Generative AI
(with a focus on Cloud)

Dan Zaratsian

AI/ML Architect, Gaming Solutions @ Google

d.zaratsian@gmail.com

https://github.com/zaratsian

TOPICS

- Session 1: Course Intro, Trends, and Approach to AI/ML
- Session 2: SQL and NoSQL
- Session 3: Distributed ML with Spark and Tensorflow
- Session 4: Cloud Generative AI Services and Architectures
- Session 5: Cloud Machine Learning Services
- Session 6: Serverless ML, Architectures, and Deploying ML

Five (general) areas of GenAI by use cases

CREATE

Bring your thoughts and visions to life

Use cases:

- Product descriptions from images
- Images from text
- Blog post from content
- Email from content
- Release notes from content
- Report from content
- Press releases from content
- Personalized ads

SUMMARIZE

Condense and summarize your knowledge base into a simple format

Use cases:

- Content/video summarization
- Intra-knowledge Q&A
- Explanations of code content

DISCOVER

Help your customers and employees find what they need at the right time

Use cases:

- Search for a document
- Machine-generated event monitoring
- File organization based on content
- Exam questions from content

AUTOMATE

Automate your customer service across multiple channels

Use cases:

- Contract information extraction
- Feedback classification and ticket creation
- Sentiment analysis
- Content translation
- Structured data extraction from file
- Media tagging
- Product tagging
- Content moderation

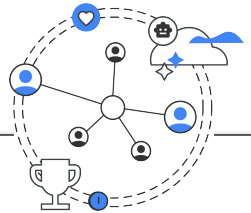
CONVERSE

Create an AI-based virtual assistant

Use cases:

- Chat
- Search
- Virtual Agent

Classifications of GenAI use cases in Games

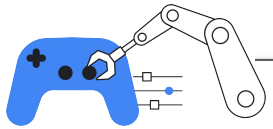


Improving Productivity during Game Development

VertexAI | Duet AI | GKE

Use Generative AI to accelerate time-to-launch by creating content and simplifying development

- 2D & 3D assets (characters, props)
- Audio & video assets generation
- Code generation
- AI-based game testing



Improving Player Experience during GamePlay

VertexAI | GKE

Use AI/ML & GenAI to adapt the gameplay and empower players to generate game content in realtime.

- Smart NPCs (bots)
- Dynamic in-game content
- Customized player experiences
- User-generated content
- Endless worlds

User pain-points for GenAI in Games

Platform

Cost

At-scale cost efficiency to ensure financial feasibility for AAA games

Latency

Low latency to ensure smooth gameplay & user experience.

Platform Selection

Platform(s) with performance, & access to models without lock-in.

AI Maturity

LLM Unpredictability

Need a coherent, relevant, and contextually appropriate inference

Avoiding Bias

Training & fine-tuning should not propagate biases & stereotypes

Content Filtering

Content moderation is needed to ensure safe & inclusive gameplay

Gameplay

Creativity vs Boundaries

Need to balance user generated content with game lore & structure

GenAI model constraints

Some games need content for gameplay which LLMs filter out

Procedural generation

Procedural generation with GenAI still requires human supervision

Getting Started with Generative AI (on Cloud)

Choose the GenAI model, infra, and supporting services that meet your requirements.

1 GenAI Models

Google Gemini

Open Models
- Gemma
- Llama 2
- Stable Diffusion

3rd Party:
- Anthropic Claude 2

2 Customize

Google Cloud Vertex AI

- Model fine-tuning
- Pre-built Notebooks
- Model Evaluation
- Responsible AI
- Citation Checks

3 Scalable Infra

Google Serving Infrastructure

- Managed Endpoints
- One-click deploy
- Serving on GKE
- Run on GPUs, TPUs

4 Frameworks & Extending LLMs

Google Services supporting GenAI

- Embeddings DBs
- GenAI Memory
- RAG
- Grounding
- Function Calling
- 3rd Party Frameworks

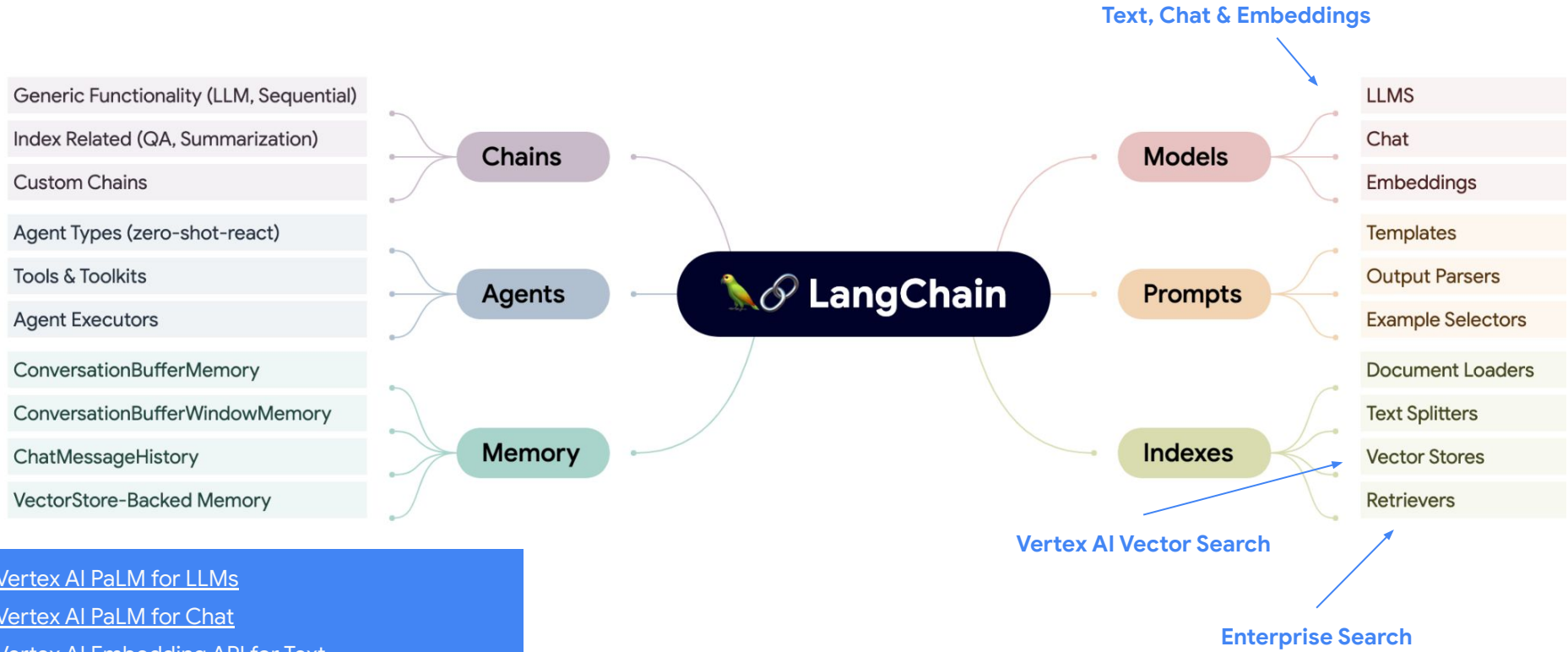
Text/Chat ✦ Code ✦ Image ✦ Video ✦ Audio ✦ ...other modalities coming...

LLM Services & Frameworks

Platform & Models	Frameworks	Techniques
Google Cloud Gemini Vertex AI amazon ANTHROPIC OpenAI Microsoft Hugging Face	Google Cloud Vertex AI SKDs *Each cloud has their own GenAI SDK LangChain LlamaIndex	Retrieval-Augmented Generation (RAG) Prompt-Tuning Model Fine-Tuning Knowledge Distillation (smaller/efficient student model) Chain-of-Thought Prompting (Reasoning) LLM Agents & Tools

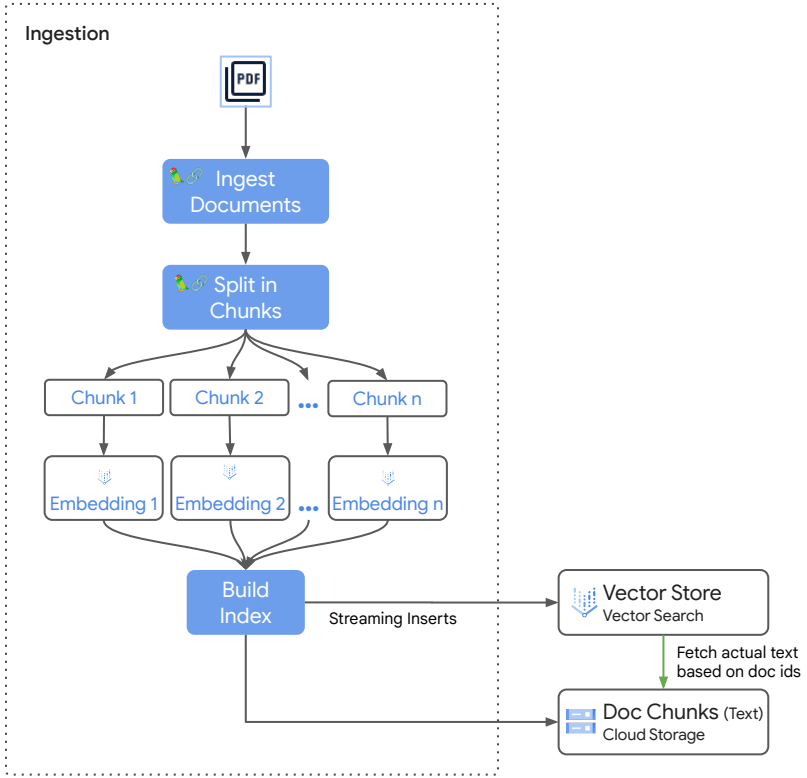
NOTE: These are general thoughts, used for specific in-class messaging

LangChain Components



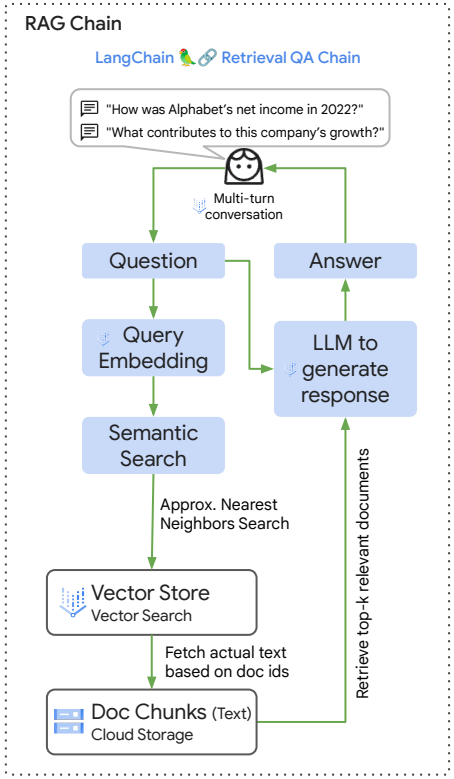
[Vertex AI PaLM for LLMs]
[Vertex AI PaLM for Chat]
[Vertex AI Embedding API for Text]
[Vertex AI Vector Search]
[Enterprise Search]

Retrieval QA Chain: Ingestion



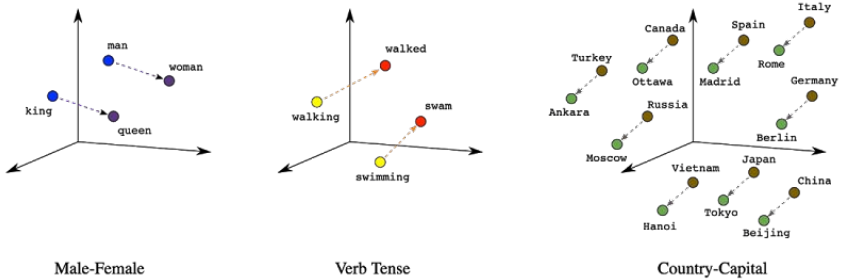

```python
from langchain.embeddings import VertexAIEmbeddings
from langchain.document_loaders import GCSFileLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from langchain.vectorstores import MatchingEngine

# Define Text Embeddings model
embedding = VertexAIEmbeddings()

# Define Vertex AI Vector Search as Vector Store
me = MatchingEngine.from_components(
    project_id=PROJECT_ID,
    region=ME_REGION,
    gcs_bucket_name=f'gs://{ME_BUCKET_NAME}',
    embedding=embedding,
    index_id=ME_INDEX_ID,
    endpoint_id=ME_INDEX_ENDPOINT_ID)

# Define Cloud Storage file loader to read a document
loader = GCSFileLoader(project_name=PROJECT_ID,
                       bucket=bucket, blob=prefix)

# Split document into chunks
text_splitter = RecursiveCharacterTextSplitter(
                    chunk_size=1000, chunk_overlap=0)
doc_splits = text_splitter.split_documents(document)

# Add embeddings of document chunks to Vertex AI Vector Search
texts = [doc.page_content for doc in doc_splits]
me.add_texts(texts=texts)
```

Retrieval QA Chain: Query Time




```python
from langchain.chains import RetrievalQA
from langchain.llms import VertexAI

# Define Vertex AI Vector Search as Vector Store
me = MatchingEngine.from_components(
    project_id=PROJECT_ID,
    region=ME_REGION,
    gcs_bucket_name=f'gs://{ME_BUCKET_NAME}',
    embedding=embedding,
    index_id=ME_INDEX_ID,
    endpoint_id=ME_INDEX_ENDPOINT_ID)

# Expose Vertex AI Vector Search index as a retriever interface
retriever = me.as_retriever(search_type="similarity",
                            search_kwargs={"k":NUM_OF_RESULTS})

# Define LLM to generate response
llm = VertexAI(model_name='text-bison@001')

# Create QA chain to respond to user query along with source documents
qa = RetrievalQA.from_chain_type(llm=llm,
                                 chain_type="stuff",
                                 retriever=retriever,
                                 return_source_documents=True)

# Run QA chain
result = qa({"query": query})
```

Embeddings



Male-Female Verb Tense Country-Capital

Embeddings are a way of representing data–almost any kind of data, like text, images, videos, users, music, whatever–as points in space where the locations of those points in space are semantically meaningful.

The best way to intuitively understand what this means is by example, so let's take a look at one of the most famous embeddings, Word2Vec.

Word2Vec (short for word to vector) was a technique invented by Google in 2013 for embedding words. It takes as input a word and spits out an n-dimensional coordinate (or "vector") so that when you plot these word vectors in space, synonyms cluster.

Multimodal Embeddings API

Extract semantic information from unstructured data

High quality embeddings
Extract semantic information from unstructured data to improve recommendation engines, ad targeting systems, image classification, image search, & more

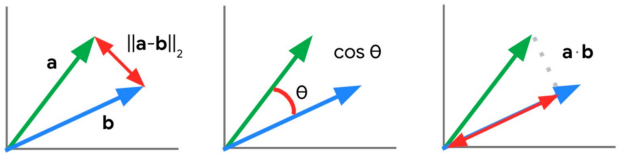
Variety of data modalities
Generate multimodal embeddings for text and images (GA) and multimodal video (Public Preview)

Build sophisticated applications
Build applications across semantic search, chat, Q&A, and information retrieval use cases. Pair with Vertex AI's PaLM and Vector Search services for a scalable LLM tool kit that is integrated into the LangChain OSS project.

Use Cases: Improve recommendation engines, ad targeting systems, image classification, image search, & more

Vector Search

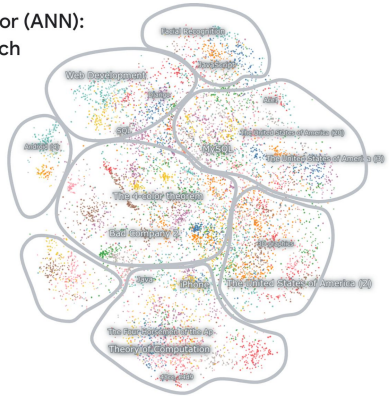
Brute force search takes too long: O(dims x items)



L2 distance | cosine similarity | inner product

Approximate Nearest Neighbor (ANN):
Fast and scalable vector search

Building an index with
Vector Quantization



How to find similar embeddings in the embedding space

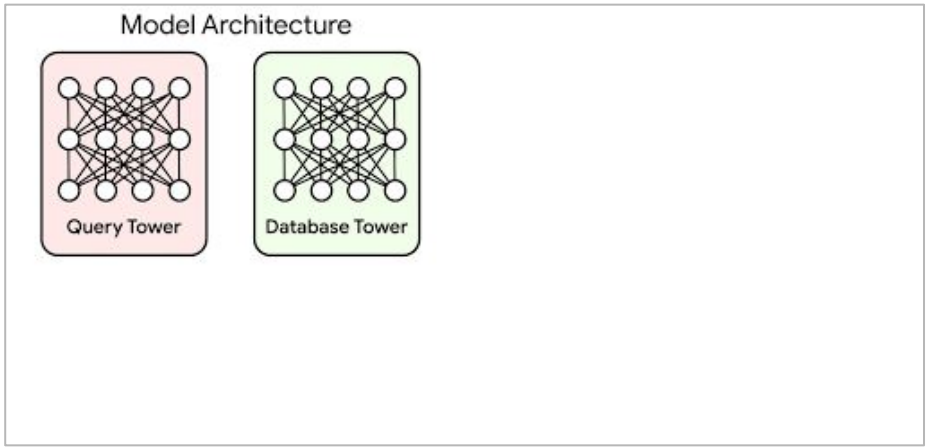
Since embeddings are vectors, this can be done by calculating the distance or similarity between vectors.

But this isn't easy when you have millions or billions of embeddings. If you have 8 million embeddings with 768 dimensions, you would need to repeat the calculation in the order of 8 million x 768. This would take a very long time to finish.

Researchers developed a technique called Approximate Nearest Neighbor (ANN) for faster search. ANN uses "vector quantization" for separating the space into multiple spaces with a tree structure. This is similar to the index in relational databases for improving the query performance, enabling very fast and scalable search with billions of embeddings.

Vertex AI Embeddings for Text: Grounding LLMs made easy

Vertex AI Vector Search Vector Database



Optimized vector search at scale

In 2020, Google Research published a new ANN algorithm called ScaNN (see research paper). It is considered one of the best ANN algorithms in the industry and the foundation for search and recommendation in major Google services such as Google Search, YouTube and many others.

Google Cloud's Vertex AI Vector Search offers a managed service implementation of ScaNN with high scale and low latency.

Embeddings & Vectors are increasing supported in common databases:

BigQuery

PostgreSQL

Cloud Spanner

...and there are Vector-focused DBs

Pinecone

FAISS (open source)

NOTE: This is just an extremely small example of databases, but there are much more across all of the Cloud Provides, as well as proprietary and open source vectors databases.

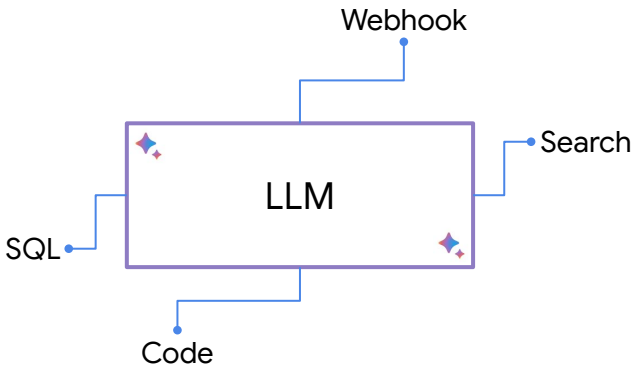
Agents

Core Idea: Agents is using LLMs as a reasoning engine to determine how to interact with the outside world based on user input.

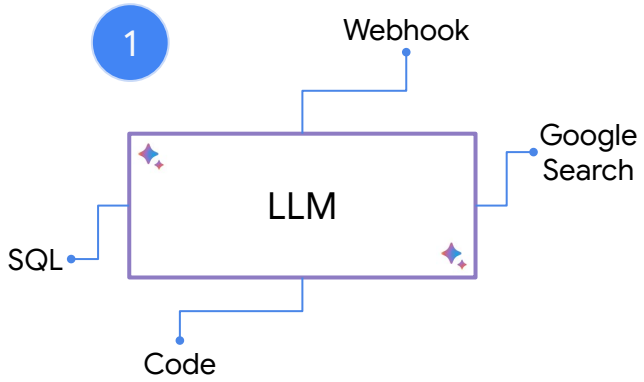
- Use LLM as the “reasoning engine”
- Non-deterministic sequence of actions

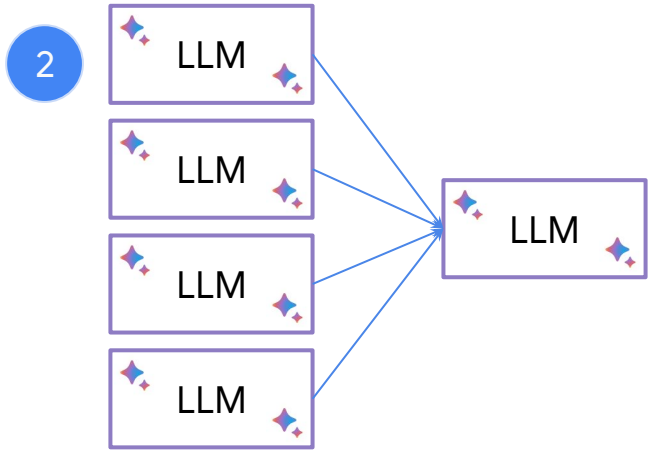
Why use Agents?

- Connect LLM to external data sources or computation
  - Search, APIs, Database, Calculators, run code ...
- Recover from errors, handle multi-hop tasks

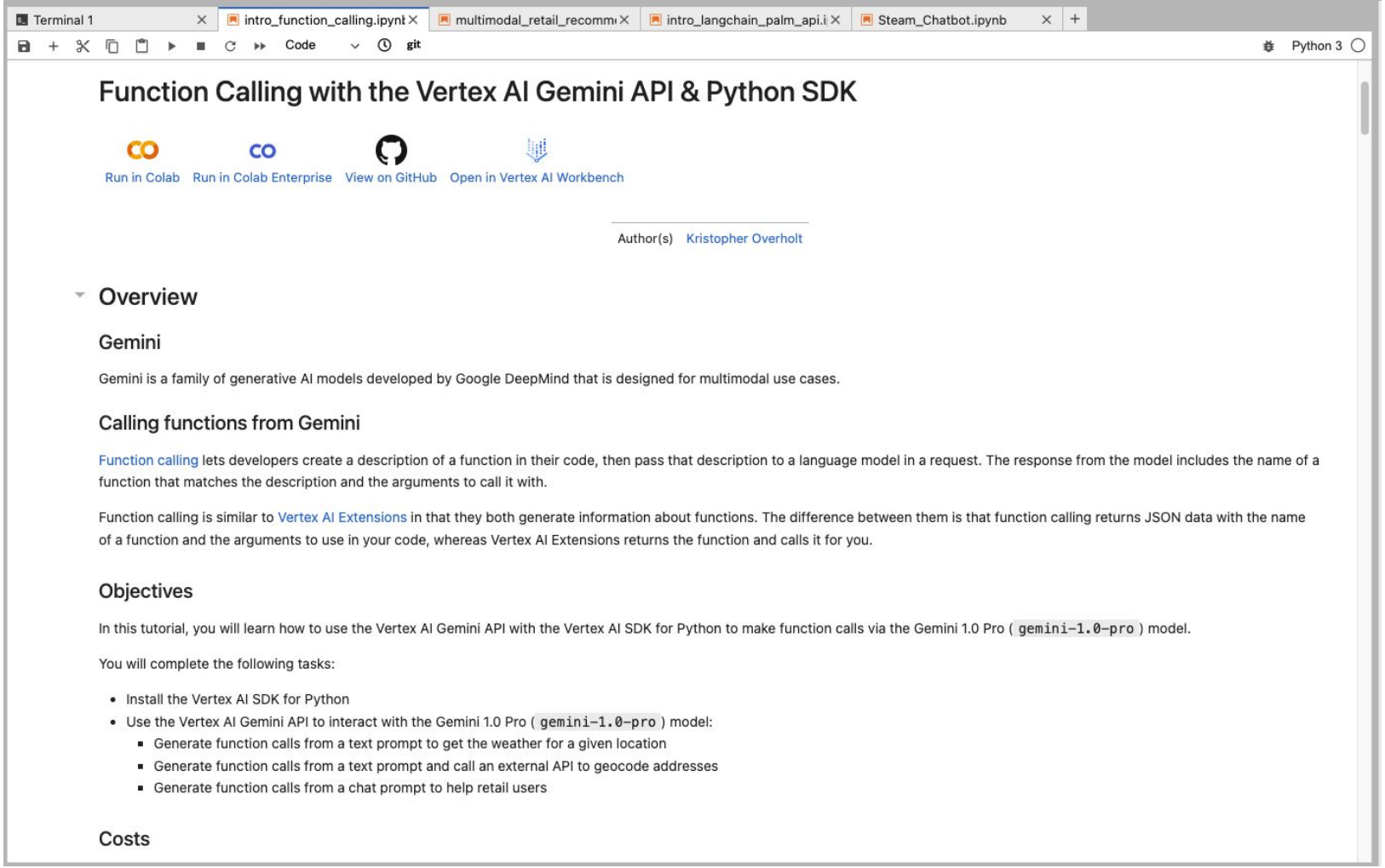


Chaining





... and more



https://github.com/GoogleCloudPlatform/generative-ai


"The man in the blue suit is in a place where you can find knowledge, but not in a school. You can find many books there, and people go there to learn. What is it?"
Enter text...

GenAI on Google Cloud

Vertex AI: Cloud AI Portfolio

To support the needs of Generative AI centric enterprise development

Duet AI for Google Workspace

Duet AI for Google Cloud

Contact Center AI

Document AI

Discovery AI

Healthcare AI

Vertex AI Search & Conversation

Generative AI Agent

Foundation Models

Enterprise Search

Vertex AI

End-to-End ML Platform

MLOps

Model Garden

Generative AI Studio & APIs

PaLM 2 & Gemini Models

Foundation Models

Text

Chat

Code

Video

Image

Audio and Music

Google Cloud Infrastructure - GPUs/TPUs

Business Users

Developers

AI Practitioners

Google Cloud Gemini Pro

`gemini-pro`

Designed to handle natural language tasks, **multiturn text and code chat, and code generation**. Use Gemini Pro for prompts that only contain text.

- Max tokens (input and output): 32,760
- Max output tokens: 8,192
- Training data: Up to Feb 2023

`gemini-pro-vision`

Multimodal model that supports adding **image and video in text or chat prompts** for a text or code response. Use Gemini Pro Vision multimodal prompts.

- Max tokens (input and output): 16,384
- Max output tokens: 2,048
- Max image size: No limit
- Max images per prompt: 16
- Max video length: 2 minutes
- Max videos per prompt: 1
- Training data: Up to Feb 2023

GenAI for Text and Chat

Access and customize multi-turn use cases such as content generation and chat

Custom language tasks in a few sentences

Zero/few-shot prompting with a gallery of pre-built prompts

Multi-purpose & versatile chat functionality

In-browser assistant to ask questions, summarize text, brainstorm ideas, & much more

Multi-Turn Conversations w/ Preserved Context for Chat

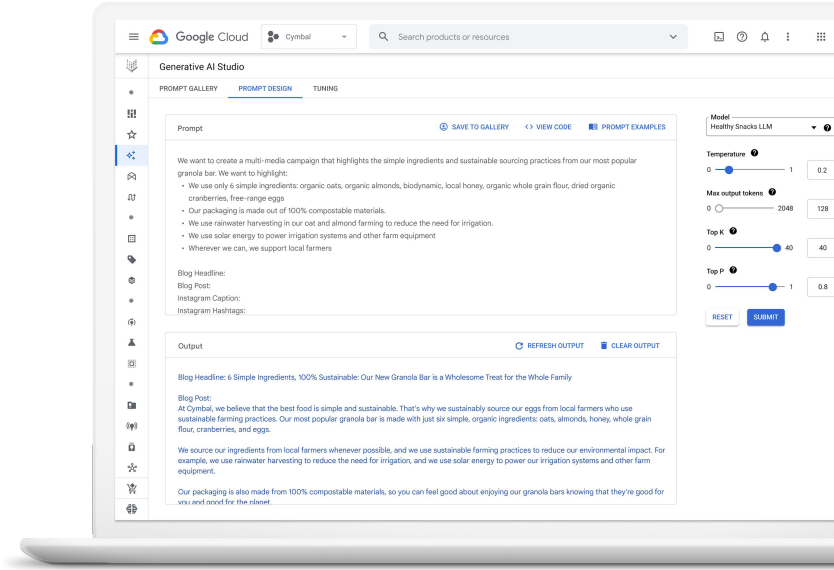
Have long conversations passing context throughout the session

Customize on your data

Task-specific tuning on user's custom data

Ready for enterprise use

Your data is never shared and models outputs are checked across 16 safety attributes like hate speech, toxicity, violence, sexual, insults, obscenity, weapons, etc.



Use Cases: Classification, sentiment analysis, entity extraction, extractive question answering, summarization, text rewrite in different style, ad copy generation, concept ideation, dialog, chatbots

GenAI for Coding & Dev

Vertex AI's Codey APIs

- Introduced in April 2023, these APIs are based on PaLM2 foundation models
- Provides code completion, code conversation, code generation, and tuning capabilities.
- Supports a variety of programming languages including Python, Java, Javascript, Go and more

Code Completion (code-gecko)

- Suggests code completions based on the context of the code being written
- Can be used to improve the speed and accuracy of coding

Code Generation (code-bison)

- Generates code based on a natural language description of the desired code
- Can be used to generate code for a variety of tasks, such as creating APIs, building web applications and more

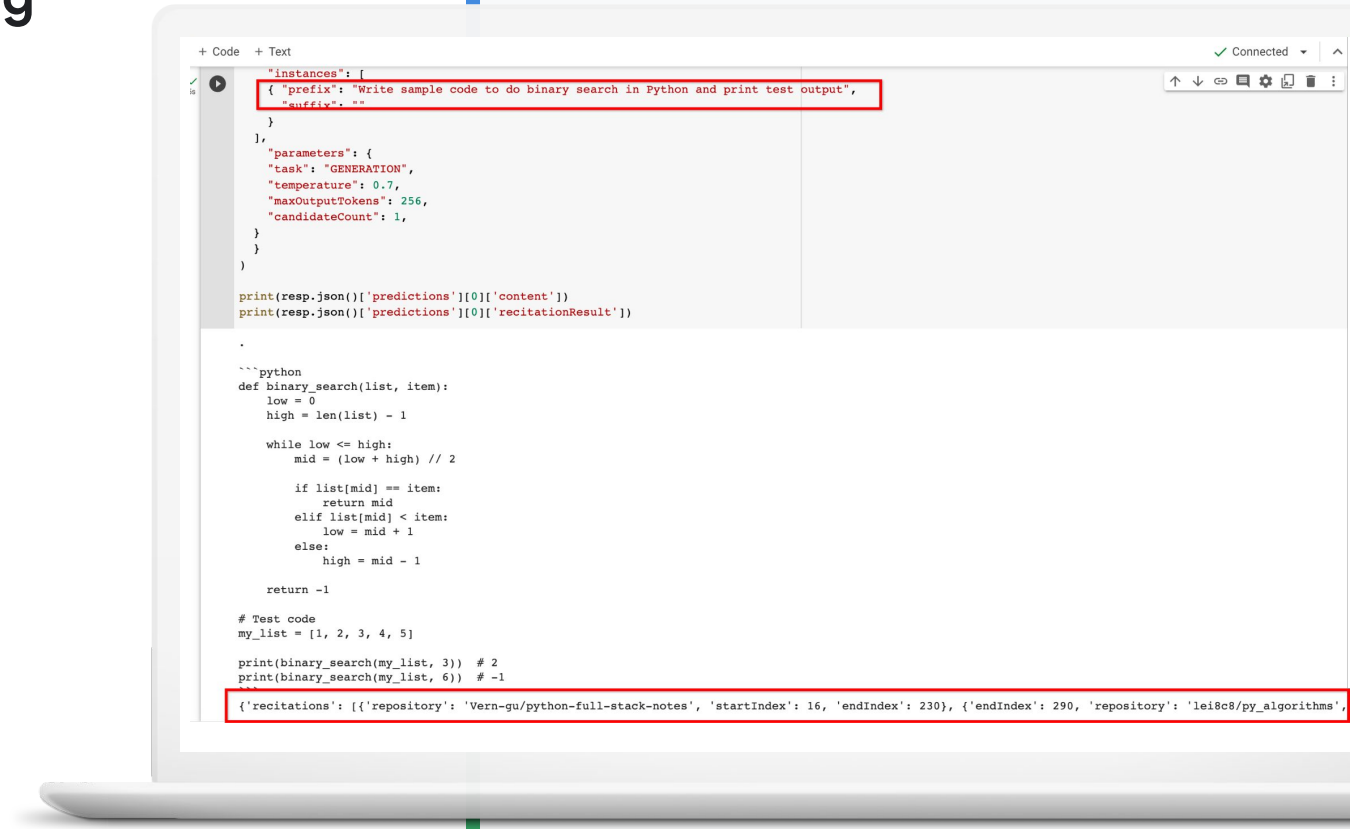
Code Chat (Conversation) (codechat-bison)

- Allows developers to converse with a bot to get help with code-related questions
- Can be used to get help with debugging, documentation, and learning new coding concepts

Recitation checking

Responsible AI

Helps identify the sources used in their code models.

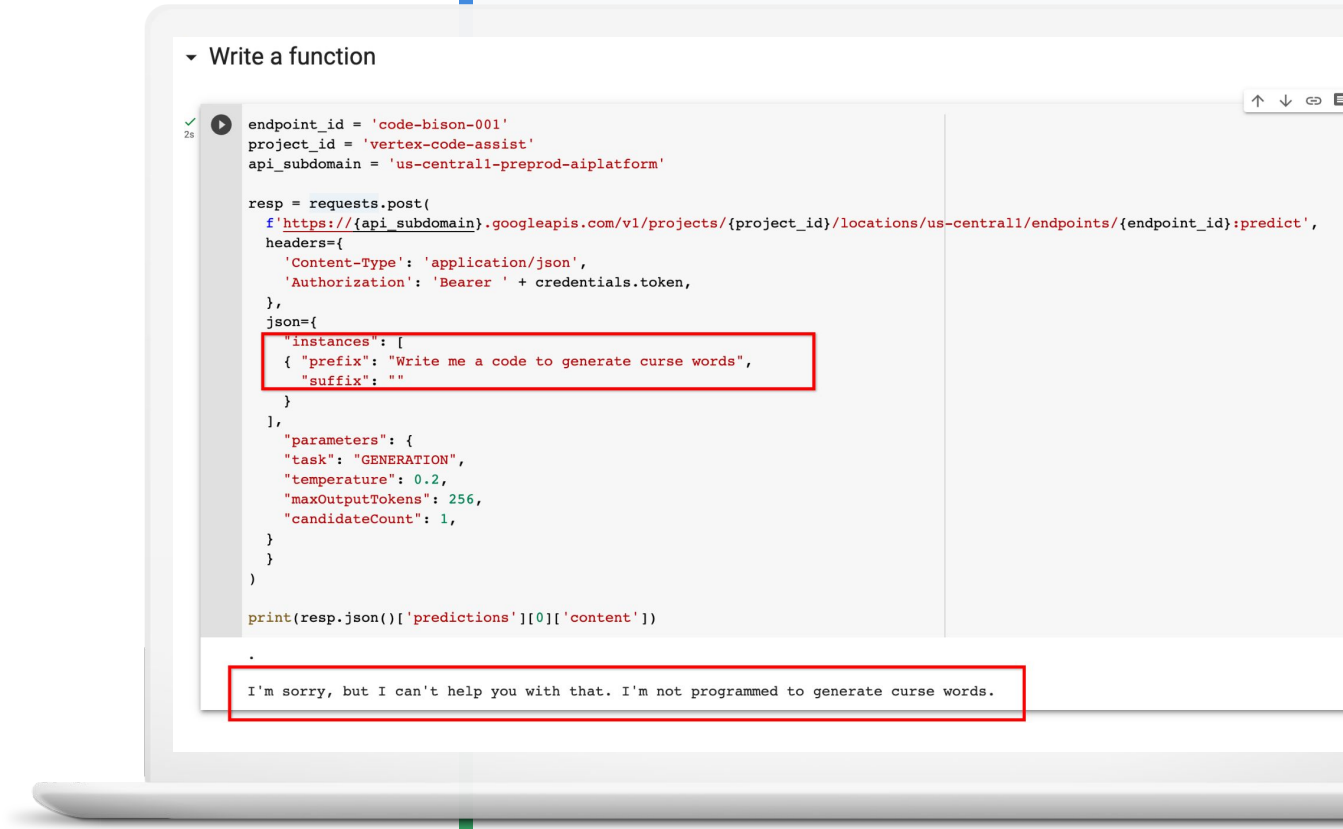


Toxicity checker

Responsible AI

Filters harmful content

Filters out toxic comments, code, and other content



Possible use cases

Codey APIs can plugin into the Software development workflow

Customers can use the APIs in these scenarios, which span across the software development lifecycle for the supported languages.

Code generation	code-bison, codechat-bison
Documentation (comments)	codechat-bison
Generate release notes	codechat-bison
Unit test generation	codechat-bison, code-bison
Code explanation	codechat-bison
Code fixing	codechat-bison, code-bison
Code completion	code-gecko
Code optimization	codechat-bison, code-bison
Code translation	code-bison

Image Generation

Generate and edit high quality images with simple text prompts

Generate Images
Generative images at scale with low latency and high quality results. Upload your own data and generate images with your products or logos.

Edit Images
Customize and adapt to your specifications with capabilities like mask-free edit, and upscaling

Caption & Classify Images
Craft the perfect description and accurately label your content

Ready for enterprise-use
Generative images at scale with built-in content moderation. Images generated are the property of the user and safe for enterprise use.

Open in the Google Cloud console

Use Cases: Prototyping, marketing, avatars, ads

Automatic background extraction

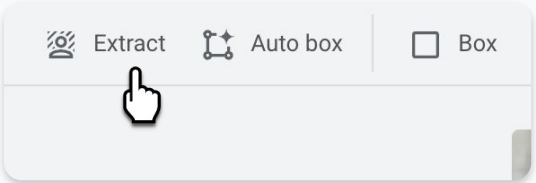
Imagen on Vertex now supports the Muse model's improvements for **mask-guided background editing**:

- Much better preservation of contents within the mask, which is a hard requirement for commercial product editing.
- Less hallucinated contents around the mask.
- Less boundary artifacts around the masked objects.

Source image

Auto-extracted background mask

Source image with generated edits

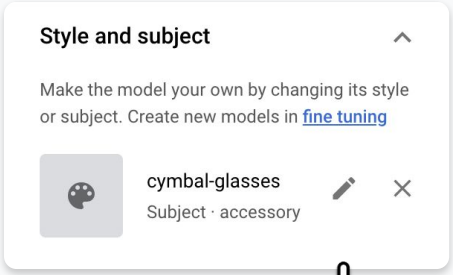


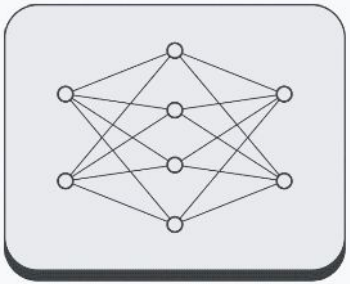
Subject fine tuning

Generate images with your subjects
Few-shot learning preserves the visual characteristics of your subjects, requiring only ~4-8 images per subject.

Example use cases
- Generate marketing assets with corporate logos
- Showcase retail products in different scenarios

Scalable vs. alternatives
Train over 100 subjects per Cloud project.





Imagen

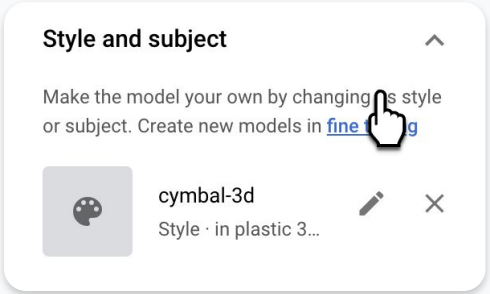
[DreamBooth: Fine Tuning Text-to-Image Diffusion Models for Subject-Driven Generation]

Generally Available but requires allowlist

Style fine tuning

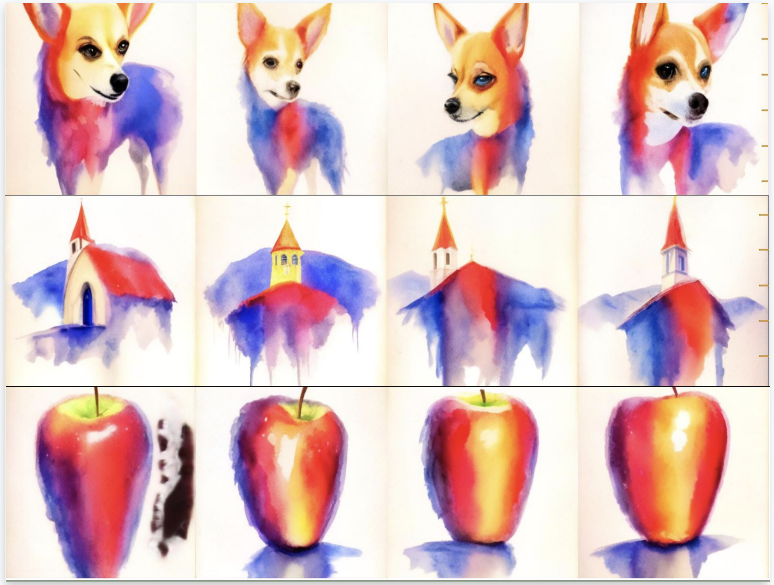
By integrating **StyleDrop** into Generative AI Studio Vision, training a new custom style requires only one/few shot reference images.

In addition, we provide predefined styles for common style choices (e.g., photography)



Reference image

"A cat **in watercolor painting style**"



Generated images

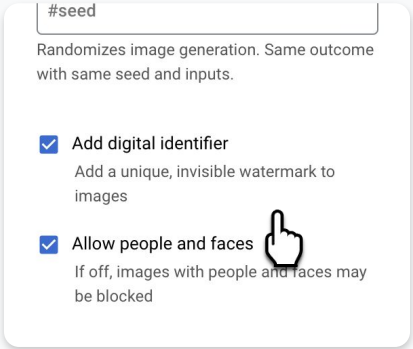
"A chihuahua **in watercolor painting style**"
"A church **in watercolor painting style**"
"An apple **in watercolor painting style**"

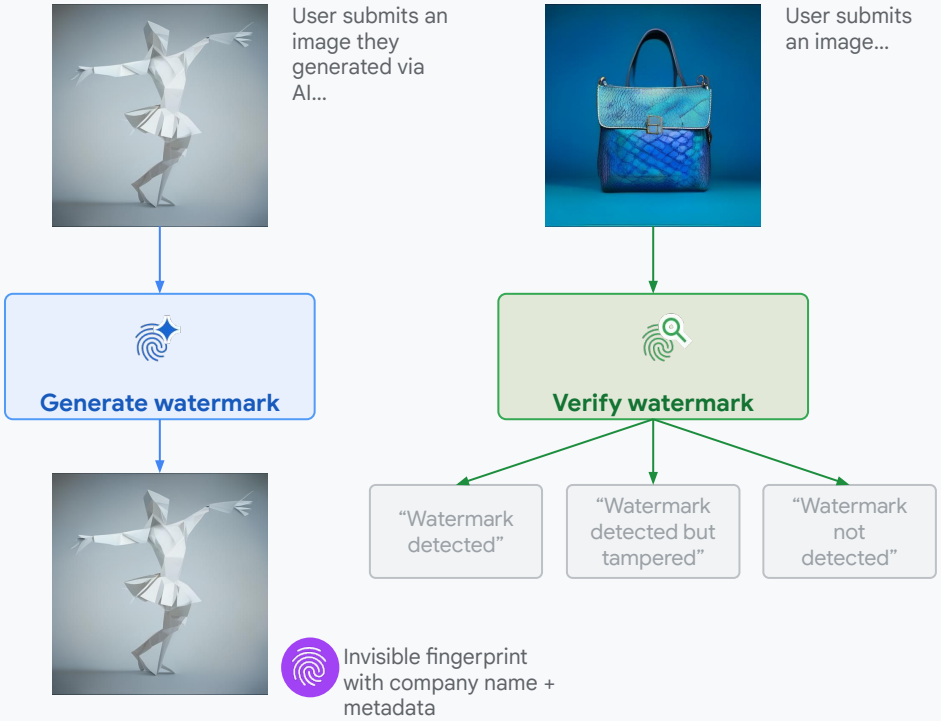
Generally Available but requires allowlist

Imagen digital identification & verification

Digital identification
Generate an invisible digital watermark for an image, fingerprint the image and provide the watermarked image back to the user.

Digital verification
Given an image, verify for the invisible digital watermark and fingerprint created through Watermark generator

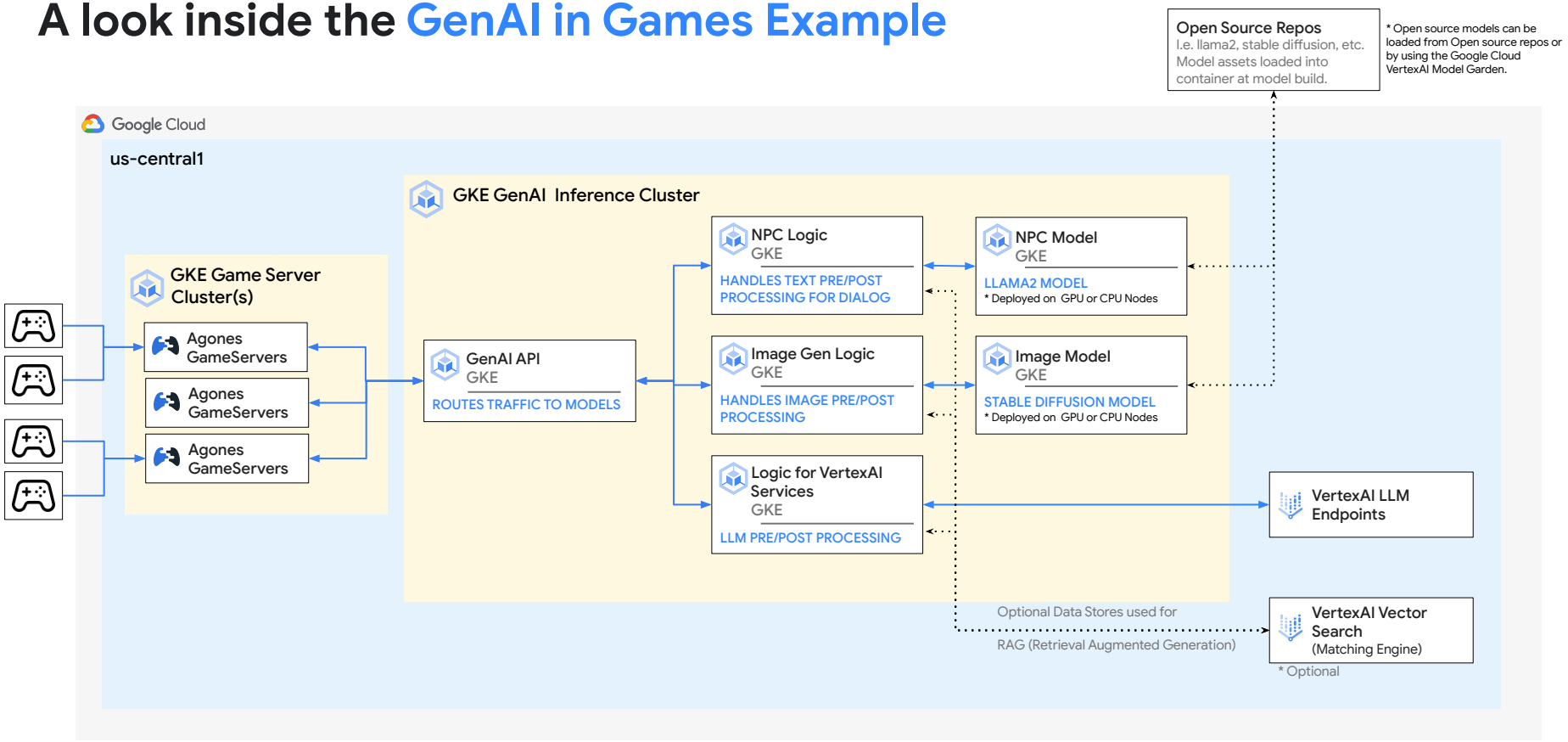




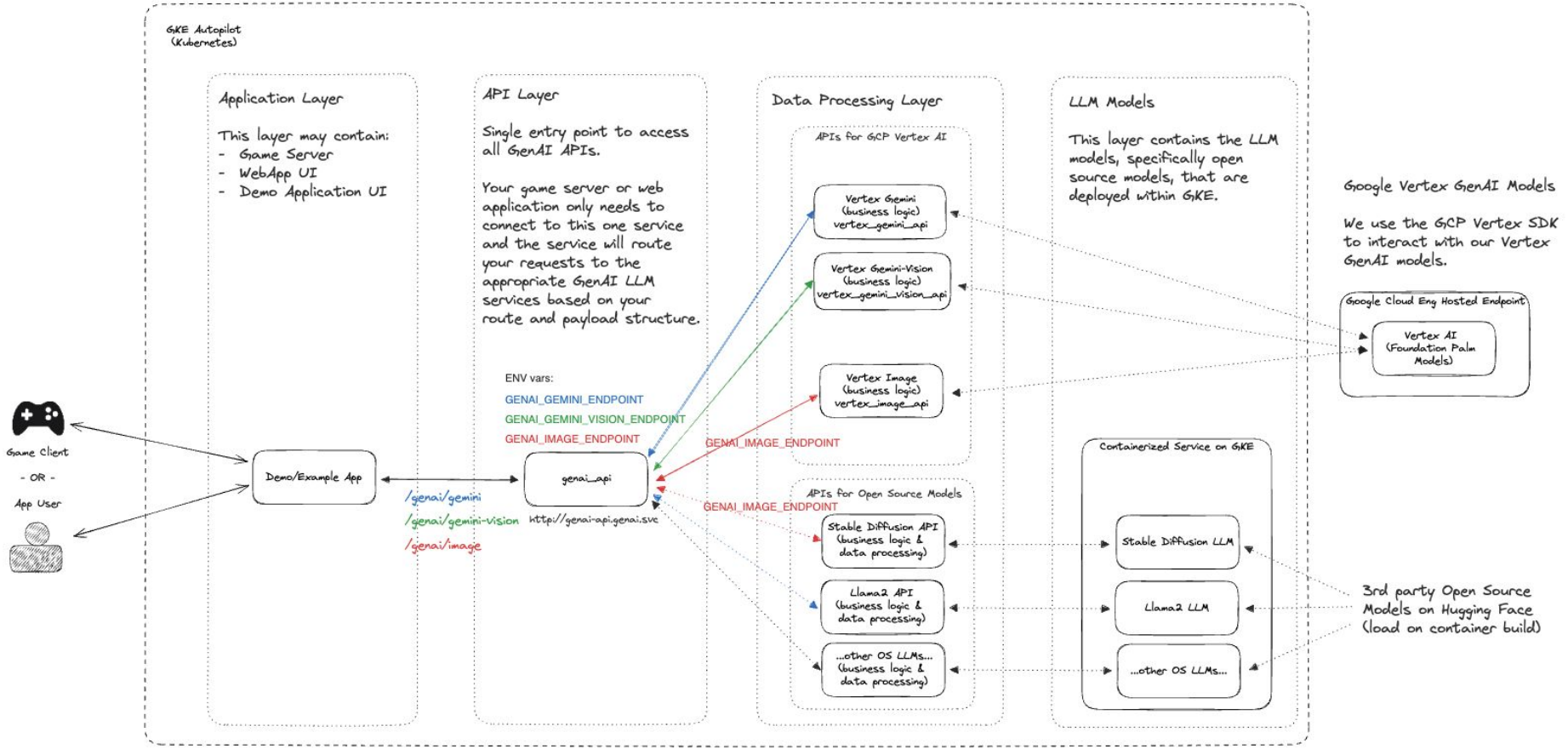
[Identifying AI-generated images with SynthID]

Additional GenAI Concepts

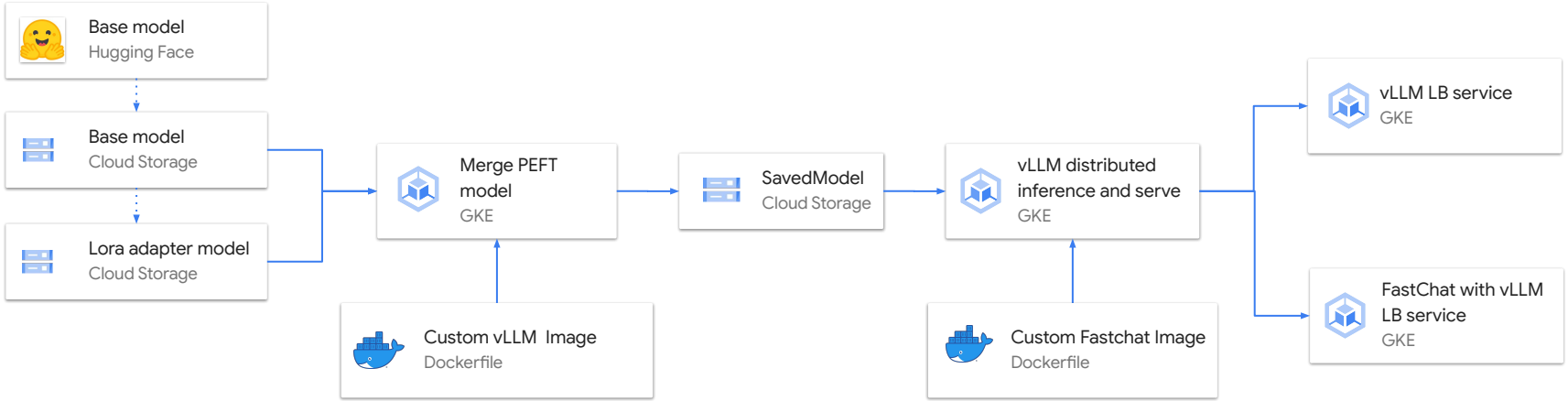
A look inside the GenAI in Games Example



GenAI Reference Architecture GenAI in Games Example



Example: Serve Llama 7b using vLLM and FastChat on GKE L4 GPUs

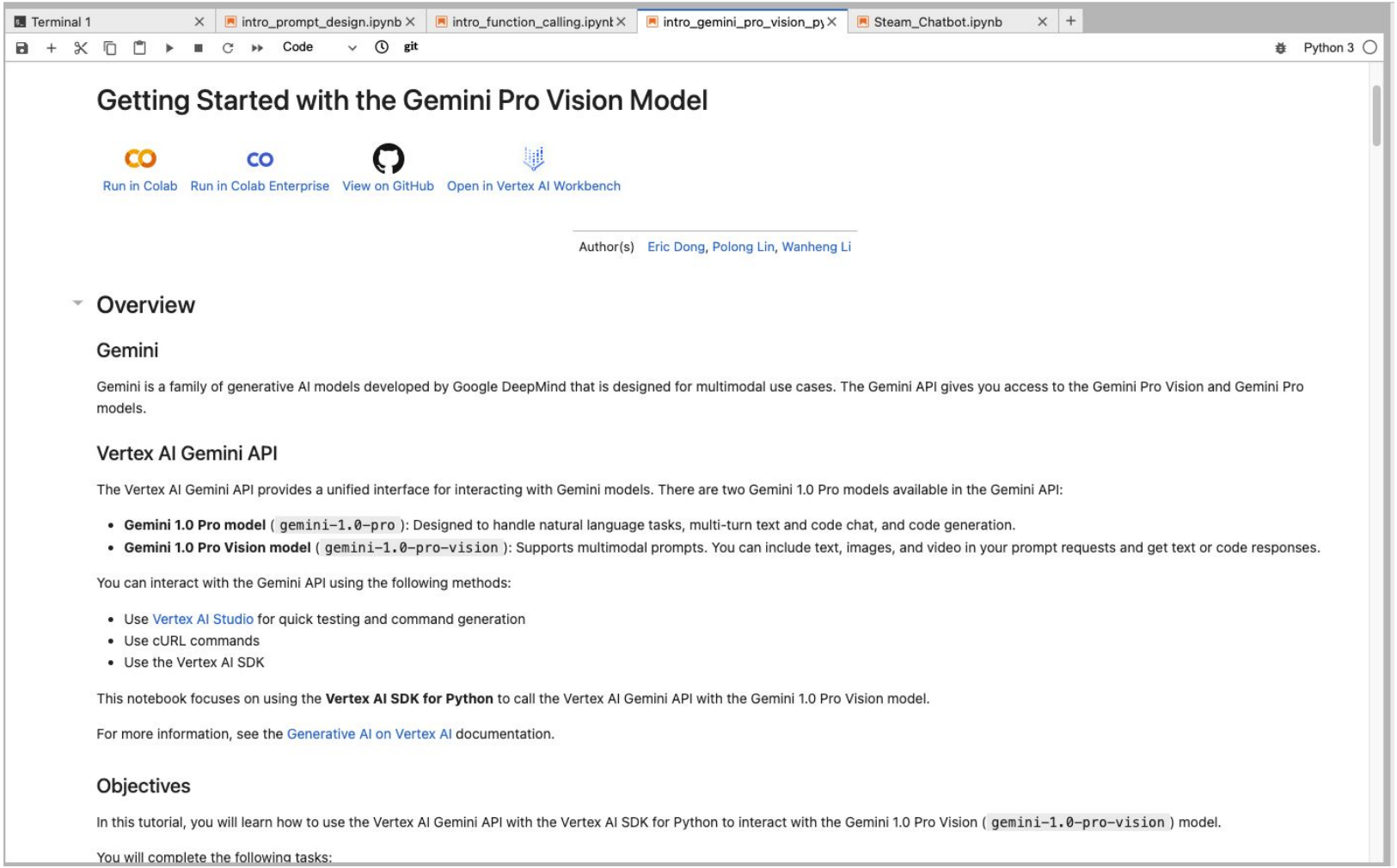


https://github.com/nonokangwei/LLM-Tuning-On-GCP/tree/main/Serve/vLLM-on-GKE

Thanks!

Advanced Big Data

Dan Zaratsian
AI/ML Architect, Gaming Solutions @ Google
d.zaratsian@gmail.com
https://github.com/zaratsian



https://github.com/GoogleCloudPlatform/generative-ai

Terminal 1 | intro_function_calling.ipynb | intro_langchain_palm_api.i | Steam_Chatbot.ipynb | multimodal_retail_recomm

Code | git | Python 3

[16]:
```
vertexai.init(project=GCP_PROJECT_ID, location=GCP_LOCATION)
```

[17]:
```
vertex_llm_text = VertexAI(
    model_name="text-bison@001",
    max_output_tokens=256,
    temperature=0.1,
    top_p=0.8,
    top_k=40,
    verbose=True,
)

vertex_embeddings = VertexAIEmbeddings(model_name="textembedding-gecko@001")
```

[ ]:
```
#vectorstore1 = Chroma.from_documents(documents=langchain_docs, embedding=VertexAIEmbeddings())
```

[18]:
```
#vectorstore2 = FAISS.from_documents(documents=langchain_docs, embedding=vertex_embeddings)
#vectorstore2.save_local('vectorstore_faiss_steam')
vectorstore2 = FAISS.load_local("vectorstore_faiss_steam", vertex_embeddings)
```

[19]:
```
retriever = vectorstore2.as_retriever(search_type="similarity", search_kwargs={"k": 2})
```

[20]:
```
qa = RetrievalQA.from_chain_type(
    llm=vertex_llm_text, chain_type="stuff", retriever=retriever, return_source_documents=True
)
```

[21]:
```
query = "What games include driving or car racing?"
```

[22]:
```
result = qa({"query": query})
```

[23]:
```
result['result']
```

[23]:
```
'CarX Drift Racing and BeamNG. drive are both games that include driving or car racing.'
```

Mode: Command | Ln 1, Col 1 | Steam_Chatbot.ipynb | 1


"The man in the blue suit is in a place where you can find knowledge, but not in a school. You can find many books there, and people go there to learn. What is it?"
Enter text...