Microsoft

MICROSOFT REACTOR · SESSION 03 OF 03

Build a Multi-Agent Career Copilot

Résumé-to-job fit analysis with four specialised agents, orchestrated in VS Code.

Live | Hands-on | Recorded

SG

Shivam Goyal

Microsoft MVP (AI) · EY

WHAT YOU'LL LEAVE WITH

- A deployed 4-agent workflow
- WorkflowBuilder orchestration
- A live MCP tool integration

`aka.ms/FoundryToolkit-Lab`

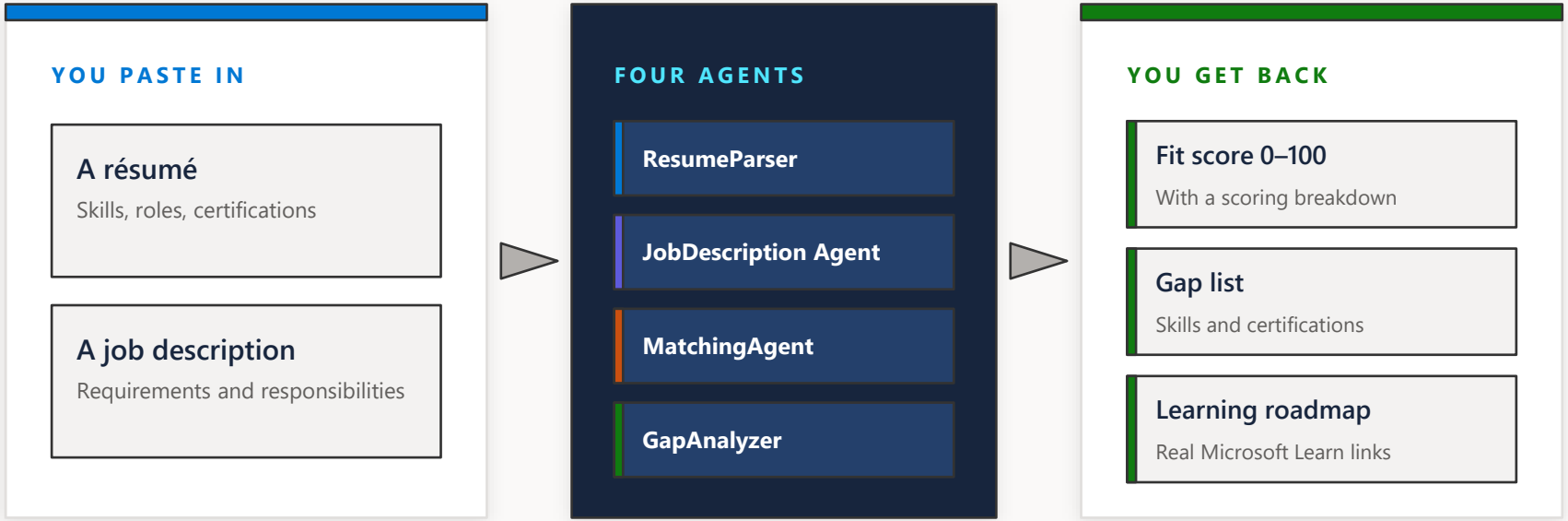
Where we're going

AGENDA

Recap	What sessions 01 and 02 gave us	01
Why multi-agent	Four specialists beat one generalist	02
The pattern	Sequential chain and content relay	03
The build	WorkflowBuilder, MCP, and the lab	04

Résumé → Job Fit Evaluator

WHAT YOU'LL BUILD TODAY



Before we start

PREREQUISITES

Carried over from Lab 01

- VS Code with the Microsoft Foundry extension
- Python 3.10 or later
- A deployed model — e.g. gpt-4.1-mini
- Foundry User role on the project
- New for Lab 02: the mcp package

Path A — Azure cloud

Full end-to-end deployment to Foundry Agent Service. Modules 00–09.

Path B — Foundry Local

No Azure subscription needed. Local testing only. Modules 00–05.

Where we left off

QUICK RECAP

SESSION 01

Toolkit intro

- What the Foundry Toolkit adds to VS Code
- Creating and running an agent locally
- Agent Inspector and the F5 debug loop

Junjie Li Senior PM, DevDiv · Microsoft

SESSION 02

Your first agent

- A single Python hosted agent, end to end
- Instructions, testing, safety boundaries
- Deployed and verified in the Playground

Bethany Jepchumba Cloud Advocate · Microsoft

Today: one agent becomes four — and they hand work to each other.

What changes in Lab 02

LAB 01 → LAB 02

WHAT YOU COMPARE	LAB 01 · SINGLE AGENT	LAB 02 · MULTI-AGENT
Agents	1	4, chained with WorkflowBuilder
Template	Basic — Agent Framework	Workflows — Agent Framework
Orchestration	Single conversational agent	Sequential pipeline, 3 edges
New package	—	`mcp`
New tool	—	`search_microsoft_learn_for_plan`

Everything you already know still applies — you're adding orchestration, not relearning the toolkit.

Why not just one agent?

SPECIALISATION BEATS GENERALISATION

One agent doing everything

Rushes the work
One pass over four jobs produces shallow output.

Hard to debug
When the answer is wrong, which part of the prompt failed?

Prompt bloat
One instruction block tries to cover four different jobs.

No reuse
You can't swap the scorer without touching everything else.

Four focused specialists

ResumeParser
One job: read the résumé accurately.

JobDescription Agent
One job: extract real requirements.

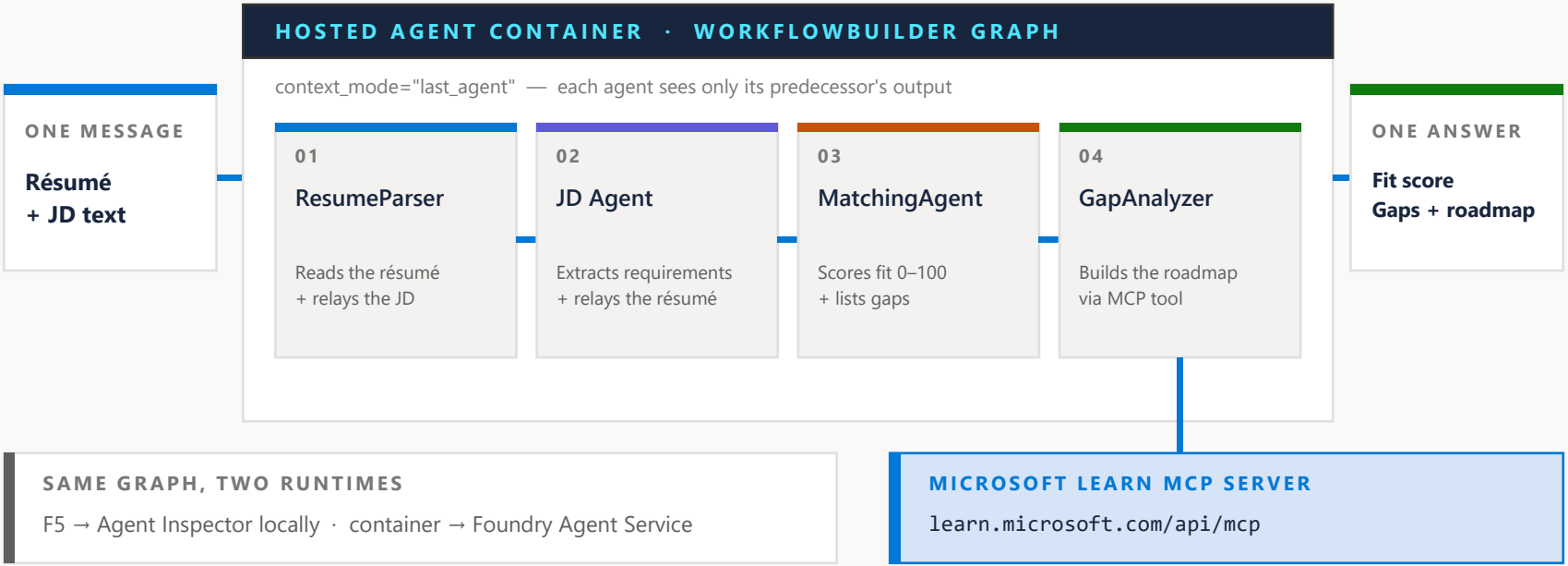
MatchingAgent
One job: score the fit honestly.

GapAnalyzer
One job: plan the path forward.

Smaller prompts, clearer failures, swappable parts.

The whole picture

END-TO-END ARCHITECTURE

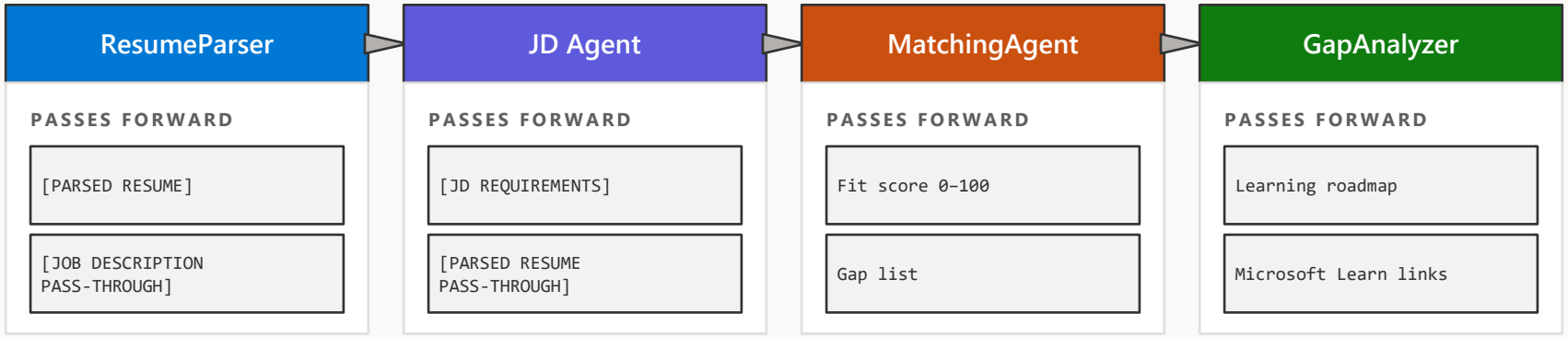


The pass-through pattern

CONTENT RELAY IN A SEQUENTIAL CHAIN

`context_mode="last_agent"`

The constraint: each agent sees only its direct predecessor's output. Anything a later agent needs must be relayed forward verbatim.



Why sequential, not fan-in? WorkflowBuilder uses OR-semantics — an executor fires as soon as any predecessor completes. Two incoming edges would run it twice.

Building the workflow

MAIN.PY · WORKFLOWBUILDER

PersonalCareerCopilot / main.py

```python
workflow_agent = (
    WorkflowBuilder(
        start_executor=resume_executor,
        output_executors=[gap_executor],
    )
    .add_edge(resume_executor, jd_executor)
    .add_edge(jd_executor, matching_executor)
    .add_edge(matching_executor, gap_executor)
    .build()
    .as_agent()
)
```

Each Agent is wrapped in an AgentExecutor with context_mode="last_agent".

READING THE GRAPH

start_executor
The only agent that receives the raw user message.

output_executors
Its response is what the caller gets back.

add_edge()
One call per step — three edges, four agents.

as_agent()
The whole graph is served as a single hosted agent.

The Microsoft Learn MCP tool

HOW THE ROADMAP GETS REAL LINKS

`search_microsoft_learn_for_plan`

GapAnalyzer's only tool. For each skill gap it queries the Microsoft Learn MCP server and returns official, citable modules — not invented URLs.

ENDPOINT

`https://learn.microsoft.com/api/mcp`

Wired with **`@tool`** + **`streamable_http_client`**

EXPECTED LOGS — DON'T PANIC

GET `learn.microsoft.com/api/mcp → 405`

Connection probe — normal.

POST `learn.microsoft.com/api/mcp → 200`

The actual tool call. This is the one that matters.

DELETE `learn.microsoft.com/api/mcp → 405`

Cleanup probe — normal.

Only worry if the POST returns an error.

From F5 to hosted agent

LOCAL → DEPLOYED

01
Trigger deploy
Deploy button in Agent Inspector, or the Command Palette

02
Build + push
Docker image built from your Dockerfile, pushed to ACR

03
Register version
agent.yaml metadata creates a hosted agent version

04
Container starts
Per-agent Entra identity resolved by DefaultAzureCredential

05
Serve requests
Same /responses endpoint as Lab 01, live in Playground

WHAT FOUNDRY SEES

One hosted agent

All four agents live in one container.

Same deployment surface as Lab 01.

EXPECT THIS TIMING

Image build 1–3 min

ACR push 30–60 s · start 30–120 s

Playground 1–2 min after Started.

DEFAULTS THAT WORK

CPU 0.25 · Memory 0.5Gi

I/O-bound — more CPU won't help.

Name: resume-job-fit-evaluator

Before you click deploy: Docker running · signed in to Azure in VS Code · Foundry User (Project Manager to deploy) at project scope.

LIVE DEMO · LAB 02 ROADMAP

Time to build.

Four agents, one workflow, one editor — follow along, or catch up after.

aka.ms/FoundryToolkit-Lab

01 **Understand the architecture**
WorkflowBuilder, agent roles, the graph

02 **Create the project**
Workflows template via the toolkit wizard

03 **Configure agents**
Four instruction sets, MCP tool, env vars

04 **Orchestration patterns**
Sequential chain, content relay, OR-semantics

05 **Test locally**
F5, Agent Inspector, three smoke tests

06 **Deploy to Foundry**
Container, ACR, hosted agent registration

07 **Verify in Playground**
VS Code and Foundry Portal

08 **Troubleshooting**
MCP errors, truncated output, versions

THAT'S A WRAP

You built a four-agent workflow.

One graph, four specialists, one MCP tool — running locally and deployed as a single hosted agent.

TAKE IT FURTHER

Swap the pattern
Sequential → concurrent fan-out

Add your own tool
Any MCP server, same @tool wiring

Evaluate it
Score the scorer before you ship

POINT YOUR PHONE · THREE THINGS WORTH KEEPING

01

The full workshop

Lab 01 + Lab 02.
50+ languages.
Finish modules 06–09.

aka.ms/FoundryToolkit-Lab

02

My GitHub

More agent samples.
Open an issue if you get stuck after today.

github.com/ShivamGoyal03

03

Say hello

Blog, talks, contact.
Tell me what you build — I want to see it.

shivamgoyal03.github.io

Star the repo · ship something this week · tag me when you do.

Questions? Let's take them now.