

# What is Amazon Bedrock?

Amazon Bedrock is a fully managed service that makes base models from Amazon and third-party model providers accessible through an API.

---

## Features of Amazon Bedrock

With Amazon Bedrock, you can explore the following capabilities:

- **Text playground** – A hands-on text generation application in the AWS Management Console.
  - **Image playground** – A hands-on image generation application in the console.
  - **Chat playground** – A hands-on conversation generation application in the console.
  - **Examples library** – Example use cases to load.
  - **Amazon Bedrock API** – Explore with the AWS CLI, or use the API to access the base models.
  - **Embeddings** – Use the API to generate embeddings from the Titan Embeddings G1 - Text model.
  - **Provisioned Throughput** – Purchase throughput to run inference on models at discounted rates.
- 

## Supported models in Amazon Bedrock

For details about the Amazon Bedrock model providers and their models and model IDs see the **Base models** pages in the Amazon Bedrock console. You can also use the [ListFoundationModels](#) API operation to retrieve information about the current list of models.

Amazon Bedrock supports the following models:

- AI21 Labs
  - Jurassic-2 Ultra
  - Jurassic-2 Mid

- Amazon
  - Titan Text G1 - Express
  - Titan Embeddings G1 - Text
- Anthropic
  - Claude v1.x
  - Claude v2.x
  - Claude Instant v1.x
- Cohere
  - Command v14.7
  - Command Light v14.7
  - Embed English v3.0
  - Embed Multilingual v3.0
- Meta
  - Llama 2 Chat 13B
- Stability.ai
  - Stable Diffusion XL 0.x
  - Stable Diffusion XL 1.x

## Model invocation logging

**Model invocation logging** can be used to collect invocation logs, model input data, and model output data for all invocations in your AWS account used in Amazon Bedrock. By default, logging is disabled.

With invocation logging, you can collect the full request data, response data, and metadata associated with all calls performed in your account. Logging can be configured to provide the destination resources where the log data will be published. Supported destinations include Amazon CloudWatch Logs and Amazon Simple Storage Service (Amazon S3). Only destinations from the same account and region are supported.

Before you can enable invocation logging, you need to set up an Amazon S3 or CloudWatch Logs destination. You can enable invocation logging through either the console or the API.

Prompt engineering

Welcome to the prompt engineering guide for large language models (LLMs) on Amazon Bedrock. Amazon Bedrock is Amazon's service for foundation models (FMs), which offers access to a range of powerful FMs for text and images.

*Prompt engineering* refers to the practice of optimizing textual input to LLMs to obtain desired responses. Prompting helps LLMs perform a wide variety of tasks, including classification, question answering, code generation, creative writing, and more. The quality of prompts that you provide to LLMs can impact the quality of their responses. These guidelines provide you with all the necessary information to get started with prompt engineering. It also covers tools to help you find the best possible prompt format for your use case when using LLMs on Amazon Bedrock.

Whether you're a beginner in the world of generative AI and language models, or an expert with previous experience, these guidelines can help you optimize your prompts for Amazon Bedrock text models. Experienced users can skip to the *General Guidelines for Amazon Bedrock LLM Users* or *Prompt Templates and Examples for Amazon Bedrock Text Models* sections.

## What is a prompt?

Prompts are a specific set of inputs provided by you, the user, that guide LLMs on Amazon Bedrock to generate an appropriate response or output for a given task or instruction.

### User Prompt:

*Who invented the airplane?*

When queried by this prompt, Amazon Titan provides an output:

### Output:

*The Wright brothers, Orville and Wilbur Wright are widely credited*

*with inventing and manufacturing the world's first successful airplane.*

(Source of prompt: AWS, model used: Amazon Titan Text G1 - Express)

A single prompt includes several components, such as the task or instruction you want the LLMs to perform, the context of the task (for example, a description of the relevant domain), demonstration examples, and the input text that you want LLMs on Amazon Bedrock to use in its response. Depending on your use case, the availability of the data, and the task, your prompt should combine one or more of these components.

Consider this example prompt asking Amazon Titan to summarize a review:

**User Prompt:**

*The following is text from a restaurant review:*

*"I finally got to check out Alessandro's Brilliant Pizza and it is now*

*one of my favorite restaurants in Seattle. The dining room has a beautiful view*

*over the Puget Sound but it was surprisingly not crowded. I ordered*

*the fried castelvetrano olives, a spicy Neapolitan-style pizza*

*and a gnocchi dish. The olives were absolutely decadent, and the pizza came*

*with a smoked mozzarella, which was delicious. The gnocchi was fresh and wonderful.*

*The waitstaff were attentive, and overall the experience was lovely.*

*I hope to return soon."*

*Summarize the above restaurant review in one sentence.*

(Source of prompt: AWS)

Based on this prompt, Amazon Titan responds with a succinct one-line summary of the restaurant review. The review mentions key facts and conveys the main points, as desired.

**Output:**

*Alessandro's Brilliant Pizza is a fantastic restaurant in Seattle with a beautiful view over Puget Sound, decadent and delicious food, and excellent service.*

(Model used: Amazon Titan Text G1 - Express)

The instruction **Summarize the above restaurant review in one sentence** and the review text **I finally got to check out ...** were both necessary for this type of output. Without either one, the model would not have enough information to produce a sensible summary. The *instruction* tells the LLM what to do, and the text is the *input* on which the LLM operates. The *context* (**The following is text from a restaurant review**) provides additional information and keywords that guide the model to use the input when formulating its output.

In the example below, the text **Context: Climate change threatens people with increased flooding ...** is the *input* which the LLM can use to perform the *task* of answering the question **Question: What organization calls climate change the greatest threat to global health in the 21st century?"**.

**User prompt:**

*Context: Climate change threatens people with increased flooding,*

*extreme heat, increased food and water scarcity, more disease, and economic loss.*

*Human migration and conflict can also be a result. The World Health Organization (WHO)*

*calls climate change the greatest threat to global health in the 21st century.*

*Adapting to climate change through efforts like flood control measures*

*or drought-resistant crops partially reduces climate change risks,*

*although some limits to adaptation have already been reached.*

*Poorer communities are responsible for a small share of global emissions,*

*yet have the least ability to adapt and are most vulnerable to climate change.*

*The expense, time required, and limits of adaptation mean its success hinge*

*on limiting global warming.*

*Question: What organization calls climate change the greatest threat*

*to global health in the 21st century?*

(Source of prompt: [https://en.wikipedia.org/wiki/Climate\\_change](https://en.wikipedia.org/wiki/Climate_change))

AI21 Jurassic responses with the correct name of the organization according to the context provided in the prompt.

**Output:**

*The World Health Organization (WHO) calls climate change the greatest threat to global health in the 21st century.*

(Model used: AI21 Jurassic-2 Ultra v1)

---

## Few-shot prompting vs. zero-shot prompting

It is sometimes useful to provide a few examples to help LLMs better calibrate their output to meet your expectations, also known as *few-shot prompting* or *in-context learning*, where a *shot* corresponds to a paired example input and the desired output. To illustrate, first here is an example of a zero-shot sentiment classification prompt where no example input-output pair is provided in the prompt text:

**User prompt:**

*Tell me the sentiment of the following headline and categorize it*

*as either positive, negative or neutral:*

*New airline between Seattle and San Francisco offers a great opportunity*

*for both passengers and investors.*

(Source of prompt: AWS)

**Output:**

*Positive*

(Model used: Amazon Titan Text G1 - Express)

Here is the few-shot version of a sentiment classification prompt:

**User prompt:**

*Tell me the sentiment of the following headline and categorize it*

*as either positive, negative or neutral. Here are some examples:*

*Research firm fends off allegations of impropriety over new technology.*

*Answer: Negative*

*Offshore windfarms continue to thrive as vocal minority in opposition dwindles.*

*Answer: Positive*

*Manufacturing plant is the latest target in investigation by state officials.*

*Answer:*

(Source of prompt: AWS)

**Output:**

*Negative*

(Model used: Amazon Titan Text G1 - Express)

The following example uses Claude models. When using Claude models, it's a good practice to use `<example></example>` tags to include demonstration examples. We also recommend using different delimiters such as H: and A: in the examples to avoid confusion with the delimiters Human: and Assistant: for the whole prompt.

**User prompt:**



*Human: Please classify the given email as "Personal" or "Commercial" related emails.*

*Here are some examples.*

*<example>*

*H: Hi Tom, it's been long time since we met last time. We plan to have a party at my house this weekend. Will you be able to come over?*

*A: Personal*

*</example>*

*<example>*

*H: Hi Tom, we have a special offer for you. For a limited time, our customers can save up to 35% of their total expense when you make reservations within two days. Book now and save money!*

*A: Commercial*

*</example>*

*H: Hi Tom, Have you heard that we have launched all-new set of products. Order now, you will save \$100 for the new products. Please check our website.*

*A:*

*Assistant:*

**Output:**

*Commercial*

(Source of prompt: AWS, model used: Claude)

---

## Prompt template

A prompt template specifies the formatting of the prompt with exchangeable content in it. Prompt templates are “recipes” for using LLMs for different use cases such as classification, summarization, question answering, and more. A prompt template may include instructions, few-shot examples, and specific context and questions appropriate for a given use case. The following example is a template that you can use to perform few-shot sentiment classification using Amazon Bedrock text models:

**Prompt template:**

*Tell me the sentiment of the following*

*{{Text Type, e.g., “restaurant review”}} and categorize it  
as either {{Sentiment A}} or {{Sentiment B}}.*

*Here are some examples:*

*Text: {{Example Input 1}}*

*Answer: {{Sentiment A}}*

*Text: {{Example Input 2}}*

*Answer: {{Sentiment B}}*

*Text: {{Input}}*

*Answer:*

Users can copy and paste this template, filling in their own text and few-shot examples to complete prompts when using LLMs on Amazon Bedrock. More templates are provided in this section.

Note

The double curly braces `{{` and `}}` mark the places to put in data-specific information in the template, and should not be included in the prompt text.

---

## Important notes on using Amazon Bedrock LLMs by API calls

If you are accessing LLMs on Amazon Bedrock via API calls, please be aware that they do not recall prior prompts and previous requests unless the previous interaction is included within the current prompt. You may need this for conversational-style interactions or follow up requests.

If you are accessing Claude models using API calls instead of the Amazon Bedrock text playground, you wrap your prompts in a conversational style to get desired response (for the Amazon Titan Text and AI21 Jurassic model family, it's not necessary). The following example demonstrates a conversational-style prompt:

**User prompt:**

*Human: Describe where sea otters live in two sentences.*

*Assistant:*

## Output:

*Here is a two sentence description of where sea otters live:*

*Sea otters live in the northern Pacific Ocean along the coasts of*

*Russia, Alaska, Canada, and the continental United States.*

*They prefer coastal habitats like rocky shores, kelp forests, and estuaries where they have access to shallow, nearshore waters.*

(Source of prompt: AWS, model used: Claude)

Notice how the main content of the prompt is wrapped like this: `\n\nHuman: {{Main Content}} \n\nAssistant: .` For Claude models, prompts sent via the API must contain `\n\nHuman:` and `\n\nAssistant: .`

To use conversational mode on Amazon Titan, you can use the format of `User: {{{`  
`\n Bot:`  when prompting the model.