Azure Functions - Overview of features, road map

Azure Functions don't need any introduction, they are server-less hosting of your code without worrying too much about infrastructure unlike a web API project. Originally started as stateless runtime, soon evolved into support of extension to develop and host state full functions called ***Dura ble Functions*** (*more about this in a separate section below*)

You can develop Azure Functions in numerous languages / framework in the following languages/frameworks.

When to use Azure Functions

Following chart shows the different use cases of Azure Functions, best use case for Azure Function is to do ***Async pattern work*** - when an event happens it would trigger the azure function(s) via web hook invoke to execute code. Well known publisher / subscriber pattern use case.

If you want to...	then...
Build a web API	Implement an endpoint for your web applications using the HTTP trigger
Process file uploads	Run code when a file is uploaded or changed in blob storage
Build a server less workflow	Chain a series of functions together using durable functions
Respond to database changes	Run custom logic when a document is created or updated in Cosmos DB
Run scheduled tasks	Execute code on pre-defined timed intervals
Create reliable message queue systems	Process message queues using Queue Storage, Service Bus, or Even t Hubs
Analyze IoT data streams	Collect and process data from IoT devices
Process data in real time	Use Functions and SignalR to respond to data in the moment

Languages / Framework Supported by Azure Functions (non Durable)

Language	1.x	2.x	3.x	4.x
C#	GA (.NET Framework 4.8)	GA (.NET Core 2.1[1])	GA (.NET Core 3.1) GA (.NET 5.0)	GA (.NET 6.0) Preview (.NET Framework 4.8)
JavaScript	GA (Node.js 6)	GA (Node.js 10 & 8)	GA (Node.js 14, 12, & 10)	GA (Node.js 14) GA (Node.js 16)
F#	GA (.NET Framework 4.8)	GA (.NET Core 2.1[1])	GA (.NET Core 3.1)	GA (.NET 6.0)
Java	N/A	GA (Java 8)	GA (Java 11 & 8)	GA (Java 11 & 8)
PowerShell	N/A	GA (PowerShell Core 6)	GA (PowerShell 7.0 & Core 6)	GA (PowerShell 7.0) Preview (PowerShell 7.2)
Python	N/A	GA (Python 3.7 & 3.6)	GA (Python 3.9, 3.8, 3.7, & 3.6)	GA (Python 3.9, 3.8, 3.7)
TypeScript[2]	N/A	GA	GA	GA

C# vs TypeScript

Fairway Architecture team recommends using C# .NET 6 and above to develop Azure Trigger or Durable Functions as Microsoft is known to keep C# / .NET tooling up to date along with better runtime performance and memory optimization support.

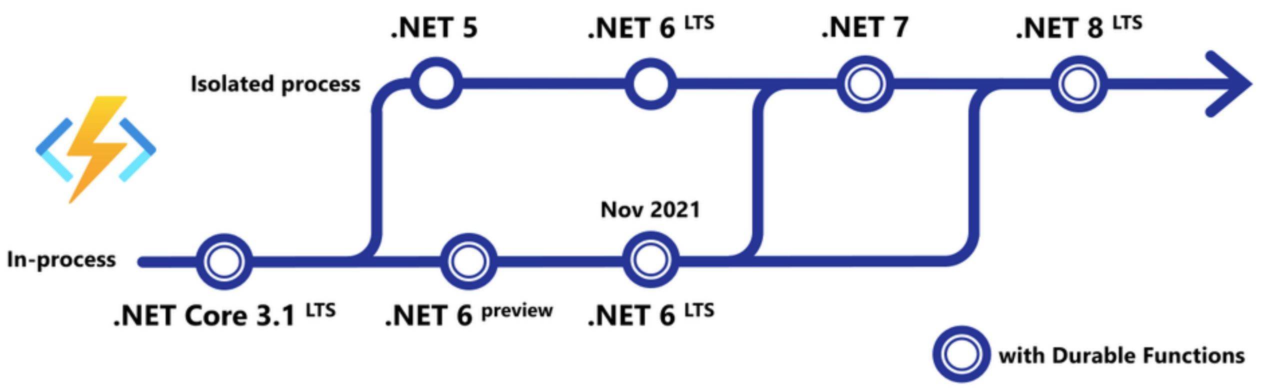
Architecture team will have a reference architecture for Azure Function in .NET / C# 6 with In-Process implementation. We will work on TypeScript reference architecture if there is a real need to use TypeScript.

Future of .NET Azure Functions - Isolated Process

Other languages supported by Azure Functions use an ***out-of-process*** model that runs a language worker in a separate, isolated process from the main Azure Functions host. This allows Azure Functions to rapidly support new versions of these languages without updating the host, and it allows the host to evolve independently to enable new features and update its own dependencies over time.

With .NET 5, support for ***isolated process*** was added to C# and other .NET based languages as you see the below diagram that shows the road map of this feature.

However, the ***durable functions*** is currently in ***preview for .NET 6***, hence our examples will be using the ***in- process*** mode as of now. In future we will update the reference architecture with the isolated sample code.



Advantages of isolated process

Main advantages we see are

- Full support for Dependency injection
- Middleware injection
- Assemblies used will not conflict with host

Durable Functions

Durable Functions is an extension of Azure Functions that lets you write state full functions in a server less compute environment. The extension lets you define state full workflows by writing *orchestrator functions* and state full entities by writing *entity functions* using the Azure Functions programming model. Behind the scenes, the extension manages state, checkpoints, and restarts for you, allowing you to focus on your business logic.

Supported languages for Durable Functions

Language stack	**Azure Functions Runtime versions**	**Language worker version**	**Minimum bundles version**
.NET / C# / F#	Functions 1.0+	In-process (GA) Out-of-process (preview)	N/A
JavaScript/TypeScript	Functions 2.0+	Node 8+	2.x bundles
Python	Functions 2.0+	Python 3.7+	2.x bundles
PowerShell	Functions 3.0+	PowerShell 7+	2.x bundles
Java (coming soon)	Functions 3.0+	Java 8+	4.x bundles

Application patterns

The primary use case for Durable Functions is simplifying complex, state full coordination requirements in server less applications. The following sections describe typical application patterns that can benefit from Durable Functions:

- Function chaining
- Fan-out/fan-in
- Async HTTP APIs
- Monitoring
- Human interaction
- Aggregator (stateful entities)

Few links for further reading

- Guide for running C# Azure Functions in an isolated process | Microsoft Docs
- Integration and automation platform options in Azure | Microsoft Docs

- Azure Functions scale and hosting | Microsoft Docs
- Durable Functions Overview - Azure | Microsoft Docs