Amazon Bedrock pricing

Amazon Bedrock is a comprehensive platform for building generative AI applications and agents with access to leading foundation models, services to deploy and operate agents, and tools for customizing, safeguarding, and optimizing applications.

Amazon Bedrock offers flexible pricing options for model inference to optimize cost while balancing speed, scale, and model access needs. This includes on-demand pricing for pay-as-you-go usage with no upfront commitments, and batch mode for cost-efficient processing of large volumes of input. AgentCore offers flexible, consumption-based pricing with no upfront commitments or minimum fee, which you can learn about by accessing the AgentCore pricing page.

Pricing overview

Service tiers and model import

Close all

Standard Tier

The Standard tier provides consistent performance at regular rates for everyday AI tasks. With the Standard tier you only pay for what you use, with no time-based term commitments. For text-generation models, you are charged for every input token processed and every output token generated. For embeddings models, you are charged for every input token processed. A token comprises a few characters and refers to the basic unit of text that a model learns to understand the user input and prompt. For image-generation models, you are charged for every image generated.

Learn more

Priority Tier

The Priority tier is a premium service tier that provides preferential compute allocation for mission-critical applications. This service tier is priced at a premium over the Standard tier. For most models that support Priority tier, customers can realize up to 25% better output tokens per second (OTPS) latency compared to Standard tier.

Learn more

Flex Tier

Flex tier offers discounted standard pricing for workloads that can trade immediate processing for cost efficiency. Perfect for non-urgent AI workloads.

Reserved Tier

Reserved service tier designed for workloads requiring predictable performance and guaranteed tokens-per-minute capacity. The Reserved tier provides the ability to reserve prioritized compute capacity, keeping service levels predictable for your mission critical applications. It also includes the flexibility to allocate different input and output tokens-per-minute capacities to match the exact requirements of your workload and control cost.

Learn more

Batch

With Batch mode, you can provide a set of prompts as a single input file and receive responses as a single output file, allowing you to get simultaneous large-scale predictions. The responses are processed and stored in your Amazon S3 bucket so you can access them at a later time. Amazon Bedrock offers select foundation models (FMs) from leading AI providers like Anthropic, Meta, Mistral AI, and Amazon for batch inference at a 50% lower price compared to on-demand inference pricing. Refer to the model list here.

Custom Model Import

Custom Model Import allows you to leverage your prior model customization investments within Amazon Bedrock and consume them in the same fully managed manner as Bedrock's existing hosted foundation models. You can import custom weights for supported model architectures and serve the custom model using On-Demand mode. There is no charge to import a custom model to Bedrock. Once you import a model, you will be able to access it on-demand without requiring to perform any control plane action. You are only charged for model inference, based on the number of copies of your custom model required to service your inference volume and the duration each model copy is active, billed in 5-minute windows. A model copy is a single instance of an imported model ready to serve inference requests. The price per model copy per minute depends on factors such as architecture, context length, AWS Region, compute unit version (hardware generation), and is tiered by model copy size.

Customization and optimization

Close all

Model Customization

Reinforcement Fine tuning
With Reinforcement fine-tuning capability in Amazon Bedrock, you can improve model accuracy without needing deep machine learning expertise or large sums of labeled data. Amazon Bedrock automates the reinforcement fine-tuning workflow: It takes your sample prompts, generates model responses, and scores them using your reward function. These prompts, responses, and scores are then used to train your custom model through an iterative RFT workflow. The entire workflow is billed at an hourly rate. You can set up inference on a custom model by creating a custom model on-demand deployment. The on-demand inference option includes a token-based pricing model that charges based on the number of tokens processed during inference.

Supervised Fine tuning
With Supervised fine tuning in Amazon Bedrock, you can customize FMs with your data to deliver tailored responses for specific tasks and your business context. You can fine-tune models with labeled data. For customization of a text-generation model, you are charged for the model training based on the total number of tokens processed by the model (number of tokens in the training data corpus x the number of epochs) and for model storage charged per month per model. An epoch refers to one full pass through your training dataset during fine-tuning or continued pretraining.

Model Distillation
With Amazon Bedrock Model Distillation, you can transfer knowledge from a larger capable model (known as "the teacher model") to a smaller, faster, and cost-efficient model (known as "the student model"), such that the student model can become as performant as the teacher for specific use-cases. With Amazon Bedrock Model Distillation you pay for what you use. Distillation is a two-step process where first synthetic data is generated from the teacher model and then the student model is trained. Synthetic data generation is charged at on-demand pricing of the selected teacher model. Fine-tuning of the student model is charged at model customization (supervised fine tuning) rates.

For Supervised fine tuning and Model Distillation, you can set up inference on a custom model by either creating a custom model on-demand deployment or purchasing Provisioned Throughput depending on your specific workload requirements and cost objectives. The on-demand inference option includes a token-based pricing model that charges based on the number of tokens processed during inference. If you purchase Provisioned Throughput, one model unit is made available with no commitment term for inference on a customized model. You will be charged for the number of hours you use in the first model unit for custom model inference. If you want to increase your throughput beyond one model unit, then you must purchase a 1-month or 6-month commitment term.

Prompt Caching

With prompt caching on Amazon Bedrock, you can cache repeated context across API calls to reduce your costs and response latencies. Prompts often contain common context or prefixes such as long, multi-turn conversations, many-shot examples and detailed instructions that refine model behavior. Using existing Amazon Bedrock APIs, you can specify the prompt prefixes that you want to cache for five minutes in an AWS account-specific cache. During that time, any requests with matching prefixes receive a discount of up to 90% on cached tokens and a latency improvement of up to 85%. Prices and performance improvements vary by model and prompt length, but your caches are always isolated to your AWS account.

Prompt Management and Optimization

Amazon Bedrock Prompt Management accelerates the creation, testing, and running of prompt through an intuitive UI and a set of APIs. You can easily test and version your prompts, compare different variants, and run them in a secure serverless infrastructure.

Prompt Optimization in Amazon Bedrock automatically rewrites prompts for better performance and more concise responses for foundation models. It integrates with Prompt Management for side-by-side comparison of original and optimized versions, and prompt life cycle management. You can also use Prompt Optimization in Bedrock Playground, or directly via API.

Tools

Close all

Guardrails

Amazon Bedrock Guardrails is the only responsible AI capability from a major cloud provider that helps you build and customize safety, privacy, and truthfulness safeguards for your generative AI applications. It evaluates user inputs and model responses based on use case-specific policies, providing an additional layer of safeguards beyond what's natively available. Safeguards from Guardrails can be applied to models hosted on Amazon Bedrock or with any third-party models (such as OpenAI and Google Gemini) via the ApplyGuardrail API. You can also use Guardrails with an agent framework such as Strands Agents, including agents deployed using Amazon Bedrock AgentCore. Guardrails helps filter hallucinations and improve factual accuracy through contextual grounding checks against RAG content and Automated Reasoning checks to deliver provably truthful responses.

Knowledge Bases

Amazon Bedrock Knowledge Bases is a fully managed Retrieval-Augmented Generation (RAG) workflow that enables customers to create highly accurate, low-latency, secure, and custom generative AI applications by incorporating contextual information from their own data sources. It supports various data sources, including S3, and Confluence, Salesforce, and SharePoint, in preview. It also offers document ingestion for streaming data. Bedrock Knowledge Bases converts unstructured data into embeddings, stores them in vector databases, and enables retrieval from diverse data stores. It also integrates with Kendra for managed retrieval and supports structured data retrieval using natural language to SQL.

Amazon Bedrock Data Automation

Amazon Bedrock Data Automation transforms unstructured, multimodal content into structured data formats for use cases like intelligent document processing, video analysis, and RAG. Bedrock Data Automation can generate Standard Output content using predefined defaults which are modality specific, like scene-by-scene descriptions of videos, audio transcripts or automated document analysis. Customers can additionally create Custom Outputs by specifying their output requirements in Blueprints based on their own data schema that they can then easily load into an existing database or data warehouse. Through an integration with Knowledge Bases, Bedrock Data Automation can also be used to parse content for RAG applications, improving the accuracy and relevancy of results by including information embedded in both images and text.

Flows

Amazon Bedrock Flows is a workflow authoring and execution feature of Bedrock for generative AI applications. It accelerates the creation, testing, and deployment of user-defined generative AI workflows through an intuitive visual builder and a set of APIs. It allows you to seamlessly link the latest foundation models, Prompts, Agents, Knowledge Base, Guardrails, and AWS services (such as Amazon Lex, AWS Lambda, Amazon S3) along with business logic to build generative AI workflows. You can easily test and version your workflows, and run it in a secure serverless environment through a visual interface or API without having to stand up your own infrastructure.

Evaluations

Model Evaluation: With Amazon Bedrock model evaluation, you pay for what you use, with no minimum volume commitments on the number of prompts or responses. For automatic (programmatic) evaluation, you only pay for the inference from your choice of model in the evaluation. The automatically-generated algorithmic scores are provided at no extra charge. For automatic (Model/LLM-as-a-judge) evaluation, you only pay for the inference from your choice of generator model and evaluator model. In an LLM-as-a-judge model evaluation job, the built-in metrics use system judge prompt templates unique to each metric and available judge model that will be charged as part of your token usage, and the judge prompts are available in the public AWS documentation for transparency. For human-based evaluation where you bring your own workteam, you are charged for the model inference in the evaluation, and a charge of $0.21 per completed human task. A human task is defined as an occurrence of a human worker submitting an evaluation of a single prompt and its associated inference responses in the human evaluation user interface. The price per task is the same whether you have one or two models in your evaluation job and also the same regardless of how many evaluation metrics and rating methods you include. The charges for the human tasks will appear under the Amazon SageMaker section in your AWS bill and are the same for all AWS Regions. There is no separate charge for the workforce, as the workforce is supplied by you. If you are using the "bring your own inference responses" feature instead of calling a Bedrock model during the evaluation job, you are only charged for the evaluator model inference (LLM-as-a-judge jobs) or the $0.21 per completed human task (human-based evaluation jobs). For an evaluation managed by AWS, pricing is customized for your evaluation needs in a private engagement while working with the AWS expert evaluations team.

RAG Evaluation: With Amazon Bedrock RAG evaluation, you pay for what you use, with no minimum volume commitments on the number of prompts or responses. If you are evaluating an Amazon Bedrock Knowledge Base, you only pay for the inference from your choice of generator model and evaluator model (the evaluation job uses an LLM-as-a-judge), as well as any charges incurred from using the Knowledge Base in the evaluation job according to Amazon Bedrock Knowledge Bases pricing. If you are using the "bring your own inference responses" feature, you are only charged for the evaluator model inference. In a RAG evaluation job, the built-in metrics use system judge prompt templates unique to each metric and available judge model that will be charged as part of your token usage, and the judge prompts are available in the public AWS documentation for transparency. Some metrics involve doing judge model inference on retrieved context from your Knowledge Base/RAG system or your ground truth answers in addition to the input prompt, which affects the costs associated with each metric - more information on each metric can be found in the public AWS documentation for evaluations.

Service tiers and model import 6
Customization and optimization 3
Tools 5

Model pricing details

Pricing is dependent on the modality, provider, and model. Please select the model provider to see detailed pricing.

Amazon Bedrock offers select foundation models (FMs) from leading AI providers like Anthropic, Meta, Mistral AI, and Amazon for batch inference at a 50% lower price compared to on-demand inference pricing. Please refer to model list here.

AI21 Labs | Amazon | Anthropic | Cohere | DeepSeek | Google | Kimi AI | Luma AI | Meta | MiniMax AI | Mistral AI | NVIDIA | OpenAI | TwelveLabs | Stability AI | Writer | Qwen | Custom Model Import

OpenAI

Standard Tier and Batch pricing

Region:
Europe (London) ▾

OpenAI models	Price per 1,000 input tokens	Price per 1,000 output tokens	Price per 1,000 input tokens (batch)	Price per 1,000 output tokens (batch)
gpt-oss-20b	$0.00011	$0.00047	$0.00005	$0.00023
gpt-oss-120b	$0.00023	$0.00093	$0.00012	$0.00047

Priority tier pricing

OpenAI models	Price per 1,000 input tokens	Price per 1,000 output tokens
gpt-oss-20b	$0.0001925	$0.0008225
gpt-oss-120b	$0.0004025	$0.0016275

Flex tier pricing

OpenAI models	Price per 1,000 input tokens	Price per 1,000 output tokens
gpt-oss-20b	$0.000055	$0.000235
gpt-oss-120b	$0.000115	$0.000465

On-Demand pricing

Regions: US East (N. Virginia), US East (Ohio) and US West (Oregon)

OpenAI models	Price per 1,000 input tokens	Price per 1,000 output tokens
GPT OSS Safeguard 20B	$ 0.00007	$ 0.00020
GPT OSS Safeguard 120B	$ 0.00015	$ 0.00060

Regions: Asia Pacific (Mumbai), South America (Sao Paulo) and Asia Pacific (Tokyo)

OpenAI models	Price per 1,000 input tokens	Price per 1,000 output tokens
GPT OSS Safeguard 20B	$ 0.00008	$ 0.00024
GPT OSS Safeguard 120B	$ 0.00018	$ 0.00071

Regions: Europe (Ireland) and Europe (Milan)

OpenAI models	Price per 1,000 input tokens	Price per 1,000 output tokens
GPT OSS Safeguard 20B	$ 0.00008	$ 0.00023
GPT OSS Safeguard 120B	$ 0.00018	$ 0.00070

Region: Europe (London)

OpenAI models	Price per 1,000 input tokens	Price per 1,000 output tokens
GPT OSS Safeguard 20B	$ 0.00011	$ 0.00031
GPT OSS Safeguard 120B	$ 0.00023	$ 0.00093

*** Priority tier pricing is at 75% premium to Standard tier pricing**
*** Flex tier pricing is at 50% discount to Standard tier pricing**

Tools and optimization pricing details

Flows | Knowledge Bases | Guardrails | Model Evaluation | Data Automation | Intelligent Prompt Routing | Prompt Optimization

Amazon Bedrock Flows

You are charged based on the number of node transitions required to execute your application. Bedrock Flows counts a node transition each time a node in your workflow is executed. You are charged for the total number of node transitions across all your flows.

All charges are metered daily and billed monthly starting February 1st, 2025.

Price per 1,000 node transitions
$0.035

Additional Charges

You may incur additional charges if the execution of your application workflow utilizes other AWS services or transfers data. For example, if your workflow invokes an Amazon Bedrock Guardrail policy, you will be billed for the number of text units processed by the policy.

Pricing examples

AI21 labs

An application developer makes the following API calls to Amazon Bedrock: a request to AI21's Jurassic-2 Mid model to summarize an input of 10K tokens of input text to an output of 2K tokens.

Total cost incurred = 10K tokens/1000 * $0.0125 + 2K tokens/1000 * $0.0125 = $0.15

Amazon

On-Demand pricing
An application developer makes the following API calls to Amazon Bedrock on an hourly basis: a request to Amazon Titan Text Lite model to summarize an input of 2K tokens of input text to an output of 1K tokens.

Total hourly cost incurred is = 2K tokens/1000 * $0.0003 + 1K tokens/1000 * $0.0004 = $0.001.

An application developer makes the following API calls to Amazon Bedrock: a request to the Amazon Titan Image Generator base model to generate 1000 images of 1024 x 1024 in size of standard quality.

Total cost incurred = 1000 images * $0.01 per image = $10

Customization (fine-tuning and continued pretraining) pricing
An application developer customizes an Amazon Titan Image Generator model using 1000 image-text pairs. After training, the developer uses custom model provisioned throughput for 1 hour to evaluate the performance of the model. The fine-tuned model is stored for 1 month. After evaluation, the developer uses provisioned throughput (1-month commitment term) to host the customized model.

Monthly cost incurred for fine-tuning = fine-tuning training ($.005 * 500 * 64), where $0.005 is the price per image seen, 500 is the number of steps, and 64 is the batch size, + custom model storage per month ($1.95) + 1 hour of custom model inference ($21) = $160 + $1.95 + 21 = $182.95

Provisioned Throughput pricing
An application developer buys two model units of Amazon Titan Text Express with a 1-month commitment for their text summarization use case.

Total monthly cost incurred = 2 model units * $18.40/hour * 24 hours * 31 days = $27,379.20

An application developer buys one model unit of the base Amazon Titan Image Generator model with a 1-month commitment.

Total cost incurred = 1 model unit * $16.20 * 24 hours * 31 days = $12,052.80

Amazon Bedrock Guardrails

Example 1: Customer support chatbot
An application developer creates a customer support chatbot and uses content filters to block harmful content and denied topics to filter undesirable queries and responses.

The chatbot serves 1000 user queries per hour. Each user query has an average input length of 200 characters and receives a FM response of 1500 characters.

Each user query of 200 characters correspond to 1 text unit.

Each FM response of 1,500 characters correspond to 2 text units.

Text units processed each hour = (1 + 2) * 1000 queries = 3000 text units

Total cost incurred per hour for content filters and denied topics = 3000 * ($0.15 + $0.15) / 1000 = $0.90

Example 2: Call center transcript summarization
An application developer creates an application to summarize chat transcripts between users and support agents. It uses sensitive information filter to redact personally identifiable information (PII) in the generated summaries for 10,000 conversations.

Each generated summary has an average of 3,500 characters that corresponds to 4 text units.

Total cost incurred to summarize 10,000 conversations = 10000 * 4 * ($0.1/1000) = $4

Example 3: Medical Protocol Verification Engine
A healthcare technology company implements Automated Reasoning checks in their clinical decision support system to validate treatment suggestions against medical guidelines.

The system processes 5,000 patient cases per month. Each case involves:

- Patient data summary: 500 characters (1 text unit)
- Diagnostic assessment: 2,000 characters (2 text units)
- Treatment recommendation: 4,500 characters (5 text units)

Text units processed per month = (1 + 2 + 5) 5,000 cases = 40,000 text units
Total cost incurred per month for Automated Reasoning checks = 40,000 ($0.17) / 1000 = $6.80

Amazon Bedrock Knowledge Bases

Pricing Example 1 (Reranking using Amazon Rerank 1.0 model)

In a given month, you make 2 million requests to Rerank API using Amazon Rerank 1.0 model – 1 million requests contain fewer than 100 documents each and hence will be charged for one request each. The remaining 1 million requests contain 120-150 documents, and hence each request will be charged for 2 requests.

Price for one request = $0.001
Total charge = 1,000,000 * $0.001 + 1,000,000*2*$0.001= $3000

Pricing Example 2: (Structured data retrieval)

An application developer creates a support chatbot that queries structured data stored in Amazon Redshift. The developer creates a Bedrock Knowledge Base and connects to Amazon Redshift. The chatbot serves 10000 user queries per hour. Each user query will cost $0.002 per GenerateQuery API to generate SQL from user query.

Total cost incurred for generating SQL per hour = $0.002*10000 = $20.
Total cost incurred in month = $20*24*30 = $1440

Anthropic

On-Demand pricing
An application developer makes the following API calls to Amazon Bedrock in the US West (Oregon) Region: a request to Anthropic's Claude model to summarize an input of 11K tokens of input text to an output of 4K tokens.

Total cost incurred = 11K tokens/1000 * $0.008 + 4K tokens/1000 * $0.024 = $0.088 + $0.096 = $0.184

Provisioned Throughput pricing
An application developer buys one model unit of Anthropic Claude Instant in the US West (Oregon) Region:

Total monthly cost incurred = 1 model unit * $39.60 * 24 hours * 31 days = $29,462.40

Cohere

On-Demand pricing
An application developer makes the following API calls to Amazon Bedrock: a request to Cohere's Command model to summarize an input of 6K tokens of input text to an output of 2K tokens.

Total cost incurred = 6K tokens/1,000 * $0.0015 + 2K tokens/1,000 * $0.0020 = $0.013

An application developer makes the following API calls to Amazon Bedrock: A request to Cohere's Command - Light model to summarize an input of 6K tokens of input text to an output of 2K tokens.

Total cost incurred = 6K tokens/1000 * $0.0003 + 2K tokens/1000 * $0.0006 = $0.003

An application developer makes the following API calls to Amazon Bedrock: A request to either Cohere's Embed English or Embed Multilingual model to generate embeddings for 10K tokens of input.

Total cost incurred = 10K tokens/1000 * $0.0001 = $.001

Customization (fine-tuning) pricing

An application developer customizes a Cohere Command model using 1000 tokens of data. After training, uses custom model provisioned throughput for 1 hour to evaluate the performance of the model. The fine-tuned model is stored for 1 month. After evaluation, the developer uses provisioned throughput (1mo commit) to host the customized model.

Monthly cost incurred for fine-tuning = Fine-tuning training ($0.004 * 1000) + custom model storage per month ($1.95) + 1 hour of custom model inference ($49.50) = $55.45

Monthly cost incurred for provisioned throughput (1-month commitment) of custom model = $39.60

Provisioned Throughput pricing

An application developer, buys one model unit of Cohere Command with a 1-month commitment for their text summarization use case.

Total monthly cost incurred = 1 model unit * $39.60 * 24 hours * 31 days = $29,462.40

Custom Model Import

Pricing Example: An application developer imports a customized Llama 3.1 type model that is 8B parameter in size with a 128K sequence length in us-east-1 region and deletes the model after 1 month. This requires 2 Custom Model Units. So, the price per minute will be $0.1570 because 2 Custom Model Units are required. The model storage costs for 2 Custom Model Units would be $3.90 for the month.

There is no charge to import the model. The first successful invocation is at 8:03 AM, at which time the metering starts. The 5-minute metering windows are from 8:03 AM - 8:07 AM; 8:07 AM - 8:11 AM, and so on. If there is at least one invocation during any 5-minute period, the window will be considered active for billing. If there is an invocation at 8:03 AM and no further invocations after 8:07 AM, the metering will stop at 8:07 AM. In this case, the bill would be calculated as follows: $0.1570 * 5 minutes * 1 five minute windows = **$0.785**.

Data Automation

Pricing example 1:
Let's say you process a 1,000 page document using BDA Custom Output. All 1,000 pages are processed using blueprint 1 which has 15 fields. The per page price for any blueprint with 30 fields or less is $0.040. The total cost would be $40.

Total pages processed = 1,000
Price per page for blueprints with less than 30 fields = $0.040

Let's say you process 2 documents using BDA Custom Output. Document 1

fields. The per page price of blueprint 1 is $0.040 since it contains 30 fields or less. The per page price of blueprint 2 is $0.045. The processing cost for Document 1 using blueprint 1 is $1.60. The processing cost for Document 2 using blueprint 2 is $0.45. The total cost of processing both documents would be $2.05.

Total pages processed = 50
Price per page for Blueprint 1 with less than 30 fields = $0.040
Price per page for Blueprint 2 with 40 fields = $0.040 + (# of additional fields above 30 *$0.0005 per field)
Number of additional fields above 30 = 40 - 30 = 10
Price per page for Blueprint 2 with 40 fields = $0.040 + (10 *$0.0005 per field) = $0.045
Charge for Document 1 using Blueprint 1 = 40 pages x $0.040 per page = $1.6
Charge for Document 2 using Blueprint 2 = 10 pages x $0.045 per page = $0.45
Total charge = Charge for Document 1 + Charge for Document 2 = $1.6 + $0.45 = $2.05

Pricing Example 3:
Let's say you setup Bedrock Knowledge Bases to use Bedrock Data Automation as a parser and then ingest a 1,000 page document. Note, cost structures differ between the Knowledge Bases parsing options. BDA uses per-page pricing, while Foundational Model parsers charge based on input and output tokens. For context, processing 1,000 pages, where 30% contain tables and 30% contain figures, typically requires 2,900 input tokens and 750 output tokens. Token consumption varies by content type, so customers are encouraged to test using their own data to get more accurate estimates. Bedrock Knowledge Bases and Bedrock Data Automation integration uses standard output, where the per page price is $0.010. The total cost would be $10.

Total pages processed = 1,000
Price per page for standard output = $0.010
Total charge = 1,000 * $0.010 = $10

Pricing example 4:
Let's say you process a 60 minute video using BDA Standard Output. The per minute price for video standard output is $0.050. The total cost would be $3.00.

Total minutes processed = 60
Price per minute for video standard output = $0.050
Total charge = 60 * $0.050 = $3.00

Pricing example 5:
Let's say you process 2,000 images using BDA Custom Output. The first 1,000 images are processed using blueprint 1, which has 10 fields. The last 1,000 pages are processed using blueprint 2, which has 40 fields. The per image price for blueprint 1 is $0.005, since it contains 30 fields or less. The per image price of blueprint 2 is $0.01. The processing cost for the first 1,000 images using blueprint 1 is $5.00. The processing cost for the second 1,000 images using blueprint 2 is $10.00. The total cost of processing all 2,000 images would be $15.00

Cost for first 1000 images = 1,000 images * $0.005 per image = $5.00
Cost for second 1,000 images = 1,000 images * ($0.005 + (# of additional fields above 30 *$0.0005 per field))
= 1,000 * ($0.005 + ((40-30)*$0.0005))
= 1,000 * ($0.005 + (10*$0.0005)) = $10.00
Total cost = $5.00 + $10.00 = $15.00

Pricing example 6:
Let's assume that you want to use Bedrock Data Automation Standard Output to process 15,000 minutes of meeting audio recordings in your organization. The total cost of processing all 15,000 audio minutes would be $90.

Total minutes processed = 15,000 minutes
Total charge = 15,000 min * $0.006 = $90

DeepSeek

On-Demand pricing

An application developer makes the following API calls to Amazon Bedrock on an hourly basis: a request to the DeepSeek-R1 model to summarize an input of 2K tokens of input text to an output of 1K tokens (including reasoning tokens):

Total hourly cost incurred = 2K tokens/1000 * $0.00135 + 1K tokens/1000 * $0.0054 = $0.0081

Flows

Example: News summarization
An application developer creates a flow to automate news summarization for traders. The flow includes an Input node that takes in S3 location, and a S3 retrieval node that retrieves 10 files that include articles from 10 major news agency in S3 (2 node transitions). It then uses an iterator node to invoke a model with a prompt node to summarize each file (+ 10 files x 2 node transitions). It then collects all the results using a collector node, write the results to S3 using S3 storage node, and complete in an Output node (+ 3 node transition). They run this flow every half hour of every week day.

The number of node transition per flow execution is: 2+1+10*2 + 3 = 25 node transitions/flow execution

The number of flow execution per month is: 24 hours *2* 5 days * 4 weeks = 960 flow executions/month.

Total monthly bill is: 25 * 960 * $0.035/1000 = $0.84

Additional charges
The bill will also include additional charges for AWS services used in the workflow execution, including Amazon S3 usages in the retrieval and storage nodes, and Amazon Bedrock foundation model usage in the prompt node.

Meta

On-Demand pricing
An application developer makes the following API calls to Amazon Bedrock: a request to Meta's Llama 2 Chat (13B) model to summarize an input of 2K tokens of input text to an output of 500 tokens.

Total cost incurred = 2K tokens/1000 * $0.00075 + 500 tokens/1000 * $0.001 = $0.002

Customization (fine-tuning) pricing
An application developer customizes the Llama 2 Pretrained (70B) model using 1000 tokens of data. After training, uses custom model provisioned throughput for 1 hour to evaluate the performance of the model. The fine-tuned model is stored for 1 month. After evaluation, the developer uses provisioned throughput (1mo commit) to host the customized model.

Monthly cost incurred for fine-tuning = Fine tuning training ($0.00799 * 1000) + custom model storage per month ($1.95) + 1 hour of custom model inference ($23.50) = $33.44

Monthly cost incurred for provisioned throughput (a 1-month commit) of custom model = $21.18

Provisioned Throughput pricing
An application developer buys one model unit of Meta Llama 2 with a 1-month commitment for their text summarization use case.

Total monthly cost incurred = 1 model unit * $21.18 * 24 hours * 31 days = $15,757.92

Mistral AI

On-Demand pricing
An application developer makes the following API calls to Amazon Bedrock on an hourly basis: a request to Mistral 7B model to summarize an input of 2K tokens of input text to an output of 1K tokens.

Total hourly cost incurred = 2K tokens/1000 * $0.00015 + 1K tokens/1000 * $0.0002 = $0.0005

An application developer makes the following API calls to Amazon Bedrock on an hourly basis: a request to Mixtral 8x7B model to summarize an input of 2K tokens of input text to an output of 1K tokens.

Total hourly cost incurred = 2K tokens/1000 * $0.00045 + 1K tokens/1000 * $0.0007 = $0.0016

An application developer makes the following API calls to Amazon Bedrock on an hourly basis: a request to Mistral Large model to summarize an input of 2K tokens of input text to an output of 1K tokens.

Total hourly cost incurred = 2K tokens/1000 * $0.008 + 1K tokens/1000 * $0.024 = $0.04

Model evaluation

Model evaluation example 1:

On-demand pricing
An application developer submits a dataset for human-based model evaluation using Anthropic Claude 2.1 and Anthropic Claude Instant in the US East (N. Virginia) AWS Region.

The dataset contains 50 prompts, and the developer requires one worker to rate each prompt-response set (configurable in the evaluation job creation as "workers per prompt" parameter).

There will be 50 tasks in this evaluation job (one task for each prompt-response set per each worker). The 50 prompts combine to 5000 input tokens, and the associated responses combine to 15,000 tokens for Anthropic Claude Instant and 20,000 tokens for Anthropic Claude 2.1.

The following charges are incurred for this model evaluation job:

Item	Number of input tokens	Price per 1000 input tokens	Cost of input	Number of output tokens	Price per 1000 output tokens	Cost of output	Number of human tasks
Claude Instant Inference	5000	$0.0008	$0.004	15000	$0.0024	$0.036	
Claude 2.1 Inference	5000	$0.008	$0.04	20000	$0.024	$0.48	
Human Tasks							50
Total							

Model evaluation example 2:

On-demand pricing
An application developer submits a dataset for human-based model evaluation using Anthropic Claude 2.1 and Anthropic Claude Instant in the US East (N. Virginia) AWS Region.

The dataset contains 50 prompts, and the developer requires two workers to rate each prompt-response set (configurable in the evaluation job creation as "workers per prompt" parameter). There will be 100 tasks in this evaluation job (1 task for each prompt-response set per each worker: 2 workers x 50 prompt-response sets = 100 human tasks).

The 50 prompts combine to 5000 input tokens, and the associated responses combine to 15000 tokens for Anthropic Claude Instant and 20000 tokens for Anthropic Claude 2.1.

The following charges are incurred for this model evaluation job:

Item	Number of input tokens	Price per 1000 input tokens	Cost of input	Number of output tokens	Price per 1000 output tokens	Cost of output	Number of human tasks
Claude Instant Inference	5000	$0.0008	$0.0040	15000	$0.0024	$0.036	
Claude 2.1 Inference	5000	$0.008	$0.0400	20000	$0.024	$0.48	
Human Tasks							100
Total							

Prompt Optimization

Example: News summarization
An application developer creates a prompt to summarize news for traders using Claude 3.5. The original prompt includes 429 tokens. The optimized prompt has 511 tokens, and includes more specific instructions and examples to generate more concise answer from the FMs. He uses the optimized prompt with 511 tokens as the input for prompt optimizer, and creates 2 new variants for Claude 3.7 and Nova Pro with 582 and 579 tokens.

The total number of input and output tokens for prompt optimization: 429 + 511 + 511 + 582 + 511 + 579 = 3,123

Total monthly bill is: 3,123 / 1000 * $0.03 = $0.09

Stability AI

On-Demand pricing

An application developer makes the following API calls to Amazon Bedrock: a request to the SDXL model to generate a 512 x 512 image with a step size of 70 (premium quality).

Total cost incurred = 1 image * $0.036 per image = $0.036

An application developer makes the following API calls to Amazon Bedrock: A request to the SDXL 1.0 model to generate a 1024 x 1024 image with a step size of 70 (premium quality).

Total cost incurred = 1 image * $0.08 per image = $0.08

Provisioned Throughput pricing

An application developer buys one model unit of SDXL 1.0 with a 1-month commitment.

Total cost incurred = 1 * $49.86 * 24 hours * 31 days = $37,095.84

TwelveLabs

On-Demand pricing

An application developer makes the following API calls to Amazon Bedrock: a request to the **Pegasus 1.2** model to describe what a 10-second-long video entails, which provides an output of 2,000 tokens.

Total cost incurred = 10 seconds * $0.00049 + 2K tokens / 1000 * $0.0075 = $0.0199

An application developer makes the following API calls to Amazon Bedrock: a request to the **Marengo Embed [3.0 or 2.7]** model to embed 10 videos, with a combined duration of 100 minutes.

Total cost incurred = 100 minutes (i.e. 6000 secs) * $0.00070 = $4.2

An application developer makes the following API calls to Amazon Bedrock: a request to the **Marengo Embed 3.0** model to by providing a text and image together, to generate an embedding that they could use to find the clip that has the bag shown in the given image, across the embedding repository that they would have crated using above example.

Total cost incurred = 1 text request * $0.00007 +1 image request * $0.0001 = $0.00017

An application developer makes the following API calls to Amazon Bedrock: a request to the **Marengo Embed [3.0 or 2.7]** model to by providing a text to generate an embedding that they could use to find matching clips from an the embedding repository that they would have crated using above example.

Total cost incurred = 1 text request * $0.00007 = $0.00007

Writer

An application developer makes the following API calls to Amazon Bedrock: a request to Writer's Palmyra X5 model to summarize an input of 10K tokens of input text to an output of 2K tokens.

Total cost incurred = 10K tokens/1000 * $0.003 + 2K tokens/1000 * $0.015 = $0.06

Next steps

Workshop

Explore common Amazon Bedrock use cases with a guided workshop →

Demo

View demos of Amazon Bedrock capabilities →

Learn
What Is AWS?
What Is Cloud Computing?
What Is Agentic AI?
Cloud Computing Concepts Hub
AWS Cloud Security
What's New
Blogs
Press Releases

Resources
Getting Started
Training
AWS Trust Center
AWS Solutions Library
Architecture Center
Product and Technical FAQs
Analyst Reports
AWS Partners

Developers
Builder Center
SDKs & Tools
.NET on AWS
Python on AWS
Java on AWS
PHP on AWS
JavaScript on AWS

Help
Contact Us
File a Support Ticket
AWS re:Post
Knowledge Center
AWS Support Overview
Get Expert Help
AWS Accessibility
Legal

Back to top ↑

Amazon is an Equal Opportunity Employer: Minority / Women / Disability / Veteran / Gender Identity / Sexual Orientation / Age.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.