Chad Green

From Zero to Serverless

Stir Trek
May 4, 2018

Who is Chad Green

- Data & Solutions Architect at ProgressiveHealth
- 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

ASK QUESTIONS DURING THE SESSIONS!

slack

THERE IS A SEPARATE CHANNEL FOR EACH TRACK!

#2018--RED [✓ #2018--ORANGE] #2018--YELLOW #2018--GREEN

#2018--BLUE #2018--PURPLE #2018--THANOS #2018--GAUNTLET

What is Serverless Computing

From Zero to Serverless

The evolution of application platforms

On-Premises

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**?

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

How can I dynamically configure my app?

The evolution of application platforms

IaaS

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?

The evolution of application platforms

PaaS

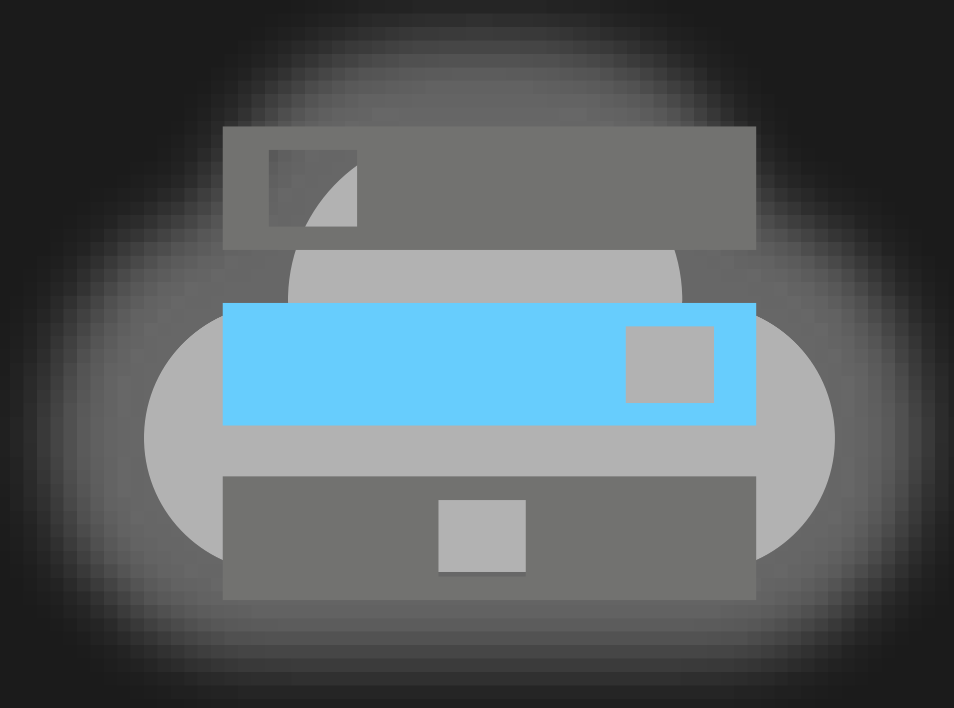
What is the right **size** of servers for my business needs?

How can I increase **server** utilization?

How many **servers** do I need?

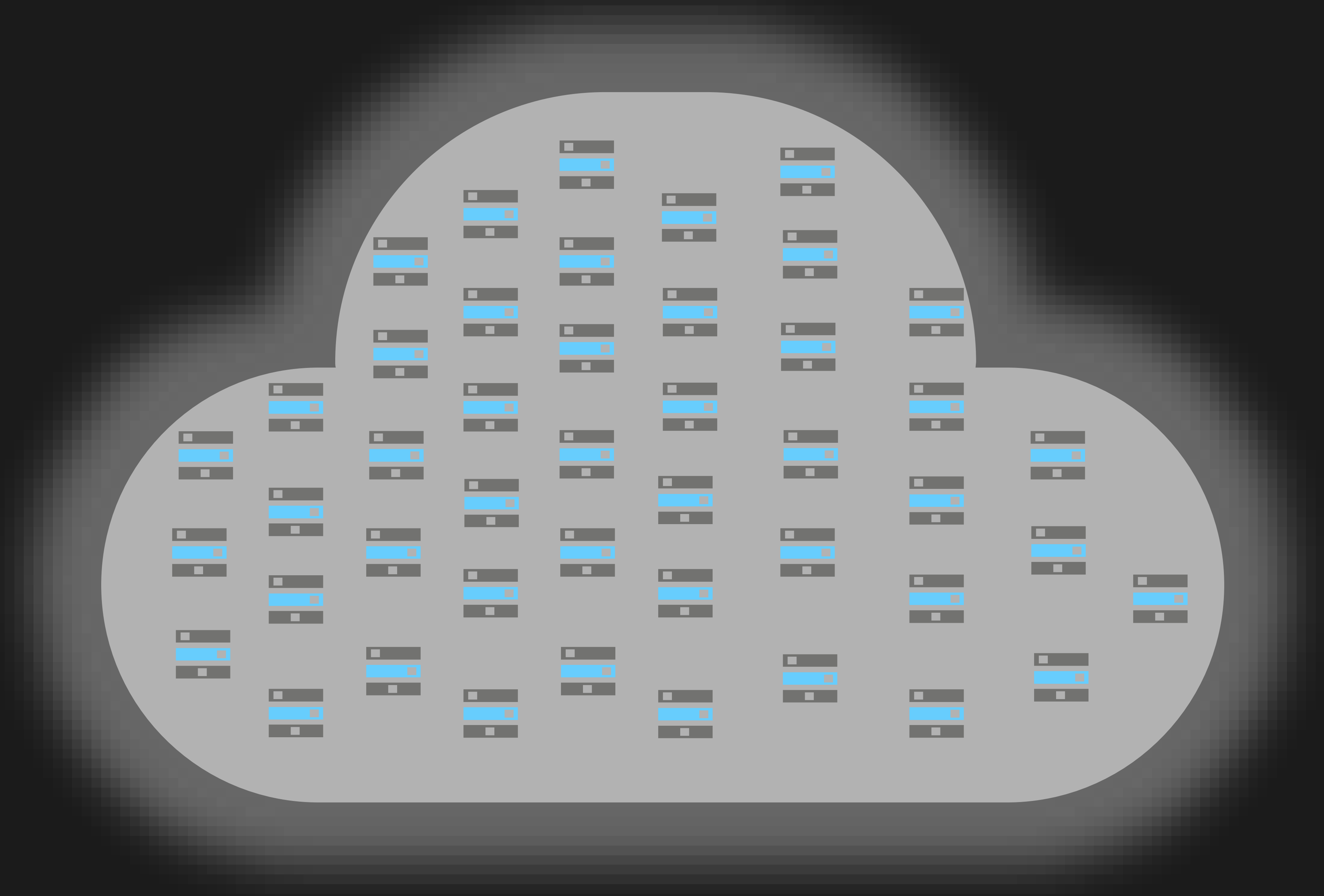
How can I **scale** my application?

⍰



The evolution of application platforms

Serverless



The platform for next generation applications

What is Serverless?

Area #1 Backend as a Service (BaaS)

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

Area #2 Functions as a Service (FaaS)

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

What is Serverless?

Abstraction of Servers

Event-Driven/Instant Scale

Micro-Billing

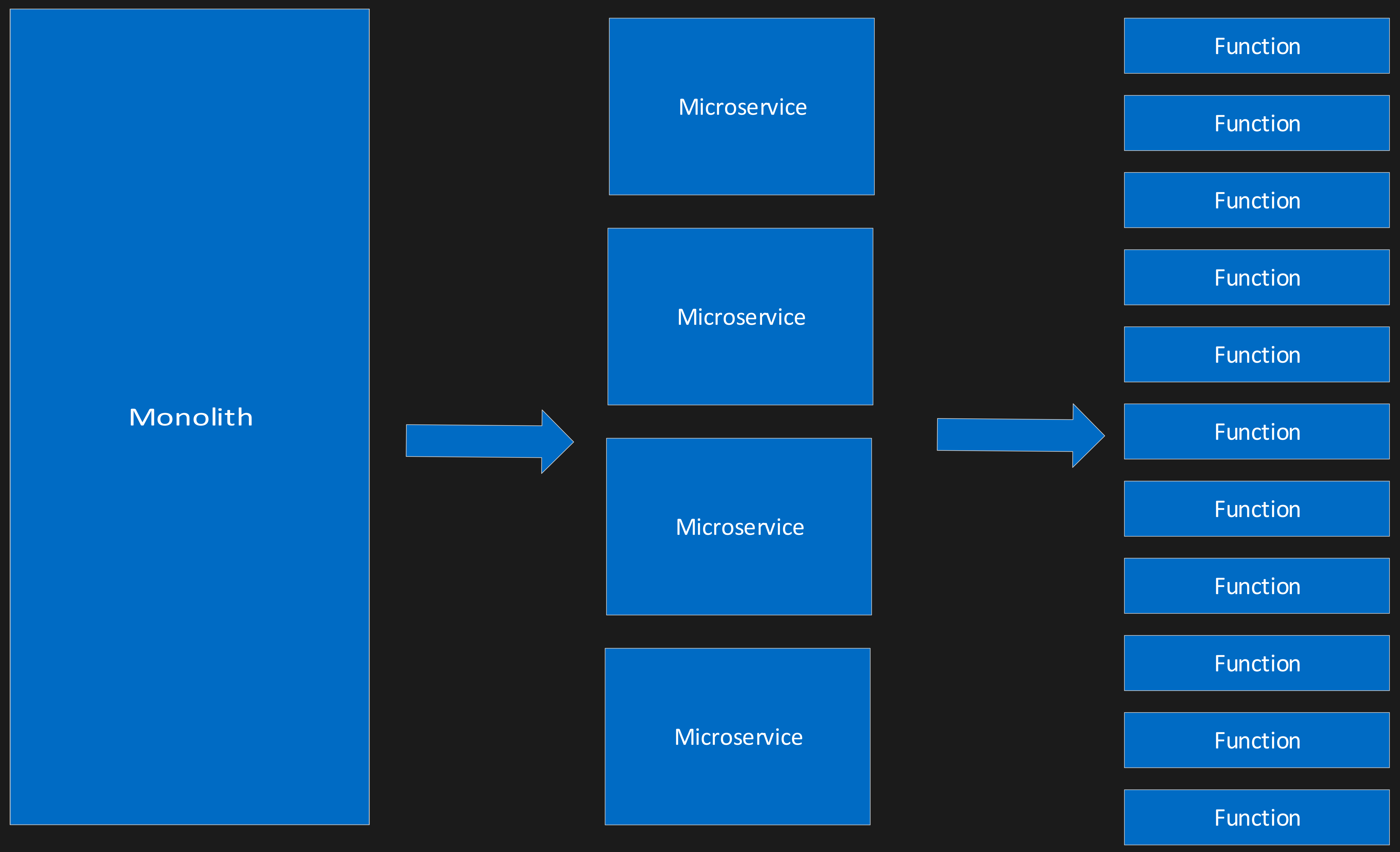
Benefits of Serverless

Manage apps not servers

Reduced DevOps

Faster Time to Market

Serverless Scale



Nano Services

Challenges of Serverless Architecture

Complexity

Organizational Support

No Runtime Optimization

Serverless Options

From Zero to Serverless

Serverless Options

- ~~Zimki~~
- Google Cloud Functions
- Amazon Lambda
- IBM Cloud Functions
- Auth0 WebTask
- Azure

Azure Serverless

Functions

Execute your code based on events you specify

Logic Apps

Design workflows and orchestrate processes

Event Grid

Manage all events that can trigger code or logic

Azure Serverless

Functions

Execute your code based on events you specify

Logic Apps

Design workflows and orchestrate processes

Event Grid

Manage all events that can trigger code or logic

Database

Storage

Security

IoT

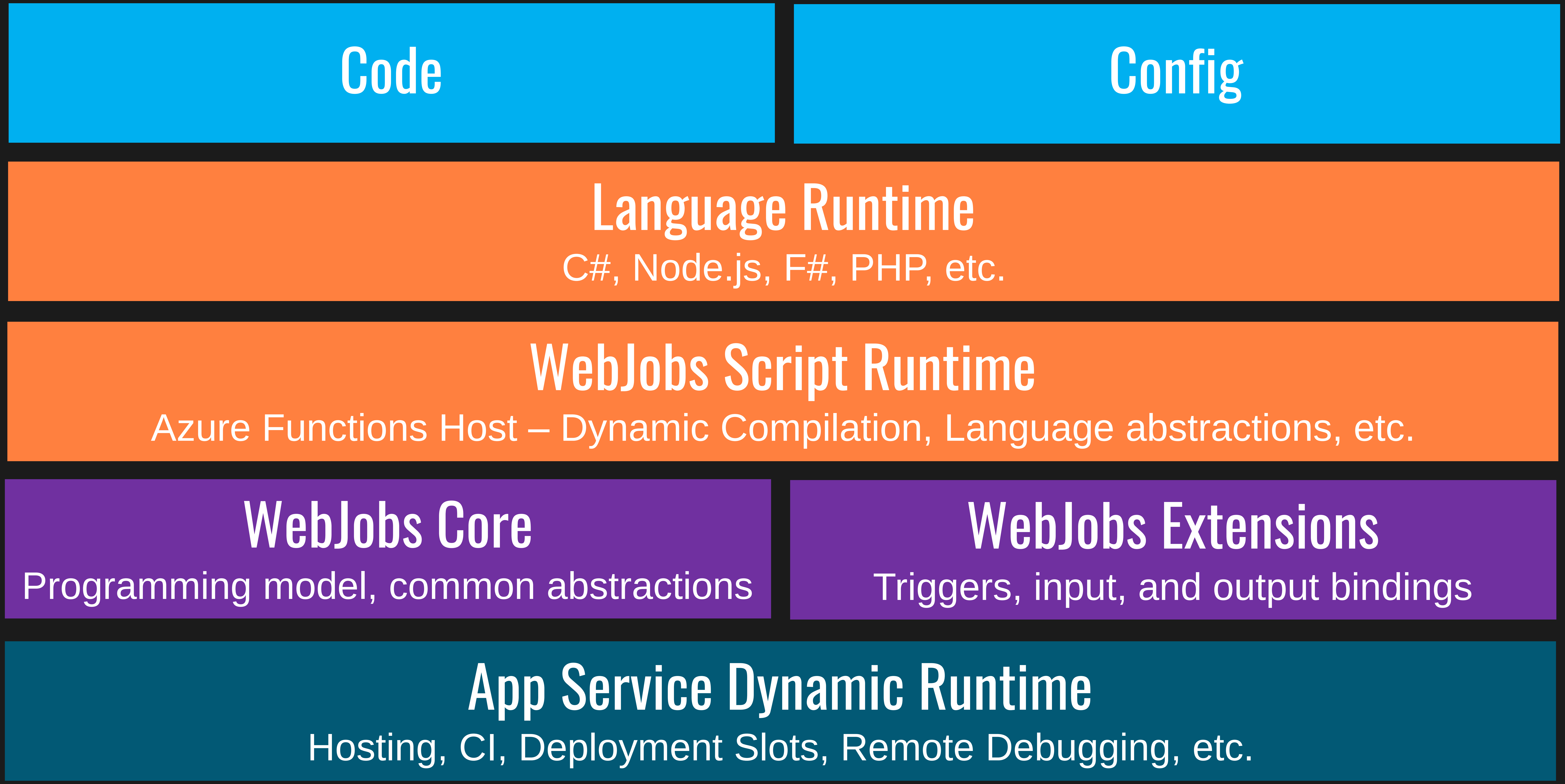
Analytics

Intelligence

Azure Functions

From Zero to Serverless

Azure Functions Architecture



Code

Config

Language Runtime

C#, Node.js, F#, PHP, etc.

WebJobs Script Runtime

Azure Functions Host – Dynamic Compilation, Language abstractions, etc.

WebJobs Core

Programming model, common abstractions

WebJobs Extensions

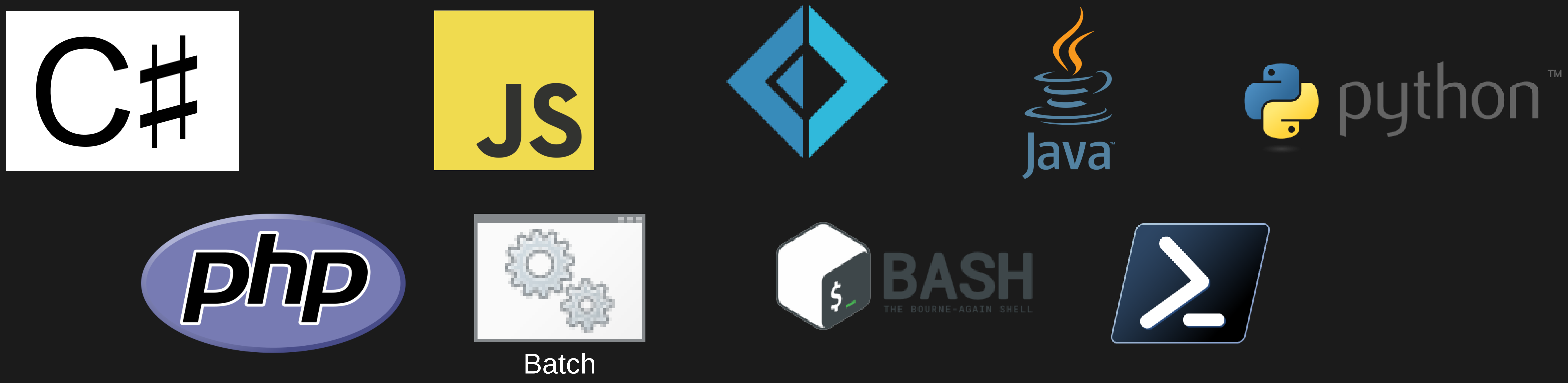
Triggers, input, and output bindings

App Service Dynamic Runtime

Hosting, CI, Deployment Slots, Remote Debugging, etc.

Features of Azure Functions

- Choice of language



Features of Azure Functions

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

Features of Azure Functions

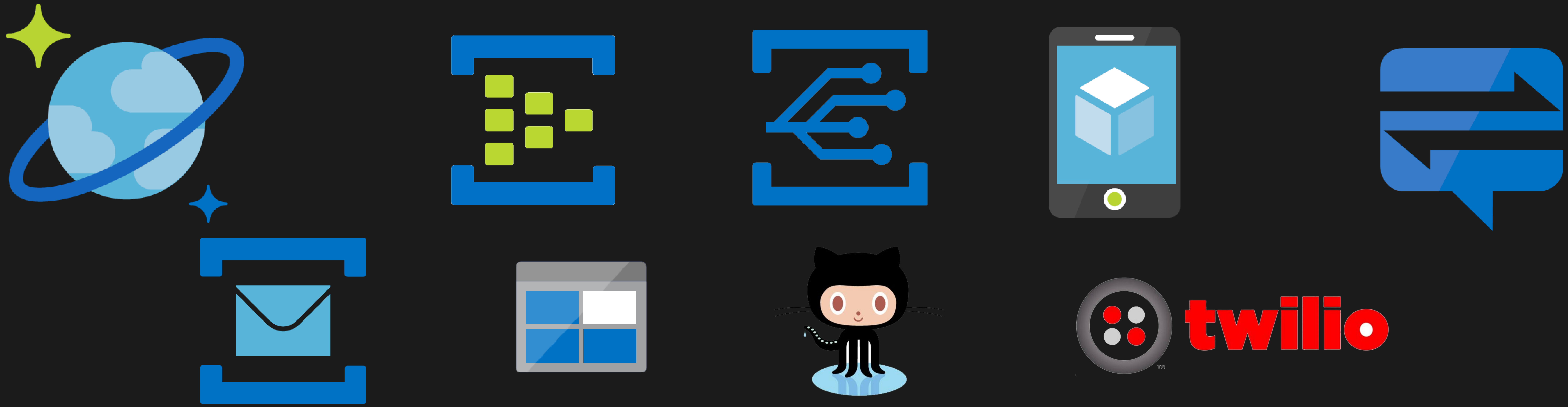
- Choice of language
- Pay-per-use pricing model
- Bring your own dependencies

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

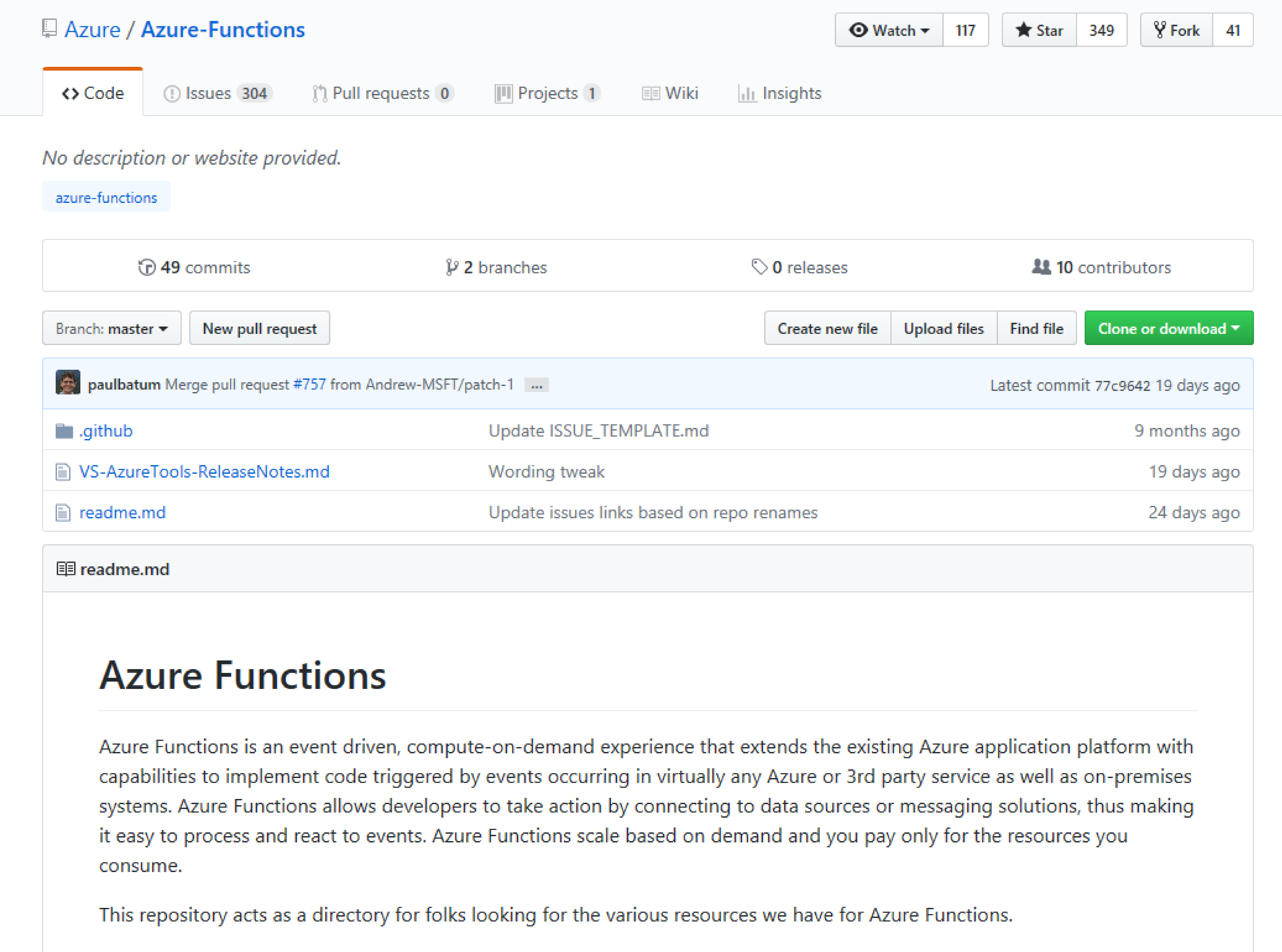


Features of Azure Functions

- 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



No description or website provided.

paulbatum Merge pull request #757 from Andrew-MSFT/patch-1 … Latest commit 77c9642 19 days ago

.github	Update ISSUE_TEMPLATE.md	9 months ago
VS-AzureTools-ReleaseNotes.md	Wording tweak	19 days ago
readme.md	Update issues links based on repo renames	24 days ago

readme.md

Azure Functions

Azure Functions is an event driven, compute-on-demand experience that extends the existing Azure application platform with capabilities to implement code triggered by events occurring in virtually any Azure or 3rd party service as well as on-premises systems. Azure Functions allows developers to take action by connecting to data sources or messaging solutions, thus making it easy to process and react to events. Azure Functions scale based on demand and you pay only for the resources you consume.

This repository acts as a directory for folks looking for the various resources we have for Azure Functions.

What can you do with Functions

- Processing data, integrating systems, working with IoT, simple API’s, and microservices
- Templates for a number of solution possibilities

Triggers and Bindings

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			⍰

Runtime Versions

Runtime 1.x

- .NET Framework 4.6
- Generally Available

Runtime 2.x (Preview)

- .NET Core 2.0
- Cross Platform

Runtime Versions

Runtime 1.x

- .NET Framework 4.6
- Generally Available

Runtime 2.x (Preview)

- .NET Core 2.0
- Cross Platform
- Language Extensions
    - Java

Runtime Versions

Runtime 1.x

- .NET Framework 4.6
- Generally Available

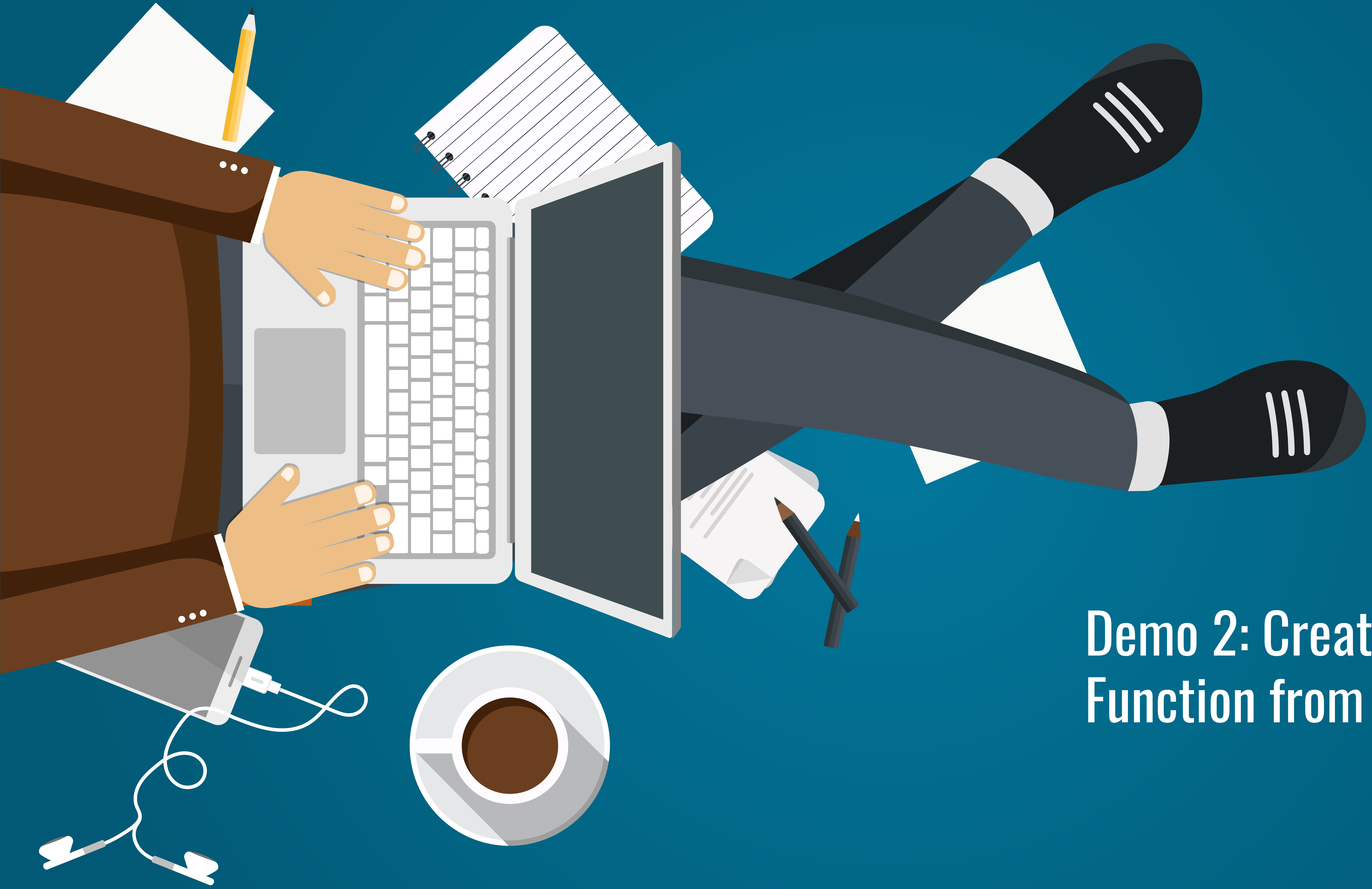
Runtime 2.x (Preview)

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

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

Demo 1: Create an Azure Function from the Portal



Demo 2: Create an Azure Function from Visual Studio

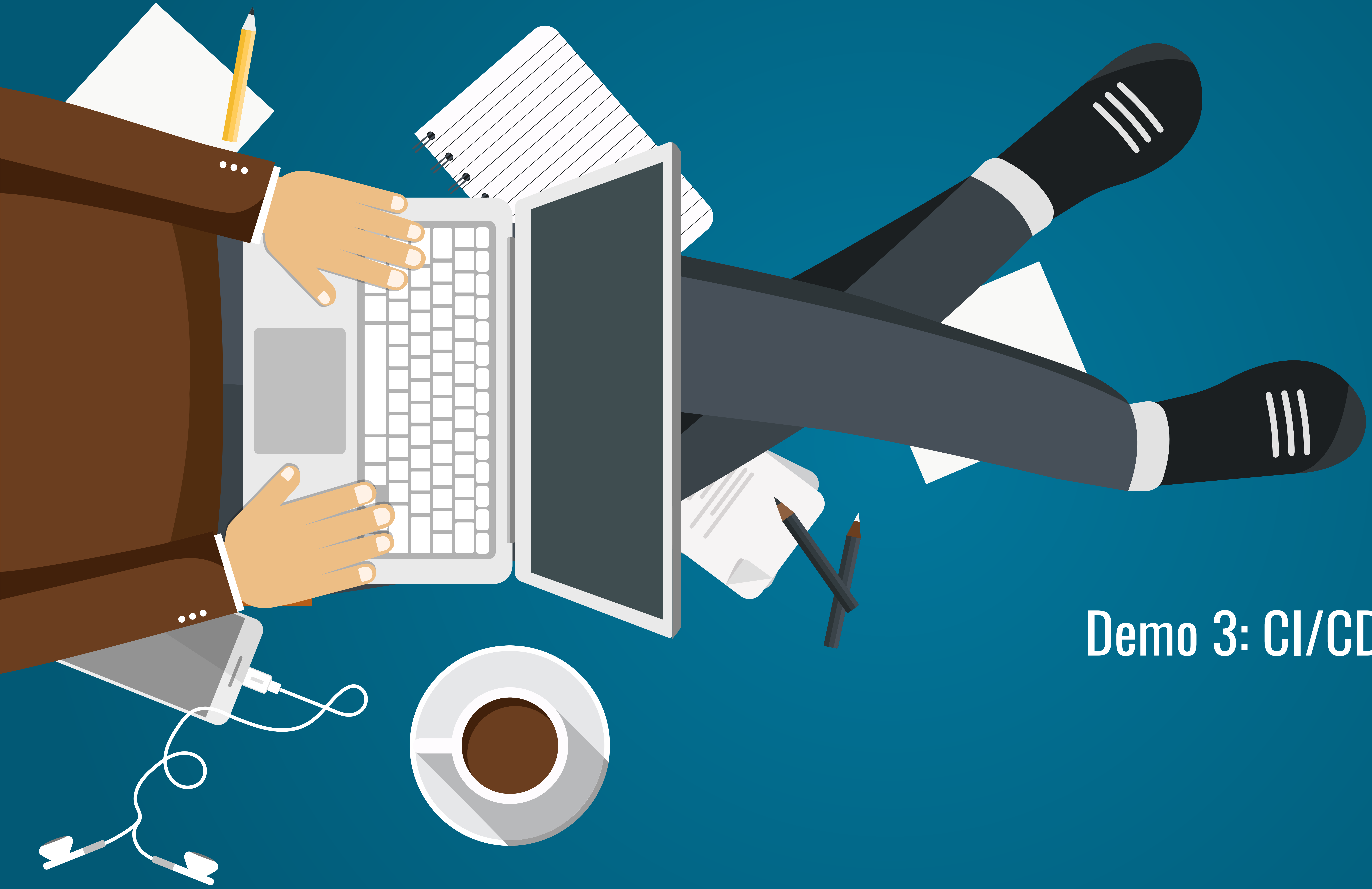
Deployment and Monitoring

Deployment Options

- Visual Studio
- Functions CLI
- Visual Studio Team Services
- Azure Resource Manager
- Maven / Jenkins

Monitoring Options

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



Demo 3: CI/CD

Proxies

- Provide more control over all functions or just select methods
- Can point to any HTTP resource

Take our current function url:

https://stirtrek.azurewebsites.net/api/HttpTriggerCSharp1?code=k9as3MKuDEAOyj3GbniZgJjWrn1cMqTAcDhbzqgAldUcYk67EX8QVg==&name=Stir%20Trek%20Attendees

Our function URL would then be like this:

https://stirtrek.azurewebsites.net/HelloWorld/{name}

Demo 4: Setting up routing and proxies

Securing your Azure Functions

- HTTPTriggers can be protected by OAuth providers
  - Azure Active Directory
  - Microsoft Account
  - Facebook
  - Google
  - Twitter

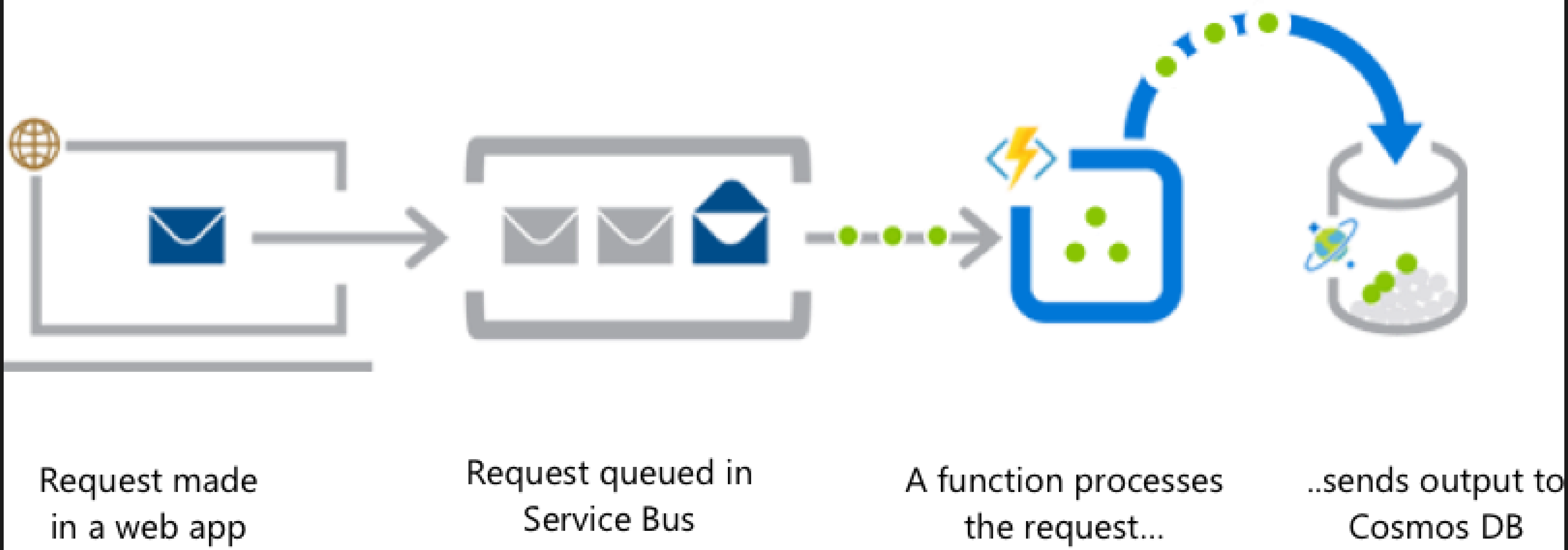
Function Timeouts

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

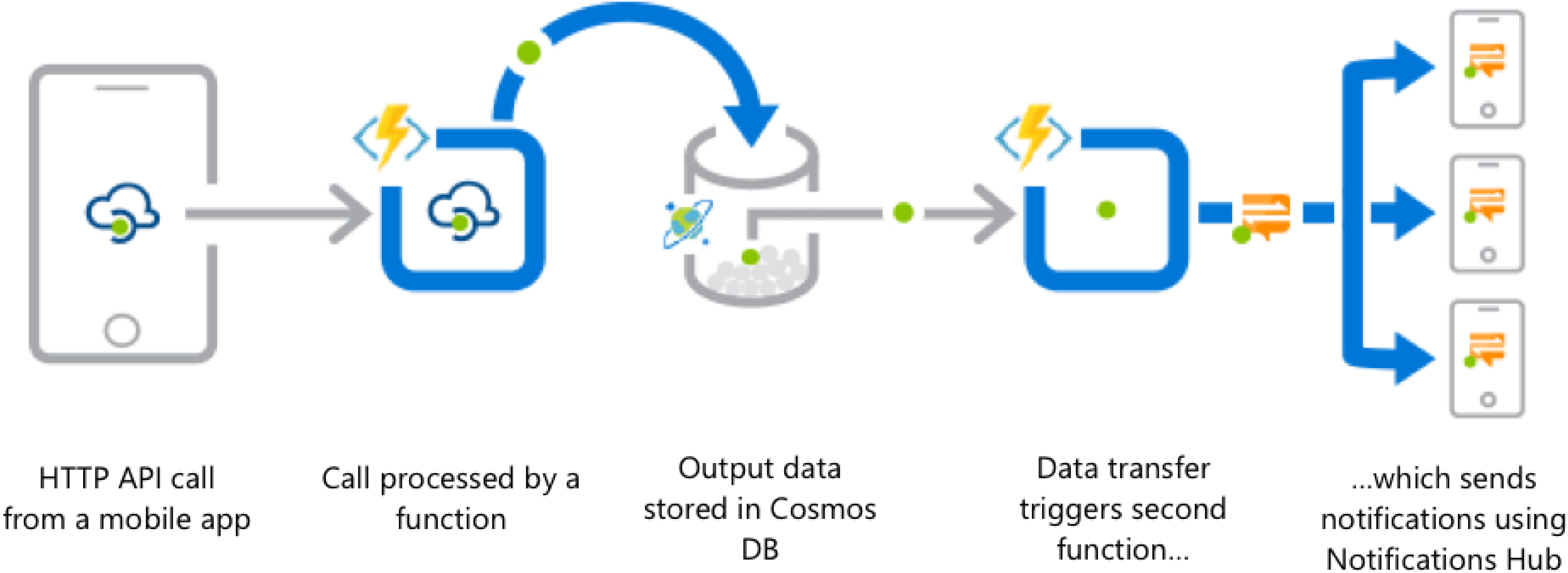
Common Scenarios

From Zero to Serverless

Web Application Backends



Web Application Backends



HTTP API call from a mobile app

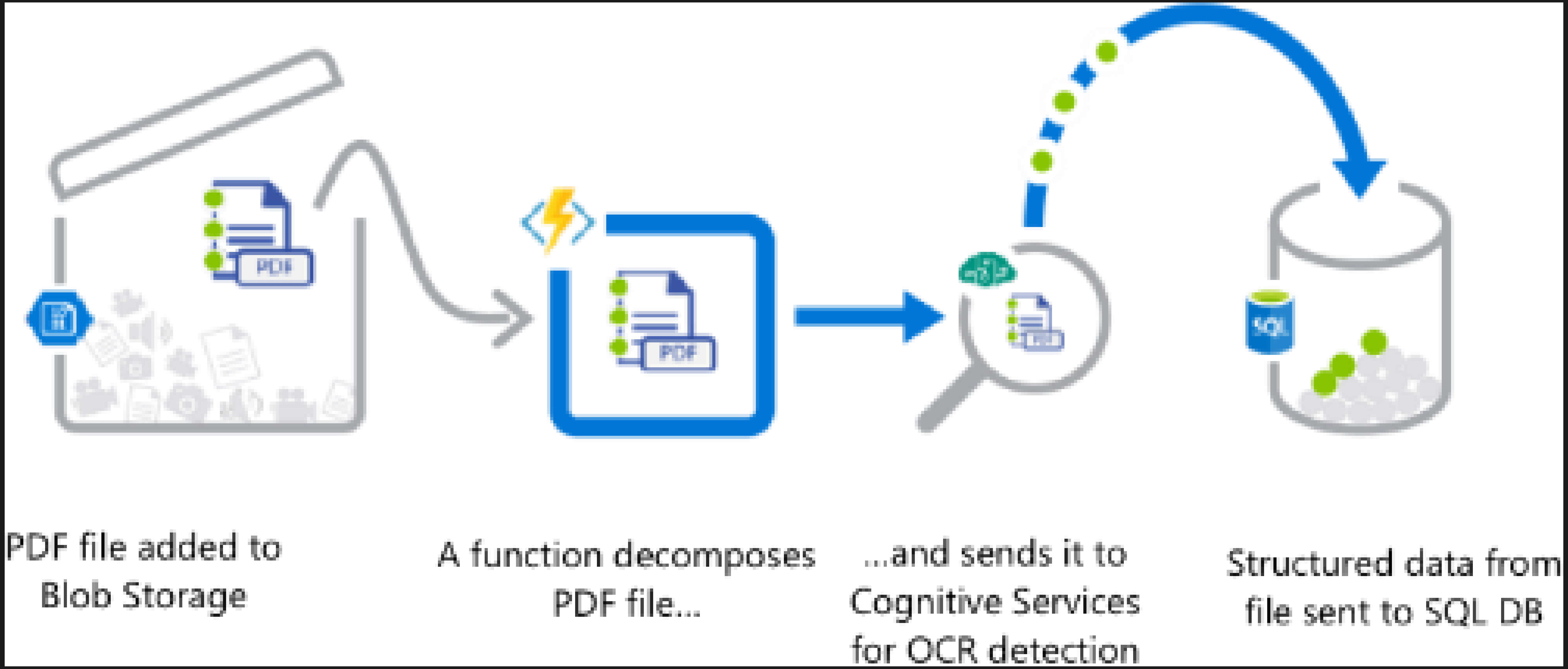
Call processed by a function

Output data stored in Cosmos DB

Data transfer triggers second function...

...which sends notifications using Notifications Hub

Real-Time File Processing



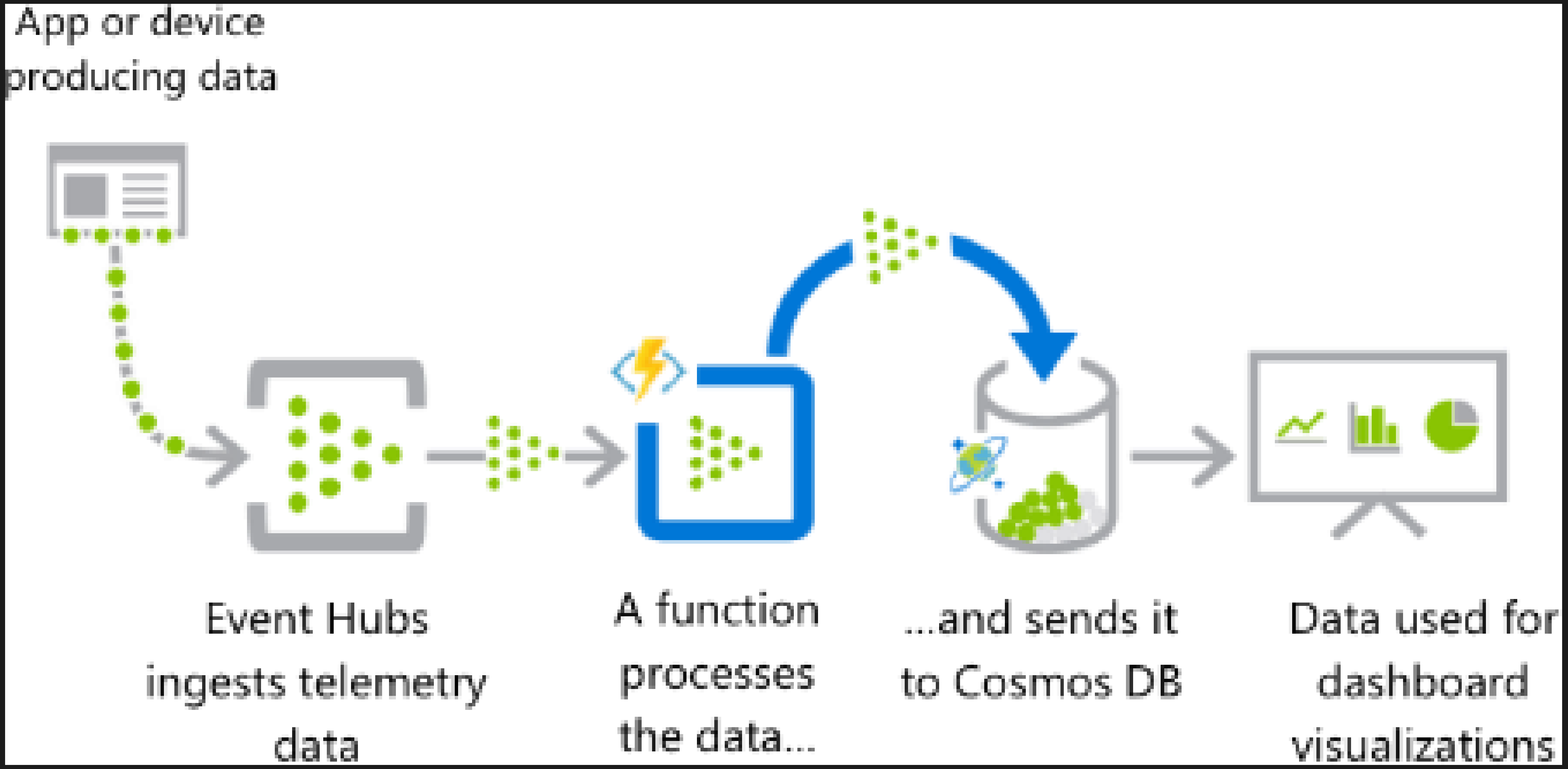
PDF file added to Blob Storage

A function decomposes PDF file...

...and sends it to Cognitive Services for OCR detection

Structured data from file sent to SQL DB

Real-Time Stream Processing



App or device producing data

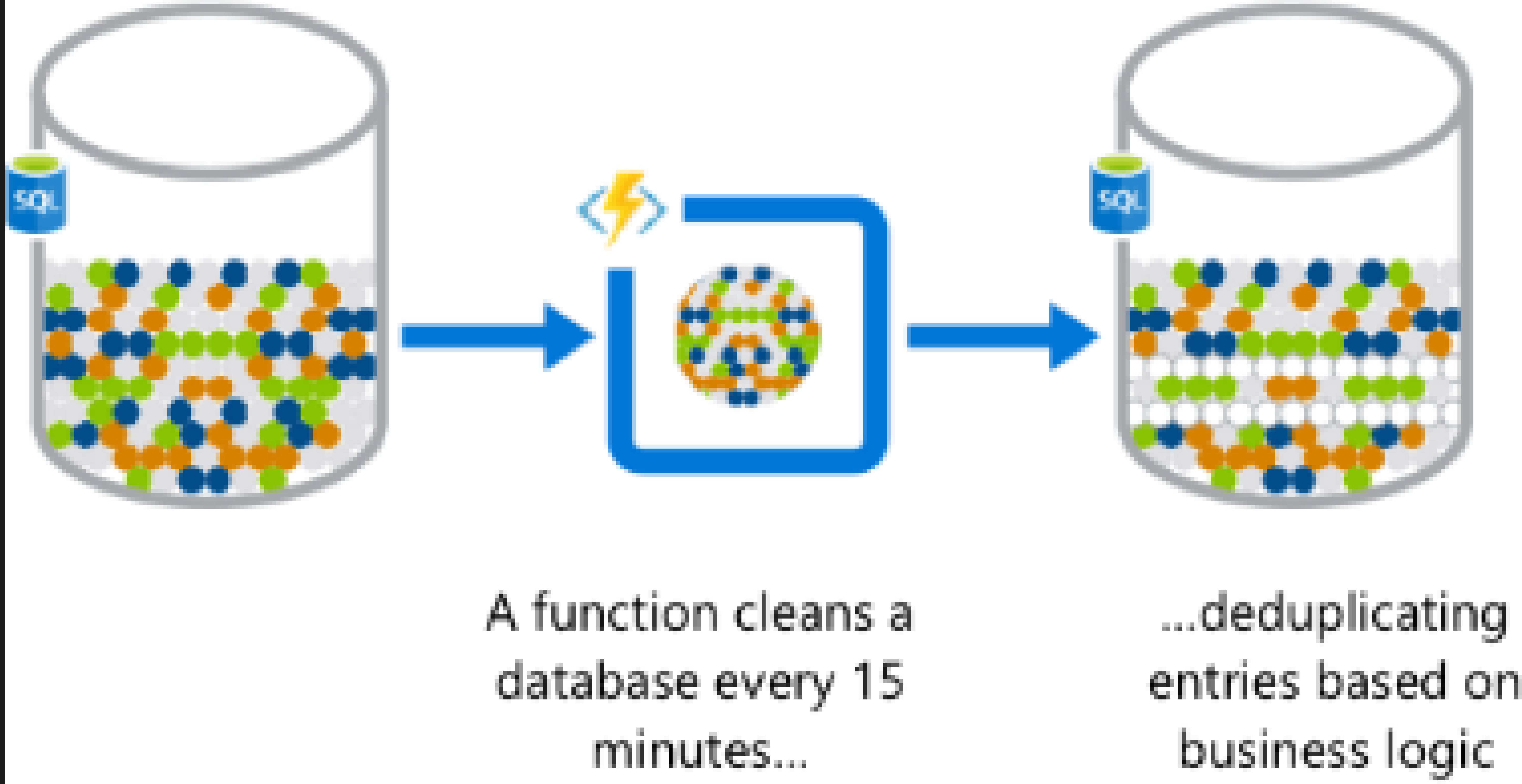
Event Hubs ingests telemetry data

A function processes the data...

...and sends it to Cosmos DB

Data used for dashboard visualizations

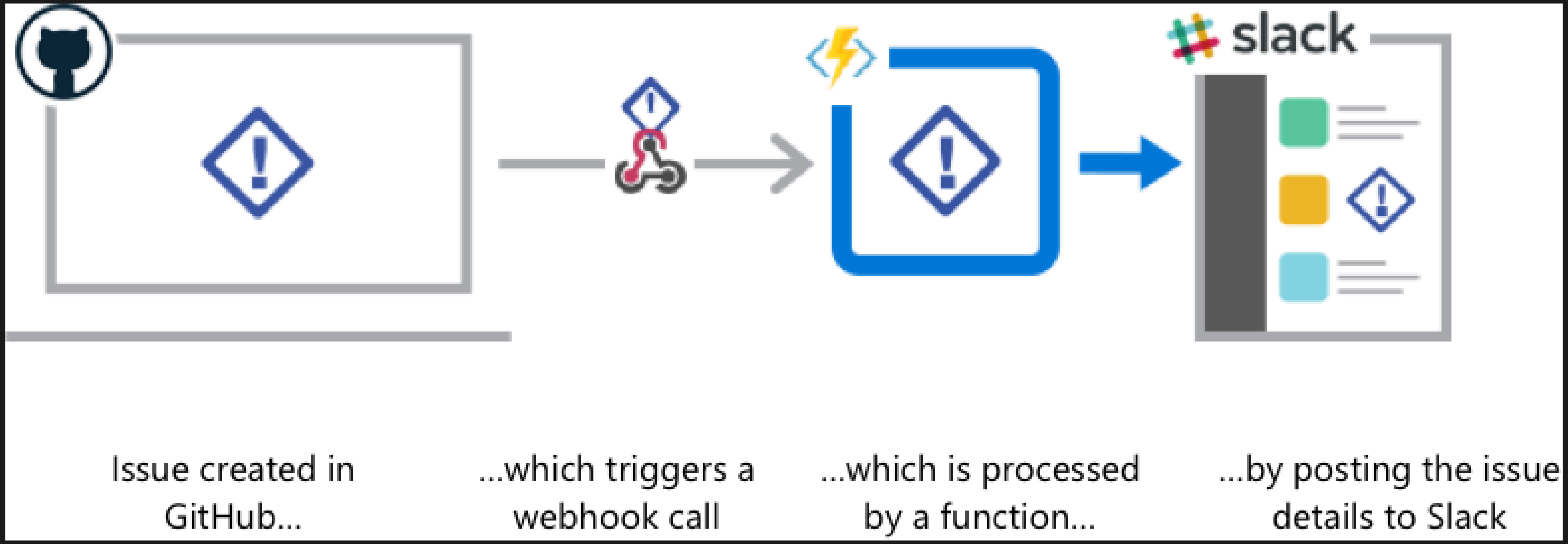
Automation of Scheduled Tasks



A function cleans a database every 15 minutes...

...deduplicating entries based on business logic

Extending SaaS Applications


slack
Issue created in GitHub...
...which triggers a webhook call
...which is processed by a function...
...by posting the issue details to Slack

Pricing

From Zero to Serverless

Pricing – General Information

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

Pricing – Two Different Pricing Plans

Consumption Plan

- Takes care of everything but your code
- Pay only when your functions are running
- Scale out automatically

App Service Plan

- You pretty much take care of everything
- Consider when:
  - Existing, underutilized VMs
  - Function apps to run continuously
  - More CPU or memory options
  - Run longer than maximum execution time
  - Require features only available on App Service plan
  - Want to run on Linux (on general availability tier)

Pricing – Consumption Plan Details

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 – Consumption Plan Details

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

From Zero to Serverless

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

Where 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

bit.ly/FZtS0518