AMAZON LEX

1)Describe Amazon Lex's architecture.

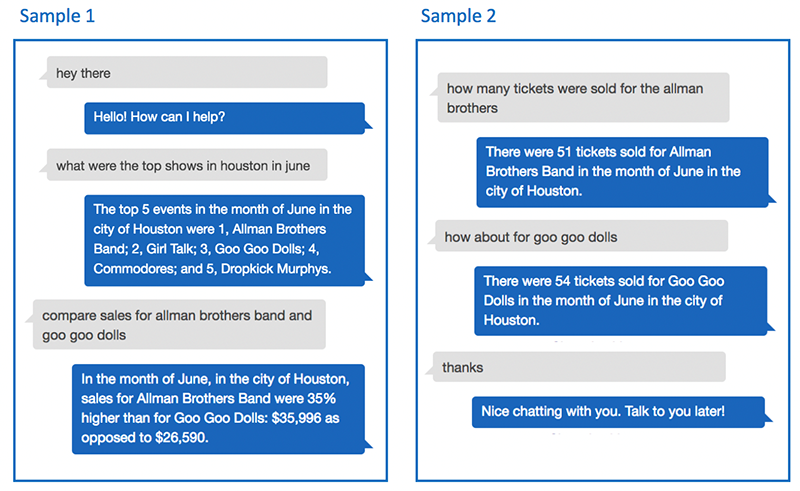
ANS:-Conversational interfaces are transforming the way people interact with software applications and services. They are untethering people from keyboards and smartphone gestures by replacing those interfaces with a more natural style of interaction: the spoken word. Increasingly, people are opting to interact with a bot when they need an answer to a question, to set a reminder, or to obtain a product or service.

With Amazon Lex, we can bring this same level of convenience to data. By allowing users to explore datasets by asking a series of questions, and maintaining a conversational context, we can provide a whole new experience and relationship with data.

This blog post shows you how to use Amazon Lex to implement a business intelligence (BI) chatbot, which we refer to as “BIBot,” although you can customize it to use a different name. BIBot can respond to user questions about data in a database, by converting the questions into backend database queries, and transforming the result sets into natural language responses. For example, the request “tell me the increase in inventory last month” could be translated to “select sum(item\_qty) from inventory where month(received\_date) = 10”.

BIBot has been integrated with a typical relational database intended for business intelligence and reporting applications. The sample database is the Amazon Redshift TICKIT database, which tracks sales activity for a fictional website where users buy and sell tickets online for music concerts and theater shows. The database is a star schema with two fact tables (sales, listings) and five dimension tables (events, dates, venues, categories, and users). See [Amazon Redshift » Sample Database](https://docs.aws.amazon.com/redshift/latest/dg/c_sampledb.html) for details.

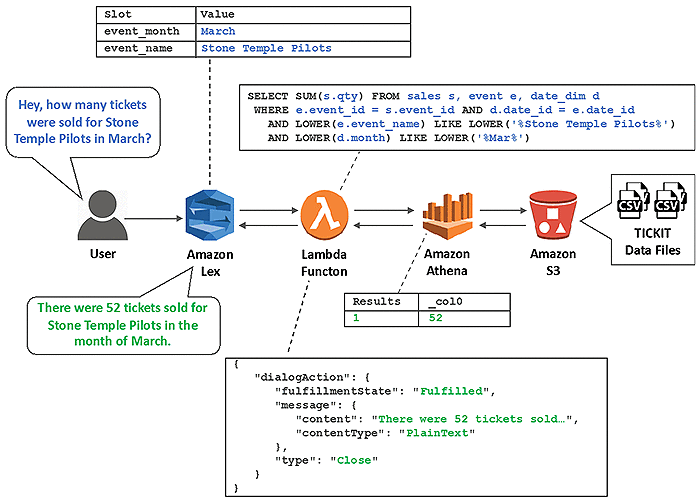
Here are some sample interactions with BIBot:



As you can see from these examples, BIBot is able to keep track of the context of your questions, by remembering that you asked about Houston in June, and that you asked how many tickets were sold. The conversation uses the “language” of the data, which in this case is ticket sales, cities, months, events, and so on. These are the facts and dimensions of the sample ticket sales database. If you adapt BIBot to use your reporting database, conversations with the bot will be in the language of your data.

**Architecture**

BIBot’s architecture is simple. A Lex bot directs each of the user’s questions to an intent, which parses the question into slots. The Amazon Lex bot then passes the intent and slot data to an AWS Lambda function, which uses the data to construct a SQL query, and execute it against an Amazon Athena database. Athena retrieves the query results from a set of CSV files stored in an Amazon S3 bucket, and returns the result set back to the Lambda function, which converts it into a natural language response.



Athena was used for simplicity and convenience, but this architecture will work with any SQL-based database, and can be adapted to other types of data sources, such as NoSQL databases.

2)What was the primary motivation for creating Amazon Lex?

Ans:- Amazon Lex enables you to build applications using a speech or text interface powered by the same technology that powers Amazon Alexa. Following are the typical steps you perform when working with Amazon Lex:

1. Create a bot and configure it with one or more intents that you want to support. Configure the bot so it understands the user's goal (intent), engages in conversation with the user to elicit information, and fulfills the user's intent.
2. Test the bot. You can use the test window client provided by the Amazon Lex console.
3. Publish a version and create an alias.
4. Deploy the bot. You can deploy the bot on platforms such as mobile applications or messaging platforms such as Facebook Messenger.

Before you get started, familiarize yourself with the following Amazon Lex core concepts and terminology:

* **Bot** – A bot performs automated tasks such as ordering a pizza, booking a hotel, ordering flowers, and so on. An Amazon Lex bot is powered by Automatic Speech Recognition (ASR) and Natural Language Understanding (NLU) capabilities. Each bot must have a unique name within your account.

Amazon Lex bots can understand user input provided with text or speech and converse in natural language. You can create Lambda functions and add them as code hooks in your intent configuration to perform user data validation and fulfillment tasks.

* **Intent** – An intent represents an action that the user wants to perform. You create a bot to support one or more related intents. For example, you might create a bot that orders pizza and drinks. For each intent, you provide the following required information:

* + **Intent name**– A descriptive name for the intent. For example, **OrderPizza**. Intent names must be unique within your account.
  + **Sample utterances** – How a user might convey the intent. For example, a user might say "Can I order a pizza please" or "I want to order a pizza".
  + **How to fulfill the intent** – How you want to fulfill the intent after the user provides the necessary information (for example, place order with a local pizza shop). We recommend that you create a Lambda function to fulfill the intent.

You can optionally configure the intent so Amazon Lex simply returns the information back to the client application to do the necessary fulfillment.

In addition to custom intents such as ordering a pizza, Amazon Lex also provides built-in intents to quickly set up your bot. For more information, see [Built-in Intents and Slot Types](https://docs.aws.amazon.com/lex/latest/dg/howitworks-builtins.html).

* **Slot** – An intent can require zero or more slots or parameters. You add slots as part of the intent configuration. At runtime, Amazon Lex prompts the user for specific slot values. The user must provide values for all required slots before Amazon Lex can fulfill the intent.

For example, the OrderPizza intent requires slots such as pizza size, crust type, and number of pizzas. In the intent configuration, you add these slots. For each slot, you provide slot type and a prompt for Amazon Lex to send to the client to elicit data from the user. A user can reply with a slot value that includes additional words, such as "large pizza please" or "let's stick with small." Amazon Lex can still understand the intended slot value.

* **Slot type** – Each slot has a type. You can create your custom slot types or use built-in slot types. Each slot type must have a unique name within your account. For example, you might create and use the following slot types for the OrderPizza intent:

* + Size – With enumeration values Small, Medium, and Large.
  + Crust – With enumeration values Thick and Thin.

Amazon Lex also provides built-in slot types. For example, AMAZON.NUMBER is a built-in slot type that you can use for the number of pizzas ordered. For more information, see [Built-in Intents and Slot Types](https://docs.aws.amazon.com/lex/latest/dg/howitworks-builtins.html).

For a list of AWS Regions where Amazon Lex is available, see [AWS Regions and Endpoints](https://docs.aws.amazon.com/general/latest/gr/rande.html#lex_region) in the Amazon Web Services General Reference.

The following topics provide additional information. We recommend that you review them in order and then explore the [Getting Started with Amazon Lex](https://docs.aws.amazon.com/lex/latest/dg/getting-started.html) exercises.

3) Who does Amazon Lex get evaluated by?

3) Who does Amazon Lex get evaluated by?

Ans:- Multiple pieces of information are often required to complete a task or to process a query. For example, when talking to an insurance agent, a caller might ask, “Can you provide me quotes for home, auto, and boat?” The agent recognizes this as a list of policy types before continuing with the conversation. Automation of such conversations requires that bots recognize and process multiple values as well. Previously, developers had to use multiple slots to capture each value in the list. In addition, they had to write code to compile these values into a list. Now, we announce support for multi-valued slots in [Amazon Lex](https://aws.amazon.com/lex/). Developers can use the multi-valued slot to directly capture multiple values contained in a single response. Amazon Lex automatically maps the list of values to the multi-valued slot. Additional processing isn’t required, and the bot design is simplified.

This post shows how you can use multi-valued slots to capture user input that contains a list of values.

## Test the bot with a multi-valued slot

You can test the bot on the Amazon Lex console with both text and speech inputs.

You can extend the design to provide dynamic responses to information requested by the customer. For example, you can provide calculated quotes for life, home, and auto insurance. These quotes may be available in a third-party system, which you can look up using an [AWS Lambda](https://aws.amazon.com/lambda) function. For more information, see [Using AWS Lambda with Amazon Lex](https://docs.aws.amazon.com/lambda/latest/dg/services-lex.html). The following screenshot shows a sample Lambda function code snippet.

You can also enable the user to take action from the bot response by providing a URL to complete the next steps (“Click this link to process your life insurance quote”).

You can now supply multiple values for a slot with higher accuracy for five or fewer values, through speech or text. Any values not contained in the slot type (such as medical, home, auto, and boat) are dropped. For more information about parsing API responses, see [Slot](https://docs.aws.amazon.com/lexv2/latest/dg/API_runtime_Slot.html).

## Conclusion

Multi-value slots enable you to automate conversations in the customer service domain. With the new Amazon Lex multi-valued slots feature, you can easily capture a list of values provided by a user without the need for additional logic. For more information about incorporating multi-valued slots into your bots, see the [Amazon Lex documentation](https://docs.aws.amazon.com/lex/).

4) What is AWS Lambda and what is its purpose?

Ans:- AWS Graviton processors are custom built by Amazon Web Services using 64-bit Arm Neoverse cores to deliver the best price performance for your cloud workloads running in Amazon EC2. Amazon EC2 provides the broadest and deepest portfolio of compute instances, including many that are powered by latest-generation Intel and AMD processors. AWS Graviton processors add even more choice to help customers optimize performance and cost for their workloads.

##### **Best Price Performance for a Broad Spectrum of Workloads**

AWS Graviton2-based general purpose burstable (T4g), general purpose (M6g), compute-optimized (C6g), and memory-optimized (R6g) EC2 instances and their variants with NVMe-based SSD storage deliver up to 40% better price performance over comparable current generation x86-based instances1 for a broad spectrum of workloads.

##### **Extensive Ecosystem Support**

AWS Graviton2 processors, based on the 64-bit Arm architecture, are supported by popular Linux operating systems along with many popular applications and services from AWS and Independent Software Vendors.

##### **Enhanced Security for Cloud Applications**

AWS Graviton2 processors feature key capabilities that enable developers to run cloud native applications securely, and at scale, including always-on 256-bit DRAM encryption and 50% faster per core encryption performance compared to first-generation AWS Graviton.

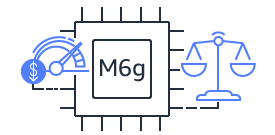
###### **EC2 Instances Powered by AWS Graviton2**

Amazon EC2 provides a wide selection of instance types optimized to fit different use cases. Instance types comprise varying combinations of CPU, memory, storage, and networking capacity and give you the flexibility to choose the appropriate mix of resources for your applications. Each instance type includes one or more instance sizes, allowing you to scale your resources to the requirements of your target workload.

Instances

Description

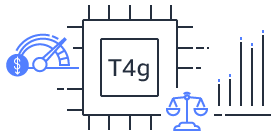
Use Cases



##### **M6g**

Best price performance for general purpose workloads with balanced compute, memory, and networking.

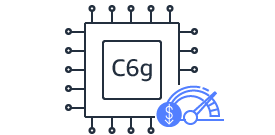
General-purpose workloads such as application servers, mid-size data stores, microservices, and cluster computing.



##### **T4g**

Best price performance for burstable general purpose workloads.

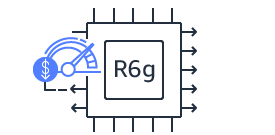
Broad range of burstable general purpose workloads such as large scale micro-services, small and medium databases, virtual desktops, and business-critical applications.



##### **C6g**

Best price performance for compute-intensive workloads.

Compute-intensive applications such as high performance computing, video encoding, gaming, and CPU-based machine learning inference acceleration.



##### **R6g**

Best price performance for workloads that process large data sets in memory.

Memory-intensive workloads such as open-source databases (MySQL, MariaDB, and PostgreSQL), or in-memory caches (Redis, KeyDB, Memcached).

Local NVMe-based SSD storage options are also available in general purpose (M6gd), compute-optimized (C6gd), and memory-optimized (R6gd) instances. Additionally, compute-optimized C6gn instances with 100 Gbps networking and support for Elastic Fabric adapter (EFA) are also available.

#### **Get Started**

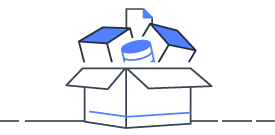
Until December 31st, 2021, all new and existing AWS customers can try the t4g.micro instances free for up to 750 hours per month. Refer to the [FAQ](https://aws.amazon.com/ec2/faqs/#t4g-instances) for additional details.

Creating an AWS account is free and gives you immediate access to the AWS Free Tier.



#### **AWS Free Tier**

The AWS Free Tier offers users an opportunity to explore products for free, with offers including products that are always free, free for 12 months, and short-term free trials.



5) What do you mean by EC2 instance?

Ans:- An Amazon EC2 instance is a virtual server in Amazon's Elastic Compute Cloud (EC2) for running applications on the Amazon Web Services (AWS) infrastructure. AWS is a comprehensive, evolving cloud computing platform; EC2 is a service that enables business subscribers to run application programs in the computing environment. It can serve as a practically unlimited set of virtual machines (VMs).

Amazon provides various types of instances with different configurations of CPU, memory, storage and networking resources to suit user needs. Each type is available in various sizes to address specific workload requirements.

Instances are created from Amazon Machine Images ([AMI](https://www.techtarget.com/searchaws/definition/Amazon-Machine-Image-AMI)). The machine images are like templates. They are configured with an operating system (OS) and other software, which determine the user's operating environment. Users can select an AMI provided by AWS, the user community or through the [AWS Marketplace](https://www.techtarget.com/searchaws/definition/AWS-Marketplace). Users also can create their own AMIs and share them.

### EC2 instance types

Instance types are grouped into families based on target application profiles. These groups include the following:

* **General purpose.**A general purpose instance is a VM that is designed to handle a variety of workloads. General purpose instances are optimized to have a high number of CPU cores, on-demand storage and memory. Some common use cases for general purpose instances include web server hosting and software development and testing.
* **Compute optimized.**Compute optimized instances are used to run big data applications that require large amounts of processing power and memory on the AWS cloud. These instances are designed and optimized for running computational and data-intensive applications that require fast network performance, extensive availability and high input/output (I/O) operations per second ([IOPS](https://www.techtarget.com/searchstorage/definition/IOPS-input-output-operations-per-second)). Examples of types of applications includes scientific and financial modeling and simulation, machine learning, enterprise data warehousing and business intelligence.
* **Graphics processing unit (GPU).**These instances provide a way to run graphics-intensive applications faster than with the standard EC2 instances. Systems that rely on [GPUs](https://www.techtarget.com/searchvirtualdesktop/definition/GPU-graphics-processing-unit) include gaming and design work. For example, Linux distributions often take advantage of GPUs for rendering graphical user interfaces, improving compression speeds and speeding up database queries.
* **Memory optimized.**Memory optimized instances use a high-speed, solid-state drive to provide ultra-fast access to data and deliver high performance. These instances are ideal for applications that require more memory and less CPU power, including open source databases, real-time big data analytics and in-memory caches.
* **Storage optimized.**Storage optimized instances are ideal for applications that require high I/O performance, such as [NoSQL](https://www.techtarget.com/searchdatamanagement/definition/NoSQL-Not-Only-SQL) databases that store and retrieve data in real time. They're also well suited for memory-intensive applications such as data processing, data warehousing, analytics workloads and log processing**.**
* **Micro.**A micro instance is meant for applications with low throughput. The micro instance type can serve as a small database server, as a platform for software testing or as a web server that does not require high transaction rates.

offers various pricing options for Amazon EC2 instances.

### Amazon EC2 instance features

Many EC2 instance features are customizable, including the storage, number of virtual processors and memory available to the instance, OS and the AMI on which the instance is based. The following are Amazon EC2 instance features:

* **Operating system.** EC2 supports many OSes, including Linux, Microsoft Windows Server, CentOS and Debian.
* **Persistent storage.** Amazon's Elastic Block Storage ([EBS](https://www.techtarget.com/searchaws/definition/Amazon-EBS-Amazon-Elastic-Block-Store)) service enables block-level storage volumes to be attached to EC2 instances and be used as hard drives. With EBS, it is possible to increase or decrease the amount of storage available to an EC2 instance and attach EBS volumes to more than one instance at the same time.
* **Elastic IP addresses.** Amazon's Elastic IP service lets IP addresses be associated with an instance. Elastic IP addresses can be moved from instance to instance without requiring a network administrator's help. This makes them ideal for use in [failover clusters](https://www.techtarget.com/searchwindowsserver/definition/failover-cluster), for load balancing, or for other purposes where there are multiple servers running the same service.
* **Amazon CloudWatch.** This web service allows for the monitoring of AWS cloud services and the applications deployed on AWS. [CloudWatch](https://www.techtarget.com/searchaws/definition/CloudWatch) can be used to collect, store and analyze historical and real-time performance data. It can also proactively monitor applications, improve resource use, optimize costs and scale up or down based on changing workloads.
* **Automated scaling.** Amazon [EC2 Auto Scaling](https://www.techtarget.com/searchaws/definition/AWS-Auto-Scaling) automatically adds or removes capacity from Amazon EC2 virtual servers in response to application demand. Auto Scaling provides more capacity to handle temporary increases in traffic during a product launch or to increase or decrease capacity based on whether use is above or below certain thresholds.
* **Bare-metal instances.** These [virtual server instances](https://www.techtarget.com/searchaws/tip/AWS-bare-metal-instances-target-a-niche-market-for-now) consist of the hardware resources, such as a processor, storage and network. They are not virtualized and do not run an OS, reducing their memory footprint, providing extra security and increasing their processing power.
* **Amazon EC2 Fleet.** This service enables the deployment and management of instances as a single virtual server. The Fleet service makes it possible to launch, stop and terminate EC2 instances across EC2 instance types with one action. [Amazon EC2 Fleet](https://www.techtarget.com/searchaws/news/252452751/EC2-Fleet-Auto-Scaling-targets-diverse-workloads) also provides programmatic access to fleet operations using an API. Fleet management can be integrated into existing management tools. With EC2 Fleet, policies can be scaled to automatically adjust the size of a fleet to match the workload.
* **Pause and resume instances.**EC2 instances can be paused and resumed from the same state later on. For example, if an application uses too many resources, it can be paused without incurring charges for instance usage.