Microsoft

Azure Functions 2.0
running serverless everywhere

Daniel Neumann
Azure Technology Solutions Professional – Microsoft
Daniel.Neumann@microsoft.com
@neumanndaniel

Session objectives

- Functions intro
- Functions 2.0
- Hosting models
- Tooling
- Durable Functions

Azure Serverless platform for event-driven apps

Development

- IDE support
- Integrated DevOps
- Local development
- Monitoring
- Visual debug history

Platform

API Management

Event Grid	Functions	Logic Apps
Manage all events that can trigger code or logic	Execute your code based on events you specify	Design workflows and orchestrate processes

Database | Storage | Automation | Intelligence | Security | IoT

Focus on code, not plumbing

No infrastructure management

Auto-scale based on your workload

No wasted resources, pay only for what you use

Azure Functions

Events

React to timers, HTTP, or events from your favorite Azure services, with more on the way

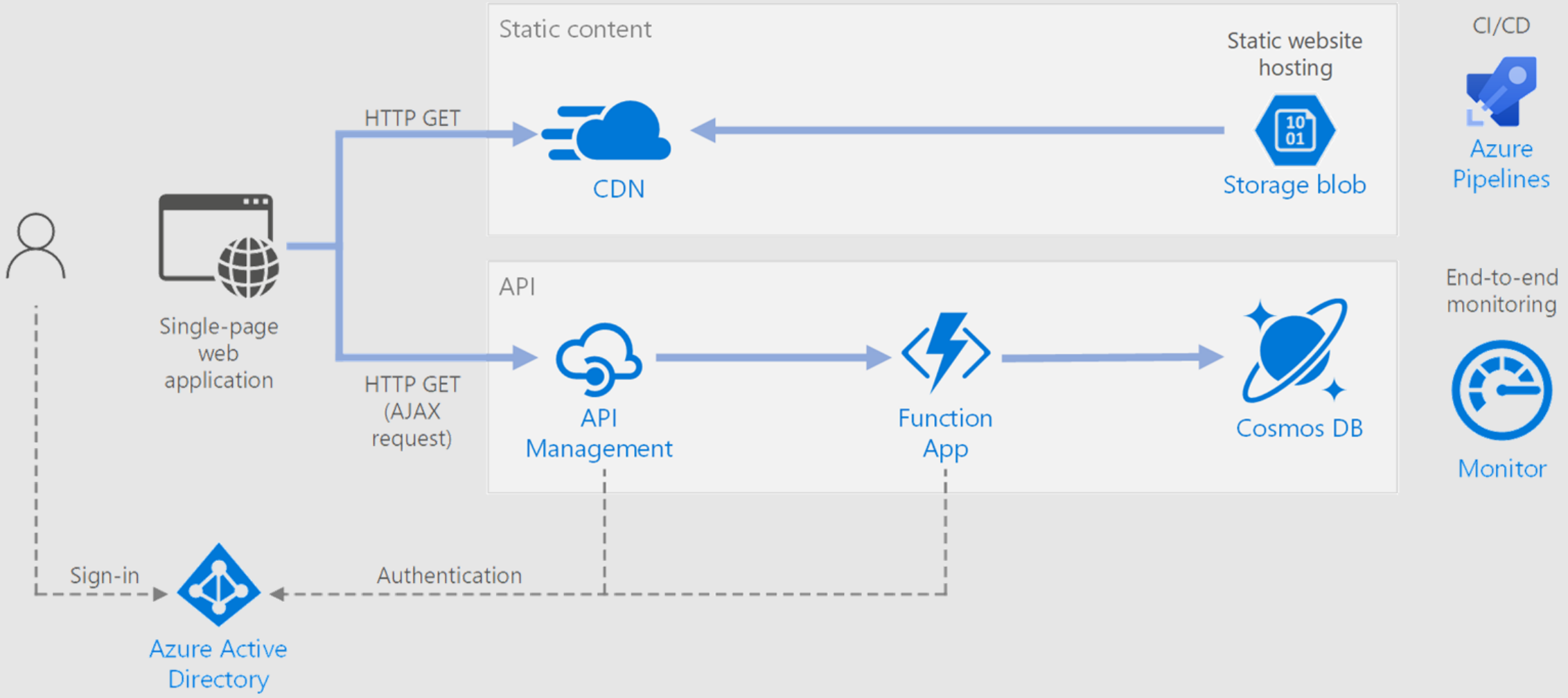
Code

Author functions in C#, F#, Node.JS, Java, and more

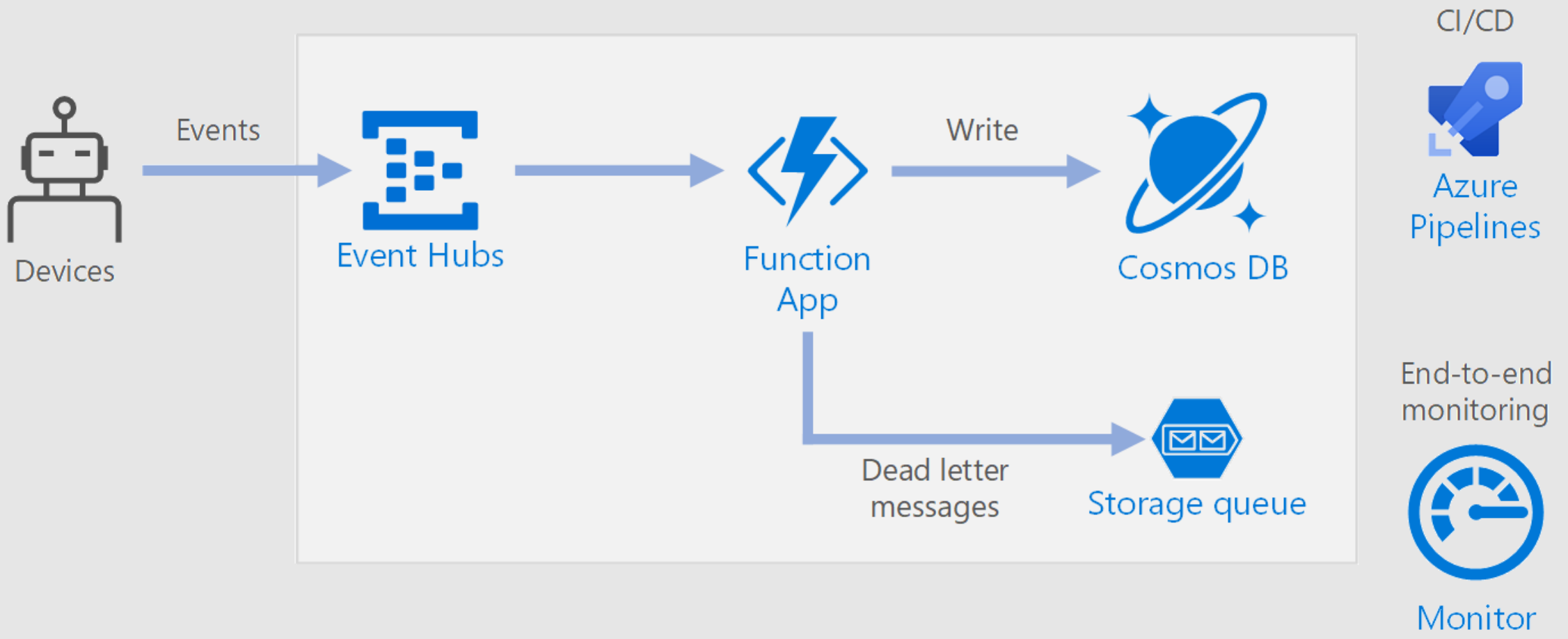
Outputs

Send results to an ever-growing collection of services

Azure Functions examples



Azure Functions examples



Functions 1.0 challenges

- Need for additional language support, e.g. Java, Python, PowerShell
- Only able to host on Windows
- No support for development on Mac and Linux
- Assembly probing and binding issues for .NET developers
- Performance issues on a range of scenarios / languages
- Lack of UX guidance to production success

Functions 2.0

- New Functions Quickstarts by Language
- Updated runtime built on .NET Core 2.1
- .NET Functions loading changes
- New extensibility model
  - Decoupled from language providers and bindings
- Run code from a package
- Tooling updates: CLI, Visual Studio & VS Code
- Durable Functions (GA)
- Consumption mode SLA

Functions runtime 1.0 and 2.0 key differences

	Functions 1.0	Functions 2.0
.NET Support	.NET Framework 4.7.1	.NET Core 2.1
Assembly isolation	No	Yes
Bindings versions	Runtime versions	User controlled
Language options	Limitations in languages and versions	Languages are external to the host
Node.js version	Node.js 6 only	Node.js 8 & 10 + future versions
Node.js native modules	Not supported	Supported
HTTP triggers	HTTP and specialized Webhooks	HTTP (supports Webhooks)
Language Runtime	Multiple languages per function app	Single language per function app
Functions Proxies	GA	GA
OpenAPI definition	Preview	Not yet available
Observability	Application Insights/WebJobs dashboard	Application Insights

Bindings and integrations

Functions 1.0

Microsoft.NET.Sdk.Functions (.NET Framework 4.6)

- HTTP
- Timer
- Storage
- Service Bus
- EventHubs
- Cosmos DB

Functions 2.0

Microsoft.NET.Sdk.Functions (.NET Standard 2.0)

- HTTP
- Timer

Microsoft.Azure.WebJobs.Extensions.Storage 3.0.0

Microsoft.Azure.WebJobs.Extensions.ServiceBus 3.0.0

Microsoft.Azure.Webjobs.Extensions.EventHubs 3.0.0

Microsoft.Azure.WebJobs.Extensions.CosmosDB 3.0.0

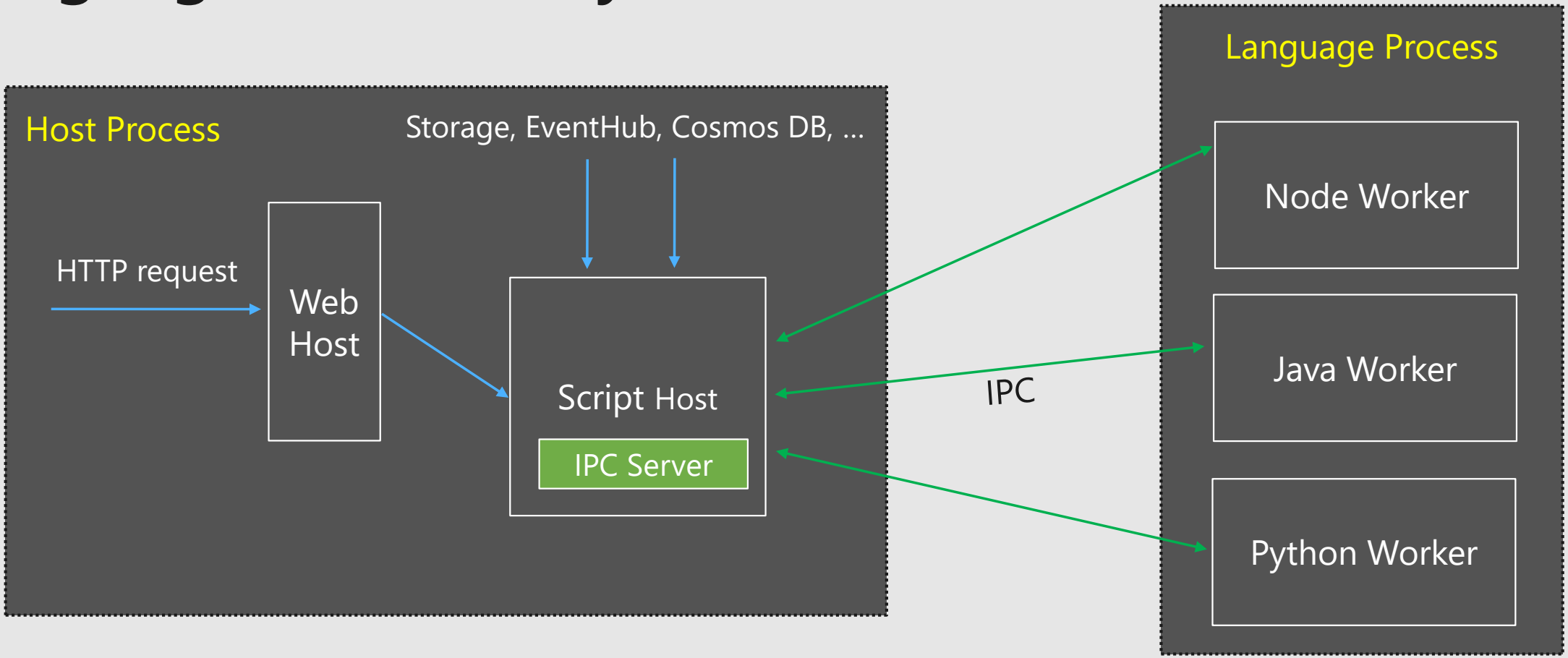
Microsoft.Azure.Webjobs.Extensions.EventGrid 2.0.0

Microsoft.Azure.Webjobs.Extensions.MicrosoftGraph 1.0.0-beta6

Microsoft.Azure.WebJobs.Extensions.DurableTask 1.4.0

Microsoft.Azure.Webjobs.Extensions.SignalRService 1.0.0-preview1-10002

Language Extensibility



- Worker and host broken into 2 separate processes
- Development of new language workers can happen independently
- Worker process crashes doesn't bring down the host

Deployment options: Run from package

Classic Deployment Issues:

1. Not atomic => inconsistent files
2. Files in use get locked
3. Multi-region inconsistencies
4. Difficult rollback

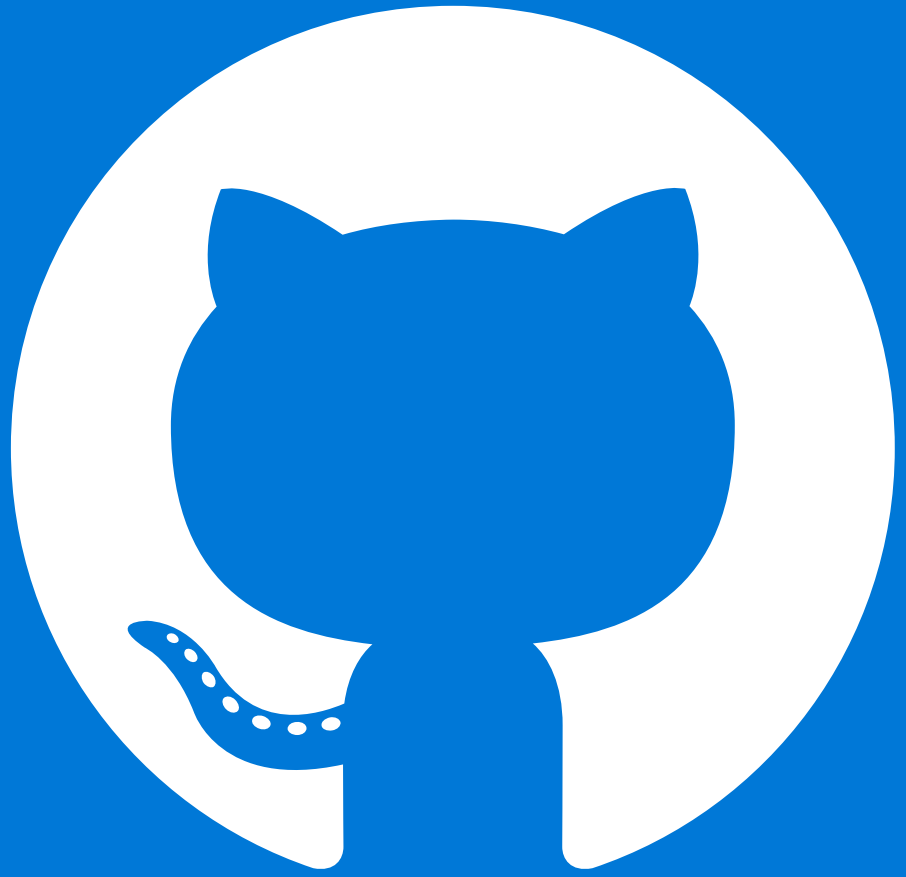
Solutions:

1. Externally hosted zip file
2. Zip file hosted within your app

Demo: Run from package

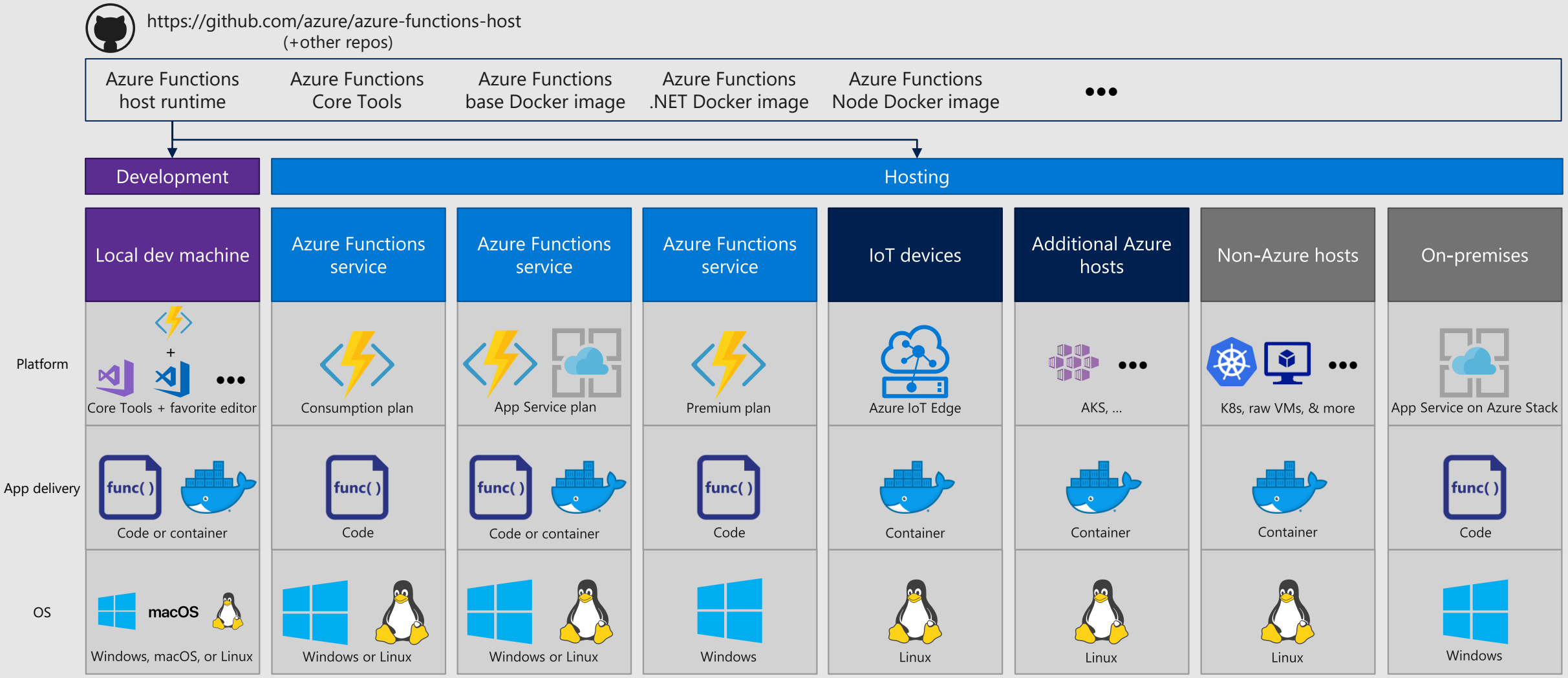
Azure Functions is an **open-source** project

Functions runtime and all extensions are fully open source



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

Functions everywhere



Demo: Python & containers

Try out the new Functions models

- Linux Consumption – *Preview*
  - https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-azure-function-azure-cli-linux
  - https://github.com/Azure/Azure-Functions/wiki/Azure-Functions-on-Linux-Preview
- Python support - *Preview*
  - https://docs.microsoft.com/en-us/azure/azure-functions/functions-create-first-function-python
- Functions on Kubernetes
  - https://medium.com/@asavaritayal/azure-functions-on-kubernetes-75486225dac0
  - https://github.com/Azure/azure-functions-core-tools#getting-started-on-kubernetes

Azure Functions Tooling Options

- Visual Studio
- VS Code
- CLI
- Portal
- Deployment Options

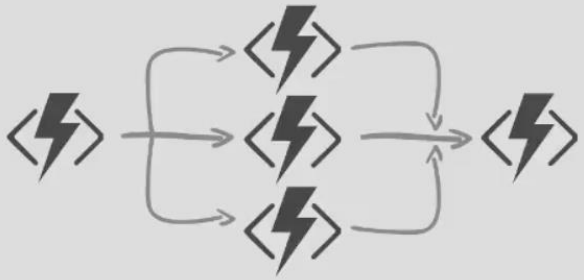
Azure Functions 2.0 - Recap

- Cross platform
- Assembly probing and binding issues addressed
- Decoupled bindings/extensions
- Language extensibility – out of process language workers
- Additional deployment options
- New tooling options
- New languages

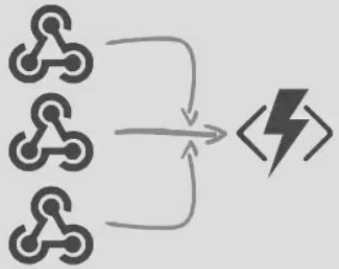
Durable Functions

What's still hard?

Manageable Sequencing
+ Error Handling / Compensation



Fanning-out & Fanning-in



External Events Correlation

Flexible Automated Long-running
Process Monitoring

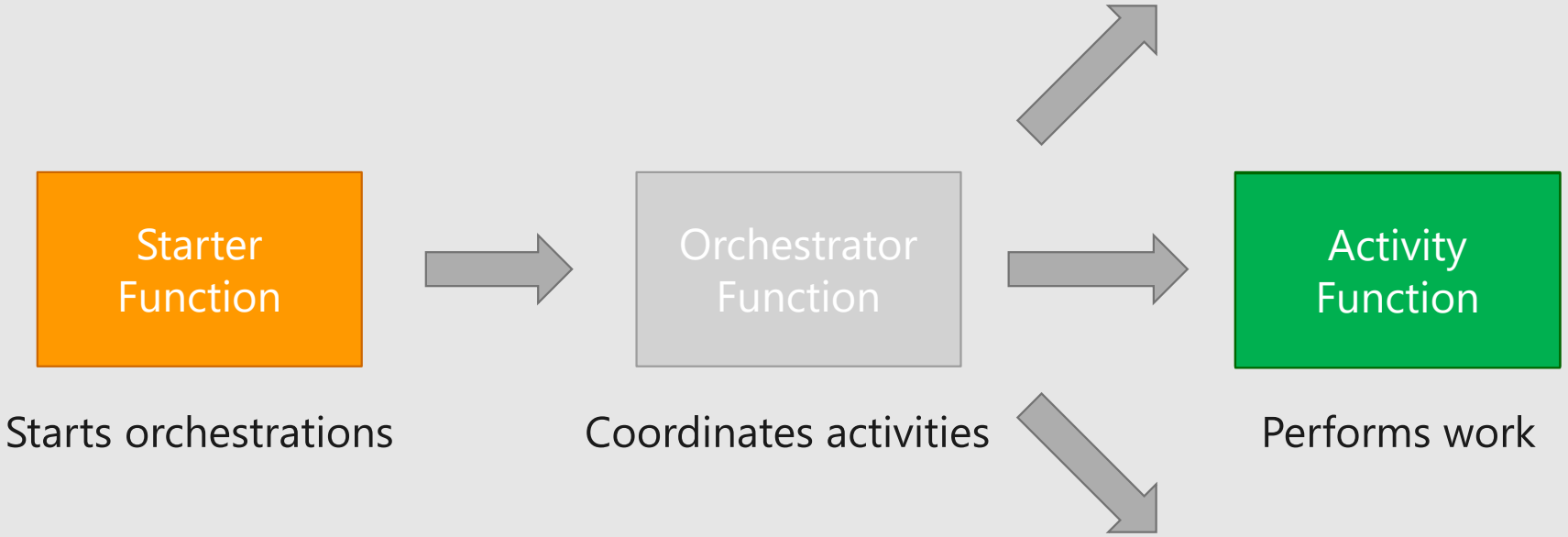
Http-based
Async Long-running APIs

Human Interaction

Durable Functions

- Write long-running orchestrations as a single function while maintaining local state.
- **Simplify complex transactions and coordination (chaining, etc.)** Easily call a Function from another Function, synchronously or asynchronously.
- All of the above using code-only. No JSON schemas. No graphical designer.
- GA (v2) - C# and JavaScript

Components



OrchestrationClient

OrchestrationContextTrigger

ActivityTrigger

What It Looks Like

Orchestrator Function

Activity Functions

```
// calls functions in sequence
public static async Task<object> Run(DurableOrchestrationContext ctx)
{
    try
    {
        var x = await ctx.CallActivityAsync("F1");
        var y = await ctx.CallActivityAsync("F2", x);
        return  await ctx.CallActivityAsync("F3", y);
    }
    catch (Exception)
    {
        // global error handling/compensation goes here
    }
}
```

Demo: Durable Functions

Microsoft

Thank you!