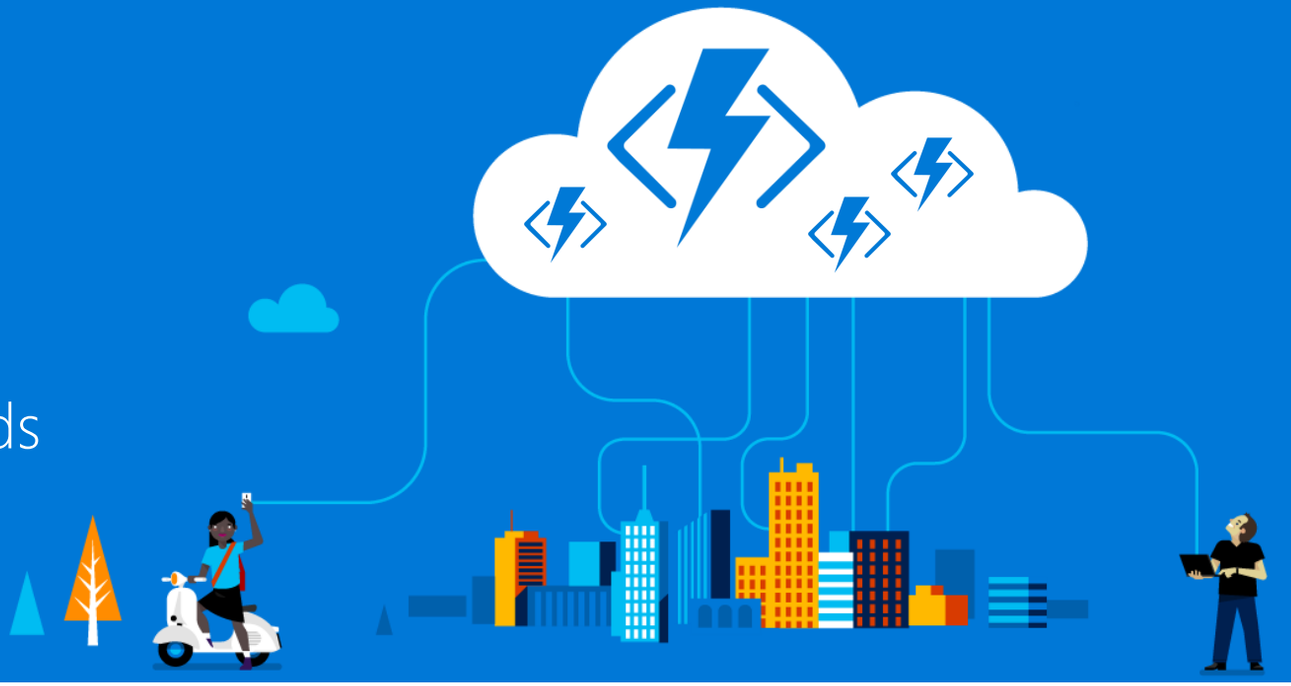
Server-less compute with .NET based Azure Functions

Alex Thissen
Cloud architect at Xpirit, The Netherlands
@alexthissen

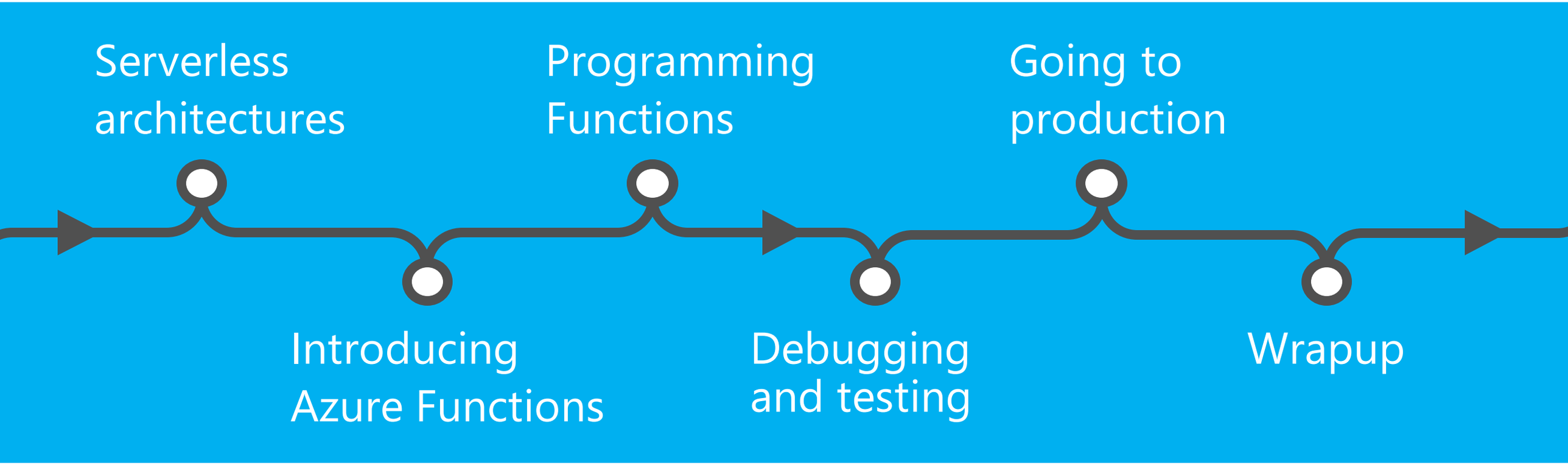


DOT NEXT

Xpirit
Think ahead. Act now.

MVP Microsoft® Most Valuable Professional

Agenda


Serverless architectures
Introducing Azure Functions
Programming Functions
Debugging and testing
Going to production
Wrapup

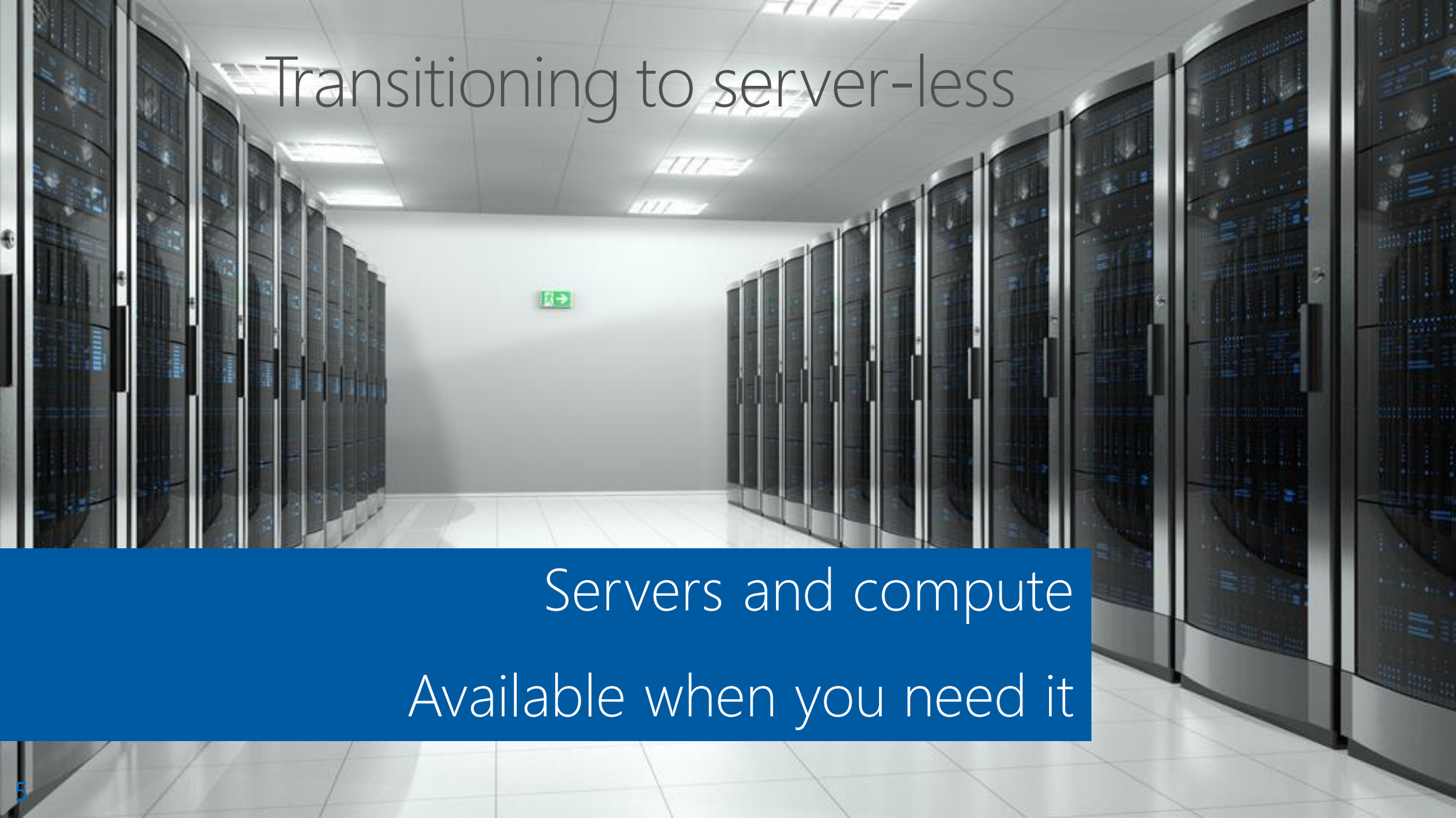
Server-less architectures and Azure Functions

Transitioning to server-less

There are no servers…

When you do not need them

Transitioning to server-less

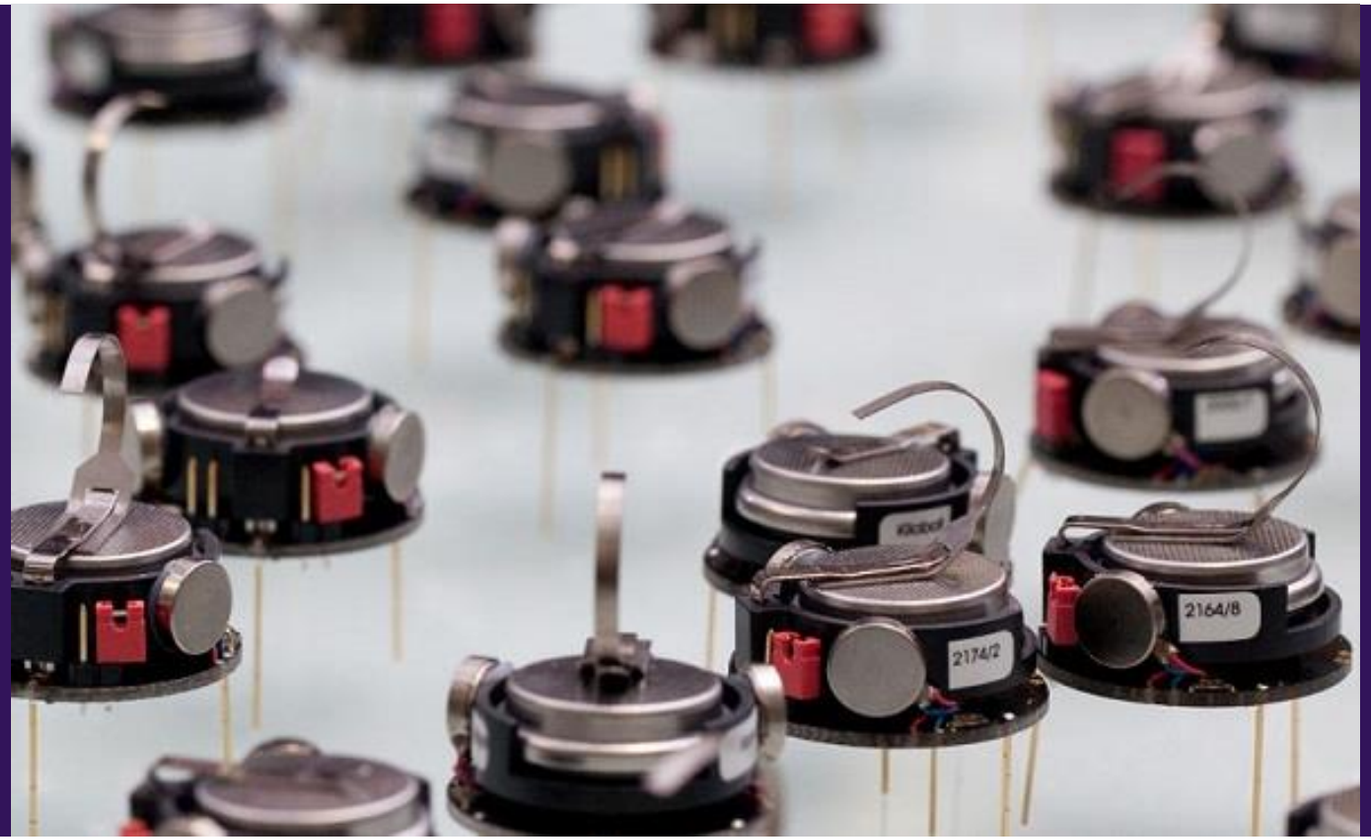


Servers and compute

Available when you need it

Functions as a Service (FaaS)

Small pieces of self-contained server-side logic



Event-driven

Responds to external triggers

Instant scaling

Abstraction of server infrastructure

Scales when needed

Pay by consumption

Charged by sub-second units of compute

Server-less platform providers

Major cloud provider offer FaaS

New competitors enter competition

Amazon Lambda
(since 2014)

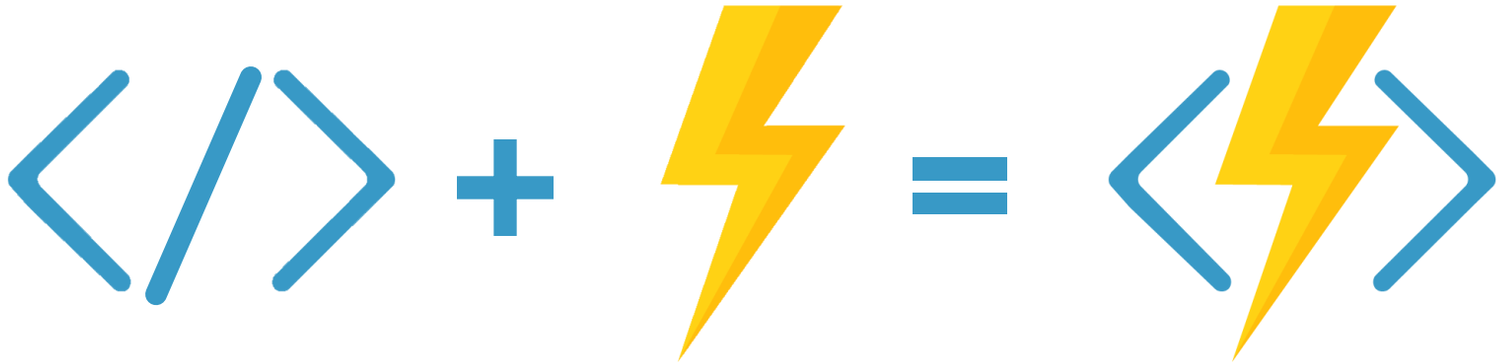
Google Cloud Functions
(since 2016)

Azure Functions
(since 2016)

Auth0 Webtask.IO
(since 2016)

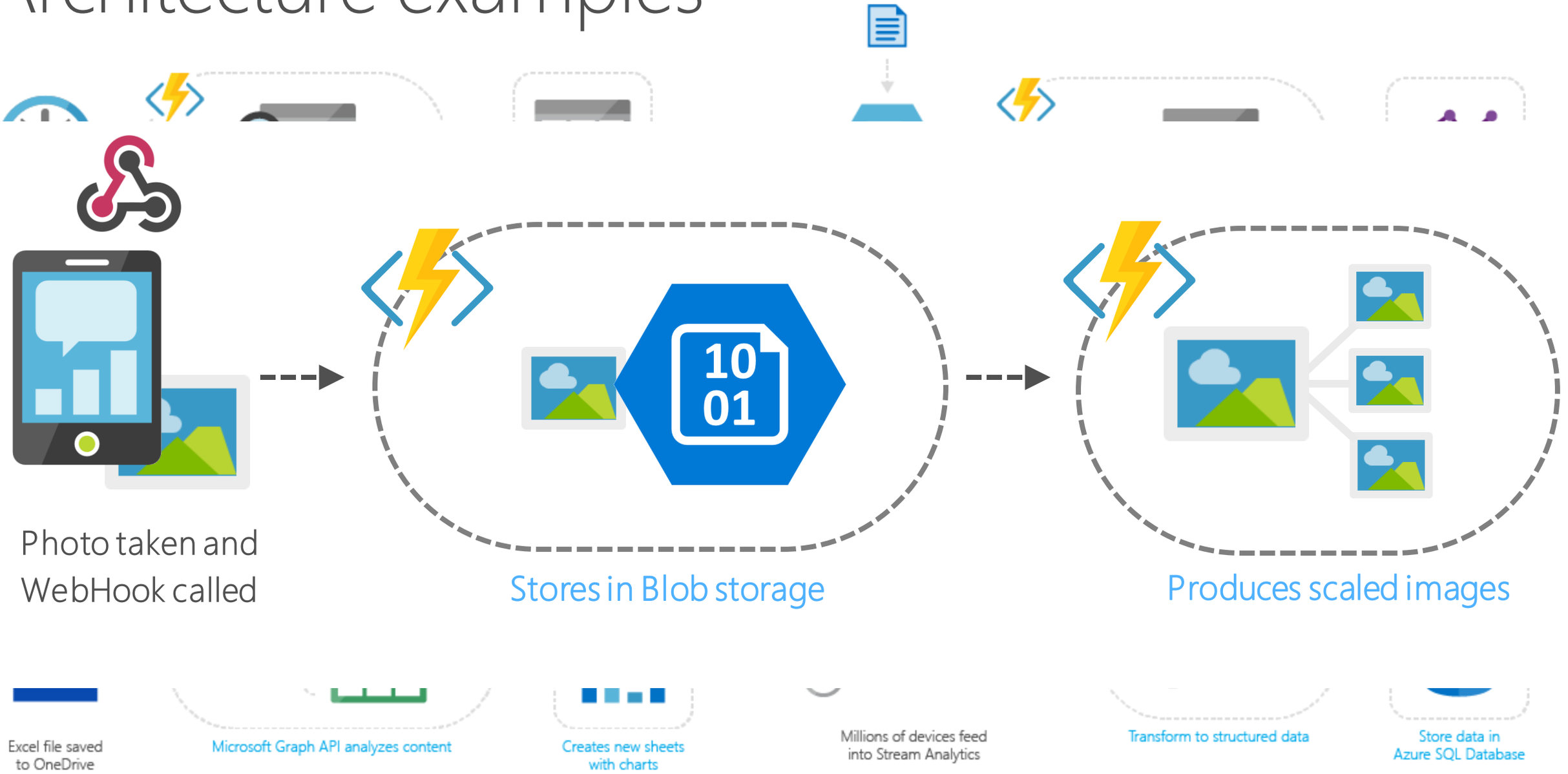
Focusing on Azure Functions

Code + Events + data = Azure Functions



Process events with server-less code

Architecture examples



Anatomy of an Azure Function App

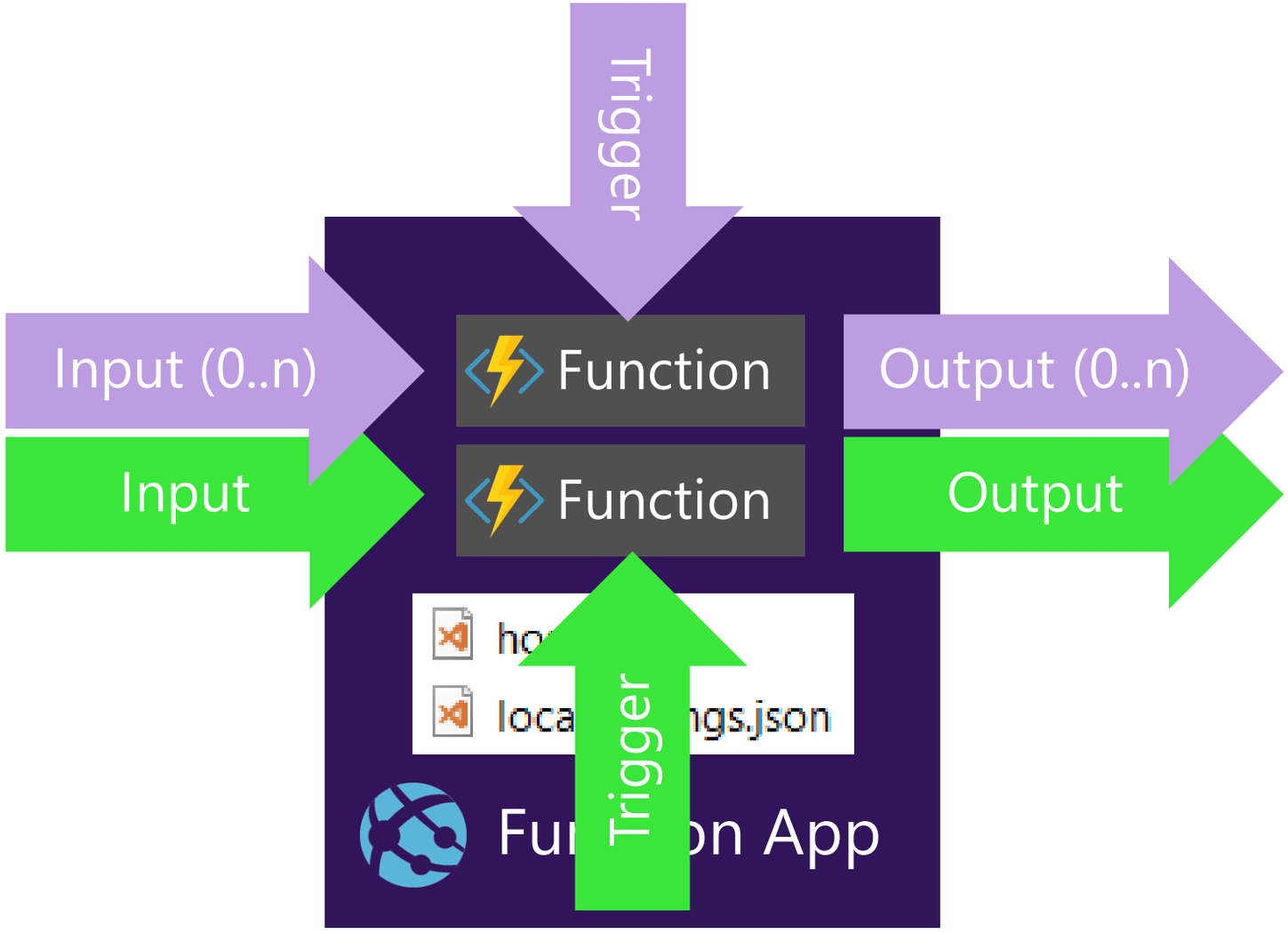
Function apps

- Hosted as Azure App Service
- JSON based configuration
- Running (multiple) functions

Bindings for input and output

Zero or more possible

Triggers and bindings can vary per function



Some trivia about Azure Functions

Open source

http://github.com/Azure/Azure-Functions

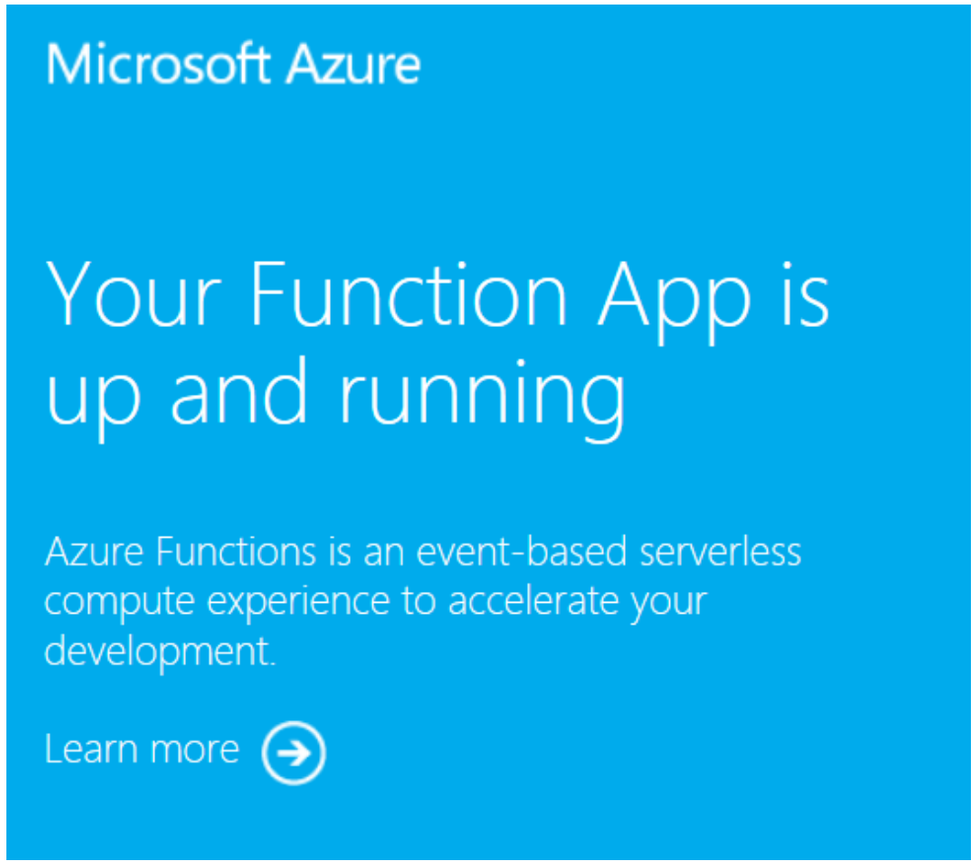
Many available languages

C#, JavaScript, F#, Java, PowerShell, Python, PHP, Batch, Bash

Limited execution time

By default 5 minutes to reach completion

Built on top of Azure WebJobs



Functions and WebJobs

Code

Config

Language Runtime
C#, Node.js, PHP, …

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 Consumption Runtime
Hosting, Deployment, CI, Remote debugging, …

Programming Azure Functions

1

Scripts

Azure Portal or Visual Studio Code

Instant feedback and testing

Less control

Azure tooling generates metadata files

Integrates with source control

2

.NET with C#

Visual Studio 2017

Achieve higher code quality

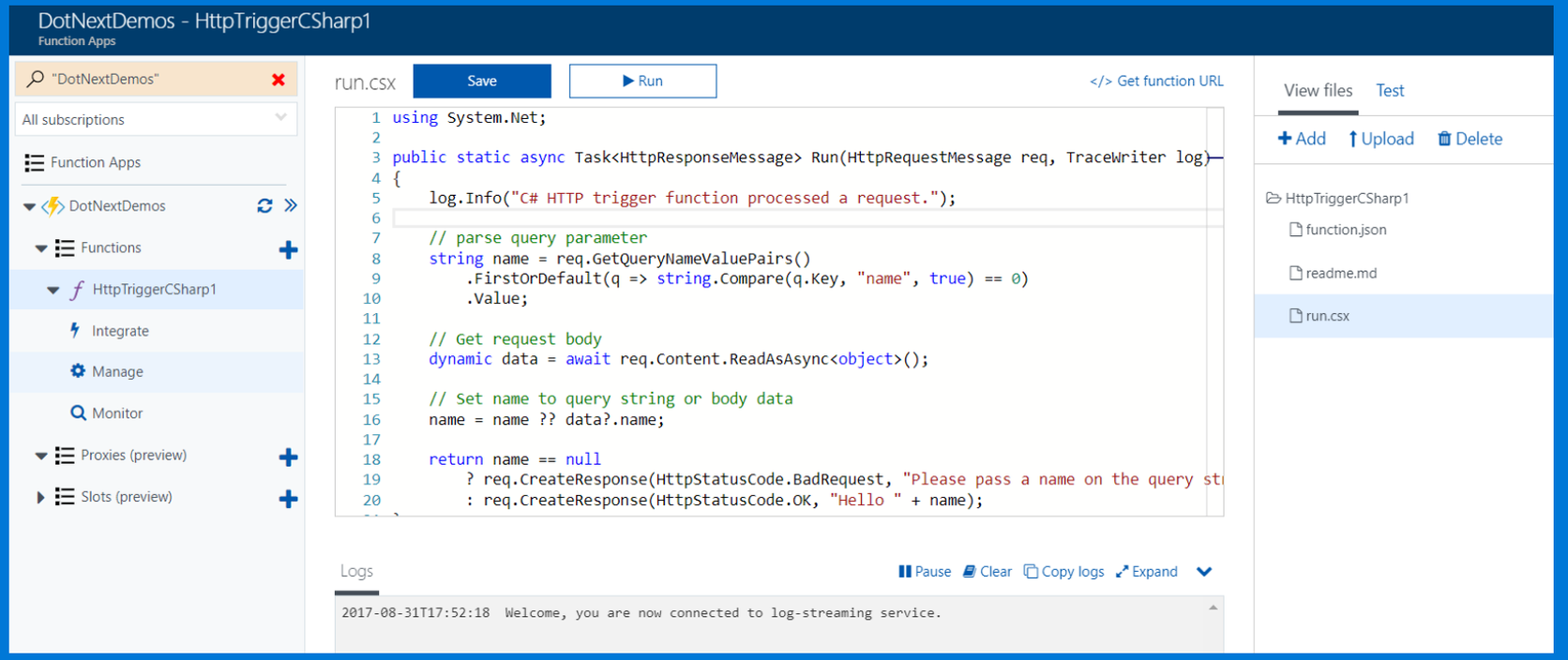
Compilation and code analysis

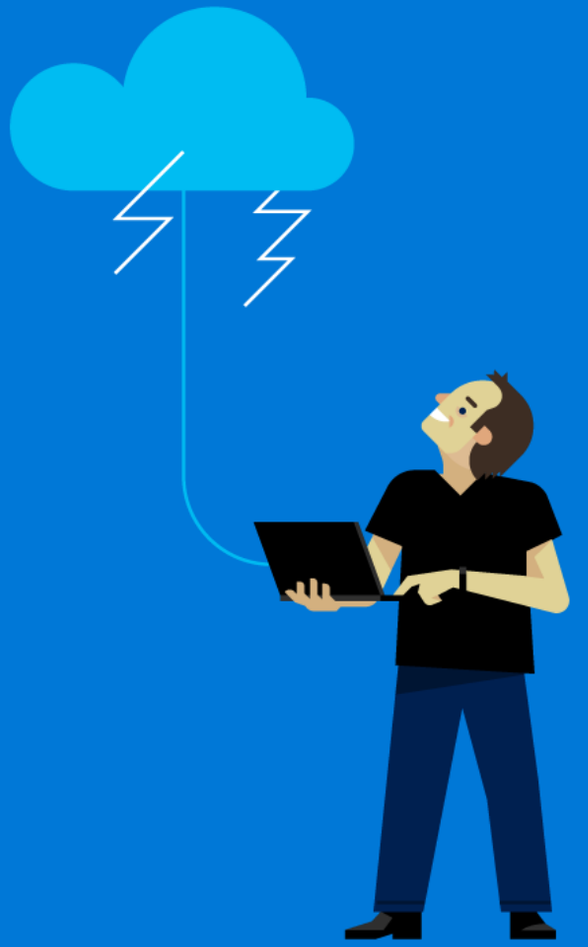
Unit testing

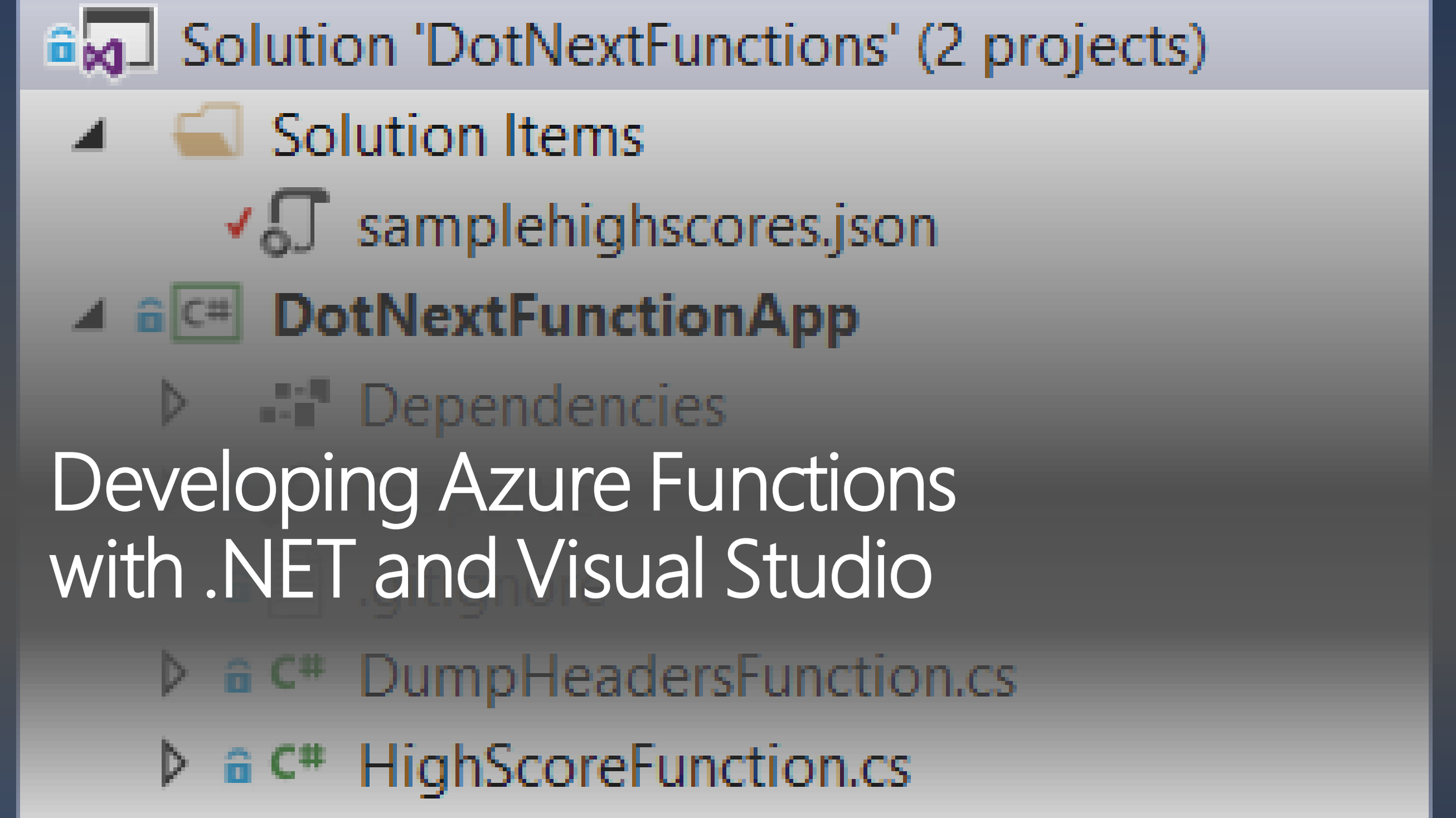
Combine in CI/CD pipelines

Demo

Creating and using Azure Functions







Developing Azure Functions with .NET and Visual Studio

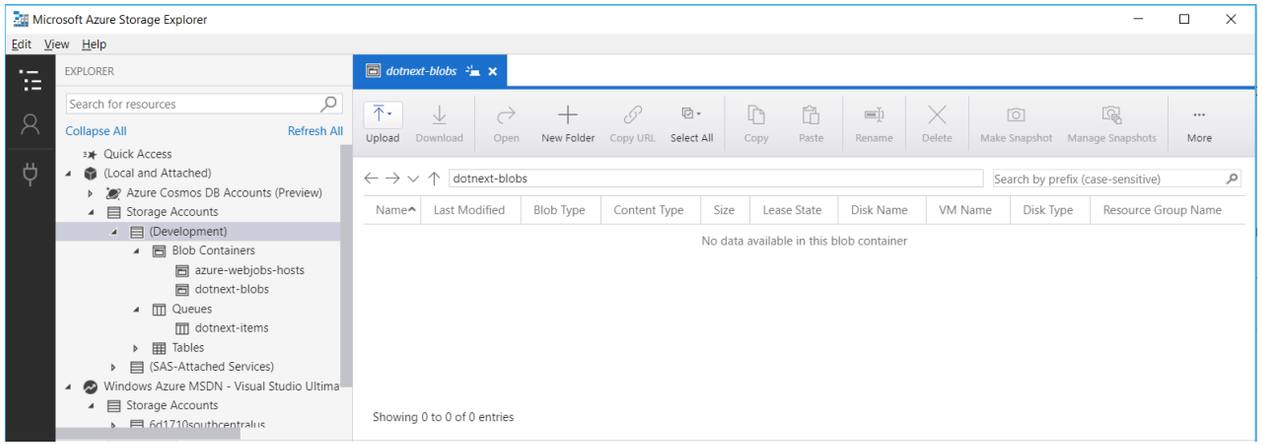
Getting started with local development

Required tooling

Visual Studio 2017 15.3 + Azure workload
Visual Studio Tools for Azure Functions
.NET Core 2.0 installation
Azure Functions Core Tools (includes CLI)

Azure Storage Explorer

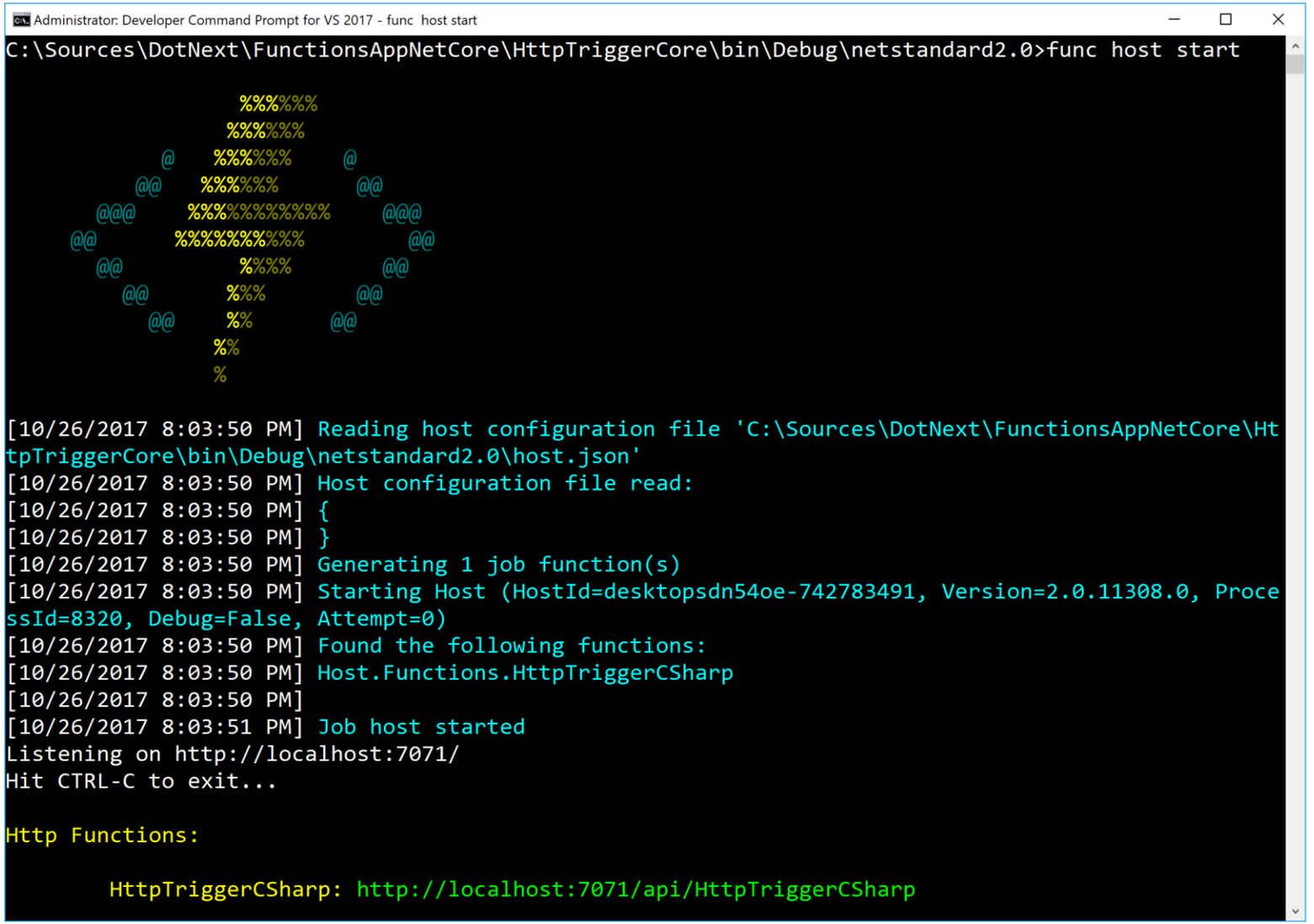
For interacting with Azure and local emulator



Useful optionals

HTTP requests: Postman or SoapUI

Command-line Interface



Part of Azure Function Core Tools

Create and manage functions locally

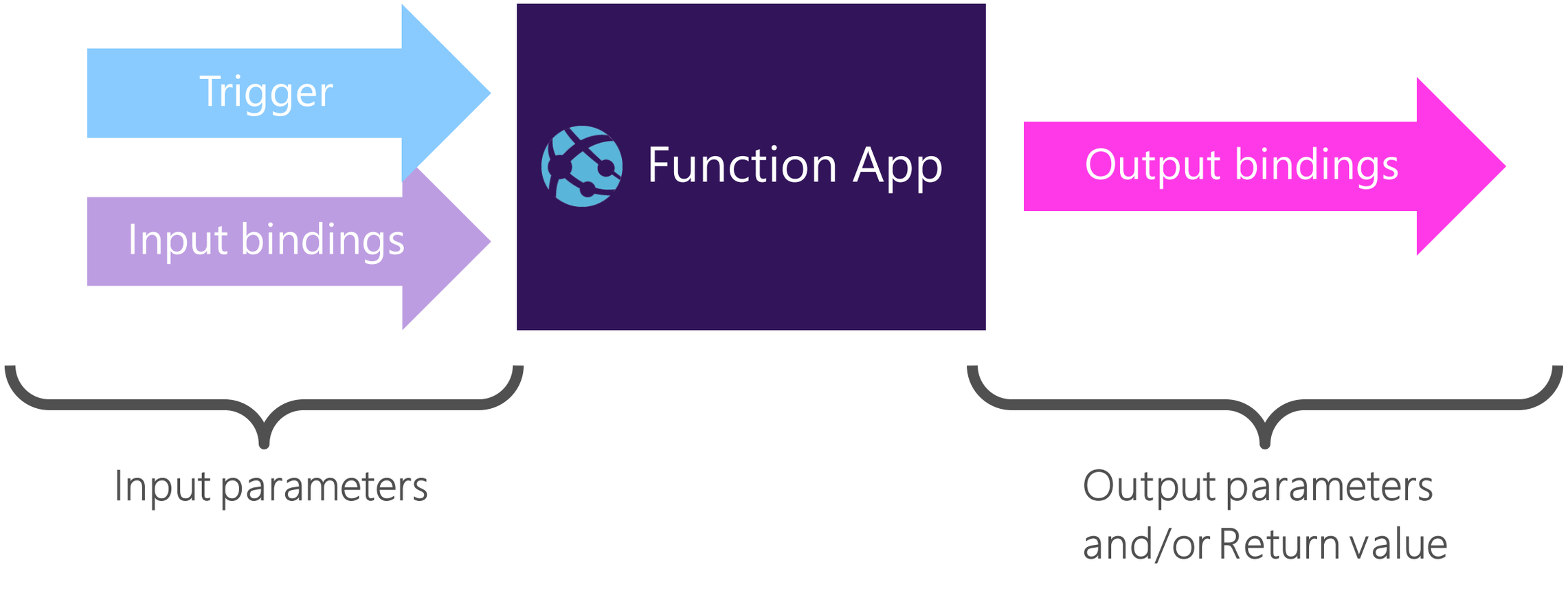
Host and run functions

Settings and security

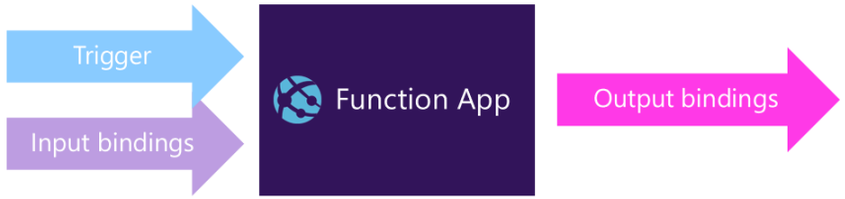
Publishing

Manage Azure subscriptions

Triggers and Bindings



Triggers and Bindings



Define function.json metadata with attributes

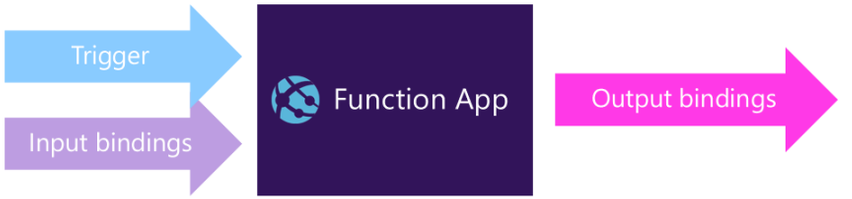
```
[FunctionName("HelloWorldFunction")]
public static HttpResponseMessage Run(
    [HttpTrigger] HttpRequestMessage req,
    TraceWriter log)
{
    log.Info("C# HTTP trigger function processed a request.");

    return req.CreateResponse(HttpStatusCode.OK, "Hello, World!");
}
```

Return value
Single output binding can be done with return type

Trigger source
Defines what this function is triggered by

Triggers and Bindings

Define function.json metadata with attributes




```
[FunctionName("ImageAnalyzer")]
[return: Queue("AnalysisResults", Connection="…")]
public static string Run(
 [QueueTrigger("ImagesToAnalyze", Connection="…")] string imageName,
 [Blob("images/{queueTrigger}",
  FileAccess.Read,
  Connection = "…")] Stream blob, TraceWriter log) {
  return ...;
}
```

Return value
Single output binding can be done with return type

Trigger source
Connection property refers to value from settings

Additional inputs (and outputs)
Attribute values can refer to trigger metadata with { }

Trigger and bindings

Growing list of supported types

HTTP WebHooks

Timers

Azure resources

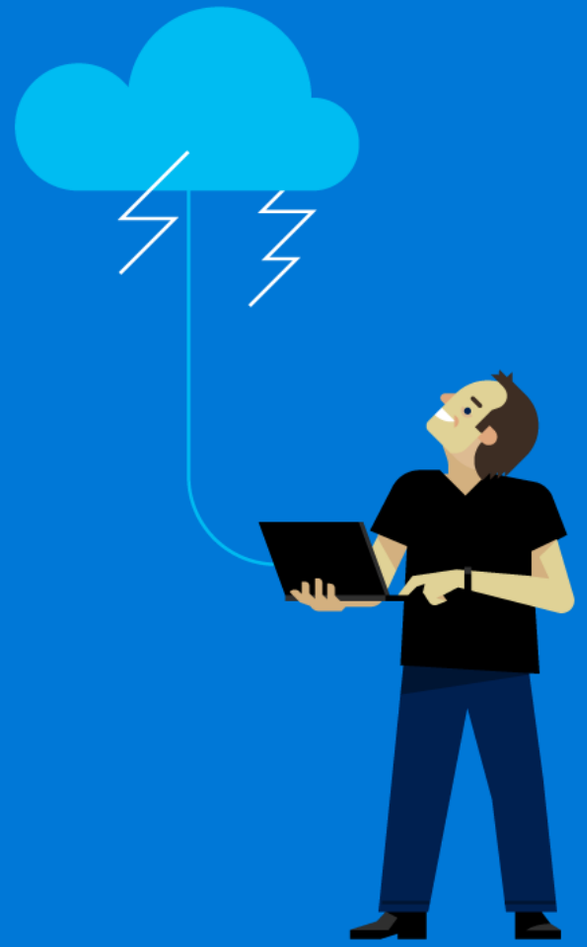
Office365 Graph and OneDrive

Abstract away details

Semantics, conversion and plumbing

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	✓		✓
Queues and topics	Azure Service Bus	✓		✓
Storage tables	Azure Storage		✓	✓
SQL tables	Azure Mobile Apps		✓	✓
NoSQL DB	Azure Cosmos DB	✓	✓	✓
Push Notifications	Azure Notification Hubs			✓
Twilio SMS Text	Twilio			✓
SendGrid email	SendGrid			✓
Excel tables	Microsoft Graph		✓	✓
OneDrive files	Microsoft Graph		✓	✓
Outlook email	Microsoft Graph			✓
Microsoft Graph events	Microsoft Graph	✓	✓	✓
Auth tokens	Microsoft Graph		✓	

Demo
Programming Azure Functions with Visual Studio 2017



Getting to know your triggers

Triggers provide metadata

Different per trigger type

Access with {metadata} syntax in other attributes

```
[FunctionName("ValidateBlobSize")]
public static void Run(
  [QueueTrigger("ImagesInput",
  Connection = "…")] string blobNameInQueue,

  [Blob("images/{queueTrigger}",
  FileAccess.Read,
  Connection = "…")] Stream blob,

  TraceWriter log) { // Your function code …
}
```

Example: Queue trigger

QueueTrigger
triggering message content if a valid string

DequeueCount

Number

ExpirationTime

Id

InsertionTime

NextVisibleTime

PopReceipt

Microsoft.Net.Sdk.Functions

Composes publishable output

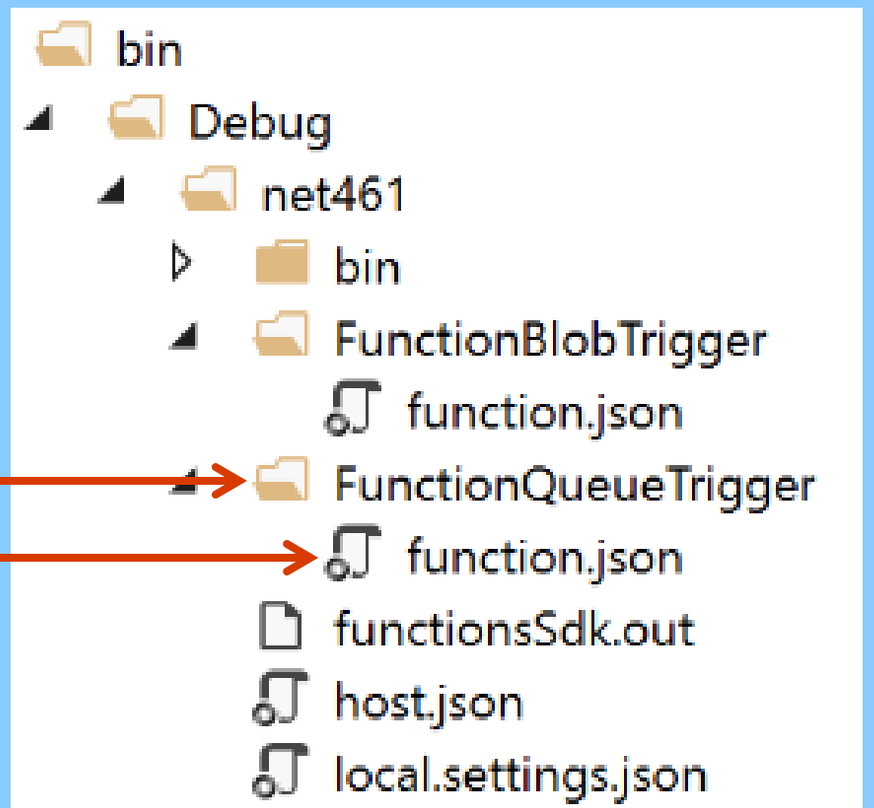
Inside build target (e.g. net461 or netstandard2.0)

Deviates from standard build output location

Translates C# attributes metadata

Each function produces one function.json file

```
[FunctionName("FunctionQueueTrigger")]
public static void Run(
    [QueueTrigger("imagesinput", Connection = "…")]string blobNameInQueue,
    [Blob("images/{queueTrigger}", FileAccess.Read, Connection = "…")]Stream blob,
    [Queue("imagestoolarge", Connection = "…")]ICollector<string> imagesTooLarge,
    TraceWriter log)
```



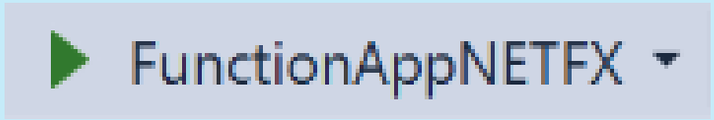
NuGet metadata package

Debugging and testing Azure Functions

Debugging

1 Visual Studio 2017/Code

Press F5 to
start debugger



2 Manually CLI hosted

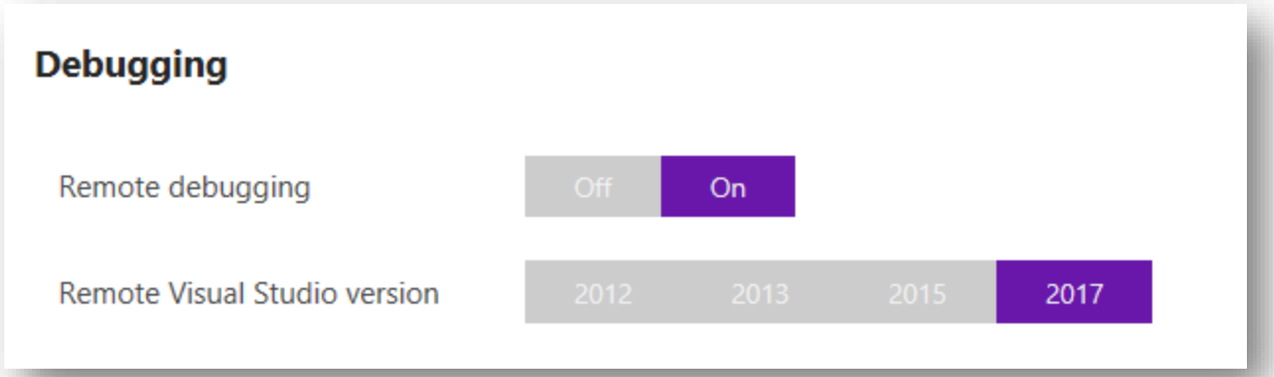
```
func host start
   --debug vs
```

or JavaScript in VSCode

```
   --debug vscode
```

Configure function app for remote debugging

Allows Visual Studio to be attached to Azure hosted app



Running and quick testing

Azure Portal for quick testing

Administrative endpoints available

http://localhost:{port}/admin/functions/{function_name}
Requires POST with JSON payload containing `{ "input" : "value" }`
Set Content-Type of request to application/json

Tools to trigger

Azure Storage Explorer

Trigger with queues and blob containers
Add items as input in queues, tables and blob
Check output in storage

Postman for HTTP

Execute locally (default port 7071)
http://localhost:7071/api/{function_name}/..
.

Unit testing your functions

Automate your test efforts

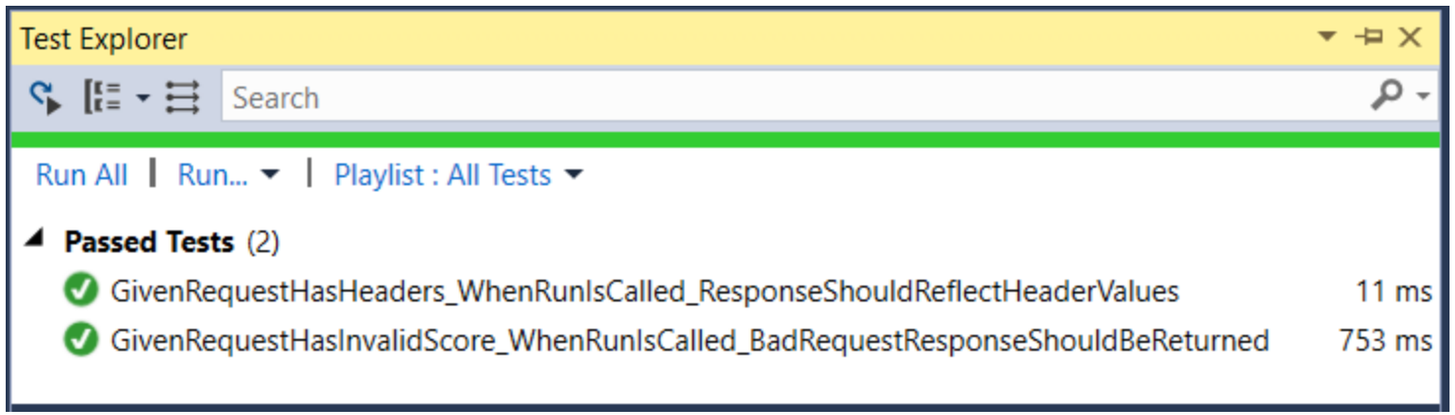
Isolate as much of your dependencies and logic

Use service locator design pattern to resolve dependencies where required

Include following NuGet packages in test project

Microsoft.NET.Test.Sdk

Microsoft.Azure.WebJobs.Extensions.* depending on your required bindings



Mocking and faking input and output

Stub or mock trigger, input and output objects

During unit testing Run binding attributes are irrelevant

Blob and queue

- Simply supply message content
- Depends on input type (`string`, `byte[]` or POCO class)

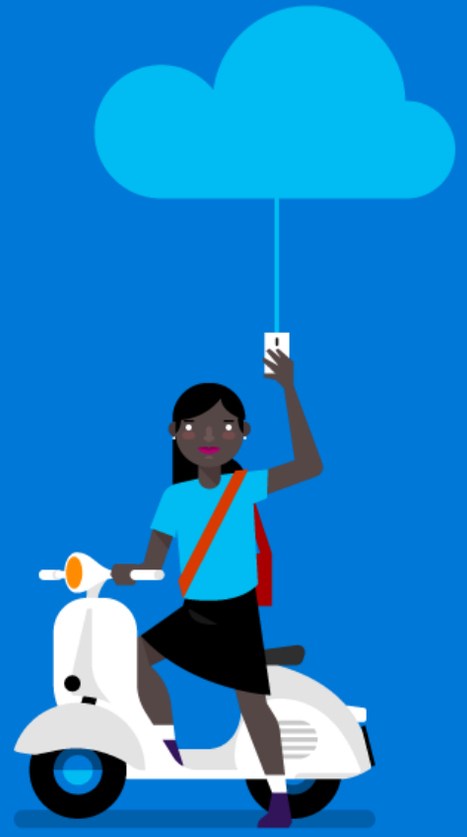
HttpRequestMessage

- Able to be constructed as dummy object
- Set properties for Body and Headers
- Similar for HttpResponse(Message)

TraceWriter

- Inherit and override Write method for mock
- Alternatively use ILogger provided through DI

Demo
Debugging and unit testing Azure Functions



Build scenarios

1 Connect to code repo

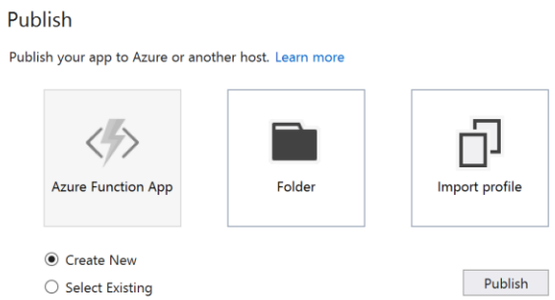
Suitable for dynamic workflow without VS2017

2 CI/CD pipeline in VSTS

High-quality build process with testing

Don't publish from VS or CLI

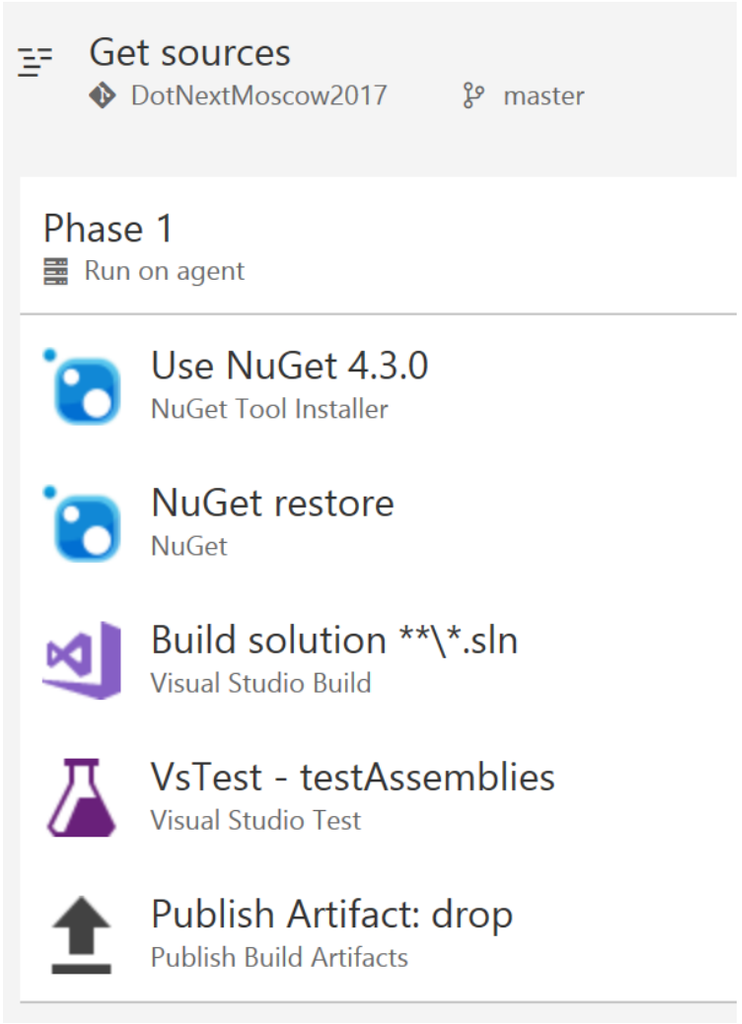
Only for quick starts and testing purposes



1

- Visual Studio Team Services
  By Microsoft
- OneDrive
  By Microsoft
- Local Git Repository
  By Git
- GitHub
  By GitHub
- Bitbucket
  By Atlassian
- Dropbox
  By Dropbox
- External Repository

2 Building Azure Functions with VSTS



Easy build pipeline

Choose ASP.NET Core (.NET Framework) template

Build targets do heavy lifting and package artefacts in ZIP file

Deployment can be part of build

Add Azure App Service Deploy task

Package to deploy: $(Build.artifactstagingdirectory)/**/*.zip

Deploying .NET compiled apps

`generatedBy` attribute will put app in ReadOnly mode

Need to take care of app settings

Working with settings

Hosted in Azure

Application settings separate

Optionally publish your local settings

```
func azure functionapp publish
  DotNextApp
  --publish-local-settings
  --overwrite-settings
```

Programmatic access

Via environment variables

```
Environment.GetEnvironmentVariable("")
```

%setting_name% for use in bindings

Local development

local.settings.json file

Connection strings separate section

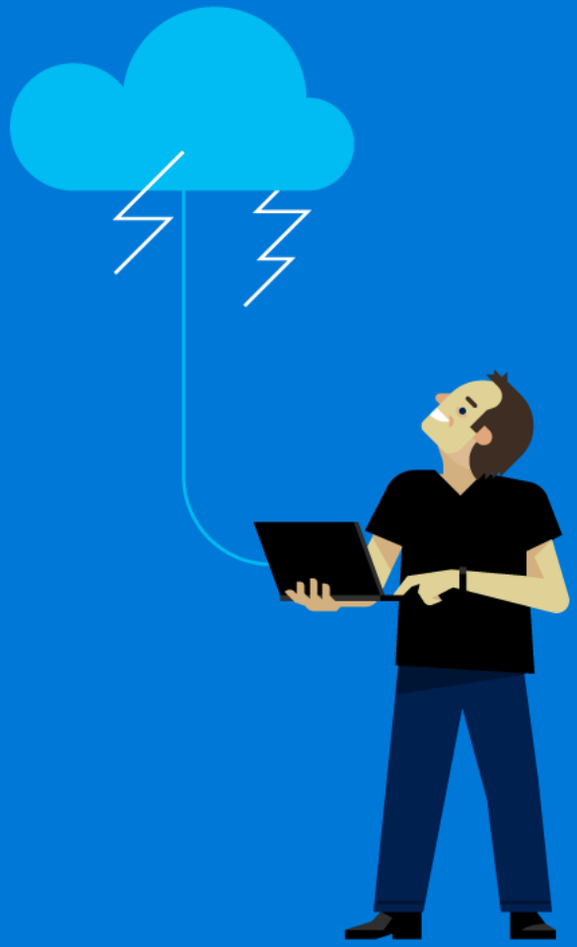
```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "",
    "key1": "value1"
  },
  "ConnectionStrings": {
    "db": {
      "ConnectionString": "...",
      "ProviderName": "System.Data.SqlClient"
    }
  }
}
```

local.settings.json file

Functions CLI

Add, delete, encrypt, decrypt and list

Demo
Deploying Azure Functions



GENERAL SETTINGS
- Function app 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
- API definition
- CORS

APP SERVICE PLAN
- App Service plan
- Quotas

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

Hosting options

Azure

Consumption-based

Scaling as needed

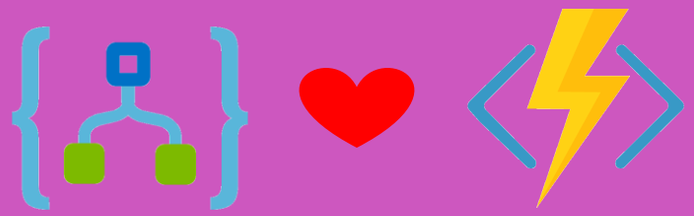
Might require time to provide compute instances

App Service plan

Available scale

Not so much server-less

Easy to combine with PaaS



On-premises

Azure Functions Runtime

Local installation of hosts

Connected via SQL Server database

Infrastructure operations

Not so much server-less

Instrumentation and monitoring

1. Simple monitoring information from portal

App Insights is enabled for your function. Open Application Insights

Success count since Oct 1st

31

Error count since Oct 1st

0

Invocation log Refresh live event stream

Function	Status	Details: Last ran	(duration)
HttpTriggerFunction (Method: POST, Uri: ...)	✔	2 hours ago	(0 ms)
HttpTriggerFunction (Method: POST, Uri: ...)	✔	2 hours ago	(0 ms)
HttpTriggerFunction (Method: POST, Uri: ...)	✔	2 hours ago	(0 ms)

Invocation details

Parameter

2017-10-30T10:08:15.335	HttpTriggerFunction Success Rate	100	1	100	100	100	0	{"LogLev
2017-10-30T10:08:15.335	HttpTriggerFunction Count	4	1	4	4	4	0	{"LogLev
2017-10-30T10:08:15.335	HttpTriggerFunction Failures	0	1	0	0	0	0	{"LogLev
2017-10-30T10:08:15.335	HttpTriggerFunction Duration	22.0566	4	22.0566	0.5558	20.0909	0	{"LogLev
2017-10-30T10:08:15.335	HttpTriggerFunction Successes	4	1	4	4	4	0	{"LogLev

Azure Functions with .NET Core 2.0

Based on .NET Standard 2.0

Requires Azure Functions Core tools 2.0

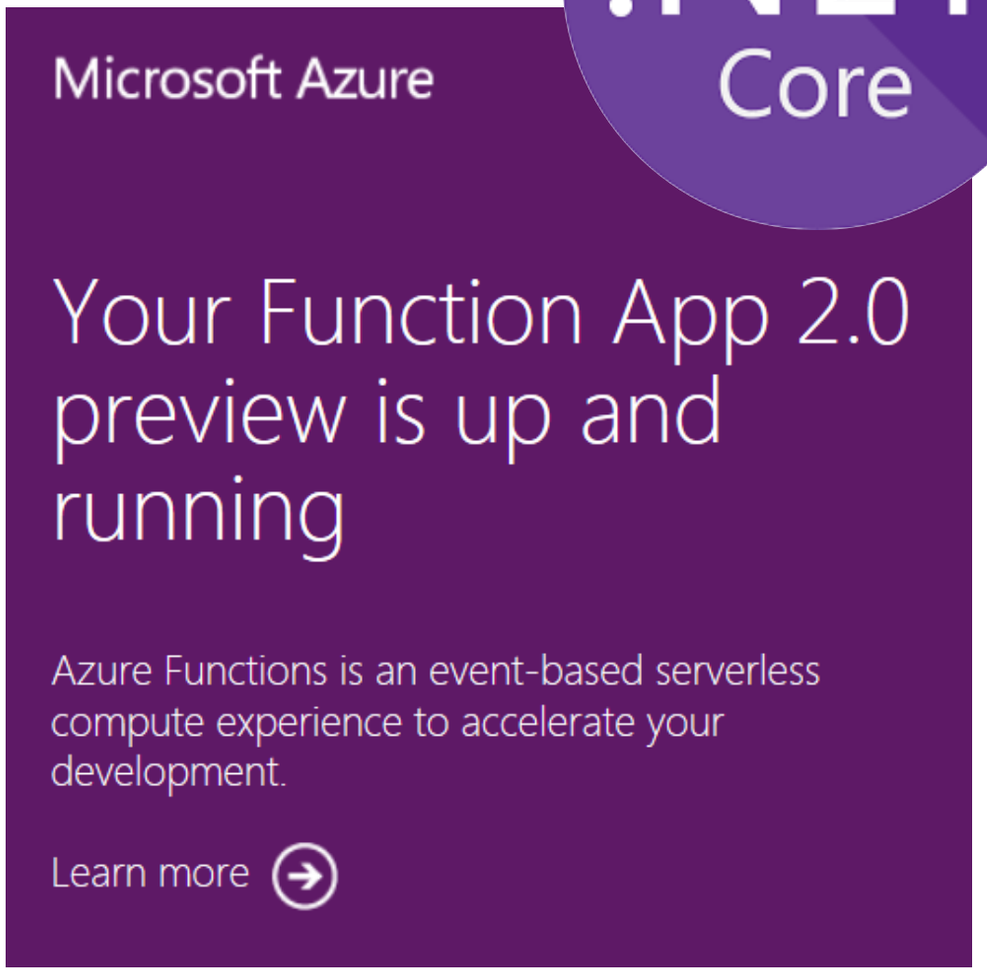
```
npm install -g azure-functions-core-tools@core
```

Switch to version 2.x runtime

Set application setting
`FUNCTIONS_EXTENSION_VERSION=beta`

When released will change to ~2

Still in beta



Functions best practices

Single responsibility principle

Do one thing focussed

Finish as quickly as possible

Avoid long running functions

Stateless, idempotent and defensive

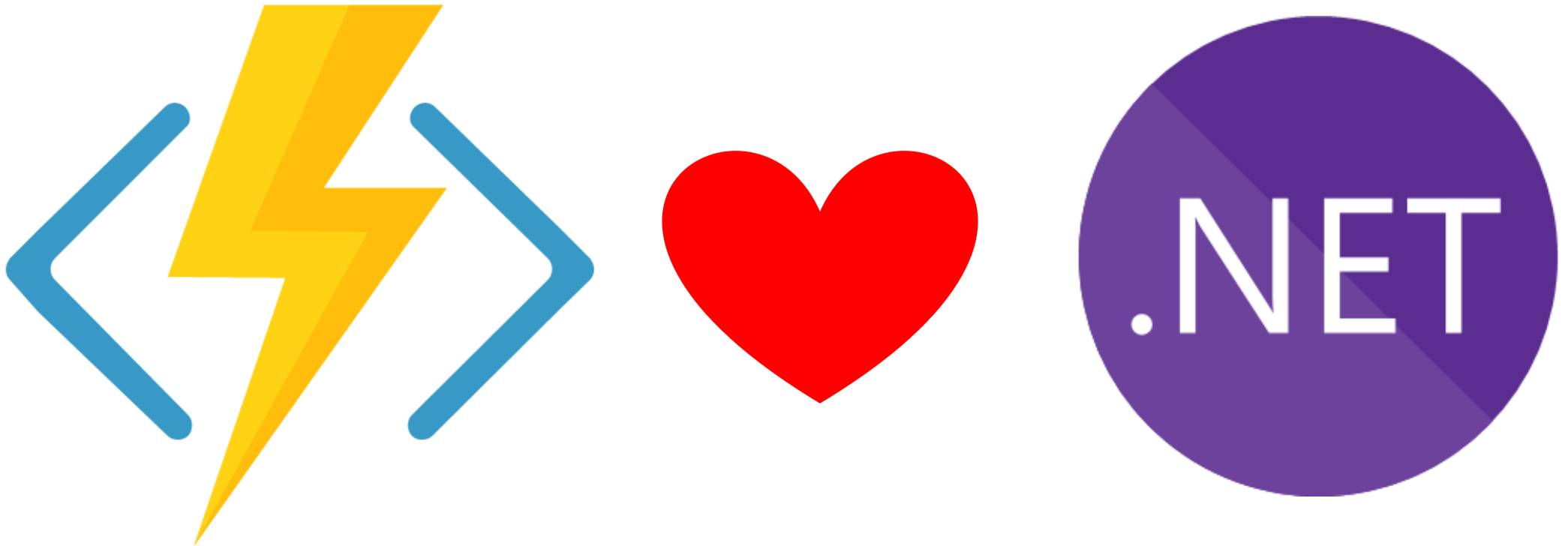
Cross function communication

Messaging: Storage Queues (small <64 KB) or Service Bus (<256 KB)

Service Bus topics for message filtering

Event Hub for high volume messaging

Summary



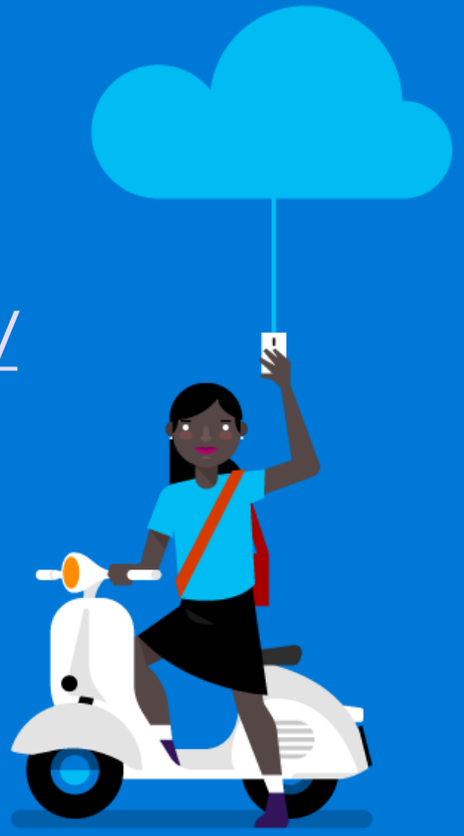
Resources

Read

https://aka.ms/tryfunctions
https://functions.azure.com
https://docs.microsoft.com/en-us/azure/azure-functions/
https://github.com/Azure/Azure-Functions

Contact @AzureFunctions

Watch Function Junction videos on Channel9



Security

3 levels of security for functions

1. Function
2. Anonymous
3. Admin

Special auth endpoint

{func_app}.azurewebsites.net/.auth/login/aad?prompt-consent