

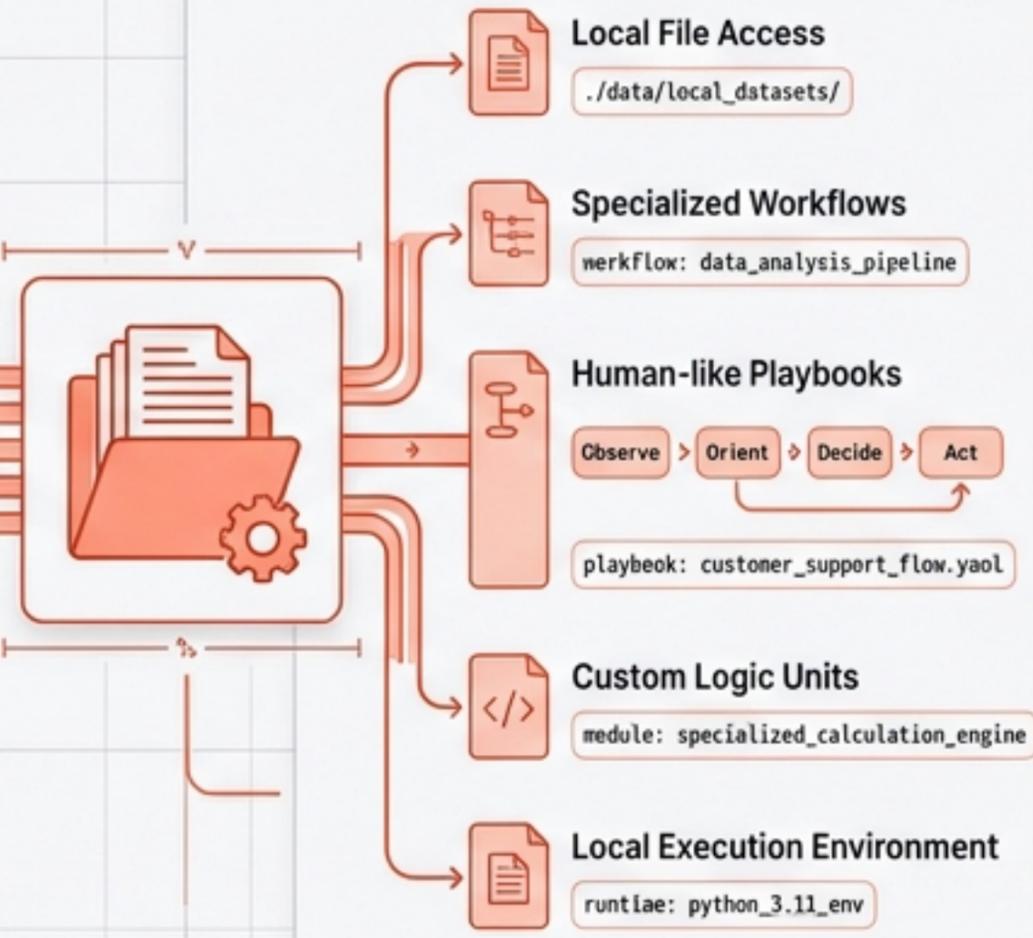
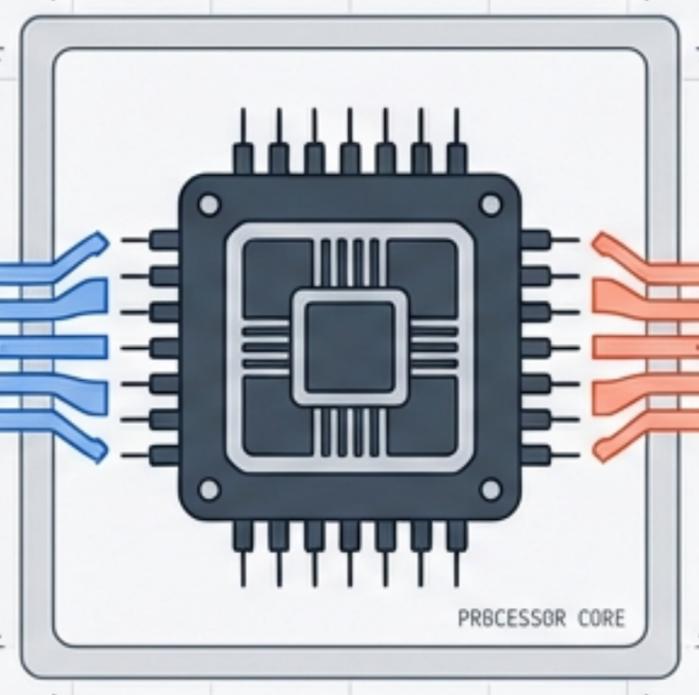
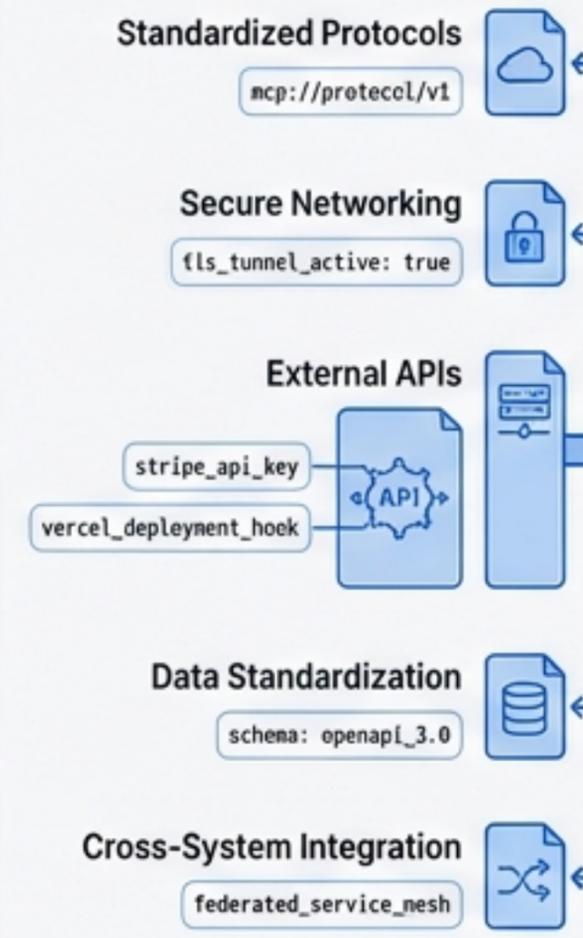
```
function init_agent(config: MCPConfig, skills: SkillRegistry) -> Agent {
  // Establish secure MCP tunnel
  let mcp_connection = await MCP.connect(endpoint, credentials);

  // Load local skill playbooks
  const skill_manifest = await fs.readFile('./skills/manifest.json', 'utf8');
  for (const skill of JSON.parse(skill_manifest)) {
```

```
0x000000000000a oex
0x08600886880a nax
0x080009068188 system call
0x0000090e6998 nov
0x000008067e68 nov three [0xBASEBRY, FECONT]
0x0000080e8783 system call
0x0000800a818# system call
```

### MCP INFRASTRUCTURE

### AGENT SKILLS



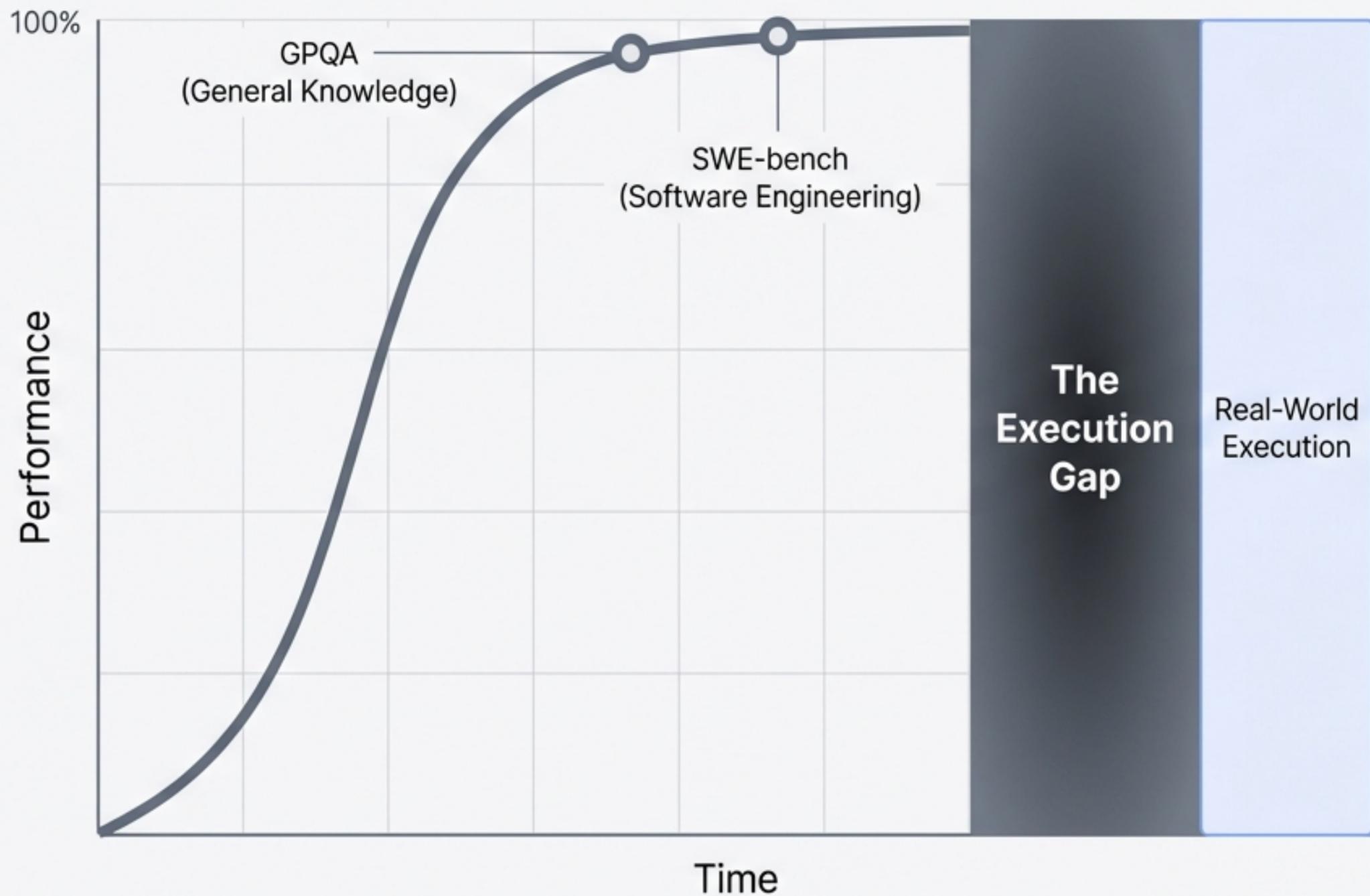
# The Agentic Blueprint

## Architecting Capable AI: A Developer's Guide to MCP and Agent Skills

```
hex30cce: 0x619066728
analysisCalled: 0x805827E7
0x80592788
0x80892787
ineliciteAI om2: 0x802027B8
0x8020276F
0x00909728
0x8F38A498
processor: on15: 0x8F3094A0
processor: on15: 0x8F906410
asynctsr: coot: 0xSF806656
processor: on12: 0xSF9069a0
```

```
0x90880808881D4 code
0x000008060188 util
0x000008001D8 edv
0x0000080010e code
0x000008001EE rev
0x00000806820e 1w1_oav_local_isites.PaADIM.N. hop
// Optimize data flow
agent.es.optimizeRouter({
  priority: 'latency',
  fallback: 'local_execution' })
```

# Intelligence Does Not Equal Agency



LLMs have mastered theoretical knowledge, approaching near-perfect performance on complex benchmarks.

**The Bottleneck:** Models are isolated. They cannot read proprietary codebases, access live systems, or execute scripts without an integration layer.

# The Anatomy of a Real-World Task

To review a codebase, an AI needs more than intelligence. It requires sequential access to external tools, live environments, and specific operating contexts.



Step 1: Read  
Proprietary Code

Status: Requires  
GitHub Integration



Step 2: Run Tests &  
Find Silent Bugs

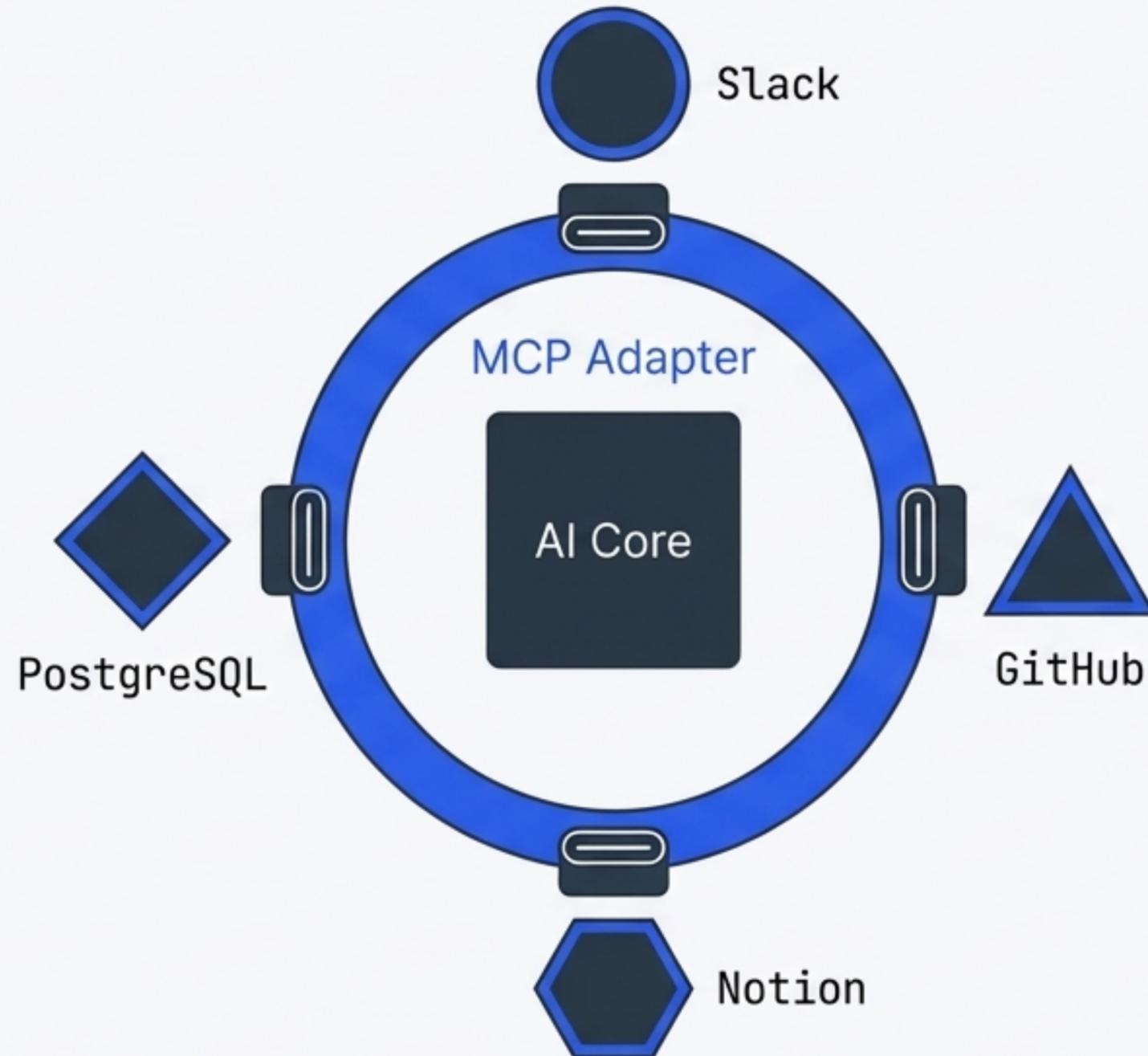
Status: Requires  
Code Interpreter



Step 3: Follow  
Team Standards

Status: Requires  
Internal Documentation

# Solution 1: The Model Context Protocol (MCP)



## The Concept

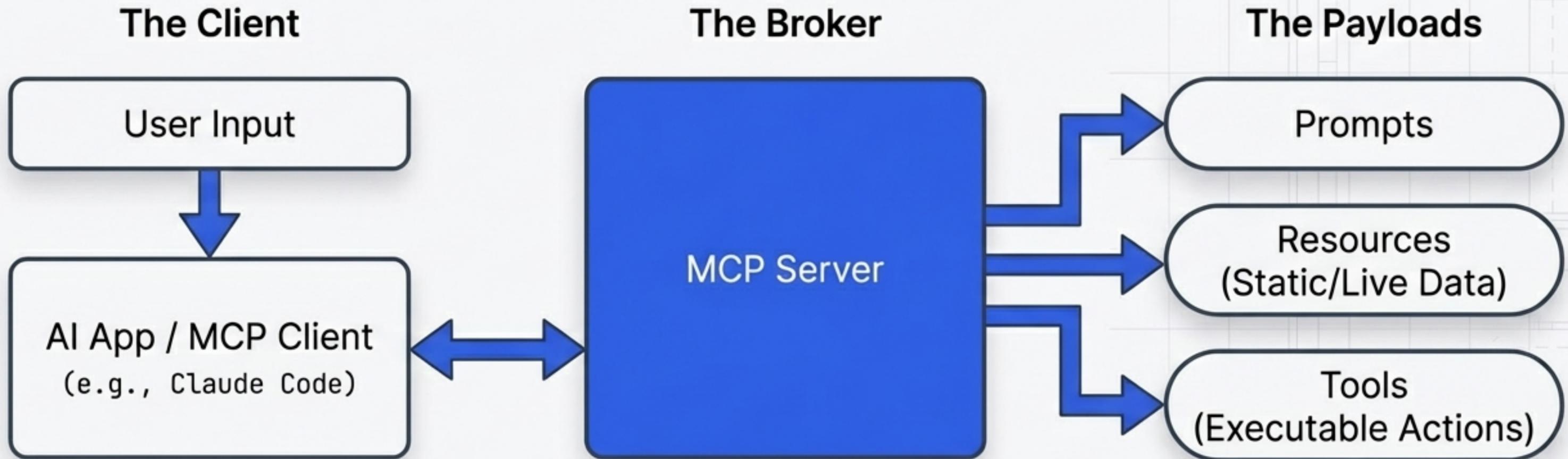
The universal USB-C standard for AI applications.

## The Value

Instead of building custom integrations for every tool, MCP provides a single open standard allowing any AI application to communicate with any external resource.

# The MCP Architecture

An open communication layer where the AI acts as the Client, requesting available tools and schemas from the MCP Server to interact with the outside world.



# The Problem with MCP: Context Rot



## The Mechanism

**MCP** injects all available tool schemas into the context window immediately upon startup.

## The Consequence

A single server might contain dozens of irrelevant tools (e.g., web search, email sorting) that consume precious tokens, rack up costs, and degrade model performance.

# Solution 2: Agent Skills



## The Concept

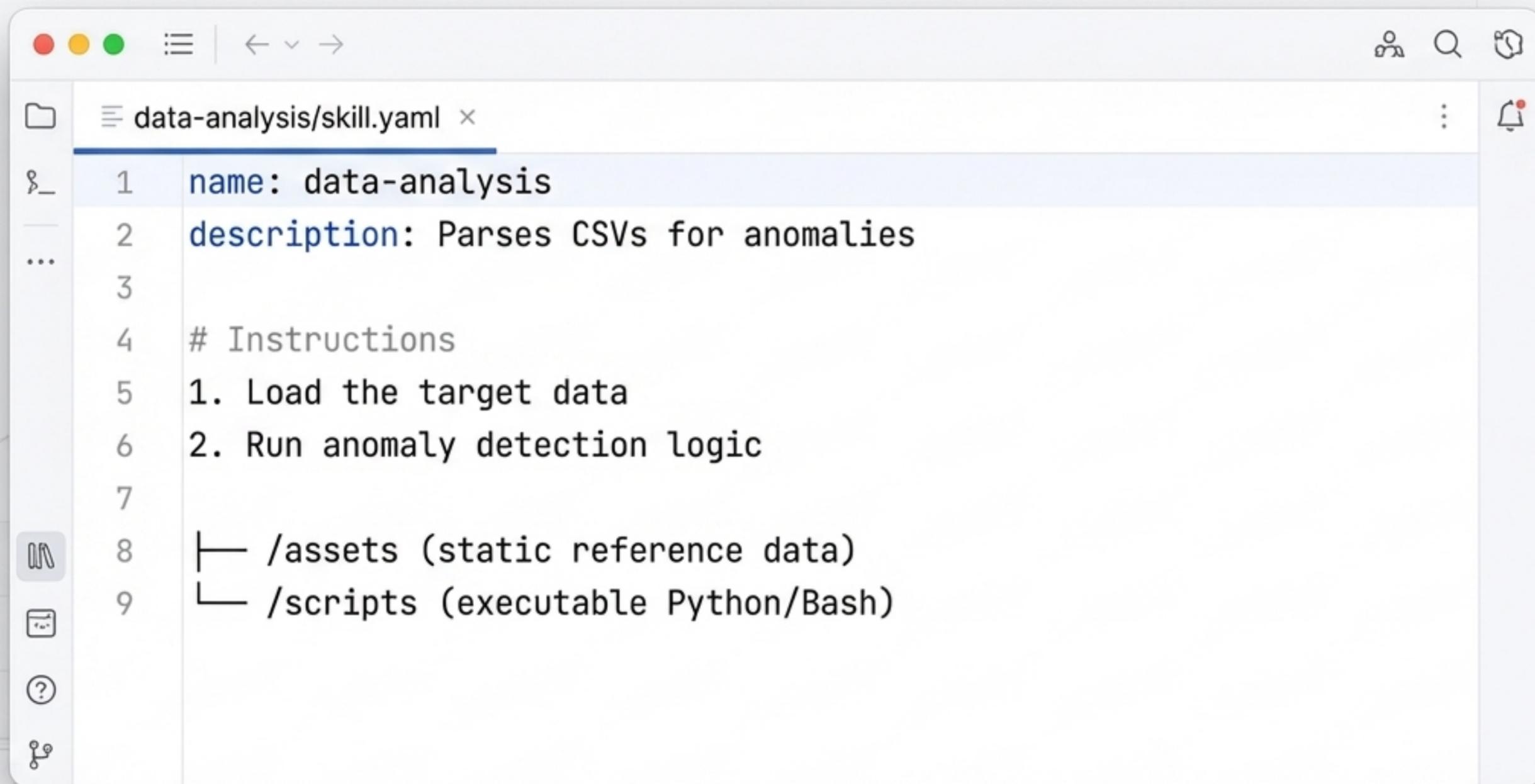
A local, markdown-based alternative that gives agents highly specific context and playbooks only when triggered.

## The Value

Bypasses token bloat by utilizing a file-system approach instead of a network-server approach. The agent only reads what it needs.

# The Anatomy of a Skill

A skill is not a server; it is a directory tree encoding procedures, references, and scripts into the agent's file system.



```
1 name: data-analysis
2 description: Parses CSVs for anomalies
3
4 # Instructions
5 1. Load the target data
6 2. Run anomaly detection logic
7
8 └─ /assets (static reference data)
9 └─ /scripts (executable Python/Bash)
```

→ [Front Matter]

→ [Body]

→ [Nested Directories]

# The Power of Progressive Disclosure

## THE CONCEPT

Unlike MCP's upfront data dump, Skills perfectly preserve the context window by feeding the agent exactly what it needs for the very next step.

### Startup: Front Matter

Loads ~100 tokens. Just the skill metadata.

### Triggered: Body Instructions

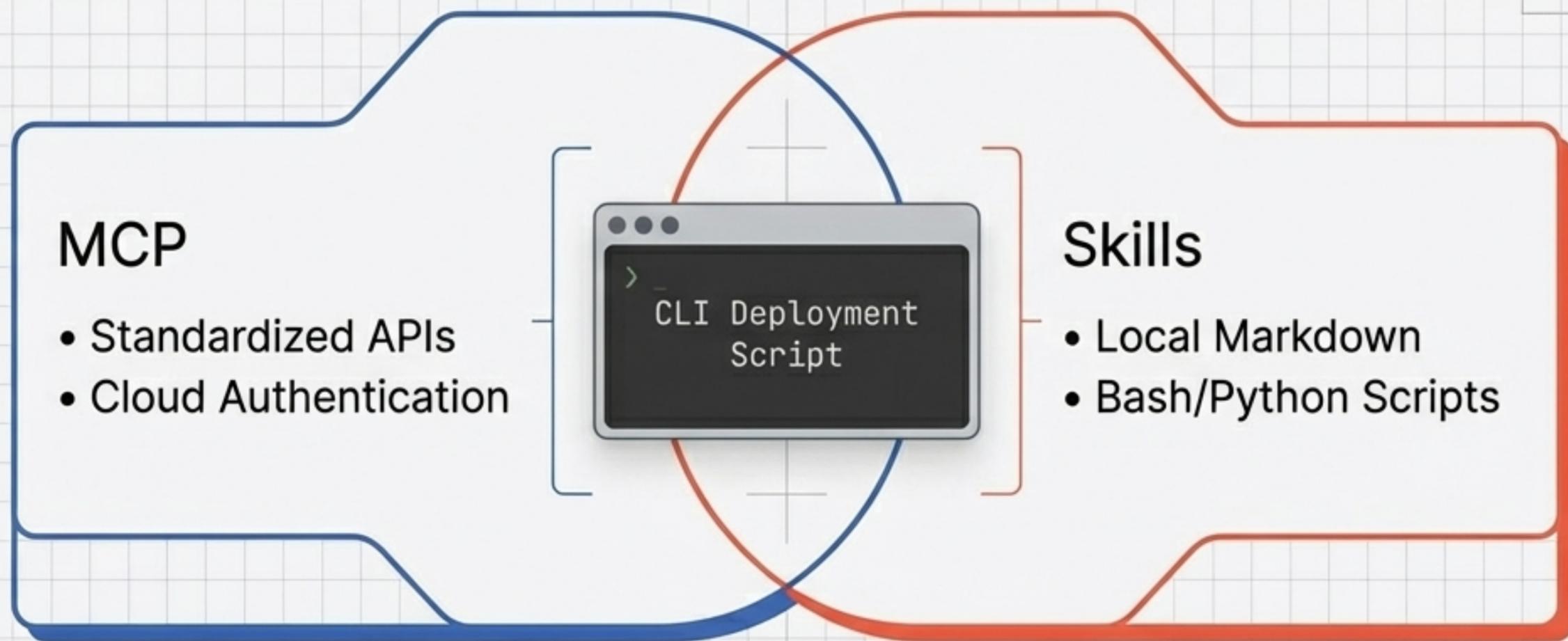
Loads up to 5,000 tokens. Loaded only if the LLM decides the skill is relevant.

### Execution: Scripts & Assets

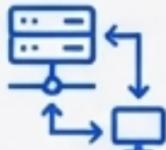
Infinite potential context. Accessed strictly on-demand via code interpreter.

# The Point of Confusion

Both frameworks give tools to agents. Because local Skills can execute CLI scripts—which in turn can authenticate and deploy to the cloud—they can mimic the behavior of an MCP server. For local coding agents, the lines blur.



# The MCP vs. Skills Matrix

	Model Context Protocol (MCP)	Agent Skills
Core Purpose	 Tool access & external integration	 Workflow instructions & playbooks
Architecture	 Client-Server / Networked JetBrains Mono	 Local File System / Directory JetBrains Mono
Setup	 Requires writing backend code/servers	 Written in plain English / Markdown
Payload Execution	 Real-time external APIs JetBrains Mono	 Local executable code/scripts JetBrains Mono
Token Efficiency	 Heavy upfront load (schemas injected at startup)	 Progressive disclosure (loaded strictly as needed)



# When to Deploy Which Framework

Use **MCP** for broad tool access and external authentications. Use **Skills** to orchestrate those tools into highly specific, token-efficient workflows.



## The Infrastructure (MCP)



"I need my agent to read and write to Notion."



### Deploy MCP.

Provides the 15 out-of-the-box API tools to create pages and fetch databases.



## The Execution (Agent Skills)



"I need my agent to analyze user interviews and scope MVPs using our company's specific formatting."



### Deploy Agent Skills.

Instructs the agent exactly how to sequence those Notion tools perfectly.