SELA

SELA|DEVELOPER|PRACTICE
JULY 3-5, 2018

Workshop
Server-less .NET solutions using Azure Functions

Alex Thissen
Cloud architect at Xpirit, The Netherlands
@alexthissen

Time table

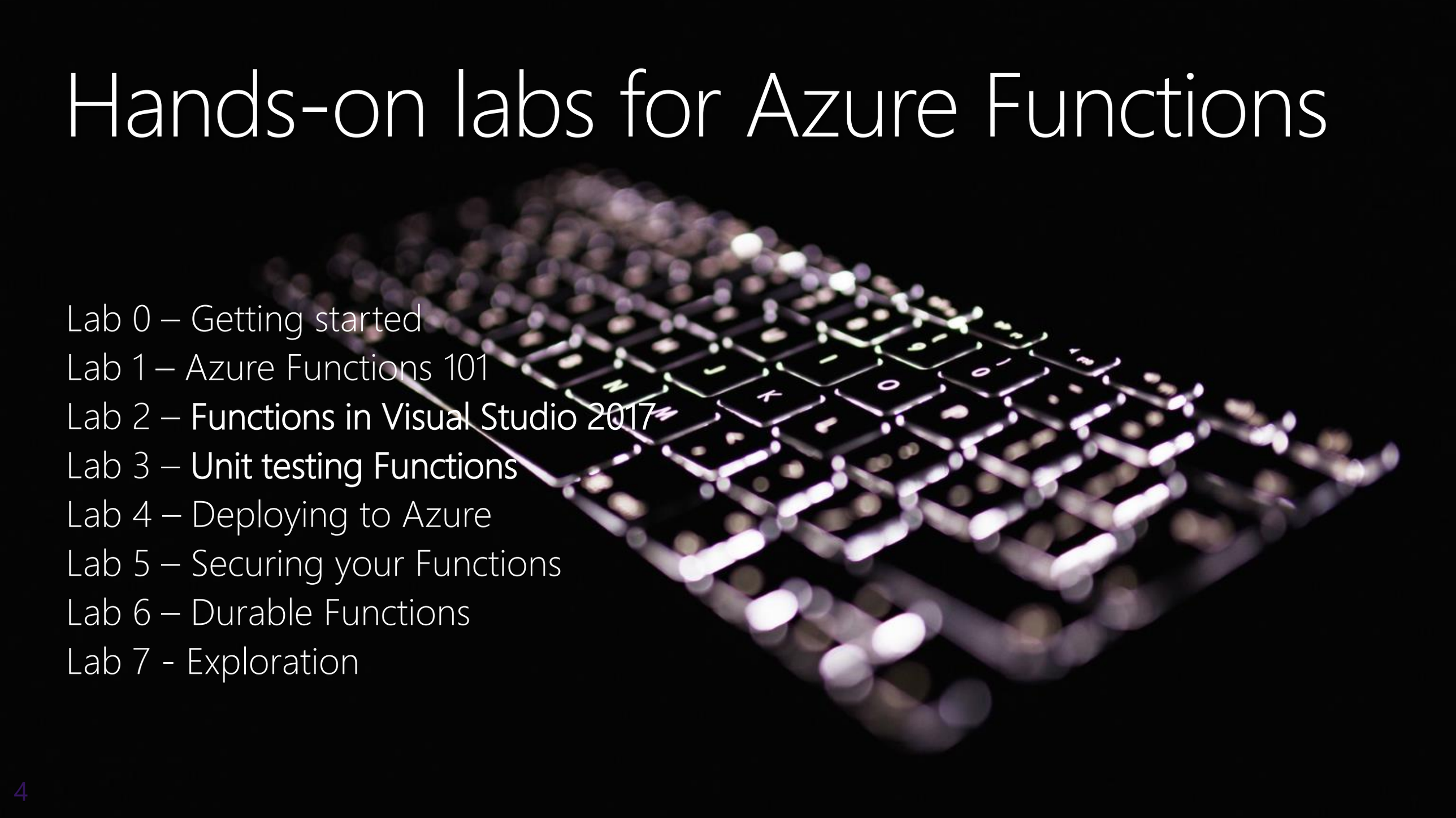
10:10 – 10:25	Introduction
10:25 – 11:30	Theory and hands-on
11:30 – 11:50	Networking break
11:50 – 13:25	Theory and hands-on
13:25 – 14:10	Lunch
14:10 – 15:35	Theory and hands-on
15:35 – 15:50	Networking break
15:50 – 16:30	Session and wrapup

Alex Thissen

Cloud Architect
Xpirit Netherlands

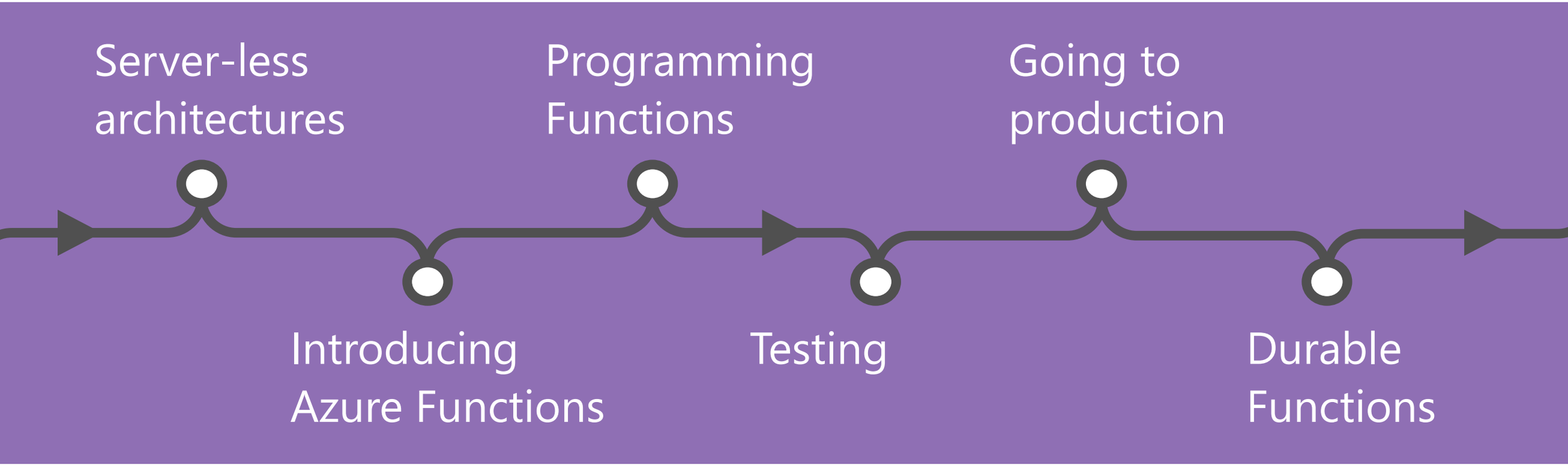
@alexthissen – athissen@xpirit.com

Hands-on labs for Azure Functions



Lab 0 – Getting started
Lab 1 – Azure Functions 101
Lab 2 – **Functions in Visual Studio 2017**
Lab 3 – **Unit testing Functions**
Lab 4 – Deploying to Azure
Lab 5 – Securing your Functions
Lab 6 – Durable Functions
Lab 7 - Exploration

Presentation agenda



Server-less architectures and Azure Functions

Clear skies shouldn't cost much

Based on @Dougward
during Global Azure Bootcamp 2018

Pay only for the lightning bolts

Handle a lot of lightning bolts

Back to sunshine after the storm

Transitioning to server-less

There are no servers…

When you do not need them

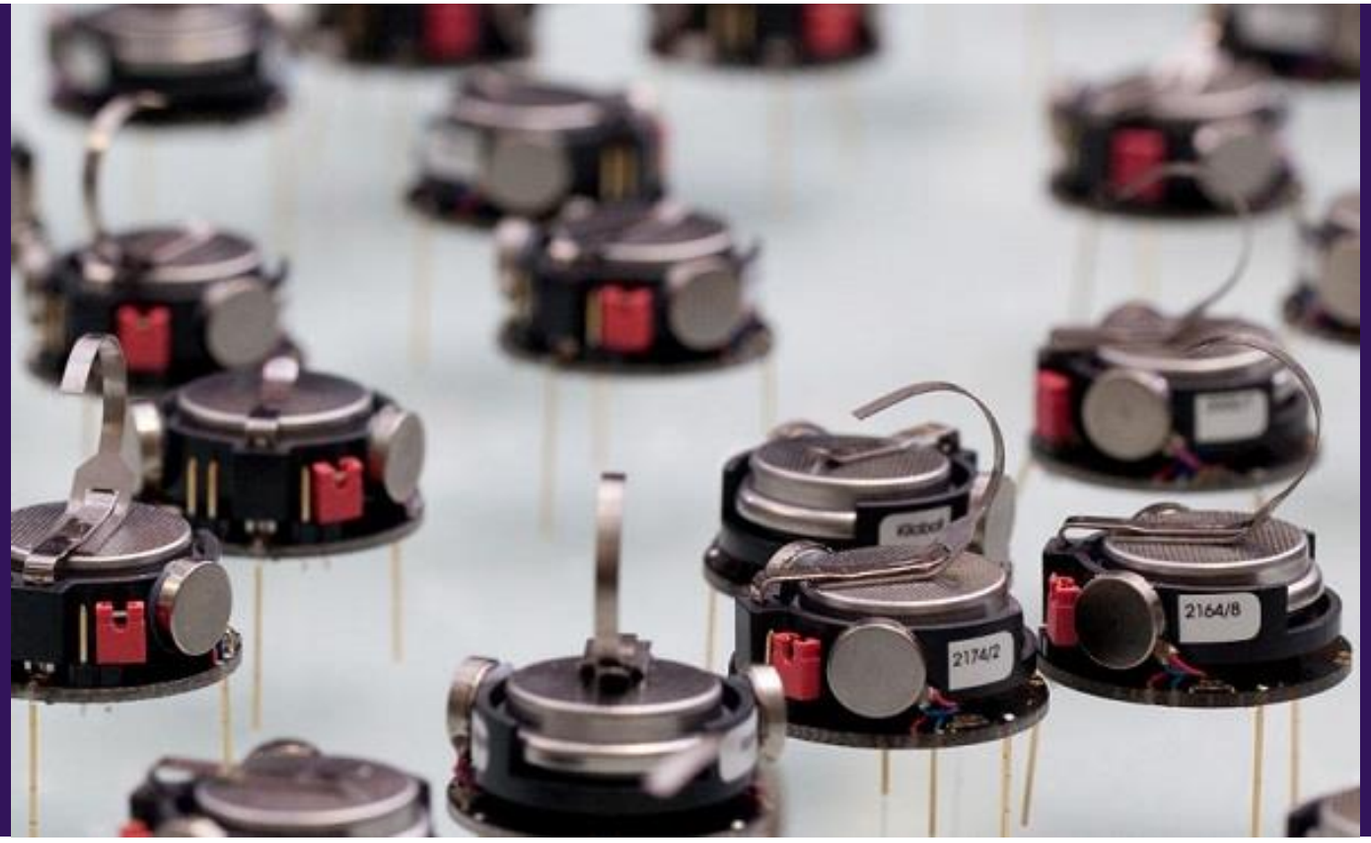
Transitioning to server-less

Available when there is work to do

Without you having to manage or control 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 GB-s and # of executions

Server-less platform providers

Major cloud provider offer FaaS

New competitors enter competition

Amazon Lambda
(since 2014)

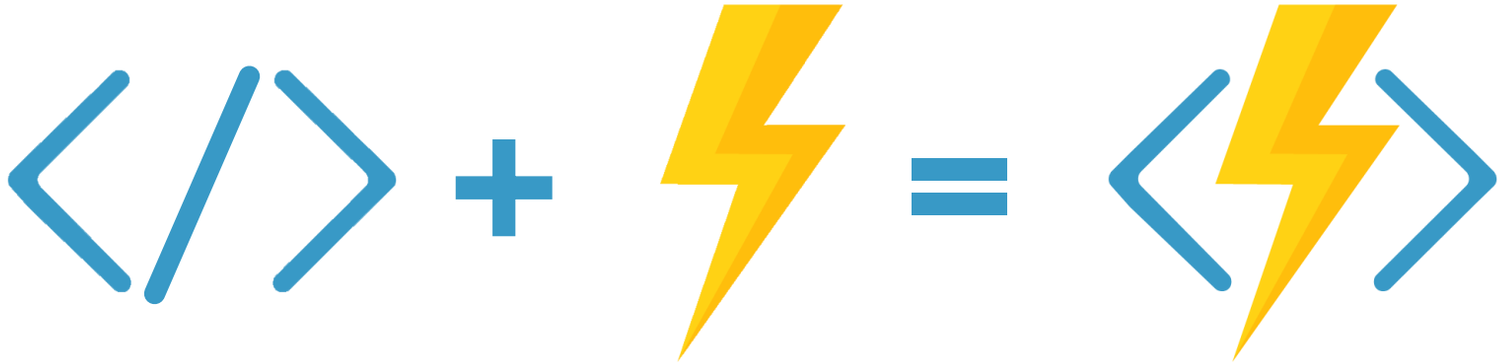
Google Cloud Functions
(since 2016)

Azure Functions
(since 2016)

Auth0 Webtask.IO
(since 2016)

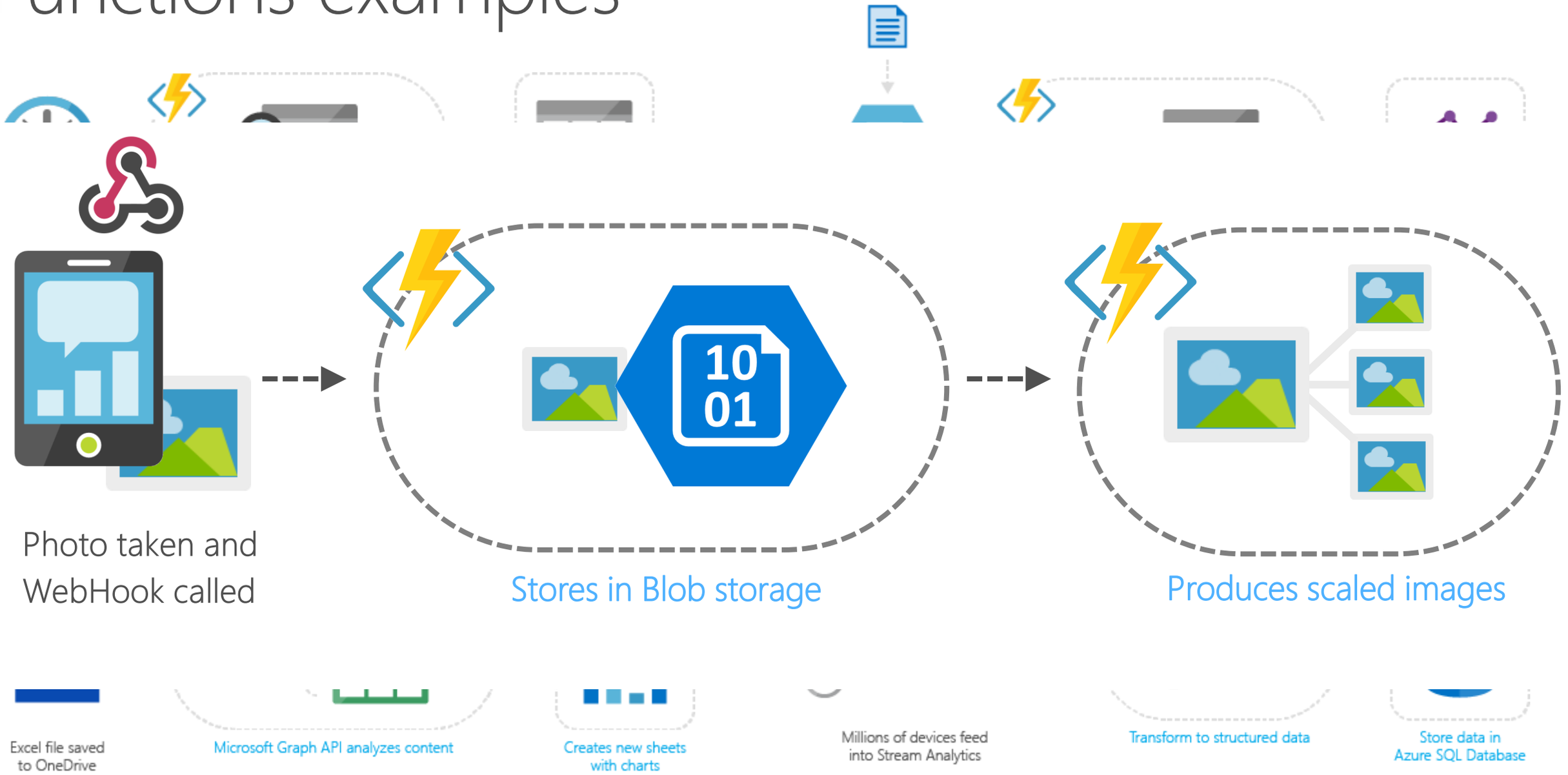
Focusing on Azure Functions

Code + Events + data = Azure Functions



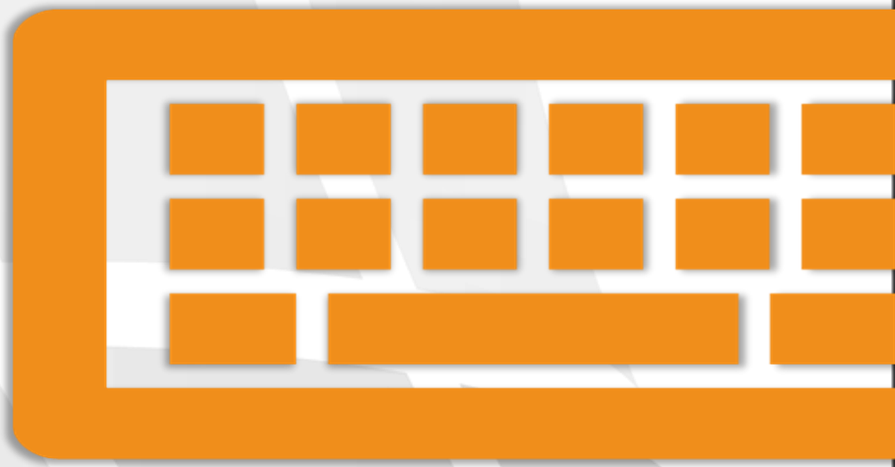
Process events with server-less code

Functions examples

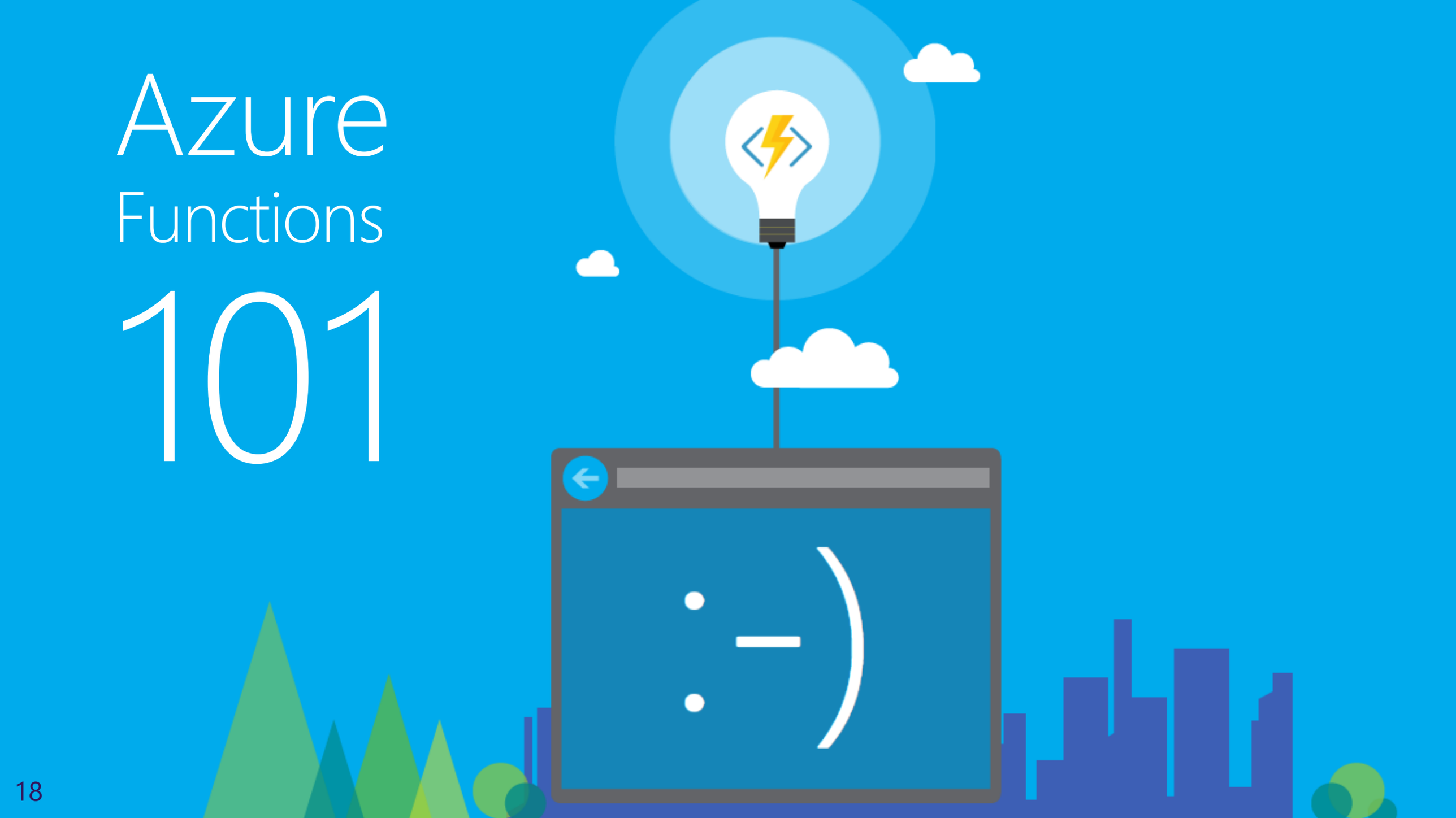


Lab 1 – Azure Functions 101

Lab



Azure Functions 101



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

About Azure Functions

Open source

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

Many available languages

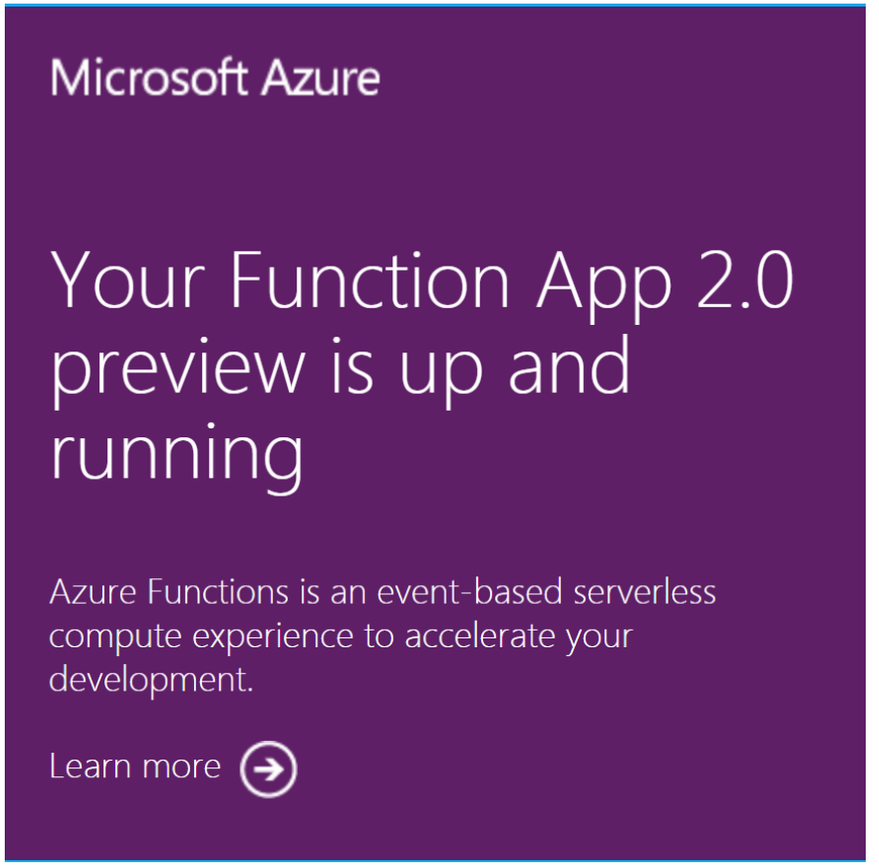
C#, JavaScript, F#, Java, PowerShell, Python, PHP, Batch, Bash

Built on top of Azure WebJobs

Support for .NET:

.NET Framework 4.6.1+

.NET Core 2.0 (in preview)



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, …

Anatomy of an Azure Function App

Function apps

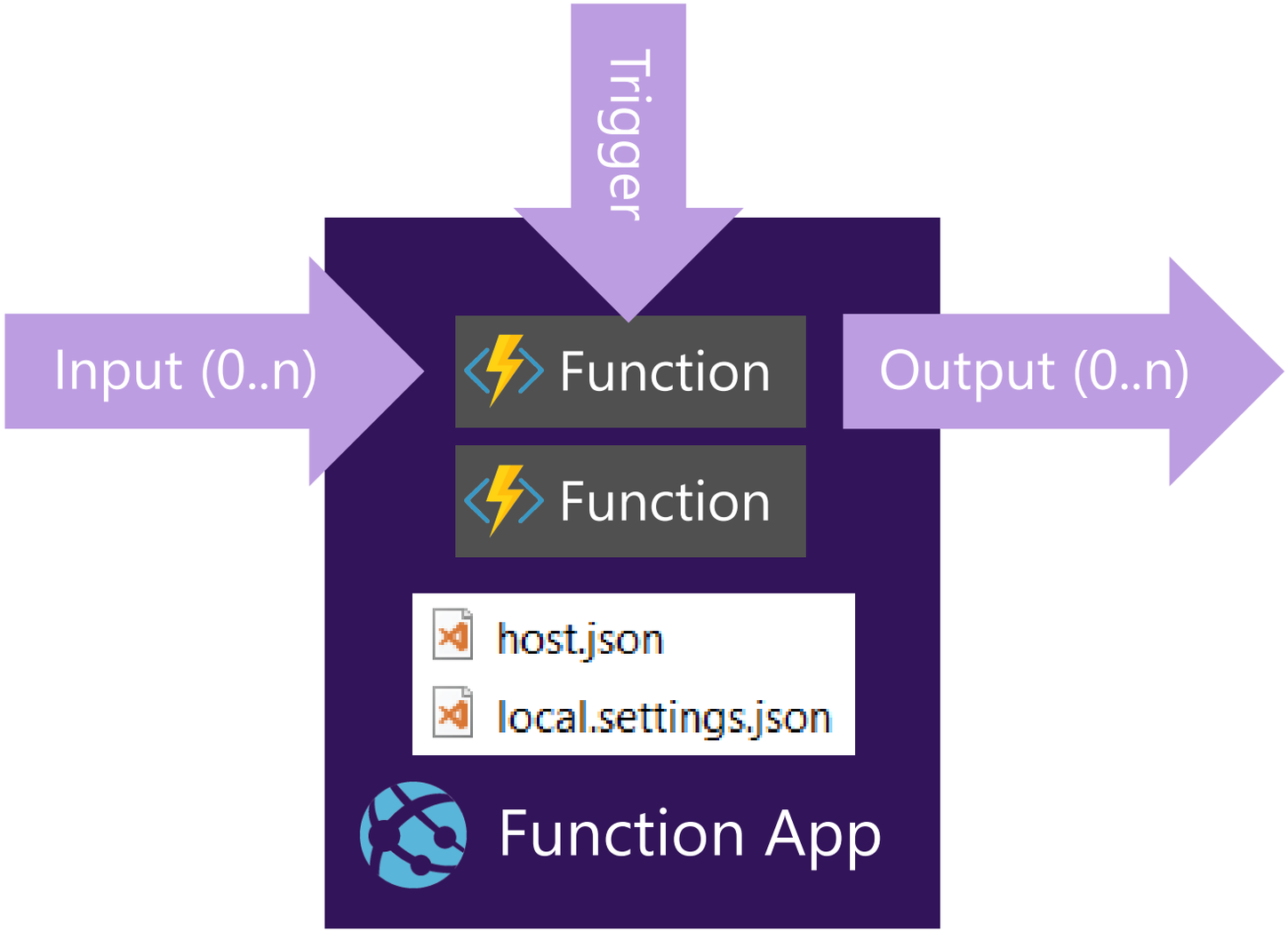
- Hosted as Azure App Service
- JSON based configuration
- Running (multiple) functions

Trigger starts execution

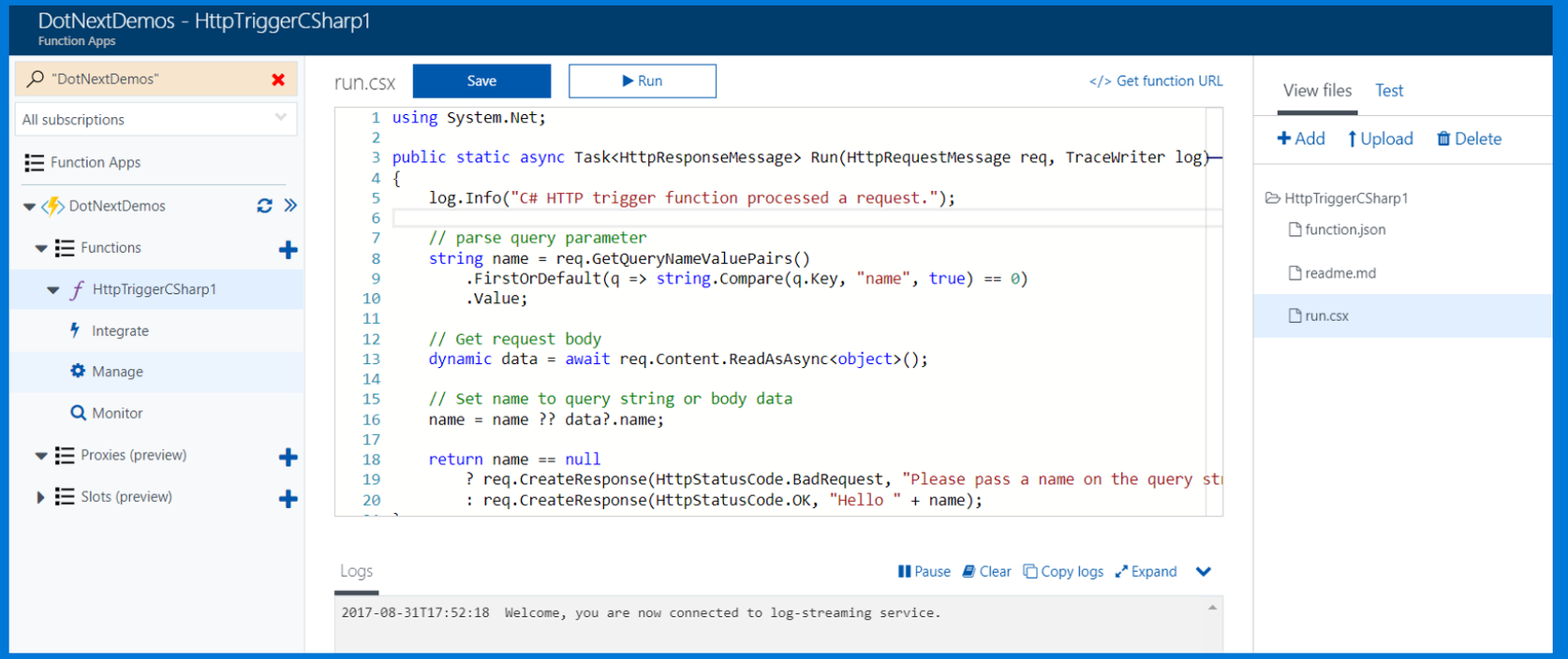
Bindings for input and output

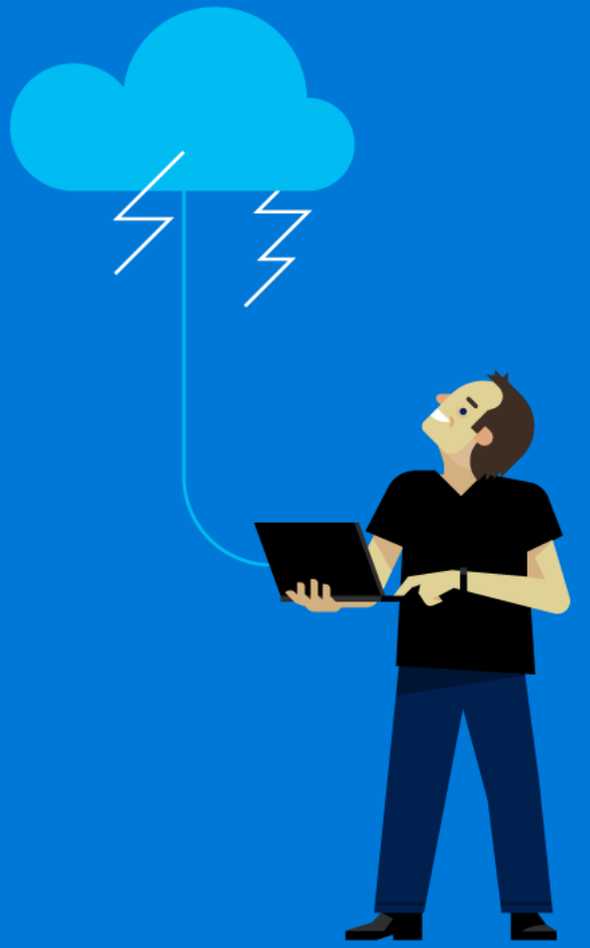
Zero or more possible

Triggers and bindings can vary per function



Demo
Creating and using Azure Functions





Azure Functions with .NET Core 2.0

Based on .NET Standard 2.0

Requires Azure Functions Core tools 2.0

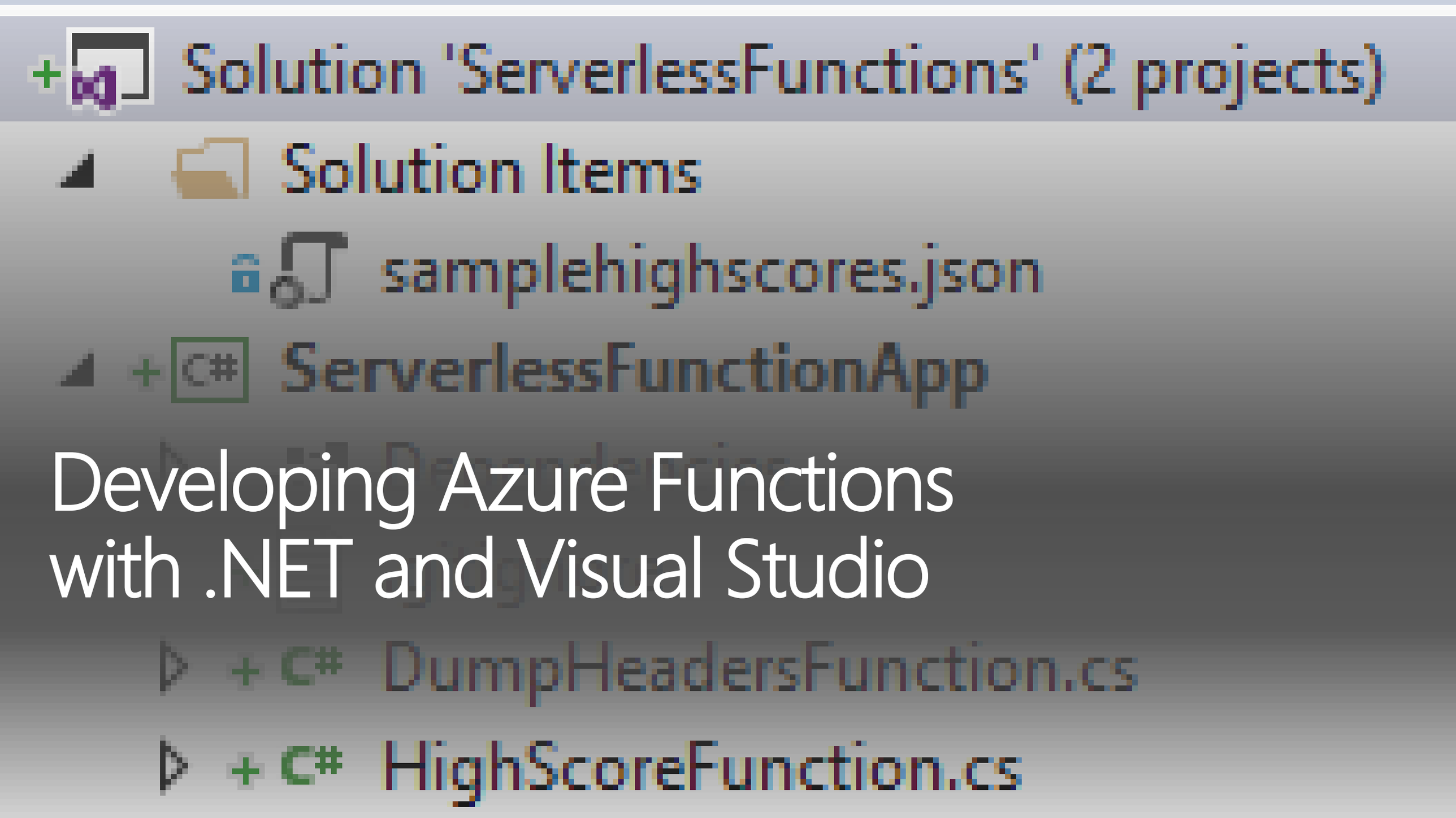
```
npm install -g azure-functions-core-tools@core
```

Still in beta

Switch to version 2.x runtime

Set application setting
`FUNCTIONS_EXTENSION_VERSION=beta`

When released will likely change to ~2



Developing Azure Functions with .NET and Visual Studio

Getting started with local development

Required tooling

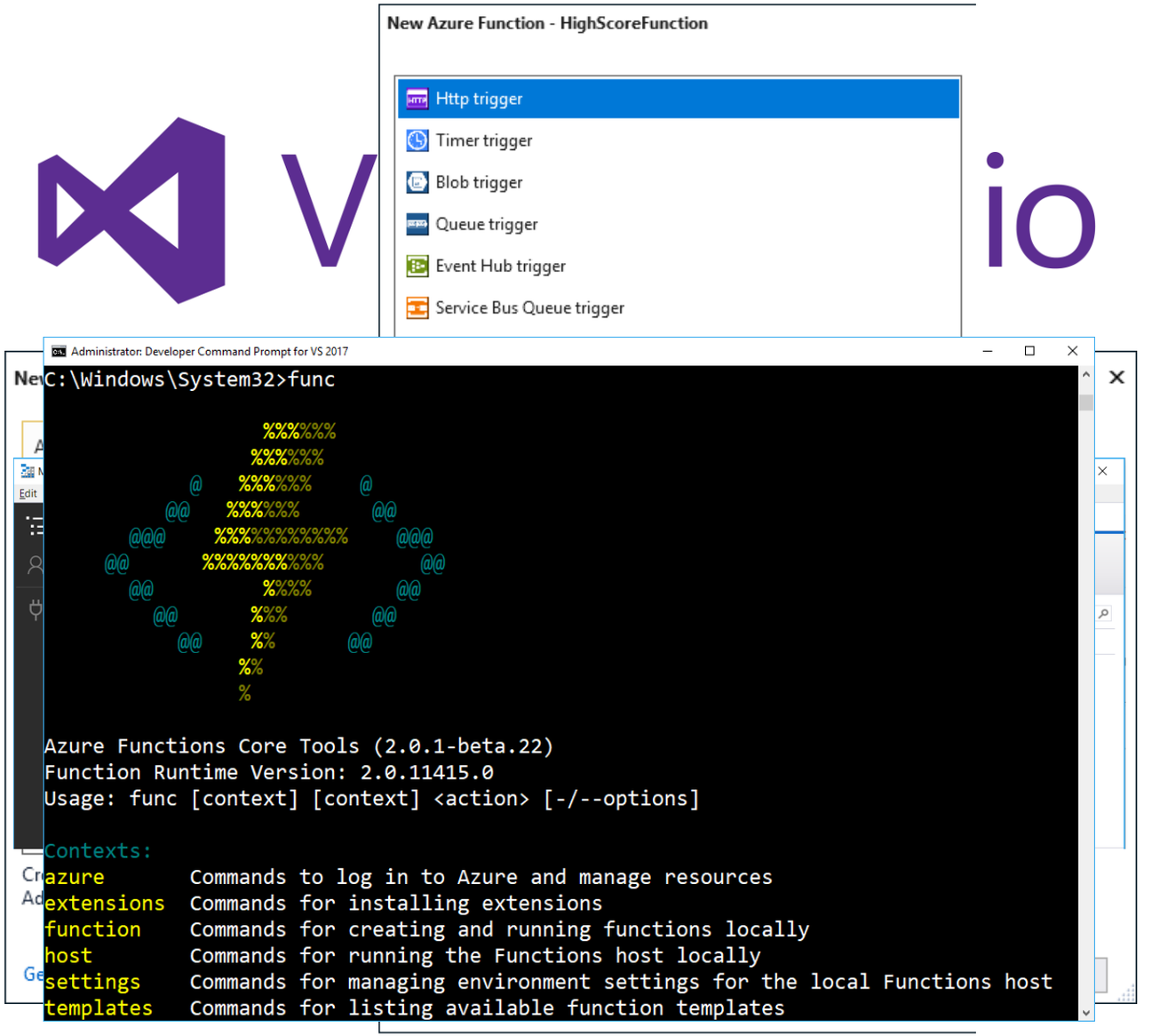
Visual Studio 2017 15.3 + Azure workload
Visual Studio Tools for Azure Functions
Azure Functions Core Tools (includes CLI)
.NET Core 2.0 installation (2.0 preview)

Azure Storage Explorer

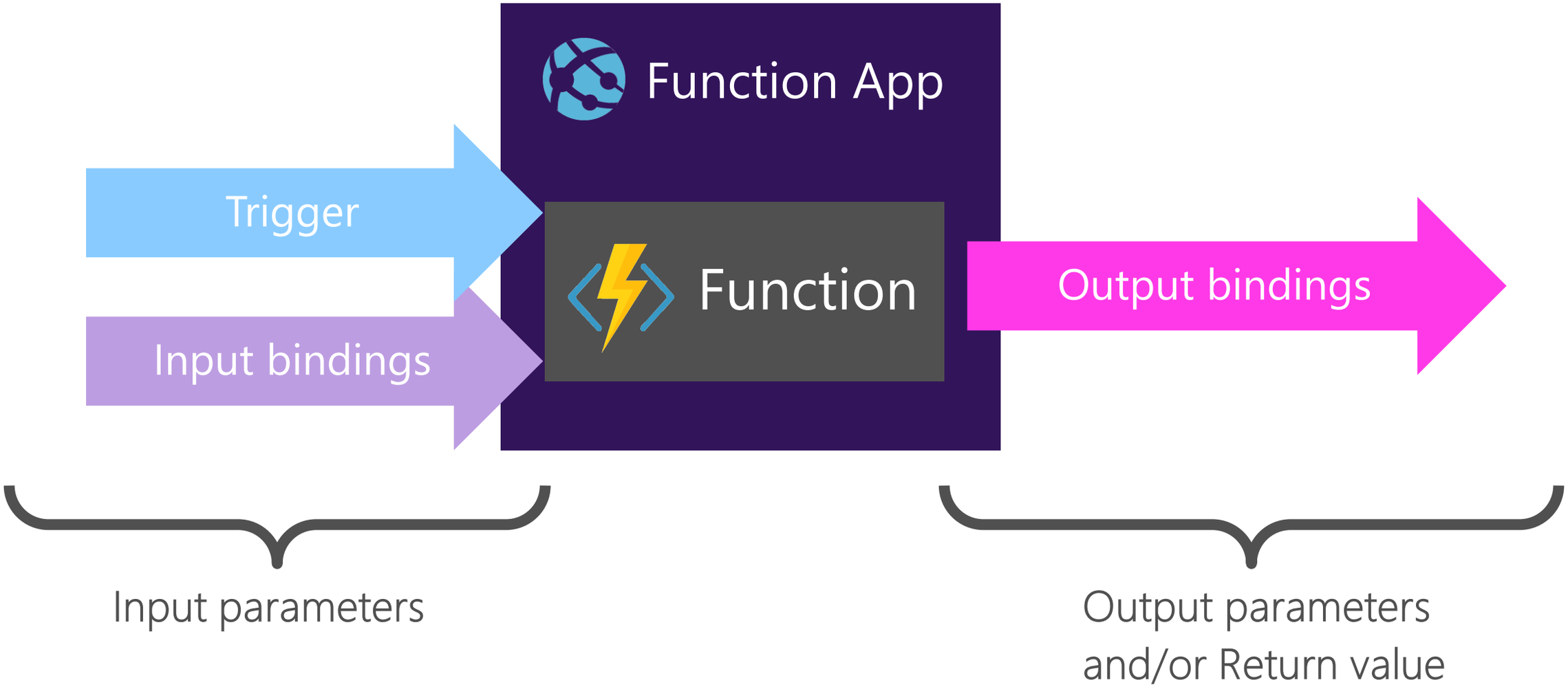
For interacting with Azure and local emulator

Useful optionals

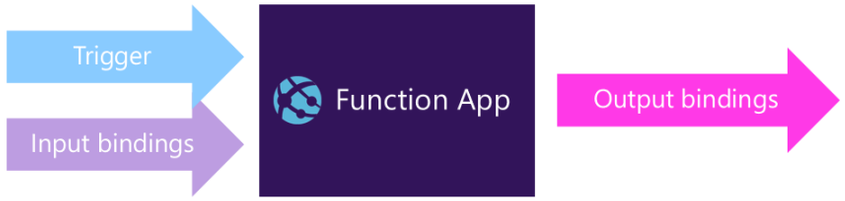
HTTP requests: Postman or SoapUI



Programming model: It's a function



Simple triggers and bindings



Define function.json metadata with attributes

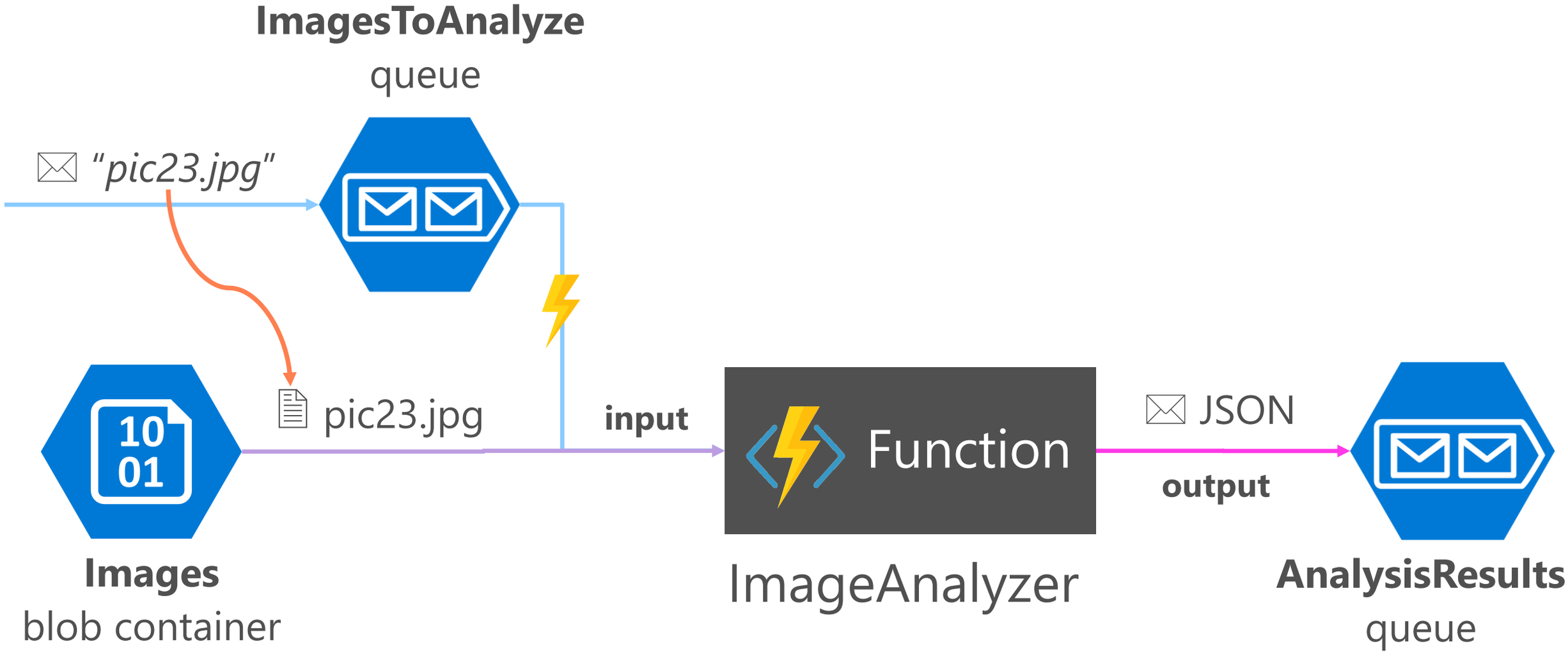
```
[FunctionName("HelloWorldFunction")]
public static HttpResponseMessage Run(
    [HttpTrigger] HttpRequestMessage req,
    TraceWriter log)
{
    log.Info("C# HTTP trigger function processed a request.");

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

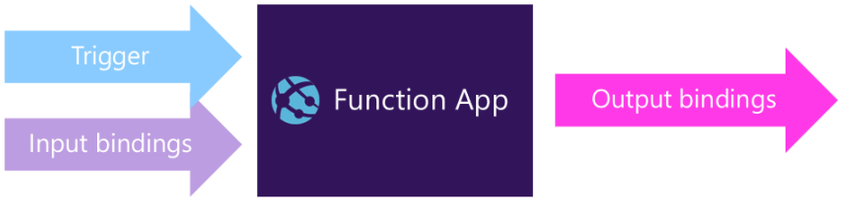
Return value
Single output binding can be done with return type

Trigger source
Defines what this function is triggered by

Triggers, input and output scenario



Abstractions over bindings



Define function.json metadata with attributes

```
[FunctionName("ImageAnalyzer")]
[return: Queue("AnalysisResults", Connection="…")]
public static AnalysisScore 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 application settings

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

Trigger and bindings

Abstraction of details

Semantics, conversion and plumbing

Growing list of supported types

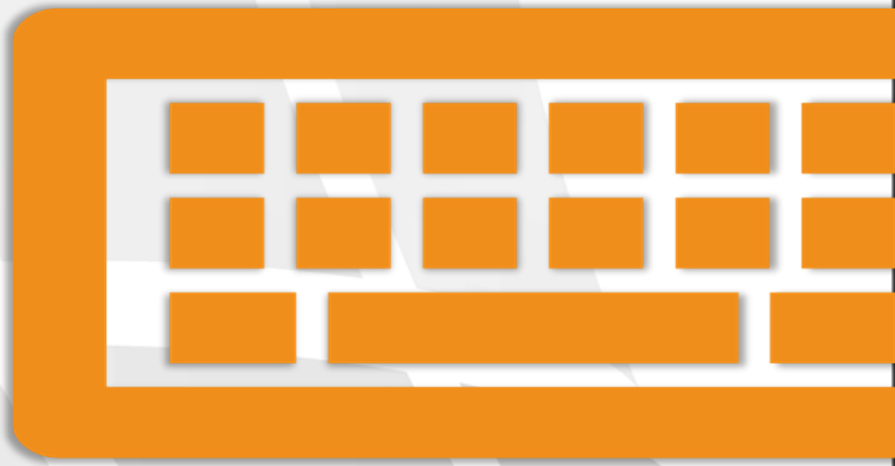
Azure resources (e.g. EventGrid and CosmosDB)

Timers and HTTP WebHooks
Office365 Graph and OneDrive
Email and SMS
Custom bindings

Type	1.x	2.x	Trigger	Input	Output
Blob Storage	✓	✓[1]	✓	✓	✓
Cosmos DB	✓	✓	✓	✓	✓
Event Grid	✓	✓	✓		
Event Hubs	✓	✓	✓		✓
External File[2]	✓			✓	✓
External Table[2]	✓			✓	✓
HTTP	✓	✓[1]	✓		✓
Microsoft Graph Excel tables		✓		✓	✓
Microsoft Graph OneDrive files		✓		✓	✓
Microsoft Graph Outlook email		✓			✓
Microsoft Graph Events		✓	✓	✓	✓
Microsoft Graph Auth tokens		✓		✓	
Mobile Apps	✓	✓		✓	✓
Notification Hubs	✓				✓
Queue storage	✓	✓[1]	✓		✓
SendGrid	✓	✓			✓
Service Bus	✓	✓	✓		✓
Table storage	✓	✓[1]		✓	✓
Timer	✓	✓	✓		
Twilio	✓	✓			✓
Webhooks	✓		✓		✓

Lab 2 – Programming Functions in VS2017

Lab



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 blobNameInMessage,

  int dequeueCount,

  [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

Azure WebJobs Cheat Sheet

Microsoft.Net.Sdk.Functions

Composes publishable output

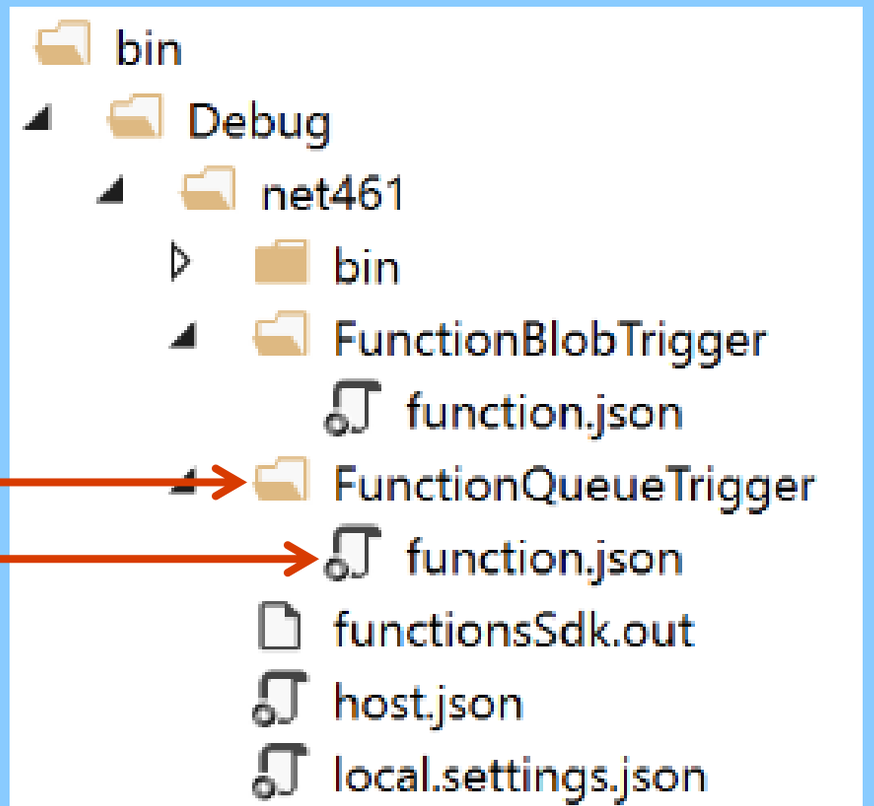
Inside build target (e.g. net471 or netstandard2.0)

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

Referenced dependencies, tooling, custom .targets files

Unit testing your functions

Automate your test efforts

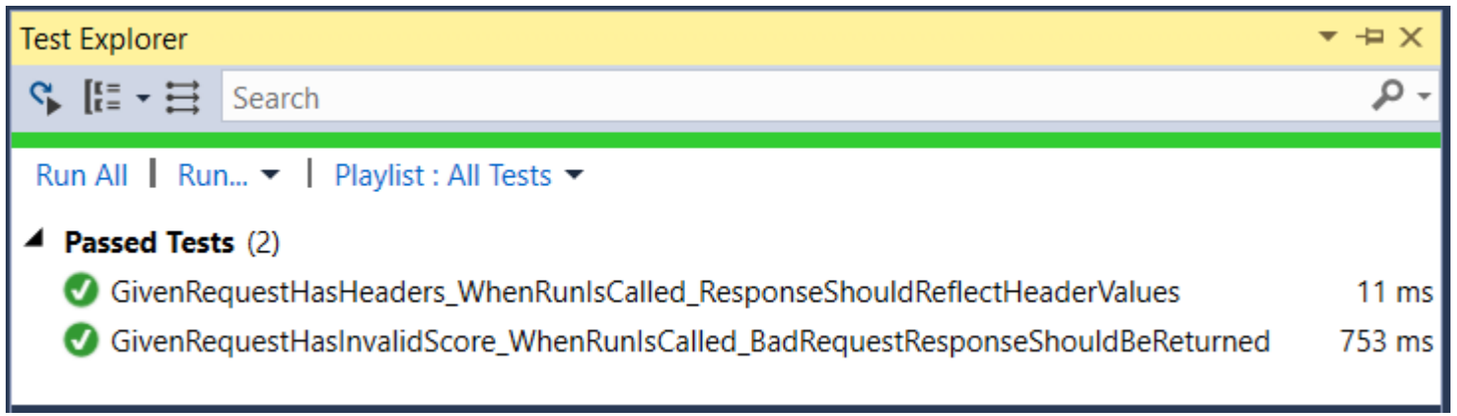
Isolate as much of your dependencies and logic

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)

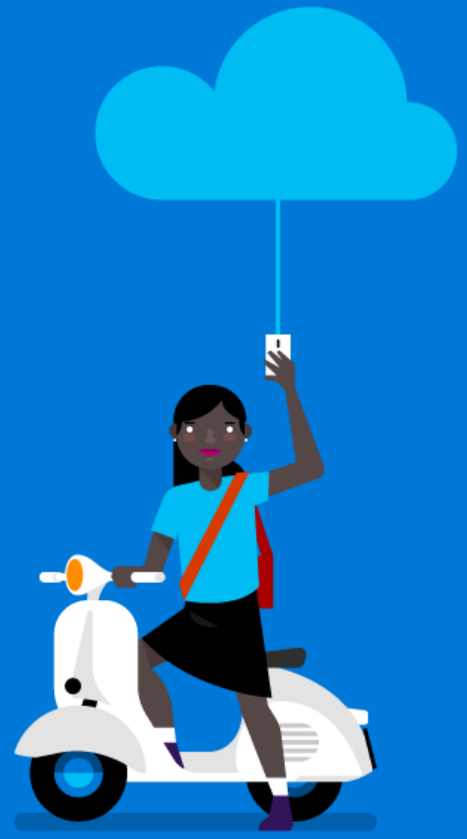
HttpRequestMessage

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

TraceWriter

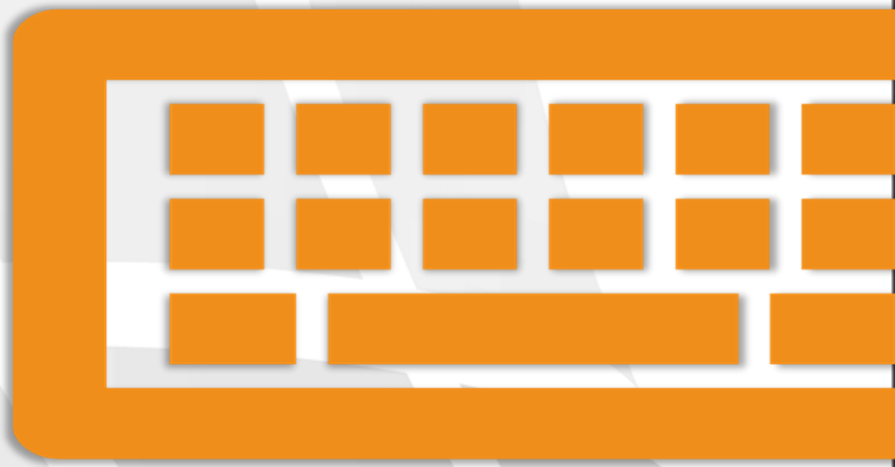
- Inherit and override Write method for mock
- Alternatively use **`ILogger`** provided through DI

Demo
Debugging and unit testing
Azure Functions



Lab 3 – Unit testing Azure Functions

Lab



Moving into production

Hosting options

Azure

Consumption-based

Scaling as needed

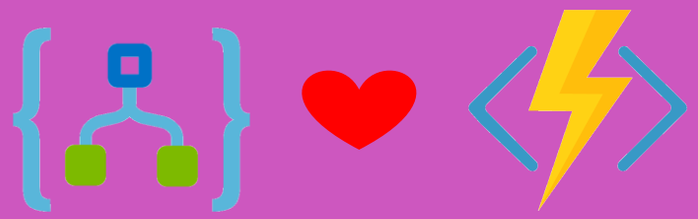
Might require time to provide compute instances

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

Build scenarios

1 Connect to code repo

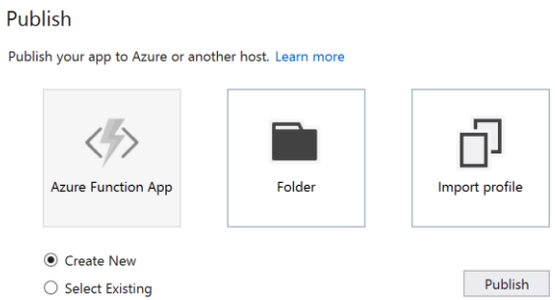
Suitable for dynamic workflow without VS2017

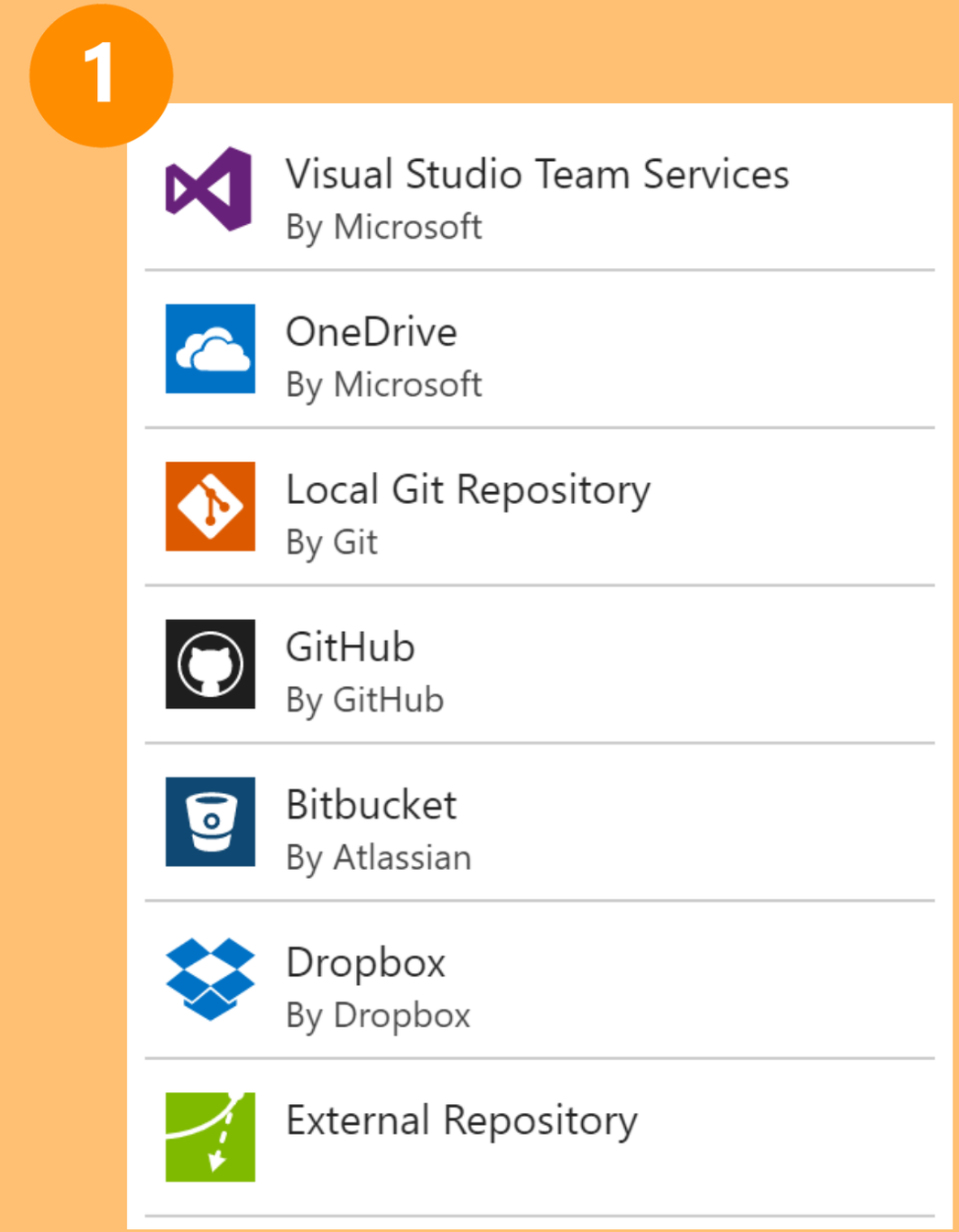
2 CI/CD pipeline in VSTS

High-quality build process with testing

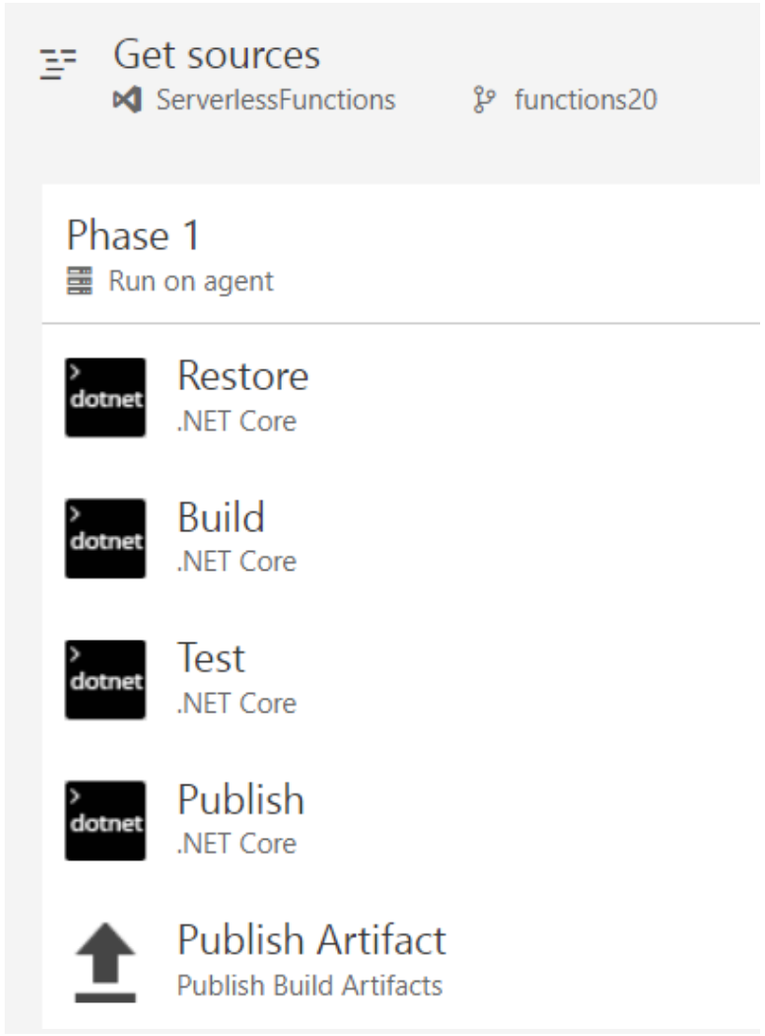
Don't publish from VS or CLI

Only for quick starts and testing purposes





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

PowerShell script

ARM templates

Working with settings

AppSettings in App Service

Deploy to Azure

- Using ARM template at creation
- PowerShell script to read and change
- Azure or Functions Tools CLI

Optionally publish your local settings

```
func azure functionapp publish FuncApp
  --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

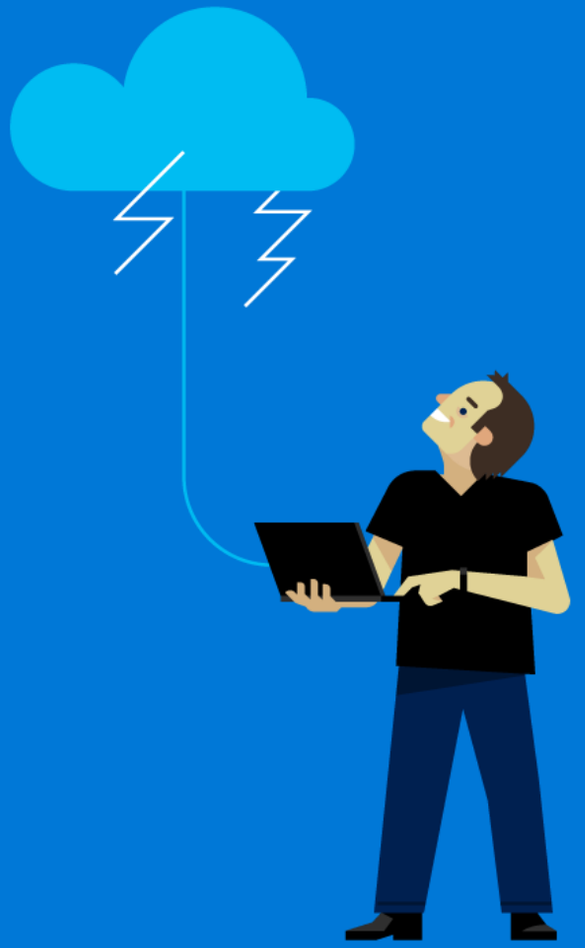
```
{
  "IsEncrypted": false,
  "Values": {
    "AzureWebJobsStorage": "",
    "key1": "value1"
  },
  "ConnectionStrings": {
    "db": {
      "ConnectionString": "...",
      "ProviderName": "System.Data.SqlClient"
    }
  }
}
```

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

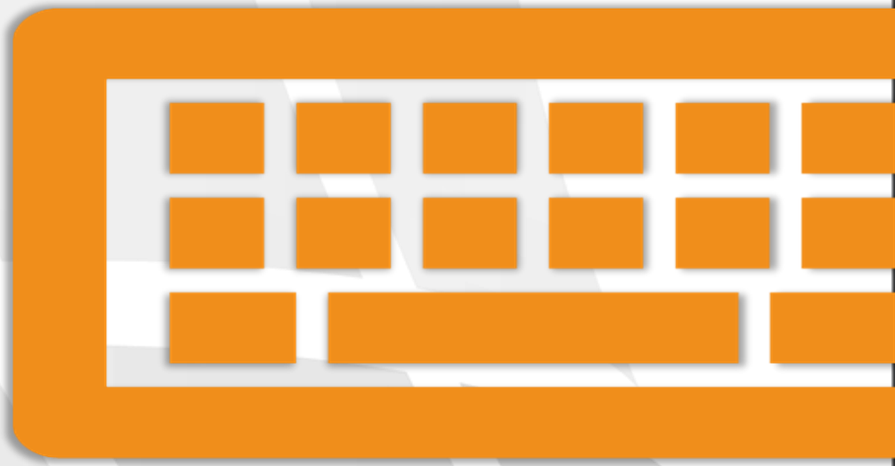
API
- API definition
- CORS

APP SERVICE PLAN
- App Service plan
- Quotas

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

Lab 4 – Deploying Function Apps to Azure

Lab



Instrumentation and monitoring

1. Simple monitoring information from portal

Refresh

Application Insights Instance	Success count in last 30 days	Error count in last 30 days	Query returned 1 items
NDCMinnesotaFunctions	1	0	Run in Application Insights

DATE (UTC)	SUCCESS	RESULT CODE	DURATION (MS)	OPERATION ID
2018-05-08 18:27:26.368	True	0	86.8796	8fbee1f8-7bb9-4541-b729-f3976a2e90a6

TABLE CHART Columns Display time (UTC+00:00)

Drag a column header and drop it here to group by that column

timestamp [UTC]	name	value	valueCount	valueSum	valueMin	valueMax	valueStdDev	customDim
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

Functions security

5 HTTP triggers security levels

Anonymous
Function
User
System
Admin

Testing tip: special auth endpoint

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

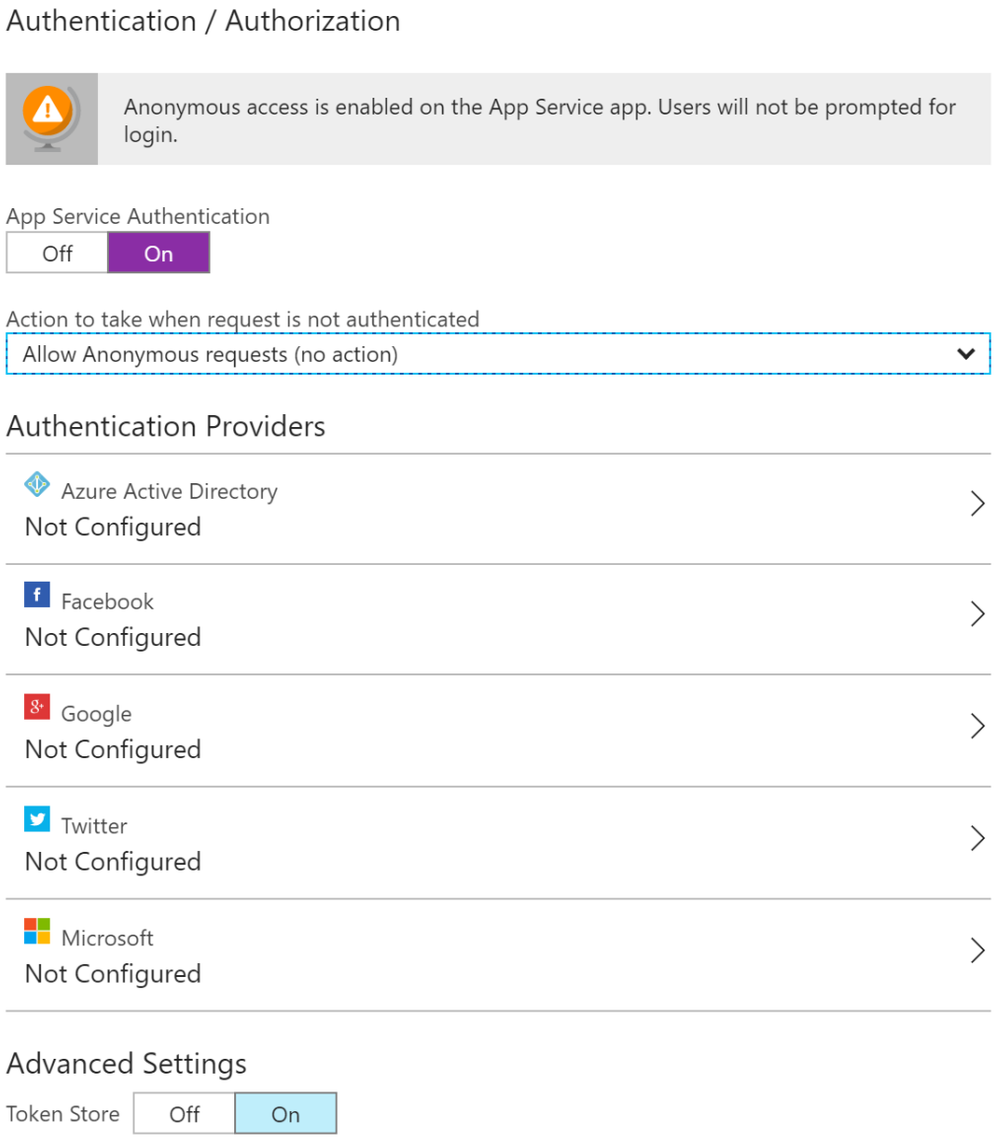
Social identities authentication

Integrate with common social identity providers

1. Azure Active Directory
2. Facebook
3. Google
4. Twitter
5. Microsoft accounts

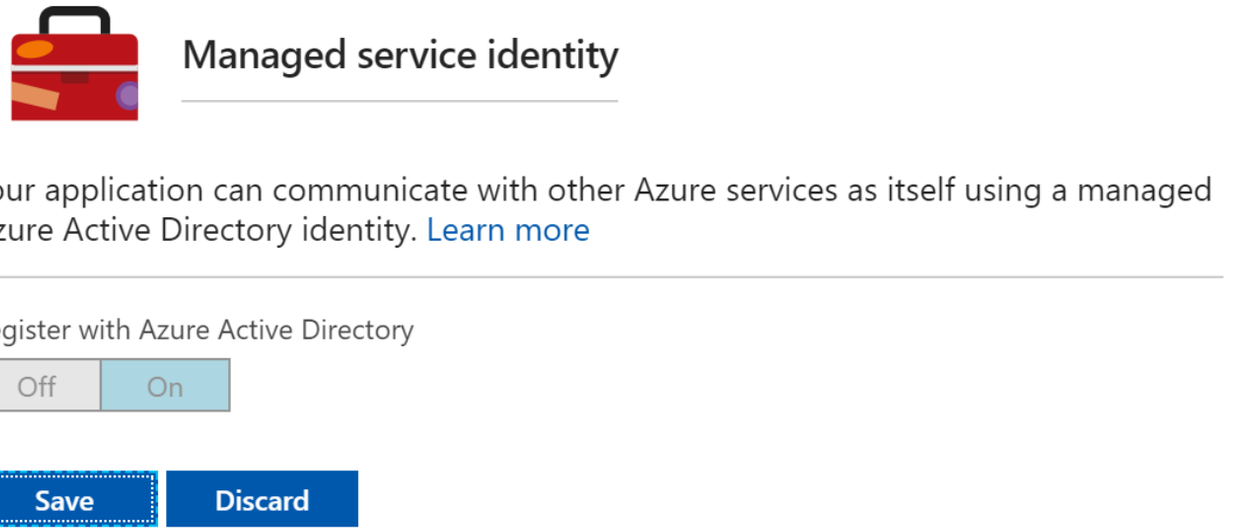
Requires separate setup

Can allow or disallow anonymous requests



Managed Service Identity

Run app as a managed Azure Active Directory identity

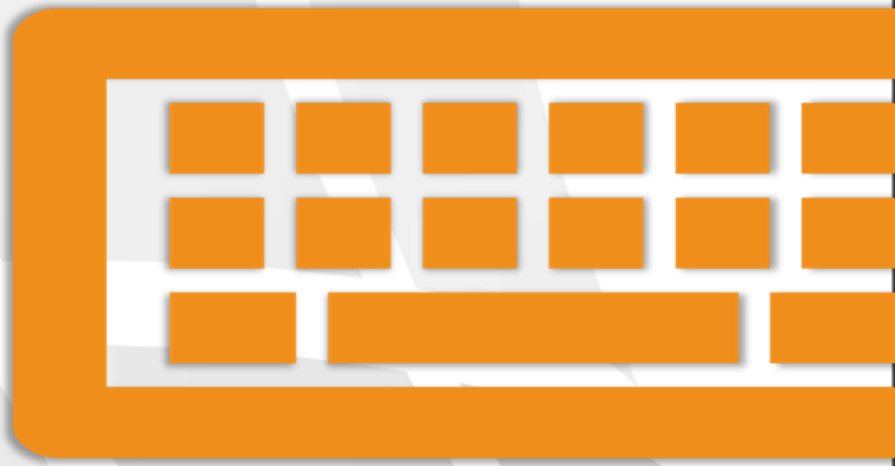


Access other Azure resources under MSI account

Requires IAM privileges for MSI on specific resource

Lab 5 – Securing Function Apps

Lab

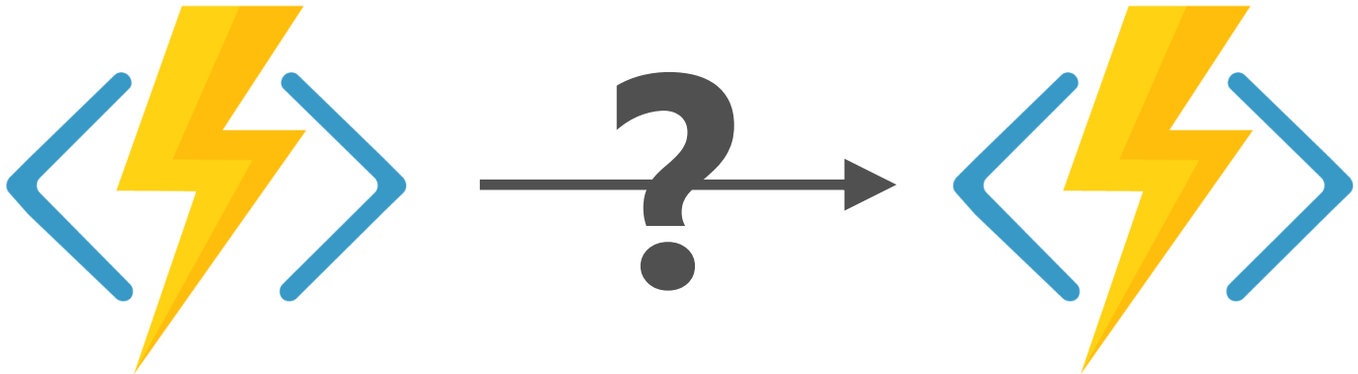


Functions calling functions

How do functions call each other?

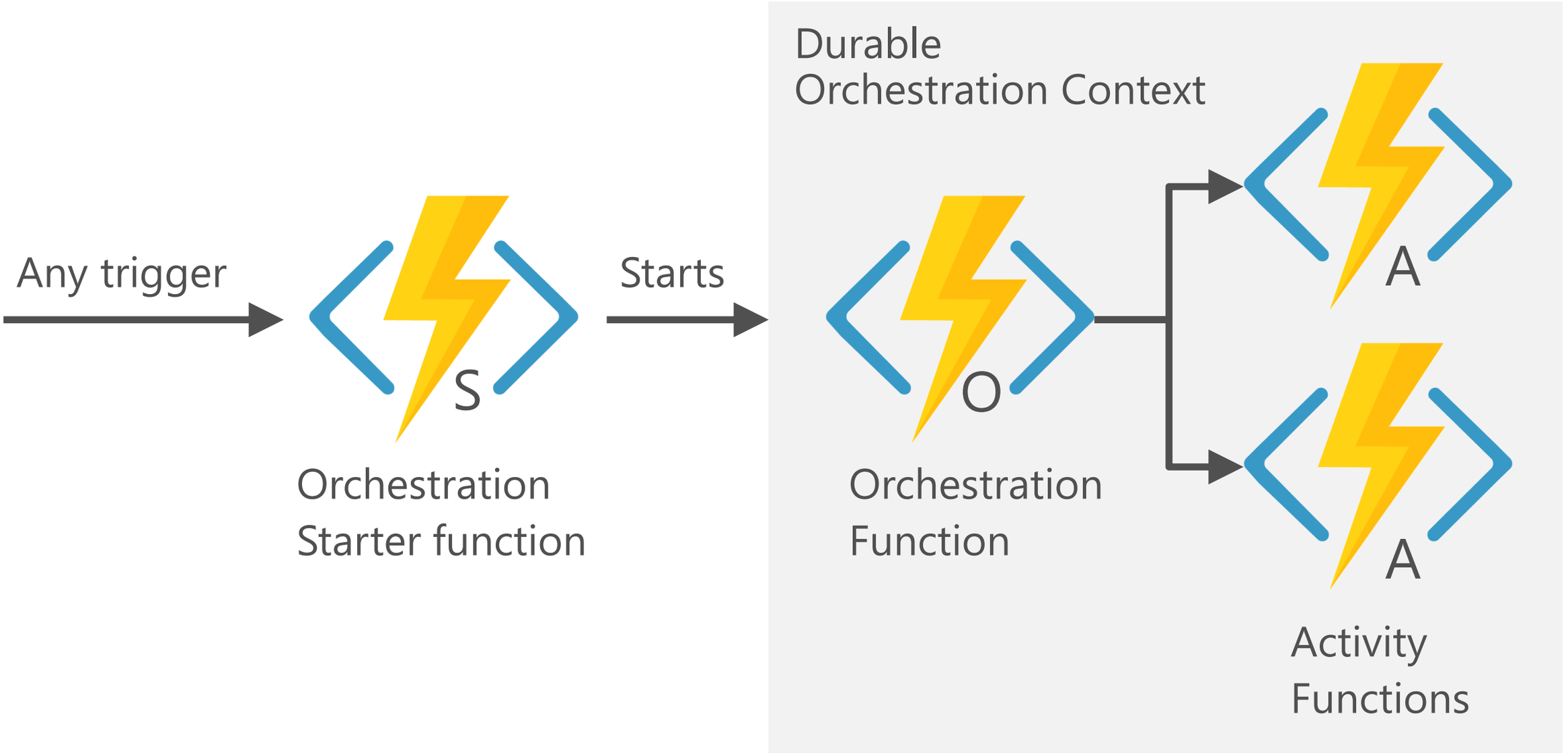
Reliable

Long running



Durable Task Framework

Function roles



Details of 3 moving parts

Orchestrator starter

- Uses trigger like regular functions
- Controls actual orchestrations
- Can inform status of orchestration

Orchestration function

- Performs actual orchestration
- Determines flow
- Call other activities using **`DurableOrchestrationContext(Base)`**

Orchestration activity

- Receives (optional) values from orchestration
- Uses **`ActivityTrigger`** with binding to data

Durable tips

Orchestration Functions can only call Activity Functions in same Function App

Keep your orchestrations small

What changes together should be deployed together

Use nameof() to avoid literals for activity and function names

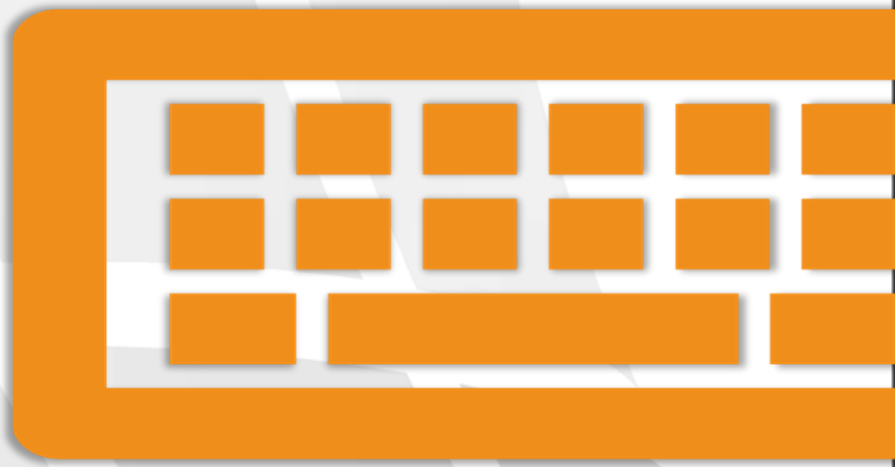
Tweaks for easier unit testing

Use DurableOrchestrationContextBase in orchestration

Use serializable types instead of DurableActivityContext

Lab 6 – Durable Extensions Framework

Lab



Azure Functions 2.0 in containers

Run functions from a Linux Docker container

CSX files by default

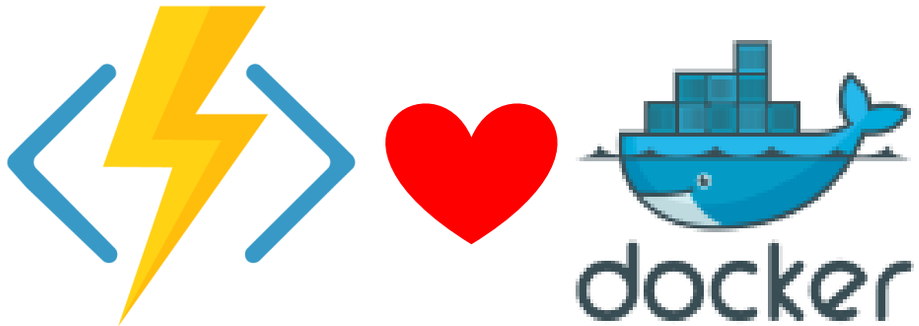
Can do same with .NET assemblies

Allows hosting in any cloud or Docker cluster

Combine with Azure Container Instances, Azure Kubernetes Services

```
docker build -t azurefunctionindocker .
```

```
FROM microsoft/azure-functions-dotnet-core2.0
ENV AzureWebJobsScriptRoot=/home/site/wwwroot
COPY bin/Release/netstandard2.0 /home/site/wwwroot
```




```
docker run -p 8080:80 azurefunctionindocker
```

Functions best practices

Single responsibility principle

Do one thing focussed

Finish as quickly as possible

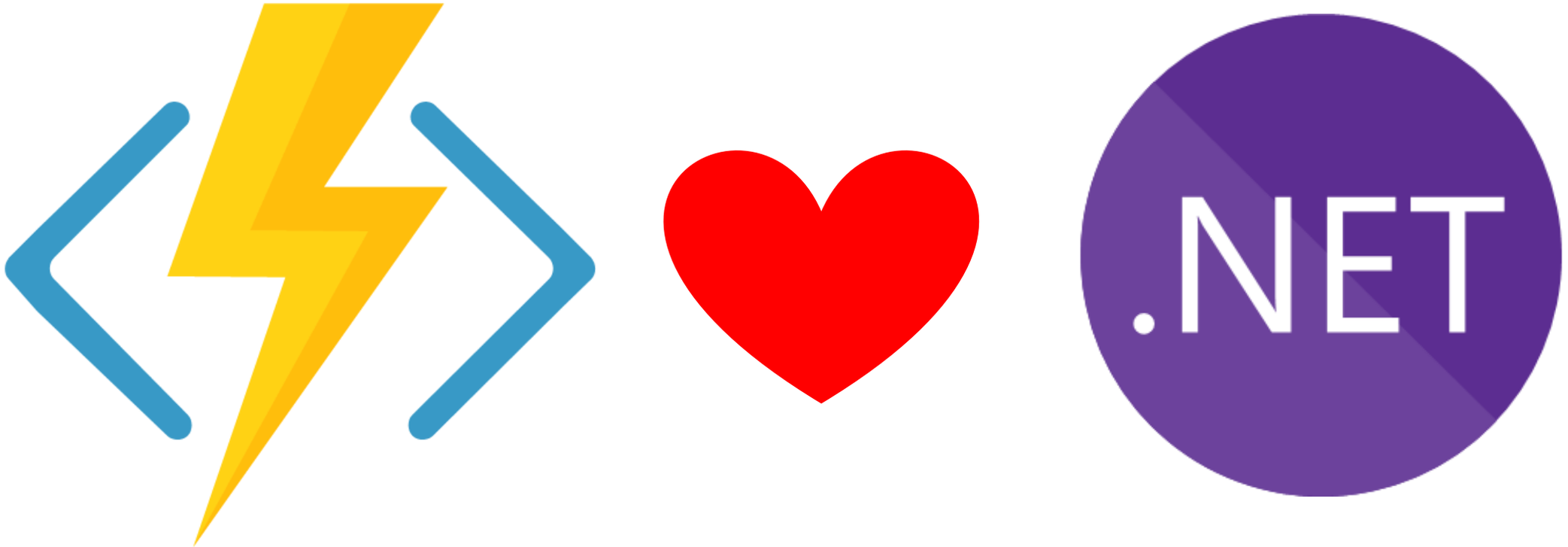
Avoid long running functions

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

Azure Functions Workshop



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