Going ServerLess with Azure Functions

Vaibhav Gujral
@vabgujral

Introduction

- Enterprise Architect with over 12 years of experience
- Microsoft Certified Azure Architect and Microsoft Certified Professional
- Working on Microsoft Azure since early days
- Worked across windows, web and cloud based systems
- Currently employed with American Title Inc.
- Speaker at OmahaMTG
- Twitter - @vabgujral
- Email – gujral.vaibhav@hotmail.com

Agenda

- What is ServerLess Computing?
- What are Azure Functions?
  - Azure Functions Tooling
- Azure Functions Concepts
  - Azure Functions Runtime
  - Triggers & Bindings
- Azure Functions Pricing
  - Consumption plan vs App Service plan
- Deploy & Monitor Azure Functions
- Best Practices
- Durable Functions & Azure Functions Proxies

What is ServerLess Computing?

- Doesn't mean No-Server, rather, think of it as Less-Server
- Abstraction of servers, infrastructure, and operating systems
- Backend-as-a-Service could be one acronym
  - Fully incorporated cloud hosted applications
  - Example – Auth0
- Function-as-a-Service is another acronym
  - Execute individual code pieces as functions
  - Examples-
    - Azure Functions
    - AWS Lambda
    - Google Cloud Functions

Pros/Cons of ServerLess

Pros

- Removes need for server management and always-on server components
- Auto-Scaling
- Reduced costs
  - Operational Costs
  - Scaling costs

Cons

- Lack of Control
- Security Concerns

Azure Functions

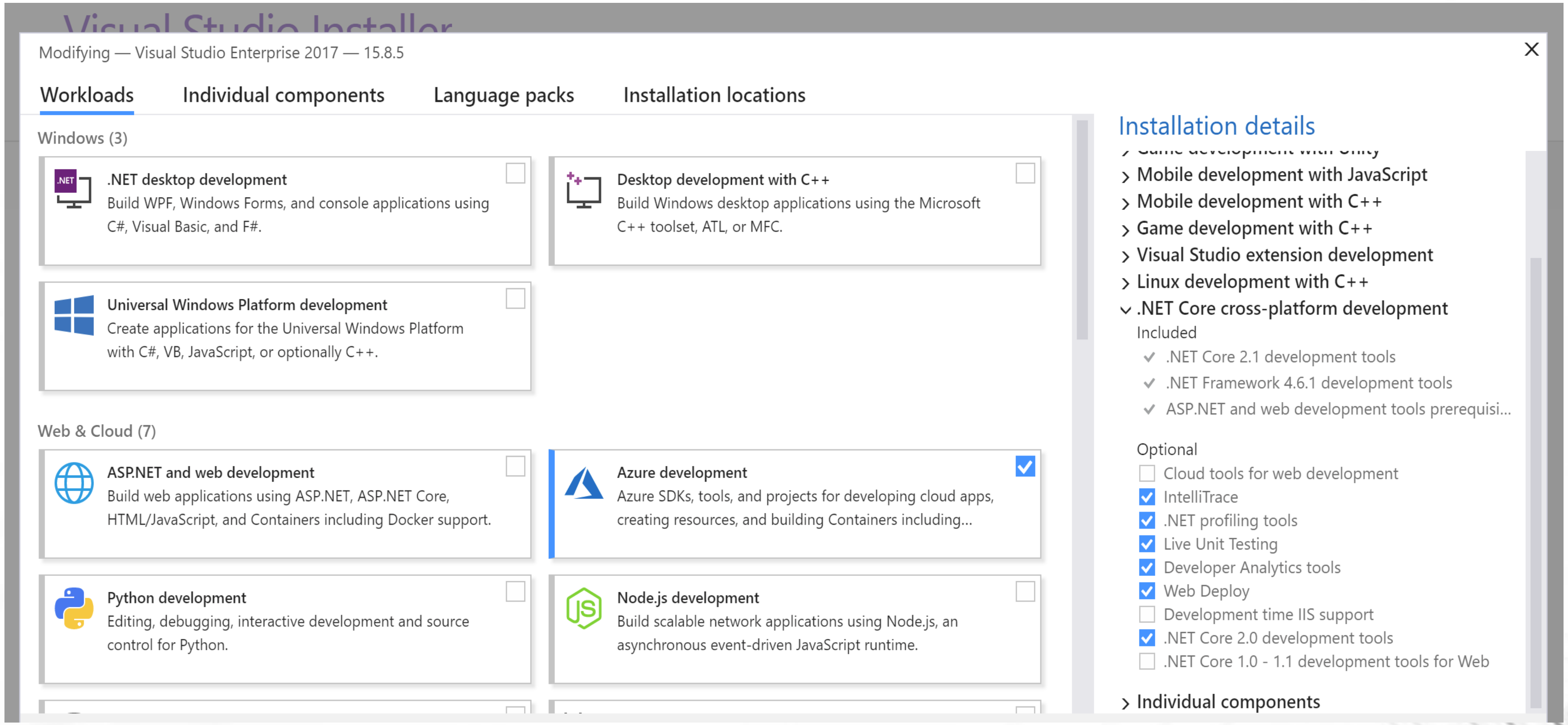
What are Azure Functions?

- Serverless compute service that can run code on-demand
- Run small pieces of code in Azure (call them as “functions”)
- Provides out of the box templates for some of the most common scenarios
- Useful in common scenarios like –
  - Connecting to Storage
  - Image processing
  - Exposing HTTP based APIs
  - IoT
  - Running a script or code in response to a variety of events etc
- Azure Functions is a serverless evolution of Azure WebJobs

Azure Functions Features

- Choice of Language –
  - Supports C#, F#, and JavaScript.
  - Java support in preview
- Pay per use (only for the time the code is executed)
- Nuget and NPM support
- Integrated Security – Oauth support for Http-triggered functions
- Seemless integration with other Azure Services
- Flexible Development
- Azure Functions runtime is open-source

Azure Function Tooling – Visual Studio



Visual Studio Installer

Modifying — Visual Studio Enterprise 2017 — 15.8.5

Windows (3)

- [ ] .NET desktop development — Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.
- [ ] Desktop development with C++ — Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.
- [ ] Universal Windows Platform development — Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

Web & Cloud (7)

- [ ] ASP.NET and web development — Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.
- [✓] Azure development — Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including...
- [ ] Python development — Editing, debugging, interactive development and source control for Python.
- [ ] Node.js development — Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Installation details

- Mobile development with JavaScript
- Mobile development with C++
- Game development with C++
- Visual Studio extension development
- Linux development with C++
- .NET Core cross-platform development

Included

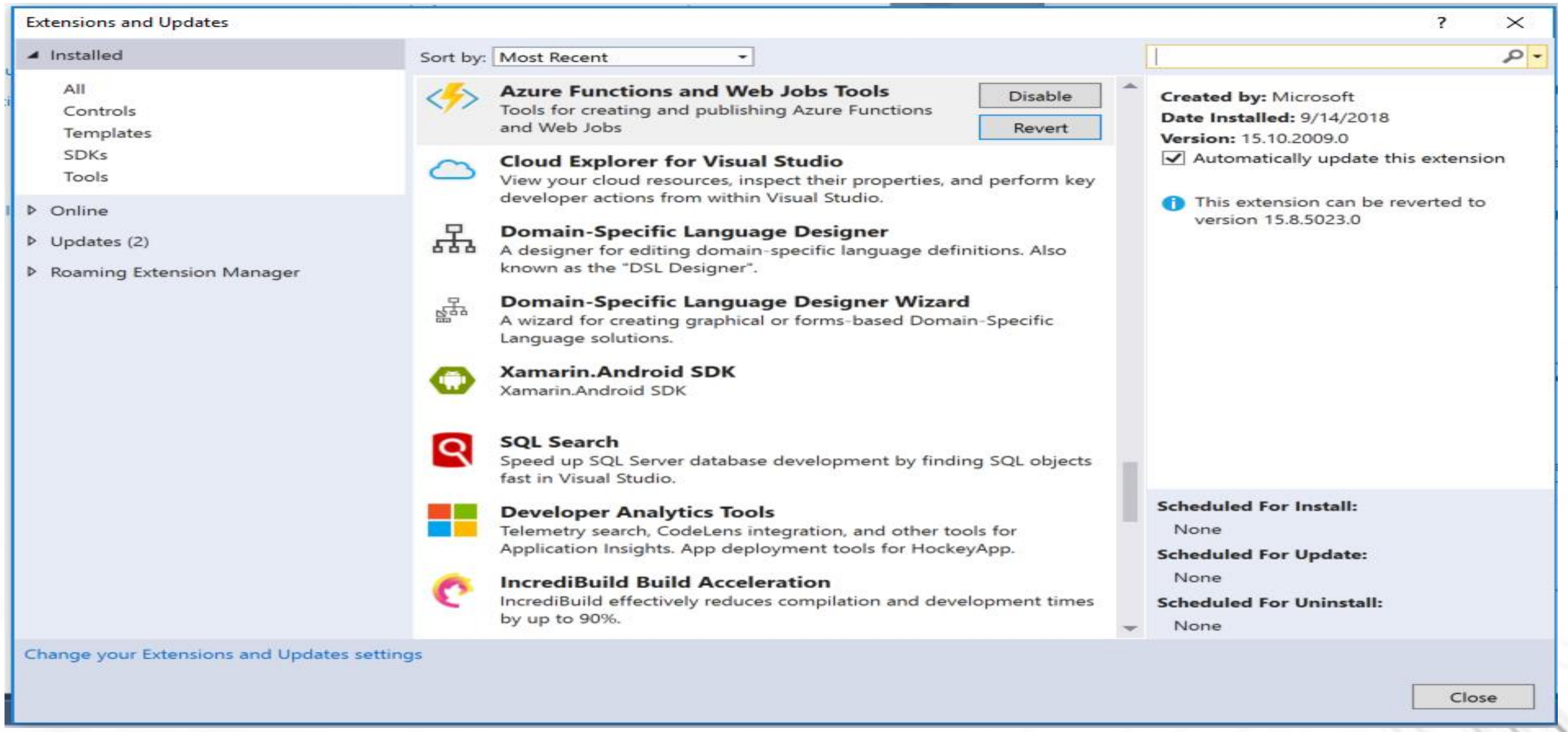
- ✓ .NET Core 2.1 development tools
- ✓ .NET Framework 4.6.1 development tools
- ✓ ASP.NET and web development tools prerequisi...

Optional

- [ ] Cloud tools for web development
- [✓] IntelliTrace
- [✓] .NET profiling tools
- [✓] Live Unit Testing
- [✓] Developer Analytics tools
- [✓] Web Deploy
- [ ] Development time IIS support
- [✓] .NET Core 2.0 development tools
- [ ] .NET Core 1.0 - 1.1 development tools for Web

Individual components

Azure Function Tooling – Visual Studio


Extensions and Updates

Azure Functions and Web Jobs Tools
Tools for creating and publishing Azure Functions and Web Jobs

Cloud Explorer for Visual Studio
View your cloud resources, inspect their properties, and perform key developer actions from within Visual Studio.

Domain-Specific Language Designer
A designer for editing domain-specific language definitions. Also known as the "DSL Designer".

Domain-Specific Language Designer Wizard
A wizard for creating graphical or forms-based Domain-Specific Language solutions.

Xamarin.Android SDK
Xamarin.Android SDK

SQL Search
Speed up SQL Server database development by finding SQL objects fast in Visual Studio.

Developer Analytics Tools
Telemetry search, CodeLens integration, and other tools for Application Insights. App deployment tools for HockeyApp.

IncrediBuild Build Acceleration
IncrediBuild effectively reduces compilation and development times by up to 90%.

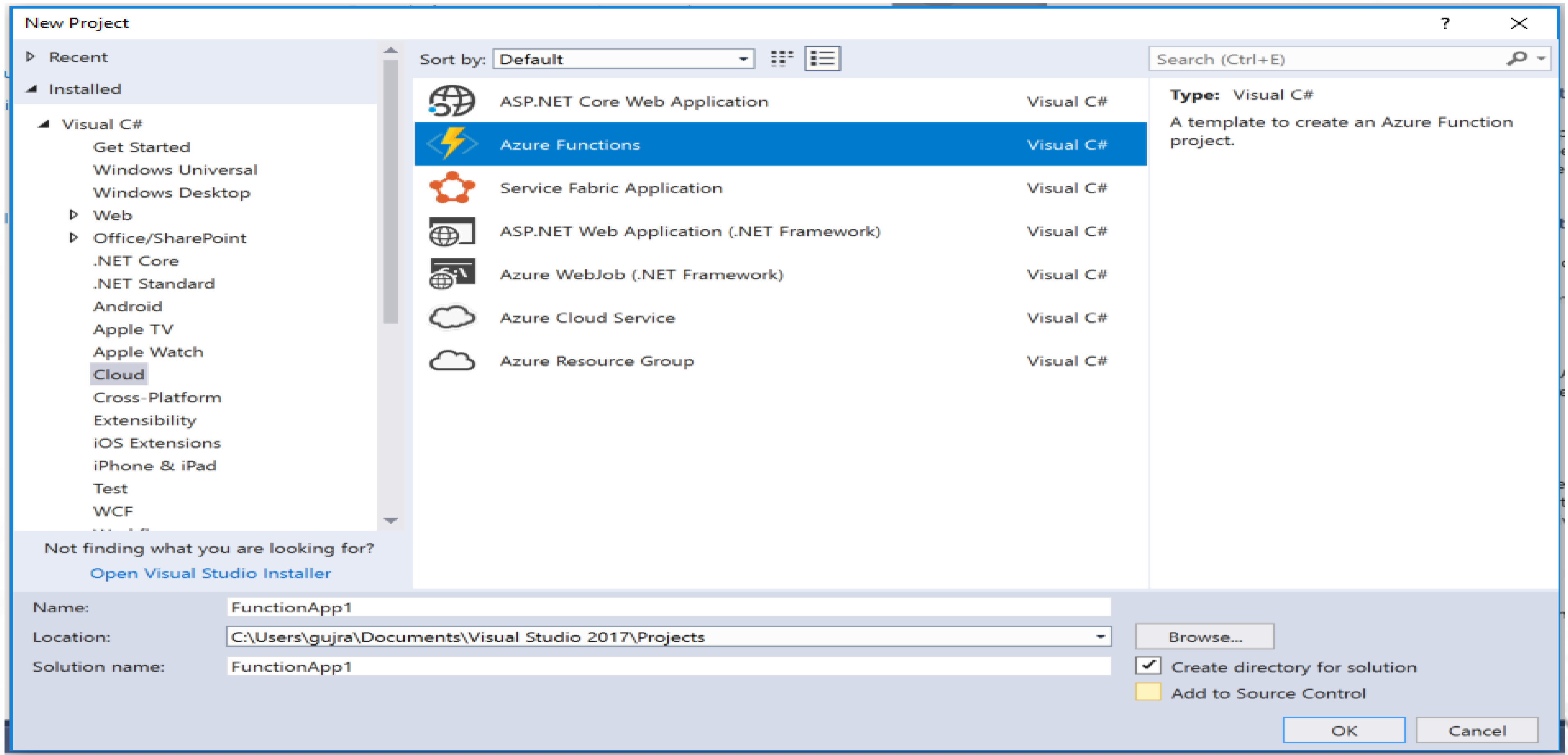
Created by: Microsoft
Date Installed: 9/14/2018
Version: 15.10.2009.0
✓ Automatically update this extension

This extension can be reverted to version 15.8.5023.0

Scheduled For Install:
None
Scheduled For Update:
None
Scheduled For Uninstall:
None

Change your Extensions and Updates settings

Azure Function Tooling – Visual Studio


New Project

- Recent
- Installed
  - Visual C#
    - Get Started
    - Windows Universal
    - Windows Desktop
    - Web
    - Office/SharePoint
    - .NET Core
    - .NET Standard
    - Android
    - Apple TV
    - Apple Watch
    - Cloud
    - Cross-Platform
    - Extensibility
    - iOS Extensions
    - iPhone & iPad
    - Test
    - WCF

Not finding what you are looking for?
Open Visual Studio Installer

Sort by: Default

Search (Ctrl+E)

ASP.NET Core Web Application	Visual C#
Azure Functions	Visual C#
Service Fabric Application	Visual C#
ASP.NET Web Application (.NET Framework)	Visual C#
Azure WebJob (.NET Framework)	Visual C#
Azure Cloud Service	Visual C#
Azure Resource Group	Visual C#

Type: Visual C#

A template to create an Azure Function project.

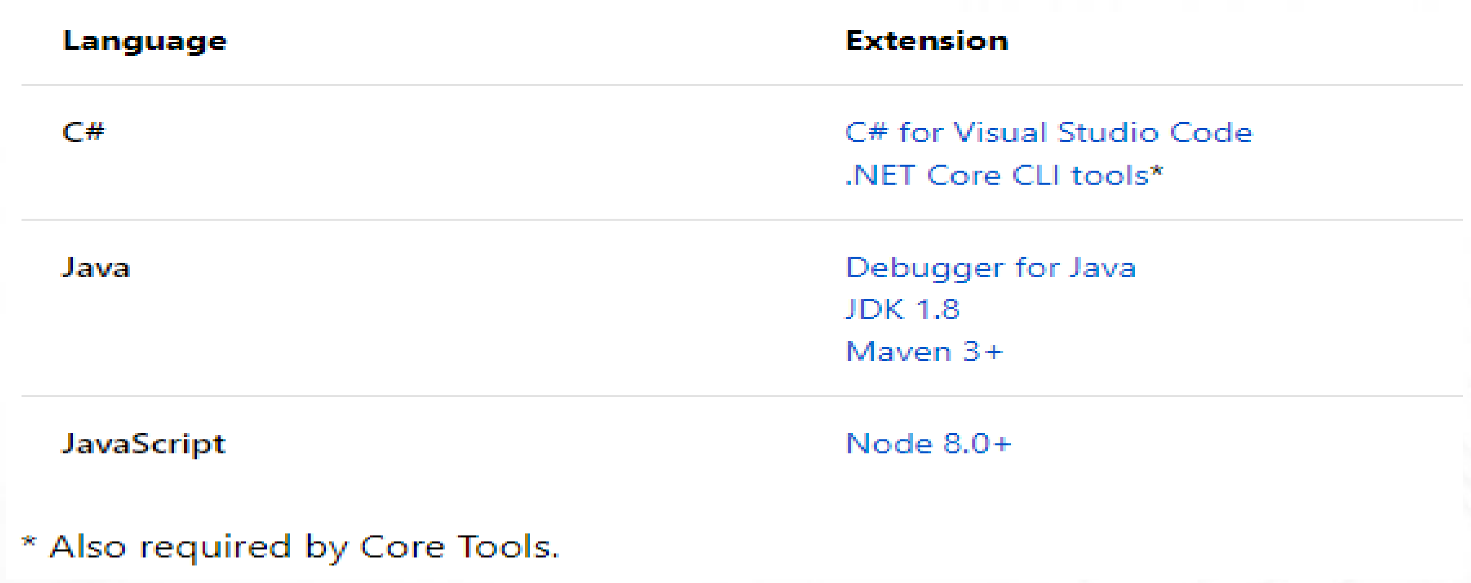
Name: FunctionApp1
Location: C:\Users\gujra\Documents\Visual Studio 2017\Projects
Solution name: FunctionApp1

Browse...
[✓] Create directory for solution
[ ] Add to Source Control

OK Cancel

Azure Function Tooling – Visual Studio Code

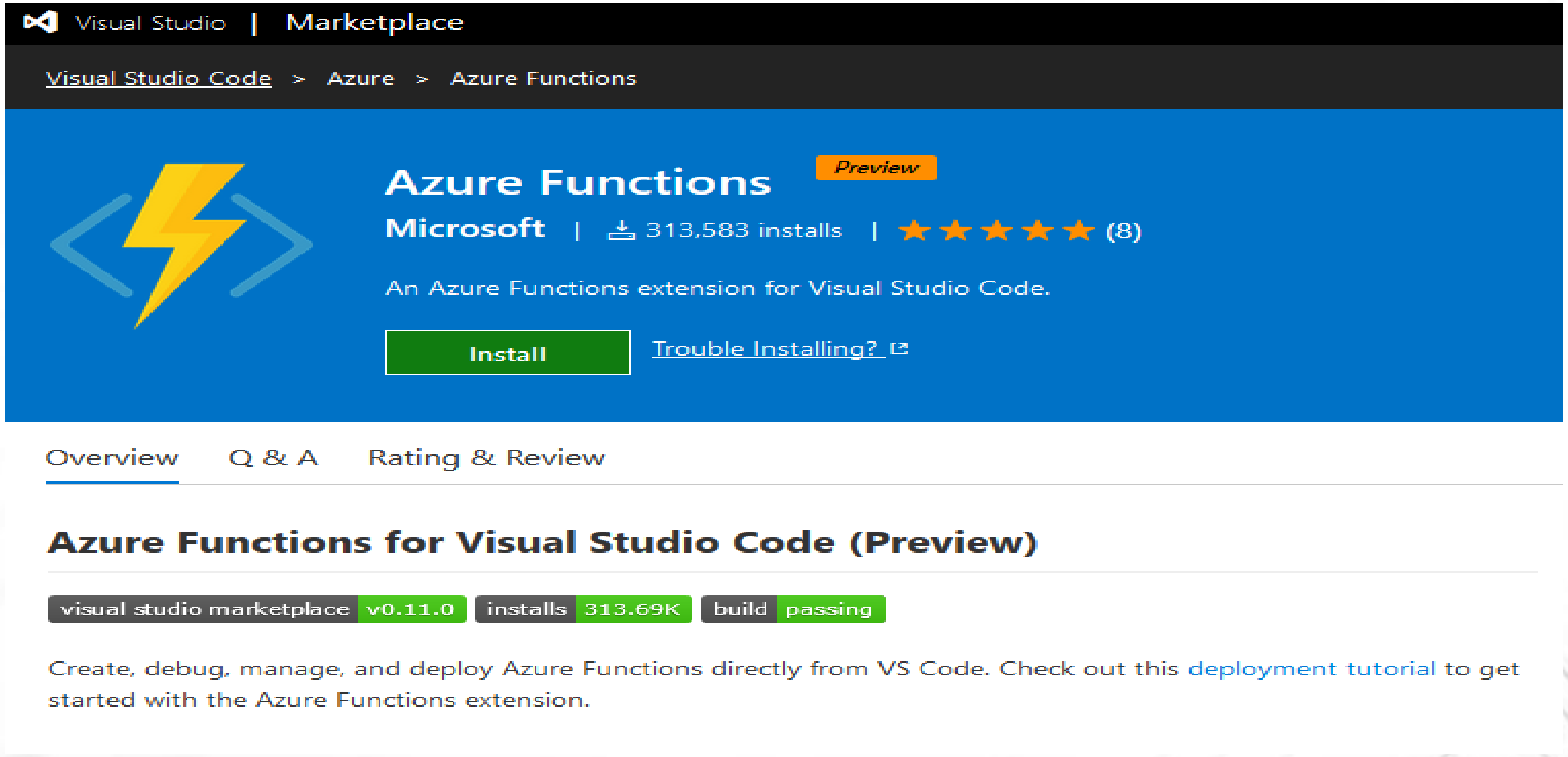
- Pre-requisites
  - Azure Function Core Tools
    - Version 2.x
  - Required language support:



Language	Extension
C#	C# for Visual Studio Code .NET Core CLI tools*
Java	Debugger for Java JDK 1.8 Maven 3+
JavaScript	Node 8.0+

* Also required by Core Tools.

Azure Function Tooling – Visual Studio Code


Visual Studio | Marketplace
Visual Studio Code > Azure > Azure Functions
Preview
Install
Trouble Installing?
Overview
Q & A
Rating & Review

Azure Functions

Microsoft | 313,583 installs | ★★★★★ (8)

An Azure Functions extension for Visual Studio Code.

Azure Functions for Visual Studio Code (Preview)

visual studio marketplace v0.11.0 | installs 313.69K | build passing

Create, debug, manage, and deploy Azure Functions directly from VS Code. Check out this deployment tutorial to get started with the Azure Functions extension.

Azure Functions Core Tools

- Azure Functions Core Tools
  - Develop/test/debug/deploy functions on local computer from command prompt or terminal
- v1.x
  - Supported on Windows
  - *npm install -g azure-functions-core-tools@v1*
- v2.x
  - Supports Windows, macOS and Linux
  - *npm install -g azure-functions-core-tools*
  - Needs .Net core 2.0 and Node.js (including npm)

Demo

Azure Functions Concepts

Azure Functions Integrations

- Functions can be integrated with various Azure and 3rd party services
- These services can either trigger the function execution or serve as input/output for function code
- Following services can be integrated with Azure Functions:
  - Azure CosmosDB
  - Azure Event Hubs
  - Azure Event Grid
  - Azure Notification Hubs
  - Azure Service Bus (queues and topics)
  - Azure Storage (blob, queues and tables)
  - On-Premises (using Service Bus)
  - Twilio (SMS messages)

What are Triggers?

- One of the Merriam-Webster's definition says –
  - "A Trigger is something that acts like a mechanical trigger in initiating a process or reaction"
- Defines how a function is invoked
- Out-of-the-box templates to trigger execution of an Azure function
- A function can have exactly one trigger
- A trigger can have an associated data, which is usually the payload that triggered the function
- Binding direction for triggers in always in

Supported Triggers

- HTTPTrigger
- TimerTrigger
- CosmosDBTrigger
- BlobTrigger
- QueueTrigger
- EventGridTrigger
- EventHubTrigger
- ServiceBusQueueTrigger
- ServiceBusTopicTrigger

What are Bindings?

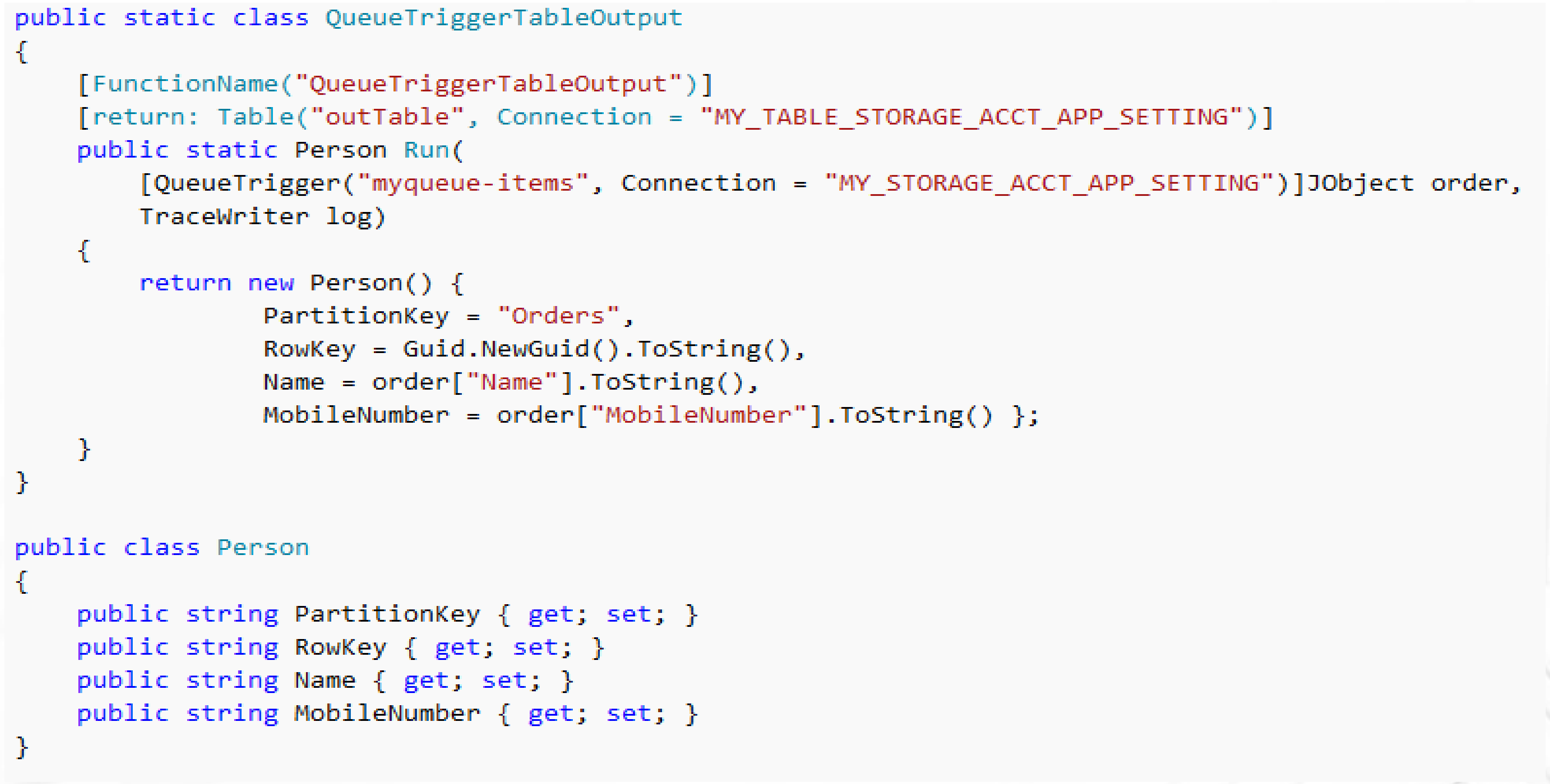
- Declarative way to make data from external services available to function code
- Bindings are optional
- Two types of bindings
  - Input
  - Output
- A function can have multiple input and output bindings
- Input and output bindings use in and out binding directions
- Some bindings support special binding direction – inout
- For runtime version 2.x, binding extensions are provided in NuGet packages, and to register an extension, package needs to be installed.

Trigger & Binding Definition

- Defined in function.json file.

```
{
  "bindings": [
    {
      "name": "order",
      "type": "queueTrigger",
      "direction": "in",
      "queueName": "myqueue-items",
      "connection": "MY_STORAGE_ACCT_APP_SETTING"
    },
    {
      "name": "$return",
      "type": "table",
      "direction": "out",
      "tableName": "outTable",
      "connection": "MY_TABLE_STORAGE_ACCT_APP_SETTING"
    }
  ]
}
```

Trigger & Binding Definition



```
public static class QueueTriggerTableOutput
{
    [FunctionName("QueueTriggerTableOutput")]
    [return: Table("outTable", Connection = "MY_TABLE_STORAGE_ACCT_APP_SETTING")]
    public static Person Run(
        [QueueTrigger("myqueue-items", Connection = "MY_STORAGE_ACCT_APP_SETTING")]JObject order,
        TraceWriter log)
    {
        return new Person() {
                PartitionKey = "Orders",
                RowKey = Guid.NewGuid().ToString(),
                Name = order["Name"].ToString(),
                MobileNumber = order["MobileNumber"].ToString() };
    }
}

public class Person
{
    public string PartitionKey { get; set; }
    public string RowKey { get; set; }
    public string Name { get; set; }
    public string MobileNumber { get; set; }
}
```

Trigger & Binding Definition

```
// From an incoming queue message that is a JSON object, add fields and write to Table Storage
// The second parameter to context.done is used as the value for the new row
module.exports = function (context, order) {
    order.PartitionKey = "Orders";
    order.RowKey = generateRandomId();

    context.done(null, order);
};

function generateRandomId() {
    return Math.random().toString(36).substring(2, 15) +
        Math.random().toString(36).substring(2, 15);
}
```

Binding Expressions

- Expressions that resolve to values from various sources.
- Most expressions are identified by wrapping them in curly braces.
- Expressions using AppSettings are wrapped in percent signs
- Types –
  - App Settings
  - Trigger File Names
  - Trigger Metadata
  - JSON Payloads
  - New Guid ({rand-guid})
  - Current Date and time ({DateTime})

Azure Functions Runtime Versions

- 1.x
  - Supports development and hosting only in Azure portal or on Windows
  - Bindings are part of runtime
    - Tightly coupled
- 2.x
  - Current version
  - Runs on .Net core 2
  - Runs on Windows, macOS and Linux
  - Uses binding extensibility model
    - Support for 3rd party binding extensions
    - Decoupling of runtime and bindings
    - Lighter execution environment

As of today, both versions are supported for production scenarios

Azure Functions Runtime Versions
- Language Support

Language	1.x	2.x
C#	GA (.NET Framework 4.7)	GA (.NET Core 2)
JavaScript	GA (Node 6)	GA (Node 8 & 10)
F#	GA (.NET Framework 4.7)	GA (.NET Core 2)
Java	N/A	Preview (Java 8)
Python	Experimental	N/A
TypeScript	Experimental	Supported through transpiling to JavaScript
PHP	Experimental	N/A
Batch (.cmd, .bat)	Experimental	N/A
Bash	Experimental	N/A
PowerShell	Experimental	N/A

Azure Functions Runtime Versions

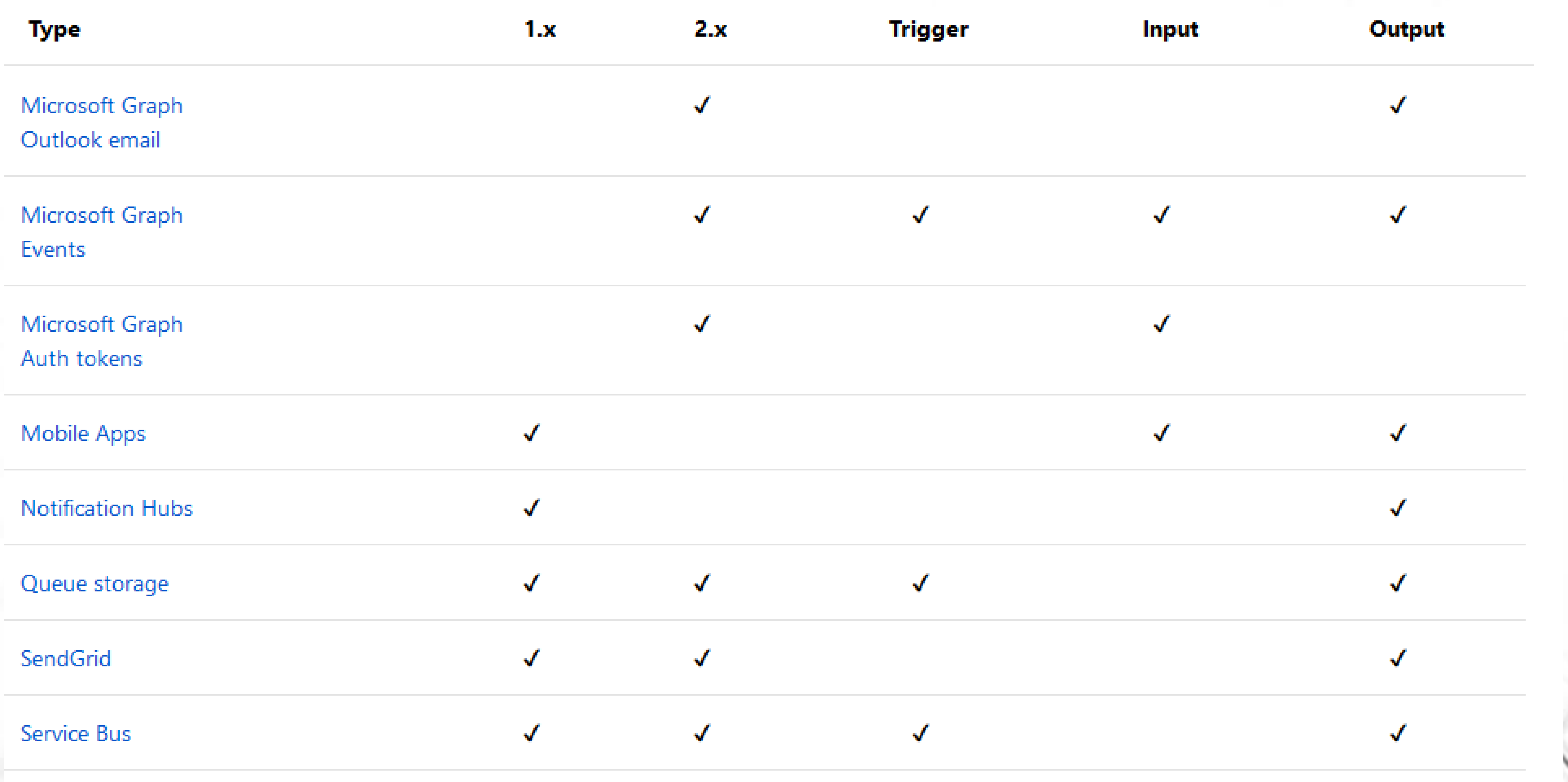
- Binding Extensions registrations

Development environment	Registration in Functions 1.x	Registration in Functions 2.x
Azure portal	Automatic	Automatic with prompt
Local using Azure Functions Core Tools	Automatic	Use Core Tools CLI commands
C# class library using Visual Studio 2017	Use NuGet tools	Use NuGet tools
C# class library using Visual Studio Code	N/A	Use .NET Core CLI

Supported Bindings

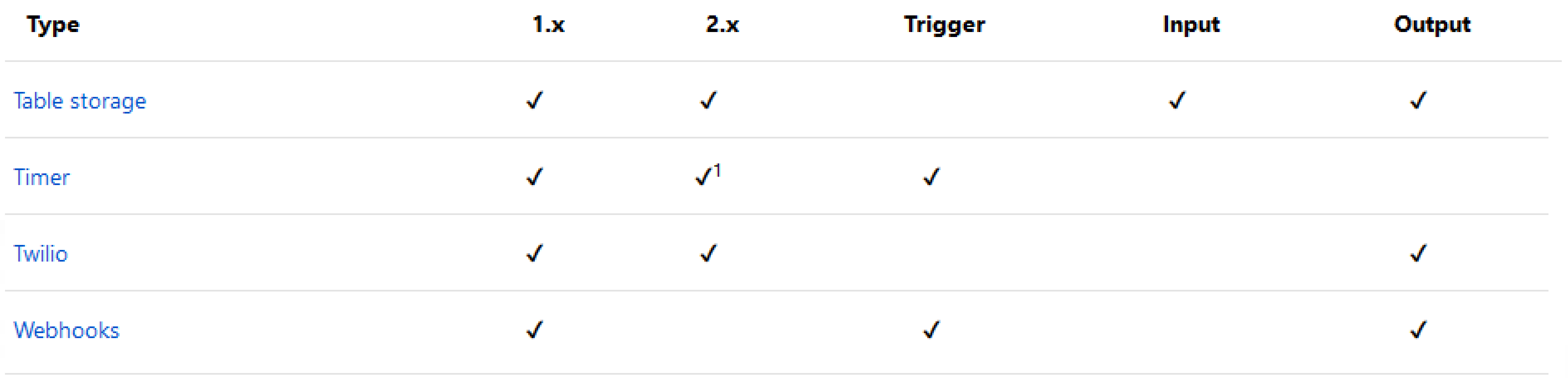
Type	1.x	2.x	Trigger	Input	Output
Blob Storage	✓	✓	✓	✓	✓
Cosmos DB	✓	✓	✓	✓	✓
Event Grid	✓	✓	✓		
Event Hubs	✓	✓	✓		✓
External File[2]	✓			✓	✓
External Table[2]	✓			✓	✓
HTTP	✓	✓[1]	✓		✓
Microsoft Graph Excel tables		✓		✓	✓
Microsoft Graph OneDrive files		✓		✓	✓

Supported Bindings



Type	1.x	2.x	Trigger	Input	Output
Microsoft Graph Outlook email		✓			✓
Microsoft Graph Events		✓	✓	✓	✓
Microsoft Graph Auth tokens		✓		✓	
Mobile Apps	✓			✓	✓
Notification Hubs	✓				✓
Queue storage	✓	✓	✓		✓
SendGrid	✓	✓			✓
Service Bus	✓	✓	✓		✓

Supported Bindings



Type	1.x	2.x	Trigger	Input	Output
Table storage	✓	✓		✓	✓
Timer	✓	✓[1]	✓		
Twilio	✓	✓			✓
Webhooks	✓		✓		✓

Azure Functions Pricing

Azure Functions Pricing

Two pricing plans:

- App Service Plan
  - Same as app service plan for a web app
- Consumption Plan
  - Pay for the time the code is executed
  - Default Hosting plan

App Service Plan

- Same as App Service Plan in App Services or a web app
- Dedicated resources based on the chosen tier like Basic or Standard
- Fixed cost
- Manual scaling and Auto scaling supported
- Run Function app just like a web app
- Useful if app service plan already exists for a web app and function app can run at no additional cost

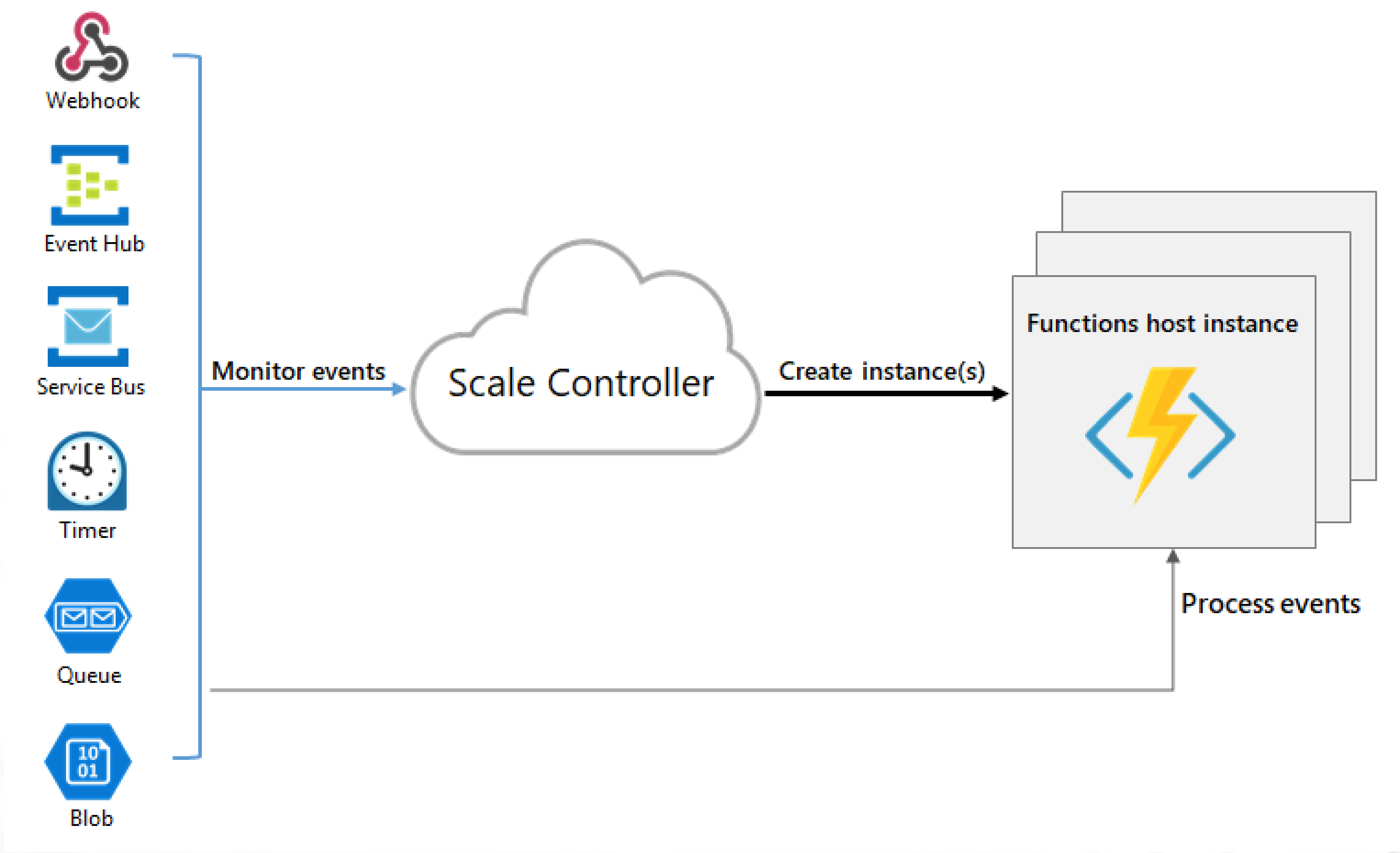
Consumption Plan

- Pay for the time that the code runs
- Automatically allocates compute power when code is running
- Adds or removes Functions Host (Function App) instances based on the number of events that its functions are triggered on
  - Each instance of function host is limited to 1.5 GB of memory
  - All functions within same function host share the same resources and are scaled together
- Function code files are stored on Azure File shares
- Function times out after configurable period of time (functionTimeout property)
  - Default – 5 mins
  - Maximum – 10 mins

Scaling under consumption plan

- Scale Controller
  - Monitors the rate of events and determines whether to scale out or scale in
  - Uses heuristics for each trigger type
- The unit of scale is a function app
- When the function app is scaled out or down, resources are de/allocated to run multiple instances of the Azure Functions host.
- The number of instances is scaled down to zero when no functions are running within a function app.
- A single function app only scales up to a maximum of 100 instances.
- New instances are allocated at most once every 10 seconds.

Scaling under consumption plan



Billing Model under consumption plan

- Usage is aggregated at the function app level
- Counts only the time that function code is executed
- The following are units for billing:
  - **Resource consumption in gigabyte-seconds (GB-s)**. Computed as a combination of memory size and execution time for all functions within a function app.
  - **Executions**. Counted each time a function is executed in response to an event trigger.

Consumption vs App Service Plan

- Can existing resource be leveraged?
- Is the function supposed to run continuously?
- What are the CPU and Memory requirements to run the code?
- Will the function run more than max execution timeout under consumption plan (10 minutes)
- Will the function need features only supported by App service plan like ASE, VNET etc.
- Whether the code needs to run on Linux or a custom OS Image

Note – Irrespective of the chosen hosting plan, a general-purpose storage account is required alongside function app to store all the files and logging function executions

Always On

- Under App service plan, Function App goes idle after few minutes of inactivity
- Only HttpTriggers can wake up a function
- Under app service plan, enable “Always on” to keep the functions running continuously and correctly
- Under Consumption plan, function apps are activated automatically
- When using a blob trigger on a Consumption plan, there can be up to a 10-minute delay in processing new blobs.

Monitor Azure Functions

- Built-in logging mechanism based on Azure Storage
  - Useful for non-prod environments with light workloads
  - Can be disabled by deleting AzureWebJobsDashboard app setting
- Built-in support for Application Insights
  - Recommended for production workloads
  - If this is enabled, built-in logging on Azure Storage should be disabled
  - Telemetry data can be further queried using Application Insights Analytics
  - Telemetry includes traces, requests, exceptions, customMetrics, customEvents and performanceCounters

Azure Functions – Continuous Deployment

- Directly deploy through source control of your choice
- Deployments are configured per-functionapp basis
- If continuous deployment is enabled, the access to function code in the portal is set read-only
- Azure DevOps (previously VSTS) offers full support for Azure Functions
- Other ways to deploy-
  - Zip Deployment
  - Through ARM template
  - Through deployment package

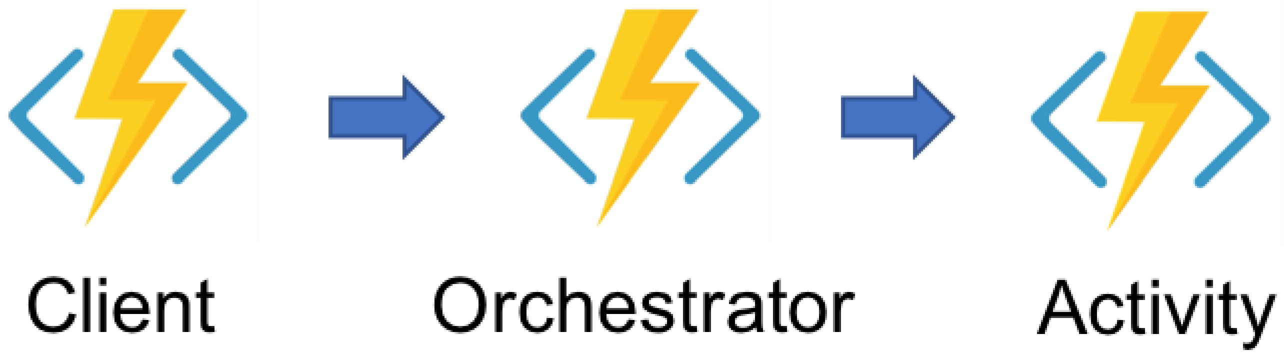
Demo

Best Practices

- Avoid long running functions
  - Refactor large functions into smaller function sets
- Use Azure Storage queues for cross-function communication
- Functions should be stateless and idempotent if possible
- Re-use external connections whenever possible
- Don't mix test and production data in same function app
- Use async code but avoid blocking calls
- Receive messages in batch whenever possible
- Write defensive functions

Durable Functions

- An extension of Azure Functions
- Enables stateful functions in server-less environment
- Simplifies complex, stateful coordination problems in serverless applications
- Provides stateful orchestration of function execution

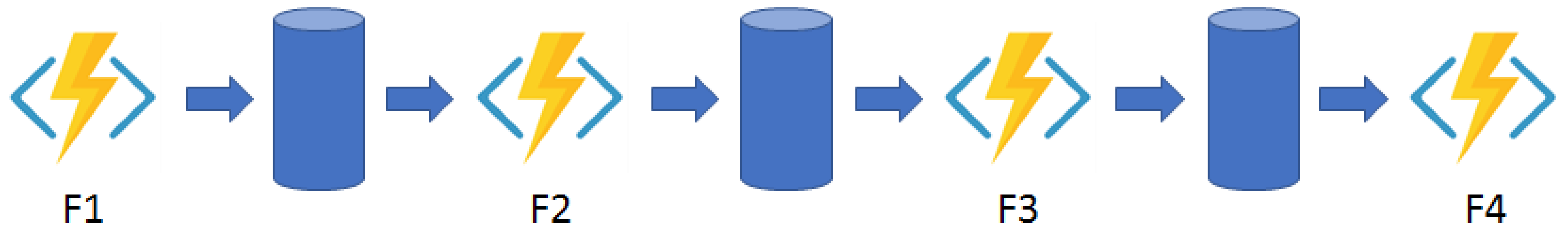


Durable Functions

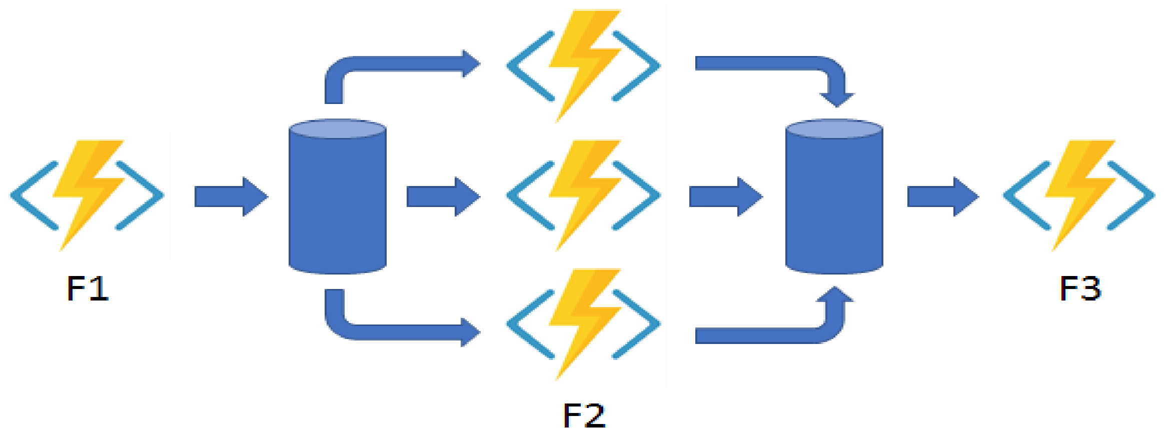
- Built on top of Durable Task Framework
  - As a serverless evolution
- Currently C#(v1 & v2), F# and Javascript(v2 only) are supported languages
- Some typical application patterns supported by durable functions includes –
  - Function Chaining
  - Fan-out/fan-in
  - Async Http APIs
  - Monitoring
  - Human Interaction

Durable Functions

- Function Chaining

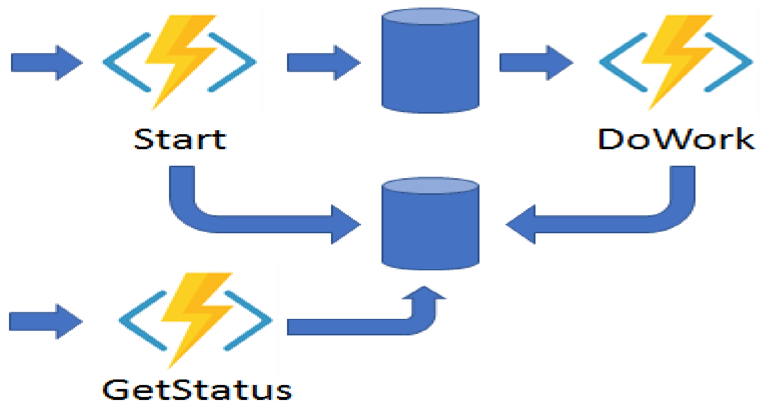


- Fan-out/fan-in

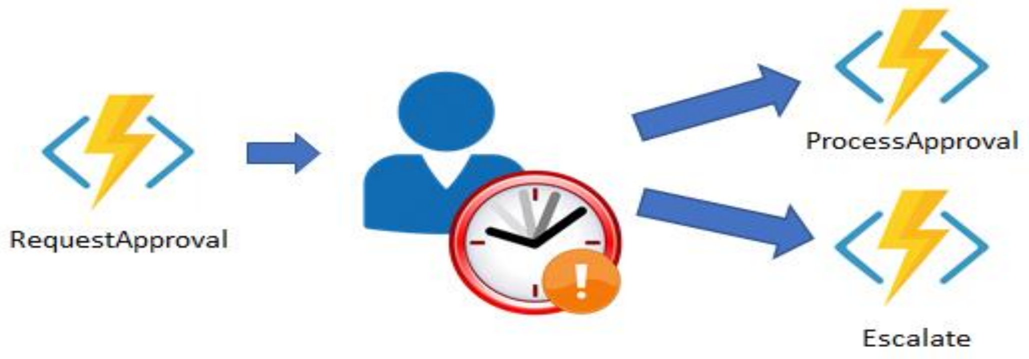


Durable Functions

- Async Http APIs

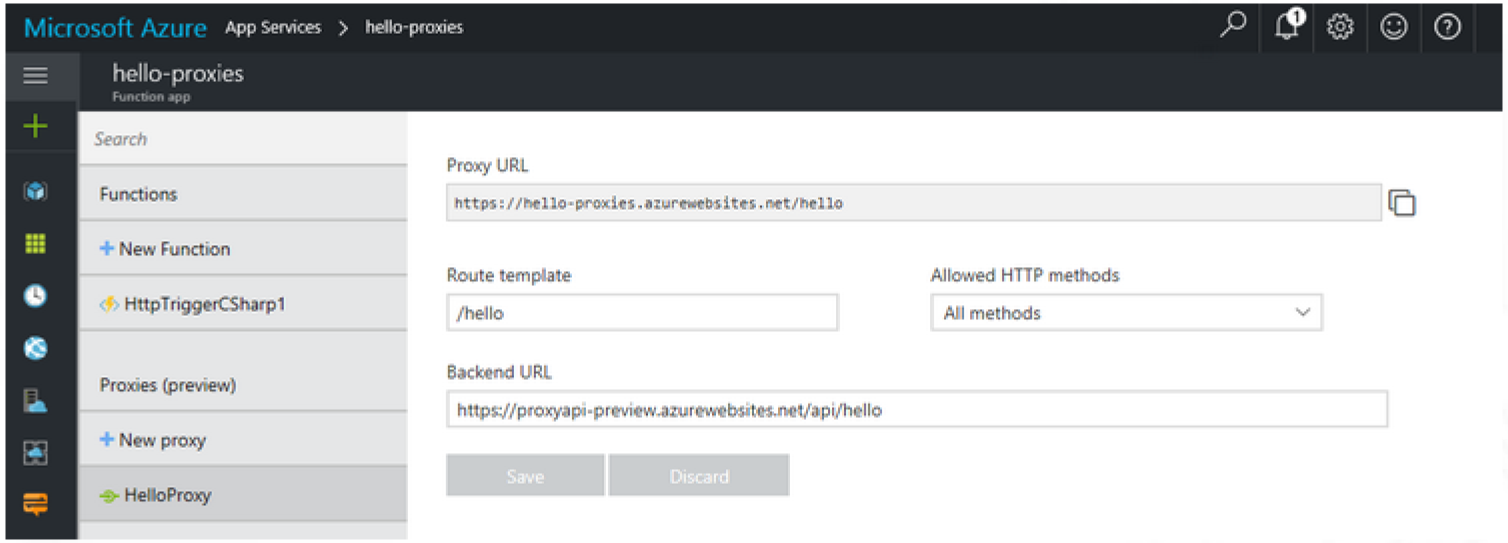


- Human Interaction



Azure Function Proxies

- Makes it easier to develop APIs by using Azure Functions
- Define a single API surface for multiple function apps
- Define an endpoint that serves as a reverse proxy to another API



Proxy URL: https://hello-proxies.azurewebsites.net/hello

Route template: /hello

Allowed HTTP methods: All methods

Backend URL: https://proxyapi-preview.azurewebsites.net/api/hello

Summary

- Serverless refers to abstraction of servers, infrastructure, and operating systems
- Azure Functions is a serverless compute service that can run code on-demand
- Rich toolkit available for Visual Studio and Visual Studio Code
- Triggers provide out-of-the-box templates to execute Azure functions
- Bindings provide the input and output payloads in a declarative way
- Two hosting plans supported – App Service plan and Consumption plan
- Azure Functions supports continuous deployment through a variety of source controls
- Azure Functions have built-in support for Application Insights for monitoring
- Upcoming features – Durable Functions and Function Proxies

References/Further Reading

- https://docs.microsoft.com/en-us/azure/azure-functions/
- https://azure.microsoft.com/en-us/pricing/details/functions/
- https://azure.microsoft.com/en-us/updates/?query=functions
- https://github.com/Azure/Azure-Functions
- https://docs.microsoft.com/en-us/sandbox/functions-recipes/

Questions…?

Thank You…!