Your APIs Are Not Ready for AI (Yet): A Lifecycle-Based Readiness Guide

KNITECon 2025

Erik Wilde
Head of Enterprise Strategy, Jentic

November 12, 2025

Erik Wilde

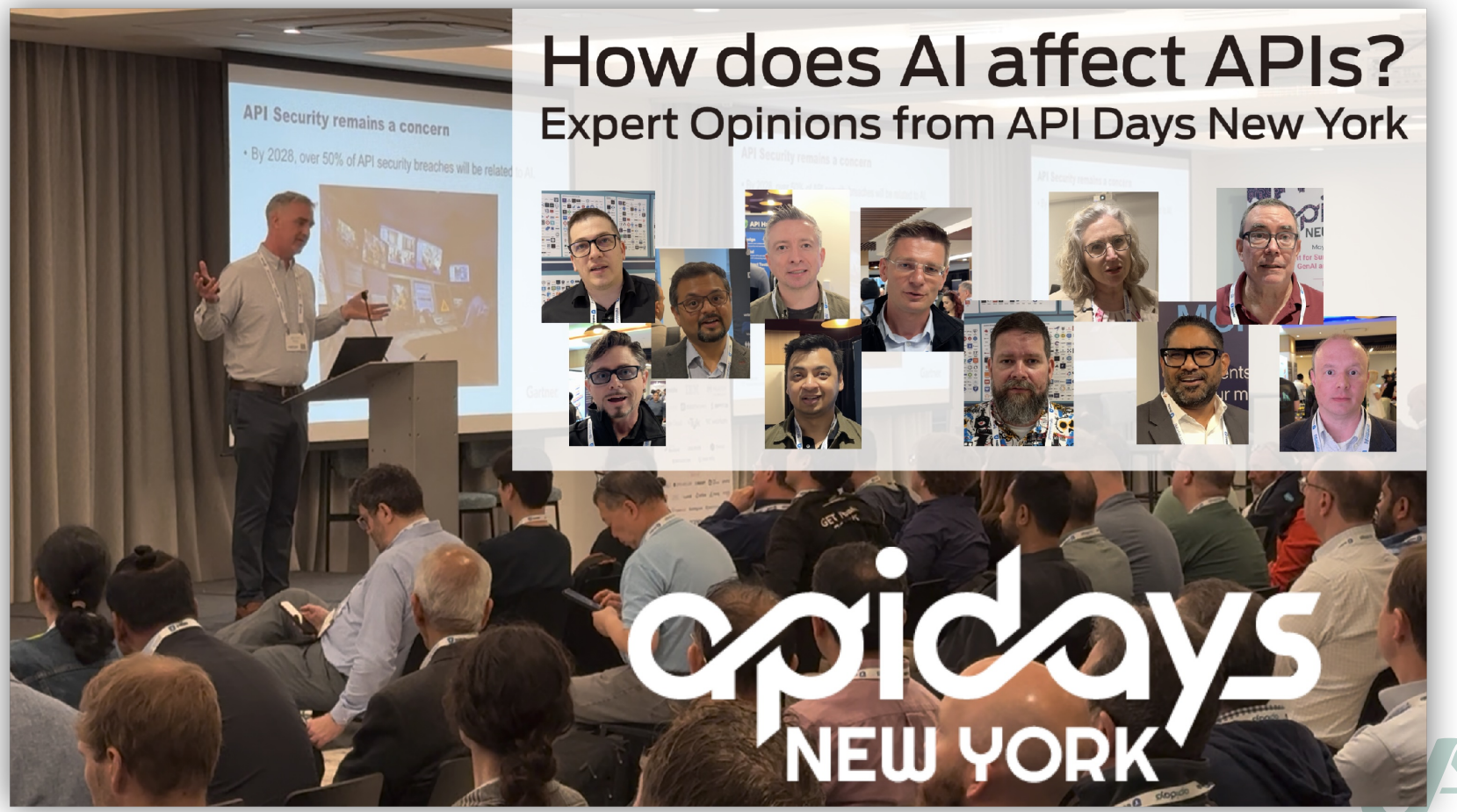
- Head of Enterprise Strategy at Jentic
- Ambassador for the OpenAPI Initiative (OAI)
- LinkedIn linkedin.com/in/erikwilde
- YouTube youtube.com/ErikWilde

Outline

1. APIs and AI: It's complicated [3]
2. Model Context Protocol (MCP) [8]
3. Supporting AI with APIs at Scale [13]
4. Conclusions [2]

Jentic

(How) does AI affect APIs?



Jentic

Bridging the Gap Between API Land and AI Land

- Most API landscapes are not (really) AI-ready
- Descriptions and designs can make AI difficult
- “Letting AI agents loose” oftentimes is not an option
  - *Limit blast radius:* Only provide access through a centralized access point
  - *Zero tolerance:* Never allow access to production APIs
- Use AI *as much as necessary* but *as little as possible*

Jentic

What AI Needs

- Access to data and capabilities
  - Accessing *data* to retrieve context
  - Accessing *capabilities* to take action
- *Model Context Protocol (MCP)* allows LLMs to “reach outside”
  - Great for initial experiments and “LLM-enabling” anything
  - Problems with scaling for more than a handful of MCP servers

Jentic

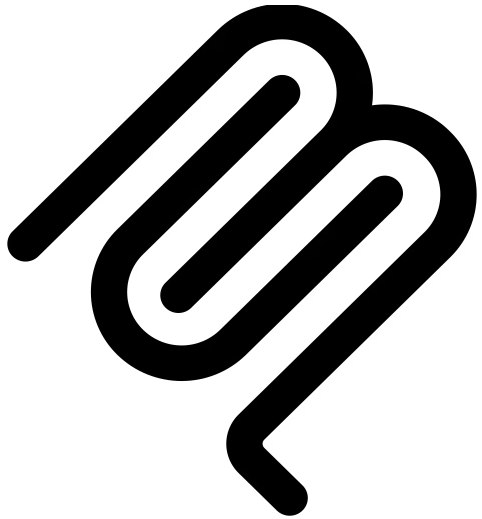
Outline

1. APIs and AI: It's complicated [3]
2. Model Context Protocol (MCP) [8]
3. Supporting AI with APIs at Scale [13]
4. Conclusions [2]

Jentic

Model Context Protocol (MCP)

- MCP is the current “standard SDK” for LLM consumers
- The initial focus was on *additional and extended context*
  - *Additional context* by providing access to the environment
  - *Extended context* by allowing MCP servers to externally manage context
- Many mapping tools generate MCP servers from OpenAPI descriptions
- *What* to expose and *how to design it* is out of scope



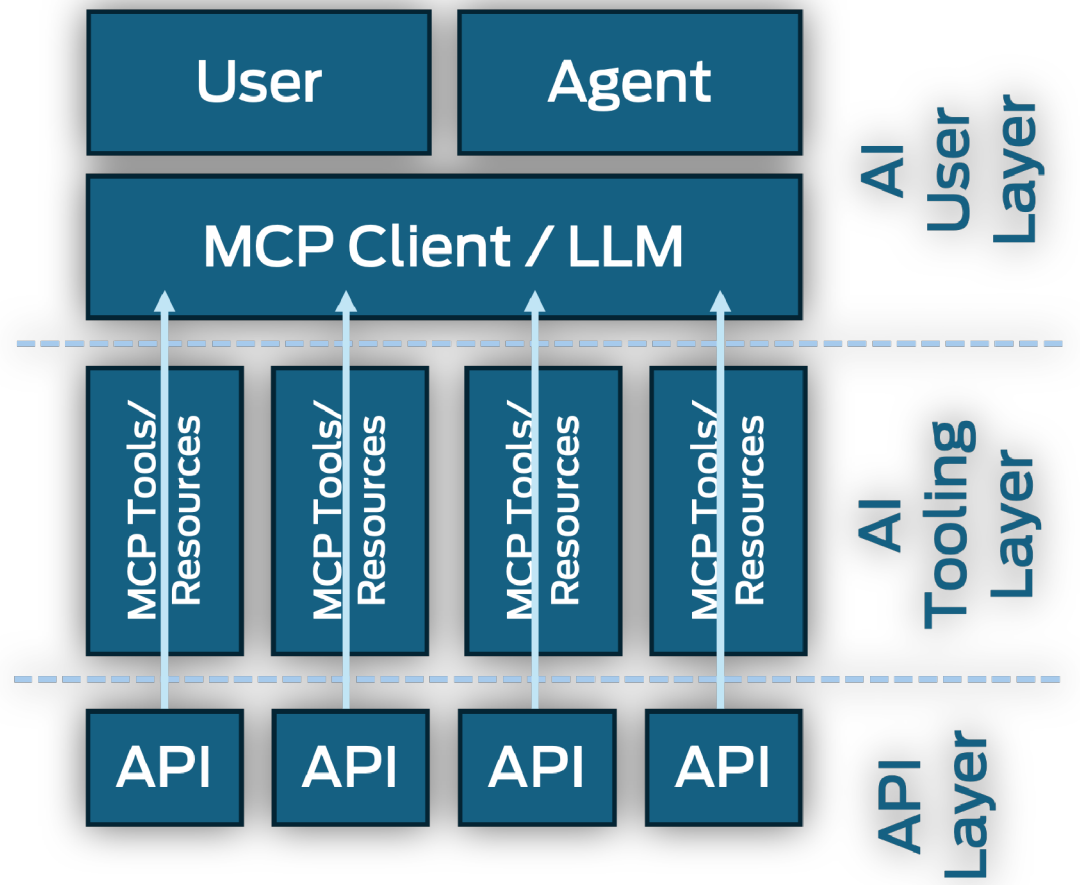
Jentic

MCP all the Things: Using MCP in 30min

1. Generate MCP from OpenAPI
2. Publish/Install MCP servers for all APIs
3. Consumers can access *all MCP tools*

Jentic

MCP all the Things



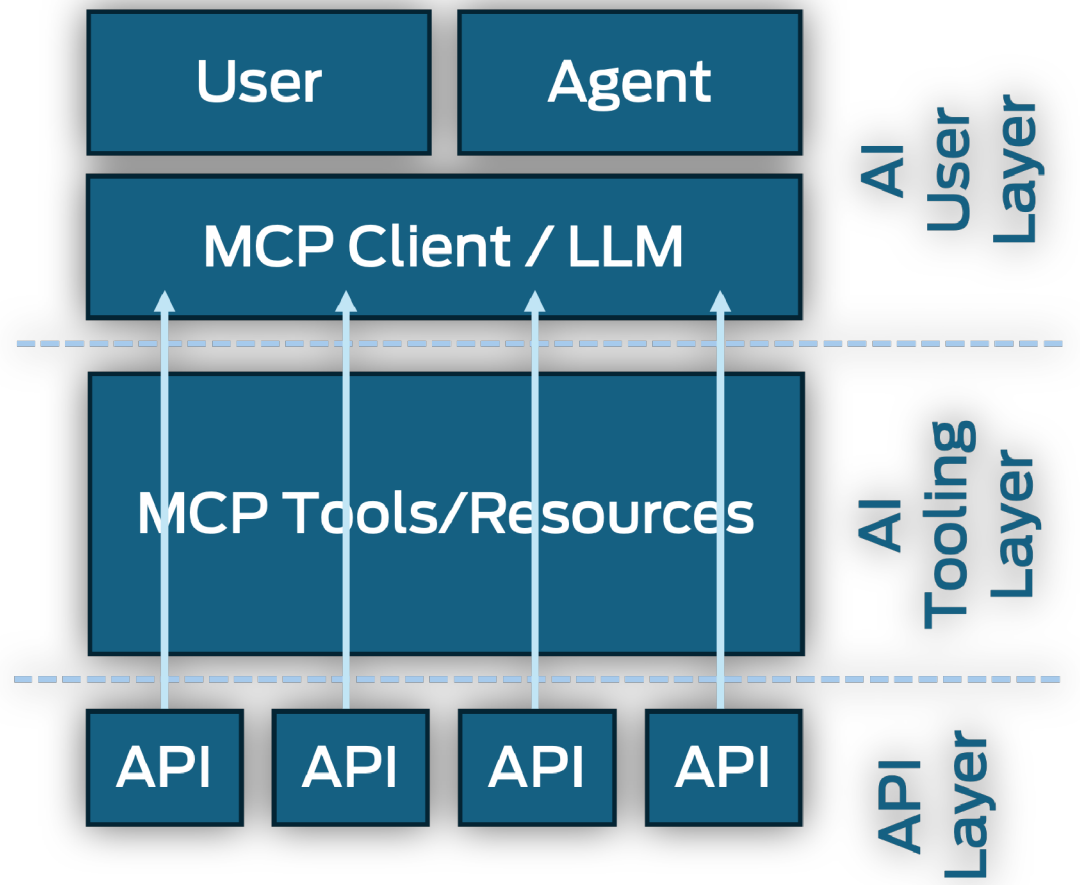
Jentic

MCP Lego Kit: Manage Context Size

1. Select required API capabilities
2. Aggregate all tools in one MCP server
3. Consumers can access *all required MCP tools*

Jentic

MCP Lego Kit



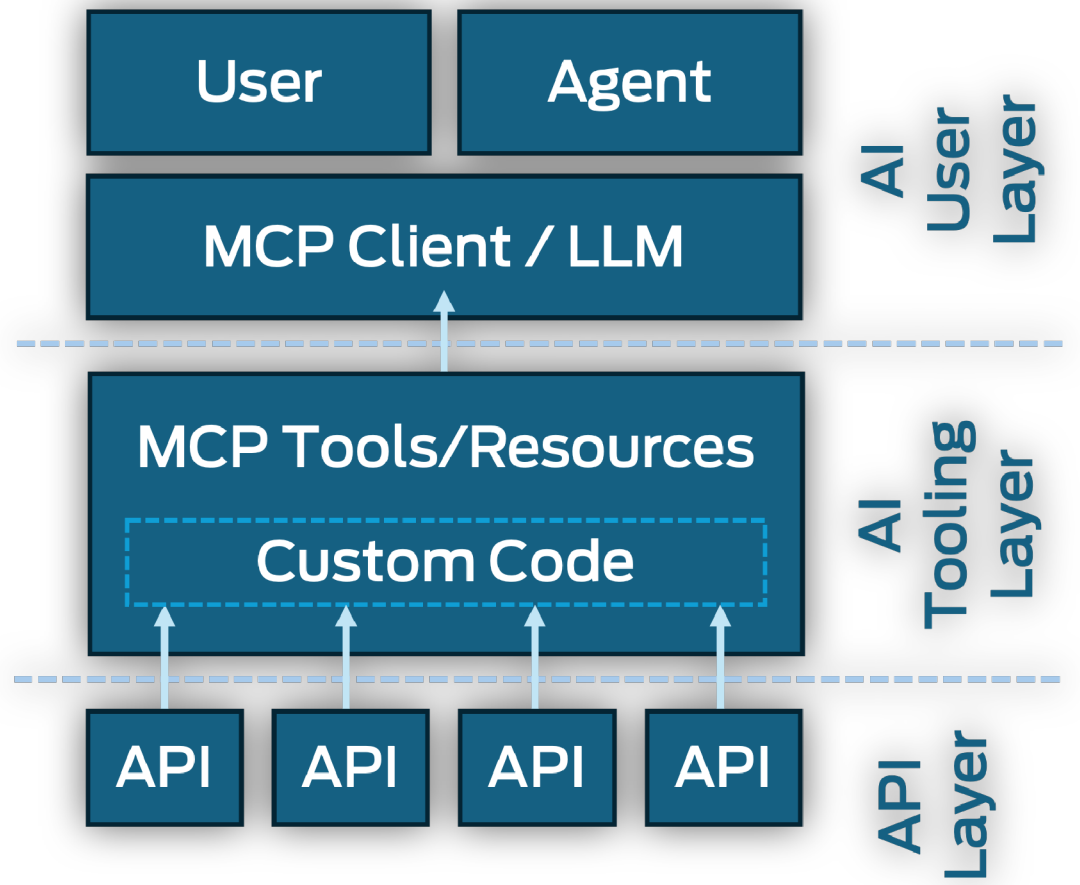
Jentic

Custom Tools in MCP: Provide Better Tools

1. Start from specific use cases
2. Design the best tools
3. Custom-made MCP server for one use case

Jentic

Custom Tools in MCP



Jentic

MCP Problems

- Problem #1: Complex APIs will overload the MCP client's context
- Problem #2: Larger API landscapes will overload the MCP client's context
- Problem #3: Poor API design translates to poor MCP results
- Problem #4: Bad AX results in unsatisfactory AI results

Jentic

Outline

1. APIs and AI: It's complicated [3]
2. Model Context Protocol (MCP) [8]
3. Supporting AI with APIs at Scale [13]
4. Conclusions [2]

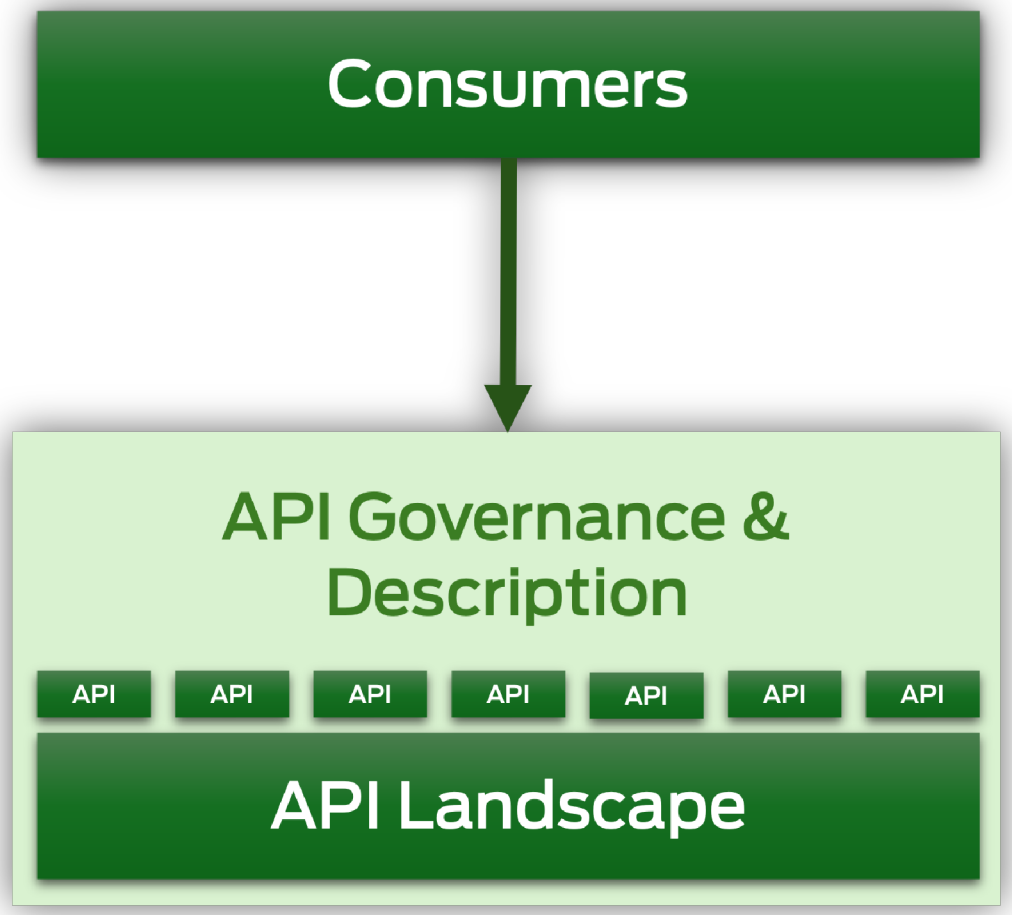
Jentic

From DX to AX

- APIs were typically consumed by developers
- Developers write code that *often fills API/UX gaps*
- AI takes the developer out of the picture
- LLMs have much more limited context than developers

Jentic

From DX to AX



Jentic

Do Your DX Homework

- Most API descriptions need quality improvements anyway
- Better descriptions benefit *everyone*, not just AI agents
  - Complete and meaningful operation summaries
  - Comprehensive examples for all operations
  - Clear error handling
  - Consistent terminology and naming
- Good DX is the foundation for good AX

Jentic

Now There's AX Homework Too

- AI agents need information developers don't typically require:
  - Everything that developers usually “just know”
  - Treat every API like an external one for developers with no context
- OpenAPI Initiative (OAI) might add AX-specific fields
- Until then, leverage description fields strategically

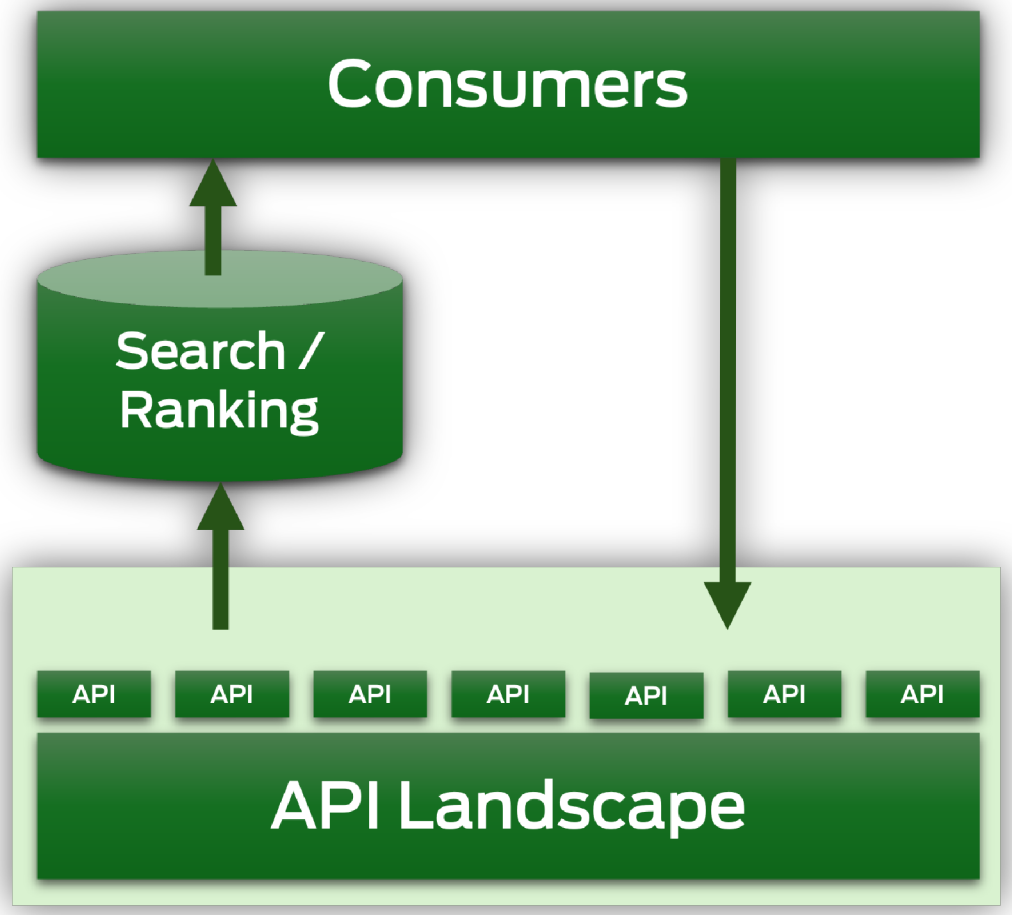
Jentic

API Complexity: GitHub v3

- 11 MB JSON
- 320k lines
- 738 paths
- 1114 operations
- 259 webhooks

Jentic

Agents are Dynamic



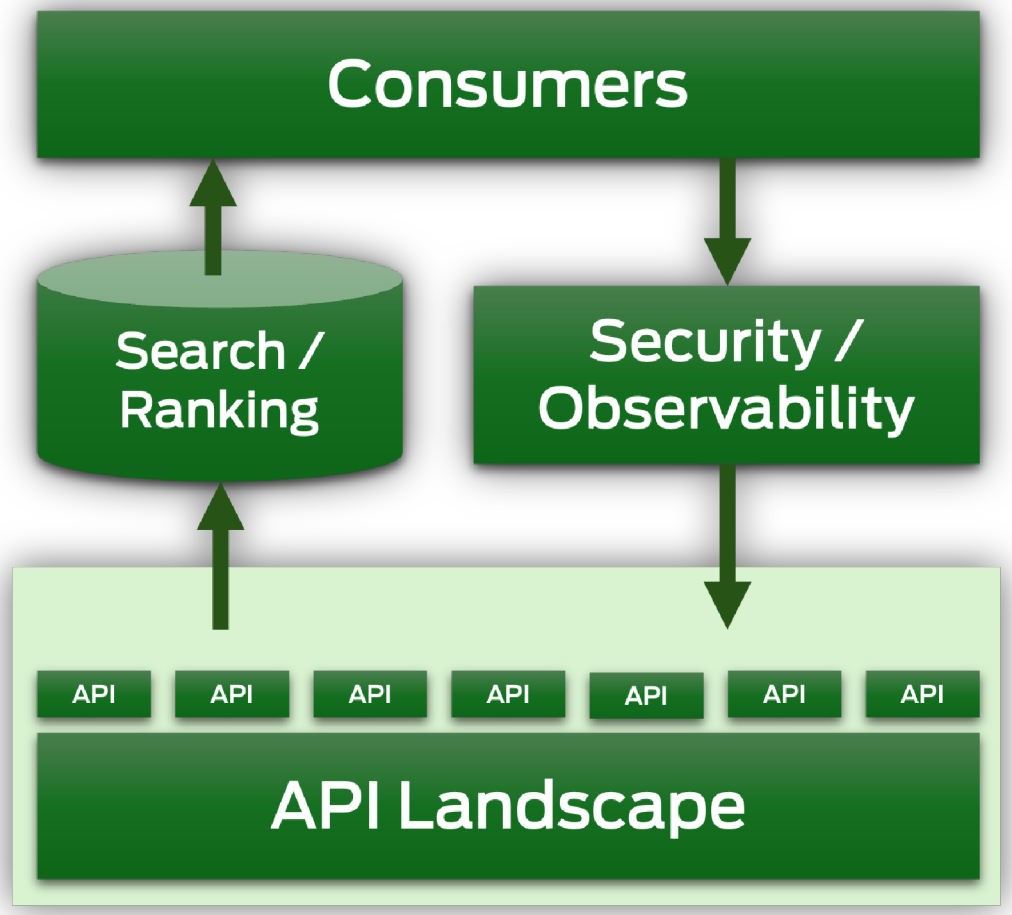
Jentic

API Microproducts

- Traditional API products bundle endpoints into coarse offerings
- With unknown use cases, it's better to decompose into fine-grained “API microproducts”
    - *OpenAPI:* Treat all operations on all endpoints as individual capabilities
    - *AsyncAPI:* Treat all messages on all channels as individual capabilities
    - *GraphQL:* Treat all queries and mutations as individual capabilities
- Allows flexible combination of capabilities for any use case

Jentic

Secure and Observe AI Consumers



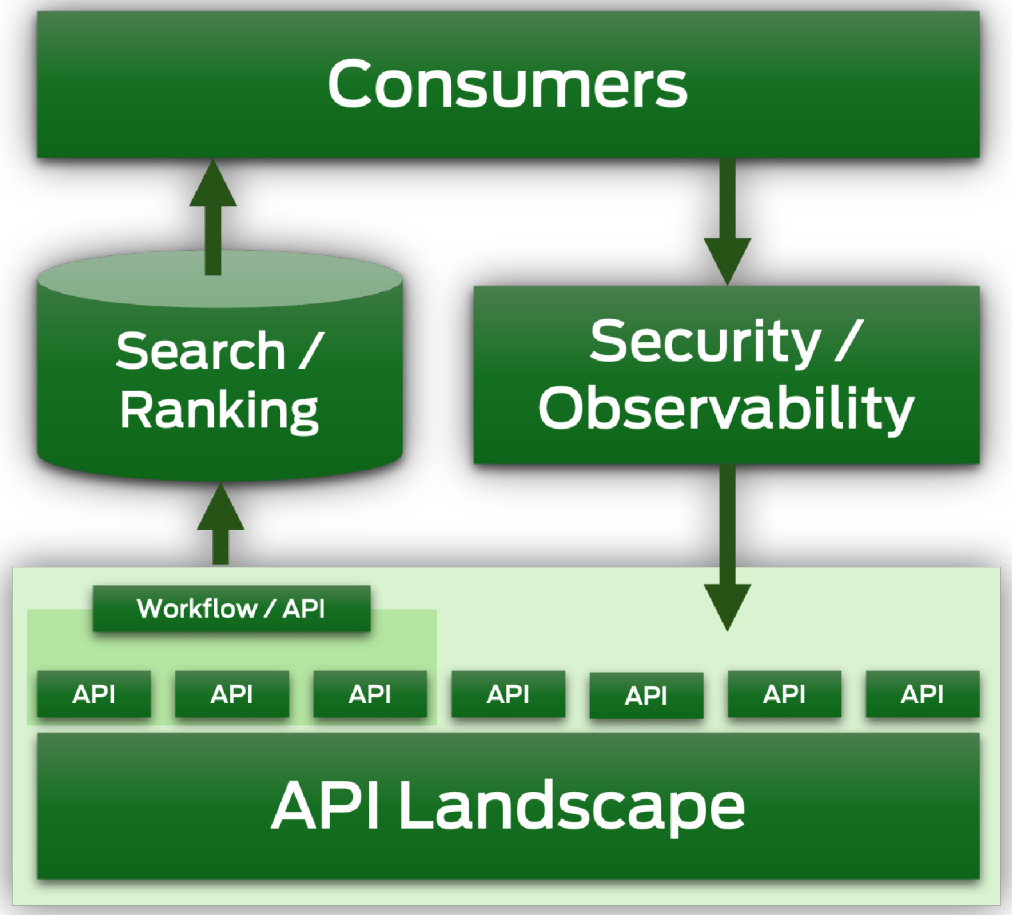
Jentic

Secure Centralized Access for Agents

- Avoid leaking secrets and tokens directly to agents
- Prevent agents from impersonating users or apps
- Ensure proper token delegation and downscoping
- Centralized access gives visibility into all agent activity
- Spot common patterns and emerging workflows

Jentic

Add Workflows



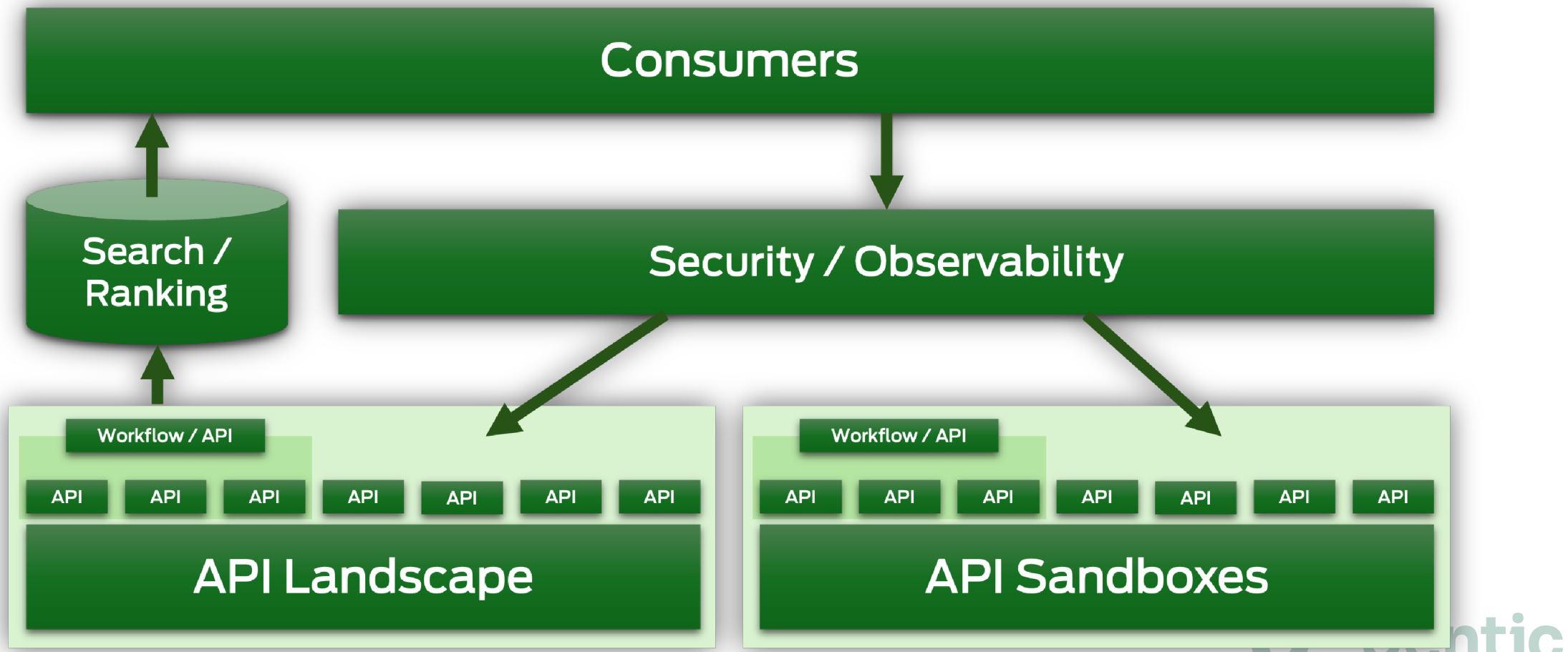
Jentic

Observing Agents to Discover Workflows

- Agents try to solve tasks in their own way
- Observing agents reveals *workflow candidates*
- More energy efficient than trial-and-error agent execution
- Workflows are published as new APIs, usable beyond AI

Jentic

Managing Exposure to Agents



Sandboxes for Secure Experimentation

- Create a controlled environment for agents
- Prevent agents from accessing production secrets
- Provide realistic data (synthetic or mock) for testing
- Centralized control: we decide where agents can play

Jentic

Outline

1. APIs and AI: It's complicated [3]
2. Model Context Protocol (MCP) [8]
3. Supporting AI with APIs at Scale [13]
4. Conclusions [2]

Jentic

Define. Describe. Discover.

- *Define* APIs for agentic actions
- *Describe* APIs for agentic understanding
- *Discover* APIs at runtime for agentic tool use

Jentic

Thank You!

- YouTube youtube.com/ErikWilde
- LinkedIn linkedin.com/in/erikwilde