**Azure Functions**

When you're concerned only about the code running your service, and not the underlying platform or infrastructure, using Azure Functions is ideal. Functions are commonly used when you need to perform work in response to an event (often via a REST request), timer, or message from another Azure service, and when that work can be completed quickly, within seconds or less.

Functions scale automatically based on demand, so they're a solid choice when demand is variable. For example, you might receive messages from an IoT solution that's used to monitor a fleet of delivery vehicles. You'll likely have more data arriving during business hours.

Using a virtual machine-based approach, you'd incur costs even when the virtual machine is idle. With functions, Azure runs your code when it's triggered and automatically deallocates resources when the function is finished. In this model, you're only charged for the CPU time used while your function runs.

Functions can be either stateless or stateful. When they're stateless (the default), they behave as if they're restarted every time they respond to an event. When they're stateful (called Durable Functions), a context is passed through the function to track prior activity.

Functions are a key component of serverless computing. They're also a general compute platform for running any type of code. If the needs of the developer's app change, you can deploy the project in an environment that isn't serverless. This flexibility allows you to manage scaling, run on virtual networks, and even completely isolate the functions.

**Azure Logic Apps**

Logic apps are similar to functions. Both enable you to trigger logic based on an event. Where functions execute code, logic apps execute *workflows* that are designed to automate business scenarios and are built from predefined logic blocks.

Every Azure logic app workflow starts with a trigger, which fires when a specific event happens or when newly available data meets specific criteria. Many triggers include basic scheduling capabilities, so developers can specify how regularly their workloads will run. Each time the trigger fires, the Logic Apps engine creates a logic app instance that runs the actions in the workflow. These actions can also include data conversions and flow controls, such as conditional statements, switch statements, loops, and branching.

You create logic app workflows by using a visual designer on the Azure portal or in Visual Studio. The workflows are persisted as a JSON file with a known workflow schema.

Azure provides more than 200 different connectors and processing blocks to interact with different services. These resources include the most popular enterprise apps. You can also build custom connectors and workflow steps if the service you need to interact with isn't covered. You then use the visual designer to link connectors and blocks together. You pass data through the workflow to do custom processing, often all without writing any code.

As an example, let's say a ticket arrives in Zendesk. You could:

* Detect the intent of the message with cognitive services.
* Create an item in SharePoint to track the issue.
* Add the customer to your Dynamics 365 CRM system if they aren't already in your database.
* Send a follow-up email to acknowledge their request.

All of those actions could be designed in a visual designer, which makes it easy to see the logic flow. For this reason, it's ideal for a business analyst role.

**Functions vs. Logic Apps**

Functions and Logic Apps can both create complex orchestrations. An orchestration is a collection of functions or steps that are executed to accomplish a complex task.

* With Functions, you write code to complete each step.
* With Logic Apps, you use a GUI to define the actions and how they relate to one another.

You can mix and match services when you build an orchestration, calling functions from logic apps and calling logic apps from functions. Here are some common differences between the two.

**Functions**

**Logic Apps**

State

Normally stateless, but Durable Functions provide state.

Stateful.

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, Log Analytics.

Management

REST API, Visual Studio.

Azure portal, REST API, PowerShell, Visual Studio.

Execution context

Can run locally or in the cloud.

Runs only in the cloud.