A Definitive Guide to Generative AI with Amazon Bedrock

# Chapter 3: Introduction to Amazon Bedrock

# 3.1 What is Amazon Bedrock

Amazon Bedrock is a fully managed service designed to facilitate the access of high-performing foundation models (FMs) sourced from leading AI startups and Amazon. It allows customers through single API access to a variety of foundational models specially designed for their specific use cases. Also, Amazon Bedrock provides rich features for generative AI applications development with focus on security, privacy and responsible AI practices.

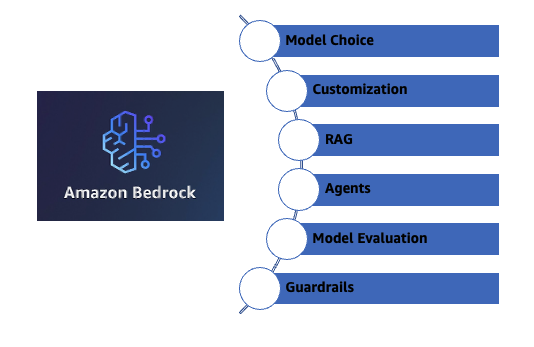
Through Amazon Bedrock users are able to privately tweak and try out different foundation models in techniques such as fine-tuning and Retrieval Augmented Generation (RAG), which can seamlessly be integrated into their enterprise systems and data sources. The serverless nature of Amazon Bedrock ensures instant start-up, model customization in private using proprietary data, deployment through AWS tools easily without managing any infrastructure.

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 that can reason through and carry out tasks for customers
* 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

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

Furthermore, Amazon Bedrock offers buying Provisions Throughput so as to optimize modeling inference and also has implemented Guardrails for protection against malicious or evil contents in the generative AI apps. It is important to mention that model evaluation and Guardrails functionality are currently in preview or limited preview release, as of April 2024, 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 topic of Amazon Bedrock in the subsequent chapter.

# 3.2 Set up Amazon Bedrock

This section will outline the steps to access Amazon Bedrock console and its playground.

# **Console Access**

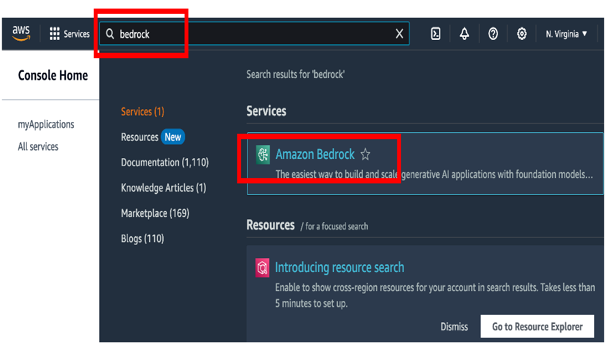
Prerequisites include the user having an AWS account with appropriate IAM access. If you do not have an AWS account, please follow the necessary steps outlined here. ( Refer: <https://docs.aws.amazon.com/accounts/latest/reference/manage-acct-creating.html> )

Charges are incurred for the hands-on exercise. The book will outline all the cost related disclaimer wherever required. You will need to have an attention all those disclaimer.

As of April, 2024 Amazon Bedrock is available some of AWS region. ( Refer: <https://docs.aws.amazon.com/bedrock/latest/userguide/bedrock-regions.html> )

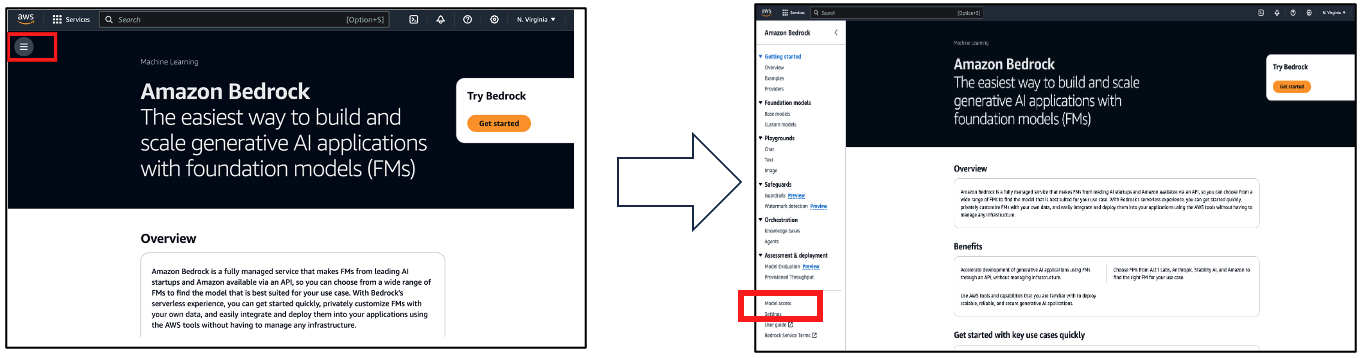
You should follow one region for performing all the hands-on exercise purpose. Book will showcase **us-east-1** ( **US East - N. Virginia** ) throughout the book. But, you can choose any region mentioned above based on functionality available.

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 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 3.3*)



*Figure 3.3 Enable Model Access on 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 all users of the 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.

# 3.3 Foundation Model on Amazon Bedrock

As of April 2024, these are the foundation models available on Amazon Bedrock, with the possibility of additional models being added and existing models being upgraded in the future. ( Refer <https://docs.aws.amazon.com/bedrock/latest/userguide/models-supported.html> )

Amazon Bedrock

Broad choice of Large Language Model

*Figure 3.4 Foundation Model available on Amazon Bedrock*

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **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 |
| AI21labs | Jurassic-2 Mid | Text | Text, Chat | 8191 | Open-book question response, summary, drafting, information extraction, brainstorming, and multilingual support |
| AI21labs | Jurassic-2 Ultra | Text | Text, Chat | 8191 | Open-book question response, summary, drafting, information extraction, brainstorming, and multilingual support |
| cohere | Command | Text | Text | 4k | Summarization, copywriting, dialogue, extraction, and question answering |
| cohere | Command Light | Text | Text | 4k | Summarization, copywriting, dialogue, extraction, and question answering |
| cohere | Embed English | Text | Embeddings | 512 | Semantic search, retrieval-augmented generation (RAG), classification, clustering,  Model attributes: 1024 dimensions |
| cohere | Embed Multilingual | Text | Embeddings | 512 | Semantic search, retrieval-augmented generation (RAG), classification, clustering,  Model attributes: 1024 dimensions, multilingual support |
| Meta | Llama 2 Chat 13B | Text | Text, Chat | 4096 | This is intended for commercial and research use in English. Fine-tuned chat models for chat applications. |
| Meta | Llama 2 Chat 70B | Text | Text, Chat | 4096 | This is intended for commercial and research use in English. Fine-tuned chat models for chat applications. |
| Meta | Llama 2 13B | Text | Text | 4096 | This is intended for commercial and research use in English. Fine-tuned chat models for chat applications. |
| Meta | Llama 2 70B | Text | Text | 4096 | This is intended for commercial and research use in English. Fine-tuned chat models for chat applications. |
| Mistral AI | Mistral 7B Instruct | Text | Text | 32k | Classification, Text generation, Code generation |
| Mistral AI | Mixtral 8X7B Instruct | Text | Text | 32k | Complex analysis and reasoning, Text generation, Code generation, multilingual support |
| Mistral AI | Mistral Large | Text | Text | 32K | Complex analysis and reasoning, Text generation, Code generation, RAG, Agents, multilingual support |
| stability.ai | Stable Diffusion XL | Text, Image | Image | 77 | Generation of art work, creative tooling, educational applications |

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

# 3.4 Model lifecycle

Amazon Bedrock is continually working to provide the latest versions of foundation models with improved capabilities, accuracy, and safety. You are allowed to test them via the Amazon Bedrock console or API as new model versions emerge, and this way you can transfer their apps in order to take advantage of the newest enhancements. Foundation models on Amazon Bedrock can have one of three states: Active, Legacy, or End-of-Life (EOL).

* Active models are the latest versions, which the providers are actively maintaining and updating, such as with bug fixes and minor improvements.
* Legacy models are previous versions that have been superseded by newer versions with superior performance.
* Amazon Bedrock will set an EOL date for Legacy models, which may vary depending on the specific model and how the user is utilizing it (e.g., On-Demand vs. Provisioned Throughput). You should plan to migrate to an Active version before the EOL date. EOL models are no longer available for use, and any requests to these versions will fail. You can check the current state of a model, including any EOL dates for Legacy versions, through the Amazon Bedrock APIs, console, and documentation.

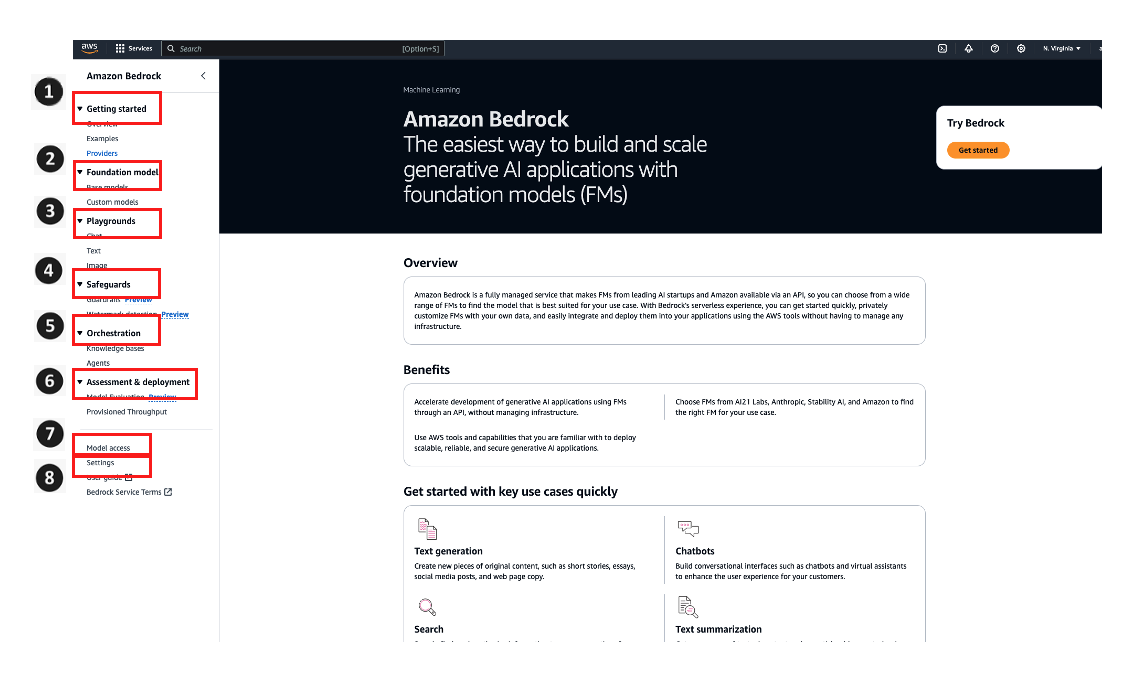
For Example, the legacy date for Titan Embeddings - Text v1.1 was November 7, 2023 with EOL date was February 15, 2024. Even, recommended model version is Titan Embeddings - Text v1.2. This is an example for model provider Amazon.

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

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

# 3.5 Amazon Bedrock console walk through

The Amazon Bedrock console offers a comprehensive suite of features to help you harness the power of state-of-the-art foundation models. Let's explore the overview of the key .



*Figure 3.5 Amazon Bedrock console*

# **Getting started**

You will see three subsections under **Getting Started** like Overview, Examples, Providers.

The navigation pane for **Amazon Bedrock** has an elaborate introduction to the foundation models in the **Getting Started** section as well as several practical examples and interactive playgrounds. On examples page, users can select from a variety of text classification prompts such as the modality (text, image or embedding), model type, category and provider they are searching for. This will help you find relevant examples quickly through filters provided. For each example, it is also possible to see its purpose, model name behind it, prompt description, desired answer required by this example along with inference configuration and API request for that sample. **Open in playground** enables you to run an example instantly. ( *Figure 3.5 – 1* )

# **Foundation models**

You will see two subsections under **Foundation models** like Base models, Custom models.

The **Foundation models** section provides users with access to Base models, which can be sorted by various attributes for easier navigation. You can also utilize filters, search functions, and access information about model providers. Additionally, the platform offers the option to customize Base models to enhance performance on specific tasks or teach them new domains of knowledge. Custom models can be created and managed under the Foundation models tab, allowing users to tailor models to their unique needs using provided training datasets. For hands-on experimentation, you can utilize the console playgrounds to explore both Base and Custom models. ( *Figure 3.5 – 2* )

You will explore Base models in this chapter. Whereas, Custom models exploration in the chapter 10 with relevant example.

# **Playgrounds**

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

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

* The **Chat** playground allows users to interact with chat models, generating responses and providing model metrics.
* The **Text** playground enables experimentation with text models, showcasing generated text based on prompts.
* Similarly, the **Image** playground facilitates experimentation with image models, generating images from provided text prompts.

Users 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.

# **Safeguards**

You will see two subsections under **Safeguards** like Guardrails and Watermark detection.

The console **Safeguards** in **Amazon Bedrock** introduce **Guardrails** for Amazon Bedrock, currently in **Preview** mode. These guardrails empower users to implement customized safeguards in line with their application needs and responsible AI policies. By leveraging Guardrails, users can construct responsible AI applications by applying tailored protections to foundation models (FMs) and agents within Amazon Bedrock. This ensures adherence to Responsible AI policies within Generative AI applications, maintaining a consistent level of AI safety across all applications. Guardrails evaluate user inputs and FM responses based on specific use case policies, applicable to various FMs and fine-tuned models, such as Anthropic Claude, Meta Llama 2, Cohere Command, AI21 Labs Jurassic, Mistral and Amazon Titan Text. You have the flexibility to create multiple guardrails with distinct control configurations, adaptable across diverse applications and use cases. Additionally, Guardrails seamlessly integrate with Agents for Amazon Bedrock, facilitating the development of generative AI applications that comply with responsible AI guidelines. Moreover, the Titan Image Generator G1 feature automatically embeds an **invisible watermark** in all generated images, aiding in watermark detection to verify the image's origin. Accessible through the console's Safeguards section, this ensures transparency and authenticity in image generation processes. ( *Figure 3.5 – 4* )

You will explore **Safeguards** in chapter 8. Both, Guardrails and Watermark detection are in preview mode as of April, 2024.

# **Orchestration**

You will see two subsections under **Orchestration** like Knowledge bases and Agents.

The console **Orchestration** in **Amazon Bedrock** empowers users to implement advanced workflows like Retrieval-Augmented Generation (RAG), leveraging knowledge bases to create contextually aware applications utilizing Large Language Models (LLMs). By accessing the **Orchestration** section and selecting **Knowledge Base**, users can integrate knowledge bases seamlessly. Furthermore, with Agents for Amazon Bedrock, developers can configure agents to execute tasks based on organizational data and user input, such as fulfilling customer requests. You can access **Agents** through the Orchestration section, allowing for efficient deployment of task-driven agents. (*Figure 3.5 – 5* )

You will explore Retrieval-Augmented Generation (RAG) in chapter 6. Whereas, Knowledge bases at chapter 7 and Agents at chapter 9.

# **Assessment and deployment**

You will see two subsections under **Assessment and Deployment** like Model Evaluation and Provisioned Throughput.

To optimize your utilization of Amazon Bedrock models, it's crucial to evaluate their performance and seamlessly integrate them into your solutions. The Model Evaluation feature facilitates thorough assessment and comparison of model outputs, enabling you to select the most suitable one for your applications. Simply navigate to **Assessment and Deployment** and select **Model Evaluation** to begin. Additionally, configuring Provisioned Throughput offers a fixed-cost solution for receiving consistent throughput levels. Access this feature through the navigation pane under **Assessment and Deployment** then select **Provisioned Throughput**. For detailed guidance, refer to the documentation on Provisioned Throughput. ( *Figure 3.5 – 6* )

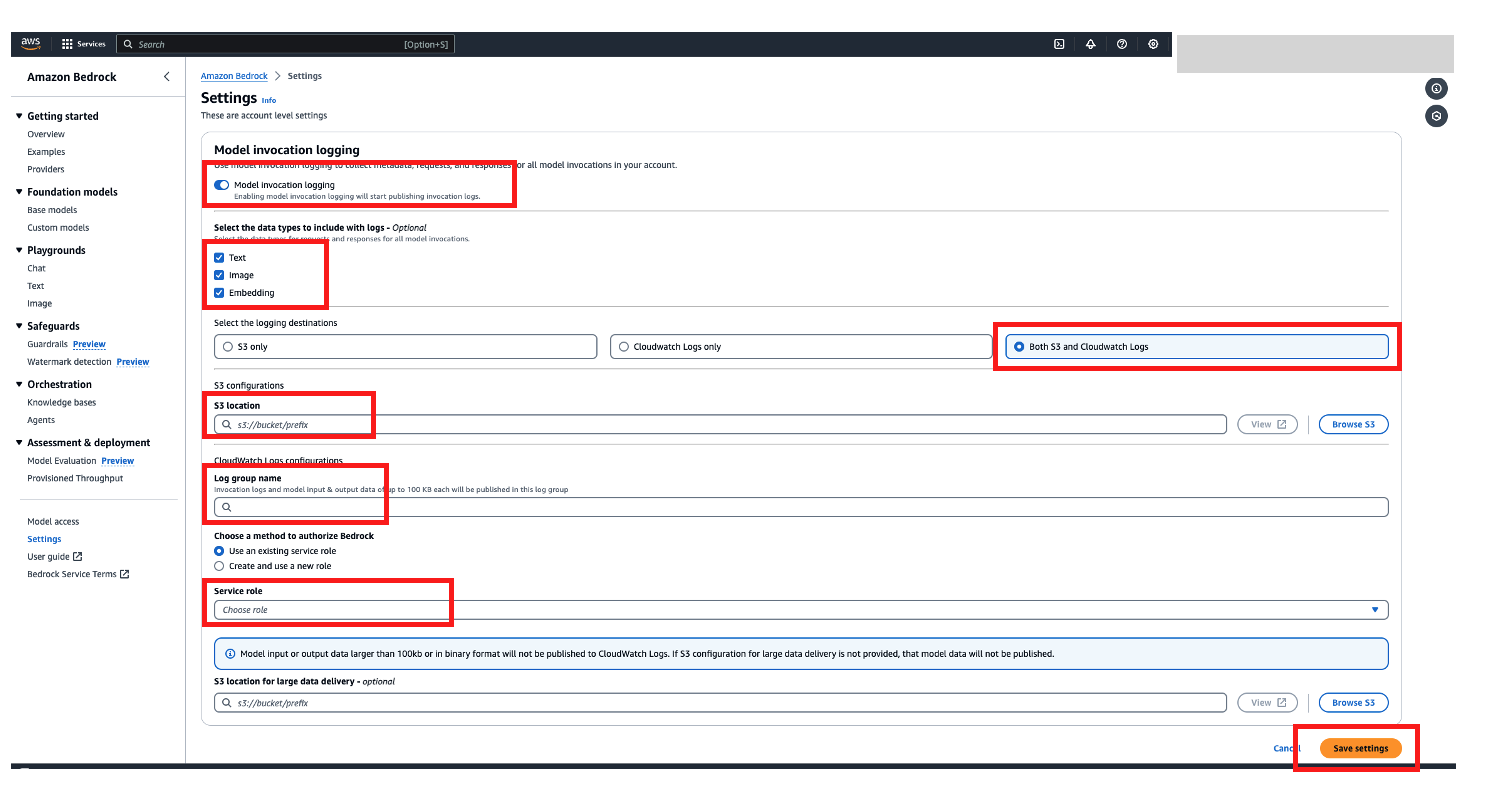
You will explore Model Evaluation in chapter 11. Whereas, Provisioned Throughput at chapter 10 and 12.

# **Model access**

To utilize a model within Amazon Bedrock, the initial step involves requesting access to the desired model. This can be accomplished by navigating to the left-hand side navigation pane and selecting Model access. ( *Figure 3.5 – 7* )

# You already learnt to edit, remove model access on section 3.2 and subsection of **Console Access**.

# **Model invocation logging**



*Figure 3.6 Amazon Bedrock console 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 – 8*) (*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/)*

# 3.6 Run model inference

Inference involves generating an output based on an input provided to a model. The bedrock Amazon enables the users to execute foundation models of their choice. Its users will give the following inputs while running inference:

**Instruction**: The input provided for the model to give a response. You will study in detail about prompt in chapter 4.

**Parameters for inference**: These values can be adjusted so as to restrict, or influence how this model responds. You will look into prompt further at subsection two.

There are different ways you can do model inference:

1. Utilize any of the Playgrounds to execute inference in a user-friendly graphical interface.

2. Send an api request.

3. Prepare a dataset of prompts with desired configurations and conduct batch inference

4. 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. 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 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. Though, you will learn some of this hyperparameters at chapter 10. For examples

**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 us 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 derived from learnt model 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 production of text or images during inference and are obtained from a trained model. You will explore inference parameters below.

# **Randomness and diversity**

We can see how a model assigns probabilities to possible next words by examining the probability distribution over them for each token in a sequence. 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**”.

Suppose we are given such a probability distribution for the following word:

*{“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** with higher probability 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 which has higher probability) would increase while that of selecting **ice cream** (the output with lower probability) will reduce. So, the answer from the model will be “**kids love to eat pizza**”.

suppose we take another example.

When the temperature is set low such as 0.1, then it makes conservative forecasts based on high certainty predictions by the model. This leads to more deterministic and cautious outputs with the model choosing high-likelihood tokens. “**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. “**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**”.

It is worth noting that optimal temperature may vary depending on what one intends to achieve with their output or based on how it should look like in specific cases of use. 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 for a next sentence.

Okay, let's consider same example prompt **kids love to eat** like previous inference parameter.

Consider a situation where this is what the model gives for probability distribution over 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 two then only top two candidates’ **pizza** and **vegetables** would be considered by model as possible outputs. 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 less probable outputs.

# **Top P**

Top P, nucleus sampling, represents a sampling strategy selects the top P most likely next tokens according to their cumulative 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 sequence.

Okay, let's consider same example prompt **kids love to eat** like previous inference parameter.

Let us assume that the model generates the following probabilities for each token:

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

Here's how the inference parameters would affect the output:

If we set Top P at 30 percent, only top 30% most possible candidates will be considered by this model: **pizza**; thus, leaving out **vegetables** and **ice cream** despite having non-zero probability.

When there is a lower value of Top P in order to narrow down on most likely outputs in favor of narrowing down candidate pool; conversely when there is a higher value of Top P so as to open up on less likely outputs.

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 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**" are 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**" are 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**

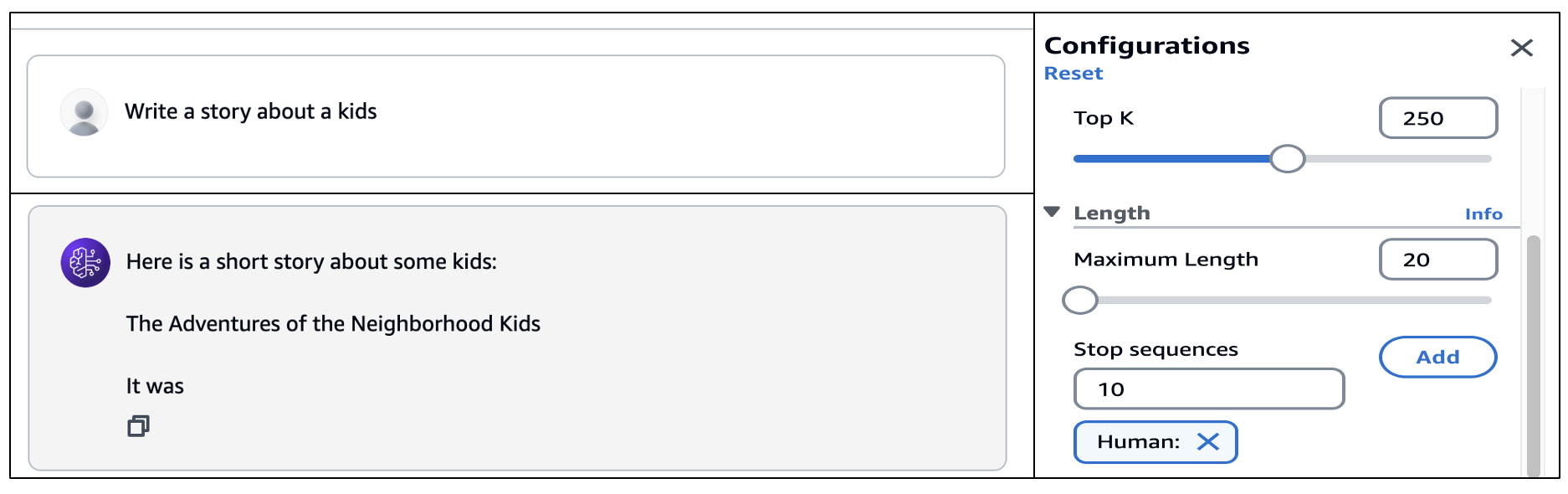
Foundation models commonly offer parameters to regulate the length and characteristics of the generated responses. Some examples of such parameters include:

**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 users 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 a user’s preferences and requirements in order to allow one better align with their interests and specific needs when using foundation models in generating texts.



*Figure 3.7 Amazon Bedrock console playground Length configurations*

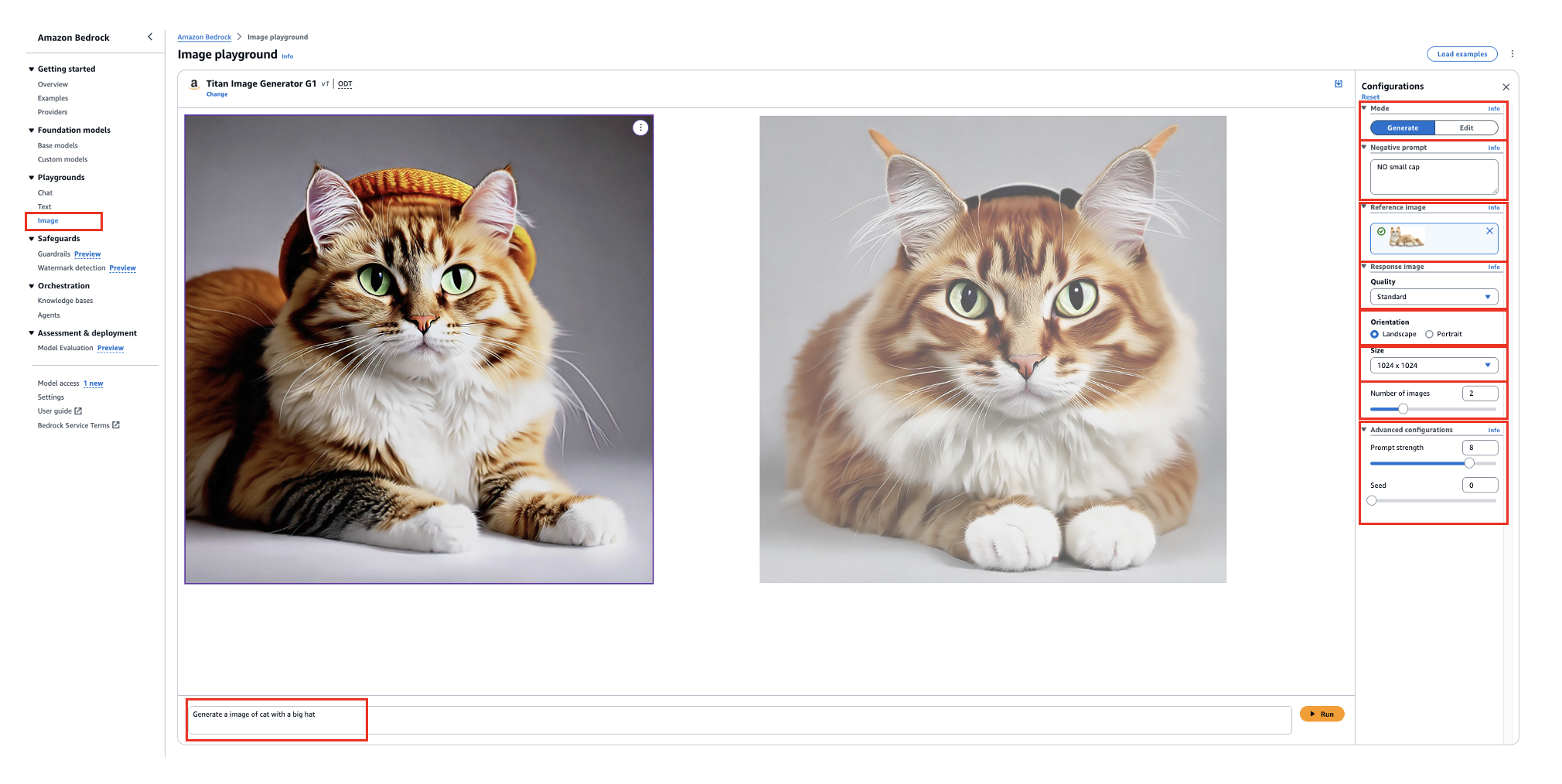
This is a picture of playground with **Length** configuration at Amazon Bedrock with a prompt “**Write a story about the kids**”. The Maximum Length is 20 and Stop sequences 10. Though, you can watch that the response started generating. But generation is incomplete due to Stop sequences parameter. (Figure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

**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 tell **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 should 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.



*Figure 3.8 Amazon Bedrock Image Playground*

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 3.7*)

# 3.7 Amazon SageMaker and Bedrock Interaction

To get the GitLab details, refer to the **appendix** section of this book. In GitLab, 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. Execute this AWS CloudFormation template. It should be noted that this step only has to be done 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/*](https://aws.amazon.com/cloudformation/).

*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** which comes before another subfolder called **Simple-SageMaker-Bedrock**. The **README** file within **Simple-SageMaker-Bedrock** directory provides clear instructions on launching a **Notebook** on Amazon SageMaker.

First, execute **Simple-SageMaker-Bedrock.ipynb** file. Inside of it there are elaborate guidelines.

*Charges will apply upon executing the Simple-SageMaker-Bedrock.ipynb file. 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.*

# 3.8 Bedrock Interaction Sample Application

To get the GitLab details, refer to the **appendix** section of this book. In GitLab, 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. Execute this AWS CloudFormation template **if you didn’t execute part of 3.7 section of this chapter, otherwise don’t execute AWS CloudFormation template**. It should be noted that this step only has to be done 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/*](https://aws.amazon.com/cloudformation/).

*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** which comes before another subfolder called **Simple-Bedrock-Application**. The **README** file within **Simple-Bedrock-Application** directory provides clear instructions on launching an **Application** on Amazon SageMaker.

First, execute **Simple-Bedrock-Application.py** file. Inside of it there are elaborate guidelines.

*Charges will apply upon executing the Simple-Bedrock-Application.py file. Therefore, it is important not to forget to clean up the application after studying the topic. Refer to the clean-up section for instructions on how to properly clean up the application.*

# 3.9 Summary

This chapter is a detailed introduction to Amazon Bedrock, which is a fully managed service designed for fast foundation models. That would be understanding 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 SageMaker.

Amazon Bedrock provides a single-entry point to both foundational AI models from top startups and Amazon that can be customized by users. It also allows users 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. The 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.