Visual Studio LIVE! EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS | VSLive! Virtcon 2020

Workshop: Serverless .NET Core Solutions using Azure Functions

Marc Duiker and Alex Thissen

Cloud architects, Xpirit

Level: Intermediate

MVP Microsoft® Most Valuable Professional

Xpirit

Microsoft® Most Valuable Professional

Marc Duiker

Cloud Architect
Xpirit Netherlands

Xpirit

@marcduiker – mduiker@xpirit.com

Microsoft® Most Valuable Professional

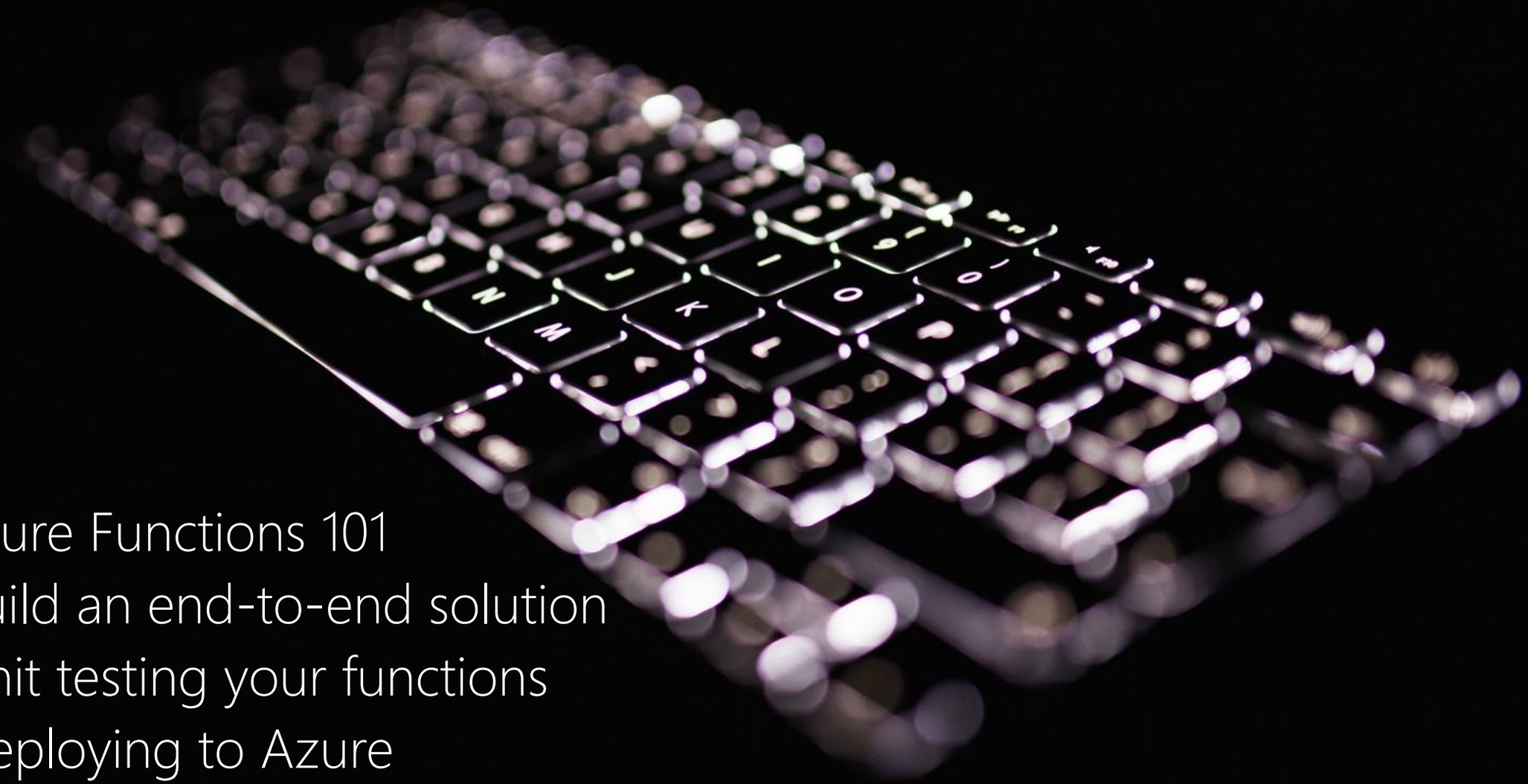
Alex Thissen

Cloud Architect
Xpirit Netherlands

Xpirit

@alexthissen – athissen@xpirit.com

Hands-on labs for Azure Functions



Lab 1 – Azure Functions 101
Lab 2 – Build an end-to-end solution
Lab 3 – Unit testing your functions
Lab 4 – Deploying to Azure

Workshop agenda

Presentations

Server-less architectures
Introduction to Azure Functions
Programming functions

10:00 AM CST

Unit testing
Going to production

11:30

Questions and Answers
Wrapup

12:45 PM

10:30

Getting started
Azure Functions 101

12:00

Building an end-to-end application
Unit testing*
Deploying to production*

After 1:00 PM

Continue at home

Labs

Browse to:
https://github.com/XpiritBV/AzureFunctionsWorkshop

```
git clone https://github.com/XpiritBV/AzureFunctionsWorkshop.git
```

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

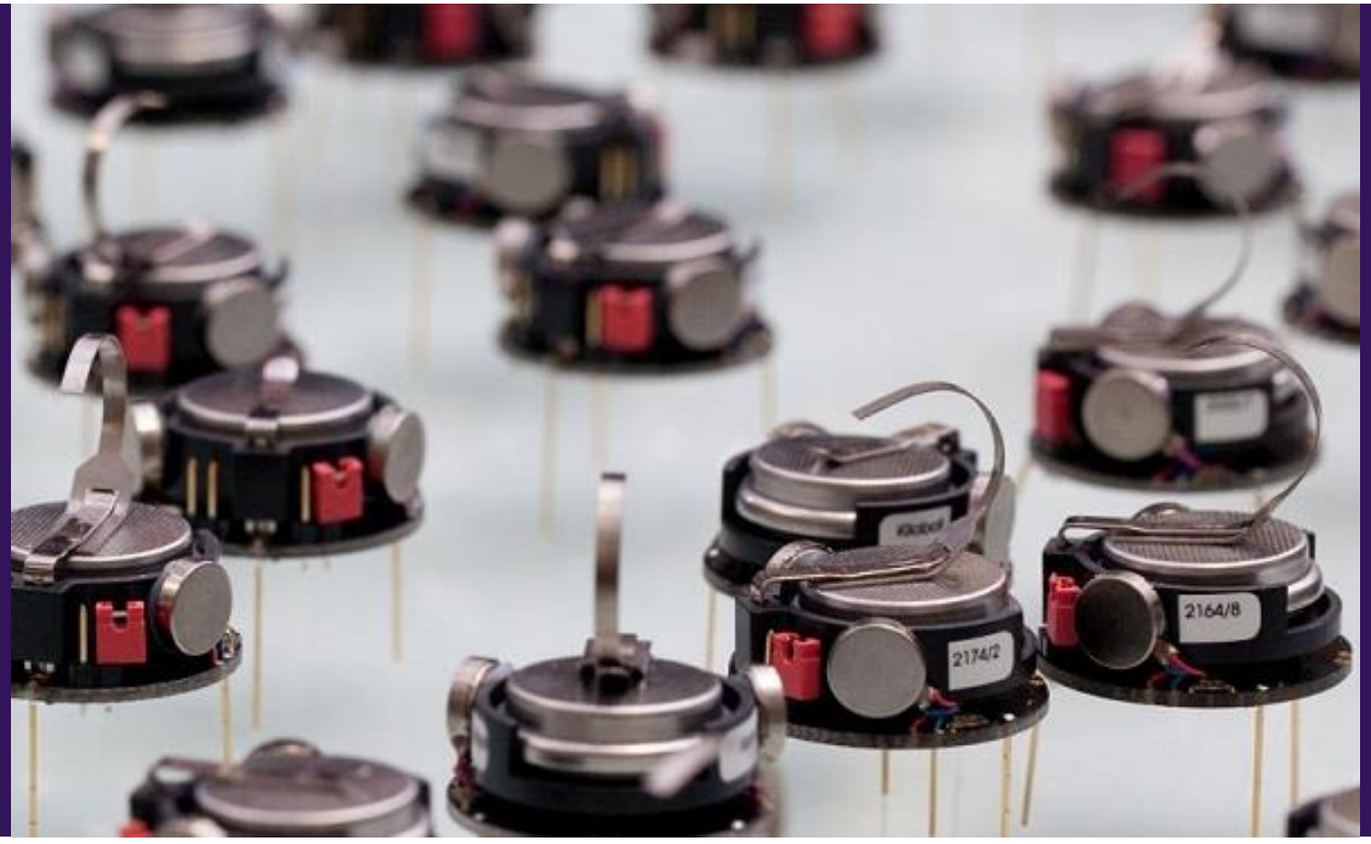
Transitioning to server-less

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)

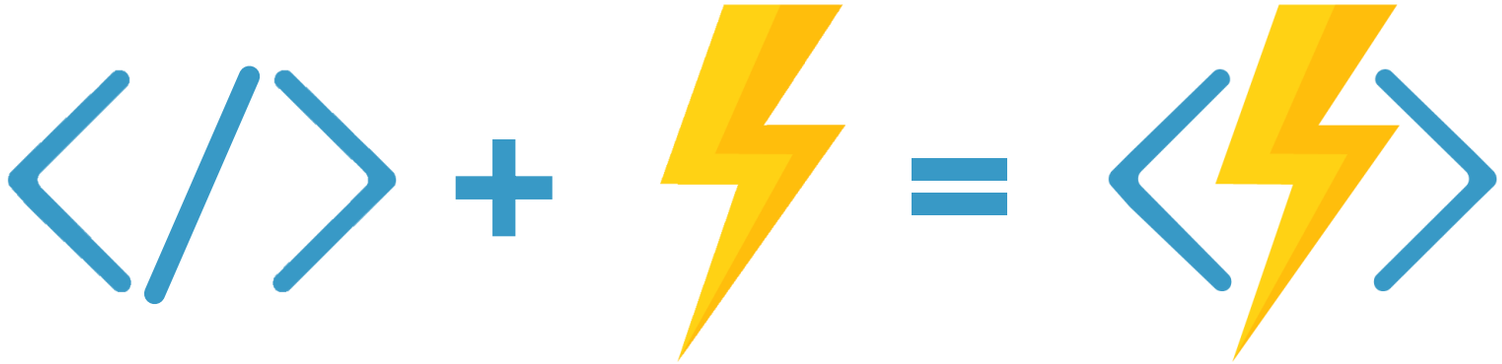
Google Cloud Functions
(since 2016)

Azure Functions
(since 2016)

Auth0 Webtask.IO
(since 2016)

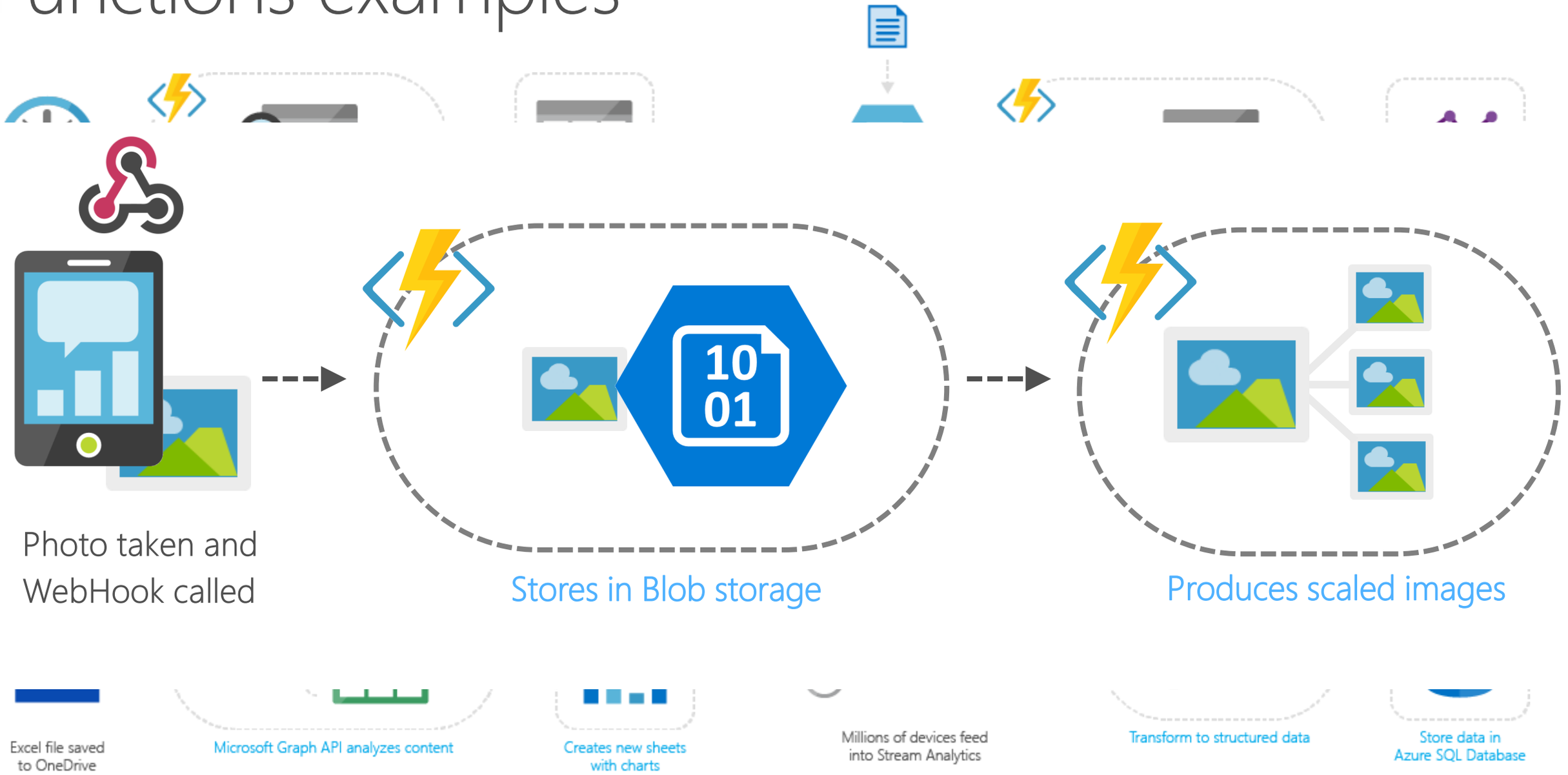
Focusing on Azure Functions

Code + Events + data = Azure Functions

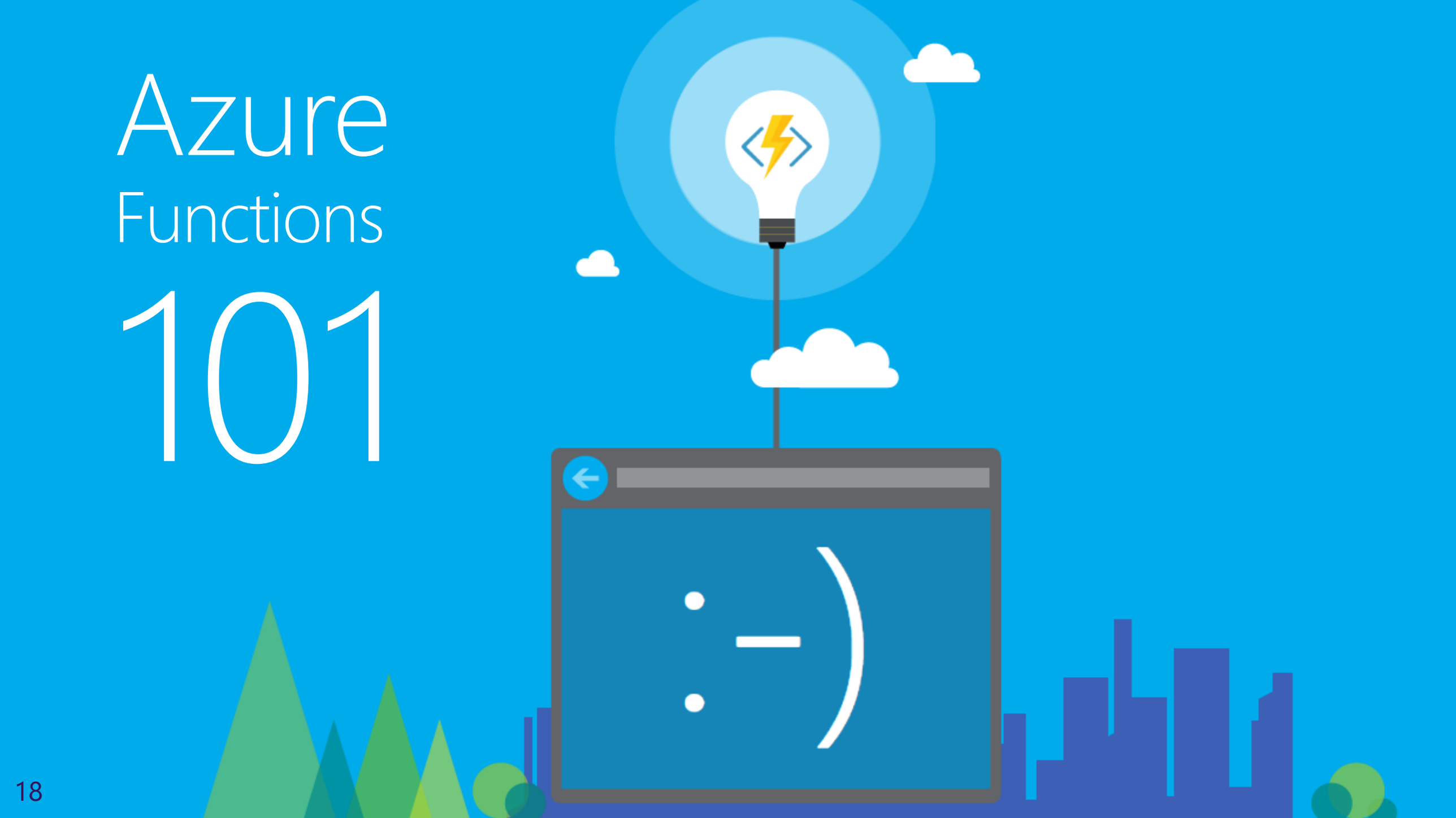


Process events with server-less code

Functions examples



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 Core with C#

Visual Studio 2019

Achieve higher code quality

Compilation and code analysis

Unit testing

Combine in CI/CD pipelines

About Azure Functions

Open source

https://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 multiple runtimes:

.NET Framework and .NET Core 1.0+

NodeJS

Python

PowerShell

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

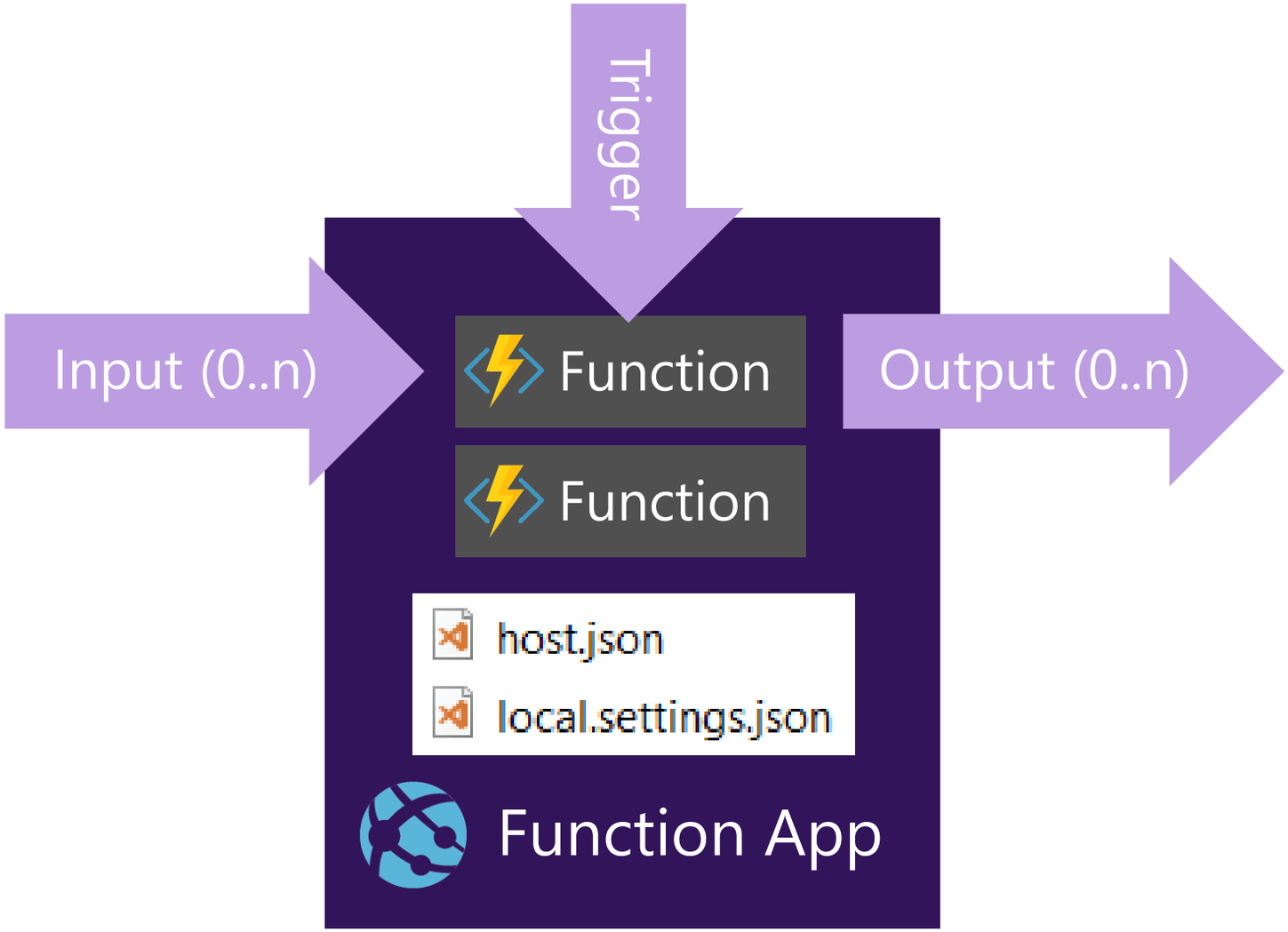
- Hosted as Azure App Service
- JSON based configuration
- Running (multiple) functions

Trigger starts execution

Bindings for input and output

Zero or more possible

Triggers and bindings can vary per function



Azure Functions with .NET Core 3.0

Based on .NET Standard 2.0

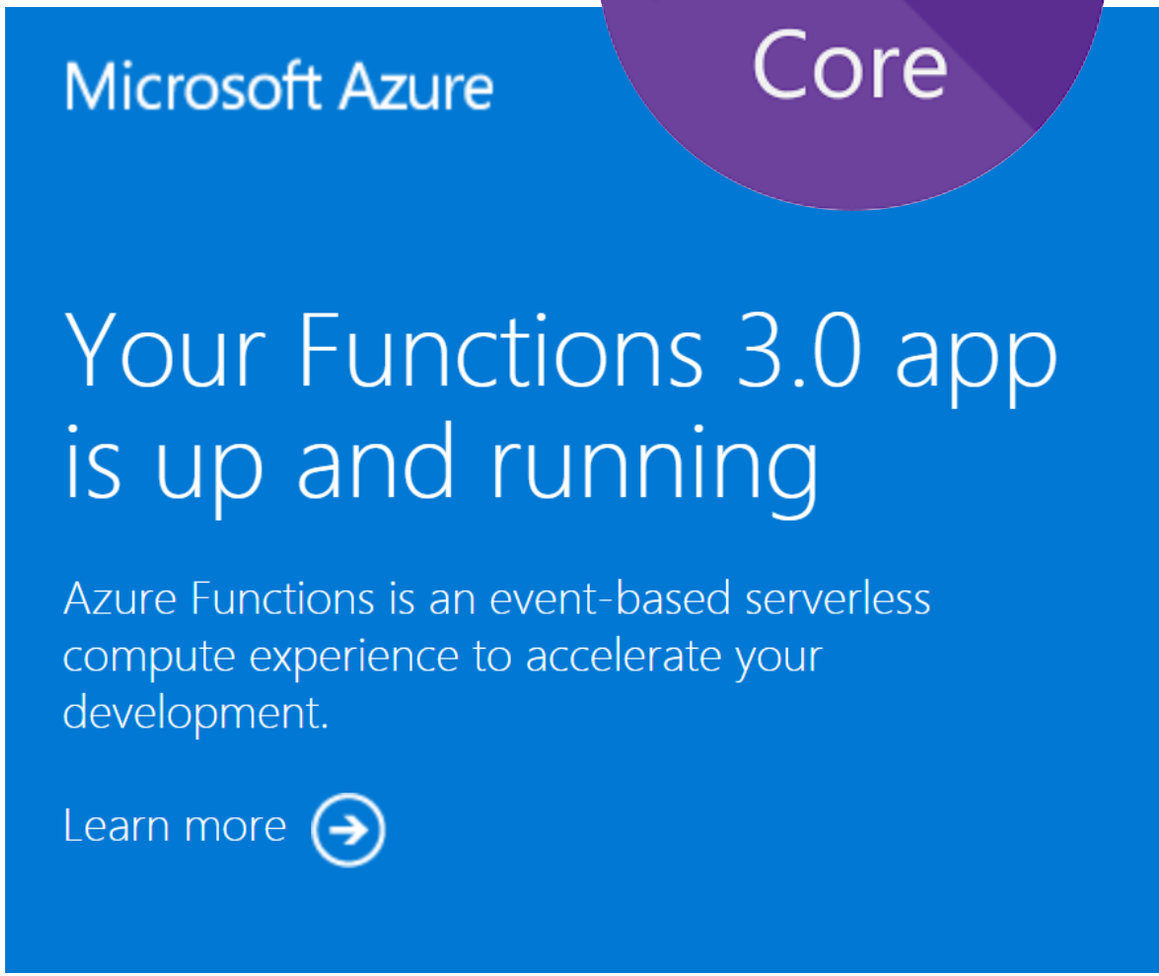
Requires Azure Functions Core tools 3.0

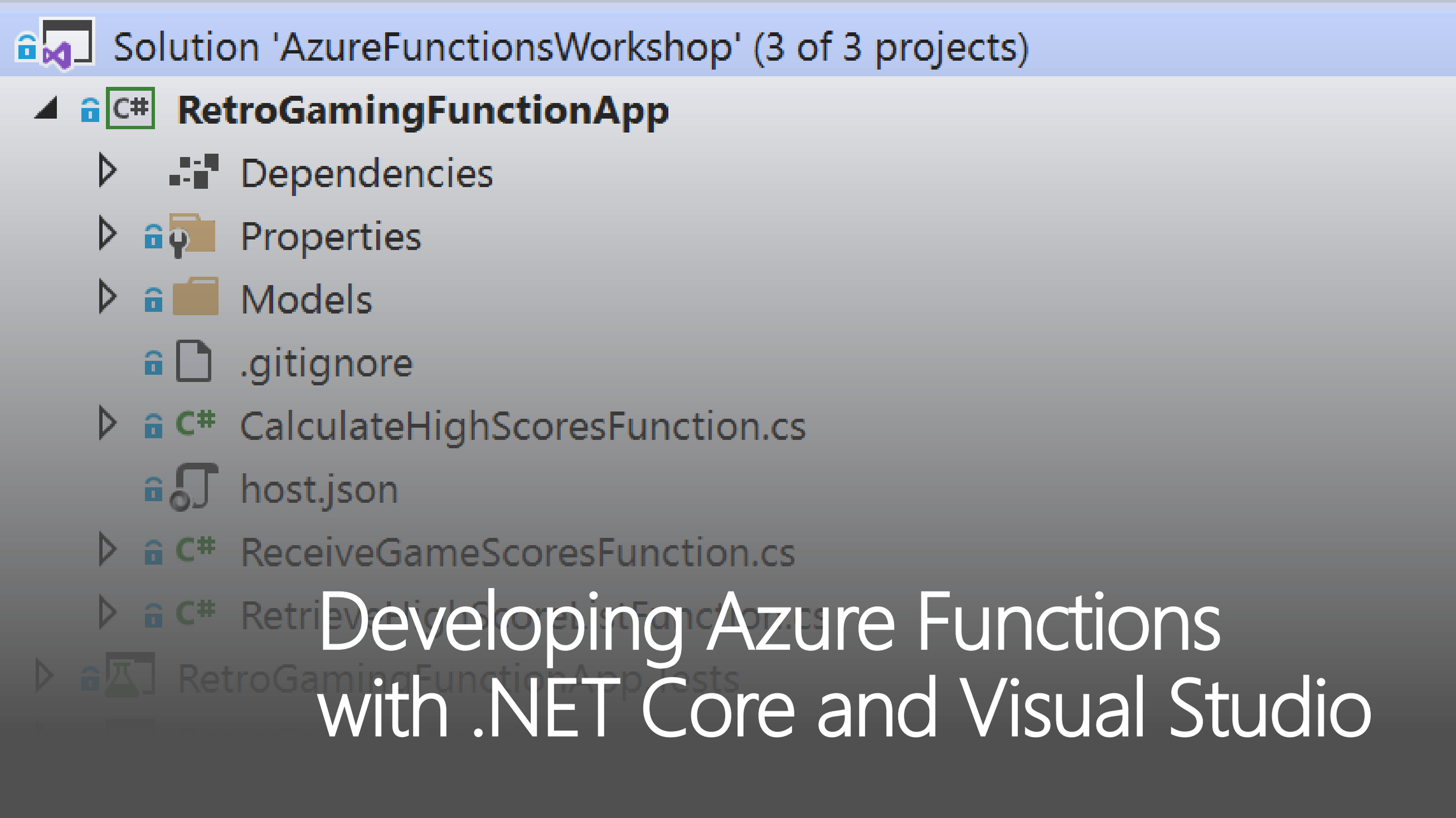
```
npm install –g azure-functions-core-tools@3
    --unsafe-perm true
```

https://go.microsoft.com/fwlink/?linkid=2135274

Switch to version 3.x runtime

Set application setting
`FUNCTIONS_EXTENSION_VERSION=3`




Solution 'AzureFunctionsWorkshop' (3 of 3 projects)
RetroGamingFunctionApp
Dependencies
Properties
Models
.gitignore
CalculateHighScoresFunction.cs
host.json
ReceiveGameScoresFunction.cs
Developing Azure Functions
with .NET Core and Visual Studio

Getting started with local development

Required tooling

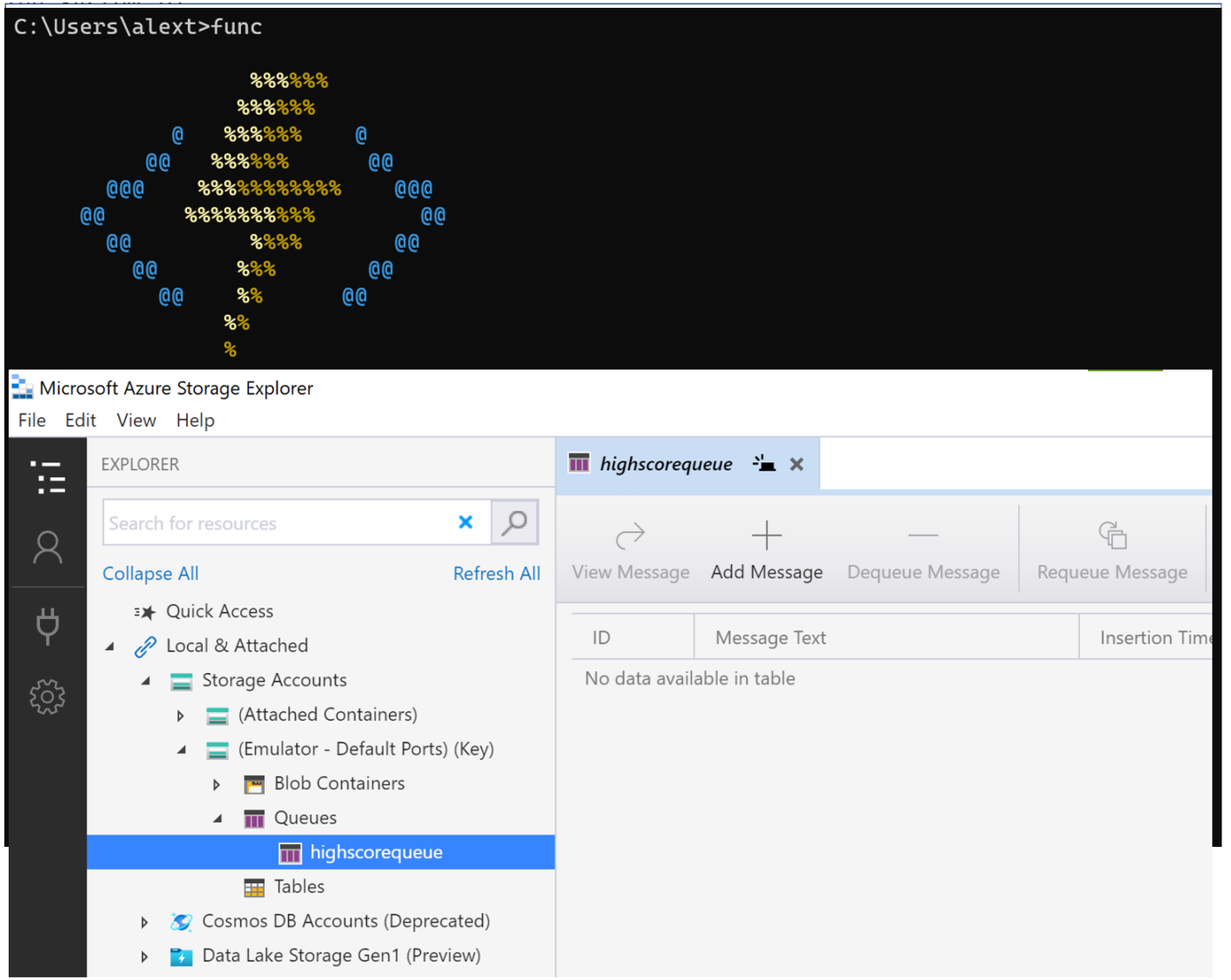
Visual Studio 2019 15.6 + Azure workload
Visual Studio Tools for Azure Functions
Azure Functions Core Tools (includes CLI)
.NET Core 3.1 installation

Azure Storage Explorer

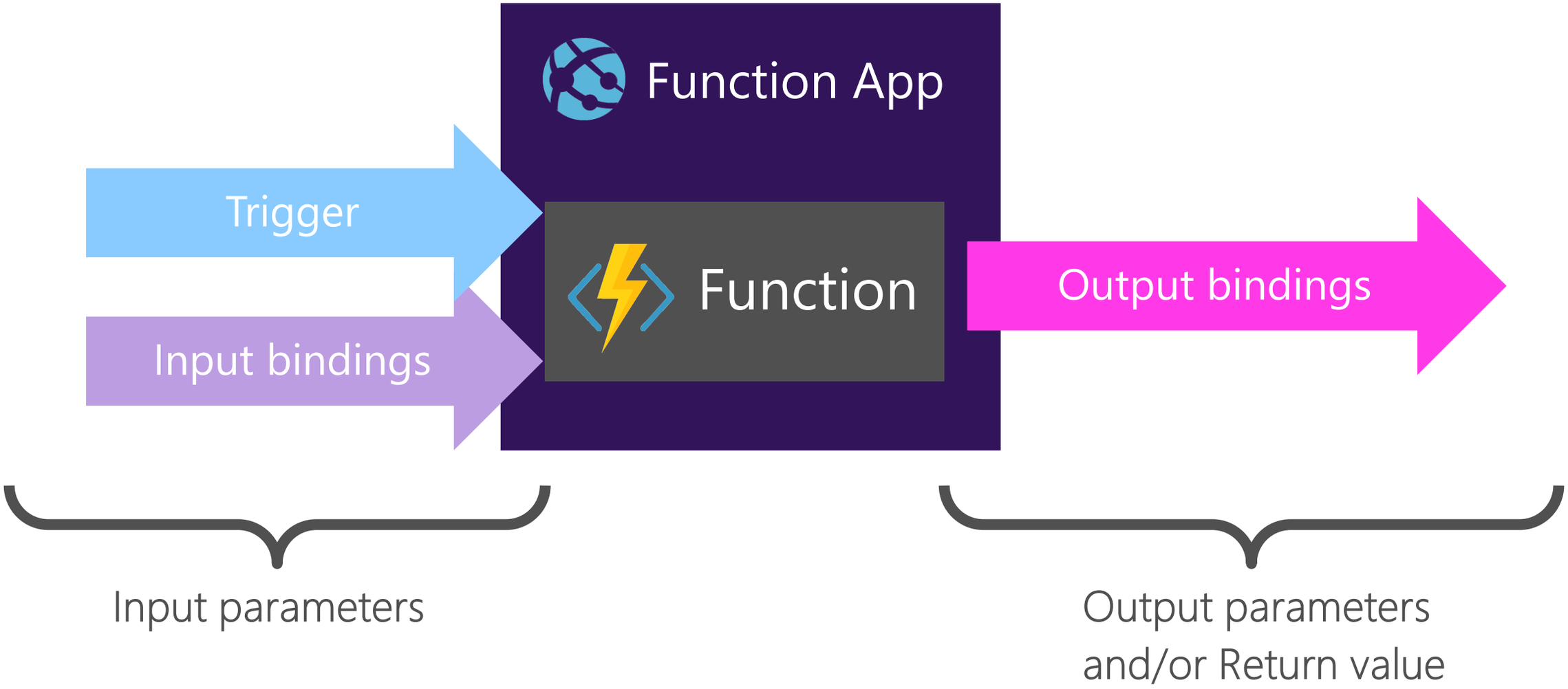
For interacting with Azure and local emulator

Useful optionals

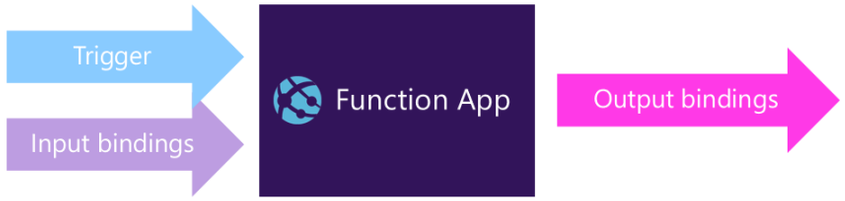
HTTP requests: Postman or SoapUI



Programming model: It's a function



Simple triggers and bindings



Define function.json metadata with attributes

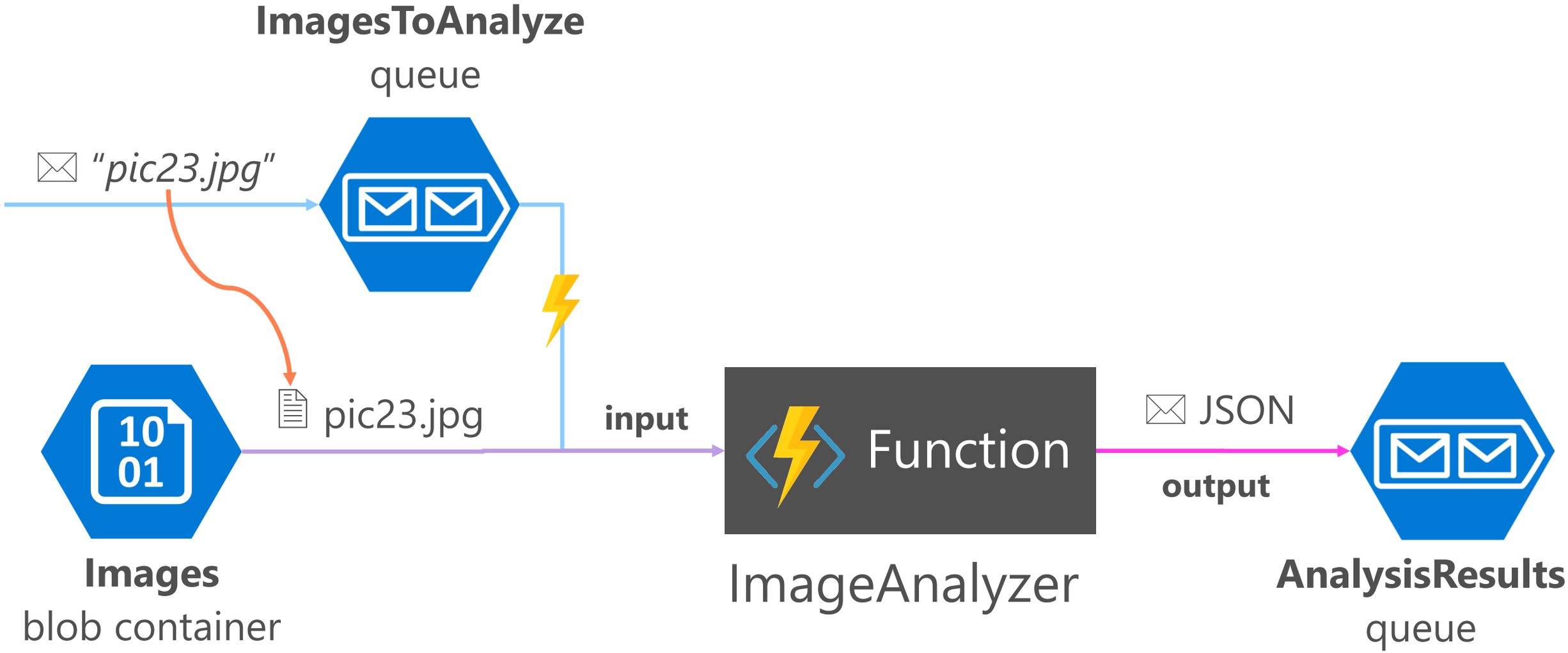
```
[FunctionName("HelloWorldFunction")]
public static async Task<IActionResult> Run(
    [HttpTrigger] HttpRequest request,
    ILogger log)
{
    log.Information("C# HTTP trigger function processed a request.");

    return new OkObjectResult("Hello, World!");
}
```

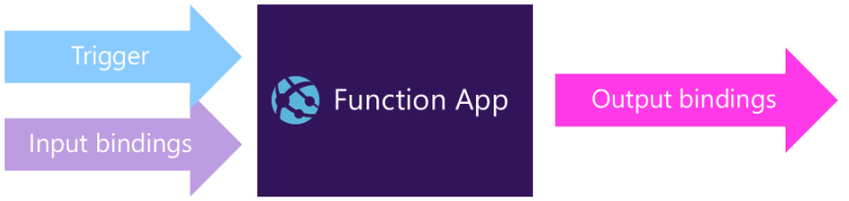
Return value
Single output binding can be done with return type

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, ILogger 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 and higher[1]	Trigger	Input	Output
Blob storage	✓	✓	✓	✓	✓
Azure Cosmos DB	✓	✓	✓	✓	✓
Dapr[3]		✓	✓	✓	✓
Event Grid	✓	✓	✓		✓
Event Hubs	✓	✓	✓		✓
HTTP & webhooks	✓	✓	✓		✓
IoT Hub	✓	✓	✓		✓
Kafka[2]		✓	✓		✓
Mobile Apps	✓			✓	✓
Notification Hubs	✓				✓
Queue storage	✓	✓	✓		✓
RabbitMQ[2]		✓	✓		✓
SendGrid	✓	✓			✓
Service Bus	✓	✓	✓		✓
SignalR		✓		✓	✓
Table storage	✓	✓		✓	✓
Timer	✓	✓	✓		
Twilio	✓	✓			✓

Lab 1 – Azure Functions 101

HTTP, Queues and Tables

Unit testing your functions

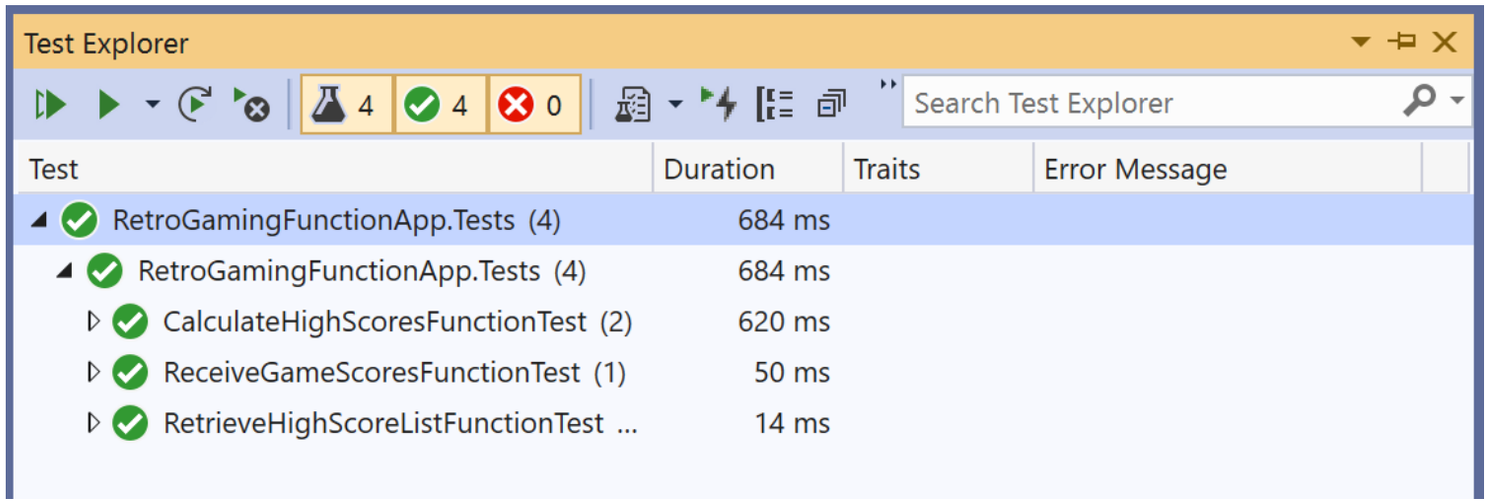
Automate your test efforts

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)

HttpRequest(Message)

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

Logger

- Use **`ILogger`** provided through dependency injection
- .NET FX: Inherit **`TraceWriter`** and override Write method for mock

Demo

Debugging and unit testing
Azure Functions

Visual Studio LIVE! EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS | VSLive! Virtcon 2020

Lab 2 – Building an end-to-end solution

Putting it all together

Lab 3 – Unit testing Azure Functions

Moving into production

Hosting options

Azure

Consumption-based

Scaling as needed

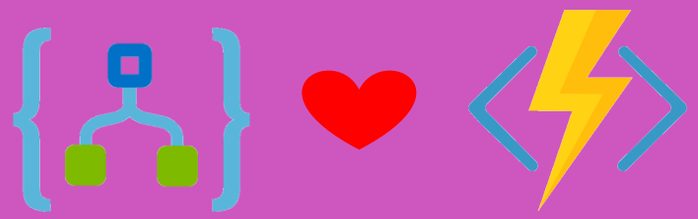
Might require time to provide compute instances

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

Microsoft.Net.Sdk.Functions

Composes publishable output

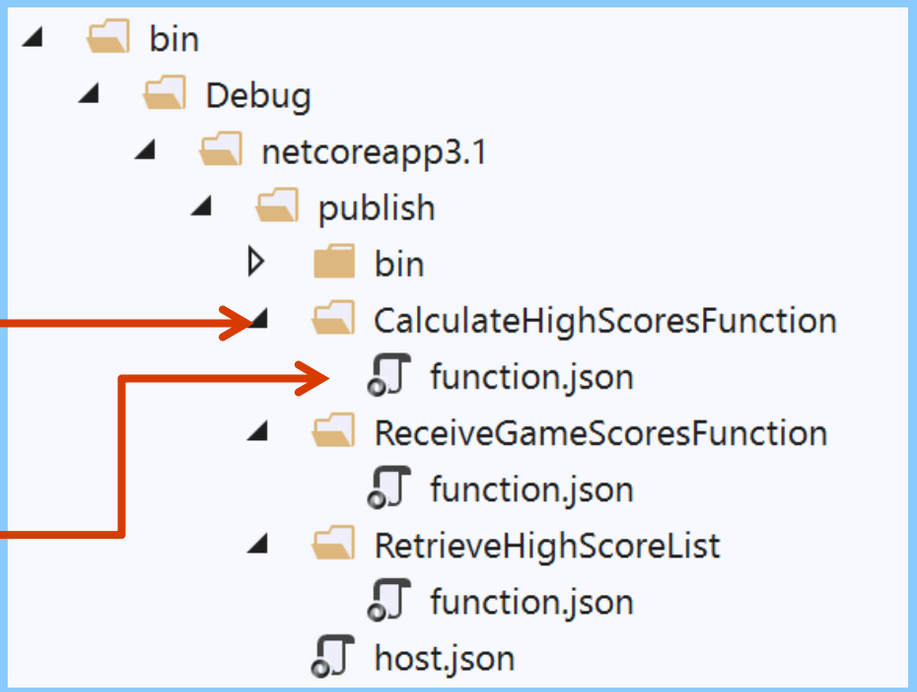
Inside build target (e.g. net471 or netstandard2.0)

Deviates from standard build output location

Translates C# attributes metadata

Each function produces one function.json file

```
[FunctionName("CalculateHighScoresFunction")]
public static async Task Run(
[QueueTrigger("gamescorequeue")]GameScoreReceivedEvent message,
[Table("HighScores")] CloudTable table,
[SignalR(HubName = "leaderboardhub")]
IAsyncCollector<SignalRMessage> signalRMessages,
ILogger log)
```



NuGet metadata package

Referenced dependencies, tooling, custom .targets files

Build scenarios

1 Connect to code repo

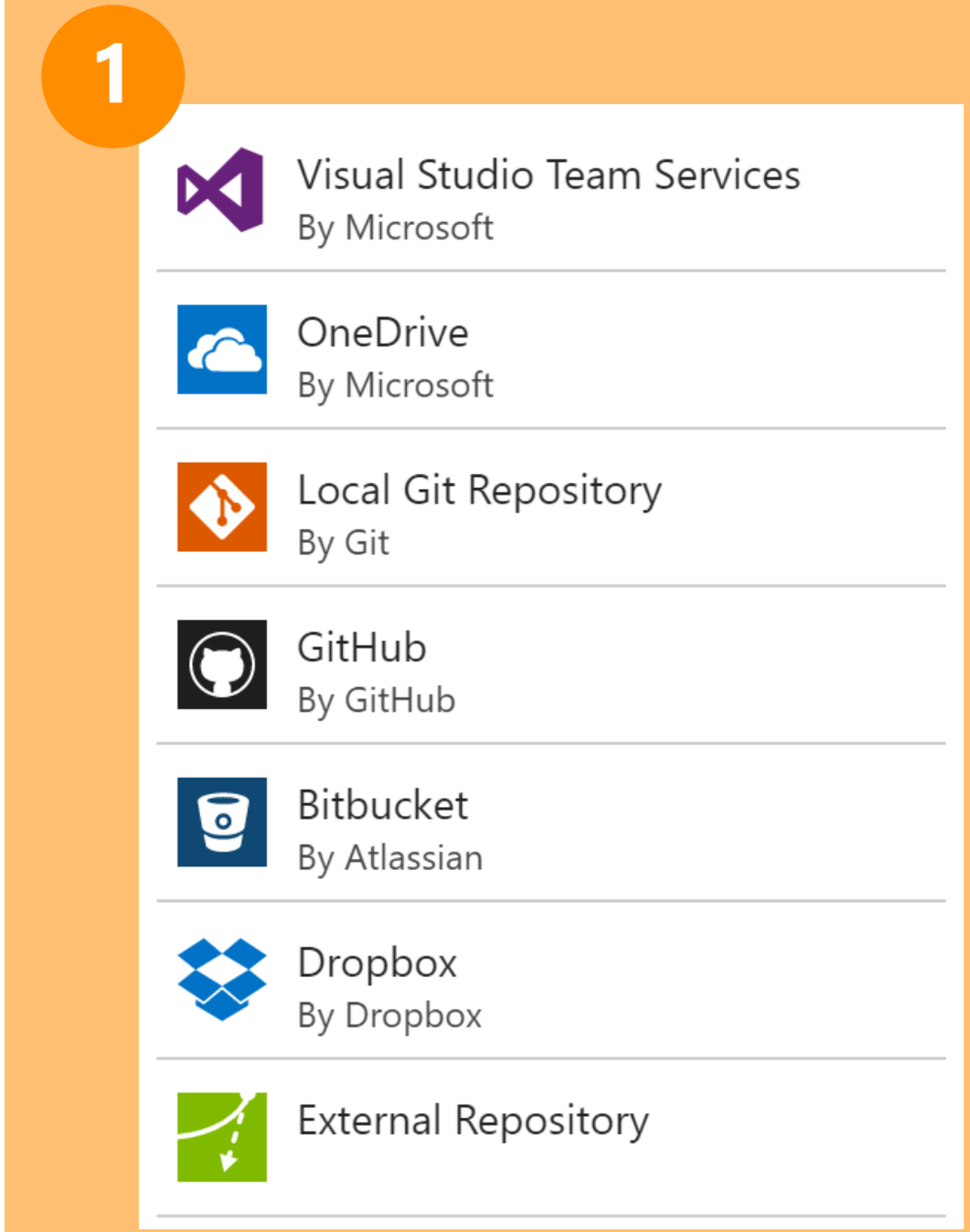
Suitable for dynamic workflow without VS2019

2 CI/CD pipeline in Azure DevOps

High-quality build process with testing

Don't publish from VS or CLI

Only for quick starts
and testing purposes



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

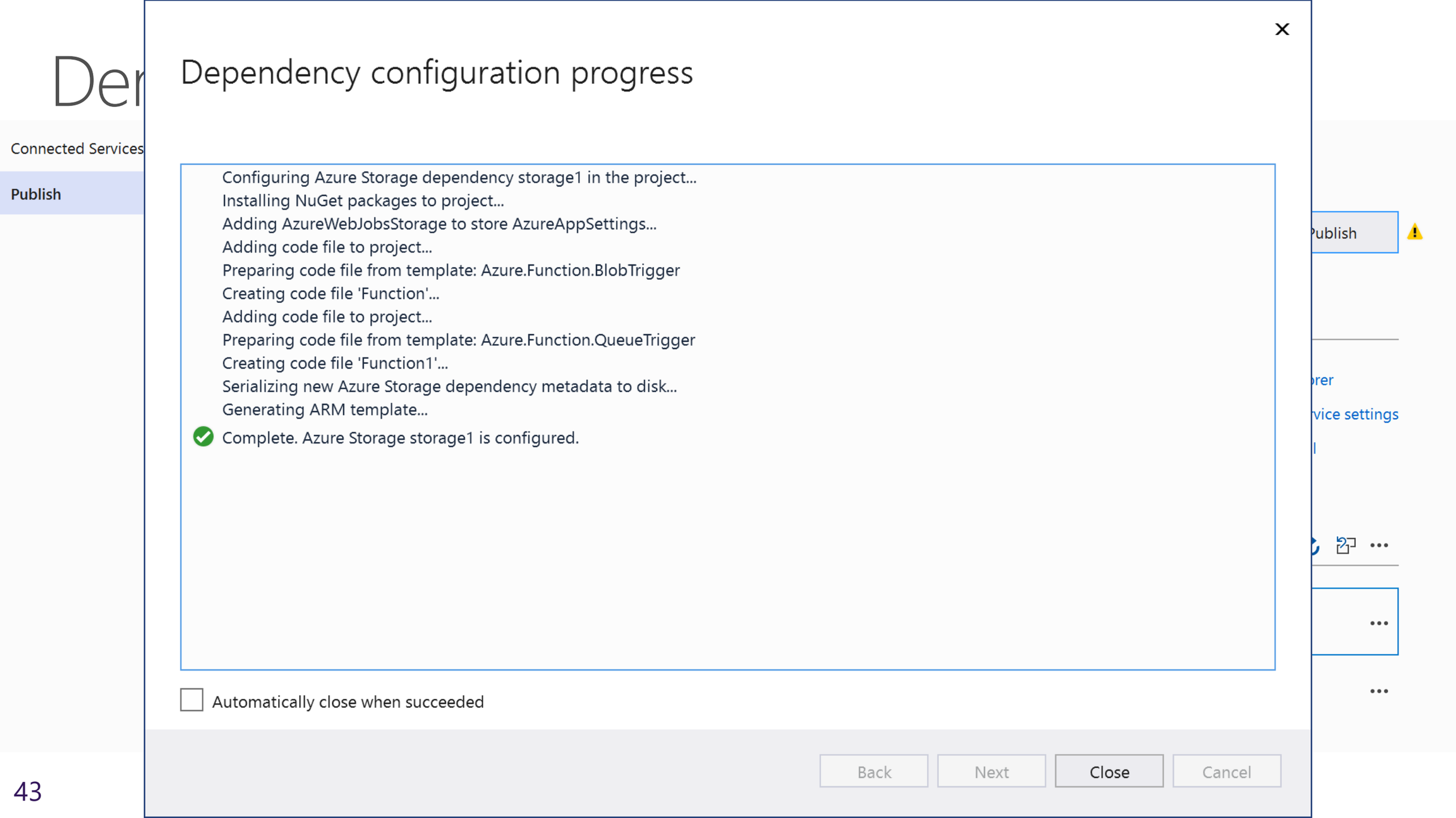
Connection strings separate section

```
{
  "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


Der
Connected Services
Publish
Publish
rer
vice settings
Dependency configuration progress
Configuring Azure Storage dependency storage1 in the project...
Installing NuGet packages to project...
Adding AzureWebJobsStorage to store AzureAppSettings...
Adding code file to project...
Preparing code file from template: Azure.Function.BlobTrigger
Creating code file 'Function'...
Adding code file to project...
Preparing code file from template: Azure.Function.QueueTrigger
Creating code file 'Function1'...
Serializing new Azure Storage dependency metadata to disk...
Generating ARM template...
Complete. Azure Storage storage1 is configured.
Automatically close when succeeded
Back
Next
Close
Cancel

Lab 4 – Deploying Function Apps to Azure

Visual Studio LIVE!
EXPERT SOLUTIONS FOR ENTERPRISE DEVELOPERS

VSLive! Virtcon 2020

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
https://github.com/XpiritBV/AzureFunctionsWorkshop

Contact @AzureFunctions

Watch Function Junction videos on Channel9