

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



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>



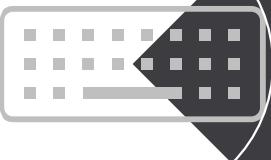
@DanWahlin

Get the Content



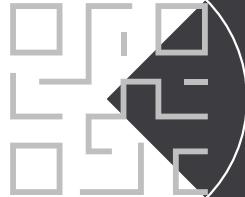
<https://aka.ms/DevIntersectionPowerAzureWorkshop>

Who are you?



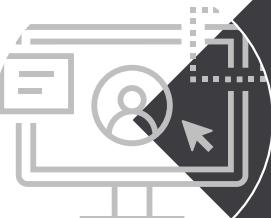
Coder

- .NET dev



Automation specialist

- Ready to code



Power Platform dev

- Time to learn more

You are developing for Contoso Coffee today

PrioritZ the list

Change Taco Tuesday to some other food

Steak Tuesday

Tamale Tuesday

Cheese and Wine Tuesday

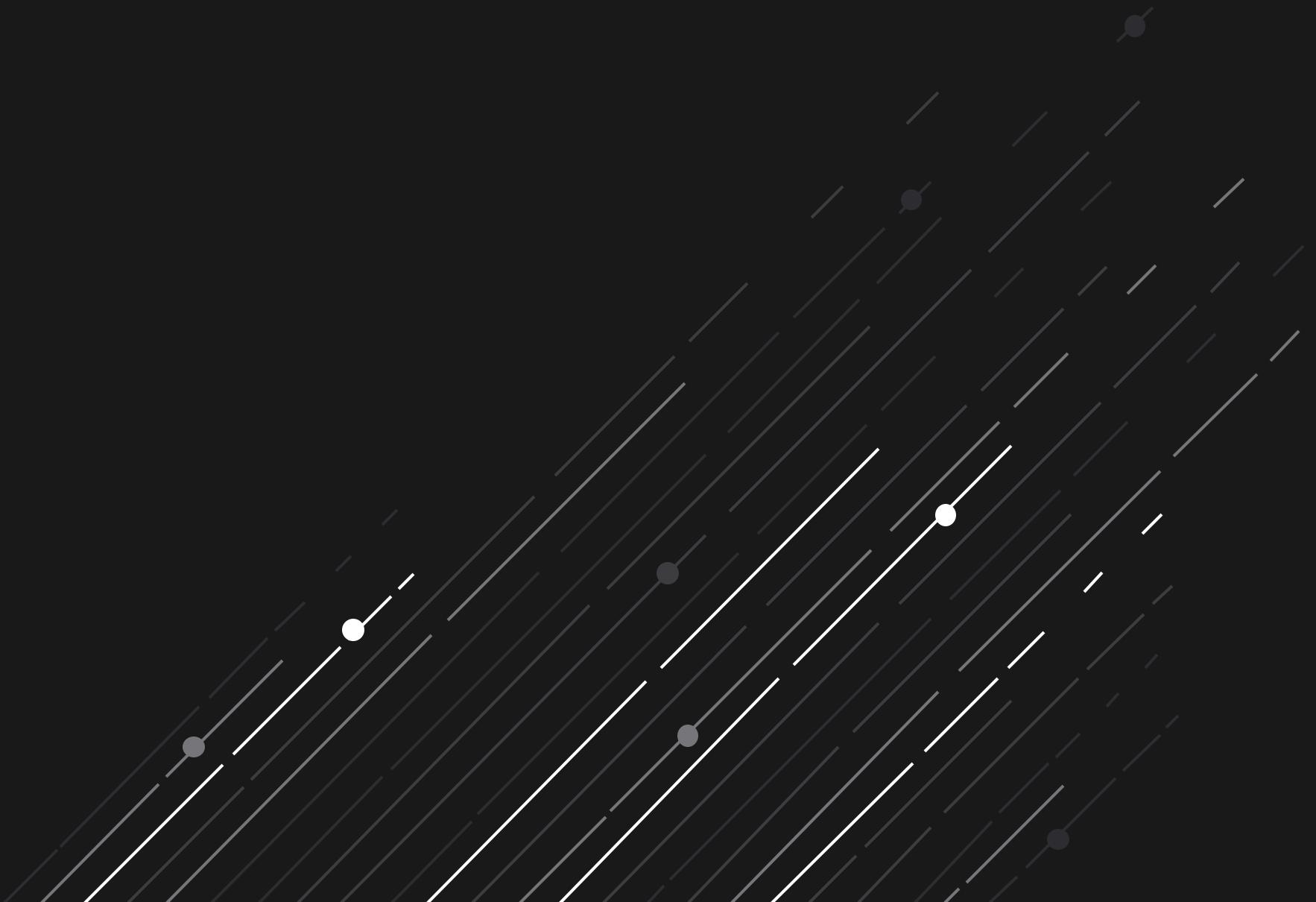
Steps

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?



Overview



Power Platform: Everyone's platform for transformation



No Code

Drag and drop experiences that are like PowerPoint



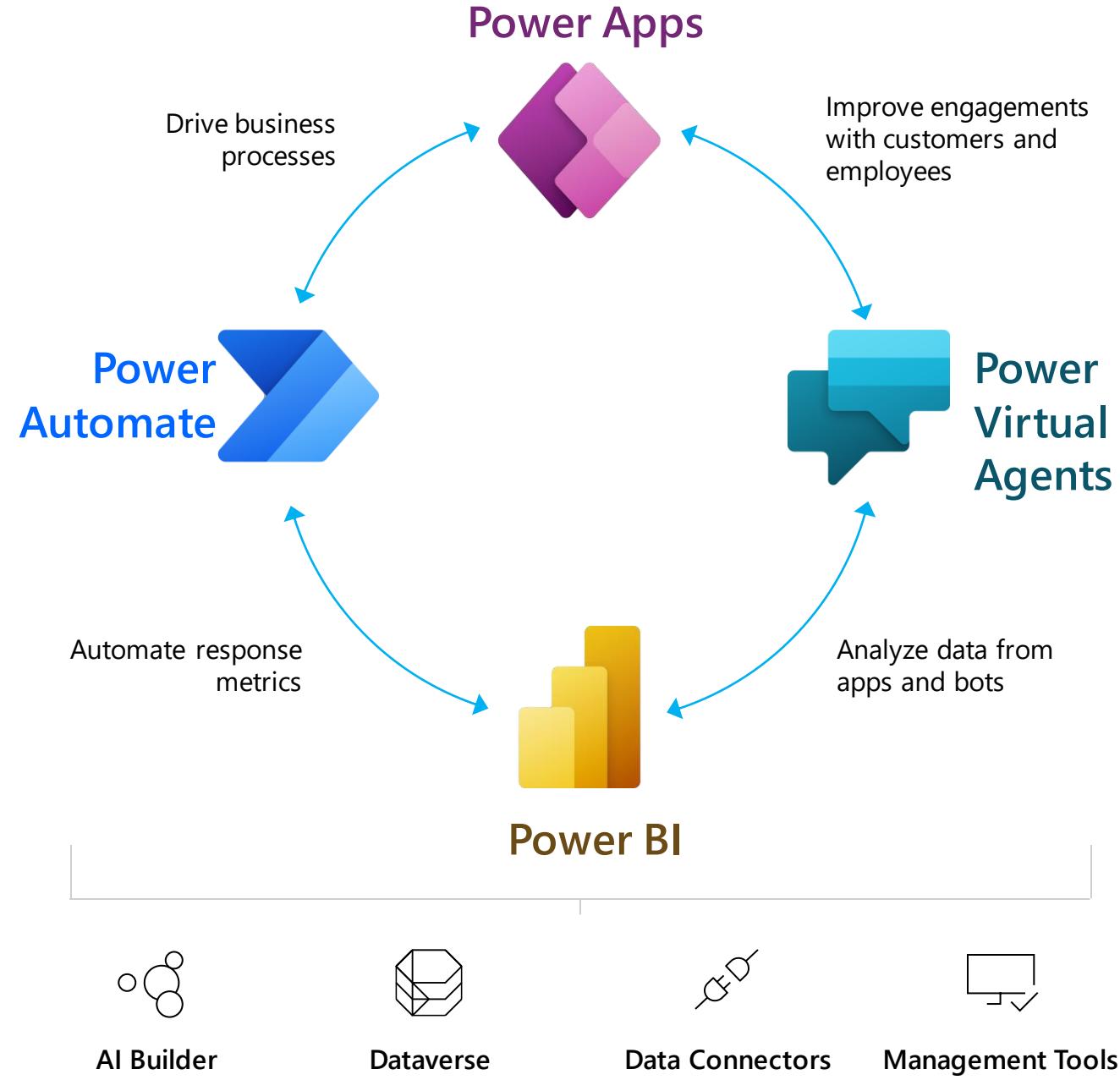
Low Code

Drag and drop plus Excel-like formulas for functionality

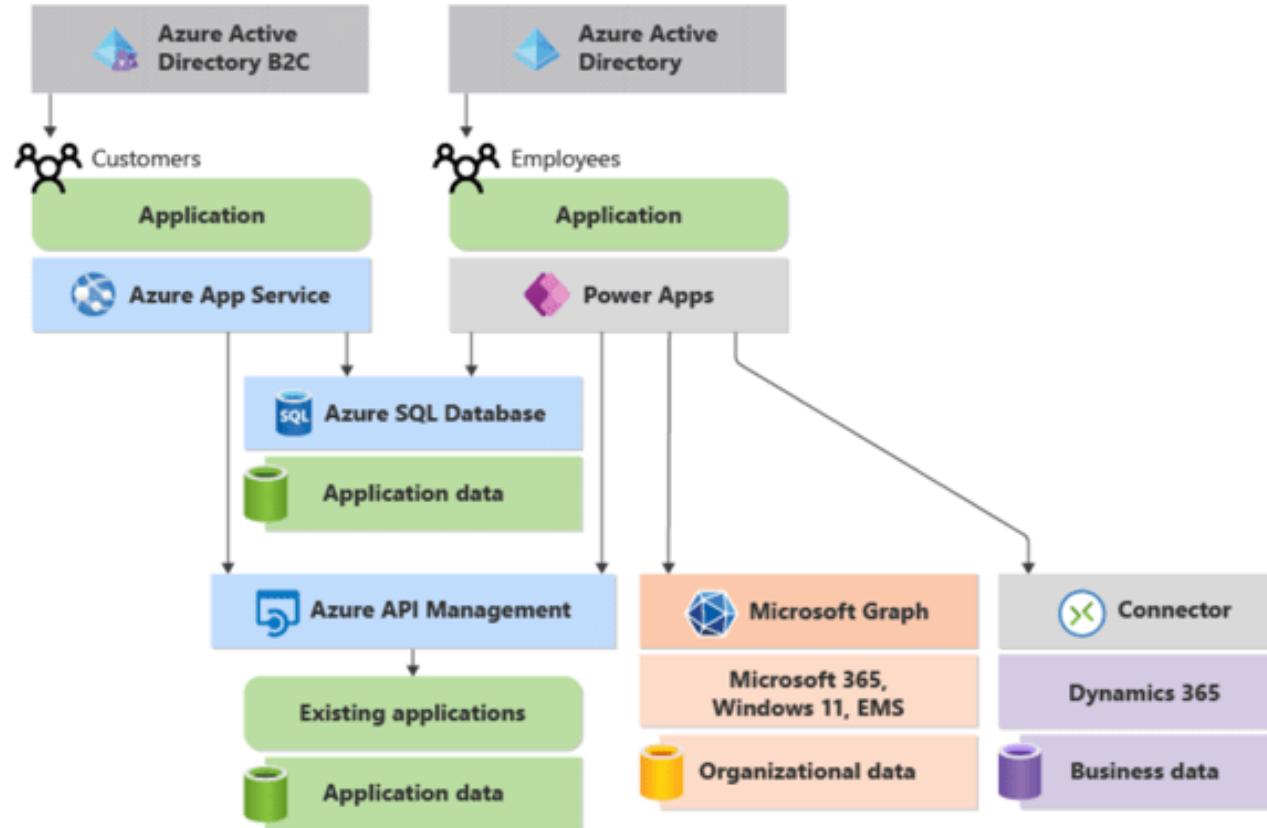


Code First

Professional development and extensibility across Microsoft



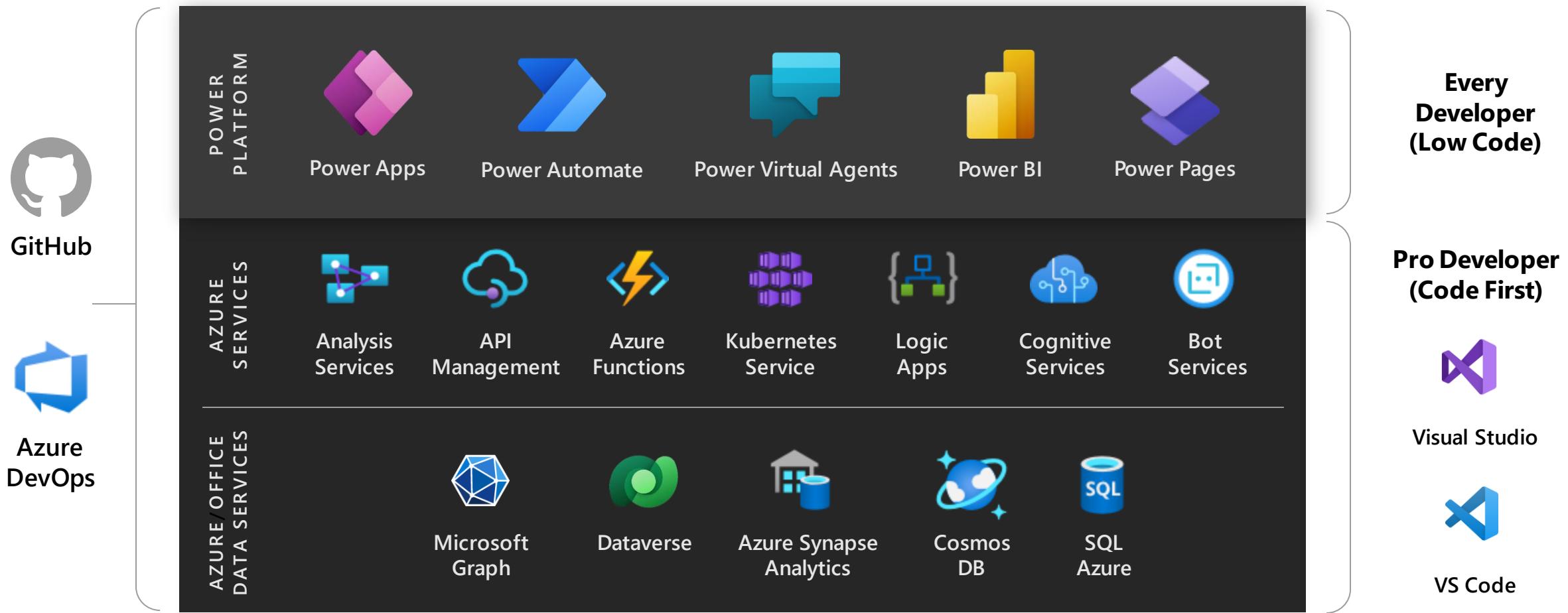
Building on the Microsoft Cloud



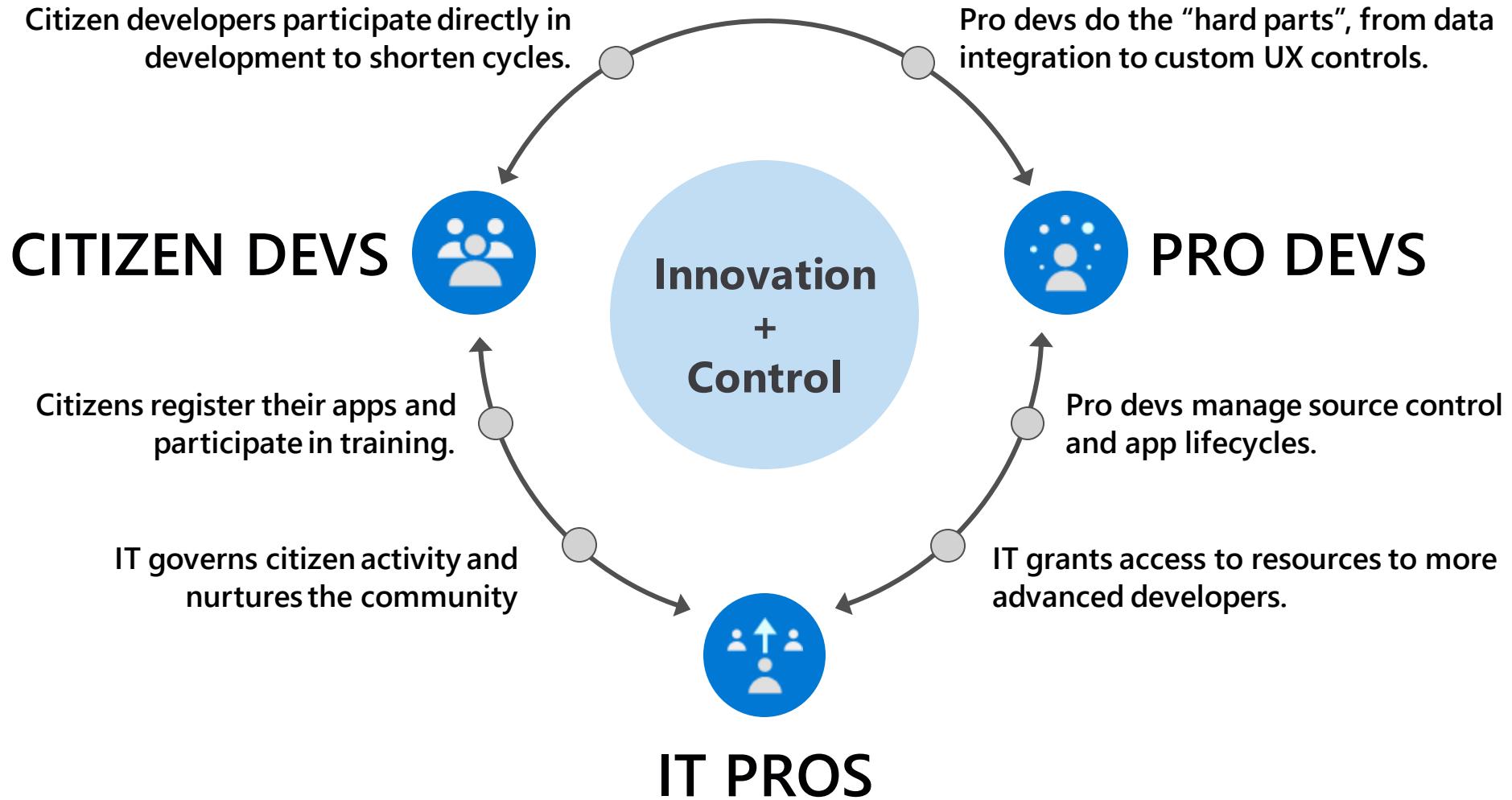
<https://aka.ms/microsoft-cloud-solutions>

Extend the Power Platform with Azure Services

Integrate Azure services with pre-built and custom APIs

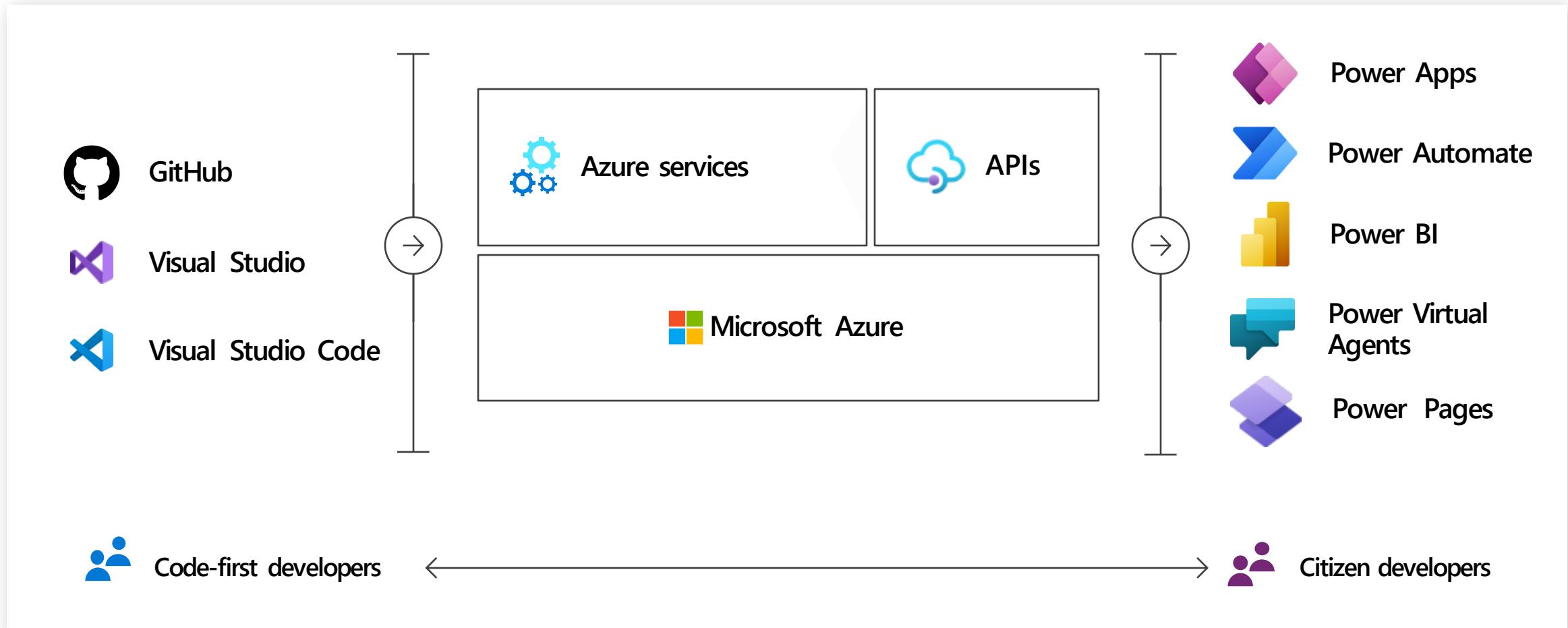


Power Platform development is a team sport



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

MALIK

OFFICE MANAGER



IT/PRO FUSION TEAM
CITIZEN DEVS

PREETI
IT SYSTEMS ADMIN



KIANA
FULL STACK DEV



CALEB
FIELD TECHNICIAN

MARIA
INVENTORY MANAGEMENT



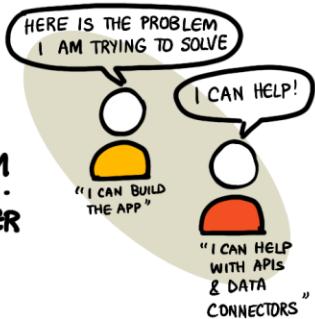
TEAM PROCESS

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

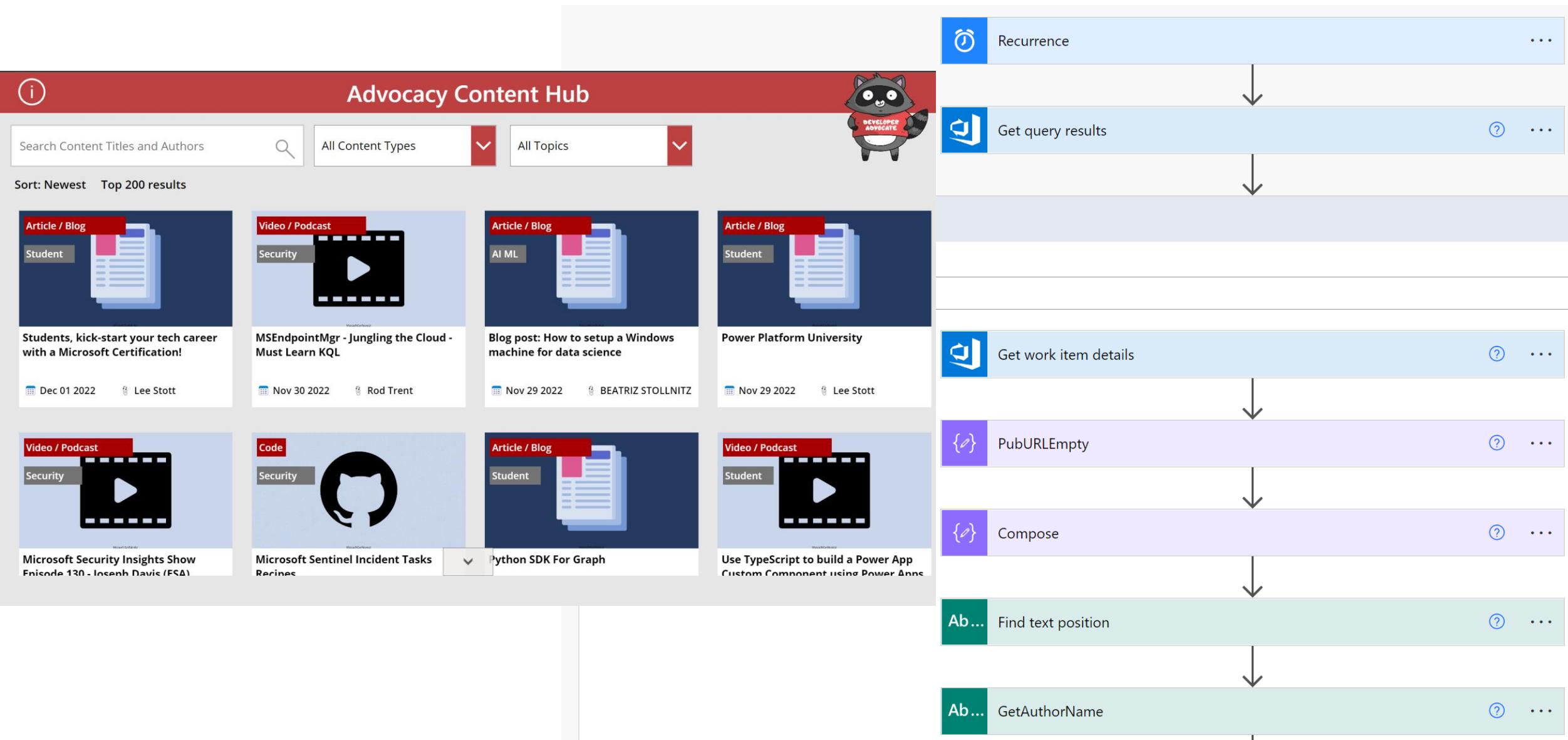
C

EXECUTION

TEAM CHECKS IN REGULARLY TO REVIEW BLOCKERS, EVALUATE PROGRESS

Fusion Development Process

Fusion Dev at Microsoft – Advocacy Content Hub

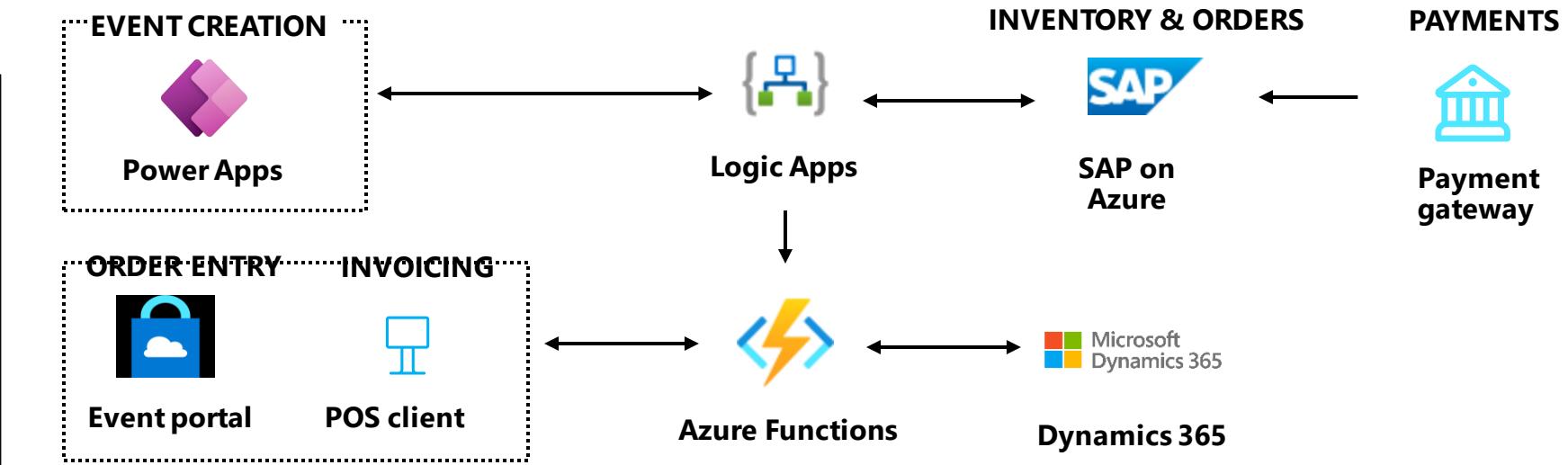


Streamline event management, like Coca-Cola UNITED



"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.

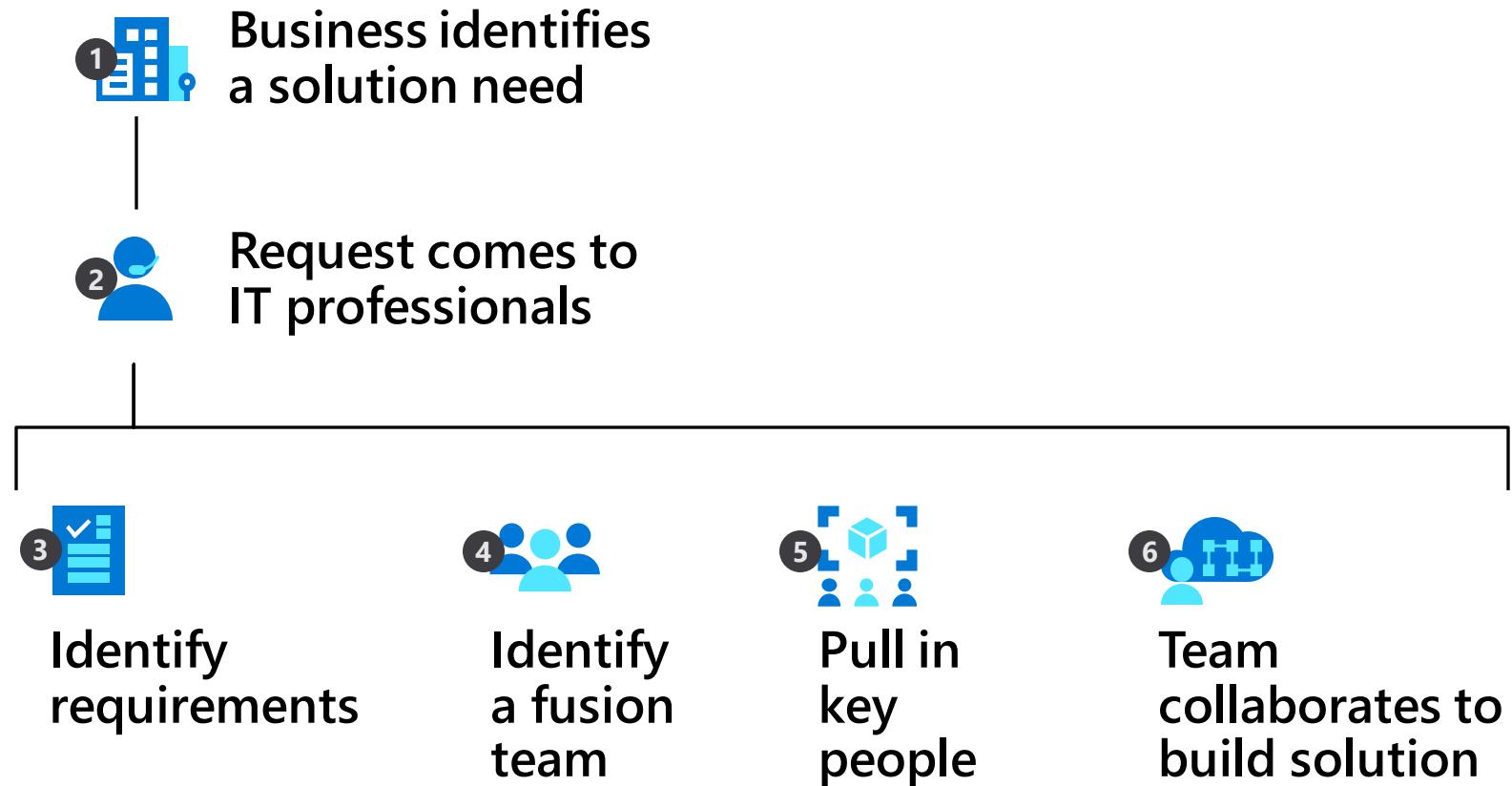
Solution

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

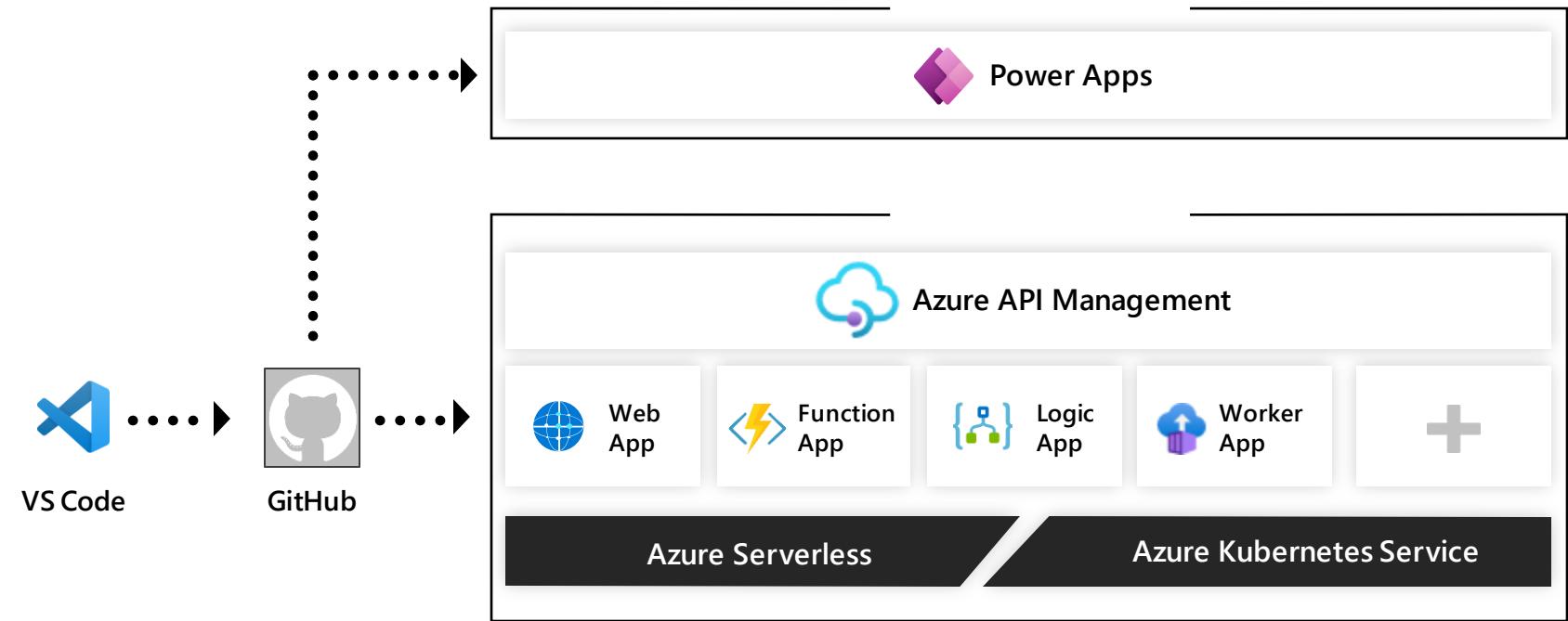


Low code to serverless for modern app dev

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

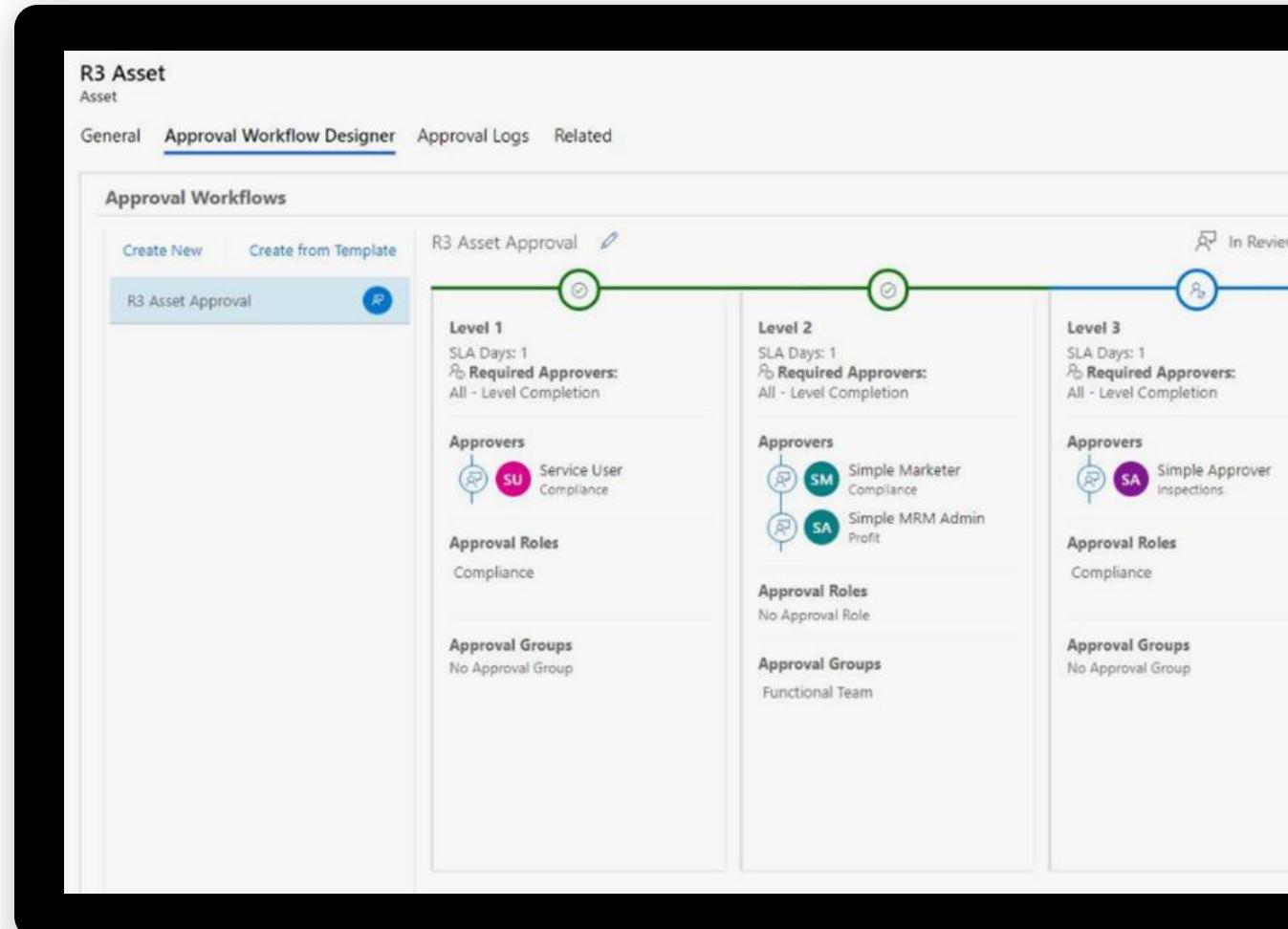
Azure API Management acts as the gateway to microservices and code deployed in Azure

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.



BUILD API

Start with API for data source or build new API.



BUILD CONNECTOR

Define connector using custom connector wizard or Power Query SDK.



BUILD SOLUTION

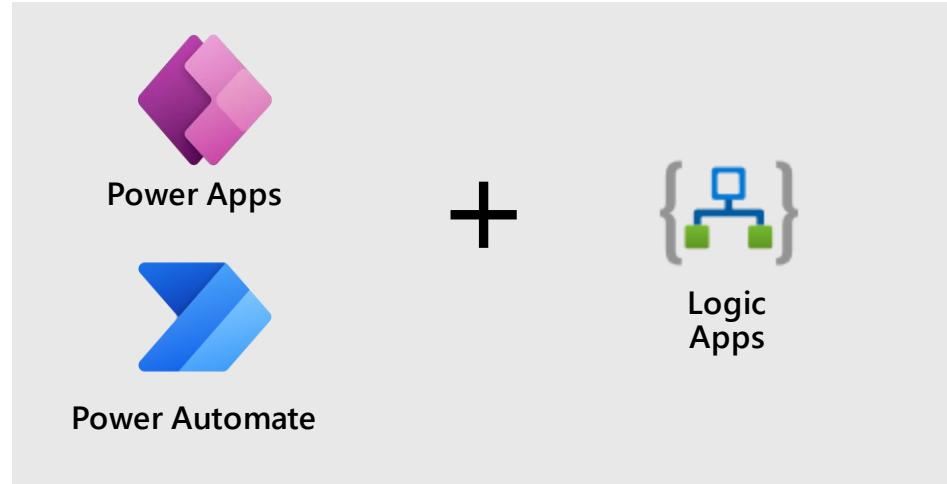
Citizen developer selects custom connector to build app, dashboard, etc.



SHARE CONNECTOR

Add connector to Power Platform or share directly with citizen developers.

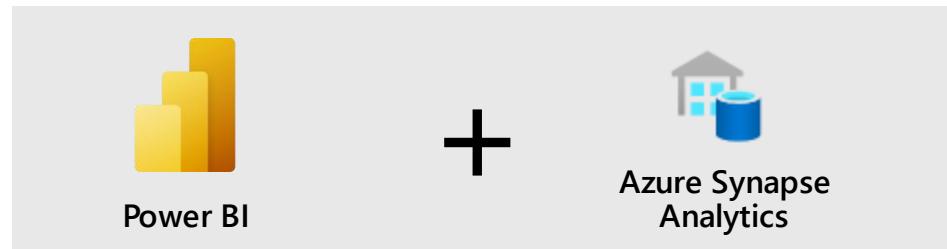
Power Platform + Azure services in action



- 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.

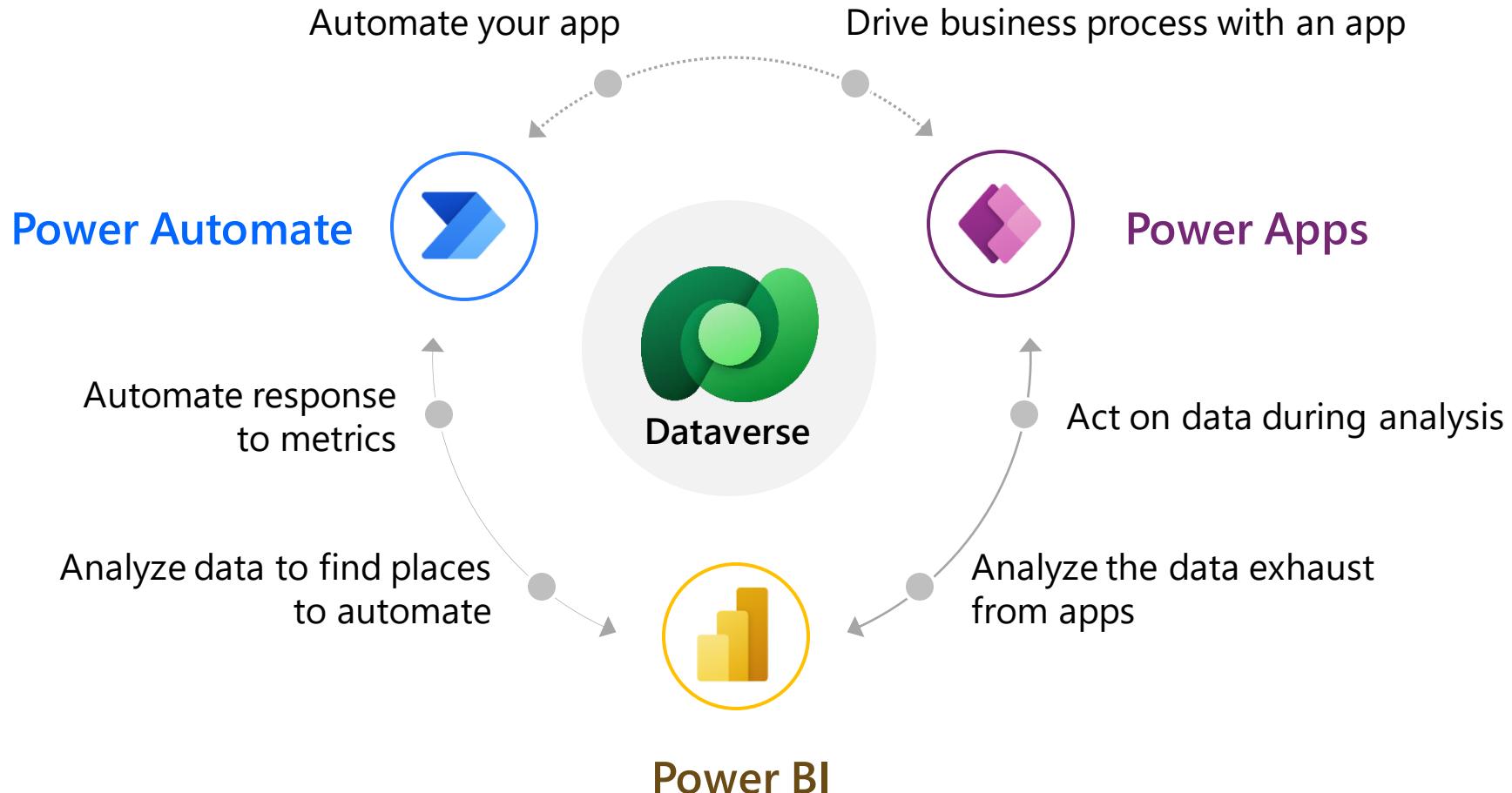


- Convert custom bots built with pro-code tools to a Skill and embed the Skill within a Power Virtual Agents bot.

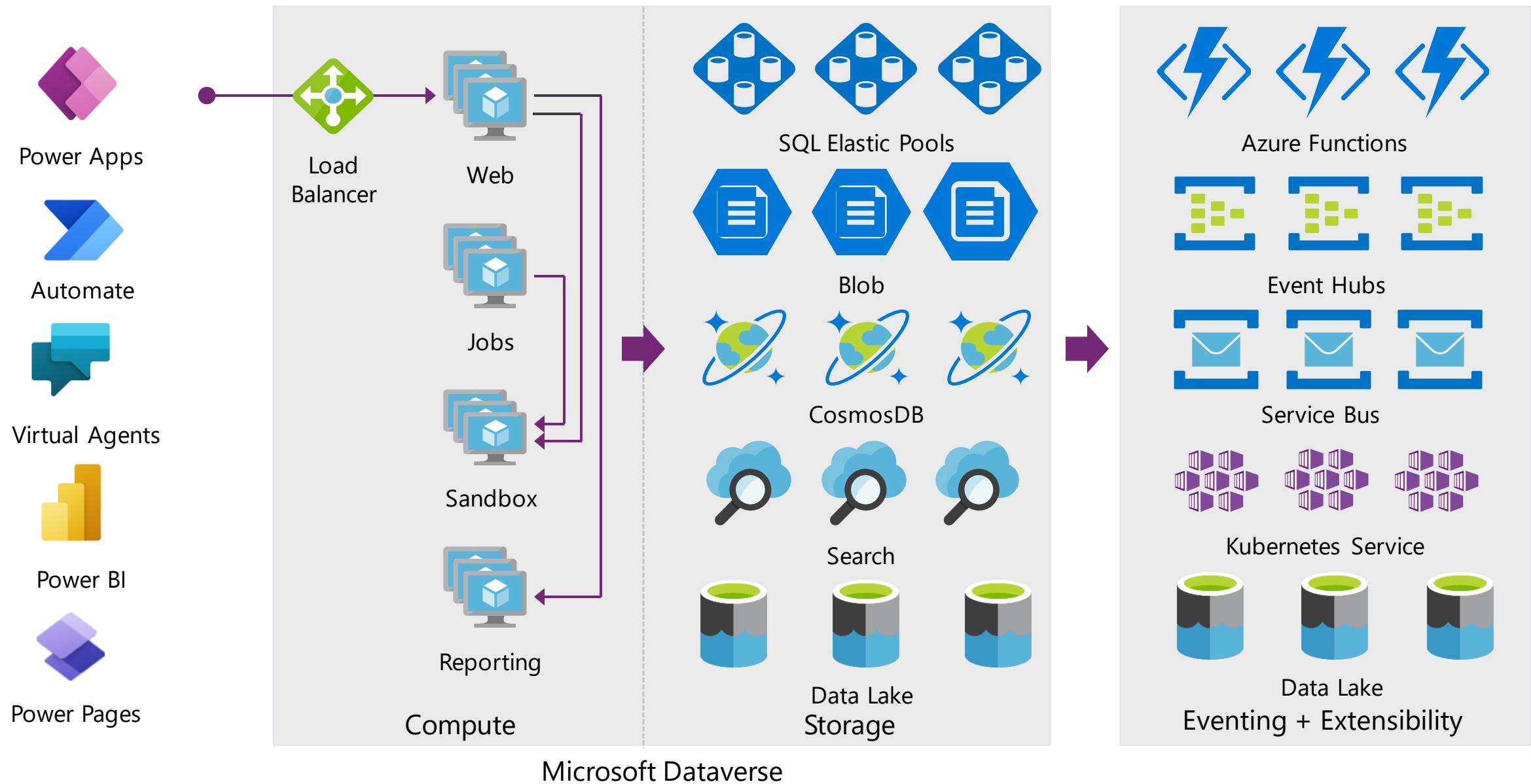


- Expand discovery of insights from all your data and reduce project development time with a limitless analytics service.

Microsoft Dataverse is fully integrated into the Power Platform

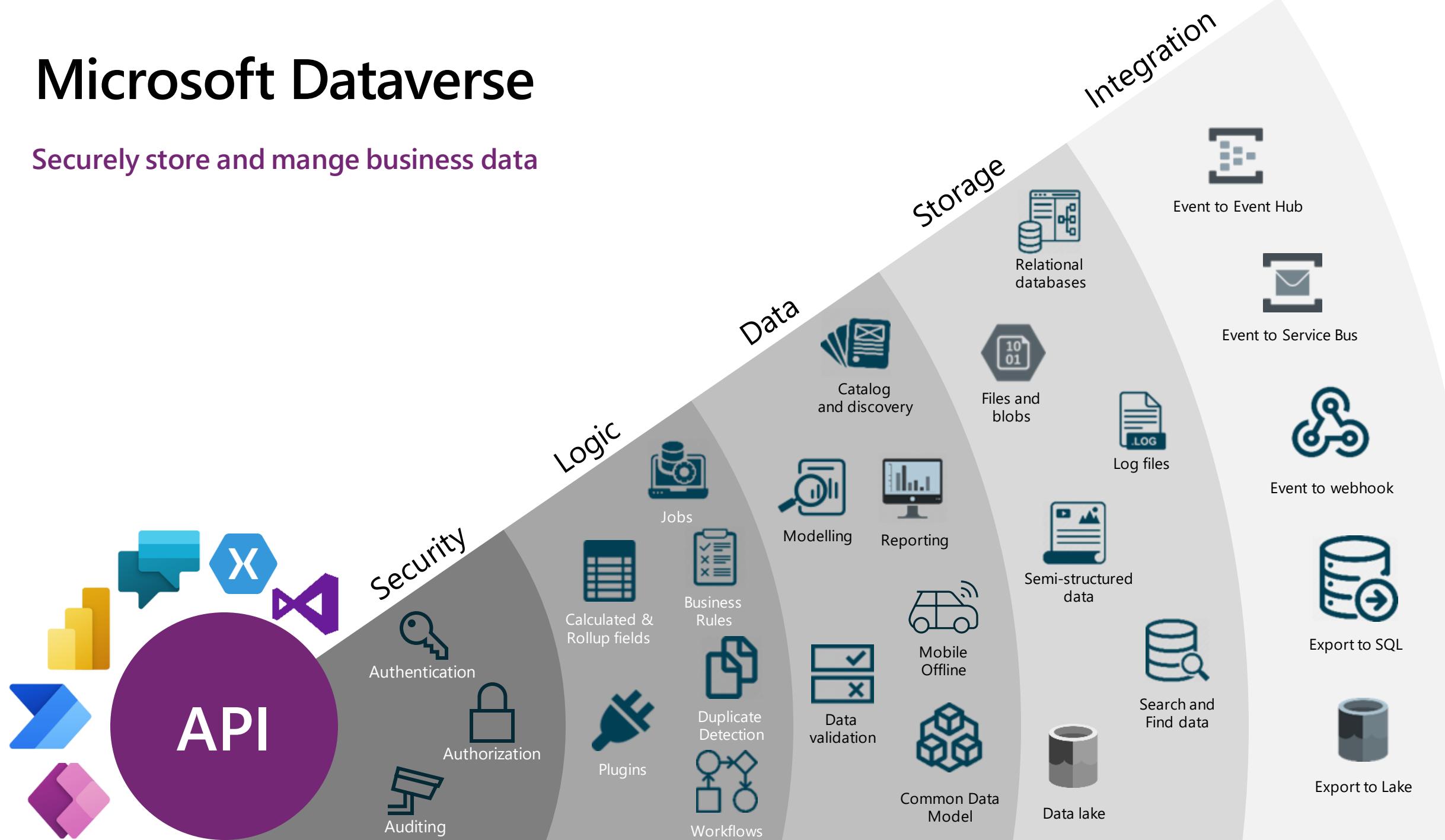


Microsoft Dataverse... Runs on Azure and Extends with Azure



Microsoft Dataverse

Securely store and manage business data



Easily integrate with Microsoft Teams

Microsoft Teams integrations

Ready-made apps



+ hundreds of partner apps

Low-code apps & workflows
(Power Platform)

Power Apps



Build custom apps

Power Automate



Automate repetitive tasks

Power Virtual Agents



Create tailored chatbots

Power BI



Visualize and discuss data

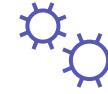
Full-code custom solutions



Departmental tools



Employee resources



Processes and workflow

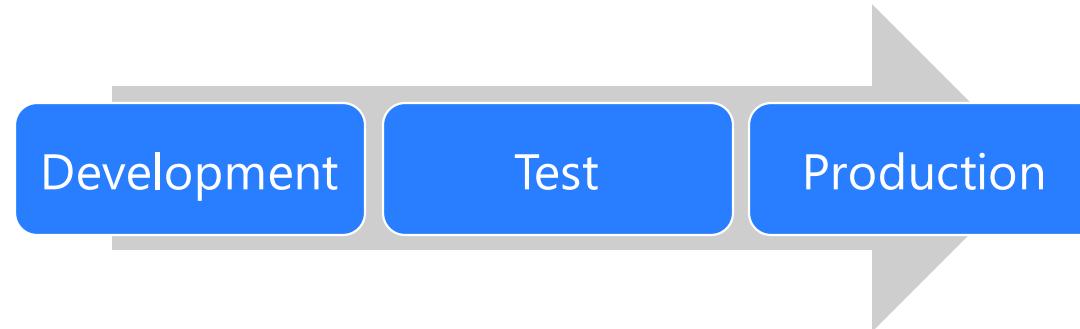


Support and info

Depth of coding required

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

Processes

Order Approval Flow

Device Procurement Process

Custom Connectors

Computer Vendor Ordering

Code Components

Order Status Visual

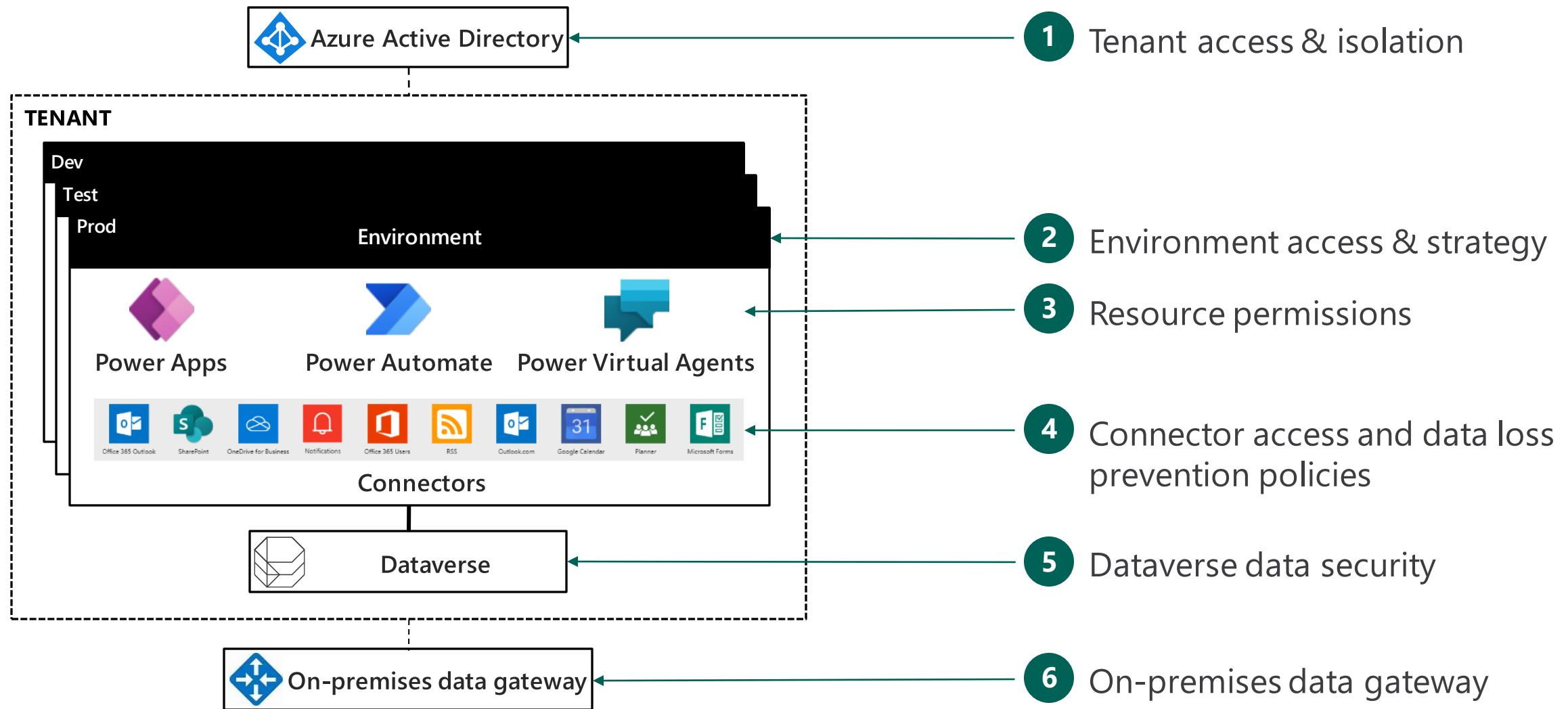
Display name ▾	Name	Type ▾
Calculate Ship Date ▾	... Calculate Ship Date	Process
Contoso CDS Model Driven User ▾	... Contoso CDS Model Driven User	Security Role
Contoso Device Order App ▾	... Contoso Device Order App	Security Role
Contoso Device Procurement App ▾	... Contoso Device Procurement App	Security 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

Manage solution components with Solution Explorer

The screenshot shows the Power Apps Solution Explorer interface. The top navigation bar includes 'Power Apps', a search bar, and an environment switcher set to 'Prioritz Dev'. The left sidebar has a search bar and lists categories: 'All (8)', 'Apps (2)', 'Chatbots (0)', 'Cloud flows (1)', 'Connection references (1)', and 'Tables (4)'. The main area displays a list of components under 'Prioritz > All'. The columns are 'Display name ↑', 'Name ↴', 'Type ↴', 'Ma...', and 'Last Modif...'. The listed components are:

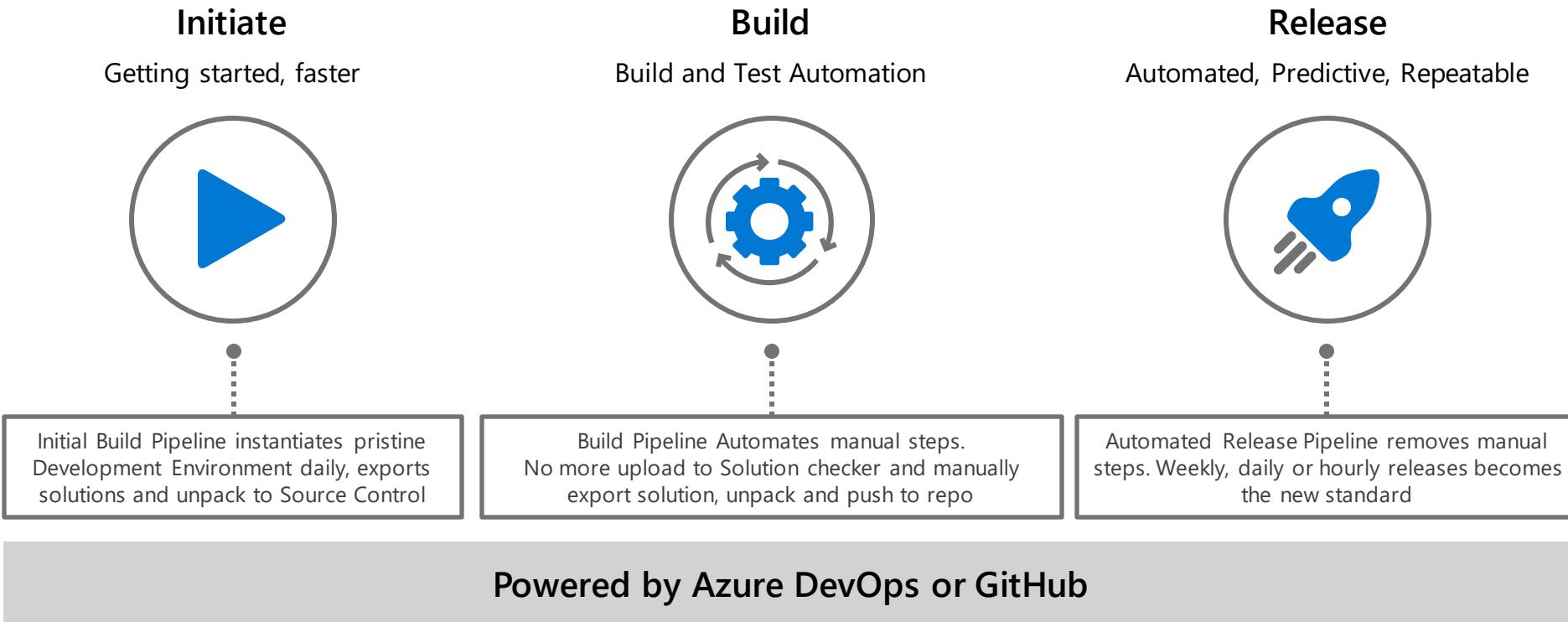
Display name ↑	Name ↴	Type ↴	Ma...	Last Modif...
Import sample data - Topics	Import sample data - Topics	Cloud Flow	No	4 weeks ago
Microsoft Dataverse Prioritz-5e308	contoso_sharedcommondataserviceforapps_5e308	Connection Reference	No	4 weeks ago
Prioritz Admin	contoso_prioritzadmin_1d21e	Canvas App	No	6 days ago
Prioritz Ask	contoso_prioritztask_fefd1	Canvas App	No	3 weeks ago
PrioritZ Topic	contoso_prioritztopic	Table	No	4 weeks ago
PrioritZ Topic Item	contoso_prioritztopicitem	Table	No	4 weeks ago
PrioritZ Topic Item Ranking	contoso_prioritztopicitemranking	Table	No	4 weeks ago
PrioritZ Topic Participant	contoso_prioritztopicparticipant	Table	No	4 weeks ago

A look at a tenant using Power Platform



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

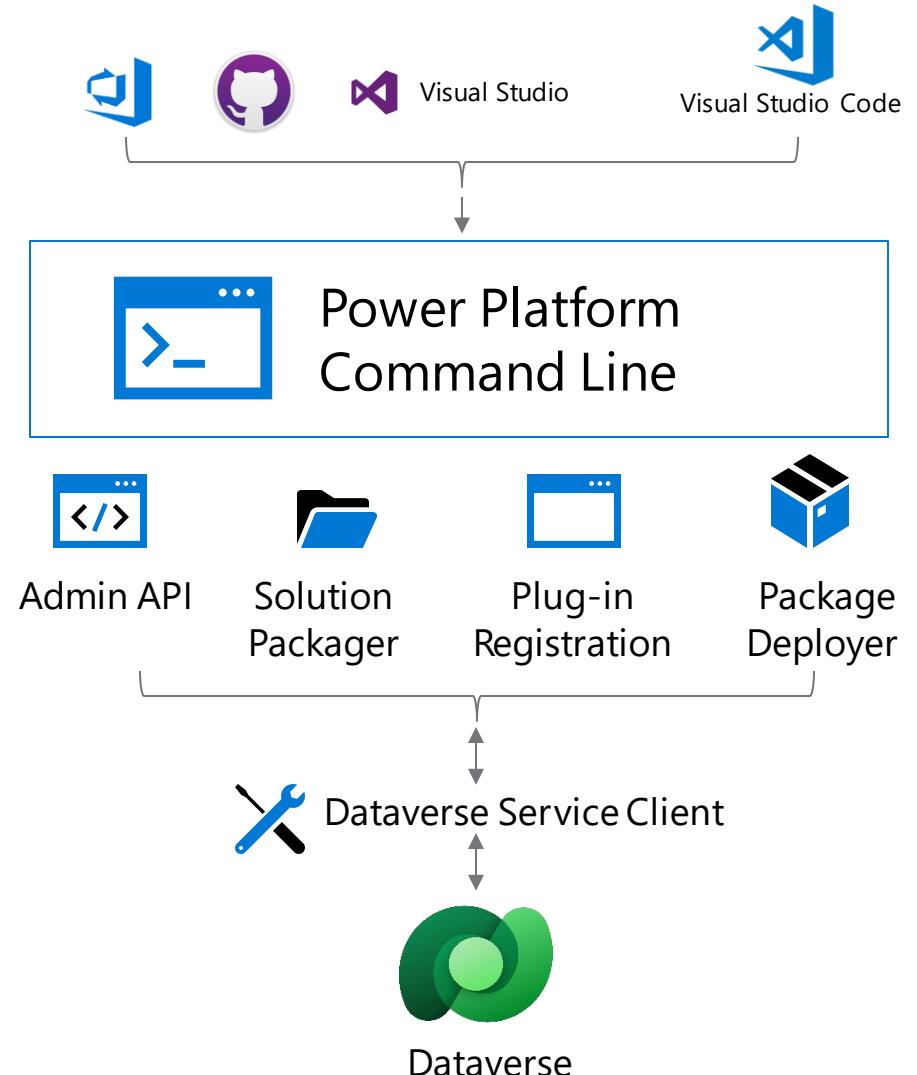
Single simplified interface

Common classes and interactions

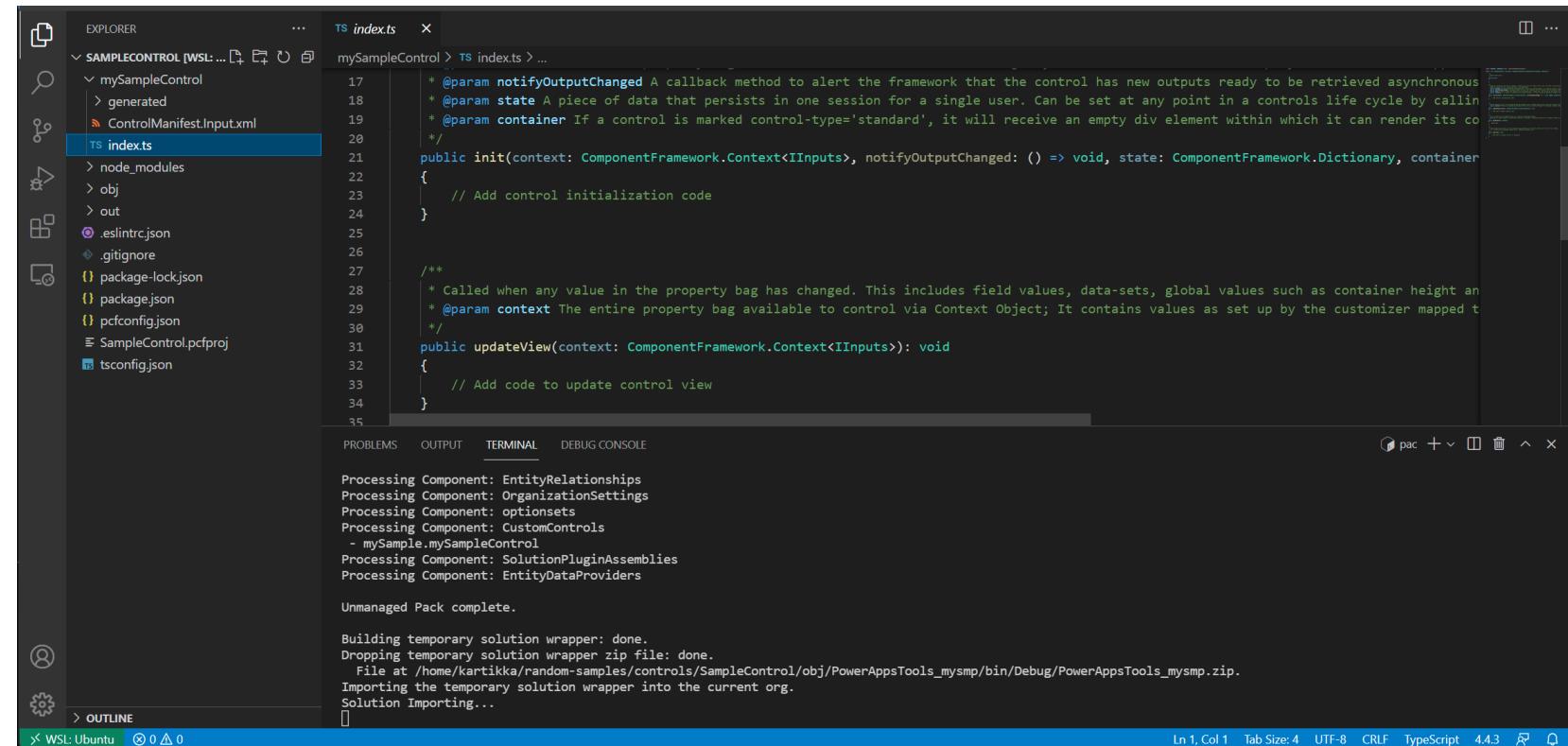
Ease of management for developer functions

Consistent CI/CD actions regardless of platform

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



Cross platform tooling support



The screenshot shows the Visual Studio Code interface with a dark theme. The Explorer sidebar on the left lists files and folders for a project named "SAMPLECONTROL". The "index.ts" file is open in the main editor area. The code is a TypeScript file with comments explaining methods like "init" and "updateView". Below the editor, the "OUTPUT" tab is selected, displaying logs of the build process for a PowerApp component named "mySampleControl". The logs show the component being processed, a temporary solution wrapper being built and dropped, and the temporary solution being imported into the current organization. At the bottom, the status bar shows "WSL: Ubuntu" and other system information.

- 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

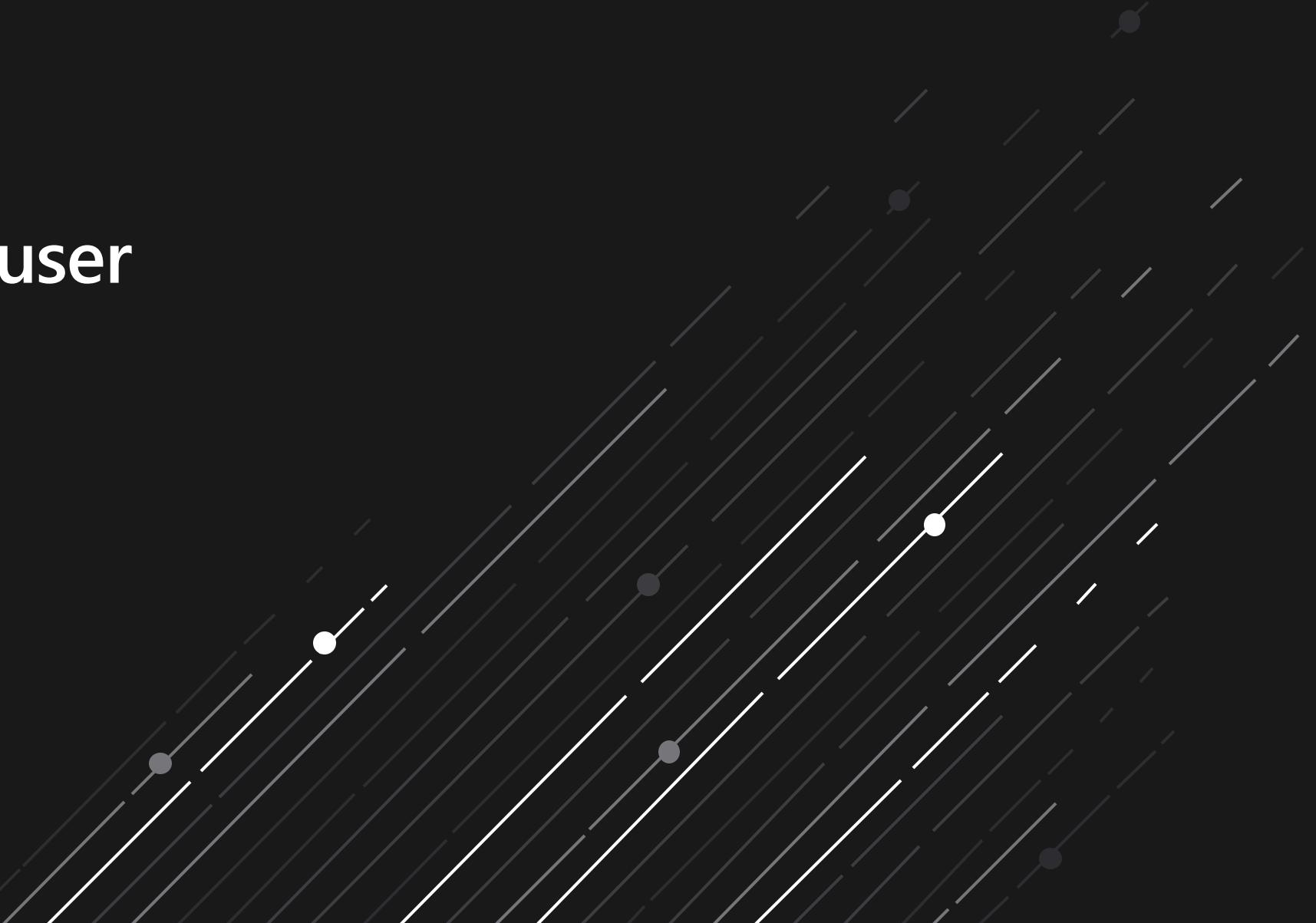
Make a small change to a table and use new column in an app

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

Extending the user experience



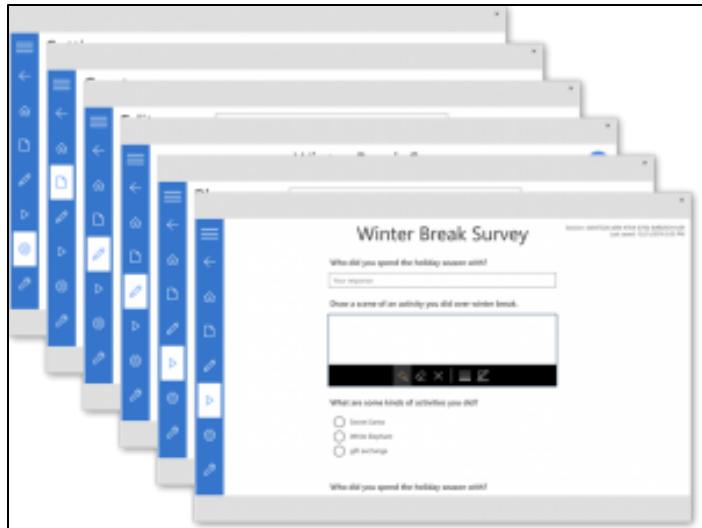
Types of components

Canvas Components

Targeted for canvas app makers

Built in Power Apps Studio using Power Fx

Works in canvas apps

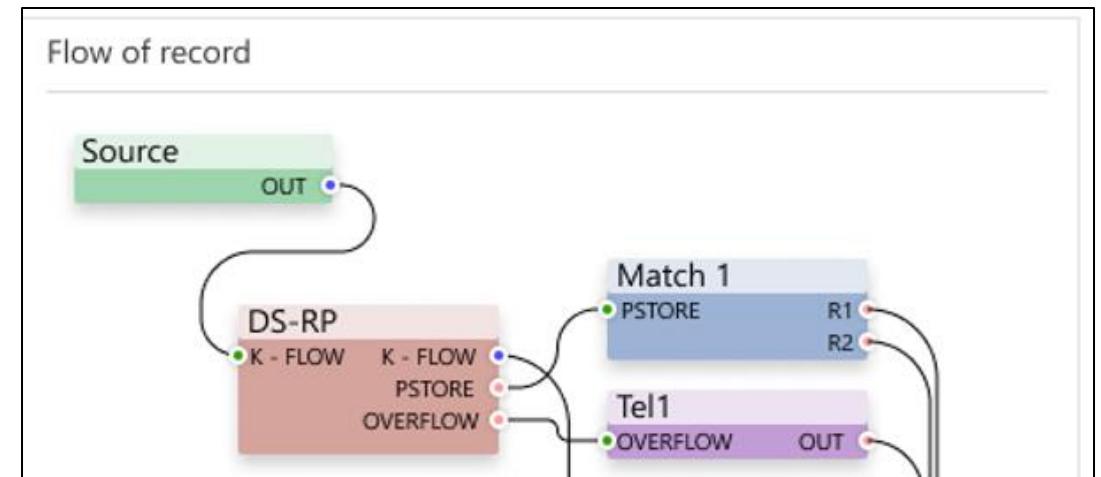


Power Apps Component Framework (PCF)

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>

PCF Gallery Categories Please select Search Authors Ideas About Twitter LinkedIn Facebook Submit a new PCF Control

The screenshot displays the Community PCF Gallery website, featuring a grid of 18 different PowerApps components and controls. Each component is shown in a separate card with a preview image, title, and edit/share icons.

- Checkmark Control**: A form with a checkmark icon next to each field.
- French Company Autocomplete**: A search interface for companies.
- Assign To Self Button**: A blue button labeled "Assign case to me".
- ShieldsIO Badge**: A badge showing various security and compliance logos.
- Account Form**: A form for creating accounts with fields for Account Name, Main Phone, and Address 1/ City.
- Password Input Control**: A password input field with placeholder text and a "Loading" indicator.
- Fluent UI Spinner**: A circular spinner with a blue and white design.
- Créer Compte**: A "Create Account" page with sections for "INFORMATIONS SUR LE COMPTE" and "SIRET".
- Lookup On Side Pane**: A side pane showing a list of items for selection.
- ABBOT**: A ticket management interface with tabs for Tickets, POFs, Informações, and Relacionados.
- ACCOUNT INFORMATION**: A section for managing account choices, including First Choice, Second Choice, Third Choice, and Fourth Choice.
- AlbanianXrm's MultiSwitch**: A control for managing multiple switch states.
- Input Formatter with Regular Expression**: A form field with validation logic highlighted.
- Codebug PayPal Button**: A button for processing payments via PayPal.
- Queue Item Tracker**: A timeline for tracking queue item changes.
- Modern Eats**: A restaurant promotional card with a photo and text.
- Codebug Sendgrid Template Viewer**: A viewer for email templates.
- SELECT SVG**: A graphic with the text "SELECT OBJECTS IN YOUR" overlaid on a map outline.

Scaffold code component using Power Platform CLI

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



The screenshot shows a code editor interface with a dark theme. On the left is the Explorer sidebar, which lists the project structure:

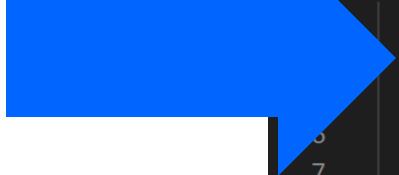
- devlearn (selected)
- └ FancyWidget
- └ generated
- └ ControlManifest.Input.xml
- TS HelloWorld.tsx
- TS index.ts
- > node_modules
- .eslintrc.json
- .gitignore
- devlearn.pcfproj
- { package-lock.json
- { package.json
- { pcfconfig.json
- ts tsconfig.json

A red box highlights the `ControlManifest.Input.xml` file in the Explorer. To the right is the Content Editor showing the XML manifest file:

```
<?xml version="1.0" encoding="utf-8" ?>
<manifest>
  <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1" >
    <!--external-service-usage node declares whether this 3rd party PCF control
        If it is not using any external service, please set the enabled="false"
    Example1:
    <external-service-usage enabled="true">
      <domain>www.Microsoft.com</domain>
    </external-service-usage>
    Example2:
    <external-service-usage enabled="false">
    </external-service-usage>
    -->
    <external-service-usage enabled="false">
      <!--UNCOMMENT TO ADD EXTERNAL DOMAINS
      <domain></domain>
      <domain></domain>
      -->
    </external-service-usage>
    <!-- property node identifies a specific, configurable piece of data that
        <property name="sampleProperty" display-name-key="Property_Display_Key">
    </property>
  </control>
</manifest>
```

Manifest defines the code component

Name, version
and type



```
devlearn > FancyWidget > ControlManifest.Input.xml
1  <?xml version="1.0" encoding="utf-8" ?>
2  <manifest>
3    <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1"
4      display-name-key="FancyWidget" description-key="FancyWidget description" control-type="virtual" >
5        <external-service-usage enabled="false"></external-service-usage>
6        <!-- property node identifies a specific, configurable piece of data that the control expects from the host application -->
7        <property name="sampleProperty" display-name-key="Property_Display_Key"
8          description-key="Property_Desc_Key" of-type="SingleLine.Text" usage="bound" required="true" />
9
10       <resources>
11         <code path="index.ts" order="1"/>
12         <platform-library name="React" version="16.8.6" />
13         <platform-library name="Fluent" version="8.29.0" />
14         <css path="css/FancyWidget.css" order="1" />
15         <resx path="strings/FancyWidget.1033.resx" version="1.0.0" />
16       </resources>
17
18       <feature-usage>
19         <uses-feature name="Device.captureAudio" required="true" />
20         <uses-feature name="Device.captureImage" required="true" />
21         <uses-feature name="Device.captureVideo" required="true" />
22         <uses-feature name="Device.getBarcodeValue" required="true" />
23         <uses-feature name="Device.getCurrentPosition" required="true" />
24         <uses-feature name="Device.pickFile" required="true" />
25         <uses-feature name="Utility" required="true" />
26         <uses-feature name="WebAPI" required="true" />
27       </feature-usage>
28
29     </control>
30   </manifest>
31
```

Manifest defines the code component - Properties

Properties exposed to hosting app



```
devlearn > FancyWidget > ControlManifest.Input.xml
1  <?xml version="1.0" encoding="utf-8" ?>
2  <manifest>
3    <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1"
4      display-name-key="FancyWidget" description-key="FancyWidget description" control-type="virtual" >
5
6      <external-service-usage enabled="false"></external-service-usage>
7      <!-- property node identifies a specific, configurable piece of data that the control expects from the
8          host application -->
9      <property name="sampleProperty" display-name-key="Property_Display_Key"
10        description-key="Property_Desc_Key" of-type="SingleLine.Text" usage="bound" required="true" />
11
12    <resources>
13      <code path="index.ts" order="1"/>
14      <platform-library name="React" version="16.8.6" />
15      <platform-library name="Fluent" version="8.29.0" />
16      <css path="css/FancyWidget.css" order="1" />
17      <resx path="strings/FancyWidget.1033.resx" version="1.0.0" />
18    </resources>
19
20    <feature-usage>
21      <uses-feature name="Device.captureAudio" required="true" />
22      <uses-feature name="Device.captureImage" required="true" />
23      <uses-feature name="Device.captureVideo" required="true" />
24      <uses-feature name="Device.getBarcodeValue" required="true" />
25      <uses-feature name="Device.getCurrentPosition" required="true" />
26      <uses-feature name="Device.pickFile" required="true" />
27      <uses-feature name="Utility" required="true" />
28      <uses-feature name="WebAPI" required="true" />
29    </feature-usage>
30  </control>
31 </manifest>
```

Manifest defines the code component - Resources

Resource files that
the component needs

```
devlearn > FancyWidget > ControlManifest.Input.xml
1  <?xml version="1.0" encoding="utf-8" ?>
2  <manifest>
3    <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1"
4      display-name-key="FancyWidget" description-key="FancyWidget description" control-type="virtual" >
5
6      <external-service-usage enabled="false"></external-service-usage>
7      <!-- property node identifies a specific, configurable piece of data that the control expects from the host application -->
8      <property name="sampleProperty" display-name-key="Property_Display_Key"
9        description-key="Property_Desc_Key" of-type="SingleLine.Text" usage="bound" required="true" />
10
11     <resources>
12       <code path="index.ts" order="1"/>
13       <platform-library name="React" version="16.8.6" />
14       <platform-library name="Fluent" version="8.29.0" />
15       <css path="css/FancyWidget.css" order="1" />
16       <resx path="strings/FancyWidget.1033.resx" version="1.0.0" />
17     </resources>
18
19     <feature-usage>
20       <uses-feature name="Device.captureAudio" required="true" />
21       <uses-feature name="Device.captureImage" required="true" />
22       <uses-feature name="Device.captureVideo" required="true" />
23       <uses-feature name="Device.getBarcodeValue" required="true" />
24       <uses-feature name="Device.getCurrentPosition" required="true" />
25       <uses-feature name="Device.pickFile" required="true" />
26       <uses-feature name="Utility" required="true" />
27       <uses-feature name="WebAPI" required="true" />
28     </feature-usage>
29   </control>
30 </manifest>
31
```

Manifest defines the code component - Features

```
devlearn > FancyWidget > ControlManifest.Input.xml
1  <?xml version="1.0" encoding="utf-8" ?>
2  <manifest>
3    <control namespace="DevInADay" constructor="FancyWidget" version="0.0.1"
4      display-name-key="FancyWidget" description-key="FancyWidget description" control-type="virtual" >
5
6      <external-service-usage enabled="false"></external-service-usage>
7      <!-- property node identifies a specific, configurable piece of data that the control expects from the host application -->
8      <property name="sampleProperty" display-name-key="Property_Display_Key"
9        description-key="Property_Desc_Key" of-type="SingleLine.Text" usage="bound" required="true" />
10
11     <resources>
12       <code path="index.ts" order="1"/>
13       <platform-library name="React" version="16.8.6" />
14       <platform-library name="Fluent" version="8.29.0" />
15       <css path="css/FancyWidget.css" order="1" />
16       <resx path="strings/FancyWidget.1033.resx" version="1.0.0" />
17     </resources>
18
19     <feature-usage>
20       <uses-feature name="Device.captureAudio" required="true" />
21       <uses-feature name="Device.captureImage" required="true" />
22       <uses-feature name="Device.captureVideo" required="true" />
23       <uses-feature name="Device.getBarcodeValue" required="true" />
24       <uses-feature name="Device.getCurrentPosition" required="true" />
25       <uses-feature name="Device.pickFile" required="true" />
26       <uses-feature name="Utility" required="true" />
27       <uses-feature name="WebAPI" required="true" />
28     </feature-usage>
29   </control>
30 </manifest>
31
```

Enable component features



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. 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. 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

PowerApps component framework Test Environment

Hello, World!

FancyWidget

✓ **Context Inputs**

Form Factor

Component Container Width	Component Container Height
<input type="text" value="300"/>	<input style="border: 1px solid black;" type="text" value="300"/>

✓ **Data Inputs**

Property sampleProperty

Value

Type SingleLine.Text

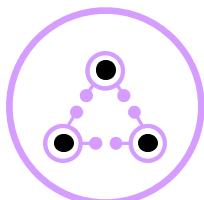
Innovate at speed with express design



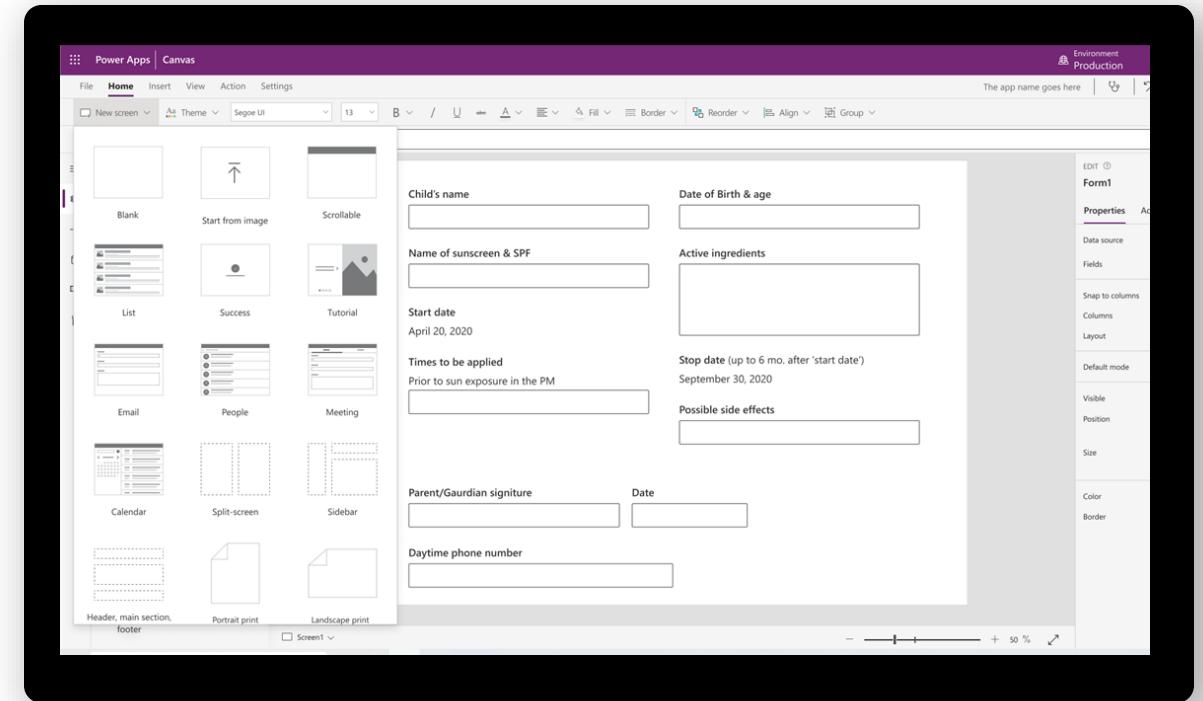
Jumpstart the app development process and empower anyone to easily create new apps



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

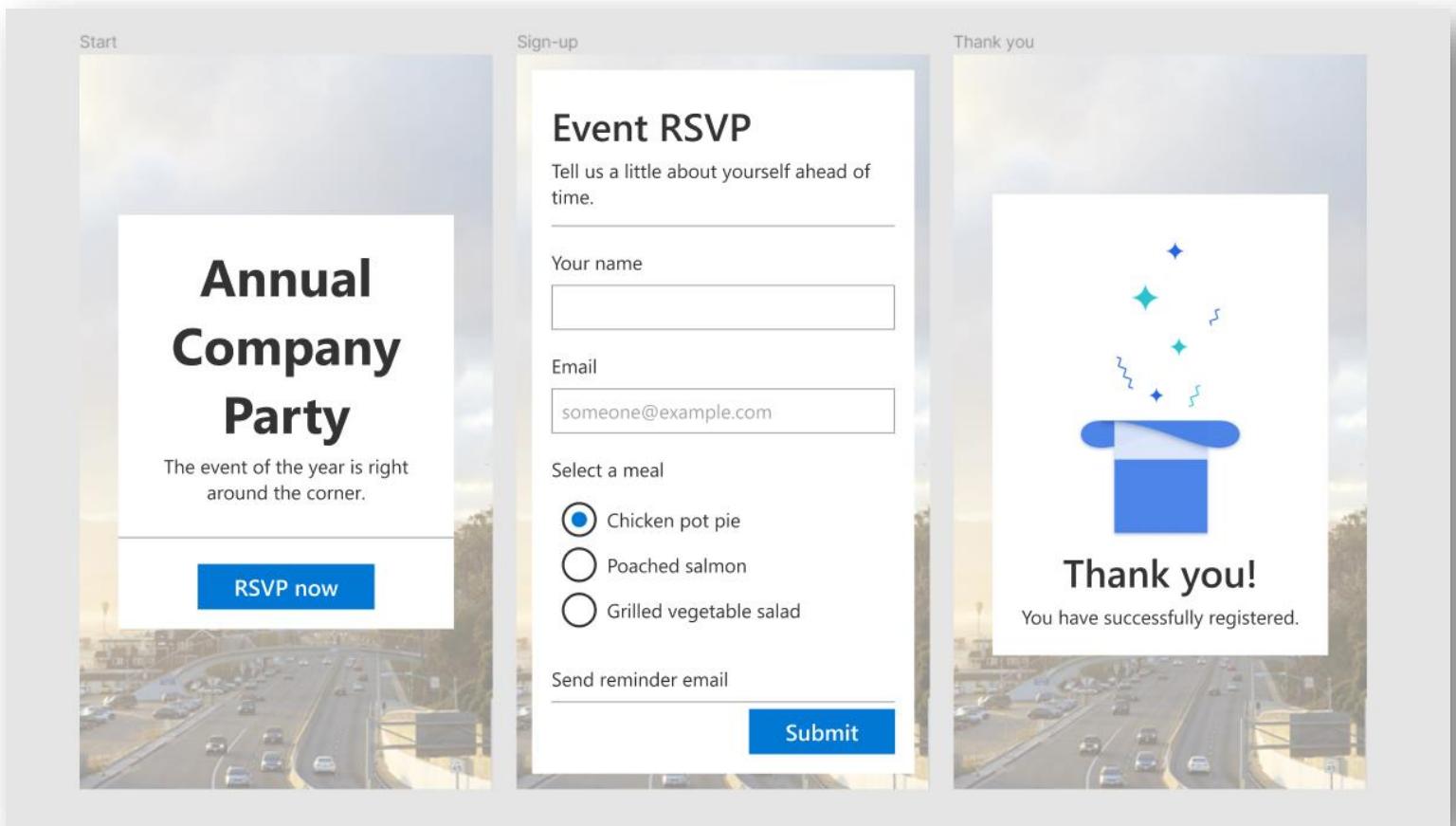
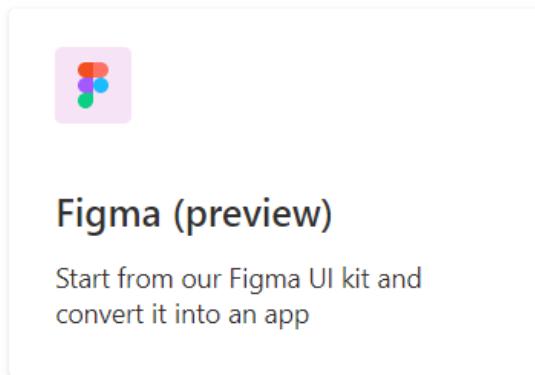


Build an app in a few steps by uploading your content and connecting to your data



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

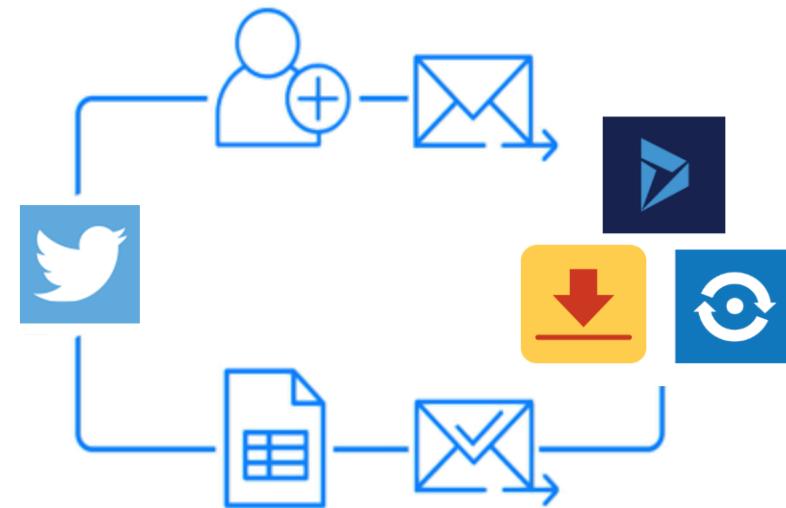
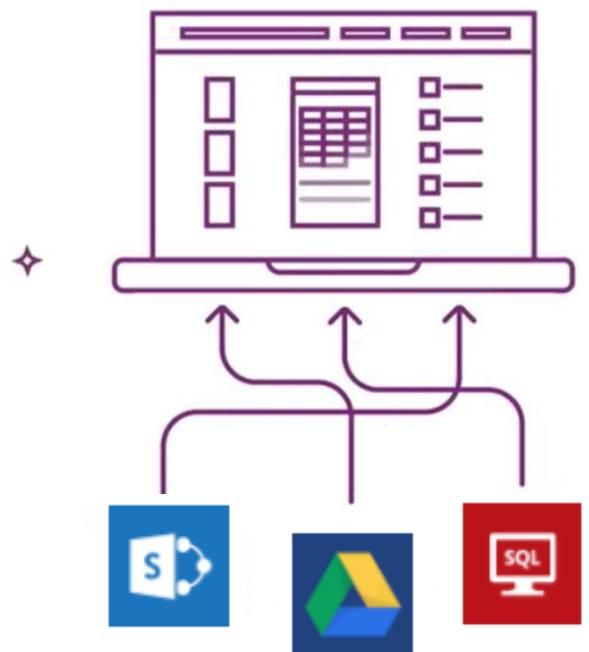
Wrapping up

- 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

Custom Connectors

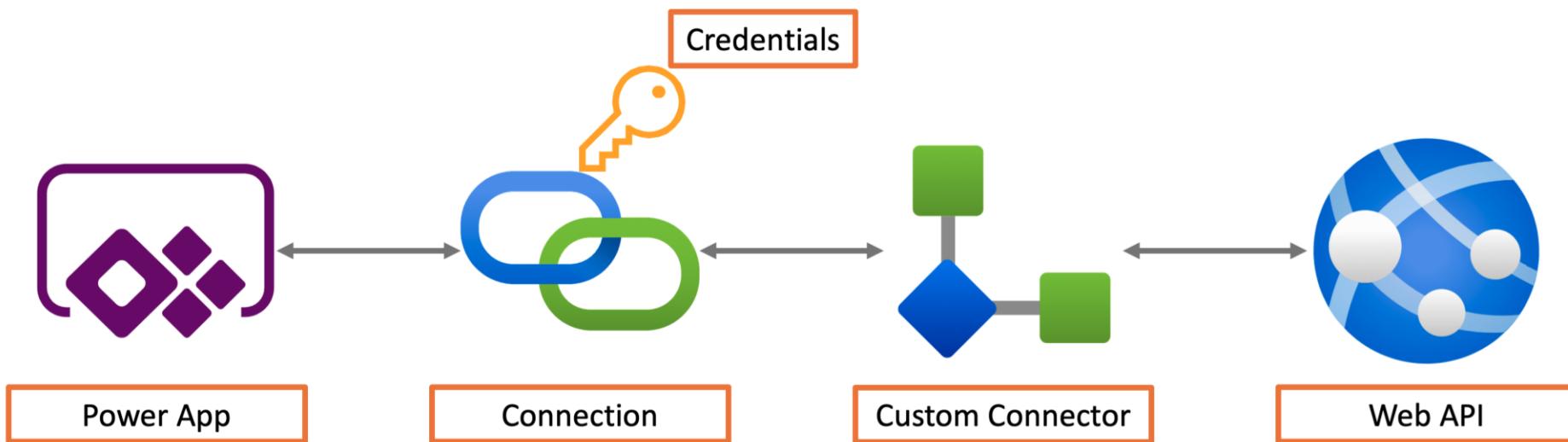


A **connector** is a wrapper around an API that allows the underlying service to talk to **Power Automate** and **Power Apps**



Connector vs Connection

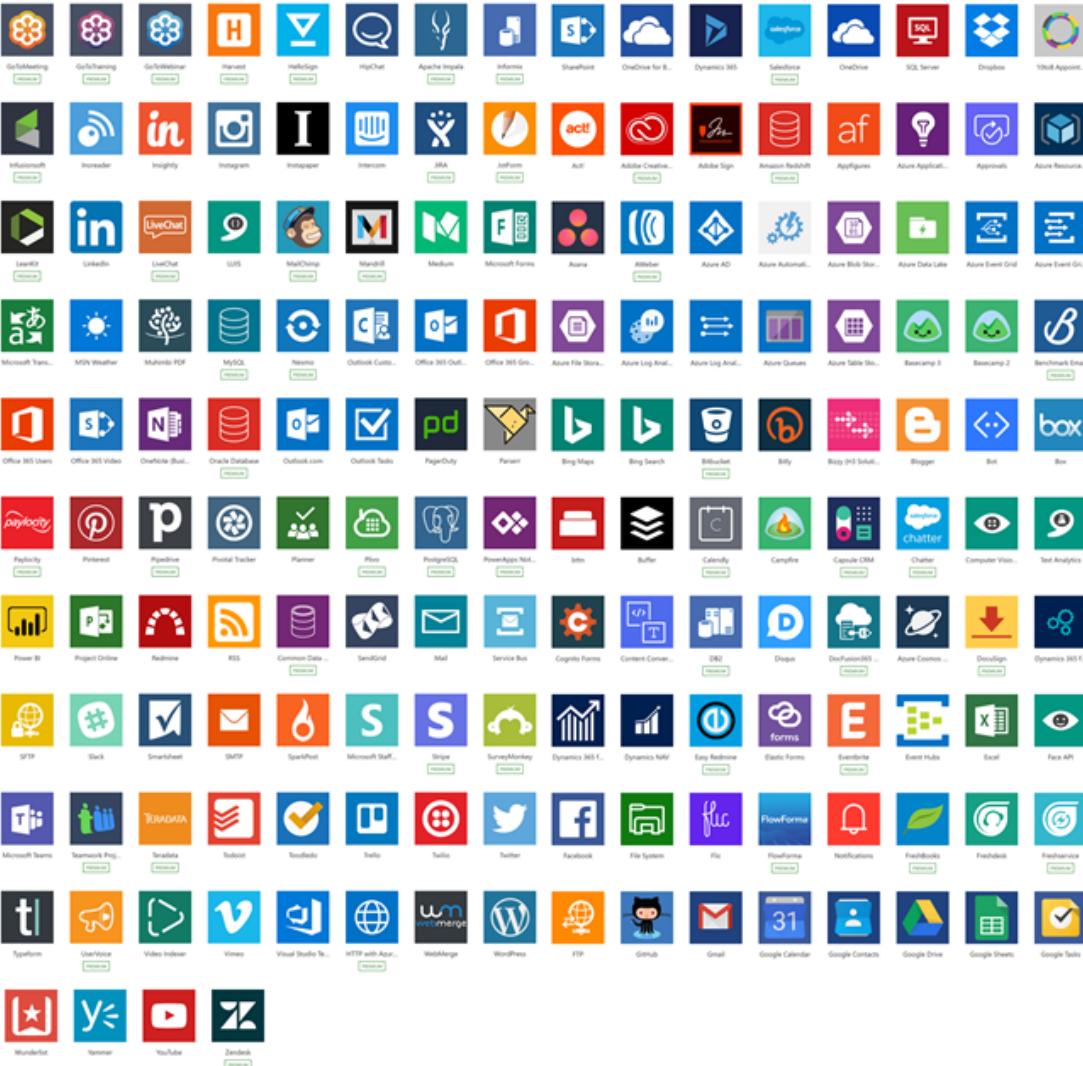
- Connector knows about the Web API's host and operation details
- Connection knows the credentials and has a reference to the connector to facilitate communication with the Web API



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

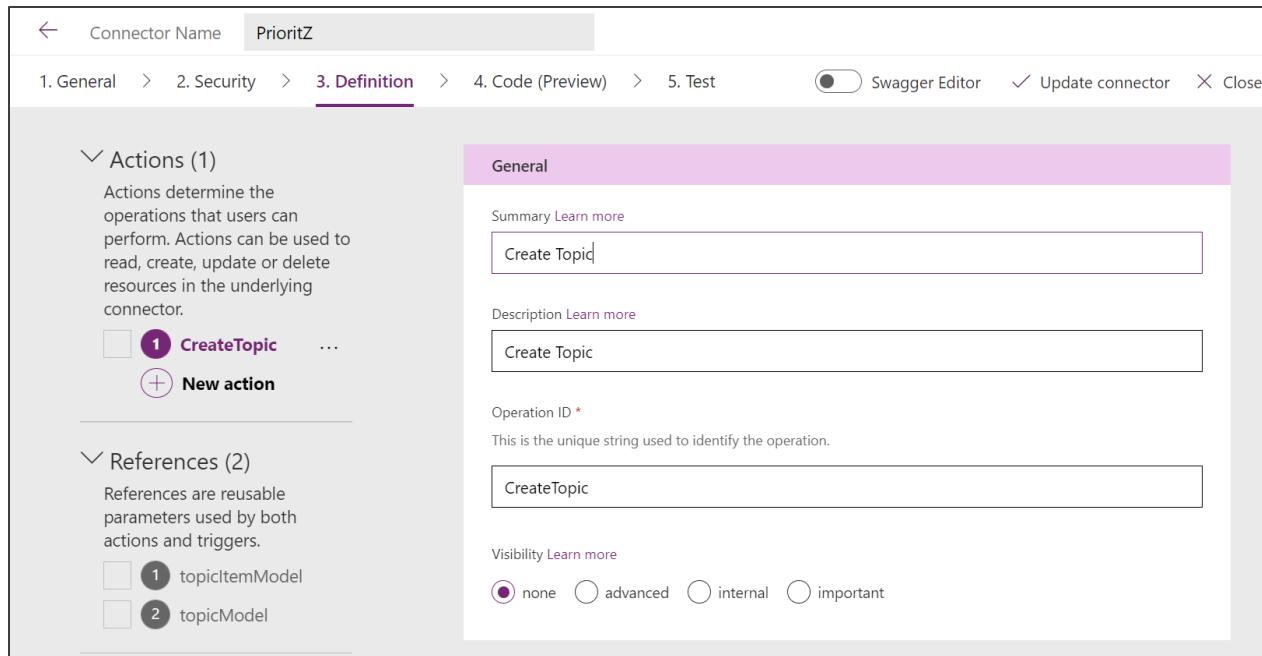


Develop a custom connector



- REST API
- Public or private
- 3rd party or custom built
- Build with Azure
 - Functions
 - Web Apps
 - API Apps
- Anonymous
- Basic
- API Key
- OAuth 2.0 (Cloud only)
- Windows (On-premises)
- 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

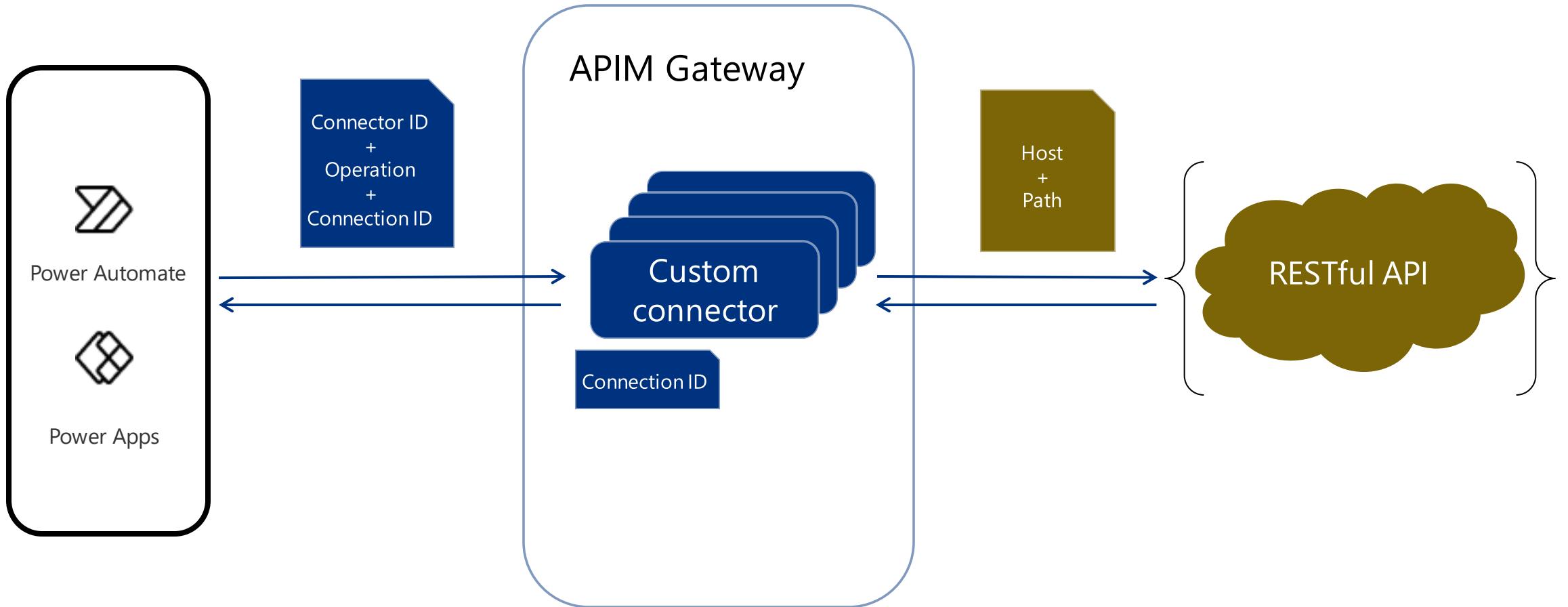


The screenshot shows the Azure Functions connector designer interface. At the top, it says "Connector Name: PrioritZ". Below that is a navigation bar with steps: 1. General > 2. Security > 3. Definition > 4. Code (Preview) > 5. Test. A "Swagger Editor" toggle switch is turned on. To its right are "Update connector" and "Close" buttons. The main area is titled "General". It contains sections for "Summary" (with a "Create Topic" input field), "Description" (with a "Create Topic" input field), "Operation ID" (a text input field containing "CreateTopic"), and "Visibility" (radio buttons for "none" (selected), "advanced", "internal", and "important"). On the left sidebar, under "Actions (1)", there's a description of actions and a list with one item: "1 CreateTopic" (selected). There's also a "New action" button. Under "References (2)", there are descriptions of references and a list with two items: "topicItemModel" (selected) and "topicModel".

```
1 swagger: '2.0'
2 info: {title: OpenAPI Document on Azure Functions, version: 1.0.0}
3 host: prioritzy.azurewebsites.net
4 basePath: /api
5 schemes: [https]
6 consumes: []
7 produces: []
8 paths:
9   /CreateTopic:
10  get:
11    tags: [name]
12    operationId: CreateTopic
13    consumes: [application/json]
14    produces: [text/plain]
15    parameters:
16      - in: query
        name: name
        description: The **Name** parameter, required: true,
        type: string
      - in: body
        name: body
        schema: {$ref: '#/definitions/topicModel'}
17    responses:
18      '200':
19        description: The OK response
        schema: {type: string}
20        summary: Create Topic
21        description: Create Topic
22    definitions:
23      topicItemModel:
24        type: object
25        properties:
26          choice: {type: string}
27          photo: {format: base64, type: string}
28      topicModel:
29        type: object
30        properties:
```

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

Connector architecture (no authentication)



Use policy templates to modify behavior

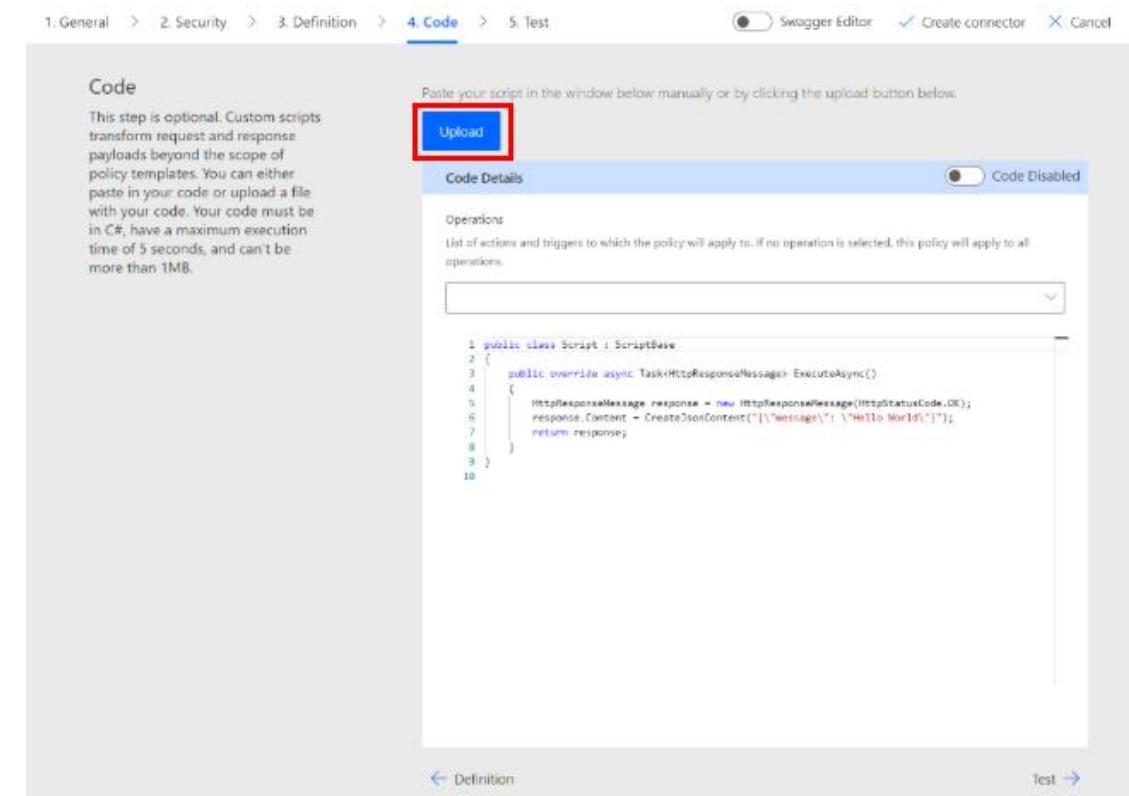
- Set host URLs
- Route requests
- Set HTTP header values
- Set query string parameters

The screenshot shows a user interface for creating a policy. At the top, there's a pink header bar labeled "Policy details". Below it, there are two input fields: "Name *" and "Template * Learn more". The "Template" field has a red exclamation mark icon in its top right corner. A dropdown menu is open over the "Template" field, listing various policy templates. The visible options include:

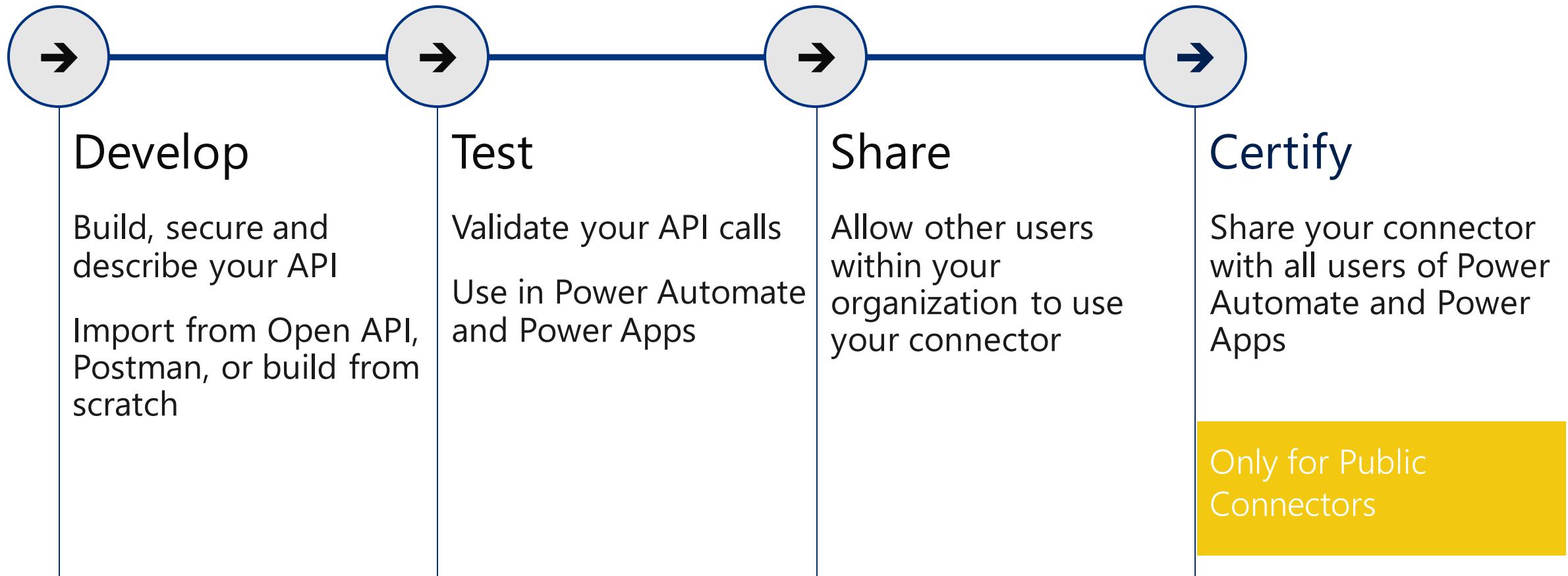
- Choose a template
- Convert an array to an object (Preview)
- Convert an object to an array (Preview)
- Set host URL
- Route request
- Set connection status to unauthenticated (Preview)
- Set HTTP header
- Set property (Preview)
- Set query string parameter
- Set header/query parameter value from URL (Preview)
- Convert delimited string into array of objects (Preview)

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



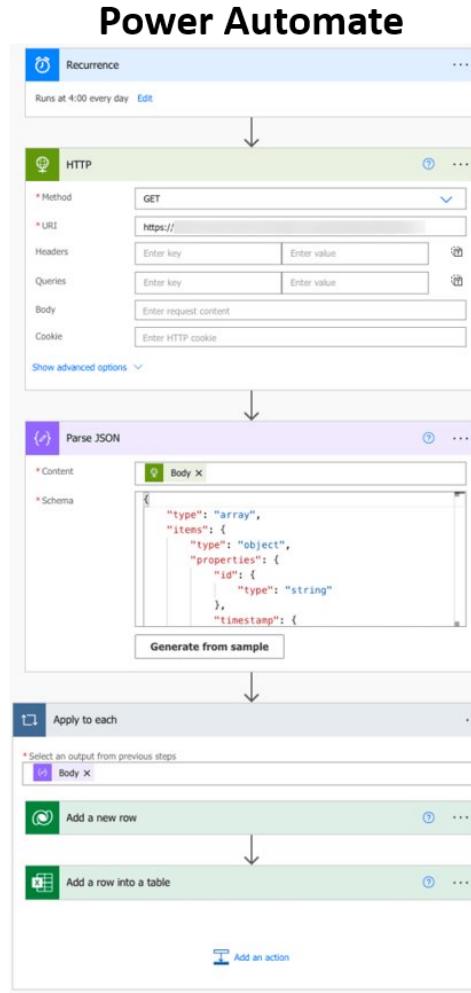
Making a connector public



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?



Azure Functions



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

Why use Custom Connectors over HTTP Requests?



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

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

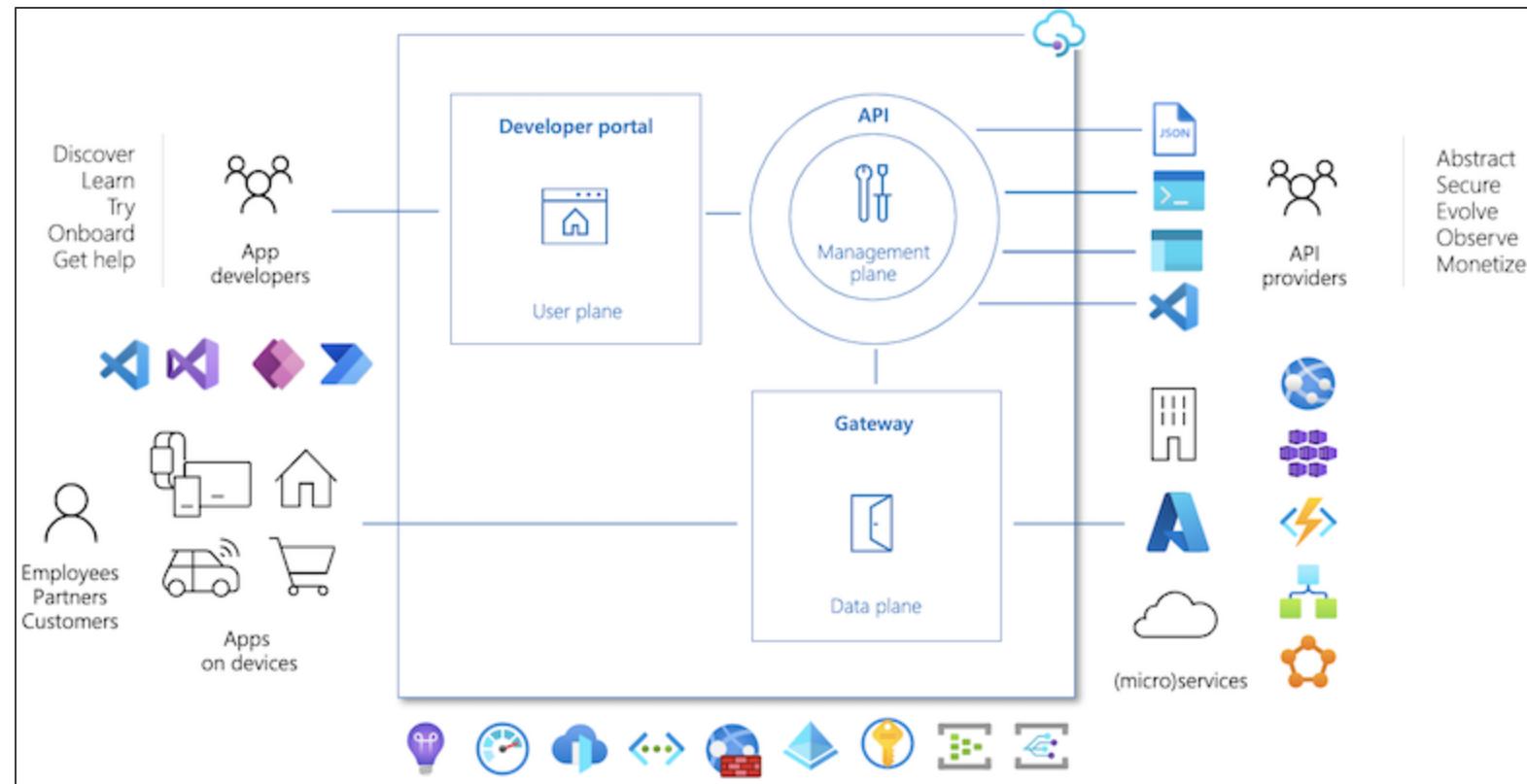
Document your API for connector use

- 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

The screenshot shows the Azure portal interface for a Function App named "PrioritZAPI". The left sidebar contains navigation links for TLS/SSL settings, Networking, Scale up (App Service plan), Scale out, Push, Properties, Locks, App Service plan, Quotas, Change App Service plan, Development Tools (Console, Advanced Tools, App Service Editor (Preview), Extensions), and API (API Management). The "API Management" link in the API section is highlighted with a red box. The main content area is titled "API Management" and includes a sub-header "Expose your HTTP trigger Functions through Azure API Management - manage, publish, and monitor them from one central location". It features fields for "API Management" (with a "Create new" button) and "API" (with a dropdown menu). A checkbox labeled "Import Functions" is checked and highlighted with a red box. Below it is an unchecked checkbox for "Enable Application Insights". A "Link API" button is also present.

Create connector from Azure API Management

contosoconferenceapi | Power Platform

API Management service

Search (Ctrl+ /) <<

Settings

Properties

Locks

APIs

APIs

Products

Subscriptions

Named values

Backends

API Tags

Schemas

Power Platform

OAuth 2.0 + OpenID Connect



Amplify your APIs with the Power Platform

Leverage your APIs by connecting them with interactive applications and automated workflows in just a few clicks. [Learn more](#)

[Create a connector](#)

 PowerApps

Transform your data into interactive applications with PowerApps, which can be shared and used on any device. [Learn more](#)

 Power Automate

Construct automated workflows and approval processes for anyone in your organization in minutes across hundreds of popular apps and services with Power Automate. [Learn more](#)

 Explore free training from Microsoft

Learn how to work together to meet challenges effectively with Microsoft Power Platform by analyzing data, building apps, automating processes, and creating virtual agents

[Build an app](#)

[Automate your workflows](#)

[Explore](#)

Create the connector

Create a connector

API
Select an API to connect to the Power Platform.

API *

Conference API

PowerApps
Select a PowerApps environment to publish your API to and create a name for your connector to display in the Power Platform.

PowerApps environment to publish to * ⓘ

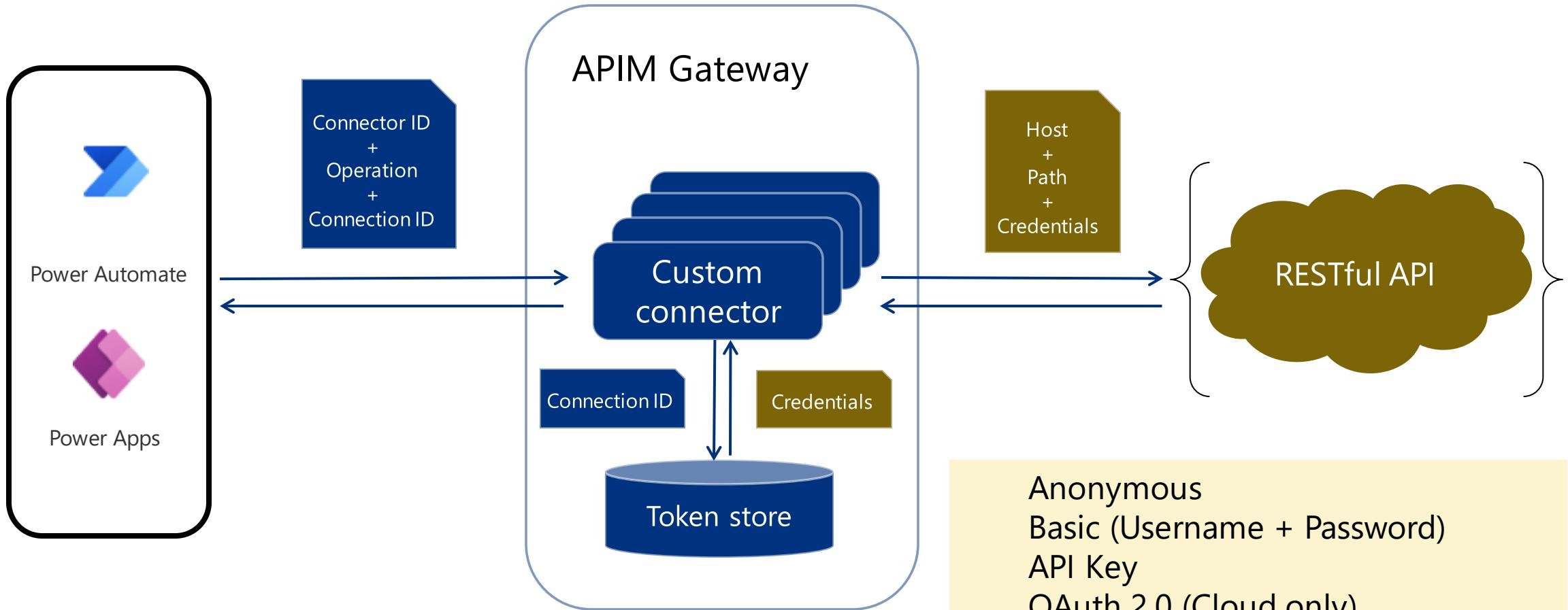
Prioritz Dev (prioritzdev)

API display name * ⓘ

Conference API

Create subscription key connection parameter

Securing connector & API



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

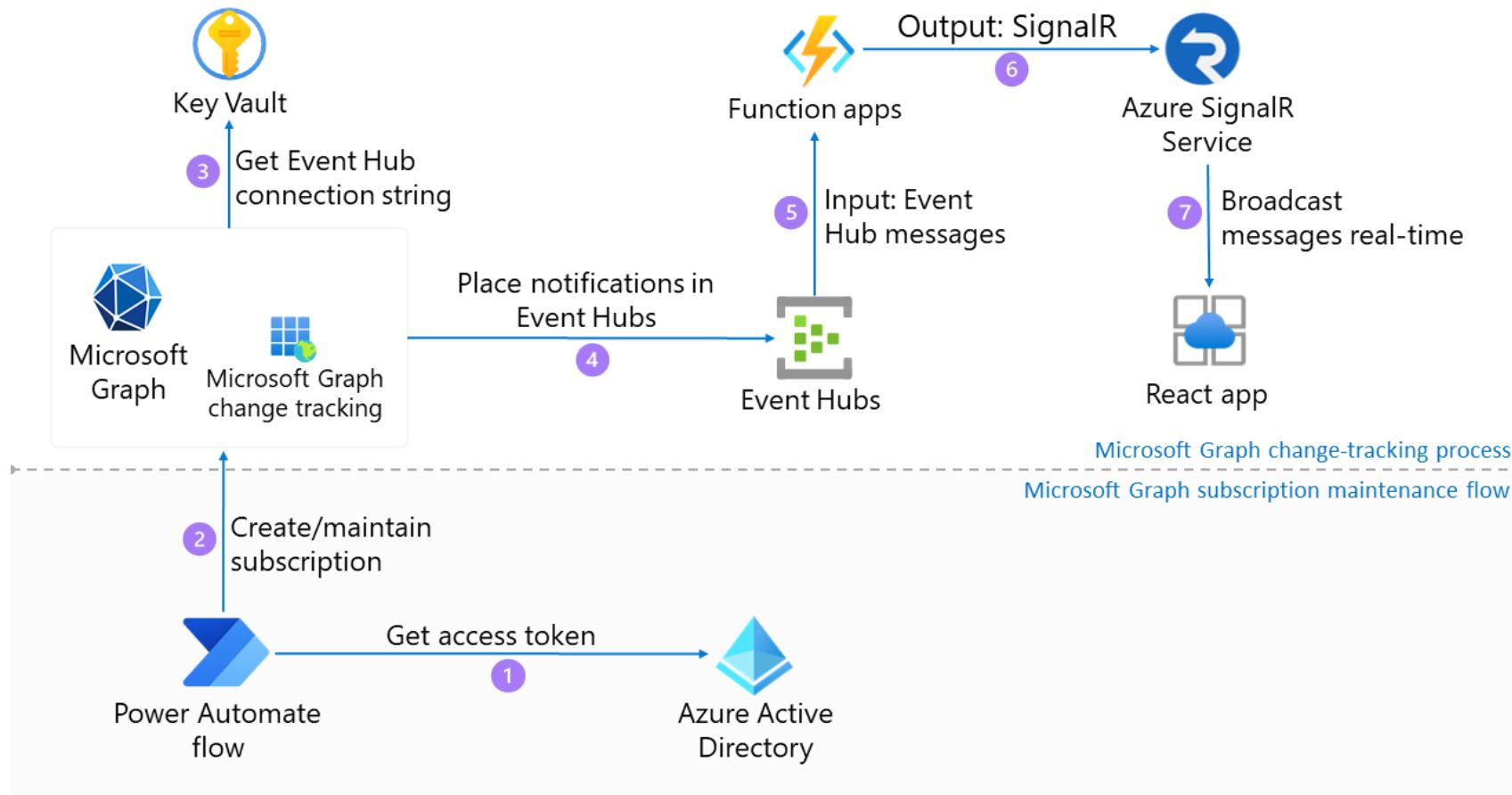
Checklist for securing connectors with Azure AD

- 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

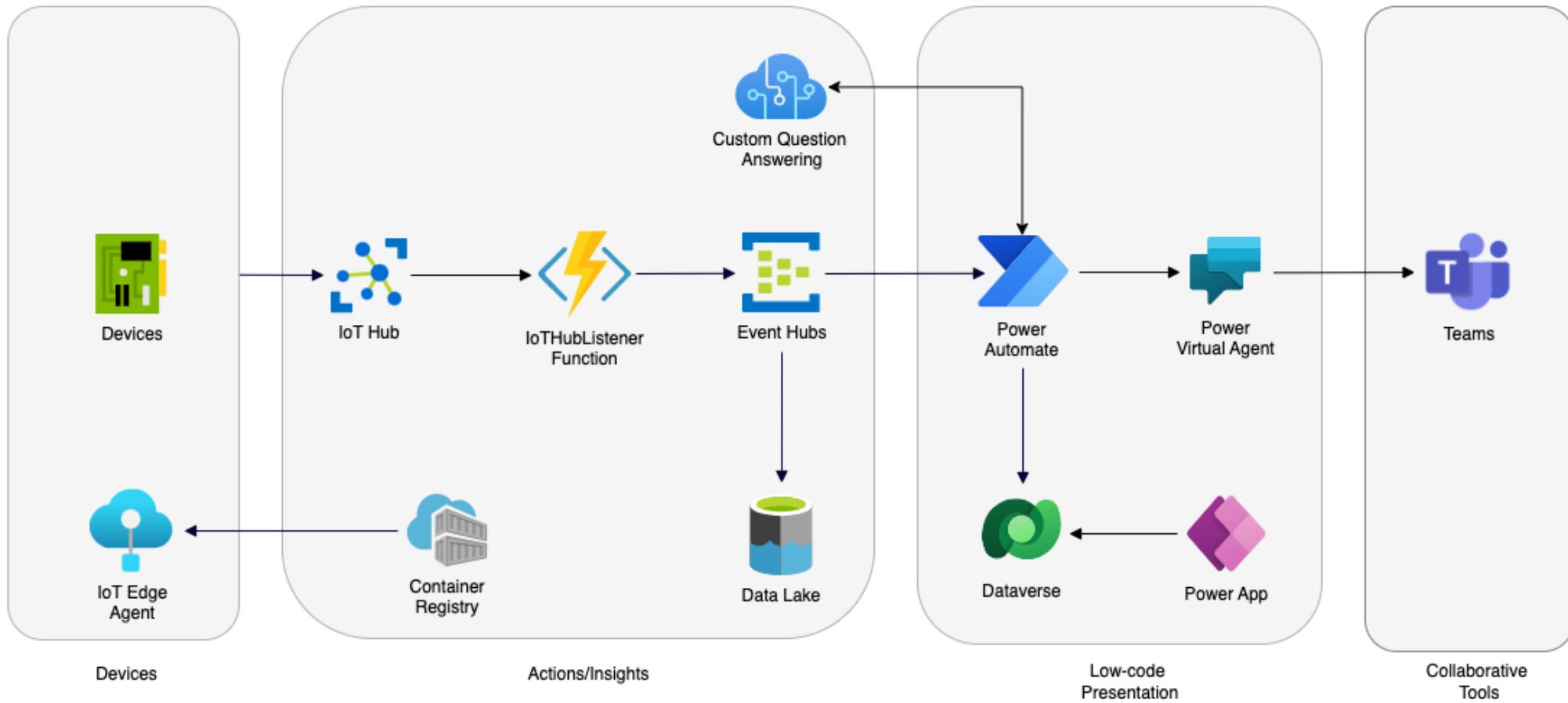
- Consider how the API will be consumed by Power Fx
- Complex object structures might be too complicated for makers
- Collaborate on how errors will be handled

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

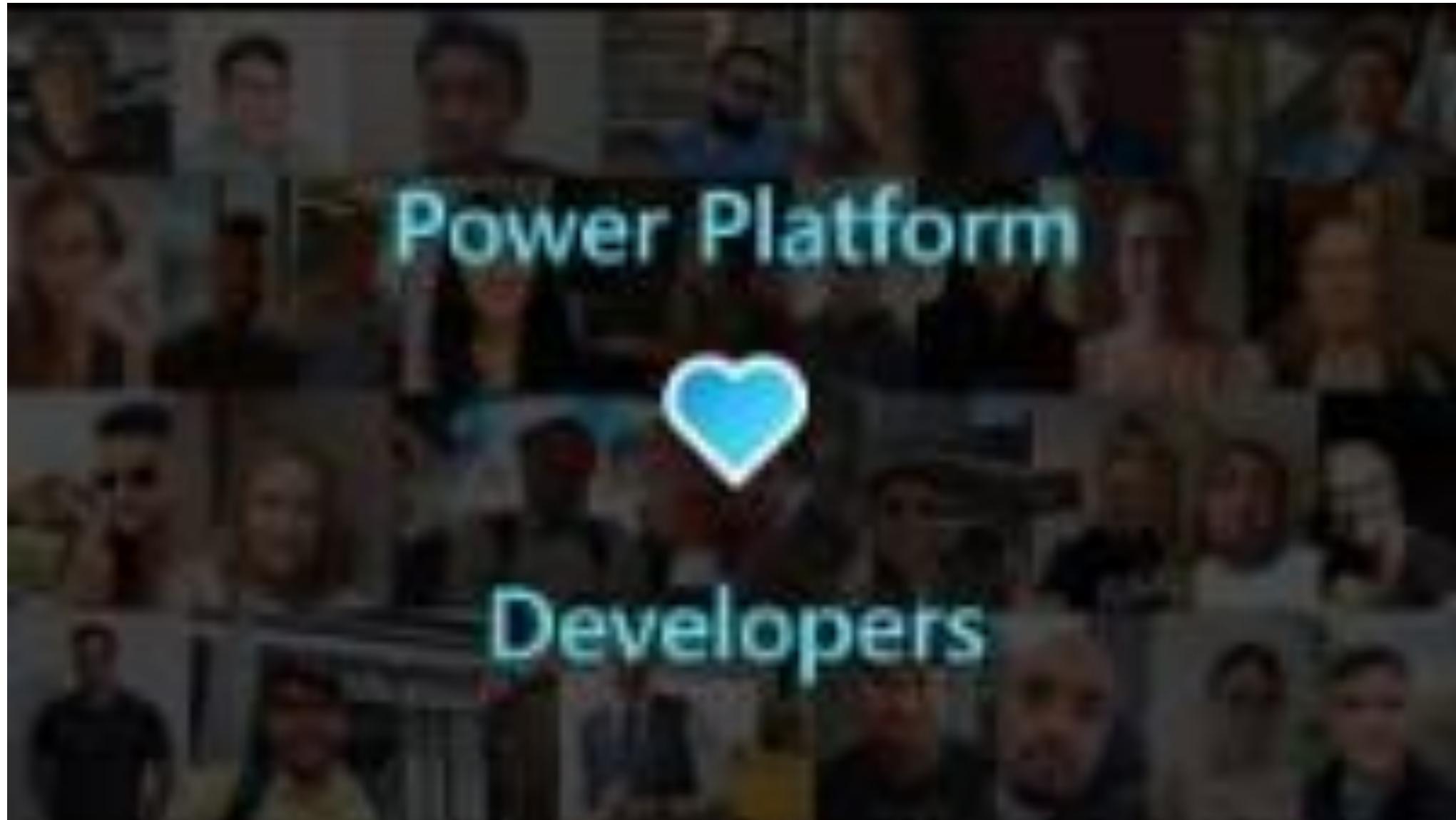
Wrapping up

- 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

Low code or code first

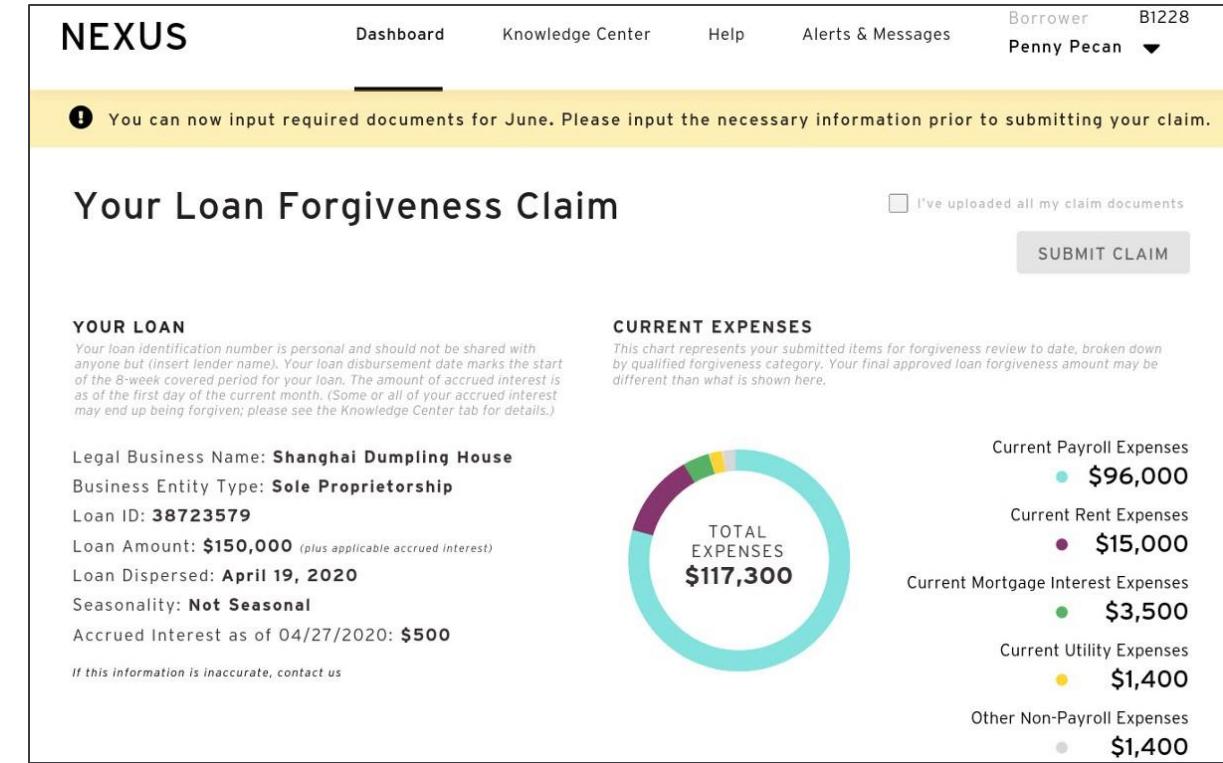


Meet professional developers using Microsoft Power Platform



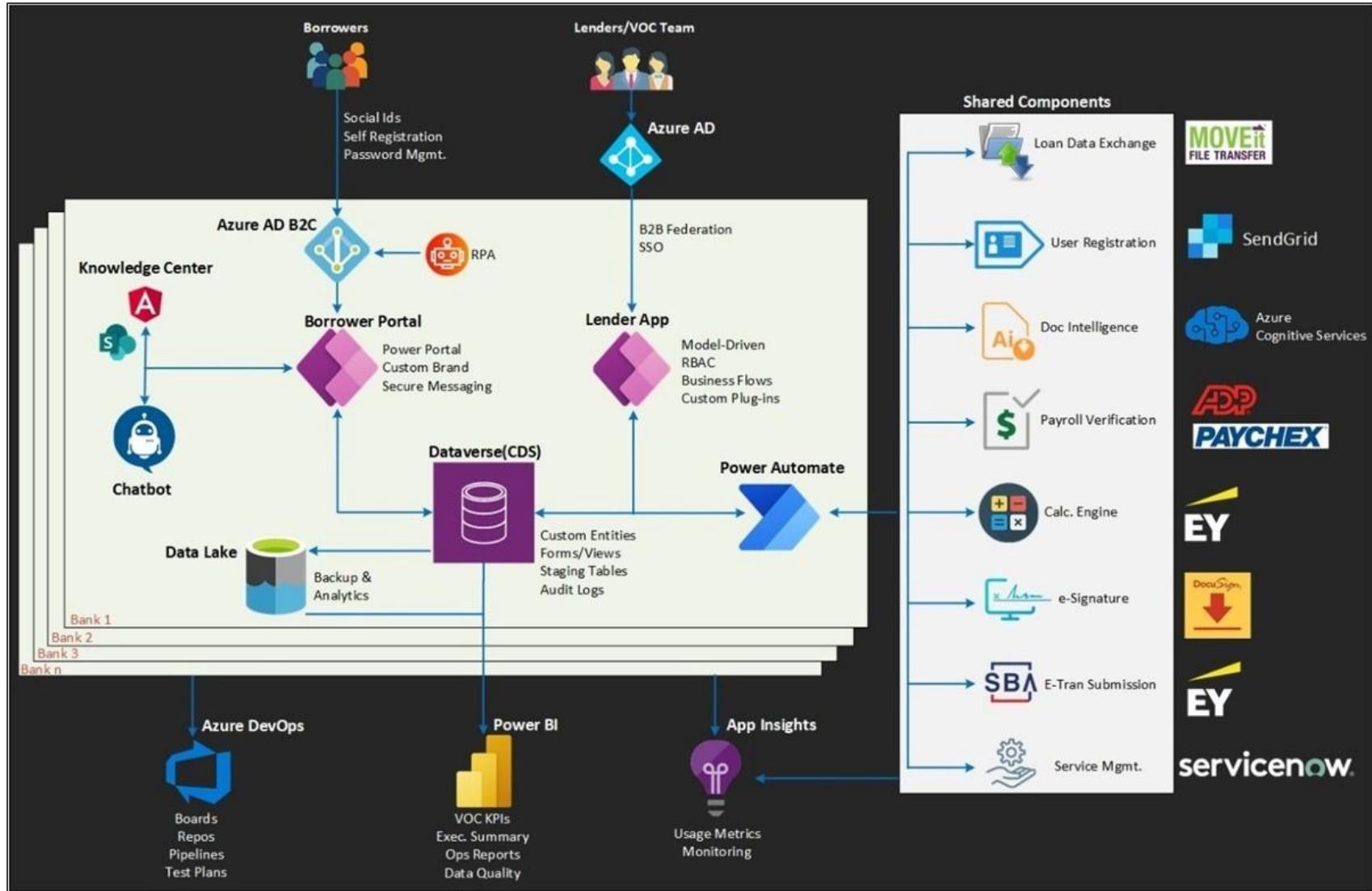
Ernst & Young processes emergency loans for small business

- 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



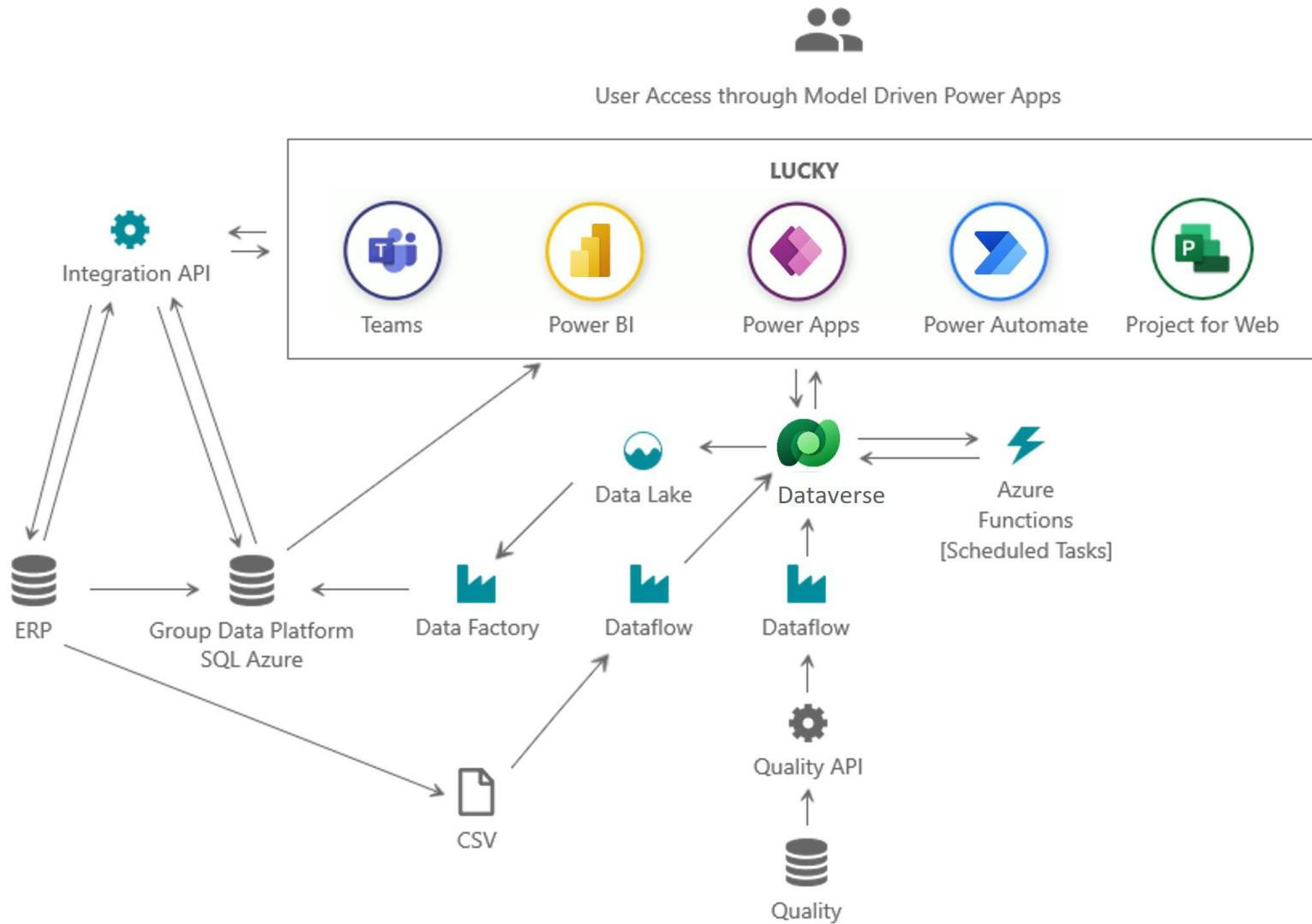
Examples of pro-dev / Azure integration

- 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

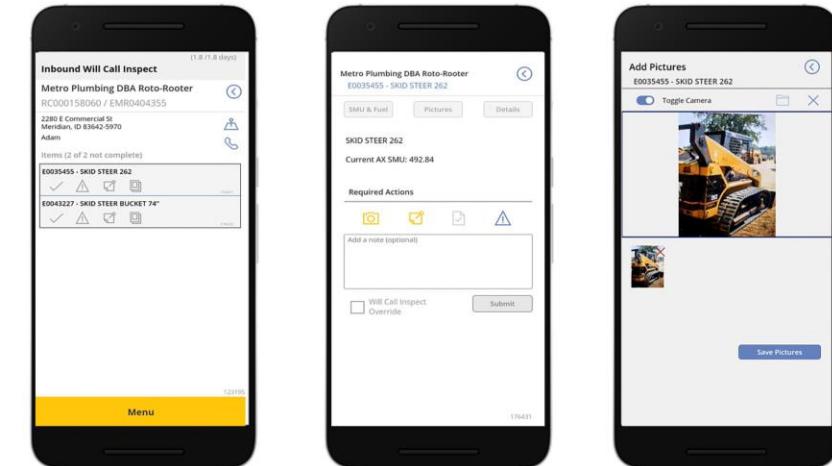
- Central platform that brought together key data that was scattered among 10 siloed business systems
- Fusion development team at Blackmores, consists of professional developers and analysts
- Each app's highly tailored interface makes it easy for different teams to use exactly the data they need
-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

Solution Architecture - Blackmores Group



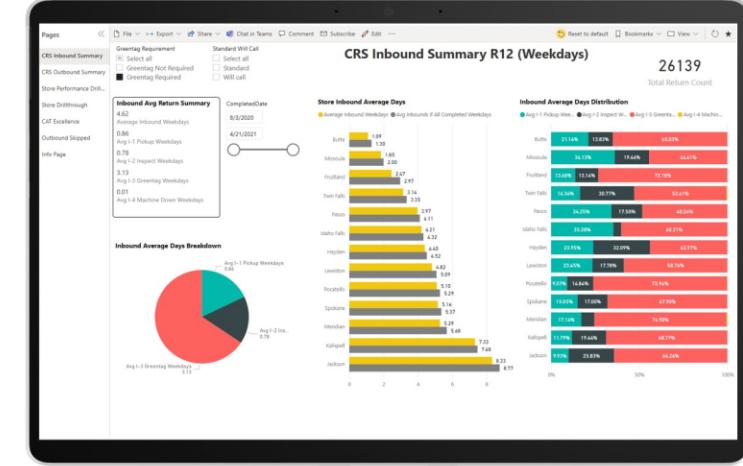
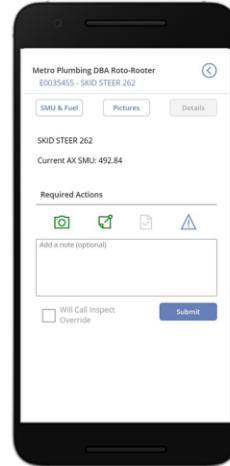
Western States Caterpillar – fusion teams

- 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

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



.ASP .NET Core

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

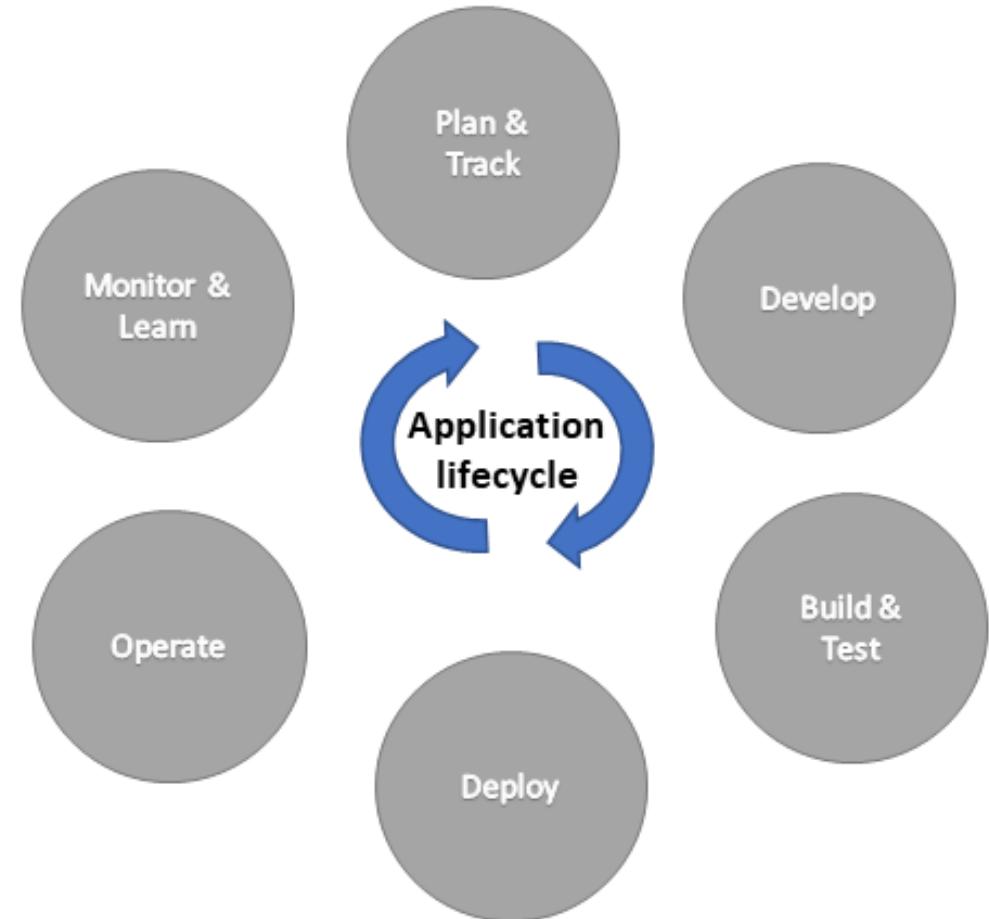
- Microsoft Power Platform is the best way for teams to build together – [blog post](#)
- Pro-dev features and learning paths – [E-book](#) | [learning path](#)
- New Power Apps developer plan – [get started](#) | [documentation](#)
- Microsoft Mechanics episode on pro-dev adoption of Power Platform – [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>

(Bonus) Power Platform Application Lifecycle Management

The background of the slide features a dark gray gradient. Overlaid on this are several thin, light gray diagonal lines of varying lengths and orientations. Interspersed among these lines are small, semi-transparent circular dots. Some dots are solid white, while others are a darker gray. One prominent white dot is located near the center-left, connected by a thin white line to another white dot further up and to the right. There are also several dark gray dots scattered across the upper right quadrant.

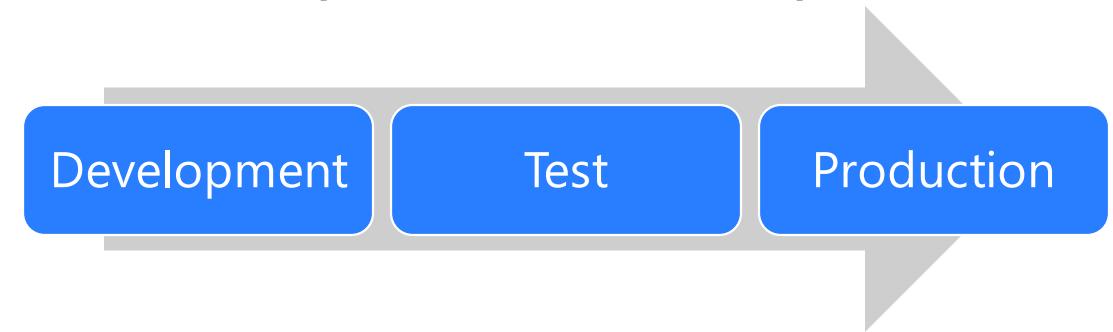
Power Platform Application Lifecycle Management(ALM)

- 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

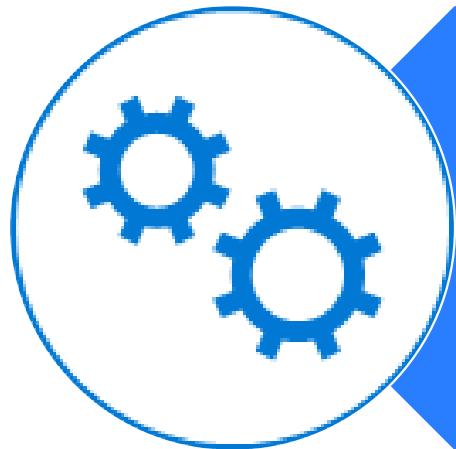


Working with solution files

- 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

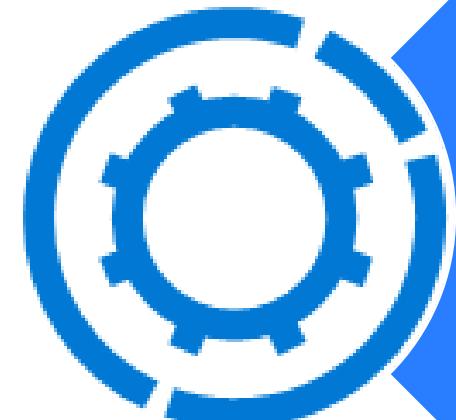


Types of solutions



Unmanaged

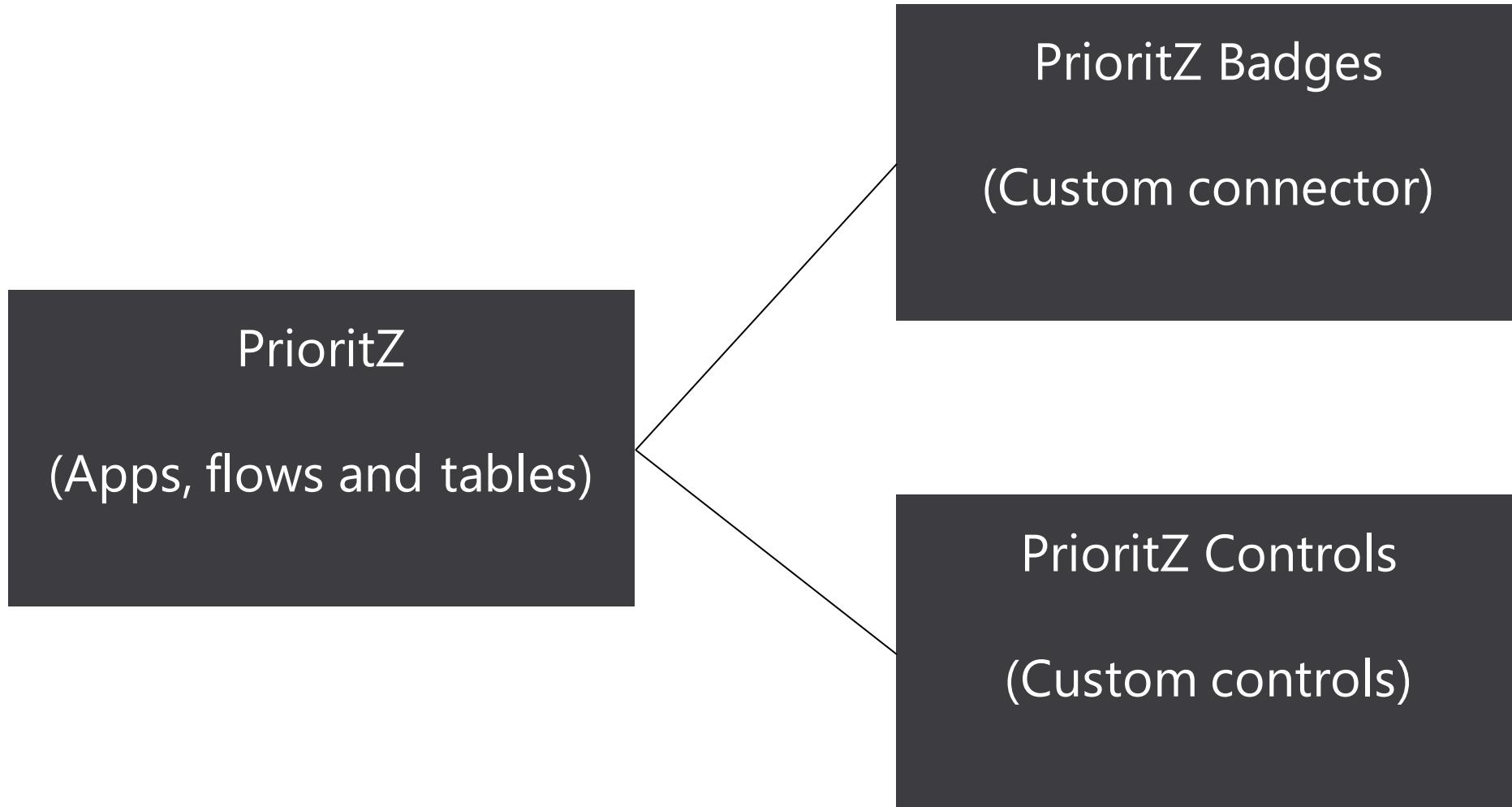
- Used during development
- Used to transport to other development environments
- Should be considered your source



Managed

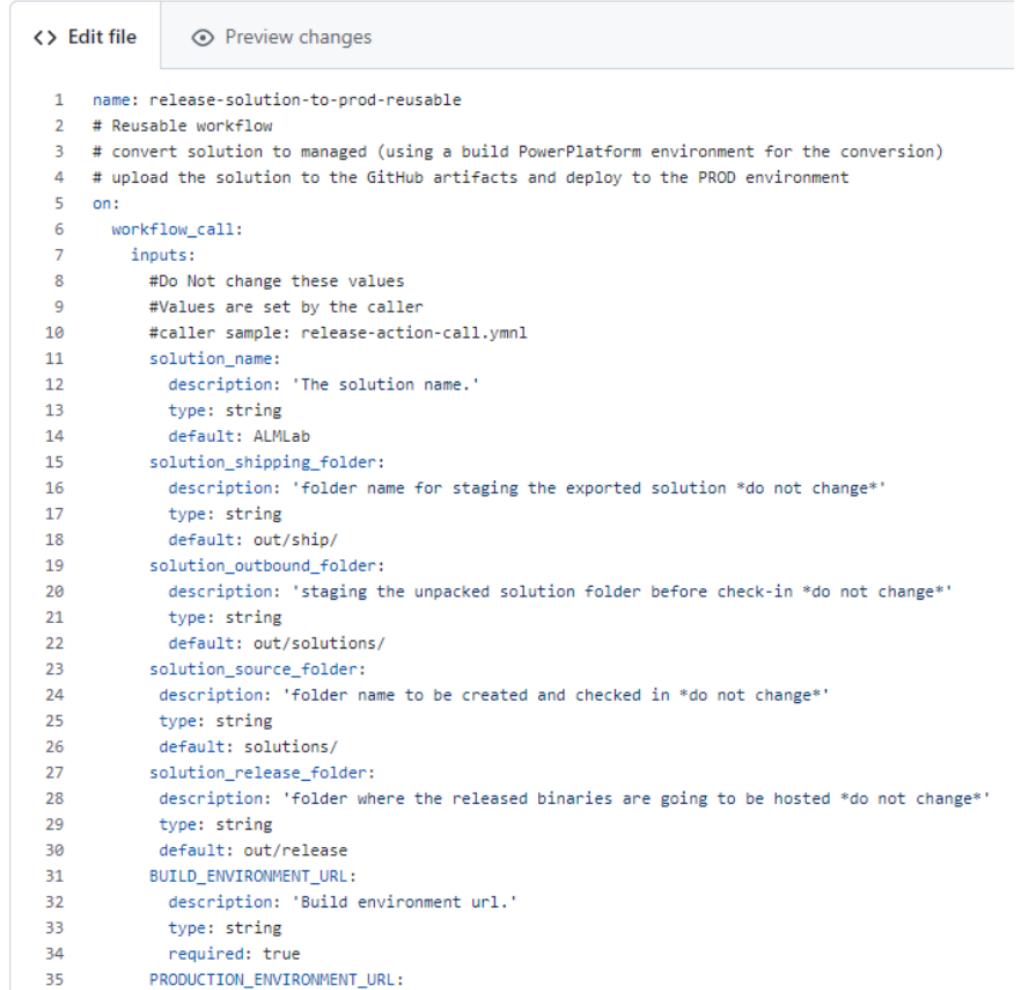
- Used to distribute to non-development environments
- Should be generated by a build server and considered as a build artifact

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



The screenshot shows a GitHub Actions workflow configuration in YAML. The interface includes tabs for 'Edit file' and 'Preview changes'. The code defines a reusable workflow named 'release-solution-to-prod-reusable'.

```
1 name: release-solution-to-prod-reusable
2 # Reusable workflow
3 # convert solution to managed (using a build PowerPlatform environment for the conversion)
4 # upload the solution to the GitHub artifacts and deploy to the PROD environment
5 on:
6   workflow_call:
7     inputs:
8       #Do Not change these values
9       #Values are set by the caller
10      #caller sample: release-action-call.ymln1
11      solution_name:
12        description: 'The solution name.'
13        type: string
14        default: ALMLab
15      solution_shipping_folder:
16        description: 'folder name for staging the exported solution *do not change*'
17        type: string
18        default: out/ship/
19      solution_outbound_folder:
20        description: 'staging the unpacked solution folder before check-in *do not change*'
21        type: string
22        default: out/solutions/
23      solution_source_folder:
24        description: 'folder name to be created and checked in *do not change*'
25        type: string
26        default: solutions/
27      solution_release_folder:
28        description: 'folder where the released binaries are going to be hosted *do not change*'
29        type: string
30        default: out/release
31      BUILD_ENVIRONMENT_URL:
32        description: 'Build environment url.'
33        type: string
34        required: true
35      PRODUCTION_ENVIRONMENT_URL:
```

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>