Prompt Engineering

Generative AI and large language models are proving to be powerful tools, but to leverage their capabilities, it's important to understand their architecture. It's also important to consider recommended practices when implementing these technologies.

The goal of this module, Prompt Engineering, is to help with these important steps.

Prompt Engineering

01 What is generative AI?

02 What is a large language model?

03 What is prompt engineering?

04 Prompt engineering best practices

In this guide to prompt engineering, you'll get answers to the questions:

- What is generative AI?
- What is a large language model?
- What is prompt engineering?

You'll also explore prompt engineering best practices.

Prompt Engineering

01 What is generative AI?

02 What is a large language model?

03 What is prompt engineering?

04 Prompt engineering best practices

Before we delve into this module, let's define the interchangeably used terms such as 'generative AI' and 'Large Language Model' (LLM).

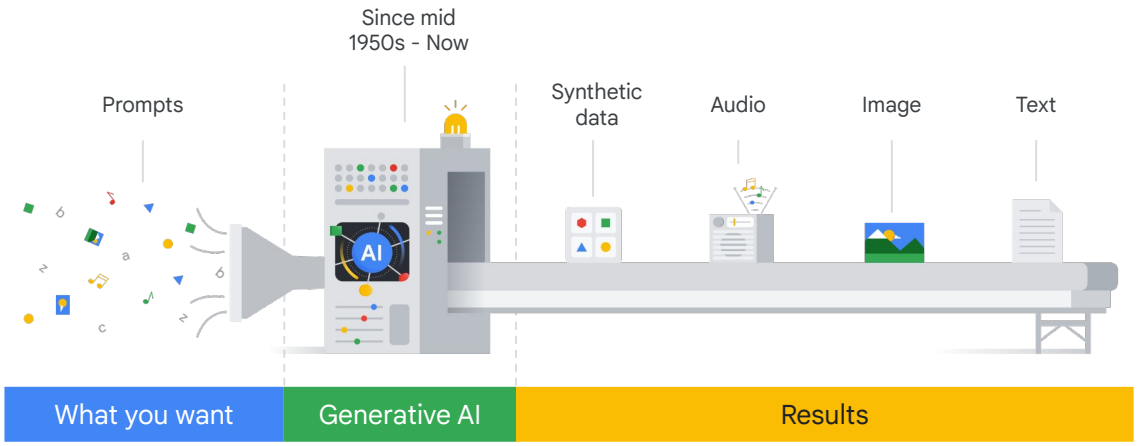
While both terms describe AI models capable of generating human-like responses based on input prompts in many references, it's important to note they're not identical.

Generative AI encompasses a broader range of models capable of generating various types of content beyond just text, while LLM specifically refers to a subset of generative AI models focusing on language tasks.

We'll thoroughly explore each term in this module.

So let's begin with an important question: What is generative AI?

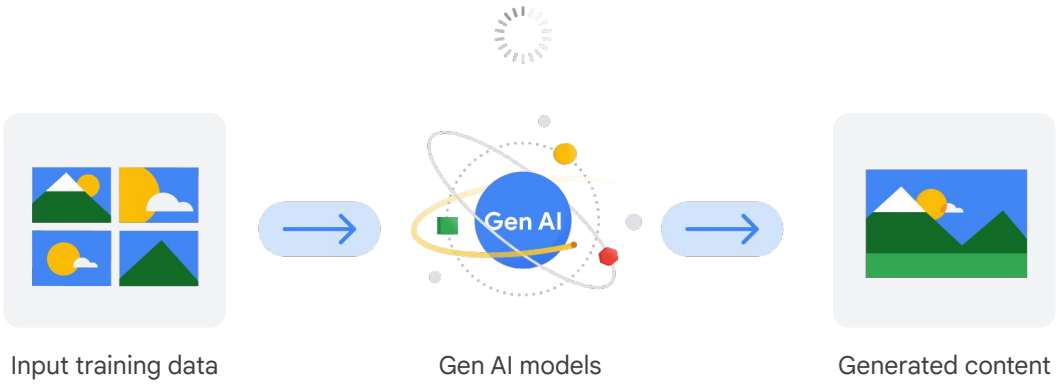
Generative AI



Generative artificial intelligence, which is commonly referred to as gen AI, is a subset of artificial intelligence that is capable of creating text, images, or other data using generative models, often in response to prompts. It has grown in popularity hugely since 2021 but artificial intelligence has been around since the mid 1950s.

By the way, a prompt is a specific instruction, question, or cue given to a computer program or user to initiate a specific action or response, but we examine this more later.

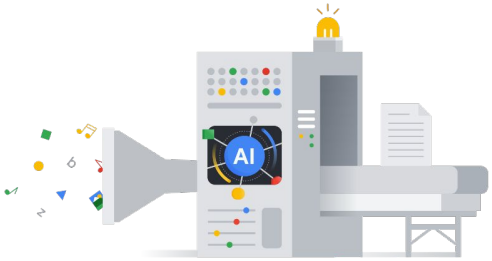
Generative AI



In its current format, gen AI models are like conversational programs that can generate content based on the inputs supplied.

Gen AI models learn the patterns and structure from input training data and then create new data with similar characteristics.

Generative AI



Generative AI

Software development

Entertainment

Healthcare

Customer service

Finance

Sales

Generative AI has uses across a wide range of industries, including software development, healthcare, finance, entertainment, customer service, and sales.

However, rather than exploring all generative AI applications, this training will specifically focus on articulating prompts to harness the power of gen AI effectively.

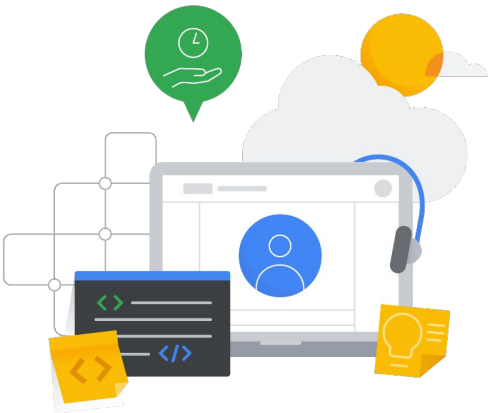
Let me introduce a scenario that we'll make reference to throughout this section of the training. We've included it to help put some of this theory into practice.

Meet Sasha

Sasha, a cloud architect, needs to create a prototype design of Google Cloud VPC network architecture for Cymbal Bank.

Sasha wants to use generative AI tools to get started—to save time and to quickly create a usable prototype design.

Having the tool inside the Google Cloud Console means that she can access it without any additional installs.



Sasha

Meet Sasha, a cloud architect, who needs to create a prototype design of Google Cloud VPC network architecture for Cymbal Bank.

Sasha wants to save time by combining her existing knowledge of cloud architecture and generative AI tools to create a usable prototype design.

Sasha was excited to learn about Gemini, since having the tool inside the Google Cloud Console means that she can access it without any additional installs.

We'll check back in with Sasha later.

Prompt Engineering

01 What is generative AI?

02 What is a large language model?

03 What is prompt engineering?

04 Prompt engineering best practices

Let's spend some time exploring large language models, which are a highly sophisticated computer programs trained on gigantic amounts of data that can be text or images. But how are they trained? And why do they need training at all?

Large Language Models

LLMs are large, general-purpose language models that can be pre-trained and then fine-tuned for specific purposes.

Large

- The size of the training dataset
- The number of parameters

General purpose

- Can solve common problems.

Pre-trained and then fine-tune

- Pre-trained with a large dataset.
- Fine-tuned with a smaller dataset.

Large language models refer to large, general-purpose language models that can be pre-trained and then fine-tuned for specific purposes.

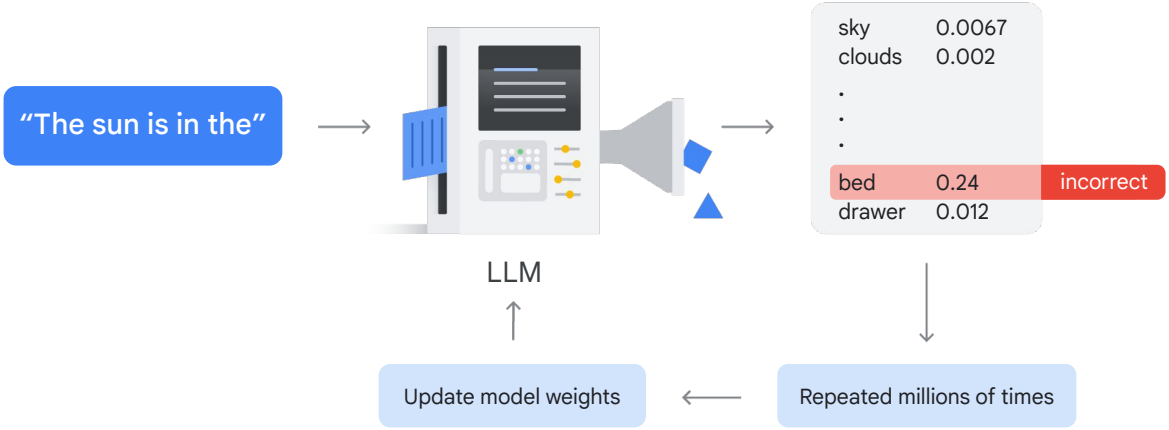
In this context, large refers to:

- The size of the **training dataset**, which can sometimes be at the petabyte scale.
- The number of **parameters**. Parameters are the memories and knowledge that the machine has learned during model training. They determine the ability of a model to solve a problem, such as predicting text, and can reach billions or even trillions in size.

General-purpose means that the models can sufficiently solve common problems. This is thanks to the commonality of a human language, regardless of the specific tasks.

Saying LLMs are **pre-trained and fine-tuned**, means that they have been pre-trained for a general purpose with a large dataset and then fine-tuned for specific goals with a much smaller dataset.

How are LLMs trained?



But how are LLMs trained?

When you submit a prompt to an LLM, it calculates the probability of the correct answer from its pre-trained model.

The probability is determined through a task called pre-training.

Pre-training an LLM involves feeding a massive dataset of text, images, and code to the model so that it can learn the underlying structure and patterns of the language.

This process helps the model to understand and generate human language more effectively.

Language model pre-training

In this way, the LLM works like a fancy autocomplete, suggesting the most common correct response to the prompt.

Hallucinations

But sometimes the LLM gives a completely wrong answer. This is called a hallucination.

Hallucinations are words or phrases that are generated by the model that are often nonsensical or grammatically incorrect.

This happens because LLMs can only understand the information they were trained on.

This means that they might not be aware of your business's proprietary or domain-specific data. Also, they do not have access to real-time information.

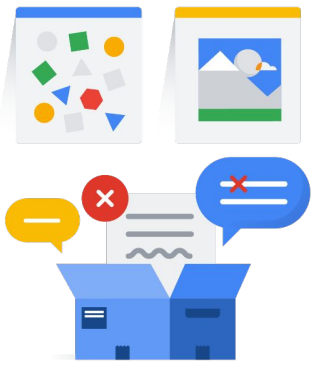
To make matters worse, LLMs only understand the information that is explicitly given to them in the prompt. In other words, they often assume that the prompt is true. They also do not have the ability to ask for more context information.

Ultimately, an LLM does not know anything outside of what it was trained on, and it cannot truly know if that information is accurate.

Hallucinations

Challenges

- The model is not trained on enough data
- The model is trained on noisy or dirty data
- The model is not given enough context
- The model is not given enough constraints



But what causes a hallucination. Hallucinations can be caused by a number of factors, including:

- The model is not trained on enough data.
- The model is trained on noisy or dirty data.
- The model is not given enough context.
- The model is not given enough constraints.

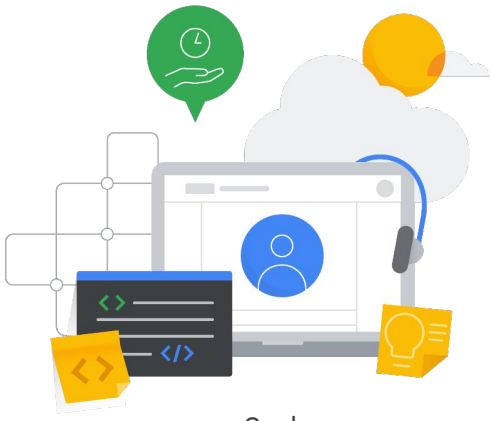
Hallucinations can be a problem for LLMs because they can make the output text difficult to understand. They can also make the model more likely to generate incorrect or misleading information. But we will see in the Prompt Engineering section that there are things we can do to minimize this problem.

Meet Sasha

Sasha, a cloud architect, needs to create a prototype design of Google Cloud VPC network architecture for Cymbal Bank.

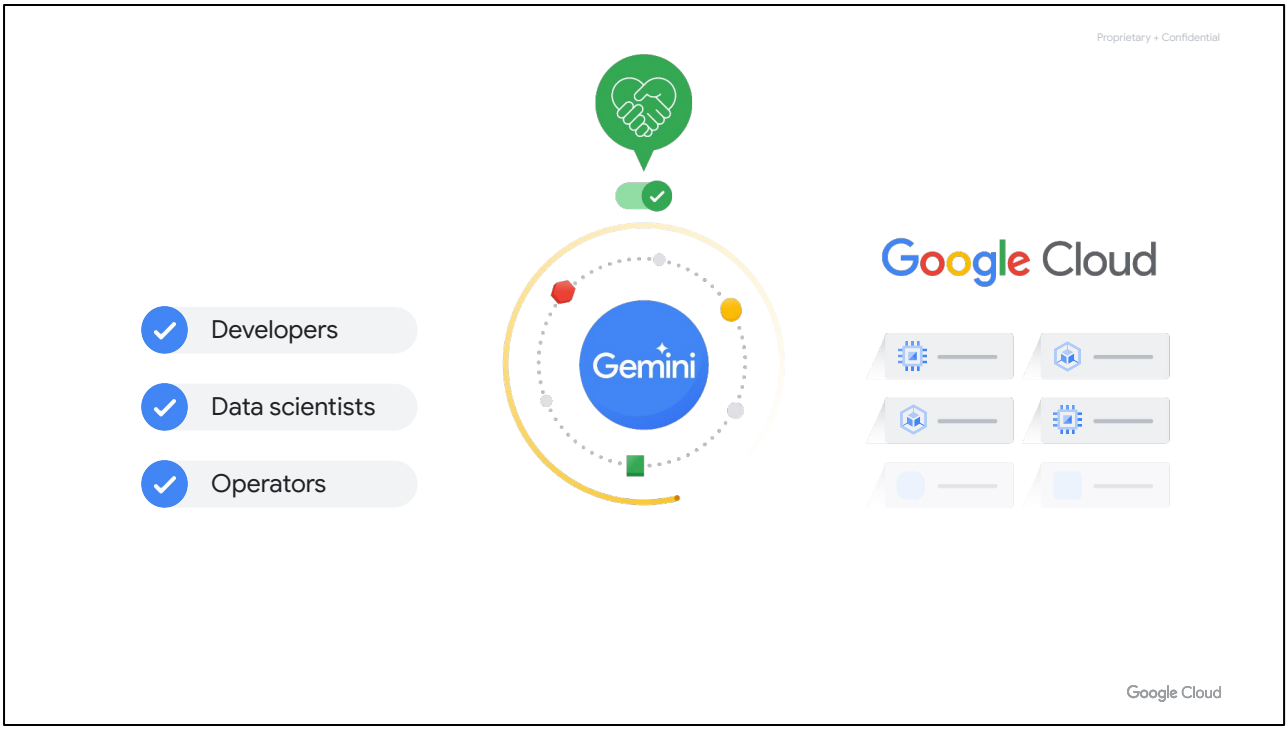
Sasha wants to use generative AI tools to get started—to save time and to quickly create a usable prototype design.

Having the tool inside the Google Cloud Console means that she can access it without any additional installs.



Sasha

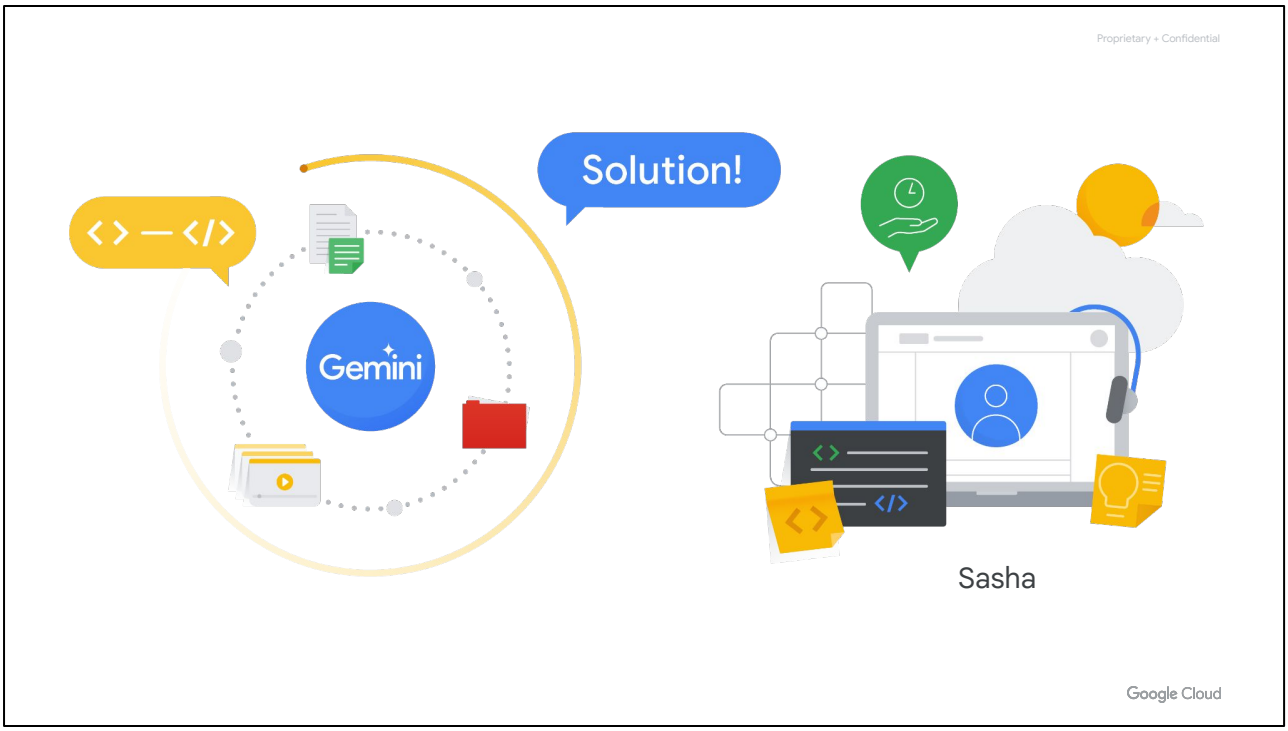
OK, but knowing where the sun is will not help Sasha with her current task.



Lucky for Sasha, Google Cloud offers a generative AI model called Gemini, which can act as an always-on collaborator.

This gen AI-powered assistant can help a wide range of Google Cloud users, including developers, data scientists, and operators.

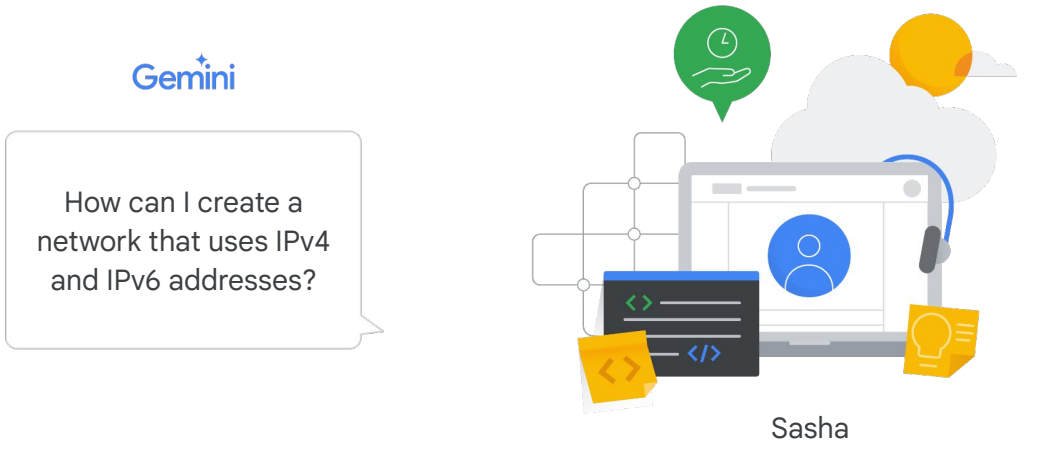
To provide an integrated assistance experience, Gemini is embedded in many Google Cloud products.



Gemini has access to a massive range of data, including Google Cloud documentation, tutorials, and samples.

With the right prompts, it can produce detailed suggestions and guides on what resources will best suit Sasha's current challenge and their configuration.

Gemini can even create detailed gcloud commands and insert them into Cloud Shell for her.



She just needs to articulate her needs in a way that gets the best response from Gemini.

For example, if she uses the prompt "**How can I create a network that uses IPv4 and IPv6 addresses?**", she will get a response that details how to do just that.

Prompt Engineering

01 What is generative AI?

02 What is a large language model?

03 What is prompt engineering?

04 Prompt engineering best practices

You've learned that a large language model is a huge object model containing a massive dataset of text. But how can you extract the information you need from this dataset? This is where prompt engineering comes in.

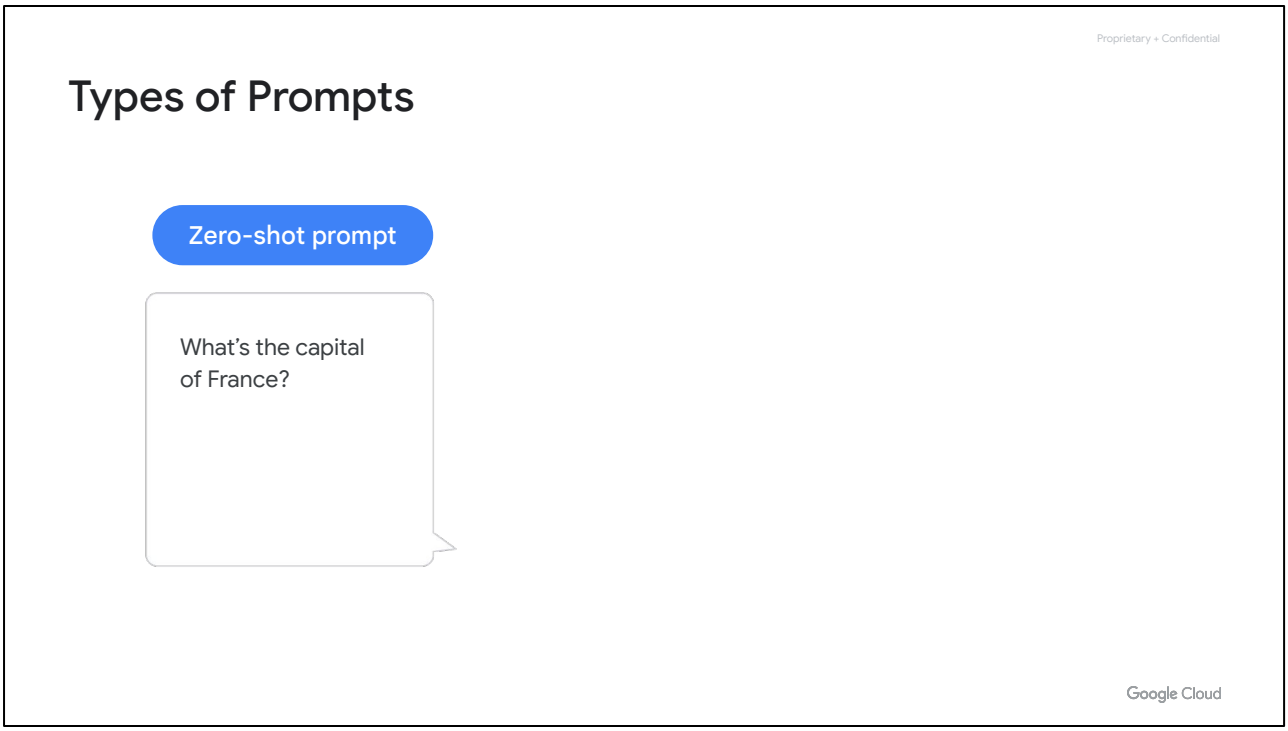
Prompt: the text you feed to your model

Prompt Engineering
(=Prompting)
(=Prompt Design)
(=Priming)
(=In-context learning):

The art and science of figuring out what text to feed your language model to get it to take on the behavior you want.

A prompt is the text that you feed to the model, and prompt engineering is a way of articulating your prompts to get the best response from the model.

The better structured a prompt is, the better the output from the model will be. Let's explore what this means.



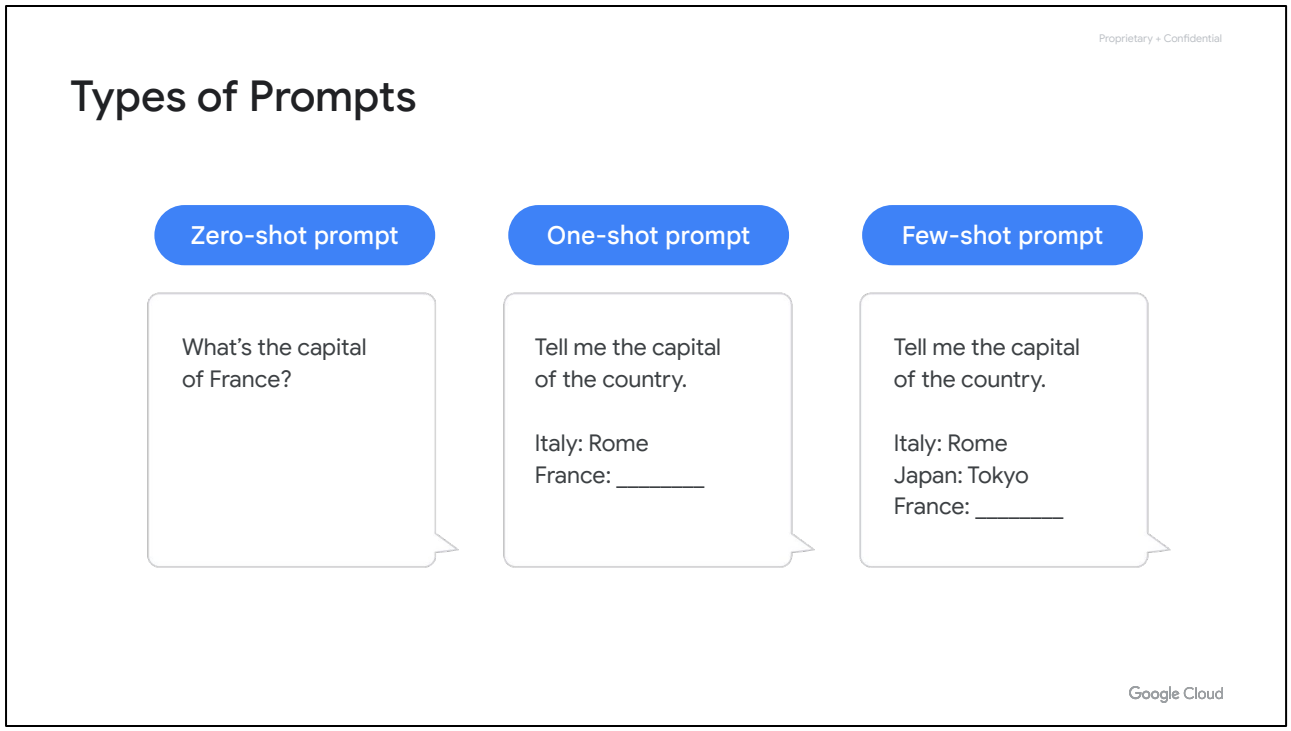
Prompts can be in the form of a question, and are categorized into four categories: zero-shot, one-shot, few-shot, and role prompts.

Zero-shot prompts do not contain any context or examples to assist the model.

For example, the prompt "What's the capital of France?" does not provide any examples of what a capital is. Clearly, that is not too important for this example. But for more specific and technical prompts, an example would help refine the scope of the response from Gemini.

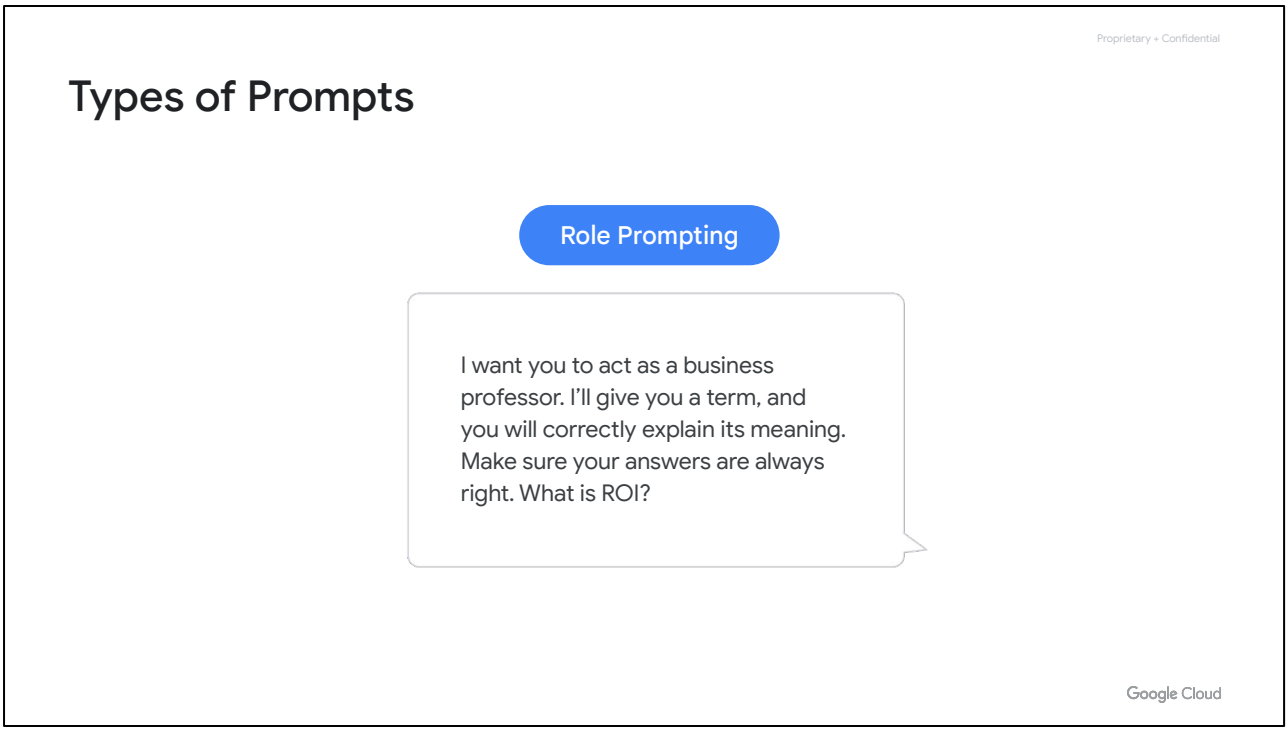
Types of Prompts

One-shot prompts, however, provide one example to the model for context. Here, we ask for the capital of France again, but we provide Italy and Rome as an example.



And few-shot prompts provide at least two examples to the model for context.

Here, our prompt is updated to also include Japan and Tokyo in our examples.

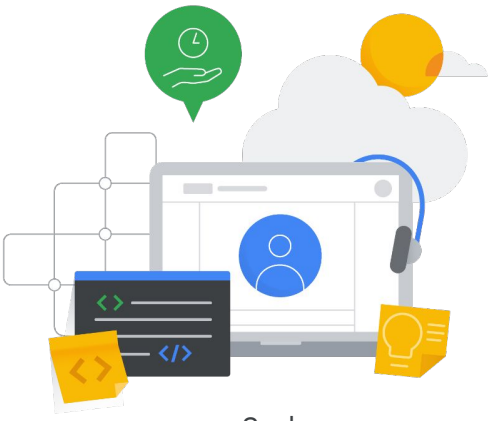


And then, there are role prompts which require a frame of reference for the model to work from as it answers the questions. In our example, we state “I want you to act as a business professor. I’ll give you a term, and you will correctly explain its meaning. Make sure your answers are always right. What is ROI? “

What prompt will Sasha use?

Gemini

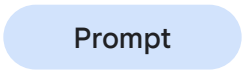
How can I use gcloud to create a VPC?



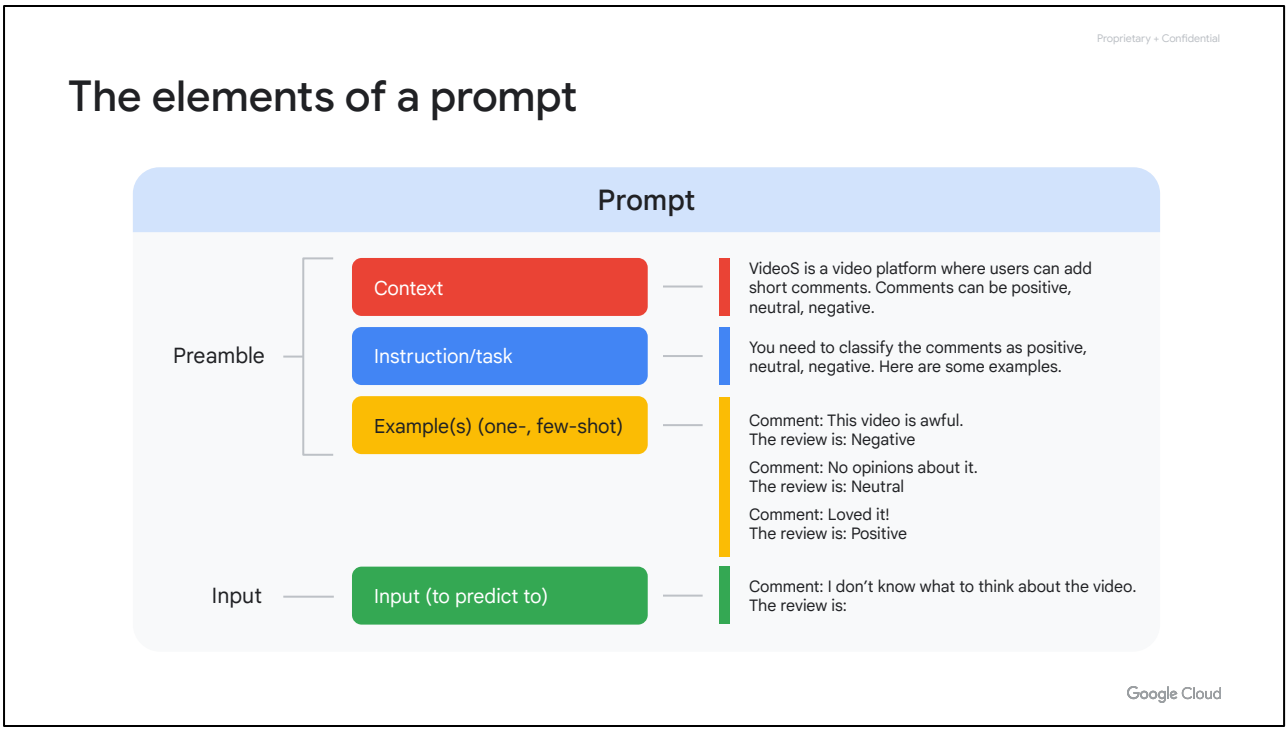
Sasha

For Sasha's needs, using role prompts might be the best solution. She can define what is required and in what context. This means that the LLM will have a clear point of reference when supplying an answer.

The elements of a prompt



Now that you've seen the types of prompts you can create, let's explore the two elements of a prompt: the preamble and the input.



The preamble refers to the introductory text you provide to give the model context and instructions before your main question or request. Think of it as setting the stage for the LLM to better understand what you want. It can include the context for the task, the task itself, and some examples to guide the model.

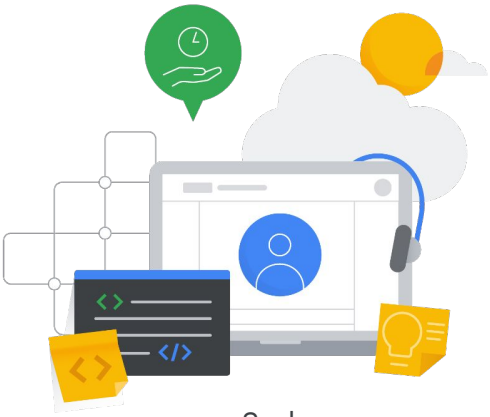
The input is the central request you're making to the LLM. It's what the instruction or task will act upon, for example "**Comment: I don't know what to think about the video. The review is**:"

Based on the preamble, Gemini reviews the input and suggests if the review is positive, neutral, or negative.

It is worth noting that not all the components are required for a prompt, and the format can change depending on the task at hand. The element order can also change.

What prompt will Sasha use?

How can I create a network that uses IPv4 and IPv6 addresses?



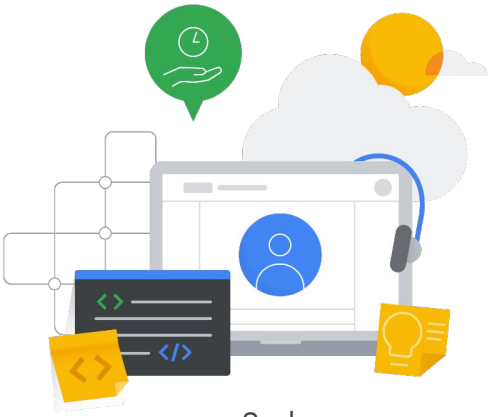
Sasha

Let's amend Sasha's original prompt "How can I create a network that uses IPv4 and IPv6 addresses?" and add a role context to the input fed into Gemini. She also adds the detail of needing a dual stack subnet.

What prompt will Sasha use?

How can I create a network that uses IPv4 and IPv6 addresses?

I want you to act as a cloud architect in Google Cloud. How can I use gcloud to create a network that uses IPv4 and IPv6 subnets?



Sasha

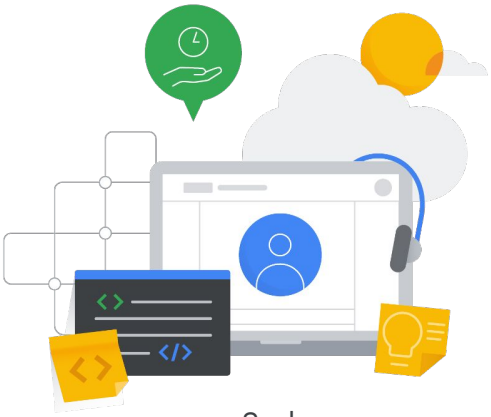
The new prompt is “I want you to act as a cloud architect in Google Cloud. How can I use gcloud to create a network that uses IPv4 and IPv6 subnets?”

What prompt will Sasha use?

How can I create a network that uses IPv4 and IPv6 addresses?

I want you to act as a Cloud Architect in Google Cloud. How can I use gcloud to create a network that uses IPv4 and IPv6 subnets?

I want you to act as a cloud architect in Google Cloud. How can I adjust the gcloud command provided to create a subnet to ensure the subnet is dual stack?



Sasha

But since Gemini maintains its own interaction context, she could have just asked “I want you to act as a cloud architect in Google Cloud. How can I adjust the gcloud command provided to create a subnet to ensure the subnet is dual stack?”

Prompt Engineering

01 What is generative AI?

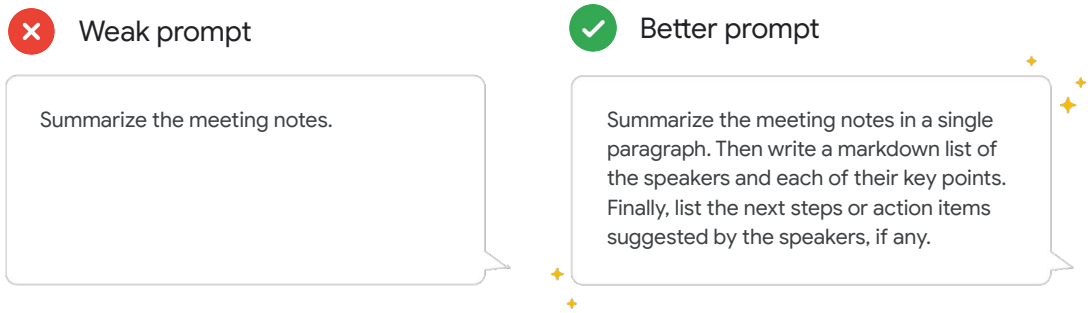
02 What is a large language model?

03 What is prompt engineering?

04 Prompt engineering best practices

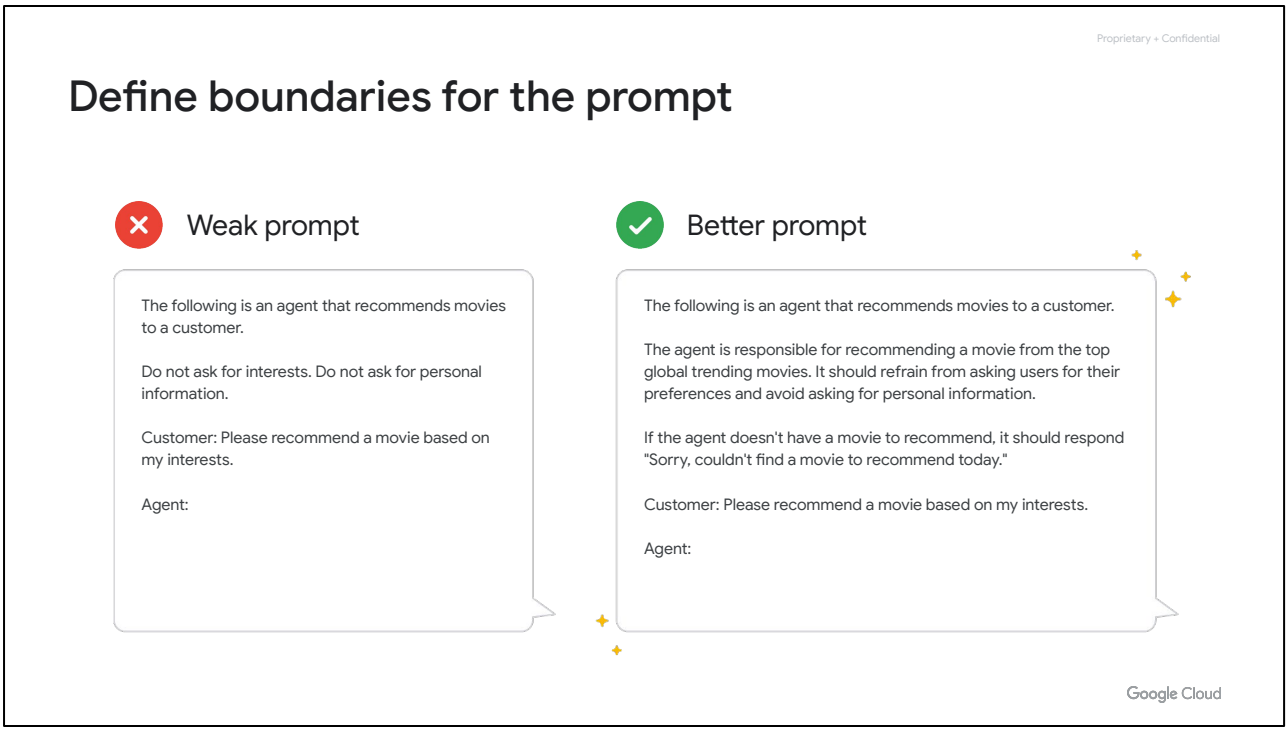
Now that you've had a chance to explore what Gen AI is, what large language models are and how they're trained, and what prompt engineering is, it's time to explore some prompt engineering best practices.

Write detailed and explicit instructions



The first best practice is to write detailed and explicit instructions.

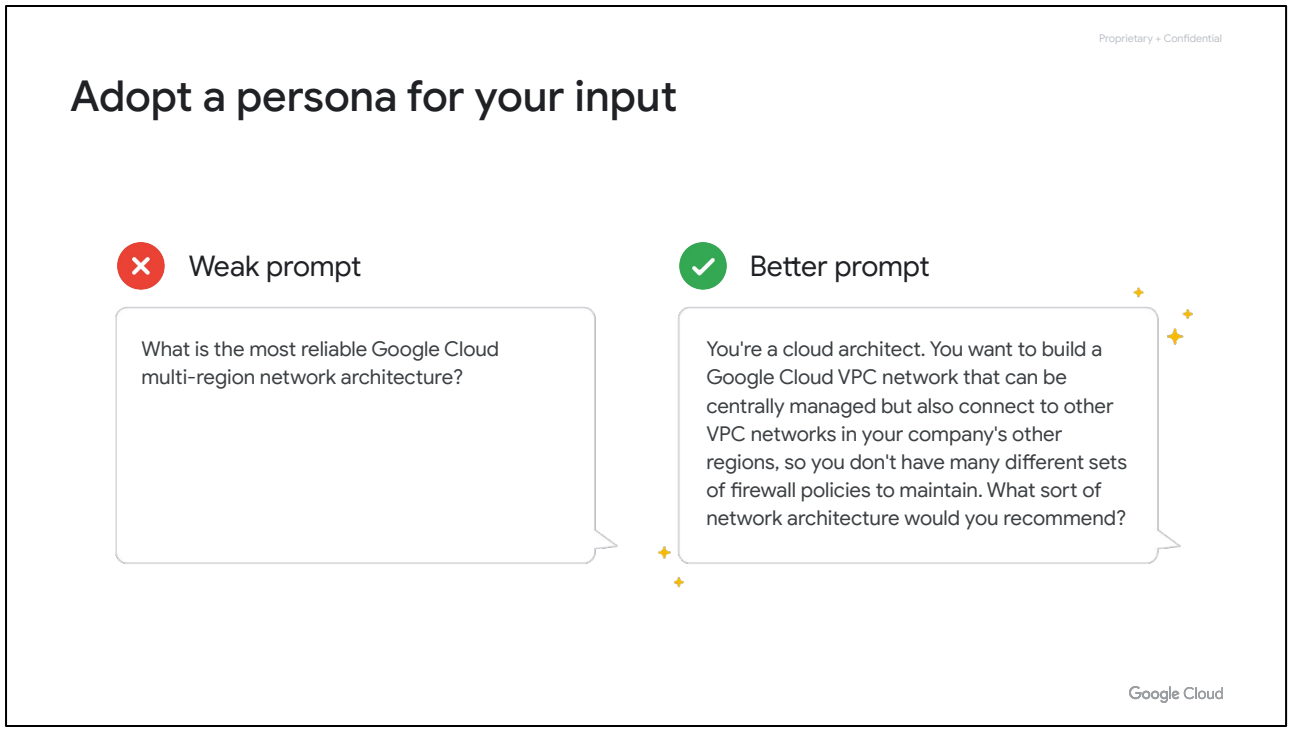
The more vague the prompt, the more chance that the model will produce a result that is not usable. Be clear and concise in the prompts that you feed into the model.



Next, be sure to define boundaries for the prompt.

It's better to instruct the model on *what* to do rather than *what not* to do.

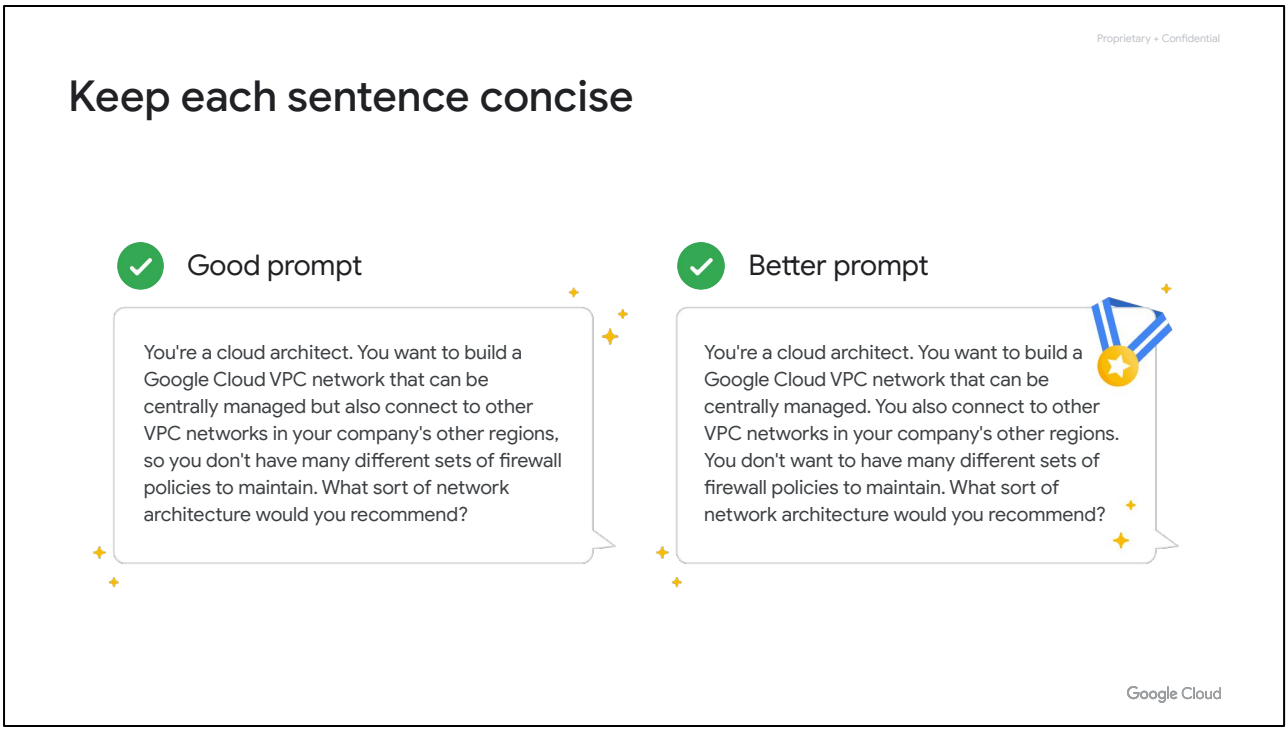
If the model gets stuck, give it a few 'fallback' outputs that work in various situations. For example, something like "I'm still learning about that" to use when unsure.



Another best practice is to **adopt a persona for your input.**

Adding a persona for the model can provide meaningful context to help it focus on related questions, which can help improve accuracy.

This prompt would help Sasha, the cloud architect, get started with prototyping a network architecture for Cymbal Bank.

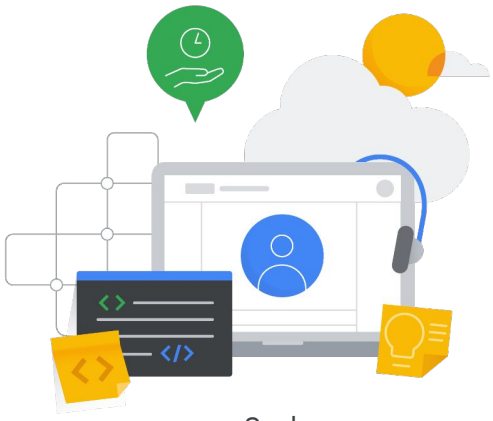


And finally, it's a recommended practice to **keep each sentence concise.**

Longer sentences can sometimes produce suboptimal results. It's best to break long sentences in a prompt into a series of shorter sentences and simpler tasks.

What prompt will Sasha use?

I want you to act as a cloud architect in Google Cloud. How can I adjust the gcloud command provided to create a subnet to ensure the subnet is dual stack?

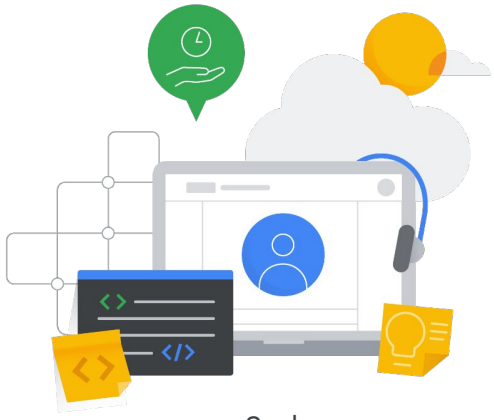


Sasha

So, let's return to Sasha, and use what we have learned so far.

What prompt will Sasha use?

You're a cloud architect. You want to build a Google Cloud VPC network that can be centrally managed. You also connect to other VPC networks in your company's other regions. You don't want to have many different sets of firewall policies to maintain. What sort of network architecture would you recommend?



Sasha

Sasha updates her prompt to:

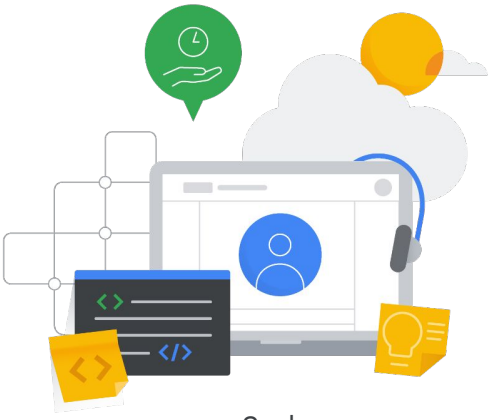
“You're a cloud architect. You want to build a Google Cloud VPC network that can be centrally managed. You also connect to other VPC networks in your company's other regions. You don't want to have many different sets of firewall policies to maintain. What sort of network architecture would you recommend?”

What prompt will Sasha use?

You're a cloud architect. You want to build a Google Cloud VPC network that can be centrally managed. You also connect to other VPC networks in your company's other regions. You don't want to have many different sets of firewall policies to maintain. What sort of network architecture would you recommend?

Gemini

I would recommend a hub-and-spoke network architecture. This architecture consists of a central hub VPC network that is connected to multiple spoke VPC networks. This architecture allows you to centrally manage the firewall policies and routes for your entire network, and it also allows you to connect to other VPC networks in your company's other regions.



Sasha

With this new prompt, Gemini proposes a hub-and-spoke network architecture, which fits Sasha's needs exactly. By refining and amending her prompts, Sasha has articulated her requirements in a way that Gemini can respond with the correct focus and level of detail.