



2025.5.26

# MCPにおける認証・認可



@tubone24

tubone-project24.xyz

mugimugi.cutedog

ソフトウェアエンジニア

# 大坪 悠 Yu Otsubo



## 職歴：

新潟県出身。大学卒業後、2016年にKDDIに新卒入社。

通信設備の運用、社内データ基盤構築のPM、Webチャットボット開発などを経験する。

スタートアップへの転職を経て2024年にKAG入社。

KAGでは生成AIを活用したアプリケーション開発に従事。

## 著書：

「K.A.G Tech Book」(KDDIアジャイル開発センター)

「やさしいMCP入門」(秀和システム)

## 趣味：

愛犬と遊ぶこと

個人開発



## モットー：

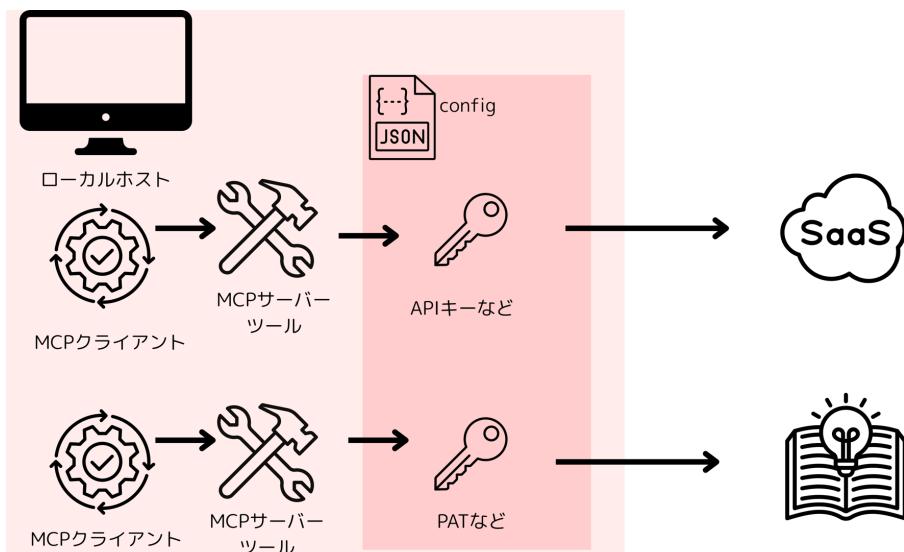
早く小さく失敗する



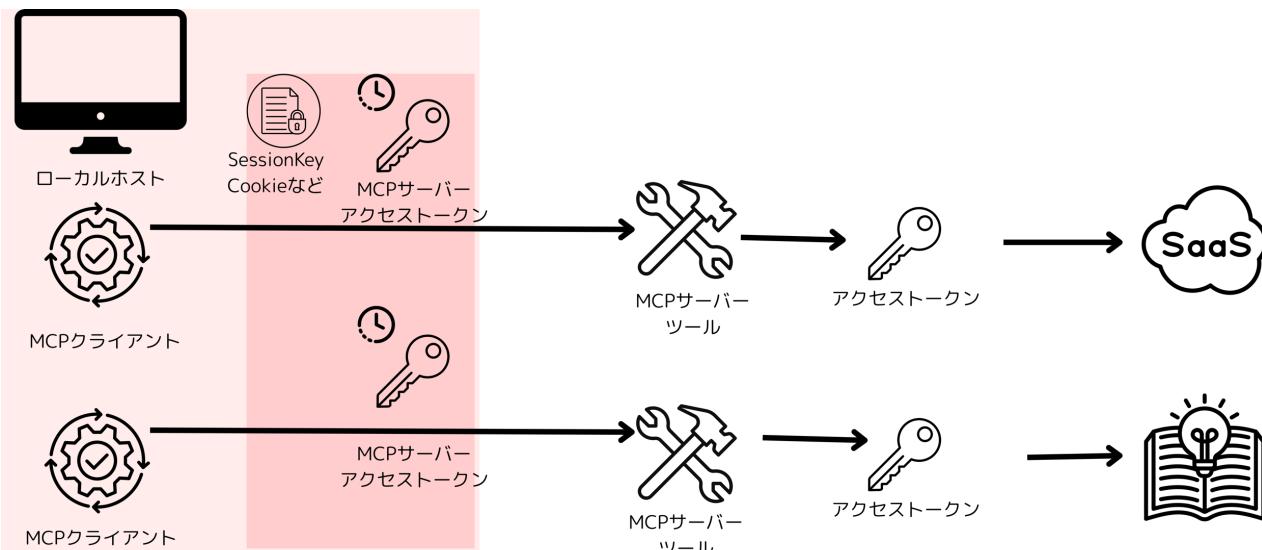
# MCPにおける認証・認可の重要性

## リモートMCPサーバーの普及や機微情報の扱いにおいて認証認可は重要

従来



これから



### 【💡ポイント💡】

ローカルMCPサーバーは接続コンフィグ(claude\_desktop\_config.jsonなど)に  
強力なアクセストークンがそのまま記載されてしまう  
一方でリモートMCPサーバーはMCPサーバー専用のアクセストークンを払い出す形

## リモートMCPサーバーに強力なAPIキーを渡すことはできない!

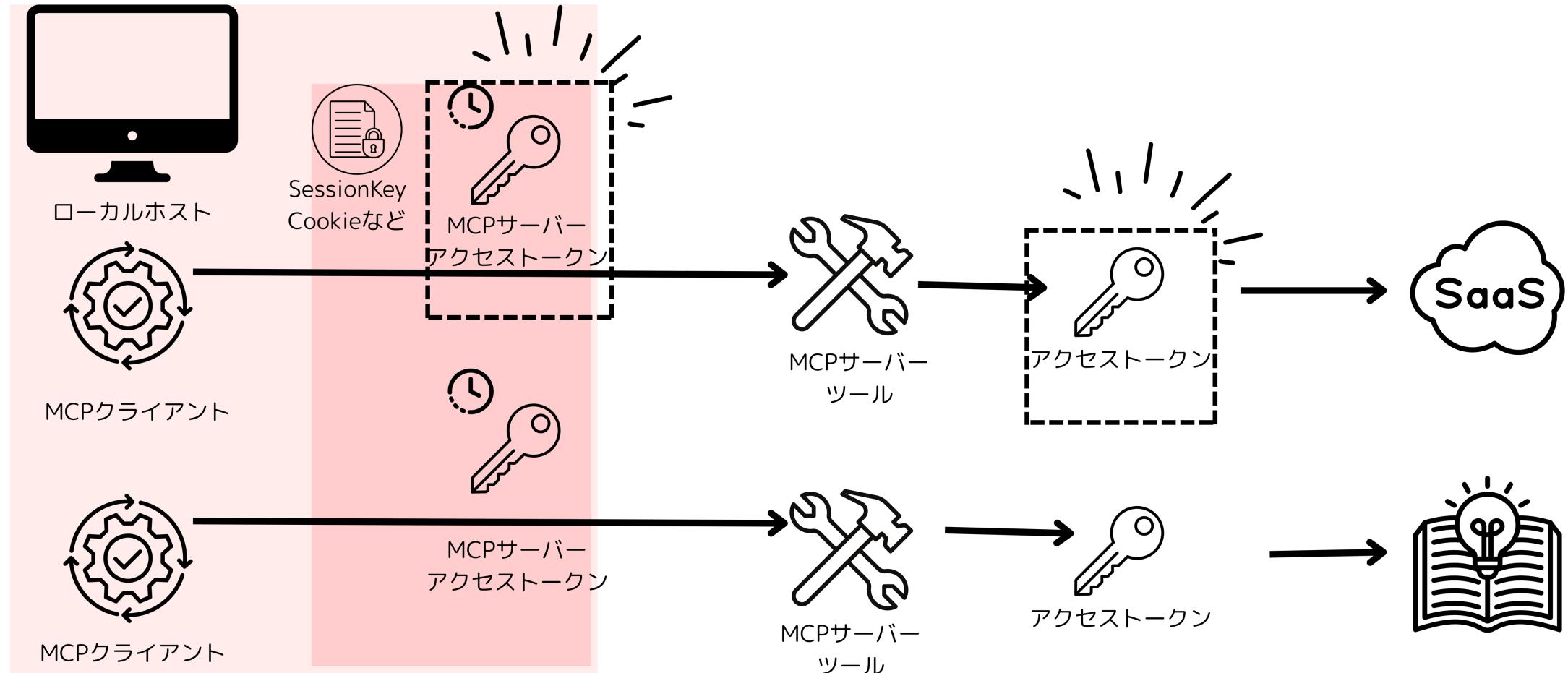
APIは静的トークン = 漏洩時の影響が大きすぎる

- 定期的なキーローテをユーザーに求める必要がある
  - 誰もやらない
- (外部の)MCPサーバー内にキーを抱え込みにくい
  - DBのID/PWとかとてもじゃないけど持てない
- SaaS権限とは別にリソースサーバーとしてのMCPサーバーの権限制御ができない
  - SharePointを読み込むけど、既存の記事は編集できない、など。

<https://github.com/modelcontextprotocol/modelcontextprotocol/discussions/234>

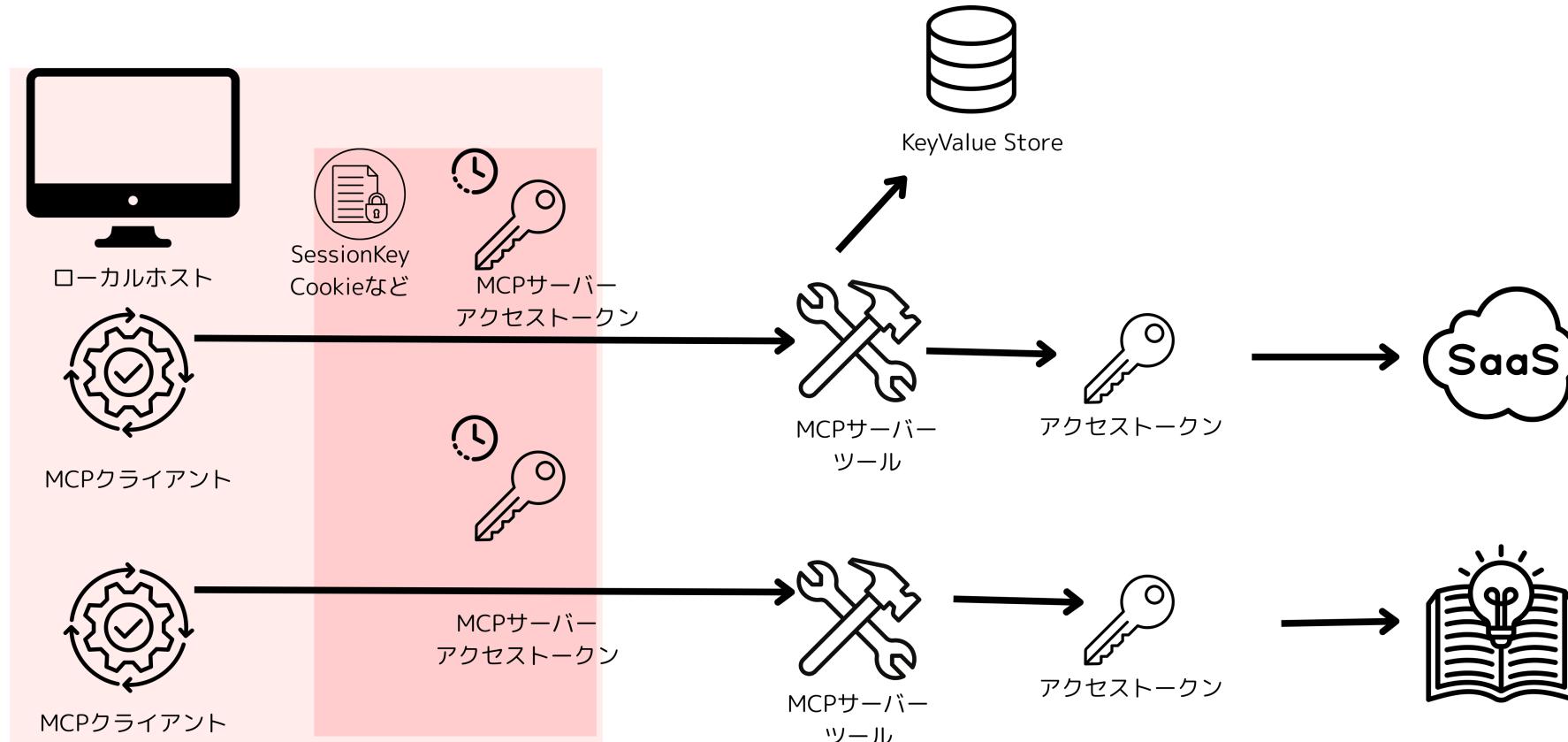
このあたりで議論されていました。

## 大きな特徴は、MCPクライアント vs サーバー間の認可と MCPサーバー vs 外部プロダクト間の認可の二つがあること



## MCPにおける認証・認可の重要性

なので、認証認可が完了すると、二つのアクセストークンが出来上がる状態。  
加えて言うと、二つのアクセストークンを紐づける仕組みが必要になる。



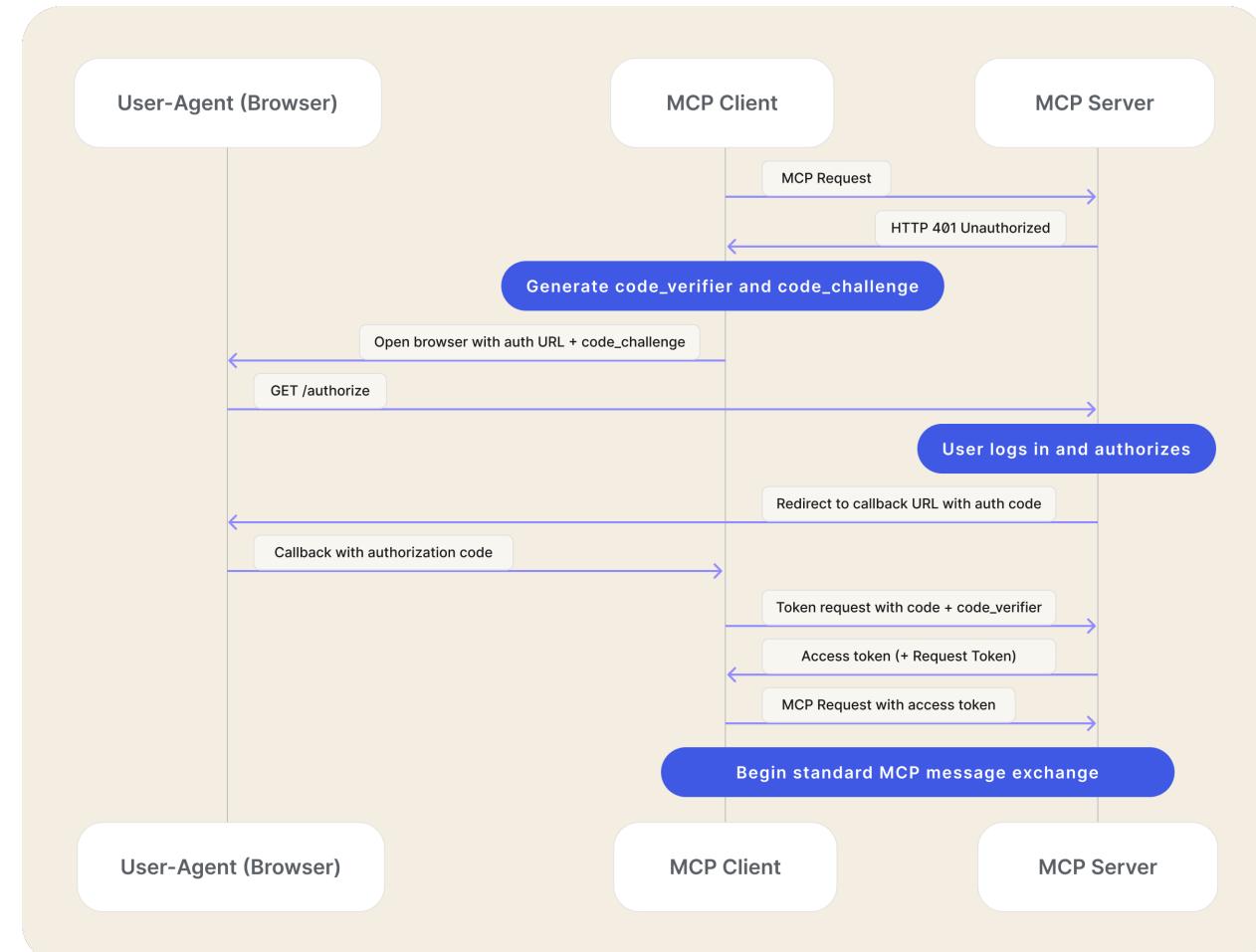
よくある手法としてKeyValueでそれぞれの認証・認可のフローで必要な情報を保存しておく。  
DBとしてTTLが設定できるとなおセッション感が出てよい。

# MCP(OAuth2.1)を支える技術

# PKCE(ピクシーと読む)を用いた認可フロー(MUST)

## Proof Key for Code Exchange by OAuth Public Clients

- 認可コードを横取りされない仕組み
  - 認可リクエスト者とトークンリクエスト者が同じである証明
    - 乱数とそのハッシュをそれぞれで突合させる
- パブリックなクライアントでも使える
  - クライアントを証明する方法=>クライアントクレデンシャル
  - Webページやスマホには埋め込めない欠点
  - クライアントを証明する代わりの方法



## メタデータディスカバリー

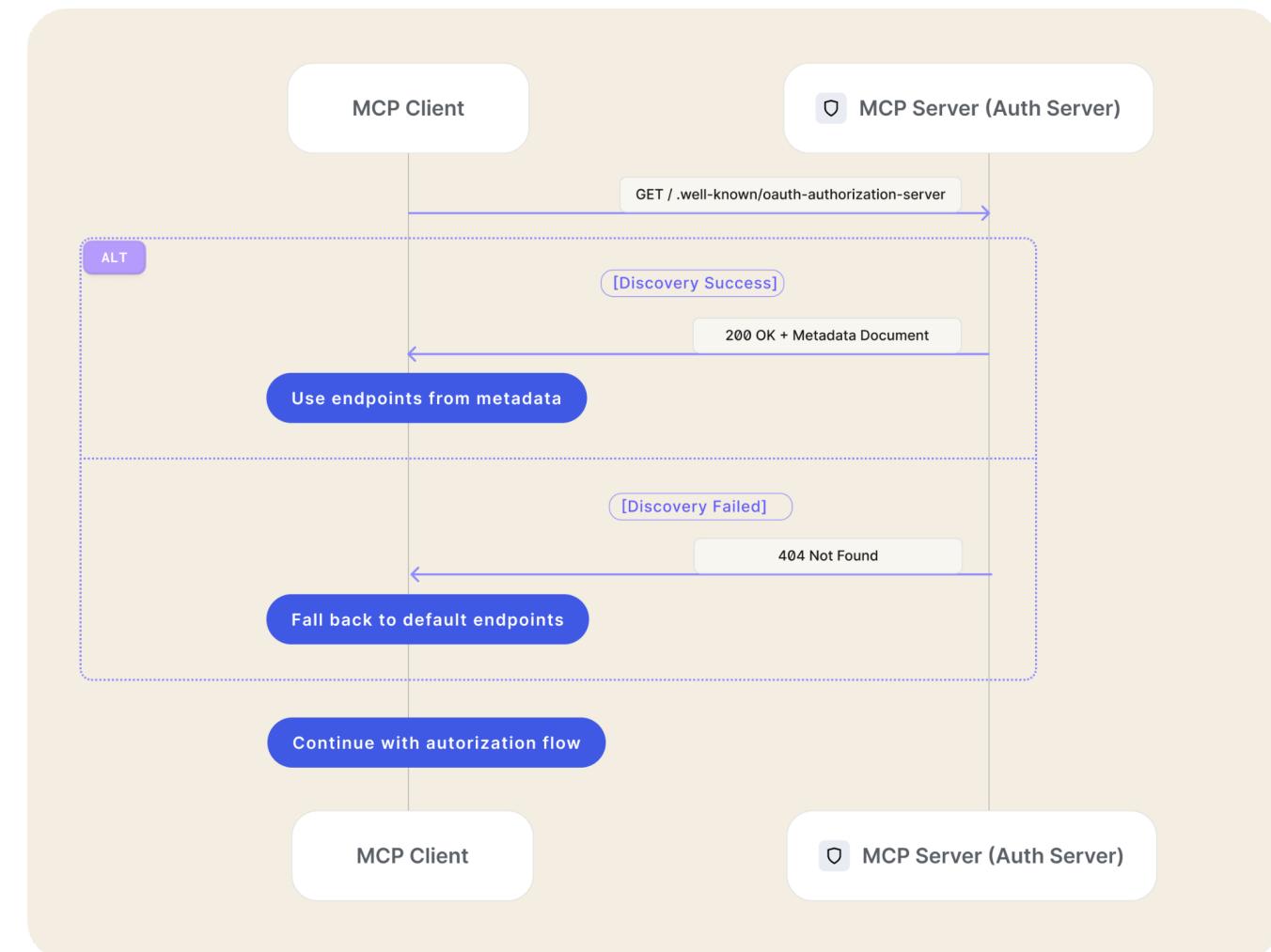
MCPサーバーにいざアクセスしても、どこを叩いて認証・トークン取得すればいいかわからない。

なので、well-knownディレクトリとして、認証に関する情報をまとめておくエンドポイントを用意することが推奨される

対応してない場合は、下記のデフォルトエンドポイントを暗黙的に利用する

### ■ デフォルトエンドポイント

- /authorize 認証エンドポイント
- /token トークンエンドポイント
- /register 動的クライアント登録エンドポイント



# 動的クライアント登録(DCR)

OAuthのクライアントを設定したことがある人はわかるかも

- 事前にクライアントIDとシークレットを登録する必要あり
- MCPクライアントに毎度設定するのは大変
- リクエストに応じてクライアントIDを発行する仕組み

Settings / Developer settings / MCP Auth Demo

**General**

MCP Auth Demo

tubone24 owns this application.

Transfer ownership

You can list your application in the [GitHub Marketplace](#) so that other users can discover it.

List this application in the Marketplace

0 users

Revoke all user tokens

**Client ID**

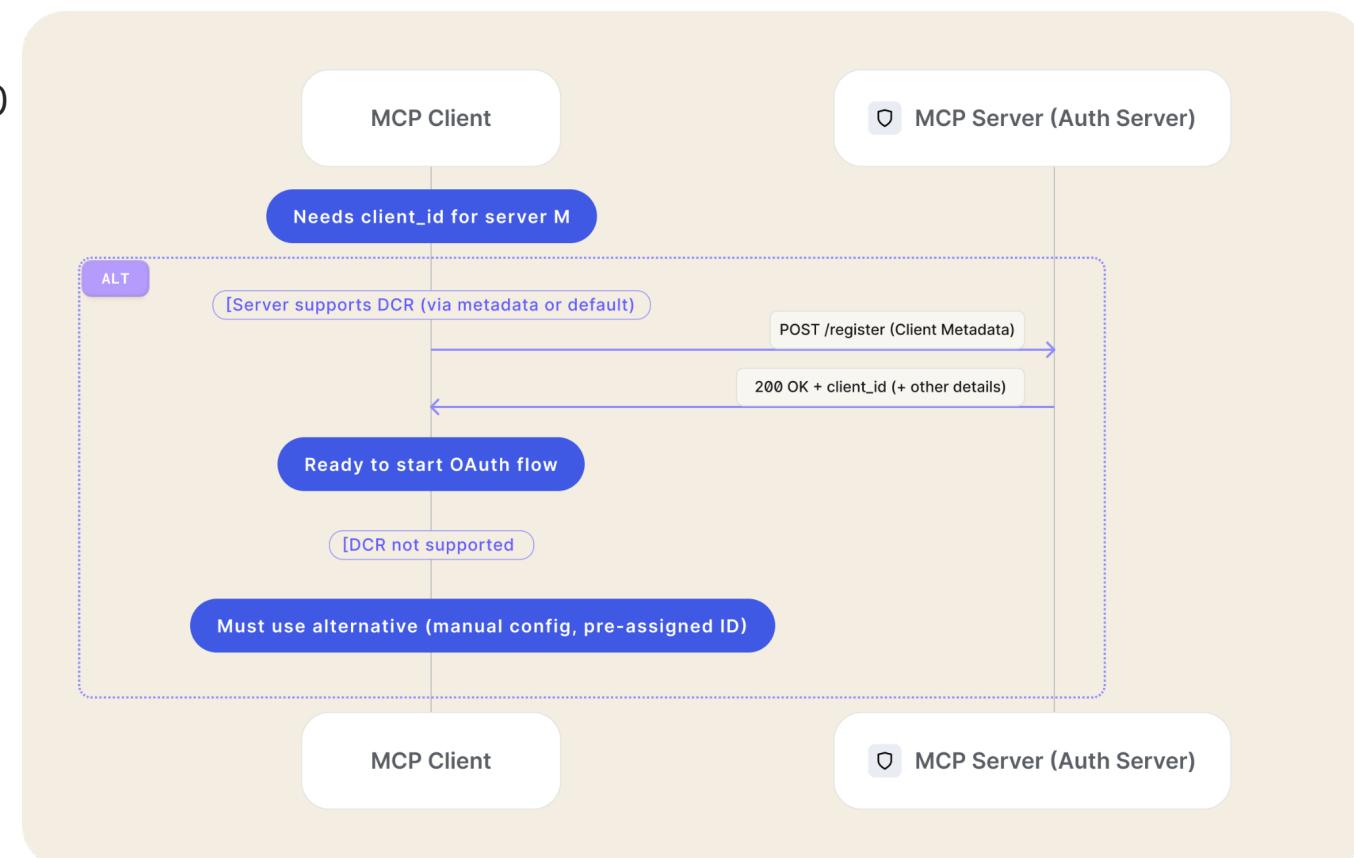
0: [REDACTED]

**Client secrets**

Generate a new client secret

Client secret: \*\*\*\*5ea2a441  
Added 3 hours ago by tubone24  
Never used  
You cannot delete the only client secret. Generate a new client secret first.

GitHubの例



デモ

(自作のリモートMCPサーバー+Inspector)

今回のコードは次のレポジトリに保存してます  
また、仕組みを作るのにCloudflare workersとKVを使いました

## tubone24/remote-mcp-oauth-github

A Model Context Protocol (MCP) server for Claude.ai custom integrations, running on Cloudflare Workers with GitHub OAuth authentication.

1

Contributor

0

Issues

0

Stars

0

Forks



<https://github.com/tubone24/remote-mcp-oauth-github>

## Streamable HTTPのエンドポイント(/mcp)にアクセスすると 認証を求められる

Streamable HTTPでアクセスします～  
<https://mcp-github-oauth-demo.tubone24.workers.dev/mcp>



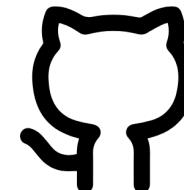
MCPホスト(クライアント)



リモートMCPサーバー  
(Cloudflare Workers)



Auth store  
(Cloudflare KV)



GitHub OAuth Server

ちゃんと認証してからにしてください～(401 Unauthorized)

```
{"jsonrpc":"2.0","error":{"code":-32000,"message":"Authentication required"}}
```



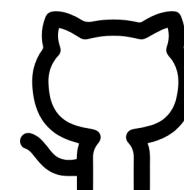
MCPホスト(クライアント)



リモートMCPサーバー  
(Cloudflare Workers)



Auth store  
(Cloudflare KV)



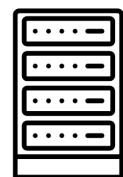
GitHub OAuth Server

## 認証に関するエンドポイントを一覧もらう これがメタデータディスカバリー

Wellknownにアクセスして認証のエンドポイント一覧もらいます～  
./well-known/oauth-authorization-server



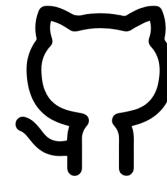
MCPホスト(クライアント)



リモートMCPサーバー  
(Cloudflare Workers)



Auth store  
(Cloudflare KV)



GitHub OAuth Server



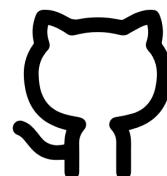
MCPホスト(クライアント)



リモートMCPサーバー  
(Cloudflare Workers)



Auth store  
(Cloudflare KV)



GitHub OAuth Server

## クライアントを自動で登録して以降のOAuthクライアントとして利用 これがDCR

今後OAuthで認証するためにもクライアントとして登録したいです！  
[/register](#)



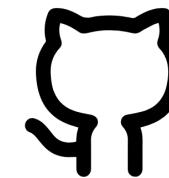
MCPホスト(クライアント)



リモートMCPサーバー  
(Cloudflare Workers)



Auth store  
(Cloudflare KV)



GitHub OAuth Server

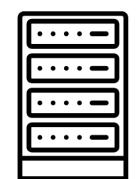
わかりました！あなたをクライアントxxxxとして登録しておきます。  
スコープなども登録しておきました。よろしく！

保存

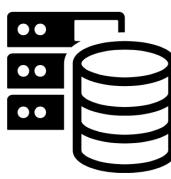
```
{  
  "client_id": "xxxx",  
  "client_secret": "yyyy"  
}
```



MCPホスト(クライアント)



リモートMCPサーバー  
(Cloudflare Workers)



Auth store  
(Cloudflare KV)



GitHub OAuth Server

## Authorization URLを発行

DCRで作成されたclient\_idに加え、code\_challengeが設定されている  
これがPKCE

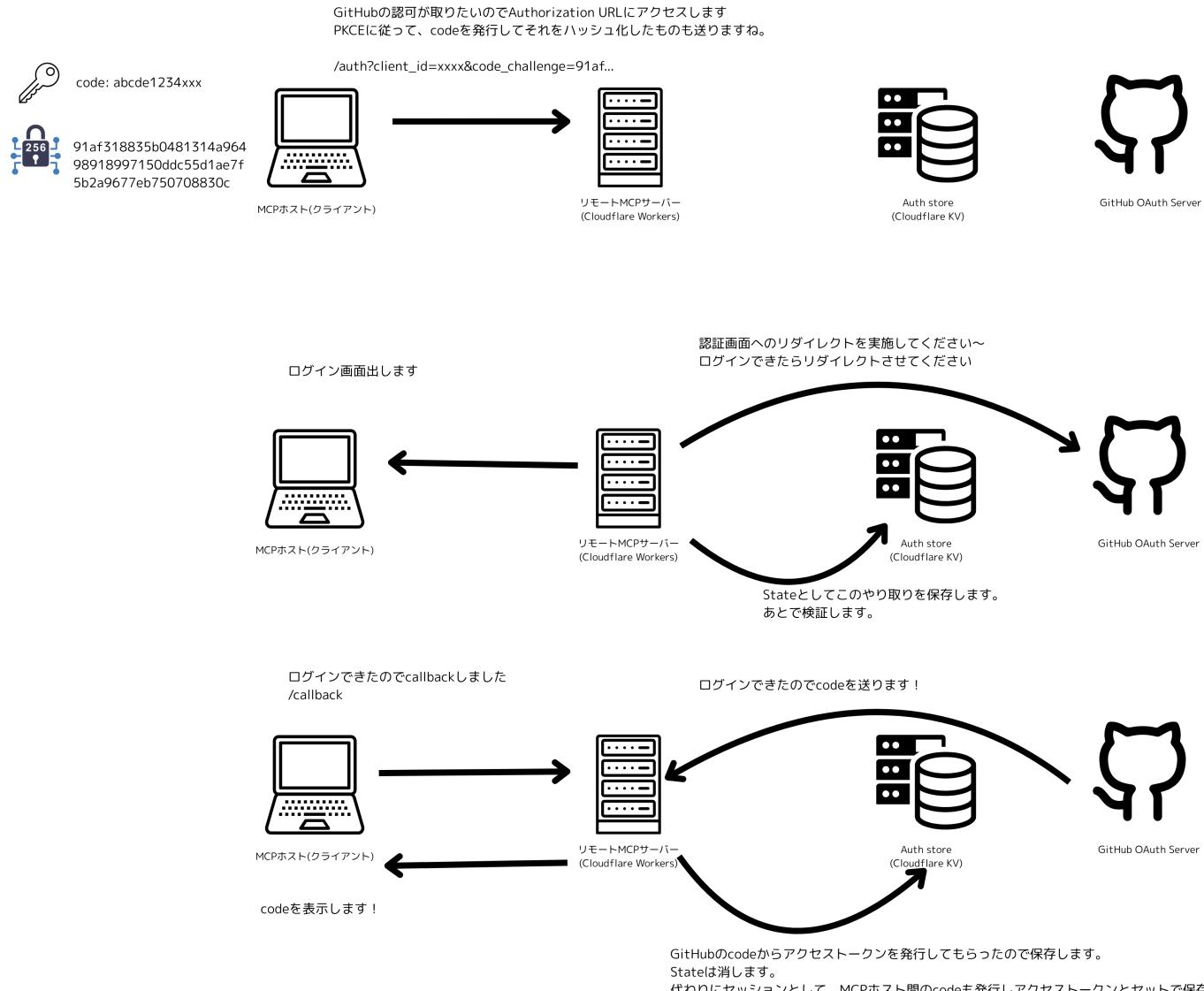
### ④ Preparing Authorization

#### Authorization URL:

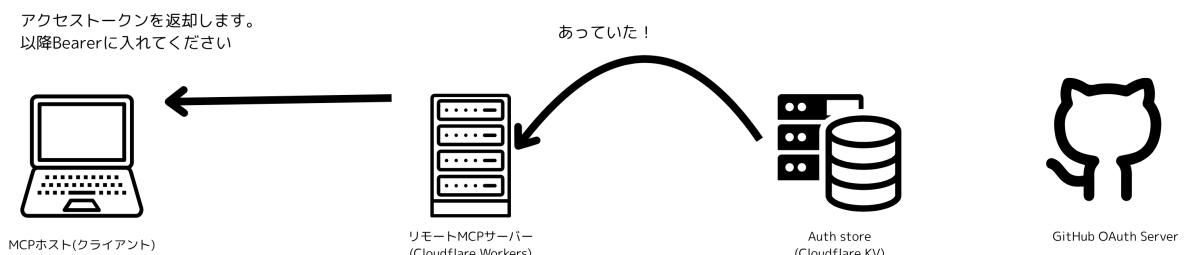
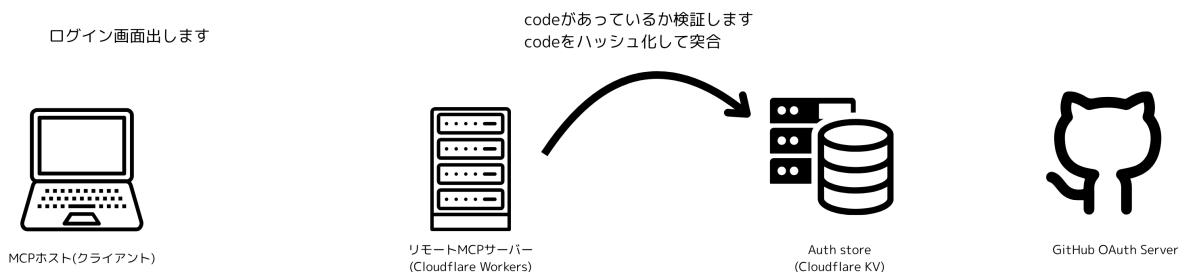
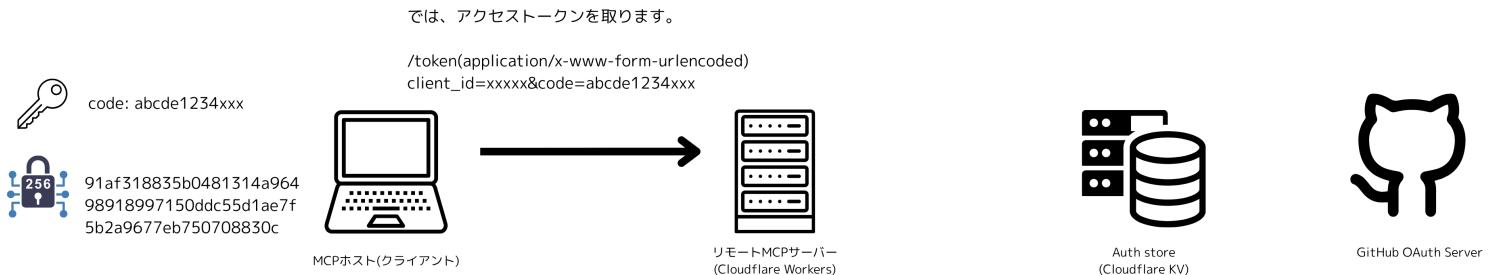
[https://mcp-qithub-oauth-demo.tubone24.workers.dev/auth?response\\_type=code&client\\_id=cac4099c-41ef-40d2-a9c4-46ab690a93de&code\\_challenge=srTVEx0Uw18aP\\_DALPi6ufbjefzTlqlfaHqB2ugC-Ys&code\\_challenge\\_method=S256&redirect\\_uri=http%3A%2F%2F127.0.0.1%3A6274%2Foauth%2Fcallback%2Fdebug&scope=read%3Auser+user%3Aemail](https://mcp-qithub-oauth-demo.tubone24.workers.dev/auth?response_type=code&client_id=cac4099c-41ef-40d2-a9c4-46ab690a93de&code_challenge=srTVEx0Uw18aP_DALPi6ufbjefzTlqlfaHqB2ugC-Ys&code_challenge_method=S256&redirect_uri=http%3A%2F%2F127.0.0.1%3A6274%2Foauth%2Fcallback%2Fdebug&scope=read%3Auser+user%3Aemail)

Click the link to authorize in your browser. After authorization, you'll be redirected back to continue the flow.

# フローにするとこんな感じですが、詳細は割愛



# トークン取得処理でcodeの検証を実施 これがPKCE



# 閑話休題

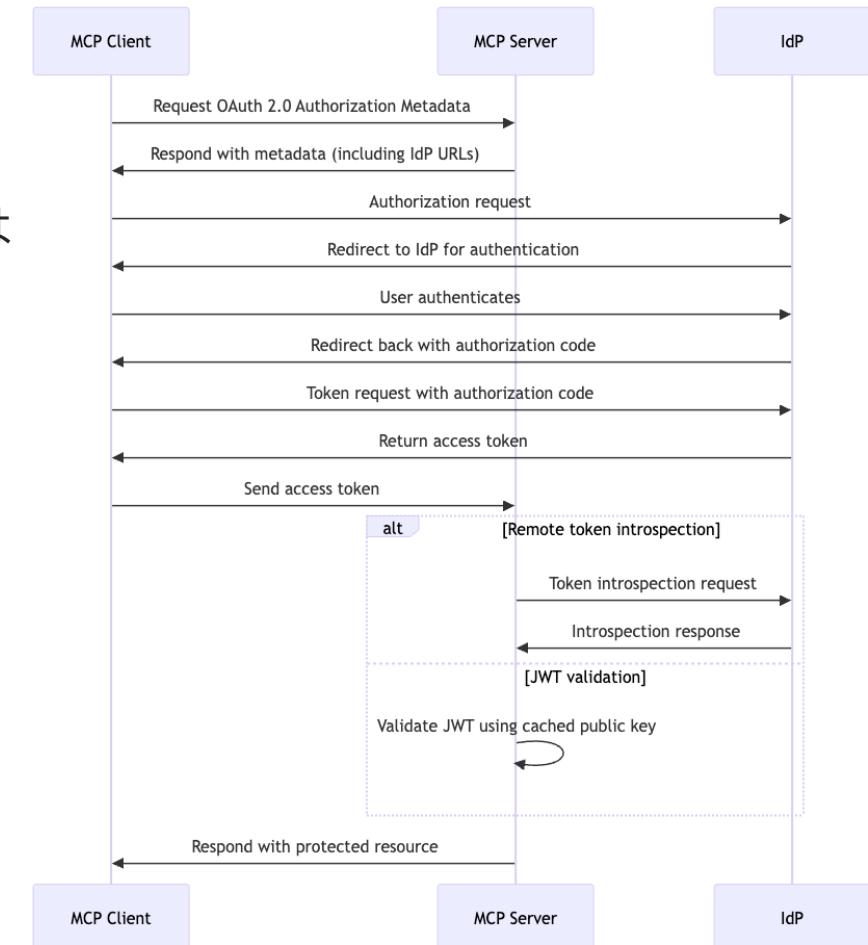
## 認証サーバーとリソースサーバーってくっつけていいの？

### 「ベストプラクティスじゃない」

が現状の規格だと、MCPサーバーがトークンの発行まで実施  
ベストプラクティスに従えば、認証サーバーとリソースサーバーは  
分けるべき。

よって下記のIssueなどにより良い方式が今なお、  
議論がされているところです。

<https://github.com/modelcontextprotocol/modelcontextprotocol/issues/205>





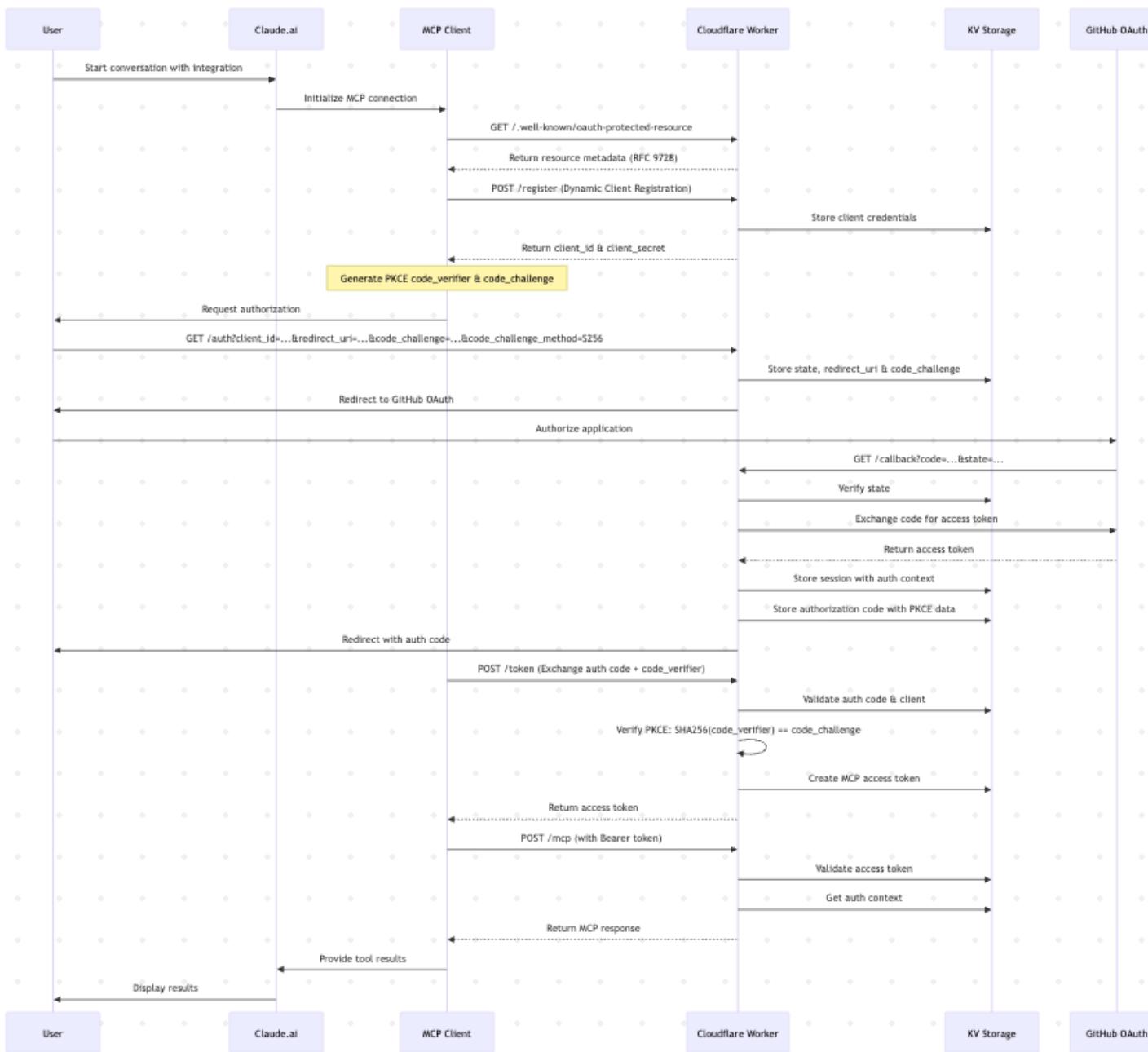
Be a Change Leader.

アジャイルに力を与え

共に成長し続ける社会を創る

# Appendix

# ちゃんとしたシーケンス



## デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

h-demo.tubone24.workers.dev/mcp

> Authentication

Server Entry Servers File

> Configuration

Connect

Disconnected

System

?

?

?

Connect to an MCP server to start inspecting

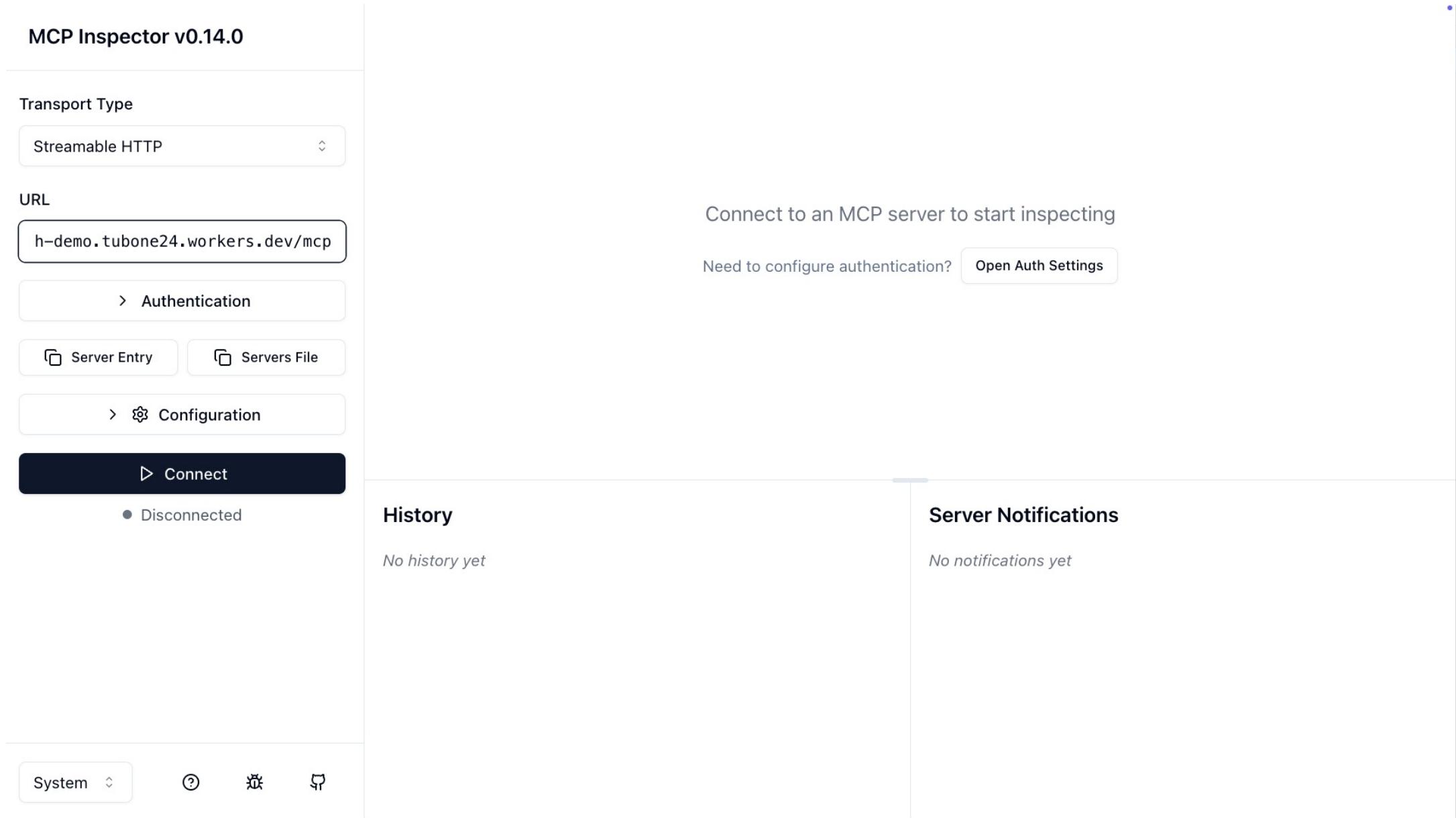
Need to configure authentication? Open Auth Settings

History

No history yet

Server Notifications

No notifications yet



## デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

[https://mcp-github-oauth-demo.](https://mcp-github-oauth-demo)

> Authentication

Server Entry Servers File

> Configuration

▷ Connect

● Disconnected

System ⚙️ ⓘ ⚡ 🛡️

### OAuth Flow Progress

Follow these steps to complete OAuth authentication with the server.

- Metadata Discovery
- Client Registration
- Preparing Authorization
- Request Authorization and acquire authorization code
- Token Request
- Authentication Complete

**Continue**

### History

No history yet

### Server Notifications

No notifications yet

## デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

[https://mcp-github-oauth-demo.](https://mcp-github-oauth-demo)

> Authentication

Server Entry Servers File

> Configuration

▷ Connect

● Disconnected

System ⚙️ ⓘ ⚡ 🛡️

### OAuth Flow Progress

Follow these steps to complete OAuth authentication with the server.

- Metadata Discovery
- Client Registration
- Preparing Authorization
- Request Authorization and acquire authorization code
- Token Request
- Authentication Complete

**Continue**

### History

No history yet

### Server Notifications

No notifications yet

## デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

<https://mcp-github-oauth-demo.tubone24.workers.dev>

> Authentication

Server Entry Servers File

> Configuration

Connect

Disconnected

System ⚙️ ⓘ ⚡ ⚡

### OAuth Flow Progress

Follow these steps to complete OAuth authentication with the server.

Metadata Discovery

OAuth Metadata Sources

Resource Metadata:

From <https://mcp-github-oauth-demo.tubone24.workers.dev/.well-known/oauth-protected-resource>

```
[{"id": "resource_id", "name": "Resource Name", "display_name": "Resource Name", "description": "Resource Description", "scopes_supported": ["read:user", "user:email"], "bearer_methods_supported": ["header", "query"], "resource_documentation": "https://github.com/cloudflare/workers-oauth-provider", "resource_policy_uri": "https://mcp-github-oauth-demo.tubone24.workers.dev/terms", "resource_tos_uri": "https://mcp-github-oauth-demo.tubone24.workers.dev/terms", "mcp_version": "2024-11-05", "mcp_endpoints": {"sse": "https://mcp-github-oauth-demo.tubone24.workers.dev/sse", "http": "https://mcp-github-oauth-demo.tubone24.workers.dev/mcp"}]}
```

History

No history yet

Server Notifications

No notifications yet

## デモがうまく行かなかった用

MCP Inspector v0.14.0

Transport Type

Streamable HTTP

URL

https://mcp-github-oauth-demo.

> Authentication

Server Entry Servers File

> Configuration

> Connect

● Disconnected

System ⚙️ ⓘ ⚡ ⚡

Metadata Discovery

▶ OAuth Metadata Sources

Client Registration

Registered Client Information

```
{  
    "redirect_uris": [  
        "http://127.0.0.1:6274/oauth/callback/debug"  
    ],  
    "grant_types": [  
        "authorization_code",  
        "refresh_token"  
    ],  
    "response_types": [  
        "code"  
    ],  
    "client_name": "MCP Inspector",  
    "scope": "read:user user:email",  
    "client_id": "1[REDACTED]",  
    "client_secret": "1[REDACTED]",  
    "client_id_issued_at": 1749733495,  
    "client_secret_expires_at": 0  
}
```

Preparing Authorization

History

No history yet

Server Notifications

No notifications yet

デモがうまく行かなかった用

Please copy this authorization code and return to the Auth Debugger:

ea[REDACTED]

Close this tab and paste the code in the OAuth flow to complete authentication.

## デモがうまく行かなかった用

The screenshot shows the MCP Inspector v0.14.0 interface. On the left, the main panel displays the configuration for a Streamable HTTP connection to the URL <https://mcp-github-oauth-demo>. It includes sections for Authentication, Configuration, and a large Connect button. A status message indicates it is Disconnected. On the right, a detailed view of the OAuth process is shown:

- Authorization Code**: A redacted authorization code is pasted into the input field.
- Token Request**: The checkbox is checked, and the Token Endpoint is set to <https://mcp-github-oauth-demo.tubone24.workers.dev/token>.
- Authentication Complete**: The checkbox is checked, and the Access Tokens section is expanded, displaying the following JSON object:

```
{  "access_token": "redacted",  "token_type": "Bearer",  "expires_in": 3600,  "scope": "read:user user:email"}
```
- History**: No history yet.
- Server Notifications**: No notifications yet.

## デモがうまく行かなかった用

The screenshot shows the MCP Inspector v0.14.0 application interface. On the left, the main configuration panel includes fields for Transport Type (Streamable HTTP), URL (https://mcp-github-oauth-demo.), and various navigation buttons like Authentication, Server Entry, Servers File, Configuration, Reconnect, and Disconnect. A green 'Connected' status indicator is present. At the bottom, there are system-related buttons for System, Help, Settings, and Logout.

The central part of the interface features a navigation bar with tabs: Resources, Prompts, Tools (selected), Ping, Sampling, Roots, and Auth. Below this is a 'Tools' section containing 'List Tools' and a 'Clear' button. It lists three tools: 'get\_user\_info', 'calculate', and 'get\_github\_repos'. The 'get\_user\_info' tool is selected, showing its description: '認証されたユーザーの情報を取得'. A large 'Run Tool' button is visible. The tool result is displayed as 'Tool Result: Success' with the output: "ユーザー情報: - ユーザー名: tubone24 - 名前: tubone(Yu Otsubo) - メール: 未設定".

At the bottom, there are two sections: 'History' showing recent tool calls (7. tools/call, 6. tools/call, 5. tools/call) and 'Server Notifications' indicating 'No notifications yet'.