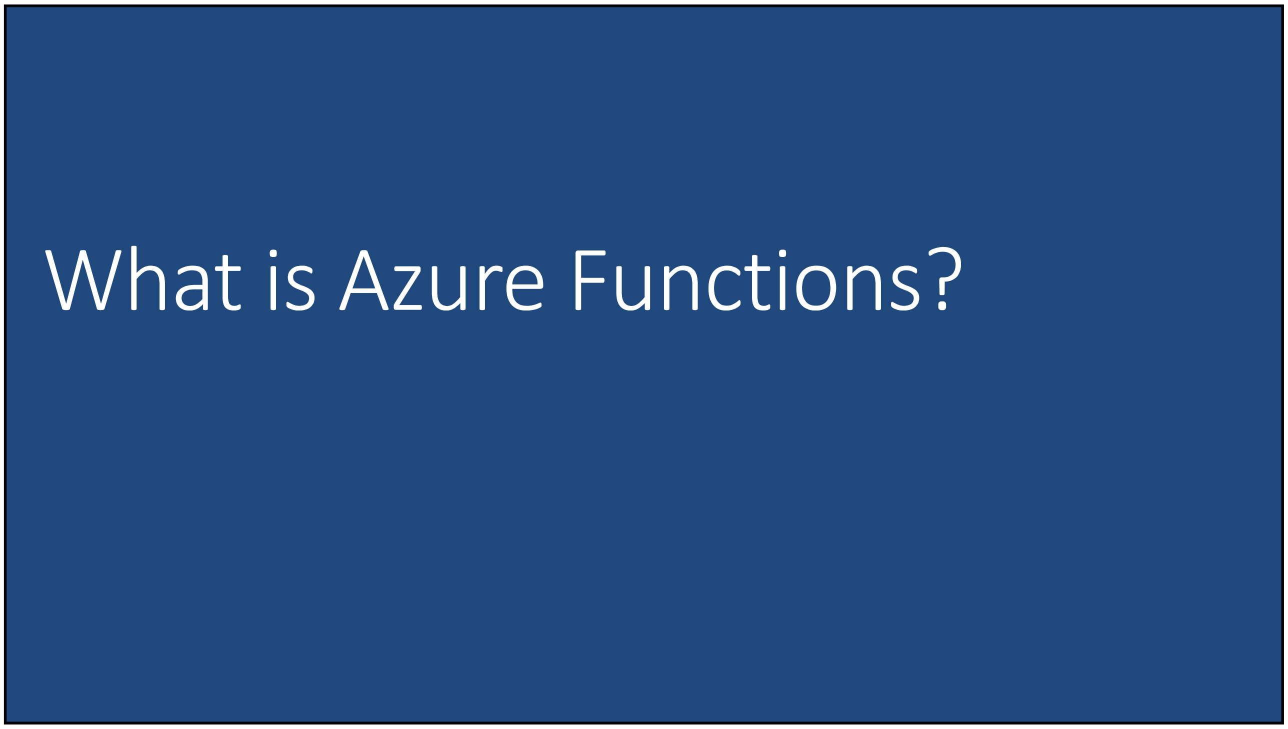
Visual Studio LIVE! EXPERT SOLUTIONS FOR .NET DEVELOPERS

ROCK YOUR CODE TOUR 2017 | CHICAGO

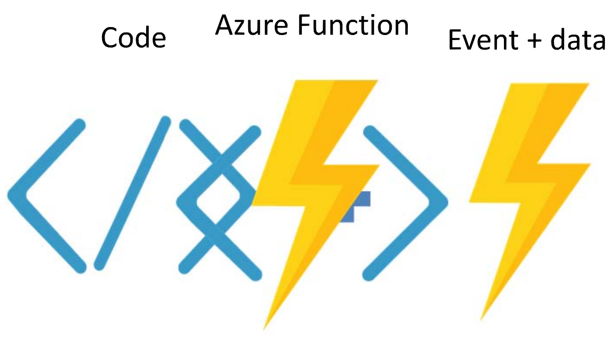
Building Awesome AF Apps

Rachel Appel
Microsoft



What is Azure Functions?

CA(7

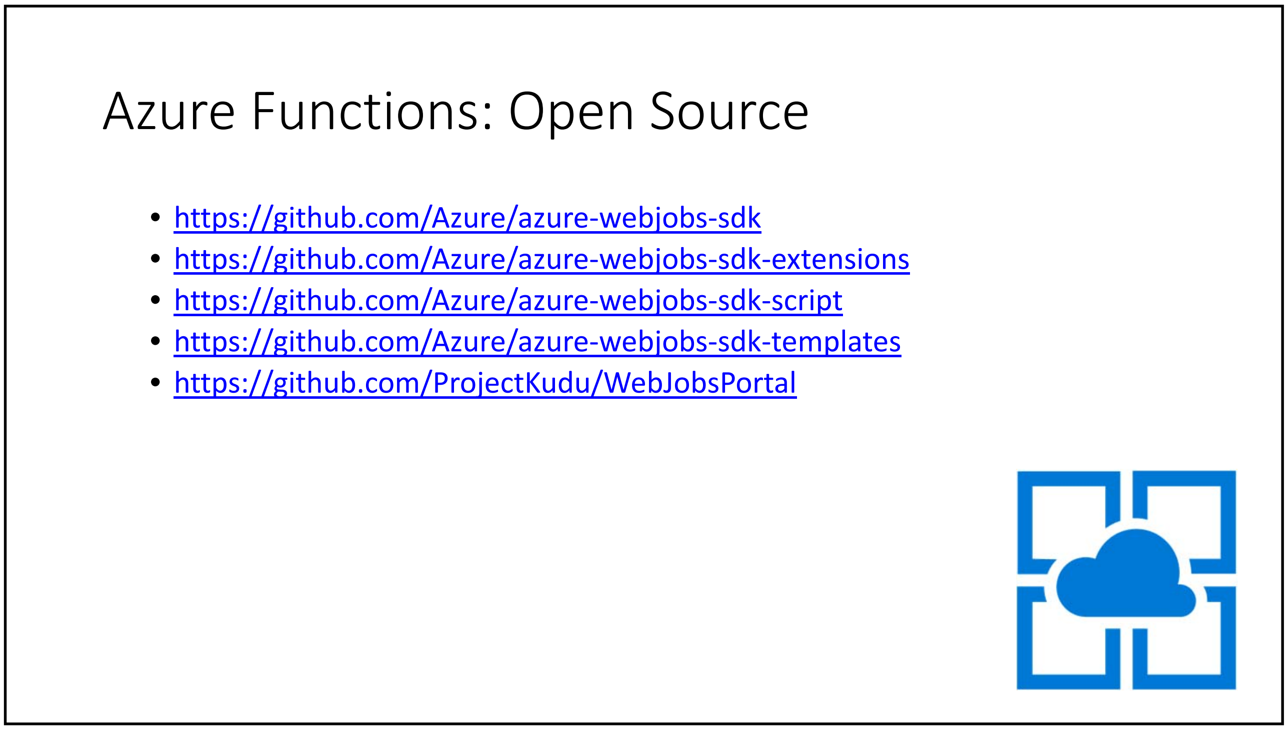


What is a Function?

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

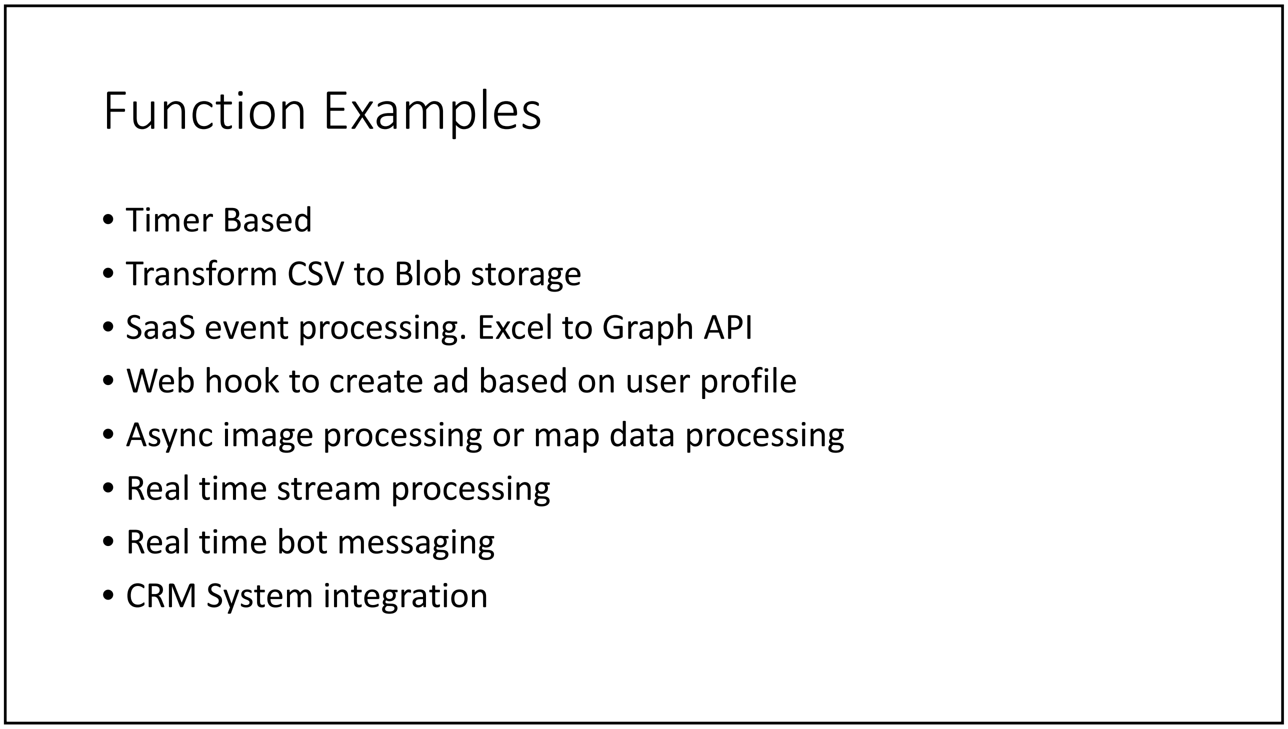
Slide 3

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



Azure Functions: Open Source

- https://github.com/Azure/azure-webjobs-sdk
- https://github.com/Azure/azure-webjobs-sdk-extensions
- https://github.com/Azure/azure-webjobs-sdk-script
- https://github.com/Azure/azure-webjobs-sdk-templates
- https://github.com/ProjectKudu/WebJobsPortal



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

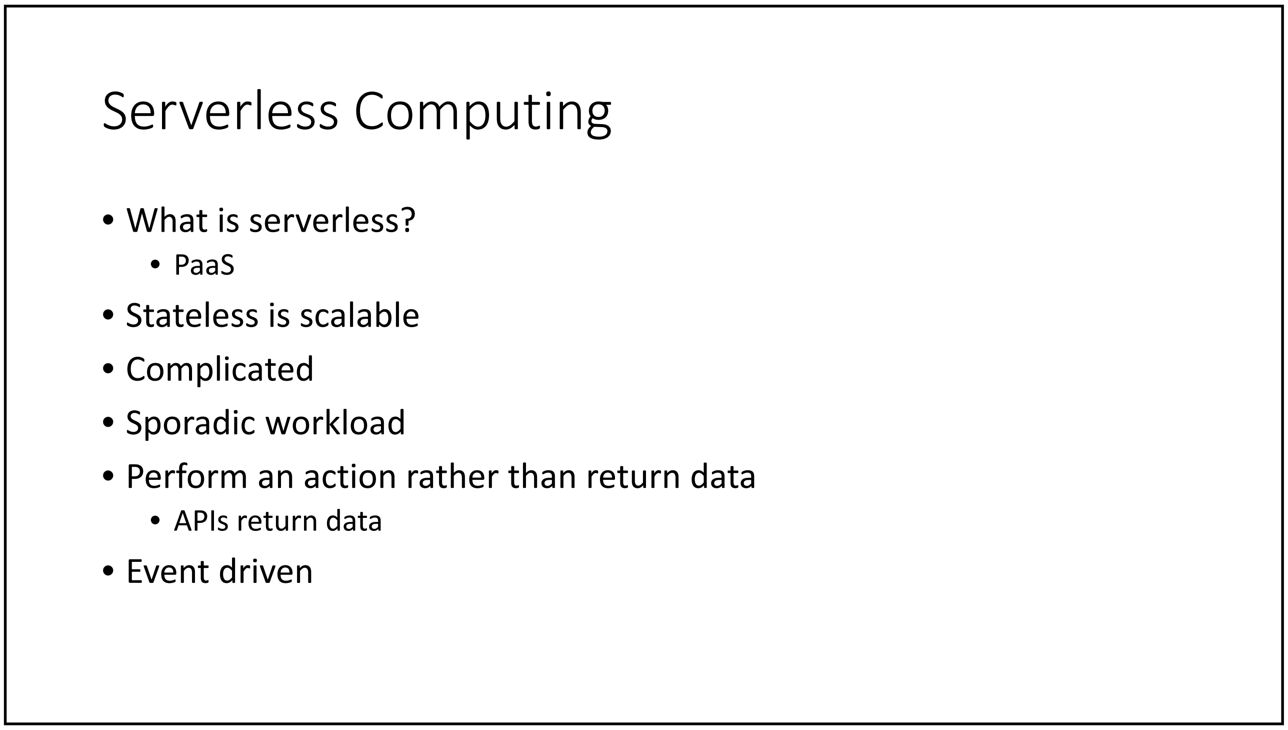
- Data Processing
- Microservice & serverless architecture
- Performs executable routine
- Does not have to be RESTful
- Service and software integration

API Apps

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

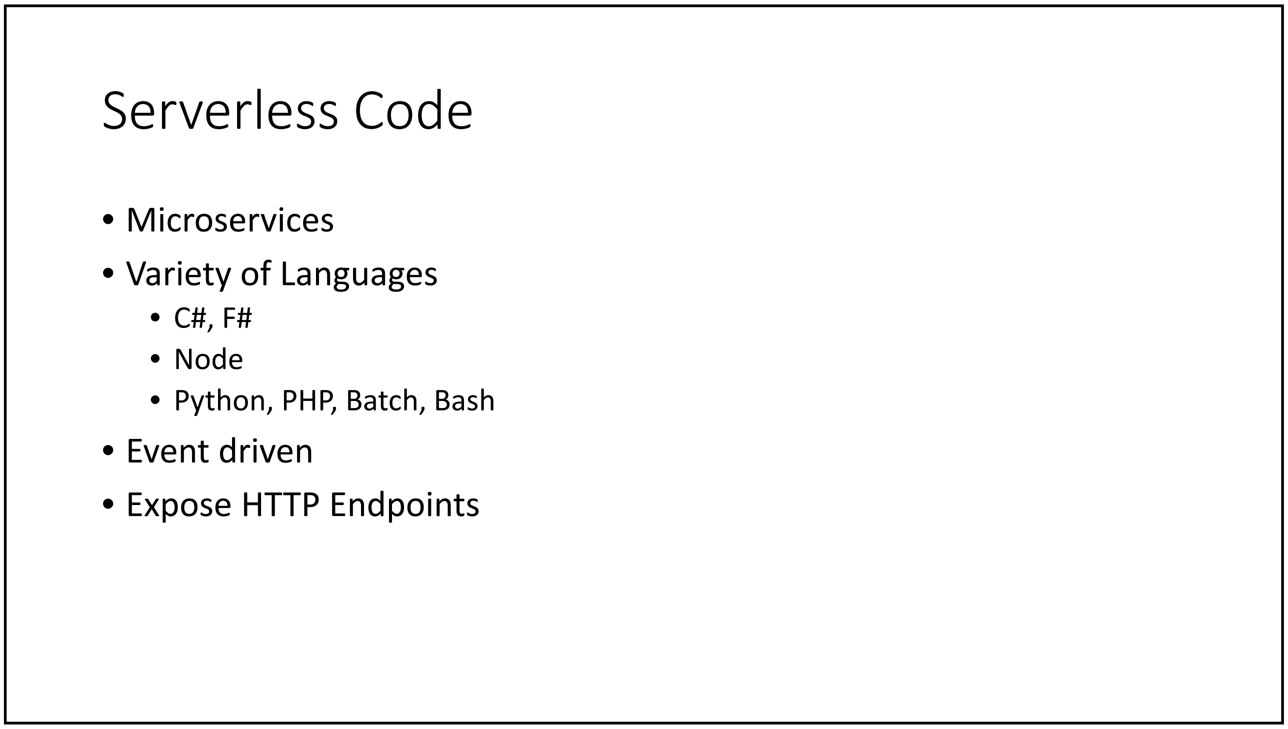
Serverless Computing

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
  - Python, PHP, Batch, Bash
- Event driven
- Expose HTTP Endpoints

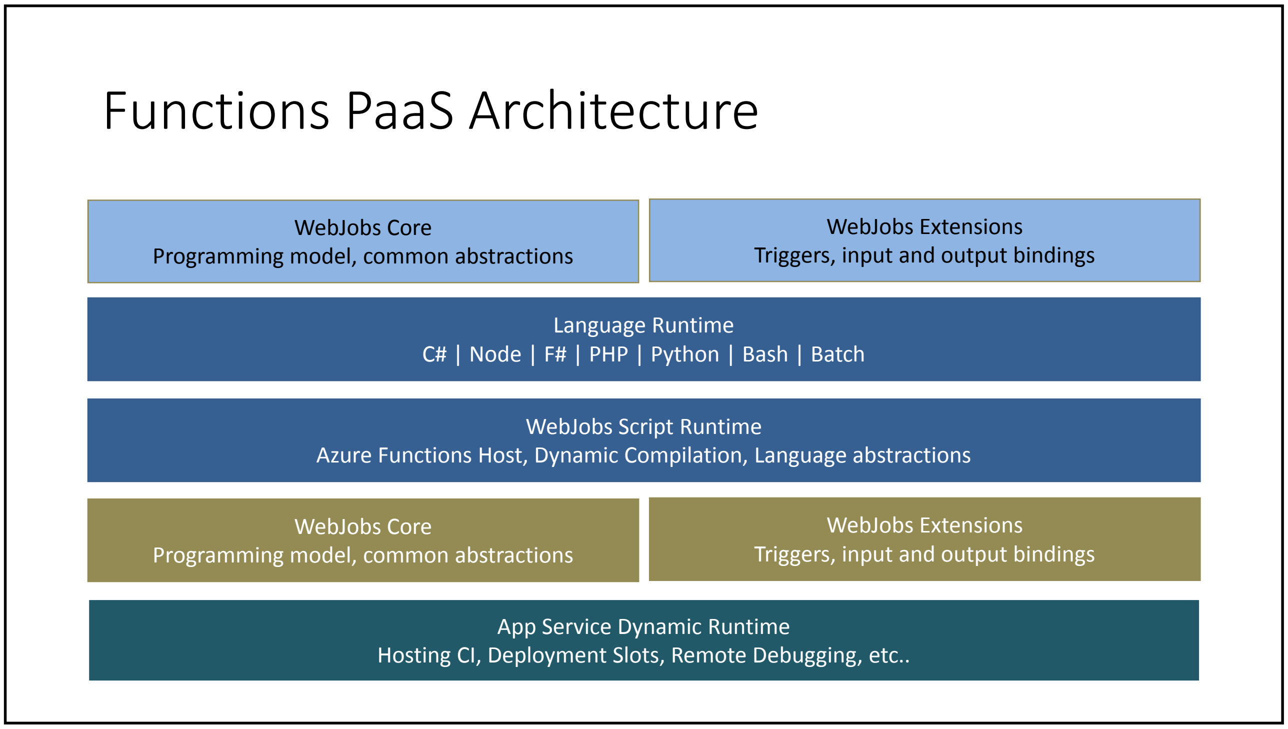
Scenarios for serverless patterns

- 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

WebJobs Core
Programming model, common abstractions

WebJobs Extensions
Triggers, input and output bindings

Language Runtime
C# | Node | F# | PHP | Python | Bash | Batch

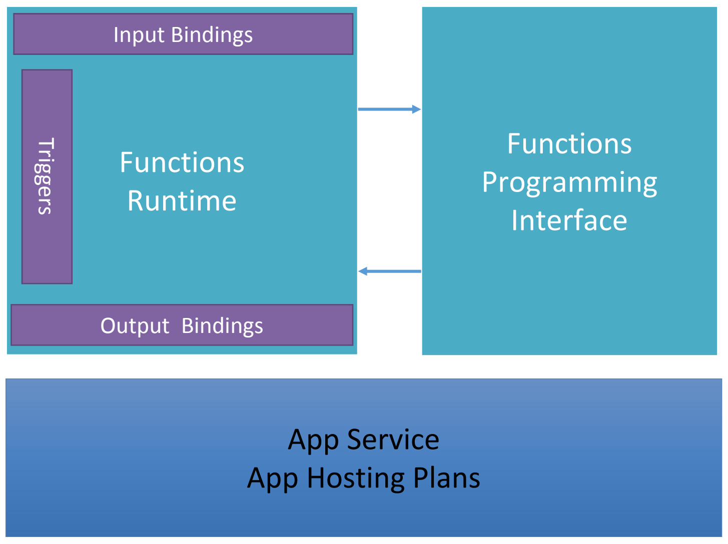
WebJobs Script Runtime
Azure Functions Host, Dynamic Compilation, Language abstractions

WebJobs Core
Programming model, common abstractions

WebJobs Extensions
Triggers, input and output bindings

App Service Dynamic Runtime
Hosting CI, Deployment Slots, Remote Debugging, etc..

Azure Functions Architecture

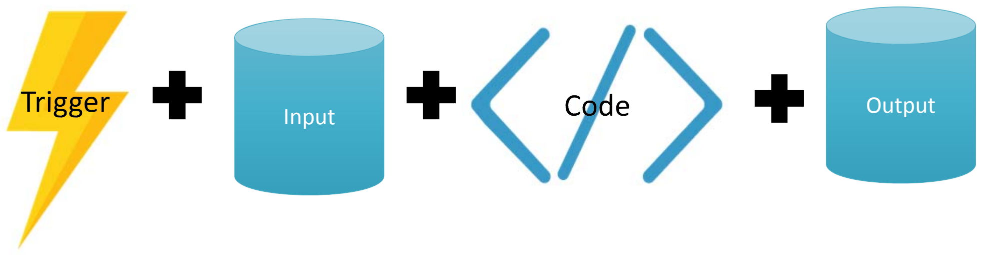


Monolithic

Microservices

Programming Functions

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

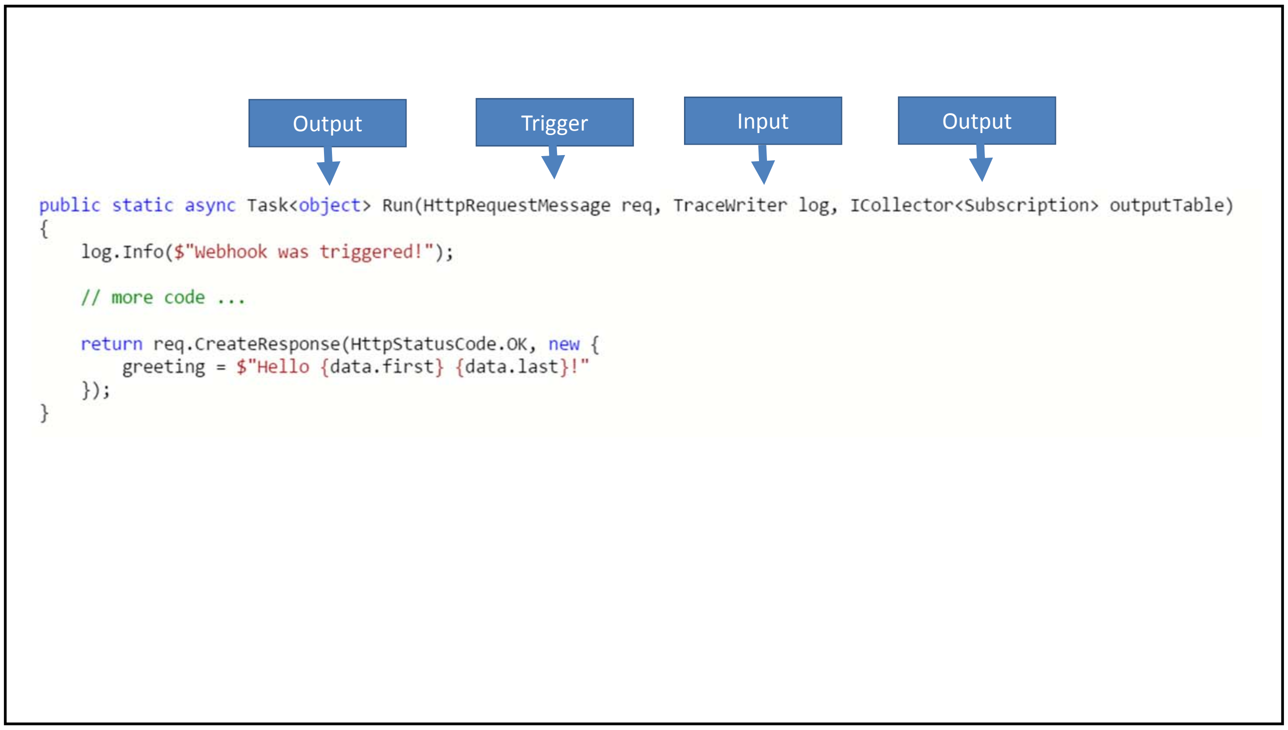
- Access objects outside of your function from within it
  - Queues, tables, blobs, endpoints, etc…
- A function may have multiple input or output bindings
- Many bindings use Azure services or 3rd party services

Input bindings

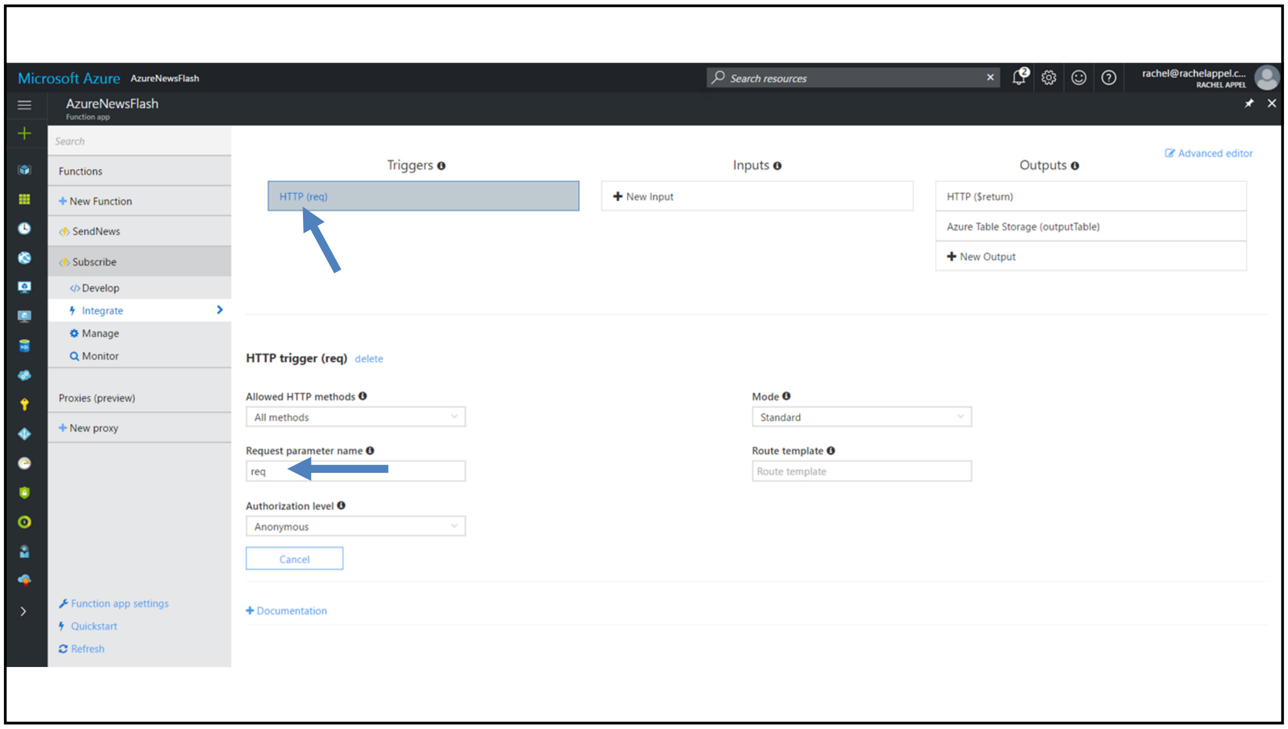
- Azure Blob Storage
- External File (Preview)
- External Table (Experimental)
- Azure Storage Table
- Azure DocumentDB Document
- Azure Mobile Table Record
- Bot Framework

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)



```
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}!"
    });
}
```

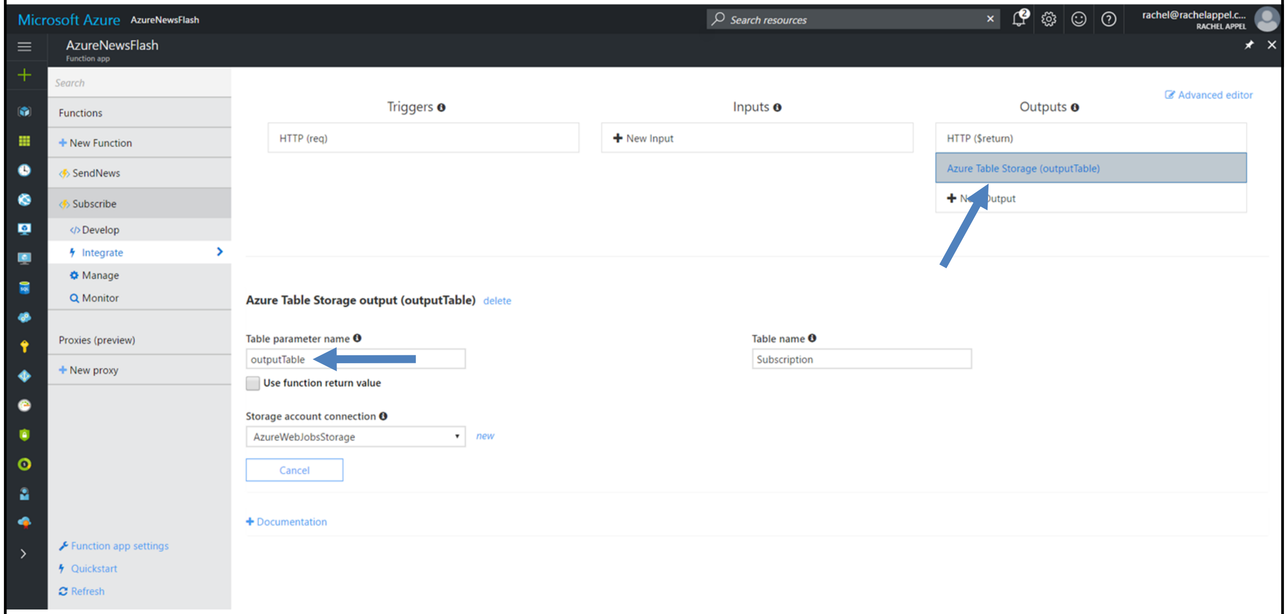


Triggers: HTTP (req)

Outputs: HTTP ($return); Azure Table Storage (outputTable)

HTTP trigger (req) delete

Allowed HTTP methods	All methods
Mode	Standard
Request parameter name	req
Route template	Route template
Authorization level	Anonymous



Triggers	Inputs	Outputs
HTTP (req)	+ New Input	HTTP ($return)
		Azure Table Storage (outputTable)
		+ New Output

Azure Table Storage output (outputTable)

Table parameter name: outputTable

Table name: Subscription

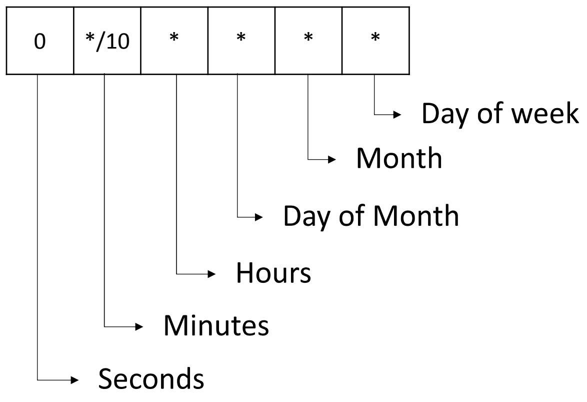
Use function return value: ☐

Storage account connection: AzureWebJobsStorage

Bindings in Depth

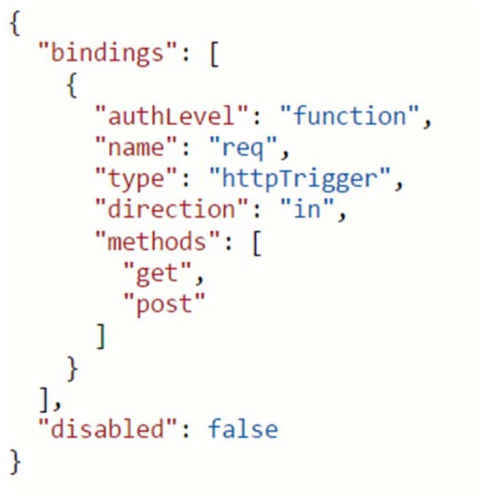
- Timer Trigger
- HTTP Request/Webhook
- Azure Storage Table
- Blob Trigger
- Queue Trigger

Timer Triggers

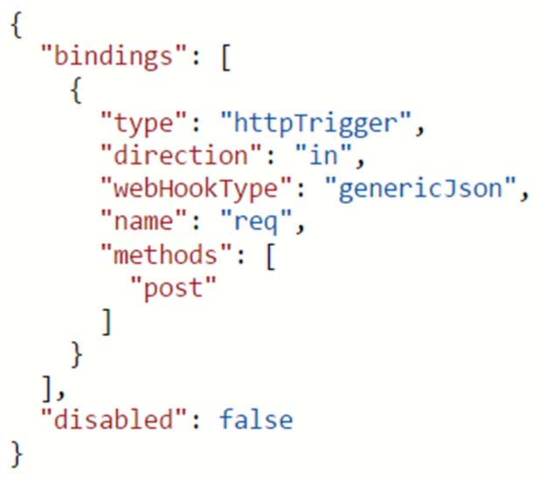


"/" helps produce step values
"*" matches all values
"0" matches only 0

HTTP & Webhook bindings

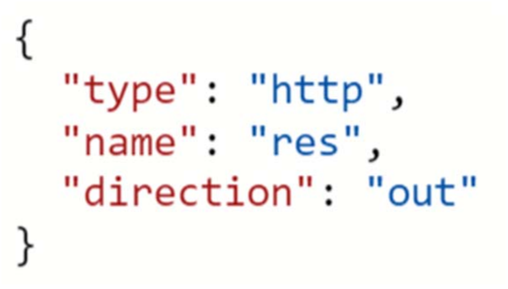


```
{
  "bindings": [
    {
      "authLevel": "function",
      "name": "req",
      "type": "httpTrigger",
      "direction": "in",
      "methods": [
        "get",
        "post"
      ]
    }
  ],
  "disabled": false
}
```



```
{
  "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

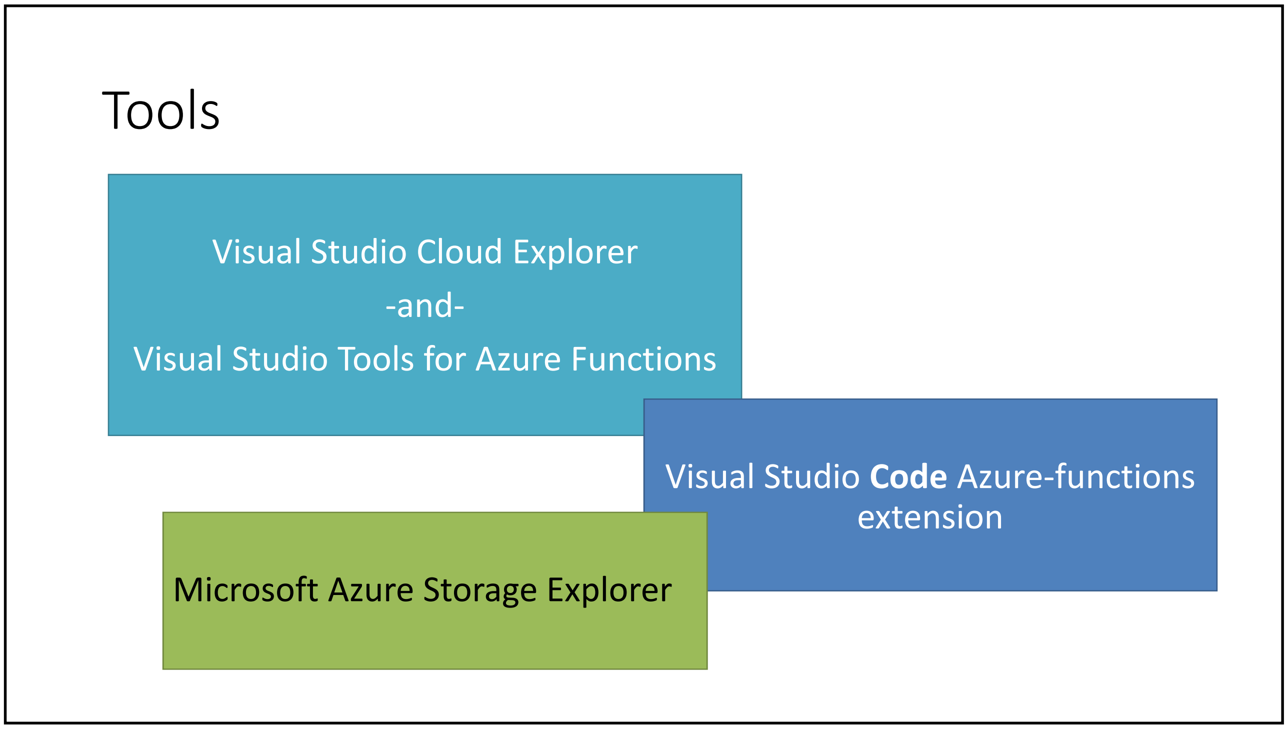
To get an environment variable or an app setting value, use System.Environment.GetEnvironmentVariable, as shown in the following code example:+

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

Visual Studio Cloud Explorer

-and-

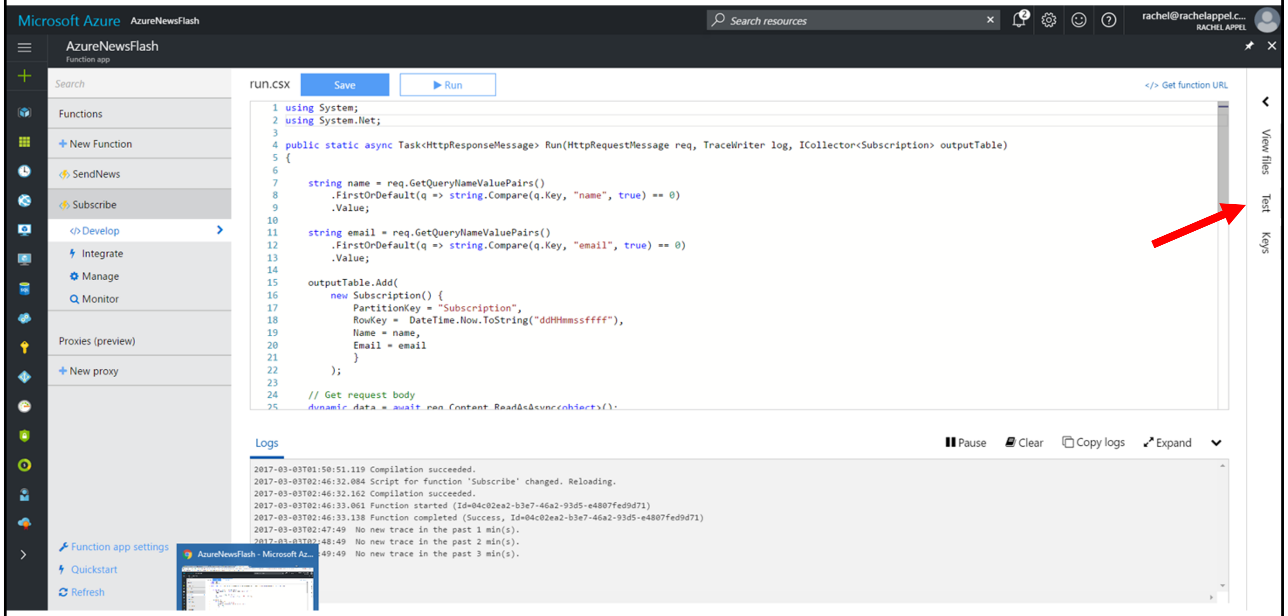
Visual Studio Tools for Azure Functions

Visual Studio **Code** Azure-functions extension

Microsoft Azure Storage Explorer

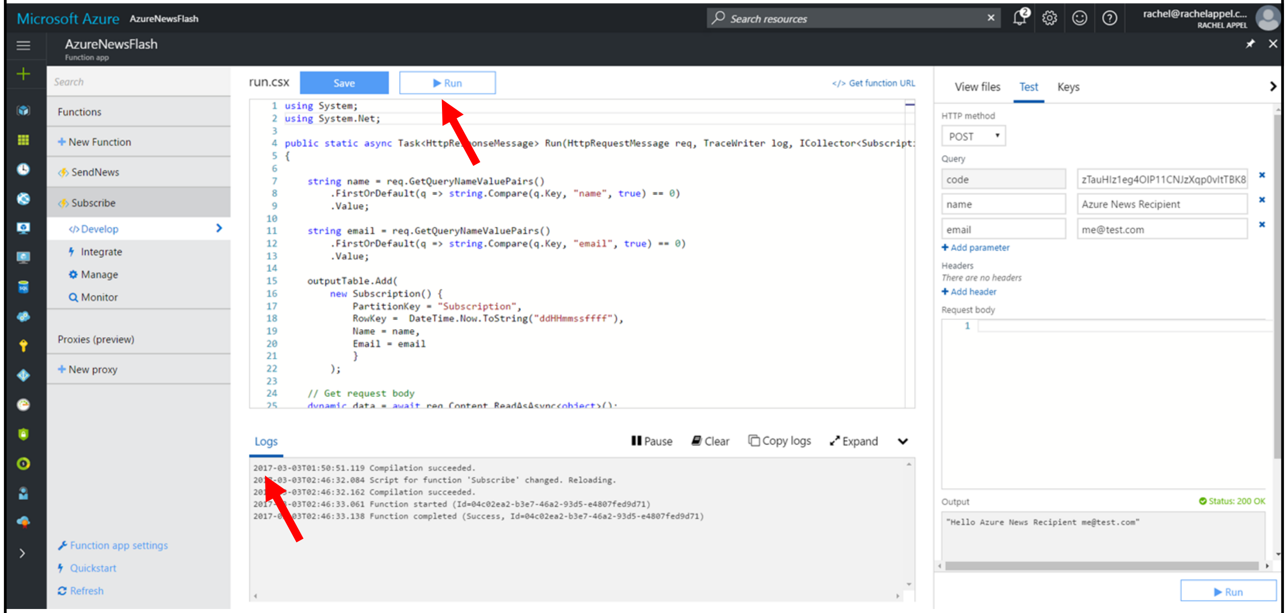
Debugging Function Apps

- Postman
- Test/Run in cloud
- Visual Studio



```
using System;
using System.Net;

public static async Task<HttpResponseMessage> Run(HttpRequestMessage req, TraceWriter log, ICollector<Subscription> outputTable)
{

    string name = req.GetQueryNameValuePairs()
        .FirstOrDefault(q => string.Compare(q.Key, "name", true) == 0)
        .Value;

    string email = req.GetQueryNameValuePairs()
        .FirstOrDefault(q => string.Compare(q.Key, "email", true) == 0)
        .Value;

    outputTable.Add(
        new Subscription() {
            PartitionKey = "Subscription",
            RowKey =  DateTime.Now.ToString("ddHHmmssffff"),
            Name = name,
            Email = email
            }
        );

    // Get request body
    dynamic data = await req.Content.ReadAsAsync<object>();
```

```
2017-03-03T01:50:51.119 Compilation succeeded.
2017-03-03T02:46:32.084 Script for function 'Subscribe' changed. Reloading.
2017-03-03T02:46:32.162 Compilation succeeded.
2017-03-03T02:46:33.061 Function started (Id=04c02ea2-b3e7-46a2-93d5-e4807fed9d71)
2017-03-03T02:46:33.138 Function completed (Success, Id=04c02ea2-b3e7-46a2-93d5-e4807fed9d71)
2017-03-03T02:47:49  No new trace in the past 1 min(s).
2017-03-03T02:48:49  No new trace in the past 2 min(s).
:49:49  No new trace in the past 3 min(s).
```



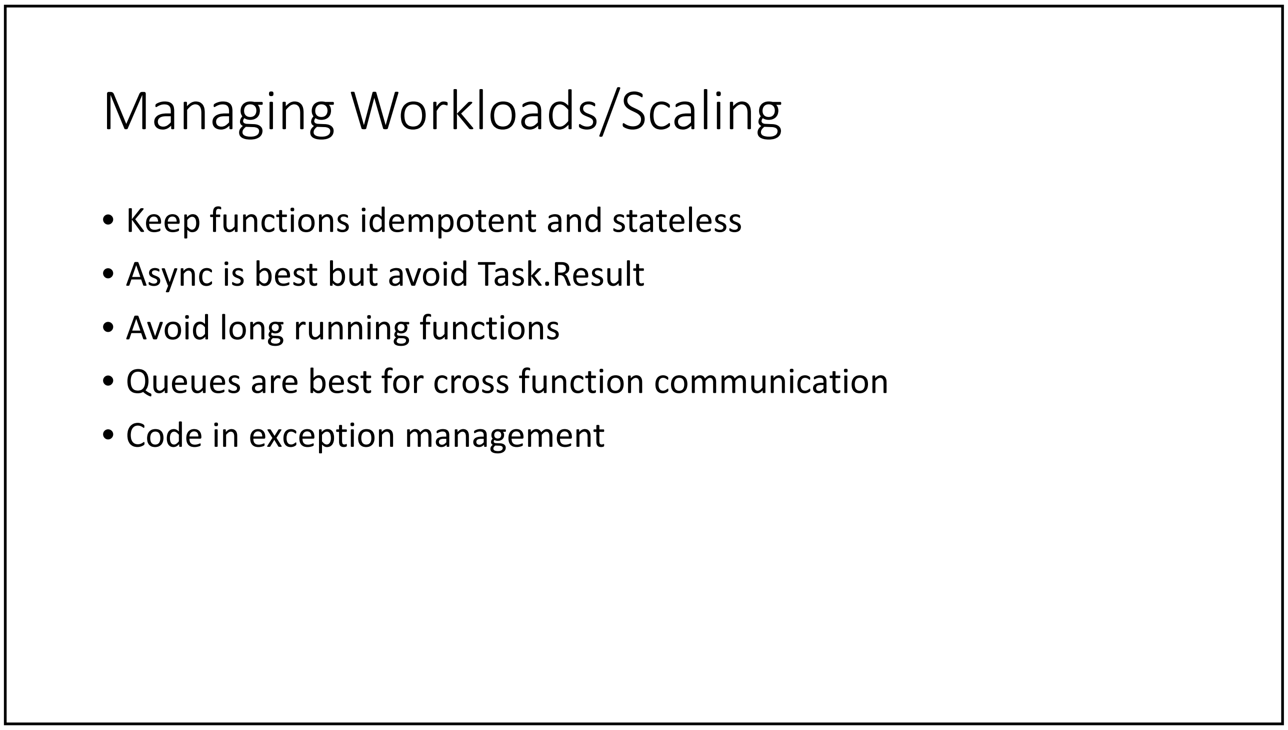
```
using System;
using System.Net;

public static async Task<HttpResponseMessage> Run(HttpRequestMessage req, TraceWriter log, ICollector<Subscript

    string name = req.GetQueryNameValuePairs()
        .FirstOrDefault(q => string.Compare(q.Key, "name", true) == 0)
        .Value;

    string email = req.GetQueryNameValuePairs()
        .FirstOrDefault(q => string.Compare(q.Key, "email", true) == 0)
        .Value;

    outputTable.Add(
        new Subscription() {
            PartitionKey = "Subscription",
            RowKey =  DateTime.Now.ToString("ddHHmmssffff"),
            Name = name,
            Email = email
            }
        );

    // Get request body
    dynamic data = await req.Content.ReadAsAsync<object>();
```

```
2017-03-03T01:50:51.119 Compilation succeeded.
2017-03-03T02:46:32.084 Script for function 'Subscribe' changed. Reloading.
2017-03-03T02:46:32.162 Compilation succeeded.
2017-03-03T02:46:33.061 Function started (Id=04c02ea2-b3e7-46a2-93d5-e4807fed9d71)
2017-03-03T02:46:33.138 Function completed (Success, Id=04c02ea2-b3e7-46a2-93d5-e4807fed9d71)
```

code	zTauHlz1eg4OIP11CNJzXqp0vltTBK8
name	Azure News Recipient
email	me@test.com

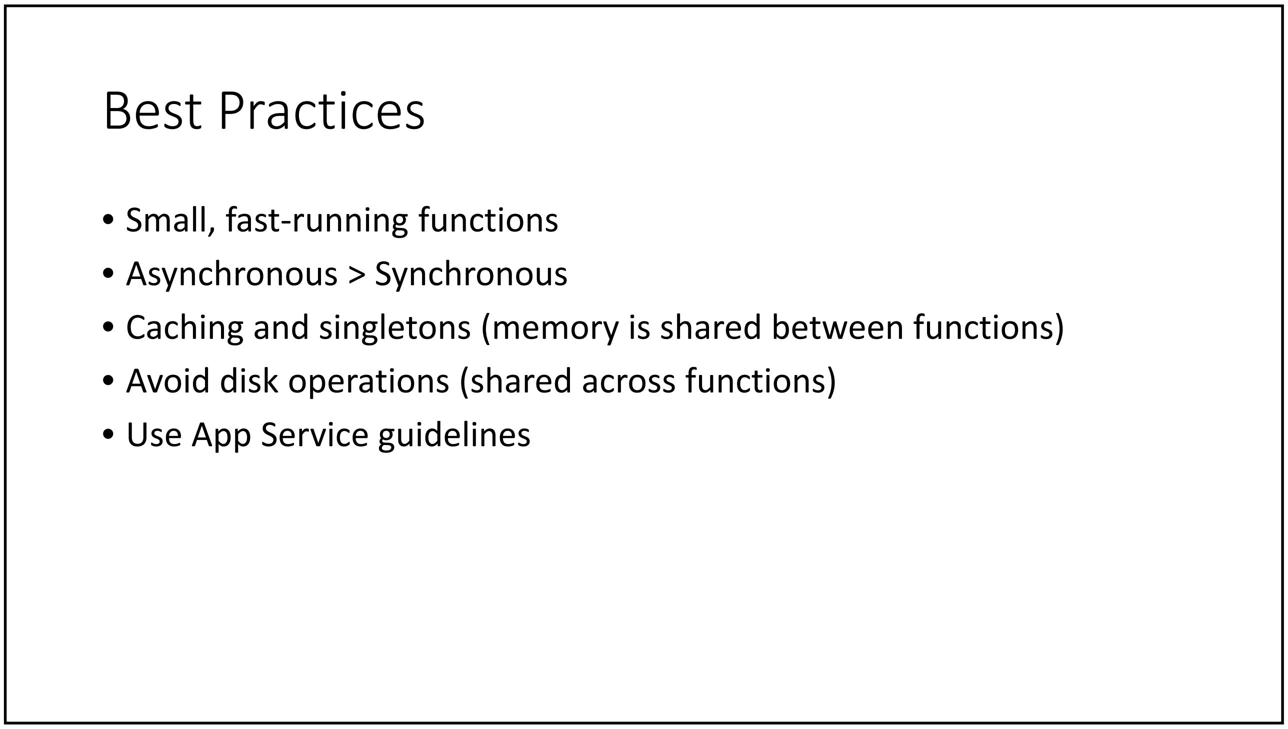
```
"Hello Azure News Recipient me@test.com"
```

Scaling & Best Practices



Managing Workloads/Scaling

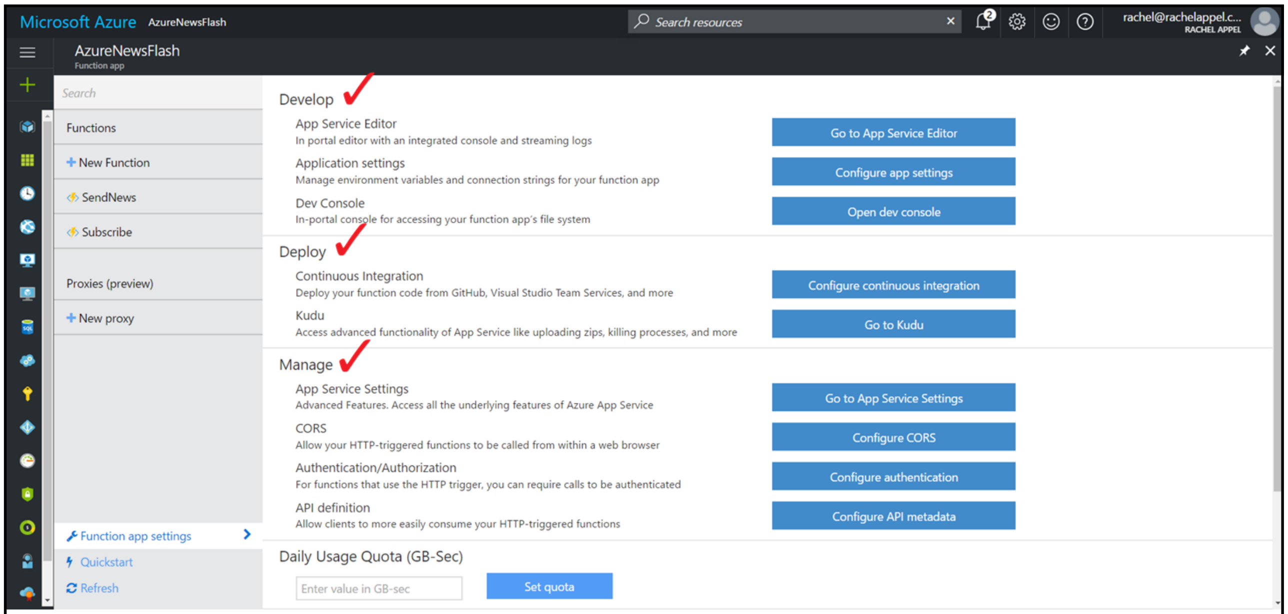
- Keep functions idempotent and stateless
- Async is best but avoid Task.Result
- Avoid long running functions
- Queues are best for cross function communication
- Code in exception management



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



Microsoft Azure AzureNewsFlash

AzureNewsFlash
Function app

Functions
+ New Function
SendNews
Subscribe
Proxies (preview)
+ New proxy
Function app settings
Quickstart
Refresh

Develop ✓
- App Service Editor — In portal editor with an integrated console and streaming logs — Go to App Service Editor
- Application settings — Manage environment variables and connection strings for your function app — Configure app settings
- Dev Console — In-portal console for accessing your function app's file system — Open dev console

Deploy ✓
- Continuous Integration — Deploy your function code from GitHub, Visual Studio Team Services, and more — Configure continuous integration
- Kudu — Access advanced functionality of App Service like uploading zips, killing processes, and more — Go to Kudu

Manage ✓
- App Service Settings — Advanced Features. Access all the underlying features of Azure App Service — Go to App Service Settings
- CORS — Allow your HTTP-triggered functions to be called from within a web browser — Configure CORS
- Authentication/Authorization — For functions that use the HTTP trigger, you can require calls to be authenticated — Configure authentication
- API definition — Allow clients to more easily consume your HTTP-triggered functions — Configure API metadata

Daily Usage Quota (GB-Sec)
Enter value in GB-sec — Set quota

Deployment

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

- 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