04

Implementing the PaLM API

In this module, you learn to ...

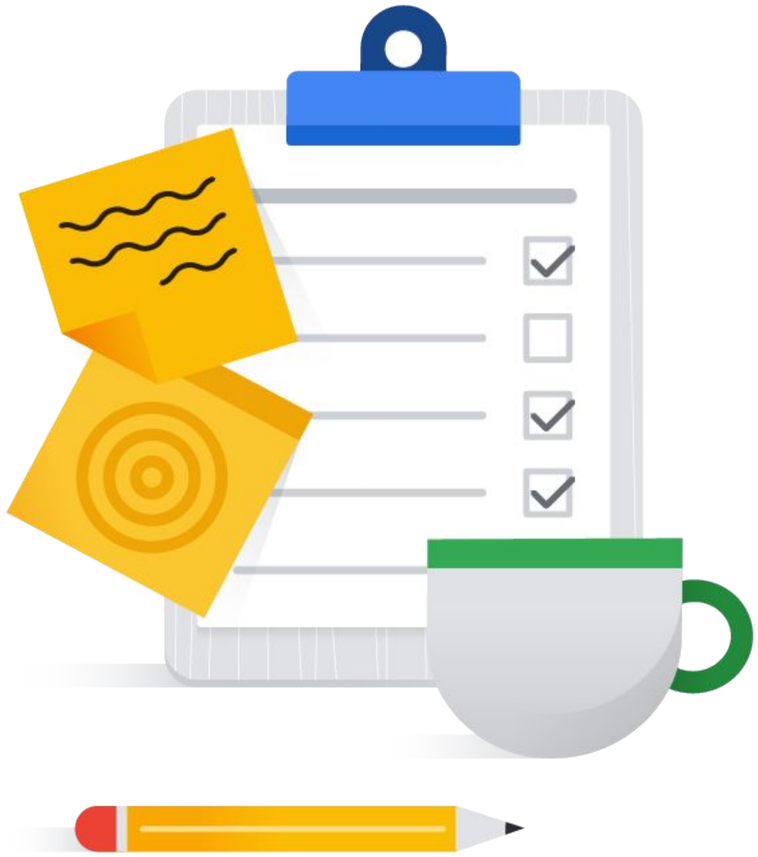
01 Use the Vertex AI API to generate content using the Pathways Language Model (PaLM)

02 Program Python applications the use PaLM to generate content

03 Integrate PaLM and GenAI into your applications

Topics

01	Introduction to the PaLM API
02	Generative AI Powered Applications with Python
03	Using the PaLM API in Applications



Pathways Language Model (PaLM) is a large language model

- LLMs are very sophisticated autocomplete applications
  - They learn patterns from large amounts of text
  - Use those patterns to generate text
- When generating text they calculate the next most likely tokens (words)
  - They aren’t smart; it’s math and statistics
- PaLM can generate text with two basic services
  - Text service for single request interactions
  - Chat service is for interactive, multi-turn interactions

The PaLM API can be used via a REST service call

- cURL is a program that makes web requests
  - -X parameter is the HTTP verb (GET, POST, PUT, DELETE, etc.)
  - -H parameters adds header variables
  - -d parameter is the body of the request, in this case the prompt

```
API_ENDPOINT="us-central1-aiplatform.googleapis.com"
PROJECT_ID="vertext-ai-dar"
MODEL_ID="chat-bison@001"

curl -X POST -H "Authorization: Bearer $(gcloud auth print-access-token)" -H
"Content-Type: application/json"
"https://${API_ENDPOINT}/v1/projects/${PROJECT_ID}/locations/us-central1/publish
ers/google/models/${MODEL_ID}:predict" -d $'{
    "instances": [
        {
            "context": "",

<<code omitted>>
```

To use the PaLM API, your application will need to be authenticated

- Ways to authenticate your application
  - Obtain an authorization token
  - Run the application using a service account

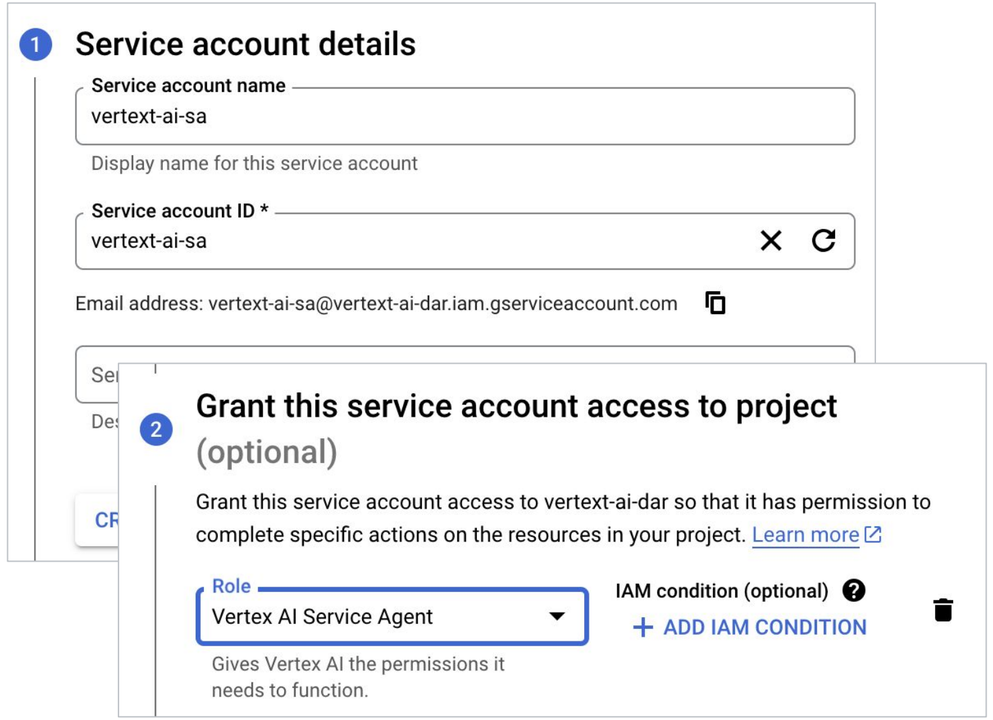
An authorization token identifies the caller of an API

- Created using the Google Cloud CLI
  - The gcloud CLI must be initialized with either a user of service account
- Set the Authorization header variable with the token generated using gcloud

```
API_ENDPOINT="us-central1-aiplatform.googleapis.com"
PROJECT_ID="vertext-ai-dar"
MODEL_ID="chat-bison"
curl -H "Content-Type: application/json" \
-H "Authorization: Bearer $(gcloud auth print-access-token)" \
"https://${API_ENDPOINT}/v1/projects/${PROJECT_ID}/locations/${LOCATION_ID}/publishers/goo
gle/models/${MODEL_ID}:predict" \
-d $'{ "instances": [ { "content": "Give me five subcategories of jazz" } ],
"parameters": { "candidateCount": 1, "maxOutputTokens": 1024, "temperature": 0.2, "topP":
0.8, "topK": 40 } }'
```

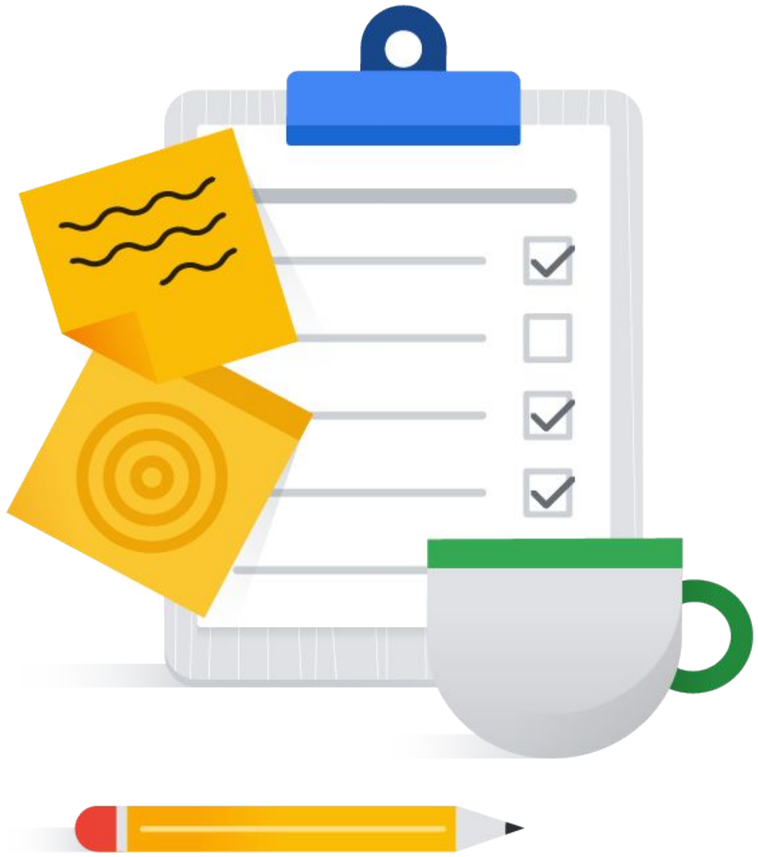
If running an application in Google Cloud, assign a service account to the runtime

- Create a service account using **IAM**
  - Assign the **Vertex AI Service Agent** role
  - Use the service account to identify the runtime
- If using Cloud Run, App Engine, or Cloud Functions, the runtime will use the Compute Engine Default Service Account by default
  - This will work as it uses the Editor role
  - Violates principle of least privilege
- You can also download Service Account keys to authenticate programs that use the language client libraries



Topics

01	Introduction to the PaLM API
02	Generative AI Powered Applications with Python
03	Using the PaLM API in Applications



You can use Vertex AI Studio to generate code for a Python, Node.js or Java app or cURL

- Click the **Get Code** button and select Python, Node.js, Java or CURL

Use this script to request a model response in your application.

1. Install the Vertex AI SDK: Open a terminal window and enter the command below. You can also install it in a virtualenv .

```
!pip install --upgrade google-cloud-aiplatform
```

2. Use the following code in your application to request a model response

```python
import vertexai
from vertexai.language_models import TextGenerationModel

vertexai.init(project="roi-genai-joey", location="us-central1")
parameters = {
    "candidate_count": 1,
    "max_output_tokens": 1024,
    "temperature": 0.9,
    "top_p": 1
}
model = TextGenerationModel.from_pretrained("text-bison")
response = model.predict(
    """write a short story about a kitten""",
    **parameters
)
print(f"Response from Model: {response.text}")
```

Use this script to request a model response in your application.

1. Install the Vertex AI SDK.

```
npm install https://github.com/googleapis/nodejs-vertexai
gcloud auth application-default login
```

2. Create an index.js file and add the following code:

```javascript
const {VertexAI} = require('@google-cloud/vertexai');

// Initialize Vertex with your Cloud project and location
const vertex_ai = new VertexAI({project: 'roi-genai-joey', location: 'us-central1'});
const model = 'gemini-pro-vision';

// Instantiate the models
const generativeModel = vertex_ai.preview.getGenerativeModel({
  model: model,
  generation_config: {
    "max_output_tokens": 2048,
    "temperature": 0.4,
    "top_p": 1,
    "top_k": 32
},
});

async function generateContent() {
  const req = {
    contents: [{role: 'user', parts: [{text: 'write a short story about a kitten'}]}],
  };

  const streamingResp = await generativeModel.generateContentStream(req);

  for await (const item of streamingResp.stream) {
    process.stdout.write('stream chunk: ' + item);
  }

  process.stdout.write('aggregated response: ' + (await streamingResp.response));
};

generateContent();
```

Vertex AI requirements for Python

- Use pip to install Google Cloud AI Platform
  - Or add to your `requirements.txt` file

```
pip install google-cloud-aiplatform >= 1.36.0
```

- Import Vertex AI

```
import vertexai
from vertexai.preview.language_models import ChatModel, InputOutputTextPair
```

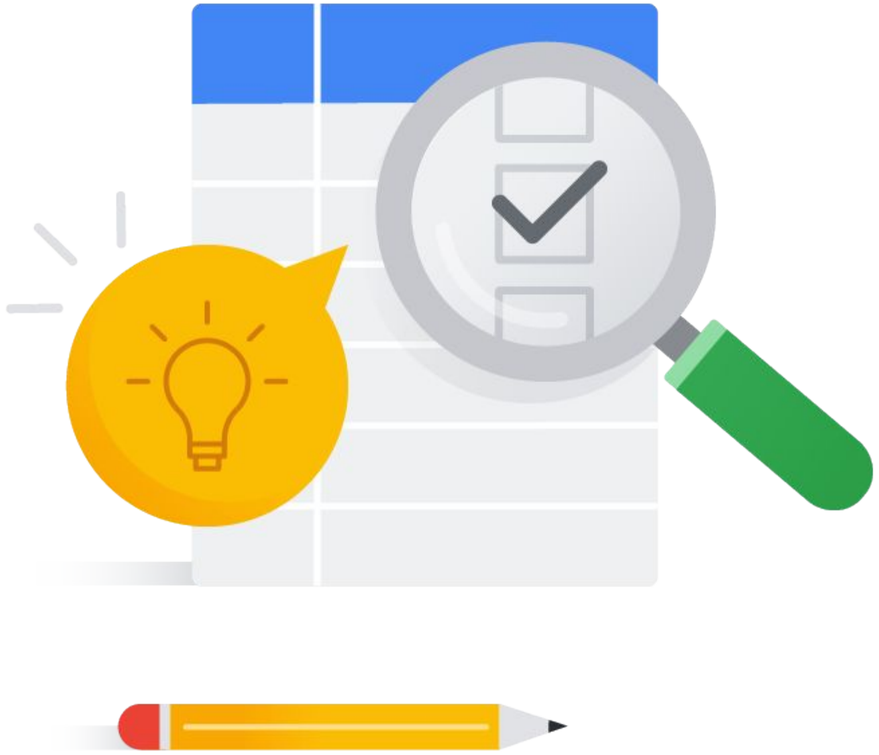
Using Vertex AI from Python

```
vertexai.init(project="vertext-ai-dar", location="us-central1")
chat_model = ChatModel.from_pretrained("chat-bison@001")
parameters = {
    "temperature": 0.2,
    "max_output_tokens": 256,
    "top_p": 0.8,
    "top_k": 40
}
chat = chat_model.start_chat()
response = chat.send_message("""List 10 fun tourist destinations""",
**parameters)
print(f"Response from Model: {response.text}")
```

Lab

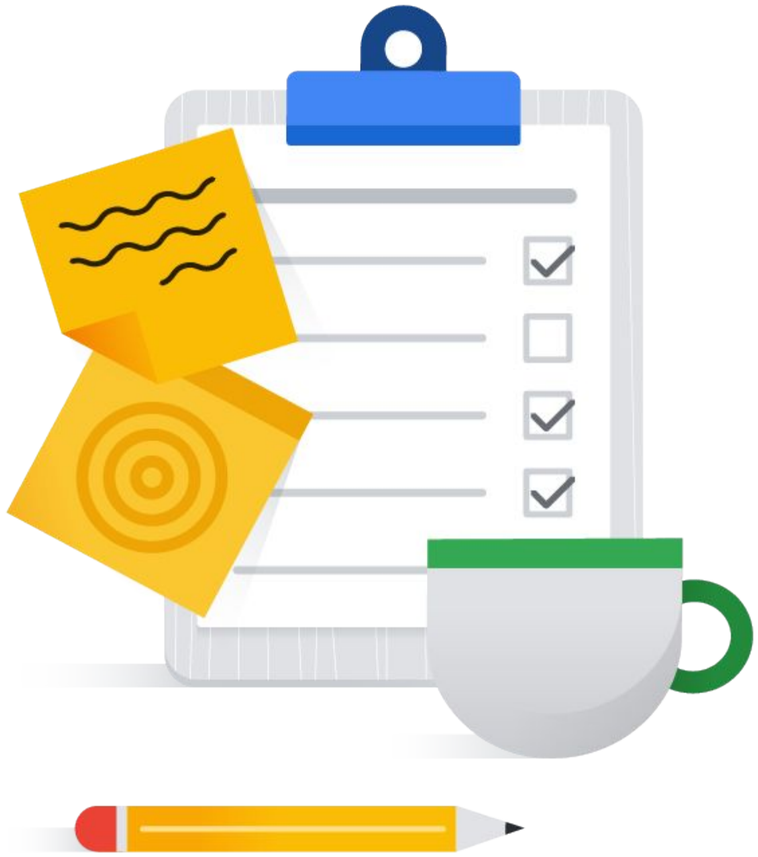
30 min

Getting Started with the PaLM API for Chatbots



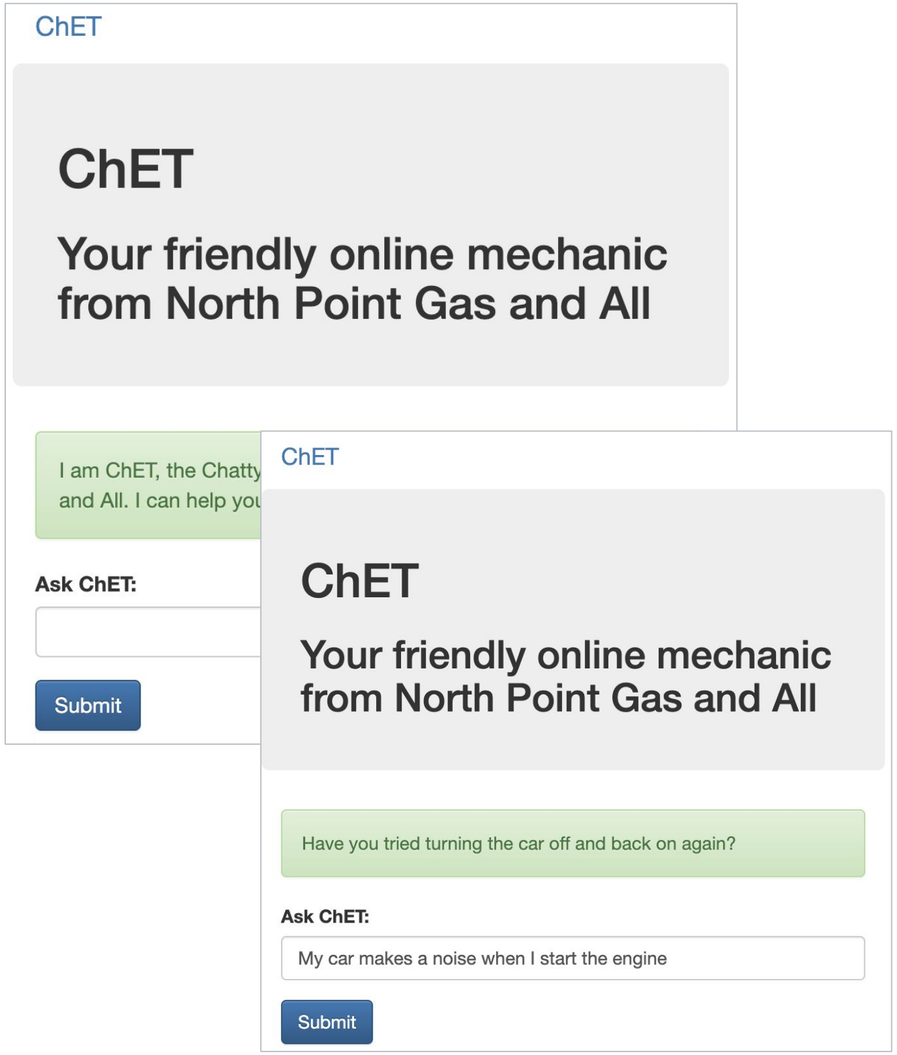
Topics

01	Introduction to the PaLM API
02	Generative AI Powered Applications with Python
03	Using the PaLM API in Applications



Python Flask Website example

- This is an example of using the chat service with the PaLM API
  - The chat will remember the history of the conversation
- Context must be added to tell the PaLM API to emulate a customer service agent for the service station
- The coding is simple as you are just submitting an HTML form and making a request to the PaLM API for a response
  - The response is displayed on the screen



Add the Python requirements

- Add Google Cloud AI Platform to the requirements.txt file
- Add the required imports at the top of the code file

requirements.txt

mixabot > requirements.txt

```
Flask
Jinja2
pytest
google-cloud-aiplatform
```

```
from flask import Flask, render_template, request
import os
import vertexai
from vertexai.preview.language_models import ChatModel, InputOutputTextPair
```

Handling web requests in Flask

- The default route will handle HTTP posts and gets
  - Post means a question was submitted from the HTML form
  - Get means there is no question (have CHeT introduce itself)
- The code for using the PaLM API is in the **get_response()** function

```
@app.route("/", methods = ['POST', 'GET'])
def main():
    if request.method == 'POST':
        input = request.form['input']
        response = get_response(input)
    else:
        input = ""
        response = get_response("Who are you and what can you do?")

    model = {"title": "CHeT", "message": response, "input": input}
    return render_template('index.html', model=model)
```

Initializing the Chat session

```
vertexai.init(location="us-central1")
chat_model = ChatModel.from_pretrained("chat-bison@001")
parameters = {
        "temperature": TEMPERATURE,
        "max_output_tokens": MAX_OUTPUT_TOKENS,
        "top_p": TOP_P,
        "top_k": TOP_K
     }
examples=[
        InputOutputTextPair(
            input_text="""When I turn my car on, there is a clicking noise. """,
            output_text="""Did you try turning the engine off and back on again?"""
        )]

chat = chat_model.start_chat(context=CONTEXT, examples=examples)
```

Initialize the API and set up the parameters

Add examples

Start the chat

Making a request to the PaLM API (continued)

```
def get_response(input):
    response = chat.send_message(input, **parameters)
    return response, chat.message_history
```

The input is the message from the user

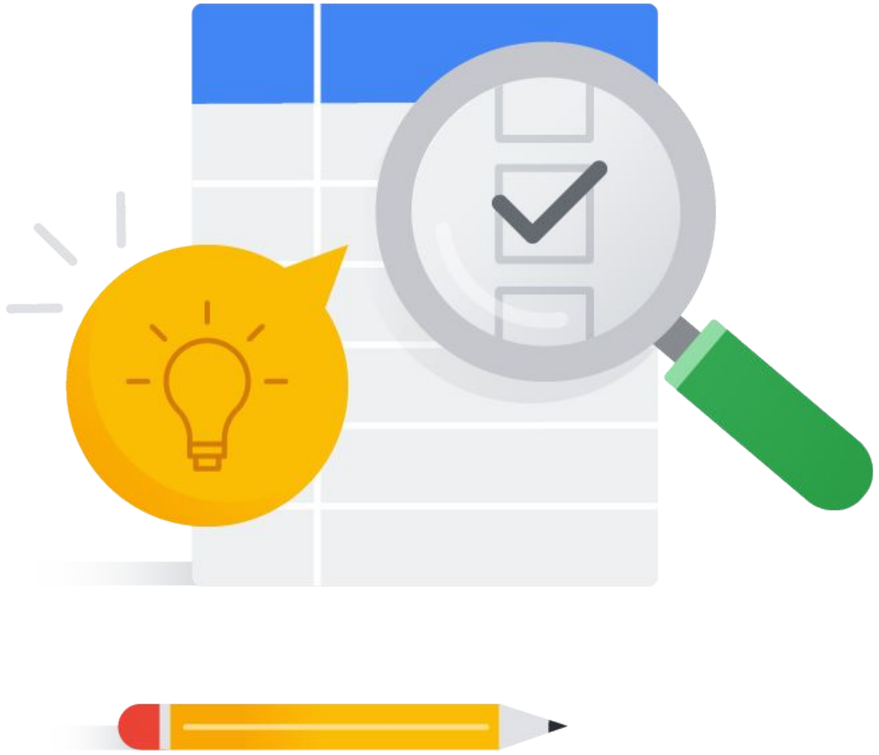
Send the message

Return the response and history if desired

Lab

30 min

Integrating the PaLM API into Chat Applications



In this module, you learned to ...

01 Use the Vertex AI API to generate content using the Pathways Language Model (PaLM)

02 Program Python applications the use PaLM to generate content

03 Integrate PaLM and GenAI into your applications

Questions and answers

Quiz question

How do you authenticate a request to the PaLM API ?

A: Using a API Key

B: Using an authorization token

C: With a Service Account

D: All of the above would work

Quiz question

How do you authenticate a request to the PaLM API ?

A: Using a API Key

B: Using an authorization token

C: With a Service Account

D: All of the above would work

Quiz question

What programming languages are Supported by the PaLM API ?

A: Python

B: Node.js

C: Swift

D: Java

E: All of the above

Quiz question

What programming languages are Supported by the PaLM API ?

A: Python

B: Node.js

C: Swift

D: Java

E: All of the above