Generative AI is changing many industries in rapid pace of innovation. Amazon Bedrock is a major offering in this area from AWS. It is a fully managed service for hosting specialized foundation models. This chapter will explore Amazon Bedrock's features, benefits, and use cases.

Amazon Bedrock offers access to various foundation models. These include text, image, embedded, and multimodal models. These models facilitate the development of many generative AI applications like virtual assistance, content generation, and image creation. Amazon Bedrock provides many features for developing these applications, including model evaluation, safeguards, and provisioned throughput.

Amazon Bedrock emphasizes security, privacy, and responsible AI practices. You will learn about Amazon Bedrock's features for safe AI development. It includes tools for model evaluation, safeguards, and best practices.

This chapter begins with an overview of the service and its key features. Next, it discusses the various foundation models provided by Amazon Bedrock. Lastly, you will learn to interact with Amazon Bedrock via the console, API, and SDK.

You will not only delve into the features and benefits of Amazon Bedrock, but also engage in a series of practical exercises to initiate the development of generative AI applications using Amazon Bedrock. You will learn various topics like model selection, customization, and evaluation. This chapter offers a thorough introduction to Amazon Bedrock and its features. It is useful for you to create generative AI applications and business leaders wanting to utilize generative AI. You will acquire the skills and information to quick start.

# **3.1 What is Amazon Bedrock**

Amazon Bedrock is a fully managed service designed to host the high-performing purpose-built foundation models (FMs) sourced from leading AI startups, a well-known service provider, and Amazon. Also, Amazon Bedrock provides rich features for generative AI application development with a focus on security, privacy, and responsible AI practices.

Amazon Bedrock allows you to privately adjust and test various foundation models. You can use methods like customization and retrieval-augmented generation (RAG). These can be easily integrated into your enterprise systems and data sources. Amazon Bedrock provides a serverless experience for accessing base models via API. You can customize these models with your own data via both API and console. It simplifies rapid building of entire generative AI solutions using AWS tools, removing the need for infrastructure management.

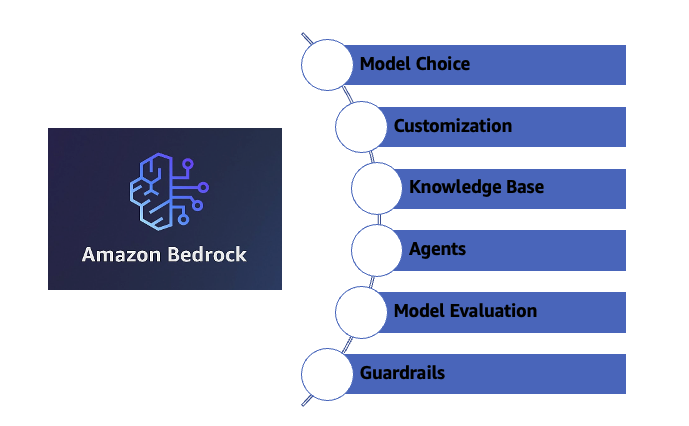
# **3.2 Features of Amazon Bedrock**

Key features of Amazon Bedrock include:

* Experimenting with prompts and configurations to generate responses.
* Augmenting response generation with information from your own data sources.
* Building agents to integrate and orchestrate with multiple tasks.
* Adapting models to specific tasks and domains by fine-tuning or continued pre-training.
* Improving application efficiency and output by purchasing provisioned throughput.
* Evaluating different models to determine the best fit for your use case.
* Implementing safeguards to prevent inappropriate or unwanted content using guardrails.

As you already understood, Amazon Bedrock aims to make it easy for you to leverage powerful foundation models and build innovative, secure, and responsible generative AI applications.

Furthermore, Amazon Bedrock offers buying provisions throughput (Chapter 18) to optimize modeling inference and has implemented guardrails (Chapter 8) for safeguarding the generative AI applications. It is important to mention that model evaluation and guardrails functionality are currently limited to some regions, subject to further development and refinement.



*Figure 3.1 Amazon Bedrock and advanced Bedrock features*

The purpose of this book is driving deep on Amazon Bedrock. This chapter onwards you will drive deep on Amazon Bedrock along with some generative AI based applications build on Amazon Bedrock. Even, you will drive deep on all the advanced topics of Amazon Bedrock in the subsequent chapter.

# **3.3 Set up Amazon Bedrock**

This section will provide you overview to access the Amazon Bedrock console and its playground.

#### **Console Access**

Prerequisites include that you are having an AWS account with appropriate IAM access. If you do not have an AWS account, please follow the necessary steps outlined here.

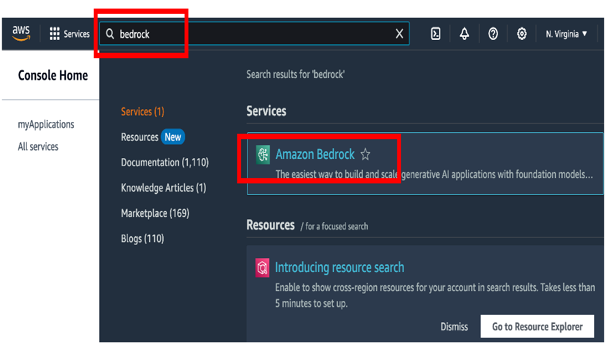
(Refer: <https://docs.aws.amazon.com/accounts/latest/reference/manage-acct-creating.html> )

Charges will be incurred for the hands-on exercise. This book will outline all the cost related disclaimer wherever required. You will need to have an attention all those disclaimers. Amazon Bedrock is available some of the AWS region.

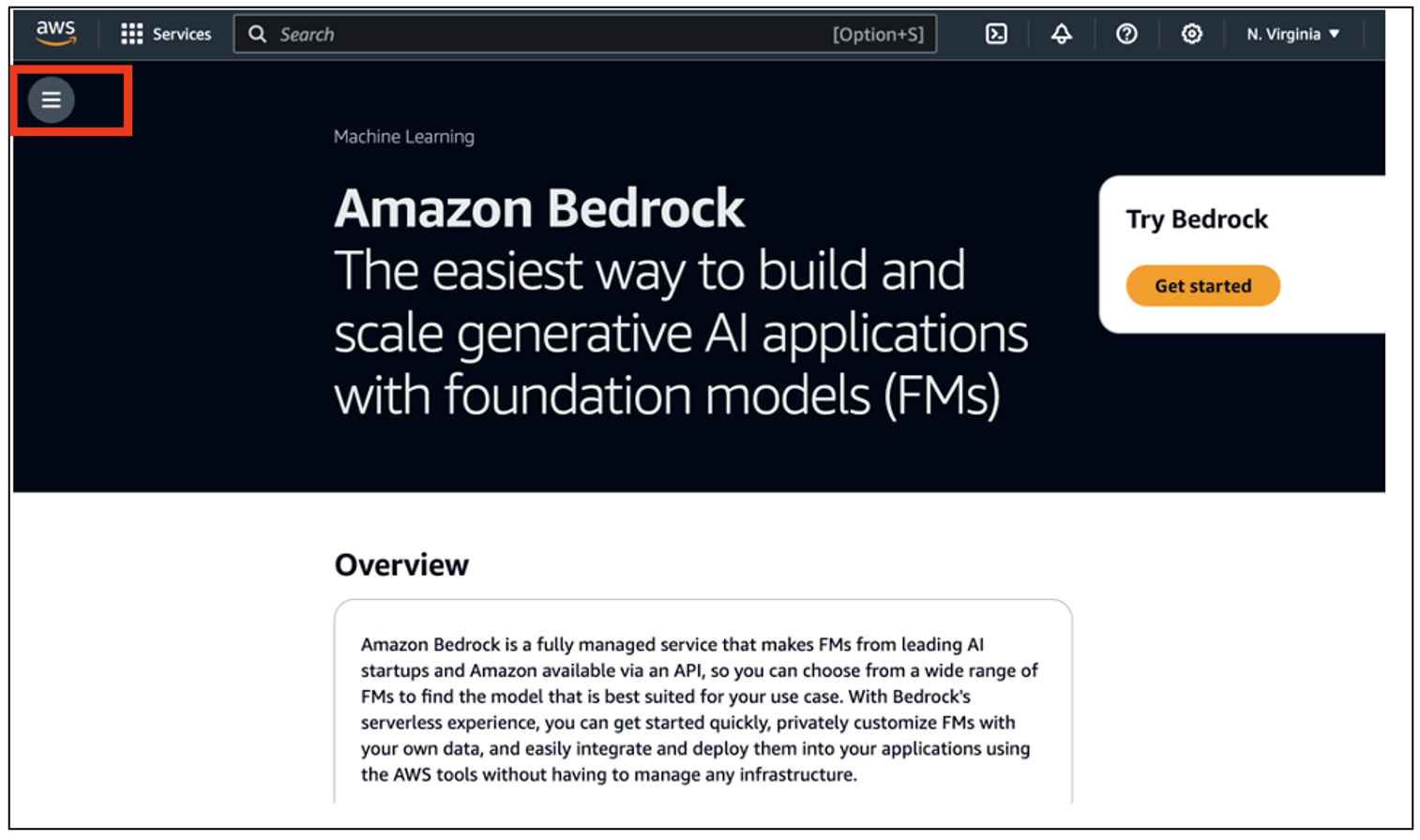
(Refer: <https://docs.aws.amazon.com/bedrock/latest/userguide/bedrock-regions.html> )

Request you to follow one region for performing all the hands-on exercise. This book will showcase us-east-1 (US East - N. Virginia) throughout the book for any of the exercises. But you can choose any region mentioned above based on Amazon Bedrock functionality and foundation models’ availability.

Log in to AWS Console. Make sure you are in right AWS region ( us-east-1). Search for Amazon Bedrock in the search option on AWS Console. Choose Amazon Bedrock. (figure 3-2)

*Figure 3-2 Navigating Amazon Bedrock at AWS console*

Click the hamburger icon (figure 3-3a) on the top left corner. You need to request access to models before they can be used. If you want to add additional models for text, chat, and image generation, you need to request access to those models in Amazon Bedrock. To do so, Click Model Access link in the left-side navigation panel of the Amazon Bedrock console. (figure 3-3b)



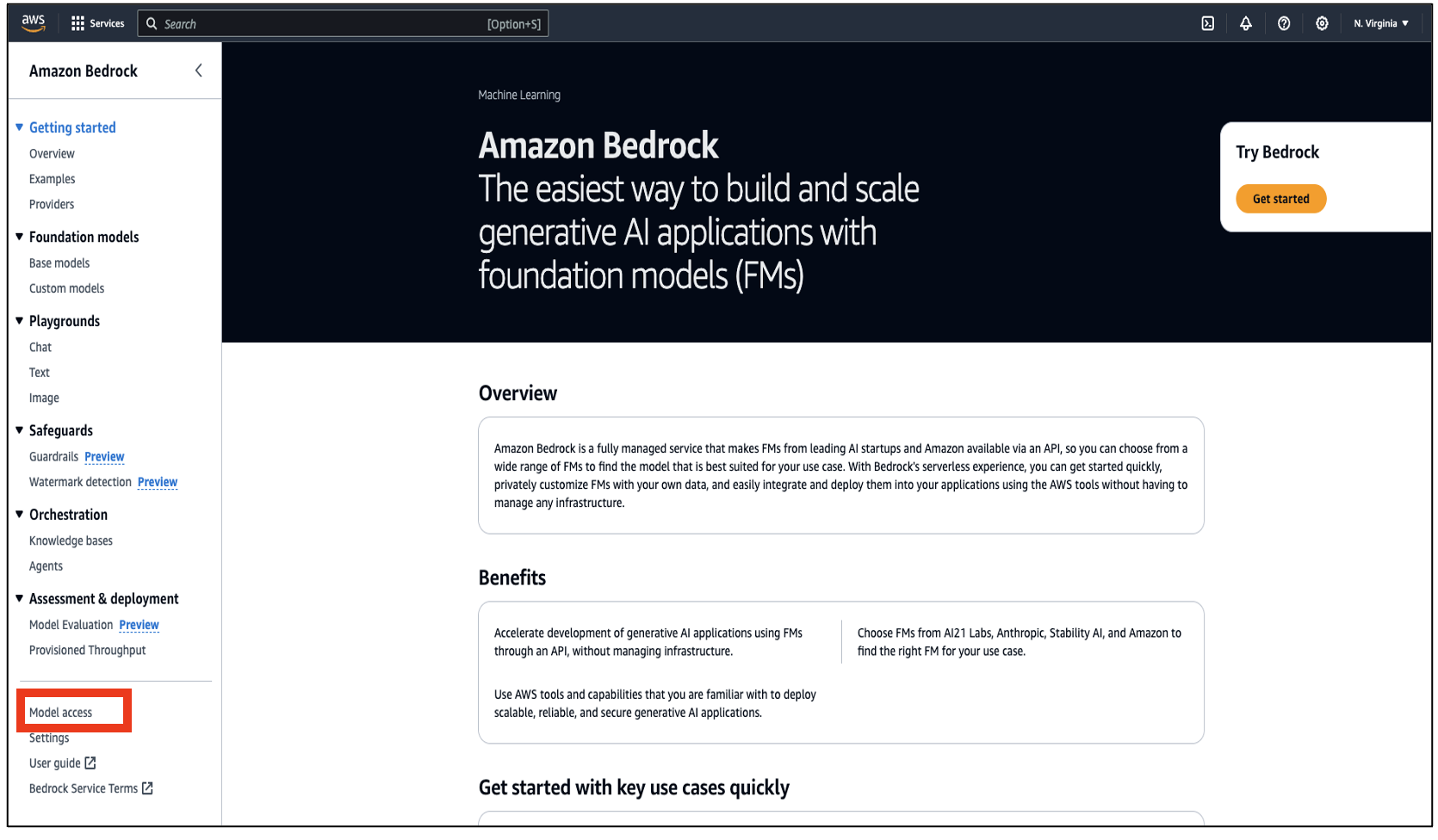
*Figure 3-3a Opening left side navigation panel of Amazon Bedrock*

By default, the account does not have access to any models. Admin users with the appropriate IAM access permissions can add access to specific models using the model access page. Once the admin adds access to models, those models become available for you and your account.

Charges are incurred when the models are used in Amazon Bedrock. You can review the **End User License Agreement** (EULA) for each model by selecting the corresponding link.

To add, edit or remove model access, select the Manage model access option.

(Refer: <https://docs.aws.amazon.com/bedrock/latest/userguide/model-access-modify.html> )

*Figure 3-3b Enable model access on Amazon Bedrock*

# **3.4 Foundation Models on Amazon Bedrock**

There are a lot of purpose-built foundation models available on Amazon Bedrock, with the possibility of additional models being added and existing models being upgraded in the future. You can refer this link to find AWS latest foundation models availability on amazon Bedrock. (Refer <https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html>)

The model provider has pretrained foundation models available on Amazon Bedrock. Key properties include the service provider, model family, model name, and model ID. It's also crucial to know the model's input and output modalities, version, maximum token limit, and relevant use cases. Understanding this knowledge is important. You should understand before beginning generative AI solution development.

**Amazon Bedrock**

**Broad choice of Foundation Models**

*Figure 3-4 Foundation models available on Amazon Bedrock*

#### **Model Provider**

The term model provider refers to the companies and organizations that offer their foundation (base) models through the Amazon Bedrock. These models typically cover a range of tasks such as text generation, image generation, question-answering, summarization, and more.

* Amazon Titan Models are trained by AWS. They can perform many tasks. These include text generation, translation, summarization, and image generation.
* Anthropic's Claude models are recognized for safe conversational AI. They also excel in text generation.
* Cohere specializes in large language models. Their specialty is on natural language understanding and processing.
* Stability AI is known for image generative models. They generate high-quality images from text prompts.

These models are accessible via the Bedrock API, enabling you to customize or adapt them to specific use cases without managing the complexity of ML model training, deployment, or scaling. (Refer: https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html)

#### **Model ID**

To access a model from Amazon Bedrock through an API call, you must provide the model ID. Each foundation model has a unique model ID. You will learn the difference between the base model and model IDs for provisioned throughput in the subsequent chapter. (Refer: <https://docs.aws.amazon.com/bedrock/latest/userguide/model-ids.html>)

#### **Model Family**

A model family consists of a group of foundation models on Amazon Bedrock. These models focus on similar tasks. They include text generation family, image generation family etc. Grouping models into families helps you to choose the right one. The selection depends on the specific problem or task for automation. Here are examples from the Amazon Bedrock model family.

Text Generation Models are used for various NLP tasks. These include content creation, summarization, translation, and question answering. Amazon Titan Text models cater to different text generation needs. Anthropic Claude Models specialize in conversational AI, focusing on accuracy and ethical responses.

Image Generation Models produce images from text descriptions. This feature is beneficial for graphic design, marketing, and content creation. Example models include stability AI’s stable diffusion is used to generate high-quality images based on text prompts.

Other model families include the embedded model family and the multimodal model family. In the following chapter, you will learn about these model family in detail.

#### **Model Version**

Model providers like Amazon Titan, Anthropic, Cohere, and Stability AI release versions of foundation models on Amazon Bedrock. Each model version is pretrained with particular datasets, architectures, and optimization methods. This process aims to improve performance, accuracy, and adaptability for specific tasks.

The model version is important. It helps providers make improvements while keeping compatibility. You check the model card on the Amazon Bedrock console for detailed information on each foundation model. Choosing the right model version in Amazon Bedrock lets you to align a model's performance with their application's needs. This ensures better results for tasks. (Refer: https://docs.aws.amazon.com/bedrock/latest/userguide/model-ids.html)

#### **Model input modalities**

Model input modalities are the various data types that foundation models can use as input for generative tasks. These can include text, images, or both, based on the model's design and architecture. This versatility allows for many applications, such as natural language processing and image generation. Some of the models, such as Amazon Titan Text Models and Anthropic Claude, support text as an input modality. On the other hand, Stability AI accepts images as an input modality. Furthermore, multimodal models can process more than one type of input simultaneously. For example, a model may take both text and images as input to generate either text or images as output, depending on the task. For example, the Anthropic Claude 3 Opus is a large multimodal. During the prompt, you can provide text and an image. Later in this book, you will learn about multimodal capabilities. (Chapter 19)

#### **Model output modalities**

The term model output modalities refer to the types of data that foundation models can generate or produce as output. Like input modalities, the nature of the model and its intended task determine the output modalities. The output can include text generation, image creation, or both. For example, you can use multimodal to make product catalogs. These catalogs can include branding statements based on product descriptions and images. Before using the models, check the model card to learn about the input and output options.

#### **Model Size**

Model size is crucial. It shows how many parameters a foundation model has. This size influences the model's abilities and performance. It also determines the resources required, like memory and computing power. Larger models have more parameters, allowing them to understand and create complex patterns. However, they demand more computational resources to operate.

The number of parameters is what a model learns during training. Models with more parameters are typically more powerful and can handle nuanced tasks like complex language generation. Larger models can generalize better, managing diverse inputs, but they may be slower and costlier to operate.

Choosing the right model size is crucial. Smaller models are faster and more efficient. They excel at simple tasks like classification and basic text generation. Larger models handle complex tasks better. These include long-form text generation and advanced natural language understanding. This book will help you select a model that fits your needs.

You can refer to the below table to understand some of the factors of the model from Amazon Titan and Anthropic model providers. But you should refer to the model card for details. You will learn how to check the model card in the next section.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Provider** | **Model Name** | **Input modalities** | **Output modalities** | **Token Size (Max)** | **Purpose** |
| Amazon | Titan Text G1 - Express | Text | Text, Chat | 8k | Text generation, Code generation, Instruction following, multilingual support, Rich text formatting, Orchestration (Agents), Fine Tuning |
| Amazon | Titan Text G1 - Lite | Text | Text | 4k | Text generation, Code generation, Rich text formatting, Orchestration (Agents), Fine Tuning |
| Amazon | Titan Image Generator G1 | Text, Image | Image | 77 | text to image generation, image editing,  Max image size: 25MB |
| Amazon | Titan Embeddings G1 - Text | Text | Embeddings | 8k | text retrieval, semantic similarity, clustering, multilingual support, Output vector size: 1,536 |
| Amazon | Titan Multimodal Embeddings G1 | Text, Image | Embeddings | 128 | Search and Recommendations on images Output vector size: 1,024 (default), 384, or 256, Max image size: 25MB |
| Anthropic | Claude 2 | Text | Text, Chat | 100k | Question answering, information extraction, removing PII, content generation, multiple choice classification, Roleplay, comparing text, summarization, document Q&A with citation, multilingual support |
| Anthropic | Claude 2.1 | Text | Text, Chat | 200k | Question answering, information extraction, removing PII, content generation, multiple choice classification, Roleplay, comparing text, summarization, document Q&A with citation, multilingual support |
| Anthropic | Claude Instant | Text | Text, Chat | 100k | Question answering, information extraction, removing PII, content generation, multiple choice classification, Roleplay, comparing text, summarization, document Q&A with citation, multilingual support |
| Anthropic | Claude 3 Sonnet | Text, Image | Text, Chat | 200k | Data processing: RAG, or the extensive search and retrieval of knowledge  Sales: product recommendations, forecasting, targeted marketing  Time-saving tasks: code generation, quality control, parse text from images, multilingual support |
| Anthropic | Claude 3 Haiku | Text, Image | Text, Chat | 200k | Customer interactions: live chat assistance that is prompt and accurate, translations  Content moderation: detect potentially behavior or customer requests  Cost-saving tasks: inventory control, efficient logistics, knowledge extraction from unstructured data, and multilingual assistance |
| Anthropic | Claude 3 Opus | Text, Image | Text, Chat | 200k | Task automation: interactive coding, arranging and carrying out intricate operations across databases and APIs  Research and Development: brainstorming and hypothesis generation, research review  Strategy: financials and market trends, forecasting, advanced analysis of charts & graphs |

*Table 3-1 Some example of model factors for Amazon titan and Anthropic*

You will learn each and every model in the subsequent chapter through an example.

# **3.5 Model Lifecycle**

Amazon Bedrock is focused on offering the latest foundation models that have better capabilities, accuracy, and safety. You can test models via the Amazon Bedrock console or API when new versions come out. This helps you update your apps with the latest features improvement. The foundation models on Amazon Bedrock have three states: active, legacy, or end-of-life (EOL).

Active models are the most recent versions that receive regular updates and bug fixes. Legacy models are older versions that have been replaced by better-performing ones. Amazon Bedrock will announce an End of Life (EOL) date for Legacy models, which can differ based on usage, like On-Demand or Provisioned Throughput. You should migrate to an Active version before the EOL date. Once a model reaches EOL, it will no longer be usable, and requests to it will fail. You can find the current status and EOL dates for Legacy models using Amazon Bedrock APIs, the console, and documentation.

For example, the legacy date for Titan Embeddings-Text v1.1 was November 7, 2023, with the EOL date being February 15, 2024. Even the recommended model version is Titan Embeddings-Text v1.2. This is an example for model provider like Amazon.

Another example, the legacy date for Claude v1.3 was November 28, 2023, with the EOL date being February 28, 2024. Even the recommended model version is Claude v2.1. This is an example for model provider like Anthropic, a third-party research organization.

(Refer <https://docs.aws.amazon.com/bedrock/latest/userguide/model-lifecycle.html>) You refer to the Amazon document for the latest information. Even, you can use two APIs like **GetFoundationModel** and **ListFoundationModels** to get relevant information through coding. You will learn to use both APIs in the example code of this chapter.

# **3.6 Amazon Bedrock Console Walkthrough**

The Amazon Bedrock console provides a comprehensive suite of features. These tools help you utilize advanced foundation models effectively. Let's take a closer look at the main features.

#### **Getting started**

You will see three subsections under Getting Started like Overview, Examples, and Providers. The Amazon Bedrock navigation pane includes a detailed introduction to foundation models in the Overview section. It also offers practical examples and interactive playgrounds in the Examples section. You can choose from various text classification prompts in the Examples section. These prompts include options for a variety of modality, such as text, image, or embedding, as well as model type, category, and provider. This will help you find relevant examples quickly through the filters provided. You can view the purpose, model name, prompt description, desired answer, inference configuration, and API request for each example. The open-in playground allows you to run an example instantly. (figure 3-5-1)

#### **Foundation models**

You will notice three subsections under the Foundation models section, which are base models, custom models, and import models respectively.

You can sort base models by various attributes for easier navigation in the Foundation models’ section. You can filter and search functions to find information about model providers. The platform allows you to customize base models for better performance on specific tasks. You can also create and manage custom models under the Foundation Models tab. This lets you tailor them to your needs using the available training datasets. For hands-on experimentation, you can utilize the console playgrounds to explore both base and custom models. Even, you can export custom models to host on Amazon Bedrock. (figure 3-5-2)

You will explore base models in this chapter. In Chapter 10, you will explore custom models with relevant examples.

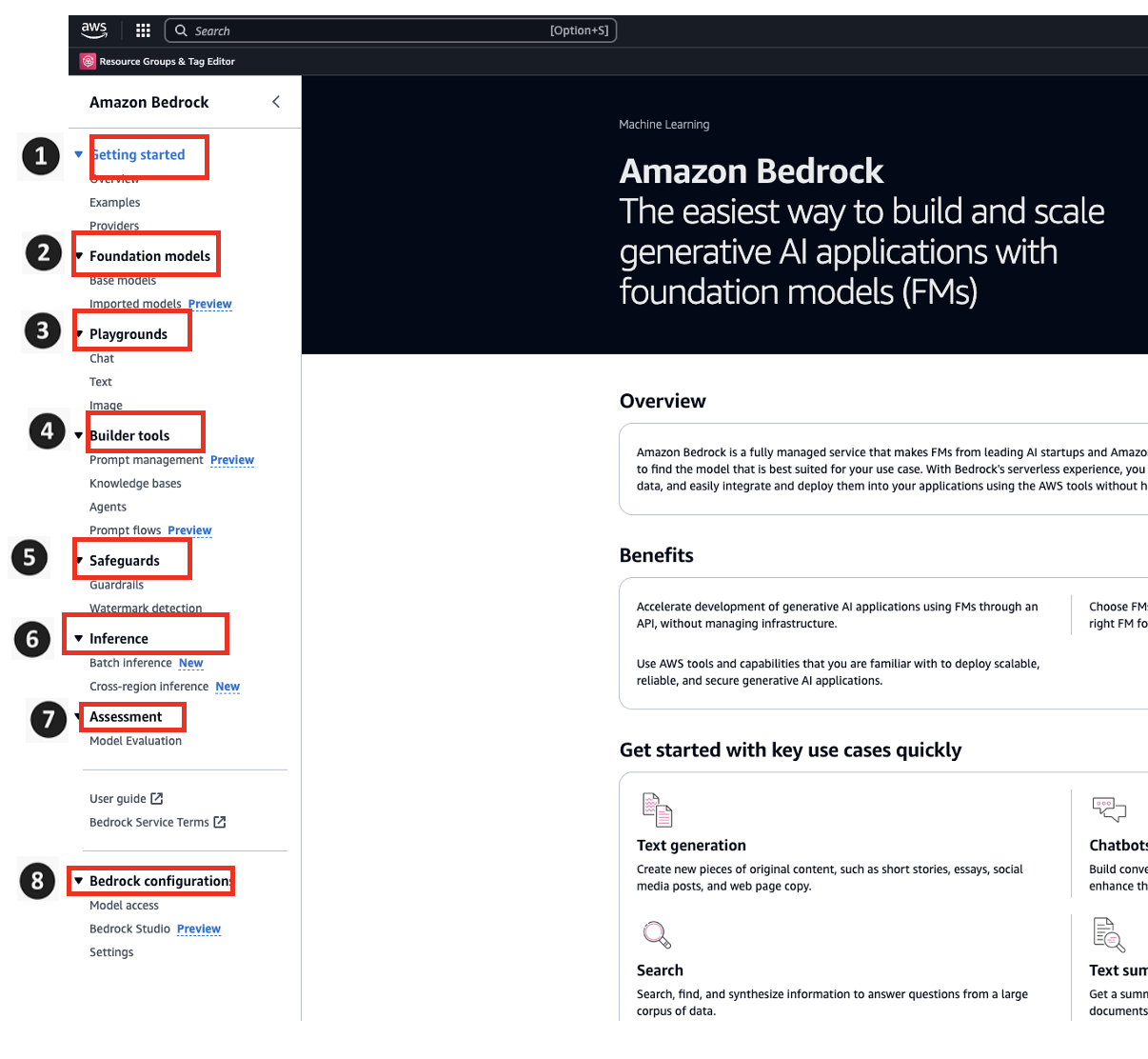
#### **Playgrounds**

You will see three subsections under Playgrounds like Chat, Text, Image.

The console Playgrounds within Amazon Bedrock offer you a space to experiment with various models before integrating them into applications. Divided into three categories, each playground serves a distinct purpose.

* The Chat section allows you to interact with language models through a chat interface. You can generate responses after selecting the model and view model metrics.
* The Text section allows you to interact with language models. You can generate responses after selecting the model and adjust the configuration.
* The Image section allows you to interact with image models. You can create images from text prompts you provide.

You can access these playgrounds in the console navigation pane under Playgrounds, enabling comprehensive testing and comparison of models before implementation. (figure 3-5-3) You will explore Playgrounds in this chapter.



*Figure 3-5 Amazon Bedrock console*

#### **Builder Tools**

You will see four subsections under Builder Tools, like Prompt management, Knowledge bases, Agents and Prompt flows.

Builder tools are crucial for creating and managing prompt flows, which are sequences of prompts and responses used in conversational AI systems. These will help you creating and managing prompts. They also allow you to preview your work and access knowledge bases. Additionally, they integrate with generative AI agents. Builder tools are useful for most of the persona. They enable the design and optimization of conversational experiences. You can test and refine prompt flows before launching them for production. (figure 3-5-4)

#### **Safeguards**

You will find two subsections under the Safeguards section, which include guardrails and watermark detection.

The console Safeguards in Amazon Bedrock introduce Guardrails for Amazon Bedrock. These guardrails empower users to implement customized safeguards in line with their application needs and responsible AI policies. This section offers a feature called Guardrails. It helps build responsible AI applications. These guardrails ensure that generative AI follows safety policies. They monitor both your input as prompt and responses from foundation models based on specific guidelines. You can set up multiple guardrails for different applications. You can also integrate guardrails with agents. This helps create compliant AI applications. The Titan Image Generator G1 adds an invisible watermark to all generated images. This feature helps verify the source of the images. You can find this in the console's Safeguards section. It ensures transparency and authenticity in image generation. (figure 3-5-5) You will explore Safeguards in Chapter 8.

#### **Inference**

You will see three subsections under Inference, like Provisioned Throughput bases, Batch inference and Cross-region inference.

Amazon Bedrock offers powerful inference capabilities for generative AI models. You can choose provisioned throughput-based inference, letting them specify the resources needed. It also supports batch inference, enabling you to process large datasets efficiently. Additionally, cross-region inference allows for simultaneous processing in multiple regions. (figure 3-5-6)

#### **Assessment**

You will see one subsection under Assessment, which is named Model Evaluation. To make the most of Amazon Bedrock models, it's important to assess their performance. The Model Evaluation feature helps you compare outputs effectively. This way, you can choose the best model for your needs. You will explore model evaluation in Chapter 11. (figure 3-5-7)

#### **Model access**

To utilize a model within Amazon Bedrock, the initial step involves requesting access to the desired model. To accomplish this, navigate to the left-hand side navigation pane and select Model Access under Bedrock Configuration. (figure 3-5-8)

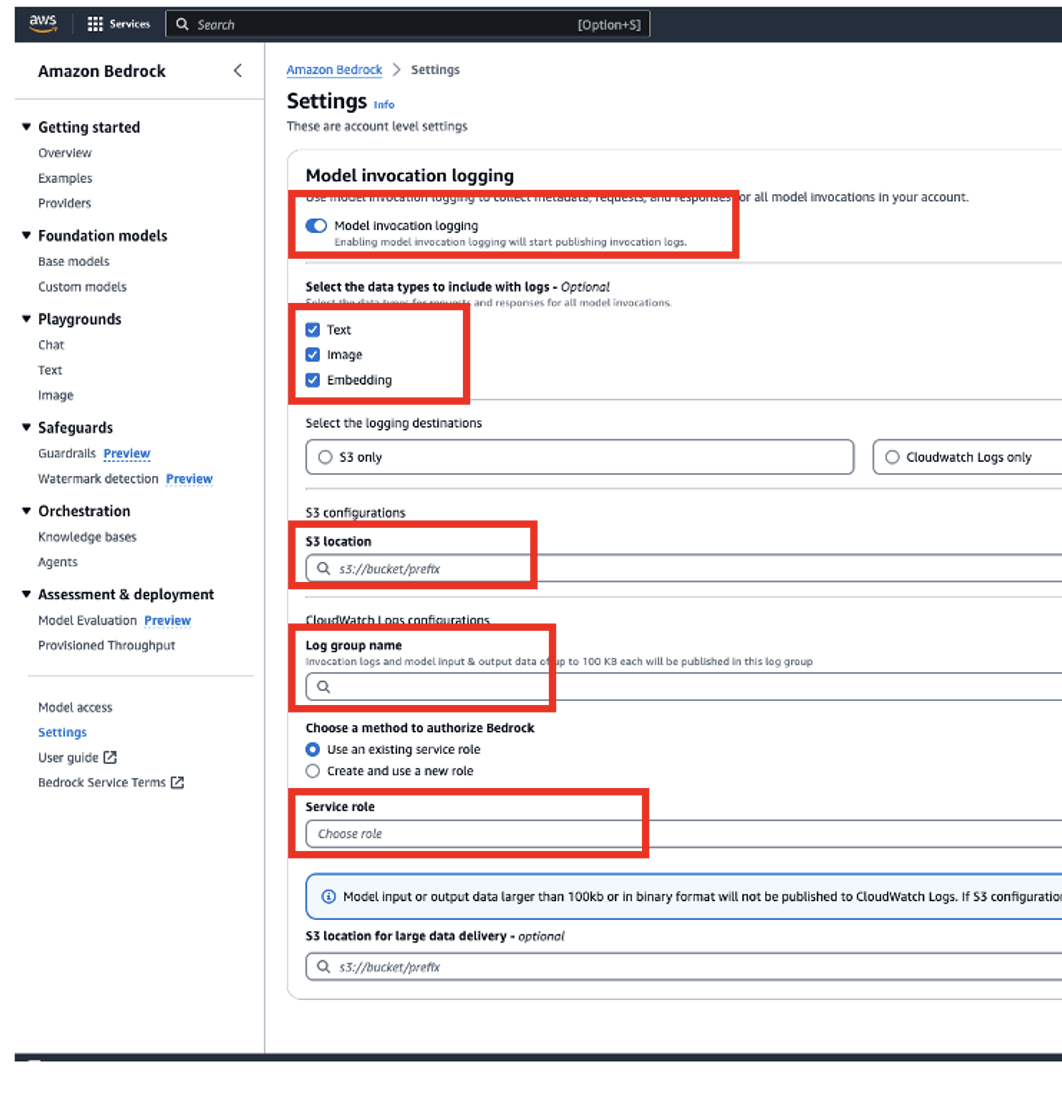
You have already learned how to edit and remove model access in section 3.2 and the corresponding subsection of Console Access of this chapter.

#### **Model invocation logging**

You can effectively monitor model invocation events by accessing the Settings option in the left navigation pane. Enable model invocation logging and select the data types to include, such as Text, Image, and Embeddings. Choose the logging destinations, including S3 only, CloudWatch Logs only, or both S3 and CloudWatch Logs, and configure S3 locations and Log group names for Amazon CloudWatch logs with appropriate service role access. (figure 3-5-9) (figure 3-6)

You should enable model invocation logging to have better observability and explainability during any hands-on exercise.

Charges are incurred from Amazon Simple Storage Service(S3) storing of logs of models’ execution at Amazon Bedrock. (Refer https://aws.amazon.com/pm/serv-s3/)

**

*Figure 3-6 Amazon Bedrock console model invocation logging*

# **3.7 Running Model Inferences**

Inference involves generating an output based on an input provided to a model. The bedrock Amazon enables you to execute foundation models of your choice. You can give the following inputs while running inference:

* **Instruction**: The input provided for the model to generate a response(prompt). You will study in detail about prompt in Chapter 4.
* **Parameters for inference**: These values can be adjusted to restrict, or influence how this model responds. You will investigate prompt further below.

There are different ways you can do model inference:

* Utilize any of the Playgrounds to execute inference in a user-friendly graphical interface.
* Send an API request.
* Prepare a dataset of prompts with desired configurations and conduct batch inference.
* Establish an agent and send the request to the API.

Inference can be done using base models, custom models or provisioned models. Provisioned Throughput is required for custom models. You will learn about Provisioned Throughput in Chapter 18. When investigating model answers from various prompts as well as inference parameters, there is a possibility of trying these tricks. Once familiar with these methods, integrate them into your application by calling the respective APIs. All the above methods you will explore in subsequent chapter.

#### **Inference parameters**

Inference parameters are adjustable values used to control or influence the model's response. The following categories of parameters are commonly present across different models. Inference parameters are not hyperparameter.

Among other things, hyperparameters are an important aspect of machine learning and artificial intelligence. These parameters control aspects of the learning algorithm and have a significant impact on the performance of the model by optimizing model weightage with minimizing loss functions. Though, you will learn some of this hyperparameters at Chapter 10. But some overview of most common hyperparameters are as follows.

* **Learning Rate**: This determines how much the model's parameters are adjusted during training in response to the estimated error each time the model weights are updated.
* **Number of layers**: This tells you how deep the neural network is and its ability to learn complex patterns.
* **Batch size**: The amount of training examples utilized for one iteration of training.
* **Dropout rate**: It is the probability that a neuron will be dropped during training to avoid overfitting.
* **Maximum sequence length**: Maximum number of tokens (words or sub-words) in an input sequence.
* **Number of attention heads**: This determines how attention is computed and distributed across different parts of the input sequence in transformer-based models.

However, inference parameters are not pre-set before the learning process but rather after the training of the model and use during inference or prediction phase. These are qualities that the model is trained on which affect how predictions are made, but they cannot be directly adjusted during training. It would not alter the base model’s behavior.

As a result, hyperparameters define the architecture and learning process of a model prior to training while inference parameters define the generating of text or images during inference and are obtained from a trained model. You will explore most common inference parameters below.

#### **Randomness and diversity**

You can see how a model assigns probabilities to possible next words by examining the probability distribution over them for each token in a sequence as an output from decoder in transformer architecture. To produce each token in the output, the model selects from this distribution. Randomness and diversity pertain to the degree of variation in a model's response. You can manage these aspects by constraining or adjusting the distribution. Foundation models commonly offer the following parameters to regulate randomness and diversity in the response.

#### **Temperature**

The temperature is the one that determines how much random the generated text would be. It works by determining the probabilities of the predicted outputs and therefore selecting either high, moderate or low probabilities depending on the value you have provided. Temperature is an element of probability mass function for the next token. On lower temperatures, function’s steepness increases making responses more deterministic while higher ones make it flatter leading to more randomness.

For instance, let us consider a prompt “**Kids love to eat**” and you want to complete this sentence leveraging generative AI. Suppose model is generating a probability distribution for the following words as a next token: **{“pizza": 0.5, "vegetables": 0.3, "ice cream": 0.2}**

If other inference parameters are held constant but temperature is set high, then this distribution will become flatter and as a result model will choose ice cream because it is less possible output than pizza that has got higher probability in such case model response would be “**Kids love to eat ice cream**.”.

If the temperature is set low, other inferential parameters fixed, distribution would be steeper. The probability of the model choosing pizza as the best guess (the output token with higher probability) would increase while that of selecting ice cream (the output token with lower probability) will reduce. So, the answer from the model will be “**Kids love to eat pizza.**”.

Let drive deep an additional example in the same topic. When the temperature is set low such as 0.1, then it makes conservative prediction based on high certainty by the model. This leads to more deterministic and cautious outputs with the model choosing high-likelihood tokens. For example, “**The pet dog generally sat on couch at the room.**”

On the other hand, setting temperature value like 0.6 allows for randomness in equal measures as conservatism. The model produces outputs that have some variance by exploring a broader range of token choices. For Example, “**The pet dog generally sat anywhere at the room.**”

High temperatures, for instance 1.0 or above improve randomness in generated texts by a large extent. In this case, there are more chances that the model would select those tokens which have lower probabilities leading to quite diverse and sometimes meaningless results. “**The pet dog generally lounged lazily everywhere at the room.**” The output could vary based on your model selection and other inference parameters value.

It is worth noting that optimal temperature may vary depending on what you intend to achieve with your output or based on how it should look like in specific use cases. Different values of temperature can be tried out during application development before selecting appropriate ones for its implementation.

#### **Top K**

Top K represents a sampling strategy selects the Top K most likely next tokens according to their probabilities and samples from this reduced set. This means that text generated is not too diverse, while still having a reasonable chance of the generated structures. For instance, when Top K is 20, the model can pick from any of the most probable twenty tokens with highest probability for a next token.

Let's consider same example prompt “**Kids love to eat**” like previous inference parameter. Suppose model is generating a probability distribution for the following words as a next token:

**{“pizza": 0.5, "vegetables": 0.3, "ice cream": 0.2}**

Here are how changes made on inference parameters would affect output. Let us say that if Top K equals to 2 then only top 2 candidates’ pizza and vegetables would be considered by model as possible next token. Because of this, ice cream could be excluded even though it had some probability.

The Top K would have a lower value to reduce the size of the candidate pool and prioritize output that are more likely. On the other hand, Top K has a higher value that expands the options to include more probable next token.

#### **Top P**

Top P, nucleus sampling, represents a sampling strategy selects the top P most likely next tokens according to their cumulative highest probabilities and samples from this reduced set. This approach helps in controlling the diversity of generated text while still ensuring the likelihood of the generated sequences. For instance, selecting a value of 30 for Top P allows the model to choose from the 30 percent most probable tokens for the next token.

Let's consider same example prompt “**Kids love to eat**” like previous inference parameter. Suppose model is generating a probability distribution for the following words as a next token:

**{“pizza": 0.5, "vegetables": 0.3, "ice cream": 0.2}**

Here are how changes made on inference parameters would affect output. If we set Top P at 50 percent, only top cumulative 50% most possible candidates will be considered by this model which is pizza. Thus, leaving out vegetables and ice cream despite having non-zero probability.

When there is a lower value of Top P to narrow down on most likely output tokens in favor of narrowing down candidate pool; conversely when there is a higher value of Top P to open on less likely output tokens.

To sum up, temperature changes the overall form of probability distribution while limiting model’s choices within pools of candidate tokens through Top K and Top P with bias towards high-probability outcomes.

#### **Combined Impact of Inference Parameters**

For instance, let us consider a prompt **Kids**. let's explore combined impact of inference parameters like temperature, Top P, and Top K.

Kids as the prompt and a high temperature (~ 1.5), the model might generate “**Kids flying through space on rainbow unicorns**." where the inclusion of "rainbow unicorns" is less likely but adds diversity.

In contrast, the model tends to select the most frequent words with low temperature (~ 0.2), thus generating more predictable and at times even more coherent text. For example, a typical output with low temperature might be “**Kids playing in the park**.”.

Again, with a Top P (~ 0.8) the model tends from a subset of tokens whose cumulative probability exceeds a threshold P. It might generate "**Kids playing outside with their friends**." where "playing outside" and "with their friends" might within the Top P subset of likely continuations prompt the model with Kids.

Again, with a Top K (~ 5) the model only considers the Top K most probable tokens at each step. It might generate "**Kids playing outside with their friends**." where "playing outside" and "with their friends" might within the Top P subset of likely continuations prompt the model with Kids.

It might generate "**Kids playing soccer in the backyard**." where "playing", "soccer", "in", "the" and "backyard" are among the top five likely words to follow "Kids” and prompt the model with "Kids”.

In summary, these techniques offer different ways to balance diversity and coherence in generated text, and their impact can vary depending on the specific context and parameters chosen.

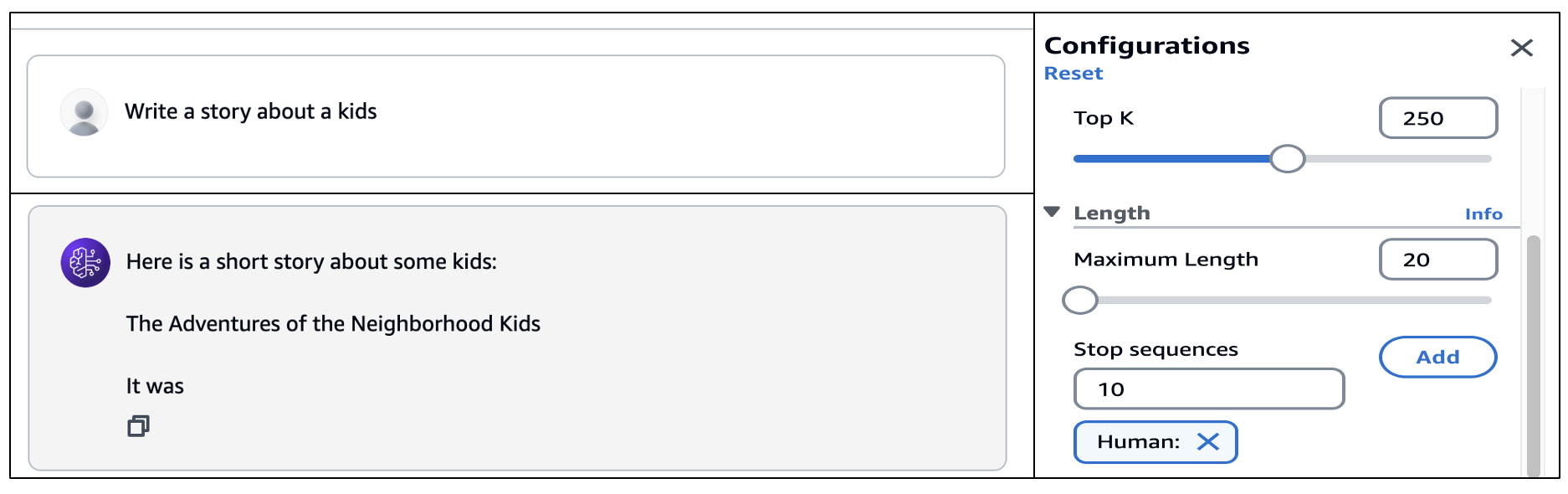
#### **Length**

Foundation models commonly offer parameters to regulate the length and characteristics of the generated responses. Some examples of such parameters include:

* **Response Length:** This allows specifying the exact minimum or maximum number of tokens (words or sub-words) to be included in the generated response.
* **Penalties** are thus used to discourage or penalize certain aspects of the response. These penalties can be imposed for examples like response length (either too short or too long), repeated tokens, high frequency of certain tokens, diversity of token types used.
* **Stop Sequences** allow you to define character sequences that would cause the model to stop generating more tokens when it encounters them. When any given stopping sequence is generated by the model, it will not add any further tokens after that point.

As such, these parameters can be configured to suit your preferences and requirements to allow better align with your interests and specific needs when using foundation models in generating texts.

This is a picture of playground with length configuration at Amazon Bedrock with a prompt “**Write a story about the kids**.”. The Maximum Length is 20 and Stop sequences is 10. Though, you can watch that the response started generating. But generation is incomplete due to Stop sequences parameter. (figure 3.7)



*Figure 3.7 Amazon Bedrock console playground length configurations*

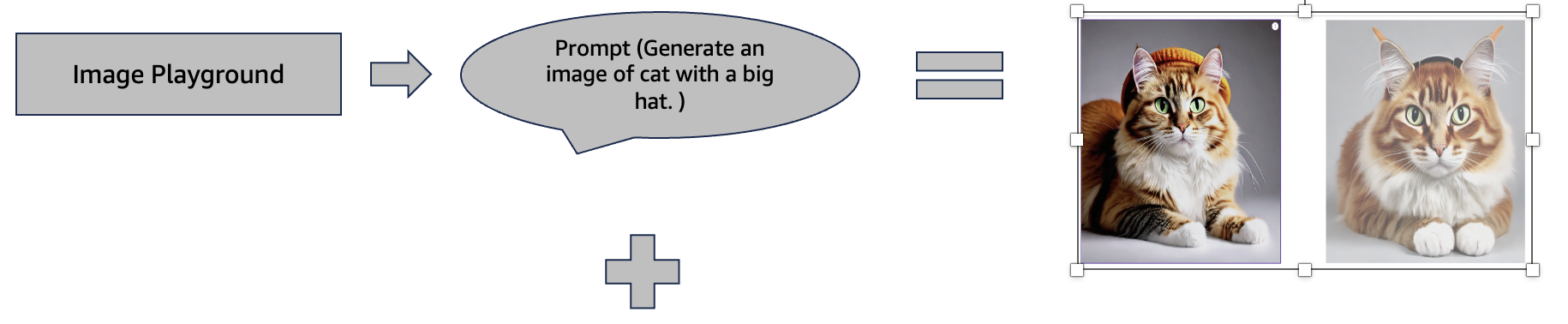
#### **Image playground**

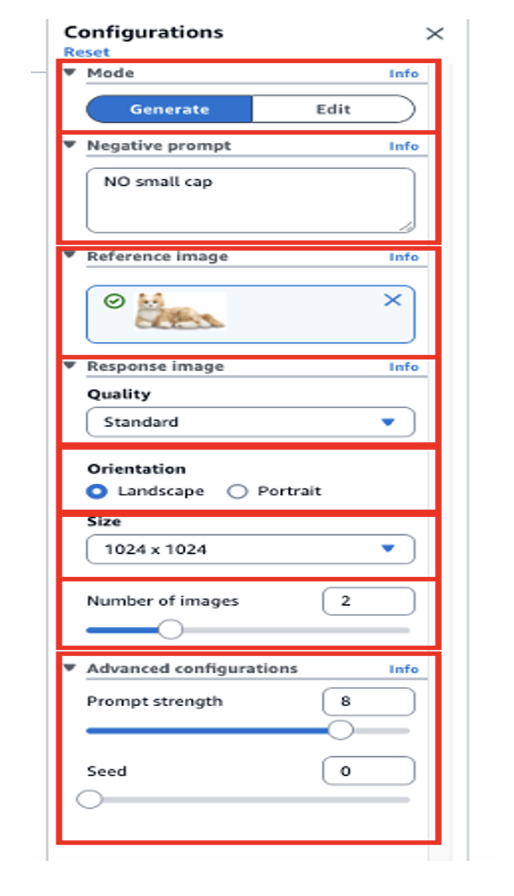
The Amazon Bedrock’s Image Playground is a platform for playing around with image models. By providing it with text prompt, it shows the model’s generated image. It also allows you to configure different settings that are specific to each of them.

You can specify Mode whether the model should recreate an entirely new image or adjust existing one by using a reference. It is possible to provide Negative prompt which would contain a list of items and concepts that the model should not generate. This may include things like cartoons or violence for instance. Reference image gives you as an option to select an image that the model will respond or edit. For any response image, quality, orientation, size and number could be chosen.

Advanced options such as Standard Orientation (Landscape/Portrait), Image Size, Number of Images, Prompt Strength and Seed can be specified for the Model.

You can see the picture of Amazon Bedrock Image Playground with a prompt “**Generate an image of cat with a big hat**.” with different configure settings like Generate Mode, Negative prompt, Reference Image, Response image, Orientation, Size, Number of images, and advanced configurations. (figure 3-8)





*Figure 3.8 Amazon Bedrock Image Playground*

# **3.8 Amazon SageMaker and Amazon Bedrock Interaction**

To get the GitHub details, refer to the appendix section of this book. In GitHub, locate the repository named **genai-bedrock-book-samples** and click on it.

Inside the **genai-bedrock-book-samples** repository is an AWS CloudFormation template that resides in the **cloudformation** folder. The task requires the execution of an AWS CloudFormation template, which should be performed **once** for all exercises in this book. A detailed guidance on how to manually execute the AWS CloudFormation template can be found in a file called **README** located within a directory named **cloudformation**. For more information about AWS CloudFormation template refer <https://aws.amazon.com/cloudformation/>.

**Disclaimer**: It is advisable to delete the AWS CloudFormation template if you are not actively participating in any exercises for some longer duration. Clear instructions for deleting the AWS CloudFormation template are provided within the README file itself.

However, in the **genai-bedrock-book-samples** folder there’s another subfolder titled **chapter3**. The **README** file within **chapter3** folder provides clear instructions on launching a **Notebook** on Amazon SageMaker.

|  |  |
| --- | --- |
| **File Name** | **File Description** |
| simple\_sagemaker\_bedrock.ipynb | 1. Understanding Amazon Bedrock client and Amazon Bedrock runtime client. 2. Understanding of list\_foundation\_models API. 3. Example of Amazon Titan LLM foundation model with and without parameters. 4. Example of Anthropic LLM foundation model with and without parameters. 5. Example of Amazon Titan Image foundation model with and without parameters. 6. Example of Amazon Titan LLM foundation model with streaming API with and without parameters.   **Dependency**:  NA |
| simple\_bedrock\_application.py | 1. Understanding Amazon Bedrock client and simple Streamlit application. 2. Example of Amazon Titan LLM foundation model with parameters. 3. Example of Anthropic LLM foundation model with parameters.   **Dependency**:  NA |

# 3.8 Bedrock Interaction Sample Application

**Disclaimer**: Charges will apply upon executing above files. Therefore, it is important not to forget to clean up the kernel after studying the topic. Refer to the clean-up section for instructions on how to properly clean up the kernel.

# **3.9 Summary**

This chapter is a detailed introduction to Amazon Bedrock, which is a fully managed service designed for hosting foundation models. You should get the overview of Amazon Bedrock concept and key features, setting up the environment, looking at foundation models, model life cycle stages, working with Amazon Bedrock console, testing models and understanding how it interacts with Amazon SageMaker.

Amazon Bedrock provides a single-entry point to both foundational AI models from top startups and Amazon that can be customized by you. It also allows you to experiment with prompts, augment response generation, build agents, modify models as well as test various models. Getting started on setting it up entails accessing console, meeting certain requirements’ criteria and seeking permission to use the already trained models. This chapter then looks at different foundation models on Amazon Bedrock that includes details such as input/output modalities of these foundations’ models and token sizes.

The model lifecycle section covers the states in which foundational models exist (Active/Legacy/End-of-Life) and how applications can be migrated from one version to another. Some of the showcased sections on this platform include Getting Started page, Foundation Models, Playgrounds and Safeguard.