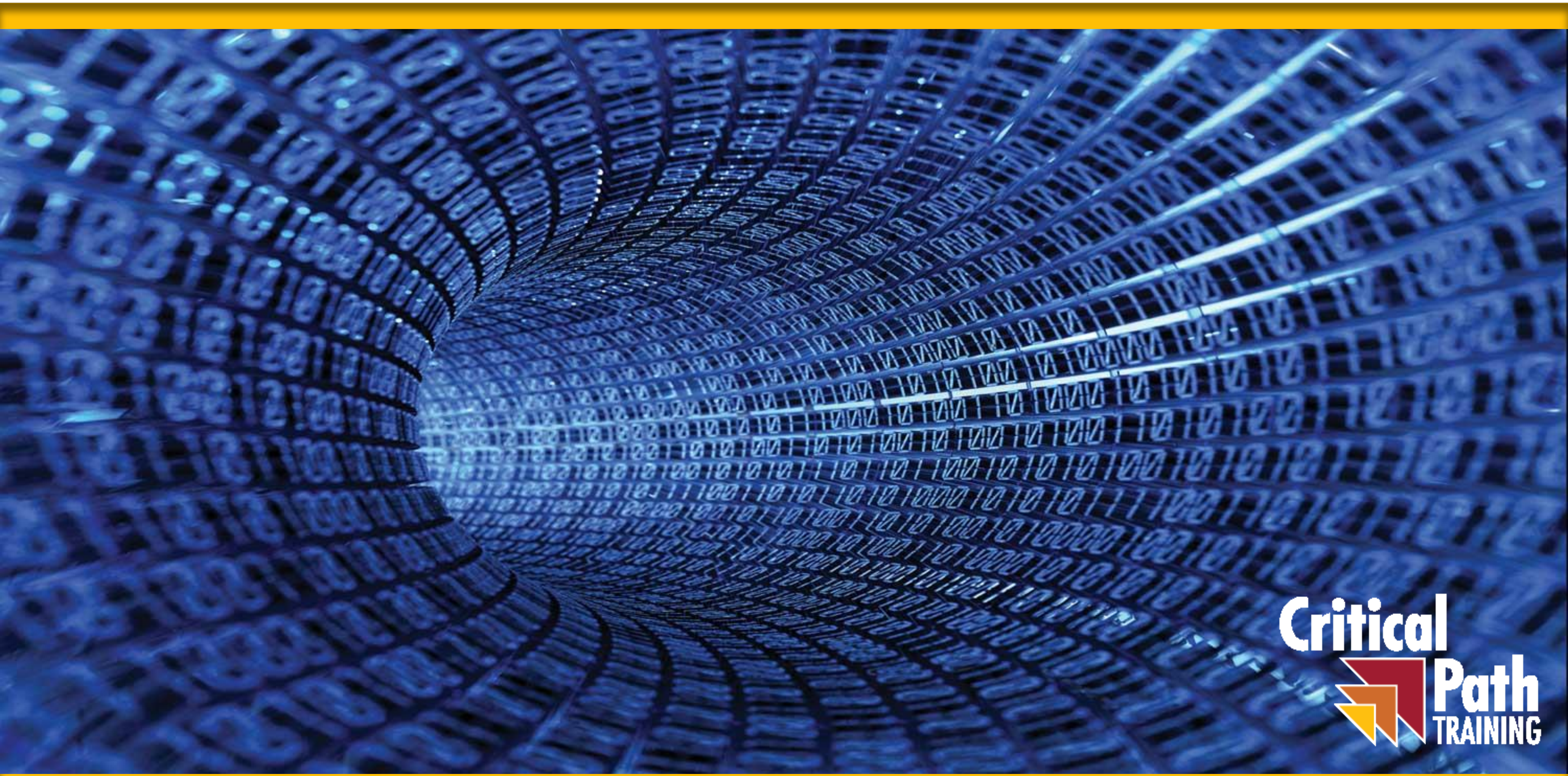
Developing with Azure Functions



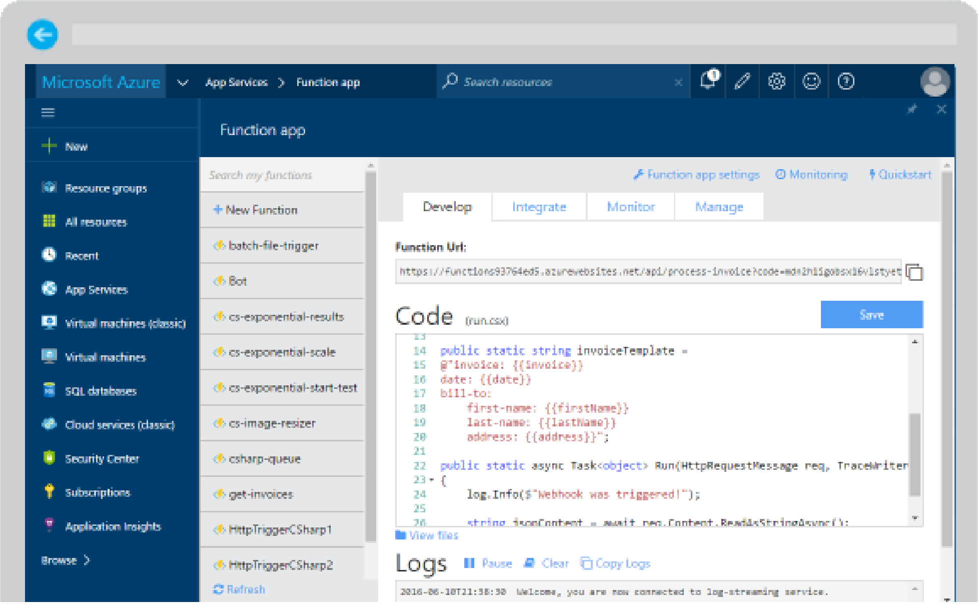
Critical Path TRAINING

Agenda

- Introduction to Developing with Azure Functions
- Creating and Testing Azure Functions in the Azure Portal
- Using Azure Functions to Create a Custom Web API
- Configuring Security and Cross-Origin Resource Sharing
- Calling Azure Functions from SPFX Web Parts
- Developing Azure Function in Visual Studio using C#

Azure Functions

Create a “serverless” event-driven experience that extends the existing Azure App Service platform by building “nanoservices” that can scale based on demand



Function Url:

https://functions93764ed5.azurewebsites.net/api/process-invoice?code=mdn2hl1gobsxl6vlstyet

Code (run.csx)

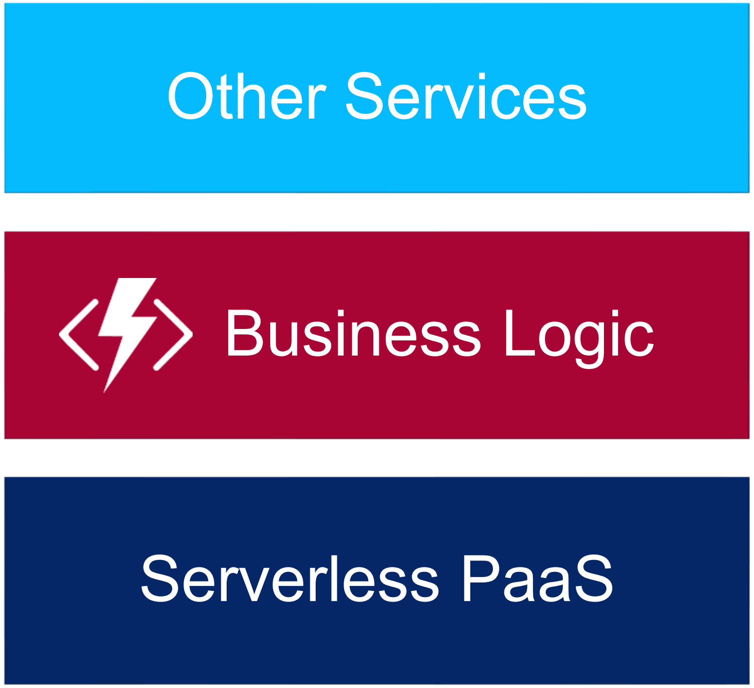
```
public static string invoiceTemplate =
@"invoice: {{invoice}}
date: {{date}}
bill-to:
    first-name: {{firstName}}
    last-name: {{lastName}}
    address: {{address}}";

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

    string jsonContent = await req.Content.ReadAsStringAsync();
```

Logs

```
2016-06-10T21:38:30  Welcome, you are now connected to log-streaming service.
```

Dual abstraction

- Serverless compute abstracts away the compute
- Bindings abstract away services you interact with



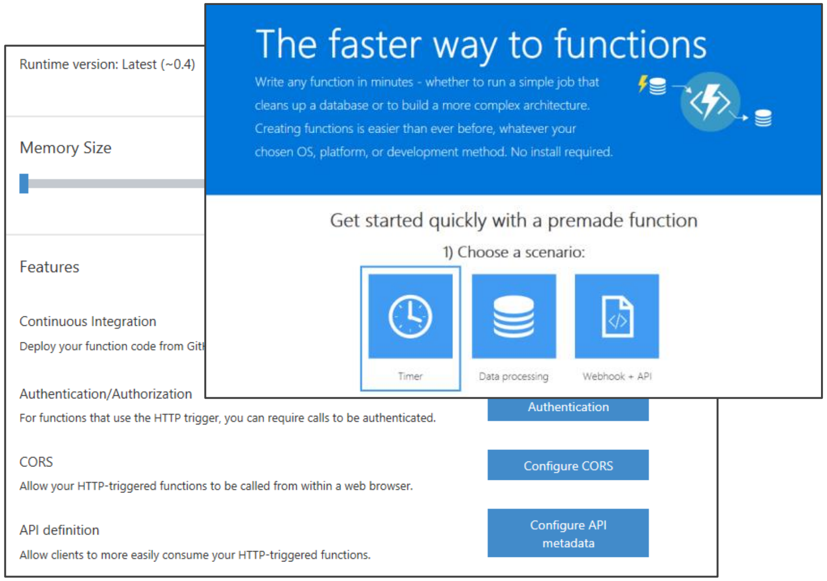
Other Services

Business Logic

Serverless PaaS

Supported Languages and Tools

Create functions in JavaScript, C#, Python, and PHP, as well as scripting options such as Bash, Batch, and PowerShell, that can be triggered by virtually any event in Azure, 3rd party services, or on premise systems



Runtime version: Latest (~0.4)

Memory Size

Features

Continuous Integration

Deploy your function code from Git

Authentication/Authorization

For functions that use the HTTP trigger, you can require calls to be authenticated.

CORS

Allow your HTTP-triggered functions to be called from within a web browser.

API definition

Allow clients to more easily consume your HTTP-triggered functions.

The faster way to functions

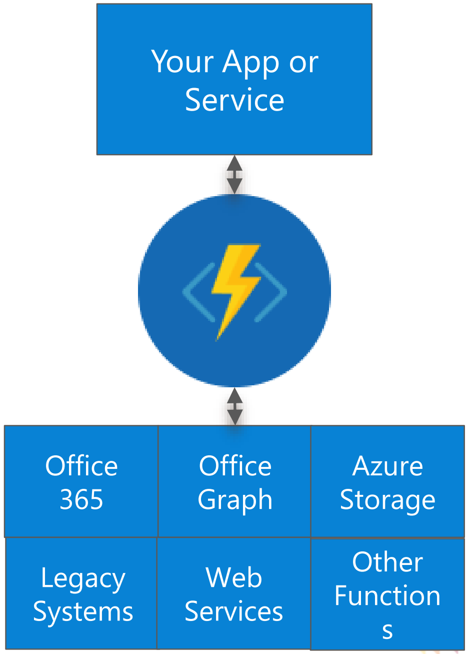
Write any function in minutes - whether to run a simple job that cleans up a database or to build a more complex architecture. Creating functions is easier than ever before, whatever your chosen OS, platform, or development method. No install required.

Get started quickly with a premade function

1) Choose a scenario:

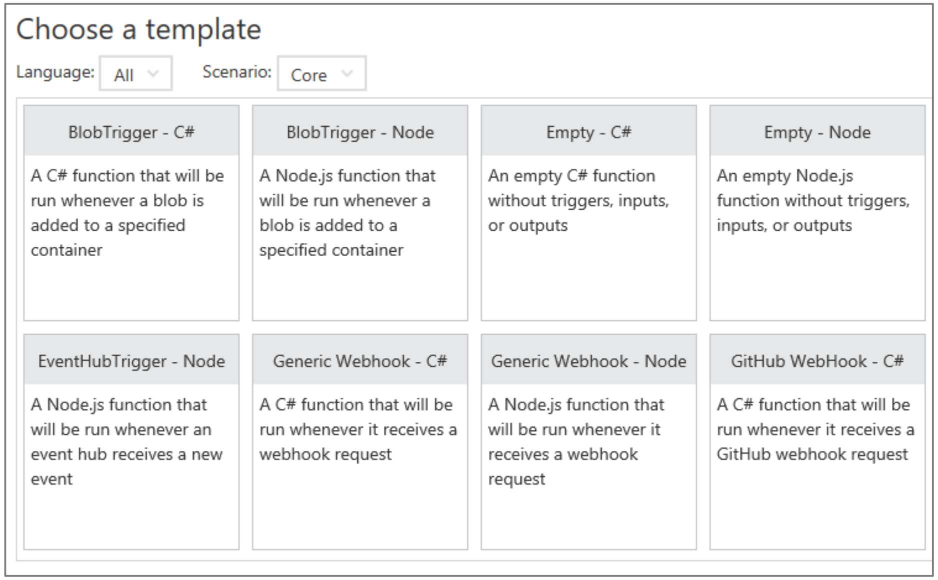
Common Scenarios

- Timer-based processing
- Azure service event processing
- SaaS event processing
- Serverless web application architectures
- Serverless mobile backends
- Real-time stream processing
- Real-time bot messaging



Function App Templates

- Function App templates
  - BlobTrigger
  - EventHubTrigger
  - Generic webhook
  - GitHub webhook
  - HTTPTrigger
  - QueueTrigger
  - ServiceBusQueueTrigger
  - ServiceBusTopicTrigger
  - TimerTrigger
  - Blank & Experimental



Choose a template

Language: All   Scenario: Core

BlobTrigger - C#
A C# function that will be run whenever a blob is added to a specified container

BlobTrigger - Node
A Node.js function that will be run whenever a blob is added to a specified container

Empty - C#
An empty C# function without triggers, inputs, or outputs

Empty - Node
An empty Node.js function without triggers, inputs, or outputs

EventHubTrigger - Node
A Node.js function that will be run whenever an event hub receives a new event

Generic Webhook - C#
A C# function that will be run whenever it receives a webhook request

Generic Webhook - Node
A Node.js function that will be run whenever it receives a webhook request

GitHub WebHook - C#
A C# function that will be run whenever it receives a GitHub webhook request

Timer Function Apps

- Run at explicitly specified intervals
  - e.g. executes once every 5 minutes
  - Configured using CRON expressions ("0 */5 * * * *")
  - Can send information to other systems, but typically don’t “return” information, only write to logs
  - Great for redundant cleanup and data management
  - Great for checking state of services
  - Can be combined with other functions

Data Processing Function Apps

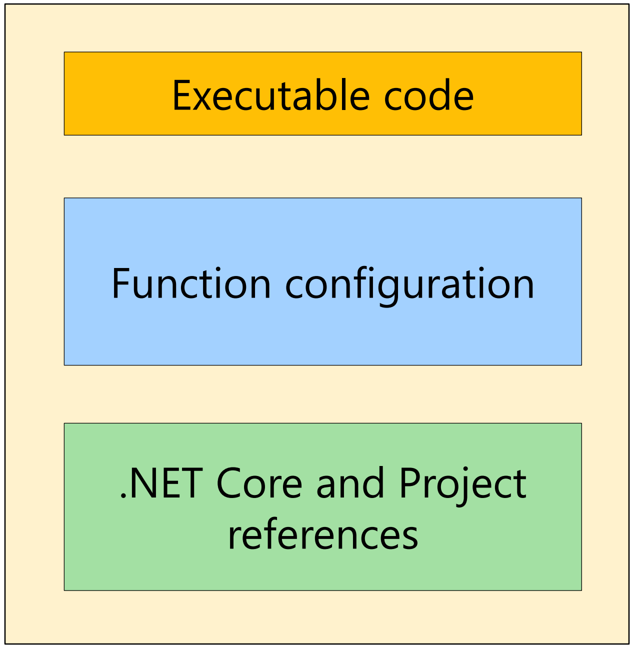
- Run when triggered by a data event, such as an item being added to a queue or container
  - Typically have in and out parameters
  - Great for responding to CRUD events
  - Great for performing CRUD events
  - Great for moving content
  - Access data across services

Webhook & API Function Apps

- Triggered by events in other services
  - GitHub
  - Team Foundation Services
  - Office 365
  - OneDrive
  - Microsoft PowerApps
- Takes in a request and sends back a response
  - Often mimic Web API and legacy web services flows
  - Typically need CORS settings managed
  - Best for exposing functionality to other apps and services
  - Great for building Logic Apps

Anatomy of a Function

- “Run” file that containing the function code
- “Function” file containing service & trigger bindings & parameters
- “Project” file containing project assembly and NuGet packages
- App Service settings, such as connection strings and API keys



Executable code

Function configuration

.NET Core and Project references

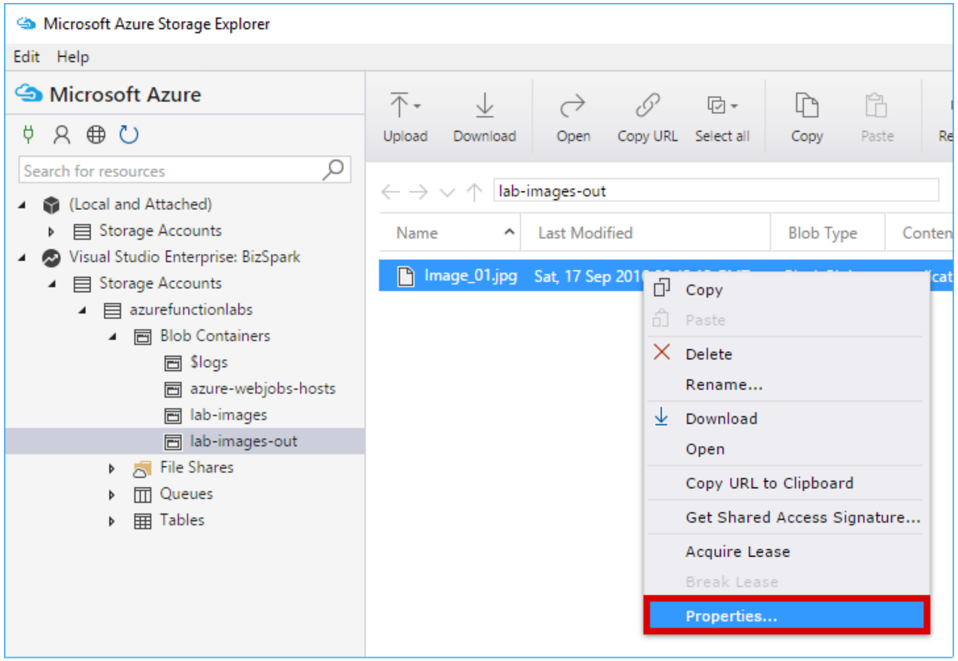
Function Bindings

- Bindings used to create connections to and from function
  - Many bindings can be “bi-directional” as well.

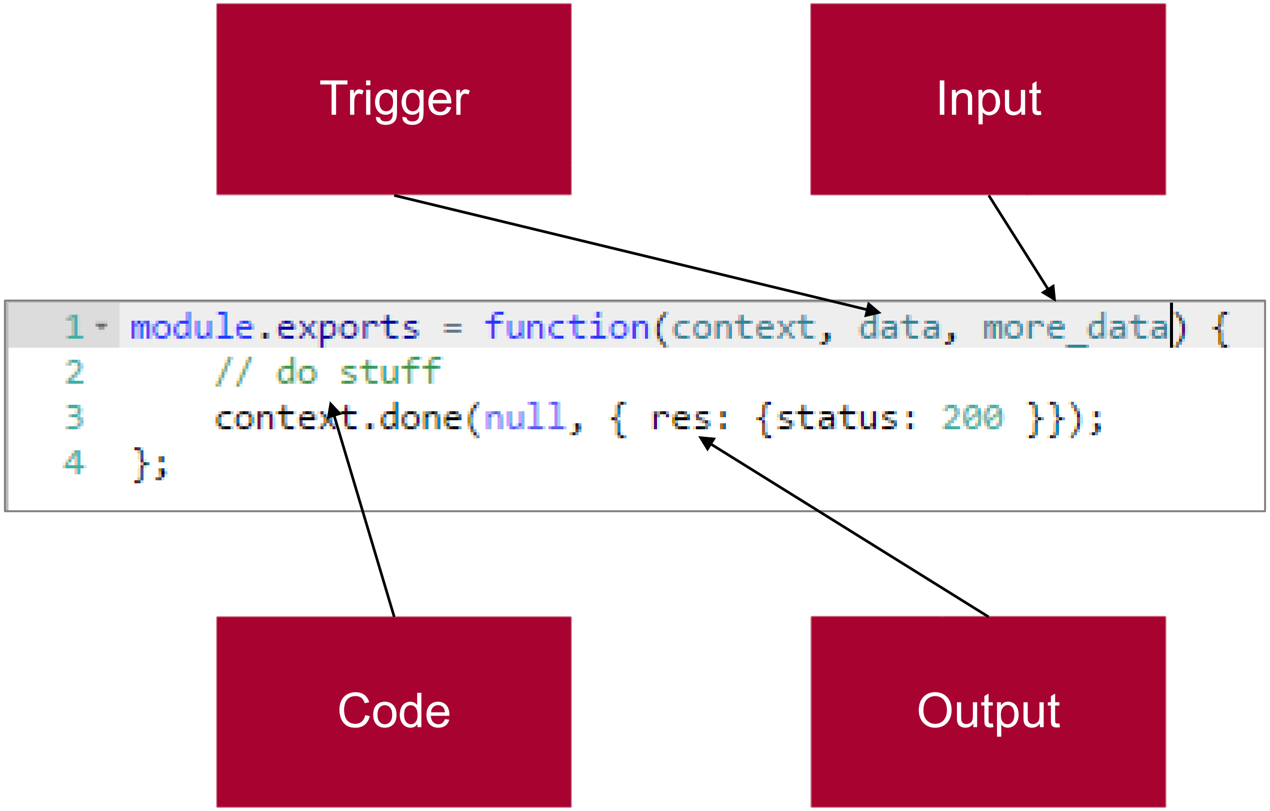
Type	Service	Trigger	Input	Output
Schedule	Azure Functions	✓		
HTTP (REST or webhook)	Azure Functions	✓		✓*
Blob Storage	Azure Storage	✓	✓	✓
Events	Azure Event Hubs	✓		✓
Queues	Azure Storage	✓		✓
Tables	Azure Storage		✓	✓
Tables	Azure Mobile Apps		✓	✓
No-SQL DB	Azure DocumentDB		✓	✓
Push Notifications	Azure Notification Hubs			✓

Testing Functions

- Command-line tools
- 3rd party products such as Postman and Swagger
- Direct web calls via cURL
- Nested functions
- Microsoft Azure Storage Explorer
- Visual Studio Cloud Explorer

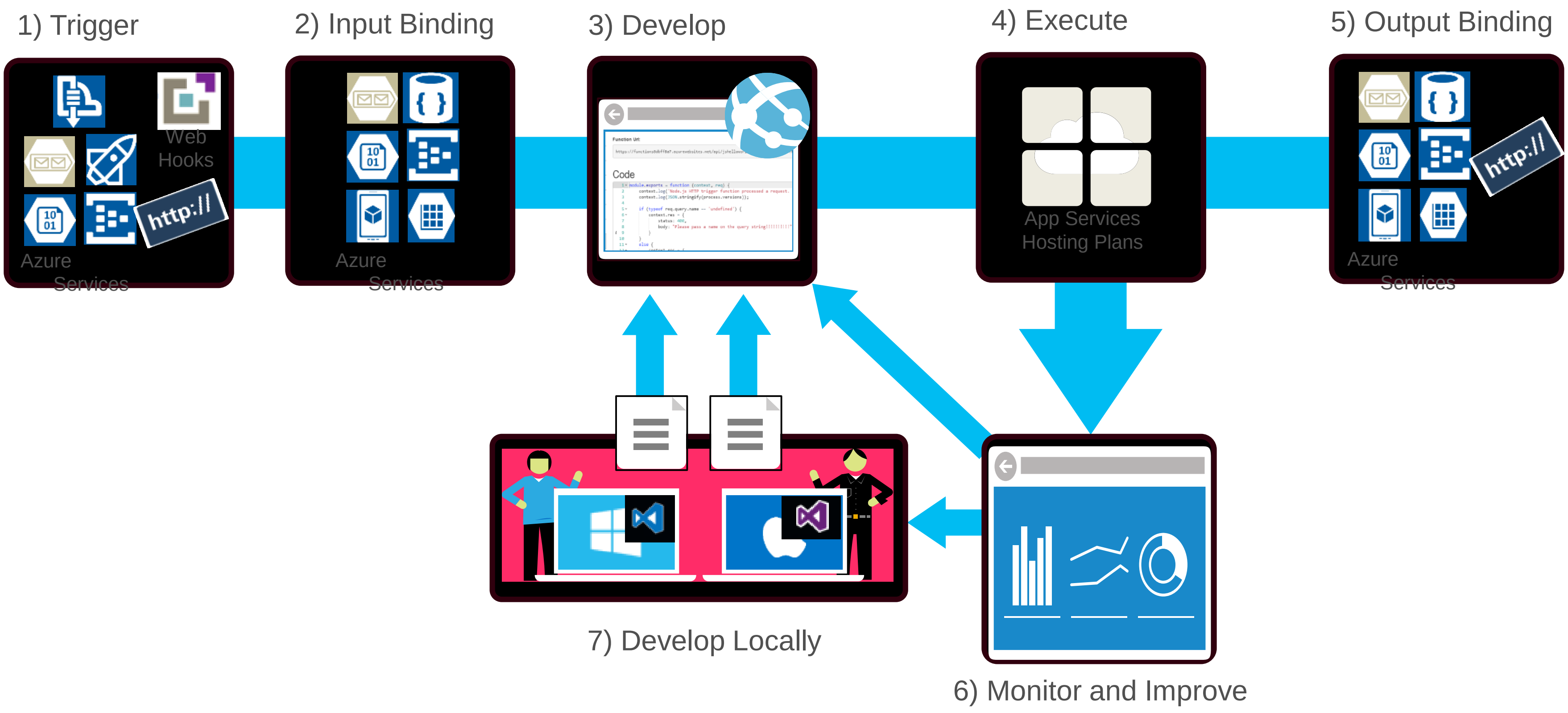


Functions programming concepts



```
module.exports = function(context, data, more_data) {
    // do stuff
    context.done(null, { res: {status: 200 }});
};
```

Azure Function Development Process



1) Trigger

Web Hooks

Azure Services

2) Input Binding

Azure Services

3) Develop

Function Url

https://functions8dbff8e7.azurewebsites.net/api/jshellowor

Code

```
module.exports = function (context, req) {
    context.log('Node.js HTTP trigger function processed a request.
    context.log(JSON.stringify(process.versions));

    if (typeof req.query.name == 'undefined') {
        context.res = {
            status: 400,
            body: "Please pass a name on the query string!!!!!!!!!!"
        }
    }
    else {
        context.res = {
```

4) Execute

App Services Hosting Plans

5) Output Binding

Azure Services

6) Monitor and Improve

7) Develop Locally