

Starting April 29, 2025, Gemini 1.5 Pro and Gemini 1.5 Flash models are not available in projects that have no prior usage of these models, including new projects. For details, see [Model versions and lifecycle](#) (/vertex-ai/generative-ai/docs/learn/model-versions#legacy-stable).

# Agent Engine API

Agent Engine (formerly known as Reasoning Engine or LangChain on Vertex AI) provides a managed runtime for your agents. You can create an agent using orchestration frameworks such as LangChain, and deploy it with Agent Engine. This service has all the security, privacy, observability, and scalability benefits of Vertex AI integration.

For more conceptual information about Agent Engine, see [Agent Engine](#) (/vertex-ai/generative-ai/docs/agent-engine/overview).

## Limitations

- Agent Engine supports only Python orchestration frameworks.
- Agent Engine is supported only in the `us-central1` region.

**Note:** Because the name of Vertex AI Agent Engine changed over time, the name of the resource is [ReasoningEngine](#) (/vertex-ai/generative-ai/docs/reference/rest/v1/projects.locations.reasoningEngines) to maintain backwards compatibility.

## Example syntax

Syntax to create and register a reasoning engine resource.

Python  
(#python)

```
class SimpleAdditionApp:
    def query() -> str:
        """
        ...
```

```
    """

    return

...

reasoning_engine = reasoning_engines.ReasoningEngine.create(
    SimpleAdditionApp(),
    display_name="",
    description="",
    requirements=[...],
    extra_packages=[...],
)
```

## Parameter list

### Parameters

display_name	Required: <b>string</b>  The display name of the <b>ReasoningEngine</b> .
description	Optional: <b>string</b>  The description of the <b>ReasoningEngine</b> .
spec	Required: <b>ReasoningEngineSpec</b>  Configurations of the <b>ReasoningEngine</b> .
package_spec	Required: <b>PackageSpec</b>  A user provided package specification, such as pickled objects and package requirements.
class_methods	Optional: <b>protobuf.Struct</b>  Declarations for object class methods.

### PackageSpec

PackageSpec contains the reference to the Cloud Storage URI storing the OpenAPI YAML file.

### Parameters

<code>pickle_object_gcs_uri</code>	Optional: <code>string</code>  The Cloud Storage URI of the pickled python object.
<code>dependency_files_gcs_uri</code>	Optional: <code>string</code>  The Cloud Storage URI of the dependency files with the <code>tar.gz</code> extension.
<code>requirements_gcs_uri</code>	Optional: <code>string</code>  The Cloud Storage URI of the <code>requirements.txt</code> file.
<code>python_version</code>	Optional: <code>string</code>  The Python version. Supported versions include Python <code>3.8</code> , <code>3.9</code> , <code>3.10</code> , and <code>3.11</code> . If not specified, the default value is <code>3.10</code> .

QueryReasoningEngine

Parameters

<code>input</code>	<code>protobuf.struct</code>  The arguments inside <code>input</code> should be consistent with the <code>query</code> class method defined in the creation step.
--------------------	---

Examples

Deploy a basic app configuration

The following example uses an application that adds two integers and a remote app with Reasoning Engine:

[Vertex AI SDK for Python](#)  
(#vertex-ai-sdk-for-python)

To learn how to install or update the Vertex AI SDK for Python, see [Install the Vertex AI SDK for Python](#) (/vertex-ai/docs/start/use-vertex-ai-python-sdk). For more information, see the [Vertex AI SDK for Python API reference documentation](#) (/python/docs/reference/aipatform/latest).

```

import vertexai
from vertexai.preview import reasoning_engines

# TODO(developer): Update and un-comment below lines
# PROJECT_ID = "your-project-id"
# staging_bucket = "gs://YOUR_BUCKET_NAME"
vertexai.init(
    project=PROJECT_ID, location="us-central1", staging_bucket=staging_bucket
)

class SimpleAdditionApp:
    def query(self, a: int, b: int) -> str:
        """Query the application.
        Args:
            a: The first input number
            b: The second input number
        Returns:
            int: The additional result.
        """
        return f"{int(a)} + {int(b)} is {int(a + b)}"

# Locally test
app = SimpleAdditionApp()
app.query(a=1, b=2)

# Create a remote app with Reasoning Engine.
# This may take 1-2 minutes to finish.
reasoning_engine = reasoning_engines.ReasoningEngine.create(
    SimpleAdditionApp(),
    display_name="Demo Addition App",
    description="A simple demo addition app",
    requirements=["cloudpickle==3"],
    extra_packages=[],
)

# Example response:
# Using bucket YOUR_BUCKET_NAME
# Writing to gs://YOUR_BUCKET_NAME/reasoning_engine/reasoning_engine.pkl
# ...
# ReasoningEngine created. Resource name: projects/123456789/locations/us-centr
# To use this ReasoningEngine in another session:
# reasoning_engine = vertexai.preview.reasoning_engines.ReasoningEngine('projec

```

## Deploy an advanced app configuration

This is an advanced example that uses LangChain's chain, prompt templates, and the Gemini API:

#### Vertex AI SDK for Python

(#vertex-ai-sdk-for-python)

To learn how to install or update the Vertex AI SDK for Python, see [Install the Vertex AI SDK for Python](#) (/vertex-ai/docs/start/use-vertex-ai-python-sdk). For more information, see the [Vertex AI SDK for Python API reference documentation](#) (/python/docs/reference/aipatform/latest).

```
from typing import List

import vertexai
from vertexai.preview import reasoning_engines

# TODO(developer): Update and un-comment below lines
# PROJECT_ID = "your-project-id"
# staging_bucket = "gs://YOUR_BUCKET_NAME"

vertexai.init(
    project=PROJECT_ID, location="us-central1", staging_bucket=staging_bucket
)

class LangchainApp:
    def __init__(self, project: str, location: str) -> None:
        self.project_id = project
        self.location = location

    def set_up(self) -> None:
        from langchain_core.prompts import ChatPromptTemplate
        from langchain_google_vertexai import ChatVertexAI

        system = (
            "You are a helpful assistant that answers questions "
            "about Google Cloud."
        )
        human = "{text}"
        prompt = ChatPromptTemplate.from_messages(
            [("system", system), ("human", human)]
        )
        chat = ChatVertexAI(project=self.project_id, location=self.location)
        self.chain = prompt | chat

    def query(self, question: str) -> Union[str, List[Union[str, Dict]]]:
        """Query the application.
```

```

    Args:
        question: The user prompt.
    Returns:
        str: The LLM response.
    """
    return self.chain.invoke({"text": question}).content

# Locally test
app = LangchainApp(project=PROJECT_ID, location="us-central1")
app.set_up()
print(app.query("What is Vertex AI?"))

# Create a remote app with Reasoning Engine
# Deployment of the app should take a few minutes to complete.
reasoning_engine = reasoning_engines.ReasoningEngine.create(
    LangchainApp(project=PROJECT_ID, location="us-central1"),
    requirements=[
        "google-cloud-aiplatform[langchain,reasoningengine]",
        "cloudpickle==3.0.0",
        "pydantic==2.7.4",
    ],
    display_name="Demo LangChain App",
    description="This is a simple LangChain app.",
    # sys_version="3.10", # Optional
    extra_packages=[],
)
# Example response:
# Model_name will become a required arg for VertexAIEmbeddings starting...
# ...
# Create ReasoningEngine backing LRO: projects/123456789/locations/us-central1/
# ReasoningEngine created. Resource name: projects/123456789/locations/us-centr
# ...

```





## Query Reasoning Engine

Query a reasoning engine.



This example uses the `SimpleAdditionApp` class from the [Deploy a basic app configuration example](#) (`#deploy_a_basic_app_configuration`).

**REST** Vertex AI SDK for Python...  
(`#rest`)


Before using any of the request data, make the following replacements:

- **PROJECT\_ID** : Your project ID  
(/resource-manager/docs/creating-managing-projects#identifiers).
- **LOCATION** : The region to process the request. Must be `us-central1`.
- **REASONING\_ENGINE\_ID** : The ID of the reasoning engine.
- **INPUT** : `protobuf.struct`: The arguments inside `input` should match the arguments inside of the `def query(self, question: str)` method defined during the Deploy a basic app configuration.  
(/vertex-ai/generative-ai/docs/model-reference/reasoning-engine#deploy\_a\_basic\_app\_configurationcreation)

HTTP method and URL:

POST `https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJECT_ID `

Request JSON body:

```
{
  "input": {
    INPUT 
  }
}
```

To send your request, choose one of these options:

```
curlPowerShell (#powershell)
(#curl)
```

★ **Note:** The following command assumes that you have logged in to the **gcloud** CLI with your user account by running **gcloud init** (/sdk/gcloud/reference/init) or **gcloud auth login** (/sdk/gcloud/reference/auth/login) , or by using Cloud Shell (/shell/docs), which automatically logs you into the **gcloud** CLI . You can check the currently active account by running **gcloud auth list** (/sdk/gcloud/reference/auth/list).

Save the request body in a file named `request.json`, and execute the following command:

```
curl -X POST \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  -H "Content-Type: application/json; charset=utf-8" \
  -d @request.json \
  "https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJECT_ID
```

## List Reasoning Engines

List reasoning engines in a project.

REST Vertex AI SDK for Python...  
(#rest)

Before using any of the request data, make the following replacements:

- PROJECT\_ID : Your project ID  
(/resource-manager/docs/creating-managing-projects#identifiers).
- PROJECT\_ID : Your project ID  
(/resource-manager/docs/creating-managing-projects#identifiers).
- LOCATION : The region to process the request. Must be us-central1.

HTTP method and URL:

GET https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJECT\_ID

To send your request, choose one of these options:

curl PowerShell (#powershell)  
(#curl)

- ★ **Note:** The following command assumes that you have logged in to the **gcloud** CLI with your user account by running **gcloud init** (/sdk/gcloud/reference/init) or **gcloud auth login** (/sdk/gcloud/reference/auth/login) , or by using Cloud Shell (/shell/docs), which automatically logs you into the **gcloud** CLI . You can check the currently active account by running **gcloud auth list** (/sdk/gcloud/reference/auth/list).



Execute the following command:

```
curl -X GET \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  "https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJ
```

## Get Reasoning Engine

Get details of a reasoning engine.

RESTVertex AI SDK for Python...  
(#rest)

Before using any of the request data, make the following replacements:

- PROJECT\_ID : Your project ID  
(/resource-manager/docs/creating-managing-projects#identifiers).
- PROJECT\_ID : Your project ID  
(/resource-manager/docs/creating-managing-projects#identifiers).
- LOCATION : The region to process the request. Must be us-central1.
- REASONING\_ENGINE\_ID : The ID of the reasoning engine.

HTTP method and URL:

GET https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJECT\_ID

To send your request, choose one of these options:

curlPowerShell (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have logged in to the **gcloud** CLI with your user account by running **gcloud init** (/sdk/gcloud/reference/init) or **gcloud auth login** (/sdk/gcloud/reference/auth/login) , or by using Cloud Shell (/shell/docs), which automatically

logs you into the **gcloud** CLI . You can check the currently active account by running **gcloud auth list** (/sdk/gcloud/reference/auth/list).

Execute the following command:

```
curl -X GET \
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \
  "https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJ
```

## Delete Reasoning Engine

Delete a reasoning engine.

**REST** Vertex AI SDK for Python...  
(#rest)

Before using any of the request data, make the following replacements:

- **PROJECT\_ID** : Your project ID  
(/resource-manager/docs/creating-managing-projects#identifiers).
- **LOCATION** : The region to process the request. Must be `us-central1`.
- **REASONING\_ENGINE\_ID** : The ID of the reasoning engine.

HTTP method and URL:

DELETE `https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJECT_ID`


To send your request, choose one of these options:

**curl** PowerShell (#powershell)  
(#curl)

★ **Note:** The following command assumes that you have logged in to the **gcloud** CLI with your user account by running **gcloud init** (/sdk/gcloud/reference/init) or **gcloud auth login** (/sdk/gcloud/reference/auth/login) , or by using Cloud Shell (/shell/docs), which automatically

logs you into the **gcloud** CLI . You can check the currently active account by running **gcloud auth list** (/sdk/gcloud/reference/auth/list).

Execute the following command:

```
curl -X DELETE \  
  -H "Authorization: Bearer $(gcloud auth print-access-token)" \  
  "https://LOCATION -aiplatform.googleapis.com/v1beta1/projects/PROJ
```

## What's next

- Learn more about using [Vertex AI client libraries](/vertex-ai/generative-ai/docs/reference/libraries) (/vertex-ai/generative-ai/docs/reference/libraries).

Except as otherwise noted, the content of this page is licensed under the [Creative Commons Attribution 4.0 License](https://creativecommons.org/licenses/by/4.0/) (https://creativecommons.org/licenses/by/4.0/), and code samples are licensed under the [Apache 2.0 License](https://www.apache.org/licenses/LICENSE-2.0) (https://www.apache.org/licenses/LICENSE-2.0). For details, see the [Google Developers Site Policies](https://developers.google.com/site-policies) (https://developers.google.com/site-policies). Java is a registered trademark of Oracle and/or its affiliates.

Last updated 2025-06-06 UTC.