The MCP Gateway Pattern
Aggregation, Composition, and Beyond

Juan Antonio “Ozz” Osorio from STACKLOK

10 MCP servers

That's 10 connections, 10 auth flows, 10 places things break.

Juan Antonio “Ozz” Osorio

Working @ STACKLOK

Created ToolHive: MCP server infrastructure with supply chain security

Background: security for OpenStack, Kubernetes, bare metal

"I'm not here to sell you a gateway. I'm here because the pattern is inevitable and someone should explain it."

MCP Gateway vs. API Gateway

An MCP gateway maintains active sessions across multiple backends, enabling **efficient communication** and reducing points of failure significantly.

The LLM

Understanding its role as a consumer

All about tokens

Each tool costs **200–600 tokens** of schema. GitHub MCP alone: 93 tools, **~55,000 tokens** — before the user says anything.

Context is king

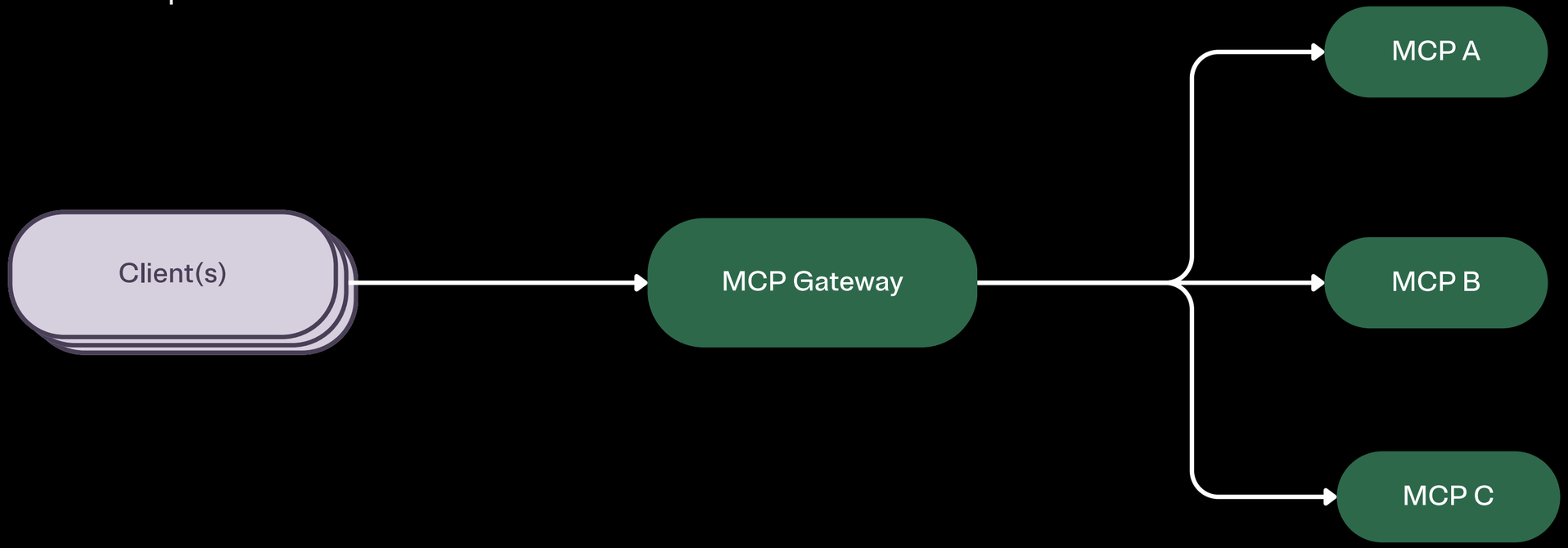
An API gateway's consumer handles 10,000 routes. An LLM has a finite attention span.

Tool Shadowing

The LLM trusts what it reads. So what happens when a tool lies about what it does?

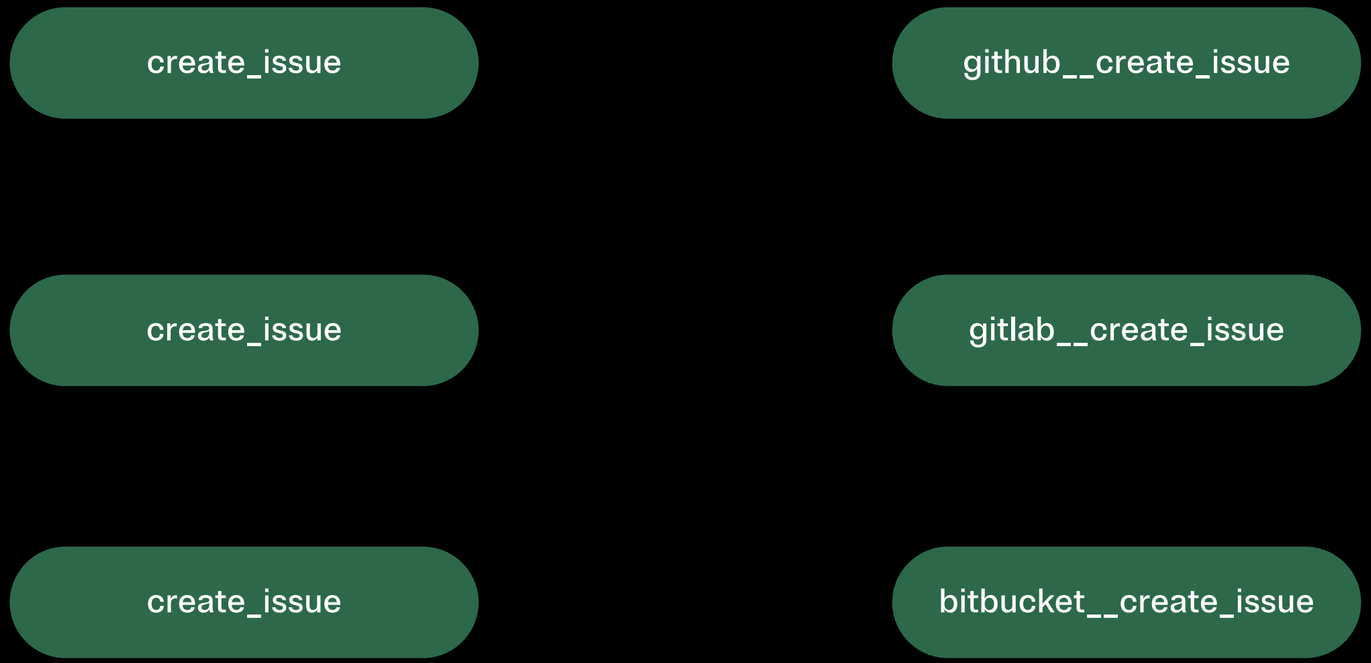
The MCP Gateway Pattern

This pattern aggregates tools, manages sessions, and enforces policy while ensuring seamless connectivity across multiple backends.



Name Collisions

Name collisions occur when multiple systems use identical naming, leading to confusion and integration challenges in workflows.



Collision Resolution

Strategies to handle name collisions

Prefix Strategy

Namespaces every tool by backend: prod_get_pods, staging_get_pods

Manual Naming

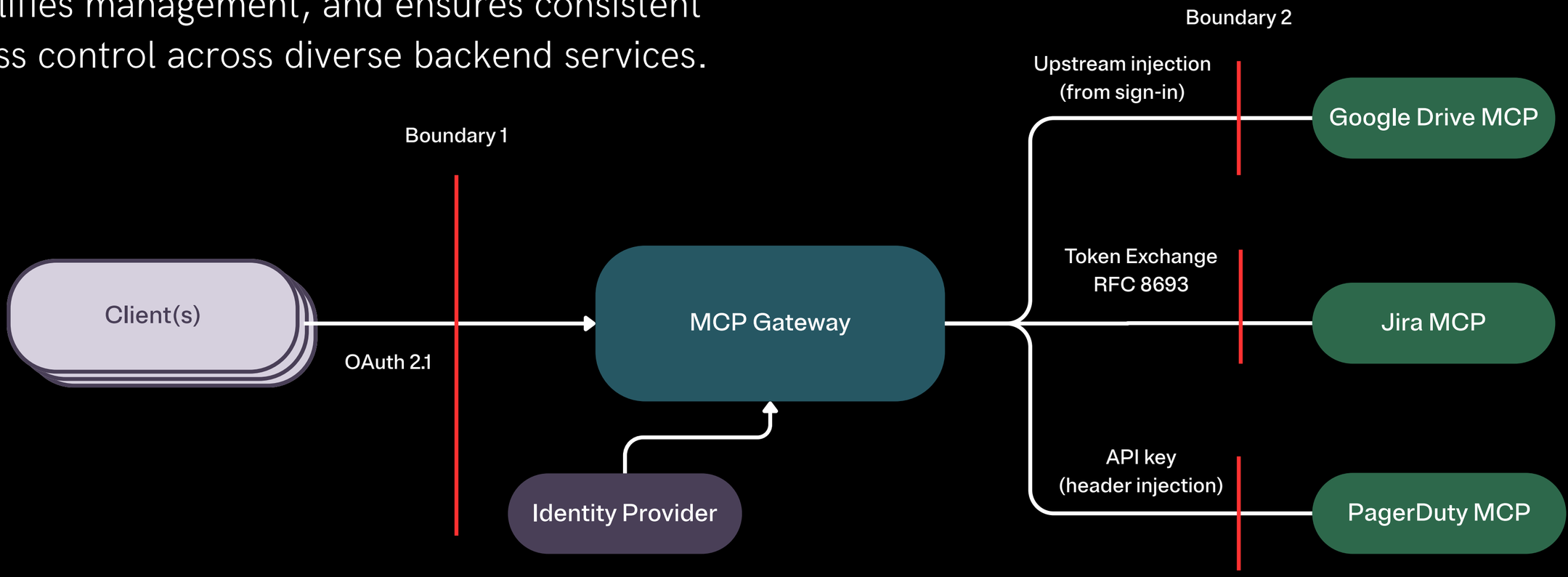
Admin defines exact names. All conflicts must be explicitly resolved.

Priority Approach

Backends ranked by preference. Highest-ranked name wins on collision.

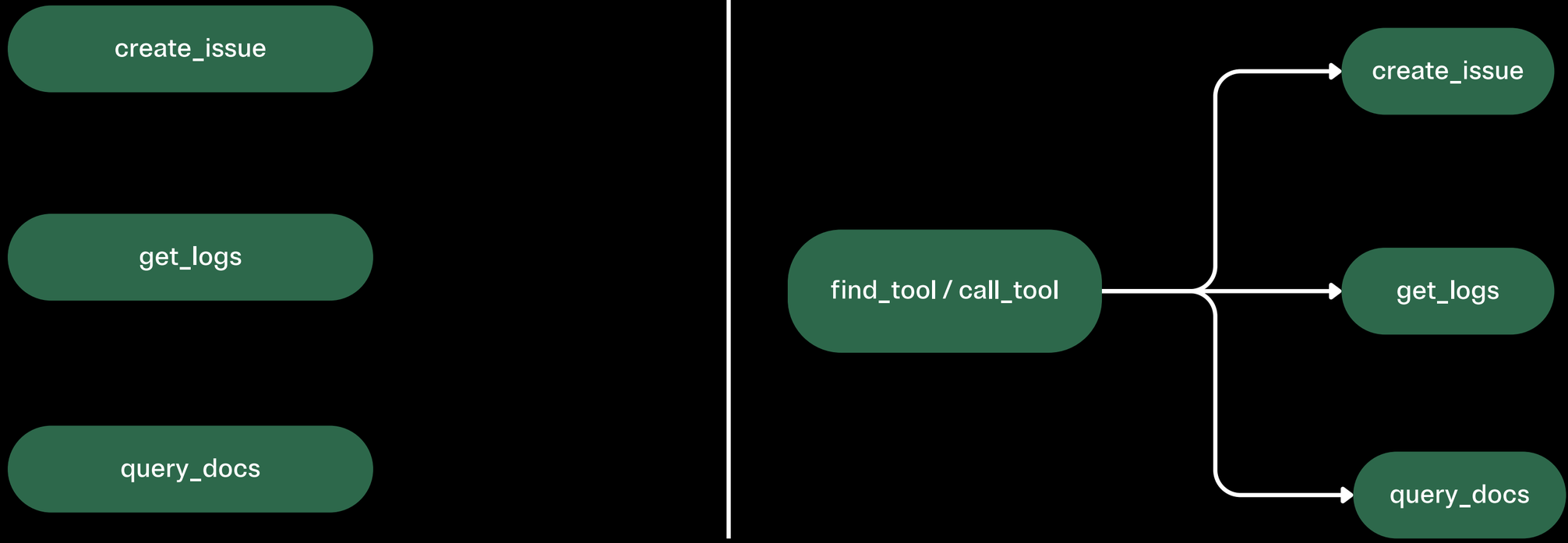
Auth Aggregation

Centralizing authentication enhances security, simplifies management, and ensures consistent access control across diverse backend services.



Meta-Tools

To simplify complexity, expose essential tools like find_tool and call_tool, significantly reducing token usage and enhancing efficiency.



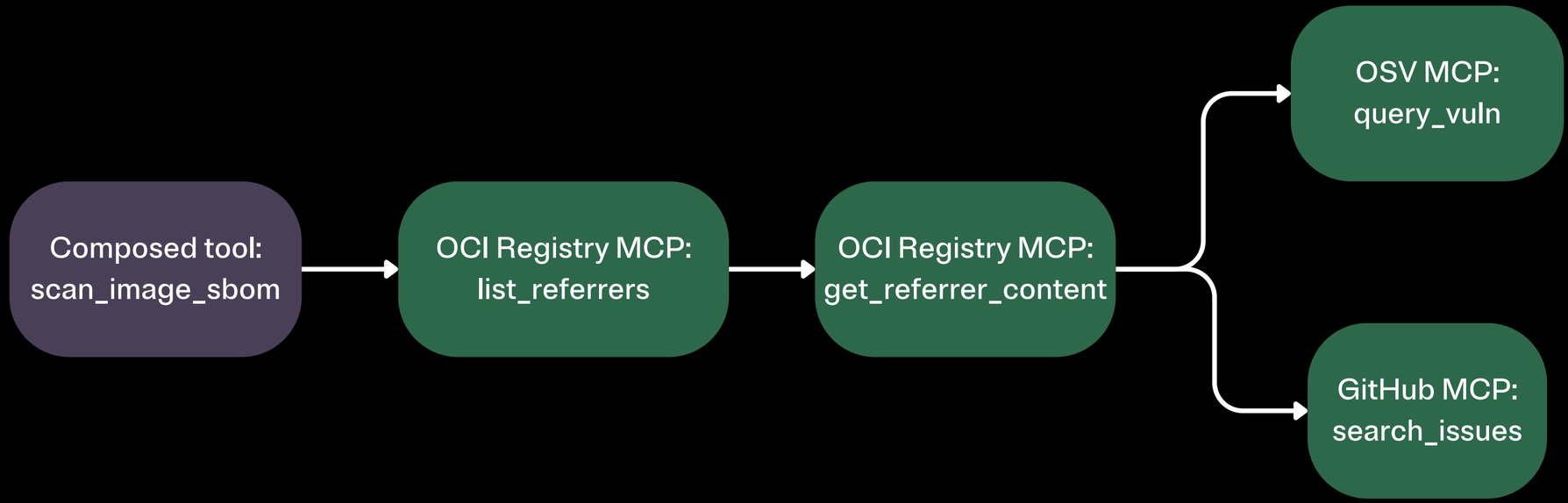
Workflow Control

Your team already knows the workflow.

Unreliable: skips steps, reorders, hallucinates
Slow: sequential LLM round-trips, no parallelism
Expensive : full context re-read per decision
Unauditable: can't prove it ran the same path twice

Workflow Composition

Deterministic workflows ensure reliable execution, allowing for better oversight and **consistent outcomes** in complex orchestration tasks.



Demo

Exploring supply chain security in action

Security Enforcement

A single gateway enforces security policies, providing centralized **auth**, logging, and rate limiting to prevent policy sprawl.

The spec is moving fast!

The pattern is inevitable.
The question is where in your stack.

There is no S in MCP

There is no S in MCP

… But you can add it in front

Resources

STACKLOK

ToolHive @ GitHub

ToolHive @ Discord

Survey