Build Generative AI Agents with Vertex AI and Flutter

Google Cloud

Welcome to this course on building generative AI agents with Vertex AI and Flutter.

In this course, you learn how to develop an app using Flutter, Google's portable UI toolkit, and integrate the app with Gemini, Google's family of generative AI models. You also use Vertex AI Agent Builder, Google's platform for building and managing AI Agents and applications.

The course consists of two modules:

1. **Flutter, Gemini, and Vertex AI** - This is the first module of the course that contains four lessons that provide an introduction to the module, and an overview of Generative AI and Gemini, Vertex AI, and Flutter.
2. **Generative AI on Vertex AI** - This module contains five lessons that include an introduction to the module, and lessons where you learn about Generative AI on Vertex AI, Vertex AI Agent Builder, Vertex AI Search, and using a Reasoning Engine Agent in Gen AI applications.

Click the **Start Course** button to begin.

INTRODUCTION

What's in It for Me?

This course helps you learn about the Gemini models and their use in generative AI applications. You learn about some of the tools that are available on the Vertex AI platform that are used to build search and reasoning agent capabilities for your Gen AI apps.

You also learn about Flutter, a UI framework that is used to build applications for mutliple platforms from a single codebase.

Let's review the course learning objectives.

Learning objectives

✓ Describe generative AI and the use of models and tools to build Gen AI applications.

✓ Implement Gen AI features in a Flutter app by integrating with Gemini and a Reasoning Engine agent.

Module 1 Introduction

In this module

You'll learn about Flutter for app development, Gemini and its use in generative AI, and Vertex AI, which are the underlying frameworks and platforms that are used to build Gen AI apps.

Let's review the module's learning objectives.

Learning objectives

- ✓ Describe the use of Gemini models for Gen AI.
- ✓ Describe the Vertex AI platform.
- ✓ Describe how Flutter can be used to develop apps.

Generative AI and Gemini Overview

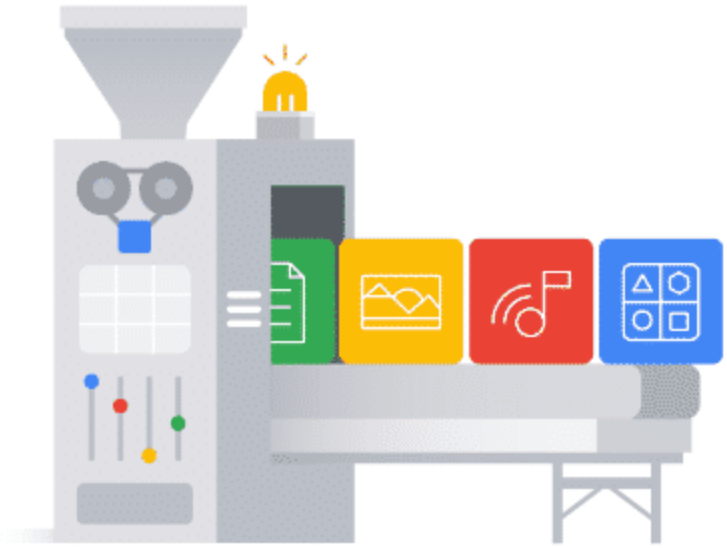
Generative AI refers to the use of AI to create new content, like text, images, music, audio, and videos.

Generative AI

Generative AI:

- Is a type of artificial intelligence (AI) technology that can produce various types of content, including text, imagery, audio, and synthetic data.
- Is powered by foundation models (large AI models) that can perform many tasks, such as text summarization, Q&A, classification, translation, image generation, and more.

- Works by using a machine learning (ML) model to learn the patterns and relationships in a dataset of human-created content. It then uses the learned patterns to generate new content.



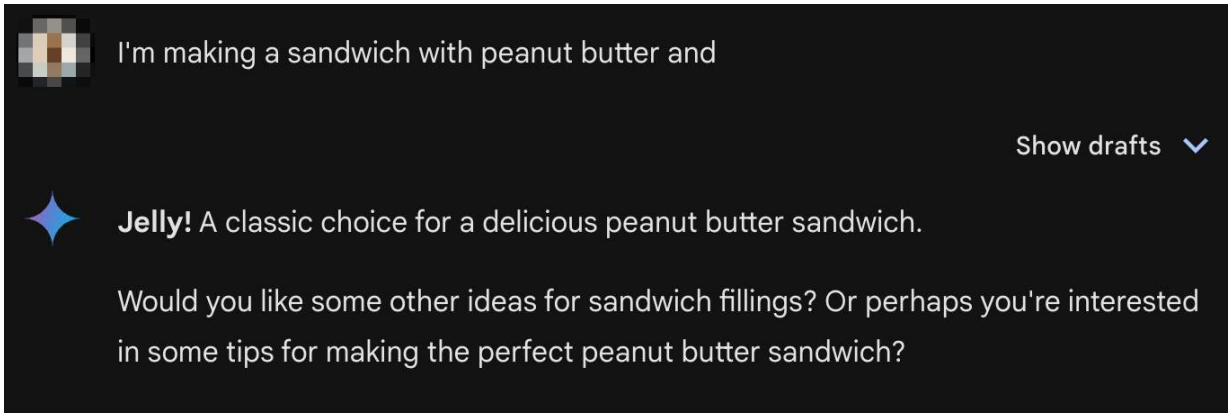
Generative AI

Generative AI models

- The process of learning from existing content is called training, and results in the creation of a statistical model.

- When given a prompt, Gen AI uses this statistical model to predict what an expected response might be, and this generates new content.
- Generative language models learn about patterns in language through training data. Then, given some text, they predict what comes next.
- Generative image models produce new images using techniques like diffusion. Then, given a prompt or related imagery, they transform random noise into images or generate images from prompts.

Here is an example using Gemini, Google's family of large language models, which are able to communicate and generate human-like text in response to a wide range of prompts and questions.



Gemini is a family of generative AI models from Google that is designed for multimodal use cases.

Gemini models

Here is the current set of Gemini models from Google:

Click each flashcard to learn more.

Gemini 1.0 Pro	The 1.0 Pro model supports a wide range of tasks using text as input.
Gemini 1.0 Pro Vision	The 1.0 Pro Vision model supports text, image, and video as inputs for a broad range of applications.

Gemini 1.5 Pro	The 1.5 Pro multimodal model supports adding image, audio, video, and PDF files in text or chat prompts for a text or code response. This model supports long-context understanding up to the maximum input token limit.
Gemini 1.5 Flash	The 1.5 Flash multimodal model supports text, code, image, audio, video, and PDF data input, and provides speed and efficiency for high-volume, cost-effective applications.

Gemini 1.0 Ultra	1.0 Ultra is a text model that is optimized for complex tasks, including instruction, code, and reasoning.
Gemini 1.0 Ultra Vision	1.0 Ultra Vision is a multimodal model that supports joint text, images, and video inputs.

For more information on the Gemini family of models, please read the documentation.

The Gemini models are part of a growing list of foundation models that you can test, deploy, and customize for use in your AI-based applications. The models are available and accessible on Vertex AI, Google's ML platform for training and deploying ML models and AI applications.

We'll review Vertex AI in the next lesson of this module.

Vertex AI Overview

What is Vertex AI?

A machine learning (ML) platform used to train and deploy ML models and AI applications.

Vertex AI

- With Vertex AI, you can use and customize large language models (LLMs), including Gemini in your AI-powered applications.
- You can access foundation models in model garden, tune models using a simple UI on Vertex AI Studio, or use models in a data science notebook.

- Vertex AI combines data engineering, data science, and ML engineering workflows, enabling your teams to collaborate using a common toolset, and scale your applications on Google Cloud.

Benefits of using Vertex AI

Some of the benefits of using Vertex AI are listed below.

To learn more, click each button to expand the item.

Access to Gemini and other models

Vertex AI provides access to Gemini and other models where you can prompt and test them with text, images, video, or code.

You can customize models for your use case with a variety of tuning options which helps simplify prompts, and reduce the cost and latency of your requests.

Open and integrated AI platform

Vertex AI is an open and integrated AI platform that provides data scientists with tools for training, tuning, and deploying ML models.
You can reduce training time and deploy models to production easily with a choice of open source frameworks.

MLOps tools for predictive and generative AI

MLOps is a set of practices that improves the stability and reliability of your ML systems by keeping your models relevant with changing data and optimal performance.

Vertex AI MLOps provide modular tools to help you collaborate across AI teams, and improve your models throughout the development lifecycle with predictive model monitoring, alerting, diagnosis, and actionable explanations.

Vertex AI Agent Builder

Vertex AI Agent Builder enables developers to easily build and deploy generative AI experiences, using natural language or a code-first approach.

With Vertex AI Agent Builder, you can create generative AI agents and applications grounded in your organization's data.

Features of Vertex AI

Here are some of Vertex AI's features. For a full list of features and more details, please read the documentation.

- Developers can build generative AI features into applications by accessing Gemini and other models using APIs.
- The Vertex AI Model Garden contains a variety of first and third party models, as well as open models, such as Gemma and LLama 3.
- Vertex AI notebooks are natively integrated with BigQuery providing a single surface across all data and AI workloads.

- Vertex AI Agent Builder provides a no-code agent builder console with powerful grounding, orchestration, and customization capabilities.

In the next module, we'll discuss Vertex AI Agent Builder in more detail.

Flutter Overview

What is Flutter?

Flutter is a free, open source, portable UI toolkit from Google that is used to create natively compiled applications for mobile, web, and desktop from a single codebase.

Flutter

- Flutter is powered by Dart, a language that is optimized for running apps on any platform.

- Flutter is an open source framework and supported by Google.
- With Flutter, you can build adaptive and responsive apps.

Benefits of using Flutter

Some of the many benefits that can be gained by using Flutter for app development are listed below.

To learn more, click each button to expand the item.

Develop and deploy apps quickly

Flutter lowers the bar to entry for building apps by accelerating app development with reduced cost and complexity of app production across platforms.

Convert concepts to production code with ease

Flutter provides a canvas for high-end user experiences that lets you easily turn concepts into production code.

Run apps on multiple platforms

Flutter lets you build apps for multiple platforms out of a single codebase, reducing the time for feature development and release updates.

Flutter apps can run on mobile (Android, iOS), web, and desktop platforms.

Support a variety of platform integrations and services

Flutter's package ecosystem supports a wide variety of platform integrations (such as camera, GPS, network, and storage), and services (such as payments, cloud storage, authentication, and ads).

Features of Flutter

Here are some of the features of Flutter:

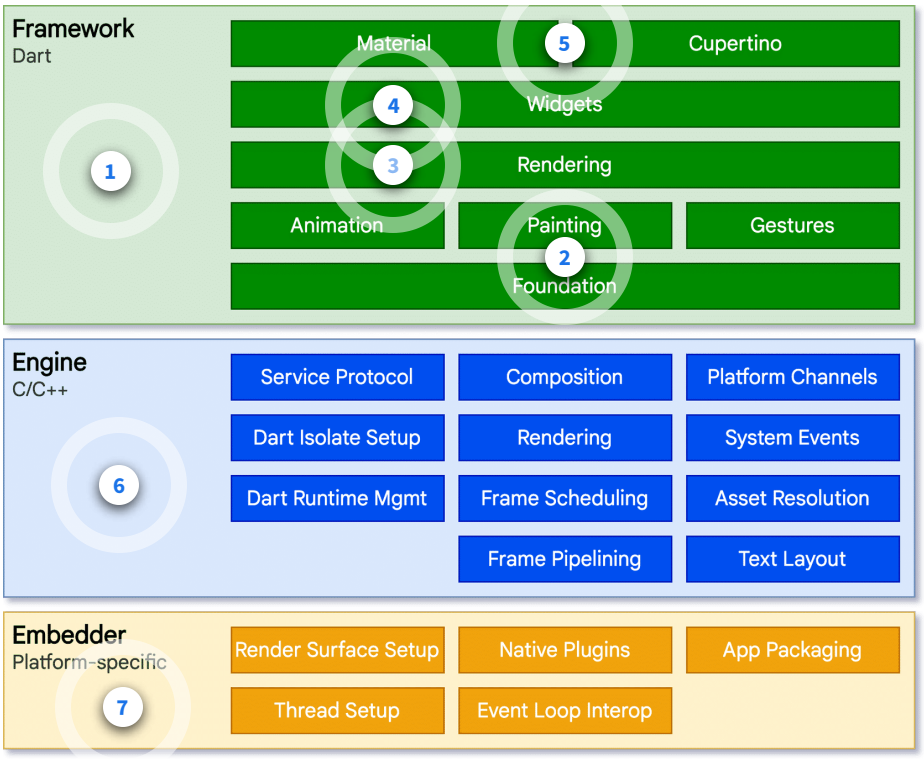
- Flutter doesn't rely on web browser technology nor the set of widgets that ship with each device, but instead, it uses its own high-performance rendering engine to draw widgets.
- Flutter apps (compositing, gestures, animation, framework, widgets, etc.) are written in Dart, a modern, concise, object-oriented language.
- Flutter includes an optimized, mobile-first rendering engine and a modern react-style framework with a rich set of widgets.
- The Flutter SDK also includes APIs for unit and integration tests, APIs to connect to third-party systems, and tools to create, build, test, debug, and profile your apps.

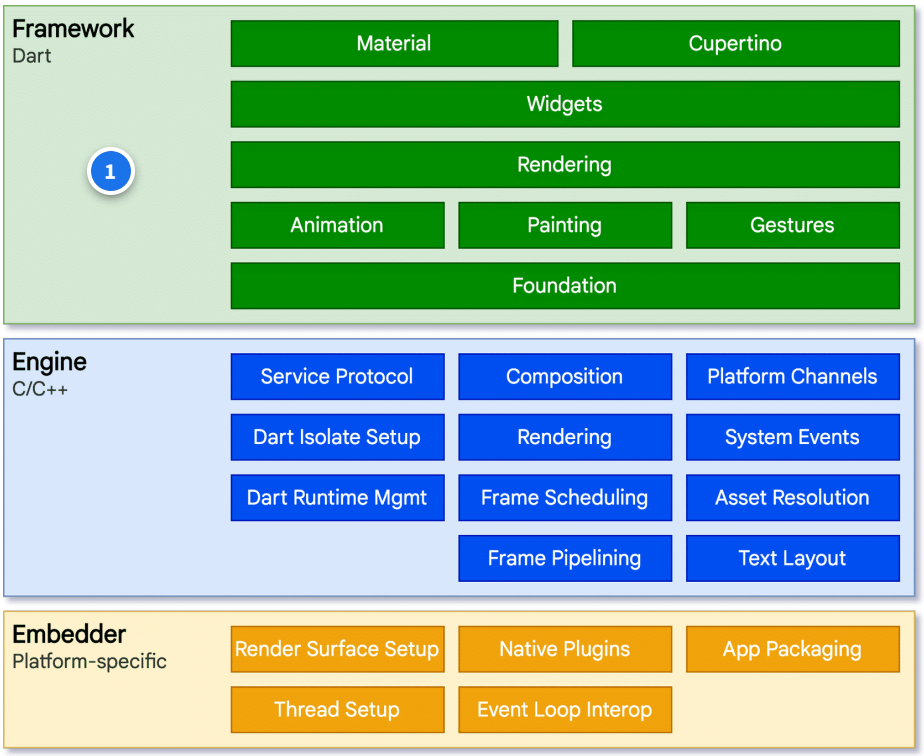
Flutter architecture

Flutter is a cross-platform UI toolkit that is designed to allow code reuse across operating systems, such as iOS and Android, while also allowing applications to interface directly with underlying platform services.

Flutter is designed as an extensible, layered system. It exists as a series of independent libraries that each depend on the underlying layer.

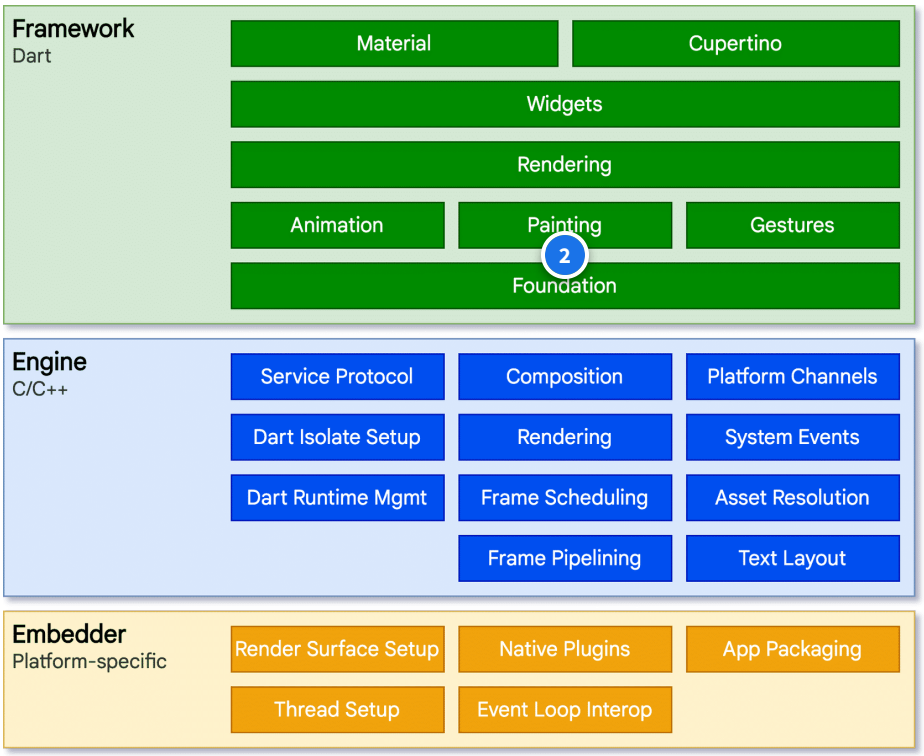
Click each marker to learn more.





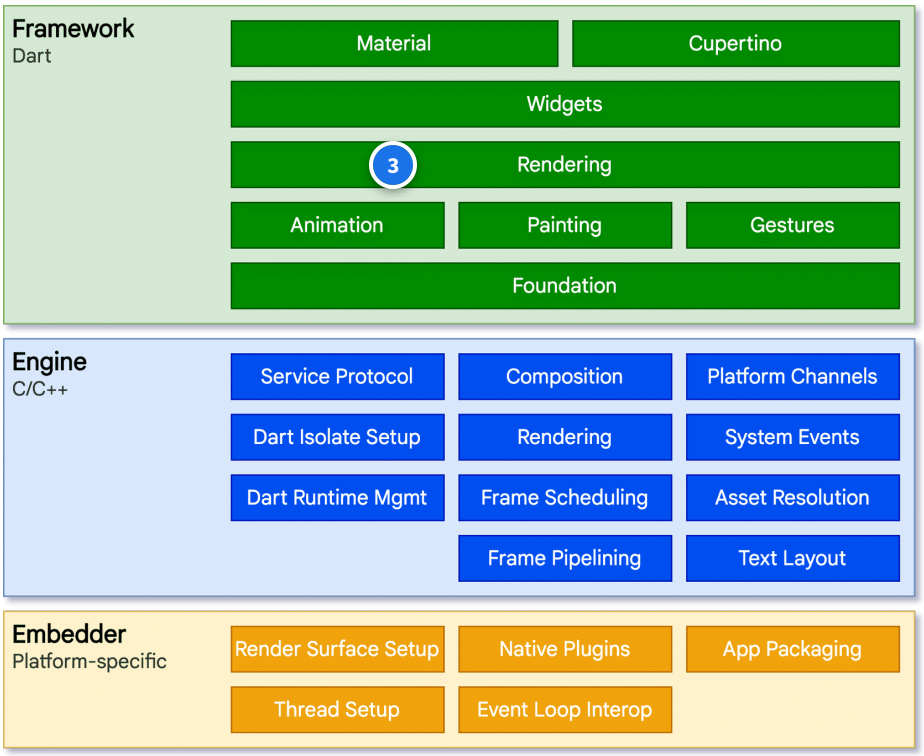
Flutter framework

Developers interact with Flutter through the Flutter framework, which provides a modern, reactive framework written in the Dart language. It includes a rich set of platform, layout, and foundational libraries, composed of a series of layers.



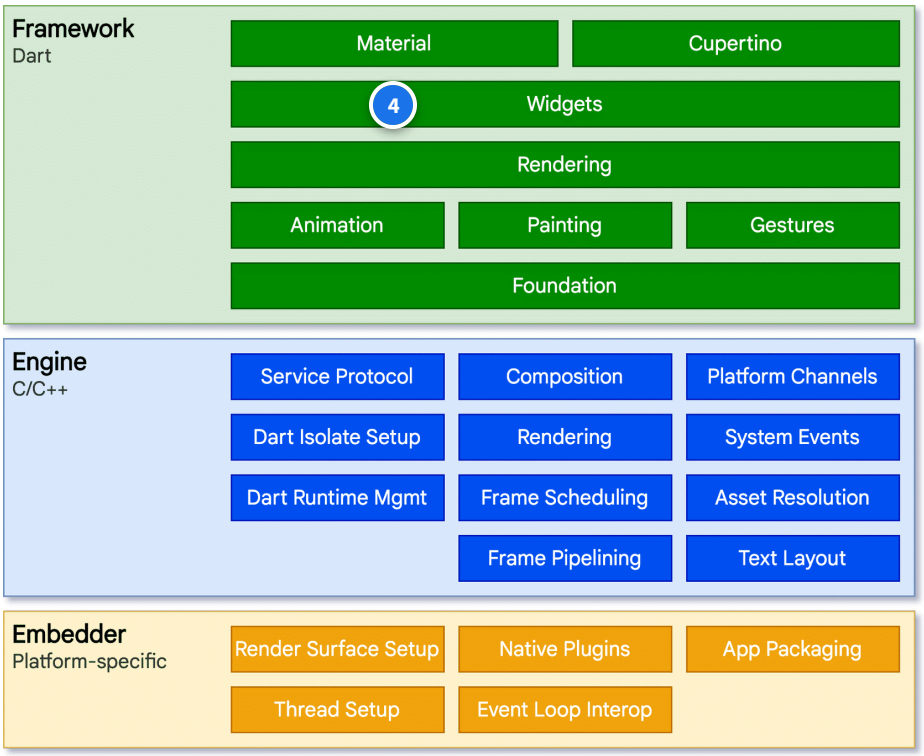
Foundation layer

Basic foundational classes and building block services for animation, painting, and gestures.



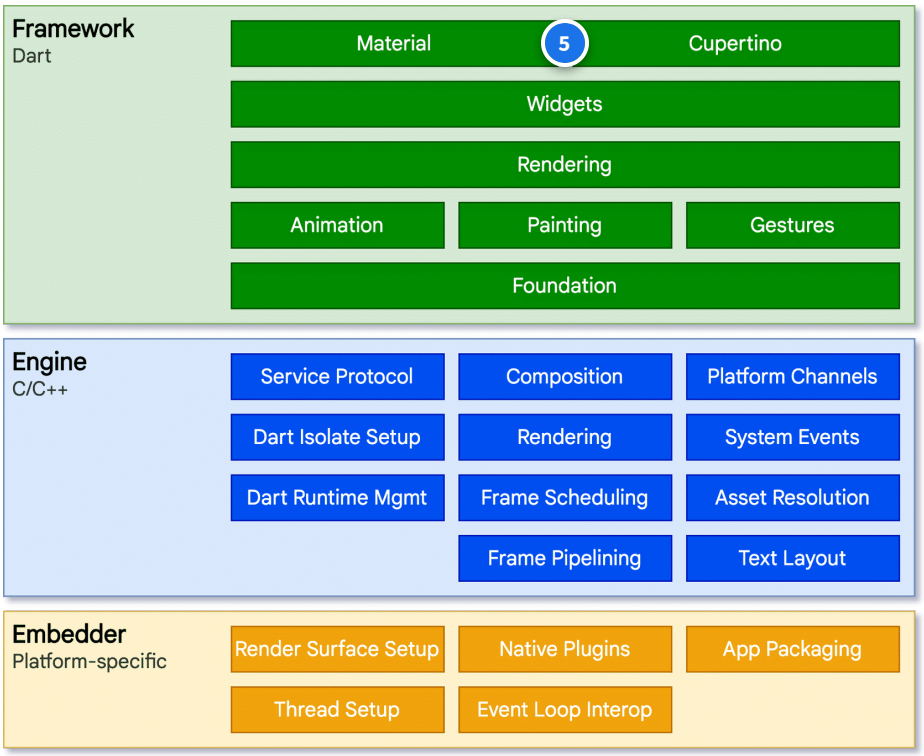
Rendering layer

The Rendering layer provides an abstraction for handling object layout.



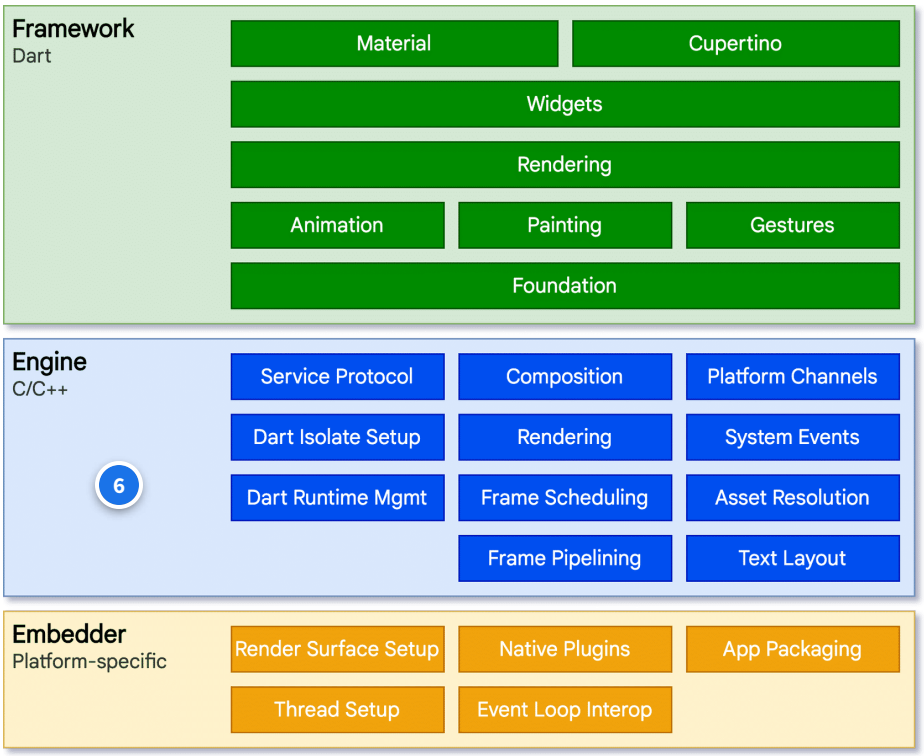
Widgets layer

The Widgets layer is used for object composition, and is where the reactive programming model is introduced.



Material and Cupertino libraries

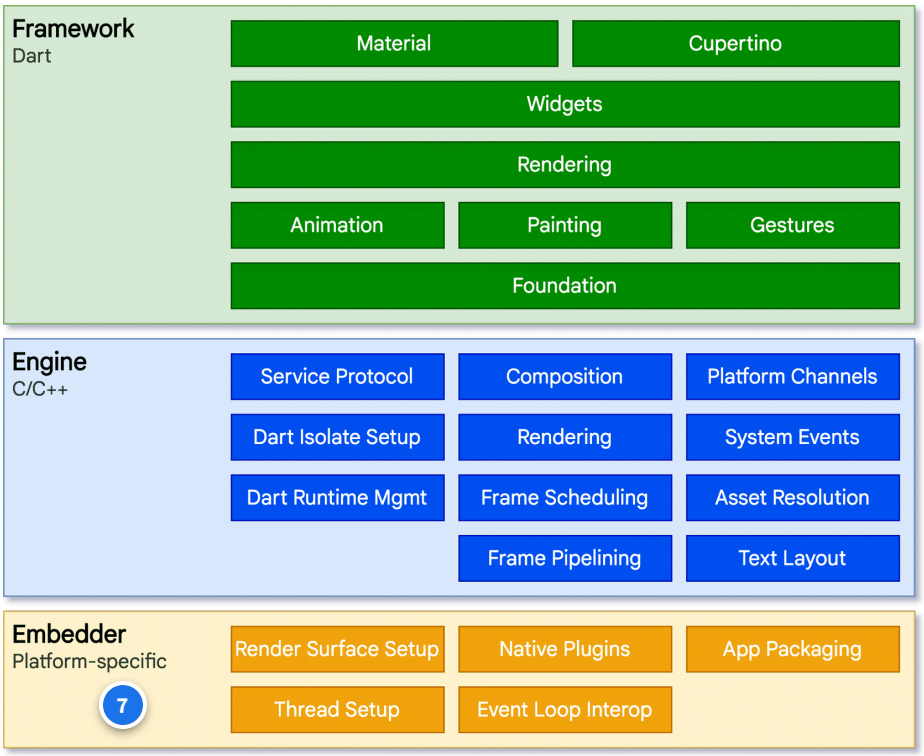
The Material and Cupertino libraries offer comprehensive sets of controls that use the widget layer's composition primitives to implement the Material or iOS design languages.



Flutter engine

The Flutter engine supports the primitives necessary to support all Flutter applications. The engine provides the low-level implementation of Flutter's core API, including graphics, text layout, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain.

The engine is exposed to the Flutter framework through dart:ui, which wraps the underlying engine code in Dart classes. This library exposes the lowest-level primitives, such as classes for driving input, graphics, and text rendering subsystems.



Embedder

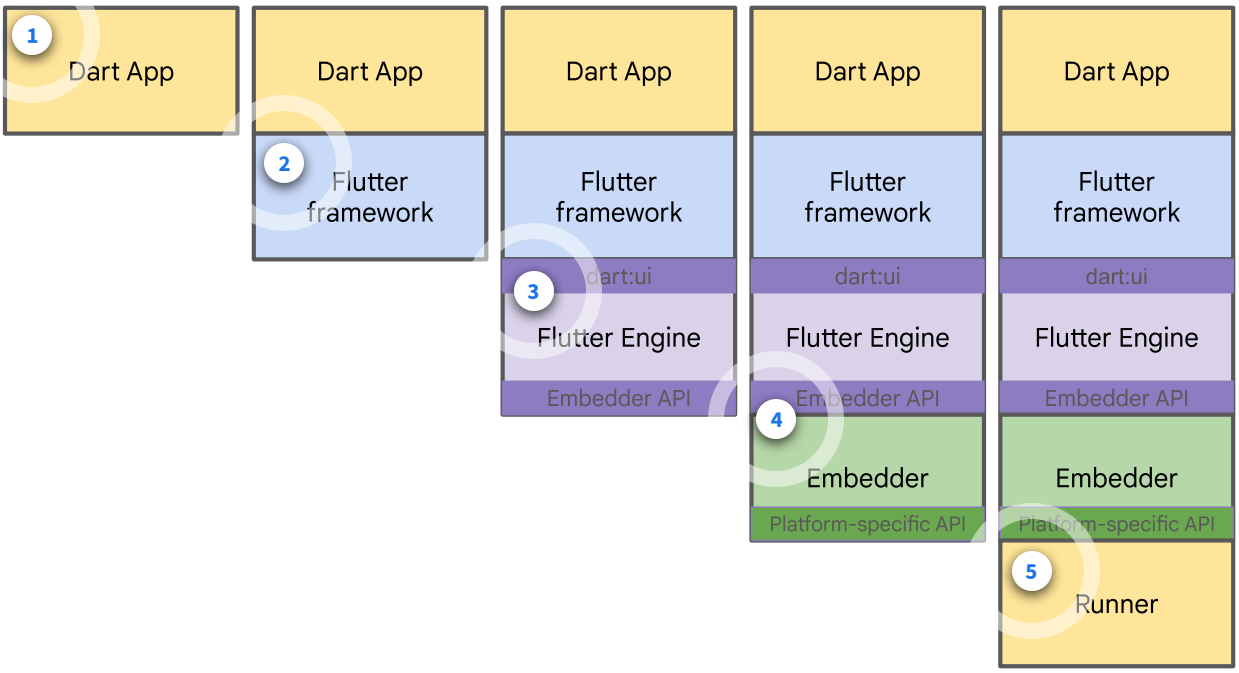
A platform-specific embedder coordinates with the underlying operating system for access to services-like rendering surfaces, accessibility, and input-and manages the message event loop.

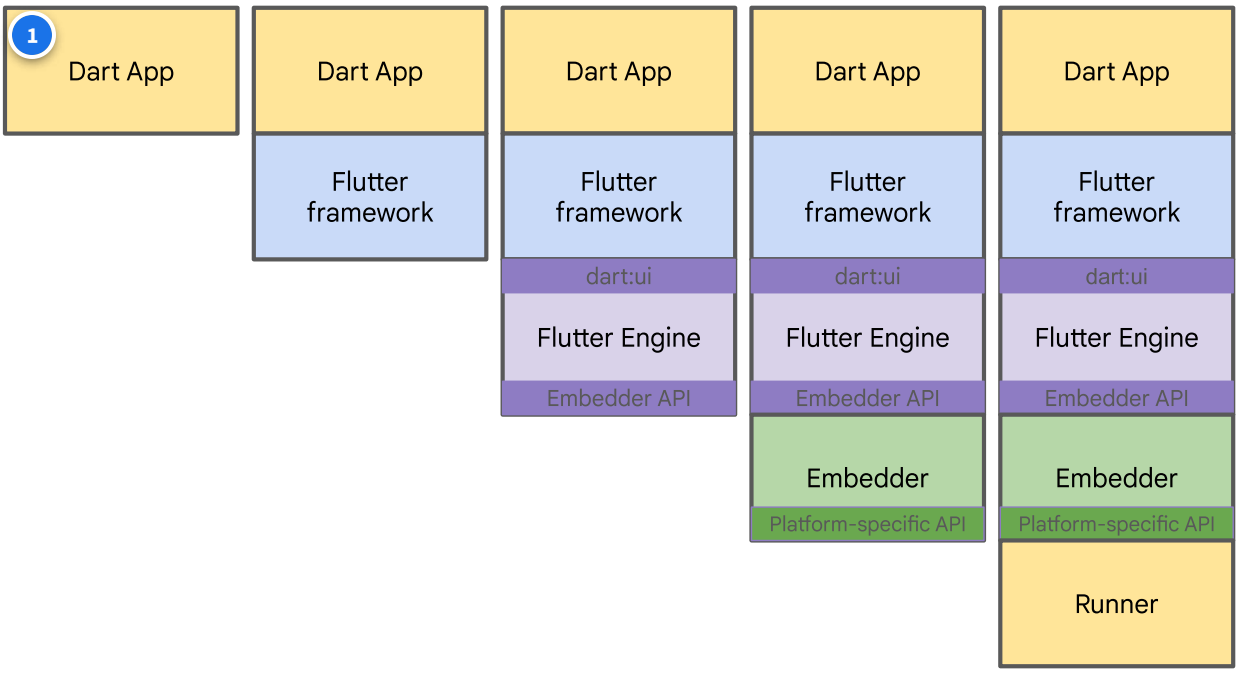
> For a more detailed description of Flutter's layered architecture, read the architectural overview.

Structure of a Flutter app

Here is an overview of the pieces that make up a regular Flutter app. It shows where the Flutter Engine sits in this stack, and highlights API boundaries.

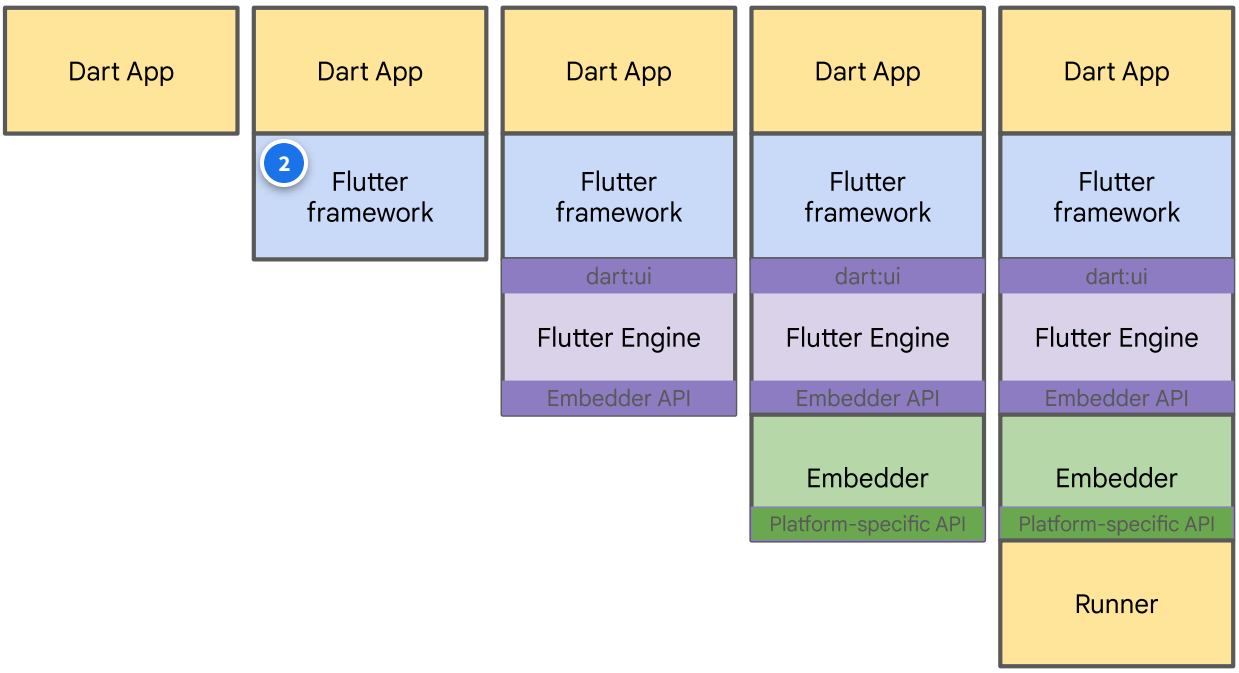
Click each marker to learn more.


1
Dart App
Dart App
2
Flutter framework
Dart App
Flutter framework
dart:ui
3
Flutter Engine
Embedder API
Dart App
Flutter framework
dart:ui
Flutter Engine
Embedder API
4
Embedder
Platform-specific API
Dart App
Flutter framework
dart:ui
Flutter Engine
Embedder API
Embedder
Platform-specific API
5
Runner



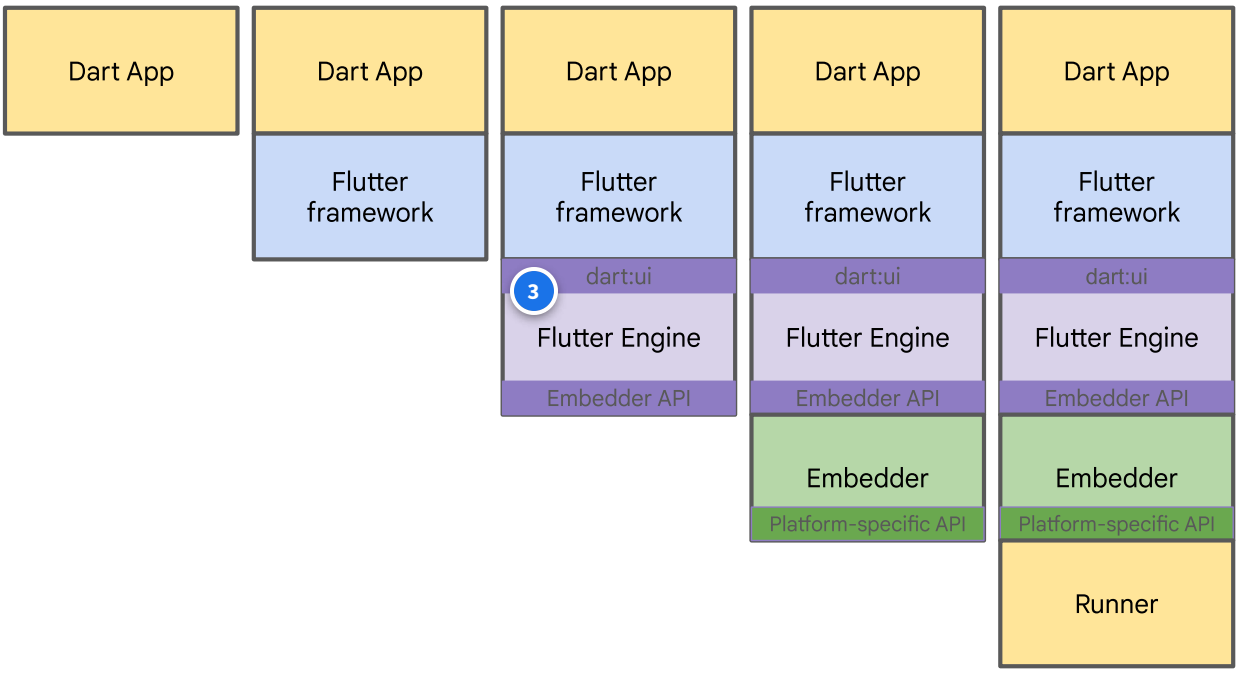
Dart App

- Composes widgets into the desired UI.
- Implements business logic.
- Is owned by the app developer.



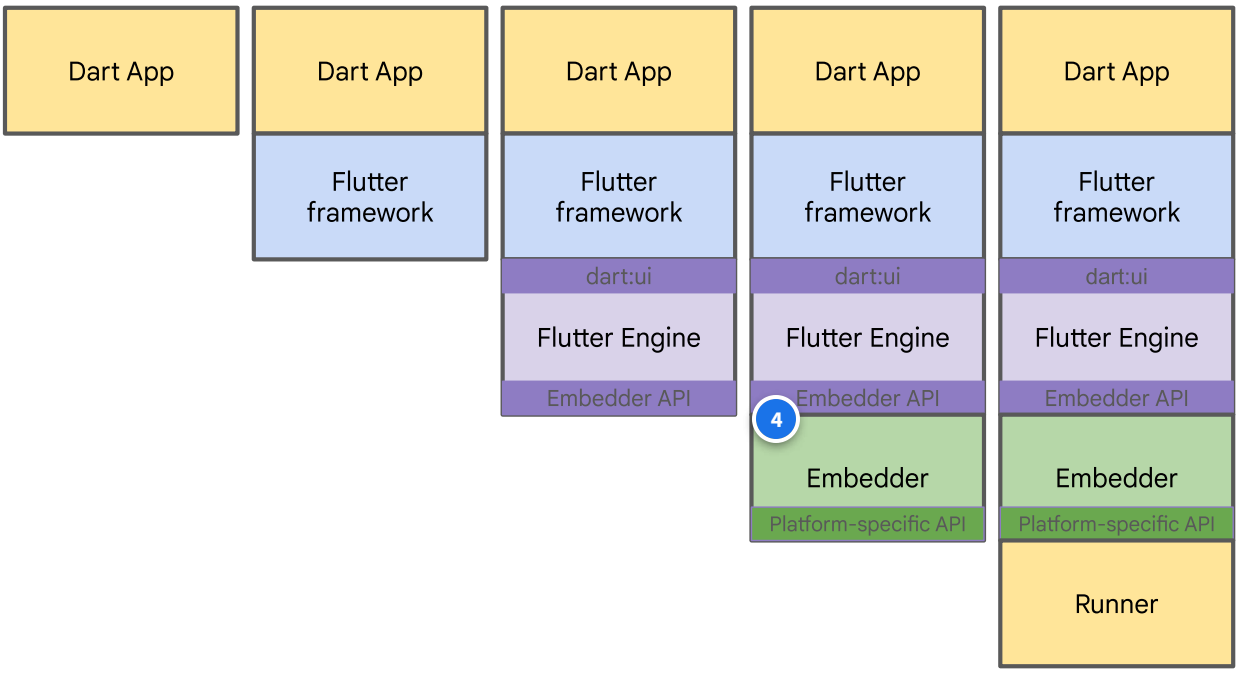
Flutter framework

- Provides a higher-level API to build apps (for example, use widgets, hit-testing, gesture detection, accessibility, and text input).
- Composites the app's widget tree into a scene.



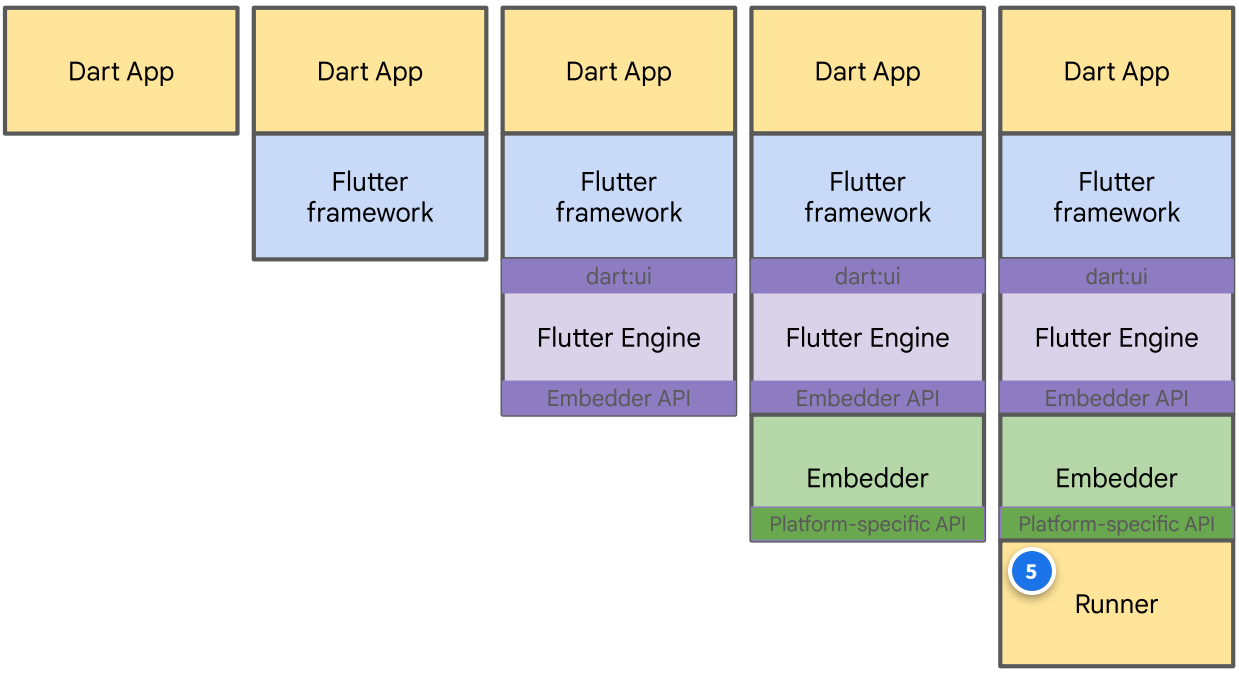
Flutter engine

- Is responsible for rasterizing composited scenes.
- Provides low-level implementation of Flutter's core APIs (for example, graphics, text layout, and Dart runtime).
- Exposes its functionality to the framework using the dart:ui API.
- Integrates with a specific platform using the Engine's Embedder API.



Embedder

- Coordinates with the underlying operating system for access to services like rendering surfaces, accessibility, and input.
- Manages the event loop.
- Exposes platform-specific API to integrate the Embedder into apps.



Runner

- Composes the pieces exposed by the platform-specific API of the Embedder into an app package that is runnable on the target platform.
- Is part of the app template that is generated by *flutter create*.
- Is owned by the app developer.

Flutter concepts

As a Flutter app developer, here are a few concepts of the UI framework to be aware of when developing your apps:

1 Flutter is a reactive, declarative UI framework, where the developer provides a mapping from the application state to the interface state.

2. Flutter explicitly decouples the user interface from its underlying state. You only create the UI description, and the framework uses that configuration to both create and/or update the user interface as appropriate.

3. The framework updates the interface at runtime when the application state changes.

4. In Flutter, you create widgets represented by immutable classes, that are used to configure a tree of objects. Widgets are the building blocks of a Flutter app's user interface.

5. Apps update their user interface in response to events (such as a user interaction) by telling the framework to replace a widget in the hierarchy with another widget. The framework then compares the new and old widgets, and efficiently updates the user interface.

6. During development, Flutter apps run in a VM that offers stateful hot reload of changes without needing a full recompile. For release, Flutter apps are compiled directly to machine code, or to JavaScript for the web.

7. Flutter apps can also be embedded into existing apps, whether they are mobile or web.

For more details on these and other concepts used in Flutter development, please read the documentation.

Developing a Flutter app

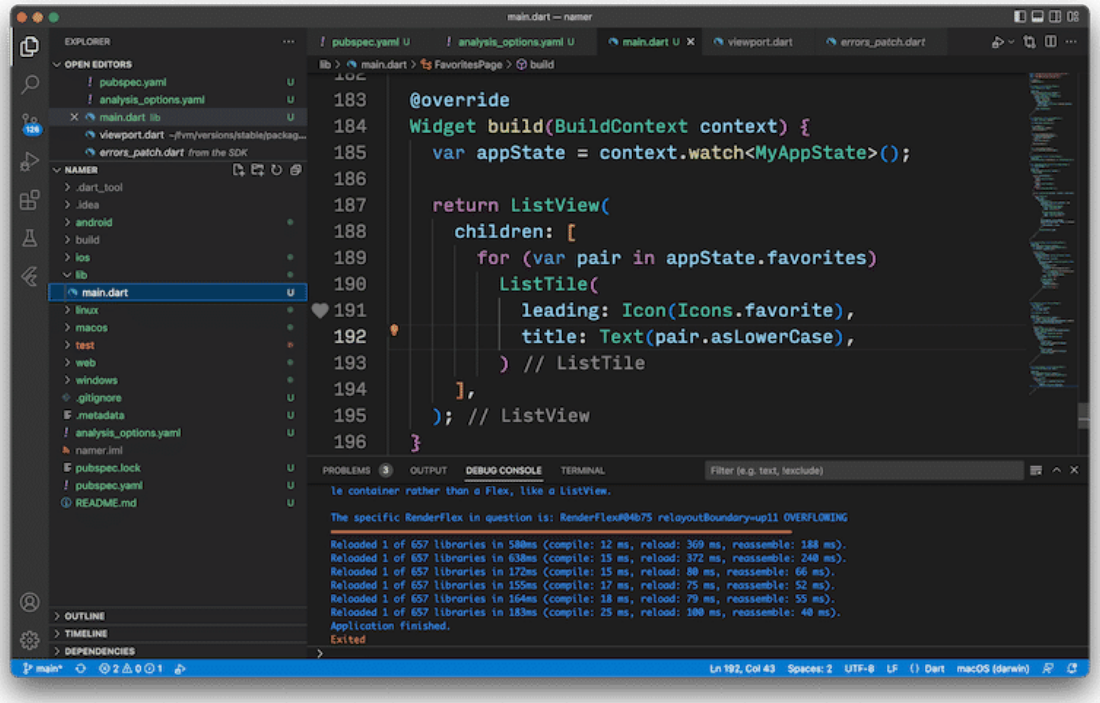
Process overview

Let's review at a high level the steps that you can take to develop a Flutter app.

Click the Start or Next button (>) to learn more.

Step 1

Set up your Flutter environment



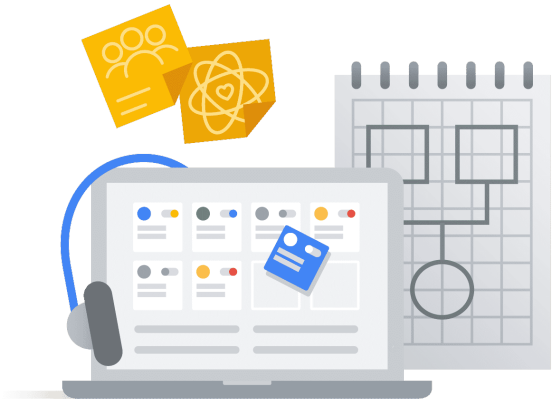
1. Choose an editor to develop your app. You can choose any editor or IDE such as VS Code, IntelliJ, Vim, etc.

2. Choose a development target. Flutter is a multi-platform toolkit that enables your app to run on multiple operating systems: iOS, Android, Windows, macOS, Linux, and web.

   Choose an operating system for which you will primarily develop. This will be your development target-the OS that your app runs on during development.

Step 2

Install Flutter and required tools



Prerequisites:

1. Verify hardware and software requirements for your development machine.
2. Install the required software tools. For example, Xcode for macOS. **Note**: The set of tools will vary depending on your development operating system, and app deployment platform.

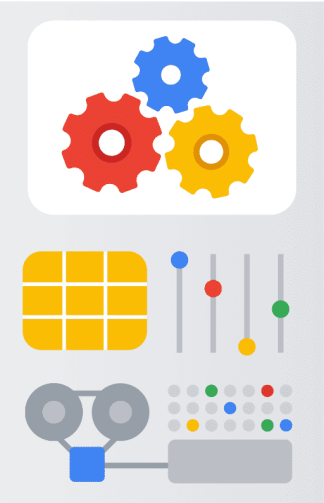
Install the Flutter SDK:

1. Download the Flutter SDK distribution.
2. Create a folder to install the SDK.
3. Unzip the distribution into the installation folder.

4. Add the /<flutter-install-folder-path>/flutter/bin folder to your PATH.

Step 3

Configure your operating system for development



After installing the required tools for your development target operating system, you may need to configure them before starting with app development on Flutter.

For example, on macOS, you have to:

1. Configure Xcode.
2. Configure a physical iOS device or iOS simulator to run your app.

For additional information, view the Flutter installation documentation for your development target platform.

Step 4

Check your development setup

```
Running flutter doctor...
Doctor summary (to see all details, run flutter doctor -v):
[✓] Flutter (Channel stable, 3.24.0, on macOS 14.4.0 23E214 darwin-arm64, locale en)
[!] Android toolchain - develop for Android devices
[!] Chrome - develop for the web
[✓] Xcode - develop for iOS and macOS (Xcode 15)
[!] Android Studio (not installed)
[✓] VS Code (version 1.92)
[✓] Connected device (1 available)
[✓] Network resources


! Doctor found issues in 3 categories.
```

Flutter includes tools that validate the setup for your development target platform.

To verify the installation of all components, run the command:

$ flutter doctor

Note: You might not need all components for development on your operating system platform. For example, if you're not developing for Android, you don't need Android Studio.

If the command output indicates that there are issues with your setup, you can rerun the command with the verbose option:

$ flutter doctor -v

Check the command output for remediation steps. You might need to install additional software or perform other tasks.

Note: If you change the configuration of your Flutter SDK or its related components, run flutter doctor again to verify the installation.

Step 5

Create a Flutter project

You are now ready to start developing your Flutter app.

From your IDE or the command line, you can:

1. Create a new Flutter application project.
2. Add code to the lib/main.dart file which is the entry point of your Flutter app.

We'll review the structure of a Flutter app in a lab in the next module.

Summary

With all the prerequisites and the Flutter SDK installed, you can start developing Flutter apps.

A Flutter app is developed in the Dart programming language.

The Flutter website contains the documentation to install and learn how to develop apps. There are samples and tutorials that teach you how to get started.

The Flutter YouTube channel contains many videos that can help you build and enhance your Flutter development skills.

Module 2 Introduction

In this module

Welcome to the second module of this course.

We'll discuss some of the tools that are part of the Vertex AI platform, and review how you can use a Reasoning Engine agent to build generative AI applications. You'll also complete a lab to integrate an AI Agent with a Flutter App using Vertex AI.

Let's review the module's learning objectives.

Learning objectives

- ✓ Describe the use of Vertex AI Search in Vertex AI Agent Builder.
- ✓ Use a Reasoning Engine agent in Gen AI applications.

Gen AI on Vertex AI Overview

Gen AI on Vertex AI

Generative AI on Vertex AI lets you build production-ready applications that are powered by generative AI models which are hosted on Google's advanced, global infrastructure.

Using Google's advanced, global infrastructure, you can build production-ready Gen AI applications on Vertex AI.

Click each flashcard to learn more.

Enterprise ready

You can deploy Gen AI apps at scale with features that include enterprise-grade security, data residency, access transparency, and low latency.

Extended capabilities

Your Gen AI apps can use a 2M token context window that is supported by the Gemini 1.5 Pro model.

Open platform

You can access many models from third-party companies in your applications.

Core capabilities

Here are some of the core capabilities of generative AI on Vertex AI.

To learn more, click each button to expand the item.

Multimodal processing

Process multiple types of input media at the same time, such as image, video, audio, and text using Gemini, Google's family of generative AI large language models.

Function calling

Extend the model's capabilities by connecting models to external APIs.

Grounding

To reduce hallucinations in model responses, connect models to external data sources.

Model tuning

Adapt models to perform specific tasks with greater precision and accuracy.

Image generation

Generate and edit images using natural language text prompts.

We'll review some of these capabilities later in this module.

Vertex AI Agent Builder

Vertex AI Agent Builder enables developers to use Google's foundation models, search, and conversational AI to create generative AI applications.

Vertex AI Agent Builder is Google's platform for building and managing AI agents and applications using natural language or code.

The platform provides a range of tools for different developer needs and levels of expertise—from a no-code console to build AI agents using natural language to open-source frameworks like LangChain on Vertex AI.

Vertex AI Agent Builder has the following features and capabilities:

- Vertex AI Agents, a natural language understanding platform that is built on large language models (LLMs).

- Vertex AI Search, a fully-managed AI-enabled search platform and grounding system.

AI Agents

Gen AI models have great content generation and analysis capabilities. To generate useful, personalized responses, model capabilities can be combined or grounded with data from external systems, such as databases or document servers.

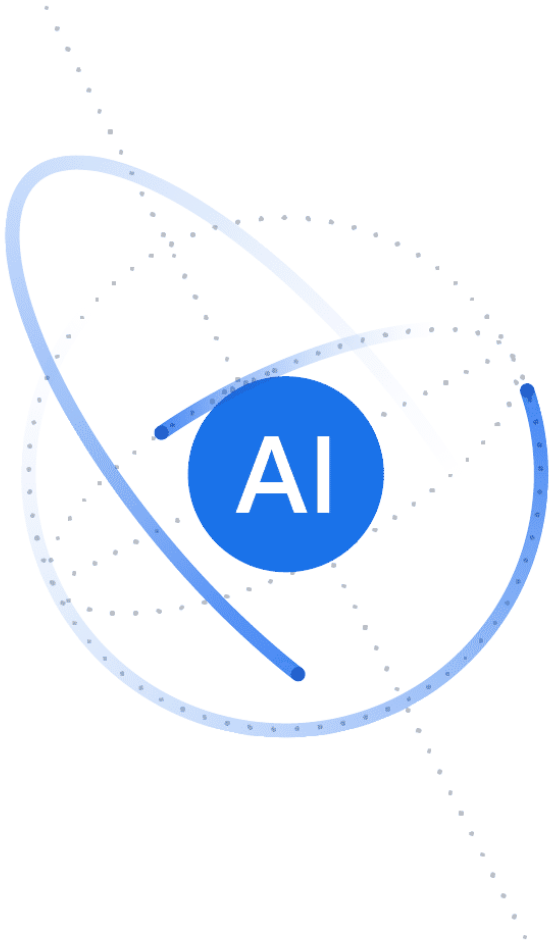
This is where AI agents come in.

An Agent

A basic building block of agent apps. An agent app typically has many agents, where each agent is defined to handle specific tasks.

Agent data is provided to the LLM to answer questions and execute tasks.

Each agent can provide information, send queries to external services, or defer conversation handling to another agent to handle sub-tasks.



AI Agent application

An AI Agent application:

- Uses the power of LLMs for reasoning and orchestration with external tools to achieve a goal.
- Grounds a model with external data sources to get the latest information and lower the risk of hallucination.

- Dynamically and semantically integrates a loosely coupled distributed system with the power of LLMs.

Capabilities of Vertex AI Agent Builder

Vertex AI Agent Builder is a suite of tools for building AI Agents. It helps to streamline agent processes with orchestration and grounding.

Click each card to learn more.

Improve the accuracy and relevance of model output with:

- Grounding with Google Search.
- Grounding on your own data with Vertex AI Search.

Connect LLMs to external tools, call APIs and services with:

- Function calling
- Extensions

Create and launch AI agents and scale with orchestration tools:

- Vertex AI Reasoning Engine (LangChain on Vertex AI)

Build conversational agents with Vertex AI Agents.

Here is more information on the capabilities available in Vertex AI Agent Builder:

To learn more, click each button to expand the item.

Design conversational interfaces

With Vertex AI Agents, you can easily design and integrate a conversational user interface into your mobile app, web application, device, bot, interactive voice response system, and other applications.

Analyze multiple types of input

Vertex AI Agents can analyze multiple types of input including text or audio inputs. Agents can also respond with text or synthetic speech.

Ground in Google search and/or your enterprise data with RAG

- To ground results from Google Search, use the Gemini API.
- To ground models in your enterprise data, use the RAG system from Vertex AI Search.
- To build your own RAG system use APIs for document processing, ranking, grounded generation, and performing checks on outputs.

Create low-code to high-code AI applications quickly

You can develop Gen AI-powered applications with a combination of low-code APIs and code-first orchestration.

- LangChain on Vertex AI lets you leverage the LangChain open source library to build custom Gen AI applications and use Vertex AI for models, tools, and deployment.
- The Firebase Genkit Vertex AI plugin provides interfaces to several Google Gen AI models through the Vertex AI API.

Maintain security and compliance

Vertex AI Agent Builder offers built-in security, compliance, and governance features, aligning with many industry certifications.

You can maintain data privacy and control over your AI apps, manage access, and ensure the responsible use of AI models and data.

For a full list of features and more details, please read the documentation.

Grounding

Vertex AI Agent Builder streamlines the process of grounding generative AI outputs in enterprise data with:

- Vertex AI Search
- RAG (retrieval augmented generation) APIs

Click each card to learn more.

Vertex AI Search

With Vertex AI Search, you can build a Google-quality search app on data you control.

You can also use the search results that you retrieve to ground Gen AI LLM responses.

RAG (retrieval augmented generation) APIs.

RAG is an architecture pattern that combines LLMs with backend information retrieval from other information sources.

The RAG pattern can help overcome some of the limitations of LLMs, such as:

We'll discuss Vertex AI Search in the next lesson.

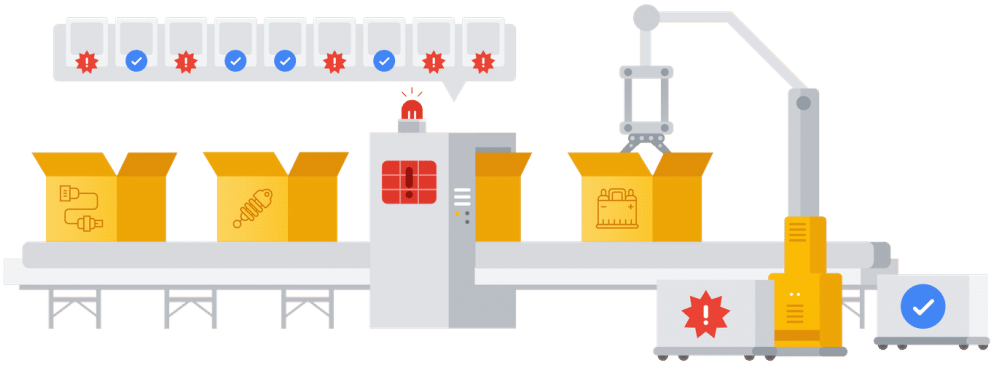
Vertex AI Search

Vertex AI Search is a fully-managed platform that is powered by large language models, that lets you build AI-enabled search and recommendation experiences for your public or private websites or mobile applications.

Grounding with Vertex AI Search

Earlier in this module, we mentioned Grounding as a way to reduce hallucinations in model responses. Let's discuss this generative AI capability in the context of Vertex AI Search.

If you provide models with access to specific data sources, then grounding tethers the model output to these data sources, and reduces the chances of inventing content.



Generative AI model

Grounding:

- Reduces model hallucinations (instances when the model generates a response that is not factual).
- Anchors model responses to specific information.
- Enhances reliability and applicability of the generated content.

In Vertex AI, you can ground supported model output in two main ways:

- With Google Search
- With your own data

Click each flashcard to learn more.

Google Search

Using Google Search, you can ground a model with publicly available web data. You can connect the model with a range of topics on world knowledge and information on the internet.

Your own data

You can ground models to your own text data using Vertex AI Search as a datastore. With Vertex AI Search, you integrate your own data to refine the model output.

Prerequisites for grounding to your data

There are a few prerequisites needed before you can ground model output to your data:

1. Enable Vertex AI Agent Builder and activate the API.
2. Create a Vertex AI Agent Builder search data store and app.
3. Link your data store to your app in Vertex AI Agent Builder. The data store serves as the foundation for grounding the model in Vertex AI.

You'll perform these steps in the lab that is part of this module.

Key features of Vertex AI Search

Here are some key features of Vertex AI Search.

Click each button to expand the items and learn more.

Natural language understanding and semantic search

Vertex AI Search provides a high-quality search experience without needing to implement and maintain systems that perform complex NLP techniques, keyword searches, or pattern matching. Capabilities to understand synonyms, correct spellings, and auto-suggest searches are included.

Generative AI

You can incorporate Gen AI-powered summarization and conversational search for unstructured documents.

Search recommendations

With ML-based content and metadata understanding, users can quickly find content that is similar to what they are currently viewing.

Vertex AI Search console and APIs

To set up a search app for your public websites, or for your structured or unstructured data, you can use the Agent Builder in the Google Cloud console or Google's APIs.

To ground text responses to a Vertex AI Agent Builder data store, you can use the Vertex AI API.

Types of Search apps

With Vertex AI Search, you can quickly build a Google-quality search app on your own data and embed a search bar in your website pages or app.

You can create the following different types of search apps:

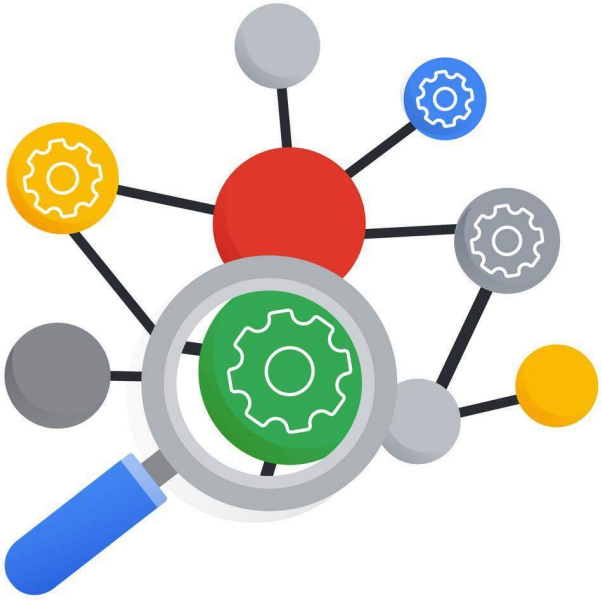
Click each tab to learn more.

GENERIC SEARCH | MEDIA SEARCH | HEALTHCARE SEARCH

Apply generic search to websites or data stores containing your proprietary data. This enables your users to search for content that you want them to see.

GENERIC SEARCH | MEDIA SEARCH | HEALTHCARE SEARCH

This type of search capability is specially designed for media content, such as movies, videos, and music. With media search, users can efficiently find the media content that they want to view or listen to.



GENERIC SEARCH | MEDIA SEARCH | HEALTHCARE SEARCH

A search capability that lets users query healthcare records that are stored in FHIR data stores. You can import FHIR resources that contain clinical data from your Cloud Healthcare API FHIR store. You can also search unstructured data, such as images, PDF files, and RTF files, referenced by the FHIR resources.

Lesson 10 of 11

Use a Reasoning Engine Agent in Gen AI Applications

Reasoning Engine (LangChain on Vertex AI) is a managed service from Google Cloud that helps you build and deploy an agent reasoning framework.

Reasoning Engine is based on LangChain, an open source framework for building chatbots and RAG systems.

Benefits of Reasoning Engine

Some of the benefits of using Reasoning Engine are listed below.

To learn more, click each button to expand the items.

Integration with Vertex AI ecosystems

Reasoning Engine lets developers integrate their Gen AI applications with Gemini models, and transparently access Function Calling, and Extensions

capabilities of Vertex AI.

Security and scalability

The Reasoning Engine managed runtime handles tasks, such as container creation, configuration of authentication, IAM, and scaling.

Vertex AI handles autoscaling, regional expansion, and container vulnerabilities.

Simplified deployment

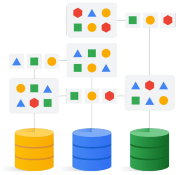
Reasoning Engine uses the same APIs as LangChain to interact with LLMs and build applications. With single click deployment, Reasoning Engine simplifies and speeds up deployment with Vertex AI LLMs.

Use cases

By utilizing LangChain's standardized interfaces, you can build different kinds of applications with Reasoning Engine. You can customize the logic of your application and incorporate any framework with a high degree of flexibility.

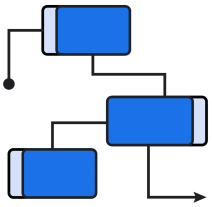
Here are some use cases of Gen AI applications using Reasoning Engine:

Connect to public APIs



Connect to databases

Use open source frameworks



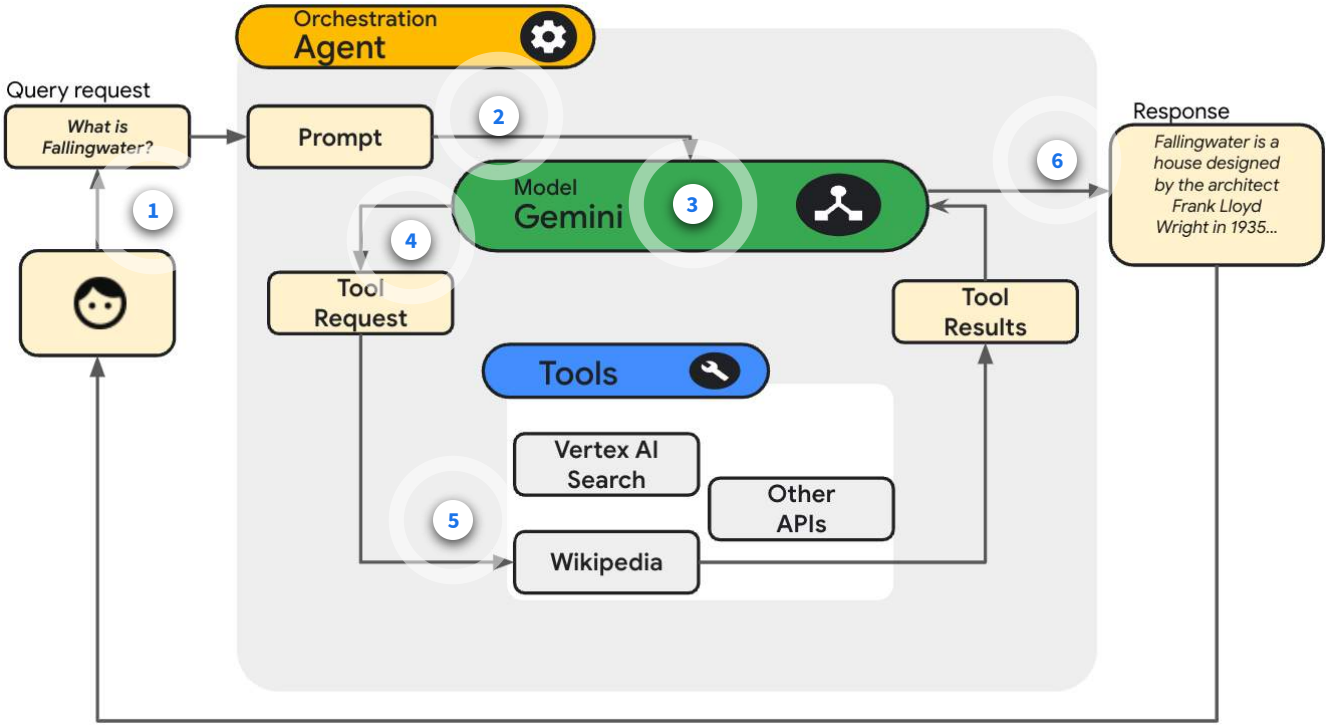
Build debug and trace agents

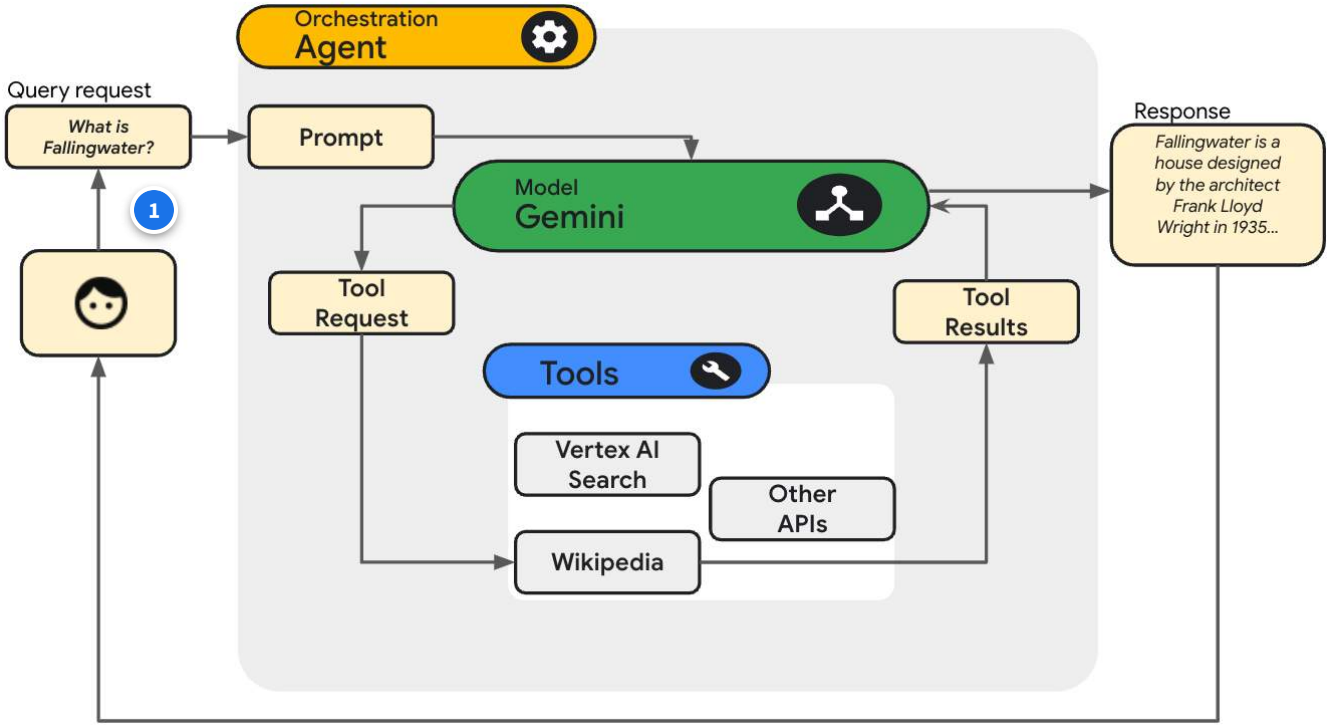
More details and links to sample implementations are available on the documentation site.

System flow

Here is a high level view of the system flow and components that are part of a Gen AI application that uses Reasoning Engine and Vertex AI.

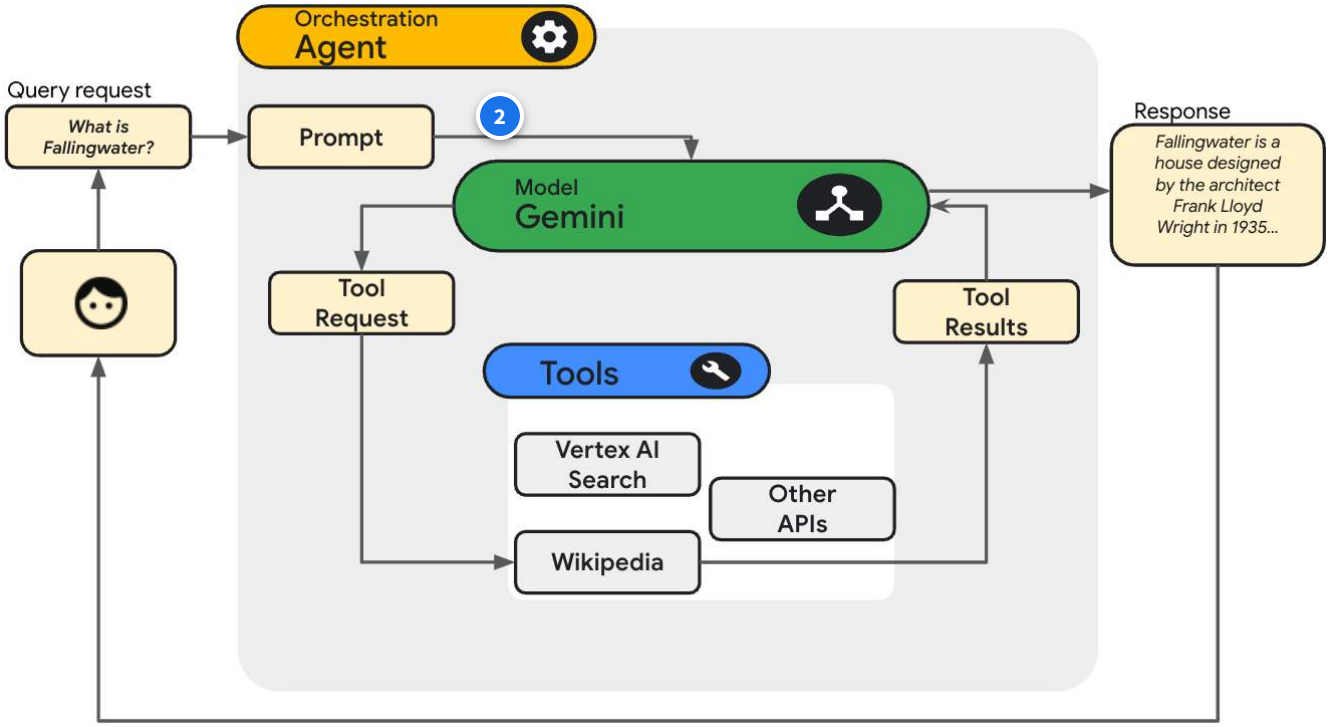
Click each marker to learn more.


Orchestration
Agent
Query request
What is Fallingwater?
Prompt
2
Model
Gemini
3
6
Response
Fallingwater is a house designed by the architect Frank Lloyd Wright in 1935...
1
4
Tool Request
Tool Results
Tools
Vertex AI Search
Other APIs
5
Wikipedia



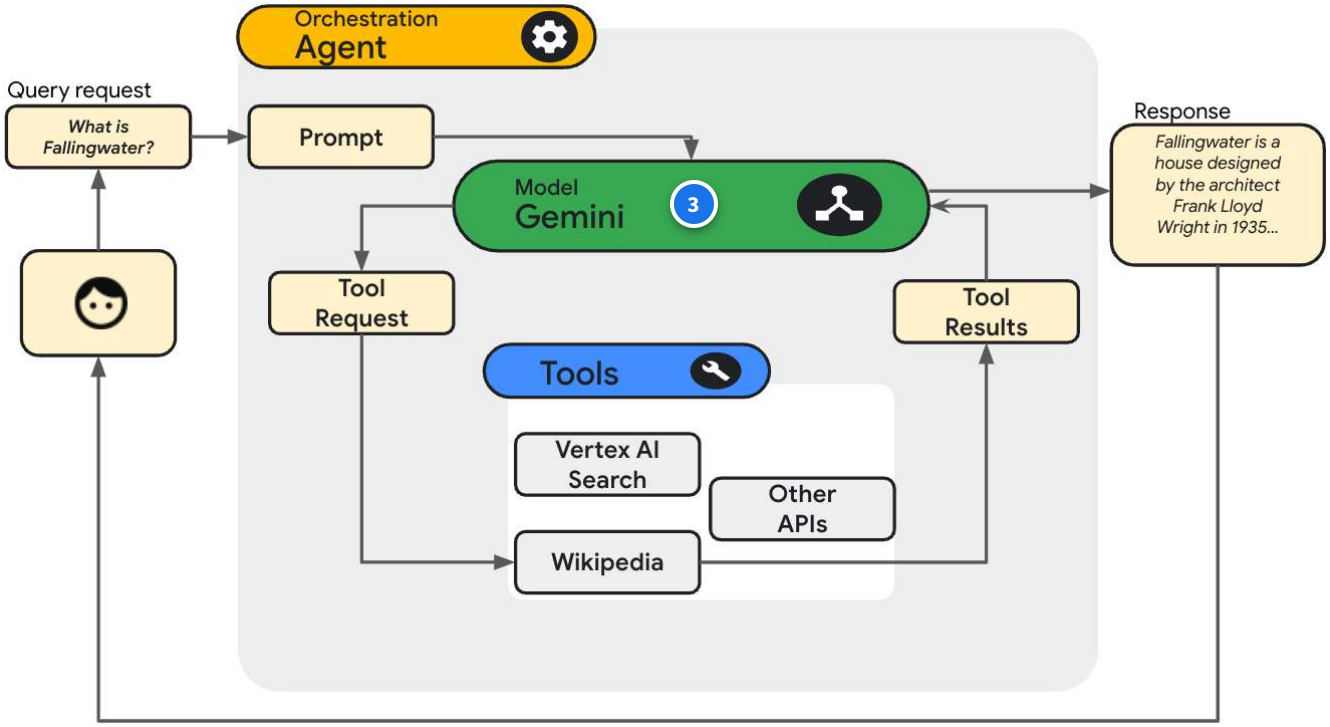
User request

User makes a query request.



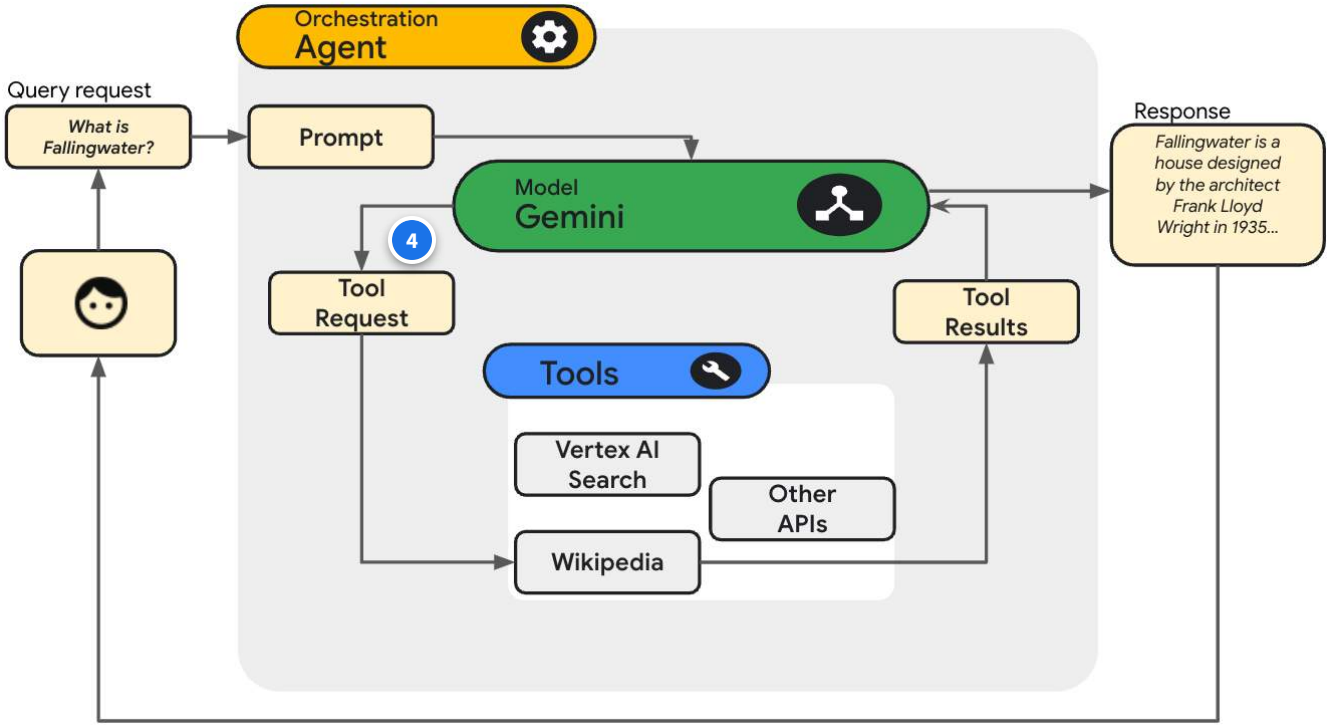
Prompt

The agent constructs a prompt from the user's request, and supplies it to the model.



Model

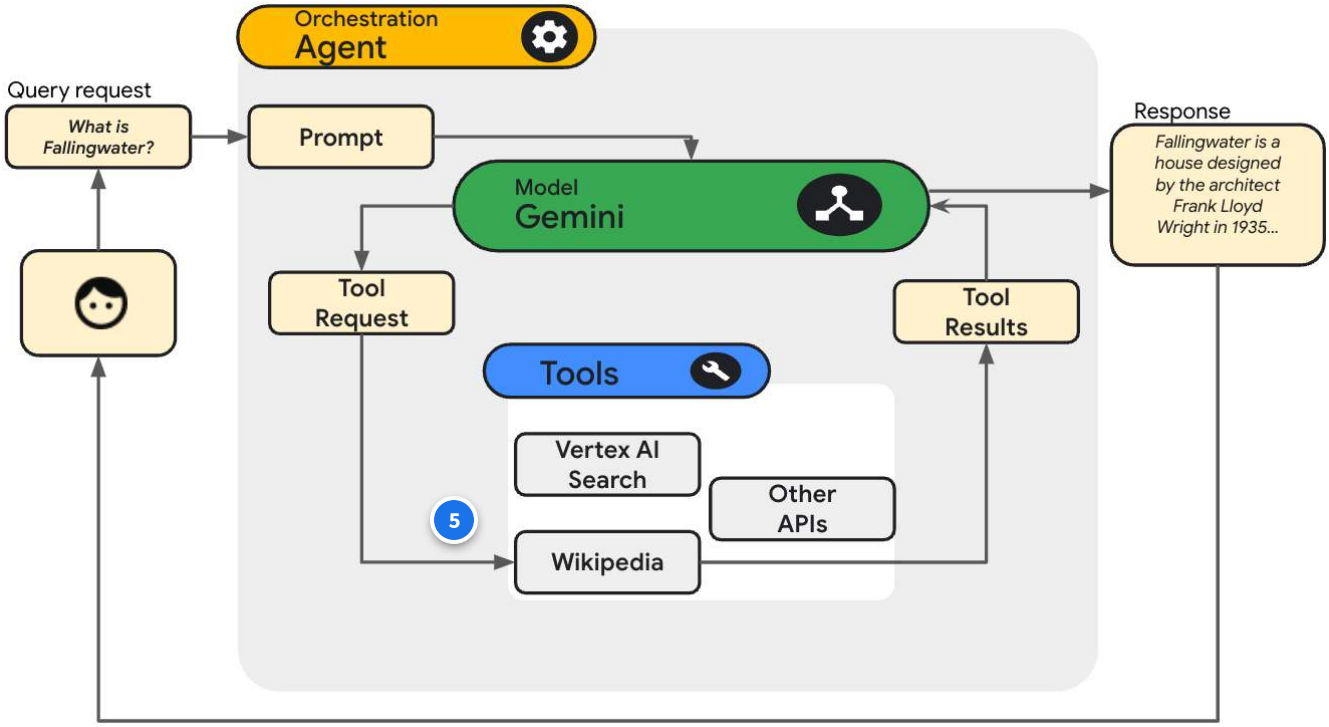
The LLM processes the prompt and determines whether it wants to use any of the tools that are configured by the agent.



Tool request

If the LLM chooses to use a tool, it uses Function Calling with the tool name and parameters that allows the agent to invoke the tool, and provide the results from the tool back to the LLM.

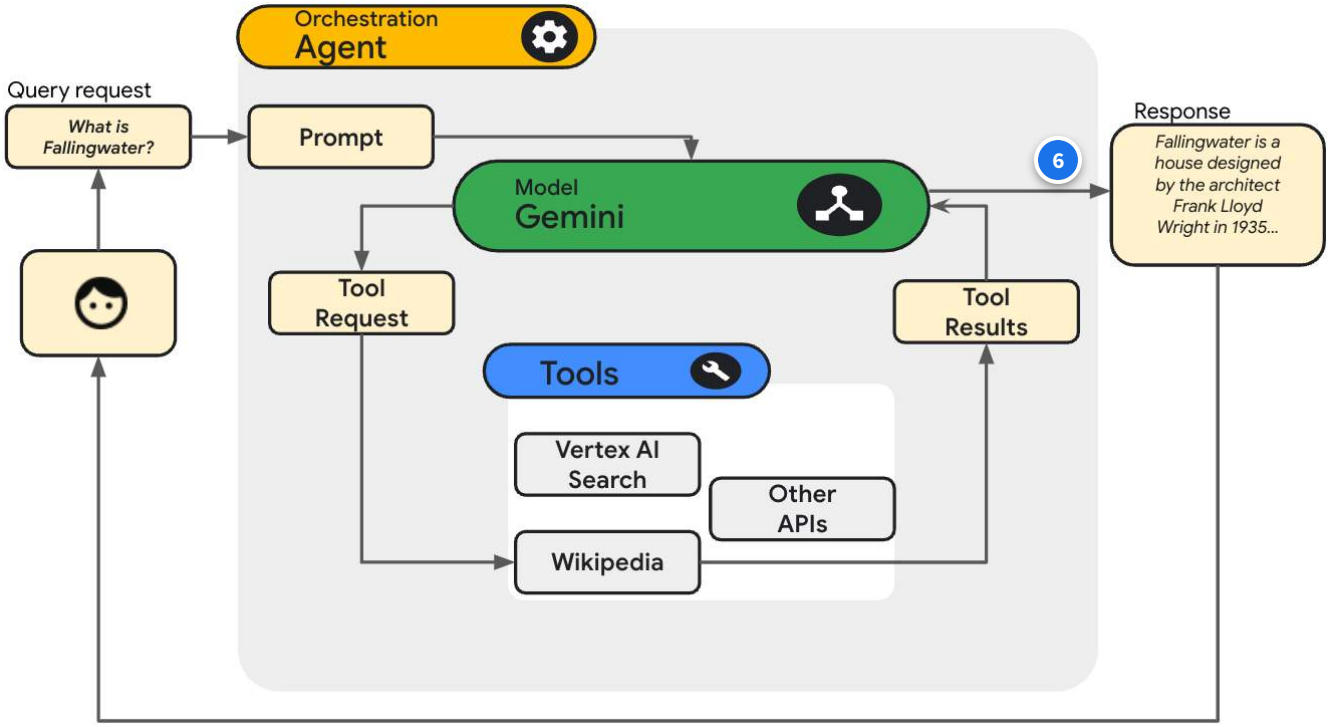
If the LLM chooses not to use any of the tools, it generates content that is relayed by the agent back to the user.



Tool invocation - Wikipedia

You can define Python functions that get used as tools with Gemini Function Calling.

In this example, a Python function tool is called, which invokes the Wikipedia API to fetch results based on the request prompt.



Tools results and response

The results from the tool are returned to the LLM and a response is generated.

Components

The main components that are used with Reasoning Engine are described below.

Click each tab to learn more.

MODEL	TOOL	ORCHESTRATION FRAMEWORK	MANAGED RUNTIME

The model processes a prompt and provides a response.
While processing a prompt, the model delegates certain tasks to the tools that you define.

MODEL	TOOL	ORCHESTRATION FRAMEWORK	MANAGED RUNTIME

You can choose to define a set of tools that communicate with external APIs (for example, a database) and provide them to the model.

Vertex AI's managed runtime is optimized to use tools based on Gemini Function Calling.

MODEL	TOOL	ORCHESTRATION FRAMEWORK	MANAGED RUNTIME

Reasoning Engine lets you leverage the LangChain orchestration framework in Vertex AI. You can use LangChain to decide how deterministic your application should be.

MODEL	TOOL	ORCHESTRATION FRAMEWORK	MANAGED RUNTIME

The Reasoning Engine managed runtime is a Vertex AI service that comes with the security, privacy, observability, and scalability features of Vertex AI. You can quickly deploy and scale your applications for production environments.

Developing a Reasoning Engine agent

Process overview

Let's review at a high level the steps that you can take to develop a Reasoning Engine agent.

Click the Start or Next button (>) to learn more.

Step 1

Set up your environment

1. Select or create a Google Cloud project with billing enabled.
2. Enable the Vertex AI and Cloud Storage APIs.
3. Ensure that you have the necessary role permissions (Vertex AI User, Storage Admin) to use Reasoning Engine.
4. Create a Cloud Storage bucket to let Reasoning Engine stage your application artifacts.

For additional details and optional steps, view the documentation.

Step 2

Install the Vertex AI SDK for Python

To install the Vertex AI SDK for Python, run:

```
pip install google-cloud-aiplatform[langchain,reasoningengine]
```

To initialize the SDK in your Python application, use the following code:

```
import vertexai

vertexai.init( project="PROJECT_ID", location="LOCATION",
staging_bucket="gs://BUCKET_NAME")
```

Use your Google Cloud project ID, region, and Cloud Storage bucket in the init() call.

Step 3

Define and configure a model

To develop your agent application, you must define and optionally configure a Gen AI model.

In your application code, define a model and its version in a variable:

```
model = "gemini-1.5-pro-001"
```

Optionally, configure the model with safety settings and other parameters.

Step 4

Define tools (Python functions)

```
def query_with_wikipedia(
    query: str = "Fallingwater",
):
    """
    Finds answer for any topic or object from Wikipedia and returns a dictionary containing the answer.

    Args:
        query: the name of object or topic to find.

    Example: {"answer": "Fallingwater is a house designed by the architect Frank Lloyd Wright in 1935."}
    """

    wiki_title = search_wiki_title(query)  # calls Wikipedia API to get wiki page title
    wiki_full_text = get_wiki_full_text(
        wiki_title
    )  # calls Wikipedia API to get full text

    return {"answer": wiki_full_text}
```

In addition to the model, you can define Python functions as tools that enable the model to call external APIs, databases, and other systems.

This allows the model to access the latest information from the external data source and lowers the risk of hallucination.

This sample function uses the Wikipedia API to find a relevant Wikipedia page for a specified object, and returns the full text of the page.

By writing the function with a proper function signature and a comment explaining its functionality and usage, it will automatically be inspected by Function Calling. This Vertex AI

service dynamically determines when the function should be invoked, and also how to map the request to the function parameters.

Step 5

Define the agent


```
model_name = "gemini-1.5-pro-001"
agent = reasoning_engines.LangchainAgent(
   model=model_name,
   tools=[query_with_wikipedia, find_product_from_googleshop],
   agent_executor_kwargs={"return_intermediate_steps": True},
)
```

With the model and tools defined, you can now create an agent.

Use the LangChain agent template that is provided in the Vertex AI SDK for Reasoning Engine.

Note: In the code shown, a second tool *find_product_from_googleshop* is provided to the Reasoning Engine agent.

Step 6

Test the agent locally

```
input_text = (
   "What is Chrome Dino pin?"
)
agent.query(input=input_text)
```

```
{'input': 'What is Chrome Dino pin?',
 'output': "The Chrome Dino game doesn't have a pin. It's a built-in browser game that you can play when
you're offline in the Google Chrome web browser. \n",
 'intermediate_steps': [[{'id': ['langchain',
     'schema',
     'agent',
     'ToolAgentAction'],
    'kwargs': {'tool': 'query_with_wikipedia',
     'type': 'AgentActionMessageLog',
     'tool_input': {'query': 'Chrome Dino pin'},
     'tool_call_id': '0bef67da-b458-4c46-bd89-28c2d06f36b5',
     'message_log': [{'lc': 1.0,
       'kwargs': {'invalid_tool_calls': [],
        'type': 'AIMessageChunk',
...
```

To ensure that the model and tools are working as expected, you can test the agent locally.

The partial output from the agent shows the intermediate steps that include the *query_with_wikipedia()* tool (Python function) that was called to generate the response.

Step 7

Deploy the agent

```
remote_agent = reasoning_engines.ReasoningEngine.create(
    agent,
    requirements=[
        "google-cloud-aiplatform[langchain,reasoningengine]",
        "requests",
    ],
)
```

You can now deploy the agent to the Reasoning Engine runtime in Vertex AI.

The deployment process creates a container image that encapsulates the agent and tools. The runtime provides a scalable, fully managed platform for agents, without the effort required to build and operate your own server infrastructure.

To deploy the agent to the runtime, use the reasoning_engines.ReasoningEngine.create() method and provide:

1. The reference to the agent instance.
2. The required Python packages and versions needed by the agent at runtime. (similar to how these are specified in a requirements.txt file)

Summary

```
response = remote_app.query(input="What is
Fallingwater by Frank Lloyd Wright?")
```

Once the deployment is complete, you can use the agent from your application or a REST API.

What Did I Walk Away With?

In this course, you learned about:

- ✓ Generative AI and the Gemini family of large language models.
- ✓ Vertex AI and its features.
- ✓ Flutter, and how it can be used to develop apps.

You also learned about:

- ✓ Vertex AI Agent Builder tools to build AI agents.
- ✓ Vertex AI Search and how it can be used to ground models.
- ✓ Building Reasoning Engine agents on Vertex AI.

This completes the module content for the course. Please return to the platform to continue with the next item in the course.