What is an AI Agent?

The Evolution of AI Assistants

From simple Q&A to intelligent agents:

Generation	Capability	Limitation
Chatbots	Pre-scripted responses	No real understanding
LLMs	Natural language understanding	No access to external data
RAG	Retrieval + Generation	Single retrieval strategy
Agents	Reasoning + Tools + Action	Full autonomy

What is an Agent?

An agent is an LLM with access to **tools**.

Component	What It Does
Reasoning	LLM analyzes the question and decides what to do
Tools	Capabilities the agent can call (search, query, etc.)
Action	Executes the selected tool(s) and returns results

How Agents Use Tools

Agents take action by calling tools—functions that get information or perform tasks.

How tool selection works:

Question	Agent Reasoning	Tool Selected
"How many companies?"	Needs a count	`query_database`
"What risks does Apple face?"	Needs content search	`search_content`
"What's in the database?"	Needs structure	`get_graph_schema`

The agent matches the question to the best tool description.

The ReAct Pattern

Agents follow **ReAct** (Reasoning + Acting):

```
1. Receive question: "How many risk factors does Apple face?"
2. Reason: "This asks for a count"
3. Act: Call Database Query Tool
4. Observe: Result = 45
5. Respond: "Apple faces 45 risk factors."
```

For complex questions, the agent may loop through multiple cycles.

Multi-Tool Example

Question: "What are Apple's main risks and which investors are affected?"

Agent process:

1. **Reason:** Need risk content AND investor relationships
2. **Act:** Call Semantic Search for Apple's risks
3. **Observe:** Risk descriptions
4. **Reason:** Now need investors
5. **Act:** Call Database Query for Apple's investors
6. **Observe:** Investor list
7. **Respond:** Combine both into comprehensive answer

Why Agents Matter

Without agents:

- Build separate interfaces for each capability
- Force users to choose which tool to use
- Complex user experience

With agents:

- Users ask natural questions
- System figures out how to answer
- Conversational, intuitive experience

Agents vs Traditional Software

Aspect	Traditional Software	AI Agents
Input	Structured commands	Natural language
Logic	Pre-programmed rules	LLM reasoning
Flexibility	Fixed workflows	Dynamic tool selection
Adaptation	Code changes needed	Learns from context

Summary

In this lesson, you learned:

- **Agents** have four components: Perception, Reasoning, Action, Response
- **Tools** are capabilities agents use to take action
- **Selection** happens through semantic matching to tool descriptions
- **ReAct pattern**: Reason → Act → Observe → Respond
- **Result**: Users ask naturally; agents figure out how to answer

Next: Learn about the Model Context Protocol (MCP).

What is MCP?

Model Context Protocol

The Problem: Tool Integration

Every AI framework has its own way of defining tools:

- **OpenAI**: Function calling with JSON schemas
- **Anthropic**: Tool use with specific formats
- **LangChain**: Tool wrappers and chains
- **Custom Agents**: Proprietary integrations

Result: Building tools once means rebuilding for each framework.

The Solution: MCP

Model Context Protocol (MCP) is an open standard for connecting AI models to external tools and data sources.

Think of it as **USB for AI tools**:

USB	MCP
Standard port for devices	Standard protocol for AI tools
Plug any device into any computer	Plug any tool into any AI model
One cable, many devices	One server, many clients

MCP vs Standard APIs: Who Is It Built For?

The biggest difference is **who the server is built for**.

Standard API (REST/GraphQL)	MCP Server
Built for Programmers	**Built for AI Models**
Assumes you've read the manual	Assumes the AI knows nothing
You must know endpoints exist	Server tells AI what's available
`/users`, `/invoices`, etc.	"Here are 5 things I can do"

"Read the Manual" vs "Ask Me"

Standard API: Passive. If you don't know the magic words, you get an error.

MCP Server: Active and self-documenting. Discovery is built-in.

```
AI: "Hello, what can you do?"

MCP Server: "I can read these 3 files (Resources), and I have
             a tool called query_graph that requires a cypher
             parameter (Tool)."

AI: "Great, I'll use that."
```

The AI doesn't need documentation—it asks the server directly.

MCP Architecture

How an Agent fits in:

Component	Role	Example
Agent	The AI client hosted in Foundry	Foundry Agent
Tools	Capabilities the agent can use	MCP Servers
MCP Server	Provides tools via the protocol	Neo4j MCP Server

The Agent has tools—MCP servers are one way to provide those tools.

What MCP Servers Provide

MCP servers can expose:

- **Tools** - Functions the AI can call (query database, search, etc.)
- **Resources** - Data the AI can read (files, database schemas)
- **Prompts** - Pre-defined prompt templates

Benefits of MCP

Benefit	Description
Standardization	One protocol for all tools
Reusability	Build once, use everywhere
Security	Controlled access to resources
Composability	Combine multiple servers
Ecosystem	Growing library of servers

Neo4j MCP Server

The **Neo4j MCP Server** is an official MCP implementation that gives AI agents the ability to:

- **Explore** your graph schema
- **Read** data using Cypher queries
- **Write** data (only when explicitly enabled)

It's the bridge between natural language questions and graph database answers.

Core Tools Provided

The Neo4j MCP Server exposes three main tools:

Tool	Purpose	Default
get_neo4j_schema	Retrieve graph structure	Always enabled
read_neo4j_cypher	Execute read queries	Always enabled
write_neo4j_cypher	Execute write queries	**Disabled by default**

Tool 1: Get Schema

```
get_neo4j_schema
```

Returns the structure of your knowledge graph:

- **Node labels** (e.g., Company, RiskFactor, Filing)
- **Relationship types** (e.g., HAS_RISK, FILED)
- **Properties** on nodes and relationships

Why it matters: The AI needs to understand your data model before it can write correct Cypher queries.

Tool 2: Read Cypher

```
read_neo4j_cypher
```

Executes **read-only** Cypher queries against the database.

Example flow:

```
User: "What risks does Apple face?"

AI uses read_neo4j_cypher:
MATCH (c:Company {name: 'Apple Inc'})-[:HAS_RISK]->(r:RiskFactor)
RETURN r.description

Result: [List of risk factors from SEC filings]
```

Safe by design: Cannot modify any data.

Tool 3: Write Cypher

```
write_neo4j_cypher
```

Executes queries that **create, update, or delete** data.

Important: This tool is **disabled by default** for safety.

To enable, you must explicitly configure:

```
{
  "neo4j_write_enabled": true
}
```

In This Lab

We'll use the Neo4j MCP Server with:

- **get_neo4j_schema** - To help the agent understand our SEC filings graph
- **read_neo4j_cypher** - To answer questions about companies and risks

Write access is not needed - we're analyzing existing data.

Summary

- **MCP** is an open standard for AI tool integration
- Works like **USB for AI** - one protocol, many tools
- **Neo4j MCP Server** provides three tools for graph access
- **Secure-by-default** design prevents accidental data modification
- **Microsoft Foundry** supports MCP out of the box

Next: Learn about Microsoft Foundry and deploy your agent.

Microsoft Foundry

Microsoft's AI App and Agent Factory

What is Microsoft Foundry?

Microsoft describes it as:

> "An interoperable AI platform that enables developers to build faster and smarter, while organizations gain fleetwide security and governance in a unified portal."

Key concept: An "AI app and agent factory" for the enterprise.

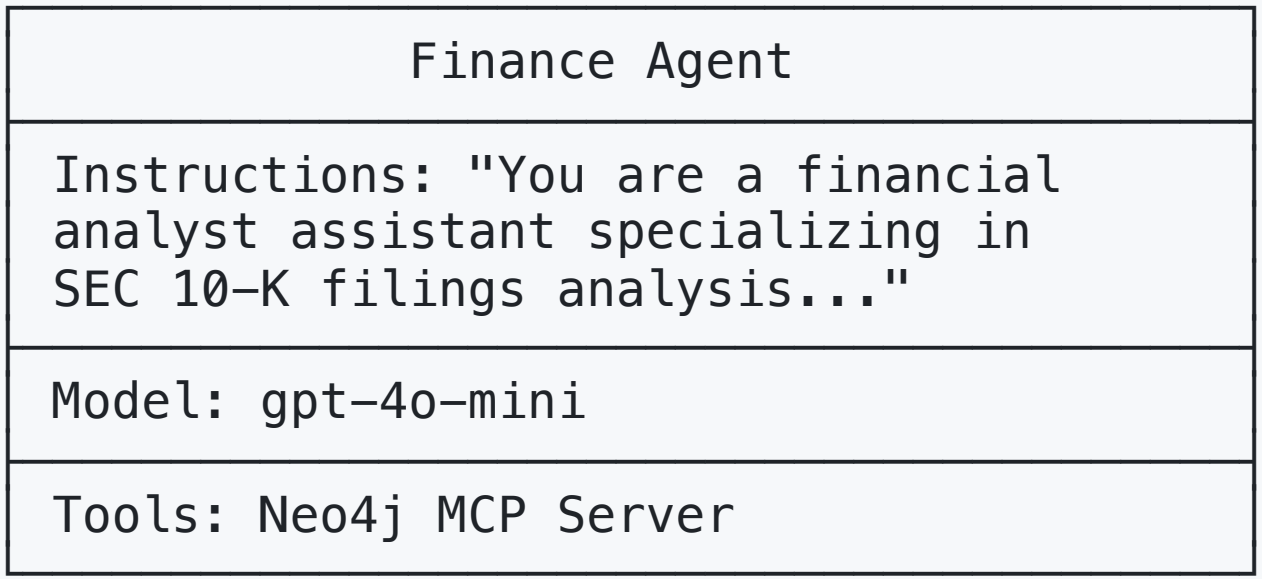
Key Foundry Components

Component	Purpose
Foundry Models	Access to 11,000+ AI models
Model Router	Auto-selects best model per task
Foundry Agent Service	Build autonomous, context-aware agents
Foundry IQ	Next-gen RAG for knowledge grounding
MCP Tool Catalogue	Unified tool discovery and management

Agents in Foundry

Foundry agents combine:

- **Instructions** - What the agent should do
- **Model** - Which LLM powers reasoning (or Model Router)
- **Tools** - MCP servers and built-in capabilities



Enterprise Governance

Foundry Control Plane provides:

Capability	Description
Observability	Monitor all agents fleet-wide
Compliance	Enforce policies across deployments
Security	Microsoft Defender + Entra ID integration
Content Safety	Built-in threat detection

Enterprise-ready from day one.

What We'll Build

A **Finance Agent** that:

Capability	How
Understands SEC filings	Knowledge graph context
Queries company data	Neo4j MCP Server
Answers risk questions	Graph traversal
Analyzes relationships	Connected data

User: "What risks does Apple face?"

Agent: Queries Neo4j, returns risk factors

The Lab Flow

1. **Create** Foundry resource and project
2. **Deploy** gpt-4o-mini model
3. **Create** finance-agent
4. **Add** Neo4j MCP tool
5. **Test** with SEC filing questions
6. **Publish** your agent