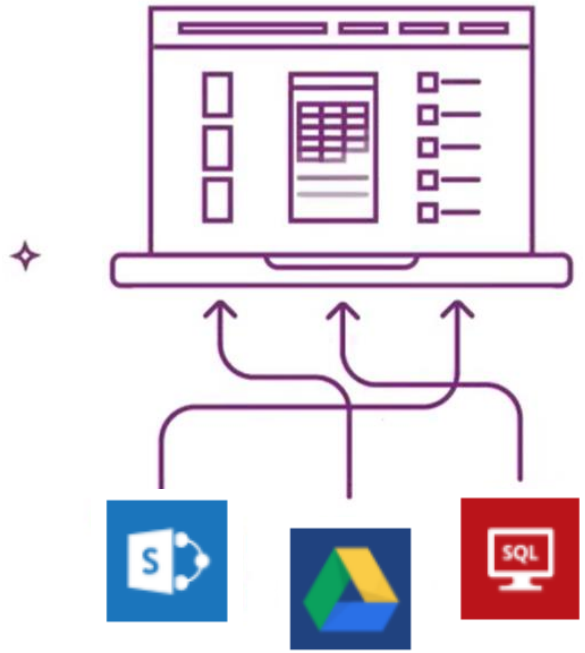
- Build new visuals or wrap existing visuals for consumption by makers

Microsoft Power Platform

# Custom Connectors

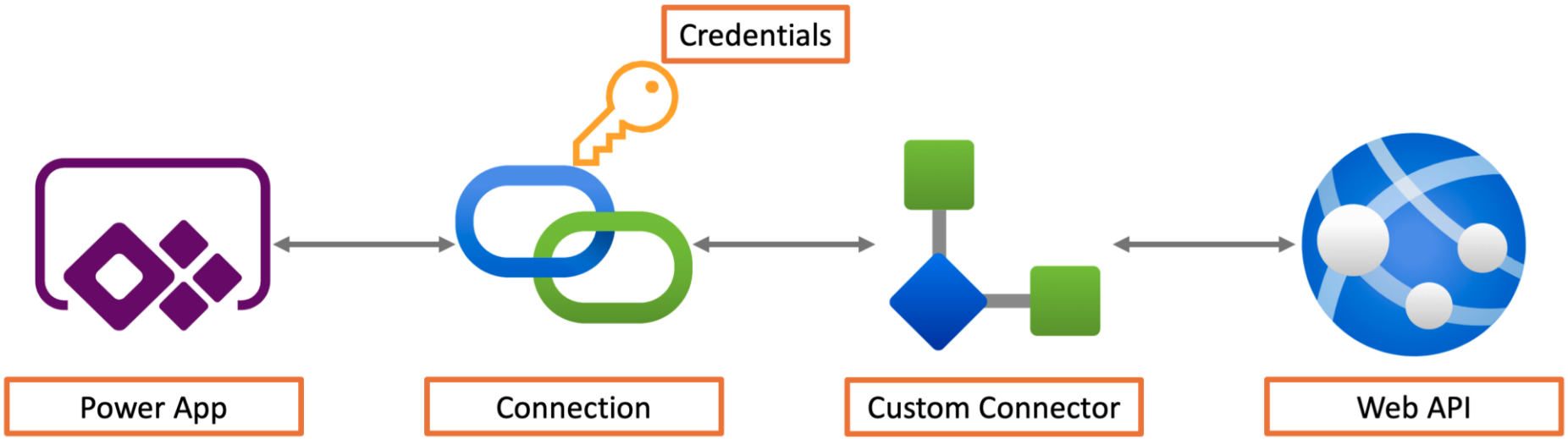


A **connector** is a wrapper around an API that allows the underlying service to talk to **Power Automate** and **Power Apps**




# Connector vs Connection

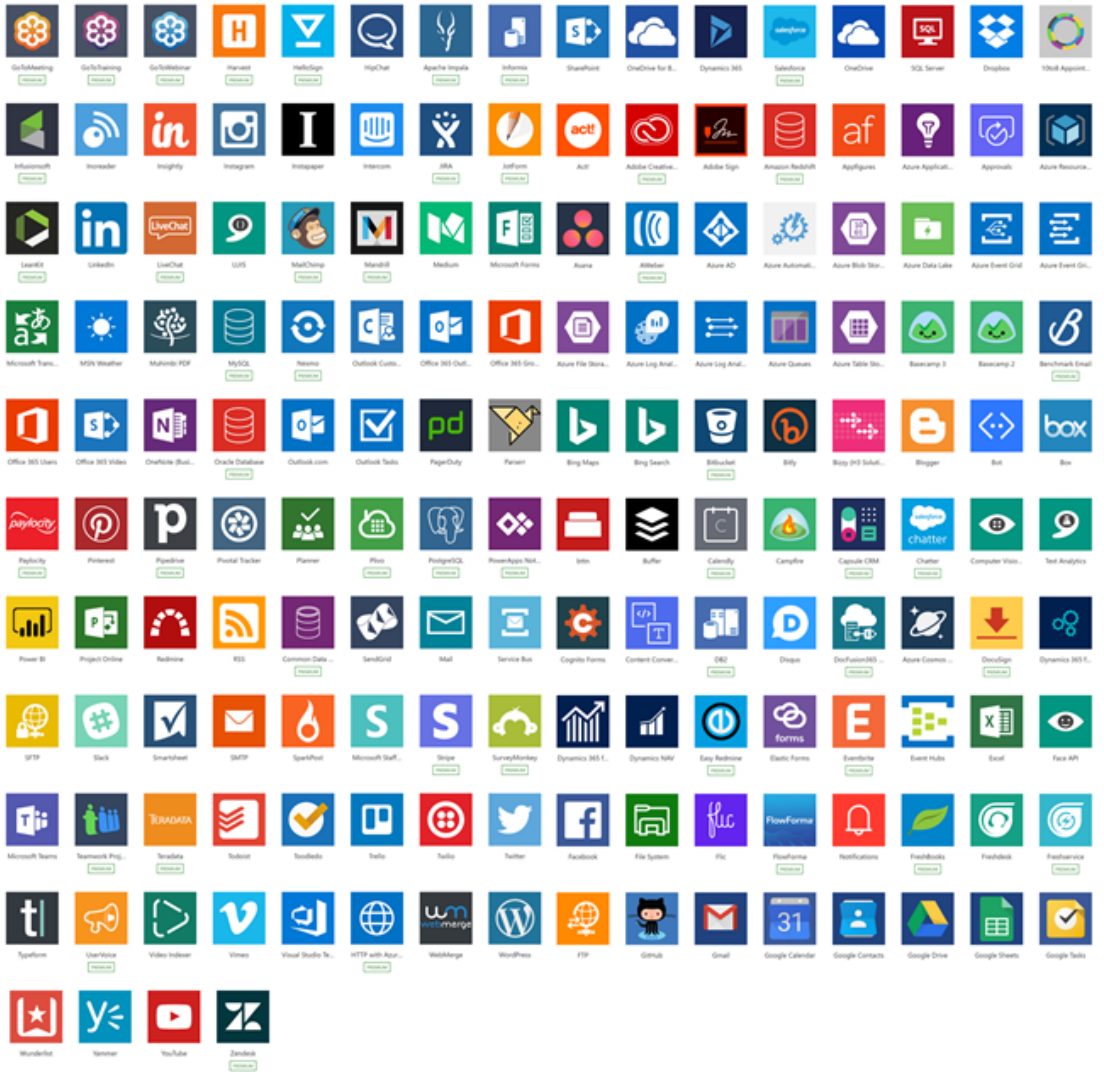
- Connector knows about the Web API's host and operation details
- Connection knows the credentials and has a reference to the connector to facilitate communication with the Web API



# Cloud and on-premises connectivity

- Built-in connectivity to cloud services, files, databases, web APIs, etc. using public connectors
- Seamless hybrid connectivity to on-premises systems via the On-Premises Data Gateway
- Connect to the service with your own security credentials
- Open source
  - https://github.com/microsoft/PowerPlatformConnectors

- With custom connectors, you can extend the reach for your organization



Hundreds of connectors today

# Develop a custom connector

## Build or select API

- REST API
- Public or private
- 3rd party or custom built
- Build with Azure
  - Functions
  - Web Apps
  - API Apps

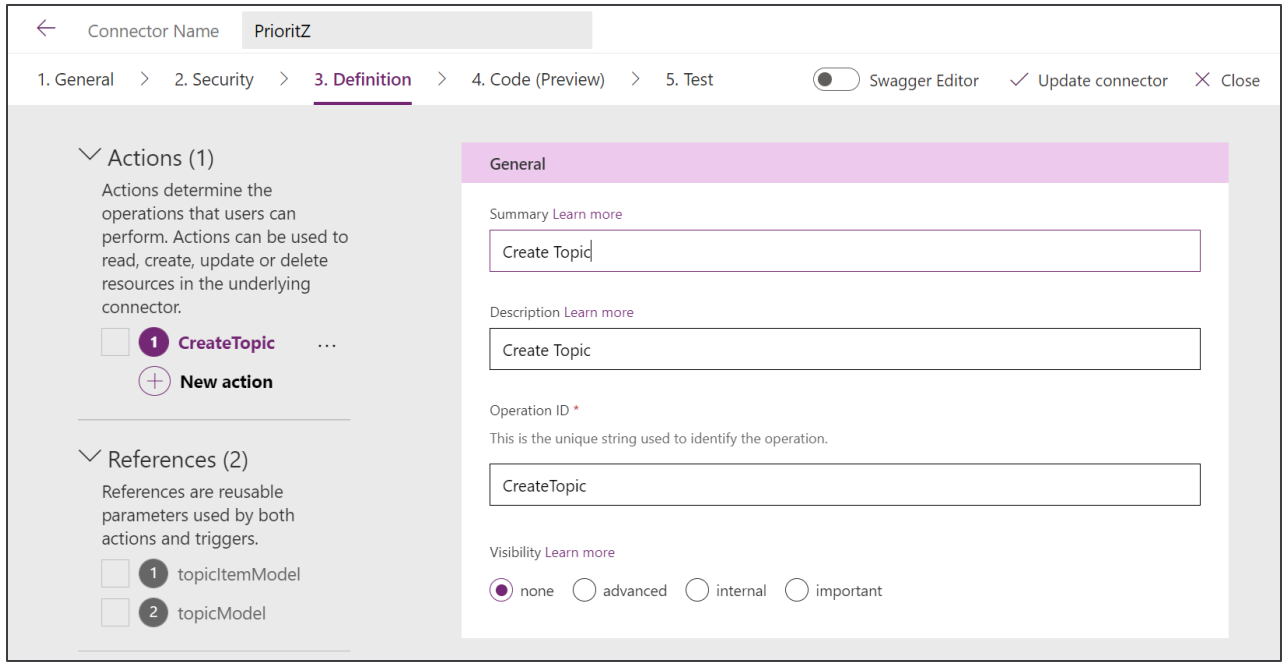
## Secure

- Anonymous
- Basic
- API Key
- OAuth 2.0 (Cloud only)
- Windows (On-premises)

## Describe

- OpenAPI (Swagger)
- Postman Collection
- From Azure Service
  - API Management
  - Functions
  - Logic Apps
- Start from scratch
  - Sample request/response
  - Tune the definitions

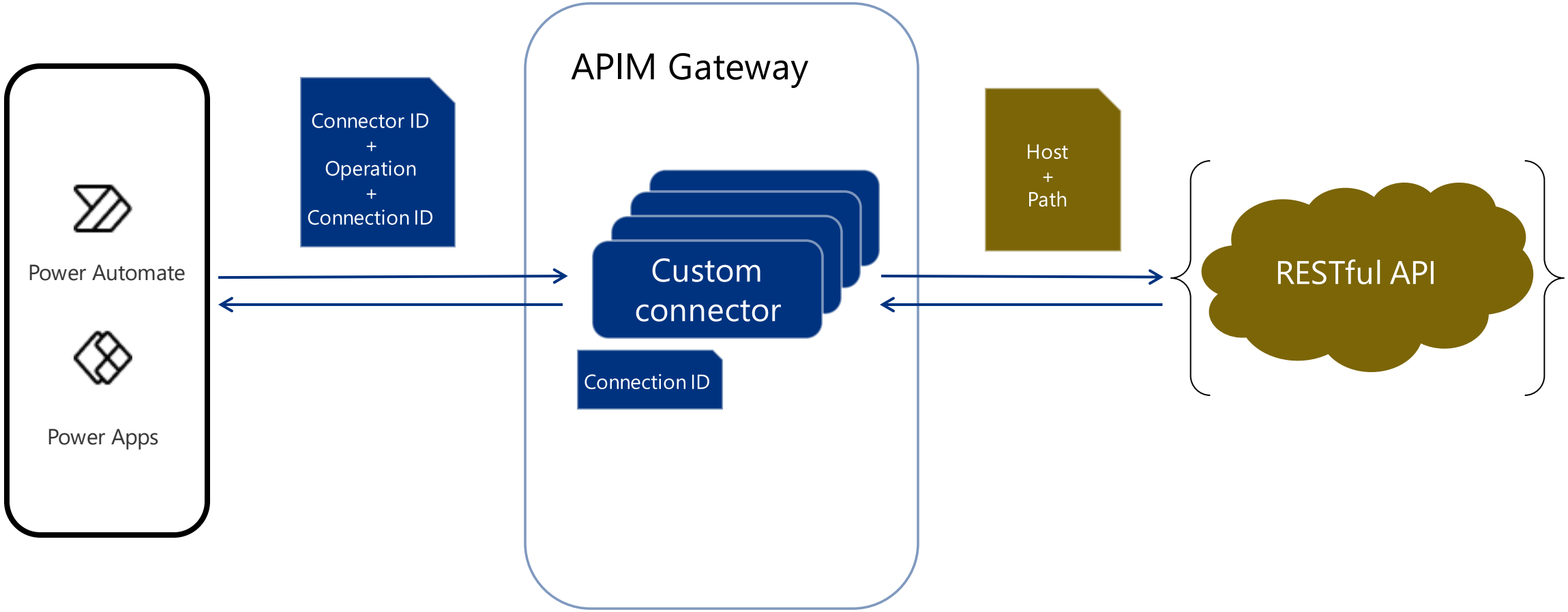
# Edit in the designer or via Swagger editor




```yaml
swagger: '2.0'
info: {title: OpenAPI Document on Azure Functions, version: 1.0.0}
host: proritzdy.azurewebsites.net
basePath: /api
schemes: [https]
consumes: []
produces: []
paths:
  /CreateTopic:
    get:
      tags: [name]
      operationId: CreateTopic
      consumes: [application/json]
      produces: [text/plain]
      parameters:
      - {in: query, name: name, description: The **Name** parameter, required: true,
        type: string}
      - in: body
        name: body
        schema: {$ref: '#/definitions/topicModel'}
      responses:
        '200':
          description: The OK response
          schema: {type: string}
      summary: Create Topic
      description: Create Topic
definitions:
  topicItemModel:
    type: object
    properties:
      choice: {type: string}
      photo: {format: base64, type: string}
  topicModel:
    type: object
    properties:
```

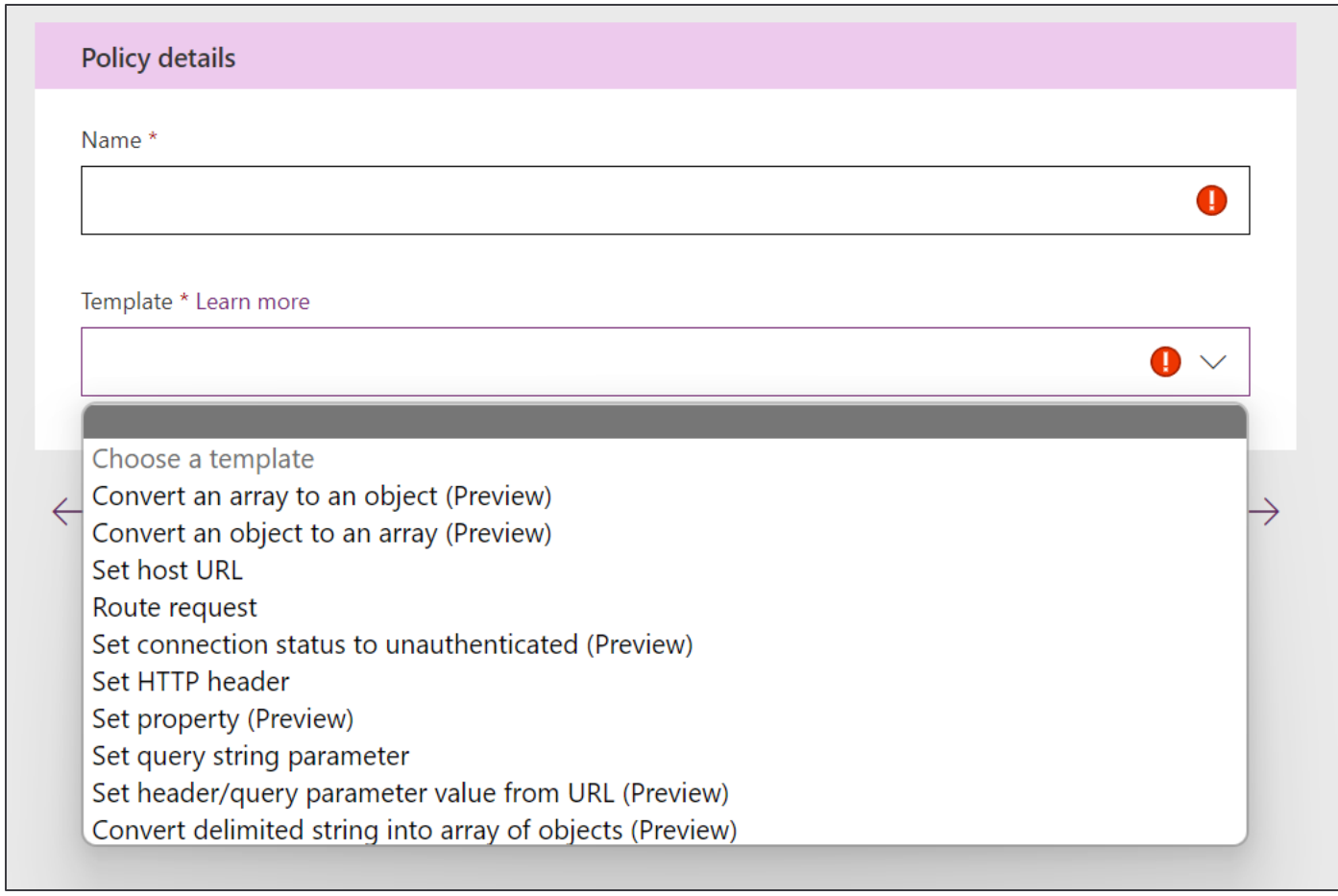
You can also use the CLI (paconn) to manage the definitions

# Connector architecture (no authentication)



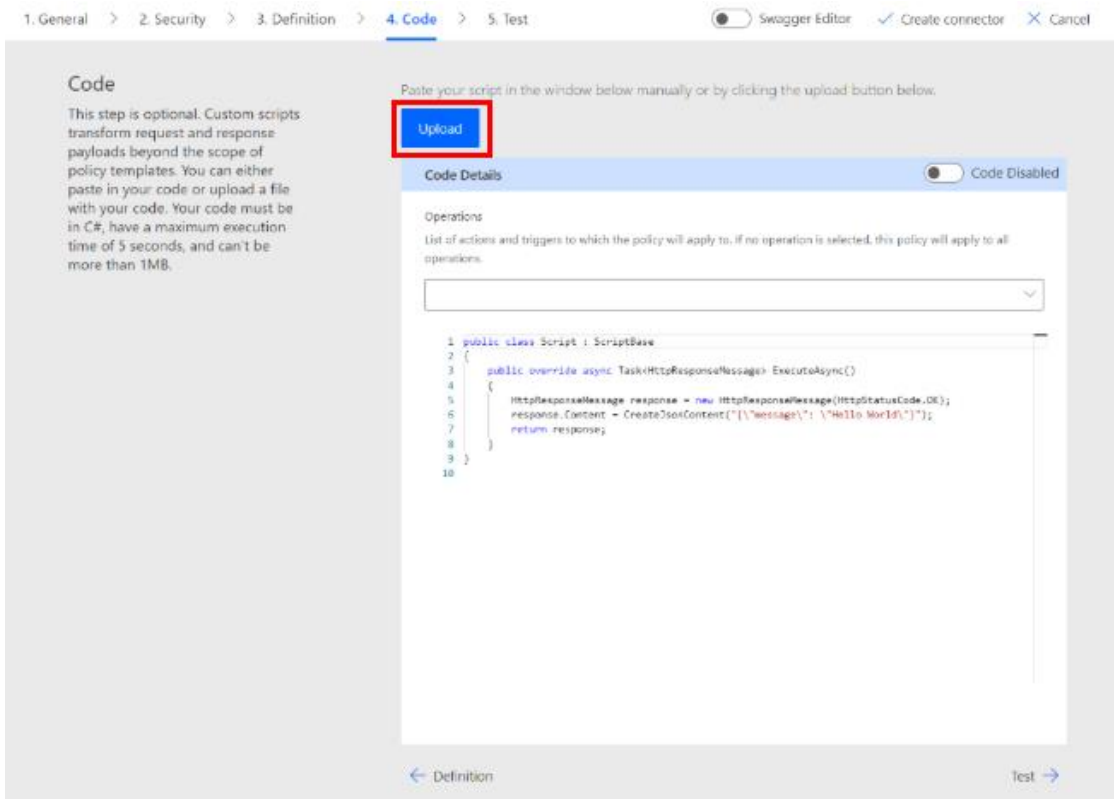
# Use policy templates to modify behavior

- Set host URLs
- Route requests
- Set HTTP header values
- Set query string parameters



# Use custom code for full control

- Custom code transforms request and response payloads beyond the scope of existing policy templates
- Transformations include sending external requests to fetch additional data
- Takes precedence over the codeless definition



# Making a connector public

## Develop

Build, secure and describe your API

Import from Open API, Postman, or build from scratch

## Test

Validate your API calls

Use in Power Automate and Power Apps

## Share

Allow other users within your organization to use your connector

## Certify

Share your connector with all users of Power Automate and Power Apps

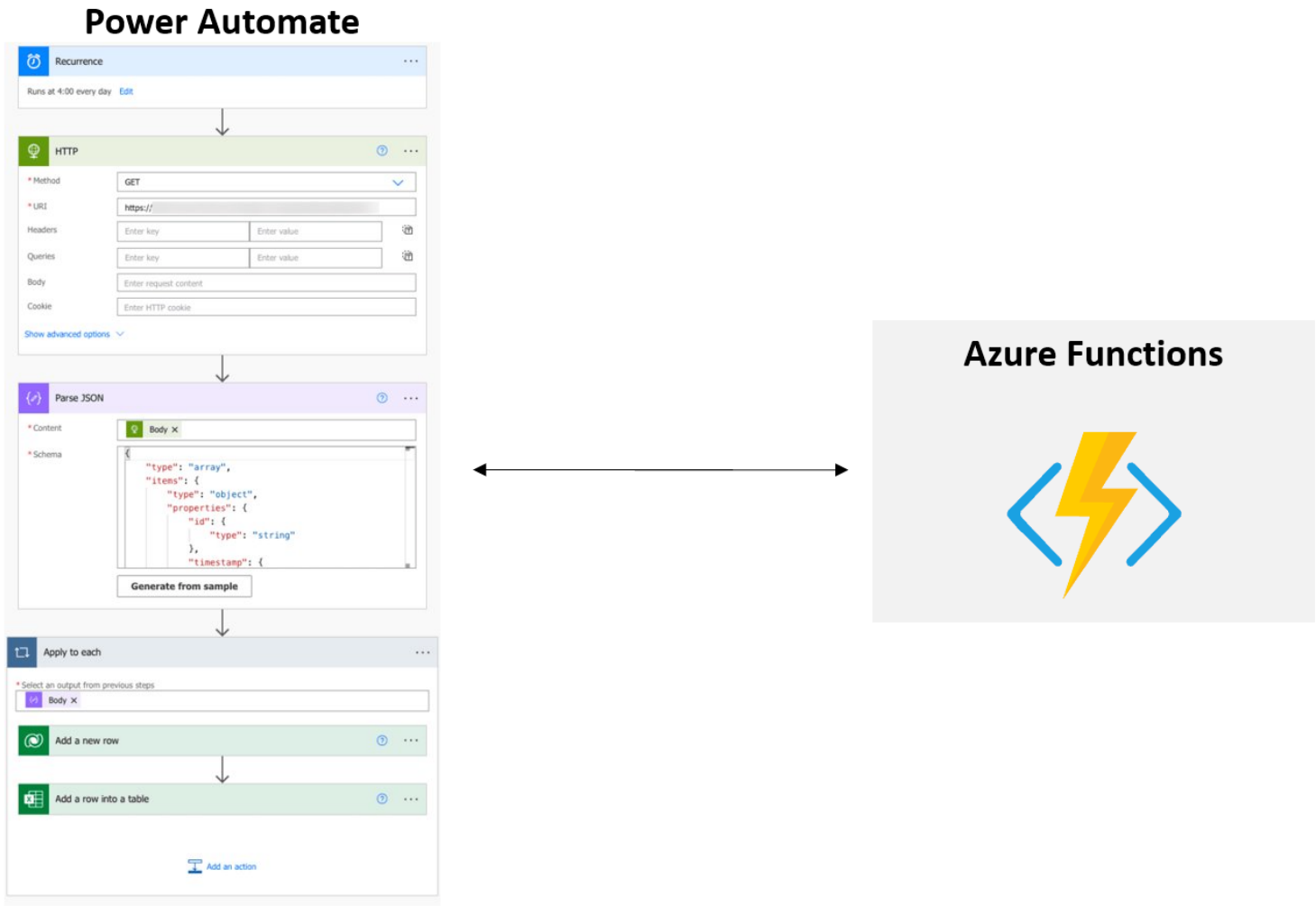
Only for Public Connectors

# Two certification paths

- Independent publishers - Does not own the underlying service behind the connector.
  - *An example is Jon Smith, a member of the public, who can submit the HubSpot connector for certification.*

- Verified publishers - Owns the underlying service behind their connector, among other differences.
  - *An example is Adobe, who has certified the Adobe Sign connector.*

# Do You Have to Create a Connector?




https://aka.ms/power-azure-github-reporting

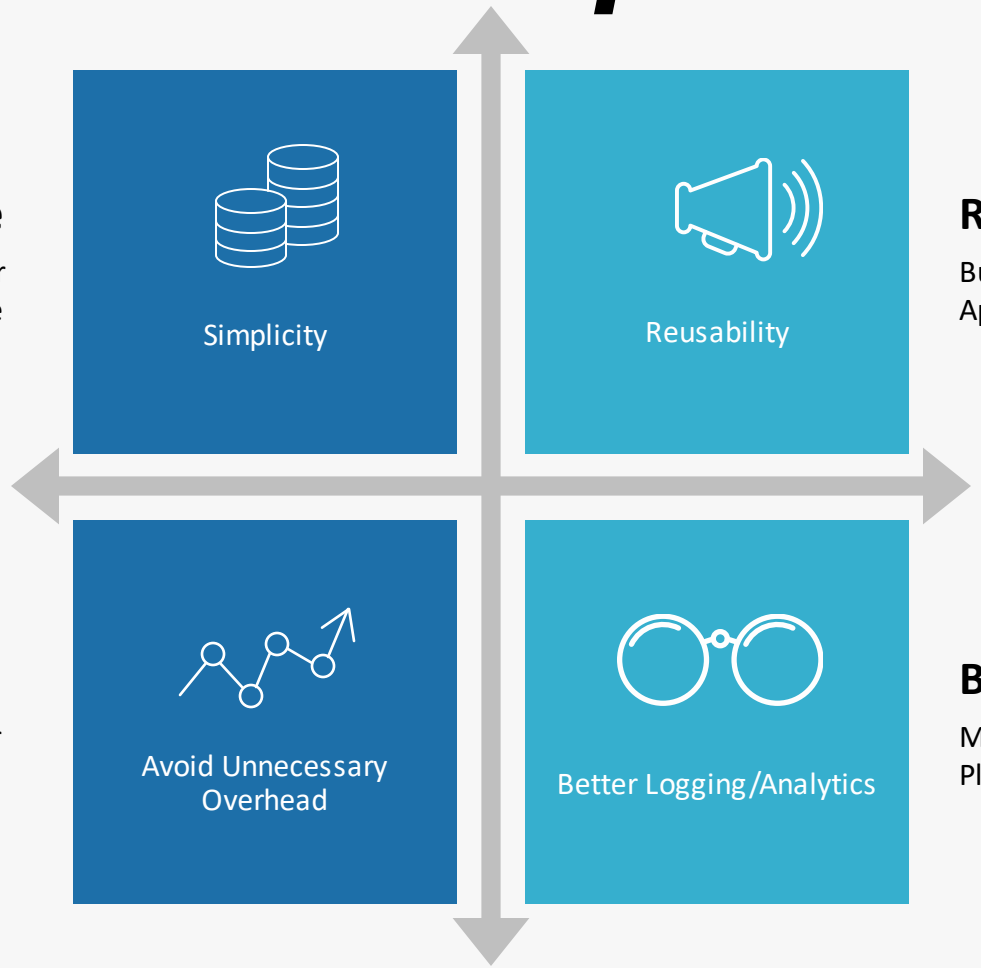
# Why use Custom Connectors over HTTP Requests?

### Keep it Simple

Makes it easier for users to call your service

### Avoid Unnecessary Overhead

Can't make direct HTTP calls in Power Apps so you must use Power Automate




### Reusability

Build it once and use it in multiple Power Apps and Flows

### Better Logging/Analytics

Monitor the connectors usage in Power Platform Admin Center

# Demo

Configure a custom connector for an existing API

Use custom code to transform API response

Use the connector from an app and flow

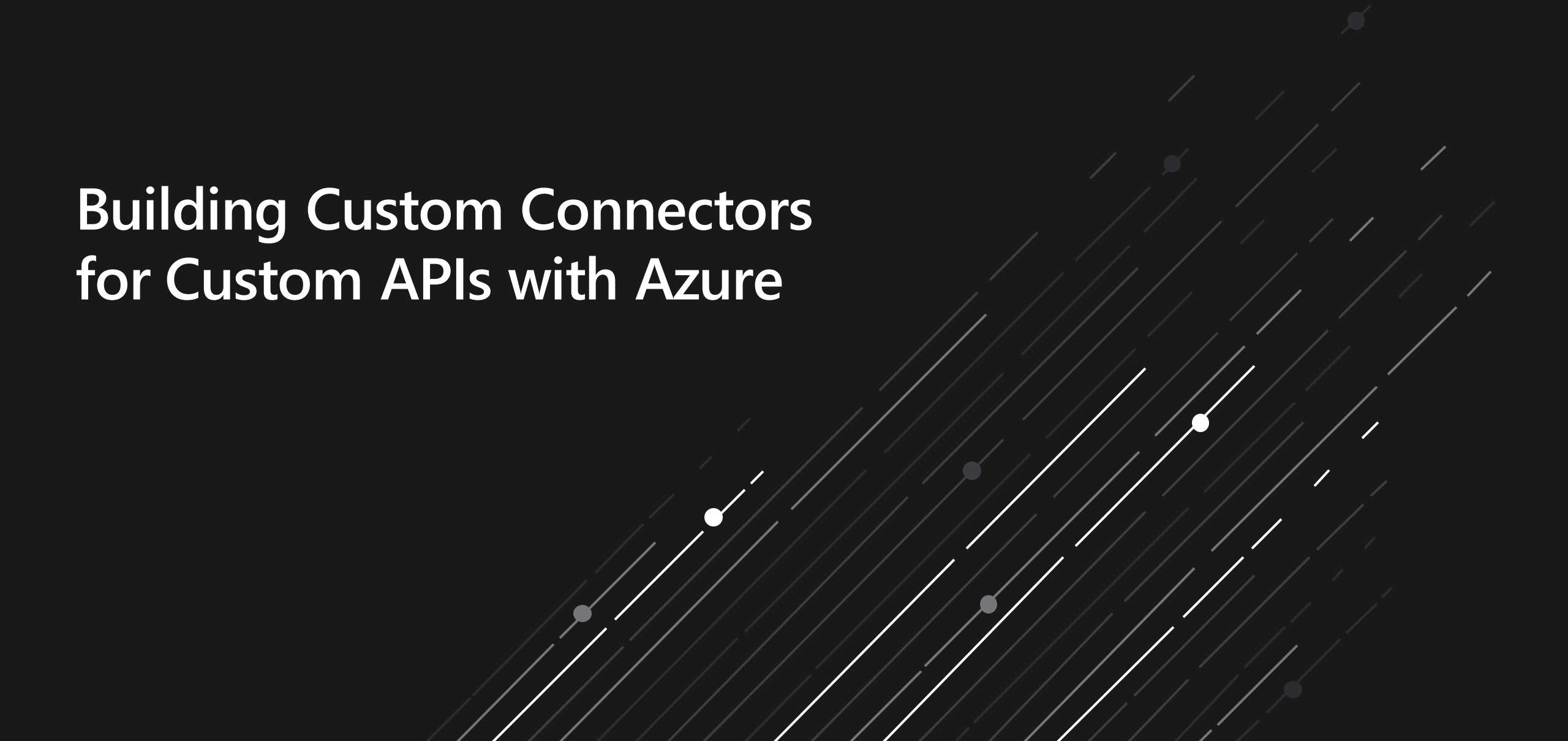
# Wrapping up

- Custom connectors make it easy for makers to consume existing APIs

- Policy templates and custom code allows transformation of APIs you don’t control

Microsoft Power Platform

# Building Custom Connectors for Custom APIs with Azure



# Use cases for custom APIs

- Shift complex logic from formulas to custom API
- Centralize reusable logic to custom API – allow use from multiple apps and flows
- Elevate permissions in custom API logic
- Integrate with existing APIs not compatible with custom connectors
- Scalability and performance

# Options for implementing and deploying custom APIs

- Azure Functions
- Azure App Services
- Azure API Management
- Any service that can host a RESTful API

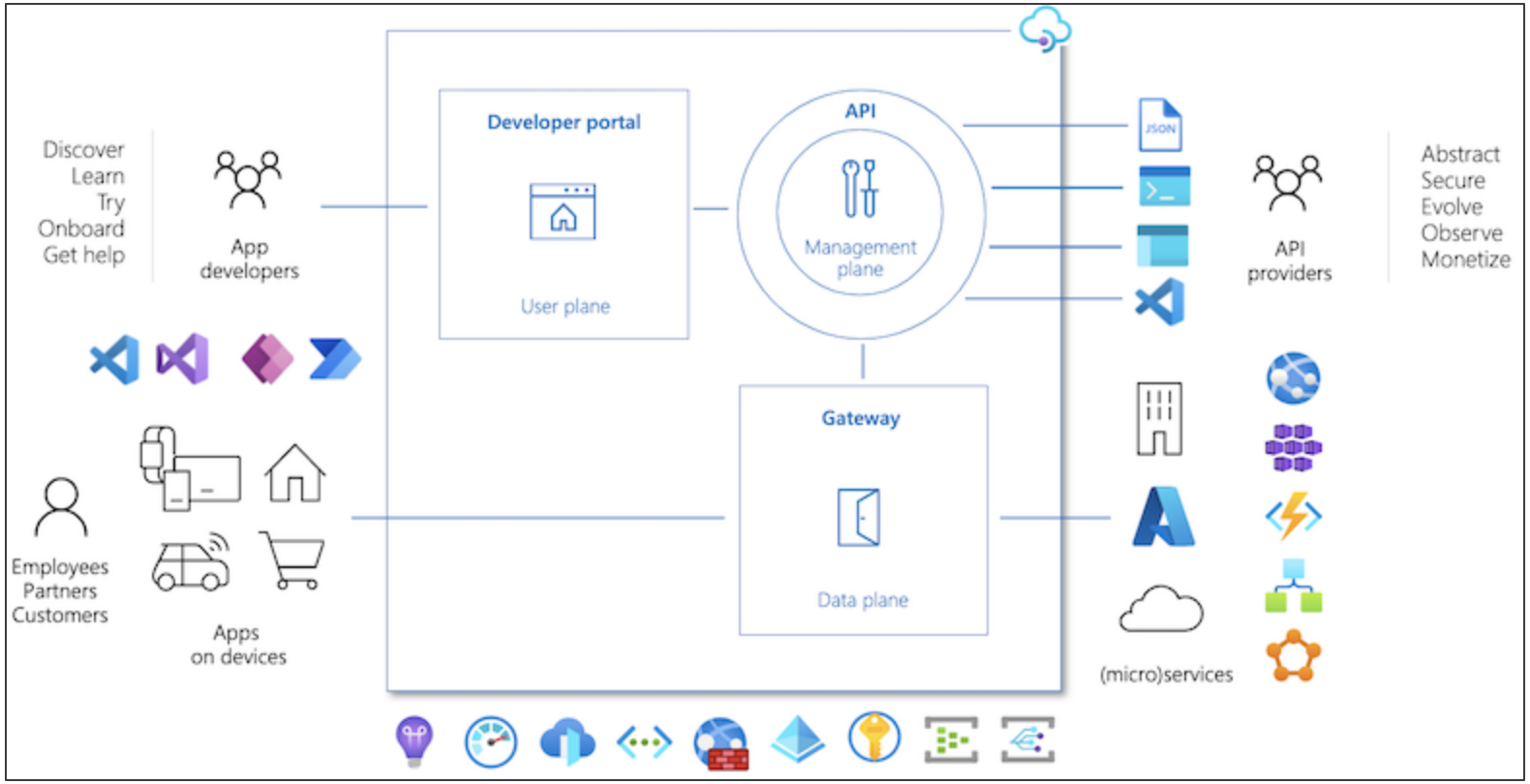
# Document your API for connector use

- Adding OpenAPI attributes automatically generates OpenAPI (Swagger) definitions and makes API ready for import

```
[FunctionName("CreateTopic")]
[OpenApiOperation(operationId: "CreateTopic", tags: new[] { "name" })]
[OpenApiResponseWithBody(statusCode: HttpStatusCode.OK, contentType: "text/plain", bodyType: typeof(string), Description = "The OK response")]
[OpenApiRequestBody(contentType: "application/json", bodyType: typeof(TopicModel))]
0 references
public async Task<IActionResult> Run(
    [HttpTrigger(AuthorizationLevel.Anonymous,  "post", Route = null)] HttpRequest req)
{
```

# Use Azure API Management

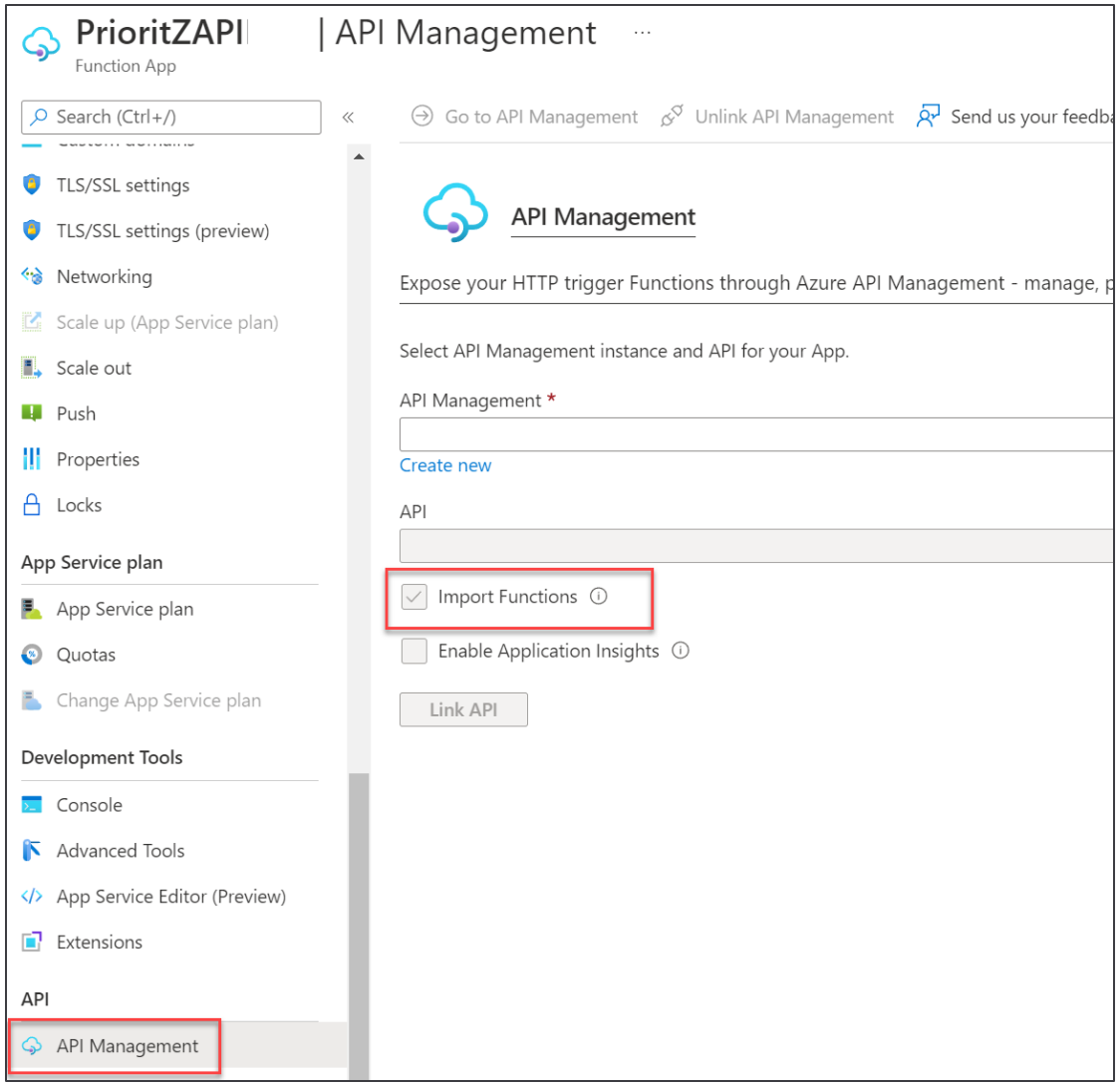
Azure API Management is a hybrid, multi-cloud management platform for APIs across all environments



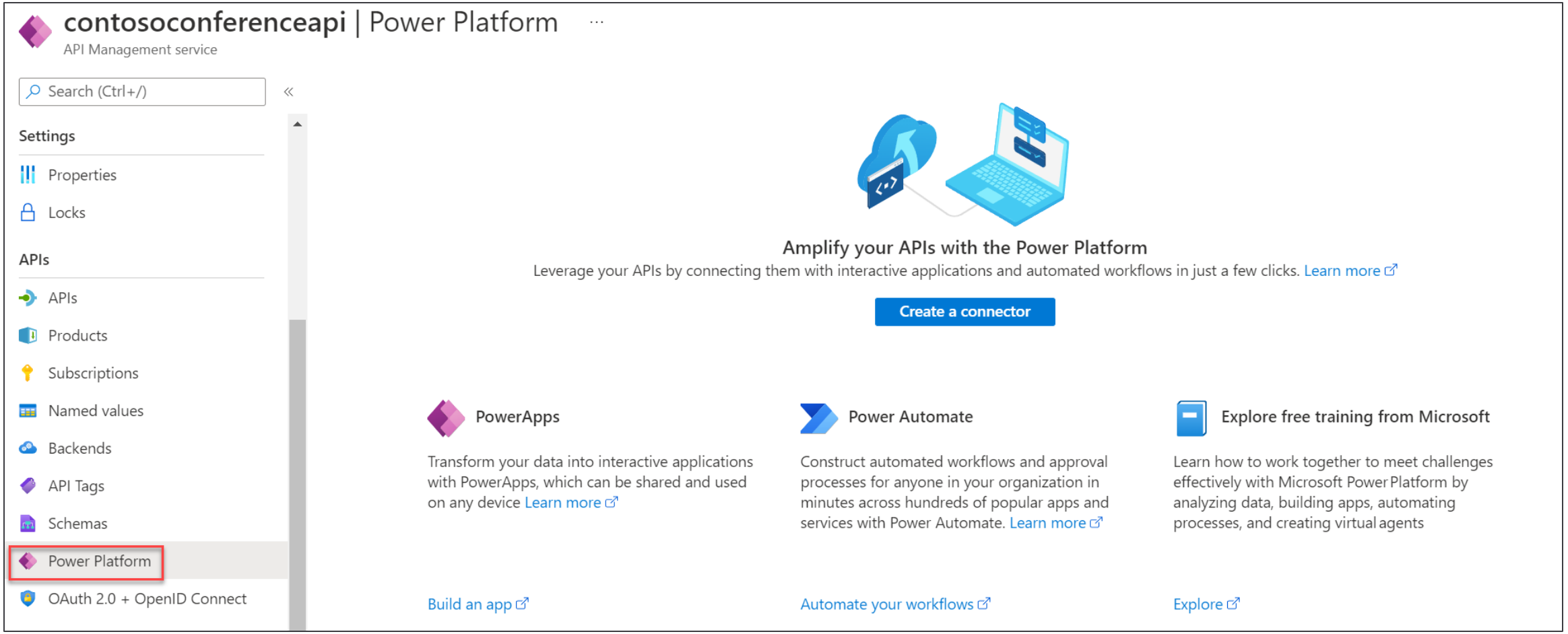
# Benefits of Azure API Management

- Consolidate API management
- Integration to create custom connector
- Subscription authentication and key management
- Power Apps hosted in Teams can only use custom connectors backed by API Management

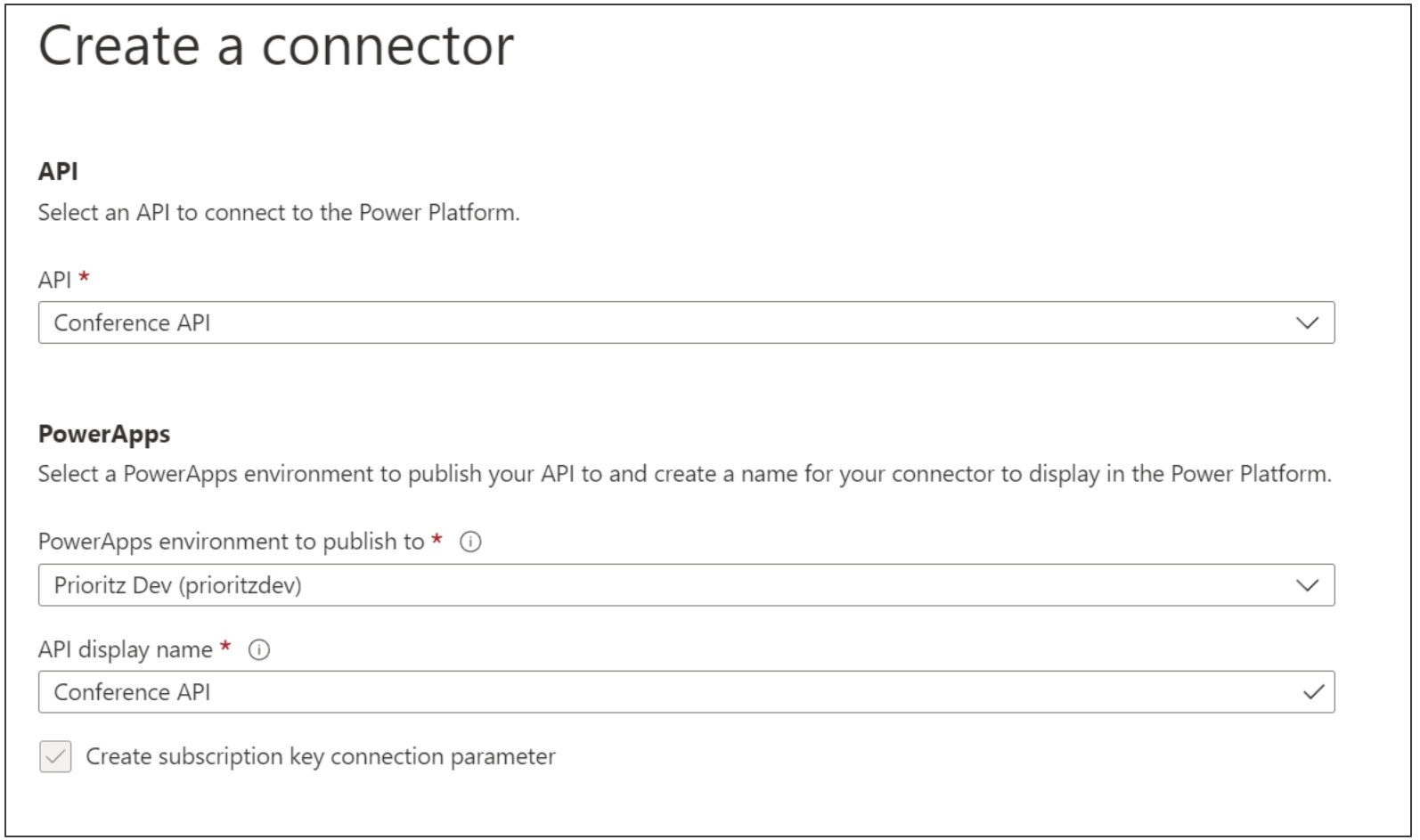
# Import functions into Azure API Management



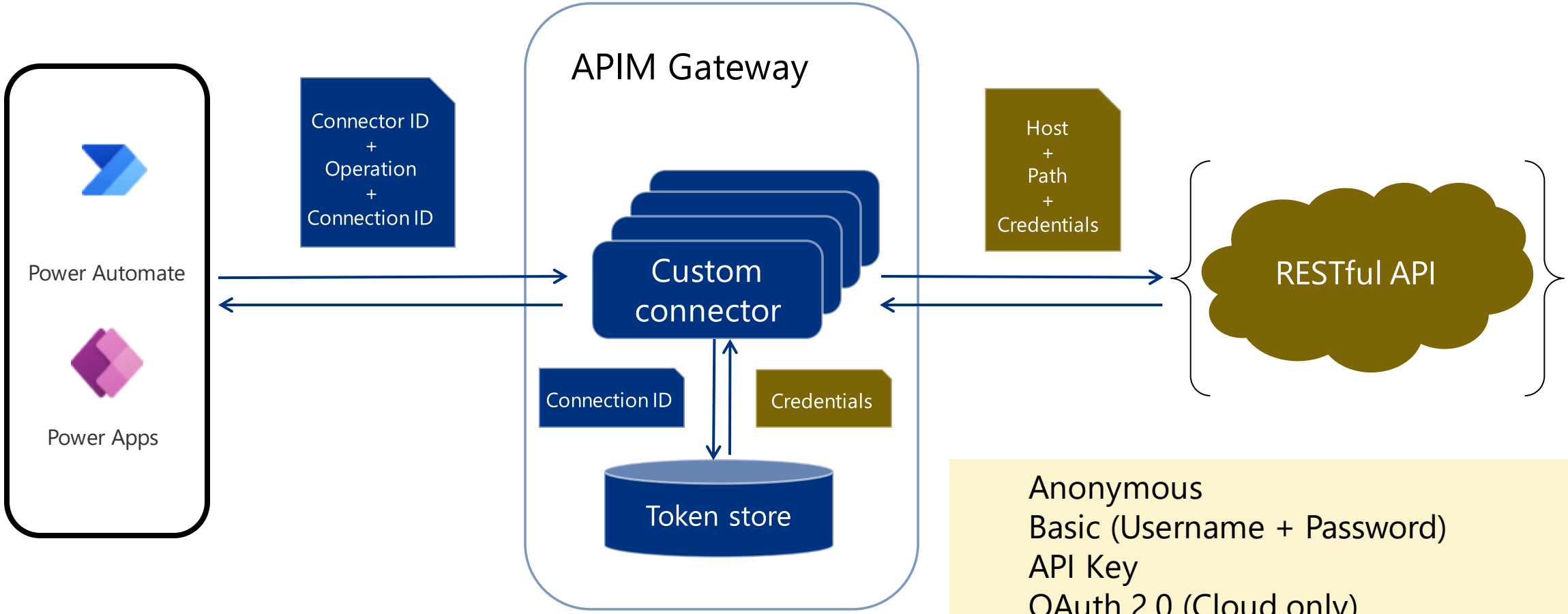
# Create connector from Azure API Management



# Create the connector



# Securing connector & API




Anonymous
Basic (Username + Password)
API Key
OAuth 2.0 (Cloud only)
– including Azure AD
Windows
- on-premises only

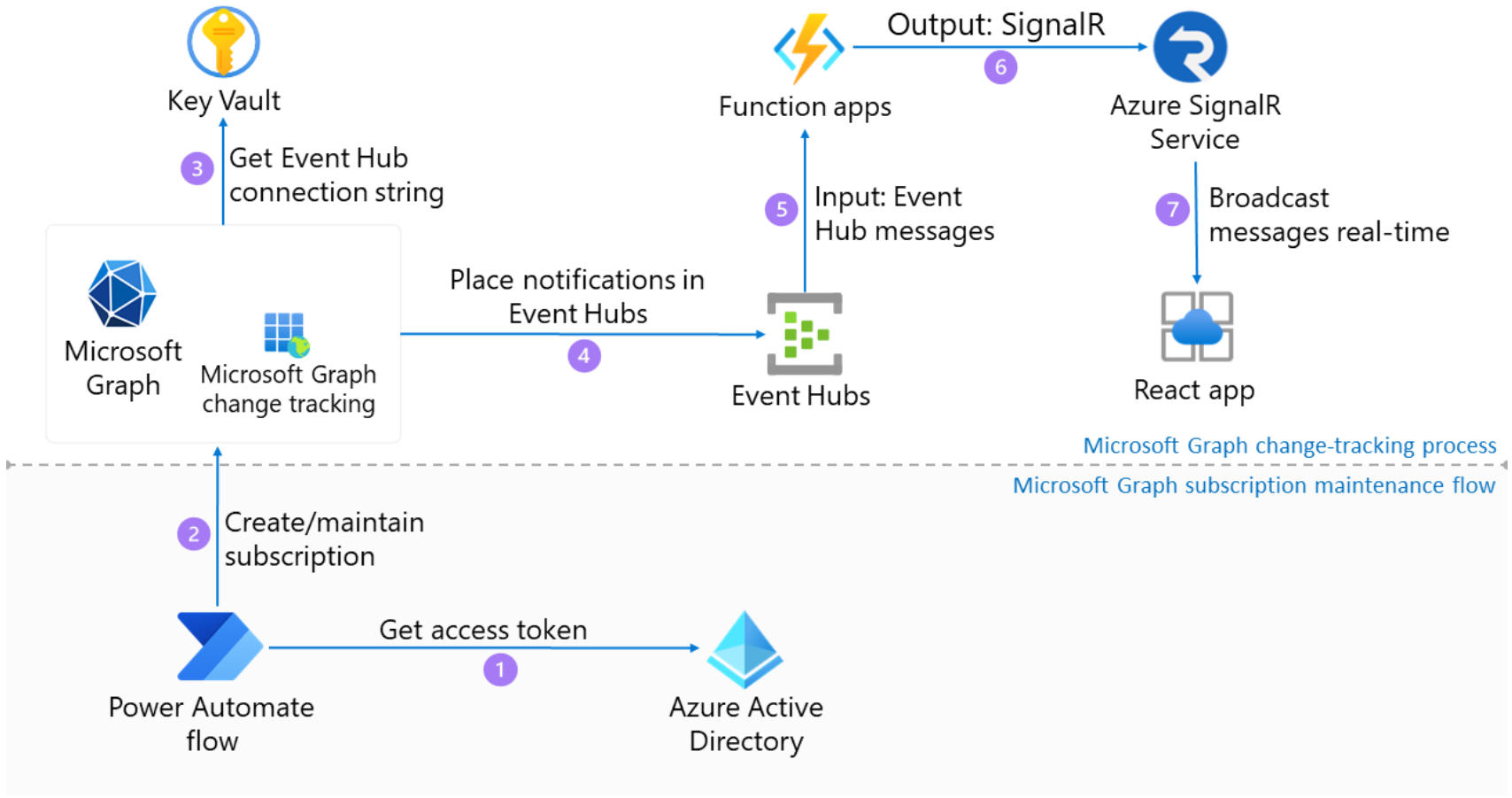
# Checklist for securing connectors with Azure AD

- Create two AAD application identities
  - One to identify and protect your service
  - Second to identify and protect your connector
- Delegate permissions correctly
  - Allow your connector's identity to make "on-behalf" calls to your service's identity
- Setting up security on your connector
  - ***Client ID***: ID of the AAD application that will identify the connector
  - ***Secret***: Secret of the AAD application ID specified above
  - ***Resource URL***: Resource identifier for your service
- Register and whitelist redirect URLs for connector's identity
- Whitelist domains for CORS on your service

# Designing APIs for Power Platform

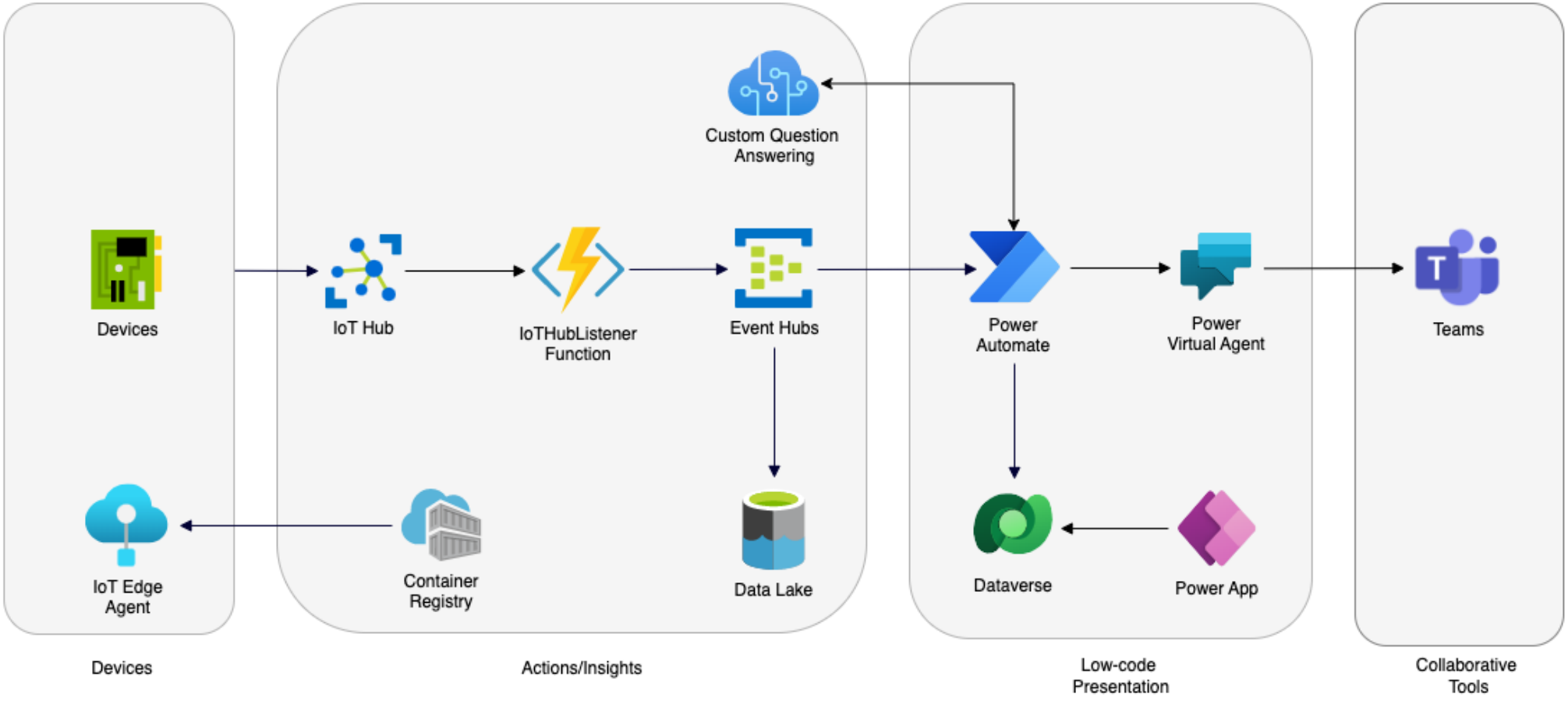
- Consider how the API will be consumed by Power Fx
- Complex object structures might be too complicated for makers
- Collaborate on how errors will be handled

# Power Automate and Real-Time Change Notifications




https://aka.ms/real-time-presence-scenario

# Power Platform & Azure IoT Edge Integration




https://github.com/appdevgbb/tornado-watchdog

# Hands on lab

Build a custom API for logic from a formula to prepare for future complexity

Use on behalf of authentication in the connector by securing with Azure AD

Configure a custom connector for the API

Adapt the Power App to use the connector

# Wrapping up

- Custom APIs and custom connectors can help remove complexity and centralize reusable logic for apps and flows
- Building custom APIs can enable makers to have access to logic previously only available through code
- Integration with Azure Services can make building, using and managing custom APIs easier

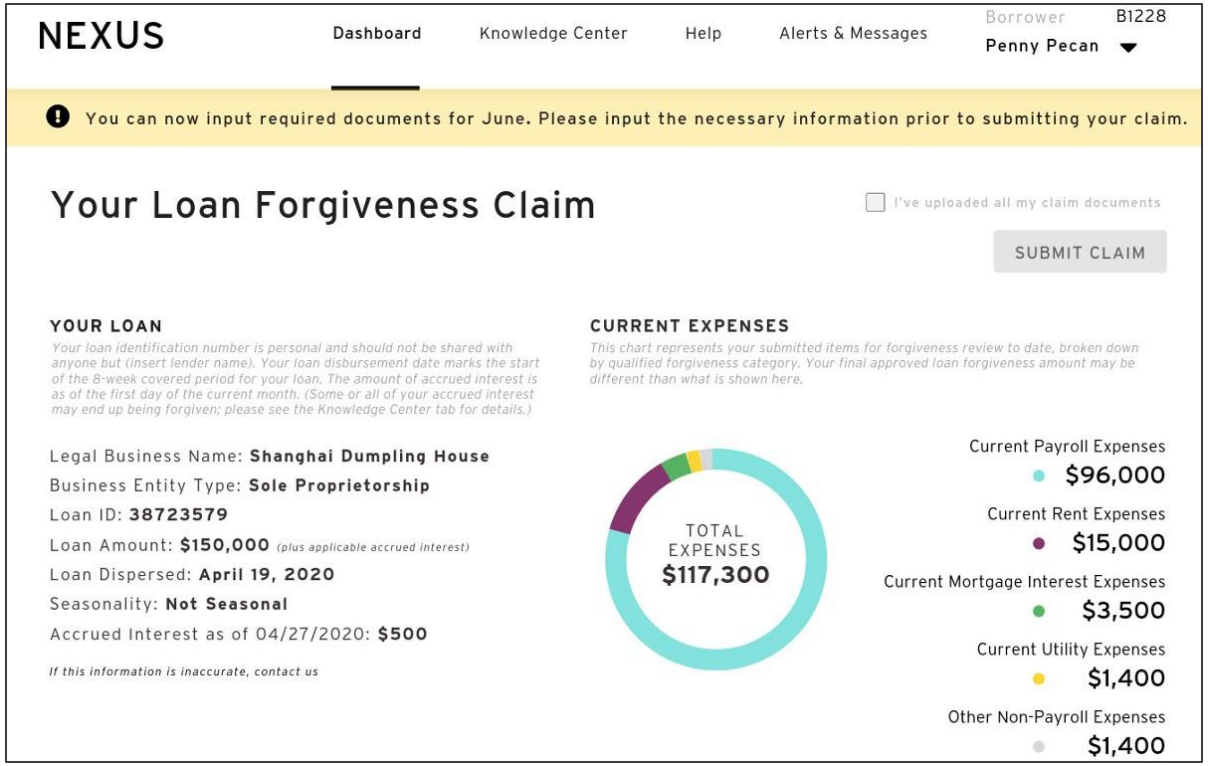
Microsoft Power Platform

# Low code or code first

# Meet professional developers using Microsoft Power Platform

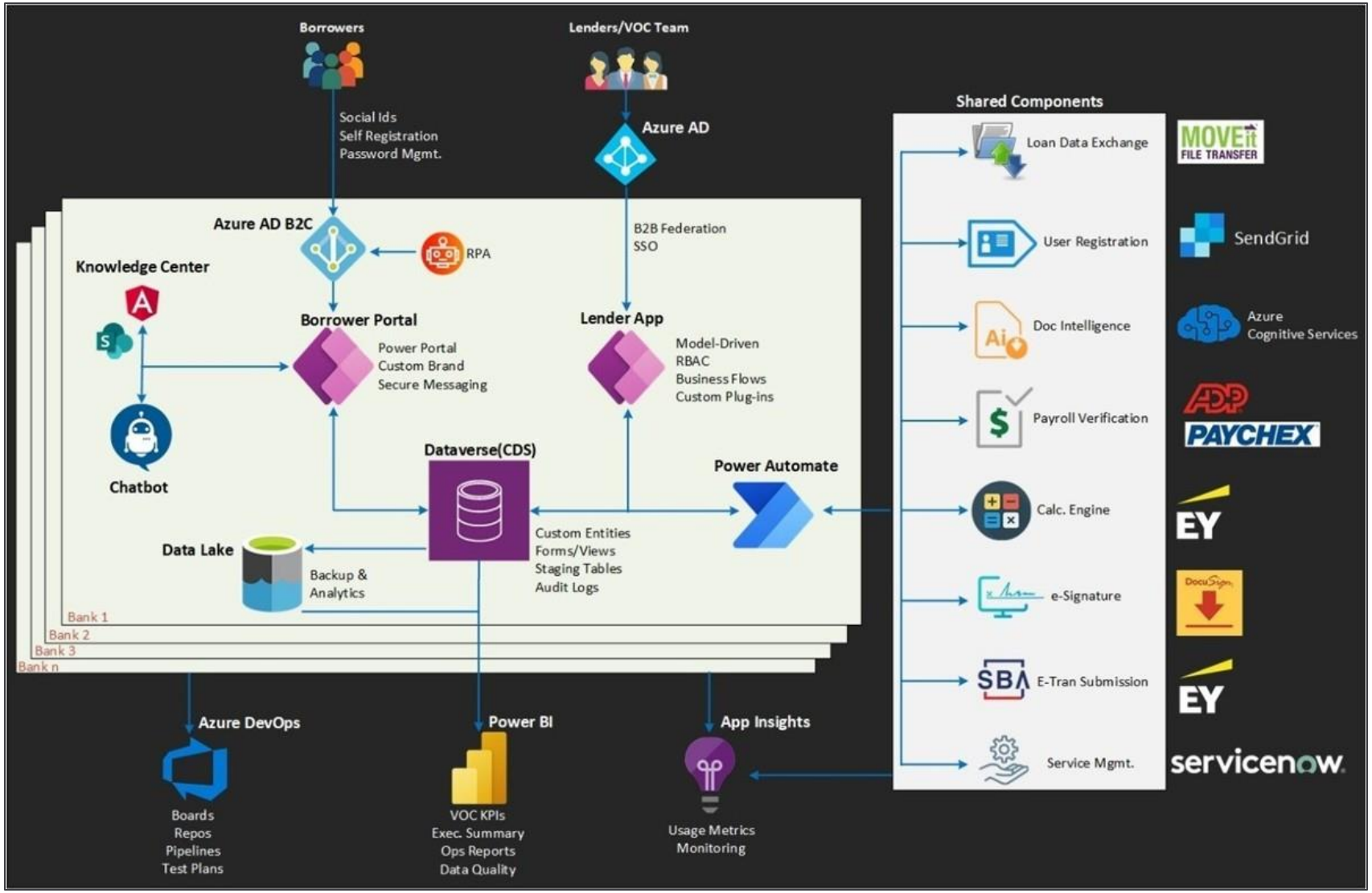
# Ernst & Young processes emergency loans for small business

- From prototype to production, in weeks, not months
- The solution works seamlessly with multiple third-party services using both prebuilt and custom connectors
- Using the tool, EY has been able to process over 50,000 loans totaling more than $10 billion




More details: aka.ms/EYPPP

# Solution Architecture



# Examples of pro-dev / Azure integration

- Calc engine calculates loan forgiveness.

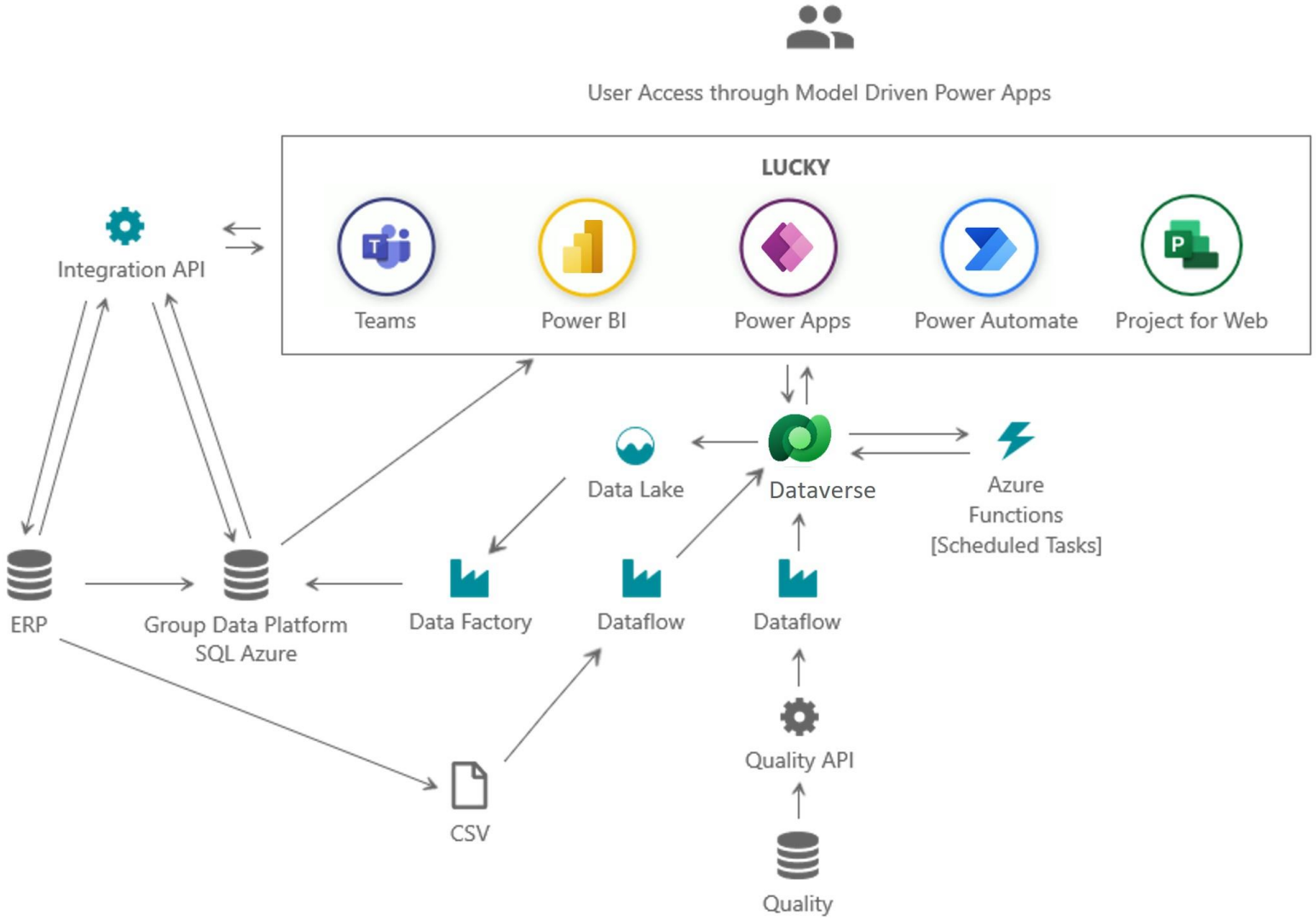
- This is a custom-coded component built as a microservice managed in Azure Kubernetes Service (AKS).

# Blackmores Group – Enabling rapid development of new products

- Central platform that brought together key data that was scattered among 10 siloed business systems
- Fusion development team at Blackmores, consists of professional developers and analysts
- Each app's highly tailored interface makes it easy for different teams to use exactly the data they need
- Their team treats Power Apps and Azure as one platform that is used to build solutions

More details: aka.ms/PowerPlatform/Blackmores

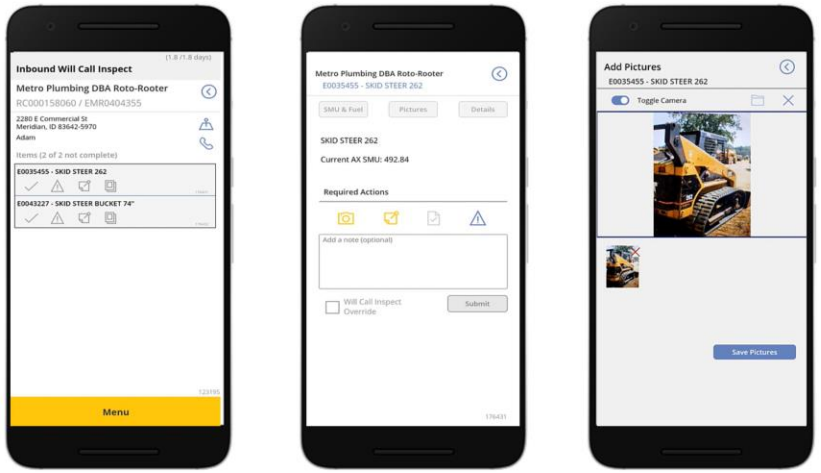
# Solution Architecture - Blackmores Group



# Western States Caterpillar – fusion teams

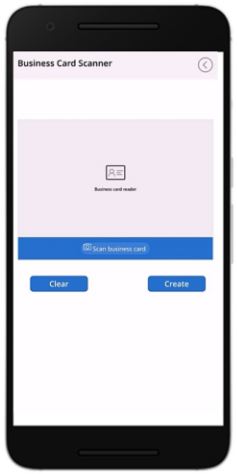
- Enabling their sales team out in the field to better understand their prospects and customer spend

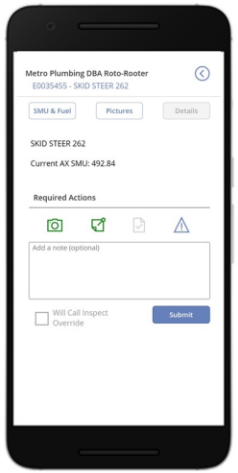
- They accomplished this by working as a fusion team using a combination of low-code technologies (Power Apps, Power Automate, Power BI) along with “pro-dev” or code-first technologies (ASP.NET Core, Custom connectors, Azure Blob storage, Azure SQL).

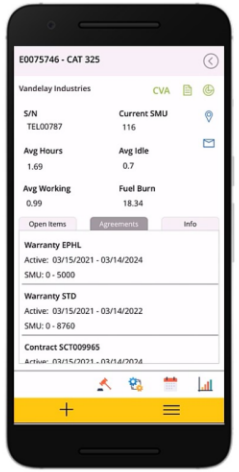


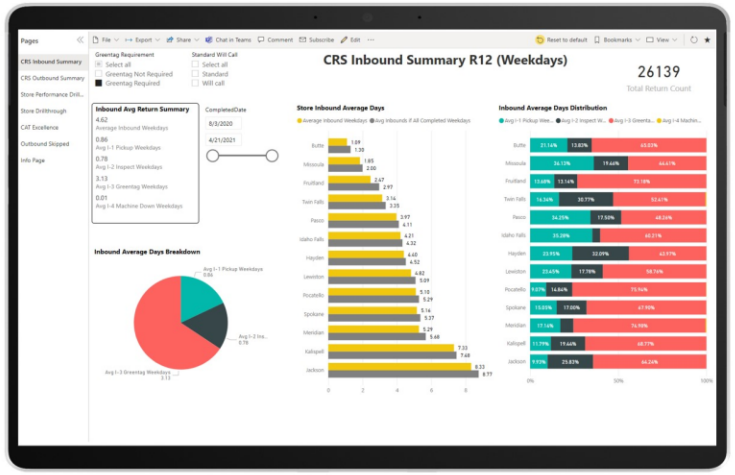

More details: aka.ms/PowerPlatform/WesternStatesCAT

# Solution Architecture - Western States Caterpillar




## AI Builder

Create prospects and contacts instantly with the Business Card scanner

## Power Apps

Mobile apps used by employees in stores – for rental returns and during sales visits for capturing opportunities and viewing customer history

## Power BI

Reports provide visibility of the equipment rentals across all stores to understand the volume of rentals and the turnaround timeframes, and total customer spend on sales

### Azure Blob

Used to store photos related to rental equipment.

### Custom Connector

### ASP .NET Core

Web APIs created to retrieve and write data for events used by the custom connector.

Custom connector used to integrate with external services – e.g. API from 3rd party auction platform.

### SQL Server Database

Primary data source for storing information used by apps and reports.

### Microsoft Dynamics AX

ERP system for all equipment transactions and sales information. Relevant data is replicated to SQL Server database.

# Examples in your organization?

# Identifying opportunities

1

Look for existing internal APIs that could become custom connectors

2

Evaluate custom visuals that might be wrapped as Power Apps code components

3

Look for projects combining data from multiple data silos – could be candidates for a custom API that consolidates

4

Use fusion teams to evaluate opportunities to ensure right mix of low-code and code first

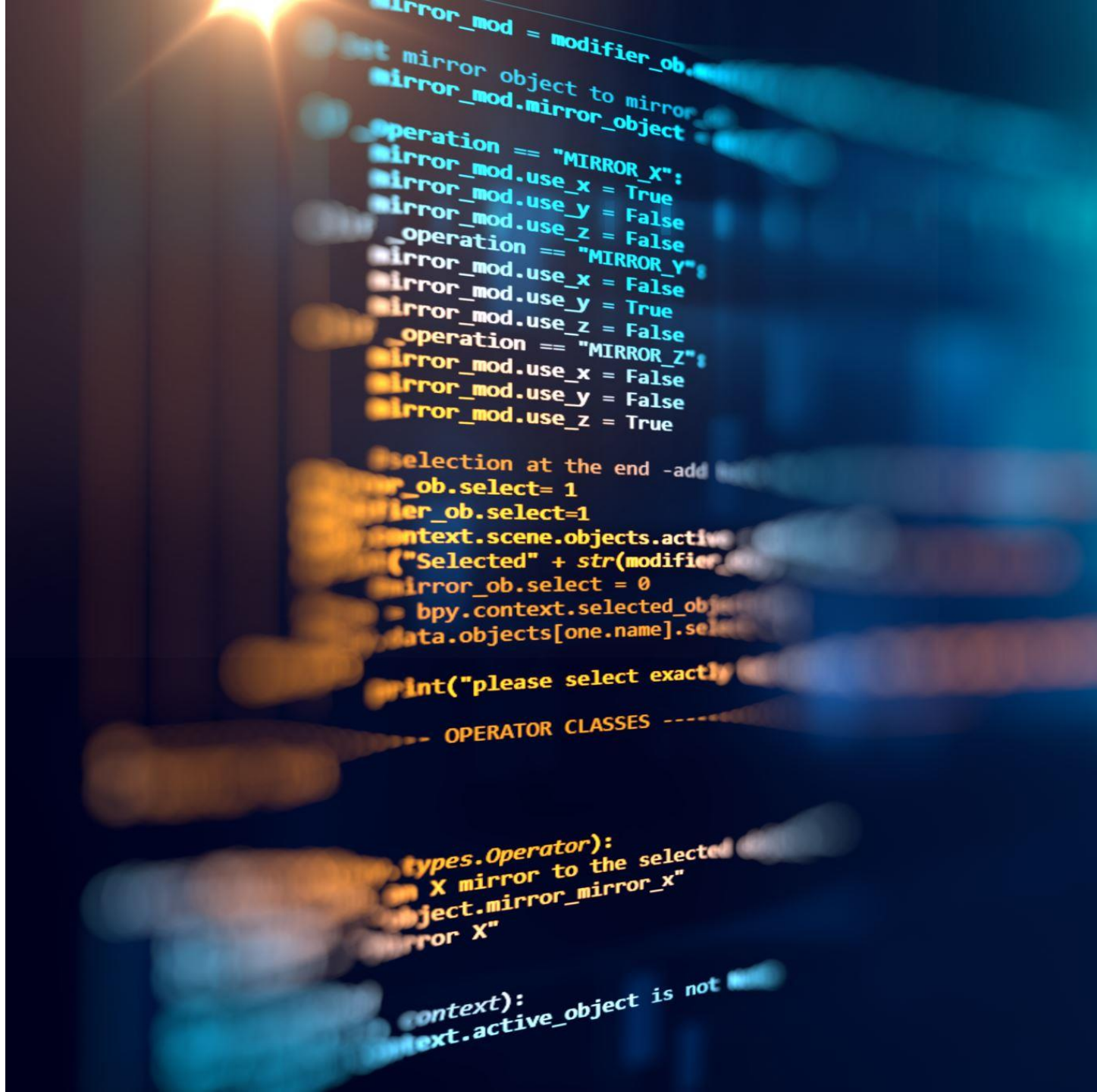
# Working as a fusion team

Recognize low-code and code first both have their place

Knowing what can be done in low-code can avoid reinvention in code first

Consider how your code-first efforts will be used to make them low-code friendly

Be ready to help troubleshoot low-code that produces code-first errors



# More Resources – aka.ms/powerforpros/#Resources

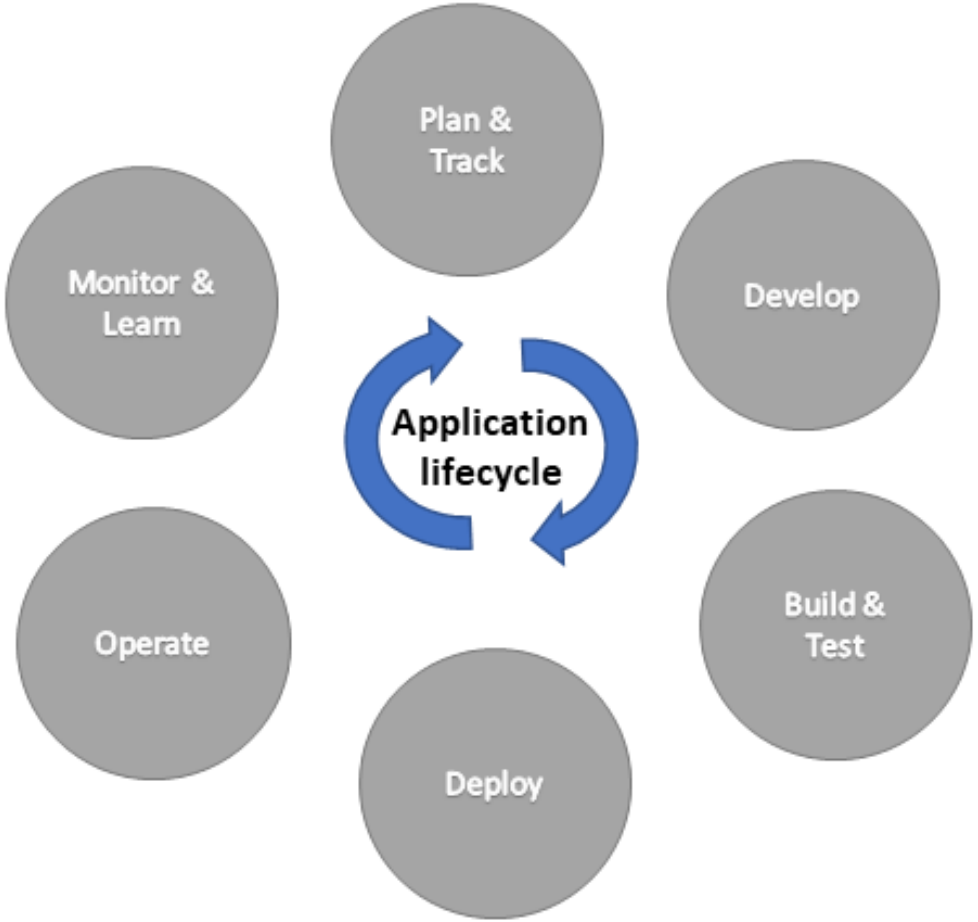
- Microsoft Power Platform is the best way for teams to build together – blog post
- Pro-dev features and learning paths – E-book | learning path
- New Power Apps developer plan – get started | documentation
- Microsoft Mechanics episode on pro-dev adoption of Power Platform – blog | video
- Low code application development on Azure – landing page | YouTube video playlist | tours
- Power Apps and Microsoft Azure better together – landing page | TEI report | ROI calculator
- Low Code Revolution video series – https://aka.ms/lowcoderevolutionshow
- POWERful Devs virtual conference and ongoing videos – video series
- Learn more about ALM with the Power Platform - http://aka.ms/almpowerplatform

Microsoft Power Platform

# (Bonus)
# Power Platform
# Application Lifecycle Management

# Power Platform Application Lifecycle Management(ALM)

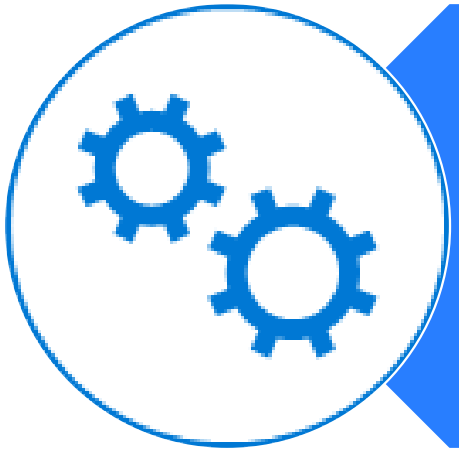
- Solutions are used to distribute components across environments through export and import
- Dataverse stores all the artifacts, including solutions
- Source control should be your source of truth for storing and collaborating on your components
- Continuous integration and continuous delivery (CI/CD) such as Azure DevOps or GitHub allow you to automate your build, test, and deployment pipeline



# Working with solution files

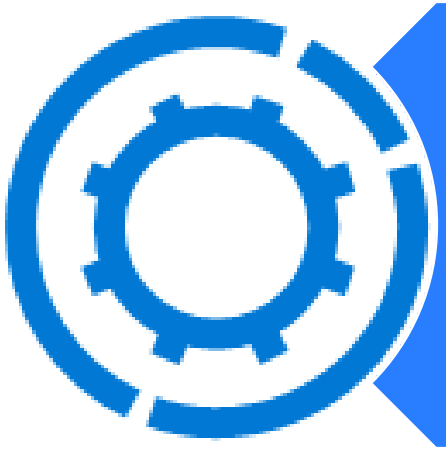
- Can be imported/exported manually or using build tools
- Solutions are exported as a single compressed file
- Using the CLI or build tools you can “unpack” into individual files appropriate for source control
- Individual files from source control can be “packed” for import into other environments

# Types of solutions



## Unmanaged

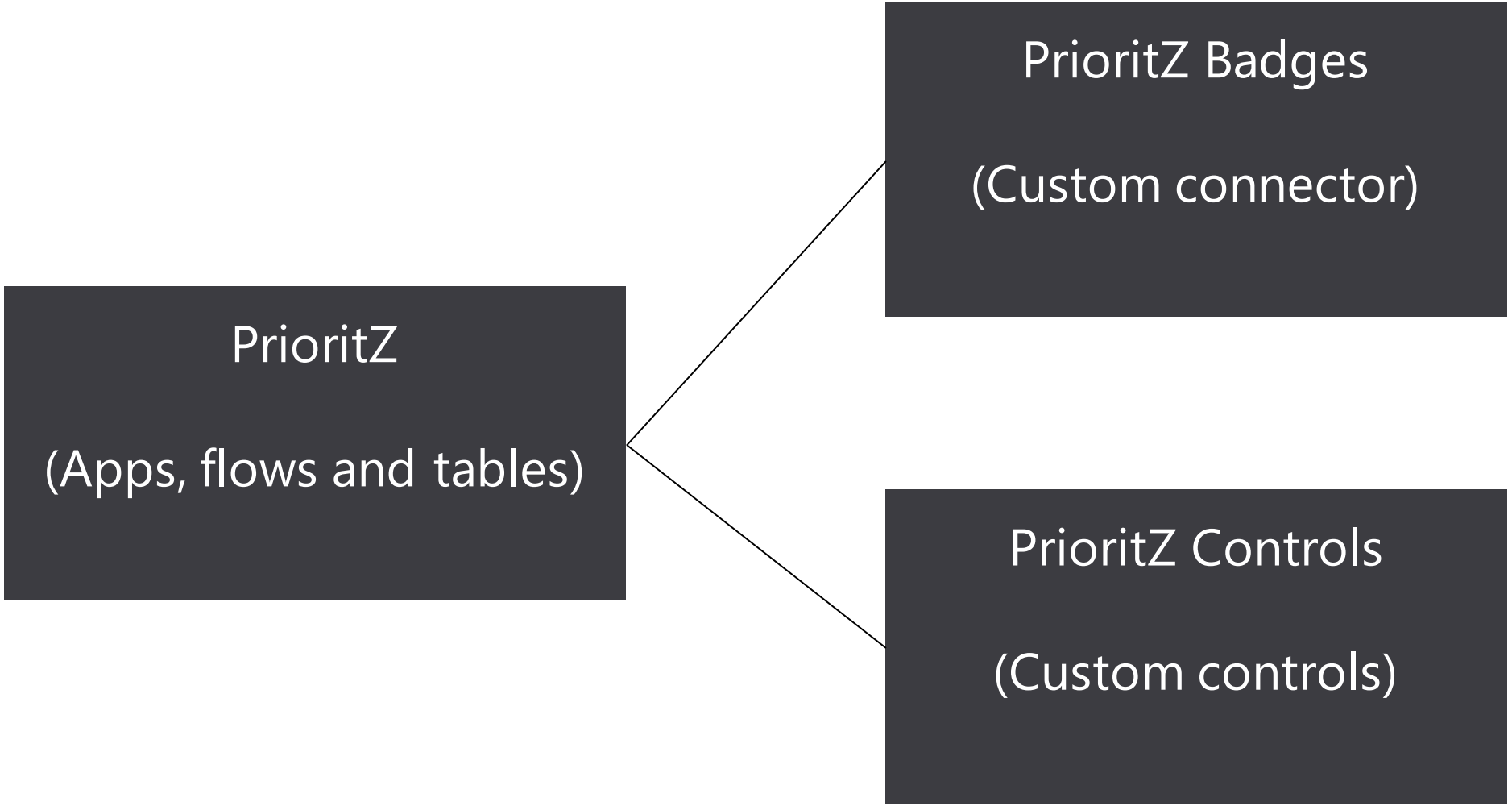
- Used during development
- Used to transport to other development environments
- Should be considered your source



## Managed

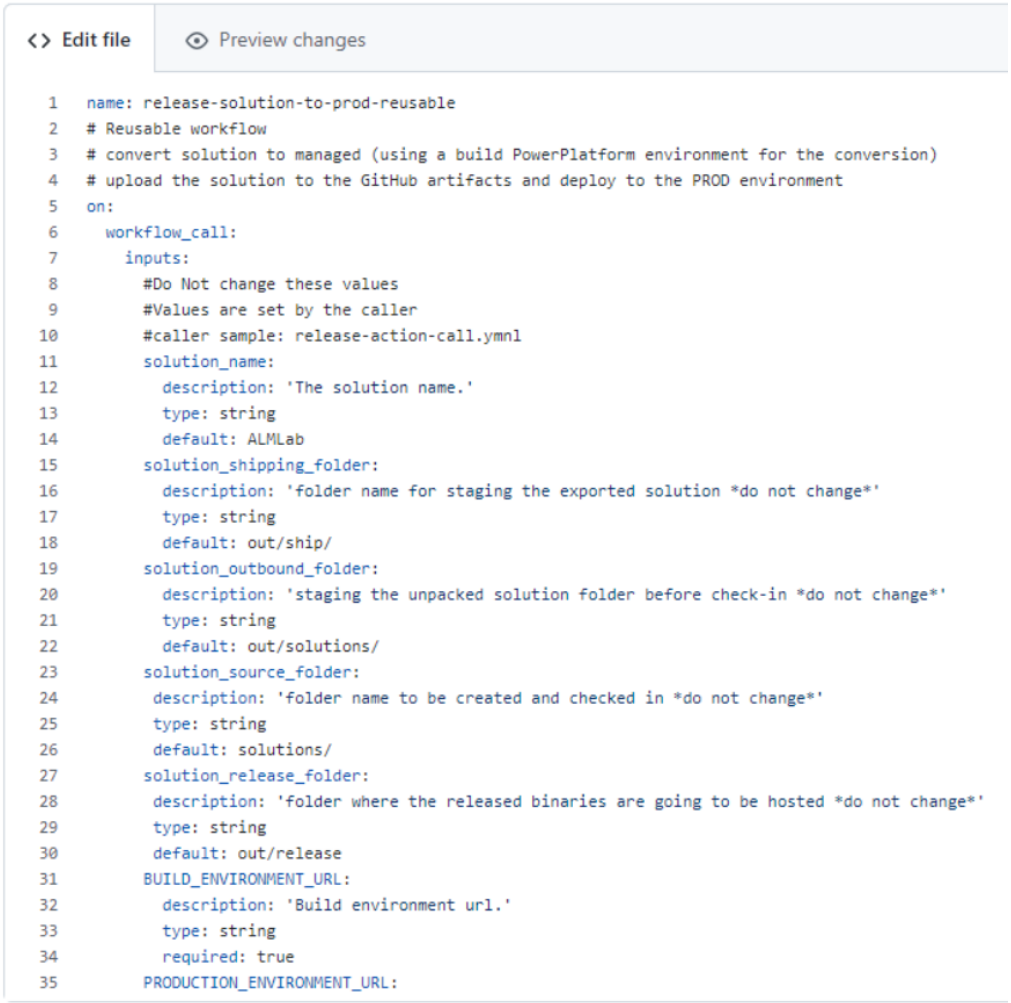
- Used to distribute to non-development environments
- Should be generated by a build server and considered as a build artifact

# Solutions can depend on other solutions



# Automate with Power Platform build tools

- Available for Azure DevOps and GitHub
- Create workflows in your repository to build, test, package, release, and deploy apps
- Actions for solutions, quality check, and environment management



# Hands on lab

Create workflows to export to source control and deploy to test environment

# Wrapping up

- Healthy ALM practices are an important part of a Power Platform project

- Build tools allow you to automate and create a repeatable ALM process

- Learn more about ALM with the Power Platform: http://aka.ms/almpowerplatform