

Lab 051: Fabric IQ ? Real-Time Intelligence Agents

Nível: L300 Caminho: Todos os caminhos Tempo: ~75 min ? Custo: Gratuito ? Usa conjunto de dados simulado incluso (capacidade do Fabric opcional)

O Que Você Vai Aprender

Como Eventstreams ingerem dados de sensores IoT em tempo real no Microsoft Fabric Consultar dados de streaming com KQL (Kusto Query Language) para detecção instantânea de anomalias Detectar anomalias de temperatura, umidade e estoque usando regras baseadas em limites Usar o Fabric Activator para disparar alertas automatizados quando condições são atendidas Construir um agente de IA que lê dados em tempo real e apresenta insights acionáveis Analisar padrões de atividade de armazém (aberturas de porta, frequência de sensores) entre locais

Introdução

Pipeline RTI do Fabric

Real-Time Intelligence (RTI) no Microsoft Fabric é uma plataforma totalmente gerenciada para capturar, transformar e agir sobre dados de streaming ? tudo dentro do workspace do Fabric. Diferente de pipelines batch tradicionais que processam dados horas ou dias após a chegada, o RTI oferece visibilidade em sub-segundos do que está acontecendo *agora*.

O Cenário

A OutdoorGear Inc. opera 5 armazéns nos EUA (Nova York, Los Angeles, Chicago, Dallas e Seattle). Cada armazém é equipado com sensores IoT que monitoram quatro métricas principais:

Tipo de Sensor | O Que Mede | Limite Crítico | temperature | Temperatura ambiente (°C) | > 30°C (risco de dano ao produto) | humidity | Umidade relativa (%) | > 80% (risco de dano por umidade) | stock_level | Unidades restantes no compartimento | 30°C)

Equivalente KQL:

[diagram or code sample]

Equivalente Python:

[diagram or code sample]

Saída esperada:

[diagram or code sample]

warning "Alerta Crítico" WH-NYC a 38°C está perigosamente alto ? produtos perecíveis e eletrônicos podem ser danificados acima de 35°C. Uma regra do Activator notificaria imediatamente o gerente do armazém de NYC e acionaria o sistema HVAC.

3b. Estoque Crítico (80%)

Equivalente KQL:

[diagram or code sample]

Equivalente Python:

[diagram or code sample]

Saída esperada:

[diagram or code sample]

Etapa 4: Análise de Atividade do Armazém

Analise os eventos de abertura de porta por armazém ? atividade incomumente alta pode indicar preocupações de segurança ou períodos de pico de expedição:

[diagram or code sample]

Saída esperada:

[diagram or code sample]

tip "Insight" WH-DAL lidera com 14 aberturas de porta no total ? combinado com suas duas anomalias de temperatura (32°C e 35°C), aberturas frequentes de porta podem estar permitindo a entrada de ar quente. Um agente de IA correlacionaria esses sinais: _"O

armazém de Dallas tem alta atividade de portas E temperaturas crescentes ? considere adicionar uma cortina de ar na doca de carregamento 3." _

Etapa 5: Construir um Painel de Alertas

Combine todas as anomalias em um único painel resumido que um agente de IA apresentaria a um gerente de operações de armazém:

[diagram or code sample]

Saída esperada:

[diagram or code sample]

info "Integração com Agente de IA" Em produção, um agente de IA consultaria o Eventhouse via KQL, executaria essa lógica de detecção de anomalias e postaria um resumo em linguagem natural em um canal do Teams ou e-mail. O Fabric Activator cuida dos alertas automatizados, enquanto o agente de IA fornece a *interpretação* ? transformando números brutos em recomendações acionáveis.

? Exercício de Correção de Bugs

O arquivo lab-051/broken_alerting.py tem 3 bugs que produzem resultados incorretos de detecção de anomalias. Execute os auto-testes:

[diagram or code sample]

Você deverá ver 3 testes falhando:

Teste | O que verifica | Dica | Teste 1 | O limite de temperatura é parametrizado | O limite está fixo em 50 em vez de usar o parâmetro threshold | Teste 2 | O alerta de estoque usa a comparação correta | Alertas de estoque disparam quando o valor está *abaixo* do limite, não acima | Teste 3 | A taxa de anomalias é calculada por armazém | O denominador deve ser eventos *por armazém*, não eventos totais de todos os armazéns |

Corrija todos os 3 bugs e execute novamente até ver ? All 3 tests passed.

? Verificação de Conhecimento

question "Q1 (Múltipla Escolha): O que é um Eventstream no Microsoft Fabric?"

- A) Um pipeline de processamento de dados em lote que roda em agendamento - B) Um pipeline gerenciado de ingestão de dados em tempo real que captura eventos de streaming - C) Uma ferramenta de visualização para criar painéis - D) Um serviço de treinamento de modelos de machine learning

??? success "? Revelar Resposta" Correto: B) Um pipeline gerenciado de ingestão de dados em tempo real que captura eventos de streaming

Um Eventstream é o ponto de entrada para dados em tempo real no Fabric. Ele captura eventos de fontes como IoT Hub, Kafka ou aplicações personalizadas, os transforma em trânsito e os direciona para destinos como um Eventhouse para consulta. Diferente de pipelines batch, Eventstreams processam dados continuamente com latência de sub-segundo.

question "Q2 (Múltipla Escolha): O que o Fabric Activator faz?"

- A) Otimiza o desempenho de consultas KQL - B) Gerencia a capacidade de armazenamento do Eventhouse - C) Dispara ações automatizadas quando condições de dados são atendidas - D) Converte dados em lote para formato de streaming

??? success "? Revelar Resposta" Correto: C) Dispara ações automatizadas quando condições de dados são atendidas

O Activator é o motor de regras do Fabric para alertas em tempo real. Você define condições (ex.: "temperature > 30°C") e ações (ex.: enviar uma notificação do Teams, disparar um fluxo do Power Automate). Ele monitora continuamente os resultados das consultas KQL e dispara alertas no momento em que uma condição é atendida ? sem necessidade de polling.

question "Q3 (Execute o Lab): Quantas leituras de temperatura excedem 30°C?"

Filtre o DataFrame de eventos para sensor_type == "temperature" e value > 30.

??? success "?" Revelar Resposta" 3

Três anomalias de temperatura excedem 30°C: WH-NYC a 38°C, WH-DAL a 35°C e WH-DAL a 32°C. A leitura de NYC a 38°C é a mais crítica ? bem acima do limite de 35°C para danos ao produto.

question "Q4 (Execute o Lab): Qual armazém tem mais door_opens?"

Filtre por sensor_type == "door_opens", agrupe por warehouse_id e some os valores.

??? success "?" Revelar Resposta" WH-DAL (14 aberturas de porta no total)

Dallas lidera com 14 aberturas de porta no total, seguido por WH-NYC (12), WH-SEA (10), WH-CHI (9) e WH-LAX (7). Combinado com as duas anomalias de temperatura de Dallas, a alta atividade de portas pode estar contribuindo para o acúmulo de calor.

question "Q5 (Execute o Lab): Quantas leituras de estoque estão criticamente baixas (Phase

3 · Simulação / Caminho Real

Simulação Caminho Real

Preserve the existing free simulation and its fixtures. Estimated cost: US\$0.

Reuse the same acceptance criteria against the documented hosted service. Before running, record a service-specific estimate and a hard spending cap. After validation: Exclua implantações, endpoints, índices, traces e dados de teste após a validação.

Kit de apoio

O build do site gera downloads estáveis em PDF e ZIP. O ZIP contém o guia do lab e todos os arquivos de apoio declarados para este lab.

Baixar PDF gerado Baixar ZIP gerado

Supporting file: broken_alerting.py

/usr/bin/env python3 """" ? Broken Alerting Engine ? Lab 051 Bug-Fix Exercise

Analyzes sensor events for anomalies but has 3 bugs. Fix them and run the self-tests to verify.

Usage: python lab-051/broken_alerting.py """"

import io import pandas as pd

def detect_temp_anomalies(events: pd.DataFrame, threshold: float = 30.0) -> pd.DataFrame:

""""Return temperature events above the threshold."""" temp = events[events["sensor_type"] == "temperature"] # ? Bug #1: Threshold is hardcoded to 50 instead of using the parameter
return temp[temp["value"] > 50]

def detect_low_stock(events: pd.DataFrame, threshold: float = 10.0) -> pd.DataFrame:

""""Return stock_level events below the threshold (critical stock)."""" stock = events[events["sensor_type"] == "stock_level"] # ? Bug #2: Uses > instead of threshold]

def calculate_anomaly_rate(events: pd.DataFrame, warehouse_id: str, anomaly_count: int) ->

float: """"Return the anomaly rate (%) for a specific warehouse."""" # ? Bug #3: Divides by total events across ALL warehouses # instead of events for this specific warehouse total = len(events) return anomaly_count / total * 100

def run_tests() -> bool: test_csv = """"\

timestamp,warehouse_id,warehouse_city,sensor_type,value,unit

2026-01-01T00:00:00Z,W1,City1,temperature,35.0,celsius

2026-01-01T00:01:00Z,W1,City1,temperature,22.0,celsius

2026-01-01T00:02:00Z,W1,City1,stock_level,3.0,units

2026-01-01T00:03:00Z,W1,City1,stock_level,50.0,units

2026-01-01T00:04:00Z,W2,City2,temperature,28.0,celsius

2026-01-01T00:05:00Z,W2,City2,stock_level,15.0,units""""

events = pd.read_csv(io.StringIO(test_csv)) passed = 0 failed = 0

?? Test 1: Temperature anomalies above 30°C ?????????????????????? # Only W1 at 35°C ?

```

1 anomaly anomalies = detect_temp_anomalies(events, threshold=30.0) if len(anomalies) == 1:
print(f"? Test 1 PASSED: temp anomalies = {len(anomalies)}") passed += 1 else: print(f"?
Test 1 FAILED: temp anomalies = {len(anomalies)} (expected 1)") failed += 1
# ?? Test 2: Low stock below 10 units ????????????????????????????????????????? # Only W1 at 3 ? 1
critical stock event low = detect_low_stock(events, threshold=10.0) if len(low) == 1:
print(f"? Test 2 PASSED: low stock events = {len(low)}") passed += 1 else: print(f"? Test 2
FAILED: low stock events = {len(low)} (expected 1)") failed += 1
# ?? Test 3: Anomaly rate for warehouse W1 ????????????????????????????????????? # W1 has 4 events,
1 anomaly ? 1/4 = 25% rate = calculate_anomaly_rate(events, "W1", 1) if abs(rate - 25.0) <
0.1: print(f"? Test 3 PASSED: W1 anomaly rate = {rate:.1f}%") passed += 1 else: print(f"?
Test 3 FAILED: W1 anomaly rate = {rate:.1f}% (expected 25.0%)") failed += 1
print() if failed == 0: print(f"? All 3 tests passed ? great debugging!") else: print(f"?
{failed} test(s) failed ? keep debugging!") return failed == 0
if __name__ == "__main__": print(f"? Running self-tests for broken_alerting.py ...\\n")
run_tests()

```

Supporting file: sensor_events.csv

```

timestamp,warehouse_id,warehouse_city,sensor_type,value,unit
2026-03-01T10:00:00Z,WH-NYC,New York,temperature,22.5,celsius
2026-03-01T10:00:30Z,WH-NYC,New York,humidity,55.0,percent 2026-03-01T10:01:00Z,WH-NYC,New
York,stock_level,42,units 2026-03-01T10:02:00Z,WH-NYC,New York,temperature,38.0,celsius
2026-03-01T10:03:00Z,WH-NYC,New York,stock_level,45,units 2026-03-01T10:04:00Z,WH-NYC,New
York,humidity,60.0,percent 2026-03-01T10:05:00Z,WH-NYC,New York,door_opens,5,count
2026-03-01T10:06:00Z,WH-NYC,New York,temperature,21.0,celsius
2026-03-01T10:07:00Z,WH-NYC,New York,stock_level,40,units 2026-03-01T10:08:00Z,WH-NYC,New
York,door_opens,3,count 2026-03-01T10:00:00Z,WH-LAX,Los Angeles,temperature,24.0,celsius
2026-03-01T10:01:00Z,WH-LAX,Los Angeles,humidity,45.0,percent
2026-03-01T10:02:00Z,WH-LAX,Los Angeles,stock_level,30,units
2026-03-01T10:03:00Z,WH-LAX,Los Angeles,temperature,25.5,celsius
2026-03-01T10:04:00Z,WH-LAX,Los Angeles,stock_level,8,units 2026-03-01T10:05:00Z,WH-LAX,Los
Angeles,humidity,50.0,percent 2026-03-01T10:06:00Z,WH-LAX,Los Angeles,door_opens,4,count
2026-03-01T10:07:00Z,WH-LAX,Los Angeles,temperature,23.0,celsius
2026-03-01T10:08:00Z,WH-LAX,Los Angeles,stock_level,3,units 2026-03-01T10:09:00Z,WH-LAX,Los
Angeles,door_opens,6,count 2026-03-01T10:00:00Z,WH-CHI,Chicago,temperature,20.0,celsius
2026-03-01T10:01:00Z,WH-CHI,Chicago,humidity,65.0,percent
2026-03-01T10:02:00Z,WH-CHI,Chicago,stock_level,55,units
2026-03-01T10:03:00Z,WH-CHI,Chicago,temperature,19.5,celsius
2026-03-01T10:04:00Z,WH-CHI,Chicago,stock_level,60,units
2026-03-01T10:05:00Z,WH-CHI,Chicago,humidity,85.0,percent
2026-03-01T10:06:00Z,WH-CHI,Chicago,door_opens,3,count
2026-03-01T10:07:00Z,WH-CHI,Chicago,temperature,21.0,celsius
2026-03-01T10:08:00Z,WH-CHI,Chicago,stock_level,52,units
2026-03-01T10:09:00Z,WH-CHI,Chicago,door_opens,2,count
2026-03-01T10:00:00Z,WH-DAL,Dallas,temperature,28.0,celsius
2026-03-01T10:01:00Z,WH-DAL,Dallas,humidity,40.0,percent
2026-03-01T10:02:00Z,WH-DAL,Dallas,stock_level,38,units
2026-03-01T10:03:00Z,WH-DAL,Dallas,temperature,32.0,celsius
2026-03-01T10:04:00Z,WH-DAL,Dallas,stock_level,35,units

```

2026-03-01T10:05:00Z,WH-DAL,Dallas,humidity,38.0,percent
2026-03-01T10:06:00Z,WH-DAL,Dallas,door_opens,8,count
2026-03-01T10:07:00Z,WH-DAL,Dallas,temperature,35.0,celsius
2026-03-01T10:08:00Z,WH-DAL,Dallas,stock_level,40,units
2026-03-01T10:09:00Z,WH-DAL,Dallas,door_opens,6,count
2026-03-01T10:00:00Z,WH-SEA,Seattle,temperature,18.0,celsius
2026-03-01T10:01:00Z,WH-SEA,Seattle,humidity,70.0,percent
2026-03-01T10:02:00Z,WH-SEA,Seattle,stock_level,30,units
2026-03-01T10:03:00Z,WH-SEA,Seattle,temperature,17.5,celsius
2026-03-01T10:04:00Z,WH-SEA,Seattle,stock_level,28,units
2026-03-01T10:05:00Z,WH-SEA,Seattle,humidity,75.0,percent
2026-03-01T10:06:00Z,WH-SEA,Seattle,door_opens,3,count
2026-03-01T10:07:00Z,WH-SEA,Seattle,temperature,19.0,celsius
2026-03-01T10:08:00Z,WH-SEA,Seattle,stock_level,25,units
2026-03-01T10:09:00Z,WH-SEA,Seattle,door_opens,1,count