andrena OBJECTS

Experts in agile software engineering

Serverless Computing - Azure Functions im Praxiseinsatz und wo die Reise hingeht

ObjektForum Mannheim

10.02.2020
Mannheim

Özhan Sahillioglu

oezhan.sahillioglu@andrena.de

Maxim Becker

maxim.becker@andrena.de

Inhalt

1. Azure Functions
   - Allgemein
   - Hosting
   - Trigger und Bindungen, Singletons
   - Skalierung
   - Dependency Injection
   - Testbarkeit
   - Continuous Integration und Delivery
2. Azure Functions im Praxiseinsatz
   - Ausgangslage
   - Migrationspfad
   - Probleme und Lösungen
3. Code-Beispiel
4. Wo geht die Reise hin

Azure Functions – Allgemein

- Serverless Computing
  - Die Großen der Branche: Azure, AWS und Google Cloud
- Azure Functions – Serverless Computing Framework von Microsoft
  - Automatische und flexible Skalierung
  - Programmiermodell auf Basis von Triggern und Bindungen
  - Umfassende Tools wie DevOps zum Erstellen, Bereitstellen und Überwachen
  - Unterstützung großer Anzahl an Programmiersprachen und Hostingoptionen
  - Windows oder Linux als Hostingplattfom
- Aktuelle Version von Azure Functions SDK v3

Azure Functions – Hosting

- Verbrauchstarif (Consumption Plan)
  - Pay-as-you-go
  - Bezahlt werden Ausführungszeit (Gbyte-Sekunden) und Anzahl Ausführungen
  - Maximale Ausführungsdauer einer Function 10 Minuten (Default 5 Minuten)
  - 400.000 GB-s und 1000.000 Ausführungen pro Abonnement im Monat kostenlos

Azure Functions – Hosting

- Premium-Tarif (Elastic Premium Plan)
  - Bezahlt werden CPU- und Speicher-Zeit
  - „Vorgewärmte“ Instanzen zum schnelleren Start
  - Unbegrenzte Ausführungsdauer (Default 30 Minuten)
  - Höhere Planungssicherheit bezüglich der Kosten

Azure Functions – Hosting

- Dedizierter App-Service-Plan
  - Bezahlt werden die Ressourcen abhängig von der gewählten Größe
  - Unbegrenzte Ausführungsdauer (Default 30 Minuten)
  - Keine automatische Skalierung (nur anhand der Skalierungsregeln)
  - Keine gesonderte Abrechnung für Functions

Azure Functions – Hosting

- Weitere Möglichkeiten
  - Docker Container (Kubernetes)
  - On Premise
  - Azure Stack
  - Azure IoT Edge

Azure Functions – Trigger und Bindungen, Singletons

Trigger

- Trigger starten die Ausführung einer Function
- Mehr als 10 verschiedener Trigger
  - Http
  - ServiceBus
  - Timer
  - EventHub
  - Storage Queues
  - Blob
  - …

Azure Functions – Trigger und Bindungen, Singletons

Bindungen

- Bindungen können Eingabe oder Ausgabe präsentieren
- Mehr als 20 verschiedener Bindungen
  - Http
  - ServiceBus
  - EventHub
  - Storage Queues
  - Storage Table
  - CosmosDb
  - Blob
  - NotificationHub
  - …
- Eigene Bindungen (Custom Bindings)

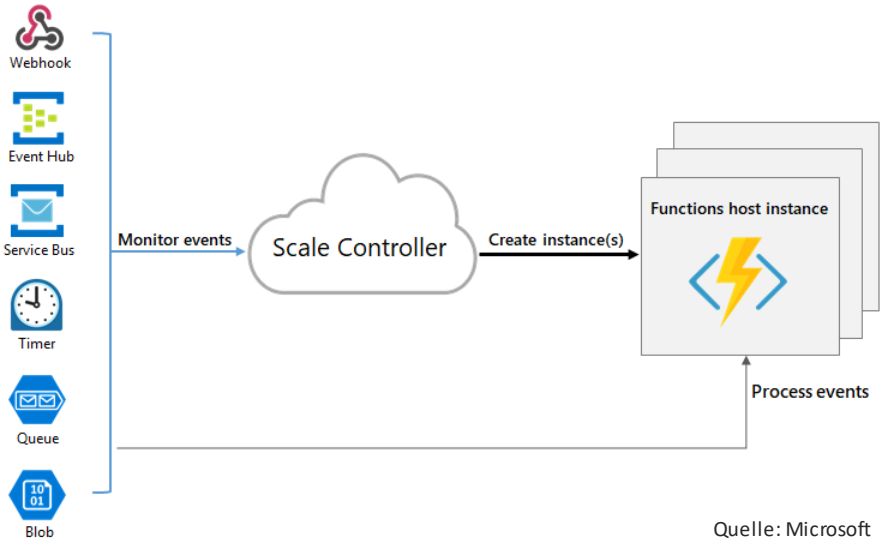
Azure Functions – Trigger und Bindungen, Singletons

Singleton

- Eine Function kann als Singleton deklariert werden
- Singletons steuern die Ausführung der Functions auf mehreren Instanzen
- Zur gleichen Zeit führt nur eine Instanz diese Function einmal aus
- Z.B. zeitgesteuerte Functions sind immer Singletons

Azure Functions – Skalierung im Verbrauchs- oder Premium-Tarif

- Für HTTP-Trigger neue Instanzen werden höchstens ein mal pro Sekunde bereitgestellt
- Für sonstige Trigger werden neue Instanzen höchstens alle 30 Sekunden bereitgestellt
- Scale Controller benötigt Informationen über alle Trigger, um Events zu überwachen



Quelle: Microsoft

Azure Functions – Dependency Injection

- Wird unterstützt in Form von ServiceCollection
- Lebensdauer der Dienste
  - Vorübergehend (Transient) – wird bei jeder Anforderung neu erstellt
  - Bereichsbezogen (Scoped) – hat die gleiche Lebensdauer wie die Function selbst
  - Singleton – hat die gleiche Lebensdauer wie der Function-Host
- Logging, Configuration und Application Insights werden automatisch hinzugefügt

Azure Functions – Testbarkeit

- Unit-Tests dank Dependency Injection sehr einfach
- Bei der Wahl der Eingangs- und Ausgangsparameter darauf achten, dass möglichst abstrakte Klassen oder Schnittstellen gewählt werden
- Es gelten die selben Regeln wie bei „normalen“ Testklassen

```
[FunctionName(nameof(Run))]
public async Task Run([BlobTrigger("container/blob.txt")]
CloudBlockBlob blob)
{
...
}
```

```
[FunctionName(nameof(Run))]
public async Task Run([BlobTrigger("container/blob.txt")]
ICloudBlob blob)
{
...
}
```

Azure Functions – Continuous Integration und Delivery

- Bereitstellung der Azure Functions als Package oder Docker Container
- Erstellen der Azure Functions mit Azure Resource Manager Template (Infrastructure-as-Code)
- Sehr gute Integration in Azure DevOps (ehemals Visual Studio Team Services)
- Alternativ Jenkins als Build-Plattform

Azure Functions im Praxiseinsatz – Ausgangslage

- 3 relativ große Web API Projekte (.Net + Angular)
- Ca. 30 Azure WebJobs mit insgesamt ca. 50 Functions (alles .Net 4.7)
- Teilweise lang laufende Function Ausführungen (> 10 Minuten)
- Hosting der Web APIs auf Linux Service Plans vom PO gewünscht, weil preiswerter als Windows Pendants
- Migration auf .Net Core für alle Komponenten angestrebt
- Entity Framework 6.x im Einsatz, deswegen nur Migration auf .Net Core 3.x möglich

Azure Functions im Praxiseinsatz – Migrationspfad

- Migration der bestehenden WebJobs zu Azure Functions aufgrund eines relativ geringen Aufwands gewählt
- Hosting der Azure Functions auf Elastic Premium Plans (Planungssicherheit und längere maximale Ausfürungsdauer)
- Schrittweise Migration einzelner WebJobs möglich
- Parallele Migration der Web-Api Projekte, um gemeinsame Framework-Basis beizubehalten
- Zum aktuellen Zeitpunkt ca. 75% der WebJobs bereits migriert und laufen produktiv

Azure Functions im Praxiseinsatz – Probleme und Lösungen

- Verwaltung der Datenbank-Migrationen erschwert, die Assembly mit Migrationen für .Net Standard 2.1 kann weder in Visual Studio noch in Rider verwendet werden.
  - Multi-Targeting der Migrations-Assembly (.Net 4.7 + .Net Standard 2.1), .Net 4.7 nur lokal zur Verwaltung der Migrationen in Visual Studio
- Assembly „System.Data.SqlClient“ wird nicht abhängig von Ausführungsplattform geladen.
  - Es existiert bereits ein Issue unter https://github.com/Azure/azure-functions-host/issues/3903
  - Als Workaround das Laden der Assemblies für die App-Domäne mit AssemblyResolve beeinflusst

Azure Functions im Praxiseinsatz – Probleme und Lösungen

- Instabile Zip-Deployments und Tools im Azure Portal auf Linux Hosts.
  - Grund ist anscheinend die aufwendigere Deployment-Methode unter Linux und die Einbindung der Packages in Containern
  - Tritt desto öfter, je mehr Function Apps einen Service-Plan teilen
  - Als Workaround die Anzahl der Apps pro Service-Plan einschränken
  - Alternativ Function App als Docker Container bereitstellen
  - Oder Windows Hosts benutzen

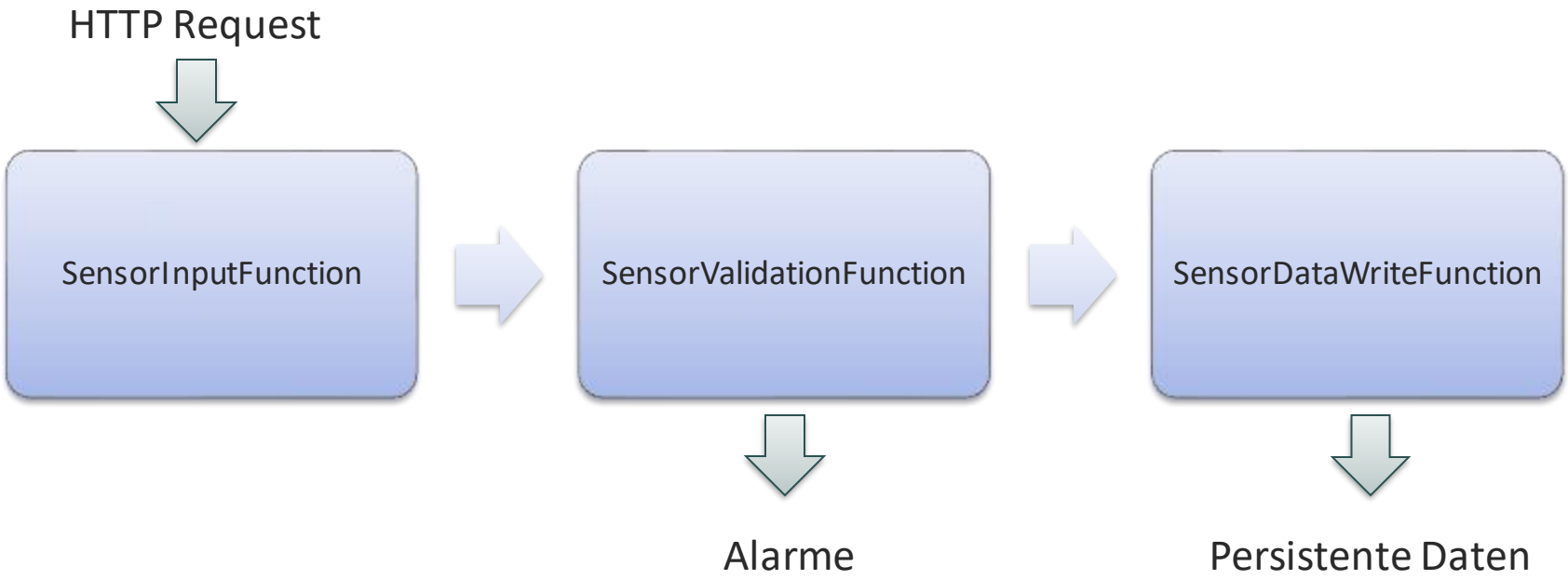
Code-Beispiel – Aufgabe

- Stadt Mannheim installiert mehrere (Hundert) Sensorboxen zum Überwachen der Luftqualität in der Stadt
- Jede Box besteht aus 4 Sensoren zum Messen der Umgebungstemperatur, Luftfeuchtigkeit, des Luftdrucks und Luftqualität ca. alle 3 Sekunden und kann die Daten per HTTP senden. Zum Schonen der Batterie erfolgt das Senden der gesammelten Daten alle paar Minuten.
- Unser Unternehmen soll eine API bereit stellen, die diese Daten entgegen nimmt, speichert und auf Anomalien überwacht.
- Die Anwendung soll ausgefallene Sensoren erkennen können

Code-Beispiel – Lösung

- Alle Komponenten unserer API sind mit Azure Functions umgesetzt
- Die Messdaten der Sensorboxen nimmt **SensorInputFunction** entgegen, bildet aus den Eingangsdaten Statistikwerte (Mittelwert, Minimum, Maximum, Standardabweichung) und sendet diese zur Weiterverarbeitung
- Aggregierte Daten werden von **SensorValidationFunction** auf Anomalien geprüft und zum Speichern weiter gesendet
- **SensorValidationFunction** prüft außerdem, ob Sensoren kontinuierlich Daten senden
- Validierte Daten werden von **SensorDataWriteFunction** persistiert

Code-Beispiel – Übersicht



Code-Beispiel – Verwendete Technologien

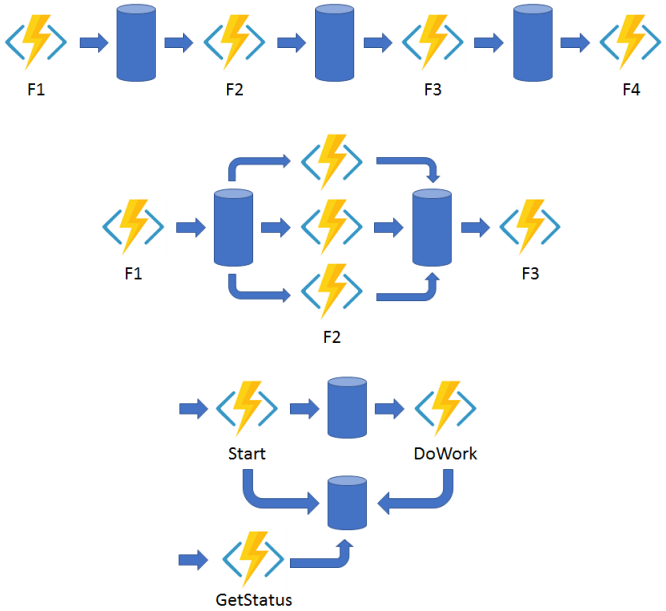
- Dependency Injection (FunctionsStartup)
- Continuous Integration und -Delivery (build.yml, resourceGroup.iac.json)
- Http Trigger (SensorInputFunction)
- Queue Binding (SensorInputFunction, SensorValidationFunction)
- Queue Trigger (SensorValidationFunction, SensorDataWriteFunction)
- Custom Binding (SensorInputFunction, [Principal])

Wo geht die Reise hin

- Azure Functions als Alternative oder Ergänzung zu Web APIs
  - Attraktiv durch schnellere Skalierung
  - Noch keine Unterstützung von Middlewares analog zu ASPNet Core (z.B. Authentifizierung über Middleware)

Wo geht die Reise hin

- Durable Functions
  - Eine Erweiterung der Azure Functions
  - Implementierung der zustandsbehafteten Workflows
  - Verkettung der Functions, Fan-in und Fan-out, Asynchrone HTTP APIs usw.



Quelle: Microsoft

Wo geht die Reise hin

- Für manche Aufgabenstellungen werden Azure Functions bevorzugt
  - Verarbeitung von IoT- oder anderen Massendaten
  - Wenn höhere Reaktionsgeschwindigkeit auf sich verändernde Anfragenanzahl erwartet wird

Quelle: heise online