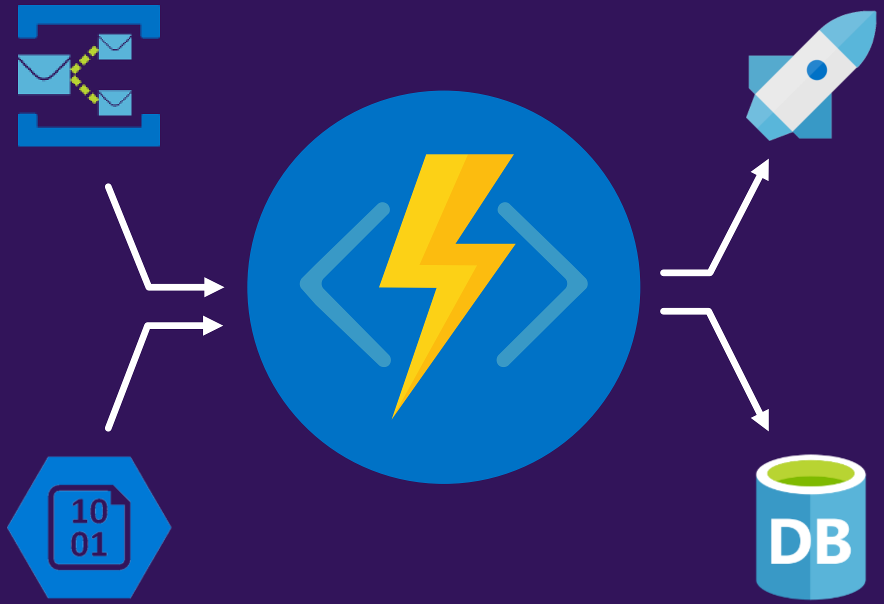
Serverless Compute with Azure Functions



Ben Coleman
Cloud Solution Architect

@BenCodeGeek

Microsoft

Serverless Computing

Serverless computing is an event-driven application design and deployment paradigm in which computing resources are provided as scalable cloud services.

In a serverless computing deployment, the cloud customer only pays for service usage; there is never any cost associated with idle time.

Source - http://searchitoperations.techtarget.com/definition/serverless-computing

Serverless application platform components

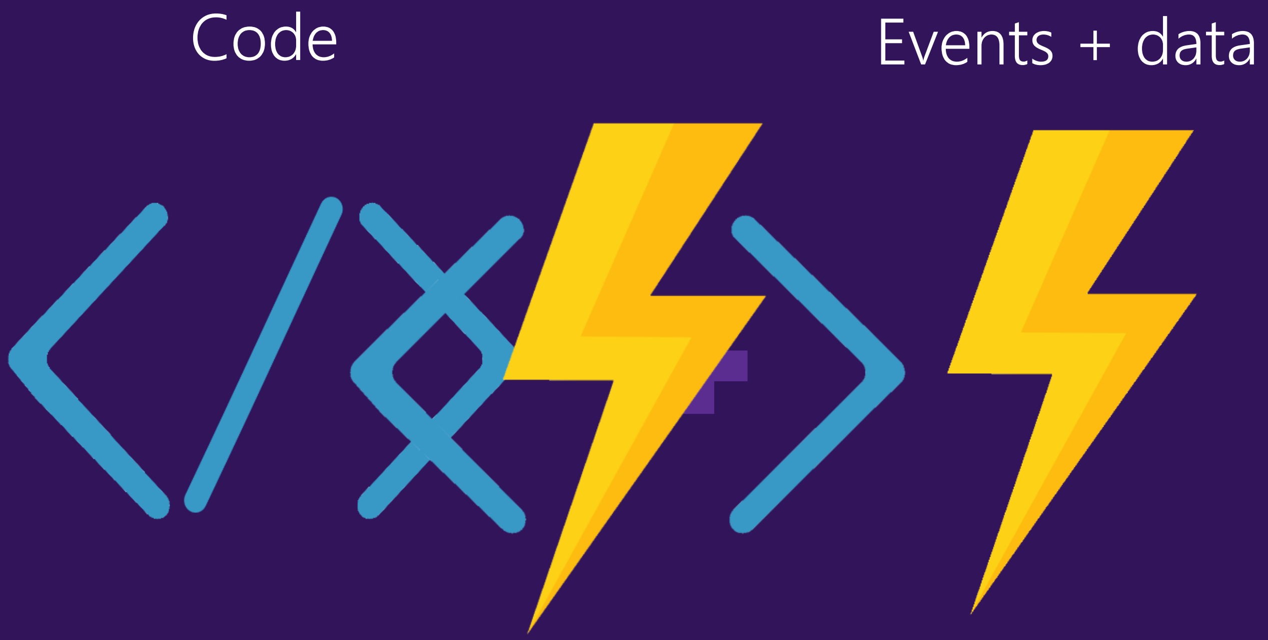
Development

- IDE support
- Integrated DevOps
- Local Development
- Monitoring
- Visual Debug History

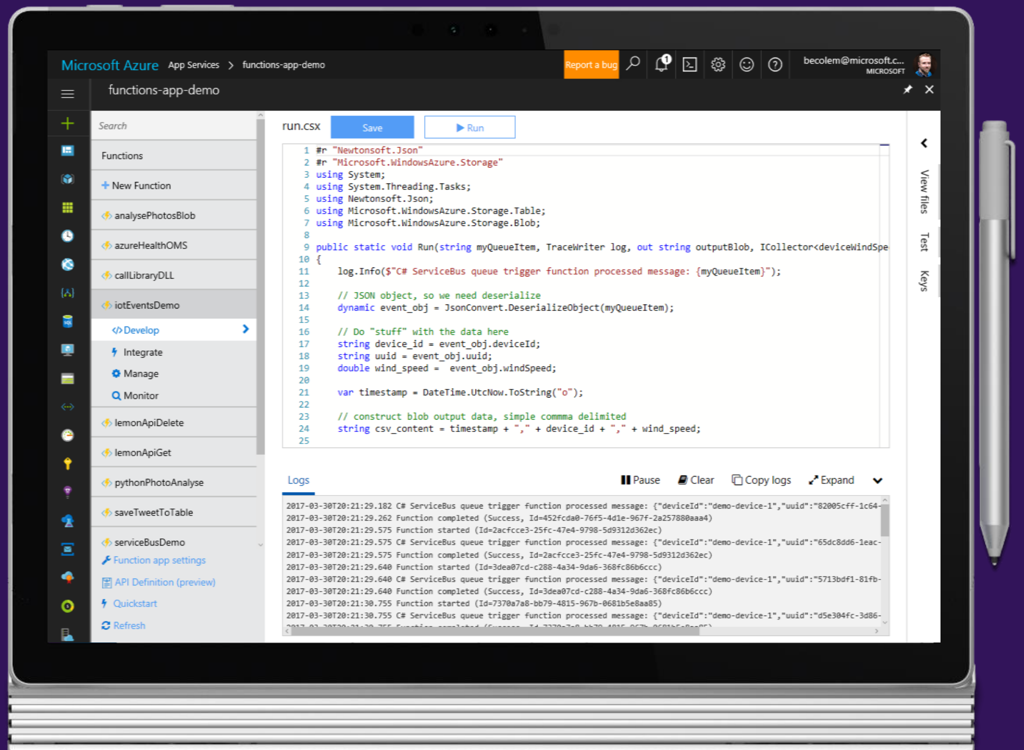
Azure Platform

vent Grid	Logic Apps	Functions
& route events trigger code or logic	Design workflows and orchestrate integration processes	Execute your code based on events & triggers you specify

Database | Storage | Security | IoT | Analytics | Intelligence



Azure Functions



```
#r "Newtonsoft.Json"
#r "Microsoft.WindowsAzure.Storage"
using System;
using System.Threading.Tasks;
using Newtonsoft.Json;
using Microsoft.WindowsAzure.Storage.Table;
using Microsoft.WindowsAzure.Storage.Blob;

public static void Run(string myQueueItem, TraceWriter log, out string outputBlob, ICollector<deviceWindSpe
{
    log.Info($"C# ServiceBus queue trigger function processed message: {myQueueItem}");

    // JSON object, so we need deserialize
    dynamic event_obj = JsonConvert.DeserializeObject(myQueueItem);

    // Do "stuff" with the data here
    string device_id = event_obj.deviceId;
    string uuid = event_obj.uuid;
    double wind_speed =  event_obj.windSpeed;

    var timestamp = DateTime.UtcNow.ToString("o");

    // construct blob output data, simple commma delimited
    string csv_content = timestamp + "," + device_id + "," + wind_speed;

```

```
2017-03-30T20:21:29.182 C# ServiceBus queue trigger function processed message: {"deviceId":"demo-device-1","uuid":"82005cff-1c64-
2017-03-30T20:21:29.262 Function completed (Success, Id=452fcda0-76f5-4d1e-967f-2a257880aaa4)
2017-03-30T20:21:29.575 Function started (Id=2acfcce3-25fc-47e4-9798-5d9312d362ec)
2017-03-30T20:21:29.575 C# ServiceBus queue trigger function processed message: {"deviceId":"demo-device-1","uuid":"65dc8dd6-1eac-
2017-03-30T20:21:29.575 Function completed (Success, Id=2acfcce3-25fc-47e4-9798-5d9312d362ec)
2017-03-30T20:21:29.640 Function started (Id=3dea07cd-c288-4a34-9da6-368fc86b6ccc)
2017-03-30T20:21:29.640 C# ServiceBus queue trigger function processed message: {"deviceId":"demo-device-1","uuid":"5713bdf1-81fb-
2017-03-30T20:21:29.640 Function completed (Success, Id=3dea07cd-c288-4a34-9da6-368fc86b6ccc)
2017-03-30T20:21:30.755 Function started (Id=7370a7a8-bb79-4815-967b-0681b5e8aa85)
2017-03-30T20:21:30.755 C# ServiceBus queue trigger function processed message: {"deviceId":"demo-device-1","uuid":"d5e304fc-3d86-
```

Serverless compute

aka - Function as a Service (FaaS)

Trigger on events & external services / feeds

Pay only per execution

Choice of languages

Open source runtime runs anywhere

App Service – Azure PaaS

Web Apps
Web apps that scale with your business

Mobile Apps
Build Mobile apps for any device

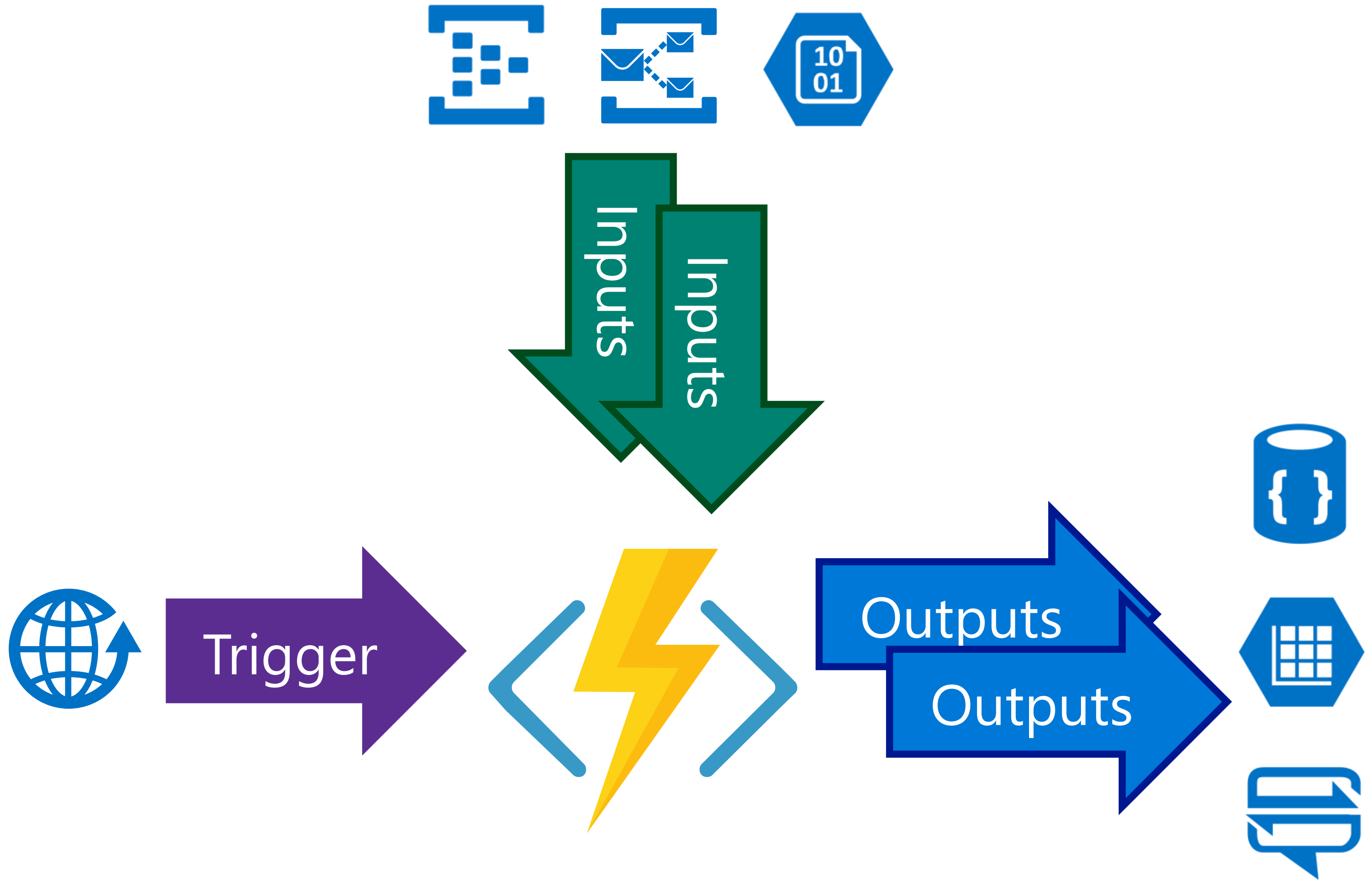
Functions
Create serverless apps without infrastructure

API Apps
Easily build and consume APIs in the cloud

Logic Apps
Automate business process across SaaS and on-premises

Triggers & Bindings

- HTTP / webhook
- Timer / scheduler
- Storage Blob
- Storage Queue
- Storage Table
- Service Bus
- DocumentDB
- Event Hubs / IoT Hubs
- Mobile Apps
- Twilio
- Notification Hubs

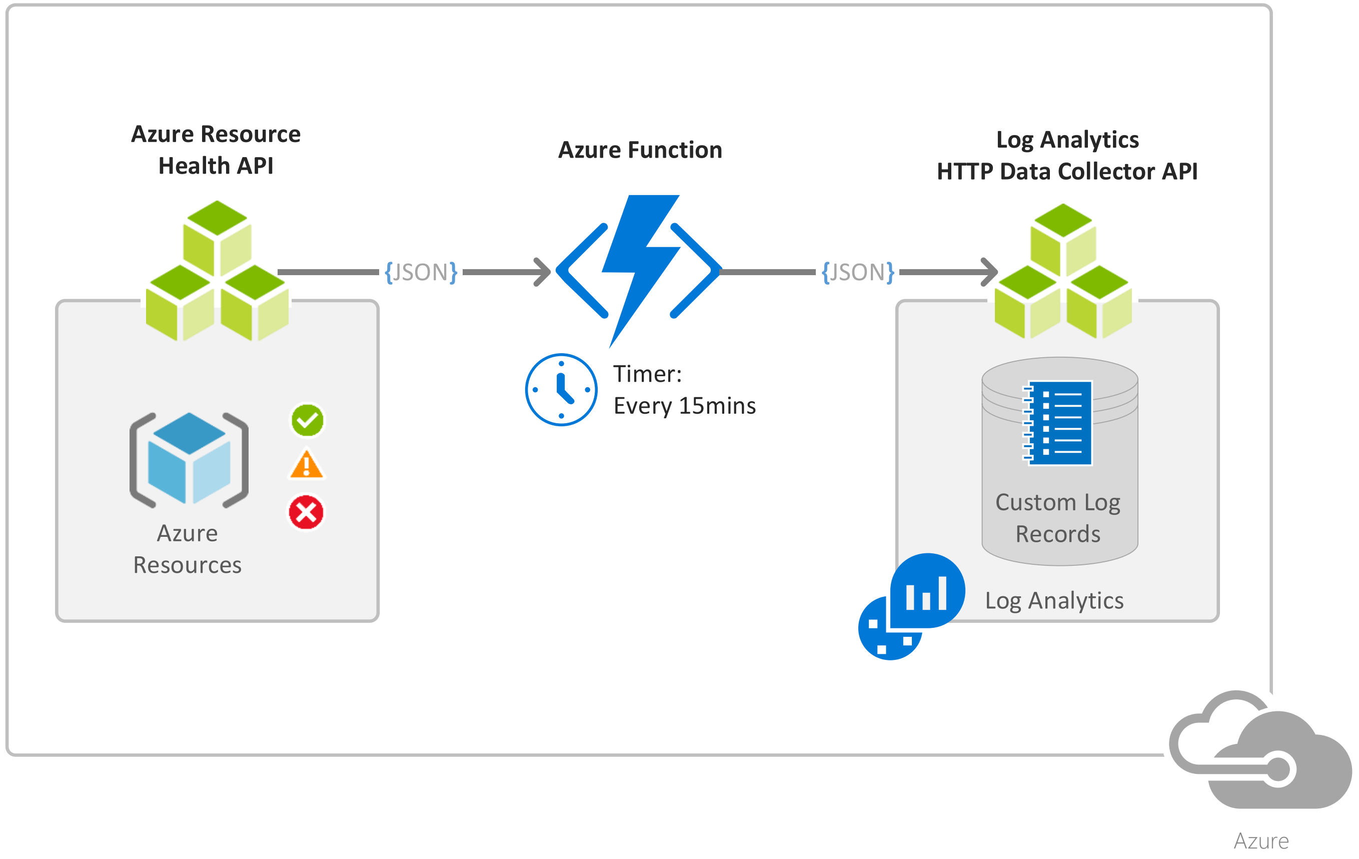


Triggers & Bindings

- However you are **not limited** to the out of the box input & output bindings

- You can write code to do any processing you wish:
  - Make HTTP calls to REST APIs
  - Connect to DBs (ADO)
  - Load an external DLL / library
  - Connect to external services
  - SSH / FTP / SCP

Example Custom Integration
Azure Health to OMS Log Analytics



Language Support

C#

F#

JS

Fully Supported Languages

Power Shell

Python

PHP

Preview / Experimental Languages

Bash

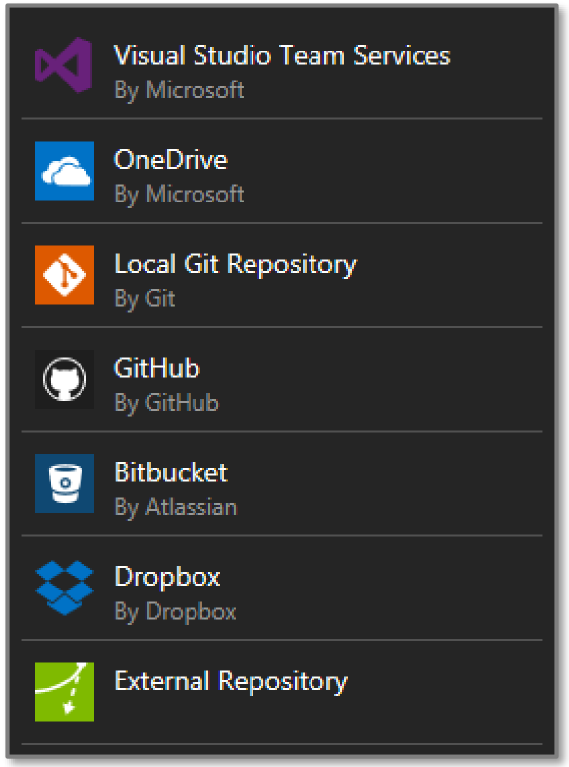
CMD

Developing Functions

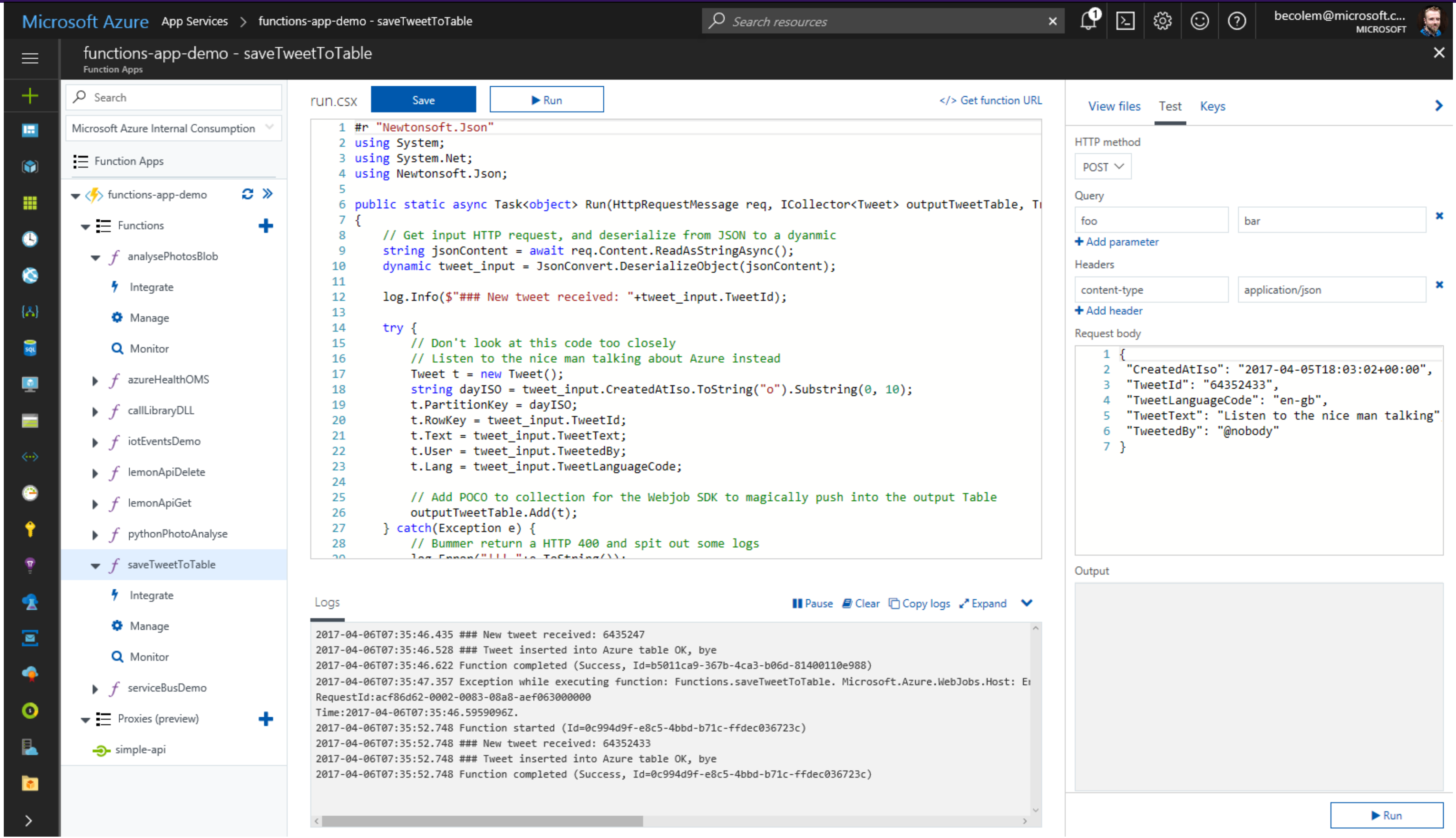
Use Azure portal & in-browser IDE for getting started, prototyping & testing

Use continuous deployment for real world usage

- Use any IDE or code editor
- Number of source control sources:
  - Github / Git
  - Bitbucket
  - Visual Studio Team Services
  - Dropbox / OneDrive



In Browser IDE



```
#r "Newtonsoft.Json"
using System;
using System.Net;
using Newtonsoft.Json;

public static async Task<object> Run(HttpRequestMessage req, ICollector<Tweet> outputTweetTable, T
{
    // Get input HTTP request, and deserialize from JSON to a dynamic
    string jsonContent = await req.Content.ReadAsStringAsync();
    dynamic tweet_input = JsonConvert.DeserializeObject(jsonContent);

    log.Info($"### New tweet received: "+tweet_input.TweetId);

    try {
        // Don't look at this code too closely
        // Listen to the nice man talking about Azure instead
        Tweet t = new Tweet();
        string dayISO = tweet_input.CreatedAtIso.ToString("o").Substring(0, 10);
        t.PartitionKey = dayISO;
        t.RowKey = tweet_input.TweetId;
        t.Text = tweet_input.TweetText;
        t.User = tweet_input.TweetedBy;
        t.Lang = tweet_input.TweetLanguageCode;

        // Add POCO to collection for the Webjob SDK to magically push into the output Table
        outputTweetTable.Add(t);
    } catch(Exception e) {
        // Bummer return a HTTP 400 and spit out some logs
```

HTTP method: POST

Query: foo | bar

Headers: content-type | application/json

Request body

```
{
 "CreatedAtIso": "2017-04-05T18:03:02+00:00",
 "TweetId": "64352433",
 "TweetLanguageCode": "en-gb",
 "TweetText": "Listen to the nice man talking"
 "TweetedBy": "@nobody"
}
```

```
2017-04-06T07:35:46.435 ### New tweet received: 6435247
2017-04-06T07:35:46.528 ### Tweet inserted into Azure table OK, bye
2017-04-06T07:35:46.622 Function completed (Success, Id=b5011ca9-367b-4ca3-b06d-81400110e988)
2017-04-06T07:35:47.357 Exception while executing function: Functions.saveTweetToTable. Microsoft.Azure.WebJobs.Host: E
RequestId:acf86d62-0002-0083-08a8-aef063000000
Time:2017-04-06T07:35:46.5959096Z.
2017-04-06T07:35:52.748 Function started (Id=0c994d9f-e8c5-4bbd-b71c-ffdec036723c)
2017-04-06T07:35:52.748 ### New tweet received: 64352433
2017-04-06T07:35:52.748 ### Tweet inserted into Azure table OK, bye
2017-04-06T07:35:52.748 Function completed (Success, Id=0c994d9f-e8c5-4bbd-b71c-ffdec036723c)
```

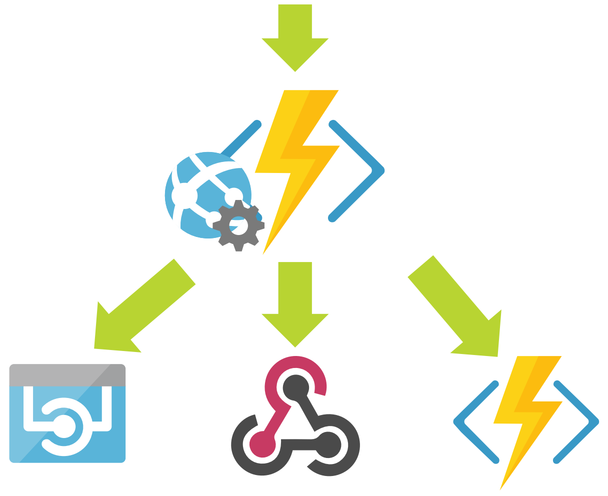
Common Use Cases

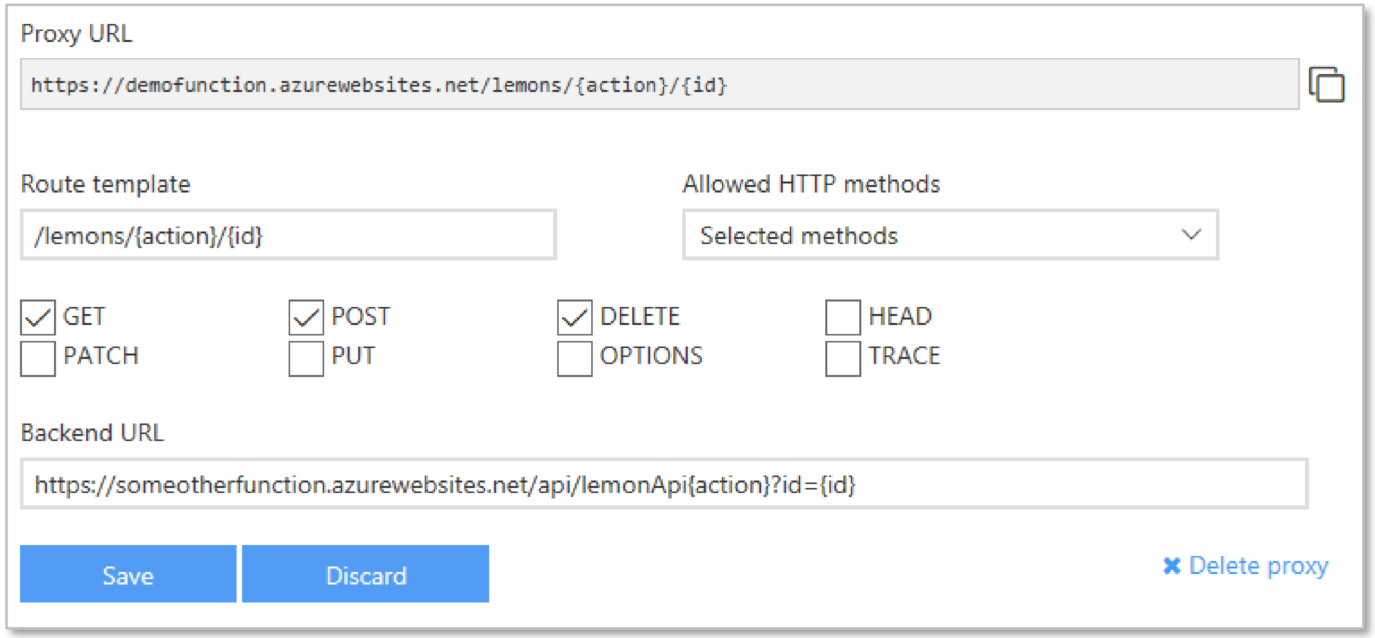
- REST APIs
- Integration logic and “glue”
- Scheduled tasks & app maintenance jobs
- Data ingestion / transform
- Monitoring / watchdogs
- Manage alerts
- Auto Scaling

Azure Function Proxies

PREVIEW

- Create easy REST APIs with custom routing
- Define a single API surface for multiple Function Apps
  - Independent scaling for microservice architecture
- Proxy to multiple APIs
  - Other Function apps
  - Azure API Apps
  - or other URL endpoints





Proxy URL

```
https://demofunction.azurewebsites.net/lemons/{action}/{id}
```

Route template: /lemons/{action}/{id}

Allowed HTTP methods: Selected methods

[✓ GET] PATCH [✓ POST] PUT [✓ DELETE] OPTIONS HEAD TRACE

Backend URL: https://someotherfunction.azurewebsites.net/api/lemonApi{action}?id={id}

DURABLE FUCTIONS

ALPHA

Allows writing of *long-running*, *stateful* function orchestrations

New Orchestrator functions

- They are stateful workflows **authored in code**.
- They can *synchronously* and *asynchronously* **call other functions** and **save output to local variables**.
- They **automatically checkpoint** their progress, so that local state is never lost

Billing & Usage Models

App Service Plan

- Dedicated resources
- Same SKUs and tiers as other App Services
- Pay a fixed hourly rate
- Can be shared with other apps

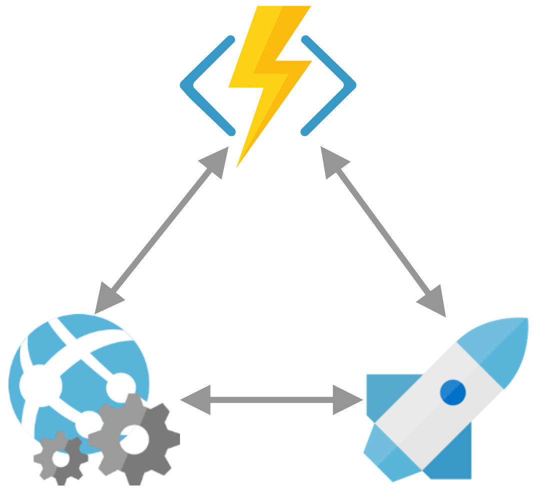
Consumption Plan

- Shared resources
- Limited on execution time and other factors
- Pay per execution
  - £0.15 per million runs
  - £0.000012 per GB/s (gigabyte seconds)

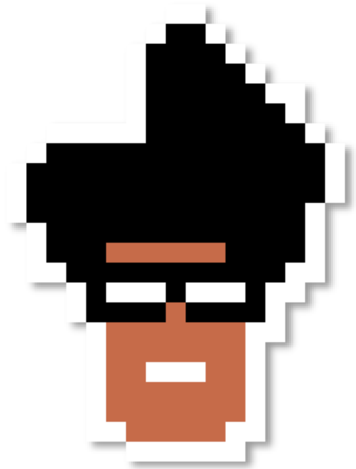
- Choose the correct service plan for Azure Functions
- Pricing Information

Functions vs WebJobs & Logic Apps

- Logic Apps are **codeless** and **workflow** based, optimised for **integration tasks**.
  - Aimed at non-developers
- If part of your integration scenario requires highly specialized logic, use a **Function app**
- Functions are the natural evolution of WebJobs.
  For very simple task scheduling on *existing* Azure Web app, you can use a **WebJob**
- Logic Apps and Functions are designed to be **combined** and used together



Warning!
Nerdy Stuff



Official Guidance and Documentation

Monitoring & Tooling

Monitoring & Metrics

- Azure Functions Integrated with Application Insights
  https://azure.microsoft.com/en-us/updates/azure-functions-now-integrated-with-application-insights/

Tooling & Debugging

- Functions tooling for Visual Studio 2017
  https://docs.microsoft.com/en-us/azure/azure-functions/functions-develop-vs

Local Development

- Azure Functions Core Tools - Local runtime
  https://docs.microsoft.com/en-us/azure/azure-functions/functions-run-local

LIVE DEMO

Try Functions!

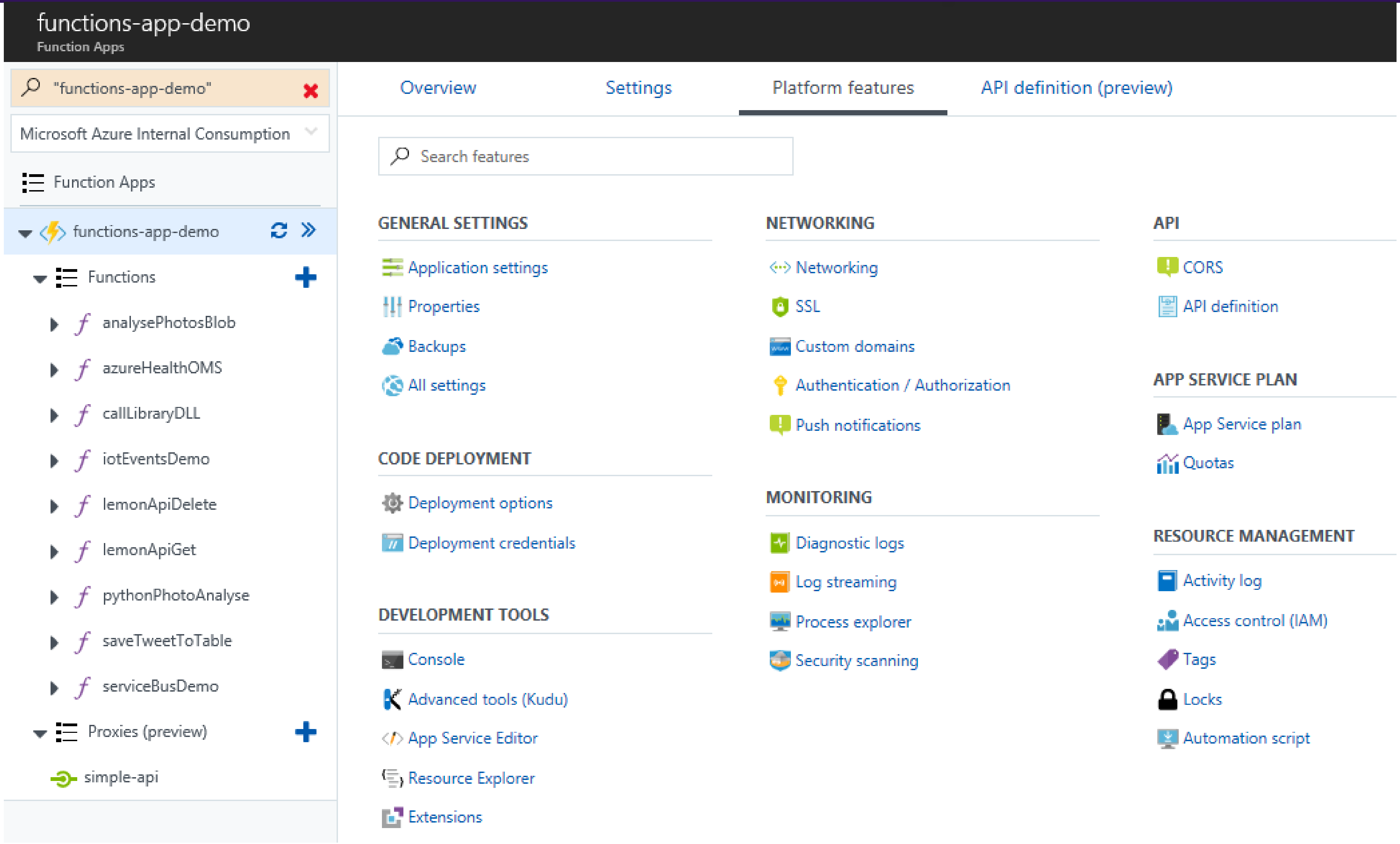
functions.azure.com/try

Reference Links

- My demo library - https://github.com/benc-uk/azure-functions
- Azure Functions on Github - https://github.com/Azure/Azure-Functions
  - Contains links to the other GitHub repos used for Functions
- Documentation - https://docs.microsoft.com/en-us/azure/azure-functions/
- Pricing - https://azure.microsoft.com/en-gb/pricing/details/functions/
- Stack Overflow - http://stackoverflow.com/questions/tagged/azure-functions
- Concurrency controls & tuning - https://github.com/Azure/azure-webjobs-sdk-script/wiki/host.json

Microsoft

New Portal (April 2017) - Easier access to features



GENERAL SETTINGS
- Application settings
- Properties
- Backups
- All settings

CODE DEPLOYMENT
- Deployment options
- Deployment credentials

DEVELOPMENT TOOLS
- Console
- Advanced tools (Kudu)
- App Service Editor
- Resource Explorer
- Extensions

NETWORKING
- Networking
- SSL
- Custom domains
- Authentication / Authorization
- Push notifications

MONITORING
- Diagnostic logs
- Log streaming
- Process explorer
- Security scanning

API
- CORS
- API definition

APP SERVICE PLAN
- App Service plan
- Quotas

RESOURCE MANAGEMENT
- Activity log
- Access control (IAM)
- Tags
- Locks
- Automation script