Day 2: AI Agents & Agentic Software

Complete Training Guide for .NET Developers

Building AI Agents | Agent Architectures | Real-world Implementation

Table of Contents

- **AI Agents vs. Agentic Software**
- **Agent Architectures** (ReAct, Plan-and-Execute, Multi-Agent Swarms)
- **Agent Toolbox** & Tool-Calling Patterns
- **Memory & Context** Management
- **Single-Agent** Implementation
- **Multi-Agent Workflows** & MCP
- **Agent Evaluation** Metrics
- **Real-World Use Cases**
- **Integration with n8n**

AI Agents vs. Agentic Software

AI Agent: Single autonomous entity

- Perceives environment
- Makes decisions
- Takes actions to achieve goals
- **Example:** Chatbot that books meetings

Agentic Software: Complete application

- Multiple autonomous components
- Coordinated decision-making
- **Example:** Full CRM with auto-response, scheduling, analysis

Agent Architectures Overview

1. ReAct (Reasoning + Acting)

- Think → Act → Observe loop
- Dynamic and adaptive

2. Plan-and-Execute

- Plan first, then execute
- Structured and efficient

3. Multi-Agent Swarms

- Specialized agents working together
- Parallel processing

ReAct Architecture

Core Concept: Alternates thinking and acting

Process Loop:

- THINK: Reason about what to do next
- ACT: Execute an action or use a tool
- OBSERVE: See the results
- Repeat until goal achieved

Best For:

✓ Dynamic, unpredictable situations
✓ Exploratory tasks (debugging, research)
✓ Tasks requiring adaptation

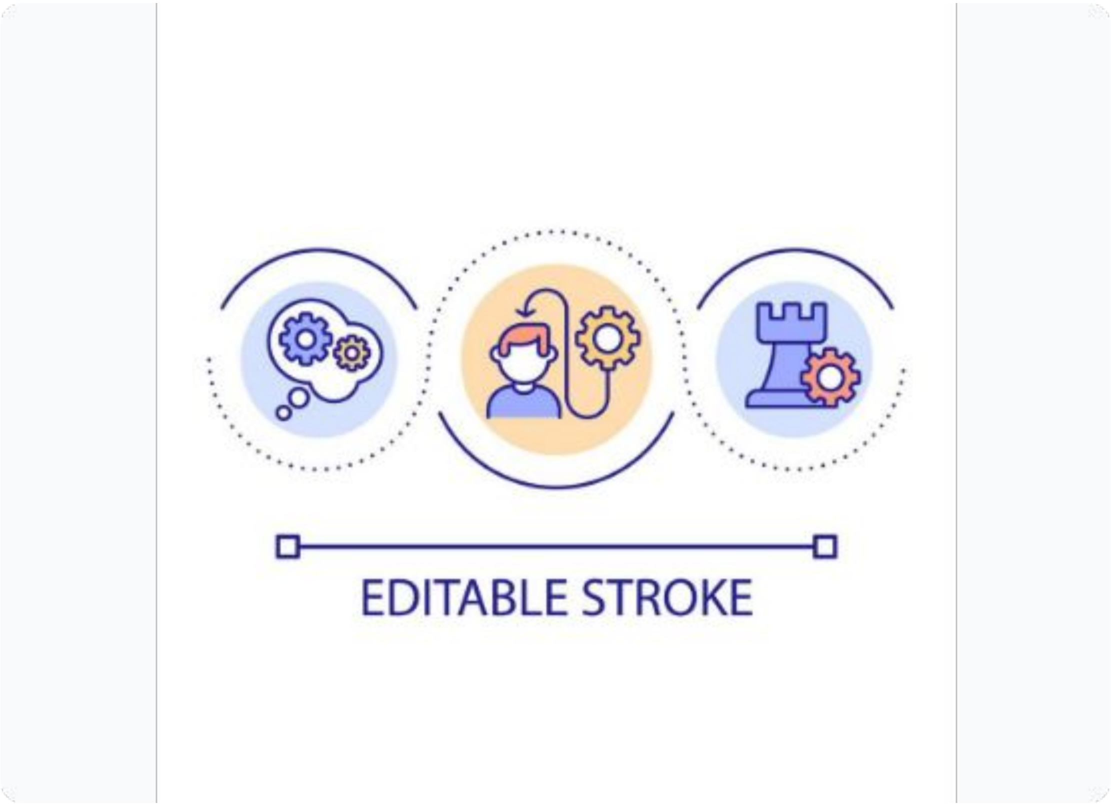
ReAct – Real-World Example

User: 'What's the weather in Paris and should I bring an umbrella?'

Agent Process:

1. **THOUGHT:** Need current weather data
2. **ACTION:** Calls GetWeather('Paris')
3. **OBSERVATION:** '22°C, Cloudy, 70% rain chance'
4. **THOUGHT:** High rain chance, recommend umbrella
5. **OUTPUT:** 'Yes, bring an umbrella!'

Strengths: Flexible, transparent, handles uncertainty
Weaknesses: Can be inefficient, no long-term planning



Plan-and-Execute Architecture

Core Concept: Create complete plan first, then execute

Process:

1. Understand the goal
2. Create comprehensive plan
3. Execute step 1
4. Execute step 2...
5. Continue through all steps
6. Verify completion

Best For:

✓ Complex, multi-step tasks
✓ Clear dependencies between steps
✓ When efficiency matters

Plan-and-Execute - Example

Task: 'Organize team meeting next Tuesday'

Planning Phase:
Step 1: Check team availability
Step 2: Find common time slot
Step 3: Book meeting room
Step 4: Create agenda
Step 5: Send invites
Step 6: Send follow-up email

Execution Phase: Execute each step sequentially

Strengths: Efficient, organized, predictable
Weaknesses: Rigid, hard to adapt mid-execution

Multi-Agent Swarms

Core Concept: Multiple specialized agents collaborate

Coordination Patterns:

- **Sequential (Pipeline):** Agent 1 → Agent 2 → Agent 3
- **Hierarchical:** Manager coordinates workers
- **Collaborative:** Agents communicate peer-to-peer
- **Competitive:** Multiple solutions, best wins

Best For:

✓ Complex problems needing diverse expertise
✓ Parallel processing for speed
✓ Quality through specialization

Multi-Agent Example: Blog Creation

Task: 'Write comprehensive blog about AI in healthcare'

Agent 1 - Research

- Searches medical journals
- Gathers statistics

Agent 2 - Writer

- Creates engaging narrative
- Structures content

Agent 3 - Editor

- Reviews and polishes
- SEO optimization

Agent 4 - Fact-Checker

- Validates claims
- Ensures accuracy

Choosing the Right Architecture

ReAct

- Exploring/researching
- Unclear path to solution
- **Example:** 'Find best hotel in Paris'

Plan-and-Execute

- Complex multi-step process
- Clear dependencies
- **Example:** 'Migrate customer data'

Multi-Agent Swarms

- Diverse expertise needed
- Quality and specialization matter
- **Example:** 'Analyze legal contract'

Agent Toolbox

Tool Categories:

- Database Operations
- Email & Communication
- External APIs
- File Operations
- Calculations & Analysis

Tool-Calling Pattern:

1. Agent receives task
2. Determines which tool to use
3. Calls tool with parameters
4. Processes result
5. Continues or completes task

Tool-Calling in .NET

Microsoft Semantic Kernel

1. Define functions with **[KernelFunction]** attribute
2. Add descriptions for discoverability
3. Register plugins with kernel
4. Enable **ToolCallBehavior.AutoInvokeKernelFunctions**
5. Agent automatically selects and calls tools

Key Libraries:

- Microsoft.SemanticKernel
- LangChain (optional)
- AutoGen (Microsoft)
- Custom implementations

Memory & Context Management

1. Conversation Memory

- **Short-term**
- ChatHistory object
- Recent conversation context
- Used for maintaining dialogue flow

2. Vector Memory

- **Long-term**
- Embeddings-based search
- Store and retrieve by meaning
- Knowledge base, documentation

3. Episodic Memory

- **Session-based**
- Track actions and outcomes
- Session-specific context
- Learning from past interactions

Conversation Memory

Purpose: Maintain dialogue context

Implementation:

- **ChatHistory** stores all messages
- System, User, and Assistant messages
- Passed to each API call

Use Cases:

- Multi-turn conversations
- Follow-up questions
- Context-dependent responses

Limitation: Limited by token window
Solution: Summarization or pruning old messages

Technologies:

- OpenAI Embeddings
- Vector stores: Pinecone, Qdrant, Chroma

Vector Memory

Purpose: Long-term knowledge storage

How it Works:

1. Convert text to embeddings (vectors)
2. Store in vector database
3. Search by semantic similarity
4. Retrieve relevant information

Use Cases:

- Product catalogs
- Documentation search
- Customer history
- Knowledge bases

Episodic Memory

Purpose: Track session-specific interactions

What it Stores:

- Actions taken
- Results observed
- Timestamps
- Context at each step

Use Cases:

- Learning from mistakes
- Session summaries
- Debugging agent behavior
- Performance analysis

Example: Track all customer support actions in a session for quality review

Single-Agent Implementation

Complete Agent Components:

1. **Kernel:** AI model connection
2. **Plugins:** Tools/functions
3. **Memory:** Conversation history
4. **Execution Settings:** Behavior configuration
5. **Chat Service:** Message handling

Flow:
User Input → Agent Reasoning → Tool Selection → Tool Execution → Result Processing → Response

Diagram of a single AI agent processing input and output

Frameworks:

- **Microsoft Semantic Kernel (recommended)**
- LangChain
- AutoGen

Multi-Agent Workflows

1. Orchestrator Pattern

- Central coordinator manages agents
- Routes tasks to specialists

2. Message Passing

- Agents communicate directly
- Share context and results

3. Shared State

- Common data store
- All agents read/write

4. Event-Driven

- Agents react to events
- Pub/sub pattern

Model Context Protocol (MCP)

Purpose: Standardize agent communication

Key Concepts:

- Standardized message format
- Context sharing
- Tool discovery
- State management

Benefits:

✓ Interoperability
✓ Easier coordination
✓ Consistent error handling
✓ Scalable architecture

Agent Evaluation Metrics

1. Task Success Rate

- Did agent complete the task?
- Measured: Yes/No or %

2. Faithfulness

- Did agent follow instructions?
- Measured: 0-100% score

3. Latency

- How long did it take?
- Measured: Seconds/milliseconds

4. Cost

- API usage & token consumption
- Measured: $ per task

Measuring Task Success

1. Rule-Based Validation

- Check if expected outcome occurred
- **Example:** Email sent = success

2. LLM-as-Judge

- Use another AI to evaluate
- Compare task vs. result

3. Human Evaluation

- Manual review of results
- Gold standard but expensive

4. Unit Tests

- Automated test suites
- Best for deterministic tasks

Faithfulness & Cost Tracking

Faithfulness

- Score how well agent followed instructions
- Did it use approved tools only?
- Did it stay on topic?
- **Evaluation:** LLM-based scoring (0-100)

Cost Tracking

- Token counting: input + output tokens
- Model pricing: GPT-4 vs GPT-3.5
- Tool calls: API costs
- **Formula:** Total Cost = (Tokens / 1000) × Price

Success Metrics:

- 70% tickets auto-resolved
- 2-minute average resolution time

Use Case 1: Customer Support

Scenario: Automated Support Ticket System

Agent Capabilities:

- Classify tickets (billing/technical/general)
- Search knowledge base
- Resolve common issues automatically
- Escalate complex issues to humans

Architecture: ReAct

- Adapts based on customer responses
- Asks clarifying questions

Use Case 2: Sales Lead Qualification

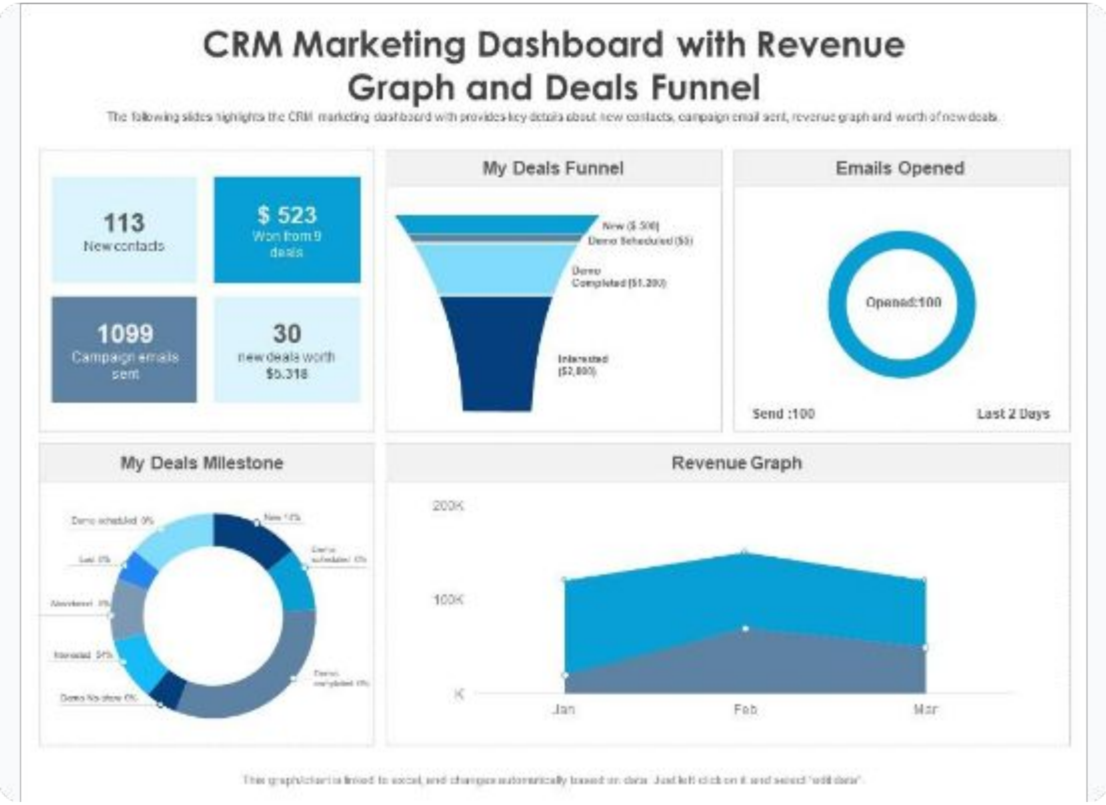
Scenario: Automated Lead Scoring & Outreach

Agent Capabilities:

- Research leads (LinkedIn, company website)
- Score lead quality (0-100)
- Generate personalized outreach
- Update CRM automatically

Architecture: Plan-and-Execute
- Systematic research process

Success Metrics:
- 3x increase in qualified leads
- 50% reduction in sales team time



CRM Marketing Dashboard with Revenue Graph and Deals Funnel

The following slides highlights the CRM marketing dashboard with provides key details about new contacts, campaign email sent, revenue graph and worth of new deals.

This graph/chart is linked to excel, and changes automatically based on data. Just left click on it and select "edit data".

Use Case 3: Code Review

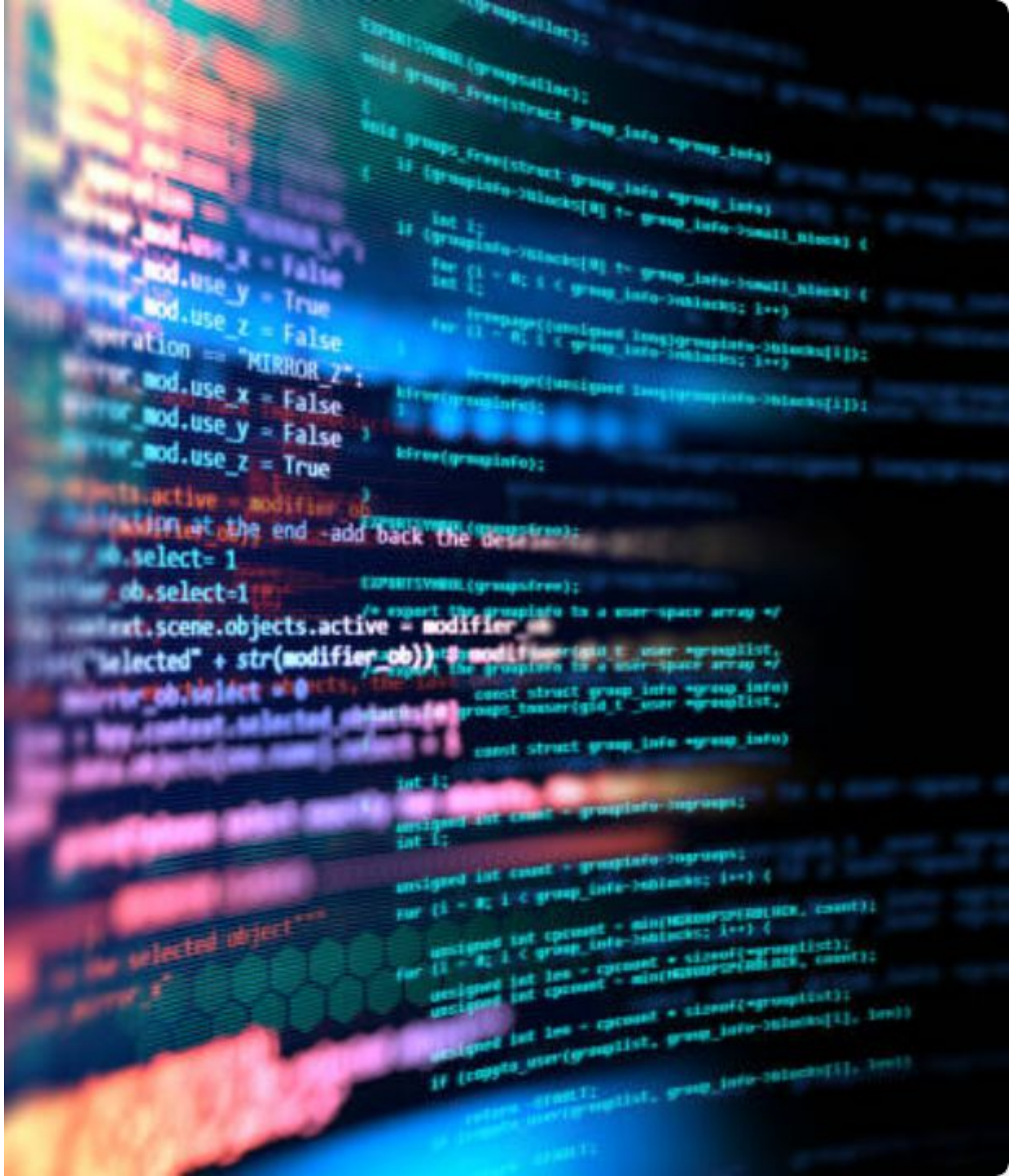
Scenario: Automated Code Review Assistant

Multi-Agent Swarm:

- **Security Agent:** Checks for vulnerabilities
- **Performance Agent:** Analyzes efficiency
- **Style Agent:** Ensures coding standards
- **Testing Agent:** Generates unit tests

Workflow: Parallel analysis → Consolidated report

Success Metrics:
• 40% faster code reviews
• 60% fewer bugs in production



Use Case 4: E-commerce

Scenario: Intelligent Order Processing

Multi-Agent System:

- **Inventory Agent:** Check stock & reserve
- **Payment Agent:** Process payment securely
- **Fraud Agent:** Risk assessment
- **Shipping Agent:** Calculate & schedule
- **Notification Agent:** Customer updates

Coordination: Event-driven with shared order context

Benefits: Parallel processing, Resilient, Scalable

Integration with n8n

What is n8n?

Visual workflow automation tool (like Zapier).

Integration Pattern:

1. Expose .NET agent via REST API
2. Create n8n workflow
3. Add HTTP Request node to call agent
4. Process agent response
5. Trigger downstream actions

Example: Email Received → n8n → Call Agent API → n8n Updates CRM → Send Reply

n8n logo with workflow automation graphics

n8n Workflow Examples

1. Customer Support

- **Trigger:** New support email
- **Agent:** Classifies and responds
- **Action:** Updates ticket system

2. Content Generation

- **Trigger:** Scheduled daily
- **Agent:** Generates social posts
- **Action:** Posts to platforms

3. Data Processing

- **Trigger:** File uploaded
- **Agent:** Analyzes and extracts insights
- **Action:** Saves to database

Best Practices

- ✓ **Start Simple:** Begin with single-agent ReAct. Add complexity only when needed.
- ✓ **Monitor & Evaluate:** Track all four metrics (Success, Faithfulness, Latency, Cost) continuously.
- ✓ **Iterative Improvement:** Review agent decisions and refine prompts and tools.
- ✓ **Handle Errors Gracefully:** Always use try-catch and provide fallback responses.
- ✓ **Security First:** Validate all tool inputs and limit agent permissions.

Common Pitfalls to Avoid

- 🚫 **Over-engineering:** Don't use swarms for simple tasks.
- 🚫 **Poor Tool Design:** Tools should be atomic and focused. Provide clear descriptions.
- 🚫 **Ignoring Context Limits:** Token windows are finite. Implement memory management.
- 🚫 **No Error Handling:** Agent calls can fail. Always plan for failures.
- 🚫 **Skipping Evaluation:** You can't improve what you don't measure.

Technology Stack Summary

Core Framework:

- **Microsoft Semantic Kernel ⭐ (recommended)**
- Alternative: LangChain

AI Models:

- OpenAI (GPT-4, GPT-3.5)
- Azure OpenAI Service
- Anthropic Claude (via API)

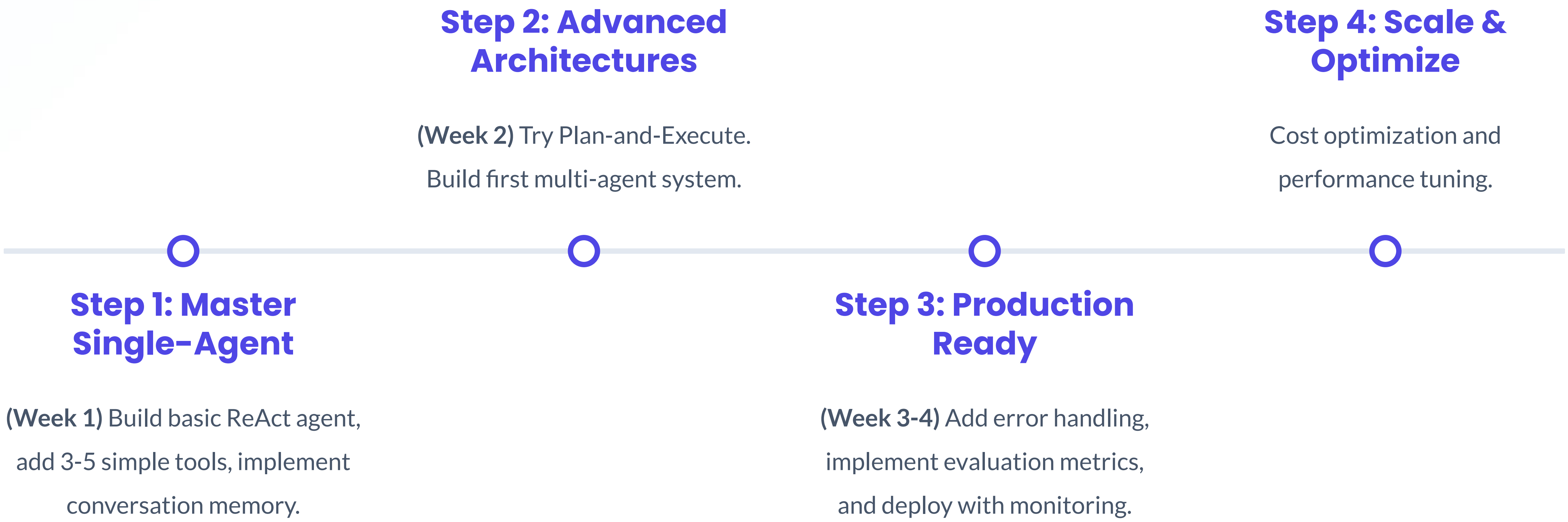
Vector Databases:

- Pinecone, Qdrant, Chroma

Orchestration:

- n8n (no-code)
- Custom ASP.NET Core APIs

Learning Path



Step 1: Master Single-Agent

(Week 1) Build basic ReAct agent, add 3-5 simple tools, implement conversation memory.

Step 2: Advanced Architectures

(Week 2) Try Plan-and-Execute. Build first multi-agent system.

Step 3: Production Ready

(Week 3-4) Add error handling, implement evaluation metrics, and deploy with monitoring.

Step 4: Scale & Optimize

Cost optimization and performance tuning.

Resources & Next Steps

Official Documentation:

- Microsoft Semantic Kernel Docs
- OpenAI API Documentation

Sample Projects:

- GitHub: Semantic Kernel samples
- Microsoft Learn modules

Community:

- Semantic Kernel Discord
- r/LocalLLaMA
- AI Engineering subreddit

Practice:

- Build a customer service bot
- Create a research assistant

Key Takeaways

- ✅ AI Agents = Autonomous systems that reason and act
- ✅ Three architectures: ReAct, Plan-and-Execute, Multi-Agent
- ✅ Tools = Functions your agent can call
- ✅ Memory = Conversation, Vector, Episodic
- ✅ Semantic Kernel = Best framework for .NET
- ✅ Evaluate: Success, Faithfulness, Latency, Cost
- ✅ Start simple, iterate, measure everything

🚀 Now go build something amazing!

Questions?

Review this presentation

Practice with the code examples

Build your first agent

The best way to learn is by building!

Happy coding! 💻🤖

FIND OUT MORE

Important Links:

- Autogen
- Agent Builder
- Agent Builder video
- n8n
- widget
- Agents Types
- Chatkit

TRY IT!

Lab

GET STARTED

Today's Task:

Build automated workflow using n8n (at least 3 nodes) to be requested as api and communicating with anyAI LLM model

integrate this workflow with small mvc application using HTTPClient

OR:

build automated RAG workflow for both (store embeddings and retrieval)