MCP

音楽CLINE

MCP SERVER MIDI COMPOSER

Who are you?

Yu Otsubo (tubone24)

ゆるふわエンジニア@KAG

妻と犬の二人暮らし。
アスパラが苦手ですが、春になると
飲食店アスパラを使ったメニュー
ばっかりでうんざりしている。

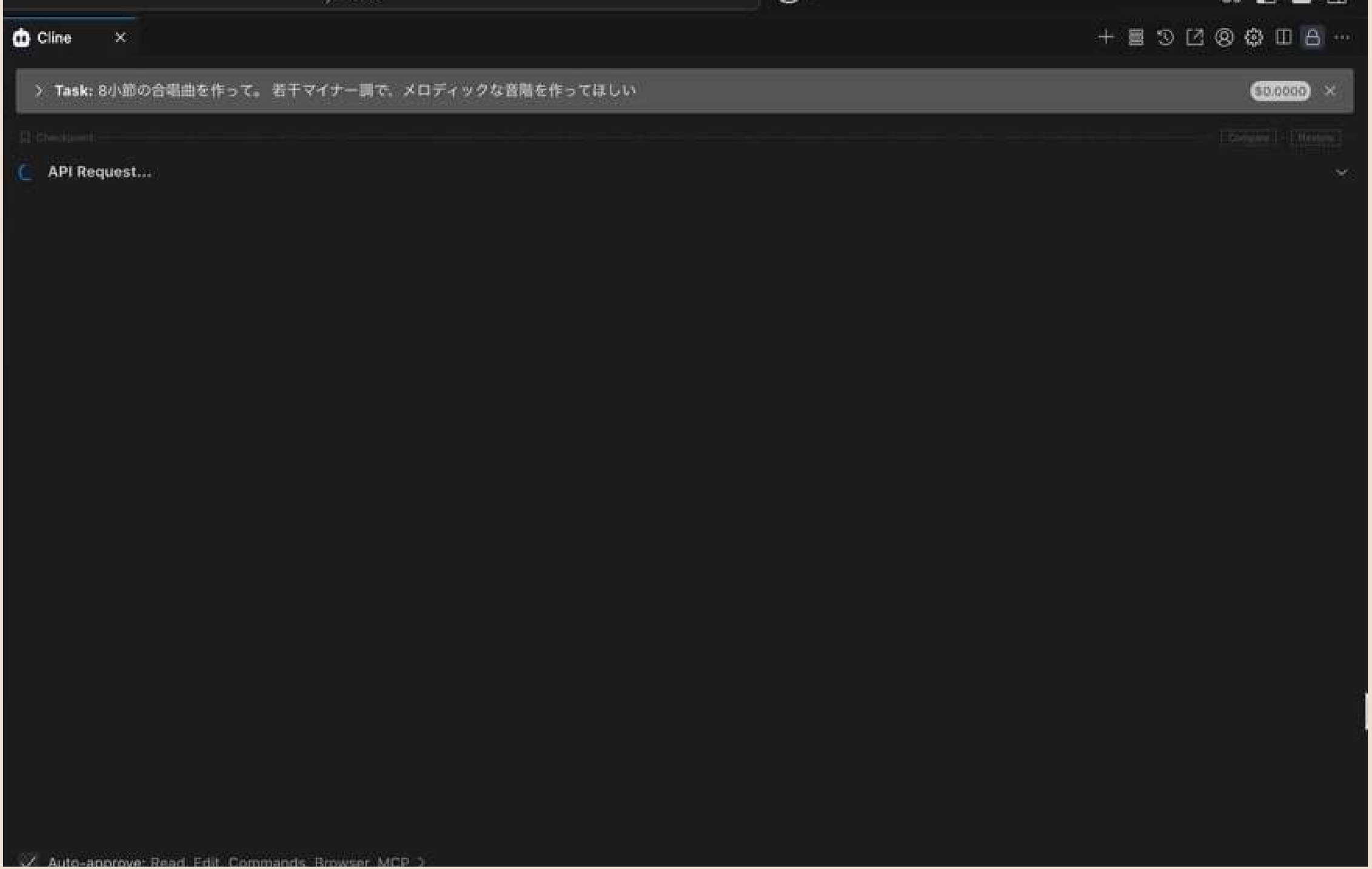
```
$ npx tubone24
```

What is a MCP?

MCPとは、AI（人工知能）モデルと外部
の
め
の
M
の略
和
B-C
ポートのような役割を果たします。

これは割愛

DEMO


Cline
Task: 8小節の合唱曲を作って。若干マイナー調で、メロディックな音階を作ってほしい
$0.0000
API Request...
Auto-approve: Read, Edit, Commands, Browser, MCP

Cline

> **Task:** 8小節の合唱曲を作って。若干マイナー調で、メロディックな音階を作ってほしい $0.0104

- メロディックな音階を使用

まず、合唱曲を作るためにMCPサーバーの「musicComposer」を使用します。このサーバーには「create_midi」というツールがあり、テキスト形式の音楽データからMIDIファイルを生成できます。

合唱曲なので、複数のボーカルパートを作成する必要があります。一般的な合唱の構成としては、ソプラノ、アルト、テナー、バスの4パートが考えられます。

若干マイナー調ということで、自然的マイナースケール（ナチュラルマイナー）ではなく、メロディックマイナースケールを使用します。メロディックマイナーは上行時に6度と7度を半音上げ、下行時は自然的マイナーに戻るスケールです。これにより、メロディアスな響きが得られます。

Aマイナーのメロディックマイナースケールを使用することにします：

- 上行時: A, B, C, D, E, F#, G#, A
- 下行時: A, G, F, E, D, C, B, A

8小節の簡単な構成を考えます：

1. 導入部（1-2小節）
2. メロディの展開（3-6小節）
3. 終結部（7-8小節）

それでは、MIDIファイルを生成するためのデータを作成しましょう。

MIDIファイルを作成するために、musicComposerのMCPサーバーを使用します。若干マイナー調でメロディックな音階を使った8小節の合唱曲を作成します。

Aマイナーのメロディックマイナースケールを基調とし、ソプラノ、アルト、テナー、バスの4声部で構成します。

Cline wants to use a tool on the `musicComposer` MCP server:

create_midi — Auto-approve

テキスト形式の音楽データからMIDIファイルを生成します。compositionが長くなる場合はinputSchemaを一度JSONファイルに書き出して読み込んで渡すようにすると安定します。

ARGUMENTS

```
{
  "title": "メロディックマイナーの合唱曲",
  "composition": {
    "bpm": 72,
    "timeSignature": {
      "numerator": 4
```

Auto-approve: Read, Edit, Commands, Browser, MCP

Cline

Task: 8小節の合唱曲を作って。若干マイナー調で、メロディックな音階を作ってほしい

$0.0539

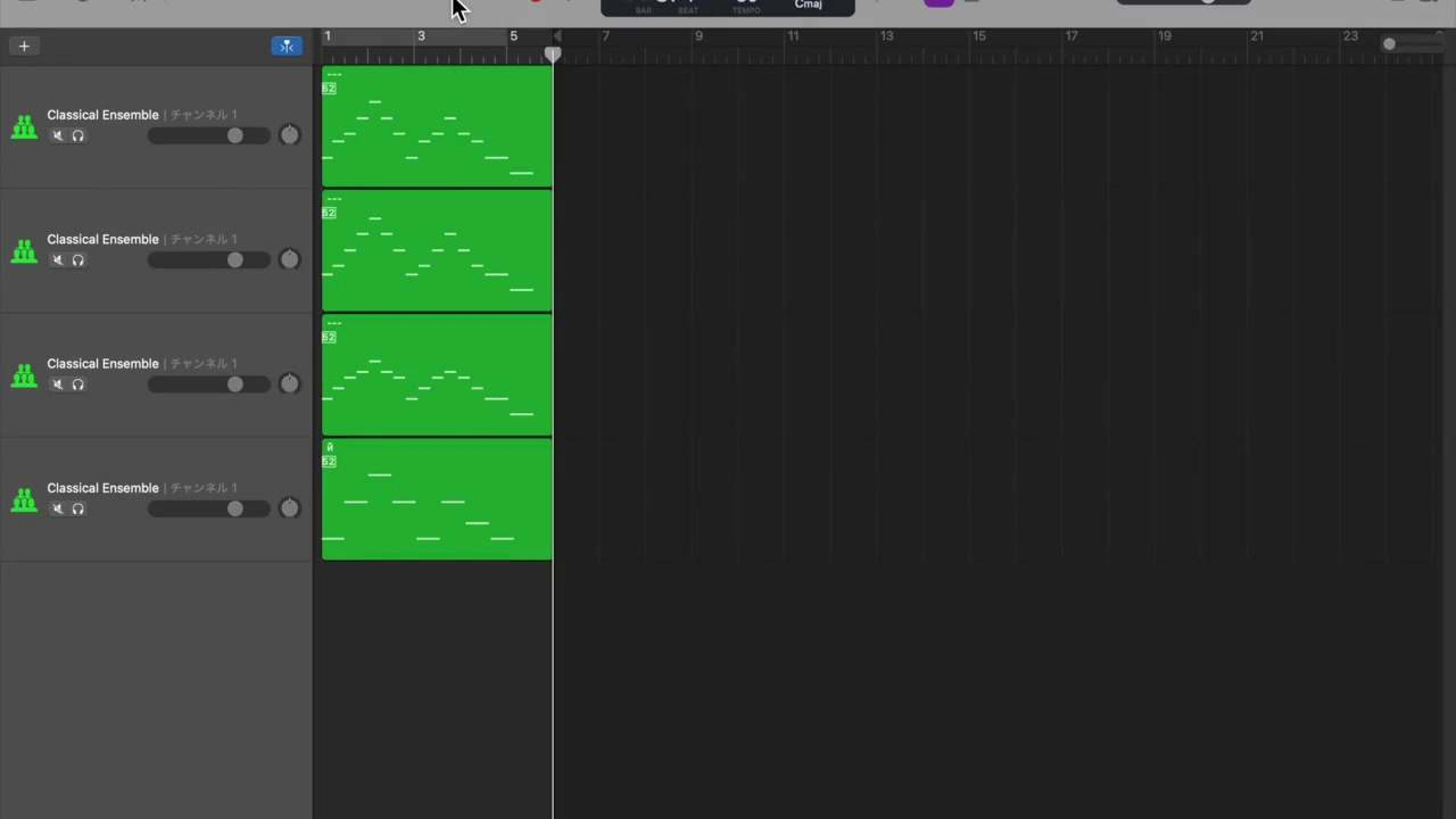

```
        { "pitch": 64, "beat": 13.0, "duration": "2", "velocity": 78 },
        { "pitch": 62, "beat": 15.0, "duration": "2", "velocity": 75 }
      ]
    },
    {
      "name": "テナー",
      "instrument": 53,
      "notes": [
        { "pitch": 60, "beat": 1.0, "duration": "4", "velocity": 75 },
        { "pitch": 59, "beat": 2.0, "duration": "4", "velocity": 72 },
        { "pitch": 60, "beat": 3.0, "duration": "4", "velocity": 75 },
        { "pitch": 62, "beat": 4.0, "duration": "4", "velocity": 78 },
        { "pitch": 64, "beat": 5.0, "duration": "2", "velocity": 80 },
        { "pitch": 65, "beat": 7.0, "duration": "2", "velocity": 78 },
        { "pitch": 64, "beat": 9.0, "duration": "2", "velocity": 75 },
        { "pitch": 62, "beat": 11.0, "duration": "2", "velocity": 72 },
        { "pitch": 60, "beat": 13.0, "duration": "2", "velocity": 75 },
        { "pitch": 57, "beat": 15.0, "duration": "2", "velocity": 72 }
      ]
    },
    {
      "name": "バス",
      "instrument": 53,
      "notes": [
        { "pitch": 45, "beat": 1.0, "duration": "1", "velocity": 85 },
        { "pitch": 48, "beat": 5.0, "duration": "1", "velocity": 82 },
        { "pitch": 52, "beat": 9.0, "duration": "1", "velocity": 80 },
        { "pitch": 45, "beat": 13.0, "duration": "1", "velocity": 85 }
      ]
    }
  ]
},
"output_path": "/Users/kagadminmac/Documents/Cline/MCP/test/melodic_minor_chorus.mid"
}
```


Response

Rich Display

「メロディックマイナーの合唱曲」のMIDIファイルを生成しました。ファイルは /Users/kagadminmac/Documents/Cline/MCP/test/melodic_minor_chorus.mid に保存されました。

Auto-approve: Read, Edit, Commands, Browser, MCP


Classical Ensemble | チャンネル 1
Classical Ensemble | チャンネル 1
Classical Ensemble | チャンネル 1
Classical Ensemble | チャンネル 1
1
3
5
7
9
11
13
15
17
19
21
23

これだけ

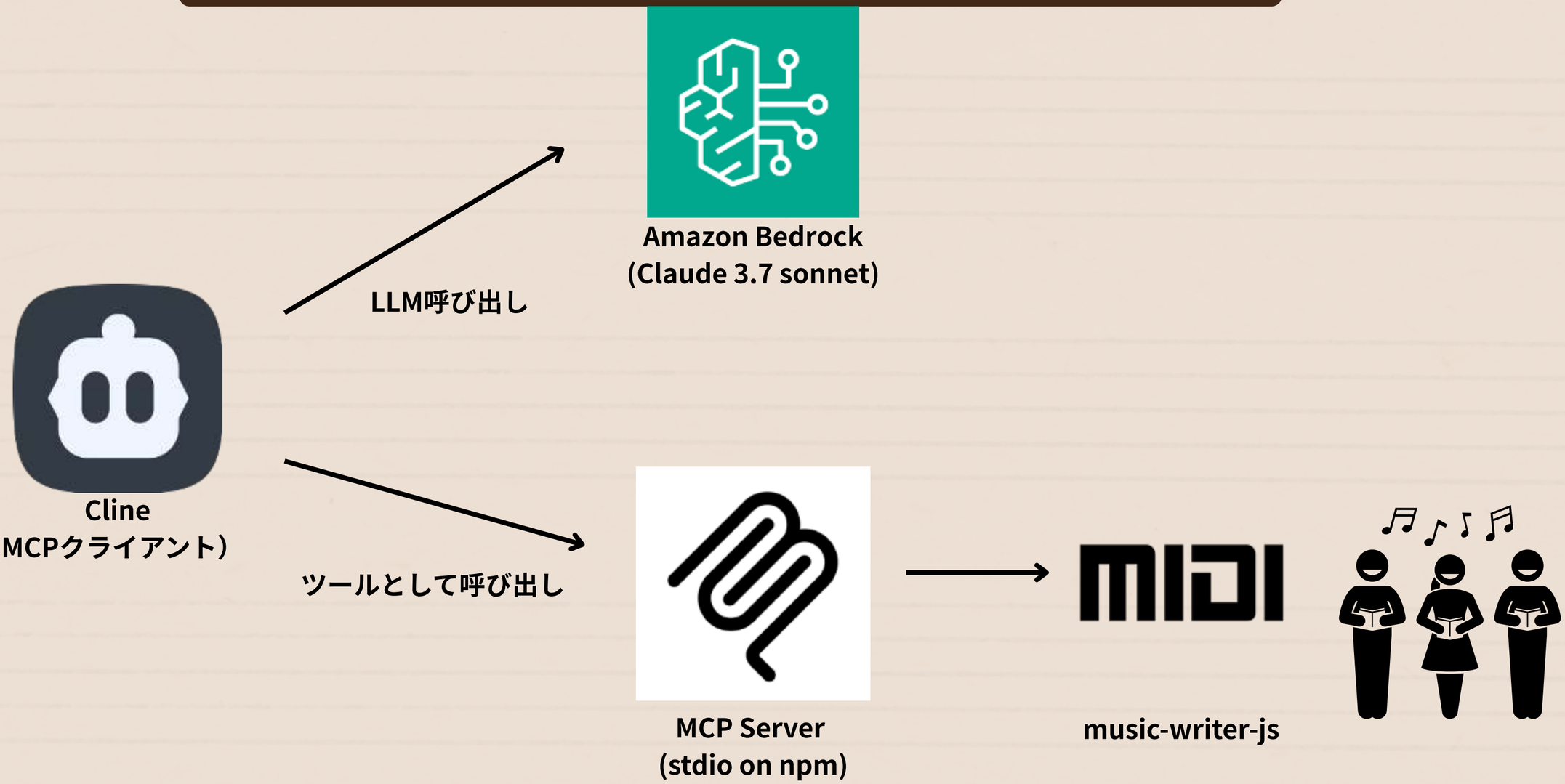
（暗すぎる）

Conclusion

音楽生成はテキスト生成系の
LLMではなく、

素直に音楽生成AI使おう。

どういう仕組み？


Amazon Bedrock
(Claude 3.7 sonnet)
LLM呼び出し
Cline
(MCPクライアント)
ツールとして呼び出し
MCP Server
(stdio on npm)
music-writer-js

tubone24/**midi-mcp-server**

MIDI MCP Server is a Model Context Protocol (MCP) server that enables AI models to generate MIDI files from text-based...

1 Contributor

0 Issues

0 Stars

0 Forks

自作して
わかったこと

注意

今回はMCPのStdIOのみ
Toolsに関してのみ。

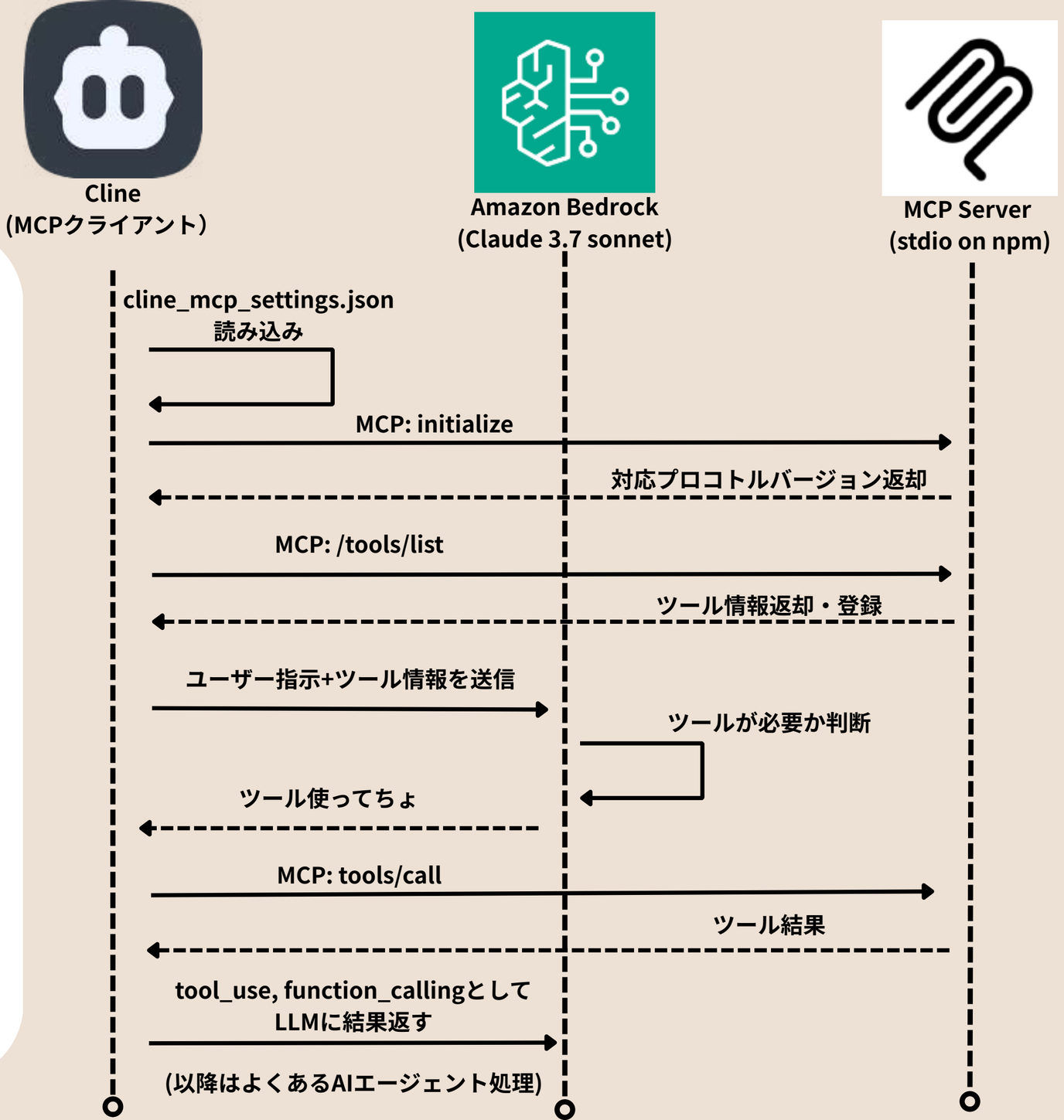
promptsとかも面白そうではある

Sequences

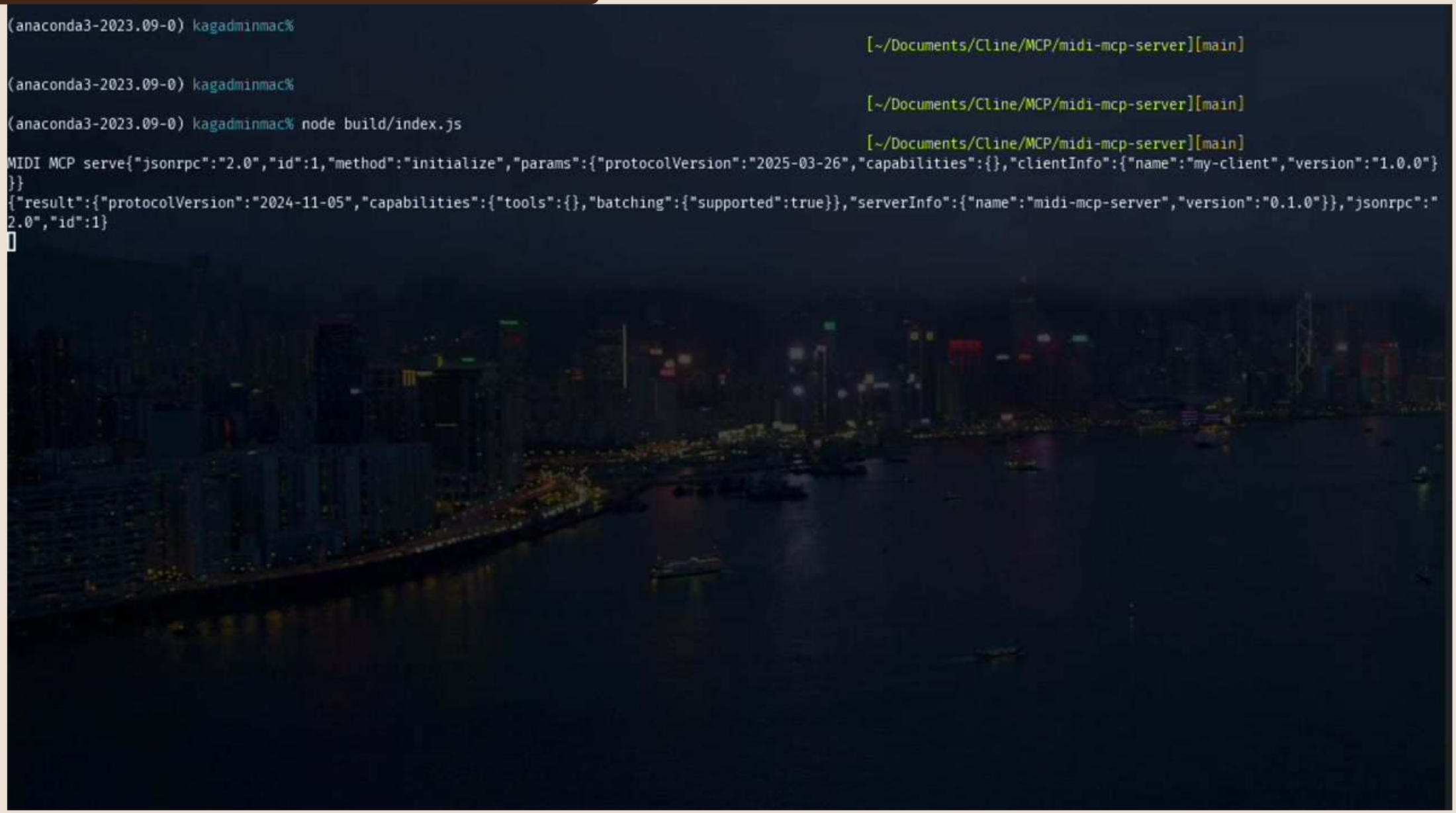
ざっくりシーケンス書くと
こんな感じ
(notificationなど省略)

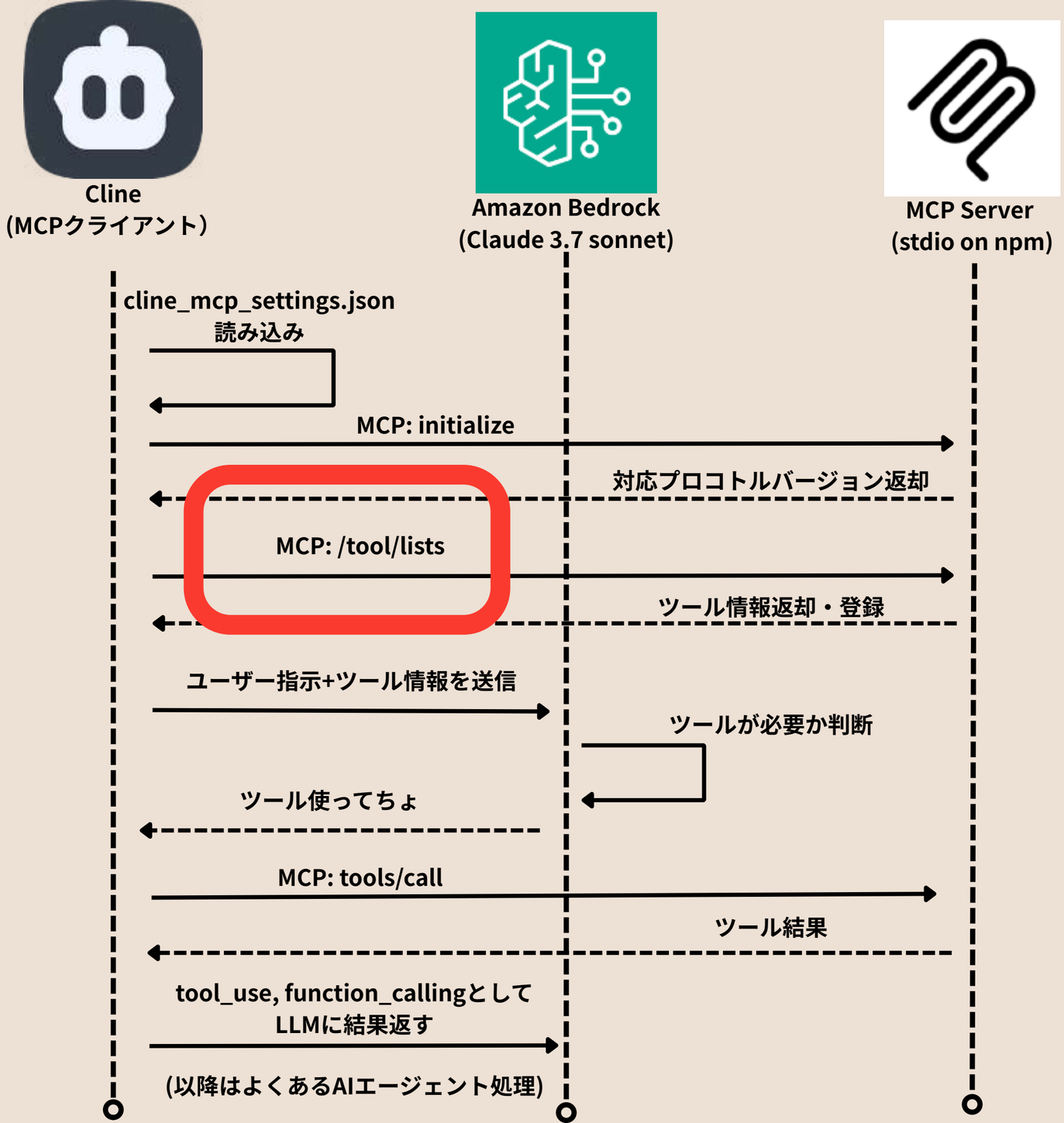
MCPに規定されるメソッドの
ハンドラーを
MCPサーバーコードに表現
していく

StdIOなら直接JSをnodeから
実行すれば結構面白いことがわかる



MCP Server STDIO




Cline
(MCPクライアント)
Amazon Bedrock
(Claude 3.7 sonnet)
MCP Server
(stdio on npm)
cline_mcp_settings.json
読み込み
MCP: initialize
対応プロコトルバージョン返却
MCP: /tool/lists
ツール情報返却・登録
ユーザー指示+ツール情報を送信
ツールが必要か判断
ツール使ってちょ
MCP: tools/call
ツール結果
tool_use, function_callingとして
LLMに結果返す
(以降はよくあるAIエージェント処理)

Input Schema

ListToolsRequestSchemaで
次のようにスキーマを正しく
指定してあげればツール引数の
構文不正で落ちることはほぼない。

Claudeさまさま。

Input Schema

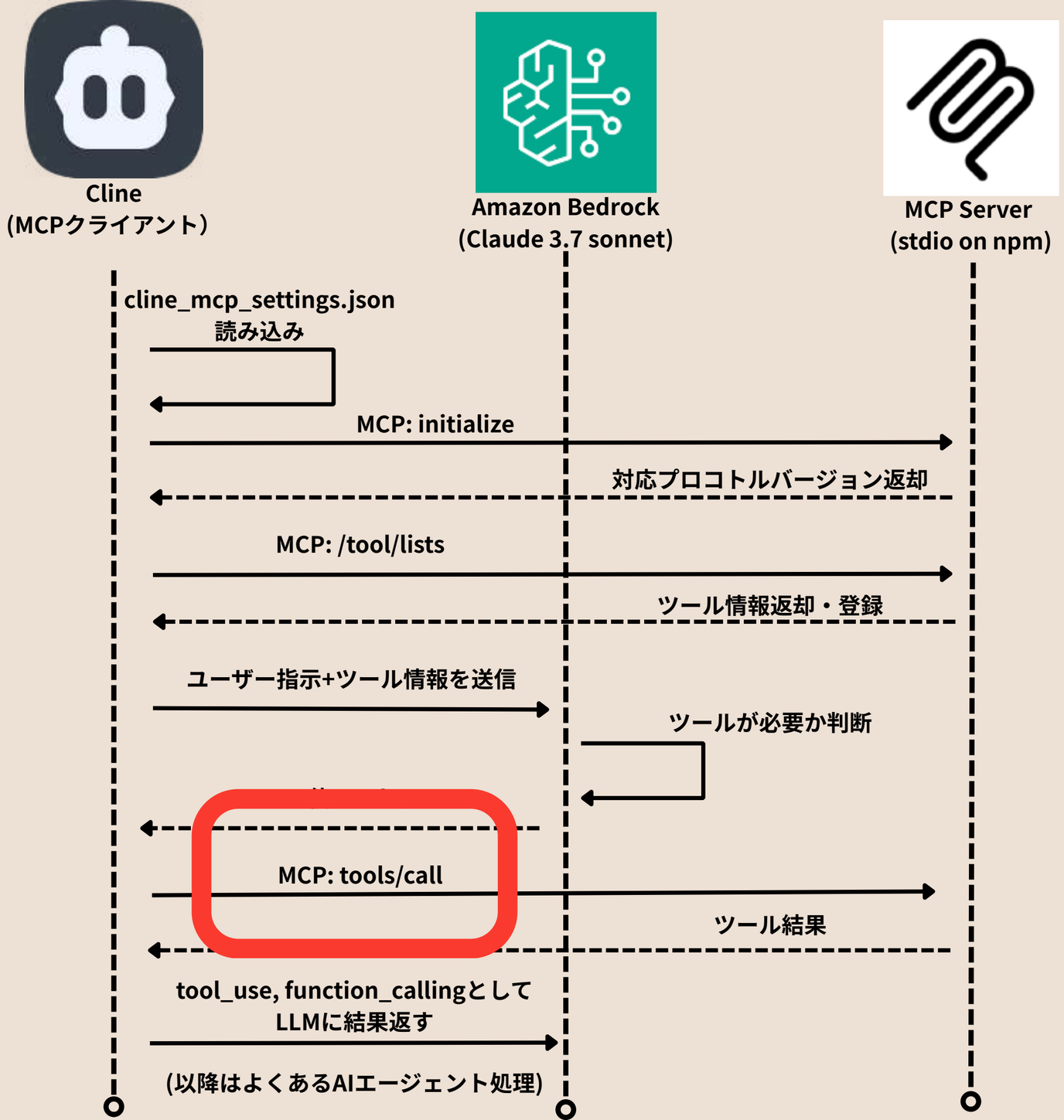
ただし、ツール引数が長くなれば
なるほど、LLMのMaxTokenに
ひっっかりやすくなる

引っかかった場合は最初から
やり直しになるので注意
（ループを作っちゃう原因に）

Input Schema

長いInputならClineであれば
一度ファイルに
出力しそちらを参照するように
するとよさそう。

InputSchemaで
「どちらか一方を使ってね」
という指定も可能(oneOf)


Cline
(MCPクライアント)
Amazon Bedrock
(Claude 3.7 sonnet)
MCP Server
(stdio on npm)
cline_mcp_settings.json
読み込み
MCP: initialize
対応プロコトルバージョン返却
MCP: /tool/lists
ツール情報返却・登録
ユーザー指示+ツール情報を送信
ツールが必要か判断
MCP: tools/call
ツール結果
tool_use, function_callingとして
LLMに結果返す
(以降はよくあるAIエージェント処理)

Output files

ファイルを出力させるとき、
MCPサーバーには現在動いている
ディレクトリ情報がわからない。
なので、rootディレクトリにファイルを
作ったりしちゃう

一応Feature Requestは上がっている。
https://github.com/cline/cline/discussions/2635

Output files

なので絶対パスでファイルを
指定させると安定する。

こちらもInputSchemaで指示を
通すことができる。

Conclusion

MCPのLTのために
変なServer作ってしまった。

自作MCP Serverで
おもしろそうなアイディアが
浮かばないので
一緒に考えてほしいです！！

THANK YOU FOR LISTENING!

See you next LT!