# Identify the technology options

Modern businesses run on multiple applications and services. How well your business runs can often be impacted by how efficiently you can distribute the right data to the right task. Automating this flow of data can streamline your business even further. Choosing the right technology for these critical data integrations and process automation is also an important consideration.

The first question to ask is whether you prefer to design the workflow in a GUI designer tool or by writing code.

Some valid reasons for using a design-first tool:

* People who design the workflow have no coding experience.
* Later designers and users can consult the graphical design to clearly understand how the workflow proceeds.

Alternatively, you can choose to use a code-first tool because:

* People who design the workflow are developers and prefer to work entirely in code.
* You want the details of a workflow to be hidden from non-coders.

Diagram

Description automatically generated

## Analyze the decision criteria

### Design First

* + In **Logic Apps**, there is a GUI designer on which you draw out the workflow. It is intuitive and easy to use but you also can delve under the hood and edit the source code for a workflow. This tool is designed for people with development skills.
  + In **Microsoft Power Automate**, extra help and templates are provided for common types of workflows. There is no way to edit the source code that the tool creates. This tool is designed for users who have a good understanding of the business process but no coding skills.

### Code First

* + Because of the extra features that are included with **Azure Functions**, including wider ranges of trigger events and supported languages, the ability to develop test code in the browser, and the pay-per-use price model, consider Azure Functions to be your default choice.
  + **Web jobs** only supports C# on Microsoft Windows.

There are two situations in which web Jobs might be a better choice:

* + - You have an existing Azure App Service application, and you want to model the workflow within the application. This requirement means that the workflow can also be managed as part of the application, for example in an Azure DevOps environment.
    - You have specific customizations that you want to make to the JobHost that are not supported by Azure Functions. For example, in a Web Job, you can create a custom retry policy for calls to external systems. This kind of policy can't be configured in an Azure Function.

### Azure Functions or Azure Apps Service Web Jobs?

* Cost: With Web Jobs, you pay for the entire VM or App Service Plan that hosts the job. Azure Function can run on a consumption plan, so you only pay when the function runs. Since this process only kicks off when a bike is returned, we might stand to save by selecting Azure Functions.
* Integrations: You want to integrate the maintenance workflow with the Logic App that you build for the bike booking and rental process in the previous unit. Although it is possible to call a Web Job from a Logic App, the integration between Logic Apps and Functions is closer. For example, you can more easily control your call to a Function from the Logic Apps designer.

## Mixing technologies

Remember that there is no need for you to use the same technology for different workflows:

If your requirements differ, you are likely to reach a different answer at the end of your decision-making process.

Furthermore, you can also call one workflow from another. For example, a workflow implemented in Microsoft Power Automate can easily call another that is built as an Azure Function.

One reason to mix the technologies used in your business processes would be to give users control over a small section of a complete workflow. Do this by implementing that section in Microsoft Power Automate, then call that flow from a Logic App, Web Job, or Function.

## Compare Azure Functions and Azure Logic Apps

|  | **Azure Functions** | **Logic Apps** |
| --- | --- | --- |
| **Development** | Code-first (imperative) | Designer-first (declarative) |
| **Connectivity** | About a dozen built-in binding types, write code for custom bindings | Large collection of connectors, Enterprise Integration Pack for B2B scenarios, build custom connectors |
| **Actions** | Each activity is an Azure function; write code for activity functions | Large collection of ready-made actions |
| **Monitoring** | Azure Application Insights | Azure portal, Azure Monitor logs |
| **Management** | REST API, Visual Studio | Azure portal, REST API, PowerShell, Visual Studio |
| **Execution context** | Can run locally or in the cloud | Supports run-anywhere scenarios |

## Compare Azure Functions and Azure Logic Apps

|  | **Azure Functions** | **WebJobs with WebJobs SDK** |
| --- | --- | --- |
| Serverless app model with automatic scaling | Yes | No |
| Develop and test in browser | Yes | No |
| Pay-per-use pricing | Yes | No |
| Integration with Logic Apps | Yes | No |
| Trigger events | Timer Azure Storage queues and blobs Azure Service Bus queues and topics Azure Cosmos DB Azure Event Hubs HTTP/WebHook (GitHub Slack) Azure Event Grid | Timer Azure Storage queues and blobs Azure Service Bus queues and topics Azure Cosmos DB Azure Event Hubs File system |