Implementation Guide

Generative AI Application Builder on AWS

Generative AI Application Builder on AWS: Implementation Guide

Table of Contents

This solution facilitates the development, rapid experimentation, and deployment of generative artificial intelligence (AI) applications

Publication date: *October 2023 (last update: April 2024)*

Generative AI Application Builder on AWS facilitates the development, rapid experimentation, and deployment of generative artificial intelligence (AI) applications without requiring deep experience in AI. This AWS Solution accelerates development and streamlines experimentation by helping you ingest your business-specific data and documents, evaluate and compare the performance of large language models (LLMs), rapidly build extensible applications, and deploy those applications with an enterprise-grade architecture.

Generative AI Application Builder on AWS includes integrations with Amazon Bedrock and its included LLMs, such as Amazon Titan, and LLMs deployed on Amazon SageMaker. Additionally, this solution has pre-built connectors to external model providers such as Anthropic and Hugging Face and enables connections to your choice of model using LangChain or AWS Lambda. Start with the no-code deployment wizard to build generative AI applications for conversational search, AI-generated chatbots, text generation, and text summarization.

This implementation guide provides an overview of the Generative AI Application Builder on AWS solution, its reference architecture and components, considerations for planning the deployment, and configuration steps for deploying the solution to the Amazon Web Services (AWS) Cloud.

This guide is intended for solution architects, business decision makers, DevOps engineers, data scientists, and cloud professionals who want to implement Generative AI Application Builder on AWS in their environment.

Use this navigation table to quickly find answers to these questions:

If you want to . . .	Read . . .
Know the cost for running this solution. The estimated cost for running this solution for a simple proof of concept in the US East (N. Virginia) Region is USD $35.64 per month.	Cost

If you want to . . .	Read . . .
Understand the security considerations for this solution.	Security
Know how to plan for quotas for this solution.	Quotas
Know which AWS Regions support this solution.	Supported AWS Regions
View or download the AWS CloudForm ation template included in this solution to automatically deploy the infrastructure resources (the “stack”) for this solution.	AWS CloudFormation template
Access the source code and optionally use the AWS Cloud Development Kit (AWS CDK) to deploy the solution.	GitHub repository

Features and benefits

The Generative AI Application Builder on AWS solution provides the following features:

Rapid experimentation

This solution allows users to experiment quickly by removing the heavy lifting required to deploy multiple instances with different configurations and compare outputs and performance. Experiment with multiple configurations of various LLMs, prompt engineering, enterprise knowledge bases, and other parameters.

Choice and configurability

With pre-built connectors to a variety of LLMs, such as models available through Amazon Bedrock and external providers Anthropic and Hugging Face, this solution gives you the flexibility to deploy the model of your choice, as well as the AWS and leading FM services you prefer.

Production-ready

Built with AWS Well-Architected design principles, this solution offers enterprise-grade security and scalability with high availability and low latency, ensuring seamless integration into your applications with high performance standards.

Extensible modular architecture

Extend this solution's functionality by integrating your existing projects or natively connecting additional AWS services. Because this is an open-source application, you can use the included LangChain orchestration layer or Lambda functions to connect with the services of your choice.

Integration with Service Catalog AppRegistry and Application Manager, a capability of AWS Systems Manager

This solution includes a Service Catalog AppRegistry resource to register the solution's CloudFormation template and its underlying resources as an application in both AWS Service Catalog AppRegistry and AWS Systems Manager Application Manager. With this integration, you can centrally manage the solution's resources.

Use cases

Question answering over enterprise data

LLMs and other foundation models have been pre-trained on a large corpus of data enabling them to perform well at many natural language processing (NLP) tasks. But most foundation models and LLMs are static and have been pre-trained, limiting their ability to accurately answer questions on topics which are either new, specialized, or proprietary. Using prompt-based learning, you can leverage the powerful NLP and text generation features of an LLM to provide richer customer experiences over your enterprise data.

Rapid generative AI prototyping

Out of the box, the solution comes bundled with various model providers and use cases. With an easy to use deployment wizard, customers can deploy pre-built use cases to enable the rapid experimentation of different generative AI prototypes and workloads.

Multi LLM comparison and experimentation

LLMs perform differently, and given your application's specific needs, you may find that one LLM suits your application better than another. This may be for reasons related to performance, accuracy, cost, creativity, or many other factors. This solution lets you quickly deploy multiple use

cases enabling you to experiment with and compare different configurations until you've found what meets your needs.

Concepts and definitions

This section describes key concepts and defines terminology specific to this solution:

admin user

Within the context of this guide, the admin user is the one responsible for managing the content contained within the deployment. This user gets access to the Deployment dashboard UI and is primarily responsible for curating the business user experience. This is our primary target customer.

application

A logical group of AWS resources that you want to operate as a unit.

business user

Within the context of this guide, the business user represents the individuals who the use case has been deployed for. They are the consumers of the knowledge base and the customer responsible for evaluating and experimenting with the LLMs.

Deployment dashboard

The Deployment dashboard is a web interface that serves as a management console for admin users to view, manage, and create their *use cases*. This dashboard enables customers to rapidly experiment, iterate, and productionize various AI/ML workloads leveraging LLMs.

DevOps user

Within the context of this guide, the DevOps user is the one responsible for deploying the solution within the AWS account and for managing the infrastructure, updating the solution, monitoring performance, and maintaining the overall health and lifecycle of the solution.

use case

Use cases are isolated applications from the overall solution which integrate with LLMs to enable richer customer experiences by enabling the addition of a natural language interface into new or existing applications. Use cases are deployable through the Deployment dashboard or on their own.

For a general reference of AWS terms, see the AWS Glossary.

Architecture overview

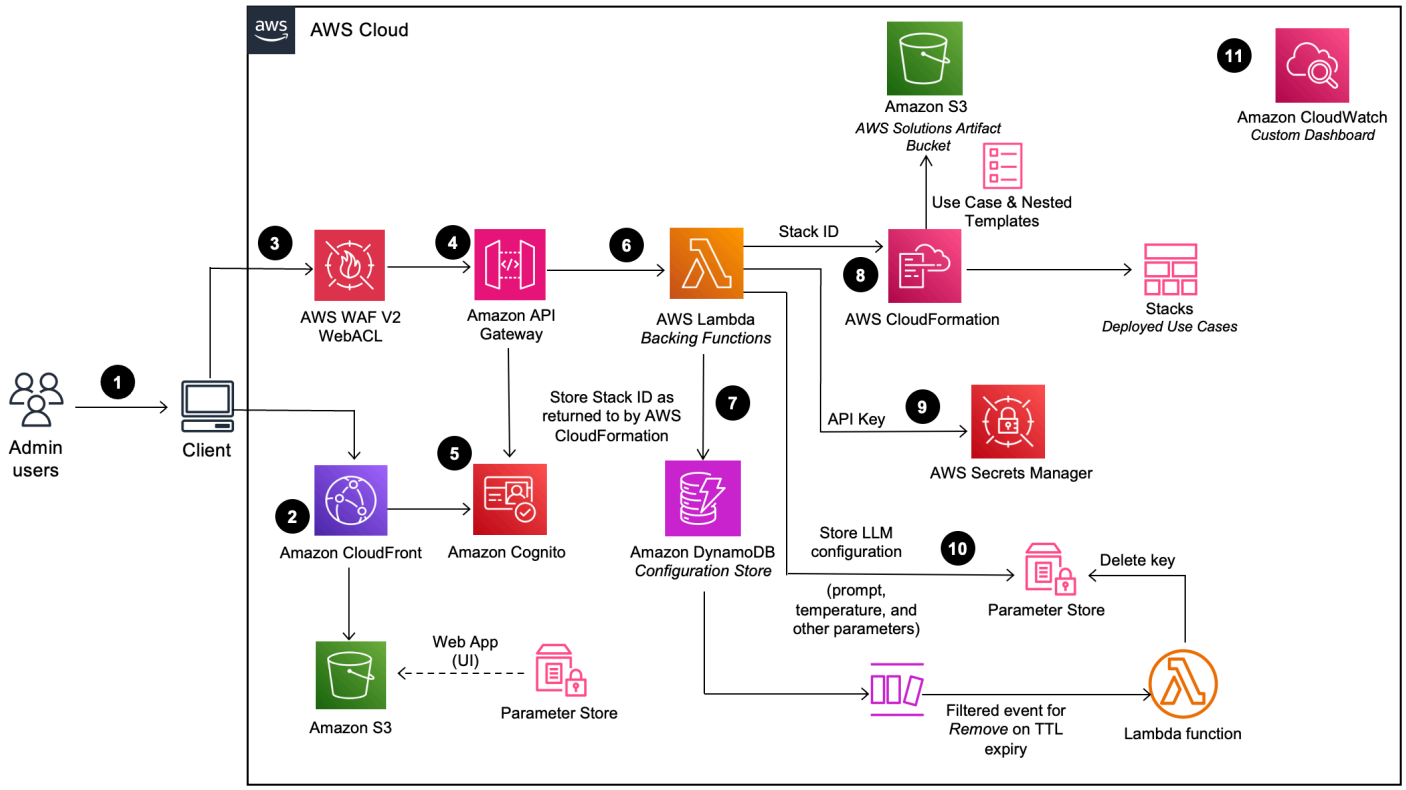
This section provides two reference implementation architecture diagrams for the components deployed with this solution.

Architecture diagrams

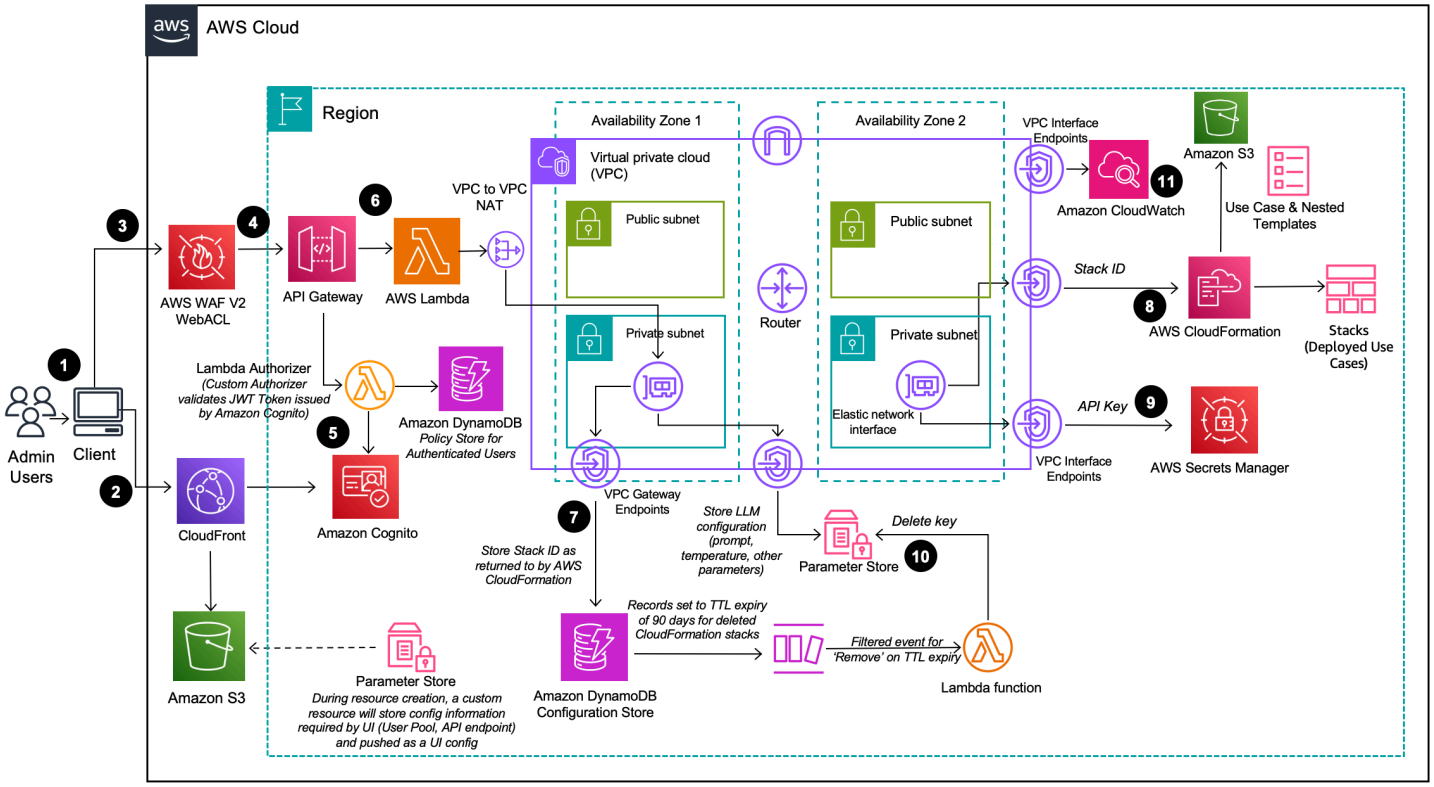
To support multiple use cases and business needs, this solution provides two AWS CloudFormation templates:

1. Deployment dashboard - The Deployment dashboard is a web interface that serves as a management console for admin users to view, manage, and create their use cases. This dashboard enables customers to rapidly experiment, iterate, and productionize various AI/ML workloads leveraging LLMs.
2. Text use case - The Text use case enables users to experience a natural language interface using generative AI. This use case can be integrated into new or existing applications, and is deployable through the Deployment dashboard or independently through a provided URL.

Deployment dashboard



Deployment dashboard architecture (when deployed with VPC option disabled)



Deployment dashboard architecture (when deployed with VPC option enabled)

> **Note**
>
> AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

The high-level process flow for the solution components deployed with the AWS CloudFormation template is as follows:

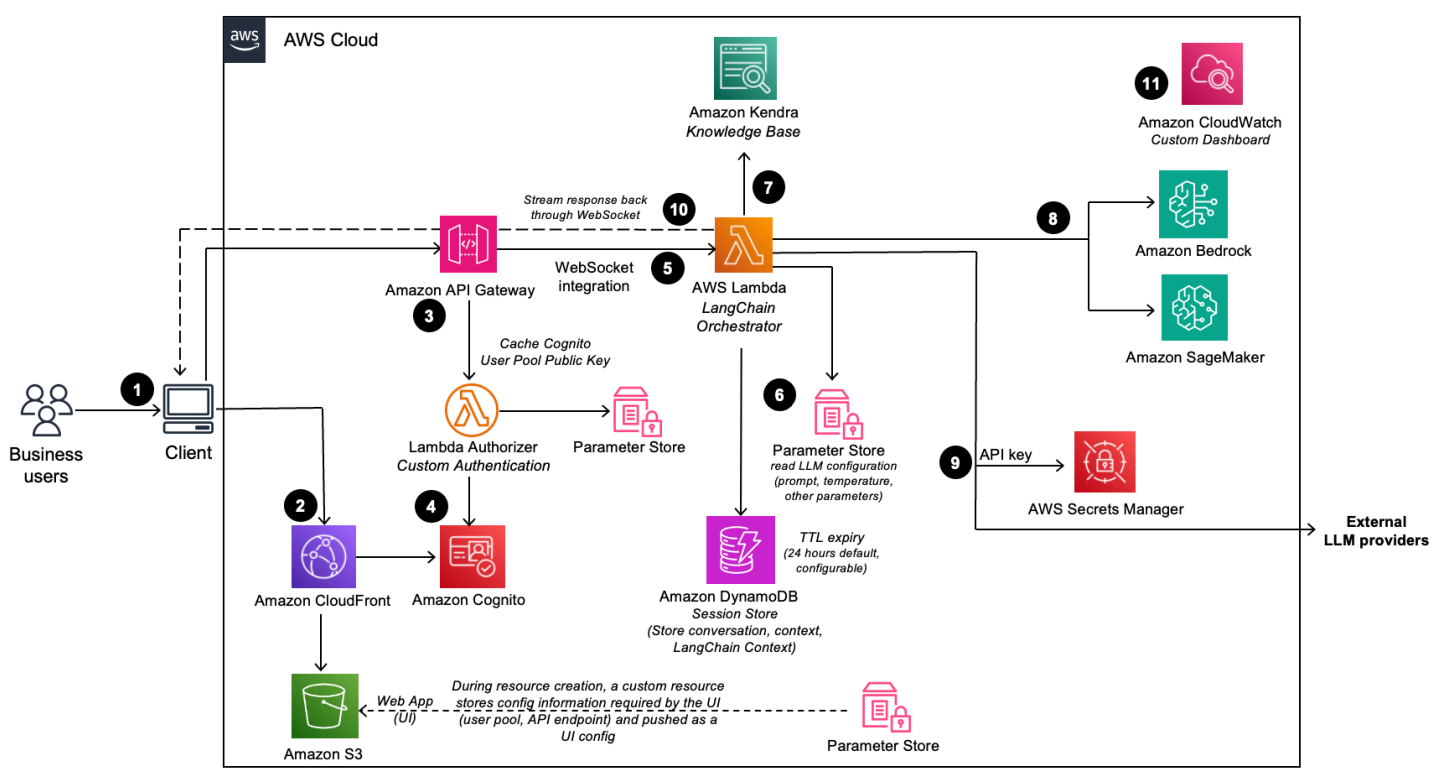
1. Admin users log in to the Deployment Dashboard user interface (UI).
2. Amazon CloudFront delivers the web UI which is hosted in an Amazon Simple Storage Service (Amazon S3) bucket.
3. AWS WAF protects the APIs from attacks. This solution configures a set of rules called a web access control list (web ACL) that allows, blocks, or counts web requests based on configurable, user defined web security rules and conditions.
4. The web UI leverages a set of REST APIs that are exposed using Amazon API Gateway.
5. Amazon Cognito authenticates users and backs both the CloudFront web UI and API Gateway.

6. AWS Lambda provides the business logic for the REST endpoints. This *Backing* Lambda function manages and creates the necessary resources to perform use case deployments using AWS CloudFormation.
7. Amazon DynamoDB acts as a configuration store for the deployment details.
8. When a new use case is created by the admin user, the *Backing* Lambda function initiates a CloudFormation stack creation event for the requested use case.
9. If the configured deployment uses an LLM accessed outside of AWS, then an API key is required and a secret is created in AWS Secrets Manager to store the API key.
10. All of the LLM configuration options provided by the admin user in the deployment wizard are saved in Parameter Store, a capability of AWS Systems Manager. The deployment uses this Parameter Store to configure the LLM at runtime.
11. Using Amazon CloudWatch, this solution collects operational metrics from various services to generate custom dashboards that allow you to monitor the solution's performance and operational health.

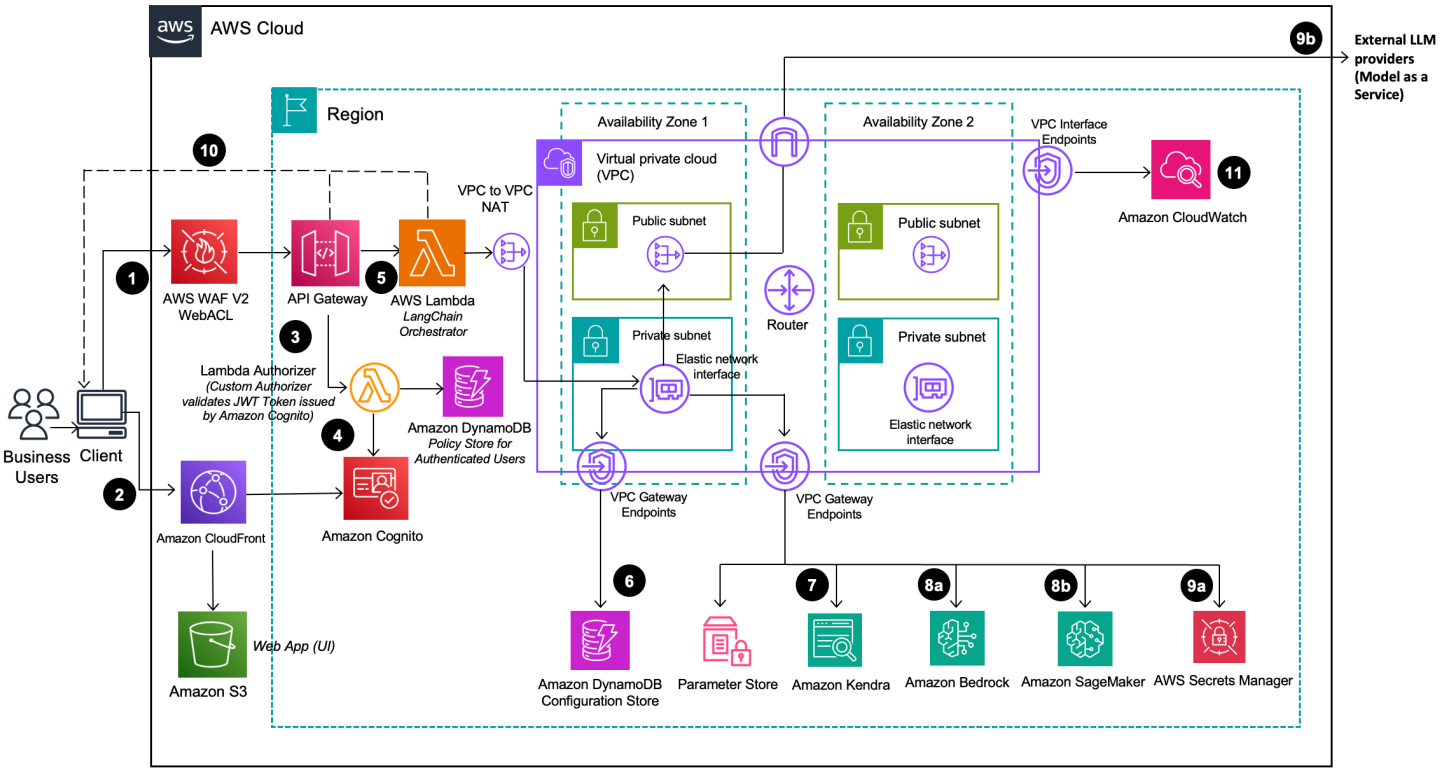
Note

- If you choose to deploy this solution in an Amazon VPC, the solution deploys the appropriate interfaces and configurations.
- Although the Deployment dashboard can be launched in most AWS Regions, the deployed use cases have certain restrictions based on service availability. See Supported AWS Regions for more details.

Text use case



Text use case architecture (when deployed with VPC option disabled)



Text use case architecture (when deployed with VPC option enabled)

The high-level process flow for the solution components deployed with the AWS CloudFormation template is as follows:

1. Business users deploy the use case using the Deployment Dashboard. Business users log in to the use case UI.
2. CloudFront delivers the web UI which is hosted in an S3 bucket.
3. The web UI leverages a WebSocket integration built using API Gateway. The API Gateway is backed by a custom `Lambda Authorizer` function, which returns the appropriate AWS Identity and Access Management (IAM) policy based on the Amazon Cognito group the authenticating user is part of.
4. Amazon Cognito authenticates users and backs both the CloudFront web UI and API Gateway.
5. The *LangChain Orchestrator* is a collection of Lambda functions and layers that provide the business logic for fulfilling requests coming from the business user.
6. The *LangChain Orchestrator* uses Parameter Store and Amazon DynamoDB to get the configured LLM options and necessary session information (such as the chat history).
7. If the deployment has knowledge base enabled, then the *LangChain Orchestrator* leverages Amazon Kendra to run a search query to retrieve document excerpts..
8. Using the chat history, query, and context from Amazon Kendra, the *LangChain Orchestrator* creates the final prompt and sends the request to the LLM hosted on Amazon Bedrock or Amazon SageMaker.
9. If using an LLM accessed outside of AWS, the API key is stored in AWS Secrets Manager. This API key must be obtained before making the API call to the model provider.
10. When the response comes back from the LLM, the *LangChain Orchestrator* streams the response back through the API Gateway WebSocket to be consumed by the client application.
11. Using Amazon CloudWatch, this solution collects operational metrics from various services to generate custom dashboards that allow you to monitor the deployment's performance and operational health.

> **Note**
>
> If you choose to deploy this solution in an Amazon VPC, the solution deploys the appropriate interfaces and configurations.

AWS Well-Architected design considerations

This solution was designed with best practices from the AWS Well-Architected Framework which helps customers design and operate reliable, secure, efficient, and cost-effective workloads in the cloud.

This section describes how the design principles and best practices of the Well-Architected Framework were applied when building this solution.

Operational excellence

This section describes how we architected this solution using the principles and best practices of the operational excellence pillar.

- We built the solution as infrastructure-as-code using Amazon CloudFormation.
- Lambda functions push custom metrics to CloudWatch and a custom CloudWatch dashboard to monitor the health of the solution.
- The solution components are highly modularized, providing the flexibility to choose which components to deploy.

Security

This section describes how we architected this solution using the principles and best practices of the security pillar.

- The Deployment dashboard and all use cases are authenticated and authorized with Amazon Cognito.
- All inter-service communications use AWS IAM roles.
- All solution roles follows least-privilege access; meaning, only the minimum permissions required are granted.
- All data storage including S3 buckets, Amazon DynamoDB, and Amazon Kendra have encryption at rest.

Reliability

This section describes how we architected this solution using the principles and best practices of the reliability pillar.

- Architecture based on serverless paradigm.
- We built the architecture for on-demand, horizontal scalability, and automatic recovery from failure of underlying infrastructure.
- The architecture includes buffering and throttling requests to not overwhelm underlying endpoints.

Performance efficiency

This section describes how we architected this solution using the principles and best practices of the performance efficiency pillar.

- The solution uses Amazon DynamoDB, a fully managed serverless NoSQL database with on-demand scaling.
- The solution uses Amazon S3 for object storage and to host a website (through CloudFront) to provide low cost, scalable, with 11 9s durability.

Cost optimization

This section describes how we architected this solution using the principles and best practices of the cost optimization pillar.

- Where possible, we built the solution to use serverless architecture; so you only pay for what you use.

Sustainability

This section describes how we architected this solution using the principles and best practices of the sustainability pillar.

- The solution's modular, componentized architecture provides the flexibility to customize resources to be provisioned for individual use cases.
- The architecture uses serverless compute and storage, which optimizes resource utilization.
- As a cloud-based solution, this solution benefits from shared resources, networking, power cooling, and physical facilities.

Architecture details

This section describes the components and AWS services that make up this solution and the architecture details on how these components work together.

Deployment dashboard

API Gateway custom authorizers

Beneath the surface, Lambda custom authorizers for API Gateway are used for all API calls (both RESTful and WebSocket based) to validate if a given user has permission to perform an action based on the group(s) they belong to. This custom authorizer is backed by a DynamoDB table containing the policies for each group. On invocation of an API, Amazon API Gateway invokes the custom authorizer Lambda function, which decodes the provided Amazon Cognito access token to determine which user groups the user belongs to. The policy table is then queried by group name to return the relevant policy for that group.

On every new use case deployment, the admin policy is updated to store a new statement allowing the **execute-api:Invoke** action on that use case's API. When use cases are deleted, the corresponding statement is removed from the policy.

For the groups created for an individual use case, only a single statement is present in the policy, allowing the **execute-api:Invoke** action on only that use case's API.

Due to this structure, any user belonging to a use case's group can access that use case's API. A single user can also be manually added to multiple groups to allow that user to use multiple use cases.

> **Warning**
>
> You can also manually edit the policies for a given group in the policy table if you want to grant access to a new use case to an existing group of users. The use case group is deleted when the use case is deleted (even if you have made manual edits), so proceed with caution when deleting a use case.

In the case where a use case stack is deployed standalone (without the use of the Deployment dashboard), an Amazon Cognito user pool is created for that deployment containing a single user

with access to that use case's API. This user pool belongs only to this use case and is not shared across other standalone deployments.

Text use case

Streaming support

In a chat application, latency is an important metric to enable a responsive user experience. The potential for LLM inferences to take from seconds to minutes, provides challenges in how to best serve content to customers. For this reason, several LLM providers allow streaming responses back to the caller. Instead of waiting for the entire inference to complete before returning a response, each token can be returned when it is available.

To support the use of this feature, the Text use case has been designed to use a WebSockets API to back the chat experience. This WebSocket is deployed through Amazon API Gateway. The use of a WebSockets API enables a connection to be created at the beginning of a chat session and for responses to be streamed through that socket. This allows frontend applications to provide a better user experience.

> **Note**
>
> Even if a model provides streaming support, this does not necessarily mean that the solution will be able to stream responses back through the WebSockets API. There is a need for the solution to enable custom logic to support streaming for each model provider. If streaming is available, admin users will be able to enable/disable this feature at deployment time.

AWS services in this solution

AWS service	Description
Amazon CloudFront	**Core**. CloudFront serves the web content hosted in Amazon S3.
Amazon S3	**Core**. Amazon S3 hosts the static web content.

AWS service	Description
Amazon API Gateway	**Core**. This service provides the REST APIs for the Deployment dashboard and the WebSocket API for the use case.
Amazon Cognito	**Core**. This service handles user management and authentication for the API.
AWS Lambda	**Core**. The solution uses Lambda functions to: • Back the REST and WebSocket API endpoints • Handle the core logic of each use case orchestrator • Implement custom resources during CloudFormation deployment
Amazon DynamoDB	**Core**. DynamoDB stores deployment information and configuration details for the Deployment dashboard. It stores chat history and conversation IDs in the Text use case to enable conversation history and query disambiguation.
AWS CloudFormation	**Core**. This solution is distributed as a CloudFormation template, and CloudForm ation deploys the AWS resources for the solution.
AWS WAF	**Supporting.** AWS WAF is deployed in front of the API Gateway deployment to protect it.
AWS Systems Manager	**Supporting**. Systems Manager provides application-level resource monitoring and visualization of resource operations and cost data. Also used to store configuration data in Parameter Store.

AWS service	Description
AWS Secrets Manager	**Supporting**. This service is used to securely store the necessary API keys if a deployment leverages a foundation model (FM) hosted outside of the AWS network.
Amazon CloudWatch	**Supporting**. This solution publishes logs from solution resources to CloudWatch Logs, and publishes metrics to CloudWatch metrics. The solution also creates a CloudWatch dashboard to view this data.
Amazon Bedrock	**Optional**. The solution leverages Amazon Bedrock to access a collection of FMs and is the recommended integration to keep your data from leaving the AWS network.
Amazon SageMaker	**Optional**. The solution can integrate with an Amazon SageMaker inference endpoint to access FMs that are hosted within your AWS account and Region and is a preferred integration to keep your data from leaving the AWS network. **Note** You must deploy the solution in the same Region where the inference endpoint is available.

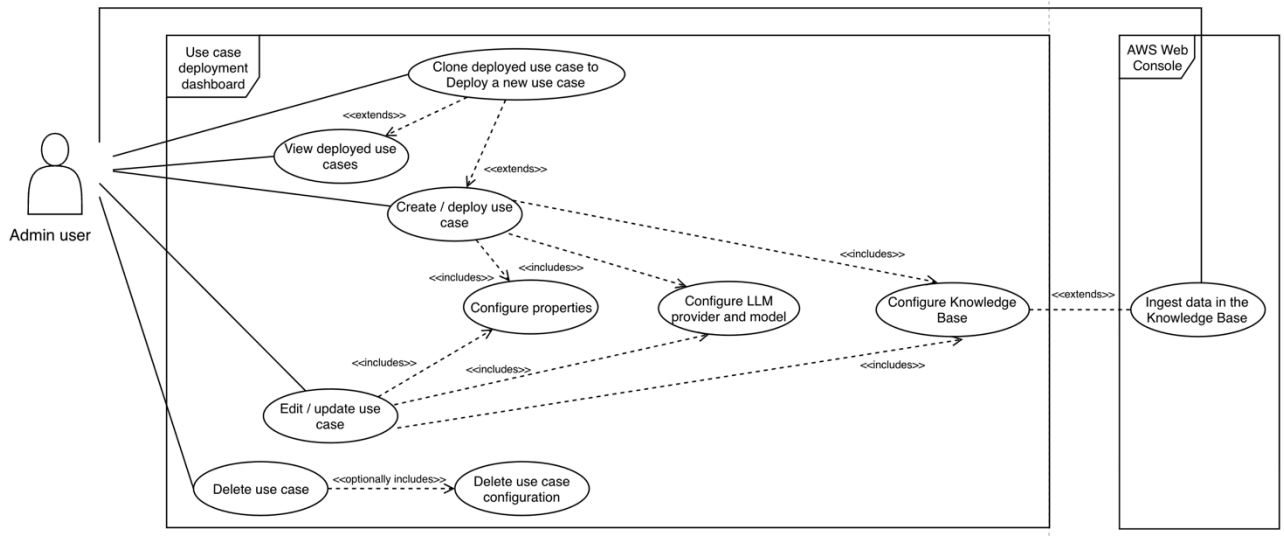
AWS service	Description
Amazon Kendra	**Optional**. In the Text use case, admin users can optionally decide to connect a Amazon Kendra index to use as a knowledge base for the conversation with the LLM. This can be used to inject new information into the LLM giving it the ability to use that information in its responses.
Amazon Virtual Private Cloud	**Optional**. The solution provides the option to deploy components with a VPC-enabled configuration. While deploying the solution with a VPC-enabled configuration, you have the option to let the solution create a VPC for you, or use an existing VPC that exists in the same account and Region where the solution will be deployed (Bring Your Own VPC). If the solution creates the VPC, it creates the necessary network components that includes, subnets, security groups and its rules, route tables, network ACLs, NAT Gateways, Internet Gateways, VPC endpoints, and its policies.

How the Generative AI Application Builder on AWS solution works

The admin user primarily interfaces with the Deployment dashboard to view, create, and manage new and existing use case deployments. Through this dashboard, the admin user has access to the following actions:

- View list of deployments
- Create new deployments
- Edit existing deployments
- Clone a deployment's configuration to create a new deployment

- Delete a deployment (deprovision the resources through a CloudFormation delete)
- Permanently delete the configuration details of a deployment

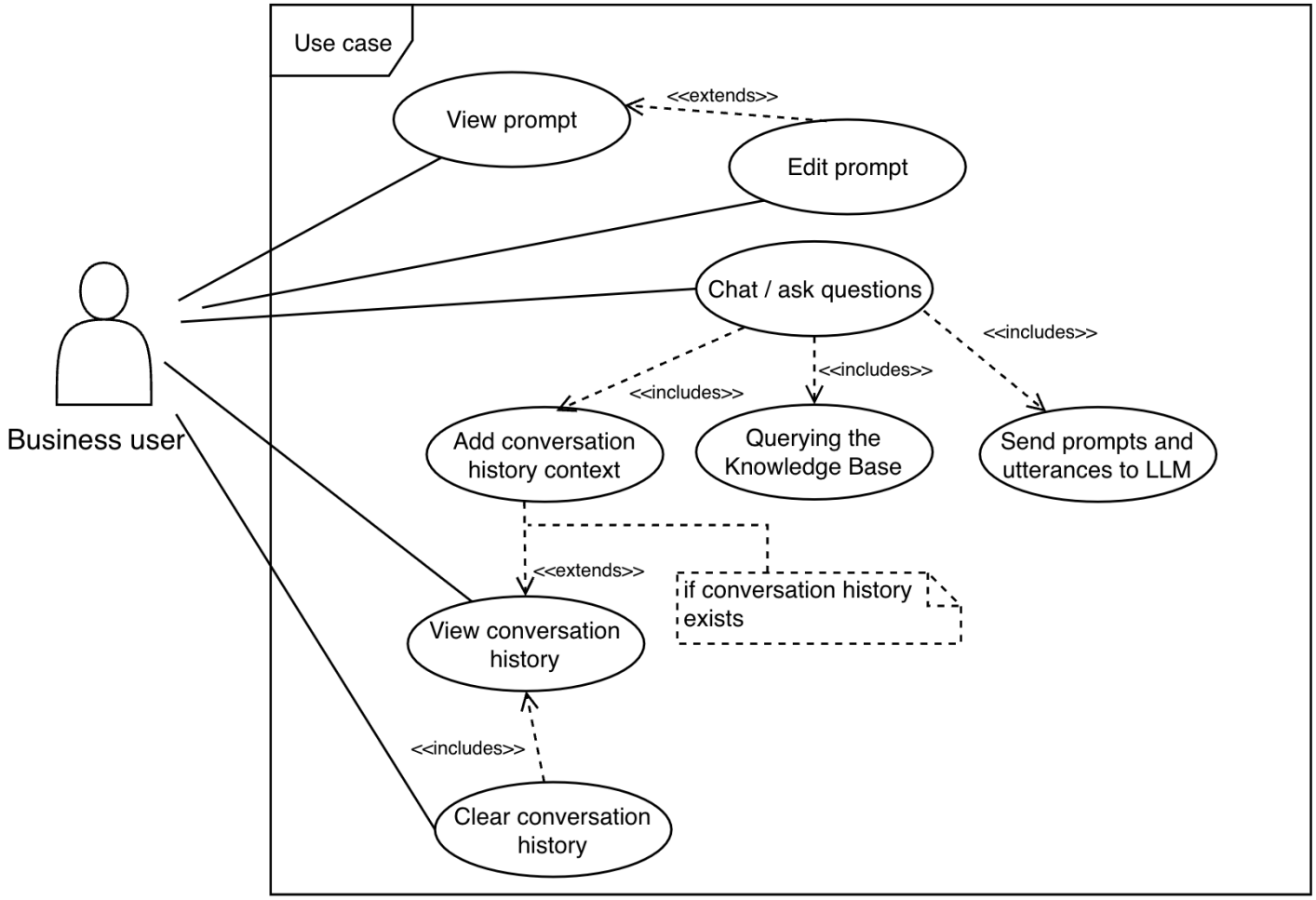


Use case diagram for the admin user of the Deployment dashboard

> **Note**
>
> The admin user might not have direct access to the AWS console. In that case, the admin user must work with the DevOps user to support actions such as ingesting data into a Kendra knowledge base.

For the Text use case, the business user gets access to a user interface enabling them to chat with the LLM. The specifics of this configuration are controlled by the deployment settings configured by the admin user. In the Text use case, the business user has access to the following actions:

- Send messages through the chat interface
- View conversation history
- Clear the conversation history



Use case diagram for the business user of the Text use case

Plan your deployment

This section describes the cost, security, Region, and quota considerations for planning your deployment.

> **Important**
>
> This solution leverages Amazon Bedrock as the primary service for accessing AI-generated models. You must first request access to models before they are available for use within the solution. For details, refer to Model access in the *Amazon Bedrock User Guide*.

Cost

With this AWS Solution, you pay only for the resources you use and there are no minimum fees or setup charges. Users pay for the dashboard used to launch Generative AI use cases and, and for any use cases that are deployed. The cost of deployed use cases depends on the configurations. Example configurations:

1. A simple Deployment dashboard which costs approximately $20 USD per month.
2. A simple production-ready chatbot use case deployed with default settings running in US East (N. Virginia), powered by Amazon Bedrock without access to documents, which also costs around $20 USD per month.
3. A scaled system in an Amazon VPC use case that supports 8000 queries per day over tens of thousands of documents, which costs around $2000 USD per month. The cost of the use case will vary depending on the configuration, such as Text use cases with different model providers, with or without Retrieval Augmented Generation (RAG) enabled, and so on.

Workload description	Estimated cost (USD/month)
Sample cost for Deployment dashboard	$20/month
Sample costs for a text-based proof of concept (includes Deployment dashboard and 1 Text use case, ~100 interactions per day)	$40/month

Workload description	Estimated cost (USD/month)
Sample costs for a highly scalable generative AI query engine (includes Deployment dashboard, 1 Text use case, and an Amazon Kendra Index for RAG upto 100K documents with ~8000 queries per day, with VPC enabled	$2000/month

Important

These examples are only intended to help you estimate the costs for your specific workloads. The use of different LLMs, configurations, or AWS services can change your costs (example, serverless/on-demand billing vs. provisioned/time-billed). To manage costs, we recommend creating a budget through AWS Cost Explorer. Prices are subject to change. For full details, refer to the pricing webpage for each AWS service used in this solution.

Sample costs for running the Deployment dashboard only

The following table provides the cost breakdown for a Deployment dashboard with default parameters and 100 active users in the US East (N. Virginia) Region for one month, which will cost about $20/month.

AWS service	Dimensions	Cost [USD]
Amazon API Gateway, DynamoDB, Amazon CloudFront, Amazon S3, Lambda, AWS Systems Manager Parameter Store	5,000 512 KB REST API calls per month without caching enabled	$1.97
Amazon Cognito	100 active users per month with advanced security features enabled and no users	$5.55

AWS service	Dimensions	Cost [USD]
	signing in through SAML or OIDC federation	
AWS WAF	10,000 web requests across 1 web ACL and 7 defined rules without any rule groups	$12.60
Total Deployment dashboard cost		**$20.12**

Sample costs for a text-based proof of concept

A Deployment dashboard can have many use cases deployed at a given time. The following table shows the cost breakdown of a use case deployed without RAG for 1 business user performing 100 queries per day with the LLM. Queries are sent as a text message on the WebSocket and the response is streamed back as tokens with the assumption that streaming is enabled. With an Amazon Bedrock Titan Text Express model, the cost of running this use case is about $15/month.

AWS service	Dimensions	Cost [USD]
Amazon API Gateway (WebSocket), CloudFront, Lambda, Amazon S3, AWS Systems Manager Parameter Store	100 chat interactions per day. Average message size 32 KB per message and 5 minutes per connection.	$0.61
Amazon CloudWatch	1.5 GB CloudWatch logs with verbose mode on for experimentation	$7.23
Amazon DynamoDB	Conversation history table, 1 GB storage LLM configuration table, 1 GB storage	$3.05

AWS service	Dimensions	Cost [USD]
Sub total of costs (not including LLMs)		**$11.15**
Amazon Bedrock (Titan Text Express)	Assumptions for 100 interactions per day: • Monthly cost for 150K input tokens per day = $3.60 • Monthly cost for 16K output tokens per day = $0.768	$4.37
Total application cost with Amazon Bedrock (Titan Text Express)	**$11.15 (Use Case cost) + $4.37 (Amazon Bedrock cost)**	**$15.52**

Note

The costs of inference calls made to services outside the AWS network are not included in these estimates. See the pricing guide of your LLM provider if not using an AWS service. Pricing guides for AWS services can be found at: Amazon Bedrock pricing and Amazon SageMaker pricing.

Sample costs for a highly scalable generative AI query engine

The following table provides the cost breakdown of a RAG-enabled use case with a Kendra index supporting 8000 interactions/day. With Amazon Bedrock's Titan Text Express model as the LLM, this use case costs about $2000/month

AWS service	Dimensions	Cost [USD]
Amazon API Gateway (WebSocket)	8000 chat interactions per day. Average message size	$38.89

AWS service	Dimensions	Cost [USD]
	32 KB per message and 5 minutes per connection.	
Amazon CloudFront	240,000 requests per month with 100 GB data transferred out to the internet and 1 GB data transferred out to the origin	$8.76
Amazon Bedrock (Titan Text Express)	Assumptions: Input tokens = promptTem plate (400) + context (400)+ chatHistory (1080) + query Input tokens (20)= 1,900 Output tokens = 160 (average) With 8,000 transactions a day, monthly cost ($21.94 daily cost x 30) = $658.20 USD	$658.20
Amazon CloudWatch	24 metrics using 5 GB data ingested for logs and 1 dashboard	$9.72
Amazon DynamoDB	DynamoDB table to keep track of conversation history with each record up to 1 KB data, 8,000 read and writes per day	$11.70

AWS service	Dimensions	Cost [USD]
Amazon Kendra	0-8,000 queries a day and up to 100,000 documents with Amazon Kendra Enterpris e Edition with 0-50 data sources	$1,008.00
AWS Lambda	Container size - 128 MB, 512 MB ephemeral storage, 2 Lambda functions used for authorization Container size - 256 MB, 512 MB ephemeral storage, 5 requests per second with 20 seconds average compute time	$20.89
Amazon S3, AWS Secrets Manager, AWS Systems Manager Parameter Store	1 MB for storage of any use case artifacts	$0.53
Total use case cost		**$1,756.29**

Note

- The costs of API calls made to any services outside of the AWS network are not included in these estimates. See the pricing guide of your LLM provider if not using Amazon Bedrock.
- You can share the Amazon Kendra index between use cases, but this can drive up the number of queries per index. If this falls outside the Amazon Kendra Enterprise edition, additional charges will apply.

Incremental cost of enabling Amazon VPC for a use case

The following table provides the cost breakdown of enabling Amazon VPC for a use case deployed in two AZs.

AWS service	Dimensions	Cost [USD]
Amazon NAT Gateway	Assumption: 2 AZ deployment, with a NAT Gateway in each AZ. 100 GB of data processed through NAT Gateway 730 hours, 100 GB data processed per month	$74.70
AWS PrivateLink (VPC Endpoints)	Assumptions: 2 AZ deployment, with 1 private subnet in each AZ and 1 VPC Endpoint with 2 elastic network interfaces (ENIs). 6 VPC endpoints, 2 ENIs per VPC endpoint, 730 hours with 1,024 GB data processed in a month	$97.84
Public IPv4 address	Assumption: 2 AZ deployment, 1 public subnet in each AZ with a NAT Gateway in each public subnet. Each NAT Gateway configured with 1 active public IPv4. 2 active public IPv4 address x 730 hours in a month x $0.005 hourly charge = $7.3 USD	$7.30
Additional cost		**$179.93**

AWS service	Dimensions	Cost [USD]
(for Amazon VPC)		

Security

When you build systems on AWS infrastructure, security responsibilities are shared between you and AWS. This shared responsibility model reduces your operational burden because AWS operates, manages, and controls the components including the host operating system, virtualization layer, and physical security of the facilities in which the services operate. For more information about AWS security, visit AWS Cloud Security.

Using foundation models on Amazon Bedrock

Amazon Bedrock hosts a collection of models from Amazon Titan models to other leading foundation models (FMs). When using Amazon Bedrock, all models are hosted within the AWS infrastructure. This means that when using Amazon Bedrock as the LLM provider, all of your inference requests will remain within the AWS network and network traffic will not leave your Region.

Note

All foundation models (FMs) available through Amazon Bedrock are hosted directly on AWS infrastructure managed and owned by AWS. Model providers do not have access to customer data such as prompts and continuations, or Amazon Bedrock service logs. For additional information about Amazon Bedrock's security posture, refer to Data protection in Amazon Bedrock in the *Amazon Bedrock User Guide*.

Anthropic and Hugging Face integrations

Important

Generative AI Application Builder on AWS allows you to build and deploy generative artificial intelligence applications on AWS by engaging the generative AI model of your choice, including third-party generative AI models that you can choose to use that AWS does not own or otherwise have any control over ("Third-Party Generative AI Models").

Your use of the Third-Party Generative AI Models is governed by the terms provided to you by the Third-Party Generative AI Model providers when you acquired your license to use them (for example, their terms of service, license agreement, acceptable use policy, and privacy policy).
You are responsible for ensuring that your use of the Third-Party Generative AI Models comply with the terms governing them, and any laws, rules, regulations, policies, or standards that apply to you.
You are also responsible for making your own independent assessment of the Third-Party Generative AI Models that you use, including their outputs and how Third-Party Generative AI Model providers use any data that might be transmitted to them based on your deployment configuration. AWS does not make any representations, warranties, or guarantees regarding the Third-Party Generative AI Models, which are "Third-Party Content" under your agreement with AWS. Generative AI Application Builder on AWS is offered to you as "AWS Content" under your agreement with AWS.

If choosing to integrate with a model accessed outside of AWS, customers must evaluate the security considerations of data leaving their AWS account. The customer is required to provide a valid API key at the time of deployment. This API key will be stored in AWS Secrets Manager for the duration of the deployment's lifecycle.

The primary piece of data sent to the foundation model is the prompt to perform inference on. Depending on the use case, the prompt can contain the user's input, previous interactions (for example, chat history), and document excerpts sourced from the configured knowledge base (for example, Amazon Kendra search result).

Note

- Ensure you only ingest the appropriate documents into your knowledge base. Any results returned by the knowledge base is eligible for inclusion into the prompt; and therefore, being sent to the LLM. If performing inference on a model accessed outside of AWS, ensure you audit the documents contained within your knowledge base.
- We recommend that you implement a policy to periodically rotate your secrets based on your organization's security policies. The solution does not automatically manage rotations on your behalf.

IAM roles

IAM roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles that grant the solution's AWS Lambda functions access to create Regional resources.

Managing your own Amazon VPC

When deploying the solution with an Amazon VPC, you have the option to use an existing Amazon VPC in your AWS account and Region. We recommended that you make your VPC available in at least two availability zones to ensure high availability. Your VPC must also have the following VPC endpoints and their associated IAM policies for your VPC and route table configurations.

Deployment dashboard

1. Gateway endpoint for Amazon DynamoDB.
2. Interface endpoint for Amazon CloudWatch.
3. Interface endpoint for AWS Systems Manager Parameter Store.

> **Note**
>
> The solution only requires com.amazonaws.region.ssm.

4. Interface endpoint for AWS CloudFormation.
5. Optional: If the dashboard will be used to deploy use cases connected to Anthropic or Hugging Face, then an Interface endpoint for AWS Secrets Manager is needed.

Use cases

1. Gateway endpoint for Amazon DynamoDB.
2. Interface endpoint for Amazon CloudWatch.
3. Interface endpoint for AWS Systems Manager Parameter Store.

> **Note**
>
> The solution only requires `com.amazonaws.region.ssm`.

4. Optional: If the deployment will use Amazon Kendra as a knowledge base, then an interface endpoint for Amazon Kendra is needed.
5. Optional: if the deployment will use any LLM under Amazon Bedrock, then an interface endpoint for Amazon Bedrock is needed.

> **Note**
>
> The solution only requires `com.amazonaws.region.bedrock-runtime`.

6. Optional: If the deployment will use Amazon SageMaker for the LLM, then an interface endpoint for Amazon SageMaker is needed.
7. Optional: If the deployment uses the Anthropic or Hugging Face connectors, then an API key is used. That is, an interface endpoint for AWS Secrets Manager is needed.

> **Note**
>
> The solution will not delete or modify the VPC configuration when using the **Bring your own VPC deployment** option. However, it will delete any VPCs that are created by the solution in the **Create a VPC for me** option. For this reason, you must be careful when sharing a solution-managed VPC across stacks/deployments.
> For example, deployment A uses **Create a VPC for me** option. Deployment B uses **Bring my own VPC** using the VPC created by deployment A. If deployment A is deleted before deployment B, then deployment B will no longer work because the VPC has been deleted. Also because deployment B is using the ENIs created by the Lambda functions, deleting deployment A might have errors and retention of residual resources.

Let the solution build an Amazon VPC for you

If you select the option to let the solution build an Amazon VPC, it will deploy as a 2-AZ architecture by default with a CIDR range 10.10.0.0/20. You have the option to use Amazon VPC IP Address Manager (IPAM), with 1 public subnet and 1 private subnet in each AZ. The solution creates

NAT Gateways in each of the public subnets, and configures Lambda functions to create the ENIs in the private subnets. Additionally, this configuration creates route tables and its entries, security groups and its rules, network ACLs, VPC endpoints (gateway and interface endpoints).

Amazon CloudFront

This solution deploys a web console hosted in an Amazon S3 bucket. To help reduce latency and improve security, this solution includes a CloudFront distribution with an origin access identity, which is a CloudFront user that provides public access to the solution's website bucket contents. For more information, see Restricting Access to Amazon S3 Content by Using an Origin Access Identity in the *Amazon CloudFront Developer Guide*.

Supported AWS Regions

Important

This solution optionally uses the Amazon Bedrock and Amazon Kendra services, which are not currently available in all AWS Regions. You must launch this solution in an AWS Region where these services are available. For the most current availability of AWS services by Region, see the AWS Regional Services List.

Generative AI Application Builder on AWS is supported in the following AWS Regions:

Region name	
US East (Ohio)	Canada (Central)
US East (N. Virginia)	Europe (Frankfurt)
US West (Northern California)	Europe (Ireland)
US West (Oregon)	Europe (London)
Asia Pacific (Mumbai)	Europe (Milan)
Asia Pacific (Seoul)	Europe (Paris)

Region name	
Asia Pacific (Singapore)	Europe (Stockholm)
Asia Pacific (Sydney)	Middle East (Bahrain)
Asia Pacific (Tokyo)	South America (São Paulo)

Note

If using a foundation model accessed outside of AWS in your deployments, check with the model provider which Regions their APIs are available in. If their APIs are only available in certain Regions, you might experience instability in the form of high latency or even time outs. It's also important to check with your organization's legal and compliance teams to evaluate the considerations of data crossing regional boundaries.

Quotas

Service quotas, also referred to as limits, are the maximum number of service resources or operations for your AWS account.

Quotas for AWS services in this solution

Make sure you have sufficient quota for each of the services implemented in this solution. For more information, refer to AWS service quotas.

Use the following links to go to the page for that service. To view the service quotas for all AWS services in the documentation without switching pages, view the information in the Service endpoints and quotas page in the PDF instead.

Deploy the solution

This solution uses AWS CloudFormation templates and stacks to automate its deployment. The CloudFormation template specifies the AWS resources included in this solution and their properties. The CloudFormation stack provisions the resources that are described in the template.

Deployment process overview

Before you launch the solution, review the cost, architecture, security, and other considerations discussed in this guide.

> **Important**
>
> If you plan to use Amazon Bedrock, you must request access to models before they are available for use. Refer to Model access in the *Amazon Bedrock User Guide* for more details.

Time to deploy: Approximately 10 minutes

Step 1: Launch the Deployment dashboard stack

Step 2: Deploy use case

Step 3: Ingest data into knowledge base

> **Important**
>
> This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. AWS owns the data gathered though this survey. Data collection is subject to the AWS Privacy Policy.
>
> To opt out of this feature, download the template, modify the AWS CloudFormation mapping section, and then use the AWS CloudFormation console to upload your updated template and deploy the solution. For more information, see the Anonymized data collection section of this guide.

AWS CloudFormation template

You can download the CloudFormation template for this solution before deploying it.

View template

generative-ai-application-builder-on-aws.template - Use this template to launch the solution and all associated components. The default configuration deploys the core and supporting services found in the AWS services in this solution section, but you can customize the template to meet your specific needs.

AWS CloudFormation resources are created from AWS Cloud Development Kit (AWS CDK) constructs.

This AWS CloudFormation template deploys Generative AI Application Builder on AWS in the AWS Cloud.

Step 1: Launch the Deployment dashboard stack

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 10 minutes

1. Sign in to the AWS Management Console and select the button to launch the `generative-ai-application-builder-on-aws.template` CloudFormation template.

   Launch solution

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

   Note

   This solution uses Amazon Kendra and Amazon Bedrock, which are not currently available in all AWS Regions. If using these features, you must launch this solution in

an AWS Region where these services are available. For the most current availability by Region, see the AWS Regional Services List.

3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, see IAM and STS Limits in the *AWS Identity and Access Management User Guide*.
5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
Admin User Email	`<Requires input>`	The email address of the admin user who will have access to the Deploymen t dashboard. An Amazon Cognito user will be created with permissions to deploy and manage use cases.

6. Choose **Next**.
7. On the **Configure stack options** page, choose **Next.**
8. On the **Review** page, review and confirm the settings. Select the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately 10 minutes.

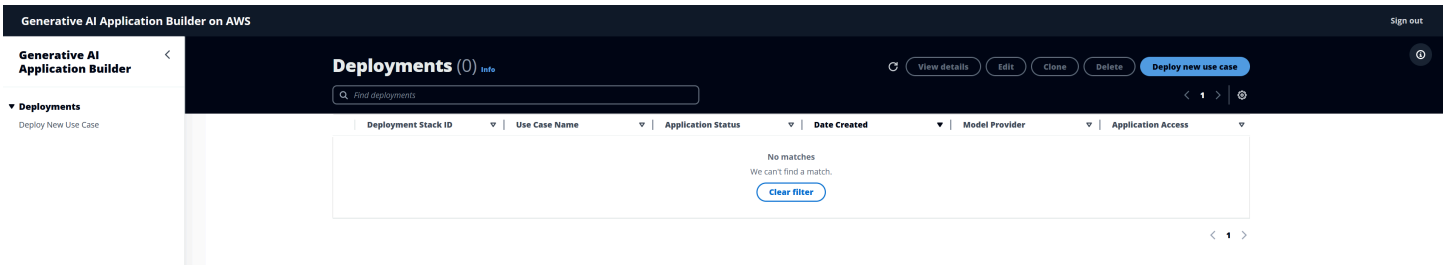
Step 2: Deploy use case

Once the stack has been successfully deployed, a sign up email is sent to the configured admin user email. Using those credentials, the admin user can sign in to the Deployment dashboard to use the web application.

> **Note**
>
> The DevOps user with access to the AWS Management Console must provide the admin user with the CloudFront URL of the Deployment dashboard UI when the stack completes. The URL can be found in the **Outputs** tab of the CloudFormation stack.

1. Sign in to the Deployment dashboard as an admin user.
2. On the application landing page, choose **Deploy new use case**.

   This launches the deployment wizard, which walks you through building the use case.



Deployment dashboard landing page - fresh deployment

> **Note**
>
> If you need to add additional users to your deployment, refer to the Managing Cognito user pool for more details.

Step 3: Ingest data into knowledge base

If deploying a knowledge base as part of your deployment (for example, enabling RAG in the Text use case), you must ingest data into your knowledge base after stack creation. This is to ensure that your deployment works as expected. Admin users might have to seek support from the DevOps user to help manage the lifecycle of AWS resources (for example, ingesting documents into an Amazon Kendra index).

Note

If using Amazon Kendra as your knowledge base, refer to the Amazon Kendra Developer Guide for information on how to use various data source connectors to help you ingest data from a wide selection sources.

Important

To prevent accidental data loss, the solution does not automatically delete the knowledge base when a deployment or stack is deleted. If you want to delete your knowledge base and stop incurring costs, see the Manual uninstall section for details on which resources are retained and how to clean them up.

Post-deployment configuration

This section provides recommendations for configuring the solution after deployment.

Amazon S3 bucket versioning, lifecycle policies, and cross-Region replication

This solution doesn't enforce lifecycle configurations on the buckets it creates. We recommend the following:

- Setting lifecycle configurations for production deployments. For details, see Setting lifecycle configuration on a bucket in the *Amazon Simple Storage Service User Guide*.
- Enabling versioning and cross-Region replication for Amazon S3 buckets based on the use case for which the solution is deployed.

Amazon DynamoDB backups

This solution uses DynamoDB for several purposes (see AWS services in this solution). The solution doesn't enable backups for the tables it creates. We recommend creating a backup of this feature for production deployments. See Backing up a DynamoDB table and Using AWS Backup for DynamoDB for details.

Amazon CloudWatch dashboard and alarms

The solution deploys a custom dashboard in CloudWatch to render charts from custom published metrics and AWS service metrics. We recommend creating CloudWatch alarms and adding notifications based on the use case for which the solution is deployed.

Custom web domains with TLS v1.2 or higher certificates

The solution deploys a web UI and Edge Optimized API Gateway using CloudFront. CloudFront's domain doesn't enforce TLS v1.2 or higher certificates. We recommend creating a custom domain using Amazon Route 53, creating a certificate using AWS Certificate Manager, or using an existing certificate if your organization has one.

For additional details, refer to the Amazon Route 53 Developer Guide and Choosing a minimum TLS version for a custom domain in API Gateway.

Scaling with Amazon Kendra

This solution provides the ability to use Amazon Kendra to perform NLP-powered intelligent search across the ingested documents. You can increase the capacity of Amazon Kendra using the following CloudFormation parameters for larger workloads:

Parameter	Default	Description
Amazon Kendra additional query capacity	`0`	The amount of extra query capacity for an index and GetQuerySuggestions capacity. An additional capacity unit for an index provides approximately 8,000 queries per day.
Amazon Kendra additional storage capacity	`0`	The amount of extra storage capacity for an index. A single capacity unit provides 30 GB of storage space or 100,000 documents, whichever reaches first.

Parameter	Default	Description
Amazon Kendra edition	`Developer`	Amazon Kendra provides Developer and Enterprise Editions to create indexes. For more information about the differences between Amazon Kendra Editions, see Amazon Kendra pricing.

To modify the values of these CloudFormation parameters, select the appropriate values at the time of stack deployment. For more information on query and storage capacity units, see Adjusting capacity.

> **Note**
>
> If the Text use case is not deployed with RAG enabled, then an Amazon Kendra index is not used or created.

Additional security considerations

Based on the use case for which you deploy the solution, review the following security recommendations:

- **Customer managed AWS KMS encryption keys** - The solution uses AWS managed AWS KMS keys by default, since these are available at no additional cost. Review your use case to determine if you should update the solution to use customer managed AWS KMS keys.
- **API Gateway throttling rules** - The solution deploys with default throttling rules on API Gateway. Based on your use case and expected transaction volumes, we recommend that you configure throttling for the APIs. For details, see Throttle API requests for better throughput in the *Amazon API Gateway Developer Guide*.
- **Enabling AWS CloudTrail** - As a recommended security practice, consider enabling AWS CloudTrail in the AWS account where the solution is deployed to log API calls in the AWS account. For details, see the AWS CloudTrail User Guide.

- **Drift detection** - We recommend configuring drift detection on CloudFormation stacks to identify and be notified of unintentional or malicious changes to the deployed solution stack. For details, see Implementing an alarm to automatically detect drift in AWS CloudFormation stacks.
- **Cognito JSON Web Tokens (JWTs)** – The solution uses Amazon Cognito-issued JWTs to authenticate with the REST API endpoints. We configured the solution with a five-minute expiry for ID tokens and access tokens. When a user logs out, their ability to generate new tokens is revoked (refresh token is revoked). However, until the expiry of the current token, any requests to the API endpoint will be successfully authenticated, since they have a valid token. Review the security considerations for your use case and adjust the token validity period.

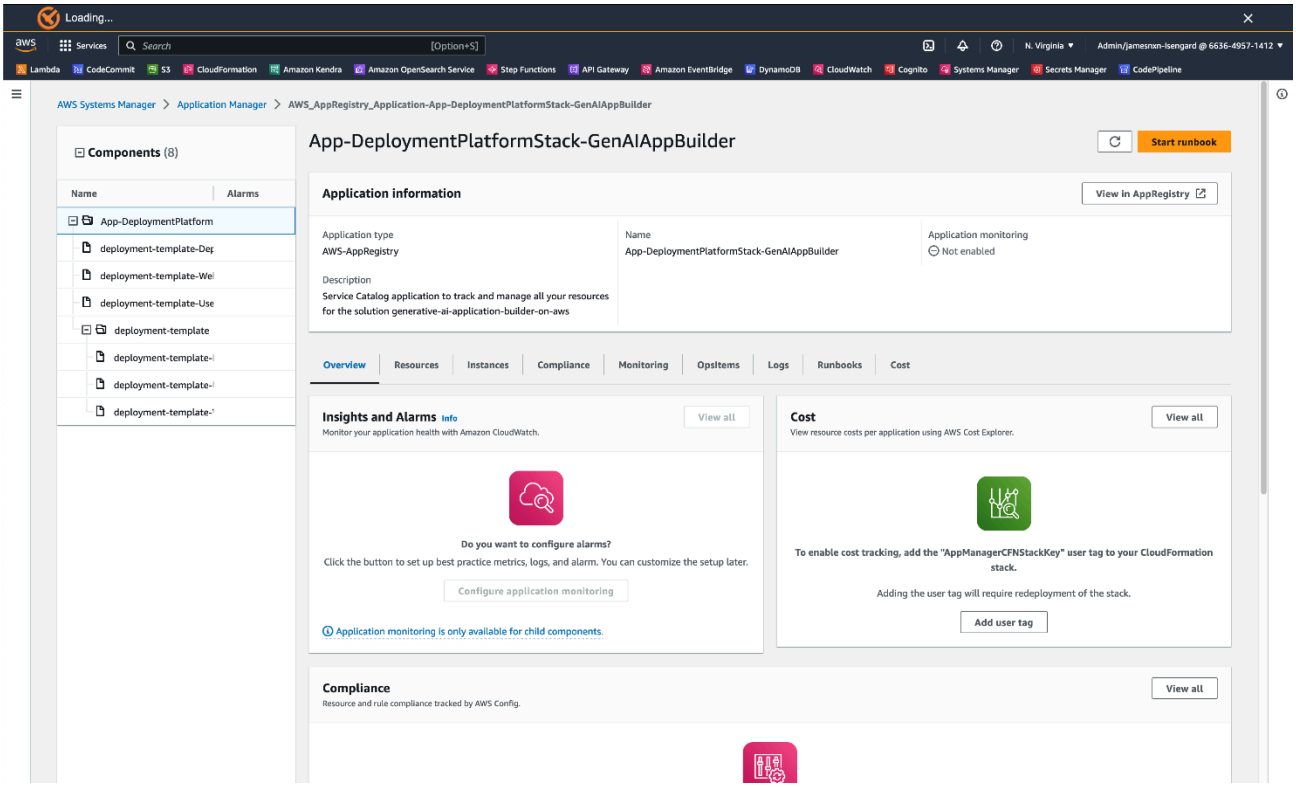
Monitoring the solution with Service Catalog AppRegistry

The Generative AI Application Builder on AWS solution includes a Service Catalog AppRegistry resource to register the CloudFormation template and underlying resources as an application in both Service Catalog AppRegistry and AWS Systems Manager Application Manager.

AWS Systems Manager Application Manager gives you an application-level view into this solution and its resources so that you can:

- Monitor its resources, costs for the deployed resources across stacks and AWS accounts, and logs associated with this solution from a central location.
- View operations data for the resources of this solution in the context of an application. For example, deployment status, CloudWatch alarms, resource configurations, and operational issues.

The following figure depicts an example of the application view for the Generative AI Application Builder on AWS stack in Application Manager.



Generative AI Application Builder on AWS stack in Application Manager

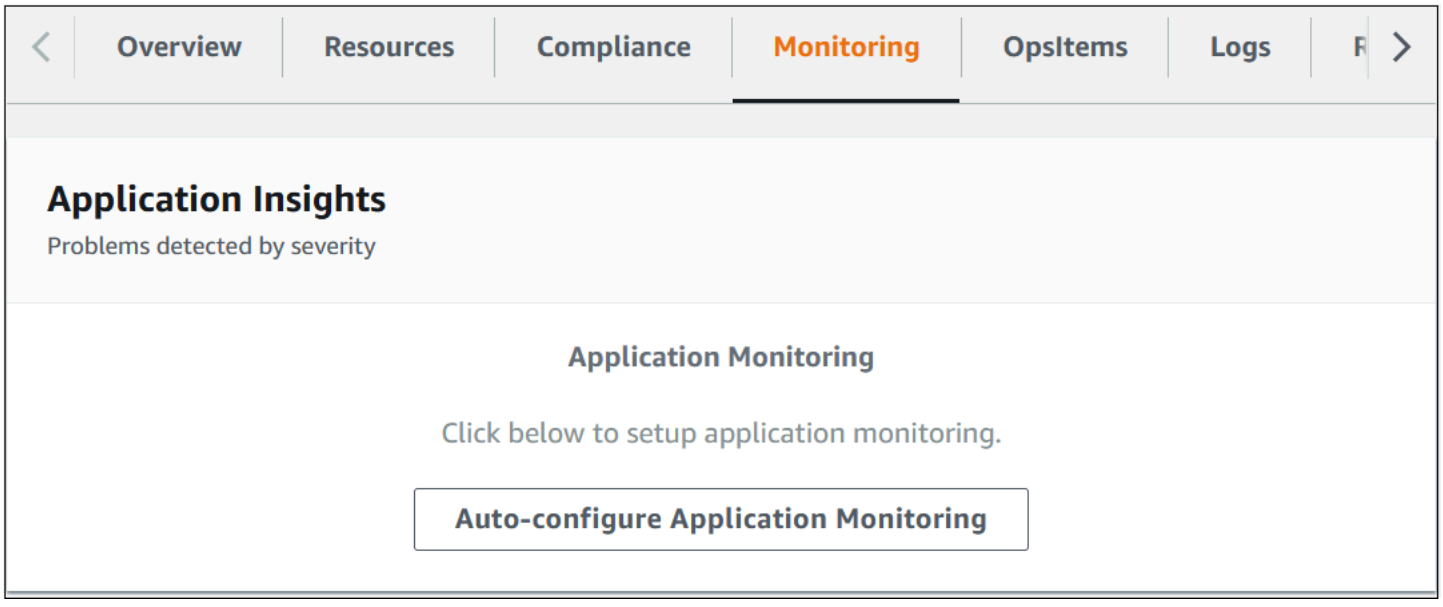
> **Note**
>
> You must activate CloudWatch Application Insights, AWS Cost Explorer, and cost allocation tags associated with this solution. They are not activated by default.

Activate CloudWatch Application Insights

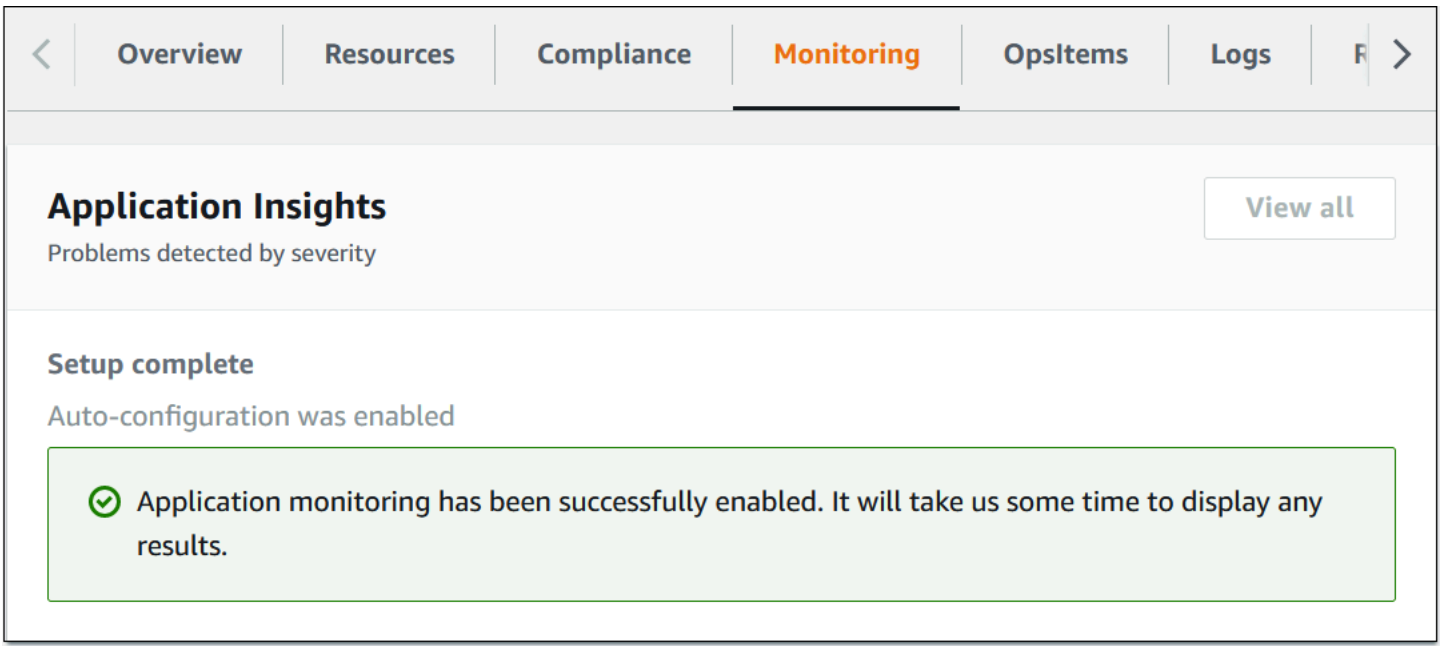
1. Sign in to the Systems Manager console.
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, choose **AppRegistry applications**.
4. In **AppRegistry applications**, search for the application name for this solution and select it.

The next time you open Application Manager, you can find the new application for your solution in the **AppRegistry application** category.

5. In the **Components** tree, choose the application stack you want to activate.
6. In the **Monitoring** tab, in **Application Insights**, select **Auto-configure Application Monitoring**.



Monitoring for your applications is now activated and the following status box appears:



Activate AWS Cost Explorer

You can see the overview of the costs associated with the application and application components within the Application Manager console through integration with AWS Cost Explorer which must be first activated. Cost Explorer helps you manage costs by providing a view of your AWS resource costs and usage over time. To activate Cost Explorer for the solution:

1. Sign in to the AWS Cost Management console.
2. In the navigation pane, select **Cost Explorer**.
3. On the **Welcome to Cost Explorer** page, choose **Launch Cost Explorer**.

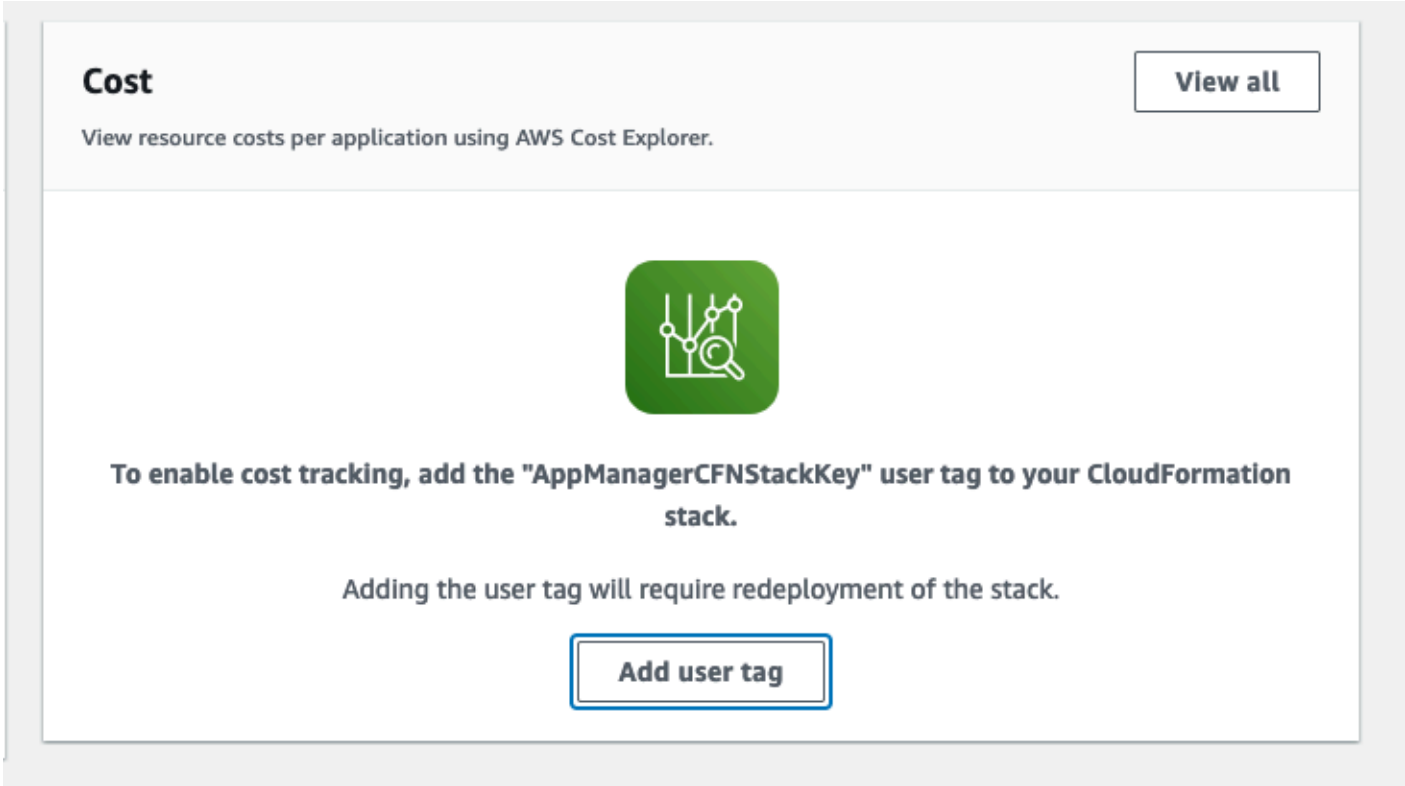
The activation process can take up to 24 hours to complete. Once activated, you can open the Cost Explorer user interface to further analyze cost data for the solution.

Confirm cost tags associated with the solution

After you activate cost allocation tags associated with the solution, you must confirm the cost allocation tags to see the costs for this solution. To confirm cost allocation tags:

1. Sign in to the Systems Manager console.
2. In the navigation pane, choose **Application Manager**.
3. In **Applications**, choose the application name for this solution and select it.

4. In the **Overview** tab, in **Cost**, select **Add user tag**.



5. On the **Add user tag** page, enter `confirm`, then select **Add user tag**.

The activation process can take up to 24 hours to complete and the tag data to appear.

Activate cost allocation tags associated with the solution

After you activate Cost Explorer, you must activate the cost allocation tags associated with this solution to see the costs for this solution. The cost allocation tags can only be activated from the management account for the organization. To activate cost allocation tags:

1. Sign in to the AWS Billing and Cost Management and Cost Management console.
2. In the navigation pane, select **Cost Allocation Tags**.
3. On the **Cost allocation tags** page, filter for the AppManagerCFNStackKey tag, then select the tag from the results shown.
4. Choose **Activate**.

The activation process can take up to 24 hours to complete and the tag data to appear.

Update the solution

If you have previously deployed the solution, follow this procedure to update the solution's CloudFormation stack to get the latest version of the solution's framework.

1. Sign in to the CloudFormation console, select your existing CloudFormation stack, and select **Update**.
2. Select **Replace current template**.
3. Under Specify template:
   a. Select **Amazon S3 URL**.
   b. Copy the link of the latest template.
   c. Paste the link in the **Amazon S3 URL** box.
   d. Verify that the correct template URL shows in the **Amazon S3 URL** text box, and choose **Next**. Choose **Next** again.
4. Under **Parameters**, review the parameters for the template and modify them as necessary. For details about the parameters, see Step 1: Launch the Deployment dashboard stack.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template will create IAM resources.
8. Choose **View change set** and verify the changes.
9. Choose **Update stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a UPDATE_COMPLETE status in approximately 10 minutes.

Note

- If you want to update an existing use case or to deploy the latest available version, you must first update the Deployment dashboard.

  Once the Deployment dashboard has been updated, you can Edit a deployment through the Dashboard to initiate a stack update of the use case.

- The solution uses CloudFront to serve the website files. This means that after updating the solution, you might not immediately see changes to the user interface until the cache expires and the latest files are pulled (default cache expiry is 24 hours). If you need to flush the cache and see the files sooner, you can create a CloudFront Invalidation.

Troubleshooting

Problem: Deploying a VPC-enabled configuration, with Create a VPC for me, fails

The Deployment dashboard stack or the use case stack fails deployment because the CloudFormation was not able to provision VPC networking resources.

Resolution

Check the quota limits for VPCs, and Elastic IPs in your account. Default limits are 5 each for Elastic IPs and VPCs per AWS account, per AWS Region.

> **Note**
>
> When the solution creates a VPC, a single VPC-enabled deployment (Deployment Dashboard or Use Case) is a 2-AZ deployment with 1 public and 1 private subnet in each AZ, each public subnet deploys 1 NAT Gateway. With 2 NAT Gateways, the deployment consumes 2 public IP addresses from the quota limit.

Some limits to be aware of (per account, per Region):

- Number of VPCs – 5
- Number of public IP addresses – 5
- Number of Gateway VPC Endpoints – 20
- Number of Inteface VPC Endpoints - 20

Problem: Use case stack can't be deleted in CloudFormation after the Deployment dashboard stack gets deleted

If the Deployment dashboard stack is deleted in CloudFormation before all of the use case stacks are deleted, the use cases can end up in a locked (unuseable) state. This is due to an IAM role created by the Deployment dashboard stack no longer exists preventing modifications to the use case stack.

Resolution

> **Warning**
>
> Ensure you clean up any manually created roles immediately after usage. These are elevated permissions that users could exploit for role elevation.

Recreate the deleted IAM role to enable the deletion of the CloudFormation stacks:

1. Open the CloudFormation console and determine the role that is associated with your locked stack.
   a. The role ARN can be found in the stack info section labled **IAM role.**
   b. The role ARN can be found in the stack info section labled **IAM role.**
2. The role name is what follows after **:role/** in the IAM role ARN (for example, **arn:aws:iam::<account-id>:role/<role-name>**)
3. Create a new role in IAM with the same name as the deleted role.
4. Select **AWS service** as the trusted entity and select **CloudFormation** from the drop down.
5. Add the necessary permissions. If you're unsure about the required permissions, you can use the AWS managed **AdministratorAccess** policy.
6. Enter the role name exactly as obtained in Step 1.
7. Return to the CloudFormation console and delete the locked stacks.
8. Once all locked stacks have been successfully deleted, return to IAM and delete any roles created in Step 2.

Problem: Use case UI does not reflect changes in settings

When use cases are updated, the UI is deployed to CloudFront. However, because CloudFront caches deployments as well as the configuration file that dictates how some settings are shown to the user, these changes might not be reflected immediately.

Resolution

The CloudFront distribution can be invalidated to force the new configuration to be propagated to frontend users.

1. Open the CloudFormation console and determine the CloudFront distribution that is associated with your use case stack.
   a. The use case stack should start with the same name you used when deploying the use case.
   b. Locate the nested stack corresponding to the UI. The nested stack name should begin with **WebAppS3UINestedStackS3UINestedStackResource**.
   c. Under the **Resources** tab, locate the resource of type **AWS::CloudFront::Distribution**, then select the physical ID. This will open the distribution in the CloudFront console.
2. Navigate to the **Invalidations** tab, then choose **Create Invalidation**, and input a path of /*. This will invalidate all paths.
3. In your own browser, delete any cookies and cached files related to the use case.

Uninstall the solution

> **Note**
>
> Deployments created through the Deployment dashboard are not intended to be managed outside of the solution. Be sure to delete and clean up any deployments from within the Deployment dashboard, before deleting the stack in CloudFormation.

You can uninstall the Generative AI Application Builder on AWS solution from the AWS Management Console or by using the AWS Command Line Interface. You must manually delete the Amazon S3 buckets, Amazon Kendra indexes, or Amazon CloudWatch logs created by this solution. AWS Solutions do not automatically delete Amazon S3 buckets, Amazon Kendra indexes, or CloudWatch logs in case you have stored data to retain.

Using the AWS Management Console

1. Sign in to the AWS CloudFormation console.
2. On the **Stacks** page, select this solution's installation stack.
3. Choose **Delete**.

Using AWS Command Line Interface

Determine whether the AWS Command Line Interface (AWS CLI) is available in your environment. For installation instructions, see What Is the AWS Command Line Interface in the *AWS CLI User Guide*. After confirming that the AWS CLI is available, run the following command.

```
$ aws cloudformation delete-stack --stack-name <installation-stack-name>
```

Manual uninstall sub-topics

Deleting the Amazon S3 buckets

This solution is configured to retain the solution-created Amazon S3 bucket if you decide to delete the AWS CloudFormation stack to prevent accidental data loss. After uninstalling the solution, you

can manually delete this Amazon S3 bucket if you do not need to retain the data. Follow these steps to delete the Amazon S3 bucket.

1. Sign in to the Amazon S3 console.
2. In the navigation pane, select **Buckets**.
3. Locate the *<stack-name>* S3 buckets.
4. Select the S3 bucket and choose **Delete**.

To delete the S3 bucket using AWS CLI, run the following command. You won't need to empty the bucket first when using the --force option.

```
$ aws s3 rb s3://<bucket-name> --force
```

Deleting the Amazon Kendra indexes

To prevent accidental data loss, this solution is configured to retain the solution-created Amazon Kendra indexes when the AWS CloudFormation stack has been deleted. After uninstalling the solution, you can manually delete the Amazon Kendra indexes that you no longer need to retain data for. Follow these steps to delete the Amazon Kendra index.

1. Sign in to the Amazon Kendra console.
2. In the navigation pane, select **Indexes**.
3. Locate and select the index you want to delete.
4. Choose **Delete** to delete the selected index.

To delete the Amazon Kendra index using AWS CLI, run the following command:

```
$ aws kendra delete-index --id<index-id>
```

Deleting the CloudWatch Logs

To prevent accidental data loss, we configured this solution to retain the CloudWatch logs if you decide to delete the CloudFormation stack. After uninstalling the solution, you can manually delete the logs if you don't need to retain the data. Follow these steps to delete the CloudWatch logs.

1. Sign in to the Amazon CloudWatch console.

2. In the navigation pane, select **Log Groups**.
3. Locate the log groups created by the solution.
4. Select one of the log groups.
5. Choose **Actions** and then choose **Delete**.

Repeat the steps until you have deleted all the solution log groups.

Use the solution

Accessing the UI

During the stack deployment process (both for the Deployment dashboard and the use cases) an email is sent to the configured email address. The email contains the user's temporary credentials they can use to sign up and access the web interface.

> **Note**
>
> The DevOps user with access to the AWS Management Console must provide the admin user with the CloudFront URL of the Deployment dashboard UI when the stack completes.

For the use cases, the admin user with access to the Deployment dashboard UI must provide the business user with the CloudFront URL of the use case UI when the deployment completes.

Once logged in, the user can interact with the solution UIs, either the Deployment dashboard in the case of admins, or the use case in the case of business users.

How to update a deployment

When on the Deployment dashboard home page (or the details page of a deployment) you can edit the configuration used by a deployment. You can only edit deployments that are in the CREATE_COMPLETE or UPDATE_COMPLETE statuses.

Except for the use case name, all other options are editable for a deployment. Just change the values you want to edit and redeploy.

> **Note**
>
> If your deployment requires an API key, you must re-enter this key every time you perform a deployment update.

Depending on the scope of edits made, the redeployment time will vary. It might take a few seconds if simple settings have changed (example, model parameters), to more than 30 minutes if

larger infrastructure related options have changed (example, request to create the Amazon Kendra index for the Text use case RAG).

Once the edit has completed successfully, the application status will report an UPDATE_COMPLETE status. At this time, you can access the deployed UI through the CloudFront URL and interact with the modified deployment.

> **Note**
>
> It may be easier to run multiple deployments side-by-side if you want to compare different settings or LLMs. Use the **Clone** feature to quickly use an existing configuration to launch a new deployment.

How to clone a deployment

When on the Deployments dashboard home page (or the details page of a deployment) you can clone the configuration used by a deployment. Cloning a deployment launches the **Deploy new use case** wizard, but with most fields pre-filled with the same values.

This is a convenience operation to help you quickly duplicate deployments with changed settings, revive a deleted deployment, or compare multiple LLMs in otherwise identical deployments.

> **Note**
>
> If your deployment requires an API key, you must re-enter this key every time you clone a deployment.

How to delete a deployment

When on the Deployments dashboard home page (or the details page of a deployment) you can delete it once you no longer need the deployment. Deleting a deployment invokes a CloudFormation stack delete operation and deprovisions the resources for the deployment.

By default, a deleted deployment still remains on the dashboard to enable the clone functionality. To completely remove a deployment from the dashboard so that it stops being tracked in the UI, choose **Permanently delete** on the delete confirmation window.

Important

Some resources are left behind during stack deletion and must be manually deleted. Refer to the Manual uninstall section for details on what resources are retained and how to clean them up.

Choosing the right LLM for your use case

Which LLM is right for your use case depends on a large set of factors specific to your needs and the type of customer experience you want to curate. This solution does not look to be prescriptive, but rather aims to give you the necessary tools to evaluate what works best for your application.

The AI-generated space is evolving rapidly, so it is incumbent on you to keep up to date on the latest models, optimization techniques, and best practices to ensure you are building the right experiences for your customers.

Note

If you're working with non-public or sensitive data, then be sure to select an LLM option using AWS services (such as Amazon Bedrock). This improves the overall security posture of your deployment by keeping data within your Region and on the AWS network.

Model parameters

LLMs often accept a wide range of parameters specific to its implementation. Model providers often provide documentation outlining the set of supported parameters and their uses.

The solution passes model parameters directly through to the underlying model so it is important to ensure parameters are set correctly. Refer to the model provider's documentation for the latest information on supported parameters.

Tips for managing model token limits

> **Note**
>
> The solution does not directly attempt to manage token limits imposed by various LLMs. Test and ensure your prompt remains within the available limits enforced by the model provider.

To help control the size of prompts, try the following:

1. Familiarize yourself with the limits imposed by the model you want to use. These values can differ dramatically across models so it's important to know what your available budget is before getting started.
2. Craft your initial prompt with that budget in mind and consider how much you want to save for any dynamic elements of the prompt. For example, user input, chat history, document excerpts, and so on.
3. Set document return limits in the Knowledge Base configuration wizard. You need to try and strike the right balance between providing the LLM with enough context to perform the task, but not so much as to exceed token limits or negatively affect latency.
4. Leave some buffer. Don't budget for the typical case, think about and experiment with the edge cases such as long input queries, large document excerpts, or long conversations.

Using Amazon SageMaker as an LLM Provider

As of v1.3.0, Amazon SageMaker is available as a model provider for Text use cases. This feature allows you to use a SageMaker inference endpoint already existing within the AWS account in the solution. Here are some ways to get started.

> **Important**
>
> The solution does not manage the lifecycle of your SageMaker endpoints. You are responsible for deleting the SageMaker endpoints once they are no longer needed to stop incurring additional charges.

Creating a SageMaker endpoint

You can use Amazon SageMaker JumpStart to quickly deploy an endpoint.

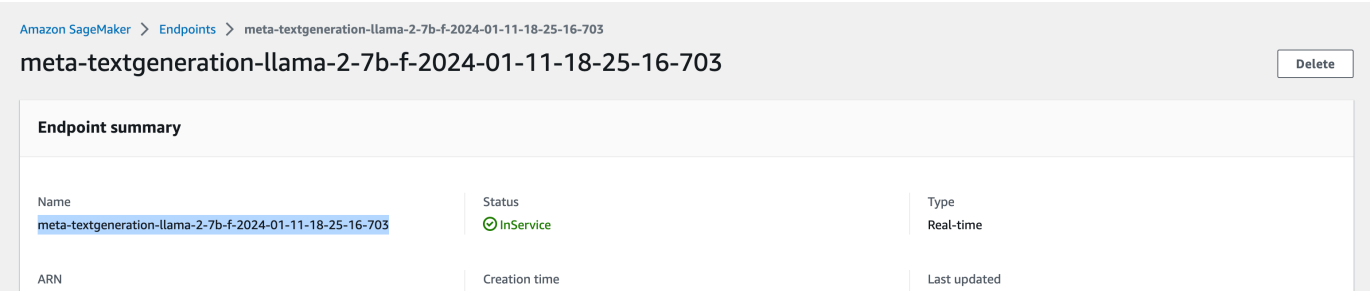
You can also use a text-generation based SageMaker endpoint and deploy using the base SageMaker service. Refer to the SageMaker JumpStart documentation for a step by step guide on how to deploy a model for inference.

> **Note**
>
> Foundation models/LLMs are typically quite large and can often require the use of large accelerated compute instances. Many of these larger instances might not be available by default in your AWS account. Refer to the default SageMaker quotas and be sure to request a quota increase before deploying to avoid common deployment failures.

Use endpoint to create a Text use case deployment

To deploy a new Text use case using a SageMaker endpoint for inference:

1. Create a new use case through the Deployment dashboard wizard and complete the forms until you reach the Models selection page.
2. On the Models page, select **SageMaker** as the model provider. This will generate a custom form requiring three key pieces of user input:
   - The name of the SageMaker endpoint you want to use. DevOps users can obtain this from the AWS console. Note that the endpoint must be in the same account and Region as the solution is deployed in.

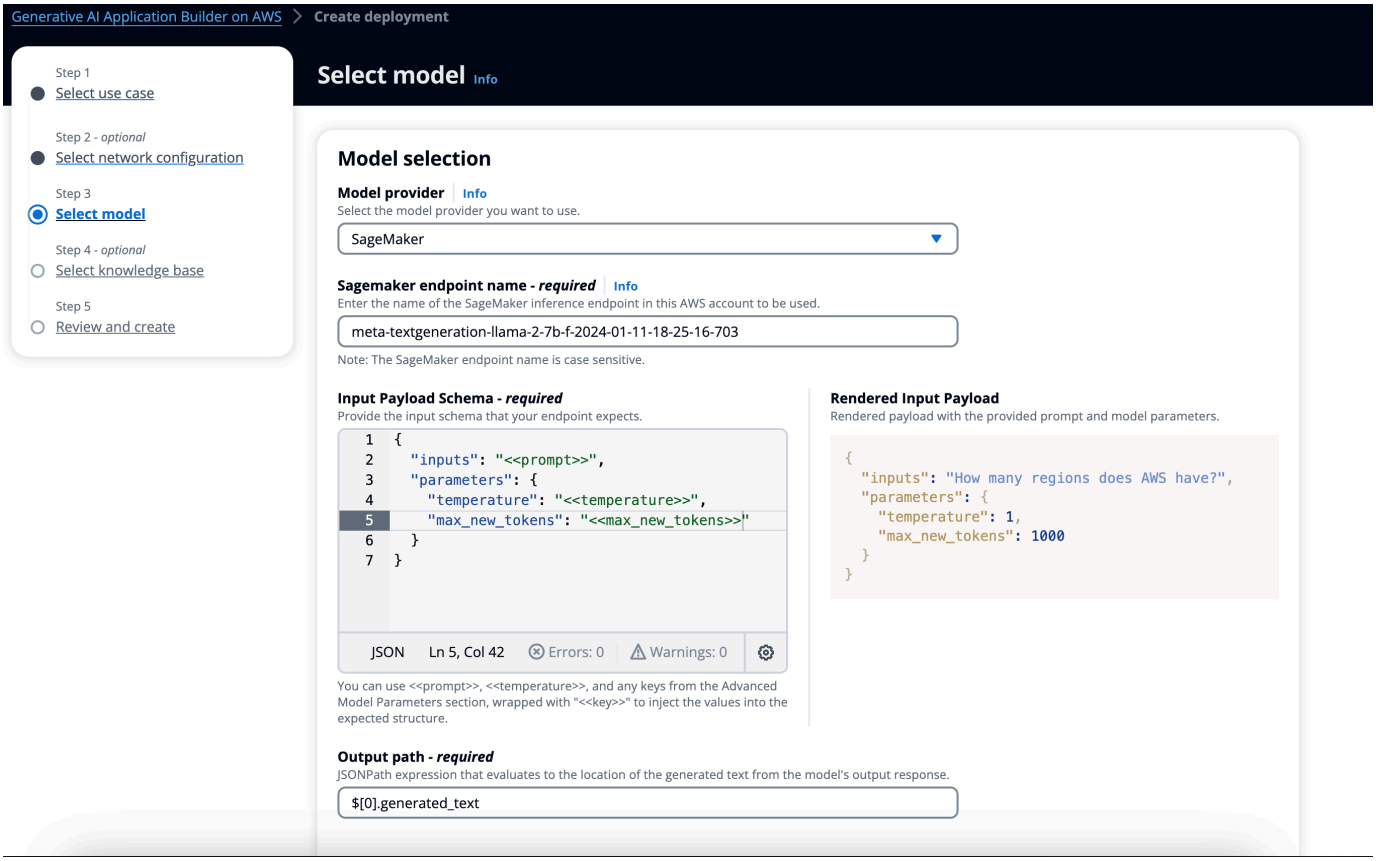


Location of the endpoint name on the AWS console

   - The schema of the input payload expected by the endpoint. To support the widest set of endpoints, admin users are required to tell the solution how their endpoint expects the input

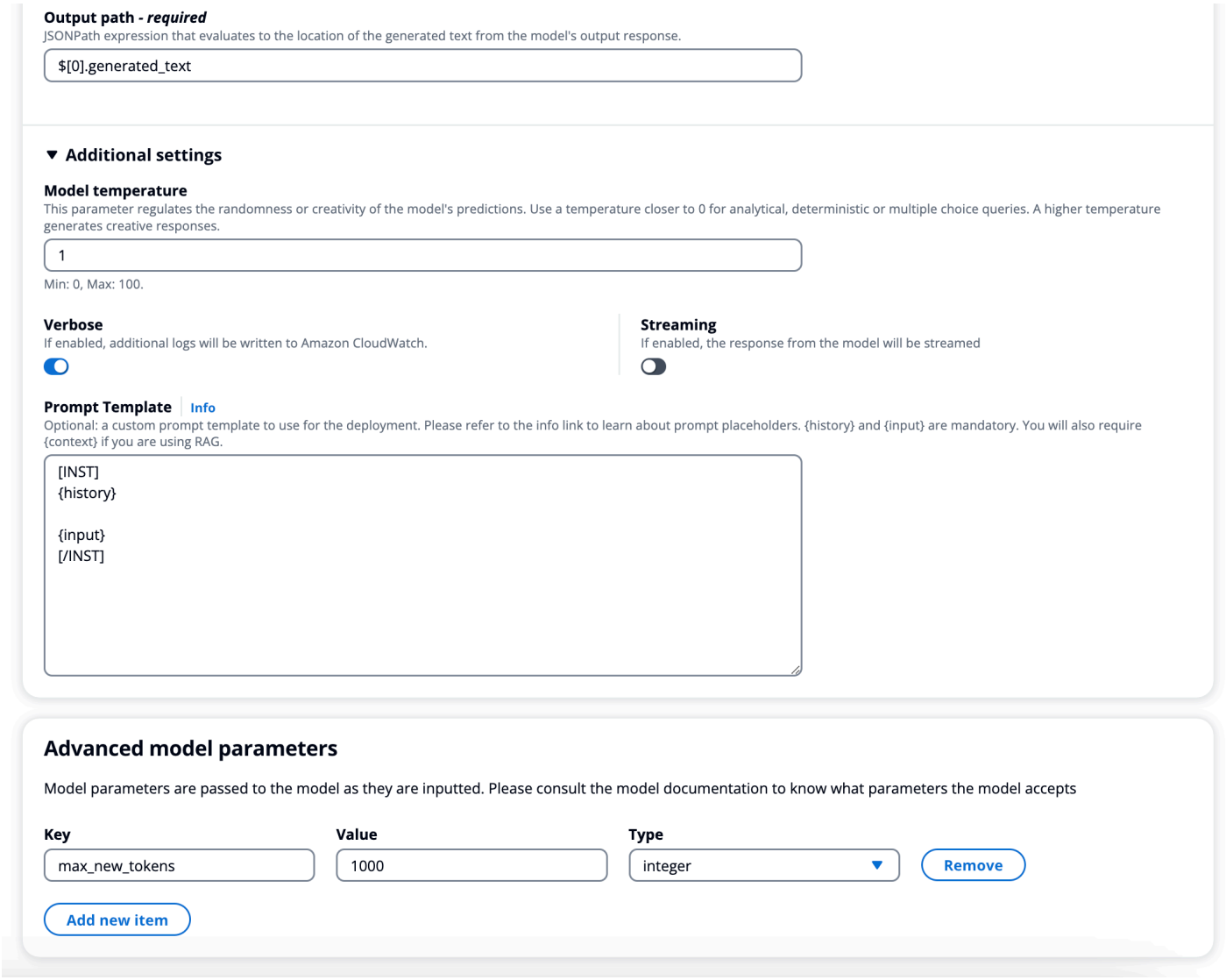
to be formatted. In the model selection wizard, provide the JSON schema for the solution to send to the endpoint. You can add placeholders to inject static and dynamic values into the request payload. The available options are:

- Mandatory placeholders: <<prompt>> will be dynamically replaced with the full input (for example, history, context, and user input as per the prompt template) to be sent to the SageMaker endpoint at runtime.
- Optional placeholders: <<temperature>>**,** as well as any parameters defined in advanced model parameters can be provided to the endpoint. Any string containing a placeholder enclosed in << and >> (for example, <<max_new_tokens>>) will be replaced by the value of the advanced model parameter of the same name.



Example input schema - setting mandatory fields, prompt and temperature, along with a custom advanced parameter, max_new_tokens. Output path must be supplied as a valid JSONPath string

3. The location of the LLMs generated string response within the output payload. This must be supplied as a JSONPath expression to indicate where the final text response shown to users is expected to be accessed from within the endpoint's return object and response.



Example of adding Advanced model parameters to use within SageMaker input schema (see Figure 2 for previous options/settings)

> **Note**
>
> SageMaker now supports hosting multiple models behind the same endpoint, and this is the default configuration when deploying an endpoint in the current version of SageMaker Studio (not Studio Classic).
>
> If your endpoint is configured in this way, you will be required to add **InferenceComponentName** to the advanced model parameters section, with a value corresponding to the name of the model you want to use.

Using Text use case

The built-in UI for the Text use case is intended to enable business users to quickly explore and experiment with the deployment created by the admin user. Configuration changes made by the business user only take effect for their session. The business user must share these changes with the admin user who can update the base deployment with those changes for all to use.

The chat UI consists of the following components:

- Chat window
- Chat input box
- Settings
- Clear conversation

Chat window

Holds different turns of the conversation. Messages starting on the right are from the business user, and messages starting on the left are from the configured LLM. A small clipboard icon exists on all LLM responses to enable easy copying of responses.

Chat input box

Pinned to the bottom of the chat window is the chat input box. This is where business users can enter their messages to be sent to the LLM. Just above the input box is the connection status. If the connection is lost (for example, due to inactivity), a new connection is automatically created the next time a chat message is sent. This request is expected to take a little longer due to the additional WebSocket connection time.

Based on the specific configuration, there might be a maximum length enforced on the input. If this limit is exceeded, users receive an alert and the message is not sent.

> **Note**
>
> If using RAG with Amazon Kendra, the Retrieve API will truncate queries to 30 token words. If expecting longer user inputs, evaluate how this might affect search performance.

Settings

To enable business users to quickly experiment with different configurations, a settings panel is available, which enables on-the-fly editing of certain deployment configuration options (example, prompt template). These changes can only be made at the start of a new session. Once a conversation is started, clearing the conversation re-enables the editing of the configuration settings.

Clear conversation

Over the course of the conversation, the solution maintains a chat history, which enables a conversational experience. This enables query disambiguation and follow-up questions. To reset a conversation and delete all chat history for this interaction, choose **Clear conversation** at the top of the chat window. Once the conversation has been cleared, a new session is created which re-enables editing of the settings.

Developer guide

This section provides the source code for the solution, an integration guide, a customization guide, and API reference.

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others.

The Generative AI Application Builder on AWS templates are generated using the AWS Cloud Development Kit (AWS CDK). See the README.md file for additional information.

Integration guide

Expanding supported LLMs

The release supports models provided by Hugging Face Hub and Anthropic, however the entire solution is designed to be easily extensible. The orchestration layer of this solution is built using LangChain, so any model provider supported by LangChain can be added to this solution. To do this, the following three components of the solution must be updated:

1. Create new `UseCaseChat` CDK stack.
   a. Clone this solution's GitHub repository, and set up your build environment by following the instructions provided in the README.md file.
   b. Copy (or create new) the `source/infrastructure/lib/anthropic-chat-stack.ts` file, paste it to the same directory, and rename it to `custom-chat-stack.ts`.
   c. Rename the class in the file to a suitable one, such as `CustomChat`.
   d. Implement the functionality to add a new layer for the model provider, as explained in step 2 below.
   e. Implement the code for `chatLlmProviderLambda` with the details of the Lambda function, that are explained in step 3 below.
2. Build and attach a Lambda layer containing the Python library of the model provider to be added.
   a. In this solution, a custom asset bundler is used to build Lambda layers, that are attached using a CDK aspects. To create a new layer for the custom model provider library, open the

LambdaAspects class in the `source/infrastructure/lib/utils/lambda-aspects.ts` file.

b. Follow the instructions on how to extend the functionality of the Lambda aspects class provided in the file. To create a new function like `getOrCreateAnthropicLayer`, also update the `LLM_LIBRARY_LAYER_TYPES` enum in the `source/infrastructure/lib/utils/constants.ts` file.

3. Extend the chat Lambda function to implement a builder, client, and handler for the new provider.

The `source/lambda/chat` contains the LangChain connections for different LLMs along with the supporting classes to build these LLMs. These supporting classes follow Builder and Factory design patterns to create the LLM.

Each handler first creates a *client*, checks the environment for required environment variables, and then calls a `get_model` method to get the LangChain LLM class. The generate method is then called to get the LLM response. LangChain currently supports streaming functionality for Anthropic. Based on streaming or non-streaming functionality, appropriate WebSocket handler is called to send the response back to the WebSocket connection using the `post_token_to_connection` method.

The `clients/builder` folder contains the classes which help build an LLM Builder using Builder pattern. It helps in creating a knowledge base (Amazon Kendra for RAG use cases), a conversation memory to maintain conversation context for LLM using Amazon DynamoDB, fetching API keys from Amazon Secrets Manager and the LLM configuration from the Amazon Parameter Store, setting the appropriate LangChain callbacks for streaming and non-streaming cases, and creating an LLM model based on the provided model configurations.

The `clients/factories` subfolder helps set the appropriate conversation memory and knowledge base, which enables easy extension to any other types you want your implementation to support.

The shared_components subfolder contains specific implementations of knowledge base and conversation memory which are instantiated inside the factories by the builder; Amazon Kendra retriever called within LangChain to retrieve documents for the RAG use cases, along with callbacks which are used by the LangChain LLM model.

To create your own implementation of a custom provider, you can:

1. Copy the `anthropic_handler.py` file and create your custom handler, say, `custom_handler.py` that creates your custom client, say `CustomProviderClient` (specified below in step b.)
2. Copy `anthropic_client.py` in the clients folder, rename it to `custom_provider_client.py` (or your specific model provider name, let's call it *CustomProvider*) and name the class within it appropriately, such as `CustomProviderClient` which inherits `LLMChatClient`.

   You can use the methods provided by `LLMChatClient` or write your own implementations to override these.

   The get_model method builds a `CustomProviderBuilder` (see step b. below), and calls the `construct_chat_model` method that constructs the chat model using builder steps. This method acts as a builder director in builder pattern.
3. Copy `clients/builders/anthropic_builder.py` and rename it to `custom_provider_builder.py` and the class within it to `CustomProviderBuilder` that inherits LLMBuilder (`llm_builder.py`). You can use the methods provided by LLMBuilder or write your own implementations to override these. The builder steps are called in sequence inside the client's `construct_chat_model` method, such as `set_knowledge_base`, `set_memory_constants` (used within LangChain memory to refer to memory, input and output, alongwith human and AI prefix, such as Human and Assistant respectively, based on the specific LLM model), `set_conversation_memory`, etc.

   `set_llm_model method` would create the actual LLM model using all of the values that are set using the methods called before it. Specifically, you can create a RAG (`CustomProviderRetrievalLLM`) or non-RAG (`CustomProviderLLM`) LLM, based on the `rag_enabled variable` as in `anthropic_builder.py` file.

   `rag_enabled variable` is set within the client initially based on the Lambda environment variable (see the constructor in `LLMChatClient` inside `clients/llm_chat_client.py`)
4. Implement your `CustomProviderLLM` or `CustomProviderRetrievalLLM` implementation in the `llm_models` subfolder based on whether you require RAG or non-RAG use case. You can copy the `llm_models/anthropic.py` file and make the necessary changes to call the Langchain model that refer to your custom provider. For example, Anthropic uses a ChatAnthropic class to create a chat model using LangChain.

The generate method generates the LLM response using the LangChain *chains* created within the `CustomProviderLLM` constructor. Based on RAG or non-RAG implementations, you could use ConversationChain or ConversationRetrievalChain.

You can also use the `get_clean_model_params` method to sanitize the model parameters as per LangChain/model requirements. For example, the solution asks the for a user input of temperature model parameter in the range `[0, 1]`, but Hugging Face models accepts it in the range `[0-100]`. So, the `denormalize_temperature` method is called on the user input that helps set the values in the correct range of `[0-100]` from `[0-1]` as required by the Hugging Face models.

5. To add your implementations of conversation memory or knowledge base, add the required implementations in the `shared_components` folder and then edit the factories and appropriate enumerations to create an instance of these classes.

   When you supply the LLM configuration, which is stored inside the parameter store, the appropriate conversation memory and knowledge base will be created for your LLM. For example, when the **ConversationMemoryType** is specified as DynamoDB, an instance of `DynamoDBChatMemory` (available inside `shared_components/memory/ddb_chat_memory.py`) is created and when the **KnowledgeBaseType** is specified as Amazon Kendra, an instance of **KendraKnowledgeBase** (available inside `shared_components/knowledge/kendra_knowledge_base.py`) is created.

6. Finally, build the program with the `npm run build` command. Once any errors are resolved, run `cdk synth` to generate the template files and all the Lambda assets.

   a. You can use the `./stage-assets.sh` script to manually stage any generated assets to the staging bucket in your account.

   b. Finally, use the `cdk deploy` command to deploy or update platform.

Customization guide

Managing Cognito user pool

When the Deployment dashboard is deployed, an Amazon Cognito user pool along with an admin user are created to provide authentication for the application. This user pool is shared across the Deployment dashboard and all use cases. The admin user created on deployment of the dashboard

is automatically granted access to all use cases deployed using the dashboard. This mechanism is provided via Amazon Cognito user pool groups.

When a use case is deployed from the dashboard, if an email is provided, a user will be created in the shared user pool, along with a user group named for the specific use case. The newly created user is then added to the group, granting the user access to the use case.

If you wish to add an additional user to a given use case, this can be achieved by creating a user in the Cognito user pool and adding them to the group(s) corresponding to the use case(s) you want the user to have access to. For a step-by-step guide, see Creating a new user in the AWS Management Console.

Similarly, if you want to create additional admin users, you must create a new user and add them to the **Admin** group in the user pool.

The user names are created by taking the portion of the provided email before the `@`, and appending the generated use case UUID (or `-admin` in the case of the admin user).

In the **Groups** tab, you can see that an **Admin** group and a group for each use case have been automatically created using the name of the use case (as provided in the wizard) and the use case UUID.

Increasing prompt and input limits

The solution adds validations and restrictions to the size of various user inputs. This is done to adhere to security best practices to help shield against various attacks which try to leverage unvalidated user inputs. These security guardrails exists across both the UI and API to ensure inputs are of a certain format and type, within valid ranges, and so on.

When it comes to the max size of the prompt template and the user text input, there might be valid use cases where you want to support much larger input sizes. The solution tries to set a default value, but it might not work well for everyone. If you want to relax these restrictions, you can build a custom version of the solution after modifying configuration files provided with the solution for the models you want to deploy.

For additional guidance on how to make these changes see the GitHub repository.

Amazon Kendra attribute filters

The solution allows you to add Amazon Kendra attribute filters in the AWS Systems Manager (SSM) Parameter Store deployed for a use case. A sample SSM Parameter configuration for the use case might look like this:

```
{
    "ConversationMemoryType": "DynamoDB",
    "KnowledgeBaseType": "Kendra",
    "KnowledgeBaseParams": {
        "NumberOfDocs": 5,
        "ReturnSourceDocs": true
    },
    "LlmParams": {
        "ModelId": "claude-v1",
        "ModelParams": {
            "max_length": {
                "Type": "integer",
                "Value": "100"
            },
            "top_p": {
                "Type": "float",
                "Value": "0.2"
            }
        },
        "PromptTemplate": "Your sample prompt",
        "Streaming": true,
        "Verbose": true,
        "Temperature": 0.2,
        "RAGEnabled": true
    }
}
```

The **KnowledgeBaseParams** property holds configuration settings needed by the knowledge base for a RAG-enabled use case. To support Amazon Kendra attribute filters, you can manually define your filter under the key **AttributeFilter** inside of **KnowledgeBaseParams**. For example, if you only want your users to query documents that are in a different language (example, Spanish), you can use a **_language_code** filter as follows:

```
{
    ...
    "KnowledgeBaseParams": {
```

```
        "NumberOfDocs": 5,
        "ReturnSourceDocs": true,
        "AttributeFilter": {
            "EqualsTo": {
                "Key": "_language_code",
                "Value": {
                    "StringValue": "es"
                }
            }
        }
    },
    ...
}
```

When an attribute filter is provided, the chat Lambda will pass it to all Amazon Kendra queries for the use case.

API reference

This section provides API references for the solution.

Deployment dashboard

REST API	HTTP method	Functionality	Authorized callers
`/deployments`	`GET`	Get all deployments.	Amazon Cognito authenticated JWT token
`/deployments`	`POST`	Creates a new use case deployment.	Amazon Cognito authenticated JWT token
`/deployments/{useCaseId}`	`GET`	Gets deployment details for a single deployment.	Amazon Cognito authenticated JWT token
`/deployments/{useCaseId}`	`PATCH`	Updates a given deployment.	Amazon Cognito authenticated JWT token

REST API	HTTP method	Functionality	Authorized callers
`/deployments/{useCaseId}`	`DELETE`	Deletes a given deployment.	Amazon Cognito authenticated JWT token
`/model-info/use-case-types`	`GET`	Gets the available use case types for the deployment	Amazon Cognito authenticated JWT token
`/model-info/{useCaseType}/providers`	`GET`	Gets the available model providers for the given use case type	Amazon Cognito authenticated JWT token
`/model-info/{useCaseType}/{providerName}`	`GET`	Gets the IDs of the models available for a given provider and use case type	Amazon Cognito authenticated JWT token
`/model-info/{useCaseType}/{providerName}/{modelId}`	`GET`	Gets the info about the given model, including default parameters.	Amazon Cognito authenticated JWT token

Note

OpenAPI and Swagger files can also be exported from API Gateway for easier integration with the API. See Export a REST API from API Gateway.

Text use case

WebSocket API	Functionality	Authorized callers
`/$connect`	Initiate WebSocket connection and authenticate user.	Amazon Cognito authenticated JWT token

WebSocket API	Functionality	Authorized callers
/sendMessage	Sends user's chat message to the WebSocket for processing with the configured LLM experience.	Amazon Cognito authenticated JWT token
/$disconnect	Endpoint called when a WebSocket connection has been disconnected.	Amazon Cognito authenticated JWT token
/$default	Default endpoint called when a non-JSON request is made. Defaults back to the same backing Lambda as /sendMessage.	Amazon Cognito authenticated JWT token

Supplemental topics

Deploying the application without the UI

The solution provides an option to deploy only the API endpoints with the workflow configuration and integrate it with another UI application. Check the **Mappings** section in the CloudFormation template to enable or disable features for deployment.

Supported LLM providers

The solution can integrate with the following LLM providers:

1. Amazon Bedrock
   - Documentation: https://aws.amazon.com/bedrock/
   - Supported models:
     - Amazon
       - Titan Text Lite
       - Titan Text Express
     - AI21 Labs
       - Jurassic-2 Mid
       - Jurassic-2 Ultra
     - Anthropic
       - Claude Instant v1
       - Claude v2
       - Claude v2.1
       - Claude v3 (Haiku, Sonnet, and Opus)
     - Cohere
       - Command Lite
       - Command
     - Meta
       - Llama2 13B Chat
       - Llama2 70B Chat

    - Mistral AI
        - Mistral 7B Instruct
        - Mistral 8x7B Instruct

2. Anthropic
    - Documentation: https://docs.anthropic.com/claude/docs
    - Supported models:
        - `claude-instant-1` (default)
        - `claude-2`
3. Hugging Face
    - Documentation: https://huggingface.co/docs/api-inference/index
    - Supported models:
        - `google/flan-t5-xxl` (default)
        - `google/flan-t5-xl`
        - `google/flan-t5-large`
        - `google/flan-t5-base`
        - `google/flan-t5-small`
4. Hugging Face - Inference Endpoints
    - Documentation: https://huggingface.co/docs/inference-endpoints/index
    - Supported models:
        - `tiiuae/falcon-40b-instruct` (default)
        - `tiiuae/falcon-7b-instruct`
        - `tiiuae/falcon-40b`
        - `tiiuae/falcon-7b`
        - `google/flan-t5-xxl`
        - `google/flan-t5-xl`
        - `google/flan-t5-large`
        - `google/flan-t5-base`
        - `google/flan-t5-small`

For the latest model parameters, best practices, and recommended uses, refer to the documentation from the model providers.

> **Note**
>
> This solution provides two Hugging Face-based model providers coded as *HuggingFace* and *HugginFace – Inference Endpoint*. Selecting HuggingFace uses the free hosted inference APIs provided by HuggingFace allowing you to quickly explore some of the smaller models on the hub. Hugging Face recommends the use of Inference Endpoints for production deployments and are required when using some of the larger models.

Viewing operational metrics for a deployment

The Deployment dashboard and use case stacks each come with their own CloudWatch dashboard tracking various operational metrics of the solution. You can use these CloudWatch dashboards to help compare different deployments. To access the dashboards:

1. Navigate to the CloudWatch console.
2. Search for the pre-built dashboards either by looking up the stack name, or UUID.

For example, the Text use case comes with graphs tracking the number of WebSocket connections, the number of user sign ins and sign ups, the amount of time the LLM took to process a completion, and so on. Customers can use these graphs to compare various *quantitative* metrics of a deployment.

> **Note**
>
> It is difficult to compare the *qualitative* results of various models applied to different use cases. Use the Clone feature to spin up multiple deployments quickly so that you can compare the outputs side by side.

Deploying the Text use case stack separately

Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 10-30 minutes

1. Sign in to the AWS Management Console and select the button to launch the AWS CloudFormation template you want to deploy.

BedrockChat.template	Launch solution
SageMakerChat.template	Launch solution
AnthropicChat.template	Launch solution
HuggingFaceChat.template	Launch solution

2. The template launches in the US East (N. Virginia) Region by default. To launch the solution in a different AWS Region, use the Region selector in the console navigation bar.

> **Note**
>
> This solution uses Amazon Kendra and Amazon Bedrock, which are not currently available in all AWS Regions. If using these features, you must launch this solution in an AWS Region where these services are available. For the most current availability by Region, see the AWS Regional Services List.

3. On the **Create stack** page, verify that the correct template URL is in the **Amazon S3 URL** text box and choose **Next**.
4. On the **Specify stack details** page, assign a name to your solution stack. For information about naming character limitations, see IAM and STS Limits in the *AWS Identity and Access Management User Guide*.

5. Under **Parameters**, review the parameters for this solution template and modify them as necessary. This solution uses the following default values.

Parameter	Default	Description
VpcEnabled	No	Should the stacks' resources be deployed within a VPC?
CreateNewVpc	No	Select Yes, if you would like to create a new VPC. **Note** Only used if **VpcEnabled** is set to Yes.
IPAMPoolId	<Optional input>	If you would like to assign the CIDR range using AWS VPC IP Address Manager, provide the IPAM pool Id to use. **Note** Only used if **VpcEnabled** is set to Yes and **CreateNew Vpc** is Yes.

Parameter	Default	Description
ExistingVpcId	<Optional input>	VPC ID of an existing VPC to be used for the use case. **Note** Input required if **VpcEnabled** is set to Yes and **CreateNew Vpc** is No.
ExistingPrivateSubnetIds	<Optional input>	Comma separated list of subnet IDs of existing private subnets to be used to deploy the AWS Lambda function **Note** Input required if **VpcEnabled** is set to Yes and **CreateNew Vpc** is No.
ExistingSecurityGroupIds	<Optional input>	Comma-separated list of security groups of the existing VPC to be used for configuring Lambda functions. **Note** Input required if **VpcEnabled** is set to Yes and **CreateNew Vpc** is No.

Parameter	Default	Description
ChatConfigSSMParameterName	*<Requires input>*	Name of the SSM parameter containing configurations required by the chat provider Lambda at runtime. Parameter value must be a JSON string. The SSM parameter is populated by the Deployment dashboard if in use. For standalone deployments of this use case, manual configuration is required.
ConsentToDataLeavingAWS	No	A complete sentence describing the parameter. Specify the supported range or character limitations, as needed. **Note** This only applies to Anthropic and Hugging Face templates.
DefaultUserEmail	placeholder@example.com	Email of the default user for this use case. An Amazon Cognito user for this email is created to access the use case.
DeployUI	Yes	Whether or not the solution will deploy the frontend UI for this deployment.

Parameter	Default	Description
ExistingCognitoGroupPolicyTableName	<Optional input>	Name of the DynamoDB table containing user group policies, used by the custom authorizer on this use case's API. Typically provided when deploying from the Deployment dashboard, but can be omitted when deploying this use case stack standalone.
ExistingCognitoUserPoolId	<Optional input>	**UserPoolId** of an existing Amazon Cognito user pool which this use case will be authenticated with. Typically provided when deploying from the Deployment dashboard, but can be omitted when deploying this use case stack standalone.
ExistingKendraIndexId	<Optional input>	Index ID of an existing Amazon Kendra index to be used for the use case. If not provided, a new index will be created for you. **Note** Only relevant if **RAGEnabled** is `true`.

Parameter	Default	Description
NewKendraIndexEdition	<Optional input>	The edition of Amazon Kendra to use for the new Amazon Kendra index to be created for this use case. Only applies if **ExistingKendraIndexId** is not supplied, see Amazon Kendra Editions. **Note** Only relevant if **RAGEnabled** is `true`.
NewKendraIndexName	<Optional input>	Name for the new Amazon Kendra index to be created for this use case. Only applies if **ExistingKendraIndexId** is not supplied. **Note** Only relevant if **RAGEnabled** is `true`.

Parameter	Default	Description
NewKendraQueryCapacityUnits	0	Additional query capacity units for the new Amazon Kendra index to be created for this use case. Only applies if **ExistingKendraIndexId** is not supplied, see CapacityUnitsConfiguration. **Note** Only relevant if **RAGEnabled** is set to `true`.
NewKendraStorageCapacityUnits	0	Additional storage capacity units for the new Amazon Kendra index to be created for this use case. Only applies if **ExistingKendraIndexId** is not supplied, see CapacityUnitsConfiguration. **Note** Only relevant if **RAGEnabled** is set to `true`.

Parameter	Default	Description
ProviderApiKeySecret	<*Requires input*>	Name of secret in Secrets Manager holding the API key used by LangChain to call the model. Must be named **<UseCaseUUID>/api-key.** **Note** This only applies to Anthropic and Hugging Face templates.
RAGEnabled	`true`	If set to `true`, the deployed use case stack uses the provided/created Amazon Kendra index to provide RAG functionality. If set to `false`, the user interacts directly with the LLM.
UseCaseUUID	*<Requires input>*	UUID to identify this deployed use case within an application. Enter UUID (8 characters long). If you are editing the stack, do not modify the value (retain the value used during creating the stack). A different UUID when editing the stack will result in new AWS resource created, and deleting the old ones.

6. Choose **Next**.

7. On the **Configure stack options** page, choose **Next**.
8. On the **Review** page, review and confirm the settings. Select the box acknowledging that the template will create AWS Identity and Access Management (IAM) resources.
9. Choose **Create stack** to deploy the stack.

You can view the status of the stack in the AWS CloudFormation console in the **Status** column. You should receive a CREATE_COMPLETE status in approximately 10-30 minutes.

Chat Config SSM Parameter

When deploying a use case, **ChatConfigSSMParameterName** is a required parameter which is normally populated by the Deployment dashboard. When performing a standalone deployment, you must create an SSM parameter containing the configuration for the use case as a JSON string, and provide the name of that parameter.

To create a suitable config parameter for a standalone deployment, create a required use case from the Deployment dashboard, and copy the parameters value. Example configuration:

```
{
    "UseCaseName": "sampleUseCase",
    "ConversationMemoryType": "DynamoDB",
    "KnowledgeBaseType": "Kendra",
    "KnowledgeBaseParams": {
        "NumberOfDocs": 2,
        "ReturnSourceDocs": false
    },
    "LlmParams": {
        "ModelProvider": "SageMaker",
        "ModelId": "default",
        "InferenceEndpoint": "endpoint",
        "ModelParams": {
            "param1": {
                "Value": "some value",
                "Type": "string"
            }
        },
        "PromptTemplate": "{history}\n\n{context}\n\n{input}",
        "Streaming": true,
        "Verbose": false,
        "Temperature": 0.1,
```

```
        "RAGEnabled": true,
        "ModelInputPayloadSchema": {
            "input": "<<prompt>>",
            "parameters": {
                "temperature": "<<temperature>>",
                "someParam": "<<param1>>"
            }
        },
        "ModelOutputJSONPath": "$output"
    }
}
```

Reference

This section includes information about an optional feature for collecting unique metrics for this solution, pointers to related resources, and a list of builders who contributed to this solution.

Anonymized data collection

This solution includes an option to send anonymized operational metrics to AWS. We use this data to better understand how customers use this solution and related services and products. When invoked, the following information is collected and sent to AWS:

- **Solution ID** - The AWS solution identifier
- **Unique ID (UUID)** - Randomly generated, unique identifier for each Generative AI Application Builder on AWS deployment
- **Timestamp** - Data-collection timestamp
- **New Amazon Kendra Index Created** - Whether or not a new Amazon Kendra index has been created
- **Amazon Kendra Edition** - The Amazon Kendra edition selected for creation
- **RAG Enabled** - Whether or not the RAG functionality is being used
- **Model Provider Name** - Name of the model provider being used
- **Model Id** - Name of the model ID used for the deployment
- **Model Params** - The set of model parameters used for the deployment
- **Streaming** - Whether or not streaming functionality is being used
- **Verbose** - Whether or not verbose logging is being used
- **Temperature** - The temperature setting used for the deployment
- **Usage Counts** - Various metric counts collected from the solution's custom CloudWatch dashboard which provides the application's usage analytics. Example stats include WebSocket error counts, Kendra latency, and so on.

AWS owns the data gathered through this survey. Data collection is subject to the AWS Privacy Policy. To opt out of this feature, complete the following steps before launching the AWS CloudFormation template.

1. Download the AWS CloudFormation template to your local hard drive.

2. Open the AWS CloudFormation template with a text editor.
3. Modify the AWS CloudFormation template mapping section from:

```
AnonymousData:
    SendAnonymousData:
      Data: Yes
```

   to:

```
AnonymousData:
    SendAnonymousData:
      Data: No
```

4. Sign in to the AWS CloudFormation console.
5. Choose **Create stack**.
6. On the **Create stack** page, specify template section, select **Upload a template** file.
7. Under **Upload a template file**, choose **Choose file** and select the edited template from your local drive.
8. Choose **Next** and follow the steps in Launch the stack in the Deploy the solution section of this guide.

Contributors

- Tarek Abdunabi
- Mukit Bin Momin
- Michael Connor
- Johny Duval
- Nihit Kasabwala
- Ibrahim Mohamed
- James Nixon
- Omar Radwan Mohsen
- Jae Shim
- Reet Takkar
- Dimitri Tchikatilov

- Jason Wreath

Revisions

Date	Change
October 2023	Initial release
October 2023	Release version 1.0.1: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG .md file in the GitHub repository.
November 2023	Release version 1.1.0: Updated built-in UI for the Text use case to display LLM responses in markdown format; updated prompt template and text input character limits; resolved various bugs and patched outstanding security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.
November 2023	Release version 1.1.1: Patched security vulnerabilities and removed support for Node.js 16.X. For more information, refer to the CHANGELOG.md file in the GitHub repository.
November 2023	Documentation update: Added Confirm cost tags associated with the solution to the Monitoring the solution with AWS Service Catalog AppRegistry section.
December 2023	Release version 1.2.0: Expanded supported models to include all text generation models available on Amazon Bedrock; increased maximum number of Kendra documents from 5 to 100; various bug fixes; and patched security vulnerabilities. For more information,

Date	Change
	refer to the CHANGELOG.md file in the GitHub repository.
December 2023	Release version 1.2.1: Fixed unit tests failure due to a change in the underlying Anthropic library. For more information, refer to the CHANGELOG.md file in the GitHub repository.
January 2024	Release version 1.2.2: Pinned and updated library versions. For more information, refer to the CHANGELOG.md file in the GitHub repository.
February 2024	Release version 1.2.3: Fixed an issue that caused create, update, and delete use cases from the deployment dashboard to fail. For more information, refer to the CHANGELOG.md file in the GitHub repository.
February 2024	Release version 1.3.0: Added support for deploying the Deployment dashboard and use cases in an Amazon VPC; added support for Amazon SageMaker as a model provider through SageMaker inference endpoint. For Text use cases, added LLM response explainability in the chat UI for RAG deployments by surfacing the document excerpts used. For more information, refer to the CHANGELOG.md file in the GitHub repository.
February 2024	Release version 1.3.1: Fixed an issue where user provisioning for use cases fails because of insufficient IAM permissions. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
March 2024	Release version 1.3.2: Patched security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.
March 2024	Release version 1.3.3: Patched security vulnerabilities; fixed an issue where chat service was failing when using Cohere and Meta models with RAG enabled. For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2024	Release version 1.4.0: Expanded supported models to include all text generation models available on Amazon Bedrock; increased prompt/input limit default values; and patched security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Notices

Customers are responsible for making their own independent assessment of the information in this document. This document: (a) is for informational purposes only, (b) represents AWS current product offerings and practices, which are subject to change without notice, and (c) does not create any commitments or assurances from AWS and its affiliates, suppliers or licensors. AWS products or services are provided "as is" without warranties, representations, or conditions of any kind, whether express or implied. AWS responsibilities and liabilities to its customers are controlled by AWS agreements, and this document is not part of, nor does it modify, any agreement between AWS and its customers.

Generative AI Application Builder on AWS is licensed under the terms of the of the Apache License Version 2.0 available at The Apache Software Foundation.

Important

Generative AI Application Builder on AWS allows you to build and deploy generative artificial intelligence applications on AWS by engaging the generative AI model of your choice, including third-party generative AI models that you can choose to use that AWS does not own or otherwise have any control over ("Third-Party Generative AI Models"). Your use of the Third-Party Generative AI Models is governed by the terms provided to you by the Third-Party Generative AI Model providers when you acquired your license to use them (for example, their terms of service, license agreement, acceptable use policy, and privacy policy).

You are responsible for ensuring that your use of the Third-Party Generative AI Models comply with the terms governing them, and any laws, rules, regulations, policies, or standards that apply to you.

You are also responsible for making your own independent assessment of the Third-Party Generative AI Models that you use, including their outputs and how Third-Party Generative AI Model providers use any data that might be transmitted to them based on your deployment configuration. AWS does not make any representations, warranties, or guarantees regarding the Third-Party Generative AI Models, which are "Third-Party Content" under your agreement with AWS. Generative AI Application Builder on AWS is offered to you as "AWS Content" under your agreement with AWS.