Amazon Bedrock Deep Dive

The easiest way to build and scale generative AI applications with foundation models

Andre Boaventura

Global Generative AI Specialist, AWS

Prompt:
Andre as Dr. Strange, marvel cinematic universe, 8K, photo

Agenda

Amazon Bedrock – Feature Highlights

Designing Generative AI Applications

Prompt Engineering

Architecture Patterns

Security

Resources

Amazon Bedrock

THE EASIEST WAY TO BUILD AND SCALE GENERATIVE AI APPLICATIONS WITH FMS

- Access a range of leading FMs via a single API
- Privately customize FMs with your own data
- Enable data security and compliance
- Build agents that execute complex business tasks by dynamically invoking APIs
- Extend the power of FMs with your data using retrieval augmented generation (RAG)
- Get the best price performance without managing infrastructure

How Amazon Bedrock works

Amazon Bedrock

Build generative AI applications using FMs through a serverless API service

Choose a FM

Use the playground to experiment with FMs and select the one that suits your needs

Use as is or customize

Fine-tune FMs as needed; Bedrock will automatically deploy the FM for inference

Send prompt

Use Bedrock API to send your prompts to the model

Receive response

Receive model response in your application

Amazon Bedrock supports leading foundation models

amazon

Amazon Titan

Text summarization, generation, classification, open-ended Q&A, information extraction, embeddings and search

AI21labs

Jurassic-2

Multilingual LLMs for text generation in Spanish, French, German, Portuguese, Italian, and Dutch

ANTHROP\C

Claude 2

LLM for conversations, question answering, and workflow automation based on research into training honest and responsible AI systems

cohere

Command and embed

Text generation model for business applications and embeddings model for search, clustering, or classification in 100+ languages

Meta

COMING SOON

Llama 2

Fine-tuned models ideal for dialogue use-cases and language tasks

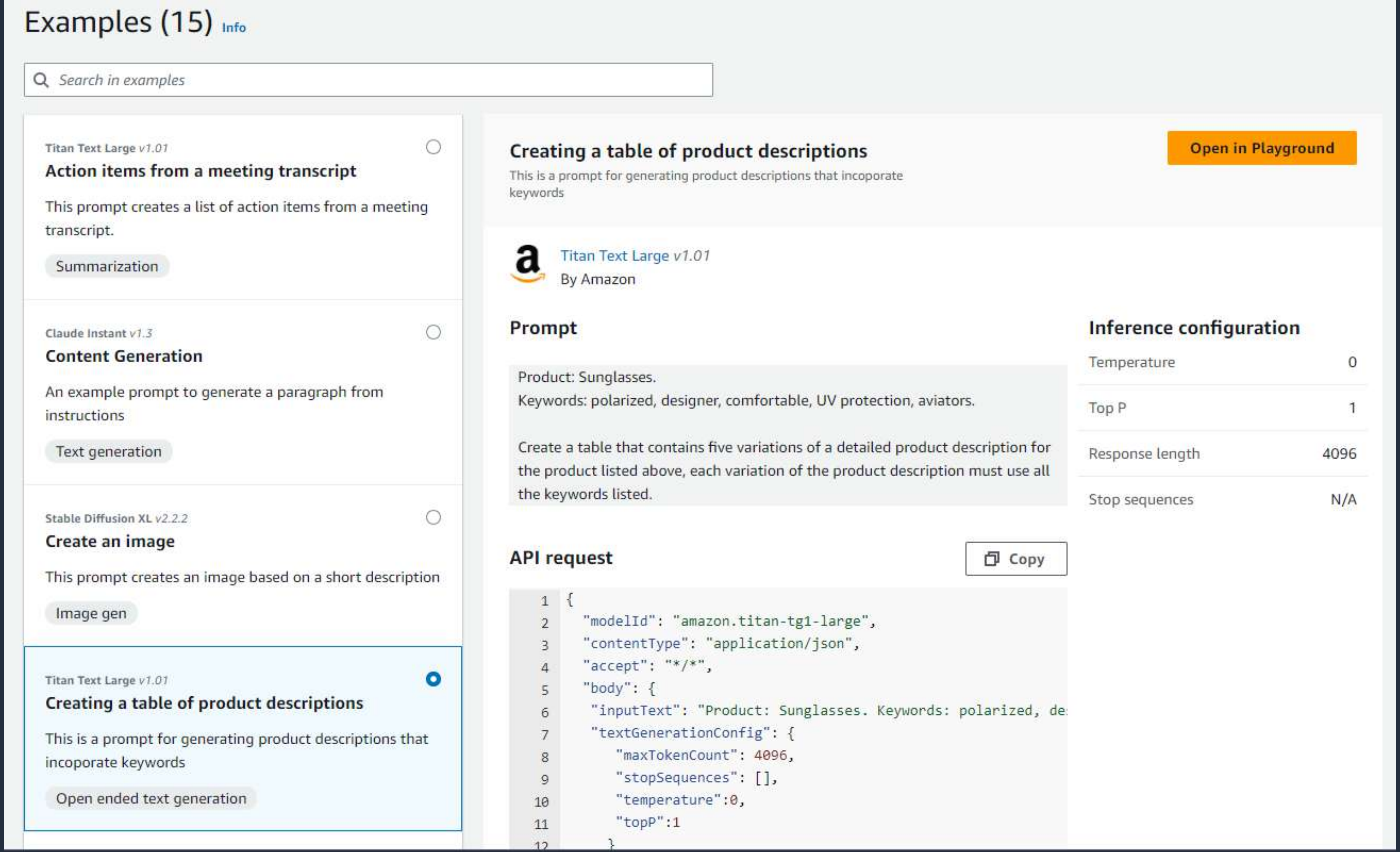
stability.ai

Stable Diffusion

Generation of unique, realistic, high-quality images, art, logos, and designs

aws

Experiment with models to try out real-life use cases



Examples (15) Info

○ Titan Text Large *v1.01*

Action items from a meeting transcript

This prompt creates a list of action items from a meeting transcript.

Summarization

○ Claude Instant *v1.3*

Content Generation

An example prompt to generate a paragraph from instructions

Text generation

○ Stable Diffusion XL *v2.2.2*

Create an image

This prompt creates an image based on a short description

Image gen

● Titan Text Large *v1.01*

Creating a table of product descriptions

This is a prompt for generating product descriptions that incoporate keywords

Open ended text generation

Creating a table of product descriptions

This is a prompt for generating product descriptions that incoporate keywords

Titan Text Large *v1.01*
By Amazon

Prompt

Product: Sunglasses.
Keywords: polarized, designer, comfortable, UV protection, aviators.

Create a table that contains five variations of a detailed product description for the product listed above, each variation of the product description must use all the keywords listed.

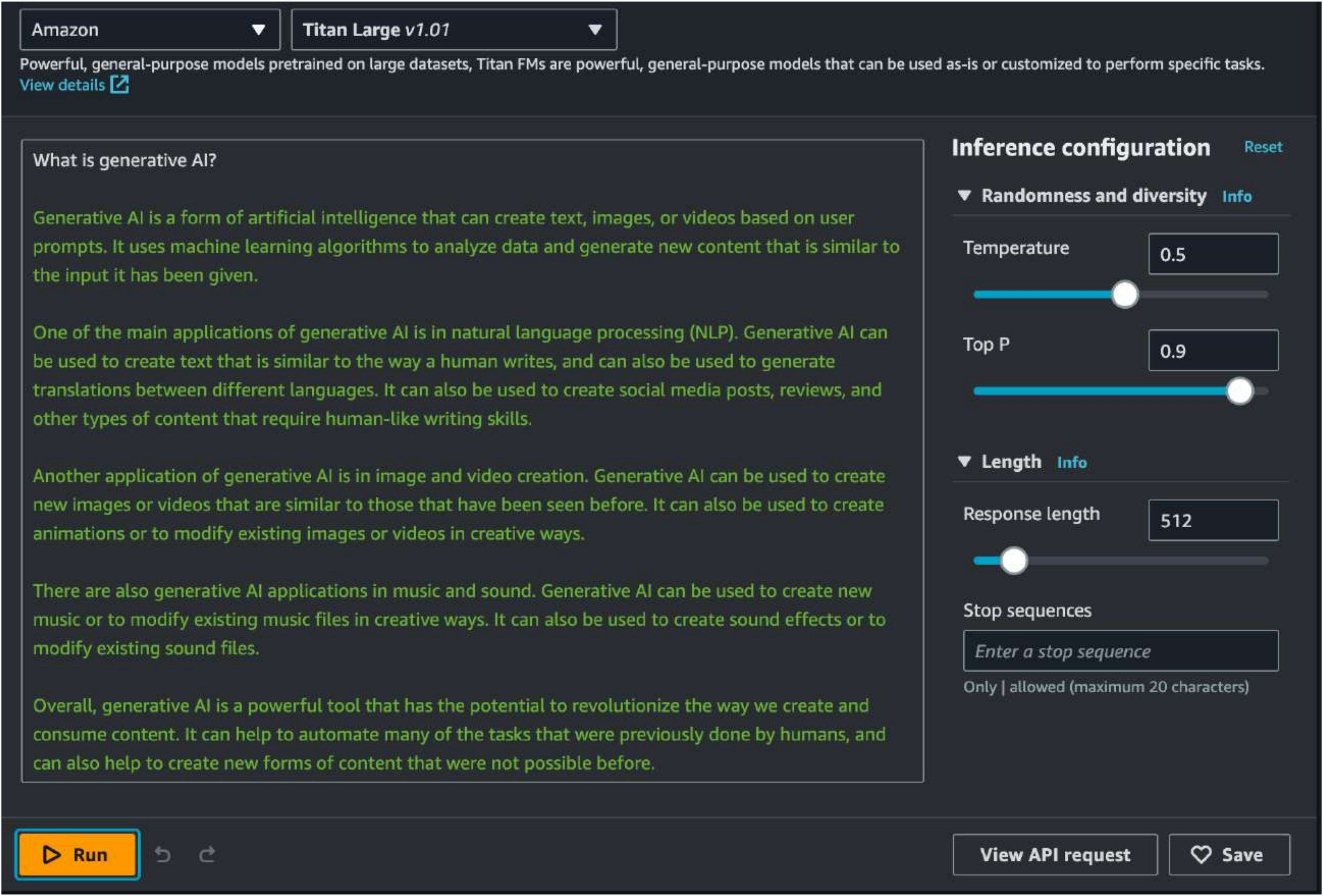
Inference configuration

Temperature	0
Top P	1
Response length	4096
Stop sequences	N/A

API request

```
{
  "modelId": "amazon.titan-tg1-large",
  "contentType": "application/json",
  "accept": "*/*",
  "body": {
   "inputText": "Product: Sunglasses. Keywords: polarized, de
   "textGenerationConfig": {
      "maxTokenCount": 4096,
      "stopSequences": [],
      "temperature":0,
      "topP":1
    }
```

Access Bedrock via the AWS console



Powerful, general-purpose models pretrained on large datasets, Titan FMs are powerful, general-purpose models that can be used as-is or customized to perform specific tasks.

What is generative AI?

Generative AI is a form of artificial intelligence that can create text, images, or videos based on user prompts. It uses machine learning algorithms to analyze data and generate new content that is similar to the input it has been given.

One of the main applications of generative AI is in natural language processing (NLP). Generative AI can be used to create text that is similar to the way a human writes, and can also be used to generate translations between different languages. It can also be used to create social media posts, reviews, and other types of content that require human-like writing skills.

Another application of generative AI is in image and video creation. Generative AI can be used to create new images or videos that are similar to those that have been seen before. It can also be used to create animations or to modify existing images or videos in creative ways.

There are also generative AI applications in music and sound. Generative AI can be used to create new music or to modify existing music files in creative ways. It can also be used to create sound effects or to modify existing sound files.

Overall, generative AI is a powerful tool that has the potential to revolutionize the way we create and consume content. It can help to automate many of the tasks that were previously done by humans, and can also help to create new forms of content that were not possible before.

Temperature: 0.5

Top P: 0.9

Response length: 512

Stop sequences: *Enter a stop sequence*

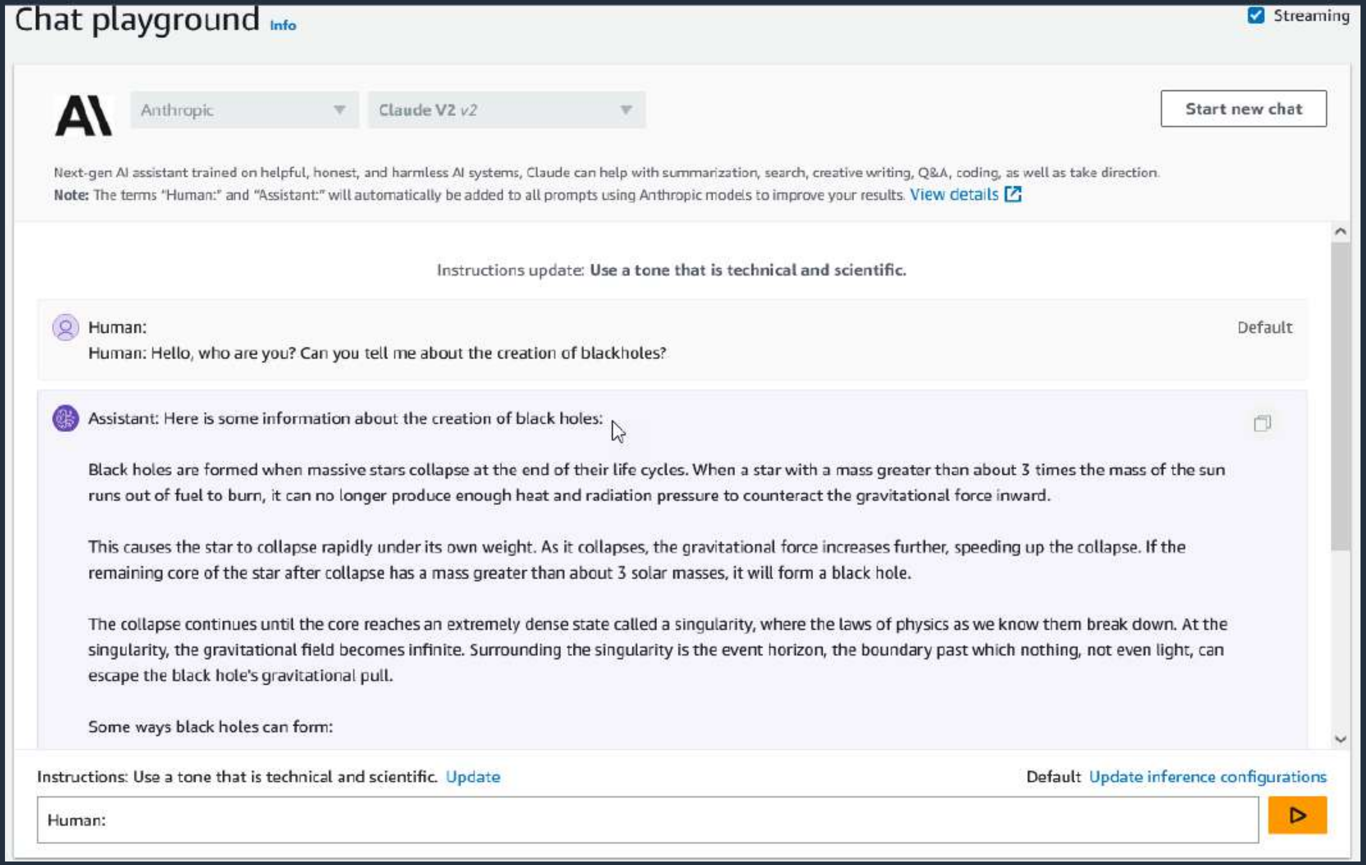
Only | allowed (maximum 20 characters)

Chat Playground

Quickly evaluate Foundation models using the Chat playground

Provide instructions to assume a role throughout the conversation in its entirety.

Prompts and responses are erased once the browser session is complete, or by clicking on ‘Start new chat’



Next-gen AI assistant trained on helpful, honest, and harmless AI systems, Claude can help with summarization, search, creative writing, Q&A, coding, as well as take direction.

Note: The terms "Human:" and "Assistant:" will automatically be added to all prompts using Anthropic models to improve your results. View details

Instructions update: **Use a tone that is technical and scientific.**

Human: Default

Human: Hello, who are you? Can you tell me about the creation of blackholes?

Assistant: Here is some information about the creation of black holes:

Black holes are formed when massive stars collapse at the end of their life cycles. When a star with a mass greater than about 3 times the mass of the sun runs out of fuel to burn, it can no longer produce enough heat and radiation pressure to counteract the gravitational force inward.

This causes the star to collapse rapidly under its own weight. As it collapses, the gravitational force increases further, speeding up the collapse. If the remaining core of the star after collapse has a mass greater than about 3 solar masses, it will form a black hole.

The collapse continues until the core reaches an extremely dense state called a singularity, where the laws of physics as we know them break down. At the singularity, the gravitational field becomes infinite. Surrounding the singularity is the event horizon, the boundary past which nothing, not even light, can escape the black hole's gravitational pull.

Some ways black holes can form:

Instructions: Use a tone that is technical and scientific. Update

Default: Update inference configurations

Human:

Amazon Bedrock API

Single API to build with generative AI

Single API to build with generative AI

Bedrock core API: InvokeModel

- Pass the model ID, type of content, and body of the request
  - Body includes the prompt and execution parameters
  - Returns model response and metadata
- Handles text-to-text, text-to-image, image-to-image, and more
- Supports current and future Amazon Titan models, third-party models, and even fine-tuned models

Bedrock core API: InvokeModel

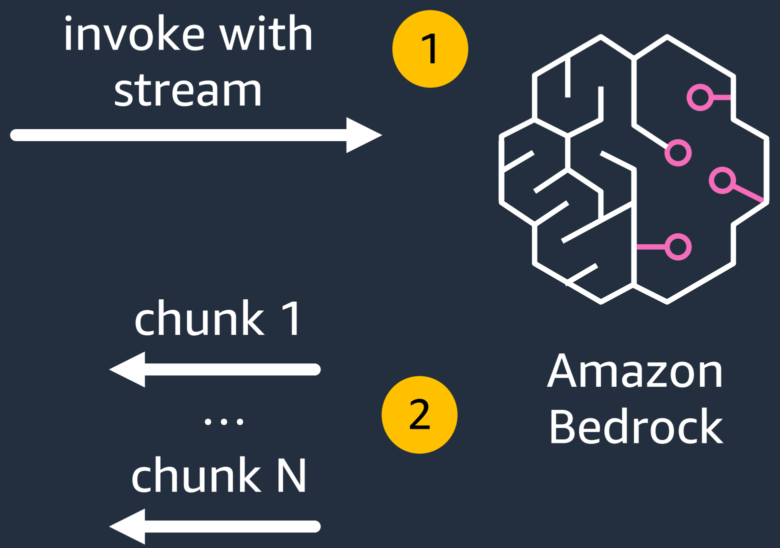
```
bedrock.invoke_model(
    modelId = model_id,
    contentType = ”…",
    accept = ”…",
    body = body)
```

Access foundation models

- Amazon Titan models
- Third-party models
- Fine-tuned models **NEW**

Bedrock core API: Streaming responses

```
response = bedrock.invoke_model_with_response_stream(
      modelId = model_id, body = body)
stream = response.get('body')
if stream:
     for event in stream:
          chunk = event.get('chunk’)
          if chunk:
               print(json.loads(chunk.get('bytes').decode()))
```



- Users can start reading the response as soon as the first chunk is available
- Initially supported for Amazon Titan models; Claude and J2 models coming soon

API operations: `invoke_model()`

PARAMETERS

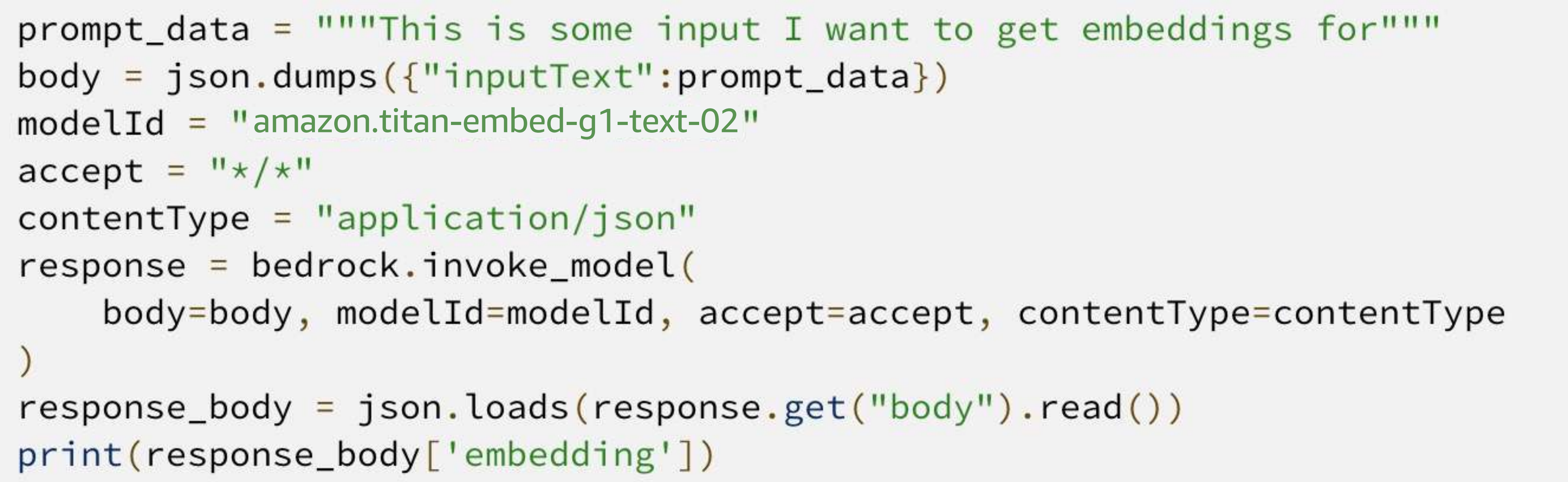
Parameters	Models				
	Titan Text Large	Claude	Command	Jurassic	Stable Diffusion
Temperature	✅	✅		✅	
TopP	✅	✅		✅	
StopSequences	✅	✅		✅	
MaxTokens	✅	✅		✅	
TopK		✅			
CountPenalty				✅	
PresencePenalty				✅	
FrequencePenalty				✅	
Prompt strength (cfg_scale)					✅
Generation step					✅
Seed					✅

`invoke_model()` – Titan Text

```
prompt_data = """Write me a blog about making strong business decisions as a leader"""

config = {"maxTokenCount":512,"stopSequences":[],"temperature":0.5,"topP":0.9}
body = json.dumps({"inputText":prompt_data,"textGenerationConfig": config})
modelId = "amazon.titan-tg1-large"
accept = "*/*"
contentType = "application/json"
response = bedrock.invoke_model(
    body=body, modelId=modelId, accept=accept, contentType=contentType
)
response_body = json.loads(response.get("body").read())
print(response_body.get("results")[0].get("outputText"))
```

`invoke_model()` – Titan Embeddings



```
prompt_data = """This is some input I want to get embeddings for"""
body = json.dumps({"inputText":prompt_data})
modelId = "amazon.titan-embed-g1-text-02"
accept = "*/*"
contentType = "application/json"
response = bedrock.invoke_model(
    body=body, modelId=modelId, accept=accept, contentType=contentType
)
response_body = json.loads(response.get("body").read())
print(response_body['embedding'])
```

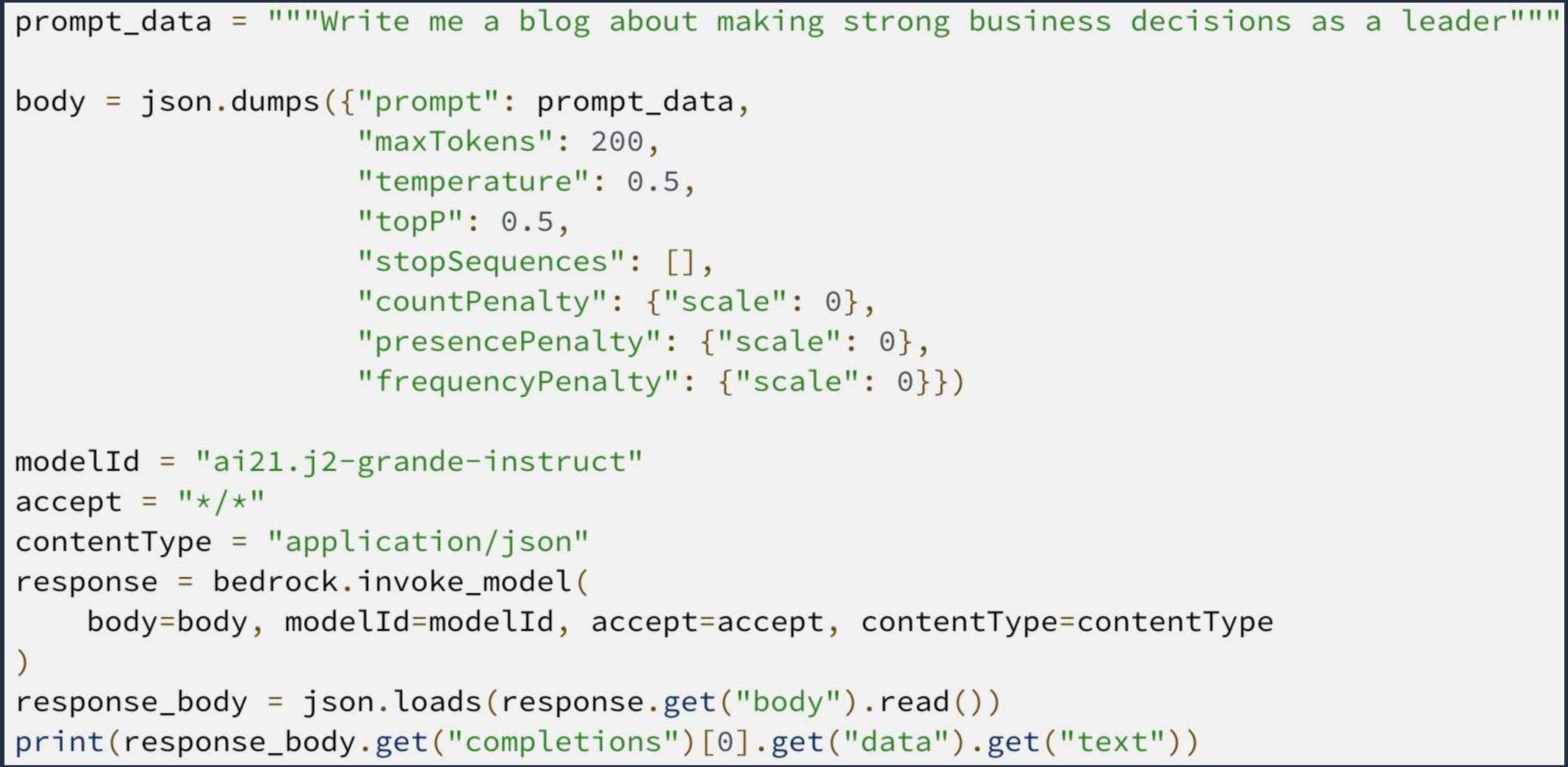
`invoke_model()` – Anthropic Claude

CLAUDE

```
prompt_data = """Write me a blog about making strong business decisions as a leader"""

body = json.dumps({"prompt": prompt_data,
                   "max_tokens_to_sample": 300,
                   "temperature": 0.5,
                   "top_k": 250,
                   "top_p": 1,
                   "stop_sequences": ["\n\nHuman:"]})
modelId = "anthropic.claude-instant-v1"
accept = "*/*"
contentType = "application/json"
response = bedrock.invoke_model(
    body=body, modelId=modelId, accept=accept, contentType=contentType
)
response_body = json.loads(response.get("body").read())
print(response_body.get("completion"))
```

`invoke_model()` – AI21 Labs Jurassic-2



```
prompt_data = """Write me a blog about making strong business decisions as a leader"""

body = json.dumps({"prompt": prompt_data,
                   "maxTokens": 200,
                   "temperature": 0.5,
                   "topP": 0.5,
                   "stopSequences": [],
                   "countPenalty": {"scale": 0},
                   "presencePenalty": {"scale": 0},
                   "frequencyPenalty": {"scale": 0}})

modelId = "ai21.j2-grande-instruct"
accept = "*/*"
contentType = "application/json"
response = bedrock.invoke_model(
    body=body, modelId=modelId, accept=accept, contentType=contentType
)
response_body = json.loads(response.get("body").read())
print(response_body.get("completions")[0].get("data").get("text"))
```

invoke_model() – Stability.ai Stable Diffusion

```
body = json.dumps({
    "text_prompts": [
        {"text": "a pizza", "weight": 1},
        {"text": "an unrealistic pizza shape", "weight": -1}
    ],
    "cfg_scale": 5,
    "seed": 123,
    "steps": 50,
    "style_preset": "cinematic",
    # For in-painting or image-to-image task, add following
    "init_image": "<base64 encoded string of image>",
    "start_schedule": 0.6 # 0-1 scale of relevancy
})
modelId = "stability.stable-diffusion-xl"
accept = "application/json"
contentType = "application/json"
response = bedrock.invoke_model (
    body=body, modelId=modelId, accept=accept, contentType=contentType
)
response_body = json.loads(response.get("body").read())
base_64_img_str = response_body["artifacts"][0]["base64"]
image = Image.open(io.BytesIO(base64.decodebytes(bytes(base_64_img_str, "utf-8"))))
print(image)
```

Integrated with LangChain

```
pip install langchain
```

```
from langchain import Bedrock
from langchain.embeddings
import BedrockEmbeddings

llm = Bedrock()
print(llm(“what is generative
AI?”))
```

Popular Python framework for developing applications powered by language models

- New LLM and embeddings class for Amazon Bedrock
- Includes code for using the LLM class in a conversation chain
- Includes code for creating an embedding from text

Summary of Amazon Bedrock Inference options

On demand

Pay-as-you-go; no usage commitments

- Pricing based on input and output token count for LLMs
- Great for prototyping, POC’s, small workloads with more relaxed requirements for throughput and latency
- Requests per minute (RPM) and Tokens per minute (TPM) enforced

Provisioned throughput

Provision sufficient throughput to meet your application’s performance requirements

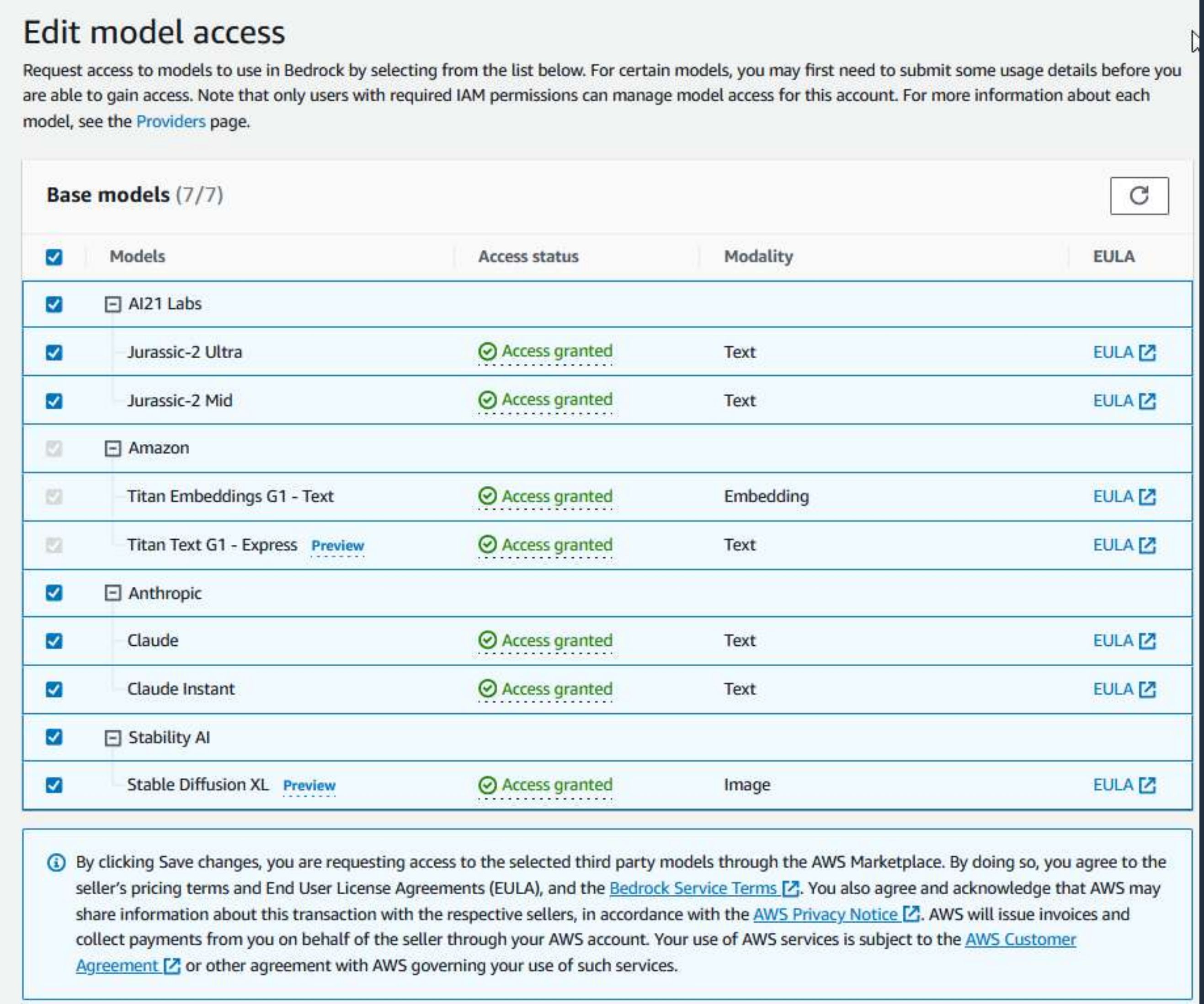
- Guaranteed throughput at a fixed cost
- Higher throughput available
- Flexible commitment term of 1 month or 6 months
- Pay hourly rate, discounted for extended commit
- Great for production workloads, or inference on custom models

Model access

Fine grained model access

Customers can now easily subscribe to specific foundation models that best fit their use cases

For Claude models, customers will continue to ‘Request’ access by completing *Anthropic’s models usage details* form.



Edit model access

Request access to models to use in Bedrock by selecting from the list below. For certain models, you may first need to submit some usage details before you are able to gain access. Note that only users with required IAM permissions can manage model access for this account. For more information about each model, see the Providers page.

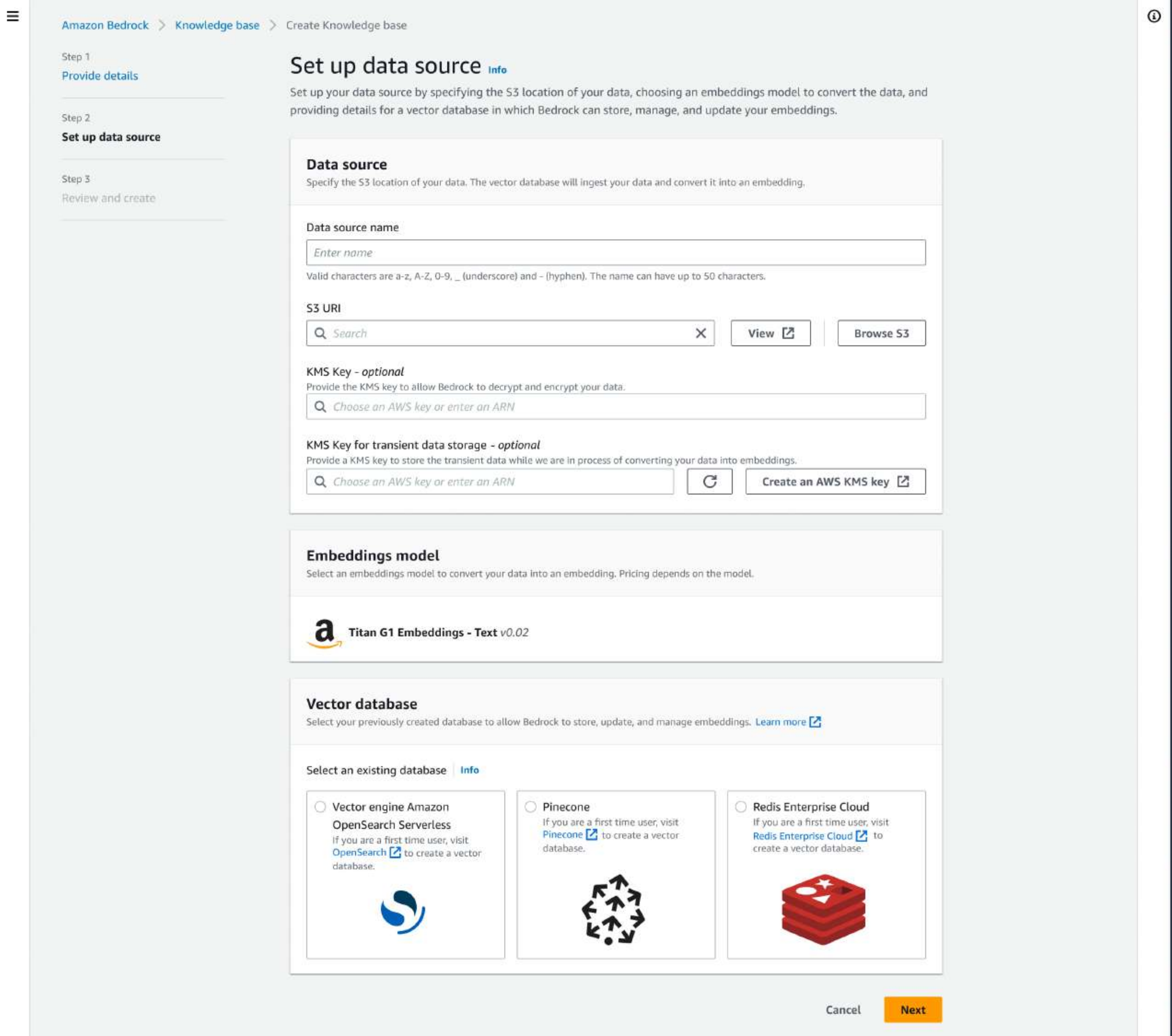
Base models (7/7)

☑	Models	Access status	Modality	EULA
☑	⊟ AI21 Labs			
☑	Jurassic-2 Ultra	✓ Access granted	Text	EULA
☑	Jurassic-2 Mid	✓ Access granted	Text	EULA
☑	⊟ Amazon			
☑	Titan Embeddings G1 - Text	✓ Access granted	Embedding	EULA
☑	Titan Text G1 - Express Preview	✓ Access granted	Text	EULA
☑	⊟ Anthropic			
☑	Claude	✓ Access granted	Text	EULA
☑	Claude Instant	✓ Access granted	Text	EULA
☑	⊟ Stability AI			
☑	Stable Diffusion XL Preview	✓ Access granted	Image	EULA

By clicking Save changes, you are requesting access to the selected third party models through the AWS Marketplace. By doing so, you agree to the seller’s pricing terms and End User License Agreements (EULA), and the Bedrock Service Terms. You also agree and acknowledge that AWS may share information about this transaction with the respective sellers, in accordance with the AWS Privacy Notice. AWS will issue invoices and collect payments from you on behalf of the seller through your AWS account. Your use of AWS services is subject to the AWS Customer Agreement or other agreement with AWS governing your use of such services.

Knowledge base for retrieval augmented generation (RAG) PREVIEW

- Connect FMs to data sources including vector engine for Amazon OpenSearch Serverless, Pinecone, and Redis Enterprise Cloud
- Enable automatic data source detection
- Provide source attribution



Set up data source Info

Set up your data source by specifying the S3 location of your data, choosing an embeddings model to convert the data, and providing details for a vector database in which Bedrock can store, manage, and update your embeddings.

Data source

Specify the S3 location of your data. The vector database will ingest your data and convert it into an embedding.

Data source name

Enter name

Valid characters are a-z, A-Z, 0-9, _ (underscore) and - (hyphen). The name can have up to 50 characters.

S3 URI

KMS Key - *optional*

Provide the KMS key to allow Bedrock to decrypt and encrypt your data.

Choose an AWS key or enter an ARN

KMS Key for transient data storage - *optional*

Provide a KMS key to store the transient data while we are in process of converting your data into embeddings.

Choose an AWS key or enter an ARN

Embeddings model

Select an embeddings model to convert your data into an embedding. Pricing depends on the model.

Titan G1 Embeddings - Text *v0.02*

Vector database

Select your previously created database to allow Bedrock to store, update, and manage embeddings. Learn more

Select an existing database | Info

○ Vector engine Amazon OpenSearch Serverless
If you are a first time user, visit OpenSearch to create a vector database.

○ Pinecone
If you are a first time user, visit Pinecone to create a vector database.

○ Redis Enterprise Cloud
If you are a first time user, visit Redis Enterprise Cloud to create a vector database.

Agents enable generative AI applications to complete tasks in just a few clicks

PREVIEW

1. SELECT YOUR FOUNDATION MODEL
2. PROVIDE BASIC INSTRUCTIONS
3. SELECT RELEVANT DATA SOURCES
4. DEVELOPER SPECIFIES LAMBDA FUNCTIONS

Breaks down and orchestrates tasks

Securely accesses and retrieves company data

Takes action by invoking API calls on your behalf

Provides fully managed infrastructure support

Custom models

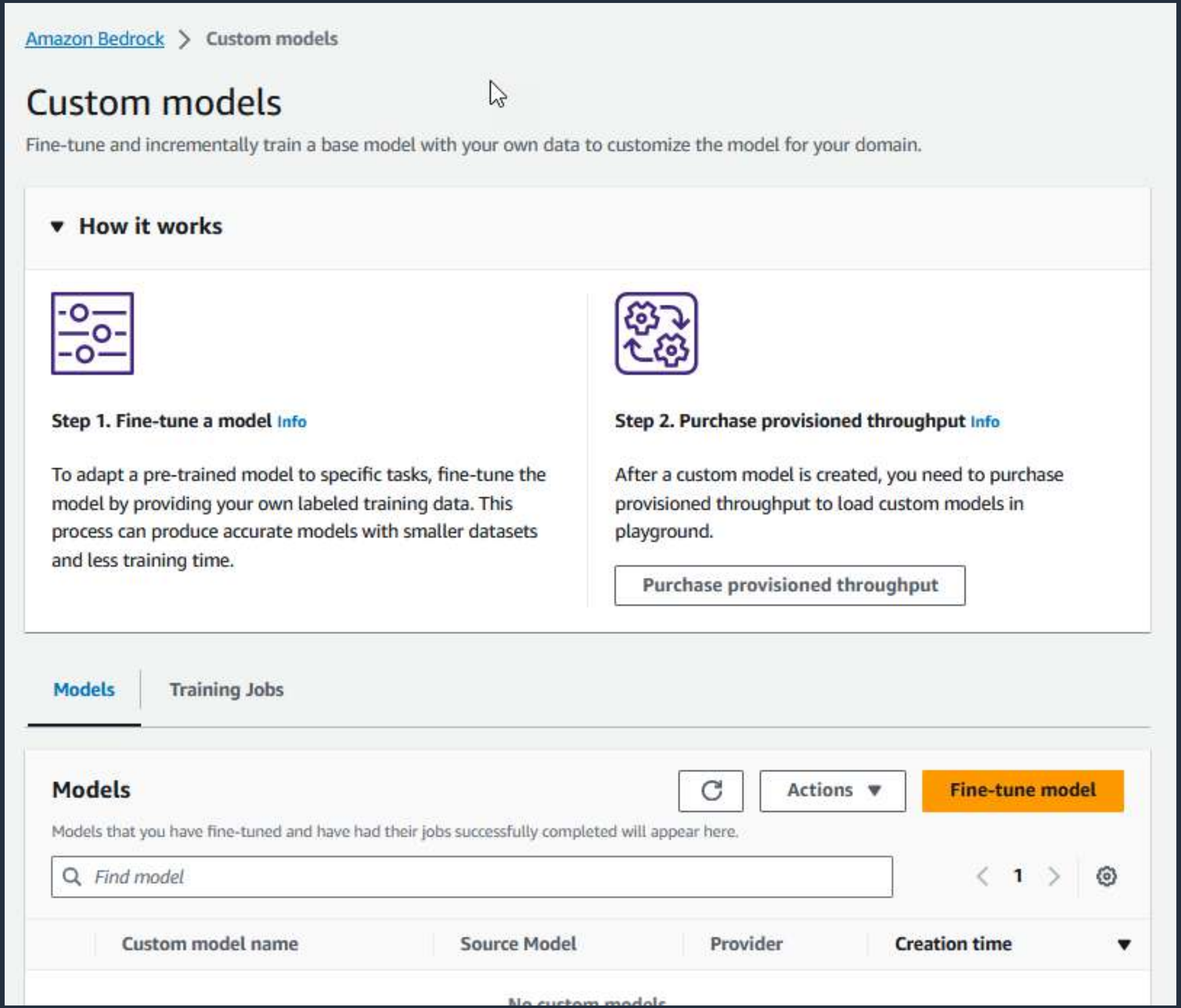
Create fine-tuned models using APIs

Fine-tuned models can be invoked the same way as base models, via the playground, or through the API on Provisioned throughput

Currently, fine-tuning is supported for Titan Text models. Support for other models will be enabled over time.

Customers with access to Amazon Titan Text models will now have access to fine tuning NEW

Tagging lets you easily manage, identify, organize, search for, and filter fine tuning jobs and models



Amazon Bedrock > Custom models

Custom models

Fine-tune and incrementally train a base model with your own data to customize the model for your domain.

▼ How it works

Step 1. Fine-tune a model Info

To adapt a pre-trained model to specific tasks, fine-tune the model by providing your own labeled training data. This process can produce accurate models with smaller datasets and less training time.

Step 2. Purchase provisioned throughput Info

After a custom model is created, you need to purchase provisioned throughput to load custom models in playground.

Models

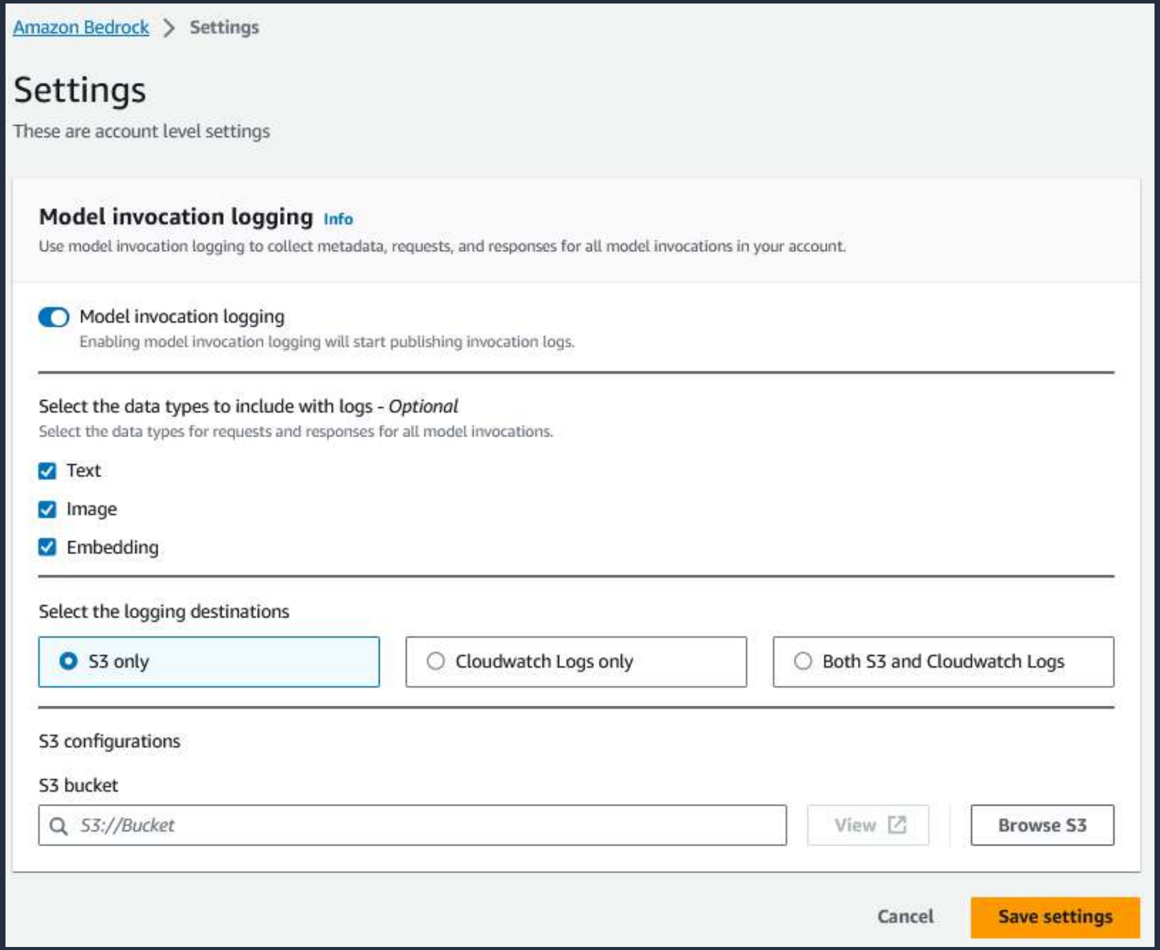
Models that you have fine-tuned and have had their jobs successfully completed will appear here.

Custom model name	Source Model	Provider	Creation time ▼

Model invocation logging

Use model invocation logging to collect metadata, requests, and responses for all model invocations in your account.

Supports both S3 and CloudWatch Logs



Amazon Bedrock > Settings

Settings

These are account level settings

Model invocation logging Info

Use model invocation logging to collect metadata, requests, and responses for all model invocations in your account.

Model invocation logging

Enabling model invocation logging will start publishing invocation logs.

Select the data types to include with logs - *Optional*

Select the data types for requests and responses for all model invocations.

☑ Text
☑ Image
☑ Embedding

Select the logging destinations

● S3 only ○ Cloudwatch Logs only ○ Both S3 and Cloudwatch Logs

S3 configurations

S3 bucket

S3://Bucket

CloudWatch metrics

Amazon
CloudWatch

CloudWatch metrics now supported:

- Number of model invocations
- Latency of invocation
- Error metrics include number of invocations with:
  - Client side errors
  - Server side errors
  - Throttling

“AWS/Bedrock” namespace, and each metric is per model (“ModelId” dimension)

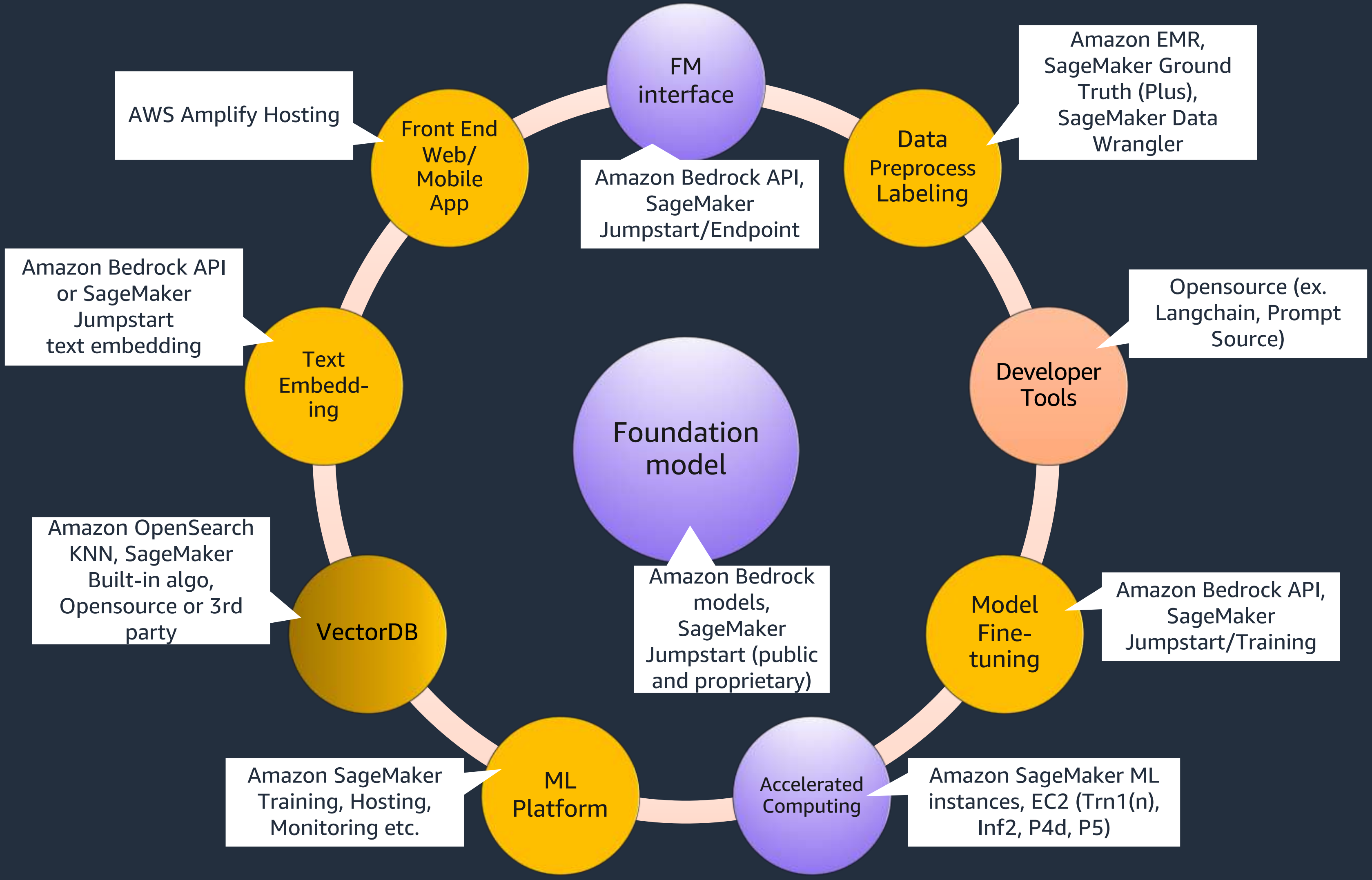
Demo

Agents, Knowledge Base, and Custom Models

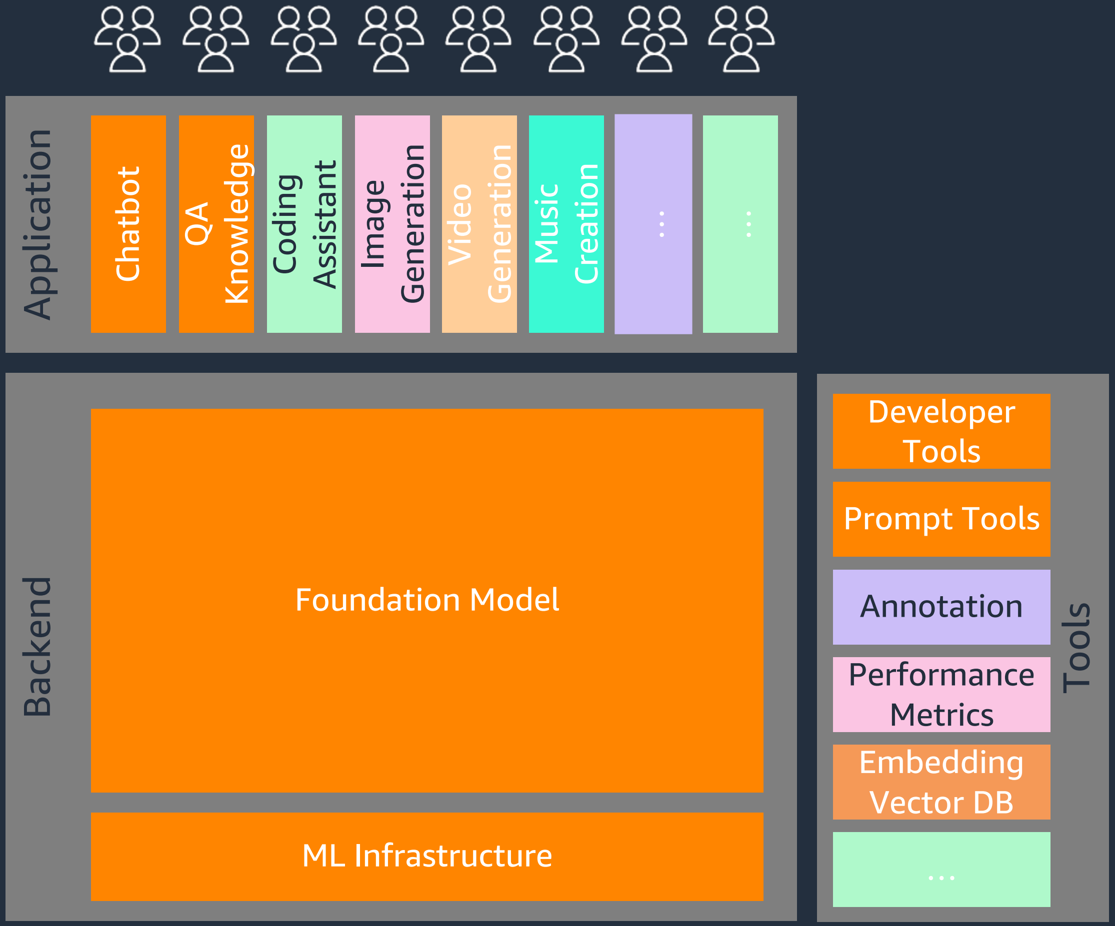
Designing Generative AI applications

Components of Generative AI application

Components to AWS services mapping



Generative AI Ecosystem



Application

Chatbot

QA Knowledge

Coding Assistant

Image Generation

Video Generation

Music Creation

…

…

Backend

Foundation Model

ML Infrastructure

Tools

Developer Tools

Prompt Tools

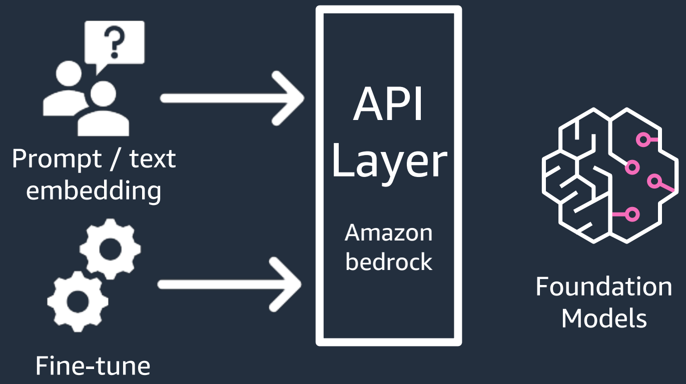
Annotation

Performance Metrics

Embedding Vector DB

…

How do I access foundation models?



Prompt / text embedding

Fine-tune

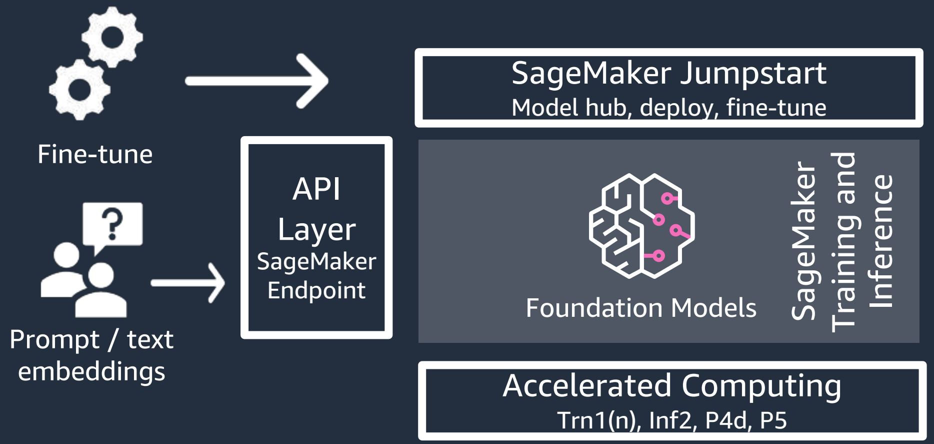
API Layer

Amazon bedrock

Foundation Models

Amazon Bedrock

- The easiest way to build and scale generative AI applications with foundation models (FMs)
- Access directly or fine-tune foundation model using API
- Serverless



Fine-tune

Prompt / text embeddings

API Layer

SageMaker Endpoint

SageMaker Jumpstart

Model hub, deploy, fine-tune

Foundation Models

SageMaker Training and Inference

Accelerated Computing

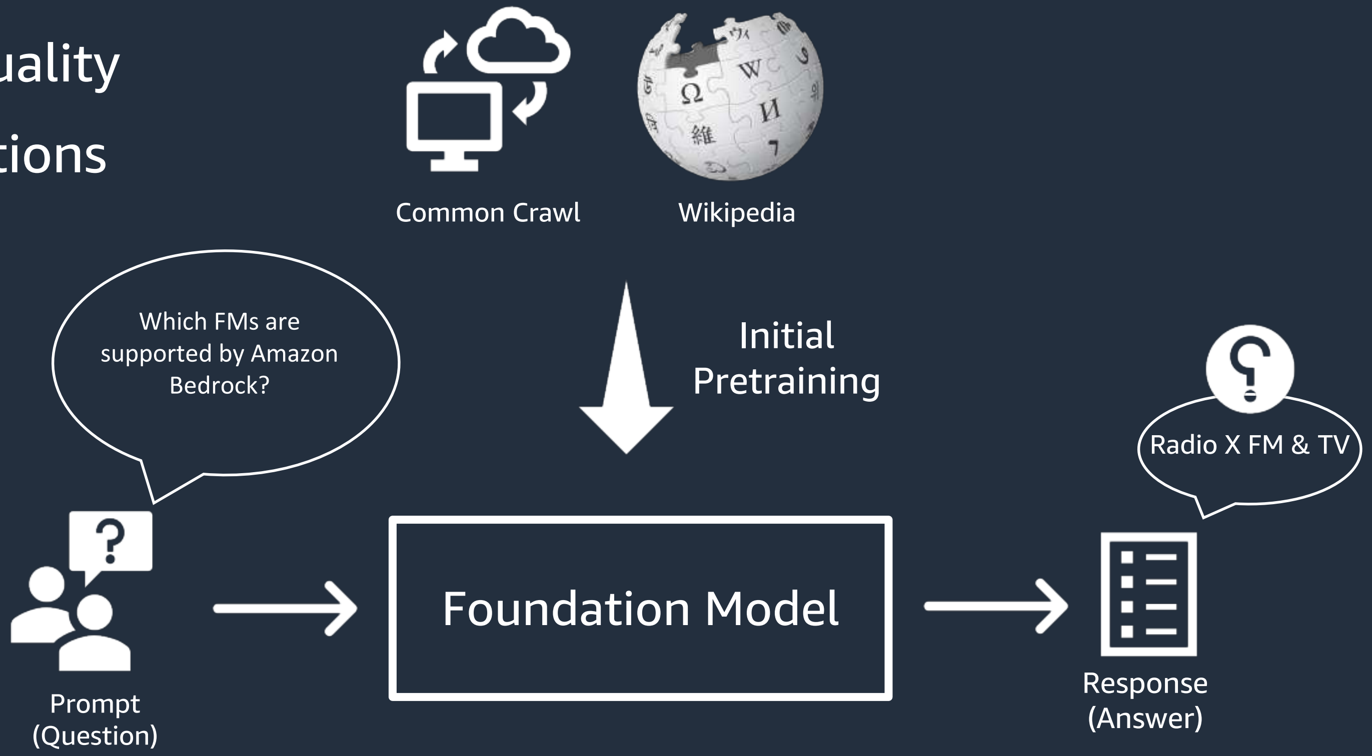
Trn1(n), Inf2, P4d, P5

Amazon SageMaker JumpStart

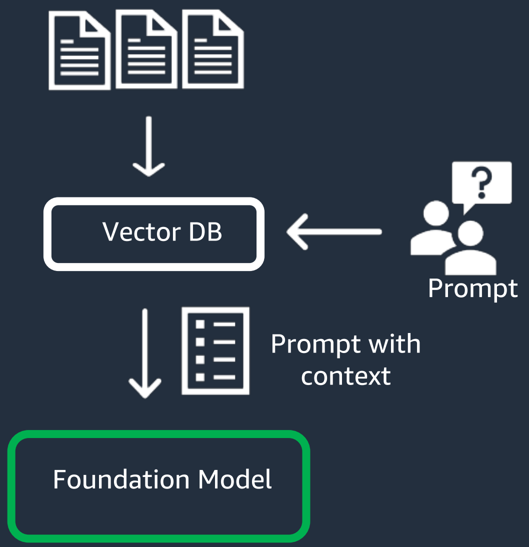
- Machine learning (ML) hub with foundation models, built-in algorithms, and prebuilt ML solutions that you can deploy with just a few clicks
- Deploy FM as SageMaker Endpoint (hosting)
- Fine-tuning leverages SageMaker Training jobs
- Choose SageMaker managed accelerated computing instance

Why customize a foundation model?

- Specific Task
- Closed-domain knowledge
- Current Knowledge
- Improving the performance/quality
- Reduce likelihood of hallucinations



How can I customize a foundation model?



Vector DB

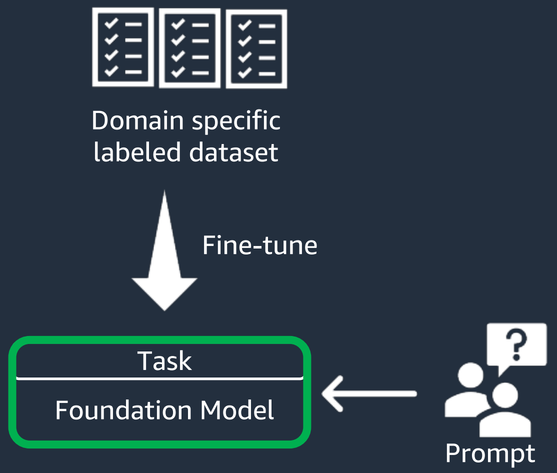
Prompt

Prompt with context

Foundation Model

Information Retrieval

- Augmented with closed-domain knowledge (RAG)
- Retrieval using dense vector representation
- Feeds prompt with the ”Context”
- Query FM w/ context to get the output
- No updates to model weights



Domain specific labeled dataset

Fine-tune

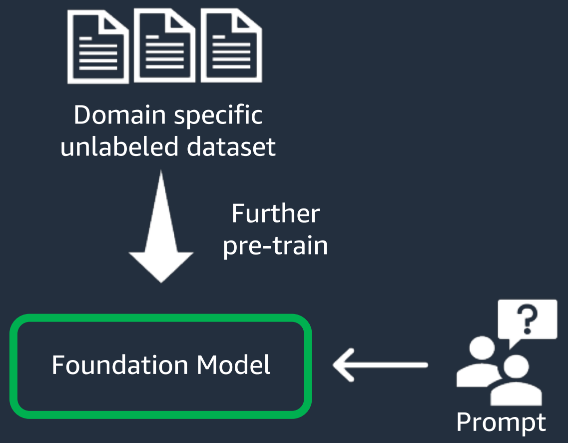
Task

Foundation Model

Prompt

Instruction Fine-tuning

- Supervised learning (Fine Tuning)
- Updates the model weight
- Small amount of data (100 to 500) - JSONL
- Fewer epochs
- Task Specific
- Supported models (T5, GPT-J, Diffusion)



Domain specific unlabeled dataset

Further pre-train

Foundation Model

Prompt

Domain Adaptation

- Self-supervised learning (further pretrain)
- Updates the model weight
- Large corpus of data (CSV, JSON, TXT)
- Custom vocabulary & tokenizer
- Supported models – GPT-JB & Variant
- Ex. Legal or fintech domain

Customization Approaches

1

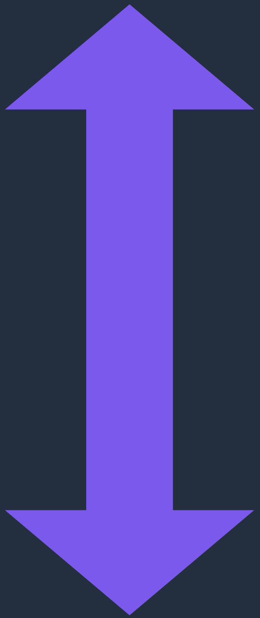
Use case prioritization

2

Which model do you start with?

3

Easiest technical solution
Lowest level of customization



- Retrieval augmented generation
- Prompt engineering with few-shot inference
- Fine-tune an existing foundation model
- Re-training an existing foundation model
- Build a foundation model from scratch

Most difficult technical solution
Highest level of customization

4

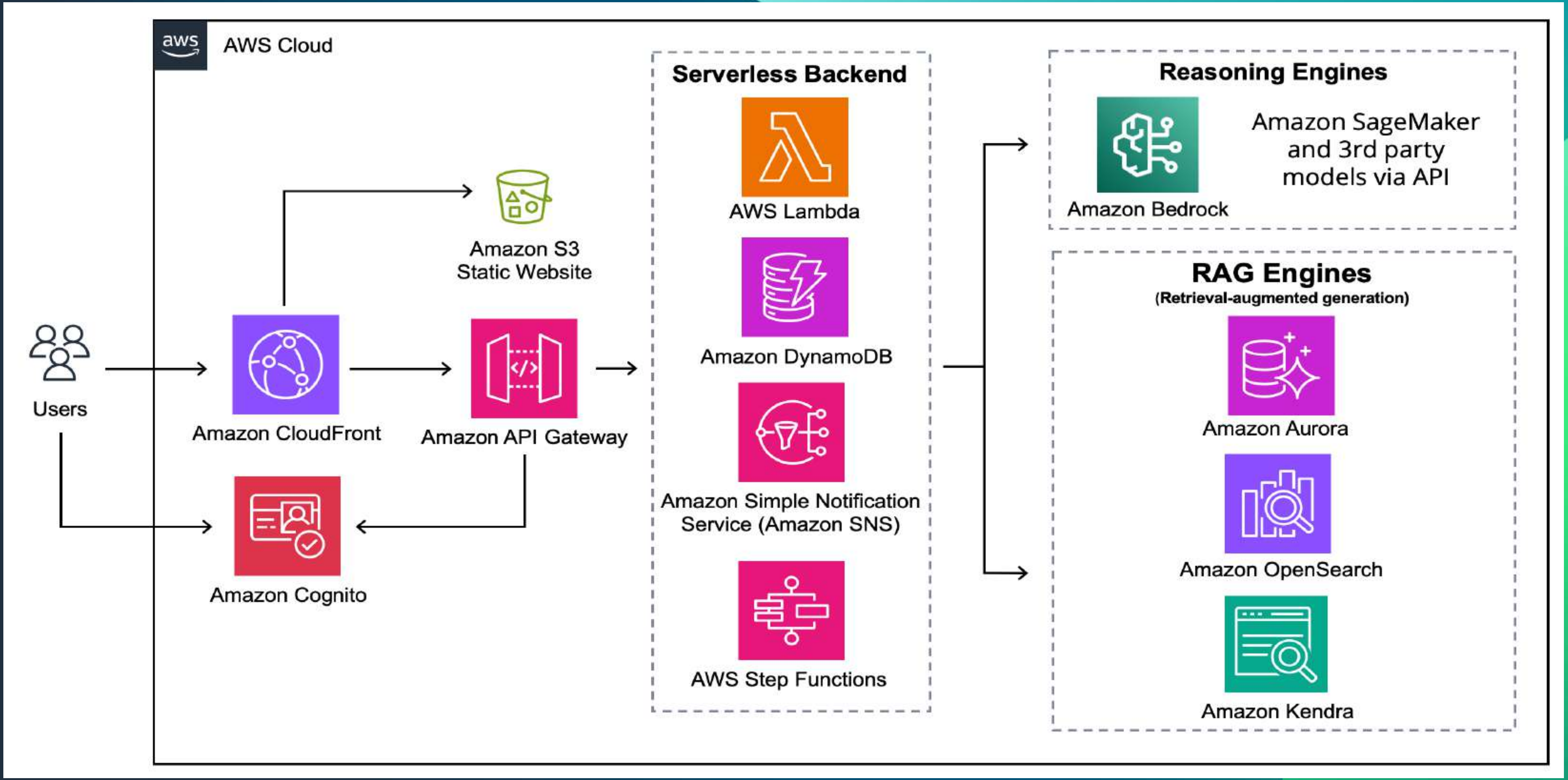
Integration with processes

Our customers look to us as trusted advisors to guide them through these choices

Demo

RAG with Amazon Bedrock

AWS GenAI Chatbot Demo

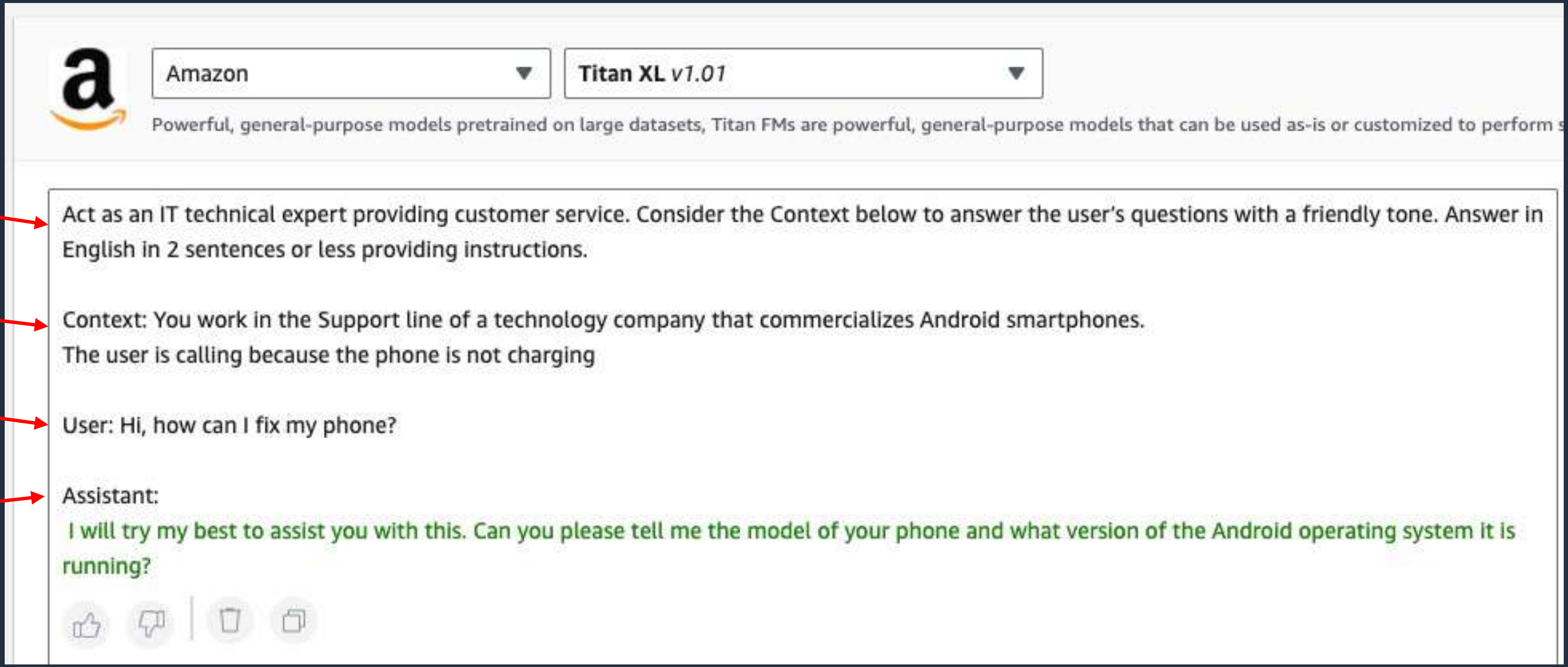


https://github.com/aws-samples/aws-genai-llm-chatbot

Prompt Engineering

(Typical) Prompt Structure

- Instructions
- Context
- User Input
- Output Indicator



Powerful, general-purpose models pretrained on large datasets, Titan FMs are powerful, general-purpose models that can be used as-is or customized to perform

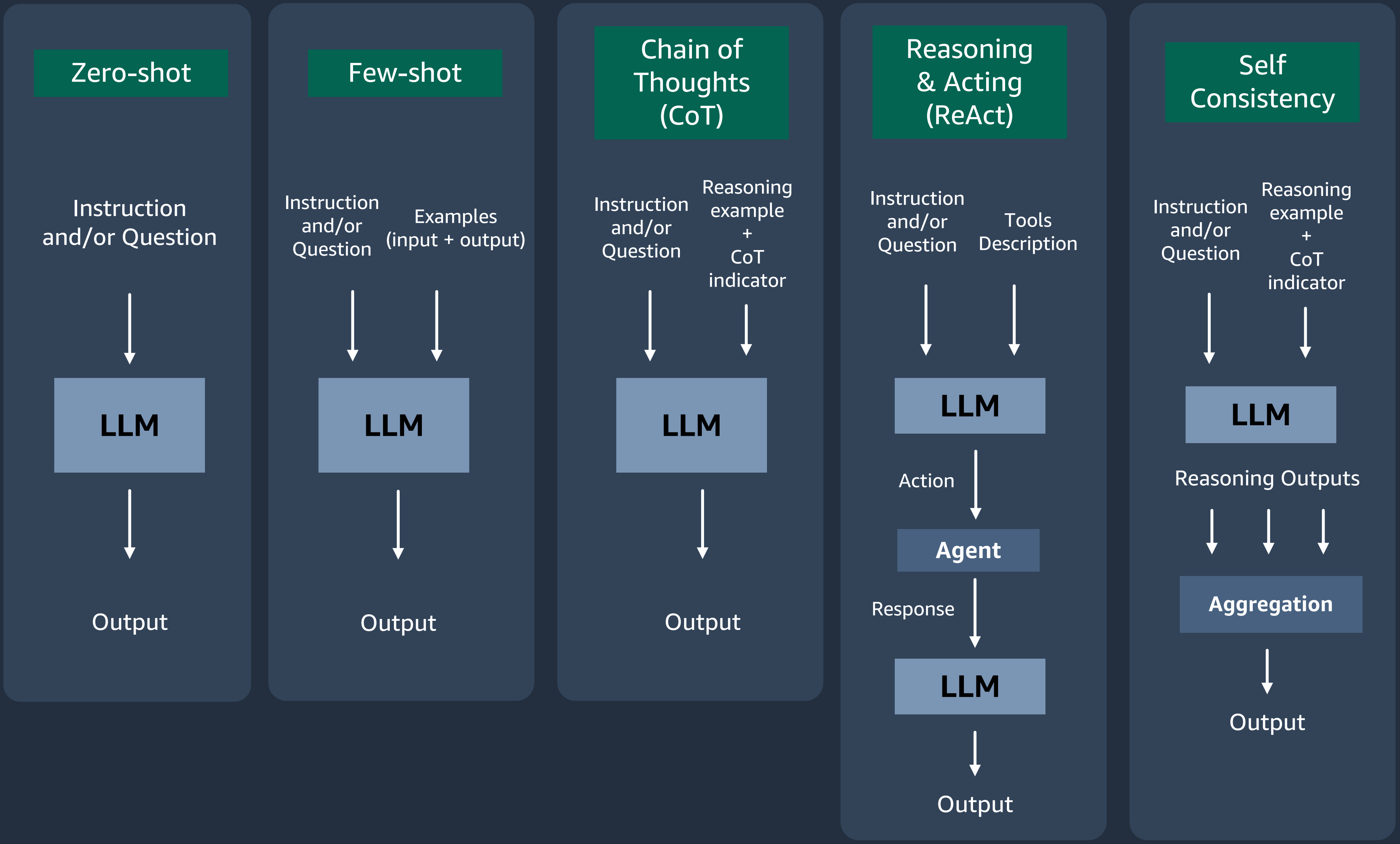
Act as an IT technical expert providing customer service. Consider the Context below to answer the user's questions with a friendly tone. Answer in English in 2 sentences or less providing instructions.

Context: You work in the Support line of a technology company that commercializes Android smartphones.
The user is calling because the phone is not charging

User: Hi, how can I fix my phone?

Assistant:
I will try my best to assist you with this. Can you please tell me the model of your phone and what version of the Android operating system it is running?

Prompt Engineering Techniques



Zero-shot

Instruction and/or Question

LLM

Output

Few-shot

Instruction and/or Question

Examples (input + output)

LLM

Output

Chain of Thoughts (CoT)

Instruction and/or Question

Reasoning example + CoT indicator

LLM

Output

Reasoning & Acting (ReAct)

Instruction and/or Question

Tools Description

LLM

Action

Agent

Response

LLM

Output

Self Consistency

Instruction and/or Question

Reasoning example + CoT indicator

LLM

Reasoning Outputs

Aggregation

Output

Among others like:

- Generate Knowledge Prompting
- Automatic Prompt Engineer
- Active-Prompt
- Directional Stimulus Prompting
- Augmented Language Models
- Multi-Modal CoT
- Graph Prompting

Plus all the possible combinations, e.g.:

- Zero-shot + CoT
- Few-shot + CoT
- Few-shot +ReAct
- etc.

Plus other factors like:

- Aggregations
- Chaining
- etc.

Prompt engineering with Amazon Bedrock

Important parameters

You can set parameters to customize results

- **Temperature**: controls randomness. Lower values pick probable tokens, higher values add randomness and diversity. Use lower for factual responses, higher for creative
- **Top_p**: also adjusts determinism with "nucleus sampling". Lower values give exact answers, higher values give diverse responses

Note:

- Only adjust one parameter at a time.
- Outcomes vary between language model types

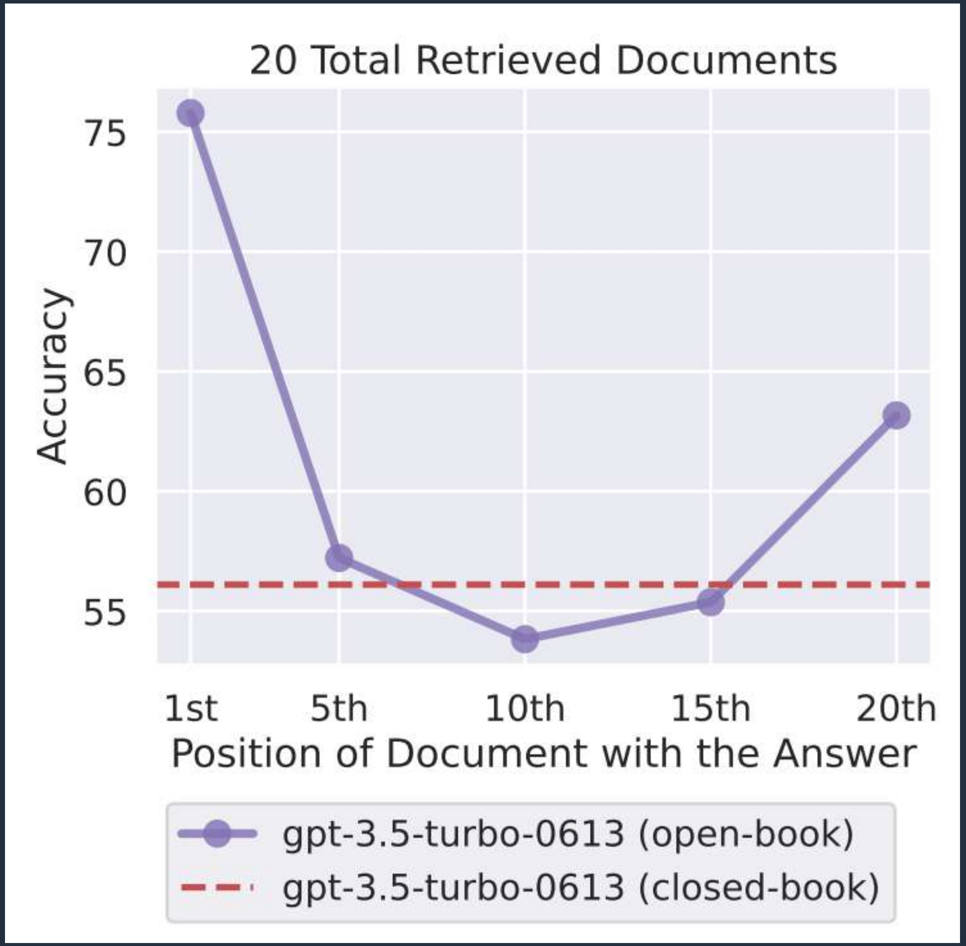
Tips for designing prompts

- Be clear and concise
- Include context if needed
- Use directives for the desired response type
- Consider the output in the prompt
- Provide an example response
- Use simple language
- Test and experiment

Does the Prompt’s format matters?

“Models are better at using relevant information that occurs at the very beginning or end of its input context”

“Performance degrades significantly when models must access and use information located in the middle of its input context”



Source:
Lost in the Middle: How Language Models Use Long Contexts
https://arxiv.org/pdf/2307.03172.pdf

Prompt Guidance for Anthropic Claude

Prompt Guidance for Anthropic Claude

Anthropic Claude

Claude is good at:

1. Dialogue and role play
2. Text summarization and Q&A
3. Translation
4. Retrieval augmented generation (RAG)
5. Coding related tasks
6. Classification, meta data extraction and analysis
7. Text and content generation
8. Content moderation

Anthropic Claude v2

- Max tokens: 100,000+
- **Most powerful model**
- Example tasks
    - Complex reasoning problems
    - High-quality content generation
    - Sophisticated roleplay
    - Code generation

Anthropic Claude instant

- Max tokens: 100,000+
- Smaller and faster model
    - 2x faster, 5x cheaper than Claude 2
- Example tasks:
    - Lightweight roleplay
    - Content moderation
    - Classification

+ Claude v2, and Claude Instant models supporting 100K tokens size are only available in N. Virginia, and Oregon regions at this time. Claude end points in Tokyo, and Frankfurt regions will cap at 44K tokens.

“Human:” / “Assistant:” formatting

- Claude is trained on alternating “Human:” / “Assistant:” dialogue:
  - Human: [Instructions]
  - Assistant: [Claude’s response]

- For any API prompt, you must start with “Human:” and end with “Assistant:”
  - Also include two new lines before each

Examples:

```
¶
¶
Human: Why is the sky blue? ¶
¶
Assistant:
```

** ¶ symbols above shown for illustration*

```


Human: Why are sunsets orange?

Assistant:
```

Being clear and direct

- Claude responds best to **clear and direct instructions**
- When in doubt, follow the **Golden Rule of Clear Prompting:** show your prompt to a friend and ask them if they can follow the instructions themselves and produce the exact result you're looking for

Example:

Human: Write a haiku about robots

Assistant: Here is a haiku about robots:

Metal bodies move
Circuits calculate tasks
Machines mimic life

Human: Write a haiku about robots. **Skip the preamble; go straight into the poem.**

Assistant: Metal bodies move
Circuits calculate tasks
Machines mimic life

Assigning roles (aka role prompting)

- Claude sometimes needs **context about what role it should inhabit**
- Assigning roles changes Claude's response in two ways:
  - **Improved accuracy** in certain situations (such as mathematics)
  - **Changed tone and demeanor** to match the specified role

Example:

Human: How is the U.S. President elected?

Assistant: The president is not voted on directly by the people. Instead, each state gets a certain number of electors based on its population. The states with more people get more electors. When people vote on Election Day in November, they are actually voting for which candidate they want their state's electors to…

Human: **You are a kindergarten teacher.** How is the U.S. President elected?

Assistant: The president is the leader of our country. Every four years, we have a presidential election to choose a new president or re-elect the current president. During the election, people called candidates try to become president by getting votes. On Election Day in November, people go to voting places…

Using XML tags

- Disorganized prompts are hard for Claude to comprehend
- Just like section titles and headers help humans better follow information, **using delineators like XML tags <></> helps Claude understand the prompt's structure**

We recommend you use XML tags, as Claude has been especially trained on XML tags

Example:

Human: Hey Claude. Show up at 6AM because I say so. Make this email more polite.

Assistant: Dear Claude, I hope this message finds you well…

Human: Hey Claude. **<email>**Show up at 6AM because I say so.**</email>** Make this email more polite.

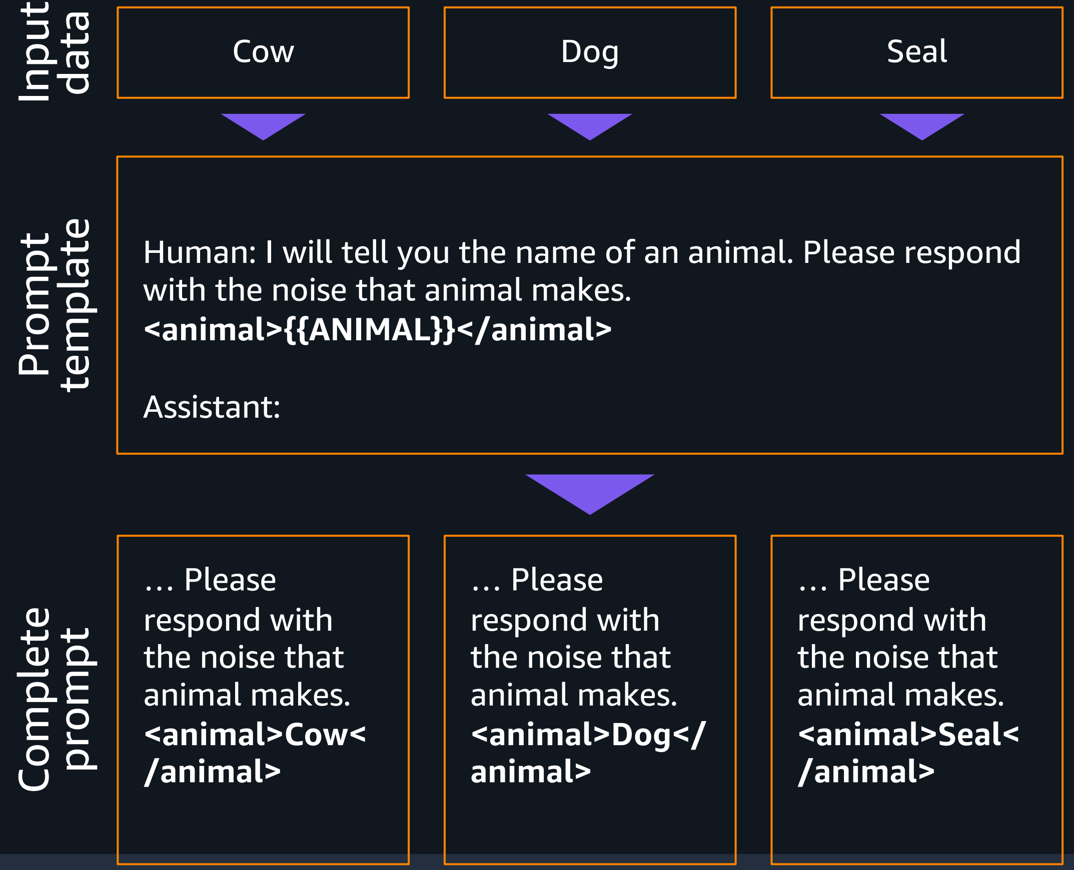
Assistant: Good morning team, I hope you all had a restful weekend…

Separating data from instructions

- Including input data directly in prompts can make prompts overly long and hard to troubleshoot
- Separating prompt structure from input data allows for:
  - **Easier editing of the prompt** itself
  - Much **faster processing of multiple datasets**

Tip: while not always necessary, we recommend using XML tags to separate out your dataset for even easier parsing

Example:



Input data

Cow

Dog

Seal

Prompt template

Human: I will tell you the name of an animal. Please respond with the noise that animal makes.
<animal>{{ANIMAL}}</animal>

Assistant:

Complete prompt

… Please respond with the noise that animal makes. **<animal>Cow</animal>**

… Please respond with the noise that animal makes. **<animal>Dog</animal>**

… Please respond with the noise that animal makes. **<animal>Seal</animal>**

Formatting output & speaking for Claude

- You can get Claude to say exactly what you want by:
  - **Specifying the exact output format** you want
  - **Speaking for Claude** by writing the beginning of Claude's response for it (after "Assistant:")

Example:

Prompt

Human: Please write a haiku about a cat. Put it in <haiku> tags.

Assistant: <haiku>

Claude's response

Curious little cat
Pouncing on strings and lasers
Naps in sunny spots
</haiku>

Thinking step by step

- Claude benefits from having time to think through tasks before executing
- Especially if a task is particularly complex, **tell Claude to think step by step before it answers**

Increases intelligence of responses but also increases latency by adding to the length of the output.

Example:

Human: Are both the directors of Jaws and Casino Royale from the same country?

Assistant: "Okay, let's think through this step-by-step:

1. The director of Jaws was Steven Spielberg. Steven Spielberg is American.

2. The director of Casino Royale (2006) was Martin Campbell. Martin Campbell is from New Zealand.

3. America and New Zealand are different countries.

Therefore, the directors of Jaws and Casino Royale are not from the same country, as Steven Spielberg is American and Martin Campbell is from New Zealand."

Using examples

- **Examples are probably the single most effective tool** for getting Claude to behave as desired
- Make sure to **give Claude examples of common edge cases.**
- Generally **more examples = more reliable responses** at the cost of latency and tokens

Example:

Human: Quote:
“When the reasoning mind is forced to confront the impossible again and again, it has no choice but to adapt.”
— N.K. Jemisin, The Fifth Season
Author: N.K. Jemisin

Quote:
“Some humans theorize that intelligent species go extinct before they can expand into outer space. If they're correct, then the hush of the night sky is the silence of the graveyard.”
— Ted Chiang, Exhalation
Author:

Assistant: Ted Chiang

Sometimes, examples are so powerful that they don’t even need to be accompanied by task instructions!

Prompt Guidance for Amazon Titan

Prompt Guidance for Titan Text

Amazon Titan Express

Remains in Limited Preview

Titan is good at:

1. Dialogue and role play
2. Text summarization and Q&A
3. Translation
4. Retrieval augmented generation (RAG)
5. Simple code generation
6. Classification, meta data extraction and analysis
7. Text and content generation

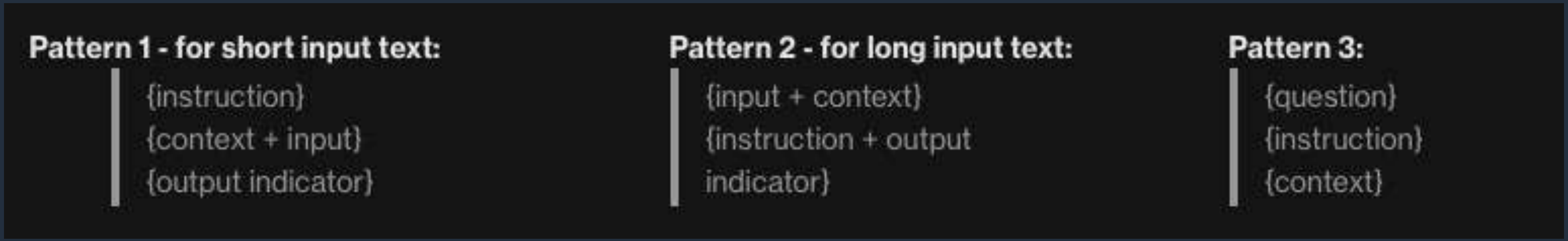
Max tokens: 8000

modelID: amazon.titan-text-express-v1

Prompt components

- The following pattern works well with Titan
  - Instruction
    - Your ask from the model (e.g. summarize, answer from the text)
  - Context
    - Sets the scene for the model (e.g. the following is a medical text)
  - Input
    - Input text for the task
  - Output indicator
    - Trigger for the model to generate the output (e.g. Output, Summary, Assistant, etc.)

Prompt and Output Guidance



Pattern 1 - for short input text:

{instruction}
{context + input}
{output indicator}

Pattern 2 - for long input text:

{input + context}
{instruction + output indicator}

Pattern 3:

{question}
{instruction}
{context}

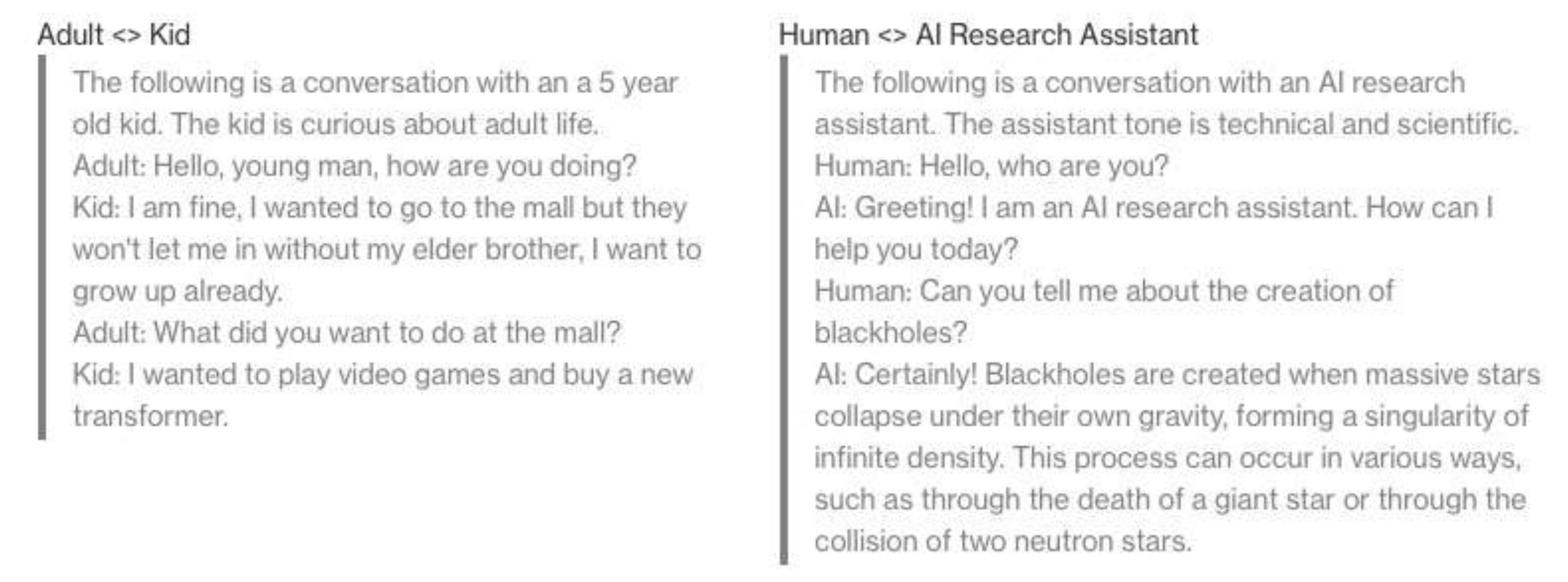
- Outputs concise and short answers by default
  - Usually single line or paragraph
  - Can get more detailed answers by adding instructions in the prompt
  - For longer {context + input} it's better to provide the instruction or output indicator at the end to get better results

Prompt and Output Guidance (continued)

- Specify output length in the instruction
  - Be specific with number of sentences, bullet points, paragraphs
  - Number of words seems to be a weaker guardrail
- Can provide default output if the model is unsure about the answers
  - Example:
    - Answer the question based on the context below. Keep the answer short. Respond "Unsure about answer" if not sure about the answer

Role Based Conversation

The model can carry out conversations in different roles



Adult <> Kid

> The following is a conversation with an a 5 year old kid. The kid is curious about adult life.
> Adult: Hello, young man, how are you doing?
> Kid: I am fine, I wanted to go to the mall but they won't let me in without my elder brother, I want to grow up already.
> Adult: What did you want to do at the mall?
> Kid: I wanted to play video games and buy a new transformer.

Human <> AI Research Assistant

> The following is a conversation with an AI research assistant. The assistant tone is technical and scientific.
> Human: Hello, who are you?
> AI: Greeting! I am an AI research assistant. How can I help you today?
> Human: Can you tell me about the creation of blackholes?
> AI: Certainly! Blackholes are created when massive stars collapse under their own gravity, forming a singularity of infinite density. This process can occur in various ways, such as through the death of a giant star or through the collision of two neutron stars.

Jurassic 2

Jurassic-2 Mid

Mid-sized model, carefully designed to strike the right balance between exceptional quality and affordability. Jurassic-2 Mid can be applied to any language comprehension or generation task including question answering, summarization, long-form copy generation, advanced information extraction and many others.

Max Tokens: 8,191

modelID: ai21.j2-mid-v1

Jurassic – 2 Ultra

AI21's most powerful model offering exceptional quality. Apply Jurassic-2 Ultra to complex tasks that require advanced text generation and comprehension. Popular use cases include question answering, summarization, long-form copy generation, advanced information extraction, and more.

Max Tokens: 8,191

modelID: ai21.j2-ultra-v1

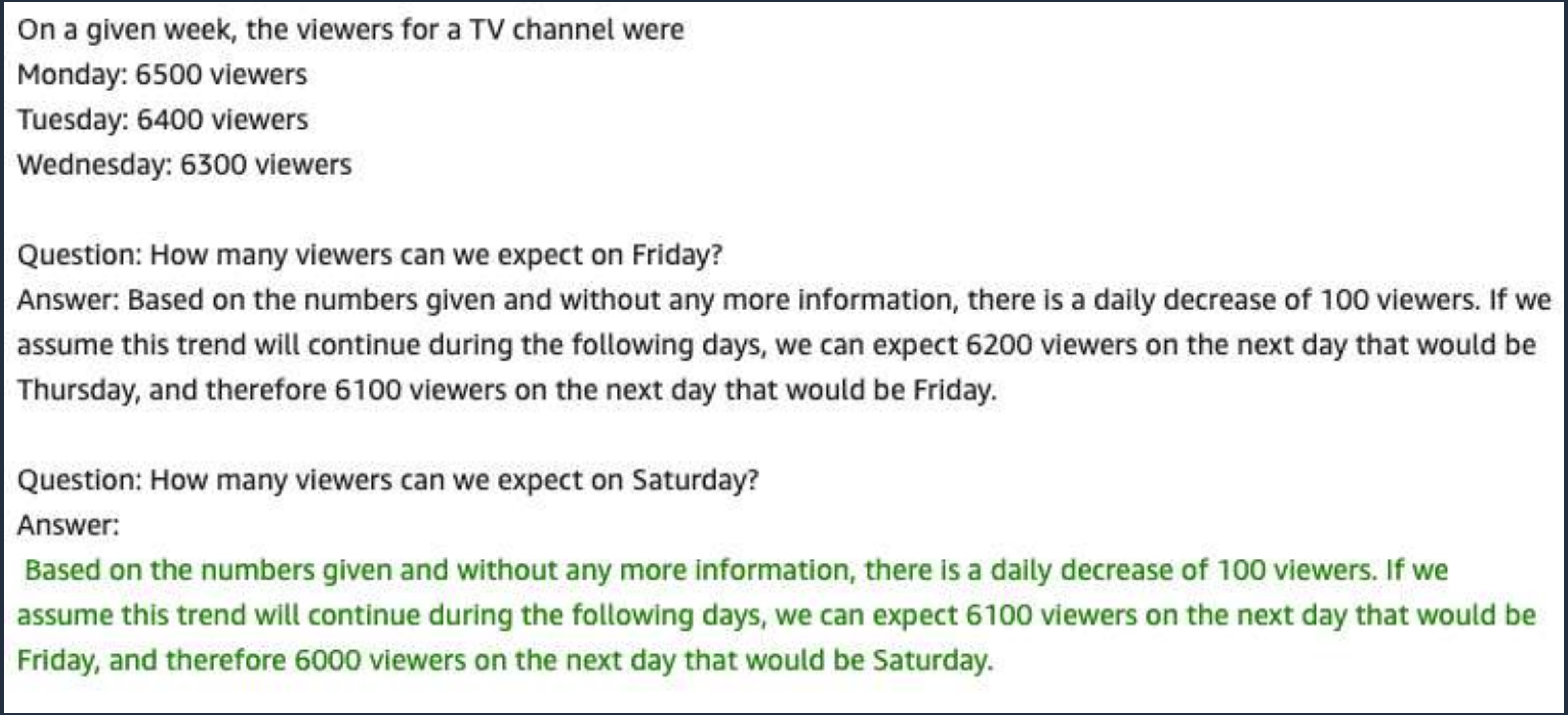
Prompt Guidance for Jurassic Instruct

- Tasks that work well with Jurassic Instruct
  - Text summarization
  - Text generation
  - Question and answering
- Works very good with zero-shot as well as few-shot prompting methods
- Provides shorter answers by default
  - For longer answers, ask explicitly. E.g. x number of sentences

Prompt components

- The following pattern works well with Jurassic Instruct
  - Specify instruction
    - Your ask from the model (e.g. summarize, answer from the text)
    - Include short but clear description of the task
  - Include additional context
    - Sets the scene for the model (e.g. the following is a medical text)
  - Avoid negative formulation
    - E.g. avoid instructions such as “no more than five sentences”
  - Switch order for long inputs
    - If input text is long, try providing text first and then instructions
  - Output indicator
    - Trigger for the model to generate the output (e.g. Output, Summary, Assistant, etc.)

Example – Question and Answering



On a given week, the viewers for a TV channel were
Monday: 6500 viewers
Tuesday: 6400 viewers
Wednesday: 6300 viewers

Question: How many viewers can we expect on Friday?
Answer: Based on the numbers given and without any more information, there is a daily decrease of 100 viewers. If we assume this trend will continue during the following days, we can expect 6200 viewers on the next day that would be Thursday, and therefore 6100 viewers on the next day that would be Friday.

Question: How many viewers can we expect on Saturday?
Answer:
Based on the numbers given and without any more information, there is a daily decrease of 100 viewers. If we assume this trend will continue during the following days, we can expect 6100 viewers on the next day that would be Friday, and therefore 6000 viewers on the next day that would be Saturday.

Prompt Guidance for Stable Diffusion

stability.ai

Stability.ai Stable Diffusion

Remains in Limited Preview

1. Start with the Subject "An image of …"
2. Provide Details including:
   - Medium
   - Artist *(if relevant)*
   - Website *(if relevant)*
   - Color
   - Lightning
   - Remarks
   - Adjectives
   - or Resolution
3. Specify the Style
4. Use Text Weights e.g.: "cat:1.5"
5. Use Negative Prompts
6. Use detailed & specific prompts
7. Rely on parameters

Prompt engineering for Stable diffusion

- Max tokens: 8192
- Provide as detailed and specific prompts as possible:
  - portrait photo of an African old warrior chief, tribal panther make up, black on gold, front profile, looking straight into the camera, serious eyes, 50mm portrait photography, hard rim lighting photography–beta –ar 2:3 –beta

Demo

Prompt Engineering on Amazon Bedrock

Security

Data privacy

You are always in control of your data

- Customer data is not used to improve Amazon Titan models for other customers, and is not shared with other foundation model providers
- Customer data (prompts, responses, fine-tuned models) remain in the region where they are created

Data security

You are always in control of your data

- Support for **AWS PrivateLink** so customers can establish private connectivity between virtual private clouds (VPCs) and the Bedrock service using VPC Endpoints
- Integration with AWS Identity and Access Management Service (IAM) to manage inference access, deny access for specific models, and enable Console access
- You can use CloudTrail to monitor API activity and troubleshoot issues as you build solutions
- Fine-tuned (customized) models are encrypted and stored using service managed keys; only you have access to your customized models through an endpoint
- Support for **Customer Managed Keys (CMK)** so customers can create and control keys to encrypt fine-tuned models
- Support for VPC configuration of fine-tuning jobs

Data protection

ENCRYPTION

CLIENT

HTTPS/TLS1.2/TLS1.3

AMAZON BEDROCK

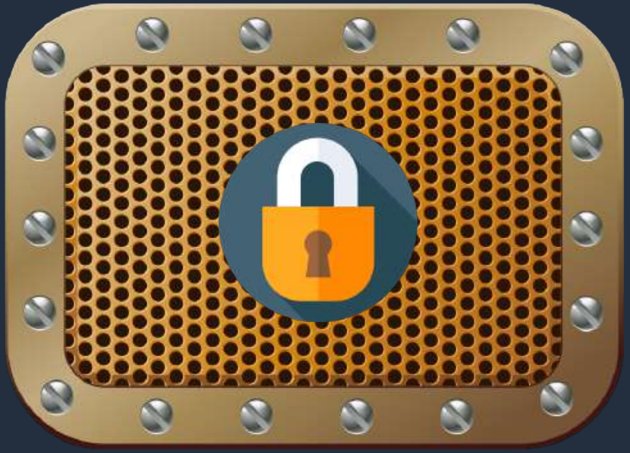
SSE-BEDROCK (Amazon Bedrock managed keys)

1 Client-side

2 In transit

3 At rest

Configurable security controls



Data privacy

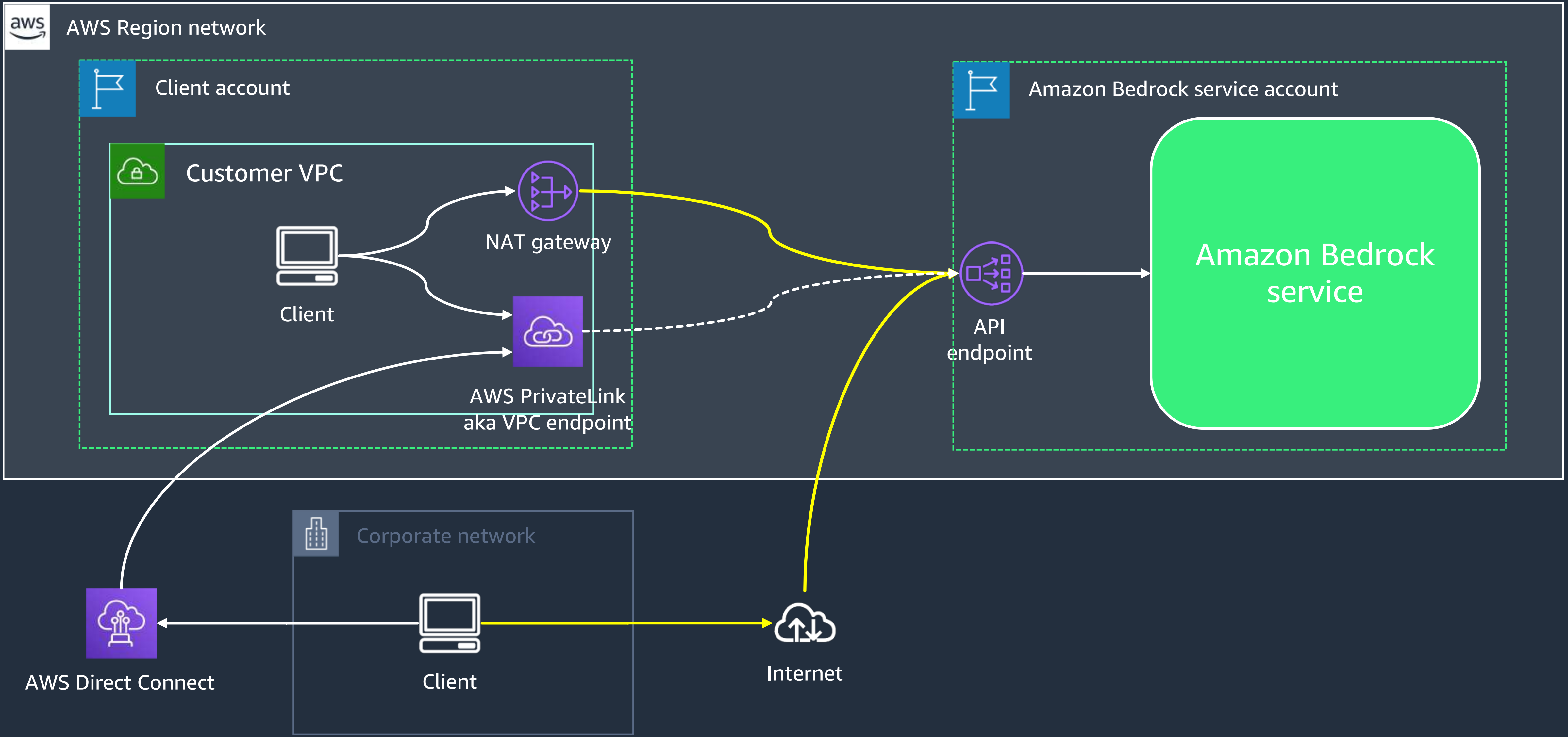
Model tenancy

AWS KMS data encryption

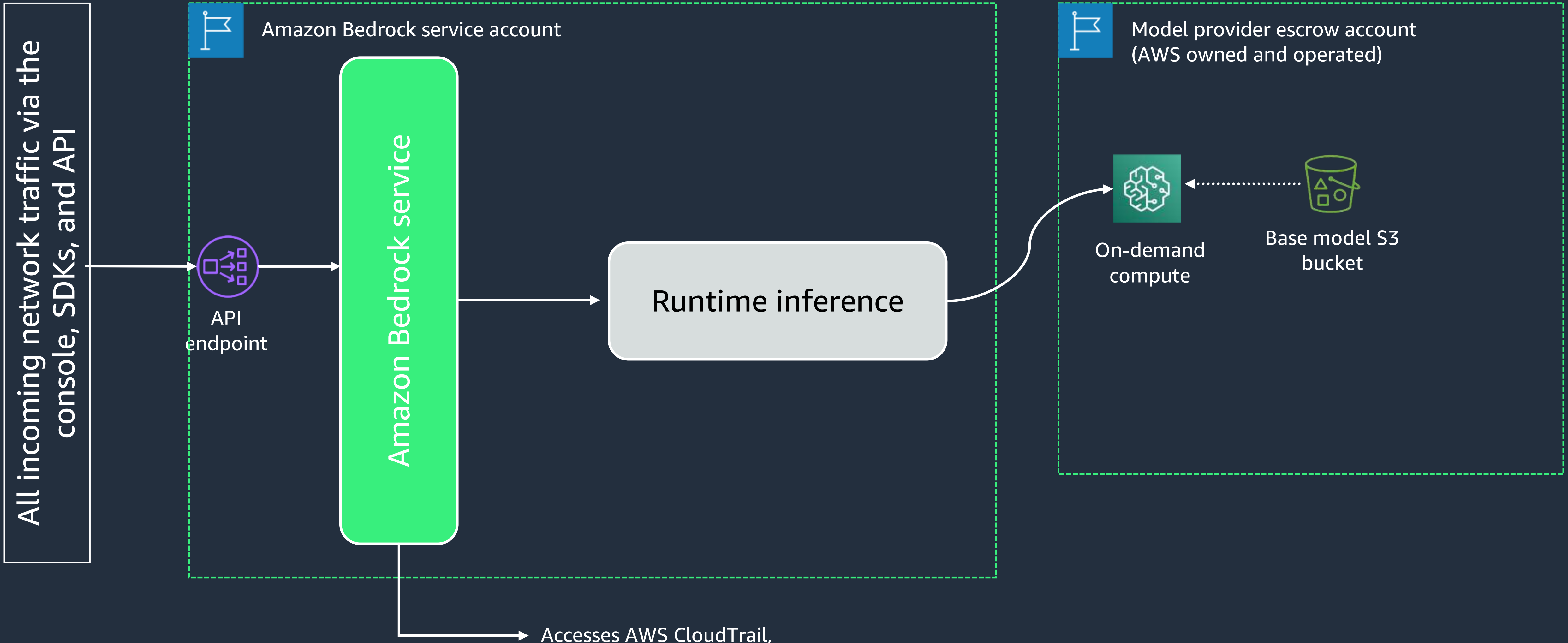
Model fine tuning

Access management

Client connectivity

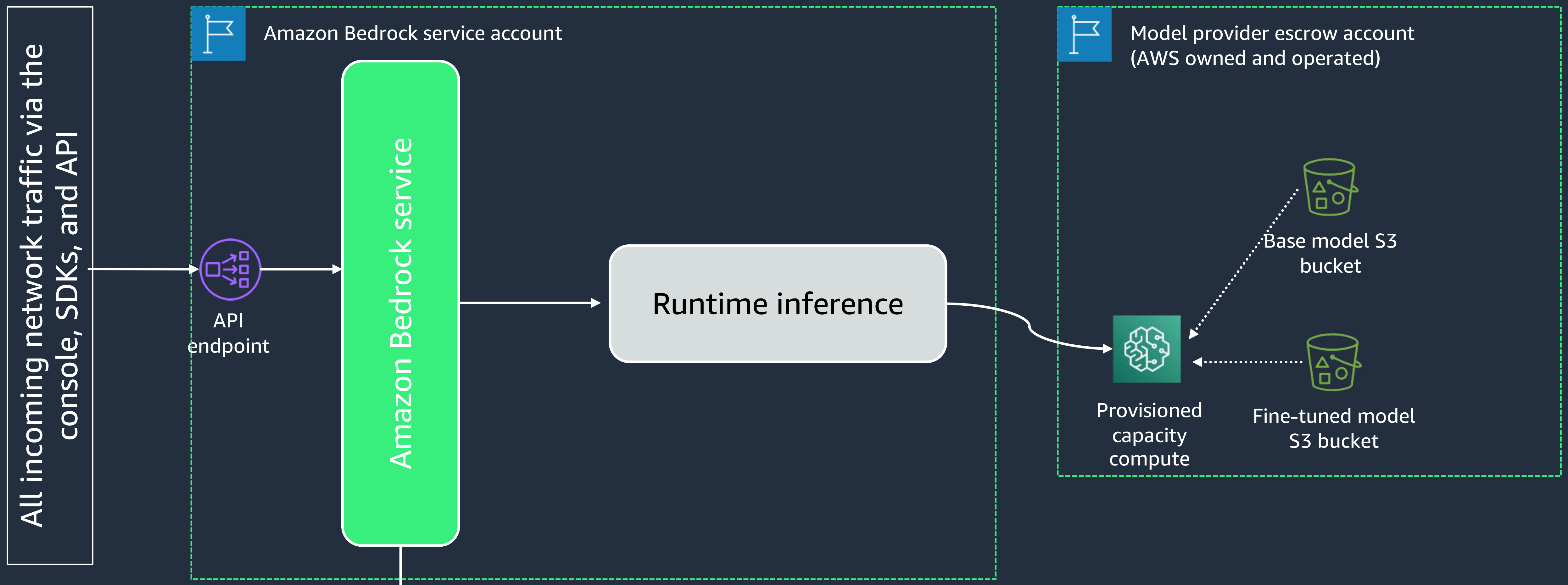


On-demand compute inference



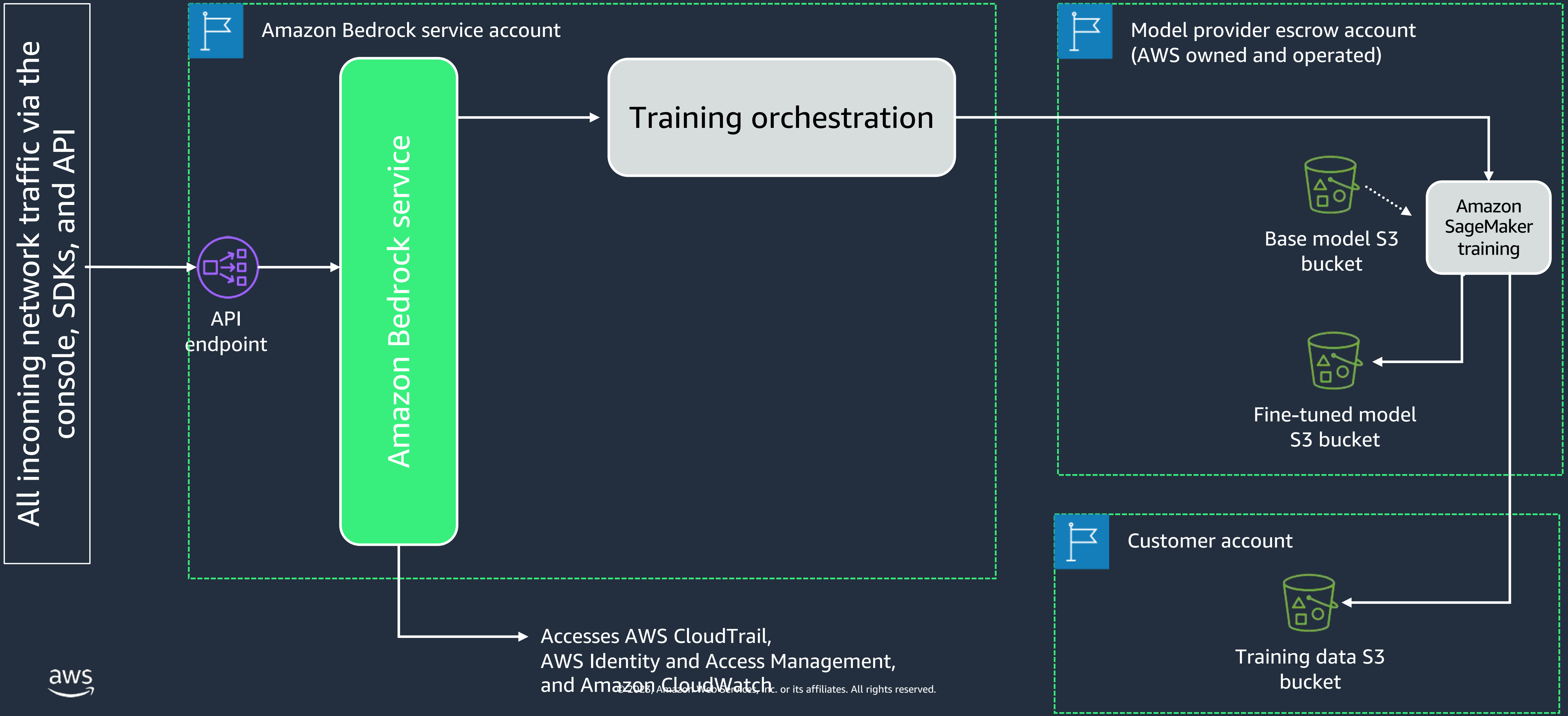
Accesses AWS CloudTrail,
AWS Identity and Access Management,
and Amazon CloudWatch

Provisioned Capacity Compute Inference



Accesses AWS CloudTrail,
AWS Identity and Access Management,
and Amazon CloudWatch

Model fine tuning



All incoming network traffic via the console, SDKs, and API

Amazon Bedrock service account

API endpoint

Amazon Bedrock service

Training orchestration

Model provider escrow account (AWS owned and operated)

Base model S3 bucket

Amazon SageMaker training

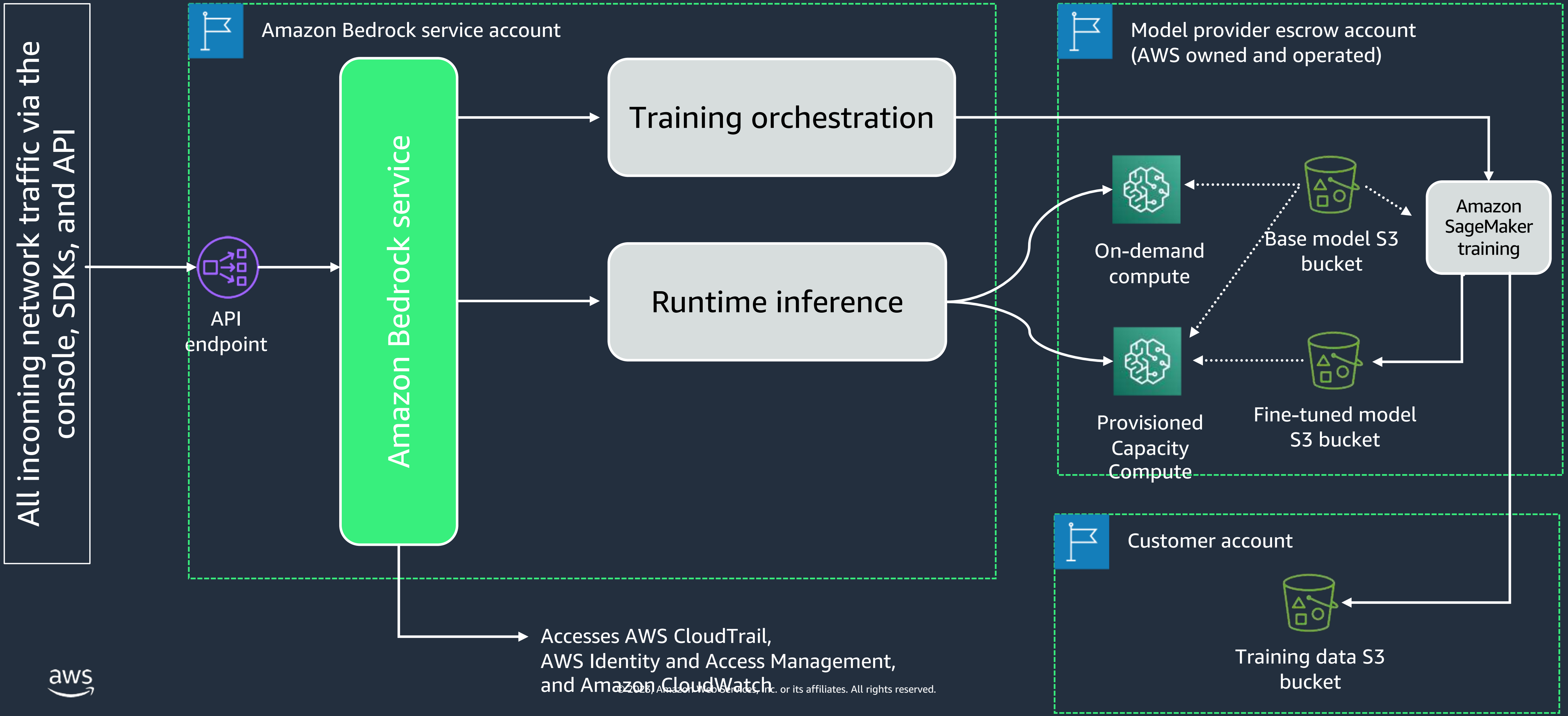
Fine-tuned model S3 bucket

Customer account

Training data S3 bucket

Accesses AWS CloudTrail,
AWS Identity and Access Management,
and Amazon CloudWatch

Complete architecture overview



All incoming network traffic via the console, SDKs, and API

Amazon Bedrock service account

API endpoint

Amazon Bedrock service

Training orchestration

Runtime inference

Model provider escrow account (AWS owned and operated)

On-demand compute

Base model S3 bucket

Amazon SageMaker training

Provisioned Capacity Compute

Fine-tuned model S3 bucket

Customer account

Training data S3 bucket

Accesses AWS CloudTrail, AWS Identity and Access Management, and Amazon CloudWatch

AWS Identity and Access Management

IAM

- Identity-based policies
- Actions
- Resources
- Tags (ABAC)

AWS IAM Fine Grained Access Controls

AMAZON BEDROCK IDENTITY BASED POLICIES

IAM Policy

```
{
   "Version": "2012-10-17",
   "Statement": [
      {
         "Sid": "BedrockConsole",
         "Effect": "Allow",
         "Action": [
           "bedrock:ListFoundationModels",
           "bedrock:InvokeModel”
          ],
         "Resource": "*"
      }
]
```

Action(s)

Resource(s)

Condition Key(s)

IAM/SCP – Example deny policy

```
{
    "Version": "2012-10-17",
    "Statement":
    {
        "Sid": "DenyInferenceForModelX",
        "Effect": "Deny",
        "Action": "bedrock:InvokeModel",
        "Resource": "arn:aws:bedrock:::foundation-model/<name-of-model>"
    }
}
```

Demo

CloudTrail and CloudWatch logs

Architecture Patterns

Architecture patterns in:

- Text generation
- Chatbot
- Summarization
- Image generation
- Question answering

Text Generation

WITH SIMPLE PROMPT

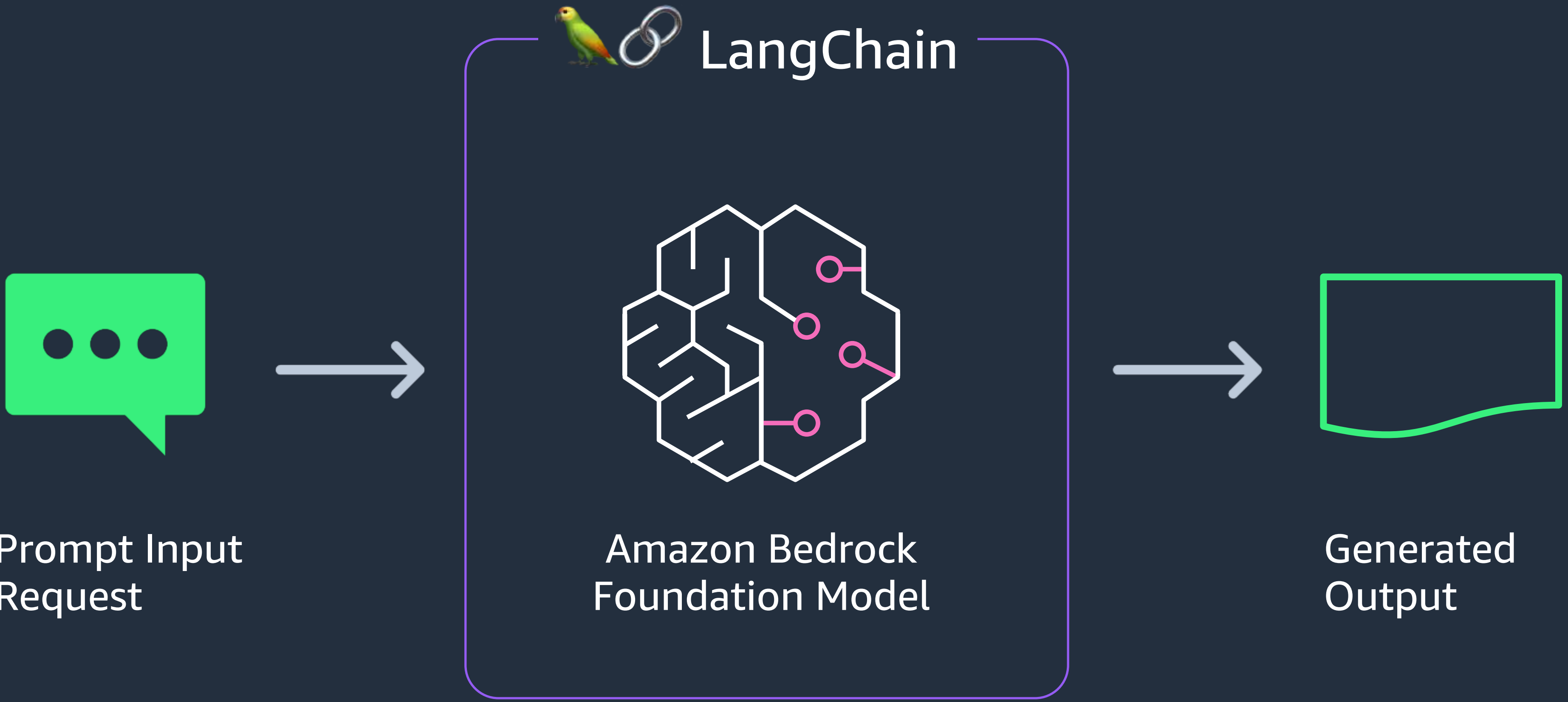
Prompt Input Request

Amazon Bedrock Foundation Model

Generated Output

Text Generation

WITH LANGCHAIN



🦜🔗 LangChain

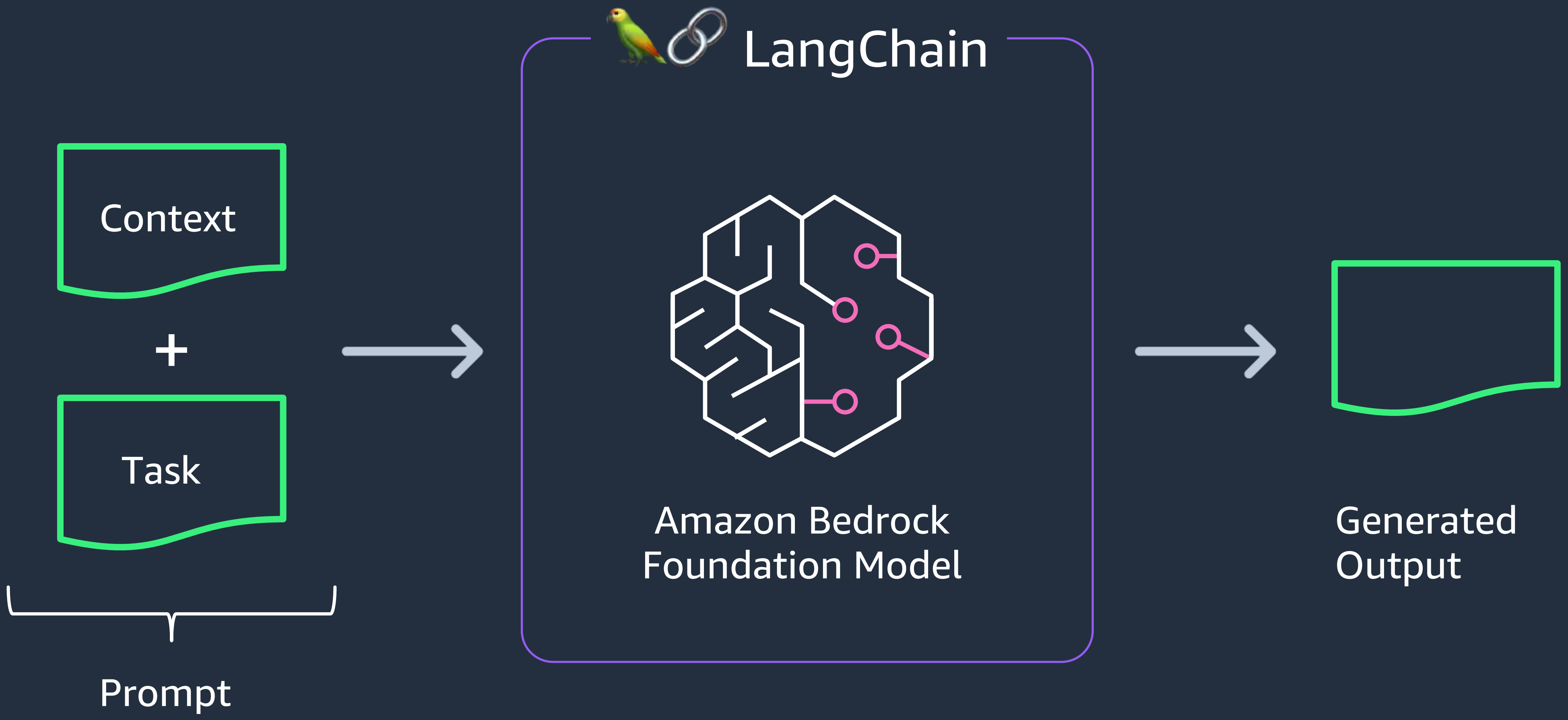
Prompt Input Request

Amazon Bedrock Foundation Model

Generated Output

Text Generation

WITH CONTEXT AND LANGCHAIN



Context

+

Task

Prompt

🦜🔗 LangChain

Amazon Bedrock
Foundation Model

Generated
Output

Text Summarization

WITH SMALL FILES

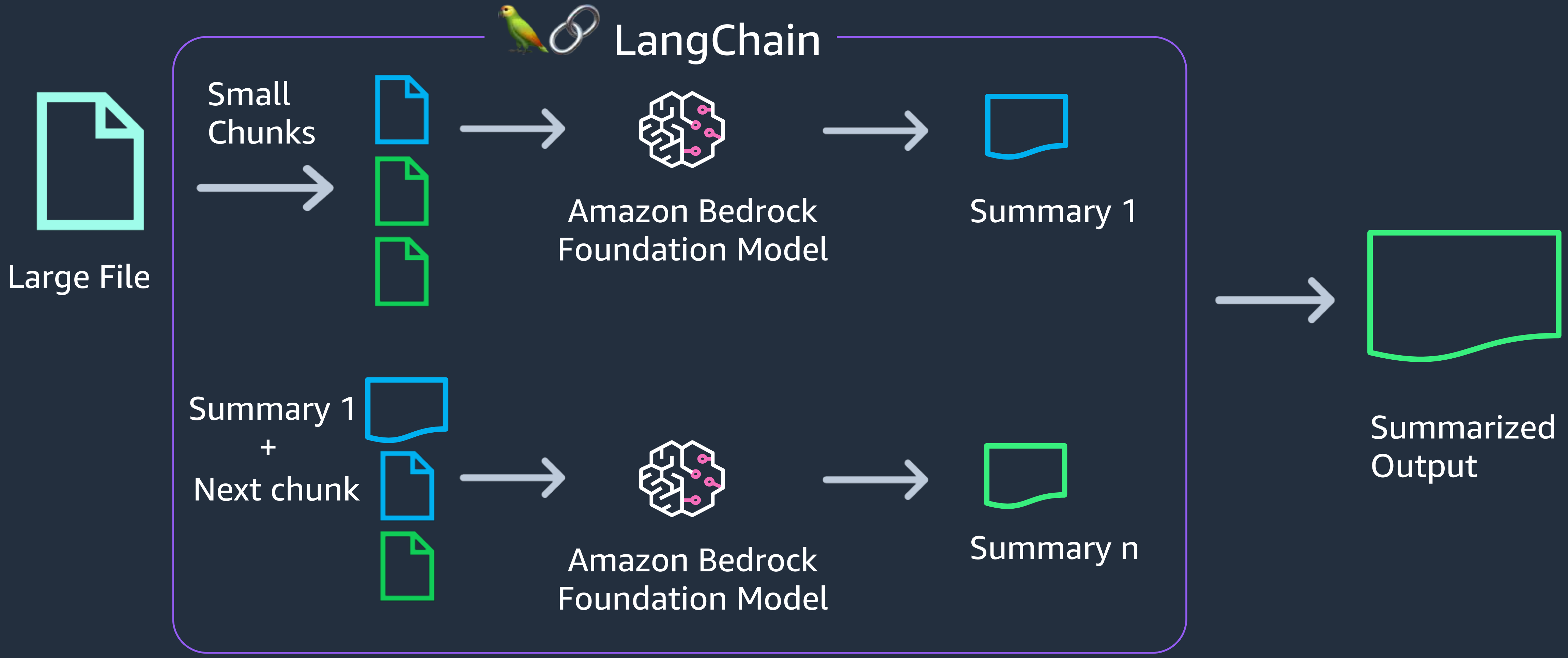
Small file

Amazon Bedrock Foundation Model

Summarized Output

Text Summarization

WITH LARGE FILES AND LANGCHAIN

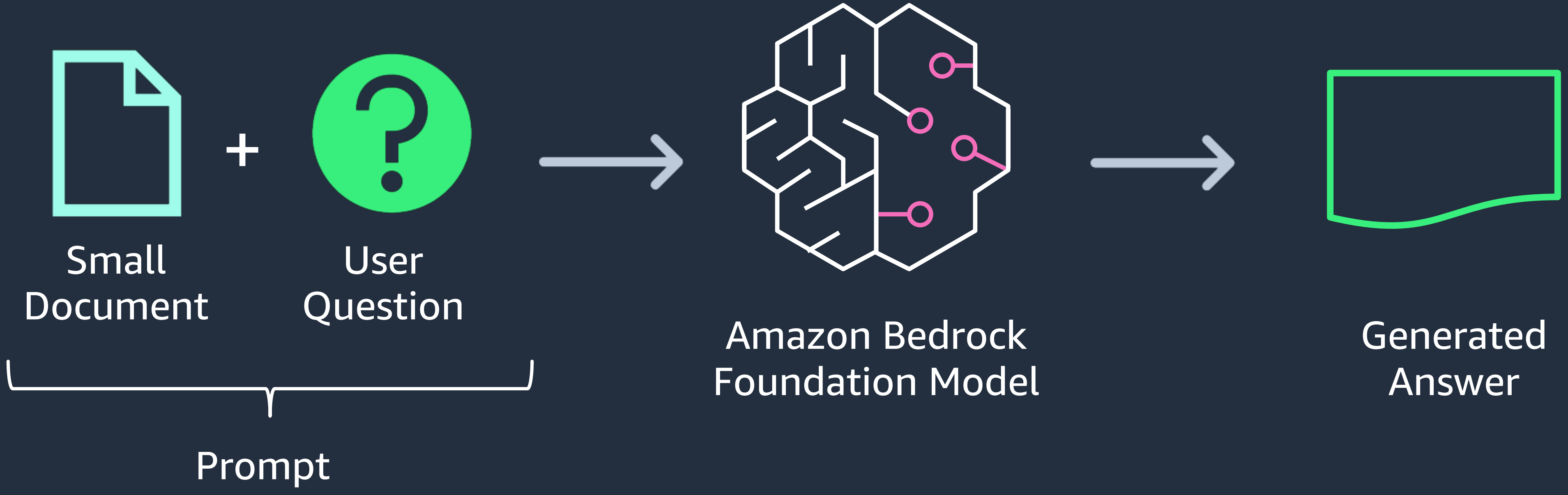


Question Answering

WITH SIMPLE PROMPT

Question Answering

WITH CONTEXT



Small Document

+

User Question

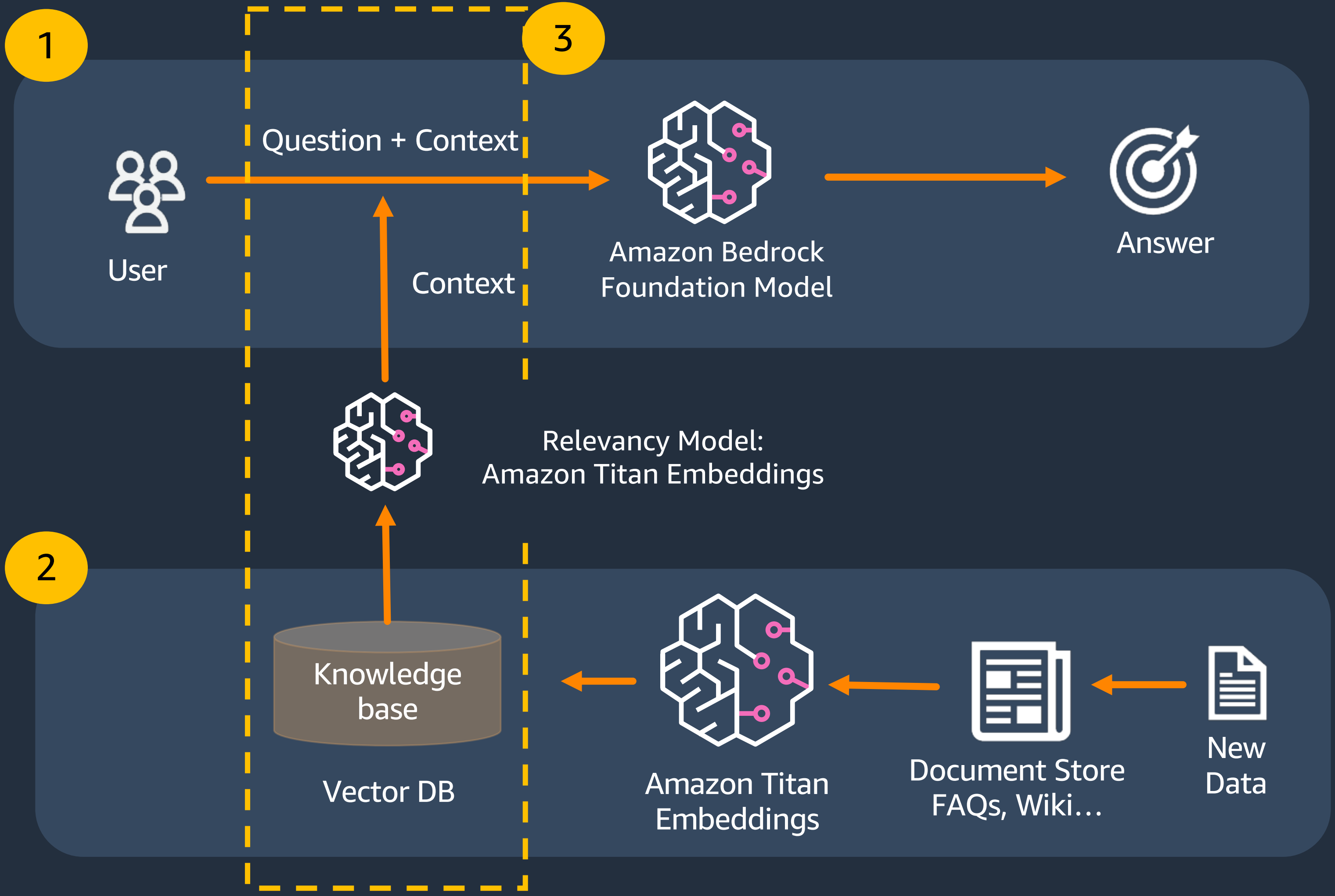
Prompt

Amazon Bedrock Foundation Model

Generated Answer

Question Answering

WITH RETRIEVAL-AUGMENTED GENERATION

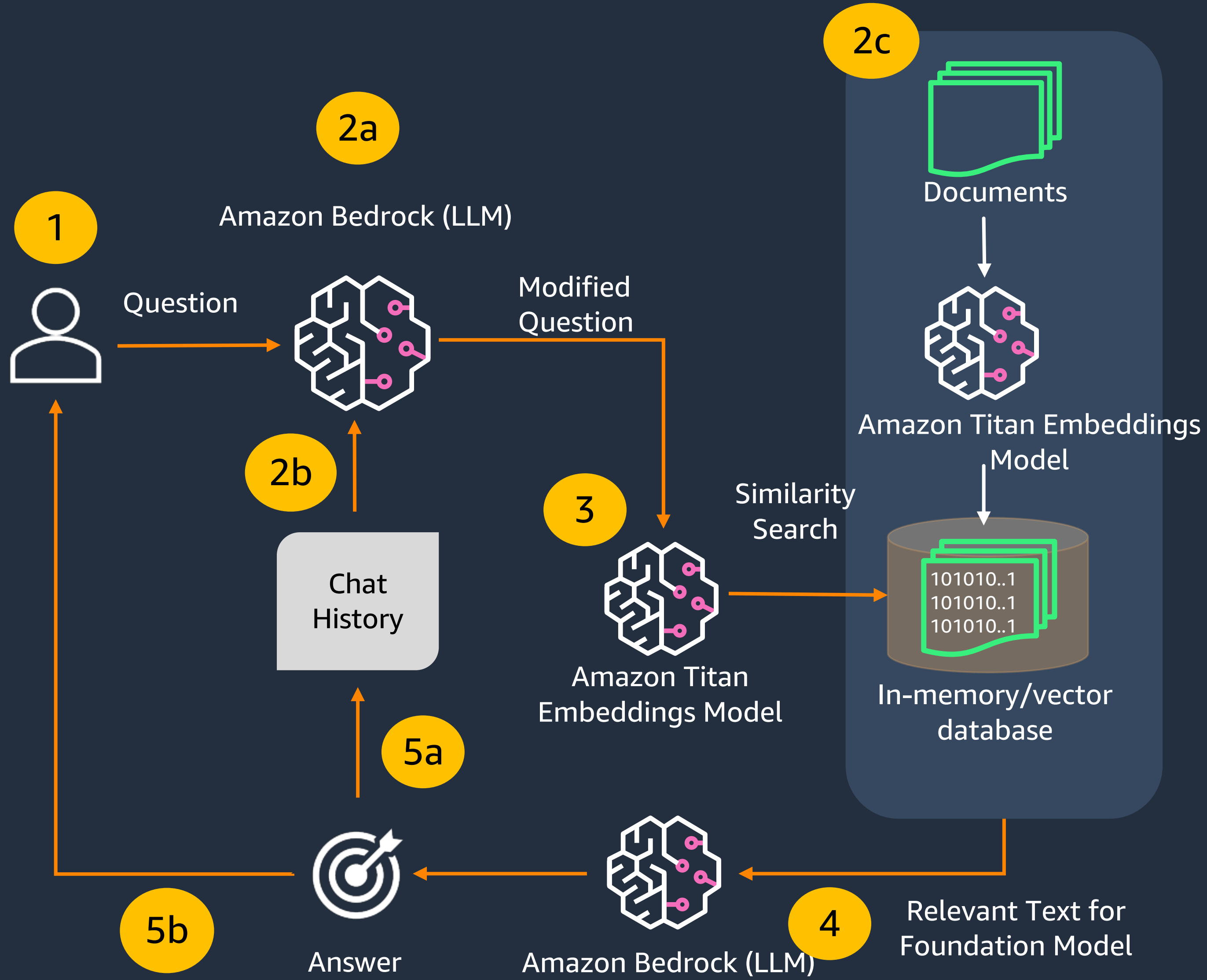


Chatbot

BASIC

Chatbot

WITH CONTEXT



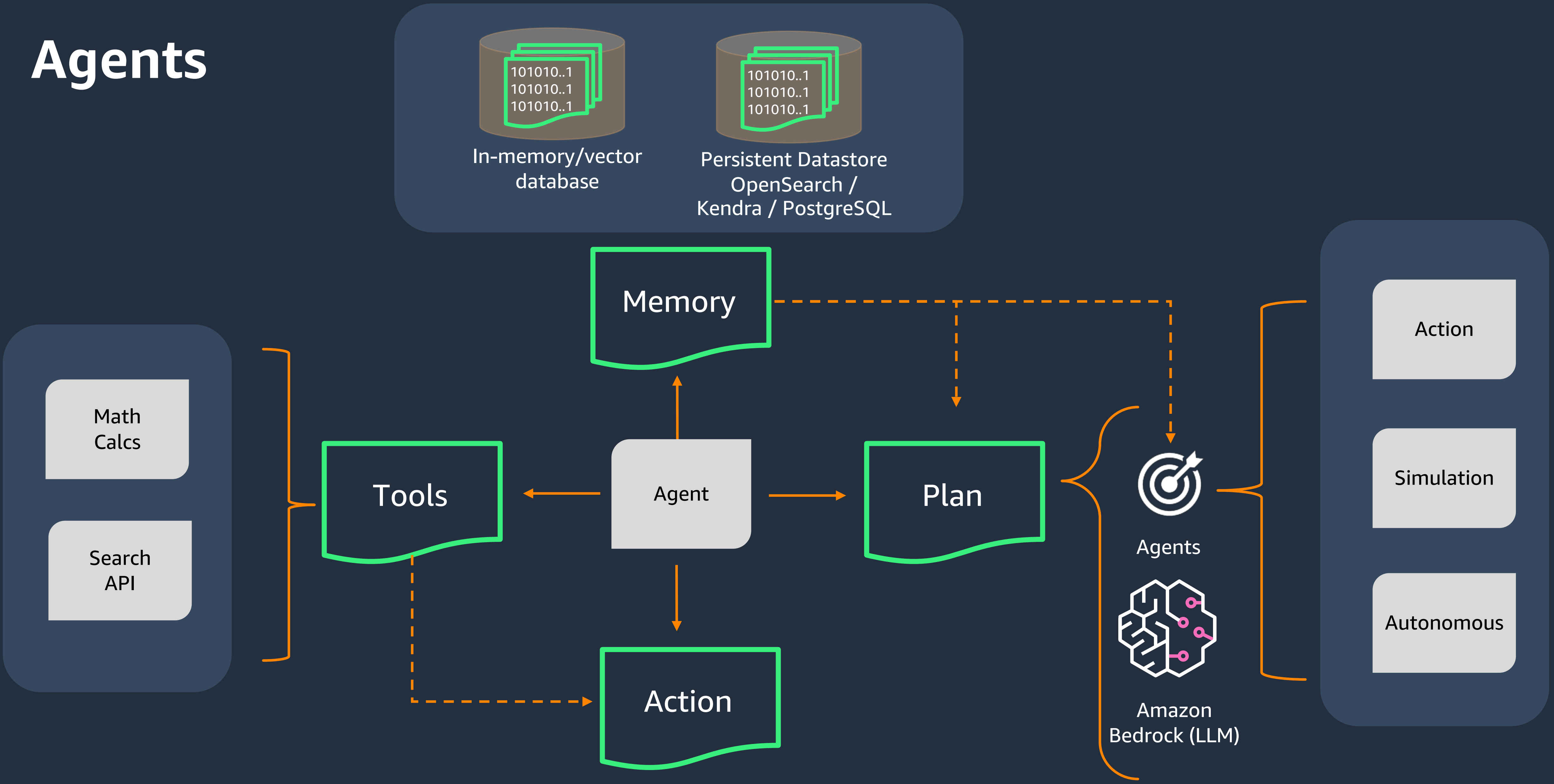
Text to Image

Prompt Input

Amazon Bedrock Stable Diffusion XL

New Image

Agents



In-memory/vector database

Persistent Datastore OpenSearch / Kendra / PostgreSQL

Memory

Math Calcs

Search API

Tools

Agent

Plan

Action

Agents

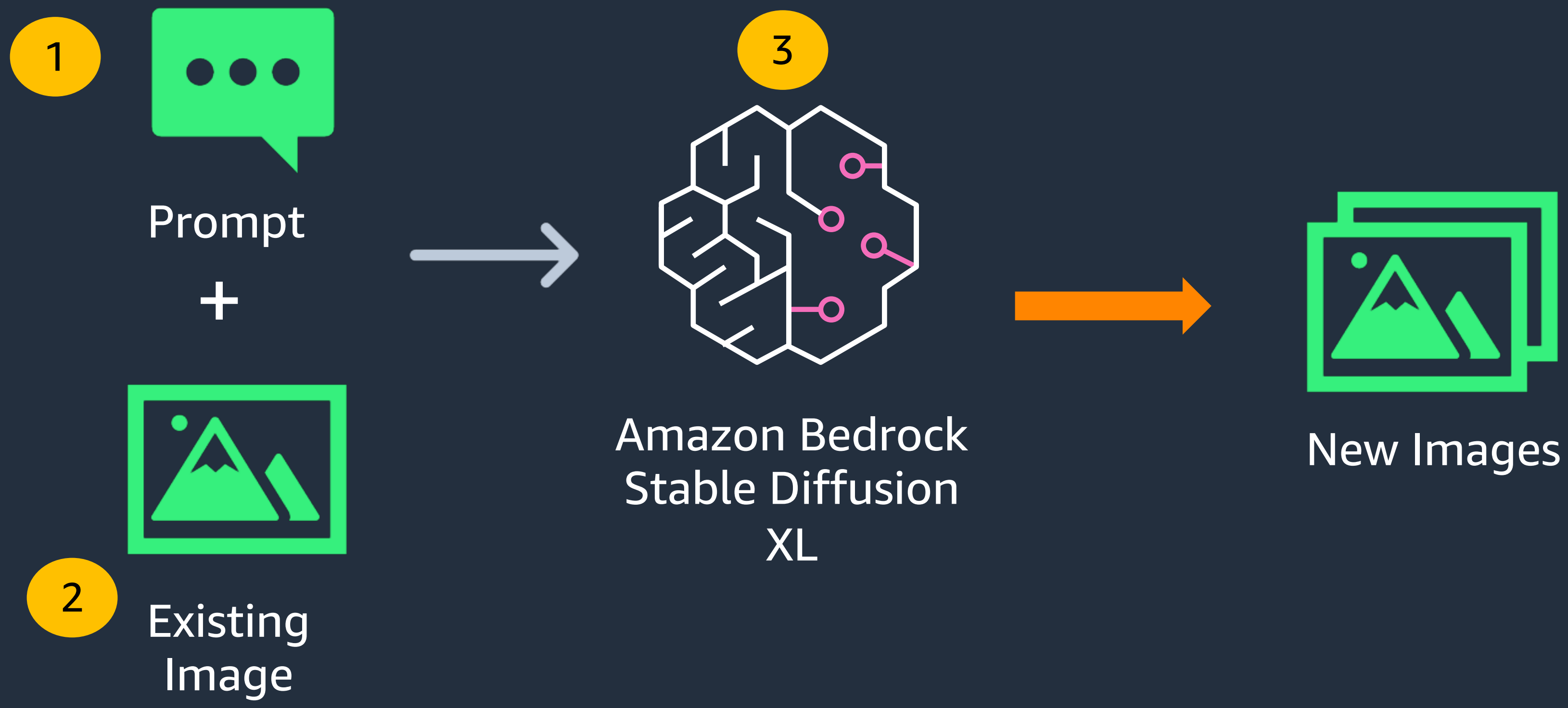
Amazon Bedrock (LLM)

Action

Simulation

Autonomous

Image to Image (In-painting)



1 Prompt

\+

2 Existing Image

3 Amazon Bedrock Stable Diffusion XL

New Images

Bedrock

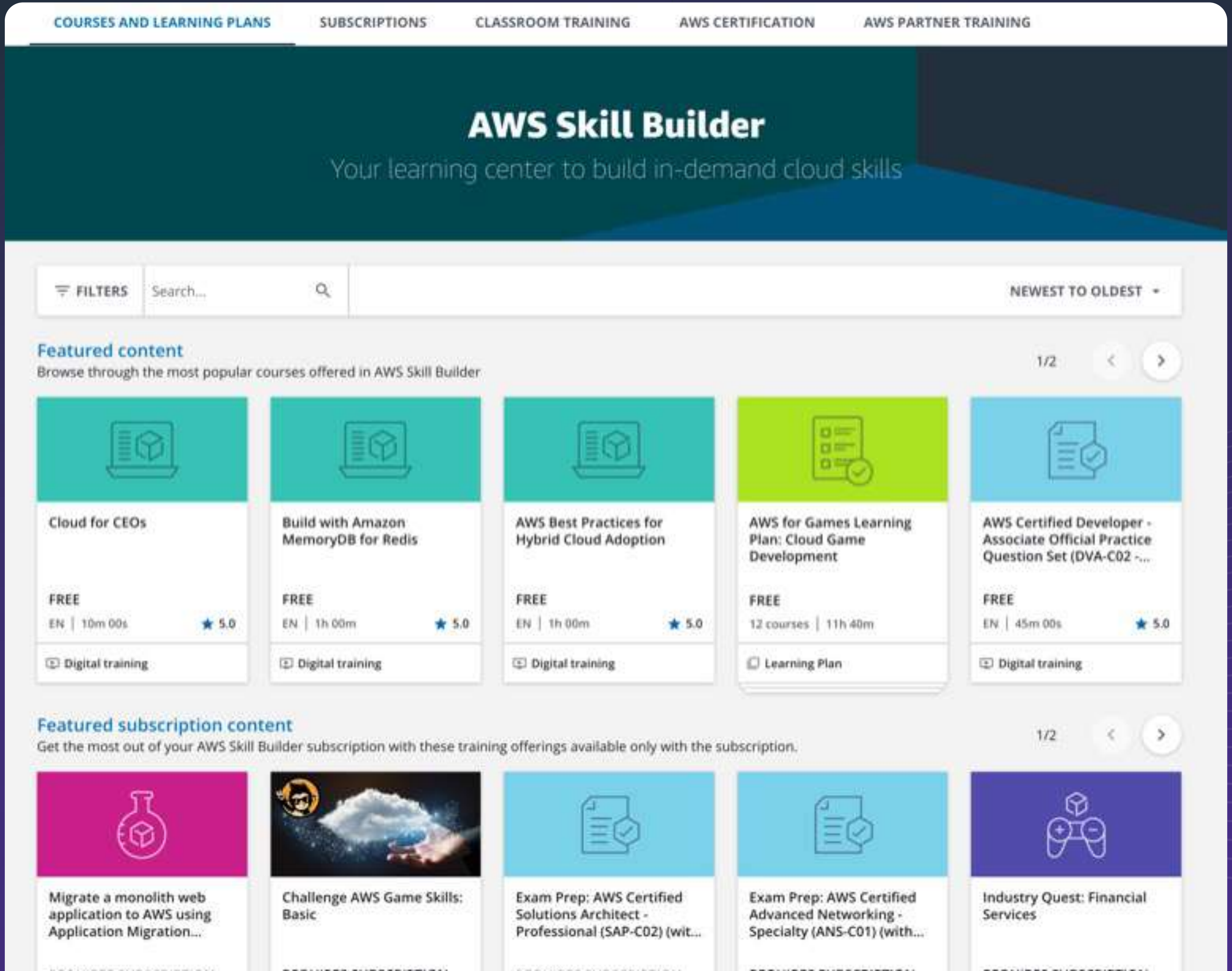
Workshop Studio

Github Repository

Resources

SKILLBUILDER.AWS

600+ digital courses



COURSES AND LEARNING PLANS
SUBSCRIPTIONS
CLASSROOM TRAINING
AWS CERTIFICATION
AWS PARTNER TRAINING

AWS Skill Builder

Your learning center to build in-demand cloud skills

FILTERS
Search...
NEWEST TO OLDEST

Featured content

Browse through the most popular courses offered in AWS Skill Builder

1/2

Title	Price	Details	Rating	Type
Cloud for CEOs	FREE	EN | 10m 00s	★ 5.0	Digital training
Build with Amazon MemoryDB for Redis	FREE	EN | 1h 00m	★ 5.0	Digital training
AWS Best Practices for Hybrid Cloud Adoption	FREE	EN | 1h 00m	★ 5.0	Digital training
AWS for Games Learning Plan: Cloud Game Development	FREE	12 courses | 11h 40m		Learning Plan
AWS Certified Developer - Associate Official Practice Question Set (DVA-C02 -...	FREE	EN | 45m 00s	★ 5.0	Digital training

Featured subscription content

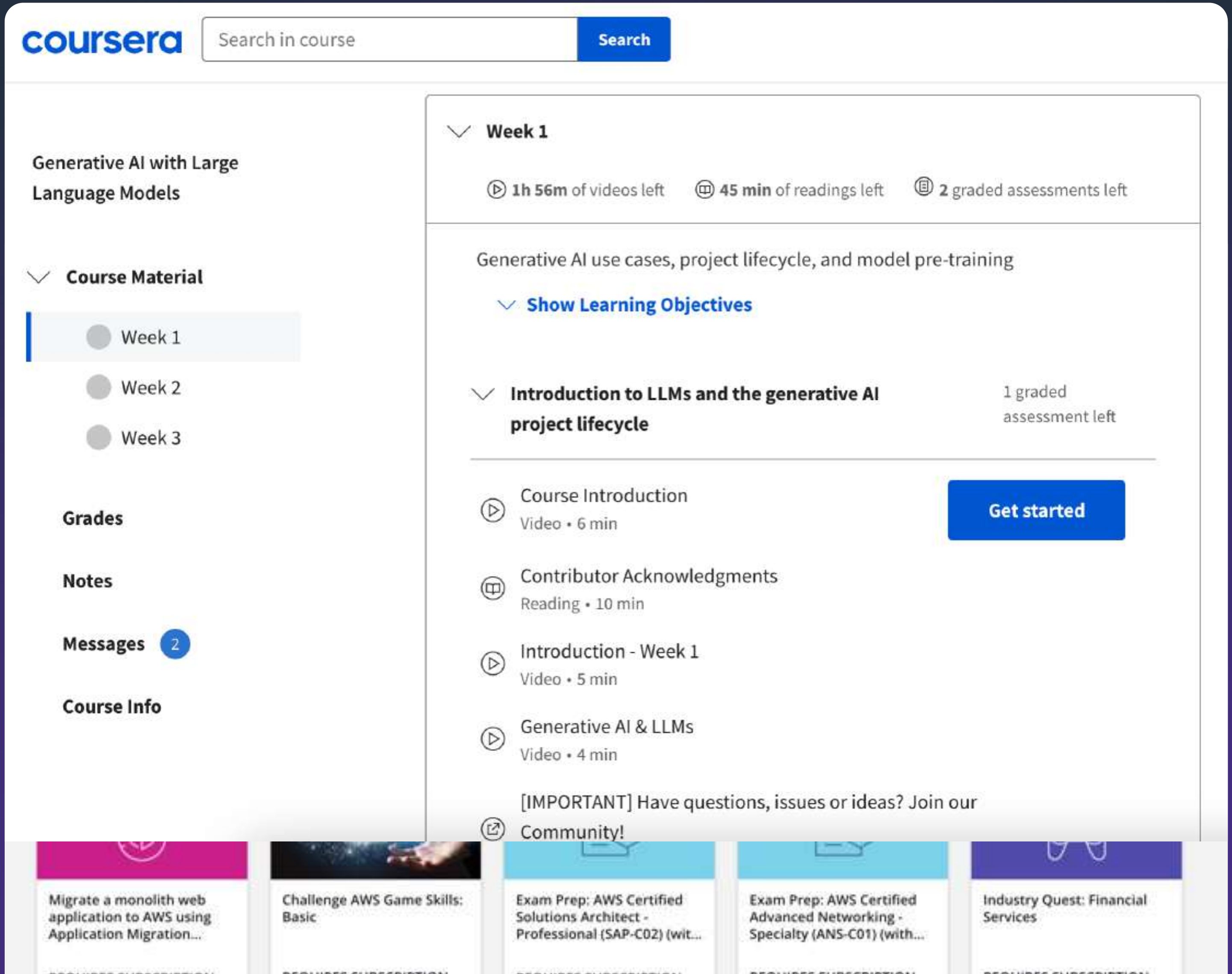
Get the most out of your AWS Skill Builder subscription with these training offerings available only with the subscription.

1/2

- Migrate a monolith web application to AWS using Application Migration...
- Challenge AWS Game Skills: Basic
- Exam Prep: AWS Certified Solutions Architect - Professional (SAP-C02) (wit...
- Exam Prep: AWS Certified Advanced Networking - Specialty (ANS-C01) (with...
- Industry Quest: Financial Services

Hands-on generative AI training opportunities

Learn the fundamentals of generative AI for real-world applications



Generative AI with Large Language Models

Week 1

1h 56m of videos left **45 min** of readings left **2** graded assessments left

Generative AI use cases, project lifecycle, and model pre-training

Introduction to LLMs and the generative AI project lifecycle 1 graded assessment left

Course Introduction
Video • 6 min

Contributor Acknowledgments
Reading • 10 min

Introduction - Week 1
Video • 5 min

Generative AI & LLMs
Video • 4 min

[IMPORTANT] Have questions, issues or ideas? Join our Community!

Migrate a monolith web application to AWS using Application Migration...

Challenge AWS Game Skills: Basic

Exam Prep: AWS Certified Solutions Architect - Professional (SAP-C02) (wit...

Exam Prep: AWS Certified Advanced Networking - Specialty (ANS-C01) (with...

Industry Quest: Financial Services

Generative AI for Business Professional (3+ hours)

This specialized plan offers a strategic overview of generative models, enabling you to grasp the pivotal role they play in driving innovation across industries. From understanding the technology's impact on market dynamics to identifying opportunities for differentiation, this learning experience empowers participants to make informed decisions that harness the power of generative AI for business's success.

AWS Partner: Generative AI on AWS Essentials (Business)
E-learning | Duration 1h 5m

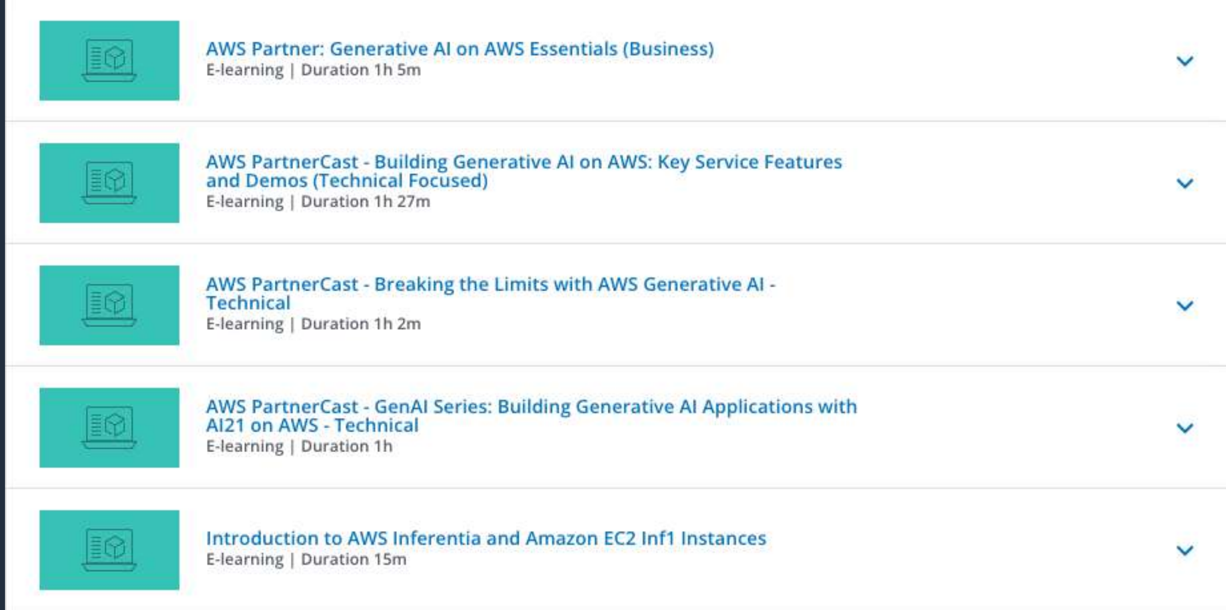
AWS PartnerCast - Generative AI Business Overview: Landscape and Market Opportunities (Business Focused) - Business
E-learning | Duration 1h 16m

AWS PartnerCast - Generative AI: The Future of Contact Center Automation and Modernization - Business
E-learning | Duration 44m

Link to enroll

Generative AI for Technical Professionals (9+ hours)

Dive into the realm of AWS Generative AI with our specialized learning plan crafted for technical professionals, including professional services, solutions architects, data analysts and engineers, and Technical Pre-Sales Architects. This comprehensive plan offers a deep exploration of generative models, equipping participants with the skills to architect innovative solutions that push the boundaries of what's possible. From honing your understanding of foundations models to orchestrating seamless integrations of generative AI into complex architectures, participants will gain practical insights and hands-on experience.



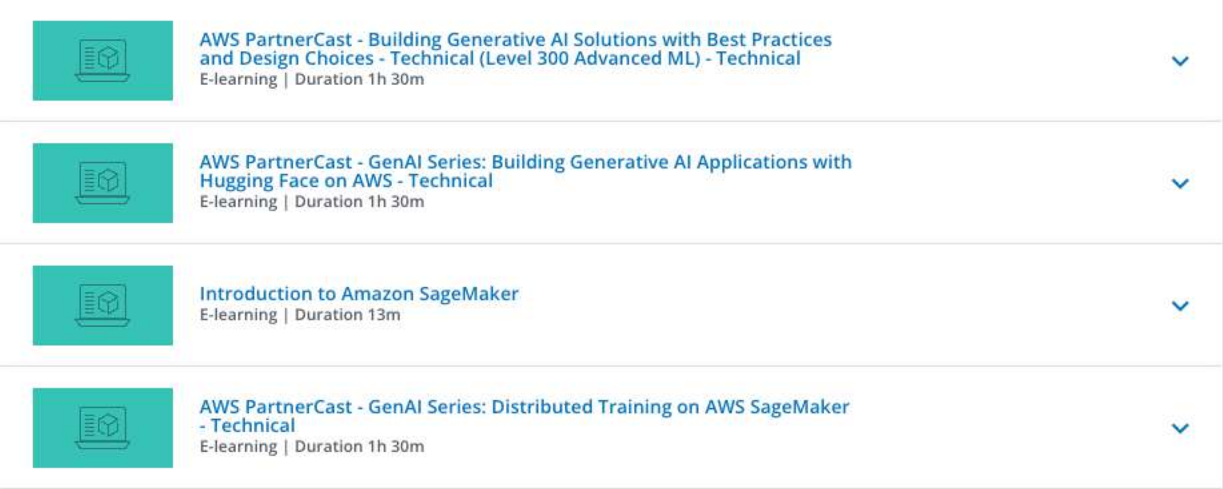
AWS Partner: Generative AI on AWS Essentials (Business)
E-learning | Duration 1h 5m

AWS PartnerCast - Building Generative AI on AWS: Key Service Features and Demos (Technical Focused)
E-learning | Duration 1h 27m

AWS PartnerCast - Breaking the Limits with AWS Generative AI - Technical
E-learning | Duration 1h 2m

AWS PartnerCast - GenAI Series: Building Generative AI Applications with AI21 on AWS - Technical
E-learning | Duration 1h

Introduction to AWS Inferentia and Amazon EC2 Inf1 Instances
E-learning | Duration 15m



AWS PartnerCast - Building Generative AI Solutions with Best Practices and Design Choices - Technical (Level 300 Advanced ML) - Technical
E-learning | Duration 1h 30m

AWS PartnerCast - GenAI Series: Building Generative AI Applications with Hugging Face on AWS - Technical
E-learning | Duration 1h 30m

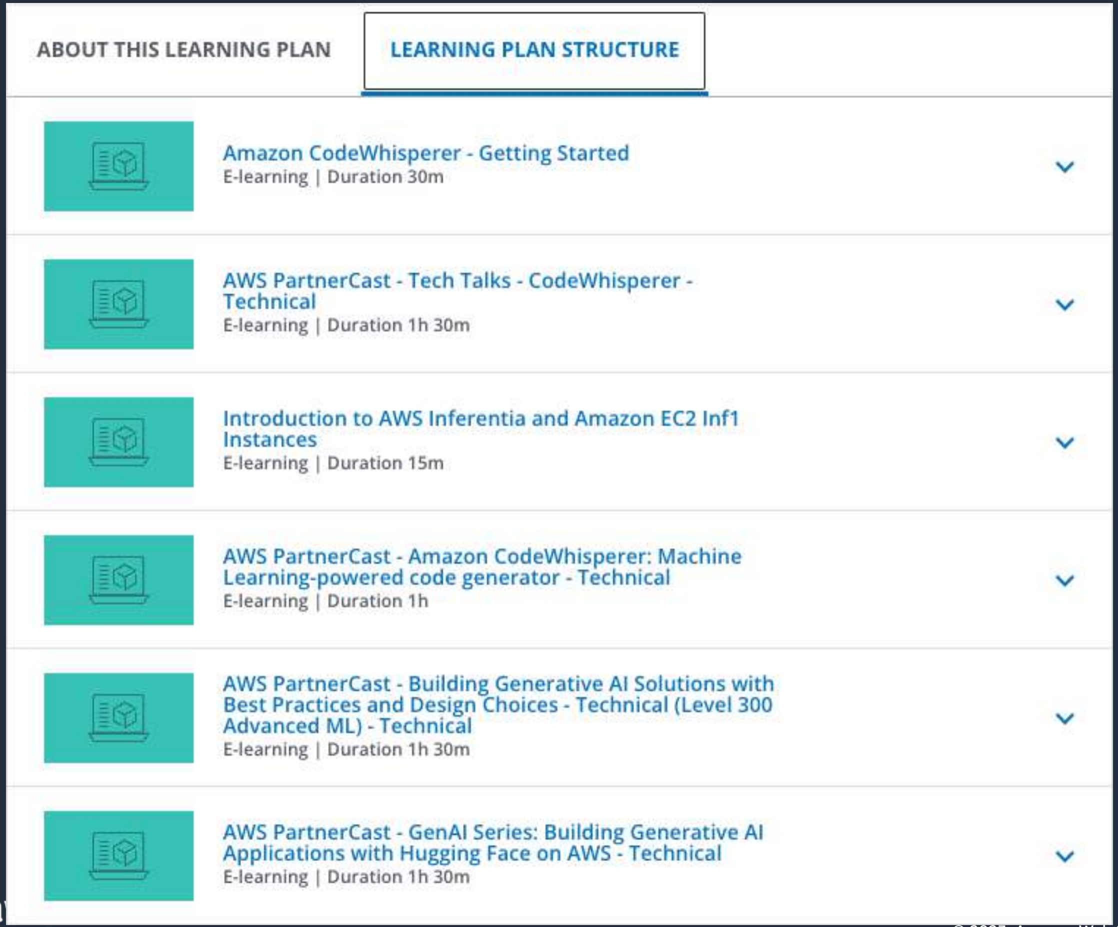
Introduction to Amazon SageMaker
E-learning | Duration 13m

AWS PartnerCast - GenAI Series: Distributed Training on AWS SageMaker - Technical
E-learning | Duration 1h 30m

Link to enroll

Generative AI for Developers (6+ hours)

Explore the cutting-edge realm of AWS Generative AI through our comprehensive learning plan tailored for developers. From understanding the fundamentals of generative models to mastering the intricacies of AWS tools such as Codewhisperer, this learning plan offers hands-on experiences, real-world use cases, and expert guidance.



Amazon CodeWhisperer - Getting Started
E-learning | Duration 30m

AWS PartnerCast - Tech Talks - CodeWhisperer - Technical
E-learning | Duration 1h 30m

Introduction to AWS Inferentia and Amazon EC2 Inf1 Instances
E-learning | Duration 15m

AWS PartnerCast - Amazon CodeWhisperer: Machine Learning-powered code generator - Technical
E-learning | Duration 1h

AWS PartnerCast - Building Generative AI Solutions with Best Practices and Design Choices - Technical (Level 300 Advanced ML) - Technical
E-learning | Duration 1h 30m

AWS PartnerCast - GenAI Series: Building Generative AI Applications with Hugging Face on AWS - Technical
E-learning | Duration 1h 30m

Link to enroll

Resources

- Amazon Bedrock Immersion Day
- Claude Advanced Prompt Examples
- Claude’s Prompt Design Guide
- Prompt Engineering Guide

Resources

- https://github.com/aws-samples/llm-apps-workshop
- https://github.com/aws-samples/aiml-genai-multimodal-agent
- https://github.com/aws-samples/amazon-sagemaker-genai-content-moderation
- https://github.com/aws-samples/aws-genai-llm-chatbot
- https://github.com/aws-samples/amazon-sagemaker-genai-datamesh
- https://github.com/aws-samples/amazon-bedrock-rag-workshop

Resources

- https://aws.amazon.com/blogs/machine-learning/build-a-secure-enterprise-application-with-generative-ai-and-rag-using-amazon-sagemaker-jumpstart/
- https://aws.amazon.com/blogs/machine-learning/a-generative-ai-powered-solution-on-amazon-sagemaker-to-help-amazon-eu-design-and-construction/
- https://github.com/aws-samples/bias-mitigation-foundation-models
- https://github.com/aws-samples/aws-ai-intelligent-document-processing
- https://github.com/aws-samples/rag-using-langchain-amazon-bedrock-and-opensearch
- https://aws.amazon.com/blogs/machine-learning/generative-ai-and-multi-modal-agents-in-aws-the-key-to-unlocking-new-value-in-financial-markets/

Resources

- Generative AI: How to Leverage AWS ML Services to Drive Value for your Customers - Business (L100)
- Generative AI on Amazon SageMaker: Key service features and demos - Technical (L200)
- Building Generative AI Solutions with Best Practices and Design Choices - Technical (Level 300 Advanced ML) (Password: AWSgai2023)
- Distributed Training and Hosting for LLM with Amazon Sagemaker (L300)
- Breaking the Limits with AWS Generative AI for Partners (L200/300)
- AWS Partners: Building Generative AI Applications with Hugging Face on AWS (L300) (Password: AWSgai2023)
- AWS Partners: Building Generative AI Applications with AI21 on AWS (Password: AWSgai2023)

Resources

- SageMaker JumpStart Examples
  - https://github.com/aws/amazon-sagemaker-examples/tree/main/introduction_to_amazon_algorithms
  - https://github.com/arunprsh/sagemaker-jumpstart-generative-ai-examples

- Fine Tuning and Domain adaptation
  - https://aws.amazon.com/blogs/machine-learning/domain-adaptation-fine-tuning-of-foundation-models-in-amazon-sagemaker-jumpstart-on-financial-data/

- RAG Examples
  - https://github.com/aws/amazon-sagemaker-examples/tree/main/introduction_to_amazon_algorithms/jumpstart-foundation-models/question_answering_retrieval_augmented_generation

Resources

- SageMaker + Kendra RAG Example
  - https://github.com/aws-samples/amazon-kendra-langchain-extensions
  - https://aws.amazon.com/blogs/machine-learning/quickly-build-high-accuracy-generative-ai-applications-on-enterprise-data-using-amazon-kendra-langchain-and-large-language-models/

- SageMaker + OpenSearch RAG Example
  - https://github.com/arunprsh/knowledge-augmented-LLMs
  - https://medium.com/@shankar.arunp/augmenting-large-language-models-with-verified-information-sources-leveraging-aws-sagemaker-and-f6be17fb10a8
  - https://catalog.workshops.aws/semantic-search/en-US/module-7-retrieval-augmented-generation
  - https://github.com/aws-samples/semantic-search-with-amazon-opensearch/blob/main/Module%207%20-%20Retrieval%20Augmented%20Generation.ipynb

Thank you!