Why Pro Developers are Essential to Low-Code Success with Power Apps

A Guide for Application Development Leaders

Microsoft Power Apps enables citizen developers to create low-code applications on their own. But gaining the participation of pro developers lets organizations deliver more value faster. This guide explains how and why application development leaders and the pro developers they manage are essential to accelerating value delivery with Power Apps.

Contents

Maximizing the value of low-code development

Most organizations need more custom apps than their pro developers can create.

Leaders of application development groups—people like you—are responsible for creating new custom solutions for their organization. To do this, organizations traditionally rely on professional software developers. These pro developers build applications using .NET, Java, and other development technologies.

But the demand for new applications has clearly outstripped the supply of pro developers. According to IDC, more new apps will be built in the next five years than all apps built in the last 40 years[1]. If you're like most organizations, you need to build more new apps than pro developers can create. There just aren't enough people available with the right skills.

Low-code development can help you meet this demand.

Adopting Microsoft Power Apps, a low-code development platform, can help your organization address this challenge. Power Apps lets *citizen developers*, people who understand your business but aren't trained as pro developers, create applications on their own. This approach is popular today—more than 500,000 organizations use Microsoft low-code tools right now, including 97% of the Fortune 500—and Gartner predicts that 70% of enterprise application development will use low-code technology by 2025[2].

Getting the most from low-code development implies pro developers and low-code developers working together.

Yet getting the most from low-code development implies more than just training citizen developers. Your pro developers can work with these citizen developers to accelerate value creation. The real power in using a low-code application platform comes from collaboration between these two groups.

Combining low-code and pro-code: Fusion development

Adopting low-code development augments the pro-code applications built by your professional developers; it doesn't replace them. In fact, you can think about your applications in three categories based on who creates them. Figure 1 illustrates the options.

[1] IDC FutureScape: Worldwide IT Industry 2020 Predictions, October 2019

[2] Gartner, Forecast Analysis: Low-Code Development Technologies, January 2021

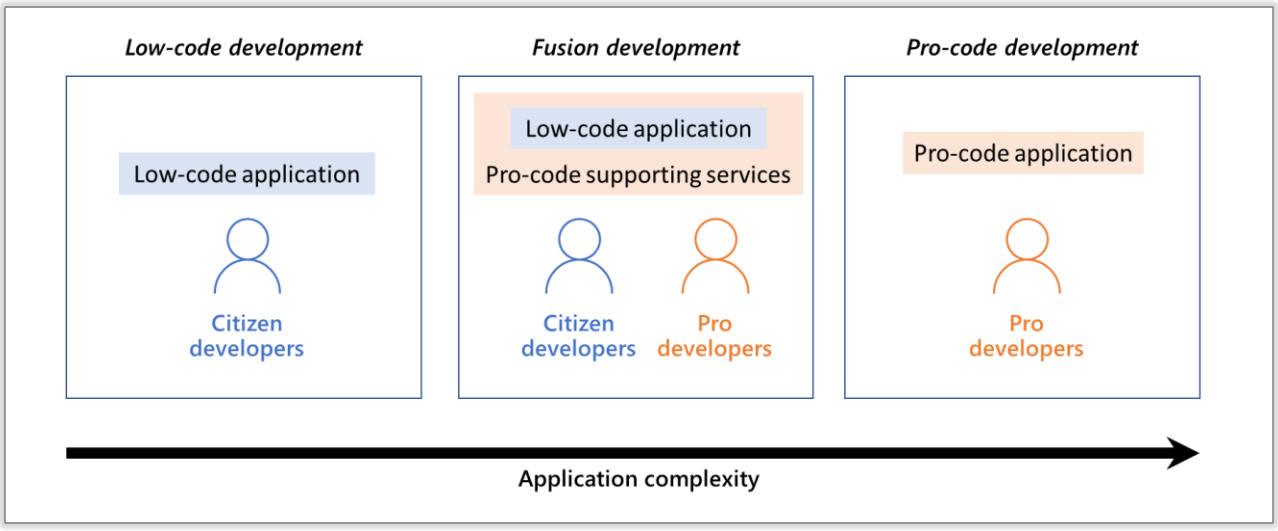


Figure 1: Using both low-code and pro-code development in your organization results in three categories of development style.

In low-code development, solutions are created solely by citizen developers.

Low-code development, shown on the left side of the figure, is done entirely by citizen developers. These developers use Power Apps to create solutions with a point-and-click approach. They can easily connect to hundreds of data sources and use a large set of templates to build apps quickly. The characteristics of an app that's a good fit for low-code development include:

- The app's users are internal to your organization rather than external customers.
- The app will handle thousands of simultaneous users, not millions.
- The app is monolithic; it doesn't use a microservices architecture.
- The acceptable latency for the app is measured in seconds rather than milliseconds.
- The app needs to be created in weeks rather than months.

In pro-code development, solutions are created solely by pro developers.

Pro-code development is at the other end of the complexity scale, shown on the far right in Figure 1. These apps are created solely by professional developers using pro-code technologies.

Solutions in this category commonly have characteristics such as these:

- The app is mission critical. It might implement an essential business process, for example, or interact directly with a large number of your customers.
- The app is capable of supporting a large number of simultaneous users, often tens of thousands or more.
- The app uses a modern architecture and platform, such as microservices built on Kubernetes.
- The app must offer low latency even under heavy load.

In fusion development, solutions are created by citizen developers and pro developers working together.

Fusion development lies in between these two options. These apps are created using low-code technology, but they also depend on pro-code supporting services or extensions. Accordingly, they're built by a *fusion team* that includes both citizen developers and pro developers. This approach is the best option for many scenarios, and it's the focus of this paper. Important examples of fusion development include these:

- Pro developers can create reusable user interface components for low-code applications. While Power Apps provides many built-in UI controls, your pro developers can also build their own components either to meet corporate standards or to support more complex scenarios. This lets you improve the user experience for your low-code apps beyond what's available out of the box.
- Pro developers can create custom connectors to external applications and logic. Among other things, this can let citizen developers easily modernize legacy applications with a new user interface, a mobile client, or in other ways.
- Pro developers can help citizen developers use a modern development process. While Power Apps lets citizen developers create real applications, the fundamentals of application lifecycle management are still important. Fusion teams let citizen developers use better development practices under the guidance of pro developers.

Enabling these fusion scenarios is a critical part of creating the best possible apps in your organization as rapidly as possible. This paper looks at each of them in more detail below.

The role of pro developers in fusion development

Pro developers are critical for fusion development.

Using low-code development in your organization doesn't imply turning your pro developers into citizen developers—far from it. The truth is that pro-code skills become even more valuable in a world that includes a low-code platform. You can now choose the best approach for your organization in each application development scenario.

For many internal applications (maybe even most of them), low-code development is likely to be the best choice. You'll get faster results with less complexity by relying solely on citizen developers. For many externally facing applications (and even some internal apps), purely pro-code development will be best. These solutions often need the scale, responsiveness, and complexity that only a pro-code solution can provide.

Pro developers working with citizen developers is often the best way to create new apps.

But for a large number of scenarios, fusion development will be your best option. Creating a fusion team with people in both roles lets you complement the power of pro development with the accelerated delivery time of low-code development. This helps you meet the growing demand for a wide variety of apps within your organization. And by using your pro developers more intelligently, your organization can make better use of their scarce—and expensive—time. Why not let citizen developers do as much as possible, freeing up your pro developers to add the value only they can provide?

Pro developers are the heroes of fusion development

As low-code development becomes more and more popular, some of your pro developers might resist this approach. Who do these citizen developers think they are, anyway?

Yet the adoption of low-code platforms is one of the major trends of our time; don't expect it to slow down. And making low-code development really successful in your organization requires pro developers. Fusion development scenarios are among the most valuable, and they're not possible without these skilled people.

The truth is clear: pro developers are the heroes of fusion development. As an application development leader, embracing this approach lets you make the work your people do even more important to your organization.

Power Apps basics

Power Apps is part of Power Platform.

Power Apps is one aspect of Microsoft's complete low-code offering, Power Platform. All the technologies in this interconnected set are designed primarily for business users rather than pro developers. The technologies in Power Platform are:

- ***Power Apps*** for building low-code applications.
- ***Power Automate*** for automating business processes.
- ***Power BI*** for creating data-driven insights.
- ***Power Virtual Agents*** for creating chatbots.

Low-code solutions built with Power Apps, the focus of this paper, are structured much like traditional applications. Figure 2 shows their main components.

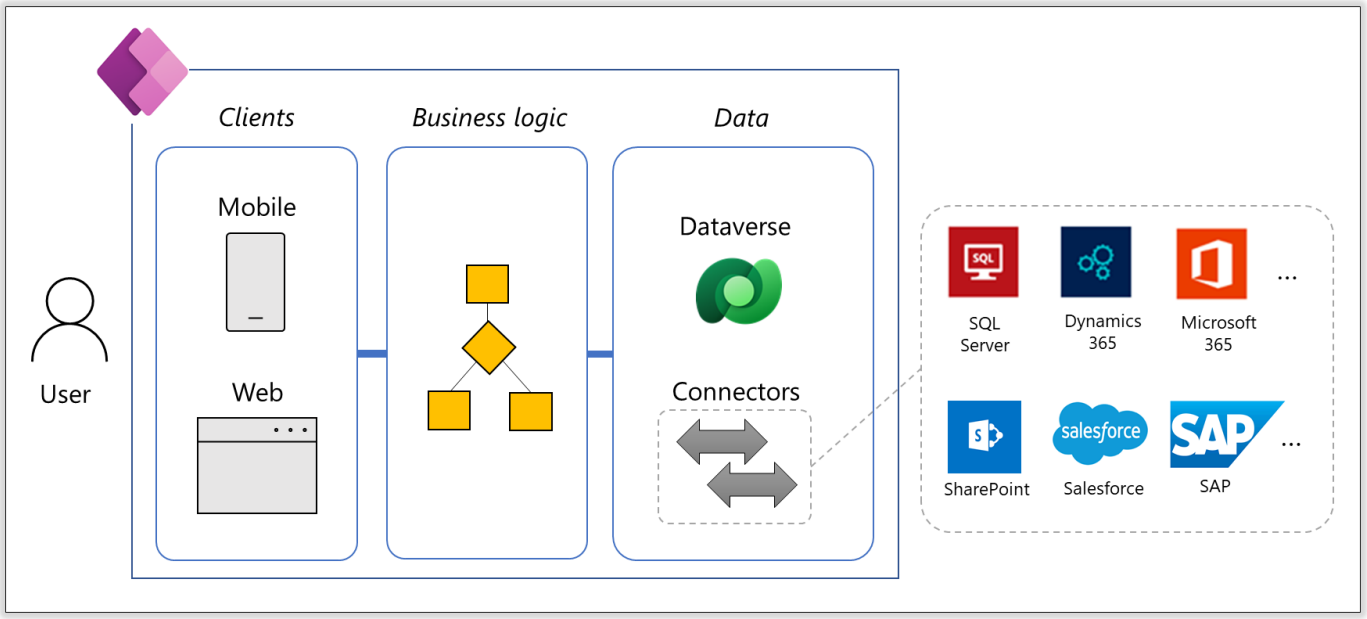


Figure 2: Power Apps supports creating business logic for mobile and web clients, along with allowing access to other data and services.

Low-code applications have the same basic structure as traditional applications.

As the figure shows, Power Apps apps can be thought of in three distinct parts—clients, business logic, and data—just like traditional applications:

- **Clients:** Apps built using Power Apps can run on the web and on iOS and Android devices. A citizen developer can create an app in

hours using built-in controls that can be customized as needed by that developer.

- **Business logic:** Unlike a traditional application, where logic is created by writing code in C#, Java, or some other programming language, a Power Apps solution can be built without writing code. Instead, an app's logic can be defined by writing Excel-like formulas. And because Power Apps runs on Microsoft Azure, apps benefit from the security, reliability, and certifications this cloud platform provides.

Power Apps provides connectors to many different services and data sources.

- **Data and services:** Traditional applications commonly rely on some kind of database management system (DBMS). Low-code apps built with Power Apps can instead use a built-in data management system called *Dataverse* (originally known as the *Common Data Service*). Along with storing data, this service also lets people define business logic, such as rules for data validation, on that data, as well as an enterprise-grade security model. Power Apps applications can also use more than 450 built-in *connectors* that make it easy to access other data sources such as SQL Server, Azure SQL Database, Oracle, SharePoint, Dynamics 365, Salesforce, Dropbox, and many others. Some connectors also allow access to functionality provided by cloud services, such as the ability to send tweets on Twitter. Power Platform also provides an on-premises data gateway to let your cloud-based low-code apps access on-premises data.

Power Apps provides a simpler development environment than Visual Studio.

To create solutions graphically, citizen developers use a web-based experience called *Power Apps Studio*. Rather than using more complex professional developer tools, this low-code development environment offers a simpler way to design applications. Figure 3 shows an example of Power Apps Studio.

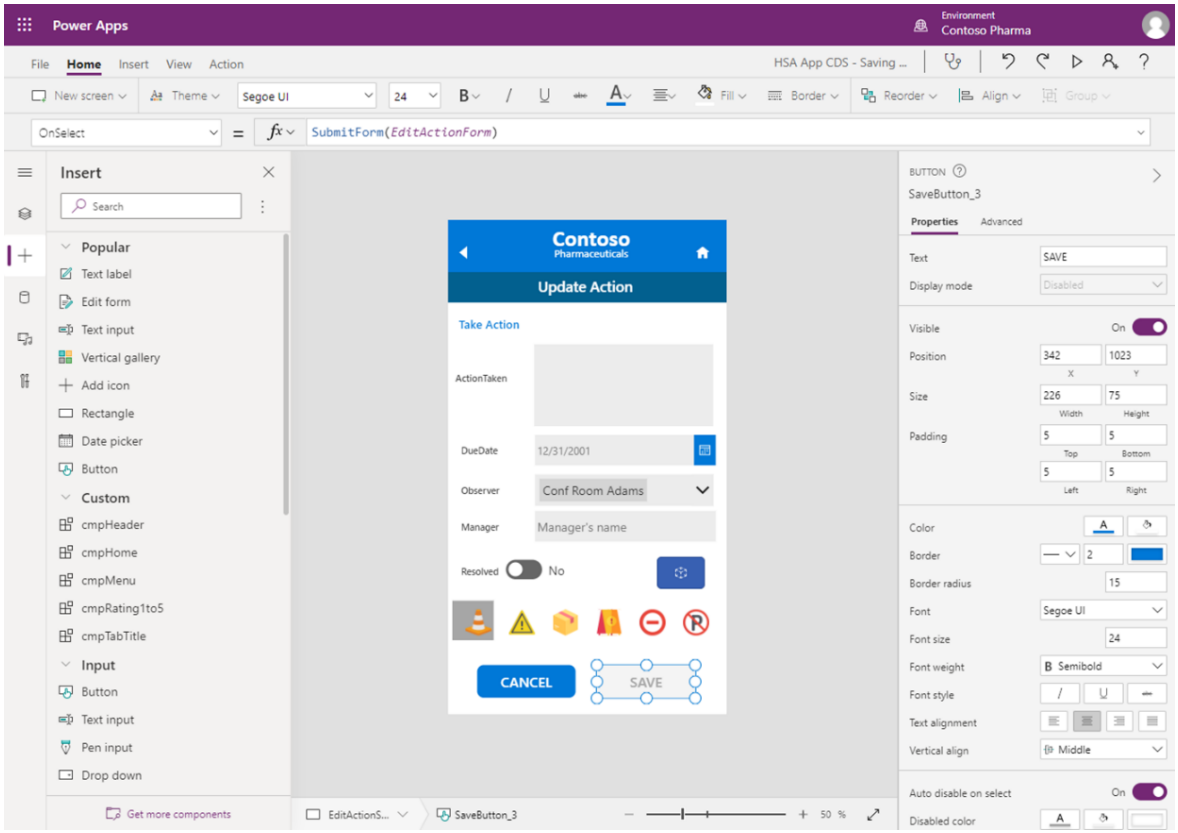


Figure 3: Power Apps Studio lets citizen developers create low-code apps graphically.

Power Apps makes it easy to create forms-based apps.

Power Apps is focused on creating forms-based apps that let their users easily create, read, update, and delete information. As Figure 3 shows, Power Apps Studio provides familiar user interface controls such as buttons, labels, and more. To create the user experience, citizen developers drag and drop these controls onto a canvas. They then set properties of those controls, including button text, size, and color as the example shows.

Apps can also take actions. In the example shown in Figure 3, clicking on the Save button executes SubmitForm, shown in the bar at the top of the screen. Other possible actions include navigating to another screen, executing conditional logic with If statements, and assigning values to variables. All of this is expressed in a syntax that's similar to Excel formulas, making it natural for business users. And while it's not shown here, using a connector to access a data source—Azure SQL Database, SharePoint, Salesforce, or something else—requires just dragging and dropping that connector into your app.

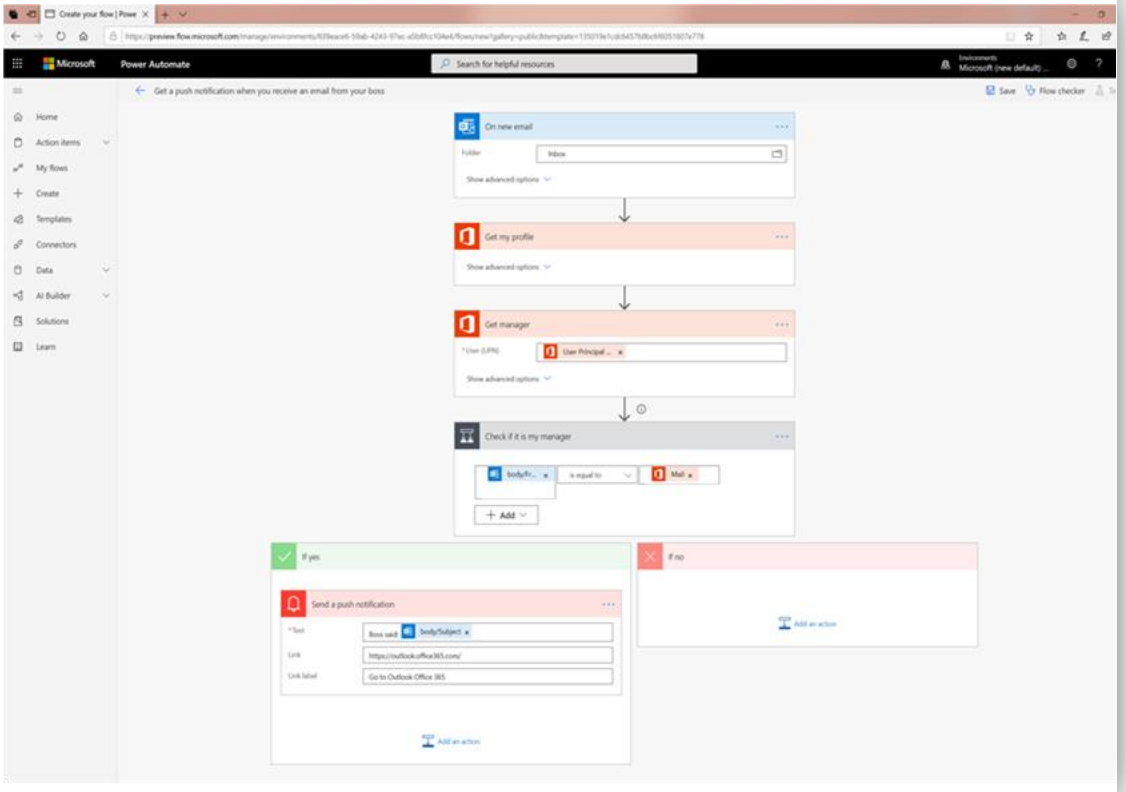
Creating pro-code applications requires a substantial amount of specialized knowledge. Creating low-code applications requires

understanding the business problems to be solved, but there's much less need for detailed technical knowledge. This is why they can be created by less technical business people, i.e., by citizen developers.

Creating workflows with Power Automate

Power Apps is designed for creating interactive business applications. But suppose you want to create a workflow that carries out a sequence of steps whenever some event occurs—how could you do this? The answer is Power Automate, a tightly integrated companion to Power Apps.

Power Automate is another aspect of Power Platform—it comes with Power Apps at no extra cost—and it also provides a low-code way to create logic. For example, if you want to be notified whenever you receive mail from your manager, you might create a Power Automate *flow* like the one shown below.



Power Automate provides a quick and easy way to automate business processes, and it includes many pre-defined flows for common scenarios such as document approval. Flows can also use the same connectors as Power Apps, letting you connect to the outside world as needed. Power Automate and Power Apps are complementary offerings, and low-code solutions often use both.

Fusion development scenarios

Fusion development is useful in many situations.

Power Apps lets citizen developers do a lot on their own. But to get the full value from this low-code technology, your organization should consider embracing fusion development. As described earlier, the scenarios in which pro developers can add the most value include these:

- Improving the user experience by creating custom controls for low-code applications.
- Creating connectors to your existing applications and to external logic. Among other things, this enables citizen developers to modernize older software.
- Helping citizen developers use a modern development process.

What follows look at each of these.

Improve the user experience with custom controls

One of the worries of using a low-code development environment is the concern that you might not be able to create exactly what you need. Since you're not working in a pro developer environment, what if your application requires something that isn't possible with low-code tools? What if you hit a cliff, for example, where the built-in Power Apps controls don't meet your app's evolving requirements? Because combining the low-code and pro-code worlds is the essence of fusion development, Power Apps addresses this concern in several different ways.

For example, this low-code platform provides a broad set of user interface controls that meet most needs, including the ability to customize an app's interface with your organization's branding. But what if you want to create a custom UI control, such as a Gantt chart or Kanban board? Or what if you want to use existing controls, such as internally developed controls or those provided by the React or Angular frameworks, in a Power Apps solution?

Pro developers can add extensions to the user interface Power Apps provides.

Both of these things can be done by using the Power Apps Component Framework. Using this technology, pro developers can create the controls a low-code app needs, then let those controls be used by citizen developers just as if they were standard Power Apps controls. Figure 4 shows how this looks.

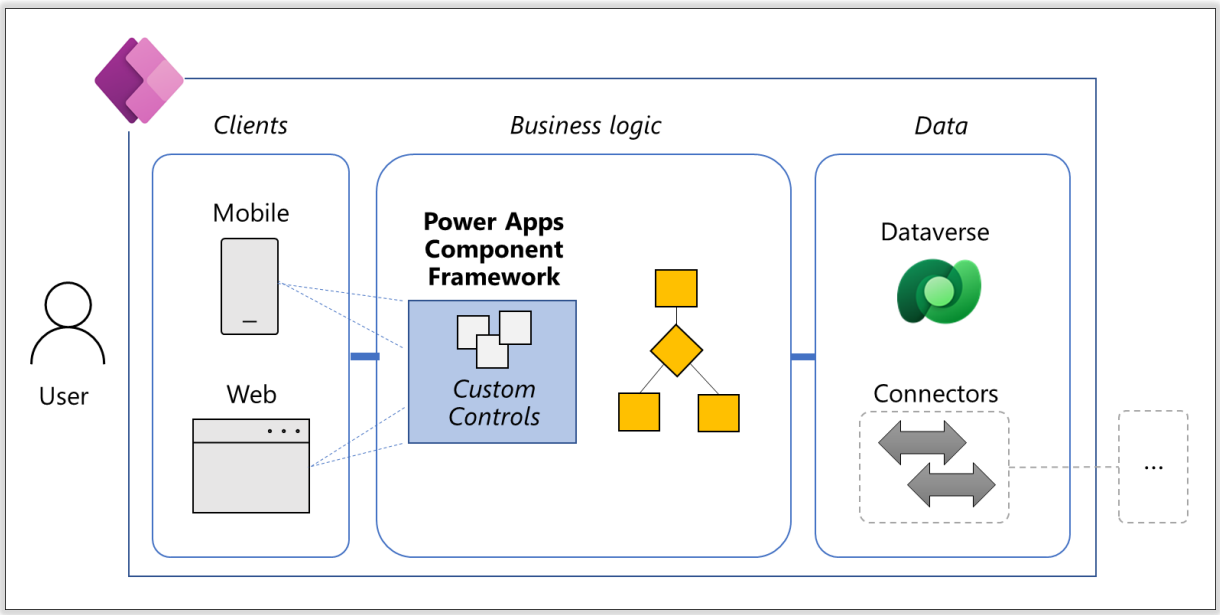


Figure 4: The Power Apps Component Framework lets pro developers create custom user interface controls.

Custom controls created with the Power Apps Component Framework are treated like built-in controls, so they can be used with both mobile clients and web browsers. To create them, pro developers use tools such as Visual Studio and Visual Studio Code. Each custom control requires doing three things:

- Writing an XML manifest that defines the component.
- Implementing the component's logic in Typescript or JavaScript.
- Specifying the component's appearance in CSS.

These three elements let the Power Apps Component Framework map between the Power Apps world and the pro-code world. And to make life easier for pro developers, the framework also comes with tools for building and testing your custom controls.

Extensibility is an essential aspect of a low-code platform.

To get a concrete sense of where custom controls are useful, suppose your organization is well into creating a Power Apps solution, then discovers that this app needs one or more UI elements that aren't available in Power Apps itself. The Power Apps Component Framework lets pro developers add these to your app. You're not forced to rebuild the entire application as a pro-code solution just to get the user interface you need. This kind of extensibility is essential

in a low-code platform, which is why the Power Apps Component Framework is an important technology.

Give low-code apps access to external applications and logic

Citizen developers can do a lot with Power Apps. But they can do even more if the Power Apps solutions they create can connect to external applications and logic.

Power Apps provides hundreds of built-in connectors.

As described earlier, Power Apps provides built-in connectors to hundreds of different services and data sources. Using connectors is incredibly simple: a citizen developer just drags a connector into Power Apps Studio, and their app can interact with whatever the connector links to.

Pro developers can also create custom connectors.

But what if a citizen developer needs a connection to something for which there is no existing connector, such as a custom-built enterprise application? Or what if your Power Apps solution needs to connect to custom pro-code logic created with, say, Azure Functions? For situations like these, pro developers can create *custom connectors*.

For example, suppose you'd like to give citizen developers access to an existing line-of business (LOB) application. It might be an on-premises ERP system or a custom-built business solution or even a twenty-year-old legacy application. Whatever the situation, pro developers can create a custom connector to this application. Citizen developers can use this custom connector just as if it were provided by Microsoft. Figure 5 shows a simple example.

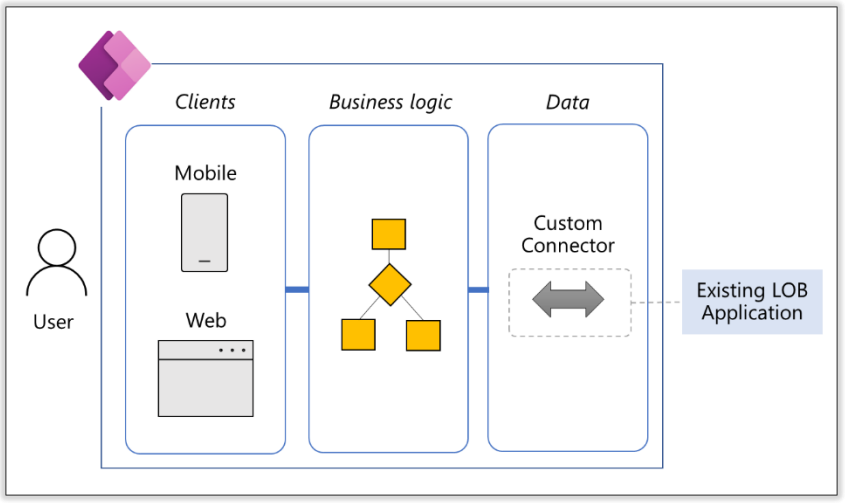


Figure 5: Pro developers can create a custom connector to external logic, such as an existing LOB application, that citizen developers can use.

To do this, you must first make sure that whatever you're trying to connect to exposes its services via a REST API. Modern software commonly does this, but older applications might not. In this case, pro developers will need to create this API. They can then build a custom connector on top of the API and expose it to Power Apps.

Custom connectors can provide access to existing applications, external Azure logic, and more.

When necessary, you're also free to build custom connectors to external logic created with Azure Functions or other services. Pro developers sometimes worry that applications created with low-code platforms won't always be able to do exactly what's required. By creating a custom connector, you're always free to access external logic running in Microsoft Azure or another pro developer environment.

There's another option for providing these custom connections that takes direct advantage of these REST APIs: using Azure API Management. This service provides an intermediary between API clients and the backend applications and services they access. By doing this, API Management helps make APIs more secure, more efficient, and easier to use and maintain.

Using API Management can also make it easier to create custom connectors for Power Apps. To see why, look at Figure 6.

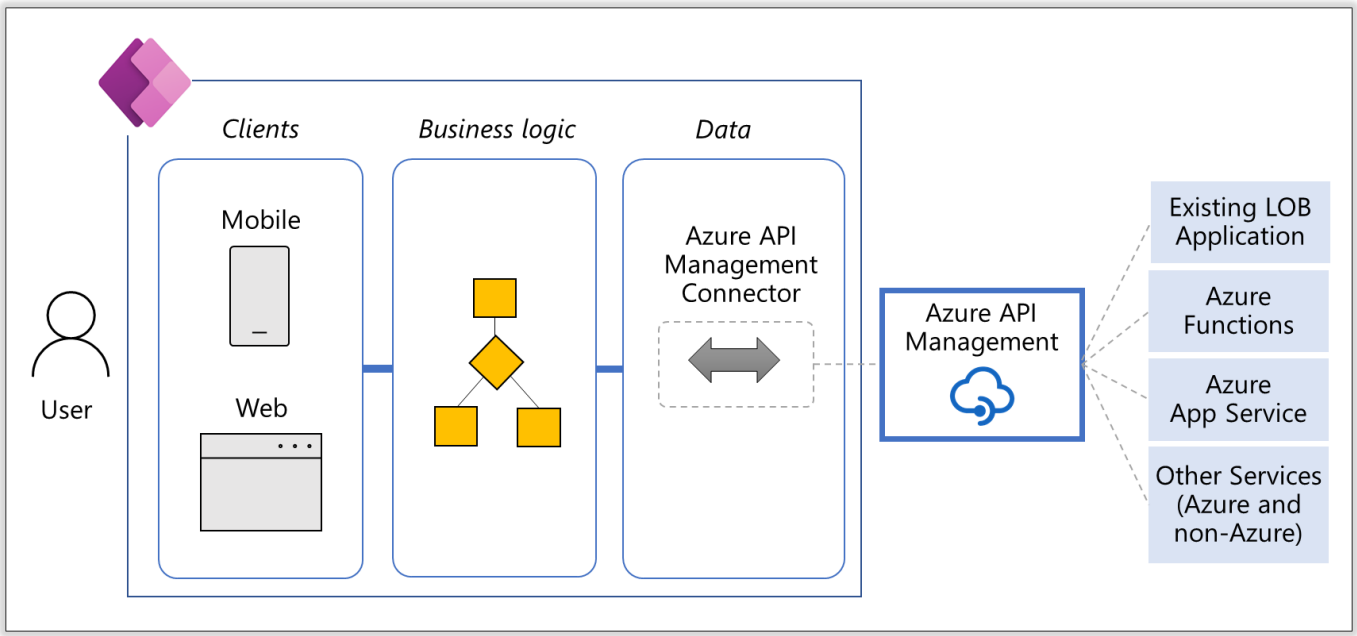


Figure 6: Using Azure API Management makes it easier for pro developers to give citizen developers access to external applications and logic.

As the figure shows, Power Apps provides a connector to API Management. A pro developer can publish any REST API into API Management, then specify that the API should be made available as a Power Platform connector. Power Apps will make the link automatically; the pro developer no longer needs to create a custom connector explicitly.

Using Azure API Management simplifies creating custom links to the outside world.

This approach can be used in any custom connector situation. The figure shows connecting to an existing LOB application, just as in Figure 5, but it also shows connecting to custom external logic built using Azure Functions, Azure App Service, or in some other way (including any internally developed APIs to your on-premises custom applications). Putting this logic behind API Management helps with flow control and other issues, and it also makes the link to Power Apps easier to build. Although it's not required—you're free to build a traditional custom connector as in Figure 5—it's the best approach in many situations.

Citizen developers can modernize the long tail of existing applications

Whether you create custom connectors on their own or use API Management, giving Power Apps access to existing applications brings a huge benefit: citizen developers can now modernize those applications by adding a mobile client, an updated user interface, and more.

For your most important applications, you've probably already had pro developers do this modernization. But many organizations have a long tail of less important applications that haven't yet been modernized. Perhaps the business value of these updates didn't justify the cost of pro developer time, or maybe your organization lacks the pro developer resources to get this work done. Whatever the reason, having pro developers create custom connecters to these legacy applications can let citizen developers do the modernization with Power Apps.

These citizen developers are less expensive and more plentiful than pro developers, and they might even better understand the business requirement for modernization. All pro developers need to do is create the connectors. Citizen developers can take it from there.

Help citizen developers use a modern development process

Using a good development process is important for low-code apps.

Power Apps lets citizen developers create useful applications without acquiring the technical expertise of pro developers. But there are some things pro developers know that are still important in low-code scenarios. One of these is the value of a good development process. Helping citizen developers understand and use a modern approach here is valuable.

In a simple Power Apps scenario, a single citizen developer creates an app on their own. There's no need to share the app among multiple developers. In a more complex scenario, however, two or more developers might work together to create the same low-code app. Power Apps supports both options, as Figure 7 shows.

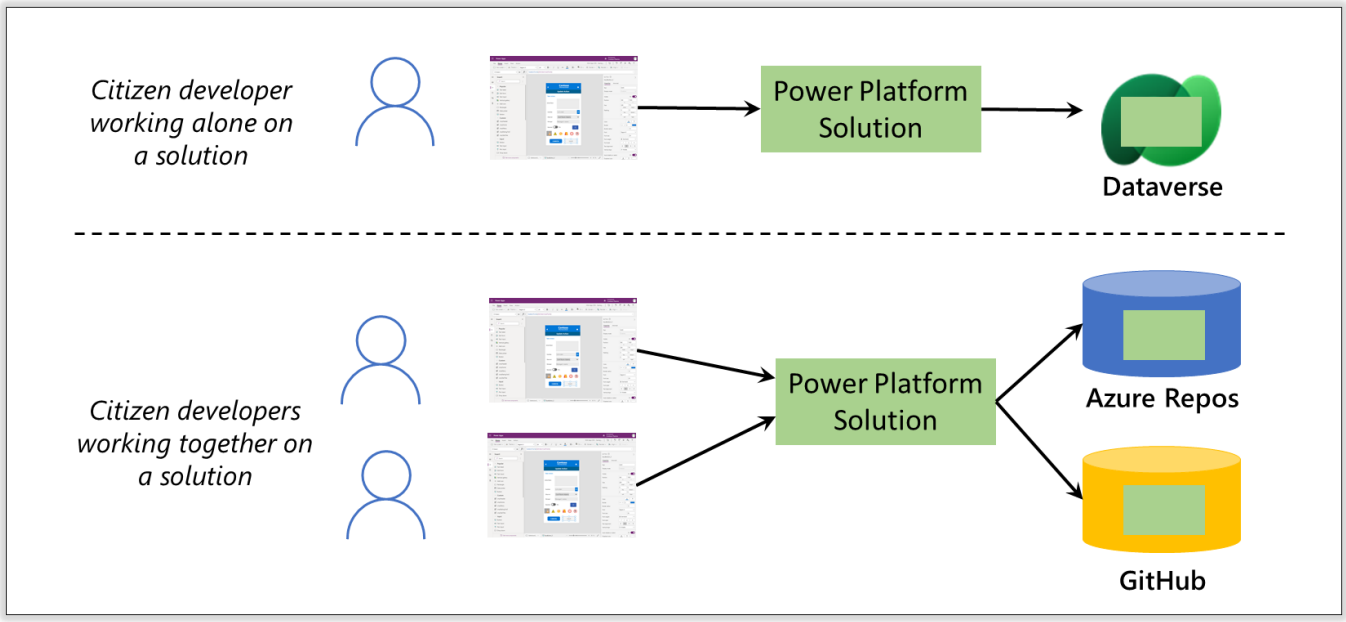


Figure 7: When citizen developers work together on a solution, they can use either Azure Repos or GitHub for source code control.

As the figure suggests, an application created with Power Apps is stored as a *Power Platform solution*. This solution can contain just that app, or it might also include other components, such as a Power Automate flow. In any case, the solution is stored as XML files. By default, those XML files are placed in Dataverse as Figure 7 shows. This approach works well when a solution is created by a single developer.

Power Apps allows using modern source code control services.

When multiple developers work together on a solution, however, safely sharing the underlying XML files typically means storing them in a repository managed by a source code control system. Power Apps provides two options for doing this: Azure Repos, part of the broader Azure DevOps service, and GitHub. Whatever choice the Power Apps developers make, they can now check out artifacts (e.g., the XML files) from the repo, modify them, then check them back in.

Once a solution is ready to run, it must be deployed into an *environment*. Each environment provides a logically isolated world inside your cloud tenant that's able to run one or more Power Platform solutions. You might create different environments for different groups in your organization, for example, giving each group control over the Power Platform solutions it runs.

Citizen developers can use separate development and production environments.

It's also common to create separate environments for developing Power Platform solutions and running those solutions in production. This lets you modify current applications in the development environment, then deploy new versions into the production environment once they're ready.

For a Power Platform solution stored in Dataverse, its developer indicates the environment in which it should run when it's deployed. The solution is loaded directly into that environment by Power Platform itself. For Power Platform solutions stored in Azure Repos or GitHub, however—solutions that are shared across multiple developers—more is required. Figure 8 illustrates what happens.

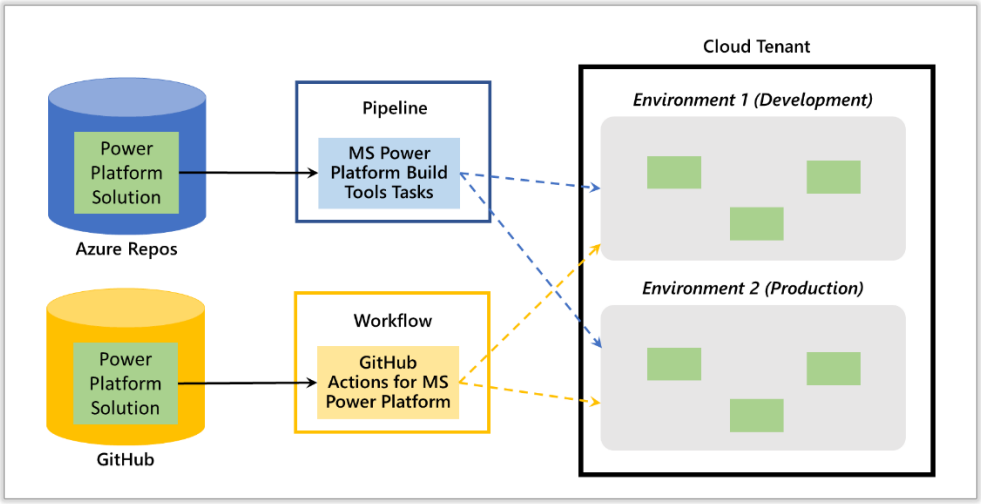


Figure 8: Power Platform solutions can be automatically deployed to environments from Azure Repos or GitHub.

As the figure shows, you can create a pipeline (using Azure Pipelines) to deploy a solution stored in Azure Repos into your development or production environments as needed. For solutions stored in GitHub, you can create a workflow that does the same thing.

In both situations, Microsoft provides the low-level tasks you need to create these automated deployments. For pipelines, these are the *Microsoft Power Platform Build Tools Tasks*, while for GitHub workflows, they're the *GitHub Actions for Microsoft Power Platform*. In both cases, the tasks/actions do similar things, such as packaging the distinct XML files kept in source code control into the format required for deploying a solution into an environment.

Power Apps works well in a CI/CD environment.

It's also worth pointing out some things these tasks/actions don't do. Because Power Platform solutions aren't compiled, there's no build step. There's also no notion of unit testing, so your pipelines and workflows need not worry about this. What you do get, however, is a way to automate deployment, helping Power Platform solutions fit into a modern continuous integration/continuous delivery (CI/CD) world.

When should pro developers use Power Apps themselves?

Fusion development, with pro developers and citizen developers working together, can bring real benefits to your organization. But the rise of low-code platforms suggests another possibility: might there be situations where it makes sense for pro developers to use Power Apps themselves?

For most organizations, the answer is yes. Among the most popular situations in which pro developers choose Power Apps over pro-code technologies are these:

Power Apps can help even pro developers get more done in less time.

- ***Building enterprise applications more quickly.*** Any development organization that builds internal enterprise solutions is often creating forms-over-data applications that create, read, update, and delete information. This is exactly what Power Apps was designed to do. With its easy-to-create user interfaces, pre-built connectors to data and services, and other productivity features, this low-code platform can help even pro developers achieve their targets more quickly. Some of your pro developers might prefer to focus solely on solutions that require

deeper technical knowledge, which is fine—those are important skills. But others are likely to embrace the Power Apps approach once they realize that it lets them do more in less time.

AI Builder makes creating AI models fast and simple.

- ***Easily creating and using machine learning models***. Power Platform includes AI Builder, a low-code artificial intelligence technology. This easy-to-learn offering lets non-specialists create AI models for forms processing, object detection, and more. And compared to pro-code machine learning options, these models can be significantly faster to create. If your organization needs AI solutions, using AI Builder with Power Apps might well be your best option.

Power Apps is a good choice for building mixed-reality apps.

- ***Creating mixed-reality applications***: Smartphones are ubiquitous in organizations today. This means that mixed-reality applications, such as the example shown in Figure 9, can be used everywhere. But creating these applications has traditionally required specialized pro developer skills. Power Apps changes this by providing support for mixed reality in a low-code platform. If your organization needs to create this kind of app, using Power Apps is likely to be faster and simpler than working solely with pro-code technologies.

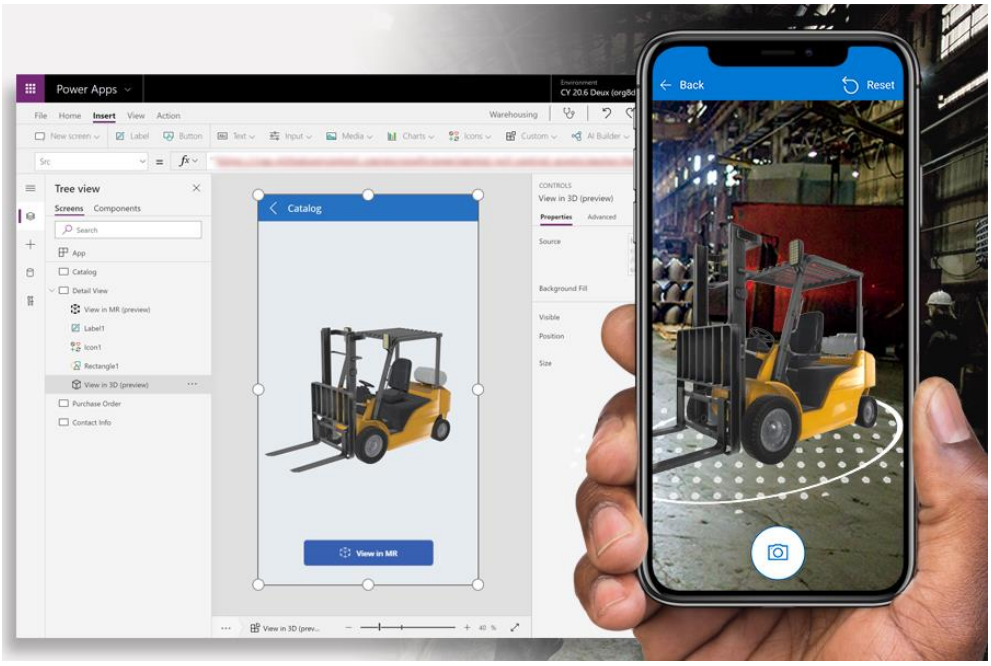


Figure 9: Power Apps provides built-in support for creating mixed-reality apps.

There are other scenarios as well where pro developers might choose to use Power Apps rather than C# or another pro-code technology. If a team is building an application that requires the business-oriented data types supported natively by Dataverse, for example, they might choose to build on this Power Platform technology. Similarly, a development team looking for a quick way to prototype an idea might build a Power Apps solution to get a concrete feeling for what's needed.

Microsoft uses Power Platform to build many of its internal applications.

The core idea is this: even for pro developers, there are quite a few cases in which using Power Apps is a good option. Many organizations are doing this today, including Microsoft: our own internal IT group now builds a large share of its applications using Power Platform technologies.

What to do now

Digital transformation requires creating more custom applications. Most organizations don't have the pro developer resources they need to do this, which is why citizen developers are essential to digital transformation success. Yet citizen developers can't do this on their own; they need pro developer support. This is what makes fusion development so important.

Embracing fusion development can make you and your team heroes.

As a development leader, you can help your pro developers find their way in this new world. You can be the one who brings fusion development into your enterprise rather than fighting the inevitable spread of low-code technology. Doing this will benefit your entire organization. Just as important, enabling citizen developers can also give your group more time to build great pro-code apps. Embracing fusion development, combining low-code and pro-code, is how you and your pro developers can become heroes in your organization.

For more information

To learn more about Power Apps and Power Platform in general, go to https://powerapps.microsoft.com/en-us/blog/microsoft-powerapps-learning-resources/.

To see how other organizations, including LEGO, Autoglass, Heathrow Airport, the American Red Cross, Standard Bank, and SNCF Railway have succeeded with Power Apps, go to https://powerapps.microsoft.com/en-us/blog/powerapps-champions/.

For more on driving real business results with Power Apps and Power Platform, watch this webinar: https://aka.ms/slbwebinar

For more on how pro developers help make low-code development successful, watch this webinar: https://aka.ms/RADWebinar

For more on using GitHub Actions to implement a modern development process, watch this webinar: https://aka.ms/PPGitWebinar