SOLO.IO

Authentication and Authorization Patterns for Agents and MCP

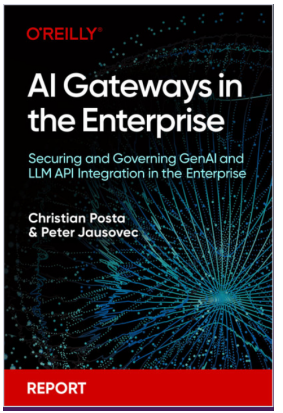
Christian Posta - Global Field CTO, Solo.io
LinkedIn: /in/ceposta
X: @christianposta
GitHub: christian-posta

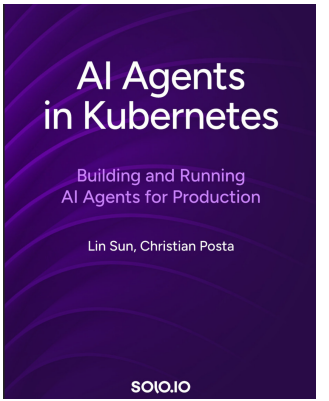
Christian Posta,
Global Field CTO, Solo.io

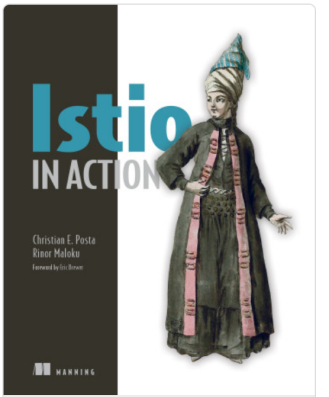
LinkedIn: /in/ceposta

X: @christianposta

GitHub: christian-posta







https://blog.christianposta.com

SOLO.IO

Agenda

* Why Agents and MCP need strong identity

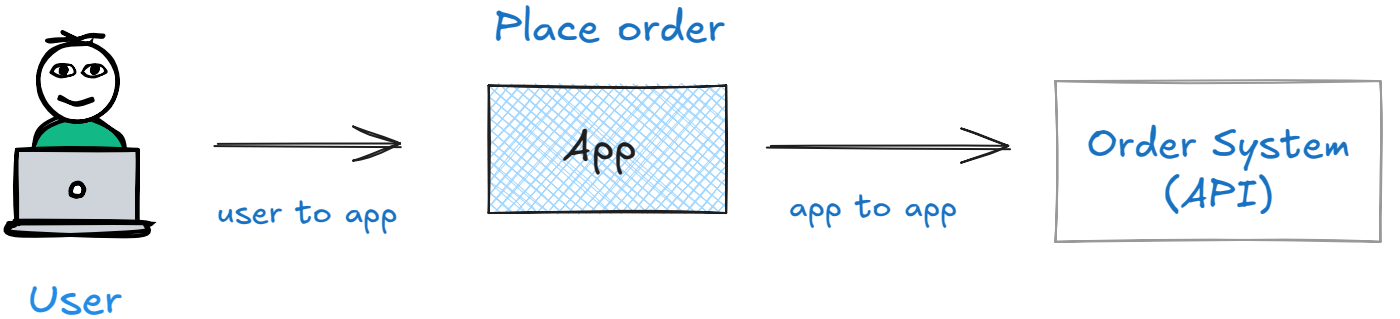
* Challenges with existing mechanisms

* Technologies that can help

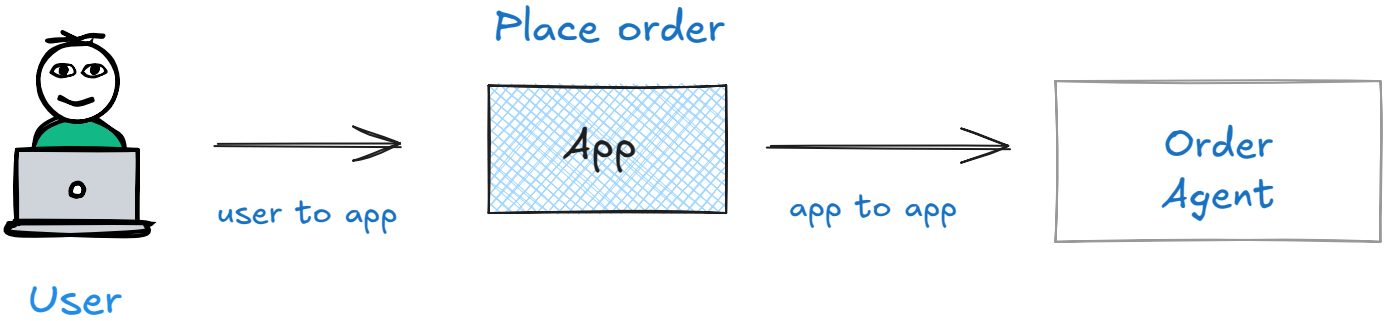
* Agent and MCP auth patterns

* Demos, demos, and more demos

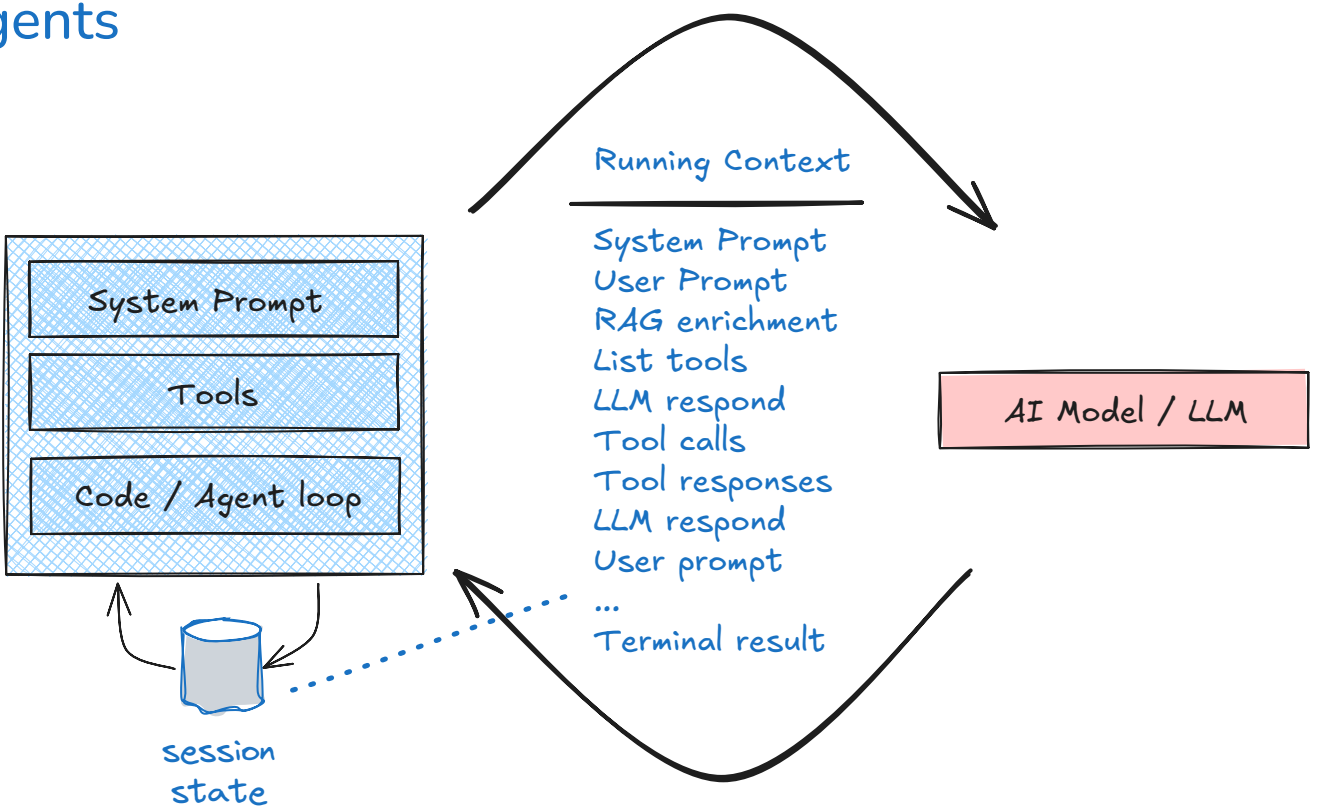
Typical Application Interaction



Are Agents Just APIs?



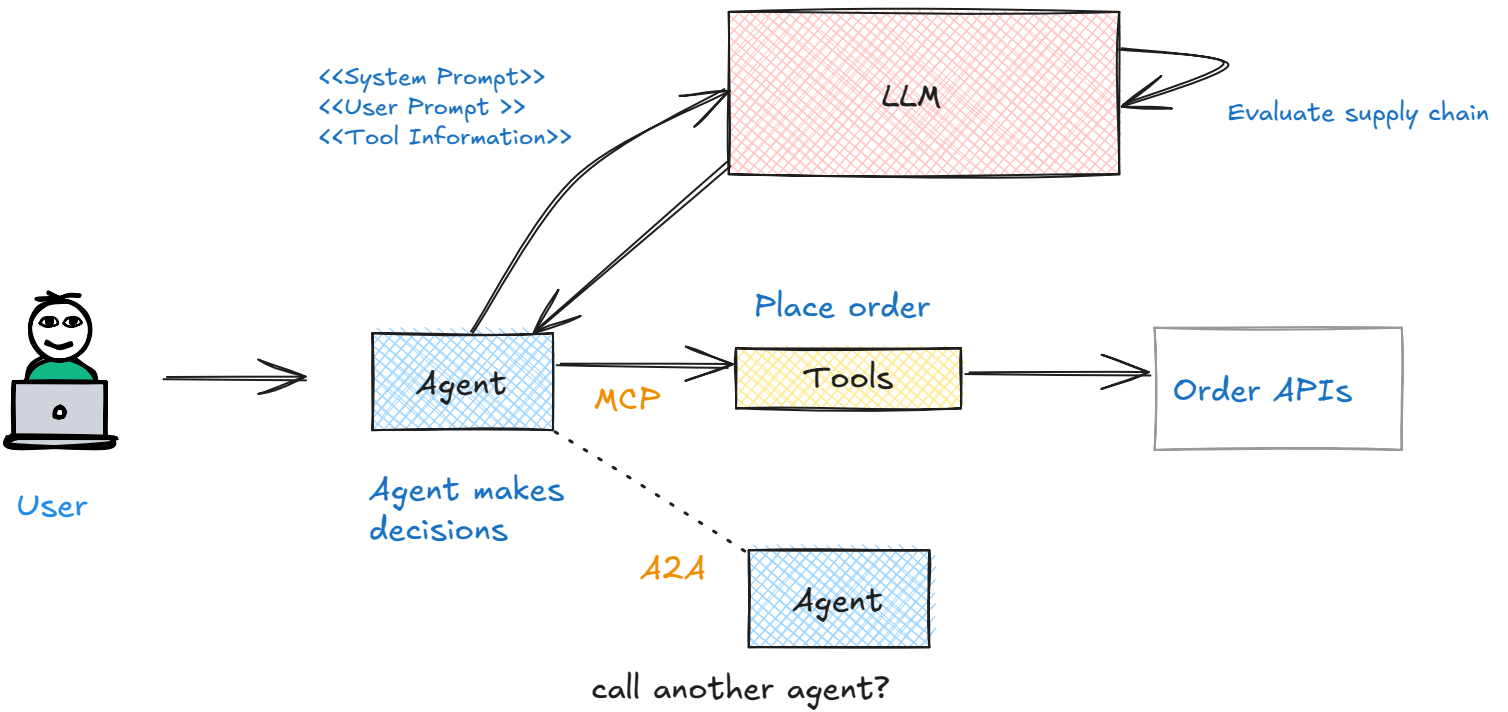
AI Agents


System Prompt
Tools
Code / Agent loop
session state
Running Context
System Prompt
User Prompt
RAG enrichment
List tools
LLM respond
Tool calls
Tool responses
LLM respond
User prompt
...
Terminal result
AI Model / LLM

Difference Between Agents and APIs

- Agents are stateful, hold sessions
- Agents make decisions based on context
- Agents may act autonomously
- Agents behave non-deterministically
- Agents are goal-driven
- Agents are driven by natural language

AI Agents


<<System Prompt>>
<<User Prompt >>
<<Tool Information>>
LLM
Evaluate supply chain
Place order
Agent
MCP
Tools
Order APIs
User
Agent makes decisions
A2A
Agent
call another agent?

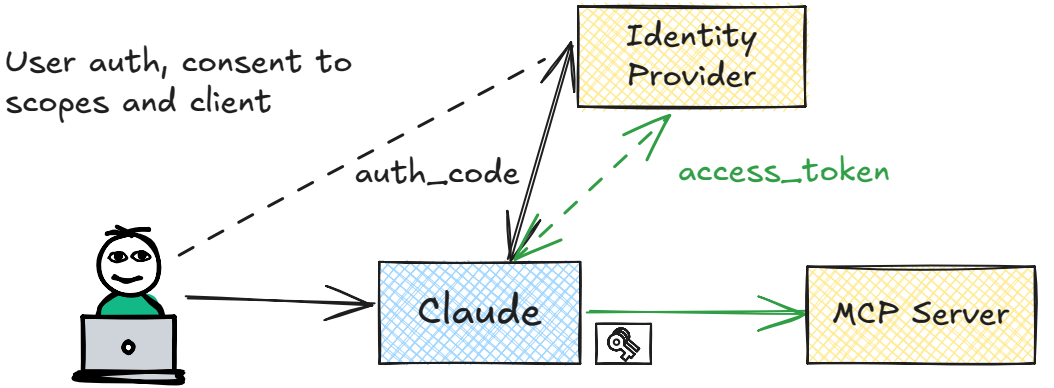
Auth for MCP and Agents

Auth Mechanisms Per Spec

- OAuth 2.1

- OAuth 2.0 / OIDC
- API Key
- mTLS
- HTTP Basic

OAuth

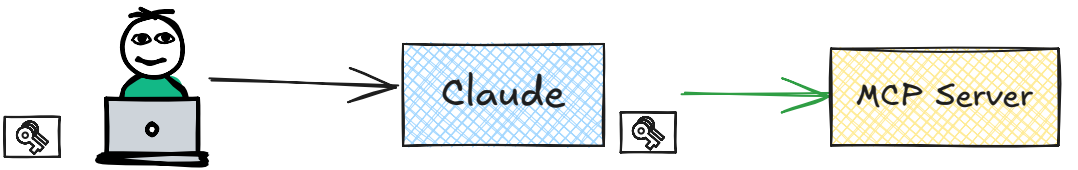


OAuth for Enterprise?

- enterprise knows users, tools, agents
- "bearer" token semantics
- scopes are too broad
- permissions are static
- no context, intent
- enterprise cares about ~~policy~~

API Key

User acquires an API key
configures API key on Agent



Agent sends API key

API Key for enterprise?

- API keys were not intended for Auth
- "bearer" token semantics, dangerous
- permissions tied to static RBAC
- usually long-lived, rarely rotated

Distill Down the Real Enterprise Auth Challenges

- Coarse grained permission models

- Bearer tokens, no proof of possession

- Static / long lived credentials

- No context / intent / capability binding (e.g., tenant, data class, risk, etc.).

- Policy bolted on, in code, very inconsistent

Agent Permissions Example

System / API	Role	Key Permissions
 **Oracle HCM**	Integration Specialist	• Read employee personal info • View compensation data • Retrieve org structures • View benefit enrollments
 **OpenText Content Server**	Integration User	• See (Level 1) • See Contents (Level 2) • Read (Level 3) • Add (Level 4) • Delete (Level 5)
 **ServiceNow ITSM**	Integration User	• Create, read, update incidents • Assign to self/others • Categorize/prioritize • Resolve/close incidents

Agent Permissions Example

> 👨‍💼 **Employee request:** “What’s our company’s work from home policy?”
> 🤖 **Agent action:** Uses the *Document Management System* API key to retrieve the current WFH policy document from the HR policies folder.

> 👨‍💼 **Employee request:** “I need to update my home address.”
> 🤖 **Agent action:** Uses the *Ticketing System* API key to create a ticket for the address change and the *Document Management System* API key to retrieve the appropriate form.

> 👨‍💼 **Employee request:** “How many vacation days do I have left this year?”
> 🤖 **Agent action:** Uses the *HR Management System* API key to query only that specific employee’s time-off balance.

Agent Permissions Example

🚨 **Unauthorized Compensation Access**

👨‍💼 **Employee request:** “What’s the average salary for people in my position?”
🤖 **Agent action:** Uses the *HR Management System* API key to query salary data across the entire department or company, accessing compensation information for multiple employees.
⚠️ **Why it’s problematic:** The HR Service Account API key has blanket read permissions to all compensation data, allowing the agent to retrieve and analyze sensitive salary information that would normally require manager-level access.

🚨 **Sensitive Document Exposure**

👨‍💼 **Employee request:** “Are there any reorganization plans for my department?”
🤖 **Agent action:** Uses the *Document Management System* API key to search all HR documents, including confidential executive planning documents labeled as “reorganization.”
⚠️ **Why it’s problematic:** The HR Documents Service Role API key can access highly confidential documents that aren’t meant for general employee consumption, potentially revealing sensitive business plans.

Principals of Agent IAM

- Strong identity for users and agents
- JIT, context/capability tokens issued via policy, bound to identities, agents
- Short lived, frequently rotated
- Enforced by policy evaluation using context
- Least privilege / zero trust posture
- Organized in central systems that can be audited, applied consistently

Laying the Foundation

Guidance for Enterprise Auth Patterns

- Tie into enterprise/federated SSO for user auth

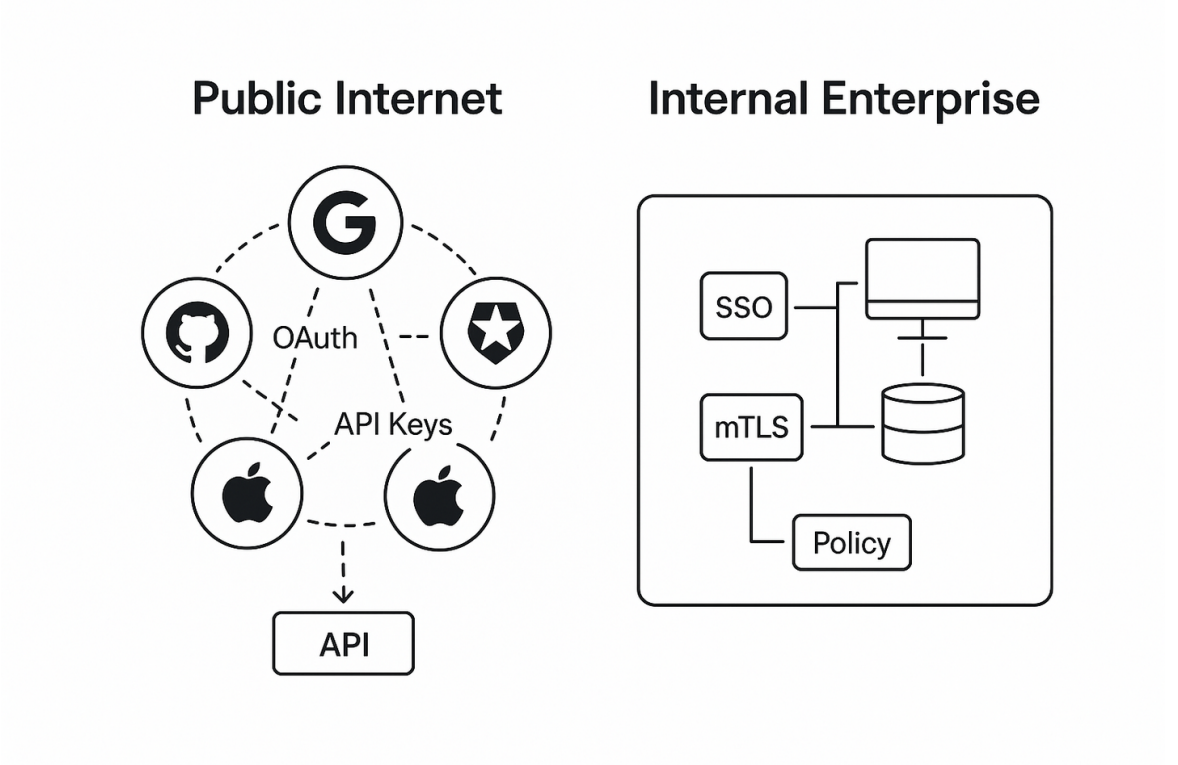
- Establish non-human workload/agent identity

- Use mTLS for mutual authentication everywhere

- Use a delegation / on-behalf-of mechanism

- Use centralized policy-driven authorization (ABAC/ReBAC)

Enterprise vs Public



Identity Based on Attestation

spiffe

https://spiffe.io

SPIFFE ID:

spiffe://acme.com/finance/finance-reporting-agent

trust domain

custom path

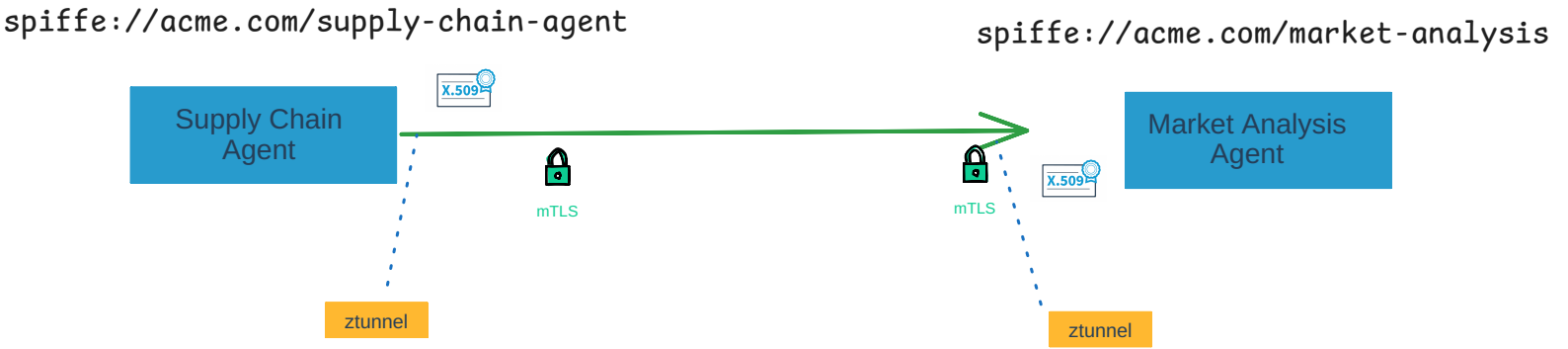
SPIFFE Verifiable Document (SVID):

spiffe://acme.com/finance/finance-reporting-agent


```
{
"sub": "spiffe://acme.com/finance/finance-reporting-agent",
"iss": "spiffe:issuer",
"aud": "spiffe:audience",
"iat": 1689000000,
"exp": 1689003600
}
```

JWT

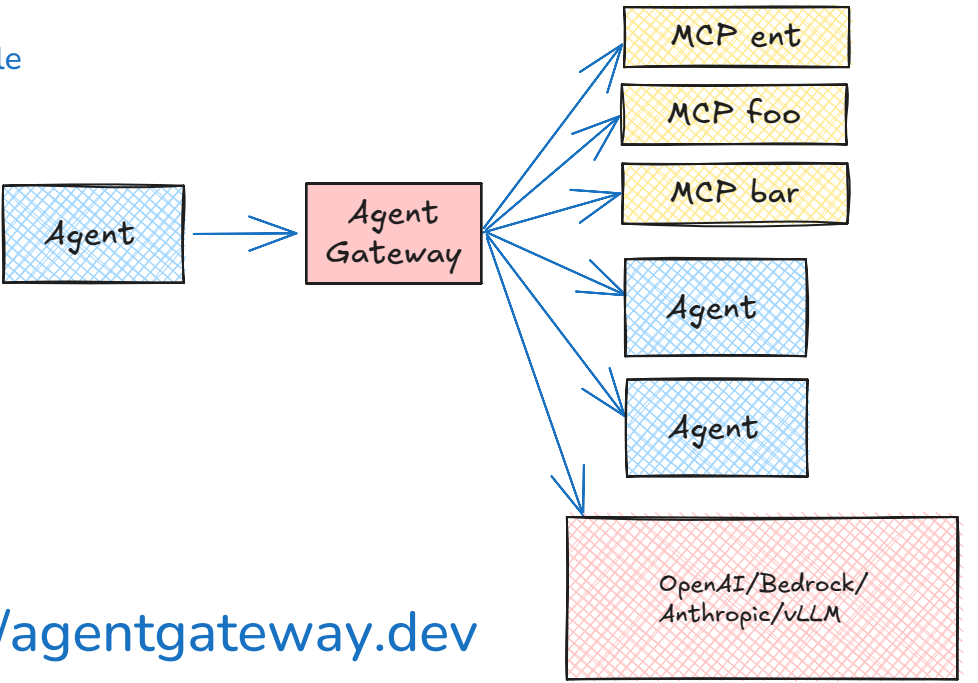
Secure mTLS Tunnels with SPIFFE



https://ambientmesh.io

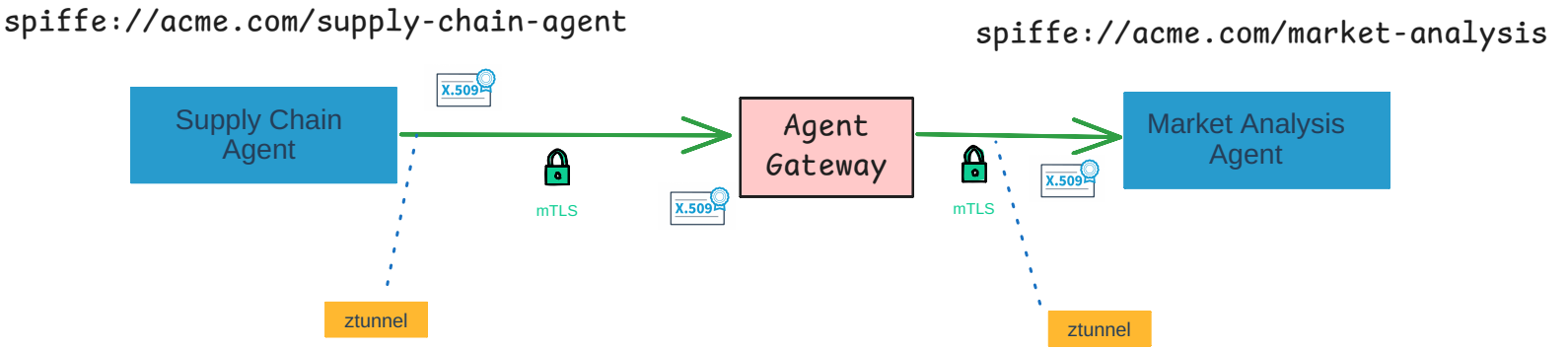
Agent Gateway for LLM, MCP, and A2A

* Kubernetes Gateway API compatible
* MCP multiplexing / proxying
* A2A routing
* JWT handling
* Downscope tokens
* Auth (user + identity)
* Call out to policy engines
* Unified API for LLM Calls
* Inference gateway routing



https://agentgateway.dev

Agent Gateway with Ambient Mesh



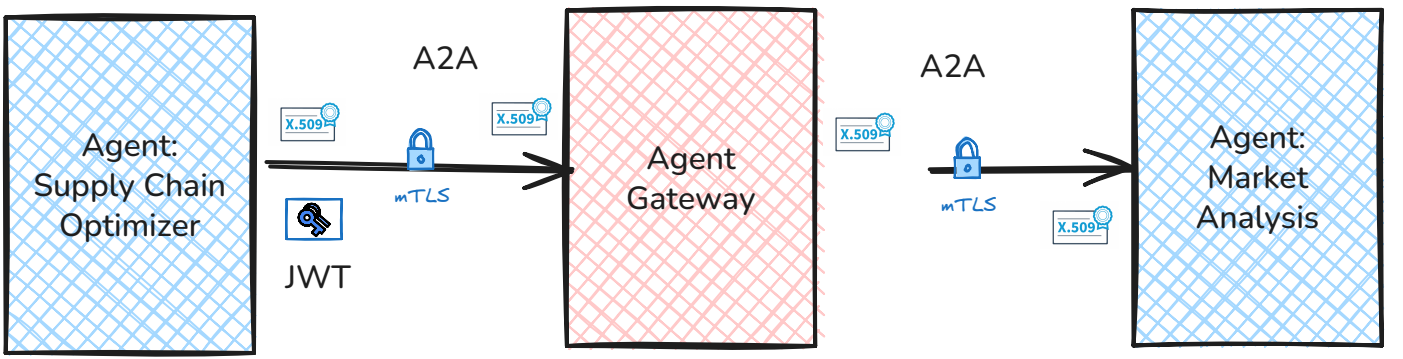
Auth Patterns

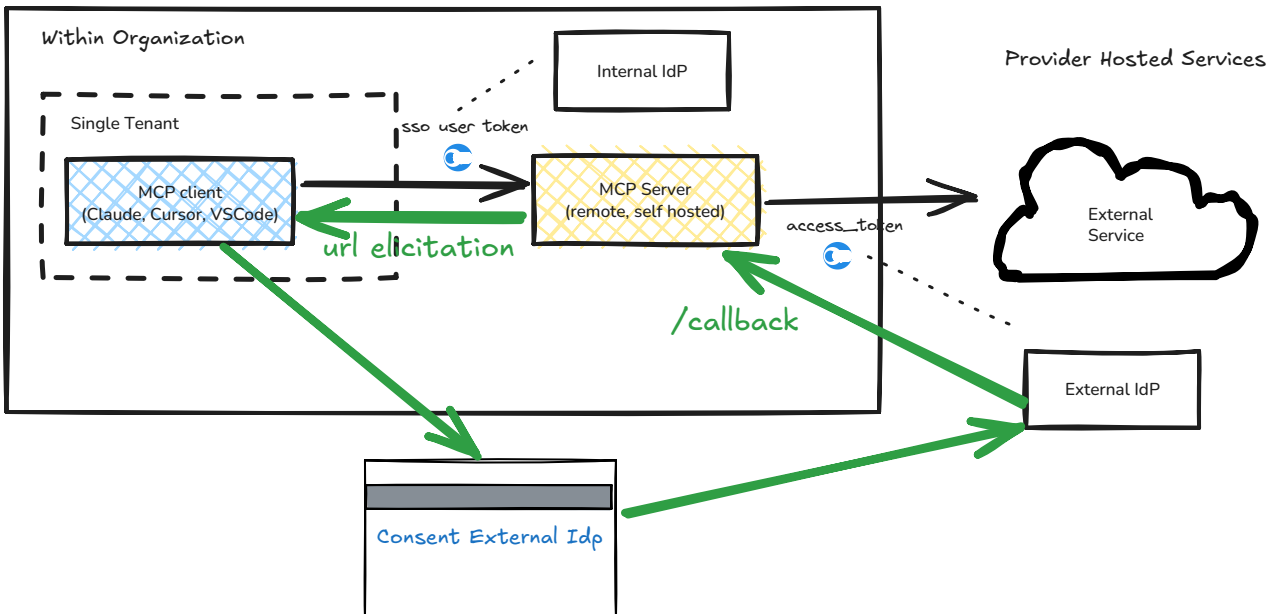
SOLO.IO

Patterns for MCP and AI Agents

- Agent to Agent (Internal)
- Agent to MCP server (Internal)
- Agent to MCP (internal) --> API (external)
- Agent (external) to MCP (internal)

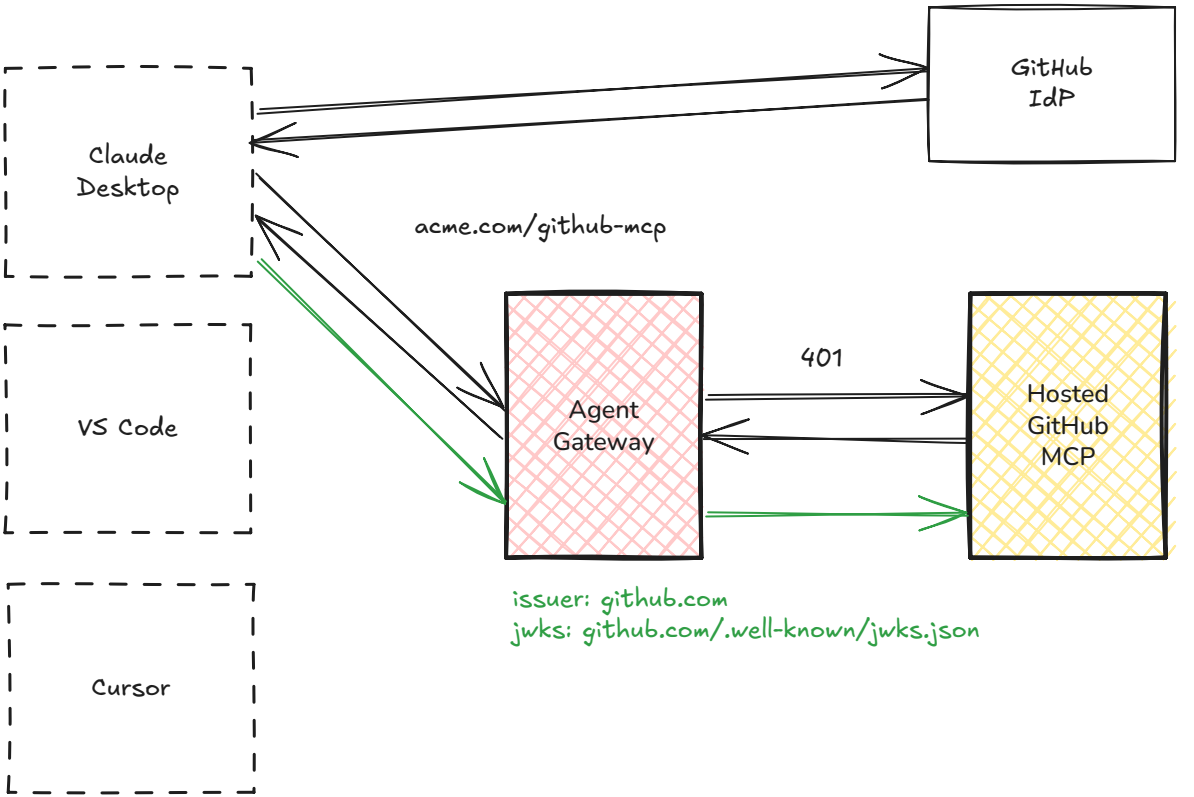
Agent to Agent (internal)




Within Organization
Internal IdP
Provider Hosted Services
Single Tenant
sso user token
MCP client
(Claude, Cursor, VSCode)
MCP Server
(remote, self hosted)
access_token
External
Service
url elicitation
/callback
External IdP
Consent External Idp

Agent to MCP (internal) to API (external)

Agent External to MCP (internal)



Demo

Thank you!

Additional Resources:

* ambientmesh.io
* agentgateway.dev
* kagent.dev
* kgateway.dev
* solo.io/blog
* blog.christianposta.com

Christian Posta
LinkedIn: /in/ceposta
X: @christianposta
GitHub: christian-posta

SOLO.IO