



# Serverless and Azure Functions

Build apps faster with serverless technologies

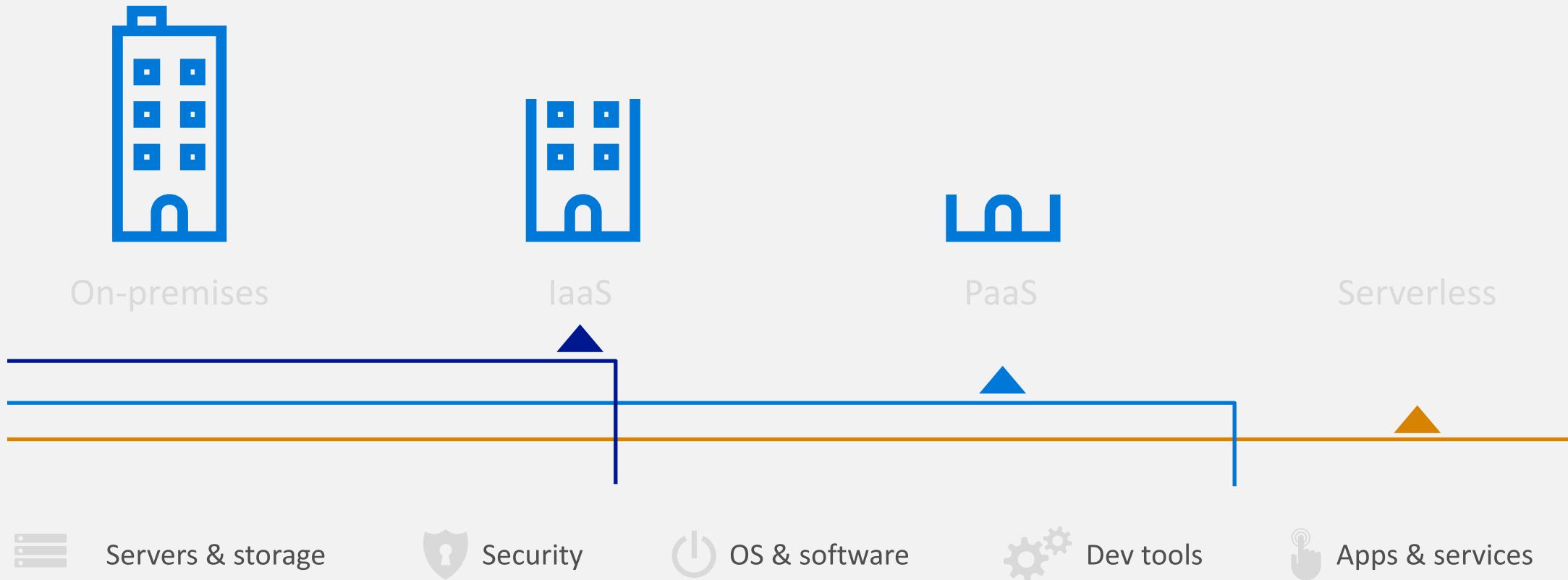


<https://aka.ms/serverless-java>

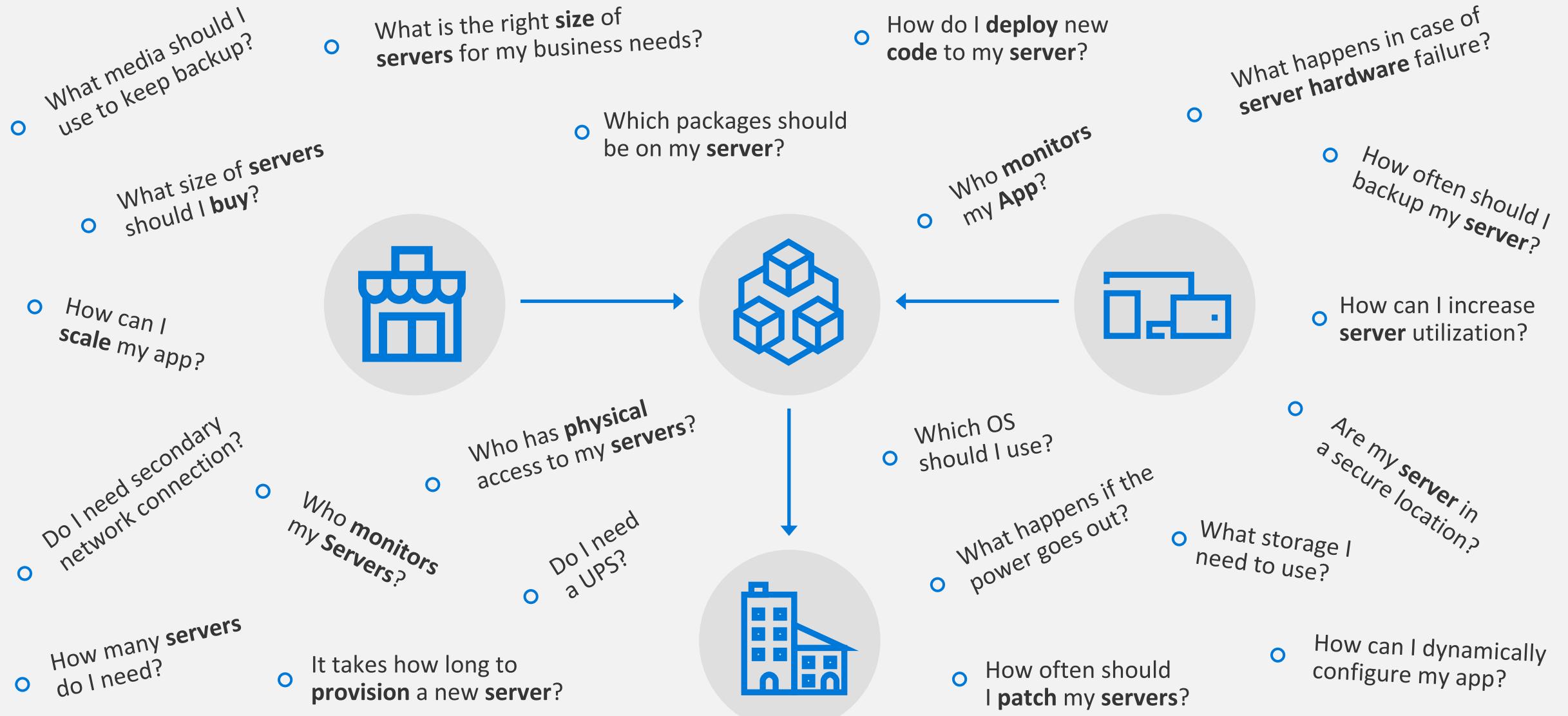


# History of cloud development

Increasingly advanced cloud technologies have led companies to entrust more and more of their IT activities to service providers



# Before cloud...



# ...then IaaS set the table stakes for digital business...

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 app?



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 OS should I use?

Who monitors my App?

# ...then PaaS, critical for digital transformation

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

How can I increase “server” utilization?

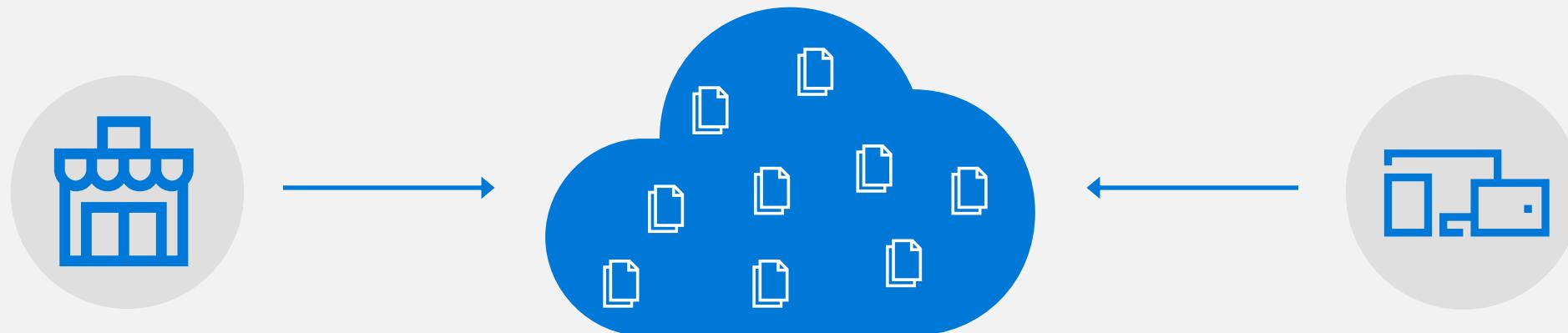
How many “servers” do I need?

How can I scale my app?



# Serverless: the platform for next gen apps, today

How do I **architect** my app to become Serverless?





# What is serverless?



## Full abstraction of servers

Developers can just focus on their code—there are no distractions around server management, capacity planning, or availability.



## Instant, event-driven scalability

Application components react to events and triggers in near real-time with virtually unlimited scalability; compute resources are used as needed.



## Pay-per-use

Only pay for what you use: billing is typically calculated on the number of function calls, code execution time, and memory used.\*

# What are the benefits?



Focus

Solve business problems—not technology problems related to undifferentiated heavy lifting



Efficiency

Shorter time to market  
Fixed costs converted to variable costs  
Better service stability  
Better development and testing management  
Less waste



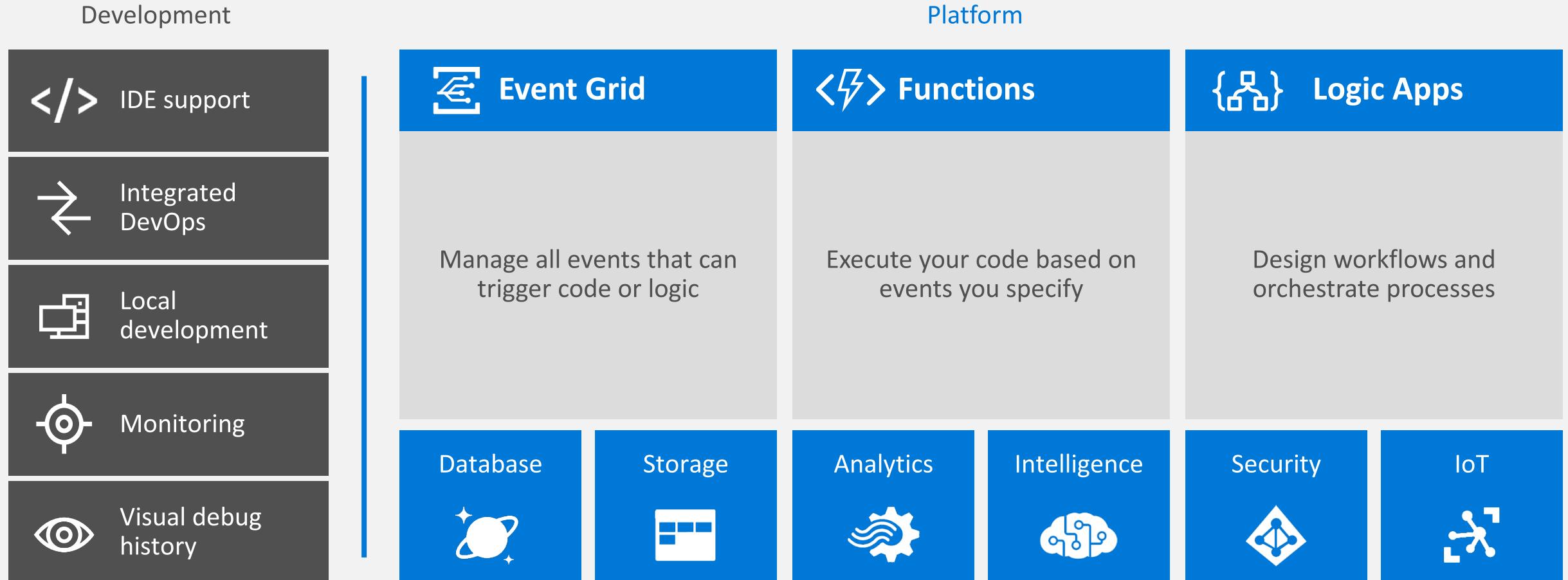
Flexibility

Simplified starting experience  
Easier pivoting means more flexibility  
Easier experimentation  
Scale at your pace—don't bet the farm on Day 1  
Natural fit for microservices



# Full integration with Azure ecosystem

Functions is the center piece of the Serverless platform



# FaaS is at the center of serverless

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



## Single responsibility

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



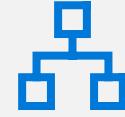
## Short lived

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

# What is Azure Functions?

An event-based, serverless compute experience that accelerates app development

## Azure Functions = FaaS++



### Integrated programming model

Use built-in triggers and bindings to define when a function is invoked and to what data it connects



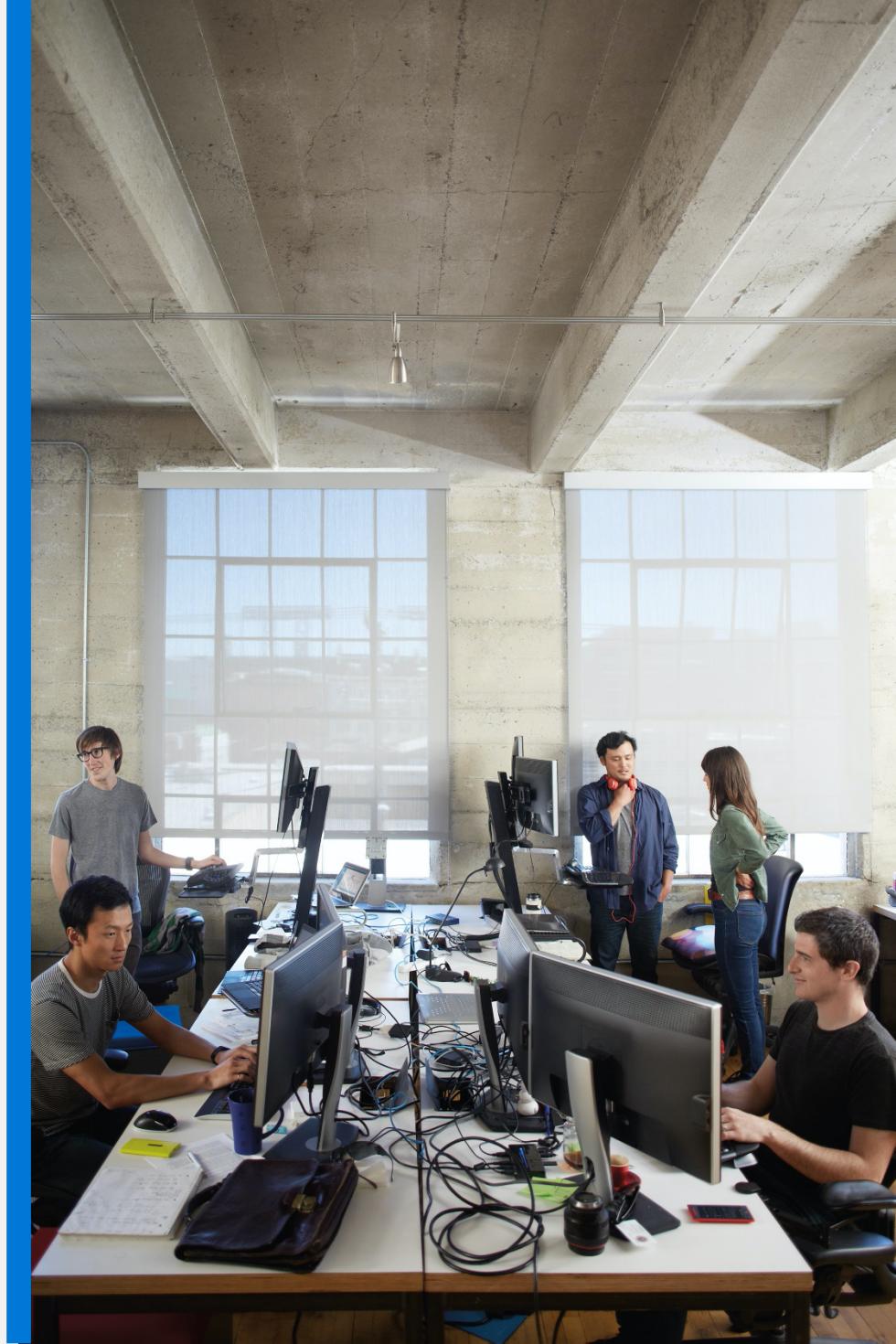
### Enhanced development experience

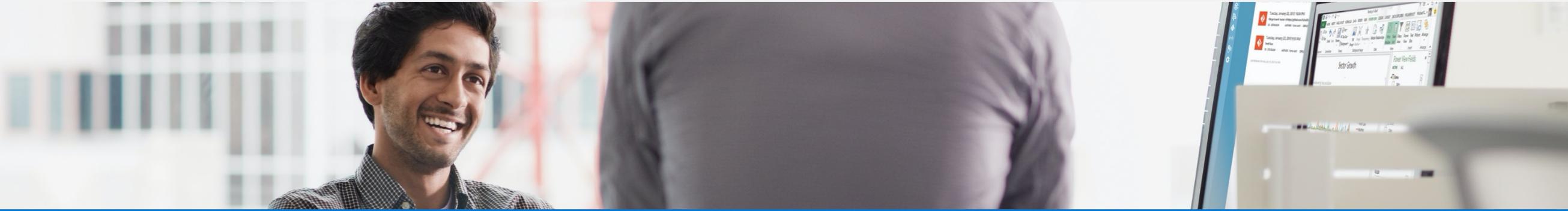
Code, test and debug locally using your preferred editor or the easy-to-use web-based interface including monitoring



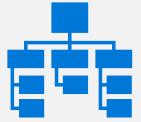
### Hosting options flexibility

Choose the deployment model that better fits your business needs without compromising development experience





# Focus on code, not plumbing



No infrastructure  
management



Auto-scale based  
on your workload



No wasted resources,  
pay only for what you use



# Boost development efficiency



## Triggers

Use triggers to define how functions are invoked  
Avoid hardcoding with preconfigured JSON files  
Build serverless APIs using HTTP triggers



## Proxies

Define one API surface for multiple function apps  
Create endpoints as reverse proxies to other APIs  
Condition proxies to use variables



## CI/CD

Save time with built-in DevOps  
Deploy functions using App Service for CI  
Leverage Microsoft, partner services for CD



## Bindings

Connect to data with input and output bindings  
Bind to Azure solutions and third-party services  
Use HTTP bindings in tandem with HTTP triggers



## Local debugging

Debug C# and JavaScript functions locally  
Use debugging tools in Azure portal, VS, and VS Code

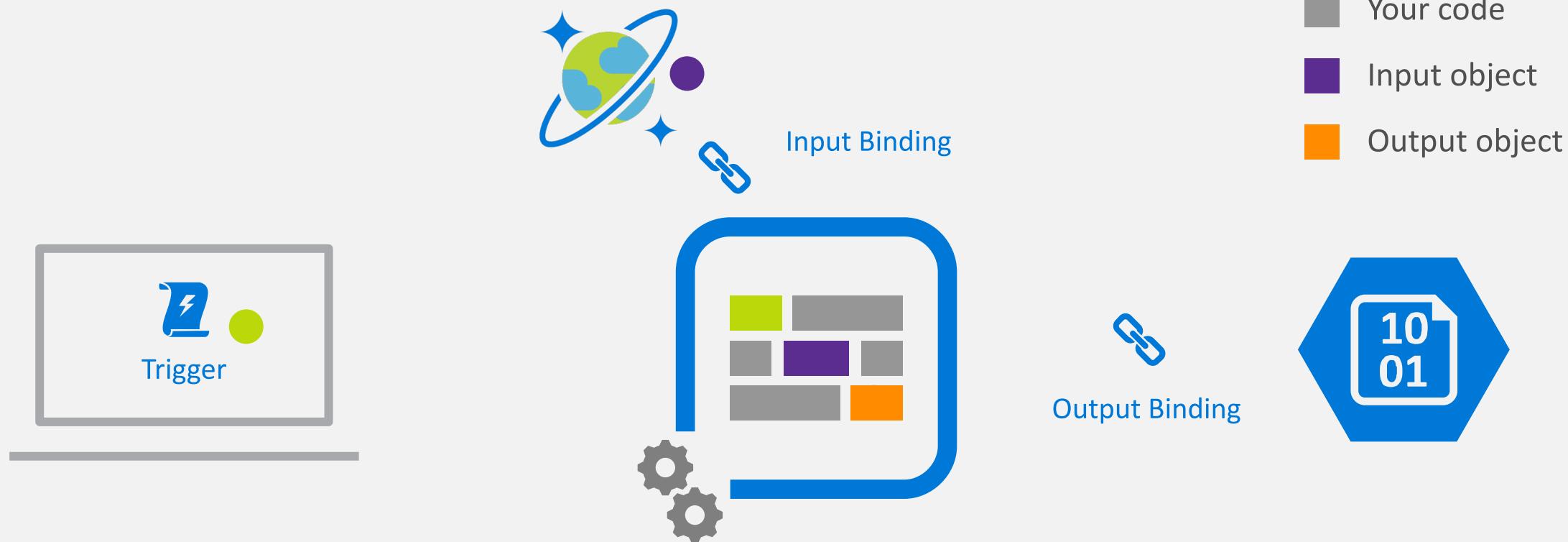


## Monitoring

Integrate with Azure Application Insights  
Get near real-time details about function apps  
See metrics around failures, executions, etc.



# Boost development efficiency





# Gain **flexibility** and develop your way



## Multiple languages

Write code in C#, JavaScript, F#, and Java  
Continuous investment in new, experimental languages



## Hosting options

Choose from six consumption plans to run Functions  
Run your first million function executions for free



## Durable Functions

Write stateful functions in a serverless environment  
Simplify complex, stateful coordination problems  
Add the extension to enable advanced scenarios



## Dev options

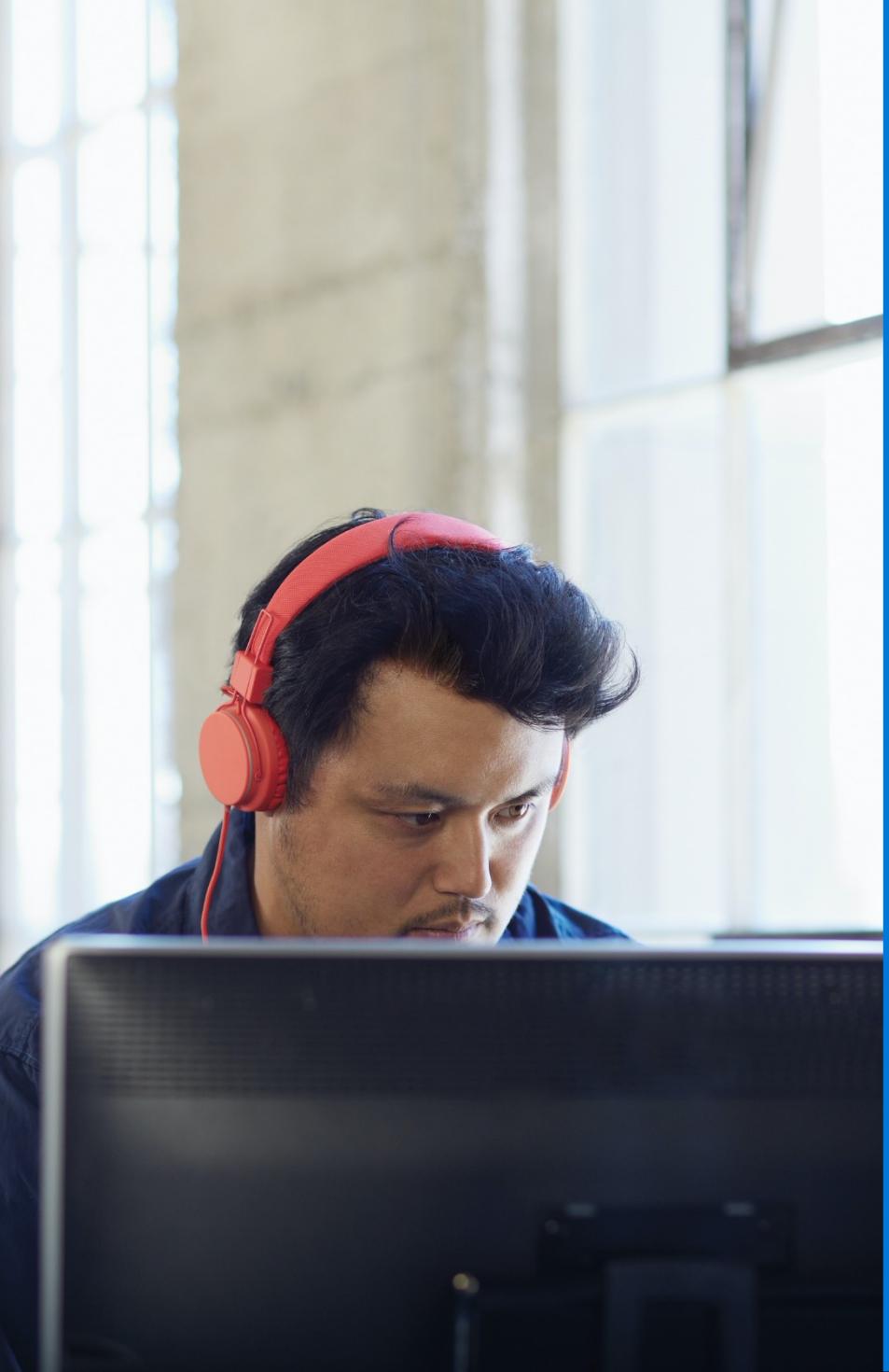
Simplify coding for new users with native Azure portal  
Select from popular editors, like VS, VS Code, CLI, Maven\*



# Gain **flexibility** and develop your way

## Hosting options

Consumption <i>Serverless</i>	AS Plan <i>Basic, Standard, Premium</i>	AS Environment <i>Network isolation</i>	Azure Stack <i>On-premises</i>	Runtime <i>Functions on your server</i>	IoT Edge <i>On devices</i>
 Only pay for what you use; charges apply per execution and per GB second	 Gain all the advantages of Functions along with Microsoft's financially-backed SLA and the always-on features of an App Service Plan	 Use a dedicated App Service cloud environment (ASE) that comes with network isolation for apps, greater scale, and secure connectivity to local vNets	 Bring the power of the entire Azure stack to your own data centers	 Run Functions on your local server; does not include the entire Azure stack	 Deploy custom Azure modules on IoT devices

A photograph of a man with dark hair and a beard, wearing red over-ear headphones. He is looking down at a computer monitor, which is visible in the foreground. The background shows a window with a view of a building.

# Sample scenarios for Functions

[Web](#) application backends

[Mobile](#) application backends

[IoT-connected](#) backends

[Conversational bot](#) processing

Real-time [file](#) processing

Real-time [stream](#) processing

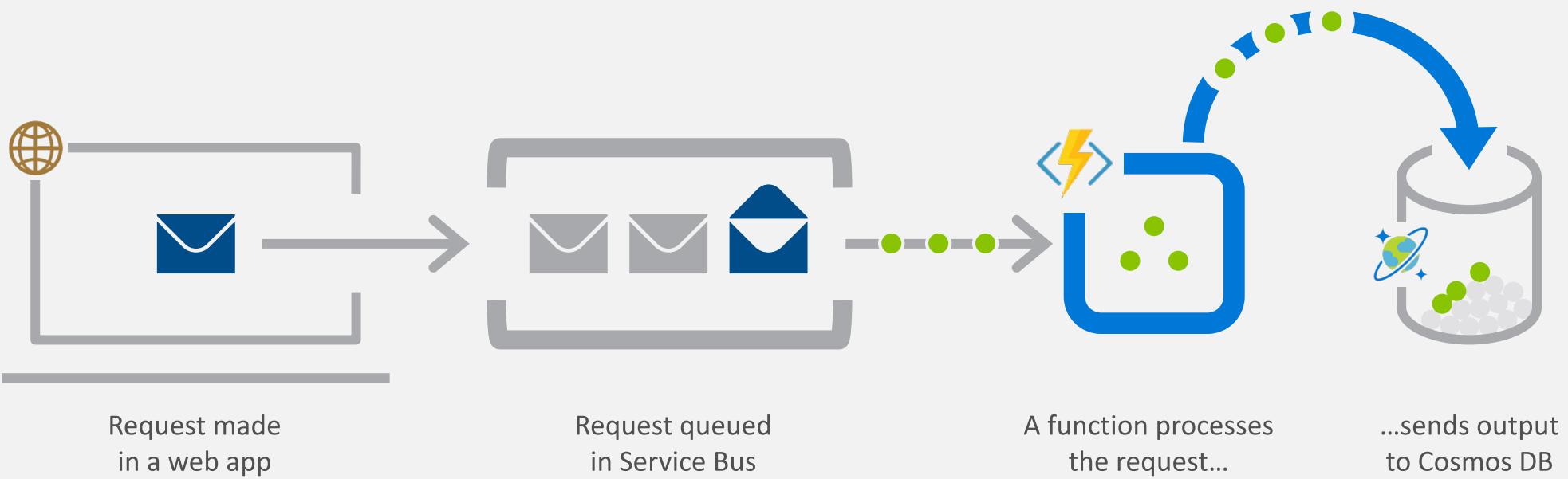
Automation of [scheduled tasks](#)

Extending [SaaS](#) Applications

## Scenario Example

### Retail

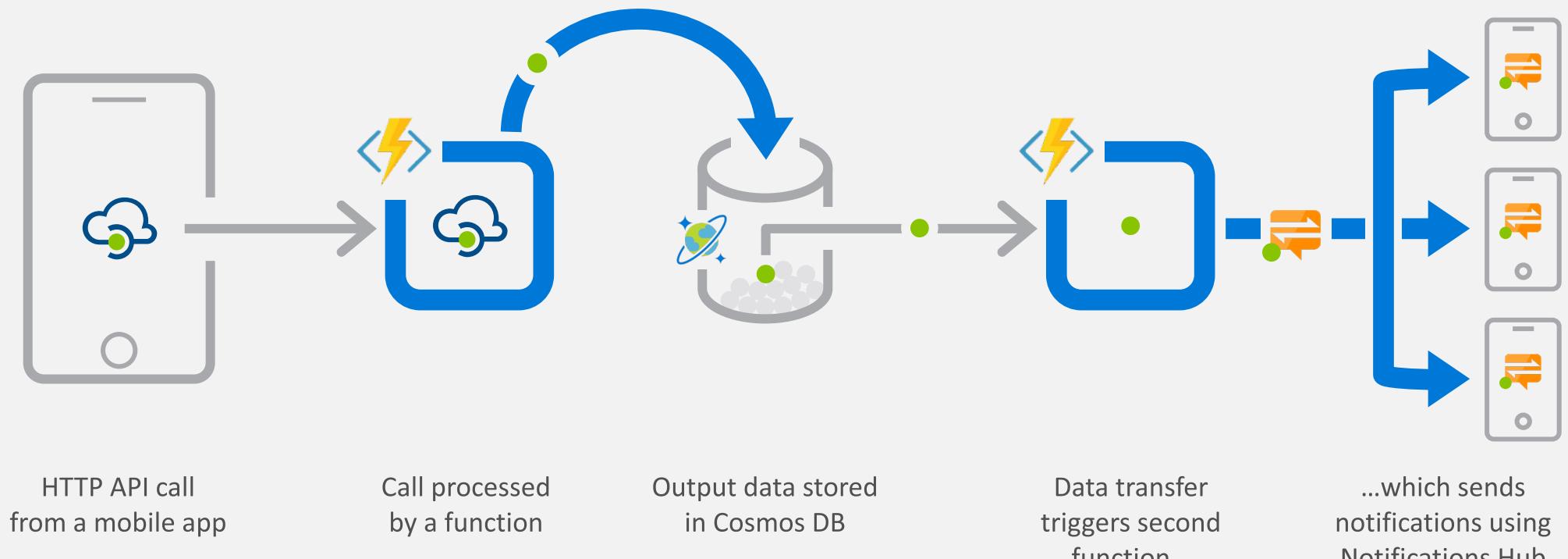
Online orders are picked up from a queue, processed and the resulting data is stored in a database.





Scenario Example  
— Financial Services —  
  
Colleagues use mobile banking to reimburse each other for lunch: the person who paid for lunch requests payment through his mobile app, triggering a notification on his colleagues' phones.

# Mobile application backends

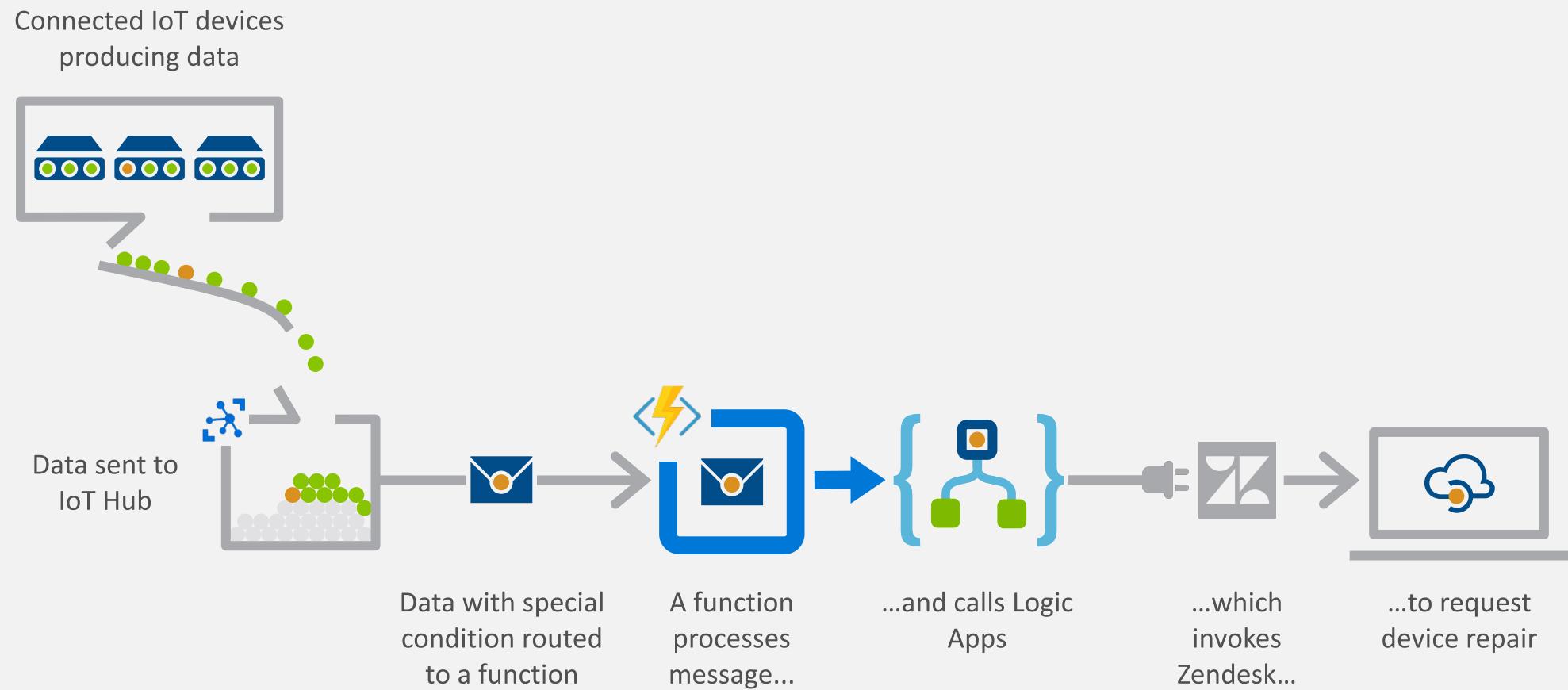




### Scenario Example — Manufacturing —

A manufacturing company uses IoT to monitor its machines. Functions detects anomalous data and triggers a message to Service department when repair is required.

# IoT-connected backends



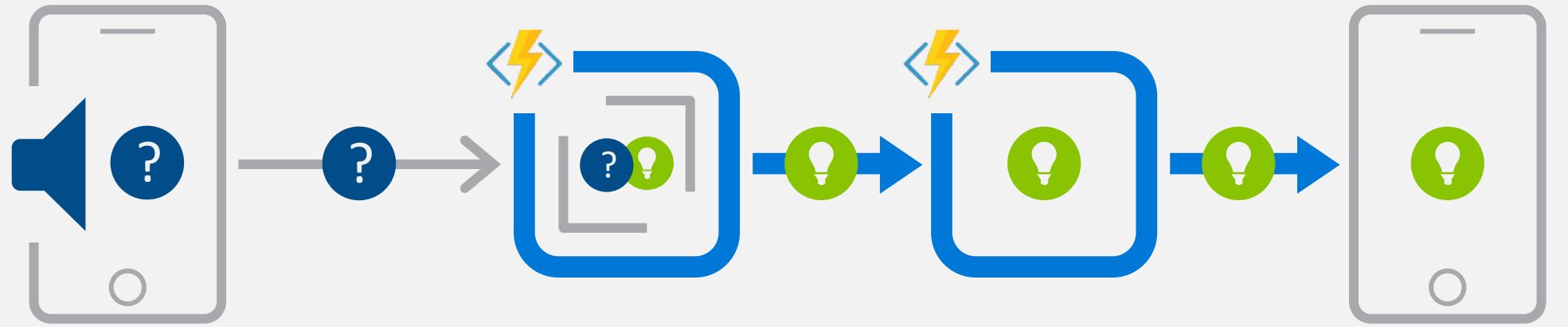
## Scenario Example

### Hospitality

Customer asks for available vacation accommodations on her smartphone. A serverless bot deciphers the request and returns vacation options.



# Conversational bot processing

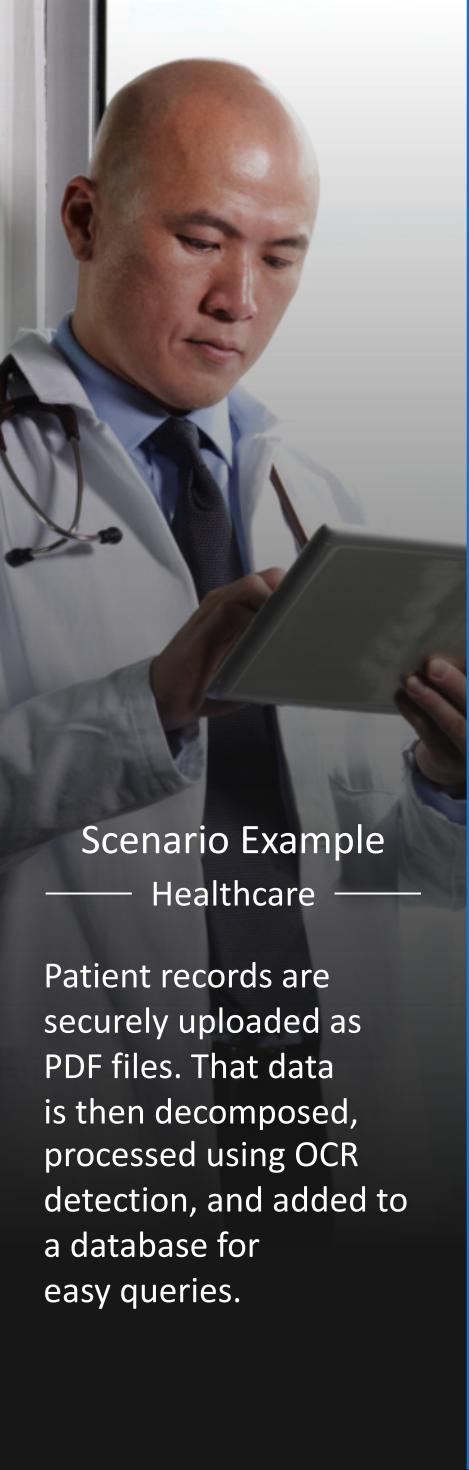


User request through conversational interface

Bot running in a function  
deciphers request using  
language understanding

Another function  
processes the  
request

...and sends response  
to original requester



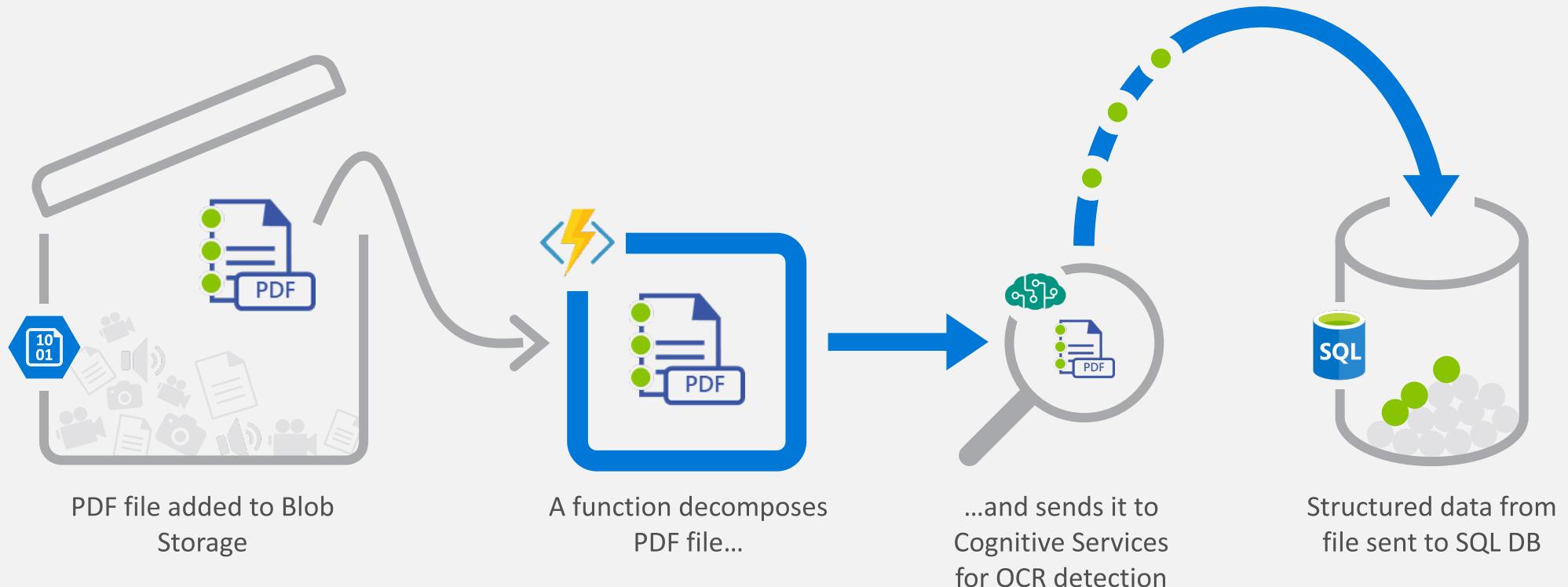
### Scenario Example

---

#### Healthcare

Patient records are securely uploaded as PDF files. That data is then decomposed, processed using OCR detection, and added to a database for easy queries.

# Real-time file processing



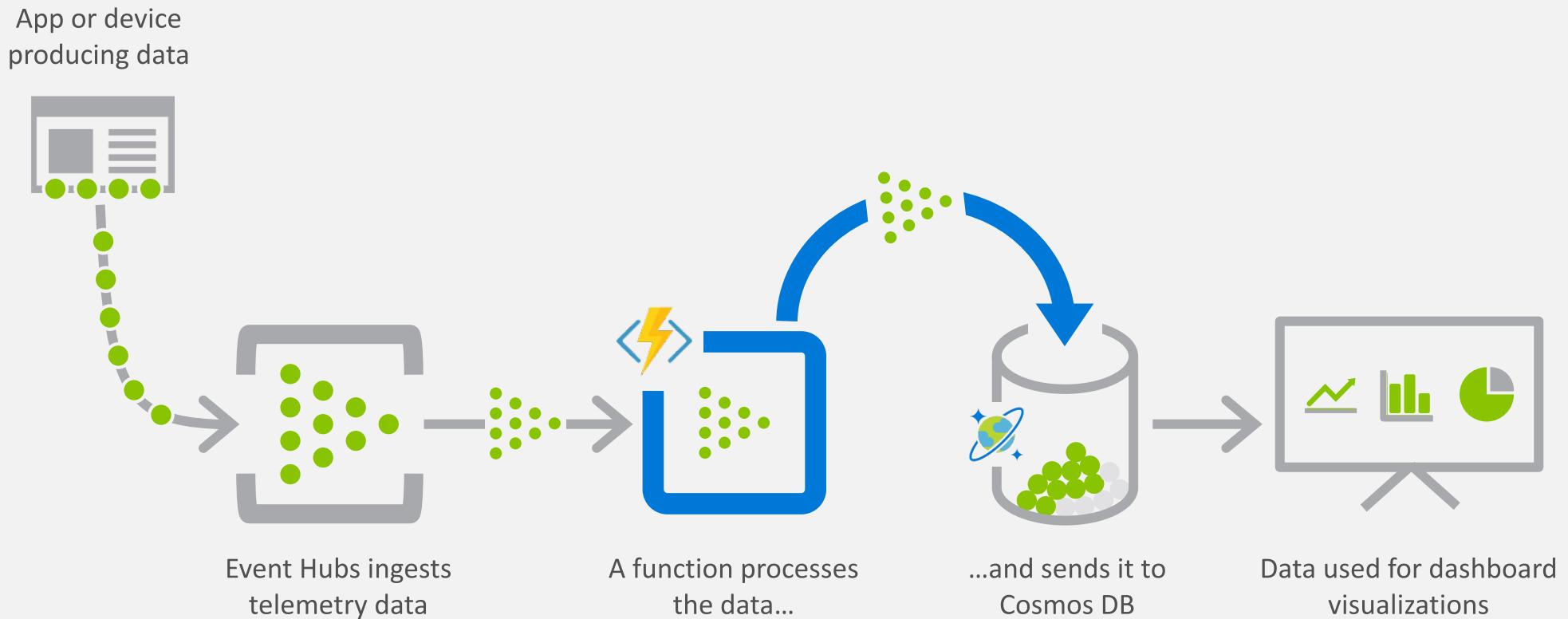
# Real-time stream processing



Scenario Example

ISV

Huge amounts of telemetry data is collected from a massive cloud app. That data is processed in near real-time and stored in a DB for use in an analytics dashboard.



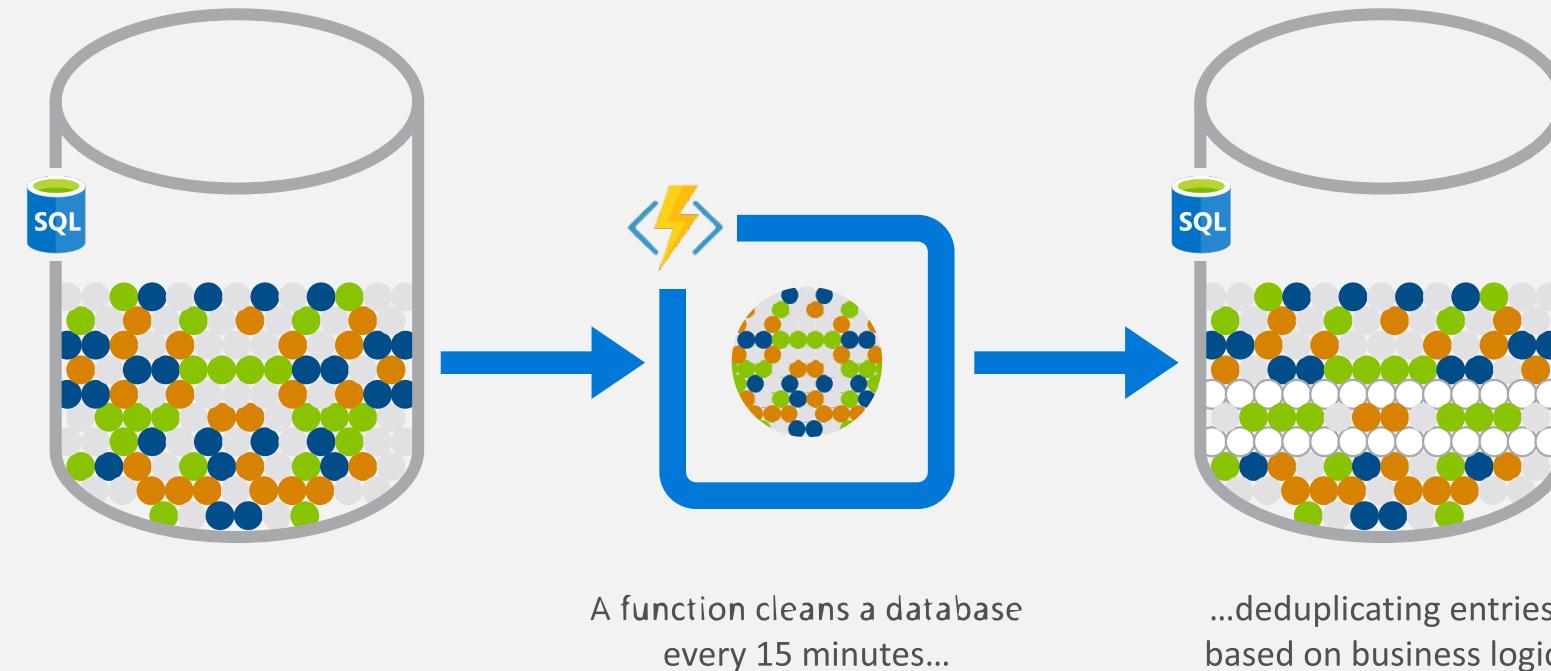
## Scenario Example

### — Financial Services —

A customer database is analyzed for duplicate entries every 15 minutes, to avoid multiple communications being sent out to same customers.



# Automation of scheduled tasks



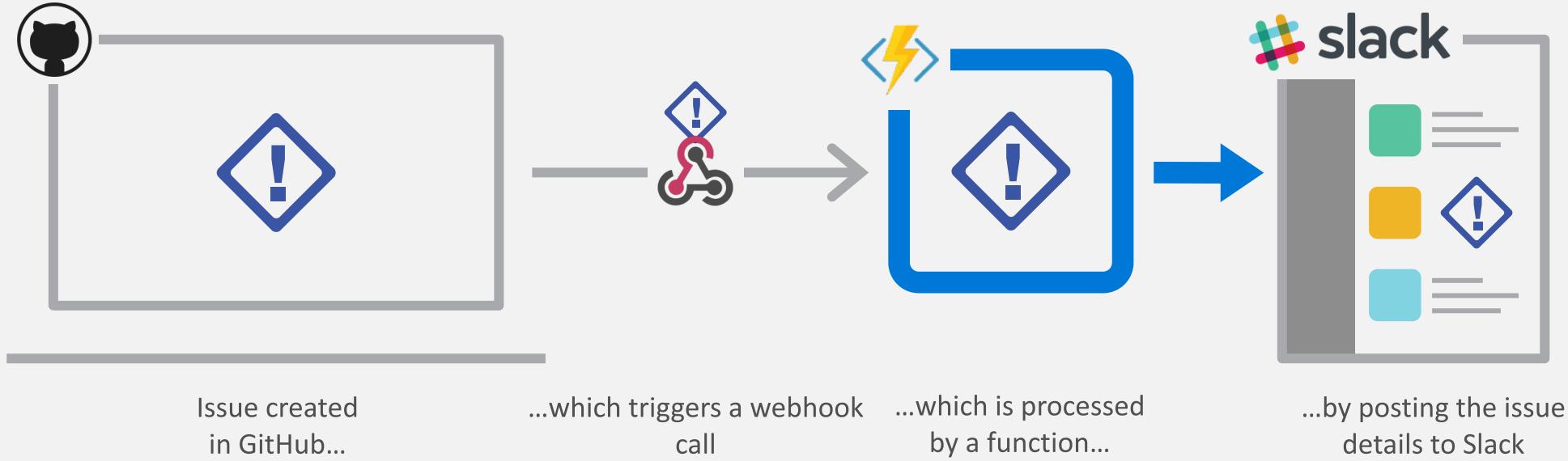
## Scenario Example

– Professional Services –

A SaaS solution provides extensibility through webhooks, which can be implemented through Functions, to automate certain workflows.



# Extending SaaS applications



# What makes Functions unique?

**Intuitive experience**  
Easy-to-use, familiar tools like the Azure portal and Visual Studio help you get running on serverless fast

**Flexible run options**  
With so many plan options, you can choose the level of access that's best for your company and customers

**Feature variety**  
A host of features, from bindings to code editors, gives you the tools needed to quickly create the best apps

**Durable Functions**  
With this extension Azure Functions help you simplify complex, stateful orchestration problems

**Consolidated tools**  
Thanks to so many built-in solutions, you can do more in Functions than AWS, which often requires external tools



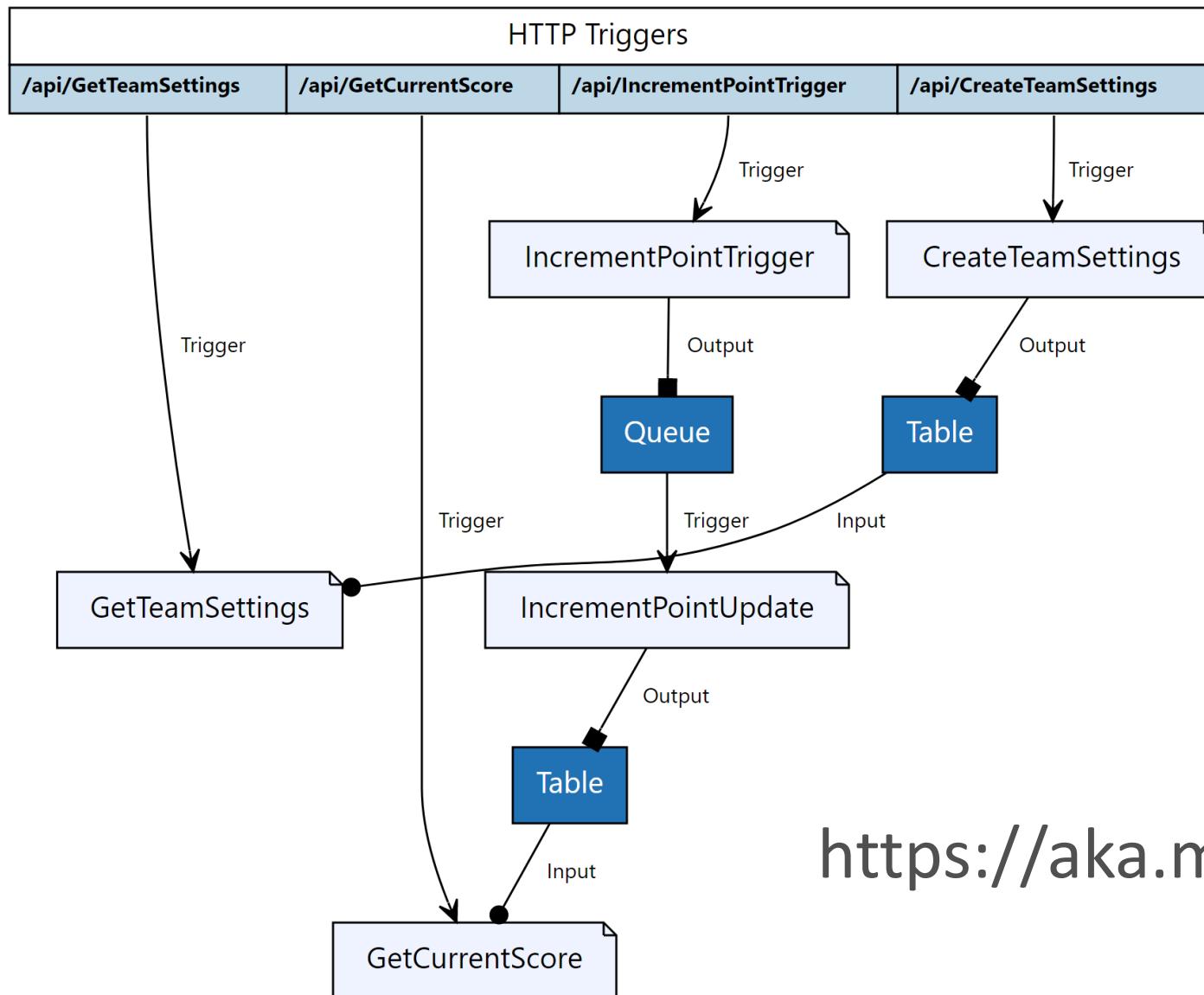
# Azure Functions is an open-source project

Functions runtime and all extensions are fully open source



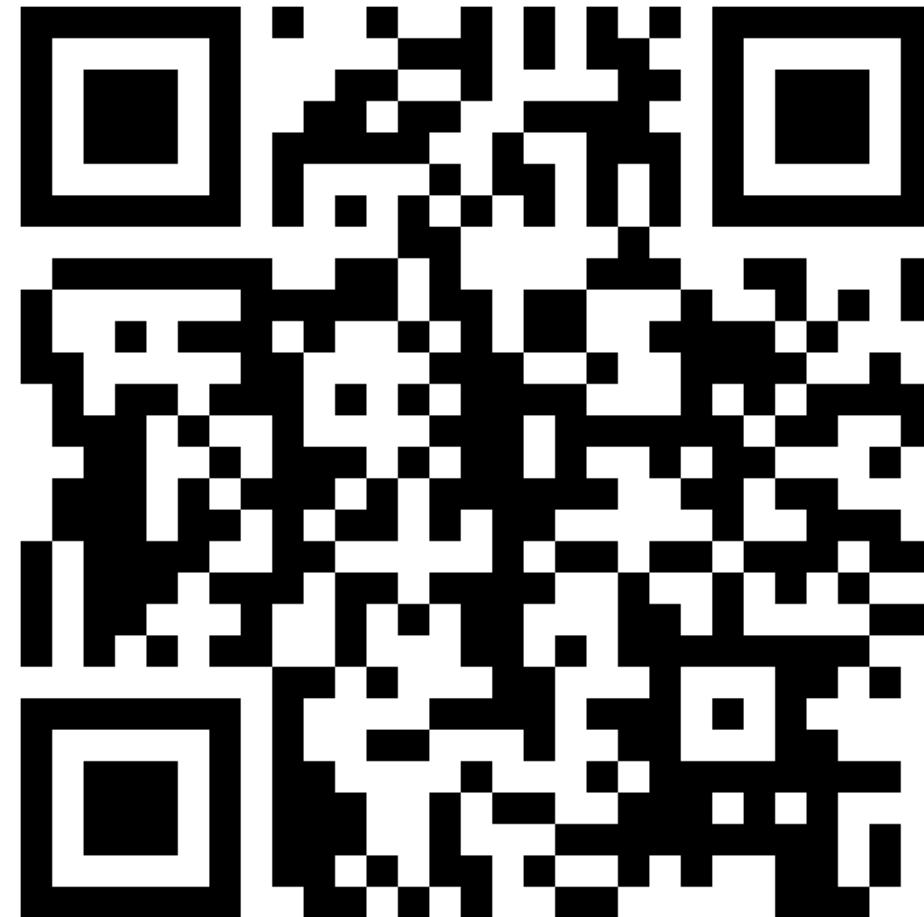
<https://github.com/Azure/Azure-Functions>

# Demo



<https://aka.ms/tug-of-war>

# Slides, Demos and more



<https://aka.ms/serverless-java>