

# Software Development Acceleration

The information in this presentation is classified:

---

## Google confidential & proprietary

---

⚠ This presentation is shared with you under NDA.

- Do **not** record or take screenshots of this presentation.
- Do **not** share or otherwise distribute the information in this presentation with anyone **inside** or **outside** of your organization.



Thank you!

# Topics

01 Google and Gemini's Approach to Developer Acceleration

02 Where can you use Gemini Code Assist?

03 Enabling Gemini Code Assist

04 Example IDE Setup: VS Code

05 How to Interact with Gemini Code Assist

06 Common dev-related tasks for Gemini or Gemini Code Assist

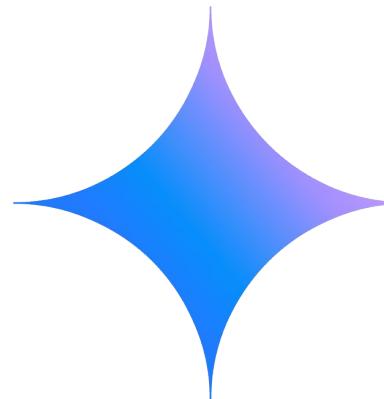
07 Prompt Guidance





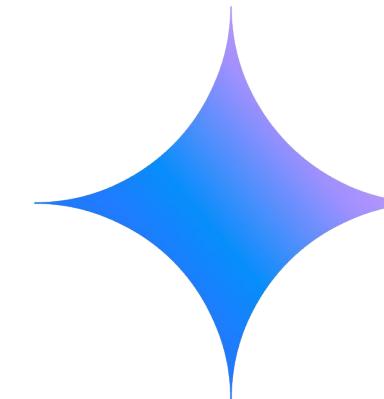
Gemini Code Assist helps development teams build, deploy, and operate applications faster and more efficiently.

# What makes Google Cloud AI the right choice?



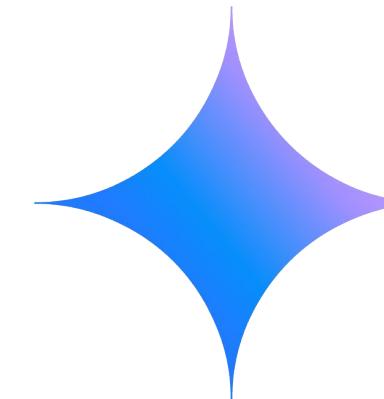
## A deep history of research and innovation

Google's generative AI builds on **over 20 years of internal R&D** across search, databases, hardware, and AI



## A deeply integrated, vertically differentiated portfolio

From super scalable AI infra, to world-class models from a variety of sources, to an intuitive AI development and deployment platform, to AI-powered assistants – you will find **the right AI for your skills and needs.**

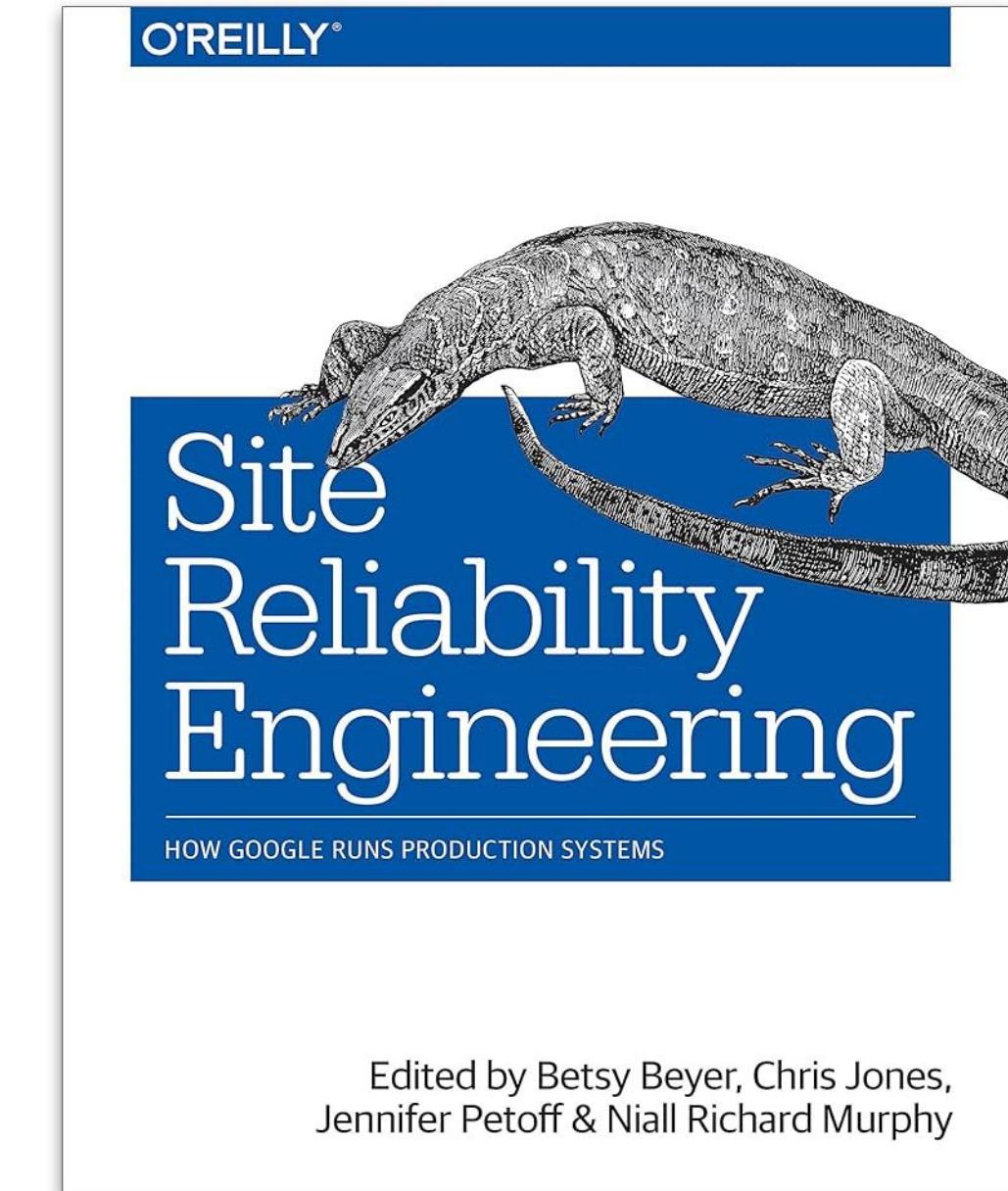
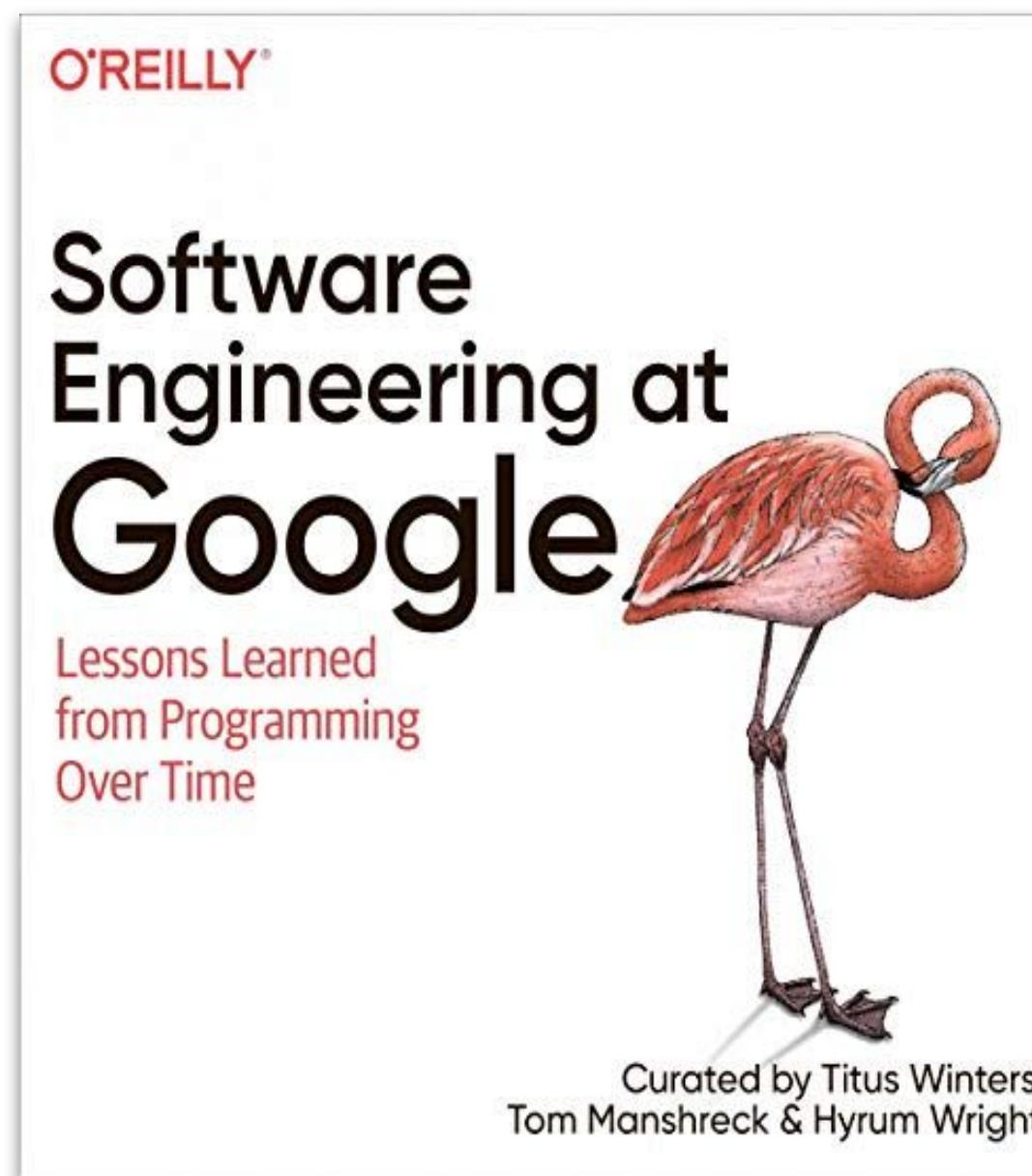


## Built on a foundation of enterprise readiness

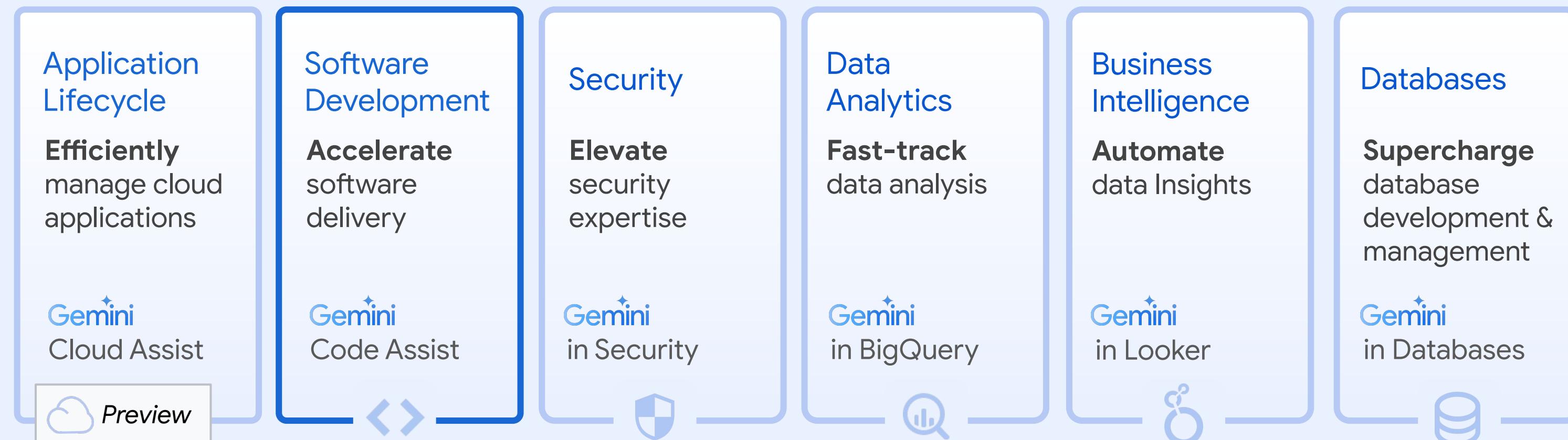
Go to production with the peace of mind provided by **data governance and indemnity, security and compliance, and safety and responsible AI.**

# Software delivery is a key part of Google's DNA.

## Gemini Code Assist continues the tradition of sharing internal learnings with customers.



# ★ Gemini for Google Cloud Portfolio



# Gemini Code Assist is your AI developer collaborator



Duet AI for Developers is now Gemini Code Assist



In-IDE Code completion, Chat and Smart actions



Powered by state-of-the-art Gemini models



Enterprise ready governance, controls, and security

Since December 2023 launch, 5,000+

**businesses** actively use Gemini Code Assist

```

  back.py  x
src > backend > back.py > ...
10 app = Flask(__name__)
11 app.config["MONGO_URI"] = 'mongodb://{}//guestbook'.format(os.environ.get('GUESTBOOK_DB_ADDR'))
12 mongo = PyMongo(app)
13
14 #Function to translate text from English to Spanish using the Cloud Translation API v3
15
16
17 @app.route('/messages', methods=['GET'])
18 def get_messages():
19     """ retrieve and return the list of messages on GET request """
20     field_mask = {'author':1, 'message':1, 'date':1, '_id':0}
21     msg_list = list(mongo.db.messages.find({}, field_mask).sort("_id", -1))
22     return jsonify(msg_list), 201
23
24 @app.route('/messages', methods=['POST'])
25 def add_message():
26     """ save a new message on POST request """

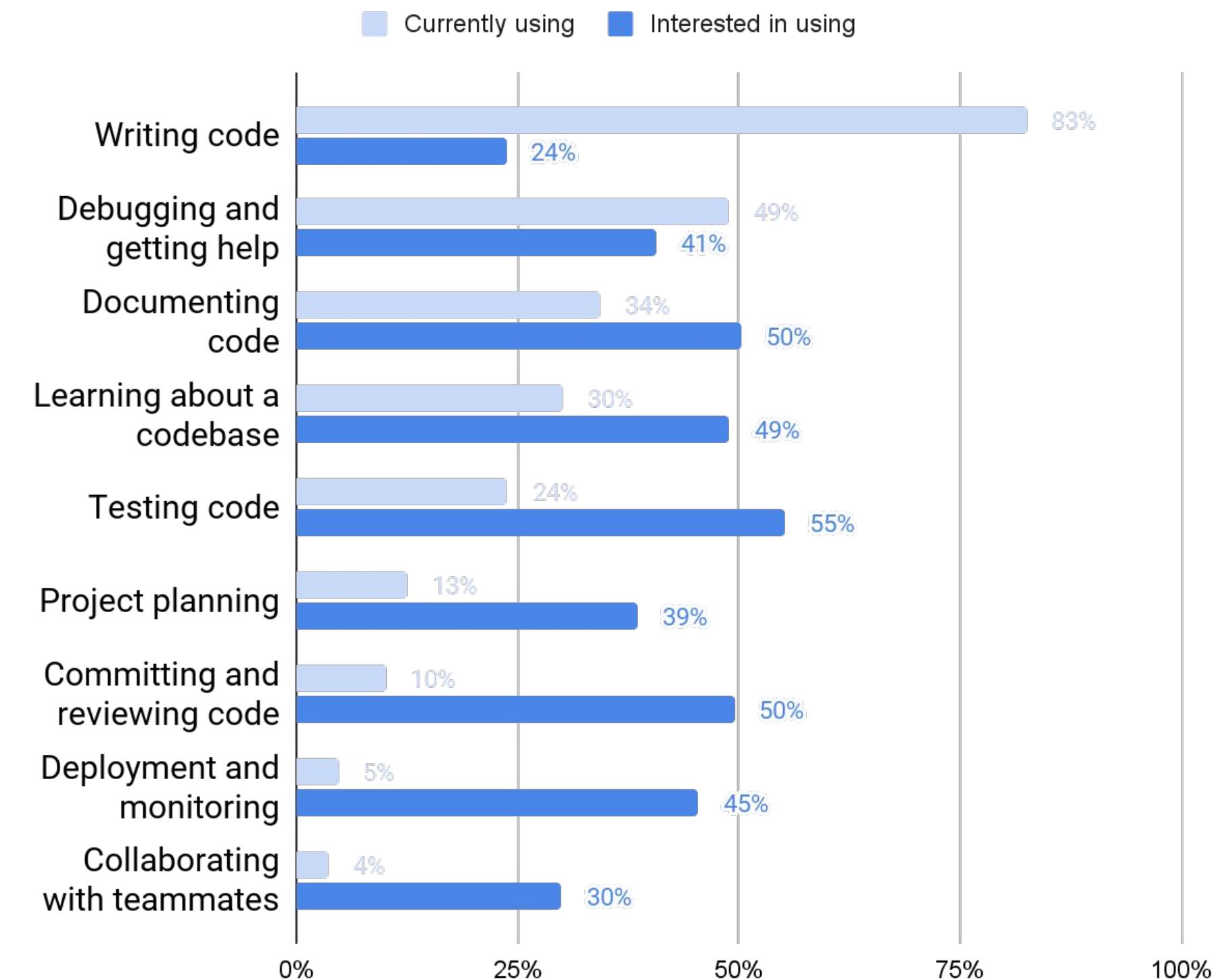
```

Cloud Code cloud-workstations-demo Spaces: 4 UTF-8 LF Python 3.10.7 64-bit Layout: U.S. Duet AI Feedback

# What do engineers want AI to do?

**When you look at what external engineers are currently using AI for, compared to what they are interested in using AI for, you notice new areas emerge.**

- Testing code
- Documenting code
- Committing and reviewing code
- Deploying and operating applications



39,042 Responses  
Source: [StackOverflow](#)

Google Cloud

# 40% faster completion time for typical dev tasks

**27%**

Less time to modify existing code  
(19:05 -> 14:58)

**56%**

Less time to Understand new code  
(4:19 -> 1:54)

**57%**

Less time to write new code  
(11:32 -> 5:00)

**53%**

Less time to write unit test  
(10:44 -> 5:02)

# Case study: Gemini Code Assist helped Wayfair increase productivity and developer satisfaction

+55%

Faster environment setup time

+48%

Higher unit test coverage

+60%

Developers report being able to focus on more satisfying work



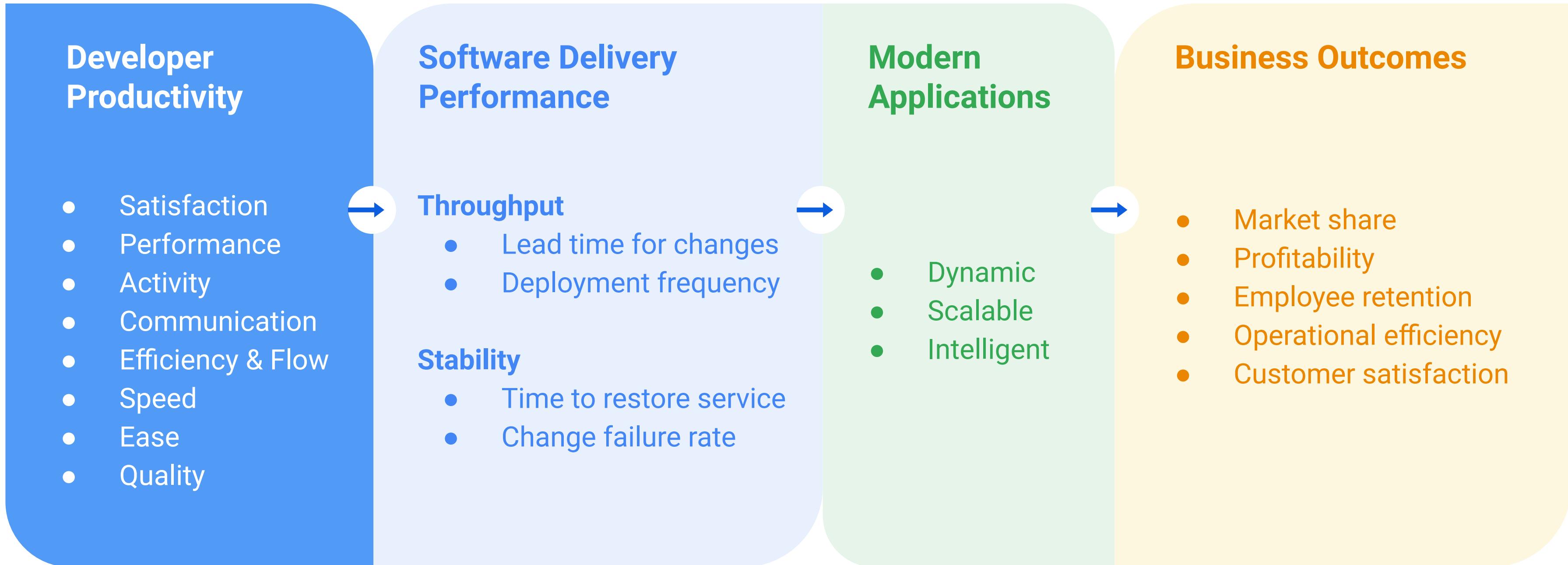


# TURING

# 33%

gain in developer productivity while using Gemini Code Assist in their development teams.

# Real Outcomes remain the Goal



# When AI helps improve developer productivity, it impacts the entire business

Generative AI in software engineering functions could increase **productivity by 20-45%**<sup>1</sup>

Code generation

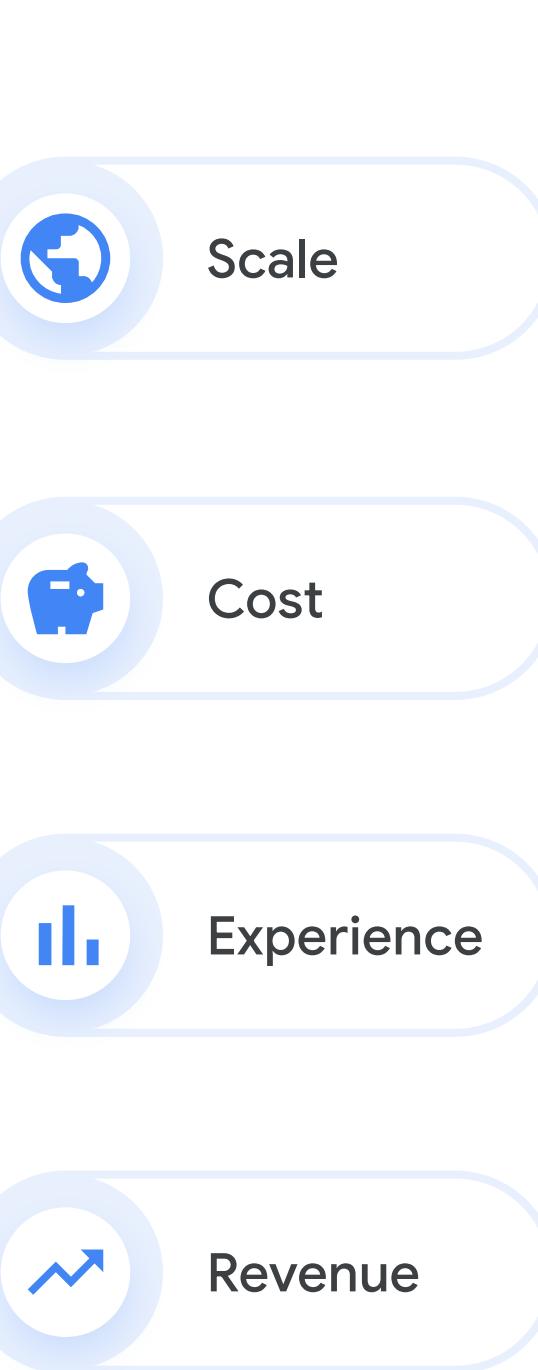
Code refactoring

Code completion

Code explanation

Code documentation

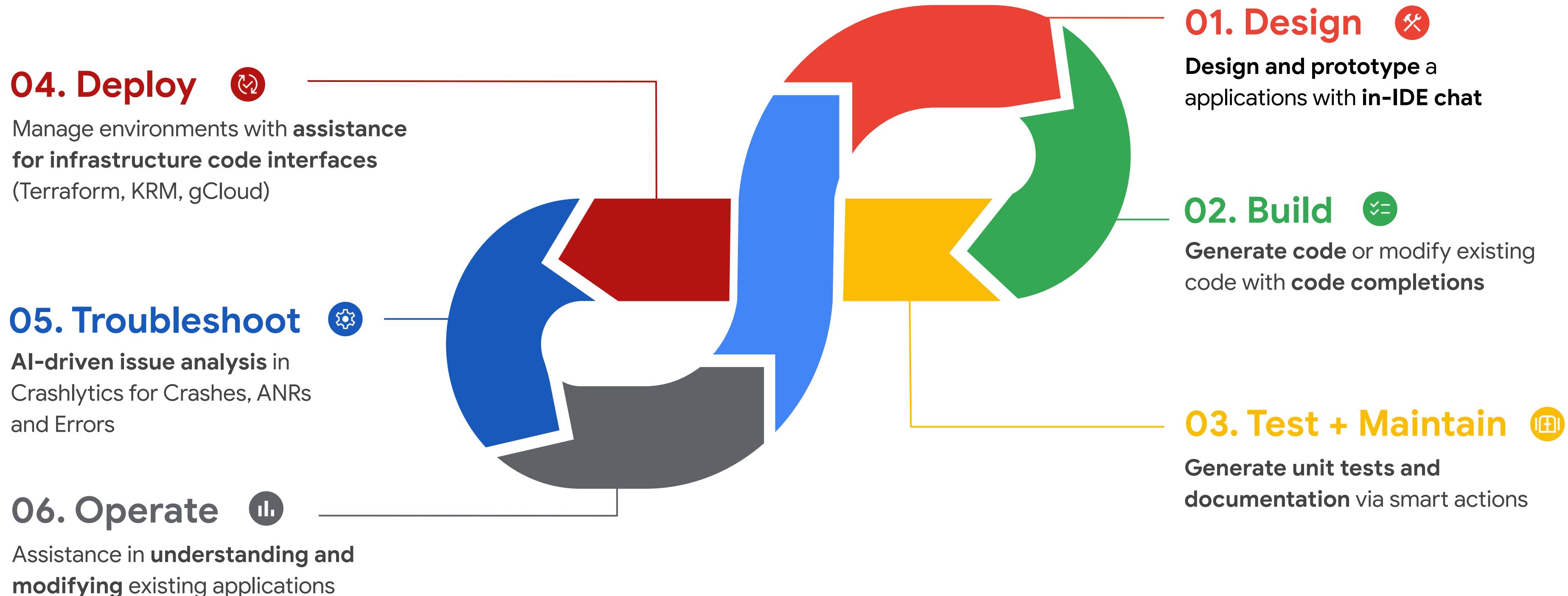
Code testing



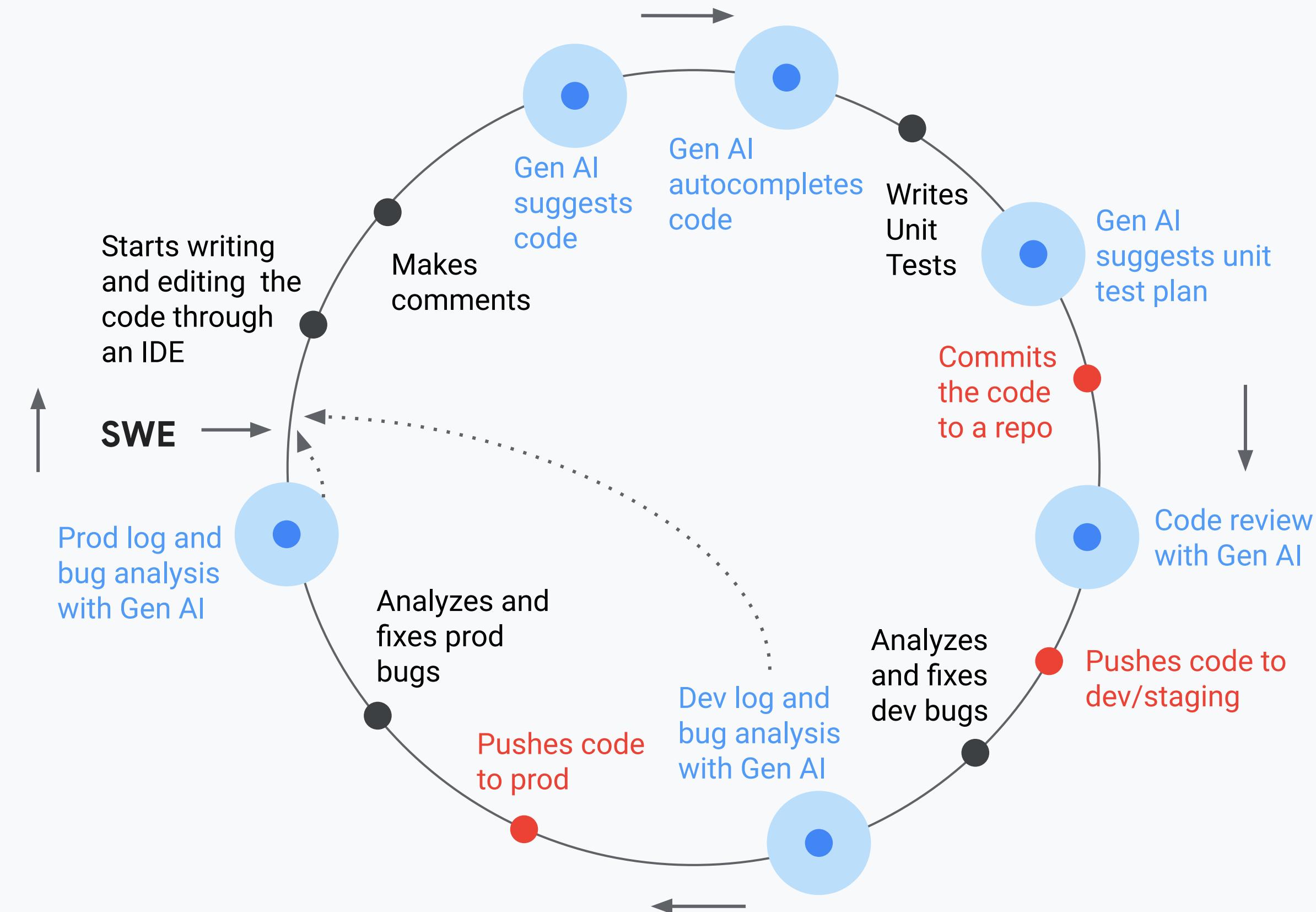
## Business Impact

- Faster time to market with higher quality products
- Grow business with less communication overhead
- Reduce costs to train and upskill developers
- Decrease technical debt
- Increase employee satisfaction & retention
- Reduce time spent on rote tasks
- Attract top engineering talent
- Accelerate product innovation
- Monetize new IP
- Increase brand value

# Gemini Code Assist improves each stage of the SDLC



# Workflow with Gemini Coding Assistant



# Topics

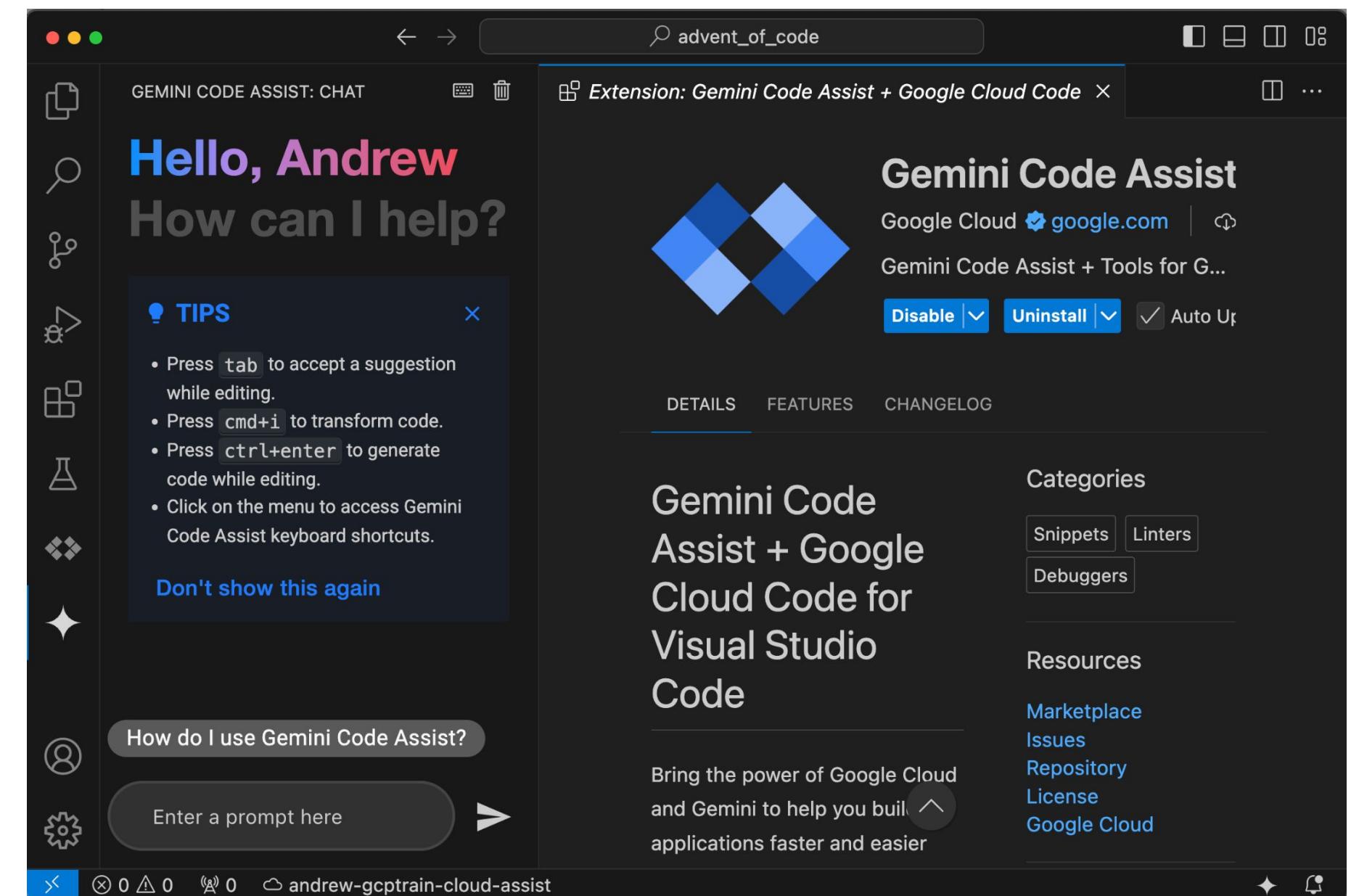
- 01 Google and Gemini's Approach to Developer Acceleration
- 02 [Where can you use Gemini Code Assist?](#)
- 03 Enabling Gemini Code Assist
- 04 Example IDE Setup: VS Code
- 05 How to Interact with Gemini Code Assist
- 06 Common dev-related tasks for Gemini or Gemini Code Assist
- 07 Prompt Guidance



# Supported IDEs

## Your Local IDE

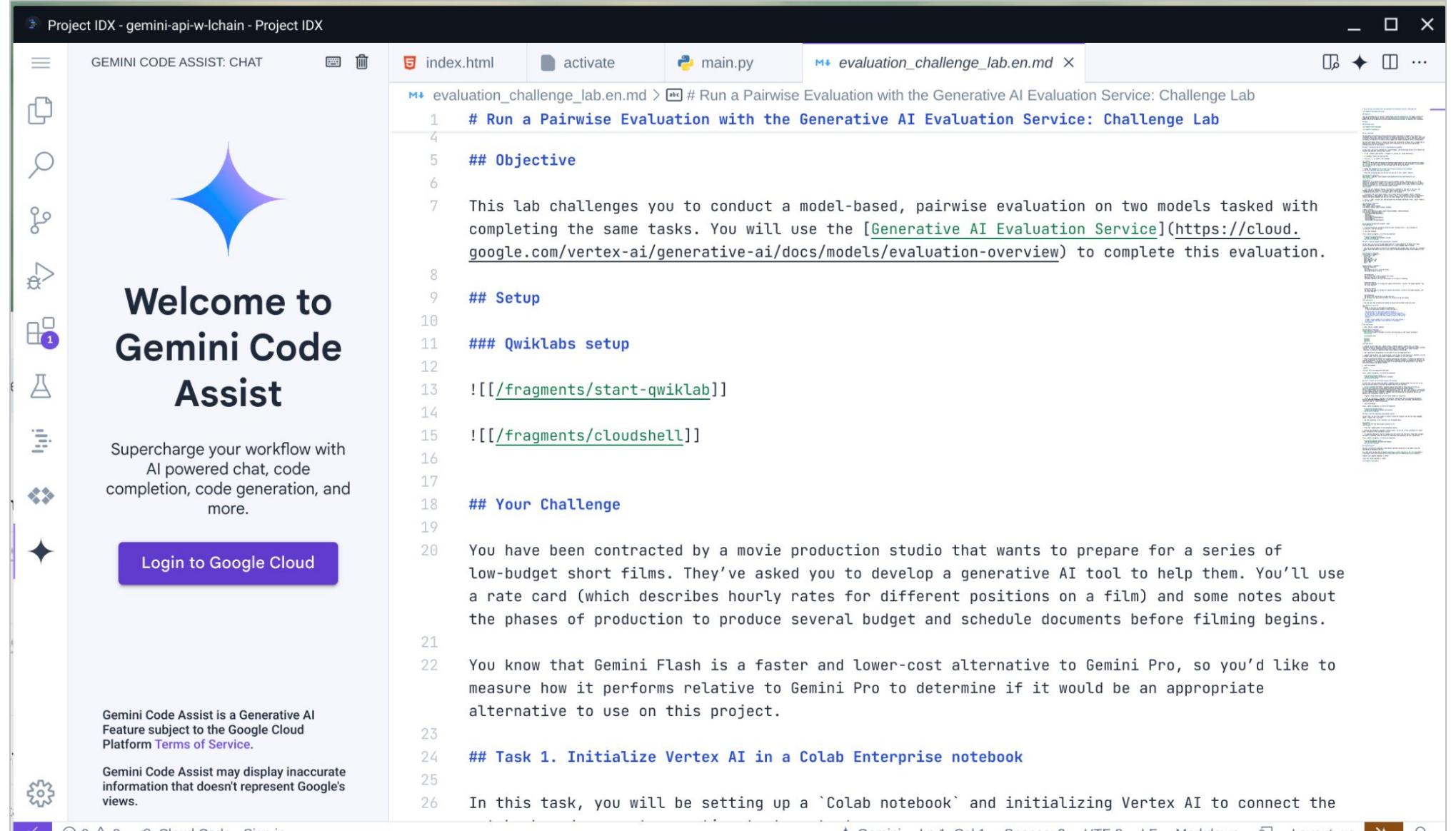
- VS Code
- Supported JetBrains IDEs:
  - IntelliJ IDEA  
(Ultimate and Community Editions)
  - PyCharm  
(Professional and Community Editions)
  - WebStorm
  - GoLand
  - There is also limited support for Rider



# Supported IDEs

## Google-Managed Online IDEs

- [Project IDX](#)
- [Cloud Shell Editor](#)
- [Cloud Workstations](#)



The screenshot shows the Project IDX interface. On the left, there's a sidebar with various icons for file operations like create, search, and refresh. The main area features a large blue starburst icon and the text "Welcome to Gemini Code Assist". Below this, it says "Supercharge your workflow with AI powered chat, code completion, code generation, and more." A purple "Login to Google Cloud" button is visible. At the bottom of this section, there are two small notes: one about Gemini Code Assist being a Generative AI Feature subject to Google Cloud Platform Terms of Service, and another about it displaying inaccurate information. The right side of the screen is a code editor window titled "Project IDX - gemini-api-w-lchain - Project IDX". It shows a file named "evaluation\_challenge\_lab.en.md" with the following content:

```

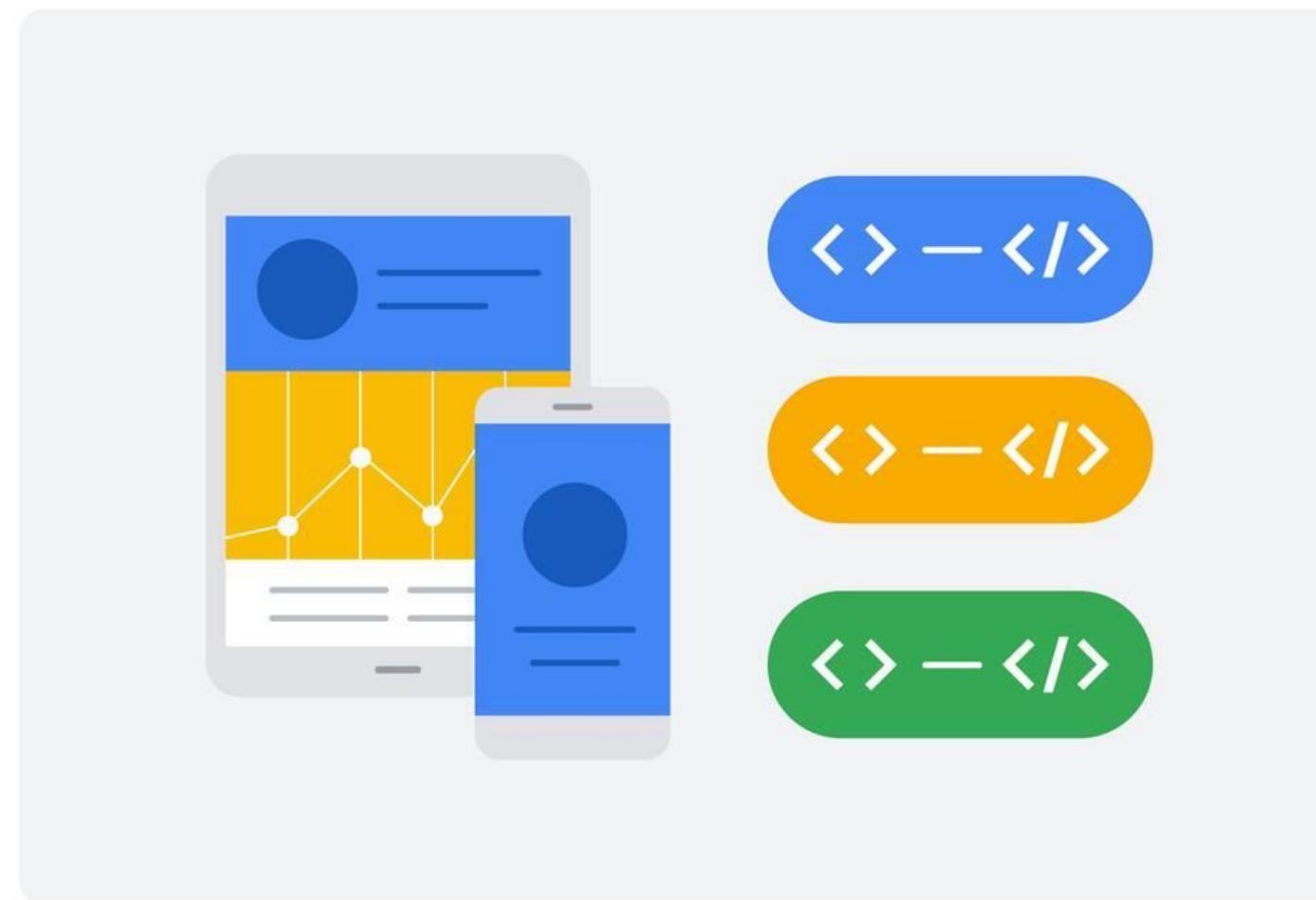
1 # Run a Pairwise Evaluation with the Generative AI Evaluation Service: Challenge Lab
2
3 ## Objective
4
5 This lab challenges you to conduct a model-based, pairwise evaluation on two models tasked with
6 completing the same tasks. You will use the [Generative AI Evaluation Service](https://cloud.
7 google.com/vertex-ai/generative-ai/docs/models/evaluation-overview) to complete this evaluation.
8
9 ## Setup
10
11 ### QwikLabs setup
12
13 ![[/fragments/start-qwiklab]]
14
15 ![[/fragments/cloudshell]]
16
17
18 ## Your Challenge
19
20 You have been contracted by a movie production studio that wants to prepare for a series of
21 low-budget short films. They've asked you to develop a generative AI tool to help them. You'll use
22 a rate card (which describes hourly rates for different positions on a film) and some notes about
23 the phases of production to produce several budget and schedule documents before filming begins.
24
25 ## Task 1. Initialize Vertex AI in a Colab Enterprise notebook
26 In this task, you will be setting up a `Colab notebook` and initializing Vertex AI to connect the

```

At the bottom of the code editor, there are status indicators: "Cloud Code - Sign in", "Gemini Ln 1, Col 1 Spaces: 2 UTF-8 LF Markdown", and "Layout: us".

# Supported Languages

- Bash
- C
- C++
- C#
- Dart
- Go
- GoogleSQL
- Java
- JavaScript
- Kotlin
- Lua
- MatLab
- MatLab
- PHP
- Python
- R
- Ruby
- Rust
- Scala
- SQL
- Swift
- TypeScript
- YAML



# Gemini is now also part of BigQuery Studio

```
-- How many english edits were done in the wikipedia dataset
SELECT
  CAST(count(TITLE) as BIGNUMERIC)
FROM
  `bigquery-public-data.wikipedia_pageviews.pageviews_2016`
WHERE
  LANGUAGE = 'en'
  AND LENGTH(TITLE) > 10
  AND TITLE LIKE '%edit%'
```

```
-- Create a dataset called orders with
-- a table customers that has the fields Id, Name, and Address
CREATE SCHEMA `orders`;
CREATE TABLE `orders.customers` (
  Id INT64 NOT NULL,
  Name STRING,
  Address STRING
);
```

The screenshot shows the BigQuery Studio interface with two main sections. On the left, there is a code editor window containing SQL and Python code snippets. On the right, there is a sidebar titled "Duet AI" which lists various AI features available for both SQL and Python notebooks.

**Duet AI in SQL query**

- ✓ Auto-completion
- ✓ Auto-generation
- ✓ Explanation

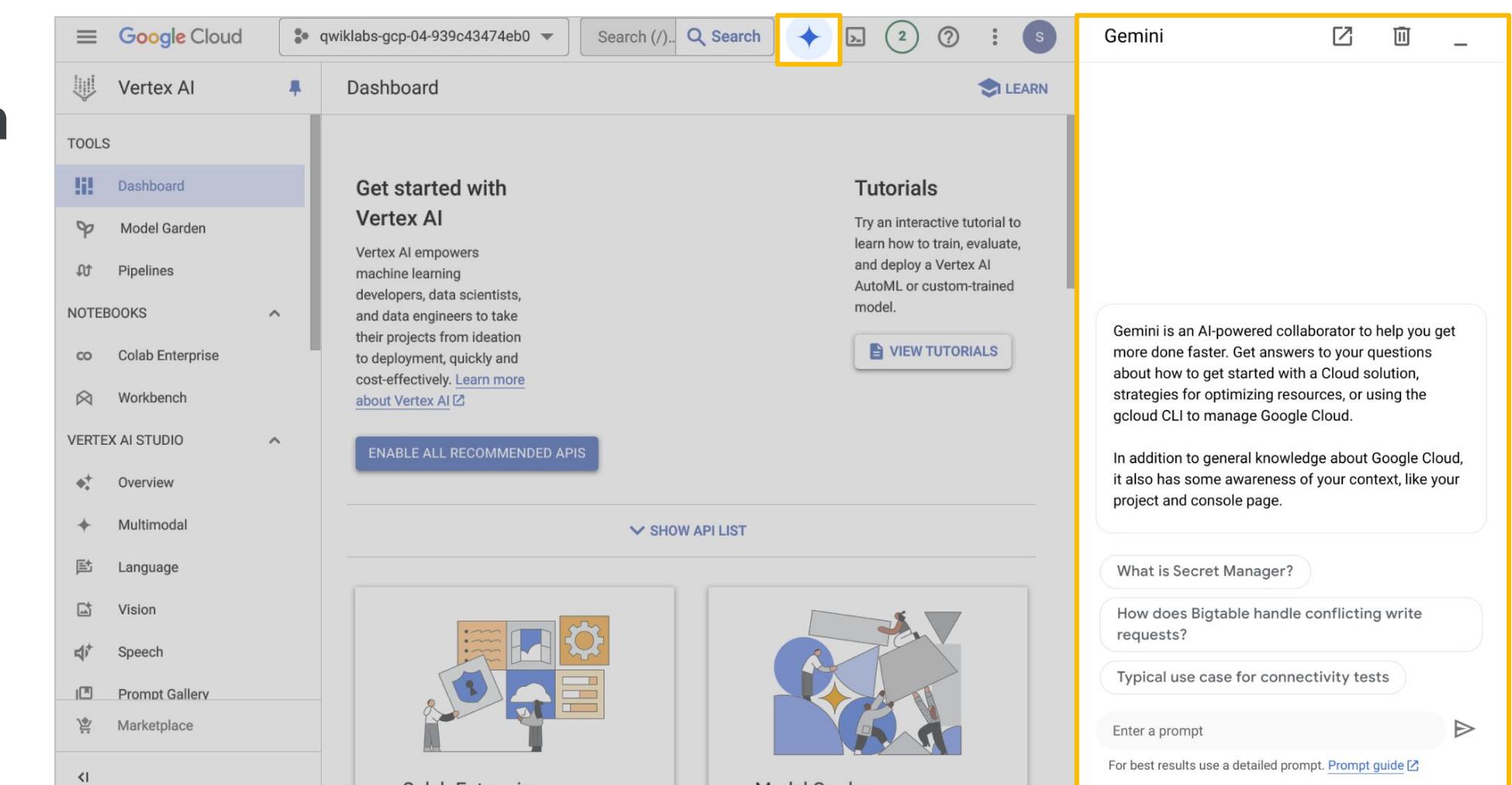
**Duet AI in Python notebook**

- Code completion
- ✓ Code generation  
Changes will not apply to open notebook tabs.

[Learn more](#) [Send feedback](#)

# Other Gemini-assisted coding/querying environments (that aren't technically Gemini Code Assist)

- BigQuery Studio
- Other Databases (via the **Gemini pane in the Cloud Console & [other locations](#)**)
  - AlloyDB for PostgreSQL
  - Cloud SQL for MySQL
  - Cloud SQL for PostgreSQL
- Colab Enterprise
- Keep your eye out for the Gemini icon:



# Topics

- 01 Google and Gemini's Approach to Developer Acceleration
- 02 Where can you use Gemini Code Assist?
- 03 [Enabling Gemini Code Assist](#)
- 04 Example IDE Setup: VS Code
- 05 How to Interact with Gemini Code Assist
- 06 Common dev-related tasks for Gemini or Gemini Code Assist
- 07 Prompt Guidance



# 1. Purchase a Gemini Code Assist subscription

Without a subscription, Gemini Code Assist is available at no cost for one user per billing account until Nov 8, 2024.

The licensing fee for a subscription is currently [\\$19 per user per month \(with a 12 month commitment\)](#).

1. Go to [Gemini for Google Cloud](#) in the Cloud Console
2. Select **Gemini Code Assist**
3. Select **Get Gemini Code Assist** and configure your subscription

## 2. Enable the Cloud AI Companion API in a Google Cloud project

1. Create or determine which project you will use to grant access to Gemini Code Assist
2. Enable the **Cloud AI Companion API**

### 3. Grant team members (or even better, a Google Group for team members) the needed roles

1. Grant your users the roles:
  - a. Cloud AI Companion User
  - b. Service Usage Consumer

# 4. Team members can then configure their environments to use this project

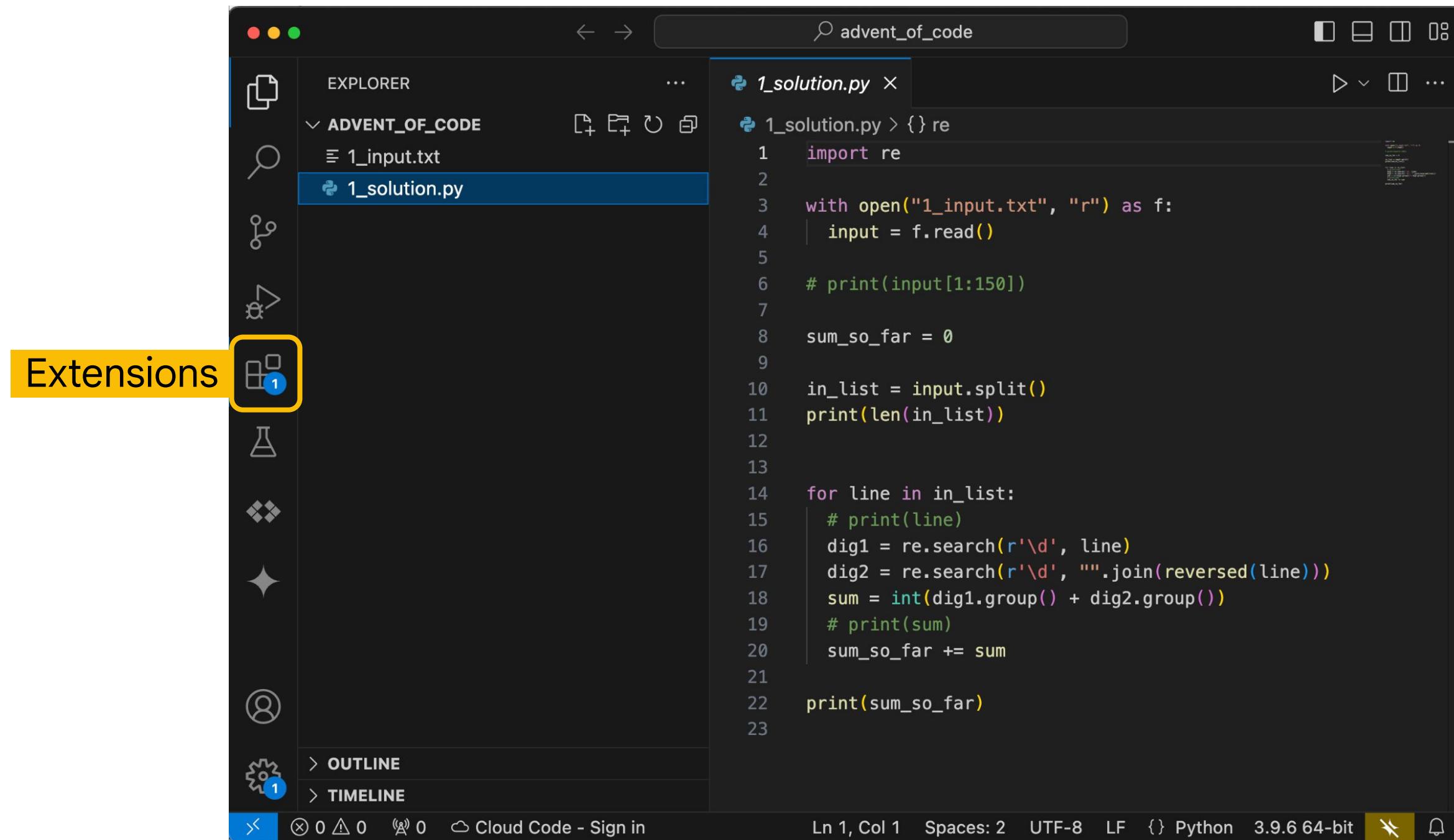
A demonstration of steps for configuring VS Code is in the next section.

# Topics

- 01 Google and Gemini's Approach to Developer Acceleration
- 02 Where can you use Gemini Code Assist?
- 03 Enabling Gemini Code Assist
- 04 Example IDE Setup: VS Code
- 05 How to Interact with Gemini Code Assist
- 06 Common dev-related tasks for Gemini or Gemini Code Assist
- 07 Prompt Guidance

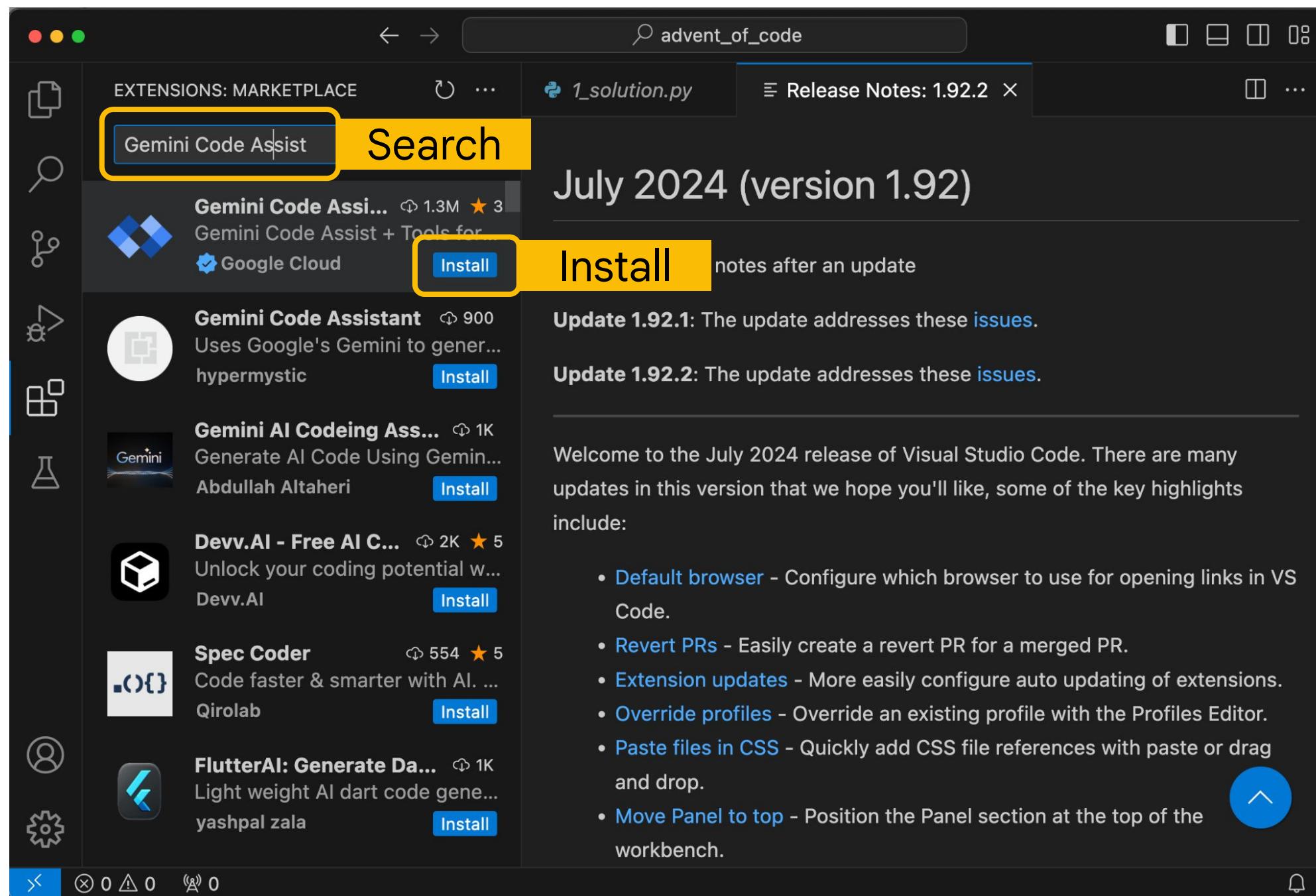


# Open VS Code. Select the Extensions panel.

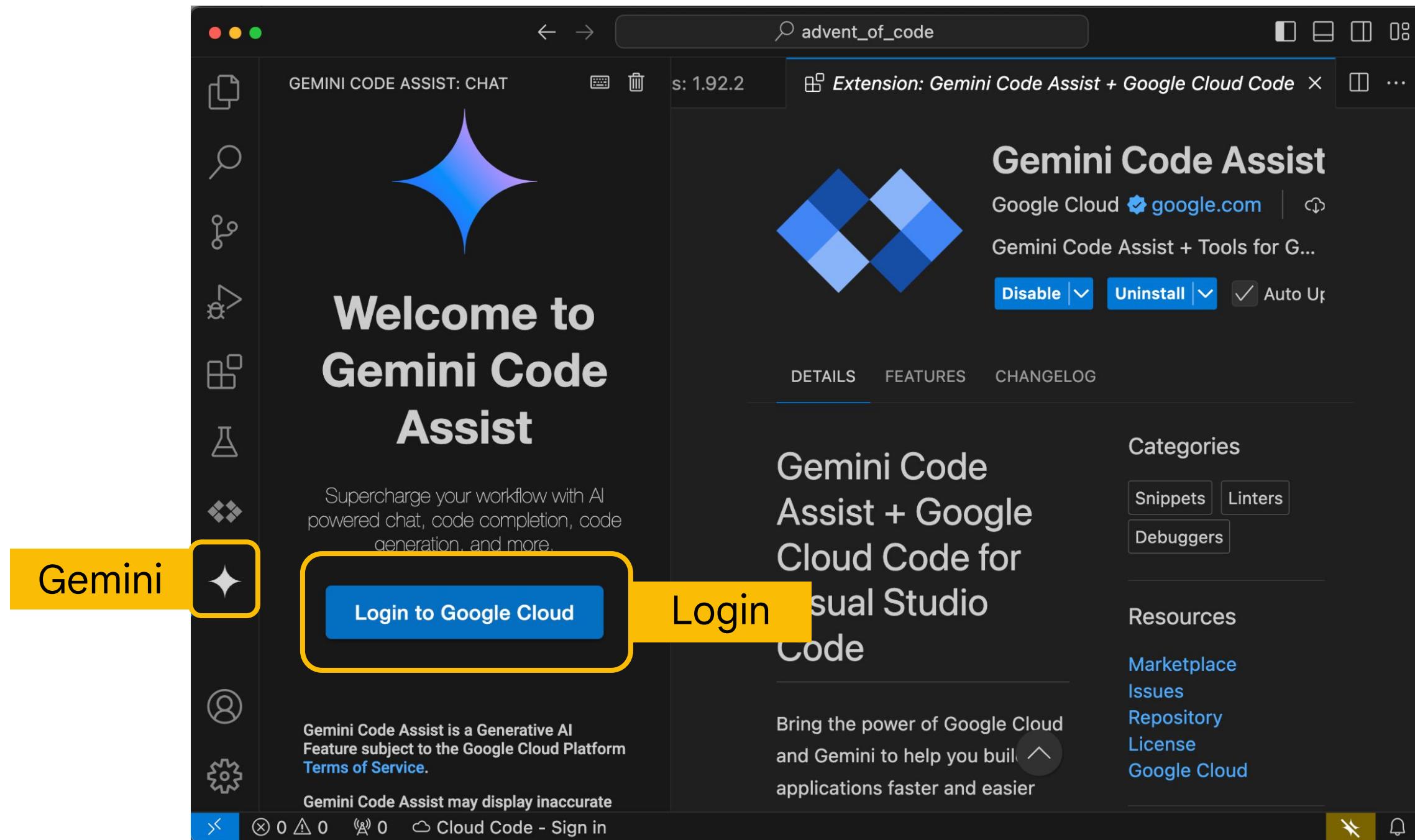


# In the Extensions panel, search for **Gemini Code Assist.**

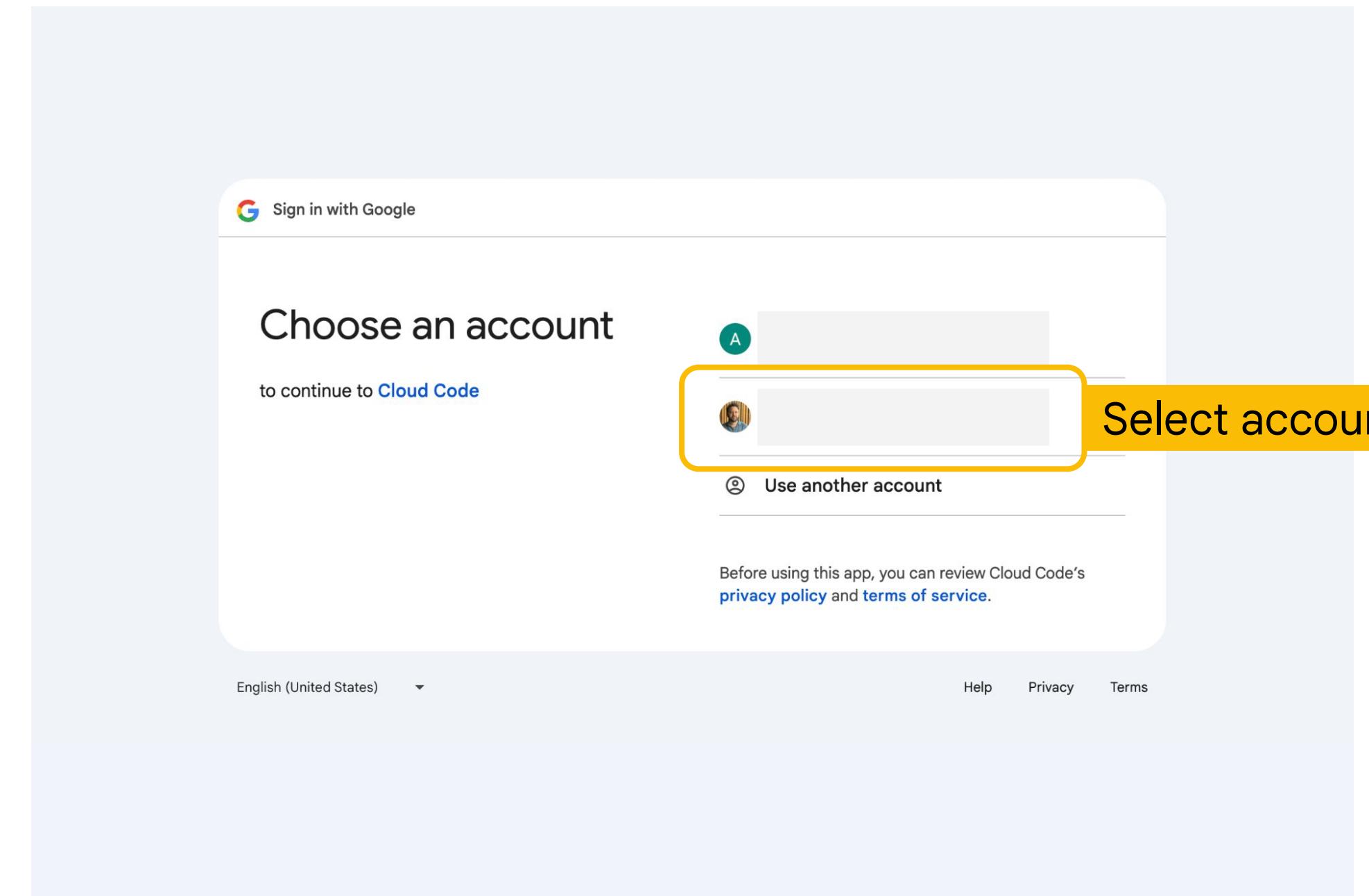
## Select **Install**.



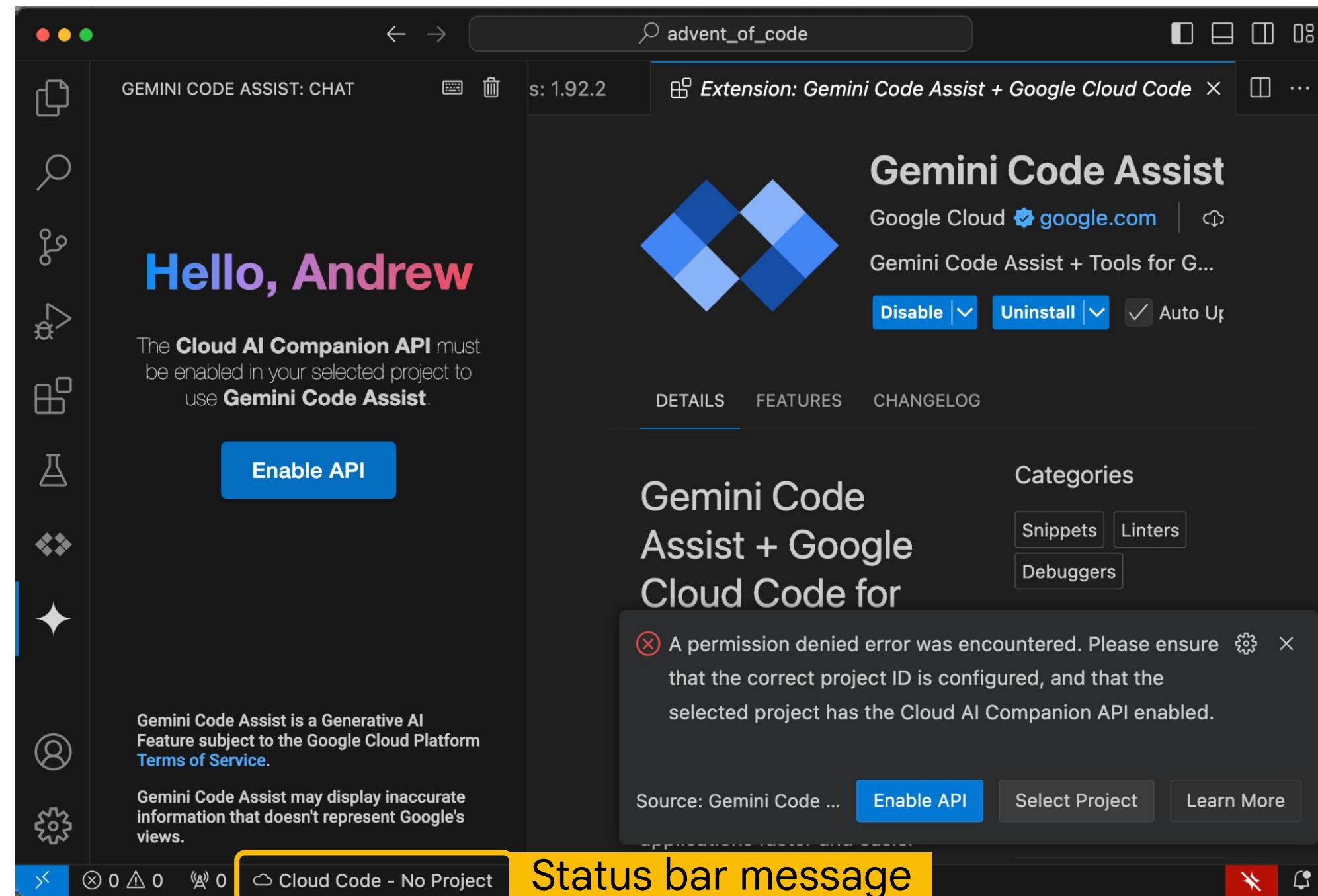
Select the newly-installed **Gemini icon** on the left nav.  
Then select **Login to Google Cloud**.



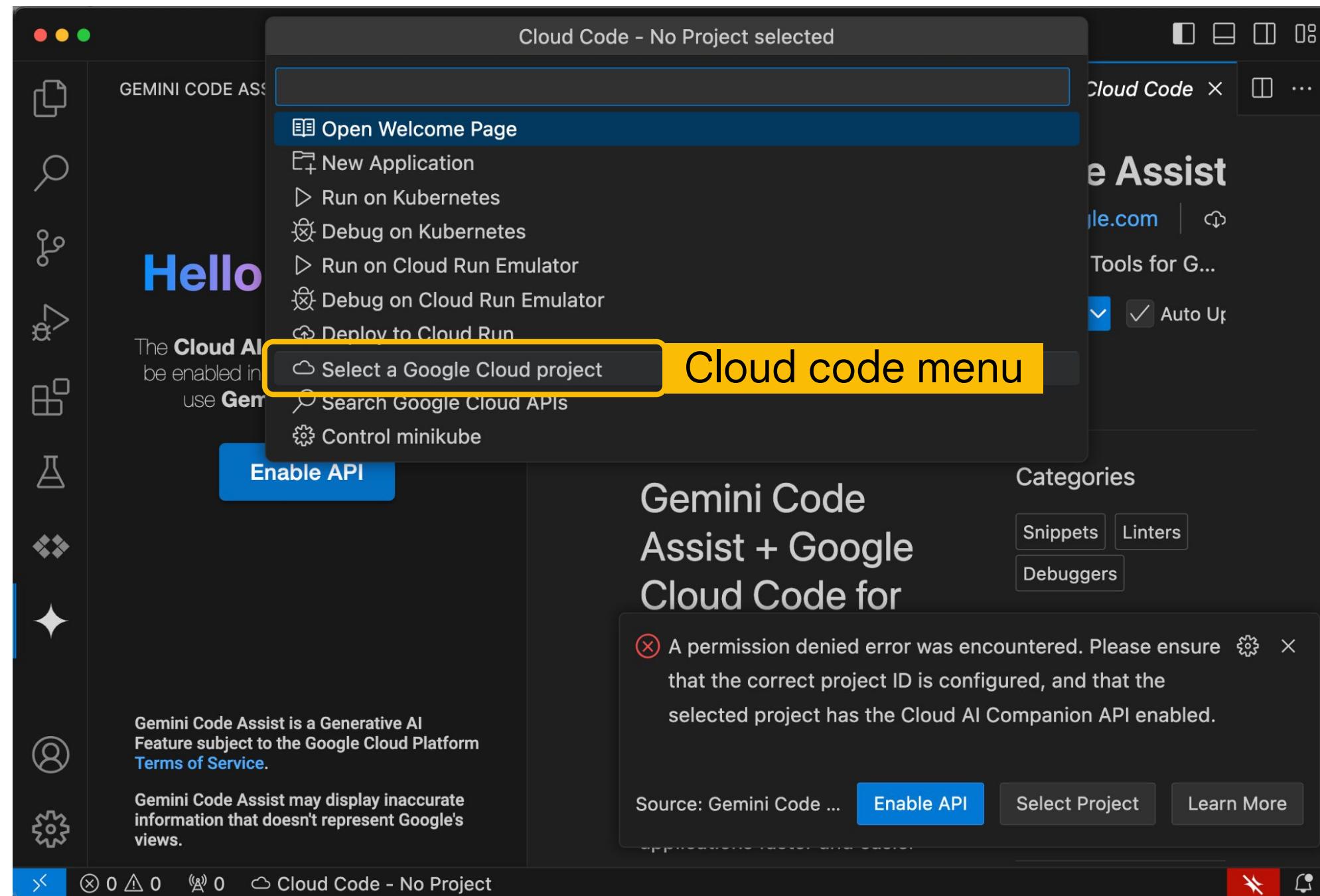
# Select your account that has been granted roles on the prepared project.



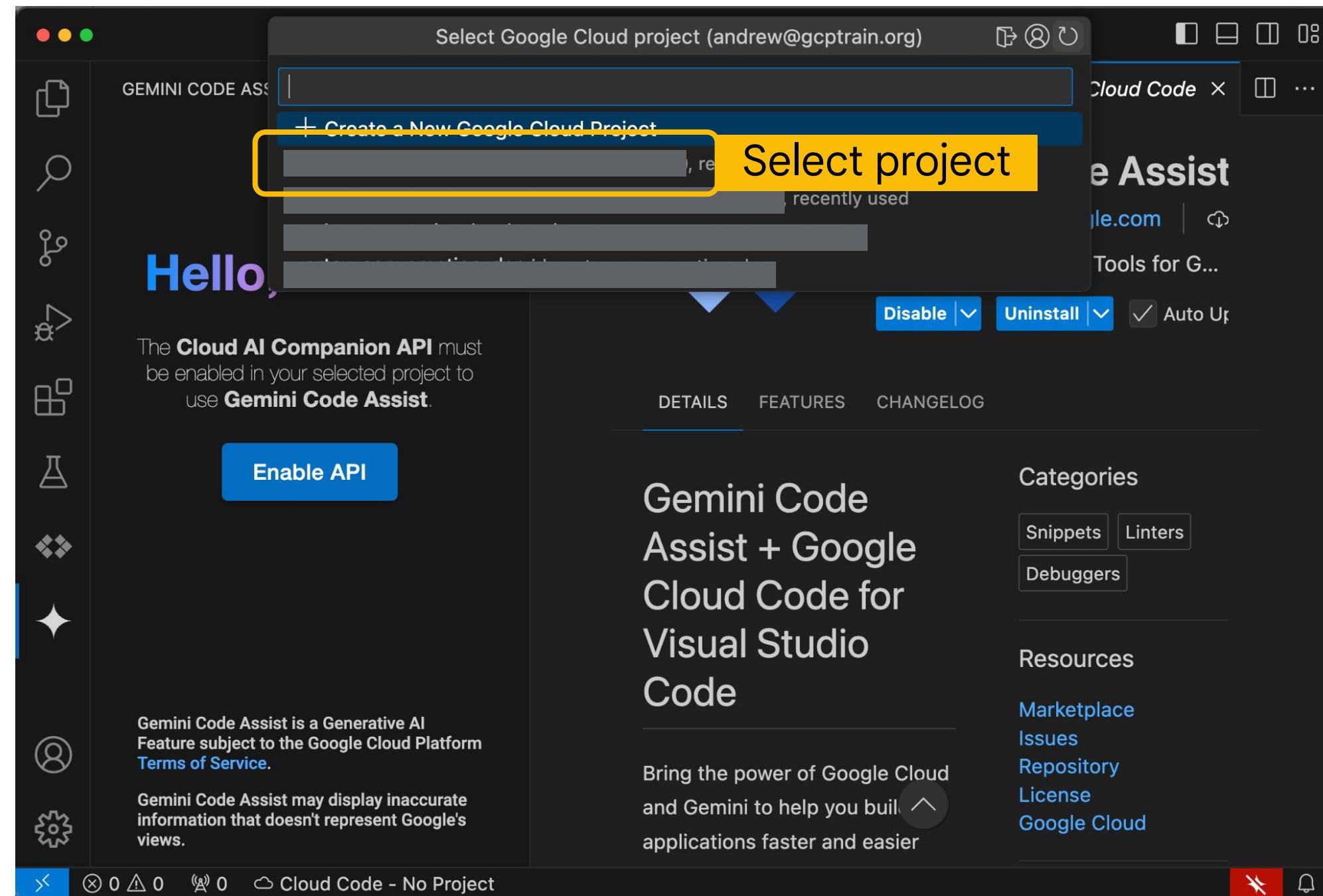
Once logged in, click the **Cloud Code - No Project** message in the status bar to open the Cloud Code menu.



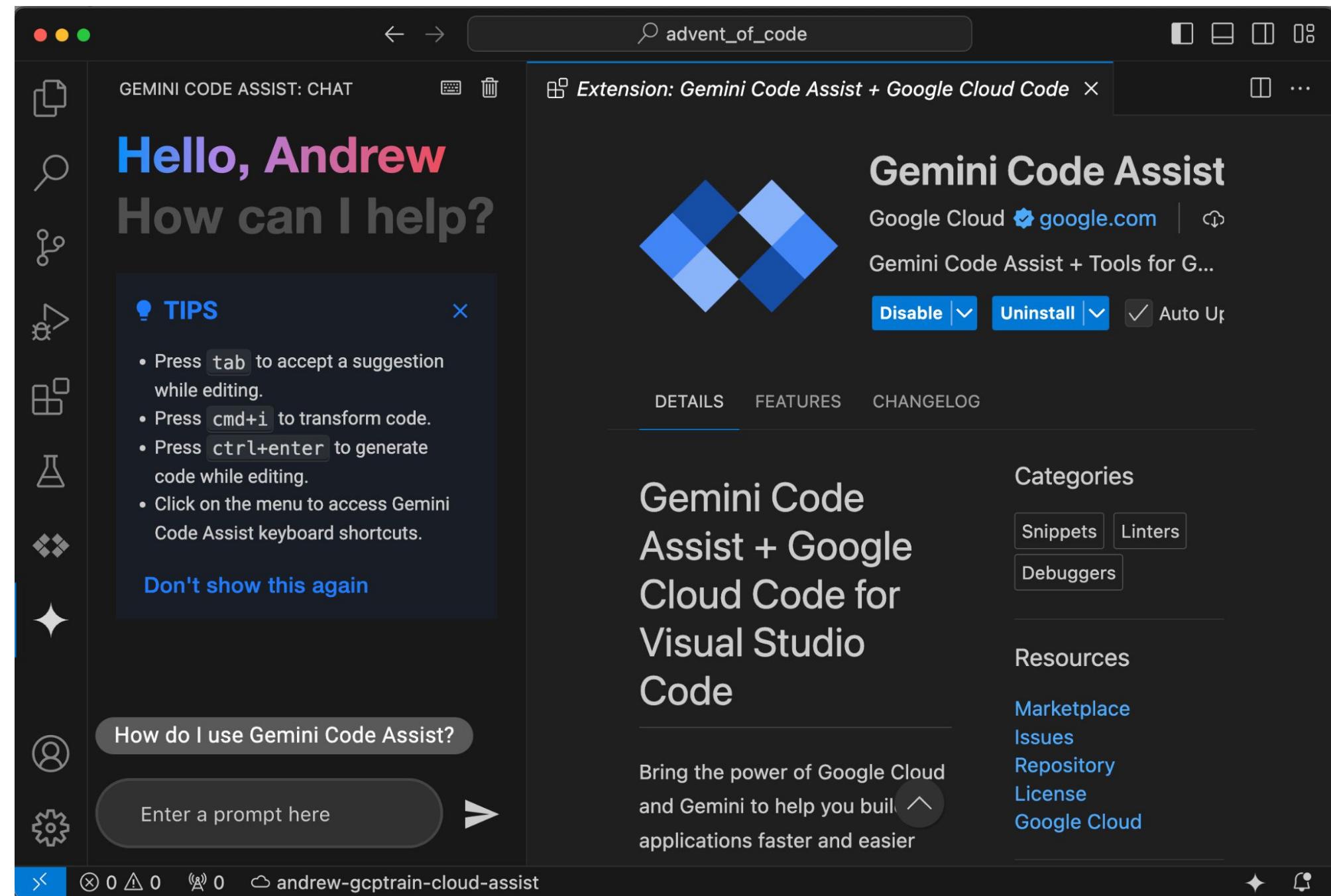
# Choose **Select a Google Cloud project** from the Cloud Code menu.



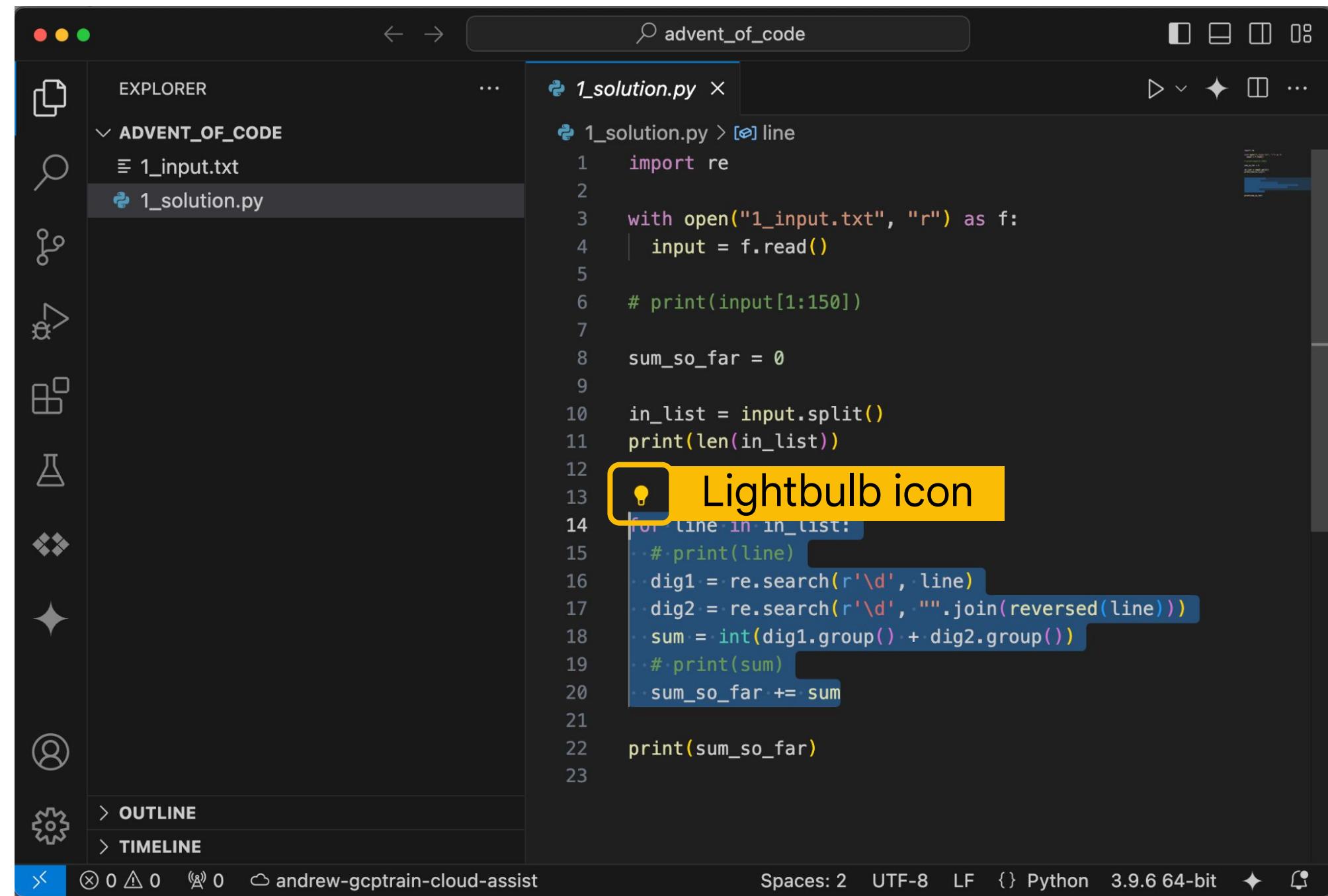
# Select the listed project that has been prepared to host your Gemini Code Assist license.



You should now be able to **chat with Gemini Code Assist**  
or **use it in your code files.**



For example, you can **highlight code** and click the **lightbulb icon** to pull up a menu of Smart Actions.

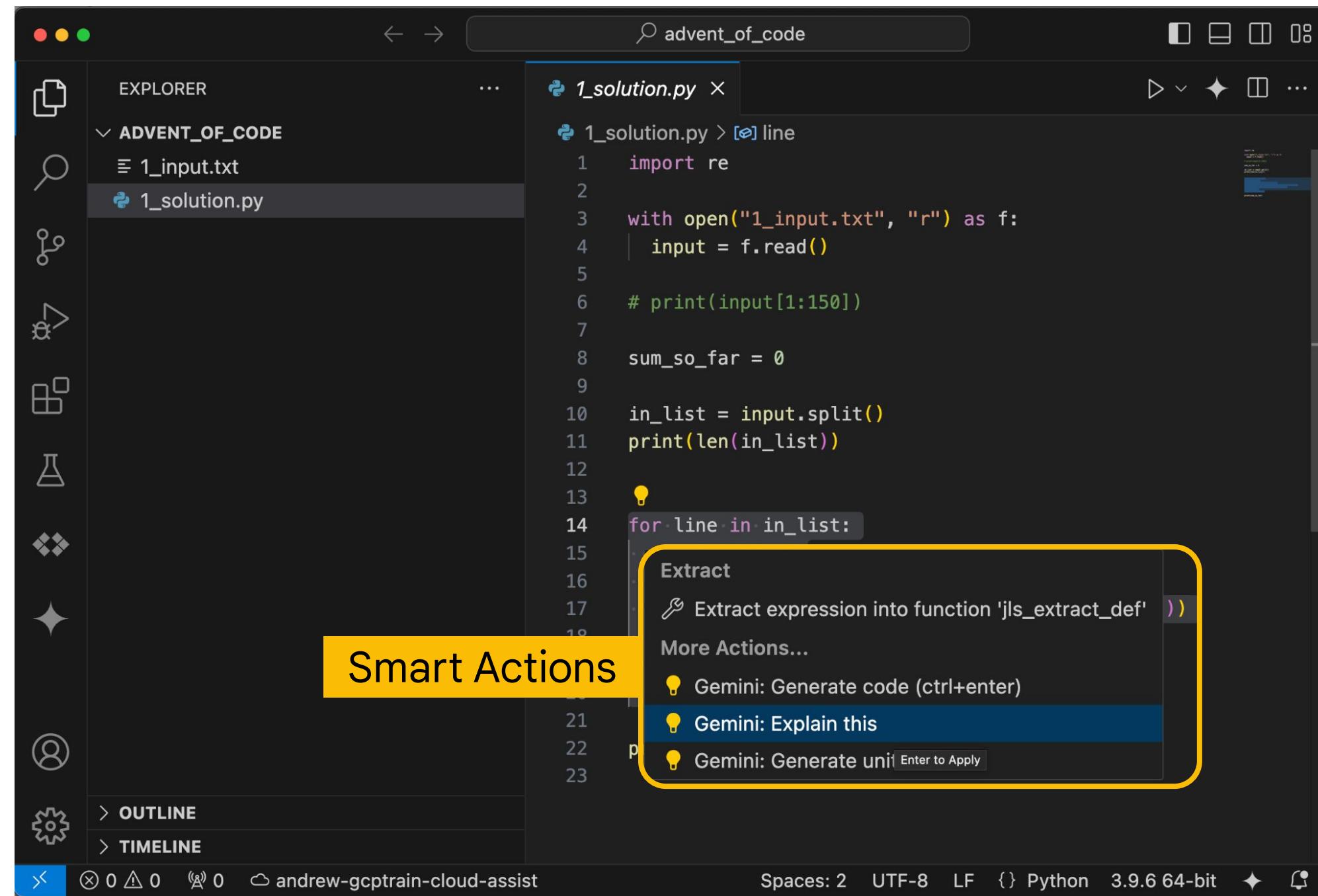


The screenshot shows a dark-themed instance of Visual Studio Code. In the center-right, there's a code editor window with a file named `1_solution.py`. A specific line of code is highlighted with a yellow background:

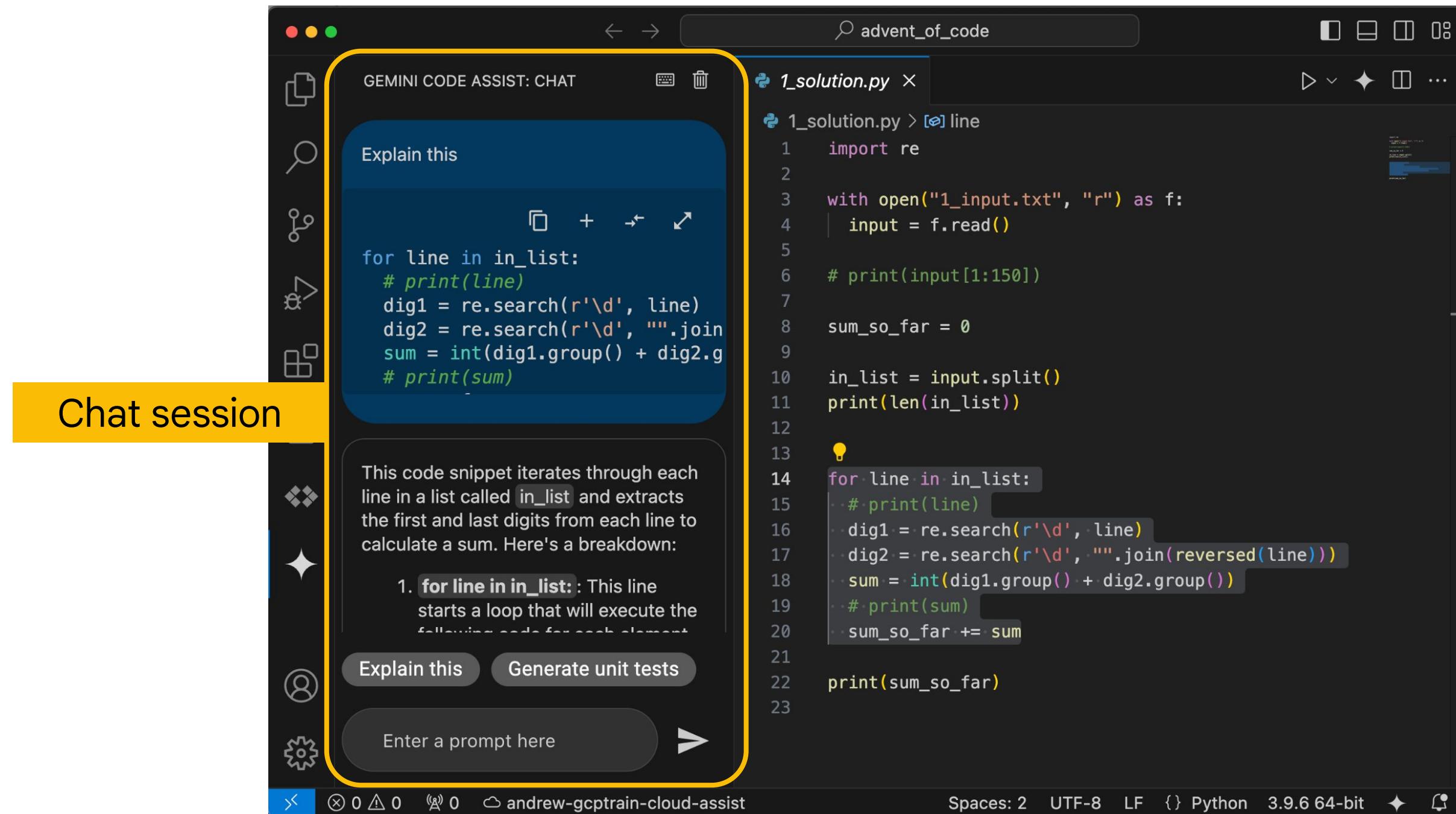
```
13     for line in in_list:  
14         # print(line) [Lightbulb icon]  
15         dig1 = re.search(r'\d', line)  
16         dig2 = re.search(r'\d', ''.join(reversed(line)))  
17         sum = int(dig1.group() + dig2.group())  
18         # print(sum)  
19         sum_so_far += sum  
20  
21     print(sum_so_far)  
22  
23 
```

A lightbulb icon is positioned next to the word `Lightbulb icon`, which is enclosed in a yellow box. The left side of the image shows the Explorer sidebar with files `1_input.txt` and `1_solution.py` listed under the `ADVENT_OF_CODE` folder. The bottom status bar indicates the file is Python 3.9.6 64-bit.

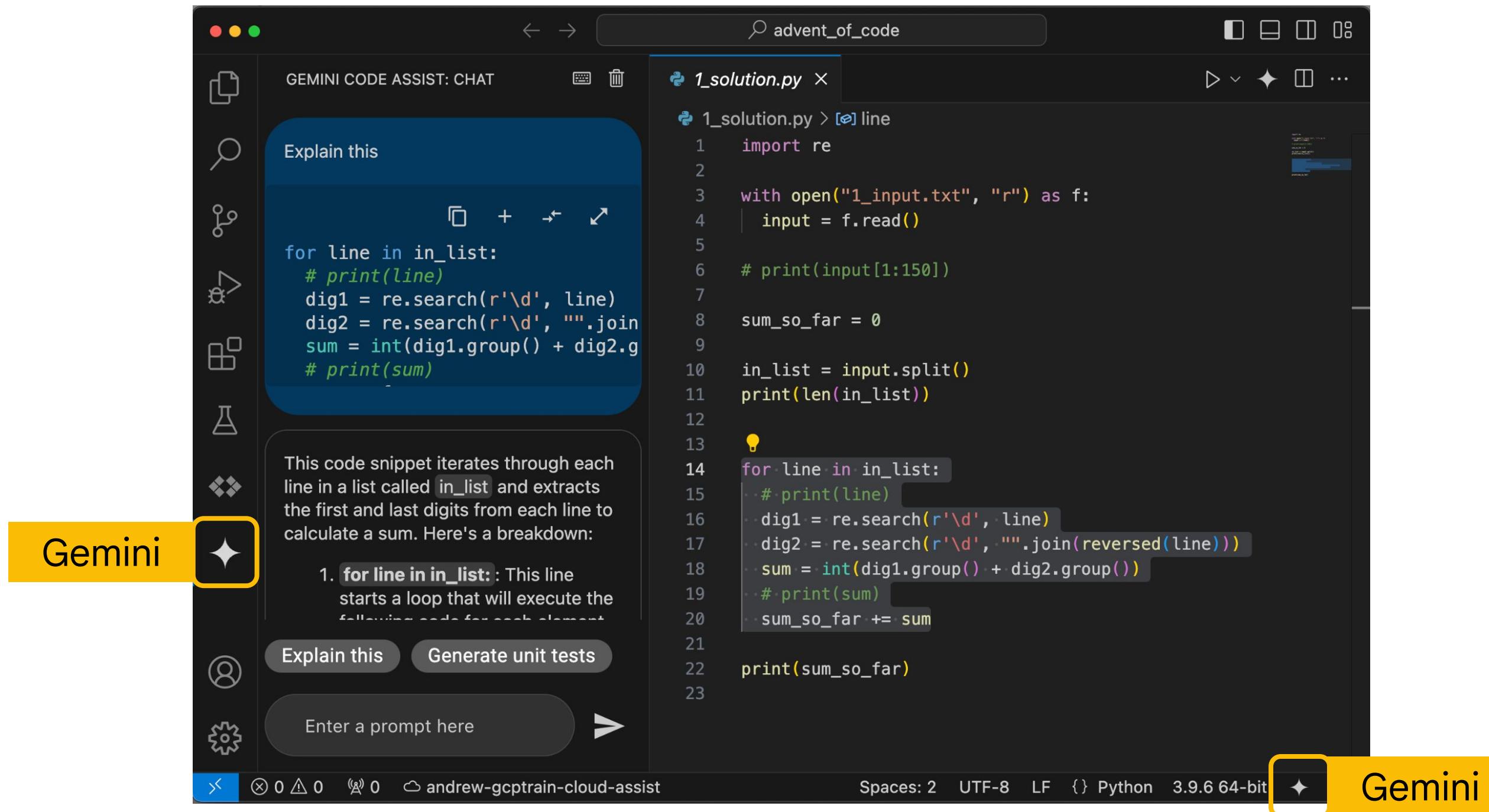
# Select Gemini: Explain this from the Smart Actions menu.



# This will start a chat session with Gemini Code Assist.



# You can always select the Gemini icons to summon or dismiss Gemini Code Assist.

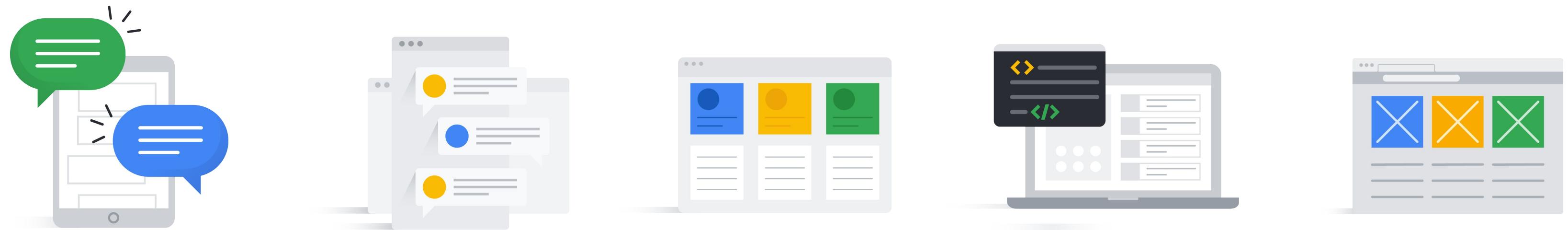


# Topics

- 01 Google and Gemini's Approach to Developer Acceleration
- 02 Where can you use Gemini Code Assist?
- 03 Enabling Gemini Code Assist
- 04 Example IDE Setup: VS Code
- 05 How to Interact with Gemini Code Assist
- 06 Common dev-related tasks for Gemini or Gemini Code Assist
- 07 Prompt Guidance



# What is included in the Context?



---

Last Message Sent  
(your prompt)

---

Chat History  
(may be summarized)

---

Contents of all open  
editor windows

---

Open more files to  
include the code

---

Close files that  
should be excluded

# Interactions

**Smart Actions**

DUET AI: CHAT

Getting started with a Cloud solution, optimizing resources, or using the CLI to manage Google Cloud.

Explain this

```
async def get_agent(session: dict[str, Any], user_id_token: Optional[str]) -> UserAgent:
    global user_agents
    if "uuid" not in session:
        session["uuid"] = str(uuid.uuid4())
    id = session["uuid"]
    if id not in user_agents:
```

The `get_agent` function is used to retrieve the agent associated with the current session. If the session does not have an agent, one is created. The function then sets the `User-Id-Token` header on the agent's client if a user ID token is provided. The function returns the agent.

Rate this answer:

Explain this Generate unit tests

Enter a prompt here >

**Chat**

main.py

```
120     except Exception as err:
121         raise HTTPException(status_code=500, detail=f"Error invoking
122
123
124     async def get_agent(session: dict[str, Any], user_id_token: Optional[str]) -> UserAgent:
125         global user_agents
126         if "uuid" not in session:
127             session["uuid"] = str(uuid.uuid4())
128         id = session["uuid"]
129         if id not in user_agents:
130             user_agents[id] = await init_agent(messages_from_dict(session["history"]))
131             user_agent = user_agents[id]
132             if user_id_token is not None:
133                 user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
134
135     return user_agent
136
137
138
139 @app.post("/reset")
140 async def reset(request: Request):
141     """Reset agent"""
142
143     if "uuid" not in request.session:
144         raise HTTPException(status_code=400, detail=f"No session to re
145
146     uuid = request.session["uuid"]
147     global user_agents
148     if uuid not in user_agents.keys():
149         raise HTTPException(status_code=500, detail=f"Current agent not
150
```

**In-Line Comments**

# Interactions: Gemini Code Assist Quick Pick Menu

Position the cursor in a code file or highlight code and press **Control+I** (for Windows and Linux) or **Command+I** (for macOS) to open the **Gemini Code Assist Quick Pick Menu**



A screenshot of a code editor showing a Python script. A floating search bar labeled "Enter a prompt or type "/" for commands" is visible. The code includes a try-except block, a route definition, a static file serving function, and a main block running the application.

```
45     except Exception as e:
46         return jsonify({ "error": str(e) })
47
48     Enter a prompt or type "/" for commands >
49     @app.route('/<path:path>')
50     def serve_static(path):
51         ... return send_from_directory('web', path)
52
53     if __name__ == "__main__":
54         app.run(port=int(os.environ.get('PORT', 80)))
```

# Interactions: Gemini Code Assist Quick Pick Menu

Enter a prompt to generate code.

Or press / to pull up a list  
of common commands to:

- /addComments
- /explain
- /transform

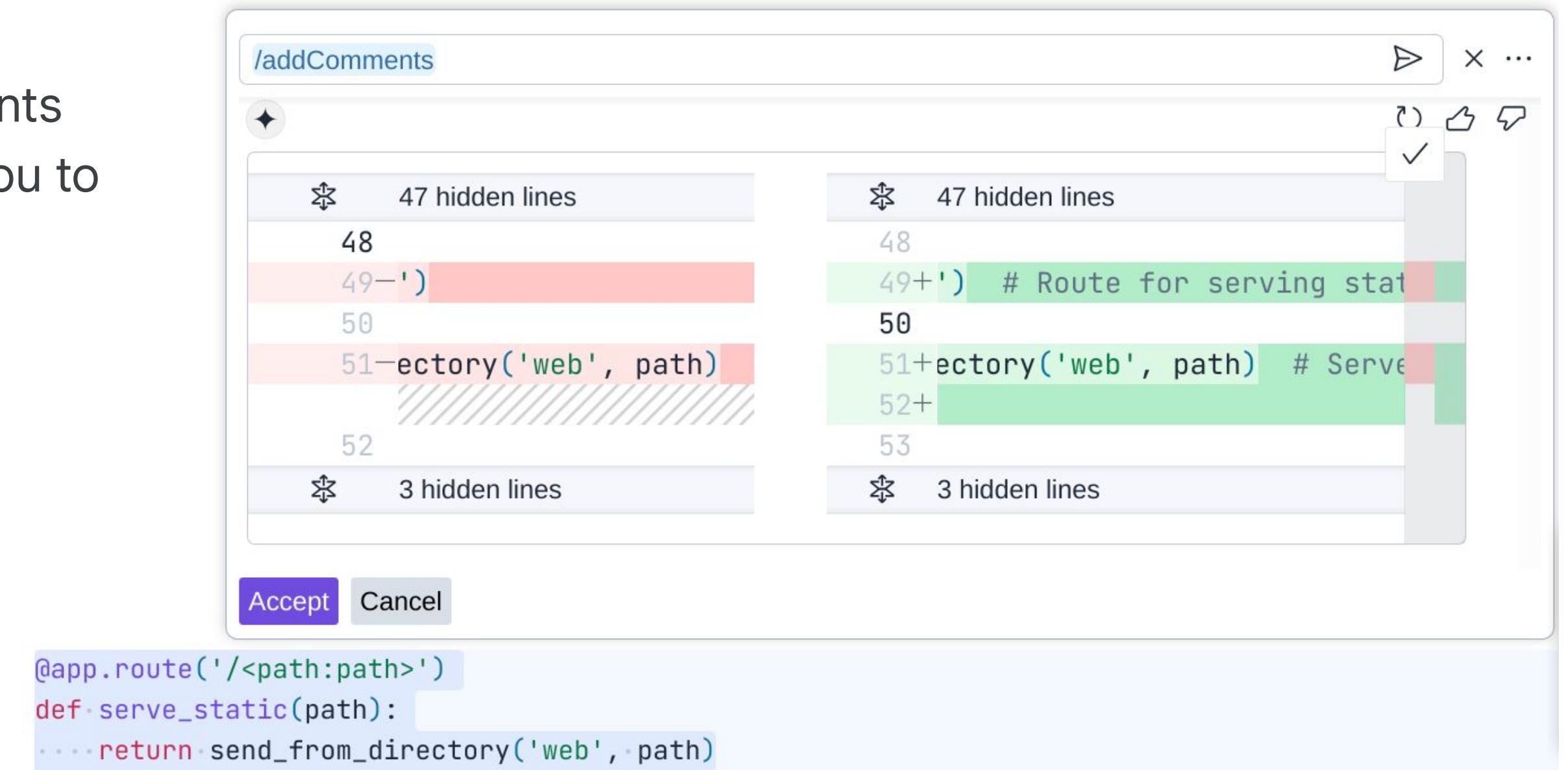
The screenshot shows a code editor with the following Python code:

```
45     except Exception as e:  
46         return jsonify({ "error": str(e) })  
47  
48     /  
49     @a /addComments  
50     de /explain  
51     .. /transform  
52  
53     if __name__ == "__main__":  
54         app.run(port=int(os.environ.get('PORT', 80)))
```

A quick pick menu is displayed, listing three commands: /addComments, /explain, and /transform. To the right of the menu, there are three corresponding options: Add comments to code, Explain code, and Edit code. The menu is triggered by pressing the '/' key at line 48.

# Interactions: Gemini Code Assist Quick Pick Menu

Any generated code or comments will be presented as a diff for you to accept or reject.



# Interactions: Smart Actions

```
121     raise HTTPException(status_code=500, detail=f"Error invoking agent: {err}")
122
123
124     async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
125         ...
126
127         session: str = str(uuid.uuid4())
128
129         session:
130         session: [""] = messages_to_dict(BASE_HISTORY)
131
132         if id not in user_agents:
133             user_agents[id] = await init_agent(messages_from_dict(session["history"]))
134
135             user_agent = user_agents[id]
136
137             if user_id_token is not None:
138                 user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
139
140
141     @app.post("/reset")
142     async def reset(request: Request):
```



More Actions...

Generate code

Explain this

Generate unit tests

session:  
= str(uuid.uuid4())

session:  
[""] = messages\_to\_dict(BASE\_HISTORY)

if id not in user\_agents:  
user\_agents[id] = await init\_agent(messages\_from\_dict(session["history"]))

user\_agent = user\_agents[id]

if user\_id\_token is not None:

user\_agent.client.headers["User-Id-Token"] = f"Bearer {user\_id\_token}"

return user\_agent

# Interactions: Smart Actions

```
121     raise HTTPException(status_code=500, detail=f"Error invoking agent: {err}")
122
123
124 async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
125
126     session_id = str(uuid.uuid4())
127
128     session: dict[str, Any] = messages_to_dict(BASE_HISTORY)
129
130     if id not in user_agents:
131         user_agents[id] = await init_agent(messages_from_dict(session["history"]))
132
133     user_agent = user_agents[id]
134
135     if user_id_token is not None:
136         user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
137
138
139 @app.post("/reset")
140 async def reset(request: Request):
```

Smart Actions

- More Actions...
- Generate code
- Explain this
- Generate unit tests

# Interactions: Smart Actions

The screenshot illustrates a developer's workflow within a code editor, specifically interacting with a Python file named `main.py`.

**Code Editor View:**

```

DUET AI: CHAT
ID TOKEN is provided. The function returns the agent.

Rate this answer: ⌘ ⌘

Explain this

if user_id_token is not None:
    user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"

```

**AI Chat Panel (Left):**

DUET AI: CHAT  
ID TOKEN is provided. The function returns the agent.  
Rate this answer: ⌘ ⌘  
Explain this

**Code Completion Suggestion (Right):**

The `if user_id_token is not None:` statement checks if the user ID token is not None. If it is not None, the `user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"` statement sets the `User-Id-Token` header on the user agent's client to the value of the user ID token. The `Bearer` prefix is required for the header.

**More Actions... (Dropdown):**

- Generate code
- Explain this (selected)
- Generate unit tests

**Code Editor Content (`main.py`):**

```

120     except Exception as err:
121         raise HTTPException(status_code=500, detail=f"Error invoking agent: {err}")
122
123
124     async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
125         global user_agents
126         if "uuid" not in session:
127             session["uuid"] = str(uuid.uuid4())
128         id = session["uuid"]
129         if "history" not in session:
130             session["history"] = messages_to_dict(BASE_HISTORY)
131         if id not in user_agents:
132             user_agents[id] = await init_agent(messages_from_dict(session["history"]))
133         user_agent = user_agents[id]
134         if user_id_token is not None:
135             user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
136
137         @app.post("/reset")
138         async def reset_agent():
139             """Reset agent"""
140
141             if "uuid" not in request.session:
142                 raise HTTPException(status_code=400, detail=f"No session to reset.")
143
144             uuid = request.session["uuid"]
145             global user_agents
146             if uuid not in user_agents.keys():
147                 raise HTTPException(status_code=500, detail=f"Current agent not found")
148
149
150

```

# Interactions: Chat interface

The image shows a split-screen interface. On the left is a 'DUET AI: CHAT' window containing a message from Duet AI about navigating Google Cloud. On the right is a code editor window titled 'main.py' showing Python code for managing agents.

**DUET AI: CHAT**

Duet AI is an experimental AI model to help you navigate Google Cloud. Get answers to your questions about getting started with a Cloud solution, optimizing resources, or using the CLI to manage Google Cloud.

**main.py**

```

genai-databases-retrieval-app > langchain_tools_demo > main.py > ...
120     except Exception as err:
121         raise HTTPException(status_code=500, detail=f"Error invoking agent: {err}")
122
123     ...
124     async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
125         global user_agents
126         if "uuid" not in session:
127             session["uuid"] = str(uuid.uuid4())
128         id = session["uuid"]
129         if "history" not in session:
130             session["history"] = messages_to_dict(BASE_HISTORY)
131         if id not in user_agents:
132             user_agents[id] = await init_agent(messages_from_dict(session["history"]))
133         user_agent = user_agents[id]
134         if user_id_token is not None:
135             user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
136         return user_agent
137
138
139     @app.post("/reset")
140     async def reset(request: Request):
141         """Reset agent"""
142
143         if "uuid" not in request.session:
144             raise HTTPException(status_code=400, detail=f"No session to reset.")
145
146         uuid = request.session["uuid"]
147         global user_agents
148         if uuid not in user_agents.keys():
149             raise HTTPException(status_code=500, detail=f"Current agent not found")
150

```

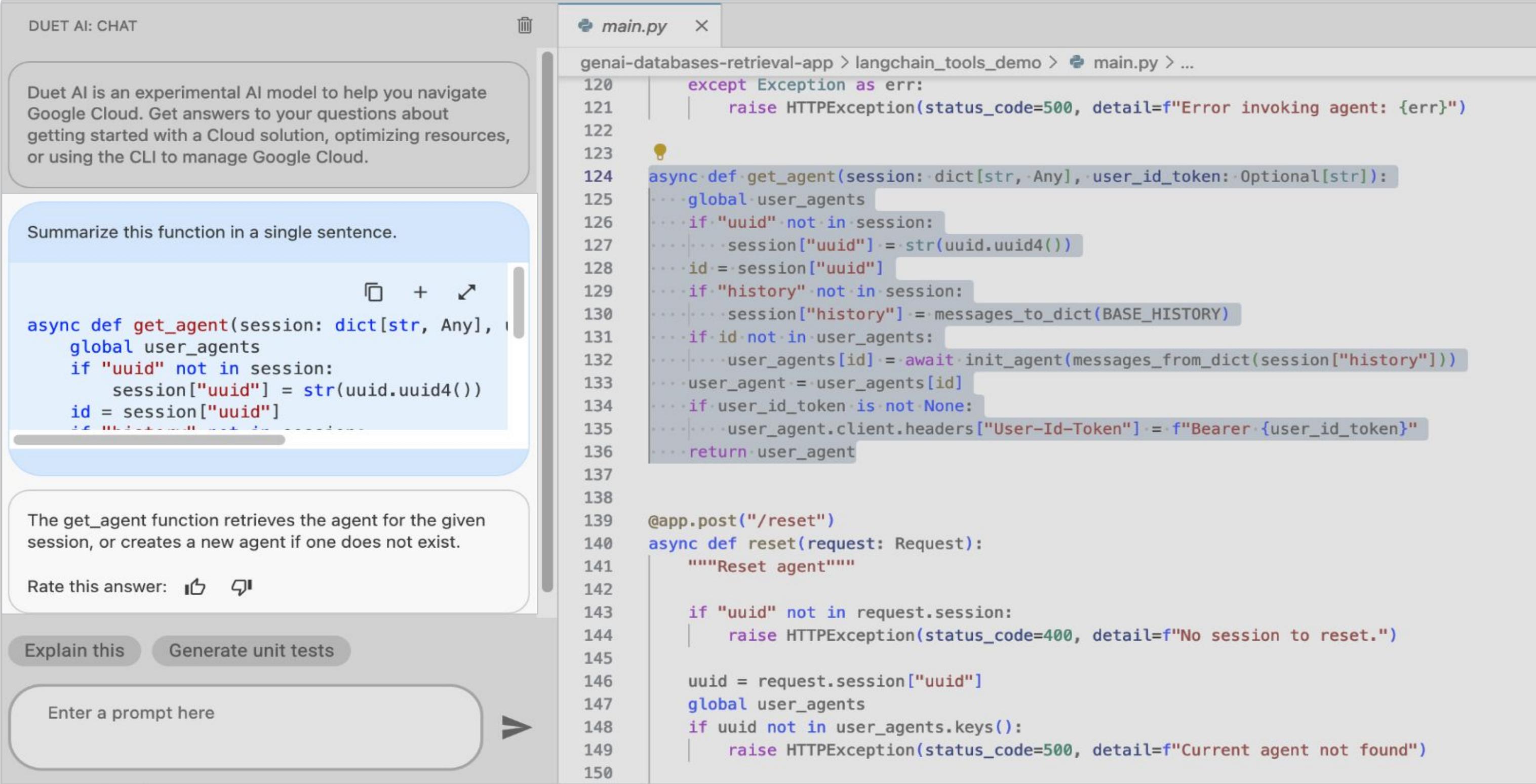
Buttons at the bottom left include 'How do I use Duet AI?', 'Explain this', 'Generate unit tests', and 'Summarize this function in a single sentence.' A large right-pointing arrow indicates a flow from the user input to the generated code.

# Interactions: Chat interface

The image shows a screenshot of a developer's workspace. On the left, there is a 'DUET AI: CHAT' window containing a message from Duet AI about navigating Google Cloud. Below this are buttons for 'How do I use Duet AI?', 'Explain this', and 'Generate unit tests'. At the bottom, there is a text input field with the placeholder 'Summarize this function in a single sentence.' followed by a right-pointing arrow. On the right, there is a code editor window titled 'main.py' showing Python code for managing user agents and sessions. The code includes functions for getting an agent, resetting a session, and handling HTTP exceptions.

```
genai-databases-retrieval-app > langchain_tools_demo > main.py > ...
120     except Exception as err:
121         raise HTTPException(status_code=500, detail=f"Error invoking agent: {err}")
122
123
124     async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
125         global user_agents
126         if "uuid" not in session:
127             session["uuid"] = str(uuid.uuid4())
128         id = session["uuid"]
129         if "history" not in session:
130             session["history"] = messages_to_dict(BASE_HISTORY)
131         if id not in user_agents:
132             user_agents[id] = await init_agent(messages_from_dict(session["history"]))
133         user_agent = user_agents[id]
134         if user_id_token is not None:
135             user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
136         return user_agent
137
138
139     @app.post("/reset")
140     async def reset(request: Request):
141         """Reset agent"""
142
143         if "uuid" not in request.session:
144             raise HTTPException(status_code=400, detail=f"No session to reset.")
145
146         uuid = request.session["uuid"]
147         global user_agents
148         if uuid not in user_agents.keys():
149             raise HTTPException(status_code=500, detail=f"Current agent not found")
```

# Interactions: Chat interface



The image shows a split-screen interface. On the left is the Duet AI Chat interface, and on the right is a code editor.

**Duet AI: CHAT**

Duet AI is an experimental AI model to help you navigate Google Cloud. Get answers to your questions about getting started with a Cloud solution, optimizing resources, or using the CLI to manage Google Cloud.

Summarize this function in a single sentence.

```
async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
    global user_agents
    if "uuid" not in session:
        session["uuid"] = str(uuid.uuid4())
    id = session["uuid"]
    if "history" not in session:
        session["history"] = messages_to_dict(BASE_HISTORY)
    if id not in user_agents:
        user_agents[id] = await init_agent(messages_from_dict(session["history"]))
    user_agent = user_agents[id]
    if user_id_token is not None:
        user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
    return user_agent
```

The get\_agent function retrieves the agent for the given session, or creates a new agent if one does not exist.

Rate this answer: ⌂ ⌂

Explain this Generate unit tests

Enter a prompt here ➤

**main.py**

```
genai-databases-retrieval-app > langchain_tools_demo > main.py > ...
120     except Exception as err:
121         raise HTTPException(status_code=500, detail=f"Error invoking agent: {err}")
122
123
124     async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
125         global user_agents
126         if "uuid" not in session:
127             session["uuid"] = str(uuid.uuid4())
128         id = session["uuid"]
129         if "history" not in session:
130             session["history"] = messages_to_dict(BASE_HISTORY)
131         if id not in user_agents:
132             user_agents[id] = await init_agent(messages_from_dict(session["history"]))
133         user_agent = user_agents[id]
134         if user_id_token is not None:
135             user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
136         return user_agent
137
138
139     @app.post("/reset")
140     async def reset(request: Request):
141         """Reset agent"""
142
143         if "uuid" not in request.session:
144             raise HTTPException(status_code=400, detail=f"No session to reset.")
145
146         uuid = request.session["uuid"]
147         global user_agents
148         if uuid not in user_agents.keys():
149             raise HTTPException(status_code=500, detail=f"Current agent not found")
```

# Interactions: Chat interface

The screenshot shows the DUET AI: CHAT interface on the left and a code editor window on the right.

**DUET AI: CHAT (Left):**

- Sources:** A list item labeled "1." is selected.
- Question:** "What happens if the user\_id\_token is null?"
- Code Snippet:**

```
async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
    global user_agents
    if "uuid" not in session:
        session["uuid"] = str(uuid.uuid4())
    id = session["uuid"]
    ...
```
- Note:** "If the user\_id\_token is null, the get\_agent() function will not set the user agent's client headers "User-Id-Token" key."
- Rating:** "Rate this answer: ⚡ ⚡"
- Buttons:** "Explain this" and "Generate unit tests".
- Text Input:** "Enter a prompt here" with a right-pointing arrow button.

**Code Editor (Right):**

```

genai-databases-retrieval-app > langchain_tools_demo > main.py > ...
120     except Exception as err:
121         raise HTTPException(status_code=500, detail=f"Error invoking agent: {err}")
122
123
124     async def get_agent(session: dict[str, Any], user_id_token: Optional[str]):
125         global user_agents
126         if "uuid" not in session:
127             session["uuid"] = str(uuid.uuid4())
128         id = session["uuid"]
129         if "history" not in session:
130             session["history"] = messages_to_dict(BASE_HISTORY)
131         if id not in user_agents:
132             user_agents[id] = await init_agent(messages_from_dict(session["history"]))
133         user_agent = user_agents[id]
134         if user_id_token is not None:
135             user_agent.client.headers["User-Id-Token"] = f"Bearer {user_id_token}"
136         return user_agent
137
138
139 @app.post("/reset")
140 async def reset(request: Request):
141     """Reset agent"""
142
143     if "uuid" not in request.session:
144         raise HTTPException(status_code=400, detail=f"No session to reset.")
145
146     uuid = request.session["uuid"]
147     global user_agents
148     if uuid not in user_agents.keys():
149         raise HTTPException(status_code=500, detail=f"Current agent not found")
150

```

The Chat interface is ideal for being able to ask follow up questions or rephrase your request.

DUET AI: CHAT

Duet AI is an experimental AI model to help you navigate Google Cloud. Get answers to your questions about getting started with a Cloud solution, optimizing resources, or using the CLI to manage Google Cloud.

Write a function to convert temperature into celsius.

Sure. Here is a function to convert temperature from Fahrenheit to Celsius:

```
def convert_fahrenheit_to_celsius(fahrenheit):
    """Converts a temperature from Fahrenheit to Celsius.

    Args:
        fahrenheit: The temperature in Fahrenheit.

    Returns:
        The temperature in Celsius.
    """
    celsius = (fahrenheit - 32) * 5 / 9
    return celsius
```

Rate this answer: ⚡ ⚡

sorry I meant Kelvin

Sure. Here is a function to convert temperature from Fahrenheit to Kelvin:

```
def convert_fahrenheit_to_kelvin(fahrenheit):
    """Converts a temperature from Fahrenheit to Kelvin.

    Args:
        fahrenheit: The temperature in Fahrenheit.
    """
    kelvin = (fahrenheit - 32) * 5 / 9 + 273.15
    return kelvin
```

4 ➤

# Interactions: In-line comments

```
14 # Summary of the functions performed in this module
15 # Add a summary of the functions performed in this module
16 # This module is the main entry point for the retrieval service. It defines the FastAPI app,
17 # parses the configuration file, and initializes the datastore. It also includes the routes for
18 # the different endpoints.
19 # The retrieval service is a web service that provides access to the datastore. It allows users
20 # to search for documents, add documents to the datastore, and delete documents from the
21 # datastore.
22
23
24
25
```

14 # Summary of the functions performed in this module  
15 # Add a summary of the functions performed in this module  
16 # This module is the main entry point for the retrieval service. It defines the FastAPI app,  
17 # parses the configuration file, and initializes the datastore. It also includes the routes for  
18 # the different endpoints.  
19 # The retrieval service is a web service that provides access to the datastore. It allows users  
20 # to search for documents, add documents to the datastore, and delete documents from the  
21 # datastore.  
22  
23  
24  
25

# Topics

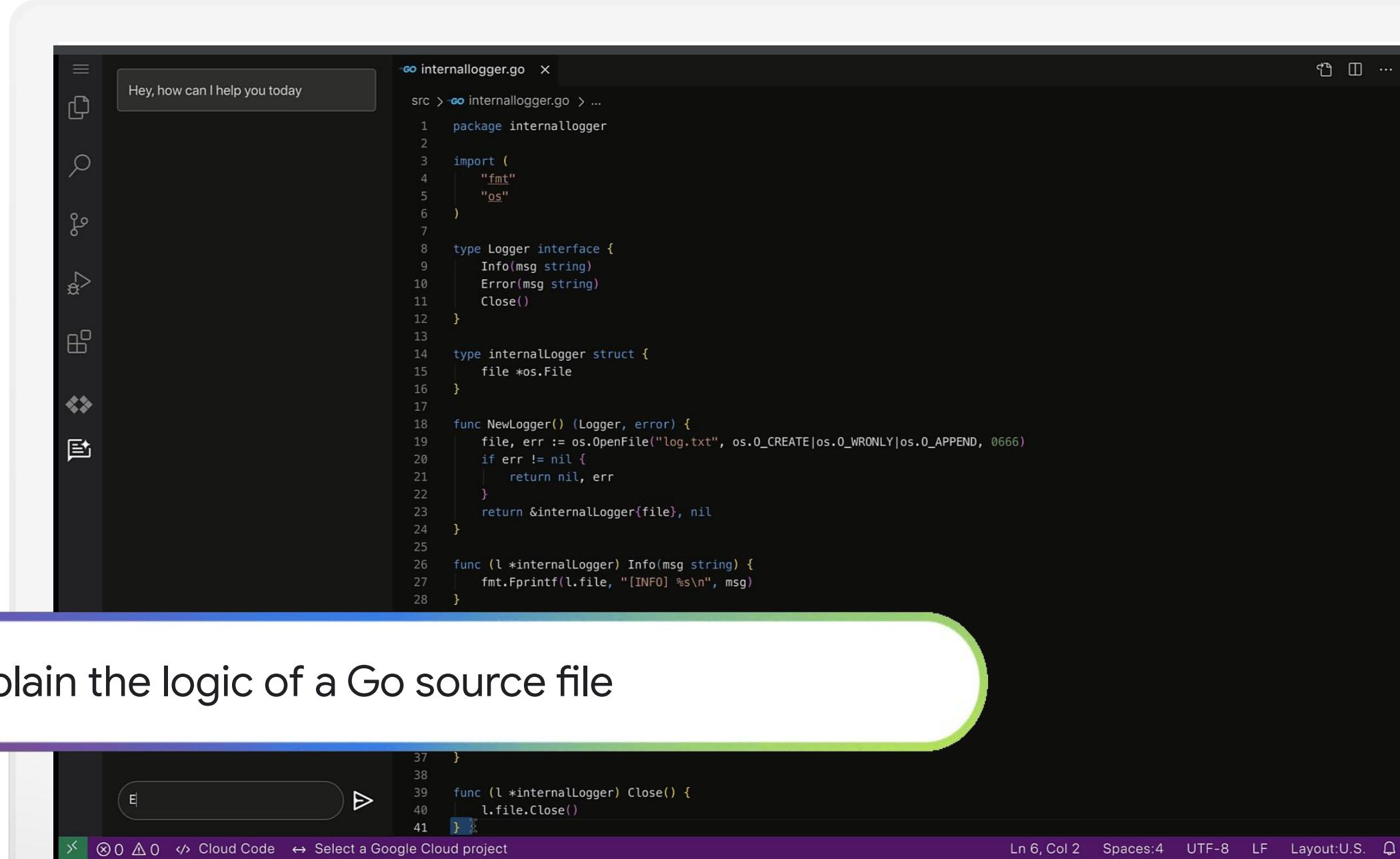
- 01 Google and Gemini's Approach to Developer Acceleration
- 02 Where can you use Gemini Code Assist?
- 03 Enabling Gemini Code Assist
- 04 Example IDE Setup: VS Code
- 05 How to Interact with Gemini Code Assist
- 06 Common dev-related tasks for Gemini or Gemini Code Assist
- 07 Prompt Guidance



# Code explanations in natural language

Helps improve:

- Code review speed
- Code maintainability
- Testing and debugging



```
internallogger.go
Hey, how can I help you today
src > internallogger.go > ...
1 package internallogger
2
3 import (
4     "fmt"
5     "os"
6 )
7
8 type Logger interface {
9     Info(msg string)
10    Error(msg string)
11    Close()
12 }
13
14 type internalLogger struct {
15     file *os.File
16 }
17
18 func NewLogger() (Logger, error) {
19     file, err := os.OpenFile("log.txt", os.O_CREATE|os.O_WRONLY|os.O_APPEND, 0666)
20     if err != nil {
21         return nil, err
22     }
23     return &internalLogger{file}, nil
24 }
25
26 func (l *internalLogger) Info(msg string) {
27     fmt.Fprintf(l.file, "[INFO] %s\n", msg)
28 }
```

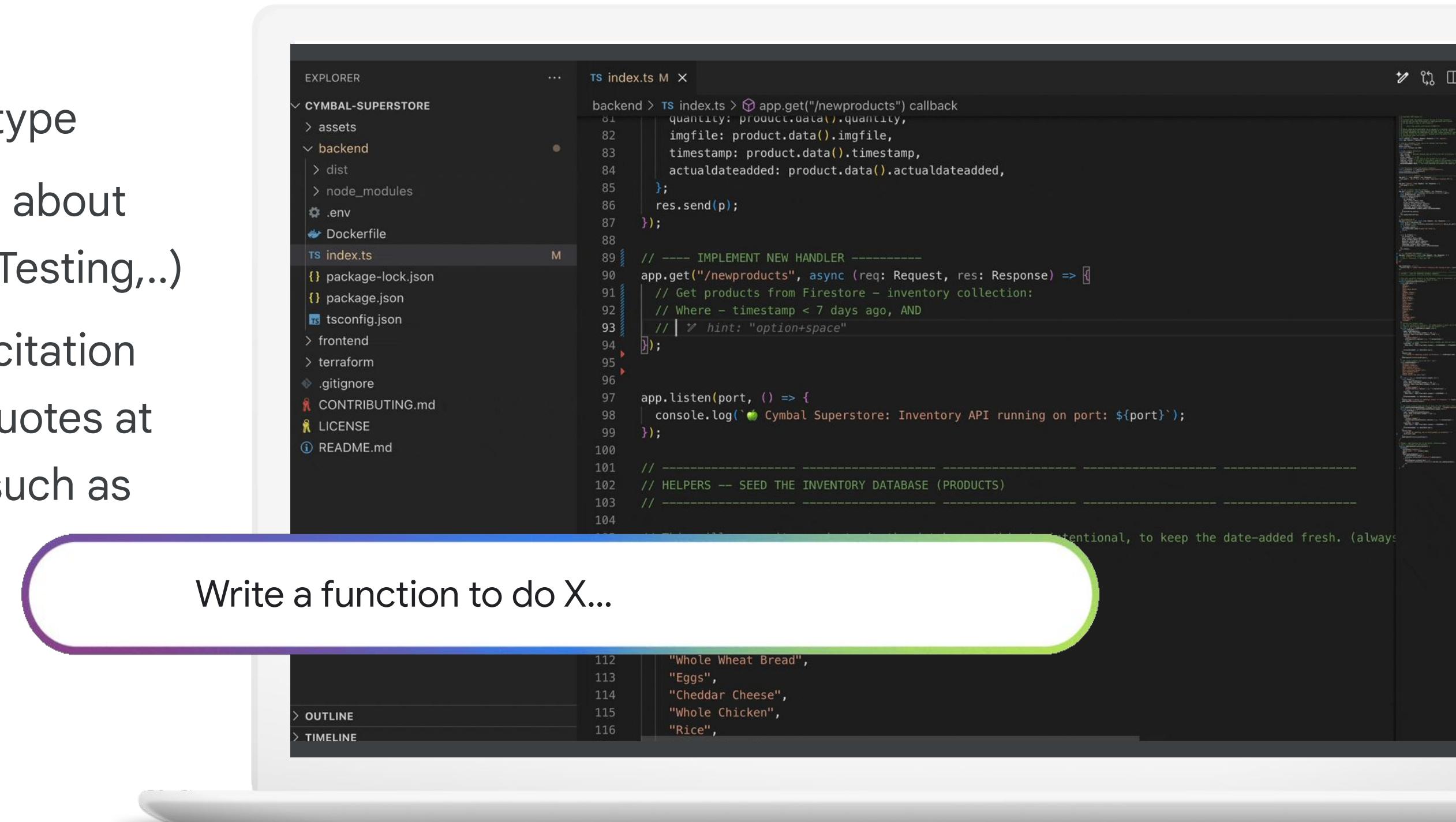
Explain the logic of a Go source file

```
37 }
38
39 func (l *internalLogger) Close() {
40     l.file.Close()
41 }
```

Ln 6, Col 2 Spaces:4 UTF-8 LF Layout:U.S.

# Code completion and generation

- Completes your code as you type
- Chat with AI pair programmer about code (e.g. Code explanation, Testing,...)
- Gemini Code Assist provides citation information when it directly quotes at length from another source, such as existing open source code.



The screenshot shows a code editor interface with the following details:

- EXPLORER** sidebar: Shows the project structure under "CYMBAL-SUPERSTORE".
- index.ts** file (selected): Contains code for a Node.js application. A specific line of code is highlighted with a blue bracket, indicating the point of code generation or completion.
- Code Completion Suggestion:** A purple callout bubble contains the text "Write a function to do X...".
- Bottom pane:** Shows additional code snippets, likely generated by Gemini Code Assist, related to the highlighted line.

```

    TS index.ts M X
    backend > TS index.ts > app.get("/newproducts") callback
    81   quantity: product.data().quantity,
    82   imgfile: product.data().imgfile,
    83   timestamp: product.data().timestamp,
    84   actualdateadded: product.data().actualdateadded,
    85   );
    86   res.send(p);
    87 );
    88 // ---- IMPLEMENT NEW HANDLER -----
    89 app.get("/newproducts", async (req: Request, res: Response) => [
    90   // Get products from Firestore - inventory collection:
    91   // Where - timestamp < 7 days ago, AND
    92   // | ✓ hint: "option+space"
    93 ];
    94
    95
    96
    97 app.listen(port, () => {
    98   console.log(` Cymbal Superstore: Inventory API running on port: ${port}`);
    99 });
    100
    101 //
    102 // HELPERS -- SEED THE INVENTORY DATABASE (PRODUCTS)
    103 //
    104
    112   "Whole Wheat Bread",
    113   "Eggs",
    114   "Cheddar Cheese",
    115   "Whole Chicken",
    116   "Rice",
  
```

# Database query completion and generation

- Provide your schema and query your databases using natural language
  - You can interact with BigQuery in its Studio, and Cloud SQL and other databases with Gemini in the Cloud Console

The screenshot shows the VS Code interface with the following details:

- EXPLORER** sidebar: Shows the project structure with the following files and folders:
  - assets
  - backend (selected)
  - dist
  - node\_modules
  - .env
  - Dockerfile
  - index.ts (selected)
  - package-lock.json
  - package.json
  - tsconfig.json
  - frontend
  - terraform
  - .gitignore
  - CONTRIBUTING.md
  - LICENSE
  - README.md
- index.ts** editor tab: The code is written in TypeScript (indicated by the 'ts' icon). The code implements a new endpoint for getting new products from Firestore:

```
backend > ts index.ts > app.get("/newproducts") callback
  81     quantity: product.data().quantity,
  82     imgfile: product.data().imgfile,
  83     timestamp: product.data().timestamp,
  84     actualdateadded: product.data().actualdateadded,
  85   };
  86   res.send(p);
  87 });
  88
  89 // ---- IMPLEMENT NEW HANDLER -----
  90 app.get("/newproducts", async (req: Request, res: Response) => {
  91   // Get products from Firestore - inventory collection:
  92   // Where - timestamp < 7 days ago, AND
  93   // |  hint: "option+space"
  94 });
  95
  96
  97 app.listen(port, () => {
  98   console.log(`🍏 Cymbal Superstore: Inventory API running on port: ${port}`);
  99 });
 100
 101 // -----
 102 // HELPERS -- SEED THE INVENTORY DATABASE (PRODUCTS)
 103 // -----
 104
```

# Create a query for an API call

# License attribution assistance

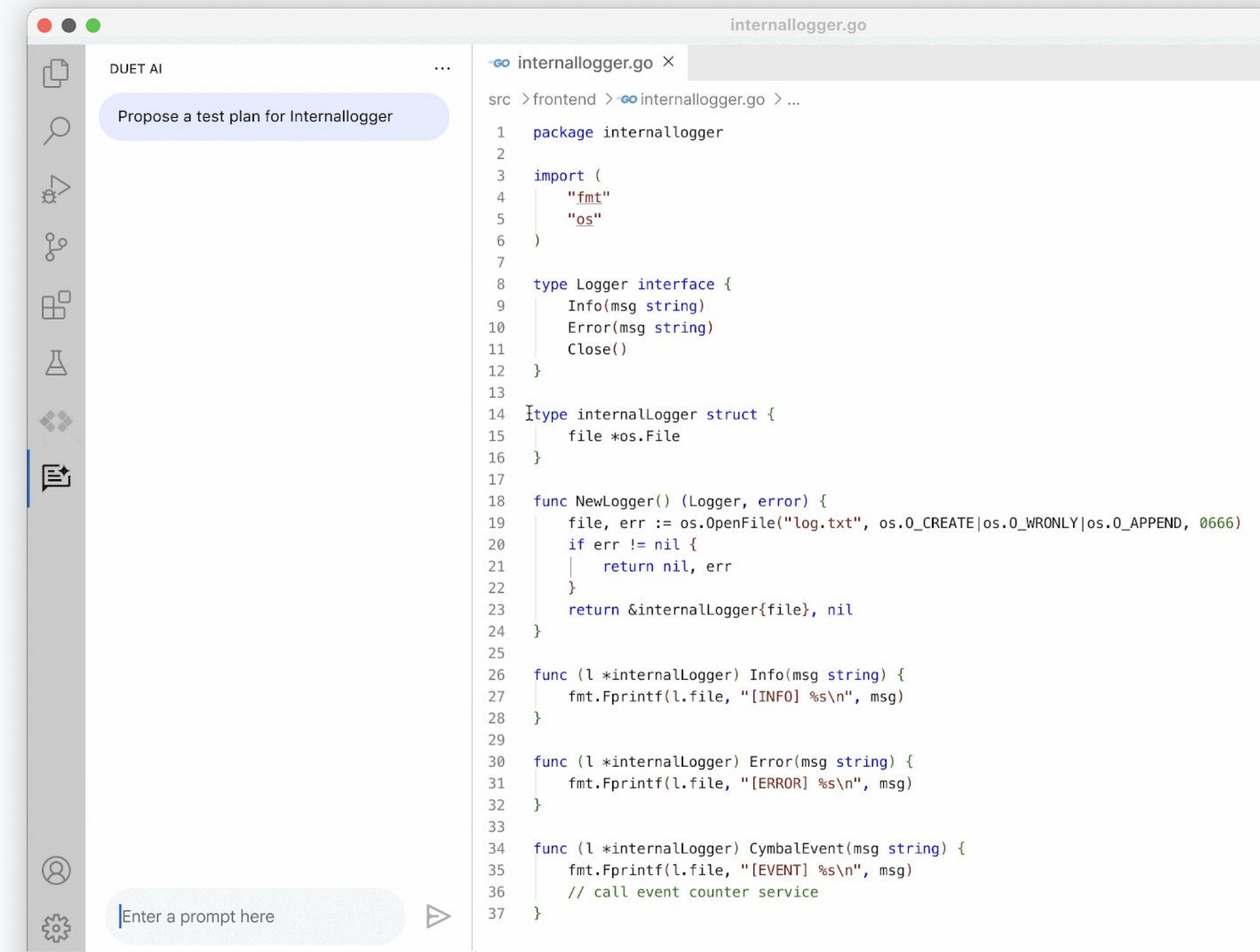
Devs are made aware when a suggestion falls under a permissive license and receive the information to provide license attribution, so their organization can stay compliant with license terms.

Real time code license flagging with recitation checking.

```
internallogger.go
internallogger.go x
src > frontend > internallogger.go > ...
1 package internallogger
2
3 import (
4     "fmt"
5     "os"
6 )
7
8 type Logger interface {
9     Info(msg string)
10    Error(msg string)
11    Close()
12 }
13
14 type internalLogger struct {
15     file *os.File
16 }
17
18 func NewLogger() (Logger, error) {
19     file, err := os.OpenFile("log.txt", os.O_CREATE|os.O_WRONLY|os.O_APPEND, 0666)
20     if err != nil {
21         return nil, err
22     }
23     return &internalLogger{file}, nil
24 }
25
26
27
28
29
30
31
32
33
34
35
36
37
```

# Test planning: Chat to suggest a test plan

Devs can ask Gemini Code Assist to **suggest a unit test plan** for the code, and iterate on it to ensure good coverage of edge cases.

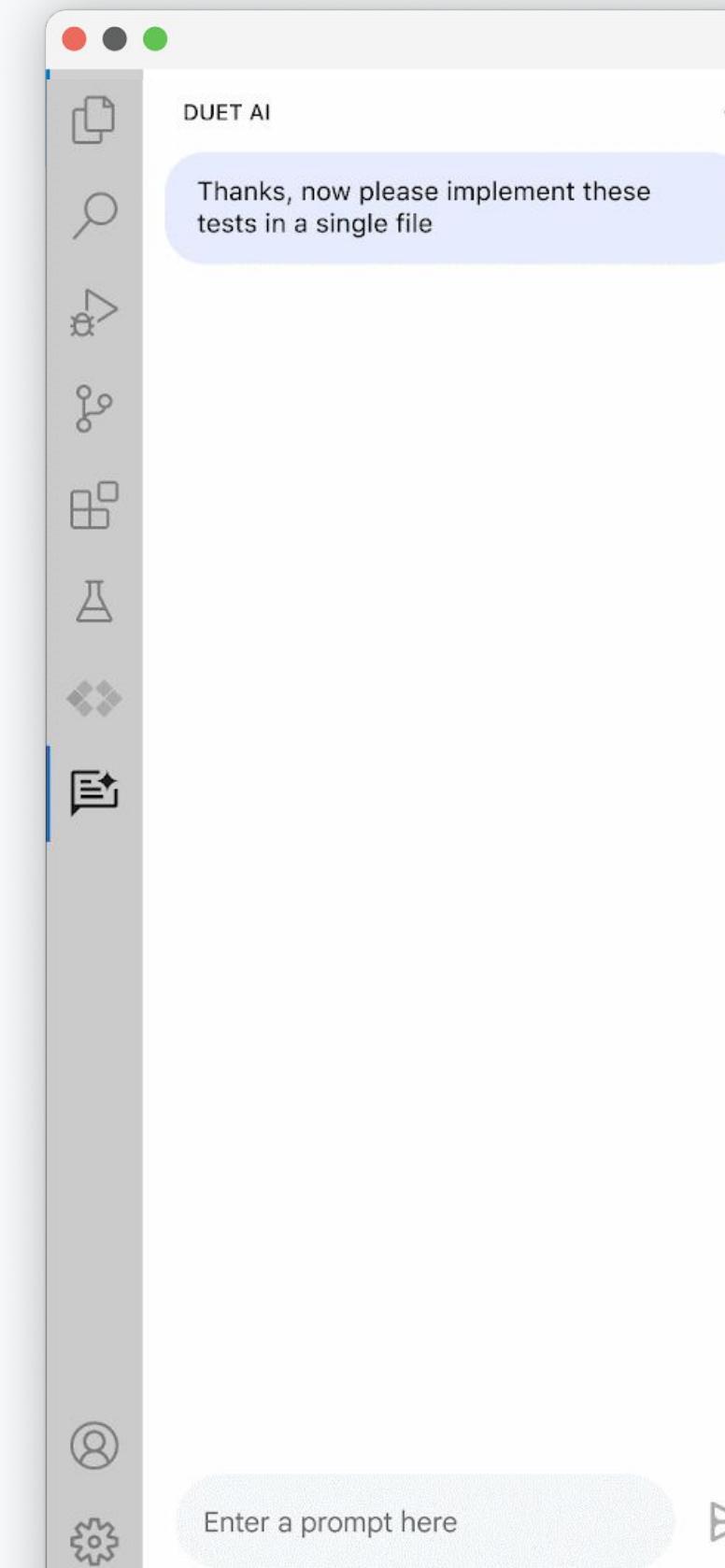


The screenshot shows the DUET AI interface. On the left is a sidebar with various icons for file operations like copy, search, and navigation. The main area has a title bar "DUET AI" and a button "...". A central input field contains the text "Propose a test plan for Internallogger". Below this is a "DUET AI" logo. At the bottom is a footer with a user icon and a gear icon, and a placeholder "Enter a prompt here". To the right is a code editor window titled "internallogger.go". The code is a Go program that defines a logger interface and an internal logger implementation. It includes functions for logging info, errors, and events to a file named "log.txt".

```
internallogger.go
internallogger.go X
src > frontend >-go internallogger.go > ...
1 package internallogger
2
3 import (
4     "fmt"
5     "os"
6 )
7
8 type Logger interface {
9     Info(msg string)
10    Error(msg string)
11    Close()
12 }
13
14 type internalLogger struct {
15     file *os.File
16 }
17
18 func NewLogger() (Logger, error) {
19     file, err := os.OpenFile("log.txt", os.O_CREATE|os.O_WRONLY|os.O_APPEND, 0666)
20     if err != nil {
21         return nil, err
22     }
23     return &internalLogger{file}, nil
24 }
25
26 func (l *internalLogger) Info(msg string) {
27     fmt.Fprintf(l.file, "[INFO] %s\n", msg)
28 }
29
30 func (l *internalLogger) Error(msg string) {
31     fmt.Fprintf(l.file, "[ERROR] %s\n", msg)
32 }
33
34 func (l *internalLogger) CymbalEvent(msg string) {
35     fmt.Fprintf(l.file, "[EVENT] %s\n", msg)
36     // call event counter service
37 }
```

# Test generation: Writing the tests

Once the test plan looks good,  
Devs can ask Gemini Code  
Assist to **implement the test  
plan defined**



The screenshot shows the DUET AI interface. On the left is a sidebar with various icons. In the center, a message from 'DUET AI' says: 'Thanks, now please implement these tests in a single file'. At the bottom, there's a text input field with the placeholder 'Enter a prompt here' and a send button. On the right, a code editor window titled 'internallogger.go' displays the following Go code:

```
internallogger.go
internallogger.go X
src > frontend > internallogger.go > ...
1 package internallogger
2
3 import (
4     "fmt"
5     "os"
6 )
7
8 type Logger interface {
9     Info(msg string)
10    Error(msg string)
11    Close()
12 }
13
14 type internalLogger struct {
15     file *os.File
16 }
17
18 func NewLogger() (Logger, error) {
19     file, err := os.OpenFile("log.txt", os.O_CREATE|os.O_WRONLY|os.O_APPEND, 0666)
20     if err != nil {
21         return nil, err
22     }
23     return &internalLogger{file}, nil
24 }
25
26 func (l *internalLogger) Info(msg string) {
27     fmt.Fprintf(l.file, "[INFO] %s\n", msg)
28 }
29
30 func (l *internalLogger) Error(msg string) {
31     fmt.Fprintf(l.file, "[ERROR] %s\n", msg)
32 }
33
34 func (l *internalLogger) CymbalEvent(msg string) {
35     fmt.Fprintf(l.file, "[EVENT] %s\n", msg)
36     // call event counter service
37 }
```

# Code translation

GEMINI: CHAT

Translate this into a react.js application

```
import React, { useState } from "react";
import { useHistory } from "react-router-dom";

const Login = () => {
  const [error, setError] = useState("");
  const [username, setUsername] = useState("");
  const [password, setPassword] = useState("");
  const history = useHistory();

  const users = {
    johndoe: {
      fullname: "John Doe",
      password: "password",
    },
    janedoe: {
      fullname: "Jane Doe",
      password: "password",
    }
  };
}
```

<> default.asp <

<> default.asp > 📁 ?

```
1  <%@LANGUAGE="VBSCRIPT" CODEPAGE="65001"%>
3  <%
15   Response.Redirect "editor.asp"
16   Response.End
17   Else
18   | error = "Invalid username/password"
19   End If
20   Else
21   | error = "Invalid username/password"
22   End If
23   End If
24 %>
25 <!DOCTYPE html>
26 <html lang="en">
27
28 <head>
29   <meta charset="utf-8">
30   <title>Login</title>
31   <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
32 </head>
33
```

# Refactoring

- Move the area calculation out the main method
- Remove unused code

Sure, here is the code with the area calculation moved out of the main method:

```
#include <stdio.h>
#include <math.h>

// Function to calculate the area of a circle
float calculate_area(float radius) {
    return M_PI * pow(radius, 2);
}

int main() {
    float radius, area;

    printf("Enter the radius of the circle: ");
    scanf("%f", &radius);

    // Calculate the area
    area = calculate_area(radius);

    // Print the area
    printf("The area of the circle is: %f\n", area);

    return 0;
}
```



# Documentation and Comments

Python: Add docstring comments

Java: add javadoc and inline  
comments

```
// A car parking lot system

import java.util.Scanner;

/**
 * This class simulates a car parking lot system.
 */
public class CarParkingLotSystem {

    /**
     * The main method of the program.
     *
     * @param args The command line arguments.
     */
    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in); // Create a scanner object

        // Initialize the parking lot with 100 spaces.
        int numSpaces = 100;
        int[] parkingLot = new int[numSpaces];
        for (int i = 0; i < numSpaces; i++) {
            parkingLot[i] = 0; // Set all spaces to empty.
        }
    }
}
```

# Generate sample data

Generate a JSON object which has list of inventory items. Each object has two attributes: productid and onhandty.

Both are strings.

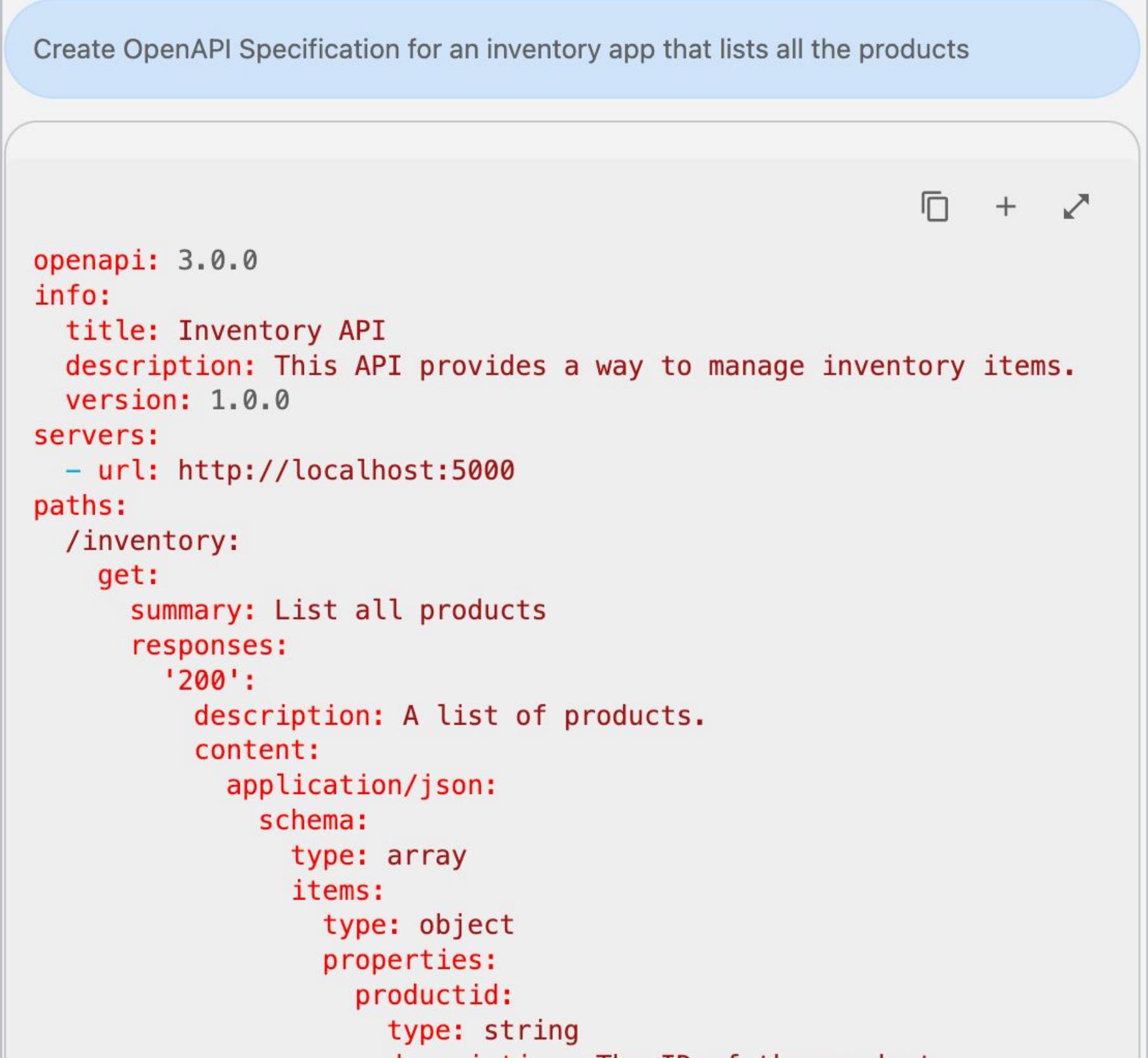
Generate three sample inventory objects.

Assign these three samples to the "inventory" variable.

```
inventory = [  
    {"productid": "12345", "onhandty": "10"},  
    {"productid": "67890", "onhandty": "20"},  
    {"productid": "11223", "onhandty": "30"},  
]
```

# Generate OpenAPI Specifications

Create OpenAPI Specification for an inventory app that lists all the products



The screenshot shows a code editor window with a light gray background. At the top, there is a blue header bar with the text "Create OpenAPI Specification for an inventory app that lists all the products". Below the header, there are three small icons: a square with a minus sign, a plus sign, and a double-headed arrow. The main content area contains the following JSON code:

```
openapi: 3.0.0
info:
  title: Inventory API
  description: This API provides a way to manage inventory items.
  version: 1.0.0
servers:
  - url: http://localhost:5000
paths:
  /inventory:
    get:
      summary: List all products
      responses:
        '200':
          description: A list of products.
          content:
            application/json:
              schema:
                type: array
                items:
                  type: object
                  properties:
                    productid:
                      type: string
                      description: The ID of the product.
```

# Generate a Data Model

Using the python sqlalchemy ORM, generate a data model for this API service. Also include a separate function and a main entrypoint that creates the database tables.

```
from sqlalchemy import Column, Integer, Float, String
from sqlalchemy.ext.declarative import declarative_base

Base = declarative_base()

class PackageInfo(Base):
    __tablename__ = 'package_info'

    product_id = Column(Integer, primary_key=True)
    height = Column(Float)
    width = Column(Float)
    depth = Column(Float)
    weight = Column(Float)
    special_handling_instructions = Column(String)

def create_tables(engine):
    Base.metadata.create_all(engine)
```

openapi: 3.0.0  
info:  
 title: Shipping and Package Information Service  
 description: This service provides shipping and package information.  
 version: 1.0.0  
servers:  
 - url: <https://shipping-package-info.example.com>  
paths:  
 /shipping-info/{product\_id}:  
 get:  
 summary: Get shipping and package info  
 parameters:  
 - name: product\_id  
 in: path  
 required: true  
 description: The numerical ID of the package.  
 schema:  
 type: integer  
 responses:  
 '200':  
 description: OK  
 content:  
 application/json:  
 schema:

# Write regular expressions

```
Regex.py > getPlayerIP
1 # Function to prase input string and get player number
2 # Input String: "20:41:05.442 player 101 (::ffff:127.0.0.1) connected"
3 # Output Player ID: 101
4
5 import re
6
7 def getPlayerID(inputString):
8     match = re.search(r'player (\d+)', inputString)
9     if match:
10         return match.group(1)
11     else:
12         return None
13
14 def getPlayerIP(inputString):
15     match = re.search(r'(\d{1,3}.\d{1,3}.\d{1,3}.\d{1,3})', inputString)
16     if match:
17         return match.group(1)
18     else:
19         return None
20
```

# Create an initial README.md structure

**Duet Demo Application**

This application is a simple Flask application that demonstrates how to use the Flask-SQLAlchemy library to connect to a PostgreSQL database and perform CRUD operations on a table of products.

**Installation**

To install the required libraries, run the following command:

```
pip install flask flask-sqlalchemy flask-cors
```

**Running the Application**

To run the application, run the following command:

```
python app_main.py
```

The application will listen on port 5000 by default. You can change the port by setting the **PORT** environment variable.

**Testing the Application**

To test the application, you can use the following curl commands:

```
curl http://localhost:5000/inventory
```

This command should return a list of all products in the inventory.

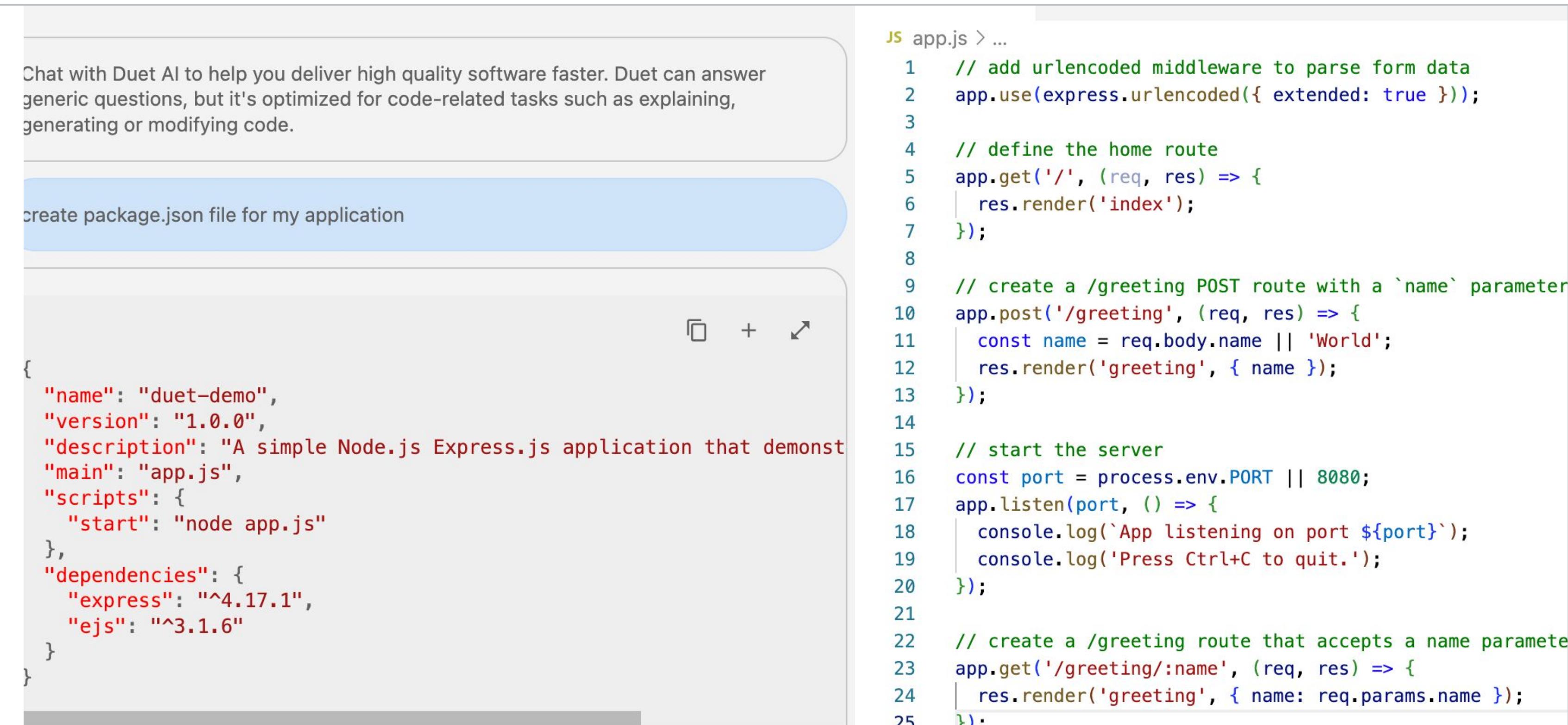
app-py > app\_main.py > ...

```

2  # the app serves following routes
3  # /inventory
4  # /inventory/<productid>#
5
6  # Import the necessary modules.
7  from flask import Flask, jsonify, request
8  from flask_cors import CORS
9  from flask_sqlalchemy import SQLAlchemy
10 import os
11
12 # Create a Flask app.
13 app = Flask(__name__)
14
15 # Define the /inventory route.
16 @app.route('/inventory')
17 def inventory():
18     """Return a list of all products in the inventory."""
19
20     # Get the list of products from the database.
21     products = get_products()
22
23     # Return the list of products as a JSON response.
24     return jsonify(products)
25
26 # Define the /inventory/<productid> route.
27 @app.route('/inventory/<productid>')
28 def product(productid):
29     """Return the details of a specific product."""
30
31     # Get the product from the database.
32     product = get_product(productid)
33
34     # Return the product details as a JSON response.
35     return jsonify(product)
36
37

```

# Create package.json or requirements.txt files



Chat with Duet AI to help you deliver high quality software faster. Duet can answer generic questions, but it's optimized for code-related tasks such as explaining, generating or modifying code.

create package.json file for my application

```
{  
  "name": "duet-demo",  
  "version": "1.0.0",  
  "description": "A simple Node.js Express.js application that demonst  
  "main": "app.js",  
  "scripts": {  
    "start": "node app.js"  
  },  
  "dependencies": {  
    "express": "^4.17.1",  
    "ejs": "^3.1.6"  
  }  
}
```

JS app.js > ...

```
1 // add urlencoded middleware to parse form data  
2 app.use(express.urlencoded({ extended: true }));  
3  
4 // define the home route  
5 app.get('/', (req, res) => {  
  | res.render('index');  
7});  
8  
9 // create a /greeting POST route with a `name` parameter  
10 app.post('/greeting', (req, res) => {  
11  const name = req.body.name || 'World';  
12  res.render('greeting', { name });  
13});  
14  
15 // start the server  
16 const port = process.env.PORT || 8080;  
17 app.listen(port, () => {  
18  console.log(`App listening on port ${port}`);  
19  console.log('Press Ctrl+C to quit.');  
20});  
21  
22 // create a /greeting route that accepts a name parameter  
23 app.get('/greeting/:name', (req, res) => {  
24  | res.render('greeting', { name: req.params.name });  
25});
```

# Create Dockerfiles

create a dockerfile for my application that expose port 5000. Include DB\_USER="user1" and DB\_PASSWORD="pass123" enviromnet variables. there is requirements.txt file present in same directory as application

```
FROM python:3.8

WORKDIR /usr/src/app

COPY requirements.txt .
RUN pip install --no-cache-dir -r requirements.txt

COPY .

ENV DB_USER user1
ENV DB_PASSWORD pass123

EXPOSE 5000

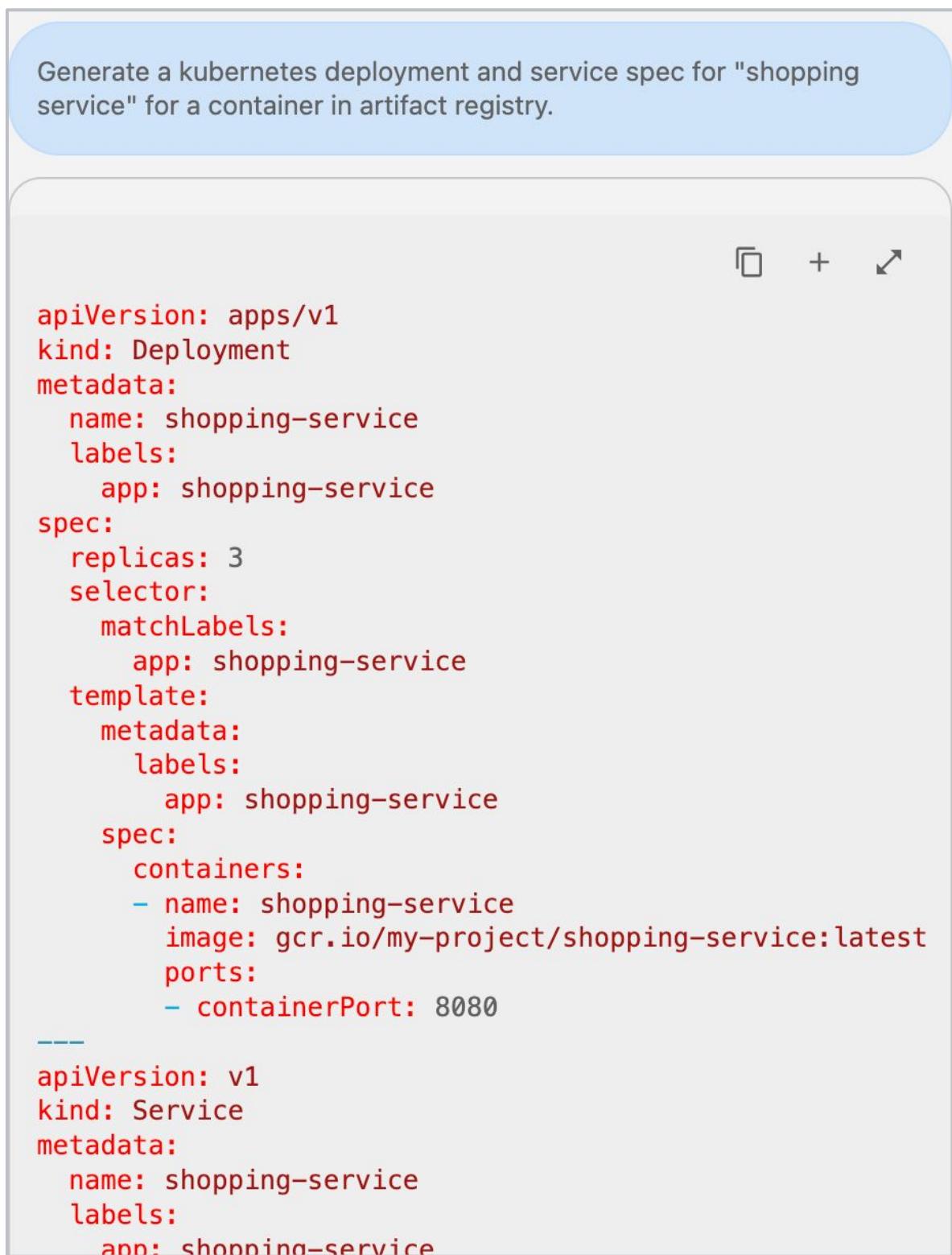
CMD ["python", "app_main.py"]
```

Rate this answer:

```
9  from flask import Flask, jsonify, request
10 import os
11
12 # Create a Flask app.
13 app = Flask(__name__)
14
15 # Define the /inventory route.
16 @app.route('/inventory')
17 def inventory():
18     """Return a list of all products in the inventory."""
19
20     # Get the list of products from the database.
21     products = get_products()
22
23     # Return the list of products as a JSON response.
24     return jsonify(products)
25
26 # Define the /inventory/<productid> route.
27 @app.route('/inventory/<productid>')
28 def product(productid):
29     """Return the details of a specific product."""
30
31     # Get the product from the database.
32     product = get_product(productid)
33
34     # Return the product details as a JSON response.
35     return jsonify(product)
36
```

# Create Kubernetes Specs

Generate a kubernetes deployment and service spec for "shopping service" for a container in artifact registry.



```
# Kubernetes deployment with two replicas and service named "shopping"
apiVersion: apps/v1
kind: Deployment
metadata:
  name: shopping
spec:
  replicas: 2
  selector:
    matchLabels:
      app: shopping
  template:
    metadata:
      labels:
        app: shopping
    spec:
      containers:
        - name: shopping
          image: gcr.io/my-project/shopping:latest
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: shopping
  labels:
    app: shopping
```

```
# Kubernetes deployment with two replicas and service named "shopping"
apiVersion: apps/v1
kind: Deployment
metadata:
  name: shopping
spec:
  replicas: 2
  selector:
    matchLabels:
      app: shopping
  template:
    metadata:
      labels:
        app: shopping
    spec:
      containers:
        - name: shopping
          image: gcr.io/my-project/shopping:latest
          ports:
            - containerPort: 8080
---
apiVersion: v1
kind: Service
metadata:
  name: shopping
spec:
```

# Help architect GCP infrastructure needs and provide gcloud commands...

To build a private build environment that is not connected to the internet in Google Cloud, you can use Cloud Build private pools. Private pools are isolated environments that allow you to build container images without exposing them to the public internet. This can be useful for building images that contain sensitive data or that need to be built in a secure environment.

To create a private pool, you can use the `gcloud` command-line tool. First, you need to create a service account that will be used to access the private pool. You can do this with the following command:

```
gcloud iam service-accounts create my-private-pool-sa
```

Next, you need to create a private pool. You can do this with the following command:

```
gcloud beta builds pools create my-private-pool \
--location=us-central1 \
--worker-pool=my-worker-pool \
--service-account=my-private-pool-sa@PROJECT_ID.iam.gserviceaccount.c
```

Once you have created a private pool, you can use it to build container images. You can do this with the following command:

# Or create infrastructure with Terraform files

```
infra.tf 7 ●  
utils > infra.tf > ...  
1 # this file creates terraform resources for the following:  
2 # 1. VPC  
3 # 2. Subnet  
4 # 3. GKE cluster named "Duet" in the subnet  
5 # 4. Service account for the GKE cluster  
6 # 5. IAM role binding for the service account  
7  
8 # Import the Google Cloud provider  
9 provider "google" {  
10   project = var.project_id  
11   region  = var.region  
12   zone    = var.zone  
13 }  
14  
15 # Create a VPC network  
16 resource "google_compute_network" "vpc" {  
17   name = "vpc-duet"  
18   auto_create_subnetworks = false  
19 }  
20
```

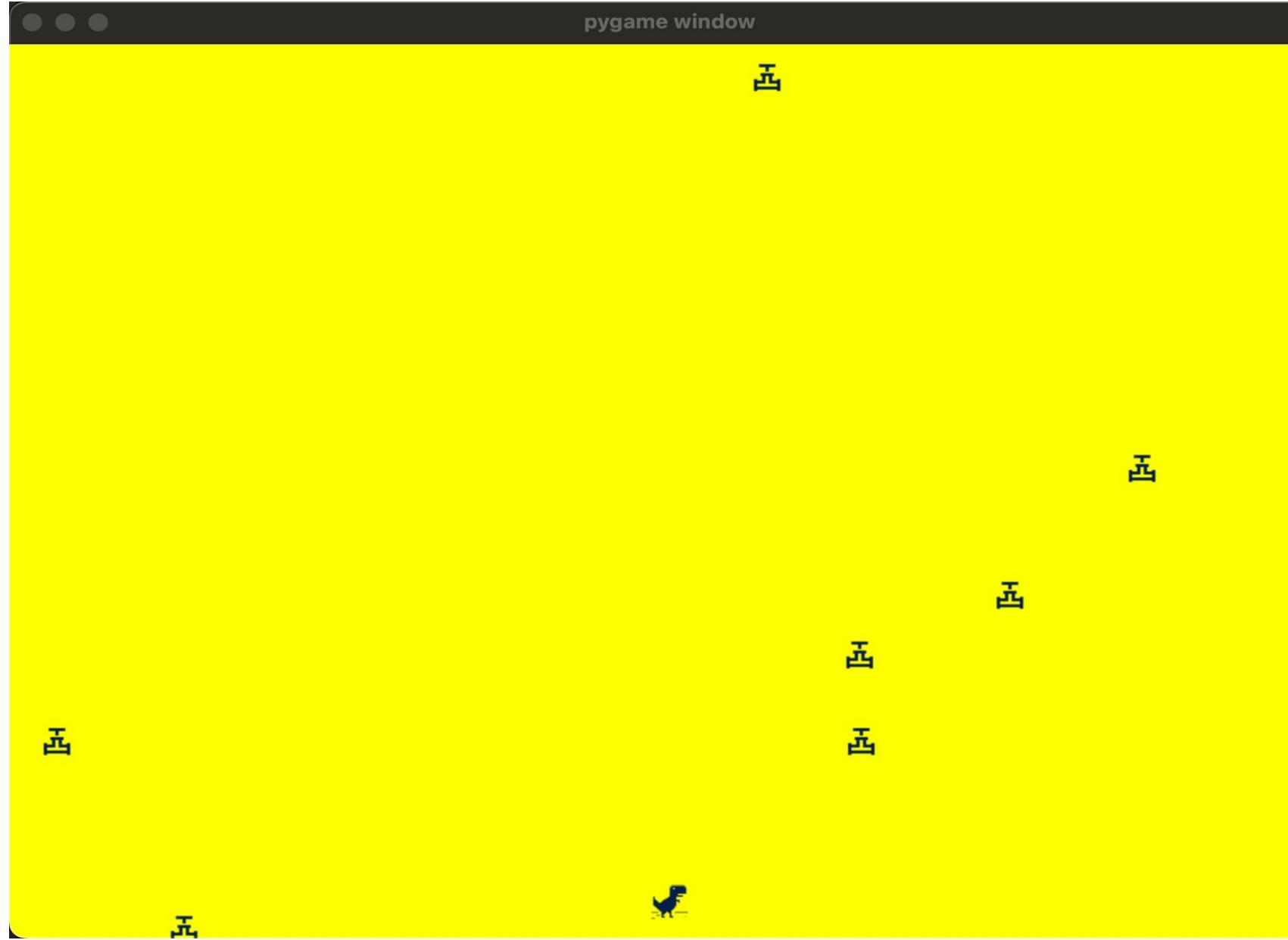
# Implement test-driven design

Write me the code that would make these tests pass.



```
def convert_fahrenheit_to_celsius(fahrenheit):  
    """Convert fahrenheit to celsius"""  
    return (fahrenheit - 32) * 5 / 9
```

# Generate small games and interactive experiences



```
game.py 1 ×  
game-py > game.py > ...  
1 import pygame  
2 import random  
3  
4 # Initialize the game engine  
5 pygame.init()  
6  
7 # Create the game screen  
8 screen = pygame.display.set_mode(800, 600)  
9 screen.fill((255, 255, 0))  
10  
11 # Load the assets  
12 dino_image = pygame.image.load("dino.png")  
13 obstacle_image = pygame.image.load("obstacle.png")  
14  
15 # Create the dino object  
16 dino = pygame.sprite.Sprite()  
17 dino.image = dino_image  
18 dino.rect = dino.image.get_rect()  
19 dino.rect.center = (400, 500)  
20  
21 # Create the obstacle objects  
22 obstacles = []  
23 for i in range(7):  
24     obstacle = pygame.sprite.Sprite()  
25     obstacle.image = obstacle_image  
26     obstacle.rect = obstacle.image.get_rect()  
27     obstacle.rect.center = (random.randint(0, 800), random.randint(0, 600))  
28     obstacles.append(obstacle)  
29  
30 # Create the sprite group  
31 sprite_group = pygame.sprite.Group()  
32 sprite_group.add(dino)  
33 for obstacle in obstacles:
```

# Topics

- 01 Google and Gemini's Approach to Developer Acceleration
- 02 Where can you use Gemini Code Assist?
- 03 Enabling Gemini Code Assist
- 04 Example IDE Setup: VS Code
- 05 How to Interact with Gemini Code Assist
- 06 Common dev-related tasks for Gemini or Gemini Code Assist
- 07 Prompt Guidance



# Write your prompt like you're talking with a person

Don't enter only keywords as you might when searching for a document online, but include the kind of detail you'd include when explaining a problem to someone. For example, instead of entering the terms "workload GKE," ask a complete question:

"what kinds of workloads does GKE support?"

# Tell Gemini Code Assist why you're trying to accomplish a task

Details about what you're trying to accomplish can help Gemini Code Assist provide a more useful answer.

For example, telling Gemini Code Assist that you want to "set up a simple, secure Google Cloud site for hosting a blog" provides Gemini Code Assist more helpful details than only asking "how to deploy a web site." Because there are often multiple correct ways to accomplish a technical goal, providing Gemini Code Assist sufficient context can help ensure a good response.

# Tell Gemini Code Assist to include all the parameters in a command

For example, when you ask Gemini Code Assist to generate code functions, you are likely to produce a more useful and detailed response by telling it to:

"make sure all methods use their required arguments."

# Include your level of expertise

It's especially helpful to match your prompt request with your expertise level when you ask for suggestion. Asking Gemini Code Assist to explain a concept or code as if you're an expert programmer—or a novice programmer—can give you different, and more appropriate, results.

For example, to direct Gemini Code Assist to respond from an expert context, you could prompt it with:

"create Python code to list all Compute Engine instances. You are an expert software developer using Google Cloud."

Similarly, you could ask Gemini Code Assist to:

"explain Kubernetes and its benefits to me in the simplest terms possible."

# Include details about products and technologies

If you're looking for answers about a specific product, technology, or technical capability, include that in your prompt. Similarly, specifying a programming language can help you get more relevant responses. If you're unsure about what technologies or products to consider, ask Gemini Code Assist to compare different options for you.

# Break complex problems into multiple requests

Writing separate prompts can help Gemini Code Assist refine and focus the answers that it gives, helping you move progressively toward a solution.

# Include schema for database queries

Include your schema.

If some fields may be referred by several names, you may want to add mention of that, even if you create additional metadata fields outside of the normal schema fields.

# An example code generation prompt

## Prompt

Generate **Python code** to create a simple **class representing a bank account**.

**Functionality:** Define a class `BankAccount` with attributes like `balance`, `account holder name`, and methods to `deposit`, `withdraw`, and `check balance`.

**Inputs:** Initial balance and account holder name during object creation, amount for deposit/withdrawal.

**Outputs:** Updated balance after transactions.

**Constraints:** Ensure sufficient balance for withdrawals and handle invalid input amounts.

# Troubleshooting



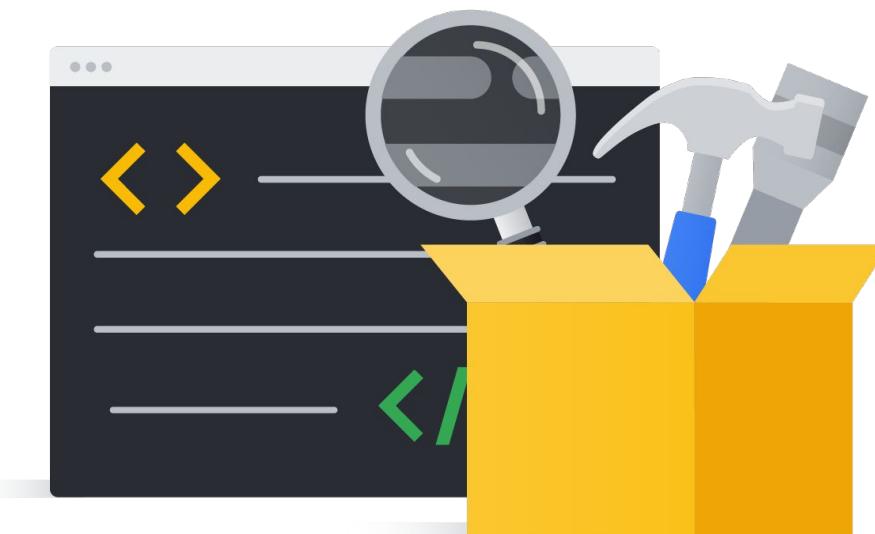
Clear the chat history and start fresh



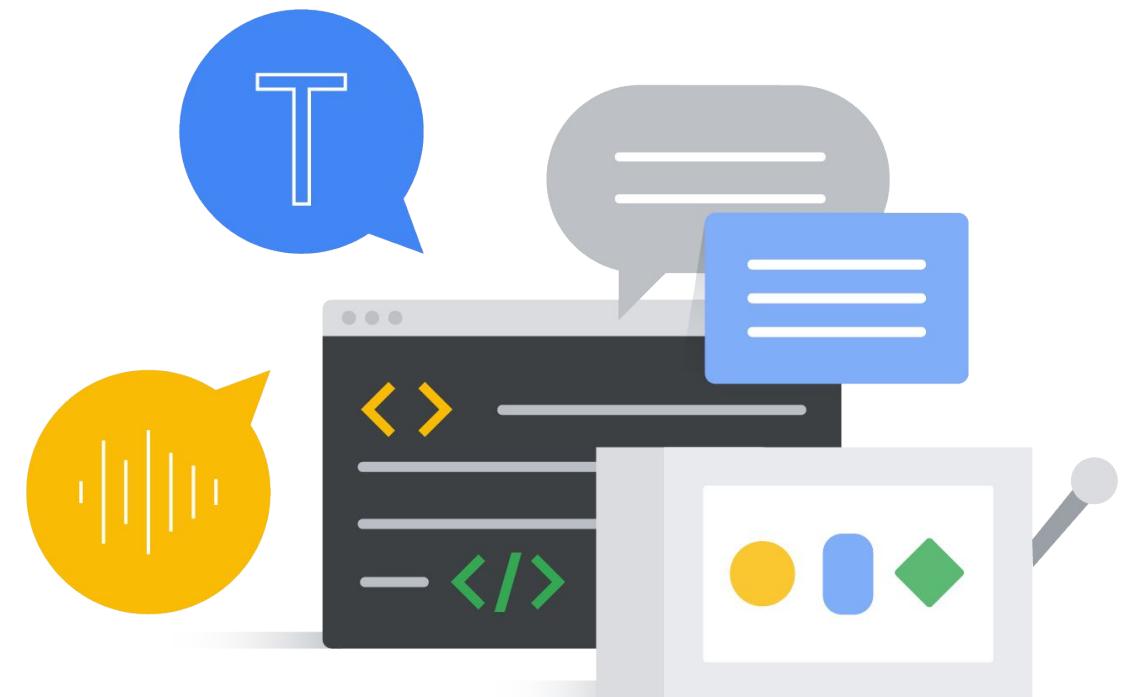
Review your open files in the editor and unselect any highlighted text



Ensure all open files belong to the codebase you're reviewing



Remember, practice makes perfect.  
Experiment with different prompts to  
observe how Gemini responds.



# Use Gemini to search the Google Cloud documentation

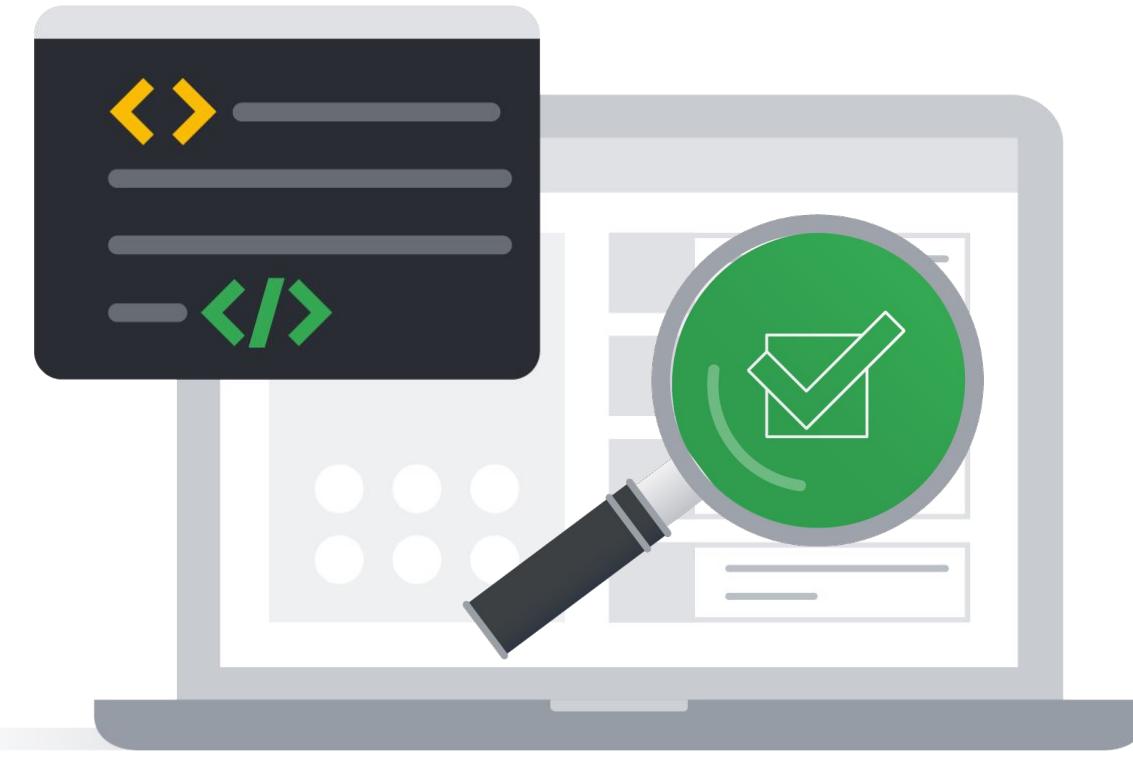
Gemini   

What should you use for a primary key in a Spanner Database

A primary key is a unique identifier for each row in a table. It is used to identify rows quickly and efficiently. In Spanner, you can use the following data types for a primary key:

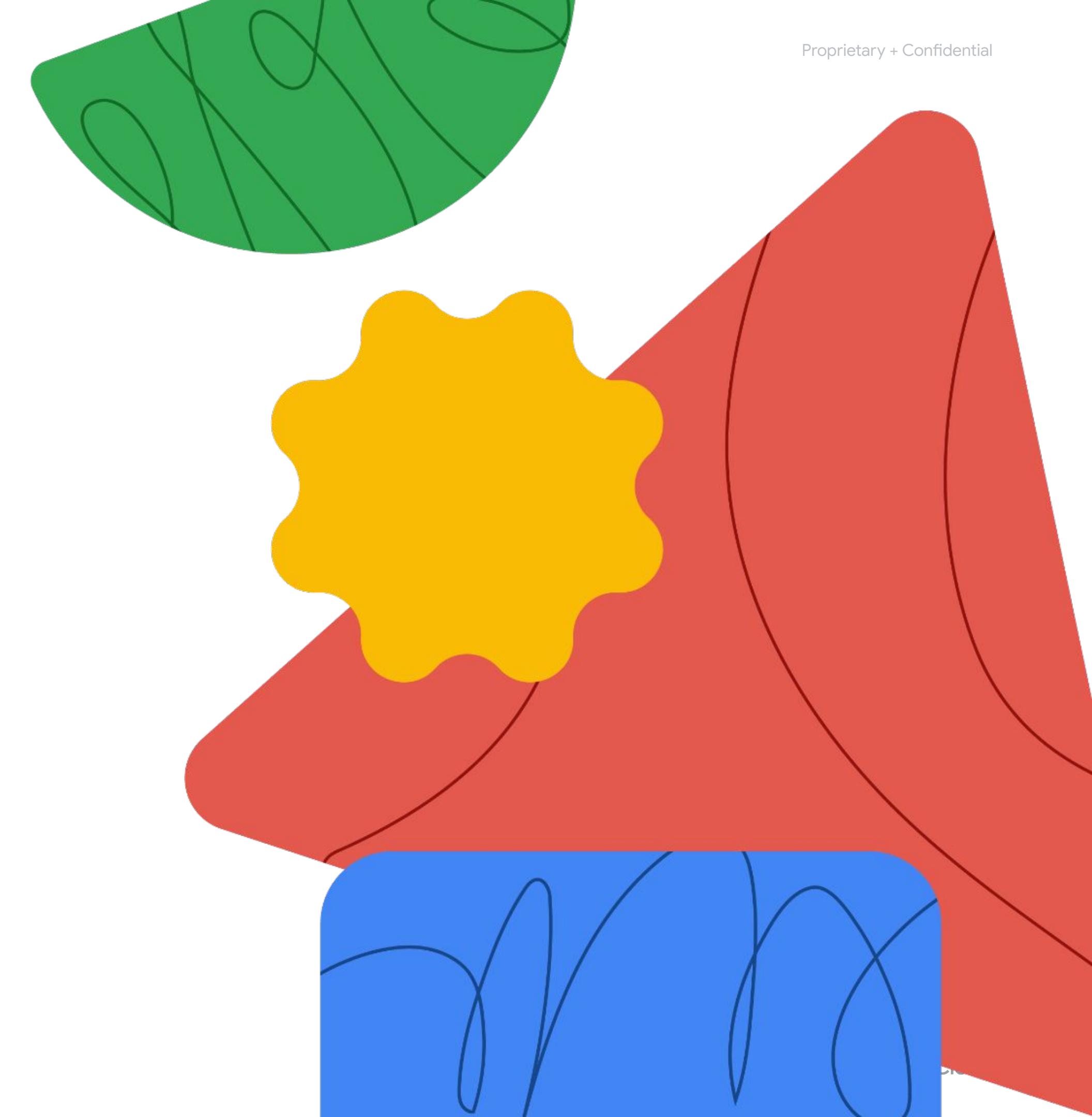
- INT64
- STRING
- BYTES
- ARRAY<INT64>
- ARRAY<STRING>
- ARRAY<BYTES>

The best data type to use for a primary key depends on the specific requirements of your application. However, there are some general guidelines that you can follow:



**Important!**  
**Be sure to carefully  
review all code responses**

# Resources



## Documentation:

- [Gemini Code Assist Product Page](#)
- [Gemini Code Assist Documentation](#)
- [Guide to set up the Gemini Code Assist subscription & project](#)
- [Guide to set up Gemini Code Assist in an IDE](#)
- [Supported languages and interfaces](#)

## Demos and Tutorials:

- [Code with Gemini Code Assist | Cloud Code for VS Code](#)
- [Develop a Python app with Gemini for Google Cloud assistance](#)
- [Analyze data with Gemini for Google Cloud assistance](#)

## Videos:

- [NEXT '24 - Get developer assistance customized to your organization code with Gemini](#)

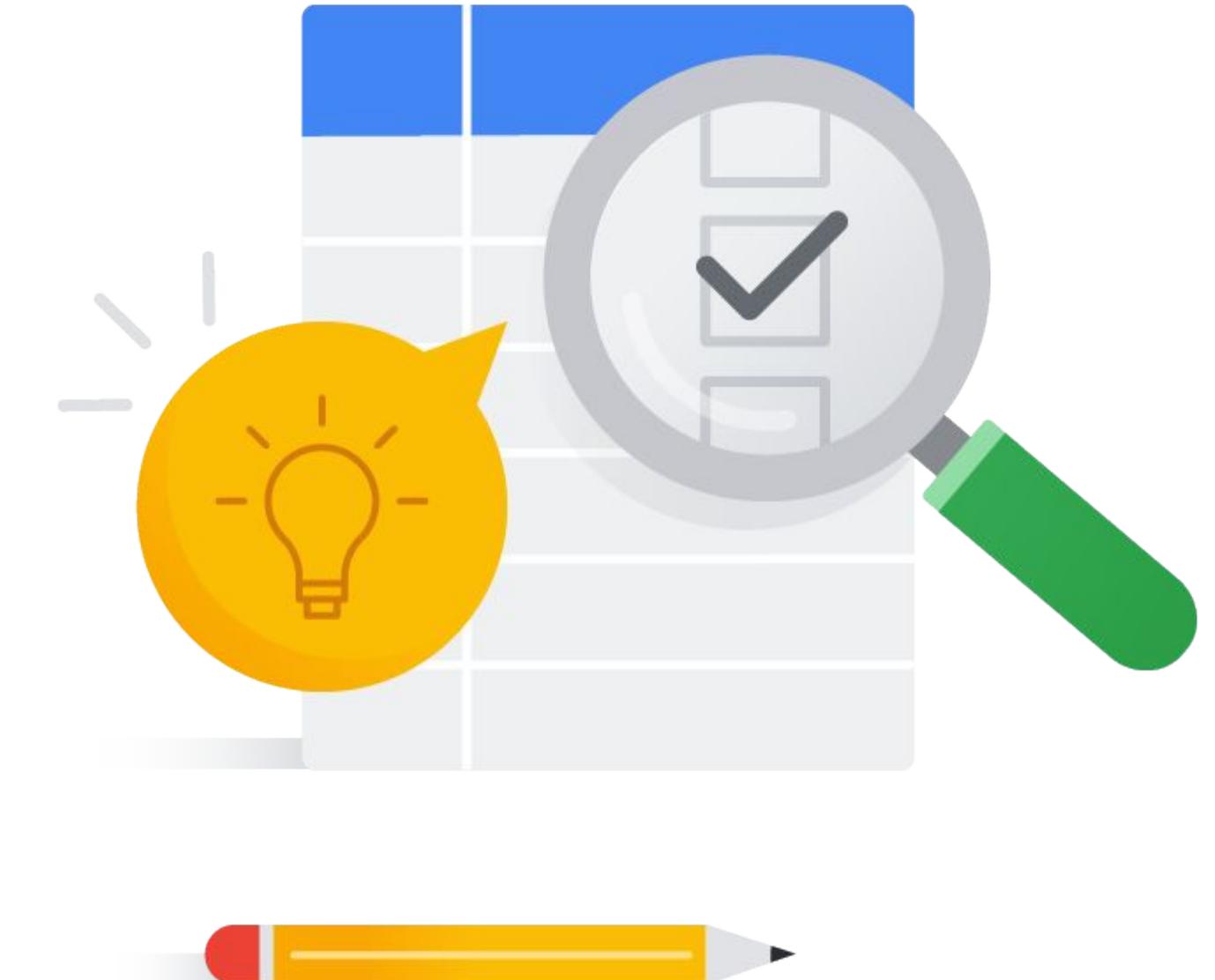
## Pricing:

- [Pricing - Gemini for Google Cloud](#)

# Lab



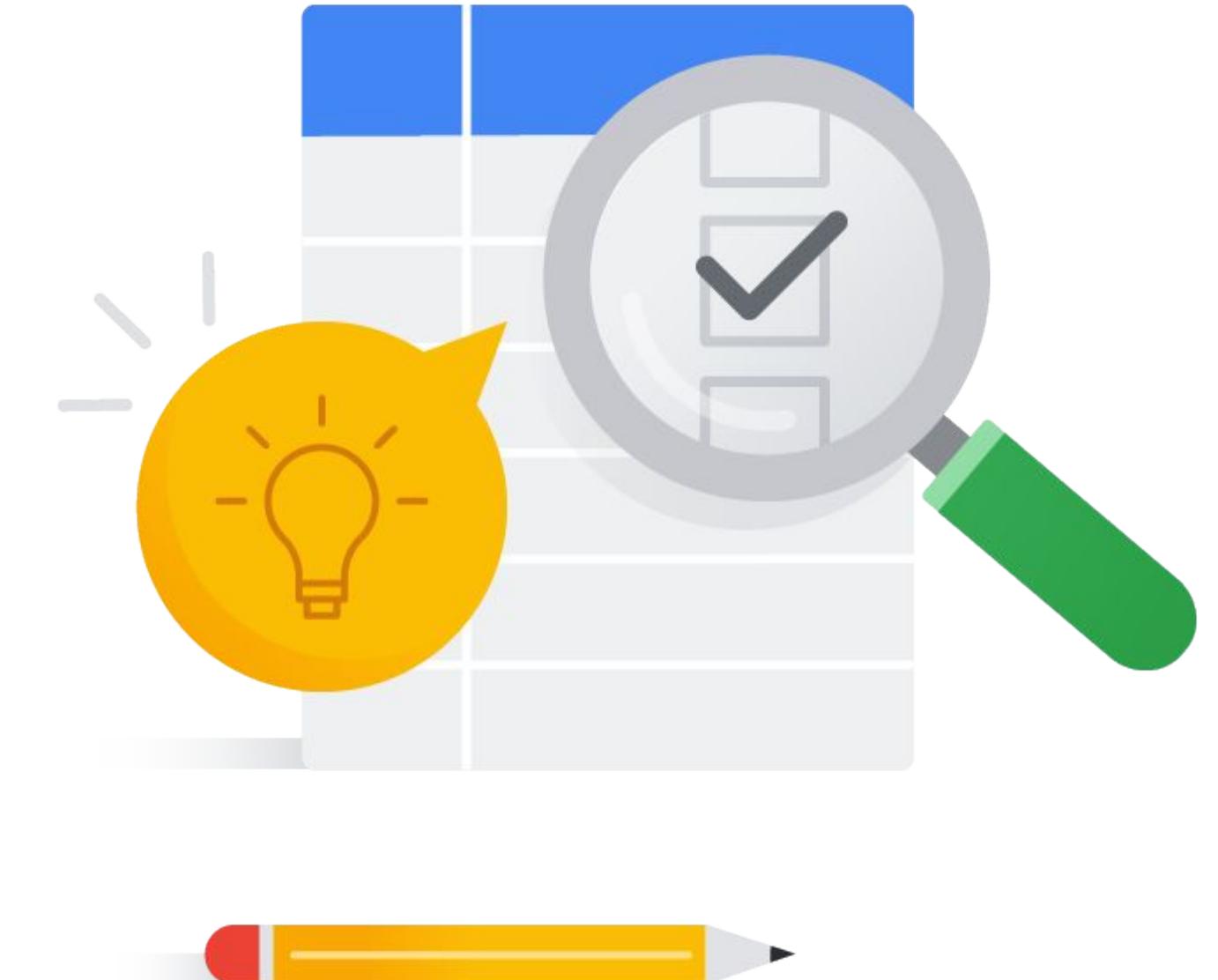
## Lab: Code Generation with Gemini



# Lab

1 hour

Lab: Code Identification and Explanation  
with Gemini



# In this module, you learned to ...

01

Add code generation to your developer workflow

02

Use generative AI to document, debug, test, optimize, and better understand your code

03

Run BigQuery queries based on natural language

04

Use Gemini within the Google Cloud Console to simplify your cloud projects



# Questions and answers



# Quiz question

List some coding-related use cases for generative AI and Codey:

---

---

---

# Quiz question

List some coding-related use cases for generative AI and Codey:

- Code generation
- Documentation and comments
- Testing
- Code Optimization
- Debugging
- Code explanations
- Code conversion

# Quiz question

To generate SQL from natural language accurately, what would you need to do?  
(Choose all that apply)

- A: Install the Gemini plug-in into your favorite database
- B: Supply the model with clear instructions
- C: Migrate your databases to BigQuery
- D: Include the database schema when requesting the SQL

# Quiz question

To generate SQL from natural language accurately, what would you need to do?  
(Choose all that apply)

- A: Install the Gemini plug-in into your favorite database
- B: Supply the model with clear instructions
- C: Migrate your databases to BigQuery
- D: Include the database schema when requesting the SQL

# Quiz question

Which of the following programming languages are supported by Codey (Choose all that apply)

- A: Python
- B: Java
- C: JavaScript
- D: SQL
- E: C#
- F: Go
- G: Terraform

# Quiz question

Which of the following programming languages are supported by Codey (Choose all that apply)

- A: Python
- B: Java
- C: JavaScript
- D: SQL
- E: C#
- F: Go
- G: Terraform

**Google** Cloud