Implementation guide

QnABot on AWS

QnABot on AWS: Implementation guide

Amazon's trademarks and trade dress may not be used in connection with any product or service that is not Amazon's, in any manner that is likely to cause confusion among customers, or in any manner that disparages or discredits Amazon. All other trademarks not owned by Amazon are the property of their respective owners, who may or may not be affiliated with, connected to, or sponsored by Amazon.

Table of Contents

Create a custom question and answer chatbot

Publication date: September 2021 (last update: April 2024)

The QnABot on AWS solution is an open source, multi-channel, multi-language conversational chatbot that responds to your customer's questions, answers, and feedback. It is built on Amazon Lex, Amazon Polly, Amazon OpenSearch Service, Amazon Translate, Amazon Comprehend, and Amazon Kendra. The QnABot on AWS solution allows customers to quickly deploy self-service conversational artificial intelligence (AI) on multiple channels including their contact centers, web sites, social media channels, SMS text messaging, or Amazon Alexa without programming.

This implementation guide provides an overview of the QnABot on AWS solution, its reference architecture and components, considerations for planning the deployment, configuration steps for deploying the QnABot on AWS solution to the Amazon Web Services (AWS) Cloud. It also includes a user's guide with prescriptive guidance for using QnABot on AWS, and a guide with an API reference for integrating the solution with the Canvas Learning Management System (LMS).

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

If you want to . . .	Read . . .
Know the cost for running this solution	Cost
Understand the security considerations for this solution	Security
Know how to plan for quotas for this solution	Quotas
Know which AWS Regions are supported for this solution	Supported AWS Regions
View or download the AWS CloudFormation template included in this solution to automatically deploy the infrastructure resources (the "stack") for this solution	AWS CloudFormation templates

Use cases

Contact centers – How can I help?

Virtual agents to automatically help resolve customer questions or guide customers to the right agent.

Informational bots – Can I answer your question?

Chatbots for everyday requests and frequently asked questions.

Enterprise productivity bots – Can I help you get more done?

Streamline internal enterprise work activities and improve efficiencies.

Features and benefits

With the solution's content management environment and the Contact Center Integration wizard, you can 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.
- Uncover insights and business trends.
- Reduce call center wait times by automating customer support workflows.
- Expand existing and grow new channels.
- Implement the latest machine learning technology to create engaging, human-like interactions for chatbots.
- Lower customer support costs.

QnABot on AWS provides the following features:

High quality speech recognition and natural language understanding

Automatic speech recognition and natural language understanding technologies to create a Speech Language Understanding system with Amazon Lex. Amazon Lex uses the same proven technology that powers Alexa. Amazon Lex is able to learn the multiple ways users can express their intent based on a few sample utterances provided by the developer. The speech language understanding system takes natural language speech and text input, understands the intent behind the input, and fulfills the user intent by invoking the appropriate response.

Context management

As the conversation develops, being able to classify utterances accurately requires managing context across multi-turn conversations. The Amazon Lex component of QnABot on AWS supports context management natively, so you can manage the context directly without the need for custom code. As initial prerequisite intents are filled, you can create “contexts” to invoke related intents. This simplifies bot design and expedites the creation of conversational experiences.

Generative responses

Integration with the provided large language model (LLM) allows QnABot to disambiguate customer questions by taking conversational context into account, and dynamically generate answers from relevant FAQs, Amazon Kendra search results, or document passages. Generated responses reduces the number of FAQs you need to maintain since concise answers are synthesized from existing documents. Responses can be customized to be short, concise, and suitable for voice channel contact center bots as well as website text bots. Text generation is fully compatible with QnABot's multi-language support allowing users to interact in their chosen languages and receive generated answers in the same language.

> **Note**
>
> By choosing to use the generative responses features, you acknowledge that QnABot on AWS engages third-party generative artificial intelligence (AI) models that AWS does not own or otherwise has 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 may 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. QnABot on AWS is offered to you as "AWS Content" under your agreement with AWS.

8 kHz telephony audio support

Higher fidelity with telephone speech interactions, such as through a contact center application or helpdesk, leveraging the Amazon Lex speech recognition engine which has been trained on telephony audio (8 kHz sampling rate).

Multi-turn dialog

Once an intent has been identified, users will be prompted for information that is required for the intent to be fulfilled (for example, if "Book hotel" is the intent, the user is prompted for the location, check-in date, number of nights, etc.). QnABot on AWS gives you an easy way to build multi-turn conversations for your chatbots. You simply list the slots/parameters you want to collect from your bot users, as well as the corresponding prompts, and the Amazon Lex component takes care of orchestrating the dialogue by prompting for the appropriate slot

Early implementation Content and Slot matching

This new capability supports creating dedicated custom Intents for a QnABot {Item ID}. You can extend QnABot to support one or more related intents. For example, you might create an intent that makes a car reservation, or assists an agent during a live chat or call (via Amazon Connect). For more details, refer to the Intent and slot matching section in the GitHub README.md file.

Custom domain names in QnABot Designer and QnABot Client

Support for using custom domain names for QnABot Designer and Client interfaces. For more details, refer to the Setup custom domain name for QnABot Designer and Client section in the GitHub README.md file.

Importing and exporting questions and answers using CLI

Import and export questions and answers using the AWS QnABot Command Line Interface (CLI) command line. More details, refer to the AWS QnABot Command Line Interface (CLI) section in the GitHub README.md file.

Support for the Kendra Redirect feature

Kendra Redirect - with the Kendra Redirect feature, you can now include a Kendra query within a Item ID. For more details, refer to the Kendra Redirect section in the GitHub README.md file.

Integration with the Canvas Learning Management System (LMS)

Students use their schools' learning management system (LMS) to keep track of their assignments, grades, and their course work. With this integration, students will be able to ask QnABot about their grades, syllabus, enrollments, assignments, and announcements. For more details, refer to the Integration with Canvas LMS section in the GitHub README.md file.

Enhanced functionality for Excel

Updated import functionality to support importing of QnABot questions and answers from a Excel file when uploaded to the Amazon Simple Storage Service (Amazon S3) data folder as well as support for importing session attributes via Excel.

Concepts and definitions

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

intent - An action in response to user input in natural language.

utterances - Spoken or typed phrases that invoke a user's intent.

slots - Input data requested to fulfill the intent.

fulfillment - A mechanism to fulfill the intent.

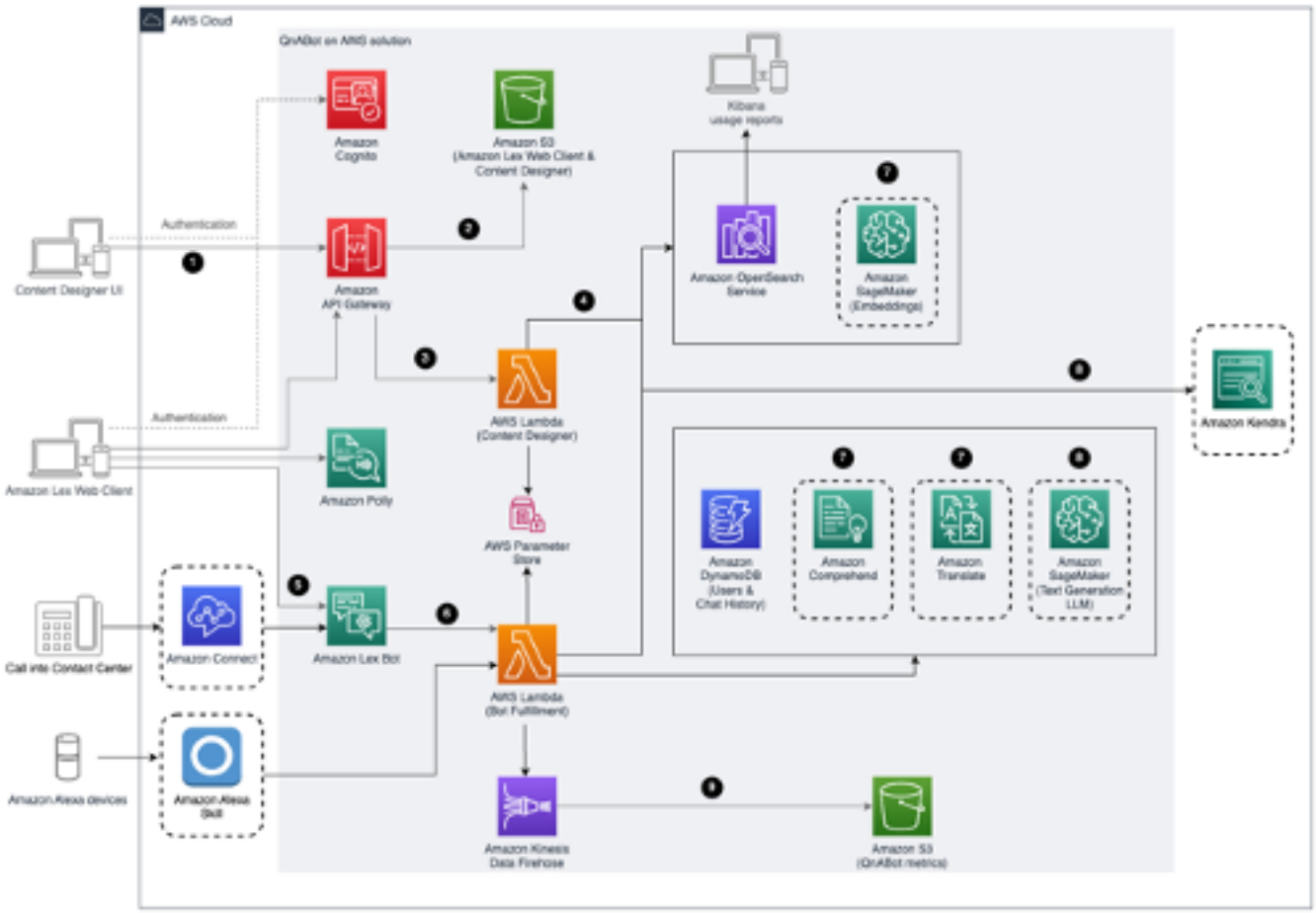
For a general reference of AWS terms, refer to the AWS glossary in *AWS General Reference*.

Architecture overview

This section provides a reference implementation architecture diagram for the components deployed with this solution.

Architecture diagram

Deploying this solution with the default parameters deploys the following components in your AWS account (components with dotted line border are optional).



QnABot on AWS architecture on AWS

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

1. The admin deploys the solution into their AWS account, opens the Content Designer UI or Amazon Lex web client, and uses Amazon Cognito to authenticate.

2. After authentication, Amazon API Gateway 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 in a questions bank index. If using text embeddings, these requests will first pass through a Machine Learning (ML) model hosted on Amazon SageMaker to generate embeddings before being saved into the question bank on OpenSearch.
5. Chatbot users interact with Amazon Lex via the web client UI or Amazon Connect.
6. Amazon Lex forwards requests to the `Bot Fullfilment`AWS Lambda function. Users can also send requests to this Lambda function via Amazon Alexa devices.
7. The `Bot Fullfilment`AWS Lambda function takes the users input and uses Amazon Comprehend and Amazon Translate (if necessary) to translate non-native language requests to the native language selected during deployment, and then looks up the answer in Amazon OpenSearch Service. If using text embeddings, these requests first pass through an ML model hosted on Amazon SageMaker to generate an embedding to compare with text saved in the question bank on OpenSearch.
8. If an Amazon Kendra index is configured for fallback and provided the `Bot Fullfilment` AWS Lambda function forwards the request to Kendra if no matches were returned from the OpenSearch question bank. The text generation LLM can optionally be used to create the search query and to synthesize a response given the returned document excerpts.
9. User interactions with the `Bot Fullfilment` function generate logs and metrics data, which is sent to Amazon Data Firehose then to Amazon S3 for later data analysis.

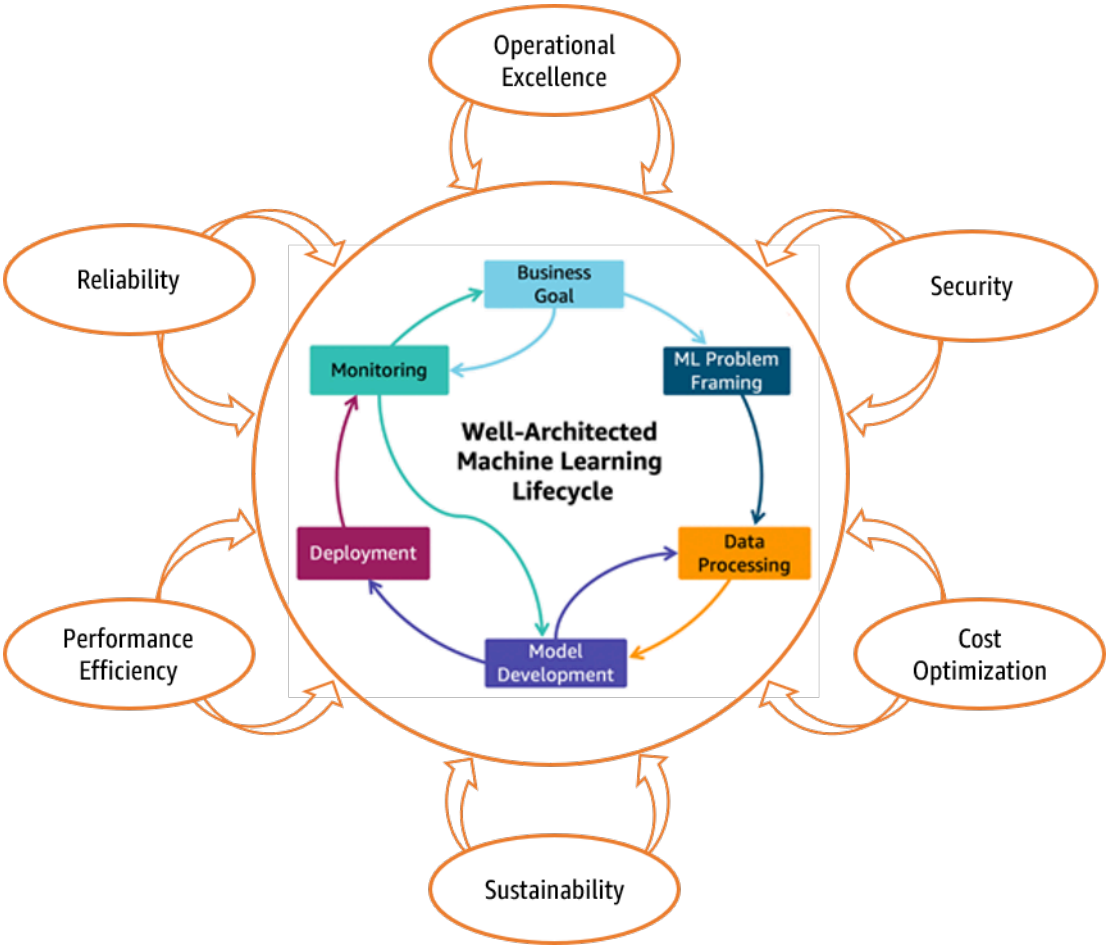
AWS Well-Architected pillars

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.

The machine-learning lifecycle is the iterative process, with instructions and best practices, to use across defined phases while developing an machine learning workload. It adds clarity and structure

for making a machine learning project successful. The Well-Architected machine learning lifecycle superimposes the Well-Architected Framework pillars to each of the machine learning lifecycle phases illustrated in the center of the following figure.



The Well-Architected machine learning lifecycle

Operational Excellence

This section describes how the principles and best practices of the operational excellence pillar were applied when designing this solution.

The QnABot on AWS solution pushes metrics to Amazon CloudWatch at various stages to provide observability into the infrastructure; Lambda functions, AI services, Amazon S3 buckets, and the rest of the solution components. Continuous integration and continuous delivery (CI/CD) and infrastructure deployment are managed in code through AWS Amplify. Data processing errors are added to the Amazon Simple Queue Service (Amazon SQS) queue and displayed in the application layer for user response.

Security

This section describes how the principles and best practices of the security pillar were applied when designing this solution.

- Content Designer UI app users and the Amazon Lex client are authenticated and authorized with Amazon Cognito.
- User permissions to app accounts are managed in the Amazon DynamoDB.
- All inter-service communications use AWS IAM roles.
- All multi-account communications use AWS IAM roles.
- All roles used by the solution follows least-priviledge access. That is, it only contains minimum permissions required so the service can function properly.
- Communication end user and Amazon API Gateway uses Bearer token generated and handed by Amazon Cognito.
- All data storage including Amazon S3 buckets have encryption at rest.

Reliability

This section describes how the principles and best practices of the reliability pillar were applied when designing this solution.

- The solution uses AWS Serverless Services wherever possible (examples Lambda, API Gateway, Amazon S3, and Amazon Lex) to ensure high availability and recovery from service failure.
- The solution protects against state machine definition errors by having automated tests performed on the solution.
- Data processing uses AWS Lambda functions, stored in DynamoDB and Amazon S3, so it persists in multiple Availability Zones (AZs) by default.

Performance Efficiency

This section describes how the principles and best practices of the performance efficiency pillar were applied when designing this solution.

- The solution as mentioned earlier uses serverless architecture throughout this solution.

- The solution can be launched in any Region that supports AWS services in this solution such as: AWS Lambda, Amazon API Gateway, AWS S3, Amazon Lex, Amazon Kendra, and Amazon Comprehend.
- The solution is automatically tested and deployed every day. As well as reviewed by solution architects and subject matter experts for areas to experiment and improve.
- The QnABot on AWS CLI supports the capability to import and export questions and answers from your QnABot setup are designed to reduce IT overhead for maintenance and upkeep.

Cost Optimization

This section describes how the principles and best practices of the cost optimization pillar were applied when designing this solution.

- The solution uses serverless architecture therefore, customers only get charged for what they use.
- The compute layer defaults to AWS Lambda, so it provides pay per use. DynamoDB indexes are selected to reduce throughput cost for queries.
- The solution provides an option to the user to use more advanced AI/ML services. Services such as Amazon Kendra, and Amazon SageMaker are optional and can be turned on or off to reduce the cost for users who don't intend to use these features.

Sustainability

This section describes how the principles and best practices of the sustainability pillar were applied when designing this solution.

- The solution utilizes managed and serverless services, to minimize the environmental impact of the backend services. A critical component for sustainability provided by the solution is maximizing the usage of the AWS AI services. The solution Serverless design (using Lambda and DynamoDB) and the use of managed services (such as AWS Amplify) are aimed at reducing carbon footprint compared to the footprint of continually operating on-premises servers.

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.

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 QnABot on AWS 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. QnABot on AWS 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 QnABot on AWS 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 QnABot on AWS 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 QnABot on AWS 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 QnABot on AWS 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.

AWS services in this solution

The following AWS services are included in this solution:

AWS service	Description
AWS Lambda	**Core.** Provides logic for chatbot interacti ons and provides extension capabilities for Amazon Translate before and after interaction with Amazon Lex.
Amazon Lex	**Core.** 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 understan ding (NLU) to recognize the intent of the text to allow you to build applications with highly engaging user experiences and lifelike conversational interactions.
Amazon OpenSearch Service	**Core.** An open-source search and analytics engine for use cases such as log analytics, real-time application monitoring, and clickstre am 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

AWS service	Description
	to securely ingest data from any source and search, analyze, and visualize it in real time.
Amazon Data Firehose	**Supporting.** A fully managed service for delivering real-time streaming data to destinations such as Amazon Simple Storage Service (Amazon S3), Amazon Redshift, Amazon OpenSearch Service, Splunk, and any custom HTTP endpoint or HTTP endpoints owned by supported third-party service providers, including Datadog, Dynatrace, LogicMonitor, MongoDB, New Relic, and Sumo Logic. Firehose is part of the Kinesis streaming data platform, along with Kinesis Data Streams, Kinesis Video Streams, and Amazon Managed Service for Apache Flink. With Firehose, you don't need to write applicati ons or manage resources. Firehose automatic ally delivers the data the AWS Lambda Bot Fullfillment function sends it to the specified destination. Firehose also transform the data it recieves before delivering it.
Amazon Polly	**Supporting.** Used for Interactive Voice Response systems. It provides the text to speech capabilities of Polly to relay the response back in the voice of choice.

AWS service	Description
Amazon S3	**Supporting.** Used to store and protect any amount of data for a range of use cases, such as data lakes, websites, mobile applications, backup and restore, archive, enterprise applications, IoT devices, and big data analytics. Amazon S3 provides the management features used to optimize, organize, and configure access to data to meet specific business, organizational, and compliance requirements.
AWS Systems Manager Parameter Store	**Supporting.** Provides secure, hierarchical storage for configuration data managemen t and secrets management. It can store data such as passwords, database strings, Amazon Machine Image (AMI) IDs, and license codes as parameter values. It can store values as plain text or encrypted data.
Amazon Translate	**Supporting.** Provides multi-language support to your customer's bot interactions. You can maintain question and answer banks in a single language, yet still offer support customers who interact with the bot in other languages through the use of Amazon Translate.
Amazon Comprehend	**Supporting.** Amazon Comprehend is a natural-language processing (NLP) service that uses machine learning to uncover valuable insights and connections in text.

AWS service	Description
Amazon Connect	**Optional.** Provides an omnichannel cloud contact center. You can set up a contact center in a few steps, add agents who are located anywhere, and start engaging with your customers. If you implement this component , you can create personalized experienc es for your customers using omnichann el communications. For example, you can dynamically offer chat and voice contact, based on such factors as customer preference and estimated wait times. Agents, meanwhile , conveniently handle all customers from just one interface. For example, they can chat with customers, and create or respond to tasks as they are routed to them.
Amazon Kendra	**Optional.** To help fulfill many self-service requests, Amazon Lex retrieves for the most accurate answers from your unstructured data sets hosted in an Amazon Kendra index. You can also use Amazon Kendra to provide semantic search capabilities to your question bank through the use of Kendra FAQs.
Amazon SageMaker	**Optional.** Another option for semantic search provided by QnABot on AWS is using Amazon SageMaker to host an inference endpoint used to generate text embeddings on your queries. These text embeddings transform QnABot's keyword matching system to instead match on the meaning or "semantic similarity" of two different queries based on the similarity of their embedding vectors.

How QnABot on AWS works

QnABot on AWS is powered by the same technology as Alexa. The Amazon Lex component provides the tools you need to tackle challenging deep learning problems, such as speech recognition and language understanding, through an easy-to-use fully managed service.

Amazon Lex integrates with AWS Lambda which you can use to easily trigger functions for execution of your back-end business logic for data retrieval and updates. Once built, your bot can be deployed directly to chat platforms, mobile clients, and IoT devices. You can also use the reports provided to track metrics for your bot. QnABot provides a scalable, secure, easy to use, end-to-end solution to build, publish and monitor your bots.

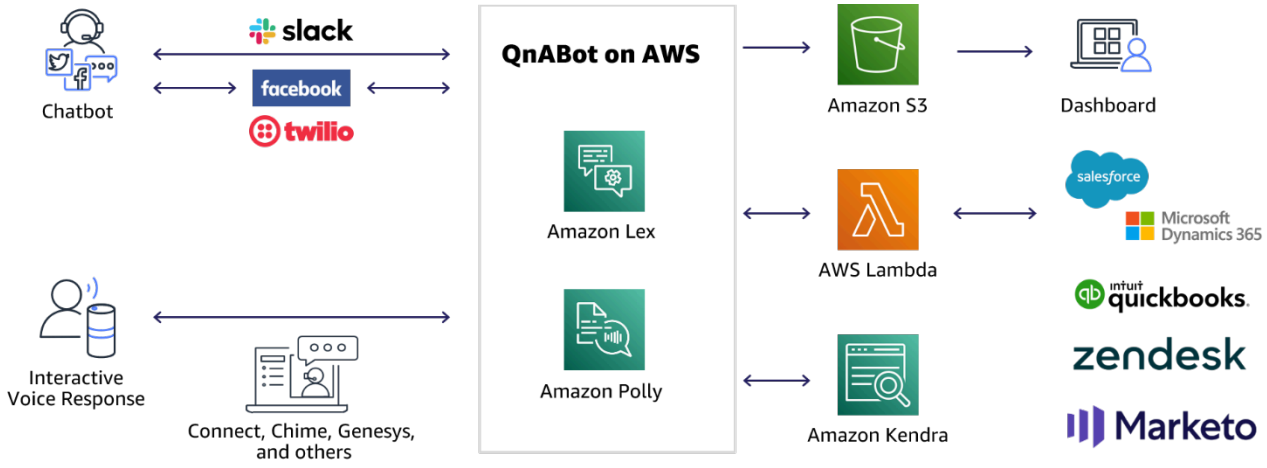
Intelligent contact centers leverage conversational UX engines like Amazon Lex in order to provide proactive service to customers. Amazon Lex uses a deep learning engine that combines automatic speech recognition (ASR) and natural language understanding (NLU) to manage the customer experience. This enables it to be natural and adaptable to customer needs.

Chatbots are the starting point for many organizations, Amazon Lex comes with both voice and text. Amazon Lex has many application integrations for popular messaging platforms such as Slack and Facebook.

For Interactive Voice Response systems you can utilize the Text to speech capabilities of Polly to relay the response back in the voice of your choice.

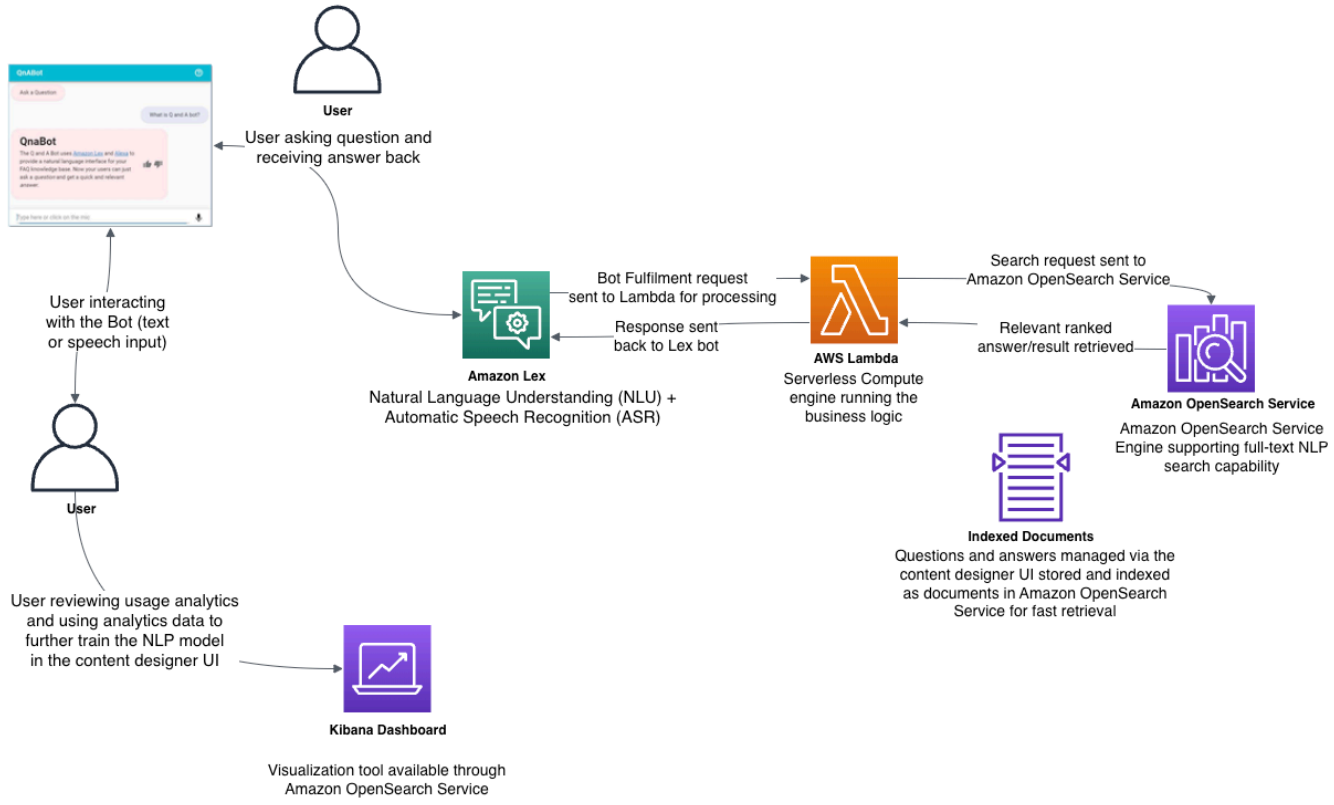
To help fulfill many self-service requests, you can integrate Amazon Lex with your data or applications to retrieve information or use Amazon Kendra to search for the most accurate answers from your unstructured data sets.

The following figure illustrates a reference architecture for how QnABot on AWS integrates with external components.



Reference architecture for QnABot on AWS integrations with external components

The following figure illustrates how Amazon Lex and Amazon OpenSearch Service help power the QnABot on AWS solution.



Solution architecture and data flow

Asking QnABot on AWS questions initiates the following processes:

1. The question gets processed and transcribed by Amazon Lex using Natural Language Understanding (NLU) and Natural Language Processing (NLP) engines.
2. The solution initially trains the NLP engine 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 a 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.
3. 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 QnABot on AWS 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 articles 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 QnABot on AWS. By default, nouns (including proper nouns), verbs, and interjections are used as keywords. Any answer returned by QnABot on AWS 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 QnABot on AWS 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. QnABot on AWS 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 an Amazon OpenSearch query:

```
"query":{
    "bool": {
        "filter": {
            "match": {
                "quniqueterms": {
                    "query": "<LIST_OF_IDENTIFIED_KEYWORDS>",
                    "minimum_should_match":
                          "<ES_MINIMUM_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"
                }
            }
        ]
    }
}
```

Plan your deployment

This section describes the cost, network security, quotas, and other considerations prior to deploying the solution.

Cost

You are responsible for the cost of the AWS services used while running this solution. As of the latest revision, the cost for running this solution in the US East (N. Virginia) Region is approximately **$547.33 per 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. If you intend to integrate Amazon Kendra and Amazon Connect, review the Amazon Kendra pricing and Amazon Connect pricing to adjust your cost estimate accordingly.

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.

Sample cost table

The following table provides a sample cost breakdown for deploying this solution with the default parameters in the US East (N. Virginia) Region for one month.

AWS service	Dimensions	Cost [$ USD]/month
Amazon API Gateway	1,000,000 REST API calls per month	$3.50

AWS service	Dimensions	Cost [$ USD]/month
Amazon Cognito	1,000 active users per month without the advanced security feature	$0.00
Amazon S3	100 GB data transfer + 1,000,000 requests — 100 records x 100 KB from Kinesis	$3.27
AWS Lambda	2,000,000 requests with 200 ms 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
Amazon Data Firehose	100,000 records per month with 100 KB per record	$0.28
Amazon DynamoDB	1 GB 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
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

AWS service	Dimensions	Cost [$ USD]/month
	Total:	**$547.33/month**
Amazon SageMaker Endpoint for text embeddings (optional)	`ml.m5.xlarge` instance running all hours in a month for 1 node	$165.60
	Total:	**$712.93/month**
Amazon SageMaker Endpoint for LLM question answering (optional)	`ml.g5.12xlarge` instance running all hours in a month for 1 node	$5,104.80
	Total: with Embeddings and LLMs	**$5,817.73/month**

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

QnABot on AWS 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 QnABot on AWS 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 allowing 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.

Children Online Privacy Protection Act (COPPA) settings for Amazon Lex

When using this solution to create or update a Amazon Lex chatbots, the parameter regarding COPPA must be set to Yes if the bot's users are subject to COPPA. For more information, refer to Data Protection in Amazon Lex in the *Amazon Lex Developer Guide*.

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 QnABot on AWS 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 with AWS IAM Identity Center

Solution administrators can also federate into the content designer UI and Kibana using single sign-on with AWS IAM Identity Center (successor to AWS IAM Identity Center), where IAM Identity Center 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.

Data storage and protection

The solution uses multiple services to store and protect your data. This solution defaults to the following when storing and protecting the customer's data:

Service / Resource	Default
Cloudwatch Logs	• Default Cloudwatch Logs set to 'Never Expire'
Dynamo DB	• User table stores chat message history (per user) – never expires. • Data fully encrypted at rest (managed by DynamoDB). • PITR enabled by default. • Continuous backups disabled. • Does not store PII data.
OpenSearch Kibana Dashboard Index	• Default expiry set to 30 days.
Amazon S3	• Default `Never Expire` for Metrics Bucket & Export Bucket. • All Buckets are enabled with SSE by default. See here for additional guidance. • Access logging is disabled, customer can configure. See additional guidance here. • Provided guidance on configuring MFA for customers.
Amazon Lex	• Default, logs not enabled. See here for additional guidance. • Encrypting conversation logs can be implemented if needed. See here for additional guidance. • Audio logs are stored in Amazon S3 (default encryption). • COPPA default is No. See here for additional configuration information.

Service / Resource	Default
	• PII reduction capability is implemented on logs.
Key Management Service	• The solution does not enforce the use of CMK. It uses SSE-S3 and AWS Managed Keys for Dynamo DB, Sagemaker, and other relevant services.
Amazon Data Firehose	• SSE Enabled via AWS owned CMK.

Supported AWS Regions

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.

Note

The default LLM model cannot be deployed into the ap-southeast-1 due to unavailability of the **ml.g5.12xlarge** instance type. However, users that are interested in the LLM features can still use the custom Lambda option.

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, see AWS service quotas.

Click one of 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.

AWS CloudFormation quotas

Your AWS account has AWS CloudFormation quotas that you should be aware of when launching the stack in this solution. By understanding these quotas, you can avoid limitation errors that would prevent you from deploying this solution successfully. For more information, refer to AWS CloudFormation quotas in the in the *AWS CloudFormation User's Guiide*.

AWS Sagemaker endpoint quota

The provided LLM Sagemaker API requires an **ml.g5.12xlarge** Sagemaker instance type, which is not enabled in AWS accounts by default and must be requested on a per Region basis. If you are planning on deploying the default LLM Sagemaker API model then you will need to request a quota increase before deploying the solution.

To do so, log into the AWS console, access AWS Service Quotas and search for Amazon SageMaker under the AWS services list. Once selected, search for the quota called **ml.g5.12xlarge for endpoint usage**. At a minimum, you will need to request a quota increase to one (you can request more to accommodate high-volume production deployments).

> **Note**
>
> The **ml.g5.12xlarge** instance type is not available in the `ap-southeast-1 region`.

Amazon DynamoDB backups

Backups for Amazon DynamoDB Tables are not setup by default. If you require backups for the data that is stored in Amazon Dynamo DB Tables, refer to Backing Up a DynamoDB Table in the *Amazon DynamoDB Developer Guide*.

For recovery of backed up data, refer to Restoring a DynamoDB Table from a Backup in the *Amazon DynamoDB Developer Guide*. Alternatively, you can use Point-in-Time Recovery for DynamoDB as your backup and recovery method.

Amazon OpenSearch Service log publishing

The OpenSearch service in this solution does not automatically log publishing to Amazon CloudWatch. To turn on log publishing, refer to Monitoring OpenSearch logs with Amazon

CloudWatch logs in the *Amazon OpenSearch Service Developer Guide*. You can also refer to Monitoring audit logs in Amazon OpenSearch Service for monitoring the audit logs for auditing the Amazon OpenSearch Service domain's fine-grained access control.

Deploy the solution

This solution uses AWS CloudFormation templates and stacks to automate its deployment. The CloudFormation template(s) to describes 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. 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

Step 1: Launch the stack.

- Launch the AWS CloudFormation template into your AWS account.
- Enter values for required parameters.
- Review the template parameters, and adjust if necessary.

Step 2: Launch the chatbot content designer.

- Update password and sign in to the content designer.

Step 3: Populate the chatbot with your questions and answers.

- Enter question and answer pairs.

Step 4: Interact with the chatbot.

- Interact with the chatbot through voice or text.

AWS CloudFormation template

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

View template

qnabot-on-aws-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 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..

Note

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

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

Step 1: Launch the stack

This automated AWS CloudFormation template deploys the QnABot on AWS 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 section in this guide, and reference to the pricing webpage for each AWS service used in this solution.

1. Log in to the AWS Management Console and select the **Launch solution** button to launch the `qnabot-on-aws-main.template` AWS 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 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 Services by Region 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 QnABot on AWS content designer.
`Username`	*`<Requires input>`*	This username will be used to sign in to QnABot on AWS content designer console.

Parameter	Default	Description
PublicorPrivate	PUBLIC	Choose whether access to the QnABot on AWS client should be publicly available or restricted to users in the QnABot on AWS UserPool.
Language	English	Language is chosen by the user during the deploymen t that will select which OpenSearch language analyzer to use. This will act as the primary Language whenever possible.
Amazon Kendra Integration		
DefaultAmazon KendraIndexid	Optional input	Index ID of an existing Amazon Kendra index, used as the default index for QnABot on AWS Amazon Kendra integration. You can use the QnABot on AWS content designer to reconfigure the Amazon 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.

Parameter	Default	Description
ElasticSearchEBSVolumeSize	10	The size in GB of the OpenSearch node instances.
Encryption	ENCRYPTED	Allows encryption at rest for S3 and Amazon OpenSearch Service, and provisions m6g.large .Amazon OpenSearch Service instances ENCRYPTED — is recommended for production environments. Selecting the UNENCRYPTED configuration provisions lower cost 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		

Parameter	Default	Description
LexV2BotLocaleIds	*<Requires input>*	Languages for QnABot on AWS voice interacti on 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.
Semantic Search and Embeddings		
EmbeddingsApi	DISABLED	Enable semantic search capability using a pre-train ed model. Selecting the SAGEMAKER automatic ally provisions a Sagemaker endpoint. Selecting the LAMBDA option allows for configuration with other models.
SagemakerInitialIn stanceCount	1	Required for SAGEMAKER EmbeddingsApi types. Sets the number of instances to deploy. Set to 0 for serverles s inference.
EmbeddingsLambdaArn	*<Requires input>*	Required for LAMBDA EmbeddingsApi types. The ARN of the Lambda function that calls the LLM model.

Parameter	Default	Description
EmbeddingsLambdaDimensions	4096	Required for LAMBDA EmbeddingsApi types. Provides the number of dimensions for embeddings returned from the Lambda function.
LLM Integration		
LLMApi	DISABLED	Enable question disambiguation and generative responses using an LLM model. Selecting the SAGEMAKER automatically provisions a Sagemaker endpoint with LLM model. Selecting the LAMBDA option allows for configuration with other LLM's.
LLMSagemakerInstanceType	ml.g5.12xlarge	Required for SAGEMAKER LLMApi types. Check account and region availability through the Service Quotas service before deploying.
LLMSagemakerInstanceCount	1	Required for SAGEMAKER LLMApi types. Sets the number of instances to deploy.
LLMLambdaArn	*<Requires input>*	Required for LAMBDA LLMApi types. The ARN of the Lambda function that calls the LLM model.

Parameter	Default	Description
Other parameters		
`LexBotVersion`	`LexV1 or LexV2`	Amazon Lex version to use for QnABot on AWS. Select **LexV2 Only** to install QnABot on AWS in AWS Regions where LexV1 is not supported.
`InstallLexResponseBots`	`True`	Required for using the elicit response feature.
`FulfillmentConcurrency`	`0`	The provisioned concurrency for the fulfillment Lambda function.
`XraySetting`	`False`	Configure Lambdas with X-Ray enabled.

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, and so on.

> **Note**
>
> In addition to the primary AWS Lambda function 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 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 deploying the stack, you will receive an email at the email address listed in the deployment parameters with the subject *QnABot on AWS Signup Verification Code*. This email contains a generated temporary password that you can use to sign 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.

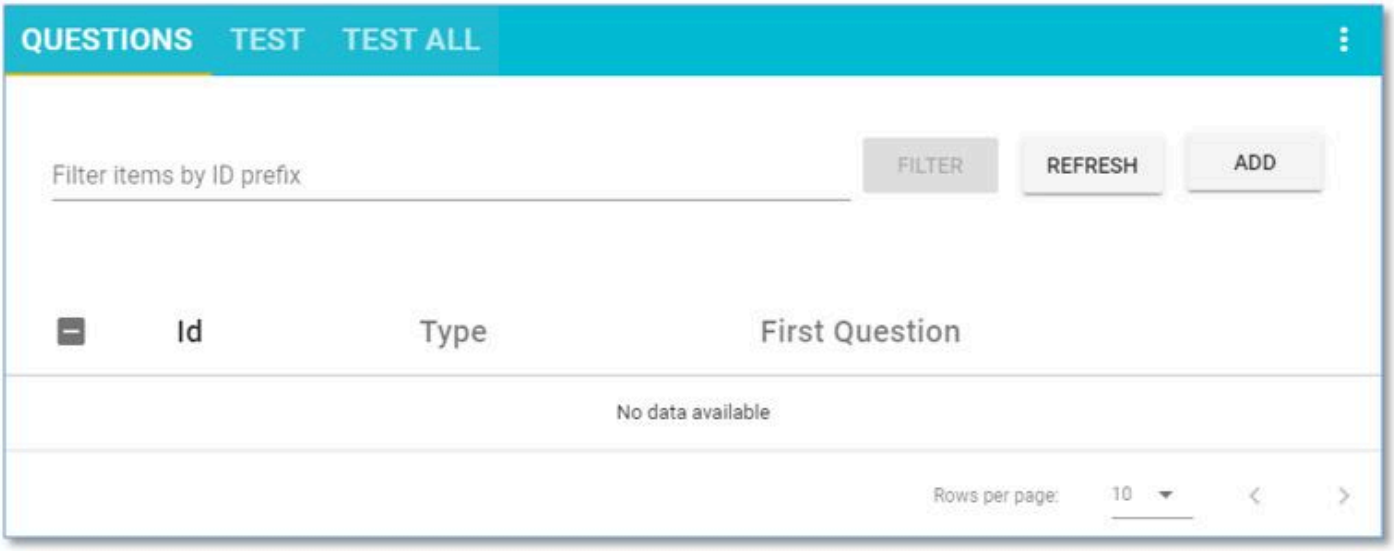
1. Open the verification email and select the link, or alternatively, you can select the **ContentDesignerURL** link from the CloudFormation console **Outputs** tab. The designer opens in a separate browser tab.
2. Log in with your username and temporary password.
   - Enter the **username** that you specified in the deployment parameters.
   - Enter the temporary **password** from the verification email.
3. Follow the prompts to change your password and sign 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: Populate the chatbot with your questions and answers

You must create or upload question and answer data through the content designer before sharing the QnABot on AWS 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 primary CloudFormation stack.
2. Enter the administrator username you provided when you launched the stack and your new password.



QnABot on AWS 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.**

   > **Note**
   >
   > We recommend that you always import the QnaUtility example of questions set as it enables support of no_hits, no_verified_identity, help, repeat and thumbs up and down feedback.

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 (⋮). The following table shows 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 your existing documents and edit them directly in the console. You can also

Id	Question	Answer
		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.

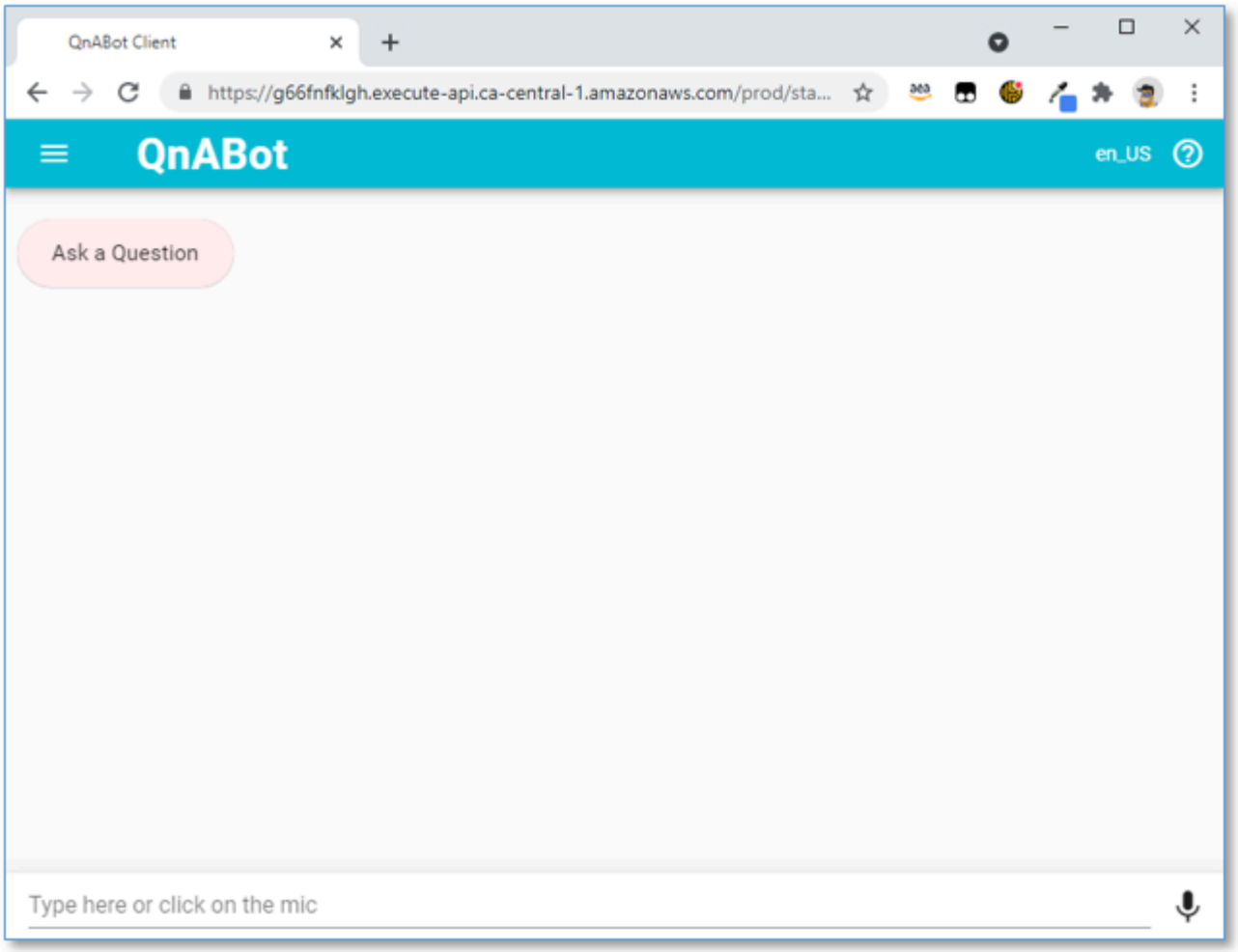
Id	Question	Answer
Admin.004	How do I troubleshoot and fix problems with Q and A Bot.	Use the Content Designer test tool to test a question, and check what items are returned, ranked in order of score. If the desired item does not have the highest score, then add the question to the item and run the test again. The desired item should now have the highest score. Ensure that you aren't creating items with duplicate questions to avoid unpredict able 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 QnABot on AWS 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 QnABot on AWS stack, choose **Output**, and then select the link to the **ClientURL**. Alternatively, launch the client by choosing **QnABot on AWS 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 QnABot on AWS chat window opens.



QnABot on AWS 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 QnABot on AWS 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 QnABot on AWS. 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 QnABot on AWS content designer, open the tools menu (☰), and choose **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.

When testing the skill, invoke the skill with the invocation name before asking questions and answers. For example, if your invocation name is my qnabot, in the Alexa skill test console, first say "Open my Q and A Bot." Alexa will reply with "Hello, please ask a question," then you can ask your QnABot a question.

> **Note**
>
> If you want to publish your new QnABot on AWS 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.

Update the solution

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

1. Log in to AWS CloudFormation console, select your existing QnABot on AWS CloudFormation stack, and choose **Update**.
2. Select **Replace current template**.
3. Enter the Amazon S3 URL:
   - https://solutions-reference.s3.amazonaws.com/qnabot-on-aws/latest/qnabot-on-aws-main.template
   - https://solutions-reference.s3.amazonaws.com/qnabot-on-aws/latest/qnabot-on-aws-vpc.template
4. Under **Parameters**, review the parameters for the template and modify them as necessary. Refer to Step 1: Launch the stack 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**
>
> **For those Upgrading from v5.4.X to v5.4.2**
> If you are upgrading from a deployment with LLMApi set to SAGEMAKER then set this value to DISABLED before upgrading to v5.4.2. After upgrading, return this value back to SAGEMAKER.

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 QnABot on AWS solution from the AWS Management Console or by using the AWS Command Line Interface.

Using the AWS Management Console

1. Log in to the AWS CloudFormation console.
2. 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*. Optionally, you can use the AWS CloudShell service to run AWS CLI commands. After confirming that the AWS CLI is available, run the following command:

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

Advanced Setup

This section provides detailed instructions on how to setup QnABot to perform the following tasks:

- Adding images to your answers
- Displaying rich text answers
- Using SSML to control speech synthesis
- Using topics to support follow-up questions and contextual user journeys
- Adding buttons to the web UI
- Integrating Handlebars templates
- Setting Amazon Lex session attributes
- Specifying Lambda hook functions
- Using keyword filters for more accurate answers and customizing “don’t know” answers
- Tuning, testing, and troubleshooting unexpected answers
- Semantic question matching using LLM text embeddings
- Importing and exporting chatbot answers
- Modifying configuration settings
- Integrating Amazon Kendra
- Using automatic translation
- Configuring the chatbot to ask the questions and use response bots
- Connecting QnABot on AWS to an Amazon Connect call center
- Configuring intent and slot matching
- Setting up a custom domain name for QnABot Designer and Client
- Using QnABot on the AWS Command Line Interface (CLI)

Integration with Canvas Learning Management System (LMS)

Students use their schools' Canvas LMS to keep track of their assignments, grades, and working through their course work. To make it easier for students to stay on track and also have easy access to a knowledge base, and help with their learning progress, you can integrate the open-source QnABot on AWS solution with Canvas LMS, and support students with in-the-moment

support. With this integration, students are able to ask the chatbot about their grades, syllabus, enrollments, assignments, and announcements.

Prerequisites

There are a few prerequisites to get started with the setup:

1. Setting up up Canvas LMS requires a running Canvas LMS environment (on-premises or /AWS environment). If you do not have Canvas LMS, you can install by following the instructions on GitHub repository.
2. Set up the open-source QnABot on AWS solution deployed in your AWS environment. If you do not have this setup or are running an older version of QnABot on AWS, you can install or upgrade by following the QnABot on AWS implementation guide.
3. Set up a companion Web UI for the chatbot. You can deploy this using the open source Lex-Web-UI project in your AWS account by following the steps outlined in this blog post.
   - During this setup, set the **EnableLogin** setting to true. This enables authentication in the chatbot and connect to an Identity provider.
   - For **BotName** and **BotAlias**, use the bot name and bot alias obtained from the QnABot on AWS solution deployment outputs.

Creating and storing the Canvas API access token

The QnABot on AWS solution uses the Canvas API to integrate with Canvas LMS. To configure the QnABot on AWS solution, follow these steps:

1. Create a new Canvas API access token. For more details on how to create a Canvas API access token, refer to How do I manage API access tokens as an admin?
2. Store the Canvas API access token in AWS Secrets Manager. With Secrets Manager you can replace hardcoded credentials in your code, including passwords. To retrieve the secret programmatically, you make an API call to Secrets Manager. This helps ensure the secret can't be compromised by someone examining your code, because the secret no longer exists in the code.
   a. Open to the Secrets Manager console. Use the same AWS region where you deployed the QnABot on AWS solution.
   b. Choose Store a new secret.
   c. For **Key** name, choose the key **API_Token**.
   d. For **Value**, copy and paste the Canvas API access token value that you created earlier.

e. Enter a descriptive **Secret name** and **Description**. Start the name with the letters **qna-**. For example, **qna-CanvasAPIKey**.

Configure QnABot on AWS settings

After you have deployed the QnABot on AWS solution, you will have access to the **QnABot Designer** console, which allows you to create and manage your knowledge bank of questions and answers.

1. Open the link that you received in your email and sign in to your ot DeQnABsigner console.
2. Choose the menu located in the top left corner of the designer console. The **Tools** option list appears.
3. Scroll to the bottom of the page and choose **ADD NEW SETTING**. Use this to store the Secrets Manager key name that you created in the above steps, so QnABot can know how to connect to Canvas LMS. Enter the New Setting values:
   - **Name** – Enter `CanvasLMS_APIKey` as the name.
   - **Value** – Use the name of the Secrets Manager key that you created in the above steps for storing the Canvas API key value. For example, `qna-CanvasLMSAPIKey`.
   - Choose **ADD** to add the new QnABot setting.
4. Create another setting.
   - **Name** – Enter `CanvasLMS_DomainName` as the name.
   - **Value** – Use the value of your Canvas endpoint. For example, `https://lms.myschool.edu`
   - Choose **ADD**.
5. Update the **IDENTITY_PROVIDER_JWKS_URLS** setting to add trusted **Identit** .**Providers**. For example: from your Lex-Web-UI Cloudformation Outputs, using the *CognitoUserPoolPubKey* value.
6. Scroll to the bottom of the Settings page and choose **Save** to update the setting.

Set up authentication

As part of the prerequisite setup, we set up the Lex-Web-UI (a companion UI solution for the chatbot) and configured the solution with the QnABot solution. The deployment sets up an Amazon Cognito User Pool to support authentication. We will now extend this User Pool to add a test student user and test out the chatbot flow.

1. Open https://console.aws.amazon.com/.
2. Navigate to the Amazon Cognito service.
3. Select **Manage User Pools**. Two user pools have already been created when you followed the steps in Setup Prerequisites earlier in this document. We will use is the Lex-Web-UI user pool.
4. Select the **Lex-Web-UI** user pool and create a test student user. Also use an {email address} as created in the Canvas LMS for the test student user.

> **Note**
>
> In this example, we are creating the user manually in Amazon Cognito. This manual user creation step is not be needed if you want to use single sign-on to access Canvas LMS.
> For more information on setting up Canvas LMS using single sign-on, refer to How do I configure SSO settings for my authentication provider?
> In this example, we are using username as the matching attribute with `sis_login_id`in Canvas LMS.
> If you want to federate single sign-on into the QnABot Designer UI and Kibana using IAM Identity Center, refer to Using QnABot Designer with IAM Identity Center.

Import Canvas questions

In the **QnABot Designer**, select the menu link on the top left and choose **Import**. From the **Examples/Extensions** section, select **Load for CanvasLMSIntegration** to load sample Canvas questions.

Amazon Lex Rebuild

Once you have loaded the questions, choose **Edit** from the **Tools** menu and choose LEX REBUILD from the top right edit card menu (⋮). This will re-train Amazon Lex using the newly added questions as training data.

Testing the experience

Launch the WebAppUrl URL as available in the Lex-Web-UI AWS CloudFormation Output and Login to the chatbot from the menu option. Use the test student Canvas LMS credential that you created in the earlier steps to login and test the setup.

Type or speak the below question(s) and see how the chatbot responds back with an answer.

- Canvas menu
- Do I have any announcements?
- Tell me about my syllabus
- Do I have any assignments due?
- What courses have I enrolled in?
- More info about my course
- What are my grades?

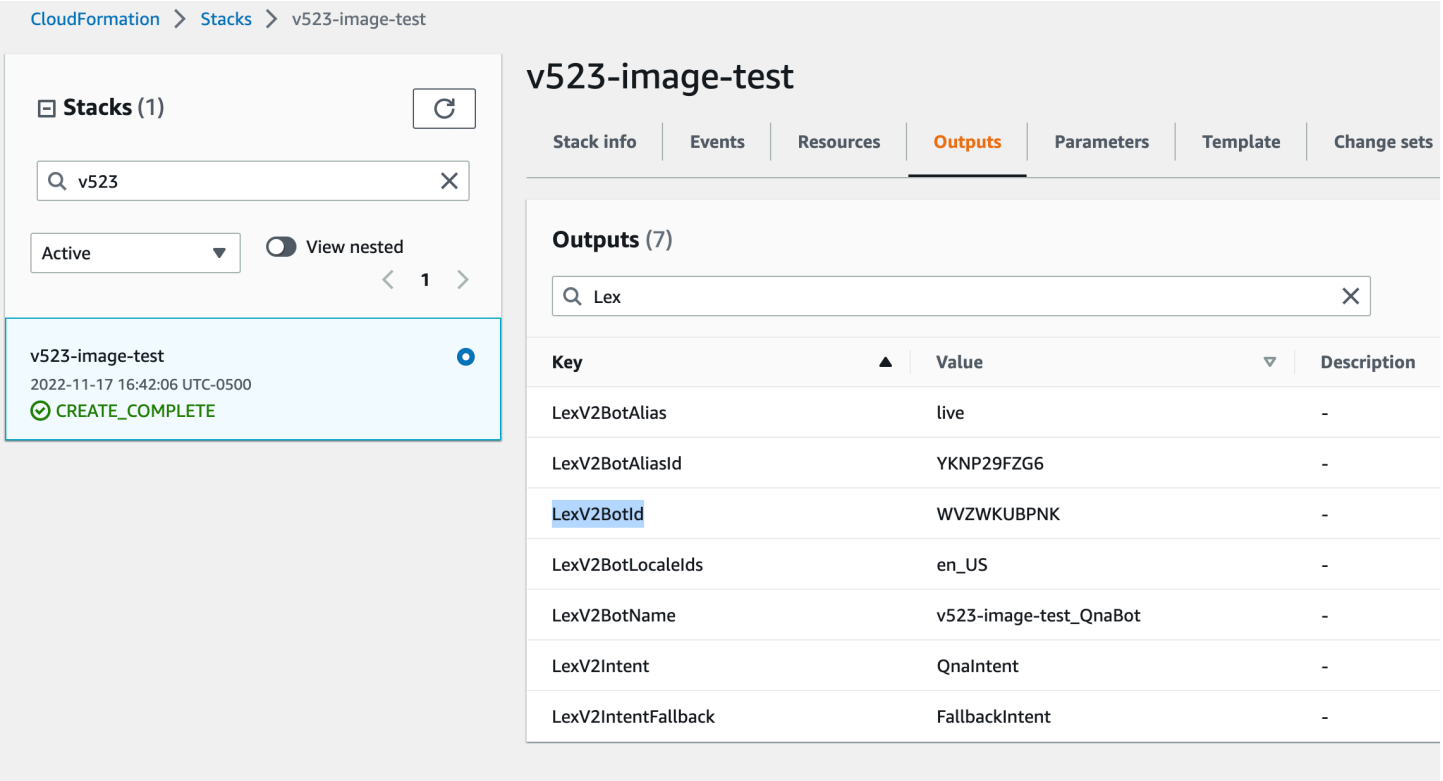
> **Note**
>
> This early example implementation supports English (en_US) language.

Deploy a custom web UI for your chatbot

QnABot on AWS includes a built-in web UI that you can use as is. Refer to section Interact with the chatbot earlier in this guide.

After you finish building your QnABot, you can separately deploy the Amazon 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 and the sample Amazon Lex web interface in the QnABot Github repository.

To deploy the Amazon Lex web UI, you must know the IDs of the Amazon Lex bots. You can find the IDs in the **Outputs** tab of the QnABot CloudFormation template.



Amazon Lex bot IDs in the CloudFormation Outputs tab

Canvas API reference

The following Canvas APIs are being used for this integration:

- User profile – To support authentication, and greeting the user.
- Grades – Student can ask questions such as *"How did i do in my Math course?"*. This supports the overall grade information (out of 100) which is aggregated by course (not by assignments).
- Course – Students can ask questions such as: *"What are my assignments for Biology 101"*
- Syllabus – To access syllabus information. The output of this is a URL to the syllabus. Student can ask about their syllabus by asking *"Tell me about my syllabus"*.
- Enrollment – Students can ask questions such as: "What courses am i enrolled in", "what courses have i signed up for".
- Announcements – anything sent by the teacher to student(s) such as: "*You have a test coming up.*". Student can ask by saying "*Do I have any announcements?*"

This integration uses the canvasapi python library to access information from Canvas LMS.

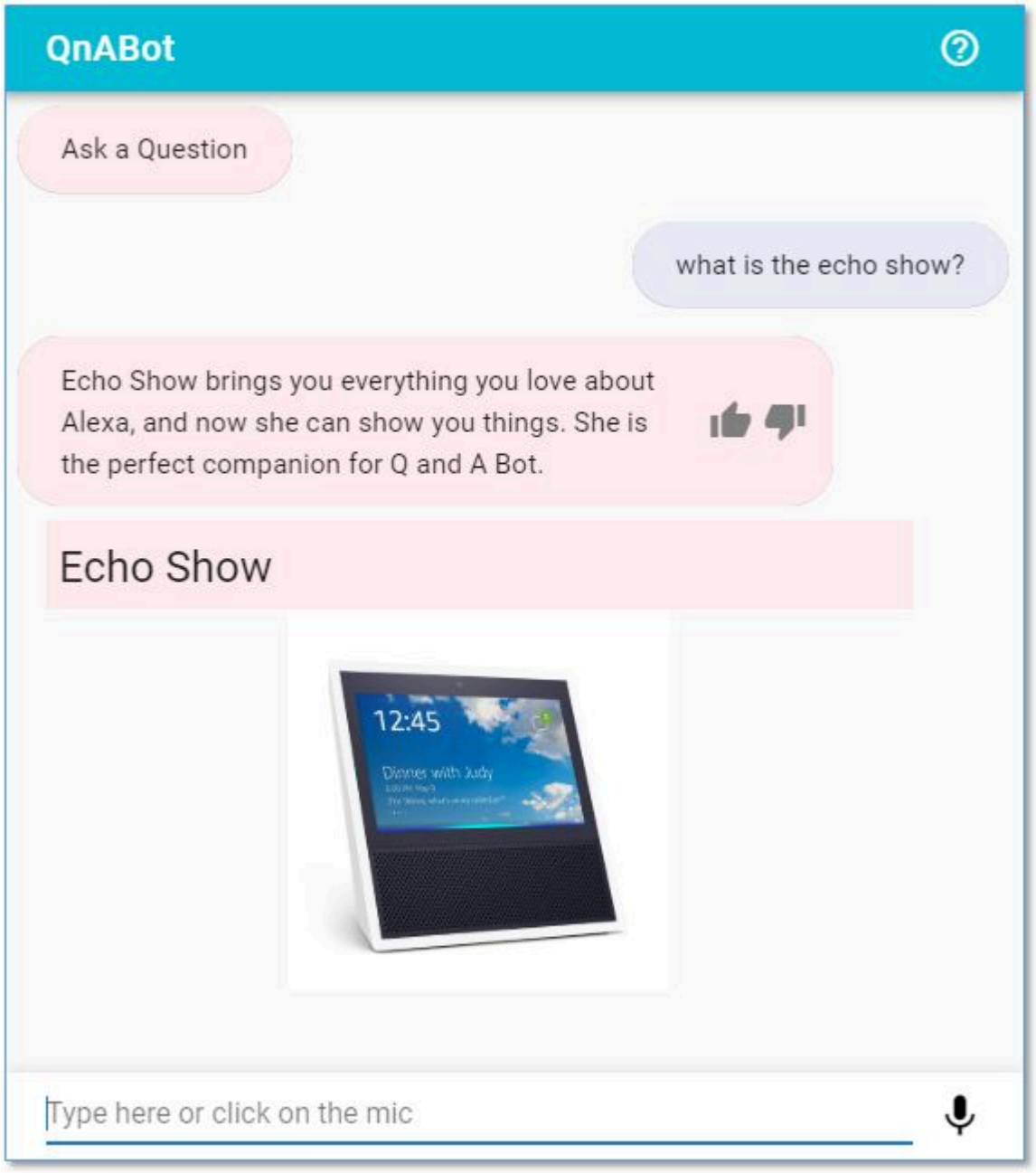
Adding images to your answers

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. For example, you can use images to display maps, diagrams, or photographs to depict places and products relevant to a 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:
   - Card Title: `Echo Show`
   - 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.



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.

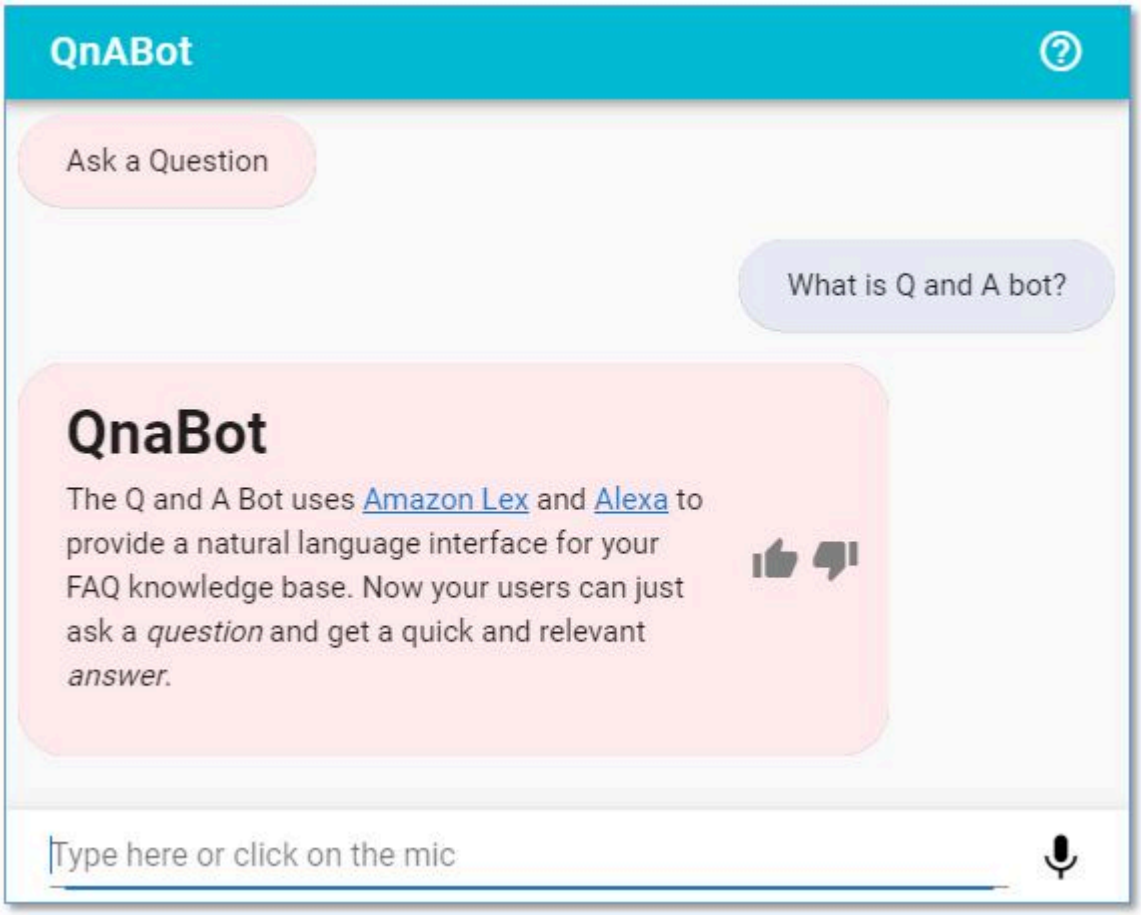
Displaying rich text answers

QnABot on AWS 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.



Example Markdown text

4. QnABot on AWS also supports inline HTML in the markdown field. Choose **ADD** to create a new item in HTML
   - Enter ID: `FireTV.001`
   - Enter question: `What is Amazon Fire TV?`
   - 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 QnABot on AWS into your living room!`
   - 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 QnABot on AWS into
your living room!
<iframe src="https://www.youtube.com/embed/OE4MrFx2XCs"></iframe>
```

5. 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 selecting the**Advanced** section and entering the following text in the SSML Answer field:

```
<speak>AWS <sub alias="Q and A">QnA</sub> Bot is <amazon:effect
name="drc">great</amazon:effect>. <sub alias="Q and A">QnA</sub>
Bot supports <sub alias="Speech Synthesis Markup Language ">SSML</sub>
using Polly's neural voice. <prosody rate="150%">I can speak very fast</prosody>,
<prosody rate="75%">or very slowly</prosody>. <prosody volume="-16dB">I can speak
quietly</prosody>, <amazon:effect name="drc">or speak loud and clear</
amazon:effect>.
I can say <phoneme alphabet="ipa" ph="tə#m##tə#">tomato</phoneme> and tomato.
Visit docs.aws.amazon.com/polly/latest/dg/supportedtags for more information.</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 response.
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`
   - Enter topic: `EchoShow`

Using topics to support follow-up questions and contextual user journeys

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 modification.
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`
   - Enter topic: `EchoShow`
6. Choose **CREATE** to save the item.
7. Choose **ADD** to create a new item for our next follow-up question. Enter the following values:
   - Enter ID: `QnABot on AWS.Cost`
   - Enter question: `How much does it cost?`
   - Enter answer:`Q and A Bot is priceless`
   - Enter topic*:* QnABot on AWS
8. Choose **CREATE** to save the item.
9. 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?"*

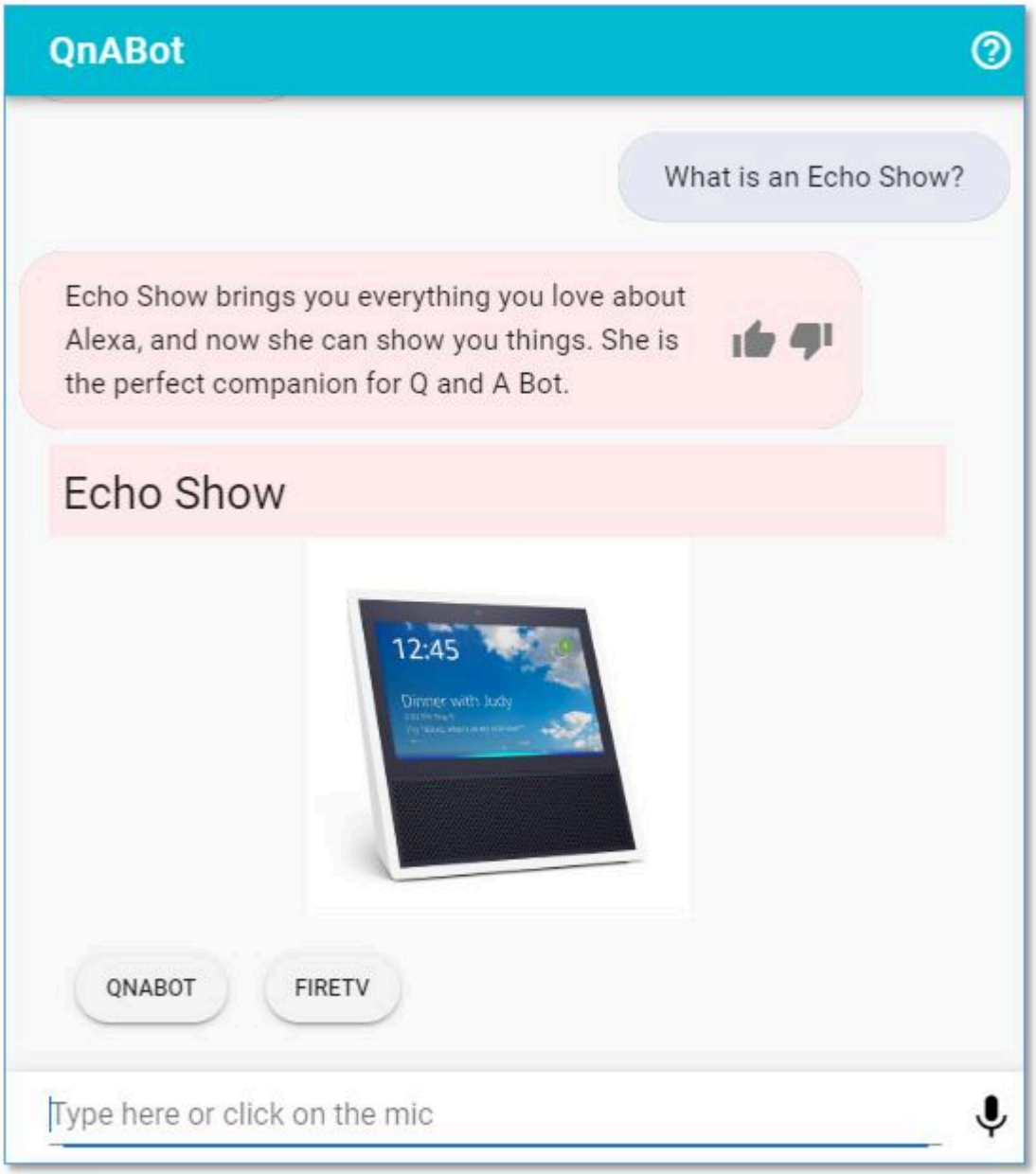
     This question changes the Topic to: 'QnABot on AWS'.
   - *"How much does it cost?"*

     The new topic allows the QnABot on AWS to respond with the correct answer

Adding 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, enter a Card Title.
3. Under **Lex Buttons**.
   - Enter Display Text: `AWS QnABot`
   - Enter Button Value: `What is Q and A bot?`
4. Select **ADD LEX BUTTON** to add another button.
   - Enter Display Text: `FireTV`
   - Enter Button Value: `What is Amazon Fire TV`?
5. Select **UPDATE** to save the item with your new buttons.
6. Use the Web UI to ask: "*What is an Echo Show?*"



Example buttons for sending the next question

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

> **Note**
>
> When integrating with Connect, QnABot maps to the Connect List Picker Template . The client sets limits on the number of characters in a field and enforces formatting using text from the QnABot plaintext response. You may need to modify the QnABot plaintext response to accodomdate these limitations with the Connect chat client.

Integrating 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 Designer, 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.

Setting Amazon Lex session attributes

Starting with Version 5.1.0, the QnABot on AWS solution provides support for a question in the designer UI to set an Amazon Lex session attribute.

In prior versions (v5.0.0 and earlier), using handlebars in an answer would set a session attribute. For example, the following code can set an attribute called `attributeName` to the value `attributeValue`.

```
"{{setSessionAttr 'attributeName' 'attributeValue'}}"
```

In v5.1.0 and later you can optionally use a question in the designer UI to define a set of name/value pairs as session attributes when the answer is returned. There is a field to set a name/value pair, an **Add** button, and a **Delete** button.

The attribute name can be a simple name like `myAttribute` or a complex name like `myAttribute.subAttribute`. You can also use the *dot* notation to set an attribute several levels deep.

> **Note**
>
> Avoid using appContext or qnabotcontext as attribute names. Setting these may have adverse effects on the system.

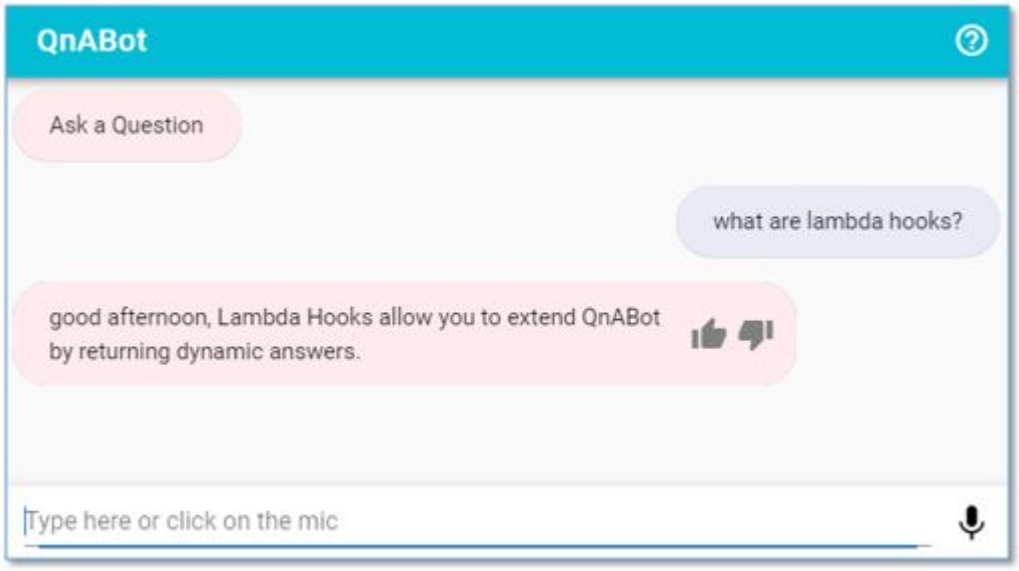
Specifying Lambda hook functions

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, QnABot on AWS 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.

QnABot on AWS 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*'.



Example Lambda hook function

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.

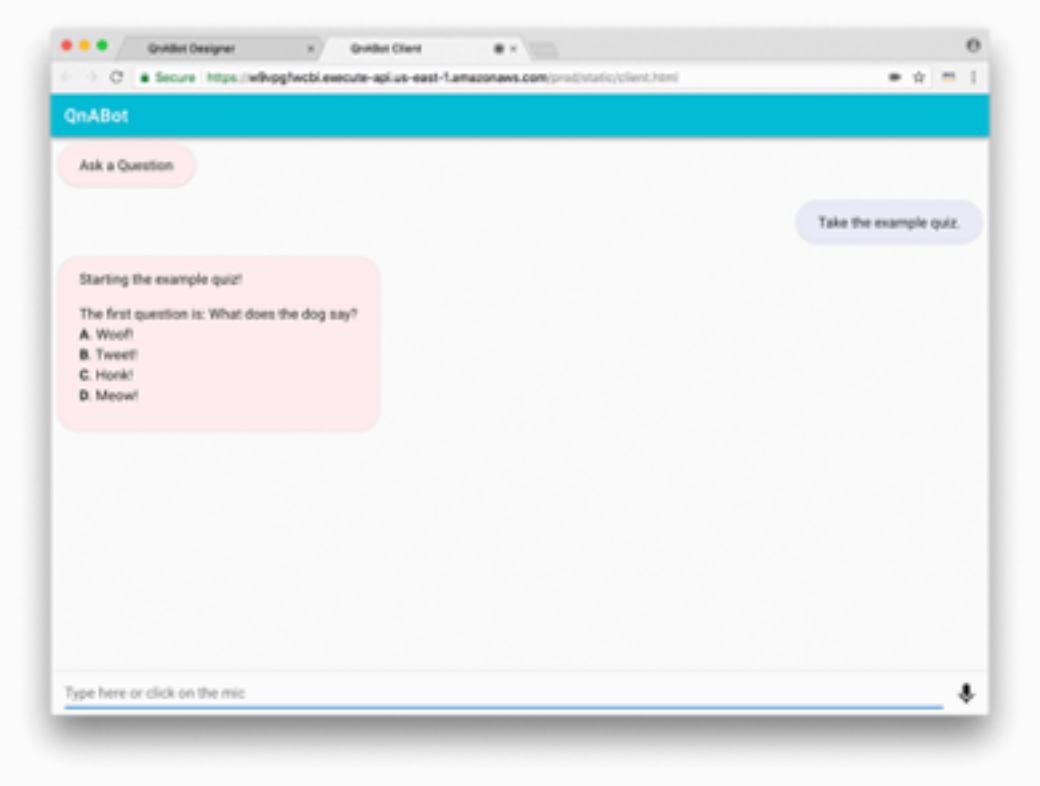
For more information about how you can package Lambda hooks, refer to Extensions README in the QnABot on AWS GitHub repository.

Quizzes

The solution's content designer allows you to set up QnABot to use its Questionnaire Bot feature to create simple quizzes for your users.

QnABot on AWS comes with a simple quiz 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 **Quiz** example.
3. After the import job has completed, return to the edit page, and examine the items **ExampleQuiz & QuizEntry**
4. Select **Start the quiz** or **Take the example quiz** to begin the quiz.



Example Quiz

For more detailed information about how you can create and customize quizzes, see QnABot Workshop.

Using keyword filters for more accurate answers and customizing “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 QnABot on AWS. By default, nouns (including proper nouns), verbs, and interjections are used as keywords. Any answer returned by QnABot on AWS 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 you have selected a non-English language for your deployment, then it will use Amazon Translate to translate these keywords back to the language your user is using for interaction. If QnABot on AWS 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 QnABot on AWS 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 by following these steps:

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 unexpected answers

Tuning answers using the content designer

By default, QnABot on AWS attempts to match an end user's question to the list of questions and answers stored in Amazon OpenSearch Service. QnABot on AWS 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 QnABot on AWS 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 QnABot on AWS. 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 the **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 QnABot on AWS a question, it is processed and transcribed by either Amazon Lex or Amazon Alexa using an Automatic Speech Recognition (ASR) engine. QnABot on AWS 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 QnABot on AWS 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

QnABot on AWS 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

QnABot on AWS 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 QnABot on AWS 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 QnABot on AWS usage and user feedback

The solution logs everything that end users say to the chatbot. Amazon 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.
   - 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, QnABot on AWS responds

with the newly imported *CustomNoMatches* response—indicating that it doesn't know the answer.

- From the web UI, speak or type "*Thumbs down*", or select the thumbs down icon beside the answer.

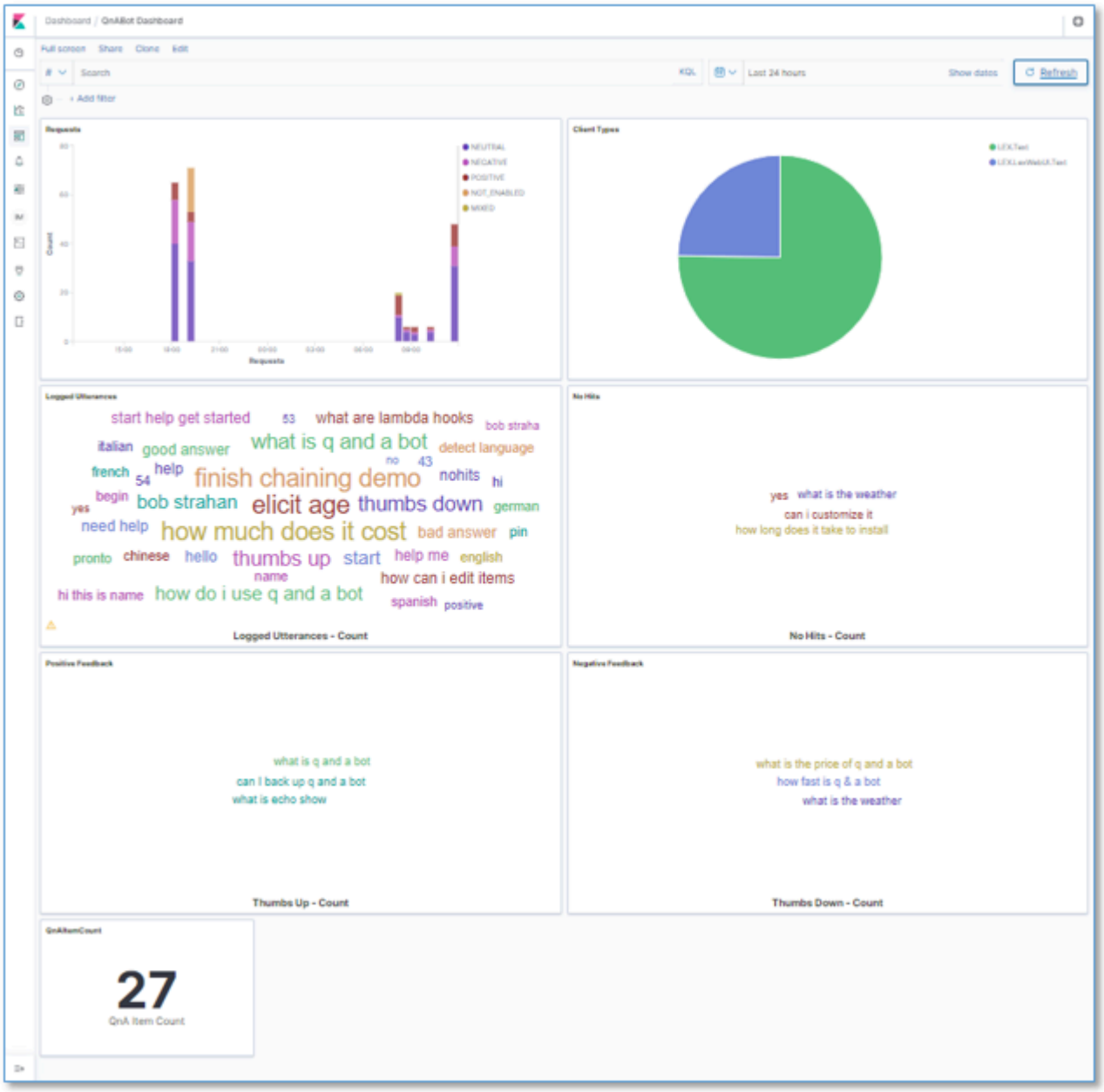
The solution publishes *Thumbs down* feedback messages to the Amazon Simple Notification Service (SNS) topic identified by the **FeedbackSNSTopic** on the **Outputs** tab of the CloudFormation stack. To learn how to subscribe to the SNS topic, and receive a message from the each time a user provides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

If you have selected an non-English language for your QnABot deployment and have multi-language enabled then you should do the following steps:

1. Use the AWS translate API to translate Thumbs up and Thumbs down to your deployment language if it is not English.
2. Add the translation of the Thumbs up and down as a question in your QnABot deployment.
3. Go to the content designer and on the top left, then select **Settings**.
4. Find the PROTECTED_UTTERANCES variable and insert that phrase in by adding a comma, then enter the translation.

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 QnABot on AWS Dashboard to visualize usage history and sentiment, all logged utterances, no hits utterances, positive user feedback, and negative user feedback.

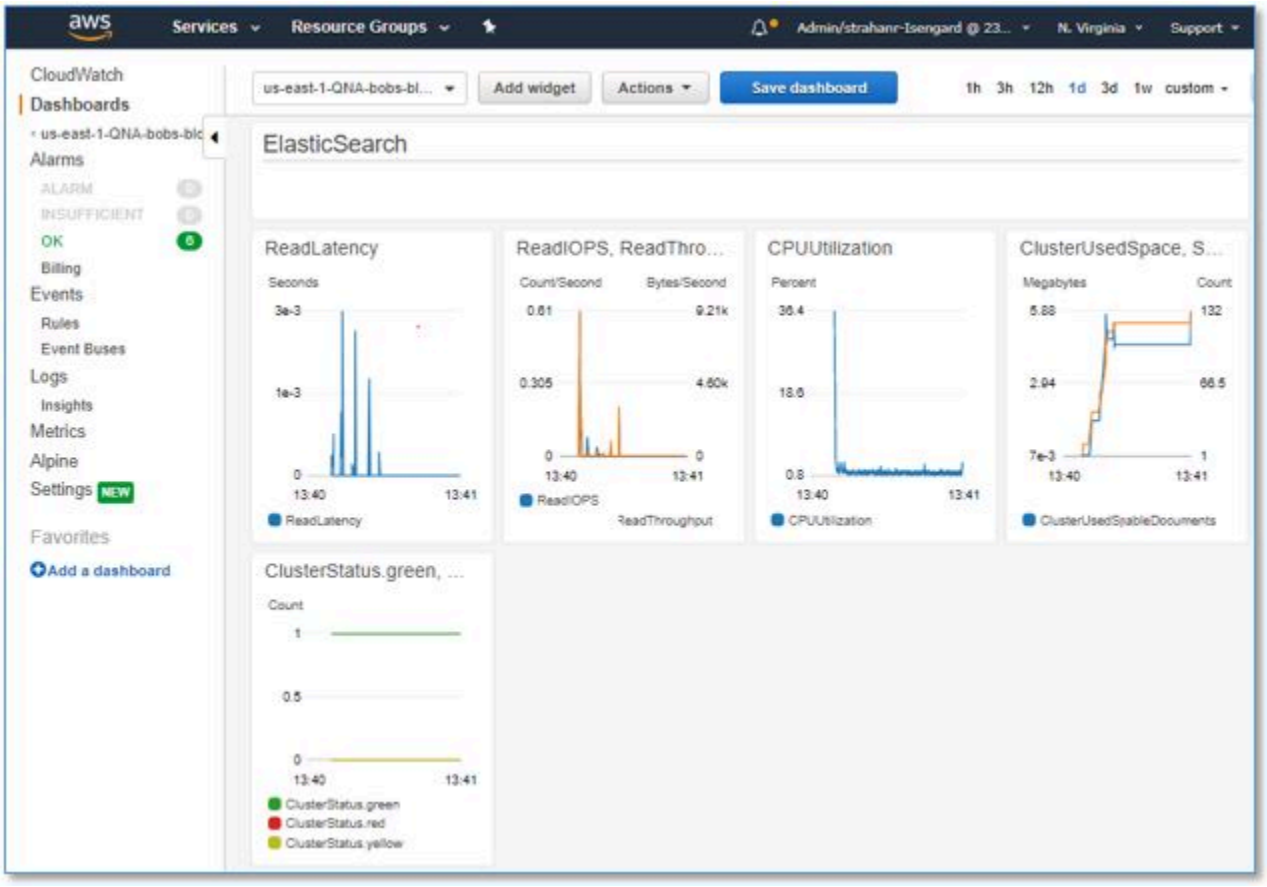


Sample Kibana dashboard

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.



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.
3. Choose the **Menu** tool in the upper right of the **>Fulfillment Lambda** wdget, select **View logs**, then and choose the **AWS Lambda** function**.**

FulfillmentLambda function

4. 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.

Semantic question matching using LLM text embeddings

QnABot can use text embeddings to provide semantic search capabilities by using large language models (LLMs). The goals of these features are to improve question matching accuracy while reducing the amount of tuning required when compared to the default Opensearch keyword-based matching. Some of the benefits include:

- Improved FAQ accuracy with semantic matching vs. keyword matching (comparing the meaning of questions vs. comparing the individual words).
- Fewer training utterances are required to match a diverse set of queries. This results in significantly less tuning to get and maintain good results.
- Better multi-language support because translated utterances only need to match the original question's meaning, not the exact wording.

For example, with semantic matching activated, *"What's the address of the Whitehouse?"* matches to *"Where does the president live?"* and *"How old are you?"* matches with *"What is your age?"*. These examples won't match using the default keywords because they don't share any of the same words.

To enable these expanded semantic search capabilities, QnABot can use:

- Embeddings from a Text Embedding model hosted on a pre-built Amazon SageMaker endpoint (recommended).
- Embeddings from a user provided custom Lambda function.

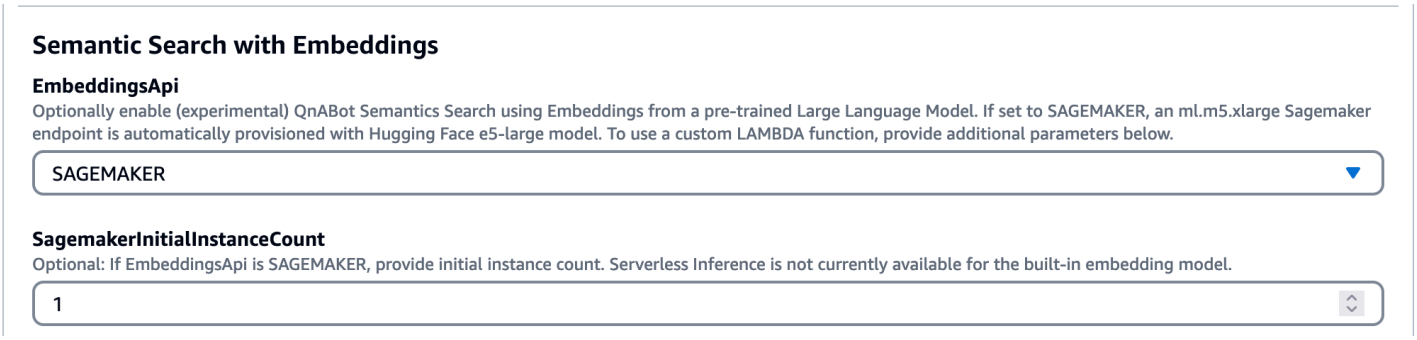
> **Note**
>
> This is an optional feature available as of v5.3.0. We encourage you to try it out on non-production instances initially to validate expected accuracy improvements and to test for any regression issues. Refer to the Cost section to see estimates of how this feature affects pricing.

> **Note**
>
> By choosing to enable the Semantic question matching using LLM text embeddings, you acknowledge that QnABot on AWS engages third-party generative artificial intelligence (AI) models that AWS does not own or otherwise has 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 may 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. QnABot on AWS is offered to you as "AWS Content" under your agreement with AWS.

Using the built-in Amazon SageMaker model

QnABot comes bundled with the ability to manage the lifecycle of a pre-built embeddings model hosted on Amazon SageMaker. In this mode, QnABot provisions a SageMaker inference endpoint running the Hugging Face el5-large model.

To activate, deploy a stack and set **EmbeddingsAPI** to SageMaker. By default, a 1 node **ml.m5.xlarge** endpoint automatically provisions. For large volume deployments, users can add nodes by setting the parameter **SagemakerInitialInstanceCount**. See the Cost section for pricing details.



Semantic Search with Embeddings

> **Note**
>
> These settings cannot be changed through the Content Designer settings page. To provision and deprovision the SageMaker instances, you must update your CloudFormation stack.

> **Note**
>
> The embeddings model provided by Sagemaker for QnABot is EL5. This only supports the English Language so if you are trying to work with a non-English language then you should use your own Embeddings model and provide that Lambda Arn in your deployment. For more information read the section below on using a custom Lambda Function.

Using a custom Lambda function

Users that wish to explore alternate pretrained or fine-tuned embeddings models can integrate a custom-built Lambda function. By using a custom Lambda function, you can build your own embeddings model or even choose to connect to an external embeddings API.

> **Note**
>
> If integrating your Lambda with external resources, evaluate the security implications of sharing data outside of AWS.

To begin, you'll need to create a valid Lambda function. Your custom Lambda function should accept a JSON object containing the input string and return an array which contains the embeddings. Record the length of your embeddings array because you need it to deploy the stack (this is also referred to as the *dimensions*).

Lambda event Input:

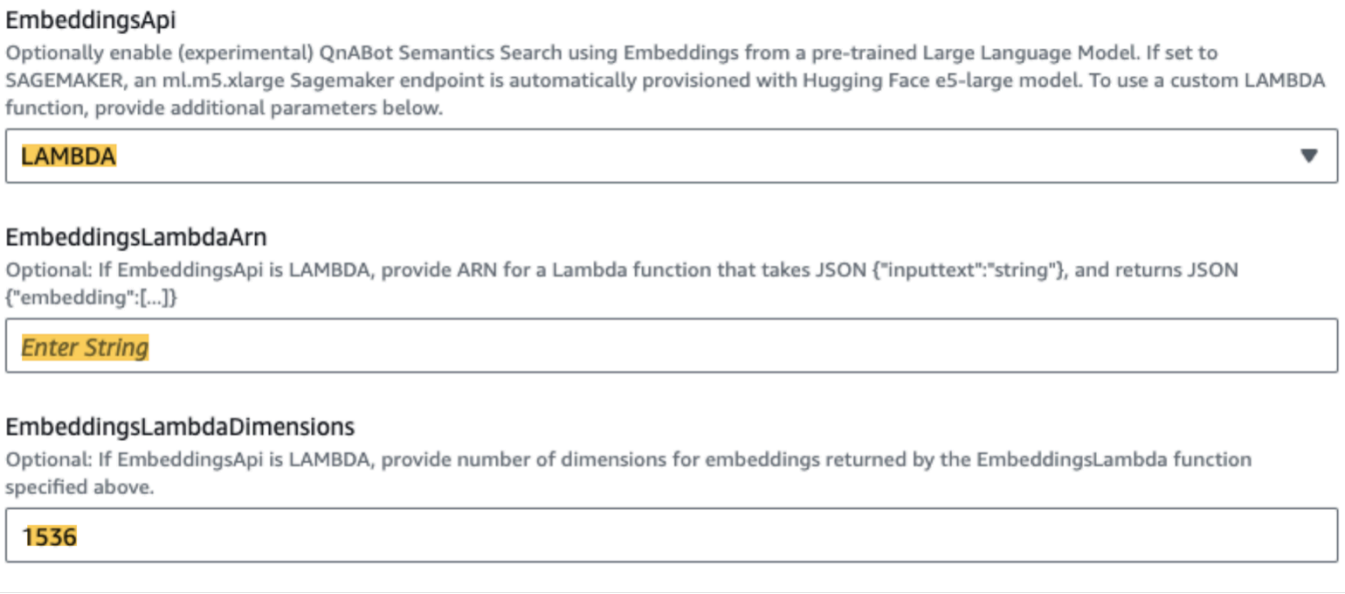
```
{
  // inputtype has either a value of 'q' for question or 'a' for answer
  "inputType": "string",

  // inputtext is the string on which to generate your custom embeddings
  "inputText":"string"
}
```

Expected Lambda JSON return object:

```
{“embedding”: [...] }
```

Once your Lambda function is ready, you can deploy the stack. To activate your Lambda function for embeddings, deploy the stack with **EmbeddingsAPI** set to LAMBDA. You will also need to set **EmbeddingsLambdaArn** to the ARN of your Lambda function and **EmbeddingsLambdaDimensions** to the dimensions returned by your Lambda function.



Semantic Search with Lambda function

> **Note**
>
> You can't change these settings through the Content Designer settings page. To correctly reconfigure your deployment, update your CloudFormation stack to modify these values.

Settings available for text embeddings

> **Note**
>
> Many of these settings depend on the underlying infrastructure being correctly configured. Follow the instructions found at Using the built-in Amazon SageMaker model or Using a custom Lambda Function before modifying any of the settings below.

Once your QnABot stack is installed with **EmbeddingsApi** activated, you can manage several settings through the Content Designer **Settings** page:

- **EMBEDDINGS_ENABLE**: To enable/disable use of semantic search using embeddings:
  - Set to FALSE to turn off the use of embeddings-based queries.
  - Set to TRUE to re-activate the use of embeddings based queries after previously setting it to FALSE.

> **Note**
>
> **EMBEDDINGS_ENABLE** will be set default to **TRUE**, if **EmbeddingsAPI is provisioned to SAGEMAKER or LAMBDA. If not provisioned, EMBEDDINGS_ENABLE will be set default to FALSE.**
> Setting TRUE when the stack has `EmbeddingsAPI` set to `DISABLED` will cause failures since the QnABot stack isn't provisioned to support generation of embeddings.

- If you turn off embeddings, you will also want to re-activate keyword filters by setting **ES_USE_KEYWORD_FILTERS** to TRUE.
- If you add, modify, or import any items in Content Designer when **EMBEDDINGS_ENABLE** is set to FALSE, then embeddings won't get created and you'll need to re-import or re-save those items after re-enabling embeddings. Similarly, if you update/change your embeddings model or parameter and need the embeddings to be recalculated, you will need to export and re-import the embeddings related items in your content designer, for example, the questions.

> **Note**
>
> This setting allows you to toggle embeddings on and off, it does not manage the underlying infrastructure. If you choose to permanately turn off embeddings, update the stack as well. This will allow you to deprovision the SageMaker instance to prevent incurring additional costs.

- **ES_USE_KEYWORD_FILTERS**: This setting should now default to FALSE. Although you can use keyword filters with embeddings based semantic queries, they limit the power of semantic search by forcing keyword matches (preventing matches based on different words with similar meanings)
- **ES_SCORE_ANSWER_FIELD**: If set to TRUE, QnABot on AWS runs embedding vector searches on embeddings generated on answer field if no match is found on question fields. This allows QnABot to find matches based on the contents on the answer field as well as the questions. Only the plain text answer field is used (not the Markdown or SSML alternatives). Tune the individual thresholds for questions and answers using the additional settings of:
  - **EMBEDDINGS_SCORE_ANSWER_THRESHOLD**
  - **EMBEDDINGS_SCORE_ANSWER_THRESHOLD**

- **EMBEDDINGS_SCORE_THRESHOLD**: Change this value to customize the score threshold on *question* fields. Unlike regular ElasticSearch queries, embeddings queries always return scores between 0 and 1, so we can apply a threshold to separate good from bad results.
  - If no question has a similarity score above the threshold set, QnABot on AWS rejects the match and reverts to:
    1. Tries to find a match using the answer field (only if **ES_SCORE_ANSWER_FIELD** is set to TRUE.
    2. Kendra fallback (only if enabled)
    3. **no_hits**
  - The default threshold is `0.85`, but you may need to modify this based on your embeddings model and your experiments.

> **Note**
>
> Use the Content Designer **TEST** tab to see the hits ranked by score for your query results.

- **EMBEDDINGS_SCORE_ANSWER_THRESHOLD**: change this value to customize the score threshold on *answer* fields. This setting is only used when **ES_SCORE_ANSWER_FIELD** is set to **TRUE** and QnABot has failed to find a suitable response using the question field.
  - If no question has a similarity score above the threshold set, QnABot on AWS rejects the match and reverts to:
    1. Amazon Kendra fallback (only if enabled)
    2. **no_hits**
  - The default threshold is `0.80`, but you may need to modify this based on your embeddings model and your experiments.

> **Note**
>
> Use the Content Designer **TEST** tab and select the **Score on answer field** checkbox to see the hits ranked by score for your answer field query results

Recommendations for tuning with LLMs

When using embeddings in QnABot, we recommend generalizing questions because more user utterances will match a general statement. For example, the embeddings model will cluster *checkings* and *savings* with *account*, so if you want to match both account types, just refer to *account* in your questions.

Similarly for the question/utterance of *transfer to an agent*, consider using *transfer to someone* as it will better match with *agent*, *representative*, *human*, *person*, etc.

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.

Follow these steps to export all the items that are in the QnABot on AWS category (items whose ID starts with "*AWS QnABot*").

1. Log in to the content designer, choose the tools menu (☰), and then choose **Export.**
2. Enter AWS QnABot in the optional filter field, and then choose **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.

   Sample JSON file for importing and exporting chatbot answers:

```
{
  "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 teh Q and A Bot
 Content Designer UI. End users ask questions using teh Lex web UI, which supports
 voice or chat, or using Alexa devices for hands free voice interaction. ",
      "r": {
        "title": "",
        "imageUrl": ""
      },
      "qid": "QnABot.002"
    }
  ]
}
```

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**.

For a step-by-step procedure on importing Excel (.xlsx) workbooks, refer to Excel workbooks import.

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.

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**). Below are a set of example of settings frequently use. For more complete information on the settings, see settings.md.

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

Configure keyword filters feature

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.

Configure words and phrases replacement in user questions

If you want to replace words or phrases in user questions, for example, you want "Thumbs Up" rewritten to be a direct match by question ID, you can use the **SEARCH_REPLACE_QUESTION_SUBSTRINGS** setting.

1. Log in to the content designer, select the tools menu (☰), and then choose **Settings**.
2. Change the value of the setting **SEARCH_REPLACE_QUESTION_SUBSTRINGS** to a json object like {"Thumbs Down": "QID::Feedback.001", "Thumbs Up": "QID::Feedback.002"}. You can add additional pairs separated by commas if you want.
3. Scroll to the bottom and select **Save**. This will now rewrite all input matching "Thumbs Down" to "QID::Feedback.001". Change the value of the setting **ES_USE_KEYWORD_FILTERS** from `true` to `false`.

Configure pre-processing and post-processing Lambda hooks

Content Designer gives you the ability to dynamically generate answers by letting you specify your own Lambda 'hook' function for any item/question defined in the content designer (hook).

In addition, a Lambda hook can be called as the first step in the fulfillment pipeline (PREPROCESS) or after processing has completed (POSTPROCESS) and before the userInfo is saved to DynamoDB and the result has been sent back to the client. You can add pre-processing and post-processing Lambda hooks (that run before preprocessing and after every question is run) via the **Settings** page.

1. Log in to the content designer, select the tools menu (☰), and then choose **Settings**.
2. Find the LAMBDA_PREPROCESS_HOOK setting and set its value to your hook name. The name of the Lambda must start with "qna-" or "QNA-" to comply with the permissions of the role attached to the Fulfillment Lambda e.g. QNA-ExampleJSLambdahook.
3. Scroll to the bottom and select Save. The lambda function specified will now be run before each question is processed.

> **Note**
>
> For more information on lambda hooks, see the lambda hooks readme.md file and the settings.md file.

Configure multi-language support

QnABot supports both voice and text interactions in multiple languages. QnABot can detect the predominant language in an interaction by using Amazon Comprehend, a natural language processing (NLP) service that uses machine learning to find insights and relationships in text. The bot then uses Amazon Translate, a neural machine translation service to convert questions and answers across languages from a single shared set of FAQs and documents.

By default the multi language feature is disabled. QnABot uses a property named ENABLE_MULTI_LANGUAGE_SUPPORT with a default value of false. You can change this setting using the Content Designer Settings page. Set it to true to enable multi language support.

QnABot converts the question posed by the user to your core language that was chosen during your deployment, using Amazon Translate, and performs a lookup of the answer in Amazon OpenSearch Service (successor to Amazon Elasticsearch Service) just as it normally does, using the Native language translation of the question. Searches are done in the language you have selected for your deployment only since QnABot documents are indexed using the language analyzer and their corresponding text analyzer (stemming, stop words, etc.) Once it finds the question, QnABot will serve up the configured answer.

You can also import the sample or extension named Language / Multiple Language Support from the QnABot Import menu option. This adds two questions to the system: Language.000 and Language.001. The first question allows the end user to set their preferred language explicitly; the latter resets the preferred language and allows QnABot to choose the locale based on the automatically detected predominant language.

When deploying the AWS QnABot solution's (version 5.5.0 and higher) CloudFormation template, you will see a language parameter in which you have the option of selecting one of the 33 languages. This Language parameter is used as the core Language for your QnABot deployment. The Language Analyzer for your Opensearch index setting will use the Language that you have specified in this parameter. In the case that your input has a low confidence rate it will default to English as that is the Backup Language that will be used.

- Custom terminology will also support your **NATIVE_LANGUAGE**.
- For the LLM, Falcon 40B is supported in the following languages (English, German, Spanish, French, with limited capabilities also in in Italian, Portuguese, Polish, Dutch, Romanian, Czech, Swedish.). If you are using one of the languages listed below then be sure to change your Prompt into your language. Otherwise if the Language is not supported then you will need to use your own Lambda function and LLM.

- For the embeddings, el5-large model only supports English, If you are using a non-English Native language then you should use your own embeddings model and provide the Lambda in your deployment.
- If using the Thumbs up and down feature, you should translate Thumbs up and down into your native language and put that phrase in the **PROTECTED_UTTERANCES** setting. This is to prevent utterances from being treated as a question by the QnAbot. To do this you can do the following steps:
  1. Use the AWS translate API to translate Thumbs up and Thumbs down to your deployment Language if it is not English.
  2. Add the translation of Thumbs up and down in the website client config inside your qnabot code and deploy.
  3. Add the translation of the Thumbs up and down as a question in your QnABot deployment.
  4. Go to the content designer, navigate to the top left and select **Settings**.
  5. Find the **PROTECTED_UTTERANCES** variable and insert that phrase in by adding a comma, and then enter in the translation.

- PII redaction will still be for Engish as that is still accurate with other languages.
- Changing the **NATIVE_LANGUAGE** should always be done from the Cloudformation Stack by changing the Language Parameter

For more information refer to README.md or QnABot Workshop Guide.

Configure Personally Identifiable Information (PII) Rejection and Redaction

QnABot can now detect and redact Personally Identifiable Information (PII) using Amazon Comprehend and regular expressions.

If **ENABLE_REDACTING** is set to `true`, the Comprehend detected PII entities will also be redacted from Amazon CloudWatch logs and Amazon Opensearch logs.

PII Rejection and Redaction

Integrating 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. Amazon 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 Amazon 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 QnABot on AWS 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 DefaultAmazon KendraIndexId when you installed or updated QnABot on AWS, then KENDRA_FAQ_INDEX will be pre-configured.
2. Replicate all items from Content Designer to the Amazon 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 Amazon 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 Amazon Kendra FAQ answers used by QnABot on AWS.

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

Using Amazon 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 as follows:

- Update the QnABot on AWS setting **ALT_SEARCH_KENDRA_INDEXES** to specify one or more Amazon Kendra indexes to use for fallback searches.
  - 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 **DefaultAmazon KendraIndexId** when you installed or updated QnABot on AWS, then ALT_SEARCH_KENDRA_INDEXES will be pre-configured.

Amazon Kendra redirect

QnABot on AWS supports multiple mechanisms for dynamic interaction flows. For example:

- Using Lambda Hooks in a given Item ID to perform additional actions, such as creating a ticket, resetting a password, and saving data to a data store.
- Using a Amazon Kendra Index as a fallback mechanism to look for answers to user's questions.

There are various options to process Amazon Kendra queries. One option is to create a custom Lambda hook and map it to an Item ID. The Lambda hook then includes the business logic to use a Amazon Kendra Index and process the query.

The Amazon Kendra Redirect feature provides a much simpler option. You can include a Amazon Kendra query within an Item ID, and QnABot will do the rest to process the Amazon Kendra request and respond back with the results.

Configuring an Item ID with Amazon Kendra redirect

You can configure an item ID with Kendra redirect UI.

Amazon Kendra redirect configuration

- Create a QnABot question as you would normally do by providing an Item ID and Questions/ Utterances.
- Expand the Advanced option.
- **Amazon Kendra Redirect: Query Text** accepts a QueryText to search for (for example what is q and a bot) and retrieve the answer from the Amazon Kendra Fallback index specified in **Settings**. Amazon Kendra searches your index for text content, question, and answer (FAQ) content. You can also use handlebars to substitute values using session attributes or slots to support dynamic queries.

- **Amazon Kendra Redirect: Confidence** score threshold provides a relative ranking that indicates how confident Amazon Kendra is that the response matches the query. This is an optional field having one of the values of: **LOW | MEDIUM | HIGH | VERY HIGH**. If no value is provided, the value for the ALT_KENDRA_FALLBACK_CONFIDENCE_THRESHOLD setting is used.
- **Amazon Kendra query arguments** is an optional field that allows filtered searches based on document attributes. For example: "AttributeFilter": {"EqualsTo": {"Key": "City", "Value": {"StringValue": "Seattle"}}}. You can also use handlebars to substitute values using session attributes or slots to support dynamic queries.

For more information on using Amazon Kendra query arguments, refer to the Amazon Kendra Query API in the *Amazon Kendra Develop Guide*.

> **Note**
>
> - Answer fields are ignored when Amazon KendraRedirect query is used.
> - Use this feature for use cases where you have Item IDs that directly need to interact with a Amazon Kendra Index as configured in **Settings**.
> - When applying Amazon Kendra query arguments, check if the document fields are searchable. Searchable, determines whether the field is used in the search. For more information, refer to Mapping data source fields in the *Amazon Kendra Develop Guide*.

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:
   - **ENABLE_WEB_INDEXER:** true
   - **KENDRA_INDEXER_URLS:** https://aws.amazon.com/lex/faqs/
   - **KENDRA_WEB_PAGE_INDEX:** Existing Amazon Kendra index ID
   - **KENDRA_INDEXER_SCHEDULER:** rate (1 day)
3. From content designer, select the tools menu (☰), and then choose **Amazon Kendra Web Crawler**.
   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?*". QnABot on AWS 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.

Query disambiguation and text generation using LLMs

QnABot on AWS can leverage large language models (LLMs) to provide a richer, more conversational chat experience. The goal of these features is to minimize the amount of individually curated answers administrators are required to maintain, improve question matching accuracy by providing query disambiguation, and to enable the solution to provide more concise answers to users, especially when using the Kendra fallback feature.

These benefits are provided through three primary features:

- **Text passage question type -** In the Content Designer web interface, administrators can store full text passages for QnABot to use. When a question gets asked that matches against this passage, QnABot can leverage LLMs to answer the user's question based on information found within the passage.
- **Query Disambiguation -** By leveraging an LLM, QnABot can take the user's chat history and generate a standalone question for the current utterance. This enables users to ask follow up questions which on their own may not be answerable without context of the conversation.

> **Note**
>
> The ability to answer follow up questions is similar to what QnABot Topics aims to solve. Consider that as an option if you're unable to use the LLM features.

- **Context based text generation for question answering -** Given a text passage sourced from your question bank or Amazon Kendra index, QnABot can use an LLM to generate concise answers to user's questions from your data source. This prevents the need for users to sift through larger text passages to find the answer.

These features (together with embeddings) enable QnABot to serve end users with a more conversational chat experience using various AI and NLP techniques. To enable the use of these

features, users must deploy QnABot with the LLM selection of their choice. Users can choose to use any of the following LLM providers:

- An open source model hosted on Amazon SageMaker.
- Any other LLM model through a user provided custom Lambda function.

> **Note**
>
> Note: These are optional features available as of v5.4.0. We encourage you to try it out on non-production instances initially to validate expected accuracy improvements and to test for any regression issues. Refer to the Cost section to see estimates of how these features affect pricing.

> **Note**
>
> By choosing to enable Query disambiguation and text generation using LLMs, you acknowledge that QnABot on AWS engages third-party generative artificial intelligence (AI) models that AWS does not own or otherwise has 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 may 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. QnABot on AWS is offered to you as "AWS Content" under your agreement with AWS.

Enabling LLM support

In this mode, QnABot on AWS provisions a Sagemaker endpoint running the Hugging Face falcon-40b-instruct model.

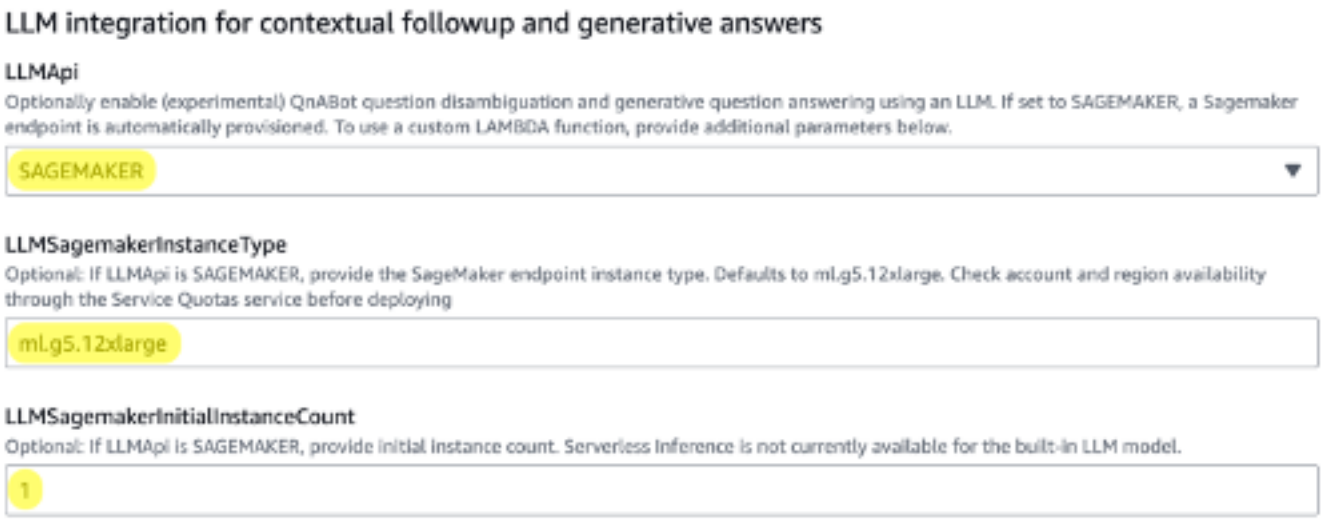
By default, a 1-node `ml.g5.12xlarge` endpoint is automatically provisioned. This is the minimum instancedeploy size which can support a Sagemaker endpoint, therefore the **LLMSagemakerInitialInstanceType** should only be set to an instance at least as big as `ml.g5.12xlarge` instance. For large volume deployments, add additional nodes by setting the parameter **LLMSagemakerInitialInstanceCount**.

See the SageMaker pricing documentation for relevant costs in your Region.

To deploy the stack using the SageMaker endpoint:

- Set **LLMApi** to **SAGEMAKER**
- If using the Kendra fallback:
  - Set **DefaultKendraIndexId** to the Index ID of your existing Kendra index containing ingested documents.
- If using text passages:
  - Enable text embeddings by setting **EmbeddingsApi** to the mechanism of your choice.

    For options, see Semantic question matching using LLM text embeddings.



LLM integration Amazon SageMaker

> **Note**
>
> The **ml.g5.12xlarge** instance type is not enabled by default in your AWS account and must be requested on a per region basis. Before deploying the solution with this option, log into

the AWS console, access AWS Service Quotas and search for Amazon SageMaker under the AWS services list. Once selected, search for the quota called **ml.g5.12xlarge for endpoint usage**. At a minimum, you will need to request a quota increase to one (you can request more to accommodate high-volume production deployments).

Note

If using a language other than English, please confirm that Falcon-40B supports that Language you are working with and make sure to change the prompts in the settings for the LLM to be in that Language. If Falcon-40B does not support the language then you will need to provide an custom Lambda for your deployment. Please read the next section to see how you can do that.

Custom Lambda function

If the pre-built options don't work for your use case, or you want to experiment with other LLMs, you can build a custom Lambda function to integrate with the LLM of your choice. The provided Lambda function takes as input the prompt, model parameters, and the QnABot settings object. Your Lambda function can invoke any LLM you choose, and return the prediction in a JSON object containing the key generated_text. You provide the ARN for your Lambda function when you deploy or update the solution.

Note

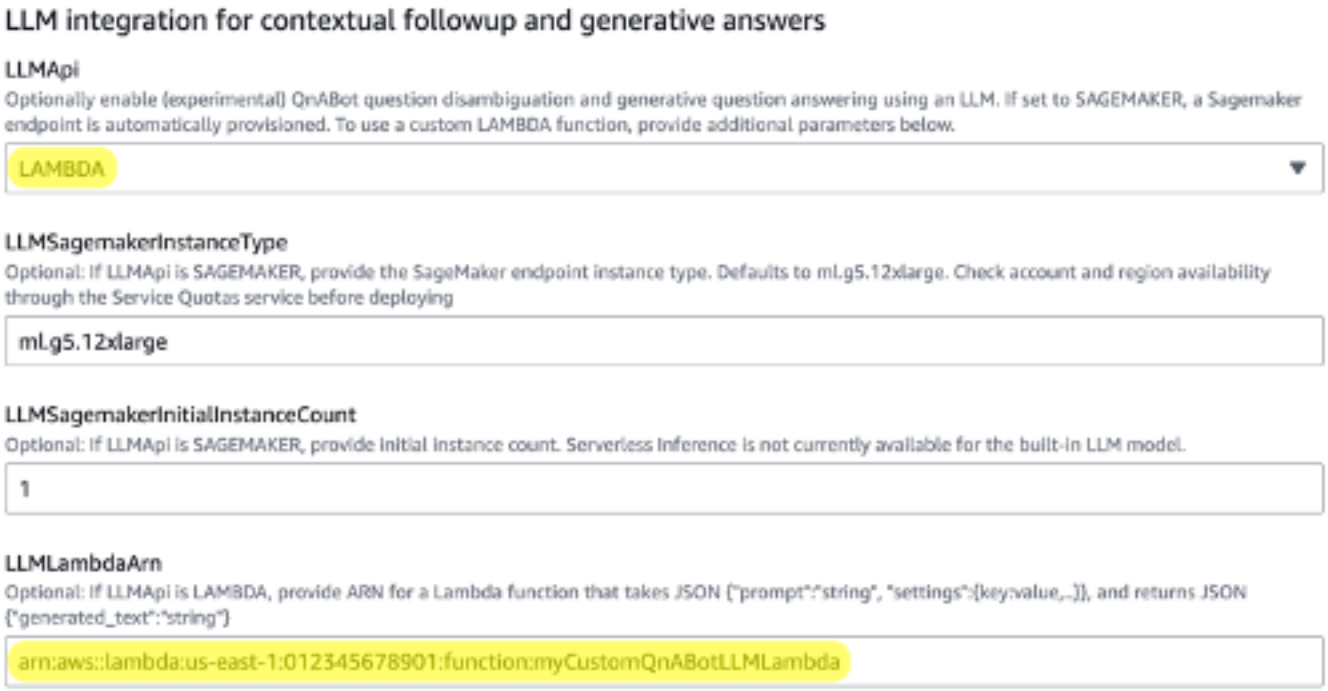
If integrating your Lambda with external resources, evaluate the security implications of sharing data outside of AWS.

To deploy the stack using a custom Lambda function:

- Set **LLMAp** to **LAMBDA.**
- Set **LLMLambdaArn** to the ARN of your Lambda function.
- If using the Kendra Fallback:
  - Set **DefaultKendraIndexId** to the Index ID of your existing Kendra index containing ingested documents.

- If using the Kendra Fallback:
  - Enable text embeddings by setting **EmbeddingsApi** to the mechanism of your choice.

    For options, see Semantic question matching using LLM text embeddings.



LLM integration LAMBDA

Your Lambda function is passed an event of the form:

```
{
  // prompt for the LLM
  "prompt": "string",

  // object containing key/value pairs for the model parameters
  // these parameters are defined on the QnABot settings page
  "parameters":{"temperature":0,...},

  // settings object containing all default and custom QnAbot settings
  "settings":{"key1":"value1",...}
}
```

and returns a JSON structure of the form:

```
{"generated_text":"string"}
```

Below is an example of a minimal Lambda function for testing, however, you must extend it to invoke your LLM.

```
def lambda_handler(event, context):
    print(event)
    prompt = event["prompt"]
    model_params = event["parameters"]
    settings = event["settings"]

    # REPLACE BELOW WITH YOUR LLM INFERENCE API CALL
    generated_text = f"This is the prompt: {prompt}"

    return {
        'generated_text': generated_text
    }
```

Query disambiguation and conversation retrieval

Query disambiguation is the process of taking an ambiguous question (having multiple meanings) and transforming it into an unambiguous, standalone question. The new disambiguated question can then be used as a search query to retrieve the best FAQ, passage, or Kendra match. For example, with the new LLM disambiguation feature enabled, given the chat history context:

```
[{"Human":"Who was Little Bo Peep?"},{"AI":"She is a character from a nursery rhyme who
 lost her sheep."}]
```

and a follow up question:

```
Did she find them again?
```

the solution can rewrite (“disambiguate”) that question to provide all the context required to search for the relevant FAQ or passage:

```
Did Little Bo Peep find her sheep again?
```

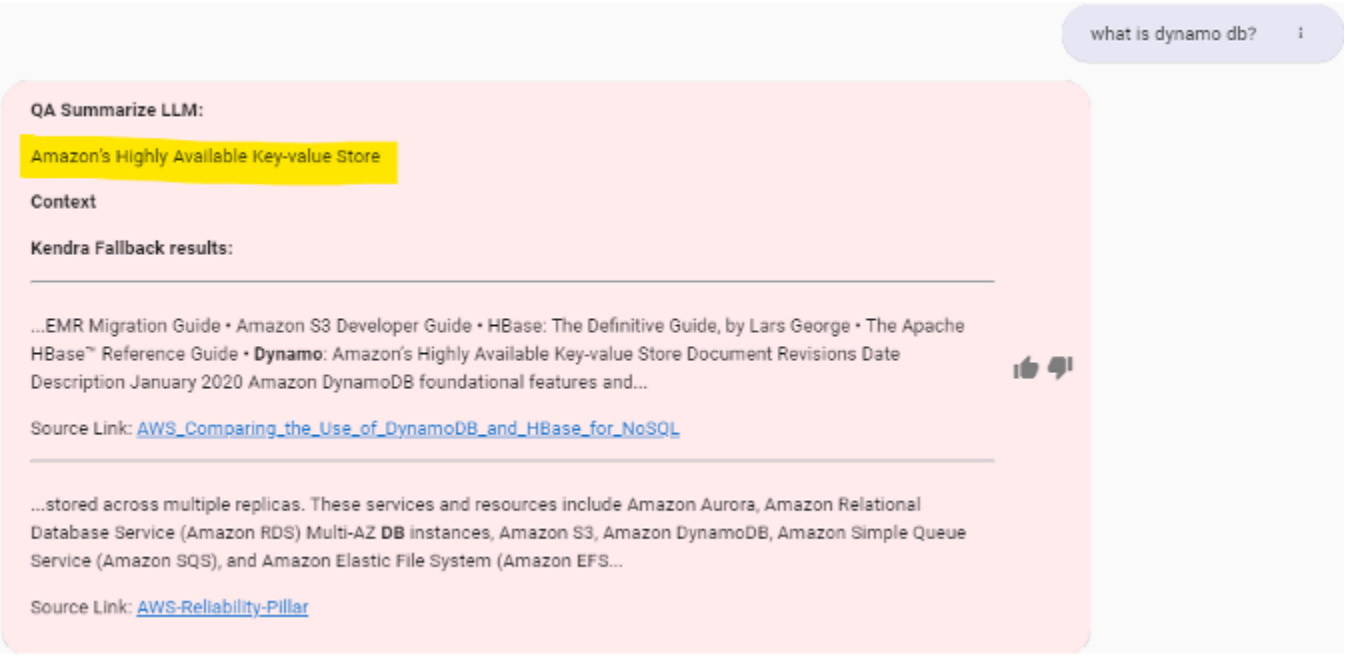
Context based text generation for question answering

Generate answers to questions from context provided by Kendra search results, or from text passages created or imported directly into QnAbot. Some of the benefits include:

- Generated answers allow you to reduce the number of FAQs you need to maintain since you can now synthesize concise answers from your existing documents in a Kendra index, or from document passages stored in QnABot as **text** items.
- Generated answers can be short, concise, and suitable for voice channel contact center bots and website/text bots.
- Generated answers are fully compatible with the solution's multi-language support - users can interact in their chosen languages and recieve generated answers in the same language.

For example, with these LLM QA features enabled, QnABot can answer questions from the AWS WhitePapers such as:

- *"What is DynamoDB?"*-> **Amazon's Highly Available Key-value Store.**
- *"What frameworks does AWS have to help people design good architectures?"* -> **Well-Architected Framework.**



Content based text generation DynamoDB

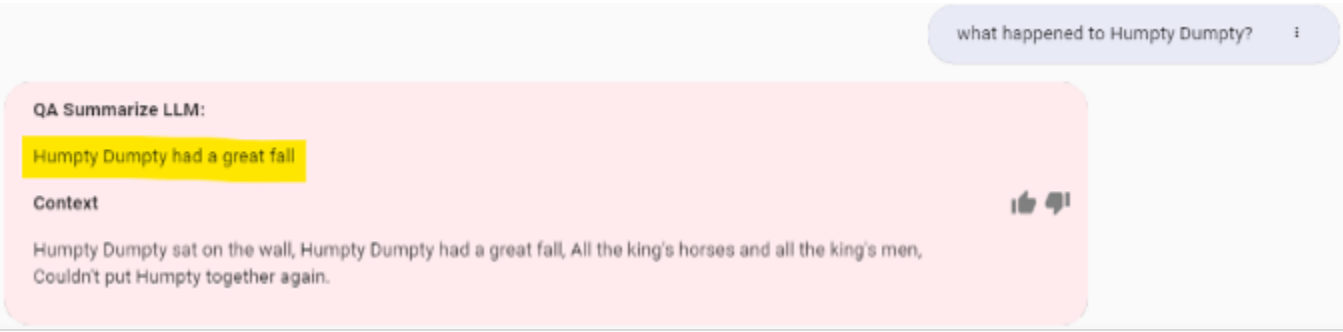
It can even generate answers to yes/no questions, like:

- *"Is Lambda a database service?"* -> **No, Lambda is not a database service.**

Even if you aren't using AWS Kendra, QnABot on AWS can answer questions based on passages created or imported into Content Designer, such as:

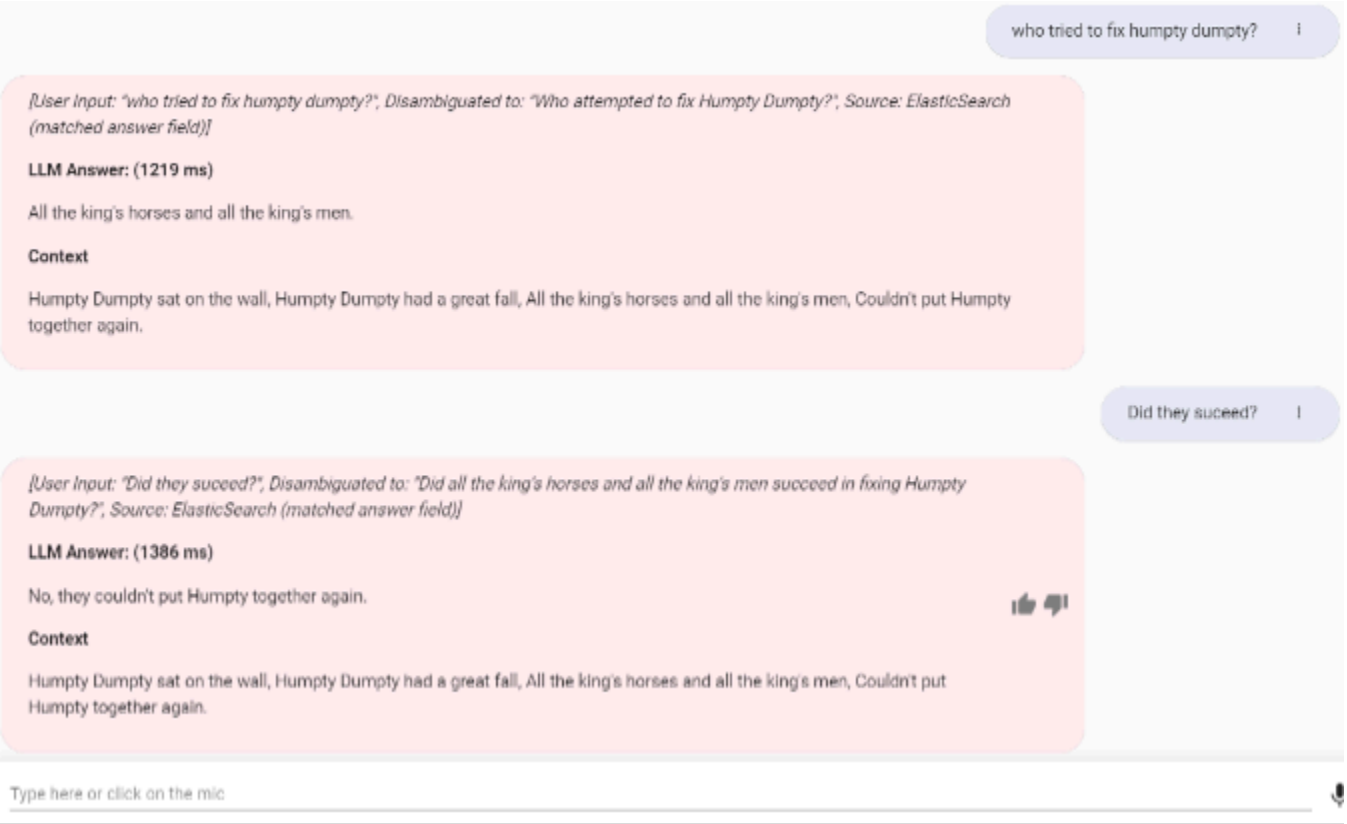
- *"Where did Humpty Dumpty sit?*" -> **On the wall.**
- "*Did Humpty Dumpty sit on the wall*?" -> **Yes**.
- *"Were the king's horses able to fix Humpty Dumpty?"* -> **No**.

all from a text passage item that contains the nursery rhyme.



Content based text generation response

You can use disambiguation and generative question answering together, as shown below:



Disambiguation and generative question answering

Settings available for LLM configuration

When the QnABot stack is installed, open Content Designer **Settings** page and configure the following settings:

- **ENABLE_DEBUG_RESPONSES -** Set to TRUE to add additional debug information to the solution's response, including any language translations (if using multi language mode), question disambiguation (before and after), and inference times for your LLM model(s).
- **ES_SCORE_TEXT_ITEM_PASSAGES -** Should be TRUE to enable the new text passage items to be retrieved and used as input context for geneartive QA Summary answers.

> **Note**
>
> 'qna' items are queried first, and if none meet the score threshold, then QnABot queries the text field of 'text' items.

- **EMBEDDINGS_TEXT_PASSAGE_SCORE_THRESHOLD -** Applies only when embeddings are enabled (recommended) and if **ES_SCORE_TEXT_ITEM_PASSAGES** is TRUE. If embedding similarity score on text item field is under threshold the match is rejected. Default threshold is 0.80.
- **ALT_SEARCH_KENDRA_INDEXES -** Set to the ID (not the name) of your Kendra index where you have ingested documents of web pages that you want to use as source passages for generative answers. If you plan to use only text passage items instead of Kendra, leave this setting blank.
- **ALT_SEARCH_KENDRA_MAX_DOCUMENT_COUNT -** The number of passages from Kendra to provide in the input context for the LLM.

Scroll to the bottom of the settings page and observe the new LLM settings:

- **LLM_API: One of SAGEMAKER, LAMBDA-** Based on the value chosen when you last deployed or updated the solution stack.
- **LLM_GENERATE_QUERY_ENABLE -** Set to TRUE or FALSE to enable or disable question disambiguation.
- **LLM_GENERATE_QUERY_PROMPT_TEMPLATE -** The prompt template used to construct a prompt for the LLM to disabiguate a followup question. The template can use the following placeholders:
  - `{history}` - Placeholder for the last **LLM_CHAT_HISTORY_MAX_MESSAGES** , messages in the conversational history to provide conversational context.
  - `{input}` - Placeholder for the current user utterance or question.
- **LLM_GENERATE_QUERY_MODEL_PARAMS -** Parameters sent to the LLM model when disambiguating follow-up questions. Default parameter: `{"temperature":0}`. Check model documentation for additional values that your model provider accepts.
- **LLM_QA_ENABLE -** Set to TRUE or FALSE to enable or disable generative answers from passages retrieved via embeddings or Kendra fallback (when no FAQ match its found).

> **Note**
>
> LLM based generative answers are not applied when an FAQ/QID matches the question.

- **LLM_QA_PROMPT_TEMPLATE -** The prompt template used to construct a prompt for the LLM to generate an answer from the context of a retrieved passages (from Kendra or embeddings). The template can use the following placeholders:

- `{context}` – Placeholder for passages retrieved from the searchc query – either a QnABot text item passage, or the top **ALT_SEARCH_KENDRA_MAX_DOCUMENT_COUNT** Kendra passages.
- `{history}` – Placeholder for the last **LLM_CHAT_HISTORY_MAX_MESSAGES** messages in the conversational history, to provide conversational context.
- `{input}` – Placeholder for the current user utterance / question.
- `{query}` – Placeholder for the generated (disambiguated) query created by the generate query feature.

- **LLM_QA_NO_HITS_REGEX -** When the pattern specified matches the response from the LLM. For example: *"Sorry, I don't know",* then the response is treated as no_hits, and the default **EMPTYMESSAGE** or Custom Don't Know ("no_hits") item is returned instead. Disabled by default, since enabling it prevents easy debugging of LLM don't know responses.
- **LLM_QA_MODEL_PARAMS -** Parameters sent to the LLM model when generating answers to questions. Default parameter: `{"temperature":0}`. Check model documentation for additional values that your model provider accepts.
- **LLM_QA_PREFIX_MESSAGE -** Message use to prefix LLM generated answer. Can be empty.
- **LLM_QA_SHOW_CONTEXT_TEXT -** Set to `TRUE` or `FALSE` to enable or disable inclusion of the passages (from Kendra or Embeddings) used as context for LLM generated answers.
- **LLM_QA_SHOW_SOURCE_LINKS** - Set to `TRUE` or `FALSE` to enable or disable Kendra source links or passage refMarkdown links (doc references) in markdown answers.
- **•LLM_CHAT_HISTORY_MAX_MESSAGES -** The number of previous questions and answers (chat history) to maintain (in the DynamoDB UserTable). Chat history is necessary for the solution to disambiguate follow up questions from previous question and answer context.

Using automatic translation

This solution supports automatic translation to the end user's language using Amazon Translate. To use automatic translation, follow these steps:

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 QnABot on AWS, 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 QnABot on AWS, and use this translation to tune the content as needed to ensure QnABot on AWS 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. Custom Terminology supports the language that you are deploying with. For more information on how to use the **Import Custom Terminology** tool in content designer, refer to the README.md file in GitHub.

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.

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**.
   - Enter **Elicit Response: ResponseBot Hook:**`QNAName`
   - 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 returna the values for the **FirstName** and **LastName** elements back to the solution as slot values in a fulfilled response.

The solution stores the returned values for **FirstName** and **LastName** 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.

9. Log in to the content designer, and choose **Add.**
10. Enter ID: **ElicitResponse.002**
11. Enter question: `Ask my age`
12. Enter answer: `Hello {{SessionAttributes.name_of_user.FirstName}} - What is your age in years?`
13. Choose **Advanced**.
    - Enter **Elicit Response: ResponseBot Hook**: `QNAAge`
    - Enter **Elicit Response: Response Session Attribute Namespace**: `age_of_user`
14. Choose **CREATE** to save the new item.
15. 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}.

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”* Enter and confirm your name.
   - Enter and confirm your name.
   - 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:
   - ID: **ElicitResponse.003**, question: “*Under 18”*, answer: “*Under 18 answer*“.
   - ID: **ElicitResponse.004**, question: “*Over 18*“, answer: “*Over 18 answer*“.

2. Edit item **ElicitResponse.002**
   - Add Chaining Rule: `(SessionAttributes.age_of_user.Age< 18) ? "Under 18" : "Over 18"`
   - Choose **UPDATE** to save the modified item.
3. Use the web UI to ask: *"Ask my name"*.
   - Enter and confirm your name.
   - 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 less than 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. For more information, see 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**.

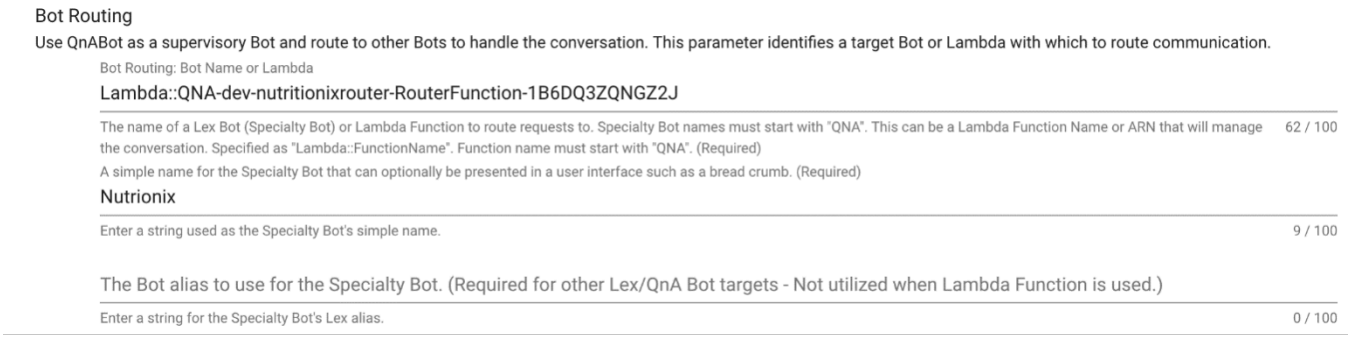
You can optionally specify an AWS Lambda function instead of a JavaScript expression when you need to evaluate complex chaining rule logic. Your Lambda function is invoked with the full user request context and should evaluate and return the next question as a simple string. Alternatively, the Lambda 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 such as a data lookup. 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.”

Configuration of bot routing

Configuration of Bot Routing is simple. Each question in QnAbot contains an optional section which allows configuration of a BotRouter.

> **Note**
>
> This is optional. Please leave empty and QnABot will not act as a BotRouter for the question being edited.



Bot routing

The example image shows an integration we've developed which communicates with the Nutritionix Bot.

- Bot Name or Lambda function- You can configure and existing Lex Bot or configure a specialty BotRouter implemented via a Lambda function.
- Simple name - A short string that we expect web User Interfaces to use as a breadcrumb to identify where in an enterprise the user is interacting.
- Lex Alias - If a target bot is configured, the alias used to communicate with the target bot.

> **Note**
>
> When integrating with other Lex Bots or Lambdas, the permission to communicate with the target Lex bot or with a new BotRouter (Lambda) need to be added to the QnABot's fulfillment lambda role.

Bot routing

Bots come in many shapes and sizes, and exist to perform a variety of automation tasks. Usually they take input from a human and respond performing some task. Bots might ask for additional input, verify the input, and respond with completion. Bots might be implemented using Amazon Lex or might be implemented using other toolsets. A great example is the nutritionix botwhere you can tell the bot what you've had for breakfast and it will respond with nutrition information.

The solution adopted by QnABot is to coordinate (route) bot requests through a Supervisory Bot, to the appropriate Bot based on questions or tasks.

Content designers associate questions or tasks (qid's) that identify a BotRouter to target for the question. This is performed using the QnABot UI Designer. Once configured, if a user asks a question or directs the bot with some instruction, QnABot responds with an answer and sets up a channel to communicate with the specialty Bot. From that point, messages or responses from the user are delivered to the Specialty Bot. Specialty Bots respond to actions and QnABot delivers the answers.

This flow continues until one of these events occurs:

1. The user user cancels the conversation with the specialty Bot by uttering "exit", "quit", "bye", or a configurable phrase defined in the settings configuration of QnABot.
2. The specialty BotRouter (custom code) responds with a message indicating the conversation should be discontinued. (**QNABOT_END_ROUTING**) .
3. The specialty Bot is a LexBot (non QnABot) that indicates fulfillment is complete.
4. If the target Bot is another QnABot, session attributes can be set by the specialty QnABot set indicating the conversation should be discontinued. (**QNABOT_END_ROUTING)**

Specialty Bots can be developed for specific parts of an organization like IT, or Finance, or Project Management, or Documentation. A supervisory Bot at an enterprise level can direct users to answers from any of their Bots.

Message Protocol for a new Bot Router implemented in Lambda

The input json payload to the target Lambda will be the following:

```
 req: {
        request: "message",
        inputText: <String>,
        sessionAttributes: <Object>),
        userId: <String>
    }
```

The expected response payload from the target lambda is the following:

```
{
    response: "message",
    status: "success", "failed"
    message: <String>,
    messageFormat:  "PlainText", "CustomPayload", "SSML", "Composite"
    sessionAttributes: Object,
    sessionAttributes.appContext.altMessages.ssml: <String>,
    sessionAttributes.appContext.altMessages.markdown: <String>,
    sessionAttributes.QNABOT_END_ROUTING: <AnyValue>
    responseCard: <standard Lex Response Card Object>
}
```

Sample bot router

The Nutrionix nodejs based sample BotRouter is provided in the github repo as a zip file at sample bot router . To use this sample you'll need to provision an API account with Nutritionix and configure the source to use your own x-app-id and x-app-key from Nutritionix.

```
          'x-app-id': process.env.xAppId,
          'x-app-key': process.env.xAppKey
```

Next, you will need to build and deploy the code into Lambda using your favorite techniques and grant permission within the QnABot Fulfillment Lambda Role using IAM to invoke this Lambda function.

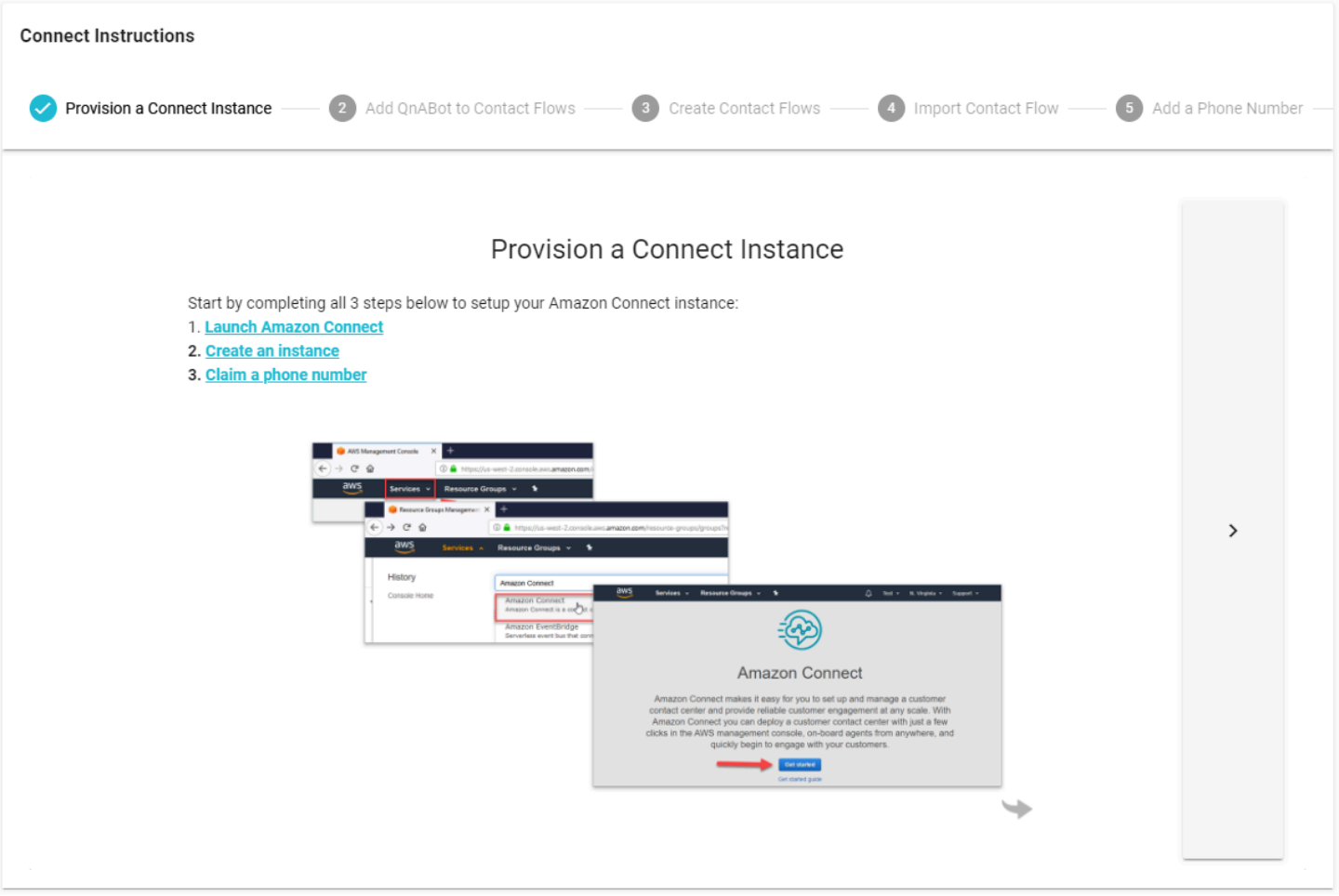
Hint: If you name the Lambda function starting with 'qna', QnABot is already configured with permissions to invoke this Lambda.

Connecting QnABot on AWS to an Amazon Connect call center

The solution can automate data collection and answer frequently asked questions using QnABot on AWS 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 QnABot on AWS to call their phone so they can speak to a human.

Using the Amazon Connect integration wizard, follow these steps to connect QnABot on AWS 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.



Amazon Connect integration wizard

Configuring intent and slot matching

QnABot supports different types of question and answer workflows. For example:

- You can create a question and answer experience to help answer frequently asked questions. In this model, the user asks a question and QnABot on AWS responds with the most relevant answer to the question (from the list of created Item IDs). For more information refer to Step 2. Populate the chatbot with your questions and answers.
- Build a diagnostic or questionnaire-based workflow, where a question from a user can result with QnABot on AWS asking follow-up questions. If you are creating a survey or building a diagnostic workflow where you may require inputs to different questions, you can use the ResponseBots and Document Chaining capabilities of QnABot. For more information, refer to Configuring the chatbot to ask the questions and use response bots.

Both of these options provide flexibility in creating an interactive chat experience. For example:

- Accepting dynamic user input in a question.
- Automatically asking a question for a given input without needing to setup document chaining.
- Validating user input against an available list of options.

With this early implementation of the intent and slot matching capability in QnABot on AWS, you can now build a richer conversational experiences. For example, you might create an intent that makes a car reservation, or assists an agent during a live chat or call (via Amazon Connect). You can use intent and slot matching also for cases where you might want better intent matching via Amazon Lex Natural Language Understanding (NLU) engine, as an alternative to QnABot default OpenSearch Service queries.

> **Note**
>
> The intent and slot matching capability in QnABot was initially implemented in version 5.2.0. The content and step-by-step procedures in this section apply to QnABot on AWS versions 5.2.0 and later.

Item ID setup

The **Item ID** setup is made of the following attributes:

- **Intent** – Represents an action that the user wants to perform. For each intent, you provide the following required information:

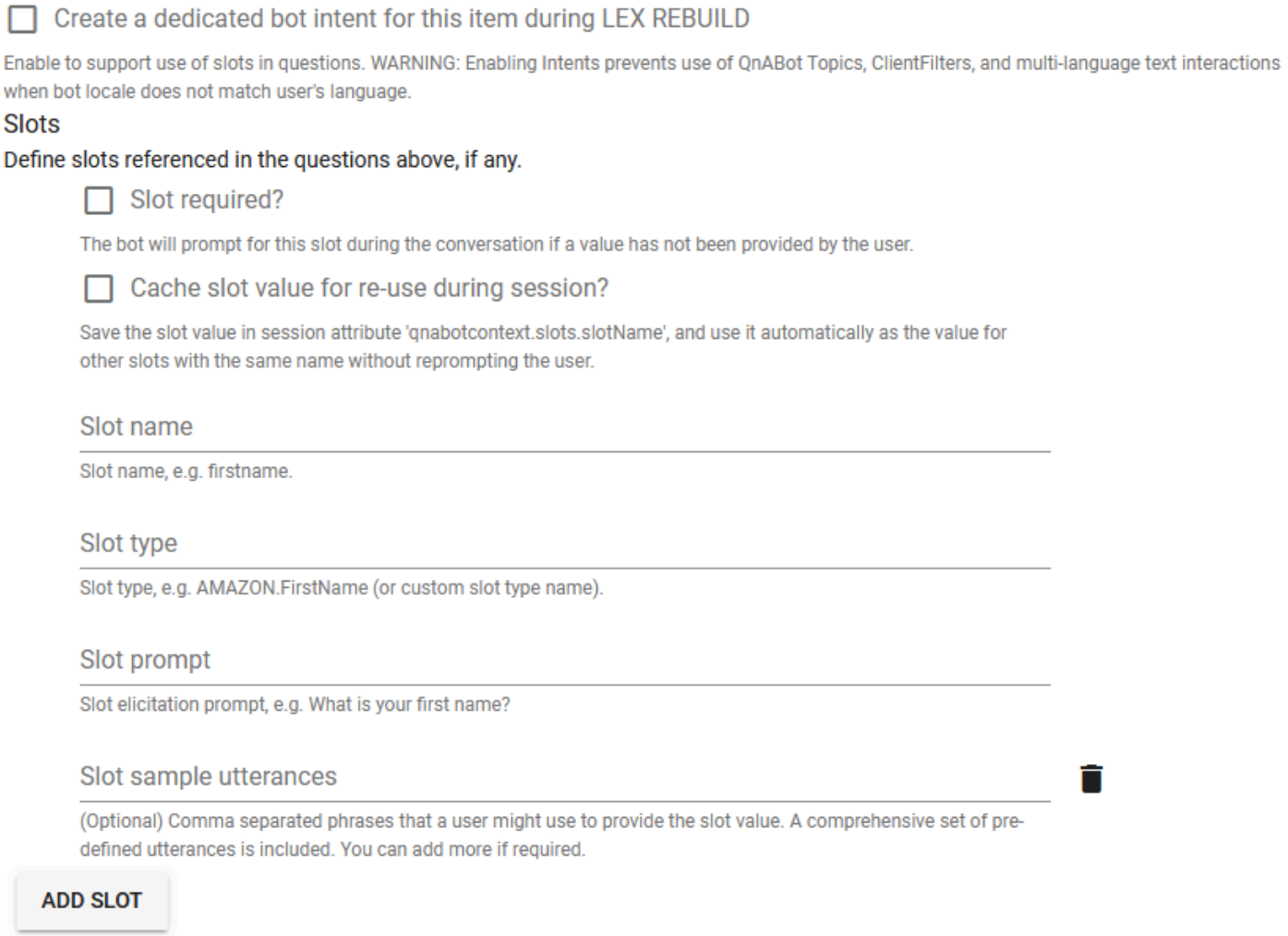
- **Intent name** – Descriptive name for the intent by providing a Item ID. For example: IntentSlotMatching.Example.Q1.
- **Sample utterances** – The intent a a user might convey. For example, a user might say "book a car" or "make a car reservation".

- **Slot** – An intent can require zero or more slots, or parameters. You add slots as part of the Item ID configuration. At runtime, Amazon Lex V2 prompts the user for specific slot values. The user must provide values for all required slots before Amazon Lex V2 can fulfill the intent.
- **Slot type** – Define the values that users can supply for your intent slots. Each slot has a type. You can create your own slot type, or you can use built-in slot types.

Creating custom intent with slots and slot types

1. Create a QnABot question as you would normally do by providing an Item ID and Questions/ Utterances.
2. Expand the **Advanced** option.
3. Check the option for Create a dedicated bot intent for this item during LEX REBUILD.

- Slots can be configured to be either required or optional. If a conversation flow requires user input, choose the **Slot Required** option.
- For each slot, provide the slot type and one or more prompts that Amazon Lex V2 sends to the client to elicit values from the user. A user can reply with a slot value when input may be needed. You can create your own custom slot type, or you can use built-in slot types.



Intent and slot configuration

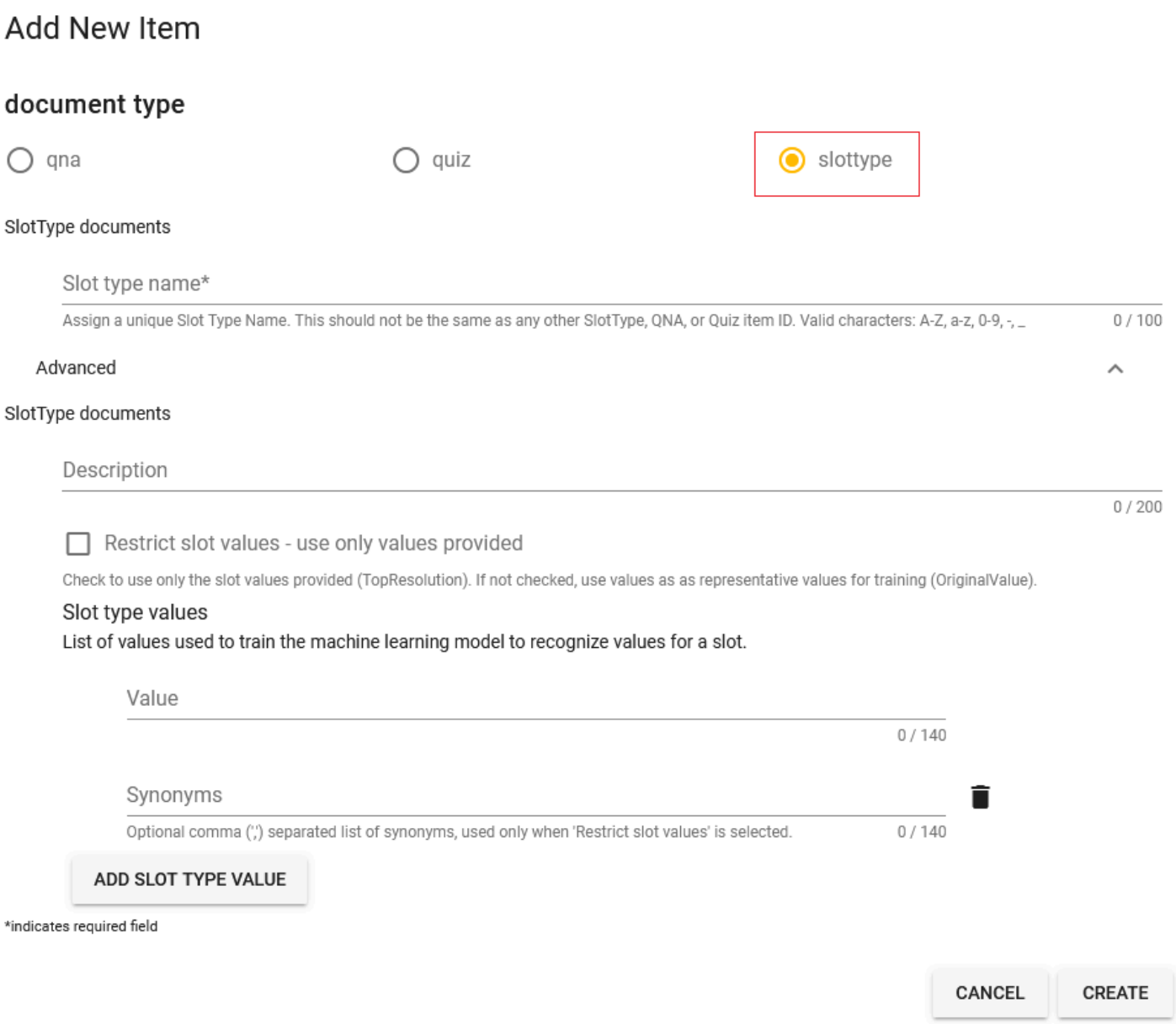
- A slot can also include optional sample utterances. These are phrases that a user might use to provide the slot value. A comprehensive set of pre-defined utterances is included (via built-in slot type or a custom slot type). You can add more if required. In most cases, Amazon Lex can understand user utterances. If you know a specific pattern that users might respond to an Amazon Lex request for a slot value, you can provide those utterances to improve accuracy. In most cases, you won't need to provide any utterances.
- Cache slot value for re-use during a session. The slot value can be stored in session variables and accessed via **qnabotcontext.slots.slotName**. When a slot value is stored in a session attribute, it is used automatically as the value for other slots with the same name without reprompting the user. This can be beneficial when you are capturing a user's profile information to support

different conversational workflows, and don't want to ask the same profile information again from the user.

Creating custom slot types

In addition to using built-in slot types, you can also create custom slot types. If an intent requires a custom slot type, you can create a custom slot type by creating a new Item and choosing the type **slottype**. Similar to built-in slot types, a custom slot type can be used across more than one intent.

- **Slot type values** – The values for the slot. If you chose Restrict to slot values, you can add synonyms for the value. For example, for the value "football" you can add the synonym "soccer." If the user enters "soccer" in a conversation with your bot, the actual value of the slot is "football."
- **Slot value resolution** – Determines how slot values are resolved. If you choose not to Restrict to slot values, Amazon Lex V2 uses the values as representative values for training. If you choose to Restrict to slot values, the allowed values for the slot are restricted to the ones that you provide.



Creating a custom slot type

Accessing slot values

To support a conversational experience, you might want to:

- Display what the user provided as slot values, such as workflows that require confirming user input.
- Use the slot values to support conditional branching via document chaining.
- Display a summary such as an order summary.

There are few ways you can access slot values within an Item ID and/or Lambda hook.

Using Handlebars you can access slot values using:

- `getSlot` – A new helper function that returns named slot value if it is defined, or default value. For example: `{{getSlot '_slotName_' '_default_'}}`.
- `Session attribute {{qnabotcontext.slots._slotName_}}` – A value in a session attribute, where `_slotName_` is the name of the slot defined in a Item ID. If the slot value is cached for re-use, the value is available in a session attribute, and can be used across item IDs.
- `{{Slots._slotName_}}` – A slot name, where `_slotName_` is the name of the slot defined in an item ID.

Import sample intent and slot types

In the **QnABot Designer**, choose the **Tools** menu link on the top left and select **Import**. From the **Examples/Extensions** section, click **Load** for IntentSlotMatching to load sample intent and slot types. This example imports:

- **IntentSlotMatching.Example.Q1** – An Item ID of type **qna** with custom intent and slot.
- **IntentSlotMatching_Example_slottype_CarType and IntentSlotMatching_Example_slottype_Confirmation Item ID** of type slottype with sample slot values.

Lex rebuild

Once you have loaded the questions, choose **Edit** from the Tools menu and choose **LEX REBUILD** from the top right edit card menu ⋮. This re-trains Amazon Lex using the newly added questions as training data.

Testing the experience

On the **Tools** menu and choose **QnABot Client** from the options. Try the below conversation flow:

```
User: Book a car
Bot: In what city do you need to rent a car?

User: Seattle
Bot: What day do you want to start your rental?
```

```
User: Today
Bot: What day do you want to return this car?

User: Next Sunday
Bot: What type of car would you like to rent? Our most popular options are economy,
midsize, and luxury.

User: Economy
Bot: Okay, should I go ahead and book the reservation?

User: Yes
Bot: Okay, I have confirmed your reservation. The reservation details are below:
     Car Type: economy
     Pick up City: Seattle
     Pick up Date: 2022-05-30
     Return Date: 2022-06-12
```

Note that the user is not prompted with the question `What day do you want to start your rental?`, because the slot value was already provided by the user in the utterance.

Notes and considerations

- Utterances must be unique across intents. Duplicate utterances across intents will cause the Amazon Lex build to fail. Suppose you have two intents `OrderPizza` and `OrderDrink` in your bot and both are configured with an `I want to order utterance`. This utterance does not map to a specific intent that Amazon Lex V2 can learn from while building the language model for the bot at build time. As a result, when a user inputs this utterance at runtime, Amazon Lex V2 can't pick an intent with a high degree of confidence.
- Topics and ClientFilters are not supported when an Item ID is activated with custom intent.
- Bot locale must be set to user's locale for QnABot on AWS multi-language text interactions.
- Always initiate a LEX REBUILD when activating Item IDs with custom intent and slots. This creates the custom intents, slots, and slot types in Amazon Lex V2, and also trains Amazon Lex using the added/updated Item IDs as training data.
- To take advantage of the additional features supported by Amazon Lex, such as confirmation prompts and regular expression to validate the value of a slot, you can also create the Amazon Lex intents and slot types in the QnABot Lex bot using the Amazon Lex console. For more information, refer to Adding intents in the *Amazon Lex V2 Developer Guide*.

- Even if the Amazon Lex intents and slot types are created in the Amazon Lex console (created outside of the **QnABot designer**), you can reference any SlotType defined in the bot in a QnABot Item ID, and also map a QID to a manually created Amazon Lex intent in QnABot on AWS.
- The **Test All** or **Test** options don't work correctly for Item IDs with custom intent.
- As you are building your knowledge bank of questions, you might have a combination of FAQ based questions and intent based questions. There may be instances where a wrong intent gets matched or a FAQ question is matched instead. To troubleshoot this issue, try the following:

- Enable the **ENABLE_DEBUG_RESPONSES** setting in QnABot. This setting provides debug information to help understand what is processing the request (such as Intent, OpenSearch Service, or Amazon Kendra).

Additional example implementation

Review the Integration with Canvas LMS example. It is an early example implementation showcasing the use of intent and slot matching.

Setting up a custom domain name for QnABot Designer and Client

This section provides information on how to set up a custom domain name and configure the QnABot on AWS solution to use the custom domain name for the Designer and Client user interfaces. The setup and configuration involve the following steps.

Step 1: Set up custom domain name for API Gateway

Use the AWS account and Region where you have deployed the QnABot on AWS solution for the below steps. These steps are explained in the AWS API Gateway documentation.

- Registering a domain name.
- Creating DNS records.
- Creating a SSL certificate for the custom domain name.
- Creating a custom domain in API Gateway.

> **Note**
>
> Deactivate the default API gateway endpoint since the custom domain name is used.

Step 2: Custom domain API Mapping setup in API Gateway

When mapping the API to the custom domain in API Gateway for the QnABot deployment, use the following settings:

Mapping 1

- **API** – Select the QnABot deployment you would like to use. The QnABot API takes on the same name as the CloudFormation Stack name you used when you deployed the QnABot on AWS solution.
- **Stage** – Use **prod**. This is the default stage created for the QnABot deployment.

Mapping 2

- **API** – Select the QnABot deployment you would like to use. The QnABot API takes on the same name as the CloudFormation Stack name you used when you deployed the QnABot on AWS solution.
- **Stage** – Use **prod**. This is the default stage created for the QnABot deployment.
- **Path** – Use **prod**. This is used for routing requests.

Step 3: Update QnABot API Resources in API Gateway

1. Navigate to API Gateway and select the QnABot API.
2. The QnABot API takes on the same name as the CloudFormation Stack name you used when you deployed the QnABot on AWS solution.
3. Navigate to the Resources section from the menu.

Step 3a: Update the /pages/client resource

1. Select the GET method for the /pages/client resource.
2. Choose **Integration Response**.
3. Expand the **302 Method Response Status**.
4. Edit the location Response header and replace the API Gateway endpoint with your custom domain name The API Gateway endpoint has an endpoint such as: `{api-id}.execute-api.{region}.amazonaws.com`.

5. Make a note of the URL encoding in the values.
6. Choose the **{tick}** icon to update the value.
7. Choose **Save**.

Step 3b: Update the /pages/designer resource

1. Select the GET method for /pages/designer resource.
2. Choose **Integration Response**.
3. Expand the **302 Method Response Status**.
4. Edit the location Response header and replace the API Gateway endpoint with your custom domain name The API Gateway endpoint will have the endpoint such as:`{api-id}.execute-api.{region}.amazonaws.com`.
5. Make note of the URL encoding in the values.
6. Choose the **{tick}** icon to update the value.
7. Choose **Save**.

Step 4: Update QnABot Cognito user pool

To access the **QnABot Designer** user interface, the deployment sets up authentication using Amazon Cognito. Update the user pool settings to update the Callback URLs to use the custom domain name.

1. Navigate to the Amazon Cognito console.
2. Choose **User Pools**.
3. Choose the **QnABot** user pool.
4. The QnABot user pool takes on the same name as the CloudFormation stack name you used when you deployed the QnABot on AWS solution. For example, UserPool-*<stack name>*
5. Navigate to **App Integration | App client** settings.
6. Update the callback URLs for app clients: `UserPool-<stack name>-client`
7. Use the custom domain name instead of the API Gateway endpoint. For example: `https://<your-custome-domain-name>/prod/static/client.html`.
8. Choose **Save Changes**.

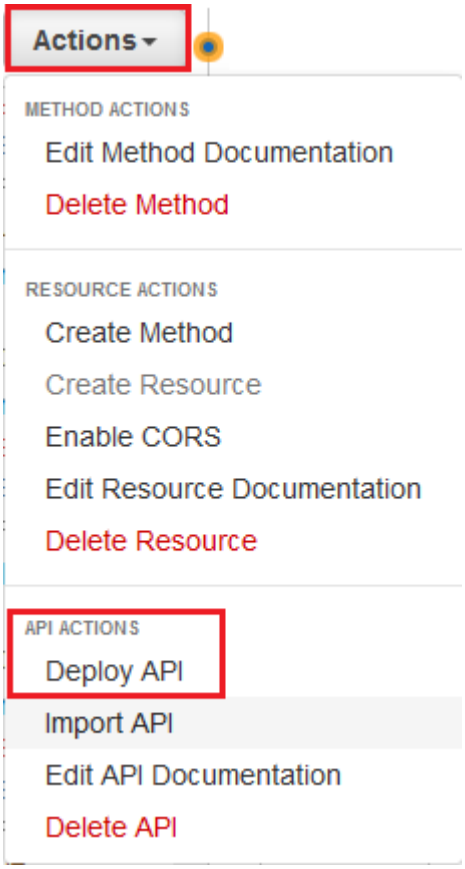
Update the callback URLs for app clients: UserPool-{stackname}-designer

1. Use the custom domain name instead of the API Gateway endpoint. For example: `https://<your-custome-domain-name>/prod/static/index.html`.
2. Choose **Save Changes**.

Step 5: Deploy the API

Now that we have updated the configurations, we will deploy the API for the changes to take effect.

1. Choose **Actions**.
2. Choose **Deploy API**.



Deploy API action

3. Choose the following:
   - Deployment stage: **prod**.

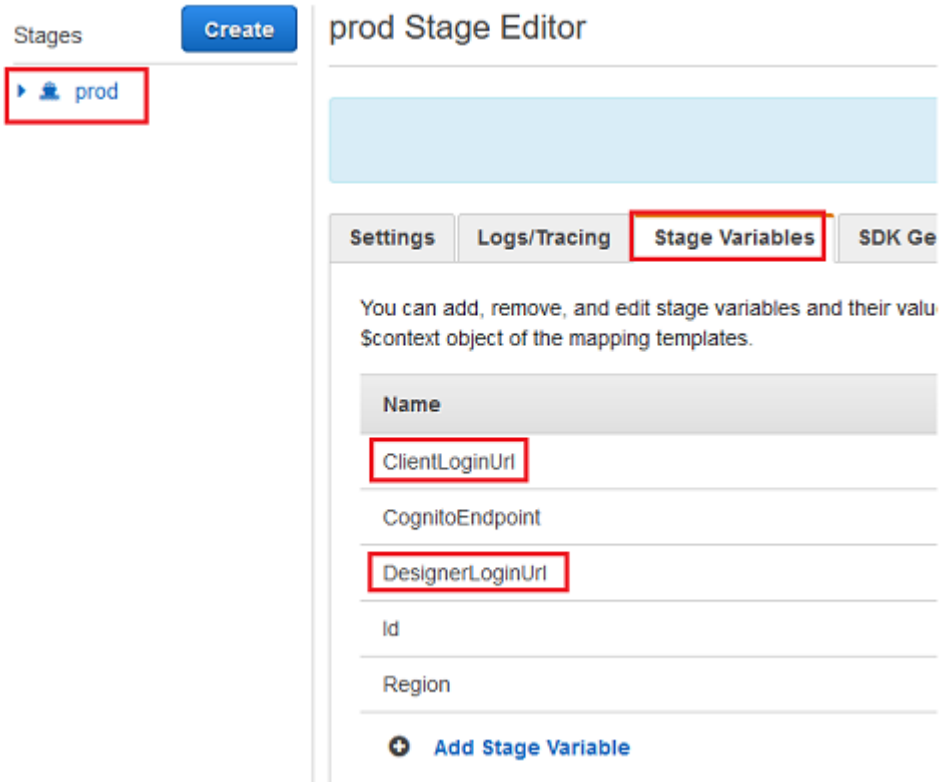
- Deployment description: enter **Updated location** response header in the GET method for the `/pages/designer` and the `/pages/client` resources.

4. Choose **Deploy**.

Step 6: Update the API Stage variables

Once the API is deployed, the Stage Editor page appears.

- Choose the **Stage Variables** tab.
- Update the values for **ClientLoginUrl** and **DesignerLoginUrl** variables to use the custom domain name.



Update stage variables

Step 7: Test the updates using the custom domain name

Launch the **QnABot Designer** in a new browser session using the custom domain name `https://<your-custom-domain-name>/prod/pags/designer to test the updates.`

Known limitation

A Cloudformation stack update of QnABot performed after the above steps, will overwrite the changes made in Steps 3, 4, 5, and 6. We are looking at better ways to automate this process, but in the meantime, if you perform a stack update after the above steps, you will need to manually re-apply the above steps 3, 4, 5, and 6 again.

Using QnABot on the AWS Command Line Interface (CLI)

The QnABot on AWS CLI supports the capability to import and export questions and answers from your QnABot setup.

Setup prerequisites

To use the CLI, the following prerequisites are required:

- Download the source directory from codebase of the QnABot on AWS solution (version 5.2.0 or higher) in GitHub
- AWS Command Line Interface (CLI).
- Python version 3.7 or higher. For more information on installing Python, see Python Setup and Usage
- AWS IAM permissions having the below IAM policy. Attach the below IAM policy to the IAM user or IAM Role that you are using for the AWS CLI. Replace the following values when creating the IAM policy:

- AWS_REGION – The AWS Region where you have deployed the QnABot on AWS solution.
- AWS_ACCOUNT_ID – The AWS Account ID where you have deployed the QnABot on AWS solution.
- YOUR_QNABOT_IMPORT_BUCKET_NAME – The name of the QnABot on AWS import bucket name. This can be found by navigating to the Resources section (in AWS CloudFormation) of the deployed QnABot on AWS CloudFormation template.
- YOUR_QNABOT_EXPORT_BUCKET_NAME – The name of the QnABot on AWS export bucket name. This can be found by navigating to the Resources section (in AWS CloudFormation) of the deployed QnABot on AWS CLoudformation template.
- YOUR_QNABOT_STACK_NAME – The name of the QnABot on AWS stack that you deployed via AWS CloudFormation.

IAM Policy

```
{
    "Version": "2012-10-17",
    "Statement": [
        {
            "Sid": "S3ReadWriteStatement",
            "Effect": "Allow",
            "Action": [
                "s3:PutObject",
                "s3:GetObject"
            ],
            "Resource": [
                "arn:aws:s3:AWS_REGION:AWS_ACCOUNT_ID:YOUR_QNABOT_IMPORT_BUCKET_NAME/
*",
"arn:aws:s3:AWS_REGION:AWS_ACCOUNT_ID:YOUR_QNABOT_EXPORT_BUCKET_NAME/*",
            ]
        },
        {
            "Sid": "CloudFormationDescribeStatement",
            "Effect": "Allow",
            "Action": "cloudformation:DescribeStackResource",
            "Resource": "arn:aws:cloudformation:AWS_REGION:AWS_ACCOUNT_ID:stack/
YOUR_QNABOT_STACK_NAME/*"
        }
    ]
}
```

Environment setup

Get started by creating a virtual environment and deploy the needed Python packages. From a directory outside of the QnABot on AWS codebase, run the following commands:

```
pip3 install virtualenv
python3 -m virtualenv .venv
source ./.venv/bin/activate
cd source
pip3 install -r requirements.txt
```

These commands set up a virtual environment and install the following Python packages:

- Boto3 Python module version 1.21.18. For more information, refer to AWS SDK for Python (Boto3).
- Choose Python module version 8.0.4. For more information, refer to the Python Click documentation.

Set Environment variables

Set the code for your Region. For example, to use the us-east-1 Region, run the following command:

```
export AWS_REGION='us-east-1'
```

Set the Python path using the following command:

```
export PYTHONPATH=${PWD}:$PYTHONPATH
```

Available commands

The >`qnabot_cli.py` file is located in the source/aws_solutions/qnabot/cli directory. Run `python3 aws_solutions/qnabot/cli/qnabot_cli.py` using the following syntax:

Usage:

`qnabot_cli.py [OPTIONS] COMMAND [ARGS]...`

Options:

`-h, --help Show this message and exit.`

Commands:

```
export - Export QnABot questions and answers from your QnABot setup.
```

```
import Import QnABot questions and answers to your QnABot setup.
```

Using the import command

Usage: `qnabot_cli.py` import [OPTIONS]

Import QnABot questions and answers to your QnABot setup. This command requires two (2) parameters: <cloudformation-stack-name>, and <source-filename>. The cloudformation-stack-name parameter is used to know the QnABot on AWS deployment to use to support the import process.

Options:

```
  -s, --cloudformation-stack-name TEXT
                                  Provide the name of the CloudFormation stack
                                  of your QnABot on AWS deployment  [required]
  -f, --source-filename TEXT      Provide the filename along with path where
                                  the file to be imported is located
                                  [required]
  -fmt, --file-format [JSON|JSONL|XLSX]
                                  Provide the file format to use for import
                                  [default: JSON]
  -d, --delete-existing-content BOOLEAN
                                  Use this parameter if all existing QnABot
                                  {qids} in your QnABot deployment should be
                                  deleted before the import process.
                                  [default: False]
  -h, --help                      Show this message and exit.
```

A successful import will output status with the below information:

```
{
    "number_of_qids_imported": <number>,
    "number_of_qids_failed_to_import": <number>,
    "import_starttime": <datetime in UTC>,
    "import_endtime": <datetime in UTC>",
    "status": "Complete",
    "error_code": "none"
}
```

Example:

```
{
    "number_of_qids_imported": 9,
    "number_of_qids_failed_to_import": 0,
    "import_starttime": "2022-03-20T21:39:28.455Z",
    "import_endtime": "2022-03-20T21:39:32.193Z",
    "status": "Complete",
```

```
    "error_code": "none"
}
```

Using the export command

Usage: qnabot_cli.py export [OPTIONS]

Export QnABot questions and answers from your QnABot setup. This command requires two (2) parameters: <cloudformation-stack-name>, and <export-filename>. The cloudformation-stack-name parameter is used to know the QnABot on AWS deployment to use to support the export process.

Options:

```
-s, --cloudformation-stack-name TEXT
                                Provide the name of the CloudFormation stack
                                of your QnABot on AWS deployment  [required]
  -f, --export-filename TEXT    Provide the filename along with path where
                                the exported file should be downloaded to
                                [required]
  -qids, --export-filter TEXT   Export {qids} that start with this filter
                                string. Exclude this option to export all
                                {qids}
  -fmt, --file-format [JSON|JSONL]
                                Provide the file format to use for export
                                [default: JSON]
  -h, --help                    Show this message and exit.
```

A successful import will output status with the below information:

```
{
    "export_directory": <string>,
    "status": "Downloaded",
    "comments": <string>,
    "error_code": "none"
}
```

Example:

```
{
    "export_directory": "../export/qna.json",
```

```
    "status": "Downloaded",
    "comments": "Check the export dhirectory for the downloaded export.",
    "error_code": "none"
}
```

Running qnabot_cli.py as a shell script

import example

```
#!/bin/bash
export AWS_REGION='us-east-1'
shell_output=$(python3 qnabot_cli.py import -s qnabot-stack -f ../import/
qna_import.json -fmt json)
STATUS="${?}"
if [ "${STATUS}" == 0 ];
then
    echo "AWS QnABot import completed successfully"
    echo "$shell_output"
else
    echo "AWS QnABot import failed"
    echo "$shell_output"
fi
```

export example

```
#!/bin/bash
export AWS_REGION='us-east-1'
shell_output=$(python3 qnabot_cli.py export -s qnabot-stack -f ../export/
qna_export.json -fmt json)
STATUS="${?}"
if [ "${STATUS}" == 0 ];
then
    echo "AWS QnABot export completed successfully"
    echo "$shell_output"
else
    echo "AWS QnABot export failed"
    echo "$shell_output"
fi
```

Integration with Canvas Learning Management System (LMS)

Students use their schools' Canvas LMS to keep track of their assignments, grades, and working through their course work. To make it easier for students to stay on track and also have easy access to a knowledge base, and help with their learning progress, you can integrate the open-source QnABot on AWS solution with Canvas LMS, and support students with in-the-moment support. With this integration, students are able to ask the chatbot about their grades, syllabus, enrollments, assignments, and announcements.

Prerequisites

There are a few prerequisites to get started with the setup:

1. Setting up up Canvas LMS requires a running Canvas LMS environment (on-premises or /AWS environment). If you do not have Canvas LMS, you can install by following the instructions on GitHub repository.
2. Set up the open-source QnABot on AWS solution deployed in your AWS environment. If you do not have this setup or are running an older version of QnABot on AWS, you can install or upgrade by following the QnABot on AWS implementation guide.
3. Set up a companion Web UI for the chatbot. You can deploy this using the open source Lex-Web-UI project in your AWS account by following the steps outlined in this blog post.
   - During this setup, set the **EnableLogin** setting to true. This enables authentication in the chatbot and connect to an Identity provider.
   - For **BotName** and **BotAlias**, use the bot name and bot alias obtained from the QnABot on AWS solution deployment outputs.

Creating and storing the Canvas API access token

The QnABot on AWS solution uses the Canvas API to integrate with Canvas LMS. To configure the QnABot on AWS solution, follow these steps:

1. Create a new Canvas API access token. For more details on how to create a Canvas API access token, refer to How do I manage API access tokens as an admin?

2. Store the Canvas API access token in AWS Secrets Manager. With Secrets Manager you can replace hardcoded credentials in your code, including passwords. To retrieve the secret programmatically, you make an API call to Secrets Manager. This helps ensure the secret can't be compromised by someone examining your code, because the secret no longer exists in the code.
   a. Open to the Secrets Manager console. Use the same AWS region where you deployed the QnABot on AWS solution.
   b. Choose Store a new secret.
   c. For **Key** name, choose the key **API_Token**.
   d. For **Value**, copy and paste the Canvas API access token value that you created earlier.
   e. Enter a descriptive **Secret name** and **Description**. Start the name with the letters **qna-**. For example, **qna-CanvasAPIKey**.

Configure QnABot on AWS settings

After you have deployed the QnABot on AWS solution, you will have access to the **QnABot Designer** console, which allows you to create and manage your knowledge bank of questions and answers.

1. Open the link that you received in your email and sign in to your ot DeQnABsigner console.
2. Choose the menu located in the top left corner of the designer console. The **Tools** option list appears.
3. Scroll to the bottom of the page and choose **ADD NEW SETTING**. Use this to store the Secrets Manager key name that you created in the above steps, so QnABot can know how to connect to Canvas LMS. Enter the New Setting values:
   - **Name** – Enter `CanvasLMS_APIKey` as the name.
   - **Value** – Use the name of the Secrets Manager key that you created in the above steps for storing the Canvas API key value. For example, `qna-CanvasLMSAPIKey`.
   - Choose **ADD** to add the new QnABot setting.
4. Create another setting.
   - **Name** – Enter `CanvasLMS_DomainName` as the name.
   - **Value** – Use the value of your Canvas endpoint. For example, `https://lms.myschool.edu`
   - Choose **ADD**.

5. Update the **IDENTITY_PROVIDER_JWKS_URLS** setting to add trusted **Identit .Providers**. For example: from your Lex-Web-UI Cloudformation Outputs, using the *CognitoUserPoolPubKey* value.
6. Scroll to the bottom of the Settings page and choose **Save** to update the setting.

Set up authentication

As part of the prerequisite setup, we set up the Lex-Web-UI (a companion UI solution for the chatbot) and configured the solution with the QnABot solution. The deployment sets up an Amazon Cognito User Pool to support authentication. We will now extend this User Pool to add a test student user and test out the chatbot flow.

1. Open https://console.aws.amazon.com/.
2. Navigate to the Amazon Cognito service.
3. Select **Manage User Pools**. Two user pools have already been created when you followed the steps in Setup Prerequisites earlier in this document. We will use is the Lex-Web-UI user pool.
4. Select the **Lex-Web-UI** user pool and create a test student user. Also use an {email address} as created in the Canvas LMS for the test student user.

> **Note**
>
> In this example, we are creating the user manually in Amazon Cognito. This manual user creation step is not be needed if you want to use single sign-on to access Canvas LMS.
> For more information on setting up Canvas LMS using single sign-on, refer to How do I configure SSO settings for my authentication provider?
> In this example, we are using username as the matching attribute with `sis_login_id`in Canvas LMS.
> If you want to federate single sign-on into the QnABot Designer UI and Kibana using IAM Identity Center, refer to Using QnABot Designer with IAM Identity Center.

Import Canvas questions

In the **QnABot Designer**, select the menu link on the top left and choose **Import**. From the **Examples/Extensions** section, select **Load for CanvasLMSIntegration** to load sample Canvas questions.

Amazon Lex Rebuild

Once you have loaded the questions, choose **Edit** from the **Tools** menu and choose LEX REBUILD from the top right edit card menu (⋮). This will re-train Amazon Lex using the newly added questions as training data.

Testing the experience

Launch the WebAppUrl URL as available in the Lex-Web-UI AWS CloudFormation Output and Login to the chatbot from the menu option. Use the test student Canvas LMS credential that you created in the earlier steps to login and test the setup.

Type or speak the below question(s) and see how the chatbot responds back with an answer.

- Canvas menu
- Do I have any announcements?
- Tell me about my syllabus
- Do I have any assignments due?
- What courses have I enrolled in?
- More info about my course
- What are my grades?

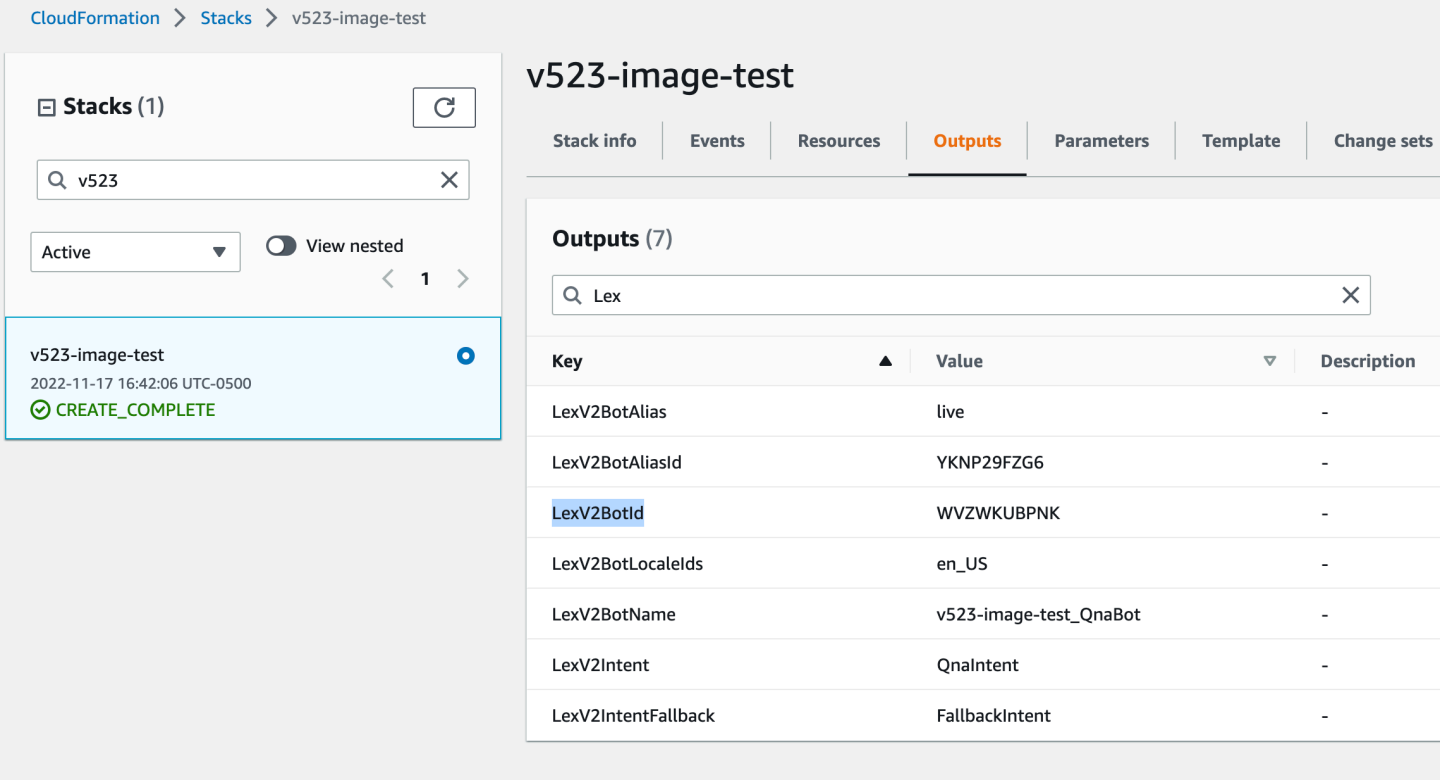
> **Note**
>
> This early example implementation supports English (en_US) language.

Deploy a custom web UI for your chatbot

QnABot on AWS includes a built-in web UI that you can use as is. Refer to section Interact with the chatbot earlier in this guide.

After you finish building your QnABot, you can separately deploy the Amazon 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 and the sample Amazon Lex web interface in the QnABot Github repository.

To deploy the Amazon Lex web UI, you must know the IDs of the Amazon Lex bots. You can find the IDs in the **Outputs** tab of the QnABot CloudFormation template.



Amazon Lex bot IDs in the CloudFormation Outputs tab

Canvas API reference

The following Canvas APIs are being used for this integration:

- User profile – To support authentication, and greeting the user.

- Grades – Student can ask questions such as *"How did i do in my Math course?"*. This supports the overall grade information (out of 100) which is aggregated by course (not by assignments).
- Course – Students can ask questions such as: *"What are my assignments for Biology 101"*
- Syllabus – To access syllabus information. The output of this is a URL to the syllabus. Student can ask about their syllabus by asking *"Tell me about my syllabus"*.
- Enrollment – Students can ask questions such as: "What courses am i enrolled in", "what courses have i signed up for".
- Announcements – anything sent by the teacher to student(s) such as: "*You have a test coming up*.". Student can ask by saying "*Do I have any announcements?*"

This integration uses the canvasapi python library to access information from Canvas LMS.

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 more information.

Reference

This section includes pointers to related resources and a list of builders who contributed to this AWS 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

- **Version** - The AWS solution version
- **Timestamp** - Data-collection timestamp
- **Region** - The AWS Region where the solution was deployed

AWS owns the data gathered though this survey. Data collection is subject to the Privacy Notice. 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. Search for S00189 and modify the AWS CloudFormation template description field to remove the solution ID. The template should be modified from:

```
"AWSTemplateFormatVersion": "2010-09-09",
"Description": "(SO0189) QnABot with admin and client websites - Version
 v5.4.5","AWSTemplateFormatVersion": "2010-09-09",
"Description": "QnABot with admin and client websites - Version v5.4.5",
```

   to:

```
"AWSTemplateFormatVersion": "2010-09-09",
"Description": "QnABot with admin and client websites - Version v5.4.5",
```

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

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
- Building a multilingual question and answer bot with Amazon Lex

Workshop

- QnABot on Workshop

YouTube demo

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

Contributors

The following individuals contributed to this document:

- Ibrahim Mohamed
- Tarek Abdunabi
- Alireza Assadzadeh

- Bob Strahan
- Bob Potterveld
- Chris Lott
- John Calhoun
- Karl Thomas
- Raj Chary
- Mohsen Ansari
- Marc Burnie
- Michael Lin
- Abhishek Patil
- Fabien Houeto
- Abhay Joshi

Revisions

Date	Change
September 2021	Release v5.0.0 – Initial AWS Solutions Implementation release. For more informati on, refer to the CHANGELOG.md file in the GitHub repository.
October 2021	Release 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.
December 2021	Release v5.1.0 – Integration with Genesys. Fixed integration with Slack and LexV2. Intelligent PII redaction with Amazon Comprehend. Bug fix for Amazon Kendra FAQ and metadata tags for questions. Added client filter support to allow same questions answered differently based on session attributes. For more information, refer to the CHANGELOG.md file in the GitHub repository.
February 2022	Release v5.1.1: Expanded language support for voice and text interactions, included support for Neural voices for Amazon Lex language locales, fixed Amazon Kendra Webcrawler data source sync issues. For more information, refer to the CHANGELOG.md file in the GitHub repository.
March 2022	Release v5.1.2: Added logic to support Amazon Connect Interactive Messages and a new set of example questions to be imported for Genesys Cloud CX. For more information,

Date	Change
	refer to the CHANGELOG.md file in the GitHub repository.
July 2022	Release v5.2.0: This release includes: early implementations of intent and slot matching and Canvas LMS integration; support for using custom domain names in QnABot on AWS Designer and Client interfaces; Command Line Interface (CLI) for QnABot; Amazon Kendra Redirect capability; ability to import QnABot questions and answers from an Excel file uploaded to an S3 data folder; support for importing session attributes using Excel files; bugs fixes and updates. For more information, refer to the CHANGELOG.md file in the GitHub repository.
September 2022	Release v5.2.1: This release includes security patches, changes for the AWS Lambda release that supports the Node.js 16 runtime, and a bug fix for the error caused by not providing an image URL in the Bot's response card. For more information, refer to the CHANGELOG .md file in the GitHub repository.
October 2022	Release v5.2.2: This release includes npm and pip security patches; improved deploymen t stability for ElasticSearch and Amazon Lex resources creation; single character utterance bug fix; and ElicitResponse bug fix. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
November 2022	Release v5.2.3: This release includes npm and pip security patches. For more information, refer to the CHANGELOG.md file in the GitHub repository.
November 2022	Release v5.2.4: This release includes npm and pip security patches. For more information, refer to the CHANGELOG.md file in the GitHub repository.
December 2022	Release v5.2.5: This release includes npm and pip security patches; documentation improvements; new unit tests; clientfilter bug fix; Kendra FAQ bug fix; missing Fulfillment Lambda widget fix; and support has been added for the latest LexV2 languages. For more information, refer to the CHANGELOG.md file in the GitHub repository.
January 2023	Release v5.2.6: This release includes npm and pip security patches. For more information, refer to the CHANGELOG.md file in the GitHub repository.
February 2023	Release v5.2.7: This release includes npm and pip security patches; and new unit tests. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
February 2023	Release v5.3.0: This release moves the solution onto Opensearch v1.3 and introduces new QnA search capabilities using text embeddings. By enabling text embeddings, users can leverage large language models to obtain semantic-based query matching. For more information, refer to the CHANGELOG.md file in the GitHub repository.
March 2023	Release v5.3.1: This release includes npm and pip security patches, and a bug fix for the fulfillment Lambda function not correctly publishing a new version. For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2023	Release v5.3.2: This release includes npm and pip security patches; a bug fix for Alexa skill reprompts; new CloudFormation parameter to configure EBS volume size for opensearch; `MetricsBucket` added to CF output; updates to Amazon Lex and Amazon Connect response limits; and miscellaneous documentation updates. For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2023	Release v5.3.3: This release includes npm security patches. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
May 2023	Release v5.3.4: This release includes npm and pip security patches and a bug fix for Amazon Connect voice responses. For more informati on, refer to the CHANGELOG.md file in the GitHub repository.
July 2023	Release v5.3.5: This release includes pip security patches and removal of the ElasticSe archUpdate custom resource to prevent CFNLambda recursion alert. For more information, refer to the CHANGELOG.md file in the GitHub repository.
July 2023	Release v5.4.0: This release introduces additional QnA search capabilities using Large Language Models (LLMs). By enabling LLMs, end users can leverage additional features such as query disambiguation, text generatio n to answer questions from a Kendra index or from the new text passage item type. This release also updates the Lambda Runtimes to Nodejs18 and Python 3.10 and adds initial support for AppRegistry integration. For more information, refer to the CHANGELOG.md file in the GitHub repository.
August 2023	Release v5.4.1: This release includes minor documentation updates to the LLM README, and additional LLM guidance in the Implement ation Guide. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
September 2023	Release v5.4.2: This release includes minor updates and bug fixes. For more information, refer to the CHANGELOG.md file in the GitHub repository.
October 2023	Release v5.4.3: This release includes fix for an issue where Alexa schema was not exporting the utterances list. and additional documenta tion on bot routing configuration. Documentation additions include data storage and protection, guidance section for implementing quizzes, PII Redactions, Multi-language support & Bot Routing. For more information, refer to the CHANGELOG.md file in the GitHub repository.
October 2023	Release v5.4.4: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.
November 2023	Release v5.4.5: Updated package versions to resolve security vulnerabilities. For more information, refer to the CHANGELOG.md file in the GitHub repository.

Date	Change
January 2024	Release v5.5.0: This release introduces core-language parameter to the QnABot deployment that supports 33 Languages and allows the user to select a core language in which the OpenSearch language analyzers will be used. This provides a more syntactic al accuracy for matching questions and answers without resorting to translation. This release also has additional enhanceme nts like Bot routing, protected utterance s settings, functional test collection and improvements in error handling. This release also has updates like Bluebird migration to native promises, upgrade to AWS SDK for JavaScript v3, Wepack 5, Vue3 and Vuetify3. It also includes documentation updates, code quality improvements, security patches and bug fixes. For more information, refer to the CHANGELOG.md file in the GitHub repository.
April 2024	Release v5.5.1: This release fixes Document Chaining issues, updates the QnABot Client from using Cognito Auth Code instead of Implicit Grant, and includes patched vulnerabi lities as a part of these changes. 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.

QnABot on AWS is licensed under the terms of the of the Apache License Version 2.0 available at The Apache Software Foundation.