Santa's Digital Workshops Showcase 🎅

Real-Time Event-Driven AI for Christmas

❄️ **Drasi + Microsoft Agent Framework** ❄️

⭐ Festive Tech Calendar 2025 ⭐

Luke Murray • @lukemurraynz

👨‍💻 About Me

Luke Murray

🔗 **LinkedIn:** linkedin.com/in/ljmurray

🐦 **Twitter/X:** @lukemurraynz

💼 **ISM (Infra, Security and Modern Workplace Service Lead) at HSO** specializing in Azure

🎯 **Microsoft MVP** | Cloud & DevOps enthusiast

Passionate about building scalable cloud solutions

🎁 What We're Building This Holiday Season

🎄 The North Pole's Tech Stack

- 🔵 **Drasi** - Real-time event graph detection (<5s latency)
- 🟢 **Microsoft Agent Framework** - Multi-agent AI orchestration
- ⭐ **Azure OpenAI** - Intelligent, context-aware recommendations
- ❄️ **Event-Driven Architecture** - EventHub → Drasi → Agents

> 🎅 Process millions of children's letters and behavior updates in real-time!

🎄 Live Demo: Christmas Event Detection ❄️

🎅 Let's See It In Action!

🎁 What happens in Santa's Workshop:

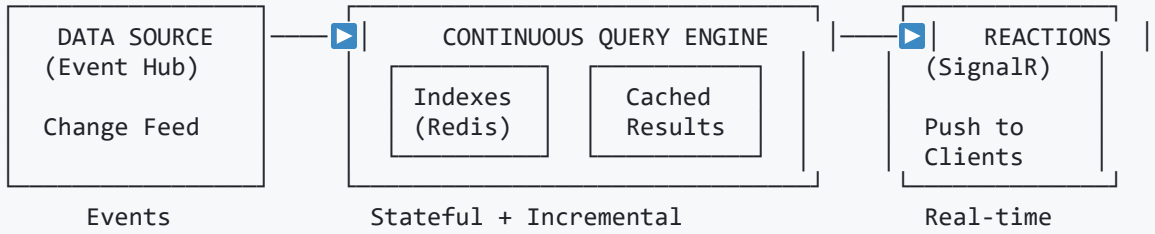
1. 📝 Child submits wishlist via frontend
2. 📨 Backend publishes to EventHub (`wishlist-events`)
3. ⚡ Drasi detects patterns in **<5 seconds**
4. 🔔 Real-time updates via SignalR
5. 🤖 AI agents provide **preference-based** recommendations

🔵 How Drasi Actually Works

It's NOT Just a Query Engine!

Common Misconception: "Drasi queries the database every time"

Reality: Drasi is a **stateful event processing platform** that maintains live query results!



🎯 What Drasi Does

1. **Bootstraps** - Loads initial state from Source
2. **Caches Results** - Stores in memory/Redis
3. **Incremental Updates** - Only processes changes
4. **Pushes Deltas** - `{added:[], updated:[], deleted:[]}`

⚡ Why It's Fast (<5s)

- No re-computation on each query
- Indexes for O(1) lookups
- Only affected results update
- Pre-computed aggregations

🔄 Drasi Data Access Patterns

Two Ways to Get Real-Time Intelligence

📊 Drasi View API (Pull Pattern)

HTTP REST Endpoint for on-demand querying

```
GET http://{query-id}-view-svc/{query-name}
```

Use Cases:

- 🤖 AI Agents querying trends
- 📈 Dashboard data retrieval
- 🔍 Ad-hoc analysis

Benefits:

- Query when needed
- No persistent connections
- Get current snapshot instantly

Example: Agent calls `QueryTrendingWishlistItems()` → Hits View API → Gets top 10 trending gifts

📡 SignalR (Push Pattern)

WebSocket Hub for real-time notifications

```
connection.on("DrasiUpdate", (data) => {
  // Live update received!
});
```

Use Cases:

- 💻 Frontend live updates
- 🔔 Real-time notifications
- 📊 Live dashboards

🟢 What is Microsoft Agent Framework?

Multi-Agent AI Orchestration for .NET

🎯 Core Concept

Build specialized AI agents that work together, call tools, and collaborate to solve complex problems.

Think: "A team of experts, each with their own role and access to live data"

⚡ Key Features

- **Agent Composition** - Create multiple specialized agents
- **Tool Calling** - Give agents access to functions and APIs
- **Streaming Responses** - Real-time token-by-token output
- **State Management** - Track conversations and context
- **Azure OpenAI Integration** - GPT-4, GPT-4o, and more

🎅 Santa's Workshop Example

```
// Create specialized elf agent
var analystAgent = chatClient.CreateAIAgent(
    name: "BehaviorAnalyst",
    instructions: """
    Analyze child behavior and recommend gifts.
    Use tools to access:
    - Wishlist data
    - Behavior history
    - Real-time trends from Drasi
    """,
    tools: elfTools
);

// Agent automatically calls tools
var response = await analystAgent
    .InvokeAsync("Recommend gifts for child-123");
```

Result: AI agent analyzes behavior, checks trends, and suggests personalized gifts!

🌐 Learn More

🎨 AG-UI & CopilotKit

The UI Layer: Connecting Users to AI Agents

🎯 AG-UI Protocol

A **standard protocol** for agent-to-UI communication (like HTTP for AI agents!)

Key Features:

- **Standardized Events** - RUN_STARTED, TEXT_MESSAGE_CONTENT, TOOL_CALLED
- **Real-time Streaming** - SSE (Server-Sent Events) for live updates
- **Tool Transparency** - See exactly which tools agents call
- **Cancellation Support** - Stop runaway executions

📦 Implementation

```
import { HttpAgent } from "@ag-ui/client";

const agent = new HttpAgent({
  url: "/api/v1/agents/elf/run",
  agentId: "elf",
});

agent.on("text", (delta) => {
  console.log(delta); // Stream tokens
});
```

Spec: github.com/ag-ui/ag-ui

⚡ CopilotKit

React components that work seamlessly with AG-UI agents.

Key Features:

- **Drop-in Components** - `<CopilotChat>`, `<CopilotTextarea>`
- **Context Injection** - Pass app state to agents automatically
- **Generative UI** - Agents render custom React components

📚 Engineering Frameworks Followed

⭐ Microsoft Best Practices & Standards

📘 ISE (Industry Solution Engineering) Engineering Playbook

- ✅ **CI/CD Automation** - `azd` hooks & validation
- ✅ **Infrastructure as Code** - Bicep modules
- ✅ **Testing** - xUnit unit & integration tests
- ✅ **Security** - Managed Identity, Key Vault
- ✅ **Observability** - Structured logging, health endpoints
- ✅ **Documentation** - Comprehensive guides & runbooks

🌐 Microsoft API Guidelines

- ✅ **URL Versioning** - `/api/v1/` pattern
- ✅ **HTTP Status Codes** - Proper 200/201/404/400
- ✅ **ETag Support** - Conditional requests (If-Match)
- ✅ **Health Endpoints** - `/healthz`, `/readyz`, `/livez`
- ✅ **Error Handling** - RFC 7807 Problem Details
- ✅ **CORS** - Proper cross-origin configuration

🎯 **Overall Compliance Score: 8.1/10** - Production-ready with enterprise standards!

📝 **Improvement Opportunities:**
• OpenAPI response annotations (`ProducesResponseType`) • Distributed tracing (Application Insights) • Code coverage metrics & gates • API pagination standards • Rate limiting headers

🎄 Santa's Tech Stack ❄️

🔵 Drasi 0.10.0-azure

- ⭐ **Continuous query engine** (Cypher)
- 🎯 Real-time pattern detection
- ⚡ <5 second event latency
- 📡 SignalR live updates
- 🔍 View API for querying
- 🔗 **Dapr** service mesh (internal)

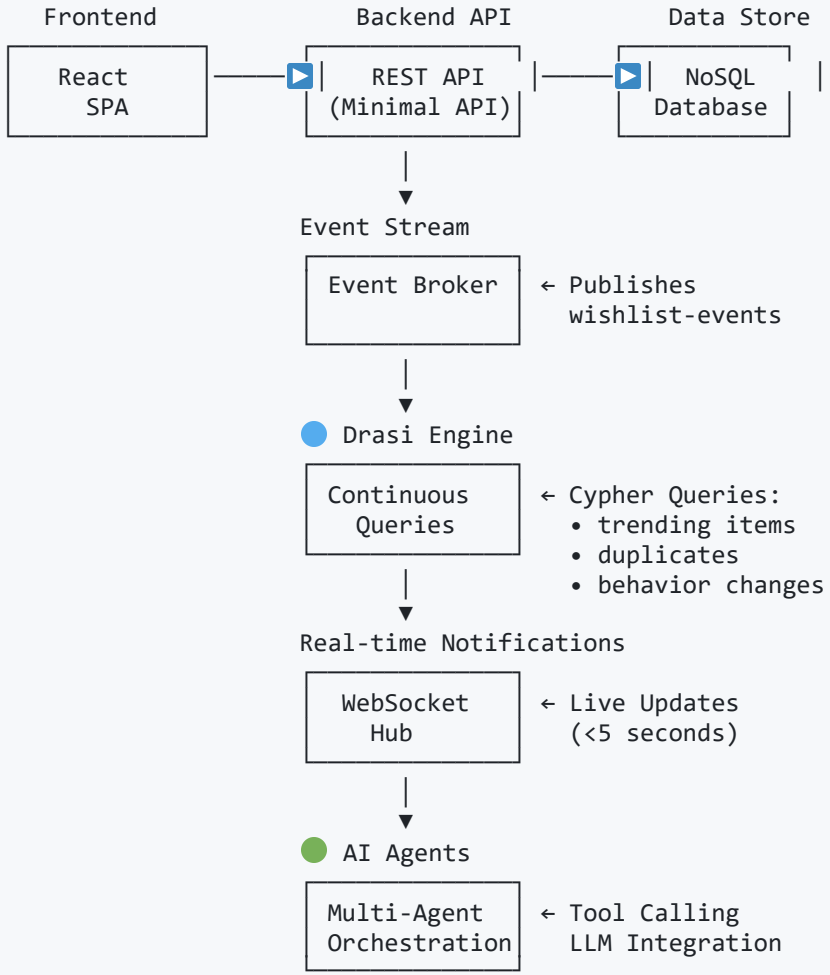
🟢 Microsoft Agent Framework

- 🎅 **Multi-agent orchestration**
- 🛠️ Tool calling (10+ tools)
- 🤖 Azure OpenAI (GPT-4)
- 📨 Streaming SSE responses
- ✨ AIFunctionFactory pattern

🎄 Logical Architecture Flow ❄️

How the Components Work Together

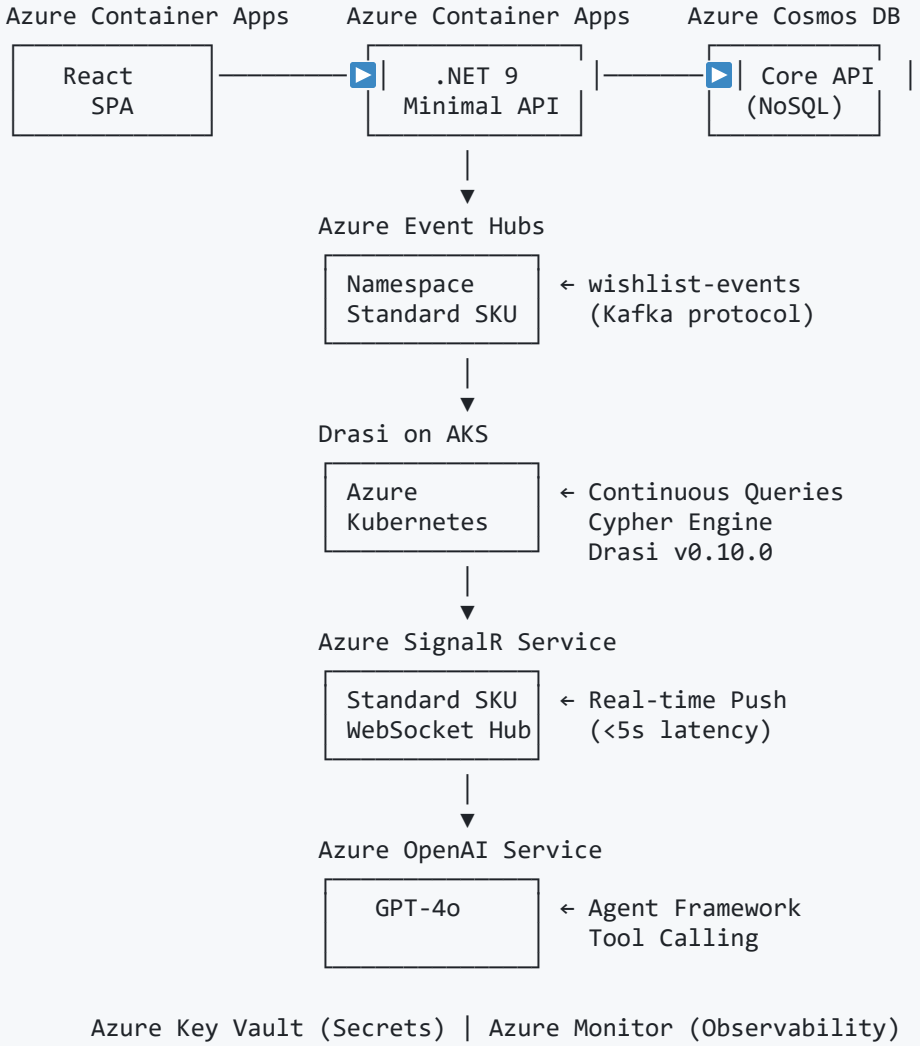
🔵 Drasi 🟢 AI/Agent 🔵 Infrastructure



Azure Architecture

Production Infrastructure on Azure

Azure Services



🎅 Drasi Continuous Query

🎁 Real Cypher Query for Trending Christmas Gifts

```
MATCH (w:`wishlist-events`)
WHERE drasi.changeDateTime(w) >= datetime() - duration({ minutes: 60 })
  AND w.requestType = 'gift'
RETURN w.text AS item,
       count(w) AS frequency
```

What this does:

- Monitors `wishlist-events` in real-time
- Filters last 60 minutes of gift requests
- Groups by item name and counts frequency
- **Maintains a LIVE result set** (not re-queried each time!)

Result: Elves see what's hot RIGHT NOW - with cached, instant responses!

🎄 2025 Trending Now 🎅

⭐ What Kids Are Asking Santa For This Year

🔥 Top 5 Hot Items

1. 🎮 **LEGO Star Wars Set** - 45 requests
2. 🎮 **Nintendo Switch** - 38 requests
3. 📚 **Harry Potter Book Collection** - 32 requests
4. 🚗 **Remote Control Car** - 28 requests
5. 🎨 **Art Supply Kit** - 25 requests

📊 Real-Time Insights

- ⚡ **Drasi Detection:** <5 second latency
- 🔄 **Live Updates:** SignalR WebSockets
- 📈 **Trending Window:** Last 60 minutes
- 🎯 **Duplicate Detection:** Smart grouping

🎁 **Powered by Drasi continuous queries monitoring wishlist-events in real-time!**

🎁 Elf's Tool Registration

⭐ AIFunctionFactory Pattern (Santa's Workshop Edition)

```
IList<AITool> tools = new List<AITool>
{
    // Child's actual Christmas wishlist
    AIFunctionFactory.Create(_toolLibrary.GetChildWishlistItems),

    // Nice/Naughty profile and behavior
    AIFunctionFactory.Create(_toolLibrary.GetChildBehaviorHistory),

    // Drasi real-time Christmas queries
    AIFunctionFactory.Create(_toolLibrary.QueryTrendingWishlistItems),
    AIFunctionFactory.Create(_toolLibrary.FindChildrenWithDuplicateWishlists),

    // Santa's inventory management
    AIFunctionFactory.Create(_toolLibrary.SearchGiftInventory),
    AIFunctionFactory.Create(_toolLibrary.CheckBudgetConstraints)
};
```

🎄 10+ Magical Tools Connected to Live Data!

🎅 Creating Specialized Elf Agents

Santa's AI-Powered Workshop Elves

```
_analystAgent = chatClient.CreateAIAgent(
    name: "BehaviorAnalyst",
    instructions: """
    You are the Behavior Analyst Elf at the North Pole.

    CRITICAL: ALWAYS call GetChildWishlistItems FIRST!
    This shows you exactly what the child requested for Christmas.

    You have access to DRASI real-time tools:
    - QueryTrendingWishlistItems: See what's hot NOW
    - FindChildrenWithDuplicateWishlists: Strong preferences

    WORKFLOW:
    1. Call GetChildWishlistItems (actual Christmas requests)
    2. Query trending holiday data with Drasi
    3. Check Nice/Naughty behavior history
    4. Provide detailed festive analysis
    """,
    tools: tools  // ← All 10+ magical tools available!
);
```

🎁 Elf Tool Implementation

🎄 GetChildWishlistItems Tool (Christmas Edition)

```
[Description("🎁 CRITICAL: Gets child's actual Christmas wishlist")]
public async Task<string> GetChildWishlistItems(
    [Description("Child ID")] string childId,
    CancellationToken ct = default)
{
    var items = new List<WishlistItemEntity>();
    await foreach (var item in _wishlistRepository.ListAsync(childId))
    {
        items.Add(item);
    }

    return $"""
        🎁 CHILD'S CHRISTMAS WISHLIST ({childId})
        {items.Count} gift(s) requested

        💡 Base recommendations on THESE festive items!
        """;
}
```

🎯 Preference-Based AI Recommendations

AI That Actually Listens to What Kids Ask For!

🎁 How It Works

When a child submits a wishlist item:

1. ⭐ **Item stored** in profile preferences
2. 🤖 **AI receives context** about what child wants
3. 🎯 **First recommendation** = the exact item requested
4. 🔗 **Related items** = accessories, similar products

Example: Child asks for "Xbox"

🎮 AI Response

```
[
  { "suggestion": "Xbox Series X", "rationale": "You asked for an Xbox!" },
  { "suggestion": "Xbox Game Pass", "rationale": "Access to hundreds of games!" },
  { "suggestion": "Extra Controller", "rationale": "For multiplayer fun!" }
]
```

Result: Personalized, relevant recommendations that match the child's actual wishes!

🎅 **No more generic recommendations! AI understands what each child actually wants.**

🔵 What is Drasi?

Event-Driven Continuous Intelligence Platform

🎯 Core Concept

Drasi detects **meaningful changes** in your data as they happen and triggers reactions automatically.

Think: "If these 3 things happen within 5 minutes, notify me immediately"

⚡ Key Features

- **Continuous Queries** - Define patterns using Cypher (graph query language)
- **Sub-5 Second Latency** - Real-time pattern detection
- **Event Graph Engine** - Tracks relationships and temporal patterns
- **Multiple Sources** - Cosmos DB, PostgreSQL, Event Hubs, and more

🎁 Santa's Workshop Example

```
// Detect trending gifts
MATCH (w:`wishlist-events`)
WHERE drasi.changeDateTime(w)
  >= datetime() - duration({ minutes: 60 })
  AND w.requestType = 'gift'
RETURN w.text AS item,
       count(w) AS frequency
ORDER BY frequency DESC
```

Result: Instant alerts when LEGO Star Wars becomes the #1 requested gift!

🌐 Learn More

Website: drasi.io

GitHub: github.com/drasi-project/drasi-platform

🎅 Multi-Elf AI Orchestration

🎄 Three Specialized Christmas Elves Working Together

1. 📊 **Behavior Analyst Elf** → Analyzes Nice/Naughty history
   - 🔍 Calls `GetChildBehaviorHistory()`
2. ** Creative Gift Elf** → Proposes magical gift ideas
   - 🎁 Calls `SearchGiftInventory()`
3. ✅ **Quality Reviewer Elf** → Validates recommendations
   - 💰 Calls `CheckBudgetConstraints()`

🎄 **Result:** Coordinated Christmas gift recommendations with festive rationales!

🎅 Santa's Nice & Naughty List Intelligence

😇 Nice Child

- 🎁 Premium toys and games
- 🎄 Fun, rewarding gifts
- ⭐ Rationale: "Earned it!"
- 🎉 Extra treats from Santa!

😈 Naughty Child

- 🪨 **Lump of Coal** (first!)
- 📚 Educational materials
- 📖 Character-building books
- 🎯 Rationale: "Learning opportunity!"

🎄 **Demo:** Generate random Christmas events (70% wishlist, 30% behavior)

🎄 Santa's Workshop Performance Metrics

Component	Metric	Value
🔵 **Drasi Detection**	Event Latency	⚡ **<5 seconds**
🔵 **Drasi Detection**	Query Throughput	🚀 **1000+ events/sec**
🟢 **Agent Framework**	Recommendation Time	⏱️ **3-8s**
🟢 **Agent Framework**	Multi-Agent Flow	🎅 **8-15s**
🟢 **Agent Framework**	Streaming Rate	💨 **~50 tokens/sec**

🎁 **Christmas Scale:** 10K+ concurrent children, 100K+ letters/hour

🌍 Real-World Use Cases

Beyond Santa's Workshop - Where This Pattern Applies

🛒 E-Commerce

- **Cart Abandonment** - Real-time intervention
- **Trending Products** - Live inventory optimization
- **Price Changes** - Dynamic recommendations
- **Fraud Detection** - Pattern recognition

🏥 Healthcare

- **Patient Monitoring** - Vital sign alerts
- **Equipment Status** - Predictive maintenance
- **Bed Management** - Resource optimization
- **Emergency Response** - Priority routing

🏭 IoT & Manufacturing

- **Device State Changes** - Real-time monitoring
- **Quality Control** - Defect detection
- **Supply Chain** - Inventory tracking
- **Predictive Maintenance** - Failure prevention

💰 Financial Services

- **Fraud Detection** - Transaction patterns
- **Market Trends** - Real-time analysis
- **Risk Assessment** - Portfolio monitoring
- **Compliance Alerts** - Regulatory checks

🎯 Key Takeaways

1. **Real-Time Event Detection** → Drasi provides <5s latency with Cypher queries
2. **Multi-Agent AI Orchestration** → Specialized agents with tool calling patterns
3. **Production-Ready Architecture** → Following Microsoft engineering best practices
4. **Tool Calling for Real Data** → 10+ tools accessing live Cosmos DB & Drasi
5. **Streaming UX** → Token-level SSE responses for better user experience

🔧 Lessons Learned & Issues Fixed

Real Challenges from Building This Demo

⚠️ Drasi CLI vs kubectl

Issue: Drasi resources are **NOT Kubernetes CRDs** - they're managed via REST API

Why it fails:

```
# ❌ This doesn't work
kubectl apply -f source.yaml
# Error: no matches for kind "Source"
```

Fix: Use `drasi` CLI exclusively

```
drasi apply -f resource.yaml
drasi list source -n drasi-system
```

⚠️ Port Forwarding Failures

Issue: `drasi apply` failed with "lost connection to pod"

Root Cause: Drasi API pod not ready before applying resources

Fix: Wait for readiness + retry logic

```
kubectl wait --for=condition=available \
  deployment/drasi-api --timeout=180s
# Retry up to 3 times
```

⚠️ Image Registry Double-Prefix

Issue: Malformed image names

```
ghcr.io/drasi-project/ghcr.io/...
```

Root Cause: Manifests had full registry paths; resource provider prepended registry again

💝 Supporting a Great Cause

Festive Tech Calendar 2025 🎄

Fundraising for Beatson Cancer Charity 🎗️

Supporting cancer patients every step of the way

Beatson Cancer Charity provides services, funds specialists, research and education to invest in a better future for cancer patients.

🎁 **Donate:** justgiving.com/page/festive-tech-calendar2025

🌟 **Learn More:** festivetechcalendar.com

Every contribution makes a difference in the lives of cancer patients and their families 💚

🎄 Merry Tech-mas! 🎅

❄️ Real-time events + AI agents = Christmas Magic! ✨

Luke Murray • @lukemurraynz

Happy Holidays! 🎉