UPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESUPESqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmrtyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnmqwertyuiopasdfghjklzxcvbnm

|  |
| --- |
| https://s3.amazonaws.com/upes-prod/media/7026/upes-logo.png  **IdeaFrameWork and FlowChart**  **Cloud Application Development**  **Submitted to**  **Saurabh Shanu sir**  **Submitted By**  Name: Rahul Khandelwal  Batch: B3 , CCVT(NH)  Sap Id: 500086812  Enrollment No: R2142201700 |

**BUILDING AWS-LEX BOT**

**Concept Node :**

**WHAT IS AMAZON LEX ?**

Amazon lex is an aws service for creating communication platform for any application using voice/speech and text . It has advanced functionality that it can convert the user speech into text known as Automatic speech recognition . Natural language understanding is used to recognize the text . With amazon lex , deep learning technologies are used and created an Amazon Alexa which nowadays very highly popular in use , which enables quickly create any chatbots .

Nowadays half of all the companies are investing on chatbots rather than mobile apps as amazon lex helps them to reduce the cost of development and operations team while delivering a smarter automated live chat experiences .

**What are the reasons for using Amazon Lex for bots ?**

* Amazon lex fuels chatbot with sophisticated NLU and ASR .
* It is easy to use , one click deployment and integrates with aws ecosystem .
* AWS-LEX is powered chatbots in call centers , in apps , on facebook and within many enterprises .
* Cloud managed services for aws for implementation and maintenance .

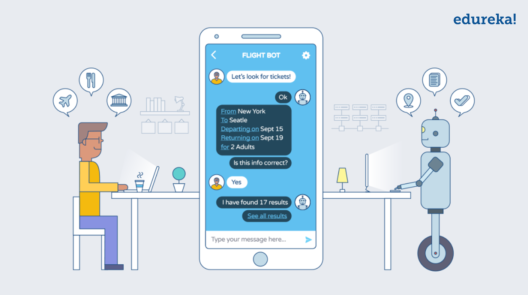
**Features and Benefits of AWS-LEX ?**

1. Cost effective
2. Deployment
3. Simplicity
4. AWS Platform integration
5. Text and Speech recognition

**Why Lex is unique and popularly used for chatbots ?**

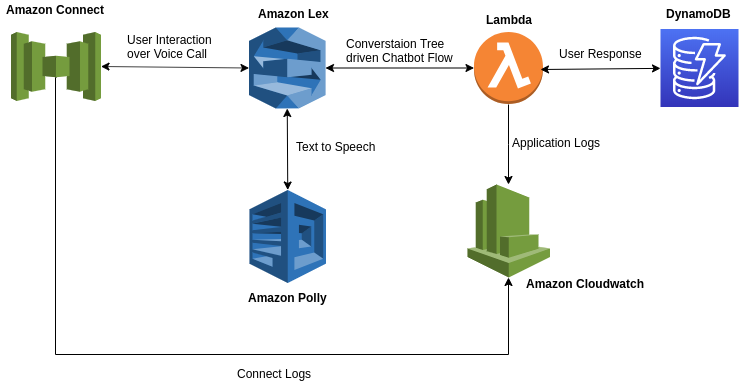
* It can scale through your business growth automatically .
* It acts as a platform fro building cross channel text and voice conversational interface .
* Applies deep learning for ASR .
* Build multi-turn conversations .

CloudHesive is used to manage Amazon lex implementation and beyond . CloudHesive is a customer obsessed Amazon premier partener specializing in consulting , managing services and SaaS platform . Amazon lex is the most user friendly platform for implementing and maintaining an intelligent chatbot . Work with CloudHesive will help to manage services that extend to cloud security , reliability , availability , scalability also with biggest benefit that is it can reduce the operation costs , improvement in productivity and running a better cloud chatbot .



* **AWS SDK** — When using the SDKs your requests to Amazon Lex are automatically signed and authenticated using the credentials that you provide. This is the recommended choice for building your applications.
* **AWS CLI** — You can use the AWS CLI to access any Amazon Lex feature without having to write any code.
* **AWS Console** — The console is the easiest way to get started testing and using Amazon Lex

**Literature Review :**



In this section we will see what are the services used and understand them that how they are helpful to create the amazon-lex bot .

Amazon Lex has the capability to integrate with other AWS platform services and vice versa. For starters, Amazon Lex can be configured to discharge business sense into Lambda.

1. **AWS-LAMBDA :** Lambda functions can be used as code hooks for initialization , confirmation and for fulfillment conditions. It is a serveless compute service that helps to run the code in response to events that helps to automatically compute the resources for user .

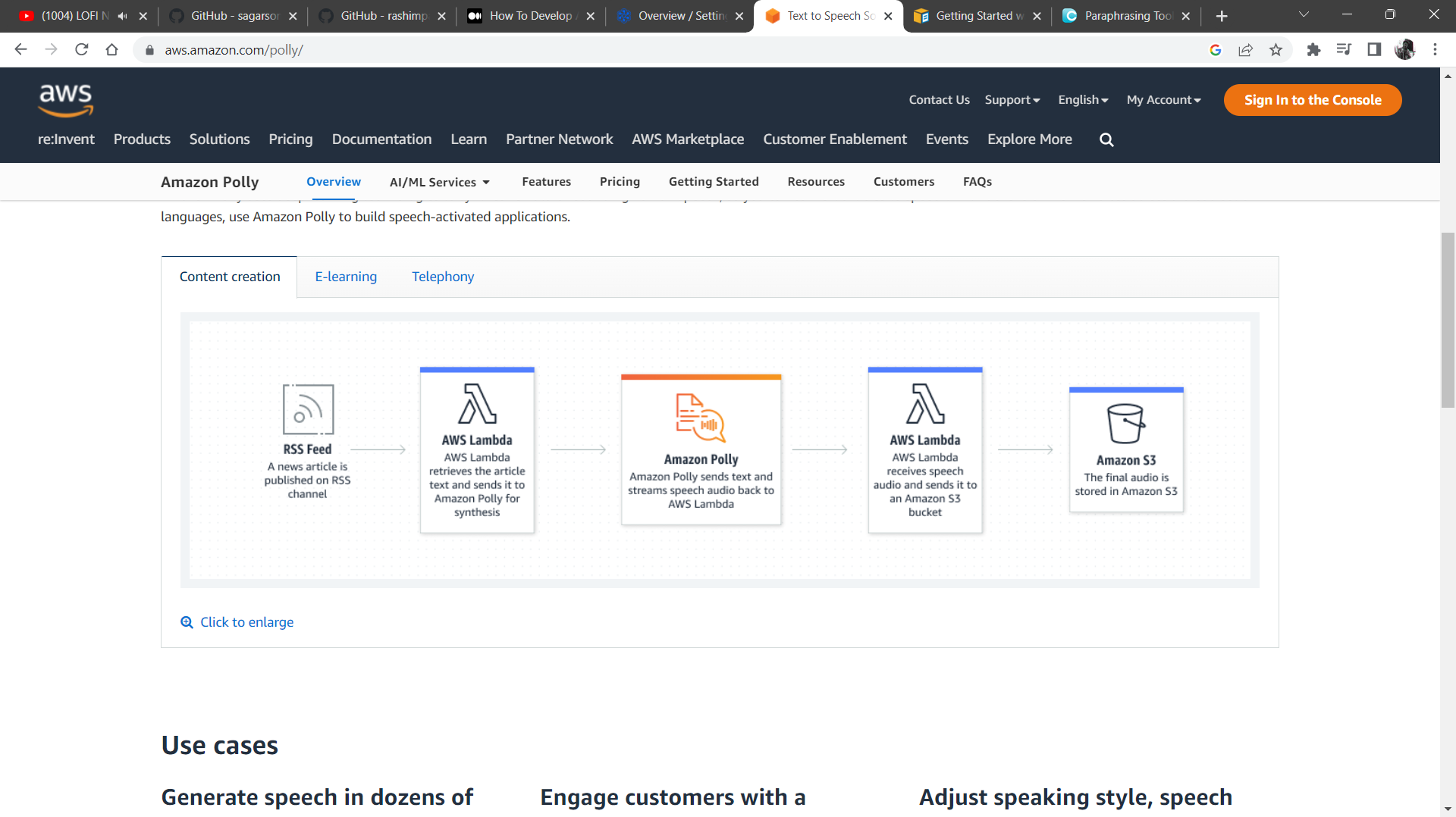
Input events used to be filled while creating a lambda function :

1. **CurrentIntent :** It provides the intents that is slot , slotDetails , confirmationstatus , name.
2. **AlternativeIntent :** Amazon Lex returns up to four indispensable intents, If you enable confidence scores. Each intent includes a score that indicates the position of confidence that Amazon Lex has that the intent is the correct intent grounded on the user's utterance.
3. **Bot :** About therequest that is processed .
   * + - **Name –** Name of the bot that further processed the request.
       - **Alias –** Alias of the bot Version that further processed the request.
       - **Version –** Version of the bot thatfurther processed the request.
4. **userId :** Value is provided by the client application , further it passes to the lambda function .
5. **InputTranscript :** If the input was text , then it will contain the text in its field. If the input is in audio stream/form then it will be extracted to the text form and then stored and recognize the slot values .
6. **invocationSource :** It is used to indicate why amazon lex is invoking the lambda function .
   * + - **DialogCodeHook :** Initialize the lambda function and to validate the user data input directly .
       - **FulfillmentCodeHook :** To fulfill the an intent it directly sets the value to lambda function .
7. **outputDialogMode :** For each user input, the customer sends the request to Amazon Lex using one of the runtime API operations, PostContent or PostText. It uses the request parameters to determine whether the response to the customer is text or voice, and sets this field consequently . The Lambda function can use this information to an correct message . For eg , if the customer expects a voice response, your Lambda function could return Speech Synthesis Markup Language( SSML) rather of text.
8. **messageVersion :** message that identifies the format of the event data going into the Lambda function and the expected format of the response from a Lambda function.
9. **SessionAttribute :** Application-specific session attributes that the client sends in the request. If you want Amazon Lex to include them in the response to the client, your Lambda function should send these back to Amazon Lex in the response.
10. **RequestAttribute :** Request-specific attributes that the client sends in the request. Use request attributes to pass information that doesn't need to persist for the entire session. If there are no request attributes, the value will be null.
11. **recentIntentSummaryView :** Information about the state of an intent. You can see information about the last three intents used. You can use this information to set values in the intent or to return to a previous intent.
12. **SentimentResponse :** The result of an Amazon Comprehend sentiment analysis of the last utterance. You can use this information to manage the conversation flow of your bot depending on the sentiment expressed by the user.
13. **KendraResponse :** The result of a query to an Amazon Kendra index. Only present in the input to a fulfillment code hook and only when the intent extends the AMAZON.KendraSearchIntent built-in intent.
14. **ActiveContexts :** One or more contexts that are active during this turn of a conversation with the user.
15. **AWS-COGNITO :** Amazon Cognito lets you fluently add user sign- up and authentication to your mobile and web apps. Amazon Cognito also enables you to authenticate users through an external identity provider and provides temporary security credentials to access the app's backend resources in AWS or any service behind Amazon API Gateway.

It is used to manage the permission of the users for the web-application , that needs to be set-up amazon cognito .

When creating a identity pool , amazon cognito provides us with the AWS IAM roles for authenticated and unauthenticated users .

1. **AWS-POLLY :**  Amazon Polly uses deep learning technologies to synthesize natural- sounding human speech, so you can convert text to speech. With dozens of naturalistic voices across a broad set of languages, use Amazon Polly to make speech- actuated operations. [Amazon](https://aws.amazon.com/lex/) lex now supports [Amazon polly](https://aws.amazon.com/polly/)  Neural Text-to-Speech (NTTS) voices for your bots, allowing your bots to respond to your users with richer, more expressive, and natural-sounding voices than standard Polly Text-to-Speech (TTS) voices. Polly NTTS voices deliver advanced improvements in speech quality through a new machine learning approach.

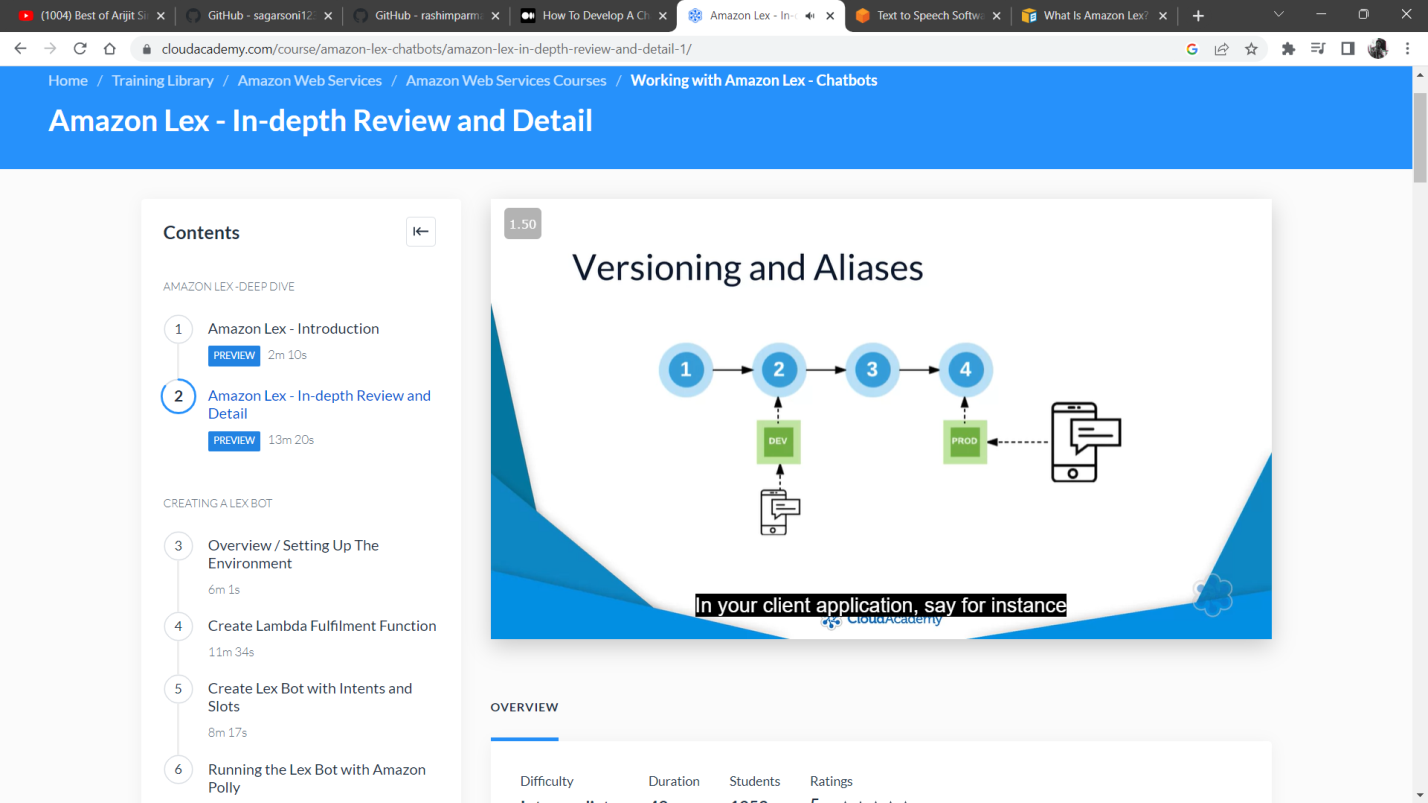
****

1. **AWS-MOBILE HUB :**  Finally, AWS Mobile Hub has templates that can be used to automatically provision Amazon Lex chatbot setup. AWS MobileHub is an admixture of different yet important AWS services for mobile app deployment and configuration. The tools included in Mobile Hub enable inventors to built, test, configure, and develop cloud - based mobile operations. These tools are put together as a package for users so that they can access it fluently through the console. The console allows inventors to elect their demanded tools and features for the mobile operations and fluently integrate them into the code . Not only that, Mobile Hub in AWS offers benefits to developers with redundant tools.
2. **Versioning and Aliases :** Amazon Lex provides versioning and alias capabilities. As a developer , you can work the versioning and alias capabilities to control the perpetration that your customer operations use. Each interpretation is a numbered snapshot of your chatbot configuration. Amazon Lex automatically tracks a interpretation for each iterative change that occurs for any resource within your chatbot configuration. This allows Amazon Lex to track and know what is being used within a particular interpretation of your chatbot that you're testing.

**Versioning** : When you interpretation an Amazon Lex resource you produce a snapshot of the resource so that you can use the resource as it was when the interpretation was made. Once you've created a interpretation it'll stay the same while you continue to work on your application .When you publish a resource, Amazon Lex makes a copy of the LATEST version interpretation and saves it as a numbered interpretation. The published interpretation can not be changed . Before you can publish a bot, you must point it to a numbered interpretation of any intent that it uses . However, Amazon Lex returns an HTTP 400 Bad Request exception, If you try to publish a new interpretation of a bot that uses the$ LATEST version interpretation of an intent. Before you can publish a numbered interpretation of the intent, you must point the intent to a numbered interpretation of any slot type that it uses. else you'll get an HTTP 400 Bad Request exception.

**Aliases** : An alias is a pointer to a specific interpretation of an Amazon Lex bot. Use an alias to allow customer operations to use a specific interpretation of the bot without taking the operation to track which interpretation that is. When you produce a alternate interpretation of the bot, you can modernize the alias to point to the new interpretation of the bot using the press or the **PutBot** operation. When you change the alias, all of your customer operations use the new version .However, you can roll back to the former interpretation by simply changing the alias to point to that interpretation, If there's a problem with the new interpretation .

Although you can test the$ LATEST interpretation of a bot in the press, we recommend that when you integrate a bot with your customer operation, you first publish a interpretation and produce an alias that points to that interpretation. Use the alias in your customer operation for the reasons explained in this section. When you modernize an alias, Amazon Lex will stay until the session downtime of all current sessions expires before it starts using the new interpretation.



1. **Migrating a Bot :** Amazon Lex V2 API uses an updated information armature that enables simplified resource versioning and support for multiple languages in a bot.

To use these new features, we need to resettle the bot. When the bot is to be resettled , Amazon Lex provides the following

Migration clones your custom intents and slot types to the Amazon Lex V2 bot. You can add multiple languages to the same Amazon Lex V2 bot. In Amazon Lex V1 you produce a separate bot for each language. You can resettle multiple Amazon Lex V1 bots, each using a different language, to one Amazon Lex V2 bot. Amazon Lex maps Amazon Lex V1 erected- in slot types and intents to Amazon Lex V2 erected- in slot types andintents.However, Amazon Lex returns a communication that tells you what to do next, If a erected- in can not be migrated.

The migration process does not resettle the following

* Aliases
* Amazon Kendra indicators
* AWS Lambda functions
* discussion log settings
* Messaging channels similar as Slack
* Tags

1. **Security in Amazon-lex** : Cloud security at AWS is the highest precedence. As an AWS client, you profit from a data center and network arch. that's erected to meet the conditions of the most security-sensitive associations.

Security is shared between the aws and the user .

**Security of the cloud**

**Security in the cloud**

* IAM for amazon lex
* Monitoring in amazon lex
* Data protection
* Infrastructure security
* Resilience in amazon lex

1. **The typical developer workflow involves the following sequenced way .**

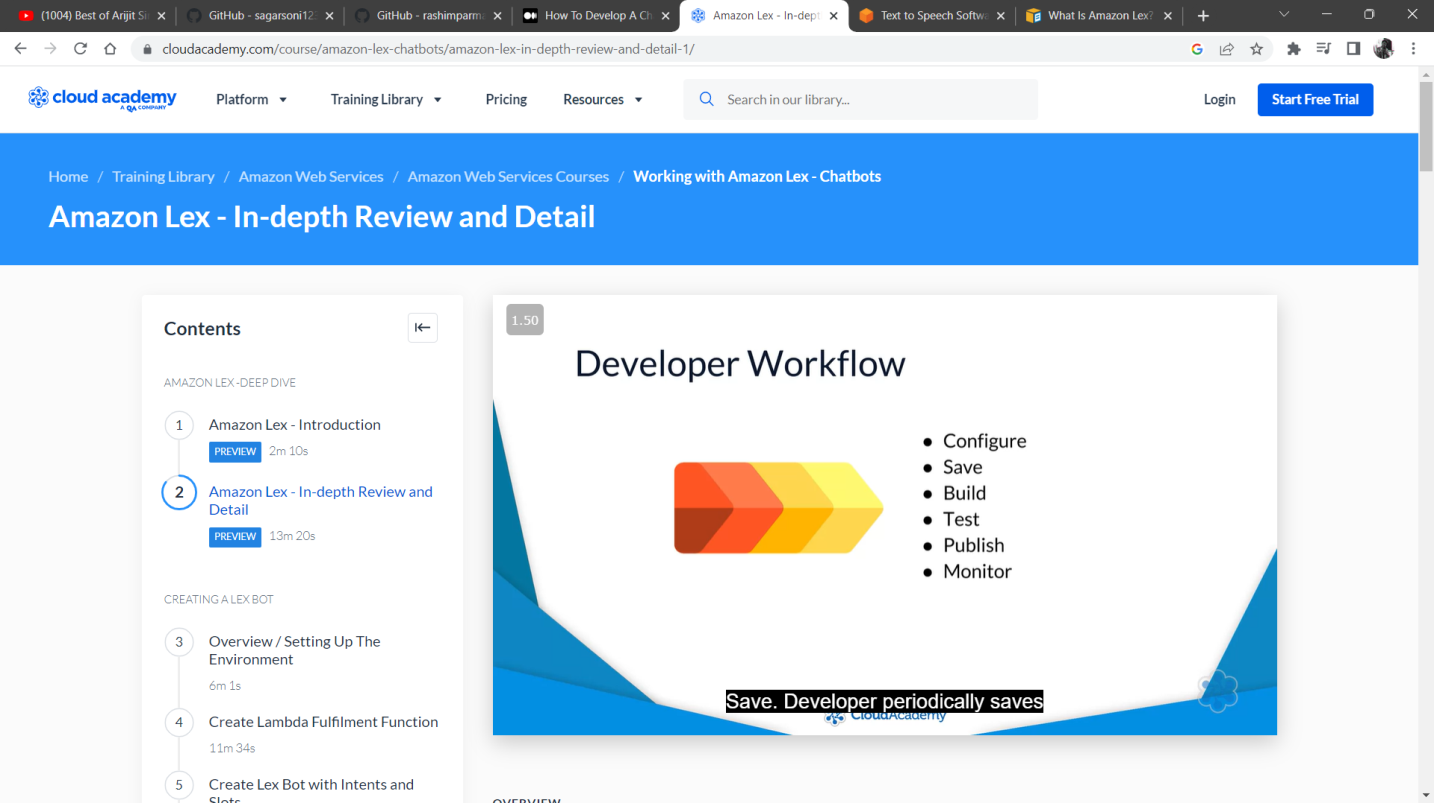
**CONFIGURE :** Developer declaratively configures their chatbot.

**SAVE :** Developer periodically saves the current chatbot configuration.

**BUILD :** Developer builds the current chatbot configuration. erecting a chatbot creates a testable interpretation of itself.

**TEST :** Developer performs conversational tests with the chatbot using the test interface within the Amazon Lex service press.

**PUBLISH :** Developer publishes the chatbot. This creates a interpretation of the chatbot, and for which can be assigned to an alias.

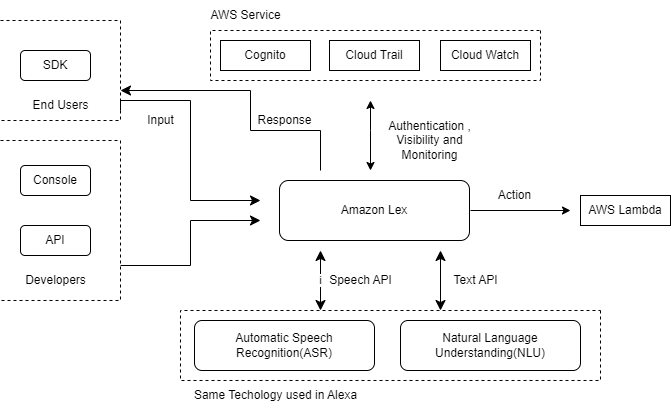


1. **DynamoDB :** It is a service provided by aws to fully manage the databse with NOSQL service . Which works with the value pair and the data provided by the amazon . It comprises of the 3 fundamental units known as TABLE , ATTRIBUTE , ITEM . It is very easy to use and works very well when if the data grows up .It requires only primary key , whitout scheme dynamodb is able to create its database table .

In Amazon lex , DynamoDB is used for storing the data which is the input from the user and then it will request to the lambda function and it will further processing the request and respond to the user’s request with an output .

1. **Amazon CloudWatch :** To track the health of your Amazon Lex bots, use Amazon CloudWatch. With CloudWatch, you can get criteria for individual Amazon Lex operations or for global Amazon Lex operations for your account. You can also set up CloudWatch reminders to be notified when one or further criteria exceeds a threshold that you define. For eg , you can cover the number of requests made to a bot over a particular time period, view the latency of successful requests, or raise an alarm when errors exceed a threshold.
2. **Cloud Trail:** CloudTrail is enabled on your AWS account when you produce the account. When supported event exertion occurs in Amazon Lex, that exertion is recorded in a CloudTrail event along with other AWS service events in Event history. You can view, search, and download recent events in your AWS account. A trail enables CloudTrail to deliver log lines to an Amazon Simple Storage Service bucket . By default , when you produce a trail in the console , the trail applies to all AWS Regions. The trail logs events from all Regions in the AWS partition and delivers the log lines to the S3 pail that you specify.

**FlowChart:**



**THANK YOU .!!**