

# Agent Development Kit (ADK) の GUI で作る ローコード エージェント開発

Google Cloud

Developer Relations Engineer,

バンダリ ハレン



## About me



バンダリ ハレン

Developer Relations Engineer,  
Google Cloud Japan

## Intro

シニア デベロッパー リレーションズ エンジニアのバンダリ・ハレン (Bhandari Haren) です。主にGoogle Cloud (GCP) のAI関連サービスをご活用いただくためのハンズオンセッション、コードサンプル、ブログ記事などを開発者の皆様へお届けしています。

### SNS:



X: @HarenB16221

<https://www.linkedin.com/in/harendra-haren-bhandari-abb09924/>

<https://zenn.dev/bharen>

An abstract graphic of a vibrant red liquid splash or wave, originating from the bottom left corner and flowing upwards and to the right. The liquid has a glossy, reflective surface with highlights and shadows, creating a sense of motion and depth. It occupies the left side of the frame, partially overlapping the white background.

# Agenda

## Agenda

- AI エージェントのご紹介
- Google Cloud でエージェントの構築
  - ADK でエージェントの開発
  - ADK Visual Builder
- ハンズオンスタート



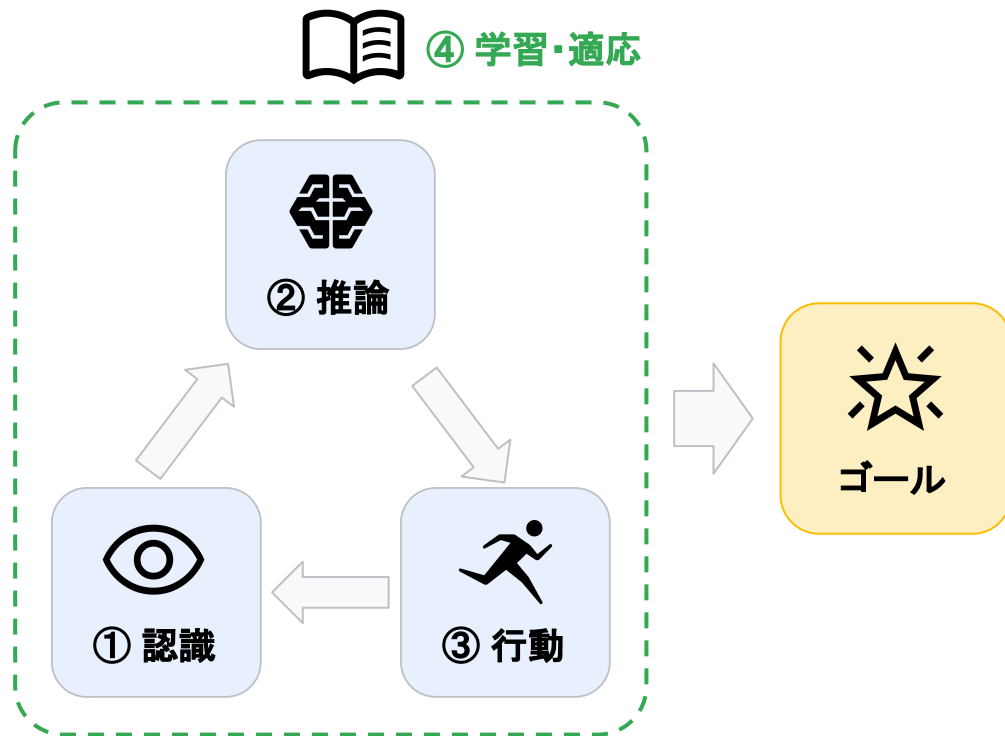
# AI エージェント のご紹介



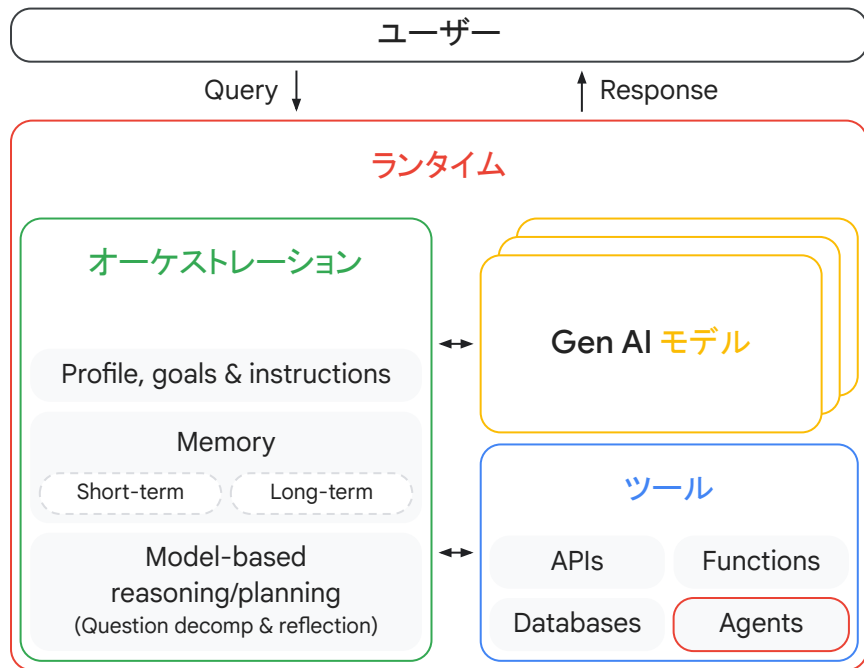
# AI エージェントとは？

ゴールを達成するために、環境を  
**認識**し、**推論・意思決定**を行い、  
自律的に行動するプログラム。

経験から**学習・適応**することも  
可能です。



# エージェントの アーキテクチャ



## AI Agent のキーコンポーネント



### モデル:

与えられた目的・指示に応じた推論、計画、およびレスポンス生成に用いる



### ツール:

他の API やサービス呼び出すことで、データを取得したり、アクションやトランザクションを実行する



### オーケストレーション:

エージェントのプロファイル(目的と指示)を保持しメモリ・セッション、ツール、提供取得されたデータなどを維持管理する



### ランタイム(実行環境):

ユーザーから呼び出されたときにエージェントを実行する環境

# AI エージェントのユースケース



ユーザー データ 分析



目的に向かって行動する



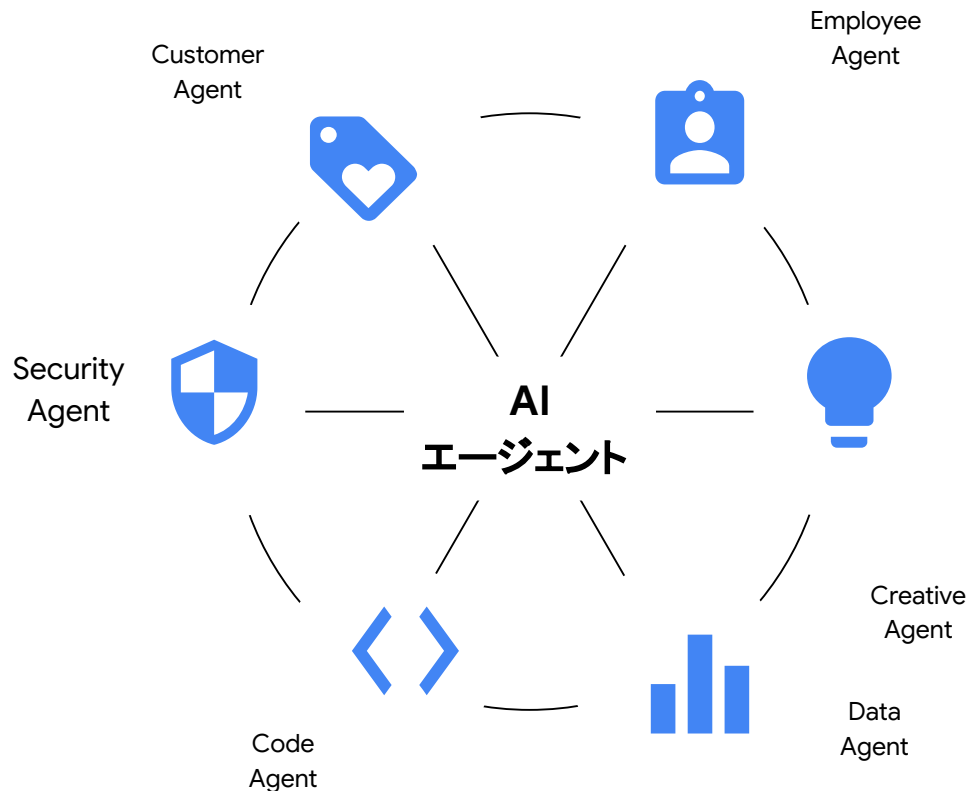
情報収集、生成



目的を達成するため計画、計画の実行

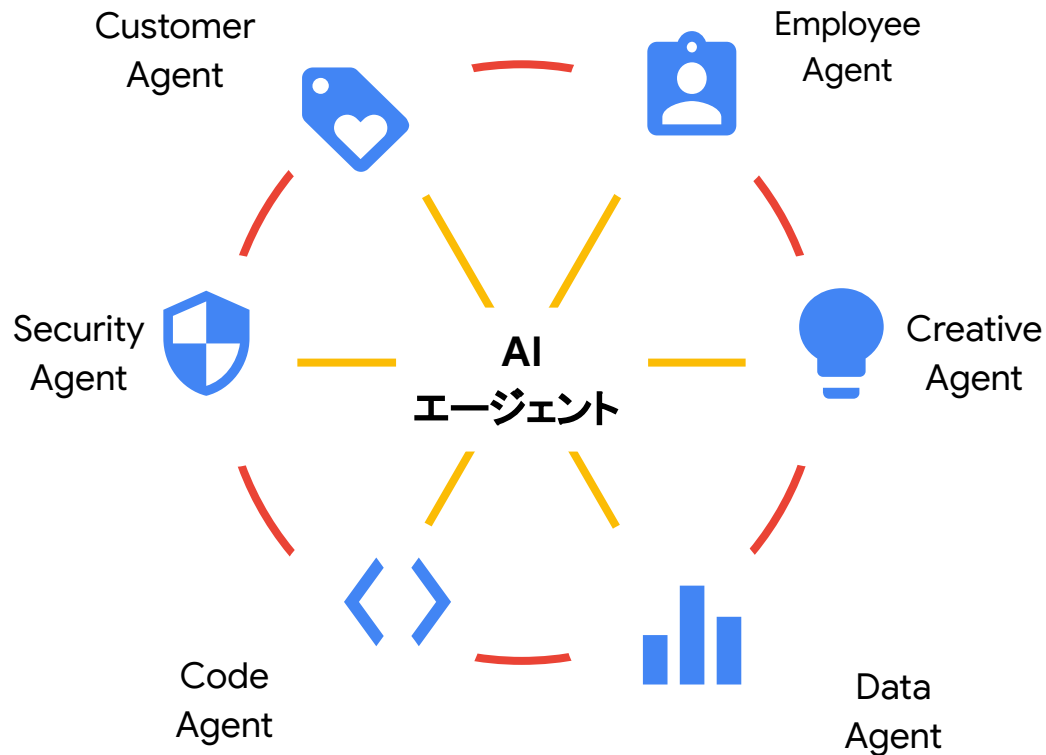


ツールやその他のリソースを活用する





# AI エージェントのユースケース



ユーザー データ 分析



目的に向かって行動する



情報収集、生成

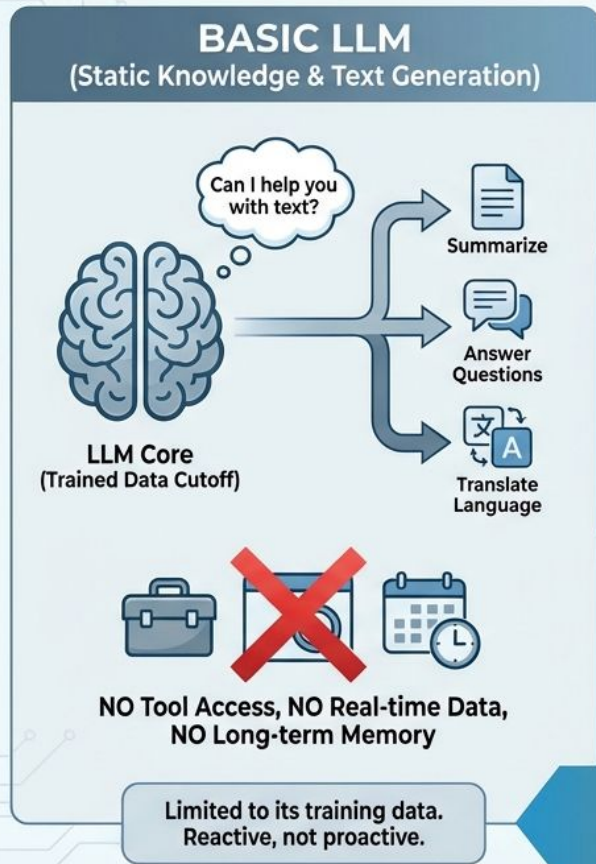


目的を達成するため計画、計画の実行

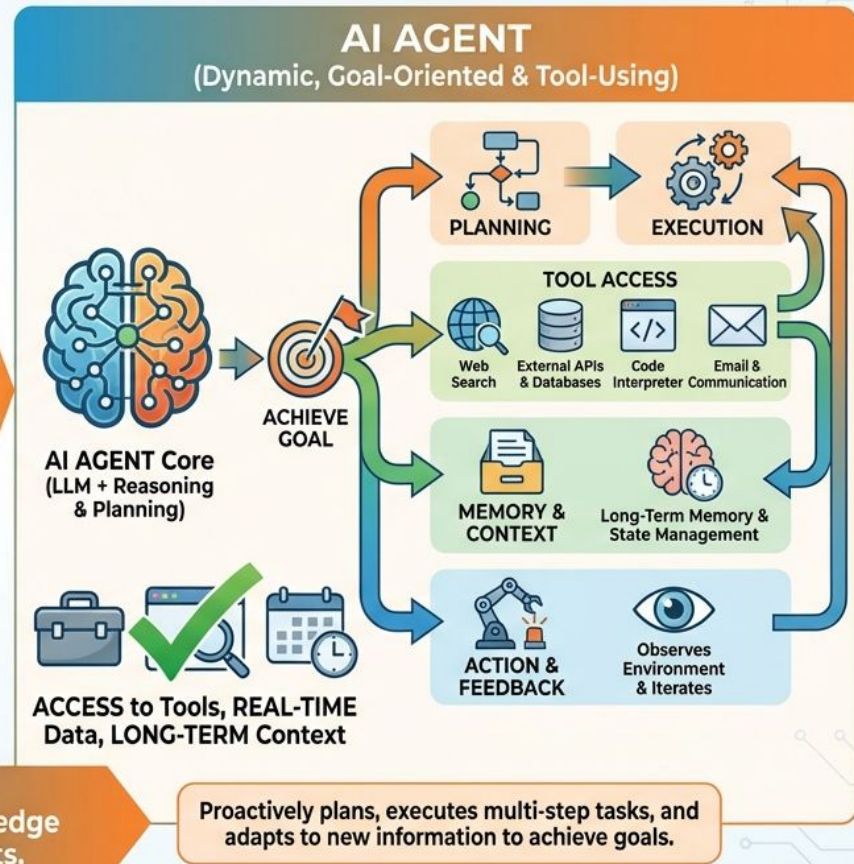


ツールやその他のリソースを活用する

# ADVANTAGES OF AI AGENTS VS. LLMs



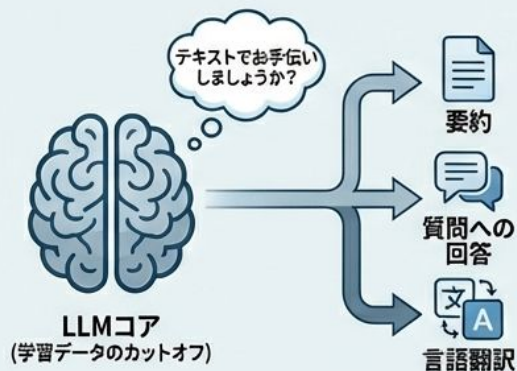
**EVOLUTION:  
FROM  
PASSIVE  
TEXT TO  
AUTONOMOUS  
ACTION**



**AI AGENTS:  
Converting Static Knowledge  
into Real-World Results.**

# AIエージェントとLLMの比較：AIエージェントの利点

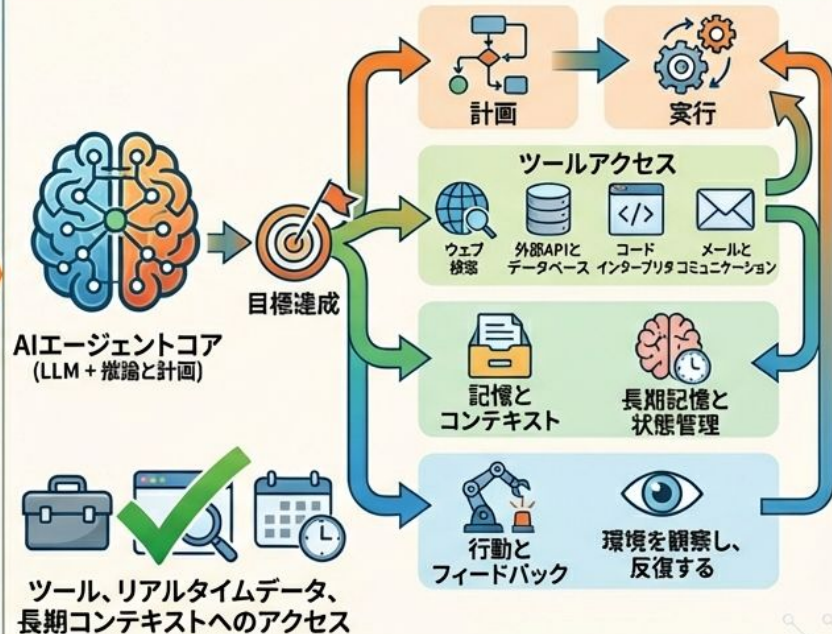
## 基本的なLLM (静的知識とテキスト生成)



学習データに限定される。  
受動的で、能動的ではない。

進化：  
受動的な  
テキストから  
自律的な行動へ

## AIエージェント (動的、目標指向、ツール使用)



AIエージェント：  
静的な知識を現実世界の成果に  
変える。

目標を達成するために、能動的に計画し、多段  
階のタスクを実行し、新しい情報に適應する。

# Gemini CLI

01

## コードとファイル

コードを生成し、  
ファイルを管理

02

## ツールの呼び出し

他のツールや MCP を  
呼び出し、ローカル開  
発の管理、テストの実  
行、クラウドサービス連  
携を実行

03

## 他のアプリとの連携

VS Code などと連携  
し、アクション実行やコ  
ンテキストを収集が可  
能

04

## 包括的なコンテキスト

プロジェクトファイル、  
データ、さらにはイメージ  
も含め、最も適切かつ効  
果的なサポートを提供



Google 検索  
ビルトイン



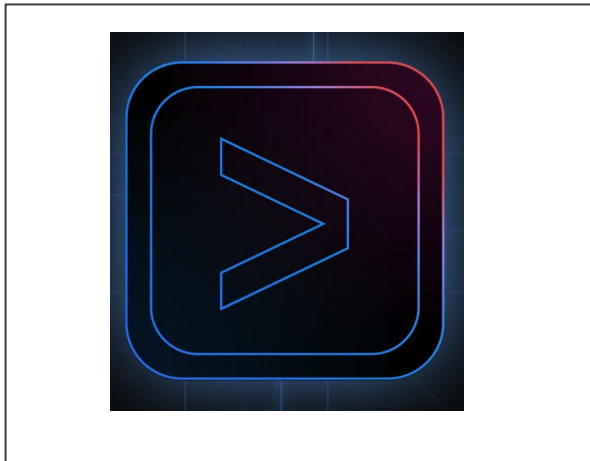
MCP で自由  
に拡張



Extensions を  
利用し Google  
Cloud サービス  
へのアクセス



# Gemini CLI Extensions (拡張ツール)



Available Now

ATLASSIAN

Canva

CONFLUENT

dynatrace

elastic

Figma

GitLab

Grafana Labs

harness

HashiCorp  
an IBM Company

MongoDB

neo4j

Pinecone

POSTMAN

qodo

Sonar

shopify

snyk

stripe

ThoughtSpot

Weights & Biases  
by CoreWeave

WIX

Coming Soon

Anomalo

arize

augment code

BROADCOM

chronosphere

DATADOG

ENDOR LABS

NetApp

netskope

new relic

Redis

Rill

transmit  
security

Gemini CLIの機能を拡張する  
Google Cloud 含め**100+**のGemini  
CLI Extensions 提供中

<https://geminicli.com/extensions/>

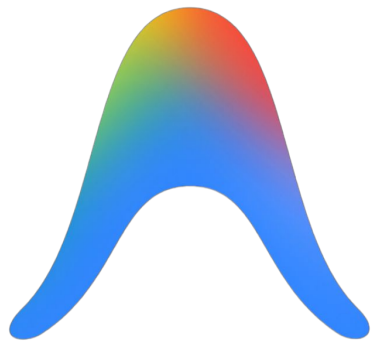
# Introducing Gemini 3

エージェント型推論とマルチモーダル性を基盤とした、当社史上最も強力なモデル

SOTA Reasoning & Multimodality

Agentic Coding

Advanced Tool Use & Planning



# Google Antigravity

1

All developers always use these **three tools**

IDE, Terminal, Browser (often)

2

Give developers a **unified agentic experience**

IDE + Terminal + Browser を統合された環境

3

**Agent first workflow**

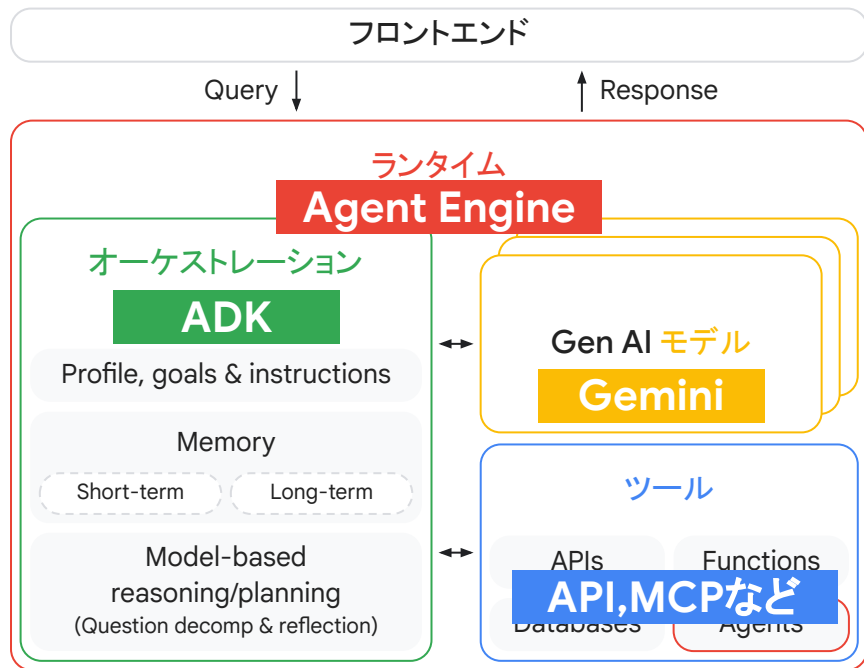
目的思考コーディング。複数のコード・エージェントのを運用

# Google Cloud で エージェント の構築





# AI エージェントの 4 つのキーコンポーネント



## AI Agent のキーコンポーネント



### モデル:

与えられた目的・指示に応じた推論、計画、およびレスポンス生成に用いる



### ツール:

他の API やサービス呼び出すことで、データを取得したり、アクションやトランザクションを実行する



### オーケストレーション:

エージェントのプロファイル(目的と指示)を保持しメモリ・セッション、ツール、提供取得されたデータなどを維持管理する



### ランタイム(実行環境):

ユーザーから呼び出されたときにエージェントを実行する環境

OSS

# Agent Development Kit (ADK)

Agent 開発に最適化された Python のフレームワーク

- オープンなエコシステム

*Works with ... anything*

- Gemini にも最適化

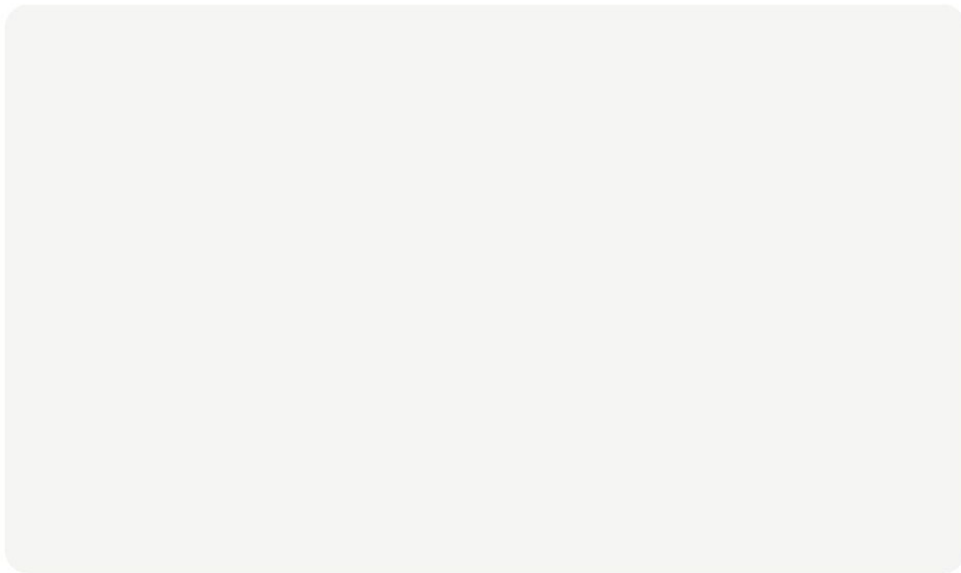
*Streaming, context, etc*

- 信頼性の高い OSS

*Google building with it*

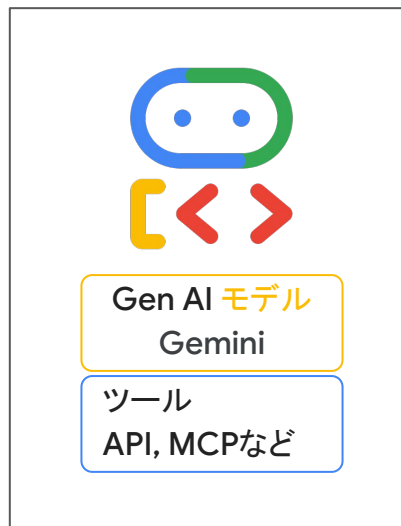
- 正確な制御 *Code + LLM mixed*

- 早い *Easy to learn*



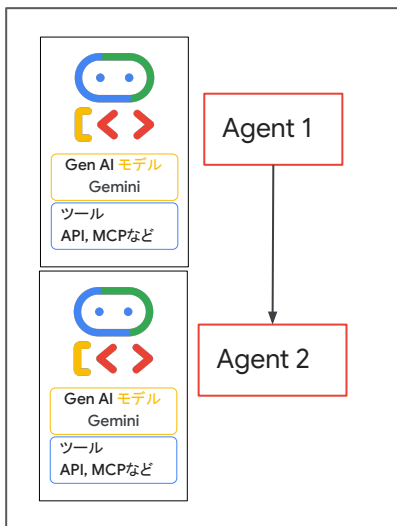
# ADK でエージェント ワークフロー を構築

LlmAgent



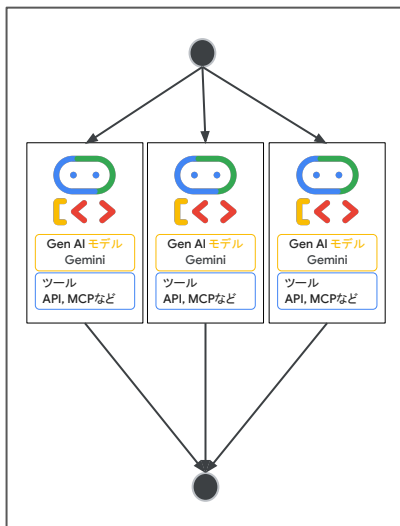
ADK の基本  
エージェント

Sequential Agent



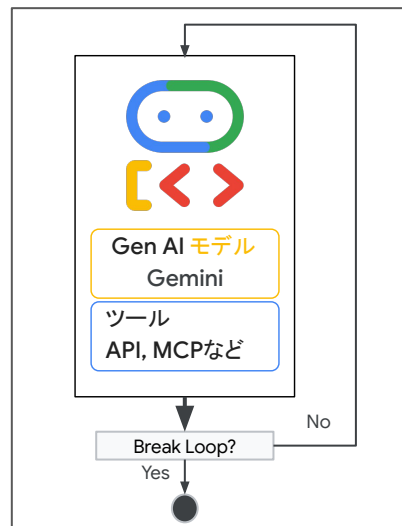
複数の エージェント を順  
番に駆動

Parallel Agent



複数の エージェント を並列  
に駆動

Loop Agent



エージェントを特定条件満  
たすまでループ  
させる

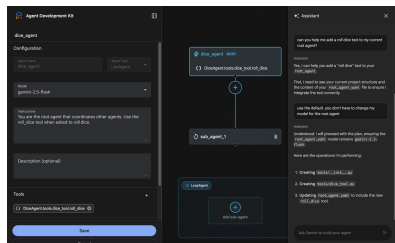
# ADK Visual Builder

The screenshot displays the ADK Visual Builder interface, which is divided into three main sections:

- Configuration Panel (Left):** This panel is for setting up the `dice_agent`. It includes fields for the Agent Name (`dice_agent`), Agent Type (`LlmAgent`), and Model (`gemini-2.5-flash`). The Instructions section contains the text: "You are the root agent that coordinates other agents. Use the `roll_dice` tool when asked to roll dice." There is also a Description (optional) field and a Tools section where the `DiceAgent.tools.dice_tool.roll_dice` tool is listed. At the bottom are "Save" and "Cancel" buttons.
- Visual Flow Diagram (Center):** This diagram illustrates the agent's workflow. It starts with a `dice_agent` (root) node, which connects to a `sub_agent_1` node. Below `sub_agent_1` is a dashed box labeled "LoopAgent" containing an "Add sub-agent" button, indicating a loop in the agent's execution.
- Assistant Chat (Right):** This panel shows a conversation with an AI Assistant. The user asks: "can you help me add a roll dice tool to my current root agent?". The Assistant responds: "Yes, I can help you add a 'roll dice' tool to your `root_agent`." It then provides instructions: "First, I need to see your current project structure and the content of your `root_agent.yaml` file to ensure I integrate the tool correctly." The user responds: "use the default, you don't have to change my model for the root agent". The Assistant then states: "Understood. I will proceed with the plan, ensuring the `root_agent.yaml` model remains `gemini-2.5-flash`." It lists the operations it is performing: 1. Creating `tools/__init__.py`, 2. Creating `tools/dice_tool.py`, and 3. Updating `root_agent.yaml` to include the new `roll_dice` tool. At the bottom is a button labeled "Ask Gemini to build your agent".

GUIとAIプロンプトで  
ADK エージェント開発

# ADK Visual Builder の仕組み (重要)



Visual Builder で  
エージェントの開発



Agent  
-tmp  
- agent.yaml

エージェントファイルが  
tmp フォルダに保管



Save

Save ボタンを  
押します



Agent  
- agent.yaml

Save ボタンを押すと  
エージェント関連ファイルが  
Agent メインフォルダに  
保管されます

# ハンズオン



# Thank you

