**Azure Logic Apps**

A business workflow can be run on Azure using Logic apps. The Logic App is a logical container for one workflow you can define using triggers and actions. A trigger can instantiate a workflow, which can consist of one or many activities (actions).

A Logic App runs on the infrastructure of an Azure region (VM’s in a data centre), not visible to us as it is abstracted away. By provisioning a Logic App, we leverage a bit of that infrastructure (indirectly via the Logic App Service) once we define a workflow and the flow gets triggered. Furthermore, we get billed by some trigger/actions that execute, and scaling done for us – if let’s say the number of request increases, so will the Logic App instances until certain boundaries are reached (see Logic App Limitations).

**Azure Functions**

Azure Functions are part of the Azure Web + Mobile suite of App Services and are designed to enable the creation of small pieces of meaningful, reusable methods, easily shared across services. We can build the Azure Function in various languages like Node.js, C#, F#, Python, PHP, and even Java. Followed by scripting languages, Bash, and PowerShell – and support for Command Line through CMD- or BAT file. Although an Azure Function can contain quite a bit of code. They are typically designed to serve a single purpose and respond to events in connected services.

**Comparing Logic Apps and Azure Functions**

**Connectivity**

**Logic app:**

* One of the fundamental concept of logic app is connectors
* The connectors provide a means to trigger a workflow and a payload into or out of the flow
* In case a connector doesn’t exist we can build a connector through the custom connector template in the Azure Marketplace

**Azure functions:**

* Azure Functions do not have connectors, yet rely on triggers and input and output bindings – like Storage, Event Hubs, Service Bus, and Cosmos DB

**Security**

**Logic app:**

* Every connector comes with its security depending on what API it exposes
* We can secure the Logic App by placing Azure API Management before it

**Azure functions:**

* With Azure Functions the security is different – we can secure accessible, public functions (HTTP and WebHook Binding) with Authorization keys
* Keys are generated when we create a function and can be regenerated

**Exception handling**

**Logic app:**

* Logic Apps exception handling is an out-of-the functionality with retries, scopes, and run after configurations

**Azure functions:**

* Azure Functions you can apply the standard try-catch

Note:- With Azure Functions we do not have the out-of-the-box retry capability like Logic Apps except for Azure Queue and Blob Storage. For more details on error handling with Azure Functions see [Azure Functions error handling](https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-error-pages).

**Scaling**

* The consumption model for Logic Apps and Azure Functions provide a specific auto-scale capability, i.e. when your load increases the Logic App or Function can scale with it to a certain point. Logic Apps can scale depending on the connector(s) you use, for instance, the File System connector current limit of 100 calls per minute.
* Like Logic Apps, Azure Functions can run under a consumption plan – instances of the Azure Functions host are dynamically added and removed based on the number of incoming events. The plan supports the auto scale – read more about it through how consumption plan works. Besides a consumption plan, Azure Functions can also run under an App Service Plan, which means specifying the number of dedicated VM’s through a Basic, Standard, Premium, or Isolated SKU. Having an App Service Plan instead of a consumption plan depends on your requirements for more processing power that exceeds the consumption plan, or you need to support scenarios where for instance your function apps need to run continuously.

**Deployment**

* With deployments of resources in Azure, one can think of the Azure Resource Manager (ARM) templates. This also accounts for deploying Logic Apps and Azure Functions. Both have built-in support for these templates. You can automate the deployment of Logic App through the Azure Portal leveraging the automation script or by using Visual Studio.
* With Azure Functions, you can quickly set up continuous deployments through triggers from sources like Bitbucket, Dropbox, Git, GitHub, OneDrive, and VSTS (see for example Leveraging Functions on the Azure Platform blog post). Additionally, Azure Functions have deployment slots in the preview, allowing deploying and testing a vNext first, before you swap the tested deployment slot with the current version in production.

**Monitoring**

* Logic Apps monitoring and Azure Functions monitoring can be done with available Azure monitoring capabilities (OMS, Application Insights, Log Analytics) and built-in features
* Logic Apps provides a run-history feature enabling you to examine specific runs in full detail. Also, you can filter this history, based on a period and the resulting run status. Furthermore, Logic Apps integrates with Operation Management Suite (OMS) in Azure – for instance with one button click you can enable integration with OMS, where you can search for tracked properties
* With Azure Functions you have access to the monitor tab, here you can see the execution history. Besides that, there is a live event stream that shows the almost real-time processing statistics in graphs. Furthermore, there’s full integration with Application Insights, where you can take advantage of the powerful Analytics queries.

**Billing**

* When comparing Logic Apps and Azure Functions, both can run under a consumption plan. That means both have a pay-per-usage billing model.

|  |  |  |
| --- | --- | --- |
| **Consumption Plan** | **Azure Logic Apps** | **Azure Functions** |
| Price/Unit of Measure | Price per Execution:  **Actions €0.000022**  **Standard Connector €0.000106**  **Enterprise Connector   €0.000844** | Execution Time **€0.000014/GB-s**  Total Executions **€0.169 per million execution**  *Consumption plan pricing includes a monthly free grant of 1 million requests and 400,000 GB-s of resource consumption per month.* |
| Details | <https://azure.microsoft.com/en-us/pricing/details/logic-apps/> | [https://azure.microsoft.com/en-us/pricing/details/functions](https://azure.microsoft.com/en-us/pricing/details/functions/) |

**Note**: Azure Functions can also run under App Service Plan – hence a different pricing model.

In case you leverage the integration account with your Logic Apps than you should be aware of additional costs.

**When to use Logic Apps**

Logic Apps are “event-driven” meaning they run based on a trigger. That is an HTTP request, message on a queue, a blob created in a container, or schedule.

Typical scenarios for Logic Apps are:

* SaaS event processing
* Timer-based processing
* Data Ingestion (see for instance the following Middleware Friday episode)
* Business Process
* Integration between (Cloud) services
* Content-based routing
* Data transformation and enrichment

**When to use Functions**

Azure Functions like Logic Apps are “event-driven” meaning they run based on associated and configured events, or “triggers”. For example, an Azure Function could be triggered by a simple timer. Such as running a process once every 24-hour or triggered by an event in a document management system. Also, when a new document is uploaded to a SharePoint library. Azure Functions can also respond to Azure-specific events. Such as an image added to a Storage Blob or a notification arriving in a Message Queue.

Typical scenarios for Azure Functions are:

* Timer-based processing
* Azure service event processing
* SaaS event processing
* Serverless web application architectures
* Serverless mobile backends
* Real-time stream processing
* Real-time bot messaging

For more details refer - [Logic app vs Azure function](https://www.serverless360.com/blog/when-to-use-logic-apps-and-azure-functions)