

Developing with Azure Functions

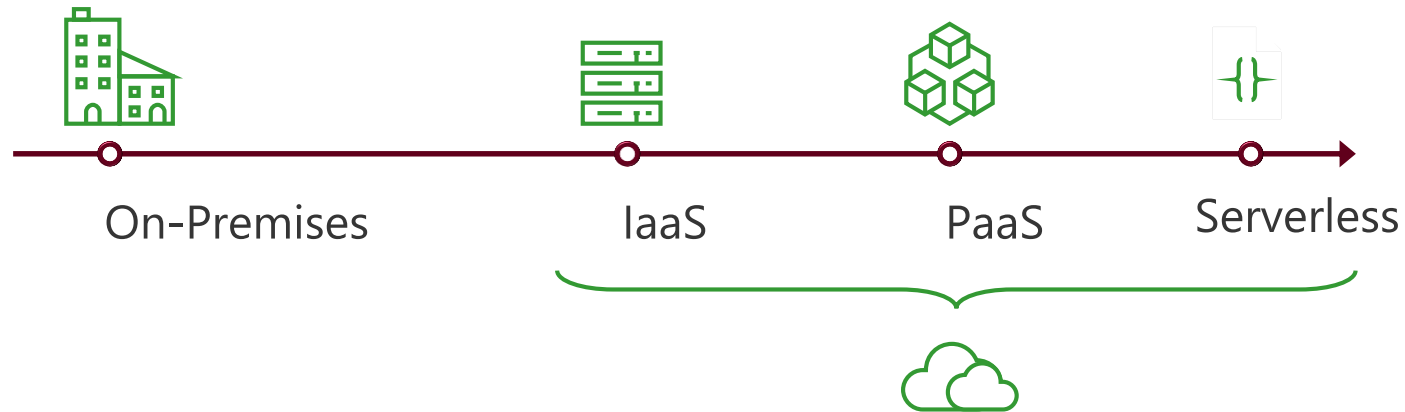


Agenda

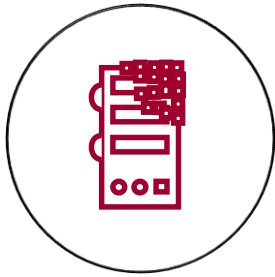
- Why Azure Functions?
- Creating Azure Functions



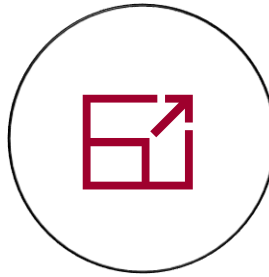
The evolution of application platforms



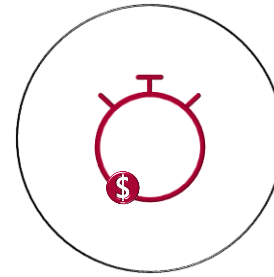
What is Serverless?



Abstraction
of servers



Event-driven/
instant scale



Micro-billing



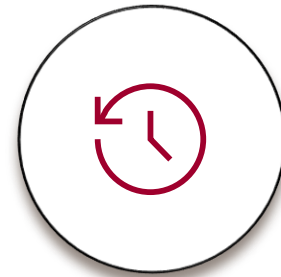
Benefits of Serverless



Manage apps,
not servers



Reduced
DevOps



Faster time
to market



Serverless application platform components

Development

 IDE support

 Integrated DevOps

 Local development

 Monitoring

 Visual Debug History

Platform

 Functions

- ✓ Developer productivity
- ✓ Triggers and Bindings
- ✓ Flexible deployment options

 Logic apps

- ✓ Visual designer
- ✓ 100+ connectors
- ✓ Functions orchestration

 Event Grid

- ✓ Manage all events in one place
- ✓ Near real-time delivery
- ✓ Broad coverage

Database



Storage



Security & Access Control



IoT



Analytics

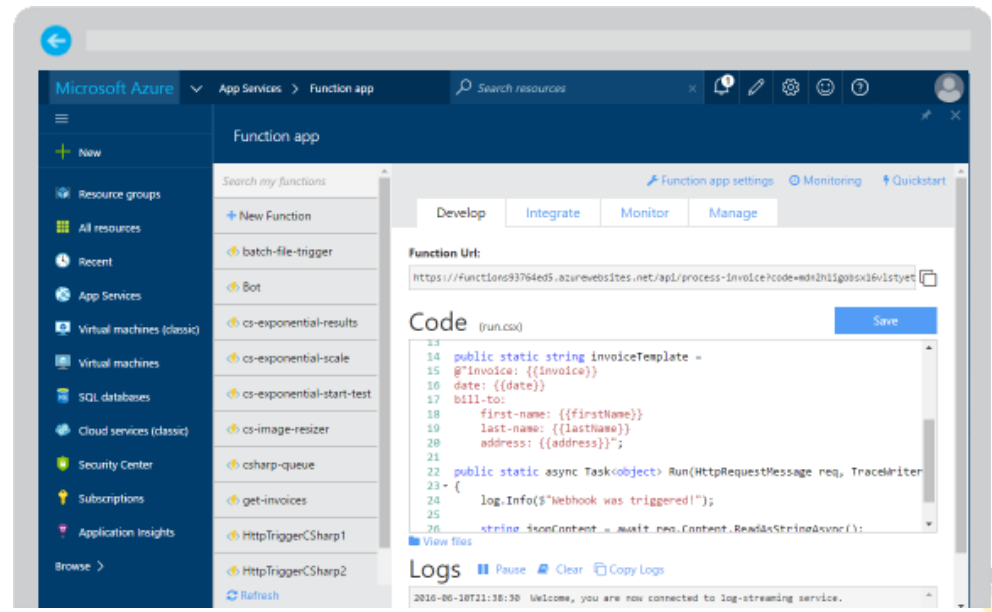


Intelligence



Azure Functions

Create a “serverless” event-driven experience that extends the existing Azure App Service platform by building “nanoservices” that can scale based on demand



Supported Languages and Tools

JavaScript

C#

Python

PHP

Bash

Batch

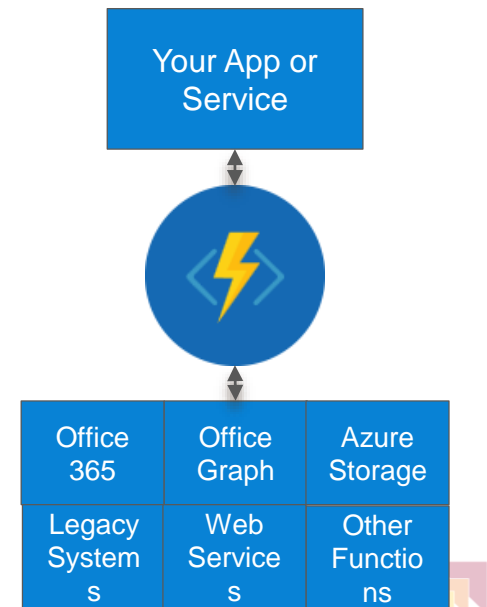
PowerShell

The screenshot displays the Azure Functions portal interface. On the left, a sidebar lists features: Runtime version: Latest (~0.4), Memory Size (with a slider), Features, Continuous Integration (Deploy your function code from GitHub), Authentication/Authorization (For functions that use the HTTP trigger, you can require calls to be authenticated.), CORS (Allow your HTTP-triggered functions to be called from within a web browser.), and API definition (Allow clients to more easily consume your HTTP-triggered functions.). The main content area has a blue header with the text 'The faster way to functions' and a sub-header 'Get started quickly with a premade function'. Below this, it says '1) Choose a scenario:' and shows three options: 'Timer' (selected with a blue border), 'Data processing', and 'Webhook + API'. To the right of these options are three buttons: 'Authentication', 'Configure CORS', and 'Configure API metadata'.



Common Scenarios

- Timer-based processing
- Azure service event processing
- SaaS event processing
- Serverless web application architectures
- Serverless mobile backends
- Real-time stream processing
- Real-time bot messaging



Function App Templates

Function App templates are categorized into general areas of Timer, Data Processing, and Webhook & API

- BlobTrigger
- EventHubTrigger
- Generic webhook
- GitHub webhook
- HTTPTrigger
- QueueTrigger
- ServiceBusQueueTrigger
- ServiceBusTopicTrigger
- TimerTrigger
- Blank & Experimental

Choose a template

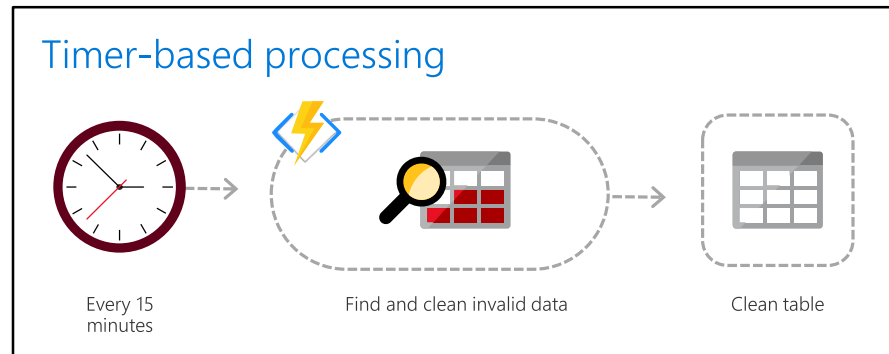
Language: Scenario:

<p>BlobTrigger - C#</p> <p>A C# function that will be run whenever a blob is added to a specified container</p>	<p>BlobTrigger - Node</p> <p>A Node.js function that will be run whenever a blob is added to a specified container</p>	<p>Empty - C#</p> <p>An empty C# function without triggers, inputs, or outputs</p>	<p>Empty - Node</p> <p>An empty Node.js function without triggers, inputs, or outputs</p>
<p>EventHubTrigger - Node</p> <p>A Node.js function that will be run whenever an event hub receives a new event</p>	<p>Generic Webhook - C#</p> <p>A C# function that will be run whenever it receives a webhook request</p>	<p>Generic Webhook - Node</p> <p>A Node.js function that will be run whenever it receives a webhook request</p>	<p>GitHub WebHook - C#</p> <p>A C# function that will be run whenever it receives a GitHub webhook request</p>



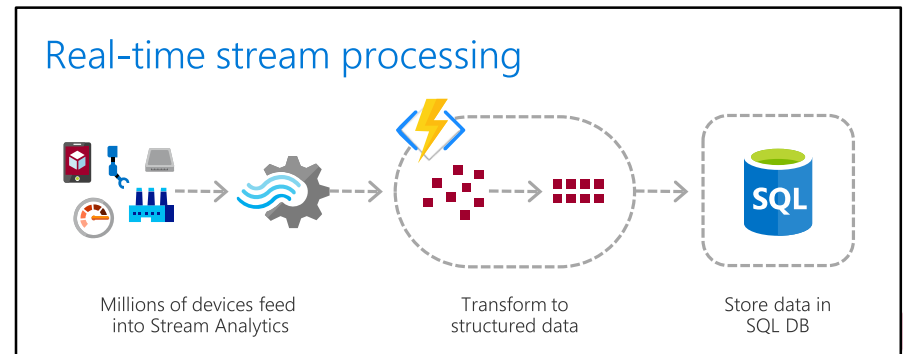
Timer Function Apps

- Run at explicitly specified intervals
 - Example interval: every day at 2:00 am
 - Intervals defined using CRON expressions
"0 */5 * * * *" - (every 5 minutes)
- What do you do with a timer app?
 - Send information to other systems
 - Write to logs
 - Perform redundant cleanup and data management
 - Check state of services



Data Processing Function Apps

- Data processing functions triggered by data event
 - an item being added to a queue
 - A file being uploaded to a container
- Data processing functions have in and out parameters
- Great for responding to CRUD events
- Great for performing CRUD events
- Great for moving content
- Access data across services



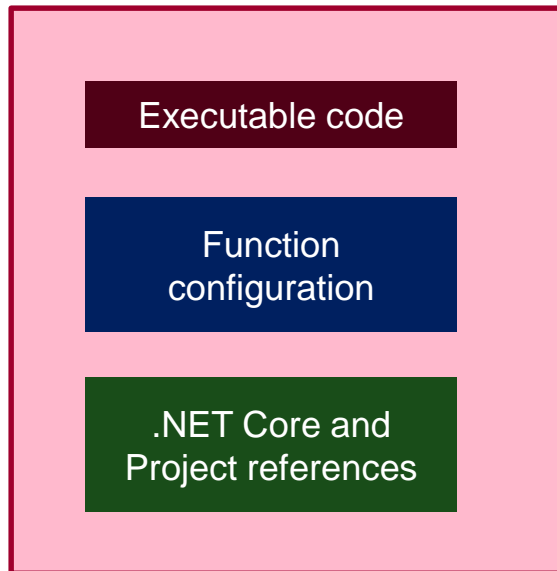
Webhook & API Function Apps

- Functions triggered by events supported in other services
 - GitHub events (e.g. repository created, wiki page updated, etc.)
 - SharePoint events (e.g. list item added, document deleted, etc.)
 - Office 365 events (e.g. mail message received)
- Takes in a request and sends back a response
 - Often mimic Web API and legacy web services flows
 - Typically need CORS settings managed
 - Best for exposing functionality to other apps and services
 - Great for building Logic Apps



Anatomy of a Function

- A function can be defined by...
 - A “Run” file containing function code
 - A “Function” file containing all service and trigger bindings and parameters
 - A “Project” file containing project assembly and NuGet package references
 - App Service settings, such as connection strings and API keys



Azure Function Types

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			✓

