Microsoft

Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS | San Diego

Building Awesome AF Apps

Rachel Appel
Rachel@rachelappel.com
http://rachelappel.com

Code Again for the First Time!

Visual Studio 25 YEARS OF CODING INNOVATION

What is Azure Functions?

CA(7

Code Azure Function Event + data

What is a Function?

- Function as the unit of work
- Functions are executed; they start and finish
- Functions have inputs and outputs

Slide 3

CA(7 Talk about dynamic compute + input/output bindings
Chris Anderson (ZUMO), 3/24/2016

Azure Functions: Open Source

- https://github.com/Azure/Azure-Functions
- https://docs.microsoft.com/en-us/azure/azure-functions/functions-overview

Function Examples

- Timer Based
- Transform CSV to Blob storage
- SaaS event processing. Excel to Graph API
- Web hook to create ad based on user profile
- Async image processing or map data processing
- Real time stream processing
- Real time bot messaging
- CRM System integration

Real World Scenarios

- Package tracking
- Vehicle tracking
- Data cleanup and ETL
- Batch processing
- IoT Solutions
  - snow depth monitor; football equipment monitor
- Internet traffic report aggregator

Function Apps vs API Apps

Function Apps

- Data Processing
- Microservice & serverless architecture
- Performs executable routine
- Does not have to be RESTful
- Service and software integration

API Apps

- CRUD operations
- API Architecture
- Manipulates or retrieves data
- RESTful
- Not generally for service and software integration

Serverless Computing

Run code, not computers

Serverless Computing

- What is serverless?
  - PaaS
- Stateless is scalable
- Complicated
- Sporadic workload
- Perform an action rather than return data
  - APIs return data
- Event driven

Serverless Code

- Microservices
- Variety of Languages
  - C#, F#, Node
- Event driven
- Expose HTTP Endpoints

Scenarios for serverless patterns

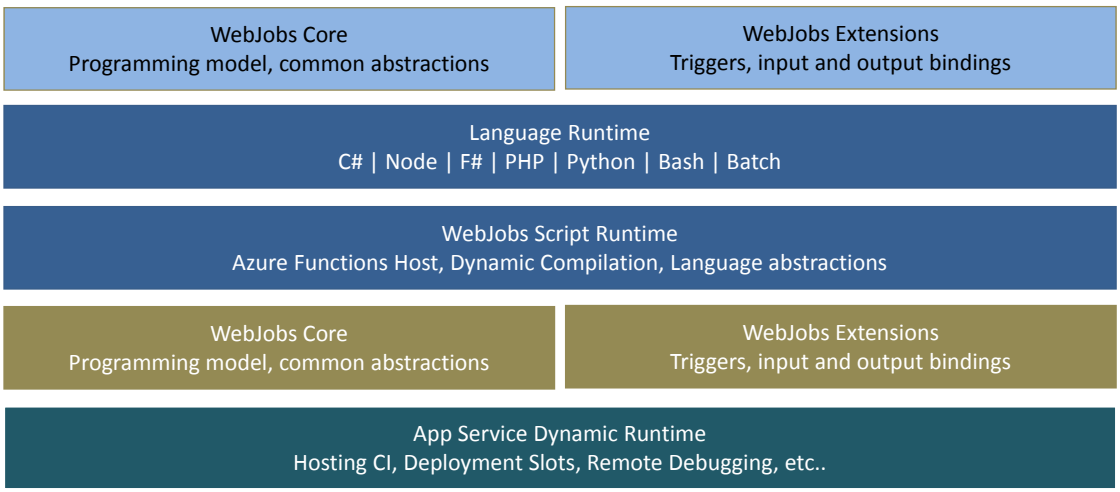
- Stateless and scale
- Too complicated for a traditional project structure
- Too simple for a traditional project structure
- Workload is sporadic (very low or high)
- Human involvement needs to stay low
- Lots of different services involved
- Integration of services or systems

Features & Benefits

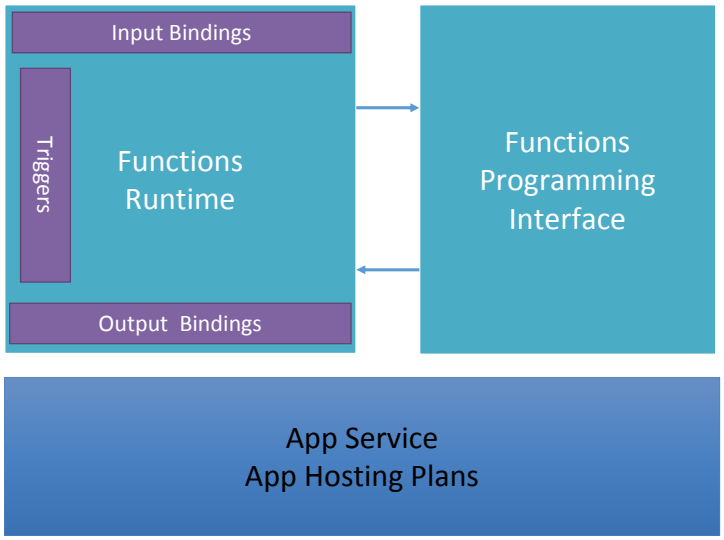
- Focus on business problems
- No worries about infrastructure
- No deployment
- Lightweight
- Cross-platform

Azure Functions Architecture

Functions PaaS Architecture



Azure Functions Architecture



Monolithic

Microservices

https://letsknoww.com/2017/05/20/monolithic-vs-microservices-architecture/

Programming Functions

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

Supported Languages

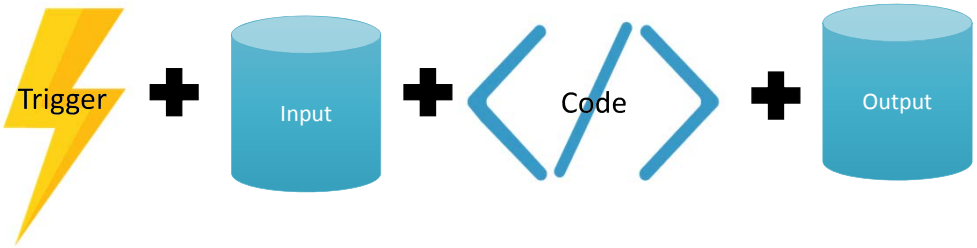
https://docs.microsoft.com/en-us/azure/azure-functions/supported-languages

Eperimental languages in version 1.x don't scale well and don't support all bindings.

Not for production use.

The version 2.x runtime doesn't support experimental languages.

Anatomy of a Function



A trigger causes a function to run

- Blob Trigger
- Event Hub Trigger
- Generic Webhook Trigger
- Github Webhook Trigger
- Http Trigger
- Manual Trigger
- Queue Trigger
- Service Bus Trigger
- Timer Trigger

Only one trigger per function allowed.

Bindings: Input and Output

- Access objects outside of your function from within it
  - Queues, tables, blobs, endpoints, etc…
- A function may have multiple input or output bindings
- Many bindings use Azure services or 3rd party services

Input bindings

- Azure Blob Storage
- External File
- External Table (e.g., SQL, MySQL,)
- Excel
- Azure CosmosDB

Output bindings

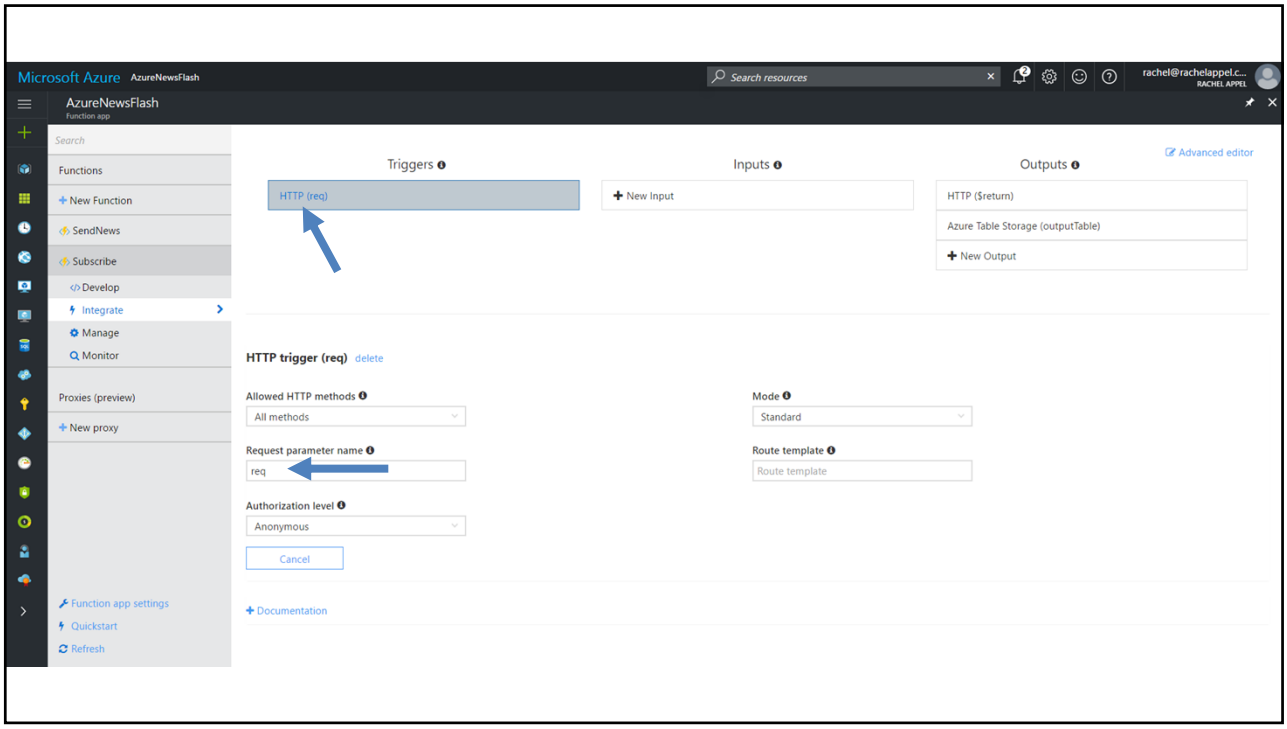
- Azure Event Hub
- Azure Queue Storage
- Azure Blob Storage
- External File (Preview)
- External Table (Experimental)
- HTTP
- Bot Framework
- Azure Service Bus
- Azure Table Storage
- Azure DocumentDB Document
- Azure Mobile Table Record
- Azure Notification Hub
- SendGrid (Preview)
- Twilio SMS (Preview)

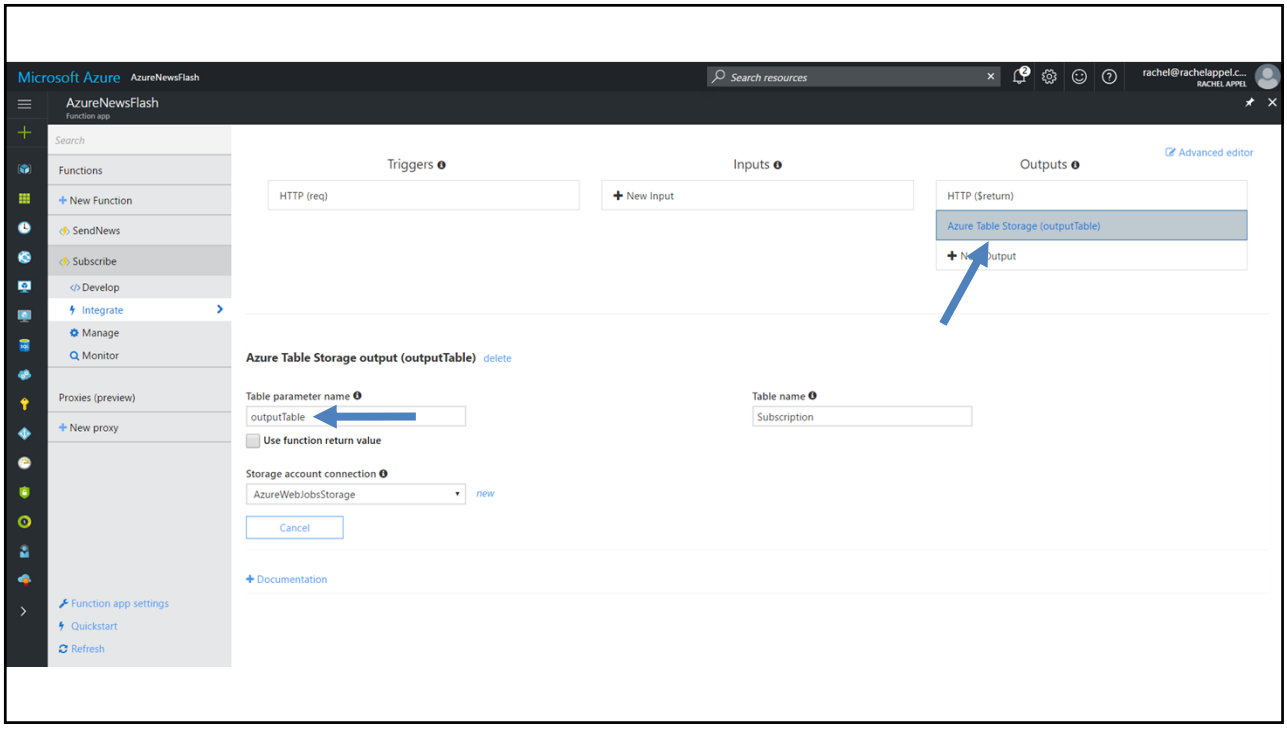
Output | Trigger | Input | Output

```
public static async Task<object> Run(HttpRequestMessage req, TraceWriter log, ICollector<Subscription> outputTable)
{
    log.Info($"Webhook was triggered!");

    // more code ...

    return req.CreateResponse(HttpStatusCode.OK, new {
        greeting = $"Hello {data.first} {data.last}!"
    });
}
```


Microsoft Azure
AzureNewsFlash
Search resources
rachel@rachelappel.c...
RACHEL APPEL
AzureNewsFlash
Function app
Search
Functions
New Function
SendNews
Subscribe
Develop
Integrate
Manage
Monitor
Proxies (preview)
New proxy
Function app settings
Quickstart
Refresh
Triggers
Inputs
Outputs
Advanced editor
HTTP (req)
New Input
HTTP ($return)
Azure Table Storage (outputTable)
New Output
HTTP trigger (req)
delete
Allowed HTTP methods
All methods
Mode
Standard
Request parameter name
req
Route template
Route template
Authorization level
Anonymous
Cancel
Documentation


Microsoft Azure
AzureNewsFlash
Search resources
rachel@rachelappel.c...
RACHEL APPEL
AzureNewsFlash
Function app
Search
Functions
New Function
SendNews
Subscribe
Develop
Integrate
Manage
Monitor
Proxies (preview)
New proxy
Function app settings
Quickstart
Refresh
Triggers
Inputs
Outputs
Advanced editor
HTTP (req)
New Input
HTTP ($return)
Azure Table Storage (outputTable)
Azure Table Storage output (outputTable)
delete
Table parameter name
outputTable
Table name
Subscription
Use function return value
Storage account connection
AzureWebJobsStorage
new
Cancel
Documentation

Bindings in Depth

- Timer Trigger
- HTTP Request/Webhook
- Azure Storage Table
- Blob Trigger
- Queue Trigger

Timer Triggers

0	*/10	*	*	*	*

- Seconds
- Minutes
- Hours
- Day of Month
- Month
- Day of week

"/" helps produce step values
"*" matches all values
"0" matches only 0

HTTP & Webhook bindings

```
{
  "bindings": [
    {
      "authLevel": "function",
      "name": "req",
      "type": "httpTrigger",
      "direction": "in",
      "methods": [
        "get",
        "post"
      ]
    }
  ],
  "disabled": false
}
```

```
{
  "bindings": [
    {
      "type": "httpTrigger",
      "direction": "in",
      "webHookType": "genericJson",
      "name": "req",
      "methods": [
        "post"
      ]
    }
  ],
  "disabled": false
}
```

HTTP & Webhook bindings

```
{
  "type": "http",
  "name": "res",
  "direction": "out"
}
```

Advanced Programming Techniques

Calling Other Functions

- Use an output trigger followed by that same trigger, but as an input trigger to the next function to trigger
- Must be inside same Function App

Reusing .csx code

#load “file.csx”

load classes, or functions

You can use a relative path with the #load directive:

#load “file.csx" loads a file located in the function folder.

#load “shared\file.csx" loads a file located in the shared folder in the function folder.

#load "..\shared\folder.csx" loads a file located in a folder at the same level as the function folder, that is, directly under wwwroot.

Imperative Binding

- https://docs.microsoft.com/en-us/azure/azure-functions/functions-triggers-bindings#advanced-binding-at-runtime-imperative-binding

Environment Variables

To get an environment variable or an app setting value, use System.Environment.GetEnvironmentVariable, as shown in the following code example:+

Copy

C#

```
public static void Run(TimerInfo myTimer, TraceWriter log)
{
   log.Info($"C# Timer trigger function executed at: {DateTime.Now}");
   log.Info(GetEnvironmentVariable("AzureWebJobsStorage"));
   log.Info(GetEnvironmentVariable("WEBSITE_SITE_NAME"));
}

public static string GetEnvironmentVariable(string name)
{
   return name + ": " +
      System.Environment.GetEnvironmentVariable(name, EnvironmentVariableTarget.Process);
}
```

Tools

Tools

Azure CLI

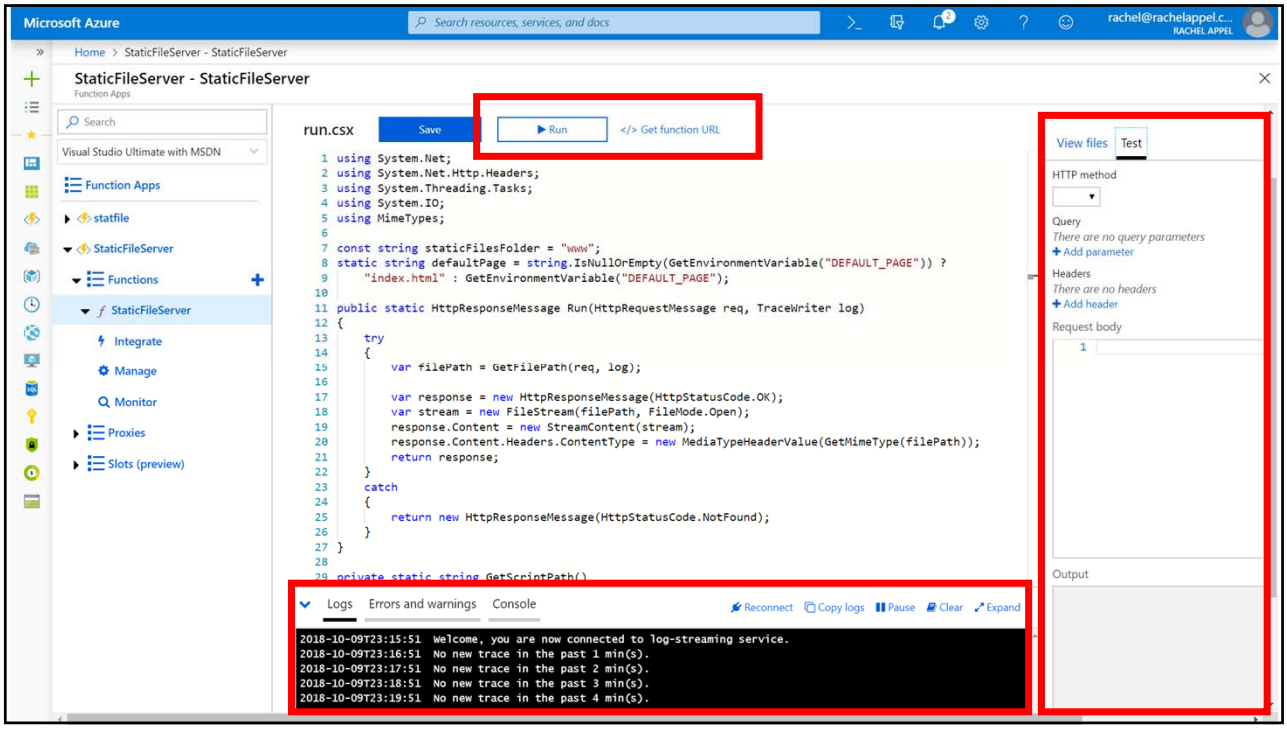
Visual Studio Cloud Explorer
-and-
Visual Studio Tools for Azure Functions

Visual Studio **Code** Azure-functions extension

Microsoft Azure Storage Explorer

Debugging Function Apps

- Postman or DevTools
- Test/Run in cloud
- Visual Studio



Scaling & Best Practices

Managing Workloads/Scaling

- Keep functions idempotent and stateless
- Async is best but avoid Task.Result
- Avoid long running functions
- Queues are best for cross function communication
- Code in exception management

Best Practices

- Small, fast-running functions
- Asynchronous > Synchronous
- Caching and singletons (memory is shared between functions)
- Avoid disk operations (shared across functions)
- Use App Service guidelines

Settings & Deployment

Deployment

- Slots (Preview)
- Github
- Functions are an App Service
- Continuous Integration
- Download and setup in Github locally, then push

Azure function app class library

- https://blogs.msdn.microsoft.com/appserviceteam/2017/03/16/publishing-a-net-class-library-as-a-function-app/

Microsoft

Questions?

Rachel Appel
Sr Content Developer for Azure
Microsoft
rachelap@microsoft.com
http://rachelappel.com