Gemini 3 Pro 现已问世。在 Google AI Studio 中免费试用
(https://aistudio.google.com?model=gemini-3-pro-preview&authuser=1&hl=zh-cn)。

Gemini 3 开发者指南

Gemini 3 是我们迄今为止最智能的模型系列，建立在先进的推理技术基础上。它旨在通过掌握智能体工作流、自主编码和复杂的多模态任务，将任何想法变为现实。本指南介绍了 Gemini 3 模型系列的主要功能，以及如何充分利用这些功能。

免费试用 Gemini 3 Pro (https://aistudio.google.com?model=gemini-3-pro-preview&authuser=1&hl=zh-cn)

不妨探索我们的 Gemini 3 应用合集
(https://aistudio.google.com/app/apps?source=showcase&%3BshowcaseTag=gemini-3&authuser=1&hl=zh-cn)
，了解该模型如何处理高级推理、自主编码和复杂的多模态任务。

只需编写几行代码，即可开始使用：

PythonJavaScript (#javascript)REST (#rest)
(#python)

```
from google import genai

client = genai.Client()

response = client.models.generate_content(
    model="gemini-3-pro-preview",
    contents="Find the race condition in this multi-threaded C++ snippet: [c
)

print(response.text)
```

认识一下 Gemini 3

Gemini 3 Pro 是新系列中的首款模型。`gemini-3-pro-preview` 最适合需要广泛的世界知识和跨模态的高级推理的复杂任务。

模型 ID	上下文窗口（输入 / 输出）	知识截点	定价（输入 / 输出）*
gemini-3-pro-preview	100 万 / 6.4 万	2025 年 1 月	2 美元 / 12 美元（<20 万个 token） 4 美元 / 18 美元（>20 万个 token）
gemini-3-pro-image-preview	65k / 32k	2025 年 1 月	$2（文本输入）/ $0.134（图片输出）**

** 除非另有说明，否则价格按每 100 万个令牌计算。 ** 图片价格因分辨率而异。如需了解详情，请参阅价格页面* (https://ai.google.dev/gemini-api/docs/pricing?authuser=1&hl=zh-cn)。

如需详细了解速率限制、批量价格和其他信息，请参阅模型页面 (https://ai.google.dev/gemini-api/docs/models/gemini?authuser=1&hl=zh-cn)。

Gemini 3 中的新 API 功能

Gemini 3 引入了新的参数，旨在让开发者更好地控制延迟时间、费用和多模态保真度。

思考等级

Gemini 3 Pro 默认使用动态思考功能来推理提示。您可以使用 `thinking_level` 参数，该参数用于控制模型在生成回答之前内部推理过程的**最大**深度。Gemini 3 将这些级别视为相对的思考许可，而不是严格的令牌保证。

如果未指定 `thinking_level`，Gemini 3 Pro 将默认为 `high`。如果不需要复杂的推理，您可以将模型的思维水平限制为 `low`，以便更快地获得低延迟的回答。

- `low`：尽可能缩短延迟时间并降低费用。最适合简单指令遵循、聊天或高吞吐量应用
- `medium`：目前不支持
- `high`（默认）：最大限度地提高推理深度。模型可能需要更长时间才能生成第一个 token，但输出结果会经过更仔细的推理。

Python JavaScript (#javascript) REST (#rest)
(#python)

```
from google import genai
from google.genai import types

client = genai.Client()
```

```
response = client.models.generate_content(
    model="gemini-3-pro-preview",
    contents="How does AI work?",
    config=types.GenerateContentConfig(
        thinking_config=types.ThinkingConfig(thinking_level="low")
    ),
)

print(response.text)
```

重要提示： 您不能在同一请求中同时使用 `thinking_level` 和旧版 `thinking_budget` 参数。这样做会返回 400 错误。

媒体分辨率

Gemini 3 通过 `media_resolution` 参数引入了对多模态视觉处理的精细控制。分辨率越高，模型读取细小文字或识别细微细节的能力就越强，但会增加 token 使用量和延迟时间。`media_resolution` 参数用于确定**为每个输入图片或视频帧分配的 token 数量上限**。

现在，您可以为每个媒体部分单独设置分辨率，也可以通过 `generation_config` 全局设置分辨率，分辨率可设置为 `media_resolution_low`、`media_resolution_medium` 或 `media_resolution_high`。如果未指定，模型会根据媒体类型使用最佳默认值。

推荐设置

媒体类型	推荐设置	令牌数量上限	使用指南
图片	`media_resolution_high`	1120	建议用于大多数图片分析任务，以确保获得最佳质量。
PDF	`media_resolution_medium`	560	非常适合文档理解；质量通常在 `medium` 时达到饱和。将该值增加到 `high` 很少能提高标准文档的 OCR 结果。
视频（常规）	`media_resolution_low`（或 `media_resolution_medium`）	70（每帧）	注意：对于视频，`low` 和 `medium` 设置的处理方式相同（70 个令牌），以优化上下文使用情况。这对于大多数动作识别和描述任务来说已经足够。
视频（文字较多）	`media_resolution_high`	280（每帧）	仅当用例涉及读取密集文本 (OCR) 或视频帧中的微小细节时才需要。

注意： **`media_resolution`** 参数会根据输入类型映射到不同的令牌数量。虽然图片是线性缩放的（**`media_resolution_low`**：280，**`media_resolution_medium`**：560，**`media_resolution_high`**：1120），但视频的压缩程度更高。对于视频，**`media_resolution_low`** 和 **`media_resolution_medium`** 的上限均为每帧 70 个词元，**`media_resolution_high`** 的上限为 280 个词元。点击此处 (https://ai.google.dev/gemini-api/docs/media-resolution?authuser=1&hl=zh-cn#token-counts)

查看完整详情

PythonJavaScript (#javascript)REST (#rest)
(#python)

```
from google import genai
from google.genai import types
import base64

# The media_resolution parameter is currently only available in the v1alpha A
client = genai.Client(http_options={'api_version': 'v1alpha'})

response = client.models.generate_content(
    model="gemini-3-pro-preview",
    contents=[
        types.Content(
            parts=[
                types.Part(text="What is in this image?"),
                types.Part(
                    inline_data=types.Blob(
                        mime_type="image/jpeg",
                        data=base64.b64decode("..."),
                    ),
                    media_resolution={"level": "media_resolution_high"}
                )
            ]
        )
    ]
)

print(response.text)
```

温度

对于 Gemini 3，我们强烈建议将温度参数保留为默认值 `1.0`。

虽然之前的模型通常可以通过调整温度来控制创造性与确定性，但 Gemini 3 的推理能力已针对默认设置进行了优化。更改温度（将其设置为低于 1.0）可能会导致意外行为，例如循环或性能下降，尤其是在复杂的数学或推理任务中。

思路签名

Gemini 3 使用思路签名 (https://ai.google.dev/gemini-api/docs/thought-signatures?authuser=1&hl=zh-cn)在 API 调用之间保持推理上下文。这些签名是模型内部思考过程的加密表示形式。为确保模型保持推理能力，您必须在请求中将这些签名原封不动地返回给模型：

- **函数调用（严格）**：该 API 会对“当前回合”强制执行严格的验证。缺少签名会导致 400 错误。
- **文本/聊天**：验证并非强制执行，但省略签名会降低模型的推理能力和回答质量。
- **图片生成/编辑（严格）**：API 会对所有模型部分（包括 `thoughtSignature`）强制执行严格的验证。缺少签名会导致 400 错误。

成功： 如果您使用官方 SDK（Python、Node、Java） (https://ai.google.dev/gemini-api/docs/function-calling?example=meeting&authuser=1&hl=zh-cn#thinking)和标准对话历史记录，系统会自动处理思维签名。您无需手动管理这些字段。

函数调用（严格验证）

当 Gemini 生成 `functionCall` 时，它会依赖 `thoughtSignature` 在下一轮中正确处理工具的输出。“当前对话轮次”包括自上次标准**用户** `text` 消息以来发生的所有模型 (`functionCall`) 和用户 (`functionResponse`) 步骤。

- **单次函数调用**：`functionCall` 部分包含签名。您必须归还。
- **并行函数调用**：只有列表中的第一个 `functionCall` 部分会包含签名。您必须按收到的确切顺序退回这些部件。
- **多步（顺序）**：如果模型调用某个工具、收到结果，然后（在同一轮对话中）调用*另一个*工具，则**两个**函数调用都有签名。您必须返回历史记录中的**所有**累积签名。

文字和流式传输

对于标准聊天或文本生成，我们无法保证签名一定会显示。

- **非流式传输**：响应的最终内容部分可能包含 `thoughtSignature`，但并非总是包含。如果返回了此类令牌，您应将其发送回去，以保持最佳性能。
- **流式传输**：如果生成了签名，则该签名可能位于包含空文本部分的最终块中。确保您的流解析器即使在文本字段为空的情况下也会检查签名。

图片生成和修改

对于 `gemini-3-pro-image-preview`，思维签名对于对话式编辑至关重要。当您要求模型修改图片时，它会依赖上一次对话中的 `thoughtSignature` 来了解原始图片的构图和逻辑。

- **编辑**：签名保证位于回答的思路部分（`text` 或 `inlineData`）之后的第一部分，以及后续每个 `inlineData` 部分。您必须返回所有这些签名，以免出错。

代码示例

多步骤函数调用（按顺序）

用户在一个回合中提出了需要两个单独步骤（查看航班 -> 预订出租车）的问题。

第 1 步：模型调用 Flight Tool。
模型返回签名 `<Sig_A>`

```
// Model Response (Turn 1, Step 1)
  {
    "role": "model",
    "parts": [
      {
        "functionCall": { "name": "check_flight", "args": {...} },
        "thoughtSignature": "<Sig_A>" // SAVE THIS
      }
    ]
  }
```

第 2 步：用户发送航班结果
我们必须发送回 `<Sig_A>`，以保持模型的思路。

```
// User Request (Turn 1, Step 2)
[
  { "role": "user", "parts": [{ "text": "Check flight AA100..." }] },
  {
    "role": "model",
    "parts": [
      {
        "functionCall": { "name": "check_flight", "args": {...} },
        "thoughtSignature": "<Sig_A>" // REQUIRED
      }
    ]
  },
  { "role": "user", "parts": [{ "functionResponse": { "name": "check_flight", "res
]
```

第 3 步：模型调用出租车工具
模型通过 `<Sig_A>` 记住航班延误情况，现在决定预订出租车。它会生成一个*新*签名 `<Sig_B>`。

```
// Model Response (Turn 1, Step 3)
{
  "role": "model",
  "parts": [
    {
      "functionCall": { "name": "book_taxi", "args": {...} },
      "thoughtSignature": "<Sig_B>" // SAVE THIS
    }
  ]
}
```

第 4 步：用户发送出租车结果
如需完成此轮对话，您必须发送整个链：`<Sig_A>` 和 `<Sig_B>`。

```
// User Request (Turn 1, Step 4)
[
  // ... previous history ...
  {
    "role": "model",
    "parts": [
       { "functionCall": { "name": "check_flight", ... }, "thoughtSignature": "<Si
    ]
  },
  { "role": "user", "parts": [{ "functionResponse": {...} }] },
  {
    "role": "model",
    "parts": [
       { "functionCall": { "name": "book_taxi", ... }, "thoughtSignature": "<Sig_B
    ]
  },
  { "role": "user", "parts": [{ "functionResponse": {...} }] }
]
```

并行函数调用

用户询问：“查看巴黎和伦敦的天气。”模型在一个回答中返回了两个函数调用。

```
// User Request (Sending Parallel Results)
[
  {
    "role": "user",
    "parts": [
      { "text": "Check the weather in Paris and London." }
    ]
  },
  {
    "role": "model",
    "parts": [
      // 1. First Function Call has the signature
      {
        "functionCall": { "name": "check_weather", "args": { "city": "Paris" } },
        "thoughtSignature": "<Signature_A>"
      },
      // 2. Subsequent parallel calls DO NOT have signatures
      {
        "functionCall": { "name": "check_weather", "args": { "city": "London" } }
      }
    ]
  },
  {
    "role": "user",
    "parts": [
      // 3. Function Responses are grouped together in the next block
      {
        "functionResponse": { "name": "check_weather", "response": { "temp": "15C"
      },
      {
        "functionResponse": { "name": "check_weather", "response": { "temp": "12C"
      }
    ]
  }
]
```

文本/上下文推理（无验证）

用户提出的问题需要进行上下文推理，但不能使用外部工具。虽然未经过严格验证，但包含签名有助于模型针对后续问题保持推理链。

```
// User Request (Follow-up question)
[
  {
    "role": "user",
```

```
    "parts": [{ "text": "What are the risks of this investment?" }]
  },
  {
    "role": "model",
    "parts": [
      {
        "text": "I need to calculate the risk step-by-step. First, I'll look at vo
        "thoughtSignature": "<Signature_C>" // Recommended to include
      }
    ]
  },
  {
    "role": "user",
    "parts": [{ "text": "Summarize that in one sentence." }]
  }
]
```

图片生成与编辑

对于映像生成，签名会受到严格验证。它们会显示在**第一部分**（文字或图片）和**所有后续图片部分**中。所有卡牌都必须在下一回合中返回。

```
// Model Response (Turn 1)
{
  "role": "model",
  "parts": [
    // 1. First part ALWAYS has a signature (even if text)
    {
      "text": "I will generate a cyberpunk city...",
      "thoughtSignature": "<Signature_D>"
    },
    // 2. ALL InlineData (Image) parts ALWAYS have signatures
    {
      "inlineData": { ... },
      "thoughtSignature": "<Signature_E>"
    },
  ]
}

// User Request (Turn 2 - Requesting an Edit)
{
  "contents": [
    // History must include ALL signatures received
    {
      "role": "user",
      "parts": [{ "text": "Generate a cyberpunk city" }]
    },
```

```
    {
      "role": "model",
      "parts": [
         { "text": "...", "thoughtSignature": "<Signature_D>" },
         { "inlineData": "...", "thoughtSignature": "<Signature_E>" },
      ]
    },
    // New User Prompt
    {
      "role": "user",
      "parts": [{ "text": "Make it daytime." }]
    }
  ]
}
```

从其他型号迁移

如果您要从其他模型（例如，Gemini 2.5）或注入并非由 Gemini 3 生成的自定义函数调用，您将无法获得有效的签名。

如需在这些特定场景中绕过严格验证，请使用以下特定虚拟字符串填充相应字段：`"thoughtSignature": "context_engineering_is_the_way_to_go"`

使用工具生成结构化输出

Gemini 3 可让您将结构化输出 (https://ai.google.dev/gemini-api/docs/structured-output?authuser=1&hl=zh-cn)与内置工具（包括依托 Google 搜索进行接地 (https://ai.google.dev/gemini-api/docs/google-search?authuser=1&hl=zh-cn)、网址上下文 (https://ai.google.dev/gemini-api/docs/url-context?authuser=1&hl=zh-cn)和代码执行 (https://ai.google.dev/gemini-api/docs/code-execution?authuser=1&hl=zh-cn)）结合使用。

PythonJavaScript (#javascript)REST (#rest)
(#python)

```
from google import genai
from google.genai import types
from pydantic import BaseModel, Field
from typing import List

class MatchResult(BaseModel):
    winner: str = Field(description="The name of the winner.")
    final_match_score: str = Field(description="The final match score.")
    scorers: List[str] = Field(description="The name of the scorer.")

client = genai.Client()
```

```
response = client.models.generate_content(
    model="gemini-3-pro-preview",
    contents="Search for all details for the latest Euro.",
    config={
        "tools": [
            {"google_search": {}},
            {"url_context": {}}
        ],
        "response_mime_type": "application/json",
        "response_json_schema": MatchResult.model_json_schema(),
    },
)

result = MatchResult.model_validate_json(response.text)
print(result)
```

图片生成

借助 Gemini 3 Pro Image，您可以根据文本提示生成和修改图片。它会使用推理功能“思考”提示，并检索实时数据（例如天气预报或股市图表），然后使用 Google 搜索 (https://ai.google.dev/gemini-api/docs/google-search?authuser=1&hl=zh-cn)进行事实依据核查，最后生成高保真图片。

新增和改进的功能：

- **4K 和文字渲染**：生成清晰易读的文字和图表，分辨率最高可达 2K 和 4K。
- **有依据的生成**：使用 `google_search` 工具验证事实，并根据真实世界的信息生成图像。
- **对话式编辑**：只需提出更改要求（例如，“将背景设为日落”）。此工作流依赖于**思维签名**来保留回合之间的视觉上下文。

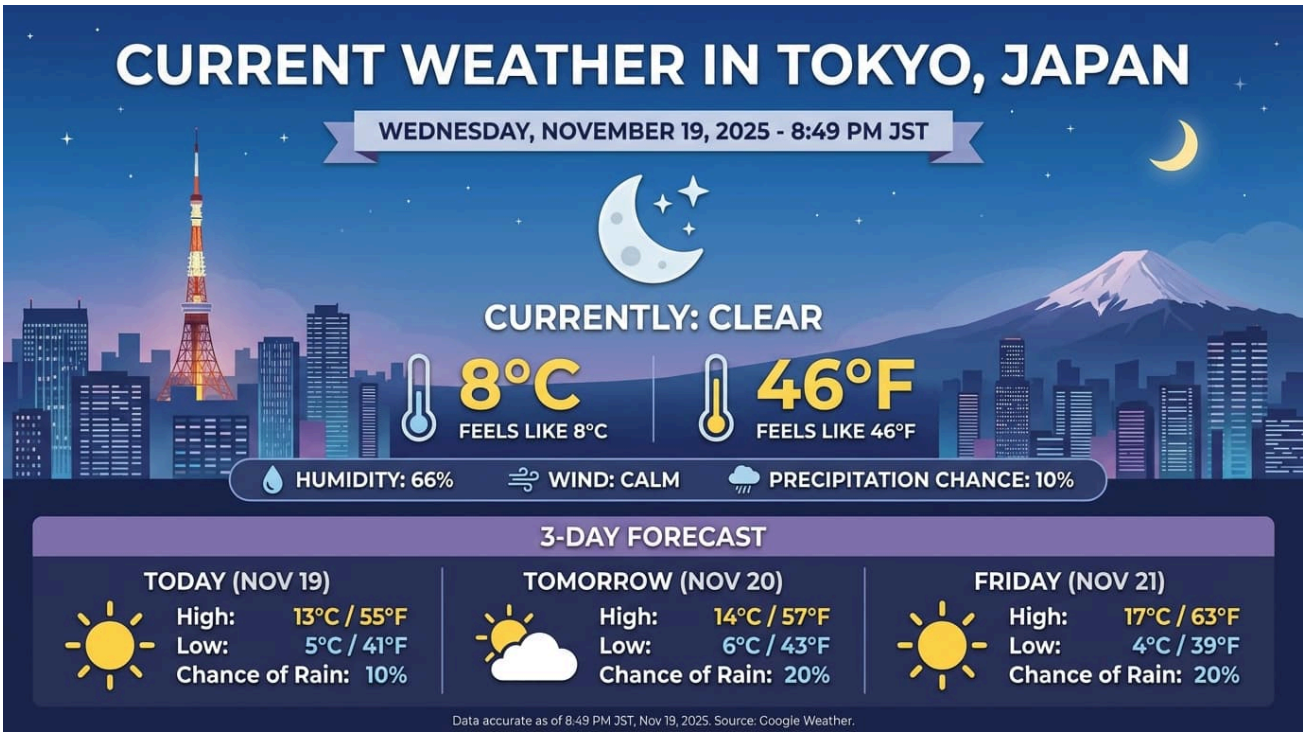
如需详细了解宽高比、编辑工作流程和配置选项，请参阅图片生成指南 (https://ai.google.dev/gemini-api/docs/image-generation?authuser=1&hl=zh-cn)。

Python JavaScript (#javascript) REST (#rest)
(#python)

```
from google import genai
from google.genai import types

client = genai.Client()

response = client.models.generate_content(
    model="gemini-3-pro-image-preview",
    contents="Generate an infographic of the current weather in Tokyo.",
    config=types.GenerateContentConfig(
        tools=[{"google_search": {}}],
```

```
        image_config=types.ImageConfig(
            aspect_ratio="16:9",
            image_size="4K"
        )
    )
)

image_parts = [part for part in response.parts if part.inline_data]

if image_parts:
    image = image_parts[0].as_image()
    image.save('weather_tokyo.png')
    image.show()
```

示例回答



从 Gemini 2.5 迁移

Gemini 3 是我们迄今为止功能最强大的模型系列，与 Gemini 2.5 Pro 相比，性能有了显著提升。迁移时，请考虑以下事项：

- **思考**：如果您之前使用复杂的提示工程（例如思维链）来强制 Gemini 2.5 进行推理，不妨尝试使用 Gemini 3 并搭配 `thinking_level: "high"` 和简化的提示。
- **温度设置**：如果您的现有代码明确设置了温度（尤其是将温度设置为较低的值以获得确定性输出），我们建议您移除此参数并使用 Gemini 3 的默认值 1.0，以避免在处理复杂任务时出现潜在的循环问题或性能下降。

- **PDF 和文档理解**：PDF 的默认 OCR 分辨率已更改。如果您之前依赖于密集文档解析的特定行为，请测试新的 `media_resolution_high` 设置，以确保准确性不受影响。
- **令牌消耗**：迁移到 Gemini 3 Pro 默认设置可能会**增加** PDF 的令牌用量，但会**减少**视频的令牌用量。如果请求现在因默认分辨率更高而超出上下文窗口，我们建议明确降低媒体分辨率。
- **图片分割**：Gemini 3 Pro 不支持图片分割功能（返回对象的像素级遮罩）。对于需要原生图片分割的工作负载，我们建议继续使用关闭了思考功能的 Gemini 2.5 Flash 或 Gemini Robotics-ER 1.5 (https://ai.google.dev/gemini-api/docs/robotics-overview?authuser=1&hl=zh-cn)。

OpenAI 兼容性

对于使用 OpenAI 兼容层的用户，标准参数会自动映射到 Gemini 等效参数：

- `reasoning_effort` (OAI) 映射到 `thinking_level` (Gemini)。请注意，`reasoning_effort` 中等映射到 `thinking_level` 高。

提示最佳实践

Gemini 3 是一种推理模型，因此您需要改变提示方式。

- **精确的指令**：输入提示应简洁明了。Gemini 3 最适合处理直接、清晰的指令。它可能会过度分析用于旧模型的冗长或过于复杂的提示工程技术。
- **输出详细程度**：默认情况下，Gemini 3 的输出详细程度较低，更倾向于提供直接、高效的答案。如果您的使用情形需要更具对话性或“聊天”风格的角色，您必须在提示中明确引导模型（例如，“以友善健谈的助理的身份解释一下”）。
- **上下文管理**：处理大型数据集（例如整本书、代码库或长视频）时，请将具体指令或问题放在提示末尾的数据上下文之后。在提问时，以“根据上述信息…”之类的短语开头，将模型的推理锚定到提供的数据。

如需详细了解提示设计策略，请参阅提示工程指南 (https://ai.google.dev/gemini-api/docs/prompting-strategies?authuser=1&hl=zh-cn)。

常见问题解答

1. **Gemini 3 Pro 的知识截止日期是什么？** Gemini 3 的知识截止日期为 2025 年 1 月。如需了解最新信息，请使用搜索基础 (https://ai.google.dev/gemini-api/docs/google-search?authuser=1&hl=zh-cn)工具。

2. **上下文窗口有哪些限制？**Gemini 3 Pro 支持 100 万个 token 的输入上下文窗口，以及最多 6.4 万个 token 的输出。
3. **Gemini 3 Pro 是否有免费层级？**您可以在 Google AI Studio 中免费试用该模型，但目前 Gemini API 中没有 `gemini-3-pro-preview` 的免费层级。
4. **我的旧 `thinking_budget` 代码是否仍然有效？**可以，为了实现向后兼容性，我们仍支持 `thinking_budget`，但建议您迁移到 `thinking_level` 以获得更可预测的性能。请勿在同一请求中同时使用这两个参数。
5. **Gemini 3 是否支持 Batch API？**可以，Gemini 3 支持批量 API (https://ai.google.dev/gemini-api/docs/batch-api?authuser=1&hl=zh-cn)。
6. **是否支持上下文缓存？**可以，Gemini 3 支持上下文缓存 (https://ai.google.dev/gemini-api/docs/caching?lang=python&authuser=1&hl=zh-cn)。启动缓存所需的最低 token 数为 2,048 个。
7. **Gemini 3 支持哪些工具？**Gemini 3 支持Google 搜索 (https://ai.google.dev/gemini-api/docs/google-search?authuser=1&hl=zh-cn)、文件搜索 (https://ai.google.dev/gemini-api/docs/file-search?authuser=1&hl=zh-cn)、代码执行 (https://ai.google.dev/gemini-api/docs/code-execution?authuser=1&hl=zh-cn)和网址上下文 (https://ai.google.dev/gemini-api/docs/url-context?authuser=1&hl=zh-cn)。它还支持标准函数调用 (https://ai.google.dev/gemini-api/docs/function-calling?example=meeting&authuser=1&hl=zh-cn)，以便您使用自己的自定义工具。请注意，基于 Google 地图进行接地 (https://ai.google.dev/gemini-api/docs/maps-grounding?authuser=1&hl=zh-cn)和电脑使用 (https://ai.google.dev/gemini-api/docs/computer-use?authuser=1&hl=zh-cn)目前不受支持。

后续步骤

- 开始使用 Gemini 3 Cookbook (https://colab.research.google.com/github/google-gemini/cookbook/blob/main/quickstarts/Get_started.ipynb?authuser=1&hl=zh-cn#templateParams=%7B%22MODEL_ID%22:+%22gemini-3-pro-preview%22%7D)
- 请参阅有关思考水平 (https://colab.research.google.com/github/google-gemini/cookbook/blob/main/quickstarts/Get_started_thinking_REST.ipynb?authuser=1&hl=zh-cn#gemini3)
  的专用 Cookbook 指南，了解如何从思考预算迁移到思考水平。

最后更新时间 (UTC)：2025-11-29。