HITACHI

KeyConf 25
@Van der Valk Hotel Amsterdam Zuidas - RAI, Amsterdam, Netherlands 🇳🇱

Keycloak meets AI: the possibility of integrating Keycloak with AI

Takashi Norimatsu
Open Source Program Office (OSPO)
Hitachi, Ltd.

Date
August 28, 2025

Self Introduction

Takashi Norimatsu Ph.D. (**tnorimat** in GitHub)

- Keycloak maintainer (since Oct 2021),
- Technical lead of Keycloak community “OAuth SIG”,
- Certified Information Systems Security Professional (CISSP)
- Senior OSS Specialist, Hitachi, Ltd., 🇯🇵

Contributing security features to Keycloak since 2017:

- W3C Web Authentication API support (Passkeys)
- Security features support
  e.g., RFC 7636 PKCE, RFC 8705 OAuth MTLS, RFC 9126 PAR, RFC 8032/8037 EdDSA,
  RFC 9207 OAuth2 Authz Server Issuer Identification, RFC 9449 DPoP, OIDC CIBA
- API security profiles support
  e.g., FAPI 1.0 Baseline, FAPI 1.0 Advanced, FAPI-CIBA,
  FAPI 2.0 Security Profile, FAPI 2.0 Message Signing,
  OAuth 2.1 for Confidential Client, OAuth 2.1 for Public Client

Contents

1. How can Keycloak be used in the context of AI?
2. Model Context Protocol (MCP) Overview
3. Authentication and Authorization in MCP
4. Keycloak in MCP

Contents

1. How can Keycloak be used in the context of AI?
2. Model Context Protocol (MCP) Overview
3. Authentication and Authorization in MCP
4. Keycloak in MCP

What AI is…

AI includes ML (Machine Learning)

ML includes Deep Learning

Deep Learning includes **Generative AI (Gen AI)**

focused in the talk

How can Keycloak be used in the context of AI?

Keycloak leverages Gen AI for … Adaptive authentication.

Ex.

KeyConf 24 Vienna:

https://youtu.be/0zWlc08CPuo?si=Od-opD4AG0V1lQto

https://keyconf.dev/2024//assets/files/Martin_Bartos-KeyConf_Adaptive_Authentication-final.pdf

Keycloak DevDay 2025 Darmstadt:

https://drive.google.com/file/d/12-vAuVmWqUb3581D8WqWq0uutLbH7tsn/view?usp=sharing

Keycloak leverages Gen AI for … Managing Keycloak by natural language.

Ex.

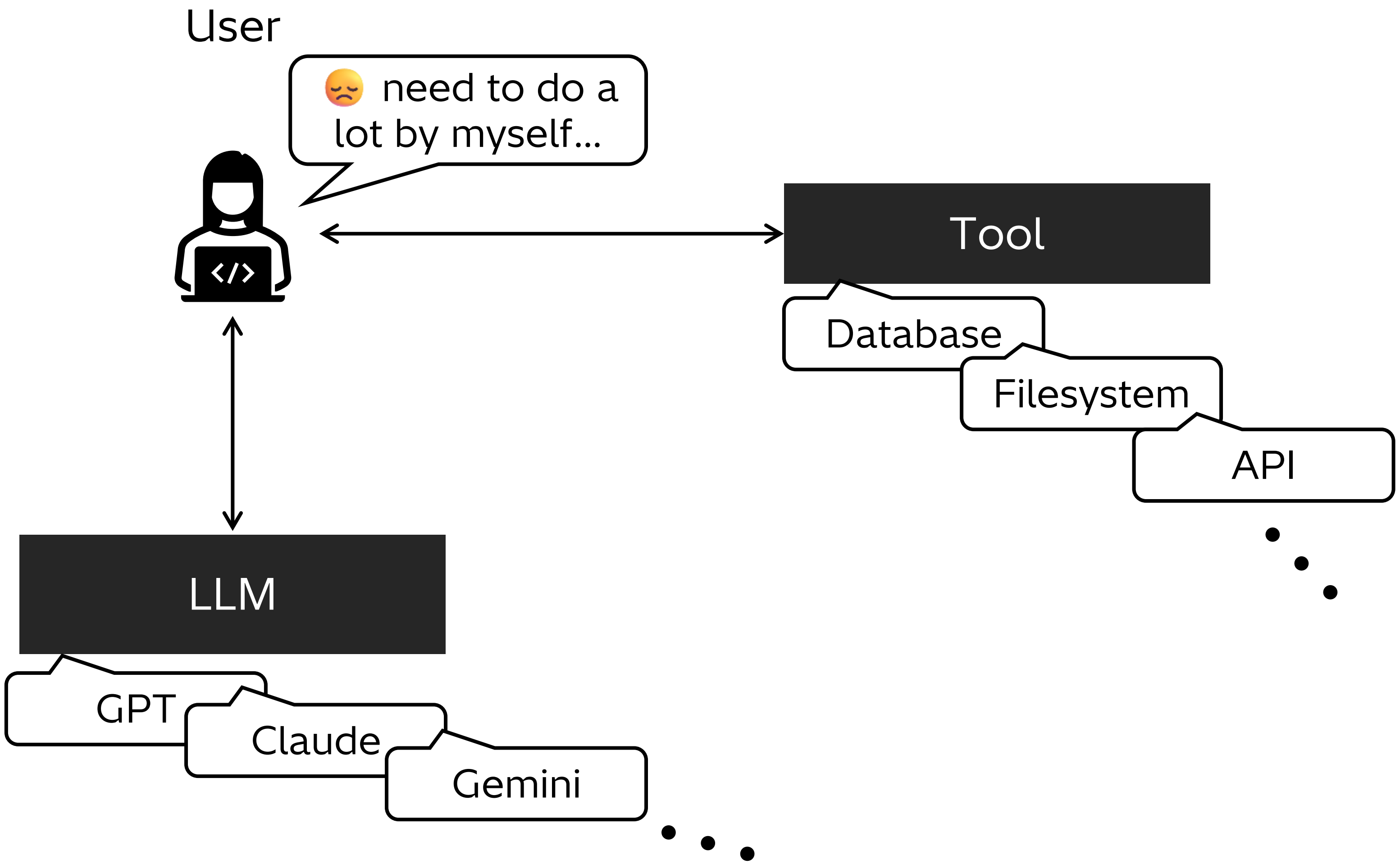
https://www.youtube.com/watch?v=umwpMYLwxXI

https://www.linkedin.com/posts/embesozzi_keycloak-llms-ai-activity-7325126855794036737-Gjut

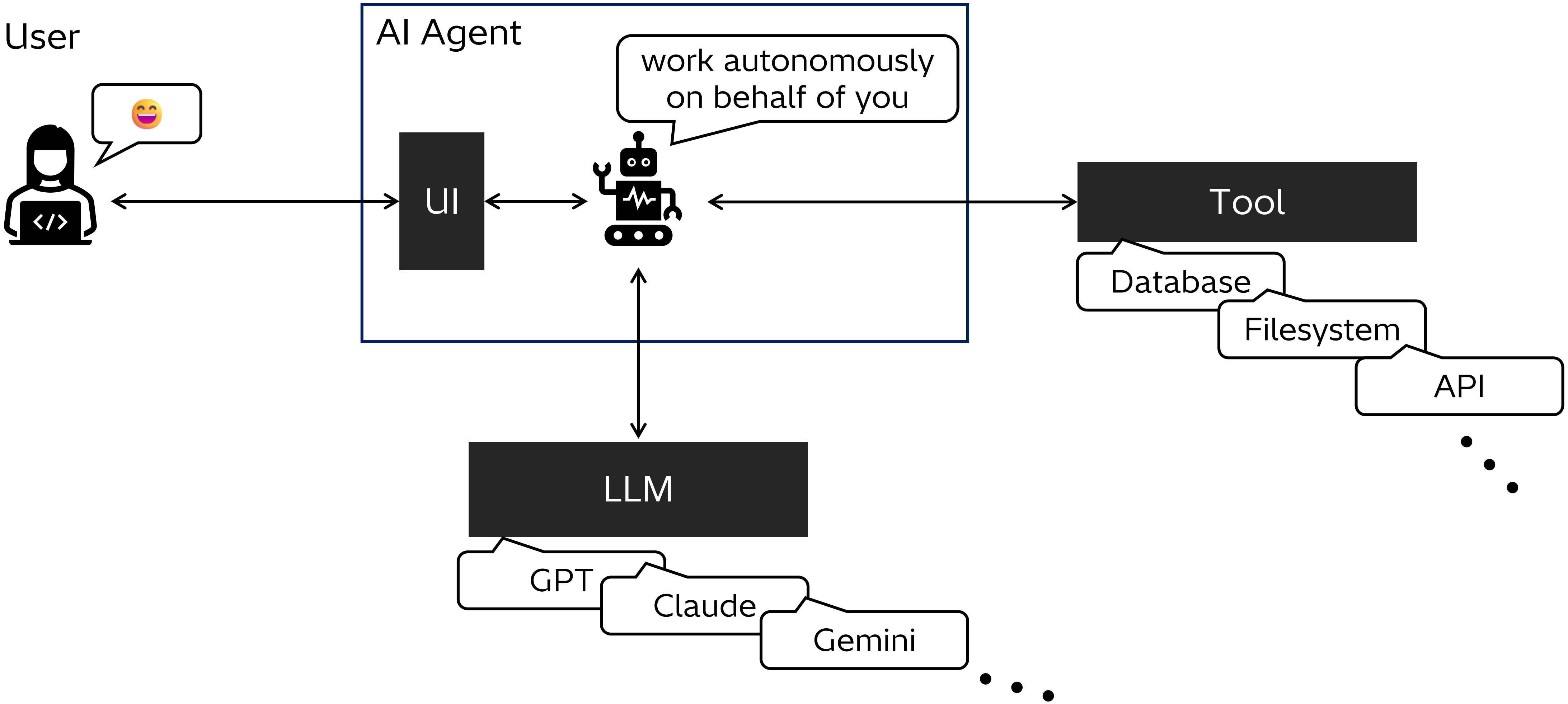
Keycloak is used as an authorization server by an AI agent using Gen AIs and external tools.

focused in the talk

AI Agent Overview



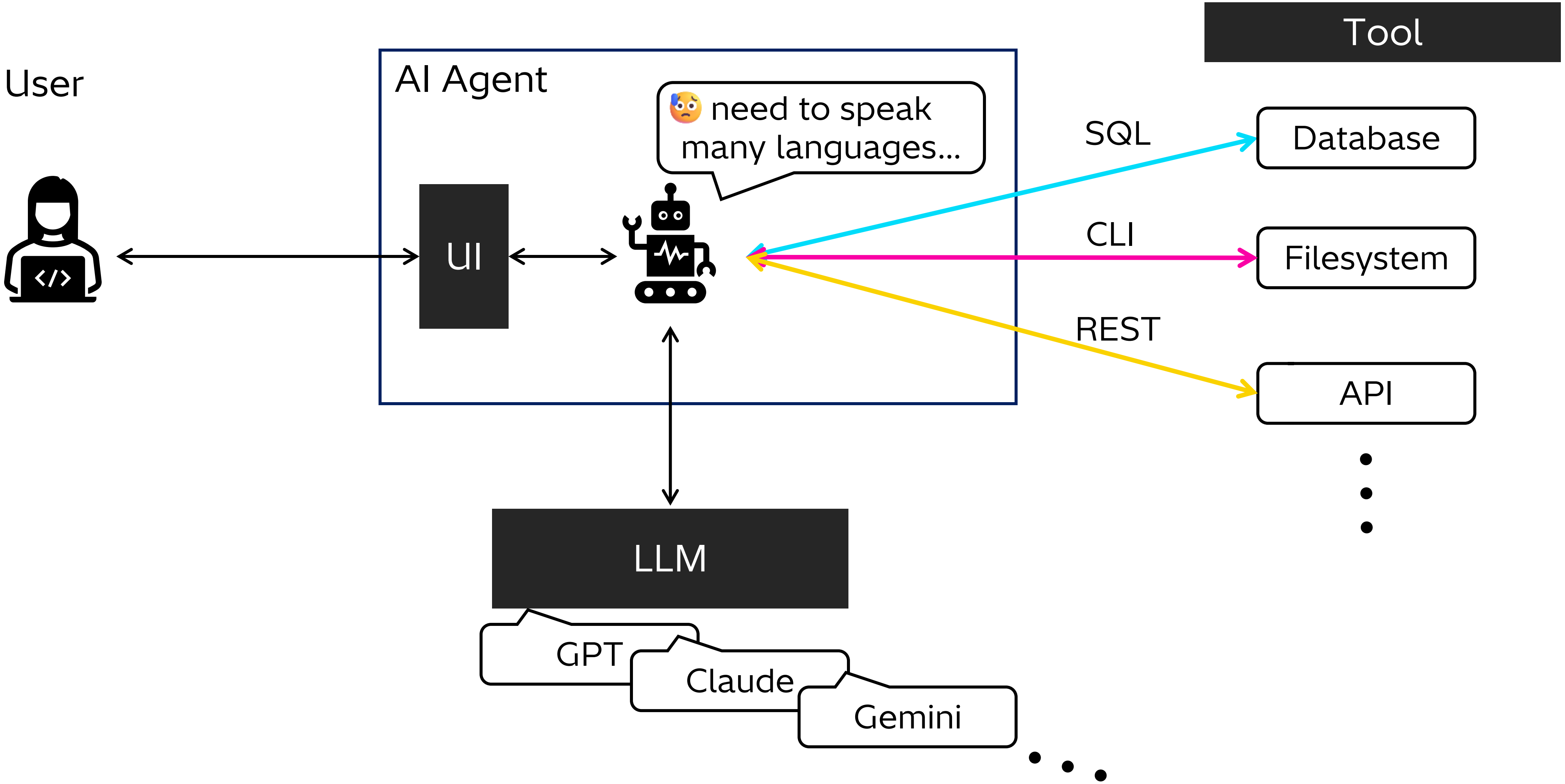
AI Agent Overview



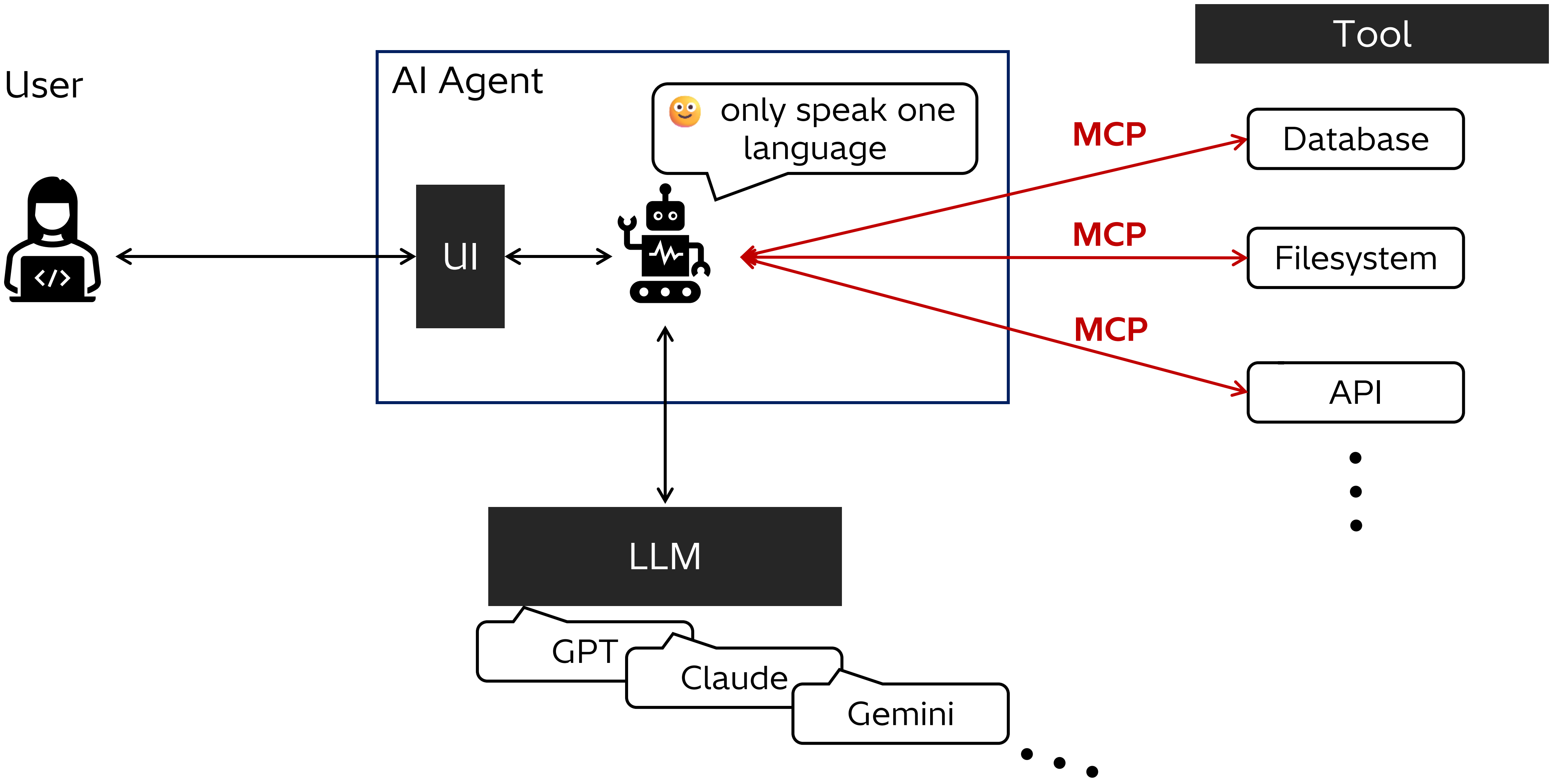
Contents

1. How can Keycloak be used in the context of AI?
2. Model Context Protocol (MCP) Overview
3. Authentication and Authorization in MCP
4. Keycloak in MCP

Model Context Protocol (MCP) Overview



Model Context Protocol (MCP) Overview



Model Context Protocol (MCP) Overview

Protocol Revision:

- 2024-11-05
- 2025-03-26
- **2025-06-18 (latest)**
- Draft (in progress)

This talk is based on this version.

Architecture: Client-Server model
(MCP Client, MCP Server)
Message Format: JSON-RPC 2.0
Connection: Stateful, negotiable

MCP Client:

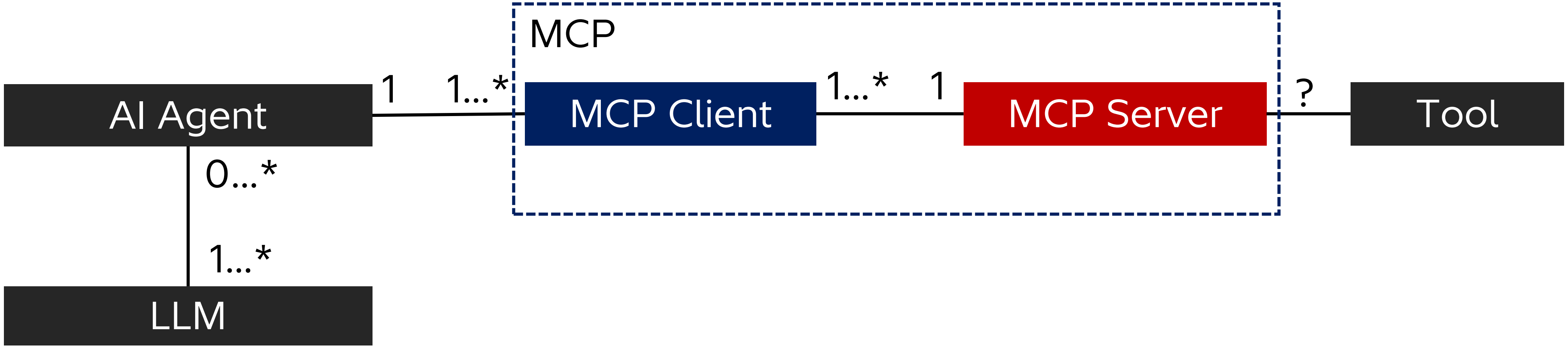
- Transport: STDIO, Streamable HTTP
- Features: Roots, Sampling, Elicitation

MCP Server:

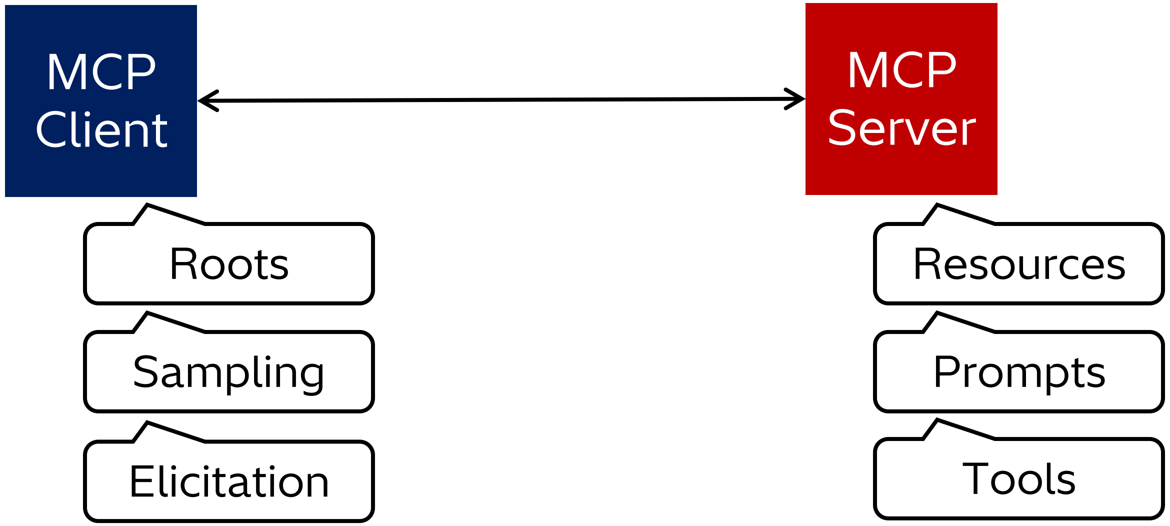
- Transport: STDIO, Streamable HTTP
- Features: Resources, Prompts, Tools,

HITACHI

Model Context Protocol (MCP) Overview



Features



- Single endpoint (Ex. https://example.com/mcp)
- Support GET and POST

Features

MCP Client triggered (Client → Server)

- Resources
  - resources/list
  - resources/read
  - resources/templates/list
  - resources/subscribe
- Prompts
  - prompts/list
  - prompts/get
- Tools
  - tools/list
  - tools/call
- Roots
  - notifications/roots/list_changed
- Initialization
  - initialize
  - notifications/initialized
- Cancellation
  - notifications/cancelled
- Completion
  - completion/complete
- Logging
  - logging/setLevel

Features

MCP Server triggered (Server → Client)

- Roots
  - roots/list
- Sampling
  - sampling/createMessage
- Elicitation
  - elicitation/create

- Resources
  - notifications/resources/list_changed
  - notifications/resources/updated
- Prompts
  - notifications/prompts/list_changed
- Tools
  - notifications/tools/list_changed
- Logging
  - notifications/message

Features

Both MCP Client and Server triggered
(Client → Server, Server → Client)

- Health Check
  - ping
- Progress
  - notifications/progress

HITACHI

Contents

1. How can Keycloak be used in the context of AI?
2. Model Context Protocol (MCP) Overview
3. **Authentication and Authorization in MCP**
4. Keycloak in MCP

MCP Deployment Patterns

#	MCP Client runs on	MCP Server runs on	Tool runs on	Tool requires AuthN & AuthZ	AuthN & AuthZ Credential Example	Tool Example
1	Local	Local	Local	No	-	Filesystem[*1]
2	Local	Local	Local	Yes	Password	Database[*2]
3	Local	Local	Remote	No	-	Server publicly available[*3]
4	Local	Local	Remote	Yes	API key, Personal Access Token(PAT)	Server serving multiple users[*4]
5	Local	Remote	Remote	No	-	Server publicly available
6	Local	Remote	Remote	Yes	OAuth access token	Server serving multiple users[*5]

*1 : **File system** - https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem

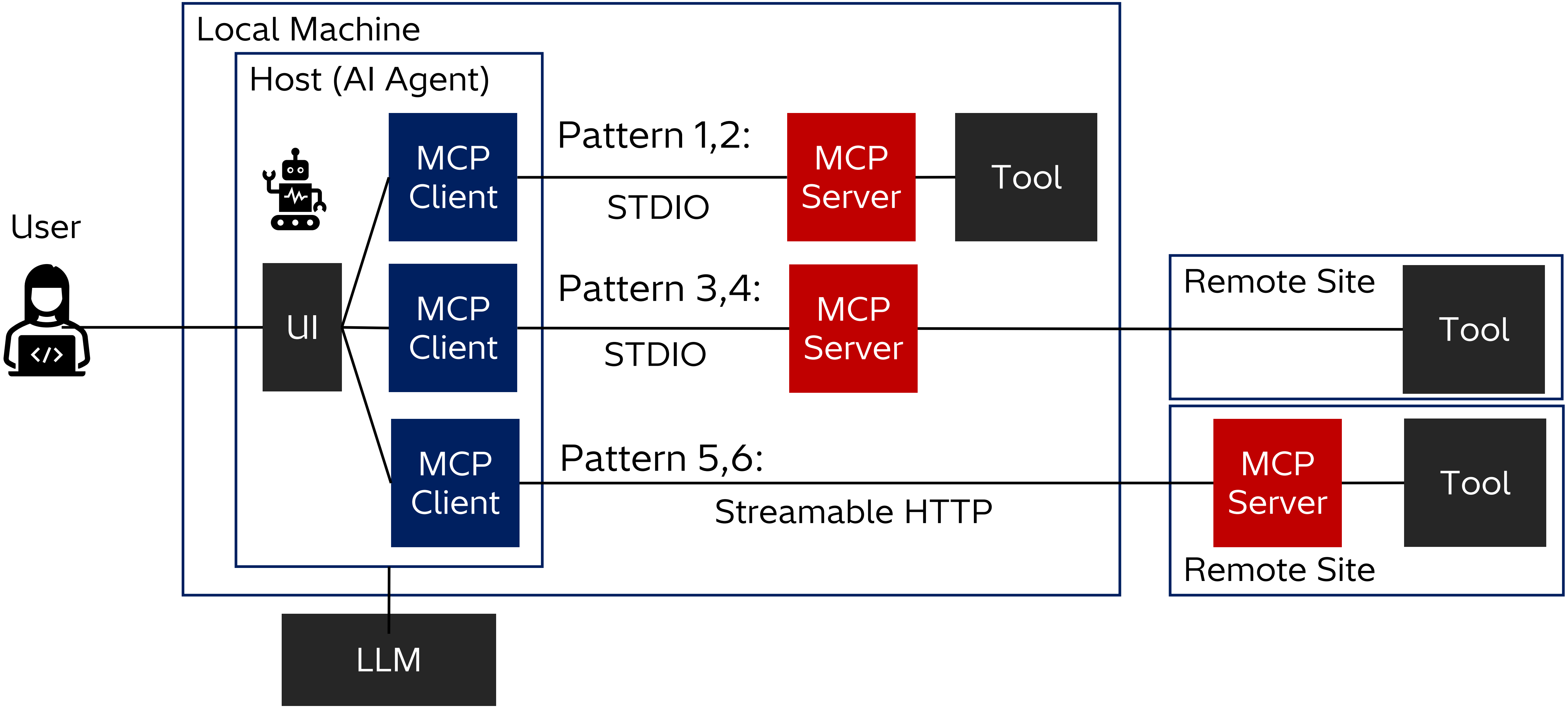
*2: **SQLite, SQL Server, PostgreSQL, and MySQL** - https://github.com/executeautomation/mcp-database-server

*3: **Open-Meteo** - https://github.com/isdaniel/mcp_weather_server

*4: **GitHub** - https://github.com/github/github-mcp-server

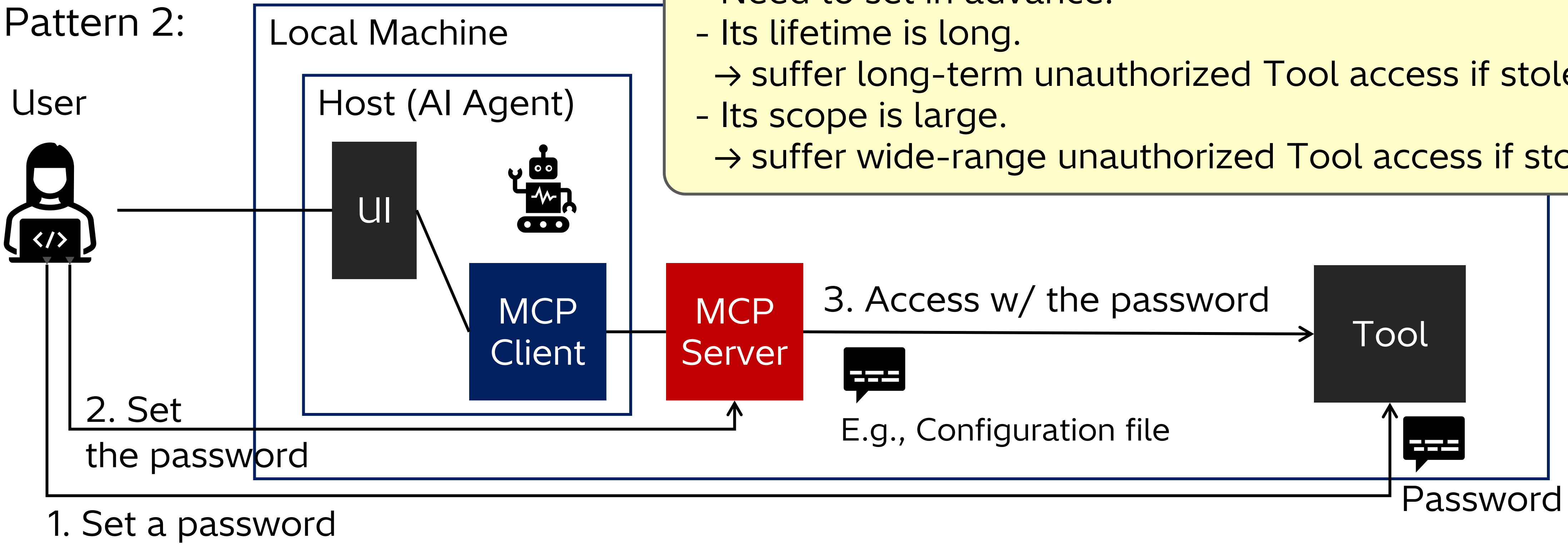
*5: **GitHub** - https://github.blog/changelog/2025-06-12-remote-github-mcp-server-is-now-available-in-public-preview

MCP Deployment Patterns



Obtaining AuthN & AuthZ credential – Out-of-band

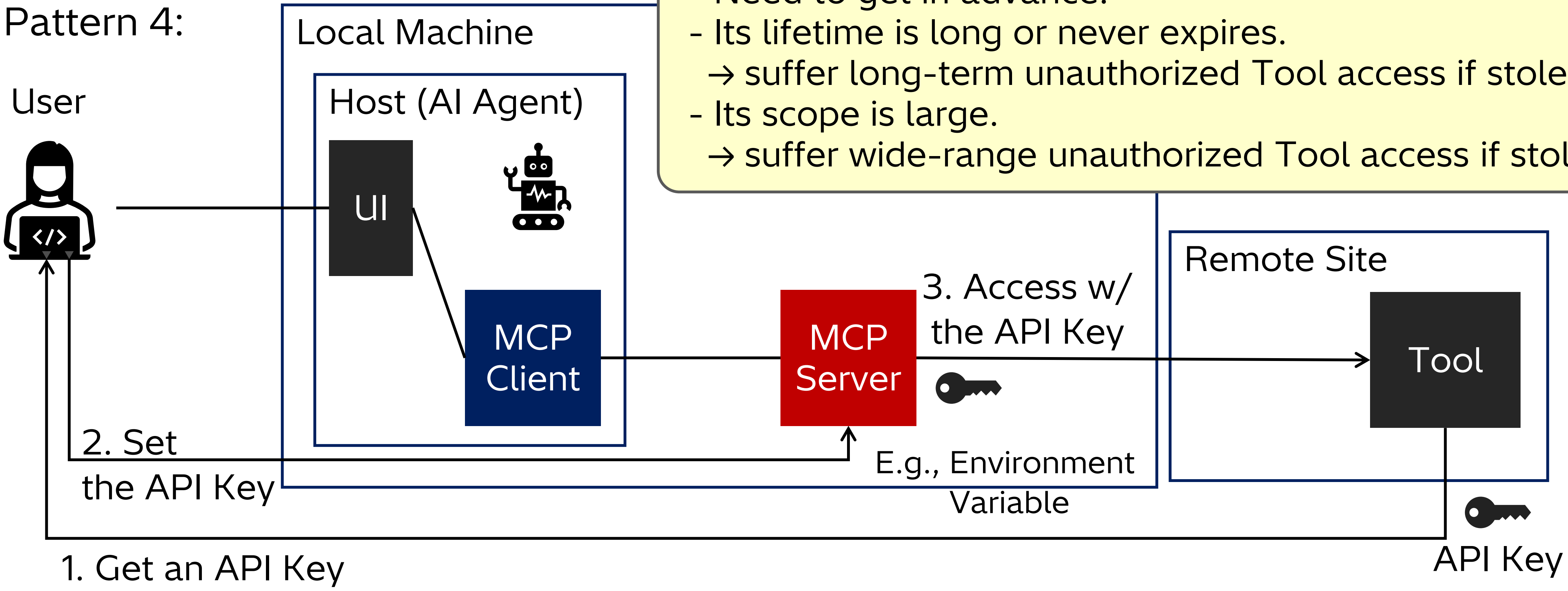
AuthN & AuthZ credential - **Password**:
- Need to set in advance.
- Its lifetime is long.
  → suffer long-term unauthorized Tool access if stolen.
- Its scope is large.
  → suffer wide-range unauthorized Tool access if stolen.



Obtaining AuthN & AuthZ credential – Out-of-band

AuthN & AuthZ credential - **API key**:
- Need to get in advance.
- Its lifetime is long or never expires.
  → suffer long-term unauthorized Tool access if stolen.
- Its scope is large.
  → suffer wide-range unauthorized Tool access if stolen.

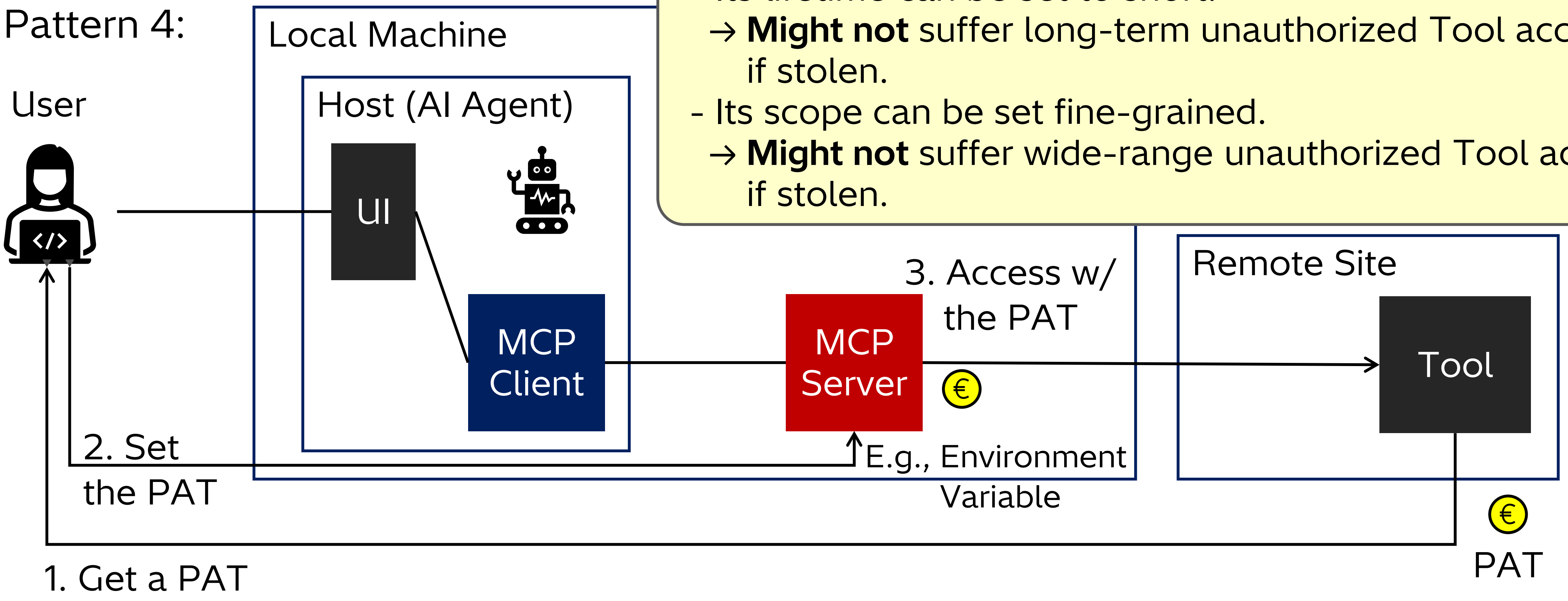
Pattern 4:



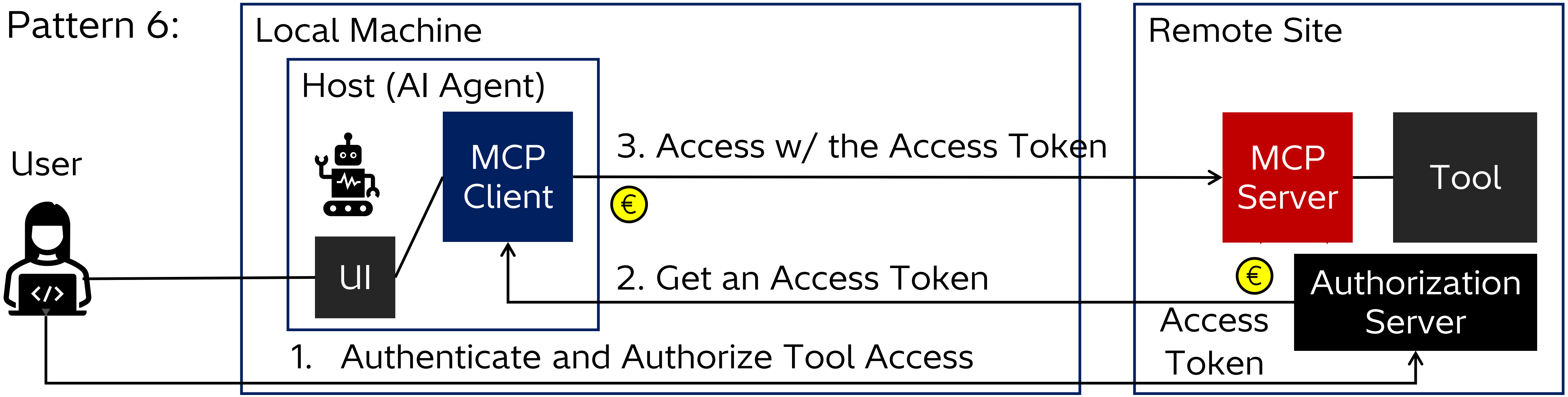
Obtaining AuthN & AuthZ credential – Out-of-band

AuthN & AuthZ credential - **PAT**:

- Need to get in advance.
- Its lifetime can be set to short.
  → **Might not** suffer long-term unauthorized Tool access if stolen.
- Its scope can be set fine-grained.
  → **Might not** suffer wide-range unauthorized Tool access if stolen.



Obtaining AuthN & AuthZ credential – On-the-fly



AuthN & AuthZ credential - **Access token**:

- Need **NOT** to get in advance.
- Its lifetime is very short.
  → **NOT** suffer long-term unauthorized Tool access if stolen.
- Its scope is very narrow.
  → **NOT** suffer wide-range unauthorized Tool access if stolen.

MCP Deployment Patterns

#	MCP Client runs on	MCP Server runs on	Tool runs on	Tool requires AuthN & AuthZ	AuthN & AuthZ Credential Example	Tool Example
1	Local	Local	Local	No	-	Filesystem[*1]
2	Local	Local	Local	Yes	Password	Database[*2]
3	Local	Local	Remote	No	-	Server publicly available[*3]
4	Local	Local	Remote	Yes	API key, Personal Access Token(PAT)	Server serving multiple users[*4]
5	Local	Remote	Remote	No	-	Server publicly available
6	Local	Remote	Remote	Yes	OAuth access token	Server serving multiple users[*5]

*1 : **File system** - https://github.com/modelcontextprotocol/servers/tree/main/src/filesystem

*2: **SQLite, SQL Server, PostgreSQL, and MySQL** - https://github.com/executeautomation/mcp-database-server

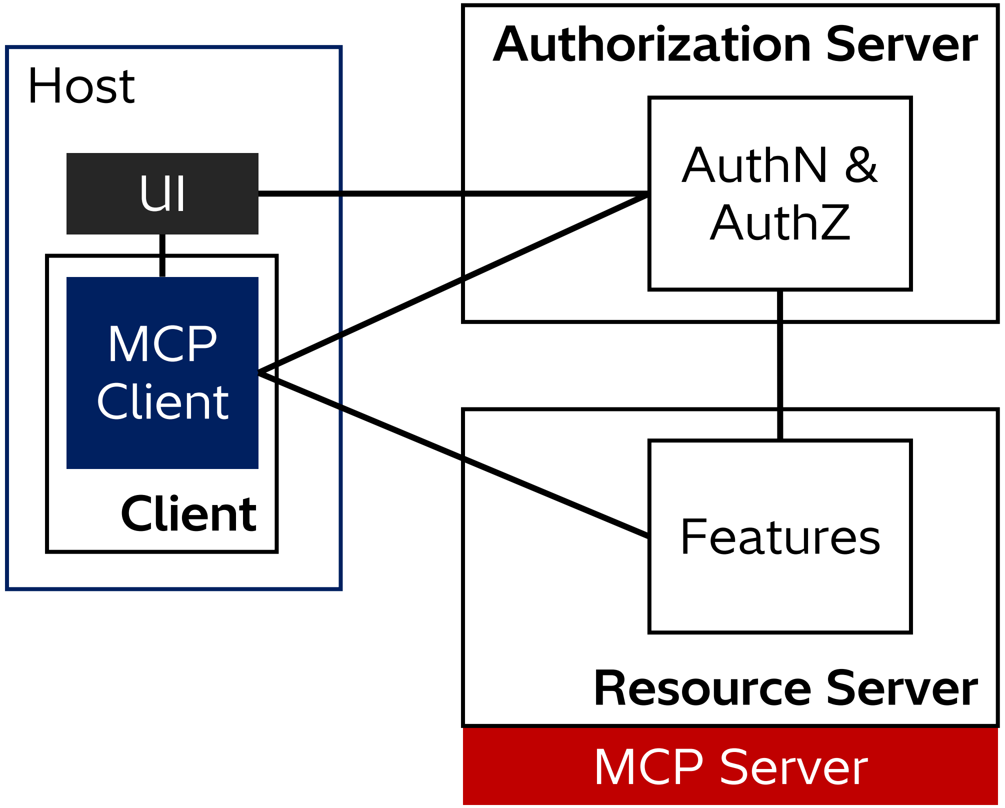
*3: **Open-Meteo** - https://github.com/isdaniel/mcp_weather_server

*4: **GitHub** - https://github.com/github/github-mcp-server

*5: **GitHub** - https://github.blog/changelog/2025-06-12-remote-github-mcp-server-is-now-available-in-public-preview

MCP in OAuth 2.0 Context

OAuth 2.0 Context:



Roles:

MCP	OAuth2
MCP Client	Client
MCP Server	Resource Server
Authorization Server	Authorization Server
User	Resource Owner

*: AuthN := Authentication, AuthZ := Authorization

Authentication and Authorization in MCP

Server Metadata Discovery:

- Advertising an Authorization Server’s metadata to a Client
- Advertising a Resource Server’s metadata to a Client

Dynamic Client Registration:

- Dynamically registering a Client to an Authorization Server

Authorization:

- Granting to a Client an access to a resource in a Resource Server
- Explicitly specifying a target resource in a Resource Server

Authentication and Authorization in MCP

Server Metadata Discovery:

- Advertising an Authorization Server’s metadata to a Client
- Advertising a Resource Server’s metadata to a Client

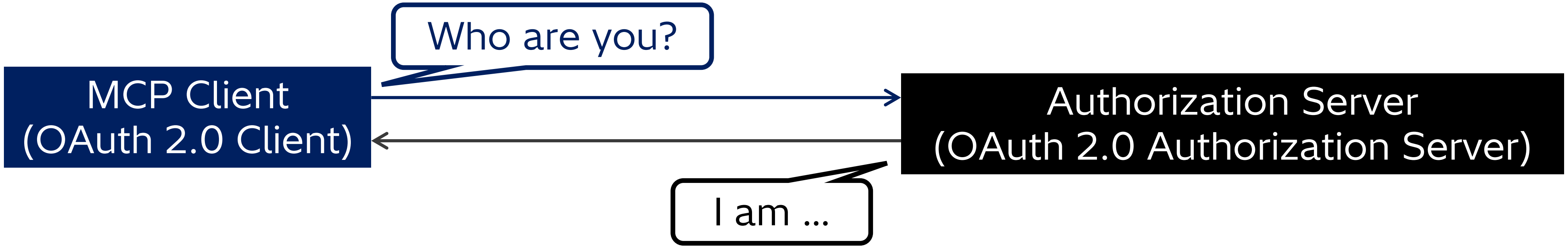
Dynamic Client Registration:

- Dynamically registering a Client to an Authorization Server

Authorization:

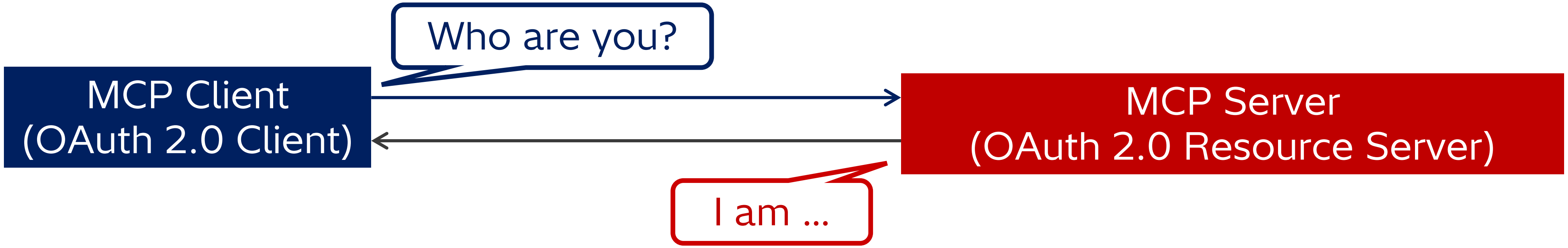
- Granting to a Client an access to a resource in a Resource Server
- Explicitly specifying a target resource in a Resource Server

Server Metadata Discovery – Authorization Server



- Advertising an **Authorization Server**'s metadata to a **Client** (OAuth2)
  → Advertising an **Authorization Server**'s metadata to an **MCP Client** (MCP)
- Specification: RFC 8414 OAuth 2.0 Authorization Server Metadata

Server Metadata Discovery – Resource Server



- Advertising a **Resource Server**’s metadata to a **Client** (OAuth2)
  → Advertising an **MCP Server**’s metadata to an **MCP Client** (MCP)
- Specification: RFC 9728 OAuth 2.0 Protected Resource Metadata

Authentication and Authorization in MCP

Server Metadata Discovery:

- Advertising an Authorization Server's metadata to a Client
- Advertising a Resource Server's metadata to a Client

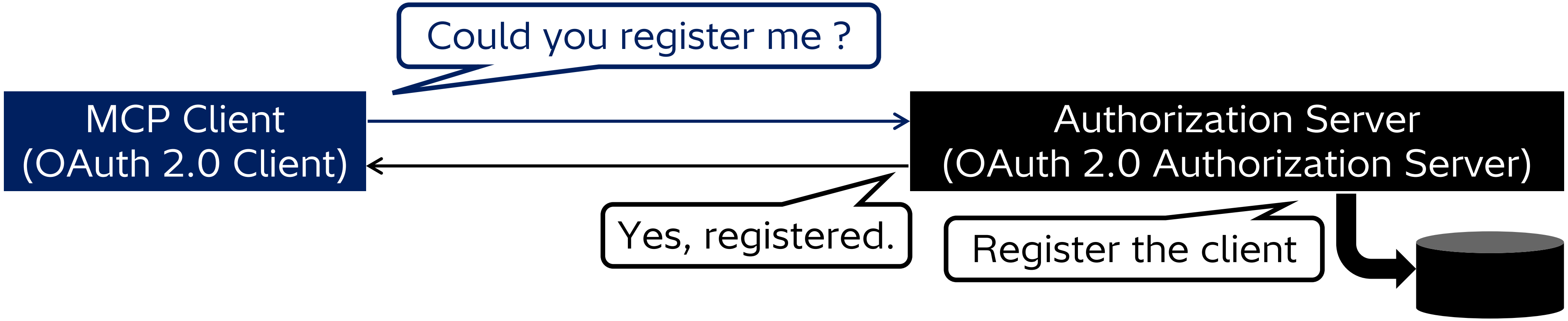
Dynamic Client Registration:

- Dynamically registering a Client to an Authorization Server

Authorization:

- Granting to a Client an access to a resource in a Resource Server
- Explicitly specifying a target resource in a Resource Server

Dynamic Client Registration



- Dynamically registering a **Client** to an **Authorization Server** (OAuth2)
  → Dynamically registering an **MCP Client** to an **Authorization Server** (MCP)
- Specification: RFC 7591 OAuth 2.0 Dynamic Client Registration Protocol

Authentication and Authorization in MCP

Server Metadata Discovery:

- Advertising an Authorization Server’s metadata to a Client
- Advertising a Resource Server’s metadata to a Client

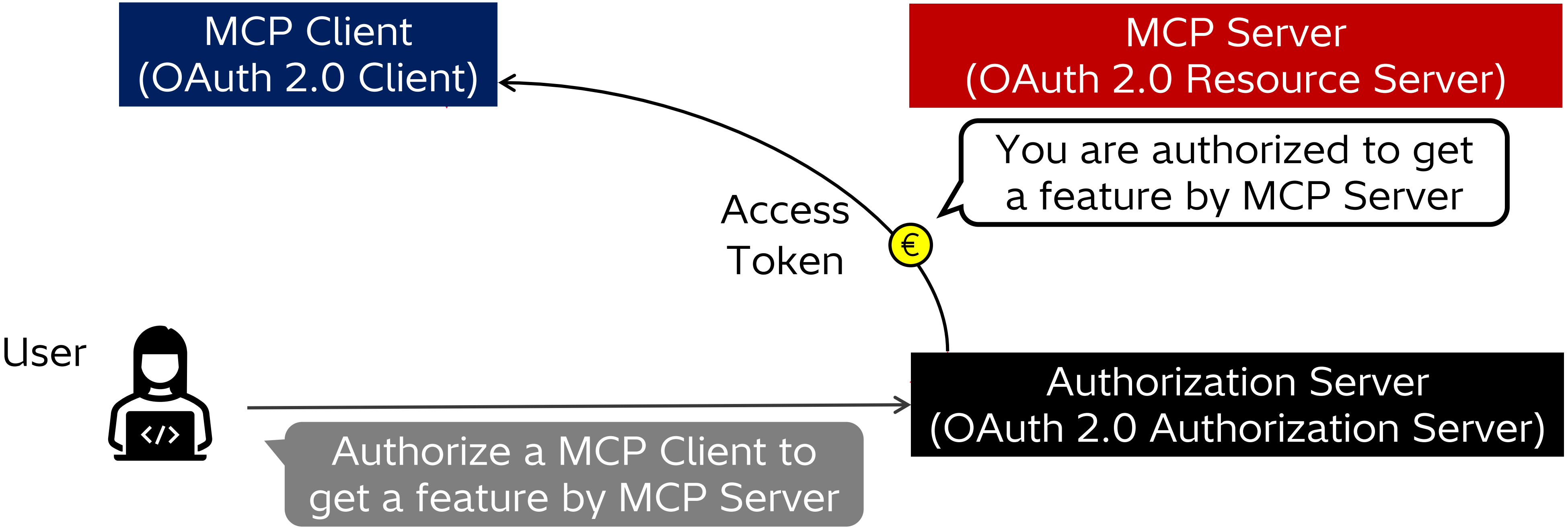
Dynamic Client Registration:

- Dynamically registering a Client to an Authorization Server

Authorization:

- Granting to a Client an access to a resource in a Resource Server
- Explicitly specifying a target resource in a Resource Server

Authorization


MCP Client
(OAuth 2.0 Client)

MCP Server
(OAuth 2.0 Resource Server)

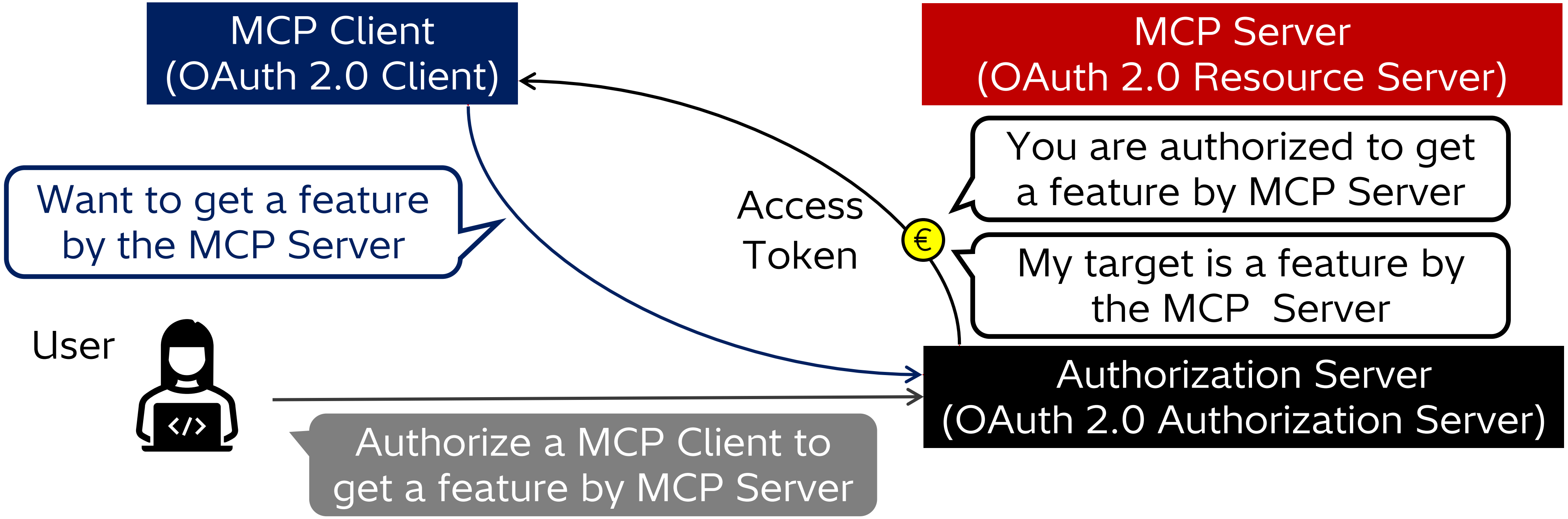
You are authorized to get a feature by MCP Server

Authorization Server
(OAuth 2.0 Authorization Server)

Authorize a MCP Client to get a feature by MCP Server

- Granting to a **Client** an access to a resource in a **Resource Server** (OAuth2)
  → Granting to an **MCP Client** get a feature by an **MCP Server** (MCP)
- Specification: Internet-Draft The OAuth 2.1 Authorization Framework
  - The next version of RFC 6749 OAuth 2.0
  - Hardening OAuth 2.0 e.g., enforcing RFC 7636 Proof Key for Code Exchange (PKCE)

Token Audience Binding


MCP Client
(OAuth 2.0 Client)
MCP Server
(OAuth 2.0 Resource Server)
You are authorized to get a feature by MCP Server
Want to get a feature by the MCP Server
Access Token
€
My target is a feature by the MCP Server
User
Authorization Server
(OAuth 2.0 Authorization Server)
Authorize a MCP Client to get a feature by MCP Server

- Explicitly specifying a target resource in a **Resource Server** (OAuth2)
  → Explicitly specifying a target resource in an **MCP Server** (MCP)
- Specification: RFC 8707 Resource Indicators for OAuth 2.0

Specification Conformance

Protocol Revision:

- **2025-06-18 (latest)**

	Authorization Server	**MCP Server**	**MCP Client**
Server Metadata Discovery (MUST)			
RFC 8414 OAuth 2.0 Authorization Server Metadata	X		X
RFC 9728 OAuth 2.0 Protected Resource Metadata		X	X
Dynamic Client Registration (SHOULD)			
RFC 7591 OAuth 2.0 Dynamic Client Registration Protocol	X		X
Authorization (MUST)			
The OAuth 2.1 Authorization Framework	X		X
RFC 8707 Resource Indicators for OAuth 2.0	❓		X

Specification Conformance

Protocol Revision:

- **2025-06-18 (latest)**

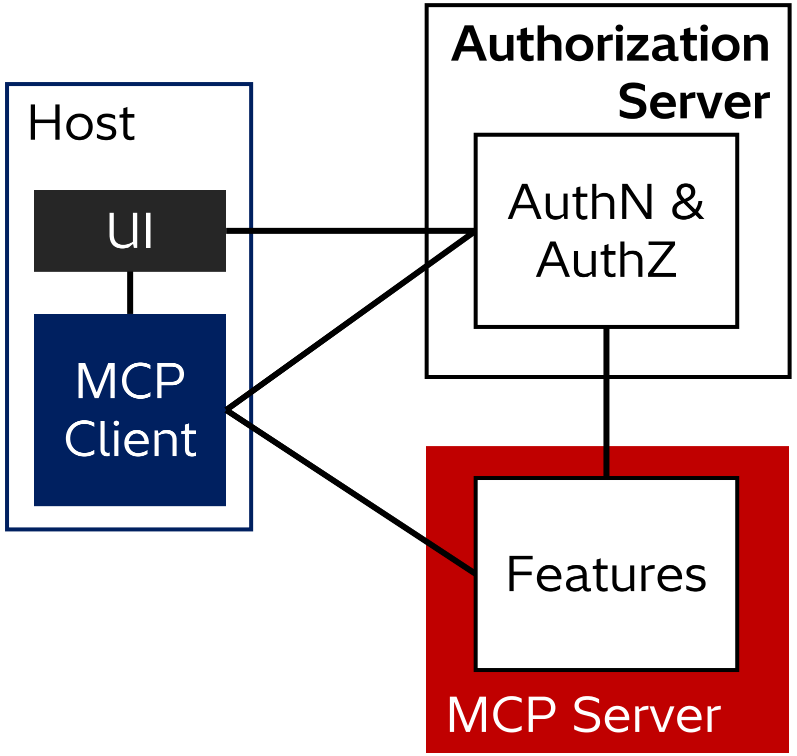
	Authorization Server	Keycloak 26.3
Server Metadata Discovery (MUST)		
RFC 8414 OAuth 2.0 Authorization Server Metadata	X	⚠
RFC 9728 OAuth 2.0 Protected Resource Metadata	-	-
Dynamic Client Registration (SHOULD)		
RFC 7591 OAuth 2.0 Dynamic Client Registration Protocol	X	✅
Authorization (MUST)		
The OAuth 2.1 Authorization Framework	X	✅
RFC 8707 Resource Indicators for OAuth 2.0	❓	❌

HITACHI

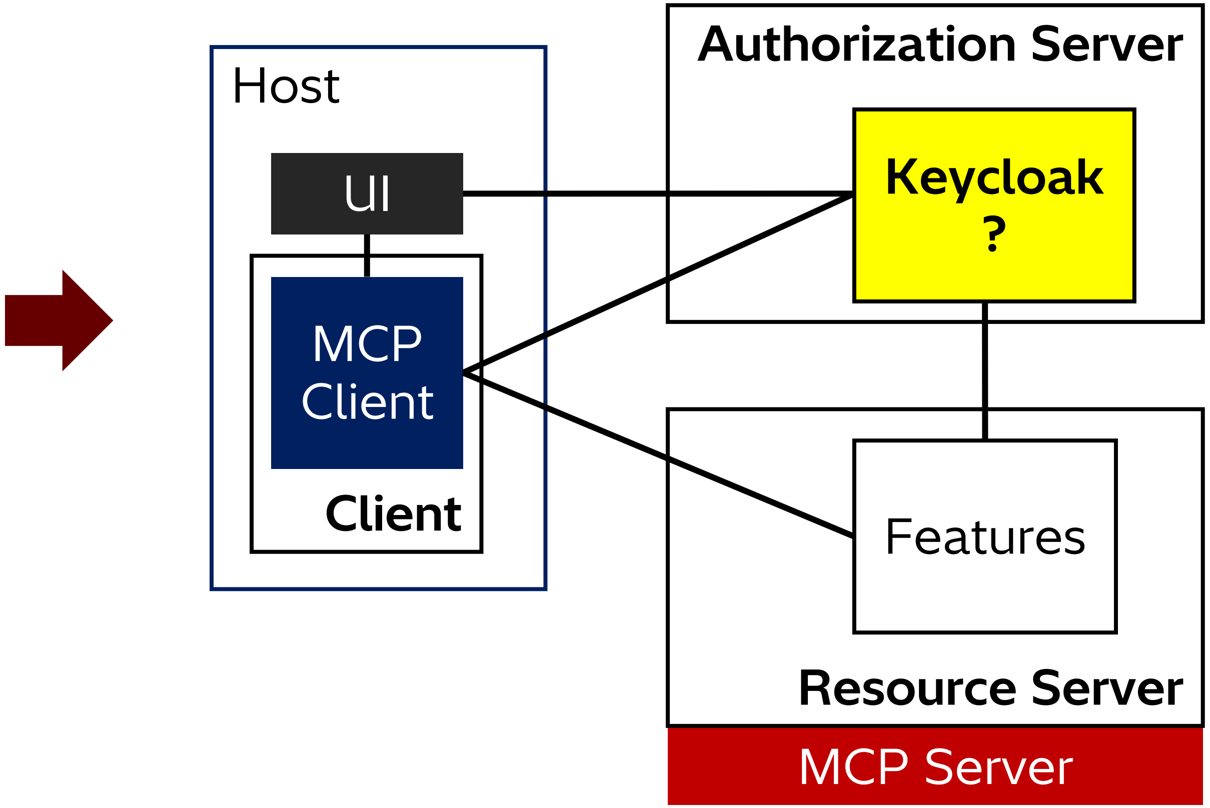
Contents

1. How can Keycloak be used in the context of AI?
2. Model Context Protocol (MCP) Overview
3. Authentication and Authorization in MCP
4. Keycloak in MCP

MCP in OAuth 2.0 Context



OAuth 2.0 Context:



*: AuthN := Authentication, AuthZ := Authorization

Two pain points Keycloak faces to comply with MCP

1. Server Metadata Discovery – Authorization Server
   RFC 8414 OAuth 2.0 Authorization Server Metadata

2. Token Audience Binding
   RFC 8707 Resource Indicators for OAuth 2.0

Issue 1: Server Metadata Discovery – Authorization Server

Keycloak’s issuer URL (issuer metadata):
https://keycloak.example.com**/realms/mcp-tools**

well-known URI crafted the URL by following RFC 8414:
https://keycloak.example.com**/.well-known/oauth-authorization-server/realms/mcp-tools**

Actual Keycloak’s well-known URI:
https://keycloak.example.com**/realms/mcp-tools/.well-known/oauth-authorization-server**

→ Keycloak needs to follow RFC 8414 for well-known URI

But, according to the merged PR (#677) , which allows to use OIDC Discovery specified server metadata, Keycloak might not need to follow RFC 8414 defining well-known endpoint URI for MCP in the next version of MCP.

Issue 1: Server Metadata Discovery – Authorization Server

Wait for the PR being merged:

Discussion: Complying with RFC 8414 about Well-Known URI for Authorization Server Metadata
https://github.com/keycloak/keycloak/discussions/40809

Issue: https://github.com/keycloak/keycloak/issues/40923

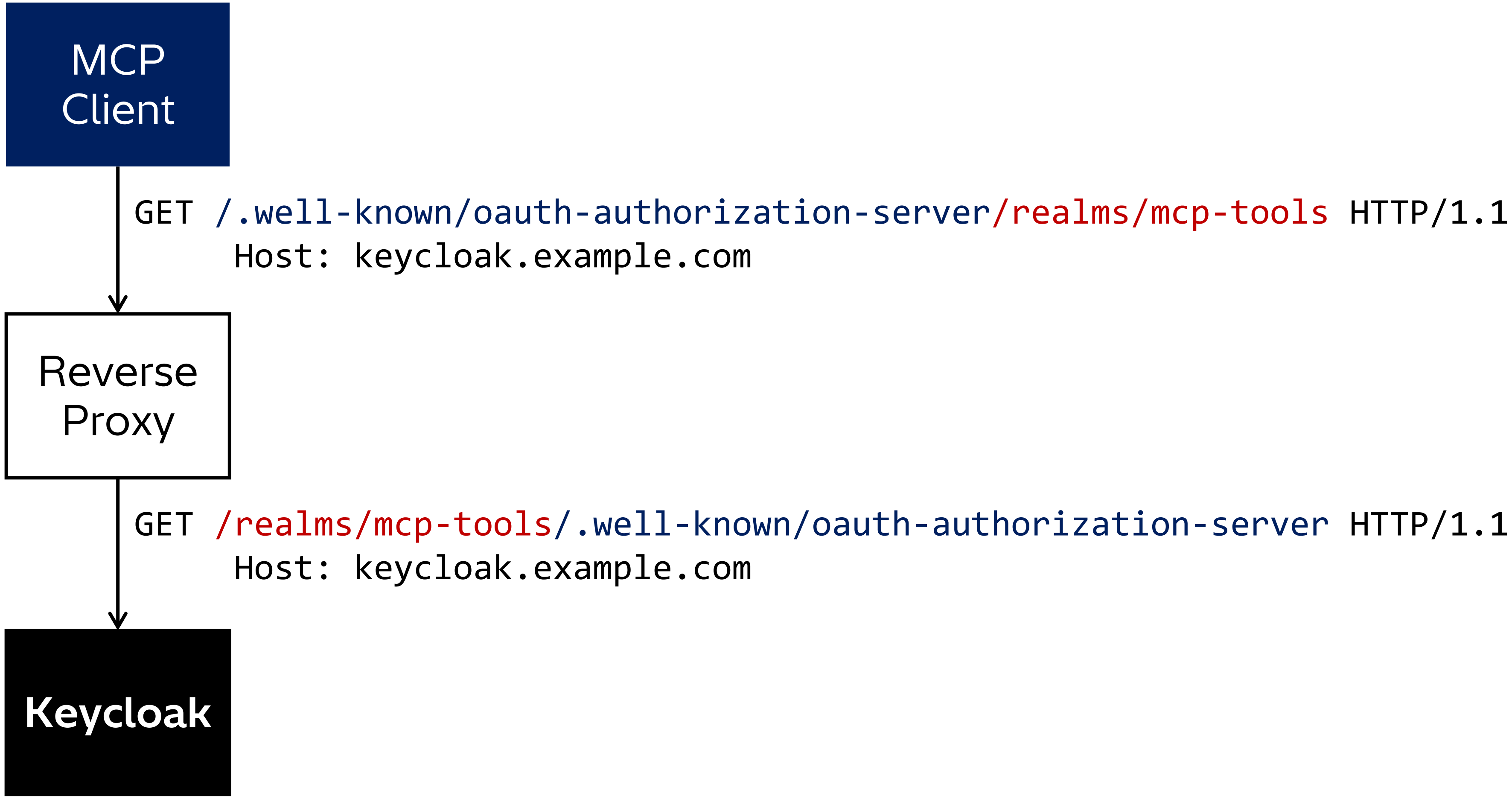
PR: https://github.com/keycloak/keycloak/pull/41440

Workaround:

Use Reverse Proxy in front of Keycloak

Solution 1: Server Metadata Discovery – Authorization Server

Use Reverse Proxy in front of Keycloak



MCP Client

```
GET /.well-known/oauth-authorization-server/realms/mcp-tools HTTP/1.1
     Host: keycloak.example.com
```

Reverse Proxy

```
GET /realms/mcp-tools/.well-known/oauth-authorization-server HTTP/1.1
     Host: keycloak.example.com
```

Keycloak

Issue 2: Token Audience Binding

P1: binding an access token with its audience represented in *resource* parameter

MCP spec says "MCP servers MUST validate that access tokens were issued specifically for them as the intended audience, according to RFC 8707 Section 2."

→ Keycloak needs to create an access token that can prove its intended audience, which is specified by the *resource* parameter sent from an MCP client.

Issue 2: Token Audience Binding

P2: consistency check of *resource* parameters

MCP spec says “MCP clients MUST include the *resource* parameter in both authorization requests and token requests.”

→ Keycloak needs to check whether the *resource* parameters of an authorization request and token request are consistent.

NOTE: treatment of *resource* parameter in RFC 8707, MCP and Keycloak

RFC 8707 allows *resource* parameter in a token refresh request.

→ MCP (2025-06-18) does not mention a *resource* parameter in token refresh requests.

RFC 8707 allows multiple *resource* parameters in an authorization, token and token refresh request.

→ MCP (2025-06-18) does not mention multiple *resource* parameter in an authorization and token request.

→ Keycloak does not allow any of duplicated parameter in an authorization request.

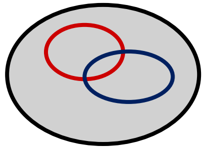
RFC 8707 requires values of *resource* parameter(s) in a token and token refresh request should be a subset of values of *resource* parameter(s) in an authorization request.

RFC 8707 allows different values of *resource* parameter(s) in a token and token refresh request.

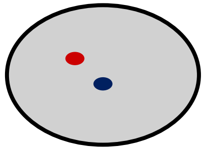
RFC 8707 security consideration recommends single *resource* parameter in a token and token refresh request.

<valid *resource* parameter values>

RFC 8707



RFC 8707
Recommend



- : authorization request
- : token request
- : token refresh request

Solution 2: Token Audience Binding

Wait for the PR being merged:

- Discussion: Support for RFC 8707 OAuth2 Resource Indicators
  - https://github.com/keycloak/keycloak/discussions/35743
- Issue: Support RFC 8707 Resource Indicators for OAuth 2.0
  - https://github.com/keycloak/keycloak/issues/14355
- PR: Add support for RFC 8707 OAuth2 Resource Indicators
  - https://github.com/keycloak/keycloak/pull/35711

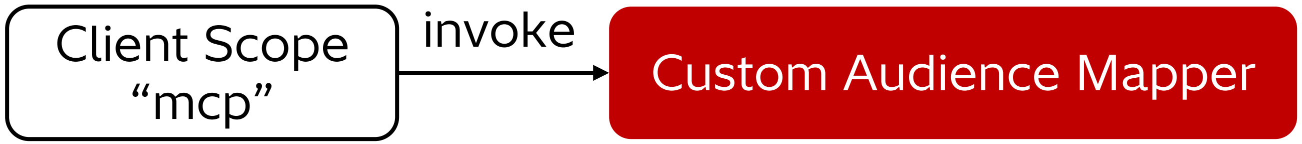
Workaround:

- P1: Develop and use custom Audience Mapper
- P2: Develop and use custom executor of client policies

Solution 2: Token Audience Binding

Workaround:
P1: Develop and use custom Audience Mapper

Token Request…

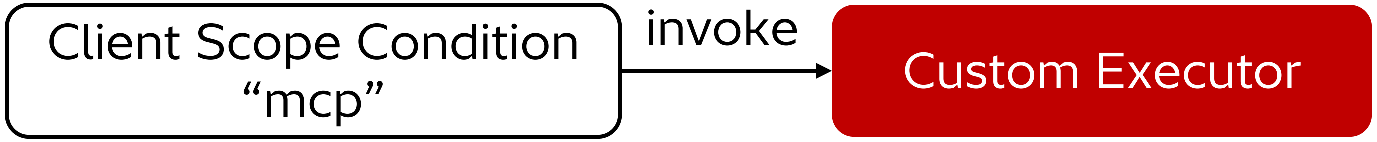


- Retrieve a *resource* parameter’s value
- Add the value to the “aud” claim of an access token

Solution 2: Token Audience Binding

Workaround:
P2: Develop and use custom executor of client policies

Authorization Request…

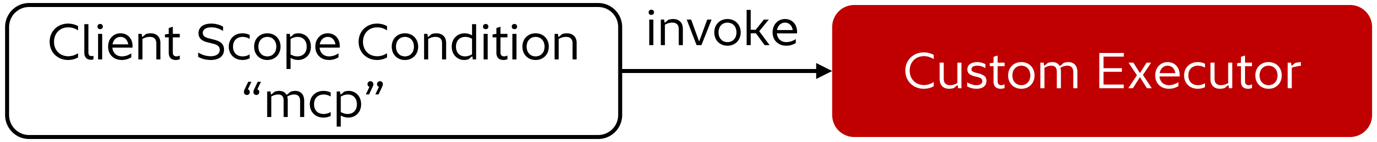

Client Scope Condition "mcp"

invoke

Custom Executor

- Retrieve a *resource* parameter's value
- Store the value temporarily

Token Request…

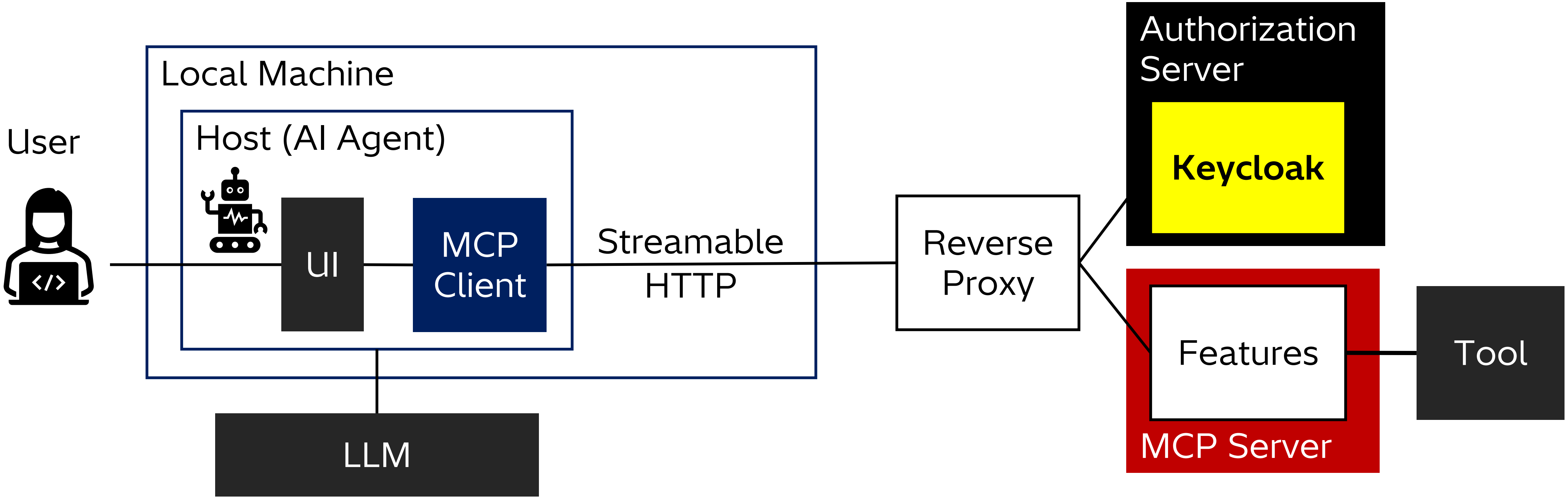

Client Scope Condition "mcp"

invoke

Custom Executor

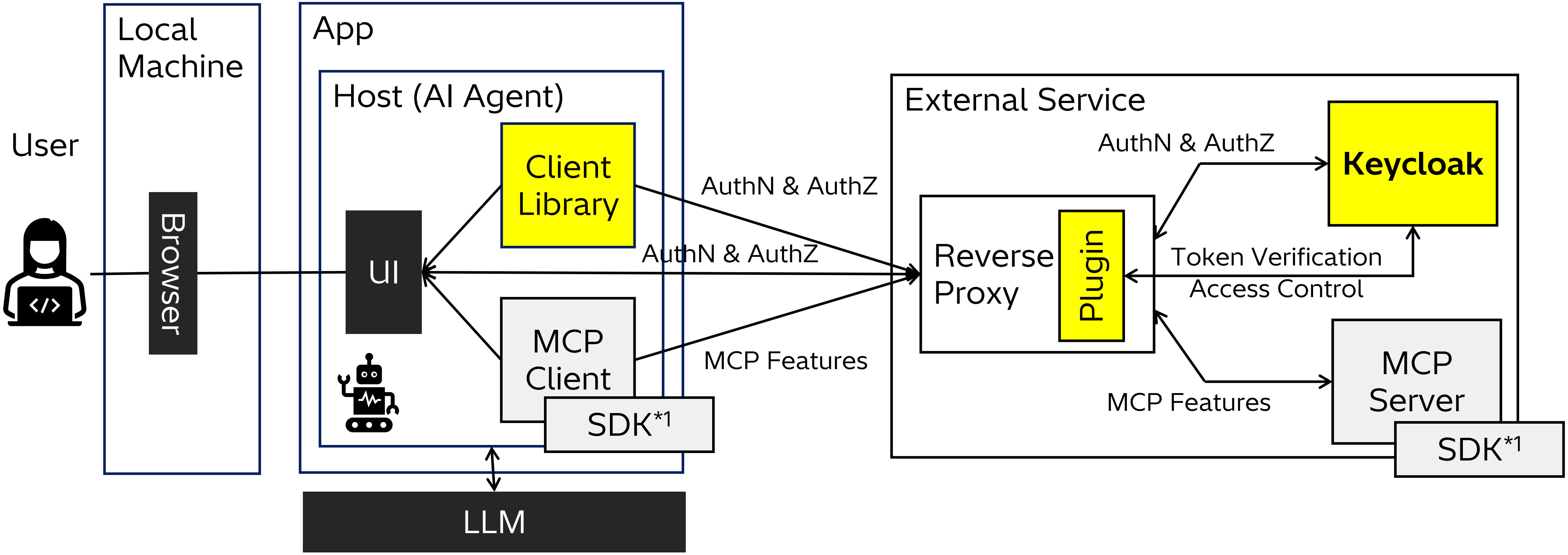
- Retrieve a *resource* parameter's value
- Exactly match the value with the stored resource parameter's value

Keycloak with MCP Server



*: AuthN := Authentication, AuthZ := Authorization

POC of Keycloak with MCP Server



🟨 AuthN & AuthZ ⬜ MCP Features

*: AuthN := Authentication, AuthZ := Authorization
*1: https://github.com/modelcontextprotocol

Why Keycloak? – its advantages

Specification Conformance:

- Keycloak supports OAuth 2.1 for both public and confidential clients[*1].
  ➔ MCP requires OAuth 2.1. Keycloak meets the requirement.

[*1]: Keycloak supports built-in security profiles for OAuth 2.1:
https://www.keycloak.org/docs/26.2.1/server_admin/index.html#_client_policies

Why Keycloak? – its advantages

Safety:

- Keycloak got certified[*1,2] as the server complying with OpenID Connect (OIDC) and Financial-grade API (FAPI) by OpenID Foundation (OIDF).
  ➔ It ensures Keycloak complies with several security standards.
- Keycloak community[*3] confirms that every version of Keycloak complies with such the security standards by running regression tests.
  ➔ It ensures every version of Keycloak complies with the security standards.

*1: The following slides shows which security standards Keycloak complies with:
https://www.keycloak-day.dev/assets/files/Norimatsu_KeycloakDevDay2025_Darmstadt.pdf

*2: The following OIDF sites shows which specifications Keycloak got certified to comply with:
https://openid.net/certification/

*3: The following Keycloak community (OAuth SIG) sites shows which specification Keycloak complies with:
https://github.com/keycloak/keycloak-oauth-sig?tab=readme-ov-file#passed-conformance-tests-per-keycloak-version

Why Keycloak? – its advantages

Safety:

- Keycloak supports a lot of policies[*1] for dynamic client registration.
  ➔ It can prevent a malicious client from registering.
- Keycloak supports passwordless user authentication like passkey[*2].
  ➔ It can prevent a malicious user from impersonating.

*1: https://www.keycloak.org/securing-apps/client-registration#_client_registration_policies
*2: https://www.keycloak.org/docs/26.2.1/server_admin/index.html#passkeys_server_administration_guide

Why Keycloak? – its advantages

Coverage:

- Keycloak can manage users and their sessions.
  ➔ No need to implement these management by yourself.
- Keycloak supports various customization.
  ➔ Easy to implement your own access control method.

Why Keycloak? – its advantages

Soundness:

- Keycloak is Open-Source Software (OSS).
  ➔ No vendor lock-in.
- Keycloak can run everywhere.
  ➔ You can store your user information on a data center in your country.
- Keycloak's community is very active
  (>29k GitHub stars, >1300 contributors).
  ➔ A lot of users and contributors, who make Keycloak more attractive.
- Keycloak is Cloud Native Computing Foundation (CNCF) incubating project.
  ➔ Easy to run and manage Keycloak on every cloud-native environment.
  Vendor-neutral governance, highly sustainable.

Key Takeaways

- MCP is a promising protocol in the field of AI agent.
- Authentication and authorization by following OAuth 2.1 is sometimes required.
- Keycloak can be used for authentication & authorization.
- Keycloak has several advantages for using that purpose in the field of AI agent.

Trademarks

- OpenID is a trademark or registered trademark of OpenID Foundation in the United States and other countries.
- Facebook is a trademark or registered trademark of Facebook, Inc. in the United States and other countries.
- CNCF is a trademark or registered trademark of The Linux Foundation in the United States and other countries.
- Other brand names and product names used in this material are trademarks, registered trademarks, or trade names of their respective holders.

HITACHI

Thank you

KeyConf 25
@Van der Valk Hotel Amsterdam Zuidas - RAI, Amsterdam, Netherlands

Keycloak meets AI: the possibility of integrating Keycloak with AI

Takashi Norimatsu	Date
Open Source Program Office (OSPO)	August 28, 2025
Hitachi, Ltd.	

HITACHI