NETWORK AUTOMATION FORUM

#!/ William Collins
Director, Technical Evangelism
itential

Is MCP just an API Abstraction?

> Or something completely different?

AUTOCON 4

THE NETWORK AUTOMATION CONFERENCE


NAF
NETWORK AUTOMATION FORUM
Agenda
Examining the Landscape
API Tradecraft
The MCP Experience
Adding Intelligence

Examining the Landscape

> MCP servers are just wrappers around APIs. Everyone has a gimmick.

> We're going to see MCP replace RESTful APIs in the near future.

> MCP offers nothing beyond existing function-calling mechanisms.

Examining the Landscape

Announced on *November 25, 2024*

What is MCP?

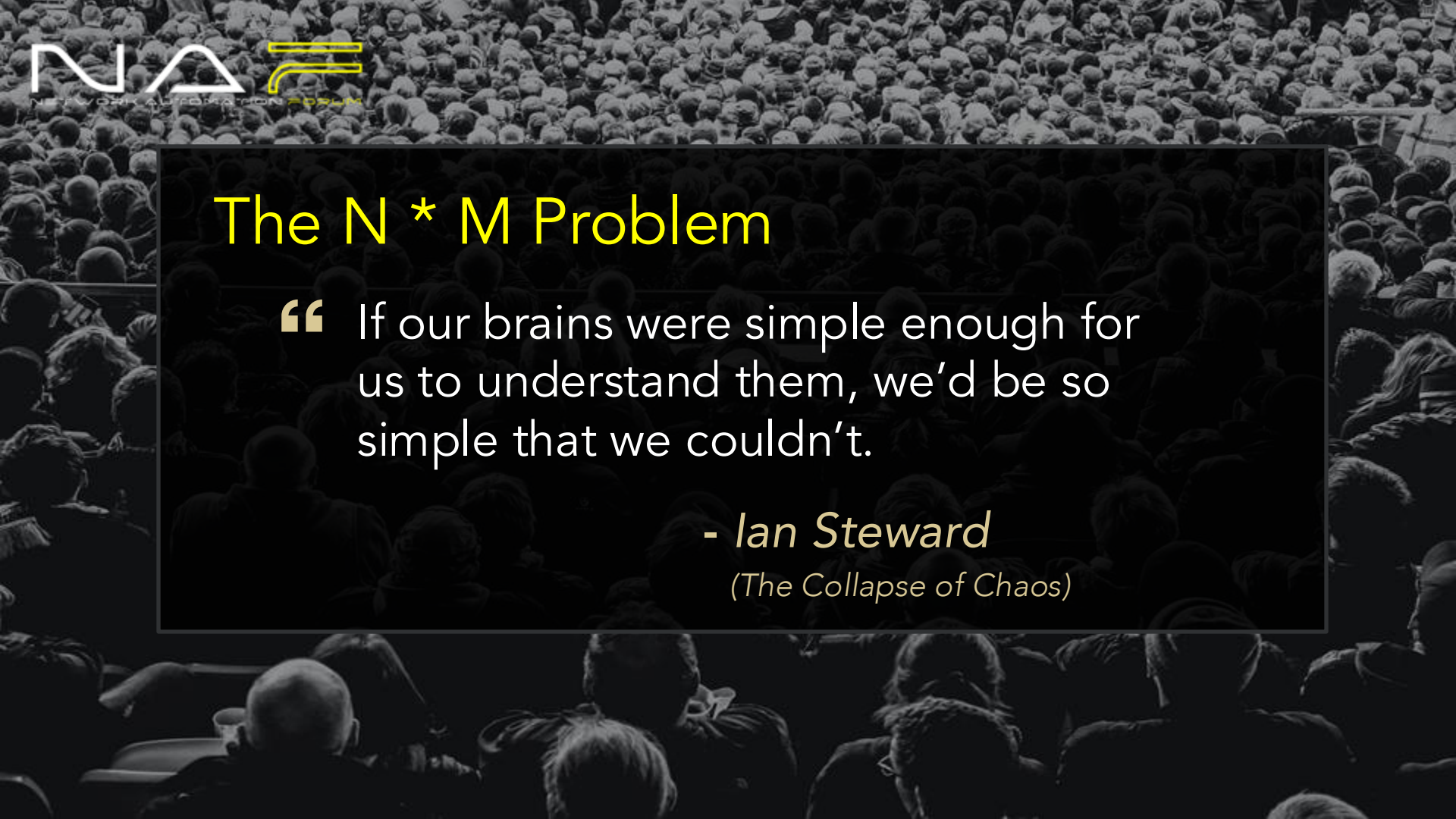
Open protocol that enables seamless integration between LLM applications and external data sources and tools.

Why does MCP Exist?

Without a standard, connecting AI applications to data sources requires custom integrations for every combination.

What was the Problem Again?

N * M Integration Problem



The N * M Problem

> “ If our brains were simple enough for us to understand them, we’d be so simple that we couldn’t.
>
> - *Ian Steward*
> *(The Collapse of Chaos)*

The N * M Problem

4 models with access to *10 tools* = **40 separate integrations**

The N * M Problem

4 models with access to *10 tools* = **40 separate integrations**

Bottleneck!

Vendor Lock-in: Changing models means re-writing integrations

High Maintenance: Since APIs change (A lot), so will the tools

Quadratic Complexity: Work scales quadratically, not linearly

The N * M Problem

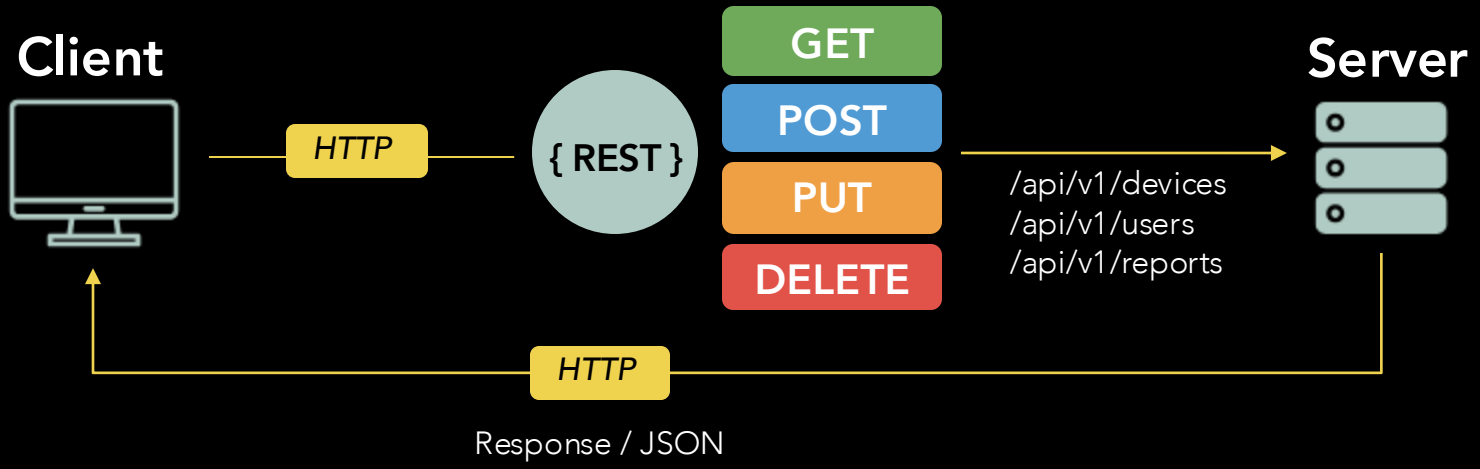
MCP shifts the equation from a multiplicative problem to an additive one

To connect ***N models*** to ***M tools*** you need ***N clients*** + ***M servers***


NETWORK AUTOMATION FORUM
API Tradecraft
“ If we open a quarrel between past and present, we shall find that we have lost the future.
- Winston Churchill

API Tradecraft

> To err is human, to send a ***200 OK*** with an error in the body is API



Architecture for building networked applications, based on a stateless, client-server, request-response model.

API Tradecraft *(RESTful API calls as we know them)*

- HTTP GET Request
  - URL Path
  - Headers
  - Timeout

⌥⌘2 nvim

```
19 def get_player_stats_api(player_id: int) -> Optional[dict]:
20     """Let's make an API call!"""
21
22     try:
23         response = requests.get(
24             f"https://api-web.nhle.com/v1/player/{player_id}/landing",
25             headers={"User-Agent": "MyApp/1.0"},
26             timeout=10
27         )
28         response.raise_for_status()
29         data = response.json()
```

API Tradecraft *(RESTful API calls as we know them)*

```python
# extract current season stats
current_season = data['featuredStats']['regularSeason']['subSeason']

# extract player info
player_name = f"{data.get('firstName', {}).get('default', '')} {data.get('lastName', {}).get('default', '')}"
team = data.get('currentTeamAbbrev', 'Unknown')
position = data.get('position', 'Unknown')

# stats
return {

    # player info
    "player_name": player_name,
    "team": team,
    "position": position,
    "games_played": current_season.get('gamesPlayed', 0),

    # scoring stats
    "goals": current_season.get('goals', 0),
    "assists": current_season.get('assists', 0),
    "points": current_season.get('points', 0),
    "shots": current_season.get('shots', 0),
    "shooting_pct": round(current_season.get('shootingPctg', 0) * 100, 1),

    # special teams
    "power_play_goals": current_season.get('powerPlayGoals', 0),
    "power_play_points": current_season.get('powerPlayPoints', 0),
    "shorthanded_goals": current_season.get('shorthandedGoals', 0),
    "shorthanded_points": current_season.get('shorthandedPoints', 0),

    # impact
    "game_winning_goals": current_season.get('gameWinningGoals', 0),
    "ot_goals": current_season.get('otGoals', 0),
    "plus_minus": current_season.get('plusMinus', 0),

    # discipline
    "penalty_minutes": current_season.get('pim', 0)
}
```

Connor's Stats

📊 BASIC STATS
Games Played: 19
Goals: 7
Assists: 20
Points: 27
+/-: -3

🎯 SHOOTING
Shots: 58
Shooting %: 12.1%

⚡ SPECIAL TEAMS
Power Play Goals: 2
Power Play Points: 11
Shorthanded Goals: 0
Shorthanded Points: 1

🏆 IMPACT
Game Winning Goals: 0
Overtime Goals: 0

⚠️ DISCIPLINE
Penalty Minutes: 8

API Tradecraft *(Can't we just use APIs for everything?)*

Manual Function Calling

- Artisan crafted JSON (tool schema)
- Lots of "glue code" connecting tools to APIs
- Build the full agent loop

```
# define tools; each API endpoint gets converted to a tool definition
tools = [
    {
        "name": "get_player_stats",
        "description": "Get current season statistics for an NHL player by name. Returns comprehensive stats
        "input_schema": {
            "type": "object",
            "properties": {
                "player_name": {
                    "type": "string",
                    "description": "Full or partial player name (e.g., 'Connor McDavid', 'McDavid')"
                }
            },
            "required": ["player_name"]
        }
    }
]
```

API Tradecraft *(Can't we just use APIs for everything?)*

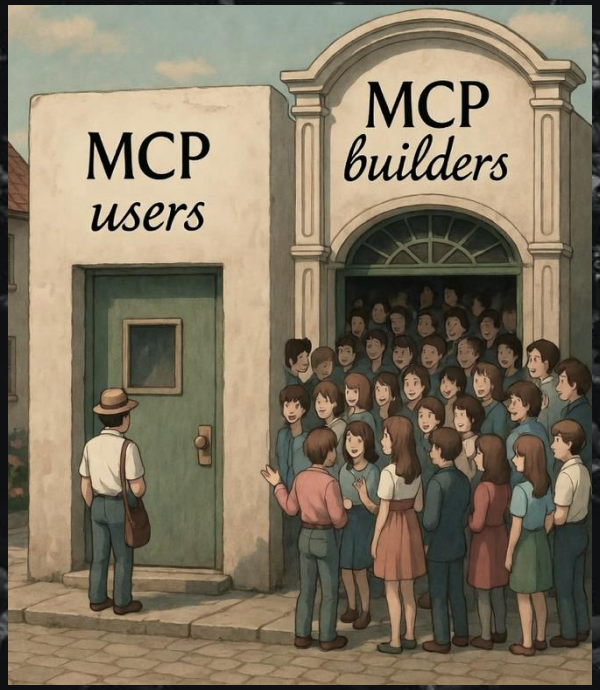
Generate via existing OpenAPI spec files *(Leeroy Jenkins Approach)*

- Automatically generated
- Lots of "glue code"
- Back to writing custom logic
  - *Filtering*
  - *Categorization*
  - *Dynamic Selection*

1 spec -> 100 tools!

- Bloated context window
- Decision paralysis
- Lower accuracy
- Slow responses
- Degraded agent performance

```python
def openapi_to_tools(spec_yaml: str) -> List[Dict[str, Any]]:
    """
    Automatically convert OpenAPI spec to LLM tool definitions.
    """
    spec = yaml.safe_load(spec_yaml)
    tools = []

    for path, path_item in spec.get("paths", {}).items():
        for method, operation in path_item.items():
            if method not in ["get", "post", "put", "delete"]:
                continue

            # create tool from operation
            tool = {
                "name": operation.get("operationId"),
                "description": f"{operation.get('summary')}. {operation.get('description', '')}",
                "input_schema": {
                    "type": "object",
                    "properties": {},
                    "required": []
                }
            }

            # convert parameters
            for param in operation.get("parameters", []):
                tool["input_schema"]["properties"][param["name"]] = {
                    "type": param.get("schema", {}).get("type", "string"),
                    "description": param.get("description", "")
                }
                if param.get("required"):
                    tool["input_schema"]["required"].append(param["name"])

            tools.append(tool)

    return tools
```



The MCP Experience

> MCP is probably the only tech that has more builders than users
>
> - *Abraham Lincoln*

The MCP Experience *(Core Conceptual Differences)*

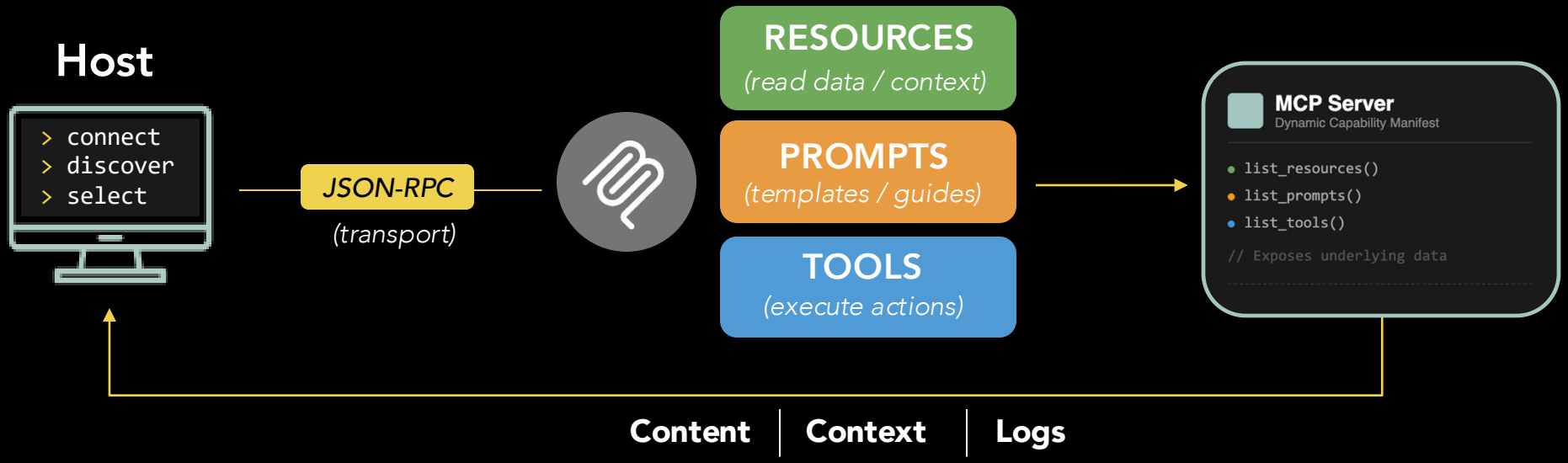
RESTful API

- Built for "humans" to write software against
- REST is an architectural style that uses HTTP *(a protocol)*
- Stateless by design; Self-contained requests, no server-side session state

MCP

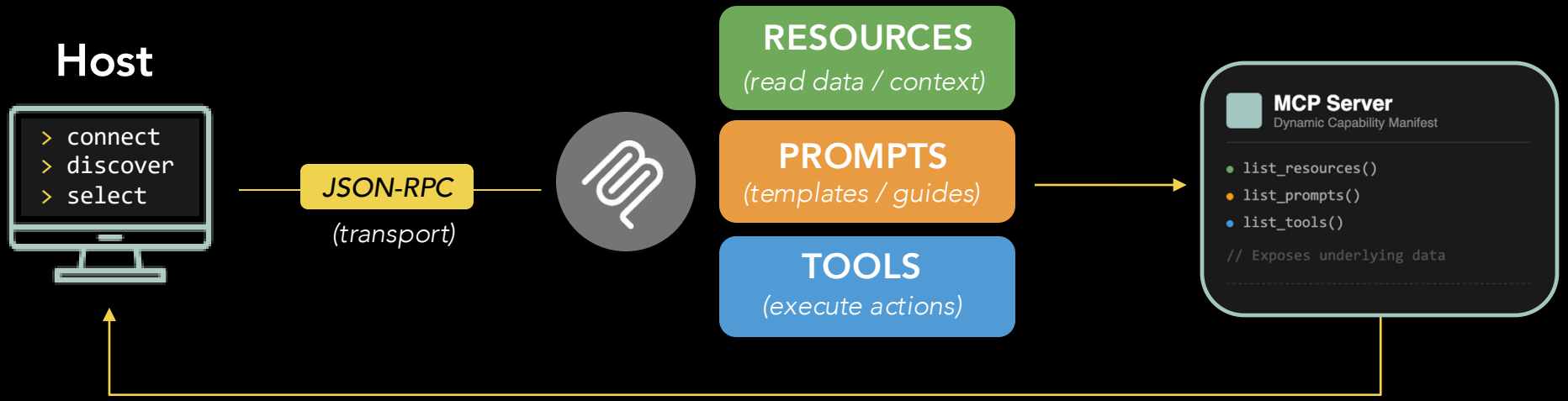
- Built for "AI models" to interact with external systems safely
- Open-Source protocol with a public specification
- Stateful by design; Maintains connection context, stateful transport

The MCP Experience *(Core Conceptual Differences)*



> Standardized protocol for connecting AI models to external data & capabilities, based on a stateful, client-server, session-based model with dynamic capability discovery.

The MCP Experience *(Core Conceptual Differences)*



Content *(The "What")*
Semantically-tagged data from resources / tools

Context *(The "Where")*
Loads data into model's context window

Logs *(The "How")*
Push diagnostics back to client

The MCP Experience *(No Manual Tool Schema)*

Decorators automatically generate schemas from your docstrings.

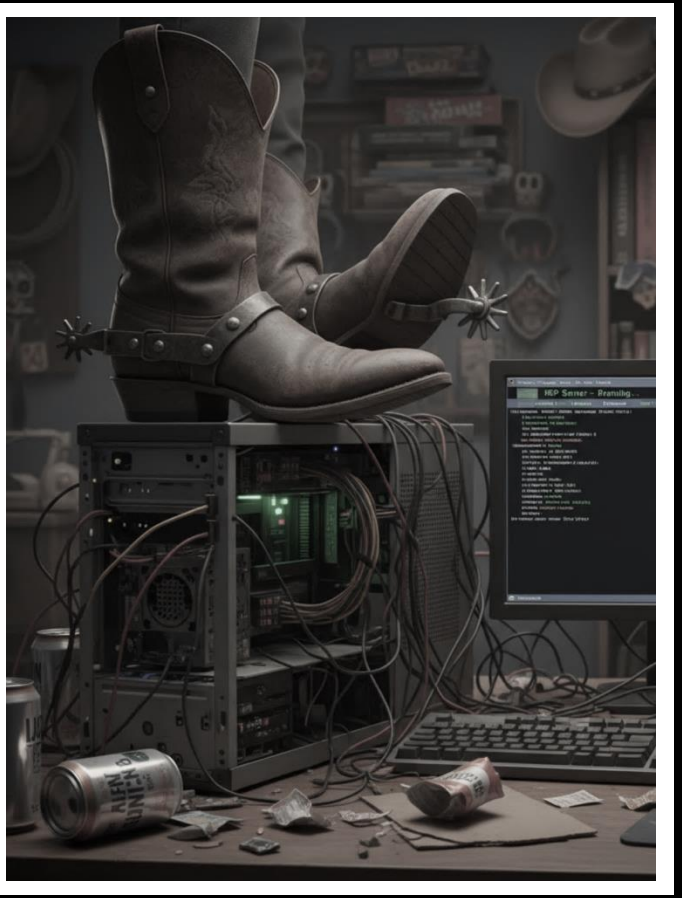
No Glue Code!

- No JSON-RPC message handling
- No agent loop
- No manual tool registration

⌥⌘1 nvim

```python
@mcp.tool()
def get_player_stats(player_name: str) -> dict:
    """
    Get current season statistics for an NHL player by name.

    Returns comprehensive stats including goals, assists, points, shooting percentage,
    special teams performance, game-winning goals, and more. Automatically handles
    player name lookup and ID conversion.

    Args:
        player_name: Full or partial player name (e.g., "Connor McDavid", "McDavid")

    Returns:
        Dictionary with 17 stats across 5 categories:
        - Player info: name, team, position, games_played
        - Scoring: goals, assists, points, shots, shooting_pct
        - Special teams: power_play_goals, power_play_points, shorthanded_goals, shorthanded_points
        - Impact: game_winning_goals, ot_goals, plus_minus
        - Discipline: penalty_minutes

    Raises:
        ValueError: If player is not found or stats cannot be fetched
    """

    # convert name to player ID
    player_id = lookup_player_id(player_name)
    if not player_id:
        raise ValueError(f"Player '{player_name}' not found")

    # fetch stats
    url = f"https://api-web.nhle.com/v1/player/{player_id}/landing"
    headers = {"User-Agent": "MCPServer/1.0"}
```

The MCP Experience *(Client-Side Setup)*



git clone https://github.com/wcollins/talks

```
WilliamMBP:~/code $ git clone https://github.com/wcollins/talks
Cloning into 'talks'...
Warning: Permanently added 'github.com' (ED25519) to the list of known hosts.
remote: Enumerating objects: 261, done.
remote: Counting objects: 100% (27/27), done.
remote: Compressing objects: 100% (22/22), done.
remote: Total 261 (delta 7), reused 23 (delta 4), pack-reused 234 (from 1)
Receiving objects: 100% (261/261), 110.64 MiB | 48.21 MiB/s, done.
Resolving deltas: 100% (96/96), done.
```

Setup venv

```
WilliamMBP:~/code/talks/2025-naf-autocon4 (main)$ uv venv --python 3.11
Using CPython 3.11.8
Creating virtual environment at: .venv
Activate with: source .venv/bin/activate
WilliamMBP:~/code/talks/2025-naf-autocon4 (main)$ source .venv/bin/activate
```

Resolve / Install Dependencies

```
(2025-naf-autocon4) WilliamMBP:~/code/talks/2025-naf-autocon4 (main)$ uv sync
Resolved 42 packages in 3ms
Installed 39 packages in 32ms
 + annotated-types==0.7.0
 + anthropic==0.73.0
 + anyio==4.11.0
 + attrs==25.4.0
 + certifi==2025.11.12
 + cffi==2.0.0
 + charset-normalizer==3.4.4
 + click==8.3.1
```

The MCP Experience *(Client-Side Setup)*

Copy example configuration file to location your specific client expects to find it

Example: *~/Library/Application\ Support/Claude/claude_desktop_config.json*

```
⌥⌘1                                                    nvim
 1 {
 2   "mcpServers": {
 3     "nhl-stats": {
 4       "command": "/Users/william/code/talks/2025-naf-autocon4/.venv/bin/python",
 5       "args": [
 6         "/Users/william/code/talks/2025-naf-autocon4/ex04_mcp_server.py"
 7       ]
 8     }
 9   }
10 }
```

Command

Executable to run MCP server

(Points to Python interpreter in my venv)

Arguments

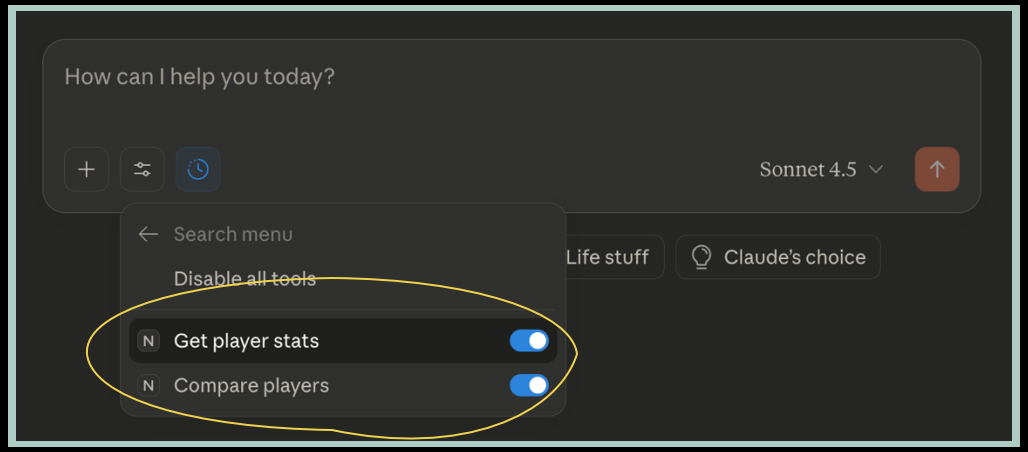
Array of args passed to command

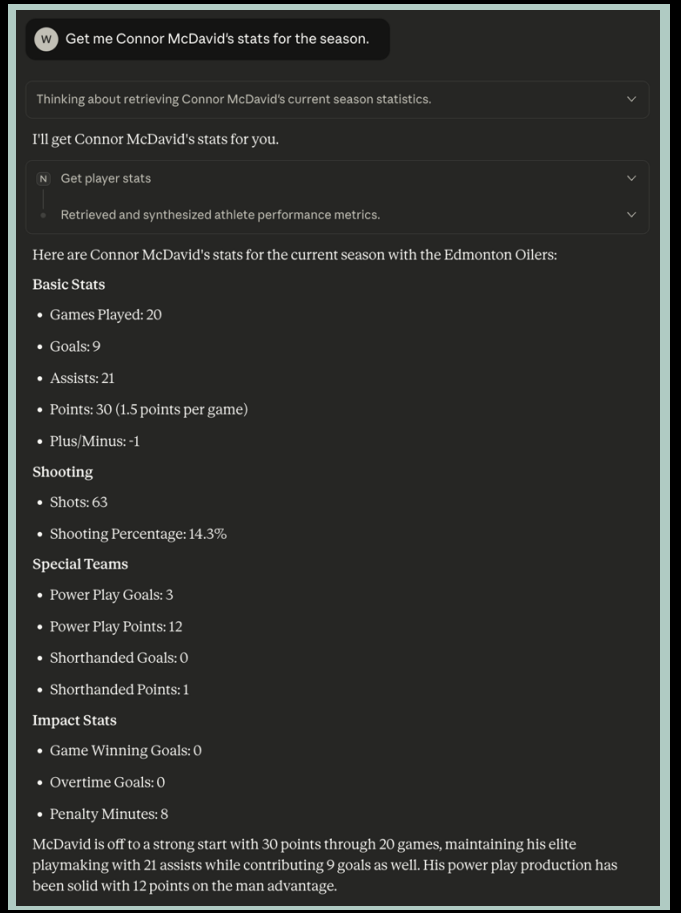
(Path to MCP server)

The MCP Experience *(Client-Side Setup)*

Toggle Tools

These tools then become part of your standard capabilities every time you start Claude Desktop.





Adding Intelligence

> MCP is the standardized interface we deserve, but not the one vendors needed right now. So, they'll question it. Because it can take it. Because it's not just another API wrapper. It's a context provider, a capability exposer. A model context protocol.

- Commissioner Gordon

The Dark Knight

Adding Intelligence *(Think Intentions, not Endpoints)*

❌ Endpoint Mapping *(API Wrapper)*

```
get_devices()
get_device_config(device_id)
validate_config(config)
get_compliance_rules()
check_rule(rule, config)
```

AI = *Systems Integrator*

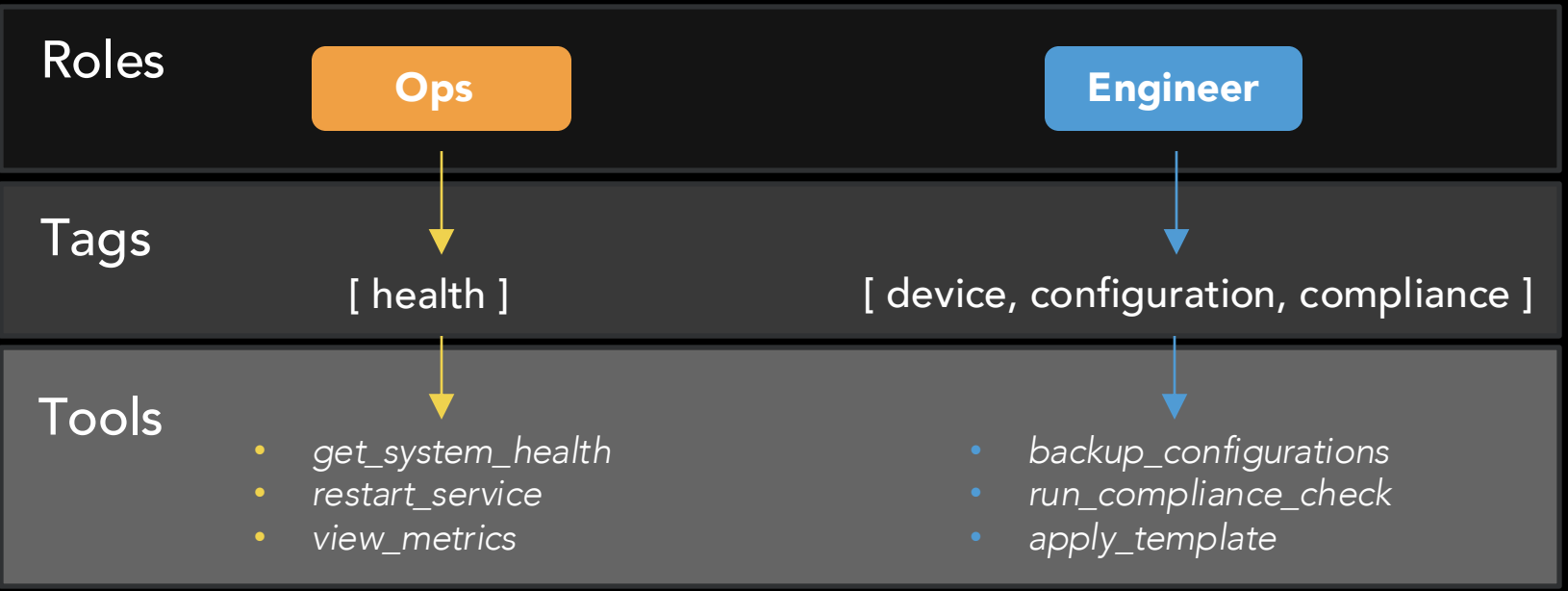
✓ Intention-Based *(Semantic Abstraction)*

```
run_compliance_check(device_group, policy)
backup_device_configurations(targets)
apply_golden_config(template, devices)
```

AI = *Domain Operator*

Adding Intelligence *(Context-Aware Tool Exposure)*

Role-Based *Capability Filtering*



Why? Reduce decision paralysis and ensure the AI is suggesting relevant capabilities.

Adding Intelligence *(Possible Evolution Paths)*

Let's give this Science Experiment Wheels!

MCP Gateway Pattern | Vendor Hosted | Private / Internal

Adding intelligence doesn't mean scale.

Intelligence (Capability)

- Can the system understand intent?
- Does it provide the right abstraction?
- Can it make good decisions?
- Does it have domain knowledge?

Scale (Capacity)

Can it handle 1,000 concurrent users?
Does it support multi-tenancy?
Can it process millions of requests?
Does it maintain performance under load?

NETWORK AUTOMATION FORUM

Thank You!

Let's Connect!

https://linktr.ee/MacroEngineered