Workshop Anomaly Detection

met Azure Stream Analytics

Met behulp van deze workshop creëer je je eigen stream analytics jobs en detecteer je waarden die afwijken van de verwachting. Het doel is om deze afwijkingen in de tijd te aggregeren om een ‘alarm’/anomaly te genereren om het moment dat een probleem wat langer aanhoudt.

Voor deze hands-on sessie maak je gebruik van Azure resources in de resource group ‘**rg-sig-stream-analytics’** en creëer je zelf resources die je aanmaakt in je eigen resource group.

# Voorbereiding:

Maak een eigen resource group aan en voeg daarin de volgende resources toe:

* Een storage account
* Een Stream Analytics Job

Start bij het toevoegen van de input en ouput bindings van de stream analytics job. Dit kan door te navigeren naar Inputs/Ouputs of binnen Query de input/output toe te voegen doormiddel van het plusje.

A screenshot of a computer screen

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

## Input 1:

Voeg een event hub toe:

* Kies voor event hub namespace: **evhns-sig-stream-analytics**
* Creëer een nieuwe consumer group (met een referentie van je naam)
* Maak gebruik van een connectionstring voor de authenticatie

Deze event hub streamt iedere 4 seconden van 4 verschillende devices telemetry (dus effectief krijg je 1 telemetry bericht per seconde binnen). Dit wordt gebruikt als input stream voor de job waar we de afwijkingen op willen detecteren.

## Input 2:

Voeg een reference input toe van SQL Database:

* Kies voor de **metadata** database op **server sqlsrv-sig-stream-analytics**
* Authentificeer met de SQL credentials:
  + Username: asaUser
  + Password: AsaIsLeuk1!
* Voeg als snapshot query een selectie van de gehele ‘Devices’ tabel

De tabel bevat metadata van de devices. Hiermee gaan wij onze telemetry uitbreiden. De tabel ziet er als volgt uit:

A screenshot of a computer

Description automatically generated

Outputs:

Je gaat twee outputs nodig hebben, beide verwijzen naar hetzelfde storage account met een verschillende container.

## Output 1:

Voeg een blob storage als output toe:

* Kies voor je eigen storage account
* Maak een container aan met de naam ‘Amsterdam’
* Maak gebruik van een connection string als authenticatie

## Output 2:

Voeg een blob storage als output toe:

* Kies voor je eigen storage account
* Maak een container aan met de naam ‘Utrecht’
* Maak gebruik van een connection string als authenticatie

# Aan de slag

We gaan starten om de query binnen de stream analytics job langzaam op te bouwen.

De code kan op twee manieren getest worden.

* Door de job te starten en te stoppen onder ‘Overview’ in de portal. Let op: Het starten en stoppen van de job kan een paar minuten duren. Tevens kan de code niet aangepast worden als de job aanstaat.
* Binnen de query kan ook getest worden. Dit kan onder ‘Query’. Hiervoor moeten eerst de input bindings geactiveerd worden wat je kunt doen door op de verschillende bindings te klikken en een moment wacht om het systeem op de achtergrond de data te sample data te laten bepalen (je weet dat dit klaar is zodra er een document icoontje verschijnt achter de input binding). Zodra je dit voor alle inputs gedaan hebt kun met deze data de query draaien door op de ‘Test query’ knop te drukken. Door na het uitvoeren van de query op de output bindings te klikken kun je het resultaat per output binding inzien.

Let op: In de uitwerkingen folder staat per stap een oplossing beschreven. Hierin wordt gebruik gemaakt van output namen zoals wij ze gedefinieerd hebben. Heeft jou output een andere naamgeving? Pas deze dan aan om de query te kunnen uitvoeren.

## STAP 1:

Test of de input binding van de event hub en de output binding van het storage account goed werken. Doe dit door een query te schrijven die alle input direct naar een output weg te schrijven (kies 1 van de 2 blob storage outputs). Het testen kan door gebruik te maken van de test query.

## STAP 2:

Verrijk de query met de locatie data uit de metadata DB.

Routeer de device telemetry op basis van de device locatie (beschikbaar in de metadata) naar de correcte output blob containers in het storage account. Dus: devices uit Utrecht naar de ‘Utrecht’ container en devices uit Amsterdam naar de ‘Amsterdam’ container.

TIP: gebruik eerst een query in de ‘WITH’ clause om de locatie gegevens erbij te verrijken, om vervolgens in 2 output queries de data weg te schrijven naar de 2 blob containers.

## STAP 3:

Controleer van ieder bericht of de data uitschieters heeft in de temperatuur. Selecteer alleen de telemetry met een temperature waarde van onder de -40 of boven de 60; deze temperaturen duiden op kapotte temperatuursensoren in het Nederlandse weer.

## STAP 4:

Constateer per device wanneer er 3 fouten of meer op een rij optreden.

TIP: Maak hiervoor gebruik van nieuwe query die het volgende HoppingWindow gebruikt:

HoppingWindow(second, 12, 4)

Gebruik tevens in de output queries system.timestamp() als timestamp aanduiding.

## STAP 5:

Naast de uitschieters in temperature willen we ook controleren of er een power waarde wordt meegegeven.

5A: Selecteer alle data waarvoor geldt dat de waarde van Power NULL is (of er is een power element in het bericht en deze bevat de waarde NULL, of het power element komt helemaal niet voor in het bericht (JavaScript undefined)).

5B: Controleer of Power wordt meegegeven in het bericht maar expliciet NULL is.

Selecteer de data waarbij WEL een power element in het bericht zit. Als er geen power element voorkomt tellen we het bericht als valide. Om het verschil te kunnen detecteren tussen expliciet null en een element wat niet voorkomt in het bericht kun je gebruik maken van een User Defined Function om middels JavaScript code het verschil te duiden.

## STAP 6:

We berichten nu elke keer op het moment dat er 3 fouten op een rij voorkomen. Dus als er 18 fouten op een rij voorkomen, zullen wij dit 16 keer in de output plaatsen. Pas de query aan zodat een afwijking die 3 of meer keer op rij plaatsvindt maar 1 keer doorgegeven wordt. Hiervoor kun je de data set joinen met zichzelf en in de join conditie middels het ‘DATEDIFF’ keyword aangeven dat er ongeveer 4 seconden tussen de gejoinde queries moet zitten. Dus bijvoorbeeld voor de join van 2 queries ‘old’ en ‘new’ krijg je dan het statement:

DATEDIFF(millisecond, old, new) BETWEEN 3000 AND 5000

## STAP 7:

Als laatste stap willen we ook nog gaan bepalen wanneer de afwijking stopt (dus wanneer de telemetry weer helemaal ‘goed’ is). Detecteer hiervoor dat er in de laatste 3 tijdperiodes nog maar 2 of minder afwijkingen plaatsvinden, en geef dit slechts 1 keer door. Geef daarnaast door of het bericht wordt verstuurd omdat het een ‘RAISE’ is van de afwijking, of dat het dus juist een ‘RESOLVE’ betreft.