Agent Development Kit（ADK）と Agent Engine で学ぶ！AI エージェント開発。基礎から実践まで

Summary	In this codelab you'll learn how to develop AI Agents using Agent Development Kit and Agent Engine.
URL	learn-ai-agent-development-with-agent-development-kit-and-agent-engine-from-basics-to-practice-in-3hours
Keywords	docType:Codelab, product:GoogleDocs
Feedback Link	
Author	Kozzy Hasebe
Analytics Account	
Project	/devsite/_project.yaml
Book	/devsite/_book.yaml
Award Behavior	AWARD_BEHAVIOR_ENABLE
Layout	scrolling
Robots	noindex

概要

AI エージェントは、かつてないほどの注目と期待を集めています。このハンズオンでは、Google が提供する生成 AI モデル Gemini、そして開発を加速する ADK や Agent Engine といったツール群の基礎を学びます。AI エージェント開発にこれから挑戦したい方を対象に、基本コンセプトの理解から、実際に簡単な AI エージェントを構築し、その動作を体験します。

学習内容

- Agent Development Kit (ADK) を利用して AI エージェントを開発する基礎を学びます。
- ADK で作成した AI エージェントを Agent Engine にデプロイする方法を学びます。

作業内容

- ハンズオンの環境を整備する
- Google AI Studio で API キーを作成し、ソースコードに設定する
- シングルエージェントのソースコードを確認する
- シングルエージェントを Web モードで起動する
- シングルエージェントを Web から試してみる
- シングルエージェント＋ツールのソースコードを確認する
- シングルエージェント＋ツールの動作を確認する
- [チャレンジ問題] シングルエージェントの機能を拡張する
- ツールを MCP サーバーで実装されている機能に置き換える
- サブエージェントを追加する (マルチエージェント構成)
- AI エージェントを Agent Engine にデプロイする
- サブエージェントに独自ツールを追加する (マルチエージェント構成)

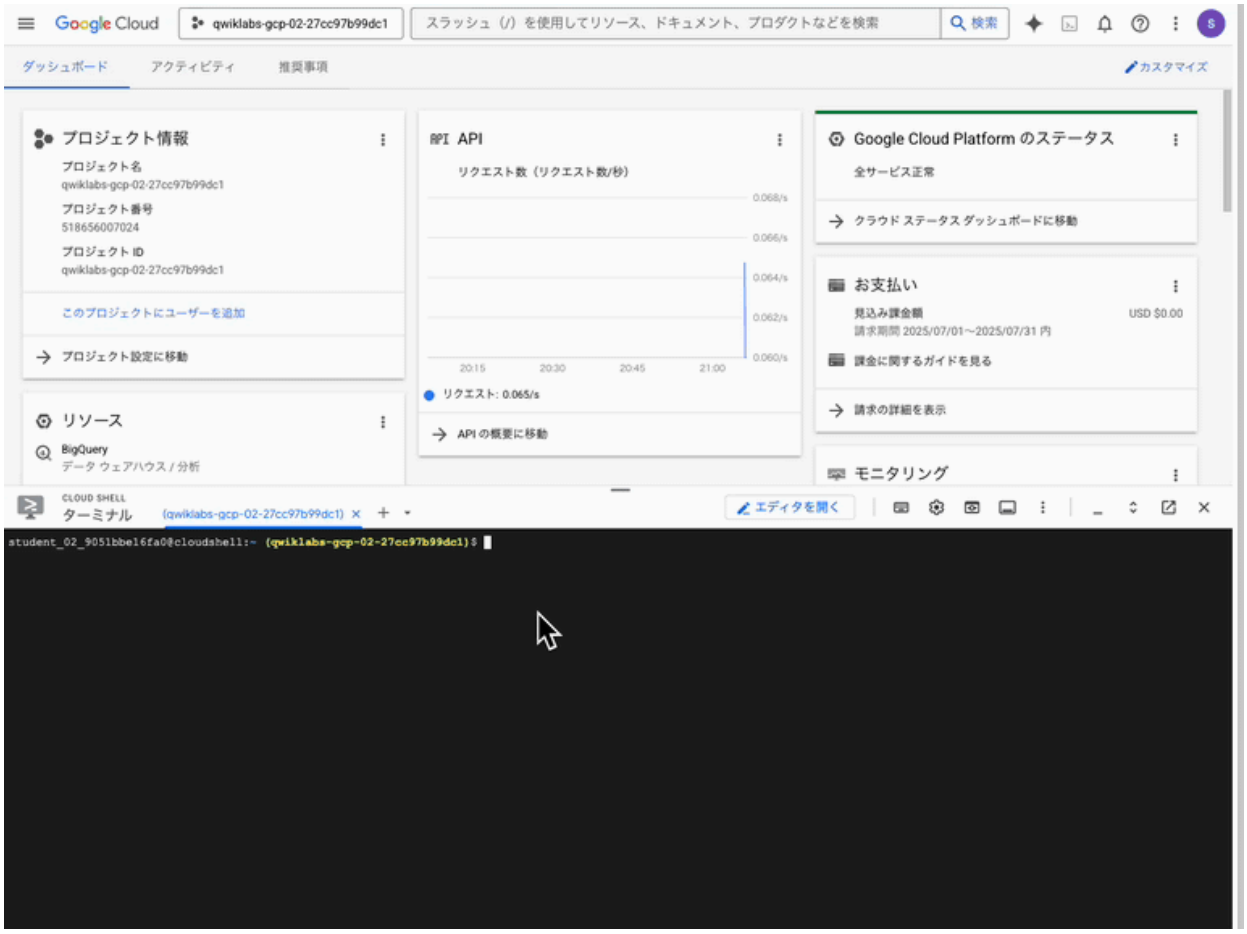
- Agent Engine にデプロイした AI エージェントを使ってみる
- [チャレンジ問題] ポッドキャストの出力をカスタマイズする
- エージェントはどのように評価 (テスト) するか？
- シンプルなテスト構成を考える
- テストを実行する
- テストケースを生成する

ハンズオンの環境を整備する

Duration: 4:00

ハンズオン用の Google Cloud プロジェクトで必要な機能 (API) の有効化、資材のダウンロード、また Cloud Shell 上にツールをインストールします。

作業動画



作業手順

1. Cloud Shell のターミナル (黒い画面) から以下のコマンドを実行し、ハンズオン環境を整備します

```
curl -s https://gist.githubusercontent.com/hasebe/62934be9c66a65376247510d7dd821c6/raw/adk-handson-setup.sh | bash
```

2. 右下にでる Git に関する通知は Never をクリック、また画面右側にでる Gemini Code Assist のペインは × をクリックし閉じておきます

ここで表示されている画面 (Chrome のタブ) を以降の手順で **[エディタ]** と呼びます。
作業手順の行頭に **[エディタ]** と書かれている場合は、こちらの画面で作業を実施します。

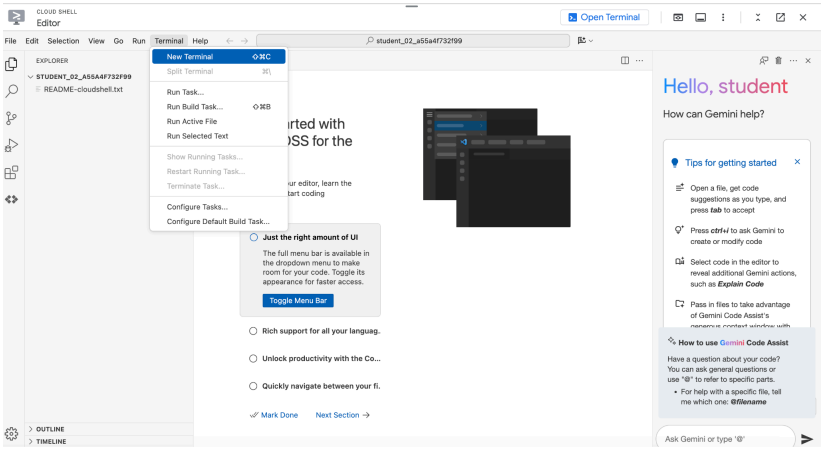
Cloud Shell Editor 操作方法一覧 (以降の手順で必要なときにご参照ください)

Duration: 2:00

ターミナルを開く

画面上部の **Terminal** -> **New Terminal** の順でクリックすると、エディタの下半分にターミナルが表示されます

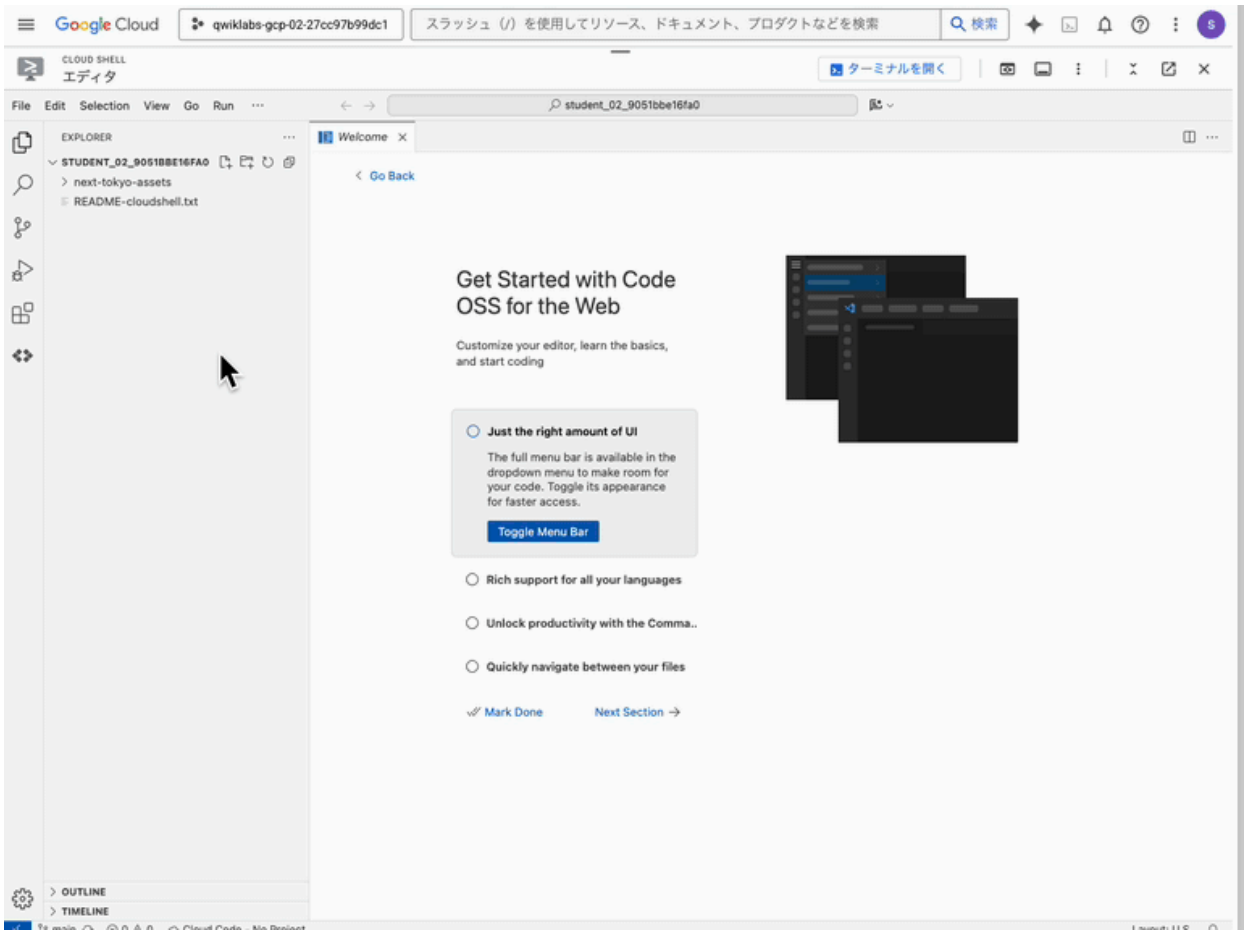
💡 ヒント: 画面の広さによっては Terminal が隠れてしまっていることがあります。その場合は ··· をクリックしてみてください。



作業フォルダを移動する

接続が切れてしまったなどで、フォルダが /home/[username]/ に戻ってしまった場合は、以下の手順で作業フォルダを移動することができます

1. [エディタ] メインメニュー -> File -> Open Folder... の順番に選択します
2. [エディタ] 移動したいフォルダを選択し、OK をクリックします
(動画では [home_directory]/next-tokyo-assets/2025/adk-agentengine-basic/ フォルダに移動しています)



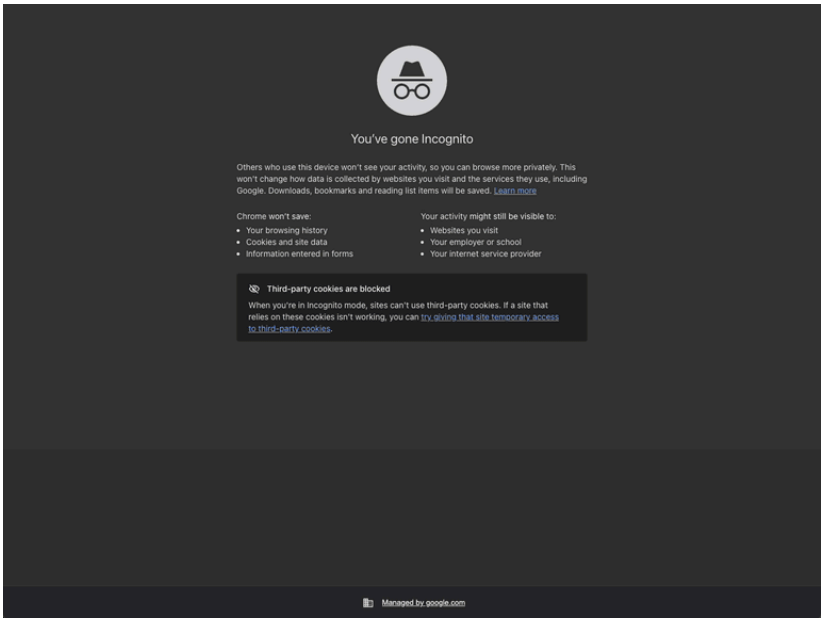
Google AI Studio で API キーを作成し、ソースコードに設定する

Duration: 5:00

Google の生成 AI モデルである **Gemini** を利用するために **API** キーを作成し、ソースコードから利用できるようにします。

API キーは Google AI Studio で作成し、作成した API キーを .env ファイルに設定します。

作業動画



作業手順

1. エディタが開いているブラウザで新しいタブを開き、以下の URL にアクセスします

```
https://aistudio.google.com/
```

2. 2 つあるチェックボックスの上のみにチェックを入れ、`Continue` をクリックします
3. 右下にある `Get API Key` をクリックします
4. 画面真ん中にある `Import projects` をクリックします
5. 新しく表示された右のウィンドウから、今回のハンズオン用のプロジェクト ID (qwiklabs-gcp- から始まるもの) をチェックし、`Import` をクリックします
6. 右上にある `Create API Key` をクリックします
7. `Select a Cloud projects` をクリックし、今回のハンズオン用のプロジェクト ID (qwiklabs-gcp- から始まるもの) を選択します
8. `Create key` をクリックし、API キーを作成します
9. 作成されたキーのコピーボタンをクリックし、API キーをクリップボードにコピーします
10. **[エディタ]** エクスプローラから `.env` ファイルを開き、`GOOGLE_API_KEY=` の後に貼り付けます

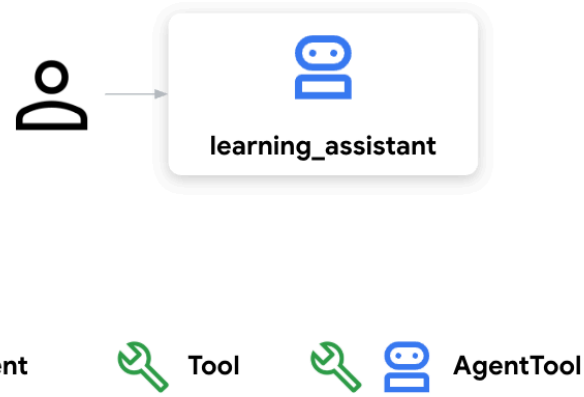
シングルエージェントのソースコードを確認する

Duration: 5:00

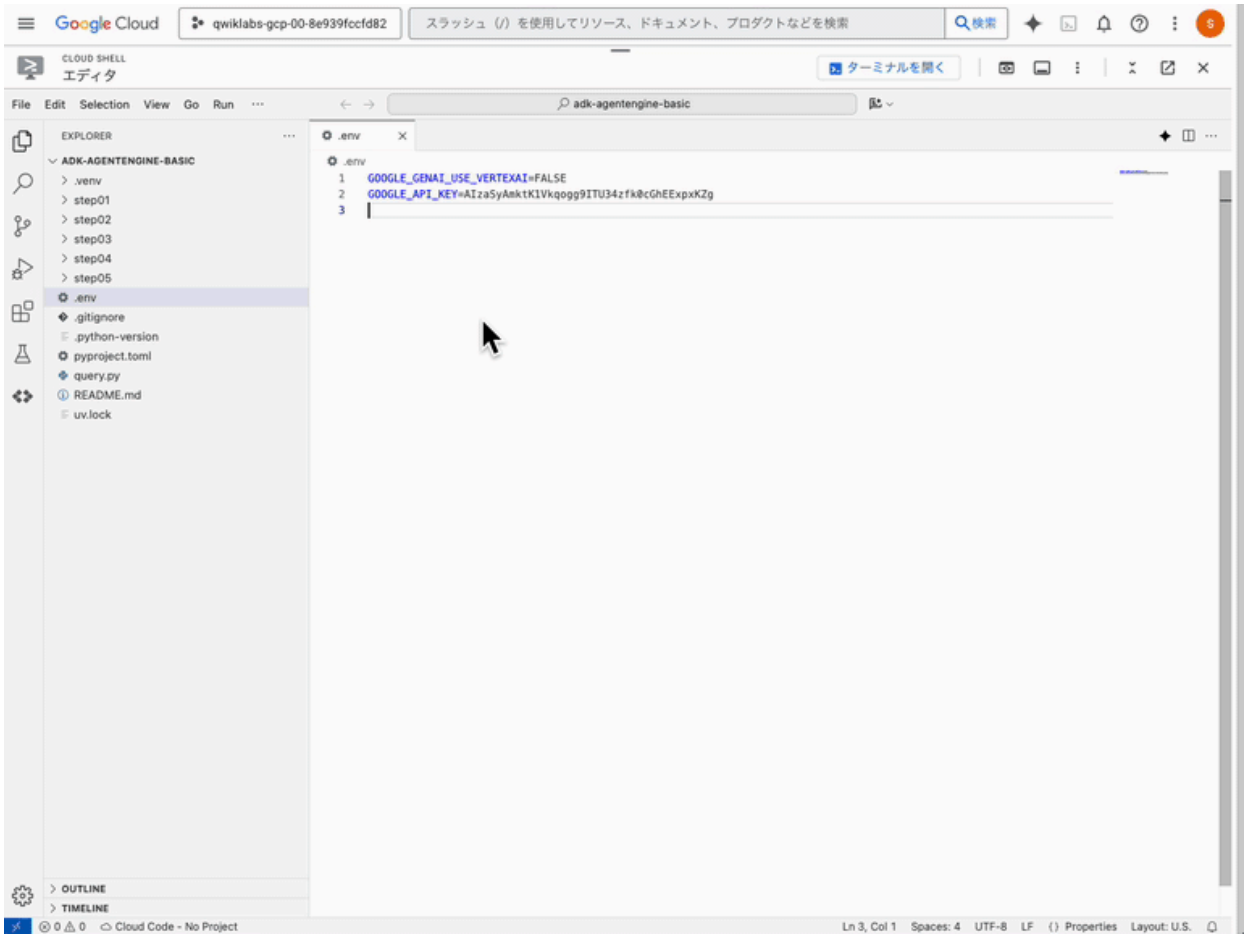
ADK は、簡単に生成 AI を活用した AI エージェントを構築することができるフレームワークです。

まず単体で動く簡単な AI エージェントはどのようなソースコードで実現されているかを見てみましょう。

エージェント構成



作業動画



作業手順

1. **[エディタ]** エクスプローラ (左側のファイル一覧ペイン) から、`step01`、`agent.py` の順にクリックします
2. **[エディタ]** `agent.py` の中身がエディタ (一番広いエリア) に表示されます
3. **[エディタ]** `agent.py` のソースコード、またパラメータ群 (name, model, description, instructions) を確認します

ソースコードのポイント

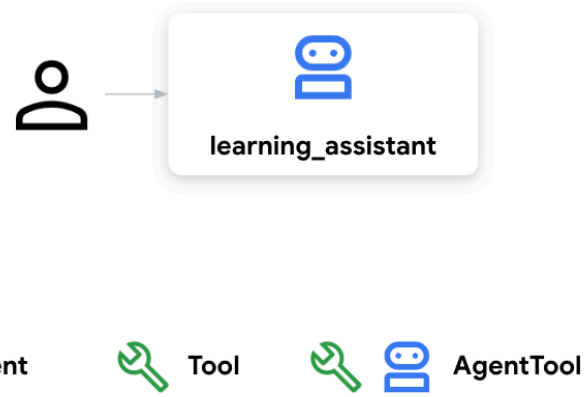
- **LlmAgent:** この AI エージェントは内部で生成 AI モデルへの問い合わせを行う
- **Name:** AI エージェント名
- **Model:** どの生成 AI モデルを利用するか (デフォルトでは Gemini 2.5 Flash を指定しています)
- **Description:** AI エージェントの説明
- **Instructions:** AI エージェントへの指示事項
- **root_agent:** 最初にユーザーからの問い合わせを受け付ける AI エージェント

シングルエージェントを Web モードで起動する

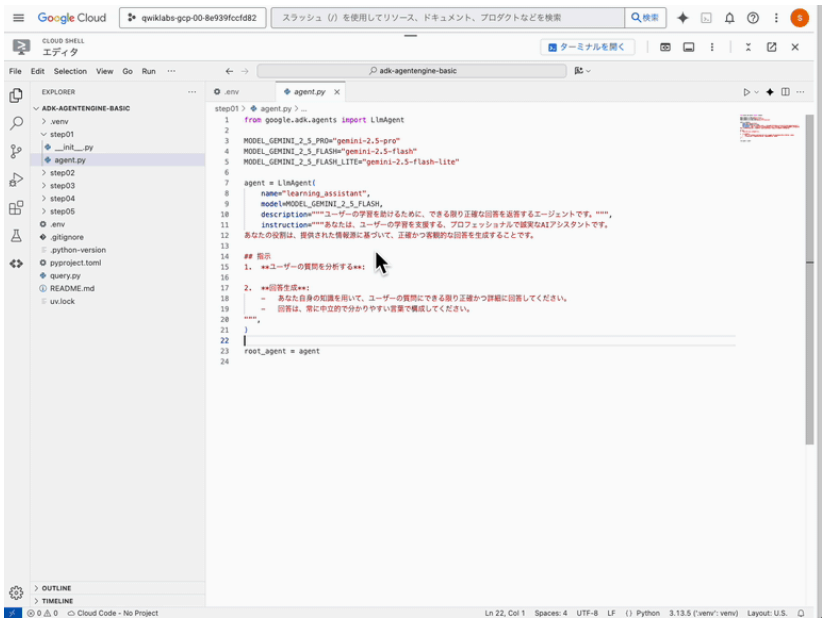
Duration: 5:00

ADK で開発されている、ハンズオン用のシンプルなシングルエージェントを起動します。

エージェント構成



作業動画



作業手順

1. **[エディタ]** ターミナルを開きます (ターミナルの開き方は **Cloud Shell Editor 操作方法一覧**をご参照ください)

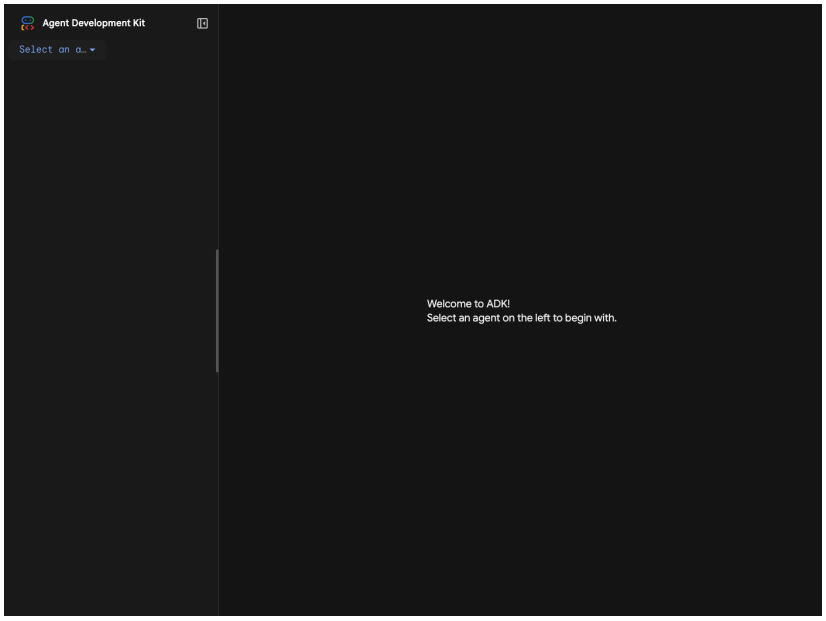
エディタ画面にターミナルを開くというボタンがありますが、そちらではありません。ご注意ください。

2. **[エディタ]** ターミナルから以下のコマンドを実行し、adk を Web モードで起動します

```
nodemon -e py --delay 5 --exec uv run adk web --port 8080
```

💡 ヒント: 上記をコピー & ペーストした初回はその手順を許可するかどうかが聞かれることがあります。その場合は `Allow` をクリックして、許可してください。
ADK の Web UI にはホットリロード機能がついていません。そのため、nodemon というツールを利用し、Python のソースコードに修正があった場合、自動的にリロードするようにしています。
3. **[エディタ]** 上のコマンドが完了した際の出力の中から、`http://127.0.0.1:8080` という表示を Mac の場合は Command キー、Windows の場合は Ctrl キーを押しながらクリックします

4. ブラウザで新しいタブが立ち上がり、以下のような ADK の Web 画面 **(Web UI)** が立ち上がったことを確認します



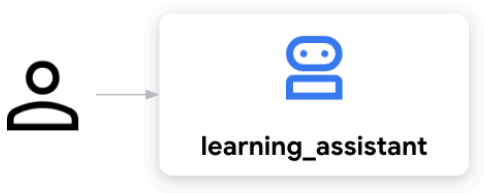
ここで表示されている画面 (Chrome のタブ) を以降の手順で **[Web UI]** と呼びます。
作業手順の行頭に **[Web UI]** と書かれている場合は、こちらの画面で作業を実施します。

シングルエージェントを Web から試してみる

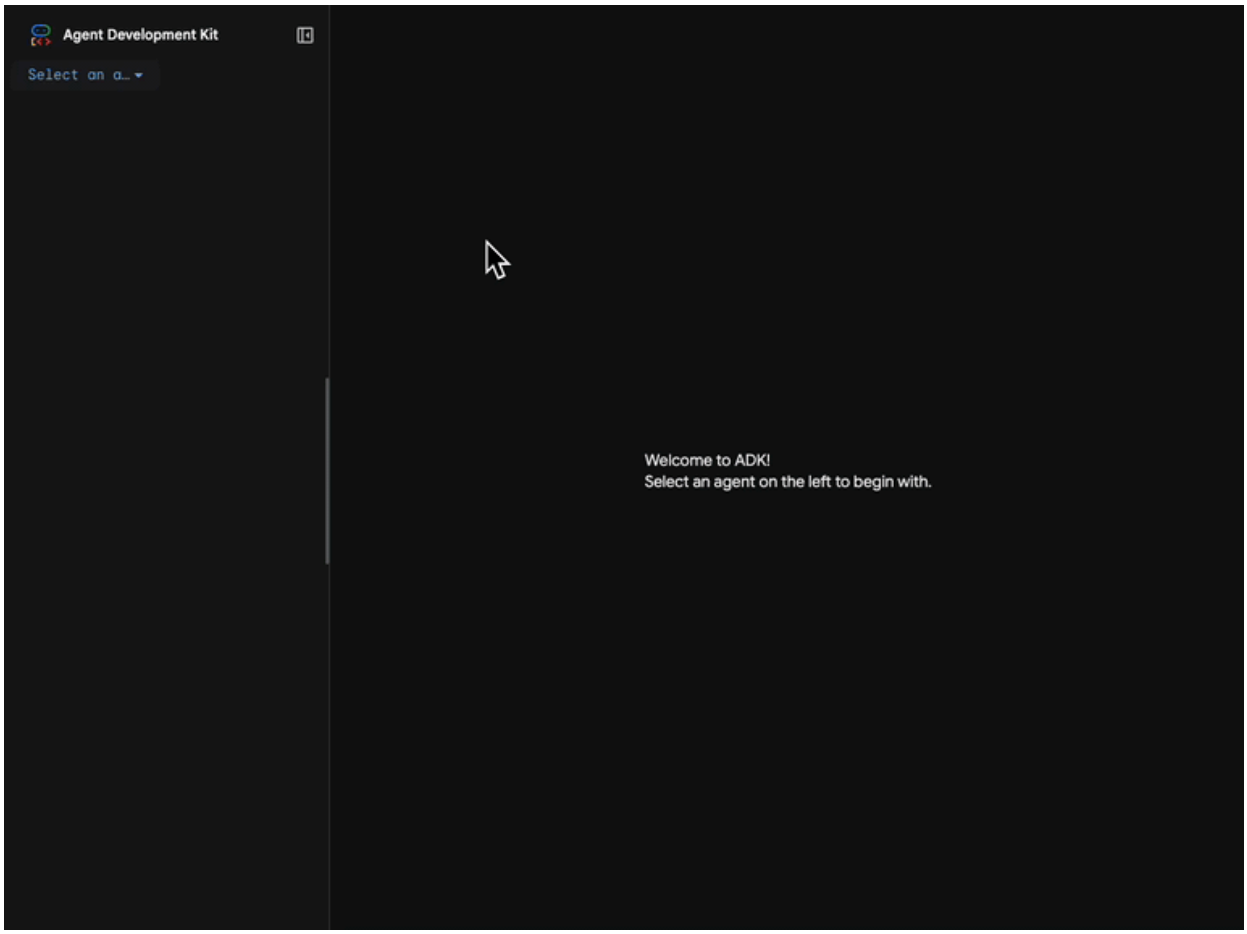
Duration: 6:00

ADK の Web モードからエージェントを触ってみます。

エージェント構成



作業動画



作業手順

1. **[Web UI]** 左上にあるエージェント選択欄から `step01` を選択します
2. **[Web UI]** チャットウィンドウから問い合わせを行います (シングルターン)
3. **[Web UI]** チャットウィンドウからフォローアップの問い合わせを行います (マルチターン)
4. **[Web UI]** 左側に表示されているパネルの Trace タブから問い合わせたメッセージをクリックし、詳細を確認します
5. **[Web UI]** 更にその中の `call_llm` をクリックし、生成 AI モデルとのやり取り詳細を確認します
6. **[Web UI]** 右上の `New Session` をクリックすると、新しいセッションが作成されます

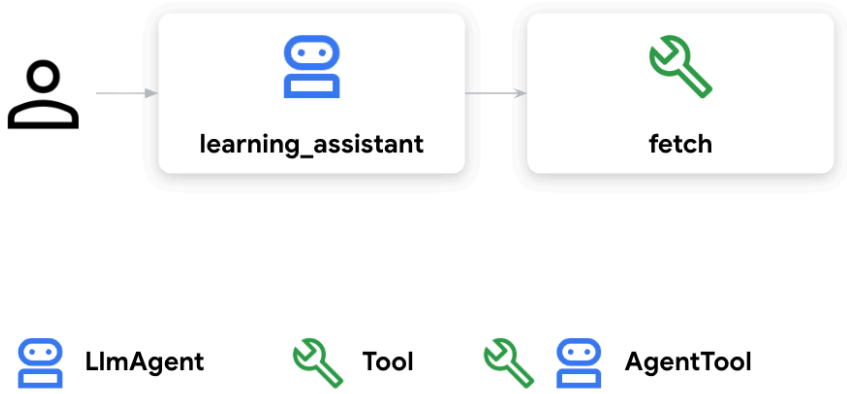
シングルエージェント＋ツールのソースコードを確認する

Duration: 5:00

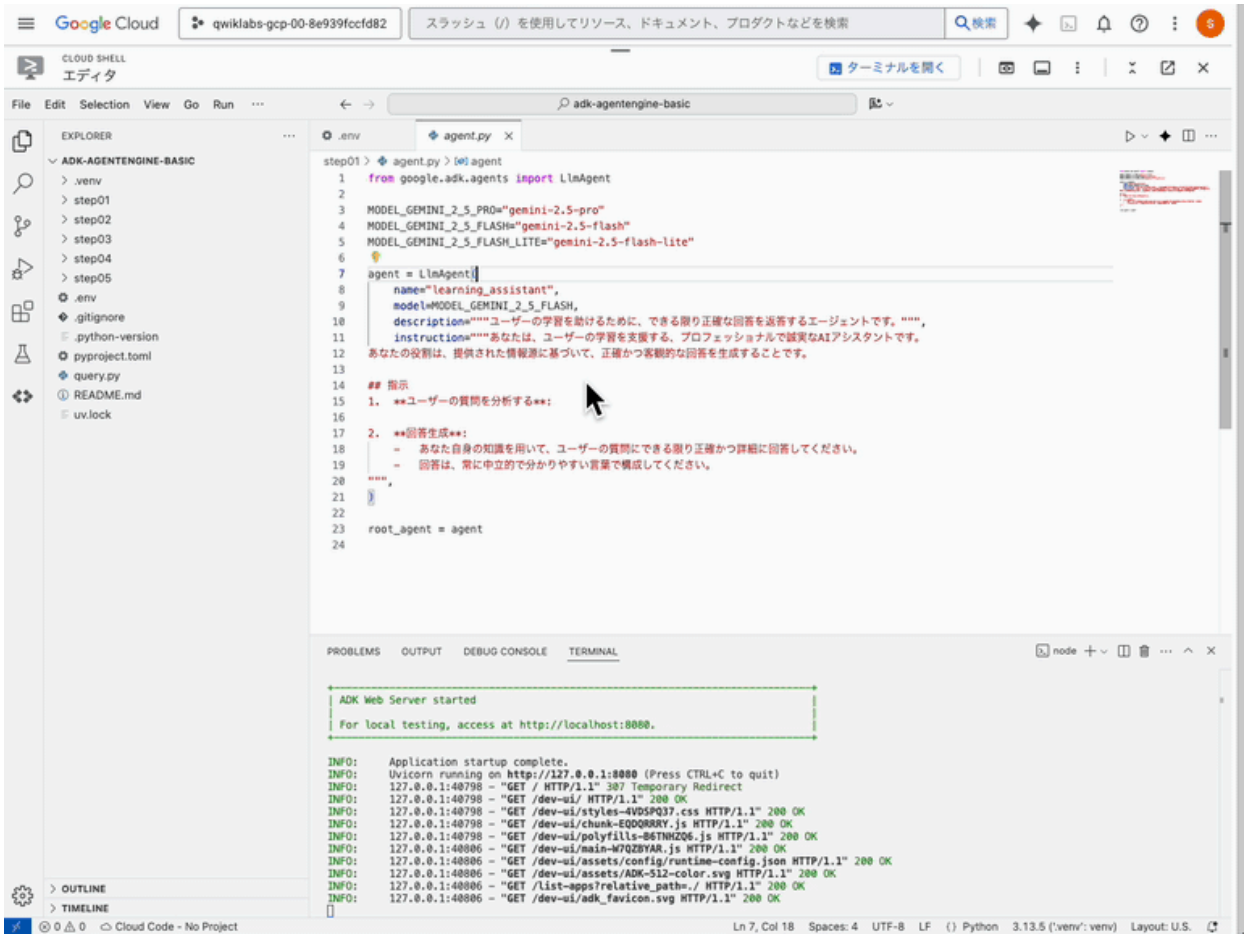
ここまでで作成した AI エージェントは生成 AI の知識のみで回答するものになっていました。ここでツールを追加して、**URL** が与えられた場合には、生成 **AI** の知識に加えてそちらの情報を合わせて回答するように更新します。

ここで追加するツールは Web ページのコンテンツを取得し、マークダウンで出力する関数である `fetch` を利用します。

エージェント構成



作業動画



作業手順

1. **[エディタ]** エクスプローラ (左側のファイル一覧ペイン) から、`step02`、`agent.py` の順にクリックします
2. **[エディタ]** `agent.py` の中身がエディタ (一番広いエリア) に表示されます
3. **[エディタ]** `agent.py` のソースコード、またパラメータ群 (tools) を確認します

ソースコードのポイント

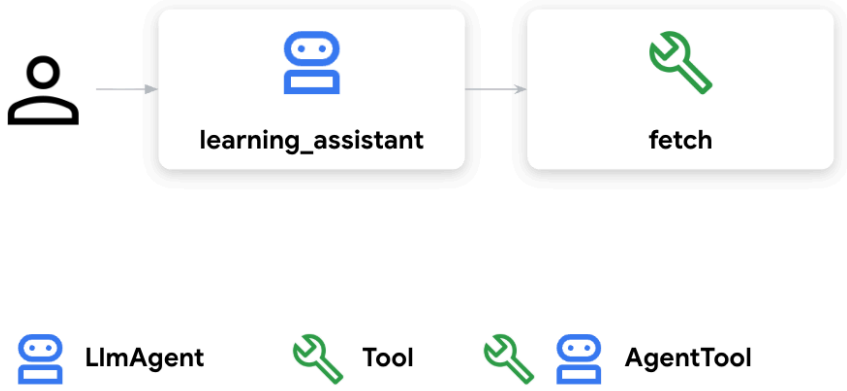
- **Tools:** どのツールを利用するか
- **fetch:** 新規に作成、追加した URL 先のコンテンツを取得し、マークダウン形式で出力する関数

シングルエージェント＋ツールの動作を確認する

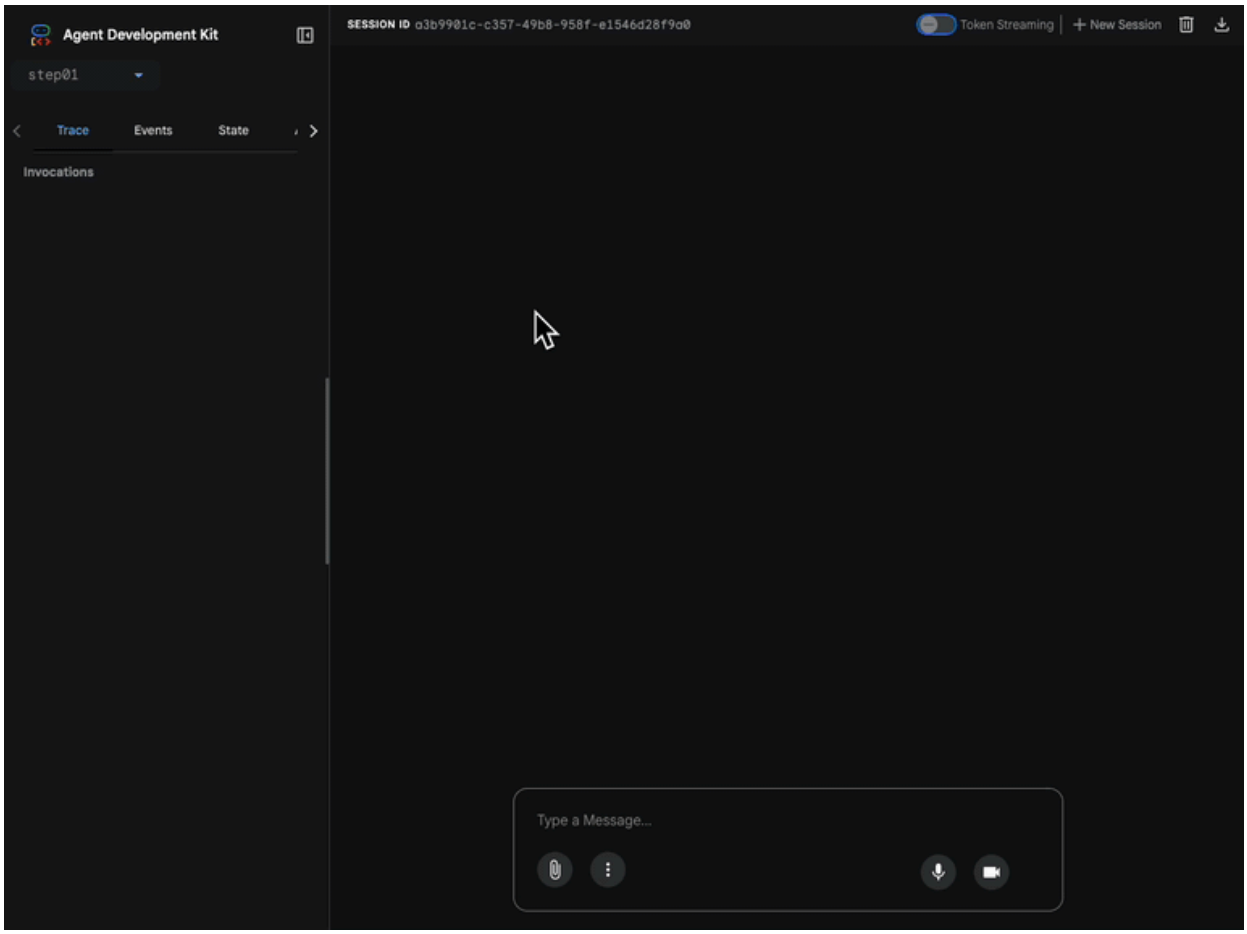
Duration: 6:00

先ほどソースコードを確認したツールを動作させてみましょう。

エージェント構成



作業動画



作業手順

1. **[Web UI]** 左上にあるエージェント選択欄から `step02` を選択します
2. **[Web UI]** チャットウィンドウから AI エージェントのできることを問い合わせます (例、あなたは何ができますか？)
3. **[Web UI]** チャットウィンドウから学習したいコンテンツを URL を含めずに問い合わせを行います
4. **[Web UI]** チャットウィンドウから学習したいコンテンツの URL を含めて問い合わせを行います
   💡 ヒント: コンテンツ (URL) が思いつかない方は、以下のページから取得すると便利です
   - Zenn トップページ
   - Qiita 週間トレンド記事一覧
5. **[Web UI]** チャットウィンドウでツール (`fetch`) が呼び出されて回答されたことを確認します

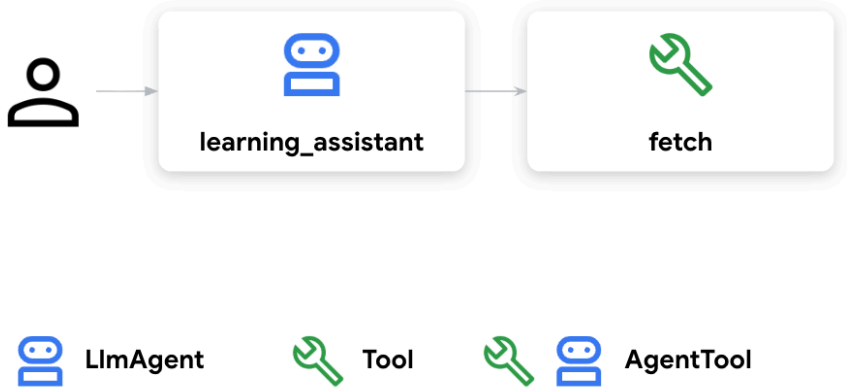
[チャレンジ問題] シングルエージェントの機能を拡張する

Duration: 7:00

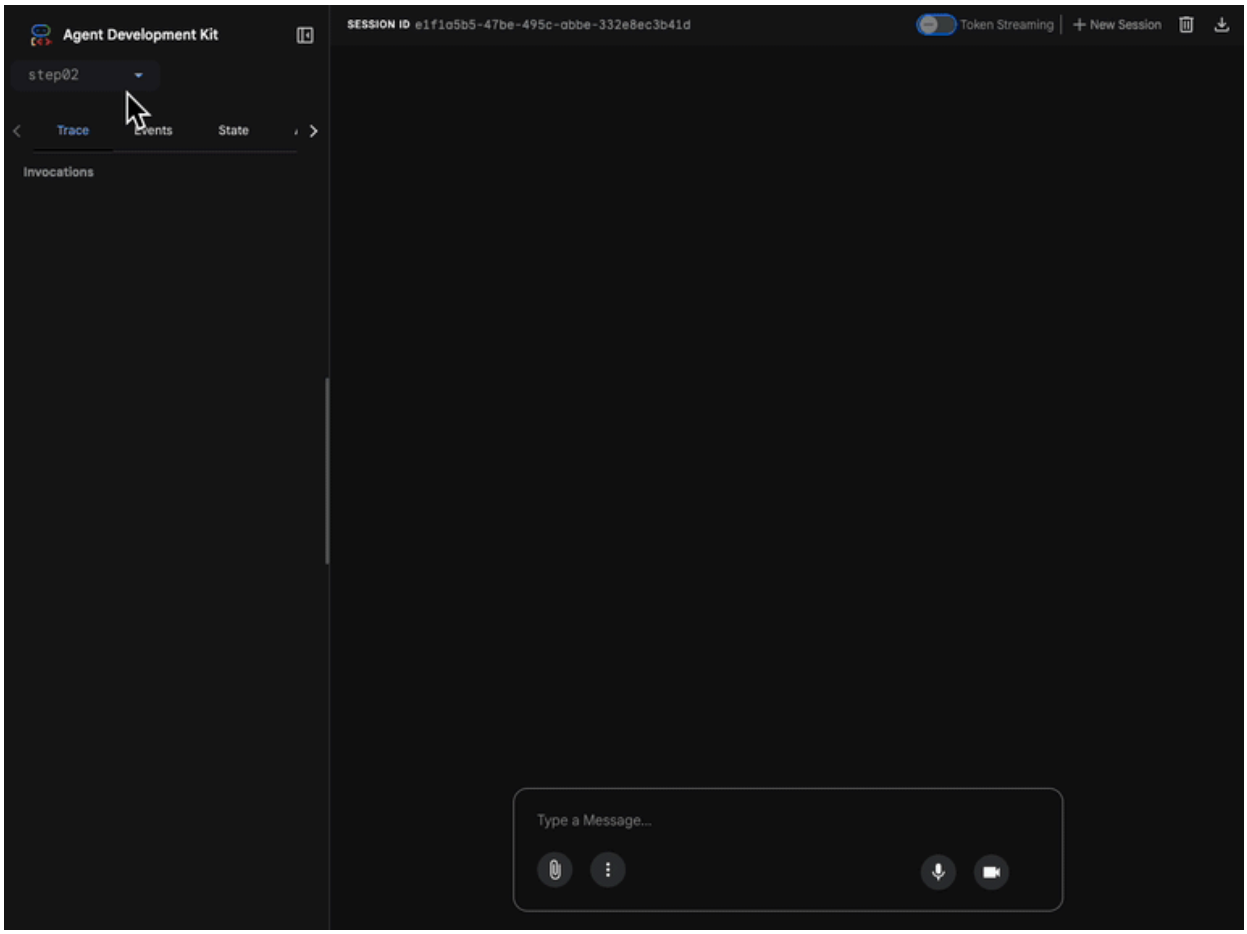
今の AI エージェントを、複数の **URL** を受け取ってその内容を元に回答するように機能を拡張します。

まず、今の状態は **1** つの **URL** しか受け取れないことを確認し、ソースコードを更新し複数の **URL** を受け取れるように更新しましょう。

エージェント構成



作業動画



作業手順

1. **[Web UI]** 左上にあるエージェント選択欄から `step02` が選択されていることを確認します
2. **[Web UI]** チャットウィンドウから AI エージェントに複数の URL を受け取れるか聞いてみます (例、あなたは複数の URL を受け取って回答できますか？)
3. **[エディタ]** エクスプローラ (左側のファイル一覧ペイン) から、`step02`、`agent.py` の順にクリックします
4. **[エディタ]** ソースコードを編集します
💡 ヒント: エージェントのソースコードの `instruction` を変更することで実現できます
5. **[Web UI]** 一度、ブラウザから Web UI をリロードします (Mac の場合は Command + R, Windows の場合は Ctrl + R)
6. **[Web UI]** 手順 2 の内容を再度聞いてみます
7. 複数の URL を受け取ることができるようになるまで、手順 1 ～ 6 を繰り返します

`step02` フォルダ内の `answer.py` に回答例が置かれています。

ツールを MCP サーバーで実装されている機能に置き換える

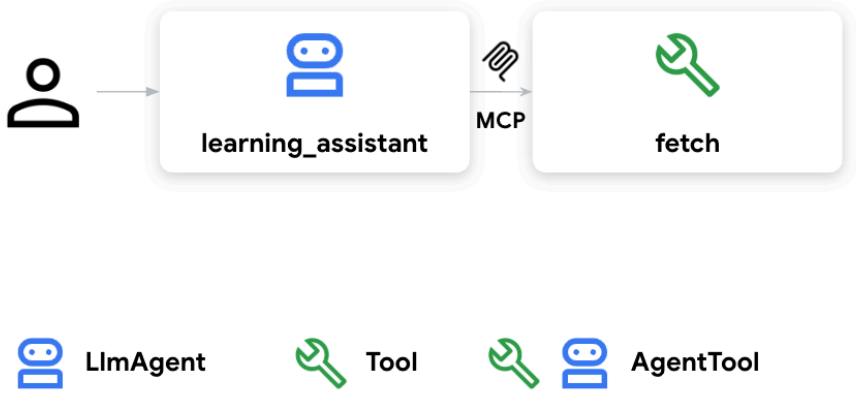
Duration: 6:00

URL からコンテンツを取得するツールを、**MCP** サーバーのリファレンス実装に含まれている機能に置き換えてみましょう。

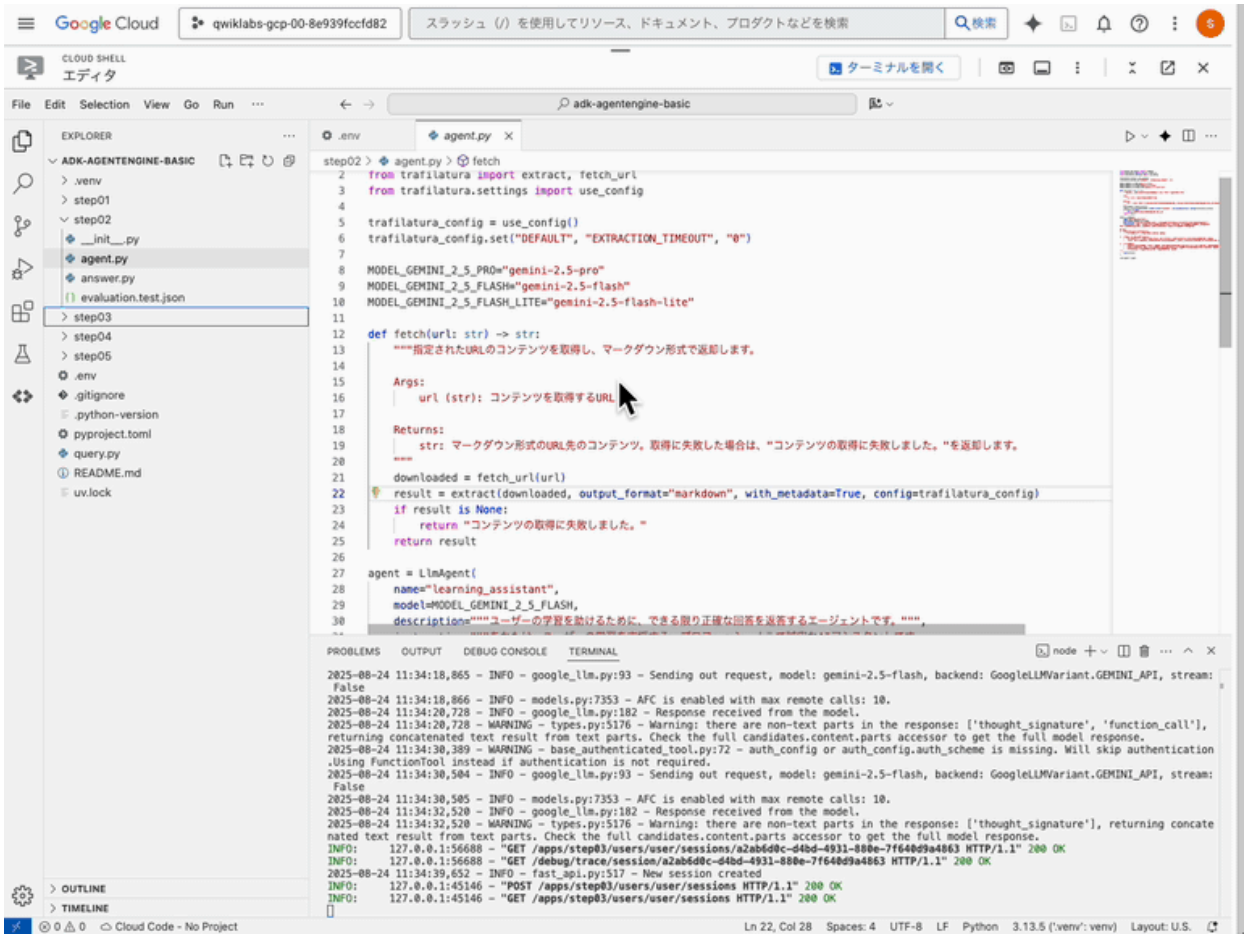
リファレンス実装は以下の URL で提供されています。

- https://github.com/modelcontextprotocol/servers/tree/main/src/fetch

エージェント構成



作業動画



作業、動作確認手順

1. [エディタ] エクスプローラ (左側のファイル一覧ペイン) から、`step03`、`agent.py` の順にクリックします
2. [エディタ] `step02` の `agent.py` との違いを確認します
3. [Web UI] 左上にあるエージェント選択欄から `step03` を選択します
4. [Web UI] チャットウィンドウから AI エージェントのできることを問い合わせます (例、あなたは何ができますか？)
5. [Web UI] チャットウィンドウから学習したいコンテンツの URL を含めて問い合わせを行います
💡 ヒント: コンテンツ (URL) が思いつかない方は、以下のページから取得すると便利です

- Zenn トップページ
- Qiita 週間トレンド記事一覧

6. [Web UI] チャットウィンドウでツール (`fetch`) が呼び出されて回答されたことを確認します

ソースコードのポイント

- **Tools:** `MCPToolSet` が使われている。これは ADK で提供されている MCP サーバーと連携するためのクラス。
- **command='uvx', args=['mcp-server-fetch']:** ローカルで `mcp-server-fetch` という MCP サーバーを立ち上げ利用している。

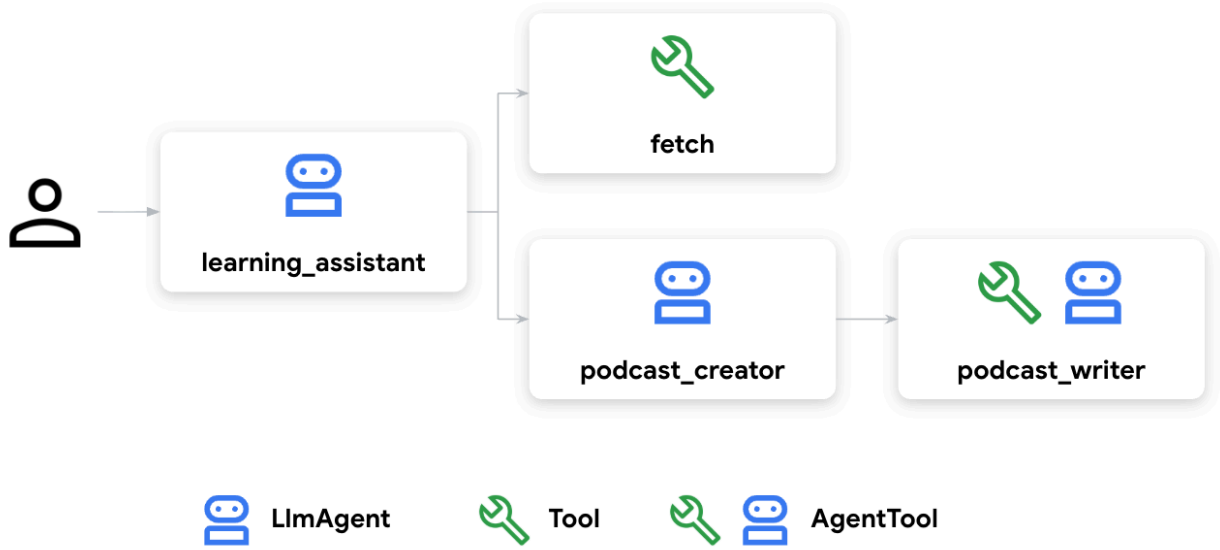
サブエージェントを追加する (マルチエージェント構成)

Duration: 7:00

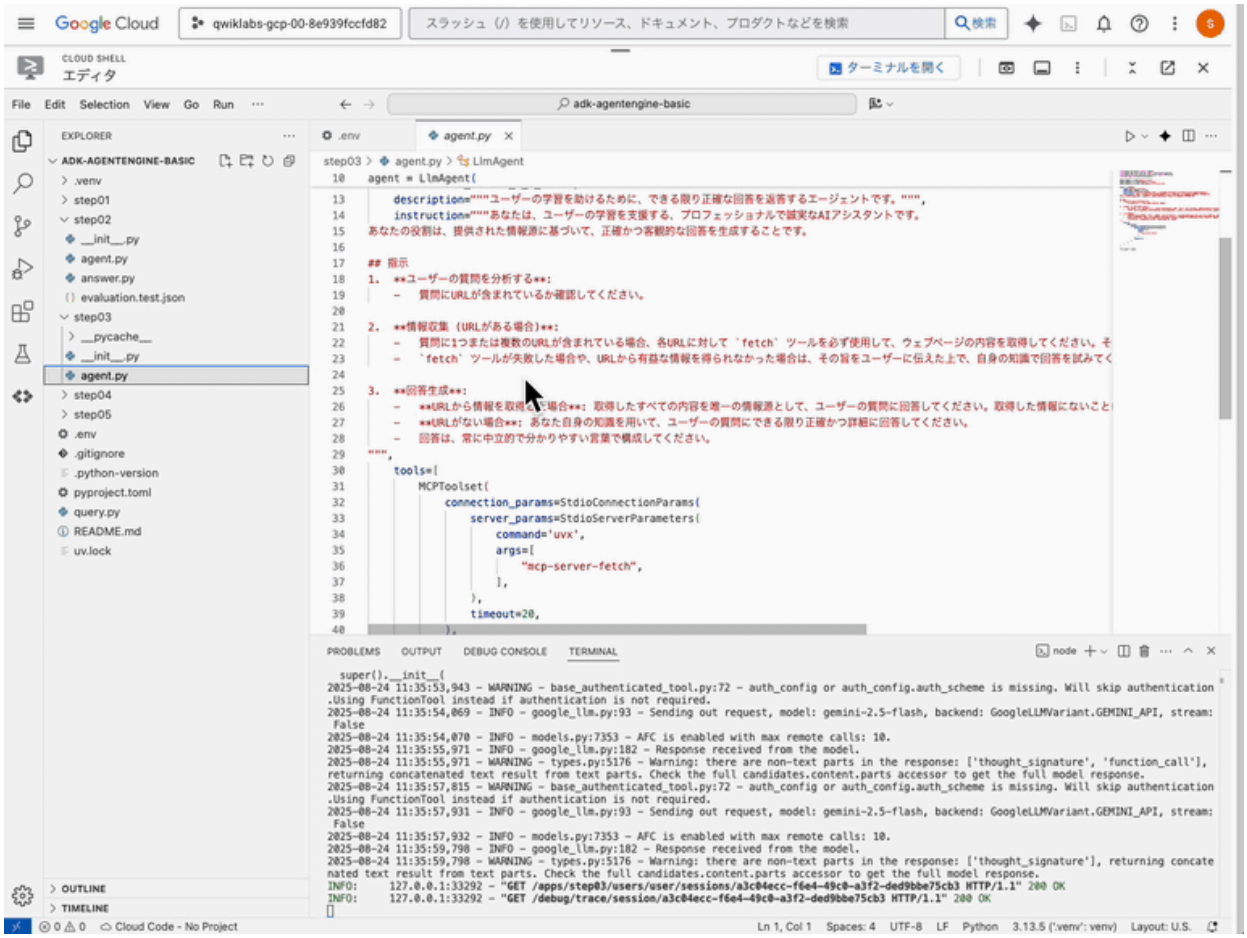
ここからは受け取った **URL** の内容からポッドキャストの台本 (スクリプト) を作成する **AI** エージェントを実装します。

そこでポッドキャストを作成する機能は別の AI エージェント (`podcast_creator`) に機能を切り分けることとします。

エージェント構成



作業動画



作業、動作確認手順

1. **[エディタ]** エクスプローラ (左側のファイル一覧ペイン) から、`step04`、`agent.py` の順にクリックします
2. **[エディタ]** `step03` の `agent.py` との違いを確認します
3. **[Web UI]** 左上にあるエージェント選択欄から `step04` を選択します
4. **[Web UI]** チャットウィンドウから AI エージェントのできることを問い合わせます (例、あなたは何ができますか？)
5. **[Web UI]** チャットウィンドウからポッドキャストにしたいコンテンツの URL を含めて問い合わせを行います
💡 ヒント: コンテンツ (URL) が思いつかない方は、以下のページから取得すると便利です
- Zenn トップページ
- Qiita 週間トレンド記事一覧

6. **[Web UI]** チャットウィンドウでツール (`fetch`) が呼び出された後、`podcast_creator_agent` への転送 (`transfer_to_agent`)、最後にツール (`podcast_writer_agent`) が呼び出され、ポッドキャストの台本が出力されたことを確認します

ソースコードのポイント

- **agent の sub_agents 設定:** ポッドキャストの生成は `podcast_creator_agent` に処理を委譲するように指定している。
- **podcast_creator_agent:** ポッドキャストの生成を取りまとめるエージェント。
- **podcast_writer_agent:** ポッドキャストの台本生成に責任を持つエージェント。`podcast_creator_agent` の 1 ツールとして設定している。

AI エージェントを Agent Engine にデプロイする

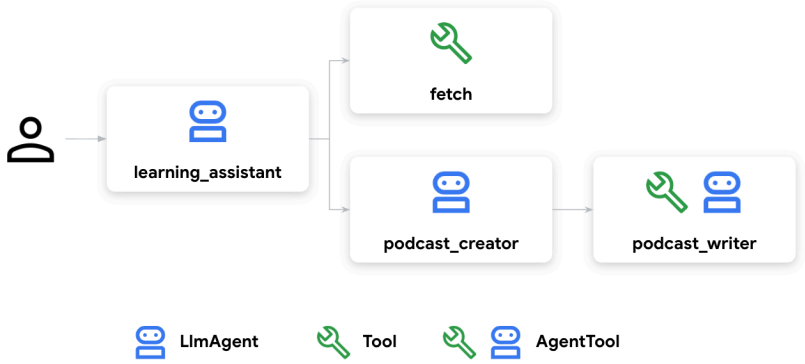
Duration: 5:00

ここまでは AI エージェントをローカル環境上 (Cloud Shell Editor) で開発、動作確認を行ってきました。そのため、外部 (例えばインターネットなど) からはアクセスできません。

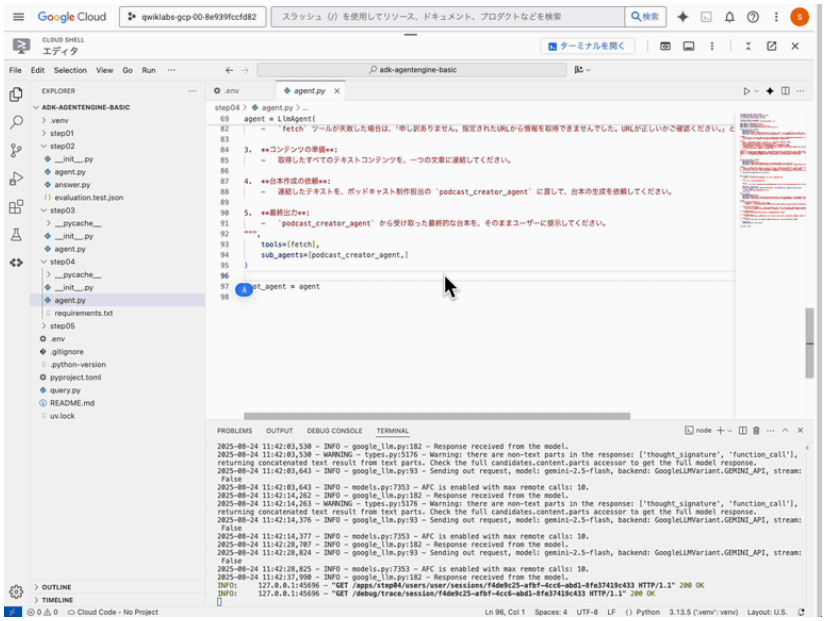
ここで、作成したポッドキャストの台本を作成する AI エージェント (`step04`) を **Agent Engine** にデプロイし、インターネットからアクセスできるようにします。

Agent Engine へのデプロイは時間がかかるため、先にコマンドを実行しておきます。

エージェント構成



作業動画



作業、動作確認手順

1. **[エディタ]** ターミナルをもう一つ新しく開きます (ターミナルの開き方は **Cloud Shell Editor** 操作方法一覧をご参照ください)
2. **[エディタ]** ターミナルの新しいセッションから以下のコマンドを入力し、デプロイを開始します

```
uv run adk deploy agent_engine --project ${GOOGLE_CLOUD_PROJECT} --region us-central1 --staging_bucket gs://${GOOGLE_CLOUD_PROJECT}-b
step04/
```

3. **[エディタ]** 少し待つとターミナルにデプロイが開始されたとのメッセージが表示されます
4. **[エディタ]** 右下のターミナル選択画面で元々起動していたターミナル (node と書いてあるはずです) をクリックし、戻ります

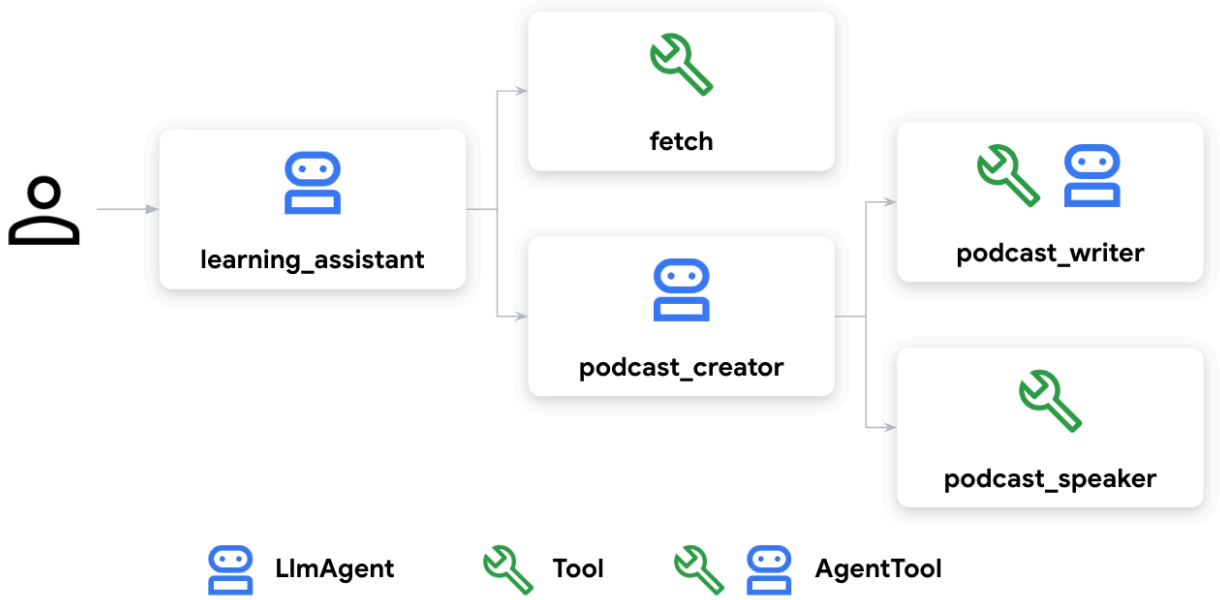
サブエージェントに独自ツールを追加する (マルチエージェント構成)

Duration: 10:00

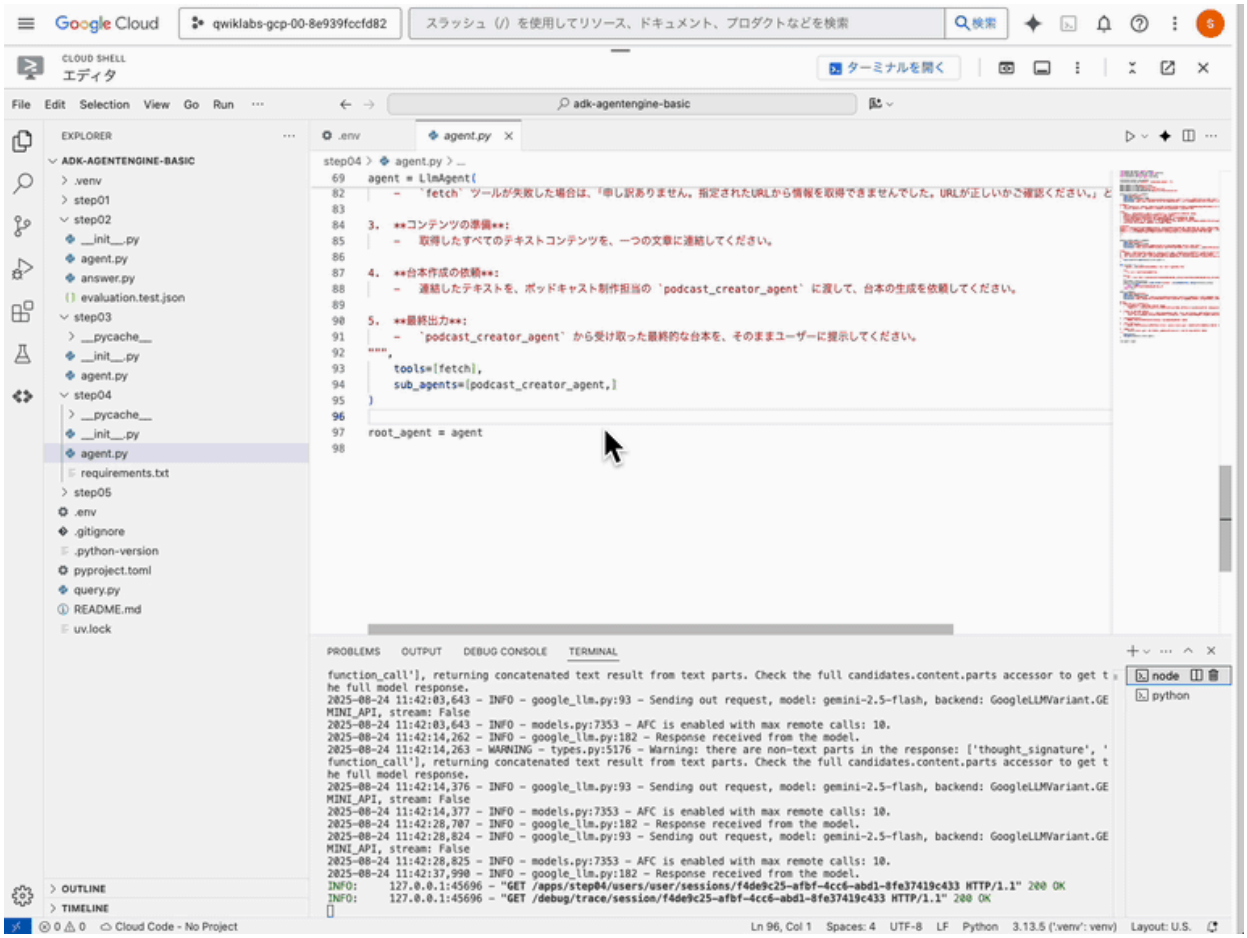
生成したポッドキャスト台本 (スクリプト) を、音声データに変換する機能を追加します。

podcast_creator に新しいツールとして音声データを作成する機能 (podcast_speaker) を追加します。

エージェント構成



作業動画



作業、動作確認手順

1. [エディタ] エクスプローラ (左側のファイル一覧ペイン) から、`step05`、`agent.py` の順にクリックします
2. [エディタ] `step04` の `agent.py` との違いを確認します
3. [Web UI] 左上にあるエージェント選択欄から `step05` を選択します
4. [Web UI] チャットウィンドウから AI エージェントのできることを問い合わせます (例、あなたは何ができますか？)
5. [Web UI] チャットウィンドウからポッドキャストにしたいコンテンツの URL を含めて問い合わせを行います
💡 ヒント: コンテンツ (URL) が思いつかない方は、以下のページから取得すると便利です

- Zenn トップページ
- Qiita 週間トレンド記事一覧

6. [Web UI] チャットウィンドウでツール (`fetch`) が呼び出された後、`podcast_creator_agent` への転送 (`transfer_to_agent`)、ツール (`podcast_writer_agent`) が呼び出され、最後に音声化を行うツール (`podcast_speaker`) が呼び出され、ポッドキャストの音声が出力されたことを確認します

ソースコードのポイント

- **podcast_creator の Tools:** `podcast_speaker` が新しく追加されている。
- **podcast_speaker:** Gemini の音声化を実現するモデル (`gemini-2.5-flash-tts-preview`) を利用して、ポッドキャストを音声化している。
- **podcast_speaker の処理:** `podcast_speaker` の実際の処理は `tools.py` に記載されている。

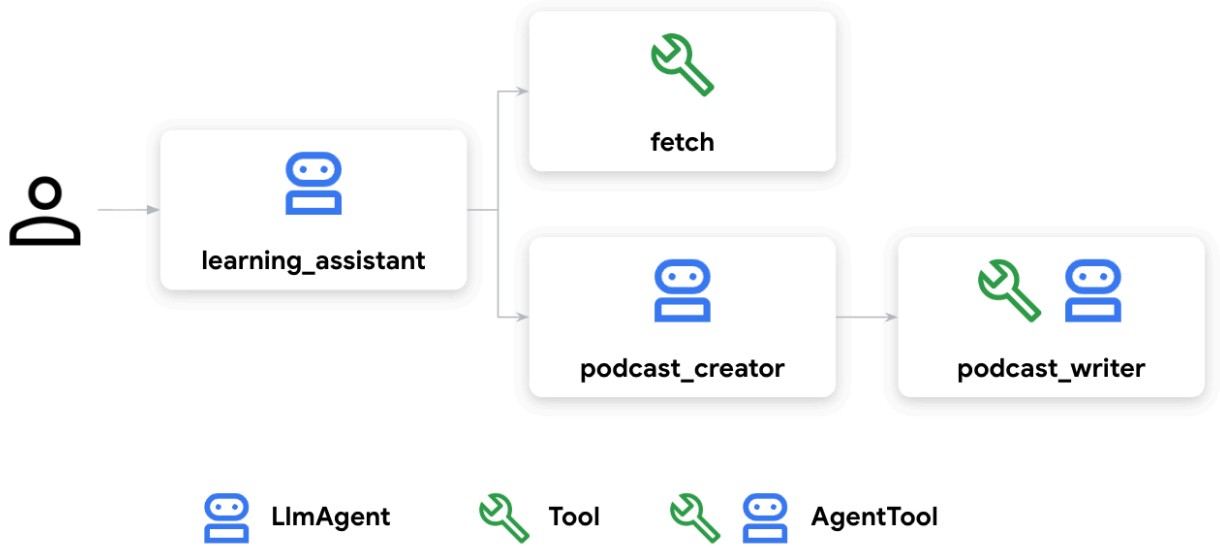
Agent Engine にデプロイした AI エージェントを使ってみる

Duration: 5:00

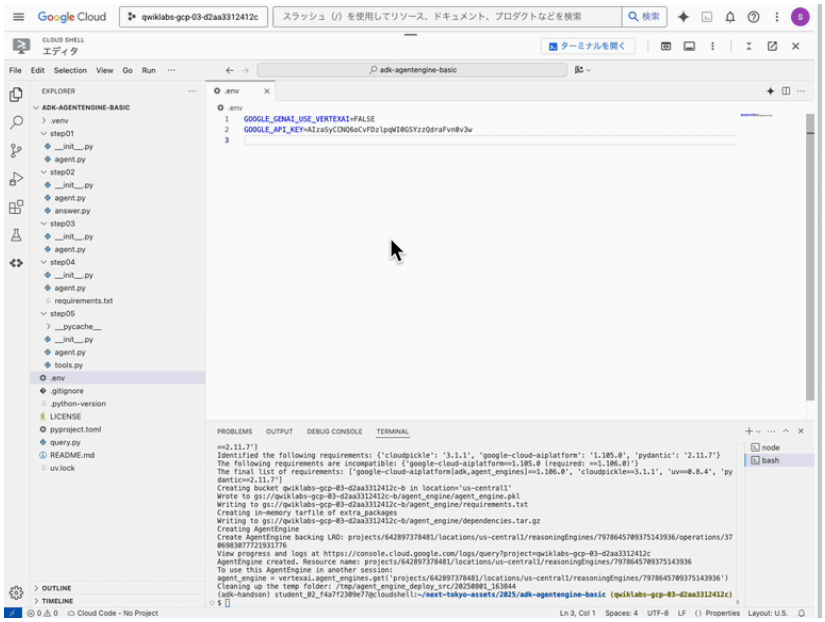
事前に URL からポッドキャスト用台本を作成する AI エージェントを Agent Engine 上にデプロイしました。

ここで、デプロイした **AI** エージェントにリクエストを投げて、ちゃんと動いていることを確認します。

エージェント構成



作業動画



作業、動作確認手順

1. **[エディタ]** 右下のターミナル選択画面から、先程 Agent Engine にデプロイをしていたセッション (bash と書かれているはずです) をクリックし、セッションを移ります
2. **[エディタ]** ログの最終行に以下のようなメッセージが表示されていれば、デプロイが完了しています

```
Cleaning up the temp folder: /tmp/agent_engine〜
```

3. **[エディタ]** 以下のコマンドを実行し、デプロイした AI エージェントに問い合わせします (例、あなたは何ができますか？)

```
uv run python query.py "あなたは何ができますか？"
```

4. **[エディタ]** 以下のコマンドを実行し、メッセージに URL を与えてポッドキャスト台本の生成を依頼します。(URL 部分は好きなものに変更してください)

```
uv run python query.py "この内容をポッドキャスト化してください。https://zenn.dev/google_cloud_jp/articles/ff74bf18e44f97"
```

5. **[エディタ]** 少し待つと、ポッドキャスト台本が出力されることを確認します

[チャレンジ問題] ポッドキャストの出力をカスタマイズする

Duration: 10:00

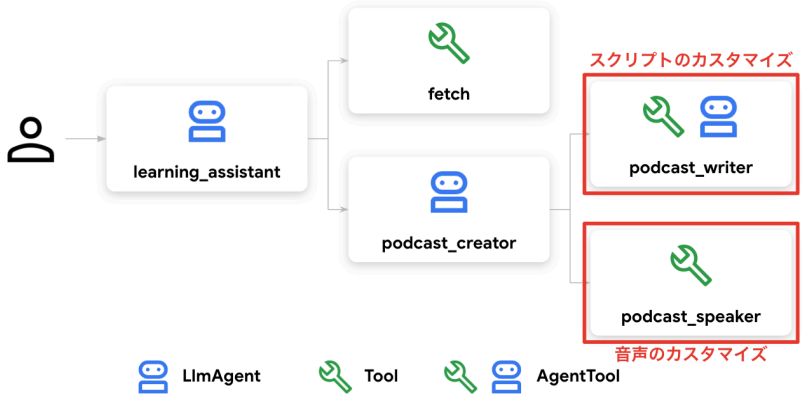
現在のポッドキャストの音声は、標準語かつ明るいトーンで出力されています。

これを `step05` フォルダにあるソースコード (プロンプト) を編集し、カスタマイズしてみましょう。

カスタマイズ例:

- 出力スクリプトのカスタマイズ:方言(例、関西弁、津軽弁)、言い回し(例、若者言葉、ギャル語)
- 音声のカスタマイズ:トーン(例、暗いトーン、怖いトーン)

エージェント構成



作業手順

1. **[エディタ]** エクスプローラ (左側のファイル一覧ペイン) から、`step05`、`agent.py` の順にクリックします
2. **[エディタ]** ソースコードを編集します
💡 ヒント: `agent.py` の `podcast_writer_agent` で出力スクリプトの内容を、同じ `step05` フォルダにある `tools.py` にある `podcast_speaker` を書き換えるとトーンをカスタマイズすることができます
3. **[Web UI]** 一度、ブラウザから Web UI をリロードします (Mac の場合は Command + R, Windows の場合は Ctrl + R)
4. **[Web UI]** 左上にあるエージェント選択欄から `step05` を選択します
5. **[Web UI]** チャットウィンドウからポッドキャストにしたいコンテンツの URL を含めて問い合わせを行います
💡 ヒント: コンテンツ (URL) が思いつかない方は、以下のページから取得すると便利です
   - Zenn トップページ
   - Qiita 週間トレンド記事一覧
6. **[Web UI]** ポッドキャストのスクリプト、音声が期待したとおりに出力されているかを確認します

エージェントはどのように評価 (テスト) するか？

Duration: 3:00

AI エージェントは生成 AI モデルを基盤としています。生成 **AI** は非決定論的（実行するたびに結果が変わりうる）な振る舞いをするため、AI エージェントのテストは通常のソフトウェア開発よりも一層重要となります。

そのため、AI エージェントはその非決定論的な振る舞いに対応したテストの考え方が必要です。
ADK では、エージェントの動作を以下の **2** つの側面から評価します。

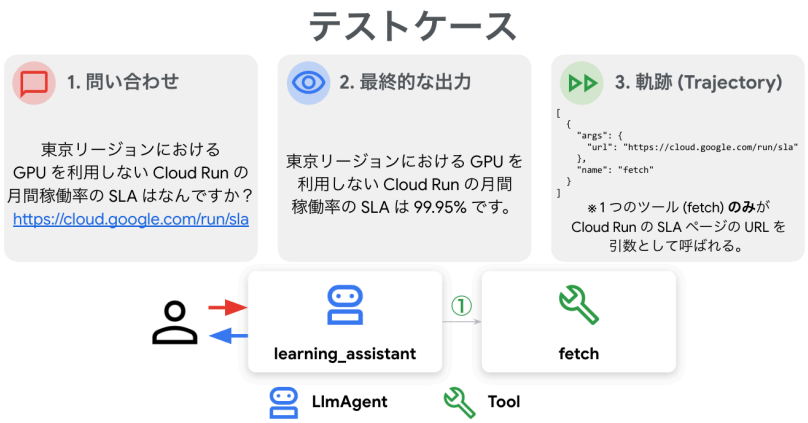
より詳細に知りたい方は “なぜエージェントを評価するのか” をご覧ください。

シンプルなテスト構成を考える

Duration: 3:00

ここでは `step02` で利用した、シングルエージェント＋ツール のテスト構成を考えます。

今回は以下のテストケースを採用します。



すでに今回のテストケースを記載したファイルは生成済みです。

作業手順

1. **[エディタ]** エクスプローラ (左側のファイル一覧ペイン) から、`step02`、`evaluation.test.json` の順にクリックします
2. **[エディタ]** ファイルの中身が上記のテストケースをどのように表しているかを確認します

テストケースのポイント

- どこに問い合わせ (入力) が書かれているか
- どこに想定の最終的な出力が書かれているか
- どこに想定の軌跡 (Trajectory) が書かれているか

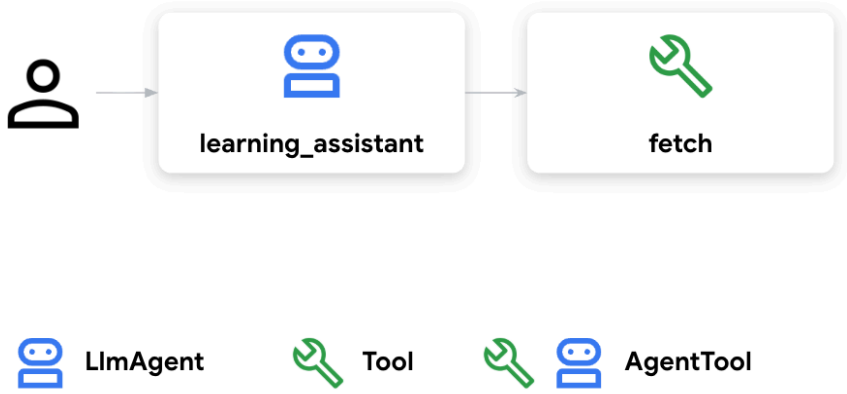
ADK のテストはデフォルトで、以下のような動作をします。
- 最終出力結果: ROUGE-1 メトリクスで比較されます。また完全一致ではなく 0.8 以上で Pass とみなされます。
- 軌跡 (**Trajectory**): Tool の呼び出し数、呼び出し順序、呼び出しの引数それぞれが完全一致した場合、Pass とみなされます。

テストを実行する

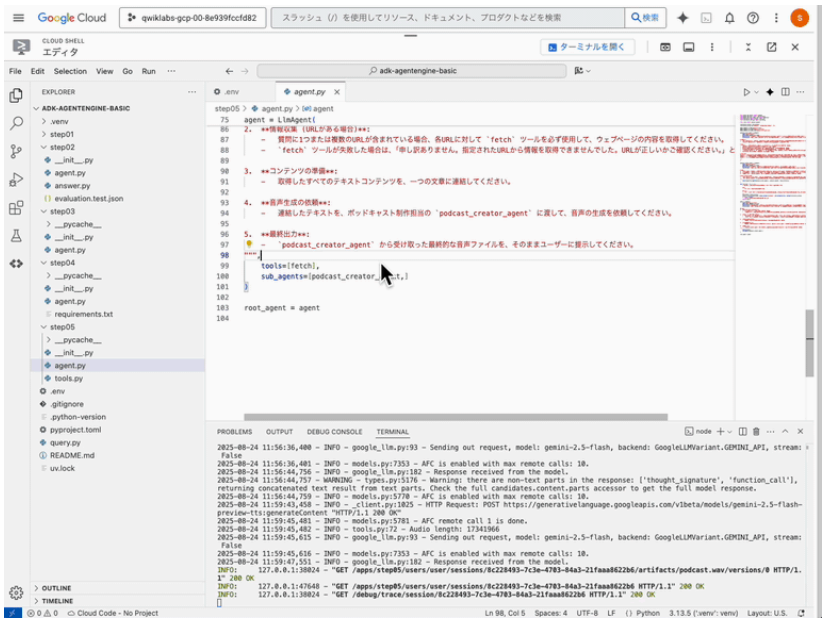
Duration: 5:00

先ほど内容を確認したテストを実行しましょう。

エージェント構成 (再掲)



作業動画



作業手順

1. **[エディタ]** ターミナルをもう一つ新しく開きます (ターミナルの開き方は **Cloud Shell Editor** 操作方法一覧をご参照ください)
2. **[エディタ]** 新しく開いたターミナルから以下のコマンドを入力し、テストを実行します

```
uv run adk eval step02/ step02/evaluation.test.json
```

3. **[エディタ]** 少し待つとターミナルにテスト実行結果が表示されます。うまくいくと以下のような結果になるはずです。

```
**********************************************************************
Eval Run Summary
evalset_step02:
  Tests passed: 1
```

```
Tests failed: 0
```

💡 ヒント: `failed` が 1 になっている場合は、手順 2 のコマンドに `--print_detailed_results` のオプションを付けて実行してみてください。そうすると実際に実行した結果とテストケースに記載した内容の両方を確認することが可能です。

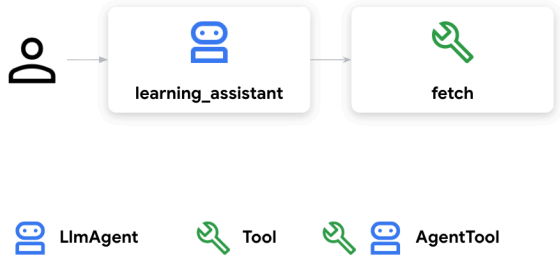
テストケースを生成する

Duration: 5:00

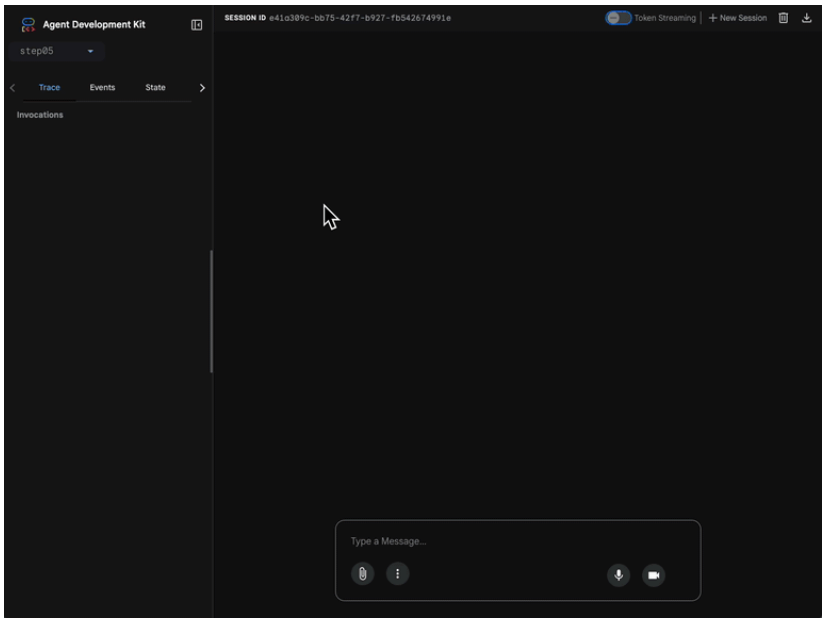
前のステップではすでに作成済みのテストケースファイルを利用して、テストを実行しました。しかしテストケースのファイルを一からスクラッチで作成するのは手間がかかる作業です。

ここでは **Web UI** で **AI** エージェントとやり取りし、その内容を元にテストケースのファイルを作成します。

エージェント構成 (再掲)



作業動画



作業手順

1. **[Web UI]** 左上にあるエージェント選択欄から `step02` を選択します
2. **[Web UI]** テストケースに適した質問を投げます

```
東京リージョンにおける GPU を利用しない Cloud Run の月間稼働率の SLA はなんですか？ https://cloud.google.com/run/sla
```

3. **[Web UI]** 結果が正しいことを確認します
4. **[Web UI]** 左のメニューから `Eval` をクリックします (一番右にあります)
5. **[Web UI]** `Create Evaluation Set` をクリックします
6. **[Web UI]** `evalset_step02` と入力し、`Create` をクリックします
7. **[Web UI]** 左のメニューに作成された `evalset_step02` をクリックします
8. **[Web UI]** `Add current session to evalset_step02` をクリックします
9. **[Web UI]** `case01` と入力し、`Create` をクリックします

10. **[エディタ]** step02 フォルダ配下に `evalset_step02.evalset.json` という名前のテストファイルが出来上がっていることを確認します

Congratulations!

Duration: 3:00

「**Agent Development Kit**（**ADK**）と **Agent Engine** で学ぶ！**AI** エージェント開発。基礎から実践まで」に
ご参加頂きありがとうございました！

本日のハンズオンが少しでも Google Cloud を使った AI エージェント開発の参考になれば、これほど嬉しいことはありません。

引き続き、このようなハンズオンイベントを続けて参りますので
ご要望、フィードバックがございましたら是非アンケートでお寄せください。
