AWS QnABot

Implementation guide

AWS QnABot: Implementation guide

Table of Contents

Create a custom question and answer chatbot

Publication date: September 2021 (last update (p. 53): October 2021)

AWS QnABot is a multi-channel, multi-language conversational interface (chatbot) that responds to your customer's questions, answers, and feedback. The solution's content management environment, and contact center integration wizard allow you to set up and customize an environment that provides the following benefits:

- Enhance your customer's experience by providing personalized tutorials and question and answer support with intelligent multi-part interaction
- Reduce call center wait times by automating customer support workflows
- Implement the latest machine learning technology to create engaging, human-like interactions for chatbots

This guide provides detailed instructions for the following tasks:

This implementation guide describes architectural considerations and configuration steps for deploying the AWS QnABot solution the Amazon Web Services (AWS) Cloud. It includes links to an AWS CloudFormation template that launches and configures the AWS services required to deploy this solution using AWS best practices for security and availability.

The guide is intended for IT architects, developers, DevOps, data analysts, and marketing technology professionals who have practical experience architecting in the AWS Cloud.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of September 2021, the cost for running this solution with the default settings in the Default AWS Region (US East (N. Virginia) is approximately **$552.33 per month**.

Example cost table

The following table provides an example monthly cost breakdown for deploying this solution with the default parameters in the US East (N. Virginia) Region (excludes free tier).

Table 1: 10 monthly active users, with 50,000 API requests per month with Amazon Lex

AWS Service	Dimensions	Cost
Amazon API Gateway	1,000,000 RESTAPI per month	$3.50
Amazon Cognito	1,000 active users per month without the advanced security feature	$0.00
Amazon CloudFront	1,000,000 requests — 1 request per download with the static bucket size of 100KB and approximately 20GB data transfer	$8.50
Amazon S3	100GB data transfer + 1,000,000 requests — 100 records x 100KB from Kinesis	$3.27
AWS Lambda	2,000,000 requests with 200ms duration	$1.23
Systems Manager Parameter Store	2,000,000 requests with 10 standard parameters	$0.00
Amazon Lex	100,000 text requests per month	$75.00
Kinesis Data Firehose	100,000 records per month with 100KB per record	$0.28
Amazon DynamoDB	1GB storage + 1 read and 1 write per second + 20 hours peak read/write per month	$11.41
Amazon Polly	10,000 requests + 50 characters per request	$4.00
Amazon Translate	100,000 requests + 50 characters per request (OPTIONAL for non-English)	$75.00

AWS Service	Dimensions	Cost
Amazon Comprehend	100,000 requests + 50 characters per request	$5.00
Amazon OpenSearch Service	M6g.large instance running all hours in a month for 4 nodes	$368.64
		TOTAL: $552.33 / month

Note
Amazon Kendra and Amazon Connect are not part of the default solution implementation, but the solution does provide the capability to integrate with them. Because the solution does not create resources for Amazon Kendra or Amazon Connect automatically, they are not included in the example cost table. We recommend reviewing Amazon Kendra pricing and Amazon Connect pricing, if you choose to integrate these services.

We recommend creating a budget through AWS Cost Explorer to help manage costs. Prices are subject to change. For full details, refer to the pricing webpage for each AWS service used in this solution.

Architecture overview

Deploying this solution with the default parameters builds the following environment in the AWS Cloud.

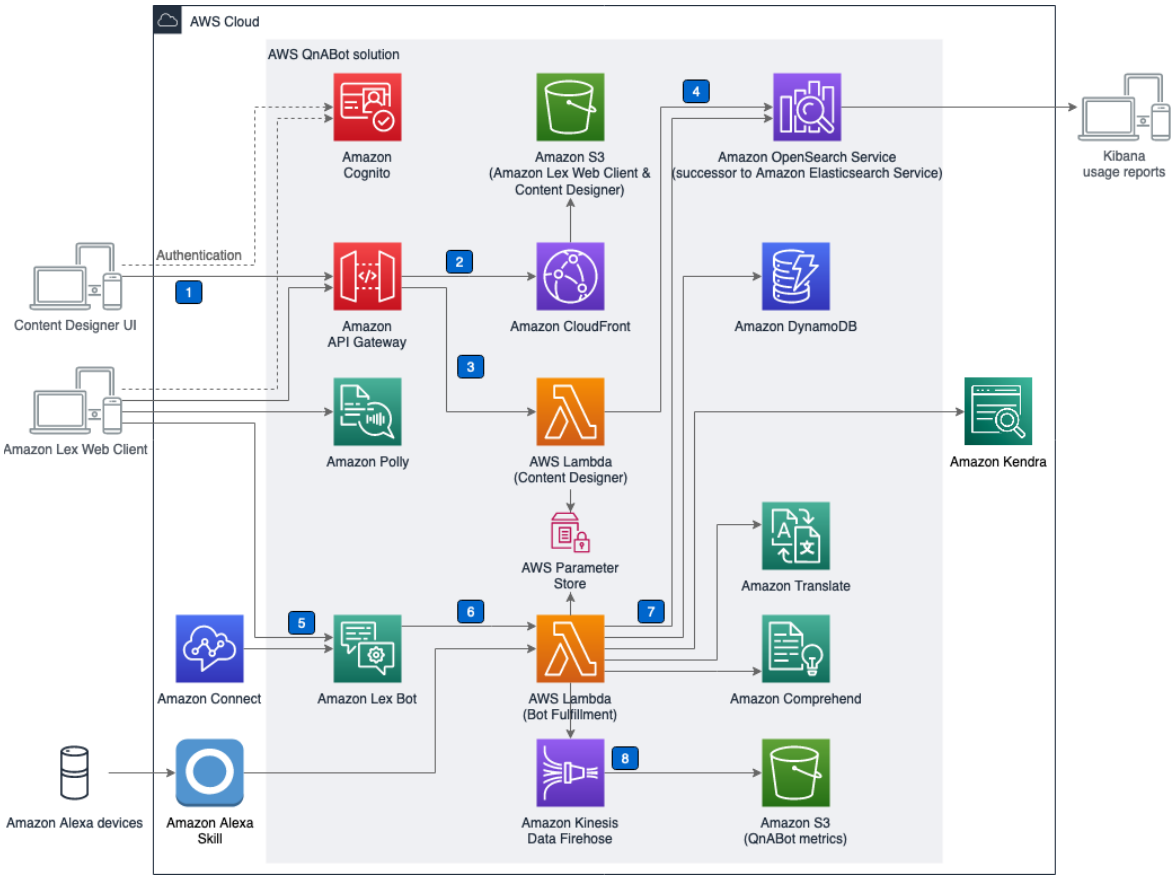

Figure 1: AWS QnABot architecture on AWS

The AWS CloudFormation template deploys the following workflows and services:

1. The admin deploys the solution into their AWS account, opens the Content Designer UI, and uses Amazon Cognito to authenticate.
2. After authentication, Amazon CloudFront and Amazon S3 deliver the contents of the Content Designer UI.
3. The admin configures questions and answers in the Content Designer and the UI sends requests to Amazon API Gateway to save the questions and answers.
4. The Content Designer AWS Lambda function saves the input in Amazon OpenSearch Service (successor to Amazon ElasticSearch Service) in a questions bank index.
5. Users of the chatbot interact with Amazon Lex via the web client UI or Amazon Connect.
6. Amazon Lex forwards requests to the AWS Lambda (`Bot Fulfillment`) function. (Users can also send requests to this Lambda function via Amazon Alexa devices).
7. The `Bot Fulfillment` function takes the users input and uses Amazon Comprehend and Amazon Translate (if necessary) to translate non-English requests to English and then looks up the answer in in Amazon OpenSearch Service. If Amazon Kendra index is configured and provided at the time of deployment, the Bot Fulfillment function also sends a request to the Amazon Kendra index.

8. User interactions with Bot Fulfillment functions generate logs and metrics data, which is sent to Amazon Kinesis Data Firehose then to Amazon S3 for later data analysis.

Solution components

During deployment, resources such as Amazon Lex, Amazon OpenSearch Service, Amazon API Gateway, and AWS Lambda are provisioned and set up in your AWS account.

Three key AWS services are at the core of the solution:

- **Amazon Lex** is a service for building conversational interfaces into any application using voice and text. Amazon Lex provides the advanced deep learning functionalities of automatic speech recognition (ASR) for converting speech to text, and natural language understanding (NLU) to recognize the intent of the text to allow you to build applications with highly engaging user experiences and lifelike conversational interactions.
- **AWS Lambda** provides logic for chatbot interactions and provides extension capabilities for Amazon Translate before and after interaction with Lex.
- **Amazon OpenSearch Service** is an open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and clickstream analysis. Amazon OpenSearch Service is a managed service that helps deploy, operate, and scale Amazon OpenSearch Service clusters in the AWS Cloud. The service offers open-source Amazon OpenSearch Service APIs, managed Kibana, and integrations with Log stash and other AWS services, enabling you to securely ingest data from any source and search, analyze, and visualize it in real time.

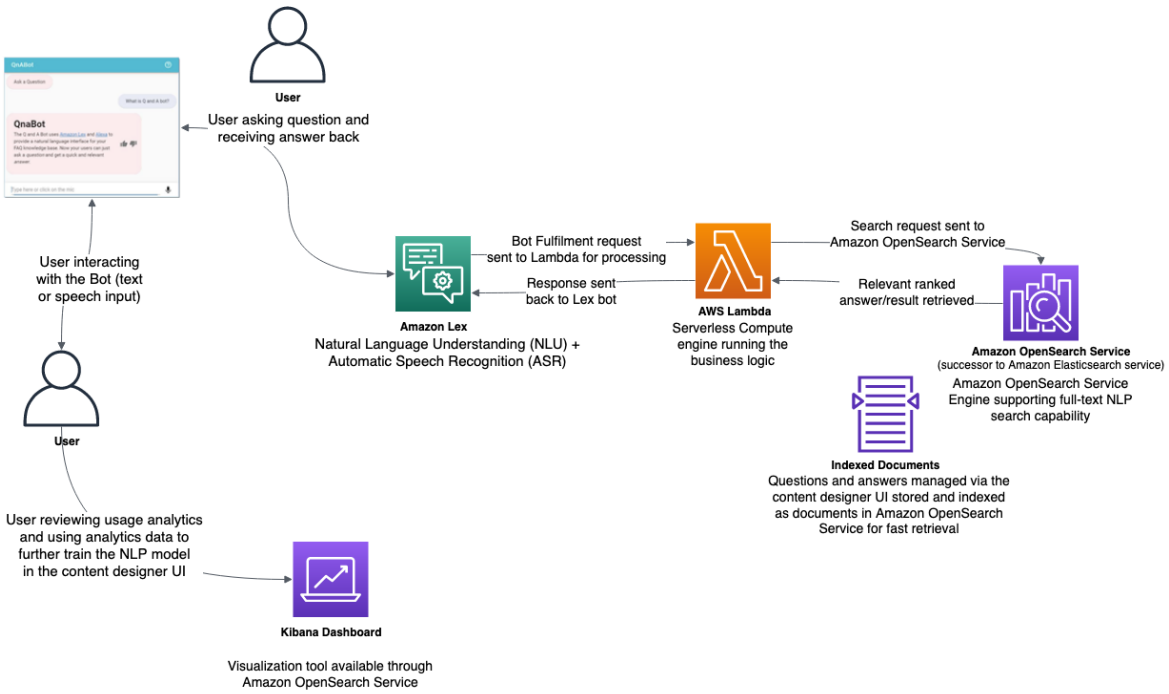

Figure 2 illustrates how Amazon Lex and Amazon OpenSearch Service help power the AWS QnABot solution.

Figure 2: Solution architecture and data flow

Asking AWS QnABot questions initiates the following processes:

1. The question gets processed and transcribed by Amazon Lex using a Natural Language Understanding (NLU) and Processing (NLP) engine.
   - The solution initially trains the NLP to match a wide variety of possible questions and statements, so that the Amazon Lex chatbot can accept almost any question a user asks. The Amazon Lex interaction model is set up with the following:
     - **Intents**: An intent represents an action that fulfills a user's spoken request. Intents can optionally have arguments called *slots*. The solution uses *slots* to capture user input and fulfill the intent via Lambda function.
     - **Sample utterances**: A set of likely spoken phrases mapped to the intents. This should include as many representative phrases as possible. The sample utterances specify the words and phrases users can say to invoke your intents. The solution updates the **Sample utterances** with the various questions to train the chatbot to understand different end user's input.
2. This question is then sent to Amazon OpenSearch Service. The solution attempts to match an end user's request to the list of questions and answers stored in Amazon OpenSearch Service.
   - The AWS QnABot uses full-text search to find the most relevant ranked document from the searchable index. Relevancy ranking is based on a few properties:
     - **Count:** How many search terms appear in a document
     - **Frequency:** How often the specified keywords occur in a given document
     - **Importance:** How rare or new the specified keywords are and how closely the keywords occur together in a phrase
   - The closer the alignment between a question associated with an item and a question asked by the user, the greater the probability that the solution will choose that item as the most relevant answer. Noise words such as article and prepositions in sentence construction have lower weighting than unique keywords.
   - The keyword filter feature helps the solution to be more accurate when answering questions, and to admit more readily when it doesn't know the answer. The keyword filter feature works by using Amazon Comprehend to determine the *part of speech* that applies to each word you say to AWS QnABot. By default, nouns (including proper nouns), verbs, and interjections are used as *keywords*. Any answer returned by AWS QnABot must have questions that match these keywords, using the following (default) rule:
     - If there are 1 or 2 keywords, then all keywords must match.
     - If there are 3 or more keywords, then 75% of the keywords must match.
     - If AWS QnABot can't find any answers that match these keyword filter rules, then it will admit that it doesn't know the answer rather than guessing an answer that doesn't match the keywords. AWS QnABot logs every question that it can't answer so you can see them in the included Kibana Dashboard.
   - The **Bot fulfillment** Lambda function generates an Amazon OpenSearch Service query containing the transcribed question. The query attempts to find the best match from all the questions and answers you've previously provided, filtering items to apply the keyword filters and using Amazon OpenSearch Service relevance scoring to rank the results. Scoring is based on 1) matching the words in the end user's question against the unique set of words used in the stored questions (*quniqueterms*), 2) matching the phrasing of the user's question to the text of stored questions (nested field: *questions.q*), and 3) matching the topic value assigned to the previous answer (if any) to increase the overall relevance score when topic value (*field t*) matches. The following example code shows the query:

```
"query": {
    "bool": {
        "filter": {
            "match": {
                "quniqueterms": {
                    "query": "<LIST_OF_IDENTIFIED_KEYWORDS>",
                    "minimum_should_match": "<ES_MINMUM_SHOULD_MATCH SETTING>",
                    "zero_terms_query": "all"
                }
            }
        },
        "should": [
            {
                "match": {
                    "quniqueterms": {
                        "query": "<USER QUESTION>",
                        "boost": 2
                    }
                }
            },
            {
                "nested": {
                    "score_mode": "max",
                    "boost": "<ES_PHRASE_BOOST SETTING>",
                    "path": "questions",
                    "query": {
                        "match_phrase": {
                            "questions.q": "<USER QUESTION>"
                        }
                    }
                }
            },
            {
                "match": {
                    "t": "<PREVIOUS_TOPIC>"
                }
            }
        ]
    }
}
```

Figure 3: Example Amazon OpenSearch Service query

Amazon Lex web client

Amazon Lex allows conversational interfaces to be integrated into applications like the Amazon Lex web client. An Amazon Lex chatbot uses *intents* to encapsulate the purpose of an interaction, and *slots* to capture elements of information from the interaction. Since AWS QnABot has a single purpose, to answer a user's question, it defines just one intent. This intent has a single slot which is trained to capture the text of the question. AWS QnABot also uses `AMAZON.FallBackIntent` to ensure that all user input is processed. To learn more about how Amazon Lex bots work, and to understand the concepts of intents, slots, sample values, fulfillment functions, refer to the *Amazon Lex Developer Guide*.

The AWS QnABot Amazon Lex web client is deployed to an Amazon S3 bucket in your account, and accessed via Amazon API Gateway.

Amazon Alexa devices

Amazon Alexa devices interact with AWS QnABot using an Alexa skill. Like an Amazon Lex chatbot, an Alexa skill also uses *intents* to encapsulate the purpose of an interaction, and *slots* to capture elements of information from the interaction.

The Alexa AWS QnABot skill uses the same Bot Fulfillment Lambda function as the Amazon Lex chatbot. When you ask a question, for example, *"Alexa, ask Q and A, How can I include pictures in Q and A Bot answers?",* your Alexa device interacts with the skill you created, which in turn invokes the Bot fulfillment Lambda function in your AWS account, passing the transcribed question as a parameter.

Content designer UI

The AWS QnABot content designer UI, like the Amazon Lex web client, is also deployed to an Amazon S3 bucket and accessed via Amazon API Gateway, and it too retrieves configuration from an API Gateway endpoint. The Content Designer website requires the user to sign in with credentials defined in an Amazon Cognito user pool.

Using temporary AWS credentials from Amazon Cognito, the Content Designer UI interacts with secure API Gateway endpoints backed by the Content Designer Lambda functions. All interactions with Amazon OpenSearch Service and Amazon Lex are handled by these Lambda functions.

Security

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

Security best practices

AWS QnABot is designed with security best practices in mind. However, the security of a solution differs based on your specific use case, and sometimes adding additional security measures will add to the cost of the solution. Th following are additional recommendations to enhance the security posture of AWS QnABot in production environments.

Amazon S3 Access logging bucket configuration

We recommend that you configure a central access logging Amazon S3 bucket, and update the S3 buckets that this solution creates to allowing access logging. For more information about Amazon S3 access logging refer to Enabling Amazon S3 server access logging in the *Amazon Simple Storage Service User Guide*.

Multi-factor authentication (MFA) in Amazon Cognito user pools

This solution creates only one user in its Amazon Cognito User Pools. MFA is not activated by default, however, we recommend using MFA for users in Amazon Cognito for a stronger security posture in production workloads. For more information about setting up MFA in Amazon Cognito, refer to Adding Multi-Factor Authentication (MFA) to a User Pool and Adding Advanced Security to a User Pool in the *Amazon Cognito Developer Guide*.

Web Application Firewall (WAF) in Amazon API Gateway

We recommend enabling WAF for API Gateway for this solution when the chatbot application is open to public in production environment. For guidance about setting up WAF, refer to Using AWS WAF to protect your APIs in the *Amazon API Gateway Developer Guide*. We also recommend reviewing the AWS Best Practices for DDoS Resiliency whitepaper for information about protecting your AWS applications from Distributed Denial of Service (DDoS) attacks.

AWS CloudFormation parameters

Before deployment, we recommend reviewing the following CloudFormation parameters:

1. The parameter **Encryption** has two possible values: `ENCRYPTED` or `UNENCRYPTED`. We recommend that you choose **ENCRYPTED** unless the solution is being used for demo purposes. This parameter determines encryption at rest for Amazon S3 and Amazon OpenSearch Service nodes.

2. The parameter **PublicOrPrivate** also has two possible values: `Public` or `Private`. We recommend choosing **Private** unless the use case for this solution dictates having the chatbot open to public without needing to sign up or register. If you select **Public**, we recommend enabling WAF in Amazon API Gateway.

Amazon Cognito

The solution uses an Amazon Cognito user pool for controlling administrative access to the AWS QnABot content designer UI, Amazon Lex web client, and Kibana dashboards. Users are also required to be members of the **{Admins}** group in the Amazon Cognito user pool.

The content designer UI requires that you sign in with credentials defined in a Amazon Cognito user pool. Using temporary AWS credentials from Amazon Cognito, the content designer UI interacts with secure API Gateway endpoints backed by the content designer's Lambda functions.

The Amazon Lex web client is deployed to an Amazon S3 bucket in your account, and accessed via Amazon API Gateway. An Amazon API Gateway endpoint provides run time configuration. Using this configuration, the web client connects to Amazon Cognito to obtain temporary AWS credentials, and then connects with the Amazon Lex service.

Single sign-on

Solution administrators can also federate into the content designer UI and Kibana using AWS SSO, where AWS SSO serves as the identity provider for the Amazon Cognito user pool. Additionally, using Amazon Cognito, you can configure a SAML or OpenID Connect identity provider to federate with as well.

When users federate into Amazon Cognito, a user profile is dynamically provisioned for them, but they will not be granted access to QnABot until they are added to the **{Admins}** group. For more information about automating using a Lambda trigger refer to Customizing User Pool Workflows with Lambda in the *Amazon Cognito Developer Guide*.

IAM roles

AWS Identity and Access Management (IAM) roles allow customers to assign granular access policies and permissions to services and users on the AWS Cloud. This solution creates IAM roles with least privileges that grant the solution's resources with needed permissions.

Design considerations

Regional deployments

This solution uses AWS services that are not currently available in all AWS Regions. You must launch this solution in an AWS Region where Amazon Lex is available. For the most current availability by Region, refer to the AWS Regional Services List.

AWS CloudFormation template

To automate deployment, this solution uses the following AWS CloudFormation template, which you can download before deployment:

aws-qnabot-main.template: Use this template to launch the solution and all associated components. The default configuration deploys Amazon Alexa, Amazon API Gateway, Amazon CloudFront, Amazon Cognito, Amazon Comprehend, Amazon Connect, Amazon DynamoDB, Amazon Kendra, Amazon Kinesis Data Firehose, AWS Lambda, Amazon Lex, Amazon OpenSearch Service, Amazon Polly, Amazon S3, AWS Systems Manager Parameter Store, and Amazon Translate, but you can customize the template to meet your specific needs.

Automated deployment

Before you launch the solution, review the cost, architecture, security, and other considerations discussed in this guide. Follow the step-by-step instructions in this section to configure and deploy the solution into your account.

Time to deploy: Approximately 30-45 minutes

> **Note**
> If you have previously deployed this solution, refer to Update the stack for update instructions.

Deployment overview

Use the following steps to deploy this solution on AWS. For detailed instructions, follow the links for each step.

Step 1. Launch the AWS CloudFormation template into your AWS account (p. 14)

Step 2. Launch the chatbot content designer (p. 17)

Step 3. Create chatbot content and load sample QandA data (p. 17)

Step 4. Interact with the chatbot (p. 20)

Step 1. Launch the stack

This automated AWS CloudFormation template deploys the AWS QnABot solution in the AWS Cloud. You must set up an AWS account before launching the stack.

> **Note**
> You are responsible for the cost of the AWS services used while running this solution. For more details, refer to the Cost (p. 2) section in this guide, and reference to the pricing webpage for each AWS service used in this solution.

1. Sign in to the AWS Management Console and use the button below to launch the `aws-qnabot-main.template` AWS CloudFormation template. Alternatively, you can download the template as a starting point for your own implementation.

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 services that are not currently available in all AWS Regions. You must launch this solution in an AWS Region where Amazon Lex is available. For the most current availability by Region, refer to 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, refer to 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.

Note
Only set `LexBotVersion` to **LexV2 Only** when deploying in an AWS Region where LexV1 is *not* supported.

Parameter	Default	Description
Authentication		
Email	*<Requires input>*	This email address will receive a temporary password to access the AWS QnABot content designer.
Username	*<Requires input>*	This username will be used to log in to the AWS QnABot content designer console.
PublicorPrivate	`PUBLIC`	Choose whether access to the AWS QnABot client should be publicly available or restricted to users in the AWS QnABot UserPool.
Amazon Kendra Integration		
DefaultKendraIndexid	<Optional input>	Index ID of an existing Kendra index, used as the default index for AWS QnABot's Kendra integration. You can use the AWS QnABot content designer to reconfigure the Kendra index settings at any time.
Amazon OpenSearch Service		
Amazon OpenSearch ServiceNodeCount	4	Number of nodes in Amazon OpenSearch Service domain. 4 is recommended for fault tolerant production deployments.
Encryption	`ENCRYPTED`	Allows encryption at rest for S3 and Amazon OpenSearch Service, and provisions m6g.large.Amazon OpenSearch Service instances — recommended for production environments. Selecting the `UNENCRYPTED` configuration provisions lower cost

Parameter	Default	Description
		t3.small.Amazon OpenSearch Service instances.
KibanaDashboardRetentionMinutes	`43200`	To conserve storage in Amazon OpenSearch Service, metrics and feedback data used to populate the Kibana dashboard are automatically deleted after this period (default 43200 minutes = 30 days). Monitor the free storage space for your Amazon OpenSearch Service domain to ensure that you have sufficient space available to store data for the desired retention period.
Amazon LexV2		
LexV2BotLocaleIds	*`<Requires input>`*	Languages for AWS QnABot voice interaction using LexV2. Specify a comma separated list of valid locale IDs. For example: `en_US`, `es_US`, `fr_CA`. For more information refer to Languages and locales supported by Amazon Lex.
Other parameters		
LetBotVersion	`LexV1 or LexV2`	Let versions to use for AWS QnABot. Select **LexV2 Only** to install AWS QnABot in AWS Regions where LexV1 is not supported.

6. Choose **Next**.
7. On the **Configure stack options** page, keep the default settings.
8. On the **Review** page, review and confirm the settings. Check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources with custom names, and the box acknowledging that AWS CloudFormation might require the CAPABILITY_AUTO_EXPAND capability.
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 30-45 minutes.

When the stack deployment is complete, the **Output** tab displays the following information:

- **ContentDesignerURL**: URL to launch the content designer UI
- **ClientURL**: URL to launch the end user client webpage
- **DashboardUrl**: URL to launch the CloudWatch dashboard for monitoring
- **FeedbackSNSTopic**: Topic name to allow feedback notifications
- **LexV1 and LexV2 bot information**: Data for configuring integration with contact centers, web clients, etc.

Note
In addition to the primary AWS Lambda function `<function(s)..>`, this solution includes the `solution-helper` Lambda function, which runs only during initial configuration or when resources are updated or deleted.
When you run this solution, you will notice both Lambda functions in the AWS console. Only the `<function>` function is regularly active. However, you must not delete the `solution-helper` function, as it is necessary to manage associated resources.

Step 2. Launch the chatbot content designer

After successful deployment, you will receive an email at the email address listed in the deployment parameters with the subject *AWS QnABot Signup Verification Code*. This email contains a generated temporary password that you can use to log in to the content designer and create your own password.

Use the following procedure to launch the content designer, reset your password, and sign in to the content designer UI.

1. Open the verification email and select the link, or alternatively, you can select the **ContentDesignerURL** link from the CloudFormation console **Outputs** tab. The designer opens in a separate browser tab.
2. Log in with your username and temporary password.
   a. Enter the **username** that you specified in the deployment parameters.
   b. Enter the temporary **password** from the verification email.
3. Follow the prompts to change your password and log in. Your new password must have a length of at least 8 characters, and contain upper-case and lower-case characters, plus numbers and special characters.
4. Log in with your username and new password.

Step 3. Create chatbot content and load sample QandA data

You must create or upload question and answer data through the content designer before sharing the AWS QnABot with your end users. Your data is stored in Amazon OpenSearch Service, which allows the data to be crawled when end users ask questions using either an Amazon Lex client UI or an Amazon Alexa hands-free device.

Use the following procedure to get started with customizing your chatbot using the solution's sample questions. You can edit the sample questions to customize the data to meet your needs.

1. From the AWS CloudFormation console, launch the content designer user interface by selecting the **ContentDesignerURL** link from the **Outputs** tab of the master CloudFormation stack.
2. Enter the administrator username you provided when you launched the stack and your new password.

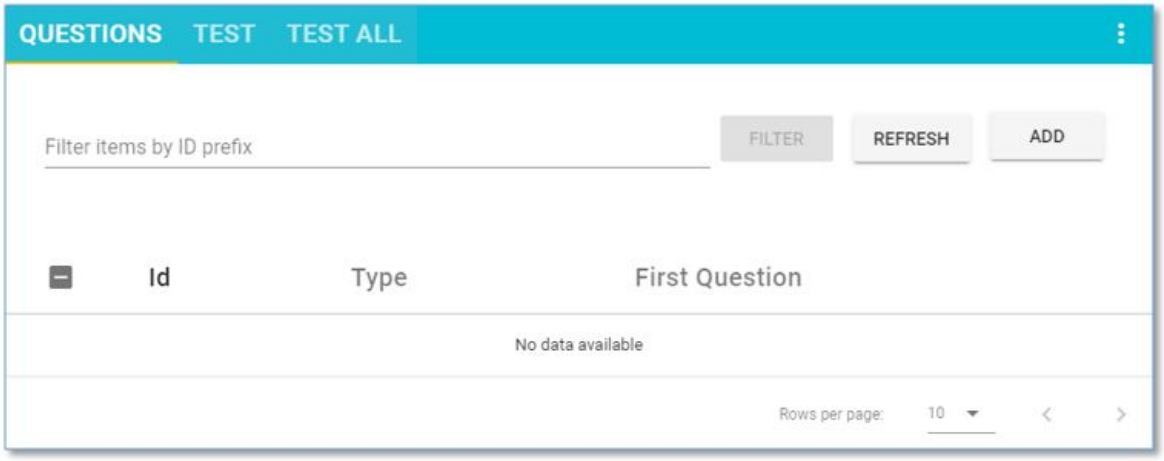

Figure 4: AWS QnABot content designer web user interface — QUESTIONS tab

3. Choose **Add.**

4. Enter the id: `AWS QnABot.001`

   Note
   Use a naming convention to identify your items within categories.

5. Enter the question: `What is Q and A Bot`
6. Enter the answer: `The Q and A Bot uses Amazon Lex and Alexa to provide a natural language interface for your FAQ knowledge base, so your users can just ask a question and get a quick and relevant answer.`
7. Select **CREATE**.
8. Repeat steps 3-7, entering the items from **Table 1: Sample QandA data** below.
   - Alternatively, you can import the items directly from a file. Select **Import from the top left tools menu (☰)**, then choose **Examples/Extensions**, find the package called **blog-samples**, and choose **LOAD.**

9. When the import is complete, choose **Edit** from the top left tools menu (☰), and then choose **LEX REBUILD** from the top right edit card menu (⋮).

Table 1: Sample QandA data

Id	Question	Answer
AWS QnABot.002	How do I use Q and A bot	Create and administer your questions and answers using the Q and A Bot Content Designer UI. End users ask questions using the Amazon Lex web UI that supports voice or chat, or using Alexa devices for hands free voice interaction.
Admin.001	How do I modify Q and A Bot content	Use the Content Designer Question and Test tools to find

Id	Question	Answer
		your existing documents and edit them directly in the console. You can also export existing documents as a JSON file, make changes to the file, and re-import.
Admin.002	Can I back up Q and A Bot content	Yes. Use the Content Designer to export your content as a JSON file. Maintain this file in your version control system or in an S3 bucket. Use the Designer UI Import feature to restore content from the JSON file.
Admin.003	Can I import Q and A Bot content from a file	Yes, the Content Designer has an import function that lets you load items from a formatted JSON file. You can create JSON files using the Export feature, or you can write your own tools to create JSON files from existing content such as a website FAQ page.
Admin.004	How do I troubleshoot and fix problems with Q and A Bot.	Use the Content Designer test tool to test a question, and check what items are returned, ranked in order of score. If the desired item does not have the highest score, then add the question to the item and run the test again. The desired item should now have the highest score. Ensure that you aren't creating items with duplicate questions to avoid unpredictable responses.
Admin.005	How can I find specific Q and A items in the Designer UI	Use the Filter feature in the **Questions** tab to filter the items list based on the ID field. Or use the **Test** tab to list all the items that match a question.
Media.001	How can I include pictures in Q and A Bot answers	Add an image attachment to the item using the Content Designer.

Step 4. Interact with the chatbot

Getting Answers using an Amazon Lex web client user interface

You can launch AWS QnABot from a Chrome, Firefox, or Microsoft Edge browser on your PC, Mac, Chromebook, or Android tablet.

1. From the AWS CloudFormation console, select the main AWS QnABot stack, choose **Output**, and then select the link to the **ClientURL**. Alternatively, launch the client by choosing **AWS QnABot Client** from the content designer tools menu (☰).
2. When your browser requests access to the microphone on behalf of the web application, allow it. The AWS QnABot chat window opens.

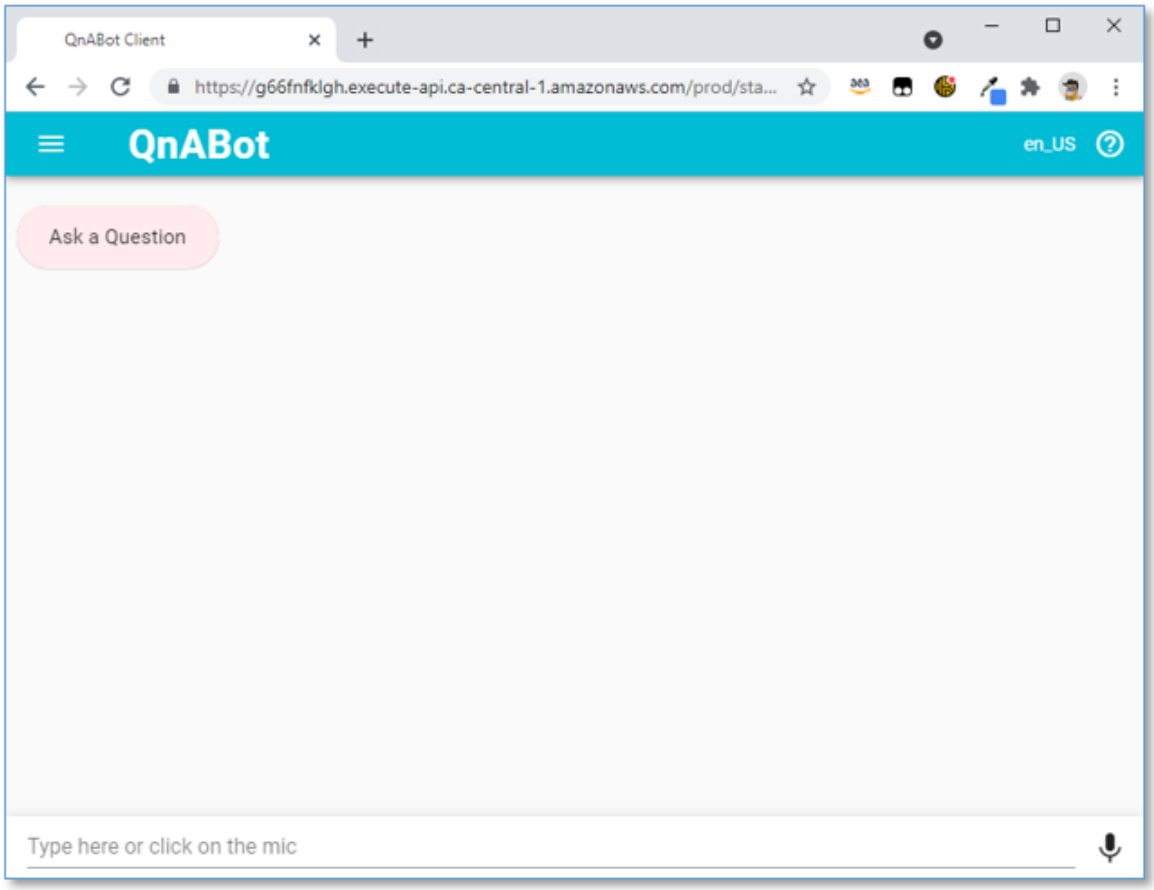

Figure 5: AWS QnABot web user interface chat window

3. Interact with the chatbot through the chat window. You can communicate through voice or text.
   - Select the microphone icon (bottom right) and say, “What is Q and A Bot?”
   - The chatbot responds with the answer you programmed in Step 3: Create chatbot content and load sample QandA data.

Getting Answers using Amazon Alexa

The AWS QnABot solution also works with Amazon Alexa, allowing your end users to get answers from your programmed content via any Amazon Alexa device, including Amazon FireTV, and any of the Amazon Echo family of devices.

> **Note**
> To integrate with Amazon Alexa, you must first use the Amazon Developer Console to create an Alexa skill for AWS QnABot. As of September 2021, this solution cannot automatically create Alexa skills. You can use the content designer to launch a walkthrough for creating an Alexa skill.

Use the following procedure to create an Alexa skill.

1. Log in to the AWS QnABot content designer, open the tools menu (☰), and select **Alexa**.
2. Follow the instructions in the console.
3. (Optional) Test your new skill in the Amazon Developer Console, even if you don't have an Alexa device nearby.

> **Note**
> If you want to publish your new AWS QnABot skill to the Alexa skills store so that other users can access it, refer to Submitting an Alexa Skill for Certification. Unpublished skills are accessible only to Alexa devices registered to your Amazon account; published skills are available to anyone.

Adding images to your answers

You can augment your answers with image attachments that can be displayed on an end users Amazon Lex web client user interface, Amazon Alexa smartphone app, or Amazon Echo Show device touch screen. Use images to display maps, diagrams, or photographs to depict places and products relevant to the question.

1. Log in to the Content Designer, and choose **Add** .
2. Enter ID: `Alexa.001` .
3. Enter question: `What is an Amazon Echo Show.`
4. Enter answer: `Echo Show brings you everything you love about Alexa, and now she can show you things. She is the perfect companion for Q and A Bot.`
5. Choose **Advanced**.
6. Enter Response card:
   a. Card Title: `Echo Show`
   b. Card ImageUrl: https://images-na.ssl-images-amazon.com/images/I/61OddH8ddDL._SL1000_.jpg
7. Choose **CREATE** to save the new item.
8. Use the web UI chat window to ask: “What is an Echo Show?” The photograph is displayed in the web UI chat.

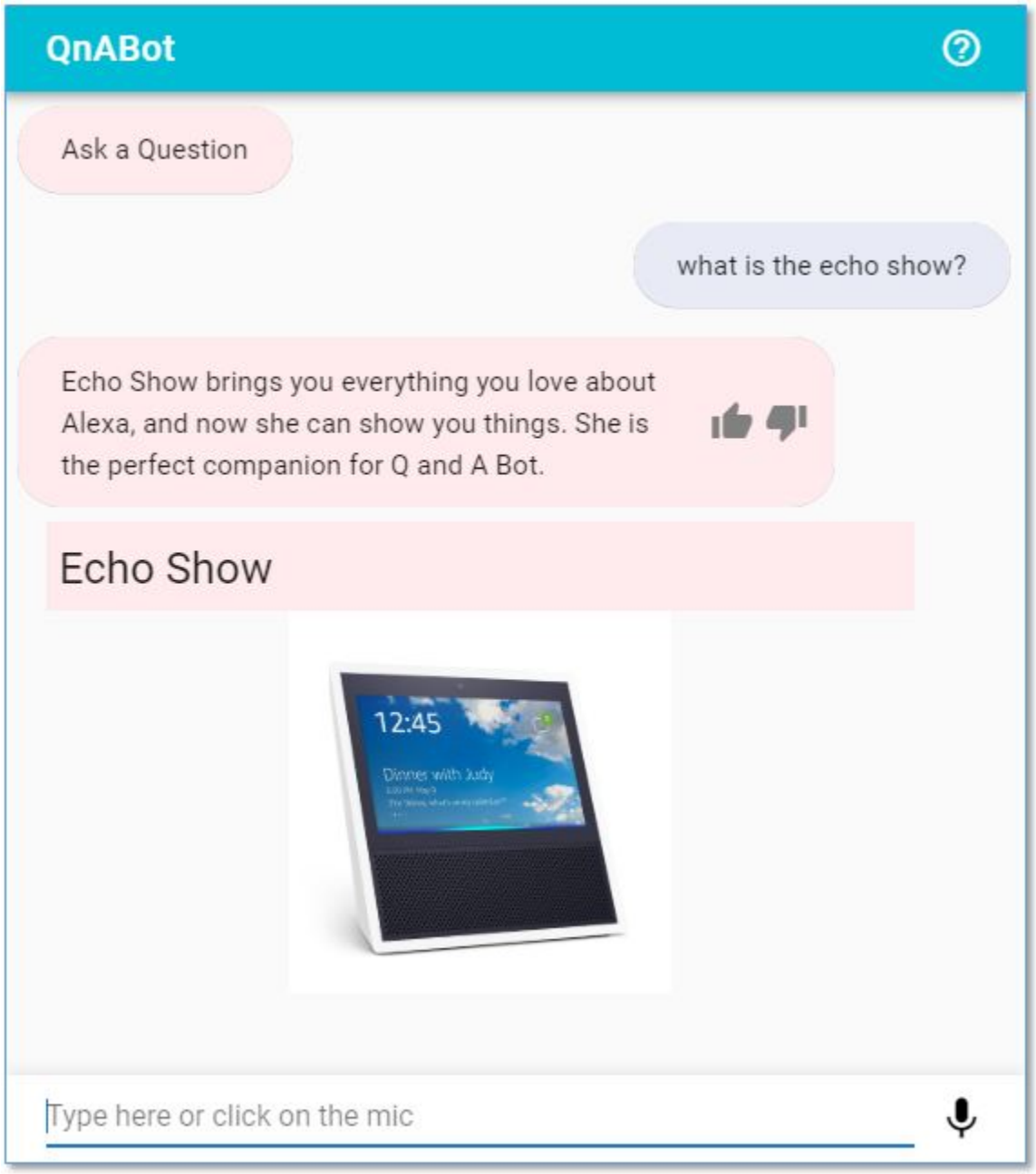

Figure 6: Example image response in the web UI chat window

9. Optionally, you can use an Amazon Echo or Echo Dot to say: “Ask Q and A, What is an Echo Show?” The card shown in the Alexa smartphone app shows the photo attachment.

10. Optionally, you can use an Amazon Echo Show to say: “Ask Q and A, What is an Echo Show?” The photo attachment is displayed on the Echo Show’s touch screen.

Displaying rich text answers

AWS QnABot supports Markdown, allowing you to create rich text versions of your answers for displaying on the web user interface, or on Slack. To use this feature, populate the **Alternate Markdown answer** field in Content Designer.

1. From the content designer, edit item *AWS QnABot001 ("What is Q and A Bot")* by opening the **Advanced** section and entering the following text in the **Markdown Answer** field:

```
# AWS QnABot
The Q and A Bot uses [Amazon Lex](https://aws.amazon.com/lex) and [Alexa](https://
developer.amazon.com/alexa) to provide a natural language interface for your FAQ knowledge
 base.
Now your users can just ask a *question* and get a quick and relevant *answer*.
```

2. Select **UPDATE** to save the modification.
3. Use the web user interface to ask: "*What is Q and A bot?*". The answer now displays the heading, links, and emphasis specified in your markdown text.

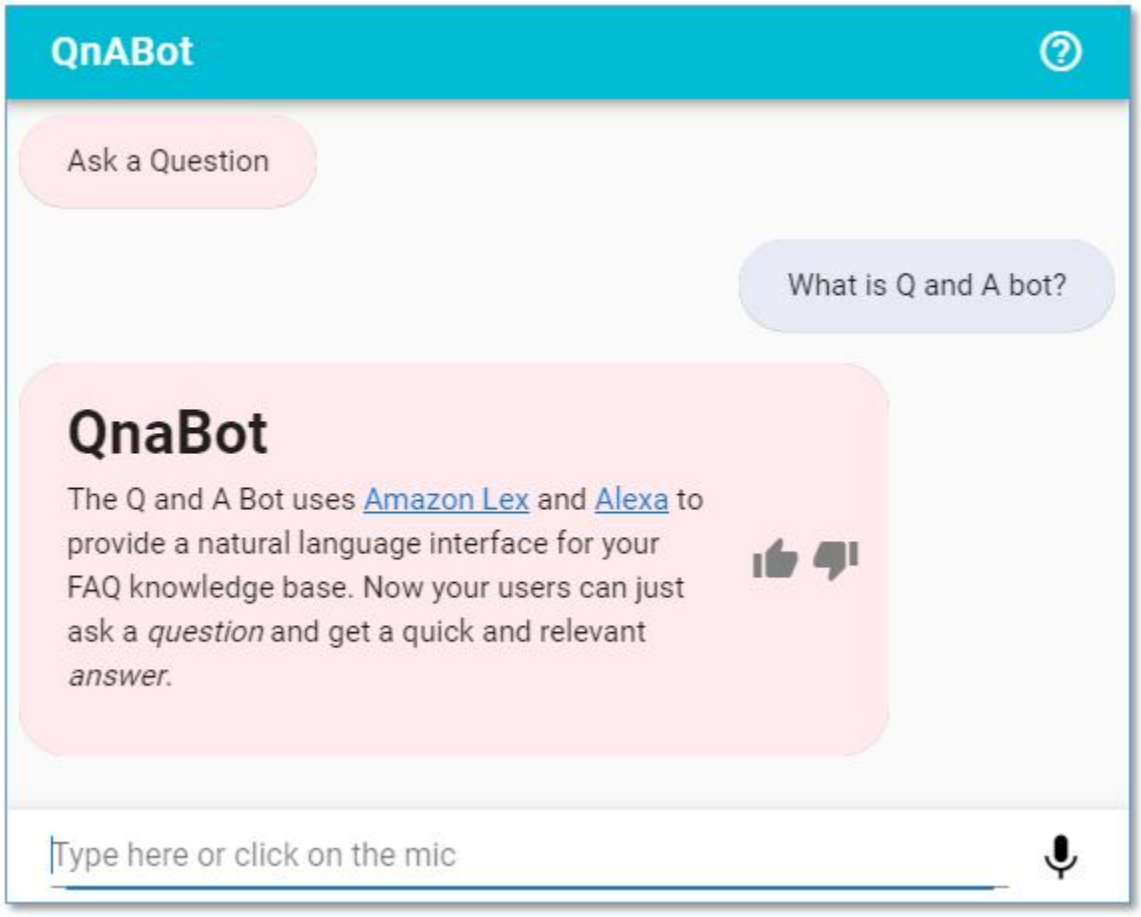

Figure 7: Example markdown

AWS QnABot also supports inline HTML in the markdown field.

1. Choose **ADD** to create a new item in HTML:
   a. Enter ID: `FireTV.001`
   b. Enter question: `What is Amazon Fire TV?`
   c. Enter answer: `Fire TV brings all the live TV and streaming content you love, and Alexa, onto the big screen. Use Alexa on the Fire TV to bring AWS QnABot into your living room!`
   d. Enter markdown answer: `**Fire TV** brings all the live TV and streaming content you love, and Alexa, onto the big screen. Use Alexa on the Fire TV to bring AWS QnABot into your living room! <iframe src="https://www.youtube.com/embed/OE4MrFx2XCs"></iframe>`
   e. Select **CREATE** to save the item.

Using SSML to control speech synthesis

The solution supports Speech Synthesis Markup Language (SSML) reference—providing additional control over the speech generation for your response. To use this feature, populate the **SSML answer** field in content designer.

1. From content designer, edit item *AWS QnABot001 ("What is Q and A Bot")* by opening the **Advanced** section and entering the following text in the SSML Answer field:

```
<speak>AWS QnABot is <emphasis level="strong">great</emphasisz7gt;. But I want to tell you
 a secret. <amazon:effect name="whispered">I am not a real human.</amazon:effect>. Can you
 believe it? </speak>
```

2. Choose **UPDATE** to save the modification.
3. Use the Web UI to ask, *using voice*: "*What is Q and A bot?*", and listen to the whispered response.

Using topics to support follow-up questions

The solution remembers the topic from the last question you asked, which allows you to ask follow-up questions, for example: "*How much does it cost?*" The correct answer depends on the context set by the previous question. To use this feature, you must assign a value to the **Topic** field in content designer.

1. From content designer, edit item *Alexa.001* ("*What is an Amazon Echo Show*"), and enter `EchoShow` as the topic in the **Advanced section**.
2. Choose **UPDATE** to save the item.
3. Edit item *AWS QnABot.001* ("*What is Q and A bot*"), and enter "*AWS QnABot*" as the topic.
4. Choose **UPDATE** to save the item.
5. Choose **ADD** to create a new item for our first follow-up question:
   - Enter ID: *Alexa.Cost*
   - Enter question: *How much does it cost?*
   - Enter answer*: For latest prices on the Echo Show, see the Amazon retail site or shopping app.*
   - Enter topic*: EchoShow*
   - Choose **CREATE** to save the item
6. Choose **ADD** to create a new item for our next follow-up question. Enter the following values:
   - Enter ID: *AWS QnABot.Cost*
   - Enter question: *How much does it cost?*
   - Enter answer*: Q and A Bot is priceless*
   - Enter topic*: AWS QnABot*
   - Choose **CREATE** to save the item
7. Use the web UI to ask the following questions and observe the context appropriate answers:
   - *"What is an Echo Show?"*
     - *The answer to this question now sets the conversation topic to 'EchoShow'.*
   - *"How much does it cost?"*
     - *The topic disambiguates this question, so it responds with the answer for the Echo Show.*
   - *"What is the Q and A bot?"*
     - *This question changes the Topic to 'AWS QnABot'.*
   - *"How much does it cost?"*
     - *The new topic allows the AWS QnABot to respond with the correct answer.*

Add buttons to the web UI

You can add buttons to your chatbot's answers to help guide your end user by suggesting what they might want to do next.

1. From Content Designer, edit item *Alexa.001* ("*What is an Amazon Echo Show*")
2. From the **Advanced** section, under **Response** card, select **Lex Buttons**.
   - Enter Display Text: *AWS QnABot*
   - Enter Button Value: *What is Q and A bot?*
3. Select **ADD LEX BUTTON** to add another button.
   - Enter Display Text: *FireTV*
   - Enter Button Value: *What is Amazon Fire TV?*
4. Select **UPDATE** to save the item with your new buttons.
5. Use the Web UI to ask: "*What is an Echo Show?*"

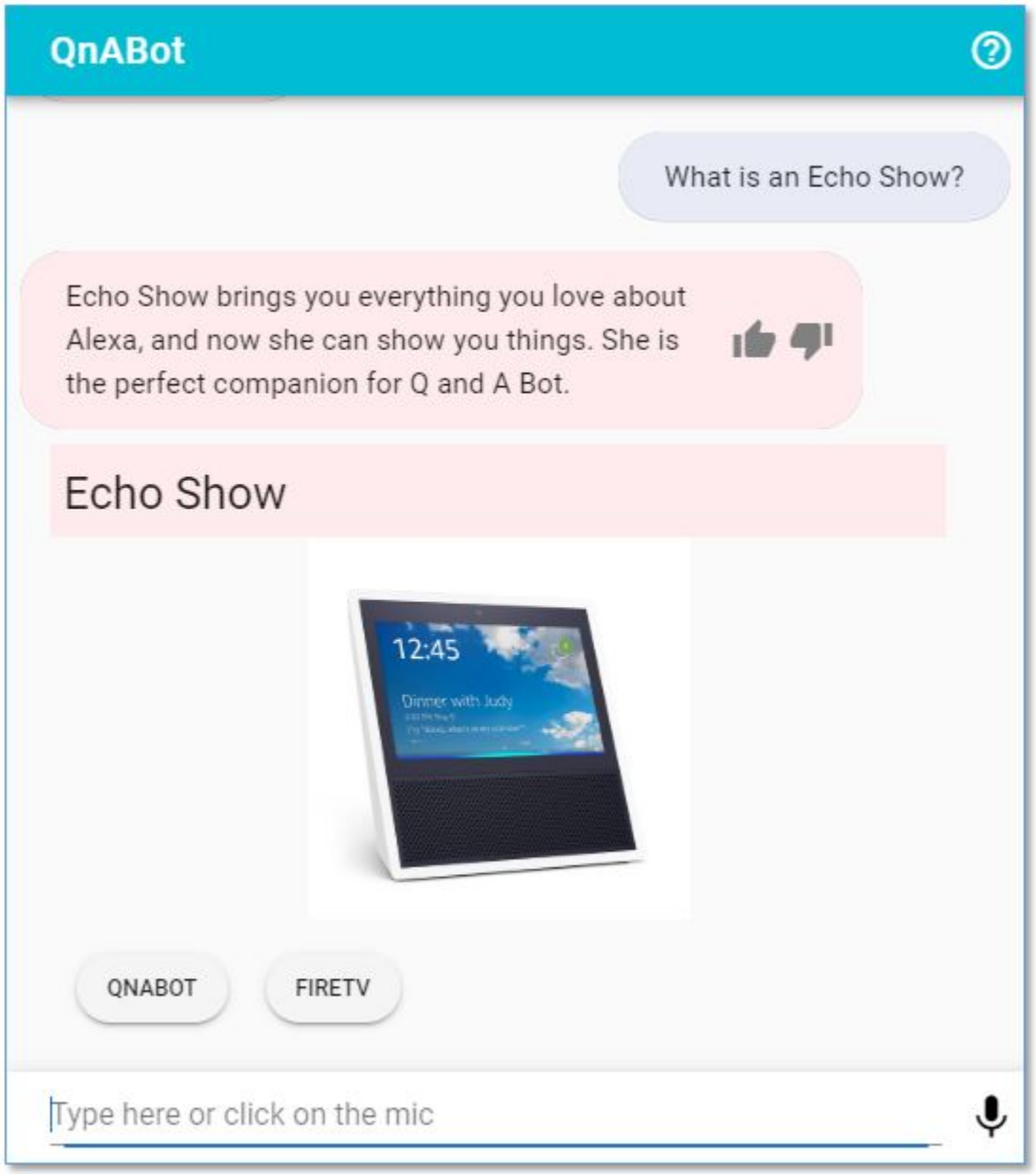

Figure 8: Example buttons

6. Choose one of the buttons to automatically send the next question to AWS QnABot.

Handlebars templates

This solution supports the Handlebars simple templating language in your answers (including in the markdown and SSML fields) which allows you to include variable substitution and conditional elements. Use the following procedure to integrate Handlebars.

1. From content designer, choose **Add**.
   - Enter ID: *Handlebars.001*
   - Enter question: *What is my interaction count?*
   - Enter answer: *So far, you have interacted with me {{UserInfo.InteractionCount}} times.*
2. Save the new item.
3. Use the web UI, or any Amazon Alexa device to say "*What is my interaction count?*" to your chatbot, and listen to it respond.
4. Ask a few more questions, and then ask "*What is my interaction count?*" again. Notice that the value has increased.
5. From Content D esigner, edit item *Handlebars.001*
6. Modify the answer to:

```
So far, you have interacted with me {{UserInfo.InteractionCount}} times.
{{#ifCond UserInfo.TimeSinceLastInteraction '>' 60}}
It's over a minute since I heard from you last.. I almost fell asleep!
{{else}}
Keep those questions coming fast.. It's been {{UserInfo.TimeSinceLastInteraction}}
 seconds since your last interaction.
{{/ifCond}}
```

7. Use the web UI, or Alexa, to interact with the chatbot again. Wait over a minute between interactions and observe the conditional answer in action.

There's a lot more that you can do with Handlebars, such as randomly selecting content from a list, setting and accessing session attributes, and generating S3 presigned URLs. For more information refer to Handlebars_README.md in the GitHub repository.

Lambda hook functions

The solution's content designer allows you to dynamically generate answers by specifying your own Lambda hook function for any item. When you specify the name, or ARN, of a Lambda function in the Lambda hook field for an item, AWS QnABot will call your function any time that item is matched to an end user's question. Your Lambda function can run code to integrate with other services, perform actions, and generate dynamic answers.

AWS QnABot comes with a simple Lambda hook function example that you can customize:

1. From the content designer, choose **Import** from the tools menu (☰).
2. Select **Examples/Extensions**, and then choose **LOAD** from the **GreetingHook** example.
3. After the import job has completed, return to the edit page, and examine the item **GreetingHookExample**. The Lambda hook field is populated with a Lambda function name.
4. Use the web UI to say "*What are lambda hooks?*". Note that the answer is prepended with a dynamic greeting based on the current time of day – in this case '*good afternoon*'.

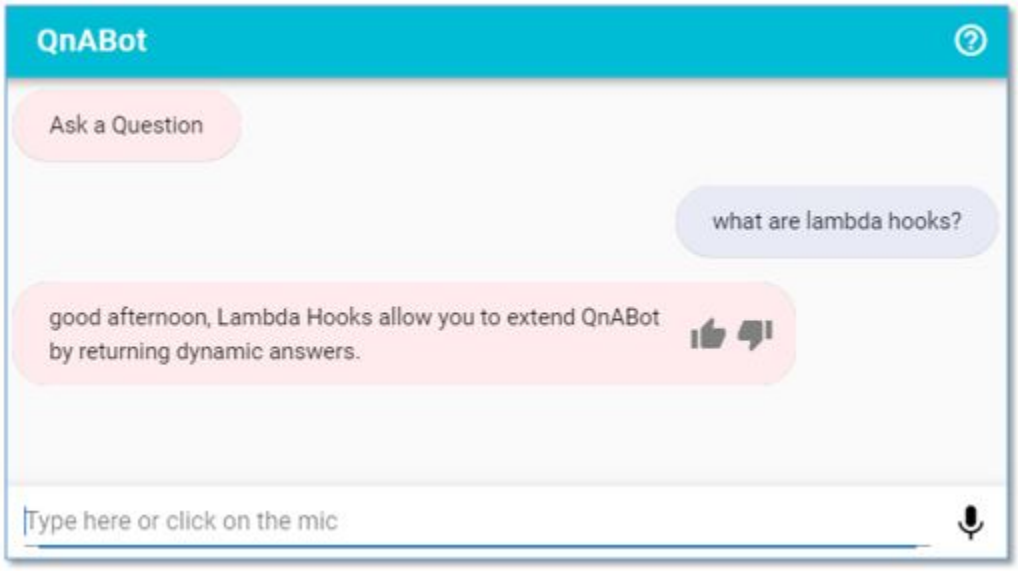

Figure 9: Example Lambda hook

5. Inspect the example function (`ExampleJSLambdahook`) using the AWS Lambda console.
6. Choose **Lambda Hooks** from the content designer tools menu (☰) to display additional information to help you create your own Lambda hook functions.

For more information about how you can package Lambda hooks, refer to the Extensions README in the AWS QnABot GitHub repository.

Keyword filters and custom “Don’t know” answers

Keyword filters

The keyword filter feature helps the solution to be more accurate when answering questions via Amazon OpenSearch Service, and to admit more readily when it doesn’t know the answer.

The keyword filter feature works by using Amazon Comprehend to determine the part of speech that applies to each word you say to AWS QnABot. By default, nouns (including proper nouns), verbs, and interjections are used as keywords. Any answer returned by AWS QnABot must have questions that match these keywords, using the following (default) rule:

- If there are one or two keywords, then all keywords must match.
- If there are three or more keywords, then 75% of the keywords must match.

If AWS QnABot can’t find any answers that match these keyword filter rules, then it will admit that it doesn’t know the answer rather than guessing an answer that doesn’t match the keywords. The solution logs every question that it can’t answer so you can see them in the Kibana Dashboard.

Custom “Don’t Know” answers

When AWS QnABot can’t find an answer, by default you’ll see or hear the response, “*You stumped me! Sadly I don’t know how to answer your question*”. You can customize this answer by creating a new item in content designer, called the *no_hits* item.

1. From content designer, choose **ADD** to create a new item:
   - Enter ID: *CustomNoMatches*
   - Enter question: *no_hits*
   - Enter answer: *Terribly sorry, but I don’t know that one. Ask me another.*
2. Choose **CREATE** to save the item.
3. Use the web UI to ask: “What are Echo Buds?”

Tuning, testing, and troubleshooting

Tuning answers using the content designer

By default, AWS QnABot attempts to match an end user's question to the list of questions and answers stored in Amazon OpenSearch Service. AWS QnABot uses full text search to find the item that is the best match for the question asked. Words that are used infrequently score higher than words that are used often, so sentence constructs such as prepositions have lower weighting than unique keywords. The closer the alignment between a question associated with an item and a question asked by the user, the greater the probability that AWS QnABot will choose that item as the most relevant answer.

The solution tries to find the best answer to questions by applying the keyword filters, and by matching the words used in the end user's question to the words used in the question fields of the stored answers —giving preference when the same words are used in the same order.

You might find that end users ask questions in ways that you haven't anticipated, resulting in unexpected answers being returned by AWS QnABot. When this happens, you can use the content designer to troubleshoot and fix the problem.

For more information, refer to the Tuning Recognition Accuracy Guide in the GitHub repository.

Testing all your questions

Use the following procedure to test your questions.

1. Log in to the content designer, and choose **TEST ALL.**
2. Use the default filename, or enter your own.
3. Optionally, if you want to test only a subset of questions, you can filter by **qid** prefix. Leave this field blank to test all the questions.
4. Select **TEST ALL**, and wait for the tests to complete.
5. Select the **view results icon** on bottom right to view the test results. Any incorrect matches are highlighted in red. Test results can be viewed in the browser, or downloaded to your computer as a CSV file.

Tuning the chatbot's Automatic Speech Recognition

When you ask AWS QnABot a question, it is processed and transcribed by either Amazon Lex or Amazon Alexa using an Automatic Speech Recognition (ASR) engine. AWS QnABot initially trains the ASR to match a wide variety of possible questions and statements, so that the Amazon Lex chatbot and Alexa skill will accept almost any question a user asks.

This solution supports `AMAZON.FallbackIntent` in both Amazon Lex and Alexa, which allows it to process anything end users say without needing to retrain and rebuild the Amazon Lex chatbot or Alexa skill.

Occasionally, the transcription shown in the web client or the Alexa app isn't accurate. This can happen with unusual words that are confused for other more common words or phrases. Use one of the following approaches to troubleshoot this error:

- Use content designer to add additional question variants that match the actual transcription shown in the web client or in the Alexa app; this allows AWS QnABot to anticipate the transcription accuracy problem, and respond anyway.
- Retrain the Amazon Lex and Alexa ASR with examples to influence the transcription to more closely match what you want – refer to **Retrain Amazon Lex** for more information.

Retrain Amazon Lex

AWS QnABot can automatically generate additional ASR training data for Amazon Lex using questions from the data you have added.

1. Log in to the content designer and choose **LEX REBUILD** from the top right edit menu (⋮)**.**
2. Wait for the rebuild to complete.

Retrain Alexa

AWS QnABot can generate additional ASR training data for Alexa using questions from the data you have added.

1. Log in to the content designer and choose **ALEXA UPDATE** from the top right edit menu (⋮).
2. Select **COPY SCHEMA** to copy the updated Alexa skill schema.
3. Log in to the Alexa Developer Console, open your AWS QnABot skill, and then use the JSON Editor to paste the new schema, replacing the existing one.
4. Save and then build the updated model.

Monitoring AWS QnABot usage and user feedback

The solution logs everything that end users say to the chatbot. Amazon Kinesis Data Firehose stores logged utterances to a new index in Amazon OpenSearch Service.

You can also allow your end users to provide feedback about the chatbot's answers. Use the following procedure to set up the feedback mechanism.

1. From content designer's top left tools menu (☰), select **Import**.
2. Open **Examples/Extensions**, and select the **LOAD** for the **QnAUtility** demo.
3. From the top left tools menu (☰), select **EDIT** and examine the newly imported items, *Feedback.001* and *Feedback.002***;** observe the list of default expressions that the end user can input to invoke feedback. (The example *QnAUtility* demo package also loads items *Help*, *CustomNoMatches, CustomNoVerifiedIdentity*.)
4. Test the feedback mechanism.
   a. Use the web UI to ask a question, such as: "*What happens if I ask an unanticipated question?*". Because we have not entered a suitable answer for this question, AWS QnABot responds with the newly imported *CustomNoMatches* response—indicating that it doesn't know the answer.
   b. From the web UI, speak or type "*Thumbs down*", or select the thumbs down icon beside the answer.

The solution publishes *Thumbs down* feedback messages to the Amazon Simple Notification Service (SNS) topic identified by the **FeedbackSNSTopic** on the **Outputs** tab of the CloudFormation stack. To learn how to subscribe to the SNS topic, and receive a message from the each time a user provides

feedback, refer to Subscribing to an Amazon SNS topic in the *Amazon Simple Service Notification Developer Guide.*

Use the following process to visualize the usage logs and feedback using the Amazon OpenSearch Service Kibana dashboard. Note that it can take up to 5 minutes for new utterances and end user feedback to become visible in the dashboard.

1. From content developer, select the top left tools menu (☰), and choose **Kibana Dashboard**. Kibana opens in a new browser tab.
2. Choose the AWS QnABot Dashboard to visualize usage history and sentiment, all logged utterances, no hits utterances, positive user feedback, and negative user feedback.

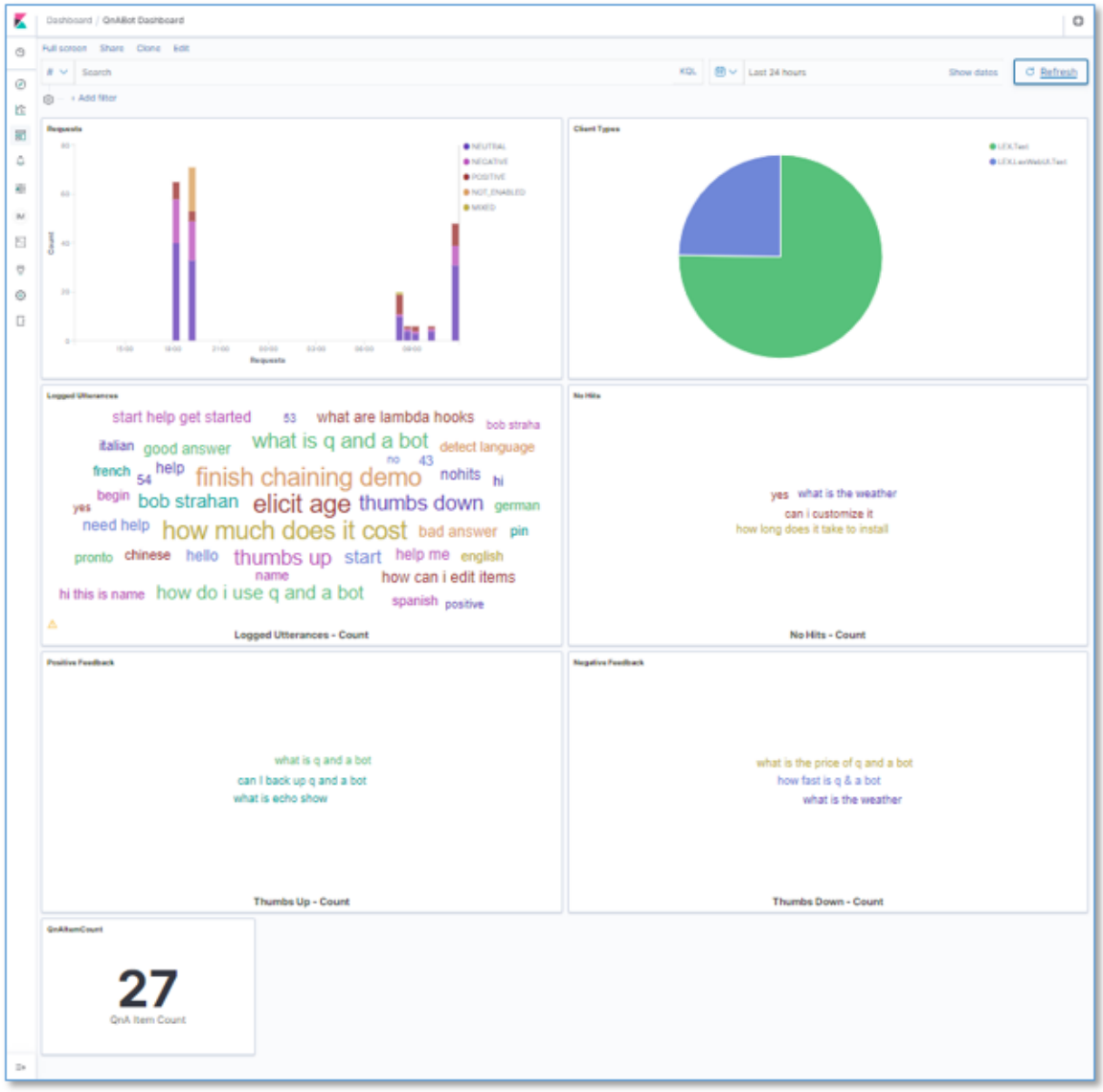

Figure 10: Sample Kibana dashboard

3. Edit Kibana to change the time span, customize and build your own visualizations, or to run your own queries.

Using Amazon CloudWatch to monitor and troubleshoot

The solution's metrics and logs are available in an Amazon CloudWatch dashboard. Use the following procedure to launch the dashboard and visualize the solution's AWS resources.

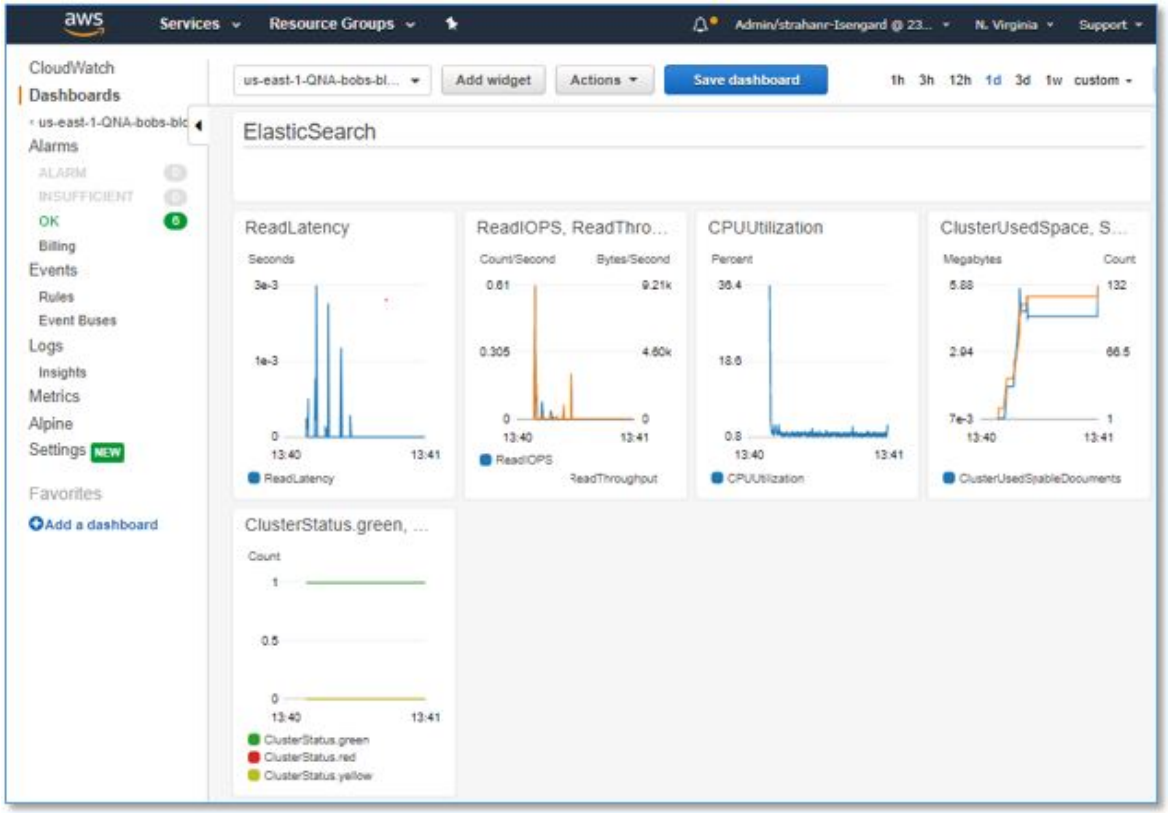

Figure 11: Sample CloudWatch dashboard

1. From the CloudFormation stack's **Outputs** tab, select the **DashboardUrl** link.
2. When troubleshooting the chatbots responses to your questions, trace the request and response using the logs created by the `Fulfillment` Lambda function.
   a. Choose the **menu** tool in the upper right of the `Fulfillment` Lambda widget, select **View logs**, then and choose the **AWS Lambda function**.

Figure 12: FulfillmentLambda function

2. Inspect the log messages. Each interaction with the solution is delimited by **START** and **END** messages. Between these messages are insights into how the solution processes the question.

Importing and exporting chatbot answers

The solution's content designer allows you to export and import your content using JSON and Excel files.

Use the export feature to create backup versions of your content that you can use to restore if you accidentally delete items or need to go back to a previous version. You can also use the exported files to load content into another instance of your chatbot to help with test deployments.

The following procedure exports all the items that are in the AWS QnABot category (items whose ID starts with "*AWS QnABot*").

1. Log in to the content designer, choose the tools menu (☰), and then select **Export.**
2. Enter `AWS QnABot` in the optional filter field, and then select **EXPORT** to generate a JSON file containing the filtered items.
3. After the export has completed, choose the download tool (bottom right) to download the exported file.
4. Open the exported file in a text editor and inspect the JSON structure.


```
{
   "qna": [
      {
         "q": [
            "What is Q and A Bot"
         ],
         "a": "The Q and A Bot uses Amazon Lex and Alexa to provide a natural
         language interface for your FAQ knowledge base, so your users can just
         ask a question and get a quick and relevant answer.",
         "r": {
            "title": "",
            "imageUrl": ""
         },
         "qid": "QnABot.001"
      },
      {
         "q": [
            "How do I use Q and A bot"
         ],
         "a": "Create and administer your questions and answers using the Q and
         A Bot Content Designer UI. End users ask questions using the Lex web UI
         which supports voice or chat, or using Alexa devices for hands free
         voice interaction. ",
         "r": {
            "title": "",
            "imageUrl": ""
         },
         "qid": "QnABot.002"
      }
   ]
}
```

Figure 13: Sample JSON file

5. Add a new item to the qna list, as shown in the following example, and save the file.

```
{
        "qid": "AWS QnABot.003",
        "q": ["What can Q and A bot do"],
        "a": "You can integrate it with your website to provide quick and easy access
 to frequently asked questions. Use it with Alexa to provide hands free answers in the
 kitchen, in the factory or in the car. Since it can display images too, use it to provide
 illustrations and photographs to enrich your answers.",
        "r": {
            "title": "",
            "imageUrl": ""
        }
}
```

6. From the Content Designer, select **Import/Export**, and then choose **From File**.
7. Import your modified JSON file. Importing items with the same ID as an existing item will overwrite the existing item with the definition contained in the JSON file.
8. From the Content Designer, enter `AWS QnABot` in the filter field, and inspect the newly imported item, **AWS QnABot.003**.

Modifying configuration settings

The solution uses AWS Systems Manager Parameter Store to hold default and custom configuration settings. You can view and edit these settings using the new **Settings** menu in the content designer.

1. Log in to the content designer, select the tools menu (☰), and then choose **Settings.**
2. Change the value of the setting **ES_USE_KEYWORD_FILTERS** from **true** to **false**.
3. Scroll to the bottom and select **Save**. This turns off the *new keyword filters feature*.

You can further customize how the keyword filters feature works by changing the following settings:

- **ES_KEYWORD_SYNTAX_TYPES**: A list of tokens representing parts of speech identified by Amazon Comprehend.
- **ES_MINIMUM_SHOULD_MATCH:** A query rule used to determine how many keywords must match an item question in a valid answer.

Explore the available configuration settings, and override the defaults to configure the solution's customize keyword filtering, answer field scoring, messages, redaction from logs and metrics (**ENABLE_REDACTING** and **REDACTING_REGEX**), and more. You can also start the debug mode (**ENABLE_DEBUG_RESPONSES**), initiate fuzzy matching (**ES_USE_FUZZY_MATCH**), and experiment with score boosting for exact phrase matches (**ES_PHRASE_BOOST**).

> **Note**
> Custom settings will not be replaced when you upgrade the solution.

Integrate Amazon Kendra

Amazon Kendra is an intelligent search service powered by machine learning. There are two ways to take advantage of Amazon Kendra's natural language processing model to enhance the solution's ability to understand human questions:

1. Use Amazon Kendra's FAQ queries to match users' questions to the answers in the solution's knowledge base. Kendra's machine learning models can handle many variations in how users phrase their questions, and this can reduce the amount of tuning needed for the solution to find the right answer from your knowledge base.
2. Use Amazon Kendra's document index as a fallback source of answers when a question/answer is not found in the solution's knowledge base.

For more information, refer to Amazon Kendra Pricing, and use the Getting started topic in the *Amazon Kendra Developer Guide* to create your Amazon Kendra index.

Using Kendra FAQ for question matching

Use the following procedure to configure the solution to use your Amazon Kendra index to answer questions from the data populated in content designer:

1. Update the AWS QnABot setting **KENDRA_FAQ_INDEX** to specify the Amazon Kendra index to use.
   - Copy/paste your Index ID from the Amazon Kendra console. *(If you previously provided a value for the parameter DefaultKendraIndexId when you installed or updated AWS QnABot, then KENDRA_FAQ_INDEX will be pre-configured.)*
2. Replicate all items from Content Designer to the Kendra index:
   - Select the menu (⋮) from the top right in Content Designer.
   - Choose **SYNC KENDRA FAQ** and wait for it to complete – it may take a few minutes.

The solution will now use Kendra FAQ queries to find matches to end users' questions. Use the setting **ALT_SEARCH_KENDRA_FAQ_CONFIDENCE_SCORE** to adjust the confidence threshold for Kendra FAQ answers used by AWS QnABot.

If Kendra FAQ cannot not find an answer that meets the confidence threshold, the solution will revert by default to using an Amazon OpenSearch Service query. The combination of Kendra FAQ and Amazon OpenSearch Service gives you the best of both worlds.

Using Kendra search as a fallback source of answers

You can add one or more data sources to your Amazon Kendra index, and configure the solution to query your index any time it gets a question that it doesn't know how to answer.

1. Update the AWS QnABot setting **ALT_SEARCH_KENDRA_INDEXES** to specify one or more Amazon Kendra indexes to use for fallback searches.
   a. The value of ALT_SEARCH_KENDRA_INDEXES should be either a single index id, or an array of quoted index ids, for example: 857710ab-example-do-not-copy OR ["857710ab-

example1-do-not-copy","857710ab-example2-do-not-copy"] *(If you previously provided a value for the parameter DefaultKendraIndexId when you installed or updated AWS QnABot, then ALT_SEARCH_KENDRA_INDEXES will be pre-configured.)*

Web page indexer

This solution can answer questions based on the content of web pages.

1. From content designer, select the tools menu (☰), and then choose **Settings.**
2. Modify the following settings:
   a. **ENABLE_WEB_INDEXER:** *true*
   b. **KENDRA_INDEXER_URLS:** *https://aws.amazon.com/lex/faqs/*
   c. **KENDRA_WEB_PAGE_INDEX:** *Existing Amazon Kendra index ID*
   d. **KENDRA_INDEXER_SCHEDULER:** *rate (1 day)*
3. From content designer, select the tools menu (☰), and then choose **Kendra Web Page Indexing**.
   a. Choose **START INDEXING**.
   b. Wait for indexing to complete (it can take several minutes).
4. Open the web UI, and ask "*What is Lex?*". AWS QnABot provides an answer with a link to the Amazon Lex FAQ page.

For more information on web page indexing, refer to the README.md in GitHub.

Automatic translation

This solution supports automatic translation to the end user's language using Amazon Translate.

1. Turn on multiple language support by setting: **ENABLE_MULTI_LANGUAGE_SUPPORT** to *true.*
2. In the web UI, ask: *Qu'est-ce que q et a bot?*
3. The chatbot replies to you in French.

The solution also supports speech recognition and voice interaction in multiple languages. When you install or update AWS QnABot, specify the languages using the CloudFormation parameter **LexV2BotLocaleIds**. The default languages are US English, US Spanish, and Canadian French, but you can customize the list to use any of the languages supported by Amazon LexV2.

Use the **ENABLE_DEBUG_RESPONSES** setting to see how local language questions are translated to English by AWS QnABot, and use this translation to tune the content as needed to ensure AWS QnABot finds the best answer to a non-English question.

The solution also supports Amazon Translate custom terminology to provide additional control over the translation of entities and phrases. Refer to the README.md file in GitHub for more information about how to use the **Import Custom Terminology** tool in content designer.

Configuring the chatbot to ask the questions and use response bots

You can configure your chatbot to ask questions and process your end user's answers. Use this feature for data collection and validation; to implement surveys, quizzes, personalized recommendations; or triage chatbot applications.

Note
The solution's built-in response chatbots are not yet available in LexV2.

Use the following procedure to configure the chatbot to ask questions.

1. Log in to the content designer and choose **Add.**
2. Enter ID: *ElicitResponse.001*
3. Enter question: *Ask my name*
4. Enter answer: *Hello. Can you give me your First Name and Last Name please.*
5. Choose **Advanced**.
   a. Enter **Elicit Response: ResponseBot Hook**: *QNAName*
   b. Enter **Elicit Response: Response Session Attribute Namespace**: *name_of_user*
6. Choose **CREATE** to save the new item.
7. Use the web UI to say: "*Ask my name*"
8. Respond by entering your name. Try responding naturally and see if chatbot confirms your name correctly. If not, you can choose NO and try again.

The **ResponseBot Hook** field specifies the name of an Amazon Lex chatbot. In this case we specified the name of a chatbot, *QNAName*, that was automatically created for us when the solution was installed. *QNAName* is a built-in response chatbot designed to process names (first and last name). It handles a variety of ways the user might state their name, and it will prompt the user to confirm or to try again. If the user confirms by choosing **YES**, the response chatbot will return the **FirstName** and **LastName** values back to the solution (as slot values in a fulfilled response).

The solution stores the returned **FirstName** and **LastName** values in a session attribute. The name of the session attribute is determined by the value you provided for **Response Session Attribute Namespace** (in this case *name_of_user*) and the slot name(s) returned by the response chatbot (in this case *FirstName* and *LastName*).

The session attribute set by **Elicit Response** can be used in other items to provide conditional or personalized responses.

1. Log in to the content designer, and choose **Add.**
2. Enter ID: *ElicitResponse.002*
3. Enter question: *Ask my age*
4. Enter answer: *Hello {{SessionAttributes.name_of_user.FirstName}} – What is your age in years?*
5. Choose **Advanced**.
   a. Enter **Elicit Response: ResponseBot Hook**: *QNAAge*
   b. Enter **Elicit Response: Response Session Attribute Namespace**: *age_of_user*
6. Choose **CREATE** to save the new item.
7. Use the Web UI to say: *"Ask my age"*

Response bots

The solution provides a set of built-in response bots that you can use out of the box:

- **QNAYesNo:** returns slot “Yes_No” with value either “Yes” or “No”
- **QNAYesNoExit:** returns slot “Yes_No_Exit” with value either “Yes”, “No”, or “Exit”
- **QNADate:** returns slot “Date” with value of date (YYYY-MM-DD)
- **QNADayOfWeek:** returns slot “DayOfWeek”
- **QNAMonth:** returns slot “Month”
- **QNANumber:** returns slot “Number”
- **QNAAge:** returns slot “Age”
- **QNAPhoneNumber:** returns slot “PhoneNumber”
- **QNATime:** returns slot “Time” with value of time (hh:mm)
- **QNAEmailAddress:** returns slot “EmailAddress”
- **QNAName:** returns slots “FirstName” and “LastName”
- **QNAFreeText**: returns slots “FreeText” and “Sentiment” (*New in 4.0.0 – used to capture free-form notes or comments that can be used in, say, trouble tickets, emails, or for feedback sentiment analysis.).*

You can also add your own Amazon Lex bots and use them as response bots. Response chatbot names must start with the letters “QNA”. The solution calls your chatbot with the user’s response, and captures all the slot names and values returned when your chatbot sends back a fulfilled message.

Advancing and branching through a series of questions

The following example configures the solution to automatically ask your age after you provide your name.

1. Log in to the content designer and edit item *ElicitResponse.00.1*
2. Choose **Advanced**.
3. Enter **Document Chaining: Chaining Rule**: *'ask my age'*
4. Choose **UPDATE** to save the modified item.
5. Use the web UI to ask: “Ask my name”
   a. Enter and confirm your name.
   b. Enter and confirm your age.

The solution automatically asks you for your age after you confirm your name. Because you specified the next question, ‘*ask my age*’, as the chaining rule, the solution automatically found and advanced to the matching item.

Next, create a *conditional* chaining rule that will branch to different items depending on previous answers.

1. Log in to the content designer and add two new items:
   a. ID: *ElicitResponse.003*, question: “*Under 18”*, answer: “*Under 18 answer*“.
   b. ID: *ElicitResponse.004*, question: “*Over 18*“, answer: “*Over 18 answer*“.
2. Edit item *ElicitResponse.002*

a. Add Chaining Rule: *(SessionAttributes.age_of_user.Age< 18) ? "Under 18" : "Over 18"*
b. Choose **UPDATE** to save the modified item.

3. Use the web UI to ask: "Ask my name"
   a. Enter and confirm your name.
   b. Enter and confirm your age.

When you confirm your age, the solution automatically branches to one of the two new items you added, depending on your age. The chaining rule is a JavaScript programming expression used to test the value of the session attribute set by elicit response; if it is < 18 then advance to the item matching the question '*Under 18'*, otherwise advance to the item matching the question '*Over 18'*.

Combine expressions with logical operators to test multiple session attributes in a single rule, and use nested expressions to implement more than two branches in a chaining rule. Use the alternate syntax *SessionAttributes('age_of_user.Age')* to avoid a processing error if the referenced session attribute does not exist.

You can also apply chaining rule expressions to all the context variables supported by the handlebars feature including UserInfo fields, Settings fields, and more – refer to the README.md file in GitHub for a list of available variables.

Identify the next document using its Qid value instead of a question using a string that starts with '*QID::*' followed by the Qid value of the document, for example, a rule that evaluates to "*QID::Admin001*" will chain to item Admin.001.

You can optionally specify an AWS Lambda function instead of a JavaScript expression when you need to evaluate complex chaining rule logic. Your Lambda function is invoked with the full user request context and should evaluate and return the next question as a simple string. Alternatively, the Lambda function may return an event object where the `event.req.question` key was updated to specify the next question – by returning an event object, your `chaining rule` Lambda function can modify session attributes, similar to Lambda Hooks. Use Lambda functions to implement chaining rules that require complex logic, data lookup, etc. A `chaining rule` Lambda function name must start with the letters "QNA", and is specified in the **Document Chaining:Chaining Rule** field as: `Lambda::FunctionNameOrARN`.

> **Note**
> If the chaining rule has an error, the solution will return the message, "Unfortunately I encountered an error when searching for your answer. Please ask me again later."

Connect AWS QnABot to an Amazon Connect call center

The solution can automate data collection and answer frequently asked questions using AWS QnABot within an Amazon Connect contact flow. Optionally, you can also configure the solution to use Amazon Connect to make outbound calls; your users can use the web UI or the Alexa skill to ask AWS QnABot to call their phone so they can speak to a human.

Use the Amazon Connect integration wizard to connect AWS QnABot to a call center.

1. Log in to the content designer, select the tools menu (☰), and then choose **Connect**.
2. Follow the step-by-step directions in the wizard to create a contact center using the solution to answer caller's questions.

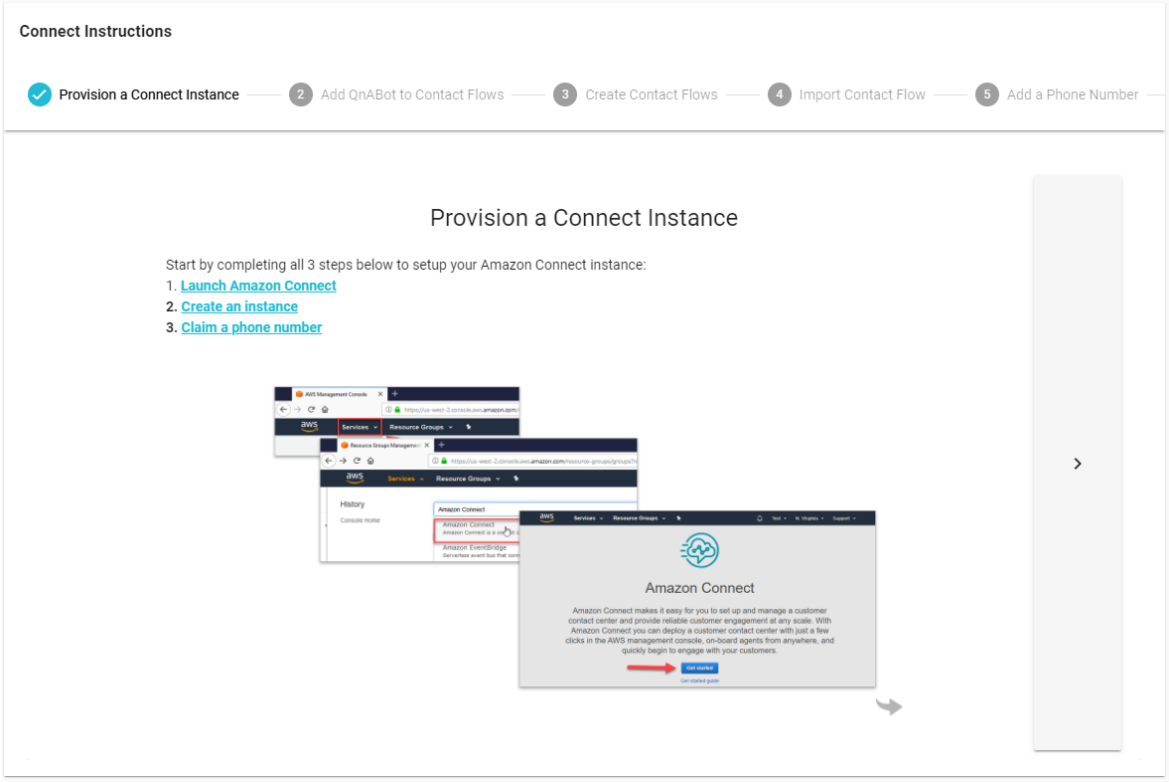

Figure 13: Amazon Connect integration wizard

Deploy a Web UI for your chatbot

After you finish building your QnABot, you can separately deploy the Lex Web UI and use it to publish the QnABot on your website. This web UI includes optional integrated user authentication, which you can use to create personalized responses from QnABot. For more information, refer to the Deploy a Web UI for Your Chatbot blog post.

Additional resources

AWS services

- Amazon Alexa
- Amazon API Gateway
- Amazon CloudFront
- Amazon Cognito
- Amazon Comprehend
- Amazon Connect
- Amazon DynamoDB
- Amazon Kendra
- Amazon Kinesis Data Firehose
- AWS Lambda
- Amazon Lex
- Amazon OpenSearch Service
- Amazon Polly
- Amazon S3
- AWS Systems Manager Parameter Store
- Amazon Translate

Related AWS Documentation

Blog posts

- Create a Question and Answer Bot with Amazon Lex and Amazon Alexa
- Create a questionnaire bot with Amazon Lex and Amazon Alexa
- Creating virtual guided navigation using a Question and Answer Bot with Amazon Lex and Amazon Alexa

Deploy a Web UI for Your Chatbot

Workshop

- AWS QnABot Workshop

YouTube demo

Multi-lingual FAQ bots with agent transfer using Amazon Lex, Amazon Kendra, Amazon Connect, and open source AWS QnABot

Update the stack

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

1. Sign in to the AWS Cloudformation console, select your existing AWS QnABot CloudFormation stack, and choose **Update**.
2. Select **Replace current template**.
3. Enter the Amazon S3 URL:

- https://solutions-reference.s3.amazonaws.com/aws-qnabot/latest/aws-qnabot-main.template
- https://solutions-reference.s3.amazonaws.com/aws-qnabot/latest/aws-qnabot-vpc.template

4. Under **Parameters**, review the parameters for the template and modify them as necessary. Refer to Step 1. Launch the Stack (p. 14) for details about the parameters.
5. Choose **Next**.
6. On the **Configure stack options** page, choose **Next**.
7. On the **Review** page, review and confirm the settings. Be sure to check the box acknowledging that the template might create AWS Identity and Access Management (IAM) resources.
8. Choose **View change set** and verify the changes.
9. Choose **Update stack** to deploy the stack.

> **Note**
> **Kendra Fallback users only**—If you allowed Kendra Fallback in an earlier release then after you update to v4.6.0 or later you must remove the custom KendaFallback 'no_hits' item.
> In Content Designer: Find the item KendraFallback and delete it.
> Find the item *CustomNoMatches* if it exists, and allow it by modifying the question "no_hits_original" to "no_hits". If the *CustomNoMatches* item does not exist, you can create it as described in the section titled *Custom "Don't Know" answers*, or by importing the *QnAUtility* demo package.

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 30 minutes.

Uninstall the solution

You can uninstall the AWS QnABot solution from the AWS Management Console or by using the AWS Command Line Interface.

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, refer to 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>
```

Source code

Visit our GitHub repository to download the source files for this solution and to share your customizations with others. Refer to the README.md file for additional information.

Revisions

Date	Change
September 2021	v5.0.0 - Initial AWS Solutions Implementation release. For more information, refer to the CHANGELOG.md file in the GitHub repository.
October 2021	v5.0.1 - Bug fix for redaction of PII in logs; documentation addition for deploying a web UI. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Contributors

- Bob Strahan
- Bob Potterveld
- John Calhoun
- Mohsen Ansari

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.