



User Guide

# Amazon Bedrock Studio



# Amazon Bedrock Studio: User Guide

Copyright © 2024 Amazon Web Services, Inc. and/or its affiliates. All rights reserved.

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

---

# Table of Contents

<b>What is Amazon Bedrock Studio?</b> .....	<b>1</b>
Workspaces .....	1
Explore mode .....	1
Build mode .....	2
Are you a first-time Amazon Bedrock Studio user? .....	4
<b>Explore Amazon Bedrock Studio</b> .....	<b>5</b>
What is a prompt? .....	5
Prompt engineering guides .....	6
Use Explore mode to chat with a model .....	7
<b>Build a chat app</b> .....	<b>10</b>
Create an initial chat app .....	10
Add a guardrail .....	16
Add a guardrail .....	16
Add a document data source .....	18
Add a function call .....	20
<b>Export a chat app</b> .....	<b>24</b>
Export your Amazon Bedrock Studio chat app .....	24
Deploy an Amazon Bedrock Studio chat app .....	26
Prerequisites for deploying a chat app .....	27
Deploy the chat app .....	28
Run a deployed chat app .....	29
Prerequisites for running a chat app .....	30
Run the chat app .....	30
Map a chat app to Amazon Bedrock API operations .....	30
Chat app with no components .....	31
Chat app with guardrail component .....	31
Chat app with data source component .....	32
Chat app with function component .....	32
Chat app with function and data source component .....	32
<b>Build a Prompt Flows app</b> .....	<b>34</b>
Create a flows app .....	35
Create an initial Prompt Flow app .....	36
Add a Knowledge Base .....	38
Add a prompt .....	41

---

Add a condition .....	42
Prompt Flow nodes .....	44
Flow input .....	45
Flow output .....	46
Condition .....	46
Prompt node .....	48
Knowledge Base node .....	48
Define inputs with expressions .....	49
Use prompts with your Flows app .....	54
Create a prompt .....	55
Add a prompt to a Prompt Flows app .....	57
Modify a prompt .....	58
Delete a prompt .....	59
<b>Influence model responses with inference parameters .....</b>	<b>60</b>
Randomness and diversity .....	60
<b>Add a data source to your app .....</b>	<b>63</b>
Document data source .....	63
Knowledge Base data source .....	63
Create a Knowledge Base component .....	64
Add a Knowledge Base component to a chat app .....	65
Add a Knowledge Base component to a Prompt Flow app .....	65
<b>Safeguard your app with a guardrail .....</b>	<b>67</b>
Guardrail policies .....	67
Content filters .....	67
Denied topics .....	69
Create a guardrail component .....	70
Add a guardrail component to a chat app .....	71
<b>Call functions from your app .....</b>	<b>73</b>
Function schema .....	73
Valid components for globally-scoped security requirements .....	74
Authentication methods .....	77
Create a function component .....	78
Add a function component to a chat app .....	79
<b>Organize your work with projects .....</b>	<b>80</b>
Create a project .....	80
Share a project .....	82

---

Stop sharing a project .....	83
Delete project resources .....	84
Delete a project .....	85
<b>Document history .....</b>	<b>87</b>

# What is Amazon Bedrock Studio?

Amazon Bedrock Studio is a web app that lets you easily prototype apps that use Amazon Bedrock models and features, without having to set up and use a developer environment. For example, you can use Amazon Bedrock to try a prompt with an Anthropic Claude model without having to write any code. Later, you can use Bedrock Studio to create a prototype app that uses an Amazon Bedrock model and features, such as a Knowledge Base or a Guardrail, again without having to write any code.

To use Amazon Bedrock Studio, you must be a member of a workspace. Your organization will provide you with login details. If you don't have login details, contact your administrator.

## Note

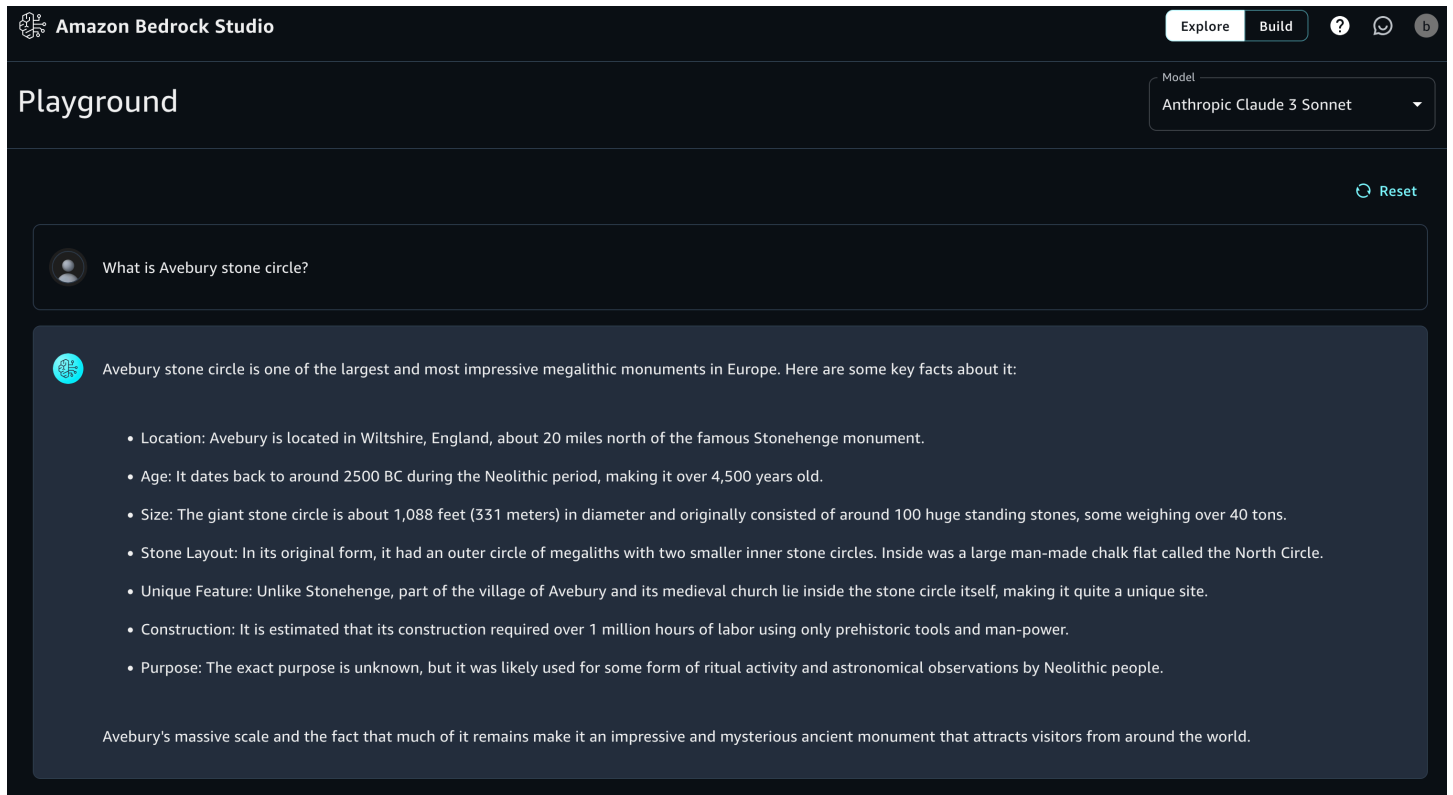
If you are administrator and need information about managing an Amazon Bedrock Studio workspace, see [Manage Amazon Bedrock Studio](#) in the Amazon Bedrock user guide.

## Workspaces

A *workspace* in Amazon Bedrock Studio is where you experiment with Amazon Bedrock models and where you can build Amazon Bedrock enabled apps. As an Amazon Bedrock Studio user, you can use a workspace in two types of user mode, *Explore* or *Build*.

### Explore mode

Explore mode provides a *playground* that lets you easily try a model by sending prompts to the model and viewing the responses. For more information, see [Explore Amazon Bedrock Studio](#).



The screenshot displays the Amazon Bedrock Studio Playground interface. At the top, the title 'Amazon Bedrock Studio' is visible on the left, and navigation buttons for 'Explore', 'Build', and user profile are on the right. The main area is titled 'Playground' and shows a model selection dropdown set to 'Anthropic Claude 3 Sonnet'. A 'Reset' button is located in the top right of the chat area. The chat history shows a user prompt: 'What is Avebury stone circle?'. The AI response provides a comprehensive overview of Avebury stone circle, including its location, age, size, layout, unique features, construction, and purpose.

Model: Anthropic Claude 3 Sonnet

Reset

What is Avebury stone circle?

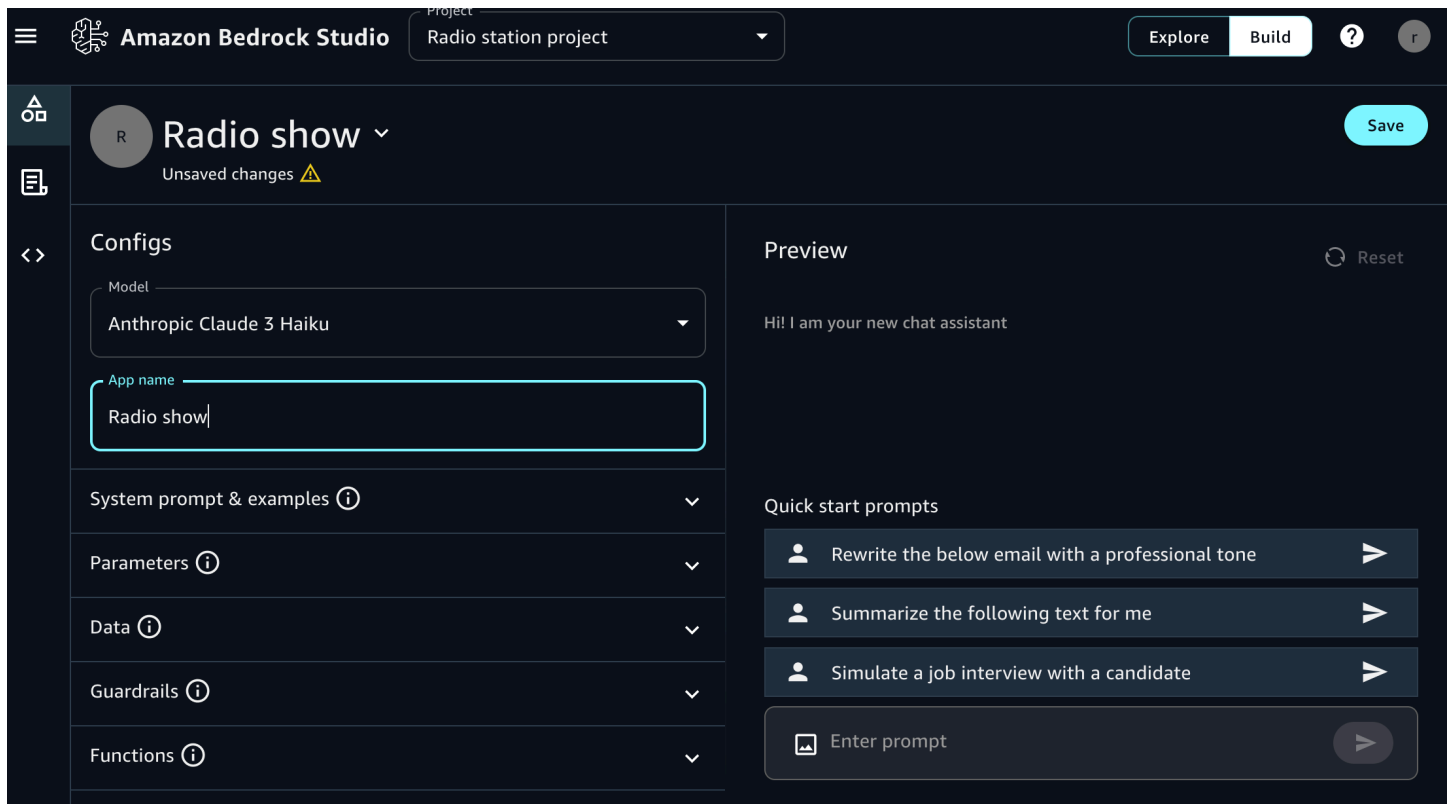
Avebury stone circle is one of the largest and most impressive megalithic monuments in Europe. Here are some key facts about it:

- Location: Avebury is located in Wiltshire, England, about 20 miles north of the famous Stonehenge monument.
- Age: It dates back to around 2500 BC during the Neolithic period, making it over 4,500 years old.
- Size: The giant stone circle is about 1,088 feet (331 meters) in diameter and originally consisted of around 100 huge standing stones, some weighing over 40 tons.
- Stone Layout: In its original form, it had an outer circle of megaliths with two smaller inner stone circles. Inside was a large man-made chalk flat called the North Circle.
- Unique Feature: Unlike Stonehenge, part of the village of Avebury and its medieval church lie inside the stone circle itself, making it quite a unique site.
- Construction: It is estimated that its construction required over 1 million hours of labor using only prehistoric tools and man-power.
- Purpose: The exact purpose is unknown, but it was likely used for some form of ritual activity and astronomical observations by Neolithic people.

Avebury's massive scale and the fact that much of it remains make it an impressive and mysterious ancient monument that attracts visitors from around the world.

## Build mode

Build mode is where you can create apps that use Amazon Bedrock models. You can create two types of apps, a *chat app* and a *Prompt Flows app*. A [chat app](#) allows users to communicate with an Amazon Bedrock model through a conversational interface, typically by sending text messages and receiving responses. A [Prompt Flows app](#) allows you to link prompts, supported foundational models (FMS), and other units of work, such as a Knowledge Base, together and create generative AI workflows for end-to-end solutions.

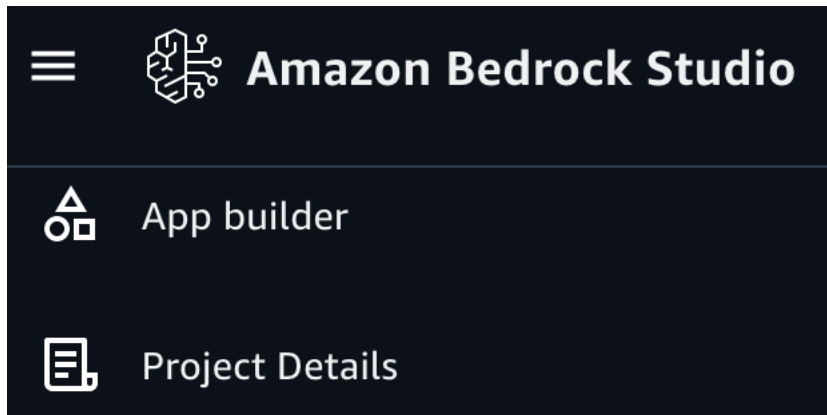


Apps that you create with Bedrock Studio can integrate the following Amazon Bedrock features.

- [Data sources](#) — Enrich apps by including context that is received from querying a Knowledge Base or a document.
- [Guardrails](#) — Lets you implement safeguards for your Bedrock Studio app based on your use cases and responsible AI policies.
- [Functions](#) — Lets a model call a function to access a specific capability when handling a prompt.
- [Prompt Management](#) — Reusable prompts that you can use in a Prompt Flows app.

Within Build mode, you use a *project* to organize the apps, prompts, and components that you use for a solution. A component is an Amazon Bedrock Knowledge Base, Guardrail, or Function. When you open a project, you can access the *App builder* and *Project details* by opening the side navigation menu.





The App builder is one way you can access the chat app or Prompt Flows app that you are currently working on. Project details is where you create and manage the apps, components, and prompts that the apps in your project use. For more information, see [Organize your work with projects in Amazon Bedrock Studio](#).

If you work on a team, you can collaborate by sharing a project with other team members. For more information, see [Share an Amazon Bedrock Studio project](#).

## Are you a first-time Amazon Bedrock Studio user?

If you're a first-time user of Bedrock Studio, we recommend that you read the following sections in order:

1. [Explore Amazon Bedrock Studio](#) – In this section, you access your Amazon Bedrock Studio workspace for the first time and use Explore mode to experiment with an Amazon Bedrock model.
2. [Build a chat app with Amazon Bedrock Studio](#) – In this section, you use Build mode to create a project in your workspace and create a simple Amazon Bedrock Studio app.
3. [Build a Prompt Flows app with Amazon Bedrock Studio](#) – In this section, you learn how to create a Prompt Flows app.

# Explore Amazon Bedrock Studio

Amazon Bedrock Studio is a web app that lets you easily prototype apps that use Amazon Bedrock models and features.

Amazon Bedrock Studio provides an *Explore* mode that lets you use a *playground* to easily experiment with a model. In the playground you can chat with a model by sending [prompts](#) to the model and viewing the responses. After experimenting with the playground, you can then use *build* mode to create a [chat app](#) or a [Prompt Flows app](#).

At any time, you can send us feedback about your experience with Amazon Bedrock by choosing the feedback button at the top of the page.

To get started with Amazon Bedrock Studio, you need an email invitation to an Amazon Bedrock Studio workspace. If you haven't received the invitation email from your organization, contact your organization's administrator.

Your organization's administrator determines the Amazon Bedrock Studio features that you have access to. If you need access to a model or feature that you don't currently have access to, contact your organization's administrator.

## Topics

- [What is a prompt?](#)
- [Use Explore mode to chat with a model](#)

## What is a prompt?

A prompt is input you send to a model in order for it to generate a response, in a process known as inference. For example, you could send the following prompt to a model.

### What is Avebury stone circle?

When you run the prompt in the playground, the playground shows a response from the model that is similar to the following.

```
Avebury stone circle is a Neolithic monument located in Wiltshire, England.  
It consists of a massive circular bank and ditch, with a large outer circle of standing  
stones
```

```
that originally numbered around 100.
```

The Explore mode playground provides quick start prompts that illustrate the kinds of prompt that you can send to a model.

The Explore playground also lets you try *multimodal* prompts, which are prompts that include images and text. This allows you to pass an image to a model and ask questions such as *What's in this image?*. Not all models support multimodal prompts.

The actual response that you get for a prompt depends on the model you use.

With Amazon Bedrock models you can use [inference parameters](#) to influence the response from a model. For example, you can use the `temperature` inference parameter to filter out lower probability responses. To experiment with inference parameters, or use an Amazon Bedrock component, such as a guardrail, you need to create a Bedrock Studio app. You can create a [chat app](#) or a [Prompt Flows](#) app. If you create a Prompt Flows app, you can also create prompts that you can customize for different use cases. For more information, see [Use prompts with your Flows app](#).

## Prompt engineering guides

Amazon Bedrock Studio provides models from a variety of model providers. Each provider provides guidance on how to best create prompt for their models.

- **Anthropic Claude model prompt guide:** <https://docs.anthropic.com/claude/docs>
- **Anthropic Claude prompt engineering resources:** <https://docs.anthropic.com/claude/docs/guide-to-anthropics-prompt-engineering-resources>
- **Cohere prompt guide:** <https://txt.cohere.com/how-to-train-your-pet-llm-prompt-engineering>
- **AI21 Labs Jurassic model prompt guide:** <https://docs.ai21.com/docs/prompt-engineering>
- **Meta Llama 2 prompt guide:** <https://ai.meta.com/llama/get-started/#prompting>
- **Stability documentation:** <https://platform.stability.ai/docs/getting-started>
- **Mistral AI prompt guide:** [https://docs.mistral.ai/guides/prompting\\_capabilities/](https://docs.mistral.ai/guides/prompting_capabilities/)

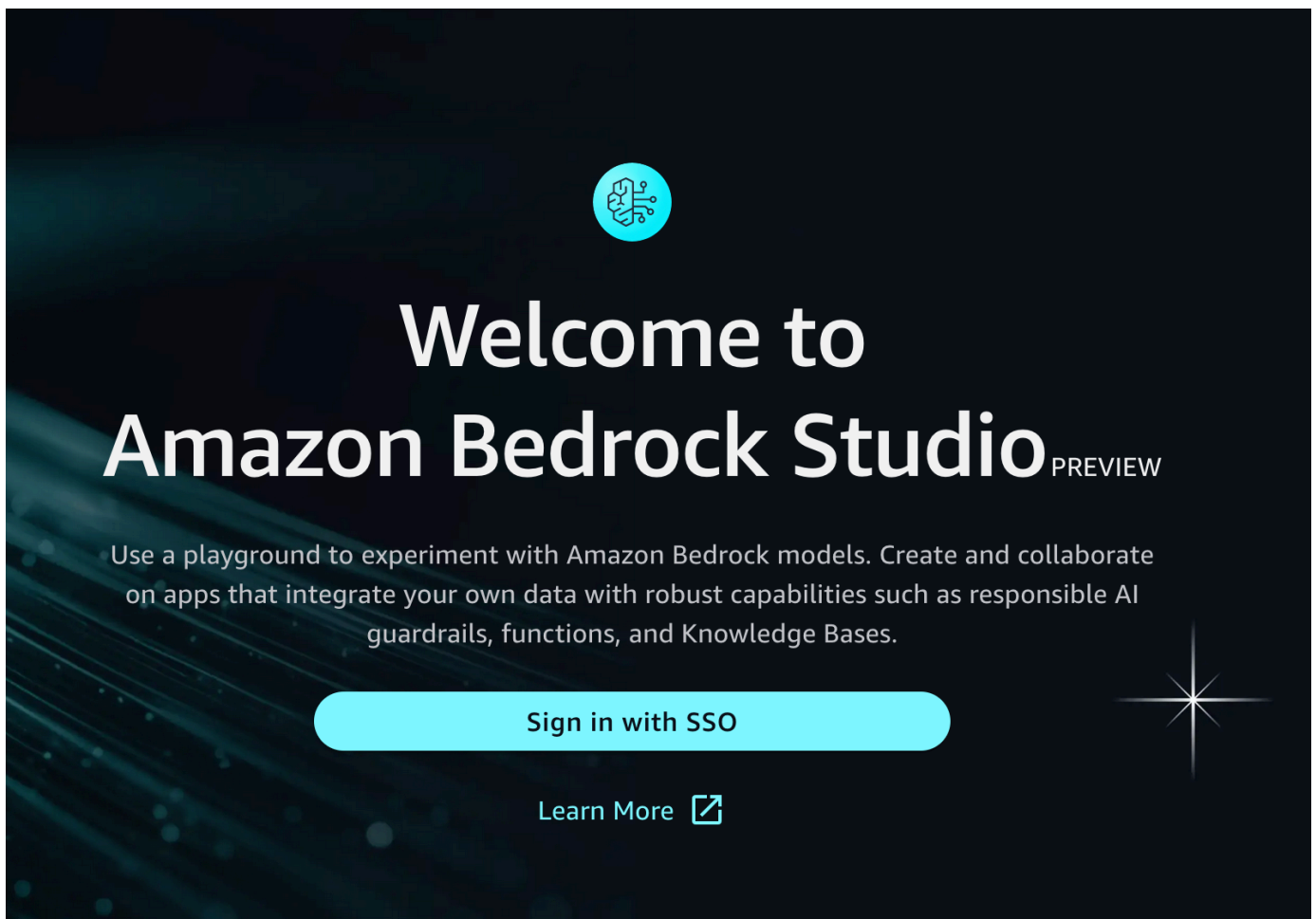
For general guidelines about creating prompts with Amazon Bedrock, see [General guidelines for Amazon Bedrock LLM users](#).

## Use Explore mode to chat with a model

These instructions show you how to use the playground in Explore mode to chat with a model. You chat with a model by sending a prompt to the model and by receiving the response that the model generates. You continue the chat by sending further prompts.

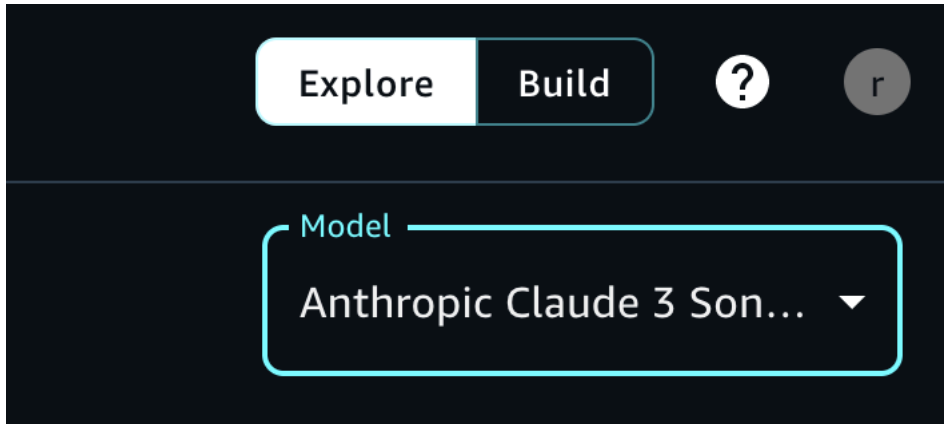
### To chat with a model

1. Open the invitation email sent to you by your organization.
2. In the email, choose the link to the workspace that you are invited to.
3. Choose **Sign in with SSO** to open your Amazon Bedrock Studio workspace.

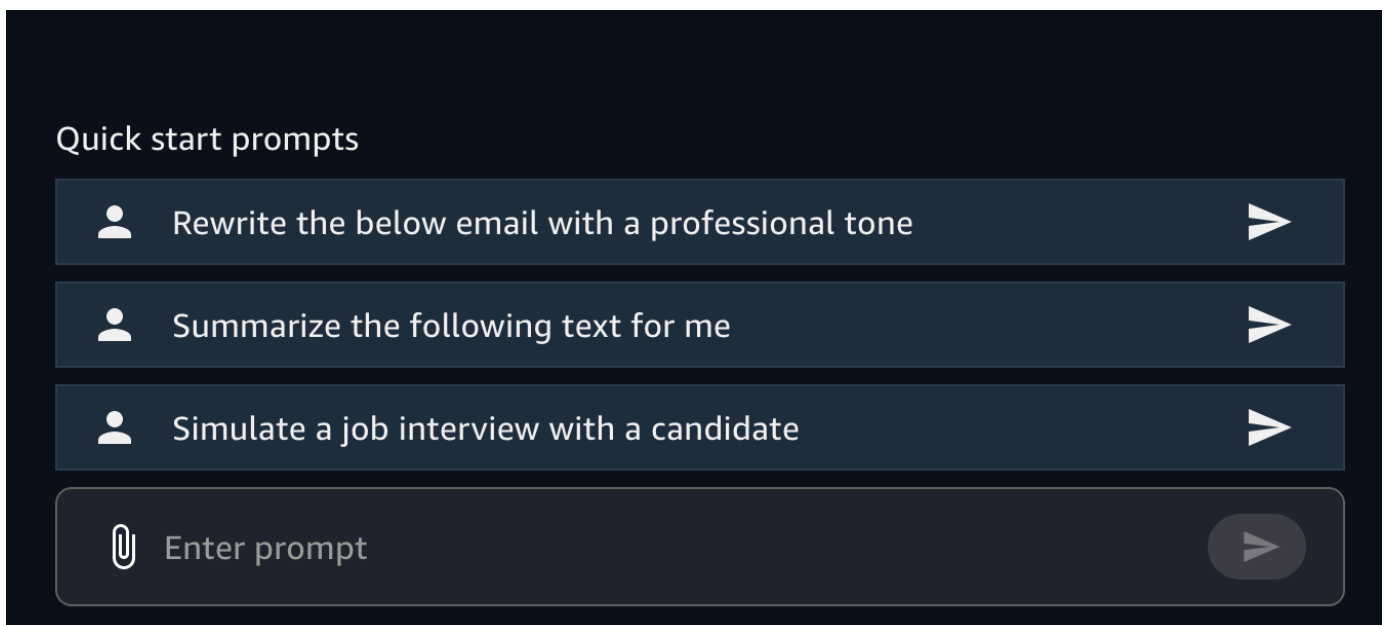


4. Enter your the user name that you use to access your organization's resources and choose **Next**.
5. Enter your password and choose **Sign in**.

- In the welcome to Amazon Bedrock Studio dialog box, choose **Start exploring** to start using the playground. The dialog box isn't shown if you have previously used the workspace.
- Make sure that you are in the Explore mode, by choosing **Explore** at the top right of the page.



- For **Model** select a model to use, or use the default model.
- In the **Quick start prompts** section at the bottom of the page, choose one of prompts.



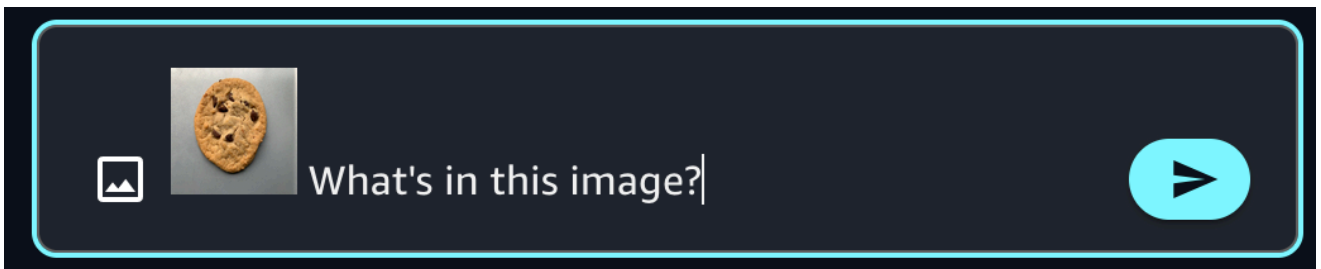
The playground sends the prompt to the model and shows the text that the model generated in response. Continue the chat until you are ready for the next step.

- Choose **Reset** to clear the playground.
- In the **Enter prompt** text box, enter **What is Avebury stone circle?**
- Press Ctrl + Enter on your keyboard to send the prompt to the model. Amazon Bedrock Studio shows the response from the model in the playground.

13. Continue the chat by entering the prompt **Is there a museum there?** and pressing Ctrl + Enter.

The response shows how the model uses the previous prompt as context for generating its next response.

14. Some models are multimodal in that that they can process text and images. To try analyzing an image, do the following.
  - a. In **Model** choose **Anthropic Claude 3 Sonnet**.
  - b. Choose the image button at the left of the **Enter prompt** text box.
  - c. In the open file dialog box, choose an image from your local computer.
  - d. In the text box, next to the image that you uploaded, enter **What's in this image?**



- e. Press Ctrl + Enter on your keyboard enter to send the prompt to the mode. The playground will show the image and underneath the image show a description of the image.
15. (Optional) Try using another model and different prompts. Different models have different recommendations for creating, or engineering, prompts. For more information, see [Prompt engineering guides](#).

Now that you are familiar with the explorer playground, try creating a Bedrock Studio app next. For more information, see [Build a chat app with Amazon Bedrock Studio](#).

# Build a chat app with Amazon Bedrock Studio

An Amazon Bedrock Studio chat app allows users to communicate with an Amazon Bedrock model through a conversational interface, typically by sending text messages and receiving responses. The model analyzes the user's input, formulate an appropriate response, and carries on a dialogue with the user. You can use a chat apps for various purposes, such as providing customer service, answering questions, offering recommendations, or engaging in open-ended conversations on a wide range of topics.

In this section, you learn how create a simple Amazon Bedrock Studio chat app in Build mode that creates playlists for a radio station. You can also create a [Prompt Flows](#) app. Later, you add the following features.

- A guardrail to prevent songs with inappropriate song titles.
- A data source that lets the app create playlists using your unique song information.
- A function that gets today's top 10 songs.

If you want to quickly experiment with sending prompts to a model, consider user the Explore view. For more information, see [Explore Amazon Bedrock Studio](#).

For more information about managing your projects and apps, see [Organize your work with projects in Amazon Bedrock Studio](#).

## Topics

- [Create an initial chat app](#)
- [Add a guardrail to your chat app](#)
- [Add a document data source to your chat app](#)
- [Add a function call to your chat app](#)

## Create an initial chat app

In this step you create a chat app that generates playlists for a radio station.

In the app, you use a system prompt to specify that the model should behave as an app that creates playlists for a radio station that plays rock and pop music. A system prompt is a type of

prompt that provides instructions or context to the model about the task it should perform, or the persona it should adopt during the conversation.

The user can then use the app to create playlists based on different themes, such as songs that are related by shared artists.

The instructions show how you can change the genre of music that the app creates playlists for by changing the system prompt.

In the app, you can experiment with the randomness and diversity of the response that the model returns by changing the [inference parameters](#).

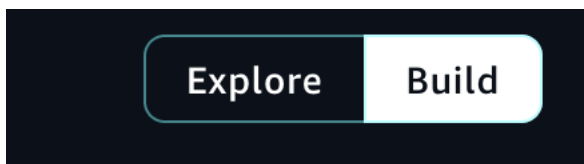
To create the app, you first need to create a *project*. A project can contain multiple apps and is also where you can add the Amazon Bedrock components that you want your apps to use. Later you will add guardrail, data source, and function calling components to your app. You can share a project with other users and groups of users. For more information, see [Share an Amazon Bedrock Studio project](#).

#### **Warning**

The name and description of a chat app are visible to all members of the Amazon Bedrock workspace. We recommend that you don't store sensitive data in these two fields. Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

### To create a chat app Bedrock Studio app

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.




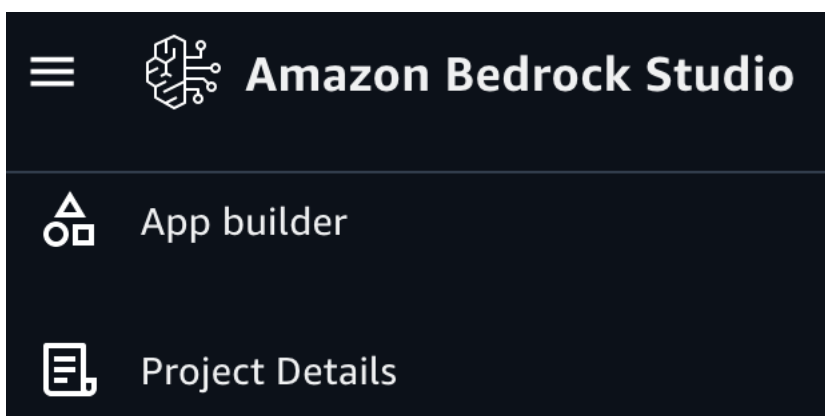
3. If this is the first time that you've opened the workspace, Amazon Bedrock Studio does the following:
  1. Creates a default project.
  2. Creates an empty chat app within the default project.



3. Creates an empty Prompt Flows app within the default project.
4. Displays a **Welcome to the build mode** dialog box with the following choices.
  - **Take a quick tour** – Gives you a quick overview of building apps with build mode.
  - **Explore app types** – Explore the different types of apps that you can create with Amazon Bedrock Studio.
  - **Build chat app** – create your first chat app.

For this procedure, choose **Build chat app**.

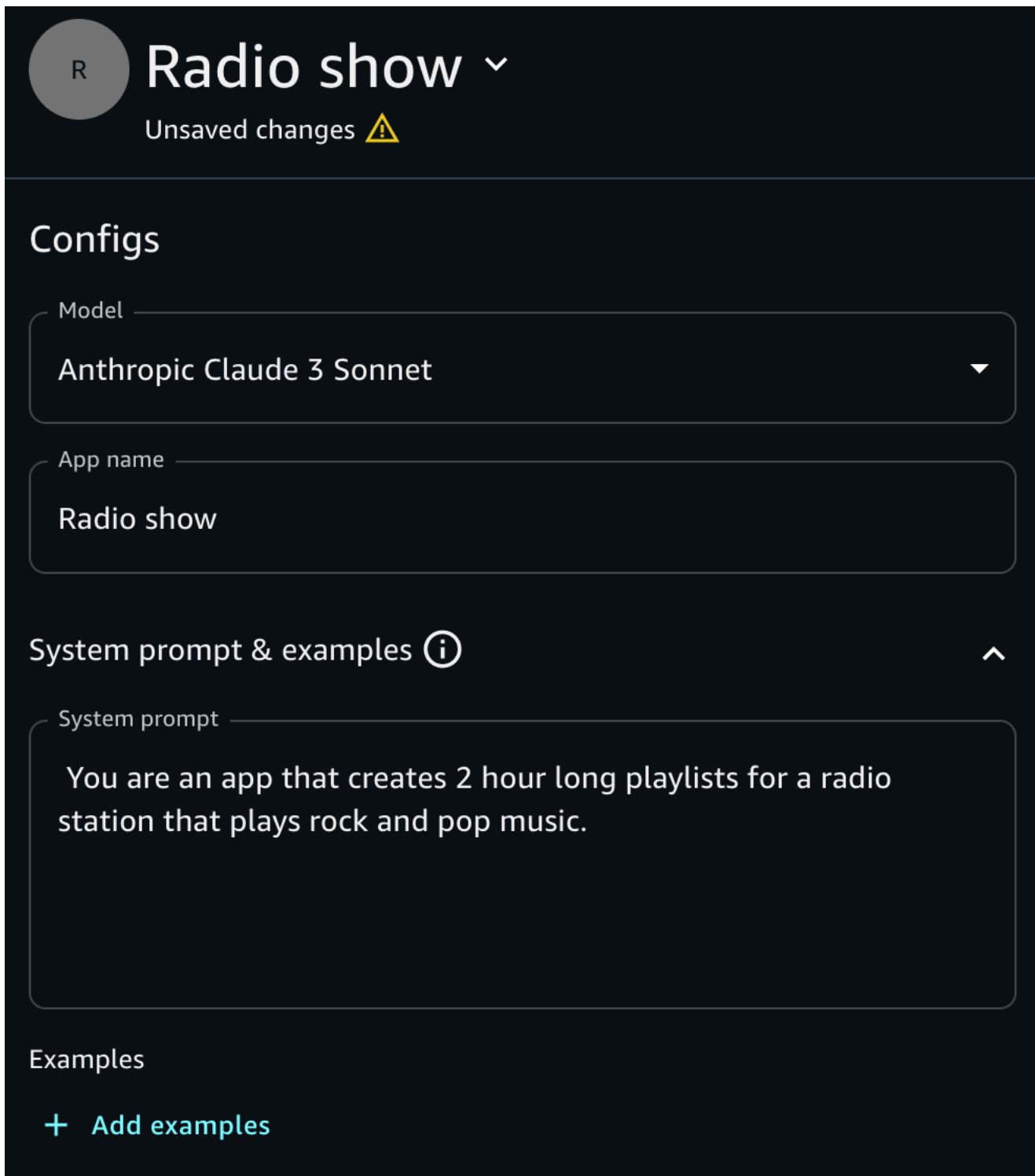
4. If you have previously open the workspace, do the following:
  - a. Create a new project by doing the following:
    - i. If you are not on the **All projects** page, choose **View all projects** in the **Projects** dropdown list that is at the top of the page.
    - ii. On the **All projects** page, choose **Create new project**.
    - iii. In the **Create new project** section, enter **Radio station project** as the name for your project in **Project name**.
    - iv. For **Project description**, enter a description for your project.
    - v. Choose **Create**. It might take a few minutes to create the project.
    - vi. In the card for your project, choose **Go to project** to open your project.
  - b. Open the navigation pane by choosing the menu  at the top left of the page.



- c. In **App builder** choose **Chat** to open the chat app that Bedrock Studio previously created for you.

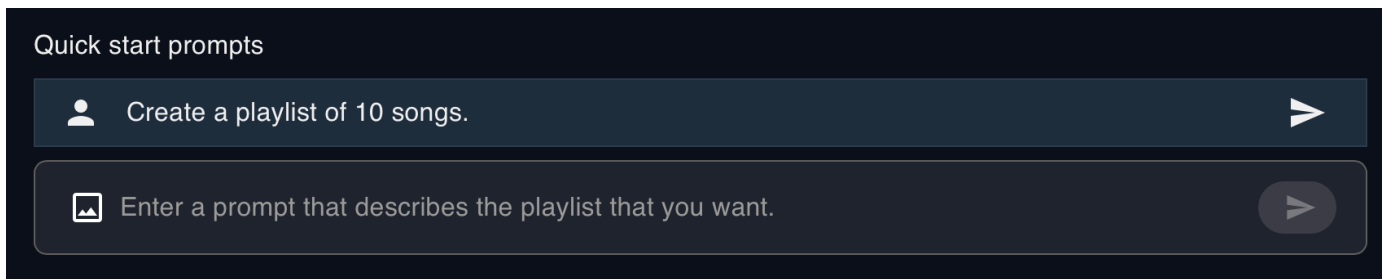
If you have previously used a different app, that app is shown. If you don't want to use that app, create a new app — On the project page, in the **Apps** section, choose **Create app** and then **Chat app**.

- d. If the **Welcome to build model** dialog is shown, choose **Build chat app**.
5. In the **Configs** pane, choose the model that you want your app to use in **Model**.
6. In **App name**, enter **Radio show** as the name for your app.
7. For **System prompt & examples**, enter **You are an app that creates 2 hour long playlists for a radio station that plays rock and pop music**.



8. In the **UI** section, update the user interface for the app by doing the following:
  - a. In **Hint text for empty chat** enter **Hi! I'm your radio show playlist creator.**
  - b. In **Hint text for user input** enter **Enter a prompt that describes the playlist that you want.**

- c. In **Quick start prompts** choose **Edit**.
  - d. Choose **Reset** to clear the list of quick start prompts
  - e. For **Quick-start prompt 1**, enter **Create a playlist of 10 songs..**
  - f. (Optional). Enter quick start prompts of your choice in the remaining quick start prompt text boxes.
9. Choose **Save** to save your app.
  10. In the **Quick start prompts** section of the **Preview** pane, run the quick start prompt that you just created by choosing the prompt.



The app shows the prompt and the response from the model in the **Preview** pane.

11. In the prompt text box (the text should read **Enter a prompt that describes the playlist that you want**), enter **Create a list of 10 songs where each song on the list is related to the next song, by musician, bands, or other connections. Be sure to explain the connection from one song to the next. Don't repeat any artists..**
12. Choose the run button (or press Ctrl + Enter on your keyboard) to send the prompt to the model.
13. Experiment with influencing the model response by changing the inference parameters in the **Parameters** section. For example, increase the randomness of the songs in the playlist by increasing the **Temperature** inference parameter.

The inference parameters you can change are *Temperature*, *Top P*, and *Top K*. Not all models support each of these inference parameters. For more information, see [Influence model responses with inference parameters](#).

14. Choose **Reset** to clear the conversation.
15. For **System prompt & examples**, enter **You are an app that creates 3 hour long playlists for a radio station that plays ambient music..** Doing this changes the system prompt to create a playlist of songs for a different genre of music and show length.

16. In the prompt text box, enter **Create a list of 10 songs where each song on the list is related to the next song, by musician, bands, or other connections. Be sure to explain the connection from one song to the next. Don't repeat any artists..**
17. Choose the run button to send the prompt to the model. Note the response from the model now includes songs from the ambient genre of music.
18. Choose **Save** to save your app.
19. (Optional) Share your project with other by following the instructions at [Share an Amazon Bedrock Studio project](#).
20. Next step: Add a guardrail to your app by following the instructions at [Add a guardrail to your chat app](#).

## Add a guardrail to your chat app

Guardrails for Amazon Bedrock lets you implement safeguards for your Amazon Bedrock Studio app based on your use cases and responsible AI policies. You can create multiple guardrails tailored to different use cases and apply them across multiple foundation models, providing a consistent user experience and standardizing safety controls across generative AI apps. You can configure denied topics to disallow undesirable topics and content filters to block harmful content in the prompts you send to a model and to the responses you get from a model. You can use guardrails with text-only foundation models. For more information, see [Safeguard your app with a guardrail](#).

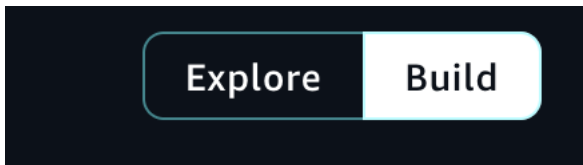
### Add a guardrail

This procedure shows you how to use a guardrail to safeguard the app you created in [Create an initial chat app](#). The guardrail prevents inappropriate language in song titles and filters out unwanted music genres.

If you just completed [Create an initial chat app](#), Bedrock Studio should be open on the app page and you don't need to do steps 1-3.

#### To add a guardrail to an Amazon Bedrock Studio app

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.



3. Open the app that you created in [Create an initial chat app](#).
4. In the **Configs** pane, choose **Guardrails** and then **Create new guardrail**.
5. For **Guardrail name**, enter **prevent\_unwanted\_songs**.
6. For **Guardrail description**, enter **Prevents inappropriate songs..**
7. In **Content filters** make sure **Enable content filters** is selected. For more information, see [Content filters](#).
8. In **Filter for prompts** make sure the filter for each category is set to **High**.
9. Make sure **Apply the same filters for responses** is selected.
10. In **Blocked messaging** do the following.
  - a. For **Blocked messaging for prompts**, enter **Sorry, your prompt contained inappropriate text..**
  - b. Clear **Apply the same message for blocked responses**.
  - c. For **Blocked messaging for responses**, enter **Sorry, but I can't respond with information that contains inappropriate text..**
11. Choose **Create** to create the guardrail.
12. In the **Configs** pane, make sure that the selected guardrail in **Guardrails** is the guardrail you just created.
13. Test the guardrail by entering **Create a list of 10 songs where each song has a swear word in the title**. In the prompt edit box.
14. Choose the run button to send the prompt to the model. The model should respond with the message **Sorry, but I can't respond with information that contains inappropriate text**.
15. Use a denied topic filter to prevent requests for music from a specific music genere. For information about denied topics, see [Denied topics](#).

To add the filter, do the following.

- a. In the **Guardrails** section of the **Configs** pane, select the guardrail and choose **Preview**.
- b. Choose **Edit** to edit the guardrail.
- c. In **Denied topics**, choose **Add topic**.

- d. For **Name**, enter **heavy metal**.
  - e. For **Definition for topic**, enter **Avoid mentioning songs that are from the heavy metal genre of music..**
  - f. For in **Sample phrases - optional**, enter **Create a playlist of heavy metal songs**.
  - g. Choose **Add phrase**.
  - h. Choose **Save**.
  - i. On the **Edit guardrail** page, choose **Update** to update the guardrail.
  - j. Test the guardrail by entering **Create a list of heavy metal songs** in the prompt edit box.
  - k. Choose the run button to send the prompt to the model. The model should respond with the message **Sorry, but I can't respond with information that contains inappropriate text**.
16. Next step: Add a data source to your app by following the instructions at [Add a document data source to your chat app](#).

## Add a document data source to your chat app

You can use your own data into your application by adding a data source to your app. Doing this allows your app to access to information that is only available to you. When your app passes a query to a data source, Amazon Bedrock Studio generates a response that include the query results from the data source. A data source can be a document or an Amazon Bedrock Knowledge Base. For more information, see [Add a data source to your app](#).

In this topic, you update the app you created in [Create an initial chat app](#) to use a CSV file as a document data source. The CSV file includes information about bands that don't have public meta data such song length, or music genre. The user can use the app to create a playlist for a radio show based on criteria such as song length or music genre.

### To add your own data to an Amazon Bedrock Studio app

1. Create a CSV file name *songs.csv* and fill with the following fictitious CSV data.

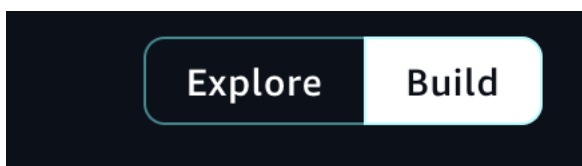
```
song,artist,genre,length-seconds
"Celestial Odyssey","Starry Renegades","Cosmic Rock",240
"Neon Rapture","Synthwave Siren","Synthwave Pop",300
```

```

"Wordsmith Warriors","Lyrical Legions","Lyrical Flow",180
"Nebula Shredders","Galactic Axemen","Cosmic Rock",270
"Electro Euphoria","Neon Nomads","Synthwave Pop",210
"Rhythm Renegades","Percussive Pioneers","Lyrical Flow",240
"Stardust Rift","Cosmic Crusaders","Cosmic Rock",180
"Synthwave Serenade","Electro Enchanters","Synthwave Pop",300
"Lyrical Legends","Rhyme Royale","Lyrical Flow",240
"Supernova Shredders","Amplified Ascension","Cosmic Rock",300
"Celestial Chords","Ethereal Echoes","Cosmic Rock",240
"Neon Nirvana","Synthwave Sirens","Synthwave Pop",270
"Verbal Virtuoso","Lyrical Maestros","Lyrical Flow",210
"Cosmic Collision","Stellar Insurgents","Cosmic Rock",180
"Pop Paradox","Melodic Mavericks","Synthwave Pop",240
"Flow Fusion","Verbal Virtuosos","Lyrical Flow",300
"Shredding Shadows","Crimson Crusaders","Cosmic Rock",270
"Synth Serenade","Electro Enchanters","Synthwave Pop",180
"Wordsmith Warlords","Lyrical Legionnaires","Lyrical Flow",240
"Sonic Supernova","Amplified Ascension","Cosmic Rock",210
"Celestial Symphony","Ethereal Ensemble","Cosmic Rock",300
"Electro Euphoria","Neon Nomads","Synthwave Pop",180
"Lyrical Legends","Rhyme Royale","Lyrical Flow",270
"Crimson Crescendo","Scarlet Serenaders","Cosmic Rock",240
"Euphoric Tides","Melodic Mystics","Synthwave Pop",210
"Rhythm Renegades","Percussive Pioneers","Lyrical Flow",180
"Cosmic Collision","Stellar Insurgents","Cosmic Rock",300
"Stardust Serenade","Celestial Crooners","Synthwave Pop",240
"Wordsmith Warriors","Lyrical Legions","Lyrical Flow",270
"Sonic Supernova III","Amplified Ascension","Cosmic Rock",180

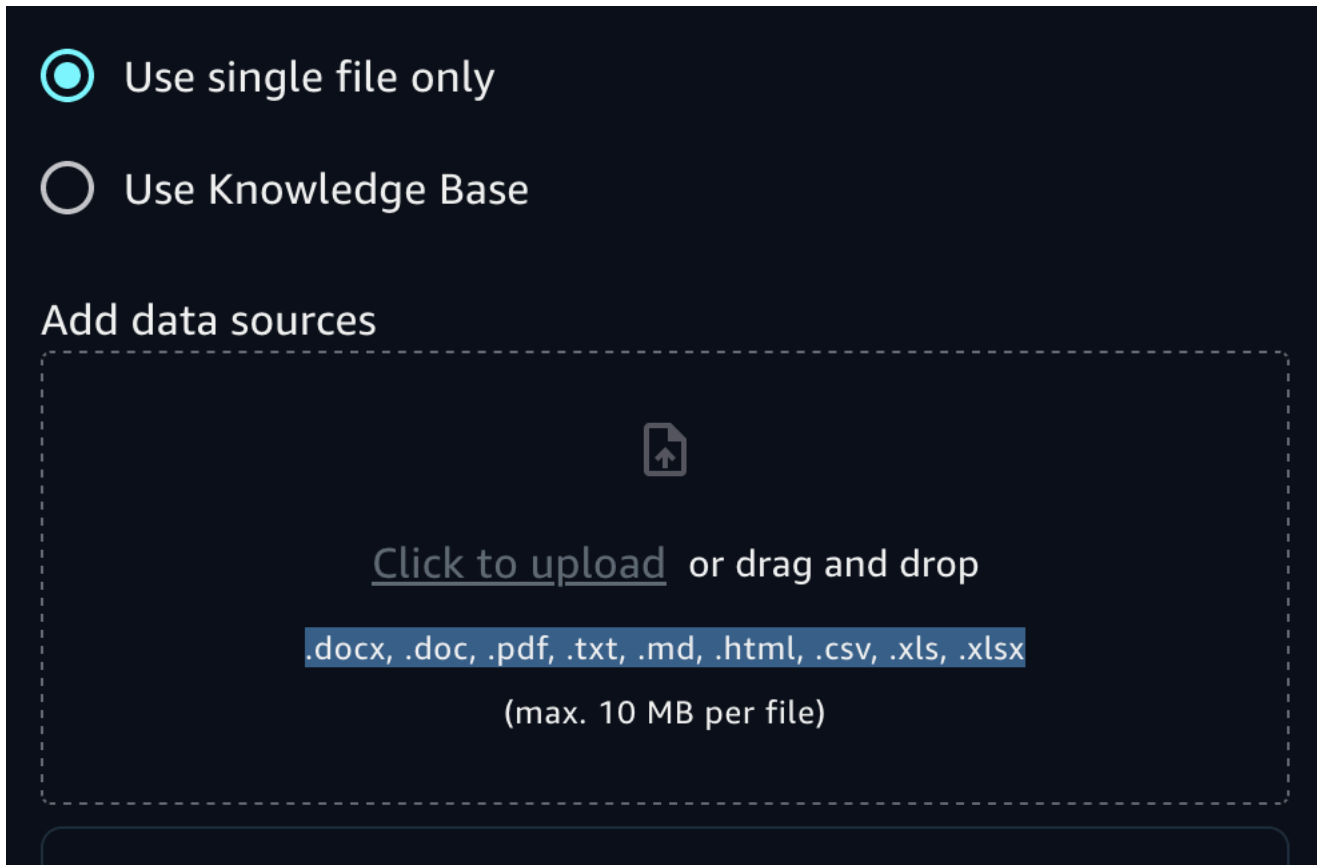
```

2. Open your Amazon Bedrock Studio workspace.
3. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.



4. Open the app that you created in [Create an initial chat app](#).
5. Disable any guardrails that you have enabled by choosing **None selected** in **Guardrails**.
6. For **Data**, choose **Use single file only**.
7. For **Add data sources**, choose **Click to upload** and upload the CSV file you created in step 1. Alternatively, add your source documents by dragging and dropping the CSV from your computer.





8. Test the data source by entering **Create a playlist of songs in the Lyrical Flow genre** in the prompt text box.
9. Choose the run button to send the prompt to the model. The model should respond with a playlist of songs from the Lyrical Flow genre that the CSV file contains.

## Add a function call to your chat app

Amazon Bedrock Studio functions let a model include information that it has no previous knowledge of in its response. For example, you can use a function to include dynamic information in a model's response such as a weather forecast, sports results, or traffic conditions. To use a function in Amazon Bedrock Studio you add a function component to your app. For more information, see [Call functions from your app](#).

In Amazon Bedrock Studio, a function calls an API hosted outside of Amazon Bedrock Studio. You either create the API yourself, or use an existing API. To create an API, you can use [Amazon API Gateway](#).

In this procedure, you add a function to the app that you created in [Create an initial chat app](#), so that users can get a list of the top 10 songs played on the radio station that day.

## To add a function to an Amazon Bedrock Studio app

1. Create a HTTPS server that implements a TopSongs function. Make sure the function adheres to the following schema.

```
openapi: 3.0.0
info:
  title: Top Songs API
  description: API to retrieve the top 10 songs played today
  version: 1.0.0

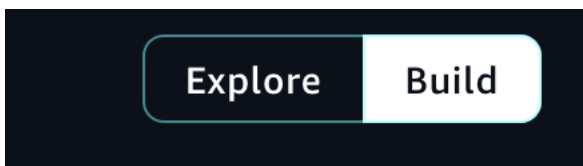
paths:
  /top-songs:
    get:
      summary: Get the top 10 songs played today
      description: >
        This endpoint returns an array of the top 10 songs played today,
        ordered by popularity. The first element in the array (index 0)
        represents the most popular song, and the last element (index 9)
        represents the 10th most popular song.
      responses:
        '200':
          description: Successful response
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/TopSongs'

components:
  schemas:
    TopSongs:
      type: array
      items:
        $ref: '#/components/schemas/Song'
      description: >
        An array containing the top 10 songs played today. The first element
        (index 0) is the most popular song, and the last element (index 9)
        is the 10th most popular song.
      example:
        - title: 'Song Title 1'
```

```
artist: 'Artist Name 1'  
album: 'Album Name 1'  
- title: 'Song Title 2'  
  artist: 'Artist Name 2'  
  album: 'Album Name 2'  
# ... up to 10 songs
```

```
Song:  
  type: object  
  properties:  
    title:  
      type: string  
      description: The title of the song  
    artist:  
      type: string  
      description: The name of the artist or band  
    album:  
      type: string  
      description: The name of the album the song is from  
  required:  
    - title  
    - artist  
    - album
```

2. Open your Amazon Bedrock Studio workspace.
3. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.



4. Open the app that you created in [Create an initial chat app](#).
5. In **Models**, choose a model that supports functions, such as *Anthropic Claude 2.1*.
6. In *Functions*, choose **Create new function**.

Alternatively, if you want to use a function that you have previously [created](#), choose the function in the **FUNCTIONS** dropdown list. You don't need to do the next step.

7. In the **Create function** pane, do the following.
  - a. For in **Function name**, enter **Top\_ten\_songs\_today**.
  - b. For **Function description (optional)**, enter **Today's top 10 songs..**

- c. For **Function schema**, enter the OpenAPI schema from step one.
  - d. Choose **Validate schema** to validate the schema.
  - e. In **Authentication method** choose the authentication method for your HTTP server. For more information, see [the section called "Authentication methods"](#).
  - f. In **API servers**, enter the URL for your server in **Server URL**. This value is autopopulated if the server URL is in the schema.
  - g. Choose **Create** to create your function.
8. For **System prompt and examples**, update the system prompt so that it describes the function. Use the following text: **You are an app that creates 2 hour long playlists for a radio station that plays rock and pop music. The function Top\_ten\_songs\_today gets the most popular song played on the radio station..**
  9. Test the function by doing the following.
    - a. Enter **What are today's top 10 songs?** in the prompt edit box.
    - b. Choose the run button to send the prompt to the model. The model should respond with the list of today's top 10 songs.

# Export an Amazon Bedrock Studio chat app

With Amazon Bedrock Studio, you can export the files for a chat app so that you can use the app outside of the Bedrock Studio web app.

When you export a chat app, Amazon Bedrock Studio exports a zip file with the AWS CloudFormation templates and other files required by your app. To use your app, you need to deploy the AWS CloudFormation templates to an AWS account. The actual contents of the zip file vary on the Amazon Bedrock Studio components that your chat app uses. After uncompressing the zip file, you deploy the contents of the zip file into your AWS account (or another AWS account, if you prefer).

A chat app can include one or more different types of Amazon Bedrock Studio components, such as a Guardrail or a Knowledge Base. When you deploy your app's components, Amazon Bedrock Studio only deploys the AWS infrastructure files. The data source files for a Knowledge Base and the secrets for a function aren't exported, and you have to configure them during the deployment. After deploying the app to an AWS account, you can run the app as a Node.js app.

## Topics

- [Export your Amazon Bedrock Studio chat app](#)
- [Deploy an Amazon Bedrock Studio chat app](#)
- [Run a deployed chat app](#)
- [Map a chat app to Amazon Bedrock API operations](#)

## Export your Amazon Bedrock Studio chat app

Use the following procedure to export a chat app to a zip file. You can then use the chat app outside of the Amazon Bedrock Studio web app.

Depending on the composition of your app, the zip package contains some or all of the following files:

- **README.md** — Instructions for deploying and running your app.
- **function-stack-\*.json** — AWS CloudFormation template that creates your function component, if any. This includes:

- An AWS Lambda [function](#) for calling the API defined in your OpenAPI schema.
- An AWS Secret Manager secret for storing credentials to use when calling your API. This secret contains an empty value, and you are expected to update this secret manually.
- **knowledge-base-stack-\*.json** — AWS CloudFormation template that creates your [Knowledge Base data source](#), if any. This includes an Knowledge Base for Amazon Bedrock configured with your selected data store and vector store. This knowledge base will not have the data you have uploaded in to Bedrock Studio, and you are expected to provide data files manually.
- **guardrails-stack-\*.json** — AWS CloudFormation template that creates a [guardrail](#) for Amazon Bedrock, if any.
- **agent-stack.json** — AWS CloudFormation template that creates an Amazon Bedrock Agent, if any.
- **invocation-policy-\*.json** — AWS CloudFormation template that creates an IAM policy with the runtime permissions that you need to talk to your deployed chat app.
- **br-studio-app-stack-\*.json** — Parent stack that orchestrates the deployment of all AWS CloudFormation stacks included in the zip package.
- **deploy-app.sh** — Helper script that you use to deploy your app infrastructure into your AWS account.
- **code-snippet.mjs** — Example code snippet that you embed in your code to invoke the app.
- **br-studio-app.mjs** — Standalone Node.js module to quickly test your deployed app.
- **aoss-encryption-policy-\*.json** — AOSS encryption policy necessary to use a Knowledge Base. This encryption policy is automatically created when your chat app contains an Amazon Bedrock Studio Knowledge Base.
- **provisioning-inline-policy.json** — An example of an AWS IAM policy that contains the permissions required to provision the chat app resources. The permissions declared in this policy file are needed when deploying the AWS CloudFormation stacks.

You can modify this policy to better suit your needs. You may create a new IAM principal with these policies, or attach these policies to an existing IAM principal in your AWS account.

- **kms-key-policy.json** — An example of an AWS KMS key policy that contains required permissions for encrypting your chat app resources.

You can modify this key policy to better suit your needs. You may create a new KMS key with this policy, or attach this policy to an existing KMS key in your AWS account.

- **api-schema\*.json** — OpenAPI schema files associated with your function components, if any.

### Important

- If you encounter access errors when exporting your chat app, contact your administrator. Workspaces created before the release of the Amazon Bedrock Studio app export feature may be using an outdated [permission boundary](#).
- Amazon Bedrock Studio needs to update permissions for projects that you created before the release of the app export feature. Amazon Bedrock Studio updates a project when you first try to export an app from the project. The update takes a couple of minutes. After the update completes, you can then export the app.
- Currently, you can't export apps that you created before June 15th, 2024. Support is coming soon.

## To export a chat app

1. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.
2. In the **Project** dropdown list, choose the project that contains the app that you want to export.
3. In the **Recent apps** section, choose the **See all apps** hyperlink.
4. On the apps pane, select the name of the chat app that you want to export.
5. If you haven't already, choose **Save** to save the app. You can't export an app unless you first save or test the app.
6. On the app page, choose **Export** to export the app. Amazon Bedrock Studio will create and download a zip file with the name **br-studio-app-export-\*.zip**.
7. Next step: [Deploy the chat app](#).

## Deploy an Amazon Bedrock Studio chat app

The following instructions show you the steps you take to deploy a chat app that you [export](#) from Amazon Bedrock Studio.

### Topics

- [Prerequisites for deploying a chat app](#)
- [Deploy the chat app](#)

## Prerequisites for deploying a chat app

Before you can deploy a chat app that you have exported, you must first do the following:

### To prepare for app deployment

1. Install the latest version of the AWS CLI on your local machine by following the instructions at [Install or update to the latest version of the AWS CLI](#).
2. Set up AWS credentials for the AWS CLI on your local machine by following the instructions at [Configure the AWS CLI](#). The credentials that the deployment script uses will follow the [order of precedence](#).
3. Create an IAM role that includes the policies from provisioning-inline-policy.json. For information about creating a role, see [IAM role creation](#).
4. (Optional) Using the AWS account that you set up in step 2, create an AWS KMS key for app export by following the instructions at [Creating keys](#). The key must be tagged with key EnableBedrock and a value of true. The key must also have a key policy that allows it to be used for encryption of your chat app resources. You may use the suggested policy declared in the kms-key-policy.json file of your zip package.
5. Create an Amazon S3 bucket to hold the app files that you export by following the instructions at [Creating a bucket](#).
6. If your app includes a Knowledge Base, copy the data source file to a folder named data/ in the Amazon S3 bucket that you created in step 5. If your app uses a document as a datasource, you supply a list of datasource files to the deployment script. For more information, see [Deploy the chat app](#).
7. If your app calls a function that requires authorization keys, provision the keys in AWS Secrets Manager by running the following command:

```
aws secretsmanager update-secret \  
  --secret-id br-studio/function-name-export-environment-id \  
  --secret-string 'secret-value'
```

Replace the following values:

- `function-name` — to the unique name of your function call component.
- `export-environment-id` — to the unique id of your function call component.
- `secret-value` — to the intended value to be used for authentication:



If the function requires API Keys, the syntax of secret-value should be: {"*key-name-1*": "*key-value-1*", "*key-name-2*": "*key-value-2*"}

If the function requires Basic authentication, the syntax of secret-value should be: {"\_\_AuthType\_\_": "BASIC", "username": "*username-value*", "password": "*password-value*"}

If the function requires Bearer token authentication, the syntax of secret-value should be: {"\_\_AuthType\_\_": "BEARER", "tokenValue": "*token-value*"}

8. Next step: [Deploy the chat app](#).

## Deploy the chat app

Before deploying your chat app, be sure to do the [prerequisite steps](#).

Deploying a chat app deploys the AWS infrastructure files that you need to run the app in AWS.

### To deploy a chat app

1. At the command prompt, do the following:
  - a. Navigate to the zip file that you exported from Amazon Bedrock Studio.
  - b. Assume the role of the AWS that you created in step 3 of [Prerequisites for deploying a chat app](#).
  - c. Run the following command:

```
./deployApp.sh \  
  [--awsRegion=value] \  
  [--s3BucketName=value] \  
  [--assetsS3Path=value] \  
  [--kmsKeyArn=value] \  
  [--dataFiles=value]
```

Replace the following values:

- *awsRegion* — with the AWS Region that you want to deploy the app to. Amazon Bedrock must be available in the Region you use. For more information, see [Supported AWS Regions](#).

- `s3BucketName` — With the Amazon S3 bucket that you created in step 5 of [Prerequisites for deploying a chat app](#). The deployment store the CFN templates and application data files in this bucket.
- `assetsS3Path` — (Optional) With the path in `s3BucketName` that you want deployment to store application files to.
- `kmsKeyArn` — (Optional) with the ARN of the KMS Key that you created in step 3 of [Prerequisites for deploying a chat app](#).
- `dataFiles` — With a comma-separated list of data source file paths. Required for apps that use a document data source.

For example, if you have a chat app with a single document as a data source, and you want to deploy the app with encryption, you can use the following command.

```
sudo sh ./deployApp.sh \  
  --awsRegion=us-east-1 \  
  --s3BucketName=my-s3-bucket-name-for-exported-chat-apps \  
  --assetsS3Path=my-prod-folder/my-chat-app \  
  --kmsKeyArn=arn:aws:kms:us-  
east-1:111122223333:key/11111111-2222-3333-4444-555555555555 \  
  --dataFiles=my-data-source.pdf
```

2. (Optional) Monitor the deployment in the AWS CloudFormation console.
3. Note the output from the script. You need it to run the chat app. It should be similar to: `node brStudioApp.mjs --question="Tell me about my documents"`.
4. Next step: [Run a deployed chat app](#).

## Run a deployed chat app

The following instructions show you the steps you take to run a deployed Amazon Bedrock Studio chat app.

### Topics

- [Prerequisites for running a chat app](#)
- [Run the chat app](#)

## Prerequisites for running a chat app

Before you can run a chat app that you have exported, you must first do the following:

### To prepare for running a chat app

1. Download and install Node.js. For more information, see [Download Node.js](#).
2. At the command prompt, install third-party Node.js libraries by running the following commands:

```
npm install minimist
npm install aws-sdk
npm install @aws-sdk/credential-providers
npm install @aws-sdk/client-bedrock-agent-runtime
npm install @aws-sdk/client-bedrock-runtime
```

3. Create or update an IAM role in which you want to run the app. For the policy, use the policy created by `deploy-app.sh` when you exported the app. The policy name is `BRStudioExportedAppInvocationRolePolicy-exportProjectId`. The policy is declared in `invocation-policy-*.json`. For more information, see [Creating roles](#).

## Run the chat app

To run your chat app, you need an IAM role with with permissions to invoke Amazon Bedrock resources. When you deploy the app, the AWS CloudFormation stack deployed through `deploy-app.sh` script provisions a suitable policy in your AWS account (declared in `invocation-policy-*.json`).

### To run the chat app

1. Switch to the IAM role that you created in step 3 of [Prerequisites for running a chat app](#).
2. Run the app by entering the command you noted in step 3 of [Deploy the chat app](#).

## Map a chat app to Amazon Bedrock API operations

A chat app that you export from Amazon Bedrock Studio works by making calls to various Amazon Bedrock operations. In general, an Amazon Bedrock Studio chat app uses the inference operations or the agent operations to send chat messages to a model. The components that an app uses,

such as a Knowledge Base, determines which operations are used. This section maps chat app component configurations to the underlying Amazon Bedrock Studio API operations.

When you [export](#) a chat app, you can see the operations that the app calls in the `code-snippet.mjs` source code file that the export creates.

## Topics

- [Chat app with no components](#)
- [Chat app with guardrail component](#)
- [Chat app with data source component](#)
- [Chat app with function component](#)
- [Chat app with function and data source component](#)

## Chat app with no components

A simple chat app without any components calls the [InvokeModelWithResponseStream](#) inference operation to send chat messages to the model and stream the response back. If the model doesn't support streaming, the app uses the [InvokeModel](#) operation instead. The operations both use the [inference parameters](#) that you configure for the app.

### Note

Amazon Bedrock also provides the [Converse API](#) as an alternative method for sending messages to a model.

For more information, see [Run model inference](#) in the *Amazon Bedrock guide*.

## Chat app with guardrail component

If a chat app uses a [guardrail component](#) (and no other components), it calls the [InvokeModelWithResponseStream](#) or [InvokeModel](#) inference operation (if the model doesn't support streaming). In the operation request, the app specifies the guardrail ID for the guardrail that safeguards the app. If the app also uses other components, other Amazon Bedrock operations might be used instead.

**Note**

If you use the Converse API, you can also specify a guardrail that the app uses.

For more information, see [Guardrails for Amazon Bedrock](#) in the *Amazon Bedrock guide*.

## Chat app with data source component

A chat app can have one of two types of [data source](#) – a document file or an Amazon Bedrock Knowledge Base. In both cases, the app uses [RetrieveAndGenerate](#) agents operation to send a message and document data source details to the model. The model then generates a response based on the information in the data source.

You can't use a guardrail with an Amazon Bedrock Studio chat app that uses a data source. In your own applications, you can configure a guardrail to safeguard responses for calls you make to `RetrieveAndGenerate`.

If you chat also uses a function component, see [Chat app with function and data source component](#).

For more information, see [Knowledge bases for Amazon Bedrock](#) in the *Amazon Bedrock guide*.

## Chat app with function component

You can set up a chat app to use a [function component](#) that calls an external API. In this scenario, the app uses an Amazon Bedrock agent to chat with the model. To send a message to the agent, the chat app uses the [InvokeAgent](#) agent operation. Based on the message content, the agent might use the function component to help meet the message requirements.

If you chat also uses a data source component, see [Chat app with function and data source component](#).

For more information, see [Agents for Amazon Bedrock](#) in the *Amazon Bedrock guide*.

## Chat app with function and data source component

If your app uses a function component and a data source component, it uses an Amazon Bedrock agent to chat with the model. To send a message to the agent, the chat app uses the [InvokeAgent](#)

agent operation. The app uses the data source and the function component to help meet the requirements of the message.

For more information, see [Agents for Amazon Bedrock](#) in the *Amazon Bedrock guide*.

# Build a Prompt Flows app with Amazon Bedrock Studio

A Prompt Flows app let you link prompts, supported foundational models (FMS), and other units of work, such as a Knowledge Base, together and create generative AI workflows for end-to-end solutions. For example, you could create Prompt Flows apps to do the following.

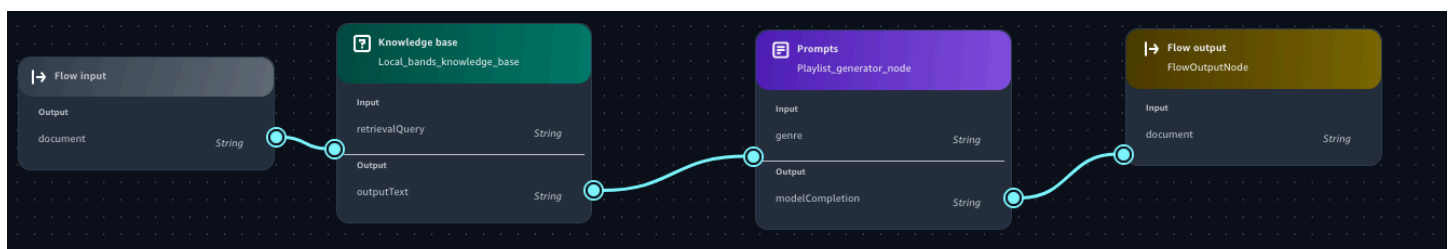
- **Create and send an email invite** – Create a flow connecting a prompt node, knowledge base node, and function node. Provide the following prompt to generate an email body: **Send invite to John Smith’s extended team for in-person documentation read for an hour at 2PM EST next Tuesday..** After processing the prompt, the flow queries a knowledge base to look up the email addresses of John Smith's extended team, and then sends the input to a function to send the invite to all the team members in the list.
- **Troubleshoot using the error message and the ID of the resource that is causing the error** – The flow looks up the possible causes of the error from a documentation knowledge base, pulls system logs and other relevant information about the resource, and updates the faulty configurations and values for the resource.

## ⚠ Important

If your workspace was created before Amazon Bedrock Studio added support for Prompt Flows apps, you won't be able to create a Prompt Flows app until your administrator updates your workspace. Ask your administrator to follow the instructions at [Update a workspace for Prompt management and Flows](#).

In this section you create a Prompt Flows app that generates a playlist of music from a Knowledge Base of songs by fictional local bands.

To create a Prompt Flows app, you use the *flow builder* which is a tool in Amazon Bedrock Studio to build and edit flow apps through a visual interface. You use the visual interface to drag and drop nodes onto the interface and configure inputs and outputs for these nodes to define your flow.



In your flow you can apply logical conditions to direct the output from a node to different destinations. You can then run the flow within Amazon Bedrock Studio and view the output.

The following list introduces you to the basic elements of a flow.

- **Prompt Flow** – A flow is a construct consisting of a name, description, permissions, a collection of nodes, and connections between nodes. When you run a flow, the input to the flow is sent through each node of the flow until the flow emits the final output from an output node.
- **Node** – A node is a step inside a flow. For each node, you configure its name, description, input, output, and any additional configurations. The configuration of a node differs based on its type.

For information about the types of nodes that Amazon Bedrock Studio supports, see [Prompt Flow nodes available in Amazon Bedrock Studio](#).

- **Connection** – There are two types of connections used in flow apps:
  - A **data connection** is drawn between the output of one node (the *source node*) and the input of another node (the *target node*) and sends data from an upstream node to a downstream node. In the flow builder, data connections are solid lines.
  - A **conditional connection** is drawn between a condition in a condition node and a downstream node and sends data from the node that precedes the condition node to a downstream node if the condition is fulfilled. In the flow builder, conditional connections are dotted lines.
- **Expressions** – An expression defines how to extract an input from the whole input entering a node. To learn how to write expressions, see [Define inputs with expressions](#).

## Topics

- [Create a flows app](#)
- [Prompt Flow nodes available in Amazon Bedrock Studio](#)
- [Define inputs with expressions](#)
- [Use prompts with your Flows app](#)

## Create a flows app

In this section you first build a Prompt Flow that generates a playlist of music from a Knowledge Base of songs by fictional local bands. Next, you use [Prompt Management](#) to add a prompt that lets you customize the playlist for different genres of music. Finally, you add a condition that



directs the flow to create a playlist of songs by local bands, if the requested genre is *Cosmic Rock*. For other genres, the flow generates a playlist with songs by well known bands.

## Topics

- [Create an initial Prompt Flow app](#)
- [Add a Knowledge Base to your Prompt Flows app](#)
- [Add a prompt to your Prompt Flows app](#)
- [Add a condition to your flows app](#)

## Create an initial Prompt Flow app

In this procedure you create an initial Prompt Flow app which has an [Flow input](#) node and a [Flow output](#) node.

### Important

If your workspace was created before Amazon Bedrock Studio added support for Prompt Flows apps, you won't be able to create a Prompt Flows app until your administrator updates your workspace. Ask your administrator to follow the instructions at [Update a workspace for Prompt management and Flows](#).

A flow contains only one flow input node which is where the flow begins. The Flow input node takes your input and passes it to the next node in a data type of your choice (String, Number, Boolean, Object and Array). In these procedures, the input to the flow is a String. To learn more about using different data types in a flow, see [Define inputs with expressions](#).

A flow output node extracts the input data from the previous node, based on the defined expression, and outputs the data. A flow can have multiple flow output nodes if there are multiple branches in the flow.

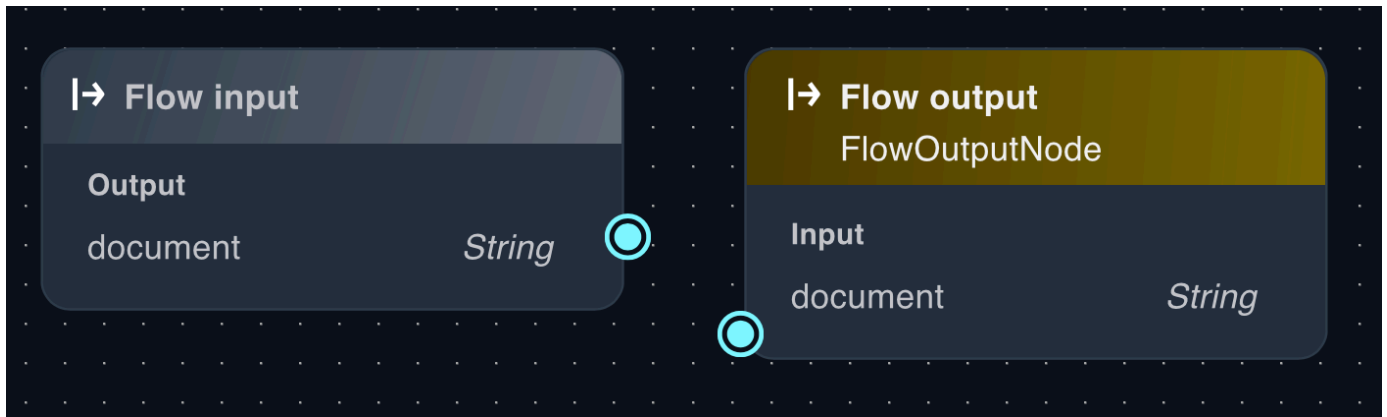
After completing the procedure, the flow app is empty, other than the Flow input and Flow output nodes. In the next step you add Knowledge Base as a data source and run the flow app for the first time.

**⚠ Warning**

The name and description of flows app are visible to all members of the Amazon Bedrock Studio workspace. We recommend that you don't store sensitive data in these two fields. Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

**To create an initial flows app**

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.
3. Open the project that you want to create your Prompt Flow flow app in, or create a new project by doing the following.
  - a. If you are not on the **All projects** page, choose **View all projects** in the **Projects** dropdown list box, at the top of the page.
  - b. On the **All projects** page, choose **Create new project**.
  - c. In the **Create new project** section, enter a name for your project in **Project name**.
  - d. Enter a description for your project in **Project description**.
  - e. Choose **Create**. It might take a few minutes to create the project.
  - f. In the card for your app, choose **Go to project** to open your project.
4. On the project page, in the **Apps** section, choose **Create app** and then **Prompt Flow app**.
5. In the flow builder, choose the flow name (**Untitled Prompt Flow-nnnn**) and enter **Local bands** as the name for the flow.
6. In the **Prompt Flow builder** pane, select the **Nodes** tab. The center pane displays a **Flow input** node and a **Flow output** node. These are the input and output nodes for your flow. The circles on the nodes are connection points. In the next procedure, you use the connection points to connect a Knowledge Base node to the Flow input node and the Flow output node.



- Next step: [Add a Knowledge Base to your Prompt Flows app.](#)

## Add a Knowledge Base to your Prompt Flows app

In this procedure, you add a [Knowledge Base](#) node as a data source to the flow that you created in [Create an initial Prompt Flow app](#). The Knowledge Base you add is Comma Separated Values (CSV) file containing a list of fictitious songs and artists. The list includes the duration (seconds) and genre of each song. For more information about Knowledge Bases, see [Add a data source to your app](#).

During the procedure, you make connections from the Flow input node to the Knowledge Base node and from the Knowledge Base node to the Flow output node. At some point, you might need to delete a node or remove a node connection. To delete a node, select the node that you want to delete and press the Delete button. To remove a connection, choose the connection that you want to delete and then press the delete button.

When you run the flow with the input **Create a playlist**, the app creates a playlist using songs only from the Knowledge Base.

### To create the flow with a Knowledge Base

- Create a CSV file name *songs.csv* and fill with the following fictitious CSV data. This is the data source for your Knowledge Base. Save the CSV file to your local computer.

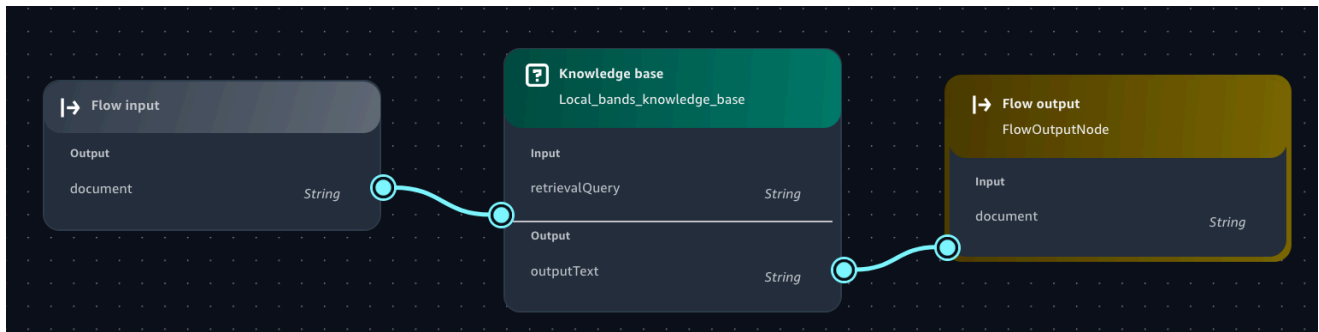
```
song,artist,genre,length-seconds
"Celestial Odyssey","Starry Renegades","Cosmic Rock",240
"Neon Rapture","Synthwave Siren","Synthwave Pop",300
"Wordsmith Warriors","Lyrical Legions","Lyrical Flow",180
"Nebula Shredders","Galactic Axemen","Cosmic Rock",270
```

```

"Electro Euphoria", "Neon Nomads", "Synthwave Pop", 210
"Rhythm Renegades", "Percussive Pioneers", "Lyrical Flow", 240
"Stardust Rift", "Cosmic Crusaders", "Cosmic Rock", 180
"Synthwave Serenade", "Electro Enchanters", "Synthwave Pop", 300
"Lyrical Legends", "Rhyme Royale", "Lyrical Flow", 240
"Supernova Shredders", "Amplified Ascension", "Cosmic Rock", 300
"Celestial Chords", "Ethereal Echoes", "Cosmic Rock", 240
"Neon Nirvana", "Synthwave Sirens", "Synthwave Pop", 270
"Verbal Virtuoso", "Lyrical Maestros", "Lyrical Flow", 210
"Cosmic Collision", "Stellar Insurgents", "Cosmic Rock", 180
"Pop Paradox", "Melodic Mavericks", "Synthwave Pop", 240
"Flow Fusion", "Verbal Virtuosos", "Lyrical Flow", 300
"Shredding Shadows", "Crimson Crusaders", "Cosmic Rock", 270
"Synth Serenade", "Electro Enchanters", "Synthwave Pop", 180
"Wordsmith Warlords", "Lyrical Legionnaires", "Lyrical Flow", 240
"Sonic Supernova", "Amplified Ascension", "Cosmic Rock", 210
"Celestial Symphony", "Ethereal Ensemble", "Cosmic Rock", 300
"Electro Euphoria", "Neon Nomads", "Synthwave Pop", 180
"Lyrical Legends", "Rhyme Royale", "Lyrical Flow", 270
"Crimson Crescendo", "Scarlet Serenaders", "Cosmic Rock", 240
"Euphoric Tides", "Melodic Mystics", "Synthwave Pop", 210
"Rhythm Renegades", "Percussive Pioneers", "Lyrical Flow", 180
"Cosmic Collision", "Stellar Insurgents", "Cosmic Rock", 300
"Stardust Serenade", "Celestial Crooners", "Synthwave Pop", 240
"Wordsmith Warriors", "Lyrical Legions", "Lyrical Flow", 270
"Sonic Supernova III", "Amplified Ascension", "Cosmic Rock", 180

```

2. Open the Prompt Flows app that you created in [Create an initial Prompt Flow app](#).
3. Add and configure a Knowledge Base node by doing the following:
  - a. In the **Prompt Flow builder** pane, select the **Nodes** tab.
  - b. From the **Data** section, drag a **Knowledge Base** node onto the flow builder canvas.
  - c. The circles on the nodes are connection points. Using your mouse, click on the circle for the **Flow input** node and draw a line to the circle on **Input** section of the Knowledge Base node that you just added.
  - d. Connect the **Output** of the Knowledge Base node in your flow with the **Input** of the **Flow output** node.



- e. Select the Knowledge Base node that you just added.
  - f. In the **Prompt Flow builder** pane, choose the **Configure** tab and do the following:
    - i. In **Node name** enter **Local\_bands\_knowledge\_base**.
    - ii. In **Knowledge Base Details**, choose **Create new Knowledge Base** to open the **Create Knowledge Base** pane.
    - iii. For **Knowledge Base name**, enter **Local\_bands**.
    - iv. For **Knowledge Base description**, enter **Songs by local bands. Includes song, artist, genre, and song length (in seconds)**.
    - v. In **Add data sources**, choose **Click to upload** and upload the CSV file (songs.csv) that you created in step 1. Alternatively, add your source documents by dragging and dropping the CSV from your computer.
    - vi. (Optional) In **Embeddings model** choose a different model, if available.
    - vii. (Optional) In **Vector store** choose a different vector store, if available.
    - viii. Choose **Create** to create the Knowledge Base. It might take a few minutes to create the Knowledge Base.
  - g. Back in the **Prompt Flow builder** pane, in **Select Knowledge Base**, select the Knowledge Base that you just created (Local\_bands).
  - h. In **Select response generation model**, select the model that you want the Knowledge Base to generate responses with.
4. Choose **Save** to save the app.
  5. Test your prompt by doing the following:
    - a. On right side of the app flow page, choose **<** to open the test pane.
    - b. Enter **Create a playlist** in the **Enter prompt** text box.
    - c. Press **Ctrl + Enter** on the keyboard or choose the run button to test the prompt.
    - d. If necessary, make changes to your flow. If you are satisfied with the flow, choose **Save**.

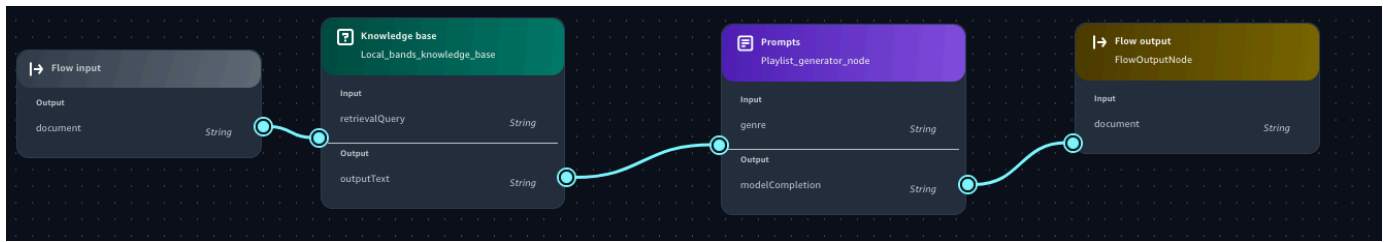
- Next step: [Add a prompt to your Prompt Flows app](#).

## Add a prompt to your Prompt Flows app

In this procedure you add a prompt to the flow by adding a [prompt node](#). The prompt allows you to easily choose which genre of songs should be included in the playlist that the flow generates. For more information, see [Use prompts with your Flows app](#).

### To add a prompt to the flow

- In the **Prompt Flow builder** pane, select **Nodes**.
- From the **Orchestration** section, drag a **Prompt** node onto the flow builder canvas.
- Select the node you just added.
- In the **Configurations** tab of the **Prompt Flow builder** pane, do the following:
  - In **Node name** enter **Playlist\_generator\_node**.
  - In **Prompt details** choose **Create new prompt** to open the **Create prompt** pane.
  - For **Prompt name** enter **Playlist\_generator\_prompt**.
  - For **Model**, choose the model that you want the prompt to use.
  - For **Prompt message** enter **Create a playlist of songs in the genre {{genre}}**.
  - (Optional) In **Model configs**, make changes to the inference parameters.
  - Choose **Save draft and create version** to create the prompt. It might take a couple of minutes to finish creating the prompt.
- In the flow builder, choose the prompt node that you just added.
- In the **Configure** tab, do the following in the **Prompt details** section:
  - In **Prompt** select the prompt that you just created.
  - In **Version** select the version (**1**) of the prompt to use.
- Update the flow paths by doing the following:
  - Delete the output from the **Knowledge Base** node that goes into the **Flow output**.
  - Connect the output from the **Knowledge Base** node to the input of the **Prompts** node.
  - Connect the output from the prompt node to the input of the **Flow output** node.
- Choose **Save** to save the flow. The flow should look similar to the following.



9. Test your prompt by doing the following:
  - a. On right side of the app flow page, choose < to open the test pane.
  - b. For **Enter prompt**, enter **Cosmic Rock**.
  - c. Press Ctrl + Enter on the keyboard or choose the run button to test the prompt. The response should be a playlist of songs in the Cosmic Rock genre.
  - d. Change the prompt to **Synthwave Pop** and run the prompt again. The songs should now be from the Synthwave Pop genre.
  - e. If necessary, make changes to your flow. If you are satisfied with the flow, choose **Save**.
10. Next step: [Add a condition to your flows app](#).

## Add a condition to your flows app

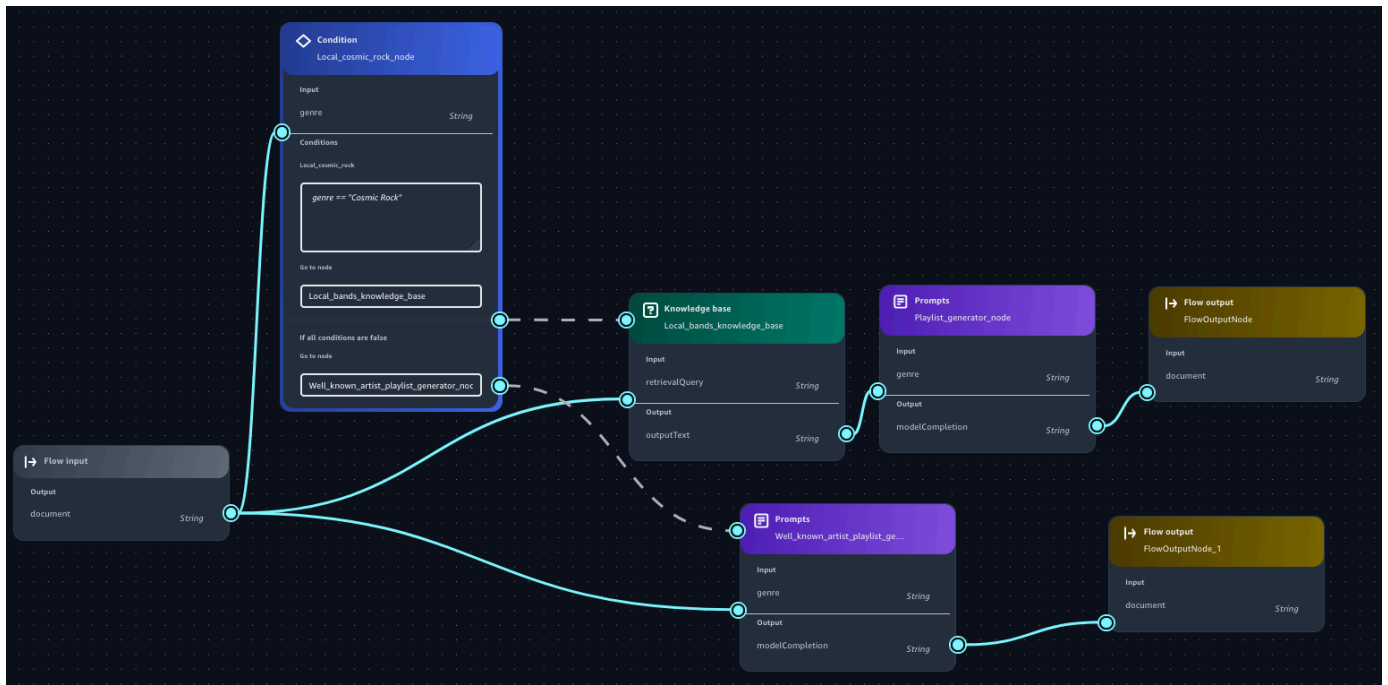
In this procedure, you add a [condition](#) node to the flow so that if you enter the prompt **Cosmic Rock**, the flow only generates a playlist from the local bands Knowledge Base. If you enter a different genre, the flows uses the playlist generator prompt to create a playlist of well known artists in that genre.

### To add a condition to the flow

1. In the **Prompt Flow builder** pane, choose **Nodes**.
2. From the **Logic** section, drag a **Condition** node onto the flow builder canvas.
3. Select the **Condition** node that you just added.
4. Add the flow that generates a playlist from local bands by doing the following:
  - a. In the **Inputs** section of the **Configurations** tab, change the **Node Name** to **Local\_cosmic\_rock\_node**.
  - b. In the **Inputs** section, change the **Name** to **genre**.
  - c. In the **Conditions** section, do the following:

- i. In **Name**, enter **Local\_cosmic\_rock**.
    - ii. In **Condition**, enter the condition `genre == "Cosmic Rock"`.
  - d. In the flow builder, choose the condition node that you just added.
  - e. Connect **Go to node** to the **Knowledge base** node.
  - f. Connect the **Output** of the **Flow input** node to the **Input** of the **Condition** node. Leave the existing connection to the **Knowledge Base** node as this ensures the prompt is passed to the Knowledge Base.
5. Add the flow that generates a playlist by well known bands by doing the following:
- a. In the **Prompt Flow builder** pane, select **Nodes**.
  - b. From the **Orchestration** section, drag a **Prompt** node onto the flow builder canvas.
  - c. Select the node you just added.
  - d. Choose the **Configurations** tab of the **prompt Flow builder** pane and do the following:
    - i. For **Node name**, enter **Well\_known\_artist\_playlist\_generator\_node**.
    - ii. In **Prompt details** section, choose the **Playlist\_generator\_prompt** prompt that you previously created.
    - iii. For **Version**, select the version (**1**) of the prompt to use.
    - iv. Connect the **Output** from the **Flow input** node to the **Input** of the prompt that you just created.
    - v. In the **Condition** node, connect the **If all conditions are false go to node** to the new prompt.
    - vi. In the **Flow builder** pane, select **Nodes**.
    - vii. From the **Other** section, drag a **Flow output** node onto the flow builder canvas.
    - viii. Connect the **Output** of the new **Prompt** to the input of the new **Flow output** node.
6. Choose **Save** to save the flow. The flow should look similar to the following.





7. Test your prompt by doing the following:
  - a. On right side of the app flow page, choose < to open the test pane.
  - b. In **Enter prompt**, enter **Cosmic Rock**.
  - c. Press Ctrl + Enter on the keyboard or choose the run button to test the prompt. The response should be a playlist of songs in the Cosmic Rock genre with bands that are only from the Knowledge Base.
  - d. Change the prompt to **Classic Rock** and run the prompt again. The songs should now be well known bands from the classic rock genre.

## Prompt Flow nodes available in Amazon Bedrock Studio

Amazon Bedrock Studio provides the following node types to build your Prompt Flow app. A node comprises of the following:

- Name – The name for the node.
- Type – the type of the node. For more information, see [Prompt Flow nodes available in Amazon Bedrock Studio](#).
- Inputs – Provide a name and data type for each input. Some nodes have pre-defined names or types that you must use. In the expression field, define the part of the whole input to use as the individual input. For more information, see [Define inputs with expressions](#).

In the flow builder, an input appears as a circle on the left edge of a node. Connect each input to an output of an upstream node.

- **Outputs** – Provide a name and data type for each output. Some nodes have pre-defined names or types that you must use. In the flow builder, an output appears as a circle on the right edge of a node. Connect each output to at least one input in a downstream node. If an output from a node is sent to more than one node, or if a condition node is included, the path of a flow will split into multiple branches. Each branch can potentially yield another output in the flow response.
- **Configuration** – You define node-specific fields at the top of the node.

### Note

Amazon Bedrock Studio supports a subset of the nodes that are available in Amazon Bedrock. For more information, see [Node types in flow](#).

## Nodes

- [Flow input](#)
- [Flow output](#)
- [Condition](#)
- [Prompt node](#)
- [Knowledge Base node](#)

## Flow input

Every flow contains only one flow input node and must begin with it. When you run the flow, the input is fed into this node and the configured output is passed to the next step.

## Configuration

- **Node name** – N/A
- **Input field name** – N/A
- **Input field types** – N/A
- **Input expression** – N/A

- **Output field name** – document
- **Output field types** – String, Number, Boolean, Object and Array.
- **Output expression** – No

## Flow output

A flow output node extracts the input data from the previous node, based on the defined expression, and returns it. A flow can have multiple flow output nodes if there are multiple branches in the flow.

- **Node name** – Any
- **Input field name** – document
- **Input field types** – String, Number, Boolean, Object and Array.
- **Input expression** – Yes
- **Output field name** – N/A
- **Output field types** – N/A
- **Output expression** – N/A

## Condition

A condition node sends data from the previous node to different nodes, depending on the conditions that are defined. A condition node can take multiple inputs.

- **Node name** – Any
- **Input field name** – Any
- **Input field types** – String, Number, Boolean, Object and Array.
- **Input expression** – Yes
- **Condition field name** – Any
- **Output field types** – String, Number, Boolean, Object and Array.
- **Output expression** – Yes

## Condition expressions

To define a condition, you refer to an input by its name and compare it to a value using any of the following relational operators:

Operator	Meaning	Supported data types	Example usage	Example meaning
<code>==</code>	Equal to (the data type must also be equal)	String, Number, Boolean	<code>A == B</code>	If A is equal to B
<code>!=</code>	Not equal to	String, Number, Boolean	<code>A != B</code>	If A isn't equal to B
<code>&gt;</code>	Greater than	Number	<code>A &gt; B</code>	If A is greater than B
<code>&gt;=</code>	Greater than or equal to	Number	<code>A &gt;= B</code>	If A is greater than or equal to B
<code>&lt;</code>	Less than	Number	<code>A &lt; B</code>	If A is less than B
<code>&lt;=</code>	Less than or equal to	Number	<code>A &lt;= B</code>	If A is less than or equal to B

You can compare inputs to other inputs or to a constant in a conditional expression. For example, if you have a numerical input called `profit` and another one called `expenses`, both `profit > expenses` or `profit <= 1000` are valid expressions.

You can use the following logical operators to combine expressions for more complex conditions. We recommend that you use parentheses to resolve ambiguities in grouping of expressions:

Operator	Meaning	Example usage	Example meaning
<code>and</code>	Both expressions are true	<code>(A &lt; B) and (C == 1)</code>	If both expressions are true:

Operator	Meaning	Example usage	Example meaning
			<ul style="list-style-type: none"> <li>A is less than B</li> <li>C is equal to 1</li> </ul>
or	At least one expression is true	$(A \neq 2) \text{ or } (B > C)$	If either expressions is true: <ul style="list-style-type: none"> <li>A isn't equal to B</li> <li>B is greater than C</li> </ul>
not	The expression isn't true	$\text{not } (A > B)$	If A isn't greater than B (equivalent to $A \leq B$ )

## Prompt node

A prompt node defines a prompt to use in the flow. The inputs to the prompt node are values to fill in the variables that you define for the prompt. The output is the generated response from the model. For more information, see [Use prompts with your Flows app](#).

- **Node name** – Any
- **Prompt** – The [prompt](#) that the prompt node uses.
- **Version** – The [prompt](#) the version of the prompt.
- **Input field name** – The name of the prompt variable.
- **Input field types** – String, Number, Boolean, Object and Array.
- **Input expression** – Yes
- **Condition field name** – modelCompletion
- **Output field types** – String
- **Output expression** – No

## Knowledge Base node

A Knowledge Base node lets you send a query to a knowledge base. For more information, see [Knowledge Base data source](#).

- **Node name** – Any
- **Knowledge base** – The [Knowledge Base](#) that the node uses.
- **Response type** – The model that the node uses to generate a response.
- **Input field name** – retrievalQuery
- **Input field types** – String
- **Input expression** – Yes
- **Condition field name** – outputText
- **Output field types** – String
- **Output expression** – No

## Define inputs with expressions

When you configure the inputs for a node, you must define it in relation to the whole input that will enter the node. The whole input can be a string, number, boolean, array, or object. To define an input in relation to the whole input, you use a subset of supported expressions based off [JsonPath](#). Every expression must begin with `$.data`, which refers to the whole input. Note the following for using expressions:

- If the whole input is a string, number, or boolean, the only expression that you can use to define an individual input is `$.data`
- If the whole input is an array or object, you can extract a part of it to define an individual input.

As an example to understand how to use expressions, let's say that the whole input is the following JSON object:

```
{
  "animals": {
    "mammals": ["cat", "dog"],
    "reptiles": ["snake", "turtle", "iguana"]
  },
  "organisms": {
    "mammals": ["rabbit", "horse", "mouse"],
    "flowers": ["lily", "daisy"]
  },
  "numbers": [1, 2, 3, 5, 8]
```

}

You can use the following expressions to extract a part of the input (the examples refer to what would be returned from the preceding JSON object):

Expression	Meaning	Example	Example result
<code>\$.data</code>	The entire input.	<code>\$.data</code>	The entire object
<code>.name</code>	The value for a field called <i>name</i> in a JSON object.	<code>\$.data.numbers</code>	[1, 2, 3, 5, 8]
<code>[int]</code>	The member at the index specified by <i>int</i> in an array.	<code>\$.data.animals.reptiles[2]</code>	turtle
<code>[int1, int2, ...]</code>	The members at the indices specified by each <i>int</i> in an array.	<code>\$.data.numbers[0, 3]</code>	[1, 5]
<code>[int1:int2]</code>	An array consisting of the items at the indices between <i>int1</i> (inclusive) and <i>int2</i> (exclusive) in an array. Omitting <i>int1</i> or <i>int2</i> is equivalent to the marking the beginning or end of the array.	<code>\$.data.organisms.mammals[1:]</code>	["horse", "mouse"]
<code>*</code>	A wildcard that can be used in place of a <i>name</i> or <i>int</i> . If there are multiple results, the results are returned in an array.	<code>\$.data.*.mammals</code>	[["cat", "dog"], ["rabbit", "horse", "mouse"]]

The following procedure shows how to use expressions to identify fields in a JSON object that you send to a prompt node. The prompt generates a playlist of songs. The JSON object you pass to the flow identifies the number of songs that you want in the playlist and the genre of music that you want the songs to represent. For example, enter the following JSON object to request a playlist of 3 songs in the pop genre.

```
{ "genre": "Pop", "number": 3 }
```

### To use an expression

1. Create an empty Prompt Flows app by doing [Create an initial Prompt Flow app](#).
2. In the flow builder, choose the **Flow input** node.
3. In the **Prompt Flow builder** pane choose the **Configure** tab.
4. In **Outputs** section, choose **Type** and then select **Object**.
5. In the **Prompt Flow builder** pane, select **Nodes**.
6. From the **Orchestration** section, drag a **Prompt** node onto the flow builder canvas.
7. Select the node you just added.
8. In the **Configurations** tab of the **Prompt Flow builder** pane, do the following:
  - a. For **Node name**, enter **playlist\_songs\_genre\_node**.
  - b. In **Prompt details** choose **Create new prompt** to open the **Create prompt** pane.
  - c. For **Prompt name**, enter **playlist\_songs\_genre\_prompt**.
  - d. For **Model**, choose the model that you want the prompt to use.
  - e. For **Prompt message** enter **Create a playlist of `{{number}}` songs that are in the `{{genre}}` genre..**
  - f. (Optional) In **Model configs**, make changes to the inference parameters.
  - g. Choose **Save draft and create version** to create the prompt. It might take a couple of minutes to finish creating the prompt.
9. In the flow builder, choose the prompt node that you just added.
10. Choose the **Configure** tab and do the following in the **Prompt details** section:
  - a. For **Prompt**, select the prompt that you just created (**playlist\_songs\_genre\_prompt**).
  - b. For **Version**, select the version (**1**) of the prompt to use.
  - c. For the **number** input in the **Inputs** section, do the following:

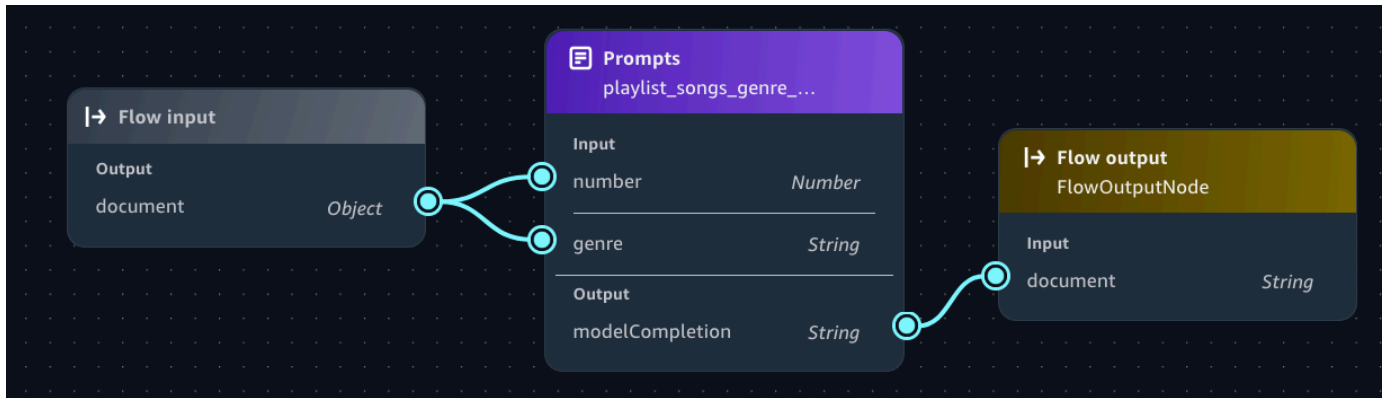


- i. Change the value of **Type** to **Number**.
  - ii. Change the value of **Expression** to `$.data.number`.
- d. For the **genre** input in the **Inputs** section, do the following:
- i. Make sure the value of **Type** is **String**.
  - ii. Change the expression for the input to `$.data.genre`.

The screenshot shows the 'Prompt Flow builder' interface with two tabs: 'Nodes' and 'Configure'. The 'Configure' tab is active. Under the 'Inputs' section, there are two input nodes being configured. The first node has a 'Name' field containing 'number', a 'Type' dropdown menu set to 'Number', and an 'Expression' field containing '\$.data.number'. The second node has a 'Name' field containing 'genre', a 'Type' dropdown menu set to 'String', and an 'Expression' field containing '\$.data.genre'.

11. Connect the output from **Flow input** node to the input **number** of the Prompt node.
12. Connect the output from **Flow input** node to the input **genre** of the Prompt node.
13. Connect the output from the prompt node to the input of the **Flow output** node.

14. Choose **Save** to save the flow. The flow should look similar to the following.



15. Test your prompt by doing the following:

- On the right side of the page, choose **<** to open the **Test** pane.
- Enter the following JSON in the **Enter prompt** text box.

```
{
  "genre": "Pop",
  "number": 3
}
```

- Press **Ctrl + Enter** on your keyboard or choose the run button to test the prompt. The response should be a playlist of 3 songs in the pop music genre.

## Use prompts with your Flows app

*Prompt Management* let you create and manage prompts for use in a Prompt Flows app. This allows you to pre-configure a prompt for a flow, by choosing the model and inference parameters that the model uses. You can also customize the prompt for different use cases by using variables. For example, you could have a prompt that creates a playlist of songs about topics that a user chooses. You can then use the prompt in a [Prompt Flows app](#).

### **⚠ Important**

If your workspace was created before Amazon Bedrock Studio added support for Prompt Management, you won't be able to create a prompt until your administrator updates your workspace. Ask your administrator to follow the instructions at [Update a workspace for Prompt management and Flows](#).

## Topics

- [Create a prompt with Prompt Management](#)
- [Add a prompt to a Prompt Flows app](#)
- [Modify a prompt with Prompt Management](#)
- [Delete a prompt with Prompt Management](#)

## Create a prompt with Prompt Management

When you create a prompt, you select a model for it and can modify inference parameters. To adjust the prompt for different use cases, you can include up to 5 variables.

You define variables in a prompt by surrounding them in double curly braces `{{variable}}`. For example, the following prompt defines two variables, `topic` and `location`.

*Generate a playlist of songs about {{topic}}. Make sure each song is by artists from {{location}}.*

When you run the prompt, you supply values for the variables. Bedrock Studio fills the prompt with the variable values and then passes the prompt to the model. For example, if you supply a `topic` value of *castle* and a `location` value of *Wales*, the model generates a playlist of songs about castles by Welsh artists.

You initially create a draft of your prompt. You can then test your prompt by inputting test values for the variables and running the prompt. These values are only for temporary testing and aren't saved to your prompt.

When you are ready, you can create a version of your prompt for use in a Prompt Flows app. You can create multiple versions of a prompt, but you can only [edit](#) the latest version. When you [delete](#) a prompt, it deletes all versions of the prompt.

### Warning

The name and description of a prompt are visible to all members of the Amazon Bedrock workspace. We recommend that you don't store sensitive data in these two fields. Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

## To create a prompt

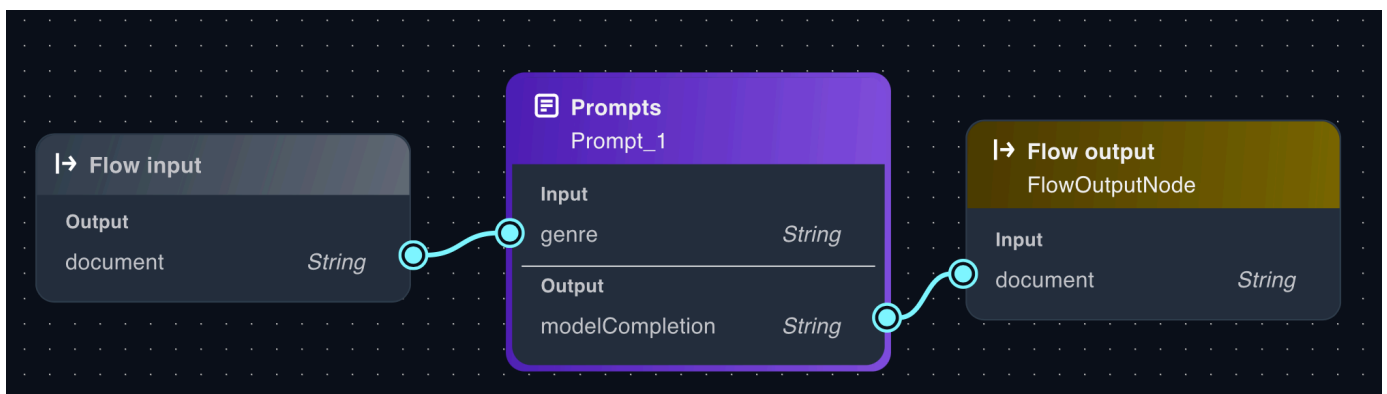
1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.
3. In the **Project** dropdown list, choose the project that you want to use or create a new project.
4. Open the navigation pane by choosing the menu at the top left of the page.
5. In **Project details** choose **Prompts**.
6. Choose **Create prompt** to start creating a prompt.
7. Choose the prompt name (**Untitled Prompt-nnnn**) and enter a name for the prompt.
8. In the **Configs** section, do the following:
  - a. For **Model**, select the model that you want to use.
  - b. (Optional) In **Parameters**, set the inference parameters values that you want to use. If you don't make changes, the prompt uses the default values for the model. For more information, see [Influence model responses with inference parameters](#).
  - c. (Optional) In **Prompt details** enter a description for your prompt.
9. In the center pane, enter **Generate a playlist of songs about {{topic}}. Make sure each song is by artists from {{location}}**. in the **Prompt message** text box.
10. (Optional) Choose **Save draft** to save a draft of your prompt.
11. Test your prompt in the right pane by doing the following:
  - a. For **Test variable values**, enter the following values for your prompt variables.
    - **topic**– Enter **castles**.
    - **location**– Enter **Wales**.
  - b. Choose **Run** to test your prompt. You should see your prompt, with populated variables, in the **Test** section. Bedrock Studio displays the response from the model underneath your prompt.
  - c. (Optional) Choose **Reset** to clear previously shown test results.
  - d. Continue to make changes to the prompt and variables until you are satisfied with the results. You can choose **Reset** to clear previously shown test results.

12. When you are ready, choose **Create version** to create a version of your prompt. If the button is disabled, wait until Bedrock Studio completes saving the prompt, which should take up to a minute.
13. Add your prompt to a [Prompt Flows app](#).

## Add a prompt to a Prompt Flows app

In this procedure, you add a prompt to an existing [Prompt Flow app](#).

1. Open the Prompt Flows app in Amazon Bedrock Studio.
2. In the **Prompt Flow builder** pane, select the **Nodes** tab.
3. From the **Orchestration** section, drag a **Prompt** node onto the flow builder canvas.
4. In the the flow builder, select the Prompt node that you just added.
5. In the **Prompt Flow builder** pane, choose the **Configure** tab and do the following:
  - a. For **Node name**, enter a name for the Prompt node.
  - b. For **Prompt** in the **Prompt details** section, select the prompt that you just created.
  - c. For **Version**, select the version of the prompt that you just created.
  - d. If you want to identify specific data from the upstream node that the prompt should use, change the value in **Expression**. For more information, see [Define inputs with expressions](#).
6. The circles on the nodes are connection points. For each variable, draw a line from the circle on the upstream node (such as the **Flow input** node) to the circle for the variable in the **Input** section of the prompt node.
7. Connect the **Output** of the prompt node to the downstream node that you want the prompt to send its output to. The Prompt Flow should look similar to the following image:



8. Choose **Save** to save your changes.

## Modify a prompt with Prompt Management

You can modify the current draft prompt or modify previous versions of a prompt. To modify a prompt, you select the version of the prompt (or current working draft prompt) that you want to modify. You then work on a draft update of the prompt. You can change the configuration for different versions of a prompt. For example, different versions of a prompt can use different Amazon Bedrock Studio models or use different inference parameters.

After testing the draft prompt, you can then save the draft as a new version of the prompt. If you want to use a new version of a prompt in a Prompt Flows app, update the version of the prompt in the app configuration. For more information, see [Add a prompt to your Prompt Flows app](#).

For more information about the changes you can make, see [Create a prompt with Prompt Management](#).

### To modify a prompt

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.
3. In the **Project** dropdown list, choose the project that contains the prompt that you want to modify.
4. Open the navigation pane by choosing the menu at the top left of the page.
5. In **Project details** choose **Prompts**.
6. In **Prompts**, select the prompt that you want to modify.
7. In the **Drafts and versions** pane, do one of the following:
  - If you want to modify the current draft prompt, choose **View and edit** in the working draft tile.
  - If you want to edit a previous version of the prompt, choose **Restore** in the tile for the prompt version. If the current draft prompt isn't already saved, Amazon Bedrock Studio offers to save the draft as new version of the prompt. To save the the draft, select **Save the current working draft as a version**. If you don't save the draft, the current draft prompt becomes the version of the prompt that you chose to edit. Amazon Bedrock Studio deletes the previous draft prompt.
  - If you want to try the prompt without making changes, choose **View** to open the prompt. You can't modify the prompt, but you can test and delete the prompt.
8. In **Configs** make changes to the model and inference parameters.

9. For **Prompt message**, use the text box to make changes to the prompt message.
10. (Optional) Choose **Save draft** to save a draft of your prompt.
11. In **Test** enter values for the prompt variables and choose run to test your changes.
12. When you are satisfied with your changes, choose **Create version** to create a new version of your prompt.

## Delete a prompt with Prompt Management

You can delete prompts that you have previously created. When you delete a prompt, Bedrock Studio checks if deleting the prompt affects any apps that use the prompt. After you confirm deletion, Bedrock Studio deletes the prompt draft and all versions of the prompt that you have created. Prompts that you delete are retained in the AWS servers for up to fourteen days.

### To delete a prompt

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.
3. In the **Project** dropdown list, choose the project that contains the prompt that you want to delete.
4. Open the navigation pane by choosing the menu at the top left of the page.
5. In **Project details**, choose **Prompts**.
6. In the **Prompts** section, select the prompt that you want to delete.
7. Choose **Delete**.
8. In the **Delete** dialog box, check if deleting the prompt affects any of your apps. You can still delete the prompt, but you will need to make changes to the apps that use the prompt.
9. Choose **Delete** if you are ready to delete prompt.



# Influence model responses with inference parameters

Inference parameters are values that you can adjust to limit or influence how a model generates a response to a prompt. For example, in the chat app you create in [Build a chat app with Amazon Bedrock Studio](#), you can use inference parameters to adjust the randomness and diversity of the songs that the model generates for a playlist.

You can apply inference parameters to models you use in [explore mode](#), [chat apps](#), and [Flows apps](#).

## Randomness and diversity

For any given sequence, a model determines a probability distribution of options for the next token in the sequence. To generate each token in an output, the model samples from this distribution. Randomness and diversity refer to the amount of variation in a model's response. You can control these factors by limiting or adjusting the distribution. Foundation models typically support the following parameters to control randomness and diversity in the response.

- **Temperature**– Affects the shape of the probability distribution for the predicted output and influences the likelihood of the model selecting lower-probability outputs.
  - Choose a lower value to influence the model to select higher-probability outputs.
  - Choose a higher value to influence the model to select lower-probability outputs.

In technical terms, the temperature modulates the probability mass function for the next token. A lower temperature steepens the function and leads to more deterministic responses, and a higher temperature flattens the function and leads to more random responses.

- **Top K** – The number of most-likely candidates that the model considers for the next token.
  - Choose a lower value to decrease the size of the pool and limit the options to more likely outputs.
  - Choose a higher value to increase the size of the pool and allow the model to consider less likely outputs.

For example, if you choose a value of 50 for Top K, the model selects from 50 of the most probable tokens that could be next in the sequence.

- **Top P** – The percentage of most-likely candidates that the model considers for the next token.
  - Choose a lower value to decrease the size of the pool and limit the options to more likely outputs.

- Choose a higher value to increase the size of the pool and allow the model to consider less likely outputs.

In technical terms, the model computes the cumulative probability distribution for the set of responses and considers only the top P% of the distribution.

For example, if you choose a value of 0.8 for Top P, the model selects from the top 80% of the probability distribution of tokens that could be next in the sequence.

The following table summarizes the effects of these parameters.

Parameter	Effect of lower value	Effect of higher value
Temperature	Increase likelihood of higher-probability tokens	Increase likelihood of lower-probability tokens
	Decrease likelihood of lower-probability tokens	Decrease likelihood of higher-probability tokens
Top K	Remove lower-probability tokens	Allow lower-probability tokens
Top P	Remove lower-probability tokens	Allow lower-probability tokens

As an example to understand these parameters, consider the example prompt **I hear the hoof beats of "**. Let's say that the model determines the following three words to be candidates for the next token. The model also assigns a probability for each word.

```
{
  "horses": 0.7,
  "zebras": 0.2,
  "unicorns": 0.1
}
```

- If you set a high **temperature**, the probability distribution is flattened and the probabilities become less different, which would increase the probability of choosing "unicorns" and decrease the probability of choosing "horses".

- If you set **Top K** as 2, the model only considers the top 2 most likely candidates: "horses" and "zebras."
- If you set **Top P** as 0.7, the model only considers "horses" because it is the only candidate that lies in the top 70% of the probability distribution. If you set **Top P** as 0.9, the model considers "horses" and "zebras" as they lie in the top 90% of probability distribution.

# Add a data source to your app

You can use your own data in your apps by adding a data source. This allows your app access to information that is only available to you. In Bedrock Studio a data source can be a document or an Amazon Bedrock Knowledge Base.

## Topics

- [Document data source](#)
- [Knowledge Base data source](#)
- [Create a Knowledge Base component](#)
- [Add a Knowledge Base component to a chat app](#)
- [Add a Knowledge Base component to a Prompt Flow app](#)

## Document data source

A document is a file that contains information that you want the model to use when generating a response. By using a document as a data source, your app users can chat with a document. For example, they can use a document to answers questions, make an analysis, create a summary, itemize fields in a numbered list, or rewrite content. Amazon Bedrock Studio doesn't store your document or its data after use.

You can use a document as a data source in a chat app. For more information, see [Add a document data source to your chat app](#).

### Note

You can't use a document data source in a Prompt Flows app.

## Knowledge Base data source

You can also use a Knowledge Base to store your data. A Knowledge Base provides you the capability of amassing data sources into a repository of information. With Knowledge Bases, you can easily build an app that takes advantage of *retrieval augmented generation (RAG)*, a

technique in which the retrieval of information from data sources augments the generation of model responses. You can only access Knowledge Bases that you create within Amazon Bedrock Studio. You can't access Knowledge Bases that you create in Amazon Bedrock.

You can use a Knowledge Base in a [chat app](#) and a [Prompt Flows app](#).

The document file or the Knowledge Base source file must be in PDF, MD, TXT, DOC, DOCX, HTML, CSV, XLS or XLSX format. The maximum file size is 10 MB.

## Create a Knowledge Base component

You can create a Knowledge base as a component in a Amazon Bedrock Studio project. If you are creating an app, you can also create a Knowledge Base when you configure the app. For an example, see [Create a flows app](#).

### Warning

The name and description of a Knowledge Base component are visible to all members of the Amazon Bedrock Studio workspace. We recommend that you don't store sensitive data in these two fields. Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

### To create a Knowledge Base

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.
3. In the **Project** dropdown list, choose the project that you want to add a Knowledge Base to.
4. In the **Components** section, choose **Create component** and then **Knowledge Base**. The **Create Knowledge Base** pane is shown.
5. For **Knowledge Base name**, enter a name for the Knowledge Base.
6. For **Knowledge Base description**, enter a description for the Knowledge Base.
7. In **Add data sources**, choose **Click to upload** and upload the document that you want the Knowledge Base to use. Alternatively, add your source documents by dragging and dropping the document from your computer.
8. (Optional) For **Embeddings model**, choose a different model, if available.

9. (Optional) For **Vector store**, choose a different vector store, if available.
10. Choose **Create** to create the Knowledge Base.
11. Use the Knowledge Base in an app, by doing one of the following:
  - If your app is a chat app, do [Add a Knowledge Base component to a chat app](#).
  - If your app is a Prompt Flows app, do [Add a Knowledge Base component to a Prompt Flow app](#).

## Add a Knowledge Base component to a chat app

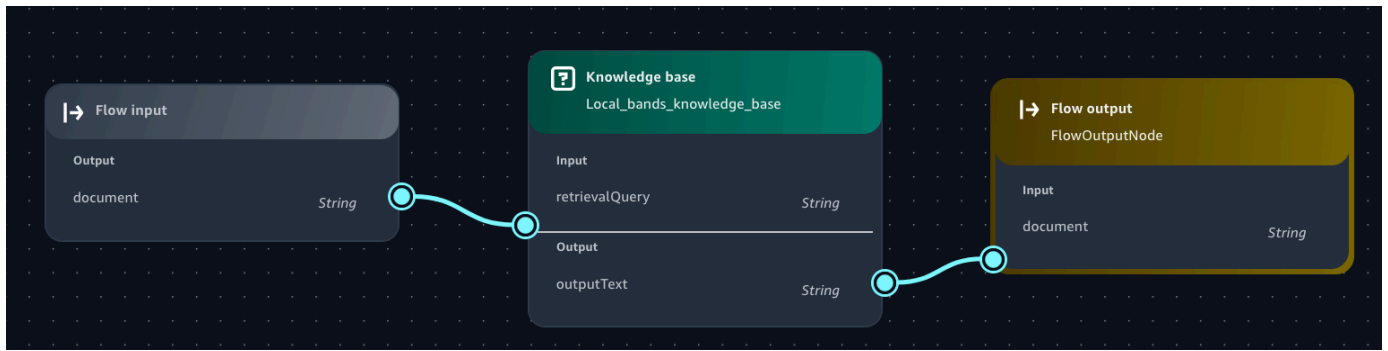
In this procedure, you add a Knowledge Base component to an existing [chat app](#).

1. Open the chat app in Amazon Bedrock Studio.
2. In the **Configs** pane, choose **Data**.
3. Select **Use Knowledge Base**.
4. For **Select Knowledge Base**, select the Knowledge Base component that you created in [Create a Knowledge Base component](#).
5. Choose **Save** to save your changes.

## Add a Knowledge Base component to a Prompt Flow app

In this procedure, you add a Knowledge Base component to an existing [Prompt Flow app](#).

1. Open the Prompt Flows app in Amazon Bedrock Studio.
2. In the **Prompt Flow builder** pane, select the **Nodes** tab.
3. From the **Data** section, drag a **Knowledge Base** node onto the flow builder canvas.
4. The circles on the nodes are connection points. Draw a line from the circle on the upstream node (such as the **Flow input** node) to the circle on the **Input** section of the Knowledge Base node that you just added.
5. Connect the **Output** of the Knowledge Base node to the downstream node that you want the Knowledge Base to send its output to. The Prompt Flow should look similar to the following image:



6. In the the flow builder, select the Knowledge Base node that you just added.
7. In the **Prompt Flow builder** pane, choose the **Configure** tab and do the following:
  - a. For **Node name**, enter a name for the Knowledge Base node.
  - b. For **Select Knowledge Base** in the **Knowledge Base Details** section, select the Knowledge Base that you just created.
  - c. For **Select response generation model**, select the model that you want the Knowledge Base to generate responses with.
8. Choose **Save** to save your changes.

# Safeguard your app with a guardrail

Guardrails for Amazon Bedrock lets you implement safeguards for your Amazon Bedrock Studio app based on your use cases and responsible AI policies. You can create multiple guardrails tailored to different use cases and apply them across multiple foundation models, providing a consistent user experience and standardizing safety controls across generative AI apps. You can configure denied topics to disallow undesirable topics and content filters to block harmful content in the prompts you send to a model and to the responses you get from a model. You can use guardrails with text-only foundation models. For more information, see [Safeguard your app with a guardrail](#).

You can use guardrails with Amazon Bedrock Studio chat apps. You can add a guardrail component when you [create a chat app](#) or you can create a guardrail component within a project. Later, you can use the guardrail component in a chat app. For more information, see [Create a guardrail component](#).

## Topics

- [Guardrail policies](#)
- [Create a guardrail component](#)
- [Add a guardrail component to a chat app](#)

## Guardrail policies

A guardrail consists of the following policies to avoid content that falls into undesirable or harmful categories.

- Content filters – Adjust filter strengths to filter input prompts or model responses containing harmful content.
- Denied topics – You can define a set of topics that are undesirable in the context of your app. These topics will be blocked if detected in user queries or model responses.

## Content filters

Guardrails in Bedrock Studio support the following content filters to detect and filter harmful user inputs and FM-generated outputs.



- **Hate** – Describes language or a statement that discriminates, criticizes, insults, denounces, or dehumanizes a person or group on the basis of an identity (such as race, ethnicity, gender, religion, sexual orientation, ability, and national origin).
- **Insults** – Describes language or a statement that includes demeaning, humiliating, mocking, insulting, or belittling language. This type of language is also labeled as bullying.
- **Sexual** – Describes language or a statement that indicates sexual interest, activity, or arousal using direct or indirect references to body parts, physical traits, or sex.
- **Violence** – Describes language or a statement that includes glorification of or threats to inflict physical pain, hurt, or injury toward a person, group or thing.

Content filtering depends on the confidence classification of user inputs and FM responses across each of the four harmful categories. All input and output statements are classified into one of four confidence levels (NONE, LOW, MEDIUM, HIGH) for each harmful category. For example, if a statement is classified as *Hate* with HIGH confidence, the likelihood of the statement representing hateful content is high. A single statement can be classified across multiple categories with varying confidence levels. For example, a single statement can be classified as *Hate* with HIGH confidence, *Insults* with LOW confidence, *Sexual* with NONE confidence, and *Violence* with MEDIUM confidence.

For each of the harmful categories, you can configure the strength of the filters. The filter strength determines the degree of filtering harmful content. As you increase the filter strength, the likelihood of filtering harmful content increases and the probability of seeing harmful content in your app reduces. The following table shows the degree of content that each filter strength blocks and allows.

Filter strength	Blocked content confidence	Allowed content confidence
None	No filtering	None, Low, Medium, High
Low	High	None, Low, Medium
Medium	High, Medium	None, Low
High	High, Medium, Low	None

## Denied topics

Guardrails can be configured with a set of denied topics that are undesirable in the context of your generative AI app. For example, a bank may want their online assistant to avoid any conversation related to investment advice or engage in conversations related to fraudulent activities such as money laundering.

You can define up to five denied topics. Input prompts and model completions will be evaluated against each of these topics. If one of the topics is detected, the blocked message configured as part of the guardrail will be returned to the user.

Denied topics can be defined by providing a natural language definition of the topic along with a few optional example phrases of the topic. The definition and example phrases are used to detect if an input prompt or a model completion belongs to the topic.

Denied topics are defined with the following parameters.

- **Name** – The name of the topic. The name should be a noun phrase. Don't describe the topic in the name. For example:
  - **Investment Advice**
- **Definition** – Up to 200 characters summarizing the topic content. The description should describe the content of the topic and its subtopics.

### Note

For best results, adhere to the following principles:

- Don't include examples or instructions in the description.
- Don't use negative language (such as "don't talk about investment" or "no content about investment").

The following is an example topic description that you can provide:

- **Investment advice refers to inquires, guidance or recommendations regarding the management or allocation of funds or assets with the goal of generating returns or achieving specific financial objectives.**

- **Sample phrases** – A list of up to five sample phrases that refer to the topic. Each phrase can be up to 1,000 characters. An sample is a prompt or continuation that shows what kind of content should be filtered out. For example:
  - **Is investing in the stocks better than bonds?**
  - **Should I invest in gold?**

## Create a guardrail component

You can create a guardrail as a component in a Amazon Bedrock Studio project. You can then add the guardrail component to a chat app. You can also create a guardrail component while you are creating a chat app. For an example, see [Add a guardrail to your chat app](#).

### **Warning**

The name and description of a guardrail component are visible to all members of the Amazon Bedrock Studio workspace. We recommend that you don't store sensitive data in these two fields. Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

### To create a guardrail component

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.
3. In the **Project** dropdown list, choose the project that you want to add a guardrail to.
4. In the **Components** section, choose **Create component** and then **Guardrail**. The **Create guardrail** pane is shown.
5. For **Guardrail name**, enter a name for the guardrail.
6. For **Guardrail description** enter a description for the guardrail.
7. In **Content filters** do the following.
  - a. Select **Enable content filters** to turn on content filtering.
  - b. For **Filter for prompts**, choose the filters that you want to apply to prompts. For more information, see [Content filters](#).

- c. If you want the filter to apply to responses that the model generates, select **Apply the same filters for responses**.
8. In **Blocked messaging** do the following:
  - a. For **Blocked messaging for prompts** enter a message to display when the guardrail blocks content in the prompt.
  - b. If you want to show a different message when the guardrail blocks content from a model's response, do the following:
    - i. Clear **Apply the same message for blocked responses**.
    - ii. For **Blocked messaging for responses**, enter a message to display when the guardrail blocks content in the response from the model.
9. Add a denied topic filter by doing the following:
  - a. Choose **Use advanced features**.
  - b. Choose **Denied topics**.
  - c. Choose **Add topic**.
  - d. For **Name**, enter a name for the filter.
  - e. For **Definition for topic**, enter a definition for the content that you want to deny.
  - f. (Optional) To help guide the guardrail, do the following:
    - i. Choose **Sample phrases - optional**.
    - ii. For **Sample phrases**, enter a phrase.
    - iii. Choose **Add phrase**.
    - iv. Add up to four more phrases by repeating the previous two steps.
    - v. Choose **Save**.

For information about denied topics, see [Denied topics](#).

10. Choose **Create** to create the guardrail.
11. Add the guardrail component to a chat app by doing [Add a guardrail component to a chat app](#).

## Add a guardrail component to a chat app

In this procedure, you add a guardrail component to an existing [chat app](#).

1. Open the chat app in Amazon Bedrock Studio.
2. In the **Configs** pane, choose **Guardrails**.
3. For **Guardrails**, select the guardrail component that you created in [Create a guardrail component](#).
4. (optional) Preview the guardrail by choosing **Preview**. From the preview you can edit the guardrail, if desired.
5. Choose **Save** to save your changes.

# Call functions from your app

Amazon Bedrock Studio functions let a model include information that it has no previous knowledge of in its response. For example, you can use a function to include dynamic information in a model's response such as a weather forecast, sports results, or traffic conditions.

In Amazon Bedrock Studio, a function calls an API hosted outside of Amazon Bedrock Studio. You either create the API yourself, or use an existing API. To create an API, you can use [Amazon API Gateway](#).

To use a function in Amazon Bedrock Studio you add a *function component* to your app. As part of the function, you define an OpenAPI schema for the API that you want the model to call. You also specify how to authenticate the call to the API. When a model receives a prompt, it uses the schema and the prompt to determine if an API should be called and the parameters that the API should receive. If the API is called, the response from the model includes the output from the API.

APIs that you call in a function must have a response size that is less than 20K.

When add a function to an app, you need to specify the app's system prompt. The system prompt needs to be at least 40 characters long and should mention the new skills that the new function introduces.

You can use functions in a [Chat app](#).

## Topics

- [Function schema](#)
- [Authentication methods](#)
- [Create an Amazon Bedrock Studio function component](#)
- [Add a function component to a chat app](#)

## Function schema

Amazon Bedrock Studio has the following requirements for the schema that you use to create a function.

- The function schema must be [OpenAPI version 3.0.0](#).

- The function schema must be in JSON or YAML format.
- The function can have no authentication, API key authentication, Bearer token authentication, or basic authentication. For more information, see [Authentication methods](#).
- You can have 0 or 1 server URL.
- All [Operation Objects](#) must have a description.
- All [Parameter Objects](#) must have a description.
- [Security scheme object](#) must have a type that is either `apiKey` or `http`.

When the type is `http`, the `scheme` field must either be `basic` or `bearer`.

When the type is `apiKey`, the `in` property must be `query` or `header`. Also, the `name` property must be defined.

- Bedrock Studio only honors [globally-scoped security requirement](#). For more information, see [Valid components for globally-scoped security requirements](#).
- Parameters (**parameter.in**) must be passed through query or path. You can't use cookies or headers to pass parameters.
- Parameters (**parameter schema type**) must be primitive types, arrays, or objects (one-level JSON). You can't pass complex nested objects.
- Parameter content (**parameter.content**) is mutually exclusive with the schema. Schema is more commonly used. Use content only for more complex types, or for complex serialization scenarios that are not covered by style and explode.
- Parameter **style** and **explode** values. `form` and `true` for query, `simple` and `false` for paths). For more information, see [Parameter Serialization](#).
- Request body content must be passed as `application/json`.
- The schema can have up to 5 APIs and an app can use up to 5 APIs across all functions. For the model to correctly choose function, it is important to provide detailed descriptions of the API, including parameters, properties, and responses.

## Valid components for globally-scoped security requirements

Bedrock Studio only honors [globally-scoped security requirements](#). That is, Bedrock Studio ignores security requirements indicated in operation objects.

When the requirement array contains a security scheme object with type `http` and scheme of `bearer` or `basic`, the array must contain a single entry. Bedrock Studio ignores further entries.

When the requirement array contains a security scheme object with type `apiKey`, you can have a maximum of 2 entries.

For example, if you have the following [components](#):

```
"components": {
  "securitySchemes": {
    "api_key_1": {
      "type": "apiKey",
      "name": "appid1",
      "in": "query"
    },
    "api_key_2": {
      "type": "apiKey",
      "name": "appid2",
      "in": "header"
    },
    "api_key_3": {
      "type": "apiKey",
      "name": "appid3",
      "in": "cookie"
    },
    "bearer_1": {
      "type": "http",
      "scheme": "bearer",
    },
    "bearer_2": {
      "type": "http",
      "scheme": "bearer",
    },
    "basic_1": {
      "type": "http",
      "scheme": "basic",
    },
    "basic_2": {
      "type": "http",
      "scheme": "basic",
    },
    "http_digest": {
      "type": "http",
      "scheme": "digest"
    },
    "oauth2_1": {
      "type": "oauth2"
    }
  }
}
```



```
    }  
  }  
}
```

The following are valid:

```
# 1 API key  
"security": [  
  {  
    "api_key_1": []  
  }  
],  
  
# 2 API keys  
"security": {  
  {  
    "api_key_1": [],  
    "api_key_2": []  
  }  
}  
  
# Bearer  
"security": {  
  "bearer_1": []  
}  
  
# Basic  
"security": {  
  "basic_1": []  
}
```

The following are invalid:

```
# Invalid: `type` must only be `apiKey` or `http`  
"security": {  
  "oauth2_1": []  
}  
  
# Invalid: `scheme` must only be `basic` or `bearer` if `type` is `http`  
"security": {  
  "http_digest": []  
}
```

```
# Invalid: `security` must only contain 1 entry if `type` is `basic` or `bearer`
"security": {
  "basic_1": [],
  "basic_2": []
}

# Invalid: `security` must not contain varying security types
"security": {
  "api_key_1": [],
  "basic_1": []
}

# Invalid: API key must only have `in` property set to `header` or `query`
"security": {
  "api_key_1": [],
  "api_key_3": []
}

# Invalid: `security` must not have more than 2 API keys
"security": {
  {
    "api_key_1": [],
    "api_key_2": [],
    "api_key_3": []
  }
}
```

## Authentication methods

Amazon Bedrock Studio supports the following methods for authenticating function calls to an API server. If you authenticate a function call, make sure the credentials you provide are correct as Bedrock Studio doesn't verify the credentials before you use them in a function call.

- **No authentication** – No authentication means that the client doesn't need to provide any credentials to access a resource or service. This method is typically used for publicly available resources that don't require any form of authentication.
- **[API keys](#)** – An API key is a unique identifier used to authenticate a client application and allow it to access an API or service. You can add a maximum of two keys.
- **[Bearer token](#)** – A bearer token is an opaque string that represents an authentication credential. It is typically obtained after a successful authentication process, such as OAuth 2.0. This method

allows the client to access protected resources without having to send the actual credentials (username and password) with each request.

### Note

Amazon Bedrock Studio is unable to assure whether the token is valid or has already expired. It is your responsibility to make sure that you provide a valid token, and to update the token to a new one before it expires. If the token expires, Amazon Bedrock won't be able to successfully call APIs with the token.

- **Basic authentication** – Basic authentication is a simple authentication scheme built into the HTTP protocol. The credentials are sent with every request, which can be a security concern if the connection is not secured using HTTPS. Basic authentication is generally considered less secure than other modern authentication methods and should be used with caution, especially in production environments.

## Create an Amazon Bedrock Studio function component

You can create a function as a component in a Amazon Bedrock Studio project. If you are creating an app, you can also create a function when you configure the app. For an example, see [Add a function call to your chat app](#).

### Warning

The name and description of a function component are visible to all members of the Amazon Bedrock Studio workspace. We recommend that you don't store sensitive data in these two fields. Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

### To create a function component

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.
3. In the **Project** dropdown list, choose the project that you want to add a function to.
4. In the **Components** section, choose **Create component** and then **Function**. The **Create function** pane is shown.

5. For **Function name**, enter a name for the function in **Function name**.
6. For **Function description**, enter a description for the function.
7. For **Function schema**, enter the JSON or YAML format OpenAPI schema for the API. Alternatively, upload the JSON or YAML for the file by choosing **Import JSON/YAML**. You can clear the text box by choosing **Reset**.
8. Choose **Validate schema** to validate the schema.
9. For **Authentication method** select the authentication method for your API server. By default, Amazon Bedrock Studio preselects the authentication based on information it finds in your OpenAPI schema. For information about authentication methods, see [Authentication methods](#).
10. Enter the information for the authentication method that you selected in the previous step.
11. For **API servers**, enter the URL for your server in **Server URL**. This value is autopopulated if the server URL is in the schema.
12. Choose **Create** to create your function.
13. Add your function to a chat app by doing [Add a function component to a chat app](#).

## Add a function component to a chat app

In this procedure, you add a function component to an existing [chat app](#). You can add up to 5 functions to an app. For each function you add, be sure to update the system prompt with information about the function.

1. Open the chat app in Amazon Bedrock Studio.
2. In the **Configs** pane, do the following:
  - a. For **System prompt and examples**, enter or update the system prompt so that it describes the function.
  - b. Choose **Functions**.
  - c. For **Functions**, select the function component that you created in [Create an Amazon Bedrock Studio function component](#).
3. Choose **Save** to save your changes.

# Organize your work with projects in Amazon Bedrock Studio

Here you learn how to manage the resources (apps, components, and prompts) in your projects.

You can manage a project by using the *App builder* and *Project details*. The App builder is one way you can access the chat app or Prompt Flows app that you are currently working on. Project details is where you create and manage the apps, components, and prompts that are in the project.

## Topics

- [Create an Amazon Bedrock Studio project](#)
- [Share an Amazon Bedrock Studio project](#)
- [Stop sharing an Amazon Bedrock Studio project](#)
- [Delete resources from an Amazon Bedrock Studio project](#)
- [Delete an Amazon Bedrock Studio project](#)

## Create an Amazon Bedrock Studio project

Within Build mode, an Amazon Bedrock Studio project is where you organize the apps, components, and prompts that you create. After you create a project you can do the following:

- Create a [chat app](#).
- Create a [Prompt Flows app](#).
- Create [prompts](#) for your flows apps.
- Create [data sources](#) for your apps.
- Create [functions](#) for your chat apps.
- Create [guardrails](#) for your chat apps.

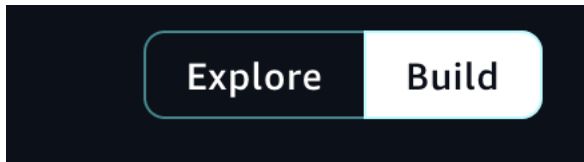
### Warning

The name and description of a project are visible to all members of the Amazon Bedrock Studio workspace. We recommend that you don't store sensitive data in these two fields.

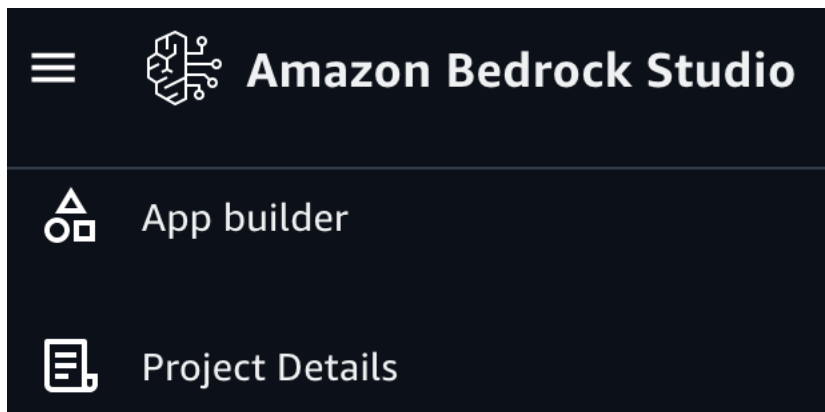
Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

## To create a project

1. Open your Amazon Bedrock Studio workspace.
2. Make sure that you are in Build mode, by choosing **Build** at the top right of the page.

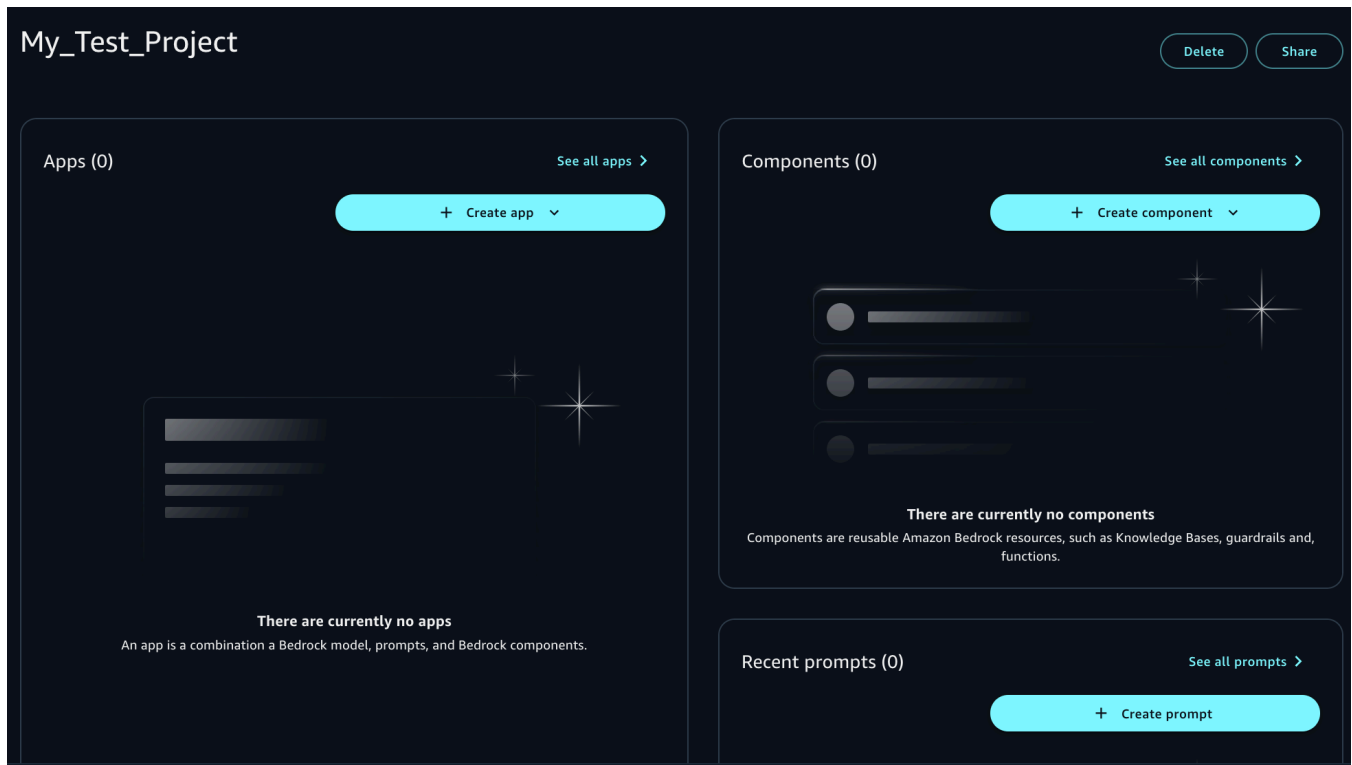


3. Choose the **Project** dropdown list box at the top of the page and then select **View all projects**
4. Choose **Create new project**.
5. For **Project name**, enter a name for your project.
6. For **Project description**, enter a name for your project.
7. Choose **Create** to create the project. It might take up to a minute to create the project.
8. In the **Projects** section, a new card for your appears. In the card, choose **Go to project**.
9. On the project page, note the following:
  - You can access the **App builder** and **Project details** navigation pane by choosing the menu at the top left of the page.



- **App builder** is where you can access the apps that you are currently working on. Amazon Bedrock Studio creates an empty [chat app](#) and an empty [Prompt Flows app](#) for you.
- **Project details** is where you can access the resources (apps, components, and prompts) that your app uses.

Choose **Overview** to get to an overview of your project. In the overview, you can also create resources for your project.



## Share an Amazon Bedrock Studio project

You can share a project with up to 20 workspace members. A workspace member can be a user or a group of users. A project can have up to 100000 users across up to 20 groups. To manage the members in your workspace, contact your administrator. For more information, see [Add or remove workspace members](#).

Note that members that you share a project with have permissions to delete the project.

### **Warning**

The name and description of projects, apps, and components are visible to all members of the Amazon Bedrock Studio workspace. We recommend that you don't store sensitive data in these two fields. Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

## To share a project

1. Open your Amazon Bedrock Studio workspace.
2. Choose **Build**.
3. At the top of the page, choose the **Project** dropdown list box and choose **View all projects**.
4. On the card for the project that you want to share, choose the menu button and the choose **Share project**.
5. For **Member type** choose **Individual user** or **Group**, depending on the type of member that you want share the project with.
6. Search for the users or groups that you want to share the project by entering the user name or group in the **Search by alias to invite members** text box.
7. In the drop down list box, the select the matching user name or group that want to share with. Bedrock Studio automatically adds the workspace member to the project.
8. Choose **Copy link** to copy the URL link that project members need to access the project.
9. Use email, or another secure communication method, to send the URL link to the members that you added to the project.

## Stop sharing an Amazon Bedrock Studio project

You can stop sharing a project with members of your workspace.

### **Warning**

The name and description of projects, apps, and components are visible to all members of the Amazon Bedrock Studio workspace. We recommend that you don't store sensitive data in these two fields. Project-based access control ensures only authorized members of a project can edit the name, description, and other fields in the project.

## To stop sharing a project

1. Open your Amazon Bedrock Studio workspace.
2. Choose **Build**.
3. At the top of the page, choose the **Project** dropdown list box and choose **View all projects**.



4. On the card for the project that you want to share, choose the menu button and the choose **Share project**.
5. In **All members with access**, choose the trash icon next to the user or group that you want to stop sharing with. Bedrock Studio automatically stops sharing the project to the member.

## Delete resources from an Amazon Bedrock Studio project

A project can contain one or more apps, components, or prompts. You can delete these resources, if you no longer need them.

### Note

If want to delete a project, you must first delete all resources in the project ([Delete an Amazon Bedrock Studio project](#)).

### To delete a resource from a project

1. Open your Amazon Bedrock Studio workspace.
2. Choose **Build**.
3. At the top of the page, choose the **Project** dropdown list box and choose the project from which you want to delete an app or component.
4. Delete an app by doing the following:
  - a. In the left navigation pane, open **Project details** and then choose **Apps**.
  - b. For the app you want to delete, choose the trash button.
  - c. In the delete app dialog box, choose **Delete** to delete the app.
5. Delete a component by doing the following:
  - a. In the left navigation pane, open **Project details** and then choose **Components**.
  - b. For the component that you want to delete, choose the menu button then choose **Delete**.
  - c. In the delete component dialog box, choose **Delete** to delete the component. After deleting the component Bedrock Studio returns you to the project page.
6. Delete a prompt by doing the following:

- a. In the left navigation pane, open **Project details** and then choose **Prompts**.
- b. For a prompt that you want to delete, choose the trash button.
- c. In the delete prompt dialog box, choose **Delete** to delete the prompt.

## Delete an Amazon Bedrock Studio project

You can delete projects that you have created and projects that others have shared to you. Before you can delete a project, you must first delete any associated apps, components and prompts.

### To delete a Amazon Bedrock Studio project

1. Open your Amazon Bedrock Studio workspace.
2. Choose **Build**.
3. At the top of the page, choose the **Project** dropdown list box and choose the project that you want to delete.
4. Delete the apps by doing the following:
  - a. In the left navigation pane, open **Project details** and then choose **Apps**.
  - b. For an app you want to delete, choose the trash button.
  - c. In the delete app dialog box, choose **Delete** to delete the app.
  - d. Repeat the previous two steps until you delete all apps in the project.
5. Delete the components by doing the following:
  - a. In the left navigation pane, open **Project details** and then choose **Components**.
  - b. For a component that you want to delete, choose the menu button then choose **Delete**.
  - c. In the delete component dialog box, choose **Delete** to delete the component. After deleting the component Bedrock Studio returns you to the project page.
  - d. Repeat the previous three steps for each component in the project.
6. Delete the prompts by doing the following:
  - a. In the left navigation pane, open **Project details** and then choose **Prompts**.
  - b. For a prompt that you want to delete, choose the trash button.
  - c. In the delete prompt dialog box, choose **Delete** to delete the prompt.
  - d. Repeat the previous two steps for each prompt in the project.

7. In the left navigation pane, open **Project details** and then choose **Overview**.
8. Choose **Delete** to delete the project. If the project still has associated apps or components, Bedrock Studio shows a dialog box. Chose **Got it** to go back to the project page and delete the remaining apps and components.
9. Contact the project teams member to let them know the project has been deleted. Bedrock Studio doesn't automatically notify team members that the project has been deleted.

# Document history for the Amazon Bedrock Studio User Guide

- **Latest documentation update:** August 20th 2024

The following table describes important changes in each release of Amazon Bedrock Studio. For notification about updates to this documentation, you can subscribe to an RSS feed.

Change	Description	Date
<a href="#">New features (Preview)</a>	You can now export a <a href="#">chat app</a> from Amazon Bedrock Studio.	August 20, 2024
<a href="#">New features (Preview)</a>	You can now use <a href="#">Prompt Flows</a> and <a href="#">Prompt Management</a> with Amazon Bedrock Studio.	July 22, 2024
<a href="#">Preview release</a>	Preview release of Amazon Bedrock Studio.	May 7, 2024