Chad Green

Serverless in Action

CodeStock
April 13, 2019

Who is Chad Green

- Director of Software Development at ScholarRx
- Community Involvement
    - Code PaLOUsa Conference Chair
    - Louisville .NET Meetup Organizer
    - Louisville Tech Leaders Meetup Co-Organizer
    - Louisville Tech Ladies Committee Member
- Contact Information

chadgreen@chadgreen.com

chadgreen.com

ChadGreen

ChadwickEGreen

Agenda

- What is Serverless Computing
- Functions as a Service
- Serverless Options
- Azure Functions Overview
- Azure Functions in Action
- Pricing
- Best Practices

What is Serverless Computing

Serverless in Action

The evolution of application platforms

What media should I use to keep **backups**?

What is the right **size** of **servers** for the business needs?

How do I **deploy** new **code** to my **servers**?

What happens in case of **server hardware** failure?

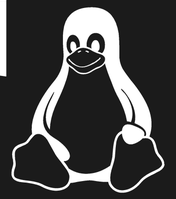
How can I increase **server** utilization?

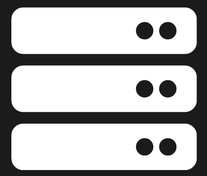
Which packages should be on my **server?**

Who **monitors** my **App**?

How often should I backup my **server**?

What size of **servers** should I **buy?**





How can I **scale** my app?

Are my **servers** in a secure location?

Do I need a secondary **network connection**?

Who has **physical** access to my **servers**?

Which **Operating System** should I use?

What **happens** if the power goes out?

How many **servers** do I need?

Who **monitors** my **servers?**

Do I need a **UPS**?

What **storage** do I need to use?

How often should I **patch** my **servers**?

How can I dynamically configure my app?

It takes how long to **provision** a new **server**?

On-Premises

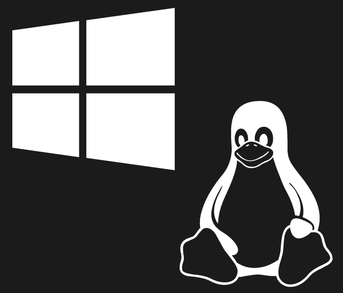
The evolution of application platforms

What is the right **size** of servers for my business needs?

How can I increase **server** utilization?

How many **servers** do I need?

How can I **scale** my application?



How often should I **patch** my **servers**?

How often should I backup my **server**?

Which packages should be on my **server**?

How do I **deploy** new **code** to my **server**?

Which **Operating System** should I use?

Who **monitors** my application?

On-Premises

IaaS

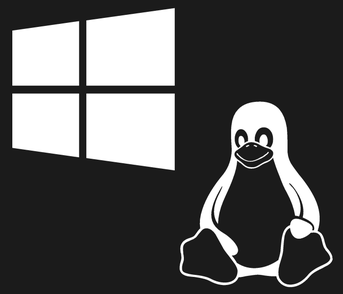
The evolution of application platforms

What is the right **size** of servers for my business needs?

How can I increase **server** utilization?

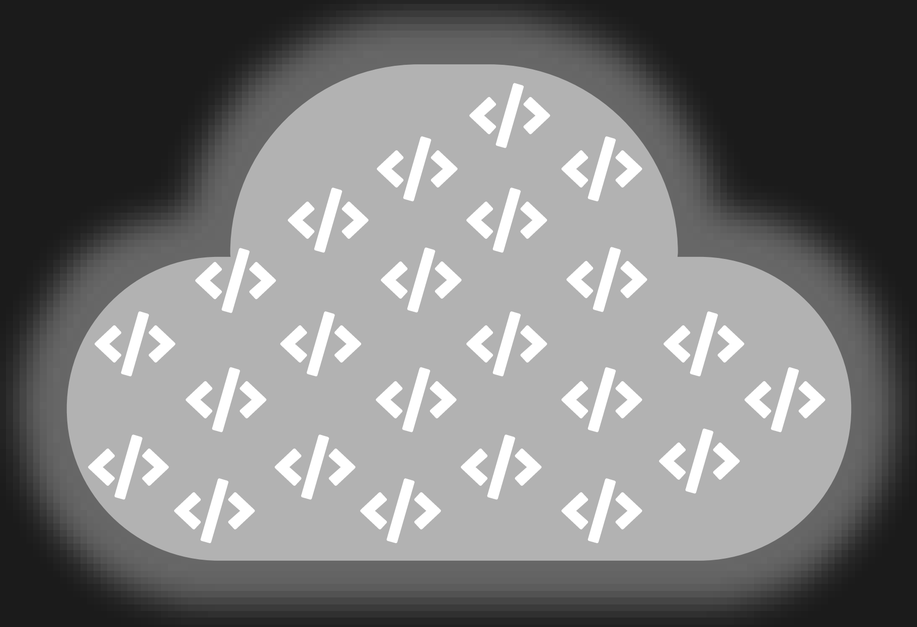
How many **servers** do I need?

How can I **scale** my application?



On-Premises — IaaS — PaaS

The evolution of application platforms



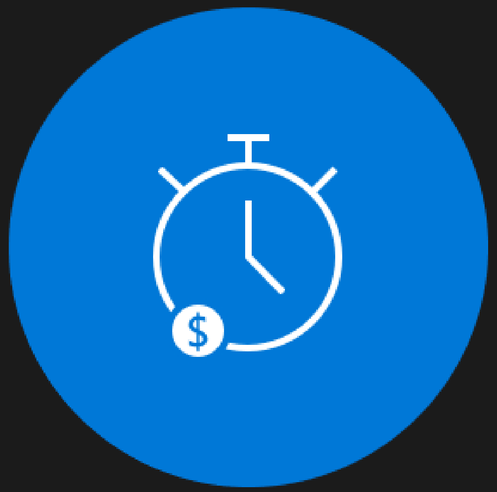
The platform for next generation applications

On-Premises — IaaS — PaaS — Serverless

What is Serverless?

Abstraction of Servers

Event-Driven/Instant Scale



Micro-Billing

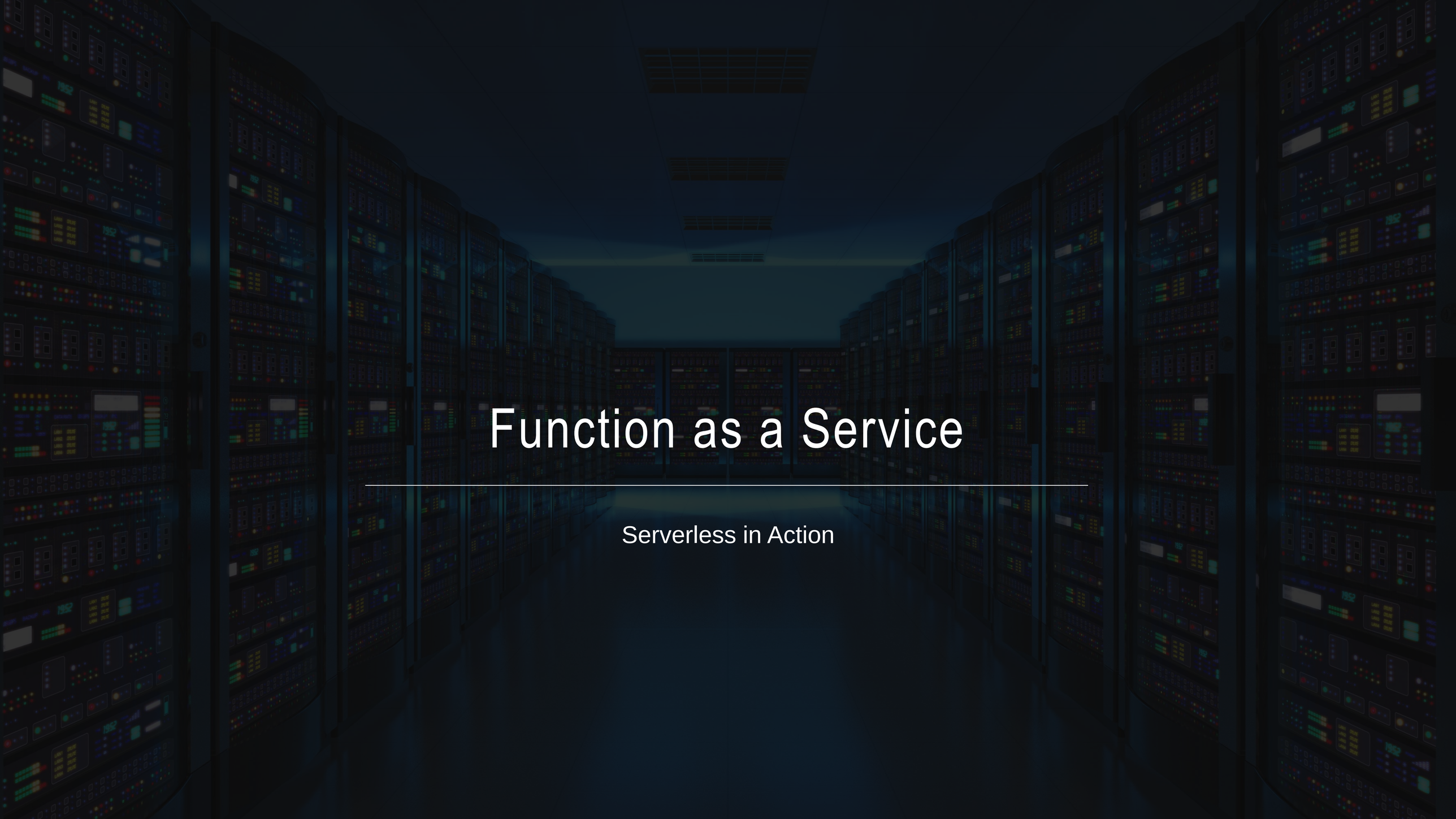
Benefits of Serverless

Manage apps not servers

Reduced DevOps

Faster Time to Market

Challenges of Serverless Architecture



Function as a Service

Serverless in Action

Serverless is more than just one thing

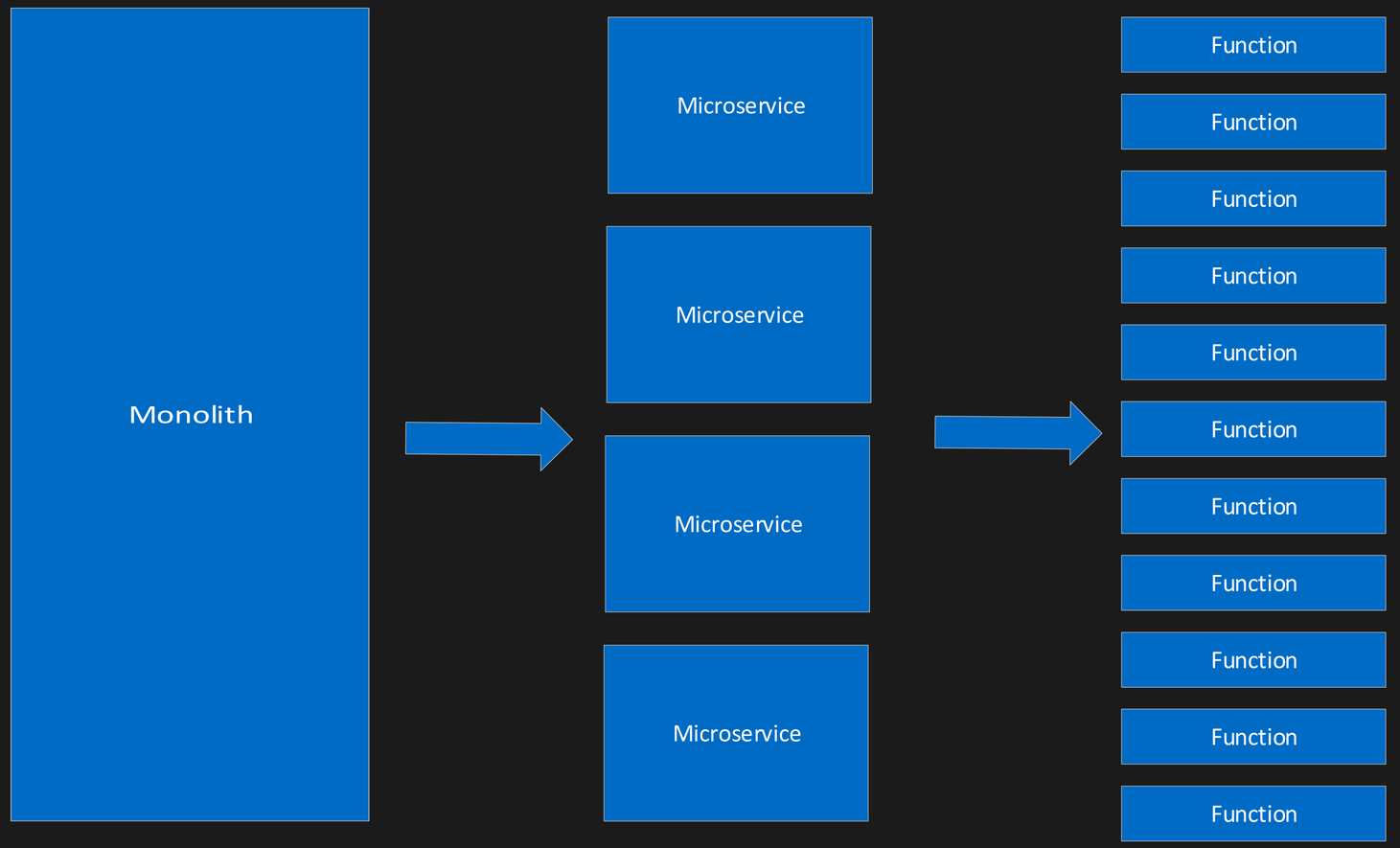
Backend as a Service (BaaS)

- Applications that significantly or fully depend on services (in the cloud) to manage server-side logic and state

Functions as a Services (FaaS)

- Application run in stateless compute containers that are event-triggered, ephemeral, and fully managed by a 3rd party

Function Scale



Nano Services

FaaS is at the center of serverless

Single responsibility

Functions are single-purposed, reusable pieces of code that process an input and return a result

Short lived

Functions don’t stick around when finished executing, freeing up resources for further executions

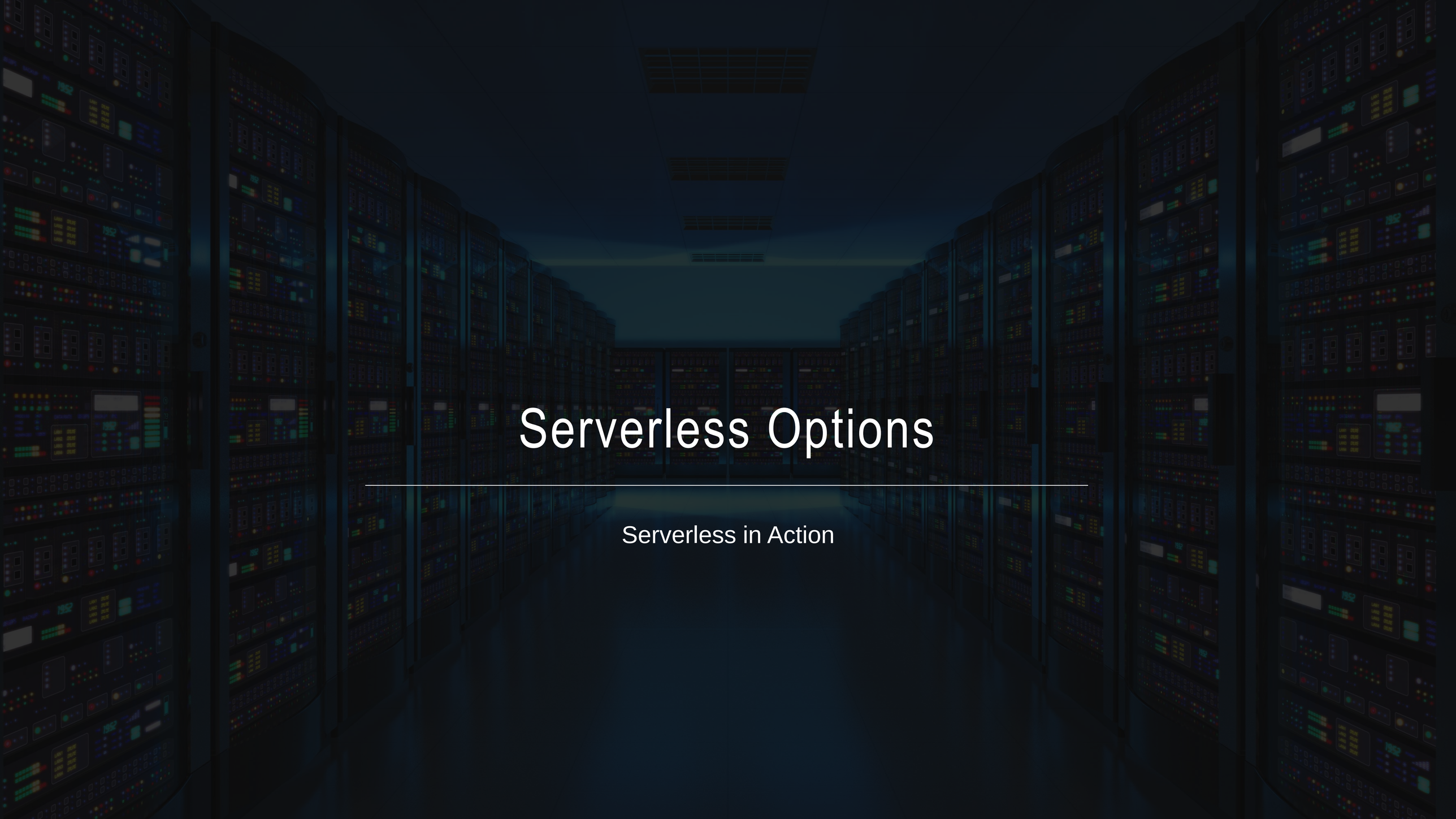
Stateless

Functions don’t hold any persistent state and don’t rely on the state of any other processes

Event driven & scalable

Functions respond to predefined events, and are instantly replicated as many times as needed

Functions-as-a-Service programming model use functions to achieve true serverless compute



Serverless Options

Serverless in Action

Market Landscape

AWS Lambda

Google Cloud Functions

IBM Cloud Functions

Auth0 WebTask

Azure

Execute code without thinking about servers. Pay only for the compute time you consume.

Run code on demand in a highly scalable serverless environment.

Event-driven serverless compute platform

Your code, All yours.

No servers. Just code.

Azure Serverless Offerings

Event Grid

Manage all events that
can trigger code or logic

Logic Apps

Design workflows and
orchestrate processes

Functions

Execute your code based
on events you specify

Database

Storage

Security

IoT

Analytics

Intelligence

Code

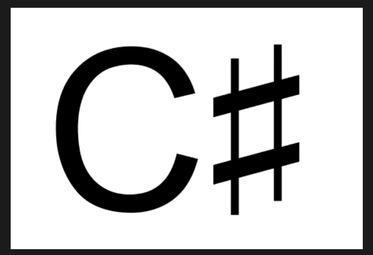
Events + data

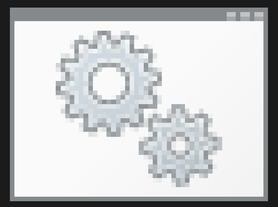
Azure Functions

Serverless in Action

Features of Azure Functions

- Choice of language

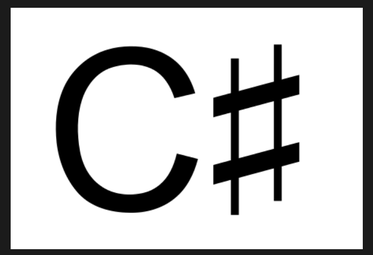


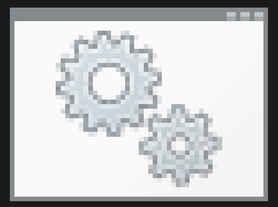


Batch

Features of Azure Functions

- Choice of language
- Pay-per-use pricing model





Batch

Features of Azure Functions

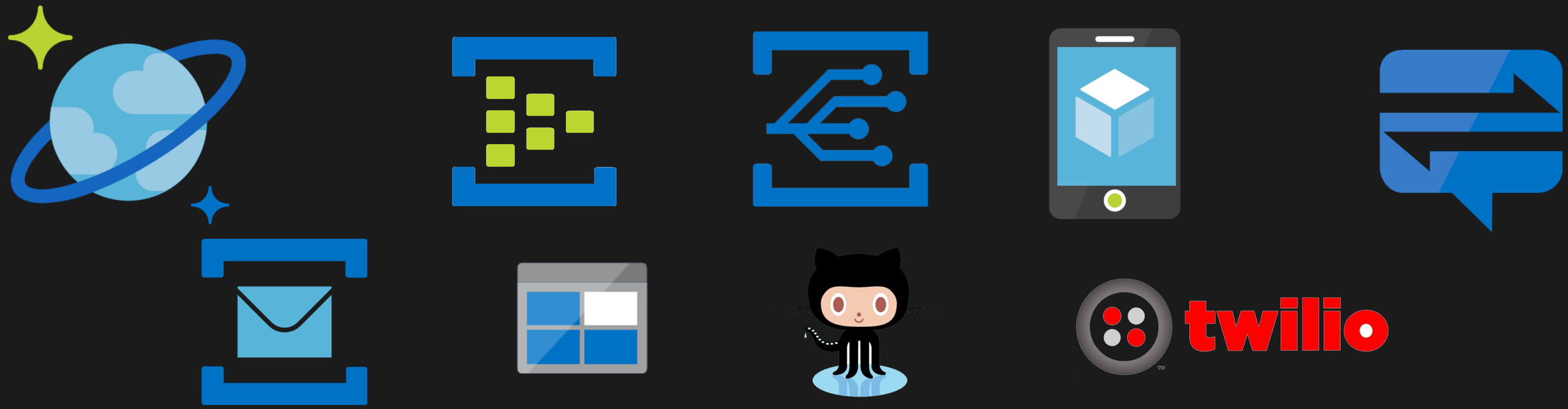
- Choice of language
- Pay-per-use pricing model
- Bring your own dependencies

Features of Azure Functions

- Choice of language
- Pay-per-use pricing model
- Bring your own dependencies
- Integrated security

Features of Azure Functions

- Choice of language
- Pay-per-use pricing model
- Bring your own dependencies
- Integrated security
- Simplified integration

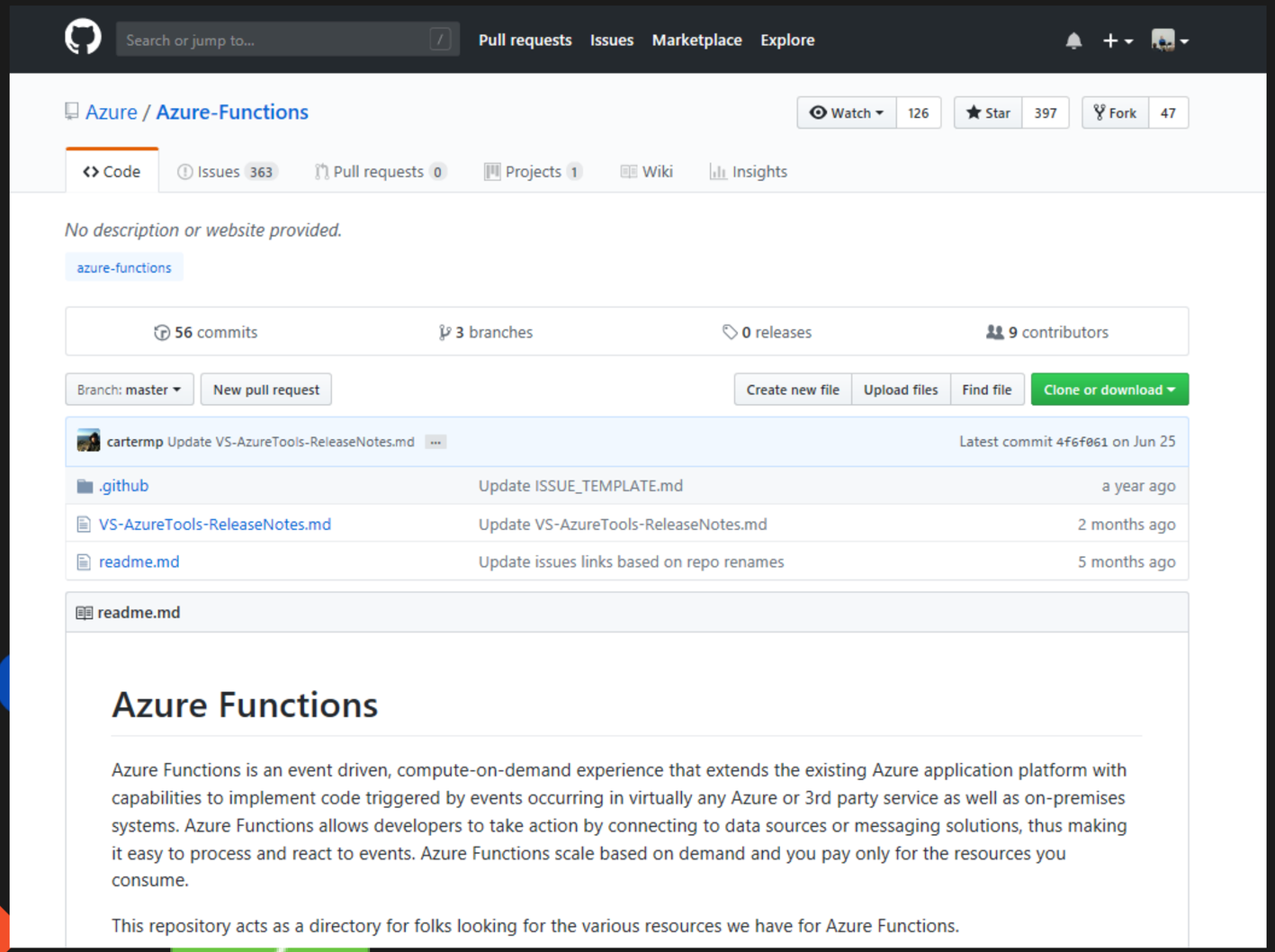


Features of Azure Functions

- Choice of language
- Pay-per-use pricing model
- Bring your own dependencies
- Integrated security
- Simplified integration
- Flexible development

Features of Azure Functions

- Choice of language
- Pay-per-use pricing model
- Bring your own dependencies
- Integrated security
- Simplified integration
- Flexible development
- Open-source



Triggers and Bindings

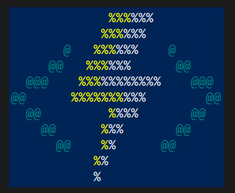
Type	1.x	2.x	Trigger	Input	Output
Blob Storage	✓	✓	✓	✓	✓
Cosmos DB	✓	✓	✓	✓	✓
Event Grid	✓	✓	✓		
Event Hubs	✓	✓	✓		✓
External File	✓			✓	✓
External Table	✓	✓		✓	✓
HTTP	✓	✓	✓		✓
Microsoft Graph Excel tables		✓		✓	✓
Microsoft Graph OneDrive Files		✓		✓	✓
Microsoft Graph Outlook email		✓			✓
Microsoft Graph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

Develop How You Want

- Azure Portal
  - Quickly get started without having to install anything else

- Visual Studio 2017
  - First class C# development experience

- Visual Studio Code
  - First class Node.js development experience
  - Edit any function project generated via CLI



- Azure Functions Core Tools (CLI)
  - Build any kind of function and edit in IDE of your choice

Runtime Versions

Runtime 1.x

- .NET Framework 4.6

Runtime 2.x

- .NET Core 2.0
- Cross Platform
- Language Extensions
  - Java
- Binding Extensions
  - Microsoft Graph
  - Durable Functions

Runtime Version Languages

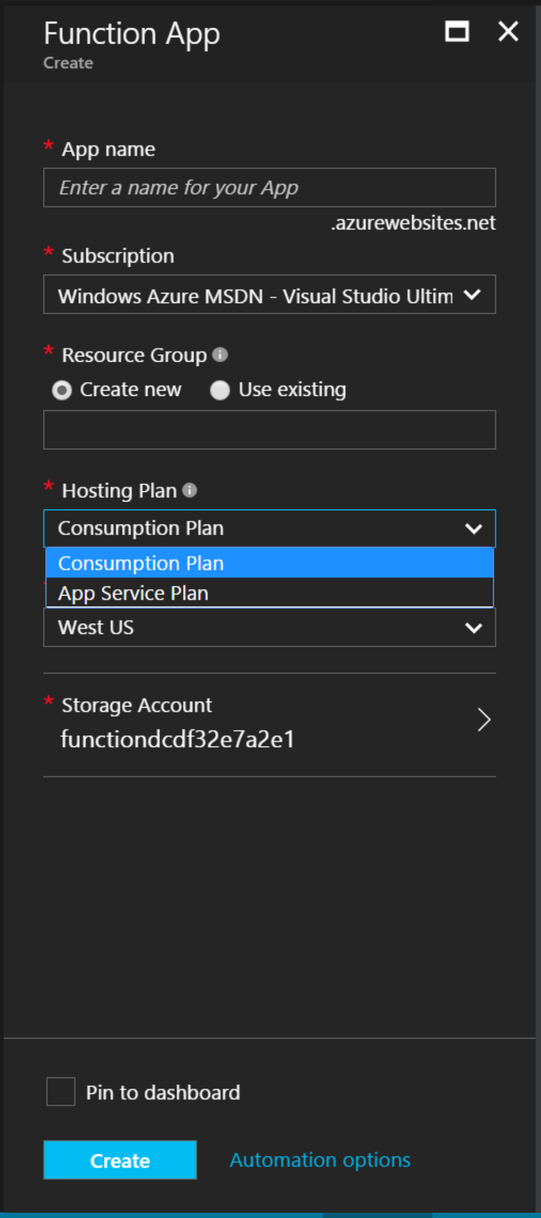
Language	1.x	2.x
C#	GA (.NET Framework 4.7)	GA (.NET Core 2)
JavaScript	GA (Node 6)	GA (Node 8 & 10)
F#	GA(.NET Framework 4.7)	GA (.NET Core 2)
Java	N/A	Preview (Java 8)
Python	Experimental	Preview (Python 3.6)
TypeScript	Experimental	Supported through transpiling to JavaScript
PHP	Experimental	N/A
Batch (.cmd, .bat)	Experimental	N/A
Bash	Experimental	N/A
PowerShell	Experimental	N/A

Consumption Plan

- Pay for what you use without the need to reserve compute resources.
- Function Apps are assigned to compute processing instances that are scaled dynamically by the platform.
- Functions can have multiple parallel executions minimizing the total time needed to process requests.
- Cost is driven by the number of executions and by accounting for memory size used and total execution time across all functions in a Function App as measured in gigabyte-seconds.

Selection guidance

- Good option if your functions run at elastic scale with potentially intermittent executions.

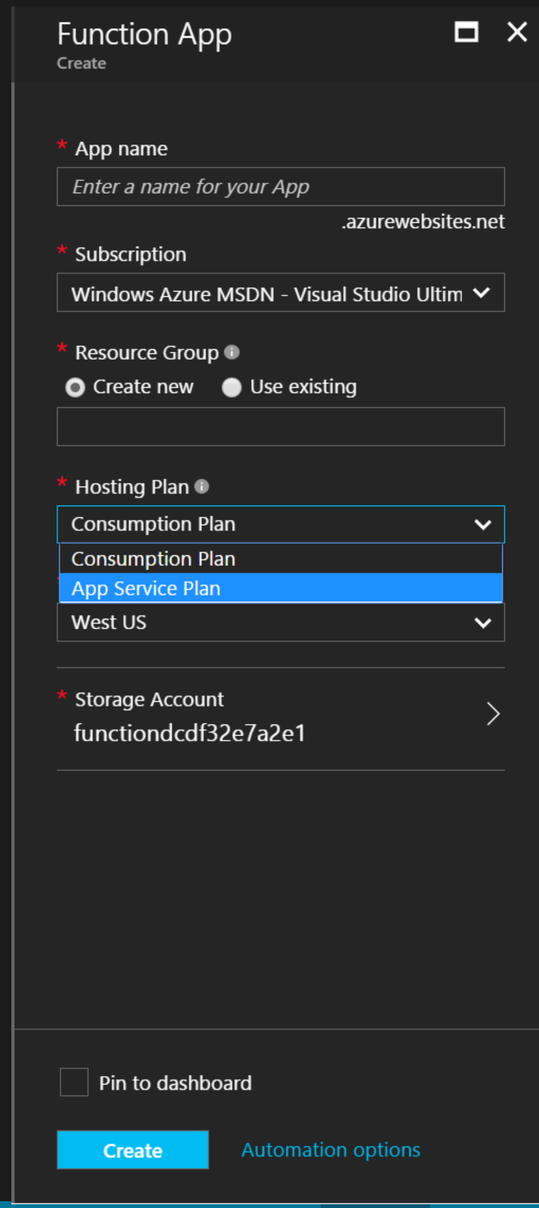


App Service Plan

- Function Apps run on dedicated VMs, just like Web Apps work today
- Dedicated VMs are allocated to your apps and they are always available whether code is being actively executed or not.

Selection guidance

- Good option if you have existing, under-utilized VMs that are already running other code
- Good option if you expect to run functions continuously or almost continuously



Premium Plan

	Consumption Plan 	*-New-* Premium Plan (Preview) 
Instance Size	Fixed at one core and 1.5Gb of memory	Configurable up to 4 cores and 14Gb of memory
Scaling	Event driven scaling	Event driven scaling
Scale Controls	None	Set min and max instances
Private Networking	None	VNET integration
Warmup Time (Cold Start)	Your app must be loaded after it is inactive	No delay after your app is inactive and scale instantly to pre-warmed instances
Cost	Consumption	Consumption and at least 1 pre-warmed instance per plan

Ways to Run Functions

Consumption
Serverless

Pay only for what you use! Metering is per execution and per Gb second.

App Service Plan
Free, Basic, Standard, Premium

All the advantages of Functions with the SLA and ‘always on’ feature of an App Service Plan

App Service Environment
Network Isolation

Your own dedicated cloud environment with network isolation for apps, higher scale, and the ability to connect securely to local vNets.

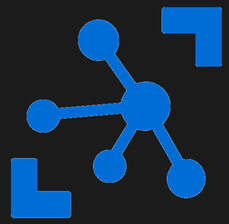
Azure Stack
On Premises

Leverage cloud innovations in on-premises infrastructure. Azure Stack brings the power of Azure to your data centers.

Azure Functions Runtime
Functions on your Server

Run your Azure Functions on our local server (without the rest of Azure)

Azure IoT Edge
On Devices



Run on IoT Devices by deploying custom modules.

Azure Functions in Action

Serverless in Action

Demo: Create an Azure Function from the Portal

Demo: Create an Azure Function from Visual Studio

Visual Studio Installer

Products

Installed

Visual Studio Enterprise 2017

15.6.6

Microsoft DevOps solution for productivity and coordination across teams of any size

Release notes

Modify | Launch | More

Welcome!

We invite you to go online to hone your skills and find additional tools to support your development workflow.

Learn

Whether you're new to development or an experienced developer, we have you covered with our tutorials, videos, and sample code.

Marketplace

Use Visual Studio extensions to add support for new technologies, integrate with other products and services, and fine-tune your experience.

Need some help?

Check out the Microsoft Developer Community where developers provide feedback and answers to many common problems.

Get help from Microsoft at Visual Studio Support.

1.15.3248.309

Visual Studio Installer

Modifying — Visual Studio Enterprise 2017 — 15.6.6

Workloads | Individual components | Language packs

Windows (3)

Universal Windows Platform development
Create applications for the Universal Windows Platform with C#, VB, JavaScript, or optionally C++.

.NET desktop development
Build WPF, Windows Forms, and console applications using C#, Visual Basic, and F#.

Desktop development with C++
Build Windows desktop applications using the Microsoft C++ toolset, ATL, or MFC.

Web & Cloud (7)

ASP.NET and web development
Build web applications using ASP.NET, ASP.NET Core, HTML/JavaScript, and Containers including Docker support.

Azure development
Azure SDKs, tools, and projects for developing cloud apps, creating resources, and building Containers including...

Python development
Editing, debugging, interactive development and source control for Python.

Node.js development
Build scalable network applications using Node.js, an asynchronous event-driven JavaScript runtime.

Summary

- Visual Studio core editor
- Universal Windows Platform development *
- .NET desktop development
- ASP.NET and web development
- Azure development
- Data storage and processing
- .NET Core cross-platform development
- Individual components
  - Visual Studio C++ core features
  - PowerShell Tools for Visual Studio 2017
  - ReadyRoll for VS2017
  - SQL Prompt Core
  - TypeScript 2.3 SDK
  - Arduino IDE for Visual Studio
  - PowerShell Pro Tools for Visual Studio 2017
  - Windows Template Studio
  - VS Live Share - Preview

Location

c:\Program Files (x86)\Microsoft Visual Studio\2017\Enterprise

Total install size: 0 KB

Modify

1.15.3248.309

Deployment and Monitoring

Deployment Options

- Visual Studio
- Functions CLI
- Azure DevOps
- Azure Resource Manager
- Maven / Jenkins

Monitoring Options

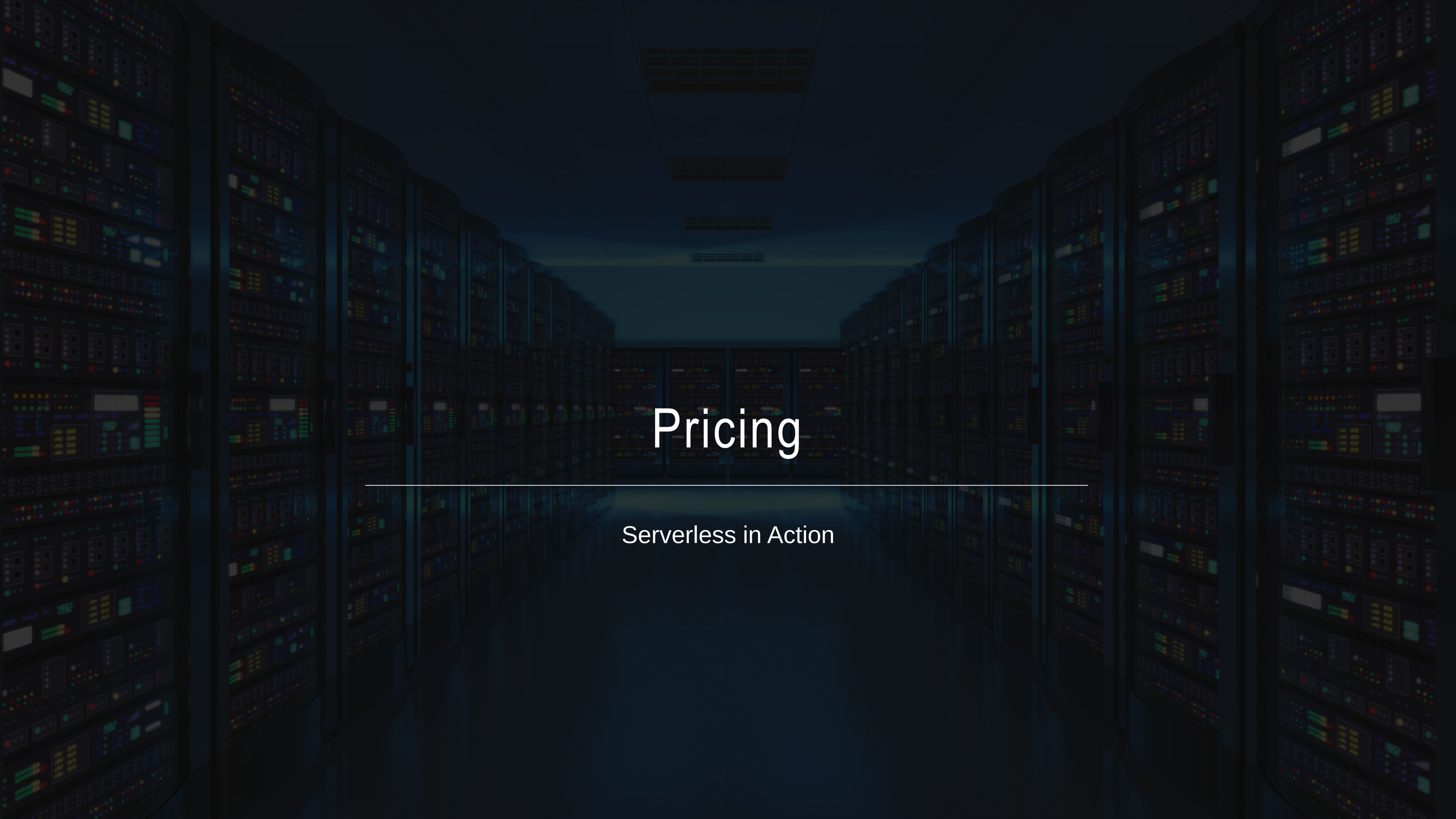
- Azure App Insights
- Function Logs
- Azure Monitor (preview)

Demo: Monitoring a Rapidly Scaling Function

Testing Your Functions

- Recommended Way
  - Abstract logic away from the Function and test that abstraction
- But I really need (want) to test the actual Function
  - Within test project, you will need to create a class that implements the ILogger which will be passed into the Functions

Demo: Testing Your Azure Functions



Pricing

Serverless in Action

Pricing – General Information

- No upfront cost
- No termination fees
- Pay only for what you use

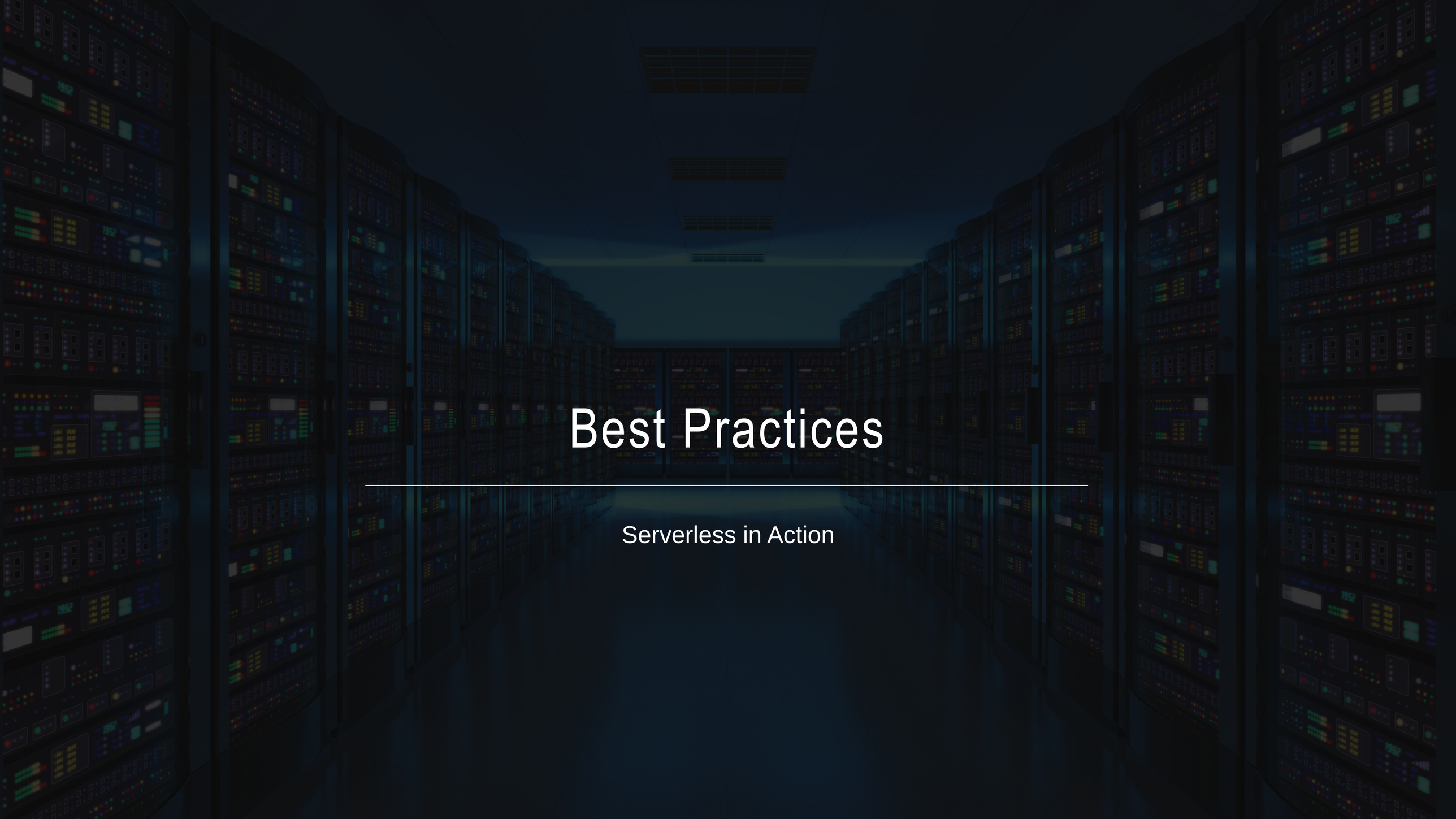
Consumption Plan Pricing

Meter	Price	Free Grant
Execution Time	$0.000016 per Gb-s	400,000 GB-s
Executions	$0.20 per million executions	1 million executions

- Gigabyte-second (GB-s) – Combination of memory size and execution time
- Executions – Each time a function is executed

Pricing Example

- Execution Time
  - 3 million executions x 1 second per execution = 3 million seconds
  - Resource consumption of 512-Mb → 1.5 million GB-s
  - 1.5 million GB-s minus grant of 400,000 Gb-s = 1.1 million Gb-s
  - Execution Total = $17.60
- Executions
  - 3 million executions minus grant of 1 million executions = 2 million executions
  - 2 million transactions at 20 cents per million = $0.40
- Grand Total: $18.00



Best Practices

Serverless in Action

Function Timeouts

- Default timeout of 5 minutes
- Maximum timeout of 10 minutes
- For longer running functions use the App Service Plan and/or Durable Functions

The absolute minimum best practices

- Functions *should* do one thing
- Functions *should* be idempotent
- Functions *should* finish as quickly as possible

General Best Practices

- Avoid long running functions

General Best Practices

- Avoid long running functions
- Cross function communication

General Best Practices

- Avoid long running functions
- Cross function communication
- Write functions to be stateless

General Best Practices

- Avoid long running functions
- Cross function communication
- Write functions to be stateless
- Write defensive functions

Scalability Best Practices

- Do not mix test and production code in the same function app

Scalability Best Practices

- Do not mix test and production code in the same function app
- Use async code but avoid blocking calls

Scalability Best Practices

- Do not mix test and production code in the same function app
- Use async code but avoid blocking calls
- Receive messages in batch whenever possible

Scalability Best Practices

- Do not mix test and production code in the same function app
- Use async code but avoid blocking calls
- Receive messages in batch whenever possible
- Configure host behaviors to better handle concurrency

How to get started

- Start small, replace 1 API or background processing item
- Integration is a great place, often it’s a new layer on top of old layers

Questions

chadgreen@chadgreen.com

chadgreen.com

ChadGreen

ChadwickEGreen

Slides: bit.ly/CM19AzureFunctions