

## Lab 074: Foundry Agent Service ? Production Operations Simulation

Level: L300 Path: ? Microsoft Foundry Time: ~120 min ? Cost: Free ? Uses mock agent data

### What You'll Learn

What the Foundry Agent Service is and how it orchestrates production multi-agent systems

How agent types (specialist, orchestrator) work together in a production architecture

Analyze agent fleet health: request volumes, latency, error rates, and status Identify

degraded agents and configuration risks (e.g., disabled content filters) Build a fleet

health dashboard from simulated monitoring data

### Introduction

The Azure AI Foundry Agent Service provides a managed platform for deploying, orchestrating, and monitoring multi-agent systems at enterprise scale. Instead of building custom orchestration, you define agents with specific tools, memory, and models ? and the service handles routing, state management, and scaling.

### Agent Types

Type | Role | Example | Orchestrator | Routes requests to specialists, manages conversation

flow | SupportRouter, Coordinator | Specialist | Handles a specific domain with dedicated

tools and memory | ProductAdvisor, OrderProcessor |

### The Scenario

You are a Platform SRE managing a multi-agent deployment for an e-commerce company. The fleet has 8 agents ? 2 orchestrators and 6 specialists ? running on Azure Container Apps.

You've been alerted that one agent is degraded and need to investigate.

Your dataset (foundry\_agents.csv) contains the current fleet status. Your job: analyze health metrics, identify issues, and produce a fleet status report.

info "Mock Data" This is an operations-analysis lab, not a live deployment lab. It uses a mock agent fleet CSV that mirrors the metrics you'd see in Azure AI Foundry's monitoring dashboard. The patterns (latency spikes, error rates, degraded status) represent common production scenarios.

### Prerequisites

Requirement | Why | Python 3.10+ | Run the analysis scripts | pandas library | Data manipulation |

[diagram or code sample]

tip "Quick Start with GitHub Codespaces" Open in GitHub Codespaces

All dependencies are pre-installed in the devcontainer.

### ? Supporting Files

note "Download these files before starting the lab" Save all files to a lab-074/ folder in your working directory.

File | Description | Download | broken\_foundry.py | Bug-fix exercise (3 bugs + self-tests)

| ? Download | foundry\_agents.csv | Dataset | ? Download |

### Step 1: Understand the Fleet Architecture

Before analyzing data, understand how the agents fit together:

[diagram or code sample]

### Key Configuration Fields

Field | Description | memory\_type | How the agent persists state: cosmos\_db (durable),

ai\_search (vector), session\_only (ephemeral), none | deployment | Infrastructure:

container\_apps (managed) or vm (self-hosted) | content\_filter | Whether Azure AI content

safety is enabled or disabled | status | Agent health: active or degraded |

### Step 2: Load and Explore the Fleet Data

[diagram or code sample]

Expected output:

[diagram or code sample]

Step 3: Analyze Request Volume and Load Distribution

How is traffic distributed across the fleet?

[diagram or code sample]

Expected output:

[diagram or code sample]

Agent | Requests | Share | Coordinator | 3,200 | 34.1% | SupportRouter | 2,100 | 22.4% | ProductAdvisor | 1,250 | 13.3% | OrderProcessor | 890 | 9.5% | QualityReviewer | 780 | 8.3% | InventoryMonitor | 560 | 6.0% | AnalyticsAgent | 420 | 4.5% | LegacyBridge | 180 | 1.9% |  
tip "Insight" The Coordinator orchestrator handles 34% of all traffic ? it's the entry point for most requests. If it goes down, the entire system is affected. The SupportRouter is the second-busiest, routing customer support queries to specialists.

Step 4: Identify Degraded and At-Risk Agents

4a ? Degraded Agents

[diagram or code sample]

Expected output:

[diagram or code sample]

4b ? High Error Rate Agents

[diagram or code sample]

4c ? Content Filter Status

[diagram or code sample]

warning "Security Risk" LegacyBridge (FA08) has its content filter disabled and runs on a self-hosted VM. This is a compliance risk ? all production agents should have content safety enabled, especially those handling customer data.

Step 5: Analyze Memory and Infrastructure Patterns

[diagram or code sample]

[diagram or code sample]

Step 6: Build the Fleet Health Report

[diagram or code sample]

? Bug-Fix Exercise

The file lab-074/broken\_foundry.py contains 3 bugs that produce incorrect fleet metrics.

Can you find and fix them all?

Run the self-tests to see which ones fail:

[diagram or code sample]

You should see 3 failed tests. Each test corresponds to one bug:

Test | What it checks | Hint | Test 1 | Total 24h requests | Should sum requests, not average them | Test 2 | Degraded agent count | Should count degraded status, not active | Test 3 | Agents without durable memory | Should count none/session\_only, not cosmos\_db |  
Fix all 3 bugs, then re-run. When you see All passed!, you're done!

? Knowledge Check

question "Q1 (Multiple Choice): What is the role of an orchestrator agent in a Foundry multi-agent deployment?"

- A) It performs a specific domain task like order processing - B) It routes requests to specialist agents and manages conversation flow - C) It stores agent memory in Cosmos DB - D) It monitors agent health and restarts failed agents

??? success "? Reveal Answer" Correct: B) It routes requests to specialist agents and manages conversation flow

Orchestrator agents act as the "traffic controller" in a multi-agent system. They receive incoming requests, determine which specialist(s) should handle them, route the conversation accordingly, and manage the overall flow. Specialists handle the domain-specific work.

question "Q2 (Multiple Choice): Why is a disabled content filter a security risk for production agents?"

- A) It makes the agent slower - B) It allows the agent to generate harmful, biased, or policy-violating content - C) It prevents the agent from accessing external APIs - D) It increases token costs

??? success ""? Reveal Answer" Correct: B) It allows the agent to generate harmful, biased, or policy-violating content

Azure AI Content Safety filters detect and block harmful content (hate speech, violence, self-harm, sexual content). Disabling the filter means the agent can produce or respond to such content without guardrails ? a compliance and reputational risk in any production deployment.

question "Q3 (Run the Lab): What is the total number of requests across the entire fleet in the last 24 hours?"

Run the Step 3 analysis on ? foundry\_agents.csv and check the results.

??? success ""? Reveal Answer" 9,380 requests

Sum of all agent requests\_24h values: 1,250 + 890 + 2,100 + 560 + 3,200 + 780 + 420 + 180 = 9,380.

question "Q4 (Run the Lab): How many agents are in a degraded state?"

Run the Step 4a analysis to find out.

??? success ""? Reveal Answer" 1 agent

Only AnalyticsAgent (FA07) is in a degraded state, with an 8.5% error rate and 850ms average latency ? significantly worse than the other agents. This likely indicates a backend connectivity issue with its AI Search memory store.

question "Q5 (Run the Lab): How many agents have their content filter disabled?"

Run the Step 4c analysis to check content filter status.

??? success ""? Reveal Answer" 1 agent

Only LegacyBridge (FA08) has content\_filter=disabled. It's also the only agent deployed on a self-hosted VM rather than Container Apps, and has the highest error rate (12.0%) in the fleet. This agent needs immediate attention.

Summary

Topic | What You Learned | Foundry Agent Service | Managed platform for multi-agent orchestration and deployment | Agent Types | Orchestrators route; specialists execute domain tasks | Fleet Monitoring | Track requests, latency, error rates, and status per agent | Degraded Detection | Identify agents with elevated error rates or latency | Content Safety | All production agents should have content filters enabled | Memory Patterns | Cosmos DB for durable, AI Search for vector, session\_only for ephemeral |

Next Steps

Lab 034 ? Multi-Agent with Semantic Kernel (building the agents themselves) Lab 033 ? Agent Observability with Application Insights (deeper monitoring) Lab 030 ? Foundry Agent + MCP (connecting agents to external tools) Lab 075 ? Power BI Copilot (visualizing fleet data with AI-assisted dashboards)

Phase 3 modernization

warning "Verified 2026-10-27" Use the Microsoft Foundry name and GA Agent Service surfaces based on the Responses API. Evaluations are GA; hosted agents remain preview and must be labeled accordingly. Do not build new dependencies on portal Workflows scheduled for

retirement on 2026-12-01.

[diagram or code sample]

Foundry Agent Service GA announcement Foundry Agent Service Foundry evaluation Foundry Workflows retirement

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\_foundry.py

```
/usr/bin/env python3 """Bug-fix: Broken Foundry Analyzer - Lab 074. Fix 3 bugs.""" import
io, pandas as pd def total_requests(agents): # Bug #1: Review the aggregation method.
return int(agents["requests_24h"].mean()) def count_degraded(agents): # Bug #2: Review
which status should be counted. return (agents["status"] == "active").sum() def
agents_without_memory(agents): # Bug #3: Review which memory types indicate risk. return
(agents["memory_type"] == "cosmos_db").sum() def run_tests():
csv="agent_id,status,requests_24h,memory_type\nF1,active,1000,cosmos_db\nF2,degraded,500,ses
sion_only\nF3,active,800,none\nF4,active,1200,cosmos_db"
a=pd.read_csv(io.StringIO(csv));p,f=0,0 t=total_requests(a);exp=3500 if t==exp:print(f"OK
T1: {t}");p+=1 else:print(f"FAIL T1: {t} (exp {exp})");f+=1 d=count_degraded(a);exp=1 if
d==exp:print(f"OK T2: {d}");p+=1 else:print(f"FAIL T2: {d} (exp {exp})");f+=1
n=agents_without_memory(a);exp=2 if n==exp:print(f"OK T3: {n}");p+=1 else:print(f"FAIL T3:
{n} (exp {exp})");f+=1 print(f"\n{'All passed!' if f==0 else f'{f} failed'}") if
__name__=="__main__": print("Tests:\n");run_tests()
```

Supporting file: foundry\_agents.csv

```
agent_id,agent_name,agent_type,model,tools_count,memory_type,deployment,status,requests_24h,
avg_latency_ms,error_rate_pct,content_filter
FA01,ProductAdvisor,specialist,gpt-4o,5,cosmos_db,container_apps,active,1250,320,1.2,enabled
FA02,OrderProcessor,specialist,gpt-4o,8,cosmos_db,container_apps,active,890,450,2.1,enabled
FA03,SupportRouter,orchestrator,gpt-4o-mini,3,session_only,container_apps,active,2100,180,0.
5,enabled FA04,InventoryMonitor,specialist,gpt-4o-mini,4,ai_search,container_apps,active,560
,220,0.8,enabled FA05,Coordinator,orchestrator,gpt-4o,2,cosmos_db,container_apps,active,3200
,280,1.0,enabled FA06,QualityReviewer,specialist,gpt-4o,3,session_only,container_apps,active
,780,350,1.5,enabled FA07,AnalyticsAgent,specialist,gpt-4o,6,ai_search,container_apps,degrad
ed,420,850,8.5,enabled FA08,LegacyBridge,specialist,gpt-4o-mini,10,none,vm,active,180,1200,1
2.0,disabled
```