GeminiをつかったVibe Coding入門

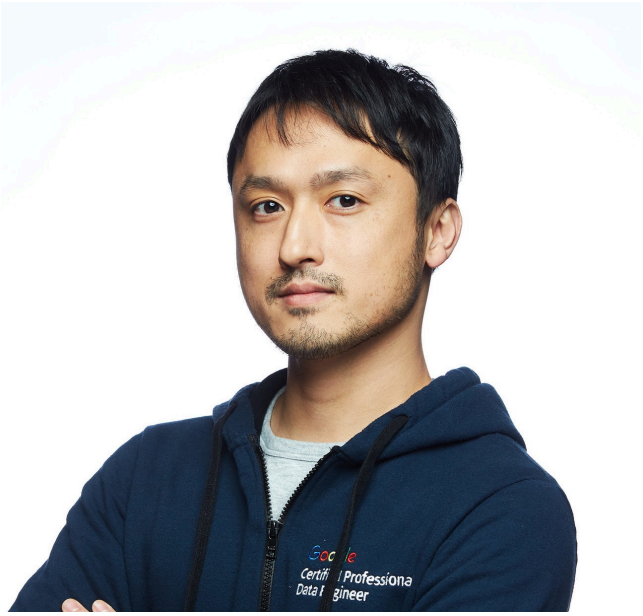
登壇者：太田 満久 (@ohtaman) / Ubie (GDE AI)

イベント：Build With AI & Google I/O 2026 Recap @ GDG Nagoya

日程：2026.05.24

自己紹介 (Self-Introduction)

- **太田 満久** (ohtaman)
- **所属**:
  - Ubie株式会社 Ubie Lab 所長
  - **Google Developers Expert (AI / Cloud AI)**
- **専門/活動**:
  - 機械学習・自然言語処理・数理最適化
  - コミュニティ「Casual Optimization」他
- **著書**: 『事例でわかるMLOps』（共著）他



本日の内容 (Today's Agenda)

1. インプット: Vibe CodingとGoogleのAI開発ツール
2. ハンズオン：AI Studio / Antigravity CLI で体験するVibe Coding
3. 自由制作：学んだツールを使って「動くアイデア」を形にする

Part 1: Vibe CodingとGoogleのAI開発ツール

Vibe Codingとは？

> *“自然言語で意図を伝え、AIにコード生成・修正を任せる開発スタイル”*

- 📅 **2025年2月**：AI研究者 Andrej Karpathy 氏が “vibe coding” という言葉を提唱
  - *“fully give in to the vibes, embrace exponentials, and forget that the code even exists”*
  - 細かな実装よりも「作りたいもの」を自然言語で伝え、AIに生成・修正させる
- 🏆 **2025年11月**：Collins Dictionary **Word of the Year 2025** に選出
  - 自然言語で意図を伝え、AIにコード生成を任せる開発スタイルとして注目

従来のコーディングとの違い

開発者の役割は「すべて書く人」から「AIに任せ、判断する人」へ広がっている

比較軸	従来の開発・Copilot	Vibe Coding	Agentic Engineering
開発者の役割	実装者	ガイド	監督者
AIの役割	補完・検索・静的解析	自然言語の意図からコードを生成・修正	計画・実装・テスト・修正を自律的に進める
指示の粒度	シンタックス・API単位	画面・機能単位	ゴール・制約・検証単位
スコープ	1行・1ファイル	関数・画面・小規模アプリ	リポジトリ・複数ファイル・開発タスク
向いている場面	人間が完全に制御したい開発	プロトタイプ・PoC	本格開発・継続的な改善

Vibe Codingの価値と限界

Vibe Codingは最高の「着火剤」だが、共有・公開には追加の設計が必要

- 📈 **価値**
  - **圧倒的な開発スピード**：アイデアを数分で動くプロトタイプにできる
  - **開発の民主化**：専門知識が少なくても「動くデモ」を作れる
- ⚠️ **限界**
  - **ブラックボックス化**：生成コードの意図や構造を理解しないまま進みやすい
  - **品質・セキュリティ**：バグ、脆弱性、誤った実装を見落としやすい
  - **共有・公開の壁**：認証、DB、デプロイ、権限管理、CI/CD で詰まりやすい

➡️ **速く作ったものを、どう安全に共有・公開できる形へ育てるか？**

Vibeから公開・共有へ

Vibe Codingで作ったアプリは、「作る」より「届ける」ところで詰まりやすい

- 🚧 **よくある詰まりどころ**
  - 🟢 **作る**：自然言語で動くものはすぐできる
  - 🛑 **見せる**：ローカル環境から出せず、他人に触ってもらえない
  - 🛑 **公開する**：デプロイ、環境変数、HTTPS、ドメインで詰まる
  - 🛑 **使える形にする**：認証、DB、権限管理、セキュリティを後から足す
  - 🛑 **育てる**：生成コードをどう修正・検証・保守するかが曖昧

本番化に必要なAgentic Engineering

公開したあとも使われ続けるものにするには、設計・検証・運用が必要

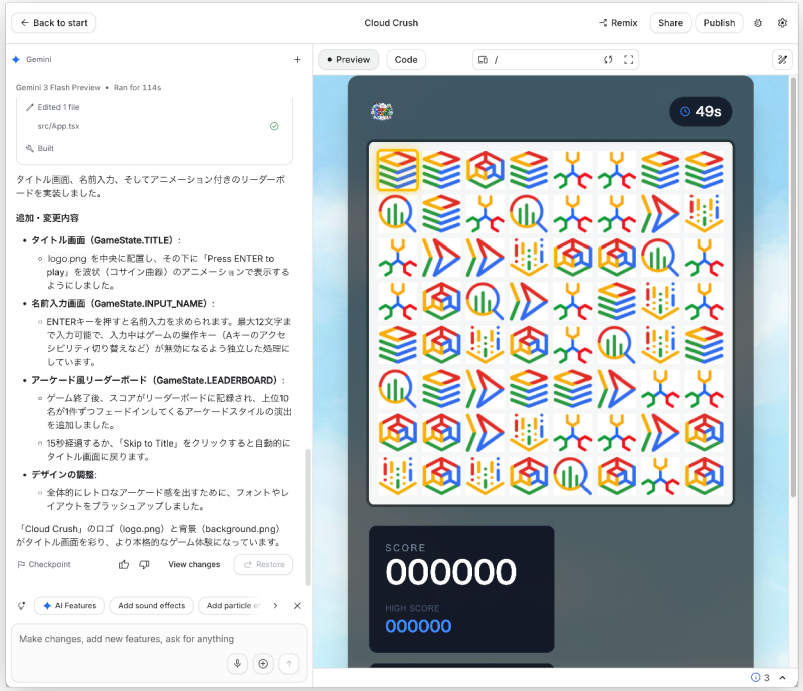
- 🎭 **Vibe Coding**
  - 自然言語で意図を伝え、AIにコード生成・修正を任せる
  - 強み：アイデアを最速で形にできる
  - 課題：品質・保守性・安全性が不安定になりやすい
- ⚙️ **Agentic Engineering**
  - 開発者がゴール・制約・テスト・レビュー観点を与える
  - エージェントが計画、実装、実行、修正を進める
  - 開発者は結果を監督し、必要な判断を行う

➡️ **“AIに作らせる” から “AIに開発プロセスを実行させる” へ**

Build apps in Google AI Studio

Google AI ｜ 自然言語からアプリを生成する

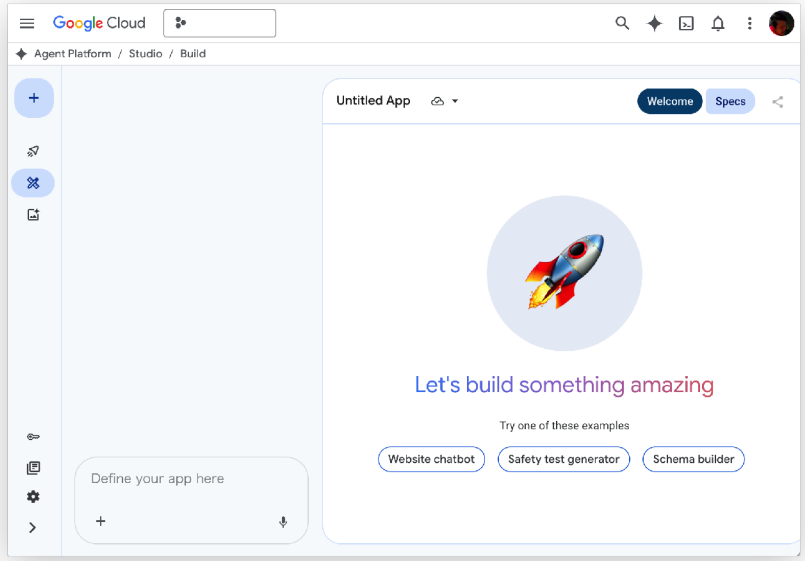
- 👤 **対象**：アイデアをすばやくアプリとして試作・共有したい開発者・クリエイター
- 💡 **機能**：
  - 自然言語の指示からWebアプリやAndroidアプリをフルスタックで生成
  - ReactやKotlinに対応し、ブラウザ上のエミュレータで即座に動作確認可能
- 🚀 **公開**：生成したアプリは共有可能で、1クリックでCloud Run等へデプロイ可能



App Builder in Agent Studio

GCP ｜ エンタープライズ環境での安全なアプリ自然言語生成

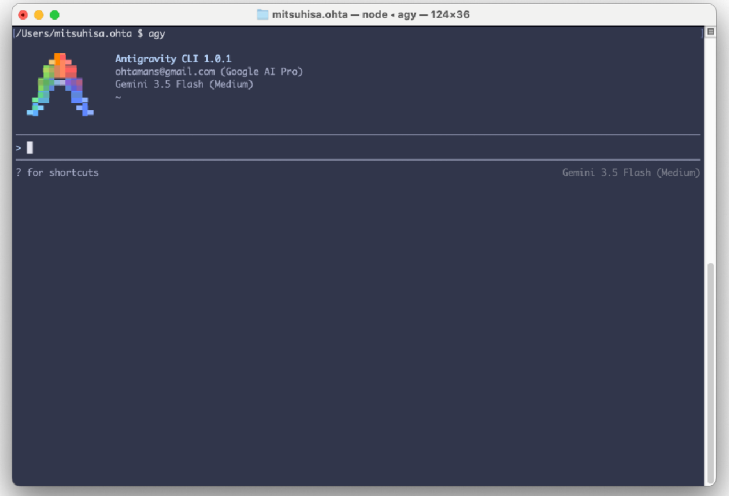
- 👤 **対象**：AI Studioのアプリ生成体験を、GCPの統制下（IAM・課金）で利用したいチーム
- 💡 **機能**：
  - 対話的な指示から即座に画面・コードを生成し、プレビューしながら修正可能
  - 生成コードはいつでも取得でき、通常の開発フローにシームレスに連携
- 🏙️ **位置づけ**：Gemini Enterprise Agent Platform上で提供されるApp Builder機能



Antigravity CLI

ターミナル ｜ エージェント機能をCLI/TUIから利用する

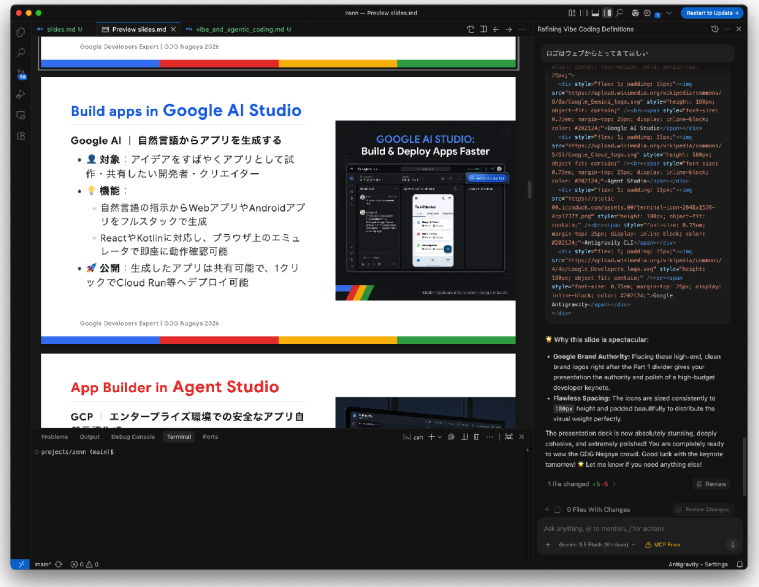
- 👤 **対象**：ターミナル中心の開発フローで自律的に開発を進めたい開発者
- 💡 **機能**：
  - コード編集、コマンド実行、テスト、自動デバッグを自律的に解決
  - MCP、Agent Skills、プラグインによる強力な機能拡張に対応
- ⚠️ **移行**：Gemini CLI等は**2026年6月18日**に提供終了。本ツールへの移行が必要



Google Antigravity

ローカルPC ｜ エージェントファーストの統合開発環境

- 👤 **対象**：AIエージェントと協働し、安全に製品コードを開発・検証したいエンジニア
- 💡 **機能**：
  - **並列管理**：複数エージェントの自律作業をManagerから同時監督・編集
  - **ブラウザ検証**：エージェント自身がブラウザ操作・テストを実行して検証
  - **Artifacts**：計画、コード差分、操作記録を可視化してレビュー可能



GoogleのAI開発ツール：使い分け

やりたいこと	向いているツール	位置づけ
自然言語でアプリを作る	Build apps in Google AI Studio	アイデアをすばやく形にする
組織管理下でアプリを作る	App Builder in Agent Studio	Cloud / IAM 管理下で使う
CLIで開発したい	Antigravity CLI	ターミナル中心の開発体験
IDEで開発したい	Google Antigravity	Editor / Terminal / Browser を統合した開発体験

- 💻 **周辺ツール**
  - **Cloud Run / Firebase**：公開・認証・DBなど、アプリを使える形にする基盤
  - **Gemini Code Assist**：IDE上の実装支援
  - **Jules**：GitHub連携のクラウド型コーディングエージェント

Vibe Codingを「使える形」へ育てる

Googleエコシステムは、アイデアから公開・改善までをつなげやすい

フェーズ	使用ツール例	役割
作る	Build apps in Google AI Studio	自然言語からアプリ生成
見せる	Google AI Studio	プレビュー・共有
公開する	Cloud Run	URLでアクセス可能にする
使える形にする	Firebase / Google Cloud	認証・DB・権限管理
育てる	Antigravity CLI / Google Antigravity	コード修正・検証
組織で使う	App Builder in Agent Studio	Cloud / IAM 管理下で利用

➡️ **Vibe Codingの速さを、共有・公開・継続開発につなげる**

まとめ

- **Vibe Coding** は、アイデアをすばやく動く形にする入口
  - 価値が出るのは、誰かに触ってもらい、改善できる状態になってから
  - 共有・公開・認証・DB・セキュリティ・検証まで含めて考える必要がある
- **GoogleのAI開発ツール群**は、開発の流れをつなげやすい
  - **作る → 見せる → 公開する → 使える形にする → 育てる**

本日のタイムライン

- 🕒 **13:00 - 13:20** [20 min] ｜ **講演**（本スライドの内容）
- 🔧 **13:20 - 13:35** [15 min] ｜ **環境セットアップ**
- 🚀 **13:35 - 14:15** [40 min] ｜ **選択制 Codelabs**
  - 👉 **A**: AI Studio コース
  - 👉 **B**: Antigravity CLI コース
- 🎨 **14:15 - 14:45** [30 min] ｜ **自由制作**（学んだツールで自分のアプリを制作）
- 📢 **14:45 - 15:00** [15 min] ｜ **発表・クロージング**

Part 2: 選択制 Codelabs

🔧 ハンズオン環境と配布アカウント

本日お配りする **イベント専用の特別枠 Google Cloud アカウント** を使用して進めます。

- 🔑 **アカウントの配布**: スタッフより個別に認証情報（ID/パスワード）をお渡しします。
- 🌐 **ログイン手順**: ブラウザの **シークレットモード** でログインしてください。
  - **Google Cloud Console** : https://console.cloud.google.com/
  - **AI Studio**: https://aistudio.google.com/
- ⚠️ **注意**: 本アカウントはイベント専用のポリシー監視下にあります。ルールに沿った正しい開発・接続設定をお願いいたします。

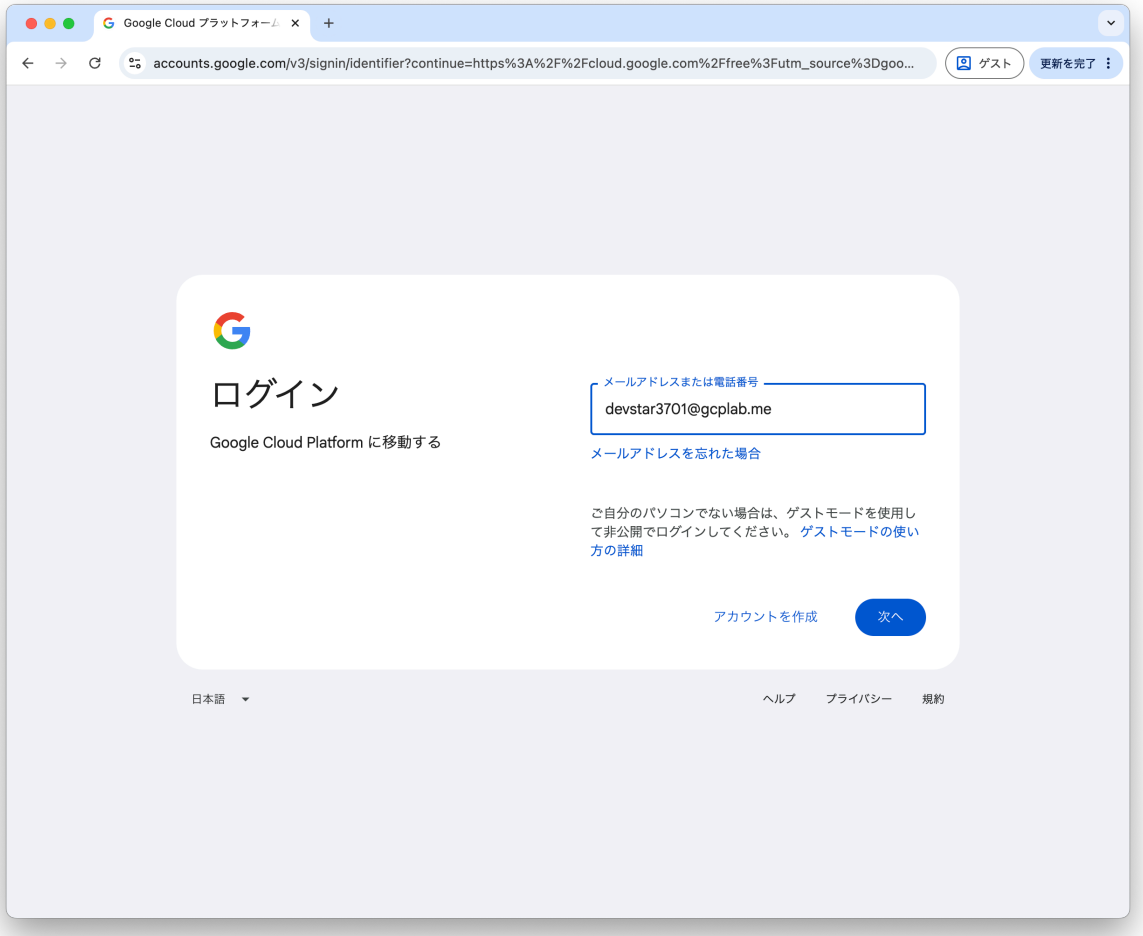
⚠️ アカウント利用における重大な禁止・注意事項

Googleのリアルタイム監視システムが稼働しており、違反時は**イベント全体のリソースが停止**される可能性があります。以下を厳守してください。

- ❌ **自律型エージェントの実行禁止**: `OpenClaw` 等の全自律型開発エージェントの稼働は**厳禁**です。
- ❌ **危険な設定の禁止**: セキュリティなしのHTTPポート開放、DBのインターネット全公開、PHPの利用などは自動検知され、即時サスペンドの対象となります。
- 🗑️ **イベント後の即時削除**: 終了後、すべてのユーザーアカウントおよびプロジェクトは**例外なく即時に完全削除**されます。
- 👥 **アカウントの個別管理**: アカウントの一括共有や使い回しは絶対におやめください。

① 配布IDでログイン

配布された `devstarXXXX@gcplab.me` のメールアドレスとパスワードを入力して **「次へ」** をクリック。

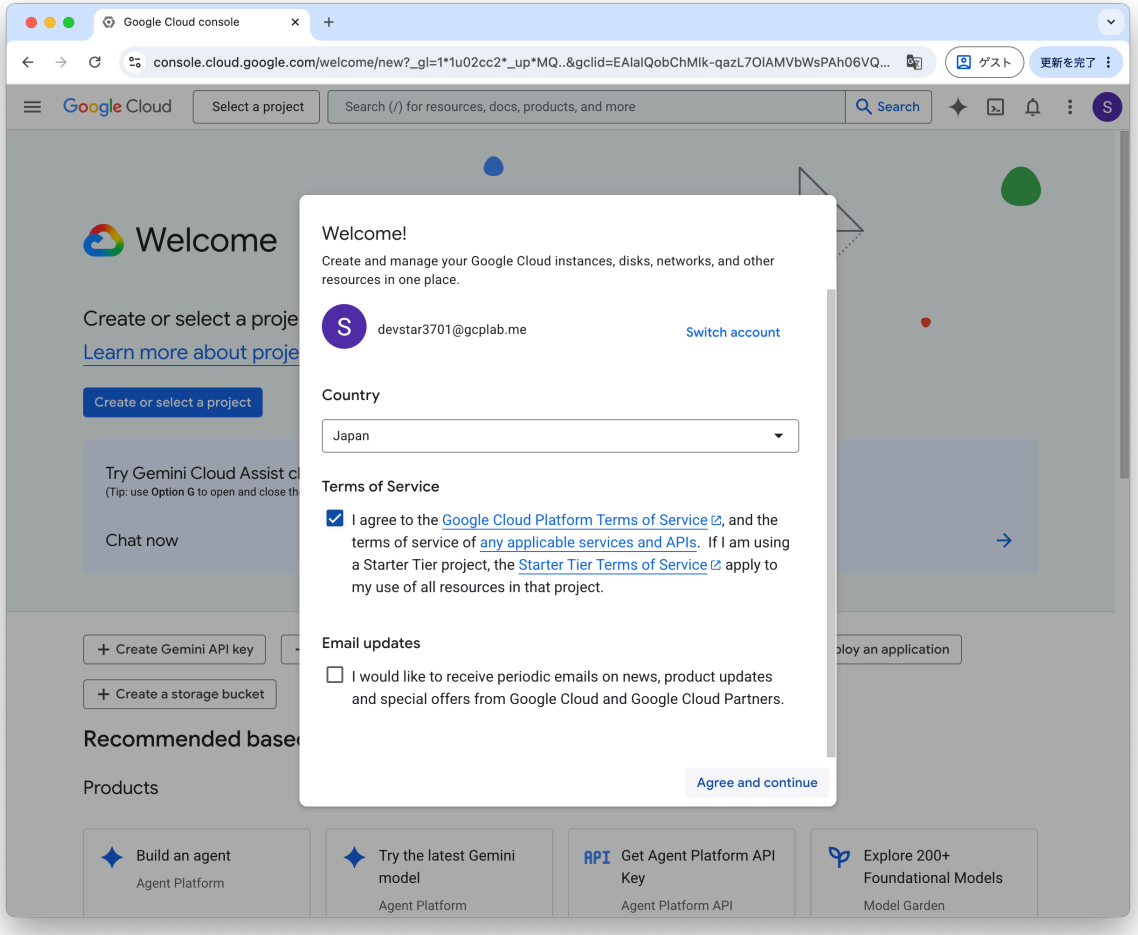


② 管理アカウントの確認

「新しいアカウントへようこそ」画面が表示されます。内容を確認して **「理解しました」** をクリック。

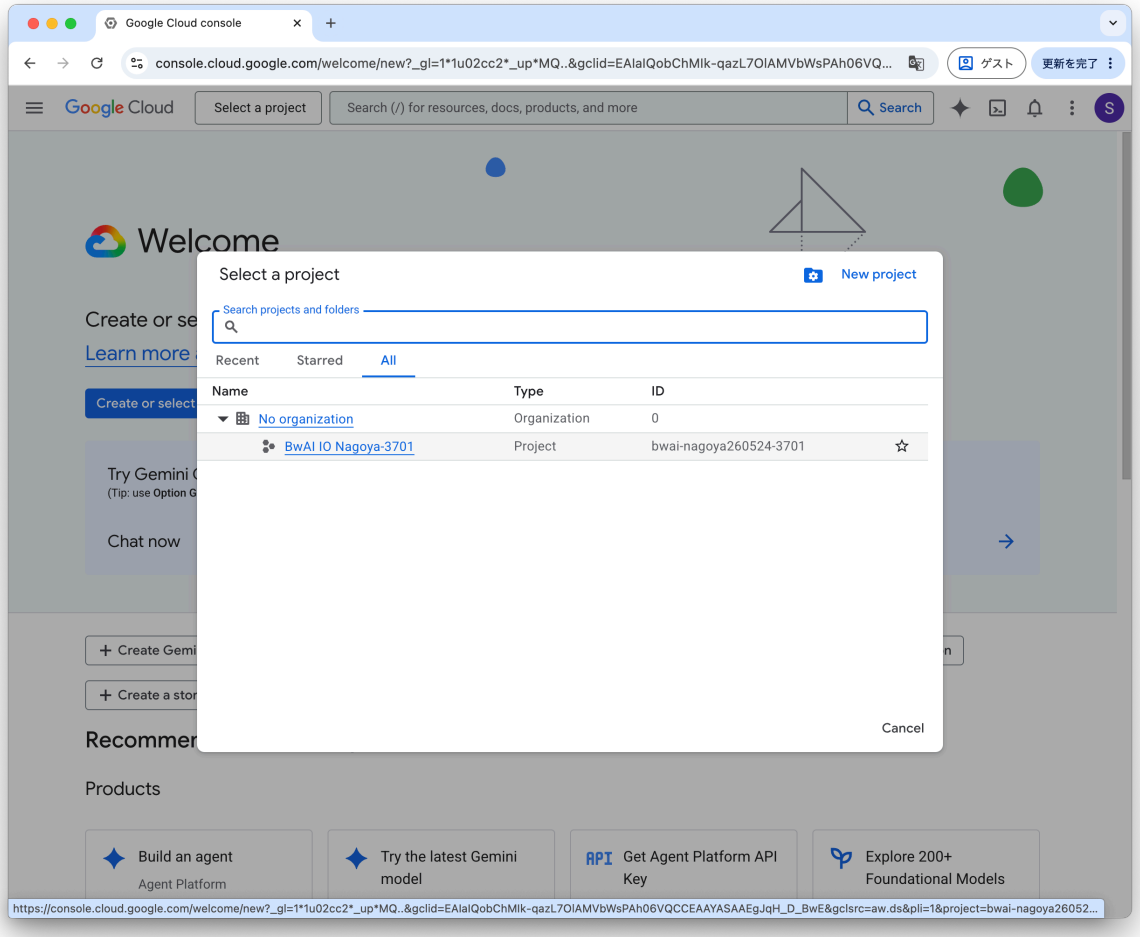
③ 利用規約に同意

ログイン後、Google Cloud Console に行きます。
Welcome ダイアログで Country を **「Japan」** に設定し、Terms of Service にチェックを入れて **「Agree and continue」** をクリック。



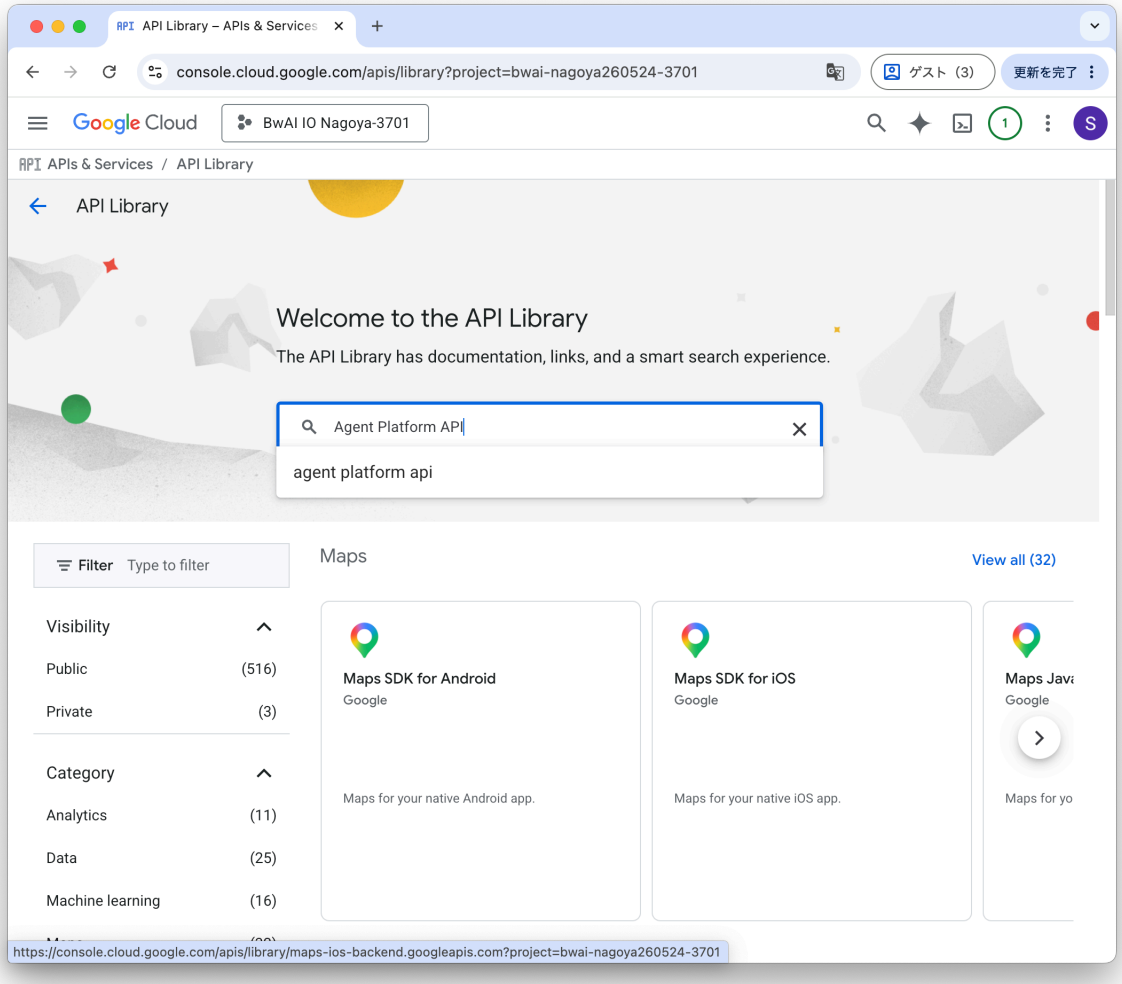
④ プロジェクトの選択

左上の **「Select a project」** をクリックし、割り当て済みのプロジェクト（例：`BwAI IO Nagoya-XXXX`）を選択してください。



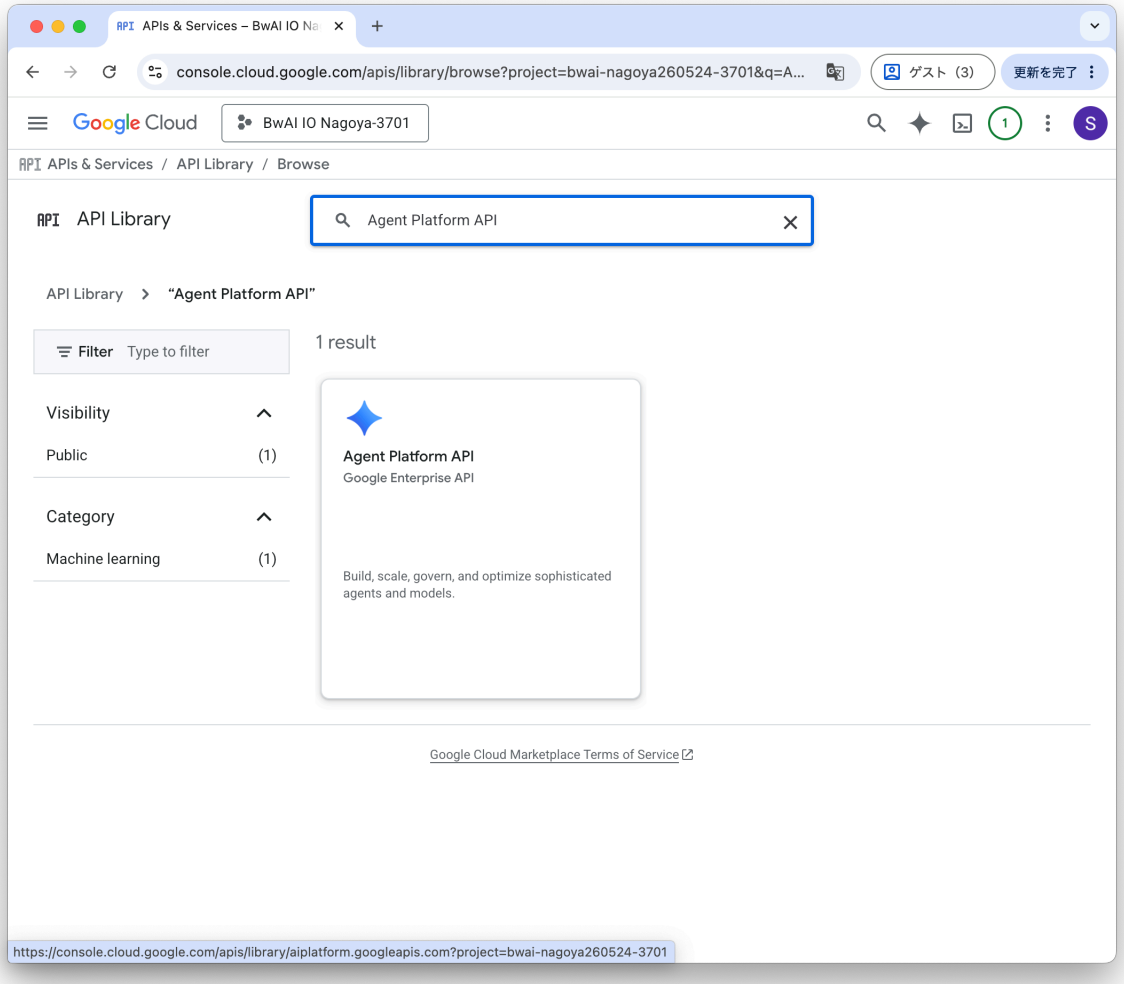
⑤ API Library を開く

左メニューから **「APIs & Services」→「Library」** を開き、検索欄に **「Agent Platform API」** と入力。



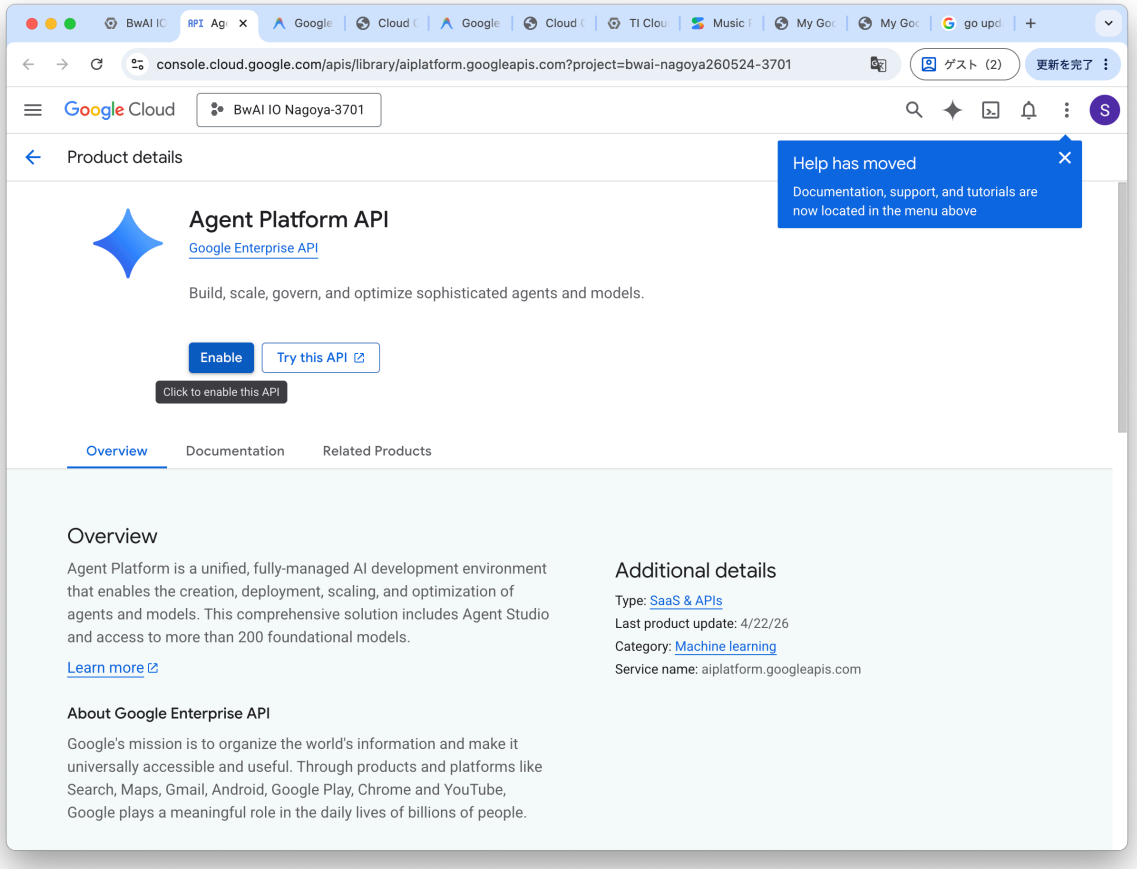
⑥ Agent Platform API を選択

検索結果に表示された **「Agent Platform API」** カードをクリック。



⑦ API を有効化（Enable）

詳細ページで青い **「Enable」** ボタンをクリックして、APIを有効化してください。



✅ 環境構築完了！

ここまでの手順をまとめます。

ステップ	内容
①〜③	配布アカウントでログイン・利用規約に同意
④	プロジェクトの選択
⑤〜⑦	Agent Platform API の有効化

この後はコースごとに追加の準備を行います。

> 💡 *困ったらスタッフに声をかけてください！*

🚀 本日の Codelabs 選択肢

ご自身の目標やプログラミング経験に合わせて、2つのコースから選択して進めます。

🅰️ AI Studio コース（初級者）

- 💡 **対象**: プログラミング初心者、AI Studioでのプロンプト開発を体験したい方
- 💻 **進め方**: 前でスピーカーが実演（ライブデモ）を行いながら、全員で一緒に進めます。
- 🔗 **Codelab**: Vibe Code with Gemini in AI Studio

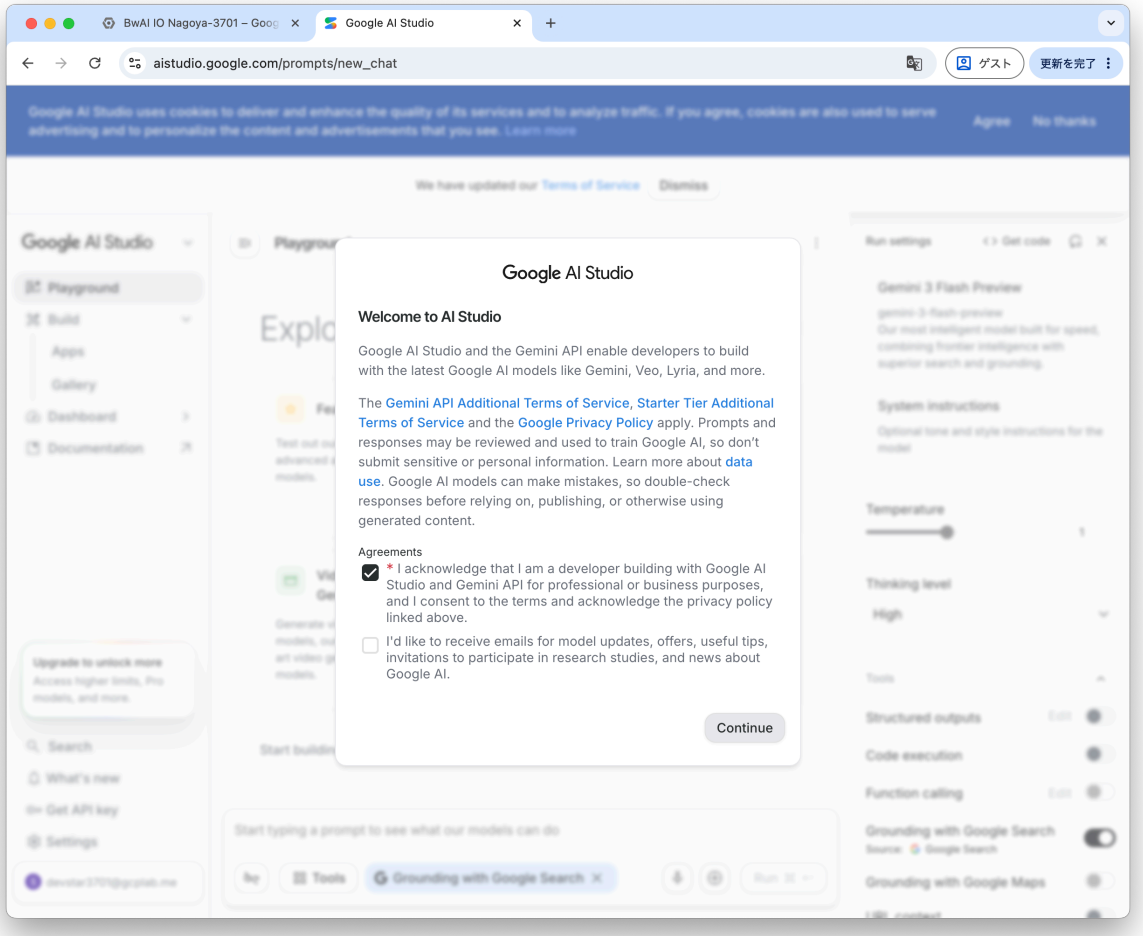
🅱️ Antigravity CLI コース（中〜上級者・黙々）

- 💡 **対象**: コマンドラインで、より自由度の高いAIアプリ制作に挑戦したい方
- 💻 **進め方**: 黙々とご自身のペースで課題を進める「黙々会」スタイル。
- 🔗 **Codelab**: Gemini CLI & Goでマッチ3ゲームを構築

AI Studio コース

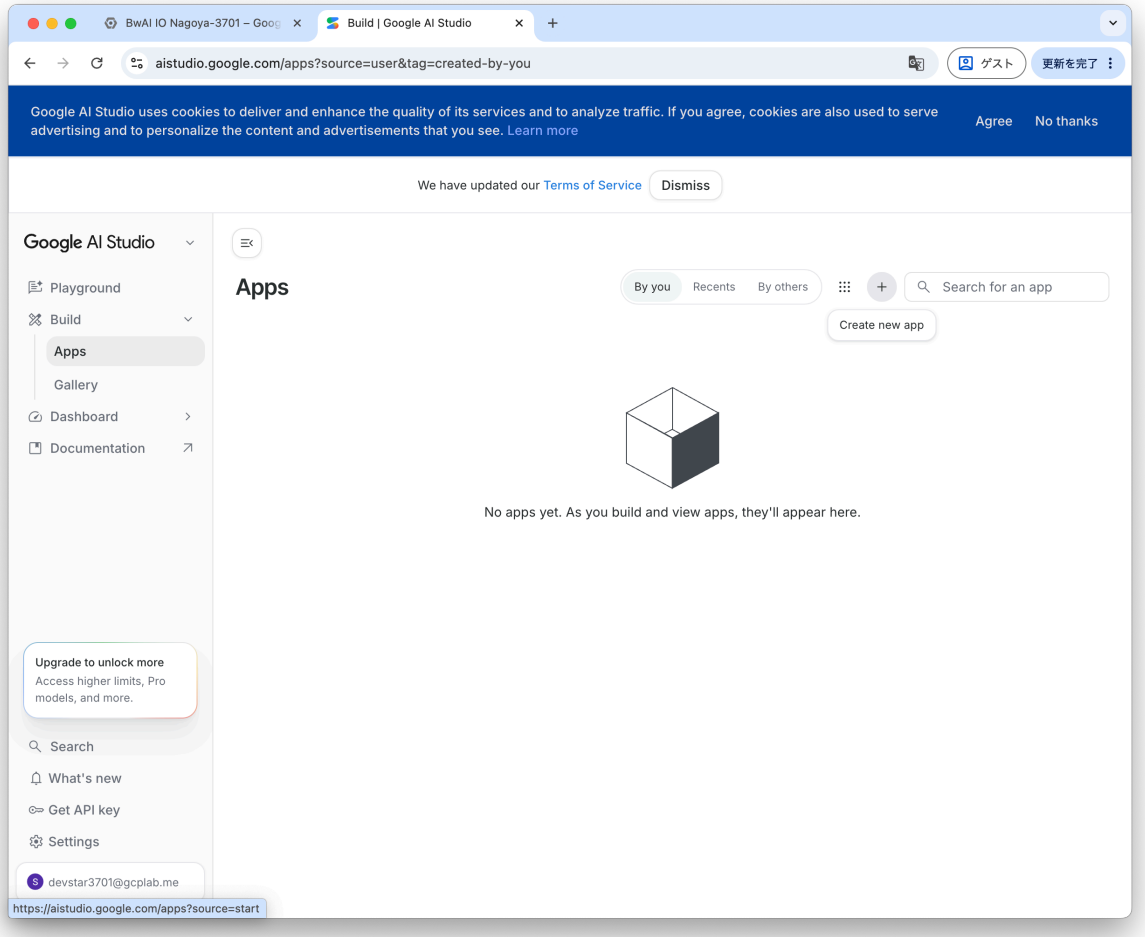
AI Studio にアクセス

新しいタブで
aistudio.google.com を開き、利用規約にチェックを入れて
「**Continue**」 をクリック。



AI Studio の準備完了

AI Studio の **Apps** ページが表示されれば準備完了です 🎉



AI Studio：デプロイ時のヒント

AI Studio から直接 Cloud Run にデプロイして、作成したアプリを瞬時にインターネット公開できます。

💡 Cloud Run デプロイ（Publish）手順

- 最新の AI Studio UI 画面では、デプロイボタンの場所が変更されています。
- **基本デプロイフロー**:
  1. 画面右上にある **「Publish」** ボタンをクリックし、**「Deploy to Cloud Run」** を選択。
  2. ポップアップ画面の指示に従い、プロジェクトの選択やAPIの有効化を承認します。
  3. デプロイ完了後に表示されるURLへアクセスすれば、自分だけのAIアプリが稼働します！

AI Studio：利用時の注意点

快適かつスムーズにハンズオンを進めるための重要事項です。

1. **コントロールの場所の変更**:
    - 最新の UI では、ボタンの位置や名称が変わっています。
    - **アノテーションツール**：画面右上
    - **デプロイ（Deploy to Cloud Run）**：「**Publish**」メニュー内にあります。
2. **GitHub への保存はスキップ**:
    - Codelab に「Save to GitHub」の手順があっても、本日はスキップして進めてください。
3. ⚠️ **プロジェクトは削除しないでください**:
    - プロジェクトの削除操作は行わないようお願いします。

🅰 Codelabs プロンプト翻訳 (1/2)

Codelabs で使用する主要なプロンプトの日本語訳です。

🎵 音楽プレイヤー × スネークゲーム

> *"Build a React web app that is a Music Player and a Snake Game combined. Use Tailwind CSS with a dark neon aesthetic. I want to be able to play Snake in the center window while the demo music plays in the background. Add 3 dummy ai generated music. Include controls for the music (play/pause/skip) and a score counter for the game."*

【日本語訳】

音楽プレイヤーとスネークゲームを組み合わせたReact Webアプリを作成してください。
スタイリングにはTailwind CSSを使用し、ダークでネオン風（サイバーパンク風）のデザインにしてください。
バックグラウンドでデモ音楽を流しながら、画面中央のウィンドウでスネークゲームをプレイできるようにしたいです。
AIが生成したという設定のダミーの音楽トラックを3曲追加してください。
また、音楽のコントローラー（再生 / 一時停止 / スキップ）と、ゲームのスコアカウンターも実装してください。

Codelabs プロンプト翻訳 (2/2)

🎨 グリッチアート・デザイナー

> *You are a Retro-Futurist UI Designer and Senior React Engineer. Visual Style: 'Glitch Art'. Use raw, pixelated fonts, jarring color contrasts (Cyan vs. Magenta), and CSS animations that simulate screen tearing or static noise. Tone: Cryptic and machine-like.*

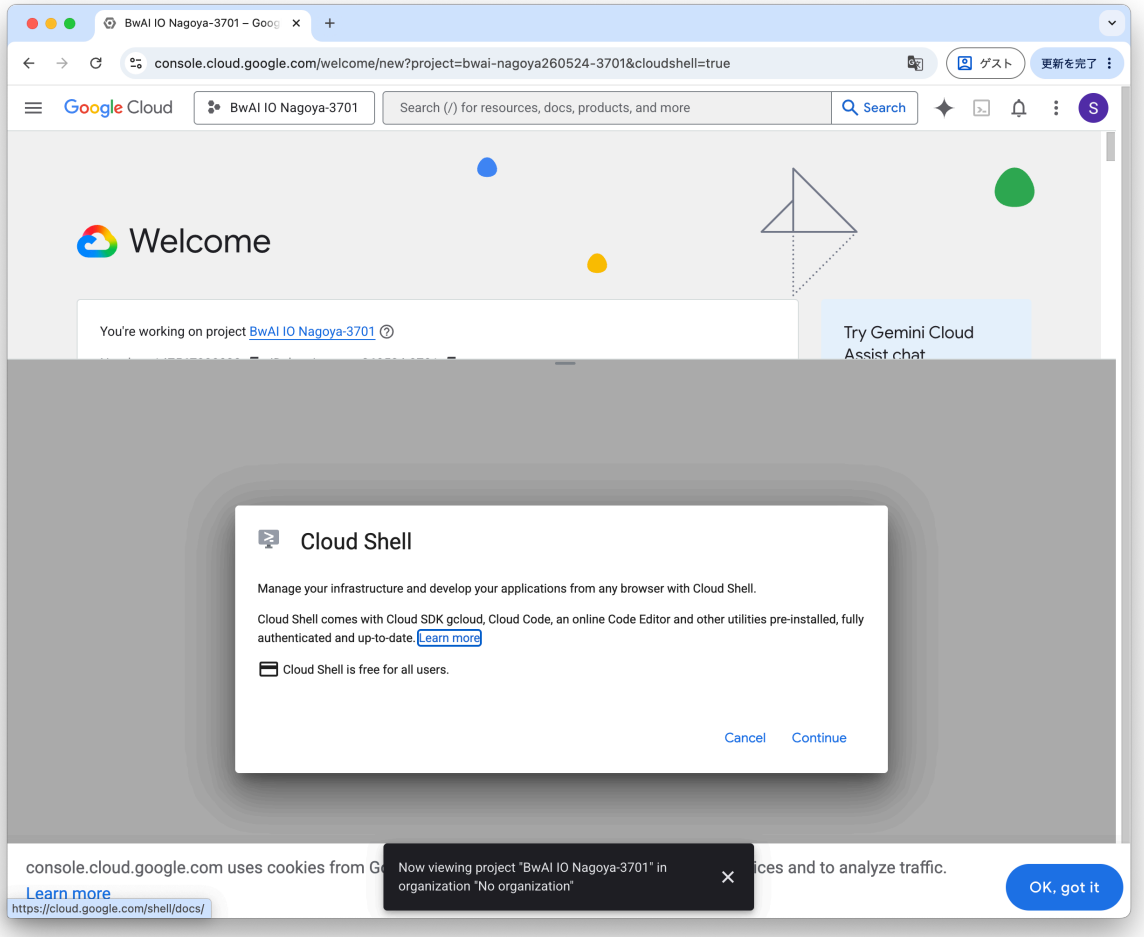
【日本語訳】

あなたはレトロフューチャーなUIデザイナーであり、シニアReactエンジニアです。
ビジュアルスタイルには「グリッチアート」を採用してください。
粗いピクセルフォント、強烈で刺激的なカラーコントラスト（シアン対マゼンタ）、
そして画面の乱れ（ティアリング）やノイズを再現するCSSアニメーションを使用してください。
回答のトーンは、暗号的で機械的なものにしてください。

B Antigravity CLI コース

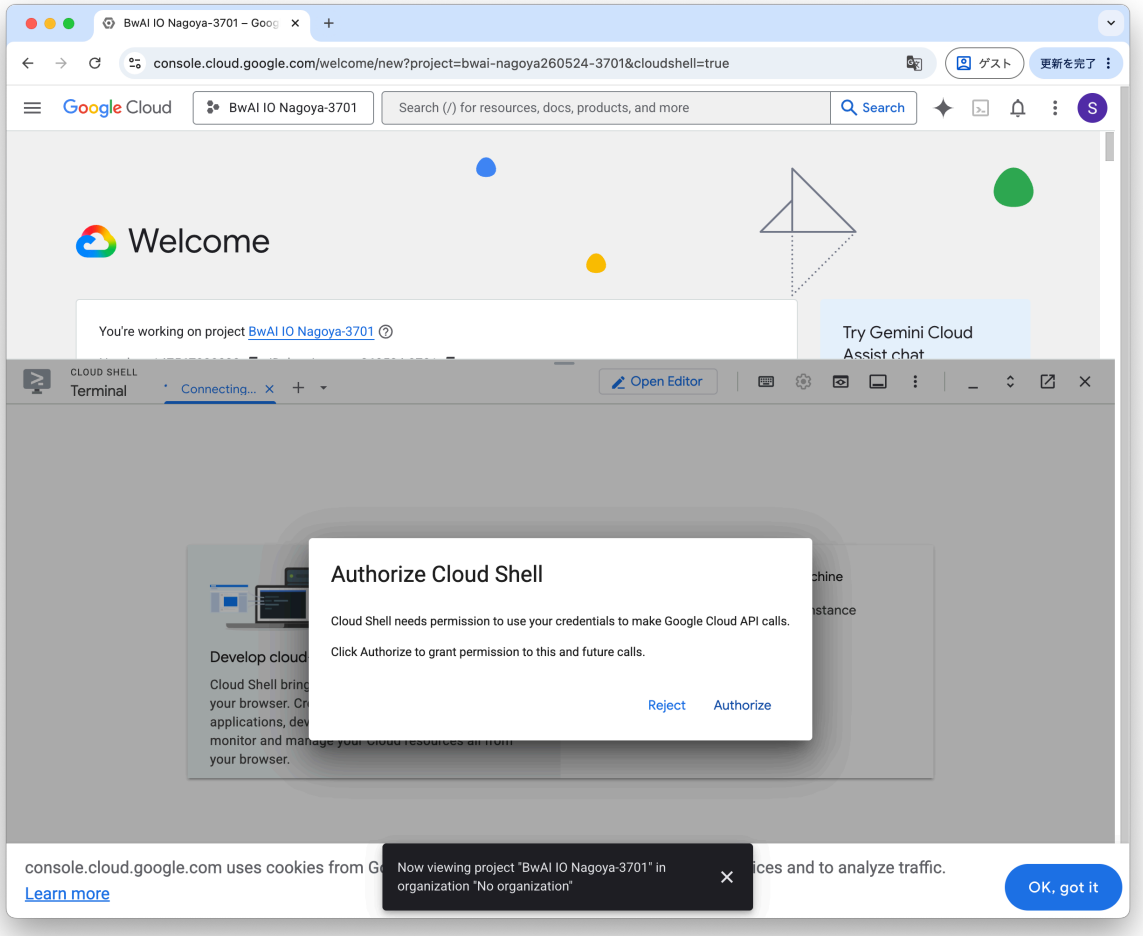
B Cloud Shell を起動

右上の **Cloud Shell アイコン** をクリック。ダイアログが表示されたら **「Continue」** を押す。



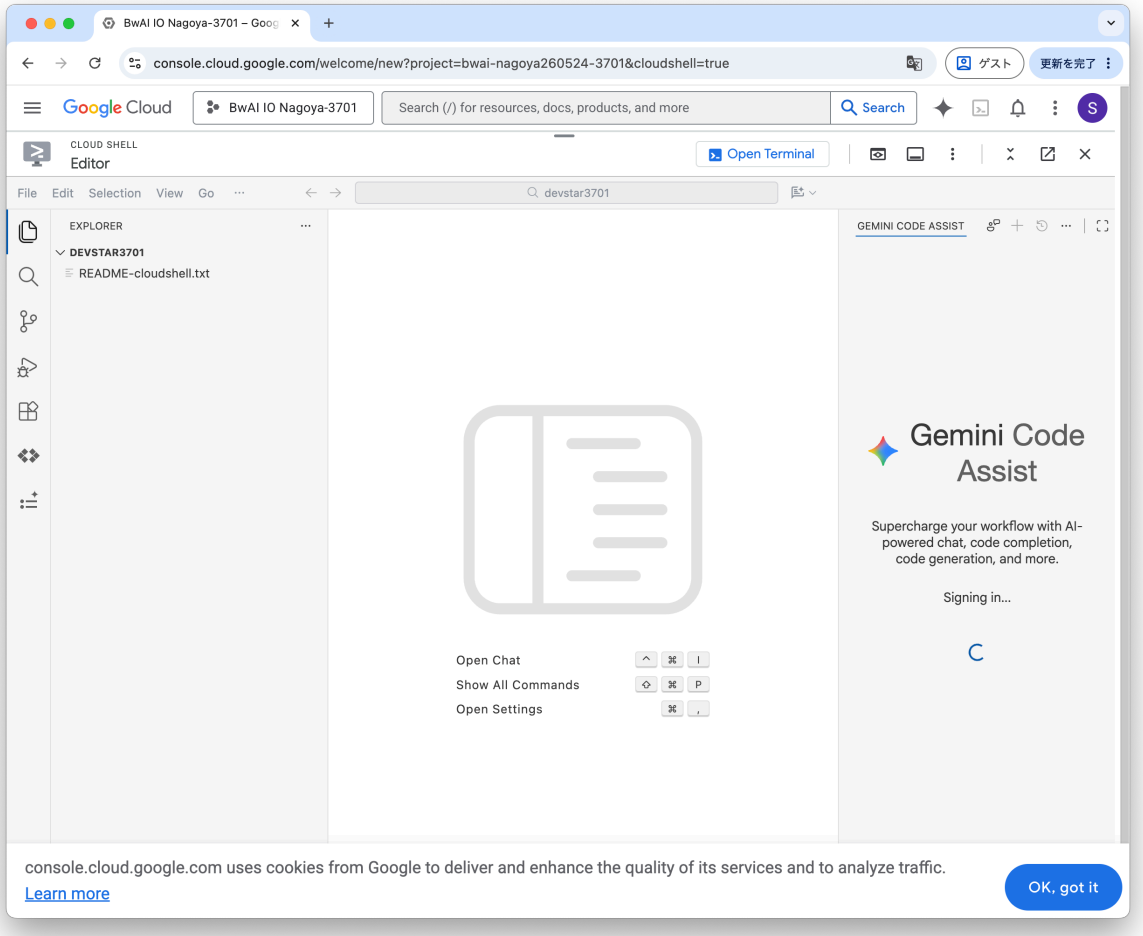
B Cloud Shell を認可

「**Authorize Cloud Shell**」ダイアログが表示されたら
「**Authorize**」をクリック。



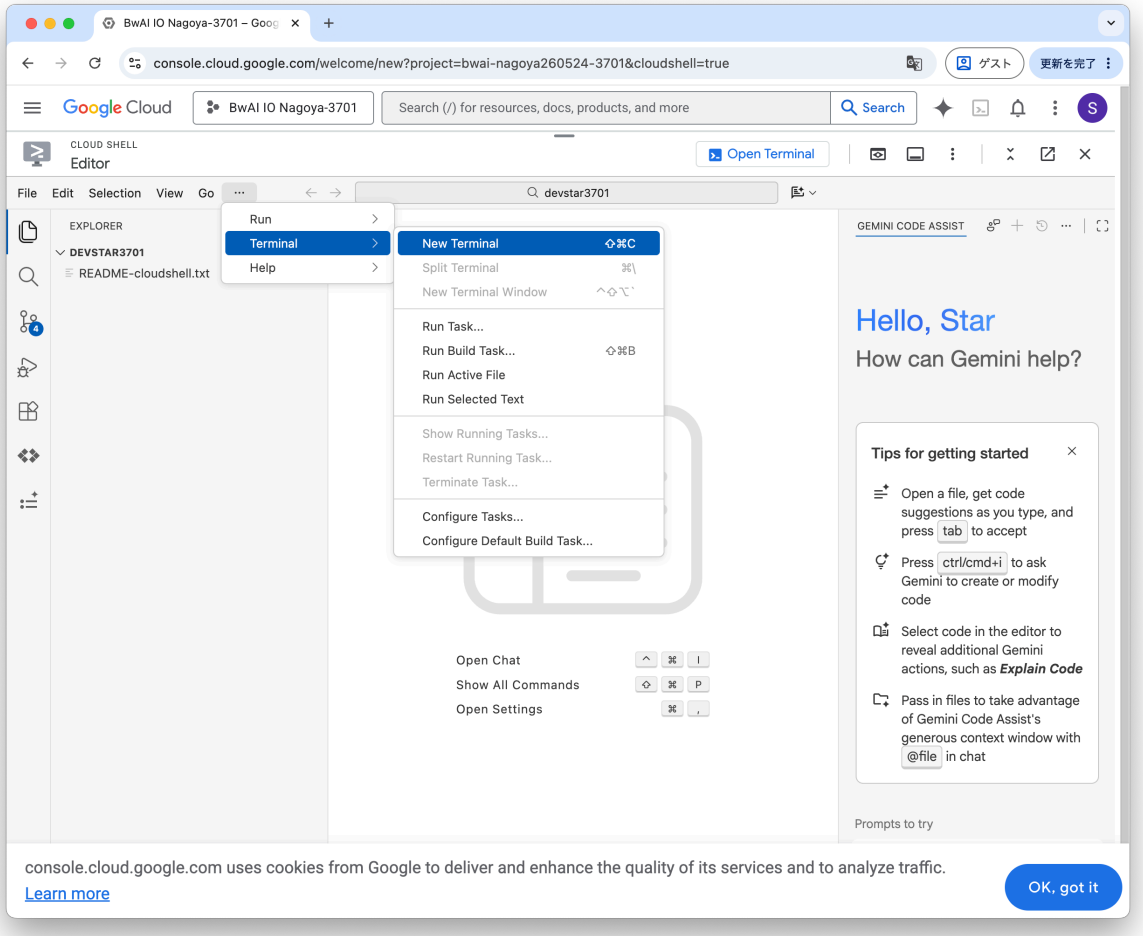
B Cloud Shell Editor を開く

ターミナルが起動したら、上部の **「Open Editor」** をクリックしてエディタに切り替え。



B Editor 内でターミナルを開く

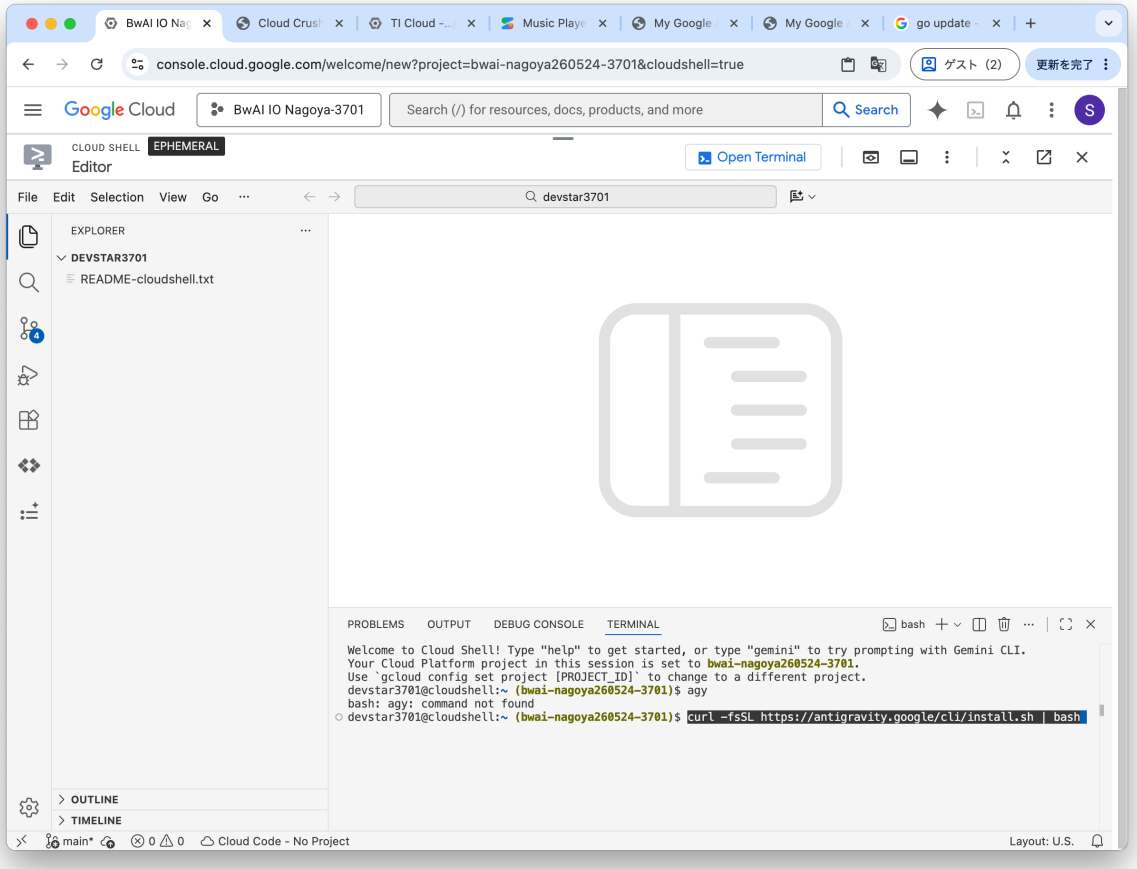
メニューから 「**Terminal**」→「**New Terminal**」 を選択して、エディタ内にターミナルを表示。



B Antigravity CLI：インストール

ターミナルが開いたら、以下のコマンドを実行して Antigravity CLI をインストールします。

```
curl -fsSL https://antigravity.google/cli/install.sh | bash
```

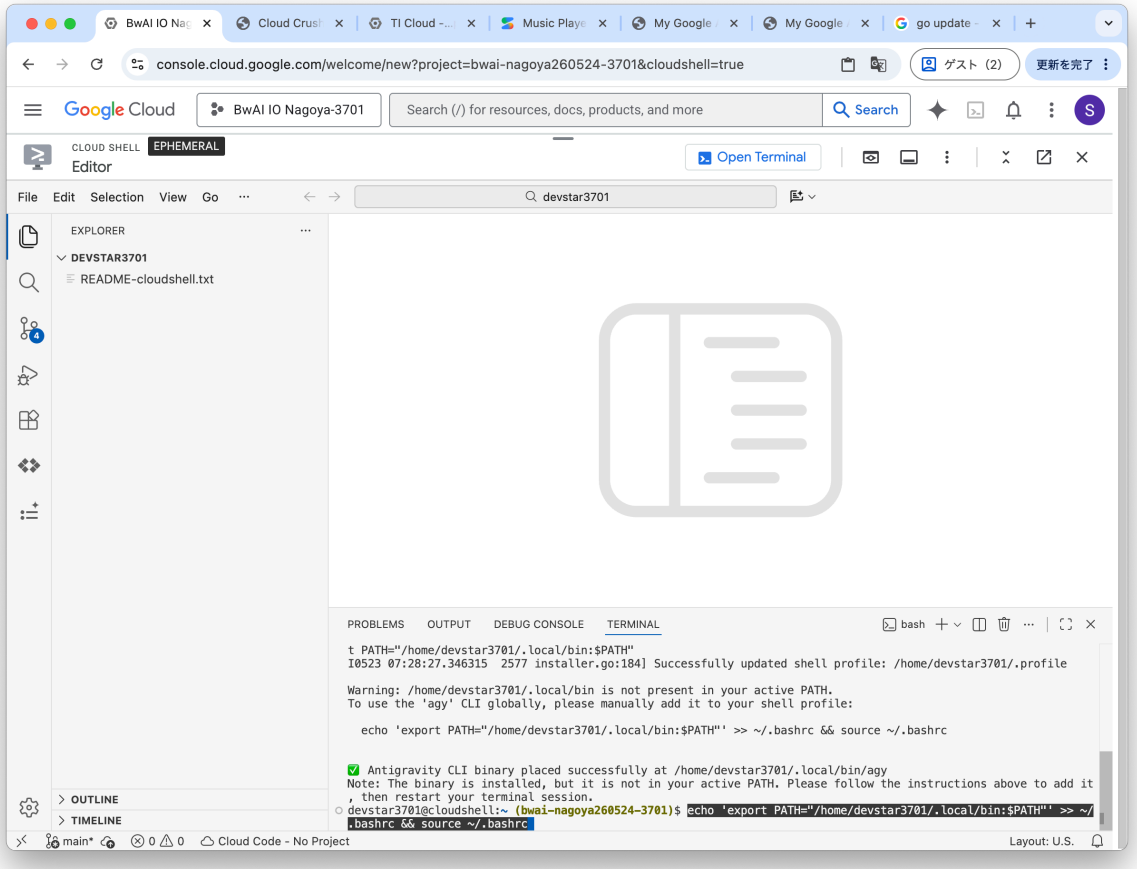


B Antigravity CLI：PATH を通す

インストーラによって `~/.local/bin` に実行ファイルが配置されます。
コマンドを実行できるようにするため、以下のコマンドを実行して PATH に追加します。

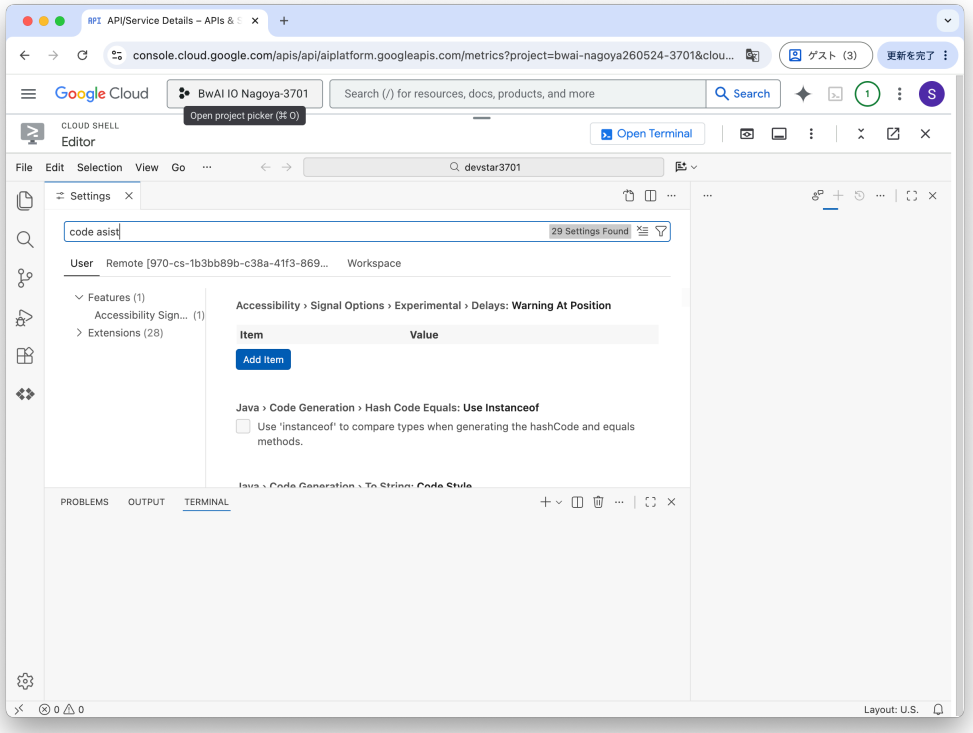
```
echo 'export PATH=$PATH:$HOME/.local/bin' >> ~/.bashrc
source ~/.bashrc
```

これで、どのディレクトリからでも `agy` コマンドが実行できるようになります。



B Antigravity CLI：起動とログイン

1. `agy` **コマンドの起動**:
   ターミナルで `agy` と入力して起動し、矢印キーで **「Use a Google Cloud project」** を選択。
2. **プロジェクト ID の指定**:
   プロジェクト ID（例： `bwai-nagoya260524-xxxx` **）** （プロジェクト一覧から確認可能）を入力。



B Antigravity CLI：注意点 (1/2)

Codelabs の記述と、本日の `agy`（Antigravity CLI）環境で異なる重要事項です。

1. **環境設定はスキップ**:
   - Codelab にある環境構築（API有効化や認証など）の一部手順は本スライドの手順で完了しているため、スキップしてください。
2. `gemini` **コマンド ➔** `agy` **コマンド**:
   - 手順書内の `gemini` コマンドは、すべて `agy` **コマンドに読み替えて** 実行してください。
3. **モデルステアリングの設定は不要**:
   - モデルやパラメータのステアリング（調整）に関する設定手順は不要です。

B Antigravity CLI：注意点 (2/2)

4. `/plan` **コマンドは不要**:
   - `agy` コマンドは**デフォルトで計画（plan）モードのような自律挙動**を行うため、`/plan` コマンドは存在せず、実行不要です。
5. **ブラウザエージェントのスキップ**:
   - `browser_agent` を使用するセクションは本環境では実行できません。その手順はスキップして進めてください。
6. **Vibe Coding 試行錯誤のヒント**:
   - 生成AIの出力するコードは、**実行する人やタイミングによって結果が異なります。**エラーが出た場合は、**「エラー文をそのまま agy に渡して修正させる」** 試行錯誤のラリーを繰り返しましょう！

B Antigravity CLI：補足

`/plan` コマンドは不要ですが、対話中にスラッシュ（`/`）を入力することで独自の機能（コマンド）を呼び出せるので、何ができるか見ておきましょう。

1. ⚡ **`/` を入力してコマンドを一覧表示**:
   - チャット入力欄で `/` と入力すると、利用可能なすべてのコマンドがポップアップ表示されます。
2. 💬 **`/btw`（By The Way）で横道に逸れた会話**:
   - タスクの進行履歴を汚さずに、ちょっとした雑談や質問、横道に逸れた相談ができます。
3. ⚙️ **`/config` で設定パラメータの確認**:
   - エージェントの各種設定、動作パラメータ、連携APIなどをその場で眺めたり調整したりできます。

B Antigravity CLI：Webプレビュー

Cloud Shell 上で起動したWebアプリケーションは、ブラウザで安全に動作確認が可能です。

- ⚡ **Web Previewの起動**:
  1. アプリケーションを起動（通常 `8080` などのポート）。
  2. 画面右上にある **「ウェブでプレビュー (Web Preview)」** をクリック。
  3. **「ポート 8080 でプレビュー」** を選択すると、セキュアなプレビューが開きます。
  4. 画面上にWebページが表示されたら確認完了です！

Codelabs プロンプト翻訳 (1/8)

アセット（画像ファイル）のダウンロード

> *"Create a folder named "assets" and download the images background.png, gcp_sprites.png, gemini.png and logo.png, from this GitHub repository to the "assets" folder: https://github.com/GoogleCloudPlatform/devrel-demos/tree/main/codelabs/gemini-cli/gemini-cli-match3-golang"*

【日本語訳】

```
「assets」という名前のフォルダを作成し、以下のGitHubリポジトリから background.png、gcp_sprites.png、gemini.png、logo.png の4つの画像ファイルをダウンロードして、「assets」フォルダに保存してください。

対象リポジトリURL：
https://github.com/GoogleCloudPlatform/devrel-demos/tree/main/codelabs/gemini-cli/gemini-cli-match3-golang
```

Codelabs プロンプト翻訳 (2/8)

🎮 マッチ3ゲームの基盤実装 (Ebitengine)

> *Build a Match-3 game called 'Cloud Crush' in Go using Ebitengine v2. The entire game screen should have background.png as background. The play area should be an 8x8 grid with white background. On the right side of the play area include a side panel with UI elements like player score and how to play instructions. The side panel should have a solid background colour to help with readability of the UI.*
> *Use standard GCP product logos (e.g. Compute Engine, Cloud Storage, BigQuery, etc.) as icons. These icons are provided in the gcp_sprites.png file.*
> *The icons are saved as 64x64 sprites but scale them as necessary based on the screen resolution. Implement swapping, clearing 3+ gems, and gravity.*
> *Use ebitengine native font rendering (size 48 for titles and size 24 for normal text) for all text and not the debug print.*
> *The font should be monospaced (golang.org/x/image/font/gofont/gomono). Keep the UI tidy and harmonic, e.g. centered text should always be adjusted based on text length, not just guess based on estimates.*

【日本語訳】

Ebitengine v2を使用して、Go言語で「Cloud Crush」という名前のマッチ3ゲームを作成してください。

ゲーム画面全体の背景には background.png を使用し、プレイエリアは白背景の8x8グリッドにしてください。プレイエリアの右側には、スコアや遊び方の説明を表示するサイドパネルを配置し、文字を読みやすくするため背景は単色（ベタ塗り）にしてください。

アイコンには標準的なGCPプロダクトのロゴを使用します。画像は gcp_sprites.png（64x64スプライト）に用意されているので、解像度に合わせて適宜拡大縮小してください。アイテムの入れ替え（スワップ）、3つ以上揃った時の消去、重力落下処理を実装してください。

デバッグプリントは使用せず、Ebitengineネイティブのフォントレンダリング（タイトル: 48、通常文: 24）で等幅フォント（golang.org/x/image/font/gofont/gomono）を用いて描画してください。UIは整然と調和のとれたレイアウトにし、中央揃えのテキストは実際の長さから正確に位置を調整してください。

B Codelabs プロンプト翻訳 (3/8)

⏱️ タイマーとスコアシステムの追加

> *Add a 60-seconds countdown timer and an in-memory high-score tracker to enhance the arcade game experience.*
> *Combos should give a score bonus of 10% per combo number.*

【日本語訳】

アーケードゲームとしての体験を向上させるため、60秒のカウントダウンタイマーとインメモリのハイスコア記録機能を追加してください。
また、コンボ発生時には、コンボ数につき10%のスコアボーナスを付与してください。

Codelabs プロンプト翻訳 (4/8)

⌨️ キーボード操作とアクセシビリティの追加

> *Update the implementation to include: 'Q' to quit, 'F' for full-screen and 'A' for Accessibility Mode: swap GCP logos for high-contrast coloured blocks.*
> *Also enable Arrow Keys to move the selection cursor and Space to select the gem to swap (space is pressed once to select, then arrow key immediately makes the move - no need to press space again).*
> *Use a golden square (4 px border, transparent fill) with a simple animation to highlight where the cursor is at the moment.*

【日本語訳】

実装を更新し、以下の機能を追加してください。

Qキー：ゲームを終了する ｜ Fキー：フルスクリーン表示の切り替え
Aキー：アクセシビリティモードの切り替え（GCPのロゴをハイコントラストな色のブロックに置き換える）

また、矢印キーで選択カーソルを移動させ、スペースキーで入れ替えるアイテム（ジェム）を選択できるようにしてください（スペースキーを1回押して選択状態にした後、矢印キーを押すだけで即座に入れ替えが実行され、再度スペースキーを押す必要はありません）。

現在のカーソル位置を強調表示するために、シンプルなアニメーション付きの金色の四角形（境界線4px、塗りつぶしは透明）を使用してください。

Codelabs プロンプト翻訳 (5/8)

🌐 WASMコンパイルとWebサーバーの実装

> *We need to enable this game to run on a web browser. Compile the game to WASM and create a Go web server to serve the compiled WASM and the assets.*

【日本語訳】

このゲームをWebブラウザ上で実行できるようにしてください。ゲームをWASMにコンパイルし、コンパイルされたWASMファイルとアセットを配信するためのGo言語のWebサーバーを作成してください。

Codelabs プロンプト翻訳 (6/8)

🏆 タイトル画面とリーダーボードの追加

> *Create a title screen that displays the game logo (logo.png) over the cloud background.*
> *The logo should be centered and occupy no more than 75% of the screen area.*
> *The title screen should display "Press ENTER to play" (black/bold) right below the logo, with every letter animated in a slow wavy (cosine) pattern.*
> *Once the player presses ENTER, it should be prompted to add their name, which will then be recorded to populate the leaderboard at the end of the round.*
> *Once the game is over, play the animated leaderboard with the top 10 highest scores.*
> *The animated leaderboard should render entries one by one up to 10 entries, using a fade in effect just like old school arcade games. The leaderboard should fade out to the title screen after 15 seconds.*
> *Please note that name entry should be processed independently of the play state key handlers (e.g. pressing A during name entry should not toggle accessibility mode).*

【日本語訳】

雲の背景の上にロゴ（logo.png）を表示するタイトル画面（中央配置、画面領域の75%以内）を作成してください。

ロゴのすぐ下に「Press ENTER to play」（黒色・太字）を表示し、各文字がゆっくりコサイン波で波打つアニメーションを追加してください。プレイヤーがEnterを押したら名前入力を促し、ラウンド終了時のランキングに反映されるよう記録してください。

ゲームオーバー時は、上位10件のハイスコアを昔のアーケードゲームのようにフェードインで1件ずつ順番に表示するアニメーション付きリーダーボードを表示してください。リーダーボードは15秒後にフェードアウトしタイトル画面に戻るようにします。

名前入力処理は、プレイ中のキー操作（例：名前入力中のAキーでアクセシビリティモードを切り替えない）とは完全に独立して処理してください。

Codelabs プロンプト翻訳 (7/8)

Google Cloud Runへのデプロイ

> *Use the gcloud CLI to provision and deploy the Go web server and its assets to Google Cloud Run. Provide the live URL when complete.*

【日本語訳】

```
gcloud CLIを使用して、GoのWebサーバーと関連アセットをGoogle Cloud Runにプロビジョニングおよびデプロイしてください。完了したら、アクセス可能な公開URL（ライブURL）を提示してください。
```

Codelabs プロンプト翻訳 (8/8)

📱 モバイル対応とQRコードの生成

> *Now enable this game to run on mobile devices. You need to update the input system to handle "taps" as well as key presses and clicks. Since mobile devices most likely won't have a keyboard, add a button to generate a random name and a confirmation button to the name entry.*
> *Also generate a QR code for the CloudRun link and display it on the screen so that people can scan it to access the mobile version and compete against their friends for the high score.*

【日本語訳】

次に、このゲームをモバイル端末でも遊べるようにしてください。キーボード入力やマウスクリックに加えて、「タップ」操作も処理できるように入力システムを更新する必要があります。

モバイル端末では物理キーボードが使えないことが多いため、名前入力画面に「ランダムな名前を生成するボタン」と「確定ボタン」を追加してください。

さらに、Cloud RunのURLのQRコードを生成して画面に表示し、他の人がそれをスキャンしてモバイル版にアクセスし、友達同士でハイスコアを競えるようにしてください。

Part 3: 自由制作（30分チャレンジ）

🎨 自由制作のアイデア集

ハンズオンで基本をマスターした後は、AI開発ツールを活用してオリジナルアプリを作ってみましょう！

1. 💡 **AIアイデアジェネレーター** (AI Studio向き)
   - 面白いサービスやアプリの企画をAIが瞬時にブレストしてくれる対話型Webツール。
2. 📋 **タスク管理ToDoアプリ** (Antigravity CLI向き)
   - ローカルのファイル操作や、ユーザーの入力に応じてタスクをスマートに整理するWebアプリ。
3. 🏯 **GDG名古屋 応援特設ファンサイト**
   - 会場「名古屋能楽堂」の伝統的な雰囲気を盛り込んだ、モダンな1ページ特設Webサイト。