

# Microsoft

Microsoft BUILD 2025

## 🤖 AI エージェントと MCP (Model Context Protocol)

AIエージェント時代の幕開け

MICRO  
SOFT GitHub

# 曲AIエージェントとは

「今年のBuildのテーマは「AIエージェント時代」です。従来のAIモデルからAIエージェントへの進化は重要なステップです。AIエージェントは、ユーザーに代わって行動しタスクを実行するAIシステムです。」

- Satya Nadella CEO, Microsoft BUILD 2025

出典: [Microsoft Build 2025基調講演まとめ：テーマは「AIエージェント時代」 - ITmedia NEWS](#)



## AIエージェントの定義

AIエージェントとは、ユーザーの代わりに自律的に行動し、タスクを実行する人工知能システムです。従来の受動的なAIと異なり、能動的に判断し行動します。

## 主な特徴

- 自律性：与えられた目標に向けて独自に判断・行動
- 目的指向：特定のタスクや目標達成に特化
- 環境認識：コンテキストを理解し適切に対応
- 連携能力：他のシステムやエージェントと協調



# GitHub Copilot Agent Mode | Visual Studio Code

## ⚡ GitHub Copilot Agent Mode

VSCode内で動作するAIエージェントで、コーディングのサポートだけでなく、質問応答や提案を通じて開発をアシストします。

- ・自然言語での質問に対するコード生成
- ・コードベースの理解と文脈に応じた提案
- ・バグ修正、リファクタリング、最適化の提案
- ・タスクの自動化とプロジェクト管理支援



出典: [Microsoft Visual Studio Blog](#)

## 類似ツール比較

VSCode用のAIコーディングアシスタントは増加傾向にあり、様々な特徴を持っています。

Cline

AI エージェントと MCP (Model Context Protocol) - Microsoft BUILD 2025

エディタ内でのAI支援に特化し、

>-

Cursor

Genspark で作成

VSCodeベースのAI特化エディタで、コ

# GitHub Copilot Coding Agent | GitHub

## ■ Coding Agentの概要

Microsoft BUILD 2025で発表された、GitHub Copilotの新機能です。従来のコード補完を超えて、より高度なコーディングタスクを自律的に実行します。

- 複雑なコード生成をエンドツーエンドで実行
- コードベースを理解し、自律的に問題解決
- ユーザーとの対話を通じて要件を明確化
- 複数のファイルにまたがる変更も管理

## ♪ 主な機能と特徴

### 🔍 コード分析

リポジトリ全体を解析し、コンテキストを理解

### ✍️ テスト生成

自動的にテストを作成・実行し検証

### 👉 リファクタリング

コードベースの最適化と再構築を支援

### 🗣️ 対話型開発

自然言語での会話を通じた開発プロセス



出典: [Publickey](#)

## ▣ 開発者への影響

- 開発効率の大幅向上：複雑な実装タスクを自動化することで、開発者はより創造的な作業に集中できる
- 学習曲線の短縮：新しい言語やフレームワークの学習を加速し、実装例を自動生成
- コードレビュー支援：潜在的な問題を事前に検出し、改善案を提案



「Coding Agentは単なるツールではなく、開発チームの一員として機能し、ソフトウェア開発プロセス全体を変革します」

# MCP (Model Context Protocol) とは

## MCPの基本概念

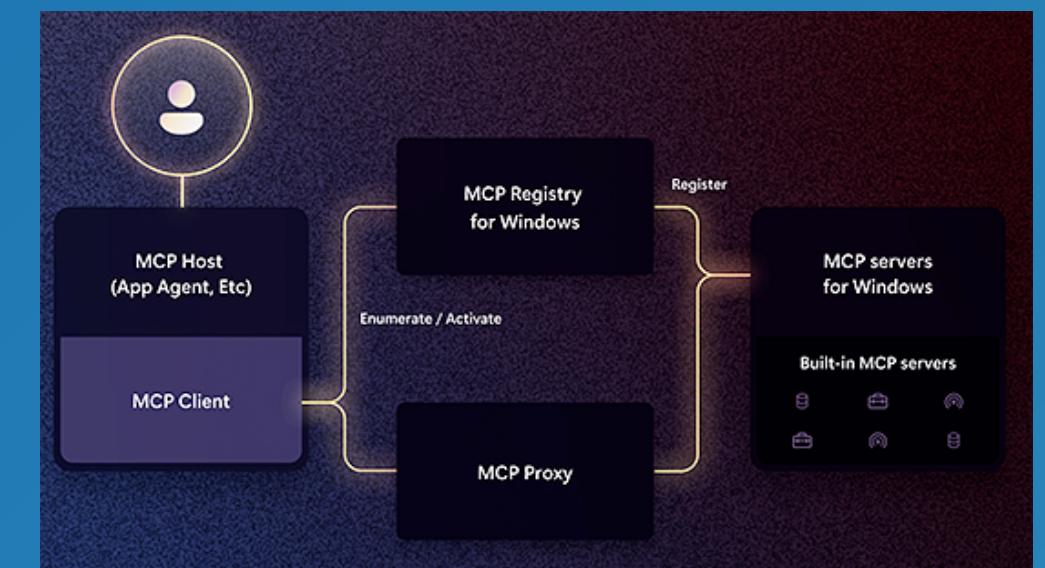
MCP (Model Context Protocol) は、AIエージェント間での滑らかな情報交換と連携を可能にするプロトコルです。Microsoft BUILD 2025で重点的に発表されました。

- 異なるAIエージェント間でのコンテキスト共有
- 複数のアプリケーション・サービス間でのシームレスな連携
- OSレベルでのAIエージェントサポート
- AIエコシステムの標準プロトコルを目指す

## Microsoftの主な発表内容

- MCPサーバーレジストリの提供
- Windows上のMCPサポート
- Microsoft 365でのMCPサポート

## MCP概念図



出典: [Publickey - Microsoft、WindowsがMCPをサポートすると発表](#)

## Microsoftの発表内容

「Microsoftは、AIエージェント同士が連携するためのプロトコルであるMCP (Model Context Protocol) を、Windows、Microsoft 365など主要製品が全面的にサポートすることを発表しました。これによりAIエージェント間での情報交換と連携が可能になります。」

MCPは、マイクロサービスがHTTP/HTTPSで連携するように、サービス、OS、アプリケーション、Webページ、AIエージェントを接続します。アプリケーションやサービスの性質は大きく変化を強いられることになるでしょう。

 MCPの登場により、AIエージェントはアプリケーション間、デバイス間を移動しながら、ユーザーの意図を理解し、タスクを代行できるようになります。

# マルチエージェント オーケストレーション

## 概要

マルチエージェント オーケストレーションは、MCPを活用して複数のAIエージェントを連携させ、複雑なタスクをこなすための仕組みです。

- 複数のエージェントが連携して問題解決
- 各エージェントが得意な分野を担当
- コンテキストと情報の共有を実現
- ワークフローの自動化と最適化

## 主な利点

### 効率性の向上

複数のエージェントが並行して作業し、処理速度を向上

### 専門性の最大化

各エージェントが得意分野に特化し、全体の質を向上

### スケーラビリティ

必要に応じてエージェントを追加・削除して柔軟に対応



## MCPによるエージェント間連携の実現

### 統一プロトコル

MCPは異なるエージェント間の会話を標準化し、相互運用性を確保します

### コンテキスト共有

会話履歴や状態情報を維持し、一貫した体験を提供します

### プラグイン拡張

新機能やツールを動的に追加し、エージェント機能を拡張できます

### セキュリティ制御

エージェント間の通信と権限を適切に管理します

# MCPサーバー

## MCPサーバーの役割

### ↔ エージェント間のコミュニケーション仲介

複数のAIエージェントの対話を管理し、情報とコンテキストを適切に転送

### 💾 コンテキストの保存と管理

会話履歴やセッション情報を保持し、一貫したエージェント体験を実現

### 🛡️ セキュリティとアクセス制御

エージェント間の通信を保護し、適切な権限管理を提供



MCPサーバー

## MCPサーバー実装の主要コンポーネント

### 📦 コンテキストストア

会話履歴や状態情報を保持

### 🔌 プラグインシステム

機能拡張の仕組み

### 📍 ルーティング

要求の適切な処理先決定

### 👤 認証・認可

アクセス管理と保護

## MCPサーバー実装

- JavaScript/TypeScript
- Python
- .NET (C#)
- Go言語

💡 MCPはオープンプロトコルなので、様々な言語で実装可能

## ▣ 利用シナリオ

- 開発環境との連携 (VSCode等)
- 社内チャットボットの拡張
- 複数AIの統合管理

## 🚀 今後の展望

Microsoftが提供する「MCPサーバーレジストリ」により、エージェントの発見と連携が容易になります。Windows組み込みのMCPサポートにより、OSレベルでのAIエージェント活用が進むでしょう。

# Claude Desktop での利用

## Claude Desktop と MCP

Claude Desktop アプリケーションは MCP プロトコルをサポートし、他のエージェントやアプリケーションと連携できます。

- エンタープライズ AI アシスタントとの連携
- 社内ツールやナレッジベースへのアクセス
- セキュアなデータ共有と処理
- カスタマイズされた企業用 AI エクスペリエンス

## 設定方法

### 1 Claude Desktop 設定を開く

右上の歯車アイコンから設定画面にアクセス

### 2 MCP 接続設定を選択

「開発者オプション」から「MCP 設定」を選択

### 3 MCP サーバーを登録

MCP サーバーの URL と認証情報を入力

## Claude Desktop デモ

Claude Desktop

c

こんにちは！MCP サーバーを通じて他のエージェントと連携できます。何をお手伝いしましょうか？

コードエディタとの連携例を見せてください

c

VSCode との連携を開始します。MCP を通じて

u

## MCP による連携機能

### コード解析

VSCode と連携したコードの理解と提案

### 文書処理

Office 文書の分析と編集支援

### データ可視化

データ分析ツールとの連携

### 情報検索

企業内情報へのセキュアなアクセス

MCP の標準化により、Claude、Copilot、その他の AI エージェント間でシームレスな連携が可能になります。

# </> MCPサーバーの開発 | .NET での実装

## .NETでのMCP開発

Microsoftは.NET向けのMCPライブラリを提供しており、C#でMCPサーバーを容易に実装できます。

### 主要コンポーネント

- MCPサーバーミドルウェア
- コンテキスト管理システム
- エージェントプロバイダー
- 認証・認可モジュール

### 開発メリット

- ASP.NET Coreとの簡単な統合
- マイクロソフト製品との親和性
- エンタープライズ環境での高い安定性

## 実装ステップ

### 1. NuGetパッケージのインストール

```
dotnet add package Microsoft.MCP.Server
```

### 2. ASP.NET Coreプロジェクトの設定

### 3. MCPミドルウェアの追加

### 4. エージェント連携の実装

### 5. 認証機構の設定

## デモプロジェクト

完全なデモは GitHub リポジトリで公開されています：

[github.com/microsoft/mcp-server-samples](https://github.com/microsoft/mcp-server-samples)

## MCPサーバー実装サンプル (C#)

```
// Program.cs
var builder = WebApplication.CreateBuilder(args);

// MCPサービスの追加
builder.Services.AddMcpServer(options => {
    options.AgentProviders.Add(new MyCustomAgentProvider());
    options.ContextStore = new InMemoryContextStore();
    options.Authentication = new ApiKeyAuthProvider(
        builder.Configuration["McpApiKey"]);
});

var app = builder.Build();

// MCPミドルウェアの設定
app.UseRouting();
app.UseAuthentication();
app.UseAuthorization();
app.UseMcpServer();

app.Run();
```

## カスタムエージェントプロバイダー

```
// カスタムエージェントプロバイダーの実装
public class MyCustomAgentProvider : IAgentProvider
{
    public async Task<AgentResponse> Process(
        AgentRequest request,
        CancellationToken cancellationToken)
    {
        // AIモデルとの連携処理
        var response = await CallAiModel(
            request.Prompt,
            request.Context);

        return new AgentResponse {
            Content = response,
            Metadata = new Dictionary<string, string>()
        };
    }
}
```

## コンテキスト管理

```
// インメモリコンテキストストアの実装
public class InMemoryContextStore : IContextStore
{
    private readonly Dictionary<string, ContextData> _contexts = new();

    public Task<ContextData> GetContextAs(
        string contextId)
    {
        if (_contexts.TryGetValue(contextId, out var context))
        {
            return Task.FromResult(context);
        }
        return Task.FromResult(
            new ContextData());
    }

    public Task SaveContextAsync(
        string contextId,
        ContextData context)
    {
        _contexts[contextId] = context;
        return Task.CompletedTask;
    }
}
```

## MCPサーバーのテスト

MCPサーバーが起動したら、次のエンドポイントでアクセス可能：

```
https://localhost:5001/mcp/v1
```

 MCPクライアントからはこのURLを指定して接続します

デモ

# Visual Studio Codeでの利用

## VS CodeでのMCP連携

Visual Studio CodeはMCPプロトコルをサポートし、AIエージェントとの連携機能を提供します。拡張機能を使って簡単に設定できます。

### 主な機能

- コードコンテキストの共有
- AIエージェントとのリアルタイム連携
- 複数エージェント間での情報交換
- 拡張機能によるカスタマイズ

### 設定手順

#### 1. MCP拡張機能のインストール

拡張機能マーケットプレイスから「MCP Client」をインストール

#### 2. MCPサーバーの設定

settings.jsonにMCPサーバーのURLと認証情報を追加

#### 3. エージェント設定

使用したいAIエージェントを選択し接続設定を行う

### 開発者向けリソース

詳細なドキュメントとサンプルコード：  
<https://aka.ms/vscode-mcp-docs>

NEW

## VS Code MCP デモ

DEMO

app.js

```
const express = require('express');
const app = express();
const port = process.env.PORT || 3000;

// MCPクライアントの初期化
const { McpClient } = require('@microsoft/mcp-client');
const mcpClient = new McpClient({
  serverUrl: 'https://localhost:5001/mcp/v1',
  apiKey: process.env.MCP_API_KEY
});

// AIエージェントとの連携
app.post('/api/analyze', async (req, res) => {
  const context = { code: req.body.code };
  const response = await mcpClient.requestAgent(
    'code-analyzer',
    'コードを分析してください',
    context
  );
  res.json(response);
});
```

MCPクライアント連携例

Visual Studio Code v2.0

## VS Code + MCP 連携例



VS CodeでのMCP活用デモ  
クリックして再生（デモイメージ）

### コード生成

要件からコードを自動生成

### デバッグ支援

エラー原因の特定と修正

## マルチエージェント協調例

VS CodeでのMCP活用により、複数のAIエージェントが協調して作業を進めることができます。

</>  
コードエージェント  
コード生成

検索エージェント  
情報収集

セキュリティエージェント  
脆弱性確認