

Lab 051: Fabric IQ ? Real-Time Intelligence Agents

Level: L300 Path: All paths Time: ~75 min ? Cost: Free ? Uses included mock dataset (Fabric capacity optional)

What You'll Learn

How Eventstreams ingest real-time IoT sensor data into Microsoft Fabric Query streaming data with KQL (Kusto Query Language) for instant anomaly detection Detect temperature, humidity, and stock anomalies using threshold-based rules Use Fabric Activator to trigger automated alerts when conditions are met Build an AI agent that reads real-time data and surfaces actionable insights Analyze warehouse activity patterns (door opens, sensor frequency) across locations

Introduction

Fabric RTI Pipeline

Real-Time Intelligence (RTI) in Microsoft Fabric is a fully managed platform for capturing, transforming, and acting on streaming data ? all within the Fabric workspace. Unlike traditional batch pipelines that process data hours or days after it arrives, RTI gives you sub-second visibility into what's happening *right now*.

The Scenario

OutdoorGear Inc. operates 5 warehouses across the US (New York, Los Angeles, Chicago, Dallas, and Seattle). Each warehouse is equipped with IoT sensors monitoring four key metrics:

Sensor Type | What It Measures | Critical Threshold | temperature | Ambient temperature (°C) | > 30°C (product damage risk) | humidity | Relative humidity (%) | > 80% (moisture damage risk) | stock_level | Units remaining in bin | 30°C)

KQL equivalent:

[diagram or code sample]

Python equivalent:

[diagram or code sample]

Expected output:

[diagram or code sample]

warning "Critical Alert" WH-NYC at 38°C is dangerously high ? perishable goods and electronics can be damaged above 35°C. An Activator rule would immediately notify the NYC warehouse manager and trigger the HVAC system.

3b. Stock Critical (80%)

KQL equivalent:

[diagram or code sample]

Python equivalent:

[diagram or code sample]

Expected output:

[diagram or code sample]

Step 4: Warehouse Activity Analysis

Analyze door open events per warehouse ? unusually high activity may indicate security concerns or peak shipping periods:

[diagram or code sample]

Expected output:

[diagram or code sample]

tip "Insight" WH-DAL leads with 14 total door opens ? combined with its two temperature anomalies (32°C and 35°C), frequent door openings could be letting hot air in. An AI agent would correlate these signals: _"Dallas warehouse has high door activity AND rising

temperatures ? consider adding an air curtain to loading dock 3."_

Step 5: Build an Alert Dashboard

Combine all anomalies into a single summary dashboard that an AI agent would present to a warehouse operations manager:

[diagram or code sample]

Expected output:

[diagram or code sample]

info "AI Agent Integration" In production, an AI agent would query the Eventhouse via KQL, run this anomaly detection logic, and post a natural-language summary to a Teams channel or email. Fabric Activator handles the automated alerting, while the AI agent provides the *interpretation* ? turning raw numbers into actionable recommendations.

? Bug-Fix Exercise

The file lab-051/broken_alerting.py has 3 bugs that produce incorrect anomaly detection results. Run the self-tests:

[diagram or code sample]

You should see 3 failed tests:

Test | What it checks | Hint | Test 1 | Temperature threshold is parameterized | The threshold is hardcoded to 50 instead of using the threshold parameter | Test 2 | Stock alert uses correct comparison | Stock alerts trigger when value is *below* the threshold, not above | Test 3 | Anomaly rate is calculated per warehouse | The denominator should be events *per warehouse*, not total events across all warehouses |

Fix all 3 bugs and re-run until you see ? All 3 tests passed.

? Knowledge Check

question "Q1 (Multiple Choice): What is an Eventstream in Microsoft Fabric?"

- A) A batch data processing pipeline that runs on a schedule - B) A managed real-time data ingestion pipeline that captures streaming events - C) A visualization tool for creating dashboards - D) A machine learning model training service

??? success "? Reveal Answer" Correct: B) A managed real-time data ingestion pipeline that captures streaming events

An Eventstream is the entry point for real-time data in Fabric. It captures events from sources like IoT Hub, Kafka, or custom applications, transforms them in-flight, and routes them to destinations like an Eventhouse for querying. Unlike batch pipelines, Eventstreams process data continuously with sub-second latency.

question "Q2 (Multiple Choice): What does Fabric Activator do?"

- A) Optimizes KQL query performance - B) Manages Eventhouse storage capacity - C) Triggers automated actions when data conditions are met - D) Converts batch data into streaming format

??? success "? Reveal Answer" Correct: C) Triggers automated actions when data conditions are met

Activator is Fabric's rule engine for real-time alerting. You define conditions (e.g., "temperature > 30°C") and actions (e.g., send a Teams notification, trigger a Power Automate flow). It continuously monitors KQL query results and fires alerts the moment a condition is met ? no polling required.

question "Q3 (Run the Lab): How many temperature readings exceed 30°C?"

Filter the events DataFrame for sensor_type == "temperature" and value > 30.

??? success "? Reveal Answer" 3

Three temperature anomalies exceed 30°C: WH-NYC at 38°C, WH-DAL at 35°C, and WH-DAL at 32°C. The NYC reading at 38°C is the most critical ? well above the 35°C threshold for

product damage.

question "Q4 (Run the Lab): Which warehouse has the most door_opens?"

Filter for sensor_type == "door_opens", group by warehouse_id, and sum the values.

??? success "?" Reveal Answer" WH-DAL (14 total door opens)

Dallas leads with 14 total door opens, followed by WH-NYC (12), WH-SEA (10), WH-CHI (9), and WH-LAX (7). Combined with Dallas's two temperature anomalies, the high door activity may be contributing to heat buildup.

question "Q5 (Run the Lab): How many stock readings are critically low (Phase 3 ·

Simulation / Real path

Simulation Real path

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: Delete hosted deployments, endpoints, indexes, traces, and test data after validation.

Supporting kit

The site build generates stable PDF and ZIP downloads. The ZIP contains the lab guide and every supporting file declared for this lab.

Download generated PDF Download generated ZIP

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